# Trellis-Search Based Soft-Input Soft-Output MIMO Detector: Algorithm and VLSI Architecture

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

***Abstract*—In this paper, we propose a trellis-search based soft-input soft-output detection algorithm and its very large scale integration (VLSI) architecture for iterative multiple-input multiple-output (MIMO) receivers. We construct a trellis diagram to represent the search space of a transmitted MIMO signal. With the trellis model, we evenly distribute the workload of candidates searching among multiple trellis nodes for parallel processing. The search complexity is significantly reduced because the number of candidates is greatly limited at each trellis node. By leveraging the trellis structure, we develop an approximate Log-MAP algorithm by using a small list of largest exponential terms to compute the LLR (log-likelihood ratio) values. The trellis-search based detector has a fixed-complexity and is very suitable for parallel VLSI implementation. As a case study, we have designed and synthesized a trellis-search based soft-input soft-output MIMO detector for a $4 \times 4$ 16-QAM system using a 1.08 V TSMC 65 nm technology. The detector can achieve a maximum throughput of 1.7 Gb/s with a core area of 1.58 mm$^2$.**

***Index Terms*—ASIC, MIMO algorithm, soft-input soft-output MIMO detection, trellis-search algorithm, VLSI.**

## I. Introduction

WIRELESS systems are adopting multiple-antenna configurations with spatial multiplexing technique to support parallel streams of wireless data. As an example, the Vertical Bell Laboratories Layered Space-Time (V-BLAST) system has been shown to achieve very high spectral efficiency [1]. One bottleneck in such multiple-input, multiple-output (MIMO) communication systems is the need to process a large amount of data received at one end of a digital communication channel to detect a noisy signal transmitted simultaneously by a number of transmit antennas.

The optimal soft-input soft-output MIMO detector is based on the log maximum *a posteriori* probability (Log-MAP) algorithm, which is too computationally intensive to be implemented in a practical MIMO receiver, because the Log-MAP algorithm requires calculating two log-sums of $\frac{Q^{N_t}}{2}$ exponential terms, where $Q$ is the constellation size (i.e., the number of possible symbols of a modulation alphabet of a transmitted signal), and $N_t$ is the number of the transmit antennas. A brute-force implementation of an optimal Log-MAP algorithm consumes enormous computing power, which makes it impractical to be employed in multiple antenna systems with higher-order modulation schemes.

There are two main approaches to MIMO detection problems: the depth-first tree-search algorithms such as sphere detection [2]–[8], and the breadth-first tree-search algorithms such as K-best detection [9]–[13]. There are many hardware implementations of the sphere detectors and the K-best detectors, such as [14]–[21], [8], [22]–[26]. To support iterative MIMO detection, several tree-search based soft-input soft-output detection algorithms are developed by researchers [27]–[30]. However, there are some drawbacks to using the tree-search based detectors. For the depth-first sphere detection, the number of visited nodes is large in the low SNR (signal-to-noise) regime, while the number of visited nodes is small in the high SNR regime. As a result, the depth-first sphere detector has a variable throughput which is undesirable in systems with strict latency requirements. In the K-best detection, the number of visited nodes is fixed independent of SNR. However, a large K value is required to achieve good performance. The main challenge for implementing the K-best detector is to sort a large number of candidates. In addition, the candidate list generated by a sphere detector or a K-best detector does not guarantee that the bit-level soft information, or the log likelihood ratio (LLR), can be found for every data bit, which will lead to some performance degradation.

In this paper, we propose a trellis-search based detection algorithm for iterative MIMO detection. We use an unconstrained trellis structure as an alternative to the tree structure to represent the search space of a MIMO signal. We propose a trellis-based approximate Log-MAP algorithm as a replacement of the typically used Max-Log algorithm for iterative MIMO detection. We search the trellis to find a number of most likely paths for each trellis node and compute a log-sum of a number of exponential terms corresponding to a hypothesized transmitted bit value. Near-optimal performance can be achieved by choosing an appropriate number of surviving paths in the trellis search process. The trellis-based detection algorithm is a very data-parallel algorithm because the searching operations at multiple trellis nodes can be performed simultaneously. The local search complexity at each trellis node is kept very low to reduce the overall processing time. Moreover, the trellis-based detector can support iterative MIMO detection by utilizing the *a priori* information from the outer channel decoder.

The rest of the paper is organized as follows. Section II summarizes the MIMO system model. Section III introduces the trellis-search based iterative MIMO detection algorithm.

Manuscript received October 14, 2011; revised January 25, 2012; accepted January 27, 2012. Date of publication February 13, 2012; date of current version April 13, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tong Zhang. This work was supported in part by Renesas Mobile, Texas Instruments, Xilinx, and by the U.S. National Science Foundation under Grants EECS-0925942 and 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/TSP.2012.2187646

Section IV shows the simulation results. Section V presents the proposed VLSI architecture. Section VI summarizes the VLSI implementation results and the architecture comparison with state-of-the-art solutions. Finally, Section VII concludes the paper.

## II. System Model

In this section, we review the system model for MIMO communication systems. 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 an $N_r \times N_t$ complex matrix and is assumed to be known perfectly at the receiver, $\mathbf{s}$ is an $N_t \times 1$ transmit symbol vector, $\mathbf{s} = [s_0\ s_1\ \ldots\ s_{N_t-1}]^T$, $\mathbf{y}$ is an $N_r \times 1$ 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 $\sigma^2$ per real component. A real bit-level vector $\mathbf{x_k} = [x_{k,0}\ x_{k,1}\ \ldots\ x_{k,B-1}]^T$ is mapped to the complex symbol $s_k$ as $s_k = \mathrm{QAM}(\mathbf{x_k})$, where the $b$th bit of $\mathbf{x_k}$ is denoted as $x_{k,b}$, $B$ is the number of bits per constellation point, and $Q$ is the constellation size ($B = \log_2(Q)$).

The optimal log maximum a *posteriori* probability (Log-MAP) detector is to compute the log-likelihood ratio (LLR) value for the *a posteriori* probability (APP) of each transmitted bit. The LLR value for each bit $x_{k,b}$ is computed as [5]

$$\begin{aligned}\mathrm{LLR}(x_{k,b}) &= \ln \sum_{\mathbf{s}:x_{k,b}=+1} \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 + \frac{1}{2}\sum_{i=0}^{N_t-1}\sum_{j=0}^{B-1} x_{i,j}L_A(x_{i,j})\right)\\ &\quad - \ln \sum_{\mathbf{s}:x_{k,b}=-1} \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 + \frac{1}{2}\sum_{i=0}^{N_t-1}\sum_{j=0}^{B-1} x_{i,j}L_A(x_{i,j})\right)\end{aligned} \tag{2}$$

where $L_A(x_{i,j})$ is the *a priori* LLR for bit $x_{i,j}$. In an iterative MIMO receiver, the *a priori* LLR $L_A(x_{i,j})$ will be provided by a channel decoder.

