ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 8, August2014

Implementation of Cordic Algorithm for FPGA Based Computers Using Verilog

M.Chakrapani[1], J.S.S.Ramaraju, Ch.N.L.Sujatha[3]

[1]M.Tech ECE Pursuing, Bhimavaram Institute of Engineering & Technology, West Godavari, A.P, India

[2]Assistant Professor, Department of ECE, Bhimavaram Institute of Engineering & Technology, West Godavari, A.P, India

[3]Assistant Professor, Department of ECE, Bhimavaram Institute of Engineering & Technology(BVRM), West Godavari, A.P, India

ABSTRACT: The CORDIC algorithmic is an iterative computing algorithm capable of evaluate various elementary functions using a unified shift-and-add approach Used to calculate a wide variety of functions. It consists of no. of adder-sub tractors, shift registers depend upon complexity of operation. This paper presents mode of operation of CORDIC algorithm and control CORDIC algorithm. The result has been shown in this paper that resolution of CORDIC algorithm is best for implementing many trigonometrically functions. Further, in this paper, control CORDIC algorithm take advantage of Angle and Quadrant correction.

KEYWORDS*:* CORDIC algorithm, rotation mode and vector mode, control CORDIC.

I. INTRODUCTION

CORDIC is an acronym for Co-ordinate Rotation Digital Computer. It is a class of shift adds algorithms for rotating vectors in a plane, which is usually used for the calculation of trigonometric functions, multiplication, division systems of DSP applications, such as Fourier transform [1]. The Jack E. Volder's CORDIC algorithm is derived from the general equations for vector rotation in 1959. The CORDIC algorithm has become a widely used approach to elementary function evaluation when the silicon area is a primary constraint. The implementation of CORDIC algorithm requires less complex hardware than the conventional method [2]. The CORDIC algorithm has found its way in various applications such as pocket calculators, numerical co-processors, to high performance radar signal processing, supersonic bomber aircraft with a digital counterpart, computation of the (Fast Fourier Transform) FFT, and at the effects on the numerical accuracy [3]. CORDIC algorithm revolves around the idea of "rotating" the phase of a complex number, by multiplying it by a succession of constant values. However, the "multiplies" can all be powers of 2, so in binary arithmetic they can be done using just shifts and adds; no actual "multiplier" is needed thus it simpler and do not require complex hardware structure as in the case of multiplier. Earlier methods used are Table look up method, Polynomial approximation method etc. for evaluation of trigonometric functions. It is hardware efficient algorithm [4]. The major problem of CORDIC algorithm is long latency. To reduce this problem [2], angle correction method and Quadrant correction of CORDIC have been adopted presented in this paper.

II. CORDIC ALGORITHM

CORDIC algorithm is derived from the general equations of vector rotation. Vector rotation can also be used for polar to rectangular and rectangular to polar conversion, for vector magnitude and as a building block in certain transforms such as the DFT and DCT [5]. The CORDIC algorithm provides an iterative method of performing vector rotation by arbitrary angle using shift and adds. The algorithm credited to Volder is derived from the general rotation transform. If a vector V with co-ordinates (x,y) is rotated through an angle ǿ then a new vector V’ can be obtained with co-ordinates (x,y) where x’ and y’ can be obtained using x , y and ǿ by the following method as shown in Fig.1

$$X = r\cos\theta \quad ,Y = r\sin\theta \qquad (1)$$

$$V' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\cos\phi - y\sin\phi \\ y\cos\phi + x\sin\phi \end{bmatrix} \qquad (2)$$

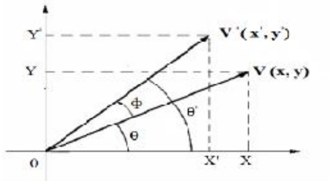


Fig 1 Rotation of a Vector V by the angle ϕ [3]

