Soft-Output Sphere Decoding: Algorithms and VLSI Implementation

Christoph Studer, *Student Member, IEEE*, Andreas Burg, *Member, IEEE*, and Helmut Bölcskei, *Senior Member, IEEE*

***Abstract*—Multiple-input multiple-output (MIMO) detection algorithms providing soft information for a subsequent channel decoder pose significant implementation challenges due to their high computational complexity. In this paper, we show how sphere decoding can be used as an efficient tool to implement soft-output MIMO detection with flexible trade-offs between computational complexity and (error rate) performance. In particular, we provide VLSI implementation results which demonstrate that single tree-search, sorted QR-decomposition, channel matrix regularization, log-likelihood ratio clipping, and imposing run-time constraints are the key ingredients for realizing soft-output MIMO detectors with near max-log performance at a chip area that is only 58% higher than that of the best-known hard-output sphere decoder VLSI implementation.**

***Index Terms*—Multiple-input multiple-output (MIMO) communication systems, soft-output sphere decoding, VLSI implementation, MIMO detection.**

I. Introduction

MULTIPLE-input multiple-output (MIMO) wireless systems employ multiple antennas on both sides of the wireless link and offer increased spectral efficiency (compared to single-antenna systems) by transmitting multiple data streams concurrently and in the same frequency band (spatial multiplexing). MIMO technology constitutes the basis for upcoming wireless communication standards, such as IEEE 802.11n and IEEE 802.16e.

The main challenge in the practical realization of MIMO wireless systems lies in the efficient implementation of the detector which needs to separate the spatially multiplexed data streams. To this end, a wide range of algorithms offering various trade-offs between performance and computational complexity have been developed [2]. Linear detection producing hard outputs constitutes one extreme of the complexity/performance trade-off region, while computationally demanding maximum-likelihood (ML) detection algorithms in combination with exact a posteriori probability (APP) computation result in the opposite extreme. In general, the computational complexity of a MIMO detection algorithm depends on the symbol constellation size and the number of spatially multiplexed data streams, but often also on the instantaneous MIMO channel realization and the signal-to-noise ratio (SNR). On the other hand, the overall decoding effort is typically constrained by the system bandwidth, latency requirements, and the quest to keep chip area and power consumption as low as possible. Implementing different algorithms, each optimized for a maximum allowed decoding effort and/or a particular system configuration, would entail considerable chip area overhead and in addition be highly inefficient since large portions of the chip would remain idle most of the time. A practical MIMO receiver design should therefore be able to cover a wide range of complexity/performance trade-offs using a single tunable detection algorithm.

Contributions: In this paper, we describe a tunable MIMO detector based on the sphere decoder [3]–[8], with performance ranging from that of hard-output successive interference cancellation (SIC) [9] to that of max-log APP detection [10]. Tuning of the detector is achieved through log-likelihood ratio (LLR) clipping, channel matrix regularization, and imposing constraints on the maximum computational complexity of the decoder (i.e., run-time constraints). With a view towards VLSI implementation, we elaborate on, and provide refinements of, the tree-search algorithm outlined in [11] leading to what we term the single tree-search (STS) approach. We describe how LLR clipping as proposed in [12] can be incorporated into the STS algorithm. A framework for systematically characterizing the complexity/performance trade-offs of the resulting class of soft-output sphere decoders is formulated. Finally, we present a suitable VLSI architecture and provide reference implementation results for max-log soft-output sphere decoding with LLR clipping.

Notation: Matrices are set in boldface capital letters, vectors in boldface lowercase letters. The superscripts ^{T} and ^{H} stand for transpose and conjugate transposition, respectively. We write $A_{i,j}$ for the entry in the ith row and jth column of the matrix $\mathbf{A}$ and b_i for the ith entry of the vector $\mathbf{b} = [\, b_1 \; b_2 \; \cdots \; b_N \,]^T$. $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. Slightly abusing common terminology, we call an $N \times M$ matrix $\mathbf{A}$, where $N \geq M$, satisfying $\mathbf{A}^H\mathbf{A} = \mathbf{I}_M$, unitary. $|\mathcal{A}|$ denotes the cardinality of the set $\mathcal{A}$. $\mathbb{E}[\cdot]$ stands for the expectation operator. The binary complement of x is denoted by $\overline{x}$.

Outline of the Paper: The remainder of this paper is organized as follows. Section II reviews the transformation of the max-log soft-output MIMO detection problem into

Manuscript received April 7, 2007; revised October 31, 2007. This paper was presented in part in [1] at the 40th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, Nov. 2006. This work was supported in part by the STREP project No. IST-026905 (MASCOT) within the Sixth Framework Programme (FP6) of the European Commission.

C. Studer is with the Integrated Systems Laboratory of the ETH Zurich, 8092 Zurich, Switzerland (e-mail: studer@iis.ee.ethz.ch). A. Burg and H. Bölcskei are with the Communication Technology Laboratory of the ETH Zurich, 8092 Zurich, Switzerland (e-mail: apburg@iis.ee.ethz.ch; boelcskei@nari.ee.ethz.ch).

Digital Object Identifier 10.1109/JSAC.2008.0802xx.

a series of tree-search problems. In Section III, we review the repeated tree-search (RTS) algorithm proposed in [13] and introduce the STS algorithm. In Section IV, we describe methods for reducing the tree-search complexity both in the RTS and the STS algorithms. A framework for evaluating the complexity/performance trade-offs of the resulting family of soft-output sphere decoders is introduced in Section V. In Section VI, we describe a VLSI architecture for the efficient implementation of max-log soft-output sphere decoding with LLR clipping. Corresponding ASIC implementation results are summarized in Section VII. We conclude in Section VIII.

II. Soft-Output Sphere Decoding

Consider a MIMO system with M_T transmit and $M_R \geq M_T$ receive antennas. The coded bit-stream is mapped to M_T-dimensional transmit symbol vectors $\mathbf{s} \in \mathcal{O}^{M_T}$, where $\mathcal{O}$ stands for the set of underlying complex-valued scalar constellation points with $|\mathcal{O}| = 2^Q$. Each symbol vector $\mathbf{s}$ is associated with a bit-level label vector $\mathbf{x}$ where, throughout the paper, symbol vectors and their associated labels will be used interchangeably. Slightly deviating from our notation rules, we denote the entries of $\mathbf{x}$ as $x_{j,b}$, where the indices j and b refer to the bth bit in the label of the constellation point corresponding to the jth entry of $\mathbf{s} = \begin{bmatrix} s_1 & s_2 & \cdots & s_{M_T} \end{bmatrix}^T$. The resulting complex baseband input-output relation is given by

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

where $\mathbf{H}$ denotes the $M_R \times M_T$ channel matrix and $\mathbf{n}$ is an i.i.d. zero-mean proper complex Gaussian distributed M_R-dimensional noise vector with variance N_o per complex entry. Throughout this paper, we assume that the receiver has perfect knowledge of the channel matrix realization. The SNR per receive antenna is $1/N_o$.

A. Computation of the Max-Log LLRs

Soft-output MIMO detection requires the computation of LLRs, denoted as $L(\cdot)$, for all bits in the label $\mathbf{x}$. In order to reduce the corresponding computational complexity, we employ the *max-log approximation* [10], [14]

$$L(x_{j,b}) = \min_{\mathbf{s} \in \mathcal{X}_{j,b}^{(0)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 - \min_{\mathbf{s} \in \mathcal{X}_{j,b}^{(1)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{2}$$

where $\mathcal{X}_{j,b}^{(0)}$ and $\mathcal{X}_{j,b}^{(1)}$ 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. Note that we do not take into account a priori information. The max-log approximation entails a performance loss compared to using the exact LLRs. For the simulation setup considered in Section V, this loss, in terms of SNR, is found to be around 0.25 dB over a large range of SNRs. We furthermore emphasize that the LLRs in (2) are normalized by the noise variance N_o in order to get rid of the factor $1/N_o$ on the right hand side (RHS) of (2). This simplifies the exposition and does not degrade the error rate performance of the max-log-based soft-input Viterbi decoder considered in Section V.

For each bit, one of the two minima in (2) is given by the metric $\lambda^{\text{ML}} = \|\mathbf{y} - \mathbf{H}\mathbf{s}^{\text{ML}}\|^2$ associated with the ML solution of the MIMO detection problem

$$\mathbf{s}^{\text{ML}} = \arg\min_{\mathbf{s} \in \mathcal{O}^{M_T}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2. \tag{3}$$

The other minimum in (2) can be written as

$$\lambda_{j,b}^{\overline{\text{ML}}} = \min_{\mathbf{s} \in \mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{4}$$

where the *counter-hypothesis* $\overline{x_{j,b}^{\text{ML}}}$ denotes the binary complement of the bth bit in the label of the jth entry of $\mathbf{s}^{\text{ML}}$. With (3) and (4) the max-log LLRs can be written as

$$L(x_{j,b}) = \begin{cases} \lambda^{\text{ML}} - \lambda_{j,b}^{\overline{\text{ML}}} & , \quad x_{j,b}^{\text{ML}} = 0 \\ \lambda_{j,b}^{\overline{\text{ML}}} - \lambda^{\text{ML}} & , \quad x_{j,b}^{\text{ML}} = 1 \end{cases}. \tag{5}$$

From (5) we can conclude that efficient max-log APP MIMO detection reduces to efficiently identifying $\mathbf{s}^{\text{ML}}$, λ^{ML}, and $\lambda_{j,b}^{\overline{\text{ML}}}$ for $j = 1, 2, \ldots, M_T$ and $b = 1, 2, \ldots, Q$ [13].

