

SPHERE DECODING FOR MULTIPROCESSOR ARCHITECTURES

Q. Qi, C. Chakrabarti

Arizona State University
Department of Electrical Engineering
{qi,chaitali}@asu.edu

ABSTRACT

Motivated by the need for high throughput sphere decoding for multiple-input-multiple-output (MIMO) communication systems, we propose a parallel depth-first sphere decoding (PDS) algorithm that provides the advantages of both parallel processing and rapid search space reduction. The PDS algorithm is designed for efficient implementation on programmable multi-processor platforms. We investigate the trade-off between the throughput and computation overhead when the number of processing elements is 2, 4 and 8, for a 4×4 16-QAM system across a wide range of SNR conditions. Through simulation, we show that PDS can offer significant throughput improvement without incurring substantial computation overhead by selecting the appropriate number of processing elements according to specific SNR conditions.

Index Terms— Sphere Decoding, Architecture, Multiprocessor

1. INTRODUCTION

The application of wireless devices has become ubiquitous in recent years. The increasing demand of robust and high throughput mobile systems has spear-headed the development of multiple-input multiple-output (MIMO) communication systems. A MIMO antenna array [1][2] coupled with orthogonal frequency division multiplexing (OFDM) has become the defacto choice for designing high bandwidth capacity and spectral efficiency communication standards such as IEEE 802.16e and 802.11n.

The performance gain of a MIMO system comes at the cost of increasing design complexity. The signal detector is one of the most important modules in a MIMO system. Maximum-likelihood (ML) detectors are impractical for high data rate MIMO systems, since their complexity increases exponentially with signal dimension. Active research on low-complexity and near ML MIMO detectors have generated several solutions, including zero-forcing equalization (ZF) [3], nulling and canceling (NC) [1], semidefinite relaxation (SR) [4][5] and sphere decoding (SD) [6]. Of these approaches, the SD algorithm is the most promising; it offers low complexity and good bit-error-rate (BER) performance under a variety of Signal-to-Noise (SNR) and constellation conditions [7].

With the emergence of wireless networks for different bandwidth and mobility scenarios, implementing the physical layer on a programmable hardware platform, such as software defined radio (SDR), is very attractive due to lower development cost, ease of verification and application versatility [8]. Current VLSI implementation of SD detectors, such as [9][10], are ASIC designs that increase the throughput, but are limited to specific communication protocol configurations and setups. In this paper, we present a design study

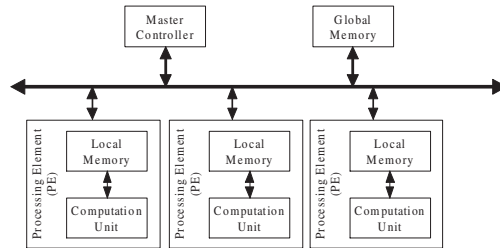


Fig. 1. Multi-core Software Radio Architecture.

for a parallel depth-first SD detector on the multiprocessor SDR platform shown in Figure 1. The main contributions of this paper are listed below.

1. We introduce a *parallel depth-first sphere decoding* (PDS) algorithm, and investigate its performance under different SNR and parallelization parameters.
2. We map the PDS on to a programmable multiple *processing element* (PE) hardware platform to providing faster decoding speed.
3. We investigate the performance of PDS with 2, 4 and 8 PEs for SNR ranging from 8dB to 22dB. We found that for a 4×4 16-QAM PDS system with $\text{SNR} \leq 16\text{dB}$, a 4-PE architecture provides near 3X throughput increase, while incurring only a small computation overhead.

This paper is organized as follows. We briefly describe a MIMO system in Section 2, followed by a review of the sphere decoding algorithm and existing VLSI implementations in Section 3. Section 4 presents the PDS algorithm and proposes a multiprocessor architecture implementation. Section 5 provides algorithm simulation results. The conclusion is given in Section 6.

