Efficient VLSI Architectures for Matrix Inversion in Soft-Input Soft-Output MMSE MIMO Detectors

Dominik Auras, Rainer Leupers, Gerd Ascheid
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University, 52056 Aachen, Germany
Email: auras@ice.rwth-aachen.de

***Abstract*—A computational complexity analysis of matrix inversion used in soft-input soft-output minimum mean square error (MMSE) MIMO detectors and a comprehensive literature comparison of corresponding VLSI implementations are presented. They indicate that the application specific integrated circuit (ASIC) proposed in this paper is — to the best of our knowledge — the most area-throughput efficient VLSI architecture reported so far, outperforming the second best by a factor of 1.7x. The ASIC achieves the IEEE 802.11n standard's peak data rate of 600 Mbit/s.**

I. Introduction

Bit-interleaved coded modulation with iterative decoding (BICM-ID) promises impressive communication performance gains. To take advantage of these in multi-antenna (MIMO) systems with high data rates, Soft-Input Soft-Output (SISO) MMSE MIMO detection is an economically reasonable near-future option for VLSI implementation. The matrix inversion is the computationally most demanding operation of the MMSE filter matrix computation. Reported circuits for this context can be roughly divided into four algorithmic categories: (a) The input matrix is inverted using a Divide-and-Conquer (D&C) approach [1]–[3]. (b) A series of rank-1 updates iteratively converges from a trivial inverse to the final one [4]. (c) The QR decomposition of a regularized channel matrix is computed [5], or (d) the matrix is decomposed into matrices that are easier to invert [6].

Contribution: We introduce an area-throughput efficient VLSI architecture performing matrix inversion based on a computationally efficient decomposition, and present a comprehensive comparison of matrix inversion circuits.

Outline: Section II gives an introduction to the context of this work. Section III provides the algorithm details and a computational complexity analysis with respect to two other algorithms. In Section IV, the implementation results are discussed and compared to related work.

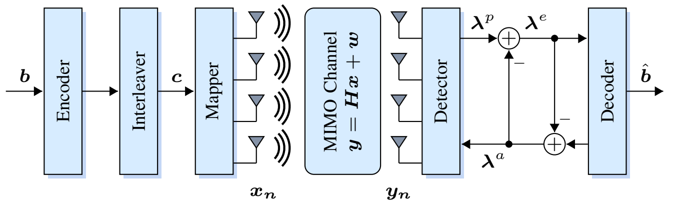

Fig. 1. Assumed MIMO BICM-ID System Model. Detector and decoder iteratively exchange information to improve the final decoding result.

II. System Model

We consider a spatial multiplexing $N_t \times N_r$ MIMO system with BICM-ID, depicted in Fig. 1. A message $\boldsymbol{b} \in \{0,1\}^{N_b}$ is encoded with rate $r = N_b/N_c$ and interleaved, yielding the code word $\boldsymbol{c} \in \{0,1\}^{N_c}$. Let $\mathcal{X} \subset \mathbb{C}$ be a modulation alphabet with $K = \log_2 |\mathcal{X}|$ bits per symbol. The code word is partitioned into N_s subvectors $\boldsymbol{c}_n \in \{0,1\}^{KN_t}$. They are subsequently mapped to symbol vectors $\boldsymbol{x}_n \in \mathcal{X}^{N_t}$ that are transmitted independently. Assuming a frequency-flat fading channel characterized by $\boldsymbol{H}_n \in \mathbb{C}^{N_r \times N_t}$, the received symbol vector at time n is $\boldsymbol{y}_n = \boldsymbol{H}_n \boldsymbol{x}_n + \boldsymbol{w}_n$ where $\boldsymbol{w}_n \in \mathbb{C}^{N_r}$ is a white Gaussian noise process with $\mathbb{E}[\boldsymbol{w}_n \boldsymbol{w}_n^H] = N_0 \boldsymbol{I}_{N_r}$. In the remainder, the time index n is dropped for convenience. Using iterative MIMO decoding following the famous Turbo Principle [9], detector and channel decoder exchange extrinsic information $\boldsymbol{\lambda}^e = \boldsymbol{\lambda}^p - \boldsymbol{\lambda}^a$ in terms of log-likelihood ratios (LLRs), where $\boldsymbol{\lambda}^p$ are the detector's posterior LLRs and $\boldsymbol{\lambda}^a$ are the prior LLRs fed back from the decoder.

