A High Throughput Sort Free VLSI Architecture for Wireless Applications

Shobana M. K*, Naseeruddin**

**Student, Department of Electronics & Communication Engineering (M.Tech VLSI Design Embedded Systems), Bellary Institute Of Technology & management, Bellary, Karnataka state, India*

***Professor Department of Electronics & Communication Engineering, Bellary Institute of Technology & management, Bellary, Karnataka state, India*

Abstract

For high data rate Multiple Input Multiple Output technology is used in wireless communications. The use of multiple antennas at both transmitter and receiver (MIMO) significantly increases the capacity and spectral efficiency of wireless systems. This project presents a Field Programmable Gate Array (FPGA) implementation for a 4 x 4 breadth first K- best MIMO decoder using a 64 Quadrature Amplitude Modulation (QAM) scheme. A project sort free approach to path extension, as well as, quantized metrics result in a high throughput, low power and area. Finally, VLSI architectural trade-offs are explored for a synthesized using the power analysis, throughput analysis in 120nm technology. The power needed is 20.0025 µW.

I.INTRODUCTION

To satisfy the growing demand of high-speed, reliable wireless communication, MIMO techniques are extensively deployed in several standards, ranging from local area networks to mobile systems. In principle, multiple transmit antennas increase the transmission rate; multiple receive antennas improve signal reliability, equivalently extending the communication range.

Complex computations are therefore required for MIMO algorithms to jointly combine the multiple input streams at the receiver end. The sphere decoding algorithm is one of the most promising solutions to achieve Maximum Likelihood (ML) performance with reasonable computational complexity. Direct mapping from algorithm to hardware is straightforward, but not power and area-efficient. Leveraging specific data patterns and application, the DSP architecture can be better optimized for reduced power and area.

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 "center" at that specific tree level, while the second task involves finding the partial branch metric or cost of extension to a node.

OBJECTIVES:

The main objective of this project is to develop a sort free algorithm using XILINX ISE 14.3where the focus is to set high throughput, low power and area. The objectives of the system are presented below:

1. Reducing complexity that grows exponentially with the number of transmitting and receiving antennas (MIMO).
2. Achieve near ML performance together with reduced complexity.

II. LITERATURE REVIEW

Multiple-Input Multiple-Output (MIMO) technology has emerged as a promising technology for achieving the high data rates of next generation wireless communication systems. MIMO systems improve either the bit-error rate (BER) performance by using Spatial diversity or the data rate via spatial multiplexing. However, Maximum-likelihood (ML) detection for high order MIMO systems faces a major challenge in computational Complexity that grows exponentially with the number of transmit and receive antennas. This limits the practicality of these systems from an implementation point of view, particularly for mobile battery-operated devices.

This reality motivated researchers to consider other suboptimal approaches for MIMO decoding, such as Zero Forcing (ZF), Minimum Mean Square Error (MMSE) and VBLAST (Proakis & Salehi, 1994; Guo & Nilsson, 2003; Myllyla et al., 2005). All of these suboptimal approaches vary in performance and complexity. Recently, the sphere decoding (SD) algorithm which is a tree-based search algorithm enabled the implementation of efficient MIMO decoders that achieve near MLD performance together with reduced complexity (Burg et al., 2005; Barbero et al., 2005; Khairy et al., 2009). Instead of the exhaustive search over all possible combinations of the transmitted symbols, the SD algorithm reduces the complexity by searching only over a finite number of symbols within a circle of radius R centered at the received vector. While the SD approach provides a near ML solution, the runtime measured by the required operations to find the optimum solution is highly dependent on the received signal to noise ratio and the channel conditions. Consequently, the SD algorithm experiences variable throughput problems as the search radius R for each symbol varies according to the noise levels and the channel coefficients. Moreover, the sequential search results in hardware implementations that are not fully pipelined.

To alleviate these problems, the fixed sphere decoding (FSD) algorithm was recently proposed (Barbero & Thompson, 2006 b). The FSD algorithm achieves a fixed throughput performance and enables fully-pipelined hardware by performing fixed number of operations per detected symbol, independent of the noise level.

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.

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.

PROPOSED SORT-FREE APPROACH

The WPE technique is illustrated in Fig. 3.3. 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 the first of the K-best extended paths; the parent who 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 [12] and [13], and also independently in [15] and [16]. In this paper, we study the complexity of the WPE approach versus traditional extension and sorting.

