

A Review of CORDIC Algorithms and Architectures with Applications for Efficient Designing

Rajkumar Tomar¹, Praveen Kumar Singh², Krishna Raj³

Abstract --- The coordinate rotation digital computer (CORDIC) algorithm is widely used in various technological fields such as digital signal processing (DSP), biomedical signal processing, robotics, communication systems, image processing etc. Due to ease of simple shift and add operations, the use of CORDIC based systems is increasing drastically. In this paper, some CORDIC based applications have been discussed along with its architectures and algorithms to reduce the number of iterations. To improve the system performance and to reduce the mean absolute percentage error (MAPE), some newly discovered algorithms such as double rotation CORDIC, triple rotation CORDIC, mixed scaling rotation (MSR), scaling free generalized micro-rotation, modified vector rotation (MVR) along with angle recoding (AR) and extended elementary angle set (EEAS) have also been discussed in detail. Use of CORDIC in applications such as in designing of digital filters, FFT computation, singular value decomposition (SVD) has also been reviewed here.

Index Terms-- Micro-rotation, CORDIC, MSR, elementary angle set, angle recoding, double rotation, triple rotation.

1. INTRODUCTION

The coordinate rotation digital computer (CORDIC) algorithm is a well-known iterative technique to perform various basic arithmetic operations including the computation of trigonometric functions, vector magnitude estimation, polar to rectangular transformation etc. It is preferred due to its simple shift-add operations, low cost and less complexity. CORDIC was invented in 1959 by Jack E. Volder [1], [2] for computation of trigonometric functions, multiplication and division. In 1971, John Walther [3], [4] showed that by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials and square roots. In [5], the author has proposed a Fourier transform computer with the help of CORDIC iterations. With the advancement in CORDIC algorithm, a pipelined CORDIC based IIR orthogonal digital filter [6] has been developed which improves the performance of the digital filter by reducing the iteration bound. Coarse grain pipelining and fine grain pipelining have been used in [6] to reduce the iteration bound of the digital IIR filter. Large number of iterations is the main drawback which affects the system speed adversely. Many algorithms such as angle recoding (AR) [7], EEAS [8], modified vector rotation (MVR) [9], mixed scaling rotation (MSR) [10], scaling free CORDIC algorithms [11], [12] etc. have been proposed to overcome this drawback and to improve the speed performance of the system. To speed up the system,

parallel and pipelined CORDIC are used.

In this paper, the concept of rotation and pseudo-rotation [13] has been discussed with the derivation of basic CORDIC equations. Normally CORDIC is used in circular rotation mode due to simplicity. The mode of CORDIC, either vectoring mode or rotational mode, is selected according to the requirement. Till now, some CORDIC based digital circuits as well as processors have been designed. CORDIC based complex multiplier has been discussed in [14] and such multiplier can be used in designing of digital filters with reduced power consumption as shown in [15]. With the advancement in CORDIC algorithm, double and triple rotation CORDIC have been proposed in [16] to minimize mean absolute percentage error (MAPE). This is not always true to make CORDIC a fastest technique but it is recommended due to ease of its hardware implementation. According to the requirement, the development of CORDIC algorithm and architecture has taken place for achieving high throughput, high signal to quantization ratio (SQNR) and reduction of hardware complexity as well as latency of implementation. To achieve high throughput, pipelined and parallel CORDIC [17] can be used.

The rest of the paper is organized as follows. Section 2 comprises the basic arithmetic of conventional and advanced CORDIC algorithms with rotation and pseudo-rotation concept. Architectures of CORDIC have been reviewed in section 3. In section 4, applications of CORDIC have been described. The conclusion with future aspects has been discussed in section 5.

2. ARITHMETIC OF CORDIC ALGORITHM

In this section, the basic arithmetic of CORDIC algorithm has been described. Double and triple rotation algorithms have also been discussed here.

Conventional CORDIC Algorithm

Coordinate Rotation Digital Computer is abbreviated as CORDIC. The concept of CORDIC based on principle of 2-D

1. Rajkumar Tomar is currently pursuing masters degree program in electronics & communication engineering in H.B.T.I kanpur, India, PH-09027909543. E-mail: tomar.raj005@gmail.com

2. Praveen Kumar Singh is currently pursuing masters degree program in electronics & communication engineering in H.B.T.I kanpur, India, PH-07784031680. E-mail: praveenhbt11@gmail.com

3. Dr. Krishna Raj is working as Associate Professor in electronics & communication engineering department in H.B.T.I kanpur, India, E-mail: kraj_biet@yahoo.com

geometry. The CORDIC algorithm involves rotation of a vector 'v' on the XY-plane in circular, linear and hyperbolic coordinate systems depending on the function to be evaluated. This is an iterative convergence algorithm that performs a rotation iteratively using a series of specific incremental rotation angles selected so that each iteration is performed by shift and add operations. The norm of a vector in these coordinate systems is defined as $\sqrt{x^2 + \mu y^2}$, where $\mu \in \{1, 0, -1\}$ represents a circular, linear or hyperbolic coordinate system respectively as shown in figure (1).

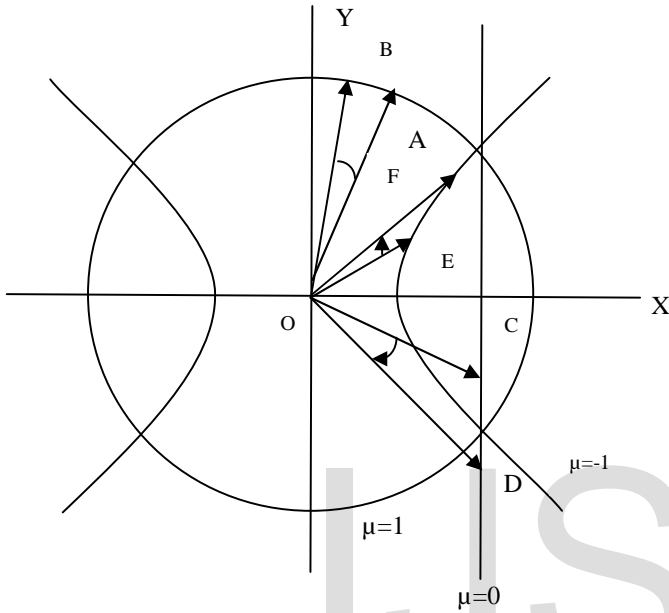


Figure 1: Circular, Linear, Hyperbolic CORDIC
The norm preserving rotation trajectory is a circle defined by $x^2 + y^2 = 1$ in the circular coordinate system. Similarly, the norm preserving rotation trajectory in the hyperbolic and linear coordinate systems is defined by the function $x^2 - y^2 = 1$ and $x=1$, respectively.

$$X_{i+1} = K_i[X_i - \mu Y_i d_i 2^{-i}] \dots\dots (1)$$

$$Y_{i+1} = K_i[Y_i + X_i d_i 2^{-i}] \dots\dots(2)$$

$$Z_{i+1} = Z_i - d_i \alpha_i \dots\dots(3)$$

$\mu = 1$, Circular rotations (basic CORDIC),

$$\alpha_i = \tan^{-1}(2^{-i})$$

$\mu = 0$, Linear rotations, $\alpha_i = 2^{-i}$

$\mu = -1$, Hyperbolic rotations, $\alpha_i = \tanh^{-1}(2^{-i})$

The CORDIC method can be employed in two different modes, namely, the rotation mode and the vectoring mode. The rotation mode is used to perform the general rotation by a given angle θ . The vectoring mode computes unknown angle θ of a vector by performing a finite number of micro-rotations. The iterative formulation of a computational algorithm for its implementation was first described by Jack E. Volder for the computation of trigonometric functions, multiplication & division. CORDIC based computing received increased attention in 1971, When John Walther [3]-[4] showed that, by varying few simple parameters, it could be used as a single algorithm for computing logarithms, exponentials [18] & square roots along with those suggested by Volder [1]. The popularity of CORDIC was much enhanced due to its

potential for efficient & low cost method, but not highly accurate. Although CORDIC may not be the fastest technique to perform these operations, but it is attractive due to the simplicity of its hardware implementation.

According to Volder the general rotation transform is

$$X' = X \cos\theta - Y \sin\theta \dots (4)$$

$$Y' = X \sin\theta + Y \cos\theta \dots (5)$$

In matrix representation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \dots (6)$$

Thus the rotational matrix is

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \dots\dots\dots (7)$$

The product of R_θ and R_ϕ equals to $R_{\theta+\phi}$ (rotation through θ then ϕ)

$$R_\theta R_\phi = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$$R_\theta R_\phi = \begin{bmatrix} \cos(\theta + \phi) & -\sin(\theta + \phi) \\ \sin(\theta + \phi) & \cos(\theta + \phi) \end{bmatrix} \dots\dots (8)$$

A rotation matrix is a matrix that is used to perform a rotation in Euclidean space. It turns the whole space around the origin as shown in figure (2).

