Probabilistic Spherical Detection and VLSI Implementation for Multiple-Antenna Systems

Chester Sungchung Park, Member, IEEE, Keshab K. Parhi, Fellow, IEEE, and Sin-Chong Park, Member, IEEE

Abstract—This paper presents a novel probabilistic spherical-detection (P-SD) method which applies the probabilistic-search algorithm to conventional depth-first SD (DF-SD). By confining the tree search into candidates which can be selected in an adaptive manner, a large number of promising candidates can be evaluated before termination. Consequently, the proposed P-SD improves the error performance of DF-SD with early termination, while retaining the hardware efficiency. An efficient VLSI architecture is proposed for implementation of the P-SD algorithm, and the results of the synthesized architecture are presented. The main advantage of P-SD is that it can fully exploit the state-of-the-art architectures of DF-SD, since it can be implemented by simply adding two functional blocks to conventional DF-SD. By analyzing the performance-complexity tradeoffs, it is concluded that our proposed P-SD is advantageous over conventional DF-SD and K-best algorithm, when the maximum-likelihood error performance is desired.

Index Terms—Adaptive signal processing, CMOS digital integrated circuits, maximum-likelihood (ML) detection, multiple-input–multiple-output (MIMO) systems.

I. INTRODUCTION

OWADAYS, multiple-input–multiple-output (MIMO) communications have emerged as one of the key elements for the next-generation wireless communications, since the channel capacity is proportional to the minimum number of antennas [1]. Practically, the use of multiple antennas can significantly improve the coverage by diversity gain and/or increase the transmission rate by multiplexing gain [2]. Current standardization activities in wireless local access networks and wireless cellular networks have chosen the use of multiple antennas to further improve the throughput of current wireless networks [3], [4].

In most of practical MIMO systems, forward-error-correction (FEC) codes such as convolutional codes and low-density parity-check (LDPC) codes [3] are used for reliable communications. Accordingly, in order to achieve near-channel-capacity performance, the maximum a posteriori (MAP) processing needs to be performed at the receiver side [5]. Recently, the implementation of MAP processing has been facilitated by using the so-called spherical detection (SD), an efficient tree-search algorithm that finds a set of vector symbols (referred to as a list in this paper) closest to the received vector symbol [6], [7](this tree-search algorithm is often called list SD (LSD), a modified version of standard SD [8]; in this paper, the terms SD and LSD are used interchangeably, unless otherwise mentioned). The utilization of SD enables accurate calculation of soft-decision values for the subsequent FEC decoding, approaching maximum-likelihood (ML) error performance.

Recalling that SD is categorized as a sequential-decoding algorithm, several tree-search strategies introduced in [9] and [10] can be considered for efficient implementation. A depth-first (DF) search strategy has been of great interest to several research groups, since it is attractive in terms of both error performance and hardware complexity [11]–[13]. The disadvantage of DF search strategy is that the computation amount (or, equivalently, the throughput of hardware for SD) is random and unpredictable. For this reason, a breadth-first search strategy utilized by K-best algorithm [14]–[17] and its variation [18] has gained much attention recently as an alternative, since it always guarantees a constant throughput. However, the computational complexity of breadth-first search strategy is prohibitively large because of unavoidable needs for sorting. Thus, the DF search strategy is still attractive from the perspective of hardware implementation, and the DF-SD with a constant throughput [19]–[21] is of significant interest.

In order to guarantee a constant throughput, the tree search of DF-SD needs to be terminated earlier, i.e., before evaluating all possible candidates for vector symbols (not always but with a high probability) [11]–[13]. Thus, the error performance of DF-SD with early termination is far from that of MAP processing, since the optimality of the resulting list (in terms of distance) is not guaranteed. Although there are a number of reports about standard DF-SD in the literature, relatively less attention has been paid to DF-SD with early termination [19]–[21], which motivated this paper.

In this paper, in order to improve the error performance, the probabilistic-search algorithm in [22] is applied to the state-of-the-art DF-SD in [11]–[13]: By confining the search into channel-adaptively selected promising candidates, a larger number of promising candidates can be evaluated before termination, which makes the resulting list closer to the optimum one. When applied to LDPC-coded MIMO orthogonal-fre-
frequency-division-multiplexing (OFDM) systems, the proposed probabilistic SD (P-SD) significantly improves the error performance of DF-SD, while maintaining the hardware efficiency. The VLSI architecture of P-SD is also proposed along with its synthesis results, showing that conventional DF-SD can be modified by simply adding two functional blocks. Furthermore, in order to compare with conventional SD, the tradeoffs between error performance and computational complexity are addressed. It is shown that the proposed P-SD provides a significant gain in either required SNR or computation amount.

The remainder of this paper is organized as follows. In Section II, the principle of SD is explained, focusing on the application to MAP processing. The conventional DF-SD and probabilistic-search algorithm are introduced in Section III and Section IV, respectively. Section V provides the solution to improving the error performance of DF-SD with early termination, the so-called P-SD, which is verified by our simulation results in Section VI. The VLSI architecture for P-SD is proposed, along with the synthesis results, in Section VII. Additionally, the tradeoffs between error performance and computational complexity are presented and compared with those of conventional SD, DF-SD, and K-best algorithm in Section VIII. Finally, Section IX concludes this paper.

II. PRINCIPLES OF SD

In a MIMO system with $N_T$ transmit antennas and $N_R$ receive antennas, the received vector symbol, $\mathbf{y} = \{y_i, i = 1, \ldots, N_R\}$, is given by

$$\mathbf{y} = \mathbf{Hs} + \mathbf{w} \tag{1}$$

where $\mathbf{H} = \{h_{i,j}, i = 1, \ldots, N_T, j = 1, \ldots, N_R\}$ is the channel-coefficient matrix, $\mathbf{s} = \{s_i, i = 1, \ldots, N_T\}$ is the transmitted vector symbol whose elements are drawn from the constellation of $M$-ary quadrature amplitude modulation (QAM), and $\mathbf{w} = \{w_i, i = 1, \ldots, N_R\}$ is the additive white Gaussian noise. Here, $\mathbf{s}$ is obtained by

$$s_i = \text{map}(\{x_{i,k}, k = 1, \ldots, \log_2 M\}) \tag{2}$$

where $\text{map}(\cdot)$ represents the bit-to-symbol mapping of $M$-ary QAM [3] and $x_{i,k}$ denotes the $k$th-coded (and interleaved) bit transmitted at the $i$th transmit antenna. According to the derivation described in [6], the soft-decision values for the subsequent decoding are given as

$$\hat{x}_{i,k} = \min_{s \in L_{x_{i,k}}} |\mathbf{y} - \mathbf{Hs}|^2 - \min_{s \in L_{x_{i,k}} = 1} |\mathbf{y} - \mathbf{Hs}|^2 \tag{3}$$

which is the Max-log approximation of a posteriori log-likelihood ratio (LLR) [13], [17] (note that the iterative demodulation and decoding method introduced in [6] is not considered throughout this paper because of difficulty in implementation). In the right-hand side of (3), the former maximization is performed over all $M^{N_T}/2$ candidates in $\{\mathbf{s} : x_{i,k} = 0\}$, while the latter maximization is over the other $M^{N_T}/2$ candidates in $\{\mathbf{s} : x_{i,k} = 1\}$. By utilizing the SD shown in [6], [13], [15], the soft-decision values in (3) can be calculated approximately yet efficiently: The maximization of squared distance $d(\mathbf{s}) = |\mathbf{y} - \mathbf{Hs}|^2$ is performed over only the candidates in a list denoted by $L$ rather than over all possible candidates, in other words

$$\hat{x}_{i,k} = \min_{\mathbf{s} \in L_{x_{i,k}}} |\mathbf{y} - \mathbf{Hs}|^2 - \min_{\mathbf{s} \in L_{x_{i,k}} = 1} |\mathbf{y} - \mathbf{Hs}|^2. \tag{4}$$

As mentioned earlier, the list is composed of candidates minimizing the (squared) distance. Hence, it is easily understood that as the number of candidates in the list $N_{CD}$ increases, the soft-decision values in (4) approach the Max-log-approximated LLR in (3).

