

Sine Wave Generation Using CORDIC Algorithm

¹Ashrita. T, ²Chidambara Rao. K

^{1,2}Dept. of ECE, AITAM, Tekkali, AP, India

Abstract

In this paper we describe an efficient CORDIC algorithm that completely eliminates the scaling factor. Besides we have proposed an algorithm to redefine the elementary angles for reducing the number of CORDIC iterations. The Coordinate Rotational Digital Computer (CORDIC) algorithm is another classic approach for sine wave generation. The particular architecture hereby presented generates the phase of the sine by self on enable and performs the CORDIC vector rotation in order to produce sine wave values at the rate of 4096 samples per cycle. The implementation was partitioned into two main blocks: a Sine magnitude generator (SMG) block and a CORDIC Logic Processor (CLP) block. The SMG produces the phase increments which drives the CLP, while performing replication in order to obtain the complete cycle of sine wave. Meanwhile, the CLP block performs the vector rotation and generates the sine magnitude. This is implemented using Spartan 3 FPGA XC3S200FT256 and XILINX ISE AND ISIM design and verification tools. MODELSIM is used to view Sine wave.

Keywords

Coordinate rotation digital computer(CORDIC)cosine/sine, FPGA, recursive architecture.

I. Introduction

The CORDIC algorithm was first introduced by Jack E. Volder in the year 1959 for the computation of Trigonometric functions, Multiplication, Division, Data type conversion, Square Root and Logarithms. It is a highly efficient, low-complexity and robust technique to compute the elementary functions. The CORDIC algorithm has found its way in various applications such as pocket calculators, numerical co-processors, to high performance radar signal processing, supersonic bomber aircraft with a digital counterpart.

The CORDIC algorithm is another classic approach for sine wave generation. This trigonometric based approach relies on vector rotations for performing successive mapping between polar and rectangular coordinates (and vice versa). Although not LUT based strategy, the CORDIC still relies on predetermined phase, amplitude and frequency values for calculating points on sine wave. The CORDIC's advantage over more traditional LUT based approaches lies on its capacity of producing simultaneous in-phase and quadrature components, as well as being a bit recursive algorithm (i.e., each iteration achieves in average a bit increment in accuracy).

It calculates the value of trigonometric functions like sine, cosine, magnitude and phase (arctangent) to any desired precision. It can also calculate hyperbolic functions (such as sinh, cosh and tanh). The CORDIC algorithm does not use Calculus based methods such as polynomial or rational function approximation. It is used as approximation function values on all popular graphic calculators, including HP-48G as the hardware restriction of calculators require that the elementary functions should be computed using only additions, subtractions, digit shifts, comparisons and stored constants. Today CORDIC algorithm is used in high performance vector rotation DSP applications, advanced circuit design, optimized low power design.

The Coordinate Rotational Digital Computer (CORDIC) has established its popularity in several important areas of application, calculation of discrete sinusoidal transforms like Fast Fourier Transform (FFT), discrete sine/cosine transforms (DST/DCT), householder transform (HT), etc. [6-8] Many variations have been suggested for efficient implementation of CORDIC with less number of iterations over the conventional CORDIC algorithm [911]. In [12] and [13] efficient scale-factor compensation techniques are proposed, which adversely affect the latency/throughput of computation.

The rest of this paper is structured as follows. Section II, reviews the existing CORDIC algorithms. Section III, deals with the micro-rotation sequence generation. The proposed CORDIC architecture is discussed in section IV. Section V, details the Field- Programmable Gate Array (FPGA) implementation and comparison with the existing architectures and section VI, concludes this paper.

