VLSI Implementation of Hard- and Soft-Output Sphere Decoding for Wide-Band MIMO Systems

Christoph Studer[1], Markus Wenk[2], and Andreas Burg[3]

[1]Dept. of Electrical and Computer Engineering, Rice University, Houston, TX 77005, USA; e-mail: studer@rice.edu

[2]Dept. of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland; e-mail: mawenk@iis.ee.ethz.ch

[3]School of Engineering, EPF Lausanne, 1015 Lausanne, Switzerland; e-mail: andreas.burg@epfl.ch

Abstract. Multiple-input multiple-output (MIMO) technology in combination with orthogonal frequency-division multiplexing (OFDM) is the key to meet the demands for data rate and link reliability of modern wide-band wireless communication systems, such as IEEE 802.11n or 3GPP-LTE. The full potential of such systems can, however, only be achieved by high-performance data-detection algorithms, which typically exhibit prohibitive computational complexity. Hard-output sphere decoding (SD) and soft-output single tree-search (STS) SD are promising means for realizing high-performance MIMO detection and have been demonstrated to enable efficient implementations in practical systems. In this chapter, we consider the design and optimization of register transfer-level implementations of hard-output SD and soft-output STS-SD with minimum area-delay product, which are well-suited for wide-band MIMO systems. We explain in detail the design, implementation, and optimization of VLSI architectures and present corresponding implementation results for 130 nm CMOS technology. The reported implementations significantly outperform the area-delay product of previously reported hard-output SD and soft-output STS-SD implementations.

Key words: VLSI implementation, MIMO-OFDM communication systems, sphere decoding (SD), single tree-search (STS) SD algorithm.

1 Introduction

The evolution of data rate and quality-of-service in modern wide-band wireless communication systems is fueled by novel physical-layer technologies providing high spectral efficiency and excellent link reliability. Multiple-input multiple-output (MIMO) technology [1, 2], which employs multiple antennas at both ends of the wireless link, in combination with spatial multiplexing, orthogonal frequency-division multiplexing (OFDM), and channel coding is believed to be the key for reliable, high-speed, and bandwidth-efficient data transmission.

Therefore, MIMO-OFDM technology is incorporated in many modern wide-band wireless communication standards, such as IEEE 802.11n [3] or 3GPP-LTE [4].

In such systems, data detection, i.e., the separation of the multiplexed data streams, is (besides channel decoding) typically among the main implementation challenges in terms of computational complexity and power consumption. Therefore, corresponding *efficient VLSI implementations* are the key to enable high-performance, low-power, and low-cost user equipment. The performance of MIMO technology critically depends on the employed data-detection algorithm and corresponding high-performance methods usually entail very high complexity. In particular, a straightforward implementation of hard-output maximum-likelihood (ML) detection and soft-output a-posteriori probability (APP) detection—both providing excellent error-rate performance—requires to exhaustively test all possible transmit symbols, which results in prohibitive complexity, even for moderate data rates and in deep submicron technologies.

The sphere-decoding (SD) algorithm [5–11] is known to be a promising means for efficient hard-output ML and soft-output APP detection. The key idea of SD is to transform MIMO detection into a tree-search problem, which can then be solved efficiently through a branch-and-bound procedure. The drawback of this approach lies in the fact that the decoding effort—measured in terms of the *number of nodes* to be examined during the tree search—depends on the instantaneous channel and noise realization. In the worst-case, the number of visited nodes, which typically corresponds to the number of clock cycles required for detection in VLSI [11, 12], is equivalent to that of an exhaustive search [13]. Since on-chip storage and higher-layer requirements limit the processing latency that may be inferred to support the processing of received data, the worst-case complexity of SD renders its application in real-world systems extremely challenging. This challenge can be mastered by limiting the maximum decoding effort by means of *early termination* of the decoding process [11, 14, 15]. This approach, however, leads to a trade-off between the maximum decoding effort and the performance of the MIMO detector. Therefore, a universally applicable VLSI architecture for SD-based MIMO detection suitable for wide-band MIMO wireless communication systems must provide a robust solution allowing for the smooth adjustment of this trade-off while minimizing the required silicon area for a given minimum performance requirement.

1.1 Contributions

In this chapter, we describe an SD-based detector architecture for wide-band MIMO communication systems and detail corresponding design and implementation trade-offs of hard- and soft-output SD. To this end, we first review the hard-output SD algorithm and the soft-output single tree-search (STS) SD algorithm. Then, a VLSI architecture suitable for efficient data detection in wide-band MIMO systems is presented and we argue that the optimization target for the parallelly-operating SD cores corresponds to minimizing the area-delay product, which differs fundamentally from minimizing the area or maximizing the throughput, as it would be the case for narrow-band systems. To arrive at SD architectures that minimize the area-delay product, we start with the VLSI implementations for hard-output SD [12] and soft-output STS-SD [11] and propose a variety of optimizations, which improve (i.e., lower) the area-delay product of the detectors. In particular, we propose a low-complexity approximation to the Schnorr-Euchner (SE) enumeration scheme and employ pipeline interleaving, which enables us to achieve the desired design goals. We finally present implementation results for 130 nm CMOS technology and perform a comparison to previously reported implementations of SD.

1.2 Outline of the Chapter

The remainder of this chapter is organized as follows. In Section 2, the MIMO system model is introduced and the employed hard-output and soft-output STS-SD algorithms are reviewed. In Section 3, we develop a receiver architecture suitable for wide-band MIMO systems and analyze the optimization goals for the SD-core implementations. The VLSI architectures for hard-output SD and soft-output STS-SD along with corresponding optimization techniques are detailed in Section 4 and Section 5, respectively. VLSI-implementation results are presented in Section 6 and we conclude in Section 7.

1.3 Notation

Matrices are set in boldface capital letters, column-vectors in boldface lowercase letters. The superscripts ^{T} and ^{H} stand for transposition and conjugate transposition, respectively. The real and imaginary part of a complex-valued number x are denoted by $\Re(x)$ and $\Im(x)$, respectively. The ℓ_2-norm (or Euclidean norm) of a vector $\mathbf{x}$ is designated by $\|\mathbf{x}\|$, the ℓ_∞-norm of $\mathbf{x}$ is $\|\mathbf{x}\|_\infty = \max_i |x_i|$, and the $\ell_{\widetilde{\infty}}$-norm is defined as $\|\mathbf{x}\|_{\widetilde{\infty}} = \max\{\|\Re(\mathbf{x})\|_\infty, \|\Im(\mathbf{x})\|_\infty\}$. The binary complement of $x \in \{0,1\}$ is denoted by $\overline{x}$. Probability and expectation are referred to as $\Pr[\cdot]$ and $\mathsf{E}[\cdot]$, respectively.

2 MIMO System Model and Sphere Decoding

In this section, we introduce the wide-band MIMO system model and summarize the hard-output and soft-output SD algorithms investigated in the remainder of the chapter.

2.1 Wide-Band MIMO System Model

We consider a coded wide-band MIMO system employing spatial multiplexing with M_T transmit and $M_\mathrm{R} \geq M_\mathrm{T}$ receive antennas (see Fig. 1) and orthogonal frequency-division multiplexing (OFDM). The information bits $\mathbf{b}$ are encoded (e.g., using a convolutional code) and interleaved (denoted by $\prod$ in Fig. 1). The resulting coded bit-stream $\mathbf{x}$ is mapped (using Gray labeling) to a sequence of transmit vectors $\mathbf{s}[k] \in \mathcal{O}^{M_\mathrm{T}}$, where $\mathcal{O}$ corresponds to the scalar complex

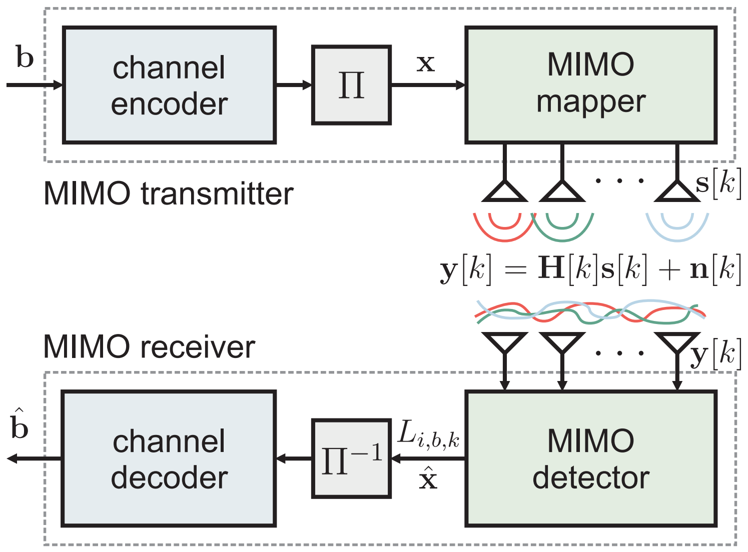


Fig. 1. Coded wide-band MIMO communication system.

constellation of size 2^Q and $k = 1, \ldots, T$ designates the OFDM-tone index; the maximum number of OFDM carriers corresponds to T. Each transmit vector $\mathbf{s}[k]$ is associated with $M_\mathrm{T}Q$ binary values $x_{i,b,k} \in \{0, 1\}$, $i = 1, \ldots, M_\mathrm{T}$, $b = 1, \ldots, Q$ corresponding to the bth bit of the ith entry (i.e., spatial stream) of $\mathbf{s}[k]$. The baseband input-output relation of the wireless MIMO channel for each OFDM tone is given by

$$\mathbf{y}[k] = \mathbf{H}[k]\mathbf{s}[k] + \mathbf{n}[k] \tag{1}$$

where $\mathbf{H}[k]$ stands for the $M_\mathrm{R} \times M_\mathrm{T}$ complex-valued channel matrix on OFDM tone k, $\mathbf{y}[k]$ is the M_R-dimensional received vector, and $\mathbf{n}[k]$ is M_R-dimensional i.i.d. zero-mean complex Gaussian distributed noise with variance N_0 per entry. We assume $\mathsf{E}[\mathbf{s}[k]\mathbf{s}[k]^H] = \frac{1}{M_\mathrm{T}}\mathbf{I}_{M_\mathrm{T}}$ in the following.

