Staggered Sphere Decoding for MIMO detection

Pankaj Bhagawat, Gwan Choi (<pankajb, gchoi>@ece.tamu.edu)

Abstract – **MIMO system is a key technology for future high speed wireless communication applications. At present MIMO systems are constrained by efficient detection algorithms/hardware architectures. In past much attention has been paid to either Sphere Decoder (Depth First Search), or K-best algorithm (breadth first search. Very little attention has been paid to architectures that combine these two approaches. In this paper we present an algorithm and architecture that takes a hybrid approach toward MIMO detection. Our results show that proposed architecture has advantages of BFS and DFS based architectures, and its complexity is estimated to be intermediate to that of Sphere Decoder and K-best architecture.**

Index Terms – **MIMO systems, K-best algorithm, Implementation/Time complexity, Sorting, Hybrid architecture.**

I. INTRODUCTION

Recently, multiple input-multiple output (MIMO) wireless systems have been receiving great deal of attention in academia. Industry is also looking at it as a viable solution for future high data rate communication systems. From the industry point of view, however, MIMO systems pose a major problem as far as implementation complexities of the concerned algorithms are concerned.

One of the major issues in implementing a MIMO based system is the very high complexity of the detection algorithm at the receiver. Among the earliest algorithms used for signal detection for un-coded MIMO was VBLAST [1]. It is based on decision feedback (DF) detectors, which use successive cancellation to detect independent spatial data streams. Although computationally efficient, VBLAST suffers from substantial error performance degradation.

The other family of algorithms under active consideration these days relies on tree based search. Two of the most notable among them is the Sphere Decoding (SD) algorithm [2], which is a depth first search (DFS) based algorithm, and K-best algorithm which is a breadth first search (BFS) based algorithm [4]. Both have their merits and demerits. For instance, SD algorithm takes smaller area than K-best algorithm but has variable throughput leading to problems in integrating it into the overall communication system [ref]. Tree based algorithms in general seems to offer good trade off between performance and complexity issues [3]. For this reason, we address these issues related to algorithms based on tree search, and eventually propose a new hybrid version of the algorithm/architecture which has some of the most desirable qualities of BFS and DFS based architectures.

This proposal is organized as follows. In section II we describe the channel model and review some of the existing detection algorithms. In section III we provide intuition behind the proposed algorithm, and present a practically feasible way of implementing it. In section IV we compare, using simulation, overall performance of the proposed algorithm/architecture with K-best and SD based detection algorithm. Finally, we conclude the paper by identifying future research directions in the realm of proposed algorithm and its architecture.

II. MIMO DETECTION UNDER FLAT FADING

A. *Channel model*

Consider a symbol synchronized MIMO system with M transmit and N receive antennas. This means M independent signals are transmitted and arrive at an array of N receivers, this, is known as spatial multiplexing. If we group all signals into vectors then the system can be viewed as transmitting an $M\times1$ vector $\mathbf{s}$ through an $N\times M$ channel matrix $\mathbf{H}$. Also, each receiver will have its own noise source (assumed Gaussian). Thus, the overall baseband system model can be mathematically represented as

$$\mathbf{x}=\mathbf{H}\mathbf{s}+\mathbf{n} \qquad (1)$$

Where $\mathbf{x}$ is $N\times1$ received vector and $\mathbf{n}$ is $N\times1$ noise vector. The $(i,j)^{th}$ element, h_{ij} , of the matrix H denotes the complex channel response from j^{th} transmit antenna to i^{th} receive antenna, $\mathbf{n}$ is zero mean gaussian vector with a covariance matrix of $\mathbf{R_n}=E\{\mathbf{n}\mathbf{n}^*\}=\sigma^2_{\mathbf{n}}\mathbf{I}$.

The objective of the MIMO detection algorithm is to compute an estimate such that:

$$\hat{s} = \arg \min_{s \in \Omega^Q} \| x - H s \|^2 \quad (2)$$

Where Ω is set of complex entries from the QAM constellation and Q is the size of QAM constellation.