II. Brief Overview of Cordic Algorithm

The CORDIC algorithm [3], can be derived from the general rotation transform, as stated on the equations below:

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

$$y' = y \cos \phi + x \sin \phi \quad (2)$$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = K_i \cdot \begin{bmatrix} 1 & -\mu_i \cdot \tan \alpha_i \\ -\mu_i \cdot \tan \alpha_i & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (3)$$

Equations 1 and 2 express the transformed Cartesian coordinates after a vector rotation by an angle of ϕ degrees. The before mentioned equations can be written as follows

$$x' = \cos \phi \cdot [x - y \tan \phi] \quad (4)$$

$$y' = \cos \phi \cdot [y + x \tan \phi] \quad (5)$$

Restricting the rotation angle so that $\tan \phi = \pm 2^{-i}$, the multiplication by $\tan \phi$ can be reduced to a simple shift operation; meanwhile arbitrary angles can be expressed in terms of these known restricted angles. So a vector rotation by an arbitrary angle can be achieved by successive elementary rotations by known angles.

In equation (4) and (5), $\cos \phi$ is constant, allowing these equations to be written as follows

$$x_{i+1} = K_i [x_i - y_i \cdot d_i \cdot 2^{-i}] \quad (6)$$

$$y_{i+1} = K_i [y_i + x_i \cdot d_i \cdot 2^{-i}] \quad (7)$$

$$\phi_{i+1} = \phi_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (8)$$

where

$$K_i = \cos(\tan^{-1} 2^{-i}) = \frac{1}{\sqrt{1+2^{-2i}}} \quad (9)$$

$$d_i = \text{sign}(\phi_i) \quad (10)$$

As the number of iterations increases so does the accuracy of the resulting Cartesian coordinate values. By feeding angles from 0° to 360° for vector rotation, the resulting x and y coordinates represent the corresponding sine and cosine scaled magnitudes. This can be achieved through pipelining the rotations. Pseudo code for the pipeline process is as follows:

Initiation: Given x_0 , y_0 and ϕ_0

For $n = 0: N-1$ /*pipeline*/

Calculate K_n

Calculate d_n

Calculate $x_{i+1}, y_{i+1}, \phi_{i+1}$

End loop

At the N-1 iteration of the loop, the values of x_N and y_N are obtained. Permitting the calculation of X_{out} and Y_{out} as seen in the equations below (For the given rotation angle, X_{out} represents the sine magnitude and Y_{out} represents the cosine magnitude):

$$X_{out} = x_N / K_N \quad (11)$$

$$Y_{out} = y_N / K_N \quad (12)$$

Where

$$K_N = \prod_{i=0}^{N-1} 1 / (\sqrt{1 + 2^{-2i}}) \quad (13)$$

A. CORDIC Algorithm Implementation

1. Phase Angle Mapping

Phase angle and sine magnitude are represented in this implementation with 16 bits, scaled value of 360° is 2^{16} .

$$1^\circ = 2^{16} / (360^\circ) = 182 \quad (14)$$

Since the choice of sampling rate, $f_s = 4096$ per cycle

$$1^\circ = 4096 / 360 = 11.3778 \text{ samples} \quad (15)$$

Therefore, sampling duration should be

$$182 / 11.3778 = 16 \text{ (approx)} \text{ and} \quad (16)$$

$$\text{resolution} = 1^\circ / 11.37 = 0.0879^\circ \text{ (approx)}$$

2. Magnitude Scaling

Magnitude is represented with 16 bits, out of these MSB is used to represent sign and rest 15 bits are used to represent the magnitude. Output represents the values in the 0 to +1 range. The results can be derived as follows:

Sin 0 = 0 is represented by the minimum values possible with 15 bits.

Sin 90 = 1.0 is represented by the maximum values possible with 15 bits, i.e., 2^{15} .

$$\text{Therefore output scaling factor} = 1 / 2^{15} \quad (17)$$

III. Generalized Micro-Rotation Selection

In the proposed generalized micro-rotation sequence, we perform multiple iterations of basic-shift, followed by non-repetitive unidirectional iterations of the micro-rotations corresponding to other shift indices, to minimize the number of iterations and achieve adequate range of convergence.

