International Journal on Emerging Technologies (Special Issue on ICRIET-2016) **7**(2): 1-4(2016)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

FPGA Implementation of Redundant CORDIC Processor

Prof. J. M. Rudagi* *and Dr. S Subbaraman*****
[*]KLE College of Engineering and Technology Chikodi/ECE Department, Chikodi, India
[**]WCE Sangli /ECE Department, Sangli, India

(Corresponding author: J.M. Rudagi)
(Received 16 September, 2016 Accepted 19 October, 2016)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Many Digital signal processing (DSP) applications are based on real time constraints. On account of this, conventional processors are not suitable for modern day DSP systems. Thus leading major issues pertaining to processors are latency and throughput. In order to overcome these issues and there by improvising in terms of performance, CORDIC is one such hardware efficient algorithm and its current trend of hardware intensive signal processing. It efficiently performs all elementary functions such as trigonometric, logarithmic, hyperbolic and exponential functions which are used in DSP systems. In this paper redundant Radix-2 CORDIC Architectures for 16-bit and 32 bit has been, implemented on Xilinx 14.2 FPGA platform, Simulated on ISim simulator and synthesized on Vertex 5 FPGA device.

Keywords: CORDIC (Coordinate Rotational Digital Computer), FPGA (Field Programmable Gate Array), DSP (Digital Signal Processing)

I . INTRODUCTION

J.E. Volder developed CO-ordinate Rotation Digital Computer (CORDIC) in 1959 to compute the rotation of two dimensional vectors [1]. Later Walther generalized this algorithm to compute logarithmic, exponential, division, hyperbolic and trigonometric functions [2]. CORDIC is an iterative algorithm for the calculation of the rotation of two dimensional vectors in linear, circular and hyperbolic coordinate systems. This rotation is carried out by a sequence of iterations. Each of this rotation over a prefixed elementary angle (micro rotation) is evaluated by means of addition and shift operations. The number of iteration of radix-2 CORDIC limits its architecture to use in high speed applications.

In this paper, the organization of work as follows. Section II gives the basics of CORDIC algorithm. Section III describes Radix-2 CORDIC Architecture. Introduction to redundant architecture has been described in Section IV. Section V gives simulation results, comparison plots and synthesis report. In the last section VI conclusion of this work has been discussed.

II . BASICS OF CORDIC ALGORITHM

The CORDIC algorithm is coordinate rotation in linear, circular and hyperbolic coordinate systems depending on which function is to be calculated. This is performed in the CORDIC algorithm by rotating a vector through a sequence of arbitrary angles whose algebraic sum approximates the desired rotation angle [1], [2]. These arbitrary angles have the property that vector rotation through each of them may be computed easily with a single shift and add operation. CORDIC operates in two modes: the rotation mode and the vector mode. In rotation mode, angle of rotation and coordinate components of original vector are given, where as in vector mode, only the coordinate of original components are given. Given angle, rotation mode is used to perform general rotation and to compute elementary operations such as trigonometric functions, multiplication, exponential, and hyperbolic functions depending on the coordinate system in which it is being rotated. The vectoring mode can be used to compute the angular argument of the original vector and to compute divisions, logarithmic functions. The number of micro rotations to be performed in both the modes depends on the application. In Cartesian plane rotating a vector by an angle θ can be arranged and equations are as follows

If the rotation angles are restricted so that $\tan(\theta) = \pm 2^{-i}$, the multiplication by the tangent term is condensed to a simple shift operation. Arbitrary angles of rotation are available by performing a series of consecutively smaller micro rotation.

If the decision at each iterations i, is which direction to relate rather than whether or not to rotate, then the cos (θ) term becomes a constant .The iterative rotation can now be expressed as [3]:

$$x_{i+1}=K[x_i + d_i 2^{-i}\, y_i] \quad (1)$$

$$y_{i+1}=K[y_i + d_i 2^{-i}\, x_i] \quad (2)$$

Where,

$K_i = 1/(1+2^{-2i})^{1/2}$; known as scale constant.

d_i is known as decision function.

