Gram-Schmidt-based QR Decomposition for MIMO Detection: VLSI Implementation and Comparison

P. Luethi†, C. Studer†, S. Duetsch‡, E. Zgraggen‡, H. Kaeslin†, N. Felber†, and W. Fichtner†
Integrated Systems Laboratory, ETH Zurich, 8092 Zurich, Switzerland
†{luethi, studer, kaeslin, felber, fw}@iis.ee.ethz.ch, ‡{seduetsc, eugenz}@ee.ethz.ch

***Abstract*— The QR decomposition (QRD) is an important prerequisite for many different detection algorithms in multiple-input multiple-output (MIMO) wireless communication systems. This paper presents an optimized fixed-point VLSI implementation of the modified Gram-Schmidt (MGS) QRD algorithm that incorporates regularization and additional sorting of the MIMO channel matrix. Integrated in 0.18 µm CMOS technology, the proposed VLSI architecture processes up to 1.56 million complex-valued 4×4-dimensional matrices per second.**

The implementation results of this work are extensively compared to the Givens rotation (GR)-based QRD implementation of Luethi et al., ISCAS 2007. In order to ensure a fair comparison, both QRD circuits have been integrated in the same IC manufacturing technology, with equal functionality, and the same numeric precision. The comparison of the implementation results clearly showed superiority of the GR-based VLSI solution in terms of area, processing cycles, and throughput.

I. Introduction

Multiple-input multiple-output (MIMO) technology is considered as one of the key elements for enabling high-throughput wireless communication. MIMO systems employ multiple antennas at both ends of the wireless link and can increase data rate by transmitting multiple data streams concurrently and in the same frequency band [1]. Consequently, many upcoming wireless communications standards, for example, IEEE 802.11n, IEEE 802.16e, and 3GPP LTE take advantage of MIMO technology. Unfortunately, the considerable throughput improvements entail a significant increase in signal processing complexity, especially on the receiver side.

The QR decomposition (QRD) is one of the key instruments for MIMO receivers, since numerous MIMO detection algorithms require the QRD of the channel matrix as starting point. The application of QRD ranges from linear detection to successive interference cancellation (SIC), and it also forms the basis of tree-search-based algorithms, such as the maximum-likelihood performance-achieving sphere decoder, e.g., [2]. Sorting of the channel matrix can be efficiently incorporated into the QR decomposition [3], termed as sorted QR decomposition (SQRD), leading to a significant error rate reduction in combination with SIC. Further reduction in terms of error rate performance for SIC can be achieved by performing the SQRD on a regularized channel matrix [4]. For tree-search-based detection algorithms, regularized SQRD significantly lowers the tree-search complexity at the cost of a negligible loss in error rate performance [2]. Note that regularized SQRD only entails a 50% higher computational effort compared to non-regularized SQRD [5] and therefore constitutes a promising candidate for SIC and tree-search-based MIMO receivers.

This work is supported by the STREP project No. IST-026905 (MASCOT) within the sixth framework program of the European Commission.

Contribution: In this work, we present a VLSI implementation of a matrix preprocessor performing regularized MGS-SQRD of 4×4-dimensional complex-valued matrices. The use of algorithmic optimizations enables to achieve close-to floating-point error-rate performance for the resulting VLSI architecture. Finally, we provide a fair comparison between this work and a Givens rotation (GR)-based reference implementation [5] – both integrated in 0.18 µm CMOS technology – and identify their corresponding pros and cons.

Outline: In the following, the system model and a brief overview of QRD-based MIMO detection are presented. Sec. II introduces the regularized SQRD that bases on a hardware-optimized version of the modified Gram-Schmidt QRD [6]. The corresponding VLSI architecture is presented in Sec. III. Finally, Sec. IV provides a comparison between this work and the GR-based reference SQRD implementation [5].

