Abstract— In MIMO communication systems, K-best decoding algorithm achieves near optimal performance with reduced complexity. Simulation results show that a configurable MIMO detector can improve system performance over a wide range of operating conditions. In this paper we present a novel configurable architecture for implementation of K-best algorithm. The proposed architecture is fully parallel and can support QPSK, 16-QAM and 64-QAM modulation schemes for a range of \( K \) values.

I. INTRODUCTION

Multiple-input multiple-output (MIMO) communication systems benefit from multi-path propagations to improve the overall bit error ratio (BER) and data rate in the communication link [1]. In a typical MIMO communication system comprising \( N_t \) transmitter and \( N_r \) receiver antennas, each of the \( N_r \) receivers receives both line-of-sight and reflected propagation components from each of the \( N_t \) transmitters. The role of the MIMO detector is to use the received signals at each of the \( N_r \) receiver antennas and to recover the transmitted symbols [2].

Maximum-likelihood (ML) detector is the optimum detection algorithm in MIMO channels. Due to high level of computational complexity in ML detectors, lattice-based decoding algorithms, such as the K-best decoding algorithm have been proposed. The design of low complexity MIMO detectors that are flexible in terms of number of supported antennas and modulation types is a challenging task. There is also a need for MIMO detectors that are able to dynamically adapt to the requirements of channel and transmission conditions. In a K-best MIMO detector throughput, latency and BER are related to \( K \), which is a fixed detector parameter.

In this paper, we will discuss the need for K-best MIMO detectors that can adaptively change \( K \) to improve system performance. We will later introduce a configurable VLSI architecture for implementation of K-best decoding algorithm. The proposed architecture supports QPSK, 16-QAM and 64-QAM modulation schemes with the ability to have different modulation schemes on different transmitting antennas.

The rest of the paper is organized as follows: Section II provides an introduction to the MIMO channel model and the K-best detection algorithm. The need for a configurable K-best architecture will be discussed in section III. In section IV, the proposed K-best architecture will be presented and finally in section V, the implementation results for a CMOS 0.18 \( \mu \)m target technology are discussed. Finally, the conclusions are drawn in section V.

II. DETECTION IN MIMO CHANNELS

A. MIMO Channel Model

In a MIMO system with \( N_t \) transmitter and \( N_r \) receiver antennas \((N_t \geq 1)\), the channel model can be described as:

\[
y_c = H_c s_c + n_c
\]

(1)

Where \( y_c \) is the \( N_r \times 1 \) vector of received symbols, \( H_c = [h_{ij}] \) is the \( N_r \times N_t \) equivalent flat-fading channel matrix, \( s_c = [s_1, \ldots, s_{N_t}] \) is an \( N_t \times 1 \) vector of transmitted symbols. The task of MIMO detector is to recover \( s_c \) from \( y_c \) by solving (1). It is possible to transform the \( N_r \)-dimensional complex equation (1) to an equivalent 2\( N_r \)-dimensional real-valued representation (2) [3]. The real operation reduces the complexity of detection since the number of children nodes per parent node in a real-valued constellation \(( \log_2 p_c \) becomes smaller than complex-based constellation \(( p_c \).

The entries of (2) are defined by real-valued amplitude modulated constellation points \( \Lambda_c \).

\[
\begin{bmatrix}
\Re[y_c] \\
\Im[y_c]
\end{bmatrix} =
\begin{bmatrix}
\Re[H_c] \\
\Im[H_c]
\end{bmatrix}
\begin{bmatrix}
\Re[s_c] \\
\Im[s_c]
\end{bmatrix} +
\begin{bmatrix}
\Re[n_c] \\
\Im[n_c]
\end{bmatrix}
\]

(2)

Equation (2) can be further simplified in the form of the real-valued channel equation (3).

\[
y = Hs + n
\]

(3)

