Software Solutions for Converting a MIMO-OFDM Channel into Multiple SISO-OFDM Channels

M. Sima*, M. Senthilvelan§, D. Iancu†, J. Glossner†, M. Moudgill†, and M. Schulte§
*University of Victoria, Department of Electrical and Computer Engineering
PO Box 3055 Stn CSC, Victoria, BC V8W 3P6, Canada
Email: msima@ece.uvic.ca
†Sandbridge Technologies, Inc., 1 North Lexington Avenue, White Plains, NY 10601, U.S.A.
e-mail: {DIancu,JGlossner,MMoudgill}@sandbridgetech.com
§University of Wisconsin-Madison, Department of Electrical and Computer Engineering
1415 Engineering Drive, Madison, WI 53706, U.S.A.
e-mail: senthilv@cae.wisc.edu, schulte@engr.wisc.edu

Abstract

We present a software approach for MIMO-OFDM wireless communication technology. We first show that complex matrix operations like Singular-Value Decomposition (SVD), diagonalization, triangularization, etc., can be executed efficiently in software using a combination of CORDIC and unitary rotation algorithms in a multithreaded SIMD processor. We then investigate and analyze the transformation of a MIMO-OFDM channel into multiple independent SISO-OFDM channels by means of the SVD. The algorithms are implemented on the Sandblaster processor. The numerical results indicate that the CORDIC-augmented processor provides a significant reduction in the computing time of more than 47% over the standard Sandblaster processor when converting a 4-by-4 MIMO-OFDM channel into four SISO-OFDM channels. The technique is applicable to emerging wireless communication protocols, such as WiMAX and Wi-Fi, and provides the flexibility required to adapt to continually changing and evolving standards without the need for expensive hardware redesigns and respins.

1. Introduction

High data-rate wireless access is demanded by many applications. Since most of the available frequency bands have already been allocated to specific services, it is often impractical or very expensive to increase bandwidth to accomodate higher data-rates. Multiple-Input Multiple-Output (MIMO) systems using multiple transmit and receive antennae is an alternative solution [1]. In particular, Orthogonal Frequency Division Multiplexing (OFDM) [2] can be used in conjunction with MIMO; a MIMO-OFDM system can significantly improve the throughput of wireless communication systems [3] at no additional bandwidth expenditure.

A main obstacle in implementing MIMO-OFDM is the very high computational requirements. For this reason, software implementation within the terminal stations is very difficult to achieve; only custom hardware solutions having been proposed so far [4]. In this paper, we address the complex matrix operations specific to software implementations of MIMO-OFDM. In particular, we consider the *QR* decomposition as a primitive task in Singular-Value Decomposition (SVD) and eigenvalue decomposition, and show that the Givens rotation method is the most appropriate technique to perform *QR* decomposition on Sandbridge architectures [5], [6]. Since the Givens rotation is computationally demanding and extensively used in many complex matrix operations, it is given architectural support through CORDIC functional units and associated instructions. Our simulations, performed on the Sandbridge Software-Defined Radio (SDR) development platform, indicate that a CORDIC-augmented Sandblaster processor provides a reduction in the computing time of more than 47% over the standard Sandblaster processor when decomposing a 4-by-4 MIMO-OFDM channel into four independent SISO-OFDM channels. In terms of arithmetic precision, the CORDIC-based approach is also superior to the standard approach with more than 0.3% when calculating the singular value matrix, and more than 5% for when calculating left and right unitary matrices in 16-bit fixed-point arithmetic. Since Sandblaster is a SIMD-VLIW processor with a powerful DSP instruction set, such an improvement within its target application domain indicates that a CORDIC-augmented Sandblaster processor is promising in implementing MIMO-OFDM systems.

To summarize, since no effective pure-software solutions to implement MIMO-OFDM appear in the literature, we propose and evaluate a CORDIC-based approach that allows complex matrix operations, such as *QR* decomposition, eigenvalue decomposition, and SVD to be executed efficiently in software on a multi-threaded Single-Instruction Multiple-Data (SIMD) processor. Specifically, the paper's contributions include:

- Demonstrating that the Givens rotation method is an efficient technique for performing *QR* decomposition, eigenvalue decomposition, and SVD in software on the Sandblaster processor.
- Providing techniques to incorporate CORDIC functional units into the standard Sandblaster architecture. Specifically, full-CORDIC and semi-CORDIC operations are defined.
- Estimating the computational effort to perform complex matrix operations including *QR* decomposition, and SVD in terms of full-CORDIC and semi-CORDIC primitives.
- Estimating the performance of the CORDIC-augmented Sandblaster processor for converting a MIMO-OFDM channel into multiple SISO-OFDM channels.

