Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2013, Article ID 496013, 12 pages
http://dx.doi.org/10.1155/2013/496013

Research Article

Fully Pipelined Implementation of Tree-Search Algorithms for Vector Precoding

Maitane Barrenechea, Mikel Mendicute, and Egoitz Arruti

Department of Electronics and Computer Science, University of Mondragon, 20500 Mondragon, Spain

Correspondence should be addressed to Maitane Barrenechea; mbarrenetxea@mondragon.edu

Received 13 February 2012; Revised 31 July 2012; Accepted 15 January 2013

Academic Editor: João Cardoso

The nonlinear vector precoding (VP) technique has been proven to achieve close-to-capacity performance in multiuser multiple-input multiple-output (MIMO) downlink channels. The performance benefit with respect to its linear counterparts stems from the incorporation of a perturbation signal that reduces the power of the precoded signal. The computation of this perturbation element, which is known to belong in the class of NP-hard problems, is the main aspect that hinders the hardware implementation of VP systems. To this respect, several tree-search algorithms have been proposed for the closest-point lattice search problem in VP systems hitherto. Nevertheless, the optimality of these algorithms has been assessed mainly in terms of error-rate performance and computational complexity, leaving the hardware cost of their implementation an open issue. The parallel data-processing capabilities of field-programmable gate arrays (FPGA) and the loopless nature of the proposed tree-search algorithms have enabled an efficient hardware implementation of a VP system that provides a very high data-processing throughput.

1. Introduction

Since the presentation of the vector precoding (VP) technique [1] for data transmission over the multiuser broadcast channel, many algorithms have been proposed in the literature to replace the computationally intractable exhaustive search defined in the original description of the algorithm. To this respect, lattice reduction approaches have been widely used as a means to compute a suboptimum perturbation vector with a moderate complexity. The key idea of lattice-reduction techniques relies on the usage of an equivalent and more advantageous set of basis vectors to allow for the suboptimal resolution of the exhaustive search problem by means of a simple rounding operation. This method is used in [2], where the Lenstra-Lenstra-Lovász (LLL) reduction algorithm [3] is used to yield the Babai's approximate closest-point solution [4]. Similar approaches can be found in [5–7]. Despite achieving full diversity order in VP systems [8, 9], the performance degradation caused by the quantization error due to the rounding operation still remains. Moreover, many lattice reduction algorithms have a considerable computational complexity, which poses many challenges to a prospective hardware implementation.

An appropriate perturbation vector can also be found by searching for the optimum solution within a subset of candidate vectors. These approaches, also known as tree-search techniques, perform a traversal through a tree of hypotheses with the aim of finding a suitable perturbation vector.

In spite of the high volume of research work published around the topic of precoding algorithms, the issues raised by their implementation have not been given the same attention. Some of the few publications on this area, such as [10–12], describe precoding systems that either have a considerable complexity in terms of allocated hardware resources or provide a rather low data transmission rate.

Despite the lack of published research in the area of hardware architectures for precoding algorithms, the implementation issues of tree-search schemes in MIMO detection scenarios have been widely studied. For example, the field programmable gate array (FPGA) implementation of the fixed-complexity sphere decoder (FSD) detector has been analyzed in [13–15], whereas the hardware architecture of the K-best tree search considering a real equivalent model was researched in [16–21]. Moreover, the implementation of a K-best detector with suboptimum complex-plane enumeration was performed in [22, 23]. A thorough review of K-best tree-search implementation techniques was carried out in [24]. The adaptation of these tree-search schemes to precoding systems implies many variations with respect to the original description of the algorithms. Even if many lessons can be learned from the hardware architecture of tree-search techniques for point-to-point MIMO systems, the peculiarities of the precoding scenario render the results of the aforementioned publications inadequate for the current research topic.

Consequently, this contribution addresses the high-throughput implementation of fixed-complexity tree-search algorithms for VP systems. More specifically, two state-of-the-art tree-search algorithms that allow for the parallel processing of the tree branches have been implemented on a Xilinx Virtex VI FPGA following a rapid-prototyping methodology. In order to achieve a high throughput, both schemes operate in the complex plane and have been implemented in a fully-pipelined fashion providing one output per cycle.

This contribution is organized as follows: in Section 2 the system model is introduced, followed by a short review provided in Section 3 on the noniterative tree-search algorithms to be implemented. Next, the general hardware architecture of the data perturbation process is outlined in Section 4 whereas the specific features of the K-best and FSE modules are analyzed in Sections 5 and 6, respectively. An analysis on the tree-search parameters for both techniques is performed in Section 7 and the hardware implementation results are shown in Section 8. Finally, some concluding remarks are drawn in Section 9.

Notation. In the remainder of the paper the matrix transpose and conjugate transpose are represented by $(\cdot)^T$ and $(\cdot)^H$, respectively. We use $\mathbf{I}_N$ to represent the $N \times N$ identity matrix and Mod$(\cdot)$ to denote the modulo operator. The set of Gaussian integers of dimension N, namely, $\mathbb{Z}^N + j\mathbb{Z}^N$, is represented by $\mathbb{CZ}^N$.

2. System Model

Consider the multiuser broadcast channel with M_T antennas at the transmitter and N single-antenna users, denoted as $M_T \times N$ where $M_T \geq N$. We assume that the channel between the base station and the N users is represented by a complex-valued matrix $\mathbf{H} \in \mathbb{C}^{N\times M_T}$, whose element $h_{n,m}$ represents the channel gain between transmit antenna m and user n. For all simulations, the entries of the channel matrix are assumed independent and identically distributed with zero-mean circularly symmetric complex Gaussian distribution and $E[|h_{n,m}|^2] = 1$.

The precoding system under study is shown in Figure 1. According to the aforementioned model, the data received at

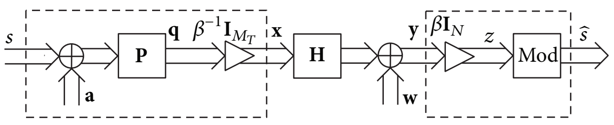


Figure 1: Block diagram of a VP system.

the N user terminals can be collected in the vector $\mathbf{y} \in \mathbb{C}^N$, which is given by

$$\mathbf{y} = \mathbf{Hx} + \mathbf{w}, \tag{1}$$

where $\mathbf{w} \in \mathbb{C}^N$ represents the additive white Gaussian noise vector with covariance matrix $E[\mathbf{ww}^H] = \sigma^2\mathbf{I}_N$.

In precoding systems such as the one depicted in Figure 1, the independent data acquisition at the receivers is enabled by a preequalization stage at the base station. This procedure, which is carried out by means of a precoding filter $\mathbf{P}$, anticipates the distortion caused by the channel matrix in such a way that the received signal is (ideally) fully equalized at the receive terminals. However, the precoding process causes variations in the power of the user data streams, and therefore, a power scaling factor $\beta^{-1} = \sqrt{E_{\text{Tr}}/E[\|\mathbf{q}\|^2]} \in \mathbb{R}_+$ is applied to the vector of precoded symbols $\mathbf{q}$ prior to transmission to ensure a certain transmit power E_{Tr}. At the user terminals, the received signal is scaled by β again to allow for an appropriate detection of the data symbols. Hence, the signal prior to the detection stage reads as

$$\mathbf{z} = \mathbf{Hq} + \beta\mathbf{w}. \tag{2}$$

From this equation, one can notice that in the event of $E[\|\mathbf{q}\|^2] > E_{\text{Tr}}$, or equivalently $\beta > 1$, an increase in the power of the noise vector is experienced at the receivers, which greatly deteriorates the error-rate performance of the system. To this respect, nonlinear signal processing approaches aim at reducing the power of the linearly precoded symbols, in such a way that a considerable performance enhancement can be attained. In VP systems, this objective is achieved by incorporating a perturbation signal prior to the linear precoding stage.

