Divide-and-Conquer Matrix Inversion for Linear MMSE Detection in SDR MIMO Receivers

Stefan Eberli*, Davide Cescato†, and Wolfgang Fichtner*
*Integrated Systems Laboratory
ETH Zurich, Switzerland
e-mail: {eberli, fw}@iis.ee.ethz.ch
†Communication Technology Laboratory
ETH Zurich, Switzerland
e-mail: dcescato@nari.ee.ethz.ch

***Abstract*—Software-defined radio (SDR) platforms represent a promising approach to facing the demands of the fast evolving field of wireless communications. The flexibility of SDR solutions is typically obtained at the cost of limited processing power, which imposes the use of low-complexity algorithms. This task is particularly challenging within multiple-input multiple-output (MIMO) communication, which is inherently characterized by heavy signal processing load. In this paper, we present a recursive matrix inversion algorithm for Hermitian positive-definite (HPD) matrices. The algorithm exhibits low computational complexity and is thus particularly suited for the HPD matrix inversion required in MIMO minimum mean squared error receivers. The implementation on a design-time configurable VLIW processor, configured with appropriate processing units and fabricated in $0.18\,\mu\mathrm{m}$ 1P/6M CMOS technology, demonstrates that real-time operation in IEEE 802.11n-like MIMO-OFDM systems with up to 52 carriers and 3 antennas at the transmitter and at the receiver is possible.**

I. Introduction

Software-defined radio (SDR) platforms are emerging as economic and flexible solutions for the heterogeneous and rapidly changing scenario of wireless communications. By offering the possibility of post-fabrication software enhancements, SDR platforms are better suited to track the evolution of wireless communication standards than dedicated VLSI radio transceivers. The reverse of the medal is that the flexibility inherent in SDR solutions comes at the expense of lower efficiency in terms of area and power consumption. Thus, dedicated VLSI implementations currently represent the preferred choice for most applications within the field of wireless communications.

Multiple-input multiple-output (MIMO) communication systems, characterized by the use of multiple antennas at both ends of the wireless link, deliver significant performance gains with respect to single-antenna systems. For transmission over a wideband frequency-selective channel, MIMO is typically combined with orthogonal frequency-division multiplexing (OFDM), which divides the channel into a set of narrowband frequency-flat parallel subchannels.

The heavy signal processing load inherent to MIMO communication is amplified even further within MIMO-OFDM systems, since a significant part of the processing effort has to be carried out on every OFDM subchannel. For real-time operation on SDR platforms, which typically have limited processing resources, only low-complexity MIMO-OFDM transceivers with computationally efficient algorithms can be considered.

The first SDR implementations of MIMO receivers have recently appeared in the literature for 2×2 MIMO systems.[1] Specifically, the results reported lie in the domain of mobile hand-held devices and in the domain of wireless local area networks. Within the former, [1] and [2] focus their discussion on the challenges and the architectural choices encountered in the next generation of SDRs for mobile phones, including MIMO-OFDM related signal processing. Within the latter, [3] describes the implementation of a 2×2 MIMO-OFDM system for the upcoming IEEE 802.11n standard on the ADRES processor core. In this paper, we address matrix inversion as the hardest computational kernel performed in the class of linear minimum mean squared error (MMSE) MIMO detectors. Compared to detectors of other classes (e.g., sphere detectors [4]), linear MMSE detectors have low computational complexity, which potentially allows a real-time capable SDR implementation.

Related Work: Matrix inversion for a generic $M_R \times M_T$ linear MMSE MIMO-OFDM receiver, based on a series of rank-1 updates, is detailed in [5]. The proposed architecture efficiently reuses its resources and is composed of M_T parallel processing units. Another approach, described in [6], is particularly designed for 4×4 linear MMSE MIMO-OFDM receivers and implemented as unrolled, pipelined architecture. The matrix inversion is performed by *Banachiewicz inversion formula* [7]: The initial matrix is partitioned into four 2×2 matrices involved in the steps leading to the inversion of the initial 4×4 matrix. Overall, this process reduces the number of operations required for the inversion compared to direct matrix inversion. Both [5] and [6] target dedicated VLSI implementations. Finally, the implementation of matrix inversion, again based on Banachiewicz inversion formula, for a 4×4 linear MIMO-OFDM SDR receiver is presented in [8], then refined in [9] and [10].

Contributions: We derive a recursive matrix inversion algorithm for Hermitian positive-definite (HPD) matrices. The proposed method has low computational complexity and is thus well suited for computing the matrix inversion required by linear MMSE detectors. As a proof of concept, we consider a 2×2, a 3×3, and a 4×4 MIMO-OFDM receiver, for

[1] An $M_R \times M_T$ MIMO system has M_T transmit and M_R receive antennas.

which the linear MMSE estimator matrices are computed on a design-time configurable very long instruction word (VLIW) processor. We evaluate the resulting computational effort and the achieved bit error rate (BER) performance of corresponding MIMO receivers, and compare our results to those achieved by similar work in the literature.

