FPGA Implementation of Sine and Cosine Generators using CORDIC Algorithm

Srinivasa Murthy H N, Roopa M

Abstract*— The current research in the design of high speed VLSI architectures for real-time digital signal processing (DSP) algorithms has been directed by the advances in the VLSI technology, which have provided the designers with significant impetus for porting algorithm into architecture. Many of the algorithms used in DSP and matrix arithmetic require elementary functions such as trigonometric, inverse trigonometric, logarithm, exponential, multiplication, and division functions and one such algorithm is CORDIC. Often trigonometric functions are used in embedded applications. Examples of this include motion control, filtering and waveform synthesis. For waveforms with few output points per cycle (for example one output point per degree) a lookup table will often suffice, and indeed this method is optimal in that it offers a reasonable compromise between speed and the need to use the microcontroller's memory efficiently. The CORDIC computing technique—a highly efficient method to compute elementary functions and this paper presents how to calculate sine and cosine values of the given angle using CORDIC algorithm. Summary of CORDIC synthesis results based on Xilinx FPGAs is given. The system simulation was carried out using ModelSim and Xilinx ISE Design Suite 9.2i. The system can be implemented using Spartan3 XC3S50 with Xilinx ISE 9.2i and VHDL.

***Index Terms*— CORDIC, Sine, Cosine, VHDL, FPGA.**

I. INTRODUCTION

The CORDIC algorithm is a well-known and widely studied iterative technique for calculating many basic arithmetic operations and elementary functions. CORDIC works by rotating the coordinate system through constant angles until the angleis reduces to zero. These functions frequently used in applications expressed in terms of basic plane rotations. Today's most of processors are designed using this iterative technique because of its hardware-efficient solutions and this is a iterative solutions for trigonometric and other transcendental functions that use only shifts and adds to perform. Its current application are in the field of digital signal processing, image processing, filtering, matrix algebra, etc.

The trigonometric functions are based on vector rotations while other functions such as square root are implemented using an incremental expression of the desired function. The trigonometric algorithm is called CORDIC, An acronym for Co-ordinate Rotational Digital Computer. The CORDIC algorithm generally produce one additional bit of accuracy for each iteration[5].

CORDIC is an entire-transfer computer and contains a special arithmetic unit consisting of three shift registers, three adder-sub tractors, and special interconnections, the CORDIC arithmetic unit can be used to solve either set of the following equations:

$$Y' = K(Y\cos\lambda + X\sin\lambda)$$
$$X' = K(X\cos\lambda + Y\sin\lambda) \quad (1)$$

Where, K is a constant. The trigonometric CORDIC algorithms were originally developed as a digital solution for real-time navigation problems. The original work is credited to Jack E. Volder. Extensions to the CORDIC theory based on work by John Walther and others provide solutions to a broader class of functions. The CORDIC algorithm has found its way into diverse applications including the 8087 math coprocessor, the HP-35 calculator, radar signal processors and robotics. CORDIC rotation has also been proposed for computing Discrete Fourier, Discrete Cosine, Discrete Hartley and Chirp Z-transforms, filtering, Singular Value Decomposition, and solving linear systems[5].

By making slight adjustments to the initial conditions and the LUT values, it can be used to efficiently implement trigonometric, hyperbolic, exponential functions, coordinate transformations etc. using the same hardware. Since it uses only shift-add arithmetic, the VLSI implementation of such an algorithm is easily achievable.

II. CORDIC ALGORITHM

The CORDIC algorithm is an iterative technique and consists of two modes of operation called rotation mode and vectoring mode. In the rotation mode, the co-ordinate components of a vector and an angle of rotation are given and the co-ordinate components of the original vector, after rotation through the given angle are computed. In the vectoring mode, the co-ordinate components of a vector are given and the magnitude and angular argument of the original vector are computed. All of the trigonometric functions can be computed or derived from functions using vector rotation, as will be discussed in the following sections.

A.Mathematical Basis of the algorithm

Vector rotation is the first step to obtain the trigonometric functions. It can also be used for polar to rectangular and rectangular to polar conversions, for vector magnitude ,and as a building block in certain transforms such as the DFT and DCT. The aim of the Algorithm is to compute the Sine and Cosine of a given angle, which we will call θ (Theta).

Suppose that we have a point on a unit circle, which may be illustrated as follows[7]:

Manuscript received on November, 2012.

Srinivasa Murthy H N, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bangalor, India.
Prof. Roopa M, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bangalore, India.

