Improving MIMO Sphere Detection Through Antenna Detection Order Scheduling

Michael Wu (Rice University, Houston, TX, USA; mbw2@rice.edu),
Chris Dick (Xilinx, San Jose, CA, USA; chris.dick@xilinx.com),
Yang Sun (Rice University, Houston, TX, USA; ysun@rice.edu),
Joseph R. Cavallaro (Rice University, Houston, TX, USA; cavallar@rice.edu)

Abstract—**This paper proposes a novel scalable Multiple-Input Multiple-Output (MIMO) detector that does not require preprocessing to achieve good bit error rate (BER) performance. MIMO processing is a key technology in broadband wireless technologies such as 3G LTE, WiMAX, and 802.11n. Existing detectors such as Flexsphere use preprocessing before MIMO detection to improve performance. Instead of costly preprocessing, the proposed detector schedules multiple search passes, where each search pass detects the transmit stream with a different permuted detection order. By changing the number of parallel search passes, we show that this scalable detector can achieve comparable performance to Flexsphere with reduced resource requirement, or can eliminate LLR clipping and achieve BER performance within 0.25 dB of exhaustive search with increased resource requirement.**

I. Introduction

Multiple-Input Multiple-Output (MIMO) is a key technique of many wireless standards such as 3G LTE, WiMAX and 802.11n. As the received signals are combined signals from the multiple transmit antennas, the main challenge for a MIMO receiver is to decouple the received signals to recover the transmit signals. This challenge, the MIMO detection problem, is known to be an integer least-squares problem that can be solved with an exhaustive search. However, an exhaustive search algorithm is cost prohibitive for wireless systems with strict latency and power requirements.

As a result, a number of suboptimal algorithms with significantly reduced complexity have been developed. Since the search process is a tree traversal, there are two main approaches to MIMO detection: depth-first algorithms such as depth-first sphere detection [1], and breadth-first algorithms such as K-best [2]. In depth-first sphere detection, the number of tree nodes visited is large in the low signal to noise ratio (SNR) range, while the number of tree nodes visited is small in the high SNR range. As a result, depth-first sphere detection has variable throughput which is undesirable in systems with strict latency requirements. An attractive alternative is K-best detection that visits a fixed number of nodes which results in fixed throughput independent of SNR. However, a large K value is required to achieve performance close to exhaustive search. The primary challenge of a high throughput low latency K-best detector design is the need to sort at each step of the algorithm.

To address the sorting complexity of the K-best detection algorithm, a number of modified sort-free algorithms similar to K-best have been developed. The key to good sort-free algorithms is finding an approximation of sort that provides good overall performance. For example, in SSFE [3], instead of sorting N values to find the best K values, the workload is first partitioned into M arrays, where M is the modulation order. Fast enumeration is used to find the best K/M values for each subarray without sorting each subarray. However, this approximation reduces the accuracy of the detector. To recover some of the performance loss, authors in [4] present a hard decision detector called Flexsphere. While the search process is similar to SSFE, Flexsphere improves upon SSFE by performing better preprocessing, V-BLAST-like antenna reordering and modified real-value decomposition (RVD), to achieve accuracy closer to optimal exhaustive search.

An existing implementation of the Flexsphere detector [5] for WiMAX shows that the V-BLAST-like preprocessing block uses significant amount of FPGA resources due to matrix inversions, while the QR decomposition and detection blocks combined use significantly less FPGA resources. In addition, the candidate list generated by Flexsphere does not guarantee bit-level reliability information, the log likelihood ratio (LLR), can not be found for all bits, leading to the need for LLR clipping [6], [7].

Instead of computing the optimal antenna detection order with a V-BLAST-like preprocessing block before detection, we propose scheduling multiple search passes through the search space, where each search uses a different permuted antenna detection order. Compared to the design in [5], the proposed design can either achieve comparable performance with reduced resource requirement, or it can achieve better performance with increased resource requirement. In addition, we will show this design can avoid the problem of LLR clipping when the number of search passes through the search space is the same as the number of antennas. Since a typical wireless system usually combines a soft-output detector and a soft-input decoder to maximize performance, we also modified the Flexsphere detector to perform soft-output decoding in this paper.

This paper is organized as follows: Section 2 gives an overview of the system model. Section 3 describes the proposed detection algorithm. Section 4 presents the performance of the proposed detector. In section 5, we present the corresponding FPGA implementation. Finally, we conclude in section 6.

