EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

Sub-optimal Deep Pipelined Implementation of MIMO Sphere Detector on FPGA

Minh Thuong Nguyen[1], Xuan Nam Tran[2], Ngo Vu Duc[3], Quang Kien Trinh[2,*], Duc Thang Nguyen[2], Tien Anh Vu[2]

[1]Military Information Technology Institute, Hanoi, Vietnam
[2]Le Quy Don Technical University, Hanoi, Vietnam
[3]Hanoi University of Science and Technology, Hanoi, Vietnam

Abstract

Sphere detector (SD) is an effective signal detection approach for the wireless multiple-input multiple-output (MIMO) system since it can achieve near-optimal performance while reducing significant computational complexity. In this work, we proposed a novel SD architecture that is suitable for implementation on the hardware accelerator. We first perform a statistical analysis to examine the distribution of valid paths in the SD search tree. Using the analysis result, we then proposed an enhanced hybrid SD (EHSD) architecture that achieves quasi-ML performance and high throughput with a reasonable cost in hardware. The fine-grained pipeline designs of 4×4 and 8×8 MIMO system with 16-QAM modulation delivers throughput of 7.04 Gbps and 14.08 Gbps on the Xilinx Virtex Ultrascale+ FPGA, respectively.

Received on 26 August 2022; accepted on 28 December 2022; published on 29 March 2023

Keywords: SDM-MIMO, Sphere Detection, K-best, MIMO, FPGA, Sub-Optimal, Wireless communication

doi:10.4108/eetinis.v10i1.2630

1. Introduction

Multiple-input multiple-output (MIMO) communication is one of the critical technologies for modern wireless systems. MIMO takes advantage of spatial diversity to wireless for simultaneous transmission of signal symbols, hence, significantly improving the system capacity and spectrum efficiency[1]. MIMO systems have become a core component of HSPA+ (3G), LTE (4G), 5G, and other modern wireless communication systems. One of the technical challenges in MIMO technology is that the receiver side needs to simultaneously detect multiple symbols accurately in real-time, especially for systems with a large number of antennas.

Maximum-likelihood (ML) detection achieves optimal performance in MIMO systems with the cost of high complexity, which exponentially increases with the number of transmission antennas and modulation order. Therefore, this algorithm is not practical for real-time MIMO detection but is typically used as a baseline design. The sphere detector (SD) has emerged as one of the primary methods that could substantially reduce the detection complexity in the MIMO system while maintaining an acceptable bit error rate (BER) (i.e., comparable to that of the optimum ML detector) [2, 3]. SD was first proposed by Fincke and Posh [4, 5] for locating the closest vectors in the given lattice. SD could be implemented using software and hardware or hybrid platforms and has been actively studied in the literature. Most SD variants focus on optimizing SD by finding the best balance between their error performance and associated computational complexity. Original SD algorithm exhibits variable complexity and decoding latency, which could be beneficial for software implementation but hardly achieves real-time throughput for practical MIMO systems. Within the boundary of this study, we focus on SD architecture that is suited for the MIMO signal detection accelerator.

Fixed-complexity sphere detection (FSD) has been proposed in [6–12], where the search tree is divided into the full-expansion stage and the single expansion stage. The former considers every possible path, while the latter expands only to the best solution within its local tree. By using a fixed number of search

*Corresponding author. Email: kien.trinh@lqdtu.edu.vn

paths, FSD overcomes the drawback of varying latency in the conventional SD and could achieve a near-ML performance at the expense of sufficiently large computation [6, 7]. Several hardware implementations have been proposed and demonstrated [8–12]. In [8], the authors introduced an FSD prototype on Virtex II FPGA having constant throughput of 600 Mbps. In [9], the authors demonstrated an FPGA-based FSD targeted for WiMax application that achieves 800 Mbps at 102 Mhz. In [10], Wu *et al*. proposed four-nodes-per-cycle parallel FSD architecture that is very area-efficient, though the achievable throughputs are just 213.3 Mbps and 27.7 Mbps on 130 nm CMOS and FPGA, respectively. Furthermore, a pipelined FPGA SD architecture using the Schnorr-Euchner algorithm was proposed in [11] that achieves near ML accuracy but with a limited throughput of 48 Mbps. In [12], the authors demonstrated a 90 nm VLSI implementation of modified FSD that limits the number of searches in sub-tree by the best child node prediction technique. The simulated result shows a maximum throughput of 2.2 Gbps, which so far is the fastest hardware of FSD implementation in our record.

K-best is another popular SD accelerator architecture and has been actively studied recently [13–22]. The K-Best SD is based on breadth-first search, which considers only the best (K) candidates at each level to be passed to the next level of the search tree. Thus, ML could be considered a special case of K-best with a sufficiently large K [13]. The conventional K-best sphere detectors use the same K value at all levels [13–18], while recent developments offer more flexible and efficient designs [19–22]. Among those, several practical K-Best SD has been prototyped on hardware [16, 17, 19–22, 28] that show advantages over FSD in both complexity and throughput.

A variant of the K-best algorithm was proposed in [19], where the search tree is split into sub-branch with a lower value of K, which can significantly reduce the complexity while achieving good BER performance for a 4x4 MIMO system. In [20], the variable K-best has been proposed to reduce the complexity without degrading the BER; the corresponding 180 nm VLSI implementation [20] exhibits a throughput of 1.5 Gbps. Furthermore, the sorting reduced K-best (SR K-best) detector was then proposed in [17], which is based on two methods of staging reduction and QR transform performed on complex numbers; the corresponding VLSI implementation on CMOS 90 nm achieves a throughput of up to 3.1 Gbps. In [21], Wu *et al*. proposed a bounded selective spanning technique that effectively removes invalid nodes from the K best nodes, thus reducing the necessary computation compared to the prior works, though the implementation on FPGA achieves moderate throughput of 484.8 Mbps for a 4×4 MIMO detector. In [28] adopted non-constant K-best algorithm has been proposed to keep more survival nodes in top search tree layers and reduce the computational complexity in bottom layers as opposed to the conventional K-best algorithm; the corresponding VLSI implementation on CMOS 90 nm for 4×4 MIMO achieves a throughput of up to 4.08 Gbps. In [18], a hybrid K-best architecture was introduced where K is maintained throughout the detection, but only a low-complexity successive interference cancellation is applied in lower levels. The corresponding proposed design implemented on 65 nm CMOS achieves the throughput of 3.2 Gbps for a 64-QAM 4×4 MIMO detector. In [16], the authors proposed a sorted-Cholesky lattice reduction K-best that exhibits 3.85 Gbps for the 8×8 MIMO system with 40 nm VLSI implementation. In the most recent work [22], Wu *et al*. proposed a robust bounded spanning with a fast enumeration and configurable SD algorithm, implemented on FPGA for a 4×4 MIMO detector. The proposed design permits an adequate balance between the detection accuracy and the hardware efficiency and could achieve quasi-ML accuracy with a 74% reduction in complexity, though the achievable throughput is moderate (494 Mbps).

This paper proposes a novel hybrid SD architecture of MIMO detection accelerator on FPGA, so-called EHSD. The proposed architecture could be viewed as a hybrid version of conventional FSD and variable K-best. As the fundamental difference, our design approach relies first on extensive simulation and statistical analysis to determine the most optimal SD configuration that balances the trade-offs between complexity, accuracy, and hardware cost. The main contributions of this work are listed as follows:

1. We proposed a statistical analysis approach to systematically study the relationship between the sphere radius, the BER performance, and design complexity to systematically optimize search tree topology and the number of calculation units for each level, aiming at the sub-optimal design that reaches quasi-ML BER performance.

2. We carried a thoroughly micro-architectural optimization for fundamental computation units and developed techniques for searching and sorting candidate points in the lattice for each SD level that could perform a global comparison for node selection with a reasonable hardware cost.

3. We have implemented on FPGA 4×4 and 8×8 MIMO configurations of the proposed EHSD detection algorithm. The fine-grained pipelined design can deliver the maximum throughput of 7.04 Gbps (14.08 Gbps) and fixed latency of 145.45 ns (400 ns) for 4×4 (8×8) MIMO systems

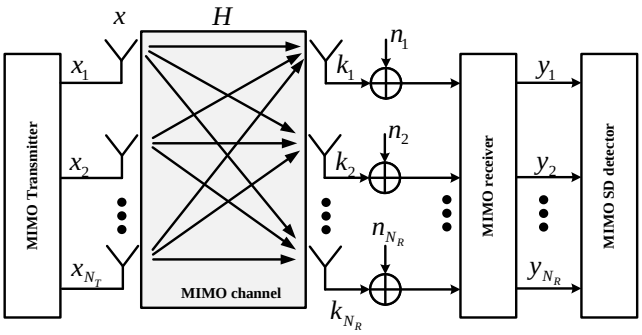

Figure 1. Block diagram of the general MIMO system with SD.

with less resource utilization compared to the prior works.

