G.Sandhya, Syed Inthiyaz, Dr. Fazal Noor Basha / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 3, May-Jun 2012, pp.1588-1595

Power and Area Optimization for Pipelined CORDIC Processor Architecture in VLSI

G.Sandhya, Syed Inthiyaz*, Dr. Fazal Noor Basha*****
** (M.Tech VLSI student, Department of ECE, KL University, and Vijayawada)
*(Asst.Professor, Department of ECE, KL University, and Vijayawada)
*** (Asst.Professor, Department of ECE, KL University, and Vijayawada)

ABSTRACT
CORDIC (Coordinate Rotation Digital Computer) is a class of shift add algorithms for rotating vectors in a plane, which is usually used for the calculation of trigonometric functions, multiplication, division and conversion between binary and mixed radix number systems of DSP applications, such as Fourier Transform. The CORDIC algorithm has become a widely used approach to elementary function evaluation when the silicon area is a primary constraint. CORDIC is a simple and hardware-efficient algorithm for the implementation of various elementary functions. Instead of using Calculus based methods such as polynomial or rational functional approximation, it uses simple shift, add, subtract and table look-up operations to achieve the rotation of vectors in a plane. It is usually implemented in either Rotation mode or Vectoring mode. In either mode, the algorithm is rotation of an angle vector by a definite angle but in variable directions. This fixed rotation in variable direction is implemented through an iterative sequence of addition/subtraction followed by bit-shift operation. The final result is obtained by appropriately scaling the result obtained after successive iterations. The CORDIC algorithm can be optimized in area and power.

Keywords – **pipelined cordic processor, cordic algorithm, post processor, optimization, cordic core, conversion modules**

1. INTRODUCTION

CORDIC (Coordinate Rotation Digital Computer) is a method for computing elementary functions using minimal hardware such as shifts adds/subs and compares. CORDIC works by rotating the coordinate system through constant angles until the angle is reduces to zero. The angle offsets are selected such that the operations on X and Y are only shifts and adds. Instead of using Calculus based methods such as polynomial or rational functional approximation, it uses simple shift, add, subtract and table look-up operations to achieve this objective. It is usually implemented in either Rotation mode or Vectoring Mode.

In either mode, the algorithm is rotation of an angle vector by a definite angle but in variable directions. This fixed Rotation in variable direction is implemented through an iterative sequence of addition/subtraction followed by bit-shift operation [1][2].

The final result is obtained by appropriately scaling the result obtained after successive iterations. Owing to its simplicity the CORDIC algorithm can be easily implemented on a VLSI system. Hardware requirement and cost of CORDIC processor is less as only shift registers, adders and look-up table (ROM) are required Number of gates required in hardware implementation, such as on an FPGA, is minimum as hardware complexity is greatly reduced compared to other processors such as DSP multipliers. It is relatively simple in design. No multiplication and only addition, subtraction and bit-shifting operation ensures simple VLSI implementation [4].

Delay involved during processing is comparable to that during the implementation of a division or square-rooting operation. Either if there is an absence of a hardware multiplier (e.g. uC, uP) or there is a necessity to optimize the number of logic gates (e.g. FPGA) CORDIC is the preferred choice. The number of logic gates for the implementation of a CORDIC is roughly comparable to the number required for a multiplier as both require combinations of shifts and addition [5].

The choice for a multiplier-based or CORDIC-based implementation will depend on the context. The multiplication of two complex numbers represented by their real and imaginary components (rectangular coordinates), for example, requires 4 multiplications, but could be realized by a single CORDIC operating on complex numbers represented by their polar coordinates, especially if the magnitude of the numbers is not relevant (multiplying a complex vector with a vector on the unit circle actually amounts to a rotation). CORDICs are often used in circuits for telecommunications such as digital down converters.

1.1 CORDIC Block Diagram

All CORDIC Processor cores are built around three fundamental blocks: The pre-processor, the post-processor and the actual CORDIC core. The CORDIC core is built using a pipeline of Cordic Pipe blocks. Each Cordic Pipe block represents a single step in the iteration processes. The CORDIC core is the heart of the CORDIC Processor Core. It performs the actual CORDIC algorithm. All iterations are performed in parallel, using a pipelined structure.