Notation: Bold uppercase and lowercase letters represent matrices and column vectors, respectively. The ith column vector of matrix $\mathbf{A}$ is denoted by $\mathbf{a}_i$, while $A_{i,k}$ stands for the element in row i and column k of $\mathbf{A}$. $\Re A_{i,k}$ represents the real part of $A_{i,k}$ and $\Im A_{i,k}$ the imaginary part of $A_{i,k}$. The superscript ^{T} denotes the transpose and ^{H} the Hermitian transpose. The expectation operator is $\mathbb{E}[\cdot]$.

A. MIMO System Model

We consider a MIMO system with M_T transmit and M_R receive antennas. The $M_R \times M_T$-dimensional matrix $\mathbf{H}$ represents the MIMO channel, the M_T-dimensional transmit signal vector is denoted by $\mathbf{s} = [s_1, s_2, \ldots, s_{M_T}]^T$, and the M_R-dimensional vector $\mathbf{n}$ represents the additive zero-mean i.i.d. complex Gaussian noise with variance σ_n^2 per complex dimension. The energy of the transmitted symbol vector is normalized such that $\mathbb{E}\left[\mathbf{s}\mathbf{s}^H\right] = \mathbf{I}_{M_T}$, where $\mathbf{I}_{M_T}$ is the $M_T \times M_T$-dimensional identity matrix. The M_R-dimensional receive vector $\mathbf{y} = [y_1, y_2, \ldots, y_{M_R}]^T$ corresponds to $\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$. The signal-to-noise ratio (SNR) per receive antenna is M_T/σ_n^2.

B. MIMO Detection Based on Regularized SQRD

Sorting and Regularization: QRD for MIMO detection starts by decomposing $\mathbf{H}$ into a unitary matrix $\mathbf{Q}$ and an upper-triangular matrix $\mathbf{R}$ with real-valued non-negative elements on the main diagonal. In order to improve the detection performance, the SQRD algorithm efficiently computes $\mathbf{H} = \mathbf{Q}\mathbf{R}\mathbf{P}^T$ such that the sorting $R_{i,i} \leq R_{j,j}$ for $i < j$

is approximated. The $M_T \times M_T$-dimensional permutation matrix $\mathbf{P}^T$ accounts for the sorting induced by the SQRD algorithm. The basic idea underlying regularized SQRD [4] is to reduce the probability of ill-conditioned channel matrices by computing the SQRD of the matrix

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma_n \mathbf{I}_{M_T} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_a \\ \mathbf{Q}_b \end{bmatrix} \bar{\mathbf{R}} \mathbf{P}^T \tag{1}$$

where $\bar{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q}_a^T & \mathbf{Q}_b^T \end{bmatrix}^T$ and $\bar{\mathbf{R}}$ is upper-triangular with real-valued elements on the main diagonal. The dimensions of matrices $\bar{\mathbf{Q}}_a$, $\bar{\mathbf{Q}}_b$, $\bar{\mathbf{R}}$ are $M_R \times M_T$, $M_T \times M_T$, and $M_T \times M_T$, respectively. Note that choosing the regularization parameter according to σ_n is termed as MMSE-SQRD [4].

MIMO Detection: In order to efficiently perform MIMO detection, the input-output relation of the MIMO channel can be transformed into $\hat{\mathbf{y}} = \bar{\mathbf{R}}\mathbf{P}^T\mathbf{s} + \hat{\mathbf{n}}$, where $\hat{\mathbf{y}} = \mathbf{Q}_a^H\mathbf{y}$ and where $\hat{\mathbf{n}}$ contains noise $\mathbf{Q}_a^H\mathbf{n}$ and additional (self-)interference. This modified I/O relation can be solved with reduced computational complexity, e.g., through SIC [4] or sphere decoding [2].

II. Gram-Schmidt-based Regularized SQRD for Fixed-Point Implementation

The MMSE-SQRD of $\bar{\mathbf{H}}$ can be performed through Gram-Schmidt orthogonalization [6] or through a sequence of unitary transformations, e.g., by using Givens rotations or Householder reflections. Algorithm 1 essentially corresponds to the original SQRD algorithm in [3], but has been refined as described in the following three subsections.