According to the QR decomposition, it follows that

$$\mathbf{H} = \mathbf{QR} \tag{5}$$

where $Q = \{q_{i,j}, i = 1, \ldots, N_T, j = 1, \ldots, N_R\}$ is unitary (i.e., $Q^H \mathbf{Q} = \mathbf{I}_{N_R}$) and $R = \{r_{i,j}, i = 1, \ldots, N_T, j = 1, \ldots, N_R\}$ is upper triangular. If $N_R$ is equal to $N_T$, $d(\mathbf{s})$ can be expressed as

$$d(\mathbf{s}) = |\hat{\mathbf{y}} - \mathbf{Rs}|^2 \tag{6}$$

where $\hat{\mathbf{y}} = \{\hat{y}_i, i = 1, \ldots, N_T\}$ is given by $\hat{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$. Owing to the upper triangularity of $\mathbf{R}$, the calculation of $d(\mathbf{s})$ can be performed recursively: Defining a partial vector symbol in the $i$th level as

$$\mathbf{s}_{(i)} = \{s_{j}, j = i, \ldots, N_T\} \tag{7}$$

the corresponding partial squared distance in the $i$th-level $T_i(\mathbf{s}_{(i)})$ is recursively calculated as

$$T_i(\mathbf{s}_{(i)}) = T_{i+1}(\mathbf{s}_{(i+1)}) + \delta_i(\mathbf{s}_{(i)}), \quad i = 1, \ldots, N_T \tag{8}$$

where the (squared) distance increment is defined as

$$\delta_i(\mathbf{s}_{(i)}) = \left(\sum_{j=i}^{N_T} r_{i,j}s_j\right)^2. \tag{9}$$

Note that $T_{N_T+1}(\mathbf{s}_{(N_T+1)})$ is set to zero, and thus, $T_1(\mathbf{s}_{(1)})$ is nothing but $d(\mathbf{s})$ from (8) and (9). Considering the spherical constraint with a radius $r$

$$d(\mathbf{s}) < r^2 \tag{10}$$

then it immediately follows that the partial vector symbols that violate the constraint

$$T_i(\mathbf{s}_{(i)}) < r^2 \tag{11}$$

also violate the spherical constraint (10). This is the idea behind the tree pruning of SD, which significantly reduces the computation amount [6], [7]. Even though the discussion in this section assumes single-carrier transmission over frequency-flat fading channels, the extension to multicarrier transmission (i.e., OFDM) over frequency-selective fading channels is quite
two terms are unique to a specific child node (due to their dependence on the $i$th symbol $s_i$). As a result, it is desirable to calculate $\delta_i(S(i))$ and, thus, $T_i(S(i))$ for all $M$ child nodes simultaneously, from the perspective of computational complexity [11] and [12]. The numbers in the parentheses shown in Fig. 1 indicate the instant (in clock cycles) when the corresponding node is visited as a parent node. Since the one-node-per-cycle operation in [12] and [13] is assumed throughout this paper, the required number of clock cycles is always equal to the number of visited parent nodes.

The tree-search strategy of DF-SD combined with the SE ordering is performed as shown in Fig. 1(a). The utilization of storage elements corresponding to the tree traversal is shown in Fig. 1(b). Suppose that a parent node in the $(i+1)$th level is visited, and its $(M-1)$ child nodes in the $i$th level are evaluated. If the child nodes are not in the first level (bottom level), i.e., $i \neq 1$, and there exists at least one child node that satisfies the spherical constraint (11), i.e., it has a value of $T_i(S(i))$ smaller than $r^2$, then the forward recursion is performed. In the forward recursion of DF-SD, the closest child node, i.e., the child node corresponding to the partial vector symbol minimizing $T_i(S(i))$, is visited as the next parent node, and the intermediate calculations $T_{i+1}(S(i))$ of the remaining $(M-1)$ child nodes (also referred to as the yet-to-be-visited nodes) are stored until they are visited as parent nodes. On the other hand, if the child nodes are in the first level ($i = 1$) or all the child nodes violate the spherical constraint (11), then the backward recursion is performed. In the backward recursion of DF-SD, a preferred child node in the upper level (that is the closest to the current level) is revisited as the next parent node. Throughout this paper, a preferred child node in each level is defined as a node whose corresponding partial distance is the smallest among the yet-to-be-visited nodes in the level and, of course, satisfies the sphere constraint. Note that, once the tree traversal is started, the original DF-SD (i.e., DF-SD without early termination) continues the search until there remain no more yet-to-be-visited nodes, satisfying the sphere constraint or, equivalently, no more preferred nodes. The earlier description of tree traversal of DF-SD is summarized in the control flow shown in Fig. 2. This is how the conventional DF-SD keeps the required number of storage elements (in the number of nodes) as low as $M(N_{PF} - 1)$.

It is widely understood that the number of visited parent nodes of DF-SD can be significantly reduced by the help of radius reduction. In this paper, the radius reduction of standard DF-SD ($N_{CD} = 1$) shown in [12] is generalized to DF-SD with arbitrary $N_{CD}$. Note that the only difference from standard DF-SD is that the largest distance in a list is chosen as a new radius each time the list is updated. Consequently, as the number of candidates in a list increases, the radius reduction becomes slower, resulting in a larger number of clock cycles (and, thus, higher computational complexity).

### IV. Probabilistic-Search Algorithm

As pointed out in [11] and [12], the number of clock cycles $N$ needs to be constrained to a predetermined value for a constant throughput, since, otherwise, it tends to randomly fluctuate with multipath fading and additive noise. In other words, the tree traversal needs to be terminated once $N$ parent nodes are visited.
Consequently, the DF-SD with early termination cannot guarantee the optimality of the resulting list (in terms of distances). More specifically, closer candidates (i.e., candidates with distances smaller than the radius) may remain to be evaluated after early termination of tree traversal, thus leading to degraded error performance [11]. In the previous example shown in Fig. 1, the evaluation of all possible candidates takes 21 clock cycles. If the number of visited parent nodes is constrained to be 11 (i.e., \( N = 11 \)), only 32 candidates are evaluated and the 32 remaining candidates are not considered. In general, the error performance is degraded more severely in DF-SD, as \( N \) becomes smaller.

Accordingly, the main focus of this paper is the way of improving the error performance of DF-SD with early termination. It is easily understood that as a larger number of closer candidates can be evaluated before termination, the subsequent calculation of soft-decision values (based on the resulting list) becomes more reliable, possibly leading to improved error performance.

