Hardware Implementation of a Low-Power K-best MIMO Detector Based on a Hybrid Merge Network

Ibrahim A. Bello*, Basel Halak†, Mohammed El-Hajjar† and Mark Zwolinski†
*Department of Computer Engineering, Ahmadu Bello University, Nigeria
ibrahimbello@abu.edu.ng

†Electronics and Computer Science, University of Southampton, United Kingdom
{bh9, meh, mz}@ecs.soton.ac.uk

***Abstract*—Multiple input multiple output (MIMO) technology is anticipated to play a key role in future wireless communications systems. However, one of the main challenges of MIMO technology is the high complexity of the signal detection, which results in a high power consumption at the MIMO receiver. In this paper, we present the hardware implementation of a K-best detector based on a single-stage architecture, targeted at low-rate and low-power applications. To achieve a low complexity, we optimise the sorting stage of the detector by systematically eliminating redundant comparators. Furthermore, the sorter incorporates different merge algorithms at selected stages in order to reduce the total comparator count. For a 64-QAM 4 × 4 MIMO system, the detector achieves a power consumption of 34 mW using the STMicroelectronics 65 nm CMOS library, which compares favourably with similar works from the literature.**

***Index Terms*—K-best, MIMO, Wireless communication algorithms, Low power**

I. Introduction

Multiple input multiple output (MIMO) technology is fast becoming an indispensable component of future wireless communications systems. When used in the spatial multiplexing mode, MIMO can significantly increase the channel capacity by simultaneously transmitting multiple data streams [1]. MIMO can also be used to improve the bit-error rate (BER) performance of the signal detection by employing spatial diversity [2].

Despite these advantages, the use of multiple antennas imposes an increased complexity in communications systems, especially at the receiver. The maximum likelihood (ML) detector offers the best BER performance [3], but it requires an exhaustive search of all possible transmitted symbol vectors, which makes it unsuitable for VLSI implementation.

Several detection algorithms have been investigated in recent years as alternatives to the ML detector. The K-best algorithm [4] offers a near-ML performance, with a fixed complexity, which has made it the algorithm of choice for the implementation of high-performance MIMO detection in hardware. Unfortunately, the K-best algorithm is beset with high power consumption and complexity challenges, especially when implemented in a pipelined fashion [5], [6], [7]. For applications requiring low data rates, such as the "Internet of Things" [8], such a high complexity is unnecessary, as such, lower-complexity architectures need to be investigated.

The aim of this work is to implement a K-best detector for low-power and low-throughput applications. To this end, we undertake the following objectives:

1) Implement a novel merge network to carry out the K-best sorting with a low complexity and latency
2) Compare the VLSI implementation of the proposed merge network with other common K-best sorting algorithms
3) Implement the K-best detector using a low-complexity single-stage architecture

The paper is organised as follows. In Section II, the MIMO system model and signal detection using the K-best algorithm are described. In Section III, the proposed merge network implementation is presented. In Section IV, the architecture of the proposed K-best algorithm is presented. In Section V, the results of our implementation are presented and compared with notable single-stage architectures from the literature. Finally, the paper is concluded in Section VI.

The following notations are used in the paper. $\Re\{\cdot\}$ and $\Im\{\cdot\}$ extract 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 $[A_{i,j}, A_{i,j+1}, \ldots, A_{i,k}]$.

II. Signal Detection

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

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

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. A QR decomposition can also be performed on the channel matrix as follows:

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

where $\mathbf{H} = \mathbf{Q}\mathbf{R}$, $\hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$, $\mathbf{Q}$ 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

Algorithm 1 K-best algorithm

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

while $i \geq 1$ **do**
  Compute $T_i \, \forall \, s_{i,j,k}$
  $\mathcal{K}_{i,1:K} \leftarrow$ KBEST($s_{i,j,k}$)
  Update $\mathcal{K}_{j,1:K}$ according to sorted $T_i \, \forall \, j > i$
  $i \leftarrow i - 1$
end while

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

receiver i.e., $N_T = N_R$. A real-valued decomposition (RVD) can be performed to transform (1) as follows:

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

which transforms the complex constellation set into the odd-valued integer set, $\mathcal{D}$, defined as $\{-\sqrt{M}+1, \ldots, \sqrt{M}-1\}$, where M is the modulation order. At the receiver, the detector attempts to obtain an estimate of the transmitted symbols, $\hat{\mathbf{s}}$, with a low bit error rate. This procedure is described in more detail in the next sections.