B. Maximum likelihood detection(ML):

ML detection gives the best possible solution to (2) by exhaustively computing the distance for each combination of input vector and choosing the one that minimizes RHS of (2). There are implementations showing that it is possible to compute exhaustively and find ML solution for 4×4 QPSK system [5]. However, this approach quickly becomes impractical as *M*, *N* or size of constellation grows beyond 4. For instance, 4×4 16-QAM system will involve 16^4=65536 combinations of symbols to be searched, clearly an impractical task.

C. Sphere decoding(depth first)

One way to circumvent exhaustive search to find ML solution is to evaluate only a small subset of transmitted vectors. Cost function given by (2) can be rewritten as (3)

$$J(s) = \| \mathbf{x - Hs} \|^2 = (\mathbf{s} - \hat{\mathbf{s}})^H \mathbf{U}^H \mathbf{U}(\mathbf{s - \hat{s}}) + \mathbf{C} \quad (3)$$

$$\mathbf{s} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \mathbf{x} \quad (4)$$

$$\mathbf{U}^H \mathbf{U} = \mathbf{H}^H \mathbf{H} \quad (5)$$

Constant C in (3) can be neglected since it has same value for all vectors. The sphere algorithm was originally proposed in [2], as an algorithm to find shortest vector in a lattice. Its application to MIMO systems appear in [6]. More discussion on implementation issues can be found in [7].

The key to the spherical decoding is to reformulate the cost function into a summation over the individual transmit antenna. In ML algorithm one has to compute cost jointly considering all transmit antennas, whereas, in spherical decoding we can consider the cost in incremental fashion. This can be utilized to prune the less promising combinations earlier on, thereby, reducing the computational complexity. To accomplish this, the summation in the cost function is forced to be a positive quantity. This implies that at each level of the tree the cost is assured to be strictly non-decreasing.

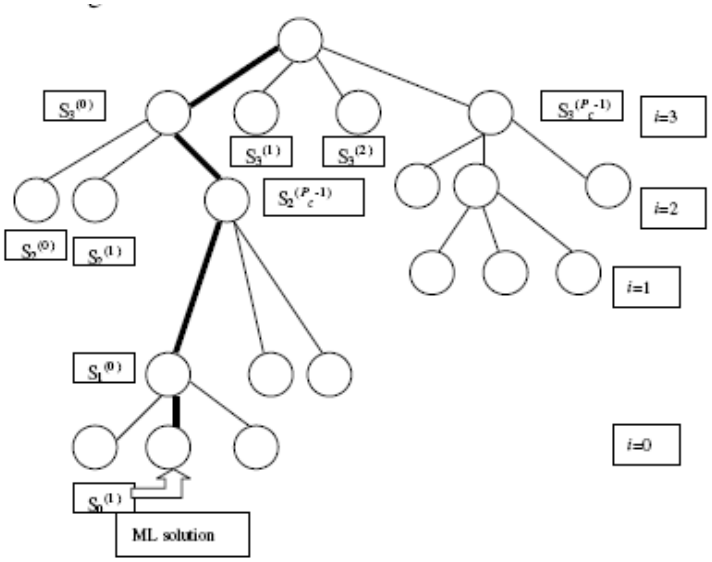

Fig.1 Tree structure for spherical search

Equation (3) can be rewritten to form summation across each transmit antenna.

$$J = \sum_{i=Q-1}^{0} T_i \le r^2 \quad \text{, where}$$

$$T_i = | u_{ii} \cdot (s_i - \hat{s}_i) + \sum_{j=i+1}^{Q-1} u_i \cdot (s_i - \hat{s}_j) |^2 \quad (6)$$

$$T_{inner} = \sum_{j=i+1}^{Q-1} u_i.(s_i - \hat{s}_j) \quad (7)$$

Equation (6) corresponds to evaluating the contribution of each of the transmit antennae in a cumulative manner instead of jointly as was done in (2). The fact that **U** is upper-triangular ensures that each term in the summation depends only on the current level decision, as well as the history of the path to reach that level in the tree. The T_{inner} sum represents the cost of path history to the current node. Therefore, while T_i must be evaluated for each potential symbol in the candidate tree, the T_{inner} term is shared among all the constellation points. Appropriately setting the radius, *r*, (which defines the "sphere" around the search center) restricts the number of nodes visited in the search. For example, consider the tree in Fig.1, which represents the search algorithm for a four transmit antenna system with Q- QAM modulation. With 4-QAM P_c has value 4. The first level in the tree corresponds to making a decision on transmit antenna Tx_i where i=3. Once the symbol has been selected, the cost function on the second level can be evaluated by adding in the additional cost of a second antenna. At each step, the cost function is compared across a radius value, which is chosen such that only candidates that are close to ML solution are retained. Removing a potential symbol from the top level removes $64(4^3)$ candidates from the search space.