B. K-best Detection

The Maximum Likelihood (ML) detector in hard-output detectors solve the following equation

\[
s_{\text{ML}} = \arg \min_{s_c} \| y - Hs \|
\]

(4)

Following standard simplification by applying QR decomposition on channel matrix \( H \) [3], [4] we can further simplify (1) and (2) to obtain (5):

\[
s_{\text{ML}} = \arg \min_{s_c} \| y - Rs \|
\]

(5)
such that
\[ H = QR \] (6)
and
\[ y_q = Q^H y \] (7)
expanding vector norm in (5) yields
\[ s_{ss} = \arg \min_{s_{ss}} \sum_{i=1}^{N_t} |y_{ss} - \sum_{j=1}^{N_t} R_{ss} s_j|^2 \] (8)

starting from \( i = N_t \), (8) can be solved recursively as follows:
\[ T_i(P_i) = T_{i+1}(P_{i+1}) + e_i(P_i)^2 \] (9)
where
\[ T_i(P_i) > T_{i+1}(P_{i+1}) \] (10)
and
\[ e_i(P_i) = y_{ss} - \sum_{j=1}^{N_t} R_{ss} s_j \] (11)

In (9)-(11), \( P_i = [s_1, s_{i+1}, \ldots, s_{N_t}]^T \) is known as partial symbol vector [4]. Using the above simplification, the original optimization problem (2) can be recursively solved by applying an iterative tree search methodology. The cost function of the resulting optimization problem is \( T_i(P_i) \), which is known as “Partial Euclidean Distance (PED)” [7]. In K-best algorithm a breadth-first tree search is conducted to search for solution of (8) i.e. the detector visits all siblings of a node before it proceeds to the next level. Instead of expanding every node at each layer of the tree, we only keep the best K nodes that have the smallest accumulated PEDs. After completing tree search, we will have K leaves with the smallest PEDs. Each path in the tree corresponds to a signal vector \( s \). The path with smallest PED is the detection result. The K-best can guarantee a fixed throughput and has BER performance that is close to ML detector. It is also possible that each of the transmitting antennas use different modulation scheme.

III. MIMO CHANNEL VARIATION

In a MIMO wireless system, as the wireless channel condition and the noise level change, the receiver estimates the channel parameters and signal to noise ratio (SNR) in order to optimize detection process. In this section we consider two different scenarios for channel variation and will analyze the effect of parameter K in each of these cases. Fig. 1 shows the simulated BER in a K-best MIMO detector for two different channel conditions. The assumption here is that the SNR remains unchanged. We assume the detector is initially operating at point 1. Due to channel variations, the operating point of detector moves to point 2 and as a result BER is increased. It is possible to improve BER by increasing K and move operating point from point 2 to point 3. The drawback of increasing \( K \) in a K-best detector is that throughput will be decreased. In Fig. 2, it is assumed that the application requires a fixed BER. Due to variations in SNR, the operating point moves from point 1 to point 2 and as a result the BER suffers, in order to maintain the required level of BER, detector should increase the \( K \) parameter at the expense of throughput. The second scenario depicted in this figure is related to the situation where channel conditions and SNR level have not changed but due to a change in application, receiver could improve the BER performance by increasing \( K \) (point 1 to 4). We can conclude that in a K-best MIMO detector, it is possible to adjust detection parameters to improve performance under different operating condition. In the next section, we propose a VLSI architecture that satisfies this need.

