Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2015, 2(6): 98-102

Research Article ISSN: 2394 - 658X

VHDL Implementation of DIT-FFT using CORDIC

M Paavani[1], A Naga Malli[1] and Kiranmayi[2]

[1] *Department of Electronics and Communication,*
Gayatri Vidya Parishad College of Engineering, Visakhapatnam, India
[2]*Senior IT Engineer, CMC Limited, Hyderabad*
mugadapavani113@gmail.com

ABSTRACT

Evaluating trigonometric functions, used in Digital Signal Processing applications like Discrete Fourier Transform (DFT), Discrete Hartley Transform (DHT) is a complicated task involving large area, high complexity and time issues. To reduce the delay at the expense of area constraints an optimized hardware efficient technique called Coordinate Rotation Digital Computer (CORDIC) is used. CORDIC technique is implemented, without use of multiplier, which eases the process. This journal presents how CORDIC technique is implemented in pipelined mode for different applications such as computing trigonometric functions to determine FFT, data conversion.

Key words: Coordinate Rotation Digital Computer (CORDIC), Discrete Fourier Transform (DFT), Discrete Hartley Transform (DHT), FFT

INTRODUCTION

CORDIC (CO-ordinate Rotation Digital Computer) is a unique reason computerized PC for calculations progressively. CORDIC calculation was initially presented by Jack E Volder in the year 1959 [1-4]. CORDIC is very proficient, low unpredictability and hearty procedure for figuring basic capacities. CORDIC calculation has its applications in pocket adding machines, sign handling (DFT, DCT, DHT and DST). Unlike general methods which make use of multiplier, CORDIC makes use of only Adders/ Subtractor and Look-Up tables. The main principle of CORDIC algorithm is to rotate the vector through a fixed angle to get new vectors. This technique is often useful in applications like data conversions, DFT implementation etc. CORDIC uses two modes for rotation of a vector [1-3]. They are

(i) Rotation Mode and (ii) Vectoring Mode

In Rotation mode, the co-ordinate part of the vector and angle of rotation are given. For a number of iterations the vector is rotated until the angle initialized converges to zero resulting in new vector.

In vectoring mode, the co-ordinate parts of a vector are given and the magnitude and angle are registered. For this situation the second co-ordinate component focalizes to zero. Generally CORDIC structure is composed of only Adder/Subtractor and lookup table.

CORDIC ARCHITECTURES

There are three architectures in which CORDIC Algorithm can be implemented. Each architecture has its own advantages and disadvantages.

Sequential /Iterative CORDIC

Fig. 1 shows the Iterative CORDIC or Sequential CORDIC as the iterations are done sequentially one after the other. The outputs/inputs are stored in the registers as shown in the figure Fig.1. The registers are initialized with input vectors and input angle. The input vectors are shifted based on the number of rotations by comparing the input angle with angle in the look up table and finally added or subtracted [2].

Parallel / Cascaded CORDIC

Parallel/Cascaded CORDIC is shown in figure Fig.2. The iterations are done in parallel; the outputs are immediately transferred to the next stage .The shift registers are increased by one bit for every stage. The input vectors are shifted based on the number of rotations and finally added or subtracted. The input vectors are shifted based on the number of rotations by comparing the input angle with angle in the look up table and finally added or subtracted.

Pipelined CORDIC

It is the most efficient of the CORDIC algorithms in which the iterations take place in multiple clock cycles. However, different processes take place concurrently such that the execution time is reduced. The structure of a pipelined architecture is as shown in the following figure Fig. 3 [1-3].