In this paper, the probability of transmission, which is defined as the probability that a particular candidate is indeed transmitted, is used to distinguish more promising candidates and evaluate them earlier. By evaluating the candidates in a descending order of this probability, the error performance can be improved, approaching that of the MAP processing. The idea behind this probabilistic-search algorithm proposed in [22] is as follows. It can be intuitively said that a candidate with a higher probability of transmission is more promising in that the distance is, with a high probability, smaller. Therefore, if the candidates with a higher probability of transmission are evaluated earlier, the resulting list becomes more similar to the optimum one. Furthermore, since a candidate with a higher probability is likely to be closer to the received vector symbol, the radius of sphere tends to be reduced more rapidly, so that a larger number of promising candidates can be evaluated before termination.

In [22], it is shown that the probability of transmission of each candidate is obtained as a closed-form expression. To be specific, the probability of transmission of \( \mathbf{s} \) decreases exponentially with

\[
P(\mathbf{s}) = \sum_{i=1}^{N_T} r_{i}^{2} u_{i}^{2} \tag{14}
\]

where \( u_{i} \) is defined as the distance of \( s_{i} \) from the closest node among all the \( M \) child nodes derived from a parent node, i.e., shaded nodes shown in Fig. 1. Since all \( M \) child nodes are evaluated within a single clock cycle, we are more interested in the probability of transmission of a partial vector symbol in the second level, i.e., the probability that the transmitted vector symbol is derived from a node in the second level \( \mathbf{s}(2) \). From (14), it follows that this probability decreases exponentially with

\[
P(\mathbf{s}(2)) = \sum_{i=2}^{N_T} r_{i}^{2} u_{i}^{2} := Q(\mathbf{u}) \tag{15}
\]

where \( \mathbf{u} = \{ u_{ji}, j = 2, \ldots, N_T \} \) denotes a vector symbol of distance in the second level. Note that the definition of \( Q(\mathbf{u}) \) is based on the dependence of \( P(\mathbf{s}(2)) \) on \( \mathbf{u} \). The most interesting part of the closed-form expression is that the probability of transmission can be easily calculated for all possible candidates, without evaluating their partial distance, if only the closest nodes as well as the channel coefficients are known to the receiver. Fig. 3 shows \( Q(\mathbf{u}) \) as a measure of probability for the previous example, assuming that \( r_{2,1}^{2} = 3.5, r_{2,2}^{2} = 2.0 \), and \( r_{3,3}^{2} = 0.5 \). Recall that \( u_{i} \) is either 1 or 1.414 in the normalized QPSK constellation.

Let us revisit the tree traversal of DF-SD shown in Fig. 1 from the viewpoint of probability of transmission. If the tree traversal of DF-SD is constrained to 11 clock cycles, many promising candidates such as \( [0\ 3]^{T} \) and \( [1\ 2]^{T} \) remain to be evaluated after termination. It is important to consider the well-known statistical property [24] that

\[
E\{r_{i}^{2}\} = (N_T + 1) - i, \quad i = 1, \ldots, N_T. \tag{16}
\]

Since \( E\{r_{i}^{2}\} \) is smaller in the upper level, the candidates deviated from the closest child node in the upper level are more likely to have high probabilities of transmission than those deviated in the lower level. For example, in Fig. 3, \( [3\ 0]^{T} \) (deviated in the third level) has a higher probability of transmission than \( [0\ 1]^{T} \) (deviated in the second level). Thus, it can be concluded that the tree-search strategy of DF-SD, which first searches the candidates deviated in the lower level (for better utilization of storage elements, as mentioned previously), is far from optimum in the case of early termination.
Suppose that the probabilistic-search algorithm in [22] is directly applied to the state-of-the-art DF-SD in [11] and [12]. Then the tree-search strategy is based on the probability of transmission, not the DF search strategy. Fig. 4 shows the tree traversal of probabilistic-search algorithm for the previous example, along with the corresponding utilization of storage elements. If the one-node-per-cycle operation is guaranteed, the evaluation of all possible candidates takes 21 clock cycles, just as in the DF-SD in Fig. 1. In this probabilistic-search algorithm, however, the candidates are evaluated in quite a different order, in more detail, which is in a descending order of probability of transmission. As a result, a larger number of promising candidates before termination, while retaining efficient utilization of storage elements.

Before discussing the details of P-SD, let us define a candidate group. Recall that the constellation of $M$-ary QAM can be divided into a few subsets (namely, $S(0)-S(5)$ shown in Fig. 5) of all the alphabets with an identical distance from a reference alphabet. When the reference alphabet itself is considered as a subset $S(0)$, it follows that $S(1)$ has four alphabets and a distance of one, $S(2)$ has four alphabets and a distance of $\sqrt{2}$, $S(3)$ has four alphabets and a distance of two, $S(4)$ has eight alphabets and a distance of $\sqrt{5}$, and $S(5)$ has four alphabets and a distance of $2\sqrt{2}$. Notice that the normalization factor is chosen to simplify the calculation of $Q(u)$, without loss of generality. Then, it readily follows that there are 3 possible subsets, $S(0)-S(2)$, for QPSK (as exemplified in Fig. 5), 10 possible subsets, $S(0)-S(9)$, for 16-QAM, and 33 possible subsets, $S(0)-S(32)$, for 64-QAM. Based on these subsets, a candidate group can be defined as a group of partial vector symbols (in the second level) of which the alphabets in each level belong to an identical subset. Therefore, a candidate group can be represented by a vector symbol of $(N_T - 1)$ subsets. Accordingly, when the closest nodes in the tree (represented by shaded nodes in the previous figures) are chosen as the reference alphabets, a candidate group is nothing but a group of candidates with an identical probability of transmission. In the previous example, a candidate group, $[S(0), S(1)]^T$, consists of two partial vector symbols, $[3, 0]^T$ and $[0, 3]^T$, or equivalently, eight candidates derived from them, $[0, 3, 0]^T, [1, 3, 0]^T, [2, 3, 0]^T, [3, 3, 0]^T, [0, 0, 3]^T, [1, 0, 3]^T, [2, 0, 3]^T$, and $[3, 0, 3]^T$. Note that $[3, 0]^T$ and $[0, 3]^T$ have an identical probability of transmission, i.e., $Q(u) = 0.5$, as shown in Fig. 3.

