IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 10, 2017, accepted February 27, 2017, date of publication March 20, 2017, date of current version April 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2684706

Geosphere: An Exact Depth-First Sphere Decoder Architecture Scalable to Very Dense Constellations

GEORGIOS GEORGIS[1], KONSTANTINOS NIKITOPOULOS[1], (Member, IEEE), AND KYLE JAMIESON[2,3]

[1]5G Innovation Centre, Institute for Communication Systems, University of Surrey, Guilford GU2 7XH, U.K.
[2]Department of Computer Science, Princeton University, Princeton, NJ 08544 USA
[3]Department of Computer Science, University College London, London WC1E 6BT, U.K.

Corresponding author: G. Georgis (g.georgis@surrey.ac.uk)

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council, EPSRC Grant under Agreement EP/M029441/1, in part by the European Research Council under the EU's Seventh Framework Programme under Grant FP/2007-2013, and in part by ERC Grant under Agreement 279976. This paper was presented at the Proceedings of the 2014 ACM SIGCOMM Conference.

ABSTRACT This paper presents the algorithmic design, experimental evaluation, and very large scale of integration (VLSI) implementation of Geosphere, a depth-first sphere decoder able to provide the exact maximum-likelihood solution in dense (e.g., 64) and very dense (e.g., 256, 1024) quadrature amplitude modulation (QAM) constellations by means of a geometrically inspired enumeration. In general, linear detection methods can be highly effective when the multiple input, multiple output (MIMO) channel is well-conditioned. However, this is not the case when the size of the MIMO system increases and the number of transmit antennas approaches the number of the receive antennas. Via our wireless open access research platform (WARP) testbed implementation, we gather indoor channel traces in order to evaluate the performance gains of sphere detection against zero-forcing and minimum mean-square errors (MMSE) in an actual indoor environment. We show that Geosphere can nearly linearly scale performance with the number of user antennas; in 4 × 4 multi-user MIMO for 256-QAM modulation at 30-dB SNR, there is a 1.7 × gain over MMSE and 2.4 × over zero-forcing and a 14% and 22% respective gain in 2 × 2 systems. In addition, by using a new node labeling-based enumeration technique, low-complexity integer arithmetic, and fine-grained clock gating, we implement for up to 1024-QAM constellations and compare in terms of area, delay, power characteristics, the Geosphere VLSI architecture, and the best-known best-scalable exact ML sphere decoder. Results show that Geosphere is twice as area-efficient and 70% more energy efficient in 1024-QAM. Even for 16-QAM, Geosphere is 13% more area-efficient than the best-known implementation for 16-QAM, and it is at least 80% more area-efficient than the state-of-the-art K-best detectors for 64-QAM.

INDEX TERMS Wireless communication, MIMO, application specific integrated circuits, sphere decoding, VLSI implementation.

I. INTRODUCTION

Multi-user, multiple-input multiple-output (MIMO) systems with spatial multiplexing constitute one of the most promising techniques to to address the ever-increasing demand for throughput while retaining the level of bandwidth usage. This is because, at least in theory, such systems can scale capacity with the number of user antennas [1]. However, in order to translate the theoretically predicted capacity gains into actual throughput, efficient methods are required to detect and demultiplex the mutually interfering information streams at the receiver side. In this direction, frequently employed solutions involve linear detectors like the *zero-forcing* (ZF) and the *minimum-mean-square-error* (MMSE) approaches. However, it is well-known in the literature [2], [3] that these methods are highly sub-optimal in cases where the MIMO channel is poorly conditioned [4], as often occurs when the number of transmit antennas approaches the one of the receive antennas. In Section VI-A, we evaluate this performance loss, in terms of achievable spectral efficiency, for the ZF and MMSE detectors via

simulations based on actual channel traces gathered using our WARP testbed implementation, and we show that ZF and MMSE detection cannot consistently increase network throughput when increasing the number of concurrently transmitting (single-antenna) users up to the number of receive antennas. To increase throughput when numbers of user-antennas approach the number of antennas at the receiver side, maximum-likelihood (ML) detection should be applied, and can be efficiently realized by means of sphere decoding.

Sphere Decoders (SD) [5], [6] avoid exhaustively searching for the ML solution by transforming the detection problem into a tree search. Sphere Decoding-based MIMO detection is not new and as such has been well-examined throughout the literature [5]–[14]. The merits of sphere-decoding-based detection have been documented in theoretical results [5], [14], [15] and supported by various efficient implementations [7], [16]–[20]. Still, while actual channel measurements are already present in the literature ([21]–[23]), the exploration of the practical throughput gains of SD have been limited. For instance, in [24] the capacity of MIMO channels in the downlink has been evaluated, but without accounting for the actual achievable throughput of specific methods. Also, Suzuki *et al.* [25] evaluated ZF against a ''list sphere decoding'' approach, their focus was on the effects of transmitter noise correlation on the error rate probability.

While recent [26] and upcoming wireless standards [27], dictate modulation schemes with very dense constellations, to the best of the authors' knowledge, no ''exact'' SD architecture has been yet proposed that is able to guarantee the ML performance when very dense QAM constellations are transmitted. The main reason is that the proposed tree *enumeration* approaches for exact SDs [7], [14] do not scale efficiently with the size of the employed constellation. In particular, most efficient implementations adopt the Schnorr-Euchner (SE) [12] tree-traversal strategy, according to which, when expanding a parent node, the children are visited in ascending order of their (partial) Euclidean metric. This strategy has the ability to substantially decrease the number of nodes that need to be visited until the ML solution is found. However, the computational complexity required to perform this sorting can determine the efficiency of the SD, especially for very dense QAM constellations. In single-dimensional (e.g., PAM) or constant envelope (e.g., PSK) constellations, where only the phase or the amplitude of the signal changes, one-dimensional ''zig-zag'' enumerations can be used to avoid the exhaustive Euclidean metric calculations and the corresponding sorting of the expanded nodes [7]. However, such one-dimensional approaches are not directly applicable to two-dimensional constellations like QAM. To do that, the complex SD tree search can be translated into a real tree search [28], [29]. However, since the height of the new, real-valued tree is double the height of the original complex-valued tree search, this results in a substantially increased number of visited nodes, and therefore, in a substantially increased processing latency [7].

In order to reduce the SE enumeration's implementation cost and increase circuit throughput without translating the complex SD problem into a real one, the ASIC-II implementation in [7] realized an alternative non-exhaustive SE scheme, which subdivided the two-dimensional QAM constellation plane into one-dimensional concentric circles, and performed partial enumeration and sorting across the concentric cycles, reducing area for both Euclidean metric calculations and node enumeration. Focusing on optimizing the throughput of [7], the work in [30] introduced pipeline interleaving and early termination. In [31], a new enumeration scheme subdivided the constellation into vertical pulse-amplitude modulated (PAM) subsets allowing the authors to present results in 64-QAM. To the best of the authors' knowledge, this is the most efficient enumeration for dense and very dense constellation, that can provide the exact SE sorting required by exact SDs. As such, it has been implemented in the benchmark architecture of this work. Geometry-based enumeration solutions such as [10], [18], and [32] aim to greatly reduce storage and computation requirements but can only sort a small subset of the constellation's nodes. Conversely, to preserve the exact optimality of the algorithm [33], [34] propose a predefined visiting order in combination with a new pruning criterion that preserves SD optimality. However, the memory requirements of these approaches make them unsuitable for very dense constellations. Therefore, what prior work shows, and the problem this work aims to address, is that optimal detection is practical only in less dense constellations.

This work presents the design and evaluation, both in terms of software-defined radio and VLSI architecture, of *Geosphere*; a depth-first SD that can provide the exact ML solution and can efficiently scale to very dense constellations like 1024-QAM. Geosphere is based on a geometrically inspired two-dimensional (2D) ''zigzag'' enumeration scheme, that can be directly applied to QAM constellations without requiring decomposing the complex-valued tree-search into a real-valued one, and perform exact node sorting while avoiding unnecessary Euclidean metric calculations. To that end, Geosphere's implementation should adhere to the two-dimensional process (*i.e.*, dual node storage and sibling detection), without excessive overhead compared to that of one-dimensional enumeration. Therefore, its implementation needs to maintain a) similar hardware logic latency, b) slightly increased yet not doubled storage requirements c) the one-node-per-cycle property of the current state-of-the-art. Based on this two-dimensional enumeration, we propose a new node labeling approach that enables the efficient mapping of our 2D zigzag method on hardware architectures. As a basis for our implementation work, we choose the best known and, to the best of our knowledge, most efficient VLSI architecture [7], able to deliver the exact ML solution. However, for benchmarking purposes, we have replaced the authors' proposed enumeration with the PAM-based enumeration of [31], since the original enumeration, evaluated for 16-QAM modulation in [7], is not efficient for very dense

constellations (as a very large number of sub-constellations is required as we also show in Section IV). Hereafter, we will refer to this new implementation as ''PAM-based-ETH SD''. Our results show that Geosphere's VLSI implementation is substantially more efficient than the PAM-based-ETH. For example, Geosphere is twice as area-efficient and 70% more energy efficient while displaying a 13% higher area efficiency than the PSK-based SD for 16-QAM in [16]. These results are achieved by a parameterizable design relying on Geosphere's enumeration scheme, by a low-complexity integer arithmetic as well as by fine-grained clock gating. To the best of our knowledge this is the first time where resource cost, delay, power consumption and scalability are all simultaneously explored and documented in the framework of sphere detection. In general, the contributions of this work can be summarized as follows:

- The actual gains of exact sphere decoding, and therefore Geosphere, against linear detection approaches, have been evaluated for actual channel traces collected by means of a software-defined implementation (using the Rice WARP v3 radio hardware) for an indoor scenario, showing spectral efficiency gains of more than two times for 4 × 4 256-QAM transmissions at 30 dB SNR and, in contrast to linear detection approaches, nearly-linear spectral efficiency increase with the number of users.
- A geometrically inspired âĂIJzigzagâĂİ enumeration and a new node labeling approach have been proposed that enable the implementation of exact sphere decoders that efficiently scale to very dense constellation symbols.
- According to the best of our knowledge, this is the first time that an exact ML hard-output SD for 1024-QAM has been implemented and evaluated in the literature.
- Two exact depth-first SDs (i.e., Geosphere and the PAM-based-ETH) have been implemented in VLSI, with their designs allowing scalability at arbitrarily dense square QAM constellations.
- It is shown for the first time (instead of simply remarking) that traditional SD architectures do not scale well with very dense constellations and to that end, the first time the traditional architectures have been implemented for such dense constellations.
- This is the first time in the open literature that resource cost, delay, power consumption and scalability are simultaneously explored and documented in the framework of sphere detection.

This work focuses on depth-first solutions for several reasons. First, in contrast to breadth-first approaches [9], [17], [35]–[37] depth-first SDs can guarantee exact ML performance. Furthermore, and as we also show in Section VII, while breadth-first SDs allow efficient parallelization and pipelining which can lead to a high processing throughput, this typically comes at a very high area and power cost. Still, approximate versions can be attained via early termination, probabilistic pruning [15], [38], [39], or metric-based approaches ([7]-ASIC-II, [31], [40]).

The rest of the paper is structured as follows: Section II begins with a primer on sphere decoding, setting up our subsequent discussion of Geosphere's design in Section III. In Section III-B we propose a node enumeration procedure in the complex domain, which is based on attaching labels to the constellation point nodes and will enable Geosphere's implementation. We then lay the groundwork for its implementation by describing the general structure of non-exhaustive SE enumeration architectures (Section IV). Section V describes the proposed VLSI SD implementation details. Algorithmic evaluation follows in Section VI, where we evaluate Geosphere using actual indoor gathered channel traces. In Section VII we evaluate the VLSI architectures' functionality and assess their power consumption by employing both simulated and actual channel traces. Finally, Section VIII concludes the manuscript.