V came into picture after anticlockwise rotation by an angle Ǿ . From Fig.1, it can be observed

$$\theta' - \theta = \phi \qquad (3)$$

$$OX' = x' = r\cos\theta' \qquad (4)$$

$$= r\cos(\phi + \theta) \qquad (5)$$

$$= (r\cos\theta).\cos\phi - (r\sin\theta).\sin\phi \qquad (6)$$

Using Fig 2

$$OX' = x' = x\cos\phi - y\sin\phi$$

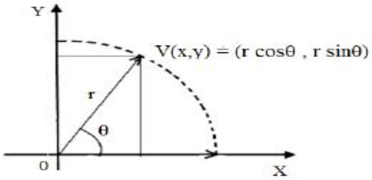


Fig 2 Vector V with magnitude r and phase θ

$$OY' = y' = r\sin\theta' \qquad (7)$$

$$= x\sin\phi + y\cos\phi \qquad (8)$$

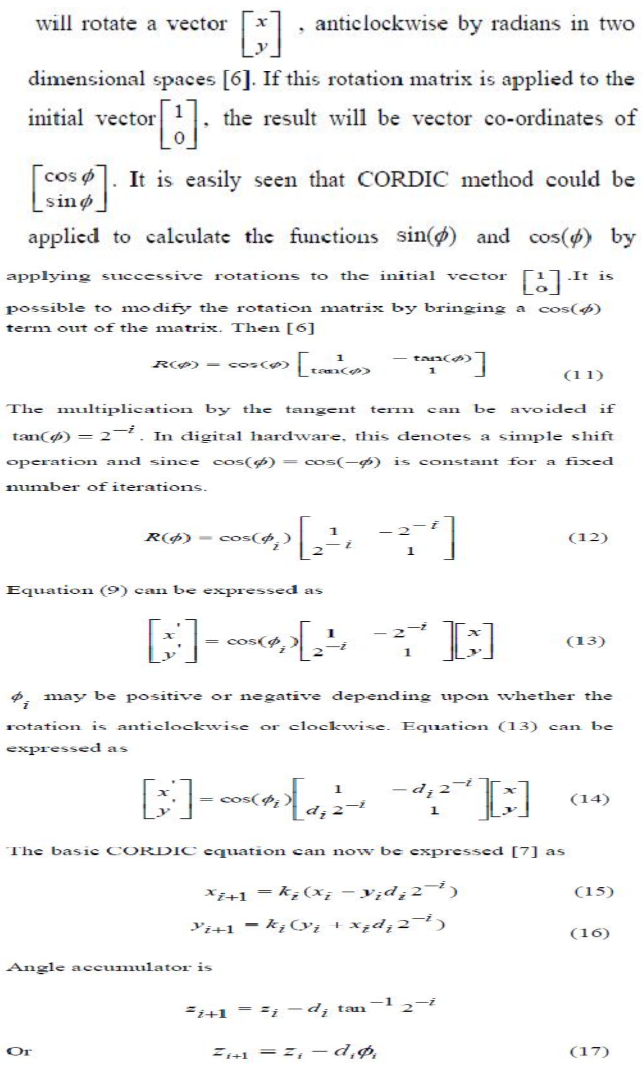
10.15662/ijareeie.2014.0308082

V´ after anticlockwise rotation of vector V by angle Ǿ is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (9)$$

It is well known that the rotation matrix

$$R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \qquad (10)$$



will rotate a vector $\begin{bmatrix} x \\ y \end{bmatrix}$, anticlockwise by radians in two dimensional spaces [6]. If this rotation matrix is applied to the initial vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, the result will be vector co-ordinates of $\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}$. It is easily seen that CORDIC method could be applied to calculate the functions $\sin(\phi)$ and $\cos(\phi)$ by applying successive rotations to the initial vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ .It is possible to modify the rotation matrix by bringing a $\cos(\phi)$ term out of the matrix. Then [6]

$$R(\phi) = \cos(\phi) \begin{bmatrix} 1 & -\tan(\phi) \\ \tan(\phi) & 1 \end{bmatrix} \qquad (11)$$

The multiplication by the tangent term can be avoided if $\tan(\phi) = 2^{-i}$. In digital hardware, this denotes a simple shift operation and since $\cos(\phi) = \cos(-\phi)$ is constant for a fixed number of iterations.

$$R(\phi) = \cos(\phi_i) \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \qquad (12)$$

