Matrix Decomposition Algorithms for MIMO receivers: Flexibility vs. Efficiency Tradeoffs in a Library-based Tool-Assisted SDR Development

Venkatesh Ramakrishnan*, Tobias Veerkamp*, Marc Adrat†, Gerd Ascheid* and Markus Antweiler†
*Institute for Integrated Signal Processing Systems, RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany
†Fraunhofer Institute for Communication, Information Processing & Ergonomics (FKIE), Wachtberg, Germany

***Abstract*—Multiple input and multiple output (MIMO) is one of the key technologies used in wireless standards like LTE and WiMaX. Matrix decomposition of the channel matrix in the form of QR decomposition (QRD) is needed for advanced MIMO demapping algorithms like sphere decoder. The implementation of QRD has to be highly efficient due to its computation-intensive nature. On the other hand, software defined radios (SDRs) require flexibility in several forms, e.g. support for different algorithms. The contradictory nature of flexibility and efficiency requires tradeoffs to be made between them in SDR development. In this paper, we analyze tradeoffs by using MMSE-SQRD as a case study. We have implemented two algorithms for performing SQRD in four different methods with varying degree of portability, efficiency and reusability. Focus will given on a library based SDR development approach in our investigations where constraint aware mapping can be performed with tool assistance at a high abstraction level.**

I. Introduction

Multiple input and multiple output (MIMO) systems can provide high throughput by exploiting multi-path propagation and diversity. It is one of the enabling technologies in upcoming wireless standards like LTE and WiMax. One of the highly complex components in a MIMO system is the MIMO demapper, which separates the superposed received data streams into the multiple transmitted streams. In order to simplify and perform MIMO demapping efficiently, several matrix decomposition methods are used in conjunction with MIMO detection schemes. Since matrix decomposition represents a computation intensive component of a MIMO receiver, its implementation has a direct impact on the over-all system efficiency and therefore has to be highly efficient. This makes the implementation of matrix decomposition challenging.

Flexibility, a key feature in software defined radios (SDRs), is sought in several forms. For example, flexibility in choosing algorithms and their implementation in a SDR hardware platform can enable the radio to efficiently use, e.g. spectrum resources, according to the different environmental conditions. Portability, the ease with which an implementation of a wireless standard can be ported to different hardware platforms, is another form of flexibility. Reusability is a common form for enabling flexibility in SDR development. For example, if an algorithm or implementation can be reused for performing a different functionality, it can (re)configured/programmed on-the-fly.

Even though a flexible solution may not always yield the same performance, e.g. with respect to energy, when compared to a dedicated solution, the difference can be made tolerable with careful engineering and application specific optimizations. A key advantage of reusability is the reduction of the system development time and time-to-market. Portability can be significantly increased with reusable components. However, identifying such reusable components and implementing them in a flexible and efficient way can be challenging.

Efficiency, which is also a prerequisite for SDRs, can take several forms. For example, energy efficiency is paramount for increasing battery life in mobile devices. Area efficiency has become important, particularly in hand-held devices, due to the growing need to accommodate more hardware elements.

It is well known that fully flexible hardware architectures, e.g. general purpose processors (GPPs), are expensive in terms of area and energy consumption. In order to improve energy efficiency and at the same time meet the computation needs of the application, heterogeneous hardware platforms are a promising solution. Apart from the processing elements (PEs), the type of implementation on a PE can play a vital role on efficiency as well. For example, hand-coded assembly implementation that is optimized for the architecture of a processor can consume less cycles, by a few orders of magnitude, when compared to a generic C implementation. However, implementation in C has high portability when compared to assembly.

From the above discussions, it is clear that flexibility and efficiency are contradictory in nature. Therefore, tradeoffs among them have to be made at every stage of SDR development. However, in order to make the tradeoff decisions it is important to quantify the differences in performance. This requires implementation of an algorithm or a component on a whole range of PEs, spanning GPPs, dedicated application specific integrated circuits (ASICs), digital signal processors (DSPs), field programmable gate arrays (FPGAs), etc. using generic C, C with intrinsics and hand written assembly (for programmable processors) and hardware description language

This research project was performed in the Ultra High-Speed Mobile Information and Communication (UMIC) research centre under the support of the Technical Center for Information Technology and Electronics (WTD-81), Germany.

(HDL) (for FPGA and ASIC). Such a comprehensive investigation is beyond the scope of this paper. One of the main focus of our investigations is to analyze the flexibility vs. efficiency tradeoffs while implementing a wireless standard using reusable algorithmic kernels.

Due to the offer of both portability and efficiency for SDR development, the library-based Nucleus methodology [1] is used as the backbone for our investigations. The Nucleus methodology is based on a standardized library, consisting of algorithmic computation intensive kernels, Nuclei. The description of a wireless standard is done using the Nuclei library as the basis. Due to the standardization of the Nuclei library, vendors can provide efficient implementations for Nuclei, known as Flavors, as a part of board support package (BSP) for a hardware platform. Tools are used for selecting an implementation from the BSP, that meets the constraints of a wireless standard like latency. This paper can also be seen as a case study for identifying a Nucleus, which can be reused in several standards for performing different functionalities.

The rest of the paper is structured as follows. The system model that is used for our investigations is explained in Section II. Our contributions and related work are given in Section III. The algorithms that were implemented for performing minimum mean squared error (MMSE) sorted QR decomposition (SQRD) are explained in Section IV. The different versions of implementing the MMSE-SQRD algorithms and their building blocks are described in Section V. Results and observations are presented in Section VI. Finally, conclusions are drawn.

Notations: A matrix $\mathbf{Q}$ is denoted by a uppercase letter, jth column of $\mathbf{Q}$ is indicated by a lowercase letter $\mathbf{q}_j$. $\mathbf{Q}_{j,k}$ indicates the matrix element at jth row and kth column. The subscript in lowercase letter (which itself has a subscript of uppercase letter) of a matrix $\mathbf{Q}_{n_T \times n_R}$ indicates the dimension $n_T \times n_R$ of $\mathbf{Q}$ (a square matrix has a single subscript). For example, $\mathbf{I}_{n_T}$ represents an identity matrix of order n_T. A matrix with a subscript in lowercase letter a indicates a submatrix $\mathbf{Q}_a$ of the augmented matrix $\bar{\mathbf{Q}}$. $\mathrm{Re}\{\mathbf{Q}_{j,k}\}$ and $\mathrm{Im}\{\mathbf{Q}_{j,k}\}$ indicate the real and imaginary parts of $\mathbf{Q}_{j,k}$ respectively. A complex and real number is represented by the symbol, $\mathbb{C}$ and $\mathbb{R}$ respectively. The superscript ^{H} indicates the Hermitian transpose. $\|.\|$ denotes the Euclidean norm operator. The superscripts $\hat{}$ and ^{T} indicate the estimated value and transpose respectively.

