Efficient Complex Matrix Transformations with CORDIC

Nariankadu D. Hemkumar Joseph R. Cavallaro

Department of Electrical & Computer Engineering,
Rice University, Houston, TX 77251

Abstract

Transformations of real and arbitrary 2×2 *matrices are employed in parallel algorithms based on Jacobi-like procedures for matrix factorizations like the eigenvalue and the singular value decompositions. Cast in the primitives afforded by the CORDIC algorithms, significant speedup may be achieved in the performance of special-purpose processor array architectures. In this paper, we discuss the use of CORDIC for unitary two-sided* 2×2 *matrix transformations. We emphasize integration of evaluation of parameters with application of transformations, using only the primitives afforded by CORDIC. Implementation alternatives are presented in both non-redundant and the redundant and on-line approaches to CORDIC.*

1 Introduction

Several real-time signal processing applications require fast computation of matrix factorizations like the eigenvalue and the singular value decomposition. A variety of parallel architectures and algorithms proposed for the computation of these factorizations use Jacobi-like procedures which are amenable to parallel implementation [1].Central to these Jacobi-like procedures is the transformation of 2×2 submatrices of the input data matrix. COordinate Rotation Digital Computer (CORDIC) [2] algorithms, intended for the computation of inverse tangents and vector rotations, afford the required primitives. Cavallaro and Luk [3], Delosme [4], and Yang and Böhme [5] have demonstrated the use of CORDIC for real 2×2 matrix transformations. Recently, redundant and on-line CORDIC based approaches have been suggested by Ercegovac and Lang [6] and Lee and Lang [7].

Arbitrary data matrices occur in time-frequency signal analysis, adaptive detection and high-resolution spectral estimation. The use of CORDIC for complex arithmetic was addressed by Hitotumatu [8]. Earlier work in applying CORDIC to complex matrix transformations is due to Deprettere and van der Veen [9] and Cavallaro and Elster [10]. In [11], Delosme suggested the use of implicit CORDIC (rotation angles are not explicitly computed) for two-sided unitary matrix transformations as required in the computation of the SVD of arbitrary matrices.

In this paper, we present a two-sided unitary matrix transformation structured to facilitate integrated evaluation and application through the use of CORDIC primitives. This two-sided transformation is a generalization of previous approaches and may used as an atomic step in parallel SVD and eigenvalue decomposition arrays implementing Jacobi-like algorithms. We discuss implementation alternatives in conventional (non-redundant) CORDIC and the redundant and on-line enhancements to CORDIC.

2 Jacobi-like Matrix Algorithms

Jacobi-like algorithms iteratively converge to the desired factorization (SVD, eigenvalue) of a given matrix A, through the application of suitable two-sided unitary (orthogonal for real matrices) matrix transformations to 2×2 sub-matrices of the input matrix as

$$A_0 = A, \quad A_{k+1} = U_k A_k V_k \quad (k = 0, 1, 2, \cdots),$$

where the unitary matrices U_k and V_k are chosen appropriately [12, 13]. Typically, two-sided transformations are employed for diagonalization or triangularization of input matrices. A two-sided transformation yielding a diagonalization may be expressed as

$$\mathcal{U} \begin{bmatrix} R_a e^{i\theta_a} & R_b e^{i\theta_b} \\ R_c e^{i\theta_c} & R_d e^{i\theta_d} \end{bmatrix} \mathcal{V} \Rightarrow \begin{bmatrix} R'_a e^{i\theta'_a} & 0 \\ 0 & R'_d e^{i\theta'_d} \end{bmatrix} \tag{1}$$

where

$$\mathcal{U} = f(\phi, \theta_\alpha, \theta_\beta, \theta_\gamma, \theta_\delta) = \begin{bmatrix} \cos\phi e^{i\theta_\alpha} & -\sin\phi e^{i\theta_\beta} \\ \sin\phi e^{i\theta_\gamma} & \cos\phi e^{i\theta_\delta} \end{bmatrix} \tag{2}$$

and $\mathcal{V}^T = f(\psi, \theta_\xi, \theta_\eta, \theta_\zeta, \theta_\omega)$.

The diagonalizing two-sided transformation (1), is central to parallel Jacobi-like algorithms for computing the SVD/eigenvalues of arbitrary/Hermitian matrices. However, for the SVD of arbitrary matrices, it may not always be possible to satisfy convergence constraints based on the parameters governing the transformation matrices while ensuring diagonalization [12]. An approach to guaranteed convergence is the completion of diagonalization in two steps, each of which requires a two-sided transformation like (2) under the relaxing constraints $\theta_\alpha = \theta_\gamma, \theta_\beta = \theta_\delta, \theta_\xi = \theta_\eta$ and $\theta_\omega = \theta_\zeta$ [12]. The two-step diagonalization may be implemented in a pipelined manner by changing the systolic schedule on the array given in [1] to decrease the overall computation time [14].

1063-6889/93 $03.00 © 1993 IEEE

2.1 $\mathcal{Q}$ Transformations

If $\theta_\alpha = \theta_\gamma, \theta_\beta = \theta_\delta, \theta_\xi = \theta_\eta$ and $\theta_\omega = \theta_\zeta$ in (2), then $\mathcal{U}$ and $\mathcal{V}$ may be factorized as

$$\mathcal{U} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix},$$

$$\mathcal{V} = \begin{bmatrix} e^{i\theta_\xi} & 0 \\ 0 & e^{i\theta_\omega} \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix}. \qquad (3)$$

Definition 1 *A Q transformation is a two-sided unitary 2×2 matrix transformation, where $\mathcal{U}$ and $\mathcal{V}$, the left and right matrices, respectively, are given by (3).*

We denote the factors as inner and outer matrices with the obvious connotation, and the resulting transformations as the inner and outer transformations (different from inner rotations *i.e* outer transformation matrices with $-\pi/2 \leq \phi, \psi \leq \pi/2$ and outer rotations where $\pi/2 \leq \phi, \psi \leq 3\pi/2$).

