Arun Babu, Monisha Manohar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 3, May-Jun 2013, pp.980-983

Power saving Run Time Reconfigurable Cordic Processor

Arun Babu*, Monisha Manohar**
* (Department of ECE, Amrita Vishwavidyapeetham University, Kollam)
** (Department of ECE, Amrita Vishwavidyapeetham University, Kollam)

ABSTRACT

The CORDIC algorithm provides an efficient method of computing trigonometric functions by rotating a vector through some angle, specified by its coordinates. This rotation is obtained by performing a number of micro rotations through elementary rotation angles, into which the total rotation angle has been decomposed. This paper presents the design and implementation of a runtime reconfigurable CORDIC processor which can be used for various calculations including rectangular to polar and polar to rectangular co-ordinate conversion. Efficient floor planning is done using Xilinx Plan Ahead 14.2 to reduce the area and device utilization.

Keywords - *CORDIC, Power saving, Reconfigurable, Floor Planning, co-ordinate conversion.*

I. INTRODUCTION

Trigonometric function evaluations have been used in countless applications in the field of Digital Signal Processing (DSP) [1]. The COordinate Rotation DIgital Computer (CORDIC) algorithm has become very popular due to its simplicity and efficient evaluation of the trigonometric and co-ordinate calculation and can also be used for digital waveform synthesis. In this paper, we have presented the design of pipelined architecture for the computation of flexible and scalable digital Sine and Cosine values using the CORDIC algorithm. We have designed the processor in such a way that the precision of the calculations can be given at runtime and also we can select one among the two cores i.e. rectangular to polar or polar to rectangular conversions, thus the power consumption is reduced as compared to the previous design. Saving area on FPGA is one of the major challenges in the designers' perspective. The design has been synthesized and implemented on a XilinxVirtex-5 (XUPV5LX110T) device using 14.2 ISE design tool suite and results are shown and discussed. We have efficiently floor planned using Plan Ahead 14.2 for efficient utilization of resources.

II. CORDIC ALGORITHM – AN OVERVIEW

Jack E. Volder's CORDIC algorithm is derived from general equations for vector rotation. The theory of CORDIC computation is to decompose the desired rotation angle into the weighted sum of a set of predefined elementary rotation angles, each of which can be accomplished with simple shift-add operation for a desired rotational angle θ This section describes the mathematics behind the CORDIC algorithm.

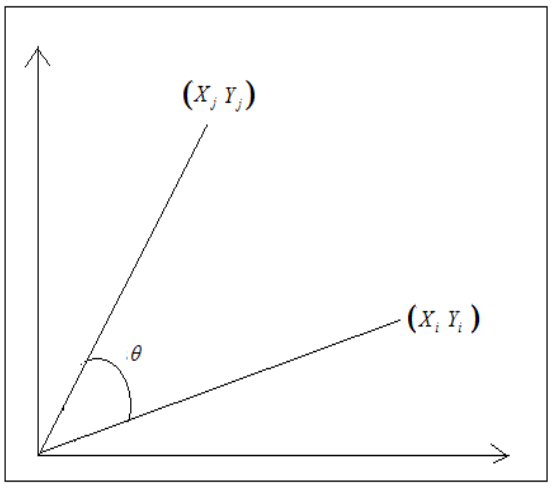


Fig I.I Basic of CORDIC Rotation

The θ 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 [2][9]:

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

The above can be modified by eliminating the $\cos\theta_n$ factor.

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

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 [3][7]

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

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

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

Where

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

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

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

Combining the above results,

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

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.

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

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

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

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.

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

Because the coefficient K is pre-computed and taken into account at a later stage, we can write as,

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

Or as

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

At this point a new variable called 'Z' is introduced. Z represents the part of the angle θ which has not been rotated yet [4].

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

For every step of the rotation S_n is computed as a sign of Z_n.

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

III. INDENTATIONS AND EQUATIONS

This algorithm is commonly referred to as driving Z to zero. The CORDIC core computes:

$$[X_j, Y_j, Z_j] = [P(X_i \cos(Z_i) - Y_i \sin(Z_i)),\ P(Y_i \cos(Z_i) + X_i \sin(Z_i)),\ 0]$$

There's a special case for driving Z to zero:

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

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

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

For this scheme there are two special cases:

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

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

IV. ARCHITECTURE

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

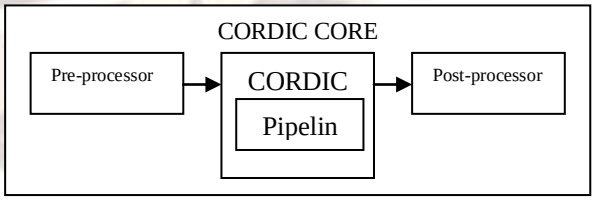


Fig: IV.I Architecture of CORDIC Processor

Pipeline of Cordic blocks [5]. Each pipeline block represents a single step in the iteration processes. Because of the arc tan 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

pre-processor. 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. 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. Each pipe or iteration step is performed by the Cordic core. It contains the tan table for each iteration and the logic needed to manipulate the X, Y and Z values[6][8].