II. System Model

We consider a MIMO-OFDM system with n_T transmitting and n_R receiving antennas. The received $n_R \times 1$ dimensional signal vector $\mathbf{y}$ can be expressed as

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

where $\mathbf{H}$ is the $n_R \times n_T$ channel matrix, $\mathbf{s}$ is the $n_T \times 1$ transmitted signal vector and $\mathbf{n}$ is the $n_R \times 1$ noise vector with zero mean and variance $\frac{1}{2}\sigma_n^2$ respectively.

MMSE solution, which takes the noise in the system into account for estimating the transmitted signal vector at the receiver side, offers better bit error rate (BER) performance when compared to zero forcing (ZF) solution (see Figure 1). Therefore, this paper considers a MMSE solution for implementation. The channel matrix $\mathbf{H}$ has to be extended to the so called *complex-valued augmented channel matrix* $\bar{\mathbf{H}}$, which is defined as

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{\frac{n_T}{E_s}}\sigma_n \mathbf{I}_{n_T} \end{bmatrix} \tag{2}$$

where E_s is the transmitted signal power. $\mathbf{I}_{n_T}$ denotes a $n_T \times n_T$-dimensional identity matrix. The inverse of the complex augmented channel matrix leads to the estimation of the transmitted symbol vector $\hat{\mathbf{s}}$.

$$\hat{\mathbf{s}} = \bar{\mathbf{H}}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_{n_T} \end{bmatrix} \tag{3}$$

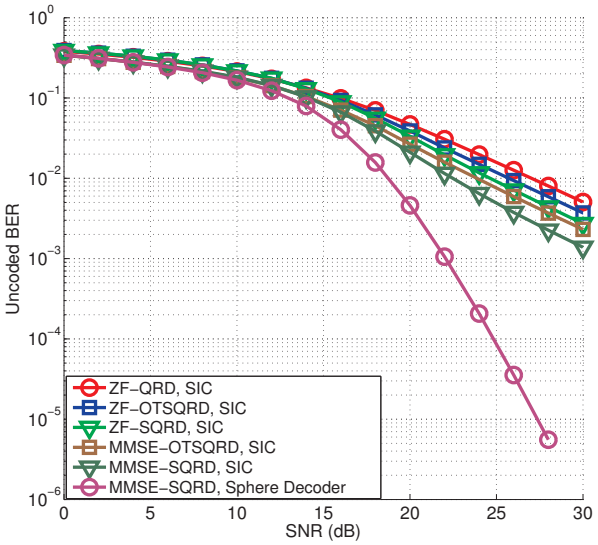

Fig. 1. Uncoded BER performance of a 4×4 MIMO system using 16QAM modulation for different MIMO preprocessing schemes

A. MIMO Processing

One of the most processing intensive blocks in the MIMO receiver is MIMO processing (also mentioned as MIMO demapping), which reverses the channel effects in order to recover the transmitted stream from the received superposed data streams. Figure 2 illustrates the block diagram of MIMO processing. The functionality of MIMO processing can be separated into MIMO preprocessing and MIMO detection. The functionality of the preprocessing block depends on the MIMO detection scheme. MIMO detection schemes can vary from the simple linear detection method to the advanced methods like sphere decoder (SD). Figure 1 illustrates the difference in BER performance between successive interference cancelation (SIC) and depth first, sphere decoding scheme.

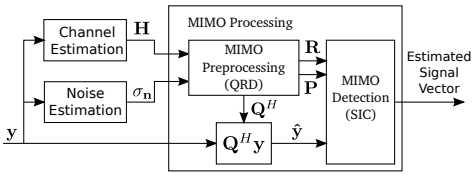

Fig. 2. Block diagram of the MIMO processing based on QRD

B. MIMO Preprocessing

For performing MIMO detection using the linear technique, the preprocessor can compute the matrix inversion of $\mathbf{H}$. Since advanced MIMO detection schemes like SIC and tree-based SD need the QR decomposition (QRD) of the channel matrix $\mathbf{H}$, the preprocessor computes QRD while using them.

Implementation of preprocessing block can be critical for wireless standards that support very high throughput and therefore should be highly efficient. To illustrate this point, let us consider two scenarios in LTE standard. Assuming a maximum speed of the mobile handset as 350 km/hour and a carrier frequency of 2.5 GHz, the coherence time is given by 1.2 milli-second (ms) [2]. With 1200 data carrying sub-carriers, time to perform the QRD of $\mathbf{H}$ belonging to each sub-carrier is: 1.2 ms / 1200 = 1 micro-second (μs). For a slightly relaxed scenario with the speed of mobile handset as 45 km/hour (coherence time of 10 ms) and other conditions being the same as above, the maximum time to perform QRD is: 8.3 μs. In order to meet such low timing, implementation of MIMO preprocessing must be highly efficient making it challenging.

As shown in Figure 1, sorting the channel matrix before performing QRD improves BER performance significantly, resulting in the so-called SQRD. We have considered only SQRD algorithms in this paper. As 2×2 MIMO is of reasonable complexity and is proposed for LTE to achieve a peak downlink download rate of 150 Mbps, we have not considered higher order MIMO systems for implementation in this paper. Moreover, most of the observations made from our results are general in nature and applicable to higher order MIMO systems as well. It is important to note that one-time full-column sorted QRD (OTSQRD) technique in a 4×4 MIMO system becomes fully sorted technique, referred as SQRD, in a 2×2 MIMO system.