In the receiver, a hard-output MIMO detector computes estimates $\hat{\mathbf{s}}[k]$ for the transmit vector, which are then used to generate binary-valued estimates $\hat{\mathbf{x}}$ for the coded bit-stream $\mathbf{x}$. If a soft-output MIMO detector is used, reliability information in the form of log-likelihood ratios (LLRs) $L_{i,b,k}$ for each coded bit $x_{i,b,k}$ is generated instead. For both detection schemes we assume coherent detection, i.e., the channel matrices $\mathbf{H}[k]$, $k = 1, \ldots, T$, and the noise variance N_0 are perfectly known by the receiver. Finally, the MIMO receiver generates estimates for the information bits $\hat{\mathbf{b}}$ using the channel decoder, which operates either on the basis of the de-interleaved (denoted by $\prod^{-1}$ in Fig. 1) bit stream $\hat{\mathbf{x}}$ for hard-output MIMO detectors or on the de-interleaved sequence of LLRs $L_{i,b,k}$ generated by the soft-output MIMO detector. Since the MIMO detector can treat the OFDM tones independently of each other, the tone index k is omitted in the remainder of the chapter.

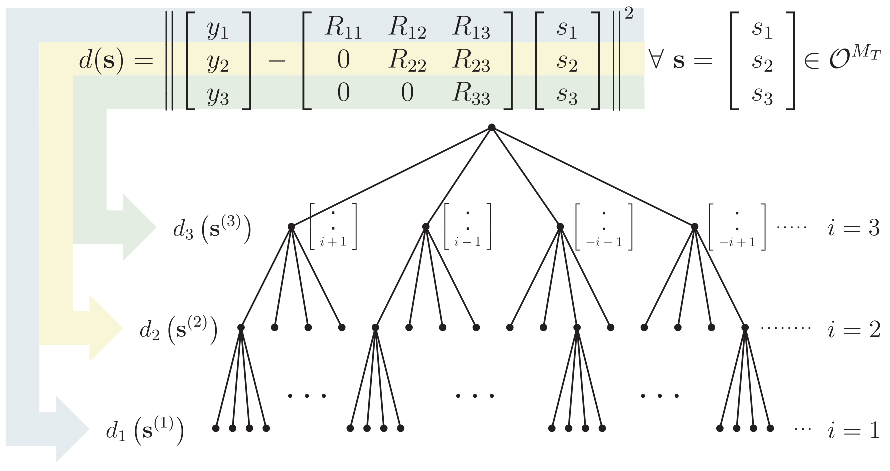

$d(\mathbf{s}) = \left\| \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \right\|^2 \quad \forall\, \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \in \mathcal{O}^{M_T}$
$d_3\left(\mathbf{s}^{(3)}\right)$
$d_2\left(\mathbf{s}^{(2)}\right)$
$d_1\left(\mathbf{s}^{(1)}\right)$
$i+1$
$i-1$
$-i-1$
$-i+1$
$i = 3$
$i = 2$
$i = 1$

Fig. 2. MIMO detection reformulated as a tree-search problem for $M_{\mathrm{T}} = 3$ spatial streams and QPSK modulation.

2.2 ML Detection using the Sphere-Decoding Algorithm

Hard-output MIMO detection using the ML-detection rule maximizes the probability of detecting the correct transmitted vector $\mathbf{s}$. The ML rule for the input-output relation (1) corresponds to [1, 2]

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \in \mathcal{O}^{M_{\mathrm{T}}}}{\arg\min} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{2}$$

and a straightforward evaluation of (2) requires an exhaustive search over all transmit-vectors $\mathbf{s} \in \mathcal{O}^{M_{\mathrm{T}}}$. Since $|\mathcal{O}^{M_{\mathrm{T}}}| = 2^{M_{\mathrm{T}}Q}$, this approach leads—even for a small number of transmit-antennas—to a prohibitive computational complexity. To alleviate this complexity issue, a variety of low-complexity algorithms have been proposed in the literature (see, e.g., [1, 2] and the references therein). Unfortunately, most of the existing low-complexity MIMO detection schemes sacrifice error-rate performance for complexity, which is not desirable for high-performance transceiver implementations. We next summarize the hard-output SD algorithm, which is able to provide ML performance at low (average) computational complexity.

Sphere-Decoding Algorithm The SD algorithm [5–9] starts with the QR decomposition (QRD) of the channel matrix $\mathbf{H} = \mathbf{QR}$, where the $M_{\mathrm{R}} \times M_{\mathrm{T}}$ matrix $\mathbf{Q}$ satisfies $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}_{M_{\mathrm{T}}}$, and the $M_{\mathrm{T}} \times M_{\mathrm{T}}$ matrix $\mathbf{R}$ is upper-triangular. The QRD enables us to rewrite the ML-detection problem (2) as follows:

$$\hat{\mathbf{s}} = \underset{\mathbf{s} \in \mathcal{O}^{M_{\mathrm{T}}}}{\arg\min} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{3}$$

with $\hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$. Thanks to the upper-triangular structure of $\mathbf{R}$, the minimization in (3) can be transformed into a tree-search problem where the nodes of the tree on level i are associated with a partial symbol vector $\mathbf{s}^{(i)} = [\, s_i \cdots s_{M_\mathrm{T}} \,]^T$ and with a corresponding partial Euclidean distance (PED) $d_i\big(\mathbf{s}^{(i)}\big)$. Fig. 2 illustrates the corresponding tree for a MIMO system with $M_\mathrm{T} = M_\mathrm{R} = 3$ using QPSK modulation. It is important to realize that when starting from the root of the tree (at level $i = M_\mathrm{T}+1$ with $d_{M_\mathrm{T}+1} = 0$), the PEDs can efficiently be computed in a recursive manner as follows:

$$d_i\big(\mathbf{s}^{(i)}\big) = d_{i+1}\big(\mathbf{s}^{(i+1)}\big) + |b_{i+1} - R_{i,i}s_i|^2 \tag{4}$$

with the definition

$$b_{i+1} = \hat{y}_i - \sum_{k=i+1}^{M_\mathrm{T}} R_{i,k}s_k \tag{5}$$

when proceeding from a parent node on level $i+1$ to one of its children on level i. Each path from the root down to a leaf corresponds to a symbol vector $\mathbf{s} \in \mathcal{O}^{M_T}$. Since the dependence of the PED d_i on the symbol vector $\mathbf{s}$ is only through $\mathbf{s}^{(i)}$, we have transformed ML detection into a tree-search problem. The ML solution (3) corresponds to the path through the tree starting by the root and leading to the leaf associated with the smallest PED.

The basic ideas underlying the SD algorithm, as described in [9, 12], are briefly summarized as follows: The search in the tree is constrained to nodes which lie within a *radius* r around $\tilde{\mathbf{y}}$ (and hence, nodes from the tree are pruned for which $d_i\big(\mathbf{s}^{(i)}\big)$ is larger than r) and tree traversal is performed depth-first, visiting the children of a given node in ascending order of their PEDs. The method using this enumeration scheme is also known as the Schnorr-Euchner (SE) SD algorithm [6]. In the following, we refer to the condition $d_i\big(\mathbf{s}^{(i)}\big) < r$ as the *sphere constraint* (SC). We additionally employ *radius reduction*, which amounts to starting the algorithm with $r = \infty$ and updating the radius according to $r \leftarrow d_1\big(\mathbf{s}^{(1)}\big)$ whenever a leaf $\mathbf{s} = \mathbf{s}^{(1)}$ has been reached. This technique avoids the problem of choosing a suitable initial radius and still leads to efficient pruning of the tree. At the same time, it guarantees that the algorithm terminates only when the ML solution has been found.

In the remainder of the chapter, the computational complexity of SD is characterized by the *number of visited nodes* (including the root node but excluding the leaves), which was shown in [11, 12] to be closely related to the throughput of corresponding VLSI implementations.

Channel-Matrix Preprocessing A common approach to reduce the complexity of SD without compromising performance is to adapt the detection order of the spatial streams to the instantaneous channel realization by performing a QR-decomposition on $\mathbf{HP}$ (rather than $\mathbf{H}$), where $\mathbf{P}$ is a suitably chosen $M_\mathrm{T} \times M_\mathrm{T}$ permutation matrix. More efficient pruning of the tree is obtained if sorting is performed such that "stronger streams" (in terms of effective SNR) correspond to levels closer to the root, i.e., if $\mathbf{P}$ is chosen such that the main diagonal entries of $\mathbf{R}$ in $\mathbf{HP} = \mathbf{QR}$ are sorted in ascending order. An effective way to accomplish this goal was proposed in [16] and will be referred to as sorted QRD (SQRD) in the following.

An additional preprocessing method which further lowers the computational complexity, while slightly reducing the error-rate performance of SD is known as *regularization*, e.g., [11]. The main idea of regularization is to realize that poorly conditioned channel realizations $\mathbf{H}$ typically lead to high search complexity due to the low effective SNR on one or more of the effective spatial streams. An efficient way to counter ill-conditioned channel matrices is to operate on a *regularized* version of the channel matrix by computing the (sorted) QR-decomposition of

$$\begin{bmatrix} \mathbf{H} \\ \alpha \mathbf{I}_{M_\mathrm{T}} \end{bmatrix} \mathbf{P} = \mathbf{QR} \tag{6}$$

where α is a suitably chosen regularization parameter, $\mathbf{Q}$ is a $(M_\mathrm{R} + M_\mathrm{T}) \times M_\mathrm{T}$ matrix satisfying $\mathbf{Q}\mathbf{Q}^H = \mathbf{I}_{M_\mathrm{T}}$, and $\mathbf{R}$ is of dimension $M_\mathrm{T} \times M_\mathrm{T}$. By partitioning $\mathbf{Q}$ according to $\mathbf{Q} = [\, \mathbf{Q}_1^T \ \mathbf{Q}_2^T \,]^T$, where $\mathbf{Q}_1$ is of dimension $M_\mathrm{R} \times M_\mathrm{T}$ and $\mathbf{Q}_2$ is of dimension $M_\mathrm{T} \times M_\mathrm{T}$, the ML rule in (3) can be approximated as

$$\hat{\mathbf{s}} \approx \underset{\tilde{\mathbf{s}} \in \mathcal{O}^{M_\mathrm{T}}}{\arg\min} \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|^2 \tag{7}$$

where $\tilde{\mathbf{y}} = \mathbf{Q}_1^H \mathbf{y}$ and $\tilde{\mathbf{s}} = \mathbf{Ps}$. Setting the regularization parameter $\alpha = \sqrt{N_o M_\mathrm{T}}$ corresponds to MMSE regularization [17], which was shown in [11] to result in a good performance/complexity trade-off for hard- and soft-output SD algorithms.

