On the VLSI Implementation of Low Complexity K-Best MIMO Decoders

Sudip Mondal and Khaled N Salama
Electrical, Computers and Systems Engineering
Rensselaer Polytechnic Institute

Ahmed Eltawil
Electrical Engineering and Computer Science
University of California–Irvine

***Abstract*—The VLSI Implementation of Maximum Likelihood (ML) detection for higher order Multiple Input Multiple Output (MIMO) systems continues to be a major challenge. Battery driven handheld devices impose strict area and power constraints while demanding guaranteed performance over a wide range of operating conditions. This paper presents a modified, low complexity K-best detector for a 4×4, 64 QAM MIMO system , which uses a modified path extension algorithm to bring down the computational complexity by more than 50%. It also exploits the structure of a QAM constellation to achieve low power operation by reducing the number of costly multiplication operations while computing the path metrics. The two new approaches, combined with a resource shared architecture leads to the implementation of a very efficient FPGA based system, where we verify the performance of the algorithm.**

I. Introduction

Spatially multiplexed Multiple Input Multiple Output (MIMO) wireless communication systems enables devices to achieve near Shannon channel capacity and are very attractive for high data rates [1]. Several VLSI Implementations of systems achieving acceptable Bit Error Rate(BER) have been reported. A high throughput MIMO detector for 16 QAM has been reported in [6] and a soft output detector for 64 PSK system has been reported [2]. Both employ the K-best algorithm to achieve a near constant throughput. However, the throughput in [6] degrades heavily with increasing K and the implementation in [2] suffers from high power consumption and large area. The VLSI implementation of a ML detector for a 4×4 , 16 QAM system, reported in [3], is relatively power and area efficient but suffers from non-uniform throughput.

In this paper, we present a low power VLSI architecture for a 4×4, 64 QAM system, and its implementation on FPGA. The reduction in power was achieved by using two important modifications: i) using approximate path metrics throughout the system rather than exact path metrics and ii) an efficient path extension algorithm, which achieves minimum number of path extensions. These two techniques, combined with Reconfigurable VLSI architecture and resource sharing, lead to a system which is very efficient in terms of resource utilization, while maintaining the same throughput as conventional systems. In the next section we briefly review the K-best MIMO detection algorithm, and the related issues. Section III discusses the VLSI Architecture, and the techniques of approximate path metric representation through Quantization and the Winner path extension technique to achieve fast extension. In Section IV, we discuss the detection performance of the algorithm, and its FPGA Implementation.

II. Background

A. MIMO System Model

The MIMO system with M transmit antennas and N receive antennas can be modeled as:

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

where s is the $N \times 1$ transmitted symbol vector, H is the $M \times N$ channel matrix, n is the $M \times 1$ iid Gaussian noise, and y is the $M \times 1$ received symbol vector. The task of the ML detector is to find:

$$\min_{s \epsilon \Omega} \|(y - H \cdot s)\|^2 \tag{2}$$

where Ω represents the tree of all possible transmitted symbol vectors as shown in Fig. 1. The root of the tree is the zero-forcing solution for the transmitted vector, given by [2]:

$$\hat{s} = (H * H)^{-1} H^* y \tag{3}$$

Hence, the algorithm is essentially a shortest path algorithm in a tree of depth N, where the path metrics are given by:

$$\|(y - H \cdot s)\|^2 = (s - \hat{s})^* L^* L (s - \hat{s}) + y^* (I - H(H*H)^{-1} H^*) y \tag{4}$$

where $L^*L = H^*H$, L is lower triangular and $(.)^*$ represents the conjugate transpose operation. The computation of L is done by the Preprocessing unit, which also does the channel estimation. A corresponding soft detector generates reliability values for the bits, as explained in [2]. As exhaustive search over a candidate space of size Q^N is impractical, the Sphere Decoding technique tries to search a subset of Ω , given by:

