Hardware-Efficient Steering Matrix Computation Architecture for MIMO Communication Systems

C. Senning, C. Studer, P. Luethi, and W. Fichtner
Integrated Systems Laboratory, ETH Zurich, Switzerland
email: {csenning, studer, luethi, fw}@iis.ee.ethz.ch

Abstract— **Beamforming (BF) improves the error rate performance of multiple-input multiple-output (MIMO) wireless communication systems by spatial separation of the transmitted data streams. Spatial separation is achieved by multiplication of the transmit vector by a steering matrix, which is obtained through the singular value decomposition (SVD) of the channel matrix. In this paper, we describe a hardware-efficient VLSI architecture for steering matrix computation using a hardware-optimized SVD algorithm. Our architecture contains a high-speed Givens rotation unit which achieves high processing throughput at low area. The resulting VLSI implementation requires 3.3 µs per steering matrix computation at an expense of 41.3 kGEs and shows a 3.5-fold hardware-efficiency gain compared to a reference SVD implementation.**

I. Introduction

Multiple-input multiple-output (MIMO) systems employ multiple antennas at both sides of the wireless link and are able to significantly improve the throughput of wireless communication systems by transmitting multiple data streams concurrently and in the same frequency band [1]. Orthogonal frequency division multiplexing (OFDM) modulation drastically simplifies equalization and thus, MIMO in combination with OFDM constitutes the basis for many upcoming wireless standards, such as IEEE 802.11n [2]. Beamforming (BF) is considered to be a key technology to further improve the performance of coded MIMO systems by spatially separation of the transmitted data streams [3]. This separation is achieved by multiplication of the transmit vectors with a *steering matrix*, which is obtained by computing the singular value decomposition (SVD) [4] of the MIMO channel matrix.

MIMO-OFDM systems that employ BF need to compute multiple SVDs concurrently, where the number of parallel tasks corresponds to the number of OFDM tones. Fast but large SVD computation architectures, e.g., systolic arrays [5], are often difficult to match to an arbitrary number of parallel problems, since one instance might be insufficient in terms of throughput and two instances might exceed the given requirements. Thus, it is often beneficial to design one small, but slower architecture and to employ multiple instantiations thereof in order to achieve a better matching to the throughput requirements of modern wireless systems. Hence, replication of low-area instances is a promising strategy to minimize the area overhead. Further reduction of the total circuit area can be achieved by improving the hardware-efficiency[1] while attaining the system's target throughput. Hardware-efficiency optimization on micro-architectural level of the low-area architecture increases the computation speed of each individual instance without a significant area expenditure. Hence, the total number of required instances can be reduced, which ultimately lowers silicon costs. A practical implementation of steering matrix computation for MIMO-OFDM systems employing a large number of tones – as it is the case for IEEE 802.11n-based systems – asks for multiple instantiations of a low-area and hardware-efficient VLSI architecture.

Contributions: Based on the more general matrix decomposition architecture described in [6], we present a dedicated steering matrix computation unit optimized for MIMO-OFDM systems. To this end, we modify the GK-SVD algorithm [7] to efficiently compute steering matrices and use a high-speed Givens rotation architecture to reduce the computation time per instance. We perform finite-precision optimizations of the architecture and provide a detailed hardware-efficiency optimization, in order to meet the throughput requirements of MIMO-OFDM systems and to reduce the total circuit area.

Outline: The remainder of this paper is organized as follows. Sec. II introduces the system model and describes the steering matrix computation algorithm. The corresponding VLSI architecture is introduced in Sec. III and the optimization in terms of hardware-efficiency is outlined in Sec. IV. The final implementation results are provided in Sec. V and we conclude in Sec. VI.

II. Steering Matrix Computation

Consider a MIMO system with M_T transmit and M_R receive antennas. The input-output relation of the baseband-equivalent wireless channel corresponds to $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$, where $\mathbf{s}$ is the M_T-dimensional transmit vector, $\mathbf{n}$ the M_R-dimensional Gaussian noise vector, and $\mathbf{H}$ the complex-valued $M_R \times M_T$-dimensional channel matrix. One method to perform BF in coded MIMO systems is to transmit $\tilde{\mathbf{s}} = \mathbf{V}\mathbf{s}$, where $\mathbf{V}$ is the steering matrix [3]. This matrix is obtained by computing the SVD of the channel matrix $\mathbf{H}$.