II. MIMO System Model

For an $N_r \times N_t$ MIMO system, the source transmits N_t signals and the destination receives signals on N_r antennas. The received signal, $\mathbf{y} = [y_0, y_1, ..., y_{N_r-1}]$, can be modeled by:

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

where $\mathbf{H} = [\mathbf{h_0}, \mathbf{h_1}, ..., \mathbf{h_{N_t-1}}]$ is the $N_r \times N_t$ channel matrix, where each element, h_{ij}, is an i.d.d. zero mean circular symmetric complex Gaussian random variable (ZMCSCG) with σ_h^2 variance. Given a binary vector $\mathbf{x} = [x_0, x_1, x_2 ... x_{L-1}]$, where $L = \log_2 M \cdot N_t$, the function $\mathrm{map}(\cdot)$ translates the binary vector $\mathbf{x}$ onto $\mathbf{s} = [s_0, s_1, ..., s_{N_t-1}]$. Each element of $\mathbf{s}$, s_i, is an element drawn from a finite alphabet $\mathbf{\Omega}$ with cardinality M and average power E_s per symbol. For example, the constellation alphabet for QPSK is $\{-1-j, -1+j, 1-j, 1+j\}$ with $M = 4$. Finally, the receiver noise, $\mathbf{n} = [n_0, n_1, ..., n_{N_r-1}]$, is an independent ZMCSCG with $\sigma_n^2/2$ variance per dimension.

We can obtain an equivalent system model in the real domain. To ensure the in-phase and the quadrature parts of the same complex symbol are adjacent neighbors [4], we perform modified real-valued decomposition (MRVD) as follows:

$$\begin{pmatrix} \Re(y_0) \\ \Im(y_0) \\ \Re(y_1) \\ \Im(y_1) \\ \vdots \\ \Re(y_{N_r-1}) \\ \Im(y_{N_r-1}) \end{pmatrix} = \begin{pmatrix} \Re(\mathbf{H}) & -\Im(\mathbf{H}) \\ \Im(\mathbf{H}) & \Re(\mathbf{H}) \end{pmatrix} \begin{pmatrix} \Re(s_0) \\ \Im(s_0) \\ \Re(s_1) \\ \Im(s_1) \\ \vdots \\ \Re(s_{N_r-1}) \\ \Im(s_{N_r-1}) \end{pmatrix} + \tilde{\mathbf{n}}, \tag{2}$$

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\tilde{\mathbf{s}} + \tilde{\mathbf{n}}. \tag{3}$$

MRVD doubles the number of elements in each vector and doubles both dimensions of $\tilde{\mathbf{H}}$. Furthermore, each element of the equivalent transmit vector, $\tilde{s}_i$, is an element drawn from a finite alphabet $\mathbf{\Omega}'$ with cardinality $Q = \sqrt{M}$. For example, the real value decomposed constellation alphabet for QPSK is $\{-1, 1\}$ and $Q = 2$.

Given $\tilde{\mathbf{y}}$ and the channel matrix $\tilde{\mathbf{H}}$, the goal of the soft-output MIMO detector at a MIMO receiver is to compute the logarithmic a-posteriori probability (APP) ratio, $L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}})$, per bit. Assuming no prior knowledge of the transmitted bits, the soft-output value per bit can be approximated with the following equation using max-Log approximation [8] and L^1-norm [1].

$$L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}}) \approx \min_{\mathbf{x} \in \mathbb{X}_{k,-1}} \frac{\left\| \tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{s}} \right\|_1}{2\sigma_n^2} - \min_{\mathbf{x} \in \mathbb{X}_{k,+1}} \frac{\left\| \tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{s}} \right\|_1}{2\sigma_n^2}, \tag{4}$$

where $\mathbb{X}_{k,-1}$ is the list of all binary vectors with the k^{th} component equal to -1, $\mathbb{X}_{k,+1}$ is the list of all binary vectors with the k^{th} component equal to $+1$, and $\tilde{\mathbf{s}} = \mathrm{map}(\mathbf{x})$.