## III. Trellis-Search Based Iterative MIMO Detection Algorithm

The LLR computation in (2) requires calculations of two log-sums of $\frac{Q^{N_t}}{2}$ exponential terms. The brute-force implementation of (2) is too complex. As a balanced tradeoff between complexity and performance, we propose to use a reduced number ($n$) of exponential terms to approximate the original Log-MAP algorithm:

$$\begin{aligned}\mathrm{LLR}(x_{k,b}) &\approx \ln \sum_{n:x_{k,b}=+1} \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 + \frac{1}{2}\sum_{i=0}^{N_t-1}\sum_{j=0}^{B-1} x_{i,j}L_A(x_{i,j})\right)\\ &\quad - \ln \sum_{n:x_{k,b}=-1} \exp\left(\frac{-1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 + \frac{1}{2}\sum_{i=0}^{N_t-1}\sum_{j=0}^{B-1} x_{i,j}L_A(x_{i,j})\right)\end{aligned} \tag{3}$$

where the $n$ exponential terms used in each log-sum computation are preferably the $n$ largest exponential terms. Based on the QR decomposition of the channel matrix ($\mathbf{H} = \mathbf{Q}\mathbf{R}$), (3) can be written as

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

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

$$d(\mathbf{s}) = \sum_{i=0}^{N_t-1}\left(|(\hat{\mathbf{y}})_i - (\mathbf{R}\mathbf{s})_i|^2 - \sigma^2 \sum_{j=0}^{B-1} x_{i,j}L_A(x_{i,j})\right). \tag{5}$$

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

In order to implement (4), we must find $n$ minimum distances $d(\mathbf{s})$ for each hypothesized transmitted data bit, i.e., $x_{k,b} = +1$ and $x_{k,b} = -1$. To realize this goal, we propose a trellis-search algorithm to find the $n$ minimum distances.

### A. Proposed Trellis Model for Iterative MIMO Detection

We introduced a trellis model for noniterative MIMO detection in [31] and [32], where we assume there is no *a priori* information available to the detector. The suboptimal Max-Log approximation algorithm was used in [31] and [32] to compute the LLRs. In this new work, we extend the trellis model introduced in our earlier work [31] and [32] to support iterative MIMO detection. The *a priori* information is incorporated into the path metrics computation to support iterative MIMO detection. We propose a more reliable LLR generation algorithm by replacing the Max-Log algorithm with the multi-term Log-MAP algorithm. We use the trellis model to find the $n$ minimum distances to compute the LLRs as shown in (4). The performance of the proposed algorithm is very close to that of the optimal Log-MAP algorithm while still maintaining low implementation complexity.

The search space of a MIMO signal can be represented with a compact trellis diagram. As an example, Fig. 1 shows the trellis diagram for a $4 \times 4$ 4-QAM system. The trellis has $N_t$ stages

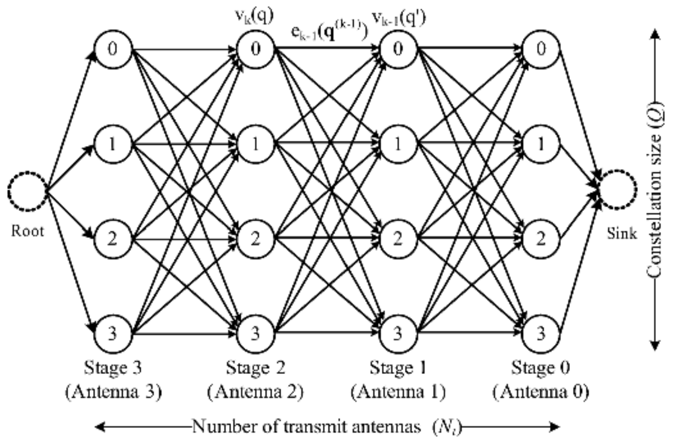

Fig. 1. A trellis diagram for a 4 × 4 4-QAM system. The trellis has $N_t$ stages corresponding to $N_t$ number of transmit antennas, and each stage contains $Q$ different nodes corresponding to $Q$ number of constellation points.

corresponding to $N_t$ transmit antennas, and each stage contains $Q$ different nodes corresponding to $Q$ symbols of a complex constellation of the transmitted signal. In other words, the trellis is formed of columns representing the number of transmit antennas and rows representing values of a number of symbols with nodes at intersections. Each trellis node is physically mapped to a transmit symbol that belongs to a known modulation alphabet of the $Q$ constellation symbols. Thus, any path through the trellis represents a possible vector ($\mathbf{s}$) of transmitted symbols. Because of the upper triangular property of the matrix $\mathbf{R}$, the stages of the trellis are labeled in descending order. The trellis is fully connected, so there are $Q^{N_t}$ number of different paths from the root node to the sink node. The nodes in stage $k$ are denoted as $v_k(q)$, where $q = 0, 1, \ldots, Q-1$.

To compute the distance metric in (5) using the trellis model, we define a weight function $e_{k-1}(\mathbf{q}^{(k-1)})$ for each edge between node $v_k(q)$ in stage $k$ and node $v_{k-1}(q')$ in stage $k-1$ as

$$e_{k-1}\left(\mathbf{q}^{(k-1)}\right) = \left|\hat{y}_{k-1} - \sum_{j=k-1}^{N_t-1} R_{k-1,j} \cdot s_j\right|^2 - \sigma^2 \sum_{b=0}^{B-1} x_{k-1,b} \cdot L_A(x_{k-1,b}) \quad (6)$$

where $\mathbf{q}^{(k-1)} = [q_{N_t-1} \ldots q_k \ q_{k-1}]^T$ is the partial symbol vector, $s_j$ is the complex-valued QAM symbol $s_j = \mathrm{QAM}(q_j)$, $B$ is the number of bits per constellation point, and $L_A(x_{k-1,b})$ is the *a priori* information for data bit $x_{k-1,b}$ provided by the outer channel decoder. In the first iteration, $L_A(x_{k-1,b})$ is not available and is set to 0. Note that the weight function not only depends on nodes $v_k(q)$ and $v_{k-1}(q')$, but also depends on all the nodes prior to node $v_k(q)$. In other words, depending on how we traverse the trellis, the weight function will get different values. We further define a path weight as the sum of the edge weights along the path. Then, the distance metric as defined in (5) can be considered as a path weight, which can be computed recursively by adding up the edge weights along the path from the root node to the sink node. If we define a (partial) path metric $d_k$ as the sum of the edge weights along this (partial) path, the path weight is then computed recursively as

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

where $d_k(q)$ and $d_{k-1}(q')$ are the path weights associated with nodes $v_k(q)$ and $v_{k-1}(q')$, respectively, and $e_{k-1}(\mathbf{q}^{(k-1)})$ is the edge weight between node $v_k(q)$ and node $v_{k-1}(q')$.

## B. Per Trellis-Node Shortest Paths Problem

In the trellis diagram, each trellis node $v_k(q)$ maps to a complex-valued symbol $s_k$ such that each path from the root node to the sink node maps to a symbol vector $\mathbf{s}$. With the trellis model, we transform the MIMO detection problem into a per-node shortest paths problem, which is defined as follows. *For each node $v_k(q)$ in the trellis diagram, find a list of $L$ most likely paths from the root node to the sink node over the node $v_k(q)$.* The $L$ most likely paths refer to the paths with the $L$ shortest distances or the lowest $L$ path weights. For each node, we only keep the $L$ most likely paths and will discard all the other paths to reduce the complexity.