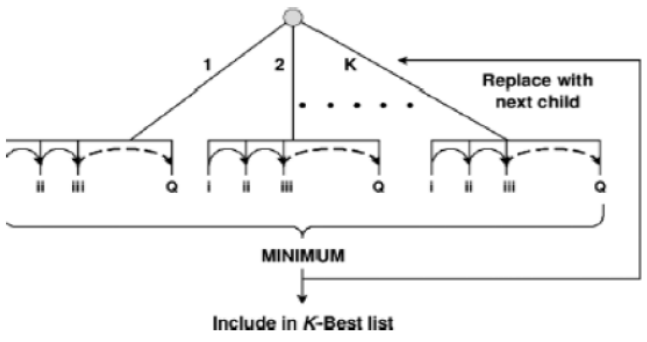

Fig 1 Sort-free, WPE approach.

K-BEST DETECTOR (KB):

The K-best detector can guarantee a fixed throughput and has a BER performance that is close to-ML. In the following, the focus will be on the VLSI implementation of the K-best algorithm. Contributions: The main contributions of this are summarized as follows:

1. A new K-best architecture is presented that operates in a pipelined and parallel fashion.
2. A simplified K-best algorithm based on the l-1norm is introduced, which greatly reduces circuit complexity and increases throughput, while only causing a small BER performance degradation.
3. Finally, a design space exploration examines different algorithm and architecture trade-offs for the implementation of the K-best algorithm. The presented designs achieve of up to 100 Mbps for K=64 or K=8.

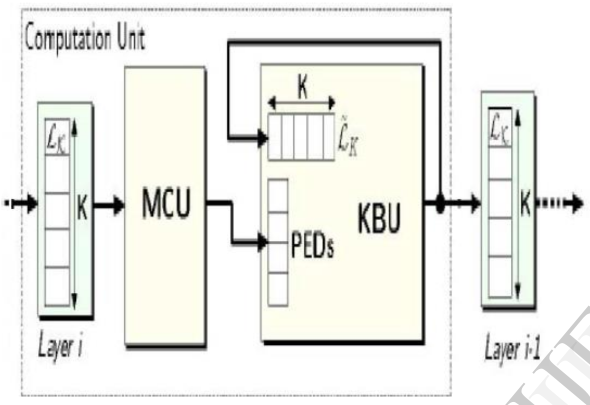

Figure2. One of 2MT pipeline stages of the K-best VLSI architecture

The K-best detector is pipelined such that one layer of the tree is always processed in one pipeline stage (Fig. 2). Each stage consists of a metric computation unit (MCU), a K-best unit (KBU) that determines the K smallest PEDs, and a register bank LK where the K smallest nodes of the previous layer are stored. Together, they form a computation unit. Resource sharing is applied such that the K nodes at the input of the stage are processed one after the other. In each cycle, the MCU delivers the PEDs of all children of a parent node in LK. These PEDs need to be sorted into a list LK where the K smallest PEDs found so far are stored. After K iterations, all children ofthe nodes in LK have been computed by the MCU. The KBU has determined the K smallest PEDs and delivers them to the next pipeline stage. In total, 2MT almost identical copies of the computation unit form the 2MT pipeline stages of the detector.

III. MIMO Detection Methods

3.1 ZERO-FORCING (ZF) DETECTOR

The ZF detector first solves

$$S_{ML} = \arg\min_{s\in\Omega^M} \| y \quad Hs \|^2 \qquad (1)$$

neglecting the constraint **s** $\in S^n$

$$\tilde{s} \overset{\Delta}{=} \arg\min_{s\in R^n} \| y \quad Hs \|$$

$$= \arg\min_{s\in R^n} \| \tilde{y} \quad Ls \| = L^{1}\tilde{y}. \qquad (2)$$

Of course, L^{1} does not need to be explicitly computed. For example, one can do Gaussian elimination: take $\tilde{s}_1 = \tilde{y}_1 / L_{1,1}$, then $\tilde{s}_2 = (\tilde{y}_2 \quad \tilde{s}_1 / L_{2,1}) / L_{2,2}$ and so forth. ZF then approximates S by projecting each $\tilde{s}_k$ onto the constellation S. $\hat{s}_k = [\tilde{s}_k] \overset{\Delta}{=} \arg\min_{s_k \in S} | s_k \quad \tilde{s}_k |$ (3)

We see that $\tilde{s} = s + L^{1}Q^{T}e$ so $\tilde{s}_k$ in 1 is free of inter symbol interference. This is how ZF got its name. However, unfortunately ZF works poorly unless ***H*** is well conditioned. The reason is that the correlation between the noises in $\tilde{s}_k$ is neglected in the projection operation (). This correlation can be very strong, especially if ***H*** is ill conditioned.

