High-Throughput Soft-Output MIMO Detector Based on Path-Preserving Trellis-Search Algorithm

Yang Sun, *Student Member, IEEE*, and Joseph R. Cavallaro, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a novel path-preserving trellis-search (PPTS) algorithm and its high-speed VLSI architecture for soft-output multiple-input-multiple-output (MIMO) detection. We represent the search space of the MIMO signal with an unconstrained trellis, where each node in stage k of the trellis maps to a possible complex-valued symbol transmitted by antenna k. Based on the trellis model, we convert the soft-output MIMO detection problem into a multiple shortest paths problem subject to the constraint that every trellis node must be covered in this set of paths. The PPTS detector is guaranteed to have soft information for every possible symbol transmitted on every antenna so that the log-likelihood ratio (LLR) for each transmitted data bit can be more accurately formed. Simulation results show that the PPTS algorithm can achieve near-optimal error performance with a low search complexity. The PPTS algorithm is a hardware-friendly data-parallel algorithm because the search operations are evenly distributed among multiple trellis nodes for parallel processing. As a case study, we have designed and synthesized a fully-parallel systolic-array detector and two folded detectors for a 4×4 16-QAM system using a 1.08 V TSMC 65-nm CMOS technology. With a 1.18 mm^2 core area, the folded detector can achieve a throughput of 2.1 Gbps. With a 3.19 mm^2 core area, the fully-parallel systolic-array detector can achieve a throughput of 6.4 Gbps.**

***Index Terms*—Application-specific integrated circuit (ASIC), multiple-input-multiple-output (MIMO) detection, shortest path algorithm, soft-output MIMO detector, VLSI architecture.**

I. Introduction

MULTIPLE-INPUT-MULTIPLE-OUTPUT (MIMO) systems have great potential to increase spectral efficiency by transmitting independent data streams on multiple antennas. As an example, the Vertical Bell Laboratories Layered Space-Time (V-BLAST) system has been shown to achieve very high spectral efficiency [1]. MIMO technologies have been adopted in many new wireless standards such as 3GPP LTE/LTE-Advanced, IEEE 802.16e/802.16m WiMAX, and IEEE 802.11n/802.11ac WLAN. There is an increasing demand for Gbps wireless systems. For example, 3GPP LTE-Advanced, IEEE 802.16m WiMAX, IEEE 802.11ac WLAN, and WIGWAM [2] target for Gbps throughput with MIMO technology. Soft-output MIMO detection poses significant challenges to the MIMO receiver design as the computational complexity increases exponentially with the number of antennas. However, the optimal soft-decision detector, the maximum *a posteriori* (MAP) detector, will consume enormous computing power and require tremendous computational resources which makes it infeasible to be used in a practical MIMO receiver. As such, researchers are seeking efficient algorithms to reduce the MIMO detection complexity.

A. Related Work

Traditionally, the MIMO detection problem is usually tackled based on tree-search algorithms. The tree-search algorithms can be often categorized into the depth-first search algorithm and the breadth-first search algorithm. The sphere detection algorithm [3]–[7] is a depth-first tree-search algorithm to find the closest lattice point. To provide soft information for outer channel decoders, a modified version of the sphere detection algorithm, or soft sphere detection algorithm, is introduced in [8]. There are many implementations of sphere detectors, such as [9]–[18]. However, the sphere detector suffers from non-deterministic complexity and variable-time throughput. The sequential nature of the depth-first tree-search process significantly limits the throughput of the sphere detector especially when the SNR is low. The K-Best algorithm is a fixed-complexity algorithm based on the breadth-first tree-search algorithm [19]–[24]. But this algorithm tends to have a high sorting complexity to find and retain the best candidates, which limits the throughput of the detector especially when K is large. There are some other variations of the K-Best algorithm, which require less sorting than the regular K-best algorithm, e.g., [25]–[29], but it is still very difficult for the K-Best detectors to achieve 1+ Gbps throughput because of the high sorting complexity.

Generally, to make a soft decision for a bit x, a maximum-likelihood (ML) hypothesis and a counter-hypothesis of this bit are both required to form the LLR. A major problem for almost all the "conventional" tree-search algorithms is that the counter-hypotheses for certain bits are missing due to tree pruning. As a consequence of missing counter-hypotheses, the magnitude of the LLRs for certain bits cannot be determined, which will lead to performance degradation.

B. Proposed MIMO Detection Algorithm

To avoid the missing counter-hypothesis problem and to reduce the search complexity, we investigate high performance MIMO detection algorithms and high-speed VLSI architectures. In our earlier work [30], we have presented a 600 Mbps soft-output MIMO detector based on a greedy graph algorithm. In

Manuscript received September 01, 2010; revised February 16, 2011; accepted April 12, 2011. Date of publication May 27, 2011; date of current version June 01, 2012. This work was supported in part by Nokia, Nokia Siemens Networks (NSN), by Xilinx, and by US National Science Foundation (under Grant CCF-0541363, Grant CNS-0551692, Grant CNS-0619767, Grant EECS-0925942, and Grant CNS-0923479).

The authors are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: ysun@rice.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2011.2147811

1063-8210/$26.00 © 2011 IEEE

this paper, we significantly improve that algorithm and architecture, and we propose a high-performance detection scheme based on a path-preserving trellis-search (PPTS) algorithm.

We use an unconstrained trellis structure as an alternative to the tree structure to represent the search space of the MIMO signal. It should be noted that the terminology "trellis" in this paper has a different meaning than the typical "trellis" used in the optimal sequence detection algorithm in communication theory. We only use the data structure of the unconstrained trellis to represent the connections of the nodes. In the trellis graph, a *path* maps to a symbol *vector* so that the path weight is the Euclidian distance between the received signal and the product of the symbol vector and the channel matrix. Because each node maps to a possible transmitted symbol in a constellation so that a path weight is an indicator of the soft probability for nodes (or symbols) on this path. The PPTS algorithm is a multiple shortest paths algorithm on the condition that every trellis node must be included at least once in this set of paths so that the soft information for every possible symbol transmitted on every antenna is always available. When computing the LLR for every transmitted bit x, we can guarantee that a ML hypothesis and a counter-hypothesis of x are both available. We will discuss this important feature in Section III. On the other hand, the K-Best algorithm or the sphere algorithm may not preserve enough soft information for every bit x. Thus the missing counter-hypothesis problem may occur, which will lead to some performance loss.

From the implementation point of view, the advantage of the PPTS algorithm is that it is a very data-parallel algorithm because the searching operations at multiple trellis nodes can be performed simultaneously. Moreover, the local search complexity at each trellis node is kept very low to reduce the processing time. For very high data rate applications, we propose a pipelined, fully-parallel, systolic-array VLSI architecture. In this architecture, nodes in the same stage of the trellis are processed in parallel, and nodes in different stages of the trellis are processed in a pipelined manner. As a result, the systolic-array detector can process one MIMO symbol per clock cycle, leading to multiple Gbps throughput performance. For a lower data rate scalable system, we propose a folded architecture to save area.

The rest of this paper is organized as follows. Section II summarizes the MIMO system model. Section III introduces the PPTS algorithm. Section IV evaluates the error performance and analyzes the sorting complexity of the PPTS algorithm. Section V describes the VLSI architecture. Section VI summarizes the VLSI implementation results. Finally, Section VII concludes this paper.

II. System Model

We consider a spatial-multiplexing MIMO system with N_t transmit antennas and N_r receive antennas ($N_r \geq N_t$). The MIMO transmission can be modeled as

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

where $\mathbf{H}$ is a $N_r \times N_t$ complex matrix and is assumed to be known perfectly at the receiver, $\mathbf{s}$ is a $N_t \times 1$ transmit symbol vector $\mathbf{s} = [s_0\, s_1\, \ldots\, s_{N_t-1}]^T$, $\mathbf{y}$ is a $N_r \times 1$ received vector $\mathbf{y} = [y_0\, y_1\, \ldots\, y_{N_r-1}]^T$, and $\mathbf{n}$ is a vector of independent zero-mean complex Gaussian noise entries with variance σ^2 per real component. A real bit-level vector $\mathbf{x_k} = [x_{k,0}\, x_{k,1}\, \ldots\, x_{k,M_c-1}]^T$ is mapped to the complex symbol s_k, where the bth bit of $\mathbf{x_k}$ is denoted as $x_{k,b}$ and M_c is the number of bits per constellation point. Throughout this paper, the complex symbol s_k and its associated bit vector $\mathbf{x_k}$ will be used interchangeably.

The optimal MAP detector is to compute the log-likelihood ratio (LLR) value for the *a posteriori* probability (APP) of each transmitted bit. Assuming there is no *a priori* information for the transmitted bit, the LLR APP of each bit $x_{k,b}$ can be computed as [8]

$$\mathrm{LLR}(x_{k,b}) = \ln \frac{\sum_{\mathbf{s}:x_{k,b}=+1} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2\right)}{\sum_{\mathbf{s}:x_{k,b}=-1} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2\right)}. \tag{2}$$

With the Max-Log approximation [8], (2) is simplified to

$$\mathrm{LLR}(x_{k,b}) \approx \frac{1}{2\sigma^2}\left(\min_{\mathbf{s}:x_{k,b}=-1}\|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2 - \min_{\mathbf{s}:x_{k,b}=+1}\|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2\right). \tag{3}$$

Note that to form LLR for bit $x_{k,b}$, both the ML hypothesis and the counter-hypothesis of bit $x_{k,b}$ are required. Otherwise, the magnitude of the LLR will be undetermined. If a (sorted) QR decomposition of the channel matrix according to $\mathbf{H} = \mathbf{QR}$ is used, where $\mathbf{Q}$ and $\mathbf{R}$ refer to an $N_r \times N_t$ unitary matrix and an $N_t \times N_t$ upper triangular matrix, respectively, then (3) is changed to

$$\mathrm{LLR}(x_{k,b}) = \frac{1}{2\sigma^2}\left(\min_{\mathbf{s}:x_{k,b}=-1} d(\mathbf{s}) - \min_{\mathbf{s}:x_{k,b}=+1} d(\mathbf{s})\right) \tag{4}$$

where the Euclidean distance, $d(\mathbf{s})$, is defined as

$$d(\mathbf{s}) = \|\hat{\mathbf{y}} - \mathbf{R}\cdot\mathbf{s}\|^2 = \sum_{k=0}^{N_t-1} \left|(\hat{\mathbf{y}})_k - (\mathbf{R}\mathbf{s})_k\right|^2. \tag{5}$$

In the equation above, $\hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$, and $(\cdot)_k$ denotes the k-th element of a vector.