II. PRIMER: SPHERE DECODER

This section provides essential background on the sphere decoder which achieves ML detection, i.e., it determines the most likely transmitted vector $\mathbf{x}^*$ chosen from a constellation $\mathcal{O}$ of size $|\mathcal{O}| = 2^Q$ (*i.e.*, Q bits per symbol):

$$\mathbf{x}^* = \arg \min_{\mathbf{s} \in \mathcal{O}^{n_c}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2, \tag{1}$$

where $\mathbf{y}$ is the received signal vector, $\mathbf{H}$ the channel matrix and $\mathbf{s}$ the transmitted signal vector. Solving Eq. 1 exhaustively would entail $|\mathcal{O}|^{n_c}$ Euclidean distance $d(\mathbf{s})$ calculations. The SD reduces this complexity by transforming the ML problem into a search in a tree of height n_c (number of transmit antennas) and branching factor $|\mathcal{O}|$ (constellation size) as shown in [5] and [6].

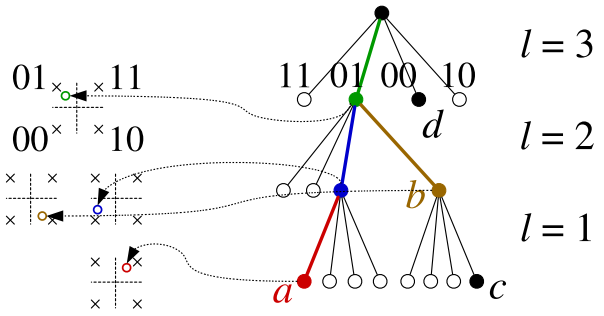


FIGURE 1. The sphere decoder tree search for $n_c = 3$ transmit antennas, each sending a QPSK-modulated symbol ($|\mathcal{O}| = 4$). The branches of the tree are numbered at the topmost level ($l = 3$). Constellation points are denoted with × and the received signal with ○. Coloring is used to depict visited nodes.

Figure 1 shows an example for $n_c = 3$ and QPSK ($|\mathcal{O}| = 4$). Each level l of the tree corresponds to a decision on the value s_l of the transmitted symbols from antennas l through n_c, i.e., the *partial symbol vector* $\mathbf{s}^{(l)} = [s_l, s_{l+1}, \ldots, s_{n_c}]$. To realize this tree search the channel matrix first has to be QR-decomposed as $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}^*\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = [r_{ij}]$ is *upper triangular* ($i, j \in [1, n_c]$). Each *partial symbol vector* (i.e., branch) $\mathbf{s}^{(l)}$ in the

tree is associated with a non-negative *branch cost* $c(\mathbf{s}^{(l)}) = \left|\widehat{y}_l - \sum_{j=l}^{n_c} r_{lj}s_j\right|^2$. We then calculate the partial Euclidean distance for $\mathbf{s}^{(l)}$ as: $d(\mathbf{s}^{(l)}) = d(\mathbf{s}^{(l+1)}) + c(\mathbf{s}^{(l)})$ where $\mathbf{s}^{(l+1)}$ is the tentative solution constructed up to the level above. Due to the QR-decomposition the Euclidean distances $d(\mathbf{s})$ become $d(\mathbf{s}) = \|\widehat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$, where $\widehat{\mathbf{y}} = \mathbf{R}\mathbf{s} + \mathbf{Q}^*\mathbf{w}$, and $\widehat{\mathbf{y}} = \mathbf{Q}^*\mathbf{y}$ is the transformed received signal. Therefore, the detection problem is transformed into the minimization of $\|\widehat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$. Since the *branch cost* is non-negative, the sphere decoder can *prune* all children below partial symbol $\mathbf{s}^{(l)}$ if they violate the sphere constraint (i.e., $d(\mathbf{s}^{(l)}) \geq r^2$).

Traversing the tree: Continuing our example of Figure 1, a conventional Schnorr-Euchner SD with radius update will initially have its radius set to infinity, then exhaustively determine the path to a leaf a that minimizes partial Euclidean distance at each level (the path is highlighted using thick lines in the figure). This entails computing distances for all children as well as all sibling nodes along the path (all nodes in this diagram). Upon reaching a, the decoder sets its sphere radius to $d(a)$ and backtracks up one level to check node b whose distance is second-closest. In the case of $d(b) < d(a)$, the sphere decoder needs to expand b, enumerate its children, and find the one with minimum distance (c). Once this is finished, the decoder backtracks up one level again to $l = 3$ and considers node d. Now $d(d) \geq d(a)$, so none of d's children or siblings (note that the nodes are sorted) could possibly be the maximum-likelihood solution, so the sphere decoder terminates and returns a as the ML solution.

Even though this pruning reduces the number of visited nodes compared to a naive exhaustive search, it can be computationally expensive. In particular, the sorting requirement of Schnorr-Euchner enumeration for higher-order constellations (*e.g.*, 16- and 64-QAM), can compromise the sphere decoder's efficiency, preventing its employment in very dense constellations.

III. GEOSPHERE: DESIGN AND NODE LABELING ENUMERATION

This section first presents the design of Geosphere's enumeration technique which we use in order to avoid exhaustively sorting the children of a node in the sphere decoder and then proposes a new node labeling approach which will aid in Geosphere's VLSI architecture implementation. In Section VI, we experimentally evaluate the relative gains under varying channel conditions.

The goal of Geosphere's enumeration technique is to determine the order in which the sphere decoder should explore the set of constellation points $\mathcal{O}$, when it is considering the possible children of a particular parent node in the tree shown in Fig. 1. We wish to explore constellation points in order of increasing branch cost, but the only soft information at our disposal is the received symbol.

However, since constellation distance is related to partial Euclidean distance by

$$c\left(\mathbf{s}^{(l)}\right) = |r_{ll}|^2 \, |\widetilde{y}_l - s_l|^2 \tag{2}$$

(where $\widetilde{y}_l = \frac{\widehat{y}_l - \sum_{j=l+1}^{n_c} r_{lj}s_j}{r_{ll}}$), it suffices to explore the constellation points in increasing Euclidean distance from the received symbol in the constellation itself, rather than as measured indirectly by the partial Euclidean distance metric.

If we were sending constellation points in one dimension (this is known as *pulse-amplitude modulation*, or *PAM*), then to find the closest constellation point to the received symbol we would first need to follow a procedure called *slicing*. Slicing compares the received symbol against decision boundaries residing on the midpoint between consecutive constellation points. Subsequent points would be determined by the *zigzag* rule, based on which we visit the next closest, unvisited constellation point from the initial closest point.

A. TWO-DIMENSIONAL ZIGZAG ENUMERATION

In the case of two-dimensional zigzag, we are in fact seeking an approximation of an expanding ring search, starting at an arbitrary, continuous-valued received symbol point ◦. One inexact way of accomplishing this would be to partition the QAM constellation into PAM subconstellations, and then zigzag ''vertically'' within each subconstellation [31]. But this approach neglects the *in-phase* component of the received symbol.

Geosphere instead first slices the received symbol to find the closest constellation point (denoted as a), and begins the two-dimensional zigzag from that exact constellation point to determine a's next sibling. Node a's Euclidean distance is calculated and stored along with its partial vector in a priority queue which is constantly sorted by the Euclidean distances of its contents. Note that the sphere decoder will then expand the branch corresponding to a and search that subtree. Once the Geosphere SD returns to the node whose child nodes it is sorting, this node is removed from the queue and the sphere decoder zigzags both horizontally and vertically, since it is searching for the next-closest sibling in (two-dimensional) Euclidean distance. This entails calculating the Euclidean distances of the nodes encountered on the horizontal and vertical directions, and adding these nodes to the queue. We avoid adding the node encountered through the horizontal zigzag to the queue if a constellation point from the target PAM subconstellation is already in our list of outstanding constellation points to explore. This ensures that we have at most one candidate constellation point per (vertical) PAM subconstellation. A description of Geosphere's algorithm can be found in [11].

Notice that as a consequence of the two-dimensional zigzag rule, the algorithm requires a priority queue of length at most $\sqrt{|\mathcal{O}|}$. By only taking zigzag steps one constellation point at a time, the algorithm defers the Euclidean distance computation until the point in time it is required, often by which time the sphere decoder has pruned the relevant subtree (we demonstrate this later in the experimental evaluation).

B. PROPOSED NODE LABELING ENUMERATION

To facilitate the design of a modular and scalable VLSI architecture which can be practical in very dense constellations

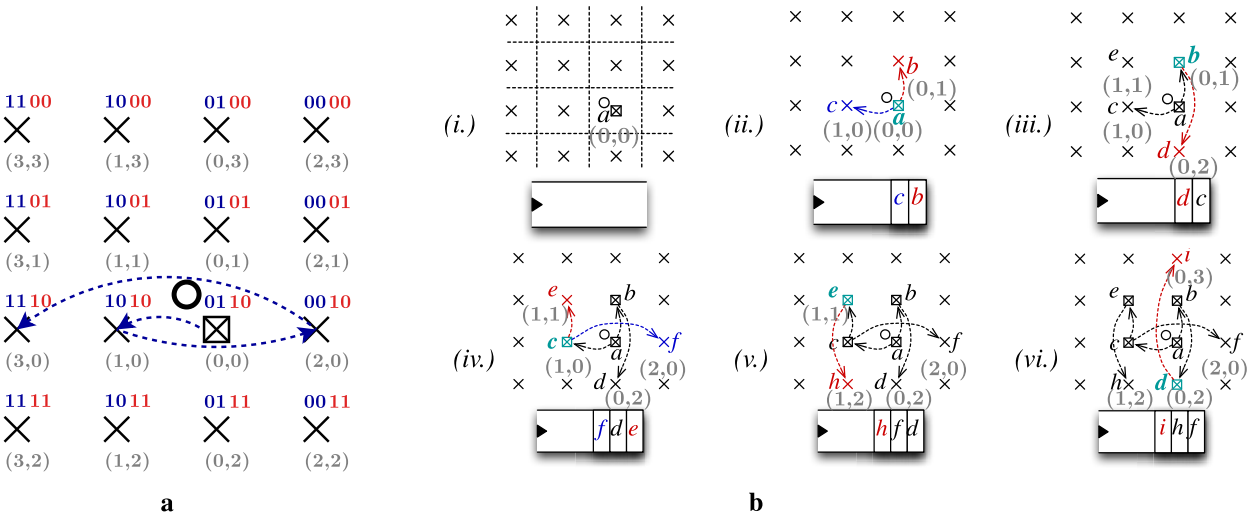


FIGURE 2. The proposed binary-mapped constellation scheme (a) and the attached node labels for the received symbol ∘ and the closest constellation point ⊠. Real indices are highlighted in dark blue and imaginary indices in red colour. Part (b) depicts Geosphere's two-dimensional zigzag enumeration in the 16-QAM constellation employing the proposed node labeling technique. We denote constellation points with ×, label points whose partial Euclidean distances have been computed, and denote points that have been explored with ⊠. Dark blue color denotes horizontal zigzags while red color denotes vertical zigzags.

with a reasonable area/power cost, we propose a new node representation approach which we apply to both Geosphere and the PAM-based-ETH SD. This approach is based on (*i*) labeling child nodes corresponding to a specific parent at a tree level and (*ii*) employing binary-mapped constellation point indices.

