Breadth-First Tree Search MIMO Signal Detector Design and VLSI Implementation

Sizhong Chen*, Tong Zhang* and Yan Xin†
*Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute, USA
†Department of Electrical and Computer Engineering
National University of Singapore, Singapore

ABSTRACT

Efficient VLSI implementation of multiple-input multiple-output (MIMO) signal detectors plays an important role in the real-life MIMO communication systems. This paper presents a nonlinear MIMO detector design solution, called relaxed K-best detector, that can support efficient VLSI implementation, while maintaining good detection performance. To the best of our knowledge, this is the first nonlinear detector ever reported in the open literature, capable of supporting 4×4 MIMO transmission with 64 quadrature amplitude modulation (QAM). A soft-output relaxed K-best detector has been designed and synthesized using Synopsys with 0.18 μm CMOS technology. With the silicon area of 20 mm^2, the detector can achieve up to 50 Mbps throughput for 4×4 MIMO with 64-QAM modulation. The detector can achieve almost the same performance as the detectors using the sphere decoding algorithm.

I. INTRODUCTION

As a topic of great current interest, multiple-input multiple-output (MIMO) wireless data transmission technology can realize dramatic improvements in terms of spectral efficiency and/or link reliability compared to what is achievable today. Hence, MIMO holds great promise for a wide use in future wireless communication systems such as the fourth-generation (4G) mobile radio systems, fixed broadband wireless systems, and wireless local area networks [1], [2]. As a key element in wireless MIMO communication devices, MIMO signal detectors can be either hard-output (i.e., only provides the hard estimation of the transmitted bits) or soft-output (i.e., provides a posteriori probability (APP) information about each bit).

The maximum-likelihood (ML) hard-output and maximum *a posteriori* (MAP) soft-output MIMO detectors based on exhaustive search typically incur prohibitive computational complexities, and therefore development of MIMO detectors with reduced computational complexity is of great practical importance. One family of reduced-complexity detectors is linear detectors based on the principles of linear minimum mean-square error (LMMSE) or zero-forcing (ZF). Although they can greatly reduce the computational complexity, they suffer from significant performance degradation. To achieve the performance closer or equivalent to the ML or MAP detection, researchers have developed several nonlinear detectors that realize hard- or soft- output detection through *non-exhaustive tree search* based on a set of additive metrics, where the goal of hard-output detection is to find one tree leaf with the best metric and the goal of soft-output detection is to find a list of tree leaves to calculate the APP information of each bit (as explained later). Because of their computation-intensive nature, those nonlinear MIMO detectors should be implemented in the form of application specific hardware in order to meet the throughput and power consumption constraints in real-life wireless communication systems.

Depending on how to carry out the non-exhaustive tree search, nonlinear detectors fall into three categories, i.e., depth-first search, metric-first search, and breadth-first search. The depth-first detection using sphere decoding algorithm [3] so far attracted the most attentions and holds the state of the art: For 4×4 MIMO transmission with 16 quadrature amplitude modulation (QAM), hard-output depth-first detectors [4] achieve much higher throughput (under high signal to noise ratio (SNR)) than their breadth-first counterparts [5], [6], while the soft-output depth-first detector [7] is the only one nonlinear soft-output detector ever reported.

However, the design and implementation of hard- or soft- output nonlinear detectors that can support 64-QAM modulation have not been addressed in the open literature. As the first attempt to fill this gap, this paper presents a breath-first hard- and soft- output detector design solution that can support 4×4 MIMO transmission with 64-QAM modulation. This design solution is essentially based on the *M*-algorithm [8], a well-known breath-first tree search algorithm. Following the convention in the existing literature on MIMO signal detection, we refer the breadth-first detector based on the principle of *M*-algorithm as *K*-best detector [5], [6]. As discussed later, the direct realization of a *K*-best detector in hardware requires implementing a sorting operation. Due to the serial nature of sorting and potentially large amount of data movement, this will lead to a throughput bottleneck and significant power consumption overhead. To tackle this challenge, instead of directly following the principle of *M*-algorithm as in the *K*-best detectors [5], [6], we modified the basic principal of the *M*-algorithm, leading to a so-called *relaxed* *K*-best detector. The basic idea is to replace the strict sorting operation with a *distributed* and *approximate* sorting that can significantly simplify the hardware implementation. The detailed detector design and hardware structure are described.