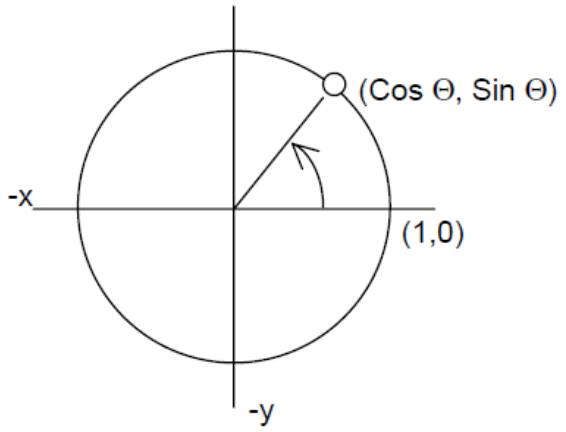

Figure.1.A point on the unit circle rotated by an angle Θ

The point on the unit circle that has a rotation of Θ has co-ordinates of (CosΘ, SinΘ). This implies that if a point on the x-axis is rotated by an angle Θ then the Sine and Cosine of the angle of rotation may be read directly off the x and y axes. The rotation may be achieved by rotating the point on the unit circle in a series of steps, which are smaller than Θ. In addition these steps may be either in an anti-clockwise direction (increase in Θ) or in a clockwise direction (decrease in Θ)[7].

The co-ordinates of a point in a two dimensional space may be represented as a vector. If the coordinates of the point are (x, y) then the point may be equally well represented as (x, y)' where the inverted comma indicates a matrix transpose function. The rotation of a point in two dimensional space may be effected by multiplying the co-ordinates of that point by a rotation matrix. Thus:

$$\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x \\ y\end{bmatrix}=\begin{bmatrix}x' \\ y'\end{bmatrix} \quad (2)$$

Where (x', y') are the co-ordinates of (x,y) rotated by an angle of Θ.This matrix operation may be expressed as follows:

$$x' = x\cos\theta - y\sin\theta \quad (3)$$
$$y' = y\cos\theta + x\sin\theta$$

For CORDIC, the final angle θ_n the angle whose sine or cosine we want to calculate and initial angle θ_1 is set to a convenient value such as 0. Rather than rotating from θ_1 to θ_n in one full sweep, we move in steps with careful choice of step values. Rearranging (3) gives us:

$$x' = \cos\theta[x - y\tan\theta]$$
$$y' = \cos\theta[y + x\tan\theta] \quad (4)$$

By restricting the rotation angles such that $\tan(\theta) = \pm 2^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations[1]. If the rotation at each iteration, i, is which direction to rotate rather than whether or not to rotate, then the $\cos(\delta_i)$ term becomes a constant (because of $\cos(\delta_i) = \cos(-\delta_i)$). The iterative rotation can now be expressed as:

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

Where, $K_i = \cos(\tan^{-1}2^{-i}) = 1/(\sqrt{1+2^{-i}})$

$d_i = \pm 1$

Removing the scale constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the $K_i's$ can be applied elsewhere in the system or treated as a part of a system processing gain. That product approaches 0.6073 as the number of iterations goes to infinity. Therefore, rotation algorithm has a gain, A_n, of approximately 1.647. The exact gain depends on the number of iterations, and obeys the relation.

$$A_n = \Pi\sqrt{1+2^{-i}} \quad (6)$$

The angle of a composite rotation is uniquely defined 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 be easily accomplished using a LUT. A better conversion method uses an additional adder-subtractor that accumulates the elementary rotation angles post iteration. The angle accumulator adds a third difference equation to the CORDIC algorithm:

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

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.

B. Rotation mode

The first mode of operation, called rotation by Volder, rotates the input vector by a specified angle(given as argument). Here, the angle accumulator is initialised 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. The equations for this are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (8)$$

Where, $d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases}$

Then,

$$x_n = A_n[x_0\cos z_0 - y_0\sin z_0]$$
$$y_n = A_n[y_0\cos z_0 - x_0\sin z_0]$$
$$z_n = 0$$
$$A_n = \Pi\sqrt{1+2^{-i}} \quad (9)$$

C. 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:

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

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (10)$$

Where, $d_i = \begin{cases} +1 & \text{if } y_i < 0 \\ -1 & \text{otherwise} \end{cases}$

Then,

$$x_n = A_n\sqrt{x_0^2 + y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}\left(\frac{y_0}{x_0}\right)$$
$$A_n = \Pi\sqrt{1 + 2^{-2i}} \quad (11)$$

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:

$$x' = -d \cdot y$$
$$y' = d \cdot x$$
$$z' = z + d \cdot \frac{\pi}{2} \quad (12)$$

Where, $d_i = \begin{cases} +1 & \text{if } y < 0 \\ -1 & \text{otherwise} \end{cases}$

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:

$$x' = d \cdot x$$
$$y' = d \cdot y$$
$$z_i = \begin{cases} z & \text{if } d = 1 \\ z - \pi & \text{if } d = -1 \end{cases} \quad (13)$$

Where, $d_i = \begin{cases} -1 & \text{if } x < 0 \\ +1 & \text{otherwise} \end{cases}$

Both reduction forms assume a modulo 2π representation of the input angle. The second reduction may be more convenient when wiring is restricted, as is often the case with FPGAs.

