ASIC Implementation of Soft-Input Soft-Output MIMO Detection Using MMSE Parallel Interference Cancellation

Christoph Studer, *Member, IEEE*, Schekeb Fateh, *Student Member, IEEE*, and Dominik Seethaler

***Abstract*—Multiple-input multiple-output (MIMO) technology is the key to meet the demands for data rate and link reliability of modern wireless communication systems, such as IEEE 802.11n or 3GPP-LTE. The full potential of MIMO systems can, however, only be achieved by means iterative MIMO decoding relying on soft-input soft-output (SISO) data detection. In this paper, we describe the first ASIC implementation of a SISO detector for iterative MIMO decoding. To this end, we propose a low-complexity minimum mean-squared error (MMSE) based parallel interference cancellation algorithm, develop a suitable VLSI architecture, and present a corresponding four-stream 1.5 mm^2 detector chip in 90 nm CMOS technology. The fabricated ASIC includes all necessary preprocessing circuitry and exceeds the 600 Mb/s peak data-rate of IEEE 802.11n. A comparison with state-of-the-art MIMO-detector implementations demonstrates the performance benefits of our ASIC prototype in practical system-scenarios.**

***Index Terms*—Very-large scale integration (VLSI), wireless communication, multiple-input multiple-output (MIMO), spatial multiplexing, iterative detection and decoding, soft-input soft-output (SISO), minimum mean-square error (MMSE), parallel interference cancellation (PIC).**

I. Introduction

THE continuous increase in data rate and quality-of-service of modern wireless communication systems can only be met by novel technologies providing higher spectral efficiency and improved link reliability. Multiple-input multiple-output (MIMO) technology [2], which uses multiple antennas at both ends of the wireless link, in combination with spatial multiplexing and channel coding is believed to be the key to reliable, high-speed, and bandwidth-efficient data transmission. Therefore, MIMO technology is incorporated in many modern wireless communication standards, such as IEEE 802.11n [3] or 3GPP-LTE [4].

In MIMO communication systems, data detection, i.e., the separation of the spatially-multiplexed data streams, and channel decoding are typically among the main challenges in computational complexity and power consumption. Therefore, corresponding *efficient* implementations are the key to facilitate high-performance, low-power, and low-cost user equipment. The performance of MIMO technology critically depends on the employed data-detection algorithm and corresponding methods usually entail very high computational complexity. In particular, the computational complexity of the optimum MIMO detection algorithm scales exponentially in the number of spatial streams [2]. Hence, research has mainly focused on the development of low-complexity algorithms including their efficient implementation in hardware. For example, MIMO detection based on the sphere-decoding (SD) algorithm [5]–[7] is able to achieve close-to-optimal error-rate performance. However, corresponding implementations require multiple parallel SD-instances to achieve the 600 Mb/s peak data-rate of IEEE 802.11n (see, e.g., [7] for details), which is due to SD's prohibitive worst-case complexity. Recent implementations of sub-optimum methods, e.g., based on the k-Best algorithm [8], [9] or on minimum mean-square error (MMSE) detection [10], exceed 600 Mb/s data-rate at the cost of inferior error-rate performance, eventually degrading the system throughput and link reliability.

All these techniques rely on a single channel-decoding step without iteratively exchanging information with the MIMO detector. However, it was demonstrated in [11] that the full potential of MIMO wireless systems can, in practice, only be achieved through *iterative MIMO decoding* and corresponding experiments based on a low-complexity FPGA prototype [12] support this fact. At the heart of an iterative MIMO decoder is a soft-input soft-output (SISO) MIMO detector (referred to as "SISO detector" in the remainder of the paper), which iteratively exchanges reliability information of the coded bits with a SISO channel decoder. SISO detection algorithms exhibit, in general, significantly higher computational complexity compared to their non-iterative counterparts (see, e.g., [11], [13]), which emphasizes on the necessity of novel low-complexity algorithms and corresponding dedicated ASIC implementations. More specifically, recent synthesis results of a SD-based SISO detector [14] achieve data rates that are 10× below that specified in IEEE 802.11n, which points at massive implementation challenges.

1) Contributions: In this paper, we describe the design and ASIC implementation of the first SISO detector achieving the 600 Mb/s peak data-rate of IEEE 802.11n. To this end, we develop a novel low-complexity variant of the SISO MMSE parallel interference cancellation (PIC) algorithm proposed

This paper was presented in part at the 36th European Solid-State Circuits Conference (ESSCIRC), Sevilla, Spain, Sept. 2010 [1].

C. Studer is with the Communication Technology Laboratory (CTL), ETH Zurich, 8092 Zurich, Switzerland, (e-mail: studerc@nari.ee.ethz.ch). S. Fateh is with the Integrated Systems Laboratory (IIS), ETH Zurich, 8092 Zurich, Switzerland (e-mail: fateh@iis.ee.ethz.ch). D. Seethaler was with the Communication Technology Laboratory (CTL), ETH Zurich, 8092 Zurich, Switzerland and is now with RobArt, 4020 Linz, Austria (e-mail: dominik.seethaler@gmail.com).

MATLAB code of a simulation environment for iterative MIMO decoding including the SISO MMSE-PIC algorithm is available for download at `http://www.nari.ee.ethz.ch/commth/research/downloads/`

in [15]. The main complexity savings (at no performance loss) are achieved by reducing the number of required matrix inversions. We design a corresponding VLSI architecture that includes all necessary channel-matrix preprocessing circuitry. The architecture employs an LU-decomposition-based matrix inversion, which outperforms existing inversion circuits for MIMO systems in terms of area per throughput. Key for achieving high throughput is the use of a custom Newton-Raphson-based reciprocal unit. We finally present a corresponding four-stream 1.5 mm^2 detector chip in 90 nm CMOS technology achieving up to 757 Mb/s. We provide measurement results of the fabricated ASIC, compare it to state-of-the-art MIMO detector implementations, and demonstrate that our prototype enables substantial performance gains in practical system-scenarios.

2) Notation: Matrices are set in boldface capital letters, vectors in boldface lowercase letters. The superscript ^{H} stands for conjugate transpose and $\mathbf{I}_M$ is the $M \times M$ identity matrix. $\mathrm{P}[\cdot]$ denotes probability; expectation and variance is referred to as $\mathbb{E}[\cdot]$ and $\mathrm{Var}[\cdot]$, respectively.

3) Outline: The remainder of the paper is organized as follows. Section II introduces the system model and summarizes the SISO MMSE-PIC algorithm. In Section III, we propose various techniques that reduce the computational complexity. The VLSI architecture is detailed in Section IV and the corresponding ASIC implementation results, along with a comparison to existing detector implementations, are provided in Section V. We conclude in Section VI.

II. Iterative MIMO Decoding

We consider a coded MIMO system, as specified in IEEE 802.11n [3], which employs spatial multiplexing with M_T transmit and $M_\mathrm{R} \geq M_\mathrm{T}$ receive antennas (see Figure 1). The information bits $\mathbf{b}$ are encoded (e.g., using a convolutional code) and the resulting coded bit-stream $\mathbf{x}$ is mapped (using Gray labeling) to a sequence of transmit vectors $\mathbf{s} \in \mathcal{O}^{M_\mathrm{T}}$, where $\mathcal{O}$ corresponds to the scalar complex constellation of size 2^Q. Each transmit vector $\mathbf{s}$ is associated with $M_\mathrm{T}Q$ binary values $x_{i,b} \in \{0,1\}$, $i = 1,\ldots,M_\mathrm{T}$, $b = 1,\ldots,Q$, corresponding to the bth bit of the ith entry (i.e., spatial stream) of $\mathbf{s}$. We assume $\mathbb{E}\left[\mathbf{s}\mathbf{s}^H\right] = E_\mathrm{s}\mathbf{I}_{M_\mathrm{T}}$, where E_s denotes the symbol variance. The baseband input-output relation of the wireless MIMO channel is given by $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$ where $\mathbf{H}$ stands for the $M_\mathrm{R} \times M_\mathrm{T}$ complex-valued channel matrix, $\mathbf{y}$ is the M_R-dimensional received vector, and $\mathbf{n}$ is M_R-dimensional i.i.d. zero-mean complex Gaussian distributed noise with variance N_0 per entry. We assume that the channel matrix $\mathbf{H}$, the noise variance N_0, and the symbol variance E_s are perfectly known at the receiver.[1]

A. Principles of Iterative MIMO Decoding

Iterative MIMO decoding is based on the ideas of turbo-decoding [17]. Here, reliability information of the coded bits—in terms of log-likelihood ratios (LLRs)—is iteratively exchanged between the SISO detector and the SISO channel decoder (see Figure 1) to successively improve the error-rate performance. In each iteration, the SISO detector computes extrinsic LLRs [11], [13]

$$L^\mathrm{E}_{i,b} = \log\left(\frac{\mathrm{P}[x_{i,b}=1\,|\,\mathbf{y}]}{\mathrm{P}[x_{i,b}=0\,|\,\mathbf{y}]}\right) - L^\mathrm{A}_{i,b} \qquad (1)$$

for the coded bits $x_{i,b}$, based on the received vector $\mathbf{y}$ and on the a-priori LLRs $L^\mathrm{A}_{i,b}$, $i = 1,\ldots,M_\mathrm{T}$, $b = 1,\ldots,Q$. The extrinsic LLRs $L^\mathrm{E}_{i,b}$, which indicate the reliability for each coded bit $x_{i,b}$, are then delivered to the SISO channel decoder, which computes *new* a-priori LLRs $L^\mathrm{A}_{i,b}$, $\forall i,b$, that are used by the SISO detector in the next iteration. After a given number of iterations (denoted by I), the SISO channel decoder provides final decisions $\hat{\mathbf{b}}$ for the information bits $\mathbf{b}$ based on the LLRs at the channel-decoder output.[2]

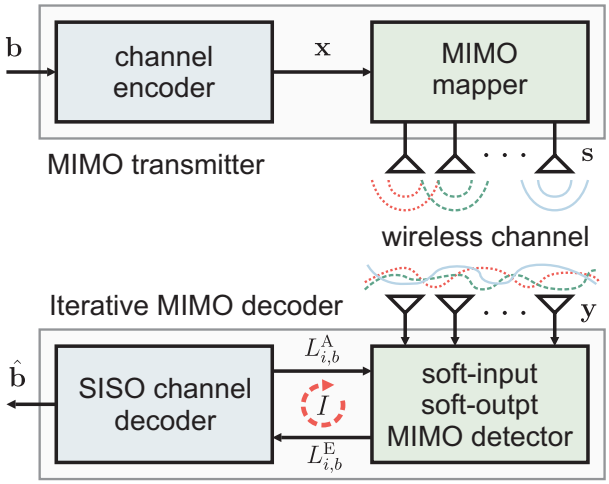

Fig. 1. Coded MIMO communication system using iterative MIMO decoding.

