A Two-Dimensional Sorting Algorithm for High-Throughput K-best MIMO Detection

Wen Fan and Amir Alimohammad

***Abstract*—This brief presents a novel two-dimensional parallel sorting algorithm for high-throughput K-best detectors utilized in multiple-input multiple-output (MIMO) systems. The proposed sorting algorithm enhances the throughput by sorting a data set in parallel and avoids the relatively long latency of the traditional algorithms. This is especially important in MIMO systems utilizing high-order modulation schemes. We used our sorting algorithm in a K-best detector with variable *K* values at different layers of the search tree, which improves the bit error rate performance and reduces the computational complexity significantly. The detector using our sorting algorithm is designed and implemented in TSMC 90-*nm* CMOS technology for 4×4 64-QAM MIMO systems. Operating at 200 MHz, the detector's throughput is 1.2 Gbps. Its equivalent gate count is 182 K.**

I. Introduction

THE K-best detection algorithm has been widely considered as a promising technique for multiple-input multiple-output (MIMO) systems as it provides a near-optimal performance with a significantly lower computational complexity compared to maximum likelihood (ML) detection [1]–[6]. In a spatial multiplexing MIMO system with N_T transmit antennas and N_R receive antennas, the real-valued system model can be written as $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$, where $\mathbf{y}$ denotes the $2N_R \times 1$ received signal vector, $\mathbf{H}$ denotes the $2N_R \times 2N_T$ channel matrix, $\mathbf{s}$ denotes the $2N_T \times 1$ transmitted signal vector, and $\mathbf{n}$ is a $2N_R \times 1$ noise vector with independent identically distributed circularly-symmetric complex Gaussian components with zero mean and unit variance. The task of the MIMO detector is to estimate the symbol vector $\mathbf{s}$ from the received signal vector $\mathbf{y}$. By performing QR decomposition on the channel matrix $\mathbf{H} = \mathbf{Q}\mathbf{R}$, an estimate $\hat{\mathbf{s}}$ for each transmitted space-time (ST) symbol $\mathbf{s}$ can be rewritten as:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\Omega^{2N_T}}{\arg\min} \|\mathbf{Q}^H\mathbf{y} - \mathbf{R}\mathbf{s}\|^2 = \underset{\mathbf{s}\in\Omega^{2N_T}}{\arg\min} \sum_{i=1}^{2N_T} \left| \hat{y}_i - \sum_{j=i}^{2N_T} r_{ij} s_j \right|^2 ,$$

where Ω is the set of real-valued constellation points, $\mathbf{Q}$ and $\mathbf{R}$ are $2N_R \times 2N_T$ unitary and upper triangular matrices, respectively, $\hat{y}_i$ is the i-th element of vector $\hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$, r_{ij} is the entry in the i-th row and j-th column of the matrix $\mathbf{R}$, and s_j is the j-th element of the vector $\mathbf{s}$. The minimum is sought over all possible real-valued $2N_T$-element ST symbols $\mathbf{s} \in \Omega^{2N_T}$. The K-best algorithm traverses the search tree from the root to leaves by expanding the K-best candidates from the upper layers. For a M-QAM MIMO system, $K\sqrt{M}$ path metrics should be computed in each layer. The partial Euclidean distance (PED) $T_i\left(\mathbf{s}^{(i)}\right)$ for the $\mathbf{s}^{(i)}$ tree node can be computed recursively by traversing down from the $2N_T$-th layer as $T_i\left(\mathbf{s}^{(i)}\right) = T_{i+1}\left(\mathbf{s}^{(i+1)}\right) + \left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2$, where $i = 2N_T, 2N_T-1, \ldots, 1$, and the distance increments $\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2$ can be calculated as:

$$\left|e_i\left(\mathbf{s}^{(i)}\right)\right|^2 = \left|\hat{y}_i - \sum_{j=i}^{2N_T} r_{ij} s_j\right|^2 .$$

The $K\sqrt{M}$ children are sorted in an increasing order of their PEDs and the K-best ones are chosen for the next layer. The algorithm finds the smallest PED at the last layer and returns the path of the tree with the smallest PED.