A. Singular Value Decomposition

The SVD of the complex-valued channel matrix $\mathbf{H}$ is defined[2] as $\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H$ [4], where $\mathbf{U}$ and $\mathbf{V}$ are complex-valued unitary matrices of dimension $M_R \times M_R$ and $M_T \times M_T$, respectively. The $M_R \times M_T$-dimensional matrix $\boldsymbol{\Sigma}$ contains $r = \min\{M_R, M_T\}$ real-valued singular values on the main diagonal. The SVD algorithm under consideration is outlined in Fig. 1 and corresponds to a modified version of the two-phase Golub-Kahan (GK) SVD algorithm [7] briefly summarized below:

[1]In the following, hardware-efficiency is measured in terms of circuit area A times the computation time T.

[2]The superscripts ^{H}, ^{T}, and * stand for conjugate transposition, transposition, and complex conjugation, respectively.

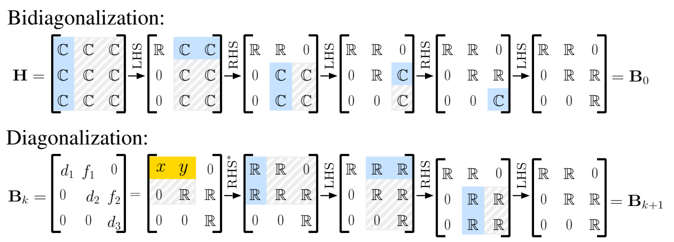

Fig. 1. Illustration of the bidiagonalization phase and diagonalization phase of the GK-SVD according to [7] for a 3×3-dimensional complex-valued matrix. All entries affected in the corresponding update have been highlighted.

1) Bidiagonalization: In the first phase of the algorithm, Givens rotations are alternately applied from the left-hand side (indicated with LHS in Fig. 1) and from the right-hand side (RHS) to the channel matrix, such that $\mathbf{H}$ is transformed into a real-valued bidiagonal matrix $\mathbf{B}_0 = \tilde{\mathbf{U}}^H\mathbf{H}\tilde{\mathbf{V}}$. Note that $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are both unitary and correspond to the total product of Givens-rotation matrices from the LHS and the RHS to the channel matrix $\mathbf{H}$, respectively.

2) Diagonalization: The second phase of the GK-SVD algorithm consists of multiple diagonalization steps (denoted by k in Fig. 1). Givens rotations are subsequently applied from both sides to the bidiagonal matrix $\mathbf{B}_k$, such that all off-diagonal entries f_i (for $i = 1, 2, \ldots, r-1$) converge to zero. To ensure convergence, the first Givens rotation in each diagonalization step (indicated with RHS* in Fig. 1) is performed with a modified input vector $[x\ y]^T$, where $y = t_{12}$ and $x = t_{11} - \mu$ uses the Wilkinson shift [7]

$$\mu = a_n + c - \text{sign}(c)\sqrt{a^2 + b_{n-1}^2} \tag{1}$$

with $c = \frac{1}{2}(a_{n-1} - a_n)$, $\mathbf{T} = \mathbf{B}_k^H\mathbf{B}_k$, and the trailing non-zero sub-matrix of $\mathbf{T}$ corresponds to

$$\mathbf{T}(n-1:n, n-1:n) = \begin{pmatrix} a_{n-1} & b_{n-1} \\ b_{n-1}^* & a_n \end{pmatrix}. \tag{2}$$

All Givens rotations performed in the diagonalization phase are also applied to the corresponding LHS and RHS unitary matrices. The diagonalization phase is stopped as soon as all off-diagonal entries are considered to be zero (cf. Sec. II-B). The result of the diagonalization phase corresponds to a SVD of $\mathbf{H}$ such that $\mathbf{\Sigma} = \mathbf{U}^H\mathbf{H}\mathbf{V}$.