B. Performance Benefits of Iterative MIMO Decoding

Figure 2 illustrates the (coded) packet error-rate (PER) versus (average receive) signal-to-noise ratio (SNR) performance of iterative MIMO decoding in a typical 40 MHz IEEE 802.11n scenario with four spatial streams ($M_\mathrm{T} = M_\mathrm{R} = 4$), 16-QAM modulation, and a rate-1/2 convolutional code. One data packet consists of 864 information bits and we declare a packet error if at least one information bit is decoded in error. We used a TGn type C channel model [19] and the max-log BCJR algorithm [20] for SISO channel decoding. We compare hard-output SD (see, e.g., [5], [7]), the k-Best algorithm as implemented in [8], the close-to-optimal SISO single tree-search (STS) SD algorithm [13], and the SISO MMSE-PIC algorithm [15], which is considered in the remainder of the paper.

Figure 2 demonstrates substantial SNR-performance improvements (compared to non-iterative hard-output SD and the k-Best algorithm) that can be achieved with iterative MIMO decoding. In particular, performing four iterations only,

[1] In practice, channel-state information (CSI) is commonly acquired through training and hence, not perfect. Since imperfect CSI penalizes the performance of all considered MIMO detection algorithms in a similar way (see, e.g., [16] for details), we assume—for the sake of simplicity of exposition—perfect CSI throughout the paper.

[2] Early-termination schemes that reduce unnecessary iterations in turbo decoders (e.g., [18]) may also be used in iterative MIMO systems to reduce power consumption or to improve the (average) throughput.

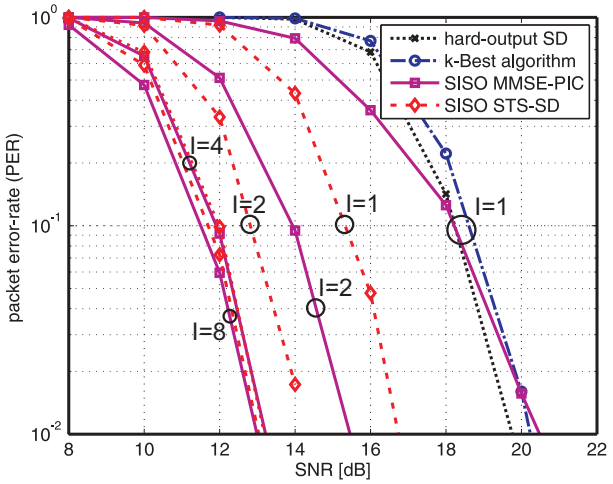

Fig. 2. PER versus SNR performance of a coded MIMO system using various iterative (using $I \in \{2, 4, 8\}$ number of iterations) and non-interative ($I = 1$) MIMO detection algorithms.

enables to achieve 10% PER at 12 dB SNR for both SISO STS-SD and SISO MMSE-PIC, compared to the more than 18 dB SNR that is required by hard-output SD and the k-Best algorithm. Hence, four iterations improve the performance by more than 6 dB SNR at IEEE 802.11n-relevant PERs. Note that the SNR-performance gain from four to eight iterations is only 0.25 dB, which indicates that performing more than four iterations does not pay off in practical systems. We emphasize that the SNR-performance advantage of iterative MIMO decoding ultimately leads to an increased system throughput (as higher data-rates can be used reliably at the same SNR), better coverage, and improved range (since the lowest data-rate can be decoded reliably at lower SNR).

We finally note that SISO STS-SD outperforms SISO MMSE-PIC for a small number of iterations. However, the SISO STS-SD algorithm requires i) roughly 8× higher computational complexity than SISO MMSE-PIC (cf. Section V-C3) and ii) SD-based algorithms exhibit—in contrast to the other considered algorithms—a non-constant throughput strongly depending on the SNR and the channel realization. Since practical MIMO receivers need to cope with varying channel conditions and transmission rates, the non-constant throughput renders implementations of SD extremely difficult (see [7] for a corresponding discussion). Both drawbacks associated with SD finally led to our decision to favor the SISO MMSE-PIC algorithm for implementation.

C. SISO MMSE-PIC Algorithm

Even for a small number of spatial streams (say $M_{\mathrm{T}} > 2$), exact computation of the LLRs in (1) entails prohibitively high computational complexity. Therefore, a variety of sub-optimum algorithms has been proposed in the literature, e.g., [13], [15]. In this paper, we focus on the SISO MMSE-PIC algorithm initially proposed by Wang and Poor in 1999 [15] in the context of multi-user detection. Since then, various algorithm optimizations have been proposed [21]–[24]. The following five paragraphs summarize the SISO MMSE-PIC algorithm as described in [23].

1) Computation of Soft-Symbols: The algorithm starts by computing estimates $\hat{s}_i$ for $i = 1, \ldots, M_{\mathrm{T}}$ (referred to as "soft-symbols") for the transmitted symbols s_i according to [21]

$$\hat{s}_i = \mathbb{E}[s_i] = \sum_{a \in \mathcal{O}} \mathrm{P}[s_i = a]\, a \tag{2}$$

where $\mathrm{P}[s_i = a] = \prod_{b=1}^{Q} \mathrm{P}[x_{i,b} = k]$ denotes to the a-priori probability of the symbol $a \in \mathcal{O}$ with $k = [a]_b$ referring to the bth bit associated with the symbol a. The reliability of each soft-symbol $\hat{s}_i$ is characterized by its variance

$$E_i = \mathrm{Var}[s_i] = \mathbb{E}\Big[|e_i|^2\Big] \tag{3}$$

with $e_i = s_i - \hat{s}_i$. The a-priori probabilities involved in the computation of the soft-symbols (2) and their variances (3) are calculated on the basis of the a-priori LLRs $L_{i,b}^{\mathrm{A}}$ delivered by the channel decoder.[3] According to [25], we have

$$\mathrm{P}[x_{i,b} = k] = \frac{1}{2}\left(1 + (2k-1)\tanh\left(\frac{1}{2}L_{i,b}^{\mathrm{A}}\right)\right) \tag{4}$$

which can be approximated efficiently in hardware through table look-ups.

As observed in [23], using *intrinsic* a-priori LLRs in the computation of (4) instead of the extrinsic ones leads, in general, to significantly better error-rate performance of the SISO MMSE-PIC algorithm. We therefore exclusively use intrinsic a-priori LLRs for the computation of (4) throughout the paper. We finally note that for most Gray mappings (including that used in IEEE 802.11n) the soft-symbols in (2) and their corresponding variances in (3) can be computed efficiently in hardware using the method proposed in [24].

2) Parallel Interference Cancellation (PIC): With the aid of the previously computed soft-symbols (2), the algorithm considers each of the i streams separately and cancels the interference in $\mathbf{y}$ induced by all other streams $j \neq i$ as follows:

$$\hat{\mathbf{y}}_i = \mathbf{y} - \sum_{j, j \neq i} \mathbf{h}_j \hat{s}_j = \mathbf{h}_i s_i + \tilde{\mathbf{n}}_i \tag{5}$$

where $\tilde{\mathbf{n}}_i = \sum_{j,j\neq i} \mathbf{h}_j e_j + \mathbf{n}$ corresponds to the remaining noise-plus-interference (NPI).

3) MMSE Filter-Vector Computation: In order to reduce the NPI in each $\hat{\mathbf{y}}_i$ of (5), a linear MMSE filter is used. These M_{T} MMSE filter vectors are computed according to [21]

$$\tilde{\mathbf{w}}_i^H = E_{\mathrm{s}} \mathbf{h}_i^H \widetilde{\mathbf{A}}_i^{-1} \tag{6}$$

where

$$\widetilde{\mathbf{A}}_i = \mathbf{H}\widetilde{\mathbf{\Lambda}}_i\mathbf{H}^H + N_0\mathbf{I}_{M_{\mathrm{R}}} \tag{7}$$

and $\widetilde{\mathbf{\Lambda}}_i$ being an $M_{\mathrm{T}} \times M_{\mathrm{T}}$ diagonal matrix having entries

$$\tilde{\Lambda}_{j,j} = \begin{cases} E_j, & j \neq i \\ E_s, & j = i. \end{cases}$$

It is important to realize that (6) requires the inversion of a $M_{\mathrm{R}} \times M_{\mathrm{R}}$-dimensional matrix for each of the M_{T} streams, for each received vector, and for each iteration, which inhibits an efficient implementation in hardware. In order to substantially reduce this computational burden, a novel low-complexity method is proposed in Section III.

[3] The LLRs are initialized as $L_{i,b}^{\mathrm{A}} = 0$, $\forall i, b$, in the first iteration.

4) MMSE Filtering: The MMSE filter vectors in (6) are then used to reduce the NPI present in the PIC vectors $\mathbf{y}_i$ in (5). The ith result of this filtering process corresponds to

$$\tilde{z}_i = \tilde{\mathbf{w}}_i^H \hat{\mathbf{y}}_i = \tilde{\mu}_i s_i + \tilde{\mathbf{w}}_i^H \tilde{\mathbf{n}}_i \tag{8}$$

with $\tilde{\mu}_i = \tilde{\mathbf{w}}_i^H \mathbf{h}_i$.