Node labeling assists in determining the sibling $\hat{\mathbf{s}}^{(l)}$ which will replace a child node and therefore the visiting order of child nodes. Each node's labels describe the number of zigzag movements required for the node to be reached from its currently visited sibling. We employ binary-mapped constellation points as depicted in Fig. 2a, whereby indexing begins from the top right constellation point and increases towards the left and bottom directions. Notice that the labels for all siblings of a particular received constellation point are constant, as they also define a specific zigzag visiting order. Then, each child node with a partial vector $\mathbf{s}^{(l)}$ can be described by its horizontal and vertical labels as $(hl(\mathbf{s}^{(l)}), vl(\mathbf{s}^{(l)}))$, where $hl, vl \in [0, \sqrt{|\mathcal{O}|}-1]$. By definition, the first child nodes selected are the first to be possibly visited (unless they violate the sphere constraint) and have thus always the labels $(hl, vl) = (0, 0)$ attached to them. Replacement nodes $\hat{\mathbf{s}}^{(l)}$ as determined through Geosphere's zigzag enumeration in the horizontal and vertical direction have the labels $(hl(\hat{\mathbf{s}}_h^{(l)}), vl(\hat{\mathbf{s}}_h^{(l)})) = (hl(\mathbf{s}^{(l)}) + 1, vl(\mathbf{s}^{(l)}))$ and $(hl(\hat{\mathbf{s}}_v^{(l)}), vl(\hat{\mathbf{s}}_v^{(l)})) = (hl(\mathbf{s}^{(l)}), vl(\mathbf{s}^{(l)}) + 1)$ respectively attached. As a consequence, horizontal labels are zero for every node in the PAM-based-ETH SD.

As an example to the aforementioned enumeration technique, Fig. 2b depicts Geosphere's 2D zig-zag enumeration with the proposed labels attached to the nodes. Node a is the first chosen child node and therefore $(hl(a), vl(a)) = (0, 0)$. Node c is its horizontal replacement, hence $(hl(c), vl(c)) = (1, 0)$ and, similarly for node b, $(hl(b), vl(b)) = (0, 1)$. In step (*iii.*) a 2D zig-zag from node b would result in its replacement from nodes d and e. While $(hl(d), vl(d)) = (0, 2)$, notice that node e would normally require the labels $(hl(e), vl(e)) = (1, 1)$. Visiting e though would have violated Geosphere's same-vertical-subconstellation rule (as the queue already contains node c which has the same vertical label as e). Continuing from node c and hence removing it from the queue, the respective labels of its siblings f and e become $(hl(f), vl(f) = (2, 0)$ and $(hl(e), vl(e)) = (1, 1)$. Notice how this combined labeling/mapping describe the number of zigzags. For instance, node i in step (*vi.*) whose labels are $(hl(i), vl(i)) = (0, 3)$ requires three leaps in the vertical direction in order to be reached from its sibling d.

This scheme allows us to determine the next sibling by adding or subtracting the label's value as an offset which is normally $\lceil \log_2 \sqrt{|\mathcal{O}|} \rceil$ bits wide, instead of storing all labels beforehand (Fig. 2a). In the cases where enumeration has reached one of the constellation's margins, the next sibling's offset can also be set to one depending on the enumeration's direction.

IV. NON-EXHAUSTIVE SE ENUMERATION ARCHITECTURES

We first describe the general features of non-exhaustive enumeration architectures as depicted in Fig. 3. We outline their structure and the modules required to traverse the tree without compromising the ML solution. This is a generalization of the architecture originally introduced in [7]-ASIC-II in order to facilitate the design description in the following section.

A. METRIC COMPUTATION (MCU)

The Metric Computation Unit detects the nodes to be possibly visited on the current tree level l and computes their PDs,

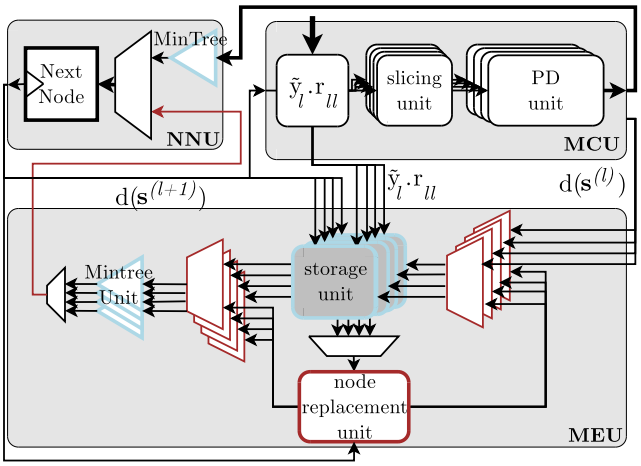


FIGURE 3. Architectural overview of sphere decoders with non-exhaustive enumeration.

based on the nodes selected at the levels above. We will refer to this process as the *forward movement* of the detector. Initially, the *MCU* computes the received symbol $\widetilde{\mathbf{y}}_l = \frac{\widehat{\mathbf{y}}_l - \sum_{j=l+1}^{n_c} r_{lj} s_j}{r_{ll}}$, common to all possible child nodes. All SD architectures execute this step once per *forward movement*. The implementation cost of the *MCU* mainly depends on n_c, the employed norm and the bit width. Normally, exhaustive SE enumeration would require computation and minimum search among the PDs of every possible child given a parent node (i.e., at most $|\mathcal{O}|$ PD calculation units). The non-exhaustive *MCU* instead contains slicing units which compare the received symbol to appropriate decision boundaries thereby mapping the symbol to a specific constellation point considered as the closest. The exact number and nature of the PD operations depend on detection requirements.

B. METRIC ENUMERATION UNIT (MEU)

The Metric Enumeration Unit operates in parallel with the *MCU* in order to determine the next parent node. The *MEU* forwards the next parent to the *MCU* in case of sphere constraint violation, or leaf occurence. In exhaustive enumeration architectures, this would entail the *MEU* a) storing the PDs of all possible children except for the node already chosen by the *MCU* and, consequently, b) searching for the minimum PD among these remaining nodes. In non-exhaustive enumeration architectures, information about a smaller set of nodes needs to be stored which reduces both area and delay requirements. The *MEU* should then determine (i.e., *node replacement unit*) the next sibling node $\hat{\mathbf{s}}^{(l)}$ which will replace the visited node in a subsequent *forward movement* based on a set of rules. These rules dictate both the optimality of the detection process, and the scalability and performance of the non-exhaustive architecture.

C. NEXT NODE UNIT (NNU)

In both exhaustive and non-exhaustive depth-first SDs, the *NNU* resides between the *MCU* and the *MEU* and its role is to select the parent $\mathbf{s}^{(l+1)}$ of the child node to be visited next in the traversal process. Selection is made between the nodes provided by the *MCU* and the *MEU* based on sphere constraint violation and leaf node validity. Both the *MCU* and the *MEU* search among at most $|\mathcal{O}|$ nodes to obtain the one with the minimum PD. This is usually carried out through a comparator network arranged in a binary tree fashion (*MinTree* in Fig. 3). The *MEU* will also employ the *NNU*'s contents to compute $\hat{\mathbf{s}}^{(l)}$.

D. LEAF STORAGE AND CONTROL UNITS

At the lowermost tree level, if a leaf or a dead-end is reached then the SD directly visits the corresponding node or selects a new node from the *MEU*. Therefore at the bottom level only the partial vector and the PD of the leaf need to be stored. Finally, a control unit guides the whole detection process controlling the dataflow inside the *MCU*, the input to the *NNU*, the writing and replacement logic in the *MEU* and finally, the detection's start and termination.

V. SCALABLE DEPTH-FIRST SDs: PAM-BASED ETH AND GEOSPHERE VLSI ARCHITECTURES

In this section, we employ the hierarchy outlined in Sect. IV to facilitate our description of the proposed VLSI architectures. We exploit the use of integer arithmetic throughout all computations and utilize the node labeling enumeration of Sect. III-B in order to present a modular approach that achieves scalability to very dense constellations. Aiming at exact detection with low power consumption, we jointly explore the energy and delay of the multiplication units and follow a low-complexity storage unit implementation approach that uses fine-grained clock gating.

Due to this work's goal in assessing the cost in very dense constellations, instead of subdividing the plane into concentric circles, we employ vertical PAM subsets due to their better scalability [31]. For instance, in the cases where $|\mathcal{O}| \in \{16, 64, 256, 1024\}$ the respective subset cardinality becomes $|\widetilde{\mathcal{O}}|_{circ} \in \{3, 9, 32, 109\}$ in the case of concentric circles and $|\widetilde{\mathcal{O}}|_{PAM} \in \{4, 8, 16, 32\}$ in the case of PAM subconstellations.

A. MCU: DESIGN AND IMPLEMENTATION

The *MCU*'s main computational burden lies in calculating $c\left(\mathbf{s}^{(l)}\right)$ as in Eq. (2). Even though the diagonal elements r_{ll} are real-valued, the division operation in $\widetilde{\mathbf{y}}_l = \frac{\widehat{\mathbf{y}}_l - \sum_{j=l+1}^{n_c} r_{lj} s_j}{r_{ll}}$ would on one hand significantly increase the latency of the *MCU* even when replaced by a fixed-point arithmetic reciprocal operation. On the other hand, the design of throughput optimized architectures is beyond the scope of this work and has already been addressed in the literature [18], [30], [41]. Introducing pipeline interleaving in order to decrease the induced latency would compromise the joint low area/power/scalability scope of this work due to the presence of the feedback loop. Instead, we shorten the critical path by computing $\widehat{\mathbf{y}}_l - \sum_{j=l+1}^{n_c} r_{lj} s_j - r_{ll} s_l$ (i.e., multiply all constellation point values by r_{ll}).

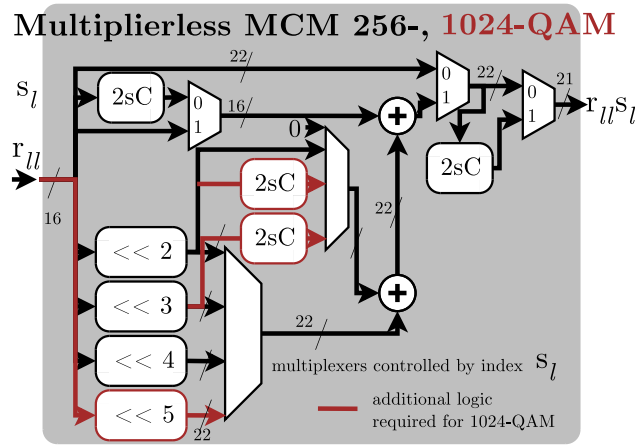


FIGURE 4. Multiplierless Multiple Constant Multiplication (MCM) architectures for integer constellation point multiplication: 256- and 1024-QAM.

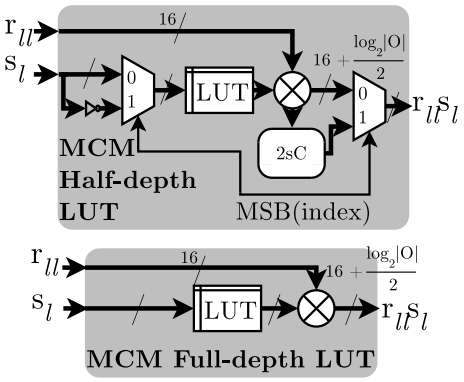


FIGURE 5. Look-Up Table-based (LUT-based) constant multipliers: Half-Depth LUT (MCM-HD, top), Full-Depth LUT (MCM-FD, bottom).

TABLE 1. Synthesis comparison for three distinct multiple constant multiplier (MCM) implementations employing 16-bit numbers and $\frac{\log_2(\sqrt{|\mathcal{O}|})}{2}+1$ binary indices as constellation point constant coefficient input: Multiplierless MCM case (M), Full-Depth LUT MCM (FD), Half-Depth LUT MCM (HD).