A. K-best Algorithm

The K-best algorithm performs the MIMO signal detection by carrying out a forward-only tree search [4]. The number of antennas is considered to be the levels of the tree, while the constellation points are considered to be the branches of the tree. Each constellation point at a higher level of the tree is considered to be a "parent" to all emerging branches. However, in this work, the RVD is considered for the tree search, which results in doubling the tree depth. The K-best algorithm is shown in Algorithm 1, where $\mathcal{K}$ is a $2N_T \times K$ matrix representing the best K solutions, $s_{i,j,k}$ is the jth child of the kth parent at the ith level. The `KBEST` function sorts the candidates according to their partial Euclidean distances (PEDs) and selects the top K results. At the end of the detection, the path with the least metric, $\mathcal{K}_1$, is submitted as the estimate of the transmitted symbol vectors, $\hat{\mathbf{s}}$. The PED of a constellation point, T_i, at a level i is computed as follows:

$$T_i = T_{i+1} + |b_i - r_{i,i} s_i| \,, \tag{3}$$

where

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

B. Reduced-Complexity K-best Detector

The number of children per parent node in the tree search can be utilised as a parameter to reduce the complexity of the K-best algorithm [9]. If the number of children per parent is denoted by λ, then the K-best detection can be characterised as KB(K, λ), where KB$(K, \sqrt{M})$ represents the original K-best algorithm. If $\lambda = 1$, then the K-best detector essentially reduces to a successive-interference-based detection, where each of the K paths extends only its best child.

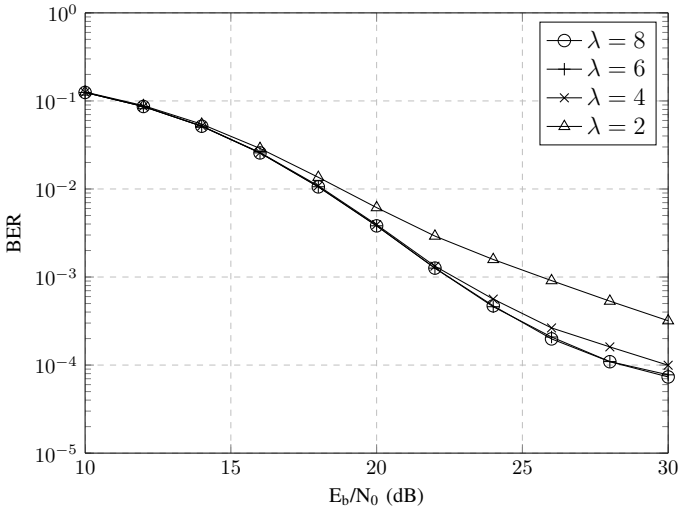

Fig. 1: BER performance of the K-best algorithm with K = 16 using different values of λ

A similar reduced-complexity K-best detector was proposed by Kim and Park [10], which was based on the orthogonal real-valued decomposition (ORVD) channel model [11]. As a result of the channel model employed in that work, two layers are processed in parallel, which results in almost a $10\times$ increase in the number of PED increment computations compared with the proposed implementation. Furthermore, the ORVD incurs a BER penalty, as the resulting tree search is not equivalent to the conventional RVD-based scheme [12].

Figure 1 shows the BER simulation for the proposed K-best detector using K = 16 for different values of λ. The K-best detector using $\lambda = 4$ displays similar performance to the original K-best detector up to a BER of 10^{-3}. On the other hand, the K-best detector using $\lambda = 2$ suffers a significant SNR loss of approximately 3 dB compared with the original K-best detector at a BER of 10^{-3}. The result of Fig. 1 is noteworthy, as the choice of λ has an impact on the sorting complexity as will be discussed in the next section.

III. Sorting

Sorting plays a prominent role in many digital signal processing applications. In the K-best algorithm, sorting is required to select the best candidates at each level of the tree search. The choice of sorting algorithm has an impact on the complexity and performance of the K-best detector. Single-cycle sort algorithms are suitable for high-throughput applications, while multi-cycle sort algorithms typically incur a lower complexity but at the expense of a longer latency. In this work, a single-cycle merge algorithm is adopted for the sorting unit.

