IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Design of High Throughput K-Best For MIMO Detector

K.Gowri[*1], N.Rajeswaran[2]

[*1,2]Department of ECE, SNS College of Technology, Coimbatore, India

gowrik2010@gmail.com

Abstract

Multiple -Input -Multiple -Output(MIMO) wireless system have been widely used in next generation wireless systems like 802.11n, Long Term Evolution(LTE) and WIMAX. Next generation wireless system has the demand of high throughput processing. An efficient high throughput pipelined based VLSI architecture for a hard output MIMO detector using K-Best lattice detector has been proposed. The key contribution is a mean of expanding the intermediate nodes of the search tree operating in the pipelined architecture. The proposed pipelined architecture will have a fixed critical path for processing. The K-Best scheme is one amount the search trees which improve the performance and reduce the computational complexity. The combined expansion and sorting cores are able to find the k-best canditates in K clock cycle. The architecture is programmed using verilog 2001 in xilinx EDA tool and it simulated in Modelsim 10.1 simulator.

Keywords: k-best detectors,Multiple-Input-Multiple-Output(MIMO),WiMAX systems.

Introduction

Due to the high spectral efficiency, multiple-input–multiple-output (MIMO) systems have attracted significant attention as the technology of choice in many standards such as IEEE 802.11n, IEEE 802.16e, IEEE 802.16m and the long term evolution (LTE) project. One of the main challenges in exploiting the potential of MIMO systems is to design low-complexity high-throughput detection schemes with near maximum-likelihood (ML) performance that are suitable for efficient very large scale integration (VLSI) realization.

MIMO technology increases the data rate just by using multiple transmit and receive antennas all working at the same frequency and without using additional transmit power. 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. However, they suffer from significant performance loss.

The other alternative is to use near-optimal non-linear detectors. Depending on how they carry out the non-exhaustive search, near-optimal non-linear detection methods generally fall into a few main categories, namely *depth*-first search, *breadth*-first search, and *best*-first search. Depth-first sphere decoding is one of the most attractive depth-first approaches whose performance is optimal under the assumption of unlimited execution time. However, the actual runtime of the algorithm is dependent not only on the channel realization, but also on the operating signal-to-noise-ratio. Thus leading to 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 most well-known approach is the *K*-best algorithm.The *K*-best detector guarantees a SNR-independent fixed-throughput with a performance close to ML. It also has fixed critical path delay independent of the constellation order, K value, and the number of antennas. Moreover, it efficiently expands a very small fraction of all possible children in the *K*-best algorithm and can be applied to infinite lattices. Finally it provides the exact *K*-best solution, i.e., the solution that implements the original *K*-best algorithm with all needed expansions.

K-Best Algorithm

The optimal solution problem can be considered as a tree-search problem with 2Nt levels. In fact, the *K*-best algorithm explores the tree from the root to the leaves by expanding each level and selecting the K- best candidates with the lowest path metric in each level that are the surviving nodes of that level. Partial Euclidean Distance (PED) denotes the distance increment between two successive nodes/levels in the tree.The size of this exhaustive expansion grows significantly when the constellation size is scaled upward. Therefore, better ways are needed to calculate the K-best candidates of each level without performing an exhaustive search. Regardless of whether dealing with the hard-decision or soft decision output, there are two main computations that play critical roles in the total computational complexity of the algorithm namely, 1) the expansion of the surviving paths, and 2) the sorting.

1) *Expansion*: An efficient expansion method called on-demand expansion scheme, which avoids the exhaustive enumeration of the children while providing all the information required for the exact *K*-best implementation with no performance degradation, which to the best of our knowledge, is the only expansion scheme to-date with a computational complexity proportional to the K value and independent of the constellation size.
2) *Sorting:* A distributed sorter, working in a pipelined structure with the on-demand expansion scheme, which finds the K best candidates in K clock cycles. It works for any value of K and independent of the constellation size. It does not compromise the BER performance and provides the exact *K*-best solution, and can be easily extended to the complex domain.In the K-best algorithm,a sort operation takes place to select the first *K* candidates with lowest PEDs.