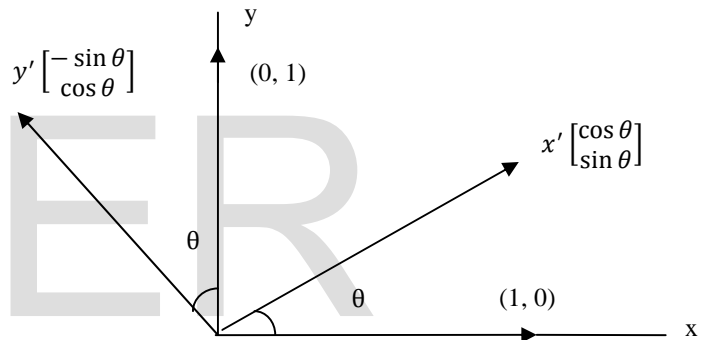


Figure 2: Representation of rotation matrix
Rotating a Vector (X_i, Y_i) By the Angle A_i (Rotations and Pseudo-rotations)

A pseudo-rotation step as shown in figure (3) increases the length of vector to: $R_{i+1} = R_i / \cos\alpha_i = R_i / (1 + \tan^2\alpha_i)^{1/2}$ [13], Whereas a real rotation does not change the length R_i of the vector R_i . Here our strategy is to eliminate the terms $(1 + \tan^2\alpha_i)^{1/2}$ and choose the angles α_i , so that $\tan\alpha_i$ is a power of 2. In real rotation the equations are:

$$X_{i+1} = X_i \cos\alpha_i - Y_i \sin\alpha_i = (X_i - Y_i \tan\alpha_i) / (1 + \tan^2\alpha_i)^{1/2}$$

$$Y_{i+1} = X_i \sin\alpha_i + Y_i \cos\alpha_i = (X_i \tan\alpha_i + Y_i) / (1 + \tan^2\alpha_i)^{1/2}$$

$$Z_{i+1} = Z_i - \alpha_i$$

In real rotation, we are using circular rotation so that $R_i = R_{i+1}$, because both are the radius of circle.

$$\text{Thus } R_i^2 = R_{i+1}^2 \dots\dots (9)$$

$$\text{While in pseudo-rotation, we have } R_{i+1} = R_i / \cos\alpha_i = R_i / (1 + \tan^2\alpha_i)^{1/2} \dots\dots (10)$$

We can write equation (9) as

$$X_i^2 + Y_i^2 = X_{i+1}^2 + Y_{i+1}^2$$

$$X_i^2 + Y_i^2 = X_{i+1}^2 + X_{i+1}^2 \tan^2(\alpha_i + \beta),$$

here angle $\beta = \text{angle } E_i O X$

$$R_i^2 = X_{i+1}^2 \sec^2(\alpha_i + \beta)$$

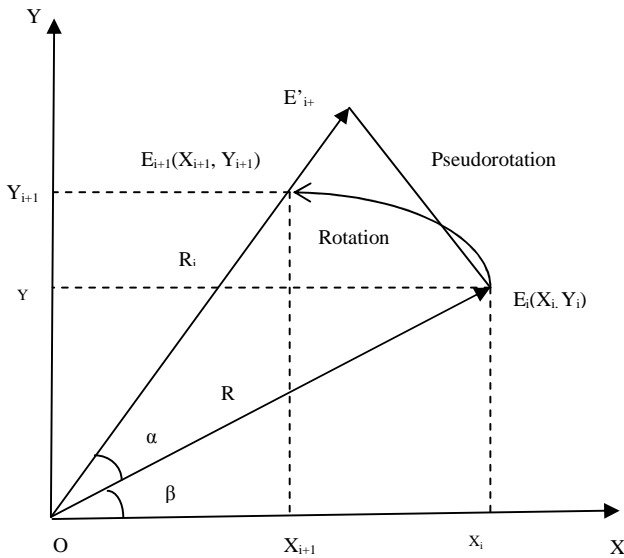


Figure 3: A pseudo-rotation step in CORDIC

$$X_{i+1} = R_i \cos(\alpha_i + \beta)$$

$$X_{i+1} = R_i (\cos\alpha_i \cos\beta - \sin\alpha_i \sin\beta)$$

$$X_{i+1} = R_i \left(\cos\alpha_i \frac{X_i}{R_i} - \sin\alpha_i \frac{Y_i}{R_i} \right)$$

Because

$$\cos\beta = \frac{X_i}{R_i}, \quad \sin\beta = \frac{Y_i}{R_i}$$

$$X_{i+1} = X_i \cos\alpha_i - Y_i \sin\alpha_i$$

$$= (X_i - Y_i \tan\alpha_i) / (1 + \tan^2\alpha_i)^{1/2}$$

Similarly, we can write

$$Y_{i+1} = X_i \sin\alpha_i + Y_i \cos\alpha_i$$

$$= (X_i \tan\alpha_i + Y_i) / (1 + \tan^2\alpha_i)^{1/2}$$

After pseudo-rotation, from equation (7) we get the equations for pseudo-rotation as:

$$X_{i+1} = (X_i - Y_i \tan\alpha_i) \dots (11)$$

$$Y_{i+1} = (X_i \tan\alpha_i + Y_i) \dots (12)$$

For m rotations the desired angle,

$$Z = \alpha_1 + \alpha_2 + \dots + \alpha_m \dots (13)$$

$$\text{After } m \text{ real rotations, } \begin{cases} X_m = X_0 \cos(\Sigma\alpha_i) - Y_0 \sin(\Sigma\alpha_i) \\ Y_m = X_0 \sin(\Sigma\alpha_i) + Y_0 \cos(\Sigma\alpha_i) \\ Z_m = Z - (\Sigma\alpha_i) \end{cases}$$

Where $K = \prod (1 + \tan^2\alpha_i)^{1/2} \dots (14)$

If the rotation angles are restricted such that $\tan(\alpha_i) = \pm 2^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations.

The iterative rotation can now be expressed as:

$$\begin{cases} X_m = K(X_0 \cos(\Sigma\alpha_i) - Y_0 \sin(\Sigma\alpha_i)) \\ Y_m = K(X_0 \sin(\Sigma\alpha_i) + Y_0 \cos(\Sigma\alpha_i)) \\ Z_m = Z - (\Sigma\alpha_i) \end{cases}$$

Where $K = \prod (1 + \tan^2\alpha_i)^{1/2} \dots (14)$

The product of the K_i 's can be applied elsewhere in the system treated as a part of a system processing gain. That product approaches 0.6073 as the number of iterations goes to infinity. Therefore, the rotation algorithm as a gain A_n , of approximately 1.647. The exact gain depends on the number

of iterations, and obeys the relation

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

The angle accumulator adds a third difference equation to the CORDIC algorithm:

$$Z_{i+1} = Z_i - d_i \tan^{-1}(2^{-i})$$

The CORDIC rotator is normally operated in one of the two modes. In the first mode (rotation mode), we rotates the input vector by a specified angle (given as an argument). In the second mode (vectoring mode), we rotates the input vector to the x-axis while recording the angle required to make that rotation.

In rotation mode, the angle accumulator is initialized with the desired rotation angle. The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision at each iteration is therefore based on the sign of the residual angle after each step. Naturally, if the input angle is already expressed in the binary arctangent base, the angle accumulator may be eliminated. For rotation mode, the cordic equations are:

$$X_{i+1} = K_i [X_i - Y_i d_i 2^{-i}]$$

$$Y_{i+1} = K_i [Y_i + X_i d_i 2^{-i}]$$

$$Z_{i+1} = Z_i - d_i \tan^{-1}(2^{-i})$$

Where

$$d_i = -1 \text{ if } Z_i < 0 \text{ otherwise } +1$$

In rotation mode these equations can be written as

$$X_n = A_n [X_0 \cos Z_0 - Y_0 \sin Z_0] \dots (15)$$

$$Y_n = A_n [Y_0 \cos Z_0 + X_0 \sin Z_0] \dots (16)$$

$$Z_n = 0, \quad A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

In the vectoring mode, the CORDIC rotator rotates the input vector through whatever angle is necessary to align the result vector with the x axis. The result of the vectoring operation is a rotation angle and the scaled magnitude of the original vector (the x component of the result). The vectoring function works by seeking to minimize the y component of the residual vector at each rotation. The sign of the residual y component is used to determine which direction to rotate next. If the angle accumulator is initialized with zero, it will contain the traversed angle at the end of the iterations. In vectoring mode, the CORDIC equations are:

$$X_{i+1} = K_i [X_i - Y_i d_i 2^{-i}]$$

$$Y_{i+1} = K_i [Y_i + X_i d_i 2^{-i}]$$

$$Z_{i+1} = Z_i - d_i \tan^{-1}(2^{-i})$$

Where

$$d_i = +1 \text{ if } Y_i < 0 \text{ otherwise } -1$$

In vectoring mode these equations can be written as

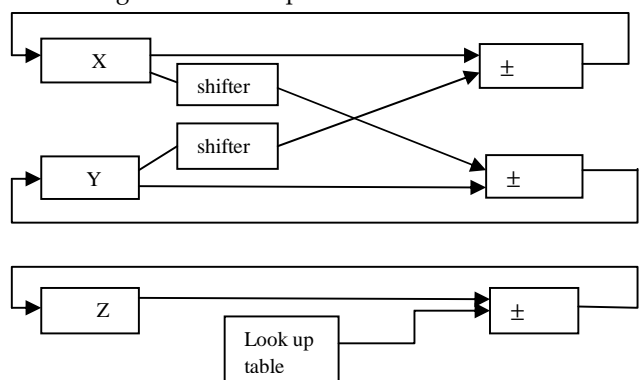


Figure 4: Hardware elements needed for the CORDIC method

$$X_n = A_n \sqrt{X_0^2 + Y_0^2} \dots\dots\dots (17)$$

$$Y_n = 0,$$

$$Z_n = Z_0 + \tan^{-1}(Y_0/X_0) \dots\dots\dots (18)$$

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

Hardware elements of CORDIC method are shown in figure (4). If very high speed is not needed, a single adder and single shifter is sufficient (as in calculator).

