Design of Mimo Detector using K-Best Algorithm

Akila. V, Jayaraj. P

Assistant Professors, Department of ECE, Surya Engineering College, Erode-638107.

Abstract- This paper presents an efficient VLSI architecture for a 4x4 64-QAM multiple-input–multiple-output (MIMO) detector. The augmentation is done by on demand expansion of intermediate nodes of the tree rather than exhaustively, along with pipelined distributed sorters. The proposed architecture has a stable critical path independent of constellation size, scalable to higher number of antennas with efficient distributed sorters. Further, modification will be carried out with the faster multiplication unit to make it scalable to higher number of antennas.

Index Terms- K-best algorithm, multiple-input-multiple-output (MIMO), on-demand expansion.

I. INTRODUCTION

Multiple-input–multiple-output (MIMO) systems have gained significant concentration as the hand-picked technology in many standards such as IEEE 802.11n, IEEE 802.16e, IEEE 802.16m and the long term evolution (LTE) paper due to the high spectral efficiency. To accomplish the potential of MIMO systems, the main challenge is to design high throughput detection device of low complexity with near maximum-likelihood (ML) performance that are suitable for efficient very large scale integration (VLSI) implementation. Unluckily, with the number of transmit antennas and the constellation size the complexity of the optimal ML detection scheme grows exponentially. Available lower-complexity detectors such as zero-forcing (ZF), minimum mean-square error (MMSE) or successive interference cancelation (SIC) detectors can greatly reduce the computational complexity but still they go through performance loss.

The other choice is to use near-optimal non-linear detectors. Near-optimal non-linear detectors can be classified as depth-first search, breadth-first search, and best-first search based on non-exhaustive search. Depth-first sphere decoding (SD) have drawn attention in depth-first approach whose performance is optimal under the assumption of unlimited execution time. However, the actual runtime of the algorithm not only accounts for the channel realization but also depends on the operating signal-to-noise-ratio (SNR). Thus obtaining a variable sustained throughput, which results in extra overhead in the hardware due to the extra required I/O buffers and lower hardware utilization.

Among the breadth-first search methods, the familiar approach is the K-best algorithm. The K-best detector assures fixed-throughput with a performance close to ML which is independent of SNR. Being fixed-throughput in nature along with the fact that the breadth-first approaches are feed-forward detection schemes, makes them especially attractive for VLSI implementation. Moreover, in spite of various published architectures for the implementation of 4×4 16-QAM systems, an efficient high-throughput application specific integrated circuit (ASIC) implementation for 64-QAM systems at high data rate is still a major challenge and has not been fully addressed in the literature.

In this paper, an efficient VLSI architecture is designed for a 4×4 64-QAM *K*-best MIMO detector, which lessen the problems described above and operates at a significantly higher throughput. It efficiently expands all the possible children and provides K-best solution.

II. K-BEST ALGORITHM

A spatial multiplexing MIMO system is considered with N_t transmit and N_r receive antennas whose equivalent baseband model of the Rayleigh fading channel described by a complex-valued N_r x N_t channel matrix $\tilde{\mathbf{H}}$.The complex baseband equivalent model can be expressed as $\tilde{y} = \tilde{H}\tilde{s} + \tilde{v}$ where $\tilde{s} = [\tilde{s_1}, \tilde{s_2}, \dots, \tilde{s_{N_t}}]^T$ denotes the N_t-dimensional complex transmit signal vector, in which each element is independently drawn from a complex constellation *O* (a symmetric *M*-QAM scheme with $\log_2 M$ bits per symbol, i.e., $|O| = M$), $\tilde{y} = [\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_{N_r}]^T$ is the N_r –dimensional received symbol vector, and $\tilde{v} = [\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_{N_r}]^T$ represents the N_r - dimensional independent identically distributed complex zero-mean Gaussian noise vector with variance σ^2, i.e., $\tilde{v}_i \sim N_c(0, \sigma^2)$. The real model can be derived equivalent to this system using a real-valued decomposition (RVD) model as follows:

$$\mathbf{y = Hs + v} \qquad (1)$$

where $y=[y_1, y_2, \dots, y_{2Nr-1}, y_{2N_r}]^T$, $s=[s_1, s_2, \dots, s_{2Nt-1}, s_{2N_t}]^T$ and **H** are the equivalent real-valued vectors with the following mappings $y_{2k-1} = \Re\{\tilde{y_k}\}$, $y_{2k} = \Im\{\tilde{y_k}\}$, $s_{2k} = \Im\{\tilde{y_k}\}$, and **v and H** are decomposed accordingly, where $\Re(.)$ and $\Im(.)$ denote the real and imaginary parts of the variables, respectively. Note that $s_i \in \Omega = \{-\sqrt{M}+1, \dots, -1, +1, \dots, +\sqrt{M}+1\}$, where Ω is the set of possible real entries in the constellation for in-phase and quadrature parts with $|\Omega| = \sqrt{M}$. The aim of the MIMO ML detection method is to find the closest transmitted vector $\hat{\mathbf{s}}$ based on the observation , i.e.,

$$\hat{\mathbf{s}} = \arg \min_{s \in \Omega^{2N_t}} \|\mathbf{y} - \mathbf{Hs}\|^2 \qquad (2)$$

The exhaustive-search ML detection is not effective to implement for large constellation sizes (i.e., 64-QAM and larger) because of its exponential complexity nature. The *K*-best

algorithm, a.k.a. the M-algorithm, is a near-ML technique to solve the above problem with a much lower complexity.

The problem in (2) can be considered as a tree-search problem with $2N_t$ levels. The *K*-best algorithm explores the tree from the root to the leaves by expanding each level and it selects the best candidates with the lowest PED in each level that are the surviving nodes of that level. Consider the problem in (2), and let us denote the QR-decomposition of the channel matrix as **H=QR**, where **Q** is a unitary matrix of size $2N_r \times 2N_t$ and **R** is an upper triangular $2N_t \times 2N_t$ matrix. Applying $\mathbf{Q}^{\mathbf{H}}$ to (1) results in

$$\mathbf{Z} = \mathbf{Q}^{\mathbf{H}}\mathbf{y} = \mathbf{R}\mathbf{s} + \mathbf{w} \qquad (3)$$

where $\mathbf{w} = \mathbf{Q}^{\mathbf{H}}\mathbf{v}$. Since the nulling matrix is unitary, the noise, **w**, remains spatially white and the norm vector in (2), which represents the ML detection rule, can be rewritten as $\hat{\mathbf{s}} = \arg\min_{s\in\Omega^{2N_t}} ||\mathbf{z} - \mathbf{R}\mathbf{s}||^2$. Exploiting the upper triangular nature of **R**, this norm vector can be further expanded as

$$\hat{s} = \arg\min_{s\in\Omega^{2N_t}} \sum_{l=1}^{2N_t} z_l - \sum_{j=l}^{2N_t} \eta_{lj}\, s_j\,|^2 \qquad (4)$$

TABLE I
K-BEST ALGORITHM

Step 1) Initialization: Set one path at level $2N_t+1$ with PED=0
Step 2) Expansion: Expand K surviving paths from the last level to $\sqrt{M}$ new children in Ω & calculate the PED of new $K\sqrt{M}$ paths.
Step 3) Sorting: Sort all $K\sqrt{M}$ existing paths PED and select the best K paths.
Step 4) If not at the last level, go to step 2, o.w. announce the path with the lowest PED.

which is a tree-search problem with levels. Starting from $l = 2N_t$, (4) can be evaluated recursively as follows:

$$T_l(s^{(l)}) = T_{l+1}(s^{(l+1)}) + |e_l(s^{(l)})|^2 \qquad (5)$$

$$e_l(s^{(l)}) = z_l - \sum_{j=l}^{2N_t} \eta_{lj}\, s_j = L_l(s^{(l)}) - \eta_{ll}\, s_l \qquad (6)$$

for $l = 2N_t, 2N_t - 1, \ldots, 0$, where $s^{(l)} = [s_l s_{l+1} \ldots s_{2N_t}]^T$, $T_l(s^{(l)})$ is the accumulated partial Euclidean distance (PED) with $T_{2N_t+1}(s^{2N_t+1}) = 0$, $e_l(s^{(l)})^2$ denotes the distance increment between two successive nodes/levels in the tree, and

$$L_l(S^{(l)}) = z_l - \sum_{j=l+1}^{2N_t} \eta_{lj}\, s_j = \eta_{ll}(\bar{z}_l - \sum_{j=l+1}^{2N_t} \bar{\eta}_{lj}\, s_j)$$

