VLSI Implementation of a Fully-Pipelined K-best MIMO Detector with Successive Interference Cancellation

Ibrahim A. Bello · **Basel Halak** ·
Mohammed El-Hajjar · **Mark Zwolinski**

Abstract Multiple-input multiple-output (MIMO) technology is envisaged to play an important role in future wireless communications. To this end, novel algorithms and architectures are required to implement high-throughput MIMO communications at low power consumption. In this paper, we present the hardware implementation of a modified K-best algorithm combining conventional K-best detection and low-complexity successive interference cancellation at different levels of the tree search. The detector is implemented using a fully-pipelined architecture, which detects one symbol vector per clock cycle. To reduce the power consumption of the entire receiver unit, costly symbol-rate operations such as multiplication are eliminated both within and outside the detector without any impact on the performance. The hardware implementation of the modified K-best algorithm achieves area and power reductions of 16% and 38%, respectively, compared with the conventional K-best algorithm implementation, while incurring a signal-to-noise ratio penalty of 0.3 dB at the target bit error rate. Post-synthesis analysis shows that the detector achieves a throughput of 3.29 Gbps at a clock frequency of 137 MHz with a power consumption of 357 mW using a 65-nm CMOS process, which compares favourably with the state-of-the-art implementations in the literature.

1 Introduction

The user demand for high-throughput wireless communications has been growing considerably in recent years. Some two decades ago, the IEEE 802.11b wireless local area network (WLAN) standard was introduced, which achieved a modest maximum downlink throughput of 11 Kbps over a single-antenna communication link. The introduction of multiple-input multiple-output (MIMO) technology to IEEE 802.11n a decade later made throughputs of over 500 Mbps possible. More recently, with the support of up to eight antennas, throughputs of several gigabits per second are attainable with wireless standards such as the IEEE 802.11ac [24]. With the expected widespread deployment of MIMO technology to diverse devices in future communications

Ibrahim A. Bello
Department of Engineering Science, University of Oxford, OX1 3PJ, United Kingdom
E-mail: ibrahim.bello@eng.ox.ac.uk

Basel Halak · Mohammed El-Hajjar · Mark Zwolinski
Electronics and Computer Science, University of Southampton, SO17 1BJ, United Kingdom
E-mail: {bh9,meh,mz}@ecs.soton.ac.uk

systems [1], it is necessary to implement novel algorithms and hardware architectures to achieve the gigabit data rates promised by MIMO technology.

A large number of algorithms have been studied for implementing MIMO detection [28]. Tree search algorithms, which achieve the maximum likelihood (ML) diversity, have attracted considerable attention, and several hardware implementations have been successfully achieved [5]. Most notably, the K-best algorithm, which implements the tree search using a breadth-first strategy, has received significant research interest as it is able to achieve the ML diversity order with a complexity that is independent of the signal-to-noise ratio (SNR).

The earliest implementations of the K-best detector [10, 33] were based on a bubble-sort tree search and were only able to achieve a few tens of megabits per second (Mbps) in throughput. In [30] and [19], single-cycle merge-sort algorithms were proposed to reduce the large latency of the bubble-sort implementations. In [26] and [22], a winner path extension was proposed, which generates the best candidates in a time independent of the modulation constellation size. More recently, the use of fully-pipelined K-best detectors has been proposed in [13] and [18], which allows vastly improved data rates to be achieved.

The main aim of this paper is to implement a K-best detector achieving the multi-gigabit data rates required by high-throughput wireless schemes, such as the IEEE 802.11ac. To this end, a K-best detector will be implemented with a throughput of one symbol vector per second, which is achieved by using fine-grained pipelining of the processing elements. The resulting implementation achieves a throughput of over 3 Gbps, which exceeds the throughputs of existing partially-pipelined K-best implementations. The main contributions of the paper are as follows:

1. We present a modified K-best algorithm combining K-best detection and successive interference cancellation at different levels of the tree search, which are determined after extensive simulations. Simulation results show that the SNR loss of the proposed algorithm for a spatial-multiplexing MIMO transmission is about 0.3 dB at a target bit error rate (BER) of 10^{-3}. Compared with a reference conventional K-best detector, the area and power consumptions of the proposed implementation were reduced by 16% and 38%, respectively.

2. The proposed implementation dispenses with costly symbol-rate precomputations outside the architecture, which are required by the state-of-the-art implementations such as [18]. To the best of our knowledge, this is the first fully-pipelined K-best detector to dispense with multiplication at the symbol rate within and outside the architecture. Given the high complexity of fully-pipelined circuits, eliminating costly operations, such as multiplication, is desirable to reduce the area and power consumption of the entire receiver unit.

3. We propose a novel pipeline schedule, which employs "register sharing" for the signal and channel inputs, which allows the proposed architecture to process up to 24 independent channel matrices concurrently, making the implementation to be applicable to fast-fading channel scenarios.

4. We compare pipelining as a technique for achieving multi-gigabit signal detection with interleaving, where several MIMO detector cores are operated in parallel. Using a 64-QAM 4×4 MIMO system, and based on the modified K-best algorithm, our results show that pipelining can achieve a throughput advantage of approximately $13\times$ compared with interleaving per unit area.

The paper is organised as follows. In Sect. 2, the MIMO system model and notations used in the rest of the paper are presented. In Sect. 3, we present the conventional K-best algorithm. Our proposed modification to the K-best algorithm is also presented in this section, and its error performance is analysed and compared with other tree search detection algorithms. In Sect. 4, the hardware implementation details of the proposed K-best detector are presented. The results of the VLSI implementation of the proposed detector are presented in Sect. 5 and compared with notable results from the literature. The paper is concluded in Sect. 6.

The following notations are used in the paper. $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denote the real and imaginary parts of a complex number, respectively; $A_{i,j}$ represents an element in the ith row and jth column of the matrix A; A_j represents the jth column of A, while $A_{i,j:k}$ represents the vector $[A_{i,j}, A_{i,j+1}, \ldots, A_{i,k}]$.

2 Background

2.1 MIMO System Model

We consider a MIMO transmitter employing N_T antennas and transmitting information symbols over a wireless link to N_R receive antennas. The $N_R \times 1$ received signal vector (RSV), $\mathbf{y}$, at the MIMO receiver is given by the following equation:

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n}, \tag{1}$$

where $\mathbf{H}$ represents the $N_R \times N_T$ channel matrix, $\mathbf{s}$ represents the $N_T \times 1$ modulated MIMO symbol vector from the transmitter, and $\mathbf{n}$ represents the additive white Gaussian noise. The entries of $\mathbf{H}$ are assumed to be independent and identically distributed with Rayleigh fading. To recover the transmitted symbol, $\mathbf{s}$, at the receiver, a QR decomposition can be performed on the channel matrix as follows:

$$\hat{\mathbf{y}} = \mathbf{Rs} + \mathbf{Q}^H \mathbf{n}, \tag{2}$$

where $\mathbf{H} = \mathbf{QR}$, $\hat{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$, $\mathbf{Q}$ is a unitary $N_R \times N_R$ matrix, and $\mathbf{R}$ is an upper triangular $N_R \times N_T$ matrix. For simplicity, we assume an equal number of antennas at the transmitter and receiver; i.e. $N_T = N_R$.

2.2 Real-Valued Channel Model

A real-valued decomposition (RVD) can be performed on the channel matrix to transform (1) as follows [30]:

$$\begin{bmatrix} \Re\{\mathbf{y}\} \\ \Im\{\mathbf{y}\} \end{bmatrix} = \begin{bmatrix} \Re\{\mathbf{H}\} & -\Im\{\mathbf{H}\} \\ \Im\{\mathbf{H}\} & \Re\{\mathbf{H}\} \end{bmatrix} \begin{bmatrix} \Re\{\mathbf{s}\} \\ \Im\{\mathbf{s}\} \end{bmatrix} + \begin{bmatrix} \Re\{\mathbf{n}\} \\ \Im\{\mathbf{n}\} \end{bmatrix}, \tag{3}$$

which transforms the complex constellation set into the integer set as follows:

$$\mathcal{D} = \{-\sqrt{M} + 1, \ldots, \sqrt{M} - 1\}, \tag{4}$$

where M is the modulation order. The QR decomposition can then be performed on the basis of the augmented channel equation in (3).