III. Path-Preserving Trellis-Search Algorithm

Computing the bit LLR in (4) requires searching for a ML hypothesis and a counter-hypothesis of this bit over a large search space. To reduce the search complexity and avoid the missing counter-hypothesis problem, we introduce a path-preserving trellis-search (PPTS) algorithm for soft MIMO detection.

A. Unconstrained Trellis Model

The Euclidean distance in (5) can be computed recursively. To visualize the recursion, we create a graph model, which resembles an unconstrained "trellis". As an example, Fig. 1 shows the trellis graph for the 4×4 4-QAM system. In this graph, nodes are ordered into N_t vertical slices or stages, where stage k corresponds to symbol s_k transmitted by antenna k. The trellis starts with a root node and ends with a dummy sink node. The stages are labeled in descending order. In each stage, there are

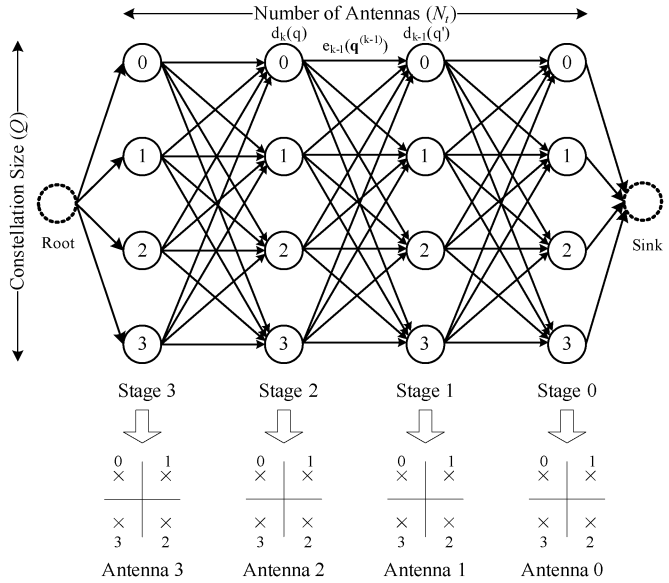


Fig. 1. Trellis graph for the 4 × 4 4-QAM system. Each stage of the trellis corresponds to a transmit antenna. There are $Q = 2^{M_c}$ nodes in each stage, where each node maps to a constellation point that belongs to a known alphabet.

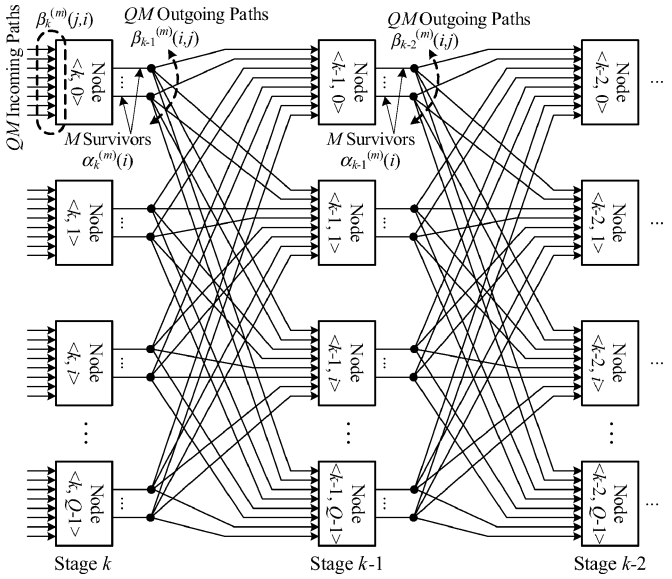


Fig. 2. Flow of the path reduction algorithm, where each node evaluates all its incoming paths and selects the best M paths.

$Q = 2^{M_c}$ different nodes, where each node maps to a constellation point that belongs to a known alphabet. Thus, each transmitted symbol *vector* is a *path* from root to sink. The trellis is fully connected, so there are Q^{N_t} number of different paths from root to sink. The nodes in stage k are denoted as $\langle k, q\rangle$, where $q = 0, 1, \ldots, Q-1$. The edge between nodes $\langle k, q\rangle$ and $\langle k-1, q'\rangle$ has a weight of $e_{k-1}(\mathbf{q}^{(k-1)})$

$$e_{k-1}\left(\mathbf{q}^{(k-1)}\right) = \left|\hat{y}_{k-1} - \sum_{j=k-1}^{N_T-1} R_{k-1,j} \cdot s_j\right|^2 \tag{6}$$

where $\mathbf{q}^{(k-1)}$ is the partial symbol vector $\mathbf{q}^{(k-1)} = [q_{k-1}\ q_k\ \ldots\ q_{N_t-1}]^T$, and s_j is the complex-valued symbol $s_j = \text{map}(q_j)$. Throughout this paper, the complex-valued symbol s_j and its associated real-valued number q_j will be used interchangeably. We define the path weight as the sum of the edge weights along this path. Then the weight of a path from root to sink is an Euclidean distance $||\hat{\mathbf{y}} - \mathbf{R}\cdot\mathbf{s}||^2$. Define a (partial) path metric d_k as the sum of the edge weights along this (partial) path. Then the path weight is computed recursively as

$$d_{k-1}(q') = d_k(q) + e_{k-1}\left(\mathbf{q}^{(k-1)}\right) \tag{7}$$

where $d_{N_T}(\cdot)$ is initialized to 0, and $d_0(\cdot)$ is the path weight (or Euclidean distance).

B. Multiple Shortest Paths Problem

We transform the soft MIMO detection problem into a multiple shortest paths problem. In the trellis graph, each trellis node $\langle k, q\rangle$ maps to a complex symbol s_k such that each path from root to sink maps to a symbol vector $\mathbf{s}$. A path weight is a measurement of the soft probability $(P(\mathbf{y}|\mathbf{s}))$ for nodes (symbols) on this path. To make a soft decision for every transmitted bit $x_{k,b}$, finding one shortest path is not enough. We want to find multiple paths which cover every node in the trellis graph. The multiple shortest paths problem is defined as follows. *For each node $\langle k, q\rangle$ in the trellis graph, find a shortest path from root to sink that must visit this node $\langle k, q\rangle$.* The corresponding shortest path weight is related to the symbol probability $(P(\mathbf{y}|s_k))$. If we can find such a conditional shortest path for each node in the trellis, we will then have one soft information value for every possible symbol transmitted on every antenna. As a result, we will have sufficient soft information values to avoid the missing counter-hypothesis problem. Thus, the LLR for every data bit can be formed accurately based on these soft information values.

C. Trellis Traversal Strategies

Because of the unconstrained trellis structure, there are Q^{N_t} different paths from root to sink that need to be evaluated. In order to reduce the search complexity, we propose a greedy algorithm that approximately solves the multiple shortest paths problem defined above. In this search algorithm, the trellis is pruned by removing the unlikely paths. However, we always preserve a predefined number of paths at each trellis node so that there is enough soft information to compute LLRs. We refer to it as the path-preserving trellis-search (PPTS) algorithm. It is a two-step algorithm which is summarized as follows.

1) Step 1—Path Reduction: The path reduction algorithm is used to prune the unlikely paths in the trellis by applying the M-algorithm [31] locally at each node. Fig. 2 illustrates the basic flow of the path reduction algorithm. In this algorithm, each node evaluates all its incoming paths and only preserves a predefined number (M) of paths that go through this node. We define the following notation to help explain the algorithm. Let $\beta_k^{(m)}(j,i)$ denote the QM incoming path candidates for node $\langle k, i\rangle$, and $\alpha_k^{(m)}(i)$ denote the M surviving path metrics selected by node $\langle k, i\rangle$. The indices i and j represent the trellis node number, where $0 \le i, j \le Q-1$. The superscript $^{(m)}$ represents the mth surviving path, where $0 \le m \le M-1$. In

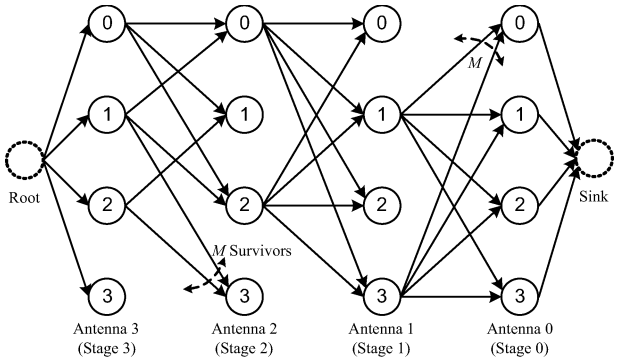


Fig. 3. Path reduction example for a 4 × 4 4-QAM trellis, where $M = 2$ incoming paths are preserved at each node.