B. Algorithm Modifications for Steering Matrix Computation

In order to perform steering matrix computation efficiently using an optimized fixed-point implementation, the following modifications have been applied to the GK-SVD [7]:

i) Since steering matrix computation only requires to compute $\mathbf{V}$, all Givens rotations from the LHS are *only applied* to $\mathbf{\Sigma}$ and hence, $\mathbf{U}$ is not explicitly computed. This modification avoids storage and computation effort for the matrix $\mathbf{U}$ and reduces the steering matrix computation time compared to a full GK-SVD.

ii) An off-diagonal entry f_i for $i = 1, 2, \ldots, r-1$ of $\mathbf{B}_k$ is considered to be zero, whenever [7]

$$f_i \leq 2^{-\varepsilon}(d_i + d_{i+1}) \tag{3}$$

applies and only integer tolerance values ε are chosen. This criterion avoids multiplications and leads to more accurate results than using the criterion employed in [6].

iii) The computational complexity of the diagonalization phase is data dependent and might require a large number of diagonalization steps. Hence, an early-termination criterion has been introduced to obtain a guaranteed throughput. As soon as $k = K_{\max}$, the diagonalization phase is stopped and the current matrix $\tilde{\mathbf{V}}$ is used as an estimate of the true steering matrix.

The parameters ε and $K_{\max}$ are used to optimize the arithmetic precision of the steering matrix computation hardware in Sec. IV and have a strong impact on the computational complexity of the modified GK-SVD algorithm.

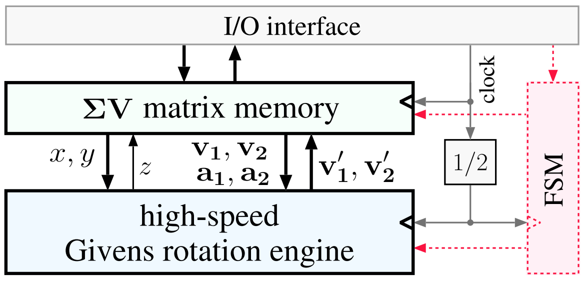

Fig. 2. Overview of the steering matrix computation architecture.

III. High-Speed VLSI Architecture

Fig. 2 provides an overview of the steering matrix computation unit designed for $M_T = M_R = 4$ MIMO systems. The architecture consists of a memory, a high-speed Givens rotation engine, and a finite state machine (FSM) which controls the memory and the rotation unit.

A. Matrix Memory

The matrix memory consists of one 32×32 bit two-port SRAM macro cell and stores $\mathbf{\Sigma}$ and $\mathbf{V}$. Prior to the SVD computation, the matrix $\mathbf{\Sigma}$ is initialized with the channel matrix $\mathbf{H}$ and $\mathbf{V} = \mathbf{I}_{M_T}$, where $\mathbf{I}_{M_T}$ is a $M_T \times M_T$-dimensional identity matrix. Each memory word corresponds to one complex-valued entry of either $\mathbf{\Sigma}$ or $\mathbf{V}$ and each real and imaginary part is represented by 16 bits. The Givens rotation engine described in the next section does only achieve moderate clock frequencies and the memory bandwidth has been identified to be critical for the overall throughput. Thus, we have decided to use the two-fold system clock for the memory in order to double the memory bandwidth.

B. Givens Rotation Engine

As shown in Sec. II, steering matrix computation mainly consists of two-dimensional vector rotations. A specialized Givens rotation engine initially developed for high-speed QR decomposition [8] is used to achieve low computation time of the modified GK-SVD algorithm.

Master-Slave CORDICs: Coordinate rotation digital computers (CORDICs) [9] can efficiently rotate complex numbers to real values and zero-out real-valued entries in $\mathbf{H}$ and are therefore, well-suited to compute the Givens rotations of the algorithm described in Sec. II. The main idea of the CORDIC is to decompose two-dimensional vector rotations into a sequence of hardware-friendly micro-rotations which only require elementary operations, such as additions/subtractions and arithmetic right shifts.

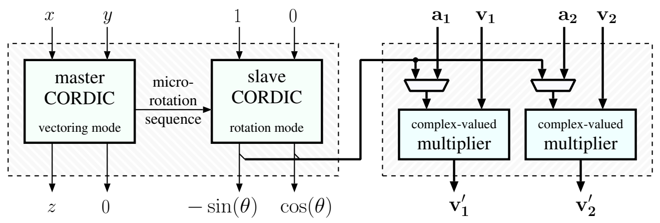

Fig. 3. High-speed Givens rotation engine: the master-slave CORDIC (shown on the left) computes $-\sin(\theta)$ and $\cos(\theta)$ to perform the corresponding rotations on $\mathbf{v}_1$ and $\mathbf{v}_2$ with complex-valued multipliers (shown on the right).