Because of the pipelined structure the core can perform a CORDIC transformation each clock cycle by ensuring the highest throughput possible. Because of the

arctan table used in the CORDIC algorithm, it only converges in the range of –1(rad) to +1(rad). To use the CORDIC algorithm over the entire 2π range the inputs need to be manipulated to fit in the –1 to +1 rad. range. This is handled by the preprocessor. The post-processor corrects this and places the CORDIC core's results in the correct quadrant. It also contains logic to correct the P-factor.

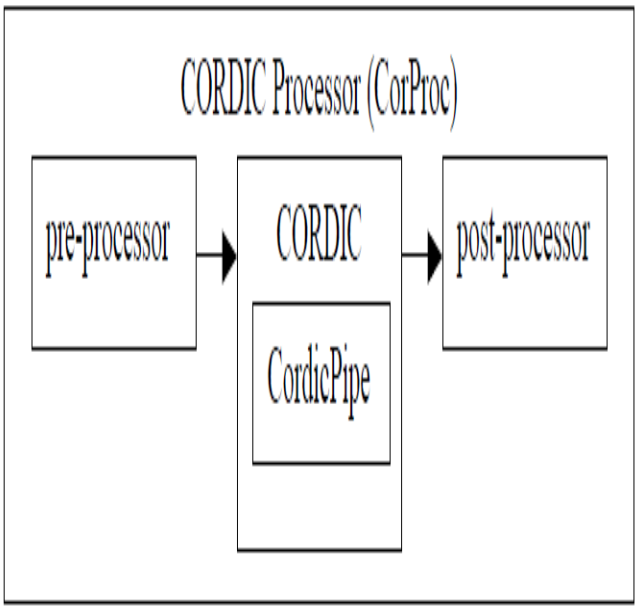
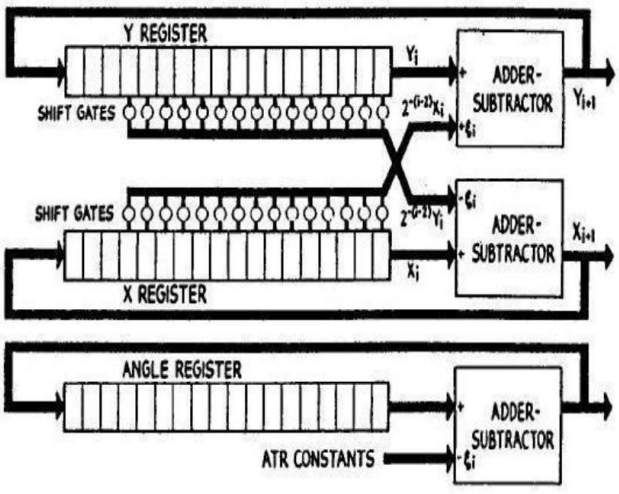


Fig: 1 CORDIC Block Diagram

2. CORDIC BASIC ARCHITECTURE

The following diagram explains the basic hardware architecture of a CORDIC processor. It shows the adders/subtractors and the shift registers. The adders/subtractors perform the addition/subtraction of binary numbers. The shift register performs the bit-shift operation in accordance with the algorithm. The constants corresponding to fixed angle values are obtained from the Look-up table implemented as a ROM.



The current research in the design of high speed VLSI architectures for real-time digital signal processing (DSP) algorithms has been directed by the advances in the VLSI technology, which have provided the designers with significant impetus for porting algorithm into architecture. Many of the algorithms used in DSP and matrix arithmetic require elementary functions such as trigonometric, inverse trigonometric, logarithm, exponential, multiplication, and division functions.

The commonly used software solutions for the digital implementation of these functions are table lookup method and polynomial expansions, requiring number of multiplication and additions/subtractions. However, digit-by-digit methods exist for the evaluation of these elementary functions, which compute faster than software solutions. CORDIC algorithm has drawn wide attention from academia and industry for various applications such as DSP, biomedical signal processing, and 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 [6].

In this paper, we first classify the CORDIC algorithm based on the number system and discuss its importance in the implementation of CORDIC algorithm. Then, we present systematic and comprehensive taxonomy of rotational CORDIC algorithms, which are subsequently discussed in depth. Special attention has been devoted to the higher radix and flat techniques proposed in the literature for reducing the latency. Finally, detailed comparison of various algorithms is presented, which can provide first-order information to designers looking for either further improvement of performance or selection of rotational CORDIC for a specific application.