![Figure 1. Effect of K and channel variations on BER performance](image)

![Figure 2. Effect of K and SNR on BER performance](image)

IV. CONFIGURABLE K-BEST ARCHITECTURE

The pipelined architecture [5] for implementation of K-best algorithm is composed of \( 2N \) processing elements (PEs) as shown in Fig. 3. In this architecture each PE is associated with one level of a real-valued search tree. The \( i \)-th PE receives \( K \) data vectors from the preceding processing element (PE \( (i+1) \)). The entries of the data vectors contain the previous vector symbols of the admissible nodes and the associated PEDs. The task of the PE is to expand each of the \( K \) parent nodes to all their children nodes, calculate the associated PEDs for each children and to identify a set of \( K \) children which have the
lowest accumulated PEDs and pass on the relevant information to the next PE. Fig. 4 shows the architecture of the proposed PE block. All candidate vector symbols and their associated PEDs (Ti) are stored in the Data Buffer block. Each PED value from previous levels generates up to 8 new PED values depending on the type of modulation scheme (64-QAM, 16-QAM or QPSK). The PED Computation block (Fig. 5) computes the PEDs of all associated children of the parent node. This block is designed in a way that it always calculates all the associated PEDs considering 64-QAM modulation. The constellation set in 16-QAM ({-3,-1,+1,+3}) and QPSK ({-1,+1}) modulation can be considered as a subset of constellation set in 64-QAM ({-7,-5,-3,-1,+1,+3,+5,+7}) modulation. In case of QPSK and 16-QAM modulation a set of multiplexers mask the values of unused PEDs and replace them with the maximum possible values. The maximum value depends on the number of data bits in the architecture; for a 16-bit fixed point system this value would be 65535 (0xFFFF). This forces the sort8 block to place the associated values for the masked PEDs at the end of the sorted queue and eventually they will be pushed down the sorting queue and have no effect in the final result. The role of the sorting block in this architecture is to sort all the generated PEDs and select the smallest K PEDs. The most challenging aspect of the design of a sorting block is to minimize the number of vertical logic levels in the architecture. In implementing the PE block we are particularly interested in sorting networks that can be configured to support a variable number of input entries (8 to 128). Sorting algorithms can be divided in to two main classes: parallel and serial. In parallel sorting all the values to be sorted are processed simultaneously in an interconnect network where in serial sorting the values arrive serially to the sorting circuit in such a way that the new value to be sorted is inserted in an already sorted list. The drawback of the serial sorting circuit is the extra clock cycles required to process each new value. Parallel-based sorting architectures are faster than serial sorting architectures but require more area. Table 1 compares the complexity of a number of parallel sorting algorithms for a 128 (16×8) input sorting network.

<table>
<thead>
<tr>
<th>Sorting algorithm</th>
<th>Number of comparators</th>
<th>Number of vertical comparator layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Batchter’s odd-even</td>
<td>1471</td>
<td>28</td>
</tr>
<tr>
<td>Parallel Bitonic</td>
<td>1792</td>
<td>28</td>
</tr>
<tr>
<td>Parallel Bubble</td>
<td>8128</td>
<td>128</td>
</tr>
<tr>
<td>Parallel Batchter’s odd-even with feedback register</td>
<td>191</td>
<td>15</td>
</tr>
<tr>
<td>Pipelined Batchter’s odd-even with feedback register</td>
<td>109</td>
<td>15</td>
</tr>
</tbody>
</table>

The Batchter’s odd-even merge-sort algorithm [6] merges two sorted sequences into one sorted sequence. The complexity and number of vertical layers in this algorithm is comparatively lower than other parallel sorting algorithms. In PE block, every clock cycle eight new PED values are computed. We can use this property to further simplify the architecture of the sorting network by using a set of feedback registers to store the K value of the previous comparison and reduce the size of sorting network to 2K. In order to further reduce the complexity, we proposed a pipelined implementation.
Fig. 4 shows the architecture of the proposed pipelined sorting block. The basic idea behind the pipelined sorting architecture is that as the new PED values arrive, they are first sorted in the `sort8` block and then either enter the pipeline register or go directly to the merge blocks and are merged with the previously stored PEDs. The role of the merge blocks is to merge two sorted sequences into one sorted output sequence. As an example, `merge16` block merges two sequences each comprised of eight elements into a sixteen element sequence [7]. In Fig. 6, the architecture of `merge16` block is depicted. The basic building block in this architecture is a simple compare/exchange (`cmpx`) cell. The two inputs entering the `cmpx` cell are compared, the smallest is connected to ‘l’ and the largest is connected to ‘h’ output. One of our main goals in this architecture is to be able to sort sequences with variable lengths. In a pipelined architecture this can be accomplished with the help of multiplexers as shown in Fig. 4.