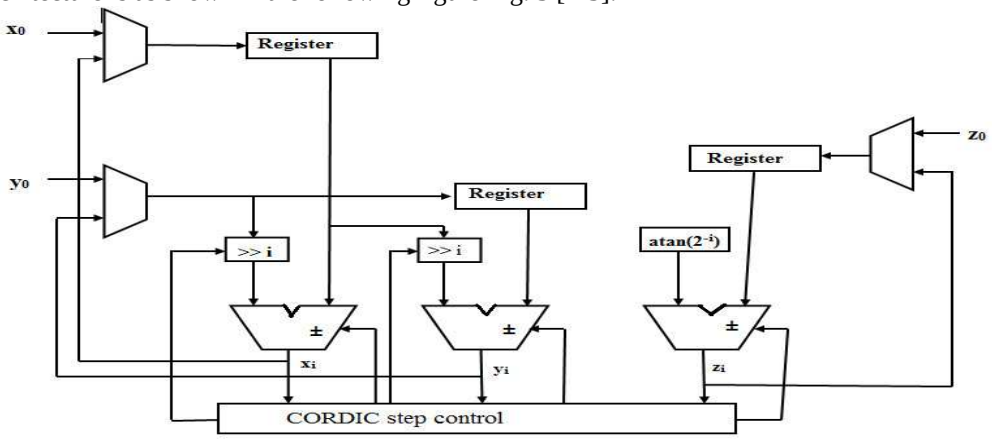

Fig.1 Sequential / Iterative CORDIC [2]

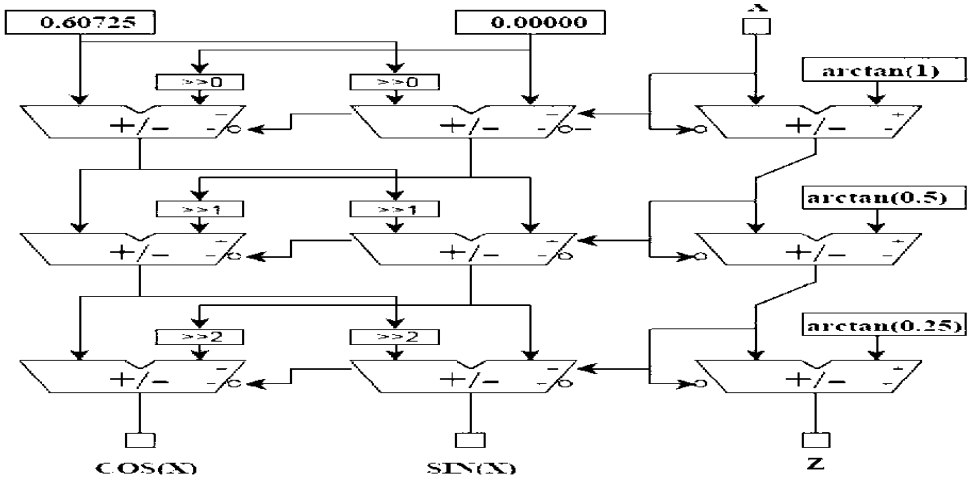

Fig.2 Parallel / Cascaded CORDIC [5]

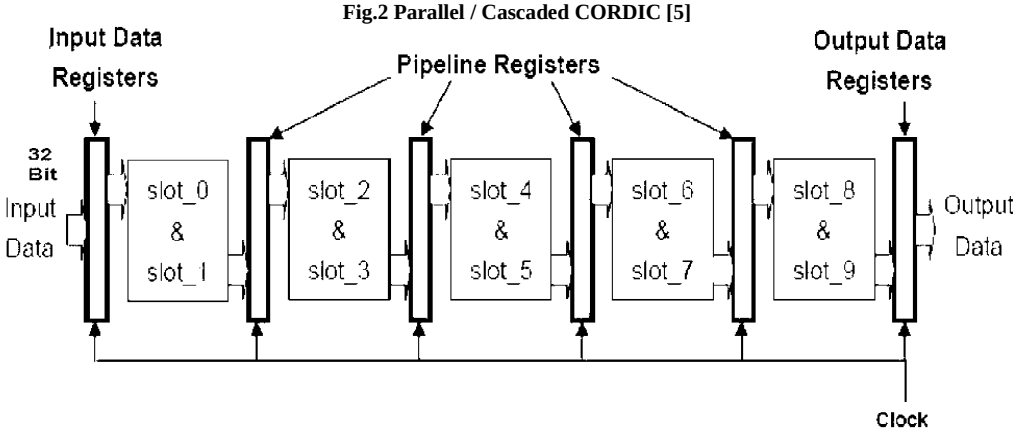

Fig.3 Pipelined CORDIC

CORDIC ALGORITHM FOR VECTOR ROTATION

Vector rotation finds its application in data conversions such as Polar to Rectangular and Rectangular to Polar, computing Trigonometric thereby computing DFT. CORDIC uses two modes for computing the trigonometric functions.