Basic CORDIC Iterations:

In CORDIC iteration, an angle whose tangent is $d_i 2^{-i}$ is pseudo-rotated in step i . Under this procedure, the angle is kept fixed, only direction d_i is to be changed. Table I describes the angle computation.

$$\begin{aligned} X_{i+1} &= X_i - Y_i d_i 2^{-i}, \\ Y_{i+1} &= Y_i + X_i d_i 2^{-i}, \\ Z_{i+1} &= Z_i - d_i \tan^{-1}(2^{-i}) = Z_i - d_i \alpha_i \end{aligned}$$

For example, we want to compute angle 40° , then the angle will be calculated in the following manner.

$$40^\circ \square 45.0^\circ - 26.6^\circ + 14.0^\circ + 7.1^\circ + 3.6^\circ - 1.8^\circ - 0.9^\circ - 0.4^\circ + 0.2^\circ - 0.1^\circ = 40.1^\circ$$

TABLE I: CORDIC Rotation through an angle α

Iteration, i	Zi	-di.alpha_i = -di.tan-1(2 ⁻ⁱ)	Zi+1
0	+40	-45.0	-5.0
1	-5.0	+26.6	+21.6
2	+21.6	-14.0	+7.6
3	+7.6	-7.1	+0.5
4	+0.5	-3.6	-3.1
5	-3.1	+1.8	-1.3
6	-1.3	+0.9	-0.4
7	-0.4	+0.4	+0.0
8	+0.0	-0.2	-0.2
9	-0.2	+0.1	-0.1

Vector Rotation Methods:

Three methods are discussed here to represent the CORDIC vector rotation.

(1) Standard Method:

$$\begin{aligned} x' &= x \cos\theta - y \sin\theta \\ y' &= y \cos\theta + x \sin\theta \end{aligned}$$

Four multiplications and two additions are required.

(2) Golub's Method [19], [25]:

$$\begin{aligned} x' &= [(x + y) (\cos\theta - \sin\theta) + x \sin\theta - y \cos\theta] \\ y' &= [y \cos\theta + x \sin\theta] \dots\dots\dots (19) \end{aligned}$$

Three multiplications and five additions are required.

(3) Buneman's Method [20]:

$$\begin{aligned} x' &= (1 + \cos\theta)(x - y \tan\frac{\theta}{2}) - x \\ y' &= (1 + \cos\theta) (x - y \tan\frac{\theta}{2}) \tan\frac{\theta}{2} + y \dots\dots\dots (20) \end{aligned}$$

Three multiplications and three additions are required because $(1 + \cos\theta)$ and $\tan\frac{\theta}{2}$ are stored constants.

Angle Recoding Scheme

CORDIC equations (4), (5) can be written in matrix form as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (21)$$

This technique [7] is used to reduce the number of iterations by encoding the angle of rotation as a linear combination of a set of selected elementary angles of micro-rotation. Matrix

related computations, special purpose algorithms (i.e. to reduce the complexity, to speed up the system etc.) and hardware are essential for faster computation of millions of rotation operation in DSP. Earlier the sign, $u(i) = \pm 1$, so n CORDIC iterations will always be required even if $\theta = 0$, because each time $u(i) = +1$ or -1 . In this algorithm $u(i) = 0$ is also permissible. This is useful if angle is known in advance. In applications such as FFT and CZT, the angle θ is known prior to computation. Hence, the trigonometric functions $\cos\theta$ and $\sin\theta$ can be evaluated and stored in advance. During computation, these prestored values will be retrieved and multiplied by x and y . This requires four real multiplications and two additions. This technique reduces the complexity upto 50%. The given angle θ is decomposed as

$$\theta = \sum_{i=0}^{n-1} u(i) \alpha_i + \zeta \dots\dots\dots (22)$$

Where angle approximation error

$$|\zeta| < \alpha_{(n-1)}$$

$$\text{For } z(0) = \theta, z(i+1) = z(i) - u(i) \alpha_i,$$

Where $i = 0, 1, \dots, (n-1)$

$$u(i) = \text{sign of } z(i)$$

In angle recoding we follow the following two steps:

$$\theta = \sum_{i=0}^{n-1} u(i) \alpha_i + \zeta, \text{ for } \zeta < \alpha_{(n-1)}$$

$\sum_{i=0}^{n-1} |u(i)|$ is minimized.

Extended Elementary Angle Set Recoding (EEAS) Scheme

In the conventional CORDIC the elementary angle set (EAS) is defined as $S = \{\sigma \tan^{-1}(2^{-l})\}$, where $\sigma \in \{-1, 1\}$ and $l \in \{1, 2, \dots, n-1\}$. In the angle recoding, the elementary angle set is defined as $S = \{\sigma \tan^{-1}(2^{-l})\}$, where $\sigma \in \{-1, 0, 1\}$ and $l \in \{1, 2, \dots, n-1\}$. AR scheme reduces number of iterations upto half that of previous one for same n bit accuracy. In vectoring mode, this method is termed as the backward angle recoding. Under EEAS [8] scheme the elementary angle set is defined as $S = \{\tan^{-1}(\sigma_1 2^{-l_1} + \sigma_2 2^{-l_2})\}$, where $\sigma_1, \sigma_2 \in \{-1, 0, 1\}$ and $l_1, l_2 \in \{1, 2, \dots, n-1\}$. Number of iterations reduces in EEAS in comparison to EAS. EEAS shows also the better error performance than that of previous one. The matrix representation of EEAS is as

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -\sigma_1(i)2^{-l_1^{(i)}} - \sigma_2(i)2^{-l_2^{(i)}} \\ \sigma_1(i)2^{-l_1^{(i)}} + \sigma_2(i)2^{-l_2^{(i)}} & 1 \end{bmatrix} \dots\dots\dots (23)$$

Scaling Factor $K = \prod K_i$

$$\text{Where } K_i = \frac{1}{\sqrt{[1 + (\sigma_1(i)2^{-l_1^{(i)}} + \sigma_2(i)2^{-l_2^{(i)}})^2]}} \dots\dots\dots (24)$$

Double and Triple Rotation CORDIC

This algorithm [16] accelerates the rotation computation of CORDIC algorithm by duplicating the elementary angle to be $2\theta_i$. By using this algorithm, the basic CORDIC equations can be written in matrix form as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^r \begin{bmatrix} x \\ y \end{bmatrix}$$

Where r is a rotation degree, $r = 1$ for conventional algorithm, $r = 2$ for double rotation and $r = 3$ for triple rotation CORDIC. For double and triple rotation the angle is $\tan^{-1}(2^{-i-1})$ and $\tan^{-1}(2^{-i-2})$ respectively. The scaling factor for these two algorithms is equal to 0.9219 and 0.9922 respectively while in case of conventional algorithm it is 0.60725. As described in [16], mean absolute percentage error (MAPE) for these two

algorithms is less than that of the conventional algorithm. MAPE is defined as

$$M = \frac{1}{n} \sum_{i=1}^n \left| \frac{A_i - F_i}{A_{avg}} \right| \dots \dots \dots (25)$$

Where A_i is the actual value, F_i is the forecast value and A_{avg} is the average actual value.

3. ARCHITECTURES AND ADVANCED ALGORITHMS BASED ON CORDIC

This section depicts the CORDIC architectures and algorithms along with their advantages and disadvantages.

1. Parallel Angle Recoding

The schemes as described in [21], [22] like EAS and EEAS reduces the number of iterations but with these schemes a necessary condition is that the angle of rotation should be known in advance. If angle is not known in advance, these algorithms take more cycle time than that of the conventional algorithm. In such case parallel angle recoding is used. It can be used in conjunction with angle recoding method to gain the advantage of reduction of iteration count, without further increase in cycle time.

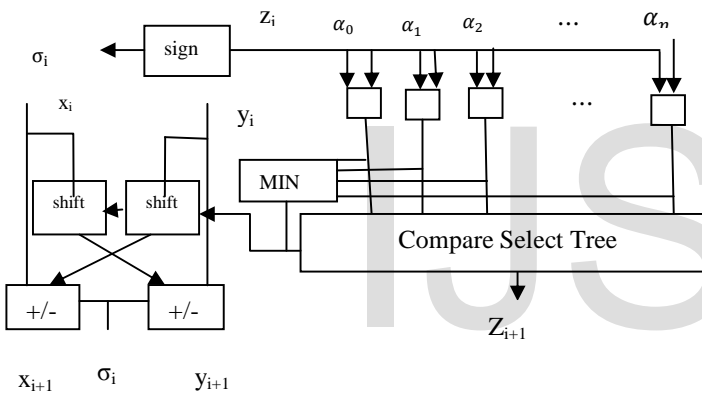


Figure 5: CORDIC Architecture for Parallel Angle Recoding
In parallel angle recoding, elementary angle is tested in parallel and the direction for micro-rotation can be determined quickly to minimize the iteration period. During each iteration, residual angle z is passed to a set of n adder - subtractor units for the computation of $\Delta_i = (z_i - \sigma_i \alpha_i)$. The $\sigma_i \alpha_i$ corresponding to the smallest difference $(\Delta_i)_{min}$ is selected as the angle of rotation. Figure (5) shows the implementation of this algorithm.

2. Hybrid CORDIC

Based on radix - 2 decomposition, any rotation angle θ with n - bit precision can be expressed as a linear combination of angles from the set $\{2^{-i}; i \in \{1, 2, \dots, n-1\}\}$, given by $\sum_{i=0}^{n-1} b_i 2^{-i}$ where $b_i \in \{0, 1\}$. Due to hardware complexity, radix - 2 decomposition is not used in the conventional CORDIC. This type of CORDIC algorithm [23], [24] is also known as coarse - fine rotation CORDIC because the rotation angle is decomposed into two set of angles, one is coarse sub-angles and the other is fine sub-angles.