A. Modified Gram-Schmidt Algorithm

The classical version of the GS algorithm has proven to be numerically less stable than the MGS algorithm [7]. The regularized MGS-QRD algorithm performs a successive orthogonalization of the columns $\bar{\mathbf{h}}_i$ starting from the regularized channel matrix $\bar{\mathbf{H}}$. To this end, the algorithm initializes $\bar{\mathbf{Q}} = \bar{\mathbf{H}}$ and iteratively computes the QRD in M_T steps $i = 1, 2, \ldots, M_T$. The final matrices $\bar{\mathbf{Q}}$ and $\bar{\mathbf{R}}$ at step $i = M_T$ correspond to the ones given in (1). For each step i, the element on the main diagonal of $\bar{\mathbf{R}}$ is computed as $\bar{R}_{i,i} = \sqrt{\bar{\mathbf{q}}_i^H \bar{\mathbf{q}}_i}$. Then, each element $k = i+1, i+2, \ldots, M_T$ on the ith row of $\bar{R}_{i,k}$ is computed as $\bar{R}_{i,k} = \bar{\mathbf{q}}_i^H \bar{\mathbf{q}}_k$ and the associated column $\bar{\mathbf{q}}_k$ is updated according to

$$\bar{\mathbf{q}}_k \leftarrow \bar{\mathbf{q}}_k - \frac{\bar{R}_{i,k}}{\bar{R}_{i,i}} \bar{\mathbf{q}}_i. \tag{2}$$

B. Iterative Sorting Strategy

The iterative sorting strategy employed in this paper corresponds to the one proposed in [3]. For each step i – prior to the computation of the elements of $\bar{\mathbf{R}}$ and the columns in $\bar{\mathbf{Q}}$ – the column $\bar{\mathbf{q}}_l$ ($l = i, i+1, \ldots, M_T$) with the smallest squared ℓ^2-norm is determined and processed first. To this end, the ith column of $\bar{\mathbf{R}}$ (and of $\bar{\mathbf{Q}}$, respectively) needs to be exchanged with the one associated with the smallest squared ℓ^2-norm. In order to avoid expensive norm recalculation in each step, an economic norm-updating strategy is used. The squared ℓ^2-norm associated with the columns of $\mathbf{Q}$ are initialized at the beginning of the MGS-SQRD algorithm as $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_{M_T}]^T$, and then iteratively updated, as shown in Alg. 1 on lines 2 and 18, respectively.