2.3 Soft-Output Single Tree-Search Sphere Decoding

In coded MIMO systems, the computation of reliability information (i.e., soft-outputs) in the form of LLRs $L_{i,b}$ for each transmitted bit $x_{i,b}$ improves (often significantly) the error-rate performance compared to hard-output detection, which only computes binary-valued estimates for $x_{i,b}$.

Computation of the Max-Log LLRs Soft-output MIMO detection amounts to computing LLR-values $L_{i,b}$ for each transmitted bit $x_{i,b}$ according to [10, 11]

$$L_{i,b} = \log\left(\frac{\Pr[x_{i,b} = 1 \mid \mathbf{y}]}{\Pr[x_{i,b} = 0 \mid \mathbf{y}]} \right). \tag{8}$$

Straightforward computation of (8) results in prohibitive computational complexity. In order to reduce the complexity of LLR computation, we employ the *max-log approximation* [10, 11]

$$L_{i,b} \approx \min_{\mathbf{s} \in \mathcal{X}_{i,b}^{(0)}} \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 - \min_{\mathbf{s} \in \mathcal{X}_{i,b}^{(1)}} \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 \tag{9}$$

where $\mathcal{X}_{i,b}^{(0)}$ and $\mathcal{X}_{i,b}^{(1)}$ are the sets of symbol vectors that have the bth bit in the label of the jth scalar symbol equal to 0 and 1, respectively. We emphasize that the LLRs in (9) are *normalized* with respect to the noise variance N_o in order to get rid of the factor $1/N_0$ on the right hand side (RHS) of (9). This normalization simplifies the exposition and does *not* degrade the error rate performance with max-log-based channel decoders (see [11] for the details).

For each transmitted bit $x_{i,b}$, one of the two minima in (9) is given by the metric $\lambda^{\text{ML}} = \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}^{\text{ML}}\|^2$ associated with the ML solution $\mathbf{s}^{\text{ML}} = \hat{\mathbf{s}}$ of the MIMO detection problem in (3). The other minimum in (9) can be written as

$$\lambda_{i,b}^{\overline{\text{ML}}} = \min_{\mathbf{s} \in \mathcal{X}_{i,b}^{\overline{\text{ML}}}} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{10}$$

where $\mathcal{X}_{i,b}^{\overline{\text{ML}}}$ corresponds to the subset of $\mathcal{O}^{M_\text{T}}$ for which the (i,b)th bit is equal to the *counter-hypothesis* $\overline{x_{i,b}^{\text{ML}}}$, denoting the binary complement of the bth bit in the label of the ith entry of $\mathbf{s}^{\text{ML}}$. With (3) and (10) the max-log LLRs (9) can be re-written as

$$L_{i,b} \approx \begin{cases} \lambda^{\text{ML}} - \lambda_{i,b}^{\overline{\text{ML}}} \ , & x_{i,b}^{\text{ML}} = 0 \\ \lambda_{i,b}^{\overline{\text{ML}}} - \lambda^{\text{ML}} \ , & x_{i,b}^{\text{ML}} = 1 \ . \end{cases} \tag{11}$$

From (11), it is obvious that soft-output MIMO detection breaks down to efficiently identifying $\mathbf{s}^{\text{ML}}$, λ^{ML}, and $\lambda_{i,b}^{\overline{\text{ML}}}$ for $i = 1, \ldots, M_\text{T}$ and $b = 1, \ldots, Q$.

Single Tree-Search Sphere Decoding Computation of the max-log LLRs in (11) requires the metrics $\lambda_{i,b}^{\overline{\text{ML}}}$, which, for given i, b, is accomplished by traversing only those parts of the tree that have leaves in $\mathcal{X}_{i,b}^{\overline{\text{ML}}}$. Since this computation has to be carried out for every bit, it is immediately obvious that soft-output MIMO detection results in significantly higher computational complexity compared to hard-output ML detection using the SD algorithm. The soft-output STS-SD algorithm proposed in [11] is key in keeping the complexity increase (compared to hard-output SD) at a minimum and is summarized next.

The main idea of the soft-output STS-SD algorithm is to ensure that every node in the tree is visited at most once, which can be accomplished by searching for the ML solution and all counter-hypotheses *concurrently*. To this end, the algorithm searches the subtree originating from a given node only if the result can lead to an update of at least λ^{ML} or one of the $\lambda_{i,b}^{\overline{\text{ML}}}$. In the ensuing discussion, the bit-label vector of the current ML hypothesis and the corresponding metric are denoted by $\mathbf{x}^{\text{ML}}$ and λ^{ML}, respectively. The soft-output STS-SD algorithm consists of two main tasks, namely *list administration* and *tree pruning*:

List administration: The algorithm is initialized with $\lambda^{\text{ML}} = \lambda_{i,b}^{\overline{\text{ML}}} = \infty$, $\forall i, b$. Whenever a leaf with corresponding bit-label $\mathbf{x}$ has been reached, the algorithm distinguishes between two cases:

1) If a new ML hypothesis is found, i.e., if $d(\mathbf{x}) < \lambda^{\mathrm{ML}}$, then all $\lambda_{i,b}^{\overline{\mathrm{ML}}}$ for which $x_{i,b} = \overline{x_{i,b}^{\mathrm{ML}}}$ are set to λ^{ML} followed by the updates $\lambda^{\mathrm{ML}} \leftarrow d(\mathbf{x})$ and $\mathbf{x}^{\mathrm{ML}} \leftarrow \mathbf{x}$; this ensures that for each bit in the ML hypothesis that is changed in the process of the update, the metric of the *former* ML hypothesis becomes the metric of the *new* counter-hypothesis, followed by an update of the ML hypothesis.
2) In the case $d(\mathbf{x}) \geq \lambda^{\mathrm{ML}}$, only the counter-hypotheses need to be checked. Here, the decoder updates $\lambda_{i,b}^{\overline{\mathrm{ML}}} \leftarrow d(\mathbf{x})$ for all i and b satisfying $x_{i,b} = \overline{x_{i,b}^{\mathrm{ML}}}$ and $d(\mathbf{x}) < \lambda_{i,b}^{\overline{\mathrm{ML}}}$.

Tree pruning: The second key aspect of STS-SD is the following tree-pruning criterion: Consider a given node $\mathbf{s}^{(j)}$ (on level j) and the corresponding label $\mathbf{x}^{(j)}$ consisting of the bits $x_{i,b}$ ($i = j, \ldots, M_{\mathrm{T}}$, $b = 1, \ldots, Q$). Assume that the subtree originating from the node under consideration and corresponding to the bits $x_{i,b}$ ($i = 1, \ldots, j-1$, $b = 1, \ldots, Q$) has not been expanded yet. The pruning criterion for $\mathbf{s}^{(j)}$ along with its subtree is compiled from two conditions. First, the bits in the partial bit-label $\mathbf{x}^{(j)}$ associated to $\mathbf{s}^{(j)}$ are compared with the corresponding bits in the label of the current ML hypothesis $\mathbf{x}^{\mathrm{ML}}$. All metrics $\lambda_{i,b}^{\overline{\mathrm{ML}}}$ with $x_{i,b} = \overline{x_{i,b}^{\mathrm{ML}}}$ found in this comparison may be affected when searching the subtree of $\mathbf{s}^{(j)}$. Second, the metrics $\lambda_{i,b}^{\overline{\mathrm{ML}}}$ ($i = 1, \ldots, j-1$, $b = 1, \ldots, Q$) corresponding to the counter-hypotheses in the subtree of $\mathbf{s}^{(j)}$ may be affected as well. In summary, the metrics which may be affected during the search in the subtree emanating from the node $\mathbf{s}^{(j)}$ are given by the set

$$\mathcal{A}\big(\mathbf{x}^{(j)}\big) = \Big\{\lambda_{i,b}^{\overline{\mathrm{ML}}} \,\Big|\, \big(i \geq j, b = 1, \ldots, Q\big) \wedge \big(x_{i,b} = \overline{x_{i,b}^{\mathrm{ML}}}\big)\Big\} \\ \cup \Big\{\lambda_{i,b}^{\overline{\mathrm{ML}}} \,\Big|\, i < j, b = 1, \ldots, Q\Big\}.$$

The node $\mathbf{x}^{(j)}$ along with its subtree is pruned if its PED satisfies

$$d\big(\mathbf{x}^{(j)}\big) > \max_{a \in \mathcal{A}(\mathbf{x}^{(j)})} a\,. \tag{12}$$

The STS-SD pruning criterion ensures that a given node and the entire subtree originating from that node are explored only if this could lead to an update of either λ^{ML} or of at least one of the $\lambda_{i,b}^{\overline{\mathrm{ML}}}$, which enables significant complexity savings compared to other tree-search based soft-output MIMO detection algorithms, e.g., [10, 18].

LLR Clipping In practical systems, it is often desirable to tune the performance and complexity of the detection algorithm at run-time. LLR clipping [19] offers a convenient way to adjust this trade-off with the STS-SD algorithm. The key idea is to bound the dynamic range of the max-log LLRs so that

$$\big|L_{i,b}\big| \leq L_{\max} \quad \forall i, b \tag{13}$$

and to incorporate the clipping constraint (13) into the STS-SD algorithm. In particular, LLR clipping can be built into the algorithm by simply applying the update [11]

$$\lambda_{i,b}^{\overline{\mathrm{ML}}} \leftarrow \min\left\{\lambda_{i,b}^{\overline{\mathrm{ML}}}, \lambda^{\mathrm{ML}} + L_{\max}\right\} \tag{14}$$

to all counter-hypotheses, after carrying out the steps in Case 1) of the list administration procedure described above. The remaining steps of the STS-SD algorithm are not affected. For $L_{\max} = \infty$, STS-SD obviously delivers the exact max-log LLRs, whereas for $L_{\max} = 0$, we obtain hard-output SD performance as the decoder's output is $\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and $L_{i,b} = 0$ for all i and b. As shown in [11], the LLR-clipping parameter $L_{\max}$ can indeed be used to gracefully adjust the performance and complexity of the soft-output STS-SD algorithm.