III. Proposed Soft-output N-Way MIMO Detector

The soft-output detector is computationally intensive as computing $L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}})$ exactly requires searching through the set of all possible binary vectors to find the hypothesis and the counter-hypothesis. To approximate $L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}})$, a soft-output MIMO detector finds a smaller set of transmit vectors, or a candidate list, $\mathcal{L}$, by excluding unlikely vectors. To compute $L_D(x_k|\mathbf{y}, \mathbf{H})$, the candidate list is divided into $\mathcal{L}_{k,-1}$ and $\mathcal{L}_{k,+1}$, where $\mathcal{L}_{k,-1}$ is the list of candidates with the k^{th} bit equal to -1 and $\mathcal{L}_{k,+1}$ is the list of candidates with the k^{th} bit bit equal to $+1$,

$$L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}}) \approx \underbrace{\min_{\mathbf{x} \in \mathcal{L}_{k,-1}} \frac{\left\| \tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{s}} \right\|_1}{2\sigma_n^2}}_{hypothesis} - \underbrace{\min_{\mathbf{x} \in \mathcal{L}_{k,+1}} \frac{\left\| \tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{s}} \right\|_1}{2\sigma_n^2}}_{counter-hypothesis}. \tag{5}$$

Although Flexsphere [4] performs close to exhaustive search as a hard decision detector, it does not perform close to exhaustive search since the generated candidate list size is very small. When computing soft-output for the k^{th} bit, the list $\mathcal{L}_{k,-1}$ or the list $\mathcal{L}_{k,+1}$ may be an empty set, in which case $L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}})$ cannot be computed since the hypothesis or the counter-hypothesis is unknown. When an empty set occurs, we can still generate an LLR value by clipping the LLR to a predetermined value [7]. This, however, leads to performance degradation, especially when the clipping value is not picked appropriately. Furthermore, $L_D(x_k|\tilde{\mathbf{y}}, \tilde{\mathbf{H}})$ may not be accurately approximated due to the small candidate list size. Although the V-BLAST-like preprocessing ensures optimal detection order, the preprocessing block is very expensive. In the proposed detection algorithm, we leverage the fact that the search algorithm is cheap and propose a novel detection method in which the preprocessing block is removed in favor of performing multiple search passes.

A. MIMO Detection Search Pass

The proposed MIMO detection scheme uses the same search algorithm as Flexsphere and SSFE. Since the search algorithm is discussed in-depth in [4], we will give a brief overview of the algorithm in this section.

Given $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{H}}$, we first perform QR decomposition on $\tilde{\mathbf{H}}$ for an equivalent system model, where the Euclidean distance of a transmit vector $\tilde{\mathbf{s}}$, is:

$$\left\| \tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{s}} \right\|_1 = \left\| \hat{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}} \right\|_1. \tag{6}$$

where $\mathbf{R}$, an upper-triangular matrix, is the effective channel matrix and $\hat{\mathbf{y}}$ is the effective received vector.

To generate a smaller candidate list, the MIMO detector excludes transmit vectors with large Euclidean distances and searches for transmit vectors with small Euclidean distances. Since $\mathbf{R}$ is upper triangular, a detector can evaluate the Euclidean distance of a transmit vector level by level. As a result, the search process can be viewed as a traversal through a tree. The search algorithm can be viewed as a greedy tree search. The algorithm traverses the tree breadth first and prunes unlikely branches level by level until there are a few complete paths left. Figure 1 demonstrates an example of the search process for a 2×2 16-QAM MIMO system. Since all branches in the first two levels are kept, the first two levels of the tree are fully expanded. At the subsequent

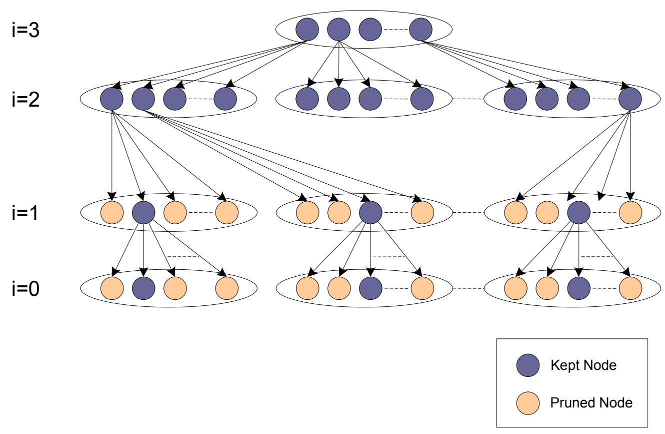

Figure 1. An example of the search process for a 2x2 16-QAM MIMO system

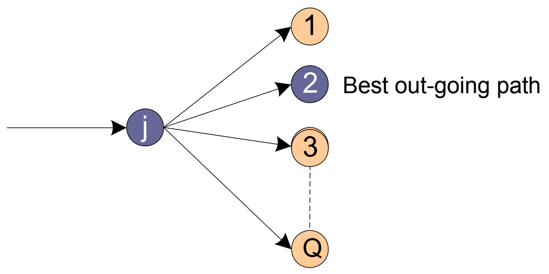

Figure 2. An example of the pruning process at node j.