Fig. 2, the stages of the trellis are labeled in descending order, starting from $N_t - 1$ and ending with 0. In stage k, each node $\langle k, i\rangle$ evaluates its QM incoming path candidates $\beta_k^{(m)}(j,i)$ and selects the best M paths from $\beta_k^{(m)}(j,i)$, where the mth best path metric is $\alpha_k^{(m)}(i)$. Note that in the edge weight function of $e_{k-1}^{(m)}(\mathbf{j}^{(k-1)})$, $\mathbf{j}$ denotes the partial symbol vector and the superscript $^{(m)}$ represents the edge weights corresponding to the mth surviving path. The α metrics are sorted so that $\alpha_k^{(0)}(i) < \alpha_k^{(1)}(i) < \ldots < \alpha_k^{(M-1)}(i)$. Next, each of the surviving paths is fully extended for the next stage so that there are QM outgoing paths leaving from each node $\langle k, i\rangle$, which are $\beta_{k-1}^{(m)}(i,j)$. This search process repeats for every stage of the trellis. The details of the path reduction algorithm are summarized in Algorithm 1.

Algorithm 1 Path Reduction Algorithm

0) **Initialization**: Set loop variable $k = N_t - 1$. For each node $\langle k, i\rangle$, initialize

$$\beta_k^{(m)}(j,i) = \begin{cases} \hat{y}_k - R_{k,k}s_k(i)|^2, & j,m = 0. \\ +\infty, & j,m \neq 0. \end{cases}$$

1) **Main Loop**:

1.a) Path Selection: For each node $\langle k, i\rangle$, select the best M paths $\alpha_k^{(m)}(i)$ from the QM path candidates $\beta_k^{(m)}(j,i)$.

1.b) Path Calculation:

for $(0 \le i \le Q-1)$

for $(0 \le m \le M-1)$

for $(0 \le j \le Q-1)$

$$\beta_{k-1}^{(m)}(i,j) = \alpha_k^{(m)}(i) + e_{k-1}^{(m)}(\mathbf{j}^{(k-1)}),$$

where $e_{k-1}^{(m)}(\mathbf{j}^{(k-1)})$ is the edge weight as defined in (6).

1.c) *Loop Update*: Set $k = k-1$. If $k \neq 0$, goto *1.a)*.

2) **Final Selection**: For each node $\langle 0, i\rangle$, select the best M paths $\alpha_0^{(m)}(i)$ from the QM path candidates $\beta_0^{(m)}(j,i)$.

As an example, Fig. 3 shows the result graph after applying the path reduction algorithm to a 4 × 4 4-QAM trellis, where each node preserves the best $M = 2$ incoming paths. After the path reduction, we can see that every node in the last stage, i.e., stage 0, has M shortest paths $(\alpha_0^{(m)}(i))$ that go through this node. Recall that each trellis node in stage k maps to a possible symbol s_k in a constellation. Thus, we have obtained a soft information value for every possible symbol s_0, the symbol transmitted by antenna 0. This is sufficient to guarantee that both the ML hypothesis and the counter-hypothesis in the Max-Log LLR calculation of (4) are available for every data bit $x_{0,b}$ transmitted by antenna 0. Then, the LLRs for data bits $x_{0,b}$, $b = 0, 1, \ldots, \log Q - 1$, can be computed as

$$\mathrm{LLR}(x_{0,b}) = \frac{1}{2\sigma^2}\left(\min_{i:b=-1}\alpha_k^{(m)}(i) - \min_{i:b=+1}\alpha_k^{(m)}(i)\right) \quad (8)$$

where $k, m = 0$.

However, aside from stage 0, not every node in stage k ($k \neq 0$) is included in a path from root to sink. For example, in Fig. 3, nodes $\langle 2,1\rangle$ and $\langle 2,3\rangle$ have uncompleted paths. Thus, we may not have enough soft information values to calculate the LLRs for data bits $x_{k,b}$ transmitted by antenna $k \neq 0$ because the counter-hypotheses for these bits can be missing. Although we can use LLR clipping [8] to saturate the LLR values, there will be some performance loss. To preserve enough soft information values for each data bit, we next introduce a path extension algorithm to find paths for every trellis node.

2) Step 2—Path Extension: To obtain soft information for every possible symbol s_k, we need to make sure every node in stage k is included in a path from root to sink. To extend node $\langle k, i\rangle$, we start to travel the trellis from this node and try to find the M most likely paths from this node to the sink node. This is achieved by extending the paths stage by stage, where the best M extended paths are selected in every stage. Fig. 4 shows an example data flow for the path extension for one node $\langle k, i\rangle$. Note that instead of waiting for the entire path reduction operation to finish, we will start the path extension operation for antenna k as soon as the path reduction algorithm has finished processing stage k of the trellis. In Fig. 4 for example, to detect antenna k, we first perform path reduction from stage $N_t - 1$ to stage k, and next we perform path extension from stage t $(t = k-1)$ to stage 0. Note that only one node's path extension process is shown in this figure. In fact, we will extend all the nodes in stage k simultaneously.

We define the following notation to help explain the algorithm. Let $\theta^{(m)}(k,i,t,j)$ denote the QM extended path candidates from node $\langle k, i\rangle$ to nodes $\langle t, j\rangle$, where $j = 0, 1, \ldots, Q-1$ and $m = 0, 1, \ldots, M-1$. Let $\gamma^{(m)}(k,i,t)$ denote the M surviving paths selected in stage t, where $m = 0, 1, \ldots, M-1$. To extend node $\langle k, i\rangle$, we first retrieve data $\beta_{k-1}^{(m)}(i,j)$ computed in the path reduction algorithm, and use it to initialize $\theta^{(m)}(k,i,t,j) = \beta_{k-1}^{(m)}(i,j)$, where $t = k-1$. Next, the best M extended paths $\gamma^{(m)}(k,i,t)$ are selected from $\theta^{(m)}(k,i,t,j)$. Then, $\gamma^{(m)}(k,i,t)$ are fully extended for the next stage to form $\theta^{(m)}(k,i,t-1,j)$. Again, the best M extended paths $\gamma^{(m)}(k,i,t-1)$ are selected from $\theta^{(m)}(k,i,t-1,j)$. This process repeats. Finally, $\gamma^{(m)}(k,i,0)$ are the result M extended

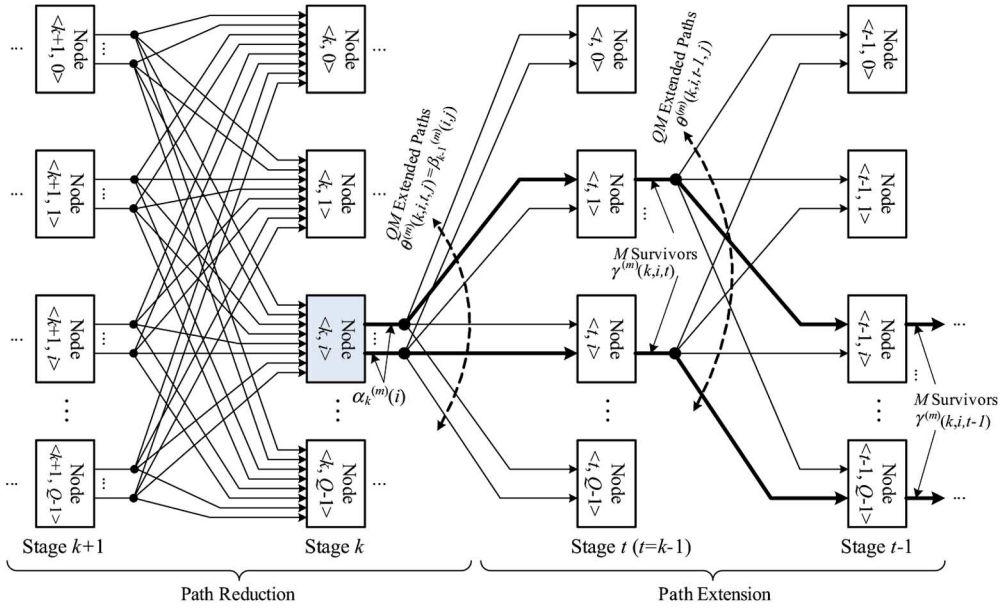


Fig. 4. Example data flow of the path extension algorithm for extending one node $\langle k, i\rangle$, where M paths are extended from this node to each of the following stages $(t, t-1, \ldots, 0$, where $t = k-1)$. All the nodes $\langle k, i\rangle$, $i = 0, 1, \ldots, Q-1$, can be extended in parallel.

paths from node $\langle k, i\rangle$ to the sink node. The path extension algorithm is summarized in Algorithm 2.

Algorithm 2 Path Extension Algorithm for Antenna k, $k = N_t - 1, N_t - 2, \ldots, 1$

0) **Initialization**: Set loop variable $t = k - 1$. For each node $\langle k, i\rangle$, initialize $\theta^{(m)}(k, i, t, j) = \beta_{k-1}^{(m)}(i, j)$.

1) **Main Loop**:

1.a) Path Selection: For each node $\langle k, i\rangle$, select the best M paths $\gamma^{(m)}(k, i, t)$ from the QM path candidates $\theta^{(m)}(k, i, t, j)$.

1.b) Path Calculation:

for $(0 \le i \le Q-1)$

for $(0 \le m \le M-1)$

for $(0 \le j \le Q-1)$

$\theta^{(m)}(k, i, t-1, j) = \gamma^{(m)}(k, i, t) + e_{t-1}^{(m)}(\mathbf{j}^{(t-1)})$,

where $e_{t-1}^{(m)}(\mathbf{j}^{(t-1)})$ is the edge weight as defined in (6).

1.c) Loop Update: Set $t = t - 1$. If $t \neq 0$ goto *1.a)*.

2) **Final Selection**: For each node $\langle k, i\rangle$, select the best M paths $\gamma^{(m)}(k, i, 0)$ from the QM path candidates $\theta^{(m)}(k, i, 0, j)$.