B. Max-Log APP MIMO Detection as a Tree Search

Transforming (3) and (4) into tree-search problems and using the sphere decoding algorithm [3]–[8] allows to efficiently compute the LLRs in (5). To this end, the channel matrix $\mathbf{H}$ is first QR-decomposed according to $\mathbf{H} = \mathbf{Q}\mathbf{R}$, where the $M_R \times M_T$ matrix $\mathbf{Q}$ is unitary, and the $M_T \times M_T$ upper-triangular matrix $\mathbf{R}$ has real-valued positive entries on its main diagonal. Left-multiplying (1) by $\mathbf{Q}^H$ leads to the modified input-output relation

$$\tilde{\mathbf{y}} = \mathbf{R}\mathbf{s} + \mathbf{Q}^H\mathbf{n} \quad \text{with} \quad \tilde{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$$

and hence, noting that $\mathbf{Q}^H\mathbf{n}$ has the same statistics as $\mathbf{n}$, to the equivalent characterization of λ^{ML} and $\lambda_{j,b}^{\overline{\text{ML}}}$ as

$$\lambda^{\text{ML}} = \min_{\mathbf{s} \in \mathcal{O}^{M_T}} \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{6}$$

$$\lambda_{j,b}^{\overline{\text{ML}}} = \min_{\mathbf{s} \in \mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}} \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2. \tag{7}$$

We next define the partial symbol vectors (PSVs) $\mathbf{s}^{(i)} = \begin{bmatrix} s_i & s_{i+1} & \cdots & s_{M_T} \end{bmatrix}^T$ and note that they can be arranged in a tree that has its root just above level $i = M_T$ and leaves, on level $i = 1$, which correspond to symbol vectors $\mathbf{s}$. In the following, the label associated with $\mathbf{s}^{(i)}$ is denoted by $\mathbf{x}^{(i)}$. The Euclidean distances $d(\mathbf{s}) = \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$ in (6) and (7) can be computed recursively as $d(\mathbf{s}) = d_1$ with the partial Euclidean distances (PEDs)

$$d_i = d_{i+1} + |e_i|^2, \quad i = M_T, M_T - 1, \ldots, 1\,, \tag{8}$$

the initialization $d_{M_T+1} = 0$, and the distance increments (DIs)

$$|e_i|^2 = \left| \tilde{y}_i - \sum_{j=i}^{M_T} R_{i,j} s_j \right|^2. \tag{9}$$

Since the dependence of the PED d_i on the symbol vector $\mathbf{s}$ is only through the PSV $\mathbf{s}^{(i)}$, we have transformed ML detection *and* the computation of the max-log LLRs into a weighted tree-search problem: PSVs and PEDs are associated with *nodes*, *branches* correspond to DIs. For brevity, we shall often say "the node $\mathbf{s}^{(i)}$" to refer to the node corresponding to the PSV $\mathbf{s}^{(i)}$. Each path from the root down to a leaf corresponds to a symbol vector $\mathbf{s} \in \mathcal{O}^{M_T}$. The solution of (6) and (7) corresponds to the leaf associated with the smallest metric in $\mathcal{O}^{M_T}$ and $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}$, respectively. The basic building block underlying the two tree traversal strategies described in the next section is the Schnorr-Euchner (SE) sphere decoder (SESD) with radius reduction [15], [16], briefly summarized as follows: The search in the tree is constrained to nodes which lie within a radius r around $\tilde{\mathbf{y}}$ and tree traversal is performed depth-first, visiting the children of a given node in ascending order of their PEDs. The basic idea of radius reduction is to start the algorithm with $r = \infty$ and to update the radius according to $r^2 \leftarrow d(\mathbf{s})$ whenever a leaf $\mathbf{s}$ has been reached. This avoids the problem of choosing a suitable initial radius and still leads to efficient pruning of the tree.

III. Tree-Traversal Strategies

Computing the LLRs in (5) requires determining the metrics $\lambda_{j,b}^{\overline{\text{ML}}}$, which, for given j, b, is accomplished by traversing only those parts of the tree that have leaves in $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}$. Since this computation has to be carried out for every bit, it is immediately obvious that LLR computation results in an order of magnitude increase in computational complexity compared to hard-output sphere decoding. The situation is further exacerbated by the fact that forcing the SESD into subtrees, when computing the minima in (7) leads to significantly less efficient tree pruning behavior, which finally results in an overall complexity increase (over hard-output SESD) of two orders of magnitude. The STS algorithm introduced below is key in reducing this computational complexity.

In the following, we discuss two tree traversal strategies for solving (6) and (7). The first approach described below was introduced in [13] and will be referred to as repeated tree search (RTS). The second algorithm builds on a tree traversal strategy outlined in [11]. With a view towards VLSI implementation, we propose refinements of the approach in [11] resulting in what we call the single tree-search (STS) strategy.

A. Repeated Tree Search (RTS)

The basic idea of the RTS algorithm described in [13] is to start by solving (6) (using the SESD) and to then rerun the SESD to solve (7) for each bit (i.e., QM_T times) in the symbol vector. When rerunning the SESD to determine $\lambda_{j,b}^{\overline{\text{ML}}}$ in (7), the search tree is prepruned by forcing the decoder to exclude all nodes from the search for which $x_{j,b} = x_{j,b}^{\text{ML}}$. For BPSK, this prepruning procedure is illustrated in Fig. 1. Following the proposal in [13] and initializing the SESD with $r = \infty$ in each of the QM_T runs required to obtain $\lambda_{j,b}^{\overline{\text{ML}}}$, will lead to significant computational complexity. It is therefore important to realize that (without compromising max-log optimality) the search radius $r_{j,b}$ can be initialized by setting it equal to the minimum value of $\|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|$ over all $\mathbf{s} \in \mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}$ found during preceding tree traversals.

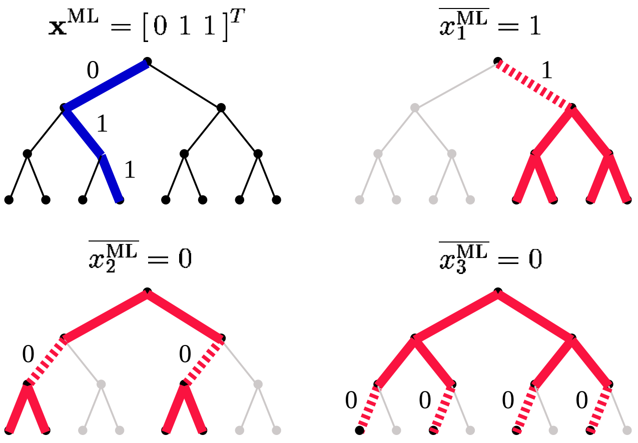
$\mathbf{x}^{\text{ML}} = [0\ 1\ 1]^T$
0
1
1
$\overline{x_1^{\text{ML}}} = 1$
1
$\overline{x_2^{\text{ML}}} = 0$
0
0
$\overline{x_3^{\text{ML}}} = 0$
0
0
0
0

Fig. 1. Example, assuming a BPSK-constellation, of the prepruning procedure in the RTS approach. Counter-hypotheses to the ML solution are found by forcing the sphere decoder through the dashed branches.

The main disadvantages of the RTS are:

i) the repeated traversal of large parts of the tree which entails a large number of redundant computations;
ii) significantly less efficient pruning behavior when computing the $\lambda_{j,b}^{\overline{\text{ML}}}$ caused by the need to minimize over the subsets $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\text{ML}}}\right)}$. The underlying reason is that pruning efficiency decreases significantly when forcing the sphere decoder through specific branches at levels further down the tree.

As noted in [17], the problem in ii) can partly be mitigated by changing the detection order in each run. The resulting need for multiple QR decompositions, however, leads to an overall increase in terms of hardware complexity.