The remainder of this paper is as follows: Section 2 describes the theoretical background of MIMO communication systems, complex matrix computation algorithms, and complex Givens rotations. Section 3 gives a brief overview of the Sandblaster processor. Section 4 shows that the Givens rotation is the most appropriate technique to perform *QR* decomposition on Sandbridge architectures. Section 5 describes our CORDIC architectural extensions. Section 6 discusses transforming a MIMO-OFDM channel to multiple SISO-OFDM channels in software and compares the performance of a standard Sandblaster processor and a CORDIC-augmented Sandblaster processor when performing this task. Section 7 presents our conclusions.

2. Background

To make the presentation self-contained, we address some issues related to matrix computation algorithms for MIMO communications. We also present the most important aspects of complex Givens rotations and the CORDIC algorithm.

2.1. MIMO communication systems

A multiple antenna communication system contains M transmitting and N receiving antennae. Each receiving antenna receives information from each of the transmitting antennae. Thus, there are $M \times N$ signal paths from transmitter to receiver.

Assume that $\mathbf{H}$ denotes the channel matrix. Then, the element h_{ij} is a complex number that models the fading gain between the i^{th} transmit and j^{th} receive antennae. The channel matrix is estimated through a training process. A pseudo-random sequence is assigned to each transmitting antenna. Then, each of the receiving antenna is able to separate the incoming signals through a correlative analysis. We assume that the transmitted signal is OFDM. Therefore, a certain number of carriers are pilots. Each pilot carries known complex information in a specified constellation. During propagation, the channel adds distortion to the carriers. We assume that the frequency spacing between the carriers is small enough to ensure flat frequency fading for each carrier on the per carrier allocated bandwidth. Thus, the communication channel parameters specific to one of the pilots from a specific antenna can be estimated through a complex division [2].

Assuming the transmitted signal vector is $\mathbf{X}$, and the noise vector is $\mathbf{N}$, the received signal vector $\mathbf{Y}$ is:

$$\mathbf{Y} = \mathbf{HX} + \mathbf{N} \tag{1}$$

By means of an SVD, a MIMO channel can be transformed into multiple independent SISO channels. Assume the channel matrix, $\mathbf{H}$, has the singular value decomposition $\mathbf{H} = \mathbf{USV}^H$, where $\mathbf{S}$ is an upper-diagonal matrix of singular values of $\mathbf{H}$, and $\mathbf{U}$ and $\mathbf{V}^H$ are unitary matrices of singular vectors. If the transmitted signal is multiplied by $\mathbf{V}$, and the received signal is multiplied by $\mathbf{U}^H$, then

$$\widetilde{\mathbf{Y}} = \mathbf{S}\widetilde{\mathbf{X}} + \widetilde{\mathbf{N}} \tag{2}$$

where $\widetilde{\mathbf{Y}} = \mathbf{U}^H\mathbf{Y}$, $\widetilde{\mathbf{X}} = \mathbf{VX}$, $\widetilde{\mathbf{N}} = \mathbf{U}^H\mathbf{N}$.

Equation 2 describes an equivalent model for multiple independent SISO channels. Once the channel matrix, $\mathbf{H}$, is estimated, its singular-value decomposition is calculated. Then, the matrix $\mathbf{V}$ is sent back to the transmitter, which in turn predistorts the signal. The received OFDM signals are now independent and can be decoded in parallel on multiprocessor systems, such as the Sandbridge SDR Platform [5], [6].

Consequently, it is imperative to perform SVD of complex matrices efficiently. The *QR* decomposition, which is a key step in solving these decompositions and many other linear algebra problems in real-time, is briefly presented in the next subsection.

2.2. Complex matrix computation algorithms

The *QR* decomposition of a matrix $A \in \mathbb{C}^{m\times n}$ is:

$$\mathbf{A} = \mathbf{Q}\begin{pmatrix}\mathbf{R}\\ \mathbf{0}\end{pmatrix} \tag{3}$$

where the matrix $\mathbf{Q} \in \mathbb{C}^{m\times m}$ is unitary, the matrix $\mathbf{R} \in \mathbb{C}^{n\times n}$ is upper triangular, and $\mathbf{0}$ is an $(m-n)$-by-n all-zero matrix. If the matrix $\mathbf{A}$ is non-singular (that is, square and has full rank), then the diagonal elements of $\mathbf{R}$ can be choosen real and positive. In this event, the factors $\mathbf{Q}$ and $\mathbf{R}$ are both unique [7]. If the matrix $\mathbf{A}$ is singular or is not square (for example, $m > n$), then the factorization is not unique (some of the columns of $\mathbf{Q}$ are arbitrarily introduced).

Many techniques in linear algebra rely on the *QR* decomposition. For example, the *QR* factorization provides an iterative procedure for approximating the eigenvalues of a diagonalizable non-symmetric matrix [8], [9]. Also, the *QR* algorithm is at the core of a two-sided Jacobi method for calculation the SVD of a square complex matrix [10], [11]. Equally important, once the triangular form $\mathbf{R}$ has been obtained, its diagonal elements can be chosen real [7]. Then, the SVD can also be performed starting from this triangular matrix with real diagonal elements [12].