Algorithm 1 MGS-SQRD with column exponents

1: $\bar{\mathbf{Q}} \leftarrow \bar{\mathbf{H}}$; $\bar{\mathbf{R}} \leftarrow \mathbf{0}_{M_T \times M_T}$; $\mathbf{P} \leftarrow \mathbf{I}_{M_T}$
2: $\boldsymbol{\xi} \leftarrow \left[\|\bar{\mathbf{q}}_1\|^2, \|\bar{\mathbf{q}}_2\|^2, \ldots, \|\bar{\mathbf{q}}_{M_T}\|^2\right]^T$; $\mathbf{e} \leftarrow \mathbf{0}_{M_T \times 1}$
3: **for** $i = 1, 2, \ldots, M_T$ **do**
4: $\quad j \leftarrow \arg\min_{l=i,i+1,\ldots,M_T} \xi_l$
5: $\quad$ exchange columns i and j in $\bar{\mathbf{Q}}$, $\bar{\mathbf{R}}$, and $\mathbf{P}$
6: $\quad$ exchange elements i and j in $\boldsymbol{\xi}$ and $\mathbf{e}$
7: $\quad \alpha \leftarrow 2\lceil \frac{\log_2 \xi_i}{2} \rceil$
8: $\quad u \leftarrow \sqrt{\xi_i \cdot 2^{-\alpha}}$
9: $\quad d \leftarrow 1/u$
10: $\quad \bar{R}_{i,i} \leftarrow u \cdot 2^{\alpha/2}$
11: $\quad \bar{\mathbf{q}}_i \leftarrow d \cdot \bar{\mathbf{q}}_i$
12: $\quad \beta \leftarrow \left\lceil \log_2\left(\max_{l=1,2,\ldots,M_R+M_T} \left[\max\left[|\Re\bar{Q}_{l,i}|, |\Im\bar{Q}_{l,i}|\right]\right]\right)\right\rceil$
13: $\quad \bar{\mathbf{q}}_i \leftarrow \bar{\mathbf{q}}_i \cdot 2^{-\beta}$
14: $\quad e_i \leftarrow e_i + \beta - \alpha/2$
15: $\quad$ **for** $k = i+1, i+2, \ldots, M_T$ **do**
16: $\quad\quad v \leftarrow \bar{\mathbf{q}}_i^H \cdot \bar{\mathbf{q}}_k$
17: $\quad\quad \bar{R}_{i,k} \leftarrow v \cdot 2^{e_i + e_k}$
18: $\quad\quad \xi_k \leftarrow \xi_k - |v|^2 \cdot 2^{2(e_i+e_k)}$
19: $\quad\quad \bar{\mathbf{q}}_k \leftarrow \bar{\mathbf{q}}_k \cdot 2^{-2\max[e_i,0]} - v \cdot \bar{\mathbf{q}}_i \cdot 2^{2\min[e_i,0]}$
20: $\quad\quad e_k \leftarrow e_k + 2\max[e_i, 0]$
21: $\quad$ **end for**
22: **end for**
23: $\bar{\mathbf{Q}} \leftarrow [\bar{\mathbf{q}}_1 \cdot 2^{e_1}, \bar{\mathbf{q}}_2 \cdot 2^{e_2}, \ldots, \bar{\mathbf{q}}_{M_T} \cdot 2^{e_{M_T}}]$

C. Column-Wise Reduced Precision Floating-Point Technique

An important characteristic of the MGS algorithm is that most of the operations are performed on entire columns of the matrix $\mathbf{Q}$. While the dynamic range of elements within one column remains relatively small, it can be large among different columns $\bar{\mathbf{q}}_i$, due to the division operation in (2) (cf. lines 9 and 11 of Alg. 1). As a consequence, we employ a column-wise floating-point technique similar to that proposed in [8] in order to reduce numeric precision issues. To this end, each column $\bar{\mathbf{q}}_i$ is associated with an individual exponent $e_i \in \mathbb{Z}$ such that $\tilde{\mathbf{q}}_i = \bar{\mathbf{q}}_i \cdot 2^{e_i}$. Note that the transformation from $\bar{\mathbf{q}}_i$ to $\tilde{\mathbf{q}}_i$ and vice versa can be computed efficiently in hardware by using arithmetic shift operations only.

III. VLSI Architecture for MGS-SQRD

The target application for the proposed VLSI implementation is channel preprocessing in MIMO-OFDM communication systems, where a large number of complex-valued channel matrices have to be processed in a short time [9]. The VLSI architecture for the sorted QR decomposition employs the MGS algorithm described in Alg. 1. The top-level diagram of the SQRD architecture is shown in Fig. 1. The main blocks of the architecture are a square root (SQRT), an inverse (INV) and a subtraction (SUB) unit, a complex-valued multiply-and-accumulate ($\mathbb{C}$-MAC) block, and various memories.

A. Throughput Optimizations

For high-throughput VLSI architectures, an important design aspect is the limitation in memory bandwidth. Many algorithms imply data dependencies, which are challenging to realize efficiently in VLSI architectures. Choosing an adequate memory organization combined with a dedicated access scheme allows to greatly reduce the number of individual