$$\|(y - H \cdot s)\|^2 < r^2 \tag{5}$$

Thus, by constraining the search to only the candidates lying inside a fixed hyper sphere of radius r (called the error bound) ,it accelerates the performance. Any path exceeding the error bound is pruned. This results in a recursive tree search, and non-constant throughput.A K-Best Sphere Decoder is one which keeps the K best paths inside the error bound at all times. This has been discussed in [4]. An alternate approach for constant throughput and near optimal performance is to traverse the tree in a breadth first manner, maintaining the K

branches with minimum path metric at each level. The two approaches are illustrated in Fig. 1. The K best approach is favored due to its constant throughput property, and ease of pipelining, and is the focus of this work. The next section discusses the VLSI architecture of the K best MIMO detector system, using approximate path metrics and efficient extension techniques.

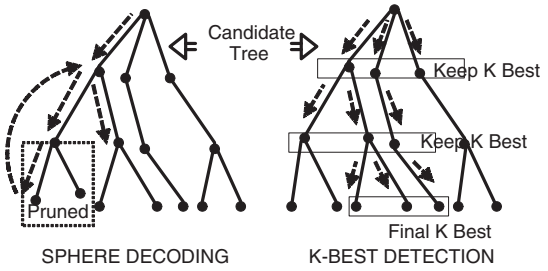

Fig. 1. Sphere Decoding vs K-Best Detection

III. VLSI Architecture for the K-Best Detector

As already mentioned, the MIMO detection problem is modeled as a shortest path problem in a tree. The K best approach is very attractive as the system can be pipelined, with each pipelined stage representing one level of the tree. The path metric computation is distributed among the N stages as:

$$(s-\hat{s})^* L^* L(s-\hat{s}) = \sum_{i=1}^{N}(|\sum_{j=1}^{i} l_{ij}(s_j - \hat{s_j})|^2) = \sum_{i}^{N} \Lambda_i^j \quad (6)$$

Accepting from the $i-1$th stage the K paths extended till level $i-1$ and having path metrics $\Gamma^2_{i-1,1}$,$\Gamma^2_{i-1,2}$,..,$\Gamma^2_{i-1,K}$ the ith stage needs to determine the K paths with metrics:

$$\Gamma_i^2 = \Gamma_{i-1}^2 + |l_{ii}^2||s_i - (\hat{s_i} - \sum_{j=1}^{i-1} \frac{l_{ij}}{l_{ii}}(s_j - \hat{s_j}))|^2 \quad (7)$$

such that we have the K minimum metrics $\Gamma^2_{i,1}$,$\Gamma^2_{i,2}$,..,$\Gamma^2_{i,K}$. This essentially involves the two tasks of:

1) Computing the *Center* $c_i = \hat{s}_i - \sum_{j=1}^{i-1} \frac{l_{ij}}{l_{ii}}(s_j - \hat{s_j})$ and
2) Computing $|l_{ii}|^2(s_i - c_i)^2$

where s_i denotes the tree node chosen at level i. This is carried out for all the K paths, for different choices of s_i. Finally the K paths resulting in minimum values of Γ_{i+1} given the K values of Γ_i, are retained and considered for the next level. The complete architecture of the system is shown in Fig. 2. 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. These two blocks constitute the basic detector cell. Each detector cell have 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 $M2$. The new path is selected by the *Minimum Finder* (MF) block. Simultaneously, a request is sent to the CC to compute the center corresponding to the new path, and the PM extends the center for the new path to its next nearest symbol. 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. The absence of data transfer among blocks, reduces power consumption as well as saving data transfer time, leading to enhanced throughput. Usually, multiple detector cells are