From implementation point of view, however, spherical algorithm has certain demerits. This algorithm relies on depth first search and hence is recursive in nature. It generally takes more hardware resources to build recursive systems. Also, bit-error rate (BER) performance of the algorithm is critically dependant on the choice of decoding radius. If high radius is chosen more number of nodes on the decoding tree will have to be processed. Too low a value of radius will result in restarting the decoding process all over.

D. K-Best algorithm(BFS)

Broadly speaking, breadth-first tree search algorithms extend all the survivor paths at each tree depth at once, and keep only best K node at each level of the tree. This is repeated till we arrive at the leaf nodes.

The K-best algorithm searches for the solution in the forward direction only. K best candidates are kept at each level of the tree. For instance, consider Fig. 1 again, in the first step of the algorithm all the nodes at level i=3 will be evaluated. Next, depending on the value of K chosen, algorithm will pick K-best (nodes with minimum cumulative metric) nodes to be considered as parent nodes for next level. In next step it will compute cost of all children of nodes kept in the last level and then pick new set of K best nodes from the current level. This continues till lowest level of the tree is reached, where we pick node with the lowest cost, which is declared as the received symbol vector. Since K-best algorithm computes constant number of nodes at all times, it results in a constant decoding throughput. The BER performance of the K-best algorithm is expected to be close to that of the ML if K is sufficiently large [4]. However, limiting the value of K can reduce the complexity of the breadth-first algorithm. Usually some kind of trade off can be reached in deciding the value K.

There are many reasons for reduction in throughput with increasing K. Some of the most important ones are: First, the number of nodes to be visited is increased; this also means that we have to sort more nodes at each level, and hence the number of clock cycles required goes up with K. Also, the length of the critical path in the circuit increases almost linearly with K [8], limiting the clock frequency. Implementation of sorting algorithms takes good amount of resources, and is inherently serial in nature. Authors in [4] suggest using bubble sort as one of the means to pick K best nodes. Sorting being serial in nature further slows down K-best algorithm with increasing K. To pick K nodes out of QK nodes at each level will require $QK(\log_2 QK)$ or QK^2, whichever is minimum. Although, as suggested in [4] a pipelined architecture can be used to yield amortized cost of 1 for sorting algorithm, this will lead to increased chip size as more number of stages now has to be implemented.

III. PROPOSED ALGORITHM/ARCHITECTURE

The proposed algorithm and its architecture is a hybrid between K-best and DFS based sphere decoder (DFS-SD). Herein, we show that the proposed architecture not only requires lesser number of clock cycles than the DFS-SD and K-best algorithm, but also has relatively constant throughput across a wide range of SNRs.

A. Procedure to enumerate QAM symbols

For illustration purposes in Fig.2 we show a part of QAM constellation. The region around each QAM point was divided in 8 parts, shown as numbers 1 through 8. 10000 points were randomly generated in each of these regions and the distances of these points from each QAM symbol in the constellation were computed, these distances were then sorted in ascending order. This procedure was repeated for every

QAM symbol. Thus, a total of 8*16 =128 regions were evaluated. We observed that under L^{∞} [ref] norm the sorted sequences always have a unique relationship with the region. In other words if we know the region then we can uniquely identify the enumerated (sorted) sequence. This way we completely avoid any kind of sorting.

