Circuits Syst Signal Process (2016) 35:3644–3674
DOI 10.1007/s00034-015-0218-y

A Survey of VLSI Implementations of Tree Search Algorithms for MIMO Detection

Ibrahim A. Bello[1] · **Basel Halak**[1] · **Mohammed El-Hajjar**[1] · **Mark Zwolinski**[1]

Received: 10 March 2015 / Revised: 28 November 2015 / Accepted: 29 November 2015
Published online: 30 December 2015

Abstract Multiple-input multiple-output (MIMO) detection algorithms have received considerable research interest in recent years, as a result of the increasing need for high data-rate communications. Detection techniques range from the low-complexity linear detectors to the maximum likelihood detector, which scales exponentially with the number of transmit antennas. In between these two extremes are the tree search (TS) algorithms, such as the popular sphere decoder, which have emerged as attractive choices for implementing MIMO detection, due to their excellent performance-complexity trade-offs. In this paper, we survey some of the state-of-the-art VLSI implementations of TS algorithms and compare their results using various metrics such as the throughput and power consumption. We also present notable contributions that have been made in the last three decades in implementing TS algorithms for MIMO detection, especially with respect to achieving low-complexity, high-throughput designs. Finally, a number of design considerations and trade-offs for implementing MIMO detectors in hardware are presented.

Keywords MIMO detection algorithms · Sphere decoding · Survey · Wireless communications · Very large-scale integration (VLSI)

✉ Ibrahim A. Bello
iabello@hotmail.co.uk; iab1g12@ecs.soton.ac.uk

Basel Halak
bh9@ecs.soton.ac.uk

Mohammed El-Hajjar
meh@ecs.soton.ac.uk

Mark Zwolinski
mz@ecs.soton.ac.uk

1 Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

1 Introduction

Multiple-input multiple-output (MIMO) techniques are fast becoming vital components of modern communications, especially with respect to achieving higher transmission rates and improved reliability. Already, MIMO has found support in the 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) [1] and WiMAX [7] wireless standards, and it is expected that MIMO will play a prominent role in the upcoming 5th generation (5G) standard, which is expected to offer data rates of tens of gigabits per second [3]. As such, new approaches from both the theoretical and hardware perspectives are needed in order to meet the expected challenges.

Despite its many advantages however, one key challenge of MIMO technology is the complexity of the receiver, which is exacerbated due to the multiple interfering signals at each receive antenna. The maximum likelihood (ML) detector offers the best bit error rate (BER) signal detection; however, its exponential complexity makes it unsuitable for real-time hardware implementation. This problem has inspired a large body of research work in investigating low-complexity alternatives to the ML detector in recent years. In particular, the sphere decoder [53] and related tree search (TS) algorithms have attracted significant research interest due to their excellent performance-complexity trade-off and several VLSI implementations have been reported [9,15,33,45,58].

In this paper, we survey some of the state-of-the-art VLSI implementations of TS algorithms from the literature and compare their results using metrics such as the throughput and power consumption. We also highlight a number of notable contributions to tree search detection over the years as summarized in Table 1. A number of papers (e.g. [51] and [38]) have focused on the algorithmic aspects of MIMO detection—this paper fills the gap by focusing on the hardware implementation aspects of MIMO detection, which will hopefully lead to a better understanding of the implications of design decisions in a practical scenario.

The remainder of the paper is organized as follows. Section 2 provides an overview of the MIMO system model. In Sect. 3, we discuss some performance metrics for comparing VLSI implementations of MIMO detectors. In Sect. 4, the ML lattice search is introduced. In Sects. 5–8, we discuss different TS algorithms and their corresponding hardware implementations from the literature. A number of design considerations for MIMO detectors are presented in Sect. 9 and the paper is concluded in Sect. 10.

The following notations are used in the paper. $A_{i,j}$ represents the element of the matrix $\mathbf{A}$ at the i_{th} row and j_{th} column. $\mathcal{S}$ and $\mathcal{D}$ represent the complex and real constellations, respectively. $\mathbf{I}_m$ represents an $m \times m$ identity matrix. $(\mathbf{A})^T$ and $(\mathbf{A})^H$ represent the transpose and Hermitian transpose of $\mathbf{A}$, respectively. $\|\mathbf{a}\|$ represents the 2-norm of the vector $\mathbf{a}$. $\overline{x}$ represents the complement of the bit, x. N_0 represents the noise power spectral density.

2 MIMO System Model

Figure 1 shows a simplified block diagram of a MIMO system with N_T transmit antennas and N_R receive antennas. The information bits at the transmitter rep-

Table 1 Notable contributions in tree search MIMO detection

Year	Work	Contribution
1985	Fincke and Pohst [21]	Formulated the shortest vector search within a lattice which formed the basis for the sphere decoder (SD)
1986	Babai [6]	Formulated the nearby lattice point to the ML solution which significantly reduces the complexity of the SD
1993	Schnorr and Euchner [44]	Made modifications to the Fincke and Pohst algorithm by visiting the lattice points in accordance with their partial costs
1999	Viterbo and Boutros [53]	Proposed the SD algorithm based on the work of Fincke and Pohst [21]
2002	Wong et al. [58]	Proposed and implemented the K-best detector
2003	Wiesel et al. [56]	Proposed the use of a table lookup for determining the SE enumeration for the SD algorithm
	Hochwald and ten Brink [24]	Invented a "List" version of the SD for soft-output decoding
2005	Burg et al. [15]	Formulated norm approximations for reducing the complexity of the Euclidean distance calculation
2006	Wenk et al. [55]	Implemented a "single-cycle sort" K-best architecture that delivers the K-best nodes in one cycle
2007	Kang and Park [26]	Proposed a high-speed implementation of the Euclidean distance computation for the SD
2008	Studer et al. [49]	Formulated a single tree search (STS) for soft-output sphere decoding and presented its VLSI implementation
	Barbero and Thompson [10]	Formulated the FSD and presented its hardware implementation in [9]
	Shabany and Gulak [46]	Proposed a serial sorting strategy for the K-best detector
2009	Azzam and Ayanoglu [5]	Formulated an "orthogonal" real-valued decomposition (ORVD) of the channel matrix
	Liao et al. [30]	Implemented a low-complexity enumeration for soft-input SD
2010	C. Studer and H. Bolcskei [50]	Extended the STS formulated in [49] to iterative decoding
2012	Liu et al. [33]	Proposed a reliability-based tree extension for the FSD
2013	Kong and Park [29]	Proposed a technique for reducing the number of multipliers required in the Euclidean distance computation
	Romano et al. [43]	Proposed the "King" SD, which uses a set of "sufficient" conditions to reduce the total number of expanded nodes compared to the conventional SD
2014	Papa et al. [40]	Extended the King SD to incorporate a priori values for SISO decoding

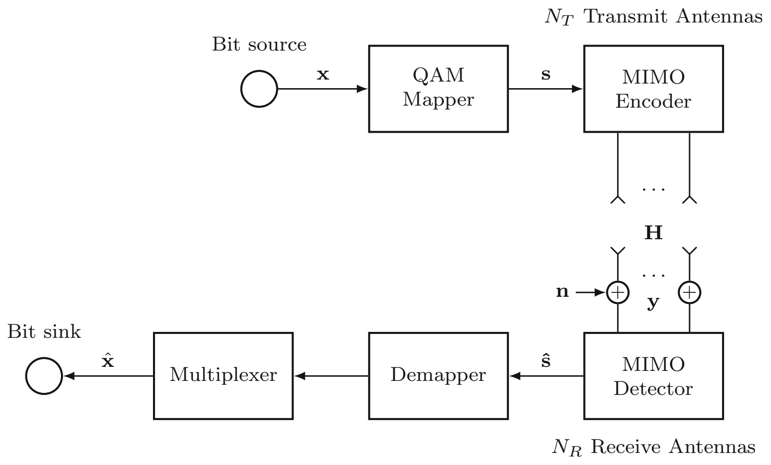

Fig. 1 Simplified MIMO system block diagram

resented by $\mathbf{x} = [\mathbf{x}^{<0>}, \mathbf{x}^{<1>}, \ldots, \mathbf{x}^{<N_T-1>}]$ are mapped to the symbol vector, $\mathbf{s} = [s^{<0>}, s^{<1>}, \ldots, s^{<N_T-1>}]$, using quadrature amplitude modulation (QAM) with Q bits per symbol. $\mathbf{x}^{<k>}$ represents the kth $1 \times Q$ bit vector, which is mapped to the kth symbol, $s^{<k>}$. A MIMO encoder then transmits the symbols spatially over N_T antennas using one of the several multiplexing/diversity techniques [20]. In this survey, we consider the case of spatial-multiplexing only, which aims to achieve the maximum data rate by transmitting independent substreams simultaneously over different antennas. Assuming a flat-fading channel, the received signal at the N_R receive antennas is given by

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{1}$$

where $\mathbf{y}$ is the $N_R \times 1$ received signal; $\mathbf{H}$ is a $N_R \times N_T$ matrix representing the MIMO channel and $\mathbf{n}$ is the $N_R \times 1$ vector of additive white Gaussian noise (AWGN). The channel matrix $\mathbf{H}$, consists of independent and identically distributed (i.i.d) complex-valued gains for each path from the transmitter to the receiver. In this paper, $\mathbf{H}$ is assumed to be square, that is, $N_T = N_R$.

At the receiver, a MIMO detector estimates the transmitted symbols using one of a broad range of detection schemes as illustrated in Fig. 2. A demapper converts the symbols to their binary equivalent and a multiplexer converts the detected parallel bit streams into a single bit stream to recover the transmitted bits. A channel decoder may also be concatenated with the MIMO detector in a technique known as iterative decoding [24] in order to improve the BER. In the next section, we present some performance metrics for comparing the hardware implementations of the MIMO detector.

3 Performance Metrics

In this paper, the results of the MIMO detector implementations are compared using the area, in kilo-gate equivalent (kGE), the throughput, hardware efficiency and the

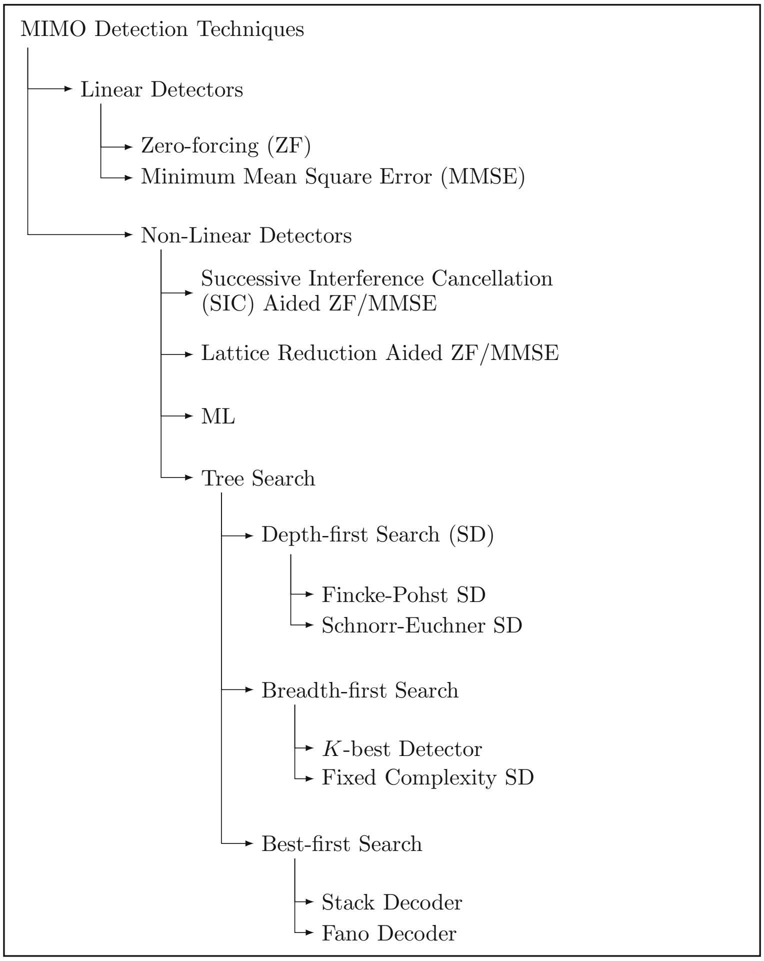

Fig. 2 Classification of MIMO detection algorithms

energy per bit, E_{bit}, which is computed as the ratio of the power consumption to the throughput and provides an indication of the energy efficiency of an implementation. The hardware efficiency of an implementation is determined as the ratio between the throughput and the area (TAR). The throughput and power consumption are normalized to the 65 nm technology at a supply voltage (V_{dd}) of 1.2 V to ensure a fair comparison. The throughput, Φ, is normalized as

$$\Phi_{\text{norm}} = (\text{Tech.}/65\text{nm}) \times \Phi,$$

while the power consumption is normalized as

$$\text{Power}_{\text{norm}} = \text{Power} \times \left(\frac{1.2\text{ V}}{\text{V}_{\text{dd}}}\right)^2 \times \left(\frac{65\text{nm}}{\text{Tech.}}\right).$$

Throughout this paper, references to the throughput and power consumption are made with respect to these definitions.

4 Maximum Likelihood Detection

The maximum likelihood detector (MLD) carries out an exhaustive search within the lattice, **Hs**, in order to find the closest point to the received vector **y**. In other words, it tries to solve for the symbol vector, **s**, that minimizes the Euclidean distance as follows:

$$\mathbf{s}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\mathcal{S}^{N_T}} \|\mathbf{y} - \mathbf{Hs}\|^2 ,$$

where $\mathcal{S}^{N_T}$ is an N_T dimensional lattice with complex entries formed from all possible combinations of the $N_T \times 1$ transmitted symbols.

The total number of points searched and compared in the exhaustive search is $|\mathcal{S}|^{N_T}$, which implies that the complexity of the MLD scales exponentially with the number of transmit antennas. Thus, for $N_T = 4$ and using 64-QAM, the MLD needs to explore a total of 64^4 possible solutions; by contrast, the same detector would need to explore only 2^4 possible solutions if using binary phase-shift keying (BPSK). The computational cost of the ML search makes it impractical for hardware implementation; however, it offers the best BER performance in an uncoded scenario. In the next section, the sphere decoder is introduced, which achieves the ML performance at a significantly reduced complexity.

5 The Sphere Decoder

5.1 Algorithm Description

The sphere decoder (SD) reduces the complexity of the MLD by considering only those lattice points that fall within the "sphere" bounded by r^2 as illustrated in Fig. 3. Mathematically, the lattice points that are considered are those whose Euclidean distances, $d(\mathbf{s})$, satisfy the relation

$$d(\mathbf{s}) = \|\mathbf{y} - \mathbf{Hs}\|^2 \leq r^2, \tag{2}$$

where the minimum metric solution is the SD output. A QR decomposition is usually performed on **H** to transform the Euclidean distance to

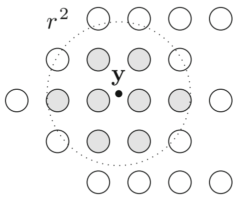

Fig. 3 Basic concept of sphere decoding

$$d(\mathbf{s}) = \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2, \tag{3}$$

with $\hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$. Due to the upper triangular property of $\mathbf{R}$, the QR decomposition of the channel matrix transforms the lattice search into an equivalent tree search, where each level of the tree corresponds to a transmit antenna.

Figure 4 illustrates the SD tree search, where the numbers within the nodes indicate a possible traversal of the tree. The tree is traversed using a depth-first search (DFS), where one node is expanded in each level before descending to lower levels. If the cumulative metric of a path is greater than r^2, then the path is pruned, which is indicated by the dotted lines in the figure.

For the purpose of hardware implementation, it is also useful to perform a real-valued decomposition (RVD) of $\mathbf{H}$, which simplifies the computation of the Euclidean distance [55]. The RVD decouples the channel equation in (1) into a new real-valued representation as follows:

$$\begin{bmatrix} \Re\{\mathbf{y}\} \\ \Im\{\mathbf{y}\} \end{bmatrix} = \begin{bmatrix} \Re\{\mathbf{H}\} & -\Im\{\mathbf{H}\} \\ \Im\{\mathbf{H}\} & \Re\{\mathbf{H}\} \end{bmatrix} \begin{bmatrix} \Re\{\mathbf{s}\} \\ \Im\{\mathbf{s}\} \end{bmatrix} + \begin{bmatrix} \Re\{\mathbf{n}\} \\ \Im\{\mathbf{n}\} \end{bmatrix},$$

where $\Re\{.\}$ and $\Im\{.\}$ denote the real and imaginary parts of a complex number, respectively. The complex constellation $\mathcal{S}$ is transformed into a *real* constellation, $\mathcal{D}$, which has odd-valued integer symbol entries defined as: $\{-\sqrt{M}+1, \ldots, \sqrt{M}-1\}$, where M is the modulation order. The M-ary tree is also converted to an equivalent tree with $2N_T$ levels and $\sqrt{M}$ children per parent node.

5.2 Partial Euclidean Distance Computation

The sequential nature of the DFS ensures that the Euclidean distance cannot be calculated at once; instead, it must be computed incrementally as the detector progresses deeper into the tree. The Euclidean distance up to any level in the tree is thus known as the partial Euclidean distance (PED). By traversing the tree from level $i = 2N_T$ to $i = 1$, the PED T_i, up to the i_{th} level is given as

$$T_i = T_{i+1} + |e_i|^2\,, \tag{4}$$

where $|e_i|^2$ is the PED increment at the i_{th} level and is given by

$$|e_i|^2 = \left|b_i - R_{i,i}s_i\right|^2, \tag{5}$$

where s_i represents a symbol at the i_{th} level and b_i is defined as

$$b_i = \hat{y}_i - \sum_{j=i+1}^{2N_T} R_{i,j}s_j. \tag{6}$$

The use of the real-valued sphere detection greatly simplifies the PED computation in hardware. For example, by setting $\Im\{e_i\} = 0$ in the real model, the multiplication

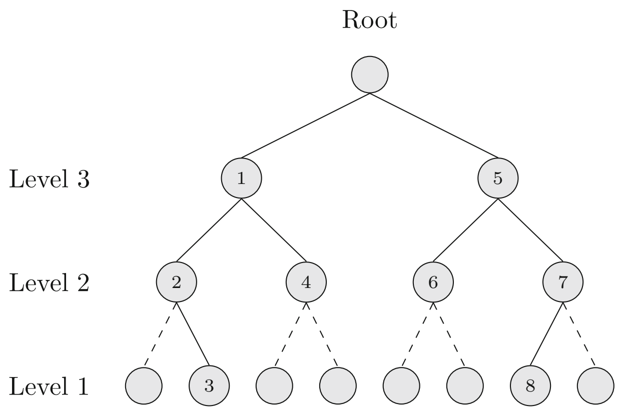

Fig. 4 Depth-first Tree Traversal using BPSK and $N_T = 3$

required in computing (4) is much simpler compared with the complex sphere detection [55]. Overall, the PED computation using the complex model is approximately three times more complex than using the real implementation [39]. However, the complex sphere detection is capable of achieving a higher throughput due to the fewer number of tree levels that needs to be processed.

5.3 Orthogonal Real-Valued Decomposition

A different real-valued channel decomposition is presented by Azzam and Ayanoglu [4], where the complex channel matrix is decomposed as follows:

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \Re\{H_{1,1}\} & -\Im\{H_{1,1}\} & \dots & \Re\{H_{1,N_T}\} & -\Im\{H_{1,N_T}\} \\ \Im\{H_{1,1}\} & \Re\{H_{1,1}\} & \dots & \Im\{H_{1,N_T}\} & \Re\{H_{1,N_T}\} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Re\{H_{N_T,1}\} & -\Im\{H_{N_T,1}\} & \dots & \Re\{H_{N_T,N_T}\} & -\Im\{H_{N_T,N_T}\} \\ \Im\{H_{N_T,1}\} & \Re\{H_{N_T,1}\} & \dots & \Im\{H_{N_T,N_T}\} & \Re\{H_{N_T,N_T}\} \end{bmatrix}. \tag{7}$$

This new channel representation has the property that adjacent columns (i.e. $\widetilde{H}_n$ and $\widetilde{H}_{n+1}$) are orthogonal to each other, that is, $\widetilde{H}_n \cdot \widetilde{H}_{n+1}^T = 0$ for odd values of n $(1, 3, \dots, 2N_T - 1)$. It can also be shown that the QR decomposition of the modified channel matrix results in $\widetilde{R}_{i,i+1} = 0$ for all odd values of i. Thus, setting $\widetilde{R}_{i,i+1}s_{i+1} = 0$, the computation of (6) at the i_{th} level is modified to

$$\begin{aligned} \widetilde{b}_i &= \hat{y}_i - \sum_{j=i+1}^{2N_T} \widetilde{R}_{i,j} s_j \\ &= \hat{y}_i - \sum_{j=i+2}^{2N_T} \widetilde{R}_{i,j} s_j. \end{aligned}$$

Meanwhile, $\widetilde{b}_{i+1}$ for the computation of the PED in the $(i+1)_{th}$ level is computed normally as

$$\widetilde{b}_{i+1} = \hat{y}_{i+1} - \sum_{j=i+2}^{2N_T} \widetilde{R}_{i+1,j} s_j .$$

Thus, $|\widetilde{e}_i|^2$ for odd-valued levels can be computed concurrently with $|\widetilde{e}_{i+1}|^2$ since $\widetilde{b}_i$ no longer depends on the previously detected symbol, s_{i+1}. The PED up to the i_{th} level is then computed as

$$T_i = T_{i+2} + \left|\widetilde{b}_{i+1} - \widetilde{R}_{i+1,i+1} s_{i+1}\right|^2 + \left|\widetilde{b}_i - \widetilde{R}_{i,i} s_i\right|^2 .$$

This result allows two adjacent levels to be processed concurrently, which can allow higher throughputs to be achieved.

5.4 Schnorr Euchner Lattice Search

The original SD algorithm [53] does not follow a particular order when visiting the nodes at a given level. This can be inefficient, as the SD might spend too much time traversing non-promising paths. The Schnorr Euchner (SE) search [44] modifies the SD algorithm by visiting the nodes according to their path metrics, which enables the solution to be reached more quickly.

If $R_{i,i}$ is factored out in (5), the PED increment at the ith level can be expressed alternatively as

$$|e_i|^2 = \left|R_{i,i}\left(c_i - s_i\right)\right|^2 , \tag{8}$$

where $c_i = b_i / R_{i,i}$ and is referred to as the SE "centre". The magnitude of the PED increment is directly proportional to the distance of the symbol from the SE centre. The PED does not have to be computed explicitly in order to determine the SE enumeration; instead, a zigzag search can be carried out by iteratively determining the closest constellation points to the SE centre [14]. The SE search also begins with an infinity radius, which is updated any time a solution with a smaller metric is found. This eliminates the problem of detection failure (where no point falls within the sphere), and the complexity required in estimating the starting radius.

5.5 Soft-Input Soft-Output Sphere Decoding

In a practical system, the information bits are typically encoded using error correction codes in order to improve reliability of the data transmission. This requires a channel decoder for decoding the bits at the receiver, as well as a MIMO detector that is capable of generating soft information regarding the reliability of each detected bit. The channel decoder and the soft-input soft-output (SISO) detector are concatenated in an iterative arrangement as shown in Fig. 5.

In each iteration, the SISO detector computes the probability, $L_{i,b}^{D}$, that the bth bit of the ith symbol in the output is a 1 or 0, given a channel observation, $\mathbf{y}$. The a priori

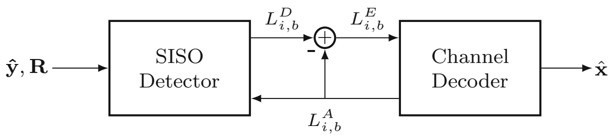

Fig. 5 Block diagram of iterative MIMO detection

reliability information computed by the channel decoder, $L^A_{i,b}$, in the previous iteration is fed back to the SISO detector to derive the new extrinsic information, $L^E_{i,b}$, which in turn is fed to the channel decoder. $L^D_{i,b}$ is expressed as a log likelihood ratio (LLR) and can be computed as [50]

$$\begin{aligned} L^D_{i,b} \triangleq & \min_{\mathbf{s}\in\mathcal{X}^0_{i,b}} \left\{ \frac{1}{N_0} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 - \log P[\mathbf{s}] \right\} \\ & - \min_{\mathbf{s}\in\mathcal{X}^1_{i,b}} \left\{ \frac{1}{N_0} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 - \log P[\mathbf{s}] \right\}, \end{aligned} \tag{9}$$

where $\mathcal{X}^0_{i,b}$ and $\mathcal{X}^1_{i,b}$ are the sets of symbol vectors with the bth bit equal to 0 and 1, respectively, and $P[\mathbf{s}]$ is the a priori reliability information computed by the channel decoder. Computing (9) for every bit is computationally expensive, and the SD can be applied to reduce the complexity by considering only those $\mathbf{s}$ for which (2) is small [24]. These solutions are stored in a candidate list and the decoder computes the extrinsic information only for the solutions within that list. As such, the algorithm is referred to as the "list" sphere decoder (LSD).

A different strategy from the LSD is the single tree search (STS) proposed by C. Studer and H. Bolcskei [50], which does not require a list for storing the possible solutions. The algorithm computes the maximum *a posteriori* probability (MAP) solution, $\mathbf{s}^{\text{MAP}}$, and its bit-wise counter-hypotheses concurrently, in a single tree search. The MAP solution is given as

$$\mathbf{s}^{\text{MAP}} = \arg\min_{\mathbf{s}\in\mathcal{S}^{N_T}} \left\{ \frac{1}{N_0} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 - \log P[\mathbf{s}] \right\}$$

and its corresponding reliability λ^{MAP} is computed as

$$\lambda^{\text{MAP}} = \frac{1}{N_0} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}^{\text{MAP}}\|^2 - \log P[\mathbf{s}^{\text{MAP}}].$$

One of the two minima in (9) corresponds to the MAP solution, as such, $L^D_{i,b}$ can be computed by determining $\mathbf{s}^{\text{MAP}}$, λ^{MAP} and its bit-wise counter-hypotheses $\lambda^{\overline{\text{MAP}}}$, which is computed as

$$\lambda^{\overline{\text{MAP}}} = \min_{\mathbf{s}\in\mathcal{X}_{i,b}^{\overline{\text{MAP}}}} \left\{ \frac{1}{N_0} \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 - \log P[\mathbf{s}] \right\},$$

where $\mathcal{X}_{i,b}^{\overline{\text{MAP}}} = \mathcal{X}_{i,b}^{\overline{x_{i,b}^{\text{MAP}}}}$. The STS-SD employs an efficient tree search strategy, where a node is traversed only once, which is achieved by descending into a sub-tree only if it would lead to an update to either λ^{MAP} or $\lambda^{\overline{\text{MAP}}}$.

The inclusion of the a priori information in the STS also modifies the SE enumeration and the geometric properties of $\mathcal{S}$ can no longer be directly applied to determine the node with the smallest metric as described in Sect. 5.4. In this case, the metric of a node, $\mathcal{M}_P(s_i)$, comprises of two separate components: the channel-based PED denoted by $\mathcal{M}_C(s_i)$ and the a priori based metric, $\mathcal{M}_A(s_i)$, which is computed as

$$\mathcal{M}_A(s_i) = -\log P[s_i] \approx \sum_{b=1}^{Q} \frac{1}{2}(|L_{i,b}^A| - x_{i,b}L_{i,b}^A),$$

for $\left|L_{i,b}^A\right| > 2$ [11]. In [30], a hybrid enumeration is proposed, where two candidates (based on $\mathcal{M}_C$ and $\mathcal{M}_A$, respectively) are selected in each iteration, and the node with the smaller metric is selected for the next visit.

Due to the inclusion of the a priori information in the tree search, the STS achieves a better performance than the LSD, which only considers candidates around the ML solution for computing the extrinsic information. It also requires less area than the LSD as it does not require a candidate list. A more in-depth discussion on the STS is provided in [49,50].

5.6 Hardware Implementations

Burg et al. [15] presented two VLSI implementations of the SD, based on the complex model of the channel matrix. In the first implementation (ASIC I), the PED is computed using the ℓ^2-norm as provided in (4), while in the second implementation (ASIC II), the PED is computed using an ℓ^∞-norm approximation given as follows:

$$T_i \approx \max(T_{i+1}, |e_i|). \tag{10}$$

By using the ℓ^∞-norm, ASIC II is able to achieve area savings of about 50 % compared to ASIC I due to the elimination of the squaring term in (4). However, the ℓ^∞-norm-based computation incurs a performance cost of 1.4 dB at high SNR values compared to ASIC I, which practically achieves the ML performance.

Both ASIC I and ASIC II are based on a serial *one-node-per-cycle* (ONPC) architecture which is illustrated in Fig. 6. The processing element (PE) consists of a metric computation unit, which computes the PED, and an enumeration unit, which determines the next node to visit according to the SE ordering. For a 4×4, 16-QAM system,

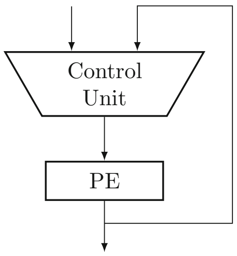

Fig. 6 Serial architecture of the DFS

ASIC II achieves a normalized throughput of 650 Mbps at a reference SNR of 20 dB, while ASIC I achieves a normalized throughput of 281 Mbps.

Borlenghi et al. [12] implemented the first STS detector based on the ONPC architecture described by Burg et al. [15]. Using a convolutional channel code with $1/2$ code rate, the STS detector achieves an SNR gain of about 5 dB compared to the hard-output SD for a target BER of 10^{-2} using 2 iterations. The implementation has 3 cores for 4-QAM, 16-QAM and 64-QAM. The 64-QAM implementation achieves a throughput of 132.9 Mbps using 2 iterations at an SNR of 24 dB, while it is capable of achieving a maximum throughput of more than 1 Gbps.

Yang et al. [59] adopt the use of a *table enumeration* [56], which stores precomputed SE orderings in a lookup table. In this method, the 16-QAM complex plane is divided into 64 sub-regions and the ordering is determined based on the location of the SE centre as illustrated in Fig. 7. Due to the symmetry of the complex plane, only the orderings based on the first quadrant need to be stored in memory. The implementation achieves a throughput of up to 231 Mbps at high SNR.

Jenkal and Davis [25] implemented a deeply pipelined detector that is capable of processing multiple received signals in order to achieve a higher throughput. Each independent received signal is assigned a separate memory unit for storing the surviving nodes; however, the received signals share the same computation resources in a time-multiplexed arrangement. The implementation achieves a throughput of 443 Mbps at 24 dB with an area consumption of 175 kGE.

Table 2 provides the relevant results of the VLSI implementations of the SD. The implementation of Jenkal and Davis [25] reports the largest area consumption among the hard-output detectors due to the additional memory required for processing the multiple received signals. The implementation of Borlenghi et al. [12] expectedly achieves the highest E_{bit} and the largest area due to the support for iterative decoding.

The throughput of the SD is variable, which necessitates additional I/O buffers in a practical system [23]. An *early termination* strategy, which stops the search after a given number of iterations, can be used to keep the complexity of the DFS bounded, although this may have a detrimental effect on the BER [13]. Multiple cores can also be employed in order to improve the attainable throughput of the SD [54]. In the next section, we discuss the K-best detector, which offers better opportunities for pipelining than the SD.

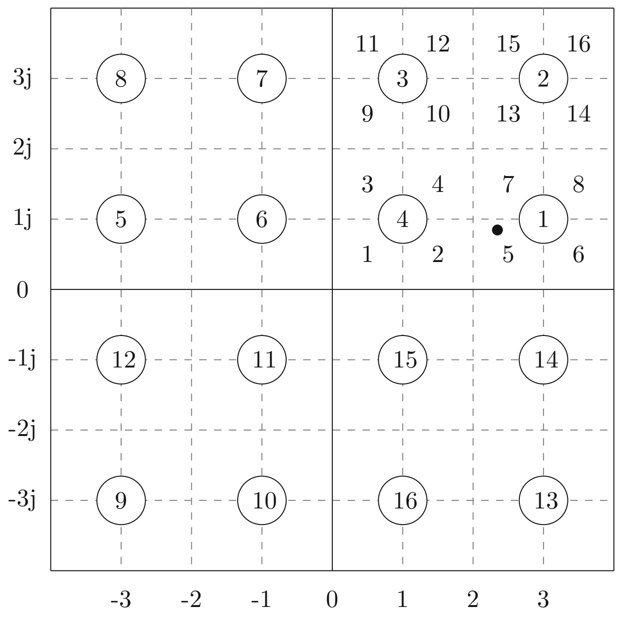

Fig. 7 Table enumeration for 16-QAM. Here, the SE centre indicated by the *shaded circle* falls in region 5 and the first symbol in the enumeration is $3 + 1j$

6 The K-best Detector

6.1 Algorithm Description

The tree search described in Sect. 5 can also be carried out in a breadth-wise manner, that is, by expanding all the nodes at a given level before descending to the next level. In Fig. 8, a possible sequence of visited nodes in a breadth-first tree search using a complex channel model is illustrated. At the end of the tree search, the detector compares the metrics of paths $\{1, 3, 5\}$ and $\{2, 4, 6\}$ and the path with the smaller metric is presented as the solution. One attractive feature of the breadth-first search (BFS) is its better support for parallelism and pipelining, due to its *forward-only* tree search. The implication, however, is that the BFS requires more memory than the DFS since more intermediate results need to be stored per level. On the other hand, the DFS expands only one node per level and the same processing unit can be reused over several cycles.

The most popular implementation of the BFS is the K-best detector, which expands a fixed number of K nodes at each level. The fixed number of expanded nodes allows the K-best to have a highly pipelined architecture, such that the output of a previous stage can serve as input to the subsequent stage with memory elements inserted in between stages for storing intermediate results. To get the "best" K nodes at a level, the K-best detector expands the children of each of the K parent nodes from the previous level and passes them to a sorting unit which sorts the candidates in ascending order with respect to their PEDs.

Table 2 Comparison of SD VLSI implementations for 4 × 4 MIMO

Work	Modulation	f_{CLK} (MHz)	Tech. (nm)	V_{dd} (V)	Φ_{norm} (Mbps)	Area (kGE)	Power$_{\mathrm{norm}}$ (mW)	E_{bit} (pJ/bit)	TAR (Mbps/kGE)	Ref. SNR (dB)
Burg [15, ASICI]	16-QAM	51	250	N/A	281	117	–	–	2.4	20
Burg [15, ASICII]	16-QAM	71	250	N/A	650	50	–	–	13	20
Yang [59]	64-QAM	108.7	90	–	231	153.93	–	–	1.5	32
Jenkal [25]	16-QAM	128	180	1.8	443	175	65.3	147.4	2.5	24
Borlenghi [12][a]	64-QAM	193	90	1	132.9	212	91.1	685.4	5.04	24

[a] Soft-input soft-output detector

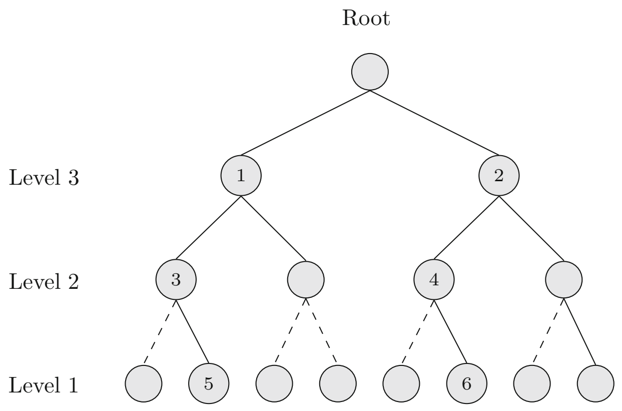

Fig. 8 Breadth-first Tree Traversal using BPSK and $N_T = 3$

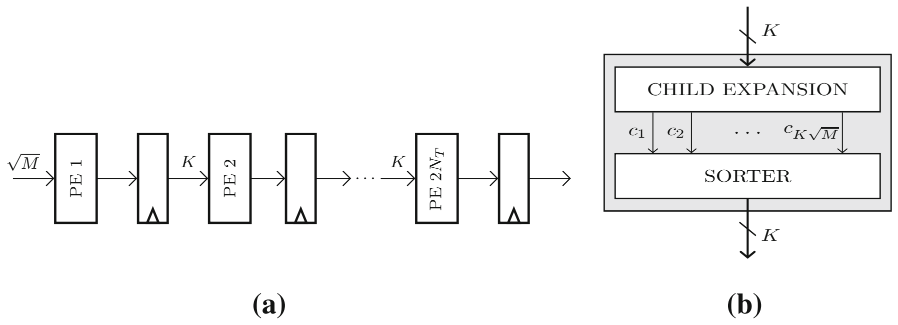

Fig. 9 Architecture of the K-best detector. **a** K-best detector pipeline. **b** Block diagram of a single PE

Figure 9 shows a simplified architecture for the K-best detector, where each pipeline stage corresponds to a level of the tree. Apart from PE 1 and PE 2, the architecture for each individual PE is basically the same for every level. PE 1 corresponds to the topmost level of the tree and has $\sqrt{M}$ inputs corresponding with the children of the root node, that is, the constellation set. PE 2 corresponds to the second level of the tree, and it selects the best K nodes out of a total of $\sqrt{M} \times \sqrt{M}$ candidates. In the remaining levels, the K best nodes are selected out of a total of $K\sqrt{M}$ candidates after a sorting operation. When $\sqrt{M} < K$, no sorter is required in PE 1, since all the nodes would need to be expanded in that case.

An attractive feature of the K-best detector, and other breadth-first schemes, is its suitability to soft-output generation. The competing solutions, which are computed concurrently with the hard-detection output (and discarded in the last level), can be used as the candidate list in the soft-output detection [23]. However, unlike the SD, the K-best detector does not guarantee that the ML solution will be found as the fixed number of extended nodes per level might exclude the actual ML solution.

6.2 Hardware Implementations

The first VLSI implementation of the K-best detector was presented by Wong et al. [58]. In this implementation, a radius constraint is computed by a ZF detector in a pre-detection stage and is used along with the K parameter in carrying out the tree pruning. Once the PED of any of the K candidate nodes exceeds the Euclidean distance between the ZF detector output and the received signal, the node is discarded subsequently from the search. The implementation achieves a relatively modest throughput of 53.8 Mbps.

Guo and Nilsson [23] implemented a similar detector to that of Wong et al. [58] and achieve an improved throughput of 287 Mbps by using a smaller K value of 5 and other hardware-level optimizations. Certain PED operations are relegated to the preprocessing stage, which reduces the complexity of the detector unit compared to that of Wong et al. [58].

Both Wong et al. [58] and Guo and Nilsson [23] employ a bubble-sort unit, which requires several cycles to select the best K nodes. Wenk et al. [55] implemented a single-cycle list merge, which merges the partially sorted children of each parent node, into one sorted list in a single step. The SE enumeration of the child nodes, for each parent, is determined in a zigzag manner based on the position of b_i on the real axis. The implementation achieves a throughput of 1.63 Gbps.

Shabany and Gulak [46] proposed a serial sorting scheme that provides a compromise between the high latency bubble-sort unit of Wong et al. [58] and the low latency, large area single-cycle merge of Wenk et al. [55]. This implementation is able to deliver K best nodes out of $K\sqrt{M}$ candidates in only K cycles. In the conventional K-best algorithm [58], all the children of the parent nodes from the previous level are expanded in parallel and sent to a bubble-sort unit. In this implementation, only the minimum metric child of each parent node is expanded in a cycle and the minimum amongst them is declared as the first node in the K-best list. Once a candidate is selected as part of the list, it is replaced by its next best sibling in the next cycle and the process is continued until all the K-best candidates are obtained.

The serial sorting procedure is illustrated in Fig. 10. In the first cycle, the minimum-metric child of p_1 ($c_{1,1}$) is compared against the minimum metric child of p_2 ($c_{2,1}$). In this case, $c_{1,1}$ is the winner and is disregarded in the next cycle and replaced by its next best sibling (i.e. $c_{1,2}$). The attractive feature of this method is that the number of cycles required to produce the K-best nodes depends only on the K value and is constant irrespective of the constellation size that is employed.

A hardware implementation of the serial K-best detector is presented in [45] and it is able to achieve a throughput of 655 Mbps. The implementation is extended in [41] to support soft-output generation by using the generated K best paths at the end of the detection to compute the LLR values. To improve the BER, selected discarded paths are also included in generating the soft outputs using ZF augmentation, which extends partial paths to full length by rounding them to the nearest constellation point [11,23]. By using a convolutional turbo encoder (code rate = 1/2), the soft-output implementation is able to achieve an SNR gain of 2.9 dB at a BER of 10^{-3} compared to the hard-output detector.

Kim and Park [28] implemented a K-best detector based on the ORVD channel model [4], which allows adjacent levels to be processed simultaneously in a pipeline

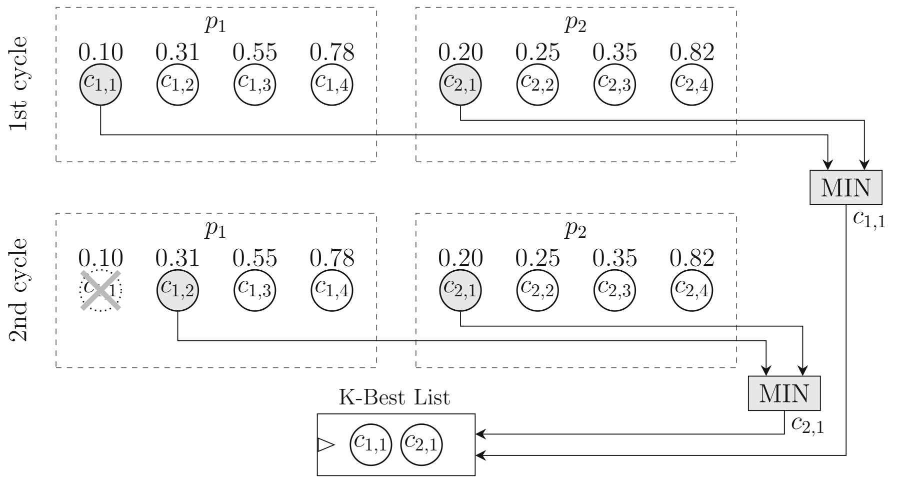

Fig. 10 Serial K-best sorting [46] for 16-QAM system with $K = 2$ showing two parent nodes p_1 and p_2. The respective PEDs are shown above each of the child nodes

stage. The total number of pipeline stages in their implementation is reduced from 8 to 3, which leads to a small area consumption. Another contribution of this work is the use of an approximate sorting scheme, where only a subset of the children of the parent nodes is considered for the sorting, which is carried out in a distributed fashion.

Two VLSI implementations of the K-best are presented. The first implementation (KB-I) consists of a single K-best detector core, while the second implementation (KB-II) consists of 4 detector cores that are interleaved in order to increase the throughput. The single-core detector is able to achieve a throughput of 404 Mbps, while the multi-core implementation improves the throughput by a factor of 4, with a corresponding increase in the area.

Liu et al. [32] implemented a configurable K-best architecture that is able to support different number of antennas (2×2 up to 4×4) and modulation schemes (quadrature phase-shift keying (QPSK) up to 64-QAM). In this implementation, an "extension number" is formulated to determine the number of nodes that are to be extended from each parent node at a given level. More nodes are expanded from the more reliable parent nodes (i.e. nodes with smaller metrics) than from the less reliable nodes. A candidate generation unit calculates all the possible values of $R_{i,j}s_j$ in (6) and makes them available to all the candidates at a given level. In a block-fading channel, the computation of the $R_{i,j}s_j$ values can also be pushed back to the preprocessing unit and performed once per frame, which leads to further energy savings.

All the previously discussed implementations are based on a multi-stage architecture, where a PE is assigned to each level of the tree. Moezzi-Madani et al. [35] implemented a single-stage architecture where a single PE is used for all levels in a folded arrangement similar to the DFS. A single-stage architecture is attractive for an application requiring moderate throughputs and where area is premium. Similar to the DFS, a higher throughput can be achieved by employing more than one detector core to operate in parallel on independent received symbol vectors. Their implementation supports antenna configurations of 2×2 up to 4×4, and a single core is able to achieve a throughput of 480 Mbps.

Table 3 provides the relevant results for the various VLSI implementations of the K-best detector. The implementation of Wenk et al. [55] (SC1) achieves the highest TAR figure, which is achieved as a result of the single-cycle sort mechanism employed. Meanwhile, the implementation of Kim and Park [28] achieves the smallest E_{bit} as a result of the relaxed sorting operation that is adopted.

6.3 Non-constant K-best Detector

The K value contributes significantly to the complexity of the K-best detector. However, a lower complexity can be achieved by using smaller K values at lower levels, without significantly affecting the BER performance.

Moezzi-Madani and Davis [34] implemented a modified K-best algorithm, which uses a non-constant K value that is decreased gradually at the lower levels. The implementation is able to save on area by up to 20 % while incurring a loss of 0.03 dB at an SNR of 20 dB compared to the conventional implementation. Another contribution of this work is a novel parallel merge algorithm (PMA) that is able to merge two sorted lists in one step, which makes it suited to high-throughput applications. However, the area cost of the PMA is relatively high with a complexity of $O(n^2)$ [36]. The throughput of the PMA-based detector is 540 Mbps with an area consumption of 131 kGE.

Tsai et al. [52] implemented a non-constant K-best detector utilizing the serial sorting proposed by Shabany and Gulak [46]. Like Yang et al. [59], this is another work that implements the SE enumeration using a table lookup; however, a real constellation is considered in this case. Instead of deciding the SE ordering by finding the nearest constellation points to the SE centre (which requires a divide operation), this implementation decides the SE ordering through (6) by finding the closest $R_{i,i}s_i$ to b_i. The enumeration module in this implementation consists only of 3 adders/subtractors and one small table that is pre-calculated.

The results of the non-constant K-best detectors are presented in Table 4. Not surprisingly, the implementations report much better TAR figures compared with the conventional K-best detectors. The implementation of Tsai et al. [52] also reports an E_{bit} figure that is smaller than that of any of the constant K-best implementations.

7 The Fixed-Complexity Sphere Decoder

7.1 Algorithm Description

The *fixed-complexity* sphere decoder (FSD) [8] is similar to the non-constant K-best detector as it expands a variable number of nodes from each level in its breadth-first detection. The FSD assigns a “node distribution” to the tree search, which determines the number of children that are extended from each parent node at each level. Typically, the FSD carries out an ML search at the topmost layer, that is $n_{N_T} = M$, where n_i is the number of nodes expanded in the i_{th} layer. The ML search at the topmost level

Table 3 Comparison of K-best VLSI implementations for 4×4 MIMO

Work	Modulation	K	f_{CLK} (MHz)	Tech. (nm)	V_{dd} (V)	Φ_{norm} (Mbps)	Area (kGE)	Power$_{norm}$ (mW)	E_{bit} (pJ/bit)	TAR (Mbps/kGE)
Wong [58]	16-QAM	10	100	350	N/A	53.8	52	-	-	1.04
Wenk [55, SC1]	16-QAM	5	132	250	N/A	1631	68	-	-	23.98
Guo [23]	16-QAM	5	100	350	2.8	287	91	21.4	74.4	3.15
Shabany [45]	64-QAM	10	282	130	1.3	1310	114	55.8	42.6	11.49
Patel [41][a]	64-QAM	10	833	65	1.3	2000	298	238.6	119.3	6.71
Kim [28, KB-I]	16-QAM	16	146	180	1.8	404	32.2	10.3	25.4	12.56
Liu [32]	Configurable	10	137.5	130	1.2	1100	491	63.6	28.9	4.48
Mondal [37, K8]	64-QAM	64	158	65	1	100	1760	237.6	2376.0	0.06
Moezzi [35][a]	16-QAM	5	526	130	1.5	480	33.5	33	68.67	14.32

[a] Soft-output detector

Table 4 Comparison of non-constant K-best implementations for 4×4 MIMO

Work	Modulation	f_{CLK} (MHz)	Tech. (nm)	V_{dd} (V)	Φ_{norm} (Mbps)	Area (kGE)	$\mathrm{Power}_{\mathrm{norm}}$ (mW)	E_{bit} (pJ/bit)	TAR (Mbps/kGE)
Moezzi [34]	16-QAM	270	250	N/A	2077	131	-	-	15.85
Tsai [52]	64-QAM	62.5	180	1.8	4154	366	78	18.8	11.35

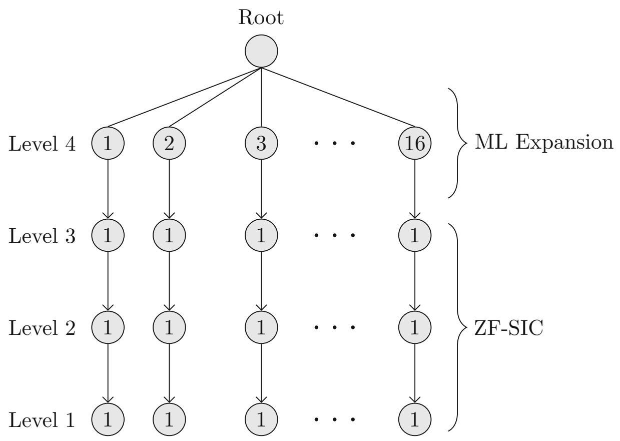

Fig. 11 FSD tree search with $N_T = 4$ and $|\mathcal{S}| = 16$

ensures that the FSD does not miss the ML solution right at the beginning of the search, which is a pitfall of the K-best detector. The number of extended nodes is decreased in subsequent layers to satisfy the relation $n_{N_T} \geq n_{N_T-1} \geq \cdots \geq n_1$ [9]. For example, a node distribution of $(1, 1, 1, 16)^T$ implies that 16 nodes are expanded at the topmost layer while only a single node is expanded from each parent in the remaining layers as illustrated in Fig. 11. In subsequent layers, a simple decision feedback equalization using a linear detector (such as zero-forcing) is carried out to extend a single node from each parent.

Since all the nodes are expanded in the topmost layer and only a simple linear detection is carried out in subsequent layers, the FSD is able to eliminate the sorting operation that is required in the conventional K-best detector [10]. Although the FSD was originally formulated for a complex constellation, it can also be used on real-valued constellations.

The FSD also introduces a novel channel ordering at the preprocessing stage, where the ML expansion at the topmost level is executed on the weakest substream, that is, the substream with the smallest post-detection SNR [57]. In subsequent layers, however, the linear detection is carried out on the substream with the largest post-detection SNR among the yet-to-be detected substreams, which is quite similar to ZF-SIC detection in V-BLAST systems [57]. In this respect, the FSD can also be considered to be a hybrid scheme combining ML and linear detection.

7.2 Hardware Implementations

The first hardware implementation of the FSD was by Barbero and Thompson [9], which was realized on an FPGA device for a 4×4 16-QAM system employing a node distribution of $(1, 1, 1, 16)^T$. The performance degradation of the implementation with respect to the ML at a BER of 10^{-3} is only 0.06 dB. The implementation achieves a throughput of 400 Mbps.

Liu et al. [33] implemented a modified FSD algorithm that uses extension numbers [32] to replace the ML search in the top layer of the tree with a reliability-based search, where more children are extended from the more reliable parent nodes. This results in an "imbalanced" tree expansion, where an unequal number of nodes is extended from each parent node. Similar to [32], this work employs the use of a candidate generation unit for precomputing the $R_{i,j}s_j$ values required to detect a received symbol vector. In this case, an ORVD channel model (7) is employed, which allows the precomputed results to be shared by adjacent layers. The implementation achieves a throughput of 1.98 Gbps with an area consumption of 88.2 kGE.

Chen et al. [17] extended the imbalanced FSD architecture to support iterative decoding. Unlike the STS-SD enumeration [30], which uses 2 symbols (channel-based and a priori-based) in deciding the node for the next visit, this implementation derives an extra symbol that is based on the a priori-based node, and is closest to the channel-based node, in order to get a better estimate of the node with the minimum $\mathcal{M}_P$. The implementation achieves a throughput of approximately 3 Gbps per iteration with an area consumption of 555 kGE.

The results of the VLSI implementation of the FSD are provided in Table 5. The results show that the FSD is a good candidate for achieving a high-throughput performance. Although the FSD requires a channel ordering in the preprocessing stage, the operations can be considered to be negligible in a slow-fading channel [9]. However, in a fast-fading channel, or when using orthogonal frequency division multiplexing (OFDM), where the preprocessor needs to be invoked on a tone-by-tone basis, the added preprocessing operations of the FSD could become significant.

8 The Best-first Tree Search

8.1 Algorithm Description

Unlike the DFS and BFS algorithms, the best-first search (BeFS) always extends along the path of the least-metric node irrespective of its level on the tree. The most popular implementation of the BeFS is the *stack decoder* [2], which maintains a sorted list for storing all the expanded nodes and always extends the tree along the path of the node at the top of the list (i.e. the least-metric node). The search is terminated whenever a leaf node emerges on top of the list and its path is presented as the ML estimate [19].

The need for a sorted list to store all the expanded nodes obviously makes the BeFS a memory-hungry algorithm, and a constraint is usually applied to the memory in order to reduce its complexity in hardware [19]. The detector may also spend too much time in the upper layers without reaching a leaf node under a given time constraint [31].

Figure 12a illustrates the BeFS for a 2-ary tree. The current best node is fetched off the top of the stack and is replaced by all of its children. However, this may be expensive if M is large. Alternatively, a node may be replaced by its best child and best sibling as illustrated in Fig. 12b [47]. The modified BeFS reduces the complexity of the BeFS as only the PED of two nodes needs to be computed at a time in order to extend the tree.

Table 5 Comparison of fixed-complexity SD implementations for 4 × 4 MIMO

Work	Modulation	f_{CLK} (MHz)	Tech. (nm)	V_{dd} (V)	Φ_{norm} (Mbps)	Area (kGE)	Power$_{\mathrm{norm}}$ (mW)	E_{bit} (pJ/bit)	TAR (Mbps/kGE)
Barbero [9]	16-QAM	100	FPGA	N/A	400	N/A	–	–	–
Khairy [27]	16-QAM	102	FPGA	N/A	800	N/A	–	–	–
Liu [33]	64-QAM	165	65	1.2	1980	88.2	102.7	51.9	22.45
Chen [17][a]	64-QAM	370	90	1	3046	555	349.2	114.6	5.49

[a] Soft-input soft-output detector

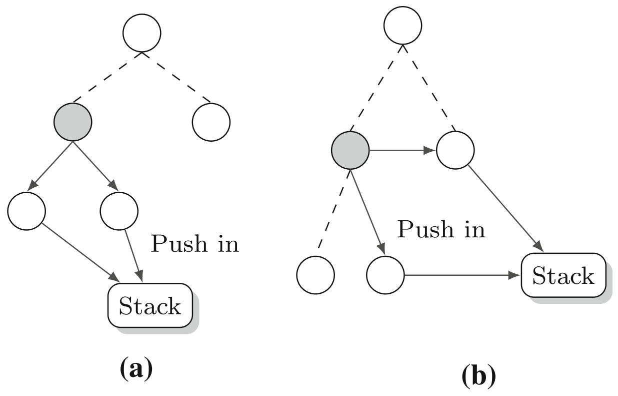

Fig. 12 Best-first tree search using BPSK [31]. The *shaded* node represents the current best node. **a** Conventional BeFS. **b** Modified BeFS

Like the DFS, the BeFS has a variable complexity; however, it achieves a better worst-case and average complexity than the DFS [18]. A more detailed discussion of the BeFS is provided in [38], where it is indicated that the BeFS achieves the best performance-complexity trade-off among all the TS algorithms as it expands the fewest number of nodes on average.

8.2 Hardware Implementations

Liao et al. [31] implemented a soft-output BeFS detector, which is capable of supporting QPSK up to 64-QAM modulations and 2×2 up to 8×8 antenna configurations. The stack is managed using a quad-dual-heap data structure [16], which reduces the complexity of identifying the best and worst nodes. This work also computes the PED incrementally, by distributing its computation over previous levels, which significantly reduces the critical path. The implementation achieves a maximum throughput of 863.6 Mbps in the 4×4, 64-QAM configuration.

Shen et al. [48] implemented a soft-/hard-output BeFS detector, which adopts features of depth-first and breadth-first search proposed by the authors in [47]. For any selected node, the detector enumerates to its best child and next best sibling and then extends along the path with the lower metric. Additionally, a Fano-like bias [42] is used to enable the detector to generate full solutions more quickly. The implementation achieves an average throughput of 199.8 Mbps for the hard-detection case and an average throughput of 83.3 Mbps for the soft-detection case over the entire SNR range.

The relevant performance metrics are provided in Table 6. As expected, the implementation of Liao et al. [31] incurs a high area consumption due to its configurable architecture, which is capable of supporting up to 8 antennas. The implementation also achieves a comparatively high throughput by setting the maximum number of visited nodes to a small value of 8 under good channel conditions; however, this degrades

Table 6 Best-first search VLSI Implementation for 4 × 4 MIMO

Work	Modulation	f_{CLK} (MHz)	Tech. (nm)	V_{dd} (V)	Φ_{norm} (Mbps)	Area (kGE)	$\text{Power}_{\text{norm}}$ (mW)	E_{bit} (pJ/bit)	TAR Mbps/kGE	Ref. SNR (dB)
Liao [31][a]	64-QAM	198	130	1.3	863.6	350	24.8	28.7	2.47	24.2
Shen [48]	64-QAM	333	65	1	199.8[b]	54.2	10.5	52.3	3.69	-
Shen [48][a]	64-QAM	333	65	1	83.3[b]	64	16.6	198.8	1.30	-

[a] Soft-output detector.
[b] Average throughput over SNR range

the performance compared to [48], which visits a larger number of nodes in its tree search.

9 Design Considerations and Trade-Offs

In the preceding sections, several algorithms for MIMO detection have been presented. The question of which algorithm to adopt does not have a straightforward answer and a number of constraints need to be considered as highlighted in Fig. 13. These design objectives are usually conflicting and achieving the "optimal" performance for one design target often means sacrificing one or more of the others. We discuss the problem in more detail subsequently.

- Computational Complexity: The straightforward means of comparing different TS algorithms is in terms of the computational complexity required to completely detect one received symbol vector. From theoretical results [38], it is established that the BeFS achieves the best performance-complexity trade-off, but this is achieved at the cost of an exponential memory requirement. A constraint may be applied to the memory of the BeFS; however, this sacrifices its attainable ML performance. Like the SD, the BeFS also suffers from a variable complexity, giving it a relatively poor worst-case throughput. The BFS schemes, in general, incur the highest computational complexity; however, their SNR-independent throughput makes them ideal for real-time applications.
- Bit Error Rate: The SNR required by an algorithm to meet a target BER has been an important metric for comparing different detection algorithms, especially in theoretical analyses. However, in hardware implementation, achieving the "optimal" BER is likely to incur a very high complexity. For a performance-centric application, the SD and BeFS algorithms are the best candidates as they achieve the ML performance. When memory is abundant, the BeFS may be preferred to the SD in order to further reduce the computational complexity. On the other hand, the

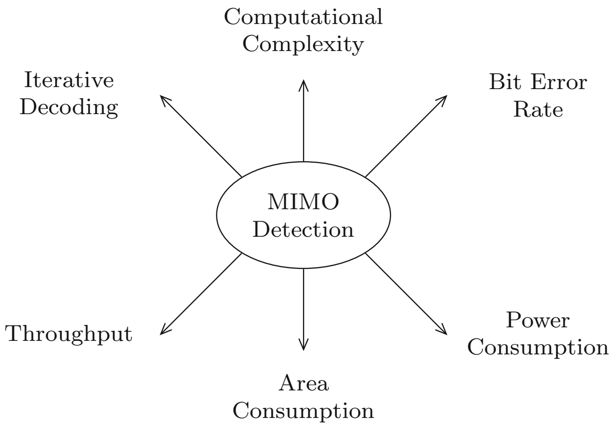

Fig. 13 Design trade-offs for MIMO detection implementation

BFS schemes deliver a near-ML performance, but with a much more predictable complexity than either the SD or BeFS.

- Power Consumption: It is very crucial that the MIMO detector achieves a low power consumption, given that it is only one part of a larger receiver unit, functioning on a tight energy budget. Due to their pipelined architecture and requirement for sorting, the K-best detectors have a tendency to incur a high power consumption; however, this shortcoming is compensated by their high-throughput performance, which generally leads to a highly energy efficient implementation. Several techniques have been presented in the literature for reducing the power consumption of the detector block. One notable observation in this regard is that the symbol detection may be separated into symbol rate processing and channel rate processing, which may be exploited in order to achieve further power reductions, especially in a block-fading channel. For example, certain Euclidean distance computations may be performed at the slower channel rate thereby simplifying the detector unit (e.g. [32]). This fact also seems to favour linear equalization techniques, as the matrix inversion can be performed once at the channel rate for multiple received symbol vectors.
- Throughput: Even with a low power consumption, the detector must have a sufficiently high throughput to achieve a high energy efficiency. From the results of the works surveyed, the BFS schemes achieve the highest throughput performance, due to their suitability to pipelining and their SNR-independent complexity. The SD and BeFS may achieve a very high peak throughput at high SNR, but their throughput degradation at low SNR limits their attainable energy efficiency to an extent. Techniques for achieving high throughput include using single-cycle merge [34,55] and approximate sorting algorithms [28] in the case of the K-best detector, and the use of channel ordering [56] and runtime constraints [13] for the SD algorithm.
- Area Consumption: The SD requires the smallest area to meet a given BER target due to its one-node-per-cycle architecture [13]. Although the BFS algorithms can be implemented using a similar folded architecture, they typically require more area to achieve the same BER compared to the SD, due to the requirement for more intermediate results and sorting (in the case of the K-best). In general, folded architectures achieve a smaller area than pipelined implementations; however, penalty is incurred on the achievable throughput. Other techniques for achieving a low area consumption include the use of real-valued detection, serial sorting and the use of norm approximations for the computation of the Euclidean distance.
- Iterative Decoding: In a practical system, the MIMO detector is concatenated with a channel decoder in order to improve the BER. A good hard-detection performance is important in order to reduce the complexity of the iterative implementation. For example, the STS-SD typically requires fewer iterations in order to meet a given BER target compared to less "optimal" detectors [12,22]. Meanwhile, the BFS schemes appear to be very attractive for soft-output generation, due to the redundant solutions already present in the hard-output detection. By contrast, the SD algorithm typically needs to be modified in order to generate soft outputs and this may lead to a degradation in the throughput. For this reason, the SD algorithm

is preferable when performance is vital, while BFS algorithms are preferable for applications requiring a moderate performance at a higher throughput.

10 Conclusion and Future Works

MIMO technology has matured from a topic of mainly theoretical interest, to becoming indispensable in modern communication systems. Although MIMO allows vastly higher data rates to be achieved, this comes at the cost of a more complex receiver design, which has inspired numerous research works in the past few decades. The sphere decoder and related tree search algorithms provide an excellent performance-complexity trade-off, which makes them suited to hardware implementation, especially for small antenna dimensions. The tree search algorithms have been extended to iterative decoding as well, which allows improved error rate performances to be achieved.

In the upcoming 5G standard, which is anticipated in the next few years, energy efficiency is likely to be a major concern, as a diverse range of platforms will become interconnected. As such, low-power design techniques and algorithms need to be investigated in order to meet the new challenges. One possible means of achieving highly energy efficient designs is by dispensing with the channel estimation completely and adopting non-coherent detection techniques. Apart from reducing the energy cost of the receiver, this has the potential of allowing a more efficient bandwidth utilization, due to the elimination of the training phase required by coherent receivers [60].

Another area for possible investigation is the use of adaptive MIMO detection, which exploits the rich wireless channel in order to achieve further energy savings. Most of the works surveyed in this paper implement a *fixed-effort* signal detection regardless of the channel condition, which can lead to unnecessary energy consumption. Hybrid detectors that switch to simpler detection strategies (e.g. linear equalization) depending on the channel state appear promising in this regard and merit more attention from future works.

Acknowledgments The authors wish to acknowledge the Petroleum Technology Development Fund (PTDF), Nigeria, for supporting this work.

References

1. I.F. Akyildiz, D.M. Gutierrez-Estevez, R. Balakrishnan, E. Chavarria-Reyes, LTE-advanced and the evolution to beyond 4g (b4g) systems. Phys. Commun. **10**, 31–60 (2014). doi:10.1016/j.phycom.2013.11.009. ISSN 1874-4907
2. J. Anderson, S. Mohan, Sequential coding algorithms: a survey and cost analysis. IEEE Trans. Commun. **32**(2), 169–176 (1984)
3. J.G. Andrews, S. Buzzi, Wan Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, J.C. Zhang, What will 5g be? IEEE J. Sel. Areas Commun. **32**(6), 1065–1082 (2014). doi:10.1109/JSAC.2014.2328098. ISSN 0733-8716

4. L. Azzam, E. Ayanoglu, Reduced complexity sphere decoding for square QAM via a new lattice representation, in *IEEE Global Telecommunications Conference, 2007. GLOBECOM '07* (Nov 2007), pp. 4242–4246. doi:10.1109/GLOCOM.2007.807
5. L. Azzam, E. Ayanoglu, Reduced complexity sphere decoding via a reordered lattice representation. IEEE Trans. Commun. **57**(9), 2564–2569 (2009). doi:10.1109/TCOMM.2009.09.070238. ISSN 0090-6778
6. L. Babai, On Lovász lattice reduction and the nearest lattice point problem. Combinatorica **6**(1), 1–13 (1986)
7. A. Bacioccola, C. Cicconetti, C. Eklund, L. Lenzini, Z. Li, E. Mingozzi, IEEE 802.16: History, status and future trends. Comput. Commun. **33**(2), 113–123 (2010). doi:10.1016/j.comcom.2009.11.003. ISSN 01403664
8. L.G. Barbero, J.S. Thompson, A fixed-complexity MIMO detector based on the complex sphere decoder, in *IEEE 7th Workshop on Signal Processing Advances in Wireless Communications, 2006. SPAWC '06* (2006a), pp. 1–5. doi:10.1109/SPAWC.2006.346388
9. L.G. Barbero, J.S. Thompson, Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems, in *IEEE International Conference on Communications, 2006. ICC '06*, vol. 7 (2006b), pp. 3082–3087. doi:10.1109/ICC.2006.255278
10. L.G. Barbero, J.S. Thompson, Fixing the complexity of the sphere decoder for MIMO detection. IEEE Trans. Wirel. Commun. **7**(6), 2131–2142 (2008). doi:10.1109/TWC.2008.060378. ISSN 1536-1276
11. S. Baro, J. Hagenauer, M. Witzke, Iterative detection of MIMO transmission using a list-sequential (LISS) detector, in *IEEE International Conference on Communications, 2003. ICC '03*, vol. 4 (May 2003), pp. 2653–2657. doi:10.1109/ICC.2003.1204433
12. F. Borlenghi, E.M. Witte, G. Ascheid, H. Meyr, A. Burg. A 772mbit/s 8.81bit/nJ 90nm CMOS soft-input soft-output sphere decoder, in *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian* (Nov 2011), pp. 297–300. doi:10.1109/ASSCC.2011.6123571
13. A. Burg, M. Borgmann, C. Studer, H. Bolcskei,. Advanced receiver algorithms for MIMO wireless communications, in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1 (Munich, Mar 2006), p. 6
14. A. Burg, M. Wenk, M. Zellweger, M. Wegmueller, N. Felber, W. Fichtner, VLSI implementation of the sphere decoding algorithm, in *Solid-State Circuits Conference, 2004. ESSCIRC 2004. Proceeding of the 30th European* (2004), pp. 303–306. doi:10.1109/ESSCIR.2004.1356678
15. A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, H. Bolcskei, VLSI implementation of MIMO detection using the sphere decoding algorithm. IEEE J. Solid-State Circuits **40**(7), 1566–1577 (2005). doi:10.1109/JSSC.2005.847505. ISSN 0018-9200
16. S. Carlsson, The deap a double-ended heap to implement double-ended priority queues. Inf. Process. Lett. **26**(1), 33–36 (1987). doi:10.1016/0020-0190(87)90033-0. ISSN 0020-0190
17. X. Chen, G. He, J. Ma, VLSI implementation of a high-throughput iterative fixed-complexity sphere decoder. IEEE Trans. Circuits Syst. II Express Br. **60**(5), 272–276 (2013). doi:10.1109/TCSII.2013.2251954. ISSN 1549-7747
18. Y. Dai, S. Sun, Z. Lei, A comparative study of QRD-m detection and sphere decoding for MIMO-OFDM systems, in *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005*, vol. 1 (Sep 2005), pp. 186–190. doi:10.1109/PIMRC.2005.1651424
19. Y. Dai, Z. Yan, Memory-constrained tree search detection and new ordering schemes. IEEE J. Sel. Top. Signal Process. **3**(6), 1026–1037 (2009). doi:10.1109/JSTSP.2009.2039657. ISSN 1932-4553, 1941-0484
20. M. El-Hajjar, L. Hanzo, Multifunctional MIMO systems: a combined diversity and multiplexing design perspective. IEEE Wirel. Commun. **17**(2), 73–79 (2010). doi:10.1109/MWC.2010.5450663. ISSN 1536-1284
21. U. Fincke, M. Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**, 463–471 (1985)
22. C. Gimmler-Dumont, F. Kienle, W. Bin, G. Masera, A system view on iterative mimo detection: dynamic sphere detection versus fixed effort list detection. VLSI Des. **2012**, 2 (2012)
23. Z. Guo, P. Nilsson, Algorithm and implementation of the k-best sphere decoding for MIMO detection. IEEE J. Sel. Areas Commun. **24**(3), 491–503 (2006). doi:10.1109/JSAC.2005.862402. ISSN 0733-8716
24. B.M. Hochwald, S. ten Brink, Achieving near-capacity on a multiple-antenna channel. IEEE Trans. Commun. **51**(3), 389–399 (2003). doi:10.1109/TCOMM.2003.809789. ISSN 0090-6778

25. R. Jenkal, R. Davis, An architecture for energy efficient sphere decoding, in *2007 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)* (2007), pp. 244–249. doi:10.1145/1283780.1283833
26. S.-H. Kang, I.-C. Park, High speed sphere decoding based on vertically incremental computation, in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007* (May 2007), pp. 665–668. doi:10.1109/ISCAS.2007.377896
27. M.S. Khairy, M.M. Abdallah, S.E.-D. Habib, Efficient FPGA implementation of MIMO decoder for mobile WiMAX system, in *IEEE International Conference on Communications, 2009. ICC '09* (June 2009), pp. 1–5. doi:10.1109/ICC.2009.5198971
28. T.-H. Kim, I.-C. Park, Small-area and low-energy-best MIMO detector using relaxed tree expansion and early forwarding. IEEE Trans. Circuits Syst. I Reg. Pap. **57**(10), 2753–2761 (2010). doi:10.1109/TCSI.2010.2046249. ISSN 1549-8328
29. B.Y. Kong, I.-C. Park, Hardware-efficient tree expansion for MIMO symbol detection. Electron. Lett. **49**(3), 226–228 (2013). doi:10.1049/el.2012.1074. ISSN 0013-5194
30. C.-H. Liao, I.-W. Lai, K. Nikitopoulos, F. Borlenghi, D. Kammler, M. Witte, D. Zhang, T.-D. Chiueh, G. Ascheid, H. Meyr, Combining orthogonalized partial metrics: efficient enumeration for soft-input sphere decoder, in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, IEEE (2009), pp. 1287–1291
31. C.-H. Liao, T.-P. Wang, T.-D. Chiueh, A 74.8 mW soft-output detector IC for 8 x 8 spatial-multiplexing MIMO communications. IEEE J. Solid-State Circuits **45**(2), 411–421 (2010). doi:10.1109/JSSC.2009.2037292. ISSN 0018-9200
32. L. Liu, F. Ye, X. Ma, T. Zhang, Junyan Ren, A 1.1-gb/s 115-pJ/bit configurable MIMO detector using 0.13-m CMOS technology. IEEE Trans. Circuits Syst. II Express Br. **57**(9), 701–705 (2010). doi:10.1109/TCSII.2010.2058494. ISSN 1549-7747
33. L. Liu, J. Lofgren, P. Nilsson, Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes. IEEE Trans. Circuits Syst. I Reg. Pap. **59**(9), 2085–2096 (2012). doi:10.1109/TCSI.2012.2185297. ISSN 1549-8328
34. N. Moezzi-Madani, W. R. Davis, High-throughput low-complexity MIMO detector based on k-best algorithm, in *Proceedings of the 19th ACM Great Lakes symposium on VLSI* (2009b), pp. 451–456
35. N. Moezzi-Madani, T. Thorolfsson, W.R. Davis, A low-area flexible MIMO detector for WiFi/WiMAX standards, in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010* (Mar 2010), pp. 1633–1636. doi:10.1109/DATE.2010.5457073
36. N. Moezzi-Madani, W.R. Davis, Parallel merge algorithm for high-throughput signal processing applications. Electron. Lett. **45**(3), 188–189 (2009a)
37. S. Mondal, A. Eltawil, Chung-An Shen, K.N. Salama, Design and implementation of a sort-free k-best sphere decoder. IEEE Trans. Very Large Scale Integr. VLSI Syst. **18**(10), 1497–1501 (2010). doi:10.1109/TVLSI.2009.2025168. ISSN 1063-8210
38. A.D. Murugan, H. El Gamal, M.O. Damen, G. Caire, A unified framework for tree search decoding: rediscovering the sequential decoder. IEEE Trans. Inf. Theory **52**(3), 933–953 (2006). doi:10.1109/TIT.2005.864418. ISSN 0018-9448
39. M. Myllyl, M. Juntti, J.R. Cavallaro, Implementation aspects of list sphere detector algorithms, in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, IEEE (2007), pp. 3915–3920
40. G. Papa, D. Ciuonzo, G. Romano, P.S. Rossi, A dominance-based soft-input soft-output MIMO detector with near-optimal performance. IEEE Trans. Commun. **62**(12), 4320–4335 (2014). doi:10.1109/TCOMM.2014.2367013. ISSN 0090-6778
41. D. Patel, V. Smolyakov, M. Shabany, P.G. Gulak, VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output k-best MIMO detector, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (2010), pp. 593–596. doi:10.1109/ISCAS.2010.5537524
42. V.T. Pham, Z. Lei, T.T. Tjhung, Fano detection algorithm for MIMO systems, in *Information, Communications & Signal Processing, 2007 6th International Conference on* (2007), pp. 1–5
43. G. Romano, D. Ciuonzo, P.S. Rossi, F. Palmieri, Low-complexity dominance-based sphere decoder for MIMO systems. Signal Process. **93**(9), 2500–2509 (2013). doi:10.1016/j.sigpro.2013.02.011. ISSN 0165-1684
44. C.P. Schnorr, M. Euchner, Lattice basis reduction: improved practical algorithms and solving subset sum problems, in *Mathematical Programming* (1993), pp. 181–191

45. M. Shabany, P.G. Gulak, A 0.13 m CMOS 655mb/s 4x4 64-QAM k-best MIMO detector, in *Solid-State Circuits Conference—Digest of Technical Papers, 2009. ISSCC 2009. IEEE International* (2009), pp. 256–257,257a. doi:10.1109/ISSCC.2009.4977405
46. M. Shabany, P.G. Gulak, Scalable VLSI architecture for k-best lattice decoders, in *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008* (2008), pp. 940–943. doi:10.1109/ISCAS.2008.4541574
47. C.-A. Shen, A.M. Eltawil, S. Mondal, K.N. Salama, A best-first tree-searching approach for ML decoding in MIMO system, in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* (2010), pp. 3533–3536
48. C.-A. Shen, A.M. Eltawil, K.N. Salama, S. Mondal, A best-first soft/hard decision tree searching MIMO decoder for a 4 4 64-QAM system. IEEE Trans. Very Large Scale Integr. VLSI Syst. **20**(8), 1537–1541 (2012). doi:10.1109/TVLSI.2011.2159821. ISSN 1063-8210
49. C. Studer, A. Burg, H. Bolcskei, Soft-output sphere decoding: algorithms and VLSI implementation. IEEE J. Sel. Areas Commun. **26**(2), 290–300 (2008). doi:10.1109/JSAC.2008.080206. ISSN 0733-8716
50. C. Studer, H. Bolcskei, Soft-input soft-output single tree-search sphere decoding. IEEE Trans. Inf. Theory **56**(10), 4827–4842 (2010). doi:10.1109/TIT.2010.2059730. ISSN 0018-9448
51. S. Sugiura, S. Chen, L. Hanzo, MIMO-aided near-capacity turbo transceivers: taxonomy and performance versus complexity. IEEE Commun. Surv. Tutor. **14**(2), 421–442 (2012). doi:10.1109/SURV.2011.032511.00136. ISSN 1553-877X
52. P.-Y. Tsai, W.-T. Chen, X.-C. Lin, M.-Y. Huang, A 4x4 64-QAM reduced-complexity k-best MIMO detector up to 1.5gbps, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (2010), pp. 3953–3956. doi:10.1109/ISCAS.2010.5537675
53. E. Viterbo, J. Boutros, A universal lattice code decoder for fading channels. IEEE Trans. Inf. Theory **45**(5), 1639–1642 (1999). doi:10.1109/18.771234. ISSN 0018-9448
54. M. Wenk, L. Bruderer, A. Burg, C. Studer, Area- and throughput-optimized VLSI architecture of sphere decoding, in *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP* (Sep 2010), pp. 189–194. doi:10.1109/VLSISOC.2010.5642593
55. M. Wenk, M. Zellweger, A. Burg, N. Felber, W. Fichtner, K-best MIMO detection VLSI architectures achieving up to 424 mbps, in *2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006. Proceedings* (2006), pp. 4 pp–1154. doi:10.1109/ISCAS.2006.1692794
56. A. Wiesel, X. Mestre, A. Pages, J.R. Fonollosa, Efficient implementation of sphere demodulation, in *4th IEEE Workshop on Signal Processing Advances in Wireless Communications, 2003. SPAWC 2003* (2003), pp. 36–40. doi:10.1109/SPAWC.2003.1318917
57. P.W. Wolniansky, G.J. Foschini, G.D. Golden, R.A. Valenzuela, V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel, in *1998 URSI International Symposium on Signals, Systems, and Electronics, 1998. ISSSE 98* (Oct 1998), pp. 295 –300. doi:10.1109/ISSSE.1998.738086
58. K. Wong, C. Tsui, R.S.-K. Cheng, W.H. Mow, A VLSI architecture of a k-best lattice decoding algorithm for MIMO channels, in *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, vol. 3 (2002), pp. III–273–III–276. doi:10.1109/ISCAS.2002.1010213
59. K.-J. Yang, S.-H. Tsai, R.-C. Chang, Y.-C. Chen, G.C.-H. Chuang, VLSI implementation of a low complexity 4x4 MIMO sphere decoder with table enumeration, in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)* (2013), pp. 2167–2170. doi:10.1109/ISCAS.2013.6572304
60. L. Zheng, D.N.C. Tse, Communication on the Grassmann manifold: a geometric approach to the noncoherent multiple-antenna channel. IEEE Trans. Inf. Theory **48**(2), 359–383 (2002). doi:10.1109/18.978730. ISSN 0018-9448