The choice of sorting algorithm has a significant impact on the computational complexity and throughput of the K-best detector, especially for the MIMO systems utilizing higher orders modulations. Two popular sorting algorithms, i.e. bubble sort [1], [7], [8] and distributed sorting [2]–[4], [9], have been used in the K-best detectors. Bubble sort is the most straightforward approach by repeatedly stepping through the list of data elements to be sorted, comparing each pair of adjacent items, and swapping them if they are in the incorrect order. Bubble sort is not particularly efficient as its average computational complexity increases quadratically with the number of data elements. Distributed sorting divides the elements to be sorted into a number of groups and then sorts each group separately. Distributed sorting provides a lower computational complexity and requires a shorter time to sort a list of data elements compared to bubble sort. Other sorting algorithms, such as dual-heap sort [10] and merge sort [5], have also been used in the K-best detection scheme. These previously utilized sorting algorithms in the published K-best detectors are one-dimensional, which may result in a relatively long latency and low throughputs when utilized in MIMO systems with high-order modulations. We propose a two-dimensional (2-D) parallel sorting algorithm for high-throughput K-best detectors. The proposed 2-D sorting also reduces the size of the storage elements significantly by decreasing the number of elements in the data set after each sorting stage.

The rest of this brief is organized as follows. Section II explains the proposed sorting algorithm. In Section III, the system architecture of the K-best detector and the 2-D sorter are presented. The implementation results and comparisons are provided in Section IV. Finally, Section V makes some concluding remarks.

II. Two-Dimensional Sorting Algorithm

The proposed algorithm rearranges the sorting elements into a matrix with K rows. Each row contains the expanded children from one parent node. Then the following five steps are performed.

Step 1) Row sort: Every row of the matrix is sorted in ascending order using the Schnorr-Euchner (SE) enumeration technique [11]. Consider a MIMO system utilizing 64-QAM modulation. Every row of the matrix contains eight elements. The SE enumeration applies a zig-zag movement around the child with the minimum PED value. The children to visit have increasing PED. Fig. 1 shows an example of the SE enumeration. The first child, $s_i^{[1]} = -1$, can be obtained by finding the minimum PED among eight PEDs. The second child can be obtained by comparing the left and right PEDs of the first child, and choosing the smaller one. For this example, the second child is $s_i^{[2]} = 1$ because $T_i(s_i = 1) < T_i(s_i = -3)$. The K rows of the matrix can be sorted in parallel.

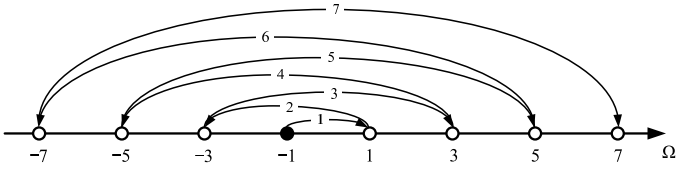

Fig. 1. Visiting order of the SE enumeration for eight expanded children utilizing 64-QAM constellation symbols.

Steps 2) Column sort and Step 3) Matrix partitioning: For a 64-QAM MIMO system, there are eight columns to be sorted in ascending order and each column has $K(i+1)$ elements at the i-th layer. After aligning the rows and columns of the matrix in ascending order, K elements are grouped in a sub-matrix such that the selected elements should be smaller than or equal to the adjacent elements that have not been selected. By enumerating all the possible cases, the matrix can be divided into three zones. Zone I contains elements that will always be included in the sub-matrix. These elements can be saved into the K-best pool. Some elements will never be among the K candidates. These elements are in Zone III and they will be discarded. The remaining elements are occasionally among the K candidates and no decisions can be made on whether they are among the K-best children. These elements are considered in Zone II and they will be stored in a new matrix for the next stage of the 2-D sorting.