B. Single Tree Search

The key to more efficient (compared to RTS) tree traversal is to ensure that every node in the tree is visited at most once. This can be accomplished by searching for the ML solution *and* all counter-hypotheses concurrently. The basic idea behind such an approach has been outlined in [11]. In the following, with a view towards VLSI implementation, we provide refinements of the idea in [11]. Specifically, we formulate update rules and a pruning criterion based on a list containing the metric λ^{ML}, the corresponding label $\mathbf{x}^{\text{ML}}$, and the metrics $\lambda_{j,b}^{\overline{\text{ML}}}$. The main idea is to search the subtree originating from a given node only if the result can lead to an update of at least one of the metrics in the list, i.e., either λ^{ML} or one of the $\lambda_{j,b}^{\overline{\text{ML}}}$. In the ensuing discussion, the current ML hypothesis and the corresponding metric are denoted by $\mathbf{x}^{\text{ML}}$ and λ^{ML}, respectively.

1) List Administration: The algorithm is initialized with $\lambda^{\text{ML}} = \lambda_{j,b}^{\overline{\text{ML}}} = \infty$ ($\forall\ j, b$). Whenever a leaf with corresponding label $\mathbf{x}$ has been reached, the decoder distinguishes between two cases:

i) If a new ML hypothesis is found, i.e., $d(\mathbf{x}) < \lambda^{\text{ML}}$, all $\lambda_{j,b}^{\overline{\text{ML}}}$ for which $x_{j,b} = \overline{x_{j,b}^{\text{ML}}}$ are set to λ^{ML} followed by the updates $\lambda^{\mathrm{ML}} \leftarrow d(\mathbf{x})$ and $\mathbf{x}^{\mathrm{ML}} \leftarrow \mathbf{x}$. In other words, 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. This procedure ensures that at all times $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ is the metric associated with a valid counter-hypothesis to the current ML hypothesis.

ii) In the case where $d(\mathbf{x}) \geq \lambda^{\mathrm{ML}}$, only the counter-hypotheses have to be checked. For all j and b such that $x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}$ and $d(\mathbf{x}) < \lambda_{j,b}^{\overline{\mathrm{ML}}}$, the decoder updates $\lambda_{j,b}^{\overline{\mathrm{ML}}} \leftarrow d(\mathbf{x})$.

2) Pruning criterion: The second key aspect of the STS algorithm is the following tree pruning criterion. Consider a given node $\mathbf{s}^{(i)}$ (on level i) and the corresponding partial label $\mathbf{x}^{(i)}$ consisting of the bits $x_{j,b}$ ($j = i, i+1, \ldots, M_T$, $b = 1, 2, \ldots, Q$). Assume that the subtree originating from the node under consideration and corresponding to the bits $x_{j,b}$ ($j = 1, 2, \ldots, i-1,\ b = 1, 2, \ldots, Q$) has not been expanded yet. The pruning criterion for $\mathbf{s}^{(i)}$ along with its subtree is compiled from two conditions. First, the bits in the partial label $\mathbf{x}^{(i)}$ are compared with the corresponding bits in the label of the current ML hypothesis $\mathbf{x}^{\mathrm{ML}}$. All metrics $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ with $x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}$ found in this comparison may be affected when searching the subtree of $\mathbf{s}^{(i)}$. Second, the metrics $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ ($j = 1, 2, \ldots, i-1$, $b = 1, 2, \ldots, Q$) corresponding to the counter-hypotheses in the subtree of $\mathbf{s}^{(i)}$ may be affected as well. In summary, the metrics which may be affected during the search in the subtree emanating from the node $\mathbf{s}^{(i)}$ are given by the set

$$\begin{aligned}\mathcal{A}\Big(\mathbf{x}^{(i)}\Big) &= \{a_l\} = \\ &= \Big\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\big|\, \big(j \geq i, b = 1, 2, \ldots, Q\big) \wedge \big(x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}\big)\Big\} \\ &\quad \cup \Big\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\big|\, j < i, b = 1, 2, \ldots, Q\Big\}.\end{aligned}$$

The node $\mathbf{s}^{(i)}$ along with its subtree is pruned if its PED $d\Big(\mathbf{s}^{(i)}\Big)$ satisfies

$$d\Big(\mathbf{s}^{(i)}\Big) > \max_{a_l \in \mathcal{A}\left(\mathbf{x}^{(i)}\right)} a_l\,. \tag{10}$$

This pruning criterion (illustrated in Fig. 2) 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 $\lambda_{j,b}^{\overline{\mathrm{ML}}}$. Note that λ^{ML} does not appear in $\mathcal{A}\big(\mathbf{x}^{(i)}\big)$ as $\lambda^{\mathrm{ML}} < \lambda_{j,b}^{\overline{\mathrm{ML}}}$ ($\forall\, j, b$).

IV. Methods for Complexity Reduction

So far, we discussed strategies which solve (2) exactly and hence do not compromise performance of the max-log APP decoder. The goal of this section is to describe methods, again with a view towards VLSI implementation, that allow to trade-off decoder complexity with (error rate) performance.

The complexity measure employed throughout this paper is the number of nodes (including the leaves, but excluding the root) visited by the decoder. In Section VI, we show that this simple complexity measure provides a good indication for the complexity of a corresponding VLSI implementation.

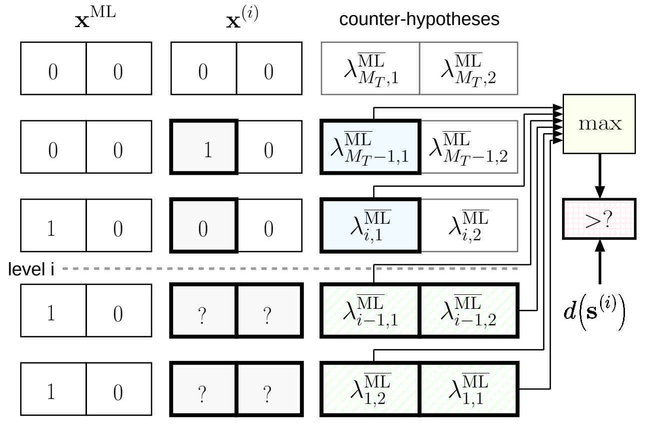
$\mathbf{x}^{\mathrm{ML}}$
$\mathbf{x}^{(i)}$
counter-hypotheses
0 0 0 0
$\lambda_{M_T,1}^{\overline{\mathrm{ML}}}$ $\lambda_{M_T,2}^{\overline{\mathrm{ML}}}$
0 0 1 0
$\lambda_{M_T-1,1}^{\overline{\mathrm{ML}}}$ $\lambda_{M_T-1,2}^{\overline{\mathrm{ML}}}$
1 0 0 0
$\lambda_{i,1}^{\overline{\mathrm{ML}}}$ $\lambda_{i,2}^{\overline{\mathrm{ML}}}$
level i
1 0 ? ?
$\lambda_{i-1,1}^{\overline{\mathrm{ML}}}$ $\lambda_{i-1,2}^{\overline{\mathrm{ML}}}$
1 0 ? ?
$\lambda_{1,2}^{\overline{\mathrm{ML}}}$ $\lambda_{1,1}^{\overline{\mathrm{ML}}}$
max
>?
$d\big(\mathbf{s}^{(i)}\big)$

Fig. 2. Example of the STS pruning criterion ($M_T = 5$ and two bits per symbol): The partial label $\mathbf{x}^{(i)}$ determines which counter-hypotheses may be affected during the search of the subtree emanating from the current node.

A. LLR Clipping

The dynamic range of LLRs is typically not bounded. However, practical systems need to constrain the magnitude of the LLR values to enable fixed-point implementation. Evidently this will lead to a performance degradation. A straightforward way of ensuring that LLR values are bounded is to clip them *after* the detection stage so that

$$\big|L(x_{j,b})\big| \leq L_{\max}, \ \ \forall j, b\,. \tag{11}$$

We emphasize that the constraint in (11) refers to the normalized LLRs $L(x_{j,b})$ as defined in (2) so that $L_{\max}$ is a *normalized* maximum LLR value.

a) LLR Clipping for RTS: It has been noted in [12] that (11) can be built into the RTS algorithm as a constraint leading to a reduction in search complexity. The basic idea is to recognize that (5) together with (11) results in an upper bound on the radius $r_{j,b}$ (as illustrated in Fig. 3). To this end, $r_{j,b}$ is initialized as described in Section III-A followed by an immediate update according to

$$r_{j,b} \leftarrow \min\left\{r_{j,b}, \lambda^{\mathrm{ML}} + L_{\max}\right\} \tag{12}$$

which ensures that (11) is satisfied. Note that as a consequence of (12), metrics associated with counter-hypotheses for which no valid lattice point is found equal $\lambda^{\mathrm{ML}} + L_{\max}$.

b) LLR Clipping for STS: LLR clipping can be built into the STS algorithm by simply applying the update

$$\lambda_{j,b}^{\overline{\mathrm{ML}}} \leftarrow \min\Big\{\lambda_{j,b}^{\overline{\mathrm{ML}}}, \lambda^{\mathrm{ML}} + L_{\max}\Big\}, \ \ \forall j, b \tag{13}$$

after carrying out the steps in Case i) of the list administration procedure described in Section III-B. The remaining steps of the STS algorithm are not affected.

