INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

**ISSN: 2454-132X**

**Impact factor: 4.295**

***(Volume 4, Issue 3)***

Available online at: www.ijariit.com

# FPGA implementation of CORDIC algorithm

*Chaitra Y*
*chetuchaya123@gmail.com*
*P. E. S. College of Engineering, Mandya, Karnataka*

*Dr. H. S. Sheshadri*
*hssheshadri@gmail.com*
*P. E. S. College of Engineering, Mandya, Karnataka*

## ABSTRACT

***Trigonometric functions have a wide variety of applications in real life. Trigonometric-related calculations which are widely found in a broad range of applications can be performed by using CORDIC algorithm. Specially SIN and COSINE waves have been very useful in medical science, signal processing, geology, electronic communication, thermal analysis and many more. Real life application requires fast calculation capabilities as much as possible. CORDIC is Coordinate Rotation Digital Computing. Cordic algorithm's used to find the sine and cosine angle in the integer form. The CORDIC algorithmic is an iterative computing algorithm capable of evaluating various elementary functions using a unified shift-and-add approach Used to calculate a wide variety of functions. It requires only addition, subtraction, bit shift and looks up the table. Where multipliers are not used. The algorithm is implemented here for an angle of 30 degrees and the Active-HDL results are shown along with sine and cosine values in the tabular form for different rotation angles. It can be used in digital synchronizers, graphics processors, scientific calculators, and so on. It offers a substantial saving of area complexity over the conventional Design. Designing of CORDIC Processor in Verilog to determine the sine and cosine The result has been shown in this paper that resolution of CORDIC algorithm is best for implementing many trigonometrical functions. Here a brief concept, design strategy, implementation of cordic architecture and summary of cordic synthesis results based on Xilinx FPGA. The system simulation is carried out using Xilinx ISE design suite 14.1. The system is implemented using Xilinx Spartan-6 XC6SLX45 FPGA with Verilog hardware description language.***

**Keywords:** *Cordic, Trigonometric functions, Rotation, Angel.*

## 1. INTRODUCTION

The cordic was introduced in 1959 by Jack E. Volder. It is very efficient to compute the values of sin, cos, tan, sinh, cosh and tanh. It requires no multiplication. The most versatile application of the cordic algorithm is used in the modern digital systems, such as mobile communication, digital signal processing, image processing and other fields.

The cordic algorithm requires only addition, subtraction and bit shift which involves in a simple successive iterative procedure to perform several computing tasks by operating in either rotation mode or vector mode following any one among linear, hyperbolic and circular trajectories.

The calculation of trigonometric functions usually follows the methods: ROM Look up Table, Polynomial approximation, and cordic algorithm. ROM look-up table is a simple, easy and direct method which can be used to reduce trigonometric functions, the only drawback is that when the data quantity increase, storage space will increase exponentially. The polynomial approximation method is rarely used because of multiplication, consumes more resources and hardware complexity. Cordic algorithm has outstanding advantages when compared with ROM look up the table and polynomial approximation. It requires no multiplication. All operations are an only shift, addition, and subtraction. Where it consumes less resources and hardware implementation easily.

### 1.1 Objectives

- To implement with shift, add and subtractor type algorithm.
- To compute trigonometric functions such as hyperbolic, linear and logarithmic functions.

### 1.2 Cordic Methodology

The name CORDIC stands for Coordinate Rotation Digital Computer. J E. Volder developed the method of computing the rotation of a vector in a Cartesian coordinate system. If a vector V with coordinates (x, y) is rotated to an angle then we get a new vector V ' can be obtained with coordinates (x', y') where x' and y' can be obtained by initial coordinates x, y and by the following method[2] as shown in figure.

