Redundant and On-Line CORDIC for Unitary Transformations

Nariankadu D. Hemkumar, *Student Member, IEEE*, and Joseph R. Cavallaro, *Member, IEEE*

Abstract—Two-sided unitary transformations of arbitrary 2×2 matrices are needed in parallel algorithms based on Jacobi-like methods for eigenvalue and singular value decompositions of complex matrices. This paper presents a two-sided unitary transformation structured to facilitate the integrated evaluation of parameters and application of the typically required transformations using only the primitives afforded by CORDIC; thus enabling significant speedup in the computation of these transformations on special-purpose processor array architectures implementing Jacobi-like algorithms. We discuss implementation in (nonredundant) CORDIC to motivate and lead up to implementation in the redundant and on-line enhancements to CORDIC. Both variable and constant scale factor redundant (CFR) CORDIC approaches are detailed and it is shown that the transformations may be computed in $10n + \delta$ time, where n is the data precision in bits and δ is a constant accounting for accumulated on-line delays. A more area-intensive approach using a novel on-line CORDIC encoded angle summation/difference scheme reduces computation time to $6n + \delta$. The area/time complexities involved in the various approaches are detailed.

Index Terms—CORDIC, redundant arithmetic, redundant and on-line CORDIC, unitary transformations, VLSI.

I. INTRODUCTION

SEVERAL real-time signal processing applications require fast computation of matrix factorizations like the eigenvalue and the singular value decomposition [25]. A variety of parallel architectures and algorithms proposed for the computation of these factorizations use Jacobi-like procedures which are amenable to parallel implementation [2], [12], [24], [26]. Central to these Jacobi-like procedures is the transformation of 2×2 submatrices of the input data matrix. Coordinate Rotation Digital Computer (CORDIC) [28, 29] algorithms, intended for the computation of inverse tangents and vector rotations, afford the kind of primitive operations needed for these transformations. Cavallaro and Luk [6], Delosme [8], and Yang and Böhme [30] have demonstrated the use of CORDIC for the 2×2 matrix transformations required in the computation of the singular value decomposition (SVD) of real matrices. Recently, redundant and on-line CORDIC based approaches

have been suggested by Ercegovic and Lang [13] and Lee and Lang [22].

Arbitrary data matrices occur in time-frequency signal analysis, adaptive detection and high-resolution spectral estimation [21]. The use of CORDIC for complex arithmetic was addressed by Hitotumatu [19] and applied to the computation of the FFT by Despain [11]. Earlier work in applying CORDIC to complex matrix transformations is due to Deprettere and van der Veen [10] and Cavallaro and Elster [5]. In [9], 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. Preliminary work describing the use of redundant and on-line CORDIC for complex matrix transformations was presented in [18].

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 be used as an atomic step in parallel eigenvalue and SVD arrays implementing Jacobi-like algorithms. We discuss implementation alternatives in conventional (nonredundant) CORDIC and the redundant and on-line enhancements to CORDIC. The paper is organized as follows. In Section II, we describe the one-sided and two-sided unitary transformations typically required in matrix factorization algorithms and present a formulation that is amenable to integrated evaluation and application using CORDIC. Sections III and IV discuss implementation in conventional and redundant CORDIC, respectively. Finally, we present the conclusions.

II. PRELIMINARIES

A. Jacobi-Like Algorithms and 2×2 Matrix Transformations

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 submatrices of the input matrix as

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

where the unitary matrices U_k and V_k are identity matrices embedded with 2×2 unitary matrices [14]. A singular value decomposition (SVD) of a matrix $A \in \mathbb{C}^{m \times n}$, is a factorization given by

$$A = U \Sigma V^H,$$

Manuscript received October 14, 1993; revised March 8, 1994. This work was supported in part by the National Science Foundation under award DDM-9202639 and by DOE Sandia National Laboratories Contract #18-4379A. A preliminary version of this paper was published as "Efficient complex matrix transformations with CORDIC," in *Proc. 11th Symp. Comput. Arithmetic*, Windsor, ON, Canada, June 29–July 2, 1993, pp. 122–129.

The authors are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, 77251-1892 USA; e-mail: hemkumar@rice.edu., cavallar@rice.edu.

IEEE Log Number 9403093.

TABLE I
INNER TRANSFORMATIONS

Type	Values	Effect
$\mathcal{I}_{main}$	$\alpha = \eta = -\beta = -\omega = (\theta_{jj} - \theta_{ii})/2$	$m_{ii} = R_{ii}e^{i\theta_{jj}}, m_{jj} = R_{jj}e^{i\theta_{ii}}$
$\mathcal{I}_{off}$	$\alpha = -\eta = -\beta = \omega = (\theta_{ji} - \theta_{ij})/2$	$m_{ij} = R_{ij}e^{i\theta_{ji}}, m_{ji} = R_{ji}e^{i\theta_{ij}}$
$\mathcal{R}_{upper}$	$\alpha = \beta = -(\theta_{ij} + \theta_{ii})/2, \eta = -\omega = (\theta_{ij} - \theta_{ii})/2$	$m_{ii} = R_{ii}, m_{ij} = R_{ij}$
$\mathcal{R}_{lower}$	$\alpha = \beta = -(\theta_{jj} + \theta_{ji})/2, \eta = -\omega = (\theta_{jj} - \theta_{ji})/2$	$m_{ji} = R_{ji}, m_{jj} = R_{jj}$
$\mathcal{C}_{left}$	$\alpha = -\beta = (\theta_{ji} - \theta_{ii})/2, \eta = \omega = -(\theta_{ji} + \theta_{ii})/2$	$m_{ii} = R_{ii}, m_{ji} = R_{ji}$
$\mathcal{C}_{right}$	$\alpha = -\beta = (\theta_{jj} - \theta_{ij})/2, \eta = \omega = -(\theta_{jj} + \theta_{ij})/2$	$m_{ij} = R_{ij}, m_{jj} = R_{jj}$
$\mathcal{D}_{main}$	$\alpha = \beta = -(\theta_{jj} + \theta_{ii})/2, \eta = -\omega = (\theta_{jj} - \theta_{ii})/2$	$m_{ii} = R_{ii}, m_{jj} = R_{jj}$
$\mathcal{D}_{off}$	$\alpha = \beta = -(\theta_{ij} + \theta_{ji})/2, \eta = -\omega = (\theta_{ij} - \theta_{ji})/2$	$m_{ij} = R_{ij}, m_{ji} = R_{ji}$

where $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m \times n}$ is a real nonnegative "diagonal" matrix of singular values [15]. If the matrix A is Hermitian, the above factorization yields the eigenvalues of A with $U = V$. For computing the eigenvalues of non-Hermitian matrices the schemes in [4], [12] may be employed.