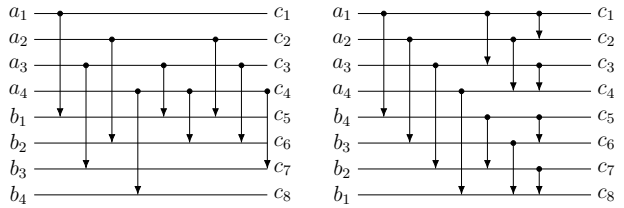

Fig. 2: Illustration of Batcher's merge algorithms: (a) odd-even merge and (b) bitonic merge.

A. Merge Algorithms

Traditional sorting algorithms, such as the bubble sort, require a large number of clock cycles, which has an undesirable impact on the attainable throughput. Merge algorithms, on the other hand, sort a data sequence in a single step and as such the desired sorted result can be obtained within one clock cycle. Merge algorithms typically employ a "divide-and-conquer" approach to sorting, by dividing the data to be sorted into sublists and then iteratively merging the sorted sublists in parallel. After each iteration, a larger sublist is formed and the merge process is repeated until a single list is obtained, which corresponds to the sorted result. In this way, a merge network is formed whose depth is proportional to the number of sublists at the input. In this paper, we will limit ourselves to the well-known Batcher's merge networks [13], which are discussed in the subsequent sections.

1) Odd-Even Merge: Given two length-N lists, $\mathbf{a} = [a_1, a_2, \ldots, a_N]$ and $\mathbf{b} = [b_1, b_2, \ldots, b_N]$, where $a_1 < a_2 < \ldots < a_N$ and $b_1 < b_2 < \ldots < b_N$, the odd-even sorter splits the two lists into their odd and even-indexed components and sorts them independently by comparing two items at a time and then swapping them if they are out of place. The first item of the odd-index merge is also the first item of the final result, while the last item of the even-index merge is the last item of the final result. After the odd and even lists have been sorted, the odd-even sorter iteratively compares the $(i+1)$th item of the odd-index merge with the ith element of the even-index merge to get the remaining items of the final length-$2N$ result.

By comparing two sublists at a time, a larger merge network with 2^p inputs (where p is some integer) can be constructed. Figure 2a illustrates the odd-even merge for two 4-item sublists, where each arrow represents a compare-and-exchange operation. The top and tip of each arrow correspond with the smaller and larger number of the comparison respectively.

2) Bitonic Merge: Another parallel merge algorithm proposed by Batcher [13] is the bitonic merge (BM), which sorts one ascending list and one descending list, otherwise known as a bitonic sequence. Thus, $a_1, a_2, \ldots, a_N, b_N, b_{N-1}, \ldots, b_1$ forms a bitonic sequence if $a_1 < a_2 < \ldots < a_N$ and $b_N > b_{N-1} > \ldots > b_1$. If two lists are constructed as follows:

$$c_{\text{min}} = \min(a_1, b_N), \min(a_2, b_{N-1}), \ldots, \min(a_N, b_1)$$
$$c_{\text{max}} = \max(a_1, b_N), \max(a_2, b_{N-1}), \ldots, \max(a_N, b_1), \quad (4)$$

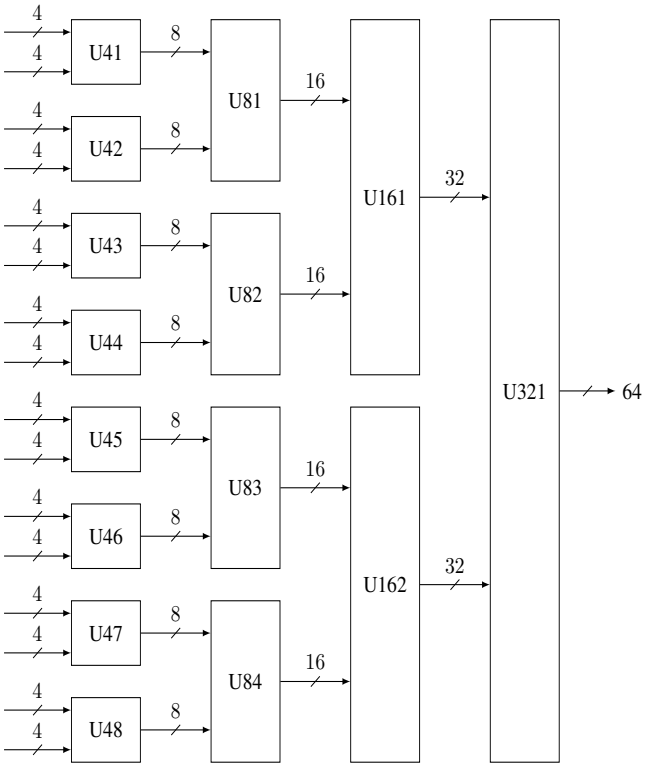

Fig. 3: Full unoptimised merge network for 64 candidates. All the candidates are included in the final sorted result and the best K candidates are taken to the next level.