QR decomposition is computationally expensive, requiring $\mathcal{O}(n^3)$ operations on general n by n matrices [7], [13]. To perform triangularization in a time-efficient manner, systolic arrays have been proposed; see for example [14]. This approach fails the programmability requirement and consequently it is not discussed any longer. Instead, our goal is to perform triangularization using SIMD instructions on a multi-threaded processor. In this paper, three potential methods to perform the *QR* decomposition are considered for software implementation on the Sandblaster processor: the Gram-Schmidt orthogonalization, Householder reflection, and Givens rotation methods. Since the last method is of particular importance for our study, we describe it in the next subsection.

2.3. The Complex Givens Rotation

Assume two complex numbers $\mathbf{a} = a_{\text{re}} + \mathrm{j}\, a_{\text{im}}$, and $\mathbf{b} = b_{\text{re}} + \mathrm{j}\, b_{\text{im}}$. Then, a complex Givens rotation can be described in terms of two rotation angles [15]:

$$\begin{pmatrix}\cos\theta_1 & \sin\theta_1 e^{\mathrm{j}\,\theta_2}\\ -e^{-\mathrm{j}\,\theta_2}\sin\theta_1 & \cos\theta_1\end{pmatrix}\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix} = \begin{pmatrix}\mathbf{r}\\ 0\end{pmatrix} \tag{4}$$

where the complex number $\mathbf{r} = r_{\text{re}} + \mathrm{j}\, r_{\text{im}}$. The rotation matrix can be decomposed into:

$$\begin{pmatrix}e^{\mathrm{j}\,\alpha_a} & 0\\ 0 & e^{\mathrm{j}\,\alpha_b}\end{pmatrix}\begin{pmatrix}\cos\theta_1 & \sin\theta_1\\ -\sin\theta_1 & \cos\theta_1\end{pmatrix}\begin{pmatrix}e^{-\mathrm{j}\,\alpha_a} & 0\\ 0 & e^{-\mathrm{j}\,\alpha_b}\end{pmatrix} \tag{5}$$

It is apparent that $\theta_2 = \alpha_a - \alpha_b$. If $\theta_2 = 0$, then the transformation describes a real Givens rotation:

$$\begin{pmatrix}\cos\theta_1 & \sin\theta_1\\ -\sin\theta_1 & \cos\theta_1\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix} = \begin{pmatrix}r\\ 0\end{pmatrix} \tag{6}$$

where a, b, and r are real numbers.

In connection with Equation 5, it should be noted that the last rotation by the angle α_b is in fact no rotation, since $e^{\mathrm{j}\,\alpha_b}$ is multiplied by zero. Consequently, the complex Givens rotation can also be described in terms of four angles $-\alpha_a$, $-\alpha_b$, θ_1, and α_a. Also, if the last rotation by the angle α_a is no longer performed, then the resulting vector has real components:

$$\begin{pmatrix}\cos\theta_1 & \sin\theta_1\\ -\sin\theta_1 & \cos\theta_1\end{pmatrix}\begin{pmatrix}e^{-\mathrm{j}\,\alpha_a} & 0\\ 0 & e^{-\mathrm{j}\,\alpha_b}\end{pmatrix}\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\end{pmatrix} = \begin{pmatrix}\rho\\ 0\end{pmatrix} \tag{7}$$

where the Polar representations for the complex numbers $\mathbf{a}$ and $\mathbf{b}$ are $\mathbf{a} = \rho_a e^{\mathrm{j}\,\alpha_a}$, and $\mathbf{b} = \rho_b e^{\mathrm{j}\,\alpha_b}$, respectively. It is apparent that $\rho = \sqrt{\rho_a^2 + \rho_b^2}$, and as such $\mathbf{r} = \rho e^{\mathrm{j}\,\alpha_a}$.

The Givens rotation is computationally demanding. For example, the direct evaluation of the real rotation shown in Equation 6 requires four multiplications, two additions, and a large memory for the cosine and sine tables. Also, finding the angle θ_1 translates to a sequence of multiplications, additions, and table lookup operations if Taylor series expansion is employed. While a desktop computer may support these operations, implementing Givens rotations on an embedded platform for wireless applications can be problematic. Thus, new solutions are needed.