3. CORDIC ALGORITHM

The CORDIC algorithm performs a planar rotation. Graphically, planar rotation means transforming a vector (X_i, Y_i) into a new vector (X_j, Y_j). This is shown in fig below. This algorithm calculates the sine and cosine values of a given angle (in radians) by transforming the co-ordinates from polar representation to representation in Cartesian form. To measure the values of sine and cosine, the coordinates corresponding to the angle on a unit circle is found, the x-coordinate of which gives the cosine values while the y-coordinate gives the sine value[8]. We start with the vector v0 aligned with the x-axis:

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$ Equation 1

In Fig 2, the x-coordinate is 1, whereas the y-coordinate is 0.

In the first iteration, this vector is rotated 45° counterclockwise to get the vector v1. Successive iterations will rotate the vector in one or the other direction by size decreasing steps, until the desired angle has been achieved. The direction of rotation of the CORDIC vector is decided by the value of β_i where β_i is the difference between the rotation value to be reached and the present angle

accumulator value. CORDIC algorithm depends on the value of this parameter β_i which in turn is decided by value of co-ordinate y. Step size is given as:

$\tan(1/(2^{(i-1)}))$; where i= 1,2,3,.... Equation 2

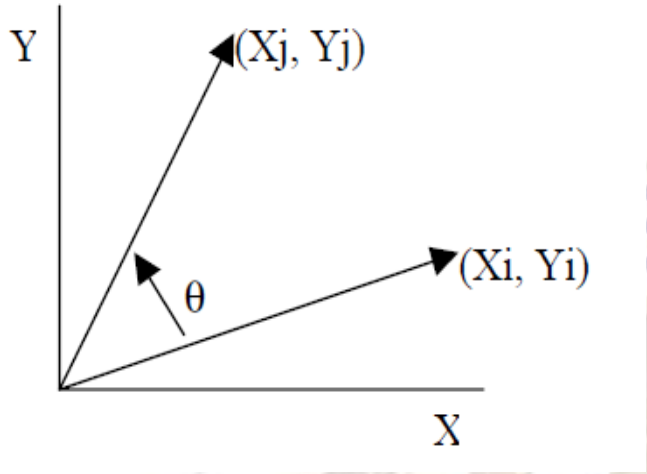


Fig: 2 Planar representation of CORDIC Algorithm

Whenever one complex number is multiplied with another the magnitude of the resultant complex number is the product of the individual magnitudes and its phase is sum of the individual phases. Thus, rotation of a vector can be achieved by simply multiplying the vector with a complex number of unit magnitude. This is known as real rotation. The subsequent values of the vector v_i is given by

$$v_i = R_i v_{i-1}$$

The rotation matrix R is given by:

$$R_i = \begin{pmatrix} \cos\gamma_i & -\sin\gamma_i \\ \sin\gamma_i & \cos\gamma_i \end{pmatrix}$$ Equation 3

Using the following two trigonometric identities

$$\cos\alpha = \frac{1}{\sqrt{1+\tan^2\alpha}}$$

$$\sin\alpha = \frac{\tan\alpha}{\sqrt{1+\tan^2\alpha}}$$ Equation 4

The rotation matrix becomes:

$$R_i = \frac{1}{\sqrt{1+\tan^2\gamma_i}} \begin{pmatrix} 1 & -\tan\gamma_i \\ \tan\gamma_i & 1 \end{pmatrix}$$ Equation 5

The expression for the rotated vector $v_i = R_i\, v_{i-1}$

$$v_i = \frac{1}{\sqrt{1+\tan^2\gamma_i}} \begin{pmatrix} x_{i-1} & -y_{i-1}\tan\gamma_i \\ x_{i-1}\tan\gamma_i & y_{i-1} \end{pmatrix}$$

Equation 6

The resultant rotations described by the vector in the figure described below are known as Pseudo-rotations. Volder proposed that this rotation angle θ can be broken down into a series of small successive shrinking angles. Restricting the angles γ_i so that $\tan\gamma_i$ takes on the value $\pm 2^{-i}$.

The multiplication with the tangent can be replaced by a division by a power of two, which is efficiently done in digital computer hardware using a bit shift. The expression then becomes:

$$v_i = K_i \begin{pmatrix} x_{i-1} & -\sigma_i 2^{-i} y_{i-1} \\ \sigma_i 2^{-i} x_{i-1} & y_{i-1} \end{pmatrix}$$ Equation 7

Where

$$K_i = \frac{1}{\sqrt{1+2^{-2i}}}$$

σi can have the values of −1 or 1 and is used to determine the direction of the rotation: if the angle β_i is positive then σi is 1 otherwise it is −1.

