Implementation of trigonometric function using CORDIC algorithms

Cite as: AIP Conference Proceedings **1930**, 020040 (2018); https://doi.org/10.1063/1.5022934
Published Online: 02 February 2018

A. S. N. Mokhtar, M. I. Ayub, N. Ismail, and N. G. Nik Daud

Implementation of Trigonometric Function using CORDIC Algorithms

A. S. N. Mokhtar[a)], M. I. Ayub[b)], N. Ismail[c)] and N. G. Nik Daud[d)]

Department of Electrical and Electronic Engineering, Faculty of Engineering, National Defense University of Malaysia, Kuala Lumpur, Malaysia

[a)]Corresponding author: anis@upnm.edu.my
[b)]ayub@gmail.com
[c)]norlaili@upnm.edu.my
[d)]nikghazali@upnm.edu.my

Abstract. In 1959, Jack E. Volder presents a brand new formula to the real-time solution of the equation raised in navigation system. This new algorithm was the most beneficial replacement of analog navigation system by the digital. The CORDIC (Coordinate Rotation Digital Computer) algorithm are used for the rapid calculation associated with elementary operates like trigonometric function, multiplication, division and logarithm function, and also various conversions such as conversion of rectangular to polar coordinate including the conversion between binary coded information. In this current time CORDIC formula have many applications in the field of communication, signal processing, 3-D graphics, and others. This paper would be presents the trigonometric function implementation by using CORDIC algorithm in rotation mode for circular coordinate system. The CORDIC technique is used in order to generating the output angle between range 0o to 90o and error analysis is concern. The result showed that the average percentage error is about 0.042% at angles between ranges 00 to 900. But the average percentage error rose up to 45% at angle 90o and above. So, this method is very accurate at the 1st quadrant. The mirror properties method is used to find out an angle at 2nd, 3rd and 4th quadrant.

INTRODUCTION

The Coordinate Rotation Digital Computer (CORDIC) is an iterative method for calculation of rotation in 2 dimensional vectors, linear, circular and hyperbolic coordinate system using add and shift operation [1]. CORDIC firstly described in 1959 by Jack E. Volder for sophisticated solution to evaluate the trigonometric function. It was developed to replace the analog navigation computer on the B-58 planes bomber due to desire for high performance and accuracy [2]. In 1971, J. Walther extended CORDIC algorithm to hyperbolic functions and nowadays found in many applications such as digital signal processing, matrix computation, digital image processing, graphics, robotic and communication [3-6].

The most important concept of the actual algorithms founded on rotator of a two dimensional (x and y) vector with circular, hyperbolic and linear coordinate systems [7]. CORDIC become more useful as John Walther reveal a few parameters; it is desirable in order to perform as single algorithm for unified implementation of wide range of elementary transcendental functions related to the logarithms, exponentials and square root [8]. Cochranv also reveals that CORDIC technique is much better especially for scientific calculating applications. Some of the famous applications are digital modulation, direct digital frequency synthesis, inverse and direct kinematics computation for robot manipulation and 3-dimension rotation for animation and graphic [9]. Although CORDIC is not fastest technique to perform these operations, but it is attractive because of the simplicity of its hardware implementation. The same iterative method could be used for all application using basic shift-add operation of the form $a \pm b.2^{-i}$ [7].

International Conference on Engineering and Technology (IntCET 2017)
AIP Conf. Proc. 1930, 020040-1–020040-7; https://doi.org/10.1063/1.5022934
Published by AIP Publishing. 978-0-7354-1622-2/$30.00

CORDIC ALGORITHM

This section would be discussing about the CORDIC computation and its basic principle. CORDIC is known as a simple algorithm to compute a lot of functions such as trigonometric that can be derived or computed from functions using vector rotations. The advantage of CORDIC algorithm is that it provides an iterative method in order to performing vector rotations by coordinate angles by using add and shift operations.