The elementary angle set, S , is defined as the union of two sets, S_1, S_2 in the following ways: $S = S_1 \cup S_2$,

Where $S_1 = \tan^{-1}(2^{-i}), i \in \{1, 2, \dots, p-1\}$,
 $S_2 = 2^{-j}, j \in \{p, p+1, \dots, n-1\}$

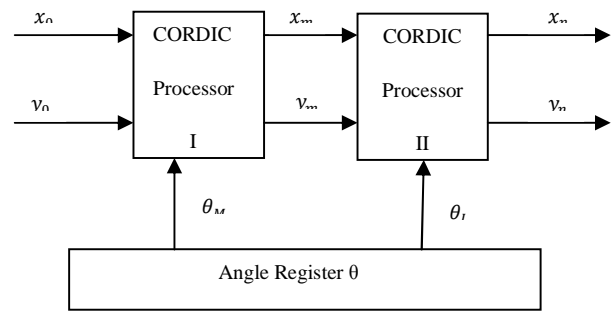


Figure 6: Hybrid CORDIC Architecture
In the second set j is sufficiently large (say $j > n/3 - 1$) such that $\tan \alpha_j = \alpha_j$; where $\alpha_j = \tan^{-1}(2^{-j})$. Let θ be the angle of rotation and it is decomposed into two set of sub-angles as $\theta = \theta_M + \theta_L$; where the coarse sub-angles $\theta_M = \sum_{i=1}^{p-1} \sigma_i \tan^{-1}(2^{-i})$, $\sigma_i \in \{1, -1\}$ and the fine sub-angles $\theta_L = \sum_{i=p}^{n-1} d_i (2^{-i})$, $d_i \in \{0, 1\}$. In hybrid CORDIC operations, a combination of coarse and fine micro-rotations are used in two stages joined in cascade manner as shown in figure (6). In conventional method, the CORDIC equations in matrix form can be written as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Here in hybrid CORDIC operations, the above matrix form can be represented in two stages. In first stage, the coarse sub-angle rotation equations are as

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & -\tan\theta_M \\ \tan\theta_M & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \dots \dots (26)$$

In second stage, the fine sub-angle rotation equations are as

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & -\tan\theta_L \\ \tan\theta_L & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix} \dots \dots (27)$$

CORDIC Architectures

Some useful basic architectures of CORDIC can be described as

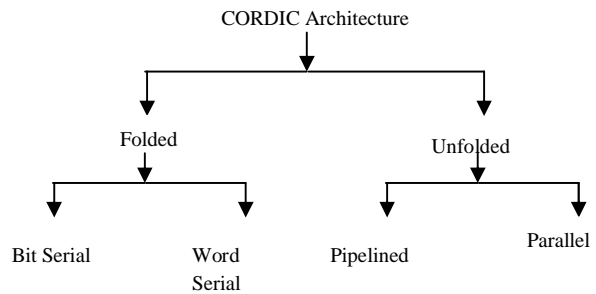


Figure 7: Taxonomy of CORDIC Architectures
Folded architectures are obtained by duplicating each of the difference equations of CORDIC algorithm into hardware and time multiplexing all the iterations into a single functional unit. In bit serial architecture, the functional unit implements the logic for one bit of each iteration while in word serial; the functional unit implements the logic for one word of each iteration. By using unfolded architectures, one can get high throughput due to elimination of read only memory. A simple iterative CORDIC architecture [18], [26] can be obtained by duplicating each of the three difference equations in hardware as shown in figure (8).

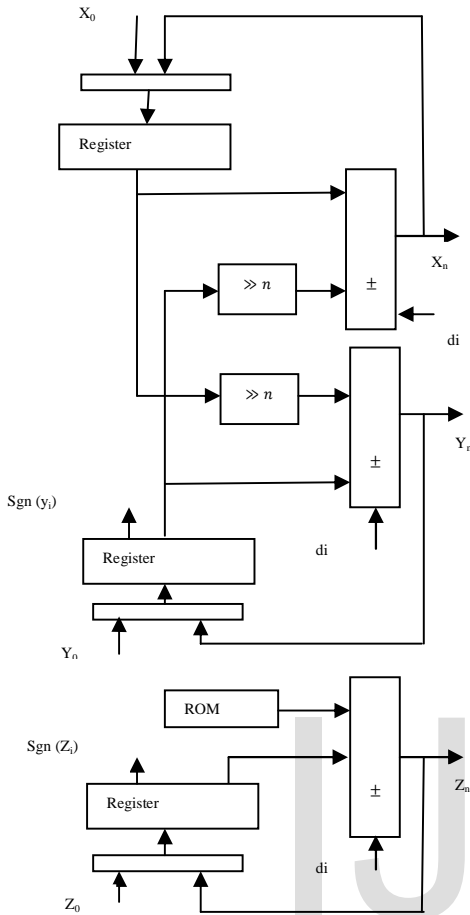


Figure 8: Iterative CORDIC Structure (Bit Parallel Design)
 The decision function d_i , is driven by the sign of the y or z register depending on whether it is operated in rotation mode or in vectoring mode. The initial values of x , y , z are loaded through multiplexers into the corresponding registers. The drawback of this iterative architecture or bit parallel design architecture is its slow design that requires a large number of logic cells. To overcome this drawback, bit serial arithmetic architecture is used. Bit serial arithmetic architecture work at much higher clock rate than the equivalent bit parallel design. In this architecture, three shift registers each having length equal to word width is used as shown in figure (9).

3. Pipelined CORDIC

Pipelining [27] is used to reduce the critical path so that the system may be speeded up. Pipelined CORDIC is used in fixed and adaptive filters, discrete orthogonal transforms, sinusoidal wave generation and other signal processing applications. In pipelined CORDIC shifters are eliminated because the shift operation can be hardwired with adders. For using three adders in each stage as shown in figure (10), the critical path is $TA + TM + T2C$, where TA is the computation time for an adder, TM is the computation time for a multiplexer, $T2C$ is the computation time for computing 2's complementer operation. It uses pipeline registers in between each iteration phase. For angle known in advance, the

sign of micro-rotation can be predetermined and the need of multiplexing can be avoided for reducing the critical path.

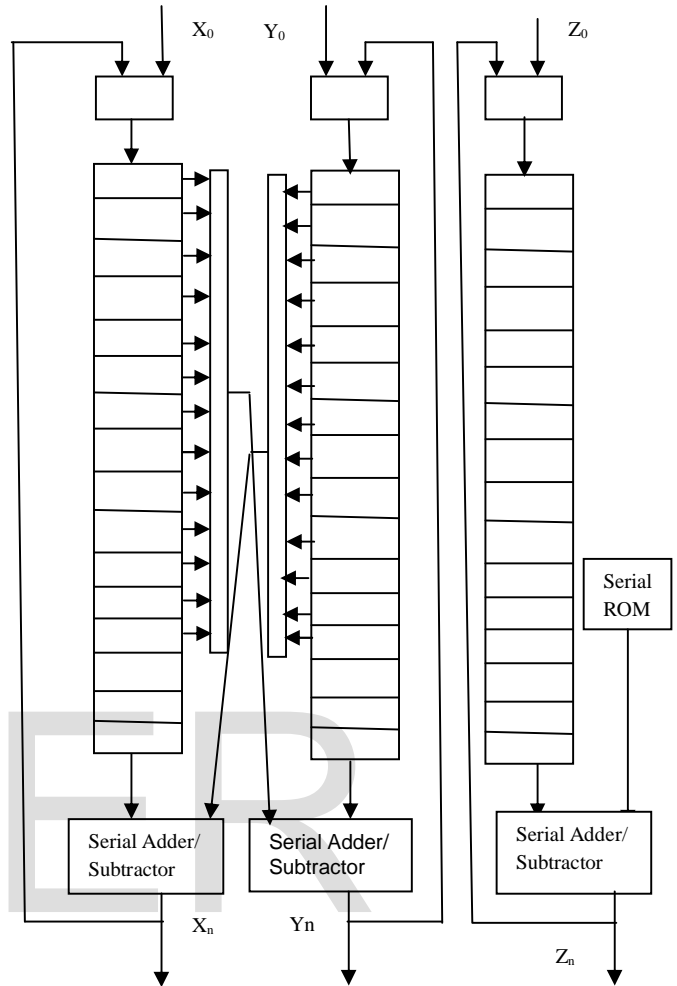


Figure 9: Bit Serial Architecture

4. Parallel CORDIC

As shown in figure (11), the architecture of parallel CORDIC [17] is quite similar to pipelined CORDIC. In this algorithm, the structure for iterative CORDIC is instantiated multiple times simultaneously. For any angle θ , we can decompose this angle into sub-angles and the computation of these angles can be processed in parallel. Angle θ can be written as $\theta = d_0\alpha_0 + d_1\alpha_1 + \dots + d_{n-1}\alpha_{n-1} \dots (28)$

Where $\alpha_i = \tan^{-1}(2^{-i})$.

Having latency equal to only one clock cycle, the parallel architectural configuration implements the shift-add/sub operations in parallel using an array of shift-add/sub stages. With multiple inputs and multiple outputs and high throughput, the parallel architecture is more efficient than the iterative architecture

5. Differential CORDIC

This type of CORDIC [28] provides faster and more efficient redundant number based implementation of both rotation mode and vectoring mode CORDIC. It uses some temporary variables. In Differential - CORDIC, the original CORDIC angle recursion is transformed in such a way that only absolute values of the angle are used. The angle equation $Z_{i+1} = Z_i - d_i \cdot \alpha_i$ can be written as $|Z_{i+1}| = ||Z_i| - \alpha_i| = ||Z_i| +$

α_i . Where $Z_{i+1} = d_i \cdot Z_i + 1$, $d_{i+1} = d_i \cdot d_i + 1$. Here Z_i is the residual angle. $D_i = \text{sign}(Z_i)$, $d_i = \text{sign}(Z_i)$, $\alpha_i = \text{sub-rotation angle}$ and $d_i = -\alpha_i$. This type of CORDIC uses redundant number system. It can serially process data and it provides data output in most significant digit first orderly. When the stages are connected in cascade manner, the processing time between two stages is over-lapped in a one-digit time-skewed manner, achieving high throughput.