A. SISO MMSE MIMO Filter Matrix

We assume a SISO MMSE MIMO detector model derived from [7][1]. The computation of the MMSE filter matrix

$$\boldsymbol{G} = \boldsymbol{C}_{\boldsymbol{y}}^{-1} \boldsymbol{H} = \left(\boldsymbol{H} \boldsymbol{C}_{\boldsymbol{x}} \boldsymbol{H}^H + N_0 \boldsymbol{I}_{N_r} \right)^{-1} \boldsymbol{H} \tag{1}$$

requires the inversion of $\boldsymbol{C}_{\boldsymbol{y}}$, where $\boldsymbol{C}_{\boldsymbol{x}} = \mathrm{diag}\{\sigma_1^2, \ldots, \sigma_{N_t}^2\}$ holds the symbol variances $\sigma_t^2 = f(\boldsymbol{\lambda}^a)$ computed from the decoder feedback $\boldsymbol{\lambda}^a$. The matrix $\boldsymbol{C}_{\boldsymbol{y}}$ is hermitian positive definite (HPD) since $\boldsymbol{x}^H \boldsymbol{C}_{\boldsymbol{y}} \boldsymbol{x} > 0, \forall \boldsymbol{x} \neq \boldsymbol{0}$ and $\boldsymbol{C}_{\boldsymbol{y}}^H = \boldsymbol{C}_{\boldsymbol{y}}$ hold. Thus we apply the well-known LDL decomposition to $\boldsymbol{C}_{\boldsymbol{y}}$. Subsequently we perform the easier inversions of the resulting matrices. This is computationally more efficient as e.g. the LU decomposition (cf. Sec. III) and the other reported algorithms, which could all also be applied. Our approach falls into the fourth algorithmic category mentioned before (d). Note that our VLSI implementation includes the computation of $\boldsymbol{C}_{\boldsymbol{y}}$ from $\boldsymbol{H}$, $\boldsymbol{\lambda}^a$ and N_0, since the evaluation context is important.

In the next section, we introduce the LDL decomposition and analyze its computational complexity. The analysis shows the superior computational efficiency of the proposed approach compared to two algorithms.

III. LDL-based Matrix Inversion

We consider the inversion of the square HPD matrix $\boldsymbol{A} \in \mathbb{C}^{M \times M}$, e.g. $\boldsymbol{A} = \boldsymbol{C}_{\boldsymbol{y}}$ from (1), which can also be used

[1]We do not perform iterations between equalizer and demapper as in [7].

in soft-output (SO) only MMSE detection [1]–[4]. First, we compute the *LDL decomposition* $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^H$ using Alg. 1, then subsequently solve $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^H\boldsymbol{G} = \boldsymbol{H}$ for $\boldsymbol{G}$ using forward and back substitution steps. The matrix $\boldsymbol{L} = [l_{ij}] \in \mathbb{C}^{M\times M}$ is lower triangular with unity diagonal, i.e. $l_{ij} = 0\ \forall i < j$ and $l_{ii} = 1$, and $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_M)$ is a real-valued diagonal matrix with positive entries $d_j > 0$.

Algorithm 1: LDL Decomposition

```
   Require: $\boldsymbol{A} \in \mathbb{C}^{M\times M}$ is HPD
1  for $j = 1 \ldots M$ do
2      $d_j = \mathrm{Re}\{a_{jj}\}$
3      for $k = 1 \ldots j-1$ do
4          $d_j = d_j - \mathrm{Re}\{l_{jk}t^*_{jk}\}$          ▷ 2 Mult., 2 Add
5      end
6      $d_j^{-1} = 1/d_j$                                   ▷ 1 Reciprocal
7      for $i = j+1 \ldots M$ do
8          $t_{ij} = a_{ij}$
9          for $k = 1 \ldots j-1$ do
10             $t_{ij} = t_{ij} - l_{ik}t^*_{jk}$           ▷ 4 Mult., 4 Add
11         end
12         $l_{ij} = d_j^{-1}t_{ij}$                         ▷ 2 Mult.
13     end
14 end
```

The annotations in Alg. 1 denote the number of real-valued operations. In total, it requires $C_{mul} = \frac{2}{3}M^3 - \frac{2}{3}M$ multiplications, $C_{add} = \frac{2}{3}M^3 - M^2 + \frac{1}{3}M$ additions and M reciprocals for a square matrix of size $M \times M$. Note that for spatial-multiplexing MIMO systems, the number of antennas is typically not more than eight, i.e. it is reasonable to assume $M \leq 8$. More specifically, $M = 4$ is a widely used choice.