The data perturbation process is supported by the modulo operator at the receivers, which provides the transmitter with additional degrees of freedom to choose the perturbation vector that is most suitable. Note that the perturbation signal must be composed of integer multiples of the modulo constant τ, namely, $\mathbf{a} \in \mathcal{Q}$ with $\mathcal{Q} = \tau\mathbb{CZ}^N$, so that it can be easily removed at the receivers by means of a simple modulo operation.

The VP system model that achieves the best error-rate performance targets the minimization of the mean square error (MSE) instead of the traditional goal of reducing the average power of the transmitted symbols [25]. In this model, the precoding matrix is designed as $\mathbf{P} = \mathbf{H}^H\mathbf{\Psi}$, with $\mathbf{\Psi} = (\mathbf{HH}^H + N\sigma^2/E_{\text{Tr}}\mathbf{I}_N)^{-1}$, and the triangular matrix used for the computation of the perturbation signal is computed as

$\mathbf{U}^H\mathbf{U} = \mathbf{\Psi}$. Finally, the optimum perturbation vector is obtained by evaluating the following cost function:

$$\mathbf{a} = \arg \min_{\hat{\mathbf{a}} \in \tau \mathbb{C}\mathbb{Z}^N} \|\mathbf{U}(\mathbf{s} + \hat{\mathbf{a}})\|_2^2. \quad (3)$$

The computation of the perturbing signal in (3) entails a search for the closest point in a lattice. Several techniques to efficiently obtain the perturbation signal will be analyzed in the following section.

3. Tree-Search Techniques for Vector Precoding

The triangular structure of the matrix $\mathbf{U}$ in (3) enables the gathering of all the solution vector hypotheses in an organized structure which resembles the shape of a tree. Following the analogy with the tree structure, the concatenation of $\psi < N$ lattice elements or nodes is referred to as a branch, where a branch of length N represents a candidate solution vector. The search for the perturbation vector is then performed by traversing a tree of N levels (each one representing a user) starting from the root level $i = N$, and working backwards until $i = 1$.

Since the elements of the solution vector belong to the expanded search space $\mathcal{Q}$, the amount of nodes that originate from each parent node in the tree equals $|\mathcal{Q}| = \infty$ in theory. However, depending on the tree-traversal strategy to be followed, the cardinality of this set can be reduced either artificially, by limiting the search space to the group $\mathcal{L}$ of closest points to the origin, or by identifying the set of eligible nodes following a distance control policy (also known as the sphere constraint). Note that, as opposed to the point-to-point MIMO detection scenario, the amount of child nodes that stem from the same parent node does not depend on the modulation constellation in use. This way, the computation of the (squared) Euclidean distances in (3) can be distributed across multiple stages as follows:

$$D_i = u_{ii}^2 |a_i + z_i|^2 + \sum_{j=i+1}^{N} u_{jj}^2 |a_j + z_j|^2 = d_i + D_{i+1}, \quad (4)$$

where

$$z_i = s_i + \sum_{j=1}^{i-1} \frac{u_{ij}}{u_{ii}} (a_j + s_j). \quad (5)$$

The partial Euclidean distance (PED) associated with a certain node at level i is denoted as d_i, while the accumulated Euclidean distance (AED) down to level i is given by $D_i = \sum_{j=i}^{N} d_j$. Since the elements of the solution vector belong to the expanded search space $\mathcal{Q}$, the amount of nodes that originate from each parent node in the tree equals $|\mathcal{Q}| = \infty$ in theory. However, depending on the tree-traversal strategy to be followed, the cardinality of this set can be reduced either artificially, by limiting the search space to the group $\mathcal{L}$ of $|\mathcal{L}|$ closest points to the origin, or by identifying the set of eligible nodes following a distance control policy (also known as the sphere constraint).

The traversal of the search tree is usually performed following either a depth-first or breadth-first strategy.

3.1. Depth-First Tree-Search Techniques

3.1. Depth-First Tree-Search Techniques. Depth-first tree-search techniques traverse the tree in both forward and backward directions enabling the implementation of a sphere constraint for pruning of unnecessary nodes based on, for example, the Euclidean distance associated with the first computed branch. The pruning criterion, which is updated every time a leaf node at level $i = 1$ with a smaller AED is reached, does not impose a per-level run-time constraint, and therefore, the complexity of these algorithms is of variable nature.

One of the most noteworthy depth-first techniques is the SE algorithm [26, 27], which restricts the search for the perturbation vector to the set of nodes with $D_i \leq R$ that lie within a hypersphere of radius R centered around a reference signal. The good performance of the algorithm is a consequence of the identification and management of the admissible set of nodes at each stage of the tree search. Every time a forward iteration is performed ($i \rightarrow i-1$) the algorithm selects and computes the distance increments of the nodes that fulfil the sphere constraint and continues the tree search with the most favorable node according to the Schnorr-Euchner enumeration [28] (the node resulting in the smallest d_i). This process is repeated until a leaf node is reached (which will result in a radius update) or no nodes that satisfy the sphere constraint are found. In any case, the SE will proceed with a backward iteration ($i \rightarrow i+1$) where a radius check will be performed among the previously computed set of candidate points. If a node with $D_i < R$ is found, the tree traversal is resumed with a forward iteration. The optimum solution has been found when the hypersphere with the updated radius contains no further nodes.

The radius reduction strategy along with the tracking of potentially valid nodes at each level of the algorithm prevents unnecessary distance computations but ultimately results in a rather complex tree-search hardware architecture.

3.2. Breadth-First Tree-Search Techniques. Breadth-first tree-search algorithms with upper-bounded complexity traverse the tree in only forward direction, identifying a set of potentially promising nodes and expanding only these in the subsequent tree-levels. These algorithms benefit from a fixed and high data-processing throughput that stems from the parallel processing of the branches. Nevertheless, the speculative pruning carried out during the tree search prevents bounded breadth-first algorithms from achieving an optimum performance.