The versatility of the $\mathcal{Q}$ transformation lies in the judicious use of combinations of inner transformations to affect the arguments of the complex data elements while using the outer rotational transformation to achieve the desired final effect. Table 1 shows some choices for parameters of the inner transformation, $\theta_\alpha, \theta_\beta, \theta_\xi$ and θ_ω. The $\mathcal{I}$ inner transformations cause the interchange of arguments along either diagonal while the $\mathcal{R}, \mathcal{C}$ and $\mathcal{D}$ inner transformations, respectively, render the data elements along either row, column and diagonal, real.

Table 1: Inner Transformations

Type	Values
$\mathcal{I}_{main}$	$\theta_\alpha = \theta_\xi = -\theta_\beta = -\theta_\omega = (\theta_d - \theta_a)/2$
$\mathcal{I}_{off}$	$\theta_\alpha = -\theta_\xi = -\theta_\beta = \theta_\omega = (\theta_c - \theta_b)/2$
$\mathcal{R}_{upper}$	$\theta_\alpha, \theta_\beta = -(\theta_b + \theta_a)/2; \theta_\xi, -\theta_\omega = (\theta_b - \theta_a)/2$
$\mathcal{R}_{lower}$	$\theta_\alpha, \theta_\beta = -(\theta_d + \theta_c)/2; \theta_\xi, -\theta_\omega = (\theta_d - \theta_c)/2$
$\mathcal{C}_{left}$	$\theta_\alpha, -\theta_\beta = (\theta_c - \theta_a)/2; \theta_\xi, \theta_\omega = -(\theta_c + \theta_a)/2$
$\mathcal{C}_{right}$	$\theta_\alpha, -\theta_\beta = (\theta_d - \theta_b)/2; \theta_\xi, \theta_\omega = -(\theta_d + \theta_b)/2$
$\mathcal{D}_{main}$	$\theta_\alpha, \theta_\beta = -(\theta_d + \theta_a)/2; \theta_\xi, -\theta_\omega = (\theta_d - \theta_a)/2$
$\mathcal{D}_{off}$	$\theta_\alpha, \theta_\beta = -(\theta_b + \theta_c)/2; \theta_\xi, -\theta_\omega = (\theta_b - \theta_c)/2$

Lemma 1 *Applying some combination of inner transformations is equivalent to applying a single inner transformation whose parameters are a sum of the corresponding parameters of the transformations used, when the parameters of individual transformations are determined using the matrix resulting from application of the previous transformations.*

In Table 2 we detail the parameters (of $\mathcal{Q}$ transformations) for different input matrices and desired transformations. The Type I $\mathcal{Q}$ transformation (upper) triangularizes an arbitrary 2×2 matrix using an $\mathcal{R}_{lower}$ inner transformation and a Givens rotation [13] on the right outer transformation. The Type II transformation diagonalizes upper triangular matrices resulting from the Type I transformation, using a combination of $\mathcal{D}_{main}$ and $\mathcal{I}_{off}$ inner transformations in that order, together with an outer two-sided diagonalization of a real 2×2 matrix (See Appendix A). The Type III $\mathcal{Q}$ transformation, which consists of an inner transformation combining an $\mathcal{R}_{upper}$ and a $\mathcal{C}_{right}$ transformation, with an outer two-sided diagonalization as in the Type II $\mathcal{Q}$ transformation, may be used to diagonalize any 2×2 Hermitian matrix.

Theorem 1 *Given an arbitrary 2×2 matrix, at most two Q transformations are needed for diagonalization. If the 2×2 matrix is Hermitian, then only one such transformation is necessary.*

Proof. (cf. [12]) Given an arbitrary 2×2 matrix choose a Type I $\mathcal{Q}$ transformation followed by a Type II $\mathcal{Q}$ transformation for diagonalization. A Type III $\mathcal{Q}$ transformation may be used to diagonalize any 2×2 Hermitian matrix. $\square$

3 CORDIC for Q Transformations

CORDIC (COordinate Rotation DIgital Computer) algorithms systematically approximate the value of certain trigonometric functions or their inverse using only a fixed sequence of additions or subtractions, and shifts. The algorithm was later unified for elementary functions by Walther [2]. CORDIC is based upon defining a vector (x_0, y_0) and applying a rotational transformation. The rotation angles are decomposed into a sequence of n known angles, such that $\theta = \pm\theta_0 \cdots \pm\theta_{n-1} = \sum_{i=0}^{n-1} \sigma_i\theta_i$, where $\theta_i = \tan^{-1} 2^{-i}$ and $\sigma_i = \pm 1$. The CORDIC recurrence relations are

$$\begin{aligned} x_{i+1} &= x_i + \sigma_i y_i 2^{-i} \\ y_{i+1} &= y_i - \sigma_i x_i 2^{-i} \\ z_{i+1} &= z_i + \sigma_i \theta_i. \end{aligned} \qquad (4)$$