Fig. 5 shows an example to extend node $\langle 2, 1\rangle$ in a 4×4 4-QAM trellis. We can see that $M = 2$ paths are extended from this node to the sink node. It should be noted that nodes $\langle k, 0\rangle, \langle k, 1\rangle, \ldots, \langle k, Q-1\rangle$ can be extended in parallel since there is no data dependency between them. After the path extension is finished, every node in stage k will be included in a path from root to sink. Thus, we have obtained a soft information

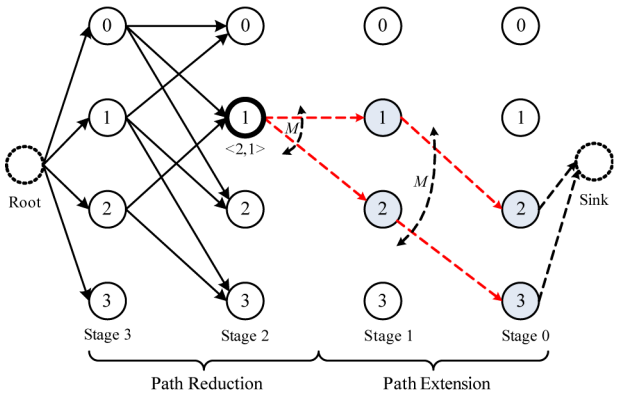


Fig. 5. Path extension example for one node $\langle 2, 1\rangle$, where $M = 2$ paths are extended from this node to the sink node.

value for every possible symbol s_k, the symbol transmitted by antenna k. This is sufficient to guarantee that both the ML hypothesis and the counter-hypothesis are available for every data bit $x_{k,b}$. Then, the LLRs for data bits transmitted by antenna $k \neq 0$ can be computed as

$$\mathrm{LLR}(x_{k,b}) = \frac{1}{2\sigma^2}\left(\min_{i:b=-1} \gamma^{(m)}(k, i, t) - \min_{i:b=+1} \gamma^{(m)}(k, i, t)\right) \tag{9}$$

where $t, m = 0$.

Note that although we keep M paths for each node $\langle k, i\rangle$ in every extension step, we only use the final smallest path weight for each node, i.e., $\gamma^{(m=0)}(k, i, t = 0)$, in (9) to compute the LLR. However, keeping multiple paths in the intermediate steps helps to improve the accuracy of the LLR values.

IV. Simulation Result and Complexity Analysis

In this section, we evaluate the error performance of the proposed PPTS detector through computer simulations. The

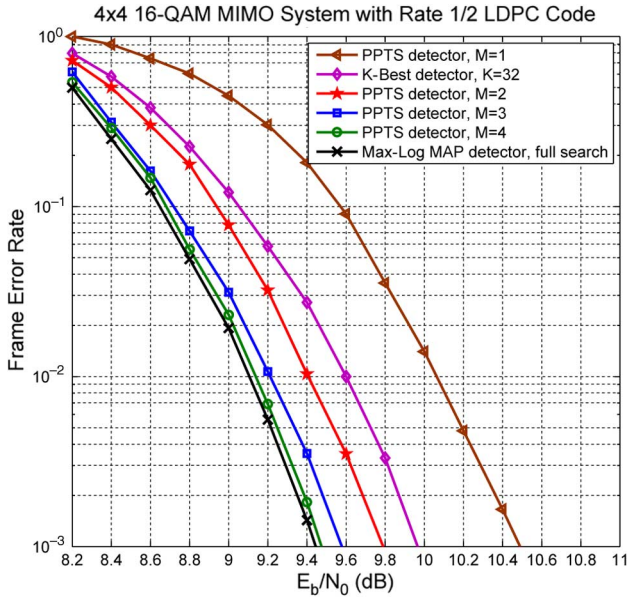


Fig. 6. Error performance of a coded 4 × 4 16-QAM MIMO system using the PPTS detection algorithm with different M values. Outer channel code is a WiMax LDPC code with rate 1/2 and length 2304.

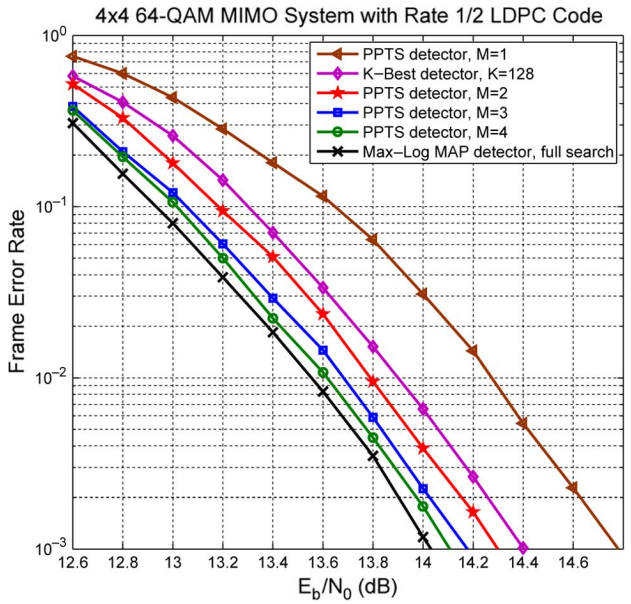


Fig. 7. Error performance of a coded 4 × 4 64-QAM MIMO system using the PPTS detection algorithm with different M values. Outer channel code is a WiMax LDPC code with rate 1/2 and length 2304.

floating-point simulations are carried out for 4 × 4 16-QAM and 4 × 4 64-QAM systems where the channel matrices are assumed to have independent random Gaussian distributions. A sorted QR decomposition of the channel matrix is used. The soft-output of the detector is fed to a length 2304, rate 1/2 WiMax layered LDPC decoder, which performs up to 20 LDPC inner iterations. Figs. 6 and 7 show the frame error rate (FER) performance of the PPTS detectors for different M values. As a reference, we also show the error performance of a Max-Log MAP detector with exhaustive search criterion, and a soft K-Best detector with $K = 2Q$. In the error performance comparison, the Max-Log MAP detector with exhaustive search criterion is considered as the baseline reference.

For a 4 × 4 16-QAM system, when $M = 1$, the PPTS detector shows about 1 dB performance loss at FER 10^{-3} compared to the baseline reference. When $M = 2$, the PPTS detector shows about 0.35 dB performance degradation. When $M = 3$, the PPTS detector shows only 0.15 dB performance degradation. When $M = 4$, the PPTS detector achieves a performance almost the same as the baseline reference. Compared to the K-Best detector with $K = 32$, the PPTS detectors with $M = 2, 3, 4$ significantly outperform the K-Best detector. For a 4 × 4 64-QAM system, when $M = 1$, the PPTS detector shows about 0.75 dB performance loss at FER 10^{-3} compared to the baseline reference. When $M = 2$, the PPTS detector shows about 0.3 dB performance degradation. When $M = 3, 4$, the PPTS detector achieves a performance that is very close to the baseline reference. Compared to the K-Best detector with $K = 128$, the PPTS detectors with $M = 2, 3, 4$ outperform the K-Best detector.

Now we discuss the complexity of the PPTS algorithm. The sorting complexity and the partial Euclidian distance (PED) computation complexity are two major contributors to the overall complexity. In terms of the sorting complexity, the PPTS detector need to carry out a (s, t) sorting operation: find the smallest s values out of t candidates. Generally, to find the s smallest values from t candidates requires at least $t - s + \sum_{t+1-s<j\leq t} \lceil \log j \rceil$ pair-wise comparisons [32]. This bound is only achievable for $s = 1, 2$. In the proposed PPTS algorithm, we employ Q concurrent small sorters (M, QM) at each stage of the trellis. Because the PPTS detector can achieve good performance with a small M value, e.g. $M = 2$, such a small sorting operation can be done quickly and efficiently. For example, when using $M = 2$ for a 16-QAM system, each sorter needs to perform a (2,32) sorting operation, which only needs to perform 35 pairwise comparisons. Therefore, the sorting complexity of the PPTS algorithm is significantly lower than the traditional K-best algorithm, which needs to perform a larger global sorting operation at each level of the tree. The proposed PPTS algorithm can achieve a near-optimal detection performance with a very low sorting complexity.

In terms of the PED computation complexity, the PPTS detector needs to evaluate the number Q^2M PEDs in each stage of the trellis. To have a fair comparison between a PPTS detector and a K-Best detector, we need to consider the error performance when choosing parameters K and M. From the performance comparison in Fig. 6 and Fig. 7, we can see that the PPTS detector with $M = 2$ outperforms the K-Best detector with $K = 2Q$ for both a 4 × 4 16-QAM system and a 4 × 4 64-QAM system. For a K-best detector with $K = 2Q$, the number of PEDs that need to be computed at each level of the tree is $2Q^2$. Thus, the PED complexities of PPTS detectors and K-Best detectors will both likely grow quadratically with the constellation size Q. However, if we compare the sorting complexity, the advantage of the PPTS detector will be more significant for large size constellations because sorting is performed in a distributed manner as opposed to a larger global sorting required by the K-Best detector.

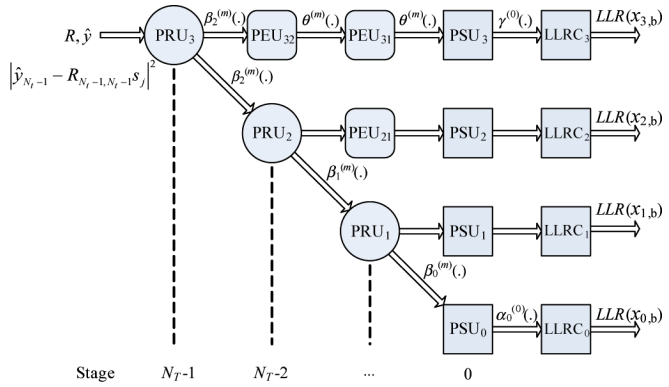


Fig. 8. Pipelined fully-parallel "systolic" architecture for the PPTS detector, where each PRU/PEU/PSU is a cluster of Q path reduction/path extension/path selection processors.