Both in the RTS and the STS algorithm, for $L_{\max} = \infty$, we obviously get the exact max-log LLRs, whereas for $L_{\max} = 0$, we obtain hard-output SESD performance as the decoder output is $\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and $L(x_{j,b}) = 0$ for all j and b. Smaller values of $L_{\max}$ lead to more aggressive pruning of the tree and hence to reduced search complexity. We shall see in Section V that as we reduce $L_{\max}$, the decoder performance

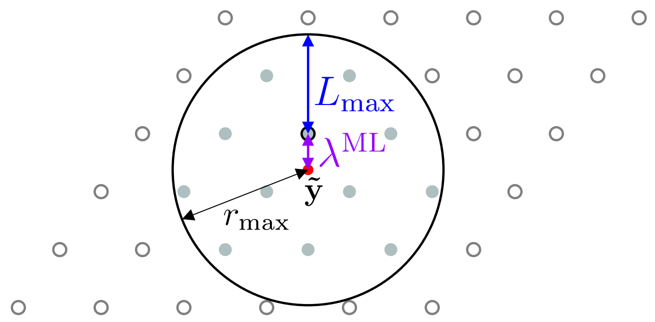

Fig. 3. LLR clipping reduces the search radius to $r_{\max} = \lambda^{\mathrm{ML}} + L_{\max}$ around the received point $\tilde{\mathbf{y}}$.

degrades gracefully, eventually resulting in hard-output ML performance. The parameter $L_{\max}$ can therefore be used to efficiently adjust the detection complexity/performance trade-off. We conclude by noting that LLR clipping as described above goes beyond the initial motivation of constraining the word-width used to represent LLR values in binary logic.

B. Sorting and Regularization

Sorting: A common approach to reduce complexity in sphere decoding 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 $M_T \times M_T$ permutation matrix. More efficient pruning of the search 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. Solving this problem exactly would result in prohibitive complexity. A heuristic algorithm resulting in a good complexity/performance trade-off was proposed in [18] and will be referred to as sorted QR-decomposition (SQRD) in the following.

Regularization: Poorly conditioned channel realizations $\mathbf{H}$ 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 this problem is to operate on a *regularized* channel matrix by computing the (sorted) QR-decomposition of

$$\begin{bmatrix} \mathbf{H} \\ \alpha \mathbf{I}_{M_T} \end{bmatrix} \mathbf{P} = \mathbf{QR} \tag{14}$$

where α is a suitably chosen regularization parameter, $\mathbf{Q}$ is a unitary $(M_R + M_T) \times M_T$ matrix and $\mathbf{R}$ is of dimension $M_T \times M_T$. Partitioning $\mathbf{Q}$ according to $\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1^T \ \mathbf{Q}_2^T \end{bmatrix}^T$, where $\mathbf{Q}_1$ is of dimension $M_R \times M_T$ and $\mathbf{Q}_2$ is of dimension $M_T \times M_T$, the max-log LLRs in (2) can be approximated as

$$L(x_{j,b}) \approx \min_{\tilde{\mathbf{s}} \in \mathcal{X}_{j,b}^{(0)}} \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|^2 - \min_{\tilde{\mathbf{s}} \in \mathcal{X}_{j,b}^{(1)}} \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|^2 \tag{15}$$

where $\tilde{\mathbf{y}} = \mathbf{Q}_1^H \mathbf{y}$ and $\tilde{\mathbf{s}} = \mathbf{Ps}$. The LLRs in (15) need to be reordered at the end of the detection process to account for the permutation induced by $\mathbf{P}$.

Note that even though $\mathbf{Q}$ is unitary, $\mathbf{Q}_1$ will, in general, not be unitary, which is the reason for the LLRs in (15) being an approximation to the exact (max-log) LLRs in (2). The basic idea underlying this approximation is to perform the QR-decomposition on the regularized channel matrix and to apply the result to the physical channel matrix. In the following, we provide a qualitative discussion of the error incurred by this procedure. We start by noting that, as a consequence of (14), we have

$$\tilde{\mathbf{y}} = \mathbf{R}\tilde{\mathbf{s}} + \tilde{\mathbf{n}} \tag{16}$$

with the effective noise-plus-(self)-interference (NPI) vector

$$\tilde{\mathbf{n}} = -\alpha \mathbf{Q}_2^H \mathbf{s} + \mathbf{Q}_1^H \mathbf{n}.$$

Eq. (16) shows that the approximation in (15) amounts to pretending that $\tilde{\mathbf{n}}$ is i.i.d. circularly symmetric complex Gaussian thereby neglecting that i) $\tilde{\mathbf{n}}$ depends on $\mathbf{s}$, ii) the self-interference term $-\alpha \mathbf{Q}_2^H \mathbf{s}$ is not Gaussian (as we are using finite constellations), and iii) $\mathbf{Q}_1$ is, in general, not unitary which results in $\mathbf{Q}_1^H \mathbf{n}$ not being i.i.d. Nevertheless, computing the covariance matrix of $\tilde{\mathbf{n}}$, by averaging over $\mathbf{n}$ and $\mathbf{s}$, allows to identify good choices for the regularization parameter α. Assuming, for simplicity of exposition, that $\mathbb{E}[\mathbf{ss}^H] = \frac{1}{M_T} \mathbf{I}_{M_T}$, straightforward manipulations reveal that

$$\mathbf{K} = \mathbb{E}\left[\tilde{\mathbf{n}}\tilde{\mathbf{n}}^H\right] = \left(\mathbf{RR}^H\right)^{-1} |\alpha|^2 \left(\frac{|\alpha|^2}{M_T} - N_o\right) + N_o \mathbf{I}_{M_T}.$$

Setting $\alpha = \pm\sqrt{N_o M_T}$ corresponds to an MMSE regularization [19], results in $\mathbf{K} = N_o \mathbf{I}_{M_T}$, and yields a good performance/complexity trade-off. We emphasize, however, that setting $\alpha = \pm\sqrt{N_o M_T}$ will not render the effective NPI vector $\tilde{\mathbf{n}}$ Gaussian. In the remainder of the paper, we denote the QR-decomposition in (14) with $\alpha = \pm\sqrt{N_o M_T}$ as MMSE-SQRD. An important practical aspect of MMSE-SQRD results from the fact that the noise variance N_o has to be estimated. We found that, in general, even slight overestimation of N_o will lead to a noticeable performance degradation, whereas slight underestimation does not seem to constitute a problem.

C. Run-Time Constraints

The computational complexity (required to find the ML solution and the LLR values) of the algorithms discussed so far depends on the realization of the random channel matrix as well as on the noise realization. Consequently, the decoder throughput is variable, which constitutes a problem in many practical application scenarios. Moreover, the worst-case complexity corresponds to an exhaustive search. In order to meet the practically important requirement of a fixed throughput, the algorithm run-time must be constrained. This, in turn, leads to a constraint on the maximum detection effort or, equivalently, a constraint on the maximum number of nodes the sphere decoder is allowed to visit. Clearly, this will, in general, prevent the detector from achieving ML or max-log APP performance. It is therefore important to find a way of imposing run-time constraints while keeping the resulting performance degradation at a minimum. Moreover,

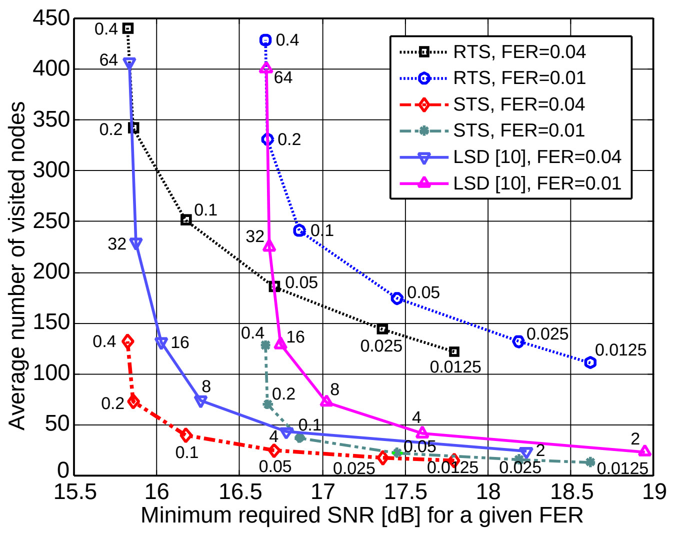

Fig. 4. Comparison of repeated tree search (RTS), single tree search (STS), and the list sphere decoder (LSD) as proposed in [10], all using SQRD preprocessing. The numbers next to the curves correspond to $L_{\max}$ for RTS and STS and to the list size in the case of the LSD.

in practice, it is highly desirable to have a smooth performance degradation as the run-time constraint becomes more stringent.