COordinate Rotation DIgital Computer (CORDIC) [16], [17] is an iterative method for performing vector rotations, e.g., Givens rotations, by arbitrary angles using only shifts and additions, as shown in Equation 8. The CORDIC algorithm is performed in one of two modes: rotation or vectoring. In **rotation mode**, the angle accumulator, z, is initialized with the desired rotation angle. The rotation decision at each iteration is made to decrease the residual angle magnitude. In **vectoring mode**, the CORDIC unit rotates the input vector to align the resulting vector with the x axis, such that y approaches 0. The result of the vectoring operation is a rotation angle, z, and the scaled magnitude of the original vector (the x component of the result).

$$\begin{cases} x[j+1] = x[j] - \sigma[j]2^{-j}y[j] \\ y[j+1] = y[j] + \sigma[j]2^{-j}x[j] \\ z[j+1] = z[j] - \sigma[j]\arctan\left(2^{-j}\right) \\ \qquad j = j+1 \end{cases} \tag{8}$$

where for rotation mode $\sigma[j] = +1$ if $z[j] \geq 0$, otherwise is -1, and for vectoring mode $\sigma[j] = -1$ if $y[j] \geq 0$, otherwise is $+1$.

The CORDIC algorithm produces one bit of accuracy each iteration [17]. Thus, the accuracy can be adjusted at run time. It is well-known that for CORDIC algorithms, only $\log_2(n)$ additional low-order bits are necessary for intermediate values to prevent round-off errors from corrupting the final result [17]. Assume the typical 16-bit precision in OFDM demodulation. Then, the CORDIC algorithm reads in three 16-bit arguments, performs intermediate computations with 20 bits of precision per result, and produces two 16-bit final results. Assuming a 5-bit representation for the iteration counter, j, $20 \times 3 + 5 = 65$ bits are to be updated during each CORDIC iteration.

3. Overview of the Sandbridge processor

Sandbridge Tech. has designed a multithreaded processor capable of executing digital-signal processing or SIMD-vector processing, embedded control, and Java code in a single compound instruction set optimized for handset radio applications [5], [6]. The SB3011 has four Sandblaster processor cores, where each core supports eight concurrent threads. The threads simultaneously execute instructions, but only one thread may issue an instruction on a cycle boundary. Since each thread writes back its results to the register files before the next instruction from the same thread issues, true data dependencies are eliminated.

The Sandblaster processor is partitioned into three units: (1) an instruction fetch and branch unit, (2) an integer and load/store unit, and (3) a SIMD-style Vector Processing Unit (VPU). Parallel operations are performed in each of these units through the use of powerful compound instructions. The VPU includes four Vector Processing Elements (VPEs), which perform arithmetic and logic operations in SIMD fashion on 16-bit, 32-bit, and 40-bit fixed-point data types. High-speed 64-bit data busses allow each VPE to load or store 16 bits of data each cycle in SIMD fashion.

Each SIMD vector instruction has a latency of one thread cycle, which corresponds to eight pipeline stages. Four out of these eight stages are execution stages. This is a fixed-time budget and each new instruction to be implemented must comply with it.

4. *QR* decomposition of complex matrices

On general-purpose machines, where only standard operations are supported in hardware, the Givens rotation method is more computationally intensive than either the Gram-Schmidt orthogonalization or the Householder reflection methods [13]. However, when certain computing primitives are implemented in hardware, the computational complexity of different decomposition methods changes. For example, when the CORDIC algorithm is implemented using dedicated functional units and instructions, a vector rotation is performed in roughly the same amount of time as a bit-serial multiplication. Hence, when hardware and instructions for the CORDIC algorithm are provided, the Givens rotation method becomes an attractive alternative.

The most popular methods to perform *QR* decomposition are Gram-Schmidt orthogonalization, Givens rotations, and Householder reflections. We analyze these methods to determine their appropriateness for software implementation on the Sandblaster processor.

4.1. Gram-Schmidt orthogonalization

Gram-Schmidt orthogonalization constructs the columns $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m$ of the unitary matrix $\mathbf{Q}$, and the entries r_{ij} of the triangular matrix $\mathbf{R}$ by succesive orthogonalization of columns $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ of the input matrix, $\mathbf{A}$. This is an iterative process, where at each step, a vector $\mathbf{q}_j \in \langle \mathbf{a}_1, \ldots, \mathbf{a}_j \rangle$ that is orthonormal to $\mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$ is found.

Gram-Schmidt orthogonalization may produce vectors (i.e., columns of matrix $\mathbf{Q}$) that are far from orthogonal when matrix $\mathbf{A}$ is close to rank-deficient [7]. Gram-Schmidt orthogonalization also relies on division, which the Sandblaster processor does not support in its standard instruction set. Division is typically not supported in DSP processors due to its long latency and infrequent occurrence in DSP applications. Although division can be performed with CORDIC operating in the linear mode [17], a divide precision of only 16 bits instead of 32 bits can be achieved for by CORDIC on the Sandblaster processor. In conclusion, Gram-Schmidt orthogonalization is not very appealing for the Sandblaster processor.

Givens rotations and Householder reflections are numerically stable algorithms for calculating the *QR* decomposition [18]. They both provide ways to compute unitary matrices that can introduce zeros into the vectors they multiply.