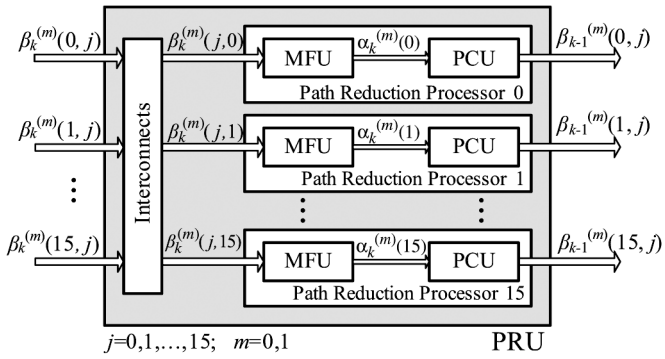


Fig. 9. Block diagram for the PRU, which contains $Q = 16$ path reduction processors.

The PPTS algorithm is a highly parallel algorithm such that the number Q of trellis nodes in a stage can be processed in parallel, which leads to a very high throughput. This is a very important feature that the detector will not be a bottleneck in a system given that the computation resources are not constrained. Although the algorithm has a quadratic complexity with the constellation size Q, we can design a scalable architecture that can be tailored for different throughput requirements. To achieve the highest throughput, we can develop a fully parallel architecture where each of the trellis nodes has a dedicated node processor. If the computation resources are limited, as a balanced tradeoff, we can develop a partial-parallel architecture where a certain number of the trellis nodes share a common node processor to save area while still maintaining the throughput requirement. Nevertheless, with the advance of VLSI technology, the PPTS algorithm has a great potential to be applied in a practical MIMO system.

V. VLSI Architecture

In this section, we describe VLSI architectures for the proposed PPTS detector. We introduce a fully-parallel "systolic" architecture to achieve the maximum throughput performance, and a "folded" architecture to reduce area for lower throughput applications. For the sake of clarity, we describe a PPTS detector architecture with $M = 2$ for the 4×4 16-QAM system. It should be noted that the architecture described can be easily scaled for other values of M and other MIMO configurations.

A. Fully-Parallel Systolic Architecture

Fig. 8 shows the fully-parallel "systolic" architecture for a $N_t = 4$ antenna system. This architecture is fully pipelined so that it can process one MIMO symbol in every clock cycle. In this architecture, the main processing elements include three path reduction units (PRUs), three path extension units (PEUs), four path selection units (PSUs), and four LLR calculation (LLRC) units. The detailed structures of these processing elements will be described in the following subsections.

In Fig. 8, three PRUs (PRU_{3-1}) and one PSU (PSU_0) are employed to implement the path reduction algorithm. The main diagonal of the systolic array is related to the path reduction data flow shown in Fig. 2. The PRU implements one main iteration loop of Algorithm 1 by employing Q path reduction processors to simultaneously process Q nodes in a certain stage (cf. Fig. 2). PSU_0 implements the *final selection* step of Algorithm 1 by using Q search units. The data flow for the path reduction is as follows. First, PRU_3 receives $\mathbf{R}$, $\hat{\mathbf{y}}$, and the precomputed $|\hat{y}_3 - R_{3,3}s_j|^2$, and it computes all the path candidates $\beta_2^{(m)}(i,j)$ in parallel, which are fed to the next PRU, i.e., PRU_2. Then, PRU_2 computes $\beta_1^{(m)}(i,j)$, which are fed to PRU_1, and so forth. Finally, PSU_0 receives $\beta_0^{(m)}(i,j)$ from PRU_1 and computes $\alpha_0^{(0)}(i)$, which are fed to LLRC_0 to compute $\mathrm{LLR}(x_{0,b})$ based on (8).

In Fig. 8, three PEUs and three PSUs (PSU_{3-1}) are employed to implement the path extension algorithm. Each row (but the last) of the systolic array is related to the path extension data flow shown in Fig. 4. The PEU implements one main iteration loop of Algorithm 2 by employing Q path extension processors to simultaneously extend Q nodes in a certain stage (cf. Fig. 4). The PSU is used to implement the *final selection* step of Algorithm 2. The data flow for the path extension is as follows. To detect antenna $k \geq 1$, $k-1$ number of the PEUs and 1 PSU are used. Let $t = k-1$. First, $\mathrm{PEU}_{k,t}$ receives $\beta_{k-1}^{(m)}(i,j)$ from PRU_k and it computes $\theta^{(m)}(k,i,t-1,j)$, which are fed to $\mathrm{PEU}_{k,t-1}$. Next, $\mathrm{PEU}_{k,t-1}$ computes $\theta^{(m)}(k,i,t-2,j)$, which are fed to $\mathrm{PEU}_{k,t-2}$, and so forth. Finally, PSU_k receives $\theta^{(m)}(k,i,0,j)$ from $\mathrm{PEU}_{k,1}$ and computes $\gamma^{(0)}(k,i,0)$, which are fed to LLRC_k to compute $LLR(x_{k,b})$ based on (9). Note that to detect antenna 1, only one PSU (PSU_1) is required.

B. PRU

The structure of the PRU is shown in Fig. 9. The PRU is used to implement the path reduction algorithm (cf. Algorithm 1: main loop). The PRU employs $Q = 16$ path reduction processors to process all the Q nodes in a certain stage in parallel. Each path reduction processor contains one minimum (min) finder unit (MFU) and one path calculation unit (PCU), where the MFU is used to select the best M paths $\alpha_k^{(m)}(i)$ from the QM incoming path candidates $\beta_k^{(m)}(j,i)$ (cf. Algorithm 1-1.a), and the PCU is used to compute the QM new extended path candidates $\beta_{k-1}^{(m)}(i,j)$ (cf. Algorithm 1-1.b).

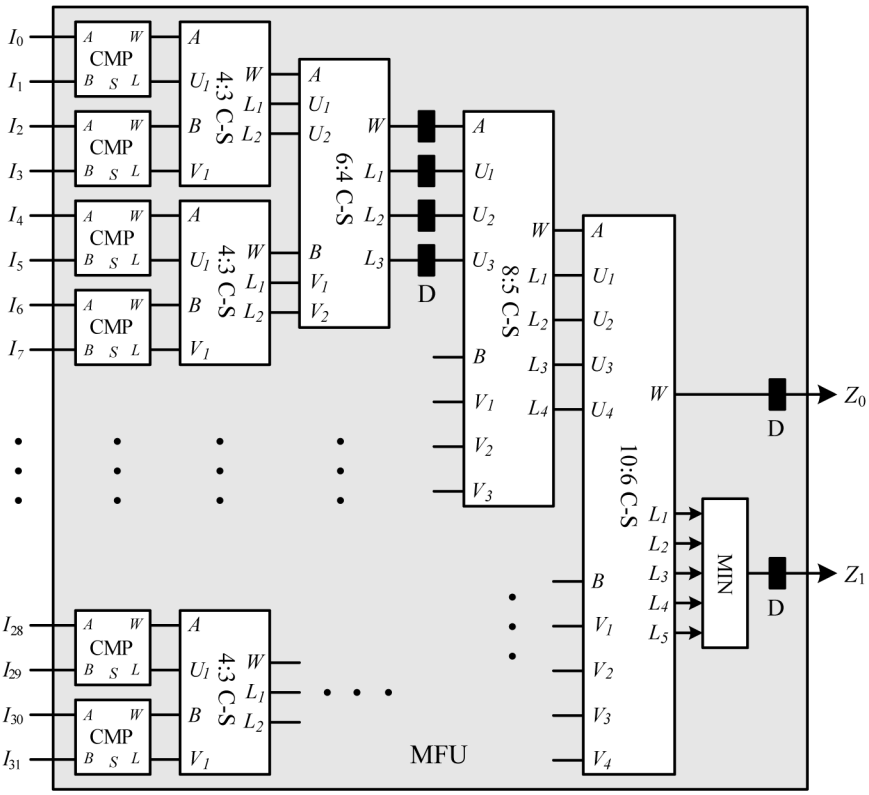


Fig. 10. Block diagram for the MFU, which uses 16 CMP units, 15 variable size C-S (compare and select) units, and 1 MIN unit to implement the (2,32) sorting.

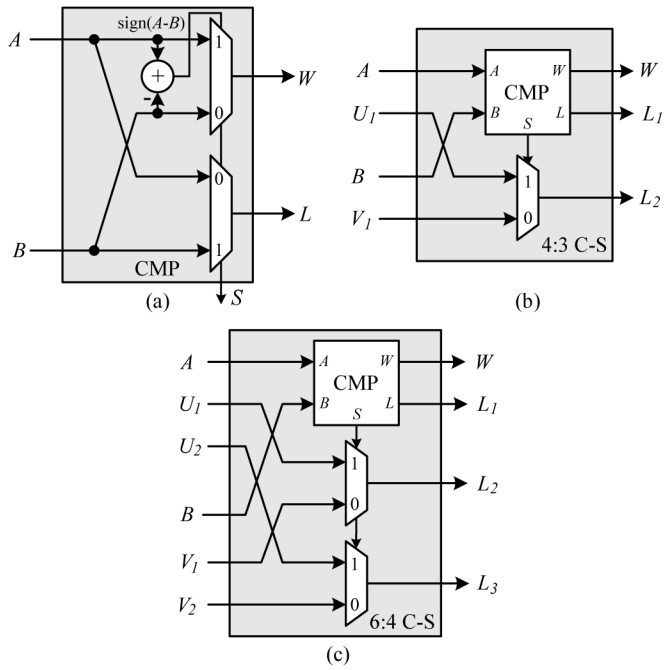


Fig. 11. Block diagram for the CMP unit, the 4:3 C-S unit, and the 6:4 C-S unit.

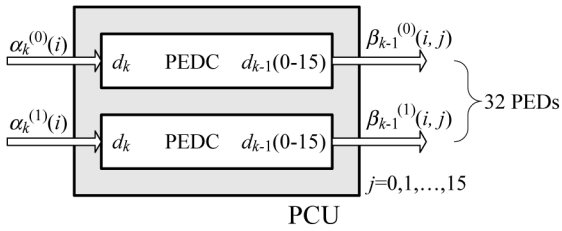


Fig. 12. Block diagram for the PCU, which employs $Q = 2$ PEDC units.