In the following, we restrict ourselves to the STS algorithm. A straightforward way of enforcing a run-time constraint is to terminate the search, on a symbol vector by symbol vector basis, after a maximum number of visited nodes. The STS decoder then returns the best solution found so far, i.e., the current ML and counter-hypotheses. A better solution is to impose an aggregate run-time constraint of ND_{avg} visited nodes for an entire block of N symbol vectors[1] [20]. The maximum number of visited nodes allocated to the detection of the kth symbol vector can, for example, be chosen according to the maximum-first (MF) scheduling strategy as

$$D_{\max}(k) = ND_{\text{avg}} - \sum_{i=1}^{k-1} D(i) - (N-k)M_T \tag{17}$$

for $k = 1, 2, \ldots, N$ where $D(i)$ denotes the actual number of visited nodes for the ith symbol vector. The concept behind (17) is that decoding a given symbol vector is allowed to consume all of the remaining run-time within the block of N symbol vectors up to a safety margin of $(N-k)M_T$ visited nodes. This margin allows to find at least the hard-output SIC solution for all remaining symbol vectors. Setting $D_{\text{avg}} = M_T$ maximizes the throughput, but reduces the performance to that of hard-output SIC. We emphasize that, under run-time constraints, there may be LLRs at the end of the decoding process that have not been updated from their initial value of ∞ and hence need to be set to $L_{\max}$.

V. Performance/Complexity Trade-offs

System engineers typically face the problem of designing a receiver that achieves a prescribed target frame error rate (FER) at a prescribed throughput. The quality of the receiver implementation can then be measured by the minimum SNR required to achieve this target FER at the specified throughput. In the following, we assess the complexity/performance trade-offs of the receiver concepts described in Sections III and IV by plotting the average (over independent channel and noise realizations) number of visited nodes as a function of this minimum SNR. Since the number of visited nodes is related to the required chip area per throughput [16], the corresponding results allow to associate a reduction in hardware complexity (e.g., chip area) to an SNR penalty.

All simulation results below are for a rate $R = 1/2$ (generator polynomials $[133_o\ 171_o]$ and constraint length 7) convolutionally encoded MIMO-OFDM system [21] with $M_T = M_R = 4$, 16-QAM constellation (using Gray mapping), $N = 64$ tones, and soft-input Viterbi decoding [22]. Note that for $L_{\max} = 0$, one has to employ a hard-input Viterbi decoder. One frame consists of 1024 randomly interleaved (across space and frequency) bits and corresponds to one OFDM symbol. A TGn type C channel model [23] is used and in all simulations, SNR is the per receive antenna SNR.

A. *Comparison of Tree-Search Strategies*

Fig. 4 compares the performance of RTS and STS max-log APP decoders, and the list sphere decoder (LSD) [10] for different target FERs and different values of $L_{\max}$. In the case of the LSD, changing the list size allows to adjust the complexity/performance trade-off.

The STS algorithm is seen to outperform the RTS strategy in terms of average complexity by a factor of 4 to 8.

The implementation of the LSD requires memory and logic for the administration of the candidate list, not accounted for in this comparison. Fig. 4 shows that even when this additional complexity is ignored, the STS is still superior to the LSD. Under stringent complexity constraints the STS shows SNR advantages over the LSD of up to 1.5 dB.

B. *Impact of Sorting and Regularization*

Fig. 5 compares the impact of sorting and regularization on the complexity/performance trade-off of the STS algorithm. Specifically, we show the trade-off curves corresponding to SQRD, MMSE-SQRD, and standard (unsorted) QRD at a target FER of 0.01. It can be seen that the improvement resulting from sorting (i.e., SQRD vs. QRD) becomes significant for stringent (but realistic) complexity constraints. Further improvements, in the low-complexity regime, are obtained from regularization using MMSE-SQRD. In the high-complexity regime, the performance penalty incurred by regularization (see the discussion in Section IV-B) eventually renders MMSE-SQRD inferior to SQRD.

C. *LLR Clipping*

Both Fig. 4 and Fig. 5 show that, as discussed in Section IV-A, adjusting the LLR clipping level $L_{\max}$ allows to sweep an entire family of sphere decoders ranging from the exact max-log APP SESD ($L_{\max} = \infty$) to hard-output

[1]In an OFDM-based MIMO system, N would, for example, be the number of OFDM tones.

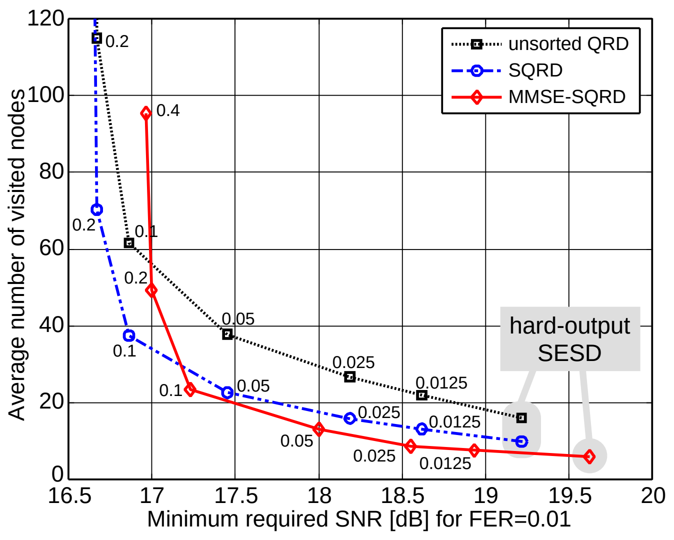

Fig. 5. Comparison of unsorted QRD, SQRD and MMSE-SQRD preprocessing applied to STS. The numbers next to the curves correspond to $L_{\max}$. For $L_{\max} = 0$, the performance equals that of hard-output SESD.

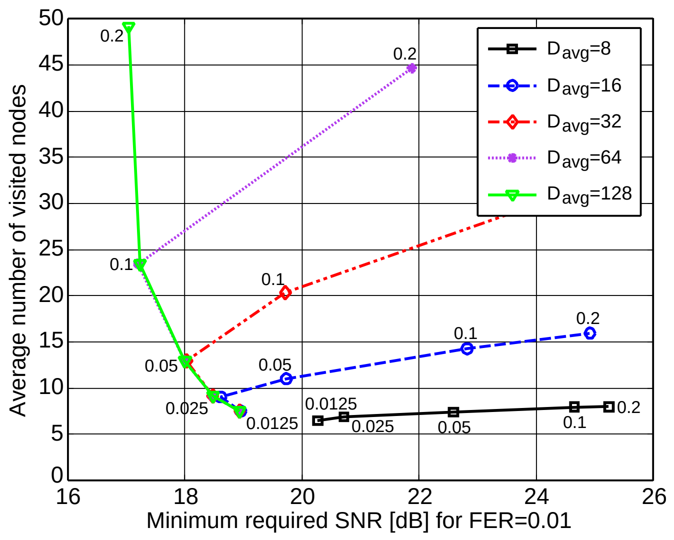

Fig. 6. Impact of imposing run-time constraints with maximum-first scheduling on STS SESD with MMSE-SQRD preprocessing.

SESD ($L_{\max} = 0$). It is interesting to observe that aggressive clipping according to $L_{\max} = 0.2$ yields close to max-log APP performance. Increasing the LLR clipping level beyond this value increases complexity without a noticeable performance improvement. Furthermore, we observe that the decoder performance degrades gracefully as we decrease $L_{\max}$ thereby reducing the average search complexity. In summary, the LLR clipping level can be used to conveniently adjust the decoder *at runtime* to a given complexity constraint.

D. Run-time Constraints

In Fig. 6, we demonstrate the impact of imposing a run-time constraint according to a maximum of ND_{avg} visited nodes for a frame of $N = 64$ symbol vectors using the strategy described in Section IV-C. The resulting curves essentially consist of two regions:

- If the LLR clipping level $L_{\max}$ is high (corresponding to high average search complexity), the run-time constrained detector is not able to compute accurate LLR values, which results in (very) poor performance, unless D_{avg} is large. For $D_{\text{avg}} = 128$, the performance is, indeed, very close to that of the unconstrained max-log APP SESD.
- For small $L_{\max}$, the performance is dominated by the impact of clipping rather than by the impact of the run-time constraint.

In summary, we can conclude that for a given run-time constraint there exists an optimum LLR clipping level, in the sense of minimizing the SNR required to achieve a certain target FER. It is therefore important to choose the LLR clipping level in accordance with the average run-time constraint.

VI. VLSI Architecture for Max-Log STS Soft-Output Sphere Decoding

Since the proposed max-log STS SESD VLSI implementation is based on the one-node-per-cycle (ONPC) VLSI architecture developed in [16] for hard-output SESD, we start our discussion by briefly reviewing relevant aspects of [16].