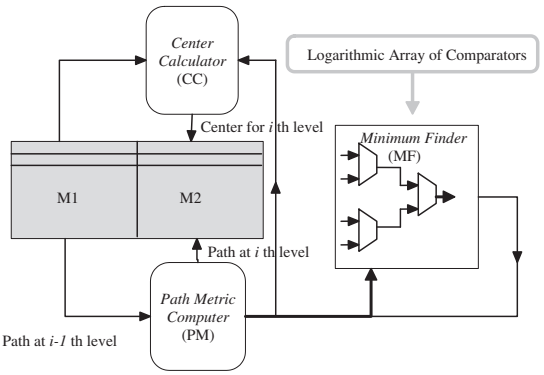

Fig. 2. The VLSI System Architecture

employed on a chip to achieve the required throughput. Each detector cell processes one received symbol. However, each detector cell processes a different tree level at any given cycle. This is used for handling the varying multiplication load to the CC of the detectors. As seen from the equation (6), different tree levels present different computational loads to the CC, due to the summation within the modulus operation varying in length with varying levels. For the ith level, the CC has to carry out $i-1$ multiply and add operations. Due to the constant throughput requirements it is necessary to allocate more resources to the CC for higher levels of the tree (i.e. for larger values of i), than for lower levels. If the maximum possible number of multipliers are allocated to each block, it results in half the multipliers sitting idle all the time. Noting that at any stage, the total number of levels processed by all the blocks collectively is fixed,we employ a bank of multipliers containing just enough multipliers to process all six levels at the same time. This bank implements the multiplications required for equation (6). Thus we require 24 multipliers, all of which are always active. The multiplier outputs are channelized to the proper adders using multiplexers. This way, the same block can be configured to process different tree levels, yet avoiding idle resources.

After the Center Calculation, the next major task of the detector cell is the Path Metric Computation. Most of the power consumed by the system is due to the large number of path metrics computed, as each computation involves three multiplications. We use an approximate Path Metric Computer unit as shown in Fig. 3. The use of approximate path metrics

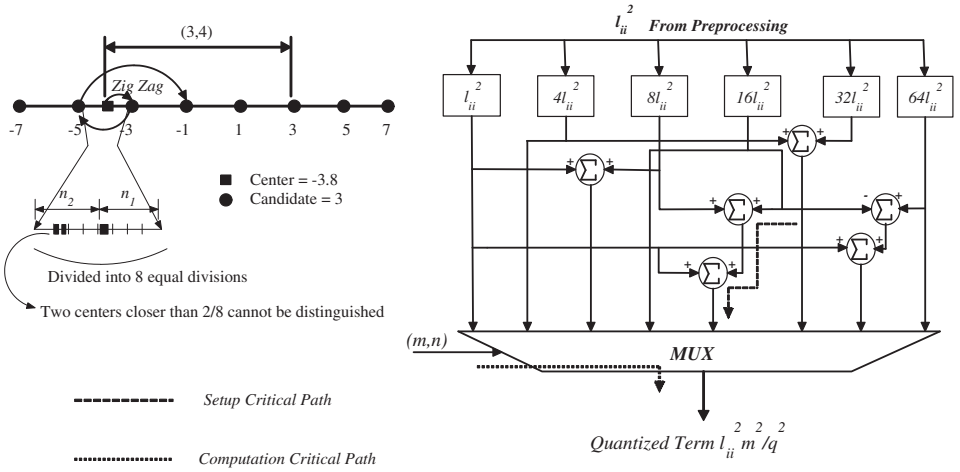

Fig. 3. Ordered Pair Representation of Quantized Distance and Path Metric Computer

instead of exact path metrics, brings down the computational by an order of magnitude. This is illustrated in the next section, which describes the Quantization Based Path metric Computer.

A. Path Metric Computation for QAM Constellation

With real decomposition of the channel matrix, the set of symbols to be chosen at each level for a 64 QAM constellation becomes limited to the odd integers $\{-7, -5, .., 5, 7\}$. In general, for a Q-QAM constellation, the symbols at each level can be represented by p where $p \epsilon \{-\sqrt{Q}+1, -\sqrt{Q}+3, .., \sqrt{Q}-3, \sqrt{Q}-1\}$. The symbols are considered in the $SchnoorEuchener$ (SE) ordering, to enure that a nearer symbol is considered before a farther symbol. For a QAM constellation this corresponds to a zigzag search, as shown in Fig. 3, and explained in [1]. Path metrics are easily computable, when expressed as an ordered pair (m, n), with m representing the integral number of symbol distance and n represents the fractional part. To achieve this, we divide the region between two consecutive symbols into q divisions. Noting that the distance between two consecutive symbols is 2 units, the distance between the center and the symbol is essentially approximated by $2m + \frac{2n}{q}$. This is illustrated in Fig. 3 where q is assumed to be 8. With this representation, the incremental metric from α to c_i becomes:

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

where β is $1/q$.