As one can guess from its name, the K-best precoder [29, 30] selects the K best branches at each level of the tree regardless of the sphere constraint or any other distance control policy. At each stage i of the K-best tree search, an ordering procedure has to be performed on the eligible $K|\mathcal{L}|$ candidate branches based on their AEDs down to level i. After the sorting procedure, the K paths with the minimum accumulated distances are passed on to the next level of the tree. Once the final stage of the tree has been

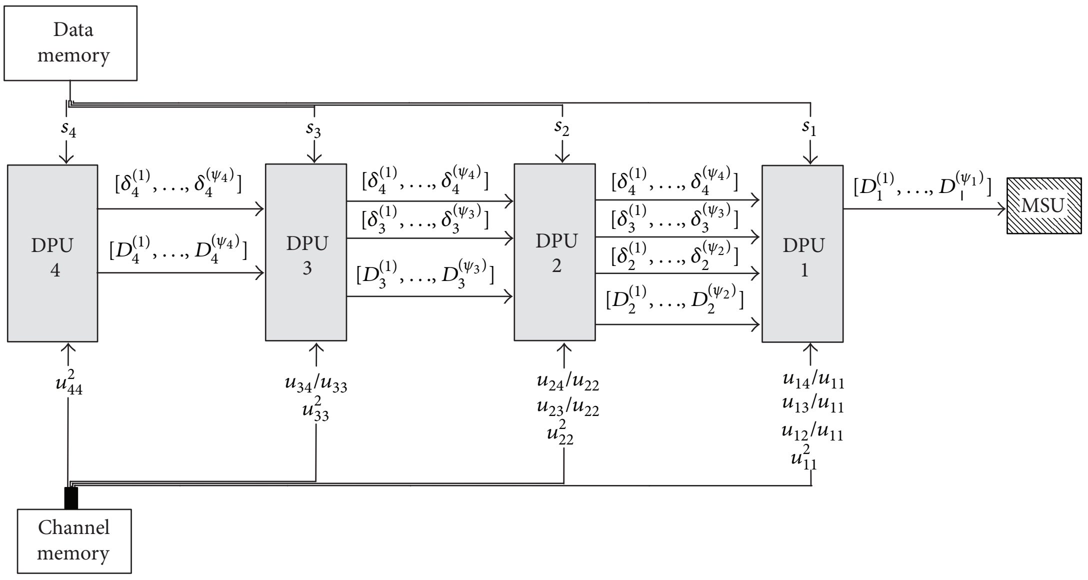

Data memory
s_4
s_3
s_2
s_1
DPU 4
DPU 3
DPU 2
DPU 1
MSU
$[\delta_4^{(1)}, \ldots, \delta_4^{(\psi_4)}]$
$[D_4^{(1)}, \ldots, D_4^{(\psi_4)}]$
$[\delta_3^{(1)}, \ldots, \delta_3^{(\psi_3)}]$
$[D_3^{(1)}, \ldots, D_3^{(\psi_3)}]$
$[\delta_2^{(1)}, \ldots, \delta_2^{(\psi_2)}]$
$[D_2^{(1)}, \ldots, D_2^{(\psi_2)}]$
$[D_1^{(1)}, \ldots, D_1^{(\psi_1)}]$
u_{44}^2
u_{34}/u_{33}
u_{33}^2
u_{24}/u_{22}
u_{23}/u_{22}
u_{22}^2
u_{14}/u_{11}
u_{13}/u_{11}
u_{12}/u_{11}
u_{11}^2
Channel memory

Figure 2: General hardware architecture of the fixed-complexity tree-search techniques for an $N = 4$ user system.

reached, the branch with the minimum Euclidean distance is selected as the K-best solution. Clearly, the main bottleneck in this scheme stems from the node ordering and selection procedures performed at every level of the tree search.

The fixed-complexity sphere encoder (FSE) was presented in [31] as a sort-free alternative to the aforementioned K-best precoder. The proposed scheme avoids the intricate sorting stages required by the K-best by defining a node selection procedure based on a tree configuration vector $\mathbf{n} = [n_1, \ldots, n_N]$. This vector specifies the number of child nodes to be evaluated at each level (n_i) following the Schnorr-Euchner enumeration. Therefore, only n_i PEDs are computed per parent node at each level, yielding a total candidate branch count of $n_T = \prod_{i=1}^{N} n_i$.

Both fixed-complexity algorithms achieve a high data-processing throughput due to their capability of parallel branch computation. This high-speed data-processing feature will be assessed by carrying out the hardware implementation of a 4×4 vector precoder based on the fixed-complexity algorithms under study. The main objectives of this study are twofold: on one hand, the quantification of the data-transmission throughput of the proposed architectures, and on the other hand, the assessment of the hardware resource allocation required for their implementation.

4. General Architecture Overview

Both tree-search schemes share the same general distance computation structure, as can be seen in Figure 2. The lack of loops in the hardware architecture of the fixed-complexity tree-search techniques enables a high throughput and fully-pipelined implementation of the data perturbation process, thus being its implementation specially suitable for a target FPGA device.

The AEDs of the candidate branches are computed by accumulating the PEDs calculated at the local distance processing units (DPUs) to the AEDs of the previous level. This way, the AEDs down to level i corresponding to the considered candidate branches, namely, $[D_i^{(1)}, \ldots, D_i^{(\psi_i)}]$, are passed on from DPU i to DPU $i-1$. The parameter ψ_i stands for the number of candidate branches at each level of the tree search, being it $\psi_i = K$ for all i for the K-best and $\psi_i = \prod_{j=i}^{N} n_j$ for the FSE model.

Two input memory blocks, named Data Memory and Channel Memory, have been included to store the data symbols and the values of the triangular matrix $\mathbf{U}$, respectively. The off-diagonal matrix coefficients are stored as u_{ij}/u_{ii}, whereas the diagonal values are in the form of u_{ii}^2 to simplify the calculation of (4) and (8). Note that the matrix preprocessing stage required by the FSE and K-best approaches has not been included in the hardware design. The computation of the intermediate points z_i requires the values of all previous $\delta_j = a_j + s_j$. To avoid redundant calculations, the set of values $[\delta_j^{(1)}, \ldots, \delta_j^{(\psi_j)}]$ for all $j > i$ is transferred to DPU i, as is shown in Figure 2.

The hardware structure of the first DPU is common in both schemes. The computation of the Euclidean distances in this level does not involve any data from previous levels, and therefore, the only operation to be performed is to select the ψ_N lattice values closest to s_N and to compute the corresponding PEDs. Given that the position of the modulation's constellation within the complex lattice is known beforehand, and considering the symmetries of the complex lattice, it is possible to select the nodes to be passed on to the next level without performing any extra distance calculations and sorting procedures. Additionally, the hardware structure of the last DPU is also equal for both algorithms, as only the most favorable child node that stems from each one of the ψ_2 parent nodes needs to be expanded at this level. Such a task can be performed by simply rounding the value of z_1 to the position of the nearest lattice point.

The main and crucial differences between the FSE and K-best tree-search algorithms rely on the DPUs of levels $1 < i < N$.

5. DPU for the K-Best

The difficulty of performing the sorting procedure in the complex plane, where the amount of nodes to be considered is higher, and the intricacy of complex-plane enumeration have led to the dominance of real-valued decomposition (RVD) as the preferred technique when implementing the K-best tree search. Nevertheless, direct operation on the complex signals is preferred from an implementation point of view as the length of the search tree is halved, and hence, the latency and critical path of the design can be shortened.

5.1. Structure of the Sorting Stage. Regardless of the domain of the signals to be used, the bottleneck in this type of systems is usually the sorting stage performed at each tree level. The number of child nodes that stem from the same parent node will be defined as B, being its value $B = |\mathscr{L}|$ for the complex-plane model, whereas $B = \sqrt{|\mathscr{L}|}$ will be required for the RVD scheme. The PED calculation and subsequent sorting procedure on the KB child nodes at each level is a computationally expensive process that compromises the throughput of the whole system. With the aim of alleviating the burden of the sorting stage, the use of the Schnorr-Euchner ordered sequence of child nodes and the subsequent merging of the sorted sublists is proposed in [17]. Even if the proposed scheme is implemented on an RVD model due to the simplicity of the local enumeration, it is possible to extend it to the complex plane if a low-complexity enumerator, such as the puzzle enumerator presented in [32], is utilized. Additionally, a fully-pipelined RVD architecture of the sorted sublists algorithm is proposed in [33] for high-throughput systems. By dividing the real axis into $2B$ regions and storing the corresponding enumeration sequences in look-up tables (LUTs), the algorithm is able to determine the child node order by means of a simple slicing procedure. Nevertheless, this technique is advantageous only when operating with RVD symbols as the amount of data to be stored and the quantity of nonoverlapping regions grow remarkably when complex-valued symbols are utilized. In any case, the use of any of the aforementioned sublist merging approaches reduces the amount of PED computations to be performed at each level to $K^2 \leq KB$.