Now, we are ready to describe the tree traversal of our proposed P-SD in detail. As mentioned before, the proposed P-SD can be seen as a special case of DF-SD of which the tree traversal is confined into a few promising candidate groups. Fig. 6 shows the tree traversal and corresponding utilization of storage elements of P-SD with four promising candidate

![Fig. 4. Example of tree traversal of probabilistic-search algorithm ($N_T = 3, M = 4$). (a) Search tree. (b) Storage elements. For clarity, the tree pruning and radius reduction are not shown.](Image)

![Fig. 5. Definition of subsets for $M$-ary QAM. The constellation is divided into a few subsets, $S(0)-S(5)$, which consist of all the alphabets with an identical distance from a reference alphabet $S(0)$.](Image)

![Fig. 6. Tree traversal and corresponding utilization of storage elements of P-SD with four promising candidate](Image)
groups of \([S(0)\ S(0)]^T, [S(0)\ S(1)]^T, [S(0)\ S(2)]^T,\) and \([S(1)\ S(0)]^T\). As shown in Fig. 3, these candidate groups have higher probabilities of transmission than the other candidate groups; specifically, \(Q(\mathbf{u})\) is smaller than or equal to two. The other candidate groups indicated by “X” marks in the figure, the less promising candidate groups, are pruned in the tree. Interestingly, as shown in Fig. 6, the order of evaluation of candidates is somewhat different from that of the probabilistic-search algorithm; in other words, the tree traversal of P-SD is not performed in a probabilistic manner: The candidates are evaluated in the order of \([S(0)\ S(0)]^T ([1\ 1]^T), [S(1)\ S(0)]^T ([0\ 1]^T, [3\ 1]^T), [S(0)\ S(1)]^T ([3\ 0]^T, [0\ 3]^T),\) and \([S(0)\ S(2)]^T ([1\ 2]^T)\) in the proposed P-SD, while they are in the order of \([S(0)\ S(0)]^T, [S(0)\ S(1)]^T, [S(0)\ S(2)]^T,\) and \([S(1)\ S(0)]^T\) in the probabilistic-search algorithm. Nonetheless, it is easily found that the proposed P-SD can evaluate a larger number of promising candidates before termination by simply pruning the less-promising candidates. All the candidate groups with \(Q(\mathbf{u})\) smaller than or equal to two can be successfully evaluated within 11 clock cycles as shown in Fig. 6, just as in the probabilistic-search algorithm. Undoubtedly, the corresponding utilization of storage elements is as efficient as that of the conventional DF-SD, since the tree traversal basically follows the DF strategy as mentioned earlier. In other words, the required number of storage elements is always kept as low as \(M(N_T - 1)\). The problems that remain to be solved are how to select a predetermined number of promising candidate groups and how to confine the tree traversal to the selected candidate groups.

The first problem, how to select a predetermined number of promising candidate groups, is considered here. As previously shown in (15), the promising candidate groups are chosen based on the channel coefficients available to the receiver. Denoting the number of promising candidate groups by \(K\), the number of subsets to be considered \(L\) is always equal to \(K\), since any candidate groups with subsets larger than \(S(K - 1)\) (i.e., larger in terms of distances) do not need to be considered (note that the \(K\) promising candidate groups always consist of subsets smaller than or equal to \(S(K - 1)\), since \(Q(\mathbf{u})\) always increases with the distances of subsets). Now, the problem is equivalent to choosing the \(K\) candidate groups that minimize \(Q(\mathbf{u})\) among \(L^{-N_T-1}\) possible candidate groups (note that a candidate group consists of \((N_T - 1)\) subsets). This problem can be efficiently solved by a tree search with \((N_T - 1)\) levels and \(L\) nodes in each level. From (15), \(Q(\mathbf{u})\) can be recursively calculated as

\[
Q_i(\mathbf{u}_{[i]}) = Q_{i+1}(\mathbf{u}_{[i+1]}) + r_{[i]}^2 u_{[i]}^2, \quad i = 2, \ldots, N_T
\]

where \(\mathbf{u}_{[i]} = \{u_{[i]}; j = i, \ldots, N_T\}\) is a partial vector symbol of distance in the \(i\)th level. Additionally, \(Q_{N_T+1}(\mathbf{u}_{[N_T+1]})\) is set to zero, and \(Q_{[0]}(\mathbf{u}_{[2]})\) is nothing but \(Q(\mathbf{u})\). In this paper, the tree search is performed using the so-called \(K\)-best algorithm [15]–[17]. Specifically, each node in the \((i + 1)\)th level is visited as a parent node, and its child nodes are evaluated. After visiting all the \(K\) nodes from \(S(0)\) to \(S(K - 1)\), \(K\) partial vector symbols in the \(i\)th level that minimize \(Q_i(\mathbf{u}_{[i]})\) are obtained. This procedure is repeated from \(i = N_T\) to \(i = 2\), as shown in (17). However, as opposed to standard \(K\)-best algorithms, the tree search always guarantees the optimality of selected candidate groups, since \(Q(\mathbf{u})\) is not jointly evaluated over multiple antennas, as shown in (15). Fig. 7 shows an example of the tree search with nine candidate groups (\(K = 9\), and thus, \(L = 9\), for four transmit and receive antennas \((N_T = N_R = 4)\) and 64-QAM \((M = 64)\). In this figure, the top level represents the fourth level (\(i = 4\)), while the bottom level represents the second level (\(i = 2\)). The numbers inside the circles in the \(i\)th level represent the squared distances of the corresponding subsets \(u_{[i]}^2\). To be specific, \(u_{[2]}^2 = 0\) for \(S(0)\), \(u_{[2]}^2 = 1\) for \(S(1)\), \(u_{[2]}^2 = 2\) for \(S(2)\), \(u_{[2]}^2 = 4\) for \(S(3)\), \(u_{[2]}^2 = 5\) for \(S(4)\), \(u_{[2]}^2 = 8\) for \(S(5)\), \(u_{[2]}^2 = 9\) for \(S(6)\), \(u_{[2]}^2 = 10\) for \(S(7)\), and \(u_{[2]}^2 = 13\) for \(S(8)\). Here, note that the number of child nodes varies with the corresponding parent node: Nine child nodes from \(S(0)\), four child nodes from \(S(1)\), three child nodes from \(S(2)\), two child nodes from \(S(3)\) and \(S(4)\), and a single child node from the four remaining subsets. It implies that only 24 partial vector symbols (among 81 partial vector symbols) need to be evaluated in the second and third levels. For example, \([S(4)\ S(1)]^T\) in the third level does not need to be evaluated, since nine more promising candidate groups have already been evaluated, e.g., \([S(0)\ S(0)]^T, [S(0)\ S(1)]^T, [S(1)\ S(0)]^T, [S(1)\ S(1)]^T, [S(1)\ S(2)]^T, [S(2)\ S(0)]^T, [S(2)\ S(1)]^T, [S(3)\ S(0)]^T, [S(3)\ S(1)]^T,\) and \([S(4)\ S(0)]^T\). This significantly simplifies the application of \(K\)-best algorithm. Furthermore, since \(r_{[i]}^2 u_{[i]}^2\) in (17) simply increases with the distance of subset, the tree traversal can be performed without any sorting. Therefore, the
computational complexity of selecting promising candidate groups is even lower than that of standard K-best algorithm.

Once a predetermined number of promising candidate groups are selected, the second problem is on how to confine tree traversal into them. Each time the backward recursion of tree traversal is performed, it is necessary to determine whether the preferred child nodes in the upper levels belong to the selected candidate groups. If there exists any preferred node belonging to the selected candidate groups, it is chosen as the next parent node, just as in the conventional DF-SD. Otherwise, the preferred node is simply ignored as if it is pruned in the tree. For this reason, it is essential to determine which subset the preferred node belongs to, since a candidate group is represented by a vector of subsets. Fortunately, this can be easily done if the tree traversal follows the SE ordering, which is often adopted in most of the conventional DF-SD architectures [11]–[13]. The idea behind this is as follows: The distance increment \( \delta(s'_{G}) \) in (9) usually tends to increase with the distance of subsets, since the closest node that minimizes \( \delta(s'_{G}) \) is chosen as the reference alphabet. Recalling that the SE ordering inherently visits the nodes in a descending order of \( \delta(s'_{G}) \), it is possible to confine the search into the selected candidate groups by simply keeping track of the number of visited child nodes derived from each node. In the previous example of Fig. 6, up to three child nodes derived from \([1]^{T}\) need to be visited as a parent node, since \([S(0) S(0)]^{T}\) and \([S(1) S(0)]^{T}\) are selected as promising candidate groups. Consequently, as shown in Fig. 6, \([2]^{T}\) is not chosen as the next parent node at the fifth clock cycle, since three child nodes \([1 1]^{T}\), \([0 1]^{T}\), and \([3 1]^{T}\) have already been visited according to the SE ordering.