2. PRELIMINARIES

A MIMO system with spatial multiplexing signaling is shown in Figure 2. It consists of M_T transmit and M_R receive antennas. Let y be the $M_R \times 1$ vector of received symbols, given by

$$y = Hs + n \quad (1)$$

where H is an $M_R \times M_T$ complex channel matrix with h_{ij} representing the complex transfer function from the j th transmit antenna to the i th receive antenna, $n = [n_1, n_2, \dots, n_{M_R}]^T$ is an $M_R \times 1$ noise vector, and $s = [s_1, s_2, \dots, s_{M_T}]^T$ is an $M_T \times 1$ vector of transmitted symbols. Each h_{ij} in H is an independent and identically distributed (i.i.d.) complex Gaussian variable with zero mean and 0.5 variance. Each n_i in n is also an i.i.d complex Gaussian

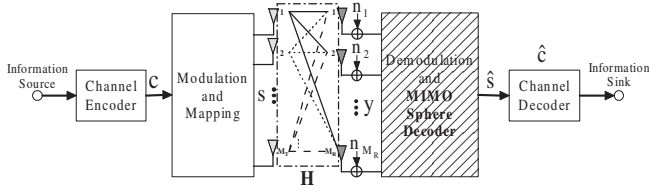


Fig. 2. Block diagram of a MIMO communication system.

variable with zero mean and σ^2 variance (σ^2 is calculated according to receiving SNR). Let x denote the source data bits. A transmitted symbol s_i is obtained by mapping every $M_c = \log_2(M)$ bits of x on to a complex constellation of size M .

Throughout this paper, we assume $M_R \geq M_T$ and channel matrix H is perfectly known to the receiver through training sequence estimation. The entries in the transmitted symbol vector s are constellations in an M -quadrature amplitude modulation (QAM). Furthermore, the transmitted data is uncoded and the transmission rate is $R = M_T M_c$ bits per effective channel use. An ML MIMO detector minimizes the BER by returning the optimal estimation of the transmitted symbol s^* that satisfies the following function.

$$s^* = \underset{s \in \mathbb{O}}{\operatorname{argmin}} \|y - Hs\|^2 \quad (2)$$

where $\mathbb{O} = M^{M_T}$ denotes all possible transmitted symbol vectors, which resides in an M_T -dimensional square lattice spanned by an M -QAM constellation in each dimension. While an ML MIMO detector provides the optimal solution, its complexity grows exponentially with the lattice dimension and the constellation size. For instance, an ML detector must consider 16, 777, 216 candidate symbol vector (for a MIMO system with $M_T = M_R = 4$ antennas and 64-QAM modulation). A SD detector on the other hand, can find the exact solution to equation (2) in polynomial time for high SNRs by searching the minimum squared Euclidean distance to a received vector y within a hypersphere of radius r over a M -dimensional finite discrete set [6]. The complexity of SD only grows to exponential when the search dimension (M_T -dimensional square lattice) is very large [11] or the SNR is very low.

3. SPHERE DECODING ALGORITHMS AND IMPLEMENTATIONS

In this section, we first explain the mathematical fundamentals of the SD algorithm. We then briefly describe the different search methods for sphere decoders. The pros and cons of the different methods are listed to better illustrate the design decisions behind the proposed parallel depth-first sphere decoding method.

To enumerate points inside a hypersphere bounded by radius r , the SD algorithm first performs the QR decomposition of the channel matrix H , where $H = QR$. The resulting Q is an $M_R \times M_R$ orthogonal matrix, and R is an $M_T \times M_T$ nonsingular upper triangular matrix [7]. The minimization equation in (2) then undergoes orthogonal transformation, and the resulting equation is shown below.

$$s^* = \underset{s \in \mathbb{O}}{\operatorname{argmin}} \|\hat{y} - Rs\|^2 \quad (3)$$

where $\hat{y} = Q^\dagger y = Rs^{ZF}$, s^{ZF} is the ZF solution, Q^\dagger is the pseudoinverse of Q , and \hat{y} is also an upper triangular matrix composed of M_T elements of the orthogonally transformed received symbols.

For description brevity and convenience, we use the following mathematical notations. In a matrix A , a_{ij} denotes the i th row and

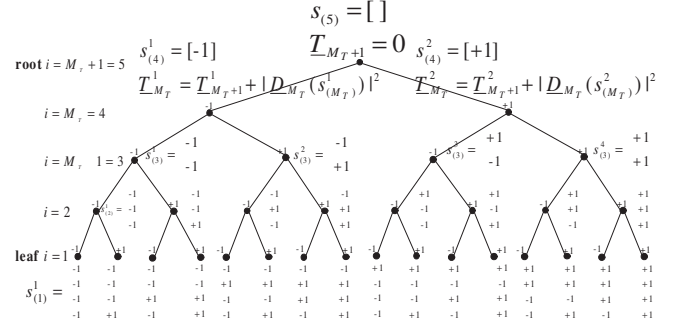


Fig. 3. A recursive weighted M -ary tree ($M = 2$) constructed using Equation (4).

j th column element. In a vector z of length N , $z_{(k)}$ denotes the *partial vector* with elements z_k through z_N , where z_i is the i th element. A *metric value* associated with partial vector $z_{(k)}$ is denoted as $\underline{C}_k(z_{(k)})$. We also use superscript to denote the candidate index of a particular value. For example, $z_{(k)}^j$ denotes the j th candidate partial vector $z_{(k)}$.