5.1.1. The Winnersec Path Extension Algorithm. The number of costly distance computations can be further reduced by implementing the winner path extension (WPE) selection approach presented in [34] and incorporated into the RVD hardware architecture of the K-best tree search in [35, 36]. The proposed scheme selects the K most favorable branches in K iterations by performing just $2K - 1$ PED computations. An illustrative example of the WPE algorithm is depicted in Figure 3 for a system with $B = 4$ and $K = 3$. The child nodes at a certain tree level i are tagged as $a_i^{(x,y)}$, where x represents the index of the parent node and y denotes the position of that certain node within the ordered Schnorr-Euchner sequence of child nodes.

The WPE sorting procedure is based on the generation and management of a node candidate list $\mathscr{A}$. This way, the child node corresponding to the kth most favorable branch is extracted from the candidate list of the kth sorting stage $\mathscr{A}_k$. The initial values in the candidate list are comprised of the AEDs down to level i of the best child nodes that stem from each one of the K parent nodes, which gives $\mathscr{A}_1 = \{a_i^{(1,1)}, \ldots, a_i^{(K,1)}\}$. The winner branch in the initial sorting stage, or equivalently the first of the K most favorable branches, is selected as the branch with the smallest AED within $\mathscr{A}_1$. The PED of the second most favorable child node that stems from the same parent node as the latest appointed winner branch is computed ($a_i^{(3,2)}$ in the example illustrated in Figure 3) and the AED of the resulting tree branch is added to the candidate list $\mathscr{A}_2$. The algorithm proceeds accordingly until the K required branches have been identified.

Additionally note that, according to the complexity analysis based on the amount of comparisons in a WPE-enabled K-best tree search published in [35], the complexity of an RVD-based tree traversal doubles that of its complex-plane counterpart.

5.2. Structure of the K-best DPU. The structure of the proposed K-best DPU is depicted in Figure 4 for a system with $K = 3$. The branch selection procedure is carried out in K fully-pipelined sorting stages following a modified version of the WPE algorithm presented in [33, 34]. First of all, the computation of the intermediate points is performed for each one of the K branches that are passed on from the previous level. The set of best child nodes that stem from each parent node can be computed by simply rounding off the value of the intermediate point to the nearest lattice point. The distance increments (d_i in (4)) for those K-best children are computed by K metric computation unit (MCU) and are accumulated with their corresponding D_{i-1} values. These distance values and their corresponding branches comprise the candidate list Λ_1. The minimum AED within Λ_1 is found at the minimum-search unit (MSU) by simple concatenation of compare-and-select blocks. The MSU also outputs the index of the first winner branch $\alpha_1 \in \{1, \ldots, K\}$ so that the appropriate value of z_i can be selected for the local enumeration procedure.

At the second stage of the sorting procedure, the $a_i^{(\alpha_1,2)}$ node needs to be identified for any parent node index α_1. This task is performed by the E_2 block, which comprises a puzzle enumerator that outputs the second most favorable node given a certain value of z_i. However, in the subsequent stages of the algorithm, the enumeration procedure will depend on the index of the previously appointed winner branches. Hence, if $\alpha_1 = \alpha_2$, the third most promising child node will need to be expanded, namely, $a_i^{(\alpha_1,3)}$, whereas the second most favorable node in the α_2 branch ($a_i^{(\alpha_2,2)}$) will be required if $\alpha_1 \neq \alpha_2$. Consequently, the new candidate branch to be included in the $\mathscr{A}_k$ candidate list at the kth sorting stage will

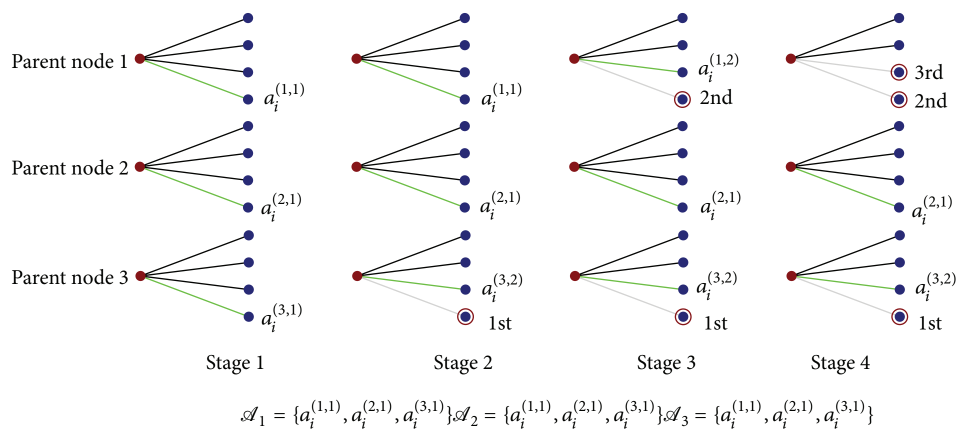

Parent node 1
Parent node 2
Parent node 3
Stage 1
Stage 2
Stage 3
Stage 4
1st
2nd
3rd
$\mathscr{A}_1 = \{a_i^{(1,1)}, a_i^{(2,1)}, a_i^{(3,1)}\}$ $\mathscr{A}_2 = \{a_i^{(1,1)}, a_i^{(2,1)}, a_i^{(3,1)}\}$ $\mathscr{A}_3 = \{a_i^{(1,1)}, a_i^{(2,1)}, a_i^{(3,1)}\}$

Figure 3: Illustrative example of the WPE distributed sorting algorithm for a system with $K = 3$ and $B = 4$.

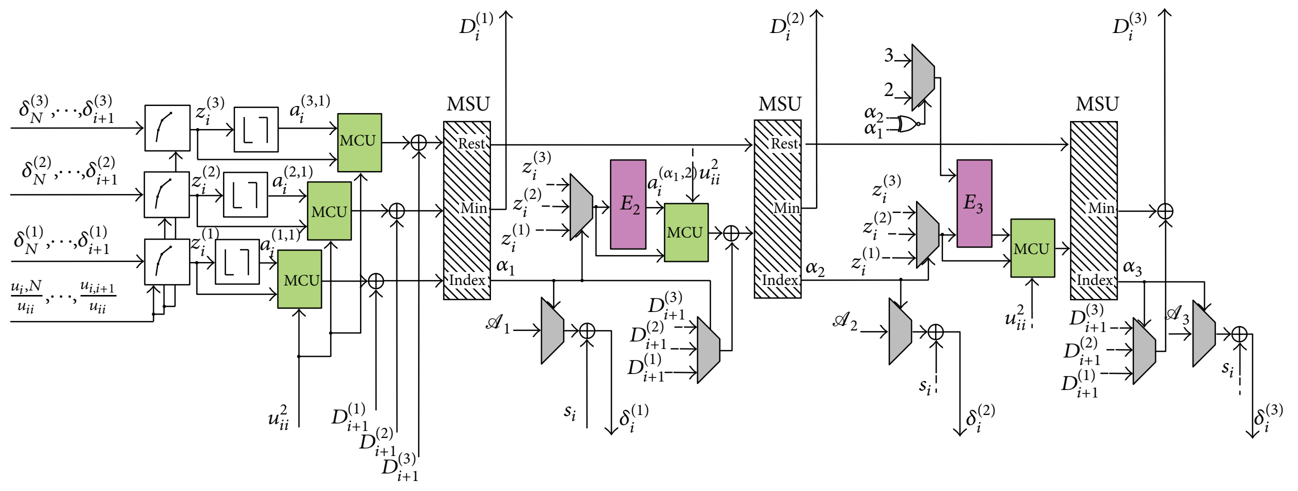


Figure 4: Block diagram of the K-best DPU.

require the expansion of the ρth most favorable child node, where ρ may take any value within the set $\{2, 3, \ldots, k\}$.