To demonstrate the proposed design solution, we designed soft-output relaxed *K*-best detectors for 4×4 MIMO transmission with 64-QAM modulation. The design entry is Verilog HDL language, which is synthesized using Synopsys tool set with 0.18 μm CMOS technology. The detector silicon area is estimated as 20 mm^2 and the throughput can be up to 50 Mbps at relatively high SNR. Concatenated with an low-density parity-code (LDPC), this soft-output detector can achieve almost the same performance as using the sphere decoding algorithm. To the best of our knowledge, this is the first soft-output detector that can support 4×4 MIMO transmission with 64-QAM modulation ever reported in the open literature.

II. Background

A. System Model

This work considers a MIMO system with *spatial multiplexing* signaling (i.e., the signals transmitted from individual antennas are independent of each other), as illustrated in Fig. 1. Let N_t and N_r represent the number of transmit and

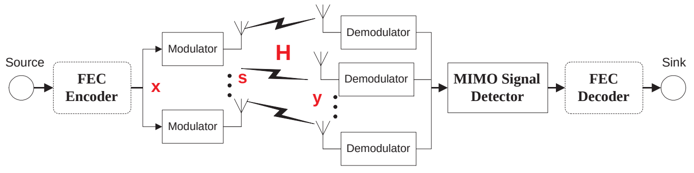

Fig. 1. A coded MIMO system model.

receive antennas, respectively. Assume that the transmitted symbol is taken from a W-QAM constellation with $W = 2^q$. At once, the transmitter maps one $qN_t \times 1$ binary vector $\mathbf{x}$ to an $N_t \times 1$ symbol vector $\mathbf{s}$. The transmission of each vector $\mathbf{s}$ over MIMO channels can be modelled as $\mathbf{y} = \mathbf{H} \cdot \mathbf{s} + \mathbf{n}$, where $\mathbf{y}$ is an $N_r \times 1$ signal vector received by a MIMO detector, $\mathbf{H}$ is an $N_r \times N_t$ channel matrix, and $\mathbf{n}$ is a noise vector whose entries are independent complex Gaussian random variables with mean zero and variance $N_0/2$.

B. MIMO Signal Detection

Following the principle of maximum likelihood (ML) detection, the task of the hard-output detector is to solve

$$\min_{\mathbf{s}\in\Omega} \|\mathbf{y} - \mathbf{H} \cdot \mathbf{s}\|^2, \tag{1}$$

where Ω contains all the W^{N_t} possible transmitted symbol vectors. The task of the soft-output detector is to compute the log-likelihood value of each bit, which is defined as $L(x_i|\mathbf{y}) = \ln \frac{P(x_i=+1|\mathbf{y})}{P(x_i=-1|\mathbf{y})}$, where x_i denotes the ith bit of the binary vector $\mathbf{x}$. Through standard simplification [9], [10], $L(x_i|\mathbf{y})$ can be approximated as:

$$\begin{aligned} L(x_i|\mathbf{y}) &\approx \max_{x_i=+1}\{\Lambda(\mathbf{x},\mathbf{y}\} - \max_{x_i=-1}\{\Lambda(\mathbf{x},\mathbf{y})\}, \\ \text{where} \quad \Lambda(\mathbf{x},\mathbf{y}) &= -\frac{1}{N_0}\|\mathbf{y} - \mathbf{H}\cdot\mathbf{s}\|^2. \end{aligned} \tag{2}$$

In a straightforward manner, hard- and soft- output MIMO detection can be realized by *exhaustively* examining all the W^{N_t} possible symbol vectors according to (1) and (2), which nevertheless leads to computational complexity prohibitive for practical applications when N_t and/or W is large.

As discussed in the literature (e.g., see [9], [10]), we may use the following well-known approach to reduce the computational complexity at the cost of potential performance degradation: Using standard matrix decompositions such as Cholesky or QR decomposition, we can obtain $\mathbf{H}^*\mathbf{H} = \mathbf{L}^*\mathbf{L}$, where $\mathbf{L} = (l_{i,j})$ is a lower triangular matrix and $(\cdot)^*$ denotes the complex conjugate transpose. Let $\hat{\mathbf{s}} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{y}$, we have

$$\begin{aligned} \|\mathbf{y} - \mathbf{H}\cdot\mathbf{s}\|^2 &= (\mathbf{s} - \hat{\mathbf{s}})^*\mathbf{L}^*\mathbf{L}(\mathbf{s} - \hat{\mathbf{s}}) \\ &\quad + \mathbf{y}^*(\mathbf{I} - \mathbf{H}(\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*)\mathbf{y}. \end{aligned} \tag{3}$$