3 Wide-band MIMO Receiver Architecture

We next show that for wide-band MIMO wireless communication systems, such as IEEE 802.11n or 3GPP-LTE, a single SD core turns out to be insufficient to simultaneously support the high bandwidth and the (error-rate) performance requirements, even when implementing the receiver in deep submicron CMOS technologies. We therefore present an architecture consisting of multiple SD-cores, which is able to meet the throughput and performance requirements of modern wide-band MIMO systems.

3.1 Run-Time Constraints

The computational complexity (required to find the ML solution or the LLR values) of the algorithms discussed above depends on the transmitted signals $\mathbf{s}$, the instantaneous realizations of the (random) channel matrix $\mathbf{H}$, and the noise vector $\mathbf{n}$. Consequently, the throughput of SD-based algorithms is variable and, in particular, random, which constitutes a significant problem in many practical application scenarios. Furthermore, the worst-case complexity of SD-based algorithms is equivalent to that of an exhaustive search [13]. Consequently, to meet the practically important requirement of a fixed throughput, the algorithm's run-time must be constrained. This, in turn, leads to a constraint on the maximum detection effort or, equivalently, to a constraint on the maximum number of nodes that the SD is allowed to visit, which will clearly prevent the detector from achieving ML performance or to be able to deliver the exact max-log LLRs in (9). It is therefore of paramount importance to find a way of imposing run-time constraints while keeping the resulting performance degradation at a minimum. Moreover, it is highly desirable in practice to have a smooth performance degradation as the run-time constraint becomes more stringent. Early-termination methods allowing for a smooth performance degradation suitable for narrow-band systems have been proposed in [11, 14]. The approach and architecture described next enables an efficient way to incorporate run-time constraints for wide-band MIMO wireless communication systems.

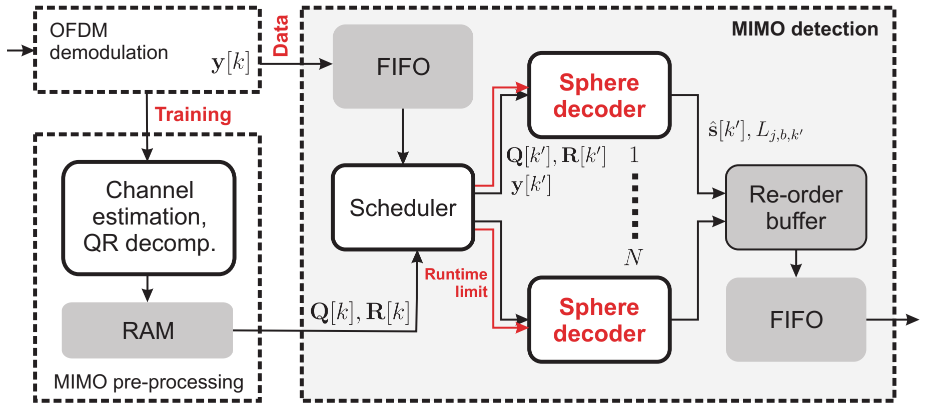


Fig. 3. High-level system architecture of a SD-based MIMO-OFDM receiver.

3.2 Receiver Architecture for Wide-band MIMO Systems

The high-level architecture of a wide-band MIMO receiver based on SD is illustrated in Fig. 3. The data flow starts with the OFDM demodulation. During a training phase, received training symbols are delivered to a MIMO preprocessing unit, which estimates the channel matrices $\mathbf{H}[k]$ for all OFDM tones and performs necessary pre-computations on $\mathbf{H}[k]$ (i.e., the sorted and regularized QRD). During the data phase, the demodulation unit and the MIMO preprocessing unit forward the received vectors $\mathbf{y}[k]$ and the results of the preprocessing unit to the MIMO detector at a constant arrival rate, which is essentially given by the communication bandwidth of the system. In the MIMO-detection block, the information required to decode a symbol is first queued in a FIFO buffer. A scheduler reads the entries of the FIFO buffer and forwards them to the next available SD core together with a runtime constraint (i.e., an upper limit on the number of nodes that are allowed to be examined by the SD). When the FIFO fills up, the runtime constraints are reduced to ensure that no data is lost. Note that this reduction of the maximum runtime degrades the quality of the detection.[1] The outputs from the N instantiated SD cores are collected and re-ordered since the variable runtime may cause decoded symbols to arrive out-of-order. The reordered symbol estimates (either hard- or soft-outputs) are finally forwarded to the interleaver and the channel decoder.

3.3 Implications on SD-Core Optimization

With the above described architecture, the average decoding effort, i.e., the number of visited nodes that can be allocated for decoding of each symbol is

[1] The particularities of the employed scheduling mechanism and the associated performance trade-offs can be found in [11].

determined by

$$\Phi \propto \frac{N}{T_{\mathrm{c}}B} \quad \text{[nodes]}$$

where B denotes the bandwidth of the system (i.e., the arrival-rate of the symbols to be decoded), T_{c} is the clock period of each SD core (assuming that one node in the tree is checked in each clock cycle), and N denotes the number of SD instances. At the system-level, the performance/complexity trade-off can now be adjusted by the choice of N, i.e., instantiating more SD cores improves the error-rate performance but clearly comes at the cost of increased silicon area. In particular, the resulting area of the presented architecture corresponds to $A_{\mathrm{tot}} = NA_{\mathrm{SD}}$, where A_{SD} denotes the silicon area of a single SD core.[2] For a large number of SD cores, the overall silicon area for a guaranteed number of visited nodes $\bar{\Phi}$ that can be used for decoding received symbols, is given by

$$A_{\mathrm{tot}} \propto \bar{\Phi} B \rho_{\mathrm{SD}} \quad \text{with} \quad \rho_{\mathrm{SD}} = T_{\mathrm{c}} A_{\mathrm{SD}}. \tag{15}$$

Consequently, it follows from (15) that *whenever* multiple SD cores are necessary to meet the performance and throughput requirements of a wide-band MIMO system, the focus for the optimization of the SD core shifts from minimizing the area or maximizing the throughput (as it is typically the case for narrow-band systems) to minimizing the corresponding *area-delay (AT-)product* ρ_{SD}. In the next two sections, we describe architectures for hard-output SD and soft-output STS-SD, which are optimized for minimum AT-product.

4 VLSI Architecture of Hard-Output SD

The hard-output SD architecture described next is based on the architecture template presented in [12]. We first summarize this architecture and then describe additional techniques that substantially improve (i.e., reduce) the associated AT-product.

4.1 High-Level Architecture

Fig. 4 shows the high-level block diagram of the proposed SD architecture. The design is comprised of a metric computation unit (MCU), a metric enumeration unit (MEU), an SC check unit, a level-select multiplexer, and a cache.

Metric computation unit: The MCU is responsible for the forward-iteration of the depth-first tree-traversal. In the implementation [12], this forward iteration includes the *sequential* evaluation of (5) and the computation of the PED in (4). In the present circuit (cf. Fig. 5) a slicer-unit performs a decision on the nearest constellation point and the MCU computes b_i (instead of b_{i+1}) in parallel to the PED of level $i+1$. The result b_i is then used in the next iteration (provided that the SC is met); this approach was shown to reduce the critical path of the SD-core without increasing the circuit area [20, 21].

[2] We neglect the area overhead in the FIFOs and re-order buffers.

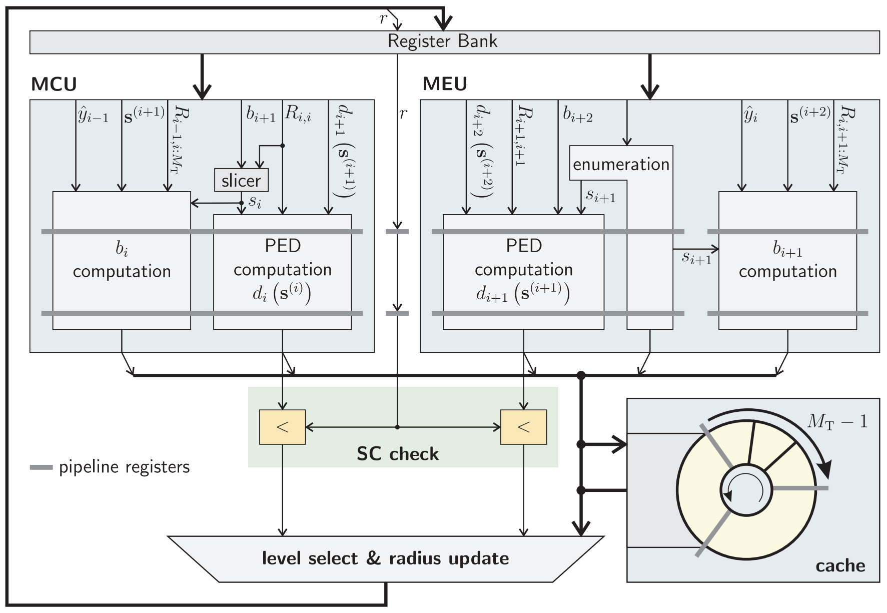


Fig. 4. High-level architecture of the hard-output SD unit. The shaded registers and the ring buffer (in the cache) are only required in when pipeline interleaving is used.

Metric enumeration unit: The MEU operates in parallel to the MCU. While the MCU processes a node on layer i, the MEU selects the next-best constellation point on layer $i+1$ according to the used enumeration scheme and computes its PED. Hence, once the SD algorithm needs to move upward in the tree (i.e., is performing backtracking), the MCU can directly start the next forward iteration as all required intermediate results have already been computed beforehand by the MEU. The register transfer-level (RTL) architecture of the MEU (cf. Fig. 5) is similar to the one of the MCU. However, the slicer-unit that determines the closest CP is replaced by an enumeration unit, which is used to determine the CP that is considered next on layer $i+1$.

Remaining units: The cache is used to store intermediate results for each level computed by the MEU and the MCU. The SC check is carried out immediately after the computation of the new PEDs. MEU, MCU, level cache, and the result of the SC check decide on which layer the SD algorithm proceeds next. If a valid leaf is found, i.e., whose metric fulfills the SC, the radius r is updated. In this case, an additional clock cycle is necessary, as the PEDs in the level cache need to be checked against the new radius.