III. Contributions and Related Work

In this paper, we have implemented two algorithms, modified Gram Schmidt (MGS) and Givens rotation (GR) for performing MMSE-SQRD. The GR algorithm is implemented in several ways that are based on algorithms and components which are highly reusable. For example, the coordinate rotation digital computer (CORDIC) algorithm, which has been used for implementing GR, can be used for realizing several other functionalities like phase correction [3]. Due to the nature of implementation (without flexibility), MGS can be used as a dedicated solution in this paper and can be used for comparing the performance of the flexible implementations of GR. The Texas Instruments C64x+ DSP is used as the PE for our analysis. For analyzing the portability vs. efficiency tradeoffs, all the implementation-variants (mentioned as *variants* in this paper) have been realized in floating-point C, fixed-point C, fixed-point C with intrinsics and hand-optimized Very Long Instruction Word (VLIW) assembly. This includes both 16-Bit and 32-Bit implementations for analyzing different input/output (I/O) data-width. Due to the high computational complexity in performing QRD, it serves as a good case study for our analyses on flexibility vs. efficiency tradeoffs.

Several methods such as GS-, LDL-, LR-, LU-, QR- and Jacobi singular value decomposition (SVD) can be used for performing matrix decomposition. Among them, QRD is a popular candidate for MIMO preprocessing. This is due to the necessity for QRD of the channel matrix in some advanced MIMO detection schemes like SD. Three most popular algorithms for implementing a QRD are: the Householder reflections algorithm [4, 5], the MGS algorithm [6, 7] and the GR algorithm [8, 9]. As we are considering only the MGS and GR algorithms, our focus is restricted to only these algorithms.

Numerous ASIC solutions have been published for implementing QRD [10–13]. Authors in [11, 14] describe a MIMO processing VLSI architecture on GR based QRD. Both architectures use the CORDIC algorithm. The tradeoffs between finite word length precision, which is determined by the number of CORDIC iterations, and latency has been investigated in [11]. Similarly, VLSI architectures based on MGS for QRD are presented in [6, 15, 16].

Several decomposition methods have been implemented on FPGAs as well [17–20]. The performance differences between the QRD algorithms, due to the differences in the fixed-point implementations, can be clearly seen in the mentioned publications.

QRD implementations on both floating- and fixed-point DSPs exist. The authors in [21] have analyzed the performance differences of numerical algebra algorithms between implementation in a fixed-point DSP, using TI C64x+ and a floating-point DSP, using TI C67x+. Several matrix decomposition methods like SVD, Cholesky, LU, QRD and Gauss-Jordan have been implemented on a floating-point TI 6713 DSP and analyzed with respect of computational complexity, latency and memory requirements in [22].

Most of the above works have predominantly focused on developing efficient implementations of matrix decomposition algorithms for MIMO receivers. Though algorithms have been implemented after making complexity and numerical stability comparisons, the analysis on flexibility vs. efficiency tradeoffs which is essential for SDR development is missing. Furthermore, reusability and portability investigations are missing as well. This paper tries to fill these gaps by analyzing and implementing the MGS and GR algorithms for performing MMSE-SQRD. In order to automate SDR development, we lay focus on the aspects that are important for tool assistance throughout our investigations.

IV. Algorithms

The MMSE-SQRD of a complex augmented channel matrix $\bar{\mathbf{H}}$ using the MGS [23] and sequence of GRs is presented in this section along with the computational complexity for a 2×2 MIMO system. The modifications for performing a one-time full column sorting before executing QRD which is equivalent to a full iterative sorting for a 2×2 MIMO system are highlighted.

A. MMSE-SQRD based on modified Gram-Schmidt

The MMSE-SQRD operation based on the MGS algorithm is listed in Algorithm 1. The augmented complex-valued channel matrix $\bar{\mathbf{H}}$ is assigned to $\bar{\mathbf{Q}}$ matrix, $\mathbf{R}$ is initialized with a zero matrix, $\mathbf{0}_{n_T}$ and the permutation matrix $\mathbf{P}$ with $\mathbf{I}_{n_T}$. First, the column norm vector α of each column in $\bar{\mathbf{Q}}$ is calculated (lines 1 to 3). Next, a full column sorting is performed. The column $\bar{\mathbf{q}}_i$ with the smallest squared l^2-norm is processed first. Finally, the actual QRD is done as shown from lines 7 to 14 in Algorithm 1.

The diagonal element $\mathbf{R}_{j,j}$ is calculated as shown in line 8 and is used for dividing the column $\bar{\mathbf{q}}_j$ (line 9). The upper diagonal elements in row j of $\mathbf{R}$ are computed according to $\mathbf{R}_{j,k} = \bar{\mathbf{q}}_j^H \bar{\mathbf{q}}_k$, where $k = j+1, j+2, ..., n_T$. In each sub-iteration, column $\mathbf{q}_k$ is updated according to line 12 in Algorithm 1. The MGS algorithm executes the QRD in n_T steps, with the result as shown in Equation 4.

$$\bar{\mathbf{H}} = \bar{\mathbf{Q}}\mathbf{R}\mathbf{P}^T = \begin{bmatrix}\mathbf{Q}_a \\ \mathbf{Q}_b\end{bmatrix}\mathbf{R}\mathbf{P}^T \tag{4}$$

Input: $\bar{\mathbf{Q}} = \bar{\mathbf{H}}, \mathbf{R} = \mathbf{0}_{n_T \times n_T}, \mathbf{P} = \mathbf{I}_{n_T}$
1: **for** $i = 1, 2, ...n_T$
2: $\quad \alpha_i = \|\bar{\mathbf{q}}_i\|^2$
3: **end**
4: **for** $i = 1, 2, ...n_T$
5: $\quad$ sort columns in $\bar{\mathbf{Q}}$ and $\mathbf{P}$, corresponding to norm α
6: **end**
7: **for** $j = 1, 2, ...n_T$
8: $\quad \mathbf{R}_{j,j} = \sqrt{\bar{\mathbf{q}}_j^H \bar{\mathbf{q}}_j}$
9: $\quad \bar{\mathbf{q}}_j = \frac{1}{\mathbf{R}_{j,j}} \cdot \bar{\mathbf{q}}_j$
10: $\quad$ **for** $k = j+1, j+2, ..., n_T$
11: $\quad\quad \mathbf{R}_{j,k} = \bar{\mathbf{q}}_j^H \bar{\mathbf{q}}_k$
12: $\quad\quad \bar{\mathbf{q}}_k = \bar{\mathbf{q}}_k - \mathbf{R}_{j,k}\bar{\mathbf{q}}_j$
13: $\quad$ **end**
14: **end**
Output: $\mathbf{Q}_{aj,i} = \bar{\mathbf{Q}}_{j,i}$ with $j = 1, 2, ...n_T$, $i = 1, 2, ...n_R$

Alg. 1: MMSE One-Time-Sorted QRD based on MGS

The MGS algorithm is mainly based on multiplication ($\mathbb{C}\times\mathbb{C}$ and $\mathbb{C}\times\mathbb{R}$), inverse and square root operations. Table I illustrates the count of basic operations in performing one MMSE-SQRD operation using MGS algorithm for a 2×2 MIMO system.