CORDIC may be used to rotate (x_0, y_0) through a given angle z_0 (σ_i is chosen to drive z to zero), or compute the inverse tangent of (y_0/x_0) (σ_i is chosen to drive y to zero and $z_n = \tan^{-1}(y_0/x_0)$). Due to the nature of the recurrence relations (4), for data of n bit wordlength, no more than n iterations need be performed. Also, the final values of the variables x and y need to be corrected for an accumulated scale factor

$$K_n = \prod_{i=0}^{n-1} k_i = \prod_{i=0}^{n-1} \frac{1}{\cos\theta_i} = \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}}.$$

3.1 CORDIC and Complex Arithmetic

Walther [2] showed that the domain of convergence of CORDIC is limited by $\theta_{max} = \sum_{i=0}^{n-1} \theta_i$. For a nonrepetitive sequence of angles θ_{max} sums to ≈ 1.73 radians (99°). Once the angle θ satisfies $|\theta| > \theta_{max}$, the CORDIC algorithms no longer converge. It is necessary that the range of input variables be restricted so that the angles remain within the domain of convergence. Since the argument of a complex number has a range $[-\pi, \pi]$, it is necessary, either to extend the range of convergence or use a modified representation for the complex data element.

Table 2: Parameters for $\mathcal{Q}$ transformations

Type	Inner Parameters	Outer Parameters
I	$\theta_\alpha = \theta_\beta = -(\theta_d + \theta_c)/2, \theta_\xi = -\theta_\omega = (\theta_d - \theta_c)/2$	$\phi = 0, \psi = \tan^{-1}(R_c/R_d)$
II	$\theta_\alpha = -(\theta_a + \theta_b)/2, \theta_\beta = \theta_\xi = -\theta_\omega = (\theta_b - \theta_a)/2$	$\tan(\phi \pm \psi) = R_b/(R_d \mp R_a)$
III	$\theta_\alpha = -\theta_\beta = -\theta_\xi = \theta_\omega = -\theta_b/2$	$\psi = \phi = \tan^{-1}(2R_b/(R_d - R_a))/2$

3.1.1 Modified Polar Representation

A complex number, $z = z_r + iz_i$ may be written as $z = R_z e^{i\theta_z}$, where

$$R_z = \text{sign}(z_r)\sqrt{z_r{}^2 + z_i{}^2}, \theta_z = \tan^{-1}(z_i/z_r), \quad (5)$$

with $-\pi/2 \leq \theta_z \leq \pi/2$. This is a modified polar coordinate representation of a complex number where a sign is ascribed to the modulus and the range of the argument is restricted to the principal values of *arctangent* [8]. Note that the sign of the modulus is positive in the right half-plane and negative in the left half-plane. This information may be encoded by an additional bit representing the half-plane of the argument and the argument may be defined by the tuple (H, θ), where H is 0 (right half-plane) or 1 (left half-plane) and $-\pi/2 \leq \theta \leq \pi/2$ (See Figure 1 for angle format).

Modified Polar Representation Angle Format

H	$-90°$	$45°$	$\cdots$	$\frac{180°}{2^L}$
	1	2		L
	MSB			LSB

Extended Range Angle Format (Daggett's)

$-180°$	$90°$	$45°$	$\cdots$	$\frac{360°}{2^K}$
1	2	3		K
MSB				LSB

Figure 1: Angle Formats for the Complex Argument

3.1.2 Extended Range CORDIC

In [15], Hu *et al.*, show how the range of convergence of the CORDIC algorithm may be expanded. Using an angle format (Figure 1) suggested by Daggett [16] and an extended angle sequence that requires two additional iterations with indices of zero, the range of the CORDIC algorithms in the circular mode is extended to $\theta_{max} \approx 3.3$ radians (189.9°).

Either of the above methods may be employed to accurately represent the arguments of complex data elements. The advantage of using the modified polar coordinate representation is that no additional CORDIC iterations are necessary. However, the addition of arguments is not straightforward. A set of pre-processing rules is needed to handle the addition of complex arguments. The rules are tabulated in Table 3, where H is the bit-value representing the half-plane of the argument, and θ is the value of the argument restricted to the closed right half-plane. To handle subtraction of arguments, a unary negation is also defined. The sign of the modulus after argument addition depends on H (if $H = 1$, then the sign is negative) since the modified polar coordinate representation (5) allows for negative moduli. On the other hand, using the extended range method simplifies argument addition, since the Daggett representation makes use of the wrap-around property of two's complement arithmetic and the argument is in the range $[-\pi, \pi]$. However, additional CORDIC iterations are necessary and the scale factor is different $(K_e = 2K_n)$.

Table 3: Addition of Arguments in Modified Polar Coordinate Representation

Unary Negation	Addition
$-(H,\theta) = (H,-\theta)$	$(H_{sum}, \theta_{sum}) = \begin{cases} (H_1,\theta_1) \\ + \\ (H_2,\theta_2) \end{cases}$ $\theta_{sum} = \theta_1 + \theta_2$ if (No overflow in θ_{sum}) $H_{sum} = H_1 \oplus H_2$ else $H_{sum} = \overline{H_1 \oplus H_2}$

3.2 Implementation of $\mathcal{Q}$ transformations

The computational primitives afforded by the CORDIC algorithms in the circular mode are vector rotations and rectangular-to-polar-coordinate conversions (inverse tangent computations). The structuring of $\mathcal{Q}$ transformations facilitates the use of these very primitives. We now show how evaluation (of parameters for the inner and outer transformations) and application, is integrated through the use of CORDIC.