The value of β_i is given as,

$$\beta_i = \beta_{i-1} - \sigma_i\gamma_i$$

Equation 8

Where the initial β value is the angle for which the sine, cosine values are to be calculated for

$$\gamma_i = \arctan 2^{-i}$$ Equation 3 Equation 9

Thus, the new x and y coordinate values can be given as

$$X_i = X_{i-1} - 2^{-i}Y_{i-1}$$

$$Y_i = Y_{i-1} + 2^{-i}X_{i-1}, \quad when\ \beta_i > 0$$ Equation 10

And

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

$$Y_i = Y_{i-1} - 2^{-i}X_{i-1}, \quad when\ \beta_i < 0$$ Equation 11

3.1 Scaling Factor

We can ignore K_i in the iterative process and then apply it afterward by a scaling factor:

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} 1/\sqrt{1+2^{-2i}}$$

Equation 12

This is calculated in advance and stored in a table, or as a single constant for a fixed number of iterations. This correction could also be made in advance, by scaling v_0 and hence saving a multiplication. Additionally it can be noted that

$$K = \lim_{n\to\infty} K(n) \approx 0.607252935008881256169 4$$

Equation 13

Here the radix 2 system is used since it avoids use of multiplications while implementing the above equation. Hence a CORDIC iteration can be realized using shifters and adders only.

4. PIPELINED CORDIC ARCHITECTURE

It is comparatively the most efficient CORDIC architecture. In this method multiple iterations take place in multiple clock cycles. It is implemented by inserting registers within the different adder stages.

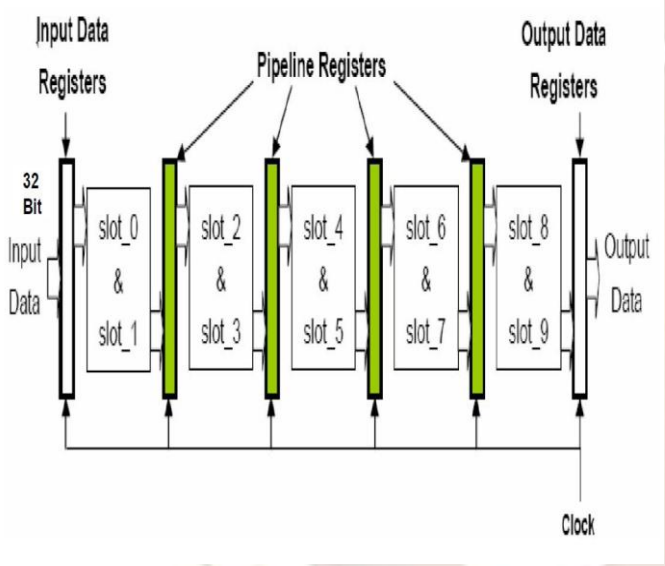


Fig: 3 Pipelined CORDIC Architecture

FPGA implementation is easy, as registers are already available, thus requiring no extra hardware. Number of iterations after which the system gives accurate result can be modeled, considering clock frequency of the system.

When operating at greater clock period power consumption in later stages reduces due to lesser switching activity in each clock period. It overcomes the disadvantages in Sequential or Iterative CORDIC structure and parallel or cascaded CORDIC. We will consider two modules for showing the optimization of pipelined CORDIC processor.

4.1 Polar TO Rectangular Conversion

A Logical Expression for sine and cosine computer is a Polar to Cartesian Co-ordinate transformer. The transformation from Polar to Cartesian is defined by:

$$X = r \cos\alpha$$

$$Y = r \sin\alpha$$

As pointed out above the multiplication by the Magnitude comes for free using the cordic rotator. The transformation is accomplished by selecting the rotation mode with xo= Polar magnitude, zo= polar phase, and yo=0. The vector result represents the polar input transformed to Cartesian space. The transform has a gain equal to the rotator gain, which needs to be accounted for some where in the system .If the gain is unacceptable, the polar magnitude may be multiplied by the reciprocal of the rotator gain before it is presented to the CORDIC rotator.