		Delay (ns)			Area (GE)		
implementation		**M**	**FD**	**HD**	**M**	**FD**	**HD**
modulation	16-QAM	0.40	0.32	0.32	1018	769	784
	64-QAM	0.42	0.35	0.34	1149	824	844
	256-QAM	0.51	0.40	0.38	1332	984	1157
	1024-QAM	0.54	0.41	0.38	1615	1165	1185
	avg	**0.467**	**0.370**	**0.355**	**1279**	**936**	**993**
		Power (mW)			EDP (mW·ns^2)		
implementation		**M**	**FD**	**HD**	**M**	**FD**	**HD**
modulation	16-QAM	0.4902	0.5620	0.5595	0.0784	0.0575	0.0572
	64-QAM	0.5077	0.5282	0.5509	0.0895	0.0647	0.0637
	256-QAM	0.4413	0.5048	0.5748	0.1147	0.0807	0.0830
	1024-QAM	0.4707	0.5507	0.6090	0.1372	0.0925	0.0879
	avg	**0.477**	**0.536**	**0.573**	**0.1049**	**0.0738**	**0.0729**

1) LOW-COMPLEXITY MULTIPLE CONSTANT MULTIPLIERS (MCMs)

As all constellation point values can be considered as coefficients of integer nature, the design of low-complexity constant coefficient multipliers is critical to maintain a low overall area cost. A flexible multiplierless approach was proposed in [32] for modulation schemes up to 64-QAM. In this manuscript, we modify the work in [32] to employ the proposed binary mapped constellation indices (Section III-B) instead of the actual constellation point values and additionally extend multiplication to 256 and 1024-QAM constellations. As we focus on exact ML approaches, we avoid the precision assumptions made in [32] through which the authors replace two's complement units by simple negation. The architecture of the multiplierless implementations (M-MCM) is depicted in Fig. 4. We employ the $\ll$ symbol to denote left arithmetic shifts and "2sC" to denote a two's complementing operation. We also consider two additional multiplier-based implementations: a) integer constellation points are stored in a $(\log_2(\sqrt{|\mathcal{O}|}) \times \log_2(\sqrt{|\mathcal{O}|}) + 1)$-bit lookup table (full-depth, MCM-FD, Fig. 5 bottom), and b) where only the positive integer values are stored and the most significant binary index bit (MSB(index)) is employed to negate both the constellation index and the product $((\frac{\log_2(\sqrt{|\mathcal{O}|})}{2} \times \log_2(\sqrt{|\mathcal{O}|}) + 1)$-bit half-depth lookup table, denoted as MCM-HD in Fig. 5 top). To enhance accuracy, we increase internal post-multiplication precision by keeping $\frac{\log_2(\sqrt{|\mathcal{O}|})}{2}$ additional bits. The fractional and total word length/precision are parameterizable.

To determine the most energy efficient solution, we evaluate the Register Transfer Level (RTL) synthesis results[1] of each multiplier implementation for $\mathcal{O} \in [16, 64, 256, 1024]$ considering a 16-bit r_{ll} input. Registers were employed at the input and output paths of each multiplier in order to obtain the minimum delay. Based on the synthesis results of Table 1 both MCM-HD and MCM-FD perform better than the M-MCM in terms of minimum delay while they require less area and consume less power. Consequently, they both achieve better area efficiency as estimated by their Area-Delay Products (ADP) at 351.43 and 356.95 GE·ns against 610.3 GE·ns of the M-MCM solution. Since our goal is the joint optimization of energy, resource cost and performance, we employ the Energy-Delay Product (EDP) metric which takes into account both the energy consumption and the critical path. Results slightly favor the MCM-HD solution in almost all cases and thus it is the solution to be subsequently employed as the proposed multiplication units are more suitable for an ASIC implementation targeting maximum precision. We note here that while the 1024-QAM constant multipliers also calculate all products for less dense constellations, implementing a flexible detector for multiple standards is outside the scope of this manuscript which instead aims to explore the scaling behavior.

[1]Synthesis was performed using Synopsys Design Compiler on 45nm TSMC library using typical case characteristics. Area is measured in Gate Equivalents (GEs) where one GE is the area of a two input NAND gate synthesized using the employed libraries.

2) SLICING UNITS

Following the computation of $\tilde{\mathbf{y}} \cdot r_{ll} = \widehat{\mathbf{y}}_l - \sum_{j=l+1}^{n_c} r_{lj} s_j$, each detector determines the closest constellation point(s) to the received symbol through its slicing units.

a: PAM-Based Architecture: During the *forward movement* on a level of the tree, the closest point in each PAM subset is found and, subsequently, $\sqrt{|\mathcal{O}|}$ PD values are computed among which the minimum is chosen as the node to be visited. Regardless of the node chosen within each subset, nodes among subsets share the same imaginary part. Additionally, all points within a subset share the same real part. Therefore, apart from fewer subsets, the subdivision into PAM subconstellations results in a more simplified design versus the *PSK ALU* of [7] in that only a single slicer unit is required to detect the imaginary part of $\tilde{\mathbf{y}} \cdot r_{ll}$. To employ the aforementioned low-complexity multipliers and allow favorable scaling of the proposed architecture in dense and very dense constellations, slicing is performed on integer intermediate decision boundaries scaled by r_{ll}. Since these boundaries are located in the midpoints between two consecutive constellation symbols, only arithmetic shifts and two's complement operations are necessary to scale the value of r_{ll}. The received symbol is compared with the scaled result and then directly mapped to the binary mapped constellation point index. Therefore in this specific architecture the real indices of the $\sqrt{|\mathcal{O}|}$ constellation points form a binary arithmetic sequence $\in [0, \sqrt{|\mathcal{O}|}]$, and can thus be pre-stored to be directly input to the PD calculators.

b: Geosphere Architecture: Owing to the much lower complexity of Geosphere's tree traversal, its *MCU* mainly consists of a two-dimensional slicing unit and a single PD calculator. The former comprises of two identical one-dimensional slicers which only differ at their first input, i.e., the real and imaginary parts of $\tilde{\mathbf{y}} \cdot r_{ll}$. Moreover, only a single sphere constraint comparator is required and hence there is no need for a *MinTree* unit in the *NNU*.

3) PD CALCULATORS

The proposed PD calculators follow the design depicted in Fig. 6. They consist of two MCM units which compute the $r_{ll} \cdot s_l$ products. The result is then subtracted from $\tilde{\mathbf{y}} \cdot r_{ll}$ and input to a generic l^2 norm unit. The implemented PD calculator also contains a partial vector generator module (Fig. 6) which calculates $\mathbf{s}^{(l)}$.

B. MEU

Whenever the *MCU* is unable to proceed further on its own, the *MEU* needs to determine the next sibling node and thus required to store the current state of the search for each tree layer and the attributes of each tree node. These attributes normally consist of the node's partial vector, its PD, and a single bit flag which verifies validity. Non-exhaustive enumeration schemes require additional attributes to fully describe each node and guide the enumeration process. In the

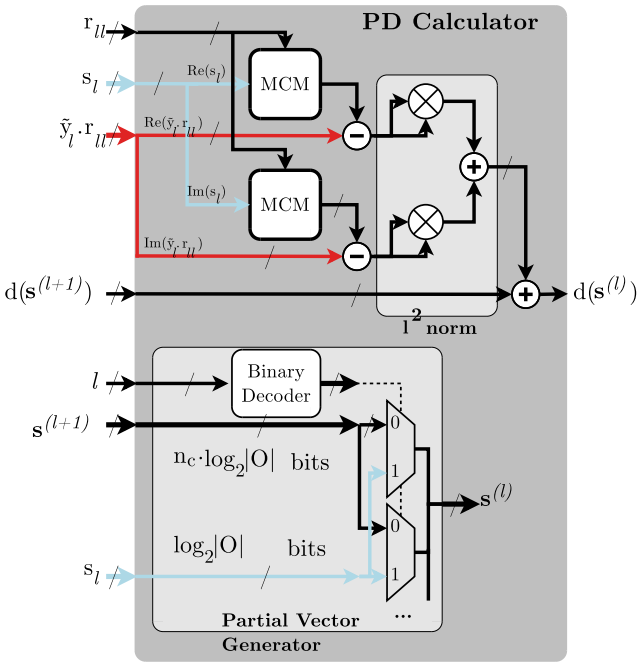


FIGURE 6. The proposed partial distance calculator/partial vector generator employing MCMs and a generic l^2 norm unit.

case of the PAM-based architecture, the proposed enumeration technique requires for each node to additionally store: a) its vertical label *vl* and b) two flags which define whether the top and bottom constellation margins (*tm* and *bm* respectively) were encountered during enumeration. Geosphere's nodes require storage of their horizontal label (*hl*), as well as two additional flags for the left and right constellation margins (*lm* and *rm* respectively). Note that to replace the node which has been selected and is currently being visited and in order to save area/power, the current state of the tree-search process also involves storage of the $d(\mathbf{s}^{(l+1)})$ and $\tilde{\mathbf{y}} \cdot r_{ll}$, both of which have already been computed by the *MCU*. In our implementations, we organize storage units as register banks.

1) STORAGE UNIT CONTENTS (PAM-BASED ETH)

Each unit requires storage of $\sqrt{|\mathcal{O}|}$ elements of: a) $\log_2 \sqrt{|\mathcal{O}|}$ bits for the partial vectors, b) parameterized width for the PDs, c) single valid bits, d) single bits for the top margin $tm(\mathbf{s}^{(l)})$, e) single bits for the bottom margin $bm(\mathbf{s}^{(l)})$, f) $\frac{\log_2 |\mathcal{O}|}{2}$ bits for the vertical label $vl(\mathbf{s}^{(l)})$.

2) STORAGE UNIT CONTENTS (GEOSPHERE)

Geosphere's storage units have exactly the same depth as in the PAM-based architecture [11]. Additionally, these units require $\sqrt{|\mathcal{O}|}$: a) single-bit registers for the left margin $lm(\mathbf{s}^{(l)})$, b) single-bit registers for the right margin $rm(\mathbf{s}^{(l)})$, c) $\frac{\log_2(|\mathcal{O}|)}{2}$-bit registers for the horizontal label $hl(\mathbf{s}^{(l)})$.

3) STORAGE UNIT ORGANIZATION

Before deciding on the organization of each register bank, we have to take into account the requirements of each specific algorithm as well as our design goals which involve low energy consumption, competitive performance and tractable scaling behavior without compromising the exact ML solution or the one node per-cycle behavior.

Both schemes require a PD-sorted priority queue due to the SE enumeration rule. In the literature, an efficient insertion-sorted queue with a relatively good scaling behavior was first presented in [42]. Ref [43] conduct a comparison among several priority queue implementations where the aforementioned shift register queue proved to be the most prevalent in single cycle enqueue/dequeue cases of small to medium storage capacity. Both of these characteristics apply to our case as storage capacity scales with $\sqrt{|\mathcal{O}|}$ and single-cycle operations are required. On the other hand, both SD architectures require parallel writing to multiple locations: the PAM-based during *forward movement*, Geosphere during its backtracking. In order to achieve insertion-based sorting, all registers in the shift register queue [42] activate during insertion. The latter is not ideal for the low energy goal of this work, particularly since all stored node attributes have to be shifted even though only the PDs would be employed as sorting keys.

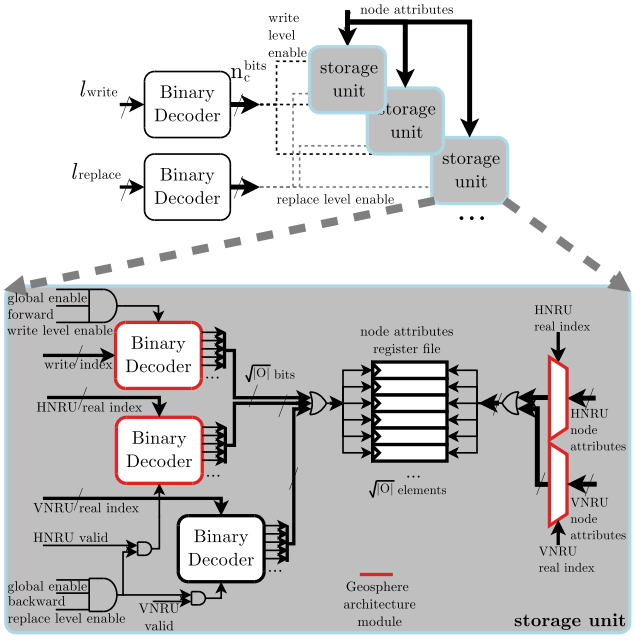


FIGURE 7. Architecture of the storage units, their clock-enabling and their node attribute storage logic.

