Relaxed K-Best MIMO Signal Detector Design and VLSI Implementation

Sizhong Chen, *Student Member, IEEE*, Tong Zhang, *Member, IEEE*, and Yan Xin, *Member, IEEE*

***Abstract*—Signal detector is a key element in a multiple-input multiple-output (MIMO) wireless communication receiver. It has been well demonstrated that nonlinear tree search MIMO detectors can achieve near-optimum detection performance, nevertheless their efficient high-speed VLSI implementations are not trivial. For example, the hardware design of hard- or soft- output detectors for a 4 $\times$ 4 MIMO system with 64 quadrature amplitude modulation (QAM) still remains missing in the open literature. As an attempt to tackle this challenge, this paper presents an implementation-oriented breadth-first tree search MIMO detector design solution. The key is to appropriately modify the conventional breadth-first tree search detection algorithm in order to largely improve the suitability for efficient hardware implementation, while maintaining good detection performance. To demonstrate the effectiveness of the proposed design solution, using 0.13-μm CMOS standard cell and memory libraries, we designed a soft-output signal detector for 4 $\times$ 4 MIMO with 64-QAM. With the silicon area of about 31 mm^2, the detector can achieve above 100 Mb/s and realize the performance very close to that of the sphere decoding algorithm.**

***Index Terms*—Detection, multiple-input multiple-output (MIMO) systems, maximum likelihood (ML), spatial multiplexing, VLSI.**

I. Introduction

DUE TO ITS potential of dramatically increasing the wireless communication spectral efficiency, multiple-input multiple-output (MIMO) technology is being seriously considered for a wide use in future high data rate wireless communication systems such as the fourth-generation (4G) mobile radio systems, fixed/mobile broadband wireless systems, and wireless local area networks [1], [2]. As the cost of the increased transmission rate, the computational complexity of MIMO signal detection grows dramatically with the number of transmit antennas and the modulation constellation size. How to design reduced-complexity signal detectors without significantly jeopardizing the detection performance is a key issue for the practical deployment of wireless MIMO communication systems.

MIMO signal detectors can be either hard-output (i.e., only provides the hard estimation of each bit) or soft-output (i.e., provides *a posteriori* probability (APP) information about each bit). In practice, most wireless communication systems use error correcting codes (ECC) demanding soft input for decoding, such as convolutional codes, Turbo codes, and low-density parity-check (LDPC) codes. Therefore, soft-output MIMO signal detection is highly desirable. The computational complexities of maximum-likelihood (ML) hard-output detection and maximum *a posteriori* (MAP) soft-output detection grow exponentially with the transmission rate measured in terms of *bits per channel use*. The computational complexity of soft-output MAP detection is significantly higher than that of hard-output ML detection. With the help of continuous scaling down of CMOS technology, it may be feasible to implement ML/MAP detectors for MIMO systems with relatively low transmission rate such as 4 $\times$ 4 MIMO with quadrature phase shift keying (QPSK) or 2 $\times$ 2 MIMO with 16 quadrature amplitude modulation (QAM) [3]. Nevertheless, the direct implementation of such optimum detections will become impractical when the bits per channel use increase beyond eight [4], and techniques to reduce computational complexity become necessary under these circumstances.

One family of reduced-complexity detectors is linear detectors based on the principles of zero-forcing (ZF) or minimum mean-square error (MMSE). Although they can greatly reduce the computational complexity, they suffer from significant performance degradation. The successive interference cancellation (SIC) detectors such as the VBLAST architecture [5] are prone to decision error propagation and can only provide modestly better performance. Recently, lattice reduction (LR) aided MMSE or SIC detectors were proposed [6], [7] and showed a great potential to improve the performance of linear detectors. However, most prior work only considered hard-output LR aided detection and effective soft-output LR aided detector design largely remains an open question [8]. To achieve the performance closer or even equivalent to the optimum detection, researchers have developed several nonlinear detectors that realize hard- or soft- output detection through *nonexhaustive 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. Because of their computation-intensive nature, those nonlinear MIMO detectors should be implemented in the form of application specific integrated circuits (ASIC) 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. VLSI design of depth-first detectors using sphere decoding algorithm [9], [10]

Manuscript received January 4, 2006; revised October 25, 2006. This work was supported by the National Science Foundation under Grant 0522457.

S. Chen and T. Zhang are with Electrical and Computer Science Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: chens6@rpi.edu; tzhang@ecse.rpi.edu).

Y. Xin is with the Electrical and Computer Engineering Department, National University of Singapore, 117576 Singapore (e-mail: elexy@nus.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2007.893621

and breadth-first detectors using the well-known M-algorithm [11] have both attracted many recent attentions [4], [12]–[15]. For 4 $\times$ 4 MIMO transmission with 16-QAM, hard-output depth-first detectors [13] achieve much higher throughput (under high signal to noise ratio (SNR)) than their breadth-first counterparts [12], [14]. A soft-output depth-first detector for 4 $\times$ 4 MIMO with 16-QAM was described in [4], and a soft-output breadth-first detector for 4 $\times$ 4 MIMO with 16-QAM was presented in [15]. In general, the average computational complexity of depth-first hard-output detection is lower than its breadth-first counterpart due to its ability to adaptively tighten the search radius constraint. For soft-output detection, it is not trivial to adaptively change the search radius for both depth-first search and breadth-first search, while the breadth-first search has the advantage that it can naturally generate an ordered candidates list for APP calculation. To the best of our knowledge, VLSI design of nonlinear tree search detectors that can support 64-QAM for moderate-size MIMO (such as 4 $\times$ 4) remains missing in the open literature.