1) MFU: The MFU is used to select the best $M = 2$ paths from $QM = 32$ path candidates. Fig. 10 shows the block diagram for the MFU unit which finds the minimum value Z_0 and the second minimum value Z_1 from its 32 data inputs (I_0 to I_{31}).

The MFU is comprised of 16 CMP (comparison) units, 15 variable size ($p : (p/2+1)$) C-S (compare and select) units, and one MIN unit. The structure of the CMP unit is shown in Fig. 11(a). The CMP unit compares two data inputs A and B, and outputs the smaller one (or the "winner"): $W = \min(A, B)$, and the larger one (or the "loser"): $L = \max(A, B)$, and the sign: $S = \text{sign}(A - B)$. The variable size $p : (p/2 + 1)$ C-S unit has p inputs $(A, U_1, U_2, \ldots, U_{p/2-1}, B, V_1, V_2, \ldots, V_{p/2-1})$ and $p/2+1$ outputs $(W, L_1, L_2, \ldots, L_{p/2})$. The different values of p for the variable size C-S unit are $4, 6, 8, \ldots, 2\log(QM)$. Output W of the C-S unit is the smallest data among all the p inputs. Outputs $L_1, L_2, \ldots, L_{p/2}$ of the C-S unit are $p/2$ candidates for the second smallest data among all the p inputs. Fig. 11(b) and (c) show the structure of the 4:3 C-S unit and the 6:4 C-S unit. The structure of the larger size C-S units, e.g., 8:5 C-S unit and 10:6 C-S unit, are omitted in this paper because they have very similar structure as the 6:4 C-S unit.

The MFU functions as follows. As shown in Fig. 10, the MFU takes $QM = 32$ data inputs and feeds them to 16 CMP units,

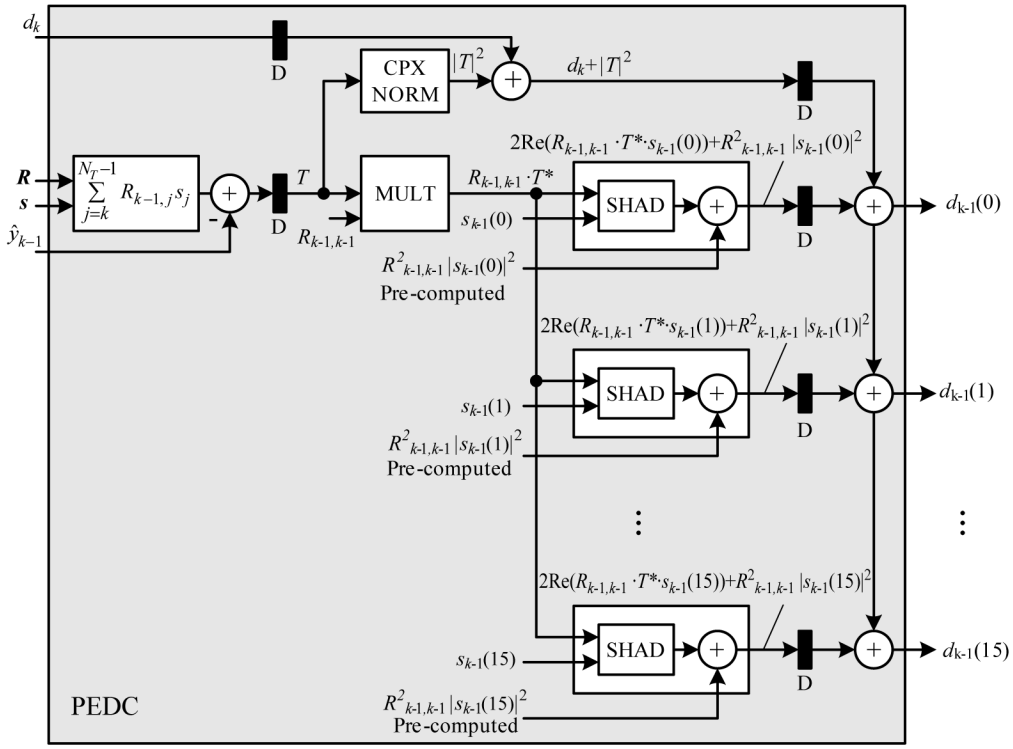


Fig. 13. Block diagram for the PEDC unit, which computes 16 PEDs in parallel.

where each CMP unit generates the winner and the loser of its two data inputs. The connection of the computational blocks in the MFU resembles a tree-like structure. Every two CMP units are connected to one 4:3 C-S unit, where the outputs of the 4:3 C-S unit are the winner (W) of its four data inputs, and two candidates (L_1, L_2) for the second winner. Every two 4:3 C-S units are connected to one 6:4 C-S unit, where the outputs of the 6:4 C-S unit are the smallest data (W) among its 6 data inputs, and three candidates (L_1, L_2, L_3) for the second smallest data. Similarly, every two 6:4 C-S units are connected to one 8:5 C-S unit, and two 8:5 C-S units are connected to a final 10:6 C-S unit. Finally, output W of the 10:6 C-S unit is the smallest data (Z_0) among the 32 data $(I_0, I_1, \ldots, I_{31})$. Outputs $L_1, L_2, \ldots, L_5$ of the 10:6 C-S unit are the five candidates for the second smallest data among the 32 data inputs. A MIN unit is used to generate the second smallest data Z_1 $(Z_1 = \min(L_1, L_2, \ldots, L_5))$.

2) PCU: Fig. 12 shows the PCU architecture which employs $M = 2$ partial Euclidean distance calculation (PEDC) units to compute $QM = 32$ path metrics in parallel. The partial Euclidean distance (PED) d_{k-1} is computed recursively as

$$d_{k-1} = d_k + e_{k-1}. \tag{10}$$

The metric increment e_{k-1} [cf. (6)] is computed as follows:

$$e_{k-1} = |T + R_{k-1,k-1} \cdot s_{k-1}|^2 \tag{11}$$

where

$$T = \sum_{j=k}^{N_T-1} R_{k-1,j} \cdot s_j - \hat{y}_{k-1}. \tag{12}$$

For a given PED d_k, we need to compute $Q = 16$ new PEDs d_{k-1}. Instead of computing each new PED separately, we can compute Q new PEDs in a group by knowing that symbol s_{k-1} is drawn from a known alphabet: $s_{k-1} \in \{\pm 1 \pm j, \pm 1 \pm 3j, \pm 3 \pm j, \pm 3 \pm 3j\}$, and $R_{k-1,k-1}$ is a real value if using a certain QR decomposition method, e.g., Gram-Schmidt QR decomposition [33]. Let $s_{k-1}(q)$, $q = 0, 1, \ldots, Q-1$, denote the complex symbol for the qth constellation point in the alphabet. Then (11) is re-expressed as

$$\begin{aligned} e_{k-1}(q) = {} & |T|^2 + R_{k-1,k-1}^2 |s_{k-1}(q)|^2 \\ & + 2\mathrm{Re}\left((R_{k-1,k-1} \cdot T^*) s_{k-1}(q)\right). \end{aligned} \tag{13}$$

We precompute $R_{k-1,k-1}^2 |s_{k-1}(q)|^2$ for different q and save them in registers. Fig. 13 shows the architecture for the PEDC unit, which computes $Q = 16$ PEDs in parallel. In this architecture, a shift and add (SHAD) unit is used to implement the constant multiplication $A \cdot s_{k-1}$, a multiplier (MULT) is used to implement $R_{k-1,k-1} \cdot T^*$, and a CPX (complex) NORM unit is used to compute the L2 norm $(|T|^2)$ of the complex signal T.

C. PEU

The PEU implements the path extension algorithm (cf. Algorithm 2: main loop). The PEU has a very similar architecture to the PRU. Fig. 14 shows the block diagram for the PEU, which employs $Q = 16$ path extension processors to extend Q nodes in a certain stage in parallel. Each path extension processor contains one MFU and one PCU, where the MFU is used to select the best M paths $\gamma^{(m)}(k, i, t)$ from QM path candidates $\theta^{(m)}(k, i, t, j)$ (cf. Algorithm 2-1.a), and the PCU is used to calculate the QM new extended path candidates $\theta^{(m)}(k, i, t-1, j)$ (cf. Algorithm 2-1.b)

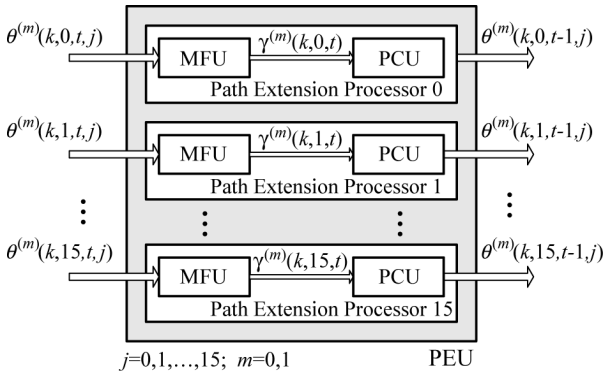


Fig. 14. Block diagram for the PEU, which contains $Q = 16$ path extension processors.

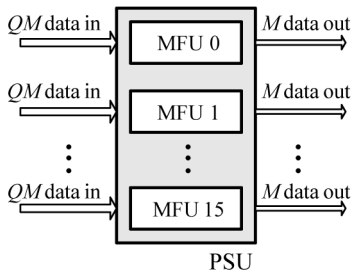


Fig. 15. Block diagram for the PSU, which contains $Q = 16$ MFUs.