The *master* CORDIC (see Fig. 3) of the Givens rotation engine only operates in vectoring mode [9] and performs a two-dimensional rotation on $\mathbf{v} = [x\ y]^T$ such that

$$\begin{pmatrix} z \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{G}(\theta)\mathbf{v} \qquad (4)$$

where[3] $z \approx \pm\sqrt{x^2+y^2}$ and $\mathbf{G}(\theta)$ corresponds to the Givens rotation matrix. In the following paragraph, we describe a high-speed Givens rotation unit. In order to perform the same rotations on the remaining entries of $\mathbf{\Sigma}$ and $\mathbf{V}$ (cf. Fig. 3) with this unit, the *slave* CORDIC computes $-\sin(\theta)$ and $\cos(\theta)$ of the corresponding matrix $\mathbf{G}(\theta)$. This computation is achieved by feeding $[1\ 0]^T$ into the slave unit (which operates in rotation mode only) and by performing exactly the same micro-rotation sequence as the master CORDIC.

Rotation with Complex-Valued Multipliers: Since the modified GK-SVD requires more rotations than vectoring operations, a more faster architecture for two-dimensional rotations with $\mathbf{G}(\theta)$ can significantly reduce the total computation time. Instead of using CORDICs to rotate two-dimensional vectors, we use complex-valued multipliers (see Fig. 3) as in [8]

$$w = \big(\cos(\theta) - j\sin(\theta)\big)(x + jy)$$

which is fully equivalent to a one-cycle vector-matrix multiplication according to $[x'\ y']^T = \mathbf{G}(\theta)[x\ y]^T$, since the real part of w corresponds to x' and the imaginary part corresponds to y'. Note that multipliers are required anyways in order to compute (2) and hence, the external values $\mathbf{a}_1$ and $\mathbf{a}_2$ can also be applied to the input of the complex-valued multipliers as shown in Fig. 3.

IV. Hardware-Efficiency Optimization

Optimization in terms of hardware-efficiency is performed in two steps. First, the arithmetic precision is adjusted to the requirements of MIMO-OFDM systems. Second, hardware-efficiency optimization on micro-architectural level is used to reduce the steering matrix computation time without a significant increase in terms of circuit area.

A. Arithmetic Precision Optimization

Optimizing the arithmetic precision of the computational units reduces the area and the critical path of the circuit and therefore, adjusting the arithmetic precision to MIMO-OFDM systems can improve the overall AT-efficiency of the architecture. In order to adjust the tolerance ε in (3), the maximum number of diagonalization steps $K_{\max}$, the number of CORDIC micro rotations, and the fixed-point requirements without a significant error rate performance loss, bit error rate (BER) simulations[4] with the proposed steering matrix computation architecture have been performed,

[3]Note that the output of the master CORDIC $z \approx \pm\sqrt{x^2+y^2}$ can also be used to compute the square root in the Wilkinson shift (1).

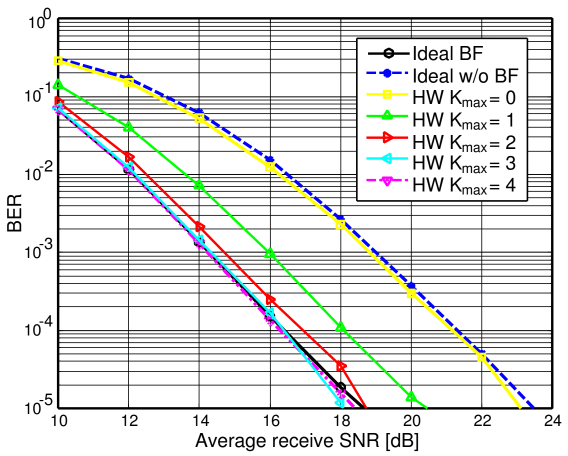

Fig. 4. Bit error rate (BER) performance of the steering matrix computation hardware dependent on $K_{\max}$. The ideal (floating-point) BER and the performance without using beamforming is shown for comparison.