Our proposed solution, depicted in Fig. 7, combines instead a parallel load/store register file approach followed by a *MinTree* unit per storage unit, which ensures selection of the node with the minimum PD (Fig. 3). Note that a single *MinTree* unit preceded by a $\sqrt{|\mathcal{O}|}\times$ larger multiplexer could possibly decrease area requirements at the expense of significantly increased latency. To reduce switching activity and therefore power consumption, both of the proposed implementations employ fine-grained clock gating on all registers inside the SD (Fig. 7). While on one hand this can increase the fanout of signals and the control logic required to implement the detectors' functionality, on the other hand it also allows for greater control over the detector as a whole entity (e.g., for stalling) or specific parts of it when required.

4) NODE STORAGE (PAM-BASED)

In the *forward movement* phase, the PAM-based *MCU* outputs $\sqrt{|\mathcal{O}|}$ nodes along with their corresponding attributes. It forwards these to all buffers and the SD control unit ensures through a level write (l_{write}) signal that only a single storage unit will be active for storing the node. Within each storage unit, all registers are active since all nodes need to be stored at this point in time. Therefore, the *write index* binary decoder in Fig. 7 is unnecessary.

5) NODE STORAGE (GEOSPHERE)

During Geosphere's *forward movement*, its *MCU* outputs only a single node to be stored into one active storage unit. Additionally, by decoding the detected node's real index, a *write index* signal activates a single position in the register file to store the node's attributes, thus saving power.

6) NODE REPLACEMENT (PAM-BASED)

In the PAM-based ETH SD, only a single sibling $\hat{\mathbf{s}}^{(l)}$ needs to be computed and therefore only a single location is updated in the storage unit. The *Vertical Node Replacement Unit* (*VNRU*) computes the node residing in the same vertical subconstellation as the currently visited node $\mathbf{s}^{(l)}$ and which will replace the latter in the next clock cycle at the same register file location. The *VNRU* consists of the *Replacement Computation Unit* (*RCU*) which computes the attributes of $\hat{\mathbf{s}}^{(l)}$ and forwards these attributes to the *Vertical Computer* (*VC*: essentially a partial distance calculator). Figure 8 depicts the architecture of the *VNRU*. Implementing the proposed enumeration technique, the *RCU* consists of a multiple constant multiplier (MCM) which multiplies the constellation value by r_{ll}. The scaled result is compared against the $\tilde{\mathbf{y}}$ r_{ll} and, combined with the sign of r_{ll} and the values of the $tm(s_l)$ and $bm(s_l)$ signals, defines the corresponding offset to be added directly to the $s^{(l)}$ index. Notice that the replacement node's label $vl(\hat{\mathbf{s}}_l)$ is always produced by incrementing $vl(s_l)$ by one. The *RCU* also outputs the *visited all nodes* signal indicating that the currently selected and visited node is the last in the PAM subconstellation. In this case the SD does not store the *VNRU*'s result, invalidates the corresponding buffer entry and bypasses the *VC*. Following the computation of $\hat{s}_l$, the *VC* can now calculate $d(\hat{\mathbf{s}}^{(l)})$ based on the depicted PD calculator.

7) NODE REPLACEMENT (GEOSPHERE)

During Geosphere's replacement phase, the current node stored in the *NNU* needs to be replaced by up to two siblings. These have to be both available in the immediate clock

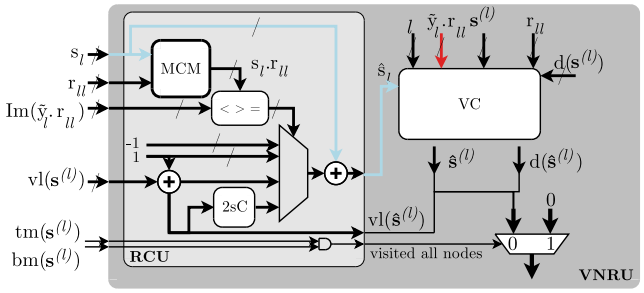


FIGURE 8. Architecture of the Vertical Node Replacement Unit (VNRU): realizing node labeling enumeration, partial distance calculation and partial vector generation.

cycle and stored in the buffer for subsequent enumerations. Therefore we have to employ an additional node replacement unit, to which we will hereafter refer as *Horizontal Node Replacement Unit* (*HNRU*). Due to the proposed enumerator's design the *HNRU* is almost identical to the *VNRU* with the exception that now, the *HNRU* has to generate an extra signal indicating the presence of a stored node in the buffer's position corresponding to the same PAM subconstellation. To that end, the *HNRU* requires input from the corresponding buffer which will store the replacement. Aiming at minimal additional logic which scales well with the constellation size, our proposed solution is depicted in Figure 9: first, a binary decoder decodes the real index of the replacement node $\hat{\mathbf{s}}^{(l)}$ to a $\sqrt{|\mathcal{O}|}$-bit output. Then, each of the decoder's output bits are employed as a mask which is applied at the valid bit array originating from the storage buffer. The masked outcome is finally compressed to a single bit using an OR operation to produce the *same subconstellation* signal. Through this signal, the SD bypasses the PD calculation.

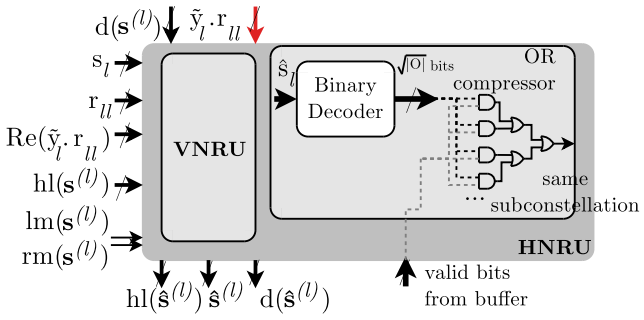


FIGURE 9. Architecture of the Horizontal Node Replacement Unit (HNRU): retaining basic VNRU functionality and additionally realizing Geosphere's same subconstellation rule.

Notice that now two sets of node attributes can be possibly generated, one originating from the *VNRU* and one from the *HNRU*. In this case, we want to ascertain that both nodes can be written to the storage unit's register file without conflict or overwriting the results already stored therein. Conflict-free access is established by the fact that the real indices of the nodes generated by the *HNRU* and the *VNRU* are by definition different. As these indices will enable the corresponding register file locations, the node attribute vectors generated by each *NRU* are de-multiplexed into a specific location of the $\sqrt{|\mathcal{O}|}$-element register file and the two de-multiplexed vectors are then merged into one by an OR operation (Fig. 7). The merged vector is subsequently employed for storage as in the PAM-based case. Finally, Geosphere's same subconstellation rule establishes that no attributes will overwrite those already stored: if the *HNRU* result's index is the same as that of a currently stored node, then the same *subconstellation signal* disables decoding of the index and thus nothing is written to the storage.

VI. GEOSPHERE'S ALGORITHMIC EVALUATION

In this section we measure Geosphere's performance gains and computational complexity requirements in real indoor office conditions. We gather channel traces by employing Rice WARP v3 radio hardware and perform trace-based simulations via OFDM modulation and demodulation using 4-, 16-, 64- and 256-QAM constellations. All clients send data using 1/2-rate convolutional coding (similar to recent 802.11 standards). We compare Geosphere with zero-forcing and MMSE systems that attempt to intelligently adapt to poorly-conditioned MIMO channels by varying the number of antennas and spatial streams they use. Finally, we evaluate Geosphere's computational complexity, comparing it with the well established PAM-based-ETH depth-first sphere decoder which, as we disussed in Section I, is the combination of of the architecture in [7] with the PAM-based enumeration of [31].

Our testbed consists of single-antenna clients and four-antenna APs, communicating over a 20 MHz wireless channel in the 5 GHz ISM band. The distance between consecutive AP antennas is about 20 cm (approximately 3.2λ, where λ is the wireless wavelength) so that the wireless channels from each AP antenna to a client are uncorrelated with each other, and thus representative of antenna spacings above λ indoors at 5 GHz [44]. We first evaluate Geosphere in an indoor environment, measuring the MIMO channels which correspond to several concurrently transmitted streams across all subcarriers and for many different client and access point (AP) positions. Our goal is to assess how well-conditioned are indoor channels, since, in that case, zero-forcing can be very efficient in demultiplexing the interfering streams. For a more detailed description of our testbed environment and our channel characterization methods, please refer to [11].

A. SYSTEM PERFORMANCE

In this section, we compare the uplink performance of ZF and MMSE serving a network of clients, against Geosphere. We consider four SNR ranges, 15 dB ±5 dB, 20 dB ± 5 dB, 25 dB ±5 dB and 30 dB ±5 dB, where the quoted SNR is the average SNR over all transmitted streams. Selecting users in a small SNR range around a specific value is a practical user selection method to keep the condition number small. Larger gains are expected for Geosphere if the users are selected randomly. In addition, in lieu of implementing a

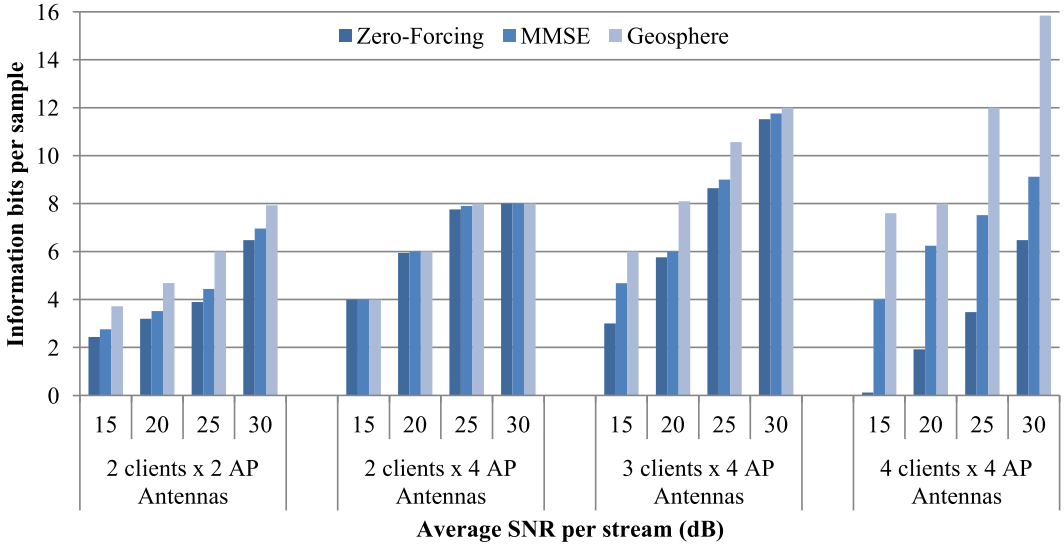


FIGURE 10. Trace-based simulations performance comparison for zero-forcing, MMSE MIMO and Geosphere: spectral efficiency for a varying number of clients, AP antennas and SNRs.

rate adaptation algorithm, we show spectral efficiency results for the constellation that achieves the best average number of effective bits per sample for the corresponding range, taking into account the average achieved packet error rate and the code rate; this emulates ideal bit rate adaptation and makes the results independent of the rate adaptation method employed.