4.2. Givens rotations

When Givens rotations are used to reduce a matrix to triangular form [18], the rotation angles are chosen to annihilate the subdiagonal elements one-by-one. The rotations are chosen in a particular order, such that an element reduced to zero by a rotation never becomes non-zero in a later rotation. The precise ordering of rotations is as follows. Starting with the first column, the elements in positions $(2, 1)$ to $(n, 1)$ are annihilated. Then, in the second column elements in positions $(3, 2)$ to $(n, 2)$ are annihilated, and so on. It may be readily verified that a zero introduced by one transformation is not affected by subsequent transformations.

4.3. Householder reflections

With the Householder reflections method, a succession of elementary unitary matrices, $\mathbf{Q}_k$, is applied to the left of $\mathbf{A}$ such that the resulting matrix $\mathbf{R}$ is upper-triangular. The computation is performed as:

$$\mathbf{R} = \underbrace{\mathbf{Q}_n \ldots \mathbf{Q}_2 \mathbf{Q}_1}_{\mathbf{Q}^*} \mathbf{A} \tag{9}$$

Since the product $\mathbf{Q} = \mathbf{Q}_1^* \mathbf{Q}_2^* \ldots \mathbf{Q}_n^*$ is also unitary, the product $\mathbf{QR}$ is a *QR* factorization of matrix $\mathbf{A}$. Householder [7], [18] has proposed a way to compute the unitary matries $\mathbf{Q}_k$ so that at each step k, all the elements of the k-th column, $\mathbf{x}$, that are below the main diagonal are zeroed by multiplication to the left by matrix $\mathbf{Q}_k$, such that:

$$\mathbf{Q}_k \mathbf{x} = \|\mathbf{x}\| \mathbf{e}_1 \tag{10}$$

where $\mathbf{e}_1$ is a canonical unit vector. Then, the reflector $\mathbf{Q}_k$, which is a unitary matrix of full rank, is:

$$\mathbf{Q}_k = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^*}{\mathbf{v}^*\mathbf{v}} \tag{11}$$

where $\mathbf{v} = \|\mathbf{x}\| \mathbf{e}_1 - \mathbf{x}$, as mentioned in [7].

The computational effort in the Householder method is lower than in the Givens rotations method, while both methods are numerically stable. Therefore, the Householder method is often preferred on general-purpose computing platforms. However, on the Sandblaster processor that features SIMD, VLIW, and multi-threading capabilities, there are many issues that make the Givens rotation method a better choice. Generally speaking, programs that exhibit a high level of parallelism and symmetry of computation benefit more from a parallel architecture like Sandblaster. With the Householder method, the size of the reflector $\mathbf{Q}_k$ is decremented each iteration, and this dynamic change is not well-suited for implementation on the Sandblaster processor or other highly parallel DSP processors. On the other hand, the Givens rotation method exhibits a regular computation pattern. For example, the Givens rotation method introduces only a single zero per matrix multiplication. The Givens rotation method also benefits from direct hardware support in the CORDIC-extended Sandblaster processor. Therefore, in the following sections, we investigate the use of the Givens rotation method to perform complex matrix operations.

5. CORDIC architectural extension

According to Equation 8, CORDIC is a four-element function. The coordinate, y, in vectoring mode, or angle, z, in rotation mode are *a priori* known to go to zero. The iteration counter, j, is incremented by one each cycle and its final value is known. Consequently, there is no need to write y or z and j back to the register file if all the CORDIC iterations are performed by a single instruction. However, if not all the CORDIC iterations are performed by a single instruction, then the coordinate y or angle z, and the iteration counter, j, must be saved between successive CORDIC instructions. In this case, each iteration of the CORDIC algorithm has four inputs and four outputs; x, y, z, and j.

Since instructions in current RISC and DSP architectures typically have two input operands and one output operand, defining a suitable CORDIC instruction is not a straightforward task. Even the attempt to pack the four output values, x, y, z, and j, into a single wide register fails on the Sandblaster processor, since 65 bits are required to implement 16-bit rotations with CORDIC, while the Sandblaster Processor has only 40-bit registers.

A solution for incorporating CORDIC instructions into the Sandblaster architecture is to augment the register file with an auxiliary register that is implicitly accessed when the CORDIC instructions are executed. Specifically, the CORDIC instruction

CORDIC Rs1, Rs2, Rt

uses the auxiliary register **Raux** as both a source register and a destination register. With this approach, the input arguments are read in from the source registers **Rs1** (and possibly from **Rs2** if additional configuration information is needed), and **Raux**, and the results are written back to **Rt** and **Raux**.

To preserve the existing connectivity of functional units to the register file, the auxiliary register should be a part of the CORDIC functional unit instead of the vector register file. To upload data to the auxiliary register, an additional instruction is needed. Such an instruction can be, for example, *Configure CORDIC*, **CFG_CORDIC**. Since either the coordinate y or the angle, z, and the iteration counter, j, have known final values there is no need to move information from the auxiliary register back to the register file when the last CORDIC iteration completes. Subsequently, this approach is referred to as the **full-CORDIC approach**.