All the vector rotations are derived from the following equations

$$X1 = x \cos \Phi - y \sin \Phi \quad \text{and} \quad Y1 = y \cos \Phi + x \sin \Phi \text{ [2]}$$

By eliminating cos Φ and by substituting tan (Φ) = 2^{-i}, we get

$$X1 = k_i(x_i - y_i d_i 2^{-i}) \quad \text{and} \quad Y1 = k_i(y_i + x_i d_i 2^{-i}) \quad \text{where } k_i = \cos\left(\tan^{-1}(2^{-i})\right) \text{ and } d_i = \pm 1$$

This process of removing the cos Φ term is called pseudo rotation. To rotate the vector originally, we need to include the cos Φ term, this can be done by just multiplying the 0.60725 to the final result, as the cos Φ term on an average yields to 0.60725 from the below equation.

$$K = \prod_{i=0}^{n-1} k_i \quad \text{where} \quad \prod_{i=0}^{n-1} k_i = \text{Cos}(\Phi_0) \cdot \text{Cos}(\Phi_1) \cdot \text{Cos}(\Phi_2) \ldots\ldots\ldots \text{Cos}(\Phi_n)$$

For recording the angles used for number of rotations, a lookup table is used in which the tan (Φ) = 2^{-i} is loaded. A better conversion method uses an adder/ subtractor that accumulate the elementary rotating angles at each iteration. The angle accumulator adds a third difference equation to the CORDIC algorithm.

Rotation Mode

In the rotation mode, the angle accumulator is initialized with the desired rotation angle (zo = Θ). 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.

For rotation mode, the CORDIC equations are

$$x_{i+1} = (x_i - y_i d_i 2^{-i}), \quad y_{i+1} = (y_i + x_i d_i 2^{-i}) \quad \text{and} \quad z_{i+1} = \left(z_i - d_i tan^{-1}(2^{-i})\right)$$

d_i = ±1 (d_i is the direction of angle of rotation) $\quad d_i = \begin{cases} +1, & if\ Zi \geq 0 \\ -1, & if\ Zi \leq 0 \end{cases}$

Vectoring Mode

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 transverse angle at the end of the iteration.

In vectoring mode, the CORDIC equations are

$$x_{i+1} = (x_i - y_i d_i 2^{-i}), \; y_{i+1} = (y_i + x_i d_i 2^{-i}) \text{ and } z_{i+1} = \left(z_i - d_i tan^{-1}(2^{-i})\right)$$

d_i = ±1 (d_i is the direction of angle of rotation) $\quad d_i = \begin{cases} +1, & if\ Yi < 0 \\ -1, & if\ Yi \geq 0 \end{cases}$

In vectoring mode, *y* converges to zero.

Polar To Rectangular Conversion

Polar to rectangular conversion using CORDIC algorithm is implemented in Rotation mode. The equations are defined b $\quad X = r(\cos \Phi) \quad$ and $\quad Y = r(\sin \Phi)$

By rotating an input vector through an angle we get a resultant vector, the input vectors are rotated until the angle converges to zero. The polar input is transformed to the rectangular coordinates. To eliminate the pseudo rotation, the output must me multiplied by the gain obtained.

Rectangular To Polar Conversion

Rectangular to Polar conversion using CORDIC is done in the vectoring mode. The inputs are applied in the rectangular transformation and the output is resulted in the polar coordinates. The outputs are the radius and the angle. Where the radius is obtained as

$$X_n = K_n\sqrt{x^2 + y^2}, \qquad Y_n = 0 \qquad \text{and} \qquad Z_n = \left(Z + tan^{-1}(\tfrac{Y}{X})\right)$$

Sine and Cosine

It is also done in the rotation mode. The inputs are rotated through the fixed angle until the converges to zero by adding/subtracting the angle initialized with the angles in the lookup table. Then all the output vectors are together resulted as cos Φ and sin Φ [3]. The elementary functions sine and cosine can be computed using rotation mode of the CORDIC algorithm if the initial vector is of unit length and is aligned with abscissa (x-axis).