As can be seen in Fig. 10 regarding the effective average number of information bits per sample for different numbers of clients and receive antennas, Geosphere consistently outperforms both zero-forcing and MMSE. Moreover, as expected, Geosphere's performance gains increase with the condition number and Λ. In particular, for the 2×2 case using 256-QAM modulation at 30 dB SNR, Geosphere's performance can be 14% higher than MMSE and 22% higher than ZF. In the case of 4×4 transmissions, Geosphere's performance is $1.7\times$ higher than that of MMSE and up to $2.4\times$ higher than that of ZF. Even in the most challenging case of two or three clients and an AP with four receive antennas (where channels are most often well-conditioned), Geosphere can achieve increased spectral efficiency against ZF and MMSE. Please note that the error-rate performance of the PAM-based-ETH decoder and therefore, its spectral efficiency, is identical to that of Geosphere and it is thus omitted from Fig. 10. Since the condition number of a matrix becomes smaller with decreasing numbers of concurrently transmitting clients, another question we may ask is whether zero-forcing or MMSE combined with an appropriate scheduling strategy could match Geosphere's performance, with fewer clients per transmission. Our simulation results in Fig. 10 reveal that Geosphere with four clients and four receive antennas achieves higher performance than both the zero-forcing and the MMSE schemes for three transmitting clients. In particular, Geosphere's spectral efficiency is 34.7% higher than that of MMSE and can be up to 37.5% higher than that of ZF at 30 dB SNR using 256-QAM modulation.

B. COMPUTATIONAL COMPLEXITY

We now quantify the computational requirements of Geosphere. To this end, we compare Geosphere against the PAM-based-ETH SD. One frequently-used measure of computational complexity in the literature is the number of visited nodes in the sphere decoder tree. However, we also require a metric that captures Geosphere's additional computation *i.e.*, the one which avoids visiting nodes. Since the dominant part of the additional computation is partial Euclidean distance calculations, this metric tracks overall complexity accurately, and so we primarily use this metric in our evaluation, as is also common in the literature [45]. For completeness and additional insight into why Geosphere improves performance, we also report the number of visited nodes. Since in an OFDM system, MIMO processing takes place on a per subcarrier basis, we report the preceding metrics as needed per subcarrier, averaged across all subcarriers.

Since the WARP platform's analog front end limits it to a maximum SNR of approximately 30 dB over the links in our testbed, which is not adequate for transmitting 256 or 1024 QAM constellations, for the following computational complexity experiments we perform simulations. We present both ***(a)*** *trace-based* simulations, driven by empirical MIMO channel measurements collected from our WARP testbed, and ***(b)*** simulation over a MIMO *Rayleigh* fading channel with independent, identically-distributed channel realizations sampled on a per-frame basis.

In Fig. 11 we show complexity for an SNR such that each constellation reaches a packet error rate of approximately 1% (*e.g.*, approximately 12, 18, 24 and 31 dB for the 2×4 measured channels and 16-, 64-, 256- and 1024-QAM constellations, respectively). We examine two MIMO cases: In the rightmost part of Fig. 11 we show complexity for two clients and four AP antennas. In this case, complexity is relatively low, due to favorable MIMO channel conditioning,

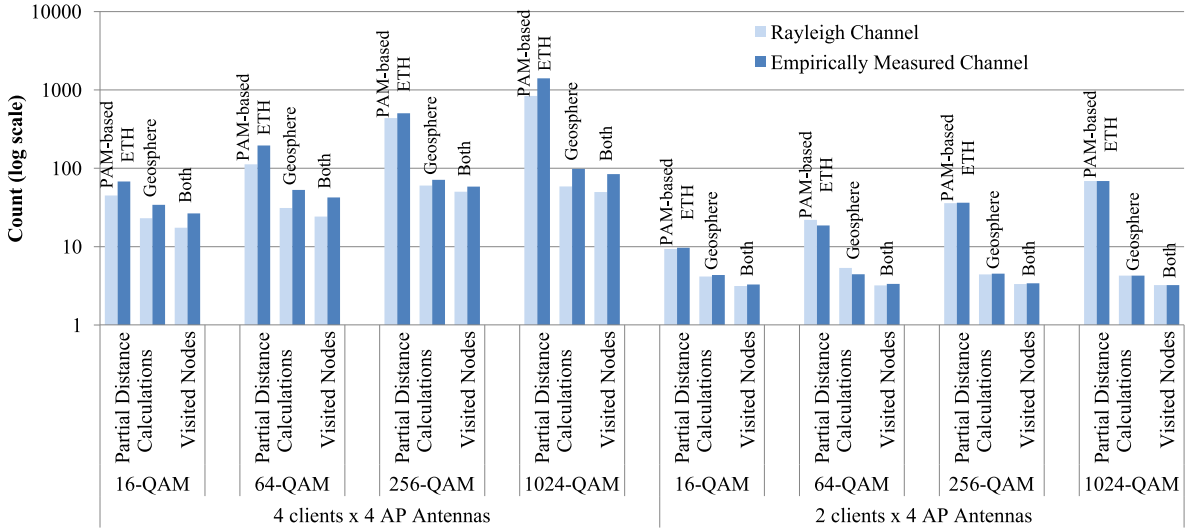


FIGURE 11. Simulation-based complexity comparison: PAM-based-ETH vs Geosphere SDs in 4 × 4 and 2 × 4 MIMO (PER=10^{-2}).

but at the cost of reduced throughput, since only two users transmit. We note that the complexity of PAM-based-ETH SD increases with constellation size, while the complexity of Geosphere is substantially smaller, independent of the constellation size, and comparable to the complexity of zero-forcing.[2] For the Rayleigh channel, Geosphere is 93% less complex than the PAM-based-ETH SD for the 1024-QAM case. We note that Geosphere's complexity remains almost constant among constellation sizes as evaluation is performed at different signal-to-noise ratios, (*i.e.*, to maintain a packet error rate of 10^{-2}).

The leftmost part of Fig. 11 shows the complexity for four clients and four AP antennas, where we need to cope with more challenging MIMO channel conditions. For the 4×4 case we see that the complexity of PAM-based-ETH SD (but not Geosphere) greatly increases with constellation size. As a result Geosphere is up to 93% less complex than the PAM-based-ETH SD for the Rayleigh channel. In addition we see that the zigzag algorithm is the main source of complexity improvement for large constellations.

As noted above, the throughput gains of Geosphere are modest for well-conditioned channels (e.g., for two users and four AP antennas). One might therefore be tempted to argue in favor of a system that switches back to zero-forcing when faced with a well-conditioned wireless channel. However, the above results show that Geosphere actually adjusts its computational complexity to the current SNR, and so complexity at high SNR is actually very small, obviating the need for a hybrid system. In the 2×4 case, Geosphere requires 87 up to 95% fewer partial distance calculations compared to the 4×4 case. We finally note that this work focuses on addressing the problem of dense and very dense constellations and to assess the corresponding performance aspects. In a different perspective, performance can also be enhanced by increasing the number of antennas. This is though an entirely different problem which is beyond the scope of this work.

[2]Zero-forcing requires $n_t \times n_r = 8$ complex multiplications, whereas Geosphere requires at most 10 complex multiplications (assuming that each partial distance calculation requires $n_t + 1$ multiplications).

VII. VLSI ARCHITECTURE EVALUATION

To jointly evaluate the resource cost, performance, energy and scalability of the proposed SD architectures in very dense constellations we instantiate the designs in $\mathcal{O} \in \{16, 64, 256, 1024\}$-QAM modulation schemes for $n_c = 4$. Our aim to compare Geosphere with the PAM-based implementation results, *i.e.* the best known hard output depth-first sphere detector which attains ML performance and whose enumeration scheme allows good scalability in dense constellations.

The proposed designs were implemented using parametric Verilog RTL code. The code was designed to allow the instantiation of all sub-modules for arbitrary modulation schemes with square constellations. We note here that only the MCMs and the slicing units have to be redesigned among the different modulation schemes. The RTL code was synthesized using the Synopsys Design Compiler and TSMC 45nm standard cell libraries under typical operating conditions (25°C and 0.9V). The RTL and post-synthesis netlists were compared with the MATLAB model to check exact ML detection performance. For purposes of comparison, we implement all detectors employing a word width of 24-bits for $d(\mathbf{s}^{(l)})$ and 16-bits for $\mathbf{R}$ to assess a worst-case resource cost. Note that word width further increases internally in order to enhance arithmetic precision (Fig. 4).

A. AREA-DELAY-ENERGY-THROUGHPUT-SCALABILITY

We first measure the area requirements, the maximum achievable frequency, the consumed energy and the attained

throughput of both *Geosphere* and the PAM-based-ETH SDs. Our goal is to illustrate *Geosphere*'s advantages when both designs are implemented using the same design principles.

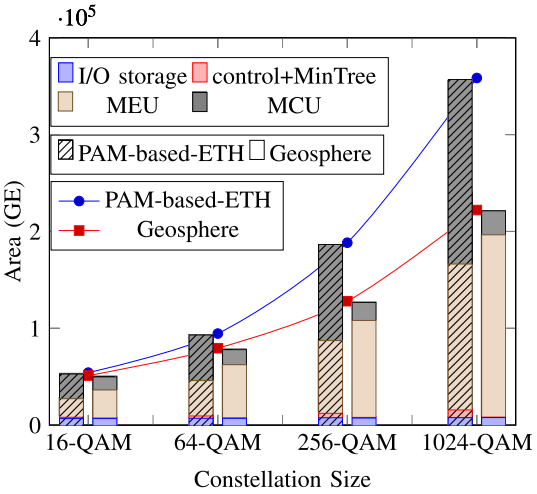


FIGURE 12. Sphere decoder architectures' resource cost and scalability: area (GE) breakdown at the maximum achievable frequency (f_{max}).

Post-synthesis area results in GE are displayed in Fig. 12 for both architecture implementations. Results show that Geosphere displays a significantly decreased cost compared to the PAM-based architecture. Between consecutive constellation sizes, Geosphere's area increases by an average factor of 1.63× while the PAM-based architecture approximately doubles its cost. This behavior can be attributed to the much larger area cost of the *MCU*, as between consecutive constellation sizes its cost in the PAM-based architecture increases by an average factor of 1.96×. Geosphere's *MCU* on the other hand increases its corresponding cost by 1.22× on average, which is attributed to the larger internal word width and the increased number of the slicer comparators. As expected, the largest contributor to Geosphere's resource cost is its *MEU*. While between consecutive constellation sizes the cost factor is on average 1.9× for both architectures, the 2D zigzag, the requirement of a second replacement node and their imposed complexity, incur a 33.7% additional cost to Geosphere's *MEU* for 16-QAM modulation. As $|\mathcal{O}|$ increases, the *MEU*'s resource cost is dominated by the registers required to store the PDs and the partial vectors. Hence, Geosphere's extra replacement unit and the extra storage unit complexity have much less impact: in the 1024-QAM case, Geosphere's *MEU* is 19.9% larger than the one in the PAM-based architecture. To obtain a more comprehensive view of the actual resource cost, we calculate the Area-Delay Product (ADP) for each architecture. The maximum achievable frequency f_{max} and thus the minimum delay per architecture and constellation size for the specified bit widths is displayed in Table 2. Results illustrate that on average Geosphere achieves a 11.8% higher frequency than the proposed PAM-based implementation. Notice also that Geosphere exhibits the same f_{max} as the PAM-based

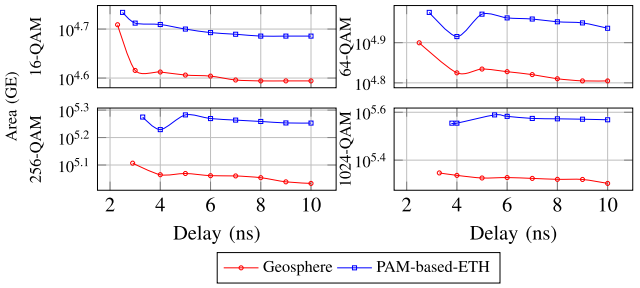


FIGURE 13. SDs joint area/delay/modulation assessment.

TABLE 2. Post-synthesis area, delay and scalability SD evaluation: PAM-based-ETH vs Geosphere.