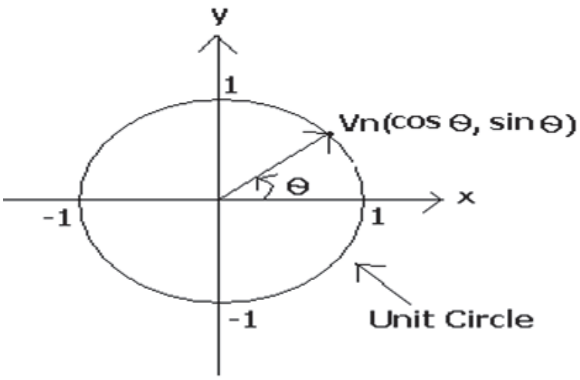

FIGURE 1. Unit cycle with vector, Vn

Basically, CORDIC algorithm comprises of two working modes which is rotation mode and vectoring mode individually. The "rotation" described by Volder, which involved rotates the input vector by a specified. The second is called "vectoring", which only involved rotation of the input vector on the x-axis while recording the angle desired during that rotation [10]. According to the Fig. 1, the coordinate on the unit circle which has a rotation of θ, has both cos θ and sine θ coordinates. This prove that if a point on the x-axis is rotated by an angle θ, the cosine and sine of that angle of rotation would be read directly for both x and y axis. This vector rotation could be achieved by rotating the coordinate on the circle system in series steps, which are much smaller than θ. These steps would be either in a clockwise (-θ) or anti-clockwise (+θ). As an example, if we desired to achieve a total rotation of 45°, first we are rotating the vector 30° in anti-clockwise, then rotate 20°-degree anti-clockwise direction and finally rotate the vector 5° clockwise.

TABLE 1. θ and tan θ values

Rotation, N	Θ	Tan Θ
0	45.000°	$1 = 2^{0}$
1	26.565°	$\frac{1}{2} = 2^{-1}$
2	14.036°	$\frac{1}{4} = 2^{-2}$
3	7.125°	$\frac{1}{8} = 2^{-3}$
4	3.576°	$\frac{1}{16} = 2^{-4}$
5	1.789°	$\frac{1}{32} = 2^{-5}$
6	0.895°	$\frac{1}{64} = 2^{-6}$
7	0.448°	$\frac{1}{128} = 2^{-7}$

The coordinates of a point in 2-dimension space is indicate as a vector. If coordinates of the points are (x, y), then the point would be indicate as $(x, y)'$ where the inverted comma presents a matrix transpose function. This rotation of a point in two dimensional spaces basically because of multiplication of the coordinates of that point by a rotation matrix,

$$\begin{bmatrix} cos\ \theta & -sin\ \theta \\ sin\ \theta & cos\ \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (1)$$

So the result is:

$$x' = x^{*}cos\theta - y^{*}sin\theta \quad (2)$$
$$y' = y^{*}cos\theta + x^{*}sin\theta \quad (3)$$

Then, both Equation (2) and (3) are divided with $\cos\theta$:

$$\frac{x'}{\cos\theta} = x - y\tan\theta \quad (4)$$
$$\frac{y'}{\cos\theta} = x + y\tan\theta \quad (5)$$

Based on to both equations above, θ are presents the rotation for each step. It is possible to separate the rotation into many steps, each of step is decreasing size such since tan θ is the power of two. As a result, tan θ can be implemented by the multiplication as a very efficient bit shift operation. Table 1 indicates the θ value along with tan θ [11-13]. Since the rotation may be either in a counter-clockwise direction or clockwise, it is clearly that these steps would be used to approximate an angle between -99.4 and +99.4 (45° + .. + 3.576° + 1.789° + 0.448°+ 26.565°+ 14.036°+ 7.125°+ 0.895°= 99.434). Furthermore, the decreasing size of steps presents that the approximation could be made arbitrarily accurate by increasing the number of rotation steps. Lookup table is used in order to keep the angle θ for every step. The table size increases with $\log_2$N where N indicates the number of step rotation. By referring to the Equation (4)-(5), tan θ is needed in order to effect the rotation of a point along the unit circle. By assuming the θ is between +90° and -90°, then the trigonometric identity yields,

$$\cos\theta = \cos(-\theta) \quad (6)$$

This shows that if our rotational steps are always the similar angles, it makes no difference if our rotation steps are clockwise our anti-clockwise direction. This would be gives us to substitute a constant in place of the cos θ in term of both Equation (4)-(5). The constant depends on the number of steps of the rotation to approximate the cosine and sine values:

$$\cos\theta = \cos(tan^{-1}2^{-i}) \quad (7)$$

For example, if 4 rotation steps are used to approximate the sine and cosine values, then the constant, A_4, is calculated as follows:

$$A_4 = \cos(tan^{-1}1) + \cos\left(tan^{-1}\frac{1}{2}\right) + \cos\left(tan^{-1}\frac{1}{4}\right) + \cos\left(tan^{-1}\frac{1}{8}\right) .... \quad (8)$$