Removing the scaling constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the K can be functional as part of a system processing gain or by initiating the rotating vector by the reciprocal of the gain of a certain number of iterations. The angle of a composite rotation is uniquely defined by the sequence of the directions of the micro rotations. That series can be represented by a decision vector. All possible decision vectors is an angular measurement system are based on set of binary arctangents. A favorable conversion method uses an additional adder- subtractor that holds the elementary rotation angles at each single iteration. The elementary angles can be expressed in any suitable angular unit either radians or degrees and are stored in small lookup table or it can be hardwired, depending on the implementation. The angle accumulator adds a third difference equation to the CORDIC algorithm

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i}) \quad (3)$$

The CORDIC in rotation mode, a vector (x, y) is rotated by an angle θ. The angle accumulator is initialized with the desired rotation angle θ. The rotation decision per iteration is made to diminish the magnitude of the residual angle in the angle accumulator. Hence, the decision per iteration is based on the sign of the residual angle after each step. Normally, the angle accumulator may be eliminated, if the input angle is already expressed in the binary arctangent base.

III. RADIX-2 CORDIC ARCHITECTURE

Radix-2 architecture is as shown in Fig 1. The main pro of this type of architecture is that the barrel shifters are of fixed size and can be implemented in the wiring. Secondly, instead of requiring storage space that is ROM that holds the arbitrary angle values, need not to be restructured after each iteration because the constants can be hardwired. The LUT values for computing angle accumulator is distributed as constant to each adder in the angle accumulator chain so that the entire CORDIC processor is compact to an array of interconnected adder-subtraction units. Unlike other architectures there is no need of registers which avoids unfolded architecture strictly to behave like combinational circuit. The delay is favorable, but processing time is reduced as compared to other iterative structures. Thus produces speed required for faster applications.

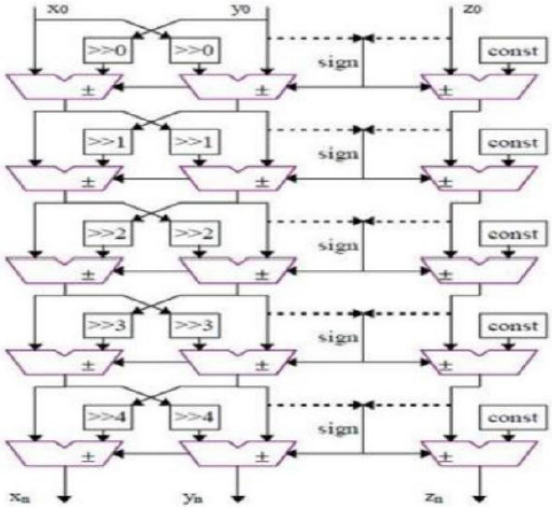


Fig. 1. Radix 2 CORDIC Architecture.

The various components required for the Radix-2 CORDIC processor in unfolded style for implementation are ROM which stores the angle values $\tan^{-1}(2^{-i})$ where i is varied from 0 to 16 for 16-bit processor. There are barrel shifters required for shifting of the intermediate values Xi and Yi. The barrel shifters carry out a right shift which can be implemented using multiplexers. For next iteration for X, Y and Z computation there are addition/subtraction unit [7].

For rotation mode Radix-2 CORDIC

$$x_{i+1}=K[x_i + d_i 2^{-i}\, y_i] \quad (4)$$

$$y_{i+1}=K[y_i + d_i 2^{-i}\, x_i] \quad (5)$$

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i}) \quad (6)$$

Where $\sigma_i = -1$ for $Z_i < 0$.

Else $\sigma_i = 1$ after n iterations we get,

$$X_n = A_n [X_0 \cos Z_0 - Y_0 \sin Z_0] \quad (7)$$

$$Y_n = A_n [Y_0 \cos Z_0 + X_0 \sin Z_0] \quad (8)$$

$$Z_n = 0 \quad (9)$$

$$A_n = \prod_{i=0}^{n} \sqrt{(1 + 2^{-2i}} \quad (10)$$

For Radix-2 CORDIC, processing gain is approximately K≈1.65.The major drawback of the conventional CORDIC algorithm is its comparatively high latency and low throughput due to the sequential nature of the iteration process with carry propagates addition and variable shifting in every iteration. To overcome these drawbacks, redundant CORDIC architecture has been implemented using redundant arithmetic. On the other hand, the carry propagate addition remained a bottleneck for additional throughput enhancement. Two most important methodologies have been employed in order to increase the speed of CORDIC implementation.