5) LLR Computation: The algorithm finally approximates the LLRs $L_{i,b}^{\mathrm{E}}$ by assuming that the M_{T} single-input single-output systems in (8) are statistical independent and that the NPI term $\tilde{\mathbf{w}}_i^H \tilde{\mathbf{n}}_i$ is Gaussian distributed with variance

$$\tilde{\nu}_i^2 = \mathrm{Var}[\tilde{z}_i] = \tilde{\mathbf{w}}_i^H \left(\sum_{j,j\neq i} E_j \mathbf{h}_j \mathbf{h}_j^H + N_0 \mathbf{I}_{M_{\mathrm{R}}} \right) \tilde{\mathbf{w}}_i. \tag{9}$$

The resulting intrinsic LLRs are then computed as [21]

$$\tilde{L}_{i,b}^{\mathrm{D}} = \log\left(\sum_{a\in\mathcal{Z}_b^{(1)}} \exp\left(-\frac{|\tilde{z}_i - \tilde{\mu}_i a|^2}{\tilde{\nu}_i^2} + \sum_{b=1}^{Q} \frac{(2[a]_b - 1)}{2} L_{i,b}^{\mathrm{A}} \right) \right) - \log\left(\sum_{a\in\mathcal{Z}_b^{(0)}} \exp\left(-\frac{|\tilde{z}_i - \tilde{\mu}_i a|^2}{\tilde{\nu}_i^2} + \sum_{b=1}^{Q} \frac{(2[a]_b - 1)}{2} L_{i,b}^{\mathrm{A}} \right) \right) \tag{10}$$

where $\mathcal{Z}_b^{(1)}$ and $\mathcal{Z}_b^{(0)}$ refer to the subsets of $\mathcal{O}$, where the bth bit is 1 and 0, respectively. Finally, approximations of the extrinsic LLRs in (1) are computed as $\tilde{L}_{i,b}^{\mathrm{E}} = \tilde{L}_{i,b}^{\mathrm{D}} - L_{i,b}^{\mathrm{A}}$.

III. Low-Complexity SISO MMSE-PIC Algorithm

The SISO MMSE-PIC algorithm described above is not well-suited for an efficient implementation in hardware. In particular, the need for multiple matrix inversions per symbol vector entails high computational complexity and requires considerable arithmetic precision. In order to alleviate these issues, we next detail a variety of techniques enabling the economic implementation of the SISO MMSE-PIC algorithm in VLSI.

A. Exact MMSE-Filter Computation at Low-Complexity

Computation of the M_{T} MMSE filter vectors in (6) requires M_{T} matrix inversions, which poses significant challenges for an efficient (in terms of area and throughput) implementation. To reduce the associated complexity, a variety of methods have been proposed in the literature, e.g., [21], [24]. More precisely, [21] proposes to sequentially perform rank-one updates, which reduces the complexity required to compute all M_{T} inverses $\widetilde{\mathbf{A}}_i^{-1}$ in (6). The approach in [24] requires the computation of M_{T} matrices similar to (7), but only one row of each inverse.

We next describe a novel approach for computation of *all* M_{T} MMSE filter vectors which involves the computation of *one* matrix similar to (7) and requires *a single matrix inversion* only. The derivation is detailed in Appendix A and amounts to computing

$$\mathbf{W}^H = \mathbf{A}^{-1}\mathbf{H}^H \tag{11}$$

where $\mathbf{A} = \mathbf{H}^H\mathbf{H}\mathbf{\Lambda} + N_0\mathbf{I}_{M_{\mathrm{T}}}$ with $\mathbf{\Lambda}$ denoting a $M_{\mathrm{T}} \times M_{\mathrm{T}}$ diagonal matrix having elements $\Lambda_{i,i} = E_i$, $\forall i$. The rows $\mathbf{w}_i^H$ of $\mathbf{W}^H$ correspond to the MMSE filter vectors in (6) up to multiplicative constants. As shown in Appendix A, such a scaling of the MMSE filter vectors does not affect the a-posteriori LLRs (10) delivered by the SISO MMSE-PIC algorithm. Consequently, the MMSE filter vectors $\tilde{\mathbf{w}}_i^H$ can be replaced by $\mathbf{w}_i^H$ without loss in terms of error-rate performance. Furthermore, the computation of $\mathbf{A}^{-1}$ in (11) can be performed in a numerically stable way (see Appendix B) and is therefore well-suited for fixed-point implementation. In summary, our method requires M_{T} times less inverses than the standard approach described in Section II-C3 and also exhibits significantly smaller complexity than the methods proposed in [21], [24].

B. Efficient LLR Computation

1) Efficient NPI-Variance Computation: The first step for reducing the computational complexity associated with LLR computation (10) amounts to simplifying the computation of the NPI variance ν_i^2 (see (9) with $\tilde{\mathbf{w}}_i^H$ being replaced with $\mathbf{w}_i^H$) based on the idea developed in [22]. As shown in Appendix C, ν_i^2 can be simplified to $\nu_i^2 = \mu_i - E_i\mu_i^2$ using $\mu_i = \mathbf{w}_i^H\mathbf{h}_i$, which requires roughly M_{R} times less complex-valued multiplications compared to that required by (9). We furthermore simplify the computation of (8). To this end, we rewrite

$$\frac{|\tilde{z}_i - \tilde{\mu}_i a|^2}{\tilde{\nu}_i^2} \tag{12}$$

in (10) as follows:

$$\rho_i \, |z_i - a|^2 \tag{13}$$

where ρ_i denotes the post-equalization SINR on the ith stream given by

$$\rho_i = \frac{\mu_i^2}{\nu_i^2} = \frac{\mu_i}{1 - E_i\mu_i}$$

and the MMSE filter output z_i is computed as

$$z_i = \frac{\tilde{z}_i}{\tilde{\mu}_i} = \frac{\tilde{\mathbf{w}}_i^H \hat{\mathbf{y}}_i}{\tilde{\mathbf{w}}_i^H \mathbf{h}_i} = \frac{\mathbf{w}_i^H \hat{\mathbf{y}}_i}{\mu_i}.$$

2) Max-Log Approximation: So far, all considered techniques for complexity reduction do not have any impact on the error-rate performance of the algorithm. Additional complexity reductions at the cost of a small performance loss (cf. Section V-B) is achieved by the application of the max-log approximation [11] to (10) with (12) being replaced with (13), which gives

$$\tilde{L}_{i,b}^{\mathrm{D}} \approx \min_{a\in\mathcal{Z}_b^{(-1)}} \left\{ \rho_i \, |z_i - a|^2 - \sum_{b=1}^{Q} \frac{(2[a]_b - 1)}{2} L_{i,b}^{\mathrm{A}} \right\} - \min_{a\in\mathcal{Z}_b^{(+1)}} \left\{ \rho_i \, |z_i - a|^2 - \sum_{b=1}^{Q} \frac{(2[a]_b - 1)}{2} L_{i,b}^{\mathrm{A}} \right\}. \tag{14}$$

Computation of (14) avoids the evaluation of $M_{\mathrm{T}} \cdot 2 \cdot |\mathcal{O}|$ exponential functions and only requires hardware-friendly minimization operations.

Algorithm 1 Low-Complexity SISO MMSE-PIC

1: Compute the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and the matched-filter output $\mathbf{y}^{\mathrm{MF}} = \mathbf{H}^H\mathbf{y}$.

2: For $i = 1, \ldots, M_{\mathrm{T}}$, compute the soft-symbols $\hat{s}_i$ and variances E_i as in detailed Section II-C1.

3: Perform PIC (cf. (5)) based on $\hat{\mathbf{y}}^{\mathrm{MF}}$ according to $\hat{\mathbf{y}}_i^{\mathrm{MF}} = \mathbf{H}^H\hat{\mathbf{y}}_i = \mathbf{y}^{\mathrm{MF}} - \sum_{j,j\neq i}\mathbf{g}_j\hat{s}_j$, $i = 1, \ldots, M_{\mathrm{T}}$, where $\mathbf{g}_j$ denotes the jth column of $\mathbf{G}$.

4: Compute the inverse $\mathbf{A}^{-1} = (\mathbf{G}\mathbf{\Lambda} + N_0\mathbf{I}_{M_{\mathrm{T}}})^{-1}$ in (11) with $\Lambda_{i,i} = E_i$, $i = 1, \ldots, M_{\mathrm{T}}$.

5: Compute the MMSE filter outputs as $z_i = \mu_i^{-1}\mathbf{a}_i^H\hat{\mathbf{y}}_i^{\mathrm{MF}}$, $i = 1, \ldots, M_{\mathrm{T}}$, where $\mathbf{a}_i^H$ is the ith row of $\mathbf{A}^{-1}$ and $\mu_i = \mathbf{a}_i^H\mathbf{g}_i$.

6: Compute the LLRs $\tilde{L}_{i,b}^{\mathrm{E}}$, $i = 1, \ldots, M_{\mathrm{T}}$, $b = 1, \ldots, Q$, according to (15).

3) Omitting the Prior Term: Additional complexity reduction is achieved by omitting the prior term $\sum_{b=1}^{Q}\frac{(2[a]_b-1)}{2}L_{i,b}^{\mathrm{A}}$ in (14). As shown in [26], this approximation does *not* result in a performance loss for Gray-mapped BPSK and 4-QAM constellations, and entails only a small loss for higher-order modulation schemes. Hence, we compute the extrinsic LLRs as

$$\tilde{L}_{i,b}^{\mathrm{E}} = \rho_i\left(\min_{a\in\mathcal{Z}_b^{(0)}}|z_i-a|^2 - \min_{a\in\mathcal{Z}_b^{(1)}}|z_i-a|^2\right). \tag{15}$$

This expression can be rewritten as $\tilde{L}_{i,b}^{\mathrm{E}} = \rho_i\lambda_b(z_i)$ with [27]

$$\lambda_b(z_i) = \min_{a\in\mathcal{Z}_b^{(0)}}|z_i-a|^2 - \min_{a\in\mathcal{Z}_b^{(1)}}|z_i-a|^2 \tag{16}$$

being a piecewise linear function for Gray mappings. Hence, (16) can be obtained efficiently in VLSI (see [16] for implementation details).