We compare the required computational complexity to compute $\boldsymbol{G}$ for $M = 4$ using Alg. 1 to the LU-based [6] and D&C-based [1]–[3] algorithms in Tbl. I. Relative to the LU-decomposition $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ with lower and upper triangular matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ respectively, we save about 28% of the multiplications and 36% of the additions for the matrix decomposition. The D&C method has a significantly higher complexity, however directly computes $\boldsymbol{C}_{\boldsymbol{y}}^{-1}$. An additional matrix multiplication (MM) is needed to obtain $\boldsymbol{G} = \boldsymbol{C}_{\boldsymbol{y}}^{-1}\boldsymbol{H}$, for which we assume to use Strassen's well-known D&C MM algorithm [8].

TABLE I. COMPUTATIONAL COMPLEXITY FOR $M = 4$

Algorithm	Operation	$\mathbb{R}\times\mathbb{R}$	$\mathbb{R}+\mathbb{R}$	$1/\mathbb{R}$
LDL	$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^H$	40	28	4
LU	$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$	56	44	4
D&C	$\boldsymbol{A} \mapsto \boldsymbol{A}^{-1}$	94	76	2
LDL + Solve	$\boldsymbol{C}_{\boldsymbol{y}} \mapsto \boldsymbol{G}$	264	124	4
LU + Solve	$\boldsymbol{C}_{\boldsymbol{y}} \mapsto \boldsymbol{G}$	280	140	4
D&C + MM	$\boldsymbol{C}_{\boldsymbol{y}} \mapsto \boldsymbol{G}$	294	284	2

Computing $\boldsymbol{G}$, denoted as operation $\boldsymbol{C}_{\boldsymbol{y}} \mapsto \boldsymbol{G}$ in Tbl. I, the LDL-based algorithm slightly outperforms the LU-based one in terms of operation count. The main difference lies in the dependencies between operations. This results in different operation schedules, which favors the LDL as we found out during the VLSI implementation. Both LDL-based and D&C-based algorithms exploit the HPD property of $\boldsymbol{A}$, by storing only half of the coefficients and reducing the number of real-valued operations. Nevertheless, the D&C-based algorithm used to compute $\boldsymbol{G}$ is more complex than the LDL and LU based algorithms, as indicated in Tbl. I. Clearly, the comparison is only fair for the computation of $\boldsymbol{G}$. For $M > 4$, we found that the advantage of the LDL decomposition based MMSE filter matrix computation strengthens. For example for $M = 8$, with 2256 multiplications (LDL) over 2480 (LU) and 2588 (D&C) the LDL based algorithm outperforms the other two.

IV. RESULTS

This section first describes our simulation setup that we used for the evaluation. Then we give an overview of our proposed architecture and its key characteristics. This is followed by an analysis of our VLSI implementation results. Subsequently we compare our results to related work.