4×4 SDs @ 24-bit $d(\mathbf{s}^{(l)})$, 16-bit **R**				
Modulation Scheme	**16-QAM**	**64-QAM**	**256-QAM**	**1024-QAM**
f_{max} **(MHz)**				
PAM-based-ETH	400	345	303	263
Geosphere	435	400	345	303
Area (GE)				
PAM-based-ETH	54240	94542	188382	358458
Geosphere	51187	79437	127955	222090
Area-Delay Product (GE·ns)				
PAM-based-ETH	135600	274172	621660	1362140
Geosphere	117730	198592	371069	732897

architecture, albeit at the immediately higher constellation density. Moreover, combining the delay results with Geosphere's much lower resource cost results in an ADP that is 14% better than that of the PAM-based architecture in 16-QAM while in very dense constellations, Geosphere is almost twice as area efficient. Figure 13 presents a joint area-delay-constellation size graphical assessment for both architectures based on synthesis results. Notice that alternate routing paths and buffer insertion tend to increase the area to meet timing requirements, hence the distance between the curves recedes at the left side of the plots and the results reported in Table 2. When timing requirements are less strict, the plots show that the actual resource ratio between the two solutions starts at 0.8× for 16-QAM Geosphere and can reach 0.54× in 1024-QAM.

To estimate the power consumed by each synthesized design, we employ the same test vectors used in the MATLAB simulations (for both the Rayleigh and the Empirically Measured Channels at PER$\in \{10^{-1}, 10^{-2}\}$). Using these vectors, the gate level netlist was simulated to generate the corresponding switching activity files for both detectors at $\mathcal{O} \in \{16, 64, 256, 1024\}$. Power consumption was estimated using Synopsys Power Compiler and the Energy-Delay-Product (EDP) figure of merit was calculated to assess the implementations' energy efficiency.

Total power results are aggregated in Tables 3 and 4 in the Rayleigh- and trace-based cases respectively. Results show that Geosphere consistently outperforms the PAM-based architecture. Even at a PER of 10^{-1} where 28–46% more nodes are being visited, the proposed design ensures

TABLE 3. Post-synthesis energy, throughput and scalability Rayleigh-based SD evaluation: PAM-based-ETH vs Geosphere.

4×4 SDs @ 24-bit $d(\mathbf{s}^{(l)})$, 16-bit $\mathbf{R}$				
Modulation Scheme	**16-QAM**	**64-QAM**	**256-QAM**	**1024-QAM**
Rayleigh Channel Evaluation				
	$\mathscr{D}_{avg}$			
SNR (dB) @ PER=10^{-1}	12	19	26	32
	25.23	29.61	71.16	94.76
SNR (dB) @ PER=10^{-2}	15	22	28	35
	17.50	24.22	50.47	49.90
Power @ PER=10^{-1} (mW)				
PAM-based-ETH	11.3486	17.9848	30.6633	53.1954
Geosphere	9.5609	12.3900	15.3246	20.5234
Power @ PER=10^{-2} (mW)				
PAM-based-ETH	11.2177	17.9233	30.6567	52.9070
Geosphere	9.3671	12.2315	15.2762	20.6007
EDP @ PER=10^{-1} (mW·ns^2)				
PAM-based-ETH	70.9287	151.2521	333.9233	768.1415
Geosphere	50.5771	77.4375	128.8798	223.4998
EDP @ PER=10^{-2} (mW·ns^2)				
PAM-based-ETH	70.1106	150.7349	333.8514	763.9771
Geosphere	49.5519	76.4468	128.4728	224.3416
Throughput @ PER=10^{-1} (Mbps)				
PAM-based-ETH	253.666	279.635	136.256	111.017
Geosphere	275.862	324.214	155.143	127.902
Throughput @ PER=10^{-2} (Mbps)				
PAM-based-ETH	365.714	341.866	192.114	210.821
Geosphere	397.714	396.367	218.744	242.886

TABLE 4. Post-synthesis energy, throughput and scalability Trace-based SD evaluation: PAM-based-ETH vs Geosphere.

4×4 SDs @ 24-bit $d(\mathbf{s}^{(l)})$, 16-bit $\mathbf{R}$				
Modulation Scheme	**16-QAM**	**64-QAM**	**256-QAM**	**1024-QAM**
Trace-based Channel Evaluation				
	$\mathscr{D}_{avg}$			
SNR (dB) @ PER=10^{-1}	14	21	28	34
	33.29	57.43	81.12	139.42
SNR (dB) @ PER=10^{-2}	16	23	30	37
	26.65	42.54	58.53	84.31
Power @ PER=10^{-1} (mW)				
PAM-based-ETH	11.3440	17.8158	30.1985	51.4689
Geosphere	9.5638	12.1718	15.0050	19.6881
Power @ PER=10^{-2} (mW)				
PAM-based-ETH	11.2067	17.8158	29.9579	51.2213
Geosphere	9.3973	12.1718	14.8876	19.6232
EDP @ PER=10^{-1} (mW·ns^2)				
PAM-based-ETH	70.9000	149.8308	328.8616	743.2109
Geosphere	50.5925	76.9287	126.1920	214.4034
EDP @ PER=10^{-2} (mW·ns^2)				
PAM-based-ETH	70.0418	148.8031	326.2415	739.6355
Geosphere	49.7117	76.0737	125.2047	213.6966
Throughput @ PER=10^{-1} (Mbps)				
PAM-based-ETH	192.249	144.103	119.538	75.501
Geosphere	208.967	167.160	136.027	86.940
Throughput @ PER=10^{-2} (Mbps)				
PAM-based-ETH	240.150	194.543	165.675	124.852
Geosphere	261.033	225.669	188.526	143.769

that circuit activity is being held at a minimum during backtracking and replacement. In particular, the PAM-based architecture consumes over 16% more power in the 16-QAM case, 31% in the 64-QAM case while in very dense constellations, Geosphere reduces consumed power by more than 60% (*e.g.*, in 1024-QAM). Furthermore, taking into account the higher achieved frequency Geosphere shows an improved energy efficiency of 29% in 16-QAM that can reach 61% in 256-QAM and surpass 70% in 1024-QAM.

We also present the throughput evaluation of both SDs based on the average number of visited nodes $\mathbf{D}_{avg}$ for each modulation scheme and SNR (Tables 3 and 4). Based on the results, Geosphere displays a 8.7% higher throughput in 16-QAM, that can reach 16% in denser constellations.

B. COMPARISON AGAINST OTHER WORKS IN THE LITERATURE

In this section, we compare the proposed detectors with other hard-output ASIC implementations in the literature. Results are displayed in Table 5, scaled to 45nm at an operating voltage $V_{DD} = 0.9$ V. Both the PAM-based and the Geosphere architectures have been re-synthesized using a 16-bit word width (a common performance/complexity tradeoff in the literature [40], [41]). Note that the presented results involve modulation schemes up to 64-QAM which up until now has been the upper margin for hard-output depth-first SD implementations. Comparison is conducted against both similar, exact-ML detectors as well as against approximate depth-first and breadth-first detectors.

Against exact-ML detectors, Geosphere achieves higher area efficiency. In particular, against the best-known hard-output exact ML detector in [16] Geosphere achieves a 13% higher area efficiency and a 15% average higher throughput. Also note that compared to the PSK-based l^2 norm architecture in [16] the proposed PAM-based implementation achieves a similar area efficiency which in denser constellations is expected to improve due to the latter's better scaling behavior (Section V). As expected, the low complexity design of the proposed implementations results in a 67 to 70% lower area cost than that of the exhaustive SE architecture (PAM-based and Geosphere respectively against the ASIC-I in [7]). In fact, even the proposed 64-QAM Geosphere implementations are 56 to 67% more area efficient compared to the 16-QAM exhaustive scheme. EDP results illustrate a similar behavior, as Geosphere displays a 77% higher energy efficiency than the exhaustive scheme (65% for the PAM-based architecture).

To emphasize on the advantages of Geosphere and its proposed realization, we also compare against implementations which sacrifice the exact ML solution for the sake of complexity reduction. Against the PSK-based SD in [7] utilizing the $l^{\widetilde{\infty}}$ norm, both Geosphere and the PAM-based implementation respectively display a 41 and 34% higher area efficiency. Note that compared with the throughput and energy optimized l^1 norm-based solution in [40], the proposed VLSI architectures are 56 to 61% more area efficient even though they achieve approximately 50% lower throughput. Moreover, Geosphere's implementation attains a slightly higher energy efficiency, despite the l^1 norm's lower

TABLE 5. Area, energy and performance comparison of the PAM-based-ETH and Geosphere SD implementations employing a 16-bit datapath against the state-of-the-art.

Detector:	**PAM-based-ETH**[a]		**Geosphere**[a]		**[7]**[b]	**[16]**[a]	**[7]**[b]	**[40]**[b]	**[46]**[b]	**[41]**[b]
					ASIC-I		ASIC-II			
$n_c \times n_{AP}$	4×4									
$\lvert\mathcal{O}\rvert$	16	64	16	64	16				64	
Type	Folded DF							Pipelined DF	BF (K-Best)	
Performance	ML						Approximate			
Norm	l^2						l^∞	l^1	l^2	
Technology (nm)[c]	45				250 @ 45			180 @ 45	130 @ 45	
Area (GE)	37689	65395	35028	54377	117000	34400	50000	175000	491000	340000
f_{max}[d]**(MHz)**	454	357	476	400	278	406	394	920	397	1205
ADP (GE·ns)	82915	183106	73558	135942	420030	84830	126750	190207	1236829	282200
Power[e]**(mW)**	9.2	12.6	6.2	8.4	9.9	-	-	25.4	24.8	331
EDP (mW·ns^2)	44.5	98.8	29.1	52.5	128.1	-	-	30.0	157.1	228.0
Mbps[f] **@ PER=10^{-1}**	288/218	289/149	302/229	324/167	176/134	257/195	250/189	583/442	3177	2880
Mbps @ PER=10^{-2}	415/273	354/201	435/286	393/226	254/167	371/243	360/236	841/552		

[a] Post-Synthesis results. [b] Silicon Implementation. [c] We employ the notation "$\mathscr{T}$ @ 45" to signify the original lithographic technology results scaled at 45nm. [d] Frequency scaling by multiplying f_{max} at $\mathscr{T}$ nm with $\frac{\mathscr{T}(nm)}{45}$. [e] Power scaling by multiplying power at $\mathscr{T}$ nm with $(\frac{0.9}{V_{DD}})^2 \cdot \frac{45}{\mathscr{T}(nm)}$. [f] Throughput of Depth-First implementations computed as $\frac{n_c \cdot \log_2 \lvert\mathcal{O}\rvert}{\mathscr{D}_{avg} \cdot t_{min}}$, where $t_{min} = \frac{1}{f_{max}}$ is the minimum delay in seconds, $\mathscr{D}_{avg}$ results from Tables III and IV.

complexity. Finally, breadth-first K-best solutions focus on throughput optimization based on a highly pipelined design. Apart from sacrificing detection optimality, these solutions require a much steeper area and power consumption premium. Against throughput-oriented efficient solutions which up until recently were restricted to the real domain (such as [46]), Geosphere is 89% more area efficient and exhibits $3\times$ lower EDP. Compared to the K-best breadth-first SD in the complex domain, [41] displays an approximately threefold increase in resources and tenfold increase in power consumption compared with the authors' real-domain implementation. Compared against [41] representing the state-of-the-art in K-best complex-domain SD, Geosphere offers exact-ML detection at twice the area efficiency and $4\times$ better energy efficiency, as respectively assessed through the ADP and EDP figures of merit (Table 5).