Since the matrices of equation (3) on the right hand side are all upper triangular, the following decomposition can be carried out recursively, starting from the M_T th row and M_T th column element.

$$\begin{aligned} \|\hat{y} - Rs\|^2 &= \sum_{i=M_T}^1 \left| \hat{y}_i - \sum_{j=M_T}^i r_{ij} s_j \right|^2 \\ &= \sum_{i=M_T}^1 \left| \hat{y}_i - \sum_{j=M_T}^{i+1} r_{ij} s_j - r_{ii} s_i \right|^2 \\ &= \sum_{i=M_T}^1 \left| \underline{V}_{i+1}(s_{i+1}) - r_{ii} s_i \right|^2 \\ &= \sum_{i=M_T}^1 \left| \underline{D}_i(s_{(i)}) \right|^2 \end{aligned} \quad (4)$$

where $s_{(i)} = [s_i s_{i+1} \dots s_{M_T}]^T$ denotes a *partial vector symbol* candidate, \underline{V}_{i+1} is the corresponding *residual vector metric* and $\underline{D}_i(s_{(i)})$ is the corresponding *branch metric*.

An M -ary tree, shown in Figure 3, can be built using equation (4) from the root level ($i = M_T + 1 = 5$) to the leaf level ($i = 1$). Here, every parent node has two child nodes ($s_i \in \{-1, 1\}$). The *partial Euclidean distance metric* (PED) [9] along a path from the root node to a node at level i is given by $\underline{T}_i = \underline{T}_{i+1} + |\underline{D}_i(s_{(i)})|^2 = \sum_{j=M_T}^i |\underline{D}_j(s_{(j)})|^2$. At the root node, the initial partial vector symbol $s_{(5)}$ is null and the associated PED \underline{T}_{M_T+1} is 0. At level $i = 4$, there are two child nodes $s_4 = +1$ and $s_4 = -1$, and the two possible partial vector symbols are $s_{(4)}^1 = [+1]$ and $s_{(4)}^2 = [-1]$. The PED of each partial vector symbol ($\underline{T}_4^1 = \underline{T}_4(s_{(4)}^1 = [+1])$ or $\underline{T}_4^2 = \underline{T}_4(s_{(4)}^2 = [-1])$) is based on its branch metric ($\underline{D}_4^1(s_{(4)})$ or $\underline{D}_4^2(s_{(4)})$) and parent PED \underline{T}_{M_T+1} . When the tree expands from level $i+1$ to level i , the length of a partial vector symbol $s_{(i)}^j$ expands by one more symbol s_i , and its cardinality increases by a factor of 2. At the leaf level ($i = 1$), the tree is fully expanded, and there are $2^4 = 16$ candidate symbol vectors. Since an M -ary tree is determined by y , H and \mathbb{O} , it has the following properties:

1. The branch metric of each edge in the tree is always *non-negative*.
2. Each node in the tree, except the leaf nodes, has M children due to the M -ary complex constellation constraint.
3. A candidate solution to Equation (4) is the accumulative PED of a route from the root node to a leaf node.

Property 1 implies that the PED is *monotonically non-decreasing* as a route traverses from level i to level $i - 1$. Property 1 and 3 together show that the ML solution s^* in Equation (2) is the $s_{(1)}^j$, $j \in \mathbb{O}$ symbol vector associated with the minimum PED among all leaf nodes. Furthermore, Property 2 implies that all sibling nodes at level $i - 1$ use the same PED \underline{T}_i when updating their individual \underline{T}_{i-1} . Even though the number of possible routes increases exponentially as M_T and M increase, these properties coupled with efficient tree pruning methods and properly defined search radius r can significantly reduce the ML solution computation to polynomial time [7].

The foremost tree pruning enumeration algorithm was introduced by Fincke and Pohst, and is known as the *F-P* enumeration in [12]. The F-P method traverses the search tree *depth-first* from the root node to leaf nodes by enumerating all child nodes in the order of increasing cardinality. It keeps any route with PED bounded by an initial hypersphere with radius r ($\underline{T}_i \leq r^2$), and terminates the ones that violate this constraint. Once a route reaches a leaf node, it returns a solution candidate $s_{(1)}$. After all solution candidates are found, the ML solution is simply the one with the least PED. If no solution is found, it means the initial radius r is too small and needs to be expanded. The F-P enumeration is repeated again until a solution is found.

