A Fast and an Accurate CORDIC Modulus Extractor Implementation Using FPGA

Mohammed Rizk, Ahmed Hossin, and Alaa Hafez

Abstract—The paper is devoted to design and implement a fast and accurate CORDIC modulus extractor for radar MTD using field programmable gate array FPGA. Modulus extractor receives real and imaginary part from Doppler filter bank and produce a modulus output suitable for amplitude processing by constant false alarm rate processor CFAR. This implementation contributes a fast propagation delay for this extractor and gives an accurate results for the modulus and angle output. This prototype of hardware implementation of CORDIC algorithm used Spartan –III series FPGA, with constraint to area efficiency and throughput architecture. The extractor results show that the conversion time is with less than 0.025% in angle measurement and 0.9% accuracy in modulus measurement which is suitable for real time applications in radar and missile control applications.

Index Terms— MTD, Modulus extractor, CORDIC

1 Introduction

In the last decade, CORDIC algorithm has drawn wide attention from academia and industry for various applications such as DSP, biomedical signal processing, software defined radio, neural networks, and MIMO systems to mention just a few. It is an iterative algorithm, requiring simple shift and addition operations, for hardware realization of basic elementary functions. Since CORDIC is used as a building block in various single chip solutions, the critical aspects to be considered are high speed, low power, and low area, for achieving reasonable overall performance.

CORDIC algorithm provides an efficient way to estimate the basic elementary functions like trigonometric operations, multiplication, division and some other operations like logarithmic functions, square roots and exponential functions. Most of the applications either in wireless communication or in digital signal processing are based on microprocessors which make use of a single instruction and a bunch of addressing modes for their working. As these processors are costs efficient and offer extreme flexibility but yet are not suited for some of these applications. The CORDIC algorithm has received increased attention after a unified approach is proposed for its implementation [1-2].

The CORDIC arithmetic processor chip is designed and implemented to perform various functions possible in rotation and vectoring mode of circular, linear, and hyperbolic coordinate systems [3]. Since then, CORDIC technique has been used in many applications [4], such as single chip CORDIC processor for DSP applications [5–6]. Recently several researches applied CORDIC algorithm in radar pulse compression, rotary encoders, and waveform generation [7-12]. This paper

2 Cordic Algorithm

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. The CORDIC algorithm performs a planar rotation. Graphically, planar rotation means transforming a vector (Xi, Yi) into a new vector (Xj, Yj)[13-14].

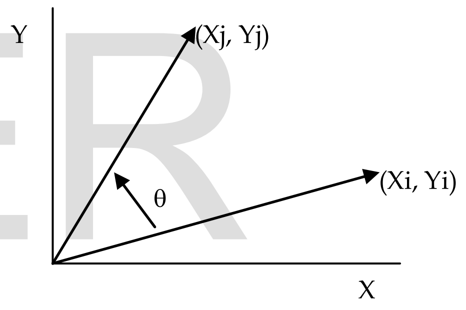


Fig. 1. Rotating the vector

Using a matrix form, a planar rotation for a vector of (Xi, Yi) is defined as

$$\begin{bmatrix} X_j \\ Y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{1}$$

The □ angle rotation can be executed in several steps, using an iterative process. Each step completes a small part of the rotation. Many steps will compose one planar rotation. A single step is defined by the following equation:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{2}$$

Equation 2 can be modified by eliminating the $\cos\theta_n$ factor.

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{3}$$

Equation 3 requires three multiplies, compared to the four needed in equation 2. Additional multipliers can be eliminated by selecting the angle steps such that the tangent of a step is a power of 2. Multiplying or dividing by a power of 2 can be implemented using a simple shift operation. The angle for each step is given by

$$\theta_n = \arctan\left(\frac{1}{2^n}\right) \tag{4}$$

All iteration-angles summed must equal the rotation angle □.

$$\sum_{n=0}^{\infty} S_n \theta_n = \theta \tag{5}$$

where

$$S_n = \{-1;+1\} \tag{6}$$

This results in the following equation for $\tan\theta_n$

$$\tan\theta_n = S_n 2^{-n} \tag{7}$$

Combining equation 3 and 7 results in

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{8}$$

Besides for the $\cos\theta_n$ coefficient, the algorithm has been reduced to a few simple shifts and additions. The coefficient can be eliminated by pre-computing the final result. The first step is to rewrite the coefficient.