$$= \eta_{ll}\bar{L}_l(s^{(l)}) \qquad (7)$$

where $\bar{z}_l$, $\bar{\eta}_{lj}$, and $\bar{L}_l(s^{(l)})$ denote the scaled z_l, η_{lj}, and $L_l(S^{(l)})$ by η_{ll}, respectively, i.e., $z_l = \bar{z}_l\,\eta_{ll}$, $\eta_{lj} = \bar{\eta}_{lj}\eta_{ll}$, and $\bar{L}_l(s^{(l)}) = L_l(S^{(l)})\,\eta_{ll}$. Based on the above formulation, the *K*-best algorithm can be described as in Table 1.

The path with the lowest PED at the last level of the tree is the hard-decision output of the detector, whereas, for a soft-decision output, all of the existing paths at the last level are considered to calculate the Log-Likelihood Ratios (LLRs)

Let us consider a $2N_r \times 2N_t$ real-model MIMO system with channel matrix **H**(1). The system is considered as a detection problem in a tree with $2N_t$ levels, ***K*** nodes per level and $\sqrt{M}$ children per node. The algorithm starts from the last row of the matrix because of the upper triangular structure of matrix **R** and goes all the way up to the first level of the detection tree.

There are two main computations that take major roles in the total computational complexity of the algorithm, namely, 1) the expansion of the surviving paths, and 2) the sorting. Therefore, for VLSI realization of the K-best algorithm these two computational cores are considered as most important one.

1) Expansion: Efficient expansion method called the *on-demand* expansion scheme is carried out, which avoids the exhaustive enumeration of the children and obtain exact *K*-best implementation with no performance loss. There are ***K*** parent nodes at each level and $\sqrt{M}$ children per parent, thus the path metrics of $\boldsymbol{K}\sqrt{M}$ children is to be computed in each level, which leads to a large computational complexity. The computational complexity is independent of constellation size and proportional to the *K* value.

2) Sorting: In this paper, a distributed sorter, working in a pipelined structure based on the on-demand expansion scheme is considered.

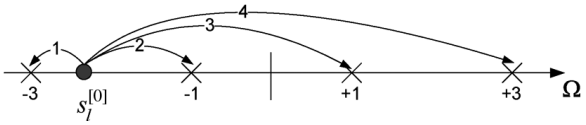


Figure 1: Order of the SE row-enumeration for four consecutive enumerations in 16-QAM

.

It obtains K best candidates in K clock cycles. It is applicable for any value of K and M.

III. PROPOSED K-BEST DETECTION SCHEME

Consider level of the tree and assume that the set of *K*-best candidates in level l+1(denoted by K_{l+1}) is known. Each node in level has possible children, so there are possible children in level *l*. The *K*-best scheme is used to find the first child (FC) of each parent node in K_{l+1}. Among these first children the one with the lowest PED is one of the *K*-best candidates in K_l for sure. That specific child is taken and replaced by its next best sibling. This process repeats *K* times to find the *K*-best candidates in level *l* (K_l). The same approach is used at each level.

A. First/Next Child Calculation

In the on-demand scheme described above, the *first* and *next* child are required to be determined. Based on the system model

in (5), the first child ($s_l^{[1]}$) of a node in K_{l+1} is the one minimizing $e_l(s^{(l)})$, i.e.,

$$s_l^{[1]} = \arg\min_{s\in\Omega^{2N_t}} e_l(s^{(l)})^2 = \arg\min_{s\in\Omega^{2N_t}} L_l(s^{(l)}) - \eta_{ll} s_l^2) \quad (8)$$

Therefore, $s_l^{[1]}$ can be found by rounding $s_l^{[0]} = L_l(s^{(l)})/\eta_{ll}$ to the nearest integer value in Ω. In order to find the next children (NC), the Schnorr-Euchner technique is employed, which implies a zig-zag movement around $s_l^{[0]}$ to select the consecutive elements in Ω. Figure. 1 shows such an enumeration for $\sqrt{M}$ = 4. The SE enumeration by changing the search direction finds the closest points in a real domain one-by-one. The procedure is described in Table II, where n_l^j denotes the number of moves, and **SignBit** represents the direction. **SignBit** alternates between positive and negative until it reaches $\pm\sqrt{M}-1$. The number of moves also increases by 2 every time and is reset to 2 if boundaries of Ω are reached.