The F-P strategy neither addresses the priority order of child nodes during tree pruning, nor specifies the criteria of selecting the proper initial radius r . Both problems are alleviated in the Schnorr-Euchner (*S-E*) enumeration in [13]. The tree is pruned depth-first and child nodes at level i are prioritized in the increasing order of PEDs. The ML solution is much more likely to reside in the leaf nodes of the left subtrees. Further more, the initial search radius r is set to infinity, and is updated with the PED of each new candidate solution, provided it is less than the current r^2 . The S-E enumeration finds eligible solutions faster than the F-P enumeration, and is the foundation of depth-first SD extensions. The ASIC implementation of modified S-E decoders are presented in [9] and [14].

Since the depth-first based SD algorithm is inherently nonparallel, it has a nondeterministic throughput. An alternative breadth-first based K-best SD is introduced in [10] to provide uniform data path and constant throughput for all SNR conditions. During tree pruning, the K-best algorithm first performs *path extension*, where each of the L survivor paths from the previous depth find M child nodes. The *radius check* procedure then examines each extended path and deletes the ones that violate the radius r^\ddagger , which is often defined as $\gamma\sigma^2 M_R$ [15]. Finally, the *path search* (PS) step is performed to limit the number of surviving paths to $L = \min(L, K)$ routes, where K is the maximum number of survivor paths allowed. The most well known hardware implementation of the K-best SD decoders include [15] and [16].

A comparison of hardware implementations for sphere decoding in [9] shows that the different designs are fine tuned for specific antenna configurations and modulation schemes. Their lack of flexibility limits their use in applications that requires multiple protocols. Recent developments in SDR display real promise in meet-

[‡]The value $\gamma \geq 1$ is a confidence constant found through empirical experiment, which provides reasonable assurance that ML solution s^* is within the radius r hypersphere.

ing high computational performance requirement, while providing programmable functionality through concurrent algorithm execution and multi-core scheduling. Mapping sphere decoding algorithm on to a multi-processor architecture becomes increasingly attractive.

The depth-first SD algorithm is inherently sequential, since it can only expand one node at a time. In comparison, the K-best algorithm can better utilize the multiprocessor architecture; its performance is limited by the number of PEs and the speed of the sorting operation. To take advantage of the parallelism offered by the K-best SD algorithm and the rapid radius reduction achieved by the depth-first algorithm, we propose a new algorithm that builds on the concepts of both these algorithms.

4. PARALLEL DEPTH-FIRST SPHERE DECODING

4.1. Algorithm Description

Parallel Depth-first Sphere Decoding (PDS) combines the advantages of both the depth-first SD and the K-best SD algorithm to achieve higher throughput. The performance improvement of the decoder comes from running *parallel non-overlapping subtree depth-first pruning* and enforcing *global radius update*. The PDS consists of the following steps:

1. *Subtree Block Assignment* (SBA): The PEDs of M candidate nodes at level M_T are calculated and sorted in increasing order ($\underline{T}_{M_T}^1 \leq \underline{T}_{M_T}^2 \leq \dots \leq \underline{T}_{M_T}^M$). Each candidate node is a root node of a subtree. There are $B = \frac{M}{q}$ blocks, where every q consecutive nodes are assigned in a block P_i , $i \leq B$.
2. *Parallel Subtree Pruning* (PSP): At level M_T , all nodes within the same block P_i are fetched from memory at once for parallel subtree pruning. Nodes in block P_i are accessed before nodes in block P_j , where $i < j$. The q subtrees rooted at level \underline{T}_{M_T} in block P_i are then parallel pruned using depth-first SD algorithm.
3. *Global Radius Update* (GRU): A subtree P_i updates the global radius r , whenever it finds a solution candidate $s_{(1)}^{(P_i)}$ (l is the solution candidate count) that is within current hypersphere and has smaller PED.

The logical flow of the PDS algorithm is shown in Figure 4(b). The SD tree consists of multiple subtrees; the root of each subtree is shown as light grey color-filled nodes in Figure 4(a). The subtree block assignment operation first sorts the subtree root nodes in ascending order of their PEDs, and then groups every q adjacent subtree root nodes into a block. The parallel tree pruning operation fetches all subtree root nodes in the same block at once, along with their PEDs. Each subtree root node initiates a depth-first subtree pruning operation. There are at most q subtree pruning operations running in parallel at any given time, and each one has to keep track of 5 pieces of information at each level: the partial vector symbol $s_{(i)}$, residual vector \underline{V}_{i+1} , the cardinality of last visited nodes at each subtree level $c_{(i)}$, vector of next-to-visit nodes at each level $\tilde{s}_{(i)}$ and the partial Euclidean distance \underline{T}_i .