Equation (9) can be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos(\phi_i) \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (13)$$

ϕ_i may be positive or negative depending upon whether the rotation is anticlockwise or clockwise. Equation (13) can be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos(\phi_i) \begin{bmatrix} 1 & -d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (14)$$

The basic CORDIC equation can now be expressed [7] as

$$x_{i+1} = k_i (x_i - y_i d_i 2^{-i}) \qquad (15)$$

$$y_{i+1} = k_i (y_i + x_i d_i 2^{-i}) \qquad (16)$$

Angle accumulator is

$$z_{i+1} = z_i - d_i \tan^{-1} 2^{-i}$$

Or $$z_{i+1} = z_i - d_i \phi_i \qquad (17)$$

10.15662/ijareeie.2014.0308082

Where i denote the number of rotation required reaching the required angle of the required vector, $k_i = \cos(\arctan(2^{-i}))$ and $d_i = \pm 1$ (direction of rotation), the product of the k_i represents K-factor.

$$k = \prod_{i=0}^{n-1} k_i \tag{18}$$

Where $\prod_{i=0}^{n-1} k_i = \cos\phi_0 . \cos\phi_1 . \cos\phi_2 . \cos\phi_3 .....\cos\phi_{n-1}$ (ϕ is the angle of rotation here for n times rotation). These ϕ_i are stored in the ROM of the CORDIC hardware as the look up table. k_i is the CORDIC gain. For 8-bit hardware CORDIC approximation method, the value of k_i as

$$\begin{aligned} k_i &= \prod_{i=0}^{7} \cos\ \phi_i = \cos\ \phi_0 . \cos\ \phi_1 . \cos\ \phi_2 .... \cos\ \phi_7 \\ &= \cos 45^o . \cos 26.565^o . \cos 14.036^o ...... \cos 0.4469^o \\ &= 0.6073 \end{aligned} \tag{19}$$

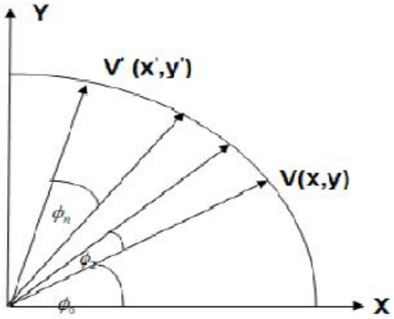


Fig 3 chaining multiple micro-rotation together

As shown in fig 3, multiple micro rotations may be chained together as the vector moves in discrete angular steps $(\phi_0, \phi_1, \phi_2, ---, \phi_n)$ from its initial position of $V(x, y)$ towards its final target position of $V^{'}(x^{'}, y^{'})$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos\phi_0 . \cos\phi_1 \ldots . \cos\phi_n \begin{bmatrix} 1 & -\tan\phi_0 \\ \tan\phi_0 & 1 \end{bmatrix} \ldots \begin{bmatrix} 1 & -\tan\phi_n \\ \tan\phi_n & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

For 8 bit CORDIC hardware, cosine and sine of angle ϕ can be represented in matrix form by using equation (19)

$$\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} = \begin{bmatrix} 1 & -\tan\phi_0 \\ \tan\phi_0 & 1 \end{bmatrix} \ldots \begin{bmatrix} 1 & -\tan\phi_7 \\ \tan\phi_7 & 1 \end{bmatrix} \begin{bmatrix} 0.6073 \\ 0 \end{bmatrix}$$

TABLE 1

FOR 8, 16, 24, 32 BIT CORDIC HARDWARE [8]