There are some variants of the ZF approach. For example, instead of computing $\tilde{s}$ as in (1), one can use the MMSE estimate (take $\tilde{s}$ = E[s|y]). This can improve performance somewhat, but it does not overcome the fundamental problem of the approach

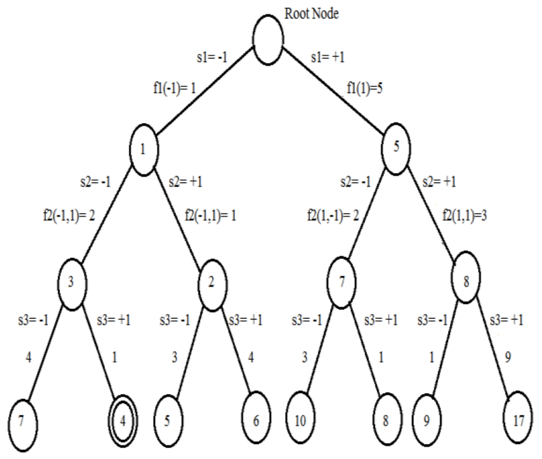

Figure 3.1 Exemplified for binary modulation ($S = \{-1, +1\}$, $|S| = 2$) and n = 3. The branch metrics $fk(s1, \ldots, sk)$ are in blue written next to each branch. The cumulative metrics $f1(s1) + \ldots + fk(s1, \ldots sk)$ are written in red in the circles representing each node. The double circle represents the optimal (ML) decision.

3.2 Sphere Decoding (SD)

The SD [2], [9] first selects a user parameter *R*, called the sphere radius. It then traverses the entire tree (from left to right, say). However, once it encounters a node with cumulative metric larger than *R*, then it does not follow down any branch from this node. Hence, in effect, SD enumerates all leaf nodes which lie inside the sphere $\|\bar{y} - L_s\|^2 \leq$ R this also explains the algorithm's name.

In Figure 3.2, we set the sphere radius to R= 6. The SD algorithm then traverses the tree from left to right. When it encounters the node "7" in the right sub tree, for which 7 > 6 =*R*, SD does not follow any branches emanating from it. Similarly, since 8 > 6, SD does not visit any branches below the node "8" in the rightmost sub tree.

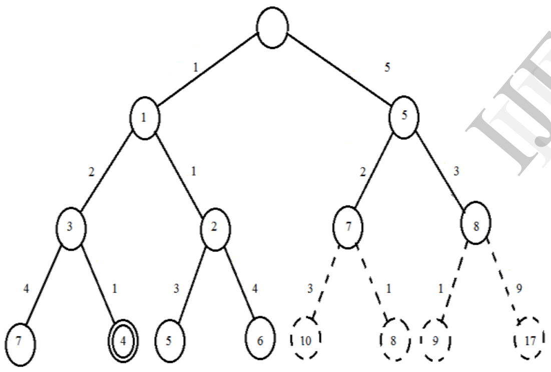

Figure 3.2 SD, No Pruning, (here : R=6)

SD in this basic form can be much improved by a mechanism called pruning. The idea is this: Every time we reach a leaf node with cumulative metric *M*, we know that the solution to (4) must be contained in the sphere$\|\bar{y} - L_s\|^2 \leq$ M. So if *M* <*R*, we can set R:=*M*, and continue the algorithm with a smaller sphere radius. Effectively, we will adaptively prune the decision tree, and visit much fewer nodes than those in the original sphere. Figure 5.3(c) exemplifies the pruning. Here the radius is initialized to *R*= ∞ , and then updated any time a leaf node is visited. For instance, when visiting the leaf node "4," *R* will be set to *R* = 4. This means that the algorithm will not follow branches from nodes that have a branch metric larger than four. In particular, the algorithm does not examine any branches stemming from the node "5" in the right sub tree.