The parallel tree pruning module shown in Figure 4(b) has two shaded blocks corresponding to the subtree PED calculation and radius r update calculation. These are the speed bottlenecks for depth-first search. Note that each subtree performs the depth-first search independently, but updates the radius r globally. For example, if subtree j and subtree k in block P_i finds new solution candidates vectors which can reduce the global radius to r_j and r_k respectively at the same time, the global radius r is updated by taking the minimum of the two, $r = \min\{r_j, r_k\}$. The newly updated r is then

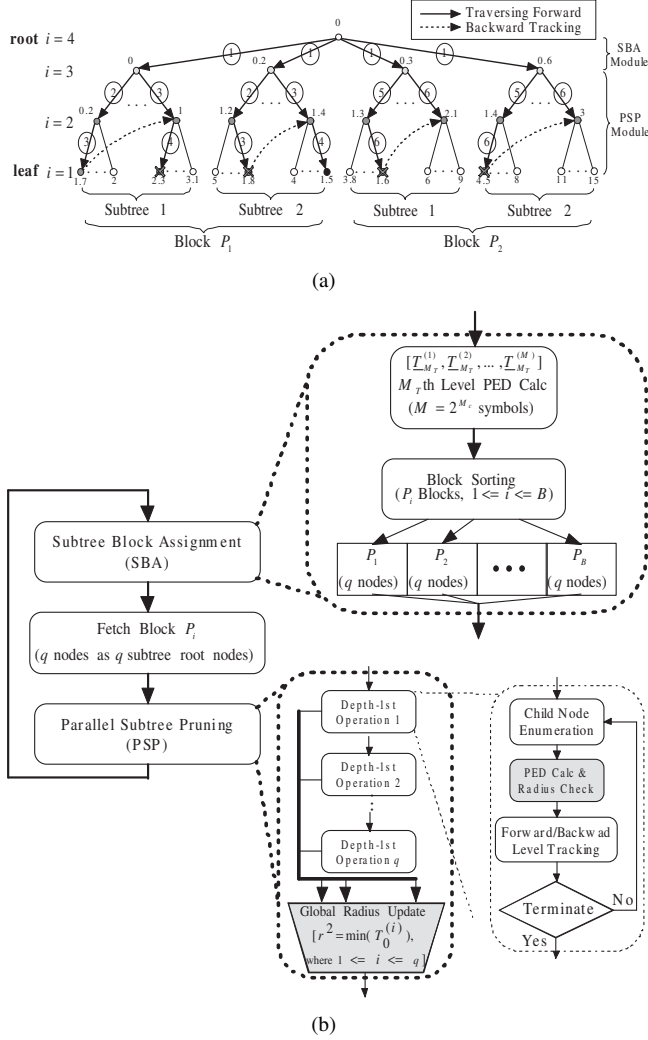


Fig. 4. Parallel Depth-first Sphere Decoder: (a) A Tree Pruning Example and (b) Logic Flow Diagram.

enforced on both subtree j and k for further processing. This helps to rapidly shrink the global radius r , and in turn minimize the number of nodes visited by each subtree. This global radius update strategy trims down the computation overhead incurred by the parallel execution of depth-first search on subtrees. Furthermore, the subtree block assignment operation arranges blocks of subtrees with higher probability of returning the ML solution s^* to be pruned first. This further decreases the number of nodes visited by the subtrees residing in a high index block.

The tree pruning example shown in Figure 4(a) is for a 3×3 4-QAM system, where each node is labeled by its weight, and each traversed edge is associated with a visiting order number. Some child nodes after level $i = 3$ are omitted for figure compactness. The visited nodes are color-filled, and crossed out leaf nodes are candidate solutions computed by the detector but rejected due to their PEDs exceeding the most recent hypersphere bound r^2 . The search is completed when all paths from the root node to each leaf node is either visited or discarded due to r^2 violation. The final solution is the path that reaches the black color-filled node.

4.2. Architecture Mapping

A simplified block diagram of the target multiprocessor architecture is shown in Figure 1. It consists of a master controller, a global memory module and multiple PEs, all connected through a shared data bus. The master controller distributes the computation load among different PEs. Each PE performs tasks, such as residual vector calculation, PED calculation, and PED sorting. The global memory holds common variables, such as the ML candidate result s^* , global radius r , receiving signal \hat{y} , triangular matrix R , subtree root nodes and their PEDs. Assuming each complex number can be stored in a word, the global memory holds at least $2M_T + M_T^2 + 2M + 1$ words. Each PE has a local memory to hold at least $M_{T-1}(2M + 1) + 2$ words for storing results of child nodes cardinality, subtree levels, intermediate PEDs and partial residual vectors.