3.2.1 Implementing Inner Transformations

We assume that the complex data elements of the input matrix are represented in rectangular coordinates. Since the inner transformation affects only the arguments of the complex data elements, it is convenient to first compute the polar coordinate representations of the data elements. The parameters for inner transformations may then be determined as they depend only on the arguments of the complex data elements (Tables 1,2). The inner transformation is accomplished by modifying the arguments of the data elements using the appropriate parameters. A transformation back to rectangular coordinates completes the inner transformation. Two CORDIC operations are required per data element to evaluate the parameters and apply the inner transformation.

Given the parameters, inner transformations may be applied using one CORDIC operation per data element by treating each data element as a vector in the complex plane and rotating it by an angle corresponding to the sum of the appropriate left and right inner transformation parameters. Figure 2 shows the steps in the integrated evaluation (of parameters) and application of inner transformations and Figure 3 outlines the application given the parameters.

In the particular cases of the inner transformations in Types I, II and III Q transformations, integrated evaluation and application may be completed in fewer steps. Note that the inner transformation in each case renders the arguments of a pair of complex data elements zero, and is equivalent to vectoring (y-reduction) the respective data. Thus, one CORDIC operation per affected data element is needed to apply the inner transformations.

```
input ← $\begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix}$
begin
1:  par do /* rect. → polar representation */
            $x_r + ix_i \rightarrow K_n X e^{i\theta_x}, \ \forall x \in \{a, b, c, d\}$
    end
2:  par do /* compute inner parameters*/
            $\{\theta_\alpha, \theta_\beta, \theta_\xi, \theta_\omega\} = f(\theta_a, \theta_b, \theta_c, \theta_d)$
    end
3:  par do /* add appropriate left and right inner
            parameters to arguments and compute
            polar → rect. representation */
            $K_n X e^{i(\theta_x + \theta^l_{in} + \theta^r_{in})} \rightarrow K_n^2 (x'_r + ix'_i)$
    end
4:  par do /* two-sided scale factor correction ($K_n^2$) */
            $K_n^2 \begin{bmatrix} x'_r \\ x'_i \end{bmatrix} \rightarrow \begin{bmatrix} x'_r \\ x'_i \end{bmatrix}$
    end
end
output → $\theta_\alpha, \ \theta_\beta, \ \theta_\xi, \ \theta_\omega, \ \begin{bmatrix} a'_r + ia'_i & b'_r + ib'_i \\ c'_r + ic'_i & d'_r + id'_i \end{bmatrix} =$

      $\begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_ii & d_r + id_i \end{bmatrix} \begin{bmatrix} e^{i\theta_\xi} & 0 \\ 0 & e^{i\theta_\omega} \end{bmatrix}.$
```

Figure 2: Inner Transformation - Evaluate & Apply

Additionally, in the case of the Type I Q transformation, the arguments of the elements in the corresponding column are reduced by the arguments of the affected elements. Hence, two pairs of data elements may be identified, where an element of the pair is rendered real and the other is rotated in the complex plane by the negative of the argument of the former element. Vectoring the elements in the lower row of the columns and using the negative of the resulting sequence of σs (cf. [7] - rotation angle in decomposed form) to drive the rotation of the data elements in the upper row, accomplishes the inner transformation

```
input ← $\begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix}, \ \theta_\alpha, \ \theta_\beta, \ \theta_\xi, \ \theta_\omega$
begin
1:  par do /* rotate data element in the complex plane
            by the sum of the appropriate left and
            right inner parameters */
            $R(\theta^l_{in} + \theta^r_{in}) \begin{bmatrix} x_r \\ x_i \end{bmatrix} \rightarrow K_n \begin{bmatrix} x'_r \\ x'_i \end{bmatrix},$
            $\forall x \in \{a, b, c, d\}$
    end
2:  par do /* scale factor correction ($K_n$) */
            $K_n \begin{bmatrix} x'_r \\ x'_i \end{bmatrix} \rightarrow \begin{bmatrix} x'_r \\ x'_i \end{bmatrix}$
    end
end
output → $\begin{bmatrix} a'_r + ia'_i & b'_r + ib'_i \\ c'_r + ic'_i & d'_r + id'_i \end{bmatrix} =$

      $\begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix} \begin{bmatrix} e^{i\theta_\xi} & 0 \\ 0 & e^{i\theta_\omega} \end{bmatrix}.$
```

Figure 3: Inner Transformation - Apply

using two pairs of tandem CORDIC operations (See Figure 4). Also, in all three cases (Types I, II & III), the inner parameters may be evaluated with an extra addition/subtraction-shift step following the application since they depend only on the arguments of the affected data elements which should now be available in the CORDIC z-registers.

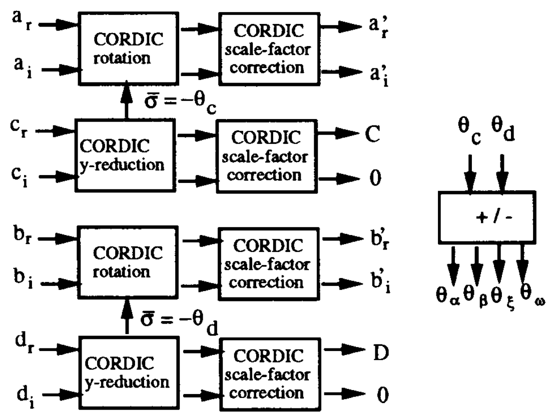

Figure 4: Type I Inner - Evaluate & Apply

3.2.2 Implementing Outer Transformations

The outer transformation is essentially a two-sided rotation. Given the parameters (ϕ, ψ), there are two approaches outlined in the literature for application of two-sided rotations using CORDIC. From the arithmetic of complex numbers, it is easy to observe that the outer transformation may be applied independently to the real and imaginary parts.