The SD algorithm can be improved in many other ways, too. The symbols can be sorted in an arbitrary order, and this order can be optimized. Also, when traveling down along the branches from a given node, one may enumerate the branches either in the natural order or in a zigzag fashion (e.g., s_k ={-5,-3,-1,-1,-3,5}versuss_k = {-1,1,-3,3,-5,5}). The SD algorithm is easy to implement although the procedure cannot be directly parallelized. Given large enough initial radius *R*, SD will solve noise problem However, depending on ***H***, the time the algorithm takes to finish will fluctuate, and may occasionally be very long.

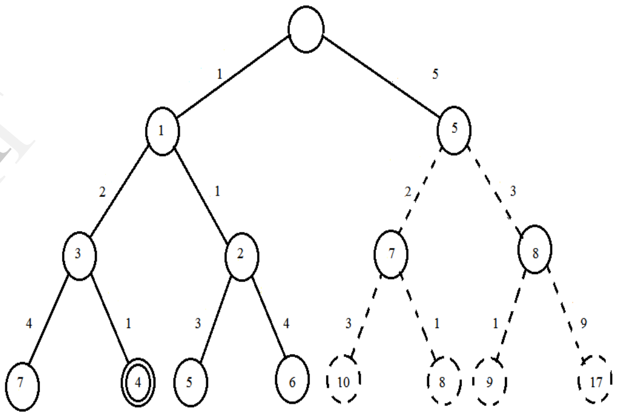

Figure 3.3 SD, Pruning (here R=∞)

3.3 FIXED-COMPLEXITY SPHERE DECODER (FCSD)

FCSD [3] is, strictly speaking, not really sphere decoding, but rather a clever combination of brute-force enumeration and a low-complexity, approximate detector. In view of the decision tree, FCSD visits all $|s|^r$nodes on layer *r*, where $r, 0 \leq r \leq n$ is a user parameter. For each node on layer *r*, the algorithm considers $\{s_1,\ldots,s_1 r\}$ fixed and formulates and solves the sub problem

$\min_{\{s_1,\ldots s_n\} s_k \in s} \{f_{r+1}(s_1,\ldots,s_{r+1}) + \ldots\_ f_n(s_1,\ldots,s_n)\}$ In effect, by doing so, FCSD will reach down to $|s|^r$ of the $|s|^n$ leaves. To form its symbol decisions, FCSD selects the leaf, among the leaves it has visited, which has the smallest cumulative metric $f_1(s_1)+\ldots+f_n(s_1..s_n)$. The sub problem (3.6) must be solved once for each combination $\{s_1\ldots s_n\}$, that is $|s|^r$ times. FCSD does this approximately, using a low-complexity method (ZF or ZF-DF are good choices). This works well because (3.6) is over determined: there are *n* observations $(\tilde{y}_1..\tilde{y}_n)$, but only *n* – *r* unknowns $(s_{r+1}..s_n)$. More precisely, the equivalent channel matrix will be a tall sub matrix of ***H***, which is generally much better conditioned than ***H***. Figure (3.4) illustrates the algorithm. Here *r*=1. Thus, both nodes “1” and “5” in the layer closest to the root node are visited. Starting from each of these two nodes, a ZF-DF search is performed.

Naturally, the symbol ordering can be optimized. The optimal ordering is the one which renders the problem (2) most well-conditioned. This is achieved by sorting the symbols so that the most “difficult” symbols end up near the tree root. Note that “difficult symbol” is nontrivial to define precisely here, but intuitively think of it as a symbol s_k for which $\sum k$, k is large.

The choice of *r* offers a tradeoff between complexity and performance. FCSD solves (1) with high probability even for small *r*, it runs in constant time, and it has a natural parallel structure. Relatives of FCSD that produce soft output also exist [4].