The rest of this paper is organized as follows. Section 2 describes the system model of the MIMO system with SD. Section 3 analyzes the distribution of valid nodes in the sphere detection search tree. The proposed EDSD architecture is presented in Section 4, followed by the hardware design of the EHSD in Section 5. Section 6 provides the simulation and implementation results. Finally, Section 7 concludes the paper.

2. System Model

Consider a MIMO system with N_T transmit and N_R receive antennas, as shown in Fig. 1. The MIMO channel is characterized by its complex channel matrix $\left(h_{ij}\right)^{N_R \times N_T} \in \mathcal{C}^{N_R \times N_T}$, whose elements are drawn independently from the complex Gaussian distribution with zero mean and unit variance. Those parameters indicate the attenuation and phase shift for each path from the transmitter to the receive antennas; they are assumed to be perfectly known in advance (i.e., through the channel estimation stage). For transmission, elements x_i of complex signal vector $\boldsymbol{x} = (x_i)^{N_T \times 1} \in \boldsymbol{\Omega} \subset \mathcal{C}^{N_T \times 1}$ are sent concurrently through N_T transmit antennas, where $\boldsymbol{\Omega}$ is the set of the signal modulation constellation. Hence, the received N_R-element complex signal vector $\boldsymbol{y} = (y_i)^{N_R \times 1} \subset \mathcal{C}^{N_R \times 1}$ can be expressed as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{n} = (n_i)^{N_R \times 1} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I})$ is a complex Additive White Gaussian Noise (AWGN) vector.

The ML detector performs an exhaustive search for all the possible symbol vectors in $\boldsymbol{\Omega}$ set to obtain the one with a minimum squared error:

$$\widehat{\boldsymbol{x}}_{ML} = \arg\min_{\boldsymbol{x} \in \boldsymbol{\Omega}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2, \tag{2}$$

The computational complexity of ML detection grows exponentially with the number of antennas in the MIMO system. Therefore, the ML detector is complicated to apply in the real MIMO system detector. Hence ML detection can be replaced by the SD to reduce the computational complexity. The SD only calculates the Frobenius norm for candidate points inside a hypersphere that is formed around the received signal vector with a predetermined radius r_{sph}. The equation (2) can be transformed as

$$\widehat{\boldsymbol{x}}_{SD} = \arg\min_{\boldsymbol{x} \in \boldsymbol{S}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2, \tag{3}$$

where $\{\boldsymbol{S} \subset \mathcal{C}^{N_T \times 1} : \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\| \leq r_{sph}\}$ is a set of all possible points in the lattice $\boldsymbol{H}\boldsymbol{x}$, whose distance to $\boldsymbol{y}$ is always smaller than the hypersphere radius r_{sph}. Choosing a suitable value of r_{sph} is essentially important for determining the SD computational complexity and BER performance. To further reduce the amount of computation in SD, equation (3) can be transformed into an identical problem by applying the QR decomposition to the channel matrix, that is $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}$ where matrix $\boldsymbol{Q}$ is a unitary matrix whose size is $N_R \times N_R$ and $\boldsymbol{Q}\boldsymbol{Q}^H = \boldsymbol{I}$ while $\boldsymbol{R}$ is an $N_R \times N_T$ upper triangular matrix. Replacing $\boldsymbol{H}$ by $\boldsymbol{Q}\boldsymbol{R}$ and after simple transformation, equation (1) becomes

$$\widetilde{\boldsymbol{y}} = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{Q}^H\boldsymbol{n}, \tag{4}$$

where $\widetilde{\boldsymbol{y}} = \boldsymbol{Q}^H\boldsymbol{y}$. Note that $\boldsymbol{Q}^H\boldsymbol{n}$ has the same statistics as $\boldsymbol{n}$, hence equation (3) is equivalently characterized as

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbf{S}} \|\widetilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|^2, \tag{5}$$

where $\widehat{\boldsymbol{y}} = \boldsymbol{R}\boldsymbol{x}$. Equation (5) can be calculated through the following cost function:

$$\mathcal{D}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) = \|\widetilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|^2 \leq {r_{sph}}^2. \tag{6}$$

Since the matrix $\boldsymbol{R}$ is the upper triangular, the cost function $\mathcal{D}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}})$ is also a partial Euclidean distance that can be calculated recursively from one transmit antenna to another:

$$\mathcal{D}_m(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) \triangleq \sum_{i=m}^{N_R} \left(\widetilde{y}_i - \sum_{j}^{N_T} R_{ij} x_j \right)^2, \tag{7}$$

$$\mathcal{D}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) = \mathcal{D}_1(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}), \tag{8}$$

$$\mathcal{D}_{m-1}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) = \mathcal{D}_m(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) + \left(\widetilde{y}_{m-1} - \sum_{i=m-1}^{N_T} R_{m-1,i} x_i \right)^2, \tag{9}$$

where $\widetilde{y}_{m-1}$ is the $(m-1)$-th element of the received signal vector after multiplication of the received signal vector by $\mathbf{Q}^H$; $R_{i,j}$ is an element of matrix $\boldsymbol{R}$ that belongs to the i-th row and the j-th column, and the cost function $\mathcal{D}_m(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}})$ is a partial Euclidean distance of the candidate symbol $\boldsymbol{x}$ at the m-th search level. For

all possible transmit symbol vectors that are satisfied $\boldsymbol{x} \in \{\boldsymbol{S} \subset \boldsymbol{\mathcal{C}}^{2N\times 1} : \left\|\boldsymbol{Rx} - \widetilde{\boldsymbol{y}}\right\| \leq r_{sph}\}$, we set $\mathcal{D}_{2N+1}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) = 0$, and have the following inequality

$$\mathcal{D}_{m-1}(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}) \leq {r_m}^2 - \mathcal{D}_m(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}), \tag{10}$$

$${r_m}^2 = {r_{sph}}^2 - \sum_{i=m+1}^{N_R} \mathcal{D}_i(\widetilde{\boldsymbol{y}}, \widehat{\boldsymbol{y}}), \tag{11}$$

where $m = 2N, 2N-1, ..., 1$ $(N = N_R = N_T)$. Choosing the value of r_{sph} critically affects the computational complexity and the system performance. If r_{sph} is large, a large number of candidate symbols are covered in the hypersphere, which improves BER with the trade-off in an increase of the computational workload. In contrast, when r_{sph} is small, the correct solution has a greater chance of staying out of the chosen hypersphere. Hence, the initial search radius and the expected quantity of lattice points in the hypersphere must be judiciously selected to balance computational complexity and system performance.

3. Statistical Analysis of the Valid Node Distribution in The SD Search Tree

In the SD search tree, the number of nodes to be traversed at the k-th level is $M^{(2N+1-k)}$, where M is the modulation order, N is the number of receive antennas in the system, and k receives value from 1 to $2N$. So, with a large value of r_{sph}, the number of valid nodes from the i-th level to $(i-1)$-th level increases by M times (i.e., exponentially). We consider nodes in the SD search tree valid if the distance of the points generated by the $\boldsymbol{Hx}$ lattice corresponding to those nodes to the received signal vector is less than the value of an corresponding partial Euclidean in that level. The number of valid nodes in the sphere at each level could be randomly distributed and depends on the model parameters, including the sphere radius r_{sph}, E_b/N_0, and channel status. Among those, the chosen spherical radius significantly influences the number of valid nodes at each level.

This section analyzes the number of valid nodes (the survival search paths) distributions at every SD level w.r.t different r_{sph} and E_b/N_0 levels. Two conventional SD models have been built on MATLAB for 4×4 and 8×8 MIMO systems. Using sufficiently large random inputs, we perform the basic SD algorithm to acquire the statistical report of the node distribution at each search tree level. Each element of the equivalent channel matrix $\boldsymbol{H}$ is also normalized to follow the standard Gaussian distribution (AWGN model adopted), and the E_b/N_0 varies from 0 to 15 dB while the radius varies from $0.3E_T$ to $1.1E_T$ with E_T is the transmitter power, considering each element of the input signal vector are normalized with the average signal energy on each transmit antenna is 10.

The simulation results for different antenna configurations are reported in Fig. 2 and Fig. 3, respectively. In those plots, the horizontal axis denotes the level number in the search tree, where the numbers of SD levels (the rank of the matrix $\boldsymbol{H}$) are 8 and 16 for 4×4 and 8×8 MIMO configurations, respectively. The y-axis represents the mean (μ_{node} in Figs. 2, 3.b) and the number of nodes corresponding to the cumulative distribution of 99.999% coverage [1] (cdf_{99} in Figs. 2, 3.a) of total survival search nodes in every level. These results were statistically evaluated for each (r_{sph}, E_b/N_0) pair. The evaluation is considered reliable when the increase in the number of random input patterns having little impact on its statistical results. In our analysis, 1 million 16-QAM vectors were generated. Note that to have a fair comparison between antenna configurations, we normalized the initial radius by the transmitter power E_T.