Outline: The reminder of this paper is structured as follows. Section II introduces the mathematical model of the linear MMSE MIMO receiver system. In Section III, the recursive matrix inversion algorithm is derived, and formulated for 2×2, 3×3, and 4×4 HPD matrices. The design-time configurable VLIW processor used for the implementation of the MIMO-OFDM receivers is described in Section IV. Section V presents the results of the implementation, whereas the corresponding BER performance curves are shown in Section VI. Section VII concludes the paper.

Notation: $\mathbf{A}^H$ denotes the Hermitian transpose of a matrix $\mathbf{A}$. If $\mathbf{A}$ is quadratic, $\mathrm{adj}\,(\mathbf{A})$, and $\det(\mathbf{A})$ stand for its adjoint and its determinant, respectively. For any scalar $a \in \mathbb{C}$, we define $\mathrm{adj}\,(a) = 1$. The notation $\mathbf{z} \sim \mathcal{CN}(\mathbf{u}, \mathbf{R})$ indicates that the random vector $\mathbf{z}$ is characterized by a circularly-symmetric complex-valued Gaussian distribution with mean $\mathbf{u}$ and covariance matrix $\mathbf{R}$.

II. Linear MMSE MIMO Receiver System Model

We consider a MIMO system with M_T transmit and $M_R \geq M_T$ receive antennas. The baseband-equivalent input-output relation is given by

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

In (1), $\mathbf{s} \in \mathbb{C}^{M_T}$ represents the transmitted symbol vector, whose entries are picked from a set of QAM constellation points with mean zero and average energy $1/M_T$. The complex-valued vector $\mathbf{y} \in \mathbb{C}^{M_R}$ denotes the signal observed at the receiver, whereas the additive noise experienced at the receiver is modeled by the random vector $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2\mathbf{I}_{M_R})$. The complex-valued entries of the matrix $\mathbf{H} \in \mathbb{C}^{M_R \times M_T}$ represent the channel gains between the transmit and the receive antennas.

In packet-based transmission schemes, the receiver estimates the channel matrix $\mathbf{H}$ from a set of training symbols at the beginning of every packet. The resulting channel estimate matrix $\hat{\mathbf{H}}$ is then used to compute the linear MMSE estimator matrix

$$\mathbf{G} = (\hat{\mathbf{H}}^H\hat{\mathbf{H}} + M_T\sigma^2\mathbf{I}_{M_T})^{-1}\hat{\mathbf{H}}^H. \tag{2}$$

The MIMO MMSE receiver computes $\hat{\mathbf{s}} = \mathbf{G}\mathbf{y}$. For hard detection, the entries of $\hat{\mathbf{s}}$ are mapped individually to the nearest constellation points and the result is declared as the transmitted signal vector. For soft detection, $\hat{\mathbf{s}}$ is further processed to obtain appropriate soft information.

III. Divide-and-Conquer Matrix Inversion

A. Algorithm

The computation of $\mathbf{G}$ as in (2) requires the inversion of the $M_T \times M_T$ matrix $\mathbf{J} \triangleq \hat{\mathbf{H}}^H\hat{\mathbf{H}} + M_T\sigma^2\mathbf{I}_{M_T}$, which, by construction, is HPD. A matrix $\mathbf{F}$ is Hermitian if it satisfies $\mathbf{F}^H = \mathbf{F}$ and positive-definite if $\mathbf{x}^H\mathbf{F}\mathbf{x} > 0$ holds for all $\mathbf{x} \in \mathbb{C}^M$ [11]. In the following, we discuss the problem of matrix inversion for the entire class of HPD matrices by considering a generic $M \times M$ HPD matrix $\mathbf{F}$.

The *direct inversion* formula

$$\mathbf{F}^{-1} = \frac{\mathrm{adj}\,(\mathbf{F})}{\det(\mathbf{F})} \tag{3}$$

is trivial for $M = 1$, since in this case $\mathbf{F}$ is a positive scalar. For $M = 2$, (3) corresponds to

$$\begin{bmatrix} a & b \\ b^* & c \end{bmatrix}^{-1} = \frac{1}{ac - bb^*}\begin{bmatrix} c & -b \\ -b^* & a \end{bmatrix}. \tag{4}$$