levels, the algorithm evaluates all branches out of each node and prunes unlikely branches with large Euclidean distances. The selectivity of the pruning process is controlled by how many branches a node keeps in the pruning process. This search algorithm only keeps the best branch for each node after the first two tree levels. All surviving paths at the last tree level are in our candidate list.

We will now describe the pruning function. Figure 2 shows the data flow at the j^{th} node at stage i. Given one incoming path with path history $\mathbf{p} = (p_0, p_1, ..., p_i)$ and euclidean distance d_m, we wish to extend the incoming path to the next level $i+1$ by picking the best out-going path among Q paths. The updated cumulative weight after connecting node p_i to the k^{th} node in level $i+1$ is:

$$d'_k = d_m + w_{j,k}^{<i+1>}, \;\; 0 \le k \le Q-1, \tag{7}$$

where $w_{i,k}^{<i+1>}$ is defined as:

$$w_{j,k}^{<i+1>} = ||\hat{y}_k - R_{k,k}s_k - \sum_{j=k+1}^{M} R_{k,j}p_j||_1, \tag{8}$$

$$= ||b_{i+1}(\mathbf{p}) - R_{k,k}s_k||_1. \tag{9}$$

The best connected node in level $t+1$ that minimizes $w_{i,k}^{<i+1>}$is simply the closest constellation point in Ω' to $R_{k,k}s_k / b_{i+1}(\mathbf{p})$. The algorithm to find the best node can be implemented with a simple round function followed by a threshold function. When the best node is found, the path history at level $i+1$ is updated by appending the best outgoing node to $\mathbf{p}$ and the Euclidean distance at level $i+1$ is updated by saving d'_m as d_m.

Algorithm 1 Proposed MIMO Detection Algorithm

```
1) Initialization: $\mathcal{L} \leftarrow \emptyset$
2) for i = 0 to N-1 do
3)       $\mathbf{y_i} \leftarrow$ circular rotate rows of $\mathbf{y}$ i times
4)       $\mathbf{H}_i \leftarrow$ circular rotate columns of $\mathbf{H}$ i times
5)       $(\hat{\mathbf{y}}_i, \mathbf{R}_i) \leftarrow$ MRVD-QRD$(\mathbf{y}_i, \mathbf{H}_i)$
6)       $\mathcal{L}_i \leftarrow search(\hat{\mathbf{y}}_i, \mathbf{R}_i)$
7)       $\mathcal{L} \leftarrow \mathcal{L} \bigcup \mathcal{L}_i$
8) end
9) Compute LLR values using candidate list $\mathcal{L}$
```

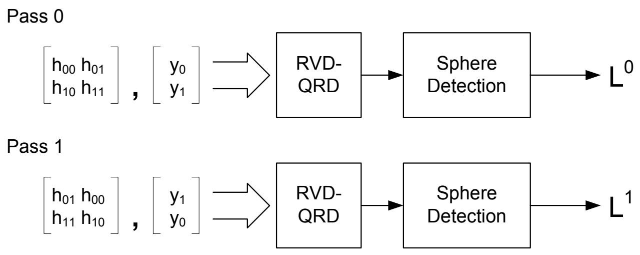

Figure 3. Proposed Detector for a 2x2 MIMO System.

B. N-Way Scheduled Search

For the proposed N-way scheduled MIMO detector, we remove the V-BLAST-like preprocessing block in Flexsphere in favor of multiple search passes. The proposed algorithm is summarized in Algorithm 1. The inputs for all search passes are the same, consisting of the channel matrix $\mathbf{H}$ and the received vector $\mathbf{y}$. Since the detection order affects the performance of the detector, we schedule multiple search passes where each pass uses a different antenna detection order to generate different lists of candidates. Without the V-BLAST-like preprocessing block, the optimal detection order is not known. We propose an antenna detection order that can be obtained by a simple circular rotation of columns of $\mathbf{H}$ and rows of $\mathbf{y}$. A search pass, consisting of RVD decomposition, QR decomposition and tree search, is performed for each permutation. The result is the i^{th} search pass processes the i^{th} antenna first. Since $\mathbf{H}$ has N_t columns, the proposed detector runs up to N_t passes for N_t possible permutations. Each detection pass generates more candidates for the candidate list. The total number of candidates generated is NC, where C is the number of candidate generated by one search pass and N is the number of search passes. Figure 3 illustrates two search passes for a 2x2 MIMO 16-QAM system.