IV. REDUNDANT ARITHMETIC

Redundant Number Systems (RNS) offer an alternate form of computer arithmetic suited to numerically intensive applications. An important property of RNS is that it captures or prevents the carry propagation [8,9], creating parallel adders with constant delay, irrespective of the operand word-length. Thus low latency results are produced in an RNS format. Traditionally CORDIC implementations are based on ripple carry addition. These, however suffer from large internal carry propagation delays. Since the adder/ subtractor unit forms a major component of CORDIC architecture, their performance will determine the overall performance of the CORDIC processor. To enhance the performance of CORDIC processors redundant arithmetic has been proposed.

Table. 1: Redundant adder addition rules.

Redundant Adder Addition Rules

X□+Y□	**X□$_{-1}$+Y□$_{-1}$**	**Intermediate carry c □**	**Intermediate sum s □**
-2	Don't care	-1	0
-1	At least one is negative	-1	1
	None is negative	0	-1
0	Don't care	0	0
1	At least one is negative	0	1
	None is negative	1	-1
2	Don't care	1	0

18/04/2016 26

This arithmetic, due to its inherent carry-free property avoids the propagation of carry from the LSB to the MSB, resulting in faster operations. This section considers radix-2 hybrid and signed-digit additions and subtractions. The following table gives redundant adder rules [5].

V. RESULTS AND DISCUSSION

Simulation Results: Radix 2 Redundant CORDIC has been simulated using Xilinx 14.1.The following fig shows the simulation results for 16 bit cos and sine values for 30 ^{o} degree angles.

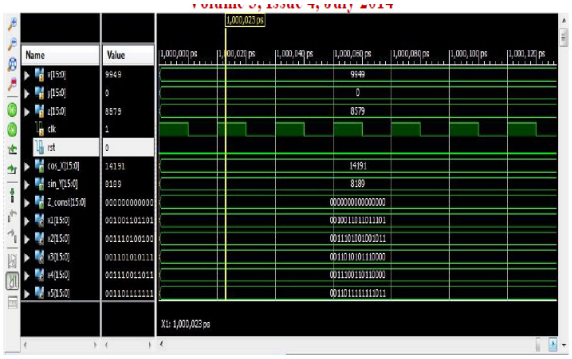
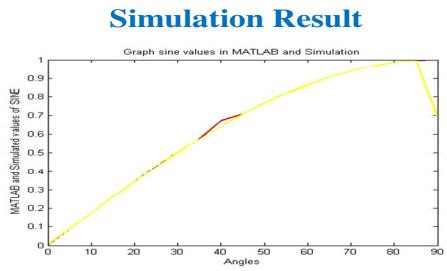


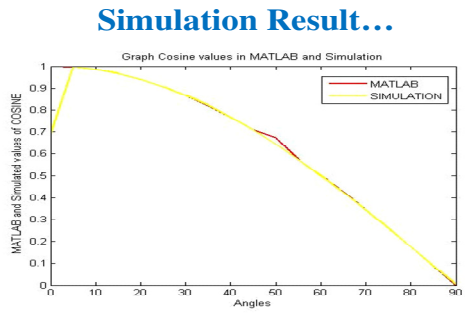
Fig. 2. Simulation waveform for 16 bit sine and cosine value.

The following graph shows the simulation results for % error vs angles.



18/04/2016 30

Fig. 3. Simulation Result for sine values.



18/04/2016 31

Fig. 4. Simulation Result for cosine values.

Synthesis Results: Redundant Radix 2 CORDIC Processor has been implemented using Vertex 5 FPGA device.