A. Organization of Micro-Rotation Sequence

In the proposed scheme, we represent the rotation angle " θ " as

$$\theta = n_1 \cdot \alpha_s + \sum_{i=1}^{n_2} \alpha_{s_i}, \quad n = n_1 + n_2 \quad (18)$$

Where α_s is the elementary angle corresponding to the basic-shift, α_{s_i} are elementary angles for other shifts, n_1 and n_2 are non-negative integers and n represents the total number of iterations. If we do not use any micro-rotation of angle α_s then n_1 is zero, and $n_2 = n$. On the other hand, if the desired angle of rotation " θ " is a multiple of then n_2 is zero and $n_1 = n$.

B. Defining the Elementary Angles

The elementary angles α_s and α_{s_i} are given by

$$\alpha_s = 2^{-s} \text{ and } \alpha_{s_i} = 2^{-s_i} \quad (19)$$

Where s is the basic-shift and $s_i > s$ is the shift for i th iteration. For basic-shift=2, we can find $\alpha_s = 7\pi/88$ and for basic-shift = 3,

we can find $\alpha_s = 7\pi / 176$.

C. Generalized Micro-Rotation Sequence Identification

We identify the micro-rotations depending on the bit representation of the desired rotation angle in radix-2 system using most-significant-1 detector. For this we restrict the maximum rotation angle to $\pi/4$ radians as the entire coordinate space $[0, 2\pi]$ can be mapped to $[0, \pi/4]$ using octant symmetry of sine and cosine functions.

If the most significant-1 location (M) of the rotation angle " θ " is smaller than the basic-shift " s ", elementary angle of the basic-shift would be used for the CORDIC iteration. For a fixed word-length of N-bit, the shift (s_i) for elementary angle is given by

$$s_i = N - M \quad (20)$$

Table 1:

Input: angle to be rotated(θ_i)
Begin
M= Most Significant-1 Location of θ_i
if (M==15)then
$\alpha = 0.25$ radians
Shift, $s_i = 2$ and $\theta_{i+1} = \theta_i - \alpha$
Else
Shift, $s_i = 16 - M$
$\theta_{i+1} = \theta_i$ with $\theta_i[M] = '0'$
End

IV. Proposed Design Architecture

On initial reset pulse and asserted enable signal architecture (fig. 1) is designed to generate the sine wave samples. Frequency of the oscillations depends on the clock frequency. It is designed to produce one sample for every 0.0879° (approx).

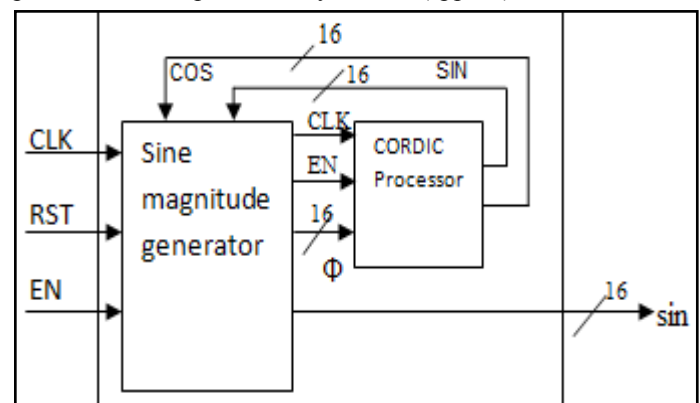


Fig. 1: FPGA Design Architecture of Sine Wave Generator

Sequential Logic-1 shown in fig. 1, generates the next state variables and D flip flops : DA and DB derives present state variables from the next state variables. These present state variables are used as select lines for the MUX:-M1, which outputs the initial phase angle for the given quadrant. Combinational circuit-1 derives the select line for MUX:-M2 to choose either initial phase value or running phase value to get add or subtracted by the phase step value held

by the resolution register. Add sub unit performs either addition or subtraction on proceeding phase value and phase step value held by resolution register. D flip flops : D0 through D15 holds the Add sub unit result for a clock cycle.

A. Sign Magnitude Generator

This module feeds the phase to CORDIC module on enable and replicates the positive and negative quarter waves to obtain the complete cycle. Sine magnitude growth through the full cycle is described in fig. 2.