As an attempt to fill this gap, this paper presents a breadth-first hard- and soft-output detector design solution that can support 64-QAM for 4 $\times$ 4 MIMO transmission. 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 [12], [14]. As discussed later, the direct realization of a K-best detector in hardware requires implementing a sorting operation. Since sorting is essentially a serial operation and may involve a large amount of data movement, it will become a throughput bottleneck and result in a significant power consumption overhead. To tackle this challenge, instead of directly following the principle of M-algorithm as in the conventional K-best detectors [12], [14], we propose to modify the original algorithm by replacing the strict sorting with a *distributed* and *approximate* sorting. Such algorithm-level modification can significantly simplify the hardware implementation, leading to so-called relaxed K-best detectors. Moreover, we developed an algorithm-level method based on the PSK enumeration technique [10], [13] to further reduce the computational complexities, particularly for higher order modulations. The corresponding VLSI architectures are further developed. To demonstrate the proposed design solution, we designed a soft-output relaxed K-best detector for 4 $\times$ 4 MIMO with 64-QAM using Synopsys tools with 0.13-μm CMOS standard cell and memory libraries. With the silicon area of about 31 mm^2 and power consumption of 1.2 W, the detector can achieve above 100 Mb/s throughput. Concatenated with a rate-1/2 length-2304 low-density parity-code (LDPC) code, the soft-output detector can achieve the performance very close to that of using original K-best and sphere decoding algorithms. To the best of our knowledge, this is the first soft-output nonlinear tree-search detector design solution capable of supporting 4 $\times$ 4 MIMO with 64-QAM ever reported in the open literature.

The remainder of this paper is organized as follows. Section II reviews the background of MIMO signal detection. Section III describes the proposed relaxed K-best detection algorithm and its VLSI architecture. The implementation results and performance analysis for 4 $\times$ 4 MIMO with 64-QAM are presented in Section IV, and conclusions are drawn in Section V.

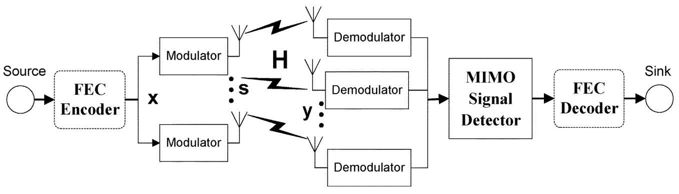

Fig. 1. Coded MIMO system model.

II. Background

A. System Model

This paper considers the 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 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 flat-fading 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 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 $L(x_i|\mathbf{y}) = \ln(P(x_i = +1|\mathbf{y})/P(x_i = -1|\mathbf{y}))$, where x_i denotes the i-th bit of the binary vector $\mathbf{x}$. Through standard simplification [10], [16], $L(x_i|\mathbf{y})$ can be approximated as

$$L(x_i|\mathbf{y}) \approx \max_{x\in\mathbb{X}_{i,+1}} \{\Lambda(\mathbf{x},\mathbf{y})\} - \max_{x\in\mathbb{X}_{i,-1}} \{\Lambda(\mathbf{x},\mathbf{y})\},$$
$$\text{where} \quad \Lambda(\mathbf{x},\mathbf{y}) = \frac{1}{N_0}||\mathbf{y} - \mathbf{H}\cdot\mathbf{s}||^2. \tag{2}$$

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 [10], [16]), we may use the following approach to reduce the computational complexity at the cost of potential performance degradation: Using standard matrix decompositions such as Cholesky 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

$$||\mathbf{y}-\mathbf{H}\cdot\mathbf{s}||^2 = (\mathbf{s}-\hat{\mathbf{s}})^*\mathbf{L}^*\mathbf{L}(\mathbf{s}-\hat{\mathbf{s}})+\mathbf{y}^*\left(\mathbf{I} - \mathbf{H}(\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\right)\mathbf{y}. \tag{3}$$

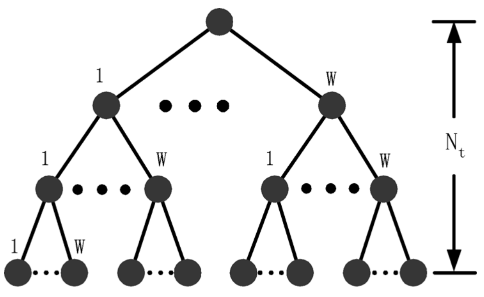

Fig. 2. An N_t-depth W-ary tree.

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 $j \leq i$. 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.

C. Breadth-First Search K-Best Detector

This work concerns the design of breadth-first tree search detectors. 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 the well-known M-algorithm and T-algorithm, primarily differ on the purging rules, and interested readers may refer to [17] for a detailed discussion. 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 the K-best detector. Since the term $-(1/N_0)$ in (5) can be omitted in the tree search, we redefine 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 $\Gamma^{(n)} = \sum_{i=1}^{n}\Lambda_i$. 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 $\Gamma^{(d)} = \Gamma^{(d-1)} + \Lambda_d$ for each modulation point.
2) *Radius Check*: Delete the extended paths whose metrics are larger than a predefined value r^2. Here r is equivalent to the radius in sphere decoding algorithm.
3) *Path Search*: Let R denote the number of the remaining extended paths. If $R > K$, then sort the R extended paths in ascending order based on the path metric and select the first (i.e., best) K paths as survivors of present depth, otherwise all the R extended paths are survivors.

Hard- and soft- output K-best detectors only differ at how to generate the output using the survivors obtained after reaching the tree leaves: 1) hard-output detector finds the best one among all the survivors and outputs the hard decision of each bit based on this final survivor; 2) 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. Moreover, in soft-output detection, if all the candidates agree on one bit position (i.e., they all contain a $+1$ (or -1) at the same position), we cannot directly calculate the L-value for this bit. To solve this problem, we may use the difference between the best and worst metrics among all the candidates as the soft information of those bits or assign a predefined L-value.

III. Relaxed K-Best Detector Design

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 paper 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 fashion.

