A Systolic VLSI Architecture for Complex SVD

Nariankadu D. Hemkumar
Joseph R. Cavallaro

CRPC-TR92227
August 1992

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

Proceedings of the 1992 IEEE International Symposium on Circuits and Systems, San Diego, CA (May 1992)

A Systolic VLSI Architecture for Complex SVD

Nariankadu D. Hemkumar Joseph R. Cavallaro

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

Abstract

A systolic algorithm for the SVD of arbitrary complex matrices, based on the cyclic Jacobi method with "parallel ordering" is presented. A novel two-step, two-sided unitary transformation scheme, tailored to the use of CORDIC algorithms for high speed arithmetic, is employed to diagonalize a complex 2×2 matrix. Architecturally, the complex SVD array is modeled on the Brent-Luk-VanLoan array for real SVD. An expandable array of $O(n^2)$ complex 2×2 matrix processors computes the SVD of an $n \times n$ matrix in $O(n \log n)$ time. A CORDIC architecture for the complex 2×2 processor with an area complexity twice that of a real 2×2 processor is proposed. Computation time for the complex SVD array is less than three times that for a real SVD array with a similar CORDIC based implementation.

1 Introduction

Real-time signal processing concerns combined with the advent of parallel algorithms and architectures, have pushed systolic arrays to the forefront of special-purpose computing. The Singular Value Decomposition (SVD) is an important matrix factorization procedure used extensively in signal and image processing algorithms. While most systolic arrays proposed for the SVD in the literature assume real input matrices, complex data matrices do occur in practice. In particular, several adaptive beam-forming algorithms [6] which determine the direction or bearing of a signal source, require complex matrix factorizations and can benefit from a complex SVD array.

In this paper, a systolic algorithm for the SVD of an arbitrary complex matrix is presented along with a CORDIC (COordinate Rotation DIgital Computer) based VLSI architecture for the systolic processor element. The expandable array structure of the Brent-Luk-VanLoan systolic array [1] with the 2×2 processors is preserved. The Jacobi-SVD method with a novel two-step, twelve-angle two-sided rotation scheme for the diagonalization of arbitrary 2×2 matrices is employed to compute the SVD of larger matrices. A new systolic scheduling scheme is proposed to improve the performance of the array by 50%. An area/time complexity analysis for either array is compared.

2 SVD and Jacobi Methods

A singular value decomposition (SVD) of a matrix $M \in C^{m \times n}$ is given by

$$M = U \Sigma V^H , \tag{1}$$

where $U \in C^{m \times m}$ and $V \in C^{n \times n}$ are unitary matrices and $\Sigma \in R^{m \times n}$ is a real non-negative "diagonal" matrix. Since $M^H = V \Sigma^T U^H$, we may assume $m \geq n$ without loss of generality.

Jacobi-type procedures are extremely amenable to parallel computation as evidenced by the number of such schemes that have been proposed [1]. In the classical Jacobi eigenvalue procedure as extended to the SVD of a square matrix $M \in C^{n \times n}$, the matrix is diagonalized via a sequence of 2×2 SVDs. Corresponding to the eigenvalue decomposition of a symmetric 2×2 matrix, a similar scheme needs to be devised for the SVD of an arbitrary 2×2 matrix.

In the context of VLSI architectures, the various methods proposed in the literature for the SVD of a complex 2×2 matrix have shortcomings [5]. They are either, too cumbersome to implement in special-purpose VLSI using CORDIC or traditional arithmetic units like the Forsythe-Henrici scheme [4], or they assume a specialized matrix structure as in the Deprettere-Van der Veen scheme [7]. The scheme due to Cavallaro-Elster [2], though implementable in CORDIC, does not efficiently adapt to systolic computation.

Ideally, a one-step diagonalization procedure for the complex 2×2 problem is desirable. It allows the straightforward adaptation of the array architectures that have been proposed for the real SVD problem, like the Brent-Luk-VanLoan systolic array, since a real 2×2 matrix can be diagonalized by a single two-sided rotation. Also, from the discussion of the diagonalization of an arbitrary complex 2×2 matrix M in [4], any simplification will require at least one additional transformation step for diagonalization. The structuring of a two-sided transformation for efficient VLSI implementation using CORDIC was motivated primarily by these considerations.

Defining a Q transformation as

$$\begin{bmatrix} c_\phi e^{i\theta_\alpha} & -s_\phi e^{i\theta_\beta} \\ s_\phi e^{i\theta_\alpha} & c_\phi e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} c_\psi e^{i\theta_\gamma} & s_\psi e^{i\theta_\gamma} \\ -s_\psi e^{i\theta_\delta} & c_\psi e^{i\theta_\delta} \end{bmatrix}, \tag{2}$$

we show that by appropriately choosing the various parameters θ_α, θ_β, θ_γ, θ_δ, θ_ϕ and θ_ψ, an arbitrary complex 2×2 matrix can be diagonalized in two steps. A detailed derivation for the two-step diagonalization scheme is given in [5]. The first Q transformation essentially performs a QR decomposition (QRD) of M. The second completes the diagonalization.

The first Q transformation which performs a QRD is given by

$$\begin{bmatrix} c_\phi e^{i\theta_\alpha} & -s_\phi e^{i\theta_\beta} \\ s_\phi e^{i\theta_\alpha} & c_\phi e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} c_\psi e^{i\theta_\gamma} & s_\psi e^{i\theta_\gamma} \\ -s_\psi e^{i\theta_\delta} & c_\psi e^{i\theta_\delta} \end{bmatrix} = \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix}, \tag{3}$$

where

$$\theta_\alpha = \theta_\beta = \frac{-(\theta_d + \theta_c)}{2}, \; \theta_\gamma = -\theta_\delta = \frac{(\theta_d - \theta_c)}{2},$$

$$\theta_\phi = 0, \text{ and } \theta_\psi = \tan^{-1}\left(\frac{C}{D}\right), \tag{4}$$

and the second Q transformation which completes the diagonalization can be written as

$$\begin{bmatrix} c_\lambda e^{i\theta_\ell} & -s_\lambda e^{i\theta_\eta} \\ s_\lambda e^{i\theta_\ell} & c_\lambda e^{i\theta_\eta} \end{bmatrix} \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix} \begin{bmatrix} c_\rho e^{i\theta_\zeta} & s_\rho e^{i\theta_\zeta} \\ -s_\rho e^{i\theta_w} & c_\rho e^{i\theta_w} \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}, \tag{5}$$

0-7803-0593-0/92 $3.00 1992 IEEE

where

$$\theta_\xi = -\left(\frac{\theta_x + \theta_w}{2}\right), \quad \theta_\eta = \theta_\zeta = \left(\frac{\theta_x - \theta_w}{2}\right),$$

$$\theta_\omega = \left(\frac{\theta_w - \theta_x}{2}\right), \quad \tan(\theta_\lambda \pm \theta_\rho) = -\left(\frac{X}{Z \mp W}\right). \quad (6)$$

SVD algorithms require costly arithmetic operations such as division and square root in the computation of rotation parameters. Increased efficiency may be obtained through the use of hardware oriented arithmetic techniques that relate better to the algorithm. The CORDIC [8, 9] algorithms, which allow easy computation of inverse tangents and vector rotations, have proven extremely useful in this context [3].

The $\mathcal{Q}$ transformation in (2) may be rewritten as

$$\begin{bmatrix} c_\phi & -s_\phi \\ s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} e^{i\theta_\alpha} & 0 \\ 0 & e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} e^{i\theta_\gamma} & 0 \\ 0 & e^{i\theta_\delta} \end{bmatrix} \begin{bmatrix} c_\psi & s_\psi \\ -s_\psi & c_\psi \end{bmatrix}, \quad (7)$$

to hint at a procedure for the use of CORDIC in the application of the $\mathcal{Q}$ transformation. As indicated by (7), the $\mathcal{Q}$ transformation can be performed in two steps. The inner two-sided unitary transformation is completed first, and followed by the two-sided rotational transformation. The inner unitary transformation, essentially affects only the arguments of the complex data elements.

We assume that the complex data elements of the input matrix are represented in orthogonal coordinates. Since the inner transformation affects only the arguments of the complex data, it is convenient to compute the polar coordinate representations of the data elements. The unitary transformation angles can then be used to modify the arguments appropriately. A transformation back to the orthogonal coordinate system completes the unitary transformation.

Once the inner rotation of the $\mathcal{Q}$ transformation has been completed, the outer two-sided rotation needs to be computed. From the arithmetic of complex numbers, it is easy to observe that the two-sided rotation may be applied independently to the real and imaginary parts of the data elements. The application of the outer transformation, for the real or imaginary parts, involves two vector rotations. The vectors are given by the rows for the left rotation and by the columns for the right rotation. CORDIC scale factor correction is postponed until both the left and right rotations are applied to employ two-sided scale factor correction.

3 A Processor Architecture

In the application of the $\mathcal{Q}$ transformation using CORDIC, a basic CORDIC processor [3], is required for each complex data element. However, additional complexity in control is required to handle extra computations required for complex data manipulations. To achieve maximum parallelism in the application of the $\mathcal{Q}$ transformation for the complex 2×2 problem, four CORDIC modules are needed.

No more than four CORDIC modules are necessary to achieve maximum concurrency in the application of the $\mathcal{Q}$ transformation. This is because each module can independently apply the inner unitary transformation sub-step of the $\mathcal{Q}$ transformation if there exists a mechanism for the broadcast of the necessary rotational angle parameters. Also, adjacent modules (row or column adjacency depending on whether the right or the left rotation is being applied) share rotation angles and exchange data in the application of the outer rotational transformation sub-step.

Due to the adjacency in the pattern of communication of data and results of CORDIC iterations between the CORDIC modules, it is more efficient to have register banks shared by neighboring CORDIC modules than a centralized register bank. Furthermore, the results of some CORDIC operations, such as the rotation angles, are needed by all the CORDIC modules. A data bus linking the four register banks is also required.

The proposed CORDIC complex 2×2 processor as shown in Figure 1, is composed of four CORDIC modules and a central control unit. The CORDIC modules are similar to those mentioned in [3], with modifications in the control units to implement the schemes for complex arithmetic using CORDIC [5].

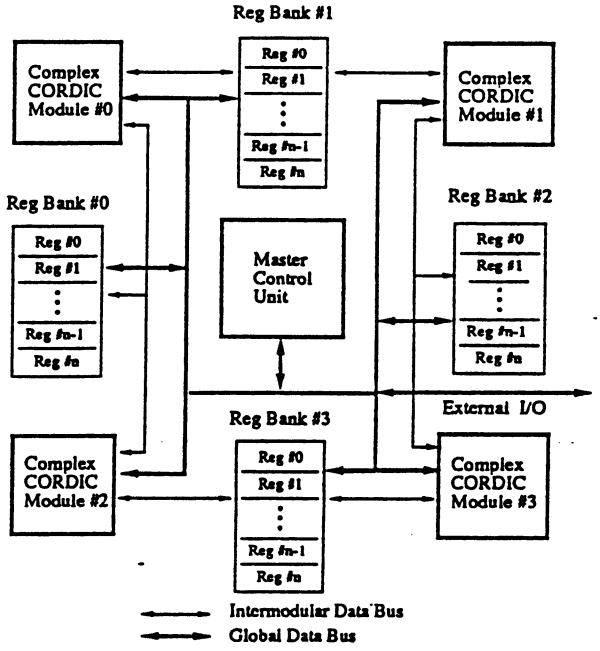


Figure 1: Complex 2×2 Processor Architecture

Let T_{Q1} and T_{Q2}, be the time required to compute the first and second $\mathcal{Q}$ transformations in the computation of the SVD. The time complexity analysis presented below is indicative of the maximum parallelism that can be exploited in the CORDIC SVD algorithm using the architecture of Figure 1. Both T_{Q1} and T_{Q2} can be split up as

$$T_{Q1} = T_{Q2} = T_{PT} + T_{IT} + T_{CR} + 2T_{VR} + 2T_{TSFC},$$

where T_{PT}, T_{IT}, T_{CR}, T_{VR} and T_{TSFC}, are the times to compute the polar transformation of a complex number represented in orthogonal coordinates using CORDIC, the inverse tangent, the complex rotation, the vector rotation and the two-sided scale factor correction, respectively. All of the times mentioned above, except for T_{TSFC}, are about the same as the time required to compute one CORDIC vector rotation, T_C [5]. For fixed-point data paths , $T_{TSFC} \approx 0.25T_C$ [3]. Thus,

$$T_{Q1} = T_{Q2} \approx 5.5T_C.$$

The total time complexity of the two-step diagonalization method is therefore

$$T_{CSVD} = T_{Q1} + T_{Q2} \approx 11T_C.$$

The principal components of the complex SVD processor are the four CORDIC modules, each of which requires a barrel shifter of $O(b^2)$ area[1] with three adders, angle ROM, control and moderately complex interconnection buses. Each CORDIC module is similar to a basic CORDIC processor of area A_C. The four

[1] b = word length in bits

register banks which serve as storage for the CORDIC modules and communication of data and angles between the processors and/or modules and the central control unit complete the list of major components in terms of area requirement. If A_{REG} is the area requirement for each of the register banks and $A_{CONTROL}$ is that of the control unit, the area complexity of the complex CORDIC SVD processor, can be expressed as

$$A_{CSVD} \approx 4A_C + 4A_{REG} + A_{CONTROL}.$$

4 A Systolic Array for CSVD

The complex SVD array is an expandable, mesh-connected array of processors, where each processor contains a 2×2 sub-matrix of the input matrix $M \in C^{n \times n}$. Assuming that n is even, the complex SVD array is a square array of $n/2 \times n/2$ processors. Before the computation begins, processor P_{ij} contains

$$\begin{bmatrix} m_{2i-1,2j-1} & m_{2i-1,2j} \\ m_{2j,2j-1} & m_{2i,2j} \end{bmatrix}$$

where $(i, j = 1, \ldots, \frac{n}{2})$. Each processor P_{ij} is connected to its "diagonally" nearest neighbors $P_{i\pm1,j\pm1}$, $(1 < i, j < \frac{n}{2})$. The complex SVD array, with 16 processors for $n = 8$, is shown in Figure 2. The interconnections between the processors facilitate data exchange to implement the "parallel ordering" of Brent-Luk.

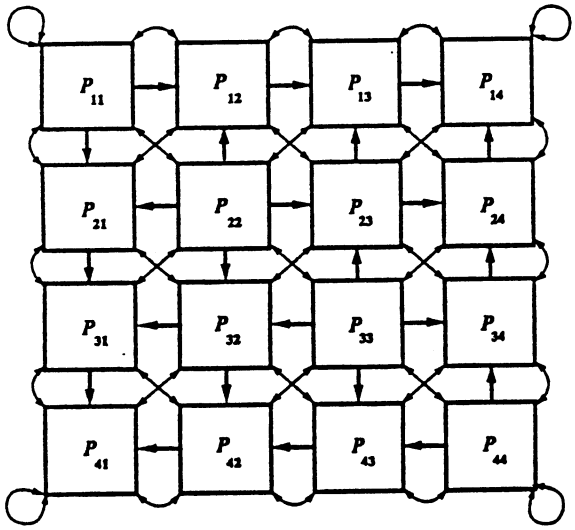


Figure 2: The Complex SVD Array

The Brent-Luk "parallel ordering" permits Jacobi transformations to be applied, in parallel, in groups of $n/2$. The parameters for the $n/2$ Jacobi transformations are generated by the $n/2$ processors on the main diagonal of the array. The diagonal processors P_{ii} $(i = 1, \ldots, \frac{n}{2})$ in the array have a more important role in the computation of the SVD when compared to the off-diagonal processors P_{ij} $(i \neq j, 1 \leq i, j \leq \frac{n}{2})$.

The application of a two-sided Jacobi transformation affects only the row and column of the diagonal processor generating the transformation. In an idealized situation, the diagonal processors may broadcast the parameters along the row and the column corresponding to their position in the array, in constant time. Each off-diagonal processor applies a two-sided transformation using the rotation angle parameters generated by the diagonal processors in the same row and column with respect to its location in the array.

It cannot realistically be expected that the transformation parameters be broadcast in constant time for any array size. However, by assuming that the transformation parameters can be transmitted between adjacent processors in constant time, Brent-Luk-VanLoan specify a scheme to stagger computations that precludes the need for broadcast of rotation parameters, but still completes a sweep of the "parallel ordering" in $O(n)$ time.

The complex SVD array as described so far is identical to the Brent-Luk-VanLoan systolic array in all respects except that the Brent-Luk-VanLoan systolic array diagonalizes a real 2×2 matrix as opposed to arbitrary 2×2 matrices in the complex SVD array. With the use of the two-step diagonalization scheme proposed in this paper and the CORDIC implementation of the scheme, it is clear that the number of transformation parameters (Eqs. 4,6) that are generated at each step of the Jacobi-SVD method is significantly greater than the real SVD case[2]. The two-step diagonalization scheme is composed of two Q transformations each of which requires the generation and application of six angles, four unitary angles and two rotational angles. Thus, the total count of the angles amounts to twelve in all; eight unitary and four rotational angles.

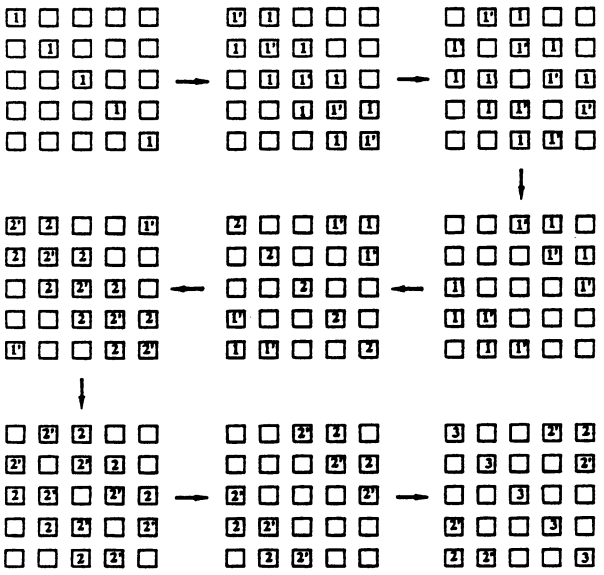
In a direct adaptation of the Brent-Luk-VanLoan systolic array for systolic computation of the SVD of a complex matrix, six angles must be propagated along both the rows and columns of processors on the main diagonal. These processors are responsible for the generation of the angles, in addition to applying them to diagonalize the 2×2 matrices stored on them. Also, while the diagonal processors are computing the second Q transformation, the immediately off-diagonal processors are idle; even though the rotational parameters needed for the application of the first Q transformation are available at the diagonal processors.

The identity of the steps in the two-step diagonalization scheme was intended to prompt an overlapping of computation across the array. The novel scheme proposed then, is to chase the first and second Q transformations down the diagonals, one behind the other as shown in Figure 3. Thus, while the processors on the main diagonal are still computing the parameters for the second Q transformation step, the immediately off-diagonal processors are applying the first Q transformation. The added systolicity and pipelining due to this new scheduling of computations, improves performance and reduces the communication load per step for the propagation of rotational parameters through the array. The processor utilization increases from 33% to 50% due to the fact that processors along any diagonal are now active twice every four computation steps as opposed to once every three computation steps.

In comparing the relative performance of the Brent-Luk-VanLoan systolic array and the complex SVD array, the convergence of the SVD-Jacobi method and the time to compute a sweep of the "parallel ordering" are the determining factors. A measure of the computational speed of the two arrays is the time required for processors on the main diagonal to start processing new data after completing a diagonalization. These times for the different SVD arrays are tabulated in Table 1, where Δ_{steps} refers to the number of computation steps before the main diagonal starts processing data again.

A direct scheme for the computation of the SVD for a real 2×2 matrix using CORDIC was shown to require 3.25 T_C [3]. From the discussion of the area/time complexity of the complex CORDIC SVD processor we know that a Q transformation requires 5.5 T_C. Again, both the times mentioned above include the time to generate the respective rotation parameters involved.

[2]Only two rotation angles are needed for real diagonalization



X -> First Q Transformation of Step X
X' -> Second Q Transformation of Step X

Figure 3: Overlapped Computations on the CSVD Array (Snapshots of activity on a 5 × 5 Array)

In all of the systolic arrays listed in Table 1, generation of the rotational parameters is done only along the main diagonal. For the direct scheme (C.SVD (B)), the processors along any diagonal are active once every three computation steps while in the pipelined scheme (C.SVD (P)) the same processors are active twice every four steps. The last column of Table 1 shows the time to complete a sweep of the "parallel ordering" relative to the real SVD array (R.SVD). The pipelining of computations in the complex SVD array allows 11.0/33.0 = 33%, saving in computation time.