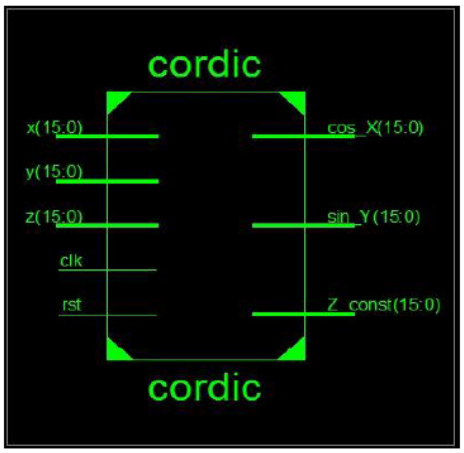


Fig . 5. Top level view.

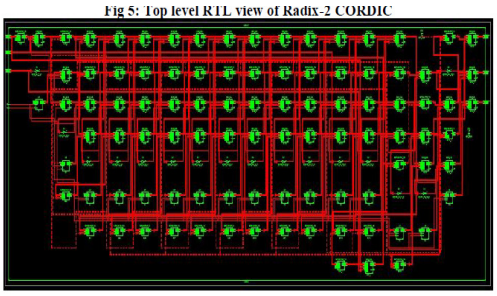


Fig. 6. RTL Schematic view.

Table 2: Synthesis Results.

Synthesis Results

- XILINX Vertex 5 FPGA, XC5Vlx30-3ff324

Parameters	Redundant	
	16	32
Max Comb Path Delay(ns)	13.4	16
Logic Delay(ns)	10.06	12.6
Route Delay(ns)	3.42	3.447
Max Operating Frequency (MHz)	74	62.4
Leakage Power(uw)	379.89	379.96
Dynamic Power(mw)	60.2	66.156

VI .CONCLUSION

Redundant Radix 2 CORDIC Processor has been designed and implemented on Xilinx14.2. Total delay is 13.4 ns for 16 bit and 16 ns for 32 bit .It operates at higher speed compared to that of non redundant CORDIC processor. The switching speed reduces due to the redundant architecture so power consumption is also less.

REFERENCES

[1]. Bhakthavatchalu, R., Sinith,M.S, Jismi,K., Nair, P.,"A Comparison of Pipelined Parallel and Iterative CORDIC Design on FPGA", Proceedings 2010 5th International Conference on Industrial and Information Systems, ICIIS 2010, July 29- Aug 01, 2010, India. pp. 239 – 243.

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

[3]. Deprettere, E., Dewilde, P.,Udo, R.," Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing", Proceedings ICASSP'84, 1984, pp. 41.A.6.1- 41.A.6.4

[4]. Andraka, R., "A survey of CORDIC algorithms for FPGA based computers", FPGA '98, in ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 191-200, 1998.

[5]. Erecegovac, M. D., Lang, T., "Digital Arithmetic", Elsevier, Amsterdam, the Netherlands, 2004.

[6]. Hu, Y. H.," Pipelined CORDIC architecture for the implementation of rotational based algorithm", in Proceedings of the International Symposium on VLSI Technology, Systems and Applications, pp. 259, May 1985.

[7]. De Lange, A. van der Hoeven, A. J., Deprettere, E. F., and Bu, J.," An optimal floating-point pipeline CMOS CORDIC Processor", IEEE ISCAS'88, pp. 2043-47, 1988.

[8]. Kamp, W., Bainbridge, A., "Multiply accumulate unit optimized for fast dot-product evaluation", Field-Programmable Technology, 2007, pp. 349–352, 12 Dec. 2007.

[9]. Parhi, K. K., "VLSI Digital Signal Processing Systems: Design and Implementation", Wiley, 1999.

[10]. Hwang, K., "Computer Arithmetic: Principles, Architectures and Design", Wiley, 1979.

[11]. Guyot, A., Herreros, Y., Muller, J., "JANUS, an on-line multiplier/divider for manipulating large numbers", in Proc. of 9th Symposium on Computer Arithmetic, pp. 106 – 111, 1989.

[12]. "ISE Simulator", Xilinx incorporation San Jose U.S.A, 2011.