$$\cos\theta_n = \cos\left(\arctan\left(\frac{1}{2^n}\right)\right) \tag{9}$$

The second step is to compute equation 9 for all values of 'n' and multiplying the results, which we will refer to as K.

$$K = \frac{1}{P} = \prod_{n=0}^{\infty} \cos\left(\arctan\left(\frac{1}{2^n}\right)\right) \approx 0.607253 \tag{10}$$

K is constant for all initial vectors and for all values of the rotation angle, it is normally referred to as the congregate constant. The derivative P (approx. 1.64676) is defined here because it is also commonly used. We can now formulate the exact calculation the CORDIC performs.

$$\begin{cases} X_j = K\left(X_i \cos\theta - Y_i \sin\theta\right) \\ Y_j = K\left(Y_i \cos\theta + X_i \sin\theta\right) \end{cases} \tag{11}$$

Because the coefficient K is pre-computed and taken into account at a later stage, equation 8 may be written as

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{12}$$

or as

$$\begin{cases} X_{n+1} = X_n - S_n 2^{-2n} Y_n \\ Y_{n+1} = Y_n + S_n 2^{-2n} X_n \end{cases} \tag{13}$$

At this point a new variable called 'Z' is introduced. Z represents the part of the angle □ which has not been rotated yet.

$$Z_{n+1} = \theta - \sum_{i=0}^{n} \theta_i \tag{14}$$

For every step of the rotation Sn is computed as a sign of Zn.

$$S_n = \begin{cases} -1 & if\ Z_n < 0 \\ +1 & if\ Z_n \geq 0 \end{cases} \tag{15}$$

Combining equations 5 and 15 results in a system which reduces the not rotated part of angle □ tozero.
Or in a program-like style:

```
For n=0 to [inf]
        If (Z(n) >= 0) then
                Z(n + 1) := Z(n) – atan(1/2^n);
        Else
                Z(n + 1) := Z(n) + atan(1/2^n);
        End if;
End for;
```

The atan(1/2^i) is pre-calculated and stored in a table. [inf] is replaced with the required number of iterations, which is about 1 iteration per bit (16 iterations yield a 16bit result).
If we add the computation for X and Y we get the program-like style for the CORDIC core.

```
For n=0 to [inf]
        If (Z(n) >= 0) then
                X(n + 1) := X(n) – (Yn/2^n);
                Y(n + 1) := Y(n) + (Xn/2^n);
                Z(n + 1) := Z(n) – atan(1/2^n);
        Else
                X(n + 1) := X(n) + (Yn/2^n);
                Y(n + 1) := Y(n) – (Xn/2^n);
                Z(n + 1) := Z(n) + atan(1/2^n);
        End if;
End for;
```

This algorithm is commonly referred to as driving Z to zero. The CORDIC core computes:
$\left[X_j, Y_j, Z_j\right] = \left[P\left(X_i\cos(Z_i) - Y_i\sin(Z_i)\right), P\left(Y_i\cos(Z_i) + X_i\sin(Z_i)\right), 0\right]$ Th
ere's a special case for driving Z to zero:

$$X_i = \frac{1}{P} = K \approx 0.60725$$
$$Y_i = 0$$
$$Z_i = \theta$$
$$\left[X_j, Y_j, Z_j\right] = \left[\cos\theta, \sin\theta, 0\right]$$

Another scheme which is possible is driving Y to zero. The CORDIC core then computes:

$$\left[X_j, Y_j, Z_j\right] = \left[P\sqrt{X_i^2 + Y_i^2},\ 0,\ Z_i + \arctan\left(\frac{Y_i}{X_i}\right)\right]$$

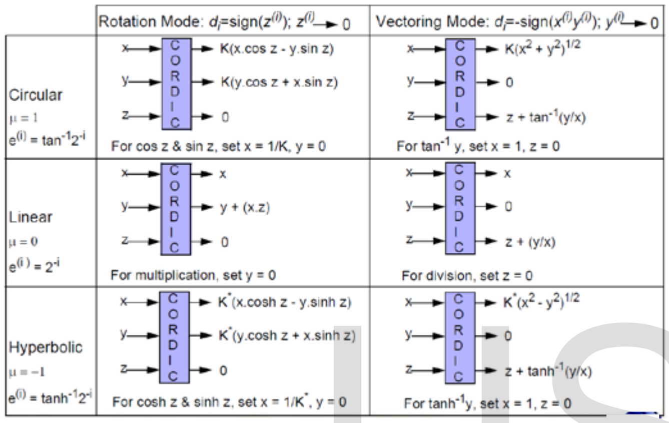
For this scheme there are two special cases:

1) $X_i = X$

$$Y_i = Y$$
$$Z_i = 0$$
$$[X_j, Y_j, Z_j] = \left[P\sqrt{X_i^2 + Y_i^2},\ 0,\ \arctan\left(\frac{Y_i}{X_i}\right) \right]$$

2)
$$X_i = 1$$
$$Y_i = a$$
$$Z_i = 0$$
$$[X_j, Y_j, Z_j] = \left[P\sqrt{1 + a^2},\ 0,\ \arctan(a) \right]$$

As a Summary of CORDIC Functions is illustrated in table (1)
Table (1): Summary of CORDIC Algorithm



	Rotation Mode: $d_i = \text{sign}(z^{(i)})$; $z^{(i)} \rightarrow 0$	Vectoring Mode: $d_i = -\text{sign}(x^{(i)} y^{(i)})$; $y^{(i)} \rightarrow 0$
Circular $\mu = 1$ $e^{(i)} = \tan^{-1} 2^{-i}$	x, y, z → CORDIC → $K(x.\cos z - y.\sin z)$, $K(y.\cos z + x.\sin z)$, 0 For cos z & sin z, set x = 1/K, y = 0	x, y, z → CORDIC → $K(x^2 + y^2)^{1/2}$, 0, $z + \tan^{-1}(y/x)$ For $\tan^{-1}$ y, set x = 1, z = 0
Linear $\mu = 0$ $e^{(i)} = 2^{-i}$	x, y, z → CORDIC → x, $y + (x.z)$, 0 For multiplication, set y = 0	x, y, z → CORDIC → x, 0, $z + (y/x)$ For division, set z = 0
Hyperbolic $\mu = -1$ $e^{(i)} = \tanh^{-1} 2^{-i}$	x, y, z → CORDIC → $K^*(x.\cosh z - y.\sinh z)$, $K^*(y.\cosh z + x.\sinh z)$, 0 For cosh z & sinh z, set x = 1/K*, y = 0	x, y, z → CORDIC → $K^*(x^2 - y^2)^{1/2}$, 0, $z + \tanh^{-1}(y/x)$ For $\tanh^{-1}$y, set x = 1, z = 0

3 The Proposed Architecture

The MTD is an enhanced configuration of moving-target indicator (MTI) that combines a series of features to improve clutter rejection and target detection. The main features in an MTD are the MTI precanceler, the doppler filter bank, use of burstto-burst PRF diversity, adaptive thresholding, and the clutter map as shown in Fig. 2.

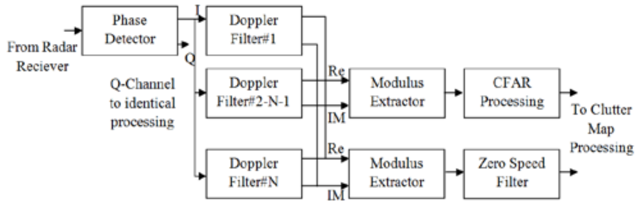


Fig. 2: MTD system block diagram

The doppler filter bank is typically based on the FFT algorithm, providing the following advantages over a delayline canceler: (1) the signal-to-noise ratio is improved by coherent integration within each of the n filters, whose bandwidth will be 1/n that of the canceler; (2) doppler frequency measurement is available, based on the filter number in which detection occurs; (3) the filter bandwidth can be adjusted by amplitude or frequency weighting (windowing), giving better range sidelobe reduction; (4) adaptive thresholding can be applied to each filter, permitting rejection of moving clutter .

The adaptive thresholding applies separate thresholds to each filter: the nonzero velocity channels use a range-cell-averaging CFAR to adapt to moving clouds of precipitation, while the zero-velocity channel threshold is generated by the clutter map, which applies a separate threshold for each range cell. In the zero-velocity channel, , targets at zero radial velocity and at the blind speeds can be detected if they exceed by a sufficient margin the clutter stored in the map. The MTD is essentially a low-PRF pulsed doppler processor. Modulus extractor receives real and imaginary part from Doppler filter bank and produces a modulus output suitable for amplitude processing by constant false alarm rate processor CFAR.