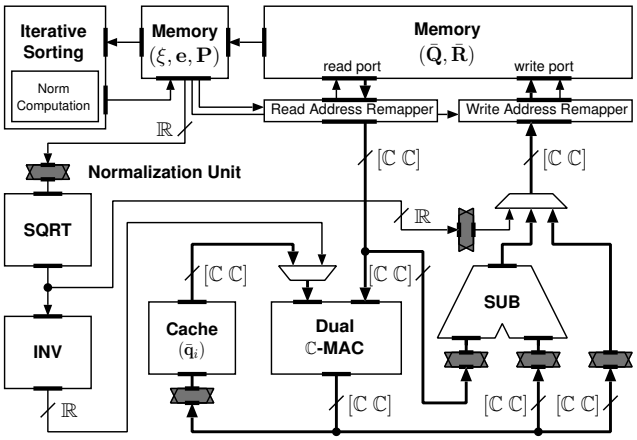

Fig. 1. VLSI architecture of the MGS-SQRD showing the main building blocks of the datapath. The dark shaded components represent normalization units for the column exponents of the matrices. $[\mathbb{C}\ \mathbb{C}]$ stands for two complex-valued signals, $\mathbb{R}$ denotes a real-valued signal.

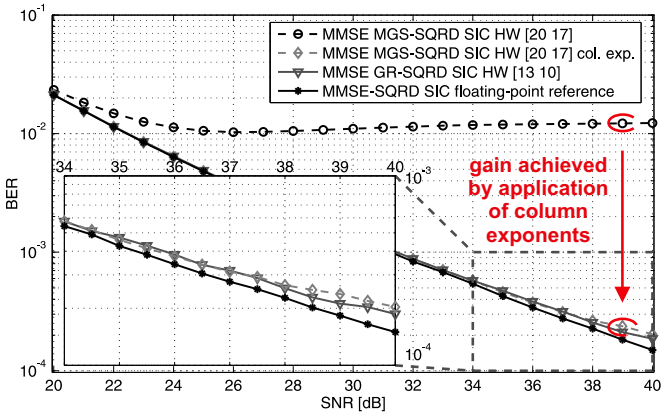

Fig. 2. BER performance of different fixed-point SQRD algorithms. The corresponding simulation employs uncoded 4×4 MIMO, 16-QAM modulation, and a detection stage performing SIC. Except for the fixed-point SQRD model, all remaining components are realized using floating-point arithmetic. We assume perfect channel state information at the receiver. [x y] denotes total number of bits including the sign bit, and fractional bits, respectively.

memory accesses, thus keeping the required memory bandwidth of the VLSI architecture non-critical. For the MGS-SQRD architecture, this aspect has been addressed by storing two matrix elements with both real and imaginary parts together at one memory location. Although this access scheme proved to work well in general for this VLSI architecture, there remained particular memory access issues in the design. For instance, a memory access conflict emerged in calculations involving two different columns of the matrix $\bar{\mathbf{Q}}$ (cf. line 16 of Alg. 1). This conflict was solved by introducing local cache registers for $\bar{\mathbf{q}}_i$, as shown in Fig. 1.

The exploration of the VLSI design space for the SQRT and INV blocks showed that moderately iteratively decomposed architectures result in the best trade-off between silicon area and processing time, without incurring detrimental effects on the circuit's overall throughput. As a consequence, the SQRT and INV blocks were designed to use three and six clock cycles for one computation, respectively.

The MGS-SQRD algorithm exhibits a computationally intensive section on line 16 of Alg. 1. This complex-valued scalar product has been realized by using dedicated $\mathbb{C}$-MAC units. The application of two parallel $\mathbb{C}$-MAC units emerged to be the most viable solution for addressing the critical design aspects, i.e., limitations in memory bandwidth, delivery of high throughput and economic use of hardware resources. The strict data dependencies in lines 7-10 of Alg. 1 offer only little opportunities for additional parallelization: The sequence of square root, inverse, and multiplication is difficult to compute efficiently in parallel. Nevertheless, any negative effects on the overall throughput have been minimized by already starting the computation of both sort order and square root, while the inner loop involving the $\mathbb{C}$-MAC at lines 15-21 of Alg. 1 is still being executed.

B. Numeric Precision

The inverse on line 9 of Alg. 1 is the key issue for adapting the MGS-SQRD algorithm to the fixed-point design space. Since inverse and division operations significantly increase the dynamic range of the result, special care needs to be taken in case of fixed-point representations. Our SQRD architecture employs column exponents introduced in Sec. II-C, which are realized as dedicated normalization units, indicated with dark shaded blocks in Fig. 1. A normalization unit checks for the largest absolute value amongst all elements of a matrix column, and then performs the corresponding arithmetic shift (of ± 15 bit positions) in order to best maintain the overall numeric precision.

