Design And Implementation Of CORDIC Algorithm Using VHDL

Sitansu Ranjan Swain
College of Engineering Bhubaneswar
Bhubaneswar, Odisha

Santosh Kumar Patnaik
National Institute of Science and Technology
Brahmapur, Odisha

Abstract: At present time, many hardware efficient algorithms exist, but these are not well known due to the dominance of software systems over the many years. CORDIC is such an algorithm which is nothing but a set of shift algorithms used for computing a wide range of functions including certain trigonometric, hyperbolic, linear and logarithmic functions. While there are numerous articles covering various aspects of CORDIC algorithm, very few have been reviewed and even fewer methods have been implemented on FPGA. This paper attempts to investigate commonly used functions that may be accomplished using CORDIC architecture.

Keywords—CORDIC; FPGA; CPLD; DSP

I. INTRODUCTION

Many digital signal processing algorithms are implemented on microprocessors and microcontrollers. Though these processors are low-cost and provide extreme flexibility, they are not fast enough for truly demanding digital signal processing tasks. With advent of reconfigurable device like CPLD and FPGA which works at higher speed, provides the higher speed of hardware solutions for DSP algorithms. Among these DSP algorithms is an iterative solution for trigonometric, hyperbolic, logarithmic, exponential etc. functions that use only shift-add operation to perform [1]. The trigonometric functions are based on vector rotation, therefore the trigonometric algorithm is called CORDIC, which stands for COordinate Rotation DIgital Computer.

In this paper, the design and implementation of rotational CORDIC algorithm is presented. Section II explains about the CORDIC Algorithm. Implementation of the CORDIC Algorithm is given in Section III. Results and Discussion are given in Section IV. Finally, Section V concludes the paper.

II. CORDIC ALGORITHM

This algorithm was specially developed for real time digital computers where the majority of the computation involves trigonometric relationships. This CORDIC algorithm can be implemented using special arithmetic units such as shift registers, adders, subtractors and special interconnect.

This algorithm is derived from general rotation transform as given below [1], [2]:

$$X' = Xcos(\emptyset) - Ysin(\emptyset) \quad (1)$$
$$Y' = Ycos(\emptyset) + Xsin(\emptyset) \quad (2)$$

Taking $cos(\emptyset)$ out, "(1)" and "(2)" can be rewritten as:

$$X' = cos(\emptyset)[X - Y * tan(\emptyset)]) \quad (3)$$
$$Y' = cos(\emptyset)[Y + X * tan(\emptyset)]) \quad (4)$$

By replacing *tan(Ø)* by $\pm 2^{(-i)}$, the multiplications in "(3)" and "(4)" can be replaced simple by shift operations. The value of *tan (Ø)* is given in Table I.

Table I: Value of tan (Ø) for i = 0, 1, 2…..9

i	tan (Ø) = 2^{-i}	Ø
0	1	45.0^o
1	0.5	26.6^o
2	0.25	14.0^o
3	0.125	7.1^o
4	0.0626	3.6^o
5	0.03125	1.8^o
6	0.015625	0.9^o
7	0.0078125	0.4^o
8	0.00390625	0.2^o
9	0.001953125	0.1^o

With this modification, "(3)" and "(4)" can be rewritten as:

$$X_{i+1} = K_i[X_i - Y_i * D_i * 2^{(-i)}]) \quad (5)$$
$$Y_{i+1} = K_i[Y_i + X_i * D_i * 2^{(-i)}]) \quad (6)$$

where

i is the iteration count

$K_i = cos(tan^{-1}(2^{-i}))$

$D_i = \pm 1$

Removing the scaling factor, iteration is simple shift-add equation. The value of K_i approaches to *0.607* as the iteration count approaches infinity [1], [3]. The direction in which way the vector should rotate is expressed as:

$$Z_{i+1} = [Z_i - D_i * tan^{-1}(2^{-i})]) \quad (7)$$

where

$D_i = -1$ If $Z_i < 0$
$D_i = +1$ otherwise

To understand the iterative operation of this rotational CORDIC algorithm, consider an example: angle $Z_0 = 30.0^0$ [4]. Through iterative method the known angles i.e. $tan(2^{-i})$ are either added or subtracted in order to reach near the given angle. Once this is achieved, the sine and cosine of the given angle can be obtained. Fig.1 pictorially shows the iteration process.