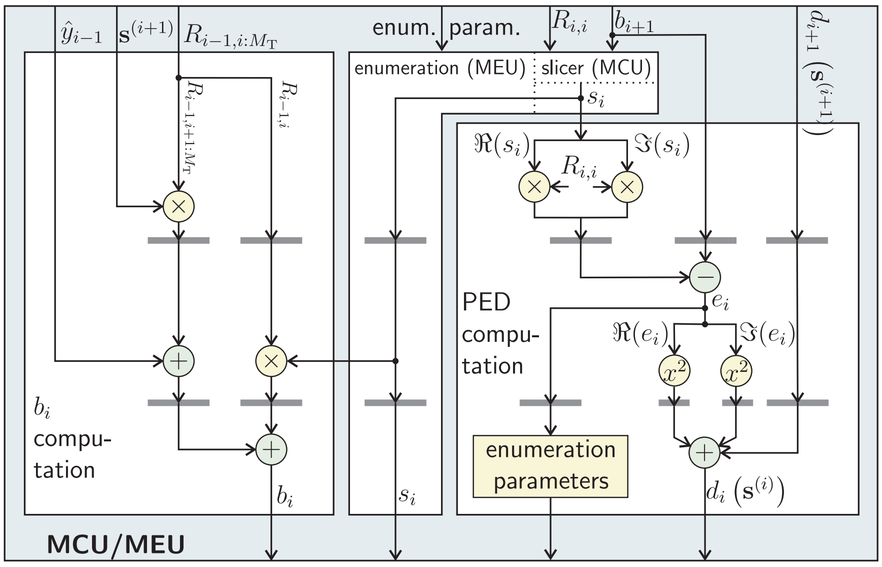


Fig. 5. RTL block diagram of the MCU and MEU. The shaded registers are only required when pipeline interleaving is applied.

4.2 Schnorr-Euchner Enumeration

The enumeration strategy (which is implemented by the *enumeration unit* in the MEU) defines the order in which the children of a node are examined. Radius reduction (cf. Section 2.2) is most efficient in combination with the SE enumeration [6,9], which visits the children of a node in ascending order of their PEDs. An important advantage of this enumeration strategy is that leaves that are more likely to lead to the ML solution (or corresponding counter-hypotheses for the STS-SD) are found early, which expedites the pruning of the tree. Moreover, enumeration of the children of a node can terminate as soon as a child violates the SC or, in the case of the STS-SD fulfills the corresponding pruning criterion.

For each visited node, SE enumeration comprises two types of operations: The first operation is to initialize the enumeration of the children by identifying the child associated with the smallest PED. This task can easily be accomplished by comparing b_{i+1} in (5) to a number of decision boundaries, i.e., by performing a slicing operation in the MCU shown in Fig. 4. The second type of operation is to enumerate the remaining children in ascending order of their PEDs, which is a non-trivial task for complex-valued constellations.[3] Unfortunately, the enumeration process has a significant impact on the complexity and on the critical path of SD implementations. Hence, reducing the complexity and critical path of the

[3] Note that for *real-valued* CPs, SE enumeration of the remaining child-nodes is immediately given by a zig-zag enumeration around the closest CP [9].

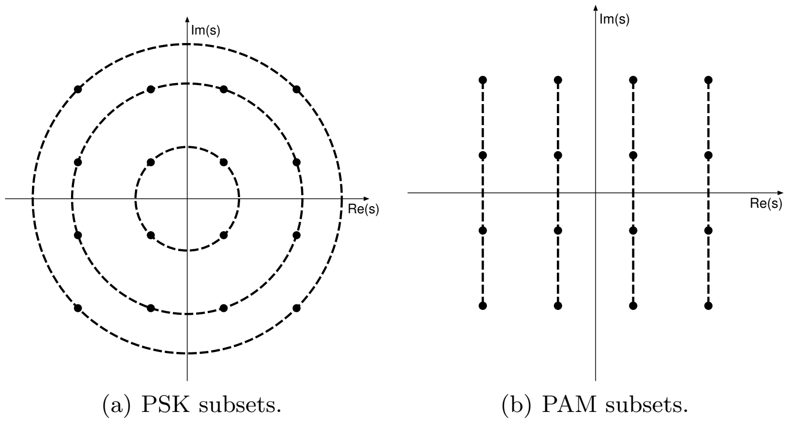


Fig. 6. The 16-QAM alphabet divided into one-dimensional subsets.

enumeration unit is *essential* to minimize the AT-product ρ_{SD} of corresponding efficient SD implementations.

Exhaustive Enumeration This method is a straightforward (but rather inefficient) solution to perform SE enumeration [12]. The idea is to first compute the PEDs of *all* children of a given node. During enumeration, a min-search (limited to the subset of children that have not yet been visited) identifies the next child. The main drawbacks of this solution are i) the area requirement to compute the PEDs of all children of a node, ii) the memory needed to store all PEDs in the cache, and iii) the fact that a min-search is costly in terms of area and timing, especially for higher order constellations. Hence, this approach is not suited for the efficient implementation of SD in hardware.

Subset Enumeration More elaborate solutions for SE enumeration were presented in [10,12,22]. The main idea of these approaches is to divide the complex-valued (i.e., two-dimensional) constellation into one-dimensional subsets, which only require to compute and store one PED per subset. The SE enumeration then chooses the child with the smallest PED among the preferred children in these subsets, which leads to a complexity reduction in the min-search stage and reduces the memory requirements in the cache.

Fig. 6 illustrates two subset enumeration schemes. For phase-shift keying (PSK) subsets proposed in [10] and [12], the constellation is decomposed into several concentric circles (see Fig. 6(a)). The second method shown in Fig. 6(b) was proposed in [22] and employs pulse-amplitude modulation (PAM) subsets (i.e., stripes). Both methods suffer from the fact that the number of required subsets becomes large when targeting higher modulation orders (e.g., 64-QAM requires eight PAM subsets), which contributes considerably to the resulting circuit area and timing of the entire architecture (as sorting across all subsets is required). In order to further reduce the complexity of SE enumeration, one needs to resort to *approximate* SE enumeration schemes such as the ones described next.

4.3 Approximate Schnorr-Euchner Enumeration Schemes

The goal of considering approximations to SE enumeration is to perform the candidate enumeration without the need for computing, caching, and comparing PEDs of multiple children of the same node. Such methods yield significant reduction in terms of circuit area and critical path delay at the cost of a (often negligible) reduction in terms of error-rate performance and are, therefore, well-suited to reduce the AT-product of corresponding SD implementations. The basic idea of most of these approaches [23, 24] is to store predefined enumeration sequences in one or multiple look-up tables (LUTs). A fixed sequence is chosen based upon several geometric rules that analyze the position of the received point b_{i+1} in the complex plane relative to the closest CP. The accuracy of these techniques (i.e., how closely they follow the Schnorr-Euchner enumeration sequence) can be adjusted by the number and complexity of the associated selection criteria together with the number of predefined LUTs.

Search-Sequence Determination This approach applies a few rules to the distance between the received point b_{i+1} and the closest CP denoted as a_i [23]. The number of rules applied to determine the position of b_{i+1} relative to the closest CP defines for how many nodes the resulting search sequence corresponds to the SE enumeration. For instance, the following first rule $\Re(a_i) \geq \Im(a_i)$ can determine the order of the first three nodes to be equal to the first three nodes of SE enumeration. Adding a second rule $1 - \Re(a_i) \geq 2\Im(a_i)$ allows to determine the SE enumeration order for the first four nodes. Each additional rule brings the search sequence closer to SE enumeration. However, in practice, a few rules or even only the first rule combined with enumeration of the remaining siblings according to a predefined order stored in look-up tables (LUTs) [24] often suffices to keep the performance loss negligible compared to SE enumeration.

Ordered $\ell_{\widetilde{\infty}}$-Norm Enumeration The approach implemented here is inspired by the $\ell_{\widetilde{\infty}}$-norm SD algorithm [12, 25]. Here, however, the $\ell_{\widetilde{\infty}}$-norm is only used for enumeration purposes, whereas the SD algorithm in [12, 25] also uses it for distance computations. The enumeration scheme initially proposed in [21] can be implemented efficiently without requiring LUTs and therefore, scales well to higher-order constellations (i.e., constellations including and beyond 64-QAM). The starting point for the enumeration is trivially determined by the closest CP (in terms of Euclidean distance). However, the subsequent CPs are enumerated according to their distance from b_{i+1} in terms of the $\ell_{\widetilde{\infty}}$-norm:

$$d_{\widetilde{\infty}} = |b_{i+1} - R_{i,i}s_i|_{\widetilde{\infty}} = \max\{|\Re(b_{i+1} - R_{i,i}s_i)|\,, |\Im(b_{i+1} - R_{i,i}s_i)|\}\,.$$