The initial ML candidate s^* is null, and the initial global radius r is infinity. A new sphere decoding cycle begins when the previous ML candidate s^* is found, and new \hat{y} and matrix R are received from preprocessing calculation. We assume that the preprocessing computation has been done, and only consider mapping the PDSB decoder logic flow shown in Figure 4(b) onto the multi-core architecture in Figure 1. The architecture mapping is as follows.

- After the preprocessing computation, the master controller assigns q PEs to perform M subtree root node (level M_T) PED calculation, where M is the constellation size. Each PE calculates $B = \frac{M}{q}$ PEDs. The results of M PEDs are approximately sorted by the master controller into B blocks. The sorting is such that any PED in block i is smaller than any PED in block j provided that $1 \leq i < j \leq B$. PEDs within a block are not sorted. The master controller then stores the approximately sorted subtree PEDs and their associated symbols in the global memory.
- The master controller assigns each PE a subtree root node from block i , starting with the first block. The master controller allocates a subtree root node from the next block to a PE whenever the PE finishes pruning its current subtree. The master controller repeats this operation until either all subtrees are pruned or a subtree root node's PED violates the global radius constraint.
- A PE prunes a subtree by running depth-first search. A subtree is M_{T-1} level deep and each parent node has at most M child nodes. A PE must track the intermediate PEDs and associated partial residual vectors to perform proper forward and backward depth-first recursion.
- Whenever a PE finds a leaf node, it compares the resulting PED with the global radius constraint r^2 , and updates the global radius r and the ML candidate s^* if necessary. After the global radius update, the master controller broadcasts new r to all PEs.

Increasing the number of PEs can increase the number of parallel depth-first subtree search operations at the cost of larger chip area and greater calculation overhead. It begs the question that how many depth-first submodules should be put in the parallel subtree pruning module and under what channel conditions. Furthermore, the multiple cores are connected by bus, and frequent simultaneous global radius by multiple cores might reduce the throughput gain. In the following section, we will provide some answers to these questions through simulations.

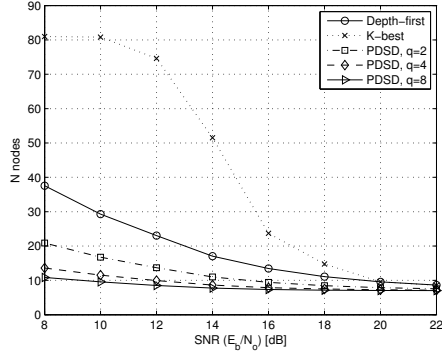


Fig. 5. Performance comparison: maximum number of nodes N , visited by any processor as a function of SNR.

5. SIMULATION RESULTS

In this section, we compare the hard-output performance of the PDSD, S-E depth-first and radius defined K-best algorithms through Monte-Carlo simulations for a 4×4 16-QAM uncoded MIMO system. Following the definition in [15], SNR is measured as:

$$\frac{E_b}{N_o} |_{dB} = \frac{E_s}{N_o} |_{dB} + 10 \log_{10} \frac{M_R}{M_T M_c} \quad (5)$$

where E_b is the energy per transmitted information bit, and E_s is the average QAM symbol energy per receive antenna. The SNR is varied from 8dB to 22dB. The S-E depth-first implementation sets the initial radius to infinity, and can only visit one node at a time. For the K-best implementation, there are at most K child nodes surviving at each stage, and the radius constraint is set to $\gamma \sigma^2 M_R$, $\gamma = 6$.

The main objective of this simulation is to compare the throughput of the aforementioned cases. We use N_i^j to represent the average number of nodes in subtree i that are visited by processor j . The total number of nodes visited by processor j is $\sum_{i \in S(j)} N_i^j$, where $S(j)$ is the set of subtrees that are assigned to processor j . We define N to represent the maximum number of nodes that are visited by any processor. If we assume that it takes a constant time to visit each node, then N can be used to represent the throughput. In reality, the processing time of the first child node is actually higher due to sorting of its nearby sibling PEDs.

In order to make a fair comparison, we set up all SD algorithms to have identical BER performance across the SNR range. The depth-first algorithm with infinite initial radius r provides ML solution. It has only one PE. For the K-best implementation, we choose $K = 32$ and assume that there are 32 PEs available. For the PDSD implementation, we consider three cases: the number of PEs (q) is set to 2, 4 and 8 respectively.