V. IMPLEMENTATION AND RESULTS

The CORDIC[10] core has been coded in VHDL using Xilinx 14.2 ISE. Efficiently resource utilization is achieved by using Plan Ahead 14.2 and analyzed the results with the original un optimized one. The designed is successfully simulated and verified the output using Isim Simulator and Synthesized using XST too and implemented in Xilinx Virtex-5 board (XUPV5LX110T). The simulation result and other results are shown

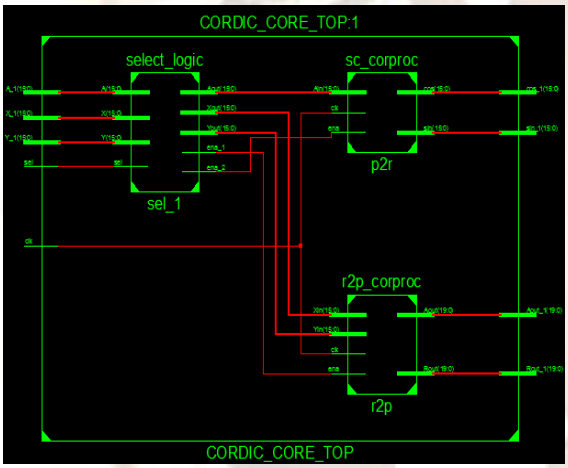
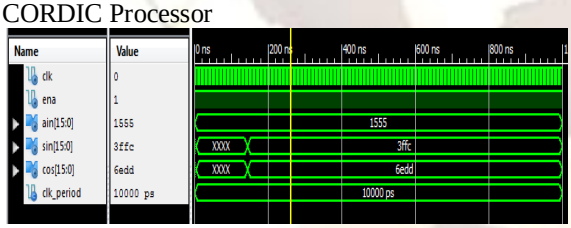


Fig: V.I RTL Schematic of the reconfigurable CORDIC Processor



Simulation results of Sine and Cosine of 30

	0 deg	30 deg	45 deg	60 deg	90 deg
Sin	0x01CC	0x3FFC	0x5A82	0x6EDC	0x8000
Cos	0x8000	0x6EDD	0x5A83	0x4000	0x01CC
Sin	0.01403	0.49998	0.70709	0.86609	1.00000
Cos	1.00000	0.86612	0.70712	0.50000	0.01403

Table V.I: Sin/Cos outputs for some common angles

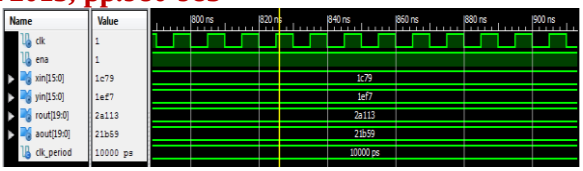


Fig: V.III Simulation results of rectangular to polar conversion

For efficient area optimization we used Plan Ahead 14.2 for better utilization of resources. Without optimization both cores are scattered in the FPGA thus device utilization is not that optimum. Thus by using floor planning techniques it is possible to get better utilization of the resources[11]. The snap shots are shown below.

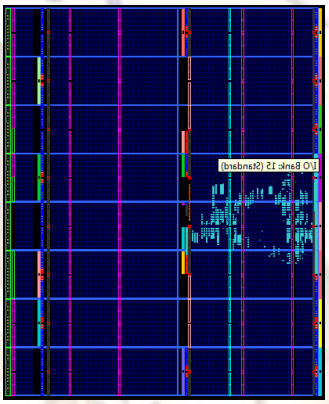


Fig: V.IV Snapshot from Plan Ahead showing the slice utilization.

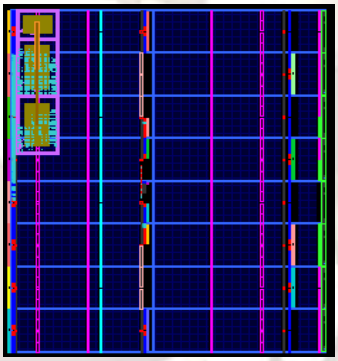


Fig: V.V Optimized floor plan

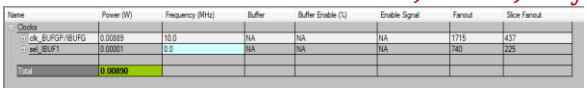
The device utilization of the optimized and the non optimized can be summarized as,