We use a layered detection method, where a layer $k$ refers to a transmit antenna $k$. The detection is performed layer by layer. In the trellis model, a layer corresponds to a stage in the trellis. In each stage $k$ of the trellis, there are $Q$ nodes, where each node corresponds to a constellation point. For each node $v_k(q)$ in stage $k$, $q = 0, 1, \ldots, Q-1$, we must find $L$ shortest paths through the trellis, which are denoted as $\lambda_k^{(l)}(q)$, $l = 0, 1, \ldots, L-1$. Then, altogether $QL$ candidates in each stage $k$ of the $N_t$ stages of the trellis are used to compute the LLRs for data bits transmitted by antenna $k$ as follows:

$$\mathrm{LLR}(x_{k,b}) \approx \ln \sum_{(q,l):x_{k,b}=+1} \exp\left(-\frac{1}{2\sigma^2}\lambda_k^{(l)}(q)\right) - \ln \sum_{(q,l):x_{k,b}=-1} \exp\left(-\frac{1}{2\sigma^2}\lambda_k^{(l)}(q)\right). \quad (8)$$

With the trellis model, the detection problem now becomes a trellis-search problem. To detect a layer $k$, we need to search for $L$ shortest paths for each node $q$ in each stage $k$ of the trellis diagram. The maximum theoretical value of the number $L$ is $Q^k$, where $k = 0, 1, \ldots, N-1$ for the first stage, second stage, and etc., of the trellis. Practically, however, the number $L$ should be kept small to reduce the complexity. The number $L$ determines the detection performance: a larger $L$ leads to better error performance. We will show later that even with a small $L$ (such as $L = 2$ for $Q = 16$), the trellis-based detector can achieve good detection performance. To implement this algorithm, an exhaustive trellis search approach would be very expensive. To reduce the search complexity, we next introduce a low-complexity trellis-search algorithm.

## C. Trellis-Search Algorithm for Iterative MIMO Detection

In order to reduce the search complexity, we propose a greedy trellis-search algorithm that approximately finds the $L$ shortest paths for each node in the trellis. In this search process, the trellis is first pruned by removing the unlikely paths. We refer to this pruning process as the "path reduction" process. In the

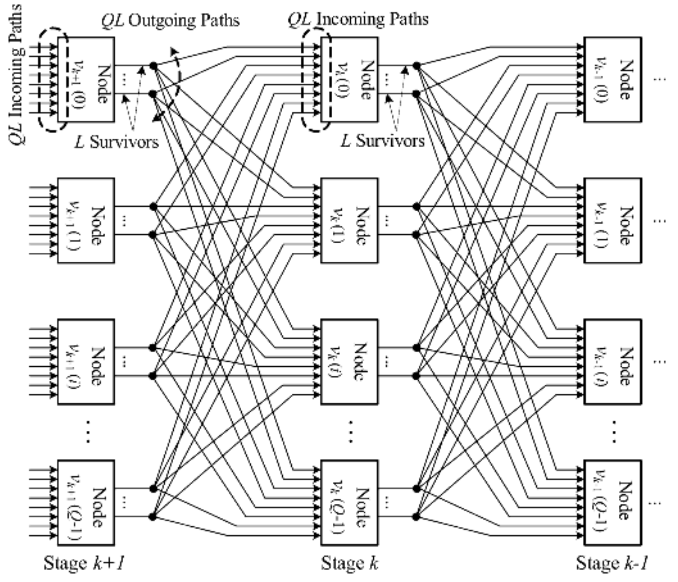

Fig. 2. Flow of the path reduction algorithm, where each node evaluates its $QL$ incoming paths and selects the best $L$ paths.

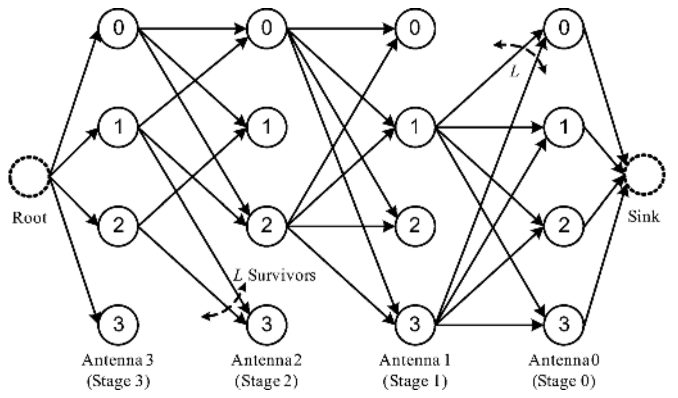

Fig. 3. Path reduction example for a $4 \times 4$ 4-QAM trellis, where $L = 2$ incoming paths are kept at each node.

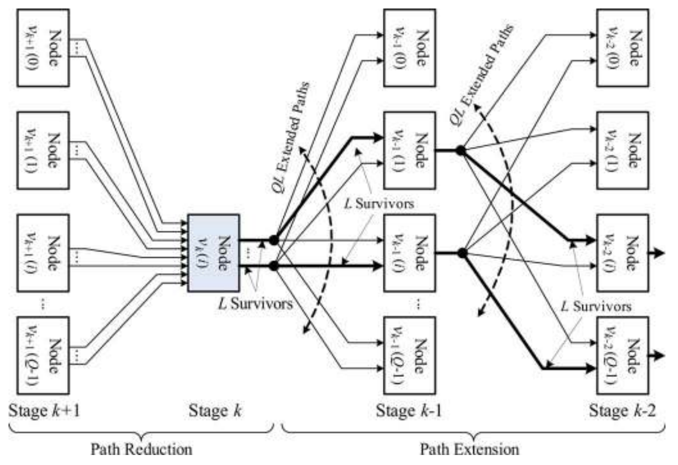

Fig. 4. Flow of the path extension algorithm for extending one node $v_k(i)$, where $L$-best outgoing paths are selected from this node to each of the following stages $(k-1, k-2, \ldots, 0)$.

path reduction process, the trellis is scanned from left to right, where each node retains the most likely $L$ incoming paths using the local information it has so far. After the trellis is pruned, a second process, called the "path extension" process, is applied to extend the uncompleted paths so that each node will have $L$ full paths through the trellis.

*1) Path Reduction:* Fig. 2 illustrates a flow graph demonstrating a path reduction process. The path reduction process is configured to prune paths for each trellis node to a smaller number of surviving paths. The stages (columns) of the trellis are labeled in descending order, starting from stage $N_t - 1$ and ending with stage 0. Note that Fig. 2 illustrates only three successive stages, $k+1$, $k$, and $k-1$ among the $N_t$ stages. As an example, we use a $Q = 4$, $L = 2$ case to explain the algorithm. In Fig. 2, each node receives $QL = 4 \times 2 = 8$ incoming paths from nodes in the previous stage of the trellis and, then, the $L = 2$ paths (the ones with the least cumulative path weights) are selected from the $QL$ candidates. Next, the $L$ survivors are expanded to the right so that each node will have the best $QL$ outgoing paths forwarded to the next stage of the trellis. This process repeats until the end of the trellis. Fig. 3 illustrates a diagram of the trellis after the path reduction process. This figure demonstrates a $4 \times 4$ 4-QAM trellis after applying the path reduction procedure, where each node keeps only $L = 2$ best incoming paths, the ones with the least cumulative path weights.

The path reduction process can effectively prune the trellis by keeping only $L$-best incoming paths at each trellis node. As a result, each trellis node in the last stage of the trellis has $L$ shortest paths through the trellis. However, other than the trellis nodes in the last stage, the path reduction process can not guarantee that every trellis node will have $L$ shortest paths through the trellis. For example, in Fig. 3, node 3 in stage 2 shows no outgoing paths because these paths were dropped as incoming paths by the respective trellis nodes in the next stage. These paths will be added as path extensions as described next.

*2) Path Extension:* An objective of the trellis-based detection algorithm is to find $L$ shortest paths for every node in the trellis. To achieve this goal, a path extension process is employed after the path reduction process to fill in the missing paths for each trellis node. The goal is to extend the uncompleted paths so that each node will have $L$ shortest paths through the trellis.

The path extension is performed stage by stage (no path extension is required for the last stage), and node by node. Fig. 4 is a flow graph demonstrating the path extension process. The path extension process is being demonstrated with respect to a node $v_k(i)$ in a stage $k$ (i.e., the highlighted node in the figure). Note that all of the nodes in the same stage can be extended in parallel and independently.