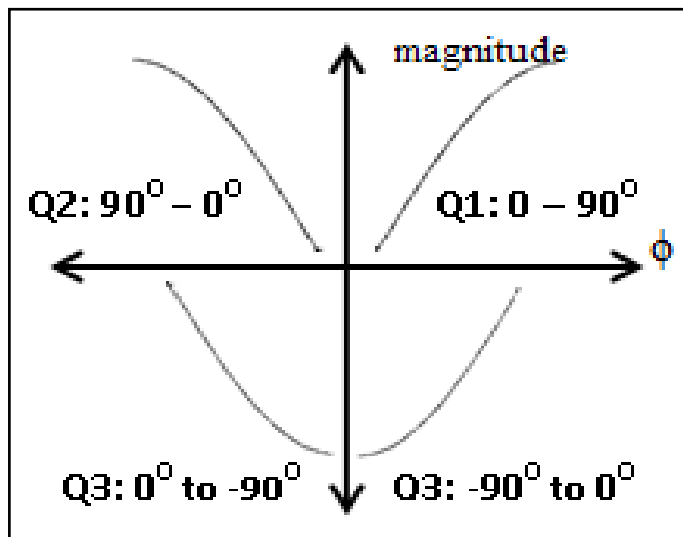


Fig. 2: Angle Mapping Over the Full Cycle of Sine Wave

B. CORDIC Processor

CORDIC processor consists of CORDIC pipe and CORDIC rotation module. Pipeline processor executes the iterations specified in the pseudo code and CORDIC rotation module performs the vector rotation by the angle specified by the pipeline processor

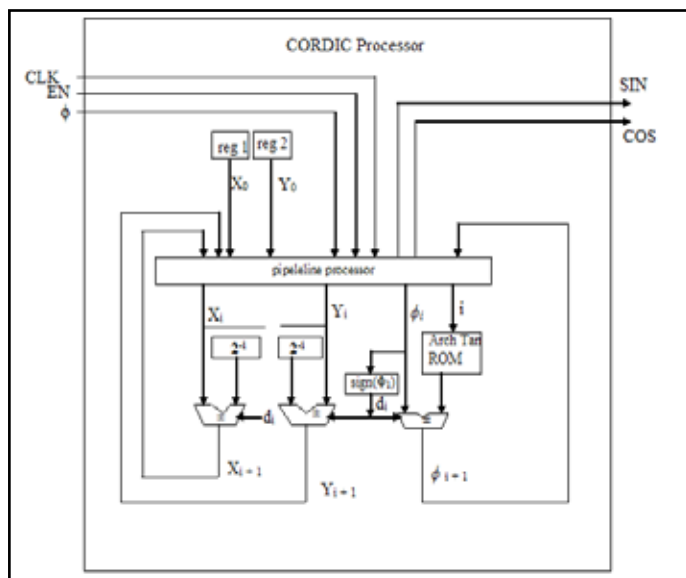
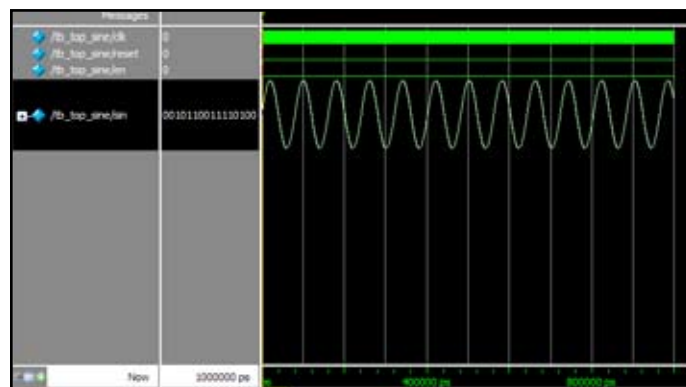


Fig. 3: CORDIC Processor

CORDIC processor performs the computations described in equations 6, 7 and 8. In this circuit reg1 holds the value 19898. It is a scaled (2^{16}) value of $0.6072(K_N)$ and reg2 holds 16 zero bits. Arch tan table is basically ROM, which stores the arch tan table described in CORDIC algorithm.