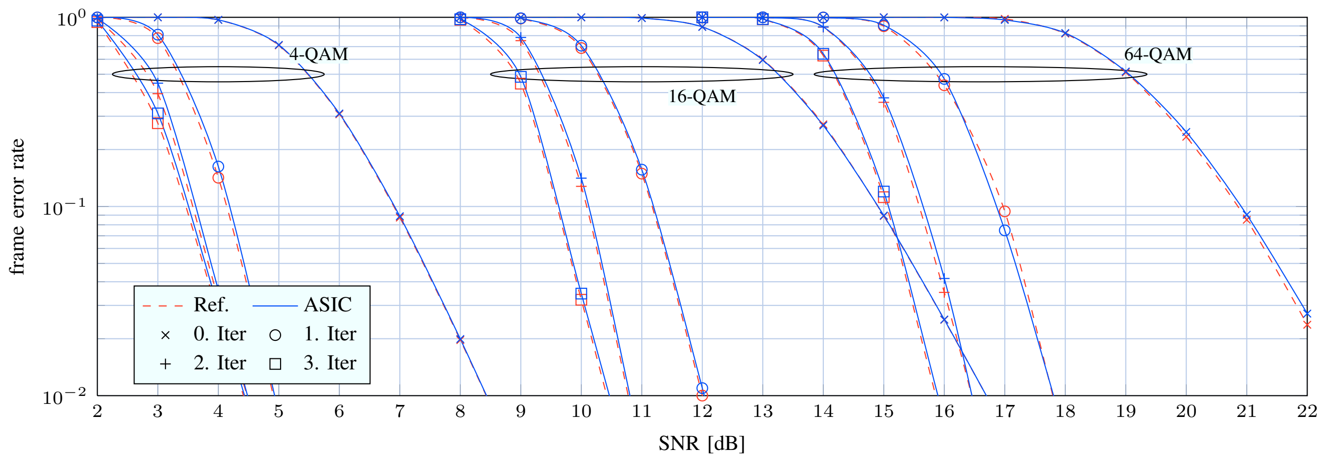

Fig. 2. Frame Error Rate over SNR. The reference is a max-log MMSE detector with posterior feedback using floating-point arithmetic. *ASIC* denotes our bit-accurate fixed-point model. *0. Iter* matches soft-output only detection, while for *1. Iter* the detector and decoder are both executed twice per symbol vector.

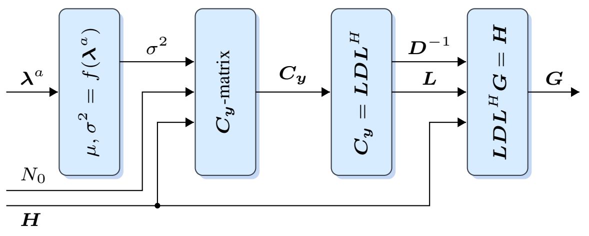

Fig. 3. Overview of the Architecture. The coarse-grained pipeline with four processing units is shifted every 18th clock cycle. The interface between the stages uses a simple handshake based protocol. Buffers for H, N_0 at the first three stages (not drawn) forward the values as required.

A. Conditions & Assumptions

A 40 MHz 802.11n-like scenario similar to [6] is considered assuming a 4×4 MIMO system with gray-mapped 4-/16-/64-QAM modulation, max-log demapping, a spatially uncorrelated Rayleigh channel and perfect channel knowledge. We use a rate-1/2 tail-biting convolutional code with polynomials $[133, 171]_8$ and a max-log BCJR decoder. A frame consists of 864 information bits. The average signal-to-noise ratio (SNR) per receive antenna is defined as SNR $= \mathbb{E}[\|Hx\|^2]/(N_r N_0)$. We determined the required word lengths to obtain an SNR loss of ≤ 0.1dB compared to the floating-point model at a frame error rate of 10%. Fig. 2 shows the frame error rate over SNR for both the reference and our ASIC model.

B. Architecture Overview

Our architecture computes G from H, N_0 and λ^a. As depicted in Fig. 3, it is organized into a pipeline of four stages that take 18 clock cycles per pipeline cycle, similar to [6]. For $N_t = 4$ antennas, 64-QAM ($K = 6$), $r = 5/6$ and a reasonable clock frequency of $f_{clk} = 540\,$Mhz, we achieve a data rate of $\Theta = \frac{rN_tK}{18} f_{clk} = 600\,$Mbit/s, which meets the maximum throughput requirement of the IEEE 802.11n standard. We use one clock cycle to exchange data between the stages. The inputs H, N_0 are forwarded as required. All data is kept in registers. We manually allocated the arithmetic units and devised operation schedules for each pipeline unit. The design shortens the critical path where possible, e.g. it uses pipelined arithmetic units. We compute the reciprocal with a lookup table followed by one Newton-Raphson iteration to improve the result's precision. The reciprocal unit input is shifted to the range $[1, 2)$. The shift is compensated at the output. Tbl. II summarizes the data path configuration.

