INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

DESIGN AND IMPLENTATION OF CORDIC HARDWARE FOR SINE AND COSINE GENERATION USING VHDL

NEHA S. SAKHALKAR, SONIYA KUWELKAR

Goa College of Engineering, Farmagudi, Ponda, Goa

Accepted Date: 05/03/2015; Published Date: 01/05/2015

Abstract: In recent research works being carried out, there are numerous applications where sine and cosine waves are used. Also, there are a wide variety of ways to generate digital sine and cosine waves, but it requires excessive memory usage when good quantization level is needed. On the other hand, CORDIC (Coordinate Rotation digital Computer) Algorithm offers an excellent alternative which has greater flexibility and quantization accuracy as a function of word length. It is a Multiplier-less approach which uses only shift and add operations to estimate the basic elementary function thus, is more economical than DSP algorithms in terms of area and power consumption. The CORDIC algorithm has become widely researched topic in various technological fields such as digital signal processing (DSP), biomedical signal processing, robotics, communication systems, image processing etc. This paper describes the design and simulation of 16-bit Pipelined hardware for calculating Sine and Cosine of an Angle using CORDIC algorithm. The design is implemented using VHDL and simulation is carried out using Model Sim 5.7g and Xilinx ISE 10.1. The system can be implemented using Virtex5 FPGA. This design aims at reduced hardware complexity and improved throughput due to the use of Pipelined Architecture of CORDIC. With the use of CORDIC algorithm, accuracy up to 3 decimal digits on 16-bit input is achieved.

Keywords: CORDIC, Pipeline, FPGA, VHDL

Corresponding Author: MS. NEHA S. SAKHALKAR

PAPER-QR CODE

Access Online On:

www.ijpret.com

How to Cite This Article:

Neha S. Sakhalkar, IJPRET, 2015; Volume 3 (9): 140-151

INTRODUCTION

Calculation of sine and cosine of given angle is an essential requirement in many areas of real life. In medical science, medical equipment that measures regular cyclical body functions like heartbeat, breathing etc. use sine and cosine waves. In signal processing, digital audio and high definition videos are based on sums of sine and cosine. In geology, earthquakes are modelled with wave equations which are solved using sums of sine and cosine. Similarly, radio communication used in electronics is based on use of combinations of sine and cosine waves. In geology, earthquakes are modelled with wave equations which are solved using sums of sine and cosine. Similarly, radio communication used in electronics is based on use of combinations of sine and cosine waves. Also, today most of the DSP applications are based on real time multimedia processing. Digital representation of multimedia data can be handled in the same way as text; however the processing rate has to be much faster . On account of this real time throughput constraint, conventional processors are not suitable for modern day DSP systems. Some hardware efficient algorithms are, therefore required for these high speed applications. These algorithms need to be implemented and optimized in hardware so as to enable them to handle real time data while maintaining an optimum trade-off between different performance parameters (area, speed and power). CORDIC is one such algorithm.

CORDIC (Coordinate Rotation Digital Computer) is a hardware efficient shift-and-add algorithm that can be used to calculate various arithmetic functions. The main concept of this algorithm is based on the very simple and long lasting fundamentals of two-dimensional geometry. The first description for iterative approach of this algorithm is firstly provided by **Jack E. Volder** in 1959[1]. CORDIC algorithm provides an efficient way of rotating the vectors in a plane by simple shift add operation to estimate the basic elementary functions like trigonometric operations, multiplication, division and some other operations like logarithmic functions, square roots and exponential functions. In 1971,CORDIC based computing received attention, when **John Walther** [2] showed that, by varying a few simple parameters, it could be used as a single algorithm for implementation of most of the mathematical functions.

Over the last 54 years, the architecture of CORDIC has seen multiple changes and multiple variants of the initially proposed algorithm have emerged [6]. Architectures such as low-latency pipelined CORDIC, mixed-scaling-rotation (MSR) CORDIC [10], scaling-free CORDIC have seen much usage in modern digital systems [9] – [15].

Large numbers of architectures have been proposed in the literature for CORDIC algorithm [6], which vary from bit-serial implementations to word parallel pipelined architectures. The choice

depends on the requirements for computing throughput and constraints that hold for area usage, latency and power dissipation.