TABLE I
WORD LENGTH AND SILICON AREA OF DIFFERENT UNITS

Unit	Word Length[a] [bit]	Area [mm^2]	Area [%]
Memory	20	0.247	25
SQRT	36	0.036	4
INV	33	0.029	3
Dual $\mathbb{C}$-MAC	20	0.347	35
Normalization	20	0.089	9
Cache	20	0.085	8
Iterative Sorting	20	0.023	2
Others	-	0.141	14

[a]Indicates the number of bits per complex dimension.

The result of this effort can be seen in the bit error rate (BER) performance plot depicted in Fig. 2. The introduction of column exponents clearly shows the significant BER performance improvement of approximately two orders of magnitude at 40 dB SNR.

C. Implementation Results

The proposed VLSI architecture of Fig. 1 has been integrated in UMC 0.18 µm 1P/6M CMOS technology, requiring a core area of 0.997 mm^2. It achieves a maximum clock frequency of 162 MHz and a throughput of up to 1.56 million SQRD/s. A summary of applied internal word lengths for various blocks and the corresponding synthesis area is given in Tbl. I, the final layout of the circuit is shown in Fig. 3.

IV. COMPARISON

A fair comparison of two different VLSI circuits implies the same functionality, numeric precision, and manufacturing technology for both devices. To compare the numeric precision of MGS-SQRD and GR-SQRD, the same simulation setup has

TABLE II
POST SYNTHESIS RESULTS FOR BOTH VLSI IMPLEMENTATIONS

	This work	Luethi et. al. [5]
Algorithm	MGS-SQRD	GR-SQRD
Memory word length [bit]	20	13
Core area[a] [mm^2] / [kGE]	0.997 / 61.8	0.785 / 48.7
Max. f_{clk} [MHz]	162	166
T_{SQRD} [ns]	641	480
Throughput [M SQRD/s]	1.56	2.08
SQRD processing cycles	104	80
AT-product[b] [ns mm^2]	639	377

[a]One gate equivalent (GE) corresponds to a two-input drive-2 NAND gate.
[b]Corresponds to the product of the core area and T_{SQRD}.

been used. Moreover, we focus on 4×4-dimensional complex-valued matrix decompositions employing regularized SQRD. The parameters of both architectures have been adjusted to provide close-to floating-point BER performance up to a SNR of 40 dB, as shown in Fig. 2. Finally, the GR-SQRD-based reference implementation [5] using a 0.25 µm CMOS process has been re-integrated in 0.18 µm technology.

A. Comparison of Algorithm and Architecture

The GR-SQRD algorithm allows an implementation by the use of unitary transformations only. The main advantage of this algorithm lies in the fact that the GR can efficiently be realized in hardware by CORDIC arithmetic [5], which preserves the total power of the operands. Algorithms with this property are well suited for fixed-point VLSI implementations because the dynamic range of the variables is strictly confined. In contrast, the MGS algorithm consists of non-unitary transformations employing division and square root operations. This leads to an increased dynamic range and renders fixed-point implementation difficult. To compensate for this, column exponents and larger word lengths are necessary (cf. Tbl. II) to maintain a comparable numeric precision (cf. Fig. 2).

The MGS-SQRD architecture contains many different computational units, some of them having low processing activities, e.g., the inverse and the square root units are used only four times during a 4×4-dimensional regularized SQRD. The required algorithmic modifications for the reduced floating-point approach further exacerbate this problem as many normalization units with low overall activity are introduced. On the other hand, the GR-SQRD reference architecture [5] holds a higher and more uniform utilization of the internal processing blocks. The reason for this is better regularity in data flow, offering the potential for better hardware-efficiency.

B. Comparison of VLSI Implementations

The results in Tbl. II demonstrate the benefits of a VLSI implementation for regularized SQRD based on Givens rotation rather than Gram-Schmidt. This is a consequence of algorithmic and architectural differences.