To eliminate the overhead for deploying auxiliary registers, the CORDIC operation may be split into two *semi-operations*. The first semi-operation updates the operand pair (x, y), while the second semi-operation updates the operand pair (z, j). This approach is referred to as the **semi-CORDIC approach**. The disadvantage of the semi-CORDIC approach is that two operations are executed to implement a full-CORDIC operation. However, the semi-CORDIC operations map well to the existing instruction format, do not require auxiliary registers, and hence have less hardware overhead than the full-CORDIC operations.

With the full-CORDIC approach, three instructions are investigated: `CFG_CORDIC`, `ROT_CORDIC`, `VEC_CORDIC`. With the semi-CORDIC approach, four instructions are investigated: `XY_ROT_CORDIC`, `ZJ_ROT_CORDIC`, `XY_VEC_CORDIC`, `ZJ_VEC_CORDIC`. For the rotation mode, the semi-operations must be issued in the order `XY_ROT_CORDIC`, `ZJ_ROT_CORDIC`. For the vectoring mode, the semi-operations must be issued in the order `ZJ_VEC_CORDIC`, `XY_VEC_CORDIC`. The constant M denotes the number of CORDIC iterations executed per CORDIC instruction.

In the Sandblaster architecture, each instruction has a latency of one thread cycle, which corresponds to eight clock cycles. Since the latency can be at most one thread cycle, and each thread cycle has four execution stages, one implementation goal is to fit as many CORDIC iterations as possible into the four execution stages. This issue, which gives the maximum value for M, is analyzed in the next subsection.

5.1. CORDIC functional unit implementation

In this subsection, only the semi-CORDIC approach is considered, since a full-CORDIC functional unit is a straightforward implementation of Equation 8.

The semi-CORDIC functional unit operating in the circular rotation mode is shown in Figure 1. In this figure only the hardware corresponding to a single CORDIC iteration is shown. However, M instances of that circuit can be deployed, which allows M CORDIC iterations to be performed using a pair of semi-CORDIC instructions.

The semi-CORDIC instructions for rotation mode are to be executed sequentially; first XY_ROT_CORDIC and then ZJ_ROT_CORDIC. Although both semi-circuits work in parallel, only the operand pair (x, y) or (z, j), corresponding to the specific semi-instruction being executed, is updated.

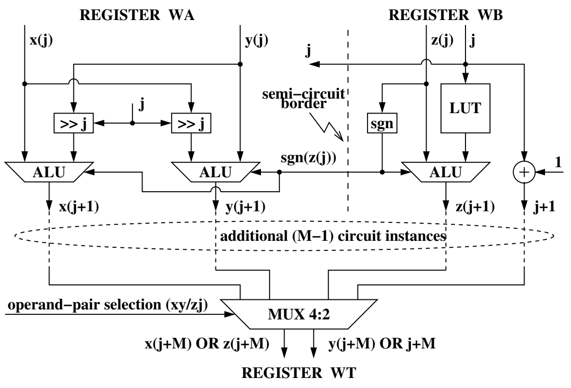

Figure 1. Semi-CORDIC unit for rotation mode

The semi-CORDIC functional unit operating in the circular vectoring mode is shown in Figure 2. Similar to the rotation case, only the hardware corresponding to a single CORDIC iteration is shown. However, M instances of that circuit can be deployed, which allows M CORDIC iterations to be performed using a pair of semi-CORDIC instructions.

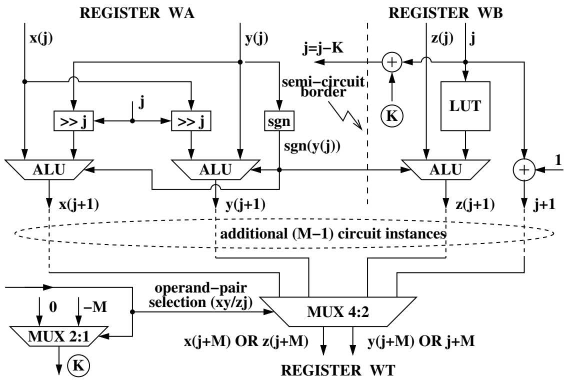

Figure 2. Semi-CORDIC for vectoring mode

The semi-CORDIC instructions for vectoring are to be executed sequentially; first ZJ_VEC_CORDIC and then XY_VEC_CORDIC. Although both semi-circuits work in parallel and only the operand pair (z, j) or (x, y), corresponding to the specific semi-instruction being executed, is updated. However, since the iteration counter, j, is incremented before the shift operations associated with the operand pair (x, y) are carried out, the iteration counter is decremented by $K = M$

Table 1. Performance figures for routines used in 4-by-4 MIMO-OFDM assuming an SVD with six sweeps.