In the first optimization step, we minimized the tolerance value, while the remaining circuit operates with floating-point precision. Simulations have shown that $\varepsilon = 4$ achieves near-optimal error rate performance. Then, using nine CORDIC micro rotations proved to induce no noticeable performance loss. In the next step, we evaluated all fixed-point parameters (i.e., number of integer and fraction bits) in the Givens rotation engine. We found that 16 bits per real and imaginary part in the memory and in the complex-valued multipliers are sufficient. Furthermore, 19 bits are required in the master CORDIC and 16 bits in the slave CORDIC to achieve a close-to-optimal BER. The final phase of our arithmetic precision optimization is shown in Fig. 4 and corresponds to the identification of the minimum $K_{\max}$ such that the BER remains near-optimal. Using less than three diagonalization steps results in a noticeable error rate performance degradation. Note that small values of $K_{\max}$ are beneficial in terms of worst-case computational complexity, but disadvantageous in terms of error rate performanc and hence, choosing $K_{\max} = 3$ leads to a good trade-off between BER performance and throughput.

B. Micro-Architectural AT-Efficiency Optimization

Since the high-speed Givens rotation engine performs two different operations on two different units, i.e., vectoring is performed with the CORDIC and rotations are carried out on complex-valued multipliers, we assume that the AT-efficiency of the entire unit[5] is heavily affected by the ratio between vectoring and rotation operations. Further influencing parameters are the CORDIC unroll factor [6], [9] and the number of complex-valued multipliers. In order to visualize

[4]We consider a coded (rate 1/2 convolutional code with constraint length 7, generator polynomials $[133_o\ 171_o]$, random interleaving) $M_R = M_T = 4$ MIMO-OFDM system without and with BF [3], 16-QAM (Gray mapping), 64 tones, and a soft-output MMSE detector. One code block contains 1024 bits, a TGn type C [10] channel model is used, and perfect channel state information is assumed.

[5]It is important to note that AT-efficiency optimization requires to consider the total circuit area, i.e., including memories and the control logic.

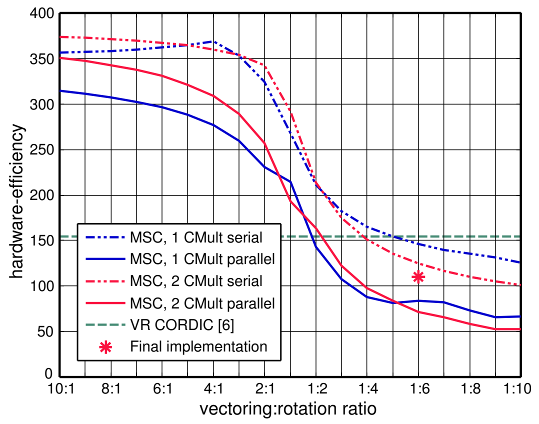

Fig. 5. Hardware-efficiency (measured in kGE µs) of the total architecture dependent on the vectoring/rotation ratio of the underlying algorithm. The master-slave CORDIC (MSC) clearly outperforms a minimum-area architecture using a vectoring/rotation (VR) CORDIC [6] for ratios lower than 1 : 5.

the dependence of AT-efficiency to the algorithm's vectoring/rotation ratio, hardware-efficiency can be written as

$$AT = A\big(N_v T_v + N_r T_r\big) \qquad (5)$$

where A, T_v, and T_r correspond to the total circuit area (in kGEs) and the time (in µs) required to perform *one* vectoring and *one* rotation operation in hardware, respectively. N_v and N_r in (5) denote the *total* number of vectoring and rotation operations, respectively, and the resulting vectoring/rotation ratio is defined as $R_{vr} = N_v/N_r$. Fig. 5 confirms our assumption that for the high-speed Givens rotation unit, AT-efficiency mainly depends on the vectoring/rotation ratio. The upper-limit of the curve corresponds to the case where vectoring and rotation operations are performed serially and the lower limit corresponds to a fully parallel operation. Note that data-dependence peculiarities of the algorithm in conjunction with the limited memory bandwidth restrict the maximum amount of parallelism and hence, the true hardware-efficiency of our final implementation lies in-between (cf. Fig. 5).