From Section II, we have that each metric increment Λ_i at depth-i can be calculated as $|P_c - P_s|^2$, where $P_c = G_i - \sum_{j=1}^{i-1} l_{i,j}s_j$, $G_i = \sum_{j=1}^{i} l_{i,j}\hat{s}_j$ and $P_s = l_{i,i}s_i$. G_i is common to each received vector and can be precomputed. P_c is common to all the paths extended from the same survivor, while P_s depends on the modulation point to which the survivor is extended. Therefore, to extend one survivor, we need to calculate only one P_c but multiple P_s and hence multiple $|P_c - P_s|^2$. In a straightforward manner, we can obtain a generic data flow at each depth, as shown in Fig. 3. Each PC block calculates P_c of one survivor path, and the PS blocks associated with the PC block calculate the metrics, i.e., $\Gamma^{(i-1)} + |P_c - P_s|^2$, of all the paths extended from the same survivor path and delete 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 outputs of all the PS blocks, i.e., the

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

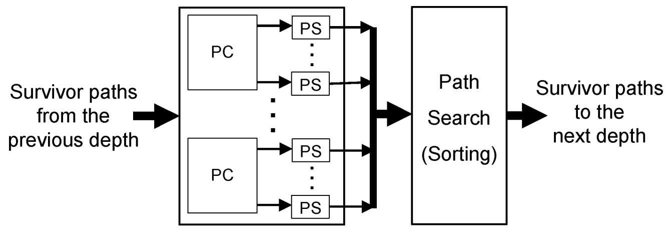
PC
PS
PS
PC
PS
PS
Survivor paths from the previous depth
Path Search (Sorting)
Survivor paths to the next depth

Fig. 3. Generic data flow at each depth of a partially parallel K-best detector.

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 design has two critical drawbacks that prevent 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, 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 [11], which can select *the best M out of N numbers* more efficiently without using strict sorting, that algorithm is still serial in nature. Moreover, from hardware implementation point of view, sorting schemes like the bubble sorting appear to be the most effective way to realize the search-the-best-K-paths operation. Besides the long latency, sorting will also incur large silicon overhead and a large amount of data movement, which will directly lead to high power consumption.

A. Improved PSK Enumeration

How to tackle the first issue above has been addressed in the context of depth-first tree search detection [10], [13] using a technique called PSK enumeration. Its basic idea is described as follows: For QAM modulation, all the modulation points locate on several circles concentric with the origin, e.g., there are 1, 3, and 9 concentric circles in QPSK, 16-QAM, and 64-QAM, respectively. 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. By identifying the boundary of such an admissible region on each circle, we do not need to explicitly examine the points outside the admissible region. This will lead to a significant saving of computational complexity. To identify the admissible region on one circle, we first locate the point that can minimize $|P_c - P_s|$ among all the modulation points on the same circle. As proposed in [13], this can be realized by observing the location of the P_c in a partitioned space as illustrated in Fig. 4(a). Let's consider the circle I. Without loss of generality, we assume $l_{i,i}$ is a positive real number (recall that $P_s = l_{i,i}s_i$). Given the position of P_c, the point that minimizes $|P_c - P_s|$ on the circle I should be the point 1 since P_c falls into the region between line $y = x$ and line $y = x/2$. After identifying the closest point on each circle, the rest of the points on the same circle can be examined in a zigzag fashion, as illustrated in Fig. 4(b), so that the corresponding extended path metric will increase monotonically. Once the first point that fails the radius check is reached, the boundary of admissible points is automatically detected and the enumeration can be terminated.

Notice that the original PSK enumeration method explicitly examines all the concentric circles. In the following, we present a modified PSK enumeration method that does not necessarily explicitly examine all the concentric circles, which may lead to further computational complexity reduction, particularly for high order modulations such as 64-QAM. Denote the circles that contain modulation points satisfying the radius check as *valid circles*. It is clear that not all the concentric circles may be valid circles, hence it is desirable to only examine those valid circles instead of all the circles. From the discussion on PSK enumeration in [10], it can be readily derived that all the valid circles fall into a continuous region. We can identify the boundary of the valid circle region as follows: For a modulation point to survive the radius check, the metric increment at depth-i must satisfy $|P_c - l_{i,i}s_i|^2 < r^2 - \Gamma^{(i-1)}$, which can be reformulated as $|(P_c/l_{i,i}) - s_i|^2 < (r^2 - \Gamma^{(i-1)})/l_{i,i}^2$. We can easily find the circle closest to the point $P_c/l_{i,i}$ on each side, as shown in Fig. 4(c). Extending from these two circles with the distance of $r_e^2 = (r^2 - \Gamma^{(i-1)})/l_{i,i}^2$ on both inward and outward directions, we obtain the boundary of the valid circle region. Any circle that falls outside does not contain any modulation points that may satisfy the radius check, hence can be simply excluded from explicit PSK enumeration. Since we can precompute each $1/l_{i,i}$ once after the MIMO channel matrix is estimated, the computation here involves multiplication instead of division.

B. Distributed and Approximate Sorting

To tackle the second issue above, we propose to modify the K-best detector by replacing the original strict sorting with a memory based distributed and approximate sorting, as illustrated in Fig. 5, where each PS block has its own sorter. Let β denote the detector parallelism factor, i.e., the total number of PS blocks that operate in parallel. Therefore, β survivor paths can be extended in parallel and all the extended paths from the same survivor path are processed by the same sorter. All the β sorters perform approximate sorting in parallel and independently from each other. The basic idea of approximate sorting can be described as *sorting with a coarse granularity*: Given the fixed radius constraint r^2, we divide the entire range of the path metric (i.e., $[0, r^2]$) into a certain number of regions and group the paths whose path metrics fall into the same region. The paths in the same group are not sorted at all. Such approximate sorting only involves the comparison with fixed threshold values, which can be directly implemented in parallel.

Intuitively, we can use a single-port memory to realize the 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^2 that is used in the radius check, we choose $l + 1$ threshold values $t_0 = 0 < t_1 < t_2 < \cdots < t_l = r^2$, and assign the range $(t_{i-1}, t_i]$ to segment S_i for $i = 1, 2, \ldots, l$. A path is simply stored into the memory segment S_i if its metric falls into the range $(t_{i-1}, t_i]$. Each segment has one counter to hold the address of the next available memory location. Clearly, the

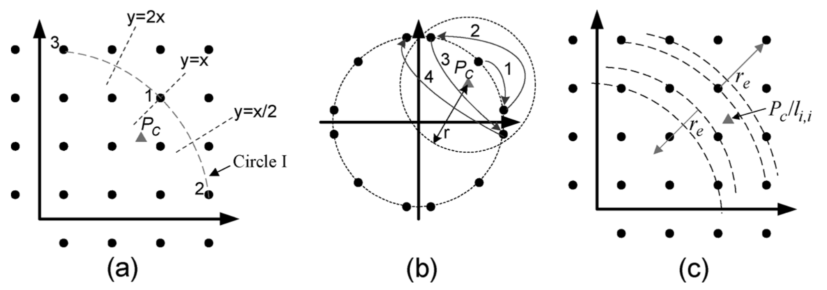

Fig. 4. Improved PSK enumeration.