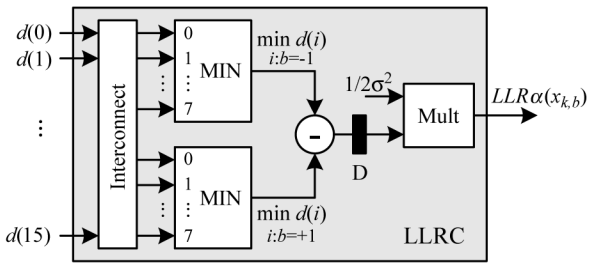


Fig. 16. Block diagram for the LLRC unit.

D. PSU

The PSU implements the *final selection* step in Algorithm 1 or Algorithm 2. As shown in Fig. 15, the PSU contains only Q MFUs to realize Q concurrent sorting (M, QM).

E. LLR Computation Unit (LLRC)

The LLRC is used to compute LLRs based on (8) or (9). Fig. 16 shows the block diagram for the LLRC unit. To compute $\log_2(Q) = 4$ LLRs for antenna k in parallel, we need four sets of hardware blocks shown in Fig. 16 to compute $\mathrm{LLR}(x_{k,b})$, $b = 0, 1, \ldots, \log Q - 1$, for our example 16-QAM system. It should be noted that the multiplier in Fig. 16 may not be required if the outer channel decoder uses a linear decoding algorithm such as the Min-Sum algorithm [34] in LDPC decoding or the Max-Log-MAP algorithm [35] in Turbo decoding. In that case, the multiplier can be replaced by a simpler normalizer.

F. Throughput Performance of the Systolic Architecture

The proposed systolic MIMO detector architecture (cf. Fig. 8) can provide very high throughput performance. This architecture is fully pipelined so that it can process one MIMO symbol in every clock cycle. Generally, if we let the clock frequency be fclk MHz, then the throughput (Mbps) for a $N_t \times N_r$ Q-QAM system can be expressed as

$$\mathrm{Throughput\_Systolic} = N_t \cdot \log_2 Q \cdot \mathrm{fclk}. \tag{14}$$

As an example, assuming a system clock of 400 MHz, the systolic architecture can provide a throughput of 6.4 Gbps for a 4 $\times$ 4 16-QAM system.

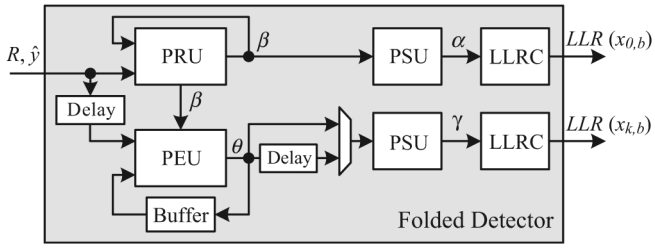


Fig. 17. Folded architecture for the PPTS detector.

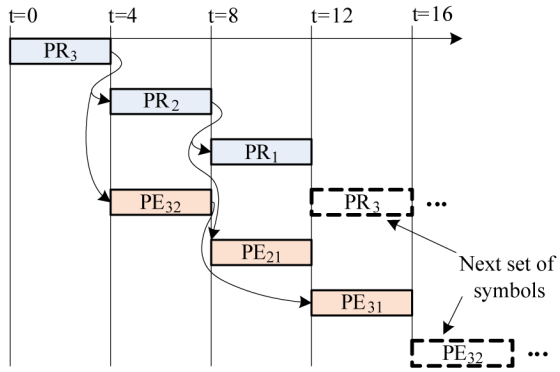


Fig. 18. Detection timing diagram for a four antenna system using the folded architecture.

G. Folded Architecture

For system applications that may require less throughput, we can fold the fully-parallel systolic architecture to reduce the parallelism and hence the hardware complexity. Fig. 17 shows the folded architecture where only one PRU and one PEU are instantiated to save area. Note that the PRU/PEU is the most area-consuming block in the PPTS detector.

Because we only have one PRU and one PEU, we need to schedule them sequentially. Fig. 18 illustrates the detection timing diagram using the folded architecture for a four antenna system. In this diagram, the PRU is scheduled to run the path reduction (PR) operations from $t = 0$ to $t = 11$, and the PEU is scheduled to run the path extension (PE) operations from $t = 4$ to $t = 15$. Note that the subscripts of the PRs and PEs in this diagram have the same meaning as that in Fig. 8. For simplicity, the final path selection operations (executed in PSU) and the LLR calculation operations are omitted in this diagram. Furthermore, as the pipeline stages for the PRU and PEU are 4 clock cycles, we can feed four back-to-back MIMO symbols in four consecutive cycles, e.g at $t, t+1, t+2, t+3$ to fully utilize the hardware. We can feed the next four back-to-back MIMO

TABLE I
FIXED POINT DESIGN PARAMETERS FOR THE 4 × 4 16-QAM SYSTEM

Scaled R	Received $\hat{y}$	PED	LLR
$Q1.9$ signed	$Q4.6$ signed	$Q4.6$ unsigned	$Q4.2$ signed

symbols at $t+12, t+13, t+14, t+15$ into the pipeline, and so forth. The throughput of the folded architecture for a four antenna system is given as

$$\text{Throughput\_folded\_4ant} = \frac{4}{3}\log_2 Q \cdot \text{fclk}. \quad (15)$$

For a larger MIMO system with $N_t \geq 4$ transmit antennas, if we still use one PRU and one PEU, the throughput for a $N_t \geq 4$ antenna system is estimated as

$$\text{Throughput\_folded\_N} = \frac{2N_t}{(N_t-1)(N_t-2)}\log_2 Q \cdot \text{fclk}. \quad (16)$$

Note that for larger MIMO systems ($N_t > 4$), the throughput is limited by the number of path extension operations. However, we can employ more than one PEU in the folded architecture to match with the processing speed of the PRU.

VI. VLSI IMPLEMENTATION RESULT

A. Fixed-Point Detector Design for 4 × 4 16-QAM System

In a 4 × 4 16-QAM MIMO transmission, typically the QAM symbol s_k is scaled by $1/\sqrt{10N_t} = 1/\sqrt{40}$ in the transmitter for the transmitted symbol to have unit energy. In the PPTS MIMO detector, instead of using the scaled s_k signal, we scale each element in the $\mathbf{R}$ matrix [cf. (5)] by $1/\sqrt{10M_t} = 1/\sqrt{40}$ and use the original QAM symbol s_k in the computation. We use the notation $Q[QI].[QF]$ to represent a fixed point number with QI number of integer bits and QF number of fractional bits so that the total word length is $QI+QF$. Table I summarizes the fixed point design parameters for the scaled R, received $\hat{y}$, PED, and LLR, where the PED is rounded to 10 bits between computational blocks. Based on the same simulation parameters as described in Section IV, this fixed-point detector has shown about 0.1 dB performance loss compared to the floating-point detector. Fig. 19 shows the fixed-point simulation result for the proposed PPTS detector.

B. ASIC Synthesis Result and Architecture Comparison

As a proof of concept, we have implemented a systolic PPTS detector with $M = 2$, and two folded PPTS detectors with $M = 2$ and $M = 4$ for a 4 × 4 16-QAM system. The three detectors have been described using Verilog HDL, and have been synthesized for a 1.08 V TSMC 65-nm CMOS technology using the Synopsys Design Compiler tool. The designs are placed and routed using the Cadence SoC Encounter tool. The power consumption is estimated using the Synopsys PrimePower tool. Table II compares the throughput and the hardware complexity of the proposed detectors with two detectors from the literature: a depth-first soft sphere detector from [16], and a soft K-Best detector from [22]. Because different technologies are used in these designs, to have a fair comparison, we scale the clock frequency and the area [16] and [22] to a common standard,

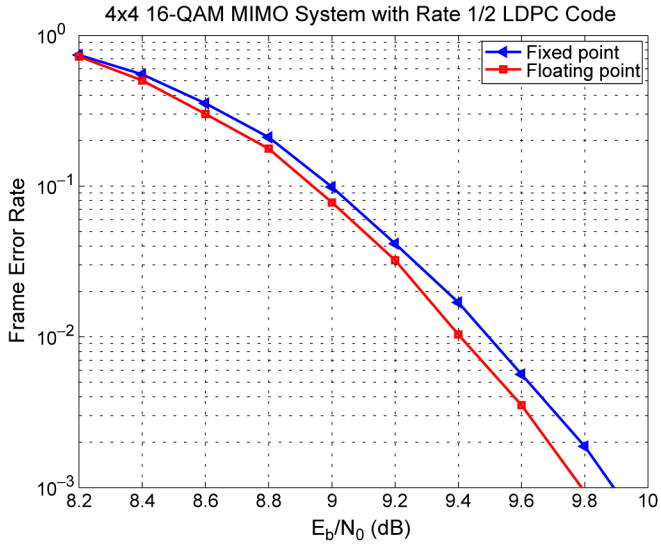


Fig. 19. Fixed-point simulation result for the proposed PPTS detector.

where the scaling factor for area is $(65\text{ nm/Feature size})^2$ and the scaling factor for frequency is (Feature size/65 nm) [36]. To compare the hardware efficiency, we define an area metric as Gate Count/Scaled Throughput.

The depth-first detector from [16] has a variable throughput of 10-95 Mbps because the number of visited nodes will change for different SNR levels. The sequential nature of the depth-first algorithm makes it hard to achieve very high throughput. The K-Best detector from [22] has a fixed throughput that will not change with the SNR level. The K-Best detector from [22] can achieve a 106 Mbps throughput when running at 200 MHz. After scaling the clocks of these two reference designs to 65-nm technology, the scaled throughputs of detector [16] and detector [22] become 38-365 and 212 Mbps, respectively.

The proposed systolic PPTS detector with $M = 2$ can achieve a 6.4 Gbps throughput when running at 400 MHz. As can be seen from the table, the proposed detectors achieve very high data throughput while still maintaining a low area requirement. Compared to the K-Best detector from [22], the PPTS detectors with $M = 2$ have a better area efficiency based on the area metric defined in Table II. The PPTS detector with $M = 2$ achieves a balanced tradeoff between hardware complexity and error performance (<0.3 dB loss). The PPTS detector with $M = 4$ achieves a close-to-optimal error performance (cf. Fig. 6) with a reasonable hardware cost. Therefore, the proposed detector is a viable solution for the Gbps MIMO detection problem as it achieves both high throughput performance and good error performance.