VIII. CONCLUSIONS

We have described Geosphere, a wireless multi-user MIMO system that, by employing sphere decoding-based detection, and in contrast to systems using linear detection methods, can achieve a linear spectral efficiency increase when increasing the number of scheduled users up to the number of access point/base station antennas. Geosphere makes the sphere decoder practical in real high-rate wireless systems using dense and very dense constellations. Its geometrically inspired enumeration combined with node labeling allowed the design and implementation of a scalable VLSI architecture that can provide detection optimality at significantly lower implementation cost compared to the best known and best scalable solution. Geosphere's depth-first exact nature, will allow the direct application of its principles to soft receiver processing in order to further approach MIMO capacity. Geosphere can be directly used with soft-output (non-iterative) receivers that perform a common tree search and with "list sphere-decoders" approximately calculating the soft information in iterative, soft-input, soft-output systems. Future work will involve extension in an iterative framework by employing already proposed strategies that extend "hard" decision to soft input, soft output detectors [45], [47], [48].

Acknowledgment

The authors would like to thank C. Jayawardena, J. Zhou and B. Congdon for their assistance.

REFERENCES

[1] I. Telatar, "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecommun.*, vol. 10, no. 6, pp. 585–596, Dec. 1999.

[2] C.-J. Chen and L.-C. Wang, "On the performance of the zero-forcing receiver operating in the multiuser MIMO system with reduced noise enhancement effect," in *Proc. IEEE Globecom*, 2005, pp. 1294–1298.

[3] W. L. Shen, K. C.-J. Lin, S. Gollakota, and M.-S. Chen, "Rate adaptation for 802.11 multiuser MIMO networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 1, pp. 35–47, Jan. 2014.

[4] H. Artes, D. Seethaler, and F. Hlawatsch, "Efficient detection algorithms for MIMO channels: A geometrical approach to approximate ML detection," *IEEE Trans. Signal Process.*, vol. 51, no. 11, pp. 2808–2820, Nov. 2003.

[5] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

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

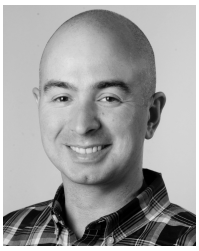
[7] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[8] A. M. Chan and I. Lee, "A new reduced-complexity sphere decoder for multiple antenna systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2002, pp. 460–464.

[9] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.
[10] B. Mennenga and G. Fettweis, "Search sequence determination for tree search based detection algorithms," in *Proc. IEEE Sarnoff Symp.*, Apr. 2009, pp. 1–6.
[11] K. Nikitopoulos, J. Zhou, B. Congdon, and K. Jamieson, "Geosphere: Consistently turning MIMO capacity into throughput," in *Proc. SIGCOMM*, 2014, pp. 631–642.
[12] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, no. 1, pp. 181–191, 1994.
[13] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2131–2142, Jul. 2008.
[14] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.
[15] B. Shim and I. Kang, "On further reduction of complexity in tree pruning based sphere search," *IEEE Trans. Commun.*, vol. 58, no. 2, pp. 417–422, Feb. 2010.
[16] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.
[17] M. Shabany, K. Su, and P. G. Gulak, "A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder," in *Proc. IEEE ICASSP*, Apr. 2008, pp. 3173–3176.
[18] M. Wenk, L. Bruderer, A. Burg, and C. Studer, "Area- and throughput-optimized VLSI architecture of sphere decoding," in *Proc. IEEE/IFIP VLSI-SOC*, Sep. 2010, pp. 189–194.
[19] M.-Y. Huang and P.-Y. Tsai, "Toward multi-gigabit wireless: Design of high-throughput MIMO detectors with hardware-efficient architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 2, pp. 613–624, Feb. 2014.
[20] M. M. Mansour, S. P. Alex, and L. M. A. Jalloul, "Reduced complexity soft-output MIMO sphere detectors—Part II: Architectural optimizations," *IEEE Trans. Signal Process.*, vol. 62, no. 21, pp. 5521–5535, Nov. 2014.
[21] H. Teague *et al.*, "Field results on MIMO performance in UMB systems," in *Proc. IEEE Veh. Technol. Conf.*, May 2008, pp. 1009–1015.
[22] N. Kita, W. Yamada, A. Sato, D. Mori, and S. Uwano, "Measurement of Demel condition number for 2×2 MIMO-OFDM broadband channels," in *Proc. IEEE VTC*, May 2004, pp. 294–298.
[23] "MIMO performance and condition number in LTE test," Agilent Technol., Santa Clara, CA, USA, Appl. Note, 2009.
[24] F. Kaltenberger, M. Kountouris, D. Gesbert, and R. Knopp, "On the trade-off between feedback and capacity in measured MU-MIMO channels," *IEEE Trans. Wireless Commun.*, vol. 8, no. 9, pp. 4866–4875, Sep. 2009.
[25] H. Suzuki, T. V. A. Tran, I. B. Collings, G. Daniels, and M. Hedley, "Transmitter noise effect on the performance of a MIMO-OFDM hardware implementation achieving improved coverage," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 6, pp. 867–876, Aug. 2008.
[26] C. Taylor. (Nov. 2015). *802.11ac Wave 2 With MU-MIMO: The Next Mainstream Wi-Fi Standard. Strategy Analytics*. [Online]. Available: https://www.qualcomm.com/media/documents/files/strategy-analytics-802-11ac-wave-2-with-mu-mimo-the-next-mainstream-wi-fi-standard.pdf
[27] J. Colwell, "5G Summit," in *Proc. 5G North Amer. Workshop*, Nov. 2014. [Online]. Available http://files-eu.clickdimensions.com/ericssoncom-anlg4/documents/5gsummit-colwellfinal.pdf
[28] A. R. Murugan, H. El Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: Rediscovering the sequential decoder," in *Proc. IEEE 6th Workshop Signal Process. Adv. Wireless Commun.*, Jun. 2005, pp. 761–765.
[29] E. Zimmermann, *Complexity Aspects in Near Capacity MIMO Detection Decoding*. Jörg Vogt Verlag, 2007.
[30] A. Burg, M. Wenk, and W. Fichtner, "VLSI implementation of pipelined sphere decoding with early termination," in *Proc. IEEE 14th Eur. Signal Process. Conf.*, Sep. 2006, pp. 1–5.
[31] C. Hess *et al.*, "Reduced-complexity MIMO detector with close-to ML error rate performance," in *Proc. 17th ACM Great Lakes Symp. VLSI*, 2007, pp. 200–203.
[32] C.-H. Yang and D. Markovic, "A flexible DSP architecture for MIMO sphere decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 10, pp. 2301–2314, Oct. 2009.
[33] K. Nikitopoulos, "Maximum likelihood detection of spatially multiplexed signals via loosely ordered depth-first sphere decoding," *Electron. Lett.*, vol. 48, no. 21, pp. 1368–1370, Oct. 2012.
[34] K. Nikitopoulos, A. Karachalios, and D. Reisis, "Exact max-log MAP soft-output sphere decoding via approximate Schnorr–Euchner enumeration," *IEEE Trans. Veh. Technol.*, vol. 64, no. 6, pp. 2749–2753, Jun. 2015.
[35] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, p. 1154.
[36] M. Shabany and P. G. Gulak, "Scalable VLSI architecture for K-best lattice decoders," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 940–943.
[37] S. Mondal, A. Eltawil, C.-A. Shen, and K. N. Salama, "Design and implementation of a sort-free K-best sphere decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 10, pp. 1497–1501, Oct. 2010.
[38] R. Gowaikar and B. Hassibi, "Statistical pruning for near-maximum likelihood decoding," *IEEE Trans. Signal Process.*, vol. 55, no. 6, pp. 2661–2675, Jun. 2007.
[39] T. Cui, S. Han, and C. Tellambura, "Probability-distribution-based node pruning for sphere decoding," *IEEE Trans. Veh. Tech.*, vol. 62, no. 4, pp. 1586–1596, May 2013.
[40] R. Jenkal and R. Davis, "An architecture for energy efficient sphere decoding," in *Proc. Int. Symp. Low Power Electron. Design*, 2007, pp. 244–249.
[41] M. Mahdavi and M. Shabany, "Novel MIMO detection algorithm for high-order constellations in the complex domain," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 5, pp. 834–847, May 2013.
[42] H. J. Chao, "A novel architecture for queue management in the ATM network," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 7, pp. 1110–1118, Sep. 1991.
[43] S.-W. Moon, J. Rexford, and K. G. Shin, "Scalable hardware priority queue architectures for high-speed packet switches," *IEEE Trans. Comput.*, vol. 49, no. 11, pp. 1215–1227, Nov. 2000.
[44] P. L. Kafle, A. Intarapanich, A. B. Sesay, J. Mcrory, and R. J. Davies, "Spatial correlation and capacity measurements for wideband MIMO channels in indoor office environment," *IEEE Trans. Wireless Comms.*, vol. 7, no. 5, pp. 1560–1571, May 2008.
[45] K. Nikitopoulos, D. Zhang, I.-W. Lai, and G. Ascheid, "Complexity-efficient enumeration techniques for soft-input, soft-output sphere decoding," *IEEE Commun. Lett.*, vol. 14, no. 4, pp. 312–314, Apr. 2010.
[46] L. Liu, F. Ye, X. Ma, T. Zhang, and J. Ren, "A 1.1-Gb/s 115-pJ/bit configurable MIMO detector using 0.13-μm CMOS technology," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 9, pp. 701–705, Sep. 2010.
[47] K. Nikitopoulos and G. Ascheid, "Approximate MIMO iterative processing with adjustable complexity requirements," *IEEE Trans. Veh. Technol.*, vol. 61, no. 2, pp. 639–650, Feb. 2012.
[48] C.-H. Liao *et al.*, "Combining orthogonalized partial metrics: Efficient enumeration for soft-input sphere decoder," in *Proc. IEEE 20th Int. Symp. Pers., Indoor Mobile Radio Commun.*, Tokyo, Japan, Sep. 2009, pp. 1287–1291.

GEORGIOS GEORGIS received the B.Sc. degree in physics from the Aristotle University of Thessaloniki, and the M.Sc. degree in computing and control and the Ph.D. degree in computing from the University of Athens, Greece. He is currently a Research Fellow with the 5G Innovation Center, University of Surrey, Guildford, U.K. His research interests include digital signal filtering, real-time multidimensional signal processing, and the design of relevant parallel algorithms and architectures, artificial intelligence, and expert systems.

KONSTANTINOS NIKITOPOULOS (M'07) received the Diploma degree in physics and the M.S. degree in electronics and telecommunications from the National and Kapodistrian University of Athens, Athens, Greece, and the Ph.D. degree from the National and Kapodistrian University of Athens, in 2005. Since then, he has held research positions with the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, with the California Institute for Telecommunications and Information Technology, University of California at Irvine, Irvine, and with the Computer Science Department, University College London. He has also been a Consultant for the Hellenic General Secretariat for Research and Technology, where he also served as a National Delegate of Greece to the Joint Board on Communication Satellite Programs of European Space Agency. He is currently a Lecturer (Assistant Professor) with the Electrical and Electronic Engineering Department, University of Surrey, Guildford, U.K., where he is actively involved with research in the 5G Innovation Center. His recent interests lie in signal processing for wireless communication systems, with an emphasis advanced transceiver design, signal processing techniques for large MIMO systems and massive parallel processing. He is a recipient of the prestigious First Grant of the U.K.'s Engineering and Physical Sciences Research Council.



KYLE JAMIESON received the B.S. and M. Eng. degrees in computer science from the Massachusetts Institute of Technology, USA, and the Ph.D. degree in computer science, from the same university, in 2008. He is currently an Assistant Professor with the Department of Computer Science, Princeton University, USA, and an Honorary Reader with University College London, U.K. His research focuses on building mobile and wireless systems for sensing, localization, and communication that cut across the boundaries of digital communications and networking. He received the Starting Investigator Fellowship from the European Research Council in 2011, the Best Paper awards at USENIX 2013 and CoNEXT 2014, and the Google Faculty Research Award in 2015.