From Figs. 2-3, some notable trends could be observed for the two models. First, increasing the sphere radius r_{sph} leads to an increase in both μ_{node} and cdf_{99} in every level for 4×4, and 8×8 MIMO systems. As mentioned earlier, when r_{sph} is too small, there may be no solutions inside the sphere, making it impossible to locate the correct solution, especially in scenarios with high E_b/N_0. On the other hand, when r_{sph} is too large, the possible solution may surge up, leading to unbearable hardware cost. The value of cdf_{99} is much larger than μ_{node} meaning that it would be too costly to process all possible valid nodes. Therefore, it is a critical design knob to set an adequate level of r_{sph} to balance the hardware efficiency under a target BER. This tendency is also observed for E_b/N_0, i.e., increase E_b/N_0 implies a similar but weaker impact on the μ_{node} and cdf_{99}.

Also, from the statistical data, the numbers of valid nodes apparently are not the same for all levels in the search tree. The peaks of the μ_{node} depend mainly on the r_{sph}, typically at the 3-rd to 6-th level in 4×4 MIMO, and 8-th to 12-th level for the 8×8 MIMO system. This can be explained by two opposite effects: the expanding number of valid nodes while lowering the search lever; and the shrinking of the sphere corresponding to the reduction of effective sphere radius $\mathcal{D}_m$ (see Eq. (10)). In detail, the solution vector is not entirely determined at the top level ($D_{N_R} = r_{sph}$), so the possible choices are small. The search tree expands rapidly at a lower level, but as soon as a certain component in the solution vector is determined, $\mathcal{D}_m$ is reduced accordingly. As a result of the combined effects, the peaks are typically found somewhere in the middle levels.

[1] For example, if this $cdf_{99} = 100$ at the i-th level means that the 99.999% number of the valid nodes ranges from 0 to 100 and the number of valid nodes greater than 100 is equal to just a small portion of 0.001%.

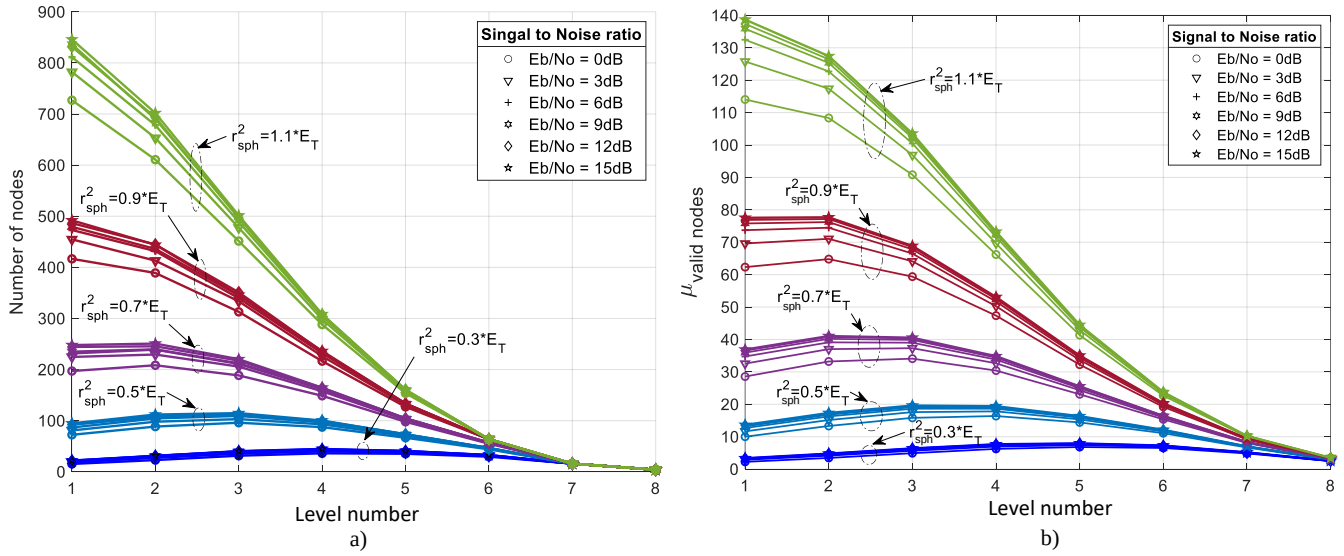

Figure 2. (a) The number of nodes corresponding to cumulative distribution of 99.999% coverage of all available valid node .and (b) the mean (μ_{node}) of valid search nodes in each level of SD search tree in 16-QAM 4×4 MIMO system, simulated for 1 million random input patter with AGWN.

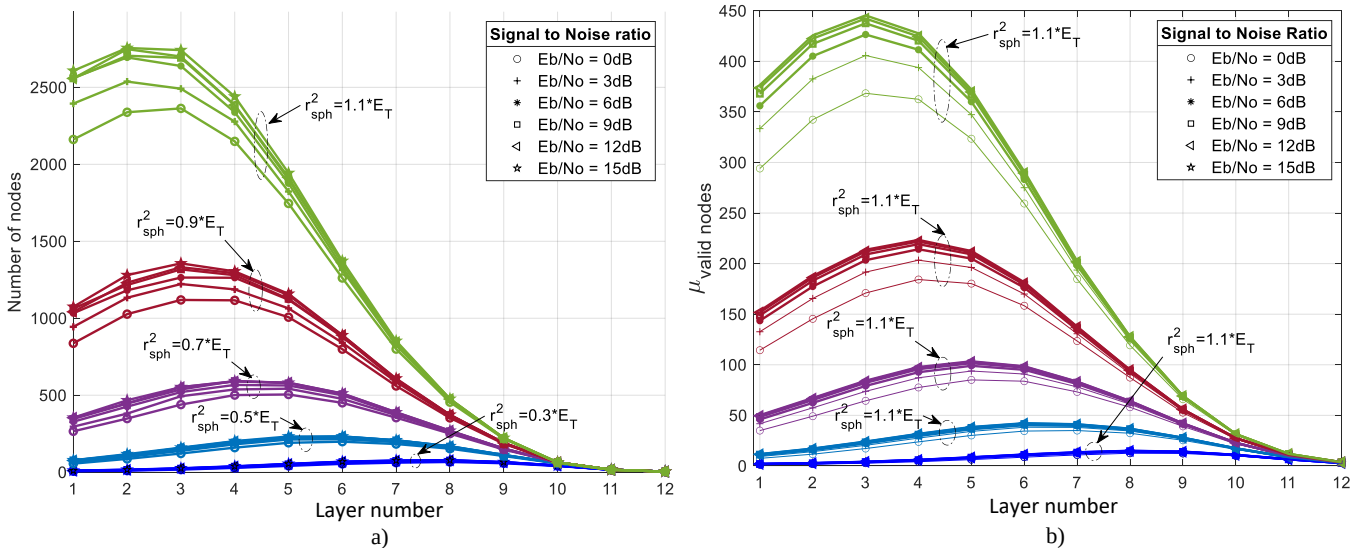

Figure 3. (a) The number of nodes corresponding to cumulative distribution of 99.999% coverage of all available valid node; (b) the mean (μ_{node}) of valid search nodes in each level of SD search tree in 16-QAM 8×8 MIMO system. Simulated for 1 million random input patterns with AGWN.

It is also noticeable that SD in an 8×8 MIMO is much more computationally intensive than a 4×4 MIMO system. Indeed, considering the same normalized initial radius $r^2_{sph} = 0.5E_T$, the maximum of both μ_{node} and cdf_{99} of 8×8 MIMO is found almost 4.3 times higher than that of 4×4 MIMO. The high cdf_{99} indicates that in the worst-case scenarios, the number of search paths could be very large. For example, let us consider 4×4 MIMO systems at $r^2_{sph} = 0.5E_T$; E_b/N_0 ranging from 0 dB to 15 dB, we would need to process maximum for about 95 to 115 nodes at 3-rd levels (see Fig. 2.a) while corresponding maximum μ_{node} is only about 15 to 20, i.e., approximately 5 to 6 times lower. Similar results can be observed for other MIMO systems and radius levels.

As a short summary, the number of valid nodes is reported to be a few orders of magnitude less than the total number of nodes in each search tree level. It confirms the fact that the required computation workload for SD must be much lower than that of the

exhaustive search in ML. Furthermore, not all the valid nodes are needed to be processed when we consider the ultimate design goal is BER performance, as we will show in the next Section. The statistical results also indicate that the critical search level, where the number of valid nodes reaches the maximum, is found somewhere at a few middle levels. These findings are foundations for the study of the topology of the SD implementation.

