Hardware Implementation of CORDIC Algorithm

Anuvab Sahoo and Dr. Mamata Panigrahy
KIIT,Bhubaneswar

***Abstract*—CORDIC algorithm provides an effective way of computing trigonometric and hyperbolic functions. This algorithm replaces multiplication operation through iterative addition and bit shifting action. This not only saves area but also improves the throughput. This paper reviews the basic CORDIC algorithm and implements folded word-serial and unfolded pipelined CORDIC architecture. These designs are implemented using Xilinx Artix-7 FPGA. Hardware utilization and speed of operation are tabulated for both designs. Accuracy for 16 bit precision CORDIC design is noted by comparing the same with Matlab simulation result.**

***Index Terms*—CORDIC, FPGA, sine, cosine, tangent, rotation, pipelined architecture**

I. Introduction

CORDIC (Coordinate rotation Digital Computer) is a simple algorithm based on 2-D geometry. This algorithm acquires attention due to its efficient hardware solution for complex trigonometric and mathematical operations.

Volder [1] introduced CORDIC algorithm that performs rotation and computes trigonometric functions by simple shift and add operation. A unified algorithm proposed by Walther [2] ensures circular, hyperbolic and linear rotation through CORDIC. This enables exponential, square root and logarithm calculations. This algorithm requires several iterations to settle at the desired result which makes the process unsuitable for high speed applications. Several researchers attempted to speed up the process. Some of them adopted higher radix [3] for each iteration that reduces total number of repetitions. Others adopted redundant arithmetic [4] and branching CORDIC [5] to reduce per iteration delay. Hardware architectures also been designed to improve the speed of operation[6] [7] [8].

Transformation like DFT , DHT, DCT along with filtering, SVD operation demands complex multiplication, division and square root computations. CORDIC algorithm solves such computations through simple hardware structures with few adders and Shifters [9]. CORDIC algorithm have its application in several areas like image processing, biomedical and DSP applications [10] and communication systems.

In this paper, hardware implementation of rotational CORDIC is presented and a comparative study between sequential and pipelined CORDIC design is also furnished.

The rest of the paper is organized as follows. Section II describes the theory behind CORDIC algorithm. Section III presents classical and pipelined CORDIC architecture. A detailed analysis of the designs along with the results are given in section IV. Finally, Section V concludes the paper.

II. CORDIC Theory

Rotating a vector from its initial position [x,y] by an angle θ results a vector at $[x_k, y_k]$ in the cartesian plane which is given by

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

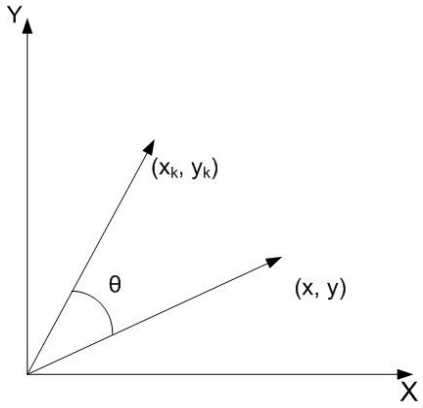

Figure 1: 2-D vector rotation

$$\begin{aligned} x_k &= x\cos\theta - y\sin\theta \\ y_k &= x\sin\theta + y\cos\theta \end{aligned} \tag{2}$$

Factoring out the $cos\theta$ term we have

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \cos\theta \begin{bmatrix} x - y\tan\theta \\ y + x\tan\theta \end{bmatrix} \tag{3}$$

The required angle of rotation θ is obtained by performing a series of small elementary rotations (ϕ). At each iteration, these small rotational angles (ϕ) decreases the error (z) between the desired and obtained angle. Value of ϕ is choosen such that it can be expressed in radix2 system. Mathematical θ and ϕ are represented as

$$\begin{aligned} \theta &= \sum_{i=0}^{n-1} d_i\phi_i \\ \tan\phi_i &= \pm 2^{-i} \end{aligned} \tag{4}$$