TABLE II

First/Next child selection procedure for node j

A) First child

A.1) $s_l^{[0]} = L_l(s^{(l)})/r_{ll}$

A.2) $s_l^{[1]} \leftarrow$, $n_l^j \leftarrow 2$

B) Next (k-th) child

B.1) SignBit= $\begin{cases} SB & s_l^{[k-1]} \neq \pm(\sqrt{M}-1) \\ -1 & s_l^{[k-1]} = (\sqrt{M}-1) \\ +1 & s_l^{[k-1]} = -(\sqrt{M}-1) \end{cases}$

where **SB=Sign**($s_l^{[k-1]} - s_l^{[k-2]}$)

B.2) $s_l^{[k]} \leftarrow s_l^{[k-1]} + n_l^j \times$ **SignBit.**

B.3) $n_l^j = \begin{cases} 2 & if\ s_l^{[k-1]} = \pm(\sqrt{M}-1) \\ n_l^j + 2 & otherwise \end{cases}$

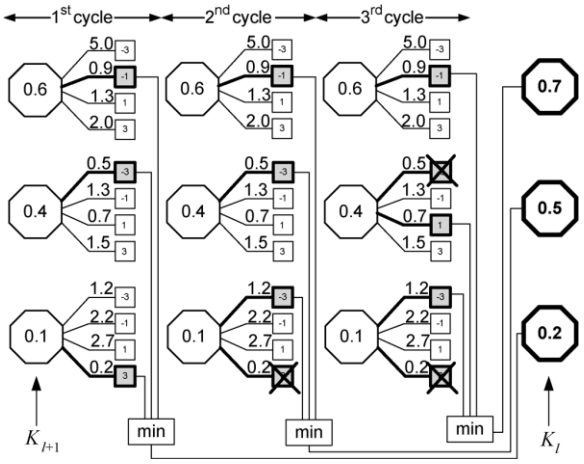


Figure 2: The proposed distributed *K*-best algorithm for $\sqrt{M}$ = 4 and *K*=3 and example PED values.

IV. VLSI IMPLEMENTATION

A pipelined structure is used, which performs the child expansion and minimization jointly in a pipelined fashion and implements the sorting in a distributed way without sacrificing the throughput. The proposed architecture with all intermediate parameters for a 4×4, 64-QAM MIMO system with *K*=10 and Ω={-7,-5,-3,-1,+1,+3,+5,+7} is shown in Figure. 3. There are $2N_t$=8 levels in the tree. The 8th level of the tree, corresponding to the last row of (3), opens up all the possible values in Ω, and calculates their corresponding PEDs.

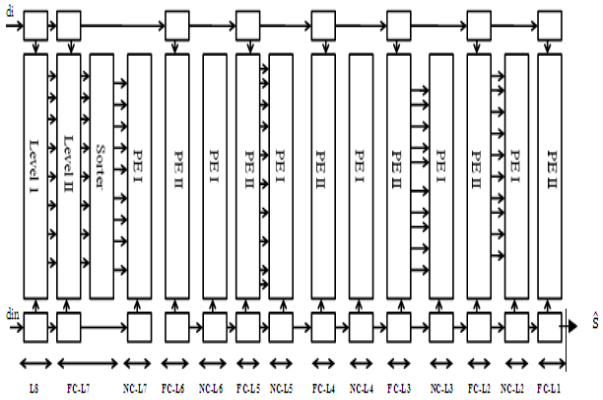


Figure 3: Proposed pipelined VLSI architecture of the *K*-best algorithm for the detection of a 4× 4, 64-QAM system with K=10

The output of this stage is |Ω|=8 PED values which is performed by Level I. First child is found and its PED is updated using the FC-Block in Level II. Then the FC with the lowest PED should be determined, which requires all the FCs to be sorted. This is done using the Sorter block. The output of the Sorter block is the sorted FCs of level 7 are loaded simultaneously to the next stage PE II block is used to generate and sort the list of all FCs of the current level and 1 PE I block is used to generate the *K*-best list of the current level to the next stage.

V. DETAILED VLSI ARCHITECTURE

The inputs to the architecture are the entries of the matrix as well as **z** the vector in (3). The basic blocks used throughout the architecture discussed as follows.