Since the second term in (3) is independent of $\mathbf{s}$ and the matrix $\mathbf{L}$ is lower triangular, we can rewrite (1) and $\Lambda(\mathbf{x},\mathbf{y})$ in (2) as

$$\min_{\mathbf{s}\in\Omega}\left(\sum_{i=1}^{N_t}\left|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)\right|^2\right) = \min_{\mathbf{s}\in\Omega}\left(\sum_{i=1}^{N_t}\Lambda_i^h\right) \tag{4}$$

and

$$\Lambda(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_t}\left(-\frac{1}{N_0}\left|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)\right|^2\right) = \sum_{i=1}^{N_t}\Lambda_i^s. \tag{5}$$

Hence, we obtain *additive metrics* with the metric increments Λ_i^h and Λ_i^s that depend only on s_j for $j \le i$. This can be leveraged to design detectors based on an N_t-depth W-ary tree as illustrated in Fig. 2, where each node has W child nodes labelled with $1, 2, \ldots, W$, respectively, corresponding to the W possible QAM points. The i-th depth of this tree corresponds to the i-th transmit antenna. The objective of the hard-output detector is to non-exhaustively search through this tree and find a tree leaf[1] that is the solution of (1). The objective of the soft-output detector is to non-exhaustively search through this tree and find a list of tree leaves, based on which the L-values can be evaluated according to (2). The detector can search the tree in a depth-first, metric-first, or breadth-first manner.

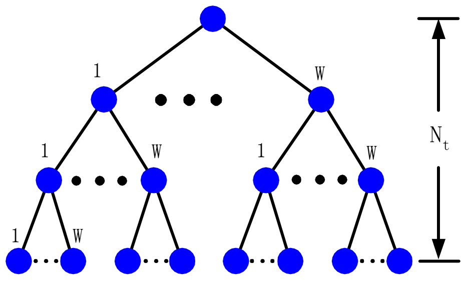

Fig. 2. An N_t-depth W-ary tree.

C. Breadth-First Search K-Best Detector

This work concerns the design of the breadth-first tree search MIMO signal detector. Broadly speaking, breadth-first tree search algorithms extend all the survivor paths at each tree depth at once, purge some paths according to certain criterion, and then continue on to the next tree depth. Various breadth-first algorithms, including *M*-algorithm and *T*-algorithm, primarily differ on the purging rules. Interested readers may refer to [11]. As mentioned earlier, following the convention in the literature on MIMO signal detection, we refer the breadth-first detector based on the principle of *M*-algorithm as *K*-best detector [5], [6]. Since the term $-\frac{1}{N_0}$ in (5) can be omitted in the tree search, we re-define the metric increment Λ_i^s for soft-output detection as $|\sum_{j=1}^{i} l_{i,j}(s_j - \hat{s}_j)|^2$, which becomes equivalent to the metric increment Λ_i^h for hard-output detection. Therefore, we simply denote the metric increment as Λ_i and define the metric of a depth-n path as $\varGamma^{(n)} = \sum_{i=1}^{n} \Lambda_i$. A *K*-best detector performs the following operations at depth d:

1) *Path Extension*: Given the modulation size of W, extend each survivor path from the previous depth with the W modulation points, i.e., calculate $\varGamma^{(d)} = \varGamma^{(d-1)} + \Lambda_d$ for each modulation point.
2) *Radius Check*: Delete the extended paths whose metrics are larger than a pre-defined value r. Here r is equivalent to the radius in sphere decoding.
3) *Path Search*: Let *R* denote the number of the remaining extended paths. Sort the *R* extended paths in ascending order based on the path metric and select the first $\min(R, K)$ paths as survivors at depth d.

Hard- and soft- output *K*-best detectors only differ at how to generate the output using the survivors obtained after reaching the tree leaves: (a) Hard-output detector finds the best one among all the survivors and outputs the hard decision of each bit based on this final survivor; (b) Soft-output detector keeps all the survivors as a list of candidates, based on which the L-values are calculated according to (2). In general, to ensure near-optimum performance, a soft-output detector typically requires a (much) larger value of *K* than that of its hard-output counterpart.

