FPGA Implementation of Fast and Area Efficient CORDIC algorithm

M. Chinnathambi
Department of Electronics and Communication,
Thiagarajar College of Engineering,
Madurai, India.
chinnaa92@gmail.com

N. Bharanidharan
Department of Electronics and Communication,
Thiagarajar College of Engineering,
Madurai, India.
bharani2410@gmail.com

S. Rajaram
Department of Electronics and Communication,
Thiagarajar College of Engineering,
Madurai, India.
rajaram_siva@tce.edu

***Abstract*—This paper presents the fast and area efficient CORDIC (Coordinate Rotation DIgital Computer)algorithm for sine and cosine wave generation. The concepts of pipelining and multiplexer based CORDIC algorithm is used todecrease the critical path delay and reducing the area respectively. A six stage CORDIC is implemented by two schemes followed by four methods, unrolled CORDIC and multiplexer based CORDIC with and without pipelining. The pipelining is included in four stages(excluding first and last stage). An 8-bit CORDIC algorithm for generating sine wave and cosine wave is designed, implementedand compared by all four methods on Xilinx Spartan3E (XC3S250E).**

Keywords—CORDIC algorithm, FPGA, Multiplexer, Pipelining, Unrolled CORDIC, Sine/Cosine

I. INTRODUCTION

The FPGA platform is much better choice for CORDIC algorithm implementation since it combines the flexibility of microprocessors as well as the speed and computational power of ASICs. The CORDIC (COordinate Rotation DIgital Computer) algorithm uses planar rotation and vectoring to compute elementary trigonometric functions when assigned with proper initial conditions. It is one of the iterative algorithm explored by Jack E. Volder [1] and later clearly refined by Walther [2] and others. CORDIC algorithm is very popular since it uses only shifts and adds to perform number of functions including certain trigonometric, vector rotations, hyperbolic, logarithmic functions. In communication applications it is widely used to implement universal modulator [3], demodulator [4]. CORDIC algorithm is used in various applications that includescalculators, mathematical coprocessor units, clock recovery circuits, waveform generators.

There are two different modes of CORDIC algorithm as rotation and vector mode. In the rotation mode, CORDIC is used for converting a vector from polar form to rectangular form and vector mode makes the reverse operation.

The objective of this paper is to design, analyze and compare how an FPGA based unrolled CORDIC performs when multiplexers are used instead of shifters and adders with and without pipelining[5-8].

The organization of the paper is as follows: Section II gives basics of CORDIC algorithm. The general unrolled CORDIC algorithm is presented in Section III; section IV explains the Multiplexer based CORDIC; section V describes the Unrolled CORDIC with pipelining; section VIpresents pipelined Multiplexer based unrolled CORDIC. Section VII presents the results and discussion followed by conclusion in Section VIII.

II. BASICS OF CORDIC ALGORITHM

The CORDIC algorithm is one of the iterative method toperform vector rotations for arbitrary angles using shifts and adds. Planar rotation for any vector A of (Xj, Yj) can be defined in matrix form as:

$$\begin{bmatrix} Xj \\ Yj \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} Xi \\ Yi \end{bmatrix} \quad (1)$$

With a small rearrangement, the stepwise rotation is performed by

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = cos\theta_n \begin{bmatrix} 1 & -tan\theta_n \\ tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} Xn \\ Yn \end{bmatrix} \quad (2)$$

The angle parameter for each step will be,

$$\theta_n = \arctan\left(\frac{1}{2^n}\right) \quad (3)$$

Rotation angle θ is represented as,

$$\sum_{n=0}^{\infty} s_n \, \theta_n = \theta \quad (4)$$

whereSn={−1; +1} depends on Zi. Rewriting the equation for rotation,

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = cos\theta_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \quad (5)$$

Cosθn can be considered as a constant K and computed at the end and for simplicity of hardware, multiplications are replaced with shift operations. Residue Z which gives the angle difference between the expected rotation and the iterative rotations is defined as:

$$Z_{n+1} = \theta - \sum_{i=0}^{n} \theta i = \theta - \sum_{i=0}^{n} \arctan\left(\frac{1}{2^n}\right) \quad (6)$$

Rotation parameter,

978-1-4799-6266-2/14/$31.00 © 2014 IEEE

$$S_n=\begin{cases}-1\ if Z_n<0\\ 1\ if Z_n\geq 0\end{cases} \quad (7)$$

With proper selection of initial values of Xi, Yi and Zi, the required function is performed using (5-7) equations.