As shown in Fig. 4, for a trellis node $v_k(i)$ (i.e., for the constellation point $i$ in stage $k$), the path extension process first retrieves the $QL = 8$ outgoing path metrics computed in the path reduction step (at stage $k$), and then an extension process in stage $k-1$ is used to select the best $L = 2$ *outgoing* paths (e.g., with minimal distance to the nodes in the next stage) from $QL = 8$ candidates. Next, each of the $L = 2$ surviving paths is extended for the next stage of the trellis (stage $k-2$). Among the $QL$ extended paths, only the best $L = 2$ paths are retained. This process repeats until the trellis has been completely traversed. As a result, the $L$ shortest paths are obtained for node $v_k(i)$.

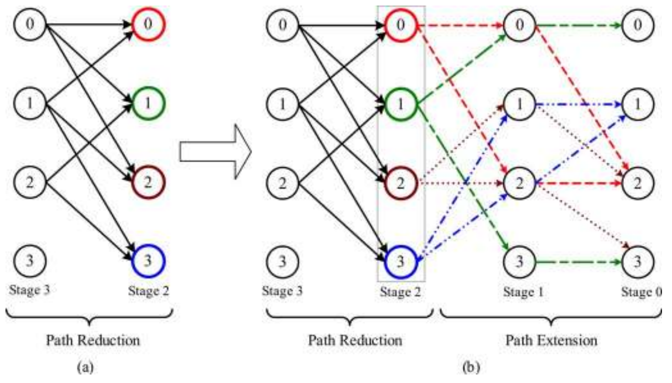

Fig. 5. Path extension example for nodes in stage 2, where $L = 2$ full paths are extended from each node $v_2(i), i = 0, 1, \ldots, Q-1$ to the end of the trellis. (a) Path reduction process. (b) Path extension process.

Fig. 4 shows a path extension process for one trellis node. In fact, all the nodes in stage $k$ are extended as necessary so that each node can find $L$ shortest paths through the trellis. Generally, as shown in Fig. 4, to detect a symbol transmitted by antenna $k$, the entire search process can be expressed as $N_t - k$ stages of path reductions followed by $k$ stages of path extensions. In other words, the path reduction process is first performed until stage $k$ of the trellis and next the path extension procedure is performed until the end of the trellis (stage 0). Note that the path extension process is to find the $L$-best *outgoing* paths extending from a particular node.

To illustrate the flow of the path extension process, we use the same trellis example to show the path extension process for extending the nodes in stage 2. Fig. 5(a) illustrates the trellis after two stages of path reduction, where $L = 2$ shortest paths are obtained for each node in stage 2. After the path reduction process, each node in stage 2 has $L = 2$ incoming paths. The objective is to find $L$ full paths through each node because the previous stage was pruned to $L$ paths. As shown in Fig. 5(b), after the path extension process, every node in stage 2 has successfully obtained $L = 2$ shortest paths through the trellis.

*3) LLR Computation:* The most important feature of the trellis-based detection algorithm is that it will always guarantee that the bit LLR can be generated for every transmitted bit. For example, after the path reduction and the path extension processes are employed, every node $v_k(q)$ has successfully found $L$ shortest paths or $L$ minimum distances denoted as $\lambda_k^{(l)}(q)$, $l = 0, 1, \ldots, L-1$. The LLR for data bit $x_{k,b}$ transmitted by antenna $k$ is then computed as follows:

$$\mathrm{LLR}(x_{k,b}) \approx \ln \sum_{(q,l):x_{k,b}=+1} \exp\left(-\frac{1}{2\sigma^2}\lambda_k^{(l)}(q)\right) - \ln \sum_{(q,l):x_{k,b}=-1} \exp\left(-\frac{1}{2\sigma^2}\lambda_k^{(l)}(q)\right). \quad (9)$$

We can separate the computation of (9) into two steps. A symbol reliability metric $\Gamma_k(q)$ is first computed for each node $v_k(q)$ as follows:

$$\Gamma_k(q) = \ln \sum_{l=0}^{L-1} \exp\left(\frac{-1}{2\sigma^2}\lambda_k^{(l)}(q)\right) = \overset{*}{\max_l}\left(\frac{-1}{2\sigma^2}\lambda_k^{(l)}(q)\right) \quad (10)$$

where the two-input $\max^*(\cdot)$ is defined as

$$\begin{aligned}\overset{*}{\max}(a,b) &\equiv \ln \sum(\exp(a) + \exp(b)) \\ &= \max(a,b) + \ln(1 + \exp(-|a-b|)). \quad (11)\end{aligned}$$

Moreover, the $n$-input $\max^*(\cdot)$ for $n = 4, 8, 16$, etc., can be recursively computed based on the Jacobian algorithm. Then, the bit LLR is computed based on the symbol reliabilities $\Gamma_k(q)$:

$$\begin{aligned}\mathrm{LLR}(x_{k,b}) &= \ln \sum_{q:x_{k,b}=+1} \exp(\Gamma_k(q)) \\ &\quad - \ln \sum_{q:x_{k,b}=-1} \exp(\Gamma_k(q)) \\ &= \overset{*}{\max_{q:x_{k,b}=+1}}(\Gamma_k(q)) - \overset{*}{\max_{q:x_{k,b}=-1}}(\Gamma_k(q)). \quad (12)\end{aligned}$$

It should be noted that the nonlinear function $f(|x|) = \ln(1 + \exp(-|x|))$ can be approximated by a lookup table to reduce the hardware complexity. For example, in our hardware implementation, we used the following eight-entry lookup table to implement the nonlinear function $f(|x|) = \ln(1 + \exp(-|x|))$. The data value in the lookup table is represented with a 6-bit fixed-point format.

## IV. Simulation Results

To evaluate the performance of the trellis-search based MIMO detection algorithm, we performed floating-point simulations for the $4 \times 4$ 16-QAM MIMO system, where the channel matrices are assumed to have independent random Gaussian distributions. A length 2304, rate 1/2 WiMAX low-density parity-check (LDPC) code is used as an outer channel code.

### A. Noniterative MIMO Detection Performance

We first show the noniterative, or "one-shot", detection performance with no outer iterations performed between the MIMO detector and the LDPC decoder. The simulation results are shown in Fig. 6 with the performance of several MIMO detection algorithms including the trellis-search based detection algorithm for several values of $L$, where $L$ is the number of surviving paths at each trellis node. For comparison, simulation results are also plotted for the optimal full Log-MAP algorithm, and for the K-best tree-search based Log-MAP algorithm, where $K = 32$. In the K-best algorithm, $K = 32$ path metrics are used to compute the log-sums to calculate LLRs. To relate the trellis-based detection algorithm to the K-best detection algorithm, the total number of the survivors kept at each stage of the trellis is $QL$ whereas the total number of the survivors kept at each level of the tree in the K-best algorithm is $K$.

From Fig. 6, one can observe that the trellis-based detector with the surviving path number $L = 2$ slightly outperforms the K-best detector with $K = 32$. The trellis-based detector with $L = 3, 4$ clearly outperforms the K-best detector with $K = 32$. The trellis-based detector with $L = 4$ performs close to the optimal full Log-MAP algorithm.