C. Avoiding Redundant Computations

In order to avoid recurrent (and hence, redundant) computations, we adopt an idea presented in [24] for our needs. To this end, we compute—prior to detection—the Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$ and the matched-filter output according to $\mathbf{y}^{\mathrm{MF}} = \mathbf{H}^H\mathbf{y}$. The remaining computations are then performed only on the basis of $\mathbf{G}$ and $\mathbf{y}^{\mathrm{MF}}$ instead of on $\mathbf{H}$ and $\mathbf{y}$, which roughly halves the number of (complex-valued) multiplications. The resulting procedure is summarized in Algorithm 1 and referred to as the *low-complexity SISO MMSE-PIC algorithm*.

IV. VLSI Architecture

In this section, we describe a VLSI architecture for the low-complexity SISO MMSE-PIC algorithm and detail the key solutions required to achieve high-throughput with low area.

A. Architectural Overview

In order to achieve high throughput, we decided to partition Algorithm 1 into eight subtasks which are executed in a parallel and (coarse-grained) pipelined fashion. The high-level VLSI architecture—along with the partitioning of the involved computations—is depicted in Figure 3. The architecture consists of eight processing units (PUs), where the six processing steps of Algorithm 1 are mapped to the PUs as shown in Figure 3. The advantages of this architectural structure are: i) It enables to achieve a high sustained throughput and ii) each PU can be designed, optimized, and verified separately, eventually requiring low development and verification time.

The channel matrix $\mathbf{H}$, the received vector $\mathbf{y}$, the a-priori LLRs $L_{i,b}^{\mathrm{A}}$, and the noise variance N_0 are fed to the input of the detector. Each PU performs the assigned computations in T_{s} clock cycles and the results of each unit are passed to the subsequent PU(s) (or to the output of the detector) every T_{s}th clock cycle, which is referred to as the "exchange-cycle." Consequently, the architecture decodes six receive vectors concurrently and in a pipelined manner. Every T_{s}th cycle, the detector delivers a new set of $M_{\mathrm{T}}Q$ LLR values $\tilde{L}_{i,b}^{\mathrm{E}}$, resulting in a sustained throughput of

$$\Theta = \frac{M_{\mathrm{T}}Q}{T_{\mathrm{s}}}f_{\mathrm{clk}} \quad \text{[bit/s]}. \tag{17}$$

Since (17) scales linearly in the clock frequency f_{clk}, the throughput of the detector is maximized by minimizing the length of the critical path of the whole design. In order to arrive at low silicon area while being able to exceed the 600 Mb/s peak data-rate of IEEE 802.11n in 90 nm CMOS technology, we chose $T_{\mathrm{s}} = 18$. This choice results in a processing latency of 108 clock cycles and requires only 450 MHz to achieve the target throughput of 600 Mb/s.

The control unit (see Figure 3) handles the synchronization between each PU and controls the input/output interface of the detector. In order to operate the PUs with more than 450 MHz without the need of an on-chip PLL, we feed two clock signals (with 90° phase offset) into the decoder, which are then used to generate—with the aid of an XOR-gate—an internal clock signal of twice the frequency (see Figure 3). To reduce dynamic power consumption in case that no data-frame needs to be processed, the clock of each PU can be gated individually.

B. Architecture of the Processing Units

All PUs share the same architectural principle and perform their assigned tasks in a time-shared fashion. The basis architecture is depicted in Figure 4 and consists of a finite state-machine (FSM) controlling the data memory, a task-specific set of arithmetic units (AUs), and an interconnection network distributing the memory contents in parallel fashion to all AUs. In order to maximize the clock frequency and to minimize circuit area, the detector exclusively employs fixed-point arithmetic. The AU and memory-internal word-lengths are optimized with the aid of numerical simulations (cf. Section V-B). The feed-through capability available in each PU allows for a parallel transfer of all the data-memory contents from one PU to the subsequent PU(s) during the exchange-cycle. Moreover,

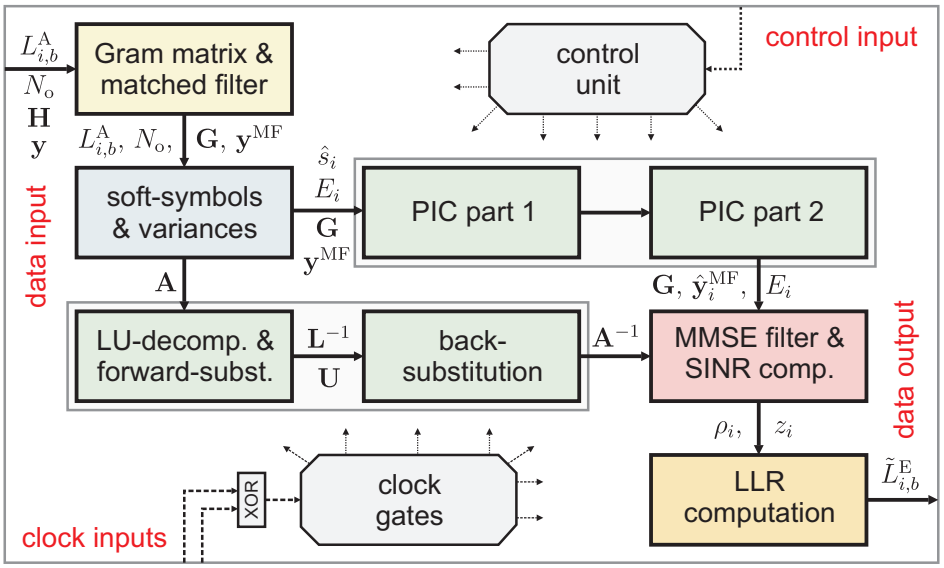

Fig. 3. High-level VLSI architecture of the low-complexity SISO MMSE-PIC detector.

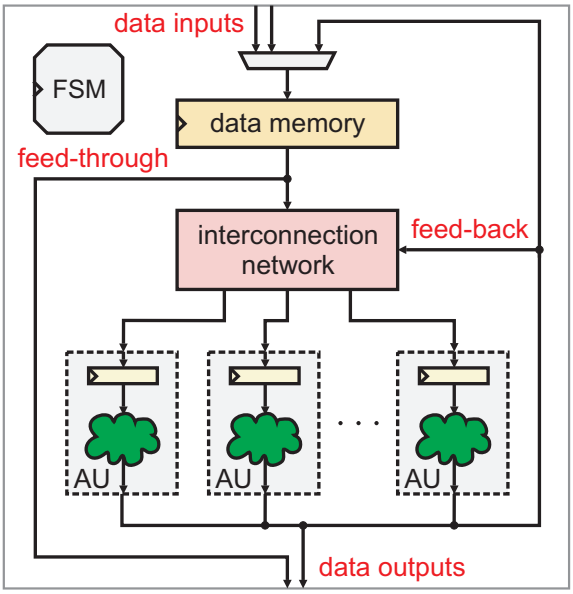

Fig. 4. Architectural principle of the processing units (PU).

a feed-back path (see Figure 4) enables the AUs to immediately use computation results in the subsequent processing cycle. Additional reduction of the critical path by $1/3$ is achieved through the insertion of pipeline registers at the input of each AU (see Figure 4). Moreover, some AUs also pass computation results during the exchange-cycle to the next PU, which reduces the number of idle AUs.

1) Arithmetic Units (AUs): The total set of AUs used in the detector corresponds to (complex-valued) adders, multipliers, arithmetic shifters (mainly used for improving numerical precision), reciprocal units computing $1/x$, and look-up tables (required in the reciprocal units and for approximation of $\mathrm{P}[x_{i,b}=k]$ in (4)). The set of AUs instantiated in each PU is determined such that all required operations can be completed in exactly $T_\mathrm{s}=18$ clock cycles. Table I shows the breakdown of AUs and memory requirements for each PU in order to quantify their complexity requirements. One can immediately observe that the AUs are distributed in a balanced way over all PUs (except for both PIC units and the LLR computation unit), which confirms the effectiveness of the algorithm-partitioning scheme shown in Figure 3.

2) Data Memories: The data memories store all intermediate values, vectors, and matrices. In order to support a high memory bandwidth and to enable parallel access to multiple data words in an irregular manner, the memories are formed by arrays of flip-flops instead of using on-chip S-RAM macrocells. We emphasize that such an implementation choice may lead to slightly sub-optimal results from a silicon-area perspective, but eventually improves the throughput and simplifies placing and routing of the design. Note that the storage requirements in each PU are rather low (see Table I) and hence, the area overhead due to peripheral circuitry present in S-RAM macrocells further reduces the advantages of S-RAMs in this architecture [28].

C. LU-Decomposition-Based Matrix Inversion

The main computational burden remaining in the low-complexity SISO MMSE-PIC algorithm corresponds to computation of $\mathbf{A}^{-1}$ in (11). Since high-throughput matrix inversion using fixed-point arithmetic is a challenging task, a variety of solutions for MIMO systems have been proposed in the literature, e.g., [29]–[31]. As it was noted in [32], inversion based on the LU-decomposition (LUD) exhibits—among the popular matrix inversion algorithms used for MIMO detection—the smallest number of arithmetic operations (even though no LUD-based architecture for MIMO systems has been described in the open literature). This complexity advantage led to the decision to employ an LUD-based matrix-inversion procedure in our design.