In rotation mode, we should make Z to zero. For generating sine and cosine values, the initial values should be Xi=1, Yi=0, and Zi=θ.

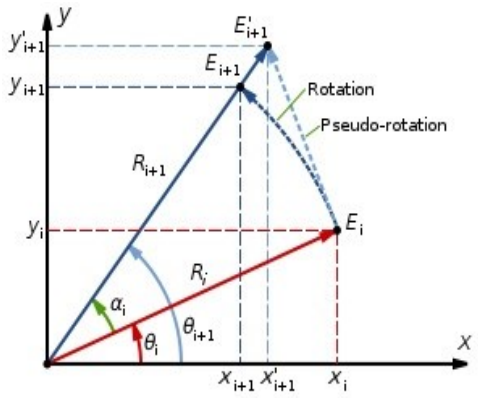

Fig.1. CORDIC angle rotations

Then at the end of the iteration, the final equation will be,

$$\begin{bmatrix}X_i\\ Y_i\\ Z_i\end{bmatrix}=\begin{bmatrix}cos\theta\\ sin\theta\\ 0\end{bmatrix} \quad (8)$$

III. General Unrolled CORDIC Algorithm

The architecture of the six stage unrolled CORDIC is shown in Fig2. This consists of only shifters, adders and subtractors andif the number of stages increase, accuracy of computation also increase. Depending on the most significant bit of previous angle, addition or subtraction of the angle value takes place in every rotation of the vector. Division is performed by just doing right shift using shift registers which gives theadvantage of using less hardware for division. Initially, Xi=1 and Yi=0 for sine and cosine wave generation. These initial values are shifted by ibits, where i is the integer {0, 1, 2, 3, 4, 5} which makes division of x and y by 1,2, 4, 8, 16, 32for each stage. In this rotation mode, the given vector is iteratively rotated to form new vectors at the intermediate stages to get the desired angle, Zi.

If the initial conditions are Xi=1 and Yi=0 then the resulting discrete sine and cosine values will vary from -1 to 1. These fractional values are not realizable in FPGA easily. Hence to make the discrete sine and cosine values to vary from -100 to 100, the initial values are multiplied by 100 i.e., Xi=100 and Yi=0.

For representing sine and cosine values in the above range 8 bit CORDIC is used as it can represent -128 to 127. Generally the constant K as in the equation (5) should be multiplied to the final results after end of iteration. But to improve the accuracy, it is multiplied to initial conditions itself. K=0.611 for six stages of CORDIC and so initial values are given as Xi=61(100x0.611), Yi=0.

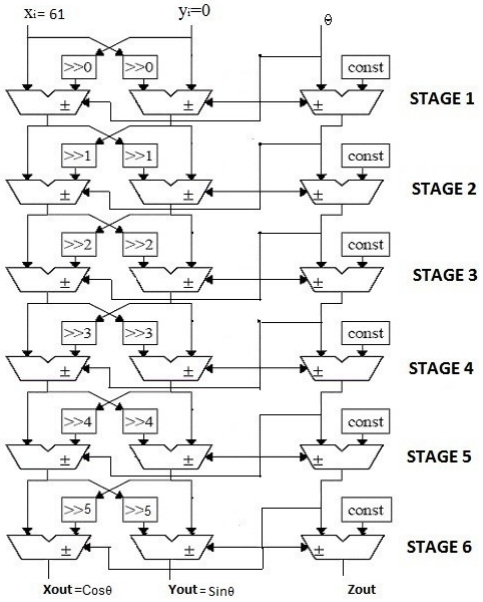

Fig.2. General unrolled CORDIC structure.