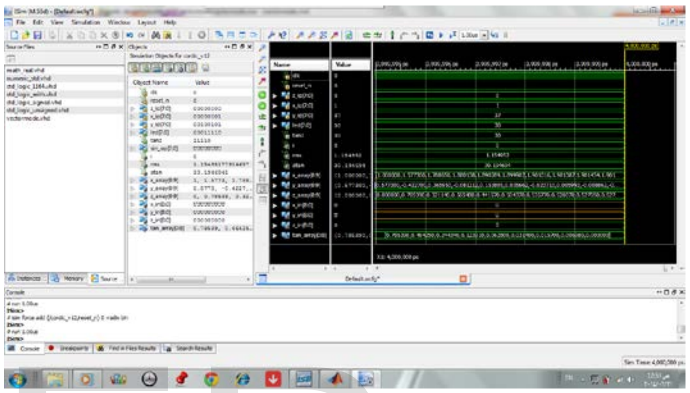


Fig. 3. VHDL simulation of the CORDIC at 30 degree

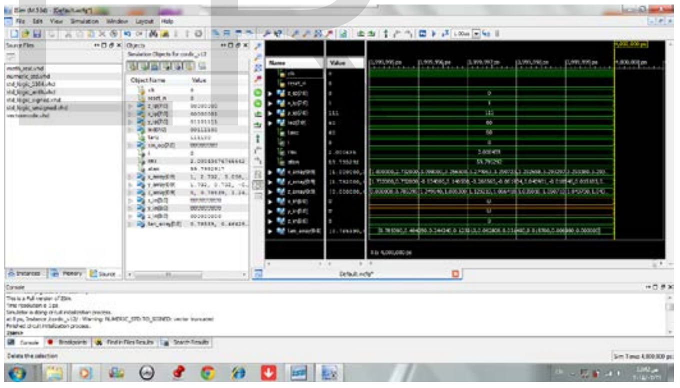


Fig. 4. VHDL simulation of the CORDIC at 60 degree

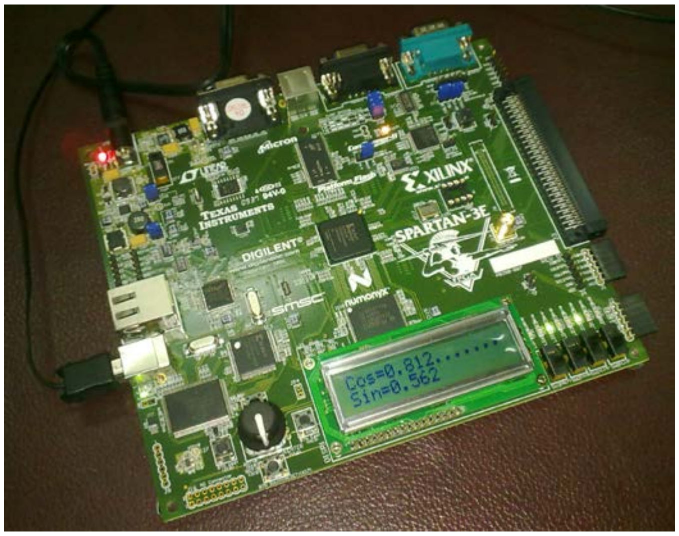


Fig. 5. CORDIC FPGA Implementation

The CORDIC algorithm is implemented into the FPGA to produce modulus output and the angle. The VHDL simulation is demonstrated in Fig. 3-4. The FPGA implementation shown in Fig. 5. Fig. 6. shows CORDIC measured compared with calculated angle concluded the measurement accuracy of the angle. Measured compared with calculated modulus data shown in Fig. 7. Angle Measurement error and Modulus output Measurement error are demonstrated in Fig. 7,8. Angle Measurement error percentage shown in Fig. 9. With maximum error of 0.025%. Fig. 11. Shows Modulus output Measurement error Percentage wit maximum error of 0.9%.