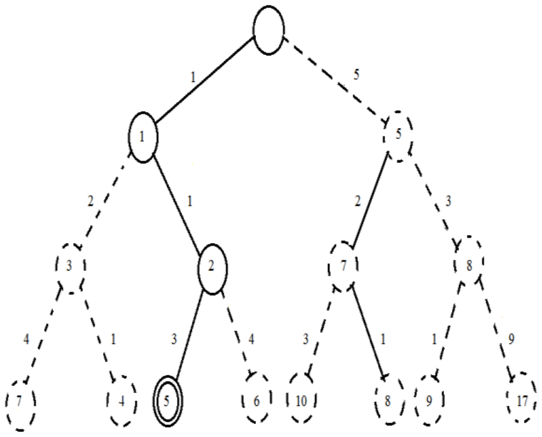

Figure 3.4 FCSD (here r=1)

IV.HARDWARE IMPLEMENTATION

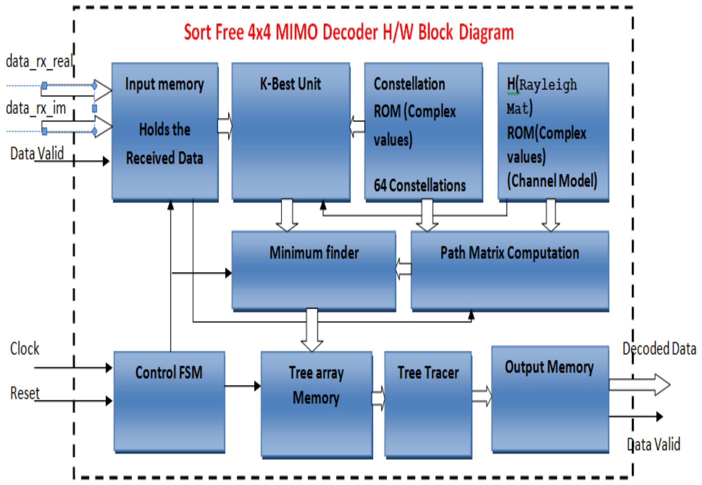

Figure 4 Sort Free 4x4 MIMO Decoder H/W Block.

The input data received will be in the Complex form which contains the data received by a receiver the received data will convoluted with the channel model (will be the sum of all the data transmitted by 4 transmitters). The data fed will already be multiplied by Unitary matrix Q’(Transpose of Q)

The 1st data from the receive memory will be 1st sent to K-best module which will give the 8-(K) best possible values which will result in minimal error. The data and the error will be stored in the Tree array memory.

Subsequent data will be sent to path matrix computation block which intern feed to Minimum finder block which will find the minimum; this will be added with the error of previous message and stored in the Tree array.

The sequence is repeated for 3 levels. Once complete Tree building is done, the tree tracer will trace the tree to find the data with minimal error and stores in output Memory.