### B. Iterative MIMO Detection Performance

By exchanging soft information between the MIMO detector and the channel decoder, an iterative receiver can significantly improve the performance compared to a noniterative

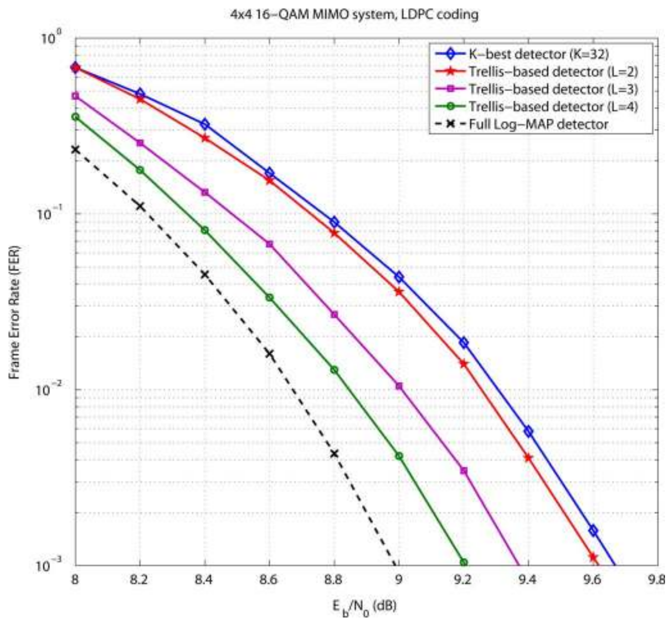

Fig. 6. The performance of the trellis-search based detector with different $L$ values.

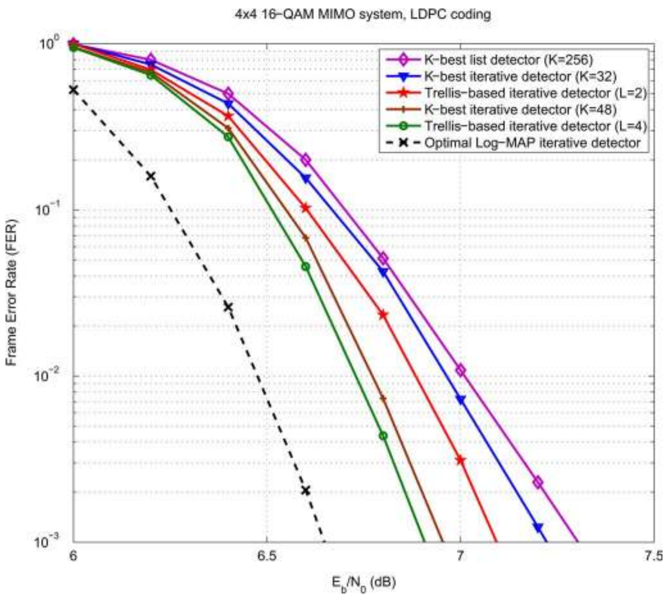

Fig. 7. The performance of the trellis-search based iterative detector with different $L$ values, four outer iterations.

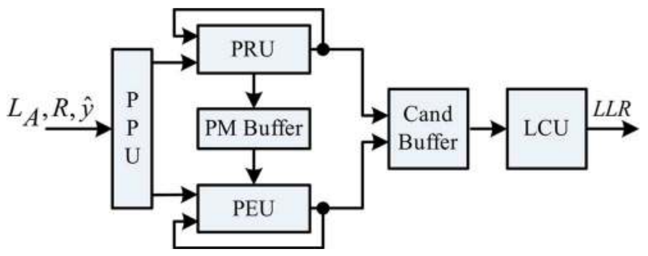

Fig. 8. Top level block diagram of the trellis-based iterative MIMO detector.

receiver. In this subsection, we compare the performance of our trellis-search based iterative detector with three K-best tree-search based iterative detectors, and an optimal full Log-MAP iterative detector. Two flavors of the K-best detectors are used in the simulation: the K-best *list* detector and the K-best *iterative* detector. For the case of the K-best list detector, the detection is performed only once and a list of candidates ($K$ of them) is generated and recorded, and, then, in each outer iteration, the LLR values are computed based on the candidate list and the *a priori* LLR values from the channel decoder. For the case of the K-best iterative detector, the detection is re-performed for each outer iteration by taking into account the *a priori* information from the channel decoder, which is somewhat similar to the proposed trellis-search detector. In the trellis-search based detection algorithm, the number of the survivors at each node is chosen to be 2, 3, and 4, i.e., $L = 2, 3, 4$.

In the simulation, the LDPC inner iteration number is 15, and the number of the outer iterations between the MIMO detector and the LDPC decoder is 4 for all the cases. As can be seen from Fig. 7, the trellis-based iterative detector with $L = 2$ outperforms the K-best list detector with a large $K$ value $(K = 256)$. The trellis-based iterative detector with $L = 2$ also outperforms the K-best iterative detector with $K = 32$, but it performs worse than the K-best iterative detector with $K = 48$. However, the trellis-based iterative detector with $L = 4$ outperforms the K-best iterative detector with $K = 48$, and it performs close to the optimal Log-MAP iterative detector.

The main advantage of the trellis-search based detector is the low sorting cost compared to the K-best detector. In [32], we have given a detailed analysis and comparison of the sorting complexity and the partial Euclidean distance (PED) computation complexity for the trellis-based detector and the tree-search K-best detector.

## V. VLSI Architecture

In this section, we describe a high-speed VLSI architecture for the proposed trellis-search based soft-input soft-output MIMO detector. As a case study, we introduce a detector architecture with the surviving path number $L = 2$ for the $4 \times 4$ 16-QAM system. In our earlier work [32], we have described a "systolic" array detector architecture and a folded detector architecture for the noniterative MIMO detection. In this work, we have significantly modified the design and extended it for the iterative MIMO detection. The major improvements are as follows: 1) a new path metric calculation method by incorporating the *a priori* information, 2) a new LLR computation method based on the proposed multi-term Log-MAP approximation algorithm [cf. (12)], and 3) a new recursive detector architecture and scheduling.

### A. Top Level Architecture

Fig. 8 shows the top level block diagram for the proposed MIMO detector. The detector consists of six main functional blocks: the path reduction unit (PRU), the path extension unit (PEU), the LLR calculation unit (LCU), the preprocessing unit (PPU), the path metric buffer (PM Buffer), and the candidate buffer (Cand Buffer). The PPU is used to precompute the initial path metrics and some constellation-dependent constant values that will be used by the PRU and the PEU. The PRU

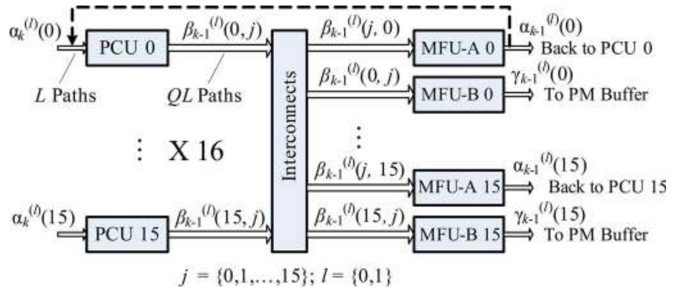
$\alpha_k^{(l)}(0)$
PCU 0
$L$ Paths
$QL$ Paths
$\beta_{k-1}^{(l)}(0,j)$
X 16
$\alpha_k^{(l)}(15)$
PCU 15
$\beta_{k-1}^{(l)}(15,j)$
Interconnects
$\beta_{k-1}^{(l)}(j,0)$
MFU-A 0
$\alpha_{k-1}^{(l)}(0)$ Back to PCU 0
MFU-B 0
$\gamma_{k-1}^{(l)}(0)$ To PM Buffer
MFU-A 15
$\alpha_{k-1}^{(l)}(15)$ Back to PCU 15
MFU-B 15
$\gamma_{k-1}^{(l)}(15)$ To PM Buffer
$j = \{0,1,\ldots,15\}$; $l = \{0,1\}$

Fig. 9. Block diagram of the path reduction unit (PRU).