Operation	Count
SQRT	2
INV	2
$\mathbb{C} \times \mathbb{C}$	9
$\mathbb{R} \times \mathbb{C}$	5

TABLE I
Operation count for performing MMSE-SQRD using MGS algorithm in a 2×2 MIMO system

B. MMSE-SQRD based on Givens rotations

The MMSE-SQRD operation performed using a series of GR is listed in Algorithm 2. Initialization of $\mathbf{Z}$ is done according to Equation 5. $\mathbf{Z}$ is a compound matrix of the augmented complex-valued channel matrix $\bar{\mathbf{H}}$, an $n_T \times n_T$-dimensional identity matrix and a zero matrix in the lower right section of matrix $\mathbf{Z}$. The right side of the $\mathbf{Z}$ matrix is used for concatenation of the GRs to form the unitary $\bar{\mathbf{Q}}$ matrix. The GR algorithm upper triangularizes the left half of matrix $\mathbf{Z}$. Permutation matrix $\mathbf{P}$ keeps track of the sorting.

$$\mathbf{Z}^{(\mathbf{0})} = \begin{bmatrix}\mathbf{H} & \mathbf{I}_{n_T} \\ \sqrt{\frac{n_T}{E_s}}\sigma_n \mathbf{I}_{n_T} & \mathbf{0}\end{bmatrix} \tag{5}$$

where $\mathbf{Z}$ has the dimension of $(n_R + n_T) \times (n_T + n_R)$. The outcome of the GR algorithm after the last iteration N is

$$\mathbf{Z}^{(N)} = \begin{bmatrix}\mathbf{R} & \mathbf{Q}_a^H \\ \mathbf{0} & \mathbf{Q}_b^H\end{bmatrix} \tag{6}$$

The column norm vector α of $\bar{\mathbf{H}}$ is calculated in lines 1 to 3 of Algorithm 2. A full column sorting is executed between lines 4 and 6. The procedure for QRD is illustrated from line 7 to 10. The MMSE-SQRD operation using GR algorithm computes the $\bar{\mathbf{Q}}$ and $\mathbf{R}$ matrices in n_T steps. In each step, a series of GR is computed and multiplied with $\mathbf{Z}$ to eliminate the $j+n_R, i+n_R-1, ..., i+1$ rows of $\mathbf{z}_i$. In other words, each rotation zeros an element in the subdiagonal of the given matrix, forming an upper-triangular $\mathbf{R}$ matrix. The concatenation of all applied GR forms the unitary matrix $\bar{\mathbf{Q}}$. In each step, the rotation is only executed on two specific rows.

The complex-valued matrix processing using GR is an extension of the real-valued matrix processing and can be separated into two operations: $\mathbb{C} \rightarrow \mathbb{R}$ and $\mathbb{R} \rightarrow 0$. $\mathbb{C} \rightarrow \mathbb{R}$ operation transforms a complex matrix entry $\mathbf{Z}_{p,k}$ from the complex plane into the real plane. The rotation matrix $\theta_{\mathbb{C}}$ of $\mathbb{C} \rightarrow \mathbb{R}$ operation is given by

$$\theta_{\mathbb{C}}(p, \phi) = \begin{bmatrix}1 & 0 \\ 0 & e^{-i\phi}\end{bmatrix} \begin{matrix}p \\ p\end{matrix} \tag{7}$$

with rotation angle

$$\phi(p, k, \mathbf{Z}) = -\arctan\left(\frac{\mathrm{Im}\{\mathbf{Z}_{p,k}\}}{\mathrm{Re}\{\mathbf{Z}_{p,k}\}}\right) \tag{8}$$

$\theta_{\mathbb{C}}(p, \phi)$ is a 2-dimensional identity matrix with $e^{-i\phi}$ at

Input: $\mathbf{Z} = \mathbf{Z}^{(0)}, \mathbf{P} = \mathbf{I}_{n_T}$

for $i = 1, 2, ...n_T$
  $\alpha_i = \|\bar{\mathbf{h}}_i\|^2$
end
for $i = 1, 2, ..., n_T$
  sort the first n_T columns in the first n_R rows of $\mathbf{Z}^{(0)}$ and $\mathbf{P}$, corresponding to α
end
for $j = 1, 2, ..., n_T$
  perform a series of GRs, θ_u such that rows $j + 1, ..., n_R + n_T$ of column $\mathbf{z}_j$ become zero
  $\mathbf{Z} = (\prod_{u=(j-1)n_R+1}^{in_R} \theta_u)\mathbf{Z}$
end

Output:

$\mathbf{R}_{j,i} = \mathbf{Z}_{j,i}$ with $i, j = 1, 2, ..., n_T$
$\mathbf{Q}_{a\ j,i}^{H} = \mathbf{Z}_{j,i+n_T}$ with $j = 1, 2, ..., n_T$, $i = 1, 2, ..., n_R$

Alg. 2: MMSE One-Time-Sorted QRD based on GR

position p, p. The elimination of real valued matrix elements is performed by $\mathbb{R} \rightarrow 0$ operation. The rotation matrix for $\mathbb{R} \rightarrow 0$ operation is given by

$$\theta_{\mathbb{R}}(q, p, \phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{matrix} q \\ p \end{matrix} \qquad (9)$$
$$\qquad\quad q \qquad\quad p$$

with rotation angle

$$\phi(q, p, k, \mathbf{Z}) = -\arctan\left(\frac{\mathrm{Re}\{\mathbf{Z}_{q,k}\}}{\mathrm{Re}\{\mathbf{Z}_{p,k}\}}\right) \qquad (10)$$

Both $\mathbb{C} \rightarrow \mathbb{R}$ and $\mathbb{R} \rightarrow 0$ operations can be further separated into basic vectoring and rotation operations. The vectoring operation rotates a complex value into the real plane and provides the rotation angle. In other words, the vectoring operation performs the computation of the rotation angle (shown in Equation 8). Furthermore, it obtains the angle needed for rotating the real valued matrix elements (shown in Equation 10).