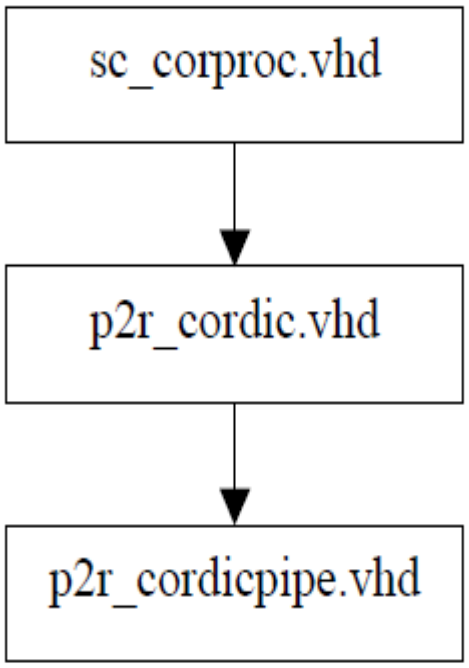


Fig: 4 Core Structure of Polar to Rectangular conversion

4.2 IO Ports

Core structure of Polar to Rectangular Conversion is described in the above figure 4.Clock input signal, Clock Enable and angle are taken as inputs. Sine and Cosine magnitudes are taken as outputs.

Port	Width	Direction	Description
CLK	1	Input	System Clock
ENA	1	Input	Clock enable signal
Ain	16	Input	Angel input
Sin	16	Output	Sine output
Cos	16	Output	Cosine output

Fig: 5 IO Ports of Polar to Rectangular conversion

5. SIMULATION RESULTS OF POLAR TO RECTANGULAR CONVERSION

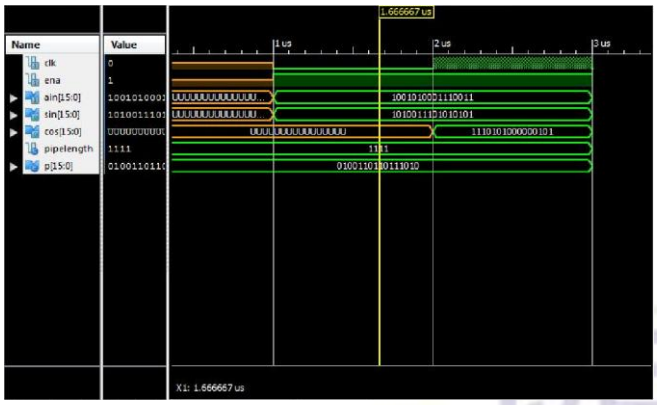


Fig: 6 Simulation result of Polar to Rectangular conversion

5.1 Synthesis Report

Maximum Frequency	189.364MHz
Minimum input arrival time before clock	5.527ns
Maximum output required time after clock	4.283ns
Maximum combinational path delay	No path found
Total REAL time to Xst completion	17.00 secs
Total CPU time to Xst completion	16.72 secs

Maximum Frequency	218.746MHz
Minimum input arrival time before clock	4.835ns
Maximum output required time after clock	4.040ns
Maximum combinational path delay	No path found
Total delay	4.040ns
Total REAL time to Xst completion	12.00 secs
Total CPU time to Xst completion	11.59 secs
Total memory usage	184912 kilobytes
Total Delay	4.283ns

Table: 1 Synthesis Report of Polar to Rectangular Conversion

5.2 Power Report

Total Supply power	0.081w
Total Current	0.002A
Junction Temperature	27.1c

Table: 2 Power Report of Polar to Rectangular Conversion

6. OPTIMIZATION IN BOTH AREA AND POWER FOR POLAR TO RECTANGULAR CONVERSION USING Xilinx PlanAhead 13.1

This can be done in Xilinx PlanAhead 13.1 by assigning required pin constraints from results obtained in Xilinx 13.1. This can be done by creating Ucf file in PlanAhead. Optimization can be analysed by observing below results. There will be decrease in both area and delay.