In order to allow a comparison with architectures that use a single vectoring/rotation CORDIC for Givens rotations, e.g., as in [6], the corresponding efficiency is also shown in Fig. 5. Using complex-valued multipliers for rotation often leads to a more efficient implementation than pure CORDIC-based architectures. The overhead of using two dedicated units for vectoring and rotation can be compensated for a ratio in the order of $R_{vr} = 1 : 5$. Furthermore, two complex-valued multipliers can achieve a better AT-efficiency and a higher throughput than a single instance. However, memory bandwidth limitations inhibit an efficiency gain by using more than two multipliers. Thus, we implemented our steering matrix computation architecture with two complex-valued multipliers. Further investigation of the AT-efficiency has shown that a CORDIC with unroll factor three is the best choice for our particular algorithm and implementation.

V. Implementation Results

The final implementation results are given in Tbl. I and correspond to post-synthesis figures for the steering matrix computation architecture. Note that the reference implementation results of MDU-I and MDU-II of [6] correspond to post-layout figures. Our presented steering matrix computation architecture clearly outperforms the reference implementation in terms of computation speed and hardware-efficiency (see Tbl. I). The 3.5-fold hardware-efficiency gain is mainly a result of joint algorithmic and architectural optimizations of the GK-SVD in Sec. II and of using the hardware-efficiency optimized high-speed Givens rotation engine described in Sec. III-B. However, we emphasize that the reference implementation [6] provides more flexibility and is programmable to perform QR decomposition as well as the full SVD of the channel matrix.

TABLE I
VLSI Implementation results of the steering matrix computation architecture in 0.18 µm (1P/6M) CMOS technology

	MDU-I [6]	MDU-II [6]	This Work
Area[a] [kGE]	42.3	38.1	42.3
Clock freq. [MHz]	133	272	149
Comp. time[b] [µs]	11.6	15.8	3.3
Efficiency [kGE µs]	489.4	602.0	136.3

[a]One gate equivalent (GE) corresponds to the area of a two-input drive-one NAND gate of size 9.7 µm^2.

[b]Corresponds to the SVD computation time of MDU-I and MDU-II using maximum precision [6] and to the modified GK-SVD computation time by using the steering matrix computation unit of this work.

VI. Conclusion

We described a hardware-efficient steering matrix computation architecture suitable for MIMO-OFDM systems with beamforming. Hardware-efficiency is achieved by modifying the Golub-Kahan SVD algorithm to efficiently compute steering matrices, by using and optimizing a high-speed Givens rotation engine, and by adjusting the arithmetic precision for MIMO-OFDM systems. The final VLSI implementation has proved to offer more than a 3.5-fold hardware-efficiency gain compared to a reference SVD implementation and is well-suited for wireless communication systems, such as IEEE 802.11n.

References

[1] H. Bölcskei, D. Gesbert, C. Papadias, and A. J. van der Veen, Eds., *Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge Univ. Press, 2006.

[2] IEEE Draft Specification, “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Enhancements for higher throughput,” *P802.11n/D1.0*, Mar. 2006.

[3] E. Akay, E. Sengul, and E. Ayanoglu, “Bit-interleaved coded multiple beamforming,” *IEEE Transactions on Communications*, vol. 55, pp. 1805–1811, Sept. 2007.

[4] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1999.

[5] N. D. Hemkumar and J. R. Cavalaro, “A systolic VLSI architecture for complex SVD,” in *Proceedings of the 1992 IEEE Intl. Symp. on Circuits and Systems*, May 1993.

[6] C. Studer, P. Blösch, P. Friedli, and A. Burg, “Matrix decomposition architecture for MIMO systems: Design and implementation trade-offs,” in *Proceedings of the 41th Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2007.

[7] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, Baltimore and London, 1996.

[8] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner, “VLSI implementation of a high-speed iterative sorted MMSE QR decomposition,” in *Proceedings of the IEEE Int. Symp. on Circuits and Systems*, May 2007.

[9] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*. Oxford University Press, New York, 2000.

[10] V. Erceg *et al.*, *TGn channel models*, May 2004, IEEE 802.11 document 03/940r4.