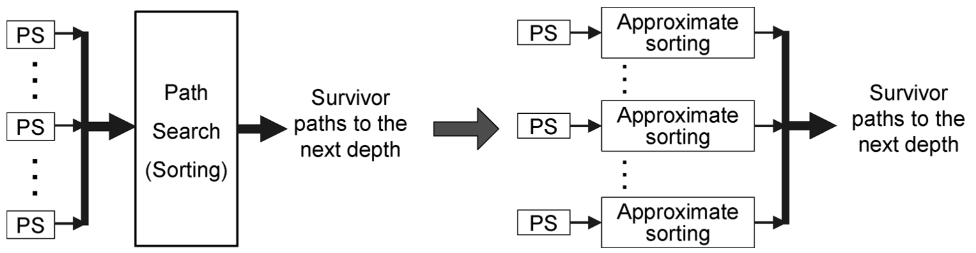

Fig. 5. Structure of the distributed and approximate sorting.

data stored within the same memory segment are no sorted at all. For the practical implementation of an approximate sorter, we need to solve the following two problems.

1) What should we do if one memory segment becomes full? Our simulation shows that if we simply throw away the subsequent incoming paths once the memory segment becomes full, there will be a significant performance degradation. In order to solve this problem, we propose to make the threshold range associated with each memory segment configurable, as illustrated in Fig. 6. Let $(u_{i-1}, u_i]$ represent the configurable threshold range associated with segment S_i. We use l registers to hold the values of $u_1, u_2, \ldots, u_l$ (notice that u_0 always equals to 0). Initially, we set $u_i = t_i$ for $i = 0, 1, \ldots, l$. Once one segment becomes full, we will hand over its range to its next (with higher value of index) segment. For example, before segment S_i becomes full, we have $u_i = t_i$. Once S_i becomes full, by using a push-button switch as shown in Fig. 6, we set $u_i = u_{i-1}$, i.e., we overwrite the register holding u_i with the value of u_{i-1}. Any further write to this segment is prevented since $u_i = u_{i-1}$. Meanwhile, the lower bound of the threshold range of the next segment S_{i+1} will automatically extend from t_i (i.e., the previous value of u_i) to u_{i-1}, so that the segment S_{i+1} now has the threshold range of $(t_{i-1}, t_{i+1}]$. In this way, the range of S_i is handed over to S_{i+1}. In case that S_{i+1} was full before S_i becomes full, i.e., S_{i+1} held the threshold range of $(t_i, t_i]$, the range of S_{i+1} now becomes $(t_{i-1}, t_i]$ and S_{i+1} is reopened to allow paths with better metrics to overwrite the paths already stored in S_{i+1}.
2) How to determine the l threshold values $t_1, t_2, \ldots, t_l$? In this work, we first calculate the radius $r^2 = 2\alpha N_r \sigma^2$, as proposed in [10] for sphere decoding, where α is a predefined constant parameter and σ is the noise standard deviation. We have $t_l = r^2$ and calculate the other $l-1$ threshold values as $t_i = i \cdot t_l / l$ for $i = 1, 2, \ldots, l-1$.

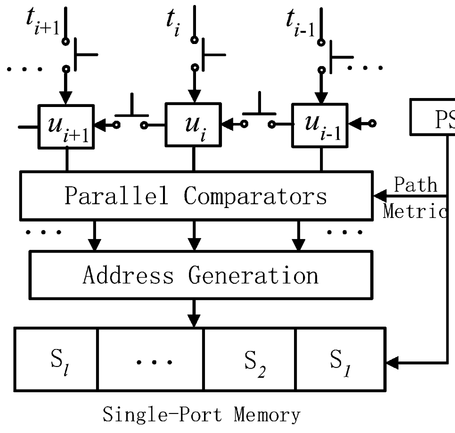

Fig. 6. Realization of the approximate sorting.

From the hardware implementation standpoint, this memory based distributed and approximate sorting has the following main advantages: 1) the computational complexity is much less than the strict sorting, leading to a great potential of higher throughput and significant power savings; 2) there is no data movement at all, which will help to further reduce the power consumption; 3) the distributed structure well matches the parallelism in the path extension and hence helps to realize high throughput.

It is clear that the choices of parallelism factor β, the sorting granularity factor l, and the total memory size directly affect the tradeoffs among the detection performance, throughput, and silicon area: A lower value of β will achieve better detection performance but reduces the achievable detection throughput. A larger size of memory will improve the detection performance at the cost of higher silicon area. A larger value of l may also improve the detection performance, given that the total memory size is big enough. In practice, we have to rely on extensive

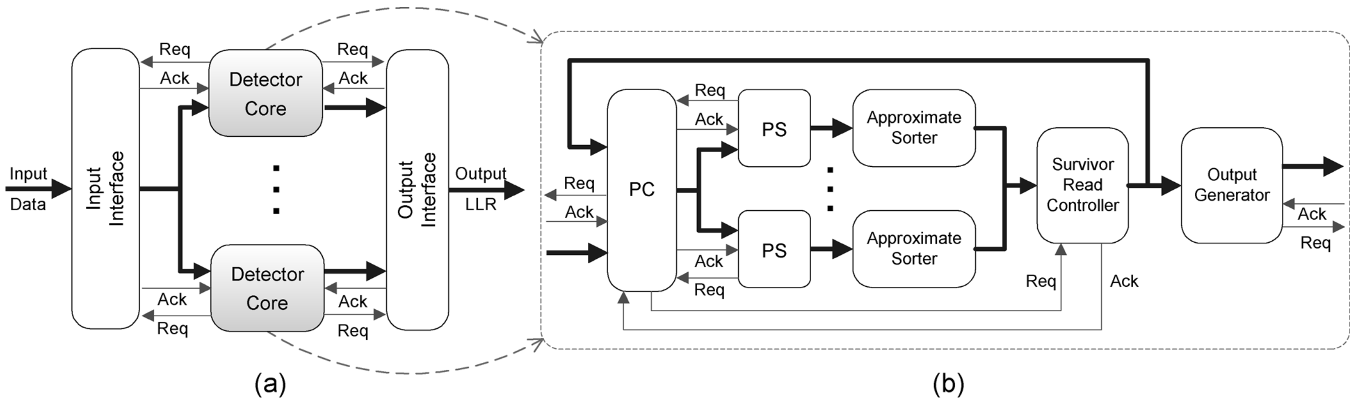

Fig. 7. Structure diagrams of (a) overall detector and (b) each recursive detector core.