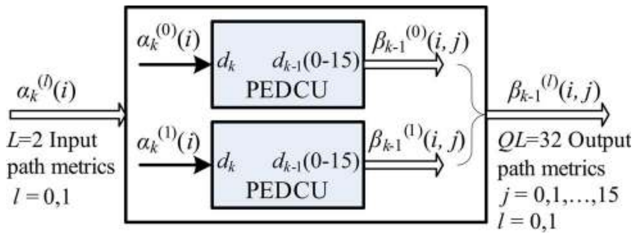

Fig. 10. Block diagram of the path calculation unit (PCU).

and the PEU are employed to implement the path reduction algorithm (cf. Fig. 2) and the path extension algorithm (cf. Fig. 4), respectively. The shortest path metrics found by the PRU and the PEU are stored in the Cand Buffer, which will then be used by the LCU to generate the LLR for each data bit based on (12). These blocks will be discussed in more detail in the following subsections.

## B. Path Reduction Unit (PRU)

Fig. 9 shows the block diagram of PRU, which implements the path reduction algorithm (cf. Fig. 2). The PRU employs $Q = 16$ path calculation units (PCUs) and $16 \times 2$ minimum finder units (MFUs) to simultaneously process all the $Q$ nodes in a trellis stage. This is a recursive architecture by reusing the hardware for processing nodes in different trellis stages. In Fig. 9, PCU $i$ is used to compute the $QL$ extended path metrics from node $v_k(i)$ to all the nodes in the next stage $k-1$. The extended path metrics are denoted as $\beta_{k-1}^{(l)}(i,j)$, where $l$ is the surviving path index ($l = 0,1,\ldots,L-1$), $i$ is the current node index, and $j$ is the node index in the next stage ($j = 0,1,\ldots,Q-1$). Next, the extended path metrics are gathered and sent to MFUs. In Fig. 9, MFU-A $i$ is used to select the best $L$ incoming paths to node $v_{k-1}(i)$, where the surviving path metrics are denoted as $\alpha_{k-1}^{(l)}(i)$, where $l = 0,1,\ldots,L-1$. Then, these surviving paths are fed back to PCU $i$ so that it can continue the processing for the next trellis stage. This operation is repeated until the trellis is completely traversed. MFU-B $i$ is used to select the best $L$ outgoing paths, denoted as $\gamma_{k-1}^{(l)}(i)$, from node $v_k(i)$ to any nodes in stage $k-1$. These best outgoing paths selected by MFU-B $i$ will be stored into the path metric buffer (PM Buffer), which will be used later in the path extension process.

*1) Path Calculation Unit (PCU):* Each PCU in Fig. 9 is used to compute $QL = 32$ path metrics in parallel. Fig. 10 shows the block diagram of PCU which employs $L = 2$ partial Euclidean distance calculation units (PEDCUs). For a given input path metric, or partial Euclidean distance, $d_k$, one PEDCU needs to compute $Q = 16$ extended PEDs in parallel, denoted as $d_{k-1}(q)$, $q = 0,1,\ldots,Q-1$. Based on the trellis structure, we first compute $Q = 16$ edge weights $e_{k-1}(q)$, and, then, $d_{k-1}(q)$ are computed as

$$d_{k-1}(q) = d_k + e_{k-1}(q). \tag{13}$$

The edge weight $e_{k-1}(q)$ is computed based on (6):

$$e_{k-1}(q) = |T + R_{k-1,k-1} \cdot s_{k-1}(q)|^2 - \sigma^2 \sum_{b=0}^{B-1} x_{k-1,b} \cdot L_A(x_{k-1,b}) \tag{14}$$

where a temporary variable $T$ is defined as

$$T = \sum_{j=k}^{N_T-1} R_{k-1,j} \cdot s_j - \hat{y}_{k-1}. \tag{15}$$

We know that $R_{k-1,k-1}$ will be a real value if using a particular QR decomposition algorithm, e.g., Gram–Schmidt QR decomposition [33]. Then, (14) can be re-expressed as

$$\begin{aligned} e_{k-1}(q) = |T|^2 &+ R_{k-1,k-1}^2 |s_{k-1}(q)|^2 \\ &+ 2\mathrm{Re}\left((R_{k-1,k-1} \cdot T^*) s_{k-1}(q)\right) \\ &- \sigma^2 \sum_{b=0}^{B-1} x_{k-1,b} \cdot L_A(x_{k-1,b}). \end{aligned} \tag{16}$$

Fig. 11 shows the hardware architecture for the PEDCU, which computes $Q = 16$ PEDs in parallel. Note that variables $R_{k-1,k-1}^2 |s_{k-1}(q)|^2$ and $\sigma^2 L_A(x_{k-1,b})$ are precomputed in the preprocessing unit (PPU). The constant multiplication of "$A = B \cdot s_j$" can be implemented using a shift and add module.

## C. Min Finder Unit (MFU)

The min finder unit (MFU) is used to select the best $L = 2$ path metrics from $QL = 32$ candidates. This type of (32,2) sorting can be done quickly by using a comparison tree. Note that the sorting cost of the trellis-based detector is much lower compared with the regular K-best detector which typically requires a larger $(QK, K)$ sorting operation.

## D. Path Extension Unit (PEU)

The PEU implements the path extension algorithm (cf. Fig. 4). As we discussed in Section III-C-2, a path extension process is employed after the path reduction process to fill in the missing paths for each node so that every node will have $L$ shortest paths through the trellis.

The PEU has a very similar architecture to the PRU. Fig. 12 shows the block diagram of PEU, which employs $Q = 16$ PCUs and $Q = 16$ MFUs so that it can simultaneously extend $Q$ nodes in a certain trellis stage. The PEU has a recursive architecture. In each iteration, PCU $i$ calculates the $QL$ extended path candidates based on the $L$ input path metrics, and then, the MFU $i$ selects the best $L$ paths from these $QL$ extended path candidates. The initial $L$ input path metrics are retrieved from the PM Buffer, and, then, the PEU performs the path extension operation recursively.

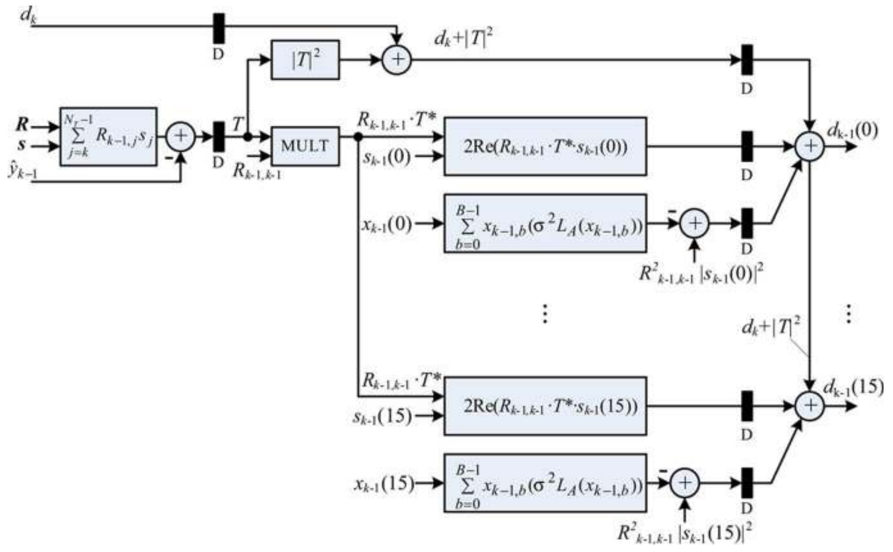

Fig. 11. Block diagram of the partial Euclidean distance calculation unit (PEDCU).