III. Partially Parallel Relaxed *K*-Best Detector Architecture

For the VLSI implementation of a *K*-best detector, the *Path Extension* (and hence *Radius Check*) can be, in theory, implemented in fully parallel, i.e., at each depth all the survivors are extended in parallel. Because of the complex computation involved in the path extension and the throughput bottleneck incurred by the *Path Search* (as discussed later), the fully parallel implementation is impractical and/or unnecessary. This work only considers the partially parallel detector that maps a certain number of path extension operations onto the same hardware processing unit in a time-division multiplexed mode.

From the Section II-B, we have that the calculation of metric increments at depth-i can be written as $|P_c - P_s|^2$, where $P_c = \sum_{j=1}^{i-1} l_{i,j}(\hat{s}_j - s_j) + l_{i,i}\hat{s}_i$ that is common to all the paths extended from the same survivor, and $P_s = l_{i,i}s_i$ that corresponds to each modulation point. Therefore, to extend one survivor, we need to calculate only one P_c but multiple $|P_c - P_s|^2$. Straightforwardly, we have the generic structure of the processing unit at each depth as shown in Fig. 3. Each PC block calculates the P_c, and each PS block calculates the metrics, i.e., $\varGamma^{(i-1)} + |P_c - P_s|^2$, of all the paths extended from the same survivor and deletes those that fail the radius check. Due to the computational complexity mismatch between PC and PS blocks, several PS blocks can share one PC block. The total number of PC blocks and PS blocks can be much less than the value of *K*. The output of all the PS blocks, i.e., the extended paths that pass the radius check, are sent to a search block that selects the best *K* extended paths as survivors.

However, such straightforward structure has two critical drawbacks that prevents it from achieving high throughput with reasonable silicon area and power consumption, particularly for high order modulation such as 64-QAM:

1) The detector *explicitly* examines the extension of each survivor with all the modulation points. Due to the complex computation involved in each path extension,

[1]Notice that each tree leaf determines one distinct path through the tree, corresponding to one distinct $qN_t \times 1$ bit vector.

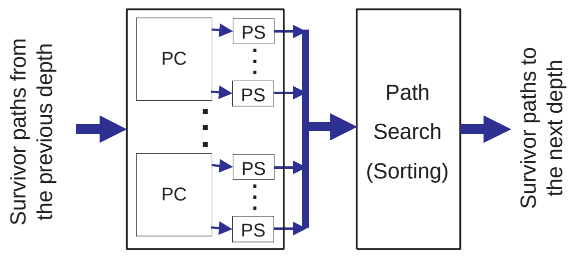
Survivor paths from the previous depth
PC
PS
PC
PS
Path Search (Sorting)
Survivor paths to the next depth

Fig. 3. Generic structure of the processing unit at each depth in the partially parallel K-best detector.

this will incur a large computational complexity overhead.

2) Due to its serial nature, the sorting at each depth will incur a large delay and hence become an essential throughput bottleneck. This fails to match the inherent parallelism within the path extension and radius check. Although there exists an algorithm, as pointed out in [8], that can select *the best M out of N numbers* more efficiently without using sorting, the most effective way to realize the search-the-best-K-paths operation in hardware is sorting, such as the bubble sorting used in [5] for hard-output K-best detector. Besides the large delay, sorting will also incur large silicon overhead and a large amount of data movement, which will directly lead to high power consumption.

The first issue can be tackled by using the PSK enumeration method first proposed in [9] and further simplified in [4]. To tackle the second issue, we developed a modified K-best detector, called *relaxed* K-best detector, where the key idea is to replace the original strict sorting with a memory based distributed and approximate sorting. Such algorithm-level relaxation can be directly leveraged to improve the hardware implementation performance. In the following, we first describe the proposed relaxed sorting, then present the overall relaxed K-best detector structure. In section IV we will present a proof-of-concept hardware design of a soft-output relaxed K-best detector that can support 4×4 MIMO transmission with 64-QAM modulation.

A. Distributed and Approximate Sorting

The principle of the distributed and approximate sorting is illustrated in Fig.4(a), where each PS block has its own sorter and all the sorters perform approximate sorting independent from each other. The basic idea of approximate sorting can be described as *sorting at the level of groups of paths*, in contrast to the strict sorting at the level of individual paths, i.e., we divide the entire range of the path metric into a certain number of regions and group the paths whose path metrics fall into the same region, and within each group there is no further sorting. Clearly, such approximate sorting only involve the comparison with fixed threshold values, which can be directly implemented in parallel.