In the sequential approach [3], columns and rows of the 2 × 2 matrix are treated as vectors for the left and right rotations, respectively, and a total of eight CORDIC operations (four each for the real and the imaginary parts) are required for application. Another scheme due to Yang and Böhme ([5], cf. Appendix A), requires only four vector rotations (two each for the real and the imaginary parts) along with additional arithmetic and scaling operations. Figure 5 depicts both approaches.

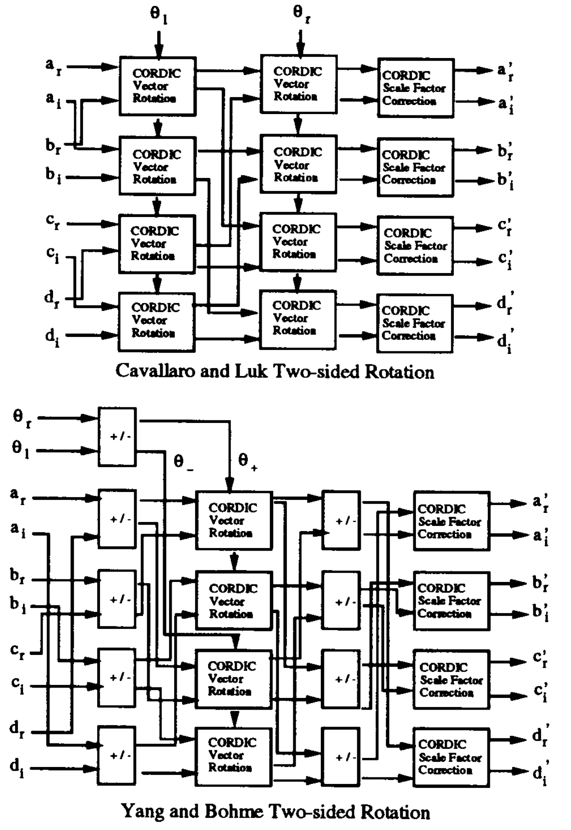
θl
θr
ar
ai
br
bi
cr
ci
dr
di
CORDIC Vector Rotation
CORDIC Scale Factor Correction
a'r
a'i
b'r
b'i
c'r
c'i
d'r
d'i
Cavallaro and Luk Two-sided Rotation
+/-
θ-
θ+
Yang and Bohme Two-sided Rotation

Figure 5: Outer Transformation - Apply

Unlike the inner transformation, we cannot generalize the evaluation of parameters for the outer transformations. However, since diagonalization or triangularization is usually the aim of two-sided unitary transformations, and the particular Q transformations listed in this paper typify both kinds, we restrict our discussion to the integrated evaluation and application of the respective outer transformations. Diagonalization is required in Types II and III Q transformations and the Type I Q transformation triangularizes the 2 × 2 matrix. Also, it may be seen from Table 2 that the Q transformations of Types II and III result in a real 2 × 2 matrix after the respective inner transformations are completed (the lower row is made real in the case of the Type I Q transformation).

In the literature, there are two methods available to compute the parameters required for the diagonalization of a real 2 × 2 matrix. Appendix A details the direct two-angle method for the diagonalization of real 2 × 2 matrices and the scheme due to Yang and Böhme [5]. A disadvantage of the direct two-angle method for diagonalization is that evaluation of the parameters, which requires two additional CORDIC operations, necessarily precedes application. The scheme due to Yang and Böhme allows simultaneous evaluation and application, using two vector rotations, along with additional arithmetic and scaling operations and we adopt this scheme for the integrated evaluation of parameters and application of the outer transformations in the Types II and III Q transformations.

The outer transformation in a Type I Q transformation is a triangularization which requires a Givens rotation [13]. It may be treated as a one-sided right rotation with the angle parameter (ψ for Type I Q transformation, see Table 2) determined by the lower row of the data matrix. Vectoring the lower row of the data matrix yields the parameter and achieves the desired result. However, the upper row (real and imaginary parts) also needs to be rotated by the angle parameter resulting from the vectoring operation. These operations may be done in parallel in a manner similar to the integrated evaluation and application of the inner transformation in the Type I Q transformation. Thus, the outer transformations may be applied and the parameters required for the Types I, II and III Q transformations may be evaluated in at most four CORDIC operations (two for the real and two for the imaginary 2 × 2 component matrices).

3.3 Area and Time Complexity

To achieve maximum parallelism in the implementation of the Q transformations, four CORDIC modules are needed. Also, no more than four CORDIC modules are necessary to achieve maximum concurrency. However, a mechanism for distribution of the necessary parameters among the modules is necessary as adjacent modules may need to share parameters and exchange data. The proposed arrangement of CORDIC modules to implement a Q transformation is shown in Figure 6.