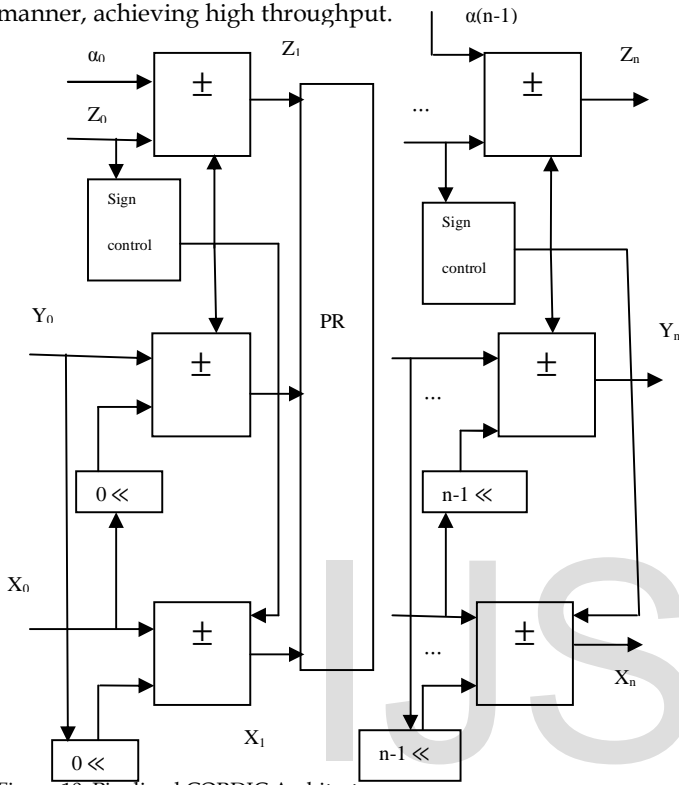


Figure 10: Pipelined CORDIC Architecture

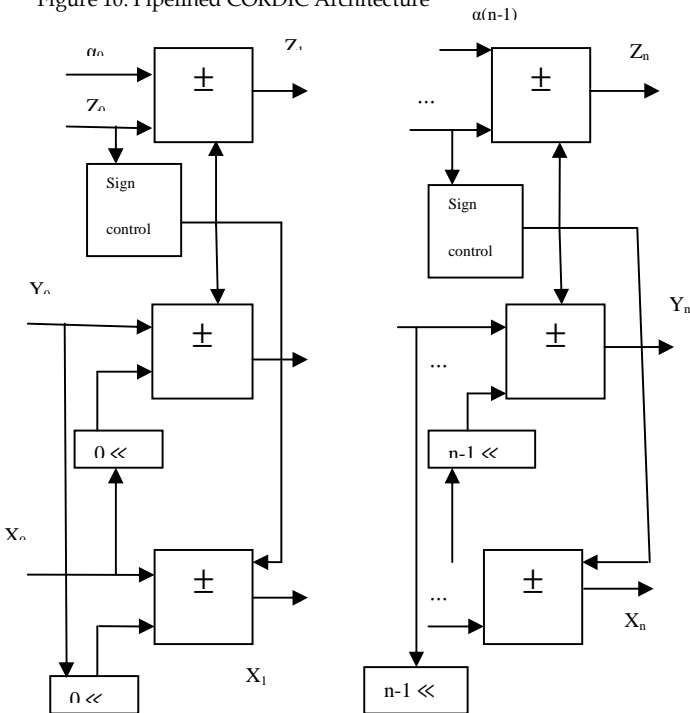


Figure 11: Architecture of Parallel CORDIC

6. MVR and MSR CORDIC

The basic CORDIC algorithm is carried out only by a sequence of shift-and-add operations. Despite its simplicity, it encounters the disadvantage of large number of iterations, which impedes the speed performance in practical implementations. Modified vector rotation (MVR) [9] and Mixed scaling rotation (MSR) [10] algorithms are advanced CORDIC algorithms after AR CORDIC, fast CORDIC, and EEAS-CORDIC. Both the algorithms reduce the number of iterations and provide better performance. In MSR CORDIC, both the phases, rotation phase and scaling phase, are mixed. This algorithm achieves better signal to quantization noise ratio (SQNR). The performance analysis of this algorithm is based on the signal to quantization ratio (SQNR) and complexity. To store the continuous values of voltages (between 0 and 1) in 8 bits, the continuous values are forced to have the approximate values among 256 different values. This introduces a round-off error. It is termed as quantization noise (or quantization error). For a quantization accuracy of N bits per sample, the SQNR can be described as

$$SQNR = 20 \log_{10} \frac{V_{\text{signal}}}{V_{\text{quantization noise}}} \dots (29)$$

In CORDIC, to calculate the SQNR, we use the following formula [9]

$$SQNR \text{ (dB)} = 10 \log_{10} \left(\frac{1}{\zeta_m^2 + \zeta_s^2} \right)$$

Where ζ_m is the residual angle error and ζ_s is the quantization error in scaling.

Scaling approximation error is defined as

$$\zeta_s \cong \left| 1 - \frac{P}{P} \right|, \text{ where } P \text{ is the scaling factor.}$$

In conventional CORDIC algorithm, using the concept of micro-rotation the target rotation angle can be decomposed into predefined elementary angles. Thus CORDIC equations can be written in matrix form as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \left(\prod_{n=0}^{N-1} \begin{bmatrix} \cos \theta_n & -\mu_n \sin \theta_n \\ \mu_n \sin \theta_n & \cos \theta_n \end{bmatrix} \right) \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \left(\prod_{n=0}^{N-1} \cos \theta_n \right) \left(\prod_{n=0}^{N-1} \begin{bmatrix} 1 & -\mu_n \tan \theta_n \\ \mu_n \tan \theta_n & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\Theta = \sum_{n=0}^{N-1} \mu_n \theta_n + \zeta \dots (30)$$

Where Θ is the expected rotation angle; N is the number of rotations. $\theta_n = \tan^{-1}(2^{-n})$, where n denotes the elementary angle (or micro-angle); and $n = 1, 2, 3, \dots, N-1$. $\mu_n \in \{1, -1\}$, indicates the direction of rotation for $n=1, 2, 3, \dots, N-1$; and ζ is the residual angle. In the modified vector rotation (MVR)-CORDIC algorithm rotation procedure is modified to reduce the iteration as well as to accelerate the computational speed. In extended MVR-CORDIC, the elementary angle set is $\theta \in \{\arctan(\sum_{i=1}^s \mu_i 2^{-s_i}) \mid \mu_i \in \{-1, 0, 1\}, s_i \in \{0, 1, 2, \dots, S\}\} \dots (31)$

1. ROTATIONAL PHASE: for $n = 0, 1, 2, \dots, N-1$.

$$\begin{bmatrix} x(n+1) \\ y(n+1) \end{bmatrix} = \begin{bmatrix} 1 & -\mu_n 2^{-n} \\ \mu_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix} \dots (32)$$

Elementary angle $\theta_n = \mu_n \tan^{-1}(2^{-n})$
Accumulation angle $z(n+1) = z(n) + \mu_n \theta_n$

2. SCALING PHASE: $\begin{bmatrix} x(N) \\ y(N) \end{bmatrix} = P \begin{bmatrix} x(N) \\ y(N) \end{bmatrix} \dots (33)$

$$P = \prod_{n=0}^{N-1} (\sqrt{1 + 2^{-2n}})^{-1}$$

Where I is defined as the Extending Factor and denotes the number of Signed Power-of-Two (SPT) terms; S denotes the

number of maximum shift.

Since the scaling factor is always greater than 1 in the existing CORDIC algorithms, it is necessary to scale down the norm of the input vector to its initial value in the scaling phase. Furthermore, the SQNR performance will be degraded due to the growth of the scaling factor. To alleviate the degradation of the SQNR performance, it is better to keep the norm of the input vector as close as to unity, during each iteration. Additionally, if the norm of the rotated vector is moved to the same as the original vector in the final micro-rotation operation, we can reduce the overhead of the scaling operation. Based on the idea, we reformulate the iterative arithmetic as follows: Mixed Scaling Rotation for $n = 1, 2, \dots, N$.

$$\begin{bmatrix} x(n) \\ y(n) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^I \eta_j(n) 2^{-s_j(n)} & -\sum_{i=1}^I \mu_i(n) 2^{-t_i(n)} \\ \sum_{i=1}^I \mu_i(n) 2^{-t_i(n)} & \sum_{j=1}^J \eta_j(n) 2^{-s_j(n)} \end{bmatrix} \begin{bmatrix} x(n-1) \\ y(n-1) \end{bmatrix} \dots (34)$$

Elementary angle $\theta_n = \tan^{-1} \left(\frac{\sum_{i=1}^I \mu_i(n) 2^{-t_i(n)}}{\sum_{j=1}^J \eta_j(n) 2^{-s_j(n)}} \right) \dots (35)$

Updated accumulation angle $z(n) = z(n-1) + \theta_n$

Amplifying factor in the nth rotation

$$P_n = \left(\sum_{j=1}^J 2^{-s_j(n)} \right)^2 + \left(\sum_{i=1}^I 2^{-t_i(n)} \right)^2$$

Scaling factor: $P = \prod_{n=1}^N P_n$

Where n denotes the n th iteration and N denotes the total number of iterations.