So that, the iteration number can be represent as:

$$x_{i+1} = A_i[x_i - y_i . d_i . 2^{-i}] \quad (9)$$
$$y_{i+1} = A_i[y_i - x_i . d_i . 2^{-i}] \quad (10)$$

where

$$A_i = \cos(tan^{-1}2^{-i}) = 1/\sqrt{1 + 2^{-2i}}, d_i = \pm 1$$

The particular angle of a rotation can be distinctively defined by the sequence of the directions of the elementary rotations. To compute the angles desired, an additional adder-subtractor that accumulates the elementary angles for iterations is used:

$$Z_{i+1} = Z_i - d_i * (2^{-i}) \tag{11}$$

Thus, the CORDIC rotator's (using rotation mode) equations are:

$$x_{i+1} = x_i - y_i * d_i . 2^{-i}$$
$$y_{i+1} = y_i - x_i * d_i . 2^{-i}$$
$$Z_{i+1} = Z_i - d_i * tan^{-1}(2^{-i})$$

where

$$d_i = -1 \ if \ Z_i < 0, +1 \ otherwise$$

METHODOLOGY

The methodology of this project is shown in Fig. 2. It starts with understanding of the CORDIC algorithm then verifying it using calculation tool. After satisfied with the result, the Verilog of the algorithm is developed. Fixed point format 14.1 is used in the calculation for 16 bits of implementation. The output is obtaining from the ModelSim after considering all constrains of the circuits before the analysis is done.

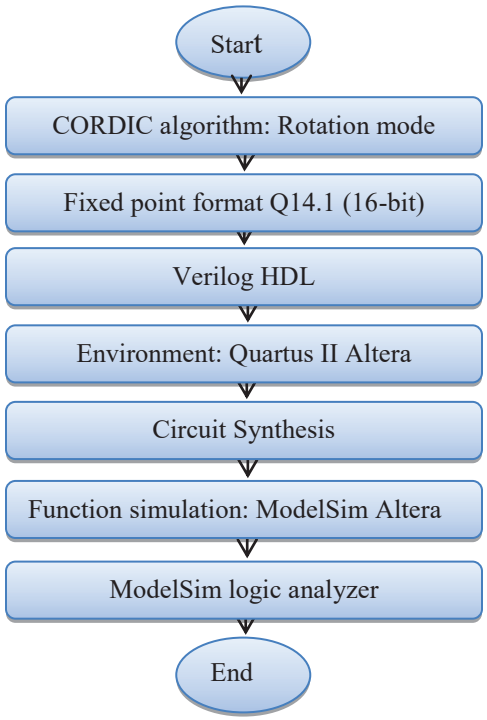

FIGURE 2. Flowchart of methodology

RESULTS AND DISCUSSION

In this project, CORDIC algorithm in rotation mode are present to calculate the sine and cosine value within phase between 0° to 90° instead of 0° to 360°. As a way to generate full-cycle sinusoidal for both cosine and sine wave, we

use the mirror properties of the sine and cosine wave (inverting and mirroring the 0° to 90° values for 2nd, 3rd and 4th cycle). First the algorithm has been tested by using Microsoft word Excel for all angles. The initial value of x in Fig. 1 is 0.607 as a constant and consideration y initial is zero since the starting point of rotation at y equal to 0. Basically the initial value of z is our angle desired that have to covert to radian before starting the calculation.

TABLE 2. Error analysis for range range between 0° to 90°

Angles	No. of Iteration	Percentage Error, (%)	
		Cos θ	Sin θ
10	14	0.041	0.029
20	15	0.041	0.046
30	15	0.042	0.039
40	15	0.041	0.042
50	15	0.042	0.041
60	15	0.039	0.042
70	13	0.038	0.042
80	14	0.046	0.042
90	15	infinity	0.042
95	15	0.032	0.042
100	15	1.217	0.005
105	15	33.724	1.949
110	15	49.847	4.795
120	15	65.693	13.710
180	15	82.847	infinity
360	15	117.153	infinity

Table 2 in second column shows the result for all quadrants that calculated by using the CORDIC iteration method in Excel Microsoft Word Office. The minimum rotation number is 13 times at 70° while the maximum is 15 times rotation. This rotation number is depending on the angle desired as the system would rotate to reach nearest angle. System would be stop at angles desired otherwise would be keep rotating. Basically, the lowest iteration number is the best system because of high speed result. But in this case, consideration of the average iteration number is 14 to reach accurate and precise output. Figure 3 represents the iteration number analysis for all quadrant angles.

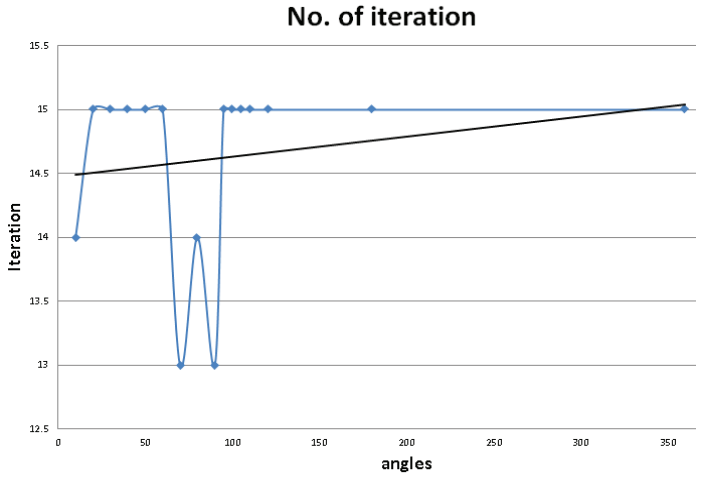

FIGURE 3. Number of iteration for angle range from 0° to 360°

By referring to Fig. 2, the percentage error comparisons are made for both sine and cosine wave. The result shows that the average percentage error for 1st quadrants is about 0.044% which is very accurate outcome. For angle ranges up to 90° until 100° is still available for application with average percentage error is 0.648% for both sine and