These enumerated sequences are stored in a LUT as shown in Fig.2

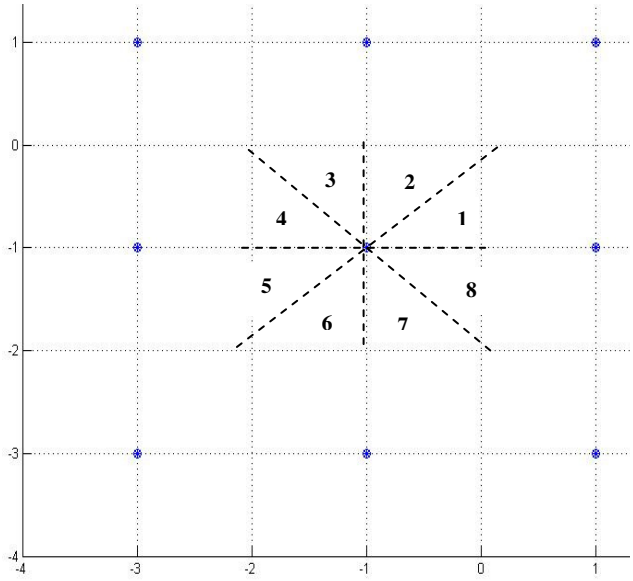

Fig. 2 Enumeration Technique

B. Architectural data flow

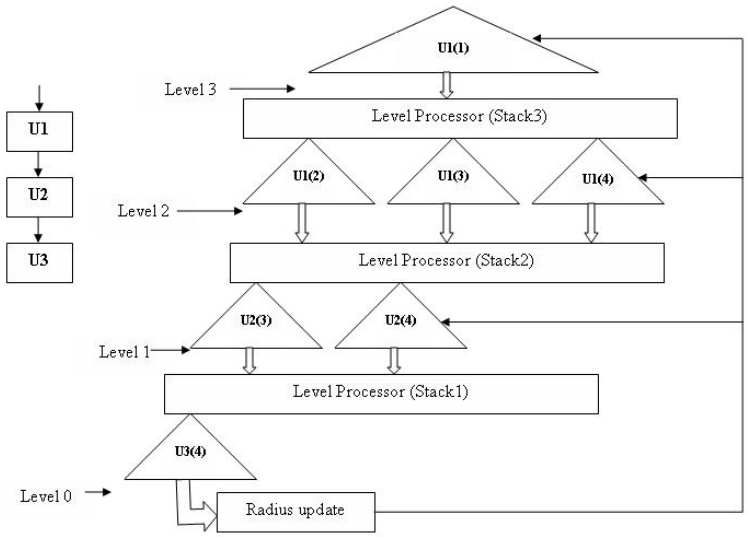

Fig. 3 High Level Architecture

The data flow for the proposed architecture is shown in Fig. 3. U1, U2, and U3 (which will be called node processors) takes ancestry of a node as input, generates list of children nodes in sorted order, and stores them in a stack (Fig.4). The box (dotted) in the right part of Fig. 4 shows what we refer to as a level processor, which operates over multiple "triangles" at each level of the tree (see Fig. 3). A level processor consists of a multi-port stack, an array of distributed metric computation units, and finally a set of tree comparators to find minimum of the metrics stored in the stack. Final output of at each level is the complete ancestry of the node with minimum metric. This information is then fed to the node processor at next level. As shown in Fig. 3 all three node processors operate concurrently in a pipelined fashion. U2(4) means U2 is operating at time instance 4. It should be noted, however, that the pipeline operates over "triangles" as shown in Fig. 3 and hence we have pipelined the recursive depth first search for a given

received vector, and we call this intra vector symbol pipelining. In the next sub-section we describe the functionality of various blocks used.

C. Details of various blocks of the architecture.

- Node processor (NP): This block takes in the complete lineage of a node **P** as input, and outputs a sorted list of children of **P** if the minimum of children nodes is less than the current radius.
  1. Find Region/Find Minimum: This unit computes the region of the received point at a particular level. One of the key contributions of this work is the observation that given the region where the point lies there exist a unique sorted sequence. These sequences are stored in a LUT.
  2. Comparator: This comparator compares the minimum of the children nodes with the current estimate of the radius. This is necessary as we don't want to store children of a parent node that are all greater than the current radius.
  3. Enumerated Sequences LUT: This is a look up table that stores the enumerated children (in sorted fashion). As mentioned earlier it is our observation that under L^{∞} norm there is one to one relationship between the region and the sorting order of nodes of a particular parent node.