simulations to choose the values of β, l, and the total memory size subject to the desirable trade-offs. As demonstrated in Section IV, with reasonable memory resource, relaxed K-best detectors using the distributed and approximate sorting can achieve the performance very close to the detector using sphere decoding algorithm with the same value of radius.

C. Detector Hardware Structure Design

Using the above two design techniques, i.e., distributed and approximate sorting and improved PSK enumeration, a relaxed K-best detector can be realized by integrating a certain number identical recursive detector cores, dependent on the target detection throughput. These identical cores operate in parallel and independently on different received signal vectors, as illustrated in Fig. 7(a). We note that the detector does not contain the precomputation blocks that perform channel matrix $\mathbf{H}$ decomposition and calculate $\hat{\mathbf{s}} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{y}$ and $G_i = \sum_{j=1}^{i} l_{i,j}\hat{s}_j$. Readers are referred to [18] for a detailed discussion on the implementation issues of the matrix decomposition functional block that computes the lower triangular matrix $\mathbf{L}$.

1) Structure of Individual Recursive Detector Core: The structure of each recursive detector core is shown in Fig. 7(b), which iterates N_t times to finish the detection of one received signal vector for a MIMO system with N_t transmit antennas. Each recursive detector core contains one PC block that is shared by several PS blocks, where the PC block performs the precalculation for the extension of each survivor path and each PS block carries out the path extension using the improved PSK enumeration method. Each PS block has its own approximate sorter.

During the path extension, for those *good* survivors that are close to the transmitted symbol vectors, their accumulated path metrics are small, which will lead to a bigger admissible region (or more modulation points to be explicitly examined) at present stage. It will take the PS block longer time to finish the path extension. On the other hand, for those *bad* survivors that are far away from the transmitted symbol vectors, less number of modulation points need to be explicitly examined, hence, PS block can finish the path extension more quickly. Therefore, the communication between PC block and PS blocks should be data-driven, as illustrated in Fig. 7: When one PS block is ready to process a new survivor path, it sends the PC block a *Req* signal. The PC block broadcasts the corresponding survivor path data to all the PS blocks through a shared bus and sends an *Ack* signal back to the PS block. After receiving the data from the bus, the PS block de-asserts the *Req* signal. The recursive detector core design is further described as follows.

Upon receiving one survivor path, each PC block performs the following operations:

1) calculates the $P_c = \sum_{j=1}^{i-1} l_{i,j}(\hat{s}_j - s_j) + l_{i,i}\hat{s}_i$, as aforementioned, which is shared among the succeeding path extensions;
2) identifies the boundary of the region containing all the valid circles (i.e., provides the indices of the inner and outer circles), as discussed in Section III-A; and
3) determines the starting point and initial direction for the zigzag search on each valid PSK circle.

Although the involved computation tends to be very complex, the PC block can be deeply pipelined to realize a high data processing throughput to feed those succeeding PS blocks.

Extending one survivor at a time, each PS block nonexhaustively examines the modulation points for path extension using the improved PSK enumeration method and sends the extended paths to its own approximate sorter. Fig. 8 shows the structure diagram of one PS block, which contains two main subblocks including modulation point selection (MPS) and path extension (PE). Each clock cycle, MPS selects and feeds one modulation point to PE for path extension. When PE detects a modulation point out of admissible region (i.e., the extended path metric $\Gamma^{(i)}$ is larger than the radius r^2), it will send a termination request to MPS so that MPS will not feed any other modulation points on the same circle to PE. Nevertheless, since PE should be deeply pipelined in order to achieve high throughput, if MPS keeps feeding the modulation points on the same circle to PE, the PE pipeline will be filled with modulation points out of admissible region when MPS receives the termination request. This may largely degrade the PE pipeline utilization efficiency and reduce the computation saving of PSK enumeration. To solve this problem, we make MPS feed the modulation points to PE alternatively among all the valid circles: As shown in Fig. 8, MPS maintains three tables, including: 1) valid circle table (VCT) that stores the indices of valid circles; 2) next modulation point table (NMPT) that stores the next modulation point to be extended on each valid circle; and 3) zigzag search

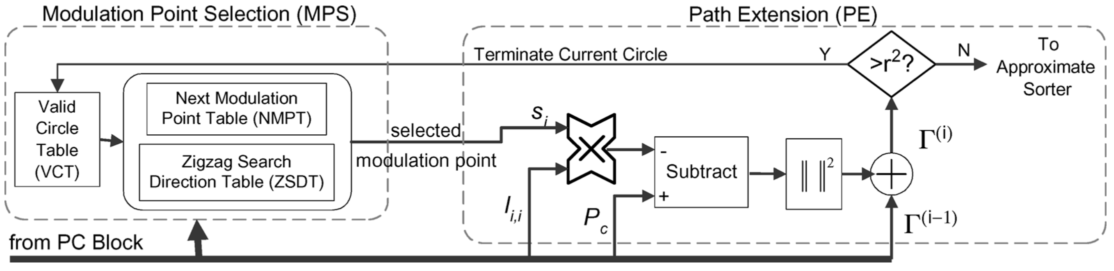
Modulation Point Selection (MPS)
Path Extension (PE)
Valid Circle Table (VCT)
Next Modulation Point Table (NMPT)
Zigzag Search Direction Table (ZSDT)
selected modulation point
Terminate Current Circle
Y
>r²?
N
To Approximate Sorter
Subtract
from PC Block

Fig. 8. Structure diagram of a PS block.