For the generation of sine and cosine values, Zi should be varied for each clock pulse. The step size is the difference between two Zi values. For more accuracy, the step size should be less. For the step size of 15°, Zi value varies as 0,15,30,..90 producing the values of 0,25,50,…100 at Y_6(sine values) and 100,…,50,25,0 at X_6(cosine values) for each clock pulse. By using the quadrature symmetry property of sine and cosine waves, remaining values are computed.

IV. Multiplexer Based CORDIC Structure

The concept for reducing the area of the CORDIC structure by using multiplexer is proposed for the ASIC implementation in paper [3]. For the FPGA based implementation this concept is adopted. Multiplexer is used instead of first three stages of general unrolled CORDIC. The output of first stage in original unrolled CORDIC architecture is equal to Xi as Yi=0 and so the output of first stage is given as

$$\begin{aligned}Y_1&= Xi = 61\\ X_1&= Xi = 61\\ Z_1&= Zi - 45\end{aligned} \quad (9)$$

In the first iteration stage, Z1 is calculated by subtraction since Zi is always +ve as it varies from 0 to 90. If Z1 is positive, then second stage output is represented as

$$\begin{aligned}Y_2 &= Y_1 - \frac{X_1}{2} = \frac{X_1}{2} = 31\\ X_2 &= X_1 + \frac{Y_1}{2} = \frac{3X_1}{2} = 91\end{aligned} \quad (10)$$

If Z1 is negative, then second stage output will be,

$$Y_2 = Y_1 + \frac{X_1}{2} = \frac{3X_1}{2} = 91$$

$$X_2 = X_1 - \frac{Y_1}{2} = \frac{X_1}{2} = 31 \quad (11)$$

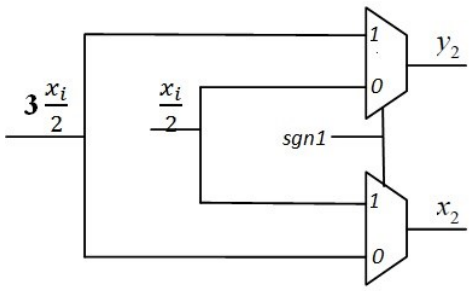

Fig.3. Multiplexers to replace second stage of unrolled CORDIC

As the second stage output is fixed, two Multiplexers are used as shown in Fig.3. Similarly the third stage is implemented by using four multiplexers with the following equations. Z2 is computed by using the equation (6) and it is used as the selection line for third stage multiplexers

.For Z1 = + ve, Z2 = + ve

$$Y_3 = Y_2 + \frac{X_2}{4} = \frac{3X_i}{2} + \frac{X_i}{8} = \frac{13X_i}{8} = 99$$

$$X_3 = X_2 - \frac{Y_2}{4} = \frac{X_i}{2} - \frac{3X_i}{8} = \frac{X_i}{8} = 07 \quad (12)$$

For Z1 = - ve, Z2 = + ve

$$Y_3 = Y_2 + \frac{X_2}{4} = \frac{X_i}{2} + \frac{3X_i}{8} = \frac{7X_i}{8} = 53$$

$$X_3 = X_2 - \frac{Y_2}{4} = \frac{3X_i}{2} - \frac{X_i}{8} = \frac{11X_i}{8} = 83 \quad (13)$$

For Z1 = + ve, Z2 = - ve

$$Y_3 = Y_2 - \frac{X_2}{4} = \frac{3X_i}{2} - \frac{X_i}{8} = \frac{11X_i}{8} = 83$$

$$X_3 = X_2 + \frac{Y_2}{4} = \frac{X_i}{2} + \frac{3X_i}{8} = \frac{7X_i}{8} = 53 \quad (14)$$