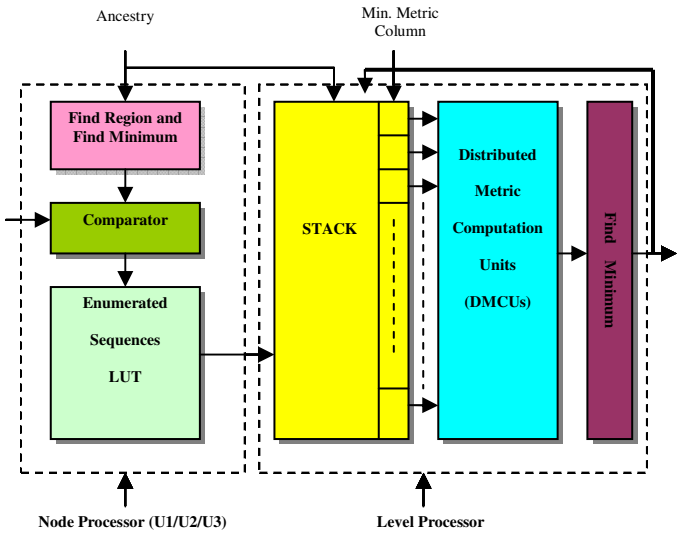

Fig. 4 Architectural Details

- Level Processor (LP): This block operates over all the surviving nodes at a particular level.
  1. Stack: It takes the sorted list of children and stores them in a stack. One row of stack is devoted to storing the sorted list of children of a particular parent node (along with its complete lineage).
  2. Distributed Metric Computation Unit: This block consists of MCUs that takes in the complete lineage of a particular node and generates the associated cumulative metrics.
  3. Find Minimum: This block takes in the cumulative metrics as input and outputs the minimum. This is then fed to the NP at next level. Since we arrange comparators in a tree fashion, height of the stack (over which it searches for the minimum) is so particular importance. For instance if the maximum height of stack is 16 then we need to use 4 stages of comparators. This parameter is important because of multistage comparators used can contribute significantly to the critical path of the overall circuit.

IV. RESULTS

In this section we present some of the architectural parameters like size of the stacks, average number of clock cycles etc. We simulated a 16-QAM 4×4 MIMO system under quasi-static flat fading channel.

Fig. 5 shows the mean and maximum size of stacks 3 and stack 1 over a range of SNR. Fig 7 shows average number of clock cycles required for various SNRs.