In the proposed K-best detection algorithm, the fifth, sixth and seventh layers have the most complicated sorting procedures among the eight layers, because they have larger K values than those of the other layers. The only difference between the fifth and the sixth layer is that every column of the sixth layer has 20 elements while every column of the fifth layer has 10 elements. In other words, Zone III of the sixth layer has 80 more elements than that of the fifth layer. Therefore, we only present the 2-D sorting algorithm for the seventh and fifth layers. For the column sort, it should be noted that each column of the matrix uses a different sorting strategy. For example, in the seventh layer, as shown in Fig. 2, for the first stage, five out of eight elements in column 1 with the largest values are selected and will be stored in a new matrix for the second stage. The remaining three elements will be stored in the K-best pool. For columns 7 and 8, only the two smallest values should be stored in the new matrix for the second stage. The column sort procedure for the fifth layer is shown in Fig. 3. For column 1, two smallest values are

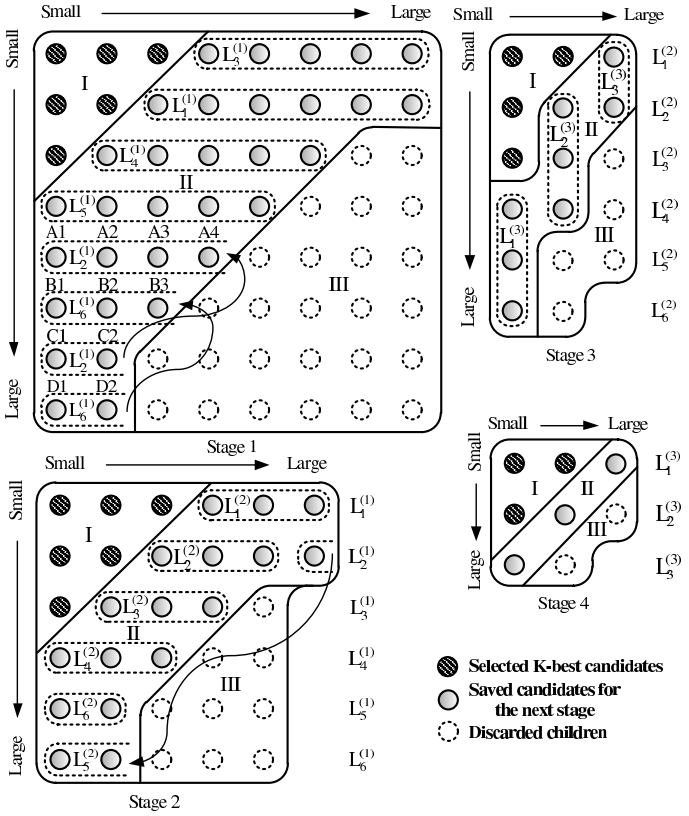

Fig. 2. Proposed 2-D sorting algorithm for the seventh layer.

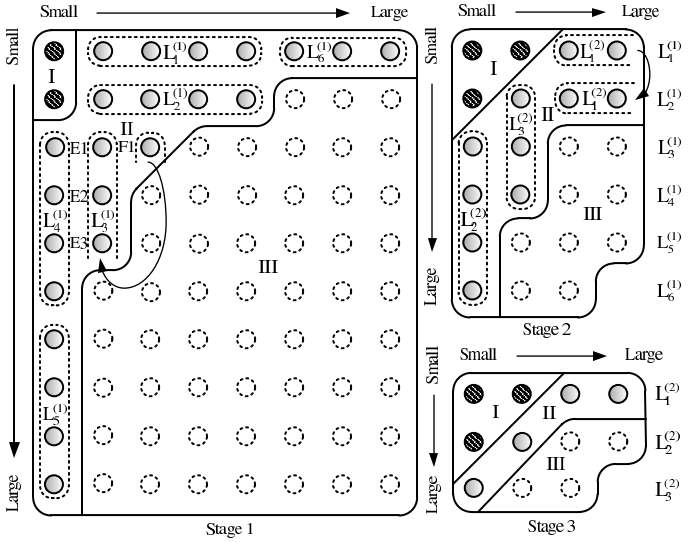

Fig. 3. Proposed 2-D sorting algorithm for the fifth layer.

selected and stored in the K-best pool, while the remaining eight elements are stored in the new matrix.