The rotation operation rotates one complex value into another by a specific angle. It is important to note that the rotation operation has to be performed, using the rotation matrices 7 and 9, on the other entries in the rows of $\mathbf{Z}$ which are affected by $\mathbb{C} \rightarrow \mathbb{R}$ and $\mathbb{R} \rightarrow 0$ operations. Table II lists the required number of these basic operations for performing MMSE-SQRD on a 2×2 MIMO system. Note that the GR algorithm can be easily parallelized, however, it needs a higher number of operations when compared to MGS algorithm for performing MMSE-SQRD.

V. Implementations

The building blocks, which have been reused for different variants of implementation, are first presented in this section followed by the variants themselves.

Basic Operation	Count	
	Vectoring	Rotation
$\mathbb{C} \rightarrow \mathbb{R}$	4	7
$\mathbb{R} \rightarrow 0$	4	16

TABLE II
Number of basic operations for performing MMSE-SQRD using GR algorithm in a 2×2 MIMO system

A. Building Blocks

1) TI IQMath Library: In order to increase the implementation efficiency and decrease the time-to-market, TI provides IQmath library [24] with implementations of well-known mathematical functions for the C64x+ DSP. This library is a collection of mathematical 32-Bit fixed-point functions. Table III lists the functions from the IQmath library that are used in our MMSE-SQRD implementations.

Algorithm	Name	Processing time in cycles
MGS	IQNsqrt	79
	IQNdiv	73
GR	IQNsin	56
	IQNcos	54
	IQNatan2	118

TABLE III
List of used functions from the IQmath library

The same functions were used for both 16-Bit and 32-Bit implementations, except inverse and square root functions. Hand written optimized implementations of these two functions, which gave a better performance when compared to the 32-Bit IQmath library functions, were used for 16-Bit implementation.

2) CORDIC: The CORDIC algorithm is versatile algorithm widely used in digital signal processing applications. Typically, it is used for performing a vector pseudo-rotation of a given two-dimensional vector in a sequence of micro iterations with discrete step size. One advantage of the CORDIC algorithm is that it can be implemented by employing, predominantly, shift and add operations.

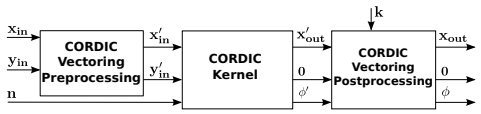

Fig. 3. Implementation of CORDIC vectoring

Figures 3 and 4 show the implementation of vectoring and rotation operations using CORDIC. $\mathbf{n}$ denotes the number of applied CORDIC iterations and $\mathbf{k}$ represents the scaling factor with which the final result is multiplied. The rotation angle is marked as ϕ.

Note that the CORDIC implementation in the rotation mode can be modified for computing sine and cosine values of a

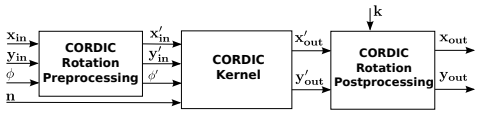

Fig. 4. Implementation of CORDIC rotation

given rotation angle ϕ, it is denoted as sincos implementation in this paper.

B. Variants of Implementation

The different variants for implementing the algorithms are presented in this section. Highly portable implementations are done in floating-point and fixed-point using generic C. Both 16-Bit and 32-Bit versions have been implemented in the fixed-point format. Highly optimized VLIW code has been hand-written in assembly suiting the DSP architecture. Since the implementations of the MMSE-SQRD algorithm using MGS algorithm are straight forward and follows exactly the same procedure as shown in Algorithm 1, it is not discussed here. MMSE-SQRD implementation using GR provides more variety in implementation. Three such variants are:

1) GR-Plain: Implementations denoted by GR-Plain use the IQmath library to execute GRs. As shown in Figure 5, implementations of the trigonometric functions (IQNatan2, IQNsin and IQNcos) from the IQmath library are used to compute the rotation angle for the GR, mentioned in Equations 8 and 10. The rotation operations are executed using complex multiplication operations.

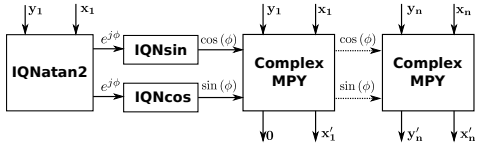

Fig. 5. Implementation of GR using the IQmath library

2) GR-CORDIC: Implementations denoted by GR-CORDIC use only the CORDIC kernel for the vectoring and rotation operations in order to perform a GR. As illustrated in Figure 6, the CORDIC implementation in vectoring operation computes also the rotation angle, which is further applied for performing rotation operations using the CORDIC kernel.

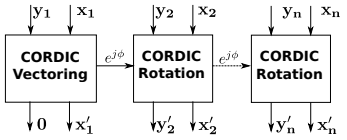

Fig. 6. Implementation of GR-CORDIC using only the CORDIC kernel

3) GR-Hybrid: In order to avoid the trigonometric operations which consume several cycles in the GR-Plain implementation (see Table III) and CORDIC rotations, which consume more cycles when compared to complex multiplications (see Tables V and VIII), the GR-Hybrid variant is implemented. The vectoring and sincos building blocks, using CORDIC kernel, are used for computing the rotation angle for GR. Fast complex multiplications are used for executing the rotation operation. Figure 7, depicts the block diagram of the hybrid implementation.

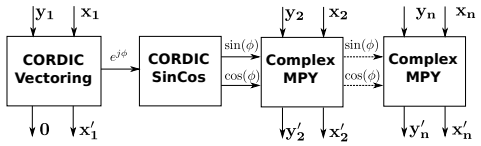

Fig. 7. Implementation of GR-Hybrid using the CORDIC kernel and complex multiplications

VI. Results

In this section, performance of the implementations for performing MMSE-SQRD in terms of BER and processing time (using cycle count) are presented.