Variable d_i indicates the direction of rotation which reduces error angle. Anti-clockwise and clockwise rotations are denoted with value +1 and -1 respectively. The desired angle of rotation θ can be expressed as a function of no. of micro-rotations (n) given by

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \prod_{i=0}^{n-1} \cos\phi_i \begin{bmatrix} 1 & -d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5}$$

978-1-5386-8333-0/18/$31.00

At each iteration *i*, small rotational angles are accumulated and the error angle (z) gets updated. Eqn.5 along with the error angle (z) can now be expressed as a function of i^{th} iteration, direction of rotation d_i and initial vector position (x_i, y_i) given by

$$\begin{aligned} x_{i+1} &= K_i[x_i - y_i.d_i.2^{-i}] \\ y_{i+1} &= K_i[y_i + x_i.d_i.2^{-i}] \\ z_{i+1} &= z_i - d_i.(tan^{-1}(2^{-i})) \end{aligned} \tag{6}$$

where

$$d_i = \pm 1$$

$$K_i = \prod_{i=0}^{n-1} cos(tan^{-1}(2^{-i})) = 1/(\prod_i \sqrt{1+2^{-2i}})$$

As the number of iteration increases, product of $\cos\phi_i$ approaches 0.6073. K_i is treated as a gain factor for this algorithm and attains a value of 1.647. Removing the scale constant K_i from Eqn.6, solves the vector rotation problem by shift and add operations.

The CORDIC rotator is normally operated in 2 modes, vectoring and rotation mode. In vectoring mode, the initial and final vector positions are given and the task is to find the difference angle between the vectors. In rotation mode, the targeted position is obtained by rotating input vector by the specified angle. The angle accumulator is initiated with desired rotation angle. After each iteration, the magnitude of residual angle (z) gets reduced. The decision of rotation at each iteration is determined by the sign of z. Eqn. 6 represents the CORDIC algorithm in rotation mode.

The micro-rotational angles (ϕ) are precomputed and stored in a look-up table. Angles stored in look-up table are represented by b bits and normalized with respect to π. Weights assigned to each positions are in the order -π, $\pi/2$, $\pi/2^2$,... and $\pi/2^{(b-1)}$. Accumulating these micro-rotational angles, limits the range between $(\pi/2)$ and $(-\pi/2)$. So, conventional CORDIC can be operated in 1^{st} and 4^{th} quadrant. Angles in 2^{nd} and 3^{rd} quadrant are pre-rotated to place them in 4^{th} and 1^{st} quadrant respectively. So, an initial rotation of $(\pm\pi)$ is made.

III. CORDIC Architecture

CORDIC processor can be implemented in many ways. These architectures differ on the basis of speed and area constraints. This section presents the basic architecture followed by pipelined architecture.

A. Basic CORDIC Architecture

Architecture of basic rotation mode CORDIC is as shown in Fig. 2. Principal modules of this design are counter, adder/subtractor, shifters, multiplexers and a ROM. This basic CORDIC structure is sequential in nature. This design contains three similar units to execute Eqn.6. Each of these units contain one multiplexer, one adder/subtractor block and a register to store previous iteration result. Iterations are obtained through simple up-counter. Counter keeps track of the current iteration, degree of shift and accessing ROM address. Decision for clock wise or anti-clockwise rotation obtained from the sign bit of the error angle *z*. Multiplication of 2^{-i} with *x* or *y* is equivalent to shift the same right by *i* bits. To update the present iteration *x* or *y* values, right shift operation is performed on both the values obtained from the previous stage. These initial values are loaded through a multiplexer and its selection is controlled via counter output. ROM stores the elementary angles and accessed through iteration count sequentially. These angular values are presented to the adder/subtractor block along with the previous iteration value for angle computation.