TABLE II. DATAPATH CONFIGURATION

Processing Unit	Add.	Mult.	Shift	LUT	Recip.	Mem [bit]	Area [kGE]
$\mu, \sigma^2 = f(\lambda^a)$	9	2	–	2	–	263	6.4
C_y-matrix	8	12	–	–	–	718	22.4
Decomposition	12	3	2	1	1	787	20.3
Solver	16	16	–	–	–	1791	69.5
Forwarding H, N_0	–	–	–	–	–	720	4.0
Total	45	33	2	3	1	4279	122.6

The architecture has been synthesized with Synopsys Design Compiler G-2012.06 in topographical mode using a 90nm standard-performance CMOS library. The ASIC occupies a total area of about 122.6 kGE[2] and achieves the target clock frequency of 540 MHz. The per-unit area distribution is given in Tbl. II. We observe that the solver dominates the design in terms of area, and that the matrix decomposition only occupies less than one third of the solver area.

Fig. 4 shows the architecture of the processing unit (PU) responsible for the matrix decomposition. The three multipliers each have two pipeline stages, while the reciprocal unit has three stages. We compute complex-valued products with three real-valued multiplications and five additions. The interface between PUs uses a handshake-based protocol. All input values are transferred to local registers during the first clock cycle.

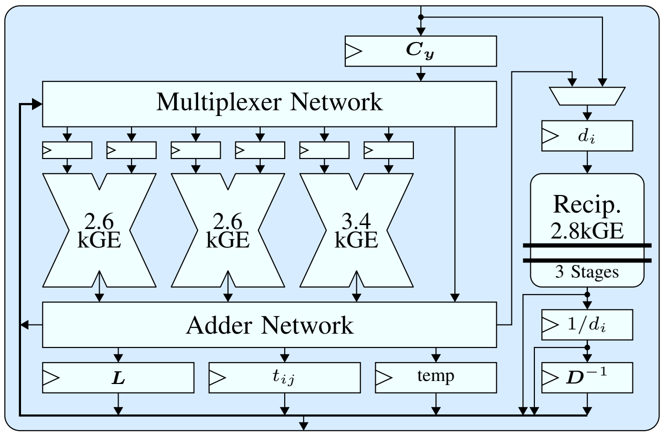

Fig. 4. Architecture of the processing unit performing the LDL decomposition. The local state machine and interface logic are excluded.

C. Comparison

Tbl. III lists circuits for the matrix inversion alone (rows 4-6), if available, and the computation of G (rows 7-9). Our efficiency metric is kGE per 10^6 Inv/s, and kGE per 10^6 G/s respectively. We treated the compound of decomposition and solver stages as matrix inversion circuits (this work and [6]). For the 9-step D&C circuit in [3], we considered the two out of nine steps that perform the inversion. We also include VLSI architectures for SO-only MMSE detection, since the proposed approach is also applicable in that context as stated in Sec. III.

Our architecture has the second best inversion efficiency of 3.0 kGE per 10^6 Inv/s, outperformed by [3] with 2.7 kGE per 10^6 Inv/s. However, while [3] computes C_y^{-1}, we compute $C_y^{-1}H$. We investigated this further and found that the simplification of our solver stage assuming the identity matrix as input could save slightly less than half of the operations. Reducing our solver area by only 13%, we would already achieve a similar efficiency.

Considering the computation of G, our ASIC is the — to the best of our knowledge — most efficient reported implementation with about 4.1 kGE per 10^6 G/s. It outperforms the second best architecture, the MMSE-PIC [6], by

[2]One gate equivalent (GE) is the area of one 2-input drive-1 NAND gate.

TABLE III. IMPLEMENTATION RESULTS AND COMPARISON WITH OTHER REPORTED CIRCUITS

Publication	This work	Studer *et al.* [6]	Eberli *et al.* [1]	Burg *et al.* [4]	Luethi *et al.* [5]	Luethi *et al.* [5]	Yoshizawa *et al.* [2]	Yoshizawa *et al.* [3]	Yoshizawa *et al.* [3]
Inversion algorithm	LDL	LU	D&C	Rank-1	GR-QR	MGS-QR	D&C	D&C 9-step	D&C 2-step
MMSE algorithm	SISO	SISO	SO	SO	SO	SO	SO	SO	SO
Matrix inversion area [kGE]	89.8	138	–	–	–	–	1320	303	–
Throughput [10^6 Inv/s]	30	31.5	–	–	–	–	174	112.5	–
Efficiency [kGE/(10^6 Inv/s)]	3.0	4.4	–	–	–	–	7.6	2.7	–
MMSE filter area [kGE]	122.6	223	383	89	48.7	61.8	2200	303	885
Throughput [10^6 $\boldsymbol{G}$/s]	30	31.5	6.0[a]	4.6[a]	4.2[a]	3.1[a]	174	25	70.2
Efficiency [kGE/(10^6 $\boldsymbol{G}$/s)]	4.1	7.1	63.8	19.3	11.6	19.9	12.6	12.1	12.6
CMOS technology [nm]	90	90	180	250	180	180	90	90	90
Clock frequency [MHz]	540	568	250	176	166	162	174	250	160