Figure 5 shows the average number of N over a wide range of SNR for all five cases. Figure 6 compares the number of nodes visited in each subtree under three different SNR conditions for all five cases. From both results, the PDSD shows the following properties.

1. All three PDSD implementations outperform the depth-first and the K-best implementations for $SNR \leq 18dB$. For example, at $SNR = 8$, the PDSD with 8 PEs is almost 3 times faster than the depth-first algorithm and 8 times faster than the K-best algorithm. The PDSD outperforms the depth-first because of the speed gain from parallel subtree depth-first search. The PDSD outperforms the K-best due to the rapid

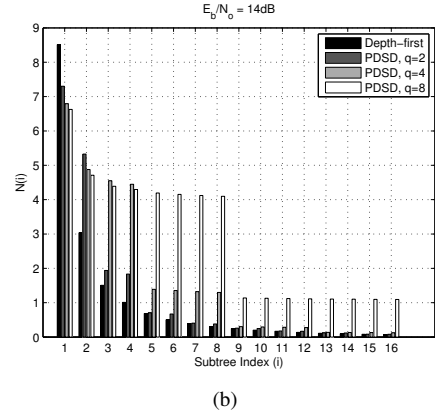
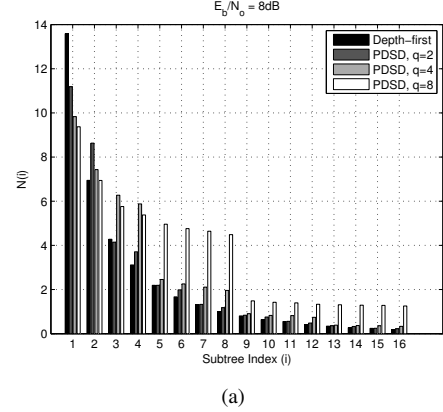


Fig. 6. Average number of nodes visited in each subtree, $N_{(i)}$: (a) at SNR = 8dB and (b) at SNR = 14dB.

radius r reduction, which in turn reduces the number of nodes visited.

2. The speed increase obtained by the PDSD over the depth-first implementation diminishes as the number of parallel depth-first PEs (q) increases under the same SNR condition. When $q = 2$ and $SNR = 8$, the speed of PDSD is almost double that of the depth-first case. Yet, when $q = 8$ and $SNR = 8$, the speed of PDSD is only thrice compared to the depth-first case (rather than 8 times). This is due to the computation overhead incurred by the 1st block of subtrees, where q PEs must prune q subtrees from level M_{T-1} to 1 at least once before updating the initial radius (set to infinity). The speed gain of PDSD is not linear with respect to the number of PEs, because the calculation overhead incurred through increasing the number of PEs reduces the speed gain obtained through parallel pruning.
3. The speed gain obtained by the PDSD over the depth-first and the K-best implementation diminishes as SNR increases. This is because as SNR increases, the noise disturbance on received symbols is reduced greatly. Reduced noise disturbance leads to lower probability of the ML symbol vector s^* residing in subtrees which do not hold the ZF solution. Therefore, the overhead incurred by depth-first subtree search in the PDSD is amplified. This result is confirmed by comparing the number of nodes visited by each subtree across different SNR conditions, as shown in Figure 6(a) and 6(b).

Next we evaluate the throughput gain and the computational overhead as a function of the number of processors. All comparisons are with respect to the depth-first case. Increasing the number of PEs increases the throughput, but also incurs additional computation overhead. Figure 7 illustrates the performance trade-off between throughput gain and extra computation incurred by visiting additional nodes for SNR = 8dB and SNR = 14dB. It shows that for SNR = 8dB, the throughput gain increases by 3X when the number of processors increases from $q = 2$ to $q = 8$. In contrast, there is only a mild throughput gain when the number of processor increases from $q = 2$ to $q = 4$ for SNR = 14dB. In general, for high SNR scenario, the effectiveness of parallel pruning is reduced, and there is no benefit in having larger number of active PEs. By examining the performance trade-off curves across all SNR scenarios, we choose $q \leq 4$ for SNR ≤ 16 dB. For SNR > 16 dB, it is sufficient to set $q = 1$, which corresponds to the original sequential depth-first MIMO detector.