In the case $M > 2$, however, the high computational complexity and the large dynamic range render the use of (3) impractical. An alternative approach is obtained by partitioning $\mathbf{F}$ as

$$\mathbf{F} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^H & \mathbf{C} \end{bmatrix} \tag{5}$$

where $\mathbf{A} \in \mathbb{C}^{p \times p}$ and $\mathbf{C} \in \mathbb{C}^{(M-p)\times(M-p)}$, with $1 \leq p < M$. Then, the inverse of $\mathbf{F}$ can be written, using the Banachiewicz formula for the inverse of a partitioned matrix [7], as

$$\mathbf{F}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1}\mathbf{B}^H\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{-1} \\ -\mathbf{S}^{-1}\mathbf{B}^H\mathbf{A}^{-1} & \mathbf{S}^{-1} \end{bmatrix} \tag{6}$$

with $\mathbf{S} \triangleq \mathbf{C} - \mathbf{B}^H\mathbf{A}^{-1}\mathbf{B}$ being the Schur complement of $\mathbf{A}$ in $\mathbf{F}$. In order for (6) to hold, $\mathbf{A}$ and $\mathbf{S}$ must be nonsingular. This is directly verified by noting that since $\mathbf{A}$ is a principal submatrix of the HPD matrix $\mathbf{F}$, both $\mathbf{A}$ and its Schur complement $\mathbf{S}$ are HPD [11, chap. 7], and hence nonsingular. As a result, the task of inverting the $M \times M$ matrix $\mathbf{F}$ can be replaced by the simpler tasks of inverting the $p \times p$ matrix $\mathbf{A}$ and the $(M-p) \times (M-p)$ matrix $\mathbf{S}$, followed by combining the resulting $\mathbf{A}^{-1}$ and $\mathbf{S}^{-1}$ according to (6).

The combination of all concepts illustrated above allows to formulate the following algorithm for HPD matrix inversion:

Algorithm 1 HPD matrix inversion

Require: $\mathbf{F}$ is an $M \times M$ HPD matrix
  if $M = 1$ or $M = 2$ **then**
    compute $\mathbf{F}^{-1}$ by direct inversion as in (3)
  else
    pick a suitable p satisfying $1 \leq p < M$
    partition $\mathbf{F}$ as in (5)
    compute $\mathbf{A}^{-1}$
    compute $\mathbf{S} = \mathbf{C} - \mathbf{B}^H\mathbf{A}^{-1}\mathbf{B}$
    compute $\mathbf{S}^{-1}$
    construct $\mathbf{F}^{-1}$ as in (6)
  end if

In order to complete the definition of Algorithm 1, we need to specify how $\mathbf{A}^{-1}$ and $\mathbf{S}^{-1}$ are computed and how p is picked. We focus on the former issue first. Since, as discussed above, $\mathbf{A}$ and $\mathbf{S}$ are HPD matrices, Algorithm 1 itself can be applied to $\mathbf{A}$ and $\mathbf{S}$ for computing $\mathbf{A}^{-1}$ and $\mathbf{S}^{-1}$, respectively.

Thus, we obtain a recursive method for the computation of the inverse of HPD matrices that we will, henceforth, call *divide-and-conquer* (D&C) matrix inversion. Partitioning as in (5) represents the divide step, whereas the conquer step results from (6). We note that the recursion ends when the matrix submitted to Algorithm 1 has dimension 2×2 or is scalar.

Picking $p = 1$ or $p = 2$ implies that $\mathbf{A}^{-1}$ will be computed by direct inversion, whereas picking $p = M-1$ or $p = M-2$ implies that $\mathbf{S}^{-1}$ will be computed by direct inversion. For even M, another strategy is given by picking $p = M/2$, from which follows that $\mathbf{A}$ and $\mathbf{S}$ have the same size.

B. Complexity Analysis

In order to assess the complexity of D&C matrix inversion, we count the multiplications and divisions required by Algorithm 1 (including the computation of $\mathbf{A}^{-1}$ and $\mathbf{S}^{-1}$). Thereby, we do not distinguish between real-valued and complex-valued multiplications, and by division we denote the real-valued operation $1/x$. Further, we do not consider additions, subtractions, and sign changes, since, depending on the implementation, these operations may be integrated within the multiply operations. Finally, we assume that the Hermitian symmetry in the matrices $\mathbf{F}$, $\mathbf{A}$, $\mathbf{S}$, and their inverses is exploited to minimize the number of operations.

We initially set the restriction that all 2×2 matrices are inverted by partitioning as in (6), and not by the direct inversion foreseen by Algorithm 1. Under this restriction, it can be verified by induction that D&C inversion of a $M \times M$ HPD matrix requires $\tilde{C}_{\text{D\&C}} = (1/2)M^3 + (1/2)M^2 - M$ multiplications and M divisions. This result is independent from the strategy used to pick p.