Intuitively, we can use a single-port memory to implement such approximate sorting as follows: We uniformly partition the memory address space into l consecutive segments. Since all the incoming paths have the metric better (i.e., less) than the radius r that is used in the radius check, we choose $l+1$ threshold values $t_0 = 0 < t_1 < t_2 < \cdots < t_l = r$, and assign the range $(t_{i-1}, t_i]$ to segment S_i for $i = 1, 2, \cdots, l$. Each path, whose metric falls into the range $(t_{i-1}, t_i]$, is simply stored into the memory segment S_i. Each segment has one counter to hold the address of the next available memory location. Clearly, among the data stored within the same memory segment, there is no sorting at all. For the practical implementation of such approximate sorting, we need to solve the following two problems:

1. What should we do if one memory segment becomes full? Our simulations show that if we simply throw away any further incoming path after the segment becomes full, there will be a significant performance degradation. To this end, we propose to make the threshold range associated with each memory segment configurable, as illustrated in Fig.4(b). Let $(u_{i-1}, u_i]$ represent the configurable threshold range associated with segment S_i. Initially, we set $u_i = t_i$ for $i = 1, 2, \cdots, l$. Once one segment becomes full, we will hand over its range to its closest next (with higher value of index) segment that is not full yet. For example, before the segment S_i becomes full, we have $u_i = t_i$. Once S_i becomes full, by using a switch as shown in Fig.4(b) we configure $u_i = u_{i-1}$ to prevent any further write to this segment. Meanwhile, the lower bound of the threshold range of the closest next segment that is not full yet will automatically extend from t_i (i.e., the previous value of u_i) to u_{i-1}. In this way, the range of S_i is handed over to the closest next segment that is not full yet.

2. How to determine the l threshold values $t_1, t_2, \cdots, t_l$? In this work, we first calculate the radius $r = \alpha N_r \sigma^2$, as proposed in [9] for sphere decoding, where α is a predefined constant parameter and σ is the noise standard deviation. We have $t_l = r$ and calculate the other $l-1$ threshold values as $t_i = i \cdot t_l / l$ for $i = 1, 2, \cdots, l-1$.

The survivor paths at each depth are feed to the next depth as follows: We start with fetching one path in segment S_1 as survivor from each single-port memory at one time, alternatively among all the single-port memories, until we have fetched K paths or all the path stored in all the S_1 segments have been fetched. If latter happens, we move on to the segment S_2, and so on. From the hardware implementation standpoint, this memory based distributed and

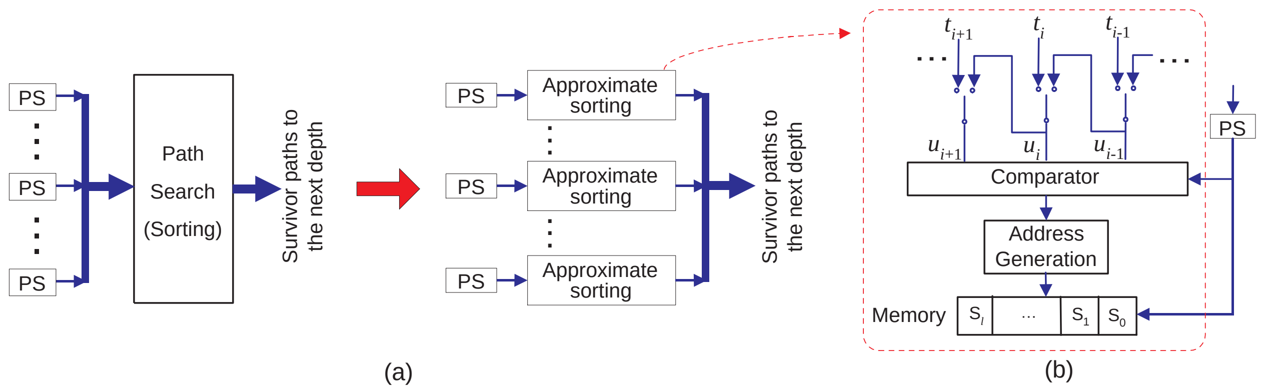

Fig. 4. (a) Structure of the distributed and approximate sorting, and (b) realization of the approximate sorting.

approximate sorting has the following main advantages: (i) The computational complexity is much less than the strict sorting, leading to a great potential of higher throughput and significant power savings; (ii) There is no data movement at all as in the strict sorting, hence further reduce the power consumption; (iii) The distributed structure well matches the parallelism in the path extension and hence help to realize high throughput.