Let T_Q be the time required to compute a Q transformation. The time complexity analysis presented below is indicative of the maximum parallelism that can be achieved on the architecture of Figure 6. Let T_C be the time to compute a basic CORDIC operation. We know that eight CORDIC operations are necessary for the evaluation and application of the inner transformation i.e. two CORDIC operations per data element in general. If only application of the inner transformation is desired the number of CORDIC operations reduces to four in all, i.e. one CORDIC operation per data element. The same is true for the evaluation and application of inner transformations in Types I-III Q transformations. Also, four CORDIC operations are necessary for both application (Types I, II & III) and the integrated evaluation and applica-

Table 4: Area and Time Complexity of Q Transformations

Implementation Methods				Apply	Evaluate & Apply
Inner	Outer	S-F. Corr.	Area	Time	Time
Rect. → Polar → Rect.	Cavallaro-Luk	K_n^2, K_n^2	$4A_C$	$(2.25 + 2.25)T_C$	$(2.25 + 3.25)T_C$
Rect. → Polar → Rect.	Yang-Böhme	K_n^2, K_n	$4A_C$	$(2.25 + 1.25)T_C$	$(2.25 + 1.25)T_C$
Complex Plane Rotation	Cavallaro-Luk	K_n, K_n^2	$4A_C$	$(1.25 + 2.25)T_C$	$(1.25^a + 3.25)T_C$
Complex Plane Rotation	Yang-Böhme	K_n, K_n	$4A_C$	$(1.25 + 1.25)T_C$	$(1.25^a + 1.25)T_C$

[a]May not apply to others than Types I, II & III

tion of outer transformations (Types II & III), if the schemes due to Yang and Böhme are used. As mentioned in the previous section, three CORDIC operations are needed for the integrated evaluation and application of the Type I outer transformation. Table 4 summarizes the area and time requirements (expressed as a sum of the times for inner and outer transformations, respectively) for applying and/or evaluating parameters for Q transformations with conventional (non-redundant) CORDIC.

3.3.1 Scale Factor Correction

The scale factors, as accumulated at the end of the inner and outer transformations, respectively, for the various methods chosen for implementation, are shown in Table 4. It is assumed that a modified polar coordinate representation of the complex argument is being used. If the extended range CORDIC algorithm is employed, K_n should be replaced by $K_e = 2K_n$. Scale-factor correction schemes, based on additional CORDIC-like iterations, are available to correct for K_n and K_n^2 in $0.25T_C$ [3, 4]. In the case of the extended range CORDIC algorithm an additional shift is necessary.

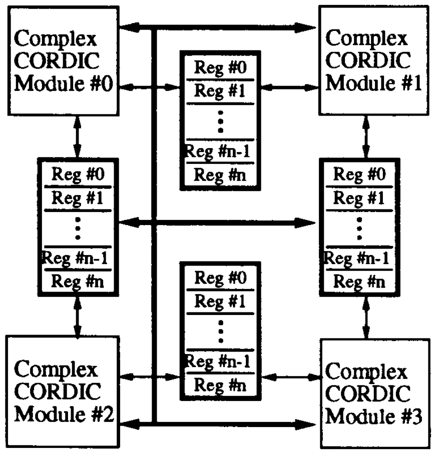

Figure 6: CORDIC Processor for Q Transformations

4 Redundant and On-Line CORDIC

In [6], Ercegovac and Lang present efficient implementations of CORDIC using redundant and on-line schemes which demonstrate considerable speed-up over conventional (non-redundant) CORDIC hardware. The CORDIC recurrence relations are modified to eliminate the need for two shifters, and carry-propagate adders are replaced by redundant adders (carry-save or signed digit). This implies that the modified y-recurrence may only be estimated, necessitating the use of a selection function, depending on the redundant representation used, to ascertain the choice of σs from the set $\{-1, 0, 1\}$. Also, the scale factor is now variable and dependent on the sequence of σs chosen.

To overcome the variable scale factor problem, Takagi *et. al.* [17], suggested a modification of the redundant CORDIC scheme to ensure a constant scale factor for sine and cosine computation. Lee and Lang [7], extended the constant scale factor redundant scheme (CFR-CORDIC) due to Takagi *et. al.* for angle calculation and rotation. Both redundant and CFR-CORDIC have been applied to matrix triangularization and SVD [6]. We now describe the use of the methodology presented in [6] for computing the SVD of real 2×2 matrices as extended to the evaluation and application of the inner and outer transformations of a Q transformation. With CFR-CORDIC, the hardware to compute scale factors is obviated.

4.1 Implementing Outer Transformations

Figure 7 shows an organization of redundant and on-line computation modules for the evaluation and application of Q transformations using redundant CORDIC. This is an extension of the diagonal and off-diagonal processor organization for the computation of the SVD described in [6], since the evaluation of the outer transformation and its application is essentially equivalent to the diagonalization of a real 2×2 matrix in the case of Types II and III Q transformations and a triangularization for Type I Q transformations. The hardware module for the evaluation of θ_l and θ_r consists of a partial redundant CORDIC module pipelined with two stages and an on-line module to compute the decomposition digits γ^l and γ^r. In addition, there are on-line modules to compute K_l, K_r and K, four partial on-line CORDIC rotation modules, an on-line multiplication module and eight (one for each of the two outputs of the four CORDIC rotation modules) on-line division and conversion to conventional representation modules. When compared to the hardware for the two-sided real 2×2 matrix transformation in [6], all modules except for the angle and scale factor evaluation modules, are duplicated. The time to evaluate and/or apply the outer transformation is $5n + 10$, where n is the data word-length, and is the same as the estimate in [6]. For further details about the composition of the individual modules shown in Figure 7 and the area and time complexity analyses, the reader is referred to [6].