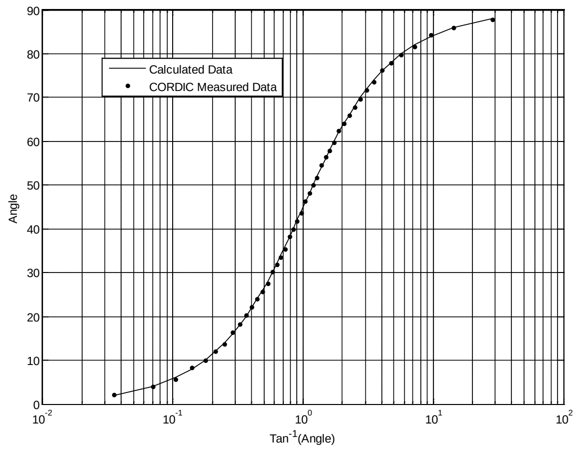


Fig. 6. CORDIC measured compared with calculated angle

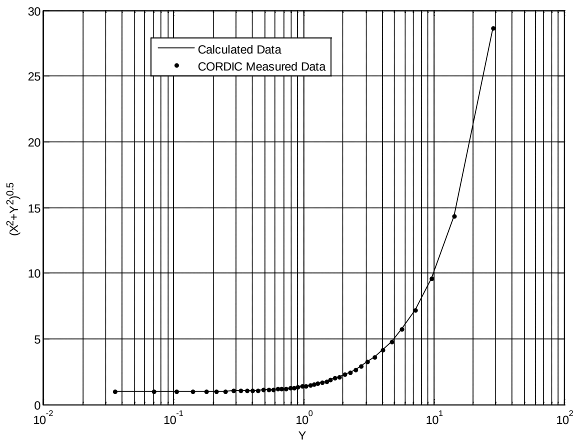


Fig. 7. Measured compared with calculated modulus data

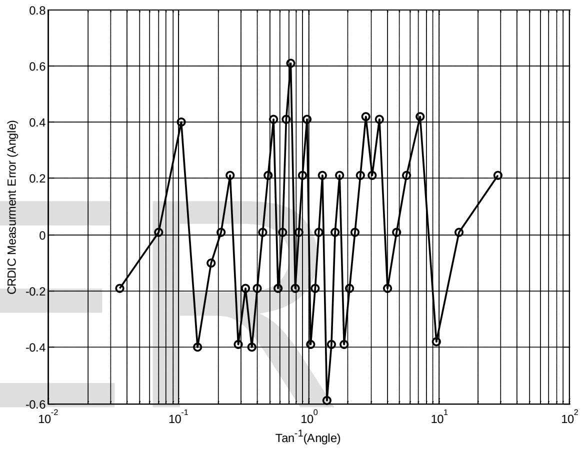


Fig. 8. Angle Measurement error

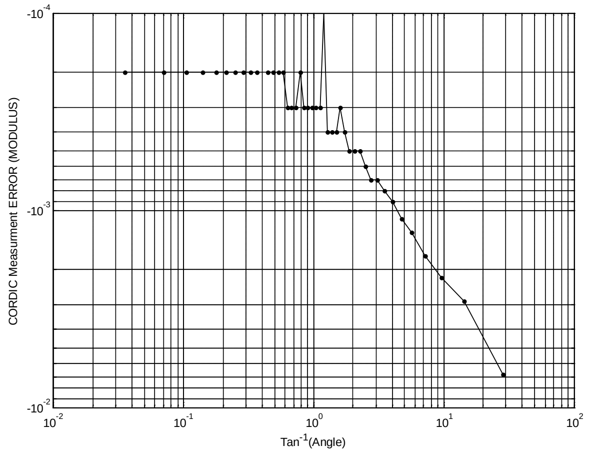


Fig. 9. Modulus output Measurement error

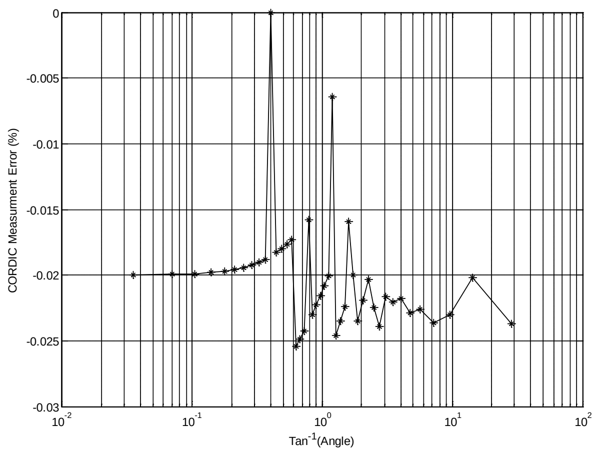


Fig. 10. Angle Measurement error percentage

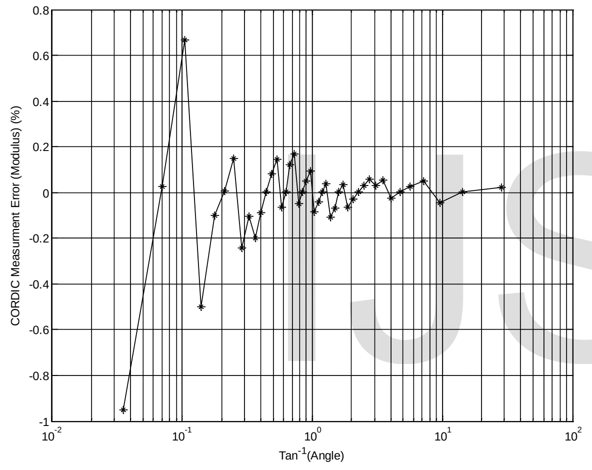


Fig. 11. Modulus output Measurement error Percentage

4 Conclusion

This paper design and implement a fast and accurate CORDIC modulus extractor for radar MTD using field programmable gate array FPGA. This implementation contributes a fast propagation delay for this extractor and gives an accurate results for the modulus and angle output. This prototype of hardware implementation of CORDIC algorithm used Spartan –III series FPGA, with constraint to area efficiency and throughput architecture. The extractor results show that the conversion time is less than 1μs with less than 0.025% in angle measurement and 0.9% accuracy in modulus measurement which is suitable for real time applications in radar and missile control applications.

References

[1] Jack E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron Computers, vol. EC-8, pp. 330–334, Sept. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in Proceedings of the AFIPS Spring Joint Computer Conference, May 1971.

[3] G. L. Haviland and A. A. Tuszynski, "A CORDIC arithmetic processor chip," IEEE Journal of Solid-State Circuits, vol. 15, no. 1, pp. 4–15, 1980.

[4] Y. H.Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE Signal Processing Magazine, vol. 9, no. 3, , 1992.

[5] A. A. J. de Lange, A. J. van der Hoeven, E. F. Deprettere, and J. Bu, "Optimal floating-point pipeline CMOS CORDIC processor," in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '88), vol. 3, pp. 2043–2047, June 1988.