[a] Throughput scaled to 90nm CMOS technology assuming: $t_{pd} \sim 1/s$.

a factor of 1.7x. It is also clearly more efficient than D&C-based circuits [1]–[3]. This suggests that the solver-based approach is more suitable than inversion and subsequent matrix multiplication in this context.

It is noticeable that only the MMSE-PIC [6] and our ASIC are designed for SISO MMSE detection. Iterative MIMO decoding requires one inversion per symbol vector per iteration, because $\boldsymbol{G}$ depends on the decoder feedback. Contrarily, most SO-only publications assume that $\boldsymbol{G}$ can be precomputed before detection. They are thus designed for a reduced inversion throughput. However both SISO ASICs are superior to the other considered circuits in terms of efficiency, and additionally achieve a large SNR gain.

Interestingly, Yoshizawa's three ASICs [2], [3] have a constant efficiency of ~12 kGE per 10^6 $\boldsymbol{G}$/s. His iterative decomposition method to scale the architecture at design-time is able to maintain the efficiency. However, considering the matrix inversion alone, the 9-step ASIC [3] outperforms the completely pipelined ASIC [2] by a factor of ~2.8x. This suggests that evaluating the matrix inversion independently of the design context might lead to wrong conclusions.

Eberli's custom VLIW processor [1] is about one order of magnitude less efficient than our ASIC. The reason for this is most likely the offered flexibility of the processor. Depending on the context, this might be a major advantage, which is however not measurable with the selected metric.

V. CONCLUSION

We presented the most area-throughput efficient ASIC reported so far that performs matrix inversion based on the LDL decomposition to compute the MMSE filter matrix. The computational complexity of the LDL-based algorithm is lower than that of the LU-based and D&C-based algorithms. The literature comparison indicates that a full solver step to obtain $\boldsymbol{C}_{\boldsymbol{y}}^{-1}\boldsymbol{H}$ directly is beneficial in terms of area-throughput efficiency, and that the matrix inversion should not be evaluated independent of the filter matrix computation.

As future work, we are planning to extend the architecture to perform all parts of SISO MMSE MIMO detection.

ACKNOWLEDGMENT

This work has been supported by the Ultra High-Speed Mobile Information and Communication Research Centre, RWTH Aachen University.

REFERENCES

[1] S. Eberli, D. Cescato, and W. Fichtner, "Divide-and-conquer matrix inversion for linear MMSE detection in SDR MIMO receivers," in *NORCHIP, 2008.*, 2008, pp. 162–167.

[2] S. Yoshizawa, Y. Yamauchi, and Y. Miyanaga, "A complete pipelined MMSE detection architecture in a 4x4 MIMO-OFDM receiver," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008, pp. 2486–2489.

[3] S. Yoshizawa, H. Ikeuchi, and Y. Miyanaga, "VLSI implementation of a scalable pipeline MMSE MIMO detector for a 4x4 MIMO-OFDM receiver," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 94, no. 1, pp. 324–331, 2011.

[4] A. Burg, S. Haene, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006, pp. 4 pp.–.

[5] P. Luethi, C. Studer, S. Duetsch, E. Zgraggen, H. Kaeslin, N. Felber, and W. Fichtner, "Gram-schmidt-based QR decomposition for MIMO detection: VLSI implementation and comparison," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, 2008, pp. 830–833.

[6] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 7, pp. 1754–1765, 2011.

[7] M. Senst and G. Ascheid, "How the framework of expectation propagation yields an iterative IC-LMMSE MIMO receiver," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–6.

[8] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969. [Online]. Available: http://dx.doi.org/10.1007/BF02165411

[9] J. Hagenauer, "The turbo principle in mobile communications," in *Proc. International Symposium on Nonlinear Theory and its Applications, Xian, China*, 2002.