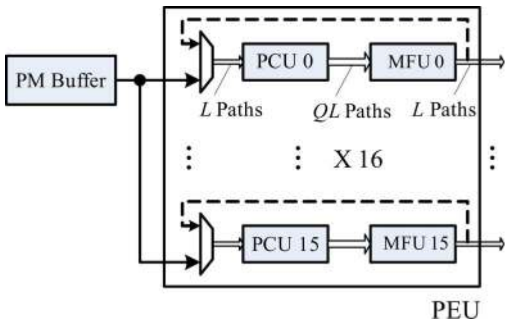

Fig. 12. Block diagram of the path extension unit (PEU).

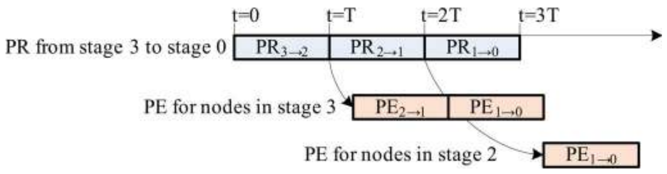

Fig. 13. Detection timeline for the $4 \times 4$ MIMO system.

To illustrate how to employ the PRU and PEU to perform MIMO detection, we show a detection timeline in Fig. 13 for the $4 \times 4$ MIMO system. First, we precompute the initial path metrics $|y_3 - R_3 s_3|^2 - \sigma^2 \sum_{b=0}^{B-1} x_{3,b} \cdot L_A(x_{3,b})$ in the PPU for all 16 nodes in trellis stage 3. Note that the trellis stages are labeled in descending order as shown in Fig. 1. Next, the path reduction (PR) operation is performed for three times, i.e., PR from stage 3 to stage 2 $(\mathrm{PR}_{3\to2})$, PR from stage 2 to stage 1 $(\mathrm{PR}_{2\to1})$, and PR from stage 1 to stage 0 $(\mathrm{PR}_{1\to0})$.

After the path reduction is completed, every node in the last stage, i.e., stage 0, has $L$ full paths through the trellis, which can be used to compute the LLR values for data bits transmitted by antenna 0 based on (12). However, for other nodes, path extension would be required as necessary to fill in the uncompleted paths. For example, as shown in Fig. 13, for nodes in stage 3, two steps of the path extension are required, i.e., PE from stage 2 to stage 1 $(\mathrm{PE}_{2\to1})$, and PE from stage 1 to stage 0 $(\mathrm{PE}_{1\to0})$.

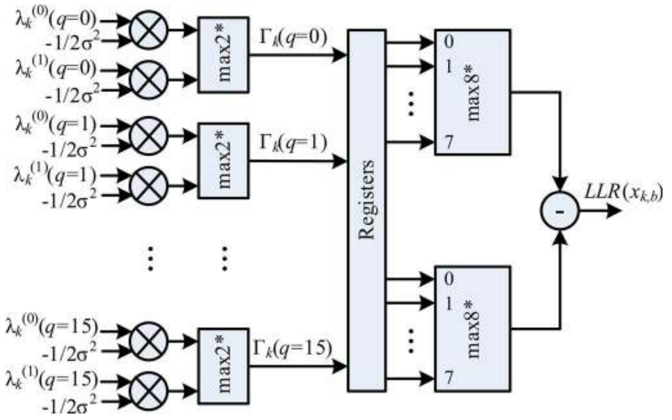

Fig. 14. Block diagram of LLR computation unit (LCU).

TABLE I
QUANTIZATION TABLE FOR $f(|x|) = \ln(1 + \exp(-|x|))$

| $\|x\|$ | $f(\|x\|)$ | $\|x\|$ | $f(\|x\|)$ |
|---|---|---|---|
| $0 \le \|x\| < \frac{12}{64}$ | $\frac{42}{64}$ | $\frac{67}{64} \le \|x\| < \frac{97}{64}$ | $\frac{16}{64}$ |
| $\frac{12}{64} \le \|x\| < \frac{28}{64}$ | $\frac{35}{64}$ | $\frac{97}{64} \le \|x\| < \frac{144}{64}$ | $\frac{10}{64}$ |
| $\frac{28}{64} \le \|x\| < \frac{45}{64}$ | $\frac{29}{64}$ | $\frac{144}{64} \le \|x\| < \frac{288}{64}$ | $\frac{3}{64}$ |
| $\frac{45}{64} \le \|x\| < \frac{67}{64}$ | $\frac{22}{64}$ | $\frac{288}{64} \le \|x\|$ | 0 |

While for nodes in stage 2, one step of the path extension is required, i.e., $\mathrm{PE}_{1\to0}$. For nodes in the last but one stage, i.e., stage 1, no path extension is needed because we can directly retrieve the path metrics from the PM Buffer (these paths were already computed by the PRU). Note that the path reduction operations and the path extension operations can be overlapped to increase the throughput. In this example, assuming each step of the path reduction operation or the path extension operation takes $T$ cycles, we can start the path reduction for the next MIMO symbol as early as $t = 3T$.

## *E. LLR Calculation Unit (LCU)*

The LCU is used to compute the LLR values for each transmitted bit $x_{k,b}$ based on (12). Fig. 14 shows the block diagram of

TABLE II
VLSI IMPLEMENTATION RESULTS AND COMPARISON FOR $4 \times 4$ DETECTORS

| Reference | This Work | [30] | [11] | [34] |
|---|---|---|---|---|
| Algorithm | Soft-Input, Soft-Output Trellis-Search ($L = 2$) | Soft-Input, Soft-Output Depth-First Tree-Search | Soft-Output, K-Best Tree-Search ($K$=5) | Soft-Input, Soft-Output MMSE-PIC |
| Constellation | 16-QAM | 16-QAM | 16-QAM | 2/4/16/64-QAM |
| Iterative Detection | Yes | Yes | No | Yes |
| Clock Frequency | 320 MHz | 250 MHz | 200 MHz | 568 MHz |
| Throughput (1 iter.) | 1.7 Gbps | 55-121 Mbps [a] | 212 Mbps [b] | 1045 Mbps [a] |
| Core Area | 1.58 mm$^2$ | N/A | 0.56 mm$^2$ | 1.3 mm$^2$ [c] |
| Scaled Area | 1.58 mm$^2$ | N/A | 0.14 mm$^2$ [d] | 0.68 mm$^2$ [c] |
| Gate Count | 1097 K | 96 K | 97 K | 359.6 K [c] |
| Technology | 65 nm | 90 nm | 130 nm | 90nm |
| $\frac{\text{Throughput}}{\text{Gate Count}}$ (Mbps/KG) | 1.55 | 0.57-1.26 | 2.19 | 2.91 |

[a] Throughput is scaled by 1.38 to account for the 90 nm technology.
[b] Throughput is scaled by 2 to account for the 130 nm technology.
[c] The preprocessing circuitry (Gram matrix and matched filter) is excluded.
[d] Area is scaled by 0.25 to account for the 130 nm technology.
[e] Area is scaled by 0.52 to account for the 90 nm technology.

LCU. In Fig. 14, we first compute a symbol reliability $\Gamma_k(q)$ for each constellation point $q$ based on (10). Next, the bit LLR value $\mathrm{LLR}(x_{k,b})$ is computed using these $Q$ symbol reliability values based on (12). The $\max^*$ function in Fig. 14 is implemented by a linear maximum (max) function with a correction lookup-table (LUT) based on (11) and Table I. Note that the input path metrics $\lambda_k^{(l)}(q)$ are retrieved from the candidate buffer (cf. Fig. 8), which is used to store the best $L$ path metrics for each trellis node.