$\eta_j(n), \mu_i(n) \in \{-1, 0, 1\}$. In MSR CORDIC, $\eta_j(n)$ or $\mu_i(n)$ may be equal to zero but it is invalid in conventional CORDIC. $\{t_i(n), s_j(n)\} \in \{0, 1, \dots, S\}$, where S denotes the number of maximum shift, I and J denote the number of SPT terms of $x(n)$ and $y(n)$ respectively. I and J are selected as

(1) Both I and J are equal to $N_{spt}/2$, when N_{spt} (no. of sum of power of two) is even.

(2) $I = (N_{spt} + 1)/2$, and $J = I - 1$, when N_{spt} is odd.

Equation (34) can be written as

$$\begin{aligned} x(n) &= \sum_{j=1}^J \eta_j(n) 2^{-s_j(n)} \cdot x(n-1) - \sum_{i=1}^I \mu_i(n) 2^{-t_i(n)} \cdot y(n-1) \\ y(n) &= \sum_{i=1}^I \mu_i(n) 2^{-t_i(n)} \cdot x(n-1) + \sum_{j=1}^J \eta_j(n) 2^{-s_j(n)} \cdot y(n-1) \end{aligned}$$

For $N_{spt} = I + J = 3$, there are four types of MSR-CORDIC have been shown in table II. It is impossible to use only the Scaling Type or Exchange-Scaling Type to implement the rotation circuits. The reason is that they perform only the scaling operation at fixed rotation angles $0, \pm\pi/2, \pi$. Hence only Types II and III can be used to construct the rotation circuits.

Scaling Free CORDIC:

Under this algorithm [11], [29], Taylor series expansions of sine and cosine functions are used to avoid the scaling operation. Here a generalized sequence of micro-rotation to have adequate range of convergence (ROC) based on the selected order of approximation of the Taylor series has been suggested. The Taylor expansions of sine and cosine of an angle " α " are given by

$$\sin \alpha = \alpha - (3!)^{-1} \cdot \alpha^3 + (5!)^{-1} \cdot \alpha^5 \dots (36a)$$

$$\cos \alpha = 1 - (2!)^{-1} \cdot \alpha^2 + (4!)^{-1} \cdot \alpha^4 \dots (36b)$$

After simulation it is noticed that the maximum percentage of error in sine and cosine functions for third order approximation is 0.0033% and 0.0168%, respectively. For angle α , CORDIC equations described by equations (1), (2) can be written as

$$X_{i+1} = (1-(2!)^{-1} \cdot \alpha^2 + (4!)^{-1} \cdot \alpha^4 \dots) X_i - (\alpha - (3!)^{-1} \cdot \alpha^3 + (5!)^{-1} \cdot \alpha^5 \dots) \cdot Y_i \dots (37a)$$

$$Y_{i+1} = (\alpha - (3!)^{-1} \cdot \alpha^3 + (5!)^{-1} \cdot \alpha^5 \dots) X_i + (1 - (2!)^{-1} \cdot \alpha^2 + (4!)^{-1} \cdot \alpha^4 \dots) \cdot Y_i \dots (37b)$$

To implement (36) by shift-add operations, it is needed to approximate the factorial terms by the power of 2 values, replacing $3 \cdot$ by 2^3 . For simplicity, Taylor series expansions are used upto third order. Thus equation (24) can be written in matrix form as

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} (1 - (2!)^{-1} \cdot \alpha^2) & -(\alpha - (2!)^{-1} \cdot \alpha^3) \\ (\alpha - (2!)^{-1} \cdot \alpha^3) & (1 - (2!)^{-1} \cdot \alpha^2) \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \dots (38)$$

The expressions for the basic-shifts, the first elementary angle of rotation α_1 and ROC for different orders of approximations for different word-length of implementations are as follows:

Basic shift $s = \left\lfloor \frac{b - \log_2(n+1)!}{n+1} \right\rfloor \dots (39)$

ROC = $n1 \times \alpha_1 \dots (40)$

Where b is the word-length, $n1$ is the number of micro-rotations and $\alpha_1 = 2^{-s}$. If the order of approximation increases, the basic-shift decreases, the first elementary angle of rotation increases and ROC is expanded.

TABLE II: For $N_{spt} = 3$, four types of MSR-CORDIC

Type	I	J	Equation
I (Scaling Type)	0	3	$x(n) = \eta_1(n) 2^{-s_1(n)} \cdot x(n-1) + \eta_2(n) 2^{-s_2(n)} \cdot x(n-1) + \eta_3(n) 2^{-s_3(n)} \cdot x(n-1)$ $y(n) = \mu_1(n) 2^{-t_1(n)} \cdot y(n-1) + \mu_2(n) 2^{-t_2(n)} \cdot y(n-1) + \mu_3(n) 2^{-t_3(n)} \cdot y(n-1)$
II (Normal Type)	1	2	$x(n) = \eta_1(n) 2^{-s_1(n)} \cdot x(n-1) + \eta_2(n) 2^{-s_2(n)} \cdot x(n-1) - \mu_1(n) 2^{-t_1(n)} \cdot y(n-1)$ $y(n) = \mu_1(n) 2^{-t_1(n)} \cdot x(n-1) + \eta_1(n) 2^{-s_1(n)} \cdot x(n-1) + \eta_2(n) 2^{-s_2(n)} \cdot x(n-1)$
III (Normal Type)	2	1	$x(n) = \eta_1(n) 2^{-s_1(n)} \cdot x(n-1) - \mu_1(n) 2^{-t_1(n)} \cdot y(n-1) - \mu_2(n) 2^{-t_2(n)} \cdot y(n-1)$ $y(n) = \mu_1(n) 2^{-t_1(n)} \cdot x(n-1) + \mu_2(n) 2^{-t_2(n)} \cdot x(n-1) + \eta_1(n) 2^{-s_1(n)} \cdot x(n-1)$
IV (Exchange-Scaling Type)	3	0	$x(n) = -\mu_1(n) 2^{-t_1(n)} \cdot y(n-1) - \mu_2(n) 2^{-t_2(n)} \cdot y(n-1) - \mu_3(n) 2^{-t_3(n)} \cdot y(n-1)$ $y(n) = \mu_1(n) 2^{-t_1(n)} \cdot x(n-1) + \mu_2(n) 2^{-t_2(n)} \cdot x(n-1) + \mu_3(n) 2^{-t_3(n)} \cdot x(n-1)$

4. APPLICATIONS OF CORDIC

In this section, the applications of CORDIC algorithm have been described. Some applications have been discussed in brief while some applications have been discussed in detail as follows.

The most basic applications of CORDIC includes the computation of sine and cosine for a given angle, polar to rectangular transformation, general vector rotation, Cartesian to polar transformation, arctangent computation, computation of vector magnitude, computation of inverse of an already computed function by CORDIC, calculation of arcsine and arccosine etc. CORDIC is also useful in direct frequency synthesis, in sine wave generation [30], motion estimation [31], digital to analog or analog to digital convertor [32], microcontrollers and also in communication technologies such as GSM, WCDMA modulator [33], digital modulation and coding for audio synthesis, direct and inverse kinematics computation for robot manipulation, planer and three dimensional vector rotation for graphics and animation. It is also used in radio frequency identification (RFID) to navigate

the location as shown in [34]. CORDIC algorithms are widely used in digital signal processing (DSP) applications such as computation of discrete Fourier transform (DFT), discrete cosine transform (DCT), discrete sine transform (DST), chirp Z transform (CZT), discrete Hartley transform (DHT) and in designing of digital filters. Some applications are as follows:

CORDIC based FFT implementation:

Fast Fourier Transform processor based on CORDIC [35] is implemented by replacing the sine and cosine twiddle factors in conventional FFT architecture by iterative CORDIC rotations which allows the reduction in read-only memory (ROM). The use of CORDIC in FFT results in the elimination of multipliers saves area, power and cost. The error analysis of CORDIC based FFT has been shown in [36]. There are three types of error in MSR-CORDIC based FFT, one is scaling error due to scaling factor, second is approximation error and the third is round off error due to using finite word length. The simple butterfly structure and its CORDIC based implementation is shown in figure (12), (13) respectively. Let a complex number be C such that

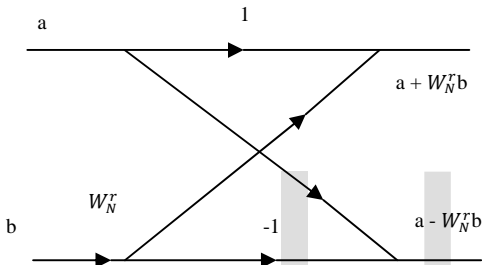


Figure 12: Basic Butterfly Structure

$C = b_R + j b_I$, then

$$C \cdot W_N^{kn} = (b_R + j b_I) \cdot \cos \frac{2\pi}{N} - j (b_R + j b_I) \cdot \sin \frac{2\pi}{N}$$

$$C \cdot W_N^{kn} = (b_R \cos \frac{2\pi}{N} + b_I \sin \frac{2\pi}{N}) + j (b_I \cos \frac{2\pi}{N} - b_R \sin \frac{2\pi}{N})$$

Initially, $X_0 = a$, $Y_0 = b$, $Z_0 = \frac{2\pi k n}{N}$

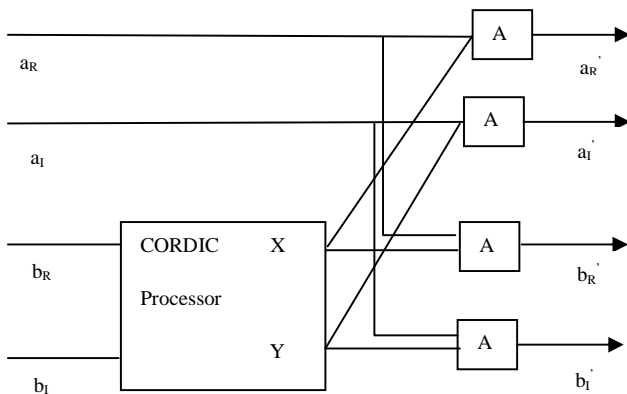


Figure 13: FFT Implementation With CORDIC

In figure (13), $a_i' = a_i + X$, $a_r' = a_r + Y$,

$$b_i' = a_i - X, b_r' = a_r - Y$$

Digital Filter Designing Using CORDIC:

By using CORDIC, orthogonal digital filter can be designed as in [6], [27]. A filter structure is said to be orthogonal if its all internal variables are uncorrelated and have unit variance, assuming a white noise input. Orthogonal filters eliminate overflow oscillations, show low round off noise and invariant

under frequency transformations.