i	$2^{-i} = \tan\phi_i$	$\phi_i = \tan^{-1}(2^{-i})$	ϕ_i *in radians*
0	1	45°	0.7854
1	0.5	26.565°	0.4636
2	0.25	14.036°	0.2450
3	0.125	7.125°	0.1244
4	0.0625	3.576°	0.0624
5	0.03125	1.7876°	0.0312
6	0.015625	0.8938°	0.0156
7	0.0078125	0.4469°	0.0078
8	0.00390625	0.2238°	0.0039
9	0.001953125	0.1119°	0.0019
10	0.0009765625	0.05595°	0.00098
11	0.00048828125	0.02798°	0.00049
12	0.000244140625	0.01399°	0.00024
13	0.0001220703125	0.00699°	0.00012
14	0.00006103515625	0.00349°	0.000061
15	0.00003051757813	0.00175°	0.000031
16	0.00001525878906	0.000874°	0.0000152
17	0.000007629394531	0.000437°	0.0000076
18	0.000003814697266	0.0002186°	0.0000038
19	0.000001907348633	0.0001093°	0.0000019
20	0.0000009536743164	0.00005464°	0.00000095
21	0.0000004768371582	0.0000273°	0.00000048
22	0.0000002384185791	0.00001366°	0.00000024
23	0.0000001192092896	0.00000683°	0.00000012
24	0.00000005960464478	0.000003415°	0.0000000595
25	0.00000002980232239	0.000001708°	0.0000000298
26	0.00000001490116119	0.000000854°	0.0000000149
27	0.000000007450580597	0.000000427°	0.00000000745
28	0.000000003725290298	0.000000213°	0.00000000371
29	0.000000001862645149	0.000000107°	0.00000000187
30	0.0000000009313225746	0.0000000534°	0.00000000093
31	0.0000000004656612873	0.0000000267°	0.00000000047

For 16 bit CORDIC hardware, the value of k_i is

$$k_i = \prod_{i=0}^{15} \cos\phi_i = \prod_{i=0}^{15} \cos\phi_0 . \cos\phi_1 \ldots \cos\phi_{15} = 0.6073$$

For 24 bit CORDIC hardware, the value of k_i is

$$k_i = \prod_{i=0}^{23} \cos\phi_i = \prod_{i=0}^{23} \cos\phi_0 . \cos\phi_1 \ldots \cos\phi_{23} = 0.6073$$

For 32 bit CORDIC hardware, the value of k_i is

$$k_i = \prod_{i=0}^{31} \cos\phi_i = \prod_{i=0}^{31} \cos\phi_0 . \cos\phi_1 \ldots \cos\phi_{31} - 0.6073$$

CORDIC gain k_i is same for all the CORDIC hardware.

III. SCALING OPTIMIZATION AND IMPLEMENTATION

We discuss here the optimization of scaling to match with the optimized set of elementary angles for the micro-rotations.

A. Scaling Approximation for Fixed Rotations

The generalized expression for the scale-factor given by (2) can be expressed explicitly for the selected set of m1 micro rotations as

$$K = \prod_{i=0}^{m_1-1} \left[(1 - 2^{-2k(i)})\right]^{-1/2}$$

where **k**(i) for 0≤i<m is the number of shifts in the **i**th micro-rotation. Except for k(i)=0 (i.e., rotation by 45), by binomial expansion, any term can be written as

$$1 - \frac{x}{2} + \frac{3x^2}{8} - \frac{5x^3}{16} + \frac{35x^4}{128} - \frac{63x^5}{256} + \frac{231x^6}{1024} - \cdots$$

where $x = 2^{-2i}$.

IV.IMPLEMENTATION OF SCALING

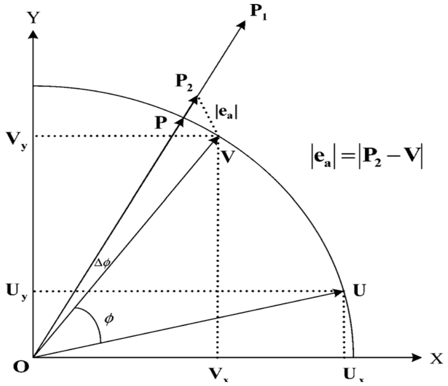
Scaling and micro-rotations could be implemented either in the same circuit in interleaved manner or in two separate stages. The implementation of scaling as well as the micro-rotation would however depend on the level of desired accuracy, and the implementation of scaling also depends on the implementation of micro-rotations. Therefore, we discuss here the realization of the scaling circuits corresponding to different implementations of micro-rotations.