A. BER

The floating-point MMSE-SQRD implementations are used as a reference for comparing the BER performance of fixed-point implementations. For all BER simulations, a i.i.d.-type of channel matrix is used. SNR in decibels (dB) is the ratio of the average energy per transmitted constellation symbol vector and the AWGN noise with zero mean and variance σ_n^2. The channel matrix entries are distributed according to $\mathcal{CN}(0,1)$. The 2×2 MIMO system employs two spatial separated streams with QPSK modulation using Gray mapping and non-systematic $\{171,133\}_8$ convolutional code with a code rate of $r=1/2$. Each carrier is used for data symbols. Perfect channel and noise estimation have been assumed. SIC is used as the MIMO detection scheme for all simulations.

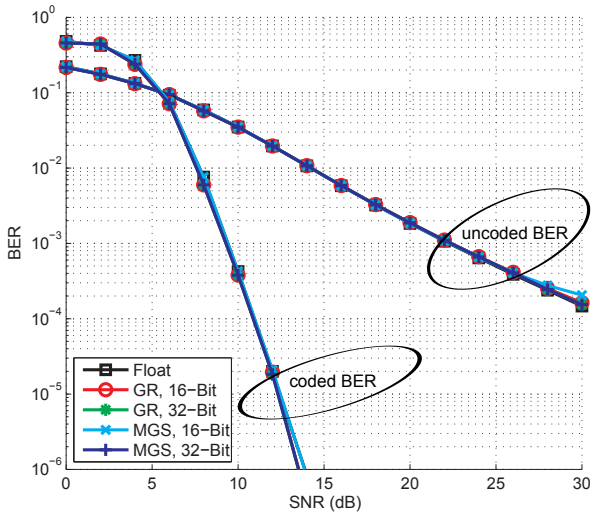

Fig. 8. Coded and uncoded BER for MGS and GR-plain implementations with 16-Bit and 32-Bit I/O data-width

Figure 8 shows the BER results of both 16-Bit and 32-Bit implementations using the MGS and GR-Plain variants.

Figure 9 illustrates the BER performance of the implementation variants using CORDIC, namely GR-CORDIC and GR-Hybrid. Note that the Q format[1] has been adjusted in the implementation to achieve closer to floating-point performance for all the variants of implementation. Therefore, performance differences are minimal between these variants. The applied Q formats for the 16-Bit and 32-Bit variants are the following, MGS: Q4.11 and Q5.26; GR-Plain: Q2.13 and Q5.26; GR-CORDIC and GR-Hybrid: Q3.12 and Q3.28.

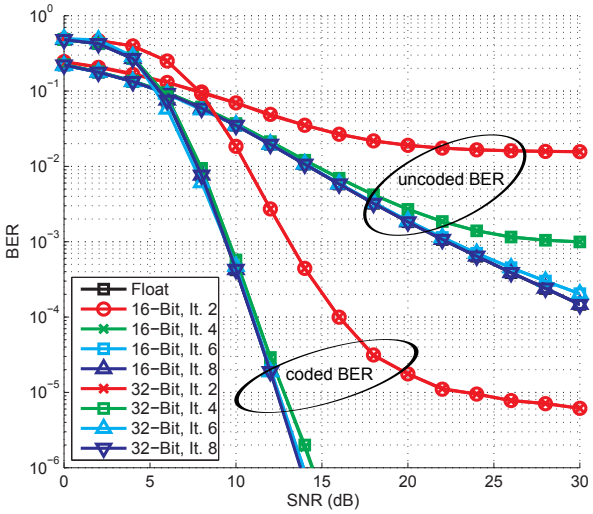

Fig. 9. Coded and uncoded BER for CORDIC based implementations with 16-Bit and 32-Bit I/O data-width for different CORDIC iterations

B. Processing Time

For all the measurements in this section, the TMS320C6000 C Compiler v6.08 with "-o3 -pm" options has been used. Program code and data are stored in L1RAM, with caches enabled. Table IV presents the processing time in terms of cycles consumed by the floating-point and fixed-point implementations of the MGS and GR-plain variants. The cycle counts for CORDIC based implementation-variants, with varying number of CORDIC iterations, are presented in Tables VI and VII. Note that only the best case cycle counts are presented in all the tables.

Implementation	Cycle count MGS	GR-Plain
Float C	11126	89536
C 32	457	1974
C 16	443	1917
ASM 32	410	1906
ASM 16	231	1762

TABLE IV
PROCESSING TIME IN CYCLES FOR PERFORMING MMSE-SQRD USING THE MGS AND GR-PLAIN IMPLEMENTATIONS

[1]Q format represents the fixed-point number format where the number of fractional bits and integer bits is specified. For example, Q2.13 represents a 16-bit number with a sign bit, 2 integer bits and 13 fractional bits.

Clock cycles taken by the CORDIC implementations have been divided into three parts: preprocessing, CORDIC kernel and postprocessing. Cycles taken by the CORDIC kernel depends on the number of CORDIC iterations. Total number of cycles for executing a CORDIC operation can be given by:

$$l_{Vec,\ Rot,\ SinCos} = l_{pre} + l_k \cdot n_{co} + l_{post} \qquad (11)$$

l_{Vec}, l_{Rot} and l_{SinCos} indicate the total number of cycles consumed for performing vectoring, rotation and sincos operations respectively (described in Section V-A2). n_{co} denotes the number of CORDIC iterations. l_{pre}, l_k and l_{post} represent the number of clock cycles for preprocessing, CORDIC kernel and postprocessing respectively. Table V outlines the number of cycles consumed for performing these operations.