The PU "LU-decomp. & forward-subst." (see Figure 3) computes the LUD $\mathbf{A}=\mathbf{LU}$, with $\mathbf{L}$ and $\mathbf{U}$ being $M_\mathrm{T}\times M_\mathrm{T}$-dimensional complex-valued lower-triangular with $L_{i,i}=1$,

TABLE I
NUMBER OF ARITHMETIC UNITS AND MEMORY REQUIREMENTS FOR EVERY PU

Arithmetic unit	add.	mult.	shift	LUT	recip.	mem. [kBit]
Gram matrix & matched filter	16	16	0	0	0	2.09
Soft-symbols & variances	8	8	7	3	0	0.44
PIC part 1 and 2	8	4	0	0	0	3.36
LU-decomposition & forward-substitution	6	10	2	1	1	1.41
Back-substitution	11	10	6	0	0	1.49
MMSE filtering & SINR computation	12	12	4	1	1	2.17
LLR computation	3	3	9	0	0	0.58
Total	64	63	28	5	2	14.52

$\forall i$, and $M_\mathrm{T} \times M_\mathrm{T}$ upper-triangular, respectively, using the in-place LUD algorithm described in [33]. Straightforward LUD-based matrix inversion amounts to inverting $\mathbf{L}$ and $\mathbf{U}$ separately, followed by computation of $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$. This approach is, however, not efficient in terms of the involved number of arithmetic operations and therefore, a more economic method is employed. In particular, the PU computing the LUD additionally solves $\mathbf{L}\mathbf{v}_i = \mathbf{e}_i$ for $\mathbf{v}_i$, $i = 1, \ldots, M_\mathrm{T}$, where $\mathbf{e}_i$ denotes the ith unit vector, using a forward-substitution procedure. Then, the subsequent PU performs back-substitution $\mathbf{U}\mathbf{x}_i = \mathbf{v}_i$ for $\mathbf{x}_i$, $i = 1, \ldots, M_\mathrm{T}$, which finally yields the desired inverse, i.e., $\mathbf{A}^{-1} = [\, \mathbf{x}_1 \cdots \mathbf{x}_{M_\mathrm{T}} \,]$, at low complexity. We note that the LUD, and both forward- and back-substitution, only require additions, multiplications, and the computation of reciprocals.

D. Newton-Raphson-Based Reciprocal Unit

At various steps of the algorithm (i.e., for the LUD and the computation of the SINR ρ_i) division operations are required. Such operations are, in general, not well-suited for fixed-point implementation and off-the-shelf division circuits usually entail a large area, a large number of clock cycles, or a long critical path. Since our goal was to maximize the clock frequency of the detector, we decided to build a custom AU that is able to compute reciprocals $1/x$ at high throughput with a precision that keeps the implementation loss sufficiently small.

1) Algorithm: Implementations of reciprocal units commonly consist of a look-up table (LUT) generating an initial guess for $1/x$ and a subsequent arithmetic circuitry performing a *small* number of Newton-Raphson iterations, e.g., [34], [35]. The procedure employed here starts by shifting the input value x according to $\tilde{x} = 2^\alpha x$, $\alpha \in \mathbb{Z}$, such that $0.5 \leq \tilde{x} < 1$. Since $1 < 1/\tilde{x} \leq 2$, the subsequent computations can be carried out with improved numerical stability.[4] Based on an initial guess $\tilde{x}_0$ of $1/\tilde{x}$ obtained from a LUT, K Newton-Raphson iterations according to $\tilde{x}_{k+1} \leftarrow 2\tilde{x}_k - \tilde{x}\tilde{x}_k^2$, $k = 1, \ldots, K$, are performed; the final result $\tilde{x}_K$ corresponds to an approximation of $1/\tilde{x}$.

2) Architectures: In our design, the LUD needs to be computed in exactly $T_\mathrm{s} = 18$ clock cycles. Hence, the reciprocal unit was allowed to consume at most three clock cycles per reciprocal-value computation. In addition, our simulations have shown that 15 bit precision (excluding the initial shift) is sufficient to arrive at a negligible implementation loss (cf. Figure 7). During the evaluation of potential architectures, we arrived at two solutions meeting the given constraints. Both architectures are depicted in Figure 5 and have been synthesized in 90 nm CMOS technology at maximum speed. The sequential architecture (7.1 kGE[5] area; 1.73 ns critical path) performs two Newton-Raphson iterations and requires a 4 bit LUT. The pipelined architecture (17.7 kGE area; 1.22 ns critical path) performs a single iteration, but requires a 8 bit LUT and additional pipeline registers, eventually requiring 2.5× larger area than the sequential architecture. Since the ultimate design goal was to maximize the clock frequency of the whole detector, we implemented the pipelined architecture. This decision finally moved the critical path of the whole detector to a 24 bit×28 bit multiplier in the back-substitution PU.

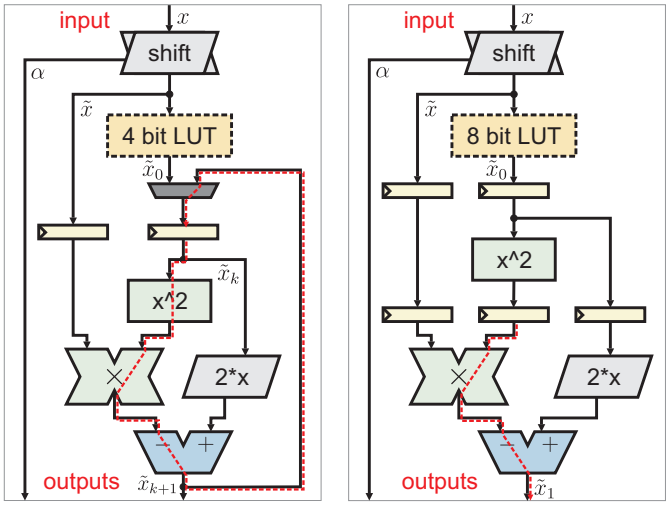

Fig. 5. Architecture of the sequential (left) and pipelined (right) Newton-Raphson-based reciprocal unit. The critical path of both architectures has been highlighted with a dashed line.

V. IMPLEMENTATION RESULTS

The low-complexity SISO MMSE-PIC algorithm, along with all improvements detailed in the previous sections, was fabricated in 90 nm (1P/9M) CMOS technology. Figure 6 shows the chip micrograph with highlighted PUs.[6] The ASIC is com-

[4]Note that rescaling by $2^{-\alpha}$ is performed at later stages in the algorithm with the aid of arithmetic shifters.

[5]One gate equivalent (GE) corresponds to a 2-input drive-1 NAND gate.

[6]Due to library constraints, no signal routing is used on the 9th metal layer.

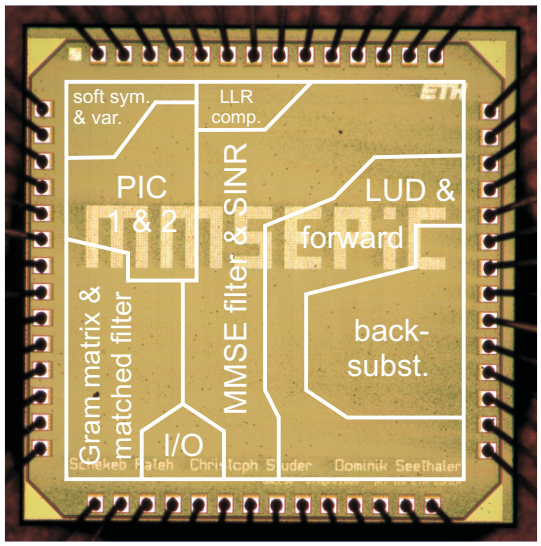

Fig. 6. SISO MMSE-PIC chip micrograph with highlighted PUs (I/O refers to logic required for the input/output interface of the chip).

TABLE II
DETAILED AREA AND POWER BREAKDOWN OF THE ASIC

Processing unit	kGE	%	mW	%
Gram matrix & matched filter	50.4	12.3	21.6	11.5
Soft-symbols & variances	34.4	8.4	13.9	7.3
PIC part 1 and 2	38.4	9.4	34.1	18.0
LU-decomp. & forward-subst.	68.1	16.6	34.7	18.4
Back-substitution	70.2	17.1	15.4	8.1
MMSE filtering & SINR comp.	112.4	27.4	50.5	26.7
LLR computation	10.3	2.5	2.6	1.4
Miscellaneous[a]	26.0	6.3	16.3[b]	8.6
Total	410.2	100	189.1	100

[a]Denotes logic used for the input/output-interface of the chip.
[b]Including 6.94 mA leakage current.

pliant to the IEEE 802.11n WLAN standard [3], includes necessary preprocessing circuitry, and supports SISO detection of four spatial streams (i.e., $M_{\mathrm{T}} = M_{\mathrm{R}} = 4$) with BPSK, QPSK, 16-QAM, and 64-QAM modulation.

A. ASIC Implementation Results

The fabricated chip has the following key characteristics (see also the comparison in Table IV).[7] Its core area is 1.5 mm^2 (at 86% cell density) corresponding to 410 kGE. A detailed area breakdown of the ASIC is shown in Table II. The PU performing MMSE filtering and SINR computation requires roughly 25% of the total silicon area. An additional 25% is occupied by both PUs performing the LUD and the back-substitution procedure. Hence, the single matrix-inversion method proposed in Section III-A reduces the total silicon area by roughly a factor of two compared to a straightforward implementation computing four inverses.

[7]All measurement results (for maximum clock-frequency and power consumption) were carried out on an HP 83 000 F660 VLSI test system. Stimuli and expected responses were generated off-line in Matlab. The design's full-scan capability was used to verify that all ASIC prototypes were fabricated without errors.

TABLE III
COMPARISON WITH OTHER 4×4 MATRIX-INVERSION CIRCUITS

Publication	This work	[29]	[30]	[31]
Cell area [kGE]	223.1[a]	89	120	383
Throughput [M inv./s]	31.5	4.6[b]	3.2[b]	6.0[b]
kGE/(M inv/s)	7.1	19.3	37.5	63.8

[a]Including Gram matrix & matched filter, soft-symbols & variances, LU-decomposition & forward-substitution, and back-substitution PUs.
[b]Throughput scaled to 90 nm CMOS technology.