Proposed Scheme

One of the main elements of our proposed scheme is to find the children of each node on-demand and in the order of increasing PED rather than calculating the PED of all the children exhaustively. In other words, the key idea of the proposed distributed *K*-best scheme is to find the first child(FC) of each parent node.Among these first children the one with the lowest PED is definitely one of the *K*-best candidates. That child is selected and replaced by its next best sibling.This process repeats times to find the *K*-best candidates in level . The same procedure is performed for each level of the tree.

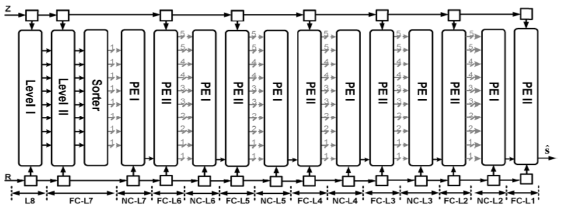

Figure 1: Pipelined VLSI Architecture Of The *K*-Best Algorithm

One challenge that needs to be addressed to achieve a highthroughput architecture is the implementation of K minimizations in each level, which is still computationally complex even with state-of-the-art bubble sorting. In order to resolve this problem, 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.

There are 8levels in the tree. The 8th level of the tree, corresponding to the last row, opens up all the possible values , and calculates their corresponding PEDs. The output of this stage k8 is resulting in 8 PED values, which is performed by Level I. For each of the nodes in , the first child is found and its PED is updated using the FC-Block in Level II. FC with the lowest PED should be determined, which requires all the FCs to be sorted. This is done using the Sorter block which sorts all eight resulting PEDs in four cycles.

Using this sorter, the number of clock cycles required for sorting is half as much as that of the classic bubble sorting. The key idea that makes this sorter faster, is the implementation of two tasks (max/min and the data exchange) in 1 clock cycle through the intermediate registers. The output of the Sorter block is the sorted FCs of level 7, that are all loaded simultaneously to the next stage (i.e., PE I). Generally speaking,in each level, 1 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.

The task of the PE I block is to take the FCs of each level as an input and generates the *K*-best list of that level one-by-one. The child node in level 7 with the lowest PED is definitely one of the *K*-best candidates in level 7. This value is passed to the PE II block. Upon the removal of this FC, its next sibling needs to be calculated, which is done by the core called NC-Block in the feedback loop of the PE I block. The PED of this sibling needs to be compared with the other FCs, already present in the NC-L7 stage. The next *K*-best candidate has the lowest PED among this new set. This process is repeated 10 times (taking 10 cycles) until all the *K*-best values of the second level of the tree are generated and passed to the PE II block in FC-L6.

LEVEL-I: The input to Level I is r^{88} and z7 its output is the PEDs of all the elements in , which are the nodes in the 8th level of the tree. The detail of the architecture is shown in Fig.2. It employs a 13-bit x 13-bit multiplier, a few adders and the absolute value block. Note that the absolute value block, representing the l^1-norm, can be replaced by a squaring operation block, l^2-norm, which can be easily implemented using a carry-save-adder technique. However, simulation results show that the difference in the BER performance is negligible Since Level I is on the feed-forward path of the architecture, a fine-grained pipelining technique can be employed inside the block in order to increase the system throughput.

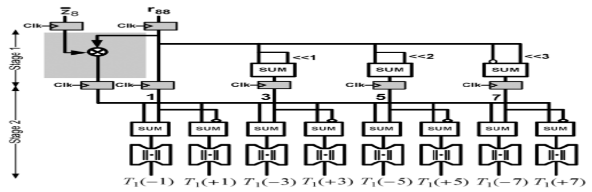
Figure 2: Architecture of level I

A 2-stage pipeline is employed in Level I, which is shown by 2 and 5 positive-edge-triggered flip-flops (FFs)/registers added in stage 1 and 2, respectively. In fact, by using the FFs the block is broken down to 2 consecutive stages, which avoids a long critical path and implies that the critical path of the architecture contains only 1 multiplication. It is assumed that all the FFs used in this paper are triggered by the positive-edge of the clock.