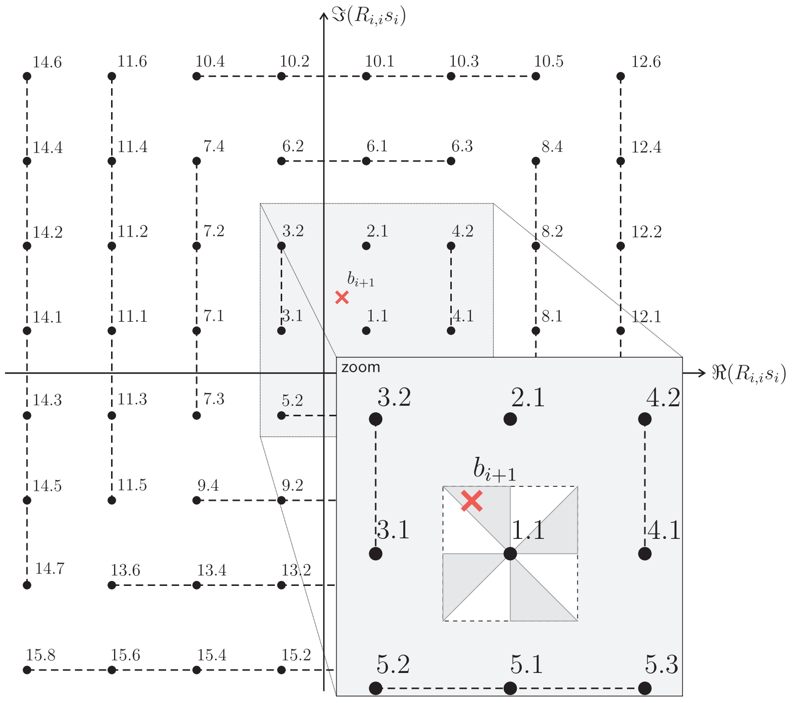
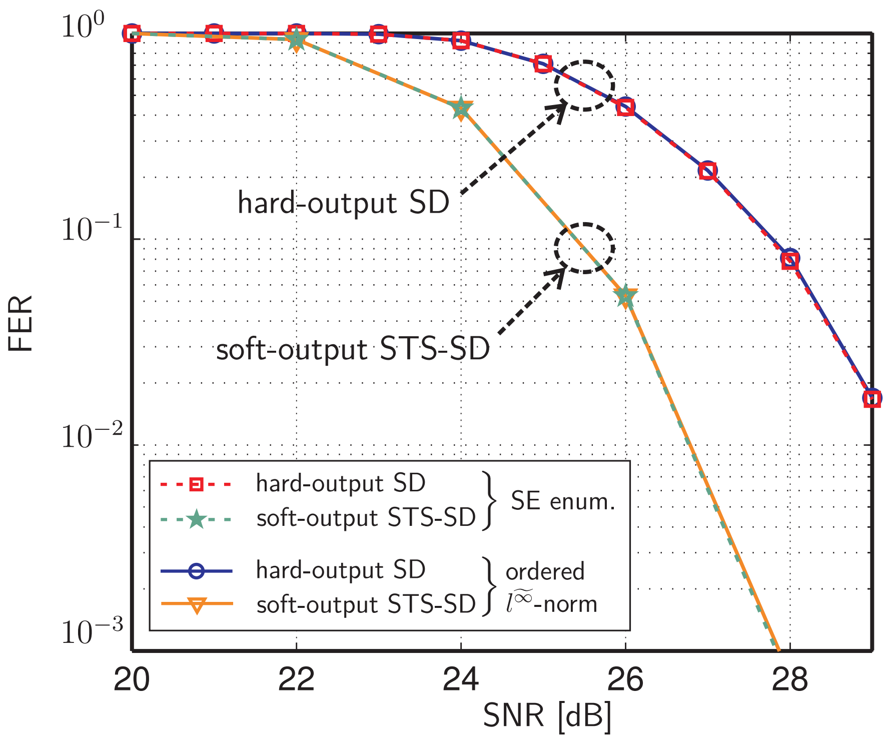


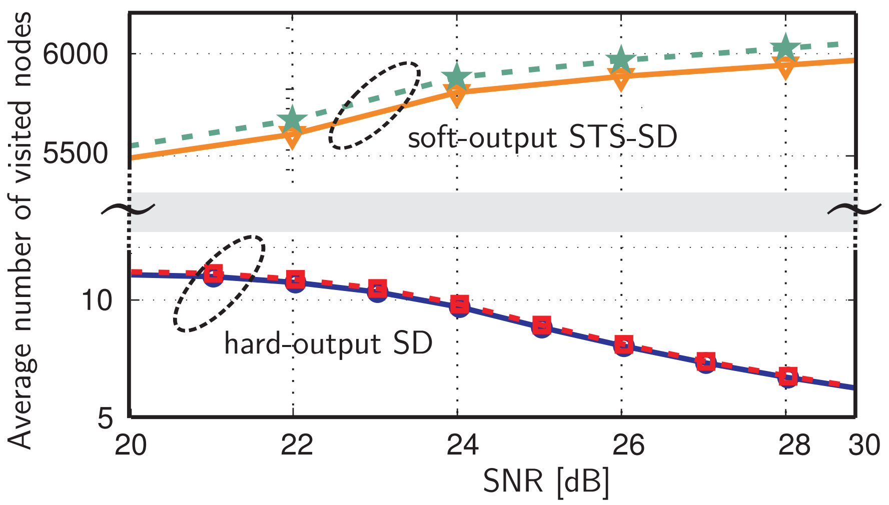
Fig. 7. Principle of ordered $\ell_{\widetilde{\infty}}$-norm enumeration for 64-QAM modulation.

To this end, the area around the closest CP is first subdivided into eight sectors as illustrated in the lower right corner of Fig. 7. The sector containing b_{i+1} is identified with simple geometric rules to define the second CP in the enumeration and the direction for the ordered $\ell_{\widetilde{\infty}}$-norm enumeration. CPs with identical $\ell_{\widetilde{\infty}}$-norm form one-dimensional subsets. All nodes within the same subset are processed before the algorithm selects the next subset. In the example provided in Fig. 7, the processing order of the one-dimensional subsets is illustrated by the leading number attached to each CP. Within each subset, zig-zag enumeration is applied around the CP closest to b_{i+1}, which is illustrated by the corresponding trailing number in Fig. 7. The members of each subset are returned in SE order and subsets are enumerated in order of increasing $\ell_{\widetilde{\infty}}$-norm.

Implementation: The above-described enumeration scheme can be split into two basic tasks: i) Tracking of the position, size, and orientation of the linear subsets and ii) zig-zag enumeration within the subsets and checking for the boundaries of the finite-size modulation alphabet. Both tasks can be implemented using simple combinational logic, comparators, and only three counters. Hence, the circuit complexity of ordered $\ell_{\widetilde{\infty}}$-norm enumeration is very low.



(a) Frame error-rate (FER) performance comparison.



(b) Computational complexity comparison.

Fig. 8. FER performance and computational complexity (in average number of visited nodes) for ordered $\ell_{\widetilde{\infty}}$-norm and SE enumeration ($M_T = M_R = 4$ using 64-QAM).

Impact on performance and complexity: Besides a reduction in terms of hardware complexity, the ordered $\ell_{\widetilde{\infty}}$-norm enumeration has an impact on the computational complexity (i.e., number of visited nodes) and on the (error-rate) performance of hard- and soft-output SD. The reason for this impact lies in the fact that the approximation does not guarantee that the children of a node are always enumerated strictly in ascending order of their PEDs (i.e., only the first three CPs always correspond to the first three CPs obtained by SE enumeration). In order to characterize the impact on performance and complexity, numerical simulations are carried out in order to verify that the loss in error-rate and increase in computational complexity is negligible. Corresponding simulation results[4] for hard- and soft-output SD are shown in Fig. 8. It can be observed that the loss in terms of the coded frame-error rate (FER) performance is negligible (see Fig. 8(a)) for both detection methods and the number of visited nodes with $\ell_{\widetilde{\infty}}$-norm enumeration is slightly smaller (i.e., approximately 5%) compared to that achieved by exact SE enumeration (see Fig. 8(b)). Hence, ordered $\ell_{\widetilde{\infty}}$-norm enumeration is well-suited for high-performance implementations of hard-output SD and soft-output STS-SD.

4.4 Pipeline Interleaving

Due to the first-order feedback path present in SD-architectures, pipelining cannot be applied in a straightforward way. Nevertheless, symbol-wise pipeline interleaving can be used to shorten the critical path and hence, to improve the AT-product of the SD-core implementation. The main idea of this approach applied to SD is to detect multiple (and independent) symbol-vectors in parallel within the same SD-unit [21, 27, 28].

Fig. 4 and Fig. 5 illustrate the location of the pipeline registers (in light grey boxes) in the VLSI architecture for three pipeline stages. The locations of the pipeline registers were chosen manually in order to balance the path delays between each pipeline stage. Furthermore, automated retiming was used during synthesis for further optimization. Besides adding the pipeline registers in the data-path, the level cache in Fig. 4 required the implementation of a ring-buffer, in which each set of entries is associated with one of the symbols in the pipeline and corresponds to one instance of the original level cache. Note that the number of pipeline stages affects the throughput and silicon area of the detector. A corresponding investigation of the resulting area/throughput trade-off is provided in Section 6.

[4] We consider coded (rate 2/3 convolutional code, constraint length 7, generator polynomials $[133_o\ 171_o]$, and random interleaving across space and frequency) MIMO-OFDM transmission with $M_\mathrm{R} = M_\mathrm{T} = 4$, 64-QAM (Gray mapping), 64 OFDM tones. One frame consists of 1536 coded bits. A TGn type C [26] channel model is used. We assume perfect channel state information at the receiver and employ minimum mean-square error sorted QR decomposition (MMSE-SQRD) [17] for SD-preprocessing. The SNR is per receive antenna.

5 VLSI Architecture of Soft-Output STS-SD

The high-level block diagram of the soft-output STS-SD implementation is shown in Fig. 9. Compared to the architecture for hard-output SD described in Section 4, modifications are necessary in the MCU and two additional units are required, one for list administration and one for the implementation of the STS-SD pruning criterion [11]. We next describe the specifics of these changes.