If K is large, the column sort will lead to a relatively long latency and will limit the throughput of the detector. Therefore, the 2-D sorting scheme is applied to the column sort to improve its efficiency. In the seventh layer, the eight elements in each column are rearranged into a 4×2 matrix. In the sixth and fifth layers, the twenty and ten elements in each column are rearranged into 5×4 and 5×2 matrices, respectively. Then the 2-D sorting process is performed similarly.

Step 4) Rebuild new matrices: After the row and column sort operations, the elements in Zone II will be stored in a new matrix for the next stage. The goal of rebuilding new matrices is to avoid the row sort operations. For example, in the seventh layer, as shown in Fig. 2, in the first stage there are 32 elements in Zone II. Note that $L_i^{(1)}$ should be smaller than or equal to $L_{i+1}^{(1)}$, where $L_i^{(1)}$, i = 1, ..., 6, denotes the i-th row of the new matrix. Therefore the new matrix has six elements in the first two rows and five elements in the remaining four rows, as shown in Fig. 2. The six elements of $L_2^{(1)}$ are merged with the four elements in row 5 ($A1 \sim A4$) and the two elements in row 7 ($C1$ and $C2$) of the matrix in the first stage, and should be sorted. After the row and column sort in the first stage, it is already known that $A1 \leq A2 \leq A3 \leq A4$, $C1 \leq C2$, $C1 \geq A1$, and $C2 \geq A2$. Therefore, we just need to insert $C1$ into $A2$, $A3$ and $A4$, and insert $C2$ into $A3$ and $A4$ in ascending order. Similarly, $L_6^{(1)}$ should be sorted, which can be performed by inserting $D1$ into $B2$ and $B3$ in ascending order, and comparing $D2$ with $B3$.

Rebuilding the new matrix for the fifth layer is slightly different. In the first stage, there are eight elements in the first column for the new matrix and the new matrix will have 23 elements. As shown in Fig. 3, we divide the first column into two parts as rows 4 and 5 of the new matrix for the second stage and each part has four elements. In this case, the ten elements in column 1 are not necessary to be fully sorted. The sorting strategy for column 1 can be modified by selecting two minimum data values as the K-best candidates, selecting four data values from the remaining eight children and sorting them, and finally sorting the other four children. The four elements of $L_3^{(1)}$ are combined with the three elements in column 2 ($E1 \sim E3$) and one element in column 3 ($F1$) of the matrix in the first stage. $L_3^{(1)}$ can be sorted by inserting $F1$ into $E2$ and $E3$ in ascending order. After the first stage, 26 and 55 elements, out of 64 and 80 elements of the matrices in Zone III, are eliminated for the seventh and fifth layers, which corresponds to 40% and 69% reduction in computational complexity, respectively. For the sixth layer (is not shown here), 135 out of 160 elements are eliminated from the matrix after the first stage, which implies 84% reduction in the computational complexity.

Step 5) Filling the K-best pool: Steps 1 to 4 will be repeated until the K-best pool is filled, as shown in the second, third and fourth stages in Fig. 2. There should be 20 children in the K-best pool for the seventh layer. After the third stage, 16 children have already been selected and four positions are remained in the K-best pool. At the fourth stage, the three children in Zone I are selected and stored in the K-best pool. The last child is chosen by comparing the three children in Zone II and selecting the one with the minimum value. In the fifth layer, there are 10-best candidates to be selected, as shown in Fig. 3. After the third stage, only two positions are left in the K-best pool, which can be obtained by sorting four children in Zone II and selecting the two minimum values.