LEVEL-II: The input to Level II is the PED values of the 8th level and its output is the PED values of the first children in the 7th level of the tree. In fact, in the Level II block, the first children of the eight nodes in the 8th level are determined. Note that due to the structure of the R matrix in, the first children in the 7th level of the tree are all the same and independent of their parents in level 8. This child is determined and is used to calculate the updated PED values of the nodes in the 7th level. Since $r^{77} = -r^{88}$, no extra input is required for the calculations in Level II.

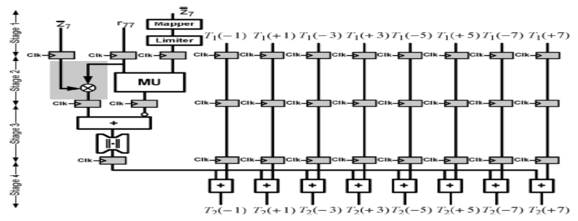
Figure 3: Architecture of level II

The architecture of Level II block is shown in fig 3. Once the first child was determined it is multiplied by r^{77}using the MU block. The input normalized $z_7$value is also multiplied by r^{77} after which the Euclidean distance between the first child and the received vector is calculated and the result is added to the PED values of the 8th level PEDs to derive the eight updated PEDs of the 7th level. The fine-grained pipelining technique has also been employed in this block to break it into four stages in order to limit the length of the critical path.

SORTER: The input to the Sorter block is the set of eight PED values of the 7th level FCs and the main task of this block is to generate the sorted list of these PED values. The architecture of the Sorter is shown in Fig. 4. The eight inputs are denoted by D0-D7 , and the outputs are stored in eight registers shown by grey flip-flops labeled by letter "N". The Ctrl signal is used to load the data in 1 clock cycle. Using this architecture, it takes four clock cycles to sort all the eight PED values.

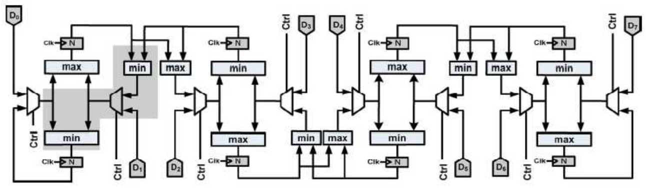
Figure 4: Architecture of Sorter block

This architecture can be used as a general sorter, which sorts K numbers in K/2 clock cycles because it implements two tasks (max/min and the data exchange) in one clock cycle through the introduction of intermediate registers. One such set of consecutive minimizations is highlighted. The main task of the sorter in this block is to receive a sorted list and to find the correct position of a new entry in the sorted list, while announcing the entry with the lowest PED every clock cycle.

PE-I BLOCK: PE I is a general block used for all the levels from level 7 to level 2. It receives the sorted list of the first children of each level and generates the best candidates of that level. For instance the output of the PE I in level 7, called NC-L7 in Fig.5 is the K=10 consecutive best candidates with the lowest PED values in level 7, generated one-by-one in series at the output.It consists of a sorter, and a block called NC-Block on the feedback path.PE I receives the sorted list of the PEDs from the preceding stage. It finds the best one with the lowest PED and announces it as the next *K*-best candidate at the output, and then calculates the next best sibling of the announced child through the NC-Block and feeds it back to the sorter to locate the correct location of the new sibling in the already sorted list in PE I.

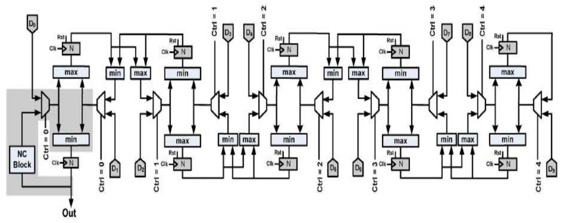
Figure 5: Architecture of PE-I block

Before the sorted PED values of the preceding stage are loaded into the PE I block, there is a reset signal, Rst that initializes all the register banks (except the one attached to the output) to the maximum possible number. This is necessary to avoid any interference from the previous values stored in them and makes them ready to process the new list. The Rst signal also initializes the control signal, Ctrl to zero, which is used to load the sorted list from the preceding stage to the PE I block.The critical path of the PE I block is highlighted.It contains a MUX, a comparator and the NC-Block. The main task of the NC-Block is to determine the next best sibling of an already announced best child. It also finds the PED value of this sibling and sends the information to the sorter in the PE I block.