A. A Brief Review of the ONPC Architecture in [16]

The VLSI architecture proposed in [16] employs two functional units:

Metric Computation Unit (MCU): The MCU handles the forward iteration in the search tree by identifying the starting-point for the SE enumeration (i.e., the current node's child that has the smallest PED) using the direct-QAM enumeration algorithm initially proposed in [10] and slightly modified in [16]. The basic idea behind this enumeration method for QAM constellations is as follows: The QAM constellation is first decomposed into subsets of constellation points that have the same modulus, referred to as phase-shift keying (PSK) subsets. Within each of these PSK subsets, the child associated with the smallest PED can be identified based on the phase of $b_i = \tilde{y}_i - \sum_{j=i+1}^{M_T} R_{i,j} s_j$ only. The corresponding minimum PEDs (one for each subset) are then computed and compared. The minimum PED across subsets identifies the starting point for the SE enumeration. If the resulting child neither corresponds to a leaf nor qualifies for pruning, the decoder proceeds in forward direction by declaring this child as the next parent node to be examined by the MCU (cf. ① in Fig. 7).

Metric Enumeration Unit (MEU): The MEU maintains a list of preferred children, one for each node between the root and the parent of the node whose children are currently under examination by the MCU. To this end, the MEU follows the MCU on its path through the tree with one cycle delay. While the MCU visits a node, the MEU considers this node's siblings and identifies the one that should be visited next according to the SE criterion. This sibling is found by applying the direct-QAM enumeration principle described above, where within each PSK subset the next (according to the SE criterion) candidate follows immediately by zig-zag enumeration along the circle. The decision on the preferred child across subsets must again be made by explicit computation and comparison of the smallest PEDs of the individual PSK subsets.

When the forward iteration stalls, either because the child

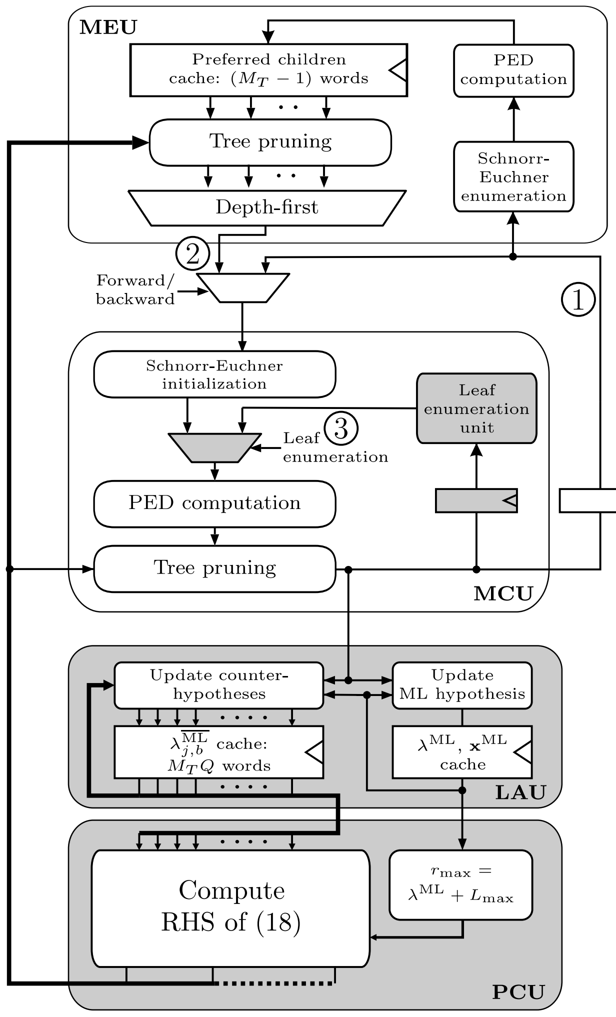
MEU
Preferred children cache: (M_T-1) words
PED computation
Tree pruning
Schnorr-Euchner enumeration
Depth-first
2
Forward/ backward
1
Schnorr-Euchner initialization
Leaf enumeration unit
Leaf enumeration
3
PED computation
Tree pruning
MCU
Update counter-hypotheses
Update ML hypothesis
$\lambda_{j,b}^{\overline{\mathrm{ML}}}$ cache: M_TQ words
λ^{ML}, $\mathbf{x}^{\mathrm{ML}}$ cache
LAU
Compute RHS of (18)
$r_{\max} = \lambda^{\mathrm{ML}} + L_{\max}$
PCU

Fig. 7. Block diagram of the proposed VLSI architecture for the soft-output STS SESD. Additional units, compared to the hard-output SESD described in [16], are highlighted.

identified by the MCU corresponds to a leaf or must be pruned, the MEU provides a new parent node to the MCU in the next clock cycle (cf. ② in Fig. 7). This parent node is chosen by the MEU, following the depth-first paradigm, from those members of the list of preferred children which do not qualify for pruning.

B. VLSI Architecture for STS SESD

The block diagram of the proposed max-log STS SESD VLSI implementation is shown in Fig. 7. Compared to the architecture for hard-output SESD described in [16], changes are made in the MCU and two additional units are required, one for list administration as described in Section III-B1 and one for the implementation of the pruning criterion as described in Section III-B2. We shall next describe the specifics of these changes.

1) Architectural Changes in the MCU: From a high-level architectural perspective, there is one fundamental difference between tree-traversal for hard-output SESD and for the STS 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 SESD considers only one child, namely the one associated with the smallest PED. The STS algorithm, however, has to compute the PEDs of all children that do not qualify for pruning according to the criterion (10) since these children may lead to updates of the metrics $\lambda_{j,b}^{\overline{\mathrm{ML}}}$. To perform this *leaf enumeration* procedure, the STS decoder must revisit the current node at level $i = 2$, which requires additional clock cycles and a leaf enumeration unit shown in Fig. 7. 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 (cf. ③ in Fig. 7).

2) List Administration and Tree Pruning: In addition to the modifications in the MCU described above, the STS algorithm requires the following two additional units:

List-Administration Unit (LAU): The LAU is responsible for maintaining and updating the list containing $\mathbf{x}^{\mathrm{ML}}$, λ^{ML}, and the $\lambda_{j,b}^{\overline{\mathrm{ML}}}$. The corresponding unit is active during the leaf-enumeration process described above. Since the update rules implemented by the LAU (see Section III-B1) require only a small number of logic operations, the silicon area of this unit is small (see Table II) and is dominated by the storage space required for the metrics λ^{ML} and $\lambda_{j,b}^{\overline{\mathrm{ML}}}$.

Pruning Criterion Unit (PCU): The PCU is responsible for computing the reference metrics, i.e., the RHS of (10), required to implement the corresponding pruning criterion. From a VLSI implementation perspective, the reference metric on level i depending on the partial label $\mathbf{x}^{(i)}$ constitutes a major problem. More specifically, this dependence causes the criterion for pruning the child of a parent node on level $i+1$ to depend on the partial label $\mathbf{x}^{(i)}$ of that child. This, in turn, implies that enumeration of the children on level i in ascending order of their PEDs according to the SE criterion can not be applied, which results in the need for exhaustive-search enumeration and is thus ill-suited for VLSI implementation [16]. An adjustment of the pruning criterion in (10) solves this problem. To this end, we define

$$\begin{aligned}\mathcal{B}\Big(\mathbf{x}^{(i+1)}\Big) &= \{b_l\} = \\ &= \Big\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\big|\, \big(j > i, b = 1,2,\ldots,Q\big) \wedge \big(x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}\big)\Big\} \\ &\quad \cup \Big\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\big|\, j \le i, b = 1,2,\ldots,Q\Big\}\end{aligned}$$

and prune the node $\mathbf{s}^{(i)}$ along with its subtree if $d\Big(\mathbf{s}^{(i)}\Big)$ satisfies

$$d\Big(\mathbf{s}^{(i)}\Big) > \max_{b_l \in \mathcal{B}\left(\mathbf{x}^{(i+1)}\right)} b_l\,. \tag{18}$$

Note that the RHS of the modified pruning criterion (18) depends on the partial label $\mathbf{x}^{(i+1)}$ rather than on $\mathbf{x}^{(i)}$. Consequently, the enumeration of the children of a node on level $i+1$ can be carried out using the SE criterion.