Thus, the total number of possible path metric increments is equal to the total number of representations (m, n). Equation(8) can be evaluated for each channel coefficient, for the possible combinations of m and n and these values can be reused for all the Path Metric Computations. This is illustrated in Fig. 3. Straightforward designs based on cascaded multipliers are implemented in [3] and suggested in [2], having zero setup time, and delay of two multipliers. It should be noted that the setup time does not affect the computation delay in the QPM. As seen in Fig. 3, n can have only two possible values for a given center, n_1 and n_2, where $n_2 = 8 - n_1$. Once computed for a given center, these values are reused and not recomputed. If the absolute difference $abs(c_j - \alpha)$ is represented using k bits, with l bits for the integral part, then m is the most significant $l - 1$ bits. Thus, the ordered pair representation does not introduce extra delay in the process. The critical path of the computation is based on the multiplexer delays and hence significantly less than multiplier delays. We carried out the ASIC implementation of a conventional multiplier based path metric computer and a quantization based version to estimate the performance gains. As seen from Table I, the area and power consumption is reduced by more than a factor of 10.

TABLE I
CONVENTIONAL AND QUANTIZED METRIC COMPARISON

	Conventional design	Quantization Based design	Improvemnt
Area (mm^2)	0.026	0.002	$10x$
Power (mW)	4.472	0.093	$48x$
FLOPS	2048	328	$\sim 6.25x$

B. Winner Path Extension

It is understood that out of all the extended paths (i.e. paths whose metrics are computed), only the K paths with the minimum metrics are of interest, while others are unnecessary. If $C(j, k)$ is the kth closest symbol to the jth center, then $C(j, k)$ cannot be included among the K-Best paths, if $C(j, k-1)$ is also not included. Hence, instead of extending all paths and sorting among them, as reported in [2],[5] and [6], we propose to compute the metric for $C(j, k)$ if and only if $C(j, k-1)$ has already been included. In the first step, all the K centers from the previous level are extended to their closest symbol. Then onwards, at each step, only the path with the minimum metric in the previous step, is extended to its next symbol. Since at each one of the K steps, only the path with the lowest metric is selected and extended, we ensure that we select all the K-Best paths and no more. Thus the number of

path extensions required is $2K-1$. These consists of all the K paths extended for the nearest candidates for the K centers, and the $K-1$ paths extended subsequently. It is important to note that this reduces the number of path computations from $K\sqrt{Q}$ to $2K-1$, almost a reduction of 75% for a 64 QAM system. In the next section we discuss the detection performance and the FPGA Implementation of the system.

IV. Detection Performance and VLSI Implementation

The system performance has been validated using C and MATLAB for for a 4×4, MIMO system with 64 QAM constellation using uncoded symbols. The FPGA and ASIC implementations were later tested against the fixed point C testbench. The use of quantization resulted in errors in each stage which accumulated at the leaf nodes. Hence, simply choosing the best leaf node resulted in around 0.4dB loss (at 17dB) for $K = 8$ and 1dB for $K = 64$. The *Explicit Path Metric computer* (EPM) was used at the last stage to recompute the path metrics without quantization. Thus, *only at the last stage* we compute the unquantized path metrics of the K retained paths and choose the best among them as the hard output. The performance is shown in Fig. 4. The FER vs SNR curves for quantization based and unquantized systems coincide with each other for both $K = 8$ and $K = 64$, showing that the performance loss is negligible.