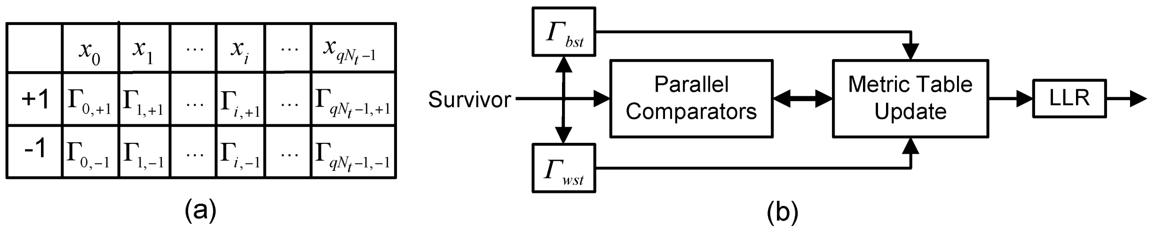

Fig. 9. Output generation design (a) metric table and (b) structure of output generator.

direction table (ZSDT) that stores the present zigzag search direction on each valid circle. All these tables are initialized by the data sent from the PC block. Each clock cycle, MPS feeds PE with one modulation point on the valid circle pointed by a valid circle pointer, updates the NMPT and ZSDT accordingly, and then make the valid circle pointer point to the next valid circle in VCT. If MPS receives a circle termination request from PE, it simply removes the corresponding circle index from VCT. This alternative modulation point fetching approach can largely improve the PE pipeline utilization efficiency for high order modulations.

The implementation of approximate sorter can be obtained in a straightforward manner from the discussion in Section III-B. 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. Controlled by the survivor read controller as shown in Fig. 7(b), approximate sorters send the survivors back to PC block as follows: We start with fetching one path at a time from segment S_1 in each single-port memory as survivor, alternatively among all the single-port memories, until we have fetched K paths or all the paths stored in the S_1 segments have been fetched. If latter happens, we move on to the segment S_2, and so on.

2) Structure of Output Generator: For hard-output detection, design of the output generator is straightforward, i.e., it simply searches for the survivor path with the best metric and outputs the qN_t hard decisions based on this best survivor. For 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 propose to use a scheme described as follows.

We maintain a metric table, as illustrated in Fig. 9(a), for the $qN_t \times 1$ binary vector, where each cell contains the best path metric, denoted by $\Gamma_{i,+1}$ or $\Gamma_{i,-1}$, among the survivors that has "+1" or "−1" on the corresponding bit position. Initially, the table entries are set as undefined. The path metric of each incoming survivor will compare against (and replace if its metric is better) one of the two metrics associated with each bit position. In order to process one survivor in one clock cycle, we may use parallel comparators as shown in Fig. 9(b). After all the last-depth survivors have been processed, the metrics in the table are used to calculate the L-values according to (2). However, it is possible that all the last-depth survivors agree on one bit position, i.e., the corresponding $\Gamma_{i,+1}$ or $\Gamma_{i,-1}$ remains as undefined, which means the L-value cannot be directly calculated. To solve this problem, as shown in Fig. 9(b), we keep the records of the best and worst metrics of the last-depth survivors, denoted as Γ_{bst} and Γ_{wst}, respectively, and use $|\Gamma_{wst} - \Gamma_{bst}|$ as the magnitude of the L-value for those bits.

IV. Performance Evaluation and VLSI Implementation

The signal detection performance of the proposed relaxed K-best detector is evaluated based on computer simulation with the following configuration: We consider LDPC-coded MIMO-OFDM system with 64-point FFT for 4×4 MIMO transmission with 64-QAM. Out of the 64 subcarriers, 48 are data carriers while the rest are used for pilots and virtual carriers, as defined in the IEEE 802.11a standard. Each subcarrier MIMO channel remains constant during transmission of a complete frame and is flat fading, i.e., all the entries in the MIMO channel matrix are independent random Gaussian variables. The LDPC code has a code rate of 1/2 and code length of 2304, and the LDPC code decoder performs up to 20 decoding iterations. There is no iteration between detector and decoder. For the definition of the MIMO channel SNR, we follow the one proposed in [10]: Let R denote the channel code rate ($R = 1$ for uncoded systems), SNR is defined as

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

where E_s denotes the average symbol energy of the QAM constellation. As pointed out earlier, the radius r^2 is calculated as

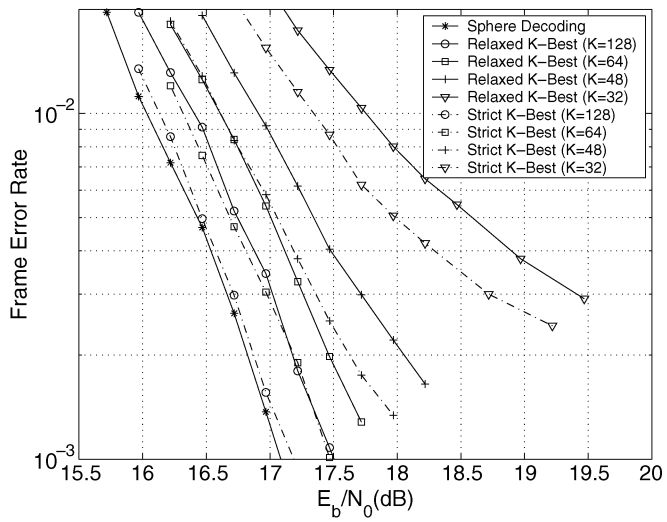

Fig. 10. Simulated FER performance for 4 × 4 MIMO 64-QAM.

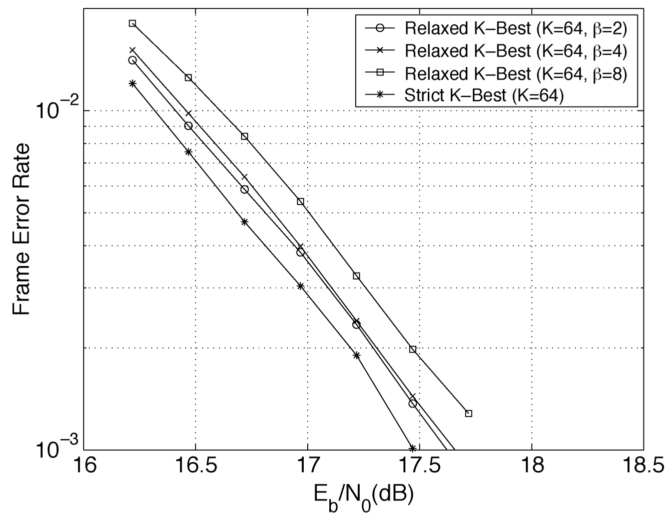

Fig. 11. Impact of parallelism factor β on performance.