Generalized Implementation of Scaling

The shift-add circuit for scaling according to (7) is shown. The scaling circuit of Shift-add scaling circuit using hardwired pre-shifted loading can use hardwired pre-shifting for minimizing barrel-shifter complexity and could be placed after the CORDIC cell of Fig. 2 to perform micro-rotation and scaling in two separate stages. The generalized CORDIC circuit for fixed rotation to perform the micro-rotation and the scaling in interleaved manner in alternate cycles is shown. The circuit of Fig. 8 is similar to that of Fig. It involves only an additional line-changer circuit to change the path of unshifted (direct) input. The structure and function of line-changer is shown. The line-changer is placed on the unshifted input data line to keep the critical path the same as that of CORDIC cell for constant cell multiplications.

ANALYSIS OF ERROR

There are two types of error encountered during the rotation mode CORDIC iterations. Those are: approximation error and round-off error. Approximation error arises due to approximation of angle of rotation and scaling factor, while the round-off error arises due to the finite word-length of the output components. We derive the expression for these two errors in the following subsections



Approximation Error

Proposed CORDIC operation and approximation error illustrates the CORDIC iteration which consists of a pseudo-micro-rotations and a scaling. In the figure, is an input vector to be rotated through angle ø. It is assumed that scaling and micro-rotations are implemented in two separate stages.P1 be the rotated vector after m1 micro-rotations given by

$$\mathbf{P}_1 = \prod_{i=0}^{m_1-1} \mathbf{R}(i)\mathbf{U}.$$

The rotation matrix **R(i)** is given by (3). The **i** th scaling factor is given by

$$S(i) \triangleq 1 + \delta_i 2^{-s(i)}$$

such that after m2 iterations of scaling we get

$$\mathbf{P}_2 = \prod_{i=0}^{m_2-1} S(i)\mathbf{P}_1.$$

After the micro-rotations, there is a discrepancy Δø between the desired angle and the resultant angle due to the limited number of micro-rotations. Moreover, P1 cannot reach P on the circle after the scaling since KA is an approximated value which is not same as the required K. Similar to the method used, the approximation error is evaluated as a distance between the desired output V and the actual CORDIC output P2 as follows:

$$\mathbf{e}_a = |\mathbf{V} - \mathbf{P}_2|.$$

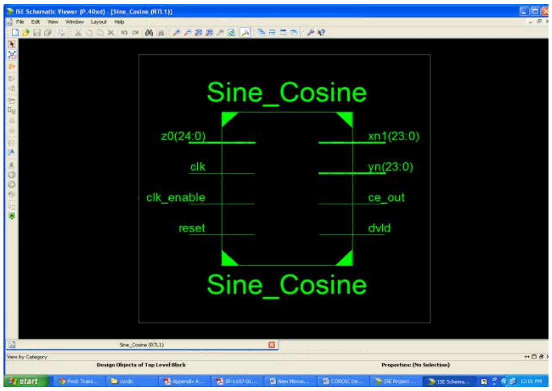
For the known and fixed angle, an expectation of the approximation error can be estimated once we know the input statistics as

$$E(|\mathbf{e}_a|^2) = \left(\left(1 - \frac{K_A}{K}\right)^2 + \Delta\phi^2\right)E(|\mathrm{U}|^2).$$

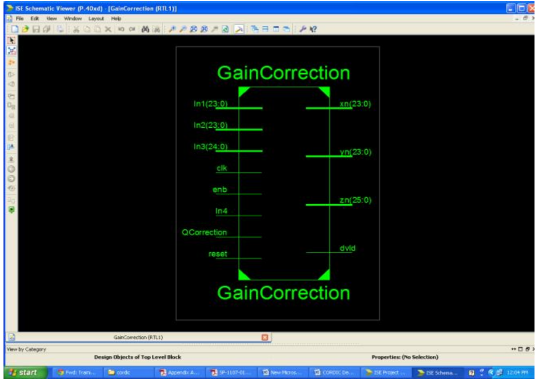
COMPLEXITY CONSIDERATIONS

We discuss here the hardware and time complexities of the proposed design. In the existing literature we do not find similar work on CORDIC implementation of known and fixed rotations. Therefore, we compare the proposed design with the conventional CORDIC design for the rotation of unknown angle. We have used the basic CORDIC processor in [3, Fig. 2] for the implementation of conventional CORDIC. In addition, we have designed a reference architecture (see Fig. 1) for straight forward implementation of fixed rotations, and we have compared the complexities and speed performance of the proposed design with the conventional and reference design.