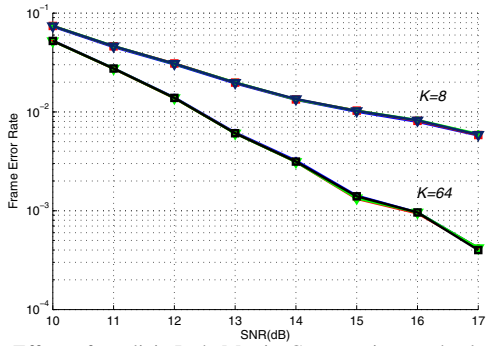

Fig. 4. Effect of explicit Path Metric Computation at the leaf nodes for $K = 8, 64$

In order to test the design, we carried out the FPGA implementation. An FPGA device of the Virtex II Pro family from Xilinx (XCV2P30) was used for the purpose. The maximum achievable clock frequency on the FPGA board was corresponding to a time period of 5.918ns which was the combined delay of a memory read and one multiplication in the EPM block. The system was clocked at $100Mhz$. The output was sent back to the PC, where it was checked against C testbench output. The resource utilization summary is presented in Table II. All LUTs are 4 inputs. It can be seen from the Table, that more than 70% of the resources are shared among the six detector cells used for the implementation. The sharing required the use of an *InterCONnect* (ICON) block, which presented the resources to appropriate detectors at appropriate cycles. It can be seen that the ICON consumes

TABLE II
FPGA Resource Utilization Report

	SLICES	FLIP FLOPS	LUTs	DSP	SHARED
AVAILABLE	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	

around 38% of slices, 37% of flip flops and almost 42% of LUTs. However, it is still advantageous, as it eliminates the replication of the costly MF and EPM blocks. The FPGA implementation requires 0.0225MBytes of RAM against an available RAM of 8MB.

V. Conclusion

A high throughput algorithm has been proposed and implemented for K-best MIMO detection. The quantization scheme is proposed for path metric computations, and it is shown to have negligible performance degradation with explicit path computations in the final stage. However it reduces the number of operations to nearly one sixth of conventional algorithms, and hence results in a large power savings. The Winner Path Extension technique minimizes path metric computations and is optimized for any sorting scheme, and is also independent of constellation size. The overall complexity is improved by a factor of 50% over those reported recently in the literature. Performance is verified using FPGA.

VI. Acknowledgement

We would like to thank the Center for Automation Technologies and Systems (CATS), who has supported this work in part through a block grant from the New York State Foundation for Science, Technology and Innovation (NYSTAR). We would also like to thank Prof. Ender Ayanoglu and Luay Azzam at the University of California, Irvine and Prof. Tong Zhang at Rensselaer Polytechnic Institute, New York, for their valuable suggestions.

References

[1] B.M. Hochwald and S. T. Brink, "Achieving near-capacity on a multiple-antenna channel," IEEE Transactions on Communications, pp. 389-399, March 2003.

[2] S. Chen, F. Sun and T. Zhang, "Nonlinear soft-output signal detector design and implementation for MIMO communication system with high spectral efficiency," IEEE Custom Integrated Circuits Conference, pp. 321-324, September 2006.

[3] A. Burg et al., "VLSI implementation of MIMO detection using the sphere decoding algorithm," IEEE Journal of Solid-State Circuits, pp. 1566-1577, July 2005.

[4] S. Baro, J. Hagenauer and M. Witzke "Iterative detection of MIMO transmission using a list sequential (LISS) detector,", IEEE International Conference on Communications, pp. 2653-2657, May 2003.

[5] Z. Guo and P. Nilsson, "Algorithm and implementation of the k-best sphere decoding for mimo detection," IEEE Journal on Selected Areas in Communication, pp. 491-503, Aug. 1995.

[6] A. Burg et. al., "K-Best MIMO detection VLSI architectutres achieving upto 424 Mbps," IEEE International Symposium on Circuits and Systems, pp. 1151-1154, May 2006.