Implementation of Scientific Calculator Using CORDIC Algorithm on FPGA

B.Swapna
Assistant Professor,
M.Tech(VLSI System Design),
Electronics & Communication Engineering,
TKR College of Engineering & Technology.

S.Sai Sree Andal
Assistant Professor,
M.Tech (VLSI System Design),
Electronics & Communication Engineering,
TKR College of Engineering & Technology.

Abstract:

COordinate Rotation DIgital Computer (CORDIC) algorithm has potential for efficient and low-cost implementation of a large class of applications which include the generation of trigonometric, logarithmic and transcendental elementary functions, complex number multiplication, matrix inversion, solution of linear systems and general scientific computation. This paper presents a brief overview of the developments in the CORDIC algorithm and its architectures.

Keywords:
CORDIC Algorithms; CORDIC Architectures; FPGA.

I.INTRODUCTION:

FIRST described in 1959 [1], CORDIC algorithm is an iterative algorithm, which can be used for the computation of trigonometric functions, multiplication and division. Last half century has witnessed a lot of progress in design and development of architectures of the algorithm for high-performance and low-cost hardware solutions. CORDIC algorithm got its popularity, when [2] showed that, by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials, and square. During the same time, [3] showed that CORDIC technique is a better choice for scientific calculator applications.

The popularity of CORDIC was very much enhanced thereafter primarily due to its potential for efficient and low-cost implementation. With the advent of low cost, low power FPGAs, this algorithm has shown its potential for efficient and low-cost implementation. CORDIC algorithm can be widely used in as wireless communications, Software Defined Radio and medical imaging applications, which are heavily dependent on signal processing. Some other upcoming applications are:

» Direct frequency synthesis, digital modulation and coding for speech/music synthesis and communication.
» Direct and inverse kinematics computation for robot manipulation.
» Planar and three-dimensional vector rotation for graphics and animation.

Although CORDIC may not be the fastest technique to perform these operations, yet it is attractive due to the simplicity and efficient hardware implementation.

The development of CORDIC algorithm and architecture has taken place for achieving high throughput rate and reduction of hardware-complexity as well as the latency of implementation. Latency of implementation is an inherent drawback of the conventional CORDIC algorithm. Angle recoding schemes and higher radix CORDIC have been developed for reduced latency realization. Parallel and pipelined CORDIC have been suggested for high-throughput computation.

This paper presents an overview of the development of CORDIC algorithm. The paper is organized as follows: Section II discusses the basics of CORDIC algorithm, different CORDIC architectures are discussed in Section III. The conclusion along with future research directions are discussed in Section IV.

II.DEFINITION OF CORDIC:

The CORDIC is very simple and iterative convergence algorithm that reduces complex multiplication, greatly simplifying overall hardware complexity. This serves as an attractive option to system designers as they continue to face the challenges of balancing aggressive cost and power targets with the increased performance required in next generation signal processing solutions. The basic principle underlying the CORDIC-based computation, and present its iterative algorithm for different operating modes and planar coordinate system.

A.Overview of CORDIC Algorithm:

CORDIC algorithm has two types of computing modes Vector rotation and vector translation. The CORDIC algorithm was initially designed to perform a vector rotation, where the vector V with components (X,Y) is rotated through the angle.

θ yielding a new vector V ‘ with component (X’,Y’) shown
in Fig. 1.

$$V' = [R][V] \qquad (1)$$

where R is the rotation matrix:

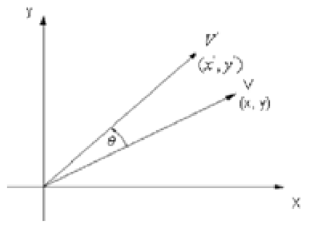

Figure 1: Vector Rotation

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

$$R=\begin{bmatrix}\frac{1}{\sqrt{1+\tan^2\theta}} & -\frac{\tan\theta}{\sqrt{1+\tan^2\theta}}\\ \frac{\tan\theta}{\sqrt{1+\tan^2\theta}} & \frac{1}{\sqrt{1+\tan^2\theta}}\end{bmatrix} \quad (3)$$

By factoring out the cosine term in (3), the rotation matrix **R** can be rewritten as

$$R=\left(1+\tan^2\theta\right)^{-1/2}\begin{bmatrix}1 & -\tan\theta\\ \tan\theta & 1\end{bmatrix} \quad (4)$$

and can be interpreted as a product of a scale-factor $K=\left[(1+\tan^2\theta)^{-1/2}\right]$ given by

$$R_c=\begin{bmatrix}1 & -\tan\theta\\ \tan\theta & 1\end{bmatrix} \quad (5)$$

In vector translation, rotates the vector V with component (X, Y) around the circle until the Y component equals zero as illustrated in Fig. 2. The outputs from vector translation are the magnitude X' and phase , of the input vector V. After vector translation, output equations are:

$$X'=K_i\sqrt{X^2+Y^2} \quad (6)$$

$$Y'=0 \quad (7)$$

$$\theta=a\tan\left(\frac{Y}{X}\right) \quad (8)$$

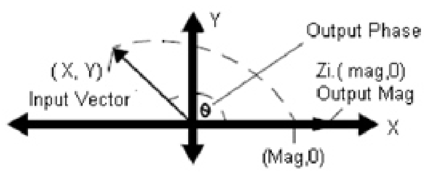

Figure 2: Vector Translation

To achieve simplicity of hardware realization of the rotation, the key ideas used in CORDIC arithmetic are to decompose the rotations into a sequence of elementary rotations through predefined angles that could be implemented with minimum hardware cost and to avoid scaling, that might involve arithmetic operation, such as square-root and division. The second idea is based on the fact the scale-factor contains only the magnitude information but no information about the angle of rotation.

B.Generalized CORDIC Algorithm:

After few years, Walther found how CORDIC iterations could be modified to compute hyperbolic functions [2] and reformulated the CORDIC algorithm in to a generalized and unified form which is suitable to perform rotations in circular, hyperbolic and linear coordinate systems. The unified formulation includes a new variable m , which is assigned different values for different coordinate systems. The generalized CORDIC is formulated as follows:

$$\begin{aligned} x_{i+1} &= x_i - m\sigma_i\,.2^{-i}.y_i\\ y_{i+1} &= y_i + \sigma_i\,.2^{-i}.x_i\\ w_{i+1} &= w_i - \sigma_i\,.\alpha_i \end{aligned} \quad (9)$$

Here $\sigma_i=\begin{cases}sign(w_i) & \text{for rotation mode}\\ -sign(w_i) & \text{for vectoring mode}\end{cases}$

III. CORDIC ARCHITECTURES:

CORDIC computation is inherently sequential due to two main bottlenecks firstly the micro-rotation for any iteration is performed on the intermediate vector computed by the previous iteration and secondly the (i+1) th iteration could be started only
after the completion of the ith iteration, since the value ofi 1

which is required to start the (i+1)th iteration could be known only after the completion of the ith iteration. To alleviate the second bottleneck some attempts have been made for evaluation of i values corresponding to small micro-rotation

angles.However, the CORDIC iterations could not still be performed in parallel due to the first bottleneck. A partial parallelization has been realized in [4] by combining a pair of conventional CORDIC iterations into a single merged iteration which provides better area-delay efficiency.

But the accuracy is slightly affected by such merging and cannot be extended to a higher number of conventional CORDIC iterations since the induced error becomes unacceptable [5].

Parallel realization of CORDIC iterations to handle the first bottleneck by direct unfolding of micro-rotation is possible, but that would result in increase in computational complexity and the advantage of simplicity of CORDIC algorithm gets degraded [6].

Although no popular architectures are known to us for fully parallel implementation of CORDIC, different forms of pipelined implementation of CORDIC have however been proposed for improving the computational throughput [7].

To handle latency bottlenecks, various architectures have been developed and reported in this review. Most of the well-known architectures could be grouped under bit parallel iterative CORDIC, bit parallel unrolled CORDIC , bit serial iterative CORDIC and pipelined CORDIC architecture.

IV. CONCLUSION:

CORDIC algorithm can be implemented by using simple hardware through repeated shift-add operations. This feature makes it attractive for a wide variety of applications. Moreover, its applications in several diverse areas including signal processing, image processing, communication, robotics and graphics apart from general scientific and technical computations have been explored. In the last half century, several algorithms and architectures have been developed to speed up the CORDIC algorithm by reducing its iteration counts and through its pipelined implementation.

ACKNOWLEDGMENT:

The authors would thanks the reviewers for their help in improving the document.

REFERENCES:

[1]J. E. Volder, "The CORDIC trigonometric computing technique," IRE.Transactions on Electronic Computers, vol. EC- 8, pp. 330–334, Sept. 1959.

[2]J. S. Walther, "A unified algorithm for elementary functions," in Proceedings of the 38th Spring Joint Computer Conference, Atlantic City, NJ, 1971, pp.379–385.

[3]D. S. Cochran, "Algorithms and accuracy in the HP-35," Hewlett-Packard Journal, pp. 1–11, June 1972.

[4]S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms,"IEEE Transactions on Computers, volume 46, no. 11, pp. 1202–1207, November1997.

[5]S. Wang and E. E. Swartzlander, "Merged CORDIC algorithm," in IEEE International Symposium on Circuits Systems (ISCAS'95),1995, volume 3, pp.1988–1991.

[6]B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," Microelectronics Journal, volume 33, Pp.77–89, 2002.

[7]E. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering and array processing," in IEEE International Conference on Acoustic, Speech, Signal Processing, ICASSP'84, March 1984, volume 9, pp.250–253.

[8]D. E. Metafas and C. E. Goutis, "A floating point pipeline CORDIC processor with extended operation set," in IEEE International Symposium on Circuits and Systems, ISCAS'91, June 1991, volume 5, pp. 3066–3069.