Many attempts were made to calculate sine and cosine of an angle with software and hardware approach. A 32-bit CORDIC based Iterative Architecture to calculate sine and cosine of an angle was presented [3].The design was implemented on various FPGAs and the results were compared. An 8-bit design was proposed and verified for its working [4].A 16-bit hardware was implemented using VHDL, the results were verified using simulation and analysis of the resources used by the design was done[6].An 8-bit design was implemented using iterative architecture and its performance was verified with theoretical values but the design is a bit slow since it uses an iterative architecture[16]

This paper presents hardware design for calculating sine and cosine value of given angle using CORDIC algorithm with limited hardware usage to achieve better speed. The main contributions of this paper are: (i) Design of CORDIC hardware (ii) simulation and results, and (iii) conclusion and future work

- CORDIC ALGORITHM

The CORDIC algorithm is an iterative technique based on the rotation of a vector which allows many transcendental and trigonometric functions to be calculated. The key aspect of this method is that it is achieved using only shifts, additions/subtractions and table look-ups which map well into hardware and are ideal for FPGA implementation.

- *Algorithm fundamentals*

Vector rotation is the first step to obtain the trigonometric functions. It can also be used for polar to rectangular and vice-versa conversions, for vector magnitude, and as a building block in certain transforms such as the Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT). The algorithm is derived from givens rotation as follows:

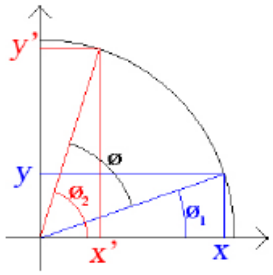

Fig.1. CORDIC Algorithm

$x' = x\cos\phi - y\sin\phi$

$$y' = y\cos\phi + x\sin\phi \quad (1)$$

For CORDIC,the final angle the angle whose sine or cosine we want to calculate and initial angle is set to a convenient value such as 0. Rather than rotating from to in one full sweep, we move in steps with careful choice of step values. Rearranging (1) gives us:

$x' = \cos\phi.[x - y\tan\phi]$

$$y' = \cos\phi.[y + x\tan\phi] \quad (2)$$

Restricting the rotation angles such that tan= $\pm 2^{-i}$, transforms the multiplication by the tangent term to a simple shift operation. Arbitrary angles of rotation are obtained by successively performing smaller elementary rotations. If i, the decision at each iteration, is which direction to rotate rather than whether to rotate or not, then cos (δi) is constant as $\cos(\delta i) = \cos(-\delta i)$. Then the iterative rotation can be expressed as:

$x_{i+1} = K_i.[x_i - y_i . d_i . 2^{-i}]$

$$y_{i+1} = K_i.[y_i + x_i . d_i . 2^{-i}] \quad (3)$$

where, $K_i = \cos(\tan^{-1} 2^{-i})$, $d_i = \pm 1$

Removing the scale constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the Ki's can be applied elsewhere in the system or treated as part of a system processing gain. That product approaches 0.6073 as the number of iterations reaches infinity. Therefore, the rotation algorithm has a gain, An ≈ 1.65. The exact gain depends on the number of iterations, and follows the following equation:

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

The angle of a composite rotation is realized by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The set of all possible decision vectors is an angular measurement system based on binary arctangents. Conversions between this angular system and any other can easily be accomplished using a LUT. A better conversion method uses an additional adder-subtract or that accumulates the elementary rotation angles post iteration. The angle accumulator adds a third difference equation to the CORDIC algorithm:

$$Z_{i+1} = z_i - d_i . \tan^{-1}(2^{-i}) \quad (4)$$

As discussed above, when the angle is in the arctangent base, this extra element is not needed. The CORDIC rotator is normally operated in one of two modes, i.e., the Rotation mode and the Vectoring mode.