V. DETAILED FLOW CHART

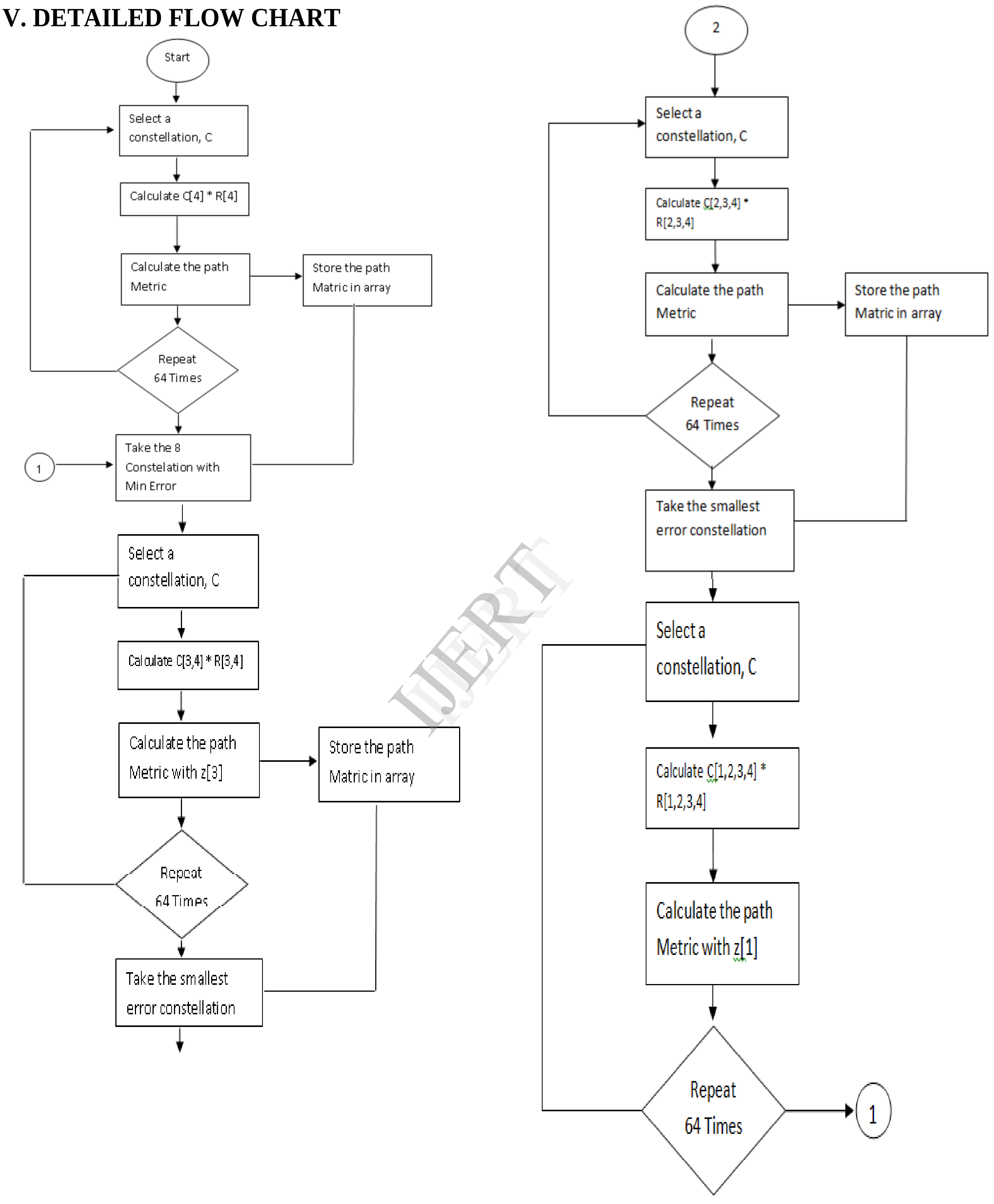
Start
Select a constellation, C
Calculate C[4] * R[4]
Calculate the path Metric
Store the path Matric in array
Repeat 64 Times
1
Take the 8 Constellation with Min Error
Select a constellation, C
Calculate C[3,4] * R[3,4]
Calculate the path Metric with z[3]
Store the path Matric in array
Repeat 64 Times
Take the smallest error constellation
2
Select a constellation, C
Calculate C[2,3,4] * R[2,3,4]
Calculate the path Metric
Store the path Matric in array
Repeat 64 Times
Take the smallest error constellation
Select a constellation, C
Calculate C[1,2,3,4] * R[1,2,3,4]
Calculate the path Metric with z[1]
Repeat 64 Times
1

Figure 5. Detailed Flow Chart

In the paper, a MIMO transmitter and receiver system has been used. In this project, four transmitter and four receivers are considered. The idea can be further extended to any number of transmitter and receivers but it would require a lot more hardware than the design present in this project. The alphabet size is assumed to be 64 also known as constellation size. Now, four transmitters can send any of the 64 constellations. The aim of the receiver should be decode the signal which is nearest to the transmitted signal i.e. the signal which has the minimum noise. At the receiver end, all the receivers would be getting the combined signals from the all the four transmitters. This can be viewed as a tree which has 64 branches emerging from the root node. Now, 64 branches from the root node denote the 64 constellations which the any of the four transmitters has sent. At next level of the tree, each of these 64 branches will have 64 more branches. Now at level 2, the tree covers the any combination of the signals from any of the 2 transmitters. At the level 2, there would 4096 combinations of the signal can be possible. In this project, four transmitter have been studied which means by the time the end of tree is reached till the level four there would be 16777216 leaf nodes. The traversing of each of these nodes from the root node is a humongous task and will require a lot of hardware support. In case, the hardware optimization is considered the amount of time to calculate the decoded signal would be very large. The maximum likelihood will traverse each node and find out the minimum path metric of each leaf node. At each level, matrix multiplication is used to calculate the noise. The noise of each branch of the tree is known as branch matrix. All the paths form the leaf root node to the leaf node is traversed. For each traversed path, the branch metric of each node in that path is added and stored in the memory. The path (from the root to the leaf node) which has the smallest path metric is the decoded signal. It is pretty clear if the maximum likelihood algorithm is used for the decoding of the signals, it will require a lot of hardware. Also, if the numbers of transmitters are more, at some point of time it would become impractical to implement the maximum likelihood.