III. Architecture Design

We utilized our proposed 2-D parallel sorting algorithm in a K-best detector for 4×4 MIMO systems utilizing 64-QAM modulation. Instead of choosing the same K value for all layers, our utilized K-best detector uses variable K values for different layers. Larger K values for the top layers can provide an improved performance while smaller K values for the bottom layers can reduce hardware complexity [6]. The system architecture of the K-best detector is shown in Fig. 4. The delay units are utilized to synchronize the inputs from $\mathbf{R}$ and $\hat{\mathbf{y}}$ with the K-best children obtained from the upper layer. The architecture is partitioned into eight blocks corresponding to the eight layers of the tree, each implemented in a pipelined and parallel fashion. Each layer uses a metric computation unit (MCU) to calculate the PEDs. In the eighth layer, all of the eight expanded children are selected as K-best candidates for the seventh layer, and hence, sorting is not required. In order to achieve high throughputs with a relatively low hardware complexity, only four clock cycles are used to process the K-best nodes in each layer (i.e. the new K-best children can be generated in four clock cycles). The K values of eight layers are [8, 20, 10, 10, 4, 4, 4, 1] and $\lceil K/4 \rceil$ = [2, 5, 3, 3, 1, 1, 1, 1] best candidates will be generated in every clock cycle, where $\lceil x \rceil$ denotes the smallest integer not less than x. In Fig. 4, the $K(i)\_8K(i+1)$ 2-D sorter in the i-th layer selects $K(i)$-best candidates from the $8K(i+1)$ expanded children of the $(i-1)$-th layer using the 2-D sorting algorithm described in Section II. For example, the 20_64 2-D sorter ($K(7)$ = 20 and $8K(8)$ = 64), selects 20-best candidates from 64 nodes in the seventh layer, as illustrated in Fig. 2. The delay units from the sixth layer to the second layer are used to store the indices of the K-best children of all upper layers and compute the PEDs of the lower layers. In the first layer, after obtaining the minimum PED, the path of the tree can be determined from the saved indices of the K-best nodes. Finally, the transmit signal $\hat{\mathbf{s}}$ can be estimated by searching in the constellation set Ω by the indices of the minimum PEDs for each layer.

The sorter blocks in Fig. 4 consist of SE sorters, buffers to store the elements in each column, column sorters, matrix rebuild units, and the K-best buffers. Fig. 5(a) shows the architecture of 20_64 2-D sorter for the seventh layer. The 16 expanded children are sorted by two SE sorters. Each sorter will sort eight children. After row sorting, eight columns of the matrix are stored using eight column buffers. The k-th column buffer is used to store $T_7(k)$ and $T_7(k+8)$. Eight elements in each column can be obtained from the column buffer after four clock cycles. The column sorters, such as the 6_8 column sorter, could have relatively long critical path delays. Reducing the critical path delays by inserting the pipeline registers in the eight column sorter requires 64 registers. Considering that the column sorters are designed as 2-D sorters, some row and column sorting operations could be done when data elements are stored in the column buffers. In this case, the workload of the column sorters are alleviated and the critical paths can be shorten without the need for pipeline registers. At the seventh layer of the proposed K-best detector, the eight elements of the column are rearranged into a 4×2 matrix and the first two steps of the 2-D sorting algorithm are the row sort and column sort. Hence, the row sort of two elements and the column sort of four elements are performed concurrently when writing eight data values into the column buffer.

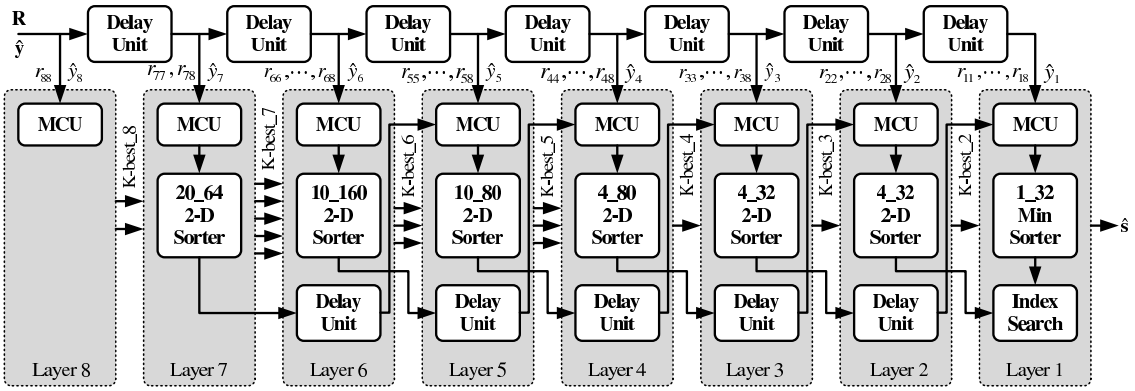

Fig. 4. Proposed architecture of the K-best detector for 4×4 MIMO systems utilizing 64-QAM modulation.