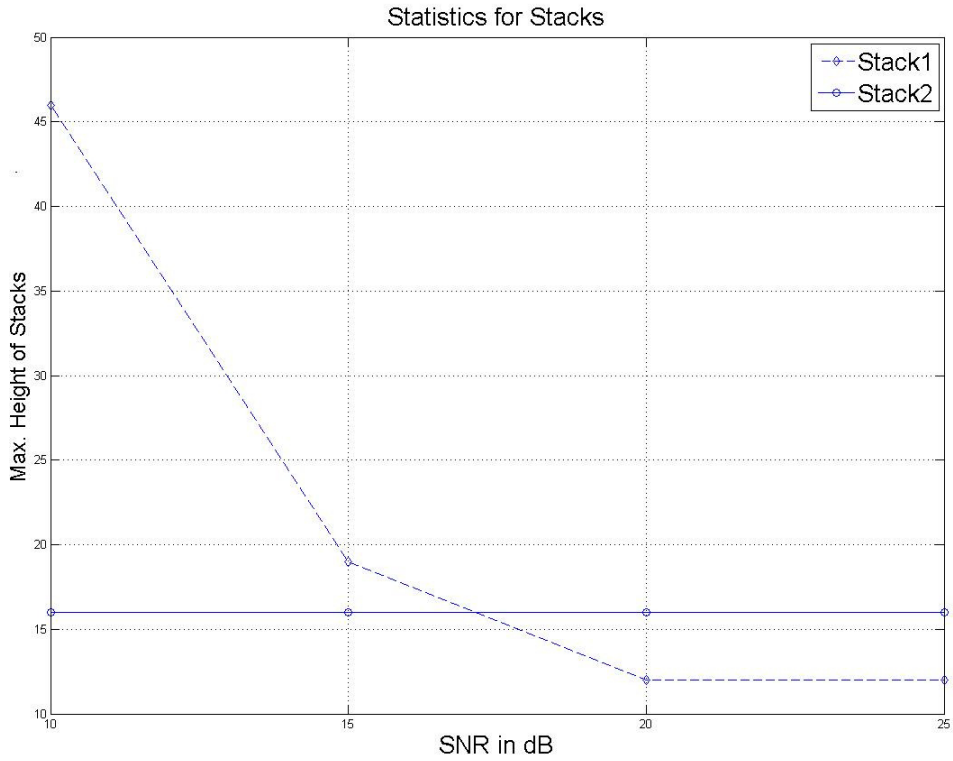

Fig. 5 Height of Stacks

In Fig. 5 we show the maximum size of stack1 and stack 2. As can be seen stack2 has maximum size of 16, and hence we would need a 4 stage tree comparator to compute the minimum (purple box in Fig.4). Also, at 10 dB maximum height of stack1 is close to 50, and this would require $\log_2 50$ or 6 stages of comparators. But we noticed that mean height of stack 1 is only around 3, and height of stack1 is less than 16 for 99.75% of times (Fig. 6). This means that if we fix the size of stack1 to 16 and discard the candidate nodes whenever stack1 is full, we will not degrade the performance too much since probability of stack1 overflowing is only 0.25% or 0.0025. Maximum height of stack 3 will always be 1, since it will be written into only once.

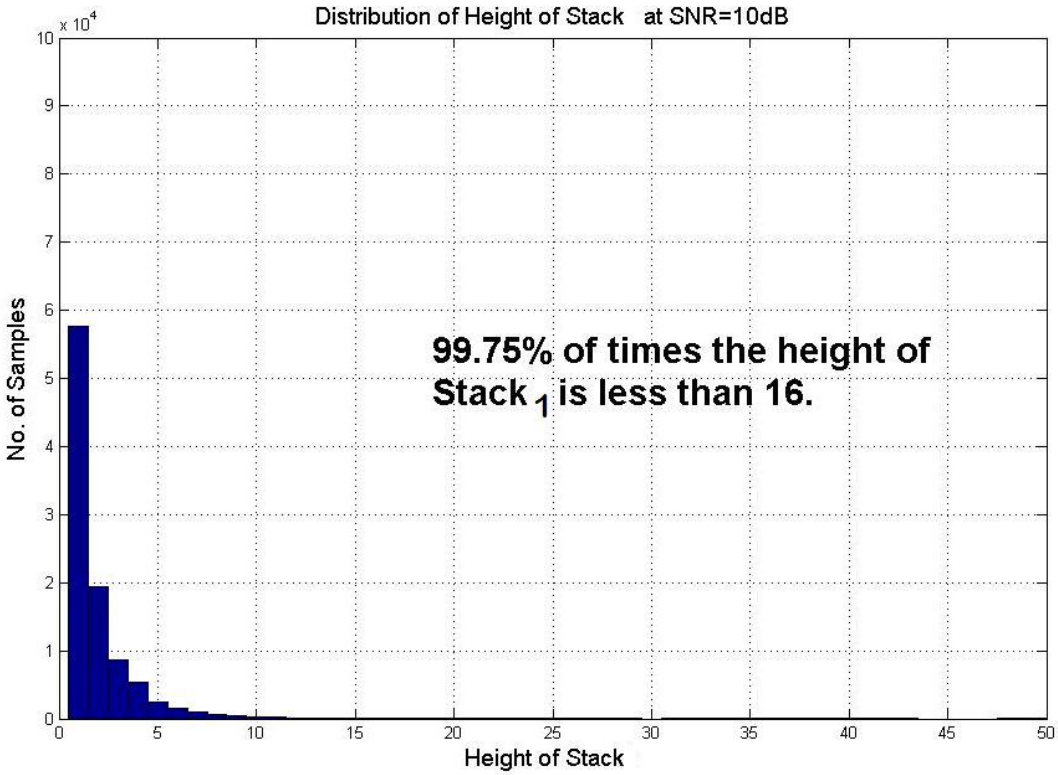

Fig. 6 Statistics for Stack1 (level 1)

In Fig. 7, we show the average (and variance) number of clock cycles taken to complete detection of one vector symbol. As can be seen the average number of clock cycles needed is fairly constant over a wide range of SNRs. Also, the variance of number of clock cycles is small at a given SNR.