The enumeration approach at each sorting stage k has been carried out by means of a puzzle enumerator unit capable of ascertaining the optimum ordered sequence of the first k child nodes in a nonsequential fashion. The node order determination in the puzzle enumerator can be carried out without performing any costly distance computations. For each sorting stage k any node in the ordered sequence of best k child nodes can be selected for expansion. This way, the desired child node in the ordered sequence is determined by an additional input variable which keeps track of the amount of already expanded child nodes for each parent node. The puzzle enumerator has been selected as the enumeration scheme to be used along with the WPE due to its lower hardware resource demand and nonsequential nature, as discussed in [32]. Note that, there are no feedback loops in the structure of the K-best DPU, and therefore, it is possible to implement it following a fully-pipelined scheme.

6. DPU for the FSE

The intricate node ordering and selection procedure required by the K-best algorithm is replaced by a simple Schnorr-Euchner enumerator in the FSE tree-search model. This derives in a considerably simpler DPU architecture of the FSE scheme.

Figure 5 depicts the structure of the FSE DPU, where the block diagram for $n_T^{(i)} = \prod_{j=1}^{i-1} n_j = 3$ and $n_i = 1$ is represented. First of all, the data of the $\{\delta_{i+1}, \ldots, \delta_n\}$ values transferred from level $i+1$ are used to compute the intermediate values z_i for each one of the parent nodes. Afterwards, the node selection procedure is performed by means of a simple rounding operation when $n_i = 1$, as depicted in the illustrative example in Figure 5, or by means of the unordered puzzle enumerator [32] for the cases where $n_i > 1$. The PEDs of the selected nodes are then computed by $n_T^{(i)}$ MCUs and accumulated to the AEDs from the previous level. Finally, as was the case with the K-best DPU, the

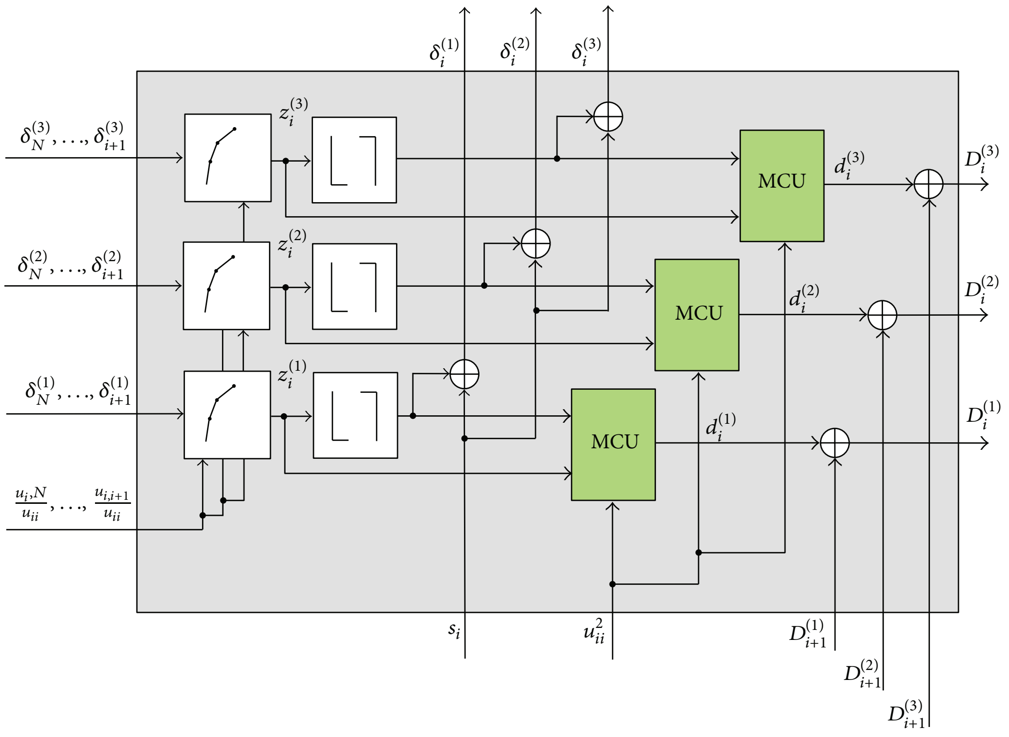


Figure 5: Block diagram of the FSE DPU.

FSE DPU does not have any feedback paths in its design, and hence, it can be easily implemented following a fully-pipelined scheme.

7. Design Considerations

This section addresses the design parameter selection for the fixed-complexity algorithms to be implemented in hardware. Additionally, the impact of applying an approximate norm for the computation of the distance increments is studied from an error-rate-performance point of view.

7.1. Choice of the Design Parameters. The configurable parameters K and n_T offer a flexible trade-off between performance and complexity for the K-best and FSE encoders, respectively. These configuration parameters establish the shape of the search tree, which in turn determines the amount of hardware multipliers required for its implementation. Embedded multipliers are scarce in FPGA devices and are considered an expensive resource in application-specific integrated circuit (ASIC) designs. Thus, the number of multiplication units required by the tree-search algorithm has been regarded as the critical factor in the current hardware architecture design. For the sake of a fair comparison, the configuration parameters of the fixed-complexity tree-search methods have been selected so as to yield a similar amount of allocated embedded multipliers.

Considering that 3 multipliers are used for the multiplication of two complex terms, the number of multiplication units required for the K-best tree-search structure can be computed as

$$N_{\mathrm{MUL},KB} = 6K + 3(N-2)(2K-1) + 3K\left[\frac{N(N-1)}{2}\right], \tag{6}$$

whereas the total amount of embedded multipliers for an FSE tree structure is given by

$$N_{\mathrm{MUL,FSE}} = 3\sum_{i=1}^{N} i n_T^{(i)}. \tag{7}$$

The number of required embedded multipliers for the K-best and FSE tree-search techniques is shown in Figure 6 for a system with $N = 4$ single-antenna users. The amount of multiplier units is given as a function of the number of candidate branches, namely, K and n_T for the K-best and FSE approaches, respectively. As one can notice, the amount of hardware resources in the K-best tree-search model grows linearly with the number of considered candidate branches. However, this constant growth rate does not apply for the FSE case. This is due to the n_i values being differently distributed through the tree configuration vector depending on the divisibility of n_T. As proved in [37], among all possible tree configuration vectors that yield the same value of n_T, the one with the most dispersedly distributed values of n_i achieves the best error-rate performance and requires the lowest amount of allocated embedded multipliers.