1) Throughput: The maximum (internal) clock frequency is 568 MHz, which results in a peak throughput of 757 Mb/s per iteration (measured for uncoded four-stream transmission using 64-QAM).[8] Hence, the detector achieves the (rate 5/6-coded) 600 Mb/s peak data-rate specified in IEEE 802.11n with margin.[9]

2) Power Consumption: The power consumption[10] of the ASIC is 189.1 mW, leading to an energy-efficiency of 0.25 nJ/bit per iteration; the leakage current is 6.94 mA at nominal voltage. The power breakdown[11] in Table IV shows that the dynamic power consumption of each PU is roughly proportional to the silicon area. Therefore, the "MMSE filtering & SINR comp." PU and both PUs performing the LUD and back-substitution are the most power-consuming components of the chip.

3) LUD-Based Matrix Inversion: In order to highlight the effectiveness of the proposed LUD-based matrix inversion, we separately compare the achieved performance to other (dedicated) inversion circuits for MIMO systems reported in the literature [29]–[31]. For our implementation, we consider the area of the PUs required to compute the matrix inversion. The throughput is measured in 4×4 matrix-inversions per second, given by $T_{\mathrm{s}}^{-1} f_{\mathrm{clk}}$. Table III shows that our implementation outperforms all other solutions in terms of throughput (measured in million inverses per second) by at least 5×. With respect kGE per throughput, our inversion method is 2.7× more efficient than the second best solution [29].

B. Fixed-Point Error-Rate Performance

In order to achieve near-optimal error-rate performance with fixed-point arithmetic, the word-lengths in the SISO MMSE-PIC architecture have been optimized using numerical simulations. The key parameters are as follows: We use 5 bit and 6 bit for the input and output LLRs, respectively. The real

[8]The throughput of a SISO detector decreases linearly with the number of iterations [36], e.g., for $I = 2$, our implementation achieves 378.5 Mb/s. Hence, in order to perform two iterations in a IEEE 802.11n-compliant transceiver, either two SISO MMSE-PIC instances are required or the iterative detection feature may only be used for the 20 MHz bandwidth mode specifying only 288.9 Mb/s throughput.
[9]The detector achieves a rate 5/6-coded throughput of 631 Mb/s/iteration.
[10]Measured at maximum throughput, V_{dd} =1.2 V core supply, $T = 300$ K, and using typical stimuli (generated at 15 dB SNR for 16-QAM with i.i.d. Rayleigh fading channel matrices). Note that for the power measurements presented in [1], i.i.d. uniformly distributed bits have been supplied to the input of the detector.
[11]The individual power results have been measured by exploiting the clock-gating capability available for each PU.

TABLE IV
ASIC IMPLEMENTATION RESULTS AND COMPARISON TO OTHER REPORTED MIMO DETECTORS

Publication	This work	Witte *et al.* [14]	Wenk *et al.* [7]	Burg *et al.* [10]	Liao *et al.* [6]	Wenk *et al.* [7]	Shabany and Gulak [8]	Liu *et al.* [9]
Detection algorithm	SISO MMSE-PIC	SISO STS-SD	soft-output STS-SD	soft-output MMSE	soft-output MBF-FD	hard-output sphere decoder	hard-output k-Best	hard-output k-Best
Iterative MIMO decoding	yes	yes	no	no	no	no	no	no
Constant throughput	yes	no	no	yes	no	no	yes	yes
CMOS technology [nm]	90	90	130	130	130	130	130	130
Clock frequency [MHz]	568	190	383	320	198	455	270	137.5
Core area [mm^2]	1.5	–	–	–	1.77 (0.85[a])	–	–	3.9 (1.87[a])
Preprocessing area [kGE] Detection area [kGE]	410[e]	– 185	– 97.1	251 67	– 350	– 38.4	– 114	– 491
Max. throughput [Mb/s]	757	45.6[b]	91.1[b] (132.8[a,b])	960 (1386[a])	431.8[c] (631.7[a,c])	1092[d] (1577[a,d])	655 (946[a])	1100 (1589[a])
Normalized hardware-efficiency [kGE/(Mb/s)]	0.54[e]	4.06[b]	0.73[a,b]	0.23[a]	0.56[a,c]	0.024[a,d]	0.12[a]	0.31
Power consumption in [mW] at [Mb/s]	189.1[e] at 757	–	–	–	58.2 (34.3[a]) at 431.8[c]	–	135 (79.6[a]) at 655	–
Energy-efficiency [nJ/bit]	0.25[e]	–	–	–	0.13[c] (0.08[a,c])	–	0.2 (0.12[a])	0.115 (0.08[a])

[a]Technology scaling to 90 nm CMOS technology assuming: $A \sim 1/s^2$, $t_{\mathrm{pd}} \sim 1/s$, and $P_{\mathrm{dyn}} \sim (1/s)(V_{\mathrm{dd}}/V'_{\mathrm{dd}})$.
[b]Assuming $M_{\mathrm{T}} = M_{\mathrm{R}} = 4$ using 64-QAM and evaluating 100 nodes per vector.
[c]Throughput is only achieved under "good channel conditions" [6] and hence, representing optimistic performance.
[d]Assuming $M_{\mathrm{T}} = M_{\mathrm{R}} = 4$ using 64-QAM and evaluating 10 nodes per vector.
[e]Area and power figures of the SISO MMSE-PIC chip include the necessary preprocessing circuitry.

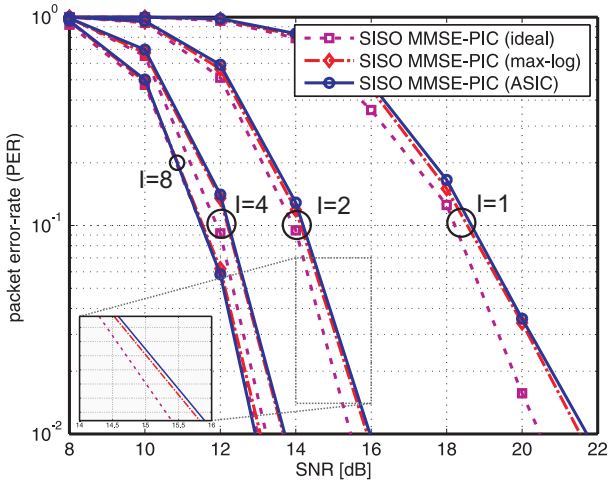

Fig. 7. PER versus SNR performance of the SISO MMSE-PIC ASIC.

and imaginary parts of the channel matrix $\mathbf{H}$ and the received vector $\mathbf{y}$ are represented with 14 bit and 16 bit, respectively. The largest word-length within the design resides in the back-substitution PU and corresponds to 28 bit.

Figure 7 compares the PER versus SNR performance of iterative MIMO decoding using the ideal (floating-point) algorithm as in [23], the low-complexity SISO MMSE-PIC algorithm detailed in Algorithm 1, and the corresponding (fixed-point) ASIC implementation. The same simulation settings as in Section II-B are used here. Note that for $I = 1$, SISO MMSE-PIC detection coincides with soft-output MMSE detection as used in [10]. One can observe a 0.5 dB to 1 dB performance degradation resulting from the max-log approximation and omitting the prior terms (see Sections III-B2 and III-B3). The implementation loss compared to the (floating-point) max-log detector is, however, less than 0.2 dB SNR, which highlights results of the fixed-point optimizations carried out during the design of the ASIC. Finally, one can observe an impressive SNR performance gain of more than 8 dB SNR compared to (non-iterative) soft-output MMSE detection at only four iterations, which enables us to conclude that using the SISO MMSE-PIC in combination with iterative MIMO decoding enables significant SNR-gains in practical system-scenarios.

C. Comparison to Recent MIMO-Detector Implementations

Table IV provides a comparison of our implemented SISO MMSE-PIC ASIC with the synthesis results of the SISO STS-SD algorithm presented in [14] and other state-of-the-art *non-iterative* MIMO detector implementations [6]–[10]. For all designs, we considered the implementation variants designed for four-stream MIMO detection supporting 64-QAM.

1) Throughput: For the throughput figures reported in Table IV, we note that the implementations [6], [7], [14] achieve non-constant throughput, i.e., the decoding effort strongly depends on the SNR and the realization of the channel-matrix. Moreover, their worst-case complexity is, in general, very high, which leads to a low aggregated throughput. In order to arrive at a fair throughput comparison, we assumed that the SISO and soft-output SD-based implemen-

tations [7], [14] visit at most 100 nodes (e.g., by enforcing run-time constraints [37]), leading to optimistic results for 64-QAM (see [7] for details). Since hard-output SD roughly requires one magnitude lower complexity than soft-output SD [37], we assume [7] to visit a maximum of 10 nodes per decoded vector. For the design in [6], we used the throughput figures stated in the paper, which are, however, only achieved under "good channel conditions" and therefore, represent optimistic performance. In summary, the given throughput results favor the implementations [6], [7], [14].

We can observe from Table IV that the throughput achieved by the proposed SISO MMSE-PIC implementation is comparable to that achieved by the detectors proposed in [6], [8] and only about 50% smaller than that of [9], [10]. It is, however, important to note that our proposed implementation, as in contrast to the implementations of [6], [8]–[10], supports iterative MIMO decoding. Compared to the synthesis results of the soft-output SD in [7] and the only SISO detector reported so far [14], our detector achieves $8\times$ and $16\times$ higher throughput, respectively.