Routine	Standard Sandblaster	full-CORDIC-augmented Sandblaster		semi-CORDIC-augmented Sandblaster	
	thread cycles	thread cycles	# CORDIC instructions	thread cycles	# CORDIC instructions
QR decomposition	243,963	116,597	1,381	118,388	1,978
SVD	3,106,339	1,594,825	16,572	1,616,317	23,736
MIMO-to-SISO	3,120,161	1,608,695	16,636	1,630,251	23,864

during the XY_VEC_CORDIC instruction. During the ZJ_VEC_CORDIC instruction, the iteration counter, j, is sent to the left semi-circuit unchanged (i.e., $K = 0$).

The proposed CORDIC instructions are vector instructions that go through eight pipeline stages, including four stages for execution. A single CORDIC iteration includes a sign detection, a shift operation, and an addition on the critical path. A conservative approach that leads to fairly inexpensive hardware is to implement four CORDIC iterations in the four pipeline stages [19], [20]. Therefore, we assume that $M = 4$ for the subsequent experiments. Thus, it takes four full-CORDIC or eight semi-CORDIC instructions to perform a real rotation with 16-bit precision. If all four vector processing elements have CORDIC functional units, four CORDIC operations can proceed in parallel.

6. MIMO-OFDM to multiple SISO-OFDM conversion in software

To perform MIMO-OFDM to SISO-OFDM conversion, the SVD of the channel matrix, $\mathbf{H}$, is calculated in fixed-point arithmetic. In this process, the CORDIC instructions are extensively utilized to perform Givens rotations. The CORDIC scale factor is eliminated by multiplying by a constant. The computational scenario at the receiver side is based on the two-sided Jacobi method [10], [11], and includes the following steps:

1) Perform *QR* decomposition on the input matrix using a complex Givens rotation for each subdiagonal element. This yields the matrices $\mathbf{Q}$ and $\mathbf{R}$. The unitary matrix, $\mathbf{Q}$, is used in building the left unitary matrix $\mathbf{U}$.
2) Transpose the upper-triangular matrix, $\mathbf{R}$, and perform the *QR* decomposition on the resulting matrix, $\mathbf{R}^{\mathrm{T}}$. This yields the matrices $\mathbf{Q}'$ and $\mathbf{R}'$. The unitary matrix, $\mathbf{Q}'$, is used in building the right unitary matrix $\mathbf{V}$.
3) Transpose the upper-triangular matrix, $\mathbf{R}'$, and reiterate Steps 1 and 2 for multiple sweeps using the matrix $\mathbf{R}'^{T}$ as input until a diagonal matrix is reached.
4) With the unitary matrices $\mathbf{Q}$ and $\mathbf{Q}'$, build the left and right unitary matrices, $\mathbf{U}$ and $\mathbf{V}$, respectively, with CORDIC unit operating in the circular rotation mode.
5) Multiply the decoded signal by $\mathbf{V}^{\mathrm{H}}$ and divide each entry by the corresponding singular value. The divisions are implemented by CORDIC operating in linear mode.

The simulations have been carried out on the Sandbridge Technologies SDR development platform using the SB3011 digital-signal processor. A 256-carrier OFDM communication system is considered, with 192 BPSK modulated active carriers. The symbol duration is 40 microseconds, and 50 symbols are grouped in 2 msec frames. The first two symbols are preambles and carry the pseudo noise training information. The remaining 48 symbols carry the user data. The channel correction algorithms are executed once every 10 msec to support urban mobility. The performance analysis has been carried out assuming a 4-by-4 MIMO configuration, which is consistent with existing communication systems. The performance figures needed to perform *QR* decomposition, SVD, and MIMO-to-SISO conversion are presented in Table 1.

For all the considered routines, it is apparent that both the full-CORDIC-augmented Sandblaster and the semi-CORDIC-augmented Sandblaster provide a significant speed-up over standard Sandblaster. For example, to complete SVD using six sweeps, the number of thread cycles (or instructions) on a full-CORDIC-augmented Sandblaster and a semi-CORDIC-augmented Sandblaster were determined to be 1,594,825 and 1,616,317, respectively. Since 3,106,339 thread cycles are needed to complete SVD on the standard Sandblaster using multiplications, divisions, standard Taylor serier expansions, and table look-up operations, the CORDIC solution provides a reduction in the computing time of more than 47% for both the full-CORDIC and semi-CORDIC approaches. This means that the cheaper semi-CORDIC approach is indeed a very good choice. The same improvement figures also apply for the MIMO-to-SISO conversion.

In terms of arithmetic precision, the CORDIC-based approach is also superior to the standard approach. The relative errors against an ideal floating-point implementation assuming that a six-sweep SVD is performed in fixed-point arithmetic in either standard or CORDIC-based approach are presented in Table 2. It is apparent that the singular value matrix, $\mathbf{S}$, and the two unitary matrices, $\mathbf{U}$ and $\mathbf{V}$ can be calculated with very good precision using CORDIC algorithm.