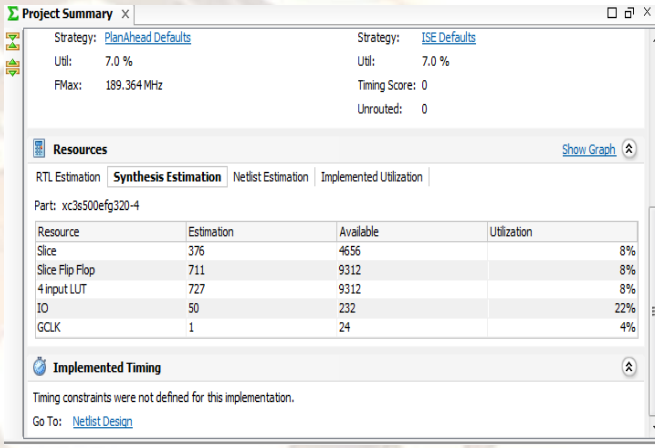


Fig: 7 Xilinx PlanAhead 13.1 Simulations

6.1 Graph of Implemented Utilization in PlanAhead 13.1:

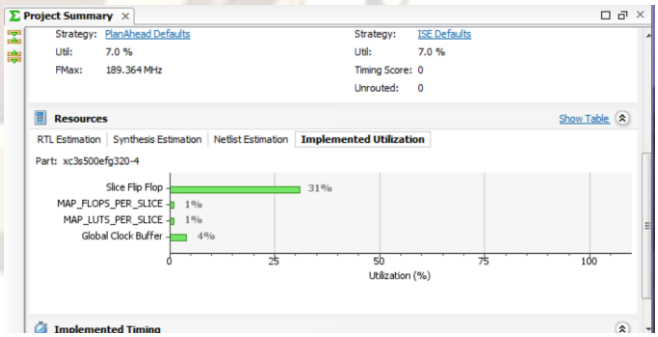


Fig: 8 Utilization in Xilinx PlanAhead 13.1

6.2 Optimized Synthesis Report

Table: 3 shows the Optimized Synthesis Report of Polar to Rectangular Conversion

7. RECTANGULAR TO POLAR CONVERSION

The rectangular to polar coordinate processor is built around the second CORDIC scheme which calculates

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

It takes two 16bit signed words as inputs (Xin, Yin), which are the rectangular coordinates of a point in a 2-dimensional space. The core returns the equivalent Polar coordinates where Rout is the radius and Aout is the angle .The rectangular to polar transformation consists of finding the magnitude and phase angle of input vector. We can immediately recognize the both functions are provided simultaneously by the vectoring mode CORDIC rotator.

The magnitude of the result will be scaled by the CORDIC rotator gain and should be accounted for elsewhere in the system. If the gain is unacceptable, it can be corrected by multiplying the resulting magnitude by the reciprocal of the gain constant.

Maximum Frequency	153.896MHz
Minimum input arrival time before clock	6.046ns
Maximum output required time after clock	4.283ns
Maximum combinational path delay	No path found
Total delay	4.281ns
Total REAL time to Xst completion	19.00 secs
Total CPU time to Xst completion	18.93 secs

Table: 3 shows the Optimized Synthesis Report of Polar to Rectangular Conversion

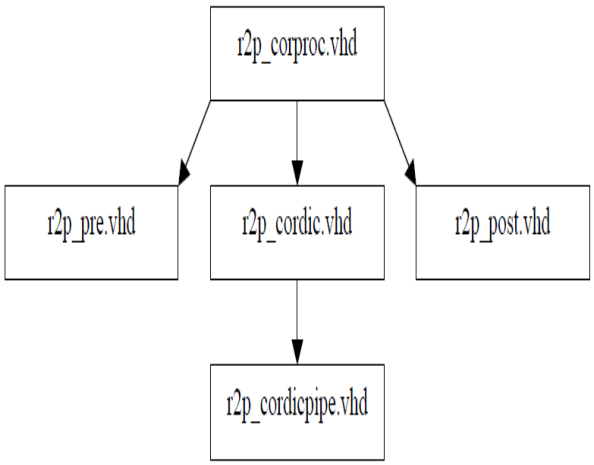


Fig: 9 Core Structure of Rectangular to Polar Conversion

7.1 IO Ports

Port	Width	Direction	Description
CLK	1	Input	System Clock
ENA	1	Input	Clock enable signal
Xin	16	Input	X-coordinate input. Signed value
Yin	16	Input	Y-coordinate input. Signed value
Rout	20	Output	Radius output. Unsigned value.
Aout	20	Output	Angle (θ) output. Singed/Unsigned value.