After each search pass, the hypothesis and the counter-hypothesis for each bit are updated using the generated candidate lists. After N search passes, the LLR values are computed for each bit by finding the difference between the hypothesis and the counter-hypothesis according to equation 5. Compared to Flexsphere, a larger list increases the probability of finding the best hypothesis and the counter-hypothesis per transmitted bit. We note that each search pass guarantees that the hypothesis and the counter-hypothesis can be found for the bits corresponding to the first antenna detected. The first two levels of the tree are completely expanded. Due to the property of the pruning function, there is a complete path through all nodes in the first two levels of the tree at the end of the search. This means the candidate list will have a candidate through each possible symbol transmitted by the first antenna. This property is illustrated by Figure 1. As a result, the two sublists for each bit transmitted by the first antenna level are always non-empty and the hypothesis and the counter-hypothesis both can be found. The LLR values corresponding to the antenna expanded first by each of the search passes do not require clipping. When $N = N_t$, there is no need for LLR clipping since each antenna is fully expanded once and each bit has a hypothesis and a counter-hypothesis, increasing performance.

IV. Performance

We compared the BER performance of our proposed N-way scheduled MIMO detector against the soft-output Flexsphere detector and the K-best detector in a flat fading Rayleigh fading channel. In our simulation, the soft output of the detector is fed to a length 2304, rate 1/2 WiMAX LDPC decoder, which performs up to 20 decoding iterations. For the K-best detector, we choose a large K value of 64 for 16-QAM and 256 for 64-QAM. For the proposed algorithm, we show the results for up to 4 detection passes. We used an LLR clipping value of 8 for all the detector configurations with the exception of the proposed detection algorithm with 4 detection passes where LLR clipping was not required.

Figure 4 compares the performance of detectors for 16-QAM. We see that Flexsphere gains more than 1 dB of performance at FER of 10^{-3} compared to the N-way scheduled MIMO detector where N = 1. This shows the benefits of V-BLAST-like preprocessing. The proposed N-way scheduled MIMO detector with N = 2 performs better than Flexsphere, while the proposed N-way scheduled MIMO detector with N = 3 performs better than the K-best MIMO detector. Finally, the proposed detector with N = 4 performs within 0.25 dB of exhaustive search at FER of 10^{-3}. This is expected as N = 4 avoids LLR clipping and generates more accurate LLR values due to a larger candidate list. Figure 5 compares the performance of the detector for 64-QAM. The result is similar to 16-QAM. For N = 2, the N-way scheduled MIMO detector performs similarly to the soft-output Flexsphere. For N = 3, the N-way scheduled MIMO detector performs similarly to the K-best detector. For N = 4, the N-way scheduled MIMO detector's performance is close to exhaustive search. The results suggest that we can remove the costly V-BLAST-like preprocessing and improve performance by increasing the number of detection passes.

V. FPGA Hardware Implementation

In this paper, we targeted a Virtex®-5 XC5VFX130T-2FF1738 FPGA. A complete soft-output Flexsphere detector is implemented with Xilinx System Generator by extending the hard-decision Flexsphere design presented by authors in [5]. The hard-decision Flexsphere detector design consists of three components: a channel preprocessor, RVD/QRD block, and a sphere detector block. The design outputs hard decisions which are the constellation points of the candidate with the smallest Euclidean distance among the 64 candidates at the last level. To perform soft-output decoding, instead of outputting the candidate with the smallest Euclidean distance, we added an LLR computation block which uses the 64 candidates to generate soft-output LLR values.