Multiplication (MU): Two types of multiplication carried out in the architecture. The multiplication of $\overline{z_l} \times \eta_{ll}$ and $s_j \times \overline{\eta_{lj}}$. The former multiplication implemented using a 13 bit × 13 bit multiplier. The later carried out with a faster multiplication unit consuming less area shown in Figure.4

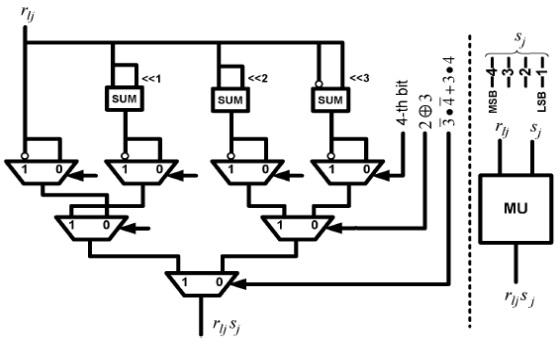


Figure 4: Alternative architecture for multiplication (MU).

Mapper: once $s_l^{[0]}$ is calculated as given in Table II, then by mapping it to the nearest odd integer in Ω first child is calculated which is carried out in two consecutive stages as shown.

Limiter: If $\overline{s_l^{[0]}}$ exceeds the boundaries of Ω, the limiter block is used to bound the values with in upper and lower range of Ω (e.g., +7/-7 for 64-QAM). The limiter block is shown in Figure. 5, with examples to determine first child for 3 values.

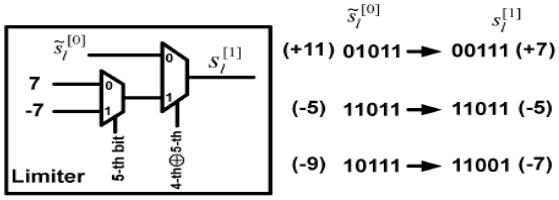


Figure 5: Architecture for the Limiter block.

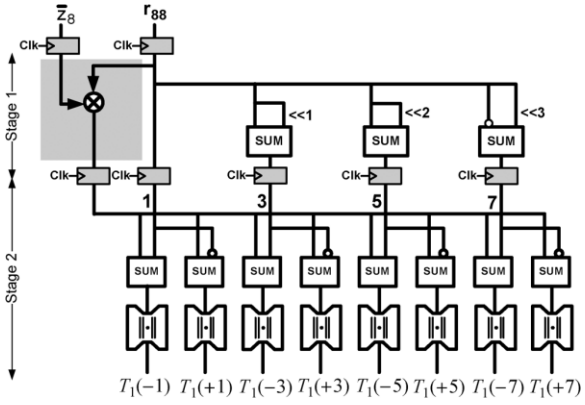


Figure 6: Architecture for Level I.

1) Level I Block: The nodes in the 8th level of tree are taken as input to the Level I and corresponding PED values are generated as output. The architecture shown in Figure. 6 involves 13 bit × 13 bit multiplier, adder and absolute value block. Absolute value block may represent either l^1 norm or l^2 norm. Former can be replaced with squaring operation and later can be replaced with carry save adder technique. Fine grained pipelining is introduced in this block to increase the system throughput. 2 stage pipelining is employed here shown by the introduction of 2 and 5 positive-edge-triggered flip-flops in stage 1 and 2 respectively. Resulting in avoidance of long critical path assuming that there is only one multiplication unit in the critical path.

2) Level II Block: The PED values of the 8th level are given as input to the Level II block and PED values of first children in 7th level are generated as output. The first children of the 7th level are same and independent of 8th level of the tree due to the structure of the **R** matrix. To find first child $\overline{z_7}$ is applied to the limiter/mapper block and it is multiplied with r_{77} by using MU block. Normalized input $\overline{z_7}$ is also multiplied with r_{77}. The Euclidean distance between the first child and received vector is calculated and it is added with PED values of 8th level, resulting in updated 8 PED values of 7th level. 4-stage pipelining is introduced to reduce the length of critical path as shown in Figure. 7.