V. RESULTS AND ANALYSIS

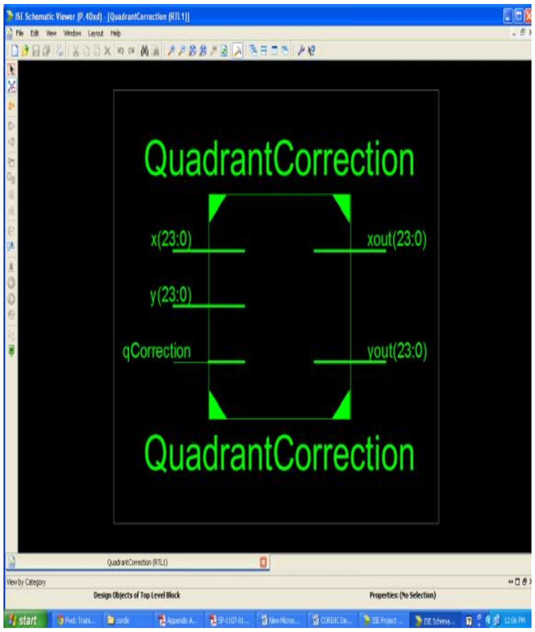


RTL Schematic of Cordic algorithm

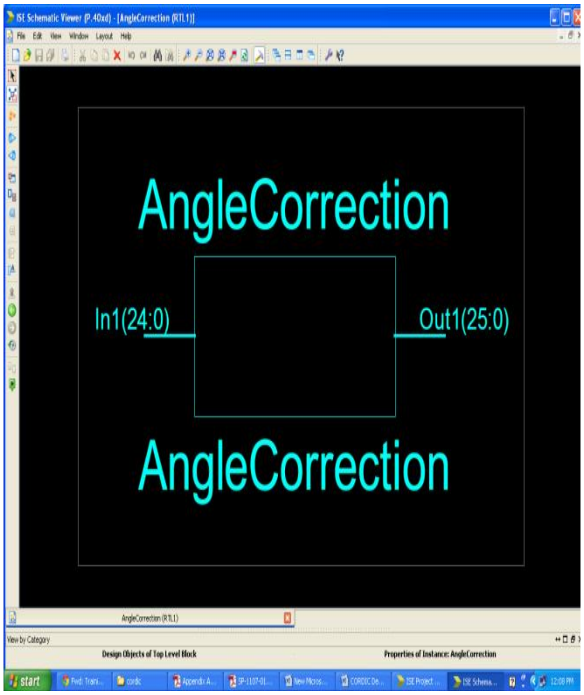


RTL Schematic of Gain Correction block

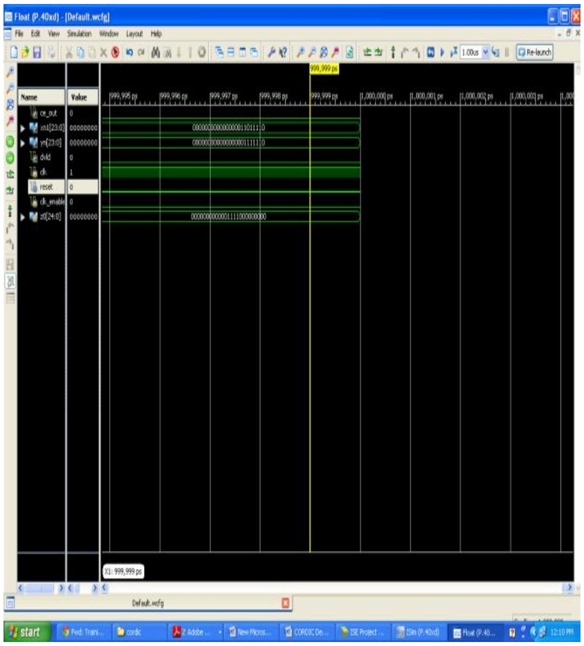
10.15662/ijareeie.2014.0308082


ISE Schematic Viewer (P.40xd) - [QuadrantCorrection (RTL1)]
File Edit View Window Layout Help
QuadrantCorrection
x(23:0)
y(23:0)
qCorrection
xout(23:0)
yout(23:0)
QuadrantCorrection
QuadrantCorrection (RTL1)
View by Category
Design Objects of Top Level Block
Properties: (No Selection)
start
12:06 PM

RTL Schematic of Quadrant Correction


ISE Schematic Viewer (P.40xd) - [AngleCorrection (RTL1)]
File Edit View Window Layout Help
AngleCorrection
In1(24:0)
Out1(25:0)
AngleCorrection
AngleCorrection (RTL1)
View by Category
Design Objects of Top Level Block
Properties of Instance: AngleCorrection
start
12:08 PM

RTL Schematic of Angle Correction



Name	Value
ce_out	0
xn1[23:0]	00000000
yn[23:0]	00000000
dvld	0
clk	1
reset	0
clk_enable	0
z0[24:0]	00000000

xn1[23:0]: 000000000000000011011110
yn[23:0]: 000000000000000001111110
z0[24:0]: 0000000000000111100000000

Simulation Results of Cordic algorithm

VI. CONCLUSION

In this paper Control CORDIC Angle correction and Quadrant correction is implemented. latency can be further reduced by Another CORDIC algorithm techniques. Overshoot problem in original CORDIC has been overcome by angle selection scheme. Due to the increasing of no. of bits complexity of hardware also increases .

REFERENCES

[1] Volder J. E,"The CORDIC trigonometric computing technique", IRE Trans. Electronic Computing, Volume EC-8, pp 330 - 334, 1959.