We now remove the restriction and assume that the choices of p across all levels of recursion lead to a total of K 2×2 direct inversions (with $0 \leq K \leq M/2$). Since a 2×2 direct inversion costs one multiplication more and one division less than 2×2 partitioned inversion as in (6), the resulting complexity of $M \times M$ D&C matrix inversion amounts to

$$\begin{aligned} C_{\text{D\&C}} &= \tilde{C}_{\text{D\&C}} + K \\ &= \frac{1}{2}M^3 + \frac{1}{2}M^2 - M + K \end{aligned}$$

multiplications and $M - K$ divisions.

Finally, we discuss the cost of computing the MMSE estimator matrix $\mathbf{G}$ from the estimated channel matrix $\hat{\mathbf{H}}$ as in (2), if D&C matrix inversion is used. Computing $\mathbf{J}$ from $\hat{\mathbf{H}}$ requires $(M_T^2 M_R + M_T M_R)/2$ multiplications, whereas computing $\mathbf{G}$ from $\mathbf{J}^{-1}$ requires $M_T^2 M_R$ multiplications. Thus, the computation of $\mathbf{G}$ from $\hat{\mathbf{H}}$ with D&C inversion of the $M_T \times M_T$ matrix $\mathbf{J}$ requires in total

$$\begin{aligned} C_{\mathbf{G},\text{D\&C}} = &\frac{3}{2}M_T^2 M_R + \frac{1}{2}M_T M_R \\ &+ \frac{1}{2}M_T^3 + \frac{1}{2}M_T^2 - M_T + K \end{aligned}$$

multiplications and $M_T - K$ divisions (with $0 \leq K \leq M_T/2$). In comparison, the use of the rank-1 update matrix inversion described in [5] requires

$$C_{\mathbf{G},\text{rank-1}} = \frac{5}{2}M_T^2 M_R + \frac{5}{2}M_T M_R - M_T^2 + M_T$$

multiplications and M_R divisions. We note that $M_R \geq M_T$ implies that $C_{\mathbf{G},\text{D\&C}} < C_{\mathbf{G},\text{rank-1}}$ holds for $0 \leq K \leq M_T/2$. Hence, the computation of $\mathbf{G}$ with D&C matrix inversion requires fewer multiplications and no more divisions than with rank-1 update matrix inversion.

C. Algorithms for 2×2, 3×3, and 4×4 Matrix Inversion

In the following, D&C matrix inversion is formulated for the special cases of 2×2, 3×3 and 4×4 matrix inversion, using the notation of Section III-A. Again, the Hermitian symmetries are exploited to minimize the number of required operations. The symbols $\mathbf{R}_i$, $\mathbf{r}_i$, and r_i employed in the algorithms indicate temporary storage locations of appropriate dimensions.

For a 2×2 HPD matrix $\mathbf{F}$, D&C inversion coincides with direct inversion and is performed as follows:

Algorithm 2 Implementation of 2×2 direct inversion

$r_1 \leftarrow ac$	$\mathbb{C}$MUL
$r_2 \leftarrow r_1 - bb^*$	$\mathbb{C}$MAC
$r_3 \leftarrow r_2^{-1}$	DIV
$x \leftarrow r_3 c$	$\mathbb{C}$MUL
$y \leftarrow -r_3 b$	$\mathbb{C}$MUL
$z \leftarrow -r_3 a$	$\mathbb{C}$MUL
$\mathbf{F}^{-1} = \begin{bmatrix} x & y \\ y^* & z \end{bmatrix}$	-

Algorithm 2 requires a total of 6 operations, split into complex-valued multiplication ($\mathbb{C}$MUL), complex-valued multiply-and-accumulate ($\mathbb{C}$MAC), and division (DIV).

In the 3×3 case, we employ D&C inversion with $p = 2$. This leads to a partitioning of $\mathbf{F}$ where $\mathbf{A}$ has dimension 2×2, $\mathbf{B}$ is 2×1, and $\mathbf{C}$ scalar. In the 4×4 case, we employ D&C inversion and set $p = 2$ again, partitioning $\mathbf{F}$ into four 2×2 submatrices. This approach coincides with the solution proposed in [10]. In both cases, $\mathbf{F}^{-1}$ is computed as follows:

Algorithm 3 Implementation of 3×3 and 4×4 D&C inversion with corresponding number of operations