To meet the target data rate of 83.965 Mbps which corresponds to a 360 sub-carriers WiMAX system, the sphere detector has a throughput of one candidate per cycle. To match the data rate, the LLR generator needs to process 64 candidates to generate 24 bit-level soft values every 64 cycles with a minimal clock frequency of 225 Mhz. The LLR generator is straight forward. We demodulate each incoming candidate into bits using a look-up table. We allocate two registers per demodulated bit. In total, we have 48 18-bit registers which are partitioned into two sets–one set for the hypothesis and one set for the counter-hypothesis. Depending on whether a

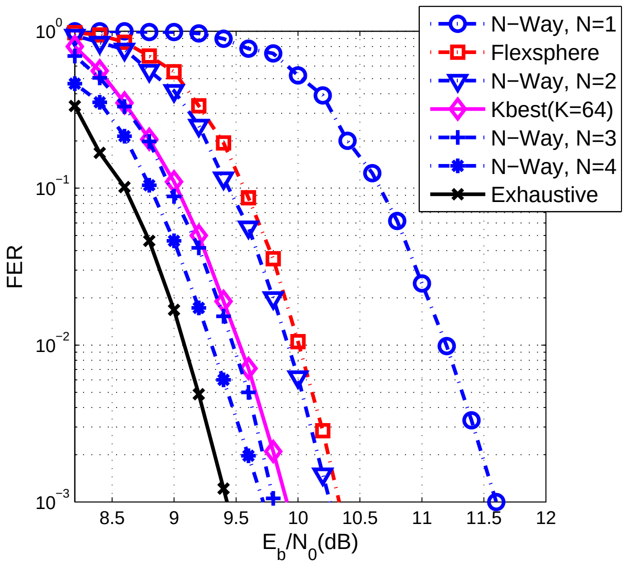

Figure 4. 4x4 16-QAM BER Performance

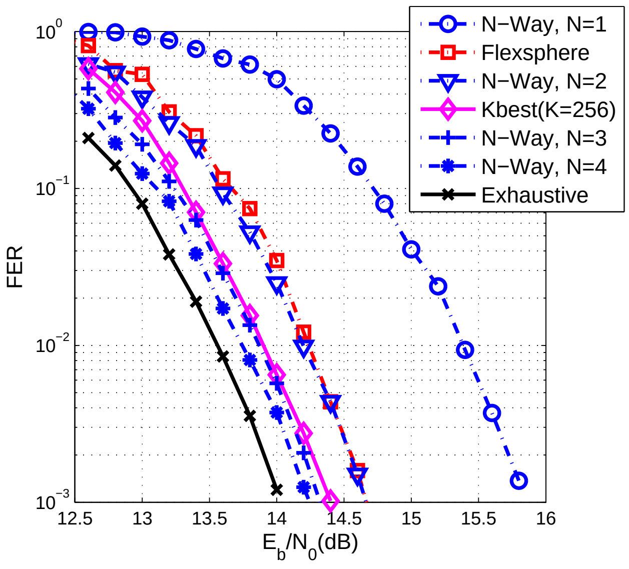

Figure 5. 4x4 64-QAM BER Performance

Table I
RESOURCE USAGE OF SOFT-OUTPUT FLEXSPHERE

Block	Slices	LUTs/FFs	DSP48	Block RAM
Preprocessing	9,999	20,339/29,954	159	105
RVD/QRD	1,715	4,418/5,556	30	27
Sphere Detector	2,445	3,113/6,525	48	12
LLR Computation	1,498	1,906/3,909	0	2
Total	15,657	29,776 /45,944	237	146

Table II
RESOURCE USAGE OF MERGE BLOCKS

N	Slices	LUTs/FFs	DSP48	Block RAM
2	65	100/201	0	0
3	137	208/401	0	0
4	202	312/602	0	0

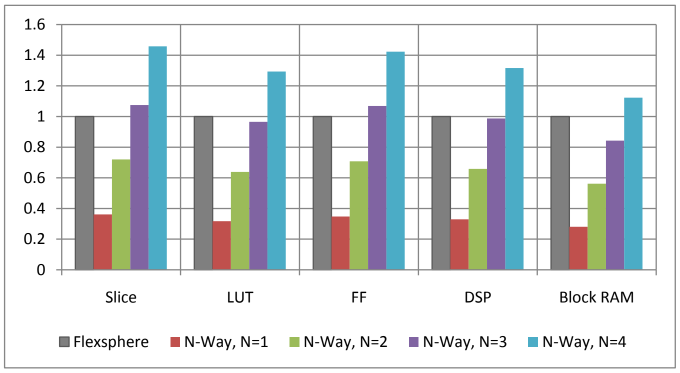

Figure 6. Resource Comparison

demodulated bit is 0 or 1, we compare the Euclidean distance against the value in the corresponding register. If the current Euclidean distance is smaller, the value in the register is replaced by the current Euclidean distance of the candidate. All 24 bits are evaluated in parallel to meet the throughput requirement. The resources required for the blocks from [5] and the LLR computation block are shown in Table I.