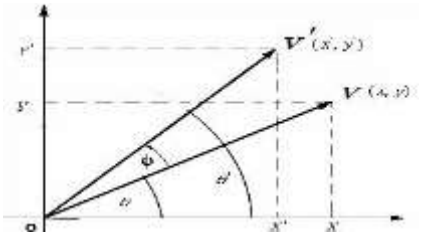


**Chart: Rotation of a vector V by an angle**

$X = r \cos\theta$, $Y = r \sin\theta$ (1)

$(x') = (x.\cos(\emptyset) - y.\sin(\emptyset))$ (2)

$(y') = (y.\cos(\emptyset) + x.\sin(\emptyset))$ (3)

$x' = \cos(\emptyset)\,[x - y.\tan(\emptyset)]$ (4)

$y' = \cos(\emptyset)\,[y + x.\tan(\emptyset)]$ (5)

Now if $\tan(\emptyset) = 2^{-i}$ the value inside the bracket can be simplified. In digital hardware ,this denotes a simple shift operation. Furthermore, if those rotations are performed iteratively and in both directions ,every value of tan (∅) is presentable with $\emptyset = \arctan(2^{-i})$ the cosine term could also be simplified and since $\cos(\emptyset) = \cos(-\emptyset)$ it is a constant for a fixed number of iterations.

This iterative rotation can now be expressed as:

$xi+1 = ki\,[xi - yi.di.\,2^{-i}]$ (6)

$yi+1 = ki\,[yi - xi.di.\,2^{-i}]$ (7)

where I denotes the number of rotation required to reach the required angle of the required vector, $\cos(\arctan(2^{-i}))$ and di = ±1. The product of the Ki's represents the so-called K factor :

$k = \pi_{i=0}^{n-1} k_i$

where $\pi_{i=0}^{n-1} k_i = \cos\emptyset_o \cos\emptyset_1 \cos\emptyset_2 \cos\emptyset_3 \cos\emptyset_4 \ldots\ldots\ldots\ldots\ldots \cos\emptyset_{n-1}$ (∅ is the angle of rotation here for n times rotation).$k_i$is the gain and its value changes as the number of

$k_i = \pi_{i=0}^{7} \cos\emptyset_i = \cos\emptyset_o . \cos\emptyset_1 . \cos\emptyset_2 . \cos\emptyset_3 . \cos\emptyset_4 . \cos\emptyset_5 . \cos\emptyset_6 . \cos\emptyset_7$

| i | $\tan(\varphi) = 2^{-i}$ | $\varphi = a\tan(2^{-i})$ |
|---|---|---|
| 0 | 1 | 45 |
| 1 | 1/2 | 26.565 |
| 2 | 1/4 | 14.036 |
| 3 | 1/8 | 7.125 |
| 4 | 1/16 | 3.576 |
| 5 | 1/32 | 1.79 |
| 6 | 1/64 | 0.895 |
| 7 | 1/128 | 0.448 |
| 8 | 1/256 | 0.224 |
| 9 | 1/512 | 0.112 |

From the TABLE 1, it can be seen that precision up to $0.4469^0$ is possible for 8-bit CORDIC hardware. To simplify each rotation, picking αi (angle of rotation in an ith iteration) such that αi = di $2^{-i}$. such that it has value +1 or -1 depending upon the rotation i. e. di∈{+1,-1} . Then

- $X_{i+1} = k_i\,[X_i - Y_i\,d_i\,2^{-i}]$ (9)
- $Y_{i+1} = k_i\,[Y_i + X_i\,d_i\,2^{-i}]$ (10)
- $Z_{i+1} = Z_i - d_i\,\tan^{-1}(2^{-i})$ (11)

The computation of $X_{i+1}$ or $Y_{i+1}$ requires an i-bit right shift and an add/subtract. If the function $\tan^{-1}2^{-i}$ is pre-computed and stored in a table (TABLE 1) for different values of i, a single add/subtract suffices to compute $Z_{i+1}$. Each CORDIC iteration thus involves two shifts, a table lookup, and three additions. If the rotation is done by the same set of angles (with + or - signs), then the expansion factor K is a constant and can be pre-computed.[3] For example, to rotate by 30 degrees, the following sequence of angles be followed that add up to ≈ 30 degrees.