1. Simulation Result



2. Synthesis Report

(i). IO Ports

Table 2: List of IO Ports for Sine/Cosine CORDIC Core

Port	Width	Direction	Description
CLK	1	Input	System clock
EN	1	Input	Clock enable signal
RESET	1	Input	Resets the core
Sin	16	Output	Sine output

V. FPGA Implementation

The proposed architecture is coded in Verilog and synthesized using Xilinx ISE 12.1i to be implemented in Xilinx Spartan 3E (XC3S200TQ144) device.

VI. Conclusions and Future Work

To increase throughput, pipelined CORDIC processors is used in this paper with the penalty that the pipelined structure would require a large amount of chip area [1]. The CORDIC algorithm is suitable for the DDS with more than 14 output bits, where the sine look-up table even with the compression is still too large for high speed operation. Cosine points also can be generated along with Sine the points because CORDIC algorithm can calculate both in-phase and quadrature components simultaneously [3]. Through programming the constant value (equation 13) in reg1 shown in fig.3, amplitude of the sine wave can be tuned. With this implementation Sine wave can be generated with the error of 0.0021% at peaks and zero crossings. This sine precision is obtained after skipping without generating the 77 samples at these crossings.

References

- [1] Supriya Aggarwal, Pramodh K. Mehar, Kavita Khare "Area-Time Efficient Scaling-free CORDIC using Generalized Micro-Rotation Selection", IEEE Transactions on very large scale integration (VLSI) systems, Vol. 20, No. 8, August 2012.
- [2] Jouko Vankka, "Methods of mapping from phase to sine amplitude in direct digital synthesis", IEEE international frequency control symposium, 1996.
- [3] Maher Jridi, Ayman Alfalou, "Direct Digital Frequency Synthesizer with CORDIC Algorithm and Taylor Series Approximation for Digital Receivers", EuroJournals Publishing, Inc. 2009.
- [4] Ray Andarka, "A Survey of CORDIC Algorithms For FPGA Based Computers", Andarka Consulting group, Inc

- [5] Tanya Vladimirova and Hans Tiggeler, "FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm", Surrey Space Centre University of Surrey, Guildford, Surrey.
- [6] J.E. volder, "The CORDIC trigonometric computing technique", IRE Trans. Electron. Comput., Vol. EC-8, pp. 330-334, sep. 1959.
- [7] K. Maharatna, A.S. Dhar, S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST", Signal Process., Vol. 81, pp. 1813-1822, 2001.
- [8] P.K. Meher, J. Walls, T.-B. Juang, K. Sridharan, K. Maharatna, "50 years of CORDIC: Algorithms, architectures and applications", IEEE Trans. Circuits Syst. I, Reg. Papers, Vol. 56, no. 9, pp. 1893-1907, Sep. 2009.
- [9] C.S. Wu, A.Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture", IEEE Trans. Circuits Syst. II, Exp. Briefs, Vol. 48, No. 6, pp. 548-561, Jun. 2001.
- [10] C.-S. Wu, A.Y. Wu, C.-H. Lin, "A high performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes", IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., Vol. 50, No. 9, pp. 589-601, sep. 2003.
- [11] Y.H. Hu, S. Naganathan, "An angle recoding method for CORDIC algorithm implementation", IEEE Trans. Comput., Vol. 42, No. 1, pp. 99-102, Jan. 1993.
- [12] M. G. B. Sumanasena, "A scale factor correction scheme for the CORDIC algorithm", IEEE Trans. Comput., Vol. 57, No. 8, pp. 1148-152, Aug. 2008.
- [13] J. Villalba, T. Lang, E.L. Zapata, "Parallel compensation of scale factor for the CORDIC algorithm", J. VLSI Signal Process. Syst., Vol. 19, No. 3, pp. 227-241, Aug. 1998.