The RVD transformation simplifies the tree search in hardware as it is easier to operate on real numbers than complex numbers. In unpipelined detectors, the complex channel model has the advantage of resulting in a higher throughput since the tree depth is shorter. However, as we will see in subsequent sections, the channel model employed becomes less relevant to the throughput of the detector in a fully-pipelined implementation.

2.3 MIMO Detection

The aim of the MIMO detector is to provide an estimate, $\hat{\mathbf{s}}$, of the transmitted symbol vectors. The maximum likelihood (ML) solution is obtained as the symbol vector which minimises the Euclidean distance, $\|\mathbf{y} - \mathbf{Hs}\|^2$. As a result of the triangular channel matrix, $\mathbf{R}$, the Euclidean distance, T, of the lattice point, $\mathbf{H\hat{s}}$, from the received signal can be computed successively as follows:

$$T(\hat{\mathbf{s}}) = \sum_{i=1}^{2N_T} \left(\hat{y}_i - \sum_{j=i}^{2N_T} r_{i,j} s_j \right)^2 . \tag{5}$$

For example, for $N_T = 2$, and using a real channel model, the Euclidean distance is computed incrementally over four levels in the following sequence:

$$
\begin{aligned}
&(\hat{y}_4 - r_{4,4} s_4)^2 + \\
&(\hat{y}_3 - (r_{3,4} s_4 + r_{3,3} s_3))^2 + \\
&(\hat{y}_2 - (r_{2,4} s_4 + r_{2,3} s_3 + r_{2,2} s_2))^2 + \\
&(\hat{y}_1 - (r_{1,4} s_4 + r_{1,3} s_3 + r_{1,2} s_2 + r_{1,1} s_1))^2 ,
\end{aligned}
$$

where the Euclidean distance at each level is referred to as the partial Euclidean distance (PED). Equation (5) describes a tree, initially with $|\mathcal{D}|$ branches at the topmost level, which correspond with the constellation set, $\mathcal{D}$. Interested readers are referred to [11] and [23] for a more in-depth discussion on tree search detection. Each branch, extended to the last level, $i = 1$, represents a potential solution. A total of $|\mathcal{D}|^{2N_T}$ solutions are possible in the ML search. As a result of the exponential complexity of the ML detector, a number of algorithms with sub-ML BER performances have been proposed as alternatives in the literature [28].

3 K-Best Algorithm

The K-best algorithm employs a breadth-first search, where the children of parent nodes retained from a previous level are expanded in parallel. The PEDs of the nodes can be computed using the ℓ^1-norm approximation as follows [6]:

$$T_i(s_i) = T_{i+1}(s_{i+1}) + |b_i - r_{i,i} s_i| , \tag{6}$$

where $T_i(s_i)$ represents the PED of a symbol at the ith level, $T_{i+1}(s_{i+1})$ denotes the PED of its parent, and

$$b_i = \hat{y}_i - \sum_{j=i+1}^{2N_T} r_{i,j} s_j. \tag{7}$$

A node refers to a symbol drawn from the real constellation set in (4) at a given level of the tree search. For brevity, $T_i(s_i)$ will be denoted by T_i in subsequent discussions. Each level corresponds with a row of the triangular channel matrix, $\mathbf{R}$. Thereafter, a sorting operation is carried out to select the best K candidates, which are passed as the parent nodes to the next level. The PED can also be computed as [6, 13]:

$$T_i = T_{i+1} + |r_{i,i}(c_i - s_i)| , \tag{8}$$

Algorithm 1 Modified K-best Algorithm

$i \leftarrow 2N_T$
Compute $T_i \, \forall \, s_i \in \mathcal{D}$
$\mathcal{K}_{i,1:\sqrt{M}} \leftarrow \mathcal{D}$
$i \leftarrow i - 1$

while $i \geq 1$ **do**
  if $i \geq I$ **then**
    Compute T_i for all children of $\mathcal{K}_{i+1,1:K}$
    $\mathcal{K}_{i,1:K} \leftarrow \texttt{KBEST}(s_{i,j,k}) \, \forall \, i,j,k$
    $\texttt{UPDATE}(\mathcal{K}_{j,1:K})$ where $i < j \leq 2N_T$
  else
    Extend the best children of $\mathcal{K}_{i+1,1:K}$ using Equation (9)
    Compute T_i for each extended child
  end if

  $i \leftarrow i - 1$
end while

$\hat{\mathbf{s}} \leftarrow \mathcal{K}_1$

---

where $c_i = b_i/r_{i,i}$ and is referred to as the Schnorr-Euchner (SE) centre. Visiting the child nodes according to their distances from the SE centre speeds up the tree search; however, computing the SE centre leads to a costly division step which is avoided in (6).

A pitfall of the K-best algorithm is that the complexity tends to be high since the operations need to be duplicated over $2N_T$ levels. In [2], a fixed-complexity sphere decoder (FSD) was proposed, which dispenses with the need for sorting, and instead, combines ML detection with low-complexity detection techniques at lower levels. The FSD relies on a vertical Bell Laboratories layered space-time (V-BLAST) [32] channel ordering at the preprocessing stage, which could result in a high complexity and throughput degradation in fast-fading channels. It is thus desirable to implement techniques that will reduce the complexity of the K-best detector, without requiring any additional preprocessing operations. A modified K-best algorithm is proposed in the next section.

3.1 Proposed K-Best Algorithm

In this paper, we propose a hybrid detector, where the K-best detection is carried out only for "upper" levels of the tree search, defined as $I \leq i \leq 2N_T$, where i is the level index, and I is some integer between 1 and $2N_T$. At $i = 1$, sorting can be avoided, since only a single path, not K paths, are required. The rationale of this technique is the fact that it is easier to make an erroneous decision in the upper levels since any error will be propagated to subsequent levels of the tree, which progressively worsens the detection symbol error rate. This observation has also been employed in non-constant K-best detectors [20, 29], where smaller K values are applied at lower levels. In the detector proposed here, the same value of K is maintained throughout the detection; however, only a low-complexity successive interference cancellation (SIC)-based extension is carried out in lower levels. K-best detection is carried out up till $i = I$. If $I = 1$, then the hybrid detector reduces to the conventional K-best detector. In levels $i < I$, the best
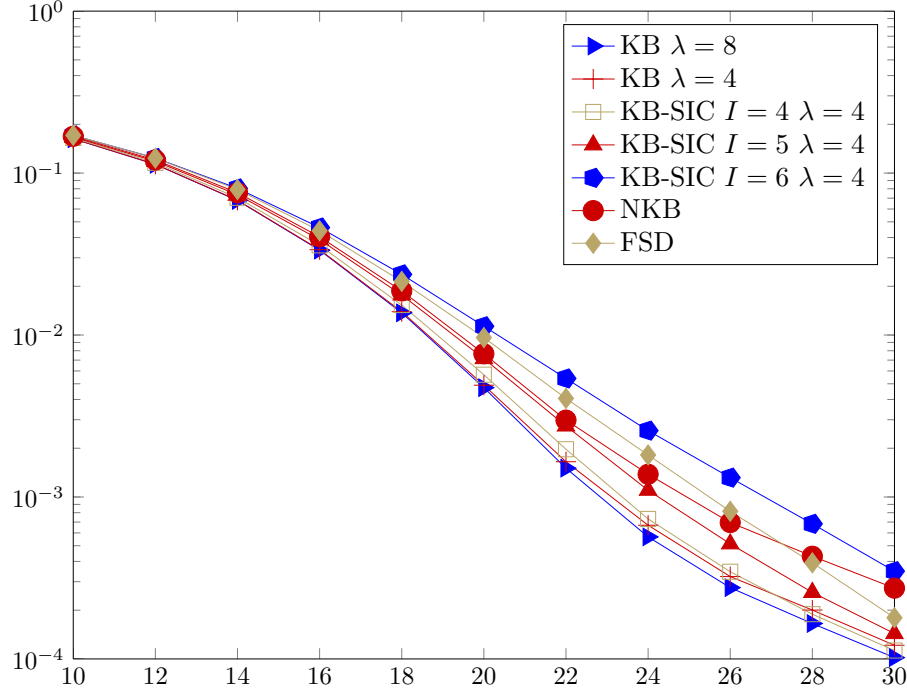
Fig. 1: BER versus SNR for different MIMO detectors using a real channel model for 64-QAM and $N_T = 4$