4. The Proposed EHSD Architecture

From the above analysis, the number of valid nodes varies across the search level, and this has been exploited for further enhancing the K-best algorithm in [17, 19, 20, 23]. In those works, a larger K is used in early levels to avoid error propagation, and a smaller K in the top levels reduces computational complexity. This pattern, however, is not necessarily optimum according to the statistical results in the previous section. In addition, apart from the statistical data, the SD micro-architecture has to be optimized according to the system BER. The EHSD proposed in this work shares a similar idea on how the BER performance can be maintained using much fewer resources for practical hardware implementation. However, in our approach, instead of empirically selecting K_i for each i-th level, we determine the level configuration for a specific BER target based on the statistical result from the analysis in the previous Section. We also propose here a sorting method that guarantees to cover the most potential valid nodes in each considered level. The latter minimizes the miss rate of correct nodes and allows the safe removal of a large number of valid nodes that are highly likely not the true solution. After that, the estimated computational workload helps to best determine the number and structure of the compute blocks in each search level, which is meaningful for hardware implementation. Finally, we quantify the intrinsic relationship between r_{sph} and E_b/N_0. It helps optimizes sphere radius based on E_b/N_0 to minimize BER, targeted for fixed hardware detector design. In the following, the design of 4×4 MIMO EHSD is described in detail. The design of a higher MIMO order detector can be developed in the same approach and will be summarized later.

4.1. Nodes Selection Using Recursive Sorting Algorithm

As analyzed above, when reaching the middle levels, the number of generated nodes is much higher than those discarded outside the hypersphere. Hence, the major workload is concentrated at a few middle levels. Limiting the number of nodes at the middle levels is crucial for reducing the computation workload and finding an appropriate resource allocation. Interestingly, our further study shows that we eventually require much fewer nodes while still maintaining good BER by adopting a sorting algorithm. The sorting algorithm takes out the best number of candidates, whose number typically is much smaller than the number of valid nodes. In the following, we will present the detailed method to optimize the value of K first by sorting and selecting the best candidates among the valid nodes.

For usual selection, all survival paths are divided into K subsets. Then the best candidates, which are passed to the next level, are chosen as the local best candidate for each subset. This is called group sorting or local sorting (LS), which is simple and straightforward to be implemented on hardware. Nonetheless, this sorting method leads to the possibility that the best node in a subset is still worse than the other nodes in the different subsets, which have been excluded. An increasing number of K could solve the problem, but it also increases the hardware cost and processing latency.

For getting the true best K nodes from all survival nodes, the global sorting (GS) algorithm was used. In this work, we proposed a recursive GS method, which is well suited for hardware implementation. Initially, all considered nodes are divided into M groups of L elements. Then each group is sorted by L-to-K sorting block to get the $M \times K$ best nodes. The $M \times K$ sorted nodes are then divided into $M \times K/L$ group to be sorted using the same sorting block. This process is repeated until K best nodes are finally retrieved. We use Batcher's sorting algorithm [24] for designing the sorting core, which is a fast and area-efficient algorithm for a recursive design.

We conduct a typical study to compare the effect of LS and GS methods. Without loss of generality, we assume that K best nodes are extracted from available valid nodes at the considered level after each sorting. To compare two sorting methods, we analyze the error probability that the true solution is not in the sorted list of K best nodes (namely *sorting error rate (SER)* [2]). The lower SER means the method has better performance and vice versa.

Fig. 4 shows the dependence of the SER at a 5-th level in the 4×4 MIMO system on the radius r_{sph} and E_b/N_0 when $K = 4$, i.e., only 4 best nodes are extracted from all valid nodes after sorting. We found that with increasing the radius and decreasing E_b/N_0 ratio, the SER using the LS method increases significantly compared to that of the GS method. Specifically, at the same level of E_b/N_0 that GS is always better than the LS method. This advantage of GS is more significant with a large

[2] In detail, the SER is statistically calculated by the number of wrong solutions divided by the total number of trials.

radius. The two methods perform similarly only when the signal level is reasonably good (i.e., $\geq$ 10 dB) and the node selection is made almost without error.

Furthermore, Fig. 5 shows the SER with fixed $r_{sph} = 3$ at several critical levels from 3 to 6 for the 4×4 MIMO system. Here we set the number of sorting nodes $K = 4$ at 3-rd, 4-th levels, and $K = 8$ at 5-th, 6-th levels, which is based on the statistical analysis on the valid node in Section 4. Very similar results can be observed here. The SERs of the two methods are the same only at very high E_b/N_0 ($\geq$ 10 dB). In contrast, the GS is superior to the LS in most other cases, especially for larger ($K = 8$) and higher levels (5-th, 6-th levels). This could be explained by the number of valid nodes at these levels becoming much larger than in cases with smaller K and lower levels. As the number of valid nodes increases, there is a higher probability that the real solution is located outside the sphere when selecting the nodes just by LS, as we discussed earlier.

Overall from this quantitative analysis, it can be concluded that the GS method exhibits much better performance compared to the LS. This advantage essentially comes with the inevitable cost of resource utilization and processing latency.

4.2. The Dependence of BER on r_{sph}

In this subsection, we investigate the dependence of BER on r_{sph}, aiming to find the best radius value to meet different channel conditions from the design. Finding the minimum radius under a system BER target is essential to reduce the amount of computation. Based on the MATLAB model in the previous Section, we conduct simulation with random inputs to evaluate

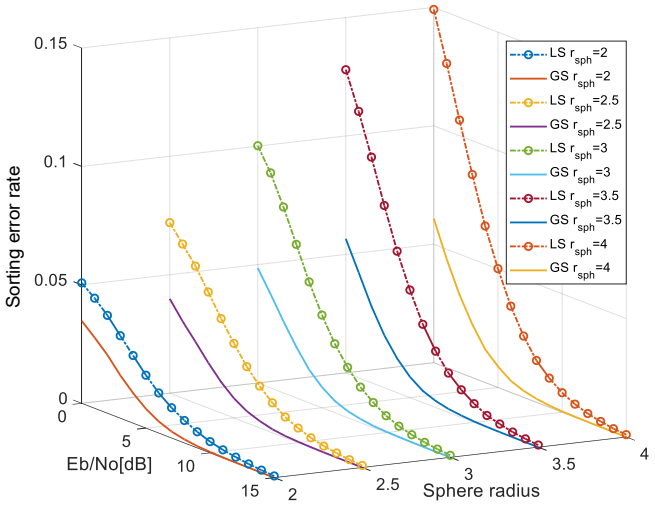

Figure 4. The sorting error rate at 5-th level using two sorting models: the local sorting (LS) that takes the 4 local best nodes and the global sorting (GS) that takes the global best 4 nodes for 4×4 MIMO with 16-QAM.

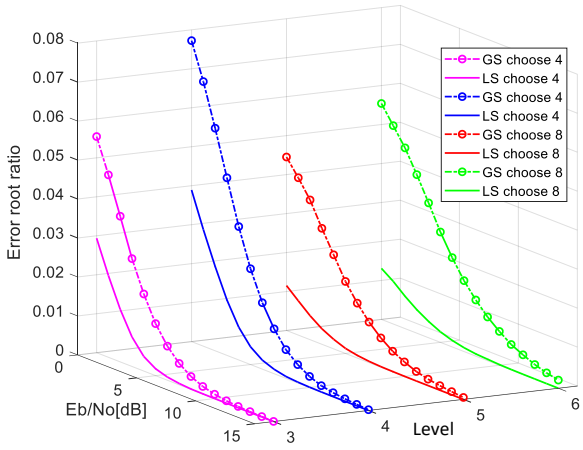

Figure 5. The ratio between sorting error rate with $r_{sph} = 3$ for two sorting models: the local sorting that takes the 4 (8) best nodes and the all-inclusive sorting that takes the 4 (8) best nodes at levels 3, 4 (5, 6), respectively.

system BER, given the ground truth solution. From the design perspective, the target BER and the SD radius can be considered as the runtime parameters, i.e., can be set during the operation mode. Fig. 6 presents the analysis of BER dependence on the SD radius and E_b/N_0 level. From Fig. 6, it is clear that increasing SD radius help to reduce the BER, but that is effective for only a small range of r_{sph}. Each BER curve has two inflection points, in which the second inflection point gives a good BER value with the smallest radius. For $r_{sph} > 3$, the BER becomes saturated. The optimal r_{sph} depends on E_b/N_0, but it is located somewhere from 2.5 to 3.5. This finding is important since it permits limiting the ranges of search nodes without sacrificing BER performance.

4.3. The Proposed EHSD Architecture

From the above analysis, we found that BER is very sensitive to small values of r_{sph} and becomes saturated with a large enough sphere radius. For example, with the 4×4 MIMO system, the simulation results from the MATLAB model indicate that $r_{sph} = 3$ seems to be the best for a wide range of E_b/N_0. In the following, we propose a dedicated architecture that could leverage both sorting and BER-radius relationship to reduce the computational workload.

The core processing block in this architecture is the level processing block (LPB). This block calculates the PDE, i.e., $\mathcal{D}_m$ in Eq. (10), corresponding to the selected modulation symbol at the m-th level. The calculation is not the same for every level because $\boldsymbol{R}$ in Eq. (4) is an upper triangle matrix. In detail, only the y_8 component is the assigned symbol at the top level, so only a single multiplication and addition is needed. Lowering the search level increases the computation complexity of

the LPB linearly. We hence propose a general structure of SD as follows:

1. *Full expanding cluster*: First few levels, as mentioned, are not the critical ones in the sense that the complexity of LBP is still low and the number of search nodes is small. In the first three levels, from 8 down to 6, the core processing will cover all search nodes because the computational complexity in these levels is not high. The maximum number of nodes is limited and is 64 at level 6. As reported in the previous section, to cover 99.999%, this level would need 25 LPBs. However, adding sorting blocks to calculate the number of valid nodes would be more complicated and costly than adding LPB blocks [20, 26].

2. *Variable K-best cluster*: The remaining levels are selectively applied K-best, where K is statistically determined and optimized with the sorting algorithm. According to the results in Fig. 2, after level 6-th, the number of valid nodes is reduced from total 256 to just 26 to 32 (for coverage of 99.999% valid nodes). However, by applying the GS sorting algorithm, this level need only $K_5 = 8$ nodes to achieve a fairly good BER. Similarly, in this particular MIMO 4×4 system, the statistical results confirm that the set of $K_{4-1} = (8, 4, 4, 4)$ is a suitable K set for the implementation.

Fig. 7 plots the simulated BER for several EHSD configurations listed in Table 1 and for Zero-forcing, Minimum-Mean-Square-Error (MMSE), conventional SD algorithms with $r_{sph} = 1 - 4$, and the baseline ML. From the figure, it is clear that all proposed EHSD outperform simple algorithms such as ZF and MMSE

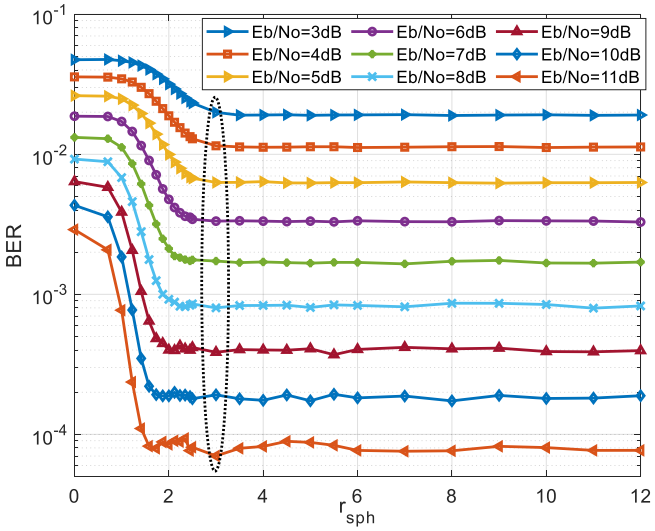

Figure 6. The dependence of BER on r_{sph} for 4×4 MIMO with 16-QAM.

Table 1. Configurations of EHSD algorithm

*Configuration	Number of chosen K_i best solutions for each level i-th							
	8-th	7-th	6-th	5-th	4-th	3-th	2-th	1-th
Config.1	4	16	32	8	6	4	4	1
Config.2	4	16	32	16	8	4	4	1
Config.3	4	16	20	24	16	8	4	1
Config.4	4	16	24	26	16	8	4	1
Config.5	4	16	8	8	4	4	4	1

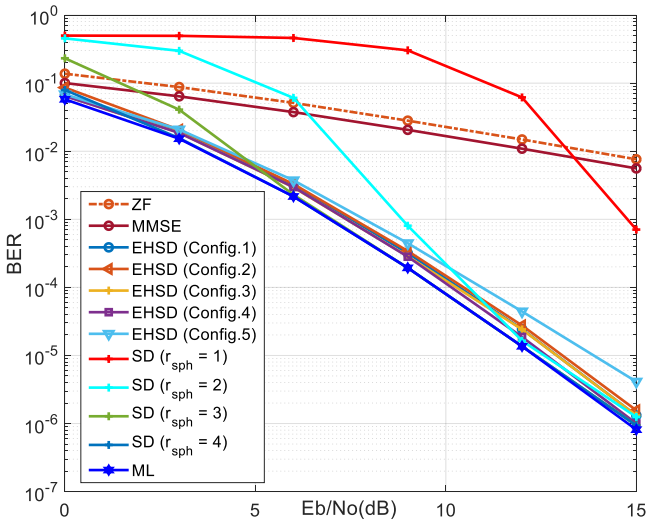

Figure 7. BER performance Comparison different EHSD configuration with ZF, MMSE, conventional SD and ML methods.

and SD with $r_{sph} < 2$ and have the BER quite close to that of SD with $r_{sph} = 4$ and ML. Among the configurations, the last one (K=[4 16 8 4 4 4 1]) has BER performance slightly worse than the other, but that comes with a significant reduction in the hardware cost, and that result is quite in line with the proposed design based on the statistical analysis mentioned above.

By applying a similar method, the suitable configuration for the 8×8 MIMO system is $K_{8\times8}$ = [4 16 28 28 24 16 12 12 8 8 8 8 8 4 4 4 1]. It is worth noting that those proposed configurations are based on extensive simulation with sufficient large random inputs, hence, are almost independent of channel conditions and input pattern.

5. Hardware Design Of The Proposed EHSD

This section describes the hardware architecture of the sphere detector block for the $N \times N$ MIMO system with 16-QAM modulation that will be implemented based on the EHSD algorithm in Fig. 8. The RTL model 4×4 and 8×8 detectors are encapsulated on VHDL and have been simulated and synthesized using Xilinx Vivado 2019.2. In the following, the hardware architecture model is described in bottom-up order, i.e., starting with the basic processing block that will be used to

build up the more sophisticated block and the whole detector.

5.1. Level Processing Block

As mentioned in Fig. 8, the basic level processing block (LPB) is the primitive calculation element. The detailed structure of an LPB is shown in Fig. 9. LPBs perform the partial Euclidean radius calculation according to Eq. (11), which could be considered as the most basic calculation of the sphere decoding algorithm. To conveniently describe the design structure and unify the name for the input data, we rewrite the eq. 11 as follows.

$$r_{m-1}{}^2 = r_m{}^2 - \left(\sum_{i=1}^{2N} x_i R_{mi} - (\boldsymbol{Q}^H \boldsymbol{y})_m \right)^2, \qquad (12)$$

In Eq. (12), r_m is the spherical radius value at the m-th level, and $(\boldsymbol{Q}^H \boldsymbol{y})_m$ is m-th element of the vector $(\boldsymbol{Q}^H \boldsymbol{y})$. Instead of working with the radius, we perform all calculations with its squared form to avoid the square root operation, which normally is very expensive not only for hardware but software implementation.

According to the above formula, the LPB mainly uses the adder and multiplier resources while $\boldsymbol{Q}^H \boldsymbol{y}$ can be calculated only once because the received signals $\boldsymbol{y}$ and $\boldsymbol{Q}^H$ are known. Hence, we assume that the $\boldsymbol{Q}'\boldsymbol{y}$ matrix has been pre-computed before entering the detector block, and the $\sum_{i=1}^{2N} x_i R_{mi}$ value is directly taken into LPB without any further processing. The remaining intensive calculation here is the sum-of-the product, which requires N multiplications and N additions. Taking into account that x_i is a modulated symbol, e.g., for 16-QAM, it has only four choices $\{-3, -1, +1, +3\}$, the multiplication can be simplified by replacing it with adders, shifters, and multiplexers, i.e., in the form of the lookup operation. In addition, because $x_i R_{mi}$ is shared by many LPB at the same level, we pre-calculate these values to enable fast lookup and saving resources.

As a result, LPB uses only one multiplier (one DSP slice in FPGA) to perform the squaring calculation. In addition, because $\boldsymbol{R}$ is the upper triangular matrix, R_m has a $2N + 1 - m$ of zero elements. Therefore, it is unnecessary to allocate resources to perform all $2 * N$ additions but based on each specific level to perform the calculation $\sum_{i=1}^{N} x_i R_{mi}$. Therefore, the LPB block requires 2 to $\{(N + N/2 + N/4 + .. + 1) + 2\}$ adders and 1 to $2N$ 4-to-1 multiplexers depending on the processing level. For example, the 8-th level (in 4×4 MIMO) requires only 2 adders, while the last level needs all 9 adders. This is the main reason for the proposal of a full expansion cluster in Fig. 8 for the first 3 levels. At micro-architectural optimization, we divide the LPB block into 3 pipeline stages as shown in Fig 9. This aims to shorten the clock period and improve system throughput.

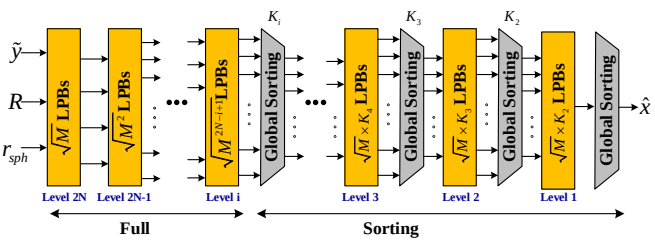

Figure 8. Block diagram of the proposed EHSD for 4×4 MIMO.

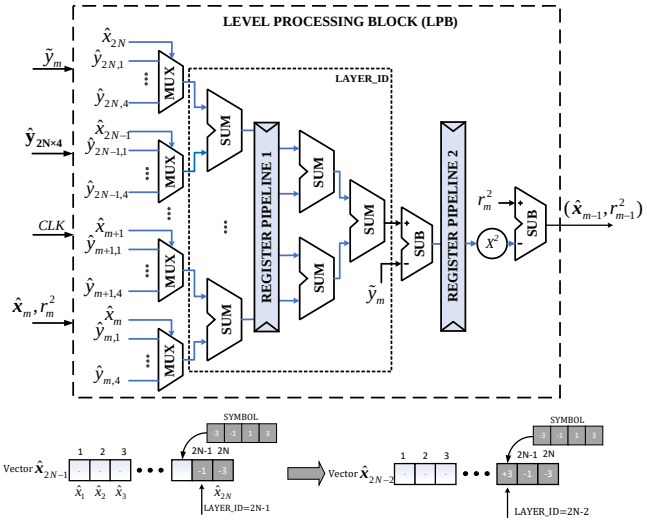

Figure 9. Detailed hardware implementation of a LPB.

The output of the LPB is vector $\hat{\boldsymbol{x}}$ according to the node level. For example, as shown in Fig. 9, after LPB processing from level $(2N-2)$-th, the output $\hat{\boldsymbol{x}}$ to the next $(2N-3)$-th level is updated $\widehat{x}_{2N-2} = +3$, assuming this symbol is selected for this particular node.

5.2. Node Selection Block

The valid node selection block (NSB) is an important component of the sphere detector. This block selects a certain number of valid nodes with priority according to the value of the sphere radius. Specifically, those valid nodes after LPB are sorted by the Batcher sorting followed by a K-best NSB, as shown in Fig. 10. Without loss of generality, it is assumed that inputs of the NSB at level $L-1$ is a set of $4K_L$ nodes, each represented by a matrix of level radius value and estimated transmitted signal $\left(r_{sph}\boldsymbol{X}\right)_{4K_L \times (2N_R+2-L)}$. The output of NSB keeps K_{L-1} best nodes, which are represented in a sorted matrix $\left(r_{sph}\boldsymbol{X}\right)_{K_{L-1} \times (2N_R+3-L)}$ by K_{L-1} rows corresponding to the K_{L-1} smallest radius r_{sph}.

Batcher sorter block: The Batcher sorter block is designed according to the algorithm in [24]. The sorter

outputs N input values and their associated data in ascending order. In our case, the radius values are the sorted value and QAM symbol and the corresponding partial Euclidean distance is the associated data. The smallest radius typically corresponds to the best node candidates though the ground truth received symbols are not always the closest node in the considered sphere.

The Batcher algorithm can be extended for arbitrarily $4K_L$ though the $4K_L$ normally is selected to be a power of 2. This especially enhances the design efficiency in hardware, where the resource is fully utilized, and the core has a balance pipelined structure. In addition, it is straightforward to design the higher input Batcher sorter by hierarchically constructing from the smaller sorter blocks.

K-best Selection Block: In the next processing step, The K-best selection block will filter out all the nodes with a larger distance from the considered points and pass to the next level (L-th) only K best nodes. Note that similar to the Batcher sorter, K-best selection can be optionally added pipelined registers for performance improvement.

5.3. The Overall Structure of Proposed EHSD

The generic structure of the EHSD is shown in Fig. 11. The input of the sphere detector block consists of the upper triangular matrix $\boldsymbol{R}$, the received vector $\widetilde{\boldsymbol{y}} = \boldsymbol{Q}^H \boldsymbol{y}$ (see Eq. (4), and the initial squared spherical radius value r_{sph}^2. The output of EHSD is the best solution inside the sphere of the 16-QAM symbol vector $\widehat{\boldsymbol{x}}$.

In addition to the main function blocks presented above, there is also the $\boldsymbol{XR}$ multiplier block that takes care of pre-computing the cases of the product $x_i R_{mi}$. $\boldsymbol{R}$ multiplier uses only multiplexer and adders, not DSP, which saves kernel resources. The calculated $x_i R_{mi}$ values are used for further processing.

Furthermore, the detector is constructed as the design in Fig. 11, i.e., for $N \times N$ MIMO, the first top levels (i-th to $2N$-th) are fully expanded while the following levels are sorted for resource optimization. The first processing level (level $2N$) has 4 LPBs corresponding to 4 possible values of $x_{2N,i}$ in vector $\widetilde{x}_{2N}$, whose entry being an element of the set $\boldsymbol{S}_r$ accompanied with initial radius values $r_{2N} = r_{sph}$. Each output of this level ($r_{(2N-1)i}$,$x_{(2N-1)i}$, $i = 1 \div 4$) will correspond to 4 possible values of $x_{(2N-1)i}$, therefore, the $(2N-1)$-th level has 16 LPBs. Similarly, we take the full expansion until level i, which requires 4^{2N-i+1} LPBs. In our implementation, the final full expansion level of 4×4 and 8×8 MIMO is 6 and 14, respectively. Note that outputs from each level are latched into pipeline registers to ensure an adequate system clock frequency. Furthermore, sorting is applied to prevent the excessively large number of LPBs. Specifically, after the i-th level, we use the NSB($(4^{2N-i+1}) \rightarrow K_i$) block to truncate ($4^{2N-i+1}$) nodes in the tree to select only K_i best nodes. This then requires $K_i \times 4 = 4K_i$ LPBs for processing at the $(i-1)$-th level. The $4K_i$ outputs of the $(i-1)$-th level are truncated to keep only K_{i-1} best nodes. This process continues until the lowest level, as illustrated in Fig. 11, where only one best full $\widetilde{\boldsymbol{x}}_1$ vector is taken.

As mentioned above, the LPB and NSB blocks support adjustable pipeline stages, so as the entire detector design. At fully pipelined, the whole 4×4 MIMO detector has 64 pipeline stages, where each LPB has 3 pipeline stages, NSB blocks have different pipeline stages depending on sorting complexity: 10 for NSB(64 – 8), 7 for NSB(32 – 8) and NSB(32 – 4), and 5 for NSB(16 – 4).

For this design, the 4×4 (8×8) MIMO detector takes 64 (176) clock cycles to decode the first symbol, from then on every clock the detector block completes decoding one symbol, the maximum throughput of the system then is defined by the maximum clock frequency.

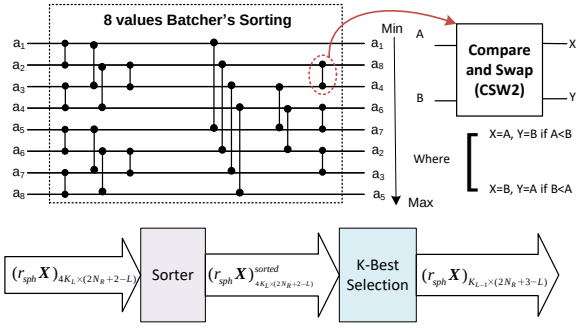

Figure 10. Detailed hardware implementation of an NSB.

6. Simulation and Implementation Results

6.1. Functional Verification

The whole design is shown in Fig. 13 has been implemented using VHDL and functionally verified using the Vivado simulator. We also implemented a corresponding MATLAB model to generate test data and make sure that the hardware model delivers the same function as expected. The MATLAB model simulates the whole MIMO channel to generate detector parameters, including $\boldsymbol{R}$, $\boldsymbol{Q}$ matrices, and input signal $\boldsymbol{y}$ vector making the corresponding output $\boldsymbol{x}$ vector. This set of data is then converted to an appropriate data format for logic simulations. Specifically, we generate 1 million data sets corresponding to 1 million transmitted signal vectors. Each element of the matrix $\boldsymbol{R}$ is represented by 15 signed bits. For the $\boldsymbol{Q}^H \boldsymbol{y}$ matrix, it is 17 signed bits. Each element of the solution vector $\boldsymbol{x}$ represents 2 bits corresponding to the four

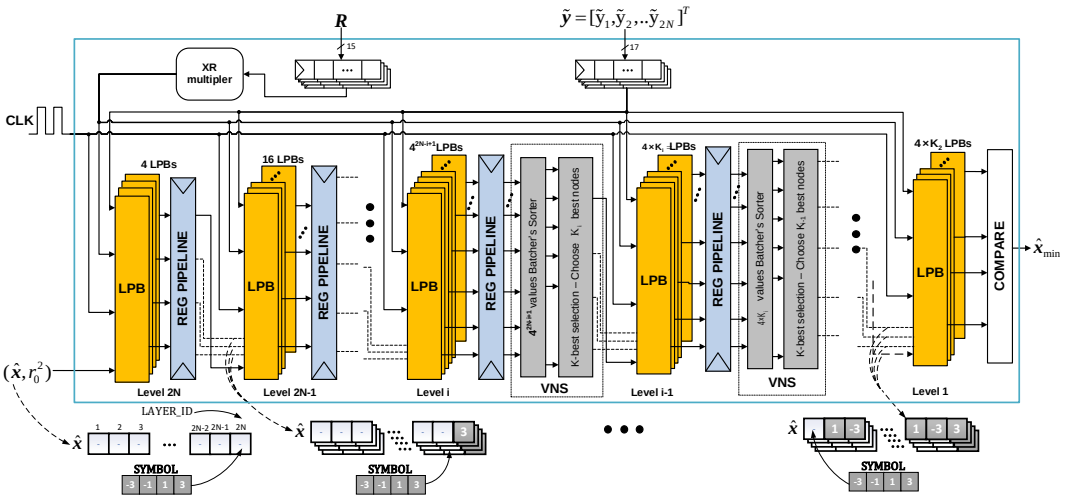

Figure 11. Structure of the hardware implementation of the proposed EHSD.

possible values of x_i. The decoded symbols are mapped back to the signal constellation to be converted to a bitstream. With 1 million symbols, so we have 16Mb bits transmitted.

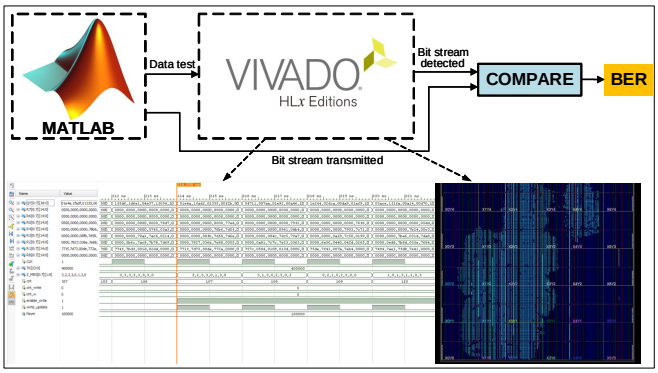

Figure 12. Verification platform of the EHSD using Vivado Simulator and MATLAB

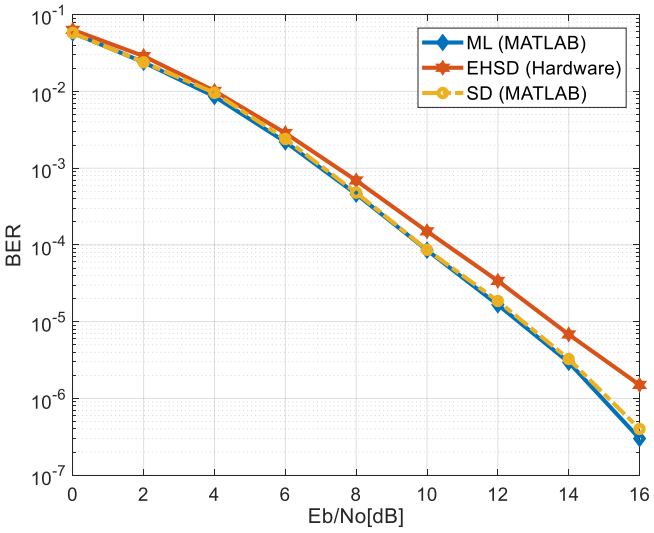

Figure 13. BER performance of the proposed hardware EHSD, ML and the conventional SD

After preliminary functional simulation, we compare the BER of the proposed EHSD block with the BER of the theoretical ML and the conventional SD decoding method obtained from the MATLAB simulation. The results in Fig. 13 show that the BER of the hardware-implemented EHSD closely matches the BER of ML and the conventional SD methods. The slight difference between the hardware model of EHSD and the conventional SD method results from limiting the number of valid nodes selected at each level as well as the precision of the fixed-point implementation used. This observation essential is a trade-off between accuracy and efficiency in terms of resources and throughput. We can always further improve the BER of the EHSD by increasing the K-best node selection. However, this would lead to an increase in resource utilization and possible throughput degradation. Nonetheless, considering hardware implementation, from the BER plots, it can be seen that EHSD exhibits similar BER compared to conventional SD while the computation workload is reduced significantly, as we have discussed in Section 4.

In order to evaluate the feasibility of the design, we synthesized the EHSD design on several Xilinx FPGA devices for determining resource utilization and the achievable maximum clock frequency and the corresponding throughput. The results are shown in Fig. 14, according to which the design can be fully implemented on some low-end and mid-end devices such as Artix7-XC7A200T, Kintex7-XC7K325. The number of LUTs that the design uses is about 69.7K LUTs accounting for 53% of the LUT resources

on the XC7A200T chip and only 9% on the Virtex7 UltraScale+ chip. The number of DSP blocks is 196, corresponding to 196 LPB blocks used as expected, which can be well satisfied by popular FPGA devices. The two main components that take up the most resources in this design are the LPB and NSB. All LPBs account for 30% of the LUTs that the design uses, while the NSB takes up 59%, almost double the design resources. Each LPB block is manually optimized separately for each decoding level, where at the most computationally complex levels, each LPB block costs 1 DSP and an average of 150 LUTs. The complexity of NSBs depends on the size of their Batcher sorting block. In this design, the 64-to-8 Batcher sorting block shows a high cost of $21.31K$, while 16-to-4 Batcher sorting occupies only $2.4K$ LUTs, i.e., almost 10 times lower than the most complex 64-to-8 sorting block but still much larger than the LUTs occupied by all LPBs.

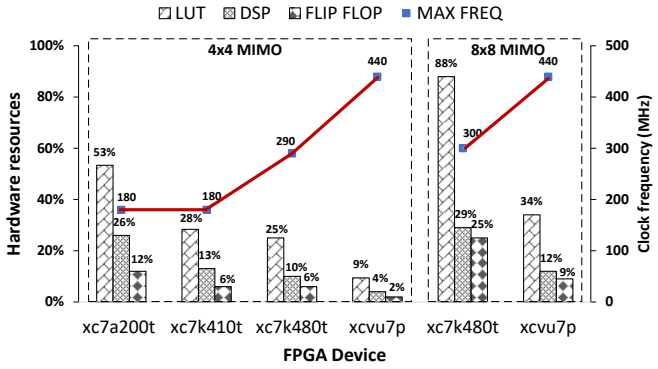

Figure 14. The EHSD resources utilization and maximum throughput frequency for (a) 4×4 and (b) 8×8 MIMO detectors.

Regarding the performance, the maximum clock frequency of the design on each chip is different due to the base technology used. With the Artix7-XC7A200T chip, the maximum clock frequency is only 180 MHz. With the high-end chip Virtex7-UltraScale+ XCVU7P, it can reach 440 MHz. The maximum clock frequency is also dependent on the pipeline architecture of the design. Maximum path delay is caused by the multiplier or adder in LPB and CSW2 blocks in NSB (see Fig. 10). It is possible to improve the maximum clock frequency of the design by adding pipeline registers at the expense of Flip-Flop (FF) resources, especially for the Batcher sorting block. For example, for the reported results, in Batcher sorting blocks for every two consecutive CSW2 blocks, we put a pipeline register, the total number of FFs used is $24.7K$, which is 72% of the total FFs used by the whole design. The total FF resources increased by 45% in the case of the deep-pipelined Batcher sorting block implementation (i.e., inserting the register for every swapping stage). The amount of FF reduces by 24% to a trade-off for approximately 50% degradation in the maximum frequency. The processing latency also depends on the number of pipeline stages. The EHSD design for the (4×4 MIMO) system requires 64 clock cycles, which correspond to 145ns (355ns) on XCVU7P (XC7A200T). The system throughput does not depend on the pipeline stage but only on the maximum frequency. For 4×4 MIMO with the 16-QAM symbol, the throughput (calculated as $f_{clk} * 16$) reaches 7.04Gbps (2.88Gbps) for XCVU7P (XC7A200T). We also have an 8×8 MIMO implemented on FPGA for evaluation. 8×8 MIMO has a much higher design complexity compared to 4×4 MIMO. It is shown that the design takes 88% LUT on XC7K480T and 34% on the high-end XCU7P. More details, the significant resource are utilized for NSB (55.6%) since the high inputs NSB (96-input-8-output) is used in this design. Regarding the performance, the processing latency is essentially larger than that of 4×4 MIMO with 400ns (i.e., 176 clock cycles). However, thanks to the deep-pipelined design, the maximum clock frequency is achievable at 440Mhz, correspondingly The throughput of the detector is almost double compared to 4×4 MIMO, as expected.