then it can be shown that all the elements in the first list are less than the elements of the second list. Furthermore, each of c_{min} and c_{max} is bitonic. After the sequence is split into c_{min} and c_{max}, the bitonic sorter compares two elements of each sublist at a time, swapping them if they are out of place. This process is continued iteratively until the final length-$2N$ list is obtained. The bitonic merge for two 4-item sublists is illustrated in Fig. 2b. Unlike the odd-even merge, which has unequal paths from the inputs to the outputs, the input-output lines of the bitonic merge all have equal lengths. However, the bitonic sorter requires more compare-and-exchange elements for a given input sequence, which increases its complexity in hardware.

B. Implementation of the Merge Network

In this work, the Batcher's merge networks will be adopted, as they produce the sorted result within one clock cycle. The odd-even merge is attractive for constructing the merge network due to the fewer comparators required compared to the bitonic merge. The merge network sorts the candidates in pairs of two λ-length sublists, where each candidate is organised as $(s_{i,j,k}, T_{i,j,k}, k)$, where k denotes the parent path, $s_{i,j,k}$ is the jth child of the kth parent after Schnorr-

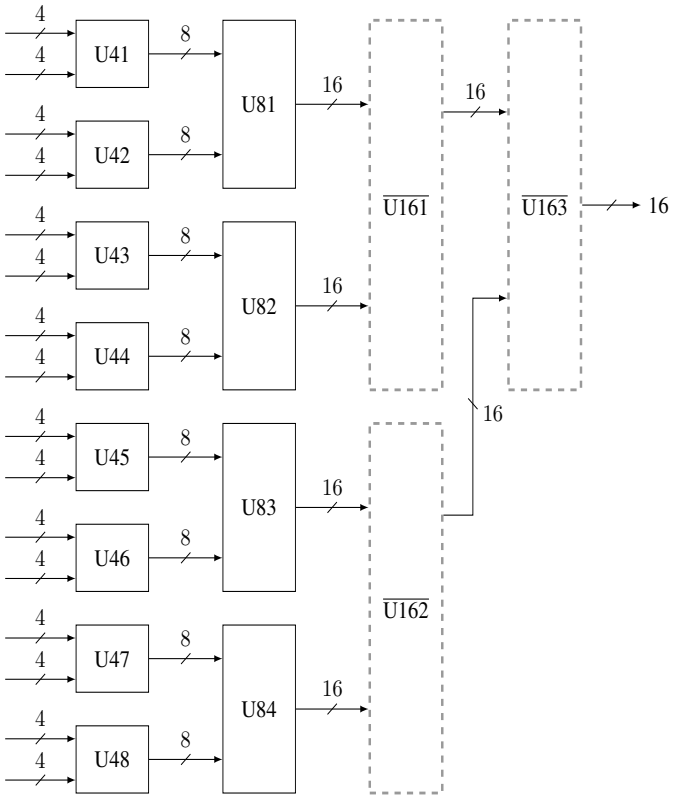

Fig. 4: Optimised merge network for 64 candidates. The merge units in the third and final stages are modified to produce only the top 16 results, while discarding the bottom results.

Euchner enumeration [14], and $T_{i,j,k}$ is its corresponding metric. Before the merge operation at the current level, k simply takes a value from the ascending sequence $1, 2, ..., K$. After the merge operation, k is updated according to the indices of the sorted PEDs at the current level.