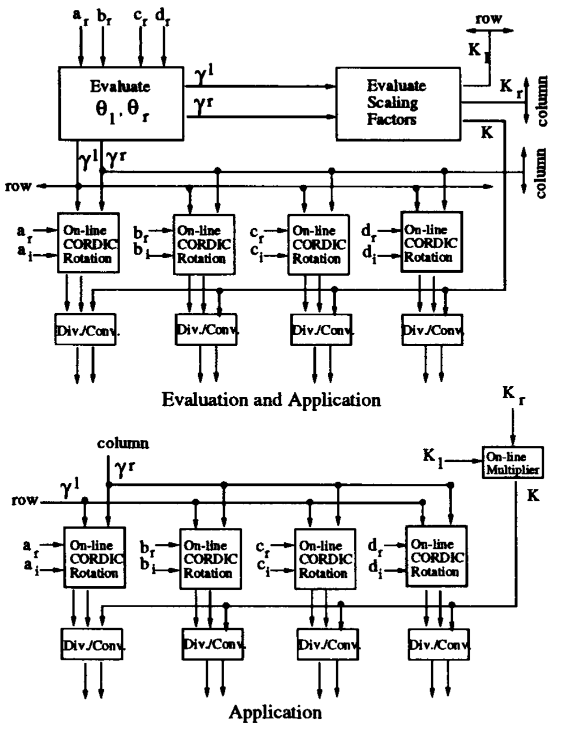

Figure 7: Redundant and On-Line CORDIC for Q Transformations

4.2 Implementing Inner Transformations

The inner transformations of Types I-III Q transformations depend on the arguments of two data elements. The hardware module for the evaluation of θ_l and θ_r in Figure 7 may be utilized to evaluate these arguments. However, instead of the two angle parameters generated in the outer transformation, decomposition digits corresponding to the four inner transformation parameters need be computed. Therefore, internal to the angle evaluation module, an additional module for computing decomposition digits is necessary. Note that the computation of the inner parameters is similar to the evaluation of θ_l and θ_r from θ_{sum} and θ_{diff} [6]. The application of the inner transformation may be treated as two consecutive rotations, corresponding to the left and right inner transformations, of each data element in the complex plane, and an organization similar to Figure 7 may be employed. However, the scale factor must be independently determined for each data element and the scaling factors corresponding to each of the four inner transformation parameters must be passed along the rows and columns. The time complexity is $5n + 10$, since it is similar to the implementation of the outer transformation.

5 Current Work

In the proposed redundant and on-line CORDIC implementations, both inner and outer transformations were applied sequentially as two consecutive rotations and the overhead due to evaluation was masked by the on-line nature of computations. However, the approaches were based on direct methods. It is possible to reduce the time required to compute inner and outer transformations, from $5n+\delta$ to $3n+\delta$, by using the schemes due to Yang and Böhme and improving the concurrency of computations. This necessitates a scheme for the on-line summation of angles [18]. Also, newer redundant and on-line CORDICschemes [19, 20] may be employed to improve implementations.

6 Conclusions

In this paper we introduced a two-sided unitary transformation (Q transformation) structured to permit integrated evaluation and application using CORDIC primitives. The Q transformation was shown to be useful as an atomic operation in parallel arrays for computing the eigenvalue/singular value decomposition of Hermitian/arbitrary matrices and three specific Q transformations which are needed in such arrays were identified. Issues relating to the use of CORDIC for complex arithmetic were addressed and implementations in both conventional (non-redundant) CORDIC and redundant and on-line modifications to CORDIC were described.

If the time to compute a CORDIC operation in non-redundant CORDIC be T_C, the Q transformations identified in this paper may be evaluated and/or applied in $2T_C$ using 4 CORDIC modules for maximum concurrency. In either case, $0.5T_C$ is required to account for scale factor correction. Using an extension of the redundant and on-line CORDIC proposed by Ercegovac and Lang for computing the SVD of real 2×2 matrices, it was shown that a Q transformation may be evaluated and/or applied in $\approx 10n$ i.e twice the time for a real 2×2 matrix diagonalization, where n is the desired bit-precision.

Acknowledgement

This work was supported in part by the National Science Foundation under awards MIP-8909498 and DDM-9202639.

Appendix A Diagonalization Methods

This appendix details the methods reported in the literature for the diagonalization of a real 2×2 matrix using a two-sided rotational transformation

$$\begin{bmatrix} \cos\theta_l & -\sin\theta_l \\ \sin\theta_l & \cos\theta_l \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\sin\theta_r & \cos\theta_r \end{bmatrix} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}.$$

A.1 Direct Two-Angle Method

The angles, θ_r and θ_l, are computed from $\theta_{sum} = \theta_r + \theta_l$ and $\theta_{diff} = \theta_r - \theta_l$, where $\theta_r \pm \theta_l = \arctan((c \pm b)/(d \mp a))$.

A.2 Yang and Böhme Diagonalization

This scheme reduces the computation required in the diagonalization. The principal advantage is that the evaluation of parameters for the rotational transformation is integrated with the application. However, a few additional arithmetic and scaling steps are necessary [5].