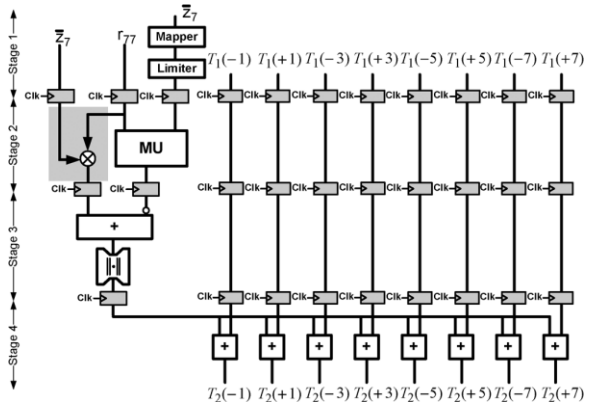


Figure 7: Architecture for Level II.

3) Sorter Block: The updated PED values of 7th level FC's are given as input to the sorter block to generate sorted list of these 8 PED values. Eight inputs are denoted as D_0-D_7 loaded with the **ctrl** signal and the outputs are stored in flip-flop denoted as "N" as shown in Figure. 8. The general sorter involves two operations in one clock cycle (min/max and data exchange) sorts **K** numbers in **K/2** clock cycles.

4) PE I Block: PE I is a common block shown in Figure. 9 used from level 7 to level 2. The sorted list of first children of each level is received and it generates **K**-best candidates of that level. For example in level 7 the output of PE I, called NC-L7 consists of lowest PED values generated one-by-one at the output in series. This block composed of a sorter, and a block called NC-Block on the feedback path. Sorted list of PEDs received from the preceding stage and selects best one with lowest PED, and it is taken as K-best candidate. NC-Block then calculates the next best sibling of that candidate and fed it back to the sorter to find the location of the new sibling that present already in the sorted list. Efficient architecture to be considered for NC-Block since it is present in the critical path.

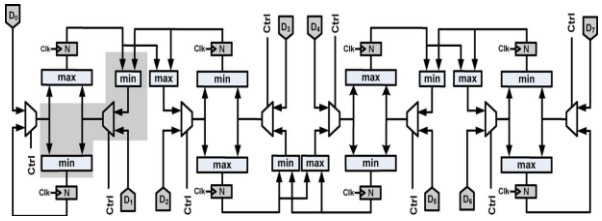


Figure 8: Architecture for the Sorter block.

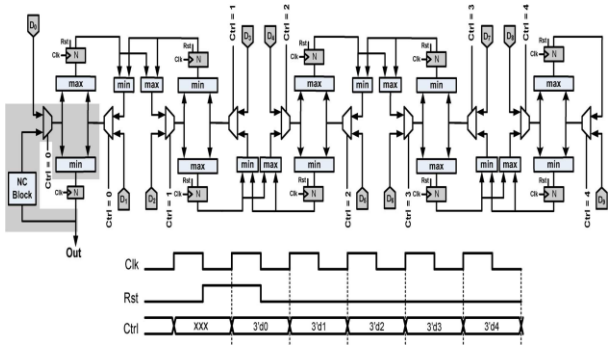


Figure 9: Architecture for the PE I block.

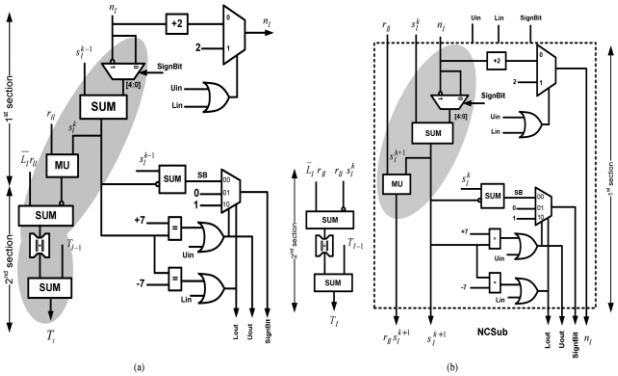


Figure 10: Architecture for the NC-Block inside the PE I block (a) Original. (b) Improved.

5) NC-Block: The NC-Block shown in Figure. 10 involves three operations. SignBit calculates the direction of the SE enumeration for the next child, check for lower/upper boundary in Ω and finally calculates the PED value of new sibling. To obtain a efficient architecture the following two methods are used in VLSI architecture:

1)Avoid multiplication: Since the value of $\overline{L_l}$ in (7) independent of current sibling and it depends only on the selected symbols till level *l*. The calculation of $\overline{L_l}$ and $\overline{L_l}$ r_{ll} can be done by FC-Block in the preceding stage and it is moved as input to the NC-Block. This is the preferred technique that removes the multiplication from the critical path.