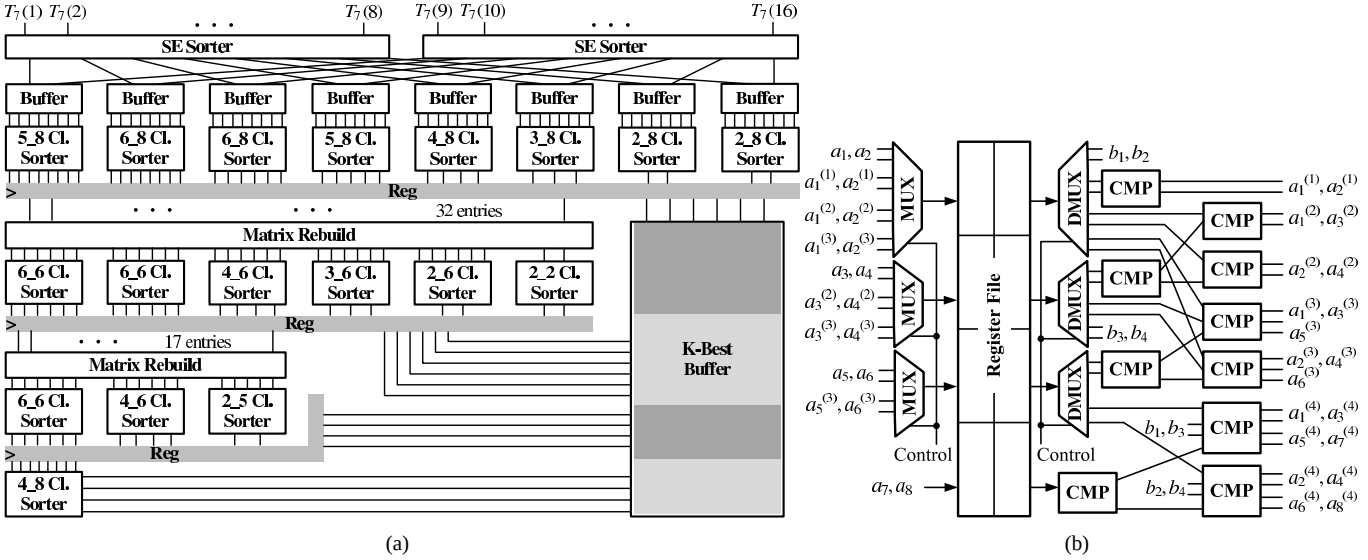

Fig. 5. (a) Architecture of the 20_64 2-D sorter in the seventh layer. (b) Architecture of the modified column buffer in layer 7.

The architecture of the modified column buffer in the seventh layer is shown in Fig. 5(b). The register file has the depth of four. In the first clock cycle, a_1 and a_2 are stored in the register file at address 0. a_3 and a_4 are written into the register file at address 1 in the second clock cycle. Meanwhile, two data values are read from address 0, they are sorted, and then $a_1^{(1)}$ and $a_2^{(1)}$ are written onto the same address in the register file. The operations in the third clock cycle include writing a_5 and a_6 into the register file at address 2, reading four data values from address 0 and 1, performing row sort and column sort on them, and then storing $a_1^{(2)}$ and $a_2^{(2)}$ back into the register file at address 0. The sorted data $a_3^{(2)}$ and $a_4^{(2)}$ are written onto address 1. In the fourth clock cycle, a_7 and a_8 are written into the register file at address 3. Six data values are read from address 0, 1 and 2, row sort and column sort operations are performed on them and they are stored in the register file. In the fifth clock cycle, eight elements are read from the register file and are sorted in ascending order for each row and each column. Utilizing the column buffers shortens the critical path of the column sorters by reusing the register file. It will not increase the hardware complexity, nor limit the throughput.