6. CONCLUSION

In this paper, we introduced a hybrid sphere decoding algorithm PDSB for multiprocessor architecture based on a combination of K-best and depth-first algorithms. We investigated PDSB algorithm performance as a function of the number of processors. We demonstrated through simulation that 2 to 4 multiprocessor PDSB implementation is nearly 2 to 3 times faster than the depth-first SD algorithm for 4×4 16-QAM MIMO detector. In this study, we validated the theoretical performance of PDSB. In the next phase, we will investigate issues in the actual implementation of PDSB on a specific multi-core processor platform.

7. REFERENCES

- [1] Foschini, G.J., "Layered space-time architecture for wireless communication in a fading environment when using multi-element antenna," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.
- [2] Telatar, I.E., "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecommun.*, vol. 10, pp. 585–595, Nov. 1999.
- [3] Grotschel, M., Lovász, L. and Schriver, A., *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, 2nd edition, 1993.
- [4] Mobasher, A., Taherzadeh, M., Sotirov, R. and Khandani, A.K., "A near maximum likelihood decoding algorithm for MIMO systems based on semi-definite programming," in *IEEE International Symposium on Information Theory (ISIT)*, Sept. 2005, pp. 1686–1690.
- [5] Sidiropoulos, N.D. and Luo, Z.-Q., "A Semidefinite Relaxation Approach to MIMO Detection for High-Order QAM Constellations," *IEEE Signal Processing Letters*, vol. 13, no. 9, Sept. 2006.
- [6] Pohst, M., "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," in *SIGSAM Bull.*, Feb. 1981, vol. 15, pp. 37–44.
- [7] Hassibi, B. and Vikalo, H., "On the sphere-decoding algorithm I, Expected complexity," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.
- [8] Lin, Y., Lee, H., Woh, M., Harel, Y., Mahlke, S., Mudge, T., Chakrabarti, C. and Flautner, K., "SODA: A Low-power Architecture For Software Radio," in *ISCA '06*, June 2006, pp. 89–101.
- [9] Burg, A., Borgmann, M., Wenk, M., Zellweger, M., Fichtner, W. and Bolcskei, H., "VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

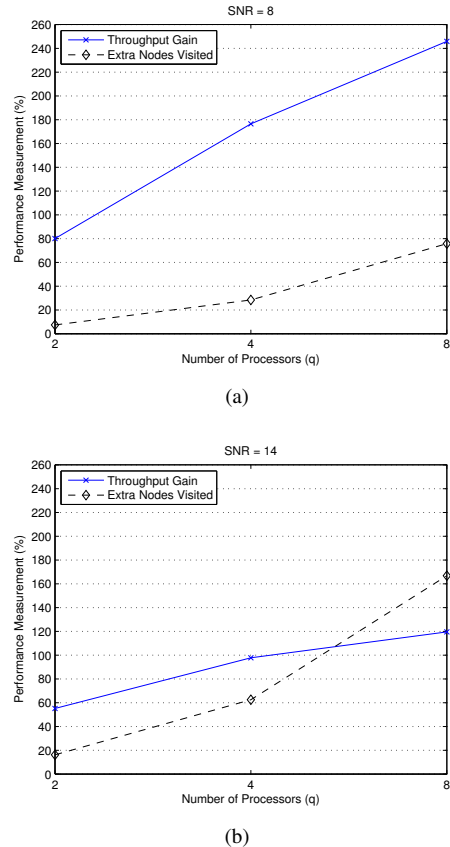


Fig. 7. Performance trade-off between throughput gain and additional number of nodes visited compared to the depth-first case: (a) SNR = 8dB and (b) SNR = 14dB.

- [10] Guo, Z. and Nilsson, P., "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 44, no. 3, pp. 491–503, March 2006.
- [11] Jalden, J. and Ottersten, B., "An exponential lower bound on the expected complexity of sphere decoding," in *ICASSP '04*, May 2004, vol. 4, pp. 393–396.
- [12] Fincke, U. and Pohst M., "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, pp. 161–163, April 1985.
- [13] Schnorr, C.P. and Euchner, M., "Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems," in *Fundamentals of Computation Theory*, 1991, pp. 68–85.
- [14] Park, S, Lee, K, Zhang, Y, Parhi, K.K., Lee, J. and Park, S.-C., "Probabilistic List Sphere Decoding for LDPC-Coded MIMO-OFDM Systems," in *ICASSP '06*, May 2006, vol. 3, pp. III–912 – III–915.
- [15] Hochwald, B.M. and Brink, S., "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, March 2003.
- [16] Chen, S., Zhang, T. and Goel, M., "Relaxed Tree Search MIMO Signal Detection Algorithm Design and VLSI Implementation," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, May 2006, p. 4 pp.