Before going to Section VI, it is worth to compare the proposed P-SD with the so-called maximum-first (MF) strategy [19]–[21]. The MF strategy efficiently schedules the maximum number of clock cycles within a block of consecutive vector symbols, based on the accumulated number of clock cycles and the number of remaining vector symbols. Hence, it significantly improves the error performance of early termination, when consecutive vector symbols experience independent fading, i.e., when the required number of clock cycles fluctuates within a block. However, the performance gain is much affected by the fading dependence and becomes trivial in the absence of independent fading. On the other hand, P-SD allocates computation resources within a vector symbol (by confining the tree search into promising candidate groups), whereas MF strategy allocates them across consecutive vector symbols (by adjusting the number of clock cycles). Thus, it is readily understood that the proposed P-SD improves the error performance of early termination independently of the fading environment. According to simulation results, it turns out that P-SD outperforms MF strategy in slow fading channels, while MF strategy performs better than P-SD in fast fading channels, which is not presented here due to space limitation.

VI. EVALUATION OF ERROR PERFORMANCE

In this section, the error performance of P-SD is evaluated and compared with conventional DF-SD for LDPC-coded MIMO-OFDM systems, relying on the simulation results. Here, four transmit and receive antennas \((N_T = N_R = 4)\), 64-QAM with gray encoding \((M = 64)\), OFDM modulation with 64 subcarriers, and a (1944, 972) irregular LDPC code are assumed, even though the proposed P-SD is generally applicable to different numbers of antennas and different modulation orders as in [11] and [13] (for the detailed information of system parameters used in our simulations, refer to [25]). The multipath fading channels are assumed to be frequency-selective (with a delay spread of 250 ns) and spatially uncorrelated due to rich scattering. The list of SD is assumed to contain 16 candidates \((N_{SD} = 16)\) (which was empirically determined, based on our simulations and other references including [6] and [15]), and the belief propagation algorithm (the so-called sum–product decoding) [26] with ten iterations is adopted for the subsequent decoding. The target bit-error rate (BER) is set to \(10^{-4}\). In the remainder of this paper, the system parameters are assumed as given in this section, unless mentioned otherwise.

In Fig. 8, the error performance of conventional DF-SD in [11]–[13] is evaluated. Here, DF-SD(N) denotes DF-SD with the allowable number of clock cycles \(N\). For instance, DF-SD(4) corresponds to the simplest case where the tree search is done over only \(M\) candidates (including the nulling and cancellation solution), while DF-SD\((infty)\) corresponds to the ideal case where the tree search is not terminated improperly, i.e., all possible candidates are evaluated. When \(N\) is equal to 50, the error performance of DF-SD is significantly degraded from the ideal case, DF-SD\((infty)\), to be specific, by an SNR of 13 dB. As \(N\) increases up to 100 and 200, the error performance is improved by 9 and 10 dB, respectively. Note that, even when \(N\) is as high as 200, the performance loss from the ideal case is nontrivial (about 3 dB), as shown in the figure.

In Fig. 9, the error performance of P-SD is evaluated, when the number of promising candidate groups is given as \(K = 2, 3, 9, 20,\) and 100. Here, P-SD\((N, K)\) denotes P-SD with two parameters \(N\) and \(K\). The figure shows that P-SD\((50, 9)\) is the best choice in terms of error performance. Generally, as \(K\) decreases from this optimum value (up to one), the error performance of P-SD\((N, K)\) tends to approach that of DF-SD(4),
since the tree search is further confined into the candidates around the nulling and cancellation solution. On the other hand, as $K$ increases from this optimum value, the error performance of $P\text{-SD}(N,K)$ tends to approach that of $DF\text{-SD}(N)$. This is not surprising at all, since a sufficiently large number of promising candidates make the tree traversal similar to that of DF-SD and not that of probabilistic-search algorithm. In addition, it is clearly shown in the figure that our proposed $P\text{-SD}$, i.e., $P\text{-SD}(50,9)$, significantly outperforms $DF\text{-SD}(50)$ by 10 dB. Noticing that the error performance of $DF\text{-SD}(200)$ is attained by $P\text{-SD}(50,9)$, it is concluded that our proposed $P\text{-SD}$ significantly reduces the required number of clock cycles, i.e., by a factor of four. Fig. 9 also shows that the performance degradation due to early termination can be successfully compensated: The error performance of $P\text{-SD}(50,9)$ approaches that of ideal DF-SD within 3 dB, without further increasing the number of clock cycles. In the remainder of this paper, $K$ is assumed to be the optimum value, i.e., $K = 9$.

Before concluding this section, let us consider a nonadaptive alternative of the proposed $P\text{-SD}$ that selects the promising candidate groups based on the average power of channel coefficients ($\{E[r_{id}^2], i = 2, \ldots, N_T\}$), instead of the instantaneous channel coefficients ($\{r_{id}, i = 2, \ldots, N_T\}$). Recalling the well-known statistical property in (16), the promising candidate groups can be selected without estimating the channel coefficients. This statistical alternative is advantageous over the proposed $P\text{-SD}$ in that the computation for preprocessing stage can be saved. As shown in Fig. 10, however, the error performance of this alternative is even worse than that of the original $P\text{-SD}$. Here, $S\text{-SD}(N,K)$ denotes a statistical version of $P\text{-SD}$ with two parameters $N$ and $K$. As a consequence, it easily follows that, in spite of additional computation for preprocessing, the $P\text{-SD}$ performs better when the promising candidate groups are selected based on the instantaneous channel coefficients rather than their statistics (e.g., average power). Hence, it can be said that the adaptive control of DF search strategy (by confining the search into channel-dependently selected candidate groups) leads to the improved error performance in our proposed $P\text{-SD}$.

VII. PROPOSED VLSI ARCHITECTURE

The VLSI architecture suitable for $P\text{-SD}$ is proposed in this section. As shown in Fig. 11, the hardware for $P\text{-SD}$ consists of several functional blocks. The operation of these functional blocks can be described by three stages, in other words, Preprocessing Stage, Pipeline Stage I, and Pipeline Stage II. In Preprocessing Stage, a predetermined number of promising candidate groups are selected ($CSEL\text{ PROB}$) based on the channel coefficients. Since the wireless channel changes slowly (compared with data transmission), Preprocessing Stage is activated only when the channel is significantly changed, or the preamble is periodically received. In Pipeline Stage I, all the child nodes derived from a parent node are evaluated ($CALC\text{ INST}$), and the next parent node is chosen ($CTRL\text{ DF}$). In Pipeline Stage II, the intermediate calculations are stored in the forward recursion (STR_FWD), the preferred nodes are newly chosen (STR_FWD or STR_FWD), and the list (and therefore, the radius of sphere) is updated (UPD_LIST). Hence, the proposed VLSI architecture of $P\text{-SD}$ is characterized by two-stage pipeline architecture, which enables the one-node-per-cycle operation [12]. Additionally, in the proposed architecture, the intermediate calculations are stored and revisited in the form of instructions, which consist of their corresponding partial vector symbols and partial distances, as depicted in [11]. In Fig. 11, the thick arrows indicate the instructions, for example, $INST\text{ PARE}$ represents an instruction for the next parent node. The detailed description of each functional block is presented next.