$$X_0 = 1, \qquad Y_0 = 0, \qquad Z_0 = \Theta$$

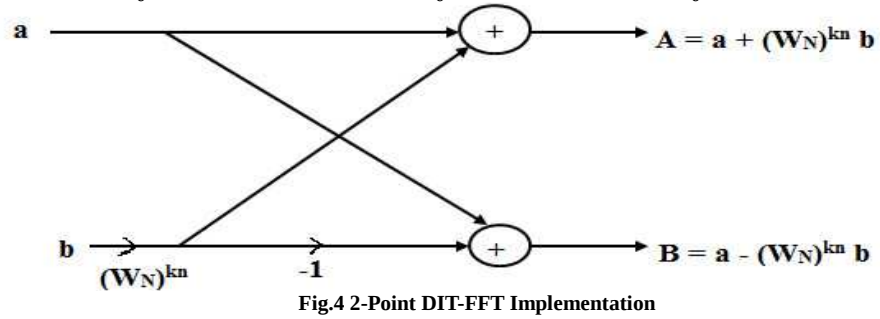

Fig.4 2-Point DIT-FFT Implementation

FFT IMPLEMENTATION

Fast Fourier change (FFT) is a proficient calculation for processing the Discrete Fourier change (DFT). FFT requires less augmentation than a basic methodology of figuring DFT. The Decimation in time (DIT) calculation is utilized for FFT calculation.

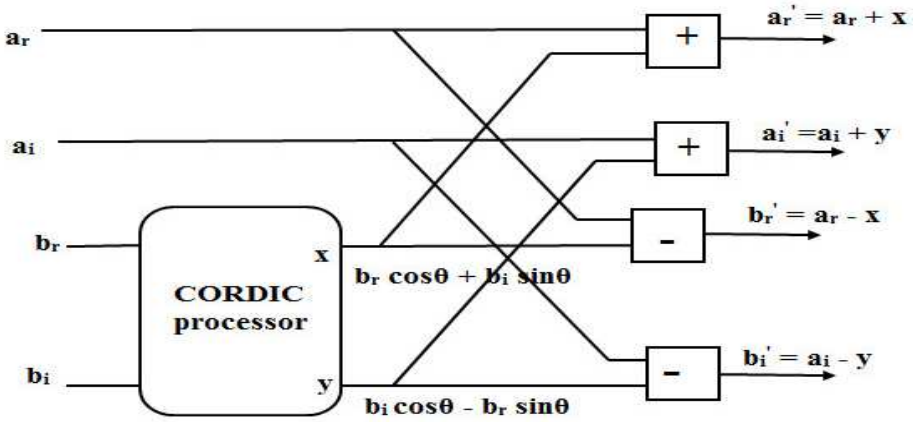

Fig.5 2-Point DIT-FFT Implementation using CORDIC Algorithm [5]

RESULTS AND DISCUSSION

Simulation result for the sine and rectified cosine is shown in the Fig.6. The Sine and Rectified Cosine is implemented in rotation mode of the CORDIC algorithm. The simulation result is for the angle between -90^0 to $+90^0$. The simulation result for 2-Point FFT using CORDIC is shown in Fig.7. The result is obtained after 16 clock cycles as the CORDIC algorithm executes in 16 clock cycles. The simulation result for the 4-Point FFT is shown in the Fig.8. The final result is obtained after 16 clock cycles as CORDIC needs 16 clock cycles to execute.

The comparison of different parameters such as delay, multiplier, adder/subtractor with and without CORDIC for 2-Point FFT is shown in the Table-1. The delay for 2-Point FFT using CORDIC is decreased compared to delay for 2-Point FFT without using CORDIC. The comparison of different parameters such as delay, multiplier, adder/subtractor with and without CORDIC for 4-Point FFT is shown in the Table-2. The delay for 4-Point FFT using CORDIC is decreased compared to delay for 4-Point FFT without using CORDIC.

Table -1 Comparison of 2-Point DIT FFT with and Without CORDIC

S.No	Parameters	2-Point FFT Without CORDIC	2-Point FFT With CORDIC
1	Delay	14.374ns	10.459 ns
2	Multipliers	4	0
3	Adders/Subtractors	6(3 Adders,3 Subtractors)	37(2 Adders,2 Subtractors, 33 Add/Sub)

Table -2 Comparison of Parameters of 4-Point DIT FFT with and without CORDIC

S.No	Parameters	4-Point FFT Without CORDIC	4-Point FFT With CORDIC
1	Delay	17.739 ns	10.459 ns
2	Multipliers	8	0
3	Adders/Subtractors	12(6 Adders,6 Subtractors)	98(4 Adders,4 Subtractors, 90 Add/Sub)