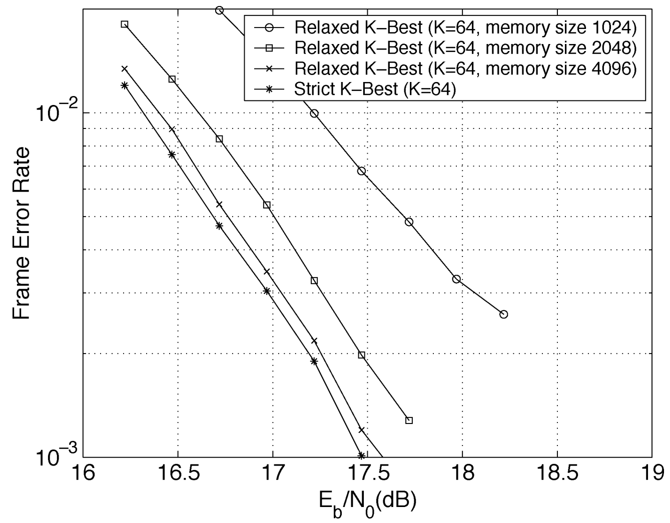

Fig. 12. Impact of total memory size on performance.

$2\alpha N_r \sigma^2$ [10], where $\alpha = 6$ is a predefined constant parameter and σ is the noise standard deviation.

For the purpose of comparison, we also carried out the simulations for a soft-output sphere detector which *exhaustively* examines all the paths that satisfy the sphere radius check in order to obtain the soft-output. The candidate list size is 128 and no early termination technique is used (i.e., the detector continually searches for better candidates even if the candidate list is full). The radius is calculated exactly the same as relaxed K-best detector described above. Fig. 10 shows the simulated frame error rate (FER) of sphere detector and the proposed relaxed K-best detectors with various K number, along with original strict K-best detectors to show the performance degradation due to the distributed and relaxed sorting. For relaxed K-best detectors, we choose $\beta = 8, l = 16$ and the total memory size is 2048 (i.e., totally 2048 paths can be stored in the memory). There are totally 16 memory blocks, among which eight memory blocks are used to store survivor paths from previous depth and the other eight blocks are used to store extended survivor paths of current depth. The size of each memory block is 128 and it is further divided into 16 segments. As shown in Fig. 10, when K is larger than or equal to 48, the relaxed K-best detection has about 0.3 dB degradation compared with the strict K-best detection and both of them have detection performance very close to the sphere detection. When K is 32, both relaxed K-best and strict K-best detectors suffer from big performance degradations. We note that the choice of ECC may have a big impact on the overall performance, e.g., if a much more powerful LDPC code (i.e., with much longer code length) is used, we may use less value of K to obtain the same performance.

Furthermore, we carried out simulations to evaluate the impact of different design parameters on the overall performance. Fig. 11 shows the impact of parallelism factor β on the performance. We change the value of β while keeping $l = 16$ and the total memory size to be 2048. Fig. 12 shows the impact of total memory size on the performance. We change the total size of memories while keeping $\beta = 8$ and $l = 16$.

To evaluate the hardware implementation feasibility, we designed one soft-output relaxed K-best recursive detector core with the following configuration: $\beta = 8$, $l = 16$, and the total memory size of 2048. The path metric is represented by 8 bits, and each entry in the matrix $\mathbf{L}$ and vector $\hat{\mathbf{s}}$ is represented by 15 and 16 bits, respectively. The design entry is Verilog HDL description, which has been synthesized using Chartered 0.13-μm standard cell and single-port SRAM libraries with 8 metal layers. Synopsys tools were used throughout the design hierarchy down to place and route. The post-layout simulation results have been verified against fixed-point C testbench. Fig. 13 shows the detector core layout, where the dark area in the layout is occupied by memory macros. The implementation metrics are summarized in Table I based on the post-layout power estimation and static timing analysis.

Due to the use of PSK enumeration method, the computational load and instantaneous throughput of the detector dynamically vary and depend on run-time channel conditions. Therefore, we carried out post-layout simulations to estimate the average detection throughput under different SNRs. Table II shows the estimated average detection throughput at four different SNRs for 4 × 4 MIMO 64-QAM when $K = 64$.

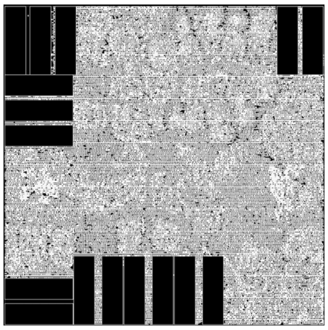
Fig. 13. Layout of one detector core.

TABLE I
IMPLEMENTATION METRICS OF THE DETECTOR CORE

V_{DD}	f_{clk}	Area	Logic Gate Count	Power
1.2 V	270 MHz	2.38 mm^2	0.28 M	94 mW

TABLE II
AVERAGE THROUGHPUT OF THE DETECTOR CORE

E_b/N_0 (dB)	17.7	17.2	16.7	16.2
Throughput (Mbps)	8.57	8.22	7.93	7.68

TABLE III
AREA AND POWER CONSUMPTION PARTITION BY FUNCTION BLOCKS

	PC and PS	Relaxed Sorter	Output Generator	Other Logic
Area	47.8%	37.4%	5.7%	9.1%
Power	65.4%	18.3%	8.2%	8.1%

The total area and power consumption can be partitioned by function blocks, as shown in Table III. From the data listed in the above tables, we may estimate that, in order to achieve above 100 Mb/s detection throughput, the detector will occupy about 31 mm^2 and consume about 1.2 W at the 0.13-μm CMOS technology node.