In order to assess the hardware resource occupation required for the the implementation of the different tree-search algorithms, the design parameter values $K = 7$

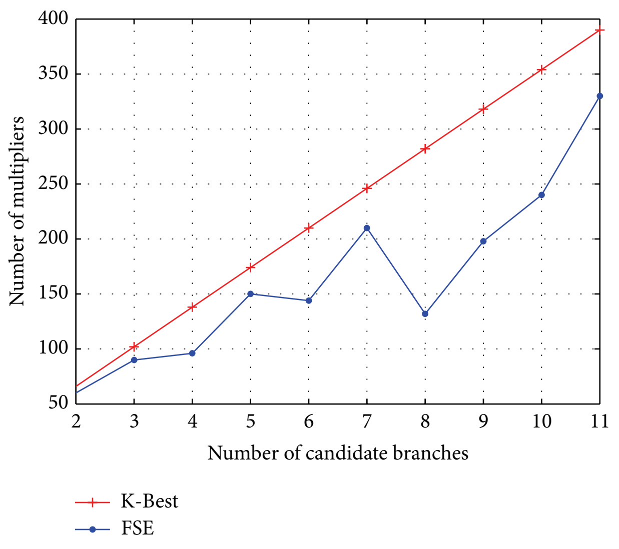


FIGURE 6: Number of required multipliers for the K-best and FSE tree-search techniques as a function of the number of candidate branches.

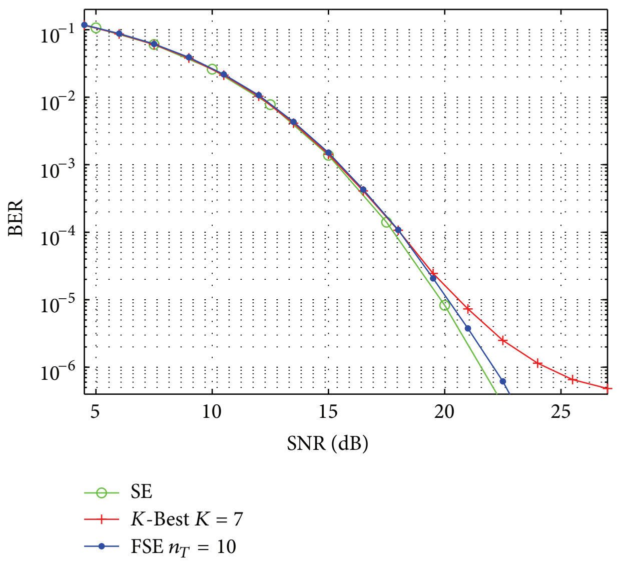


FIGURE 7: BER performance of the implemented FSE and K-best tree-search structures.

and $n_T = 10$ have been selected for the K-best and FSE models, respectively. This choice of parameters ensures a similar multiplier occupation for both schemes and offers a significantly better error-rate performance than other lower K and n_T pairs, for example, $K = 6$ and $n_T = 9$.

The BER versus SNR curves of the implemented fixed-complexity schemes are depicted in Figure 7. As one can notice, the error-rate performance of the implemented models is close to the optimum set by the SE in the low-to-mid SNR range. However, a performance degradation of 0.5 dBs is noticeable for the FSE model at high-SNRs, whereas the performance gap of the K-best structure increases with the SNR, reaching up to 3 dBs at a BER of 10^{-6}.

7.2. Implementation of an Approximate Norm. A significant portion of the hardware resources in the implementation of any tree-search algorithm is dedicated to computing the ℓ^2 norms required by the cost function in (3). Additionally, the long delays associated with squaring operations required to compute the PEDs account for a significant portion of the latency of the fixed-complexity tree-search architectures. It is possible to overcome these problems by using an approximate norm that prevents the use of the computationally expensive squaring operations.

The application of the modified-norm algorithm (MNA) [38] entails two main benefits: on one hand, a simplified distance computation scheme that immediately reduces silicon area and delay of the arithmetic units can be performed, and on the other hand, a smaller dynamic range of the PEDs is achieved. The key point of the MNA is to compute the square root of the accumulated and partial distance increments, namely, $E_i = \sqrt{D_i}$ and $e_i = \sqrt{d_i}$, respectively. Hence, the accumulation of the distance increments in this equivalent model gives $E_i = \sqrt{E_{i+1}^2 + e_i^2}$. An approximate norm can now be applied to get rid of the computationally expensive squaring and square root operations, such that $E_i \approx f(|E_{i+1}|, |e_i|)$. This way, the accumulated distance computation in (4) can be reformulated as

$$E_i = E_{i+1} + e_i, \tag{8}$$

with

$$e_i = u_{i,i} \left(\left| \Re (a_i + z_i) \right| + \left| \Im (a_i + z_i) \right| \right) \tag{9}$$

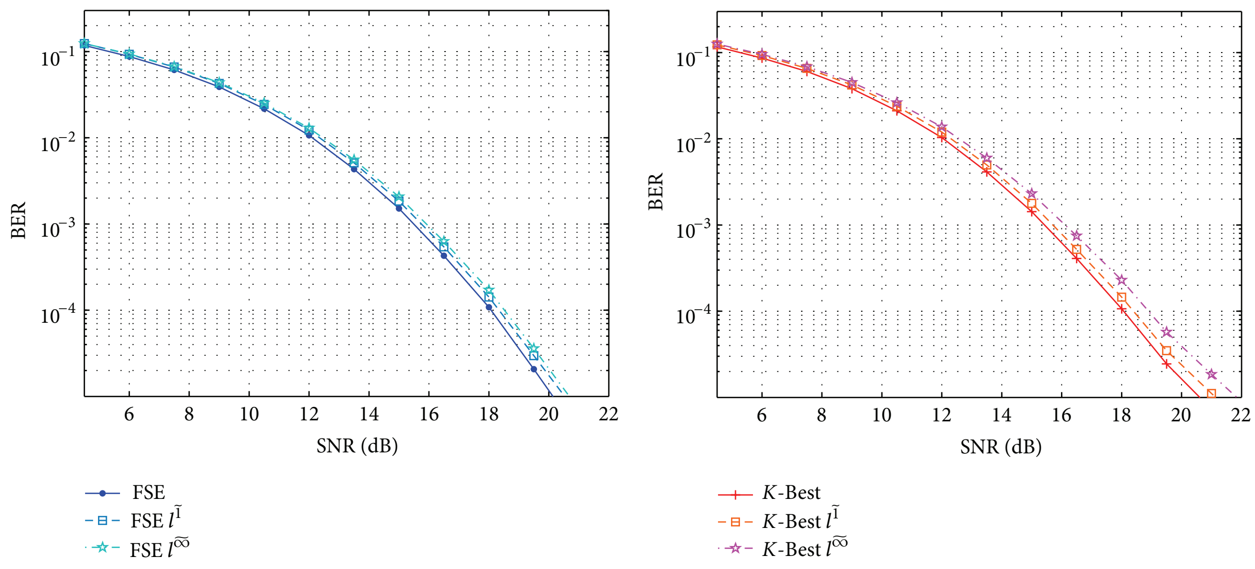
for the $\tilde{\ell^1}$-norm variant of the algorithm. The norm approximation can also be performed following the $\tilde{\ell^\infty}$-norm simplified model, in which case the following expressions should be considered

$$E_i = \max \left(E_{i+1}, e_i \right), \tag{10}$$

with

$$e_i = u_{i,i} \left[\max \left(\left| \Re (a_i + z_i) \right|, \left| \Im (a_i + z_i) \right| \right) \right]. \tag{11}$$