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$, this matrix can be represented as shown in figure (14).

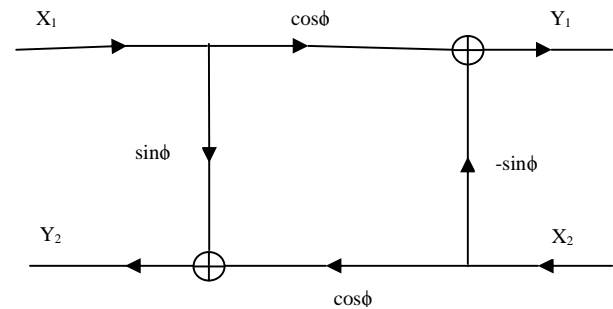


Figure 14: Representation of CORDIC basic equations

Sometimes the above matrix equations can be represented in figure as shown in figure (15).

In [37], the author has used CORDIC for the implementation of finite impulse response. Transfer function approach as well as state space approach, both approaches can be used for filter designing. The schur algorithm [27] is used in designing of orthogonal digital lattice filter. In [38], [39], orthogonal digital filter and orthogonal double rotation (ODR) filter have been proposed respectively. Let the transfer function be

$$T(z) = \frac{N(z)}{D(z)}, \text{ where } N(z) = \sum_{i=0}^n N_i z^{n-i} \text{ and}$$

$$D(z) = \sum_{i=0}^n D_i z^{n-i} \dots \dots (41)$$

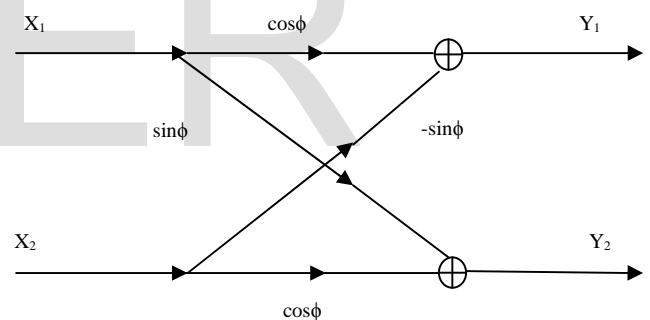


Figure 15: Another Representation for CORDIC equations

The necessary condition for such digital filter is

$$E(z) \cdot E_*(z) = D(z) \cdot D_*(z) - N(z) \cdot N_*(z) \dots \dots (42)$$

$$\text{Where } E_*(z) = z^n E(z^{-1}) \dots \dots (43)$$

$$N_g(z) = \begin{bmatrix} N(z) \\ E(z) \end{bmatrix}, k_0 = \frac{N_g(\infty)}{D_N(\infty)}, \text{ Where } k_0 = \begin{bmatrix} k_{01} \\ k_{02} \end{bmatrix}$$

The new denominator and numerator polynomials of transfer function is

$$D'(z) = z^{-1}(I - k_0' k_0) - 1/2 \{D(z) - k_0' N_g(z)\}, \dots \dots (44)$$

$$N_g'(z) = (I - k_0 k_0') - 1/2 \{N_g(z) - k_0 D(z)\} \dots \dots (45)$$

But in [6], the author has proposed more reliable cascade orthogonal digital IIR filter. In this filter, the individual section of cascade interconnection can be synthesized by using scattering matrix (SM) but to synthesize the whole cascade interconnection, chain scattering matrix (CSM) is used. The two delay sections, degree-1 and degree-2, have been used to reduce the order of cascade interconnected sections by one degree and by two degree respectively depending upon the poles being real or complex. Degree - 1 section has been shown in figure (16).

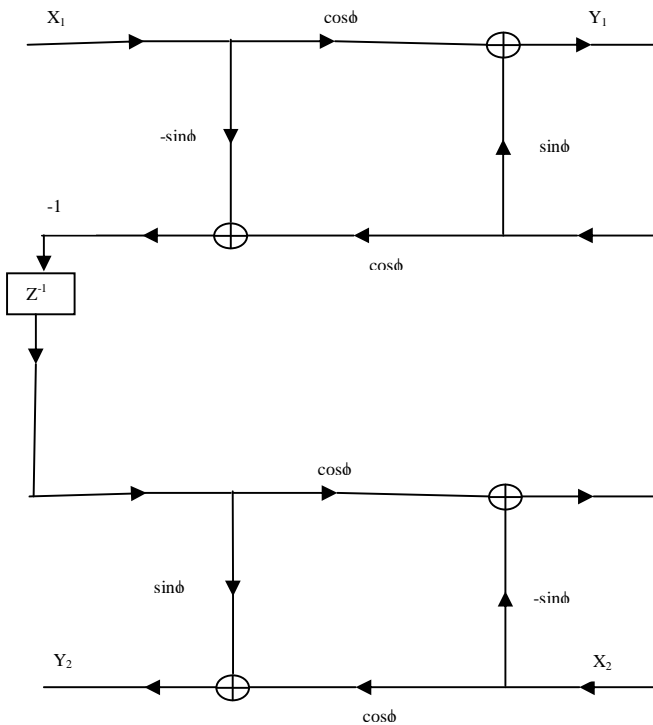


Figure 16: Degree - 1 section with CORDIC architecture for first order reduction

CORDIC ALGORITHM FOR SVD

For singular value decomposition (SVD), a M*M matrix is divided into $\lfloor \frac{M}{2} \rfloor * \lfloor \frac{M}{2} \rfloor$ blocks. Each block is a 2*2 matrix, which can be mapped to a CORDIC processor. The two-sided rotation is applied to each 2*2 matrix to nullify the two off-diagonal elements. Singular value decomposition (SVD) [40] of a matrix is given by $U \Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix of singular values.

$$\begin{bmatrix} \cos \alpha_1 & -\sin \alpha_1 \\ \sin \alpha_1 & \cos \alpha_1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \cos \alpha_2 & -\sin \alpha_2 \\ \sin \alpha_2 & \cos \alpha_2 \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

The values of α_1 and α_2 can be obtained in the following manner.

$$\alpha_1 + \alpha_2 = \tan^{-1} \left(\frac{c+b}{d-a} \right) \dots\dots (46)$$

$$\alpha_2 - \alpha_1 = \tan^{-1} \left(\frac{c-b}{d+a} \right) \dots\dots\dots (47)$$

Application of CORDIC in Communication system

By selecting the required operating mode whether rotational or vectoring, CORDIC can be used in different modulation schemes like amplitude modulation (AM), frequency modulation (FM), phase modulation (PM) and also in digital modulation schemes. CORDIC can be used in (phase locked loop) PLL and direct digital synthesizer (DDS) in the following manner. In ADPLL [41], [42], all components are digitized. The difference between ADPLL & DPLL is that the digital PLL has an analog component called charge pump which translates the phase difference into corresponding voltage, but in an ADPLL, there is no analog components exists. The block diagram of ADPLL is shown in figure (17). The Basic building blocks of ADPLL includes frequency Sampling Filters (Hilbert filter), Phase detector (CORDIC Subsystem), Loop filter (F(z)), Direct Digital Synthesizer.

Phase detection is done by using Hilbert filter & CORDIC algorithm. The phase difference is compensated by the controller or loop filter F(Z). It tunes the frequency of a direct digital synthesizer (DSS).

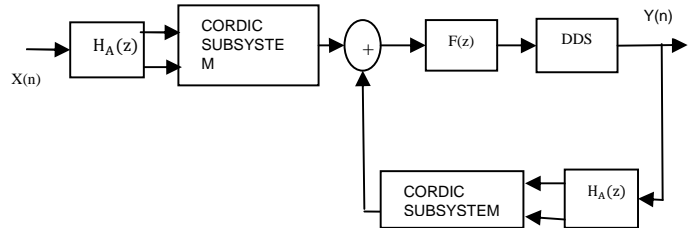


Figure 17: Block diagram of All Digital Phase Locked Loop

CORDIC can be used in direct digital synthesizer (DDS) as shown in figure (18). Direct Digital Synthesis (DDS) is an electronic method for digitally creating arbitrary waveforms and frequencies from a single, fixed source frequency. The main components of DDS are: phase accumulator, phase-to-amplitude conversion (often a sine look-up table) and DAC. The digital word in the phase register, M represents the amount; the phase accumulator is incremented each clock cycle. If fclk is the clock frequency, then the frequency of the output sine wave is calculated as $f_{out} = \frac{M * f_{clk}}{2^n}$

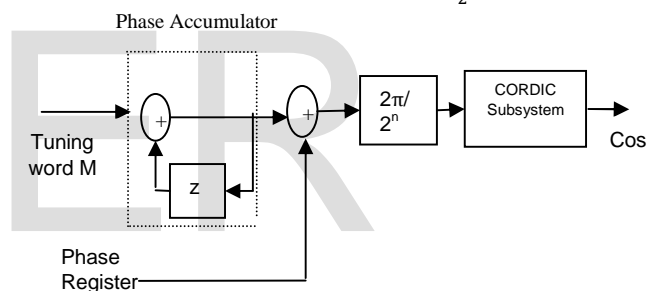


Figure 18: Block diagram of Direct Digital Synthesizer using CORDIC subsystem

5. CONCLUSION

In this paper, we have reviewed the existing CORDIC algorithms, architectures and applications. Some advanced CORDIC algorithms and its DSP based applications have been discussed specially. Due to simplicity of operation, CORDIC technique is preferred in various fields including digital signal processing (DSP), advanced digital signal processing (ADSP), communication, graphics etc. There are many algorithms which are used to reduce the number of iterations but latency of hardware complexity, error performance, signal to quantization ratio (SQNR) etc. are also considerable factors which affect the system performance. Where angle is known in advance and high SQNR is required, mixed scaling rotation (MSR) CORDIC algorithm is most suitable. Where hardware complexity is major issue, generalized micro-rotation algorithm is used. For high throughput applications, pipelined CORDIC architecture is used. In future, we would have more advanced version of existing CORDIC algorithm and it would be useful for designing of high speed processors.