PE-II BLOCK: The output of PE I is the serial list of *K*-best candidates of the current level, generated one-by-one at the output. As each of the *K*-best candidates is generated, it is sent to the PE II block to calculate the first children of the next level and sort them as they arrive. The architecture of the PE II block is shown in Fig.6, where D_{in} is the input port and D0-D9 are the output ports. The first child of the *K*-best candidate of the previous stage and its updated PED value are calculated by the FC-Block, and then using the sequential sorter, the calculated PED values are sorted. The PE II block is connected to the output of the PE I block in a pipelined fashion. In the proposed architecture for PE II, the sorted PEDs are stored in the register banks, depicted by N -bit registers and denoted by RB0-RB8. At every clock cycle, 2 register banks are updated at the same time. Once the last element enters the sorter, it updates the first two register banks, thus the first two are guaranteed to have the 2 smallest PED values.

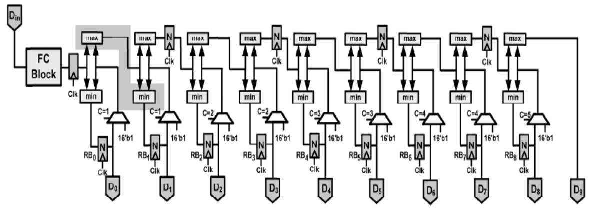

Figure 6: Architecture for the PE II block

Therefore, at the next clock cycle, they can be transferred to the following PE I block. After the second clock cycle, the next 2 register banks are updated. Therefore, the PED values are transferred to the next level on a pair-by-pair basis.Once the last element comes in and the first two register banks are sent to the next stage, the internal min/max functions should be initialized to the highest positive number to avoid the comparison between the first element of the next iteration and the last element of the current iteration.This makes the core utilization 100% as PE I and PE II are fully pipelined with zero latency.

Simulations and Results

A. Modelsim Environment

The proposed design has been simulated in MODELSIM for the further study of the design.

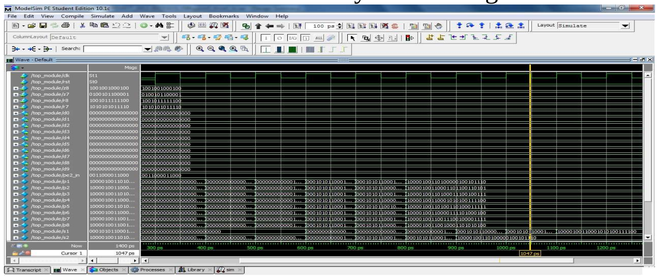

Figure8: Output Waveform Of Pipelined VLSI Architecture

The above figure is the output waveform of Pipelined VLSI Architecture and generates the smallest number of visited nodes, as well as the highest achieved throughput.

Conclusion

Employing multiple antennas at transmitter as well as both transmitters and receiver sides enables considerable performance enhancement in wireless communications systems.A high-throughput *K*-best algorithm suitable for high-order constellation schemes, which has the smallest number of visited nodes. The key innovation are the introduction of an on-demand expansion and distributed sorting scheme operating in a pipelined fashion. The presented reduced constellation search method results in optimal performance while significantly reducing the symbol detection complexity. Multi-Input Multi-Output (MIMO) systems will form the core for wireless communications standards like IEEE 802.11n, IEEE 802.16e and LTE,WiMax.

Future Enhancements

The adders of K-Best MIMO detector can be replaced by fast adder as square root carry select adder inorder to achieve high throughput. These approaches can be produce an approximate solution with lesser complexity.

References

[1] Mahdi Shabany, Glenn Gulak P, "A 675 Mbps, 4x4 64-QAM *K*-Best MIMO Detector in 0.13 µm CMOS" IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 20, No. 1,January 2012.

[2] Burg A, (2005)"VLSI implementation of MIMO detection using the sphere decoding algorithm," IEEE J. Solid-State Circuits, vol. 40, no.7, pp. 1566–1577.

[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] 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.