VII. CONCLUSION

We propose a novel soft-output MIMO detector architecture based on the PPTS algorithm. The PPTS algorithm is a search-efficient algorithm based on a path-preserving trellis search approach. We introduce a path reduction and a path extension algorithm to reduce the search complexity while still maintaining sufficient soft information values to form the LLRs. We avoid the missing counter-hypothesis problem by keeping multiple paths during the trellis search process. Simulation results show

TABLE II
ARCHITECTURE COMPARISON

Reference	[16]	[22]	Systolic	Folded I	Folded II
Algorithm	Depth-First	K-Best (K=5)	PPTS (M=2)	PPTS (M=2)	PPTS (M=4)
MIMO Configuration	4x4 16-QAM	4x4 16-QAM	4x4 16-QAM	4x4 16-QAM	4x4 16-QAM
Clock Frequency	71 MHz	200 MHz	400 MHz	400 MHz	400 MHz
Scaled Clock Frequency [a]	273 MHz	400 MHz	400 MHz	400 MHz	400 MHz
Throughput	10-95 Mbps	106 Mbps	6.4 Gbps	2.1 Gbps	2.1 Gbps
Scaled Throughput [a]	38-365 Mbps	212 Mbps	6.4 Gbps	2.1 Gbps	2.1 Gbps
Core Area	1.9 mm^2	0.56 mm^2	3.19 mm^2	1.18 mm^2	2.44 mm^2
Scaled Area [a]	0.13 mm^2	0.14 mm^2	3.19 mm^2	1.18 mm^2	2.44 mm^2
Gate Count	56.8 K	97 K	2.22 M	820 K	1.7 M
Power	N/A	626 mW [b]	312 mW	113 mW	239 mW
Technology	250 nm	130 nm	65 nm	65 nm	65 nm
$\frac{\text{Gate Count}}{\text{Scaled Throughput}}$ (KG/Mbps)	0.15-1.6	0.46	0.35	0.39	0.81

[a] Frequency, throughput, and area are all scaled to 65-nm technology.
[b] This power was reported for a 53.3 Mbps hard-output K-Best detector at 100 MHz clock frequency and 2.8 V supply in a 350-nm technology.

that the PPTS algorithm can achieve very good error performance with a small M value. The PPTS algorithm is a data-parallel algorithm and is very suitable for high speed VLSI implementation. Based on the PPTS algorithm, we have synthesized a systolic detector and two folded detectors for a 4×4 16-QAM system. Compared with tree-search based detectors, the proposed detectors have a significant improvement in terms of detection throughput and area efficiency. The proposed MIMO detector has great potential for application in the next generation Gbps wireless systems by achieving very high throughput and good error performance.

REFERENCES

[1] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Techn. J.*, vol. 1, no. 2, pp. 41–59, 1996.

[2] G. Fettweis, T. Hentschel, and E. Zimmermann, "WIGWAM—A wireless gigabit system with advanced multimedia support," in *Proc. VDE Kongress*, 2004, pp. 18–20.

[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, no. 170, pp. 463–471, Apr. 1985.

[4] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[5] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[6] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8-1, pp. 2806–2818, Aug. 2005.

[7] H. Vikalo and B. Hassibi, "On the sphere-decoding algorithm II. Generalizations, second-order statistics, and applications to communications," *IEEE Trans. Signal Process.*, vol. 53, no. 8-1, pp. 2819–2834, Aug. 2005.

[8] B. 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.

[9] B. Widdup, G. Woodward, and G. Knagge, "A highly-parallel VLSI architecture for a list sphere detector," in *Proc. IEEE Int. Conf. Commun.*, 2004, pp. 2720–2725.

[10] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[11] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, pp. 1544–1552, Sep. 2004.

[12] Y. Zhang, J. Tang, and K. K. Parhi, "Low complexity list updating circuits for list sphere decoders," in *Proc. IEEE Workshop Signal Process. Design Implement.*, 2006, pp. 28–33.

[13] J. Antikainen, P. Salmela, O. Silven, M. Juntti, J. Takala, and M. Myllyla, "Application-specific instruction set processor implementation of list sphere detector," *EURASIP J. Embed. Syst.*, vol. 2007, no. 3, pp. 1–14, 2007.

[14] Q. Qi and C. Chakrabarti, "Sphere decoding for multiprocessor architectures," in *Proc. IEEE Workshop Signal Process. Design Implem.*, 2007, pp. 50–55.

[15] X.-M. Huang, C. Liang, and J. Ma, "System architecture and implementation of MIMO sphere decoders on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 2, pp. 188–197, Feb. 2008.

[16] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.

[17] M. Myllyla, M. Juntti, and J. R. Cavallaro, "Architecture design and implementation of the increasing radius—List sphere detector algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2009, pp. 553–556.

[18] J. W. Choi, B. Shim, A. C. Singer, and N. I. Cho, "Low-complexity decoding via reduced dimension maximum-likelihood search," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1780–1793, Mar. 2010.

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

[20] K. Higuchi, H. Kawai, N. Maeda, M. Sawahashi, T. Itoh, Y. Kakura, A. Ushirokawa, and H. Seki, "Likelihood function for QRM-MLD suitable for soft-decision turbo decoding and its performance for OFCDM MIMO multiplexing in multipath fading channel," in *Proc. IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun. (PIMRC)*, 2004, pp. 1142–1148.

[21] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 930–935, Jun. 2005.

[22] Z. Guo and P. Nilsson, "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.

[23] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 1151–1154.

[24] Q. Li and Z. Wang, "Improved K-best sphere decoding algorithms for MIMO systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 1159–1162.

[25] S. Chen, T. Zhang, and Y. Xin, "Relaxed K-Best MIMO Signal Detector Design and VLSI Implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 3, pp. 328–337, Mar. 2007.

[26] M. Shabany, K. Su, and P. G. Gulak, "A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 3173–3176.

[27] R. Fasthuber, M. Li, D. Novo, P. Raghavan, L. Van Der Perre, and F. Catthoor, "Novel energy-efficient scalable soft-output SSFE MIMO detector architectures," in *Proc. IEEE Int. Symp. Syst., Arch., Model., Simulation*, 2009, pp. 20–23.

[28] S. Mondal, A. Eltawil, C.-A. Shen, and K. N. Salama, "Design and Implementation of a Sort-Free K-Best Sphere Decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 10, pp. 1497–1501, Nov. 2009.

[29] T. Cupaiuolo, M. Siti, and A. Tomasoni, "Low-complexity high throughput VLSI architecture of soft-output ML MIMO detector," in *Proc. Des., Autom. Test Eur. Conf. Exhib. (DATE)*, 2010, pp. 1396–1401.

[30] Y. Sun and J. R. Cavallaro, "High throughput VLSI architecture for soft-output MIMO detection based on a greedy graph algorithm," in *Proc. ACM Great Lakes Symp. VLSI Design*, 2009, pp. 445–450.

[31] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, no. 2, pp. 169–176, Feb. 1984.

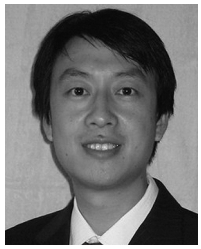
[32] D. E. Knuth, *Art of Computer Programming Volume 3: Sorting and Searching*. Boston, MA: Addison-Wesley, 1998.

[33] P. Luethi, C. Studer, S. Duetsch, E. Zgraggen, H. Kaeslin, N. Felber, and W. Fichtner, "Gram-Schmidt-based QR decomposition for MIMO detection: VLSI implementation and comparison," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 2008, pp. 830–833.

[34] J. Chen, A. Dholakai, E. Eleftheriou, M. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[35] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 1009–1013.

[36] J. R. Hauser, "MOSFET device scaling," in *Handbook of Semiconductor Manufacturing Technology*, R. Doering and Y. Nishi, Eds. Boca Raton, FL: CRC Press, 2008, ch. 1.3, pp. 8–21.



Yang Sun (S'07) received the B.S. degree in testing technology and instrumentation and the M.S. degree in instrument science and technology from Zhejiang University, Hangzhou, China, in 2000 and 2003, respectively, and the Ph.D. degree in electrical and computer engineering from Rice University, Houston, TX, in 2010.

His research interests include parallel algorithms and VLSI architectures for wireless communication systems, digital signal processing systems, multimedia systems, and general purpose computing systems.

Dr. Sun was a recipient of the 2008 IEEE SoC Conference Best Paper Award, the 2008 IEEE Workshop on Signal Processing Systems Best Paper Award (Bob Owens Memory Paper Award), and the 2009 ACM GLSVLSI Best Student Paper Award.

Joseph R. Cavallaro (S'78–M'82–SM'05) received the B.S. degree from the University of Pennsylvania, Philadelphia, in 1981, the M.S. degree from Princeton University, Princeton, NJ, in 1982, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1988, all in electrical engineering.

From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, NJ. In 1988, he joined the faculty of Rice University, Houston, TX, where he is currently a Professor of electrical and computer engineering. His research interests include computer arithmetic, VLSI design and microlithography, and DSP and VLSI architectures for applications in wireless communications. During the 1996-1997 academic year, he served at the National Science Foundation as Director of the Prototyping Tools and Methodology Program. He was a Nokia Foundation Fellow and a Visiting Professor with the University of Oulu, Finland, in 2005 and continues his affiliation there as an Adjunct Professor. He is currently the Director of the Center for Multimedia Communication at Rice University.

Dr. Cavallaro was Co-chair of the 2004 Signal Processing for Communications Symposium at the IEEE Global Communications Conference and General/Program Co-chair of the 2003, 2004, and 2011 IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP).