A proposed efficient architecture for OFDM MIMO systems using SVD

Naveen Rathee
Department of Electronics and Communication Engineering
CMJ University, Shilong

Dr.S.C.Gupta
Department of Electronics and Communication Engineering
RPIIT, Batsara, Haryana, India

Abstract- This paper presents a technique for Antenna beam forming in high data rate OFDM-MIMO systems. The technique makes use of the Singular Value Decomposition (SVD) algorithm for matrix decomposition using Givens Rotation. In this paper a hardware oriented two-step diagonalization SVD scheme is derived from simple two-sided unitary transformations developed to ensure hardware and performance efficiency using CORDIC. Each unitary transformation step in the diagonalization procedure is identical in structure to permit pipelined systolic execution. The simulation results are obtained for fixed point models using SVD algorithm. An overall architecture is created in Matlab for 4X4 Complex valued matrices elements and then simulated. The VHDL code is then written to describe the architecture of the overall design and is the synthesized using Xilinx ISE 10.1 software for virtex-4 target device.

Keywords- OFDM, MIMO, SVD, CORDIC.

I.INTRODUCTION

Orthogonal Frequency Division Multiplexing (OFDM) is a popular method for high-rate data transmission in wireless environments. In OFDM, the channel bandwidth is divided into several narrow sub bands. The frequency response over each of these sub bands is flat. Hence, a frequency-selective channel is transformed into several flat-fading sub channels. The time domain waveforms of the subcarriers are orthogonal, yet the signal spectra corresponding to different subcarriers overlap in frequency. Therefore, the available bandwidth is used very efficiently. The data rate of the system is aggregate of the data rate per sub channel. These features make OFDM suitable for high data rate applications. Another advantage of OFDM systems is that they are less susceptible to various kinds of impulse noise. These characteristics result in reduced receiver complexity. MIMO (Multiple Input Multiple Output) systems use multiple antennas equalize the received signal to remove the effect of channel on the signal. Most of equalization detection algorithms need to invert a matrix which is either the channel state information (H) or a nonlinear function of it (f (H)). Increasing the number of transmitter and receiver antennas in the system, results in a higher data rate. At the same time, dimensions of matrix f (H) increase, requiring more computations to invert the matrix in less time. This makes the matrix inversion block a bottleneck in these systems. Matrix decomposition algorithms [1], such as the singular value decomposition (SVD) or the QR Decomposition (QRD), have applications in various signal processing fields. The SVD, for example, is used in array processing or data compression, but can also be applied to MIMO systems in order to increase the system performance by the use of beamforming and power allocation. The QRD, for example, is a key prerequisite for many advanced MIMO detectors, such as the sphere decoder [2]. Both these algorithm decomposition techniques are mainly base on a specific sequence of Givens rotations [1]. CORDIC (coordinate rotation digital computer) algorithms have shown to be a suitable tool to efficiently perform Givens rotations in hardware [3]. Due to the relatively high computational complexity of the SVD, systolic arrays based on the Jacobi method have been proposed [3]–[5]. However, in MIMO-OFDM systems [6] for example, multiple problems need to be solved concurrently, where the number of parallel tasks corresponds to the number of OFDM tones. The throughput of fast but large architectures (e.g., systolic arrays) is often difficult to match to an arbitrary number of problems, e.g., one systolic array might be insufficient in terms of throughput but two might exceed the available circuit area.

II.SVD ALGORITHM

SINGULAR VALUE DECOMPOSITION

Singular value decomposition (SVD) of a matrix $M \in C^{mxn}$ is given by $M = U\Sigma V^H$ Where $U \in C^{mxm}$and $V \in C^{mxn}$are unitary matrices and $\sum \in R^{mxn}$ *is a real nonnegative* diagonal" matrix. Since $M^H = V\Sigma^T U^H$, we may assume m ≥ n without loss of generality. The singular values may be arranged in any order, for if $P \in R^{mxm}$ and Q $\in R^{nxn}$ are permutation matrices such that PΣQ remains "diagonal", then

M = (UP^T) (PΣQ) $(Q^T V^H)$ is also an SVD. It is customary to choose P and Q so that the singular values are arranged in non-increasing order:

σ1 ≥ ….. ≥ σr > 0, σr+1 = _ _ _ = 0, Where r = rank (M).

If the matrices *U, Σ and V* are partitioned by columns as

U = [u_1, u_2, _ _ _, u_m], *Σ* = diag [σ_1, σ_2, _ __, σ_n] and V = [v_1, v_2, _ _ _, v_n], Then σ_i is the i^{th} singular value of M, and u_i and v_i are the left and right singular vectors corresponding to σ_i. If M is real, then the unitary matrices U and V are real and hence orthogonal.

The SVD procedure under consideration bases on the Golub-Kahan algorithm described in [7] and mainly performs the SVD in two phases:

1) Bidiagonalization: First, a memory is initialized with M = {IM, A, IN} where IL stands for an L × L identity matrix. During the bidiagonalization phase, Givens rotations are successively applied to A from the left-hand side (LHS) and from the right-hand side (RHS), such that the M × N dimensional inner matrix A gets bidiagonal and real-valued (denoted by B0) as illustrated in Fig. All Givens rotations applied to A from the LHS and RHS are applied to the corresponding identity matrices. The resulting unitary matrices are denoted by $\tilde{U}$ and $\tilde{V}^H$ and the memory content after the bidiagonalization phase corresponds to M= {$\tilde{U}$, B0, $\tilde{V}^H$ } where A = $\tilde{U}$B0 $\tilde{V}^H$.

2) Diagonalization: The diagonalization phase consists of multiple diagonalization steps (indicated with k) and is illustrated in Fig. Givens rotations are subsequently applied from the LHS and from the RHS to the bidiagonal matrix B_k such that all off-diagonal entries

f_i (for i = 1, 2, . . . , r − 1) of B_k become zero. The diagonalization phase is stopped whenever all fi are considered to be zero and all d_i (for i = 1, 2, . . . , r) correspond to the unordered singular values. In order to ensure convergence of the diagonalization phase and to reduce the overall computation time of the SVD, the first Givens rotation of each diagonalization step is performed with a modified input vector $[x\ y]^T$, where y = t_{12} and x = t_{11} − μ uses the Wilkinson shift [7].

$$\mu = a_n + c - \text{sign}(c)\sqrt{a_{n+}^2 b_{n-1}^2}$$

With $c = \frac{1}{2}(a_{n-1} - a_n)$, $T B_K^H B_k$, and the trailing non-zero sub matrix of T corresponds to

$$T(n-1:n, n-1:n) = \begin{pmatrix} a_{n-1} & b_{n-1} \\ b_{n-1}^* & a_n \end{pmatrix}$$

analogous to the bidiagonalization phase, all Givens rotations are also applied to the corresponding unitary matrices such that finally, $M = \{U, \Sigma, V^H\}$ is the SVD in (8). 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 in [8, 9, 10].

Bidiagonalization

$$A = \begin{bmatrix} C & C & C \\ C & C & C \\ C & C & C \end{bmatrix} \xrightarrow{LHS} \begin{bmatrix} R & C & C \\ 0 & C & C \\ 0 & C & C \end{bmatrix} \xrightarrow{RHS} \begin{bmatrix} R & R & 0 \\ 0 & C & C \\ 0 & C & C \end{bmatrix} \xrightarrow{LHS} \begin{bmatrix} R & R & 0 \\ 0 & R & C \\ 0 & 0 & C \end{bmatrix} \xrightarrow{RHS} \begin{bmatrix} R & R & 0 \\ 0 & R & R \\ 0 & 0 & R \end{bmatrix} \xrightarrow{LHS} \begin{bmatrix} R & R & 0 \\ 0 & R & R \\ 0 & 0 & R \end{bmatrix} = B_0$$

Diagonalization

$$B_k = \begin{bmatrix} d_1 & f_1 & 0 \\ 0 & d_2 & f_2 \\ 0 & 0 & d_3 \end{bmatrix} = \begin{bmatrix} x & y & 0 \\ 0 & R & R \\ 0 & 0 & R \end{bmatrix} \xrightarrow{RHS} \begin{bmatrix} R & R & 0 \\ R & R & R \\ 0 & 0 & R \end{bmatrix} \xrightarrow{LHS} \begin{bmatrix} R & R & R \\ 0 & R & R \\ 0 & 0 & R \end{bmatrix} \xrightarrow{RHS} \begin{bmatrix} R & R & 0 \\ 0 & R & R \\ 0 & R & R \end{bmatrix} \xrightarrow{LHS} \begin{bmatrix} R & R & 0 \\ 0 & R & R \\ 0 & 0 & R \end{bmatrix} = B_{k+1}$$

Fig.1. Bidiagonalization and Diagonalization phases' Illustration for SVD for a complex valued 3 x 3 matrices [7].