- *Rotation Mode*

The first mode of operation, called rotation by Volder [1], rotates the input vector by a specified angle (given as an argument). Here, the angle accumulator is initialized with the desired rotation angle. The rotation decision based on the sign of the residual angle is made to diminish the magnitude of the residual angle in the angle accumulator. If the input angle is already expressed in the binary arctangent base, the angle accumulator is not needed [1]. The equations for this are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \qquad (5)$$

Where,

$d_i = -1$ if $z_i < 0$

$=+1$, otherwise

Then,

$$x_n = A_n[x_0.\cos z_0 - y_0.\sin z_0]$$

$$y_n = A_n[y_0.\cos z_0 + x_0.\sin z_0] \qquad (6)$$

- *Vectoring Mode*

In the vectoring mode, the CORDIC rotator rotates the input vector through whatever angle is necessary to align the result vector with the x axis. The result of the vectoring operation is a rotation angle and the scaled magnitude i.e. the x component of the original vector. The vectoring function works by seeking to minimize the y component of the residual vector at each rotation. The sign of the residual y component is used to determine which direction to rotate next. When initialized with zero, accumulator contains the traversed angle at the end of the iterations. The equations in this mode are:

$x_{i+1} = K_i.[x_i – y_i . d_i . 2^{-i}]$

$y_{i+1} = K_i.[y_i + x_i . d_i . 2^{-i}]$

$$z_{i+1} = z_i – d_i . \tan^{-1}(2^{-i}) \quad (7)$$

where,

$d_i = +1$ if $y_i<0$

=-1,otherwise

Then,

$x_n = A_n (x_0^2 + y_0^2)^{-1/2}$

$y_n = 0$

$$z_n = z_0 + \tan^{-1}(y_0/x_0) \quad (8)$$

The CORDIC rotation and vectoring algorithms as stated are limited to rotation angles between $-\pi/2$ and $\pi/2$. For composite rotation angles larger than $\pi/2$, an additional rotation is required. Volder describes an initial rotation of $\pm \pi/2$. This gives the correction iteration:

$x' = -d.y$

$y' = d.x$

$z' = z + d.(\pi/2)$

where,

$d_i = +1$ if $y_i<0$

=-1, otherwise (9)

There is no growth for this initial rotation. Alternatively, an initial rotation of either π or 0 can be made, avoiding the reassignment of the x and y components to the rotator elements. Again, there is no growth due to the initial rotation

$x' = d.x$

$y' = d.y$

$z' = z$, if d=1

$= z - \pi$,if d=-1

Where,

$d_i = -1$ if $x<0$

$= +1$,otherwise (10)

- *Evaluation of Sine and Cosine*

In rotational mode the sine and cosine of the input angle can be computed simultaneously. Setting the y component of the input vector to zero reduces the rotation mode result to:

$x_n = A_n.x_0.\cos z_0$

$y_n = A_n.x_0.\sin z_0$ (11)

If x0 is equal to 1/An, the rotation produces the unscaled sine and cosine of the angle argument, z0. Very often, the sine and cosine values modulate a magnitude value. Using other techniques (e.g., a LUT) requires a pair of multipliers to obtain the required modulation. The algorithm performs the multiply as part of the rotation operation, and therefore eliminates the need for a pair of explicit multipliers. The output of the CORDIC rotator is scaled by the rotator gain. If the gain is not acceptable, a single multiply by the reciprocal of the gain constant placed before the CORDIC rotator will yield unscaled results

- IMPLEMENTATION AND RESULTS

A. *Design Methodology*

In this paper, the FPGA implementation for calculating the sine and cosine values of given angle using CORDIC algorithm is presented. The module was implemented by using Xilinx ISE Design Suite 9.2i and VHDL. The ModelSim simulator was used to verify the functionalities of the module and this module was described in VHDL and synthesized using the Xilinx ISE Design Suite. Fig.2 shows the top-level RTL schematic of the implemented design and Fig.3 shows Internal Diagram of RTL Schematic.

The implemented design has following specifications:

- 16 – Bit Input/Output in which 13 LSBs are used for storing fractional part and 3 MSBs are used to store integer part including sign

- Pipelined architecture is to achieve a trade-off between area and speed.Also,this architecture avoids ROM requirement since accumulated angle values can be directly hardwired

- 7-stage pipeline is used due to which 2 iterations are calculated in every clock cycle