A.2.1 Evaluation and Application

The following are first computed from the input :

$$p_1 = \frac{(d+a)}{2},\ p_2 = \frac{(d-a)}{2}, q_1 = \frac{(c-b)}{2},\ q_2 = \frac{(c+b)}{2}.$$

Then, two vectoring (CORDIC y-reduction) operations are performed

$$r_1 = sign(p_1)\sqrt{p_1^2+q_1^2},\quad r_2 = sign(p_2)\sqrt{p_2^2+q_2^2},$$
$$\theta_- = \arctan(q_1/p_1),\quad \theta_+ = \arctan(q_2/p_2).$$

The required angles θ_l and θ_r, and the results of the diagonalization, s_1 and s_2, are then computed from

$$\theta_l = \frac{(\theta_+ - \theta_-)}{2},\ \theta_r = \frac{(\theta_+ + \theta_-)}{2}, s_1 = r_1 - r_2,\ s_2 = r_1 + r_2.$$

A.2.2 Application

As above, the following are first computed :

$$p_1 = \frac{(d+a)}{2},\ p_2 = \frac{(d-a)}{2}, q_1 = \frac{(c-b)}{2},\ q_2 = \frac{(c+b)}{2}.$$

Then, the angles θ_+ and θ_- are realized as $\theta_- = (\theta_r - \theta_l)$, $\theta_+ = (\theta_r + \theta_l)$, and two vector rotations (CORDIC z-reductions)

$$R(\theta_-)\begin{pmatrix}p_1\\q_1\end{pmatrix} \rightarrow \begin{pmatrix}r_1\\t_1\end{pmatrix},\quad R(\theta_+)\begin{pmatrix}p_2\\q_2\end{pmatrix} \rightarrow \begin{pmatrix}r_2\\t_2\end{pmatrix}$$

are performed. The transformed elements, a', b', c' and d' are computed from the results of the vector rotations as

$$a' = r_1 - r_2,\ b' = t_2 - t_1, c' = t_1 + t_2,\ d' = r_1 + r_2.$$

References

[1] R. P. Brent, F. T. Luk, and C. F. Van Loan, "Computation of the Singular Value Decomposition Using Mesh-Connected Processors," *Journal of VLSI and Computer Systems*, vol. 1, no. 3, pp. 242–270, 1985.

[2] J. S. Walther, "A Unified Algorithm for Elementary Functions," *AFIPS Spring Joint Computer Conf.*, pp. 379–385, 1971.

[3] J. R. Cavallaro and F. T. Luk, "CORDIC Arithmetic for an SVD Processor," *Journal of Parallel and Distributed Computing*, vol. 5, no. 3, pp. 271–290, June 1988.

[4] J. M. Delosme, "A Processor for Two-Dimensional Symmetric Eigenvalue and Singular Value Arrays," *IEEE 21th Asilomar Conf. on Circuits, Systems, and Computers*, pp. 217–221, November 1987.

[5] B. Yang and J. F. Böhme, "Reducing the Computations of the SVD Array given by Brent and Luk," *SIAM J. Matrix Analysis Appl.*, vol. 12, no. 4, pp. 713–725, 1991.

[6] M. D. Ercegovac and T. Lang, "Redundant and On-Line CORDIC: Application to Matrix Triangularization and SVD," *IEEE Trans. Computers*, vol. 39, pp. 725–740, June 1990.

[7] J. Lee and T. Lang, "Constant-Factor Redundant CORDIC for Angle Calculation and Rotation," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 1016–1025, Aug. 1992.

[8] S. Hitotumatu, "Complex Arithmetic through CORDIC," *Kodai Math. Sem. Rep.*, vol. 26, pp. 176–186, 1975.

[9] E. F. Deprettere and A.-J. van der Veen, "Parallel VLSI Matrix Pencil Algorithm for High Resolution Direction Finding," *IEEE Transactions on Signal Processing*, vol. 39, pp. 383–394, Feb. 1991.

[10] J. R. Cavallaro and A. C. Elster, "A CORDIC Processor Array for the SVD of a Complex Matrix," in *SVD and Signal Processing II (Algorithms, Analysis and Applications)*(R. Vaccaro, ed.), pp. 227–239, New York: Elsevier, 1991.

[11] J. M. Delosme, "Parallel Implementations of the SVD using Implicit CORDIC Arithmetic," in *SVD and Signal Processing II (Algorithms, Analysis and Applications)* (R. Vaccaro, ed.), pp. 33–56, New York: Elsevier, 1991.

[12] N. D. Hemkumar and J. R. Cavallaro, "An Efficient Parallel Implementation of the Jacobi SVD Algorithm for Arbitrary Matrices," Tech. Rep. ECE TR9212, Rice University, Houston, TX, Sept. 1992.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations, Second Edition*. Baltimore, MD: Johns Hopkins Univ. Press, 1989.

[14] N. D. Hemkumar and J. R. Cavallaro, "A Systolic VLSI Architecture for Complex SVD," in *IEEE ISCAS*, vol. 3, (San Diego, CA), pp. 1061–1064, May 1992.

[15] X. Hu, R. G. Harber, and S. C. Bass, "Expanding the Range of Convergence of the CORDIC Algorithm," *IEEE Trans. on Computers*, vol. 40(1), pp. 13–21, January 1991.

[16] D. Daggett, "Decimal-Binary Conversions in CORDIC," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 335–339, Sept. 1959.

[17] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," *IEEE Trans. Computers*, vol. 40, no. 9, pp. 989–995, Sept. 1991.

[18] N. D. Hemkumar and J. R. Cavallaro, "Redundant and On-Line CORDIC for Complex Matrix Transformations," Tech. Rep. ECE TR9301, Rice University, Mar. 1993.

[19] J. Duprat and J.-M.Muller, "The CORDIC Algorithm: New Results for Fast VLSI Implementation," *IEEE Transactions on Computers*, vol. 42, pp. 168–178, Feb. 1993.

[20] H. Dawid and H. Meyr, "VLSI Implementation of the CORDIC Algorithm using Redundant Arithmetic," in *IEEE ISCAS*, (San Diego, CA), pp. 1089–1092, 1992.