III.CORDIC ALGORITHM AND ARCHITECTURES FOR SVD COMPUTATION

CORDIC Algorithms

The Coordinate Rotation Digital Computer (CORDIC) algorithms allow fast iterative hardware calculation of sin, cos, arctan, sinh, cosh, arctanh, products, quotients, square roots, and conversion between binary and mixed radix number systems. Real-time signal processing concerns [8], combined with the performance and hardware advantage in the VLSI setting, makes CORDIC an attractive alternative to traditional arithmetic units for special-purpose hardware design.

In a conventional sequential computer, the calculation of rotation angles through costly square root and division operations, or computation of sines/cosines in software, proves expensive. Also, matrix-vector products involve costly multiplication and division operations. In the context of special-purpose SVD
architectures, primitive CORDIC operations like vector rotations and inverse tangent calculations help increase efficiency by more effectively mapping the algorithm to hardware [9, 10].

Special VLSI structures have been shown possible for the SVD [11, 12, 13]. For computing the SVD the circular mode of CORDIC is used. This method is very useful for fixed-point calculations. It is a combination of shifts and adds and does not require any multiplications [14],[15],[16],. In the figure 2 of parallel Fixed point CORDIC the three parallel data paths are provided for the x, y and z recurrence equations. The processor is composed of several registers to hold the final and temporary values of x, y and z. For a fixed-point implementation, shifters are used in the x and y data paths to produce multiplication by 2^{-j}**.** A read-only memory (ROM) is used to store the angles for the three CORDIC modes viz., the circular (m = 1), the linear (m = 0) and the hyperbolic (m = -1) modes. Three fixed-point adders provide the additions required in the CORDIC recurrence relations. A control unit oversees the overall sequencing and angle selection depending on which of the six CORDIC operational modes is chosen. The control unit is responsible for choosing between transformation type based on the values in the y and z registers for the y reduction and z-reduction operation modes respectively.

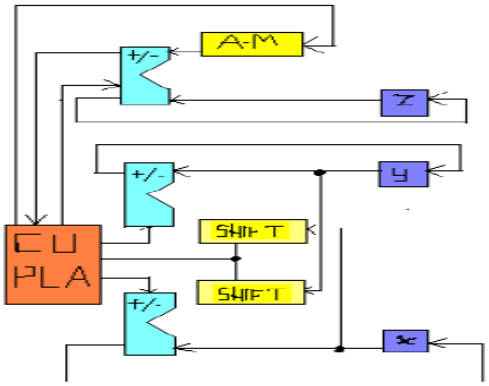

Fig.2.Fixed point CORDIC block diagram

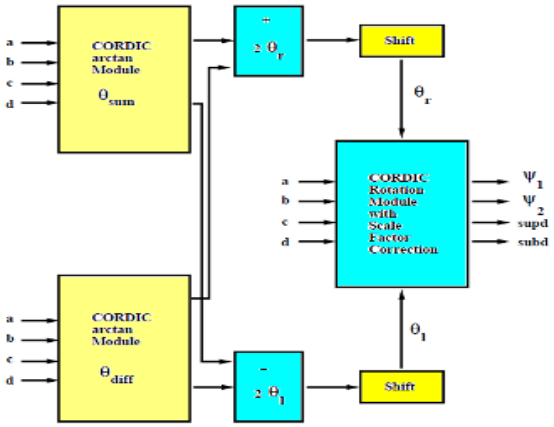

Fig.3. Fixed point SVD using CORDIC block diagram

Transformation types:

For $\delta_j > 0$

$X_{j+1} = x_j + my_j 2^{-j}$

$Y_{j+1} = y_j - x_j 2^{-j}$

$Z_{j+1} = z_j + \beta_j$

For $\delta_j < 0$

$X_{j+1} = x_j - my_j 2^{-j}$

$Y_{j+1} = y_j + x_j 2^{-j}$

$Z_{j+1} = z_j - \beta_j$

IV.TWO SIDED UNITARY TRANSFORMATION

Complex arithmetic and matrix transformations require a significantly greater number of computational steps than the corresponding real operations. A real Givens Rotation is given by

$$\begin{bmatrix} C & S \\ -S & C \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \quad \text{---------- (1)}$$

Where$\theta = tan^{-1}\left(\frac{b}{a}\right)$, c = cosθ, s = sinθ and $r = \sqrt{a^2 + b^2}$.