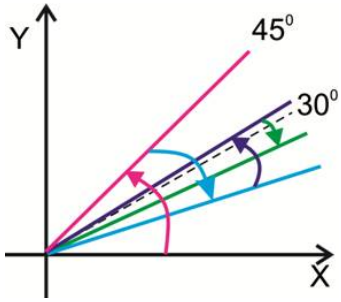


Fig.1: Iteration process of the CORDIC algorithm

The iteration process is as given below,
Let us start with the initial angle of 45.0^o

$45.0^o > 30.0^o$
$45.0^o - 26.6^o = 18.4^o$ ($<30.0^o$)
$18.4^o + 14.0^o = 32.4^o$ ($>30.0^o$)
$32.4^o - 7.1^o = 25.3^o$ ($<30.0^o$)
$25.3^o + 3.6^o = 28.9^o$ ($<30.0^o$)
$28.9^o + 1.8^o = 30.7^o$ ($>30.0^o$)
$30.7^o - 0.9^o = 29.8^o$ ($<30.0^o$)
$29.8^o + 0.4^o = 30.2^o$ ($>30.0^o$)
$30.2^o - 0.2^o = 30.0^o$ ($=30.0^o$)

After eight iterations, the angle which is obtained is 30.0^o as given below,

$45.0^o-26.6^o+14.0^o-7.1^o+3.6^o+1.8^o-0.9^o+0.4^o-0.2^o$
$=30.0^o$

Through this process, the sine and cosine of the 30.0^o can be obtained. This process is also applicable for any angle from 0^o to 90^o. The design steps used for CORDIC processor are given below:

Step1: Read the angle Z_0 whose sine and cosine functions are to be determined

Step2: Store the pre-calculated $(tan^{-1}(2^{-i}))$ values in an array

Step3: Assign $X_0 = 0.607252935$, $Y_0 = 0$

Step4: Calculate $Z_{i+1} = Z_i - D_i * tan^{-1}(2^{-i})$

Step5: Take $D_i = 1$ if $Z_i \geq 0$ and $D_i = -1$ if $Z_i < 0$

Step6: if $Z_i \geq 0$ then

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

else if $Z_i < 0$ then

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

Step7: Repeat steps 4, 5 and 6 till the angle (Z_i) approaches "0" to obtain the X_n $(=cos(Z_0))$ and Y_n $(=sin(Z_0))$

As is evident from the above design steps, the design of CORDIC processor needs basically three operations: addition, subtraction and shifting, therefore, require very less hardware to implement it in FPGA. This also makes the coding complexity comparatively simple [4], [5]. The flow chart representation of this CORDIC algorithm is given in Fig. 2. In this flow chart, it is clearly shown, how adder/subtractor and shifting operations are used to obtain the sine and cosine of a given angle.

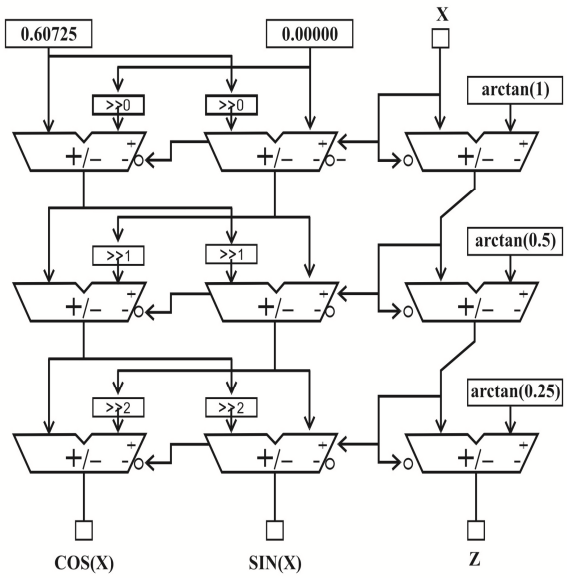


Fig.2: Flow chart for the CORDIC algorithm

III. Implementation of the CORDIC Algorithm

The CORDIC algorithm has been designed using VHDL. The angle (represented in degree) whose sine and cosine has to be calculated is first converted to radian. This radian value of the angle is then converted to 18 bit binary which is used as the input to the VHDL code. The obtained output is in binary form. To validate the result, it has to be converted to decimal form. This CORDIC algorithm has been successfully synthesized and implemented in Spartan -6 FPGA board. The summary of the design is given in Fig. 3.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	875	960	91%
Number of Slice Flip Flops	705	1920	36%
Number of 4 input LUTs	1666	1920	86%
Number of bonded IOBs	54	66	81%
Number of GCLKs	1	24	4%

Fig. 3: Device utilization summary

IV. Simulation and Results

The simulation has been carried-out for more than fifty angles. Here, simulation results for three angles (9^o, 42^o, 87^o) are given. The procedure followed for finding the sine and cosine of angel 9^o is as given below:

At first the angle 9^o is converted to radian which is equal to "*0.1570796326794897*". This radian angle is then converted to binary number "*0.00101000001101100101111*". This binary number is now used as the input to the VHDL code. The output is "*0.11111100110110010010 01*" for *X* and "*0.00101000000011000001 01*" for *Y*. After converting to decimal number, these binary number becomes "*0.987688340595137*" and "*0.156434465040230*" for X and Y respectively. The *X* and *Y* are nothing but the $cos(9^o)$ and $sin(9^o)$ respectively. The same procedures have been followed to validate the other angles (42^o and 87^o). Fig.4 shows the simulation results for these angles.