The implementation of an approximate norm impacts the error-rate performance of the VP system differently depending on the tree-search strategy used in the perturbation process. This fact is shown in Figure 8, where the BER performance degradation introduced when approximating the ℓ^2 norm by the suboptimum $\tilde{\ell^1}$ and $\tilde{\ell^\infty}$ norms is depicted for the FSE and K-best tree-search approaches. For the FSE case depicted in Figure 8(a), the use of an approximate norm only affects the accumulated distances related to the candidate branches, but not the branches themselves. This is due to the fact that the nodes expanded at each level where $n_i \le 2$ are the same regardless of the norm used to compute the distance increments to z_i. In the K-best model, on the other hand, the node selection procedure is solely



(a) FSE

(b) K-best

Figure 8: BER performance degradation introduced by approximating the ℓ^2 norm by the simplified $\ell^{\tilde{1}}$ and $\ell^{\tilde{\infty}}$ norms in the FSE ($n_T = 10$) (a) and K-best ($K = 7$) (b) tree-search approaches.

Table 1: Hardware resource occupation and throughput of the tree-search architectures under study.

	K-Best $K = 7$	FSE $n_T = 10$	K-Best $\ell^{\tilde{1}}$ $K = 7$	FSE $\ell^{\tilde{1}}$ $n_T = 10$
Number of occupied slices (39,360)	25%	10%	26%	10%
Number of slice registers (314,880)	10%	5%	10%	4%
Number of slice LUTs (157,440)	20%	7%	20%	7%
Used as logic	11%	6%	11%	6%
Used as memory	27%	2%	27%	2%
Number of DSP48e1s (576)	40%	40%	31%	29%
Q (Gbps)	5.52	5.63	5.34	5.62

based on previously computed distances, and therefore, the introduction of an approximate norm will noticeably alter the structure of the candidate branches. Consequently, a higher error-rate performance degradation of the K-best algorithm with an approximate norm can be expected when compared to the norm-simplified FSE model.

The implementation of the approximate $\ell^{\tilde{1}}$ norm yields a high-SNR performance loss of 0.22 dB and 0.25 dB for the FSE and K-best fixed-complexity algorithms, respectively. Due to the worse approximation of the Euclidean distances performed by the suboptimum $\ell^{\tilde{\infty}}$ norm, the performance gap with respect to the optimum FSE and K-best structures is widened in this case. This way, a performance loss of 0.45 dBs is experienced by the simplified $\ell^{\tilde{\infty}}$-FSE model, whereas an error-rate degradation of 0.85 dBs is suffered by the K-best in the high-SNR regime. In any case, the implementation of an alternative norm does not alter the diversity order of the VP scheme.

The computational complexity reduction yielded by both norm approximation approaches is similar, whereas the performance is slightly better for the $\ell^{\tilde{1}}$ norm. Consequently, the $\ell^{\tilde{1}}$ norm-simplified model will be considered for hardware implementation.

8. Implementation Results

The proposed tree-search architectures have been implemented on a Xilinx Virtex VI FPGA (XC6VHX250T-3). The occupation results have been obtained by means of the place and route tool included in the System Generator for DSP software.

Table 1 depicts the device occupation summary of the implemented vector precoders for an $N = 4$ users system with $B = 25$ eligible lattice points. Even if the FSE and K-best models use a similar amount of embedded multipliers (DSP48e1), the device occupation in terms of slices is considerably higher for the latter. This is due to the longer latency of the K-best architecture caused by the distributed sorting procedure, which ultimately results in a great amount of data being stored in several pipeline stages. As a consequence to this, around 27% of the slice LUTs are used as memory in the K-best implementation, as opposed to the 2% utilized by the FSE for the same purpose. Other than the higher occupation due to pipeline registers, the difference in latency between the two designs is of minor importance as both structures are fully-pipelined and therefore output a processed data vector at every clock-cycle. As already anticipated, the utilization of the approximate $\ell^{\tilde{1}}$ norm yields a notable reduction in the

Table 2: Throughput, area occupation and BER loss with respect to the optimum for the K-Best applied to MIMO detection and the proposed K-Best and FSE approaches for VP.

	[16]	[19]	[17]	[22]	[36]	K-Best (proposed)	FSE (proposed)
Systems	4×4 16-QAM $K = 10$	4×4 16-QAM $K = 5$	4×4 16-QAM $K = 5$	4×4 64-QAM $K = 64$	4×4 64-QAM $K = 64$	4×4 16-QAM $K = 7$	4×4 16-QAM $n_T = 10$
Throughput (Mbps)	10	53.3	424	75	100	5520	5630
Area (kGE)	52	91	68	1790	1760	1732	374
Mbps/kGE	0.19	0.58	6.23	0.04	0.056	3.18	15.05
BER loss at 20 dB	<0.5 dB	<0.5 dB	~0.75 dB	~0.5 dB	<0.5 dB	0.7 dB	0.3 dB

amount of allocated embedded multipliers for both fixed-complexity tree-search models.

The maximum throughput of the implemented architectures in terms of processed gigabits per second is also shown in Table 1 for a 16-QAM modulation constellation. For a system with N users and a constellation of P elements, the throughput for fully-pipelined architectures can be computed as

$$Q = N f_{\text{clock}} \log_2(P), \qquad (12)$$

where f_{clock} represents the maximum working frequency of the design as given by the *Post-Place and Route Static Timing Report*. Both tree-search algorithms achieve a very high data-processing throughput (in the range of 5 Gbps) due to the loopless parallel structure that enables the processing of a new data vector at every clock cycle. Note that the maximum working frequency of the designs presented in this contribution can be obtained by using (12) and the results in Table 1.

Additionally note that a higher throughput can be achieved by increasing the order of the modulation in use. In such a case, the modifications to be performed in the proposed architecture are minimal. These include an update of the considered lattice values ($|\mathscr{L}|$) and the adaptation of the first DPU where the straightforward node sequencing is performed. Furthermore, given the low hardware resource occupation required by the proposed FSE tree-search architectures, a higher data processing throughput can be easily obtained by running several tree-search instances in parallel.

Table 2 compares the area and throughput of the proposed K-best and FSE hardware architectures with similar structures used in point-to-point MIMO detection. Even if a direct comparison should be done carefully due to the already described differences between multiuser precoding and MIMO detection scenarios, it is worth noting the high Mbps/kGE ratio of the presented FSE approach.

9. Conclusion

This paper has addressed the issues of a fully-pipelined implementation of the FSE and K-best tree-search approaches for a 4×4 VP system. The sorting stages required by the K-best scheme have been performed by means of the WPE distributed sorting strategy along with a nonsequential complex-plane enumerator, which has also been incorporated into the FSE structure to determine the child nodes to be expanded in those tree levels $i < N$ where $n_i > 1$. The design parameters that establish the performance-complexity trade-off of these nonrecursive tree-search approaches have been set so as to yield a similar count of allocated embedded multipliers. Additionally, the use of an approximate norm to reduce the computational complexity of the PED calculations has been contemplated.