Other QRD implementations have been described in the literature, e.g., [8], [10]. Unfortunately, those implementations do not perform SQRD. Furthermore, the architecture described in [8] is designed for FPGA implementation, while the ASIC design in [10] performs the QRD of real-valued matrices. Thus, a fair comparison with our VLSI implementations of MGS-SQRD and GR-SQRD is currently not possible.

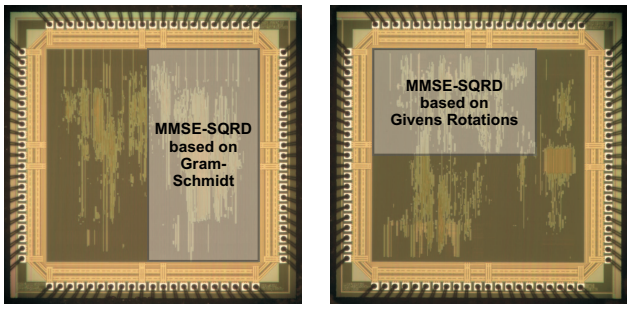

Fig. 3. Chip micrographs of the Gram-Schmidt-based (left) and the Givens rotation-based (right) VLSI implementation of sorted QR decomposition. Both designs were integrated in UMC 0.18 µm 1P/6M CMOS technology. Adjacent core logic is not related to the presented designs.

V. CONCLUSIONS

Compared to the MGS-SQRD algorithm proposed in this paper, the GR-SQRD described in [5] exhibits a number of significant economic benefits for a VLSI implementation: The GR-SQRD algorithm has a smaller numeric range of operands throughout and is well-suited for fixed-point CORDIC arithmetics. With respect to an efficient VLSI realization, the GR-SQRD has a more regular data flow employing fewer computational units of distinct nature. This fact leads essentially to a higher and more uniform utilization of all components. Moreover, the GR-SQRD-based VLSI implementation delivers a higher throughput combined with a smaller circuit size and hence, offers a superior hardware-efficiency.

While unfavorable for the modified Gram-Schmidt method, this is an important insight gained through our work as the modified Gram-Schmidt approach currently is more widely adopted for VLSI implementations in practice. We believe this is because, firstly, Gram-Schmidt prevails in software programs that use floating-point arithmetics and secondly, Givens rotations require a more sophisticated degree of understanding in VLSI design if CORDIC arithmetics are employed.

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] C. Studer, A. Burg, and H. Bölcskei, "Soft-Output Sphere Decoding: Algorithms and VLSI Implementation," *IEEE J-SAC*, vol. 26, no. 2, pp. 290–300, Feb. 2008.
[3] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, "Efficient Algorithm for Decoding Layered Space-Time Codes," *IEE Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.
[4] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, "MMSE Extension of V-BLAST based on Sorted QR Decomposition," in *Proc. of IEEE VTC, Fall*, vol. 1, Oct. 2003, pp. 508–512.
[5] 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 *Proc. of IEEE ISCAS*, May 2007, pp. 1421–1424.
[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins Univ. Press, 1996.
[7] A. Björck, "Numerics of Gram-Schmidt Orthogonalization," *Linear Algebra and Its Applications*, vol. 198, pp. 297–316, Feb. 1994.
[8] H. Kim, W. Zhu, J. Bhatia, K. Mohammed, A. Shah, and B. Daneshrad, "An efficient FPGA based MIMO-MMSE Detector," in *Proc. of EUSIPCO*, Sept. 2007, pp. 1131–1135.
[9] D. Perels, S. Haene, P. Luethi, A. Burg, N. Felber, W. Fichtner, and H. Bölcskei, "ASIC Implementation of a MIMO-OFDM Transceiver for 192 Mbps WLANs," in *Proc. IEEE ESSCIRC*, Sept. 2005, pp. 215–218.
[10] C. Singh, S. Prasad, and T. Balsara, "VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition," in *Proc. of VLSI Design 2007*, Jan. 2007, pp. 836–841.