Finally, we note that the value of l and the total memory size affect the trade-off between silicon area and detection performance: a bigger value of l and/or a larger size of memory will improve the detection performance at the cost of higher implementation complexity. In practice, we have to rely on the extensive simulation to choose the value of l and total memory size subject to the desired silicon area vs. detection performance trade-offs. As demonstrated in Section IV, with reasonable memory resource, the relaxed K detector using the distributed and approximate sorting can achieve almost the same detection performance as the detector using sphere decoding algorithm with the same value of radius r.

B. Detector Hardware Structure

This section discusses the overall relaxed K-best detector structure design. As mentioned above, to reduce the computational complexity of each PS block in the path extension, we use the PSK enumeration technique proposed in [9] and further simplified in [4]. The basic idea can be described as follows: For QAM modulation, all the constellation points locate on several circles concentric with the origin, e.g., there are 1, 3, and 9 such concentric circles in QPSK, 16-QAM, and 64-QAM, respectively. It can be proved that all the points on the same circle that satisfy the radius check are always adjacent and hence form a single admissible region along the circle. We can identify the boundary of the admissible region of each circle and do not explicitly examine the points outside the admissible region. In this way, a large percentage of computational complexity can be saved compared with explicitly examining all the constellation points in the path extension. Readers are referred to [9] and [4] for a more detailed description.

Fig.5 shows two partially parallel relaxed K-best detector structures with different trade-offs between silicon area and throughput. The recursive structure in Fig.5(a) uses the same depth processing unit (DPU) for all the tree search depths, while the unrolled structure in Fig.5(b) maps each tree search depth onto different DPU. The recursive structure occupies less silicon area and can readily support a variable number of tree depths (i.e., variable number of transmit antennas), while the unrolled structure can realize higher throughput. Moreover, since the expected number of survivor paths at the first one or two tree depth may be less than the value of K, particularly for soft-output detection with a large value of K, the number of functional elements and the size of memory at the first one or two stages can be less than that of the following stages. Hence, contrary to the first impression, the silicon area of the unrolled structure is less than N_t times of the area of the recursive structure.

Similar to the generic structure of a partially parallel K-best detector discussed above, as shown in Fig.5, the path extension block at each depth contains several PC blocks, and each PC block is shared by several PS blocks. Extending one survivor path at a time, each PS block consists of one or more PSK units, where each PSK unit non-exhaustively examines the constellation points on the same circles once a time. Since one PS block connects one approximate sorting block and all the PSK units may generate extended path that passes the radius check every clock cycle, the number of PSK units in each PS block is limited by the ratio between the speed of approximate sorting and the speed of PSK unit (notice that, because of the simple structure of approximate sorting and inherent serial nature of PSK operation, the speed of the approximate sorting can be several times faster than that of a PSK

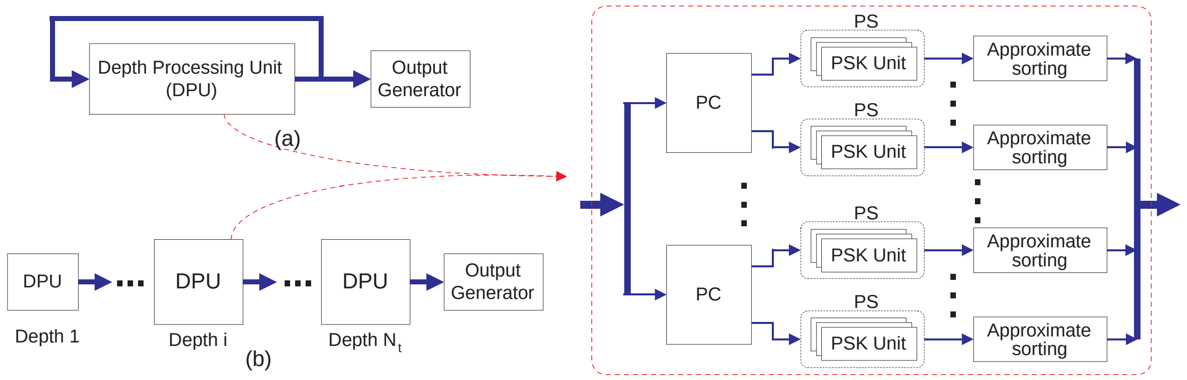

Fig. 5. Two possible structures of partially parallel relaxed *K*-best detector.

unit). In practice, the numbers of PC blocks and PS blocks are dependent on the target modulation size and desired throughput.