D.Evaluation of Sine and Cosine using CORDIC

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:

$$x_n = A_n \cdot x_0 \cdot \cos z_0$$
$$y_n = A_n \cdot x_0 \cdot \sin z_0 \quad (14)$$

By setting x_0 equal to $1/A_n$, the rotation produces the unscaled sine and cosine of the angle argument, z_0. Very often, the sine and cosine values modulate a magnitude value [1]. Using other techniques (e.g.,a look up table) requires a pair of multipliers to obtain the modulation. The CORDIC technique 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[1]. It is worth noting that the hardware complexity of the CORDIC rotator is approximately equal to that of a single multiplier with the same word size.

III. IMPLEMENTATION

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. The following figure 2 shows the top-level RTL schematic of the Sine/Cosine generators.

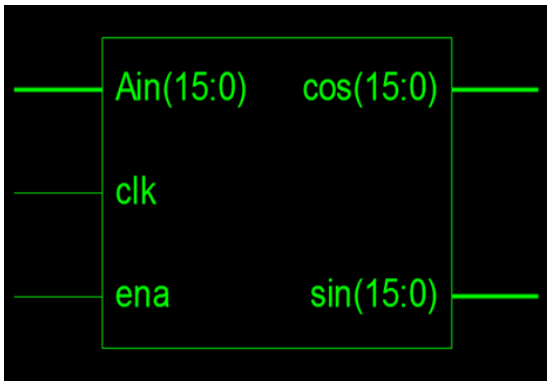

Figure 2.Top-level RTL schematic of the Sine/Cosine generators.

Figure 3 consists the ModelSim results for binary input angle Ain=45deg and binary outputs Xn(cos(Z0)), Yn(sin(Z0)) in the form of waveform and their corresponding magnitude respectively.

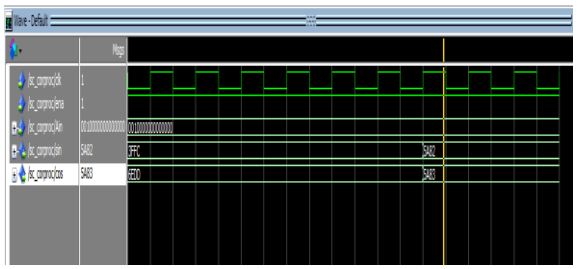

Figure 3. Sine/Cosine value for the given input angle Ain.

The following table shows the synthesis report for the CORDIC algorithm for calculating sine and cosine of the given angle.

Table 1. Synthesis report for the CORDIC algorithm.

Number of Slices	392(51%)
Maximum Frequency	171.283Mhz
Minimum period	9.541nSec

IV. CONCLUSION

We have successfully simulated an CORDIC algorithm for calculating the Sine and Cosine of an angle, on ModelSim simulator using the VHDL language. Our System can be implemented on Xilinx Spartan 3 XC3S50

using ISE Design Suite 9.2i and VHDL language. The design is more efficient and consumes less resources and is less time intensive. Our system has a maximum frequency of 171.283Mhz was reached with a minimum period of 9.541ns. 392(51%) slices are used.

REFERENCES

1. R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, pp. 191 – 200.
2. 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"
3. V. Sharma, FPGA Implementation of EEAS CORDIC based Sine and Cosine Generator, M. Tech Thesis, Dept. of Electronics and Communication Engineering, Thapar University, Patiala, 2009.
4. S. Panda, Performance Analysis and Design of a Discrete Cosine Transform Processor using CORDIC Algorithm, M. Tech Thesis, Dept. of Electronics and Communication Engineering, NIT Rourkela, Rourkela, Orissa, 2010.
5. R. K. Jain, B. Tech Thesis, Design and FPGA Implementation of CORDIC-based 8-point 1D DCT Processor, NIT Rourkela, Rourkela, Orissa, 2011.
6. J. Volder, "The CORDIC Trigonometric Computing Technique," IRE Transactions on Electronic Computing, Vol EC-8, Sept 1959, pp. 330-334.
7. Samuel Ginsberg, Compact and Efficient Generation of Trigonometric Functions using a CORDIC algorithm
8. J. S. Walther, "A unified algorithm for elementary functions," Proceedings of the Spring Joint Computer Conference, 1971, pp. 379-385.

AUTHOR PROFILE

Srinivasa Murthy H N, I received the Bachelor of Engineering degree in Electronics and communication Engineering from Channabasaveshwara Institute of Technology, Gubbi, Tumkur of Visvesvaraya Technological University, Belgaum, Karnataka in 2010. Currently, persuing the M.Tech degree in VLSI Design and Embedded Systems from Dayananda Sagar College of Engineering, Bangalore, Karnataka. My field or area of interests are Low power VLSI design, VLSI architectures and algorithms and SoC design for communication systems.

Roopa. M. received B.E, Degree from U.V.C.E Bangalore University, Bangalore In 1990, M.E, Degree from U.V.C.E, Bangalore University, Bangalore, India in 1994, pursuing Ph.D in the Department of Applied Electronics, Gulbarga University, Gulbarga, INDIA, currently she is working as Assistant Professor in the Department of Electronics and communication, Dayananda sagar college of engineering, Bangalore, INDIA, her research interests include Design of Digital circuits, VLSI Design and Embedded systems.