child of each of the K-best paths, $s_i^{[1]}$, extended up till $i = 1$, is derived as follows:

$$s_i^{[1]} = \arg\min_{s_i \in \mathcal{D}} |b_i - r_{i,i} s_i|. \tag{9}$$

No sorting operation is carried out to select the best K candidates for levels less than I. However, at the last level, a minimum (MIN) search amongst all the K-best candidates is carried out to determine the hard-detection output. It should be noted that at each level below I, this technique will always select the minimum-metric candidate from each parent node. However, the K candidates so selected may differ from the K-best candidates selected by the conventional K-best algorithm. The K-best detection is summarised in Algorithm 1, with the proposed SIC detection steps highlighted in bold. $s_{i,j,k}$ represents the jth child of the kth parent at the ith level. $\mathcal{K}$ represents a $2N_T \times K$ matrix of K-best symbols. The KBEST function sorts all the candidates at a given level in ascending order and selects the top K results. The UPDATE function permutes the previously detected paths, $\mathcal{K}_{i+1:2N_T,1:K}$, according to the sorted PEDs of $\mathcal{K}_{i,1:K}$.

Unlike conventional SIC-aided linear detection [32], no slicing operation is required to obtain the detected symbols in the proposed algorithm. Apart from the SIC detection, a low-complexity SE enumeration is adopted where only the best $\lambda \leq \sqrt{M}$ children of a node are enumerated [3]. For the conventional K-best detector, $\lambda = \sqrt{M}$ for all extended parent nodes. In our implementation, $\lambda = \sqrt{M}$ is selected for the first two levels of the tree search, while $\lambda < \sqrt{M}$ is applied in subsequent levels.

Table 1: Number of nodes expanded for the different tree search detectors in Fig. 1 for $N_T = 4$ and 64-QAM using a real channel model

Detector	Number of expanded nodes	$N_T = 4$
KB	$\sqrt{M}(1 + \sqrt{M} + K(2N_T - 3)) + K$	728
NKB	$\sqrt{M} + \sum_{i=1}^{2N_T - 1} K_{i+1}\sqrt{M}$	272
FSD	$\sum_{i=1}^{2N_T} \prod_{j=i+1}^{2N_T+1} n_i n_j$	1480
KB-SIC	$\sqrt{M}(1 + \sqrt{M}) + \sum_{i=I}^{2N_T - 2} K\lambda + K(I - 1)$	312

3.2 Performance Analysis

In this section, we compare the performance of the proposed hybrid K-best algorithm (KB-SIC) described in the previous section with the conventional K-best algorithm employing $\lambda = 8$, for a MIMO system transmitting over a Rayleigh flat-fading channel using four antennas and 64-QAM. The BER simulation is shown in Fig. 1. The value of K is selected as 16 for both the conventional and hybrid K-best detectors. A non-constant K-best (NKB) detector [13] employing K values of $[K_8\,K_7\,\ldots\,K_1] = [8\,8\,8\,8\,4\,2\,2\,1]$ is shown, where K_i is the number of candidates extended at the ith level, and K_{2N_T} corresponds with the number of constellation points. A fixed-complexity sphere decoder employing a minimum mean square error V-BLAST channel ordering is also shown. The FSD uses a real channel model and a node distribution of $[n_1\,n_2\,\ldots\,n_8] = [1\,1\,1\,1\,2\,2\,8\,8]$, where n_i is the number of nodes extended per parent in the ith level. A total of $800,000$ symbol vectors were used for the simulation, with a new random channel matrix, with independent and identically distributed gains, generated once for every four symbol vectors. All detectors are based on the ℓ^1-norm approximation of the PED proposed in [6].

The hybrid K-best detector is simulated for $I = 4, 5$ and 6. The BER simulation shows that increasing the value of I also increases the BER. This is because larger values of I increase the number of levels that erroneous detections will be propagated to. KB-SIC with $I = 4$ and $\lambda = 4$ suffers an SNR loss of about 0.3 dB and 0.6 dB at a BER target of 10^{-3}, compared with the conventional K-best detector having $\lambda = 4$ and $\lambda = 8$, respectively. By contrast, KB-SIC with $I = 6$ suffers over 1-dB SNR loss compared to the conventional K-best detectors. Despite the V-BLAST preprocessing, the FSD with the adopted node distribution shows a reduced error performance compared with the K-best detectors, which is as a result of the low-complexity detection adopted at lower levels. The performance of the K-best detectors can be improved by using a V-BLAST channel ordering or sorted QR decomposition [35] in the preprocessing stage, while the FSD can be further improved by increasing the node distribution.

3.3 Complexity Analysis

The complexity of tree search algorithms is typically defined as the number of nodes visited in the tree search [11]. As a result of the SIC detection in some levels, the complexity of KB-SIC is reduced compared with the conventional K-best algorithm as shown in Table 1. For ease of
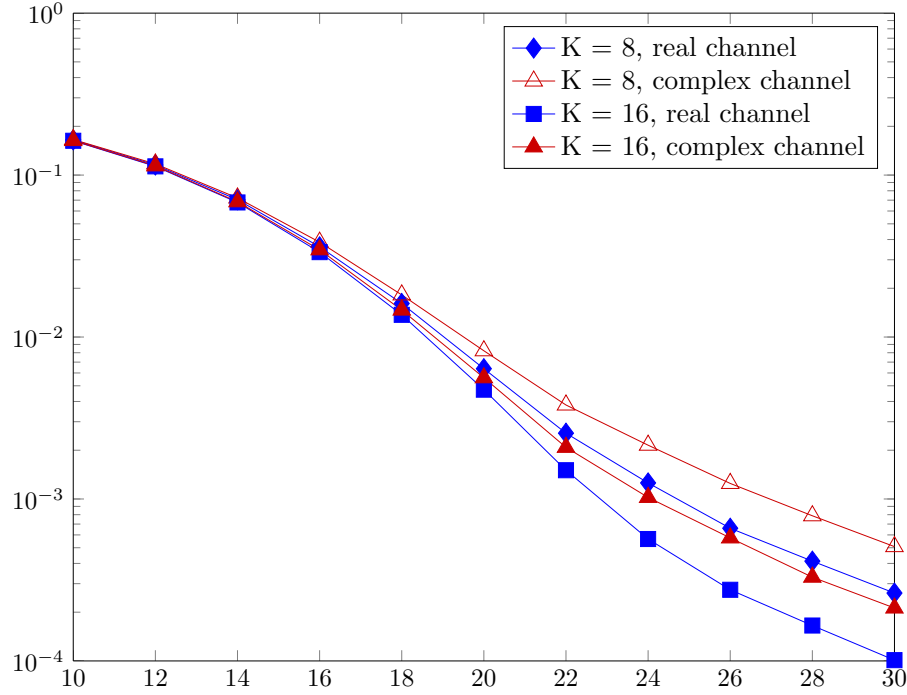
Fig. 2: Performance of the K-best detector for different values of K using real and complex channel models for 64-QAM and $N_T = 4$. All K-best implementations are based on the ℓ^1-norm approximation.

comparison, all detection algorithms are based on a real channel model. For $I = 4$ and $\lambda = 4$, KB-SIC expands 312 nodes, which is more than a 50% reduction compared with the number of nodes expanded by the conventional K-best (KB) detector. For both KB-SIC and KB, only the best child nodes of each parent are expanded in the last level, which simplifies the SE enumeration.

The FSD expands the largest number of nodes compared with the other algorithms in Fig. 1. However, it should be noted that the FSD can achieve a much-reduced number of expanded nodes if a complex channel model is used. For example, the complex FSD with a node distribution of [1 1 1 64] expands just 512 real nodes, while achieving a near-ML performance [2]. Each node in the complex channel model is counted as two real channel model nodes. This result suggests that it is more advantageous to implement the FSD based on a complex channel model rather than on a real channel model. On the other hand, the K-best detector expands a fewer number of nodes in the real channel model. For example, the real K-best algorithm expands 728 nodes and 1384 nodes, respectively, for $K = 16$ and 32, while the complex K-best algorithm expands 4256 and 8384 real nodes, respectively, for the same values of K. Furthermore, the real channel model exhibits a better BER performance for the same value of K compared to the complex channel model as shown in Fig. 2. A more rigorous comparison of the real and complex channel models is provided in [9].

Although the FSD expands a fewer number of nodes compared with the K-best detector, it also requires a mandatory V-BLAST preprocessing step. In a slow-fading channel, this additional preprocessing can be ignored; however, in a fast-fading channel, this can impose additional complexity at the receiver side as the V-BLAST preprocessing requires computationally expensive operations such as finding the channel matrix inverse. As will be shown later in the paper, the
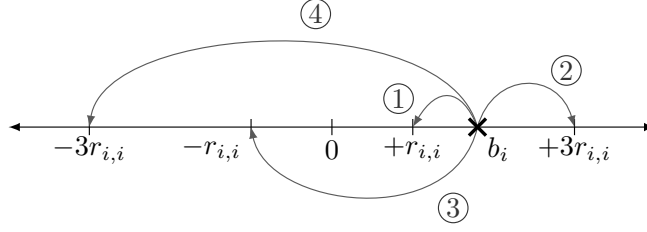
Fig. 3: SE enumeration based on a real axis for 16-QAM. The encircled numbers indicate the child ordering. In this case, b_i falls within the positive $r_{i,i}s_i$ axis and the enumeration is $\{+1, +3, -1, -3\}$.

K-best algorithm can be implemented entirely with only simple operations, such as shifts and additions, without any impact on the performance.

4 Hardware Implementation

The proposed K-best detector is implemented for a MIMO system employing 64-QAM and 4×4 antenna configuration. I and λ are both selected as 4. The inputs, $\hat{\mathbf{y}}$ and $\mathbf{R}$, are represented using signed 14 bits. The 64-QAM symbols of the real constellation set are represented using three bits. The PEDs are represented using unsigned 13 bits. All variables are represented using two's complement fixed-point format. In the next sections, the hardware implementation details of the proposed detector are presented.

4.1 Schnorr-Euchner Enumeration

In this work, the Schnorr-Euchner enumeration [25] is employed to list the children of each parent node according to their metrics. The child nodes can be enumerated in a zigzag fashion by finding the node that minimises the PED increment term as follows:

$$s_i^{[k]} = \arg\min_{s_i \in \mathcal{D}} |b_i - r_{i,i}s_i| \,,$$

where k represents the current iteration of the SE enumeration, and $s_i \notin \{s_i^{[1]}, s_i^{[2]}, \ldots, s_i^{[k-1]}\}$, where each symbol is drawn from the real constellation set, $\mathcal{D}$. The process is repeated until all $\sqrt{M}$ children of the parent node are listed. Note that the SE enumerations of the children of all the K parents are executed in parallel. An SE enumeration for 16-QAM is illustrated in Fig. 3, where the numbers within the circles indicate the enumeration ordering, or the distance of b_i from $r_{i,i}s_i$. To speed up the procedure, a tabular enumeration is employed, where the possible enumerations are precomputed and stored into a lookup table (LUT). Simulation results show that this has negligible impact on the BER [31]. It should be noted that the computation of $r_{i,i}s_i$ does not require any multipliers. Since s_i is drawn from a known integer set, $r_{i,i}s_i$ can simply be obtained using adders and shifters [34].

There are 14 possible enumerations overall, based on the location of b_i on the $r_{i,i}s_i$ axis. However, due to the symmetry of the $r_{i,i}s_i$ axis, it is sufficient to compute only half of the enumerations by comparing $|b_i|$ with $|r_{i,i}s_i|$. That is, the enumeration with b_i on the positive $r_{i,i}s_i$ axis is comparable with the enumeration with b_i on the corresponding location on the negative
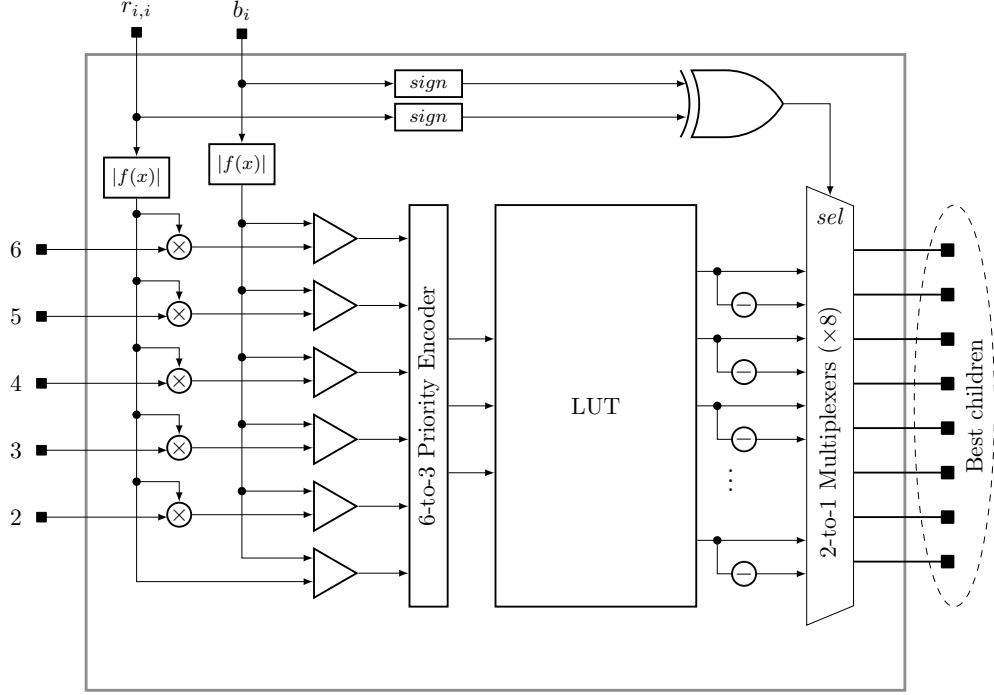
Fig. 4: SE enumeration unit using a lookup table. An XOR gate is used to decide whether to "flip" or "keep" the computed enumeration based on the signs of b_i and $r_{i,i}$.

$r_{i,i}s_i$ axis with the symbol signs flipped. The actual enumeration can then be determined by "flipping" the computed enumeration if b_i and $r_{i,i}$ have different signs as follows:

$$E(b_i, r_{i,i}) = \begin{cases} +E(|b_i|, |r_{i,i}|) & \text{if } sign(b_i) = sign(r_{i,i}) \\ -E(|b_i|, |r_{i,i}|) & \text{if } sign(b_i) \neq sign(r_{i,i}), \end{cases}$$

where $E(b_i, r_{i,i})$ computes the enumeration based on b_i and $r_{i,i}$ and $sign(.)$ returns the most significant bit of its argument. A circuit to compute the tabular enumeration is shown in Fig. 4. The circuit consists of six comparators, which compare $|b_i|$ with integer multiples of $|r_{i,i}|$. The outputs of these comparators are passed to a priority encoder, which selects the appropriate enumeration from the LUT. In the proposed detector, the full enumeration ($\lambda = 8$) is only employed in the second level ($i = 7$), while only the top $\lambda = 4$ best children are enumerated in levels $i >= I$. In levels $i < I$, only the best child of a parent node is enumerated.