For Z1 = - ve, Z2 = - ve

$$Y_3 = Y_2 - \frac{X_2}{4} = \frac{X_i}{2} - \frac{3X_i}{8} = \frac{X_i}{8} = 07$$

$$X_3 = X_2 + \frac{Y_2}{4} = \frac{3X_i}{2} + \frac{X_i}{8} = \frac{13X_i}{8} = 99 \quad (15)$$

Multiplexer for three stages reduce the one clock pulse delay for first output than the ordinary unrolled CORDIC. When the adders are replaced with Multiplexers, the area is reduced up to 3rd stage. For the replacement of adders and shifters with Multiplexers, there is exponential increase in the number of Multiplexers i.e., 6, 14, 30 Multiplexers are needed for constructing three, four and five stages respectively.

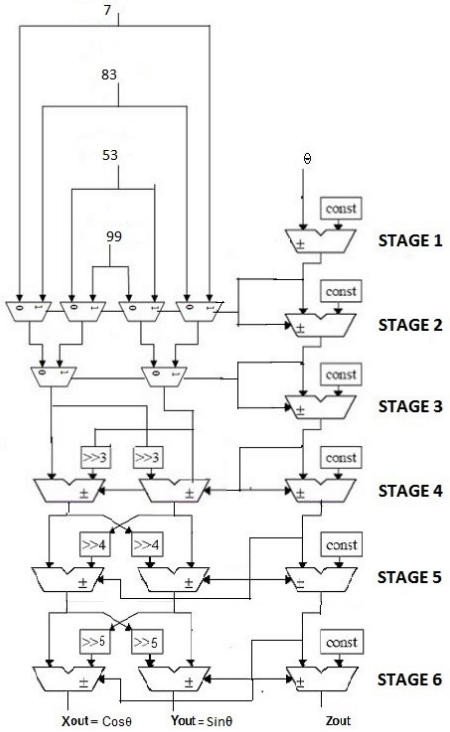

Fig.4. Multiplexer based CORDIC for first three stages and ordinary CORDIC for remaining three stages

V. Unrolled Cordic With Pipelining

Generally a pipeline is a set of simple data processing elements which are connected in series, so that the first element output is the input of the next element. Pipelining concept is mainly used to decrease the critical path delay and making the system suitable for high speed applications. Hence unrolled CORDIC with pipelining has more maximum frequency of operation than the ordinary one. But there are some disadvantages in using pipelining such as it increases area on FPGA and also there is N-Clock delays for the first output when N pipeline registers are used. After that N clock delays, output will appear one by one for each clock pulse. The position & number pipelined registers are iteratively computed and the optimized result is taken which uses pipeline registers at intermediate four stages(excluding first and last stages) as shown in Fig.5.

VI. Pipelined Multiplexer Based Cordic

Pipelining in CORDIC increases area but improves the speed of operation and increases area while Multiplexer based CORDIC reduces the area utilization and decreases speed of operation. Hence there is trade-off between area and speed of operation in pipelined multiplexer based CORDIC[9]. First three stages are replaced by multiplexers while fourth and fifth stages are pipelined using registers.

Due to pipelining at two stages there is increase in two clock pulse delay for the first output while the usage of multiplexers reduces one clock pulse delay for first output and so there is only one clock pulse delay for the first output than usual.