C. Impact of List Administration and Tree Pruning on Complexity

We argued throughout the paper that, for a ONPC architecture, the number of visited nodes is equal to the number of clock cycles required for decoding thus reflecting the true

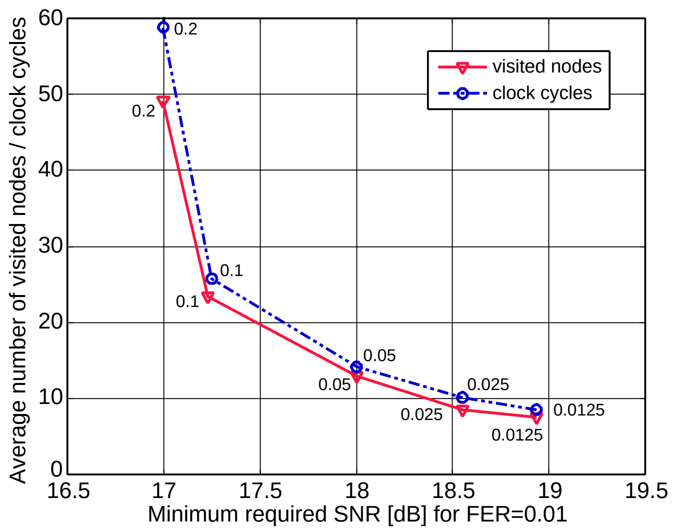

Fig. 8. Average number of visited nodes of the STS algorithm compared with the average number of clock cycles of the corresponding VLSI implementation (with MMSE-SQRD preprocessing). The numbers next to the curves correspond to LLR clipping levels.

silicon complexity of the algorithm. However, for the proposed STS architecture the number of clock cycles will be larger than the number of visited nodes shown in the numerical results in Section V, for two reasons: First, modifying the pruning criterion (10) to result in (18) leads to less efficient pruning as

$$\max_{b_l \in \mathcal{B}\left(\mathbf{x}^{(i+1)}\right)} b_l \geq \max_{a_l \in \mathcal{A}\left(\mathbf{x}^{(i)}\right)} a_l .$$

The corresponding complexity increase is, however, significantly smaller than what would be incurred if exhaustive search enumeration on (10) would be applied. The second reason for the number of clock cycles being higher than the number of visited nodes is that every time the leaf-enumeration process is performed, one additional cycle is consumed to detect the end of the enumeration process. Consequently, the proposed VLSI architecture no longer strictly follows the ONPC paradigm. The results in Fig. 8 show, however, that the impact of the two effects discussed above leads to the number of clock cycles being only slightly higher than the number of visited nodes.

D. Architectural Considerations for RTS

In this section, we would like to discuss architectural considerations for possible implementations of the RTS strategy. As described in Section III-A, the RTS approach corresponds to repeated runs of a hard-output SESD which, in principle, can be implemented efficiently using the ONPC VLSI architecture introduced in [16]. However, forcing the decoder to search only over the set $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}$ when computing the counter-hypotheses $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ requires to constrain the search to a subset of admissible constellation points, which, moreover depend on the (bits to symbol) mapping. Consequently, as depicted in Fig. 9, straightforward zig-zag enumeration can no longer be applied. In addition, as demonstrated in Fig. 9, different

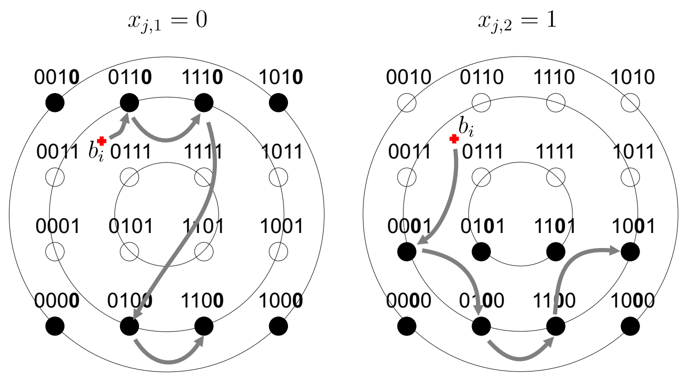

Fig. 9. Example for SE enumeration in a PSK subset (of 16-QAM with Gray mapping) when searching for $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ with $b = 1, 2$ and with $x_{j,b}^{\mathrm{ML}} = 1$ for $b = 1, 2$. The bits under consideration have been set in boldface font.

counter-hypotheses will result in different sets of allowed constellation points, which induces an irregularity that results in an increase in hardware complexity. The problem can be mitigated to a certain extent by adjusting the mapping. However, this, in general, results in a (bit error rate) performance degradation. Alternatively using exhaustive search enumeration, as described in [16], to compute λ^{ML} and the counter-hypotheses $\lambda_{j,b}^{\overline{\mathrm{ML}}}$, is not a viable option as it results in significant overhead in terms of chip area and in an increase in the length of the critical path. For a quantitative analysis of the impact of exhaustive-search enumeration (in hard-output SESD) the interested reader is referred to [16].

VII. ASIC Implementation Results

In order to assess the true silicon complexity (chip area and achievable clock frequency) of the proposed STS-based soft-output SESD, we implemented the VLSI architecture described in the previous section in 0.25 μm CMOS technology for a MIMO system with $M_T = M_R = 4$ using 16-QAM modulation. The resulting chip layout is shown in Fig. 10. The design parameters of the decoder are summarized in Table I which, for reference, also contains the design parameters of an ℓ^2-norm hard-output SESD, following the design princi-

TABLE I
DESIGN PARAMETERS OF HARD-OUTPUT SESD AND SOFT-OUTPUT STS ASICS IN 0.25 μm CMOS TECHNOLOGY

	Hard-output SESD	Soft-output STS
Gate equivalents[a]	34.4 kGE	56.8 kGE
Core area	1.2 mm^2	1.9 mm^2
Max. clock frequency[b]	73 MHz	71 MHz
AT-product[c]	0.164 μm^{2}s	0.268 μm^{2}s

[a] To provide technology-independent area characterization, the number of gate equivalents (GEs) is specified. One GE corresponds to the area of a two-input drive-one NAND gate.

[b] The results on clock frequency are extracted from a post-layout static timing analysis and are representative for the manufactured ASIC within an accuracy of a few percent.

[c] The area-timing (AT) product of a VLSI circuit is a measure for its true silicon complexity. It is given by the product of the chip area divided by the maximum allowed clock frequency.

TABLE II
DETAILED CHIP AREA BREAKDOWN OF THE DIFFERENT FUNCTIONAL UNITS IN A HARD-OUTPUT SESD AND IN A SOFT-OUTPUT STS SESD

	Hard-output SESD		Soft-output STS	
	Area [kGE]	Area [%]	Area [kGE]	Area [%]
Mem. $(\mathbf{y}, \mathbf{R})$	4.6	13.4	4.5	7.9
MCU	16.6	48.3	18.5	32.6
MEU	11.9	34.6	10.9	19.2
Output buffer	0.8	2.3	5.0	8.8
Control logic	0.5	1.6	0.5	0.9
LAU	-	-	8.4	14.7
PCU			6.4	11.3
LLR Comp.			2.6	4.6
Total	34.4	100	56.8	100

ples employed for the ℓ^∞-norm hard-output SESD described in [16].

Hardware Complexity: We can see from Table I that the chip area required by the soft-output STS SESD is only 58% higher than that required by a corresponding ℓ^2-norm hard-output SESD. The detailed area breakdown in Table II shows that most of the area increase results from the LAU, the PCU, and the arithmetic unit that computes the LLRs. Further area increase is due to the need to store the LLRs in the output buffer of the ASIC. The additional Schnorr-Euchner enumeration unit in the MCU required for leaf enumeration adds only 1.9 kGEs to the overall area. The soft-output STS SESD ASIC shows only slightly lower maximum clock frequency than the corresponding hard-output SESD. The reason underlying this only negligible reduction in maximum clock frequency is that most of the additional logic required by the STS SESD ASIC can be kept off the critical path and has thus little influence on the maximum clock frequency.

Detection Throughput: Fig. 11 shows the complexity/performance trade-off of the reference ℓ^2-norm hard-output SESD and the soft-output STS described in Section VI in terms of the throughput

$$\Theta = \frac{RQM_T}{\mathbb{E}[C]} f_{\text{clk}} \quad [\text{bit/s}]$$

measured in *information*-bits per second as a function of the minimum required SNR to achieve a FER of 0.01. Here, f_{clk} is the maximum clock frequency of the circuit under consideration and $\mathbb{E}[C]$ denotes the average (over channel and noise realizations) number of clock cycles required to detect a symbol vector. Note that the dedicated hard-output SESD implementation achieves a slightly higher throughput than the STS SESD implementation with[2] $L_{\max} = 0$. This is due to the slightly higher maximum clock frequency of the corresponding hard-output SESD (see Table I).