REFERENCES

- [1] J.E. Volder, "The CORDIC trigonometric computing technique", IRE Trans. Electron. Computers, vol. EC-8, pp. 330-334, Sept. 1959.
- [2] J. E. Volder, "The birth of CORDIC," J. VLSI Signal Process., vol. 25, pp. 101-105, 2000.
- [3] J. S. Walthers, "A unified algorithm for elementary functions", in Proc.38th Spring Joint Computer Conf., Atlantic City, NJ, 1971, pp.379-385.
- [4] J. S. Walthers, "The story of unified CORDIC," J. VLSI Signal Process., vol. 25, no. 2, pp. 107-112, June 2000.
- [5] Alvin M. Despain, "Fourier Transform Computers using CORDIC Iterations".
- [6] Jun Ma, Keshab K. Parhi, Ed F. Deprettere, "Pipelined CORDIC-Based Cascade Orthogonal IIR Digital Filter" IEEE transactions on circuits and systems—ii: analog and digital signal processing, vol. 47, no. 11, november 2000.
- [7] Yu Hen Hu, S. Naganathan, "An Angle Recoding Method for CORDIC Algorithm Implementation", IEEE transactions on computers, vol. 42, no. 1, january 1993.
- [8] C.-S. Wu, A.-Y.Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes", IEEE Trans. Circuits Syst. II: Anal. Digital Signal Process, vol. 50, no. 9, pp. 589-601, Sep. 2003.
- [9] Chih-Hsiu Lin and An-Yeu Wu, "Mixed-Scaling-Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for High-Performance Vector Rotational DSP Applications", IEEE Transactions on circuits and systems, vol. 52, no. 11, November 2005.
- [10] Supriya Aggarwal, Pramod K. Meher and Kavita Khare, "Area-Time efficient scaling free CORDIC using generalized micro-rotation selection", IEEE trans. on VLSI, vol. 20, 2012.
- [11] Francisco J. Jaime, Miguel A. Sánchez, Javier Hormigo, Julio Villalba, and Emilio L. Zapata, "Enhanced Scaling-Free CORDIC", IEEE transactions on circuits and systems—i: regular papers, vol. 57, no. 7, July 2010.
- [12] Behrooz Parhami, "Computer arithmetic: algorithms and hardware designs", Oxford University Press, 2000 edition.
- [13] Krishna Raj, Praveen Kumar Singh, Rajkumar Tomar, "A Review of Low Cost Multiplier Using CORDIC Subsystem", 2012 2nd International Conference on Power, Control and Embedded Systems, Dec.17, 2012.
- [14] Krishna Raj, Praveen Kumar Singh, Rajkumar Tomar, "Power Saving in FIR Filter using CORDIC Subsystem as a Multiplier", National Conference on Electronics & Communication Systems, April 5, 2013, IPEC Ghaziabad.
- [15] Pongyupinpanich Surapong, Faizal Arya Samman and Manfred Glesne, "Design and Analysis of Extension-Rotation CORDIC Algorithms based on Non-Redundant Method", International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 5, No.1, March, 2012.
- [16] Ramesh Bhakthavathalu, Parvathi Nair, "Low Power Design Techniques Applied to Pipelined Parallel and Iterative CORDIC Design", 978-1-4244-8679-3/11/\$26.00 ©2011 IEEE.
- [17] Ray Andra, Andra consulting group, "A Survey of CORDIC algorithms for FPGA based computers", Andra consulting group, North Kingston.
- [18] R. d. Singleton, "An algorithm for computing the mixed radix fast Fourier transform", IEEE Trans. Audio Electroacoust, vol. AU-15, pp. 45-55, June 1967.
- [19] O. Buneman, "Inversion of the Heinholtz (or Laplace-Poisson) operator for slab geometry", Inst. for Plasma Res., Stanford Univ., Stanford, Calif, SUIPR Rep. 467, p. 5, Apr. 1972.
- [20] P. K. Meher, J. Walls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures and applications", IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 9, pp. 1893-1907, Sep. 2009.
- [21] T. K. Rodrigues and E. E. Swartzlander, "Adaptive CORDIC: Using parallel angle recoding to accelerate CORDIC rotations", in 40th Asilomar Conf. on Signals, Syst. and Computers, ACSSC'06, Oct.-Nov. 2006, pp. 323-327.
- [22] S. Wang, V. Piuri, and E. E. Swartzlander Jr., "Hybrid CORDIC algorithms", IEEE Transactions on Computers, vol. 46, no. 11, pp. 1202-1207, 1997.
- [23] B. Lakshmi and A. S. Dhar, "CORDIC Architectures: A Survey", Hindawi Publishing Corporation VLSI Design Volume 2010, Article ID 794891, 19 pages.
- [24] Alvin M. Despain, "Fourier Transform Computers using CORDIC Iterations", IEEE transactions on computers, VOL. c-23, NO. 10, OCTOBER 1974.
- [25] Uwe Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Springer Series on Signals and Communication Technology, Third Edition.
- [26] K.K. Parhi, "VLSI Digital Signal Processing, Design and Implementation", Wiley interscience publications.
- [27] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," IEEE Trans. Computers, vol. 45, no. 3, pp. 307-318, Mar. 1996.
- [28] Krishna Raj, Praveen Kumar Singh, Rajkumar Tomar, "Arithmetic of CORDIC and its Scaling Free Operation using Micro-rotation Selection", National Conference on Electronics & Communication Systems, April 5, 2013, IPEC Ghaziabad.
- [29] Shen-Fu Hsiao, Yu-Hen Hu, Tso-Bing Juang, "A Memory-Efficient and High-Speed Sine/Cosine Generator Based on Parallel CORDIC Rotations", IEEE signal processing letters, vol. 11, no. 2, february 2004.
- [30] Jie Chen, K. J. Ray Liu, "A Complete Pipelined Parallel CORDIC Architecture for Motion Estimation", IEEE transactions on circuits and systems—ii: analog and digital signal processing, vol. 45, no. 6, June 1998.
- [31] M. B. Yearly, Rainer J. Fink, Harini Sundaresan, David W. Guidry, "Design of a CORDIC Processor for Mixed-Signal A/D Conversion", IEEE transactions on instrumentation and measurement, vol. 51, no. 4, August 2002.
- [32] Jouko Vankka, Jaakko Ketola, Johan Sommarek, Olli Väinänen, Marko Kosunen, Kari A. I. Halonen, "A GSM/EDGE/WCDMA Modulator With On-Chip D/A Converter for Base Stations", IEEE transactions on circuits and systems—ii: analog and digital signal processing, vol. 49, no. 10, October 2002.
- [33] Yen-Chang Huang, Chien-Chang Lai and Yu-Jung Huang, "Hardware Implementation of Triangulation Method Based on CORDIC Algorithm", 978-1-4244-6694-8/10/\$26.00 ©2010 IEEE.
- [34] Pooja Choudhary, Dr. Abhijit Karmakar, "CORDIC Based Implementation of Fast Fourier Transform", International Conference on Computer & Communication Technology (ICCCCT)-2011.
- [35] Sang Yoon Park and Ya Jun Yu, "Fixed-Point Analysis and Parameter Selections of MSR-CORDIC with Applications to FFT Designs", IEEE Transactions on signal processing, Aug. 2012.
- [36] Krzysztof Wawryn, Robert T. Wirski, and Bogdan Strzeszewski, "Implementation of Finite Impulse Response Systems Using Rotation Structures", ISITA 2010, Taichung, Taiwan, October 17-20, 2010.
- [37] Sailesh K. Rag and Thomas Kailath, "Orthogonal Digital Filters for VLSI Implementation", IEEE Transactions on circuits and systems, vol. CAS-31, no. 11, November 1984.
- [38] Jin-Gyun Chung, and Keshab K. Parhi, "Pipelining of Orthogonal Double-Rotation Digital Lattice Filters", 1058-6393/93 \$03.00 Q 1993 IEEE.
- [39] Joseph R. Cavallaro and Anne C. Elster, "A CORDIC Processor Array for the SVD of a Complex Matrix", Elsevier Science Publishers, 1991.
- [40] Abhishek Das, Suraj Dash, B. Chitti Babu and Ajit Kumar Sahoo, "A Novel Phase Detection System for Linear All-Digital Phase Locked Loop", 978-1-4673-0455-9/12/\$31.00 ©2012 IEEE.
- [41] T. Menakadevi, M. Madheswaran, "Direct Digital Synthesizer using Pipelined CORDIC Algorithm for Software Defined Radio", International Journal of Science and Technology, Volume 2 No.6, June 2012, ISSN 2224-3577.