cosine wave. But the percentage error increase rapidly staring from 105° to 360°. We can conclude that this method is only suitable for angle range starting from 0 to 90°. The rest can be calculating by using the mirror properties of the sine and cosine wave as mention before. Figure 4 represents the error analysis for both sine and cosine wave by using scatter chart.

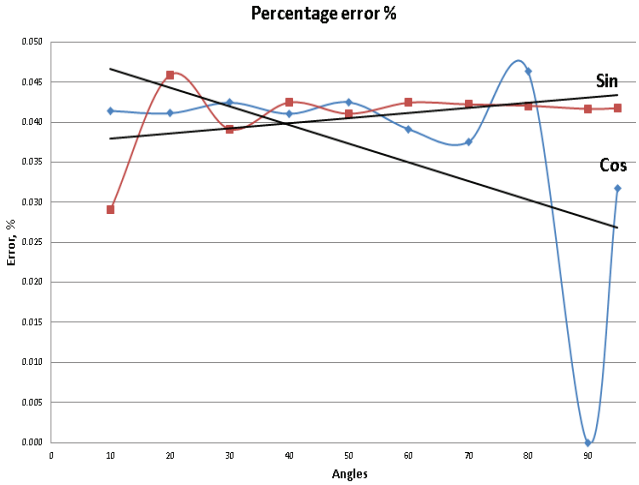

FIGURE 4. Percentage error for both sine and cosine wave within angle range from 0° to 90°

CONCLUSION

The implementation of trigonometric function using CORDIC algorithms was being presents. It includes the CORDIC algorithm implementation in order to generate both cosine and sine wave for error analysis. The increase the iteration number, the stability of percentage error for both cosine and sine wave are increase. As we decrease the number of iteration, the speed of the system would be increase but the stability of error is decrease. The results have shown which this might be pretty much awakened to the fact within long term applications.

ACKNOWLEDGMENTS

This research work is supported by the UPNM Short Grant (UPNM/2016/GPJP/5/TK/4).

REFERENCES

1. J. E. Volder, IRE Transactions on Electronic Computers **3**, 330-334 (1959).
2. J. Volder, Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology **25**, 101-105 (2000).
3. S. Sathyanarayana, Signal Processing **1**, 343-346 (2007).
4. A. S. N. Mokhtar, M. B. I. Reaz, K. Chellappan and M. A. M. Ali, "Scaling free CORDIC algorithm implementation of sine and cosine function," in World Congress on Engineering (2013), pp. 926-929.
5. L. Vachhani, IEEE Transactions on Industrial Electronics **56**, 4915-4929 (2009).
6. V. Sharma and M. G. Bansal, "FPGA implementation of EEAS CORDIC based sine and cosine generator," Master thesis, Thapar University, 2009.
7. P. K. Meher, J. Valls, J. Tso-Bing, K. Sridharan and K. Maharatna, IEEE Transactions on Circuits and Systems I: Regular Papers **56**, 1893-1907 (2009).
8. J. S. Walther, "A unified algorithm for elementary functions," in ACM Spring Joint Computer Conference (1971), pp. 379-386
9. D. S. Cochran, Hewlett-Packard Journal **June**, 10-11 (1972).

10. T. Vladimirova and H. Tiggeler, “FPGA implementation of sine and cosine generators using the CORDIC algorithm,” in Military and Aerospace Programmable Logic Device International Conference (1999), pp. 26-28.
11. S. Hsiao and J. Chen, Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology **278**, 267-278 (1998).
12. N. G. Nik Daud, F. R. Hashim, M. Mustapha and M. S. Badruddin, “Hybrid modified booth encoded algorithm-carry save adder fast multiplier,” in Information and Communication Technology for The Muslim World (ICT4M), 2014, The 5th International Conference on (2014), pp. 1-6.
13. A. S. N. Mokhtar, S. A. Karim, S. P. Chew, S. M. F. S. M. Dardin, L. S. Supian, F. R. Hashim, “FPGA Implementation of Cordic Algorithm in Digital Modulation,” Journal of Fundamental and Applied Sciences, **9(3S)**, (2017).