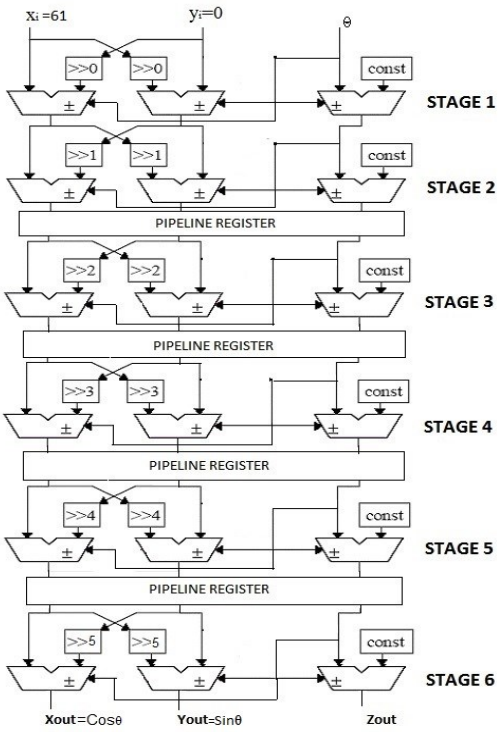

Fig.5. Unrolled CORDIC with pipelining at intermediate stages

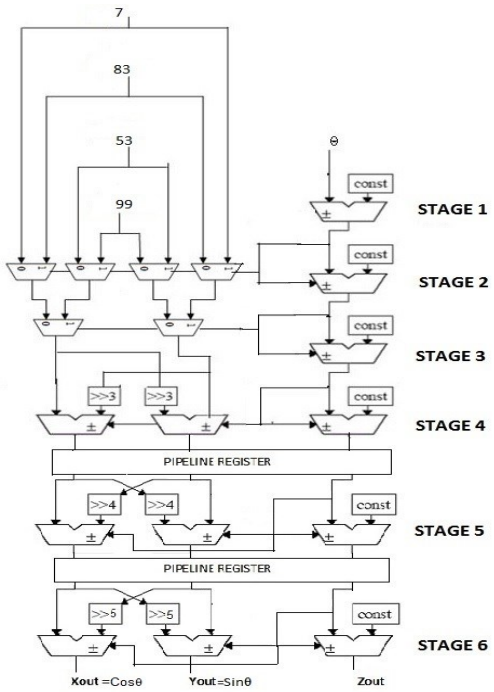

Fig.6. Pipelined multiplexer based CORDIC

VII. Results and Discussion

All the four schemes are implemented in Xilinx Spartan3E (XC3S250E). The first output of N stage unrolled 8 bit CORDIC appears only at (N+1)thclock pulse due to N iterations. But after the (N+1) clock cycles, output will appear for each clock cycle.

TABLE I. COMPARISON OF FOUR SCHEMES BASED ON FIRST OUTPUT APPEARANCE

Scheme	Clock pulse at which first output appears
General unrolled CORDIC	7
Multiplexer based CORDIC	6
Pipelined unrolled CORDIC	11
Pipelined multiplexer based CORDIC	8

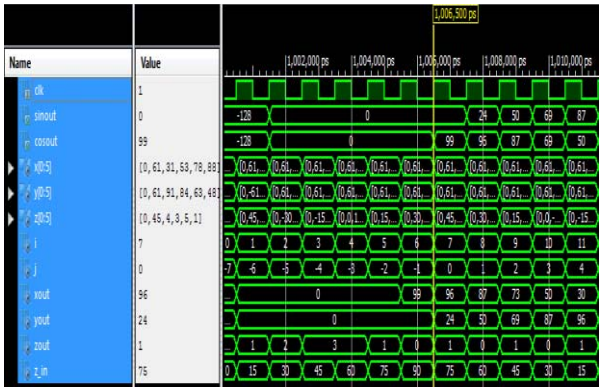

Fig.7. Simulation output of Unrolled CORDIC

In Fig.7, the "clk" is the input signal which decides the frequency of sine and cosine wave generation while "sinout" is the discrete sine values and "cosout" is the discrete cosine values. In unrolled CORDIC, due to six iteration stages output appears at 7th clock pulse. Since first stage is replaced without using any Multiplexers, the clock pulse will appear on sixth clock pulse itself in multiplexer based CORDIC.

Pipelined unrolled CORDIC uses 4 pipeline registers and so first output will appear on 11th clock pulse(7+4=11) while pipelined multiplexer based CORDIC has first output at 10th clock pulse(6+2=11). Comparatively there is more percentage of error in multiplexer based schemes due to quantization effects of initial values.

When implemented in Xilinx FPGA's, there is more trade-off between critical path delay and area on FPGA as shown in table II.