Array	Δ_{steps}	T_C	Rel. Perf.
R. SVD	3	9.75 T_C	1.0 T_{RSVD}
C. SVD (B)	3	33.0 T_C	3.38 T_{RSVD}
C. SVD (P)	4	22.0 T_C	2.26 T_{RSVD}

Table 1: Relative Performance of SVD Arrays

The estimates of the relative speed of the real and complex SVD arrays depend on the rate of convergence for the SVD-Jacobi scheme and nature of the matrix data (real or complex). A simulation of the complex SVD array was performed on the Connection Machine [5] to observe the convergence behavior in terms of the number of sweeps required for a given matrix (array) size. A similar study was also performed by Brent, Luk and VanLoan [1] to observe the convergence rate for the real SVD using "parallel ordering" and the two-sided rotation scheme for diagonalizing a real 2 × 2 matrix. The convergence behavior of the SVD-Jacobi method with "parallel ordering" for real data matrices is given in [1]. The number of sweeps required for the convergence of the complex SVD scheme is greater than that for the real SVD, but the convergence behavior is identical. Also, since the time required to complete a sweep in the complex SVD array is 2.26 times greater than the real SVD array (Table 1), the overall time for computation is less than three times that for the real SVD array.

5 Conclusions

In this paper, a novel VLSI hardware oriented two-step diagonalization scheme for the SVD of a complex 2 × 2 matrix, the basic step in a Jacobi-type procedure for the computation of the SVD, was presented. Each step in the scheme was a two-sided unitary transformation (Q transformation), designed to be efficiently implementable in hardware using CORDIC. A systolic array similar in structure to the Brent-Luk-VanLoan systolic array, with an enhanced scheduling and data exchange algorithm designed to efficiently implement the two-step diagonalization scheme was proposed for the solution of the complex SVD problem. The behavior of a processor in the complex SVD array, illustrating the systolic computation, was detailed. An architecture for the complex 2 × 2 processor, exploiting the parallelism in the CORDIC implementation of the two-step diagonalization scheme, was presented. In spite of the involved nature of computations in diagonalizing a complex 2 × 2 matrix, the time for completing a sweep in the complex CORDIC SVD array is less than three times that for a CORDIC based implementation of the real SVD array.

6 Acknowledgments

Use of the Connection Machine was provided by the Center for Research on Parallel Computation under NSF Cooperative Agreement No. CCR-8809615 with support from the Keck Foundation and Thinking Machines Corporation. This work was supported in part by the National Science Foundation under Research Initiation Award MIP-8909498.

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*, 1(3):242–270, 1985.

[2] J. R. Cavallaro and A. C. Elster. A CORDIC Processor Array for the SVD of a Complex Matrix. In R.J. Vaccaro, editor, *SVD and Signal Processing II (Algorithms, Analysis and Applications)*, pages 227–239. Elsevier, New York, 1991.

[3] J. R. Cavallaro and F. T. Luk. CORDIC Arithmetic for an SVD Processor. *Journal of Parallel and Distributed Computing*, 5(3):271–290, June 1988.

[4] G. E. Forsythe and P. Henrici. The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix. *Transactions of the American Mathematical Society*, 94(1):1–23, January 1960.

[5] N. D. Hemkumar. A Systolic VLSI Architecture for Complex SVD. Master's thesis, Rice University, Department of Electrical and Computer Engineering, April 1991.

[6] C. M. Rader. Wafer-Scale Systolic Array for Adaptive Antenna Processing. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 2069–2071, April 1988.

[7] A. J. Van der Veen and E. F. Deprettere. A Parallel VLSI Direction Finding Algorithm. *Proc. SPIE Advanced Algorithms and Architectures for Signal Processing*, 975(III):289–299, August 1988.

[8] J. Volder. The CORDIC Trigonometric Computing Technique. *IRE Trans. Electronic Computers*, EC-8(3):330–334, Sept. 1959.

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