2)Broken critical path: The critical path is broken down in to two smaller parts. As can be seen from the Figure. 10(a), the original structure has 3 adders along with the MU unit. Using scheduling method, the first part of the critical path calculates the next sibling and further it is moved to the FC-Block in the preceding block as shown in Figure. 10(b). For each it calculates both the first and second best child and it sends to the NC-Block. The NC-Block calculates the second best child while determining the third child and it goes on. This obviously shows that NC-Block always calculates one child ahead. The first section calculates the second best child is added to the preceding FC-Block. The second section consists of two adders whose complexity independent of *K* value.

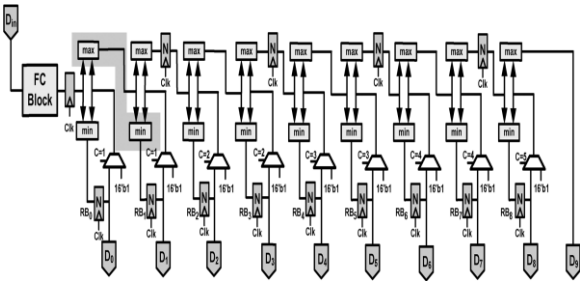


Figure 11: Architecture for the PE II block.

6) PE II Block: As each of the *K*-best candidates are found, it is transferred to the PE II block to calculate the first children of next level and sort them. The architecture of PE II block is shown in Figure. 11., where D_{in} served as input to generate output namely D_0-D_9. At first, FC-Block calculates the first child of K-best candidate and its PED values are updated. To sort these PED values, it is followed by a sequential sorter. PE II block is connected to the output of PE I bock in a pipelined fashion hence the process is carried out on the basis of clock cycles. Two register banks are updated at the same time in every clock cycle.

7) FC-Block: The main assignment of this block is to calculate $\overline{L_l}$ in (7). Based on this $\overline{L_l}$ value, the first child of current parent node and its PED values are calculated. Pipelining has been used by the introduction of FFs on all the forward paths. The proposed architecture for the FC-Block consists of 5 pipeline levels. In first two levels $\sum_{j=l+1}^{2N_t} \eta_j \; s_j$ is calculated. $\overline{L_l}$ is calculated in third level, which is used to find first child using mapper and limiter bock. The next best child needed for NC-Block is determined by finding the number of moves and direction of moves for the SE enumeration. Finally the PED value of announced first child is calculated in the level 5 as shown in Figure. 12.