		3×3	4×4
$\mathbf{R}_1 \leftarrow \mathbf{A}^{-1}$	Direct inversion	6	6
$\mathbf{R}_2 \leftarrow \mathbf{B}^H \mathbf{R}_1$	$\mathbb{C}$MAC	4	8
$\mathbf{R}_3 \leftarrow \mathbf{R}_2 \mathbf{B}$	$\mathbb{C}$MAC	2	8
$\mathbf{S} \leftarrow \mathbf{C} - \mathbf{R}_3$	$\mathbb{C}$SUB	1	4
$\mathbf{Z} \leftarrow \mathbf{S}^{-1}$	Direct inversion	1	6
$\mathbf{Y} \leftarrow -\mathbf{R}_2^H \mathbf{Z}$	$\mathbb{C}$MAC	2	8
$\mathbf{R}_4 \leftarrow \mathbf{Y}\mathbf{R}_2$	$\mathbb{C}$MAC	4	8
$\mathbf{X} \leftarrow \mathbf{R}_1 - \mathbf{R}_4$	$\mathbb{C}$SUB	4	4
$\mathbf{F}^{-1} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{Y}^H & \mathbf{Z} \end{bmatrix}$	-	-	-
	Total	24	52

Algorithm 3 requires the arithmetic operators already necessary for Algorithm 2, supplemented by complex-valued subtraction (ℂSUB). A total of 24 operations in the 3×3 case and 52 operations in the 4×4 case is needed to compute $\mathbf{F}^{-1}$.

IV. Processor Architecture

The VLIW-like processor considered in this paper for the implementation of matrix inversion is part of the framework for design-time configurable adaptive stream processing engines (ASPEs) described in [12]. The framework permits the designer to configure the ASPE's datapath with functional and storage units tailored to the application domain targeted. The software is written in a dedicated assembly language. An example of ASPE configuration for SDR is reported in [13], where the baseband processing of a single-antenna OFDM receiver is implemented.

Figure 1 depicts the block diagram of the ASPE configuration selected for this paper, which, in the following, is named *ASPE B*. The controlpath comprises the sequencer (SEQ) which is responsible of controlling the program flow. Dictionary-based code compression is supported thanks to an index and a dictionary memory. The VLIWs are split into 16 bit wide control words and fed to the corresponding datapath units. The datapath is especially designed for 16 bit complex-valued arithmetics. Thereby, memory access bottlenecks are easily avoided by storing the real and imaginary parts in the lower and upper half of the same 32 bit data word, respectively.

In agreement with our analysis in Section III-C, the datapath has been equipped with two complex-valued multiply-and-accumulate units (ℂMAC0-1), one real-valued divider unit (DIV), and one complex-valued arithmetic logic unit (ℂALU). The storage requirements are covered by a small register-file (REG) of eight registers, four 256-word storage units (RAM0-3), an input buffer (I-BUF), and an output buffer (O-BUF).

For an effective implementation, the number of pipeline stages inside each unit was chosen to level the length of the critical timing paths, across all units. In addition, the data-level parallelism inherent to most signal processing algorithms is exploited by designing all units, except the I-BUF and the O-BUF, as two-way single-instruction multiple-data (SIMD) units. By doing so, each unit performs the same operation on two different sets of data, doubling the available processing performance. The number of pipeline stages and the two-way SIMD nature of the units are visible in Figure 1.

The data network (D-Net) connects functional and storage units. It is controlled by the sequencer and can change its interconnect configuration at run-time, at each clock cycle. This reconfigurability allows the concurrent operation of up to all datapath units, thus maximizing the available processing power. The D-Net also permits to configure the datapath as a processing chain containing both storage and functional units. In this way, the demand for an expensive, high-bandwidth register file for temporary data storage is avoided – with a resulting advantage over conventional VLIW processors that require such register files to operate.

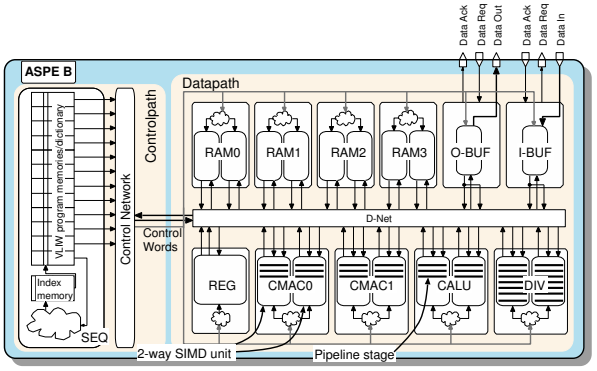

Fig. 1. Block diagram of ASPE B, with datapath configuration tailored for matrix inversion.