5.1 Architectural Changes in the MCU

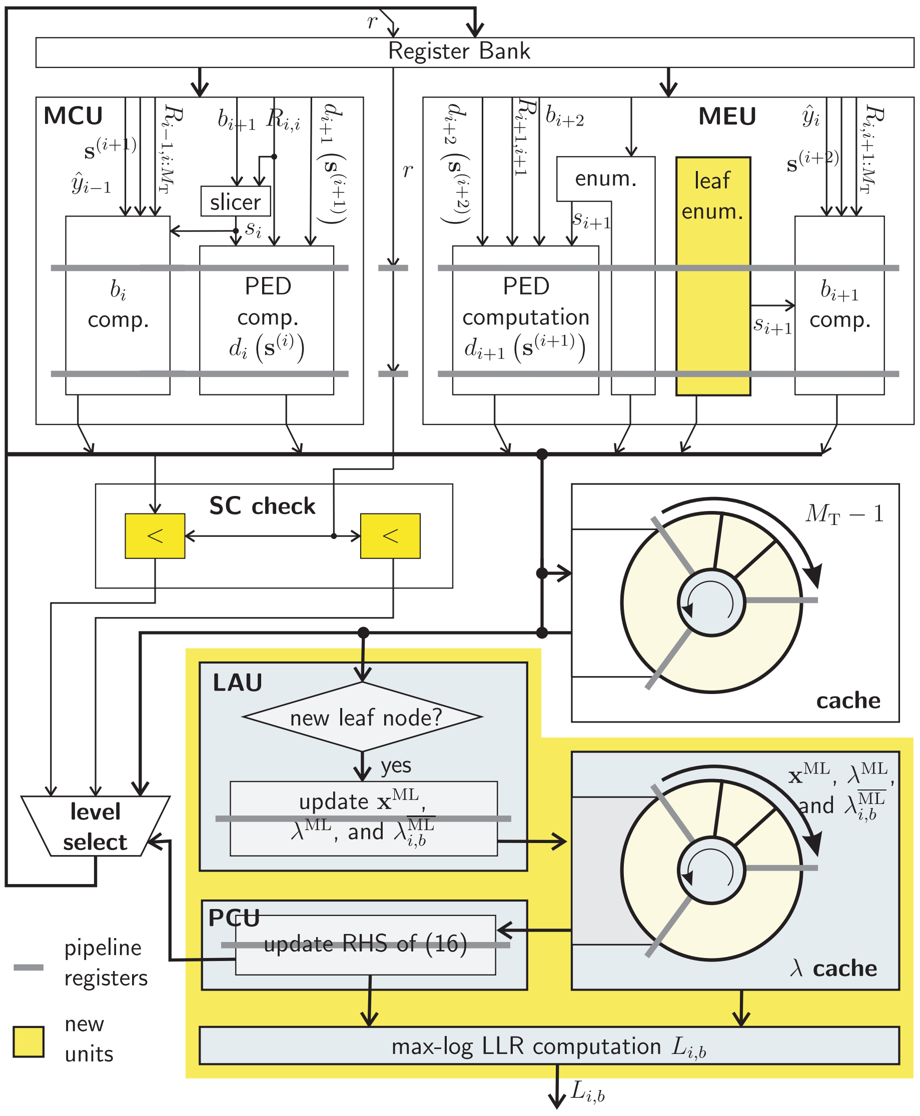
From a high-level perspective, there is one fundamental difference between tree-traversal for hard-output SD and for the soft-output STS-SD algorithm: When the node currently examined by the MCU is on the level just above the leaves (i.e., on level $i = 2$), the hard-output SD algorithm considers only one child, namely the one associated with the smallest PED. The STS-SD algorithm, however, has to compute the PEDs of all children that do not qualify for pruning according to the criterion (12) since these children may lead to updates of the metrics $\lambda_{i,b}^{\overline{\mathrm{ML}}}$. To perform this *leaf enumeration* procedure, STS-SD must revisit the current node at level $i = 2$, which requires additional clock cycles and a leaf enumeration unit shown in Fig. 9. This unit does, however, not require an additional arithmetic unit for the PED computation as it can reuse the PED computation unit in the MCU (see Fig. 9).

The computational complexity involved in this leaf-enumeration approach can be reduced significantly, by taking advantage of the Gray mapping of the information bits for the constellation symbols [21]. The leaf nodes of interest to the computation of max-log LLR values, are obtained by flipping every bit of the leaf that is closest to b_2. Furthermore, by considering the distance differences between the constellation symbols, it can be shown that no costly squaring operations are necessary. The data path able to carry out these computations merely encompasses a shifter, an adder and a multiplication.

5.2 List Administration and Tree Pruning

In addition to the modifications in the MCU described above, the soft-output STS-ST algorithm requires two additional units [11]:

List-Administration Unit (LAU) The LAU is responsible for maintaining and updating the list containing $\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and the $\lambda_{i,b}^{\overline{\mathrm{ML}}}$. The corresponding unit is active during the leaf-enumeration process described above. Since the update rules implemented by the LAU require only a small number of logic operations, the silicon area of this unit is small and is dominated by the storage space (λ cache) required for the metrics λ^{ML} and $\lambda_{i,b}^{\overline{\mathrm{ML}}}$. This cache needs to provide storage for all the metrics of all symbols being processed in parallel by pipeline interleaving.



r
Register Bank
MCU
$\mathbf{s}^{(i+1)}$
$\hat{y}_{i-1}$
$R_{i-1,i:M_T}$
b_{i+1}
$R_{i,i}$
$d_{i+1}\left(\mathbf{s}^{(i+1)}\right)$
slicer
s_i
b_i comp.
PED comp. $d_i\left(\mathbf{s}^{(i)}\right)$
r
$d_{i+2}\left(\mathbf{s}^{(i+2)}\right)$
$R_{i+1,i+1}$
b_{i+2}
MEU
$\hat{y}_i$
$\mathbf{s}^{(i+2)}$
$R_{i,i+1:M_T}$
enum.
s_{i+1}
leaf enum.
PED computation $d_{i+1}\left(\mathbf{s}^{(i+1)}\right)$
s_{i+1}
b_{i+1} comp.
SC check
<
<
$M_T - 1$
cache
LAU
new leaf node?
yes
update $\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and $\lambda^{\overline{\mathrm{ML}}}_{i,b}$
$\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and $\lambda^{\overline{\mathrm{ML}}}_{i,b}$
level select
PCU
update RHS of (16)
λ cache
pipeline registers
new units
max-log LLR computation $L_{i,b}$
$L_{i,b}$

Fig. 9. Block diagram of the proposed VLSI architecture for the soft-output STS-SD. Additional required units (compared to hard-output SD) are highlighted.

Pruning Criterion Unit (PCU) The PCU is responsible for computing the RHS of (12). From an implementation perspective, the reference metric on level j depending on the partial label $\mathbf{x}^{(j)}$ constitutes a major problem. More specifically, this dependence causes the criterion for pruning the child of a parent node on level $j+1$ to depend on the partial label $\mathbf{x}^{(j)}$ of that child. This, in turn, implies that enumeration of the children on level j in ascending order of their PEDs according to the SE criterion cannot be applied, which results in the need for exhaustive-search enumeration (see Section 4.2). As mentioned above, exhaustive enumeration is ill-suited for the efficient implementation in VLSI (cf. [12]). A modification of the pruning criterion in (12) proposed in [11] solves this problem. To this end, define

$$\mathcal{B}\Big(\mathbf{x}^{(j+1)}\Big) = \Big\{\lambda_{i,b}^{\overline{\mathrm{ML}}} \,\big|\, \big(i > j, b = 1, \ldots, Q\big) \wedge \big(x_{i,b} = \overline{x_{i,b}^{\mathrm{ML}}}\big)\Big\} \\ \cup \Big\{\lambda_{i,b}^{\overline{\mathrm{ML}}} \,\big|\, i \le j, b = 1, \ldots, Q\Big\}$$

and prune the node $\mathbf{x}^{(j)}$ (corresponding to the partial symbol vector $\mathbf{s}^{(j)}$) along with its subtree if $d\Big(\mathbf{x}^{(j)}\Big)$ satisfies

$$d\Big(\mathbf{x}^{(j)}\Big) > \max_{b \in \mathcal{B}\left(\mathbf{x}^{(j+1)}\right)} b\,. \qquad (16)$$

Note that compared to (12), the RHS of the modified pruning criterion (16) depends on the partial label $\mathbf{x}^{(j+1)}$ rather than on $\mathbf{x}^{(j)}$. Consequently, the enumeration of the children of a node on level $j+1$ can be carried out using SE enumeration or an approximation thereof (see Section 4.2 and Section 4.3).

The approach described above entails a slight increase in terms of complexity compared to the original pruning criterion for the STS-SD algorithm in (12). Nevertheless, the corresponding complexity increase is significantly smaller than what would be incurred if (12) would be applied directly. A corresponding detailed discussion can be found in [11].

6 Implementation Results and Comparison

In the following, we present implementation results for hard-output SD and soft-output STS-SD in a 130 nm CMOS technology. Furthermore, a comparison to existing hard- and soft-output SD implementations demonstrates the performance advantage of the AT-product-optimized VLSI architectures detailed in this chapter.

6.1 Implementation Results for Hard-Output SD

The AT-diagram in Fig. 10 shows the synthesis results of hard-output SD with ordered $\ell_{\widetilde{\infty}}$-norm enumeration and pipeline interleaving with different number of pipeline stages.[5] The proposed architectures have been implemented with support for multiple modulation schemes (BPSK, QPSK, 16-QAM, and 64-QAM) and for up to four spatial streams (configurable at runtime).

Fig. 10 shows that the architecture with three pipeline stages achieves the best AT-product. Nevertheless, the architectures with more than three pipeline stages come close to the one achieving the optimal AT product, whereas the architectures with fewer pipeline stages are clearly outperformed in terms of the AT-product. As a comparison, implementation results of previously reported hard-output SD implementations are also included in Fig. 10 (the results are also summarized in Tbl. 1). It can be seen that the proposed hard-output SD implementation without pipelining already outperforms all existing designs without pipelining by a least 23% in terms of area and by at least 28% in terms of clock frequency[6]. Furthermore, the AT-product (in [kGE/MHz]) of the proposed architecture with pipeline interleaving is more than 2× better than that of the pipelined implementation in [27].

6.2 Implementation Results for Soft-Output STS-SD

Ordered $\ell_{\widetilde{\infty}}$-norm enumeration and pipeline interleaving can also be applied to the soft-output STS-SD architecture described in Section 4. Corresponding implementation results for soft-output STS-SD are shown in Tbl. 2 and are compared to existing soft-output SD implementations [11,24]. The AT-optimized implementation is superior in terms of area and clock frequency compared to the soft-output detector described in [24]. Note that the original implementation of soft-output STS-SD in [11] only supports 16-QAM modulation, which is the main reason for the smaller area in the unpipelined case. For hard-output SD, pipeline interleaving with three pipeline stages appears to be optimal in terms of the AT-product. As the additional units required for soft-output STS-SD do not influence the critical path, STS-SD was also implemented with three pipeline stages. Tbl. 2 shows that pipeline interleaving also improves the AT-product for soft-output SD implementations and yields a gain of more than 30% compared to the unpipelined design.