2) Silicon Area: For the silicon area results reported in Table IV, it is important to note that our low-complexity SISO MMSE-PIC implementation and the soft-output MMSE detector implementation proposed in [10] contain the necessary preprocessing circuitry, i.e., the operations required to decompose the channel matrix prior to detection. All other implementations shown in Table IV require an additional QR-decomposition of $\mathbf{H}$, which roughly entails an additional 250 kGE for IEEE 802.11n-compliant receivers (cf. [10] in Table IV). The missing preprocessing circuitry in [6]–[9], [14] is therefore reflected in lower silicon area, improved hardware-efficiency, lower power consumption, and optimistic energy-efficiency figures. If we incorporate the additional preprocessing area of 250 kGE for these designs, the silicon area of the proposed low-complexity SISO MMSE-PIC implementation is in-between that of the detector implementations in [7], [8], [10] and those reported in [6], [9], [14].

3) Hardware-Efficiency: In terms of hardware-efficiency, we attain comparable results as in [6], [7]. We can, hence, conclude that our design is competitive to recently reported soft-output MIMO detectors (which are unable to support iterative MIMO decoding and do not include the necessary preprocessing circuitry). Compared to the SISO STS-SD implementation [14], we achieve $8\times$ better hardware efficiency, which is a result of the low throughput achieved by SD-based methods in combination with 64-QAM.

4) Energy-Efficiency: Our detector exhibits $3\times$ and $2\times$ worse energy-efficiency compared to the designs in [6], [9] and the design in [8], respectively. The reason for this penalty is mainly due to i) the preprocessing circuitry present in our design and ii) the fact that the SISO MMSE-PIC ASIC supports iterative MIMO decoding, whereas all other designs reporting energy-efficiency [6], [8], [9] are non-iterative.

VI. Conclusion

In this paper, we presented a novel low-complexity soft-input soft-output (SISO) minimum mean-square error (MMSE) parallel interference cancellation (PIC) algorithm for systems employing iterative MIMO decoding. We described a corresponding VLSI architecture and implemented the first ASIC of a SISO detector in 90 nm CMOS technology. The fabricated chip includes all the necessary channel-matrix preprocessing operations and exceeds the 600 Mb/s peak data-rate of the IEEE 802.11n WLAN standard.

The SISO MMSE-PIC ASIC was shown to enable significant SNR gains over recently reported non-iterative MIMO-detector implementations, while being competitive with respect to achievable throughput and silicon area. These SNR-performance gains ultimately lead to improved link reliability and system throughput in practical system-scenarios. In summary, the presented MIMO detector represents the next step towards achieving close-to-optimal performance in wireless MIMO communication systems and is therefore well-suited for transceiver designs where outstanding throughput and excellent link-reliability are the ultimate design goals.

Acknowledgements

The authors would like to thank A. Burg, N. Felber, F. Gürkaynak, H. Kaeslin, and M. Wenk for their assistance during the ASIC design. Furthermore, the authors gratefully acknowledge the support from H. Bölcskei, W. Fichtner, and Q. Huang.

Appendix A
Single Matrix Inversion

In this section, we show that the standard MMSE filter vectors $\tilde{\mathbf{w}}_i^H$ in (6) (see also [21]) can be replaced by

$$\mathbf{w}_i^H = \mathbf{h}_i^H \Big(\mathbf{H} \mathbf{\Lambda} \mathbf{H}^H + N_0 \mathbf{I}_{M_\mathrm{R}} \Big)^{-1} \tag{18}$$

without changing the output of the SISO MMSE-PIC algorithm. Consequently, the M_T matrix inversions required to compute the vectors $\mathbf{w}_i^H$ can be replaced by a *single* matrix inversion.

The proof is accomplished by first showing that $\mathbf{w}_i^H = \tilde{\mathbf{w}}_i^H c_i$ with c_i being a real-valued constant and by subsequently proving that the scaling with c_i of $\tilde{\mathbf{w}}_i^H$ does not change the output of the SISO MMSE-PIC algorithm. To this end, we state the Sherman-Morrison formula [38]

$$\Big(\widetilde{\mathbf{A}}_i + \mathbf{u}\mathbf{v}^H \Big)^{-1} = \widetilde{\mathbf{A}}_i^{-1} - \frac{\widetilde{\mathbf{A}}_i^{-1} \mathbf{u}\mathbf{v}^H \widetilde{\mathbf{A}}_i^{-1}}{1 + \mathbf{v}^H \widetilde{\mathbf{A}}_i^{-1} \mathbf{u}}. \tag{19}$$

with $\widetilde{\mathbf{A}}_i$ given in (7). We furthermore set $\mathbf{u}\mathbf{v}^H = \mathbf{h}_i \Delta_i \mathbf{h}_i^H$, where $\mathbf{u} = \mathbf{h}_i$ and $\Delta_i = E_i - E_\mathrm{s}$. Note that (19) corresponds to $\mathbf{A}^{-1}$ in (18). With the definition (18) we obtain

$$\mathbf{w}_i^H = \mathbf{h}_i^H \widetilde{\mathbf{A}}_i^{-1} - \frac{\mathbf{h}_i^H \widetilde{\mathbf{A}}_i^{-1} \mathbf{h}_i \Delta_i \mathbf{h}_i^H \widetilde{\mathbf{A}}_i^{-1}}{1 + \Delta_i \mathbf{h}_i^H \widetilde{\mathbf{A}}_i^{-1} \mathbf{h}_i}. \tag{20}$$

Using (6) enables us to rewrite (20) according to

$$\mathbf{w}_i^H = {E_\mathrm{s}}^{-1} \tilde{\mathbf{w}}_i^H \left(\frac{1 + \Delta_i {E_\mathrm{s}}^{-1} \tilde{\mathbf{w}}_i^H \mathbf{h}_i - \Delta_i {E_\mathrm{s}}^{-1} \mathbf{h}_i^H \tilde{\mathbf{w}}_i}{1 + \Delta_i {E_\mathrm{s}}^{-1} \tilde{\mathbf{w}}_i^H \mathbf{h}_i} \right).$$

Furthermore, by noting that the terms $\tilde{\mathbf{w}}_i^H \mathbf{h}_i$ and $\mathbf{h}_i^H \tilde{\mathbf{w}}_i$ are real-valued, we have $\tilde{\mathbf{w}}_i^H \mathbf{h}_i = \mathbf{h}_i^H \tilde{\mathbf{w}}_i$, which leads to $\mathbf{w}_i^H =$

$\tilde{\mathbf{w}}_i^H c_i$ with $c_i = 1/\left(E_\mathrm{s} + \Delta_i \tilde{\mathbf{w}}_i^H \mathbf{h}_i\right)$ being a real-valued and stream-dependent constant.

We next show that a scaling of $\tilde{\mathbf{w}}_i^H$ does not change the LLRs provided by the SISO MMSE-PIC algorithm. The only term in the computation of the SISO MMSE-PIC LLRs (10) that depends on the MMSE filter vector $\tilde{\mathbf{w}}_i^H$ is

$$\frac{\left|\tilde{z}_i - \tilde{\mu}_i a\right|^2}{\tilde{\nu}_i^2} = \frac{\left|\tilde{\mathbf{w}}_i^H \hat{\mathbf{y}}_i - \tilde{\mathbf{w}}_i^H \mathbf{h}_i a\right|^2}{\tilde{\mathbf{w}}_i^H \left(\sum_{j,j\neq i} E_j \mathbf{h}_j \mathbf{h}_j^H + N_0 \mathbf{I}\right) \tilde{\mathbf{w}}_i}.$$

which can equivalently be written as

$$\frac{\left|\tilde{z}_i - \tilde{\mu}_i a\right|^2}{\tilde{\nu}_i^2} = \frac{\left|c_i \tilde{\mathbf{w}}_i^H \hat{\mathbf{y}}_i - c_i \tilde{\mathbf{w}}_i^H \mathbf{h}_i a\right|^2}{c_i \tilde{\mathbf{w}}_i^H \left(\sum_{j,j\neq i} E_j \mathbf{h}_j \mathbf{h}_j^H + N_0 \mathbf{I}\right) \tilde{\mathbf{w}}_i c_i^*}$$

for any real-valued and stream-dependent constant c_i, which concludes the proof.

We finally note that (18) can be used to obtain all M_T MMSE filter vectors *concurrently*. To this end, replace the $\mathbf{h}_i^H$ in (18) by $\mathbf{H}^H$ and compute

$$\mathbf{W}^H = \mathbf{H}^H \left(\mathbf{H}\mathbf{\Lambda}\mathbf{H}^H + N_0 \mathbf{I}_{M_\mathrm{R}}\right)^{-1} \tag{21}$$

where $\mathbf{W}^H = [\,\mathbf{w}_1 \cdots \mathbf{w}_{M_\mathrm{T}}\,]^H$ contains *all* MMSE filter vectors on its rows.

Appendix B
Numerically Stable Inverse of Small Matrix

We show that $\mathbf{W}^H$ in (21) can be written in the form of (11). In fact, the key drawback of (21) is that an $M_\mathrm{R} \times M_\mathrm{R}$ matrix needs to be inverted while (11) requires a matrix inversion of dimension $M_\mathrm{T} \times M_\mathrm{T}$ only (note that $M_\mathrm{R} \geq M_\mathrm{T}$).

To this end, define $\widetilde{\mathbf{H}} = \mathbf{H}\mathbf{\Lambda}^{\frac{1}{2}}$, where $\mathbf{\Lambda}^{\frac{1}{2}}$ is a real-valued $M_\mathrm{T} \times M_\mathrm{T}$ diagonal matrix with $\Lambda_{i,i}^{\frac{1}{2}} = \sqrt{E_i}$, $\forall i$ and $\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{\Lambda}^{\frac{1}{2}} = \mathbf{\Lambda}$. The matrix in (21) can now be rewritten as

$$\mathbf{W}^H = \mathbf{\Lambda}^{-\frac{1}{2}} \widetilde{\mathbf{H}}^H \left(\widetilde{\mathbf{H}}\widetilde{\mathbf{H}}^H + N_0 \mathbf{I}_{M_\mathrm{R}}\right)^{-1}. \tag{22}$$

Application of the singular-value decomposition [38] to the matrix inverse in (22), leads to [36]

$$\mathbf{W}^H = \mathbf{\Lambda}^{-1} \left(\mathbf{H}^H \mathbf{H} + N_0 \mathbf{\Lambda}^{-1}\right)^{-1} \mathbf{H}^H \tag{23}$$

which only requires to invert an $M_\mathrm{T} \times M_\mathrm{T}$ matrix.