Table: 4 IO Ports of Polar to Rectangular conversion

8. SIMULATION RESULT OF RECTANGULAR TO POLAR CONVERSION

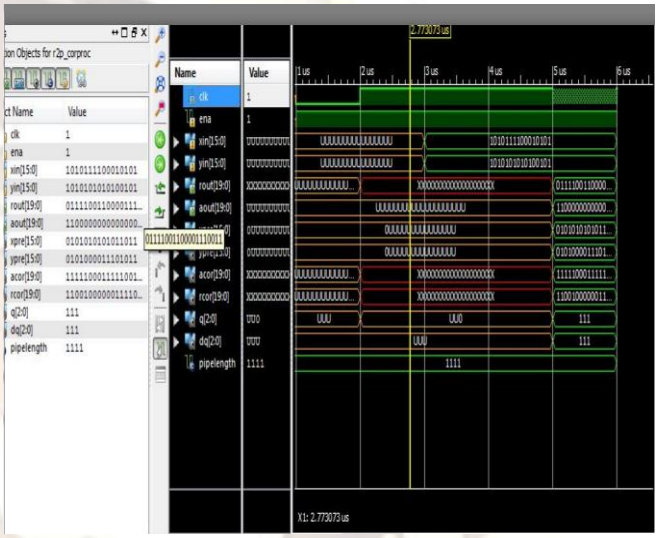


Fig: 10 Simulation Result of Rectangular to Polar Conversion

8.1 Synthesis Report

Table: 7 Synthesis Report of Rectangular to Polar Conversion

8.2 POWER REPORT:

Total Supply power	0.081w
Total Current	0.002A
Junction Temperature	27.1c
Ambient temperature	82.9c
Source voltage vccint	1.2v

Table: 6 Power Report of Rectangular to Polar Conversion

9. OPTIMIZATION IN BOTH AREA AND POWER OF RECTANGULAR TO POLAR CONVERSION USING Xilinx PlanAhead 13.1

This can be done in Xilinx PlanAhead 13.1 by assigning required pin constraints from results obtained in Xilinx 13.1. This can be done by creating Ucf file in PlanAhead. Optimization can be analysed by observing below results. There will be decrease in both area and delay.

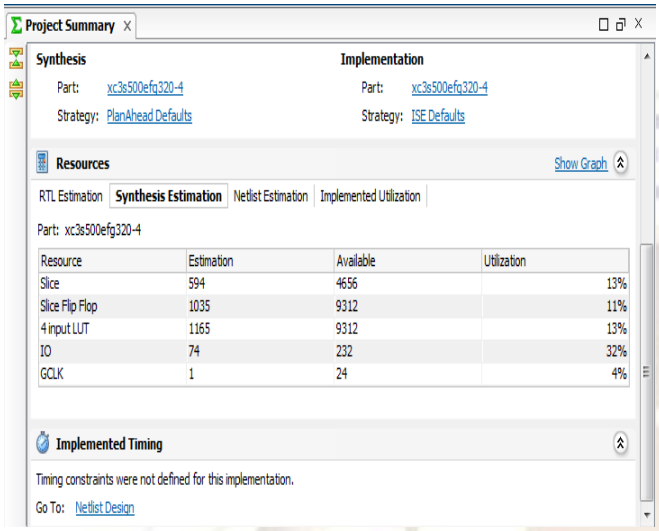


Fig 11 Xilinx PlanAhead 13.1 Simulation

9.1 Graph of Implemented Utilization in PlanAhead 13.1:

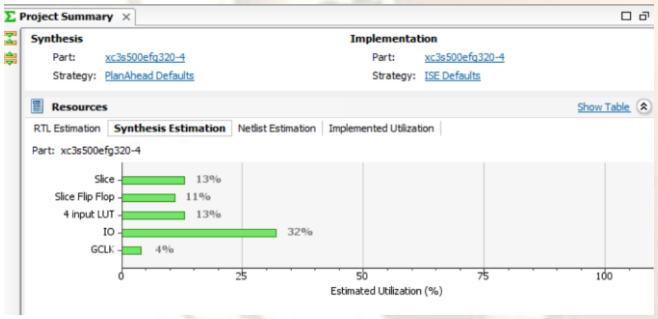


Fig: 12 Utilization in Xilinx PlanAhead 13.1

9.2 Optimized Synthesis Report

Maximum Frequency	153.899MHz
Minimum input arrival time before clock	6.03ns
Maximum output required time after clock	4.21ns
Maximum combinational path delay	No path found
Total delay	4.03ns
Total REAL time to Xst completion	12 secs
Total CPU time to Xst completion	11.9secs

Table: 7 Optimized Synthesis Report of Rectangular to polar Conversion

10. CONCLUSIONS

Pipelined CORDIC Processor is designed and optimized both in area and Power. The Throughput can be increased by decreasing latency in each individual Pipeline stages. Scale factor errors can be reduced by scaling down successive iterations or by reducing no of micro rotations.CORDIC is generally faster than other approaches when a hardware multiplier is unavailable , or when the number of gates required to implement the functions, it supports should be minimized (e.g., in an FPGA). The CORDIC algorithm has become a widely used approach to elementary function evaluation when the silicon area is a primary constraint. This Pipelined Architecture can be easily integrated in VLSI Technology due to its regularity and modularity. The Architecture can be used in digital sine cosine generator in various dsp applications.