Figure 3 illustrates the merge network for 16 sublists each comprising four elements. The merge units labelled U4X, U8X, U16X and U32X denote merge units for 4×4, 8×8, 16×16 and 32×32 inputs respectively. With $K = 16$, the candidates are merged in pairs using eight U4X units operating in parallel, and the outputs are successively doubled at every stage until the final 64 sorted result is obtained. Thereafter, the upper 16 results are selected and forwarded to the next level. The operation of the Batcher's merge network makes it more convenient to adopt K values that are powers of two; however, it is possible to construct a merge network with non-power-of-two K values by simple architectural modifications. For example, to construct a merge network for $K = 10$, U46, U47, U48 and U84 can be discarded and the bottom inputs to U83 and U162 can be replaced with dummy candidates having $T_i = \infty$. These dummy candidates will be automatically

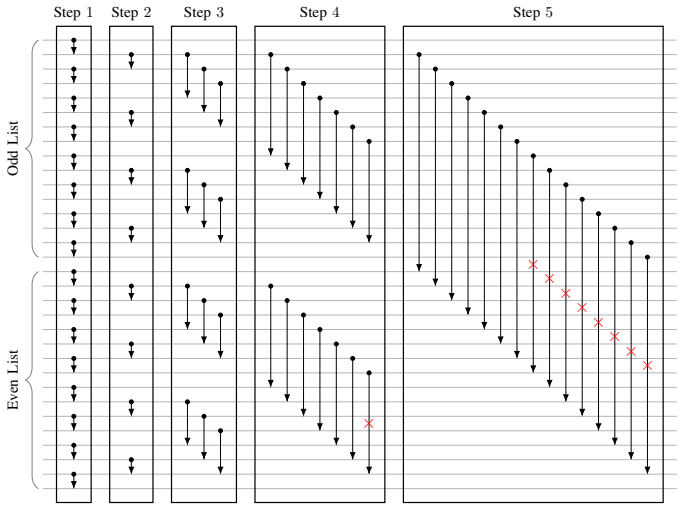

Fig. 5: Modified U16 using the odd-even merge. Only one comparator can be eliminated from the bottom half of the 4th step given K = 16.

relegated to the bottom of the 64-length sorted output at the end of the merge operation. It needs to be mentioned that other than the choice of $\lambda < \sqrt{M}$, and fixed-point quantisation effects, the proposed merge network achieves an exact sorting of the candidates.

1) Area Optimisation: The merge network in Fig. 3 includes several redundant candidates in the final sorted result, which will eventually get discarded and play no further role in the detection process. This is inefficient and leads to an unnecessary increase in the hardware complexity of the detector. Since only $K = 16$ candidates are required, U321 can be replaced with a simpler U16 unit as shown in Fig. 4. Similarly, U161 and U162 can be replaced with simpler U16 equivalents having only 16 outputs, instead of the 32 outputs in the original merge network. This also has a timing advantage, as the U16 element requires one less comparator stage compared with U32.

To construct the optimised U16 unit, all the comparators that do not contribute to the final output are eliminated. Interestingly, the bitonic sorter requires fewer comparators to implement the optimised U16 than the odd-even sorter. This is due to the unique property of the bitonic sorter whereby the upper and lower halves of the sorted result are generated in the first step of the merge process as shown in (4). Therefore, the lower half can be discarded early in the sorting, and the subsequent sort operations can be carried out on the upper half only. By contrast, although the odd-even sorter requires fewer comparators in general, the two halves of the final sorted result are only determined in the last stage.

The modified U16 unit, implemented using the odd-even and bitonic sorters, is shown in Figs. 5 and 6 respectively, showing the individual comparator operations. While the bitonic U16 unit is able to eliminate up to 32 comparators, the odd-even U16 unit is only able to eliminate 9 comparators.

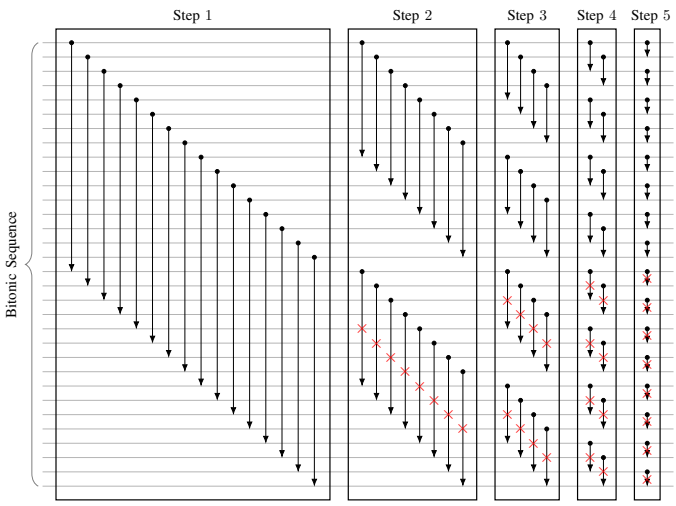

Fig. 6: Modified U16 using the bitonic merge. All the comparators in the lower halves of steps 2 to 5 can be eliminated using the bitonic algorithm given K = 16.

TABLE I: Comparison of full and optimised bitonic and odd-even merge networks for K = 16 and λ = 4

Sorter	Bitonic Merge		Odd-Even Merge		Hybrid
Design	Full	Optimised	Full	Optimised	Optimised
Area [kGE]	114.7	72.7	93.2	69.7	62.6
Comparators	576	432	463	340	316

Thus, a hybrid merge network can be constructed with a bitonic merge for the third and final stages and odd-even merge for the remaining stages. The optimised U16 unit with 16 outputs is denoted by $\overline{\text{U16}}$. Overall, the hybrid merge unit achieves an area saving of approximately 30% compared with the unoptimised network using odd-even merge for all the stages. The optimised hybrid merge unit is compared with the unoptimised bitonic and odd-even merge networks in Table I. The area is given in kilo-gate equivalent (kGE).

2) Pipelining the Merge Network: The inputs to the proposed merge network described in the previous section pass through a total of 12 comparators in series before emerging at the output. This results in a large combinational delay, which limits the attainable clock frequency of the detector. The combinational delay of the merge network can be reduced by inserting one or more pipeline registers at suitable locations, thereby splitting the merge network into two or more smaller merge networks. This will however entail an increase in the area consumption of the design. Furthermore, the latency of the detection for one symbol vector is increased by one clock cycle. However, this is compensated by an increased clock frequency, which allows a higher throughput to be achieved overall. If pipeline registers are inserted after the first step of the U16 blocks in the third stage of the merge network in Fig. 4, then the merge network can be split into two nearly identical merge networks with seven and five comparators.

TABLE II: Comparison of hardware implementations of different sorting algorithms

Sorter	BS	DS	RS	HM	P-HM
Area [kGE]	40.8	16.2	5	62.6	72.4
T_{crit} [ns]	1.78	4.43	3.43	9.50	4.25

3) Results and Discussion: Table II compares the hardware implementation results of the proposed hybrid merge (HM) network with other sorting algorithms, based on a 65 nm CMOS technology. The relaxed sorter [9] incurs the smallest area consumption, however, this is at the expense of a degradation to the BER performance. Despite its simplicity, the bubble sorter incurs a relatively large area, which is as a result of the registers required for storing the temporary sorting results as well as the input elements. Furthermore, a counter needs to be created to keep track of the number of iterations, which is the largest among the algorithms compared. The distributed sort algorithm [15] was implemented with the assumption that all the child nodes are already enumerated via a table lookup. In order to find the MIN element in each iteration, assuming $K = 16$, two elements are compared at a time, resulting in a critical path of eight comparators in series. This makes the distributed sort to have a longer critical path length than the bubble sort, which only needs to compare two elements in each cycle. As expected, the proposed hybrid merge incurs the largest area consumption and longest critical path. The pipelined hybrid merge (P-HM) increases the area by 15.5% while reducing the critical path length by 55.3%, which is almost the same as that of the distributed sort algorithm.

The comparatively large area of the bubble sort, relative to the distributed and relaxed sort algorithms, as well as its large latency, further underscores its unsuitability to the K-best algorithm. For small constellation sizes, however, such as 4-QAM, the bubble sort is attractive since only a few iterations are required to get the final sorted result. The distributed sorter is suited to larger constellation sizes since the number of clock cycles required depends on K only. However, it should be noted that larger constellation sizes tend to require larger values of K since the BER performance degrades with the modulation order. Where area is not critical, the hybrid merge is attractive as it achieves an exact sorting and also produces the sorted results within a single cycle. As noted, the hybrid merge can be pipelined in order to reduce the critical path at the expense of a larger area and additional clock cycles to obtain the sorted results. In the next section, the hardware implementation for the K-best detector, based on the pipelined hybrid merge network, will be presented.