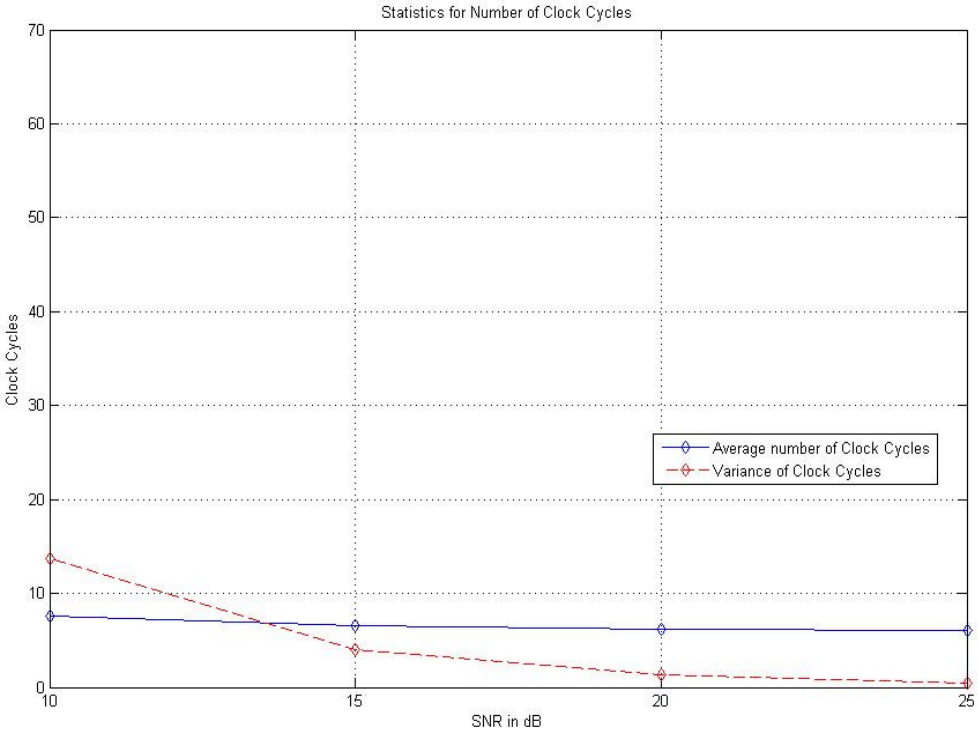

Fig.7 Number of Clock Cycles.

Comparing this to [8] and K-best [4], we find that the proposed architecture has several advantages:

- Presented algorithm/architecture does not use an explicit sorting network (unlike K-best algorithm), and still has desired properties of the breadth first search. In that, it has a fairly constant throughput across a wide range of SNR.
- Due to absence of a sorting unit it takes less number of clock cycles as compared to K-best (assuming bubble sort with one clock cycle per comparison).
- Due to small variance at a particular SNR, integrating this design in ASIC is much easier since less buffers would be needed.

V. FUTURE WORK

So far we have explored an "intra-vector" pipelined architecture for the sphere decoding problem. However, in order to achieve still higher throughput we will explore "inter-vector" pipelining along with the present approach.