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1711	69120	2%
Number of Slice LUTs	1852	69120	2%
Number of fully used LUT-FF pairs	1673	1890	88%
Number of bonded IOBs	122	640	19%
Number of BUFG/BUFGCTRLs	2	32	6%

Fig: V.VI Device utilization of non optimized architecture

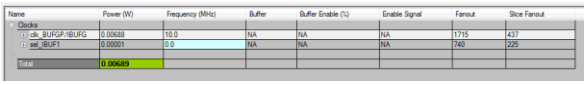
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1211	69120	2%
Number of Slice LUTs	1452	69120	2%
Number of fully used LUT-FF pairs	1273	1890	68%
Number of bonded IOBs	90	640	14%
Number of BUFG/BUFGCTRLs	2	32	6%

Fig: V.VII Device utilization of optimized architecture



Name	Power (W)	Frequency (MHz)	Buffer	Buffer Enable (%)	Enable Signal	Fanout	Slice Fanout
Clocks							
clk_BUFGP/IBUFG	0.00889	10.0	NA	NA	NA	1715	437
sel_IBUF1	0.00001	0.0	NA	NA	NA	740	225
Total	0.00890						

Fig: V.VIII Power consumption of non optimized architecture



Name	Power (W)	Frequency (MHz)	Buffer	Buffer Enable (%)	Enable Signal	Fanout	Slice Fanout
Clocks							
clk_BUFGP/IBUFG	0.00688	10.0	NA	NA	NA	1715	437
sel_IBUF1	0.00001	0.0	NA	NA	NA	740	225
Total	0.00689						

Fig: V.IX Power consumption of non optimized architecture

From the above results it is evident that by properly optimizing the floor plan the device utilization is improved and the power consumption is reduced.

Now using the select logic only one of the cores needs to be active at a particular time keeping the other one idle. Hence it is possible to further reduce the power consumption i.e.: runtime power saving architecture.

VI. CONCLUSION

A runtime reconfigurable power saving CORDIC Processor is designed and successfully implemented and the result is verified. Using efficient way of floor planning we are able to utilize the resources properly and reduce the power consumption. A select logic is utilized for selecting the particular core for the intended application thus by keeping the other one idle. Thus the power consumption is minimized. The number of iterations of the CORDIC processor is also reconfigurable, which yields a more reliable convergence of the result. This can be well adapted to the implementation of FFT, DCT and other DSP applications where multiplications are needed. This flexible core can be well utilized for the generation of digital signal waveforms.

REFERENCES

Journal Papers:

[1] Amritakar Mandal and Rajesh Mishra *Journal of Signal Processing Theory and Applications (2012) 1: 27-35 doi:10.7726/jspta.2012.1003*

Books:

[2] Mios D Ercegovac and Tomas Lang Digital *Arithmetic* (Morgan Kaufman Publishers).

Proceedings Papers:

[3] J.E. Volder, *"The CORDIC Trigonometric Computing Technique", IRE Trans. Electronic Computers, vol. EC-8, pp. 330-334*, Sept. 1959.

[4] J.S. Walther, "A unified Algorithm for elementary functions", Proc. AFIPS Spring Joint Computing Conf., vol. 38, pp. 379-385, 1971.

[5] R. Bakthavatchalu, M.S. Sinith, P.Nair, K.Jismi, *"A comparison of pipelined, parallel and iterative CORDIC design on FPGA", Industrial and Information Systems (ICIIS), 2010 International conference,* pp 239-243, September 2010

[6] Antelo, E., Lang, T. and Bruguera, J. D., 2000. *Very-high radix CORDIC rotation based on selection by rounding.J. VLSI Signal Processing, 25:2, 141–153*

[7] K. Maharatna, S. Banerjee, A.Troya and E. Grass *,"Virtually Scalingfree adaptive CORDIC rotator", IEEE Proc, Comput. Digit Tech ,Volume 151, Issue 6,p 448-456, November 2004*

[8] K. Maharatna, S. Banerjee, E. Grass ,M.Krstic and A.Troya, *"Modified Virtually Scaling-free adaptive CORDIC rotator Algorithm and architecture", IEEE Transactions on circuits and systems for video technology, Vol.15, No.11, November 2005*

[9] E.I.Garcia, R.Cumplido, M.Arias,*'Pipelined CORDIC design on FPGA for digital sine and cosine waves generator", IEEE 2006 3rd International conference, pp 1-4, December 2006*

[10] Y.H.Hu, *"The quantization effects of cordic algorithm", IEEE transactions on signal processing, Vol 40, No 4, April 1992*

[11] K.Kota, J.R.Cavallaro, *"Numerical accuracy and hardware tradeoffs for cordic arithmetic for special purpose processors", IEEE transactions on computers, Vol 42, No 7, July 1993.*