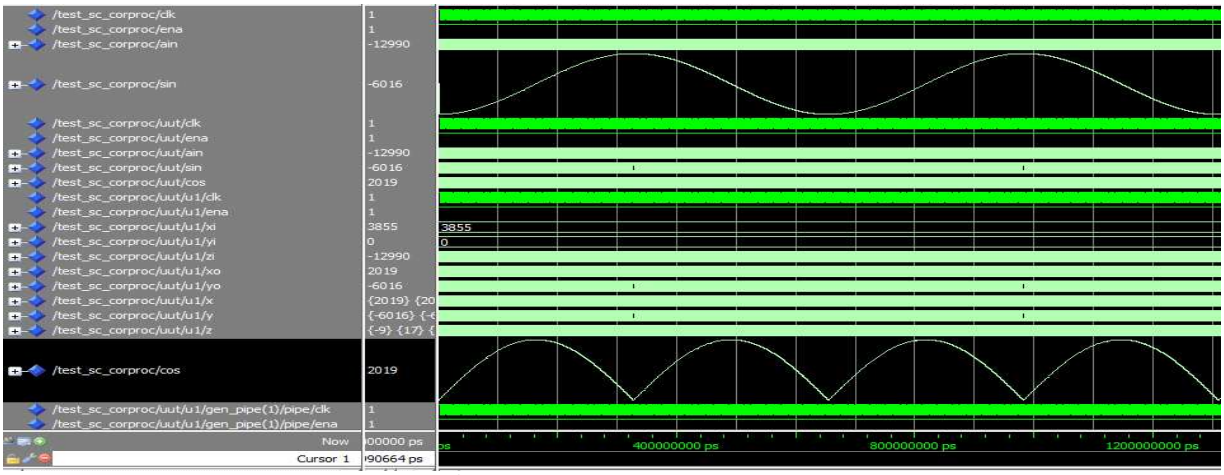

Fig.6 Sine and Rectified Cosine Implementation using CORDIC Algorithm

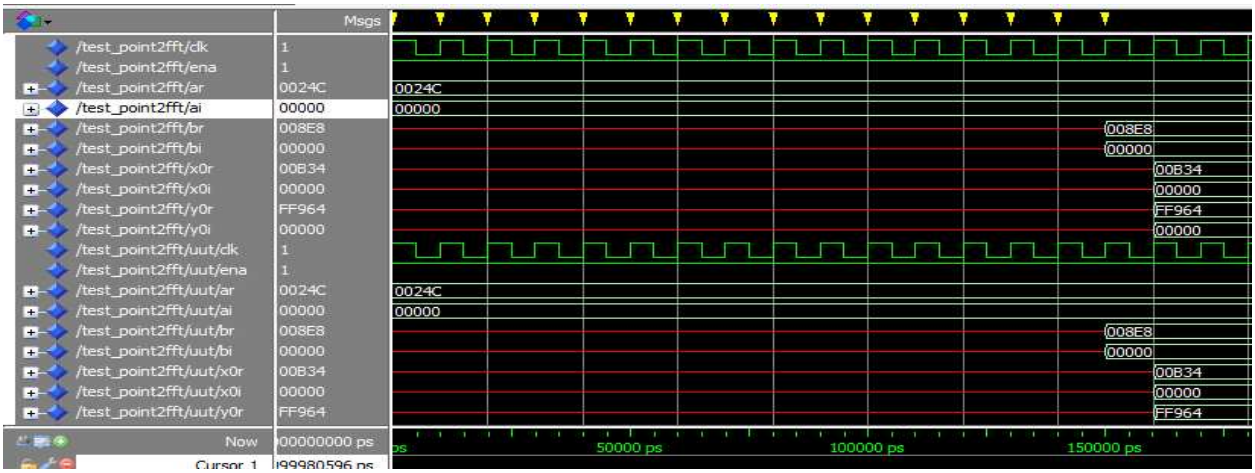

Fig.7 2-Point FFT using CORDIC

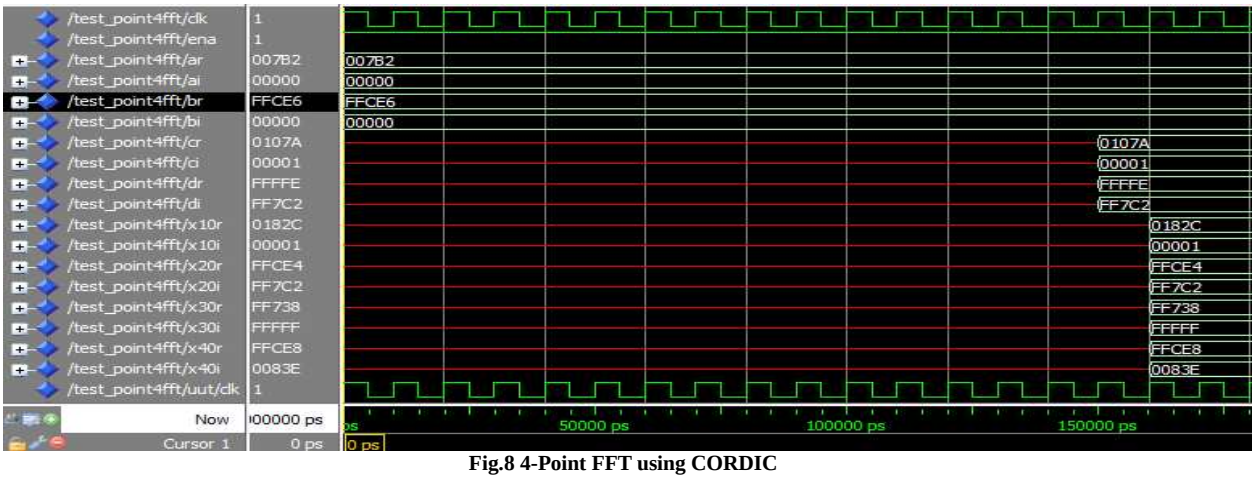

Fig.8 4-Point FFT using CORDIC

This section illustrated different results obtained by writing the VHDL code, synthesized and simulated in XILINX and MODELSIM respectively. The first simulation result shows the Sine and Cosine implementation using CORDIC Algorithm in rotation mode. The next simulation result is obtained for Rectangular to polar conversion using CORDIC algorithm in Vectoring mode. In this result the polar coordinates are obtained from rectangular coordinates. The next simulation result is obtained for Polar to Rectangular conversion using CORDIC algorithm in Rotation mode. In this result the Rectangular coordinates are obtained from Polar coordinates. The final tables are the synthesized results obtained from Xilinx. The table shows the comparison results between the 2-point DIT-FFT and 4-point DIT-FFT with and without CORDIC for different parameters like delay and the components used.

CONCLUSION

This proposed journal reduces the area and time-complexities hardwired pre-shifting scheme in barrel-shifters of the proposed circuits. The main application DIT-FFT is implemented using CORDIC adopting the pipelined architecture which