Since the proposed design is a modified Flexsphere detector, we used the design proposed by authors in [5] to implement our current proposed design. The RVD-QRD, the sphere detector, and the LLR Computation blocks are reused while the channel preprocessor is removed. Each search pass is one pass through all three blocks. Therefore, the throughput of an implementation with one copy of these three blocks is reduced N times for N search passes.

To improve the achievable throughput of the proposed detector, we use more resources and perform search passes in parallel. We simply replicate the serial implementation N times, where each instance of the detector performs a detection pass independently to generate the hypothesis and the counter-hypothesis per transmitted bit. The output is N lists of the hypotheses and counter-hypotheses per transmitted bit that need to be merged into one list. The merger block is a fairly simple block, where the minimum value among N values is found for each transmitted bit. Since the resource cost depends on N, the resources required by merging N lists are listed in Table II.

The overall resource required is approximately N times the serial implementation of the proposed detector plus the cost of merger block for value N and is shown in Table III.

Figure 6 shows the cost of different realizations of the detector compared to the soft-output Flexsphere, where the cost of the Flexsphere is normalized to one. To achieve comparable performance to soft-output Flexsphere with preprocessor, we need at least two search passes. Although we need an additional RVD/QRD block, sphere detector block and LLR generator block for N = 2, we save a substantial amount of resources compared to the Flexsphere detector because we eliminate the expensive preprocessing block. The amount of resources required for N = 4 increased around 45 percent compared to the soft-output Flexsphere detector. However, increasing the number of search passes to four increases the performance of the detector by another 0.5 dB over the soft-output Flexsphere design.

Table III
TOTAL RESOURCE USAGE OF N-WAY SPHERE DETECTOR

N	Slices	LUTs/FFs	DSP48	Block RAM
1	5,658	9,437/15,990	78	41
2	11,274	19,018/32,525	156	82
3	16,827	28,743/49,117	234	123
4	22,832	38,515/65,381	312	164

VI. CONCLUSION

In this paper, we presented a scalable detection algorithm that does not require preprocessing to achieve good performance. We propose scheduling search passes with different antenna detection order, where the i^{th} antenna is positioned as the first layer for the i^{th} search pass. We show that by changing the number of search passes, we can achieve BER performance 0.25 dB from exhaustive search and eliminate LLR clipping. We also show that we can achieve comparable performance to Flexsphere with two search passes and eliminate the costly processing, resulting in a more area-efficient design.

REFERENCES

[1] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm," *IEEE J. Solid-State Circuit*, vol. 40, pp. 1566–1577, July 2005.

[2] K. Wong, C. Tsui, R. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *IEEE Int. Symp. on Circuits and Syst.*, vol. 3, May 2002, pp. 273–276.

[3] M. Li, B. Bougard, E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, and F. Catthoor, "Selective Spanning with Fast Enumeration: A Near Maximum-Likelihood MIMO Detector Designed for Parallel Programmable Baseband Architectures," in *ICC '08. IEEE International Conference on Communications*, May 2008.

[4] K. Amiri, C. Dick, R. Rao, and J. R. Cavallaro, "A High Throughput Configurable SDR Detector for Multi-user MIMO Wireless Systems," *Springer Journal of Signal Processing Systems*, vol. 62, pp. 233–245, February 2011.

[5] C. Dick, M. Trajkovic, S. Denic, D. Vuletic, R. Rao, F. Harris, and K. Amiri, "FPGA Implementation of a Near-ML Sphere Detector For 802.16E Broadband Wireless Systems," in *SDR '09: Proceedings of the 2009 SDR Technical Conference and Product Exposition.*

[6] Y. de Jong and T. Willink, "Iterative Tree Search Detection for MIMO Wireless Systems," *IEEE Transactions on Communications*, June 2005.

[7] E. Zimmermann, "Complexity Aspects in Near-capacity MIMO Detection-Decoding," Ph.D. dissertation, TU Dresden, 2007.

[8] B. Hochwald and S. Brink, "Achieving Near-Capacity on a Multiple-Antenna Channel," *IEEE Tran. Commun.*, vol. 51, pp. 389–399, Mar. 2003.