Table 2. SVD implementation error: 16-bit fixed-point versus floating-point.

Standard approach			CORDIC approach		
S	**U**	**V**	**S**	**U**	**V**
0.90%	5.99%	5.52%	0.57%	0.01%	0.07%

Assuming that the channel decorrelation is executed 100 times per second, the total number of thread cycles required to convert the MIMO-OFDM channel into indepenent SISO channels is 1,608,695 $\times$ 100 = 160,869,500 for the full-CORDIC approach, and 1,630,251 $\times$ 100 = 163,025,100 for the semi-CORDIC approach. Since a thread runs at 75 MHz, three threads of a SB3011 processor are needed to perform MIMO-to-SISO conversion, which means a processor occupancy of less than 10%. The remaining threads are used to implement the communication protocol.

7. Conclusion

When hardware and instructions for the CORDIC algorithm are provided, the Givens rotation method becomes an attractive method to perform *QR* decomposition and SVD in software. The CORDIC-augmented Sandblaster processor exhibits a significant speed-up and provides better arithmetic precision over the standard Sandblaster processor when decomposing a MIMO-OFDM channel into multiple independent SISO channels. Since Sandblaster is a SIMD-VLIW processor with a powerful DSP instruction set, such an improvement within its target application domain indicates that a CORDIC-augmented Sandbridge is promising in implementing MIMO-OFDM systems.

References

[1] D. Gesbert et al., "From Theory to Practice: an Overview of MIMO Space-Time Coded Wireless Systems," *IEEE J. on Selected Areas in Comm.*, vol. 21, no. 3, pp. 281–302, Apr. 2003.

[2] R. D. van Nee and R. Prasad, Eds., *OFDM for Wireless Multimedia Comm.*, Artech House Publishers, 2000.

[3] H. Bölcskei, *Principles of MIMO-OFDM wireless systems*, 2004. [Online]. Available: http://www.nari.ee.ethz.ch/commth/pubs/p/crc03

[4] D. Perels et al., "ASIC Implementation of a MIMO-OFDM Transceiver for 192 Mbps WLANs," *European Solid-State Circuits Conf. (ESSCIRC 05)*, pp. 215–218, Sept. 2005.

[5] J. Glossner, E. Hokenek, and M. Moudgill, "Multi-threaded Processor for Software Defined Radio," in *Proc. the 2002 Software Defined Radio Technical Conf.*, vol. I, San Diego, California, Nov. 2002, pp. 195–199.

[6] J. Glossner et al., "Sandblaster Low-Power Multi-threaded SDR Baseband Processor," in *Proc. Third Workshop Applications Specific Processors (WASP'04)*, Stockholm, Sweden, Sept. 2004, pp. 53–58.

[7] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Soc. for Industrial and Applied Math. (SIAM), 1997.

[8] V. N. Kublanovskaya, "On some algorithms for the solution of the complete eigenvalue problem," *USSR Comput. Math. and Math. Physics*, no. 3, pp. 637–657, Mar. 1961.

[9] J. Francis, "The QR transformation – Part 1/2," *The Computer Journal*, vol. 4, no. 3/4, pp. 265–272/332–345, Oct. 1961/Jan. 1962.

[10] E. Kogbetliantz, "Solution of Linear Equations by Diagonalization of Coefficients Matrix," *Quarterly of Applied Math.*, vol. XIII, no. 2, pp. 123–132, 1955.

[11] G. Forsythe and P. Henrici, "The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix," *Trans. of the Am. Math. Soc.*, vol. 94, no. 1, pp. 1–23, Jan. 1960.

[12] J. Charlier, M. Vanbegin, and P. V. Dooren, "On Efficient Implementations on Kogbeliantz's Algorithm for Computing the Singular Value Decomposition," *Numerische Mathematik*, vol. 52, no. 3, pp. 279–300, May 1988.

[13] C. Meyer, *Matrix Analysis and Applied Linear Algebra*, Soc. for Industrial and Applied Math. (SIAM), 2000.

[14] M. W. Gentleman and H.-T. Kung, "Matrix triangularization by systolic arrays," in *Proc. Int'l Soc. for Optical Engineering (SPIE), Real-Time Signal Processing IV*, vol. 298, Jan. 1981, pp. 19–26.

[15] T. F. Coleman and C. F. van Loan, *Handbook for Matrix Computations*, Soc. for Industrial and Applied Math. (SIAM), 1988.

[16] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sept. 1959.

[17] J. Walther, "A unified algorithm for elementary functions," in *Proc. Spring Joint Computer Conf. of the Am. Fed. of Information Processing Societies (AFIPS)*, vol. 38, Arlington, Virginia, 1971, pp. 379–385.

[18] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1996.

[19] D. Patterson and J. Hennessy, *Computer Architecture. A Quantitative Approach*, Morgan Kaufmann, 1996.

[20] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.