TABLE II. IMPLEMENTATION OF FOUR SCHEMES IN XILINX

Scheme	Number of slice Flip Flop	Number of occupied slices	Critical path delay (ns)
General unrolled CORDIC	134	160	9.0
Multiplexer based CORDIC	109	131	9.6
Pipelined unrolled CORDIC	187	166	6.3
Pipelined multiplexer based CORDIC	125	133	6.1

Thus pipelined unrolled CORDIC provides maximum operating speed as the critical path delay is reduced than the unrolled CORDIC without pipelining with the cost of area. Multiplexer based CORDIC provides more area reduction but comparatively less speed of operation.

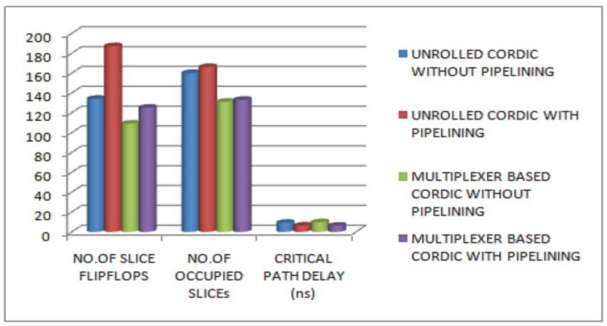

Fig.8. Comparison of four schemes based on results obtained by Xilinx implementation

VIII. CONCLUSION

In this paper, detailed analysis of four schemes of CORDIC for sine and cosine wave generation is made by comparing the results obtained from Xilinx FPGA implementation. Due to the changes we made in initial values as said in section III, we got only 2.42% error in computing sine and cosine values which vary from -100 to 100. Among the four methods, multiplexer based CORDIC with pipelining is much better as it has good tradeoff between area and critical path delay. From the results, as said in the above sections, the multiplexer based CORDIC reduces both area and speed of operation while pipelining increases both area and speed of operation. Hence based on the particular application, one of the four scheme is selected.

REFERENCES

[1] J.E. Volder, “The CORDIC Trigonometric Computing Technique”, IRE Transactions on Electronic computer, vol. EC-8, pp. 330-334, 1959.

[2] J. Walther, “a unified algorithm for elementary functions,” proc. Spring joint comp. con & vol.38, pp.379-385, 1971.

[3] Vankka, J.; Kosunen, M.; Hubach, J.; Halonen, K.; , "A CORDIC-based multicarri QAM modulator," Global Telecommunications Conference, 1999. GLOBECOM’99, vol.1A,no.,pp. 173-177vol.1a,1999 .

[4] Chen, A.; McDanell, R.; Boytim, M.; Pogue, R.;, "Modified CORDIC demodulator implementation for digital IF-sampled receiver," Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE , vol.2, no., pp.1450-1454 vol.2, 14-16 Nov 1995.

[5] V.Naresh ,B.Venkataramani and R.Raja “An area efficient multiplexer based CORDIC” 2013 International Conference on Computer Communication and Informatics (ICCCI -2013), Jan. 04 – 06, 2013, Coimbatore

[6] Peter Nilsson, “complexity reduction in unrolled CORDIC architectures “ Electronics, circuits, and systems,2009.ICECS 2009, pp.868-871.

[7] Nilsson, P “Complexity reductions in unrolled CORDIC architectures” Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on 13-16 Dec. 2009,pp 868 – 871

[8] Naveen Kumar, Amandeep Singh Sappal ”Coordinate Rotation Digital Computer Algorithm: Design and Architectures” (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 4, 2011

[9] Deprettere, E.; Dewilde, P.; Udo, R.;, "Pipelined cordic architectures for fast VLSI filtering and array processing," Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84. , vol.9, no., pp. 250- 253, Mar 1984.

[10] U. Meyer-Baese, “Digital Signal Processing with Field Programmable Gate Arrays”, 3rd Edition. Berlin: Springer, 2007