- Final output will be available after 7 clock cycles

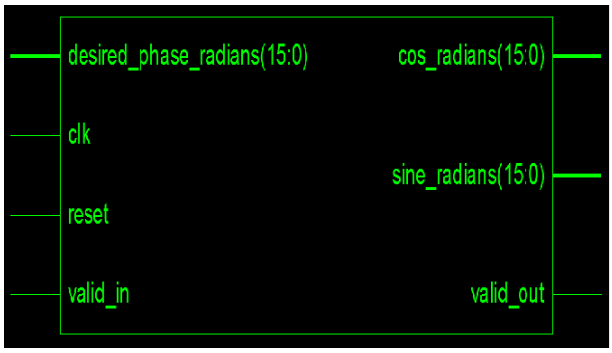

Fig.2.Top-level RTL schematic

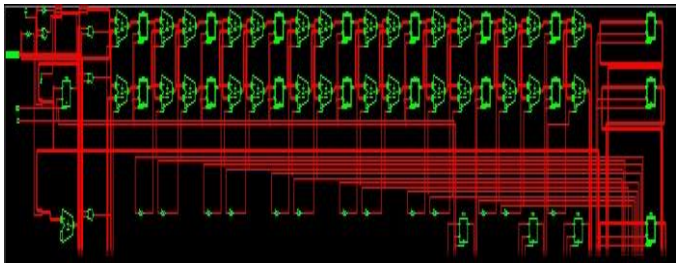

Fig.3. Internal Diagram of RTL Schematic

B. Simulation Results

The generated core has been simulated for sine and cosine functions by operating it in the rotation mode. Figure 4,5,6 show the simulated sine and cosine values of certain angles calculated using 16-bit Parallel CORDIC.

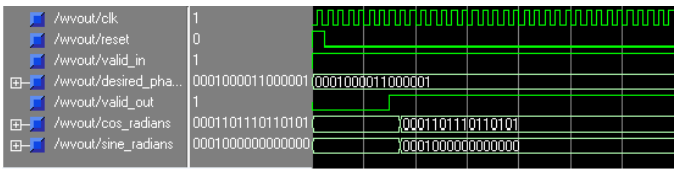

Fig.4.Simulation result for Input = 30^0

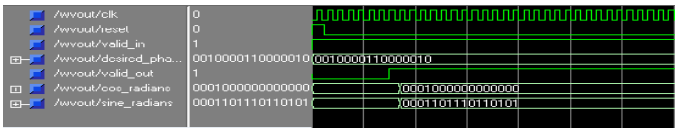

Fig.5.Simulation Result for Input = 60^0

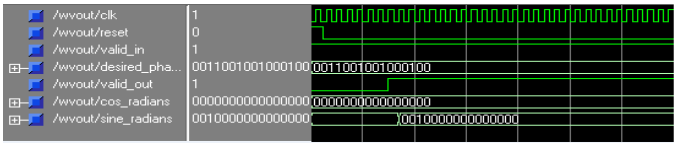

Fig.6.Simulation Result for Input = 90^0

C. Verification and Performance Analysis of Design

Above simulation results can be efficiently interpreted using Observation Table given in Table 1. Also, Device Utilization Summary of the implemented design is shown in Table 2. Both the observed results and device utilization summary can be used to analyze the results effectively. As shown in Table.1, accuracy of up to 0.0001 is achieved using the proposed design above when calculating sine and cosine. The accuracy can be further improved by increasing buffer size for input and output.

Table 1. Observation Table

	Input ɵ	30^0	60^0	90^0
Cosɵ	Observed	1BB5	1000	0000
	Calculated	0.8659	0.5	0
	Theoretical	0.866	0.5	0
	Deviation of output	0.0001	0	0
Sinɵ	Observed	1000	1BB5	2000
	Calculated	0.5	0.8659	1
	Theoretical	0.5	0.866	1
	Deviation of output	0	0.0001	0

Table. 2. Device Utilization Summary

	Used	Available	Percentage Usage
No.of slice registers	446	19200	2%
No.of slice LUTs	882	19200	4%
No.of occupied slices	243	4800	5%
No.of bonded IOBs	52	400	13%

V FUTURE SCOPE

The design can be enhanced to work with floating point notation instead of fixed point notation to represent fractional numbers. The accuracy of results given by proposed design can further be increased using larger buffer size to store inputs and outputs. A trade-off between no. of pipeline stages and no. of bits can be established to achieve higher accuracy and speed. This design can also be further improved for DSP or communication applications which use Sine - Cosine values as their inputs

VI CONCLUSION

Hardware for calculating the Sine and Cosine of an angle based on CORDIC algorithm is successfully designed and simulated on Model Sim simulator using the VHDL language. The design is more efficient and consumes less resources and is less time intensive. As targeted, the simulation results and theoretical values are consistent. The proposed design can generate accurate results up to 3 decimal places with 16-bit inputs and outputs.