The Givens Rotation can be generalized to handle the case of complex arithmetic. A Complex Given Rotation can be can be described by two rotation angles as formulated in Coleman and Van Loan [17] as

$$\begin{bmatrix} cos\theta_1 & sin\theta_1\left(e^{i\theta_2}\right) \\ -sin\theta_1\left(e^{-i\theta_2}\right) & cos\theta_1 \end{bmatrix}\begin{bmatrix} a_r + ia_1 \\ b_r + ib_1 \end{bmatrix} = \begin{bmatrix} r_r + ir_1 \\ 0 \end{bmatrix} \quad \text{----------(2)}$$

The above rotation matrix is derived by first applying the simple unitary transformation

$$U = \begin{bmatrix} e^{-i\theta_a} & 0 \\ 0 & e^{-i\theta_b} \end{bmatrix} \quad \text{---------- (3)}$$

to convert the complex numbers to real values. This is followed by a real Givens rotation (1) to zero out the second component. However, in order to avoid four complex rotations, the complex conjugate of (3) is applied to the left side of the real Givens rotation, giving the complex Givens rotation in (2).In the expanded form equation (2) can be written as

$$\begin{bmatrix} e^{-i\theta_a} & 0 \\ 0 & e^{-i\theta_b} \end{bmatrix}\begin{bmatrix} cos\theta_1 & sin\theta_1 \\ -sin\theta_1 & cos\theta_1 \end{bmatrix}\begin{bmatrix} e^{-i\theta_a} & 0 \\ 0 & e^{-i\theta_b} \end{bmatrix} = \begin{bmatrix} cos\theta_1 & sin\theta_1\left(e^{i\theta_2}\right) \\ -sin\theta_1\left(e^{-i\theta_2}\right) & cos\theta_1 \end{bmatrix}$$

The angle θ_1 and θ_2 can be determined from the input vectors as

A =$\sqrt{a_r^2 + a_i^2}$, $\theta_a = tan^{-1}\left(\frac{a_i}{a_r}\right)$

B =$\sqrt{b_r^2 + b_i^2}$, $\theta_b = tan^{-1}\left(\frac{b_i}{b_r}\right)$ ----------(4)

Then from the above angles and radii $\theta_1 = tan^{-1}\left(\frac{B}{A}\right)\ and\ \theta_2 = \theta_a - \theta_b$ ---------- (5)

The complex SVD can be approached by using the SVD-Jacobi algorithm [18]. As a basic step in the complex SVD-Jacobi algorithm, a 2x2 matrix, possibly with complex data elements, is transformed into a real diagonal matrix. Several two-sided unitary transformations have been suggested in the literature [18] to diagonalize a complex 2_2 matrix. Another scheme for the diagonalization of an arbitrary 2x 2 matrix is due to Kogbetliantz [19]. Deprettere and van der Veen [20] considered input matrices with a specialized structure f or a CORDIC SVD array based on the SVD-Jacobi method.

The various methods proposed for the SVD of a complex 2x2 matrix mentioned have shortcomings. They are either, too cumbersome to implement in special-purpose VLSI using CORDIC or traditional arithmetic units. The scheme does not efficiently adapt to systolic computation. Two-sided rotations are used in the diagonalization of a real 2x2 matrix in the Brent-Luk-Van Loan systolic array [21]. In the development of a systolic scheme to diagonalize a complex matrix, it is important to express these methods as two-sided unitary rotations/transformations.

Two Q transformations are sufficient to compute the SVD. The first Q transformation essentially performs a QR decomposition of M. where

$$M \triangleq \begin{bmatrix} a_r + ia_i & b_r + ib_i \\ c_r + ic_i & d_r + id_i \end{bmatrix} = \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \quad \text{---------- (6)}$$

Also as we know that the QR decomposition of m x n matrix $M\epsilon\ C^{mxn}$ is given by $M = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$ ---------- (7)