	Kernel	Vectoring		Rotation		Sine Cosine	
	l_k	l_{pre}	l_{post}	l_{pre}	l_{post}	l_{pre}	l_{post}
Float C	493	372	372	348	348	348	496
C 32	8	38	52	38	52	38	52
C 16	8	25	29	24	25	24	26
ASM 32	6	14	16	18	20	18	20
ASM 16	6	12	14	16	15	16	15

TABLE V
PROCESSING TIME IN CYCLES FOR IMPLEMENTING THE BUILDING BLOCKS DESCRIBED IN SECTION V-A2, SEPARATED INTO PRE-, POST-PROCESSING AND CORDIC KERNEL

Impl.	CORDIC iterations (n_{co}) 2	4	6	8	10
Float C	45544	80764	115130	149642	184158
C 32	3243	3739	4235	4731	5227
C 16	2340	2855	3338	3830	4327
ASM 32	1759	2134	2500	2878	3247
ASM 16	1481	1855	2227	2600	2972

TABLE VI
PROCESSING TIME IN CYCLES FOR PERFORMING MMSE-SQRD USING THE GR-CORDIC IMPLEMENTATION, ($l_{GR-CORDIC}$)

Impl.	CORDIC iterations (n_{co}) 2	4	6	8	10
Float C	33677	51515	68807	86082	103353
C 32	1684	1940	2199	2452	2708
C 16	1374	1632	1886	2142	2398
ASM 32	1137	1330	1521	1718	1904
ASM 16	953	1145	1337	1529	1721

TABLE VII
PROCESSING TIME IN CYCLES FOR PERFORMING MMSE-SQRD USING THE GR-HYBRID IMPLEMENTATION, ($l_{GR-Hybrid}$)

Equations for estimating the cycle counts while using GR-CORDIC and GR-Hybrid implementations are given in Equation 12 and 13 respectively.

$$l_{GR-CORDIC} = l_{Sort} + n_{Vec} \cdot (l_{Vec}(n_{co}) + l_{oa}) + \\ + n_{Rot} \cdot (l_{Rot}(n_{co}) + l_{oa}) \quad (12)$$

$$l_{GR-Hybrid} = l_{Sort} + n_{Vec} \cdot (l_{Vec}(n_{co}) + l_{om}) + \\ + n_{SinCos} \cdot (l_{SinCos}(n_{co}) + l_{om}) + \\ + n_{CMPY} \cdot (l_{CMPY}) \quad (13)$$

l_{Sort} represents the processing cycles for performing one-time sorting (which is a full sorting in a 2×2 MIMO system). l_{Vec}, l_{Rot} and l_{SinCos} indicate the cycles consumed for vectoring, rotation and sincos operations respectively. l_{CMPY} denotes the average cycle count for performing a complex multiplication. n_{Vec}, n_{Rot}, n_{SinCos} and n_{CMPY} represent the number of vectoring, rotation, sincos and complex multiplication operations. The number of required complex multiplications is the sum of all rotations listen in Table II. The factors l_{oa} and l_{om} mark the overhead due to the address and memory & address operations respectively. Note that the cycle counts for the implementations using CORDIC depend on the number of CORDIC iterations. The processing time taken by the constant items is presented in Table VIII.

	l_{Sort}	l_{CMPY}	l_{oa}	l_{om}
32-Bit	151	6	4	7
16-Bit	50	5	1	9

TABLE VIII
PROCESSING TIME IN CYCLES FOR CONSTANTS

C. Discussion

Moving from a floating-point C implementation to a highly optimized assembly code brings the most improvement by a factor of 48 in MGS and the least improvement by a factor of 31 in GR-CORDIC implementation (for 2 CORDIC iterations). The maximum and minimum reductions in terms of cycle count when moving from a floating-point to a fixed-point implementation are noticed in GR-Plain (by a factor of 45) and GR-CORDIC (by a factor of 14) (for 2 CORDIC iterations) respectively. In fixed-point implementations, maximum and minimum difference between C 32 and ASM 16 are found in MGS (by a factor of 2) and GR-Plain respectively (by a factor of 1.1), other implementations vary between the factors, 1.6 and 1.7. Even with 4 CORDIC iterations, the minimum difference between MGS and GR based implementations is found to be a factor of 5 (while using GR-Hybrid). When we compare with GR-CORDIC, the difference rises to a factor of 8 for 4 CORDIC iterations and almost by a factor of 10 for 6 CORDIC iterations, which is a more fair comparison when considering BER performance. However, one of the key drawbacks of the SQRD implementation using MGS algorithm is the bad numerical stability, predominantly due to the division operation resulting in a high dynamic range. This results in a degradation in BER performance at higher dBs, which can be noticed in MGS 16-bit implementation (Figure 8). Moreover, the GR algorithm can be easily parallelized. This makes it attractive for VLSI implementations.

When the C64x+ DSP is operated at the maximum clock frequency of 584 MHz, 16-Bit and 32-bit assembly implementations of MGS algorithm need only 0.39 μs and 0.7 μs respectively for performing a SQRD. Among the GR based 16-Bit assembly implementations, GR-Plain, GR-CORDIC and GR-Hybrid (both with 6 CORDIC iterations) need 3 μs, 3.8 μs and 2.3 μs respectively for one SQRD operation. From the processing time, it can be clearly seen that only the fixed-point C and assembly implementations of the MGS algorithm can meet the timing constraints while considering the most demanding scenario of 1 μs for one SQRD operation in LTE. However, other 16-Bit assembly implementations become attractive for other scenarios that are less demanding, e.g. lower number of data carrying sub-carriers, lower speed of the mobile set, etc. For example, in the second scenario described for LTE in Section II-B, CORDIC based implementations can easily meet the timing constraint of 8.3 μs and can be more attractive due to the reusability.

Our results show the amount of performance that needs to be sacrificed when going for flexible implementations. Though, the MGS algorithm does not offer the same amount of flexibility in terms of implementation-variants like GR algorithms and is numerically unstable, it is very suitable for DSP architectures. On the other hand, SQRD implementation using CORDIC offers full flexibility, where CORDIC iterations can be varied depending on the need. Moreover, CORDIC algorithm can be reused for implementing several other functionalities, like shown in [3].

Our analysis has highlighted that merely increasing portability is not a solution for SDRs, particularly for new standards that demand high performance. Performance requirements and energy efficiency have to be considered by all means. However, this will decrease portability when adopting traditional approaches for SDR development. This calls for more sophisticated SDR development approaches.