V. IMPLEMENTATION RESULTS

Based on the proposed architecture, a 4×4 (QPSK/16-QAM/64-QAM) MIMO detector has been designed, synthesized and mapped to a 0.18 µm CMOS standard cell library. After synthesis and mapping, a clock frequency of $f_{clk}=47$ MHz has been achieved with a logic complexity of 300K equivalent gates (equivalent gates number is calculated by dividing total area by the area of a two-input drive-1 NAND gate). All the arithmetic computations are based on 16-bit fixed-point numbers. Data buffers have been implemented using registers to increase the speed at the expense of silicon area. The latency of each `PE` cell is 2K cycles and the overall latency of the architecture is $14K+1$ cycles. Throughput of the detector is $\frac{(f_{clk}.N.log_2(p_r)/2K Mbps. Direct comparison of the proposed architecture with previously reported architectures tends to be quite difficult since previously published architectures focus on implementation and optimization of a detection algorithm for a specific $K$ value and modulation scheme. Our solution, on the other hand, has been designed with the aim of supporting variable $K$ as well as three different modulation schemes for each antenna and as a result there is an associated hardware cost for silicon area and timing. The size of the data bus can significantly affect the area. In majority of the previously reported works, this parameter is missing and it makes...
comparison between architectures even more difficult. In order to be able to compare core areas implemented in different CMOS technologies, one solution would be to convert all the reported core areas to equivalent gate numbers. Table I. shows performance figures of the proposed architecture and compares them with some of the reported architectures for K-best algorithm.

<table>
<thead>
<tr>
<th>Reference</th>
<th>[8]</th>
<th>[9]</th>
<th>[9]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>1 ~ 16</td>
</tr>
<tr>
<td>Number of Antennas</td>
<td>4x4</td>
<td>4x4</td>
<td>4x4</td>
<td>4x4</td>
</tr>
<tr>
<td>Modulation</td>
<td>16-QAM</td>
<td>16-QAM</td>
<td>16-QAM</td>
<td>64-QAM, 16-QAM, QPSK</td>
</tr>
<tr>
<td>Tech [µm]</td>
<td>0.35 µm</td>
<td>0.25 µm</td>
<td>0.25 µm</td>
<td>0.18 µm</td>
</tr>
<tr>
<td>Equivalent gate number</td>
<td>91 K</td>
<td>68 K</td>
<td>110 K</td>
<td>300 K</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>100 MHz</td>
<td>132 MHz</td>
<td>52 MHz</td>
<td>47 MHz</td>
</tr>
<tr>
<td>Latency (cycles)</td>
<td>240</td>
<td>49</td>
<td>89</td>
<td>14K+1 (15 ~ 225)</td>
</tr>
<tr>
<td>Throughput</td>
<td>53 Mbps</td>
<td>424 Mbps</td>
<td>83 Mbps</td>
<td>12 ~ 564 Mbps</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

Designing a multi-mode flexible MIMO detector represent a challenging task for next generation wireless receivers. In a K-best decoder we can achieve better performance under different channel conditions by changing K parameter. In this paper a configurable VLSI architecture for K-best decoding algorithm has been presented. The proposed architecture is easily configurable for different modulation schemes (QPSK, 16-QAM and 64-QAM) and K values (1~16).

ACKNOWLEDGMENT

The authors would like to thank Altera Corporation, Ontario Centres of Excellence (OCE), and Natural Sciences and Engineering Research Council (NSERC) for their financial support. Technology access from Canadian Microelectronic Corporation (CMC) is also appreciated.

REFERENCES