[6] A. A. J. de Lange and E. F. Deprettere, "Design and implementation of a floating-point quasi-systolic general purpose CORDIC rotator for high-rate parallel data and signal processing," in Proceedings of the 10th IEEE Symposium on Computer Arithmetic, pp. 272–281, June 1991.

[7] Yu, Ji-yang, Huang, Dan; Pei, Nan; Zhao, Siyang; Guo, Jian; Xu, Yong "CORDIC-based design of matched filter weighted algorithm for pulse compression system" 2012 IEEE 11th International Conference on Signal Processing (ICSP), Beijing, China, Oct. 2012

[8] Misans, P. Derums, U.; Kanders, V., "FPGA implementation of elementary generalized unitary rotation with CORDIC based architecture," NORCHIP, Nov. 2012.

[9] Causo, M.,"Parallel scaling-free and area-time efficient CORDIC algorithm," 2012 19th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Paris, France, Dec., 2012.

[10] Dezhi Zheng, Shaobo Zhang; Yuming Zhang; Chen," Fan Application of CORDIC in capacitive rotary encoder signal demodulation," 2012 8th IEEE International Symposium on Instrumentation and Control Technology (ISICT), July 2012.

[11] Aggarwal, S. Khare, K.,"Efficient Window-Architecture Design Using Completely Scaling-Free CORDIC Pipeline," 2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), Jan. 2013.

[12] Aggarwal, S. Meher, P.K.; Khare, K.,"Scale-Free Hyperbolic CORDIC Processor and Its Application to Waveform Generation," IEEE Transactions on Circuits and Systems I, Vol. 60, No. 2, Feb. 2013.

[13] B. Lakshmi and A. S. Dhar, "CORDIC Architectures: A Survey," Hindawi Publishing Corporation VLSI Design Volume 2010, Article ID 79489.

[14] Er. Manoj Arora, Er. R S Chauhan, Er.Lalit Bagga," FPGA Prototyping of Hardware Implementation of CORDIC Algorithm," International Journal of Scientific & Engineering Research, Volume 3, Issue 1, January-2012.