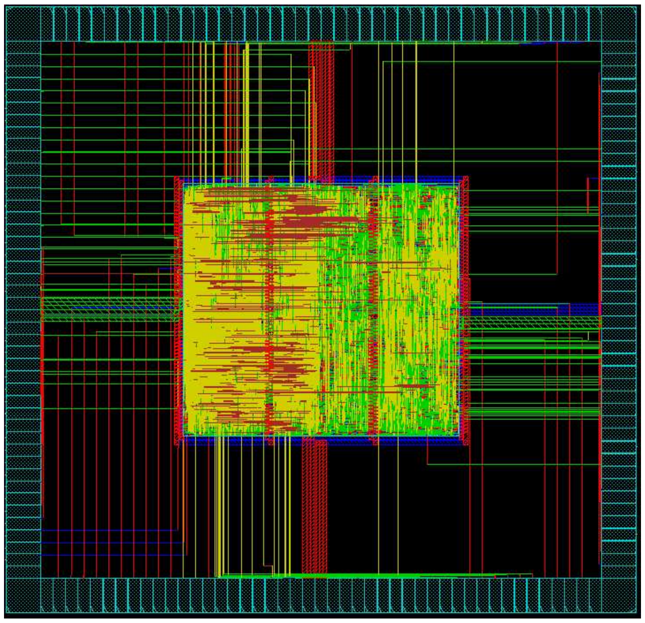
Fig. 10. Layout of an STS-based soft-output SESD ASIC in 0.25 μm CMOS technology.

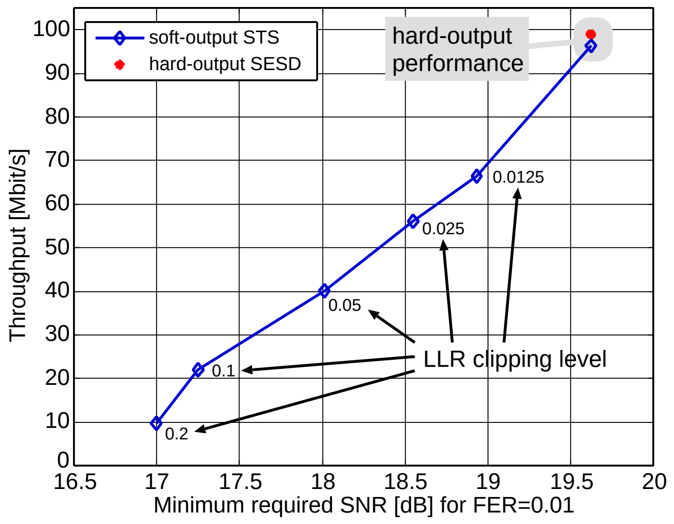

Fig. 11. Throughput characteristics of the soft-output STS SESD and the reference hard-output SESD VLSI implementation with MMSE-SQRD preprocessing.

VIII. CONCLUSIONS

Sphere decoding is a suitable tool to implement MIMO detection with variable complexity/performance trade-off. In particular, adjusting the LLR clipping level and imposing maximum run-time constraints is an efficient way of realizing an entire family of decoders with (error rate) performance ranging from exact max-log soft-output to hard-output SIC. The keys to achieving low hardware complexity are the single tree-search strategy described in Section III-B, MMSE-SQRD preprocessing, LLR clipping, and run-time constraints with maximum-first scheduling. Our VLSI implementation results indicate that the silicon area required by a soft-output STS SESD is only about 58% higher than the area required for a corresponding ℓ^2-norm hard-output SESD implementation. This paves the way for a VLSI implementation of iterative MIMO detection based on sphere decoding.

[2]Recall that for $L_{\max} = 0$, the (error rate) performance of the STS SESD algorithm corresponds to that of a hard-output SESD.

REFERENCES

[1] C. Studer, M. Wenk, A. Burg, and H. Bölcskei, "Soft-output MIMO detection algorithms: Performance and implementation aspects," in *Proc. of the 40th Asilomar Conf. on Signals, Systems, and Computers*, Oct. 2006, pp. 2071–2076.
[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] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, pp. 463–471, Apr. 1985.
[4] 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, Sept. 1994.
[5] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," *Colloq. GRETSI*, vol. 14, pp. 611–614, Sept. 1993.
[6] 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, July 1999.
[7] O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Communications Letters*, vol. 4, no. 5, pp. 161–163, May 2000.
[8] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.
[9] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge Univ. Press, 2003.
[10] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, Mar. 2003.
[11] J. Jaldén and B. Ottersten, "Parallel implementation of a soft output sphere decoder," in *Proceedings Asilomar Conf. on Signals, Systems and Computers*, Nov. 2005, pp. 581–585.
[12] M. S. Yee, "Max-Log-Map sphere decoder," in *Proc. IEEE ICASSP 2005*, vol. 3, Mar. 2005, pp. 1013–1016.
[13] 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.
[14] B. Steingrimsson, T. Luo, and K. M. Wong., "Soft quasi-maximum-likelihood detection for multiple-antenna wireless channels," *IEEE Transactions on Signal Processing*, vol. 51, no. 11, pp. 2710–2719, Nov. 2003.
[15] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.
[16] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.
[17] P. Marsch, E. Zimmermann, and G. Fettweis, "Smart candidate adding: A new low-complexity approach towards near-capacity MIMO detection," in *Proceedings of the 13th European Signal Processing Conf. (EUSIPCO)*, Sept. 2005.
[18] 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.
[19] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE Vehicular Technology Conf. (Fall)*, vol. 1, Oct. 2003, pp. 508–512.
[20] A. Burg, M. Borgmann, M. Wenk, C. Studer, and H. Bölcskei, "Advanced receiver algorithms for MIMO wireless communications," in *Proceedings of the Design Automation and Test Europe Conf. (DATE)*, vol. 1, May 2006, pp. 593–598.
[21] H. Bölcskei, D. Gesbert, and A. J. Paulraj, "On the capacity of OFDM-based spatial multiplexing systems," *IEEE Trans. Communications*, vol. 50, no. 2, pp. 225–234, Feb. 2002.
[22] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.
[23] V. Erceg *et al.*, *TGn channel models*, May 2004, IEEE 802.11 document 03/940r4.

Christoph Studer (S'06) was born in Solothurn, Switzerland on December 25, 1979. He received the MSc degree in electrical engineering from the ETH Zurich, Zurich, Switzerland, in 2005, where he is currently working toward the Dr. sc. techn. degree.

In 2005, he was a Visiting Researcher with the Smart Antennas Research Group, Stanford University, Stanford, CA, USA. Since 2006, he has been a Research Assistant with the Integrated Systems Laboratory (IIS) at ETH Zurich. His research interests include signal processing for wireless communications and the design of VLSI circuits and systems.

Mr. Studer was the recipient of an ETH Medal in 2005 for his Master's Thesis.

Andreas Burg (S'97–M'05) was born in Munich, Germany, in 1975. He received the Dipl.-Ing. degree in 2000 from the ETH Zurich, Zurich, Switzerland, in 2000. He then joined the Integrated Systems Laboratory of ETH Zurich, where he received the Dr. sc. techn. degree in 2006. From 2006 to 2007, he held positions as postdoctoral researcher jointly at the Integrated Systems Laboratory and the Communication Technology Laboratory at ETH Zurich. In 2007 he co-founded Celestrius AG, an ETH-spinoff in the field of MIMO wireless communication.

In 1998, he worked at Siemens Semiconductors, San Jose, CA. During his doctoral studies, he was a visiting researcher with Bell Labs Wireless Research for a total of one year. His research interests include the design of digital VLSI circuits and systems and signal processing for wireless communications.

In 2000, Mr. Burg received the "Willi Studer Award" and an ETH Medal for his diploma and his diploma thesis, respectively. Mr. Burg was also awarded an ETH Medal for his Ph.D. dissertation in 2006.

Helmut Bölcskei (M'98–SM'02) was born in Mödling, Austria on May 29, 1970, and received the Dipl.-Ing. and Dr. techn. degrees in electrical engineering/communication theory from Vienna University of Technology, Vienna, Austria, in 1994 and 1997, respectively. In 1998 he was with Vienna University of Technology. From 1999 to 2001 he was a postdoctoral researcher in the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA. He was in the founding team of Iospan Wireless Inc., a Silicon Valley-based startup company (acquired by Intel Corporation in 2002) specialized in multiple-input multiple-output (MIMO) wireless systems for high-speed Internet access. From 2001 to 2002 he was an Assistant Professor of Electrical Engineering at the University of Illinois at Urbana-Champaign. He has been with ETH Zurich since 2002, where he is Professor of Communication Theory. He was a visiting researcher at Philips Research Laboratories Eindhoven, The Netherlands, ENST Paris, France, and the Heinrich Hertz Institute Berlin, Germany. He is a cofounder of the ETH spin-off company Celestrius AG, where he serves as Chief Scientist. His research interests include communication and information theory with special emphasis on wireless communications, signal processing and quantum information processing.

He received the 2001 IEEE Signal Processing Society Young Author Best Paper Award, the 2006 IEEE Communications Society Leonard G. Abraham Best Paper Award, the ETH "Golden Owl" Teaching Award, and was an Erwin Schrödinger Fellow (1999-2001) of the Austrian National Science Foundation (FWF). He was a plenary speaker at several IEEE conferences and served as an associate editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and the *EURASIP Journal on Applied Signal Processing*. He is currently on the editorial board of "Foundations and Trends in Networking", serves as an associate editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and is TPC co-chair of the 2008 IEEE International Symposium on Information Theory.