IV. K-Best Architecture

In this paper, the K-best detector will be implemented using a single-stage architecture, where a single core is reused for all the tree levels in a recursive manner. The simplified architecture is shown in Fig. 7, showing the K-best candidates at the ith level fed back from the merge network to the PED computation blocks. $T_{i+1,k}$ represents the accumulated PED

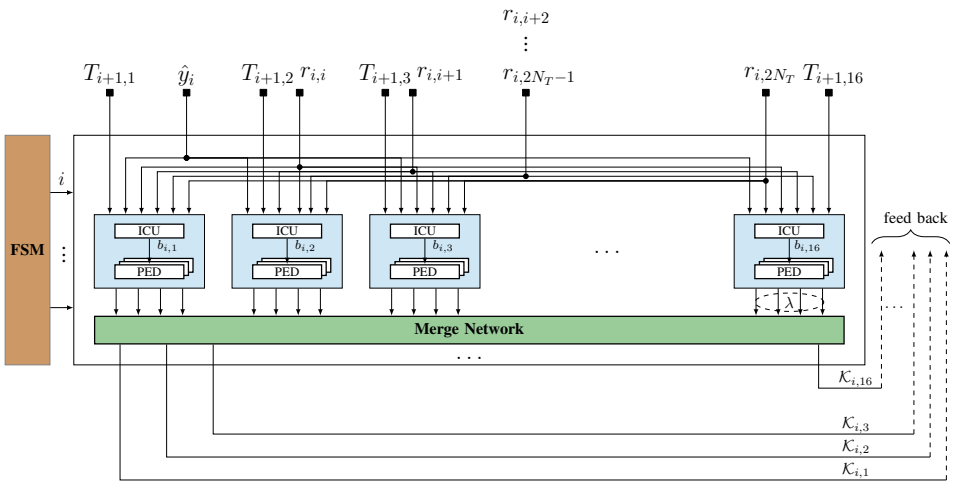

Fig. 7: Simplified architecture of the proposed single-stage K-best detector

up to the $(i+1)$th level for the kth path, while $b_{i,k}$ represents b_i computed for the kth path. The processing element consists of K expansion units, each of which consists of an interference cancellation unit (ICU), which computes b_i, and a PED unit which computes T_i. The outputs of each of the expansion unit are the λ best children of the kth path selected after enumeration from a lookup table [16]. A single finite state machine (FSM) determines the current level of the tree search, i, and issues other control signals to the datapath of the K-best detector.

A single-stage architecture ensures a much lower power consumption compared with fully-pipelined implementations [5], [6], which may utilise much more resources than required for simple low-throughput applications. The PED in the proposed architecture is computed using the ℓ^1-norm approximation presented in [17]. The signal and channel inputs, $\hat{\mathbf{y}}$ and $\mathbf{R}$, are represented using 14-bit signed Q-format representations, which are determined after extensive simulations in MATLAB.

V. Results and Discussion

The results of the proposed detector are compared with other single-stage K-best implementations from the literature as shown in Table III. All the implementations are targeted at a MIMO system employing 64-QAM. The throughputs and power consumption results are scaled to the 65 nm CMOS technology at a supply voltage of 1.05 V using a similar approach to [21]. The area consumption is obtained after a place-and-route step, while the power is estimated with the aid of switching activities captured during post-synthesis gate-level simulations.

TVLSI'07 [18] adopts a radius-based pruning strategy and as such does not have a fixed throughput. The implementation achieves a comparable throughput to the proposed detector; however, our implementation outperforms it significantly in terms of the power consumption and energy-efficiency. TVLSI'10 [19] employs three detector cores in order to increase the throughput. With three cores, our implementation achieves more than $\times 3$ of the throughput of TVLSI'10 [19], while reducing the area consumption by approximately half.

DATE'10 [20] achieves a throughput of 214 Mbps for a 2×2 antenna configuration with a relatively small area of 24 kGE. Our implementation requires 14 clock cycles to completely detect one symbol vector in the 2×2 configuration, resulting in a slightly improved throughput performance of 234 Mbps. With the pessimistic assumption that the proposed implementation consumes the same power in 2×2 as the 4×4 case, our implementation will achieve an energy-per-bit of 146 pJ/bit in the 2×2 configuration, which is not significantly higher than that of DATE'10 [20]. Although ISCIT'15 [9] achieves a higher throughput than the proposed implementation, this is at the cost of a reduced BER performance due to the relaxed sorting algorithm that was employed.