1) $CALC\text{ INST}$ calculates the $M$ instructions corresponding to child nodes from a parent node ($INST\text{ CHIL}$), based on a received vector symbol ($RX\text{ SIG}$) and an instruction corresponding to the parent node ($INST\text{ PARE}$). The hardware implementation for calculating these instructions is
shown in Fig. 12, in the case of 64-QAM. In addition to the calculation of instructions, CALCINST sorts the calculated instructions in terms of their distances. This enables the one-node-per-cycle operation by reducing the computational complexity of the subsequent calculations, for example, the update of a list of candidates. Generally, to sort numbers in an ascending (or descending) order is computationally prohibitive. In this case, however, the instructions can be easily sorted, since their distances are dependent on the position of $b_{t+1}$ in the 2-D space, as shown in (13). For example, in the case of 64-QAM, by dividing the 2-D space into 256 rectangular regions and finding which rectangular region $b_{t+1}$ belongs to, the instructions can be easily sorted in an ascending order of partial distance, as shown in Fig. 13: The instructions are sorted in the order of 111101, 110101, 111111, 110111, and so on. The position of $b_{t+1}$ is easily found by finding which rectangular region $(|\text{Re}(b_{t+1})|, |\text{Im}(b_{t+1})|)$ belongs to the first quadruple.

2) CTRLDF controls the tree traversal according to the DF search strategy shown in Fig. 2. To be specific, it chooses either forward or backward recursion (FWD or BWD) for the next clock cycle and generates an instruction for the next parent node (INSTPARE), based on the calculated instructions (INSTCHIL) or the instructions corresponding to the preferred nodes (INSTPREF).

3) STRFWD stores the remaining $(N_T - 1)$ calculated instructions (INSTCHIL(D)) onto the pipeline register (PIPE_REG(2)) if the tree traversal is in forward recursion. It also stores an instruction corresponding to the preferred child node (INSTPREF) among the $(N_T - 1)$ calculated instructions if the child node belongs to the selected promising candidate groups (i.e., VALID = 1).

4) STRBWD selects and stores the preferred instructions (INSTPREF), among the instructions stored previously (in forward recursion) (INSTSTR(D)), if the tree traversal is in backward recursion. Just as in STRFWD, the preferred instructions are stored only if the corresponding node belongs to the selected promising candidate groups (i.e., VALID = 1).

5) UPDLIST updates a list of candidates (LIST) based on the calculated instructions (INSTCHIL) and, subsequently, updates the radius (RAD).
6) **CGSEL** selects a predetermined number of candidate groups (CGSEL) as preprocessing, based on the channel coefficients estimated from the received preamble (CHQDEF).

7) **CTRL** controls the tree traversal based on the probabilistic-search algorithm by determining whether each of the child nodes belongs to the selected promising candidate groups (CGSEL) and setting STR_FWD and STR_BWD knowing it (VALID). This is done by simply keeping track of the number of visited child nodes for each node, as mentioned in Section V.

The proposed VLSI architecture is quite similar to that proposed in [11] and [12], since the proposed P-SD is the same as DF-SD except that the tree traversal is based on the probabilistic-search algorithm as well as the DF search strategy, as aforementioned in Section V. From the perspective of hardware architecture, P-SD can be implemented by simply adding two functional blocks **CGSEL** and **CTRL** to the conventional DF-SD: The other functional blocks (CALCINST, CTRL_DF, STR_FWD, STR_BWD, UPDLIST, and PIPE_REG) are common to the conventional DF-SD. Therefore, it is readily understood that most of the state-of-the-art architectures for DF-SD (e.g., [11], [12]) are also applicable to the P-SD. This is one of the advantages of our proposed SD.

Whenever the nodes at the bottom of tree are evaluated, UPDLIST updates a list of candidates, based on the corresponding distances. To be specific, $N_{CD}$ candidates are newly chosen so as to minimize the distances among $M$ evaluated candidates and $N_{CD}$ previously stored candidates. Since the calculated instructions are already sorted in terms of their distances, the question is on how to merge two separately sorted sets, a set of distances of $M$ evaluated candidates and the other set of distances of $N_{CD}$ previously stored candidates. How to merge two separately sorted sets into a new sorted set is described as shown in Fig. 14. The pseudocodes in Fig. 14 (left side) show how to generate a set of $Q$ sorted numbers $(\text{output}_1, \text{output}_2, \ldots, \text{output}_Q)$ from two separately sorted sets, a set of $P$ sorted numbers $(\text{input}_1, \text{input}_2, \ldots, \text{input}_P)$, and a set of $Q$ sorted numbers $(\text{input}_2, \text{input}_2, \ldots, \text{input}_2)$. For example, when $P$ and $Q$ are equal to 4 and 16, respectively, Fig. 14 (right side) shows how to generate output[6], the sixth number of the set, and the corresponding hardware implementation is described as shown in Fig. 15. One of nine numbers $(\text{input}_1, \text{input}_1, \text{input}_3, \text{input}_4, \text{input}_2, \text{input}_2, \text{input}_2, \text{input}_2, \text{input}_2)$ is chosen as output[6] based on 16 comparisons between two sorted sets. Note that all the necessary comparisons are performed in parallel to enable one-clock-cycle operation.

The required number of comparisons is approximately proportional to $Q^2$, when $P$ is larger than $Q$. Therefore, it is readily understood that most of the state-of-the-art architectures for DF-SD (e.g., [11], [12]) are also applicable to the P-SD. This is done by simply keeping track of the number of visited child nodes for each node, as mentioned in Section V.

![Fig. 13. Sorting of calculated partial distances in the case of 64-QAM. By dividing the 2-D space into 256 rectangular regions and finding which rectangular region $\ell_{i+1}$ belongs to, the instructions can be easily sorted in an ascending order of partial distance.](image1)

![Fig. 14. Merging of two separately sorted sets into a new sorted set. The required number of comparisons is approximately proportional to $Q^2$, when $P$ is larger than $Q$.](image2)

![Fig. 15. Hardware implementation for merging two separately sorted sets ($P = 4, Q = 16$). All the necessary comparisons are performed in parallel to enable one-clock-cycle operation.](image3)
has shown the possibility of efficient hardware implementation [13].