So, in the above flowchart another decoding technique called fixed sphere decoding has been used which is more practical to implement in hardware. In this implementation, the “fixed” (also known as “K”) has been limited to eight. In this technique, the noise is calculated for all the 64 constellations at the level 1 of the tree. Out of these 64 constellations, only eight nodes are selected. The selection criterion is the nodes which have the minimum branch metric value.

The branch metric for the each node at level 1 is the difference of correct signal and received signal. The correct signal is calculated by multiplication of each constellation(C) (which represents nodes in the tree) with channel realization metric (R) as shown as first step in the flowchart. Now difference of each multiplication with received signal is stored into the memory. At this point of time, all the path metric of the nodes (corresponding to the each constellation) is known. Now, using the stored value, the eight nodes for which the path metric is minimum are selected. All the other nodes would not be traversed further and would be discarded. Now, for these eight nodes further traversing will be done. Each of the eight nodes would have 64 child nodes.

One of these eight nodes is selected; again the difference of the correct signal with the received signal is calculated. At the level 2, the received signal is combination of one constellation with another constellation. For the correct signal calculation, the constellation of the parent node is fixed (one of the eight node selected above) i.e. one constellation is fixed. The second constellation can have 64 values as represented by the 64 child node of the selected node. So, 64 branch metric are calculated for the 64 nodes and stored into the memory. Once the entire 64 branch metric are available, the path metric of the parent node is added to these branch metric to calculate the path metric of the 64 child nodes. The child node which has the minimum path metric is selected and all the remaining 63 nodes are discarded. This selected node will have 64 child nodes (at level 3). The received signal at this level would be combination of 3 constellation points. For the correct signal calculation, first 2 nodes (representing the selected constellations above at level1 & level2) are fixed. The third signal can be any of the 64 constellation point. The difference between all the 64 correct signal and received signal is calculated and stored in the memory. Once all the 64 noise are available, the node which has the smallest noise is selected for further and other 63 nodes are rejected. Again, the same process is repeated at the final level (level 4) by keeping the first 3 constellations fixed. The 64 differences between the correct signal and received signal is calculated. The node with the smallest difference (minimum noise) is selected and stored. In this way, the first path of the complete tree is traversed and a received signal has been decoded. The same process of depth first search tree traversing is done for the remaining seven nodes selected at the level 1. At the end of it, 8 paths would be available and the path which has the minimum path metric would be the decoded signal from the decoder. This implementation is present in each of the four receivers.

IV. VLSI ARCHITECTURE

The detector cell architecture of the system is shown in Fig. 3. 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 the (i-1) the level. In each cycle, the computes, in parallel, the PEDs points of their children of one of the K parent nodes. 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. 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 Min Finder (MF) block, which is implemented using a logarithmic arrangement of K comparators. The MF is pipelined with registers after every comparator

V. SIMULATION RESULTS AND VERIFICATION

An FPGA implementation of the system utilizing six parallel detectors was carried out using a Xilinx XC2VP30device. The experimental results are shown the simulation results in Fig. 5 the entire decoder performance. Fig 6.Denotes the Synopsys power report, where the power required is 20.0025 µW for the KB unit. The area analysis using Synopsys is given in fig 7.The detection problem and the computational complexity is reduced. The throughput value obtained is about453Mbps. The Min Finder (MF) block, which is implemented using a logarithmic arrangement of K comparators and the MF is pipelined with registers after every comparator. Therefore, the power consumption is reduced and the computational complexity is reduced. The applications like smart antennas are used with the technology.