reduces the latency. Using CORDIC as a replacement of multiplier in 2-point DIT FFT reduces delay by 37.4% as compared to 2-point DIT-FFT using multipliers. In 4-point DIT-FFT the delay is reduced by 69.6% when used CORDIC in place of multipliers.

REFERENCES

[1] Pramod K Meher, Javier Valls, Tso Bing Juang, K Sridharan and Koushik Maharatna, 50 Years of CORDIC: Algorithms, Architectures, and Applications, *IEEE Transactions on Circuits and Systems*, **2009**, 56(9), 1893.
[2] Esteban O Garcia, Rene Cumplido and Miguel Arias, Pipelined CORDIC Design on FPGA for a Digital Sine and Cosine Waves Generator, *3rd International Conference on Electrical and Electronics Engineering*, Veracruz, **2006**.
[3] B Lakshmi and AS Dhar, Parallel CORDIC-Like Architecture: for Fast Rotation Implementation, *International Conference on Electrical and Electronics Engineering*, Bali, **2011**.
[4] B Lakshmi and AS Dhar, High Speed Architectural Implementation of CORDIC Algorithm, *TENCON 2008, (International Technical Conference of IEEE)*, IEEE Region 10 Conference, Hyderabad, **2008.**
[5] Roberto Sarmiento, Felix Tobajas, Valentin de Armas, Roberto Esper-Chain, Jose F Lopez, Juan A Montiel Nelson and Antonio Nunez, A CORDIC Processor for FFT Computation and its Implementation using Gallium Arsenide Technology, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **1998**, 6 (1), 18.