V. Implementation Results

In the following, the computation on ASPE B of the linear MMSE estimator matrix $\mathbf{G}$ is discussed for an IEEE 802.11n-like MIMO-OFDM system with 52 subcarriers. The latency imposed by the computation of all 52 $\mathbf{G}$ matrices from corresponding $\hat{\mathbf{H}}$ matrices, which we denote as *preprocessing latency*, infers a delay in the detection of the payload data. For real-time operation, the receiver must process the received payload data fast enough to compensate the inferred delay before the end of the data packet. The duration of one OFDM symbol is 4 µs. Assuming that the decoding of one OFDM symbol requires 2.5 µs and that the shortest packets contain at least 10 OFDM symbols, we obtain the maximum preprocessing latency allowed for real-time operation as $T_{pp} = 10(4\,\mu\text{s} - 2.5\,\mu\text{s}) = 15\,\mu\text{s}$.

Three different antenna configurations (2×2, 3×3, and 4×4) are considered. The matrix inversions required to compute $\mathbf{G}$ in the three distinct cases are performed according to Algorithm 2 and Algorithm 3.

A. Silicon Realization

ASPE B, as detailed in Figure 1 and with a datapath word-width of 16 bit, was fabricated on a $0.18\,\mu\text{m}$ 1P/6M CMOS technology. The post-route clock frequency reported by the back-end design tool is 250 MHz, whereas the total silicon area results in 3.7 mm^2. Table I reveals the area breakdown of the functional units of ASPE B. Note that the areas for all units refer to complete two-way SIMD units. Thus, for instance, the DIV unit area is the sum of the areas of the two real-valued dividers operating in SIMD manner.

B. Operation in a 2×2 MIMO-OFDM System

In the 2×2 case, ASPE B is programmed to compute blocks of four $\mathbf{G}$ matrices at a time. In this manner, the two-way SIMD units and the integrated pipeline stages are fully exploited, maximizing the matrix inversion throughput. The computation of four 2×2 $\mathbf{G}$ matrices from corresponding 2×2 $\hat{\mathbf{H}}$ matrices requires 44 clock cycles, 20 of which are dedicated to 2×2 direct inversion (as in Algorithm 2) of the

TABLE I
POST LAYOUT SILICON-AREA BREAKDOWN OF ASPE B IN 0.18 µm 1P/6M CMOS TECHNOLOGY

Unit	Area [mm^2]	Complexity[a] [kGE]
DIV	0.40	41
$\mathbb{C}$MAC0	0.37	38
$\mathbb{C}$MAC1	0.38	39
$\mathbb{C}$ALU	0.10	11
RAM[0-3]	0.22 each	23 each
I-BUF	0.10	11
O-BUF	0.11	11
SEQ	0.78	81
other	*0.58*	*59*
ASPE B	3.7	383

[a]The area of one gate equivalent (GE) corresponds to the silicon area occupied by one low-drive 2 input NAND, which amounts to 9.67 µm^2 on the considered technology.

four $\mathbf{J}$ matrices. The resulting implementation allows real-time operation for the considered IEEE 802.11n-like MIMO-OFDM system, since the preprocessing latency for the computation of all 52 $\mathbf{G}$ matrices amounts to $2.3\,\mu\text{s} < T_{pp}$.

C. Operation in a 3×3 MIMO-OFDM System

The two-way SIMD units of ASPE B are exploited again to compute two 3×3 $\mathbf{G}$ matrices in parallel. Following the steps of Algorithm 3 results in an implementation that requires 118 clock cycles to compute two $\mathbf{G}$ matrices from corresponding $\hat{\mathbf{H}}$ matrices. For the considered IEEE 802.11n-like system, this accounts for a preprocessing latency of $13\,\mu\text{s} < T_{pp}$, allowing real-time operation.

D. Operation in a 4×4 MIMO-OFDM System

If D&C matrix inversion as in Algorithm 3 is used, 166 clock cycles are needed to compute two 4×4 $\mathbf{G}$ matrices from corresponding 4×4 $\hat{\mathbf{H}}$ matrices by exploiting the two-way SIMD units of ASPE B. As a comparison, if the rank-1 update algorithm is used for matrix inversion on ASPE B, the same task requires 400 clock cycles. The preprocessing latency for the considered IEEE 802.11n-like system using D&C matrix inversion is approximately $17.3\,\mu\text{s} > T_{pp}$. Thus, our real-time constraint is not fulfilled.

E. Comparison with Similar Work

Table II presents the number of clock cycles and the corresponding processing time necessary for the computation of one $\mathbf{G}$ matrix, for the methods presented in this paper and for methods proposed in related work.