4.2 Sorting

Sorting plays an important role in the complexity, performance and throughput of the K-best detector. In this paper, we employ the Batcher's bitonic and odd-even algorithms, which utilise an interconnection of comparators to sort an input list, to compute the best K candidates. From level 6 to 4, a total of $\lambda = 4$ children are expanded from each K parent, and these are sent in pairs to a merge unit. Each candidate is organised as $(s_{i,j,k}, T_{i,j,k}, k)$, where $s_{i,j,k}$ is the jth child of the kth parent node at the ith level and $T_{i,j,k}$ is its corresponding metric. This process is
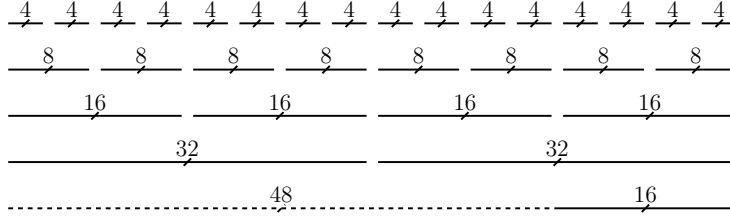
Fig. 5: Merge network for selecting 16 candidates. The dashed line represents the discarded candidates at the output of the merge network.

continued successively until all $K \times \lambda = 64$ candidates are obtained. However, since K is selected as 16 for the proposed detector, the bottom 48 candidates, and all associated comparators, are discarded. A tabular Schnorr-Euchner enumeration, described in the previous section, is used to presort the children of each parent, which reduces the complexity and latency of the merge unit.

The merge process is illustrated in Fig. 5. The first row represents the λ children of all the K parents. Each subsequent horizontal line represents the output of a stage of the merge network. Four merge stages are thus required to obtain the fully sorted result. In level 7, eight children are expanded from eight constellation points resulting in a total of three merge stages. To reduce the latency, the merge network is pipelined such that the best K children are produced within two clock cycles. A more detailed description of the merge network has been presented in a separate paper [4].

4.3 Pipeline Schedule

The K-best detector is typically implemented using a multi-stage architecture, where each stage corresponds with a level in the tree search. Multiple received signal vectors can be processed concurrently, such that a new detected symbol vector is generated after every C_1 clock cycles, where C_i is the number of clock cycles required to process the ith stage. If a long-latency sorting algorithm is employed, such as the bubble sort [33], or distributed sort [26], then achieving a throughput of 1 Gbps and over is challenging, unless the clock frequency is increased considerably.

The detection latency can be reduced by employing a merge-sort algorithm, such as the Batcher's odd-even merge, as described in the previous section. However, a more dramatic improvement to the throughput is obtained by *fully* pipelining the multi-stage architecture, such that a new result is generated in every clock cycle. The architecture is fully pipelined by ensuring that no single operation takes more than one cycle to complete, and a different RSV is processed at the next clock cycle in every pipeline stage. Large combinational blocks, such as the merge network, are broken into smaller combinational units in order to reduce the latency.

Figure 6 illustrates the pipeline schedule for a MIMO system employing $N_T = 4$. PAU represents the path update operation, while INFR represents the interference cancellation step in (7). PED$^+$ does not execute any arithmetic operation: In this pipeline stage, the outputs of the first stage of the pipelined merge network are propagated to the second stage, where the fully sorted PEDs are obtained. The PED computation at the topmost level is denoted by PED. For the first 17 clock cycles, the normal K-best operations are carried out for the first RSV, while the low-complexity SIC-based detection is performed from the 18th to the 21st clock cycles. Normal K-best detection is resumed in the 22nd clock cycle, corresponding with the start of the last tree level.

Signals	Time slots

	1	2	3	4	5	6	7	⋯	18	19	20	21	22	23	24	25	26	27
	\multicolumn{8}{c}{K-best}	\multicolumn{4}{c}{SIC}	\multicolumn{6}{c}{K-best}
$\hat{\mathbf{y}}^{(1)}$	PED	INFR	PED	PED⁺	PAU	INFR	PED	⋯	INFR	PED	INFR	PED	INFR	PED	PAU	<u>PED</u>		
$\hat{\mathbf{y}}^{(2)}$		PED	INFR	PED	PED⁺	PAU	INFR	PED ⋯		INFR	PED	INFR	PED	INFR	PED	PAU	<u>PED</u>	
$\hat{\mathbf{y}}^{(3)}$			PED	INFR	PED	PED⁺	PAU	INFR PED ⋯	PED		INFR	PED	INFR	PED	INFR	PED	PAU	<u>PED</u>
⋮				⋱						⋮								⋮
$\hat{\mathbf{y}}^{(24)}$															<u>PED</u>	INFR	PED	PED⁺

Fig. 6: Pipeline schedule for a K-best detector with $N_T = 4$

In the top level of the tree search ($i = 8$), only a PED operation is executed to expand the $\sqrt{M}$ constellation points, which marks the beginning of the first RSV, denoted by $\hat{\mathbf{y}}^{(1)}$. In the second clock cycle, the interferences of the expanded constellation points are cancelled from the signal entry at level 7 as follows:

$$b_7 = \hat{y}_7 - r_{7,8}\hat{s}_8.$$

At the same time, the PEDs of the constellation points for the second RSV are computed. The PEDs of the candidates at the seventh level for the first RSV are then computed according to (6). The candidate nodes are sent immediately to the pipelined merge network. The remaining tree levels are processed similarly until the sixth and seventh levels where only a low-complexity SIC detection is carried out. These levels are processed within two clock cycles, and no path update operation is executed since no sorting is carried out. In the final level ($i = 1$), a minimum-metric search is carried out, instead of full sorting, and the level is processed within three clock cycles. Overall, 24 clock cycles are required to completely process one RSV. Thus, in order to fill the pipeline, such that one result is generated in every clock cycle, 24 RSVs need to be processed concurrently by the K-best detector. By comparison, the conventional K-best algorithm requires 28 RSVs to achieve a full pipeline, which increases the area and power consumptions. In the next sections, the data movement of various variables and intermediate results within the pipeline will be discussed.

4.3.1 Signal and Channel Inputs

Since multiple RSVs are processed in the pipelined detector, multiple registers need to be allocated to the channel entries, $\hat{\mathbf{y}}$ and $\mathbf{R}$. In a straightforward implementation, the registers will need to be replicated 24 times for a 4×4 MIMO system, and multiplexers can then be used to select the appropriate register corresponding with the current RSV. In this work, we propose a register-sharing approach, where registers are shared amongst the RSVs as soon as the registers become available. This is based on the observation that not all inputs and intermediate results are required for the entire duration of the pipeline. For example, $r_{8,8}$ and $\hat{y}_8$ are only required in the first clock cycle for the expansion of the constellation points, while $r_{1,1}$ is only required in the 23rd clock cycle. Therefore, assuming a new channel realization for each RSV, a shift register of
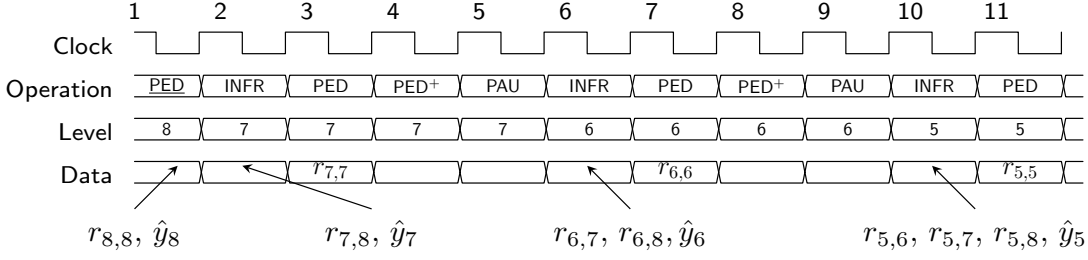
Fig. 7: Data movement of signal and channel inputs in the K-best pipeline
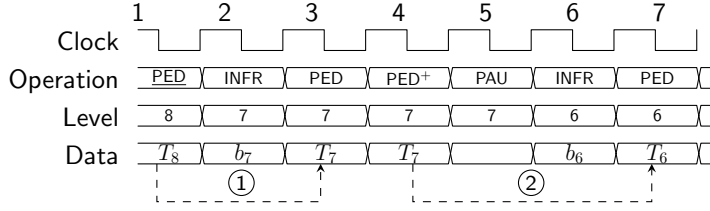
Fig. 8: Data movement of the PED in the K-best pipeline

length 23 is required to ensure that $r_{1,1}$ is correctly read to compute the PEDs of the last-level nodes for all RSVs.