Now, let us describe CGSEL_PROB that is unique to our proposed P-SD. Based on the channel coefficients \( \{ r_{i,d,k} \mid i = 2, \ldots, N_T \} \), this functional block selects the \( K \) promising candidate groups that minimize \( Q(\mathbf{u}) \). As aforementioned in Section V, this can be efficiently performed by utilizing a simplified version of \( K \)-best algorithm [15]–[17]. Each node in the \( (i+1) \)th level is visited as a parent node, and all the child nodes from a parent node (in the \( i \)th level) are evaluated within a single clock cycle (i.e., one-node-per-cycle operation). The corresponding hardware implementation is described as shown in Fig. 16. In addition, the evaluated partial vector symbols \( K \) previously stored partial vector symbols in the level are merged into \( K \) partial vector symbols in the subsequent clock cycle, as shown in Fig. 14. Consequently, it is easily understood that the selection of promising candidate groups takes only 19 clock cycles: A single clock cycle for \( K \) previously stored partial vector symbols and \( K \) previously stored partial vector symbols in the subsequent clock cycle, as shown in Fig. 14. Consequently, it is easily understood that the selection of promising candidate groups takes only 19 clock cycles: A single clock cycle for

The proposed VLSI architecture for four transmit and receive antennas \( (N_T = N_R = 4) \) and 64-QAM \( (M = 64) \) was modeled in Verilog HDL and synthesized with Synopsys Design Compiler using a 0.25-\( \mu \)m CMOS technology. The prelayout simulation results were bit-accurately verified, and the maximum clock frequency of 51 MHz was verified using prelayout static timing analysis. The synthesis results show that the total hardware area amounts to 270.6 kilo gate-equivalents (KGE), and the additional functional blocks, (CGSEL_PROB) and CTRL_PROB, occupy only a small fraction, i.e., 3.8% (10.3 KGE), as shown in Fig. 17. The synthesis results also show that the critical path of 19 ns exists in CALC_INST and CTRL_DF, which ranges from INST_PARE(D) to INST_PARE through INST_CHIL in Pipeline Stage I. Note that CGSEL_PROB belongs to Preprocessing Stage that is independent of the critical path, and CTRL_PROB has a small path delay of 13 ns and a small hardware area of 1.2 KGE (0.4%) in Pipeline Stage II. Therefore, recalling that P-SD can be implemented by simply adding these two functional blocks to DF-SD (e.g., [11], [12]), any state-of-the-art architectures of DF-SD that promise small hardware area and high clock frequency can be readily exploited in the proposed VLSI architecture of P-SD.

Before going to Section VIII, let us consider the required hardware complexity of our proposed SD, when the hardware throughput is given by a wireless standard. In order to achieve a hardware throughput \( TP \) in vector symbols per second with a clock frequency \( f_{clk} \) in hertz, multiple hardware units may need to operate in parallel, and the required number of hardware units is given by \( TP \cdot N / f_{clk} \) [11]. Consequently, the saving of clock cycles can be seen as that of hardware complexity. For example, to achieve a hardware throughput of 12 mega-vector-symbols per second given by IEEE 802.11n [25], the required number of hardware units is equal to 0.24\( N \), assuming a clock frequency of 50 MHz. Therefore, as shown in the previous section, a required SNR of 25 dB is achievable at the expense of 12 hardware units \( (N = 50) \) in P-SD and 48 hardware units \( (N = 200) \) in DF-SD, implying that the proposed P-SD provides the saving of 36 hardware units (75%).

VIII. COMPARISON: PERFORMANCE-COMPLEXITY TRADEOFFS

In this section, the tradeoffs between performance and complexity are presented to verify the advantage of our proposed P-SD. Even though there have been several reports in the literature [10], [12], [27], relatively less attention has been paid to rigorous comparison of performance and complexity in the presence of early termination [19]–[21]. Furthermore, the performance evaluation is highly dependent on both the system parameters [e.g., the number of antennas \( (N_T, N_R) \) and modulation order \( (M) \)] and the channel parameters (e.g., delay spread and maximum Doppler frequency), whereas the complexity analysis is significantly affected by the choice of VLSI architectures (e.g., pipelining and parallel processing). Since a rigorous analysis of performance and complexity of SD is an independent research topic, the focus of this section is on the comparison of the proposed probabilistic-search strategy with conventional DF and breadth-first search strategies. Specifically, the performance is measured by the required SNR for a BER of \( 10^{-4} \), whereas the complexity is measured by the required computational complexity of tree traversal. Note that since the hardware complexity such as hardware area and power consumption highly depends on the VLSI architecture, the computational complexity that is independent of hardware architecture is often considered as a more reasonable measure of complexity. Additionally, note that the preprocessing such as QR decomposition and selection of promising candidate groups, which is carried out even less frequently (e.g., when channel coefficients are changed), is not considered in analyzing the computational
complexity, since it requires only a small fraction of computation. For the purpose of comparison, conventional SD methods, DF-SD in [11]–[13], and K-best algorithm in [16], [17], are considered here. However, since we are focused on the comparison of tree-search strategies, algorithmic modifications proposed in [12], [13], [16], [17] are not considered here.

To facilitate rigorous comparison, the tree traversal of SD is categorized into distance calculation, sorted-set merging, memory access (including write and read operations), sorting, and minimum search. The analysis of computational complexity is summarized in Table I, where $N_i$ denotes the number of clock cycles in the $i$th level, and therefore, it follows that $N = N_1 + N_2 + N_3 + N_4$ and $N_4 = 1$.

Assuming the one-node-per-cycle operation as in [12], [13], [16], the computational complexity of distance calculation is simply proportional to the number of clock cycles $N$, as shown in Table I. Here, $P_{CM}$ denotes the computational complexity of distance calculation for $M$-ary QAM. As readily understood from Fig. 12, the distance calculation includes $(64 + 4N_T)$ additions and 2 multiplications for 16-QAM and $(256 + 4N_T)$ additions and 2 multiplications for 64-QAM for each clock cycle. In more detail, the computational complexity amounts to $(64 + 4N_T)ADD + 2MUL_F + 6MUL_L + 4N_TMUL_2 + 3SQ$ for 16-QAM and $(256 + 4N_T)ADD + 2MUL_F + 16MUL_L + 4N_TMUL_2 + 3SQ$ for 64-QAM, where $ADD$, $MUL_F$, $MUL_L$, $MUL_2$, and $SQ$ denote the computational complexity of addition, full multiplication, constant-coefficient multiplication, variable-coefficient multiplication, and squaring, respectively.

Two separately sorted sets are merged into a new sorted set, based on their distances. While DF-SD and P-SD merge $M$ evaluated candidates and $N_{CD}$ previously stored candidates in the first level to update the list, K-best algorithm additionally merges $M$ evaluated partial vector symbols and $N_1$ previously stored candidates in the second level, and $M$ evaluated partial vector symbols and $N_2$ previously stored ones in the third level. In Table I, $SM_{PQ}$ denotes the computational complexity of sorted-set merging with two parameters $P$ and $Q$. Note that the sorted-set merging is dominated by the number of comparisons. Consequently, the computational complexity approximately amounts to 21 comparisons for $(P, Q) = (64, 6)$, 136 comparisons for $(P, Q) = (64, 16)$, and 2080 comparisons for $(P, Q) = (64, 64)$.