The useful children selected by the column sorters will be stored in the pipeline registers. Then the children in Zone II will be passed to the matrix rebuild unit to be rearranged in the next stage while those in Zone I will be saved in the K-best buffer. Three pipeline registers are utilized in the 20_64 2-D sorter and four stages are used to obtain the 20-best children. The number of children stored in the K-best buffer in four stages are 6, 6, 4 and 4, respectively. For the sixth layer, 20-best candidates will be generated in four clock cycles with five children in every clock cycle.

IV. Implementation Results and Comparisons

The variable K-best detector utilizing our proposed 2-D sorting algorithm is simulated in floating-point and fixed-point representations, implemented in Verilog-HDL, and synthesized using Synopsys Design Compiler. The word-lengths and the fractional length of the signals in the K-best detector are chosen based on the fixed-point simulation results of 4×4 64-QAM MIMO systems. When seven integer bits and six fractional bits are allocated to the signals after QR decomposition (r_{ij} and $\hat{y}_i$) and six integer bits and six fractional bits are allocated to PEDs (T_i), the bit error rate (BER) simulation results show a very close performance to that of the

TABLE I
IMPLEMENTATION RESULTS OF DIFFERENT MIMO DETECTORS UTILIZING 64-QAM

	[3]	[6]	[12]	[13]	[14]	[15]	[16]	This work
Antenna	4×4	4×4	2×2, 4×4	4×4	4×4	4×4	4×4	4×4
Detection	K-best	Var. K-best	Fixed sphere decoding	Radius-adaptive K-best	K-best	K-best	Search sphere detector	Var. K-best
SNR independent	Yes	Yes	Yes	No	Yes	Yes	Yes	No
K	10	8 †	8 †	8	10	64	N/A	16†
Technology	0.13-μm	0.18-μm	65-nm	65-nm	0.13-μm	40-nm	65-nm	90-nm
Max. freq. (MHz)	282	62.5	165	158	417	893	450	200
Processing cycles	10	1	2	13	10	N/A	N/A	4
Gate count (K)	114	366	88	210	340	355	184	182
Throughput	675 Mbps	1.5 Gbps	1.98 Gbps	285 Mbps	1 Gbps	200 Mbps	300 Mbps	1.2 Gbps
Latency (μs)	2.4	0.55	N/A	N/A	0.36	N/A	N/A	0.26

† Equivalent K values in the traditional K-best algorithm for the purpose of detection performance comparison.

floating-point model. Compared to bubble sorting, distributed sorting, and merge sorting, which require 3969, 1798, and 2881 comparison operations, respectively, our proposed 2D sorting algorithm uses 822 comparison operations. The K-best detector has the gate count of 182 K when synthesized in TSMC 90-nm technology. Operating at f_{clk} = 200 MHz, the K-best detector can achieve the throughput of $(\log_2 M \times N_T \times f_{clk})/N_c = 1.2$ Gbps, where M is the constellation size and N_c is the average number of clock cycles required for calculating the K-best nodes in one layer, which is four in our design. The implementation results of our detector are compared with several other designs in Table I. For a fair comparison, the computational complexity of the QR decomposition was not considered. Although the designs in [6] and [12] provide higher throughputs (for K = 8) than ours, our simulation results in Fig. 6 show that their BER performances are inferior to that of ours. Also, our detector has the shortest latency compared to other designs.

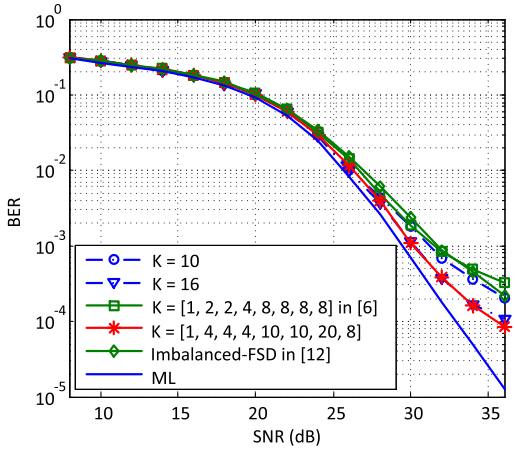

Fig. 6. The BER performance of different detection schemes for a 4×4 MIMO system utilizing 64-QAM modulation.