## VI. VLSI IMPLEMENTATION RESULTS AND ARCHITECTURE COMPARISON

As a case study, we have developed a trellis-search based iterative MIMO detector ASIC module for a $4 \times 4$ 16-QAM MIMO system. The fixed-point design parameters are summarized as follows. Each element in the $\mathbf{R}$ matrix is scaled by $\frac{1}{\sqrt{10N_t}} = \frac{1}{\sqrt{40}}$, and this scaled $R$ is represented with 11 bits signed data S2.9 (two integer bits with nine fractional bits). The received signal $y$ is represented with 11 bits signed data S5.6. The path metrics (PMs) are rounded to 13 bits between computational blocks. The LLR values are represented with 7 bit signed data S5.2. With this configuration, the fixed-point simulation result shows about 0.1–0.2 dB performance degradation compared to a floating-point detector.

The proposed detector has a pipelined architecture, where the pipeline stages for the PRU and PEU are $T = 4$. To maximize the throughput, we can feed four back-to-back MIMO symbols in four consecutive cycles, e.g., at $t, t+1, t+2, t+3$ into the pipeline to fully utilize the hardware. As shown in Fig. 13, the processing times for the path reduction process and the path extension process are both $3T = 12$ cycles, i.e., the iteration bound is 12 cycles. Thus, we can feed another four back-to-back MIMO symbols into the pipeline at $t+12, t+13, t+14, t+15$, and so forth. Furthermore, we can overlap the path reduction process with the path extension process to hide the processing delay. As a result, the maximum throughput of the detector is $\frac{4\times16\times fclk}{12} = \frac{16}{3} fclk$.

We have described the proposed detector with Verilog HDL and we have synthesized the design for a 1.08 V TSMC 65 nm CMOS technology using Synopsys Design Compiler. With a 320 MHz clock frequency, the proposed detector can achieve a maximum throughput of 1.7 Gb/s. Table II summaries the VLSI implementation results, and it also provides a comparison of the trellis-search based iterative MIMO detector with a state-of-the-art iterative sphere MIMO detector from [30], a noniterative K-best MIMO detector from [11] (we note that no VLSI implementation of an iterative K-best detector was reported in the open literature to the best of our knowledge), and a linear MMSE-PIC based iterative MIMO detector from [34]. In the table, the error-correction code decoder area is not included in the analysis. The sphere detector in [30] has a variable throughput, which will change with the SNR level. For example, in [30], the average number of visited nodes for the $4 \times 4$ 16-QAM system (with 4 iterations) varies from 120 (at 14 dB SNR) to 470 (at 11 dB SNR). In the architecture in [34], a high-throughput flexible detector is proposed based on the linear MMSE-PIC algorithm.

From Table II, one can observe that the proposed detector can achieve a very high data throughput (1.7 Gb/s) while still maintaining a low area requirement (1.58 mm$^2$). The throughput-to-area ratio (measured with $\frac{\text{Throughput}}{\text{Gate Count}}$) of the trellis-search based iterative detector is higher than the tree-search based iterative sphere detector from [30], and is comparable to the noniterative K-best detector from [11]. As expected, the trellis-search based detector consumes more area than the linear MMSE-PIC detector.

In order to develop an iterative detection and decoding system, a high throughput error-correction code decoder would be required as well as some intermediate buffers in between the detector and the decoder. Nowadays, multi-Gb/s decoders, such as LDPC decoders, are feasible with reasonable complexity [35], [36]. Thus, it is very important to develop a high speed detector module to meet the overall throughput requirement of the iterative detection and decoding system. Our proposed detector provides a viable solution for the high-throughput iterative MIMO detection problem as it achieves both high throughput performance and good error performance.

## VII. Conclusion

In this paper, we presented a trellis-search based iterative MIMO detector which can achieve a higher throughput than the traditional tree-search based detectors. The proposed detector employs a path reduction operation in a MIMO trellis where a predefined number of candidates are retained at each trellis node, and a path extension operation where the trellis is extended to fill in the missing paths. The proposed detection algorithm guarantees that a fixed number of path metrics can be found with respect to each hypothesized value of a transmitted bit. As a result, the LLR can be more accurately generated. The advantageous detection performance of the trellis-based detector can be achieved with a small number of paths through the trellis. The trellis-based detector has low complexity and low latency. As the next generation wireless systems are targeting multiple Gb/s data rate, the proposed detector provides a feasible solution for realizing such a high data rate system.

## References

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

[2] 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.

[3] 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.

[4] 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, 2003.

[5] B. Hochwald and S. Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, pp. 389–399, Mar. 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] 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. 1–14, Mar. 2010.

[9] 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.*, May 2002, vol. 3, pp. 273–276.

[10] 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)*, Sep. 2004, vol. 2, pp. 1142–1148.

[11] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, pp. 491–503, Mar. 2006.

[12] 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.*, Sep. 2006, pp. 1151–1154.

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

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

[15] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, pp. 1566–1577, Jul. 2005.

[16] 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.

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

[18] 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. Embedded Syst.*, vol. 2007, no. 3, pp. 1–14, 2007.

[19] 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.

[20] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, pp. 290–300, Feb. 2008.

[21] M. Myllyla, M. Juntti, and J. R. Cavallaro, "Architecture Design and Implementation of the Increasing Radius—List Sphere Detector Algorithm," Apr. 2009, pp. 553–556.

[22] 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, pp. 328–337, Mar. 2007.

[23] 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.*, Mar. 2008, pp. 3173–3176.

[24] 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., Architectures, Modeling, Simulat.*, Jul. 2009, pp. 20–23.

[25] 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, Oct. 2010.

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

[27] 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.

[28] C. Mehlfhrer, D. Seethaler, G. Matz, and M. Rupp, "An iterative MIMO-HSDPA receiver based on a K-best-MAP algorithm," in *Proc. IEEE Global Telecommun. Conf. (IEEE GLOBECOM)*, Nov. 2006, pp. 1–5.

[29] H. Kim, D.-U. Lee, and J. D. Villasenor, "Design tradeoffs and hardware architecture for real-time iterative MIMO detection using sphere decoding and LDPC coding," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 6, pp. 1003–1014, Aug. 2008.

[30] E. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr, "A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 9, pp. 706–710, Sep. 2010.

[31] Y. Sun and J. R. Cavallaro, "Low-complexity and high-performance soft MIMO detection based on distributed M-algorithm through trellis-diagram," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2010, pp. 3398–3401.

[32] Y. Sun and J. R. Cavallaro, "High-throughput soft-output MIMO detector based on path-preserving trellis-search algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, pp. 1–13, May 2011, vol..

[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.*, Dec. 2008, pp. 830–833.

[34] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE J. Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, Jul. 2011.

[35] Y. Sun and J. R. Cavallaro, "Multi-layer parallel decoding algorithm and VLSI architecture for quasi-cyclic LDPC codes," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 1776–1779.

[36] T. Mohsenin and B. M. Baas, "High-throughput LDPC decoders using a multiple split-row method," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2007, pp. II-13–II-16.

**Yang Sun** (S'07–M'11) received the B.S. degree in testing technology and instrumentation and the M.S. degree in instrument science and technology, both 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.

From 2010 to 2011, he was with Broadcom Corporation, Sunnyvale, CA. In 2011, he joined Qualcomm Incorporated, San Diego, CA. 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 received 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/IEEE GLSVLSI Best Student Paper Award. He has served on the Technical Committee for the 2011 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP) and the 2011 ACM/IEEE GLSVLSI conference. He was Session Co-Chair for the 2012 ACM/IEEE GLSVLSI conference.

**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 at 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), and Program Co-Chair for the 2012 ACM/IEEE GLSVLSI.