VI. Conclusion

In this paper, we have presented the VLSI implementation of a low-power K-best MIMO detector, targeted at low-throughput applications such as the "Internet of Things". In order to reduce the complexity and power consumption of the detector, a single-stage architecture is presented, where a single processing element is reused for all levels of the tree search. We also presented the implementation of a novel low-complexity hybrid merge algorithm combining features of the odd-even and bitonic merge algorithms. For a 64-QAM 4×4 MIMO system, our implementation achieves a throughput of 109 Mbps and a power consumption of 34 mW using the STMicroelectronics 65 nm CMOS library. To achieve a higher throughput, several cores can be instantiated in an interleaved fashion. A possible area for future research will include optimising each individual comparator in the merge network so as to reduce the latency of the detection without the use of pipeline registers.

TABLE III: Implementation results of single-stage K-best detectors for 64-QAM MIMO

Reference	This work		TVLSI'07 [18]†	TVLSI'10 [19]	DATE'10 [20]	ISCIT'15 [9]
# Cores	1	3	N/A	3	1	1
K	16	16	64	64	10	10
MIMO	4×4	4×4	4×4	4×4	2×2	4×4
Detection	Hard	Hard	Soft	Hard	Soft	Hard
Tech. [nm]	65	65	130	65	130	65
V_{dd} [V]	1.05	1.05	1.2	1	1.5	1.05
Area [kGE]	285	855	N/A	1760	24	206
f_{clk} [MHz]	137	137	270	158	287	287
Φ [Mbps]	109	327	100	100	214	300
P [mW]	34	103	919	238	27	27
E_{bit} [pJ/bit]	312	312	9190	2376	125	91

† Throughput is SNR dependent

REFERENCES

[1] D. Gesbert, M. Shafi, D.-s. Shiu, P. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281 –302, Apr. 2003.

[2] M. El-Hajjar and L. Hanzo, "Multifunctional MIMO systems: A combined diversity and multiplexing design perspective," *IEEE Wireless Communications*, vol. 17, no. 2, pp. 73–79, 2010.

[3] M. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[4] K.-w. Wong, C.-y. Tsui, R.-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*, vol. 3, 2002, III–273–III–276 vol.3.

[5] 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*, vol. 61, no. 2, pp. 613–624, Feb. 2014.

[6] M. Mahdavi and M. Shabany, "A 13 gbps, 0.13 μm CMOS, multiplication-free MIMO detector," *J Sign Process Syst*, pp. 1–13, Jun. 6, 2016.

[7] Ibrahim Bello, "Energy-efficient architectures for multi-gigabit MIMO detection," PhD thesis, University of Southampton, Jul. 2017.

[8] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5g era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.

[9] 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)*, Oct. 2015, pp. 281–286.

[10] 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*, vol. 57, no. 10, pp. 2753–2761, 2010.

[11] L. Azzam and E. Ayanoglu, "Reduced complexity sphere decoding for square QAM via a new lattice representation," in *IEEE Global Telecommunications Conference, 2007. GLOBECOM '07*, Nov. 2007, pp. 4242–4246.

[12] T.-H. Liu, "Comparisons of two real-valued MIMO signal models and their associated ZF-SIC detectors over the rayleigh fading channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 12, pp. 6054–6066, Dec. 2013.

[13] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30-May 2, 1968, spring joint computer conference*, ACM, 1968, pp. 307–314.

[14] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems.," in *Math. Programming*, 1993, pp. 181–191.

[15] M. Shabany and P. Gulak, "Scalable VLSI architecture for k-best lattice decoders," in *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, 2008, pp. 940–943.

[16] A. Wiesel, X. Mestre, A. Pages, and J. Fonollosa, "Efficient implementation of sphere demodulation," in *4th IEEE Workshop on Signal Processing Advances in Wireless Communications, 2003. SPAWC 2003*, 2003, pp. 36–40.

[17] 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*, vol. 40, no. 7, pp. 1566 –1577, Jul. 2005.

[18] S. Chen, T. Zhang, and Y. Xin, "Relaxed k-best MIMO signal detector design and VLSI implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 328–337, 2007.

[19] S. Mondal, A. Eltawil, C.-A. Shen, and K. Salama, "Design and implementation of a sort-free k-best sphere decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 10, pp. 1497–1501, 2010.

[20] N. Moezzi-Madani, T. Thorolfsson, and W. Davis, "A low-area flexible MIMO detector for WiFi/WiMAX standards," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, Mar. 2010, pp. 1633–1636.

[21] F. Borlenghi, E. Witte, G. Ascheid, H. Meyr, and A. Burg, "A 772mbit/s 8.81bit/nJ 90nm CMOS soft-input soft-output sphere decoder," in *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian*, Nov. 2011, pp. 297–300.