REFERENCES

1. J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Trans. Electron. Computers, vol. EC-8, pp. 330–334, Sept. 1959.

2. J. S. Walther, “A unified algorithm for elementary functions,” in Proc. 38th Spring Joint Computer Conf., Atlantic City, NJ, 1971, pp. 379–385.

3. Ashutosh Gupta, Manoj Duhan, Soloman Raju Kota,” HDL Implementation of Sine—Cosine Function Using CORDIC Algorithm in 32-Bit Floating Point Format”, The ICFI University Press,2009

4. Aman Chadha, Divya Jyoti and M. G. Bhatia," Design and Simulation of an 8-bit Dedicated Processor for calculating the Sine and Cosine of an Angle using the CORDIC Algorithm" IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)",2011

5. Srinivasa Murthy H N, Roopa M F,"FPGA Implementation of Sine and Cosine Generators using CORDIC Algorithm", International Journal of Innovative Technology and Exploring Engineering", November 2012

6. Pramod K. Meher, Javier Valls, Tso-Bing Juang, K. Sridharan, Koushik Maharatna,"50 Years of CORDIC: Algorithms, Architectures and Applications", IEEE transactions on Circuits and Systems – I , VOL.56, NO. 9, September 2009

7. Sharaf El-Din El-Nahas, Ammar Mottie Al Hosainy, Magdy M. Saeb, "Design and Implementation of Cosine Transforms Employing a CORDIC Processor" 24th National Radio Science Conference (NRSC 2007), March 2007.

8. B. Vennela and K. Sanath Kumar,"Design of NCO by using CORDIC Algorithm inASIC-FPGA Technology", Advances in Electronic and Electric Engineering, Volume 3, Number 9 2013

9. J. Villalba, J. A. Hidalgo, E. L. Zapata, E. Antelo, and J. D. Bruguera, "CORDIC architectures with parallel compensation of scale factor," Proc. Application Specific Array Processors Conf., pp. 258–269, Jul. 1995.

10. Zhi-Xiu Lin and An-Yeu Wu, "Mixed-Scaling_Rotation CORDIC (MSR-CORDIC) Algorithm and Architecture for Scaling-Free High-Performance Rotational Operations," Proc. Acoustics, Speech, and Signal Processing Conf., vol. 2, pp. 653–656, Apr. 2003.

11. K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified Virtually Scaling-Free adaptive CORDIC Rotator Algorithm and Architecture," IEEE Trans. Circuits Syst. Video Tech., vol. 5, no. 11, pp. 1463–1474, Nov. 2005.

12. S. Aggarwal, P. K. Meher, and K. Khare, "Area-Time Efficient Scaling-Free CORDIC Using Generalized Micro-Rotation Selection," IEEE Trans. VLSI Syst., vol. 20, no. 8, pp. 1542–1546, Aug. 2012.

13. F. J. Jaime, M. A. Sanchez, J. Hormigo, J. Villalba, and E. L. Zapata, "Enhanced Scaling-Free CORDIC," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 7, pp. 1654–1662, Jul. 2010.

14. M. G. Buddika Sumanasena, "A Scale Factor Correction Scheme for the CORDIC Algorithm," IEEE Trans. Comput., vol. 57, no. 8, pp. 1148–1152, Aug. 2008.

15. Elisardo Antelo, Julio Villalba and Emilio L. Zapata, "Low-Latency Pipelined 2D and 3D CORDIC Processors," IEEE Trans. Computers, vol. 57, no. 3, pp. 404-417, 2008.

16. Kavya Sharat, Sagar D M, Dr, B.V. Uma," Calculation of Sine and Cosine of an Angle using the CORDIC Algorithm", International Journal of Innovative Technology and Research (IJITR),Vol 02, Issue 02, March 2014.