30.0 ≈45.0 - 26.6 + 14.0 - 7.1 + 3.6 + 1.8 - 0.9 + 0.4 -0.2 + 0.1 = 30.1

In effect, what actually happens in CORDIC is that z is initialized to 30 degrees and then, in each step, the sign of the next rotation angle is selected to try to change the sign of z; that is, di= sign ( zi) is chosen, where the sign function is defined to be -1 or +1 depending on whether the argument is negative or non-negative.[4]- [7] Table 2 shows the process of selecting the signs of the rotation angles for the desired rotation of +30 degree.

**Table 2: Choosing the signs of the rotation angles to force z to zero**

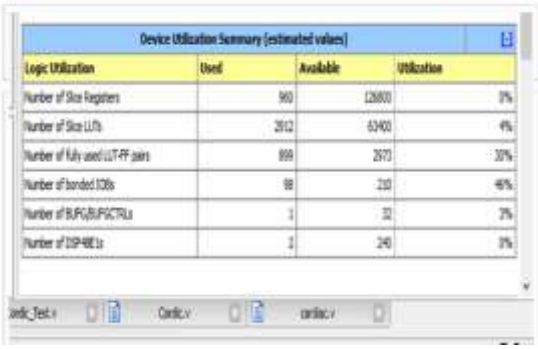
| i | $Z_i - \phi$ | $Z_{i+1}$ |
|---|---|---|
| 0 | +30.0-45.0 | -15 |
| 1 | -15.0+26.6 | 11.6 |
| 2 | +11.6-14.0 | -2.4 |
| 3 | -2.4+7.1 | 4.7 |
| 4 | +4.7-3.6 | 1.1 |
| 5 | +1.1-1.8 | -0.7 |
| 6 | -0.7+0.9 | 0.2 |
| 7 | +0.2-0.4 | -0.2 |
| 8 | -0.2+0.2 | 0 |
| 9 | +0.0-0.1 | -0.1 |

### 1.3 Results

**Fig 2: simulation results in Xilinx**

This is the simulation results in Xilinx which represents sin and cosine waveform.



| Device Utilization Summary (estimated values) | | | |
|---|---|---|---|
| Logic Utilization | Used | Available | Utilization |
| Number of Slice Registers | 960 | 126800 | 1% |
| Number of Slice LUTs | 2612 | 63400 | 4% |
| Number of fully used LUT-FF pairs | 899 | 2673 | 33% |
| Number of bonded IOBs | 88 | 210 | 41% |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% |
| Number of DSP48E1s | 1 | 240 | 1% |

**Fig 3: shows device utilization summary**

**Fig 4: FPGA Implementation results**

## 2. CONCLUSION

Cordic is not the fastest way to perform multiplications or to compute logarithms and exponentials but, since the same algorithm allows the computation of most mathematical functions using very simple basic operations, it is attractive for hardware implementations. It presents mathematical and analytical aspects of implementing the cordic algorithm for sinusoidal calculations. This method is much simpler than the conventional methods as the computations are reduced to only shift and add operations instead of complex multiplication operations.

## 3. ACKNOWLEDGEMENT

This project has been carried out at our department with the advanced FPGA kit purchased under the financial assistance from VGST grants (Government of Karnataka) under Kfist during 2016 -2018 for the development of medical image analysis laboratory. We express our sincere thanks for the facilities provided at the college for implementing this project.

## 4. REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. EC-8, pp. 330–334, Sep. 1959.
[2] K. Maharatna, A. S. Dhar, and S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST," Signal Process., vol. 81, pp. 1813–1822, 2001.
[3] P. K. Meher, J.Walls, T.-B. Juang, K. Sridharan, and 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.
[4] C. S. Wu and A. Y. Wu, "Modified vector rotational CORDIC (MVRCORDIC) algorithm and architecture," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 48, no. 6, pp. 548–561, Jun. 2001.
[5] C.-S.Wu, A.-Y.Wu, and 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.
[6] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," IEEE Trans. Compute., vol. 42, no. 1, pp. 99–102, Jan. 1993.
[7] M. G. B. Sumanasena, "A scale factor correction scheme for the CORDIC algorithm," IEEE Trans. Compute., vol. 57, no. 8, pp. 1148–1152, Aug. 2008.