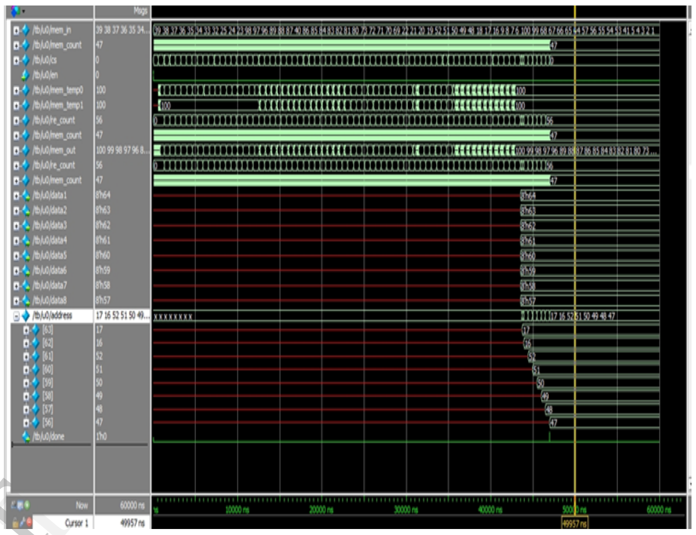

Figure 5.1 Ascending order Sorter Output

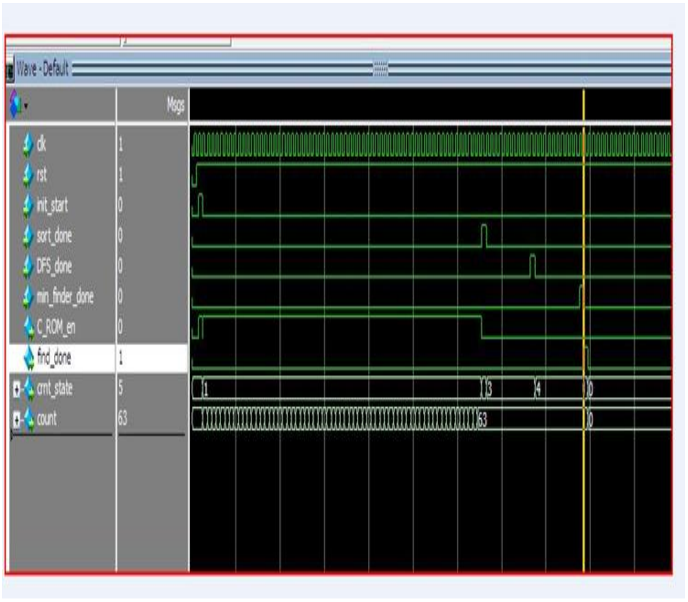

Figure 5.2 Sort Free Output

V. CONCLUSION

A novel, high –throughput, VLSI architecture for the K-best MIMO detector system has been simulated using Synopsys, ModelSim and verified. The use of 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. A 4 x 4 breadth first K-best MIMO decoder using a 64 Quadrature Amplitude Modulation (QAM) scheme is carried out. A novel sort free approach to path extension, as well as, quantized metrics result in a high throughput, low power and area is obtained.

References

1. SudipMondal, Ahmed Eltawil, , IEEE, Chung-An Shen, "Design andImplementation of a Sort Free K-Best Sphere Decoder," IEEE Trans. VLSI , 2010
2. B. M. Hochwald and S. Brink, "Achieving near-capacity on a multipleantenna channel," IEEE Trans. Commun., vol. 51, no. 3, pp. 389– 399,Mar. 2003.
3. M. O. Damen, E. Gamal, and G. Caire, "On maximum-likelihooddetection and the search for the closet lattice point," *IEEE Trans.Inform. Theory*, vol. 49, no.10, Oct 2003
4. H. Vikalo, B. Hassibi, and U. Mitra, "Sphere-constrained ML detectionfor frequency-selective channels," IEEE Trans. Commun., vol. 54, no. 7, pp. 1179–1183, Jul. 2006.
5. B. Hassibi and H. Vikalo, "On the sphere decoding algorithm. Part I: The expected complexity, "IEEE Trans. Signal Process., vol. 53, no. 8, pp. 2806–2818, Aug. 2005.
6. C. P. Schnorr and M. Euchener, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," Math. Program., vol. 66, pp. 181–191, 1994.
7. J.Wang and B. Daneshrad, "Performance of linear interpolation-basedMIMO detection for MIMO-OFDM systems, " in Proc. IEEE Wireless Commun. Netw. Conf., Mar. 2004, pp. 981–986.