When considering 4×4 matrices, the most similar work is represented by the 4×4 linear MMSE estimator matrix computation for MIMO SDRs presented in [10]. Although the matrix inversion in [10] coincides with the 4×4 D&C matrix inversion presented in Algorithm 3, there is a substantial difference that deserves special attention. Reference [10] is restricted to the special case of Alamouti space-time block coding (STBC) which leads to Hermitian matrices with null-entries, whose inversion is significantly simpler than the matrix inversion presented in this paper, which applies to the entire class of HPD matrices. Nevertheless, D&C inversion is more efficient in terms of required clock cycles – 83 versus 202 in [10]. This result is achieved by exploiting the Hermitian symmetry of $\mathbf{J}$, its submatrices, and the corresponding inverses, and by leveraging on the SIMD architecture of ASPE B.

The work described in [7] implements a dedicated, pipelined, and completely unrolled architecture for the computation of the 4×4 linear MMSE matrix following the D&C matrix inversion as in Algorithm 3. Clearly, this implementation is the most performing one in terms of processed 4×4 $\mathbf{G}$ matrices per unit of time. However, the achieved performance comes at the cost of a very large area, between 6.23 and 8.81 mm^2 depending on the selected word-width, in a 90 nm CMOS process.

TABLE II
PROCESSING LATENCY FOR THE COMPUTATION OF ONE $\mathbf{G}$ MATRIX

Matrix size, Method	Cycles	Time[a] [ns]
2×2 Direct (this work)	$44/4 = 11$	44 @ 250 MHz
3×3 D&C (this work)	$118/2 = 59$	236 @ 250 MHz
4×4 D&C (this work)	$166/2 = 83$	332 @ 250 MHz
4×4 Rank-1 (this work)	$400/2 = 200$	800 @ 250 MHz
4×4 D&C [10]	202	505 @ 400 MHz
4×4 D&C [7]	1	6 @ 160 MHz
4×4 Rank-1 [5]	102	579 @ 176 MHz

[a]Processing time computed according to the clock frequency of the corresponding work.

VI. BER PERFORMANCE

In this section, we present the BER performance of MIMO MMSE soft-decision receivers (cascaded with a soft-input Viterbi decoder [14]), when the matrix $\mathbf{G}$ is computed according to the assembler code implementations of D&C matrix inversion presented in Sections V-B, V-C, and V-D.

The BER performance is determined by Monte Carlo simulations. In every simulation cycle, randomly generated bits are sent through a rate 1/2 convolutional encoder (with generator polynomials $[133_o\ 171_o]$ and constraint length 7) and successively Gray-mapped onto points of a 64-QAM constellation. The resulting complex-valued symbols are stacked to build the vector $\mathbf{s}$ and are transmitted over the MIMO channel according to (1). The entries of the channel matrix $\mathbf{H}$ are independent and identically distributed according to $\mathcal{CN}(0, 1)$. The signal-to-noise ratio (SNR) at the receiver is given by $1/\sigma^2$. We assume that the receiver perfectly estimates the channel (i.e., that $\hat{\mathbf{H}} = \mathbf{H}$), as well as the noise variance σ^2.

Figure 2 presents the resulting BER performance for 2×2, 3×3, and 4×4 MIMO systems. The curves corresponding to ideal floating-point matrix inversion are provided as a reference, whereas the curve corresponding to the ASPE B implementation of the rank-1 update matrix inversion serves as a comparison in the 64-QAM 4×4 case. The dashed threshold at BER $= 2 \cdot 10^{-7}$, visible in all plots, indicates the maximum BER tolerated in order to achieve a packet error rate (PER) of 10% or less, with packets of length $L = 2^{19}$ bits. The threshold has been derived by computing BER $= 1 - (1 - \text{PER})^{1/L}$, which holds under the pessimistic assumption that the bit errors are uniformly distributed over the received data.

Figure 2 shows that in the 2×2 and the 3×3 case, the performance of the linear detector implemented on ASPE B exhibits a small implementation loss compared to the ideal detector using floating-point arithmetic, until reaching the $2 \cdot 10^{-7}$ BER threshold. Thus, a PER of 10% can be obtained without major SNR penalties.

The bottom left plot of Figure 2 reveals that, in the 4×4 case with 64-QAM, the error floor of the rank-1 update based MMSE receiver is significantly higher than the error floor of the D&C based MMSE receiver. We also observe that, both D&C and rank-1 update matrix inversion, lead to a sensible implementation loss. At the 10% PER threshold, the implementation loss amounts to 3 dB and 5 dB for the D&C and the rank-1 update matrix inversion, respectively. This suggests that a wordwidth of 16 bits is not enough for an effective 4×4 MIMO detection with 64-QAM. However, for 16-QAM 4×4 MIMO transmission, a significantly smaller implementation loss is obtained (as the bottom right plot in Figure 2 shows).