Library based approaches are highly desirable for increasing reusability in system development. However, in order to increase portability and efficiency, the components of the library have to be standardized for which efficient implementations can be provided by vendors for a PE [1]. Moreover, the components of a library should be independent of wireless standards and must be reusable. We have shown that using algorithmic kernels, like CORDIC, can enhance reusability and offer more flexibility in implementations. At the same time, they can be implemented efficiently as well. Enhanced versions of the implementations using the algorithmic kernel, suiting architecture of a PE, as highlighted by GR-Hybrid implementation, can always be implemented for better performance, if needed. Still, portability effort is considerably low.

The architecture of a PE plays a huge role in reducing processing time and increasing energy efficiency. This has been highlighted by more DSP friendly MGS algorithm compared to GR. Therefore, extreme care must be taken while

"mapping" an algorithm onto a PE, particularly while using heterogeneous hardware platforms. This in turn highlights the need for quick "mapping" exploration at a high abstraction level, presumably at component level, for reducing the time needed for performing the exploration. We have derived equations that accurately estimates the cycles which could indeed can be used by tools to perform mapping exploration automatically.

VII. Conclusions and Outlook

In this paper, we have presented the analysis on flexibility vs. efficiency tradeoffs that need to be made while developing SDRs by using the computation intensive MMSE-SQRD, which is widely used in MIMO receivers, as a case study. Two algorithms for performing SQRD were investigated and efficiently implemented in several versions. Flexible versions of implementations with varying degree of reusability, portability and efficiency have been implemented. Though the dedicated implementations differ in processing time by good margin when compared to flexible implementations, efficient implementations with reusable algorithms can still provide flexibility and can be used in scenarios with tight constraints, like latency. Accurate equations which can be used for performing constraint-aware mapping at a high abstraction level with tool-assistance have been derived. As outlook, the analysis of flexibility vs. efficiency tradeoffs can be continued by implementing more computation-intensive wireless physical layer algorithms on different hardware architectures.

References

[1] V. Ramakrishnan *et al.*, "Efficient and portable SDR waveform development: The Nucleus concept," in *Proc. IEEE Military Communications Conf. MILCOM 2009*, 2009, pp. 1–7.

[2] G. Ghosh *et al.*, *Fundamentals of LTE*, 1st ed. Prentice Hall, August 2010.

[3] J. Valls *et al.*, "The use of cordic in software defined radios: a tutorial," *Communications Magazine, IEEE*, vol. 44, no. 9, pp. 46 –50, 2006.

[4] Y. Wang *et al.*, "Parallel MIMO detection algorithm based on householder transformation," in *Proc. Int. Symp. Intelligent Signal Processing and Communication Systems ISPACS 2007*, 2007, pp. 180–183.

[5] K.-L. Chung *et al.*, "The complex Householder transform," vol. 45, no. 9, pp. 2374–2376, 1997.

[6] C. K. Singh *et al.*, "VLSI Architecture for Matrix Inversion using Modified Gram-Schmidt based QR Decomposition," in *Proc. th Int VLSI Design Held jointly with 6th Int. Conf. Embedded Systems. Conf*, 2007, pp. 836–841.

[7] C. K. Singh *et al.*, "A Fixed-Point Implementation for QR Decomposition," in *Proc. IEEE Dallas/CAS Workshop Design, Applications, Integration and Software*, 2006, pp. 75–78.

[8] R.-H. Lai *et al.*, "A modified sorted-QR decomposition algorithm for parallel processing in MIMO detection," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2009*, 2009, pp. 1405–1408.

[9] D. Wubben *et al.*, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. VTC 2003-Fall Vehicular Technology Conf. 2003 IEEE 58th*, vol. 1, 2003, pp. 508–512.

[10] N. W. Gabriel L. Nazar, Christina Gimmler, "Implementation Comparisons of the QR decomposition for MIMO Detectionson," 2010.

[11] C. Studer *et al.*, "Matrix Decomposition Architecture for MIMO Systems: Design and Implementation Trade-offs," in *Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers ACSSC 2007*, P. Blosch, Ed., November 2007, pp. 1986–1990.

[12] I. LaRoche *et al.*, "An efficient regular matrix inversion circuit architecture for MIMO processing," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2006*, 2006.

[13] Z.-Y. Huang *et al.*, "High-throughput QR decomposition for MIMO detection in OFDM systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2010, pp. 1492 – 1495.

[14] P. Luethi *et al.*, "VLSI Implementation of a High-Speed Iterative Sorted MMSE QR Decomposition," in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.*, May 2007, pp. 1421–1424.

[15] K.-H. Lin *et al.*, "Implementation of QR decomposition for MIMO-OFDM detection systems," in *15th IEEE International Conference on Electronics, Circuits and Systems, 2008. ICECS 2008.*, 2008, pp. 57–60.

[16] P. Luethi *et al.*, "Gram-Schmidt-based QR decomposition for MIMO detection: VLSI implementation and comparison," in *Proc. IEEE Asia Pacific Conf. Circuits and Systems APCCAS 2008*, 2008, pp. 830–833.

[17] K. Mohammed *et al.*, "A MIMO Decoder Accelerator for Next Generation Wireless Communications," no. 99, p. 1, 2009, early Access.

[18] M. S. Khairy *et al.*, "Efficient FPGA Implementation of MIMO Decoder for Mobile WiMAX System," in *Proc. IEEE Int. Conf. Communications ICC '09*, 2009, pp. 1–5.

[19] J. Eilert *et al.*, "Efficient Complex Matrix Inversion for MIMO Software Defined Radio," in *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2007*, 2007, pp. 2610–2613.

[20] A. Irturk *et al.*, "Architectural Optimization of Decomposition Algorithms for Wireless Communication Systems," in *Proc. IEEE Wireless Communications and Networking Conf. WCNC 2009*, 2009, pp. 1–6.

[21] Z. Nikolic *et al.*, "Design and implementation of numerical linear algebra algorithms on fixed point DSPs," *EURASIP J. Adv. Signal Process*, vol. 2007, no. 2, pp. 13–13, 2007.

[22] T. Haustein *et al.*, "Real-time signal processing for multiantenna systems: algorithms, optimization, and implementation on an experimental test-bed," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 136–136, 2005.

[23] A. Bjoerck, "Numerics of Gram-Schmidt orthogonalization," *Linear Algebra and its Applications*, vol. 197-198, pp. 297 – 316, 1994.

[24] *TMS320C64x+ IQmath Library User's Guide*, Texas Instuments, December 2008.