V. CONCLUSION

We proposed a novel 2-D sorting algorithm that can be utilized in K-best detectors. We employed the proposed sorting algorithm in a 4×4 MIMO system utilizing 64-QAM modulation. The variable K-best detector architecture was designed and implemented using TSMC 90-nm technology. It operates at 200 MHz and achieves the throughput of 1.2 Gbps with the latency of 0.26 μs. The simulation and implementation results indicate that the proposed sorting technique can be efficiently used in MIMO systems requiring high throughput, short latency, and a near-optimal detection performance, with a relatively low hardware complexity.

REFERENCES

[1] M. Wenk *et al.*, "K-best MIMO detection VLSI architectures achieveing up to 424 Mbps," in *Proc. IEEE Int. Symp. Circuits. Syst.*, 2006, pp. 1151–1154.

[2] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-best MIMO signal detector design and VLSI implementation," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 15, no. 3, pp. 328–337, 2007.

[3] M. Shabany and P. G. Gulak, "A 675 Mbps, 4×4 64-QAM K-best MIMO detector in 0.13 um CMOS," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 1, pp. 135–147, 2012.

[4] T.-H. Kim and I.-C. Park, "Small-area and low-energy K-best MIMO detector using relaxed tree expansion and early forwarding," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 10, pp. 2753–2761, 2010.

[5] N. Heidmann, T. Wiegand, and S. Paul, "Architecture and FPGA-implementation of a high throughput K+ best detector," in *Automation and Test in Europe Conference and Exhibition*, 2011, pp. 1–6.

[6] P.-Y. Tsai, W.-T. Chen, X.-C. Lin, and M.-Y. Huang, "A 4×4 64-QAM reduced-complexity K-best MIMO detector up to 1.5 Gbps," in *Proc. IEEE Int. Symp. Circuits. Syst.*, 2010, pp. 3953–3956.

[7] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoder for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, 2006.

[8] H.-L. Lin, R. C. Chang, and H.-L. Chen, "A high-speed SDM-MIMO decoder using efficient candidate searching for wireless communication," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 3, pp. 289–293, 2008.

[9] L. Liu *et al.*, "A 1.1-Gb/s 115-pJ/bit configurable MIMO detector using 0.13-um CMOS technology," *IEEE Trans. Circuits Syst. II*, vol. 57, no. 9, pp. 701–705, 2010.

[10] C. H. Liao, T. P. Wang, and T. D. Chiueh, "A 74.8 mW soft-output detector IC for 8×8 spatial-multiplexing MIMO communications," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 411–421, 2010.

[11] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Math. Program*, vol. 66, pp. 181–191, 1994.

[12] L. Liu, J. Lofgren, and P. Nilsson, "Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes," *IEEE Trans. Circuits Syst. I*, vol. 59, no. 9, pp. 2085–2096, 2012.

[13] C.-A. Shen and A. Eltawil, "A radius adaptive K-best decoder with early termination: algorithm and VLSI architecture," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 9, pp. 2476–2486, 2010.

[14] M. Mahdavi and M. Shabany, “Novel MIMO detection algorithm for high-order constellations in the complex domain,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 5, pp. 834–847, 2013.
[15] C.-A. Shen, C.-P. Yu, and C.-H. Huang, “Algorithm and architecture of configurable joint detection and decoding for MIMO wireless communications with convolutional codes,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 2, pp. 587–599, 2016.
[16] E. P. Edeva and G. P. Fettweis, “Efficient architecture for soft-input soft-output sphere detection with perfect node enumeration,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 9, pp. 2932–2945, 2016.