Figure 7 shows the pipeline schedule and the clock cycles in which the channel inputs are read. The non-diagonal entries of the triangular channel matrix are read row-by-row and sent to a shift register for the computation of the interference terms. Non-diagonal elements in the upper rows of the triangular matrix require longer shift registers than those in lower rows. For example, $r_{5,6}, \ldots, r_{5,8}$ require a shift register of length 10, while $r_{7,8}$ requires a shift register of length 2. The diagonal elements are read one clock cycle later for the computation of the PED. Unlike some authors [27], no assumption is made about the sampling rate of the channel, and the proposed pipeline schedule can potentially support a new set of channel realizations in every clock cycle. By utilizing the register-sharing technique, the detector is able to process a maximum of 24 independent channel matrices concurrently.

4.3.2 Partial Euclidean Distance

The PEDs and intermediate interference cancellation results, b_i, can also be stored in time-multiplexed registers similar to the channel inputs described in the previous section. In the first clock cycle of the pipeline schedule, T_8 is computed for the first RSV. In the second clock cycle, b_7 for the first RSV is computed; however, its value is only used in the third clock cycle where T_7 is computed. Thus, a single register, multiplexed amongst 24 successive RSVs, is sufficient to store the computed values of b_7. The computed T_7 values are propagated into the pipelined merge network, and the fully sorted T_7 values are obtained in the fourth clock cycle. In the seventh clock cycle, the computed T_7 values are consumed to compute T_6. Therefore, T_7 can be stored in a shift register of length two, which reduces the number of T_7 registers by more than 85% compared with the direct implementation allocating a dedicated register to each RSV. However, for levels 2 and 1, only a single register is required to store the values of T_3 and T_2, since the SIC detection in levels 3 and 2 eliminates the PAU pipeline stage. The data movement of the
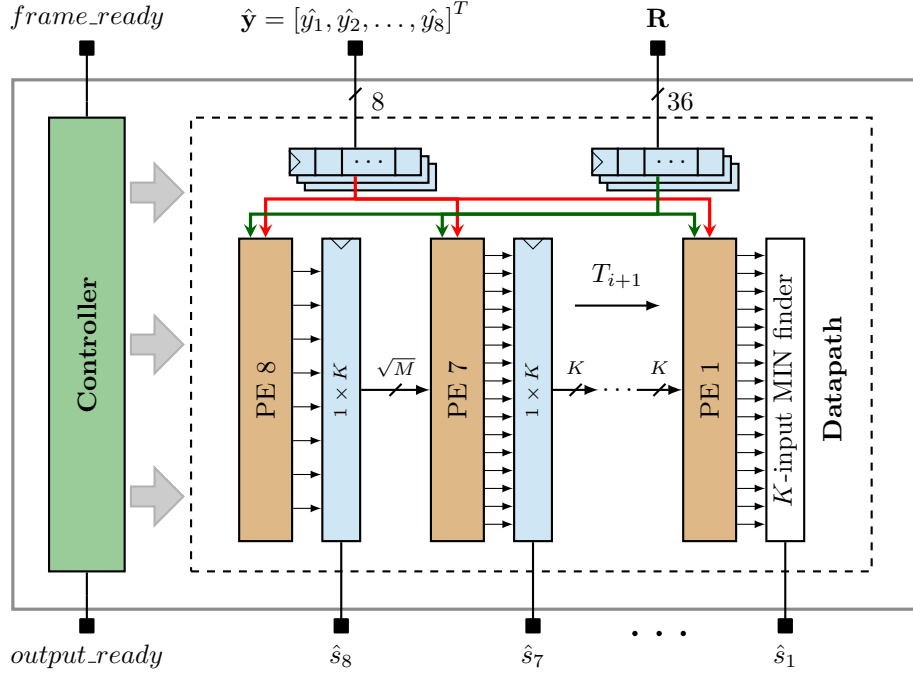
Fig. 9: Overall architecture of the fully-pipelined K-best detector. The entries of the input signal and triangular channel matrix are stored in shift registers, which are sized to ensure that the inputs are correctly read in the appropriate pipeline stage.

PEDs and interference terms is illustrated in Fig. 8, with the number of clock cycles required for holding the values of T_8 and T_7 shown in the circles.

4.4 Overall Architecture

The overall architecture of the K-best detector is demarcated into a controller unit and a datapath as shown in Fig. 9. The inputs to the detector include the signal vector, comprising eight entries, and the triangular matrix, comprising 36 entries. All inputs to the detector are real numbers represented in fixed-point format. The datapath comprises eight processing elements (PEs), with each PE corresponding with a level of the tree search. A PE finds the best K candidates at a level, and these are forwarded as the parent nodes to the next PE. All the PEs adopt a multiplier-free datapath to reduce the area and power consumption. The controller oversees the operation of the datapath and is implemented as a finite state machine (FSM), whose states correspond with each of the pipeline stages shown in Fig. 6. A $frame\_ready$ signal is used to launch the FSM from its idle state to the first state of the pipeline. After the pipeline is filled, the controller asserts an $output\_ready$ signal, and the detected symbol vectors become available at the next clock cycle. In the next section, the processing elements at the various levels of the tree search will be discussed in more detail.

4.4.1 Processing Elements 8 and 7

The first two processing elements in Fig. 9 correspond with levels eight and seven of the tree search. The two PEs expand the constellation points and their $\sqrt{M}$ children to derive the initial K paths. Initially, only $\sqrt{M}$ symbol registers, corresponding with the constellation points, are filled in level 8. However, after the path update operation of level 7, the topmost symbols are expanded to fill K symbol registers. All $\sqrt{M}$ children of the top level constellation points are expanded. For PE 8, no SE enumeration is required; however, in PE 7, a tabular enumeration is used to list the $\sqrt{M}$ children of each level 8 constellation point, according to their PEDs, as illustrated in Fig. 4.

4.4.2 Processing Elements 6 to 4

The PEs in this level are quite similar. The main difference is in the computation of (7), with PEs at lower levels requiring longer adder chains to sum up the interference terms, $r_{i,j}\hat{s}_j$. Each PE in these levels consists of K expansion units for computing (6) for each of the child nodes of the previous K parents, and a merge network, for computing the best K candidates. The SE enumeration units in these levels are simplified compared to that of PE 7 since only the top λ child nodes are required. The PEs are divided into three pipeline stages: *INFR, PED, PED$^+$*. As such, each PE processes three RSVs concurrently.

4.4.3 Processing Elements 3 and 2

These processing elements perform the SIC detection described in Sect. 3.1. The minimum-metric node of each parent is precomputed and is determined dynamically based on the values of b_i and $r_{i,i}$. Since only a single child node is required (i.e. $\lambda = 1$), this simplifies the implementation of the lookup table compared with PEs 7 to 4. Each PE in this stage comprises two pipeline stages: *INFR* and *PED*. As such, a total of four RSVs are processed concurrently in levels 3 and 2.

4.4.4 Processing Element 1

This is the final processing element in the datapath. Unlike PEs 7 to 4, no sorting is required, since only one path is needed to obtain the hard-detection output. A minimum-metric path unit compares the best child of each K parent from level 2 and successively determines the minimum-metric candidate by comparing two candidates at a time. After the best node at the last level is obtained, a path update operation is performed, which updates the previously detected symbols up to level 8 according to the path index of the best node in level 1. In contrast to the symbol registers in previous levels, only a single symbol register is required to store the last-level symbols, $\hat{s}_1$, for all RSVs, since the last-level symbols are held for only one clock cycle.