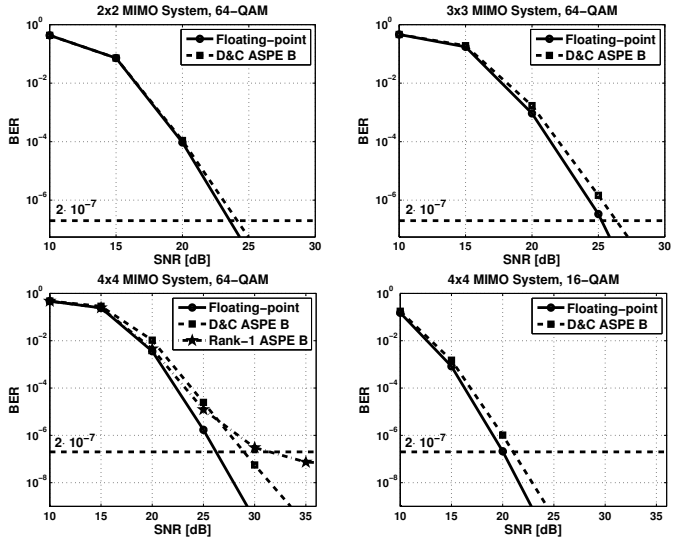

Fig. 2. BER performance curves for different matrix inversion implementations: reference floating-point, D&C on ASPE B, and rank-1 update on ASPE B.

VII. Conclusions

In this paper, we presented D&C matrix inversion as a recursive, low-complexity inversion algorithm for HPD matrices. We employed D&C matrix inversion for the computation of the linear MMSE estimator matrix for 2×2, 3×3, and 4×4 MIMO-OFDM systems, and presented corresponding implementation results on a design-time configurable VLIW processor.

The proposed D&C matrix inversion method is well suited for the implementation in SDR platforms. For the CMOS technology considered in this work, the 2×2 and 3×3 solutions for IEEE 802.11n-like MIMO-OFDM systems are real-time capable and deliver enough numerical precision to be of practical interest. Nevertheless, for larger matrices, inversion remains a demanding task, as testified by the 4×4 implementation, and requires dedicated VLSI solutions.

VIII. Acknowledgements

This research was supported by the Swiss Innovation Promotion Agency (CTI), project nr. KTI-8537. Many thanks go to Matthias Braendli, Benjamin Dietrich, and Lukas Haas, who were substantially involved in the design of ASPE B.

References

[1] M. Woh, S. Seo, H. Lee, Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer Berlin / Heidelberg, 2007, vol. 4599/2007, ch. The Next Generation Challenge for Software Defined Radio, pp. 343–354.

[2] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *Computer*, vol. 40, no. 10, pp. 62–69, Oct. 2007.

[3] B. Bougard, B. De Sutter, S. Rabou, D. Novo, O. Allam, S. Dupont, and L. Van der Perre, "A coarse-grained array based baseband processor for 100mbps+ software defined radio," in *Design, Automation and Test in Europe, 2008. DATE '08*, Munich, Germany, Mar. 2008, pp. 716–721.

[4] J. Antikainen, P. Salmela, O. Silven, M. Juntti, J. Takala, and M. Myllyla, "Application-specific instruction set processor implementation of list sphere detector," in *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, Pacific Grove, CA, Nov. 4–7, 2007, pp. 943–947.

[5] 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*, May 2006.

[6] 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*, Seattle, WA, USA, May 18–21, 2008, pp. 2486–2489.

[7] F. Zhang, Ed., *The Schur Complement and Its Applications*, ser. Numerical Methods and Algorithms. Springer US, March 30 2006, vol. Volume 4.

[8] J. Eilert, D. Wu, and D. Liu, "Efficient complex matrix inversion for MIMO software defined radio," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, New Orleans, LA, May 27–30, 2007, pp. 2610–2613.

[9] D. Wu, J. Eilert, D. Liu, D. Wang, N. Al-Dhahir, and H. Minn, "Fast complex valued matrix inversion for multi-user STBC-MIMO decoding," in *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, Porto Alegre, Mar. 9–11, 2007, pp. 325–330.

[10] J. Eilert, D. Wu, and D. Liu, "Implementation of a programmable linear MMSE detector for MIMO-OFDM," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, Las Vegas, NV, Mar. 31–Apr. 2008, 2008, pp. 5396–5399.

[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge, U.K., 1985.

[12] T. Boesch, "Adaptive stream processor for network multimedia consumer electronic devices," Ph.D. dissertation, ETH Zurich, 2004.

[13] S. Eberli, A. Burg, T. Boesch, and W. Fichtner, "An IEEE 802.11a baseband receiver implementation on an application specific processor," in *Circuits and Systems, 2007. MWSCAS 2007. 50th Midwest Symposium on*, Montreal, Que., Aug. 5–8, 2007, pp. 1324–1327.

[14] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.