[2] Terence K. Rodrigues and Eari E.Swartzlander ,Jr.Fellow,"Adaptive CORDIC :Using parallel Angle Recoding to Accelerate Rotations", IEEE Transactions on Computers, Vol .59,NO.4, April 2010

[3] Pramod K.Meher, Senior Member ,IEEE, Javier Valls, Member, IEEE, Tso- Bing Juang , Member, IEEE, K.Sridhan, Senior Member, IEEE, and Koushik Maharatna, Member , IEEE " 50 Years of CORDIC: Algorithms, Architectures, and Applications" , IEEE transactions on circuits and systems,VOL.56.NO.9,September, 2009.

[4] Alan Sultan, " CORDIC: How Hand calculators Calculate" Integre Technical Publishing Co.,Inc. college Mathematics journal VOL.40 NO.2 , March 2009.

[5] Yu Hen Hu "CORDIC-Based VLSI Architectures for Digital signal processing", IEEE Signal processing-1053-5888/92/$3.00, 1992.

[6] Dr. Dobbs , Michael Pascale, "Using CORDIC methods for Computations in micro-controllers" ,1 September 2000.

[7] Y.H .Hu, "CORDIC based VLSI architecture for digital signal processing",IEEE signal processing Mag., pp 16-35, july 1992.

[8] Kia Bazargan , "CORDIC Algorithms" EE-5324-VLSI Design 2, University of Minnesto,Spring 2006.

[9] Tanya Vladimirova and Hans Tiggler,"FPGA implementation of sine and cosine generators Using the CORDIC Algorithm" surrey space centre ,University of Surrey , Guildford , Surrey , GU2 5XH.

[10] Eric Keller,"Dynamic circuit specialization of CORDIC processor", Nov 7, 2000.

[11] Quiang Gao, "Low complexity Arithmetic Implementation for DSP Radio Receiver" university of strathcyclde.(qiang.gao@eee.strath.ac.uk).

[12] Jean Duprat and Jean – Michel Muller " The CORDIC Algorithm :New Results for Fast VLSI implementation" IEEE Transactions on computers ,Vol 42.No.2,Feb 1993.