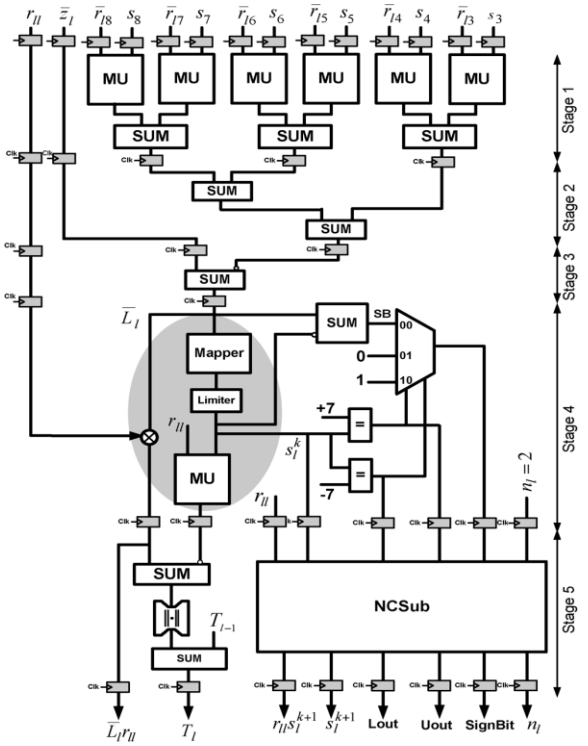
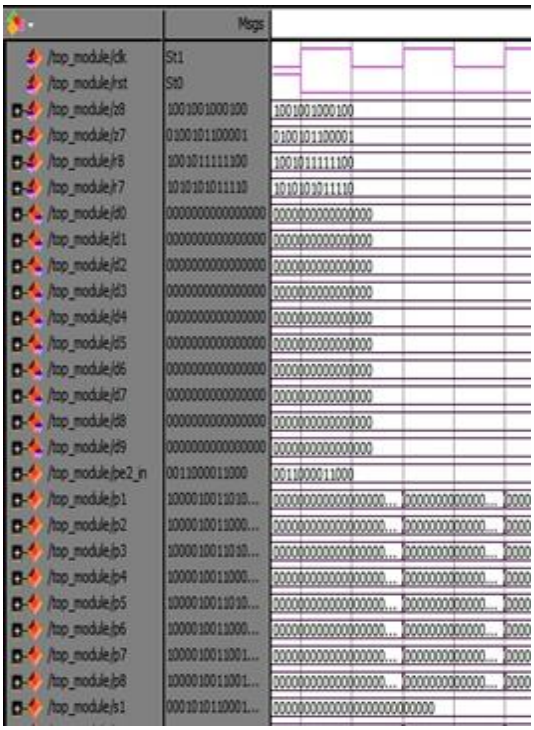
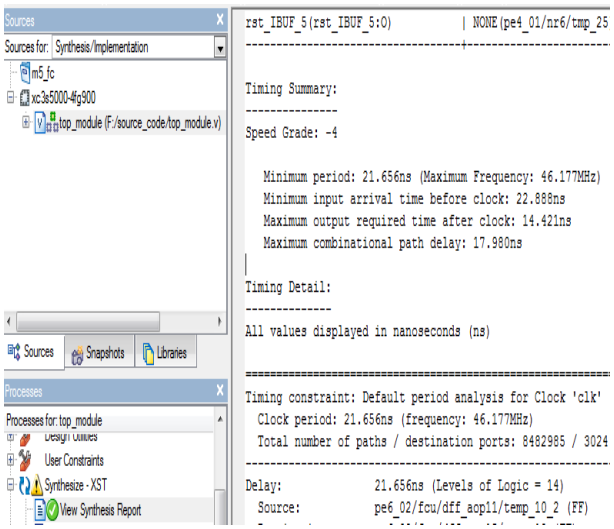


Figure 12: Architecture for the FC-Block inside the PE II block.

VI. RESULT ANALYSIS

A) Simulated waveform





The best child with lowest PED value is found as shown in the simulated waveform. Array multiplier unit involved in this structure obtaining delay of 21.656 ns, throughput of 46.177 MHz carried out in XILINX ISE 9.1i.

VII. CONCLUSION AND FUTURE WORK

The efficient pipelined architecture is simulated with reduced delay achieving low data rate. It supports for any value of K and M thus scalable to higher number of antennas. Further, modifications can be done with multiplication unit by using fast multiplier techniques involved in Vedic mathematics to improve the throughput and reduce the delay.

REFERENCES

[1] "A 675 Mbps, 4 x 4 64-QAM K-Best MIMO Detector in 0.13 µm CMOS" Mahdi Shabany and P. Glenn Gulak, IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 20, No. 1, January 2012

[2] Chung-An Shen And Ahmed M. Eltawil,(2010) "A Radius Adaptive K-Best Decoder With Early Termination: Algorithm And VLSI Architecture", IEEE Transactions On Circuits And System-I: Regular Papers, Vol. 57, No. 9.

[3] Chen S and Zhang T,(2007) "Low power soft-output signal detector design for wireless MIMO communication systems," in Proc. Int. Symp. Low Power Electron. Design, pp. 232–237.

[4] Chen S, Zhang T, and Xin Y,(2007) "Relaxed K-best MIMO signal detector design and VLSI implementation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 3, pp. 328–337.

[5] Li Q and Wang Z,(2006) "An improved K-best sphere decoding architecture for MIMO systems," in 40th Asilomar Conf. Signals, Syst. Comput., pp. 2190–2194.

[6] Guo Z and Nilsson P,(2006) "Algorithm and implementation of the K-best sphere decoding for MIMO detection," IEEE J. Sel. Areas Commun.,vol. 24, no. 3, pp. 491–503.

AUTHORS

First Author – Akila. V, Assistant Professors, Department of ECE, Surya Engineering College, Erode-638107., Email: vakilavenkatachalam@gmail.com

Second Author – Jayaraj. P, Assistant Professors, Department of ECE, Surya Engineering College, Erode-638107., Email: jayece015@gmail.com