6.2. Comparison To The State-of-the-Art

In Table 2, we compared our implementation of the proposed 4×4 (8×8) MIMO with some of recent SD detectors implementations. The proposed algorithm has polynomial complexity as K-best, but the configuration has been optimized to achieve the best BER ratio with the least use of computational resources. Regarding the throughput, the maximum throughput of our proposed detectors is superior to that of other published works, even with ASIC implementation ones. Particularly, the throughput of the proposed algorithm for 4×4 is higher 13-14 times higher than the throughput offered by K-best implementations in [22] and [21]; for 8×8 MIMO, our EHSD shows 4 times throughput higher than recent ASIC implementations in [17] and [16]. As can be seen, the throughput of the EHSD with 8×8 configurations is achieved to surpass the upper limit of the 5G radio network communication (10 Gbps). The latency of EHSD is as small as about 1/19 (about 1/7) compared to the MBD-FD [17] and equals 1/5 (1/2) of LDLR [16].

LUT and DSP resources of EHSD take up less, and the maximum operating frequency is higher than that in [21, 22] thanks to the rigorously microarchitectural optimization. While most of the other 8×8 MIMO decoder requires ASIC implementation, we are the first to confirm that the design could be optimized to fit modern FPGA devices with fairly good performance. This result indicates the technological feasibility of FPGA for high throughput MIMO detection.

Table 2. Comparison of the proposed MIMO detector and the state-of-the-art.

Work	Proposed	[22]'21	[21]'17	Proposed	[17]'14	[16]'21
Antenna size	4×4	4×4	4×4	8×8	8×8	8×8
Modulation	16-QAM	16-QAM	16-QAM	16-QAM	64-QAM	64-QAM
Algorithm	Hybrid	K-best(R-BSFE)	K-best(BSS-EFE)	Hybrid	K-best(GR-LR)	K-best(LDLR)
Technology	FPGA	FPGA	FPGA	FPGA	ASIC (90nm)	ASIC(40nm)
LUT($\times 10^3$)	69.7	77.4	112.6	269.6	–	–
DSP	196	573	495	548	–	–
Flip Flop	33,999	–	–	144, 327	–	–
Gate (kG)	–	–	–	–	585	3, 837
Max Clock (MHz)	440	305	252	440	65	641
Latency	145.45ns	–	–	400ns	$2.88\mu s$	$710ns$
Throughput (Mbps)	7, 040	519.14	484.8	14, 080	3120	3, 846

7. Conclusion

In this work, we have proposed a generic design methodology to optimize the decoder for practical hardware implementation. Based on statistical analysis, we proposed an enhanced model of SD detection where a significant amount of calculations can be reduced without degrading the system BER. The model is specifically optimized for hardware implementation, which could deliver fixed latency and decent throughput at a reasonable expense of resources. The deep-pipelined version of the EHSD detector has been implemented for the 4×4 (8×8) MIMO system that achieves a throughput of 7.04 Gbps (14.08 Gbps), and latency of 145.45 ns (400 ns). Our implementation results also indicate that a reconfigurable hardware platform with its undeniable advantages in cost-efficiency and resource capability could be seriously considered for practical implementation of MIMO decoding and MIMO relay stations in modern wireless networks.

References

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

[2] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *on IEEE Transactions on Signal Processing*, vol. 53, pp. 2806-2818, Aug. 2005.

[3] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," *in Proc. Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. 2, p. 1051–1055, Nov. 2001.

[4] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima," *SIGSAM Bull.*, vol. 15, no. 1, pp. 37-44, 1981.

[5] U. Fincke, M. Pohst, "Improved methods for calculating vectors of short length," *Mathematics of Computation*, 1985.

[6] L. G. Barbero and J. S. Thompson,, "Fixing the complexity of the sphere decoder for MIMO detection,"*on IEEE Trans. Wireless Commun.*, vol. 7, pp. 2131-2142, 2008.

[7] L. G. Barbero, J. S. Thompson, "Extending a fixed-complexity sphere decoder to obtain likelihood information for Turbo-MIMO systems," *on IEEE Trans. Vehicular Tech.*, vol. 57, no. 5, pp. 2804-2814, 2008.

[8] Thompson, L. G. Barbero and J. S., "Rapid Prototyping of a Fixed Throughput Sphere Decoder for MIMO Systems," *in IEEE International Conference on Communications*, Istanbul, Turkey, June 2006.

[9] M. S. Khairy, M. M. Abdallah, S. E.-D. Habib, "Efficient FPGA implementation of MIMO detector for mobile WiMAX system," *in Proceedings of the IEEE ICC*, Dresden, Germany, 2009.

[10] B. Wu and G. Masera, "A Novel VLSI Architecture of Fixed-Complexity Sphere Decoder," *in 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, Lille, France, 2010.

[11] X. Nguyen, M. Le, N. Pham and V. Ngo, "A pipelined Schnorr-Euchner sphere decoder architecture for MIMO systems," *in 2015 International Conference on Advanced Technologies for Communications (ATC)*, Ho Chi Minh, Viet Nam, 2015.

[12] Xi Chen, Guanghui He, Member, IEEE, and Jun Ma, "VLSI Implementation of a High-Throughput Iterative Fixed-Complexity Sphere Decoder," *on IEEE Transactions on Circuits and Systems*, vol. 60, no. 5, pp. 272 - 276, 2013.

[13] B. Zheng, M. Wen, F. Chen, N. Huang, F. Ji and H. Yu,, "The K-Best Sphere Decoding for Soft Detection of Generalized Spatial Modulation," *on IEEE Transactions on Communications*, vol. 65, no. 11, pp. 4803-4816, 2017.

[14] S. Suh and J. R. Barry, "Reduced-Complexity MIMO Detection via a Slicing Breadth-First Tree Search," *in IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1782-1790, 2017.

[15] B. Halak, M. El-Hajjar, H. Ahmed, "Hardware Efficient Architecture for Element-Based Lattice Reduction Aided K-Best Detector for MIMO Systems," *on Journal of Sensor and Actuator Networks*, 2018.

[16] Z. Liang et al., "A 3.85-Gb/s 8 × 8 Soft-Output MIMO Detector With Lattice-Reduction-Aided Channel Preprocessing," *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 2, pp. 307-320, 2021.

[17] Liao, Chun-Fu and Wang, Jhong-Yu and Huang, Yuan-Hao, "A 3.1 Gb/s 8x8 Sorting Reduced K-Best Detector With Lattice Reduction and QR Decomposition," *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2675-2688, 2014.

[18] Ibrahim A, Bello, Basel Halak, Mohammed El-Hajjar, Mark Zwolinski, "VLSI Implementation of a Fully-Pipelined K-Best MIMO Detector with Successive Interference Cancellation," *in Circuits Systems and Signal Processing*, 2019.

[19] C. A. Shen and A. M. Eltawil, "An Adaptive Reduced Complexity K-Best Decoding Algorithm with Early Termination," *in Proc. IEEE CCNC*, Las Vegas, NV, USA, 2010.

[20] P. Tsai, W. Chen, X. Lin and M. Huang, "A 4×4 64-QAM reduced-complexity K-best MIMO detector up to 1.5Gbps," *in Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, Paris, France, 2010.

[21] Y. Wu and J. McAllister, "Bounded selective spanning with extended fast enumeration for MIMO-OFDM systems detection," *in IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, pp. 2556-2568, Sept. 2017.

[22] Y. Wu and J. McAllister, "Configurable Quasi-Optimal Sphere Decoding for Scalable MIMO Communications," *in IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2675-2687, 2021.

[23] H. Fang, L. Ge and G. Zhu, "An improved radius adaptive K-Best algorithm for MIMO system," *in IEEE International Conference on Progress in Informatics and Computing*, Shanghai, China, 562-566.

[24] M. Ouyang, "Sorting sixteen numbers," *in 2015 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-6, 2015.

[25] W. Min, "Analysis on Bubble Sort Algorithm Optimization," *in 2010 International Forum on Information Technology and Applications*, Kunming, China, 2010.

[26] Pedro Cervantes-Lozano, Luis F. González-Pérez, Andrés D. García-García, "Analysis of Parallel Sorting Algorithms in K-best Sphere-Decoder Architectures for MIMO Systems," *in 2011 International Conference on Reconfigurable Computing and FPGAs*, pp. 321-326, 2011.

[27] C. Liao, T. Wang and T. Chiueh, "A 74.8 mW Soft-Output Detector IC for 8 ×8 Spatial-Multiplexing MIMO Communications," *in IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 411-421, 2010.

[28] Meng-Yuan Huang and Pei-Yun Tsai,"Toward Multi-Gigabit Wireless: Design of High-Throughput MIMO Detectors With Hardware-Efficient Architecture," *on IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 613-624, Feb.2014.