Provided performance results have shown a close-to-optimal performance and a very high achievable throughput in the range of 5 Gbps for both techniques. Nevertheless, the error-rate performance of the FSE has been shown to considerably outperform the K-best in the high-SNR range. Additionally, the provided FPGA resource occupation results have demonstrated the greater efficiency of the FSE architecture when compared to the K-best fixed-complexity structure.

Due to the good performance, occupation results, and simplicity of implementation, it is concluded that the FSE is best suited for the practical implementation of fixed-complexity and high-throughput vector precoders.

Acknowledgments

The authors would like to thank the Department of Education, Universities and Research and the Department of Industry, Trade and Tourism of the Basque Government.

References

[1] B. M. Hochwald, C. B. Peel, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication, part II: perturbation," *IEEE Transactions on Communications*, vol. 53, no. 3, pp. 537–544, 2005.

[2] C. Windpassinger, R. F. H. Fischer, and J. B. Huber, "Lattice-reduction-aided broadcast precoding," *IEEE Transactions on Communications*, vol. 52, no. 12, pp. 2057–2060, 2004.

[3] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[4] L. Babai, "On lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[5] D. Seethaler and G. Matz, "Efficient vector perturbation in multi-antenna multi-user systems based on approximate integer relations," in *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO '06)*, pp. 1–5, September 2006.

[6] S. Hur, N. Kim, H. Park, and J. Kang, "Enhanced lattice-reduction-based precoder with list quantizer in broadcast channel," in *Proceedings of the IEEE 66th Vehicular Technology Conference (VTC '07)*, pp. 611–615, October 2007.

[7] F. Liu, L. Jiang, and C. He, "Low complexity MMSE vector precoding using lattice reduction for MIMO systems," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 2598–2603, June 2007.

[8] M. Taherzadeh, A. Mobasher, and A. K. Khandani, "LLL lattice-basis reduction achieves the maximum diversity in MIMO systems," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '05)*, pp. 1300–1304, September 2005, maximum diversity; MIMO fading channels; MIMO broadcast systems; lattice-reductionaided decoding; point-to-point system;multiple-access system.

[9] M. Taherzadeh, A. Mobasher, and A. K. Khandani, "Communication over MIMO broadcast channels using lattice-basis reduction," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4567–4582, 2007.

[10] K. H. Lin, H. L. Lin, R. C. Chang, and C. F. Wu, "Hardware architecture of improved Tomlinson-Harashima Precoding for downlink MC-CDMA," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '06)*, pp. 1200–1203, December 2006.

[11] A. Burg, D. Seethaler, and G. Matz, "VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 07)*, pp. 673–676, May 2007.

[12] P. Bhagawat, W. Wang, M. Uppal et al., "An FPGA implementation of dirty paper precoder," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 2761–2766, June 2008.

[13] L. G. Barbero and J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," in *Proceedings of the IEEE International Conference on Communications (ICC '06)*, vol. 7, pp. 3082–3087, June 2006.

[14] L. G. Barbero and J. S. Thompson, "FPGA design considerations in the implementation of a fixed-throughput sphere decoder for MIMO systems," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '00)*, pp. 1–6, August 2006.

[15] L. G. Barbero and J. S. Thompson, "Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2804–2814, 2008.

[16] K. W. Wong, C. Y. Tsui, R. S. K. Cheng, and W. H. Mow, "A VLSI architecture of a k-best lattice decoding algorithm for MIMO channels," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, pp. 273–276, May 2002.

[17] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 1151–1154, September 2006.

[18] Q. Li and Z. Wang, "Improved k-best sphere decoding algorithms for MIMO systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 1159–1162, May 2006.

[19] Z. Guo and P. Nilsson, "Algorithm and implementation of the k-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491–503, 2006.

[20] M. Shabany and P. G. Gulak, "Scalable VLSI architecture for k-best lattice decoders," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 940–943, May 2008.

[21] C. A. Shen and A. M. Eltawil, "A radius adaptive k-best decoder with early termination: algorithm and VLSI architecture," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 9, pp. 2476–2486, 2010.

[22] S. Chen, T. Zhang, and Y. Xin, "Relaxed k-best MIMO signal detector design and VLSI implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 328–337, 2007.

[23] M. Mahdavi, M. Shabany, and B. V. Vahdat, "A modified complex k-best scheme for high-speed hard-output MIMO detectors," in *Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '10)*, pp. 845–848, August 2010.

[24] C. A. Shen, A. M. Eltawil, and K. N. Salama, "Evaluation framework for k-best sphere decoders," *Journal of Circuits, Systems and Computers*, vol. 19, no. 5, pp. 975–995, 2010.

[25] D. A. Schmidt, M. Joham, and W. Utschick, "Minimum mean square error vector precoding," in *Proceedings of the IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '05)*, vol. 1, pp. 107–111, September 2005.

[26] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, 1999.

[27] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, 2003.

[28] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," in *Proceedings of the International Symposium on Fundamentals of Computation Theory (FCT '91)*, vol. 529, pp. 68–85, September 1991.

[29] J. Zhang and K. J. Kim, "Near-capacity MIMO multiuser precoding with QRD-M algorithm," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers (ACSSC '05)*, vol. 1, pp. 1498–1502, November 2005.

[30] R. Habendorf and G. Fettweis, "Vector precoding with bounded complexity," in *Proceedings of the 8th IEEE Signal Processing Advances in Wireless Communications (SPAWC '07)*, pp. 1–5, June 2007.

[31] M. Barrenechea, M. Mendicute, J. Del Ser, and J. S. Thompson, "Wiener filter-based fixed-complexity vector precoding for the MIMO downlink channel," in *Proceedings of the IEEE 10th Workshop on Signal Processing Advances in Wireless Communications (SPAWC '09)*, pp. 216–220, ita, June 2009.

[32] M. Barrenechea, M. Mendicute, I. Jimenez, and E. Arruti, "Implementation of complex enumeration for multiuser mimo

vector precoding," in *Proceedings of the EURASIP European Signal Processing Conference (EUSIPCO '11)*, pp. 739–743, August 2011.

[33] P. Y. Tsai, W. T. Chen, X. C. Lin, and M. Y. Huang, "A 4 × 4 64-QAM reduced-complexity k-best MIMO detector up to 1.5 Gbps," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '10)*, pp. 3953–3956, May 2010.

[34] S. Mondal, W. H. Ali, and K. N. Salama, "A novel approach for k-best MIMO detection and its VLSI implementation," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 936–939, May 2008.

[35] S. Mondal, A. M. Eltawil, and K. N. Salama, "Architectural optimizations for low-power k-best MIMO decoders," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3145–3153, 2009.

[36] S. Mondal, A. Eltawil, C. A. Shen, and K. N. Salama, "Design and implementation of a sort-free K-best sphere decoder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 10, pp. 1497–1501, 2010.

[37] M. Barrenechea, *Design and implementation of multi-user mimo precoding algorithms [Ph.D. dissertation]*, University of Mondragon, Mondragon, Spain, 2012.

[38] A. Burg, M. Wenk, M. Zellweger, M. Wegmueller, N. Felber, and W. Fichtner, "VLSI implementation of the sphere decoding algorithm," in *Proceedings of the 30th European Solid-State Circuits Conference (ESSCIRC '04)*, pp. 303–306, September 2004.