To further evaluate the area and power savings gained by using the proposed relaxed sorting approach, we also implemented bubble sorters that perform strict sorting and are used in the original strict K-best detectors [12], [14]. Bubble sorter is of particular interest because its regular dataflow is highly suitable for high-speed hardware implementations. In this work, we implemented the bubble sorters for the strict K-best detectors in two different approaches: 1) *Strict Design I*: All the data associated with each survivor path, i.e., the path symbols and path metric, are directly feed into the bubble sorter for strict sorting; 2) *Strict Design II*: Only the path metrics are feed into the bubble sorter for strict sorting, while the associated path symbols are stored in memory. The path metric in the bubble sorter is linked to its own path symbols in the memory through a memory address tag. In general, the second approach may reduce the energy consumption because of the reduced data movement in the bubble sorter. As shown in Fig. 10, the performance of a relaxed K-best detector with $K = 64$ is very close to that of a strict K-best detector with $K = 48$, thus, the bubble sorters are designed for $K = 48$. To further improve the throughput of the bubble sorter, we may introduce a sorting parallelism factor β, i.e., we implement β identical bubble sorters, each one follows one PS block and their outputs are further compared to obtain the globally sorted paths. Table IV summarized the area, power and throughput at 17.7 dB of different sorting realizations. In order to achieve the same throughput of relaxed K-best detector, both strict K-best designs will incur much higher (i.e., $2 \sim 3$ times) area and power consumption overhead.

TABLE IV
COMPARISON OF RELAXED K-BEST AND STRICT K-BEST SORTERS

	Area	Power	Throughput
Relaxed, $\beta = 8$	0.89 mm^2	17.2 mW	8.57 Mbps
Strict Design I, $\beta = 1$	0.54 mm^2	10.5 mW	1.43 Mbps
Strict Design I, $\beta = 6$	3.23 mm^2	63.2 mW	8.57 Mbps
Strict Design II, $\beta = 1$	0.37 mm^2	7.4 mW	1.43 Mbps
Strict Design II, $\beta = 6$	2.24 mm^2	44.1 mW	8.57 Mbps

V. CONCLUSION

This paper for the first time presents a nonlinear tree search MIMO signal detector design solution that can support hardware implementations for 4 $\times$ 4 MIMO transmission with 64-QAM. Belonging to the family of breadth-first tree search detectors, this design solution applies appropriate algorithm-level modifications to tackle the hardware implementation bottlenecks of the conventional breadth-first tree search MIMO detection. To demonstrate its effectiveness, we designed one soft-output detector for 4 $\times$ 4 MIMO transmission with 64-QAM at 0.13 μm CMOS technology node, which can realize very good detection performance and achieve high detection throughput with reasonable silicon area and power consumption.

REFERENCES

[1] "Special issues on MIMO systems and applications (I/II)," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, Apr./Jun. 2003.
[2] N. J. , Ed., "Special issue on gigabit wireless," *Proc. IEEE*, vol. 92, no. 2, pp. 195–197, Feb. 2004.
[3] D. G. *et al.*, "A 28.8 Mb/s 4 × 4 MIMO 3G CDMA receiver for frequency selective channels," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 320–330, Jan. 2005.
[4] ——, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1544–1552, Sep. 2004.
[5] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. URSI ISSSE*, 1998, pp. 295–300.
[6] H. Yao and G. W. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *Proc. IEEE Global Telecommun. Conf.*, 2002, pp. 424–428.
[7] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice-reduction," in *Proc. IEEE Int. Conf. Commun.*, 2004, pp. 798–802.
[8] D. L. Milliner and J. R. Barry, "A Lattice-reduction-aided soft detector for multiple-input multiple-output channels," presented at the IEEE Global Telecommun. Conf., San Francisco, CA, 2006.
[9] 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, Apr. 1985.
[10] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.
[11] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.
[12] 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 *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, pp. III-273–III-276.

[13] A. B. *et al.*, “VLSI implementation of MIMO detection using the sphere decoding algorithm,” *J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.
[14] Z. Guo and P. Nilsson, “VLSI architecture of the Schnorr-Euchner decoder for MIMO systems,” in *Proc. IEEE CAS Symp. Emerging Technol.*, 2004, pp. 65–68.
[15] ——, “Algorithm and implementation of the k-best sphere decoding for MIMO detection,” *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.
[16] S. Baro, J. Hagenauer, and M. Witzke, “Iterative detection of MIMO transmission using a list-sequential (LISS) detector,” in *Proc. IEEE Int. Conf. Commun.*, 2003, pp. 2653–2657.
[17] J. B. Anderson, “Limited search trellis decoding of convolutional codes,” *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 944–955, Sep. 1989.
[18] L. Davis, “Scaled and decoupled Cholesky and QR decompositions with application to spherical MIMO detection,” in *Proc. IEEE Wireless Commun. Netw.*, 2003, pp. 326–331.

Sizhong Chen (S’07) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1997 and 2000, respectively. Currently, he is pursuing the Ph.D. degree in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute, Troy, NY.

His research interests include VLSI architectures for communication systems and high performance/low power circuits design. Currently, he is working on MIMO system algorithm design and VLSI implementation.

Tong Zhang (S’98–M’02) received the B.S. and M.S. degrees in electrical engineering from the Xian Jiaotong University, Xian, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2002.

Currently, he is an Assistant Professor with the Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY. His current research interests include design of VLSI architectures and circuits for digital signal processing and communication systems, with the emphasis on error-correcting coding, MIMO signal processing, and asynchronous VLSI signal processing.

Yan Xin (S’00–M’03) received the B.E. degree in electronics engineering from Beijing University of Technology, Beijing, China, in 1992, and the M.Sc. degree in mathematics, the M.Sc. degree in electrical engineering, and the Ph.D. degree in electrical engineering from University of Minnesota, Minneapolis, in 1998, 2000, and 2003, respectively.

He is currently with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include space time coding, MIMO, network information theory, cognitive radio, and VLSI implementation.

Dr. Xin was a corecipient of the 2004 IEEE Marconi Paper Prize Award in wireless communications.