Unfortunately, the formulation in (23) exhibits poor numerical stability. Consider the case where near-perfect a-priori information is available, i.e., the variances $E_i \approx 0$, $\forall i$. In this case, the entries of $\mathbf{\Lambda}^{-1}$ will be arbitrarily large and hence, computation of the MMSE filter matrix would require a prohibitively large dynamic range. We therefore rewrite (23) to [36]

$$\mathbf{W}^H = \left(\mathbf{H}^H \mathbf{H}\mathbf{\Lambda} + N_0 \mathbf{I}_{M_\mathrm{T}}\right)^{-1} \mathbf{H}^H \tag{24}$$

which, in general, exhibits superior numerical stability.

Appendix C
Efficient NPI-Variance Computation

The computation of the NPI variance ν_i^2 (see (9) with $\tilde{\mathbf{w}}_i^H$ being replaced with $\mathbf{w}_i^H$) can be simplified as follows. Using the definition of $\mathbf{\Lambda}$ (cf. (11)), we can write

$$\begin{aligned}\nu_i^2 &= \mathbf{w}_i^H \left(\mathbf{H}\mathbf{\Lambda}\mathbf{H}^H - E_i \mathbf{h}_i \mathbf{h}_i^H + N_0 \mathbf{I}_{M_\mathrm{R}}\right) \mathbf{w}_i \\ &= \mathbf{w}_i^H \left(\mathbf{H}\mathbf{\Lambda}\mathbf{H}^H + N_0 \mathbf{I}_{M_\mathrm{R}}\right) \mathbf{w}_i - E_i \left(\mathbf{w}_i^H \mathbf{h}_i\right)^2 \end{aligned} \tag{25}$$

where $\mathbf{w}_i^H \mathbf{h}_i \in \mathbb{R}$. From (18) it follows that

$$\mathbf{w}_i^H \left(\mathbf{H}\mathbf{\Lambda}\mathbf{H}^H + N_0 \mathbf{I}_{M_\mathrm{R}}\right) = \mathbf{h}_i^H.$$

Consequently, (25) can be simplified to

$$\nu_i^2 = \mathbf{h}_i^H \mathbf{w}_i - E_i \left(\mathbf{w}_i^H \mathbf{h}_i\right)^2 = \mathbf{w}_i^H \mathbf{h}_i - E_i \left(\mathbf{w}_i^H \mathbf{h}_i\right)^2.$$

References

[1] C. Studer, S. Fateh, and D. Seethaler, "A 757 Mb/s 1.5 mm^2 90 nm CMOS soft-input soft-output MIMO detector for IEEE 802.11n," in *Proc. IEEE Europ. Solid State Circuits Conf. (ESSCIRC)*, Sevilla, Spain, Sept. 2010, pp. 530–533.

[2] A. Paulraj, R. Nabar, and D. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge Univ. Press, 2003.

[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] A. Burg, M. Wenk, M. Zellweger, M. Wegmueller, N. Felber, and W. Fichtner, "VLSI implementation of the sphere decoding algorithm," in *Proc. IEEE Europ. Solid State Circuits Conf. (ESSCIRC)*, Leuven, Belgium, Sept. 2004, pp. 303–306.

[6] C.-H. Liao, T.-P. Wang, and T. D. Chiueh, "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.

[7] M. Wenk, L. Bruderer, A. Burg, and C. Studer, "Area- and throughput-optimized VLSI architecture of sphere decoding," in *Proc. IEEE/IFIP Int. Conf. VLSI and System-on-Chip (VLSI SoC)*, Madrid, Spain, Sept. 2010.

[8] M. Shabany and P. G. Gulak, "A 0.13 µm CMOS, 655 Mb/s 4×4 64-QAM k-best MIMO detector," in *Dig. Techn. Papers, IEEE ISSCC*, San Francisco, CA, USA, Feb. 2009, pp. 256–257.

[9] L. Liu, F. Ye, X. Ma, T. Zhang, and J. Ren, "A 1.1-Gb/s 115-pJ/bit configurable MIMO detector using 0.13-μm CMOS technology," *IEEE Trans. Circ. Systems II*, vol. 57, no. 9, pp. 701–705, Sept. 2010.

[10] A. Burg, S. Haene, M. Borgmann, D. Baum, T. Thaler, F. Carbognani, S. Zwicky, L. Barbero, C. Senning, P. Greisen, T. Peter, C. Foelmli, U. Schuster, and P. Tejera, "A 4-stream 802.11n baseband transceiver in 0.13 µm CMOS," in *Dig. Techn. Papers, Symp. on VLSI Circuits*, Kyoto, Japan, Jun. 2009, pp. 282–283.

[11] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Comm.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[12] L. Boher, R. Rabineau, and M. Hélard, "FPGA implementation of an iterative receiver for MIMO-OFDM systems," *IEEE J. on Sel. Areas in Comm.*, vol. 26, no. 6, pp. 857–866, Aug. 2008.

[13] C. Studer and H. Bölcskei, "Soft–input soft–output single tree-search sphere decoding," *IEEE Trans. Inf. Th.*, vol. 56, no. 10, pp. 4827–4842, Oct. 2010.

[14] E. M. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr, "A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding," *IEEE Trans. Circ. Systems II*, vol. 57, no. 9, pp. 706–710, Sept. 2010.

[15] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Comm.*, vol. 47, no. 7, pp. 1046–1061, Jul. 1999.

[16] S. Häne, "VLSI circuits for MIMO-OFDM physical layer," Ph.D. dissertation, ETH Zürich, Switzerland, 2008.

[17] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. IEE Int. Conf. on Comm. (ICC)*, Geneva, Switzerland, May 1993, pp. 1064–1070.
[18] J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, San Jose, CA, USA, Sept. 2009, pp. 487–490.
[19] V. Erceg *et al.*, *TGn channel models*, May 2004, IEEE 802.11 document 03/940r4.
[20] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Th.*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
[21] M. Tüchler, A. C. Singer, and R. Koetter, "Minimum mean squared error equalization using a priori information," *IEEE Trans. Sig. Proc.*, vol. 50, no. 3, pp. 673–983, Mar. 2002.
[22] A. Dejonghe and L. Vandendorpe, "Turbo-equalization for multilevel modulation: an efficient low-complexity scheme," in *Proc. IEE Int. Conf. on Comm. (ICC)*, vol. 3, New York City, NY, USA, Apr. 2002, pp. 1863–1867.
[23] M. Witzke, S. Bäro, F. Schreckenbach, and J. Hagenauer, "Iterative detection of MIMO signals with linear detectors," in *Proc. Asilomar Conf. on Signals, Systems and Computers (ACSSC)*, Monterey, CA, USA, Nov. 2002, pp. 289–293.
[24] A. Tomasoni, M. Ferrari, D. Gatti, F. Osnato, and S. Bellini, "A low complexity turbo MMSE receiver for W-LAN MIMO systems," in *Proc. IEE Int. Conf. on Comm. (ICC)*, vol. 9, Istanbul, Turkey, Jun. 2006, pp. 4119–4124.
[25] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Th.*, vol. 42, no. 2, pp. 429–445, Mar. 1996.
[26] S. Fateh, "VLSI implementation of soft-input soft-output MMSE parallel interference cancellation," Master's thesis, Dept. Information Technology and Electical Engineering, ETH Zürich, Mar. 2009.
[27] I. B. Collings, M. R. G. Buttler, and M. McKay, "Low complexity receiver design for MIMO bit-interleaved coded modulation," in *Proc. IEEE 8th Int. Symp. on Spread Spectrum Techniques and Applications (ISSSTA)*, Sydney, Australia, Aug. 2004, pp. 12–16.
[28] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Proc. IEEE 53th Int. Midwest Symp. on Circuits and Systems (MWSCAS)*, Seattle, WA, USA, August 2010, pp. 129–132.
[29] A. Burg, S. Häne, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, Kos, Greece, May 2006, pp. 4102–4105.
[30] J. Eilert, D. Wu, and D. Liu, "Implementation of a programmable linear MMSE detector for MIMO-OFDM," in *IEEE Int. Conf. on Acoustics, Speech, and Sig. Proc. (ICASSP)*, Las Vegas, NV, USA, March 2008, pp. 5396–5399.
[31] S. Eberli, D. Cescato, and W. Fichtner, "Divide-and-conquer matrix inversion for linear MMSE detection in SDR MIMO receivers," in *Proc. IEEE NORCHIP*, Trondheim, Norway, Nov. 2008, pp. 162–167.
[32] S. Eberli, "Application-specific processor for MIMO-OFDM software-defined radio," Ph.D. dissertation, ETH Zürich, Switzerland, Series in Microelectronics, vol. 200, Hartung-Gorre Verlag Konstanz, 2009.
[33] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins Univ. Press, 1996.
[34] B. Gestner and D. V. Anderson, "Single Newton-Raphson iteration for integer-rounded divider for lattice reduction algorithms," in *Proc. IEEE 51th Int. Midwest Symp. on Circuits and Systems (MWSCAS)*, Knoxville, TN, USA, Aug. 2008, pp. 966–969.
[35] R. Zimmermann, "Computer arithmetic: Principles, architectures, and VLSI design," Lecture notes, Integrated Systems Laboratory, Dept. Information Technology and Electrical Engineering, ETH Zürich, Tech. Rep., Mar. 1999.
[36] 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.
[37] C. Studer, A. Burg, and H. Bölcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Comm.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.
[38] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge Univ. Press, 1985.