As shown in Fig.5, an output generator is used to generate the detector output based on the last-depth survivors. For the hard-output detection, the design of the output generator is straightforward, i.e., it simply searches the survivor path with the best metric and output the qN_t hard decisions based on this best survivor. For the soft-output detection, we need to evaluate the L-value of each bit according to (2) based on all the last-depth survivors. In this context, we can use a simple scheme described as follows:

We maintain a table, as illustrated in Table I, for the $qN_t \times 1$ binary vector $\mathbf{x}$, where each cell contains the best path metric, denoted by $\varGamma_{i,+1}$ or $\varGamma_{i,-1}$, among the survivors that has "$+1$" or "-1" on the corresponding bit position. The metrics in the table are initially set as undefined. Meanwhile, we keep a metric called $\varGamma_{bst}$ that represents the overall best path metric. The survivor paths sent from the approximate sorting blocks are processed one by one. The metric of each incoming survivor path is compared to $\varGamma_{bst}$ first. If its metric is better than $\varGamma_{bst}$, then we replace $\varGamma_{bst}$ with this new path metric and directly replace the metrics of the corresponding cells without any further comparison. Otherwise, it is compared in parallel with the metrics of the corresponding cells, and replace the metrics that are worse than this new path metric. After all the survivor paths have been processed, the L-values can be directly calculated based on the table according to (2).

TABLE I
METRIC TABLE FOR CALCULATING THE SOFT OUTPUT.

	x_0	x_1	$\cdots$	x_i	$\cdots$	x_{qN_t-1}
$+1$	$\varGamma_{0,+1}$	$\varGamma_{1,+1}$	$\cdots$	$\varGamma_{i,+1}$	$\cdots$	$\varGamma_{qN_t-1,+1}$
-1	$\varGamma_{0,-1}$	$\varGamma_{1,-1}$	$\cdots$	$\varGamma_{i,-1}$	$\cdots$	$\varGamma_{qN_t-1,-1}$

Finally, we note that it is possible that all the survivor paths in the survivor paths agree on one bit position, i.e., the corresponding $\varGamma_{i,+1}$ or $\varGamma_{i,-1}$ remains as undefined. As a result, the L-value of this bit cannot be directly calculated. To solve this problem, in this work, we use the worst metric among all the metrics in the table to replace all the undefined entries.

IV. DESIGN EXAMPLE

In this work, to evaluate the hardware implementation effectiveness of the proposed relaxed *K*-best detector, we designed a soft-output relaxed *K*-best detector that can support 4×4 MIMO transmission with 64-QAM modulation. The design entry is Verilog description, which is synthesized using Synopsys tool sets with $0.18\mu m$ standard cell and single-port SRAM libraries. To achieve high throughput, we use the 4-stage unrolled detector structure as described above. The design parameters of the detector are outlined as follows: The value of *K* is 256 for 64-QAM. There are one PC block and 16 PS blocks at each stage. Each PS block consists of 2 PSK units. To support the pipelining, each approximate sorter contains two single-port memory blocks that receive the data from the current depth and provide the data to the next depth, alternatively. Each single-port memory can store totally 128 path data and is partitioned into 16 segments. Since the number of survivor paths at most can be 64 at the first tree depth, the first stage is specifically simplified as follows: It only contains one PC block and one PS block, and the approximate sorter is replaced by two single-port memory blocks, each of which can store 64 path data.

This detector operates with two synchronized clock signals: each PSK unit operates with a clock frequency of 200 MHz, all the other functional blocks including the PC blocks and the approximate sorters operate with a clock frequency of 400 MHz. The entire detector occupies about 20 mm^2 of silicon area, among which 7 mm^2 is occupied by the single-port SRAMs. The detection throughput actually depends on the average number of survivors generated at each depth, which largely depends on the run-time environment such as average SNR and instantaneous channel gain. To estimate the average number of survivor paths and hence the throughput, we did computer simulation with the following assumptions: We use orthogonal frequency division multiplexing (OFDM) with 64-point FFT as in the IEEE 802.11a standard for the MIMO transmission. Each sub-carrier MIMO channel is flat fading, i.e., all the entries in the MIMO channel matrix are independent random Gaussian variables. To define the MIMO channel SNR, we follow the one presented in [9]: Let R denote the channel code rate ($R = 1$ for uncoded systems), SNR is defined as:

$$\frac{E_b}{N_0}\Big|_{dB} = \frac{E_s}{N_0}\Big|_{dB} + 10\log_{10}\frac{N_r}{R \cdot N_t \cdot q},$$

where E_s denotes the average symbol energy of the QAM constellation. In the simulation, the soft-output is feed to a rate-1/2 LDPC code with the code length of 2048. Based on the simulation results, this detector can achieve a throughput of 30 Mbps at the SNR of 15.7dB and 50Mbps at the SNR of 17.2 dB. We note that this detector can also be configured to support the soft-output detection for 16-QAM and QPSK and hard-output detection for 64-QAM, 16-QAM, and QPSK, where much higher throughput can be realized mainly because of the much less average survivor paths, particularly in hard-output detection. For the above coded MIMO-OFDM system, Fig.6 shows the simulated block error rate performance. For the purpose of comparison, we did the same simulation with the soft-output detection using the popular sphere decoding algorithm, where the radius r and the maximum number of survivors for calculating the soft-output are same as in the above relaxed K-best detector. As shown in Fig.6, the relaxed K-best detector only incurs about 0.1$\sim$0.2 dB degradation. However, it remains unknown of how to (or whether it is possible to) practically implement such a sphere decoder that can support 4$\times$4 MIMO transmission with 64-QAM modulation in hardware with reasonable silicon area and high throughput.

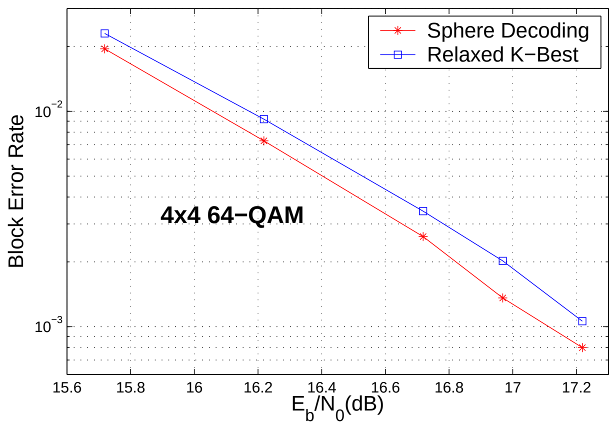

Fig. 6. Simulation results for 4$\times$4 64-QAM.

V. Conclusions

This paper for the first time presents a nonlinear MIMO signal detector hardware design solution that effectively supports 4$\times$4 MIMO transmission with 64-QAM modulation. The detector employs the breadth-first tree search style to realize nonlinear detection. An algorithm-level modification is developed to tackle the implementation challenge of the conventional breadth-first tree search MIMO detection scheme. A proof-of-concept hardware prototype of a soft-output detector that supports 4$\times$4 MIMO transmission with 64-QAM has been designed using 0.18 μm CMOS technology. It can achieve up to 50 Mbps throughput with a silicon area of 20 mm^2.

References

[1] M. Shafi et al. (ed.), "Special Issues on MIMO Systems and Applications (I/II)," *IEEE Journal on Selected Areas in Communications*, vol. 21, April/June 2003.

[2] N. Jayant (ed.), "Special Issue on Gigabit Wireless," *Proceedings of the IEEE*, vol. 92, Feb. 2004.

[3] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, pp. 463–471, April 1985.

[4] A. Burg et al., "VLSI implementation of MIMO detection using the sphere decoding algorithm," *Journal of Solid-State Circuits*, vol. 40, pp. 1566 – 1577, July 2005.

[5] K.-W. Wong, C.-Y. Tsui, R. S. 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*, May 2002, pp. III–273–III–276.

[6] Z. Guo and P. Nilsson, "VLSI architecture of the Schnorr-Euchner decoder for MIMO systems," in *Proc. of IEEE CAS Symposium on Emerging Technologies*, 2004, pp. 65–68.

[7] D. Garrett et al., "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1544–1552, Sept. 2004.

[8] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Transactions on Communications*, vol. 32, pp. 169–176, Feb. 1984.

[9] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, March 2003.

[10] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. of IEEE International Conference on Communications*, May 2003, pp. 2653–2657.

[11] J. B. Anderson, "Limited search trellis decoding of convolutional codes," *IEEE Transactions on Information Theory*, vol. 35, pp. 944–955, Sept. 1989.