Using the complex Givens rotation [17] the QR decomposition of a complex 2x2 matrix (6) can be computed. The second completes the diagonalization. To illustrate the steps in the diagonalization, each Q transformation is presented as a combination of sub transformations used. The first sub transformation
is an R transformation which renders the bottom row of M real. It is followed by a real Givens rotation which zeros the lower left (2, 1) element. This completes the first Q transformation which is defined as

$$\begin{bmatrix} C_\phi e^{i\theta_\alpha} & -S_\phi e^{i\theta_\beta} \\ S_\phi e^{i\theta_\alpha} & C_\phi e^{i\theta_\beta} \end{bmatrix} \begin{bmatrix} Ae^{i\theta_a} & Be^{i\theta_b} \\ Ce^{i\theta_c} & De^{i\theta_d} \end{bmatrix} \begin{bmatrix} C_\psi e^{i\theta_\gamma} & S_\psi e^{i\theta_\gamma} \\ -S_\psi e^{i\theta_\delta} & C_\psi e^{i\theta_\delta} \end{bmatrix} = \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix} \quad \text{---------- (8)}$$

Where $\theta_\alpha = \theta_\beta = -\left(\frac{\theta_c+\theta_d}{2}\right), \theta_\gamma = -\theta_\delta = \left(\frac{\theta_d-\theta_c}{2}\right), and\ \theta_\phi = 0, \theta_\psi = tan^{-1}\left(\frac{C}{D}\right)$

Next, a D and I transformation are combined with a two-sided rotation to generate the Q transformation for the second step. The D transformation converts the main diagonal elements to real values and the I transformation takes advantage of the fact that the lower-left element is zero, to convert the upper-right
element to a real value. After the D and I transformations are applied, the 2x2 matrix is real. Thus, the second Q transformation can be written as

$$\begin{bmatrix} C_\lambda e^{i\theta_\xi} & -S_\lambda e^{i\theta_\eta} \\ S_\lambda e^{i\theta_\xi} & C_\lambda e^{i\theta_\eta} \end{bmatrix} \begin{bmatrix} We^{i\theta_w} & Xe^{i\theta_x} \\ 0 & Z \end{bmatrix} \begin{bmatrix} C_\rho e^{i\theta_\zeta} & S_\rho e^{i\theta_\zeta} \\ -S_\rho e^{i\theta_\omega} & C_\rho e^{i\theta_\omega} \end{bmatrix} = \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix} \text{---------- (9)}$$

Unlike the first Q transformation, there are two possible sets of values for the unitary transformation angles. This is due to the fact that the left and right unitary angles in a D transformation are interchangeable.

The choice is between $\theta_\xi = -\left(\frac{\theta_x+\theta_\omega}{2}\right), \theta_\eta = \left(\frac{\theta_x-\theta_\omega}{2}\right)$----------(10)

and $\theta_\zeta = \left(\frac{\theta_x-\theta_\omega}{2}\right), \theta_\omega = \left(\frac{\theta_\omega-\theta_x}{2}\right), and\ \theta_\xi = -\left(\frac{\theta_x}{2}\right), \theta_\eta = \left(\frac{\theta_x}{2}\right)$--------- (11)

$\theta_\zeta = \left(\frac{\theta_x}{2}\right) - \theta_\omega, \theta_\omega = -\left(\frac{\theta_x}{2}\right)$, However, the rotation angles for the second Q transformation are given by

$tan^{(\theta_\lambda-\theta_\rho)} = -\left(\frac{X}{Z-W}\right)$

$tan^{(\theta_\lambda+\theta_\rho)} = -\left(\frac{X}{Z+W}\right)$ no matter which of(9) or(10) is chosen for the unitary angles.

V.SYSTOLIC ARRAY

One linear systolic array, the most efficient SVD algorithm is the Jacobi-like algorithm given by Brent and Luk [22]. The array implements a one-sided orthogonalization method due to Hestenes [23] and requires O (mnlog n) time and O (n) processors to compute the SVD of a real m x n matrix. It is capable of executing a sweep of the SVD-Jacobi method in O (n) time and is conjectured to require O (log n) sweeps for convergence. The proof of convergence for the Jacobi-SVD procedure with “parallel ordering"
is due to Park and Luk [24]. The Brent-Luk-Van Loan systolic array is primarily intended to compute the SVD of a real n x n matrix, although the SVD of an m x n matrix can be computed in m + O(n log n) time.
The SVD systolic array is an expandable, mesh connected array of processors, where each processor contains a 2x2 sub matrix of the input matrix M ε R^{nxn} . Assumed n as even, this systolic array is a square array of n/2 x n/2 processors. Before the computation processor P_{ij} contains $\begin{bmatrix} m_{2i-1,2j-1} & m_{2i-1,2j} \\ m_{2i,2j-1} & m_{2i,2j} \end{bmatrix}$ where (i , j= 1,…….n/2). Each processor P_{ij} is connected to its “diagonally” nearest neighbors $P_{i\pm1,j\pm1}$ (1 < i , j < n/2). The SVD systolic array with 16 processors for n=8 is shown below.