With respect to memory access, DF-SD and P-SD require $N_1$ write operations and $(N_1 - 1)$ read operations of $N_{CD}$ candidates to update a list and $(N - 1)$ write/read operations of $M$ partial vector symbols (i.e., either write operations for forward recursion or read operations for backward recursion). On the other hand, K-best algorithm performs $N_2$ write operations and $(N_1 - 1)$ read operations of $N_{CD}$ candidates to update the list, $N_2$ write operations and $N_2$ read operations of $N_1$ partial vector symbols in the second level, $N_3$ write operations and $N_3$ read operations of $N_2$ partial vector symbols in the third level, and a write operation and a read operation of $N_3$ partial vector symbols in the fourth level. In Table I, $ME_j$ denotes the computational complexity of a single write/read operation of $J$ vector symbols.

While DF-SD and P-SD require the sorting of $M$ evaluated candidates only in the first level to update a list, K-best algorithm requires the sorting of $M$ evaluated partial vector symbols at every level. In Table I, $ST_j$ denotes the computational complexity of sorting $J$ numbers.

DF-SD and P-SD search the minimum distance in order to find a preferred child node among $M$ yet-to-be-visited nodes for the SE ordering, as opposed to K-best algorithm. In Table I, $MS_j$ denotes the computational complexity of minimum search for $J$ numbers.

Based on earlier analysis, the tradeoffs between performance and complexity are evaluated with respect to distance calculation, sorted-set merging, and memory access, as shown in Figs. 18–20, respectively. The number of clock cycles is given as follows: $N = 20, 50, 60, 75, 100, 150$, and 200 for DF-SD, $N = 20, 30, 40, 50, 75, 100, 150$, and 200 for P-SD, and $(N_1, N_2, N_3, N_4) = (6, 6, 6, 1), (6, 6, 16, 1), (6, 16, 16, 1), (16, 16, 16, 1), (16, 64, 16, 1), (16, 64, 64, 1)$, and $(64, 64, 64, 1)$ for K-best algorithm. Here, note that the expectation of computational complexity is taken for both DF-SD and P-SD, since $N_1$ is not constant but randomly varying. It is shown that the proposed P-SD is definitely advantageous in terms of both performance and complexity, when it is compared with
DF-SD. Interestingly, the proposed P-SD is also advantageous over \(K\)-best algorithm, when the ML error performance (i.e., the required SNR of 22 dB) is desired, implying that the probabilistic-search strategy can be considered as an alternative to conventional DF and breadth-first search strategies. For instance, while \(K\)-best algorithm requires 193 clock cycles (distance calculation), 283,248 comparisons (sorted–set merging), and 18,544 write/read operations (memory access) to approach the ML error performance, our proposed P-SD requires 200 clock cycles, 3,604 comparisons, and 13,600 write/read operations. Recalling that the ML detection can be seen as DF-SD without radius reduction and SE ordering earlier (before termination) by simply ignoring less promising candidate groups. Furthermore, even though the proposed P-SD is targeted for LDPC-coded MIMO-OFDM systems, it is readily applicable to many detection methods such as multiuser detection at cellular networks as well as linear constellation precoding.

**Fig. 20.** Tradeoffs between performance and complexity (3): Memory access. The computational complexity is measured by the number of required write/read operations.

**Fig. 19.** Tradeoffs between performance and complexity (2): Sorted–set merging. The computational complexity is measured by the required number of comparisons.

**IX. CONCLUSION**

In this paper, the probabilistic-search algorithm was applied to the conventional DF-SD to improve the error performance. Our proposed P-SD evaluates more promising candidates earlier (before termination) by simply ignoring less promising candidate groups, while maintaining the hardware efficiency of DF-SD. Based on the performance and complexity tradeoffs, it was clearly shown that the proposed P-SD is more efficient than conventional SD such as DF-SD and \(K\)-best algorithm when the ML error performance is desired. Since the proposed P-SD can be implemented by simply adding two functional blocks, it can be generally applied to most state-of-the-art DF-SD architectures reported in the literature. This is one of the advantages of our proposed P-SD in that it can fully exploit any algorithmic and architectural improvements for DF-SD. For example, it can be combined with the MF strategy [19]–[21]: Once the maximum number of clock cycles has been scheduled for a specific vector symbol, the optimum number of promising candidate groups is applied to confine the tree search into more promising candidate groups.

**ACKNOWLEDGMENT**

The authors would like to thank J. Lee and Y. Zhang for their valuable comments. They would also like to thank the anonymous reviewers for their comments and suggestions.

**REFERENCES**


Since 1988, he has been with the University of Minnesota, Minneapolis, where he is currently a Distinguished McKnight University Professor with the Department of Electrical and Computer Engineering. His research addresses VLSI architecture design and implementation of physical-layer aspects of broadband communications systems. He is currently working on error control coders and cryptography architectures, high-speed transceivers, and ultrawideband systems. He has authored over 400 papers, the text book VLSI Digital Signal Processing Systems (Wiley, 1999), and coedited the reference book Digital Signal Processing for Multimedia Systems (Marcel Dekker, 1999).

Dr. Parhi was the recipient of numerous awards including the 2004 F. E. Terman Award by the American Society of Engineering Education, the 2003 IEEE Kiyo Tomiyasu Technical Field Award, the 2001 IEEE W. R. G. Baker Prize Paper Award, and a Golden Jubilee Award from the IEEE Circuits and Systems Society in 1999. He has served on the Editorial Boards of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON VLSI SYSTEMS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE SIGNAL PROCESSING LETTERS, and IEEE SIGNAL PROCESSING MAGAZINE. He served as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS from 2004 to 2005 and serves on the Editorial Board of the Journal of VLSI Signal Processing.

He has served as Technical Program Cochair of the 1995 IEEE VLSI Signal Processing Workshop and the 1996 ASAP Conference and as the General Chair of the 2002 IEEE Workshop on Signal Processing Systems. He was a Distinguished Lecturer for the IEEE Circuits and Systems Society from 1996 to 1998.

Sin-Chong Park (M’97) received the B.S. degree in applied physics from Seoul National University, Seoul, Korea, in 1964, the M.S. degree in physics from Purdue University, West Lafayette, IN, in 1973, and the Ph.D. degree in chemical physics from University of Minnesota, Minneapolis, in 1978.

From 1978 to 1982, he was the Head of the Laser Physics Laboratory, Agency for Defense Development, Daejeon, Korea. From 1983 to 1997, he was the Executive Director of the Semiconductor Division and the Vice President of the Electronics and Telecommunications Research Institute, Daejeon. From 1993 to 1998, he was a National Project Leader of GDRAM Research. Since 1998, he has been a Professor with the School of Engineering, Information and Communications University, Daejeon, where he serves as director of the System Integration Technology Institute. He has authored or coauthored over 330 papers. He is the holder of 37 patents. His research interests include next-generation communications and system-on-a-chip.

Chester Sungchung Park (S’02–M’06) was born in Minnesota. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2000, 2002, and 2006, respectively.

During his doctoral studies, he was a Visiting Researcher at the University of Minnesota, Minneapolis. From 2006 to October 2007, he was with Samsung Advanced Institute of Technology, Yongin, Gyeonggi, Korea, where he conducted research on modulation and coding for wireless modems and high-density memories. Since November 2007, he has been a Senior Staff System Engineer with Ericsson Research, Research Triangle Park, NC. His research interests include communication algorithms and VLSI implementations, focusing on multiple-input–multiple-output orthogonal frequency-division multiplexing and channel coding.