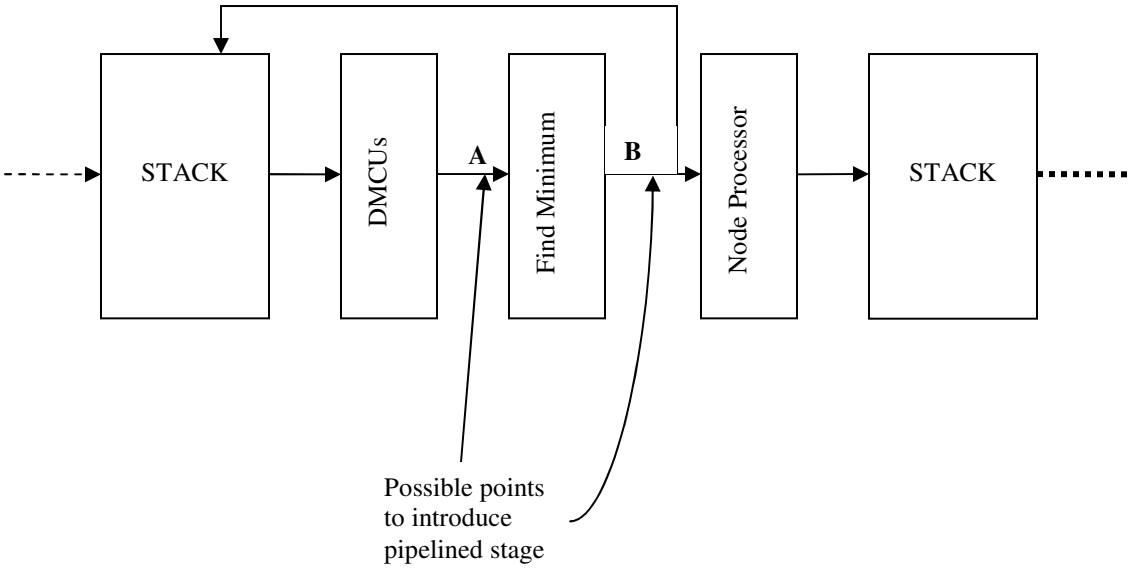

Fig.8 An "inter vector" pipelining scheme

Fig. 8 shows a "inter vector" pipelining scheme. Two stages of pipeline can be introduced, first at point **A** and second at point **B,** hence there will be two vectors being processed simultaneously**.** Since, number of clock cycles taken to detect a vector symbol is a random variable effective number of clock cycles needed to process these two vectors will be $\max(clk_1,clk_2)/2$, where clk_1 and clk_2 are number of clock cycles needed for vector 1 and vector 2 respectively. Processing of two vectors at a time would also require stacks of double the size.

Another direction which we will be going in, will involve feasibility study of present architecture for higher order modulation schemes like 64 QAM, and more number of transmit and receive antennas (e.g. 6x6 or 8x8 MIMO systems). This architecture with real valued decomposition will also be evaluated.

REFERENCES

[1] P.W.Wolniansky, G.J.Foschini, G.D.Golden, and R.A.Valenzuala, "V-BLAST:An architecture for realizing very high data rates over the rich-scattering wireless channel," *Proc. IEEE ISSSE 1998,*pp.295-300,Sept. 1998.

[2] U. Fincke and M. Pohst , " Improved methods for calculating vectors of short length in a lattice, including complexity analysis," Math. Compute., vol.44,pp463-471, April 1985.

[3] Burg, A.,Borgmann, M., Simon, C., Wenk, M.,Zellweger, M.,Fichtner,W.,"Performance tradeoffs in the VLSI implementation of the sphere decoding algorithm,: Fifth International Conference on 3G Mobile Communication technologies, pp. 93-97,2004.

[4] Kwan-wai Wong, Chi-ying Tsui, Cheng, R.S.-K., Waiho Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," *Proc*. IEEE ISCAS, Vol. 3, pp273-276.

[5] A.Burg, N.Felber, and W. Fichtner, “A 50 Mbps 4×4 maximum likelihood decoder for multiple-input multiple output systems with QPSK modulation,” in Proc. IEEE Int. Conf Electron., Circuits, Syst.(ICECS),vol.1,2003,pp.332-335.

[6] E.Viterbo and J.Boutros, “A universal lattice decoder for fading channels,” *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639-1642, July 1999.

[7] D.Garrett, L.Davis, S.ten Brink, B Hochwald, and G. Knagge,”Silicon complexity for maximum likelihood MIMO detection using spherical decoding”, *IEEE J. Solid State Circuits*, vol.39,pp 1544-1552,2004.

[8] Burg, A., Borgmann, M., Wenk, M., Zellweger, M., Fichtner, W., Bolcskei, H.,”VLSI implementation of MIMO detection using the sphere decoding algorithm” IEEE J. Solid State Circuits, vol.40, pp 1566-1577, July 2005.