REFERENCES:

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

[2] LindlbauerN.,www.cnmat.berkeley.edu/~norbert/CORDIC/node3.htmlhtt.

[3] Krsti M., Troyu A., Muharutnu K. and Grass E., *.Optimized low-power synchronizer design for the IEEE 802.11a standard., Frankfurt (Oder), Germany, 2003.*

[4] Proakis J. G.,Manolakis D. G., *.Digital signal processing principles, algorithms and applications., Prentice Hall, Delhi, 2008.*

[5] N. Takagi, T. Asada, and S.Yajima, "*Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," IEEE Transactions on Computers, vol. 40, no. 9, 1991.*

[6] J.-A. Lee and T. Lang, "*Constant-Factor Redundant CORDIC for Angle Calculation and Rotation," IEEE Transactions on Computers, vol. 41, no. 8, 1992.*

[7] H. Lin and A. Sips, "*On-Line CORDIC Algorithms," IEEE Transactions on Computers, vol. 39, 1990, pp. 1038 1052.*

[8]. J. Duprat and J.-M. Muller, "*The CORDIC Algorithm: New Results for Fast VLSI Implementation," IEEE Transactions on Computers, vol. 42, 1993, pp. 168–178.*

[9]. H. Dawid and H. Meyr, "*The Differential CORDIC Algorithm: Constant Scale Factor Redundant Implementation without Correcting Iterations," IEEE Transactions on Computers, vol. 45, no. 3, 1996.*

[10]. S. Wang, V. Piuri, and E. Swartzlander, "*Hybrid CORDIC Algorithms," IEEE Transactions on Computers, vol. 46, 1997, pp. 1202–1207.*

[11]. C. Li and S. Chen, "*A Radix-4 Redundant CORDIC Algorithm with Fast On-Line Variable Scale Factor Compensation," in Int. Symposium of Circuit and Systems (ISCAS'97), HongKong, Jun. 1997, pp. 639–642.*

[12]. R.R. Osorio, E. Antelo, J. Bruguera, and E. Zapata, *"Digit On-Line Large Radix CORDIC Rotator," in Int. Conf.On Application-Specific Array Processors, Strasbourg, France, Jul. 1995, pp. 247–257.*

[13]. Shen-Fu Hsiao and Jean-Marc Delosme, "*Householder CORDIC Algorithm," IEEE Transactions on Computers, vol. 44, no. 8, 1995.*

[14]. Shen-Fu Hsiao and Jen-Yin Chen, *"Design, Implementation and Analysis of a New Redundant CORDIC Processor with Constant Scaling Factor and Regular Structure," Journal of VLSI Signal Processing, vo. 20, 1998, pp. 267–278.*

Authors Profile:

G.Sandhya was born in ponnur, Guntur District, Andhrapradesh, INDIA. She received B.Tech degree in EEE from SRI VENKATESWARA UNIVERSITY, Tirupathi. Her Research interests include Digital system design, VLSI Testing. Presently, she is in KL UNIVERSITY persuing M.Tech VLSI COURSE.

S. Inthiyaz was born in Vijayawada, Krishna District, and Andhra Pradesh, India. He received B.Tech degree in ECE from JNTU, Hyderabad and M. Tech from JNTU Kakinada, Kakinada. His research interests include Low power VLSI and design of fault model circuits. Presently, he is working as an Assistant Professor in KL University, Guntur.

Dr. Fazal Noorbasha was born on 29th April 1982. He received his, B.Sc. Degree in Electronics Sciences from BCAS College, Bapatla, Guntur, A.P., and Affiliated to the Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2003, M.Sc. Degree in Electronics Sciences from the Dr. HariSingh Gour University, Sagar, Madhya Pradesh, India, in 2006, and M.Tech. Degree in VLSI Technology, from the North Maharashtra University, Jalgaon, Maharashtra, INDIA in 2008, and Ph.D.