In Jacobi-like methods for complex matrices, a 2×2 unitary matrix, which may be written as

$$\begin{bmatrix} e^{i\alpha} \cos \phi & -e^{i\beta} \sin \phi \\ e^{i\gamma} \sin \phi & e^{i\delta} \cos \phi \end{bmatrix},$$

with $\alpha - \beta - \gamma + \delta \equiv 0 \pmod{2\pi}$, is embedded in an identity matrix of appropriate dimensions to form the right or left transformation matrix [14]. If a 2×2 submatrix $m = [a_{pq}], (p, q = i, j)$ of a matrix A is given by

$$\begin{bmatrix} R_{ii}e^{i\theta_{ii}} & R_{ij}e^{i\theta_{ij}} \\ R_{ji}e^{i\theta_{ji}} & R_{jj}e^{i\theta_{jj}} \end{bmatrix},$$

then a two-sided transformation yielding a diagonalization may be expressed as

$$\begin{bmatrix} e^{i\alpha} \cos \phi & -e^{i\beta} \sin \phi \\ e^{i\gamma} \sin \phi & e^{i\delta} \cos \phi \end{bmatrix} \begin{bmatrix} R_{ii}e^{i\theta_{ii}} & R_{ij}e^{i\theta_{ij}} \\ R_{ji}e^{i\theta_{ji}} & R_{jj}e^{i\theta_{jj}} \end{bmatrix} \begin{bmatrix} e^{i\epsilon} \cos \psi & e^{i\eta} \sin \psi \\ -e^{i\zeta} \sin \psi & e^{i\omega} \cos \psi \end{bmatrix} \Rightarrow \begin{bmatrix} R'_{ii}e^{i\theta'_{ii}} & 0 \\ 0 & R'_{jj}e^{i\theta'_{jj}} \end{bmatrix}. \quad (1)$$

The two-sided transformation of (1), which diagonalizes an arbitrary 2×2 matrix, is central to parallel Jacobi-like algorithms for computing the SVD/eigenvalues of arbitrary/Hermitian matrices. Although (1) may be employed to diagonalize a 2×2 matrix, the computation of the parameters governing the left and right unitary matrices is not amenable to CORDIC implementation [17]. An alternate approach is to complete the diagonalization in two steps, each of which requires a two-sided transformation like (1) under the relaxing constraints $\alpha = \gamma, \beta = \delta, \xi = \eta$ and $\omega = \zeta$ [17]. The two-step diagonalization may be implemented in a pipelined manner by changing the systolic schedule on the arrays given in [2], [3] to decrease the overall computation time. Details of the modified schedule are beyond the scope of this paper and the reader is referred to [17]. We note that, on arrays which implement the Jacobi method, the input data matrix is partitioned into 2×2 submatrices across processors. Also, evaluation of parameters and application of transformations

is required at processors along the main diagonal while off-diagonal processors apply transformations upon receiving parameters from the appropriate diagonal processors [2], [3].

B. $\mathcal{Q}$ Transformations

Consider the 2×2 transformation matrices in (1) and let the left and right 2×2 matrices be denoted by $\mathcal{U}$ and $\mathcal{V}$, respectively. If $\alpha = \gamma, \beta = \delta, \xi = \eta$ and $\omega = \zeta$, then $\mathcal{U}$ and $\mathcal{V}$ may be factorized as

$$\mathcal{U} = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\beta} \end{bmatrix}, \quad (2)$$

$$\mathcal{V} = \begin{bmatrix} e^{i\eta} & 0 \\ 0 & e^{i\omega} \end{bmatrix} \begin{bmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{bmatrix}.$$

We shall denote the factors as the inner and outer matrices, where the connotation of inner and outer is the obvious, and the resulting transformations as the inner and outer transformations (different from inner rotations and outer rotations [24]).

Definition 1: A $\mathcal{Q}$ 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 (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 I shows some choices of the parameters α, β, η and ω and their effect on the elements of m . 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. We now state a result regarding the combination of inner transformations.

Lemma 1: If $\tau = f(\alpha, \beta, \eta, \omega)$ is an inner transformation, then

$$a^{(\lambda)} = \tau_u^{(\lambda)} \dots \tau_u^{(2)} \tau_u^{(1)} a \tau_v^{(1)} \tau_v^{(2)} \dots \tau_v^{(\lambda)} = \tau'_u a \tau'_v,$$

where $\alpha' = \sum_{x=1}^{\lambda} \alpha^{(x)}, \dots, \omega' = \sum_{x=1}^{\lambda} \omega^{(x)}$. When $\tau^{(x)}, 1 \leq x \leq \lambda$, is a transformation from Table I, determine $\tau^{(x)} = f(\alpha^{(x)}, \beta^{(x)}, \eta^{(x)}, \omega^{(x)})$ from $a^{(x-1)}$ with $a^{(0)} = a$.

In Table II, 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

TABLE II
PARAMETERS FOR Q TRANSFORMATIONS

Type	Inner Parameters	Outer Parameters
I	$\alpha = \beta = -(\theta_{jj} + \theta_{ji})/2, \eta = -\omega = (\theta_{jj} - \theta_{ji})/2$	$\phi = 0, \psi = \tan^{-1}(R_{ji}/R_{jj})$
II	$\alpha = -(\theta_{ii} + \theta_{ij})/2, \beta = \eta = -\omega = (\theta_{ij} - \theta_{ii})/2$	$\tan(\phi \pm \psi) = R_{ij}/(R_{jj} \mp R_{ii})$
III	$\alpha = -\beta = -\eta = \omega = -\theta_{ij}/2$	$2\psi = 2\phi = \tan^{-1}(2R_{ij}/(R_{jj} - R_{ii}))$

2×2 matrix using an $\mathcal{R}_{\text{lower}}$ inner transformation and a Givens rotation [15] on the right outer transformation. The Type II transformation diagonalizes an upper triangular matrix using a combination of $\mathcal{D}_{\text{main}}$ and $\mathcal{I}_{\text{off}}$ inner transformations in that order, together with an outer two-sided diagonalization of a real 2×2 matrix ([6, cf. Appendix B]). The Type III Q transformation, which consists of an inner transformation combining an $\mathcal{R}_{\text{upper}}$ and a $\mathcal{C}_{\text{right}}$ transformation with an outer two-sided diagonalization as in the Type II 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: Given an arbitrary 2×2 matrix choose a Type I Q transformation followed by a Type II Q transformation for diagonalization. A Type III Q transformation may be used to diagonalize any 2×2 Hermitian matrix. (cf. [17].) $\square$

C. The CORDIC Algorithms

The Coordinate Rotation Digital Computer (CORDIC) technique was initially developed by Volder [28] and later unified for elementary functions by Walther [29]. CORDIC is based upon defining a vector (x_0, y_0) and applying a rotational transformation. Of the three rotational modes possible - linear, circular and hyperbolic, in the context of SVD algorithms, only the circular mode is of interest. In the circular mode, CORDIC decomposes the rotation angle θ into a sequence of n known angles $\sigma_i \theta_i$, such that $\theta = \sum_{i=0}^{n-1} \sigma_i \theta_i$, where $\theta_i = \tan^{-1} 2^{-i}$ and $\sigma_i = \pm 1$. The CORDIC recurrence relations are

$$\begin{aligned} x_{i+1} &= x_i + \sigma_i y_i 2^{-i} \\ y_{i+1} &= y_i - \sigma_i x_i 2^{-i} \\ z_{i+1} &= z_i + \sigma_i \theta_i. \end{aligned} \quad (3)$$

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 (3), 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 divided by an accumulated scale factor

$$K_n = \prod_{i=0}^{n-1} k_i = \prod_{i=0}^{n-1} \frac{1}{\cos \theta_i} = \prod_{i=0}^{n-1} \sqrt{1 + \sigma_i^2 2^{-2i}}. \quad (4)$$

Several schemes have been proposed in the literature for scale factor correction. Haviland and Tuszynski [16] Ahmed [1] and Delosme [8] have proposed schemes to correct for K_n using CORDIC like iterations. Cavallaro and Luk [6] postpone

Fig. 1. Angle formats for the complex argument. (a) Modified polar representation angle format. (b) Extended range angle format (Daggett's).

correction until two CORDIC operations are performed and correct the final values of the variables by K_n^2 .

III. CORDIC FOR Q TRANSFORMATIONS

A. CORDIC and Complex Arithmetic

Walther [29] 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 $\theta_{\max}$ sums to $\approx 1.7433(99^\circ)$. 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.

Modified Polar Coordinate Representation of Complex Data: A complex number, $z = z_r + iz_i$ may be written as $z = R_z e^{i\theta_z}$, where

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

with $\theta_z \in [-\pi/2, \pi/2]$. 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* [19]. Note that the sign of the modulus is +ve in the right half-plane and -ve 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 $\theta \in [-\pi/2, \pi/2]$ (See Fig. 1 for angle format).

Extending the Range of CORDIC Algorithms: In [20], Hu et al., show how the range of convergence of the CORDIC algorithm may be expanded. Using an angle format (Fig. 1) suggested by Daggett [7] 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

TABLE III
ADDITION OF ARGUMENTS IN MODIFIED POLAR COORDINATE REPRESENTATION

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

extended to $\theta_{max} \approx 3.3141(189.9^\circ)$. The angle formats of Fig. 1 are alternate two's complement representations of $\theta(\text{mod } \pi)$.

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 are needed to handle the addition of complex arguments. The rules are tabulated in Table III, where H is the bit-value representing the half-plane of the argument, and θ 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 $-ve$) 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 ($K_e = 2K_n$).

B. Implementing Q Transformations Using CORDIC

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 Q transformations facilitates the use of these very primitives. We now show how evaluation of the parameters for the inner and outer transformations and their respective application is integrated through the use of CORDIC.

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 on the arguments of the complex data elements (Tables I, II). 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 and apply the parameters for 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. Fig. 2 shows the steps

```

input ←  $\begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix}$ 
begin
  step 1: par do /* transform data elements from rectangular to polar representation using CORDIC y-reductions */
     $x_r + ix_i \rightarrow K_n X e^{i\theta_x}, \forall x \in \{a, b, c, d\}$ 
  end
  step 2: par do /* compute inner parameters from  $\{\theta_a, \theta_b, \theta_c, \theta_d\}$  */
     $\{\alpha, \beta, \eta, \omega\} = f(\{\theta_a, \theta_b, \theta_c, \theta_d\})$ 
  end
  step 3: par do /* add appropriate left and right inner parameters to arguments and transform elements back to rectangular representation using CORDIC z-reductions */
     $K_n X e^{i(\theta_x + \theta_{in}^L + \theta_{in}^R)} \rightarrow K_n^2(x'_r + ix'_i)$ 
  end
  step 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 →  $\alpha, \beta, \eta, \omega, \begin{bmatrix} a'_r + ia'_i & b'_r + ib'_i \\ c'_r + ic'_i & d'_r + id'_i \end{bmatrix} = \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\beta} \end{bmatrix} \begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix} \begin{bmatrix} e^{i\eta} & 0 \\ 0 & e^{i\omega} \end{bmatrix}$ 

```

Fig. 2. Algorithm CORDIC inner transformation—Evaluation and application.

```

input ←  $\begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix}, \alpha, \beta, \eta, \omega$ 
begin
  step 1: par do /* rotate data element in the complex plane using CORDIC z-reductions and the sum of the appropriate left and right inner parameters */
     $R(\theta_{in}^L + \theta_{in}^R) \begin{bmatrix} x_r \\ x_i \end{bmatrix} \rightarrow K_n \begin{bmatrix} x'_r \\ x'_i \end{bmatrix}, \forall x \in \{a, b, c, d\}$ 
  end
  step 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 →  $\begin{bmatrix} a'_r + ia'_i & b'_r + ib'_i \\ c'_r + ic'_i & d'_r + id'_i \end{bmatrix} = \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\beta} \end{bmatrix} \begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix} \begin{bmatrix} e^{i\eta} & 0 \\ 0 & e^{i\omega} \end{bmatrix}$ 

```

Fig. 3. Algorithm CORDIC inner transformation—Application.

in the integrated evaluation (of parameters) and application of inner transformations and Fig. 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 of) the respective data. The Type I inner makes a_{ji} and a_{jj} real; for Types II and III, the elements pairs are $\{a_{ii}, a_{ij}\}$ and $\{a_{ij}, a_{ji}\}$, respectively ((6) shows the effect of Type I inner transformations). Thus, one CORDIC operation per affected data element is needed to apply the

Fig. 4. CORDIC Type I inner transformation—Evaluation and application.

Fig. 5. CORDIC outer transformation: Application. (a) Cavallaro and Luk two-sided Rotation. (b) Yang and Böhme two-sided rotation.

inner transformations

$$\begin{bmatrix} R_{ii}e^{i\theta_{ii}} & R_{ij}e^{i\theta_{ij}} \\ R_{ji}e^{i\theta_{ji}} & R_{jj}e^{i\theta_{jj}} \end{bmatrix} \xrightarrow{\text{Type I Inner}} \begin{bmatrix} R_{ii}e^{i(\theta_{ii}-\theta_{ji})} & R_{ij}e^{i(\theta_{ij}-\theta_{jj})} \\ R_{ji} & R_{jj} \end{bmatrix}. \quad (6)$$

Additionally, in the case of the Type I $\mathcal{Q}$ transformation (6), 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 one 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 resulting sequence of σ s (cf. [23]—rotation angle in decomposed form) to drive the rotation of the data elements in the upper row, accomplishes the inner transformation using two pairs of tandem CORDIC operations (See Fig. 4). Also, in all three cases (Types I, II & III), the individual 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.

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 [6], 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 ([30], cf. Appendix B), 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.

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 $\mathcal{Q}$ 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 $\mathcal{Q}$ transformations and the Type I $\mathcal{Q}$ transformation triangularizes the 2×2 matrix. Also, it may be seen from Table II that the $\mathcal{Q}$ 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 $\mathcal{Q}$ transformation).

In the literature, there are two methods available to compute the parameters required for the diagonalization of a real 2×2 matrix. Appendix B details the direct two-angle method for the diagonalization of real 2×2 matrices and the scheme due to Yang and Böhme [30]. A disadvantage of the direct two-angle method for diagonalization is that the evaluation of parameters, which requires two additional CORDIC operations, necessarily precedes application. The scheme due to Yang and Böhme

TABLE IV
AREA AND TIME COMPLEXITY OF Q TRANSFORMATIONS

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

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 [15]. It may be treated as a one-sided right rotation with the angle parameter (ψ for Type I Q transformation, see Table II) 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 (recall that the lower row is real after the Type I inner transformation). 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 with at most four CORDIC operations (two for the real and two for the imaginary 2×2 component matrices).

C. Area and Time Complexity

To achieve maximum parallelism in the implementation of the Q transformations, no more than four CORDIC modules are needed. However, a mechanism for distribution of the transformation 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 Fig. 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 Fig. 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, and III) and the integrated evaluation and application of outer transformations (Types II and 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 evalu-

Fig. 6. Architecture to implement Q transformations.

ation and application of the Type I outer transformation. Table IV 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 (nonredundant) CORDIC (The time complexities for evaluation and application are for the Q transformations described in this paper).

Scale Factor Correction: Table IV lists the scale factors as accumulated at the end of the inner and outer transformations for the various methods chosen for implementation. The values correspond to the case when a modified polar coordinate representation of complex numbers is being used. Otherwise, K_n should be replaced by $K_e = 2K_n$ if the extended range CORDIC algorithm is employed. Scale-factor correction schemes, based on additional CORDIC-like iterations, are available to correct for K_n and K_n^2 in $0.25T_C$ [6], [8]. In the case of the extended range CORDIC algorithm an additional shift is necessary.

IV. REDUNDANT AND ON-LINE CORDIC FOR Q TRANSFORMATIONS

In [13], Ercegovic and Lang present efficient implementations of CORDIC (fixed or floating point) using redundant and on-line schemes which demonstrate considerable speed-up over conventional (nonredundant) CORDIC hardware. The

Fig. 7. Extended Ercegovac-Lang redundant CORDIC for Q transformations. (a) Evaluation and application. (b) Application.

CORDIC recurrence relations are modified to eliminate the need for one shifter, 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 σ s from the set $\{-1, 0, 1\}$. Also, the scale factor is now variable and dependent on the sequence of σ s chosen (see (4)).

To overcome the variable scale factor problem, Takagi *et al.* [27], suggested a modification of the redundant CORDIC scheme to ensure a constant scale factor for sine and cosine computation. Lee and Lang [23], 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 [13], [22]. Since the approach is essentially the same for both variable and constant scale factor methods, we first detail the variable scale factor organizations and then discuss the differences *vis-à-vis* CFR-CORDIC.

A. Extending Ercegovac-Lang CORDIC to Q Transformations

Redundant and on-line CORDIC architectures to compute the SVD of real 2×2 matrices were presented in [13]. Since both the inner and the outer transformations are akin to two-sided rotations, the organizations in [13] extend naturally to Q transformations. Apart from the need to replicate certain modules to handle additional data, the essential composition of the hardware remains the same. Figures 7(a) and 7(b) show the organization of redundant and on-line computation modules for the evaluation and/or application of Q transformations. To avoid reproduction and due to space constraints, the reader is referred to [13] for further details about the components of the individual modules shown in Fig. 7 and the respective area and time complexity analyses.

Implementation of Outer Transformations: The outer transformation is a diagonalization of a real 2×2 matrix in the case of Types II and III Q transformations and a triangularization (a special case of diagonalization) for Type I Q transformations. The evaluation of the parameters for the outer transformation and the computation of the scale factors is

done with dedicated hardware distinct from that required for application, as in [13]. The hardware module for the evaluation of parameters (see Fig. 7(a)) consists of a partial redundant parallel CORDIC module pipelined with two stages for the interleaved computation of up to two angle sequences. This unit computes the decomposition digits (σ s) for θ_{sum} and θ_{diff} in the case of diagonalization, and a single angle parameter θ , in the case of triangularization. However, in the case of diagonalization, the parameters of interest are θ_l and θ_r (see Appendix B). A scheme to facilitate the on-line computation of the decomposition digits γ^l and γ^r corresponding to the required parameters θ_l and θ_r from the angle sequences for θ_{sum} and θ_{diff} was described in [13] (cf. discussion later in this paper). This hardware is included in the evaluation module but is not needed in the case of the Type I Q transformation. Additionally, since the scale factor is variable, an on-line module is needed to compute scale factors for the decomposition digit sequences.

The hardware for application of the outer transformation (see Fig. 7(b)) consists of four partial on-line CORDIC rotation modules which sequentially apply the left and right rotations, to the four vectors corresponding to the columns and rows, respectively, of the real and imaginary component 2×2 matrices of the input data matrix. An on-line multiplication module to compute the final scale factor K from the product of K_l and K_r and eight on-line division and conversion to conventional representation modules, complete the requirements for the application hardware. Except for the angle and scale factor evaluation units, all other modules are duplicated for the implementation of the outer transformation when compared to the two-sided real 2×2 matrix transformations described in [13]. The time to evaluate and apply the outer transformation is $5n + 10$ and the time for application is $5n + 4$, where n is the data word-length. These are the same as the estimates in [13].

Implementation of Inner Transformations: The parameters for the inner transformations of Types I-III Q transformations depend on the arguments of at most two data elements (see Table II). The hardware module for the evaluation of θ_l and θ_r , shown in Fig. 7(a) may be utilized to evaluate these arguments since the computation is similar (the decomposition digits of

the arguments are computed just as θ_{sum} and θ_{diff} , and the inner parameters are a similar function of the arguments as θ_l and θ_r are of θ_{sum} and θ_{diff} . Only, instead of the two angle parameters generated in the outer transformation, decomposition digits corresponding to the four inner transformation parameters need to be computed. However, by utilizing the fact that if we ignore signs, in terms of magnitudes, there are at most two different angle sequences. Also, this implies that there are at most two different scale factors¹ (K_1, K_2 , and at most three different scale factor products; two squares and a product, $K_1^2, K_2^2, K_1 K_2$). Thus, by adding the ability to generate a complementary negative sequence in the evaluation units, we may compute all necessary angle sequences; and no additional modules for computing decomposition digits are necessary.

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 similar² to Fig. 7(b) may be employed. 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. Due to the reasons stated above, three additional on-line multipliers (at most four different scale factor products) are needed for the application only organization. The time complexity is similar to the implementation of the outer transformation.

B. Enhancing Performance With Additional Hardware

In using an approach similar to that suggested by Ercegovic and Lang [13] for the outer transformation and adapting it to implement the inner transformation we exploited the solutions already available. 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. In order to employ the scheme due to Yang and Böhme in the implementation of the outer transformation and improve the concurrency in that of the inner transformation a scheme for the on-line summation/difference of angle sequences needs to be devised.

On-Line Summation and Difference of Rotation Angles: In the computation of the left and right angles using the direct two-angle method for diagonalizing a real 2×2 matrix (see Appendix B), an on-line addition/subtraction of angles followed by a division by two is necessitated and a solution was proposed in [13]. Given a sequence of inputs $\sigma_j = \{0, \pm 1/2, \pm 1\}$, it was required that an equivalent stream $\gamma_j = \{0, \pm 1\}$ be generated such that

$$\sum_{j=0}^{n-1} \sigma_j \tan^{-1}(2^{-j}) = \sum_{j=0}^{n-1} \gamma_j \tan^{-1}(2^{-j}),$$

$$\sigma_j = \{0, \pm 1/2, \pm 1\}, \gamma_j = \{0, \pm 1\}. \quad (7)$$

¹ Squares of scale factors are intermediate results in scale factor computation and no additional on-line multipliers are needed.

² Figure 7(b) does not show the additional modules and Fig. 7(a) and (b) does not show the per CORDIC unit distribution of scale factors as required for inner transformations.

With an on-line delay $p = 2$, a selection function for the value γ_j was defined using two fractional bits of a residual function. This scheme is used in the implementation of the outer and the inner transformations as discussed earlier.

A similar problem arises in the application of the Yang and Böhme two-sided rotation scheme (see Appendix B) where the angles θ_+ and θ_- are generated from the sum and difference of θ_r and θ_l . Only, there is no division by two to keep the range of the angle the same as that of the input streams. Here, given a sequence of inputs $\sigma_j = \{0, \pm 1, \pm 2\}$, it is required that an equivalent stream $\gamma_j = \{0, \pm 1\}$ be generated. However, due to the potential for overflow they cannot be related by (7). Hence, we propose that two extra rotations with index $j = 0$ be performed using the two σ_0 s of the incoming streams. Then, an equivalent stream $\gamma_j = \{0, \pm 1\}$ needs to be generated from the remaining $\sigma_j = \{0, \pm 1, \pm 2\}$ values such that

$$\sum_{j=1}^{n-1} \sigma_j \tan^{-1}(2^{-j}) = \sum_{j=0}^{n-1} \gamma_j \tan^{-1}(2^{-j}),$$

$$\sigma_j = \{0, \pm 1, \pm 2\}, \gamma_j = \{0, \pm 1\}. \quad (8)$$

In spirit, this is similar to the scheme proposed in [20] for extending the range of CORDIC. In view of (8), it is more convenient to use the modified polar coordinate representation of complex numbers so that the argument is restricted to the principal values of *arctangent*. Indeed, the previous organizations do implicitly assume that the modified polar coordinate representation is employed.

In order to perform the computation in (8) on-line, potentially with a delay p , we define the residual³

$$z[j] = 2^j \left(\sum_{i=0}^{j+p-1} \sigma_i \tan^{-1}(2^{-(i+1)}) - \sum_{i=0}^{j-1} \gamma_i \tan^{-1}(2^{-i}) \right).$$

Proceeding in a manner similar to [13], we may write the recurrence as

$$z[j+1] = 2 \left(z[j] + \sigma_{j+p} 2^j \tan^{-1}(2^{-(j+p+1)}) - \gamma_j 2^j \tan^{-1}(2^{-j}) \right),$$

where

$$z[0] = \sum_{i=0}^{p-1} \sigma_i 2^i \tan^{-1}(2^{-(i+1)}).$$

The selection function may be simplified, if the recurrence is further decomposed into two through the addition of an extra variable

$$w[j] = z[j] + \sigma_{j+p} 2^j \tan^{-1}(2^{-(j+p+1)}),$$

which converts the recurrence to

$$z[j+1] = 2(w[j] - \gamma_j 2^j \tan^{-1}(2^{-j})).$$

Note that now, in addition to storing the constants $2^j \tan^{-1}(2^{-j})$, the constants $2^j \tan^{-1}(2^{-(j+1)})$ need to be stored in the ROM [13]. In Appendix A we realize a

³ The residual reassigns the meaning of the indices ($\sigma_j = \sigma_{j-1}$) so as to treat the conversion as a separate problem.

Fig. 8. Organizations for outer transformations. (a) Evaluation and application. (b) Application.

selection function to determine γ from $\hat{w}$, an estimate of w . The selection function for a carry-save adder, identical to the one proposed in [13] for (7), uses an on-line delay $p = 2$ and two fractional bits of w , and is given by

$$\gamma_j = \begin{cases} 1 & \text{if } \hat{w}[j] \geq 1/2 \\ 0 & \text{if } -1/2 \leq \hat{w}[j] \leq 1/4 \\ -1 & \text{if } \hat{w}[j] \leq -3/4. \end{cases}$$

Improving the Performance of Outer Transformations: For evaluation and application of outer transformations, we propose an organization as shown in Fig. 8(a). The organization is similar to that proposed in Fig. 7(a). The Yang and Böhme diagonalization scheme for real 2×2 matrices is employed and the redundant parallel CORDIC unit evaluates θ_+ and θ_- from the appropriate inputs. Note that the decomposition digits γ^l and γ^r for θ_l and θ_r do not need to be computed although they are not needed locally (generation of γ^l and γ^r from θ_+ and θ_- is as in [13]). However, the accompanying scale factors do not need to be computed since the appropriate left and right angle sequences will be combined before application to compute the required θ_+ and θ_- angle sequences and concomitant scale factors.

The transformation is applied locally using four on-line CORDIC units which are fed appropriate inputs. The Yang and Böhme method requires arithmetic and scaling steps after rotation to complete the diagonalization and for this purpose we use on-line hardware. This approach requires $3n + 12$ (See Fig. 9(a)) time steps to evaluate and apply the outer transformation. In the case of the Type I $\mathcal{Q}$ transformation where the outer transformation is a triangularization, the driving CORDIC y -reduction is computed on the redundant parallel CORDIC unit. The on-line CORDIC units apply the rotation to the row vectors of the real and imaginary parts. The arithmetic or scaling steps used in the case of the two-sided rotation with the Yang and Böhme scheme, are not necessary.

Figure 8(b) shows the proposed organization of computation modules for the application of outer transformations. We use

Fig. 9. Timing of outer transformations. (a) Evaluation and application. (b) Application.

the Yang and Böhme two-sided rotation scheme to reduce the computation time. However, an on-line sum and difference of the incoming left and right transformation parameters needs to be computed. The on-line summation and difference module shown in Fig. 8(b) employs the method proposed earlier in this section. The two-sided rotation is applied separately to the real and imaginary parts of the input 2×2 matrix. We expect the time to apply the outer transformation using the organization proposed in Fig. 8(b) to be $3n + 14$ (Fig. 9(b)). The on-line CORDIC units shown in Fig. 8(b) allow two additional rotations with index $j = 0$ to account for the potential increase in the range of the angle parameters due to the sum/difference

Fig. 10. Organizations for inner transformations. (a) Evaluation and application (type I exemplified). (b) Application.

TABLE V
TIME COMPLEXITY OF Q TRANSFORMATIONS USING REDUNDANT CORDIC

Implementation Methods		Apply	Evaluate & Apply
Inner	Outer	Time	Time
2 Complex Plane Rotations	Ercegovic-Lang	$(5n + 4) + (5n + 4)$	$(5n + 10) + (5n + 10)$
1 Complex Plane Rotation	Yang-Böhme	$(3n + 12) + (3n + 14)$	$(3n + 10) + (3n + 12)$

computation and thus have a depth of $n + 2$ stages instead of the usual n stages. Note that the scale factors need to be computed locally.

Improving the Performance of Inner Transformations: Integrated evaluation and application of the inner transformations of Types I, II and III Q transformations may be performed using an organization of redundant and on-line CORDIC units as shown in Fig. 10(a) (the set of inputs shown correspond to Type I inner transformation). The two-stage pipelined redundant parallel CORDIC unit, similar to the one used in the evaluation and application of the outer transformation, computes the arguments of the two data elements from which the inner parameters may be determined. The on-line computation of the inner parameters from the two arguments is done using the scheme from [13] for the generation of the left and right angles. Up to four on-line CORDIC units may be needed to apply the inner transformations. In the case of Type II and III Q transformations, only two data elements are affected. The time to evaluate and apply the inner transformation using the hardware of Fig. 10(a) is $3n + 10$ (Fig. 11(a)).

In order to reduce the computation time in the application of inner transformations, we propose to compute, in an on-line manner, the sums of the corresponding left and right inner parameters of each data element and use it to rotate the data element in the complex plane. Four sums and the corresponding scaling factors are computed using the on-line scheme presented earlier. This scheme is similar to the one employed in the application of the outer transformation using the Yang and Böhme two-sided rotation.

The rotations are applied using four on-line CORDIC units as shown in Fig. 10(b). These CORDIC units allow two

additional rotations with indices $j = 0$ and have $n + 2$ stages as opposed to the n stages that normally make up an on-line CORDIC unit. The CORDIC units are similar to the ones shown in Fig. 8(b) for the application of the outer transformation. The time to apply an inner transformation using the hardware in Fig. 10(b) is $3n + 12$ (Fig. 11(b)). Table V summarizes the time required to evaluate and/or apply the inner and the outer transformations of a Q transformation using the various organizations proposed in this section in Figs. 7–10.

The various component modules that are needed in the redundant and on-line organizations presented earlier are listed in Table VI. The area requirements in terms of the requisite component modules for the different organizations is summarized in Table VII. Comparison of the A_{eval} and $A_{app(n)}$ is discussed in [13]. $A_{app(n+2)}$ is approximately $((n + 2)/n)A_{app(n)}$. The difference between $A_{((\theta_1 \pm \theta_2)/2)}$ and $A_{(\theta_1 \pm \theta_2)}$ is that the latter requires an additional ROM table (need to store the constants $2^j \tan^{-1}(2^{-(j+1)})$ in addition to $2^j \tan^{-1}(2^{-j})$).

C. Constant Scale-Factor Redundant CORDIC Implementations

An examination of the various organizations of redundant and on-line CORDIC modules as proposed in this section reveals the need for the dedication of hardware to compute scale factors for each stream of angles generated and applied. Schemes have been proposed for redundant CORDIC which do not have the variable scale factor problem. Takagi *et al.* [27], modified the redundant CORDIC scheme to ensure a constant scale factor for sine and cosine computation. Lee and Lang [23], extended the constant scale factor redundant scheme

Fig. 11. Timing of inner transformations. (a) Evaluation and application. (b) Application.

TABLE VI
COMPUTATION MODULES FOR REDUNDANT
AND ON-LINE CORDIC ARCHITECTURES

Description	Abbreviation
2-stage pipelined redundant CORDIC	A_{eval}
On-line CORDIC	$A_{app(n)}$
On-line CORDIC ($n+2$ stages)	$A_{app(n+2)}$
On-line half of sum/difference (Ercegovic-Lang)	$A_{((\theta_1 \pm \theta_2)/2)}$
On-line sum/difference (Hemkumar-Cavallaro)	$A_{(\theta_1 \pm \theta_2)}$
Scale Factor	A_{sfc}
On-line Multiply	A_{mul}
On-line Divide	A_{div}
On-line Division/Conversion	$A_{div/conv}$
On-line Add-Subtract/Conversion	$A_{add/conv}$

(CFR-CORDIC) due to Takagi *et al.* for angle calculation and rotation.

CFR-CORDIC and On-line Summation/Difference of Angles: In [22], Lee and Lang address the issue of computing the decomposition digits for θ_l and θ_r from θ_{sum} and θ_{diff} while ensuring constant scale factors. This scheme extended the Ercegovic and Lang scheme with variable scale factors to CFR-CORDIC. The central idea was to choose the decomposition digits γ_l and γ_r from the set $\{1, -1\}$ for $0 \leq j < n/2$ and from $\{1, 0, -1\}$ for $n/2 \leq j < n$. This ensures that the scale factors remain constant since the choice of γ_j for rotations corresponding to $n/2 \leq j < n$ does not affect the scale factor for n -bit data.

The selection functions then are,

$$0 \leq j < n/2 : \gamma_j = \begin{cases} 1 & \text{if } \hat{w}[j] \geq 0 \\ -1 & \text{if } \hat{w}[j] < 0 \end{cases}$$

and

$$n/2 \leq j < n : \gamma_j = \begin{cases} 1 & \text{if } \hat{w}[j] \geq 1/2 \\ 0 & \text{if } -1/2 \leq \hat{w}[j] \leq 1/4 \\ -1 & \text{if } \hat{w}[j] \leq -3/4 \end{cases}$$

The selection function for the second half of the iterations is the same as that for the variable scale factor scheme. As mentioned in [22], the selection function satisfies the constant scale factor requirement but does not satisfy the convergence constraints. By using the fact that

$$\sigma_j 2^j \tan^{-1}(2^{-(j+p+1)}) \leq 2^{-p}, \sigma_j \in \{\pm 1, \pm 2\},$$

we may infer that the correction iteration scheme suggested in [22] is suitable for the case of the on-line scheme for summation/difference of angles presented earlier when adapted to CFR-CORDIC.

Let m be the frequency indicator for a correcting iteration, p the on-line delay and t the number of fractional bits used in estimating w , then the relation

$$2^{-t} + 2^{-(p+1)} < 2^{1-m}$$

was shown to hold in [22] which results in a solution where $m = p + 1$ and $t = p + 2$. With an on-line delay of $p = 3$, a correcting iteration of the form

$$z[j+m]^c = w[j+m] - \gamma_{j+m}^c 2^{j+m+1} \tan^{-1} 2^{-(j+m)},$$

$$w[j+m]^c = z[j+m]^c,$$

where γ_{j+m}^c is chosen according to the selection function for the first half, must occur with an interval of 4 or less iterations; and $t = 5$ fractional bits are used in computing the estimate $\hat{w}$ of w .

Area and Time Complexities for CFR-CORDIC Implementations: With the use of CFR-CORDIC, the need to compute scaling factors is eliminated. Nevertheless, the organizations proposed in Figs. 7–10 are still valid. Only, all hardware dedicated to the computation of scaling factors is no longer necessary. Thus the area requirement is considerably reduced [23]. The time for computation is reduced with respect to the variable scale-factor schemes, but not substantially. This is due to the need for additional correcting iterations necessary to ensure convergence while keeping the scale-factor a constant.

The discussion on the area requirements for the variable scale factor organizations also holds for the case of CFR-CORDIC. As mentioned earlier, the dedication of hardware to compute scale factors is obviated. However, to better account for the tradeoffs, it is necessary to know the extent to which various methods to reduce the number of correcting, scaling and regular iterations are employed in a particular implementation. For example, in [23], a combination of radix-2 and radix-4 arithmetic is motivated to minimize the number of overall iterations (additional and scaling iterations). An

TABLE VII
AREA COMPLEXITIES FOR REDUNDANT AND ON-LINE CORDIC ARCHITECTURES

Organization	Area (Breakdown by Component Modules)
Extended Ercegovac-Lang Application	$4A_{app(n)} + A_{mul} + 8A_{div/conv} (+ 3A_{mul} \text{ if inner})$
Ext. Ercegovac-Lang Evaluate & Apply	$A_{eval} + 2A_{((\theta_1 \pm \theta_2)/2)} + A_{afc} + A_{mul} + 4A_{app(n)} + 8A_{div/conv}$
Outer Apply	$2A_{(\theta_1 \pm \theta_2)} + 2A_{afc} + 4A_{app(n+2)} + 8A_{div} + 8A_{add/conv}$
Outer Evaluate & Apply	$A_{eval} + 2A_{((\theta_1 \pm \theta_2)/2)} + 2A_{afc} + 4A_{app(n)} + 8A_{div} + 8A_{add/conv}$
Inner Apply	$4A_{(\theta_1 \pm \theta_2)} + 4A_{afc} + 4A_{app(n+2)} + 8A_{div/conv}$
Inner Evaluate & Apply	$A_{eval} + 2A_{((\theta_1 \pm \theta_2)/2)} + 2A_{afc} + 4A_{app(n)} + 8A_{div} + 8A_{add/conv}$

additional concern is the fact that the conventional on-the-fly conversion used in the case of the variable scale factor organizations cannot be used and a solution was proposed in [23]. For further details on the on-fly-conversion scheme and for more on the area/time complexities of CFR-CORDIC implementations, we refer the reader to [22], [23]. Also, it was suggested by one of the referees that an adequate repetition of iterations could be used to make the scale factor much simpler and the division after straightforward.

V. SUMMARY AND CONCLUSION

The paper may be summarized as follows. A formulation of two-sided 2×2 unitary transformations termed the $\mathcal{Q}$ transformation was introduced. The $\mathcal{Q}$ transformation poses the required computations in terms of CORDIC primitives and three typically desired transformations in the context of Jacobi-like methods for computing the eigenvalues/SVD of Hermitian/arbitrary matrices were shown to be implementable as $\mathcal{Q}$ transformations. Potential imbalance in computation time in arrays implementing Jacobi-like algorithms (diagonal vs. off-diagonal processing time) due to the overhead of the evaluation of parameters for the $\mathcal{Q}$ transformations was circumvented by integrating the evaluation of parameters with application. Issues relating to the use of CORDIC for complex arithmetic were addressed and implementation in conventional (nonredundant) CORDIC was presented. Alternatives in both variable scale factor and constant factor redundant (CFR) CORDIC implementations were proposed. A novel on-line scheme for the summation/difference of CORDIC encoded angles was shown to be a critical operation in reducing computation time. An architecture with four conventional (nonredundant) CORDIC modules was shown to require time equal to two full CORDIC operations to evaluate and/or apply the $\mathcal{Q}$ transformations detailed in the paper. For the redundant and on-line CORDIC organizations, the initial direct approach required $\approx 10n$ time and a more area-intensive solution accomplished the transformations in $\approx 6n$, where n is the data precision in bits.

From the discussion on the evaluation and application of two-sided rotational transformations (outer transformations), it may be concluded that the SVD of a real 2×2 matrix may be computed in $\approx 3n$ time as opposed to the $\approx 5n$ time previously reported in the literature. The on-line summation/difference scheme presented in the paper could prove useful in other applications of redundant and on-line CORDIC. Also, other two-sided 2×2 unitary transformations may be cast as $\mathcal{Q}$

transformations to avail of the approaches presented in this paper.

APPENDIX A SELECTION FUNCTION FOR ON-LINE SUM/DIFFERENCE OF ANGLES

We show how the selection function for an on-line computation of γ_j in (8) is realized and the method is similar to Appendix B in [13]. Recall the recurrences

$$\begin{aligned} w[j] &= z[j] + \sigma_{j+p} 2^j \tan^{-1}(2^{-(j+p+1)}) \\ z[j+1] &= 2(w[j] - \gamma_j 2^j \tan^{-1}(2^{-j})). \end{aligned}$$

The intervals $[L_k, U_k]$ of $w[j]$ need to be determined so that $z[j+1]$ remains bounded when choosing $\gamma_j = k$. From the recurrence equations and the range of σ_j we may write

$$\begin{aligned} U_1 - 2[2^j \tan^{-1}(2^{-(j+p+1)})] &= 2(U_k - k 2^j \tan^{-1}(2^{-j})) \\ -U_1 + 2[2^j \tan^{-1}(2^{-(j+p+1)})] &= 2(L_k - k 2^j \tan^{-1}(2^{-j})). \end{aligned}$$

Substituting $k = 1, 0, -1$ successively in the above relations we get

$$\begin{aligned} U_1 &= 2^{j+1} (\tan^{-1}(2^{-j}) - \tan^{-1}(2^{-(j+p+1)})) \\ L_1 &= 2^{j+1} \tan^{-1}(2^{-(j+p+1)}) \\ U_0 &= 2^j (\tan^{-1}(2^{-j}) - 2 \tan^{-1}(2^{-(j+p+1)})) \\ L_0 &= -U_0, \quad U_{-1} = -L_1, \quad L_{-1} = -U_1. \end{aligned}$$

To eliminate the dependence on j , we determine the bounds

$$\begin{aligned} L_1 &\leq 2^{-p} \\ U_0 &\geq \pi/4 - 2 \tan^{-1}(2^{-(p+1)}). \end{aligned}$$

To ensure a positive overlap ($U_0 > L_1$), $p \geq 2$. This implies $L_1 \leq 2^{-2}$, $U_0 \geq 2^{-1}$. It is therefore possible to use an estimate of $w[j]$ with two fractional bits and a suitable selection function for carry-save adders may be chosen as

$$\gamma_j = \begin{cases} 1 & \text{if } \hat{w}[j] \geq 1/2 \\ 0 & \text{if } -1/2 \leq \hat{w}[j] \leq 1/4 \\ -1 & \text{if } \hat{w}[j] \leq -3/4 \end{cases}$$

APPENDIX B

REAL 2×2 MATRIX DIAGONALIZATION SCHEMES

This appendix details the methods reported in the literature for the diagonalization of a real 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, using a two-sided rotational transformation

$$R^T(\theta_l) \begin{bmatrix} a & b \\ c & d \end{bmatrix} R(\theta_r) = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix},$$

where $R(\theta)$ is given by $\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$.

A. Direct Two-Angle Method

In this method, two angles, θ_{sum} and θ_{diff} are computed as

$$\theta_{\text{sum}} = \tan^{-1} \left(\frac{c+b}{d-a} \right), \quad \theta_{\text{diff}} = \tan^{-1} \left(\frac{c-b}{d+a} \right).$$

These angles are related to the required angles θ_l and θ_r by

$$\theta_l = \frac{(\theta_{\text{sum}} - \theta_{\text{diff}})}{2}, \quad \theta_r = \frac{(\theta_{\text{sum}} + \theta_{\text{diff}})}{2}.$$

B. 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 [30].

Evaluation and Application: First, the following are computed from the input matrix:

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

Then, two vectoring (CORDIC y -reduction) operations are performed on (p_1, q_1) and (p_2, q_2) , yielding

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

The required angles θ_l and θ_r , and the results of the diagonalization σ_1 and σ_2 , are then computed from

$$\theta_l = \frac{(\theta_+ - \theta_-)}{2}, \theta_r = \frac{(\theta_+ + \theta_-)}{2}, \\ \sigma_1 = r_1 - r_2, \sigma_2 = r_1 + r_2.$$

Two-Sided Rotation: A variation of the above method may be used when the parameters θ_l and θ_r are known and an application of the two-sided rotational transformation to $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, yielding

$$R^T(\theta_l) \begin{bmatrix} a & b \\ c & d \end{bmatrix} R(\theta_r) = \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix},$$

is desired. 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 from θ_l and θ_r by

$$\theta_- = (\theta_r - \theta_l), \theta_+ = (\theta_r + \theta_l),$$

and two vector rotations (CORDIC z -reductions)

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

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

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

ACKNOWLEDGMENT

The authors wish to thank the anonymous referees for their valuable comments and suggestions.

REFERENCES

- [1] H. M. Ahmed, J. M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *IEEE Comput.*, vol. 15, no. 1, pp. 65-82, 1982.
- [2] R. P. Brent and F. T. Luk, "The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays," *SIAM J. Sci. Stat. Comput.*, vol. 6, no. 1, pp. 69-84, 1985.
- [3] R. P. Brent, F. T. Luk and C. F. Van Loan, "Computation of the singular value decomposition using mesh-connected processors," *J. VLSI Comput. Syst.*, vol. 1, no. 3, pp. 242-270, 1985.
- [4] R. L. Causey, "Computing eigenvalues of non-Hermitian matrices by methods of Jacobi type," *J. Soc. Indust. Appl. Math.*, vol. 6, no. 2, pp. 172-181, 1958.
- [5] 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. New York: Elsevier, 1991, pp. 227-239.
- [6] J. R. Cavallaro and F. T. Luk, "CORDIC arithmetic for an SVD processor," *J. Parallel Distrib. Comput.*, vol. 5, no. 3, pp. 271-290, 1988.
- [7] D. Daggett, "Decimal-binary conversions in CORDIC," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 335-339, 1959.
- [8] J. M. Delosme, "A processor for two-dimensional symmetric eigenvalue and singular value arrays," in *IEEE 21th Asilomar Conf. Circuits, Syst. and Comput.*, 1987, pp. 217-221.
- [9] —, "Parallel implementations of the SVD using implicit CORDIC arithmetic," in *SVD and Signal Processing II (Algorithms, Analysis and Applications)*, R. Vaccaro, Ed. New York: Elsevier, 1991, pp. 33-56.
- [10] E. F. Deprettere and A.-J. van der Veen, "Parallel VLSI matrix pencil algorithm for high resolution direction finding," *IEEE Trans. Signal Processing*, vol. 39, pp. 383-394, 1991.
- [11] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 993-1001, 1974.
- [12] P. J. Eberlein, "On the Schur decomposition of a matrix for parallel computation," *IEEE Trans. Comput.*, vol. C-36, no. 2, pp. 167-174, 1987.
- [13] M. D. Ercegovic and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 725-740, 1990.
- [14] G. E. Forsythe and P. Henrici, "The cyclic Jacobi method for computing the principal values of a complex matrix," *Trans. Amer. Math. Soc.*, vol. 94, no. 1, pp. 1-23, 1960.
- [15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, second ed. Baltimore, MD: Johns Hopkins Univ. Press, 1989.
- [16] G. L. Haviland and A. A. Tuszynski, "A CORDIC arithmetic processor chip," *IEEE Trans. Comput.*, vol. C-29, no. 2, pp. 68-79, 1980.
- [17] N. D. Hemkumar and J. R. Cavallaro, "An efficient parallel implementation of the Jacobi SVD algorithm for arbitrary matrices," Preprint 93-077, AHPARC, Univ. Minnesota, Minneapolis, MN, Aug. 1993. *SIAM J. Matrix Anal. Appl.*, (in second review).
- [18] —, "Efficient complex matrix transformations with CORDIC," in *Proc. IEEE 11th Symp. Comput. Arithmetic*, Windsor, Canada, June 1993, pp. 122-129.
- [19] S. Hitotumatu, "Complex arithmetic through CORDIC," *Kodai Math. Sem. Rep.*, vol. 26, pp. 176-186, 1975.
- [20] X. Hu, R. G. Harber, and S. C. Bass, "Expanding the range of convergence of the CORDIC algorithm," *IEEE Trans. Comput.*, vol. 40, no. 1, pp. 13-21, 1991.
- [21] R. Kumaresan and D. W. Tufts, "Singular value decomposition and improved frequency estimation using linear prediction," *IEEE Trans.*

- Acoust., Speech, Signal Processing*, vol. ASSP-30, no. 4, pp. 671-675, 1982.
- [22] J. Lee and T. Lang, "SVD by constant-factor-redundant CORDIC," in *Proc. IEEE 10th Symp. Comput. Arithmetic*, Grenoble, France, June 1991, pp. 264-271.
 - [23] ———, "Constant-factor redundant CORDIC for angle calculation and rotation," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 1016-1025, 1992.
 - [24] F. T. Luk, "A triangular processor array for computing singular values," *J. Linear Algebra Appl.*, vol. 77, pp. 259-273, 1986.
 - [25] J. M. Speiser, "Signal processing computational needs," in *Proc. SPIE Advanced Algorithms and Architectures for Signal Processing I*, 1986, vol. 696, pp. 2-6.
 - [26] G. W. Stewart, "A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian Matrix," *SIAM J. Sci. Stat. Comput.*, vol. 6, no. 4, pp. 853-864, 1985.
 - [27] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Comput.*, vol. 40, no. 9, pp. 989-995, 1991.
 - [28] J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Comput.*, vol. EC-8, no. 3, pp. 330-334, 1959.
 - [29] J. S. Walther, "A unified algorithm for elementary functions," *AFIPS Spring Joint Comput. Conf.*, pp. 379-385, 1971.
 - [30] B. Yang and J. F. Böhme, "Reducing the computations of the SVD array given by Brent and Luk," *SIAM J. Matrix Anal. Appl.*, vol. 12, no. 4, pp. 713-725, 1991.

Nariankadu D. Hemkumar (S'86) was born in Madras, India on March 11, 1968. He received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1989, the M.S. and Ph.D. degrees, both also in electrical engineering, from Rice University, Houston, TX, in 1991 and 1994, respectively.

In 1994, he joined Crystal Semiconductor Corporation, Austin, TX. His research interests include computer arithmetic, VLSI architectures and algorithms, and synthesis and design for fault tolerance of array processors. He is a recipient of the Govt. of India National Talent Scholarship and the Larsen & Toubro (India) Scholarship, and is a member of Eta Kappa Nu.

Joseph R. Cavallaro (S'78-M'82) was born in Philadelphia, PA on August 22, 1959. He received the B.S. degree from the University of Pennsylvania, Philadelphia, PA, in 1981, the M.S. degree from Princeton University, Princeton, NJ, in 1982, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1988, all in electrical engineering.

From 1981 to 1983, he was with AT&T Bell Laboratories, Holmdel, NJ. In 1988, he joined the faculty of Rice University, Houston, TX, where he is an Associate Professor of Electrical and

Computer Engineering. His research interests include computer arithmetic, fault tolerance, VLSI design and microlithography, and VLSI architectures and algorithms for parallel processing and robotics.

Dr. Cavallaro is a recipient of the NSF Research Initiation Award 1989-1992 and the IBM Graduate Fellowship 1987-1988, and is a member of Tau Beta Pi and Eta Kappa Nu.