6.3 The Case for Multiple SD-Cores

In Section 2, we argued that a single SD core is insufficient to meet the bandwidth and error-rate performance requirements of modern wireless communication standards such as IEEE 802.11n, where a throughput of 600 Mb/s is required. From Tbl. 1, we observe that a single instance of hard-output SD meets the throughput requirement when early-termination and block-processing according to [11, 14] are applied. For soft-output STS-SD, however, the number of visited nodes is significantly increased: From seven for hard-output SD to a

[5] The results were obtained by synthesizing the RTL description in VHDL with different timing constraints.

[6] The clock frequencies of all designs are normalized to 130 nm CMOS technology.

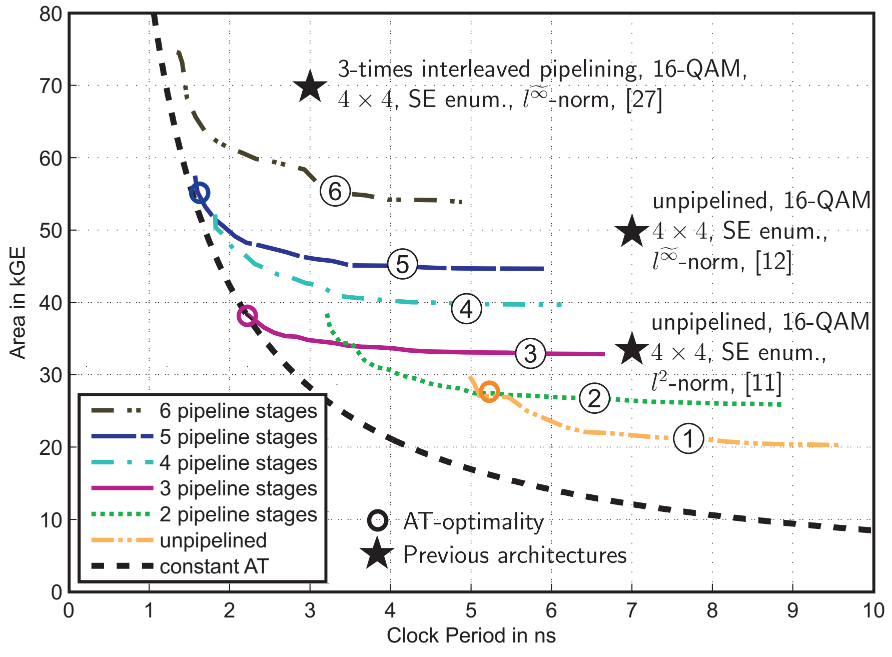


Fig. 10. AT-diagram of hard-output SD with different number of pipeline stages. The optimal synthesis results (in terms of the area/delay trade-off) are highlighted by circles and implementation results of previous architectures are indicated by stars. All designs are scaled to 130 nm CMOS technology.

least 100 for soft-output STS-SD [11]. To illustrate the necessity for multiple soft-output STS-SD cores, we assume LLR clipping is used to adjust the average complexity to $D_{\text{avg}} = 100$ for 64-QAM modulation which typically results only in a negligible error rate performance degradation. In this case, the throughput of one STS-SD core is 92 Mb/s and therefore, in order to meet the throughput requirement of IEEE 802.11n, seven AT-optimized soft-output STS-SD cores are required.

7 Summary and Conclusion

In order to meet the throughput and latency requirements of modern wideband wireless communication systems, such as IEEE 802.11n or 3GPP-LTE, using the sphere decoding (SD) algorithm, multiple parallel detection cores are necessary. Therefore, the main optimization goal for each SD core is to minimize the area-delay product, which represents the hardware-efficiency. Ordered $\ell^{\widetilde{\infty}}$-norm enumeration and pipeline interleaving are two key techniques that are both suitable to achieve this goal. The approximate enumeration strategy significantly reduces circuit area and the critical path-delay and corresponding simulations

Table 1. Implementation results and comparison of hard-output SD.

	[12]	[11]	[27]	This work		
CMOS tech.	250 nm	250 nm	130 nm	130 nm		
Antennas	4×4	4×4	4×4	1×1 to 4×4		
Modulation	16-QAM	16-QAM	16-QAM	BPSK to 64-QAM		
Norm	$\ell_{\widetilde{\infty}}$	ℓ_2	$\ell_{\widetilde{\infty}}$	ℓ_2		
Enumeration	SE	SE	SE	ordered $\ell_{\widetilde{\infty}}$-norm		
Pipeline stages	no	no	3×	no	3×	5×
Area[a] [kGE]	50	34.4	70	27.1	38.4	55.3
Freq. [MHz]	137[b]	140[b]	333	196	455	625
[kGE/MHz]	0.37	0.25	0.21	0.14	0.08	0.09
Throughput for $D_{\text{avg}} = 7$[c] [Mb/s]	470	480	1141	672	1560	2143

Table 2. Implementation results and comparison of soft-output algorithms.

	[24]	[11]	This work	
CMOS Technology	130 nm	250 nm	130 nm	
Modulation	64-QAM	16-QAM	BPSK to 64-QAM	
Algorithm	MBF-FD	STS-SD	STS-SD	
Enumeration	tabular	SE	ordered $\ell_{\widetilde{\infty}}$-norm	
Pipeline stages	no	no	no	3×
Area[a] [kGE]	350	56.8	70.4	97.1
Max. frequency [MHz]	198	137[b]	183	383
AT-product [kGE/MHz]	1.77	0.41	0.38	0.25

[a]One GE corresponds to the area of a two-input drive-one NAND gate.

[b]Scaled from 250 nm to a 130 nm CMOS technology by multiplying with 250/130.

[c]D_{avg} denotes the average number of nodes used for block processing [11, 14].

show, that the performance loss is negligible. With pipeline interleaving, the critical path of each SD core can significantly be reduced, which additionally improves the AT-product. A design-space exploration with different number of pipeline stages reveals that an architecture with three pipeline stages (for hard-output and soft-output SD) is the most efficient in terms of the AT-product and should, therefore, be preferred for implementation.

Acknowledgments. The authors would like to thank L. Bruderer, H. Friederich, and P. Luethi for their contributions during the design and implementation of the SD architectures. Furthermore, the authors gratefully acknowledge the support from H. Bölcskei, N. Felber, H. Kaeslin, and W. Fichtner. This work was partially supported by the STREP project MASCOT (IST-026905) within the Sixth Framework of the European Commission and by the Swiss National Science Foundation (SNSF) Grants PP002-119052 and PA00P2-134155.

References

1. A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications.* Cambridge Univ. Press, 2003.
2. H. Bölcskei, D. Gesbert, C. Papadias, and A. J. van der Veen, Eds., *Space-Time Wireless Systems: From Array Processing to MIMO Communications.* Cambridge Univ. Press, 2006.
3. *IEEE Draft Standard; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications; Amendment 4: Enhancements for Higher Throughput*, P802.11n/D3.0, Sep. 2007.
4. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 9)*, 3GPP Organizational Partners TS 36.212, Rev. 8.3.0, May 2008.
5. U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Computation*, vol. 44, no. 170, pp. 463–471, Apr. 1985.
6. C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, no. 2, pp. 181–191, Sep. 1994.
7. E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," *Colloq. GRETSI*, vol. 14, pp. 611–614, Sep. 1993.
8. E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.
9. E. Agrell, T. Eriksson, A. Vardy, and K. Z. r, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.
10. B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.
11. C. Studer, A. Burg, and H. Bölcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.
12. A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.
13. C. Studer, D. Seethaler, and H. Bölcskei, "Finite lattice-size effects in MIMO detection," in *Proc. of 42th Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, CA, USA, Oct. 2008.
14. A. Burg, M. Borgmann, M. Wenk, C. Studer, and H. Bölcskei, "Advanced receiver algorithms for MIMO wireless communications," in *Proc. of DATE*, Mar. 2006, pp. 593–598.
15. C. Studer, "Iterative MIMO decoding: Algorithms and VLSI implementation aspects," Ph.D. dissertation, ETH Zürich, Switzerland, Series in Microelectronics, vol. 202, Hartung-Gorre Verlag Konstanz, 2009.
16. D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K.-D. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEE Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

17. D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE 58th VTC*, Oct. 2003, pp. 508–512.
18. R. Wang and G. B. Giannakis, "Approaching MIMO channel capacity with reduced-complexity soft sphere decoding," in *Proc. of IEEE Wireless Communications and Networking Conf. (WCNC)*, vol. 3, Mar. 2004, pp. 1620–1625.
19. M. S. Yee, "Max-Log-Map sphere decoder," in *Proc. IEEE ICASSP 2005*, vol. 3, Mar. 2005, pp. 1013–1016.
20. E. M. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr, "A scalable VLSI architecture for soft-input soft-output depth-first sphere decoding," 2009, available online at http://arxiv.org/abs/0910.3427.
21. M. Wenk, L. Bruderer, A. Burg, and C. Studer, "Area- and throughput-optimized VLSI architecture of sphere decoding," in *Proc. of IEEE/IFIP Int. Conf. on VLSI and System-on-Chip (VLSI-SoC), Sept. 2010*, Madrid, Spain, Sept. 2010.
22. C. Hess, M. Wenk, A. Burg, P. Luethi, C. Studer, N. Felber, and W. Fichtner, "Reduced-complexity MIMO detector with close-to ML error rate performance," in *Proc. 17th ACM Great Lakes Symposium on VLSI*, 2007, pp. 200–203.
23. B. Mennenga and G. Fettweis, "Search sequence determination for tree search based detection algorithms," in *IEEE Sarnoff Symposium*, Apr. 2009, pp. 1–6.
24. L. Chun-Hao, W. To-Ping, and C. Tzi-Dar, "A 74.8 mW soft-output detector IC for 8×8 spatial-multiplexing MIMO communications," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 411–421, Feb. 2010.
25. D. Seethaler and H. Bölcskei, "Performance and complexity analysis of infinity-norm sphere-decoding," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 1085–1105, Mar. 2010.
26. V. Erceg and et al., *TGn channel models.* IEEE 802.11-03/940r4, May 2004.
27. A. Burg, M. Wenk, and W. Fichtner, "VLSI implementation of pipelined sphere decoding with early termination," in *Proc. EUSIPCO*, Sep. 2006, invited paper.
28. J. Lee, S.-C. Park, and S. Park, "A pipelined VLSI architecture for a list sphere decoder," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS'06)*, Sep. 2006, pp. 397–400.