5 Results and Discussion

In this section, we will present the implementation results of the proposed K-best detector and compare with other notable MIMO detector implementations for a 64-QAM 4×4 MIMO system. We will also compare the proposed implementation with a conventional K-best detector utilizing K-best detection in all stages of the tree search in order to assess the impact of the SIC-based detection in the proposed implementation. To ensure a fair comparison with other works, the power consumption (P) is scaled to a common technology reference of 65 nm at a core voltage

Table 2: Comparison of the proposed implementation with 64-QAM 4×4 MIMO Detectors

Reference	[22]	[27]	[14]^a	[16]	[17]	[7]	[13]	[3]	[18]	ASIC I	ASIC II
Detector	KB	KB	KB	FSD	KB	FSD	NKB	KB	KB	KB	KB-SIC
Output	Hard	Hard	Soft/Hard	Hard	Hard	SISO^b	Hard	Hard	Hard	Hard	Hard
Pipelining	N/A	Partial	Partial	Partial	Partial	Partial	Full	N/A	Full	Full	Full
Tech. [nm]	65	130	130	65	130	90	90	65	130	65	65
V_{dd} [V]	1	1.3	–	1.2	1.2	1	1.3	1.05	1.3	1.05	1.05
Area [kGE]	1760	114	223	88.2	340	555	232	157.5	950	1753	1467
K	64	10	8	N/A	10	N/A	≤ 8	10	10	16	16
Φ [Mbps]	100	1350	1710	1980	2000	3046	5649	300	26600	3288	3288
Symb. / cycle	0.03	0.1	0.13	0.5	0.10	0.25	1	0.04	1	1	1
f_{clk} [MHz]	158	282	285	165	417	370	170	287	556	137	137
P [mW]	181.9	88.1	258	78.6	1301.6	370.2	152.7	26.7	780.9	580	357
E_{bit} [pJ/bit]	1819	65.2	301	39.7	651	121.5	27	91	29.4	176	108.6

^{a} Supply voltage not available. Original power and E_{bit} figures presented.
^{b} Soft-input soft-output. Throughput is per iteration.

of 1.05 V according to $1/U^2$, while the throughput (Φ) is scaled according to S, where U is the ratio of the voltage to the reference voltage, and S is the ratio of the target technology to the reference technology [8].

Two detectors are implemented. The first implementation (ASIC I) is based on the conventional K-best algorithm while the second implementation (ASIC II), is based on the proposed hybrid KB-SIC algorithm. Both implementations employ $\lambda = 4$. The power consumption of the proposed detector is determined using Power Compiler after a post-synthesis gate-level simulation, while the area consumption is determined after a place-and-route step in Cadence Encounter, and is expressed in terms of the gate equivalent (GE). One GE is the area of one two-input drive-1 NAND gate. The throughput of the K-best detector is computed as follows:

$$\Phi = \frac{N_T \times \log_2 M \times f_{\text{clk}} \times R}{N_{\text{clk}}} \, \text{Mbps},$$

where f_{clk} is the clock frequency of the detector, and R is the code rate, which is equal to one for the hard-detection case considered. For the fully-pipelined detector, the number of clock cycles required to generate a symbol vector, N_{clk}, after the pipeline is filled, is equal to one. Thus, at a clock frequency of 137 MHz, the detector achieves a throughput of 3288 Mbps, which makes it suitable to high-throughput standards such as the IEEE 802.11ac. As presented in Table 2, ASIC II achieves area and power consumption figures of 1467 kGE and 357 mW, respectively, which correspond with reductions in the area and power consumption by 16% and 38%, respectively, compared with the reference detector, ASIC I.

5.1 Comparison with State-of-the-Art

The proposed detector is compared with notable ASIC implementations of MIMO detectors in Table 2. All detectors are based on a 64-QAM 4×4 MIMO communication system. As expected, our implementation achieves a higher throughput than all the partially-pipelined detectors (i.e. detectors with symbol-vectors-per-cycle less than one), even at the moderate clock frequency of 137 MHz. Apart from Mondal et al. [22], our design employs the largest K value, which has beneficial effects on the BER, but is also partly responsible for the comparatively large area. In [3], only a subset of the children of a parent node is considered for the sorting similar to the proposed implementation. However, in that implementation, an approximate sorting scheme was used leading to more than a 3-dB SNR loss at a target BER of 10^{-3}. Furthermore, the work employed a folded architecture resulting in a modest throughput of 300 Mbps.

Huang and Tsai [13] and Mahdavi and Shabany [18] employ a similar fully-pipelined tree search as our implementation. In the case of Huang and Tsai [13], a low complexity is achieved by employing small non-constant values of K depending on the tree level. However, this results in an appreciable performance loss as highlighted in Fig. 1. Furthermore, the use of small K values makes the architecture less suitable for generating accurate reliability information on the detected bits in a soft-output implementation. It should also be mentioned that the area presented does not include the contribution of the channel matrices as is the case in the proposed implementation.

Mahdavi and Shabany [18] report a high scaled throughput of over 20 Gbps based on a complex-model K-best implementation. To reduce the complexity of the architecture, $\hat{y}_i$ and $r_{i,j}$ are scaled by $r_{i,i}$ outside the architecture in order to compute the complex-domain SE enumeration at each level. To achieve the scaling, both the real and imaginary parts of $\hat{y}_i$ and $r_{i,j}$ are divided by $r_{i,i}$. It should be noted that the impact of these extra-architectural overheads is not reflected in the results presented in Table 2. To implement the receiver unit with low power consumption and high throughput, it is important that the MIMO detector, as well as

interfacing architecture, is implemented with low complexity. In [18], it can easily be shown that the total number of divisions required by the architecture scales quadratically with the number of antennas. Overall, 20 real 16-bit divisions are required to detect one symbol vector for $N_T = 4$ using this architecture. Although the scaling of $r_{i,j}$ could be done infrequently if the channel is fairly stationary, the scaling of $\hat{y}$ must be carried out for every RSV, which could significantly impact the throughput and power consumption in a practical scenario. Furthermore, if the divisions are implemented using combinational logic, the impact on the area of the overall receiver unit is likely to be considerable.

Due to the use of a precomputed tabular enumeration, and the computation of the PED using (6), our implementation completely avoids multiplications at the symbol rate, which are required by [13] and [18]. In fact, if the QR decomposition is implemented using the CORDIC implementation of the Givens rotation [12], the signal detection could be realised completely multiplier free.

5.2 Cost of Pipelining

The proposed detector is implemented using a multi-stage architecture, where multiple tree searches are executed concurrently. To compute the hardware cost of pipelining, we implement a multi-stage detector with single-tree processing. In [15], it is shown that several unpipelined single-tree multi-stage (STMS) detectors can be interleaved to achieve a higher detection rate. The key advantage of the STMS detector is its low power consumption; however, its throughput is tied to the latency and the channel model employed.

We implement the STMS detector using the ST 65 nm technology, and it achieves a post-layout area of 790 kGE and a latency of 25 clock cycles. We can determine the pipelining cost by comparing the throughput-to-area ratios (TAR) of the pipelined detector to that of the unpipelined STMS detector as follows:

$$\text{Relative TAR} = \frac{\text{TAR of pipelined detector}}{\text{TAR of unpipelined detector}},$$

which gives the throughput advantage of the pipelined detector over the STMS detector per kilo gate equivalent of the area. Assuming the same clock frequency, the relative TAR is 13.46. This implies that given the same area, the fully-pipelined detector achieves a throughput of more than $13\times$ compared with the unpipelined STMS detector. Thus, we can conclude that despite the additional complexity incurred, pipelining is more hardware efficient than interleaving several unpipelined detectors in order to achieve gigabit data rates.

5.3 Complex Versus Real Fully-Pipelined K-Best Detector

In the following, we highlight the relative advantages of the real and complex model K-best detector with respect to different metrics. We conclude that in most scenarios, implementing the K-best detector using a real channel model is advantageous.

5.3.1 Performance

Both the real and complex K-best detectors are "suboptimal" algorithms with respect to achieving the ML bit error rate performance. Given the same value of K, however, the real K-best detector achieves a better BER performance as highlighted in Fig. 2. Intuitively, we can attribute

this performance discrepancy to the fact that since there are more branches per level in the complex channel model, more potential solutions will be discarded for the same value of K compared with the real channel model.