V. Conclusions

The CORDIC algorithms presented in this paper are well-known in the research and supercomputing circle. A common attempt is made to accomplish common functions using CORDIC architecture and explains how the algorithm works and explores the implementation specific to FPGA.

References

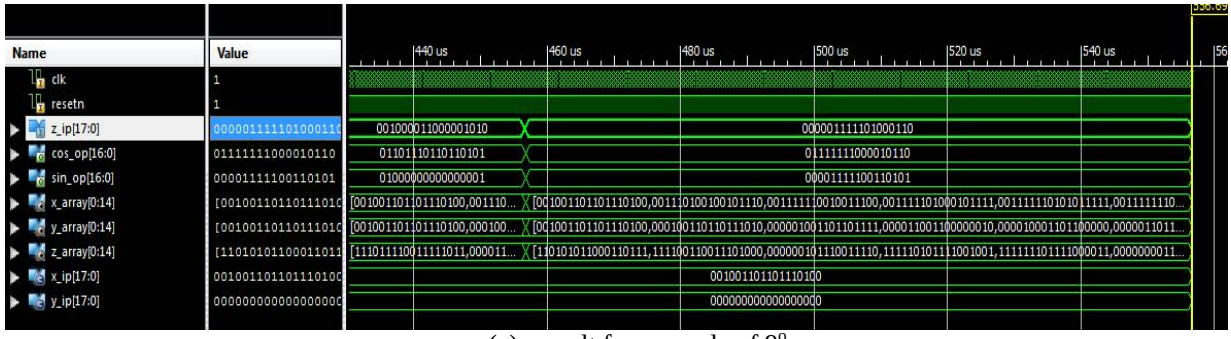
[1] Tanya V, et al, "FPGA Implementation of Sine and Cosine Generator using CORDIC Algorithm", International Conference on Military and Aerospace application of Electronics, 1998.

[2] Manoj Arora, et al, "FPGA Prototyping of Hardware Implementation of CORDIC Algorithm", International Journal of Scientific & Engineering Research, Volume 3, Issue 1, 2012.

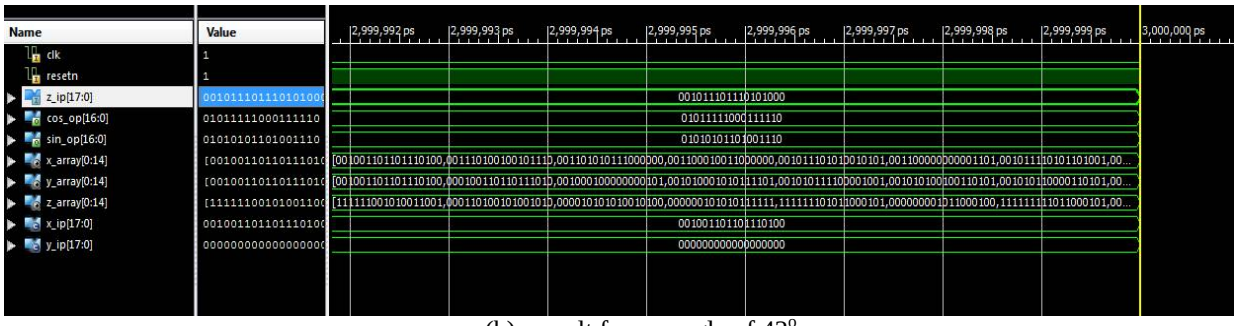
[3] J. Volder, "The CORDIC Trigonometric Computing Technique", IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330-334, 1959.

[4] R. Andraka, "A survey of CORDIC algorithm for FPGA based computers", International Symposium on Field Programmable Gate Arrays, no. 2, pp. 191-200, 1998.

[5] J. Walther, "A Unified Algorithm for Elementary Functions", Proceedings of Spring Joint Computer Conference, vol. 38, pp. 379-385, 1971.



(a) result for an angle of 9^o



(b) result for an angle of 42^o

(c) result for an angle of 87^o

Fig.4: Simulation results for angles of 9^o, 42^o and 87^o

Sitansu Ranjan Swain received his BE degree in Electronics & Telecommunication Engineering from Utkal University in 1991.He received his MTech (Microelectronics) from ITBHU, Varanasi,Banaras Hindu University in 2006. He is presently working as an Assistant Professor in College of Engineering, Bhubaneswar. His main research interest is in microelectronics.

Santosh Kumar Patnaik did his Ph.D. in Electronics & Communication Engineering at IIT Kharagpur and presently working as an Associate Professor at National Institute of Science and Technology, Odisha, India.