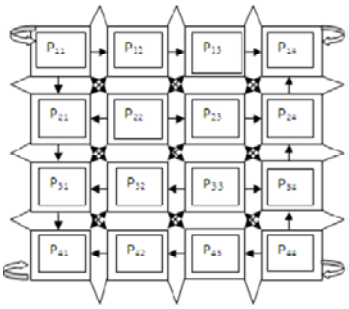

Fig.4.Matrix data elements and Rotation angles shown by SVD Systolic Array

The interconnections between the processors facilitate data exchange to implement the\ parallel ordering" of Brent Luk. The "parallel ordering" permits Jacobi rotations to be applied, in parallel, in groups of n=2. The angles for the n=2 Jacobi rotations are generated by the n=2 processors on the main diagonal of the array. The diagonal processors P_{ii} (i = 1,.,.,., 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} ($i \neq j$; $1 \leq i$; $j \leq n/2$). The application of a two-sided Jacobi rotation affects only the row and column of the diagonal processor generating the angles. In an idealized situation, the diagonal processors may broadcast the rotation angles, along the row and the column corresponding to their position in the array, in constant time. Each off-diagonal processor applies a two-sided rotation using the angles generated by the diagonal processors, in the same row and column with respect to its location in the array.
A proposed architecture for SVD-CORDIC matrix decomposition which operates on 4x4 complex valued dimensional matrices is shown below.

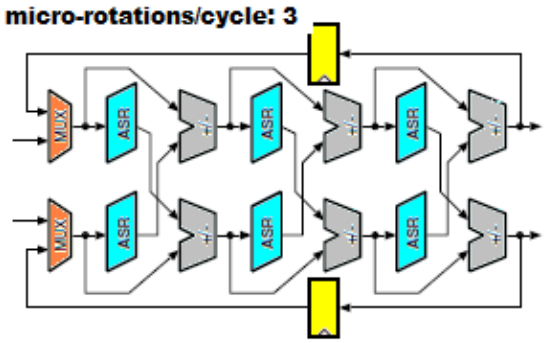

Fig.5. Architecture of SVD CORDIC

The SVD-CORDIC Architecture consists of three main units i.e. MEMORY (REGISTERS), ARITHMETIC UNIT and INSTRUCTION BASED SEQUENCER CONTROL UNIT. The matrix memory (registers) provides storage for three complex valued 4 × 4 matrices M= {M1, M2, M3,} which is sufficient to store the result of an SVD. A complex value in M is stored at a single memory address, is 32 bits wide, and each real and imaginary part requires 16 bits. The matrix memory consists of a two port 48×32 bit SRAM and requires 0.06mm2 in 0.18 µm CMOS technology. The matrix memory interface allows to read or write two different real or imaginary parts in at most two clock cycles.
Givens rotations, square-roots, multiplications, and additions/subtractions are required to compute the SVD. Givens rotations and the square root can efficiently be computed by CORDIC, whereas multiplications and additions/subtractions are computed in a multiply-accumulate (MAC) unit. CORDICs can efficiently compute two dimensional rotations [9] by performing a series of micro rotations with the aid of shifts and additions/subtractions. To keep the circuit area low, a single CORDIC is used by the means of time sharing and has been designed to support vectoring and rotation. A complex-valued Givens rotation is performed by three real valued vectoring CORDICs. To compute the trailing sub matrix of T = Bk a real-valued multiply-accumulate (MAC) unit has been instantiated. The multiplier can be switched off in order to perform additions or subtractions if required so in the operation. This Instruction based sequencer consists of a 64×20 bit instruction RAM (of size 0.04mm2 in 0.18 µm CMOS technology) that provides storage for 64 instructions. The finite state machine (FSM) decodes instructions, generatates memory addresses, and provides control signals for the AU.

VI.FPGA IMPLEMENTATION

The Implementation was done on a complex –valued 4x4 SVD for 0.18 µm CMOS Technology. Table below shows the results.