5.3.2 Complexity

Implementing the K-best detector using the real channel model has three major advantages with respect to the hardware complexity. Firstly, it simplifies the PED blocks by substituting complex-valued symbols with integers; secondly, the SE enumeration can be obtained using simple integral comparisons as illustrated in Fig. 3, and finally, since the complex K-best detector requires a larger K value to achieve the same BER as the real channel model, its complexity is further increased. There is yet no easy analytical method to determine which channel model to adopt for all communications requirements to achieve the lowest complexity. This can only be conclusively determined using actual hardware implementations based on the two models.

5.3.3 Throughput

Both the real and complex channel models can achieve a throughput of one symbol vector per clock cycle using a fully-pipelined approach. However, the simplified datapath of the real channel model is advantageous as there is better potential to achieve a higher maximum clock frequency, thereby, achieving a higher throughput. However, in single-tree architectures [3, 21], the shorter latency of the complex channel model is advantageous, as the throughput is now directly proportional to the latency. As a result, the real channel model is more attractive for applications where high throughput is the most critical requirement.

6 Conclusion

In this paper, we have presented the VLSI implementation of a fully-pipelined K-best detector based on a 65-nm CMOS process. The detector is based on a hybrid K-best algorithm, which utilises a successive interference cancellation detection at lower levels of the tree search to reduce the complexity of the detector compared with the conventional K-best algorithm, incurring a 0.3-dB SNR loss at a target BER of 10^{-3}. The implementation results indicate that the hybrid detector reduces the area and power consumption by approximately 16% and 38%, respectively, compared with a reference fully-pipelined K-best detector employing K-best detection at all levels of the tree search. Using a real channel model and tabular enumeration, the proposed implementation also eliminates the use of multiplication at the symbol rate, which helps to reduce the overall power consumption of the receiver unit. At a clock frequency of 137 MHz, the detector achieves a throughput of over 3 Gbps making it suitable to low-latency wireless standards such as the IEEE 802.11ac. A potential area for future research is to implement the detector using a different sorting algorithm, such as the multi-cycle winner path extension, in order to further explore latency, power and area tradeoffs.

References

1. J. G. Andrews, S. Buzzi, Wan Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, and J.C. Zhang. What Will 5g Be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, 2014.
2. L. G. Barbero and J.S. Thompson. A Fixed-Complexity MIMO Detector Based on the Complex Sphere Decoder. In *IEEE 7th Workshop on Signal Processing Advances in Wireless Communications, 2006. SPAWC '06*, pages 1–5, 2006.

3. I. A. Bello, B. Halak, M. El-Hajjar, and M. Zwolinski. VLSI implementation of a scalable K-best MIMO detector. In *2015 15th International Symposium on Communications and Information Technologies (ISCIT)*, pages 281–286, October 2015.

4. I. A. Bello, B. Halak, M. El-Hajjar, and M. Zwolinski. Hardware Implementation of a Low-Power K-Best MIMO Detector Based on a Hybrid Merge Network. In *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 191–197, July 2018.

5. I. A. Bello, B. Halak, M. El-Hajjar, and M. Zwolinski. A Survey of VLSI Implementations of Tree Search Algorithms for MIMO Detection. *Circuits, Systems, and Signal Processing*, pages 3644–3674, 2015.

6. A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei. VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE Journal of Solid-State Circuits*, 40(7):1566 – 1577, 2005.

7. X. Chen, G. He, and J. Ma. VLSI Implementation of a High-Throughput Iterative Fixed-Complexity Sphere Decoder. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(5):272–276, 2013.

8. R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

9. J. Fink, S. Roger, A. Gonzalez, V. Almenar, and V. M. Garciay. Complexity assessment of sphere decoding methods for MIMO detection. In *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 9–14, 2009.

10. Z. Guo and P. Nilsson. Algorithm and implementation of the K-best sphere decoding for MIMO detection. *IEEE Journal on Selected Areas in Communications*, 24(3):491–503, 2006.

11. B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *IEEE Transactions on Signal Processing*, 53(8):2806–2818, 2005.

12. B. Hassibi. An efficient square-root algorithm for BLAST. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II737–II740, 2000.

13. M.-Y. Huang and P.-Y. Tsai. Toward Multi-Gigabit Wireless: Design of High-Throughput MIMO Detectors With Hardware-Efficient Architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61 (2):613–624, 2014.

14. C. Ju, J. Ma, C. Tian, and G. He. VLSI implementation of an 855 Mbps high performance soft-output K-Best MIMO detector. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2849–2852, May 2012.

15. T.-H. Kim and I.-C. Park. Small-Area and Low-Energy-Best MIMO Detector Using Relaxed Tree Expansion and Early Forwarding. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(10):2753–2761, 2010.

16. L. Liu, J. Lofgren, and P. Nilsson. Area-Efficient Configurable High-Throughput Signal Detector Supporting Multiple MIMO Modes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(9):2085–2096, 2012.

17. M. Mahdavi and M. Shabany. Novel MIMO Detection Algorithm for High-Order Constellations in the Complex Domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(5):834–847, 2013.

18. M. Mahdavi and M. Shabany. A 13 Gbps, 0.13 μm CMOS, Multiplication-Free MIMO Detector. *Journal of Signal Processing Systems*, 88(3):273–285, 2017.

19. N. Moezzi-Madani and W. R. Davis. Parallel merge algorithm for high-throughput signal processing applications. *Electronics letters*, 45(3):188–189, 2009.

20. N. Moezzi-Madani and W. R. Davis. High-throughput low-complexity MIMO detector based on K-best algorithm. In *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, pages 451–456, 2009.

21. N. Moezzi-Madani, T. Thorolfsson, P. Chiang, and W. R. Davis. Area-Efficient Antenna-Scalable MIMO Detector for K-best Sphere Decoding. *Journal of Signal Processing Systems*, 68(2):171–182, 2012.

22. S. Mondal, A. Eltawil, C.-A. Shen, and K. N. Salama. Design and Implementation of a Sort-Free K-Best Sphere Decoder. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(10):1497–1501, 2010.

23. A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire. A unified framework for tree search decoding: rediscovering the sequential decoder. *IEEE Transactions on Information Theory*, 52(3):933–953, 2006.

24. E. Perahia and M. X. Gong. Gigabit wireless LANs: an overview of IEEE 802.11 ac and 802.11 ad. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(3):23–33, 2011.

25. C. P. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In *Math. Programming*, pages 181–191, 1993.

26. M. Shabany and P. G. Gulak. A 0.13 μm CMOS 655mb/s 4x4 64-QAM K-Best MIMO detector. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 256–257, 2009.

27. M. Shabany and P. G. Gulak. A 675 Mbps, 4x4 64-QAM K-Best MIMO Detector in 0.13 μm CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(1):135–147, 2012.

28. S. Sugiura, S. Chen, and L. Hanzo. MIMO-Aided Near-Capacity Turbo Transceivers: Taxonomy and Performance versus Complexity. *IEEE Communications Surveys Tutorials*, 14(2):421–442, 2012.

29. P.-Y. Tsai and X.-C. Lin. Improved K-best sphere decoder with a look-ahead technique for multiple-input multiple-output systems. In *International Symposium on Intelligent Signal Processing and Communications Systems, 2008. ISPACS 2008*, pages 1–4, February 2009.
30. M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner. K-best MIMO detection VLSI architectures achieving up to 424 Mbps. In *2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006. Proceedings*, pages 1151–1154, 2006.
31. A. Wiesel, X. Mestre, A. Pages, and J. R. Fonollosa. Efficient implementation of sphere demodulation. In *4th IEEE Workshop on Signal Processing Advances in Wireless Communications, 2003. SPAWC 2003*, pages 36–40, 2003.
32. P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela. V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel. In *1998 URSI International Symposium on Signals, Systems, and Electronics, 1998. ISSSE 98*, pages 295–300, October 1998.
33. K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow. A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels. In *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, volume 3, pages III–273–III–276 vol.3, 2002.
34. B. Wu and G. Masera. A Novel VLSI Architecture of Fixed-Complexity Sphere Decoder. In *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, pages 737–744, 2010.
35. D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer. MMSE extension of V-BLAST based on sorted QR decomposition. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 1, pages 508–512 Vol.1, October 2003.