TABLE: Experimental results

Parameters	Total units	Throughput/Area(mm^2)	Clock frequency	SVD Execution time(µs)	Power Consumption
Cordic microrotations	**12**	**High/(0.43)**	**133MHz**	**12.50**	**170Mw**

VII.CONCLUSIONS

In this paper we described Design and implemented SVD matrix decomposition technique using Givens rotations. Low area was achieved using single CORDIC unit. The low-area MDUs have been shown to be suitable for MIMO OFDM systems, since they can be easily adapted to individual throughput requirements by the use of replication. A hardware oriented two-step diagonalization SVD scheme is derived from simple two-sided unitary transformations developed to ensure hardware and performance efficiency using CORDIC. In this paper we aimed for development of a systolic algorithm and, a hardware and performance efficient architecture implementable in VLSI, for computing the Singular Value Decomposition of an arbitrary complex matrix.

VIII.REFERENCES

[1] G. H. Golub and C. F. van Loan, Matrix Computations, 3rd ed. The Johns Hopkins University Press, Baltimore and London, 1996.
[2] Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoder algorithm," IEEE Journal of Solid-State Circuits, vol. 40, no. 7, pp. 1566–1577, July 2005.
[3] 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.
[4] N. D. Hemkumar and J. R. Cavalaro, "A systolic VLSI architecture for complex SVD," in Proceedings of the 1992 IEEE Intl. Symp. On Circuits and Systems, May 1993.
[5] S.-F. Hsiao and J.-M. Delosme, "Parallel singular value decomposition of complex matrices using multidimensional CORDIC algorithms," IEEE Trans. on Signal Processing, vol. 44, no. 3, pp. 685–697, Mar. 1996.
[6] H. Bölcskei, D. Gesbert, C. Papadias, and A. J. van der Veen, Eds., Space-Time Wireless Systems: From Array Processing to MIMO Communications. Cambridge Univ. Press, 2006.
[7] H. M. Ahmed, J. M. Delosme, and M. Morf. Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing. IEEE Computer, 15(1):65{82, January 1982.
[8] H. M. Ahmed. Signal Processing Algorithms and Architectures. PhD thesis, Dept. of Electrical Engineering, Stanford Univ., Stanford, CA, June 1982.
[9] L. H. Sibul and A. L. Fogelsanger. Application of Coordinate Rotation Algorithm to Singular Value Decomposition. IEEE Int. Symp. Circuits and Systems, pages 821,824, 1984.
[10] J. M. Speiser and H. J. Whitehouse. A Review of Signal Processing with Systolic Arrays. Proc. SPIE Real-Time Signal Processing, 431(VI):2 6, August 1983.
[11] 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.
[12] A. M. Finn, F. T. Luk, and C. Pottle. Systolic Array Computation of the Singular Value Decomposition. Proc. SPIE. Vol. 341. Real-Time Signal Processing V, pages 34, 43, 1982.
[13] K. Kota. Architectural, Numerical and Implementation Issues in the VLSI Design of an Integrated CORDIC SVD Processor. Master's thesis, Rice University, Department of Electrical and Computer Engineering, May 1991.
[14] C. Bridge, P. Fisher, and R. Reynolds. Asynchronous Arithmetic Algorithms for Data- Driven Machines. IEEE 5th Symposium on Computer Arithmetic, pages 56, 62, May 1981.
[15] F. Briggs and K. Hwang. Computer Architectures and Parallel Processing. Mc- Graw Hill, 1984.
[16] A. Bunse-Gerstner. Singular Value Decompositions of Complex Symmetric Matrices. J.Comp. Applic. Math., 21:41, 54, 1988.
[17] T. F. Coleman and C. F. Van Loan. Handbook for Matrix Computations. SIAM, Philadelphia, PA, 1988.
[18] 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 1990.
[19] E. G. Kogbetliantz. Solution of Linear Equations by Diagonalization of Coefficients Matrix. Quarterly of Applied Mathematics, 14(2):123{132, 2007.
[20] 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 2006.
[21] 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.
[22] R. P. Brent and F. T. Luk. The Solution of Singular-Value and Symmetric Eigen value Problems on Multiprocessor Arrays. SIAM Journal of Scientific and Statistical Computing, 6(1):69, 84, January 1985.
[23] M. R. Hestenes. Inversion of Matrices by Biorthogonalization and Related Results. J. Soc. Indust. Appl. Math, 6:51, 90, 2004.
[24] F. T. Luk and H. Park. A Proof of Convergence for Two Parallel Jacobi SVD Algorithms. IEEE Trans. on Computers, 38(6):806,811, June 2003.