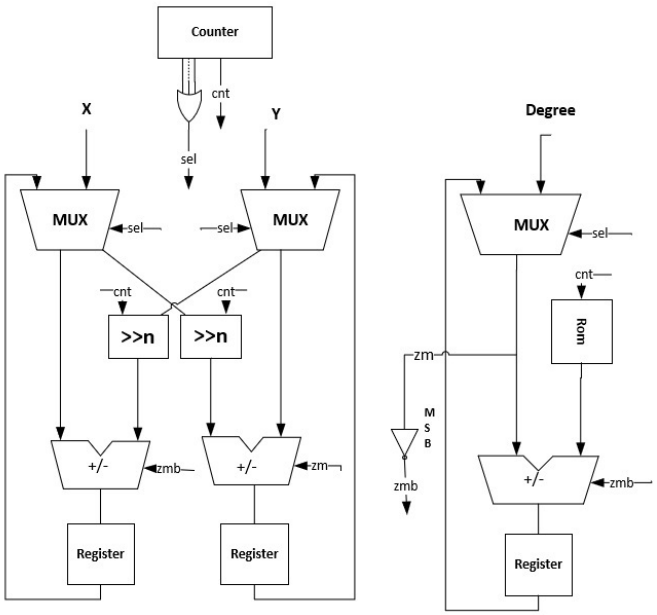

Figure 2: Basic CORDIC Architecture

Main problem with this design is its slow speed of operation. Shifters need several layers of logic and requires storage memories like ROM. As a result, it will require a large number of logic cells. For each *n* bit input, this architecture requires *n* clock cycles to provide the desired output which degrades the throughput. This can be overcome by using pipelined architecture.

B. CORDIC Pipelined Architecture

In the pipelined architecture, the iteration process is unfolded that defines unique shifter for each iteration. In this architecture, the shifters perform fixed shift which means it can be implemented by hardwiring. This in turn eliminates the need for iteration counter. Also, requirement for storing elementary angles in ROM is avoided by directly providing the same to individual processing elements. The lookup tables for angle accumulator are therefore distributed as constants to each adder-subtractor in the angle accumulator chain. The constants can be hardwired instead of storing them in memories. Here, the first output results after *n* clock cycles and then the next stream of data output is computed in the successive clock

Table I: Performance Model of CORDIC Pipelined Architecture

Angle	x	y	z	x_m	y_m	z_m	x_{err}	y_{err}	z_{err}
40	1.2607	1.0635	0.0009	1.2615	1.0585	0.0004	-0.0008	0.0050	0.0005
50	1.0581	1.2617	-0.0009	1.0585	1.2615	-0.0004	-0.0004	0.0002	-0.0006
85	0.1440	1.6396	0.0000	0.1435	1.6405	-0.0004	0.0005	-0.0009	0.0004
100	0.2856	-1.6216	-0.0048	0.2860	-1.6217	-0.0013	-0.0004	0.0001	-0.0035
135	1.1641	-1.1646	-0.0005	1.1644	-1.1645	0.0009	-0.0003	0.0000	-0.0014
170	1.6221	-0.2861	0.0000	1.6217	-0.2860	0.0013	0.0004	-0.0001	-0.0013
200	1.5464	0.5644	0.0004	1.5475	0.5632	0.0009	-0.0011	0.0012	-0.0004
225	1.1645	1.1636	0.0004	1.1645	1.1644	0.0009	0.0000	-0.0008	-0.0004
240	0.8232	1.4263	0.0005	0.8234	1.4261	0.0008	-0.0002	0.0002	-0.0003
275	0.1416	-1.6396	0.0005	0.1435	-1.6405	0.0017	-0.0019	0.0009	-0.0012
300	0.8237	-1.4263	0.0000	0.8234	-1.4261	-0.0008	0.0003	-0.0002	0.0008
350	1.6220	-0.2861	0.0000	1.6217	-0.2860	0.0013	0.0003	-0.0001	-0.0013

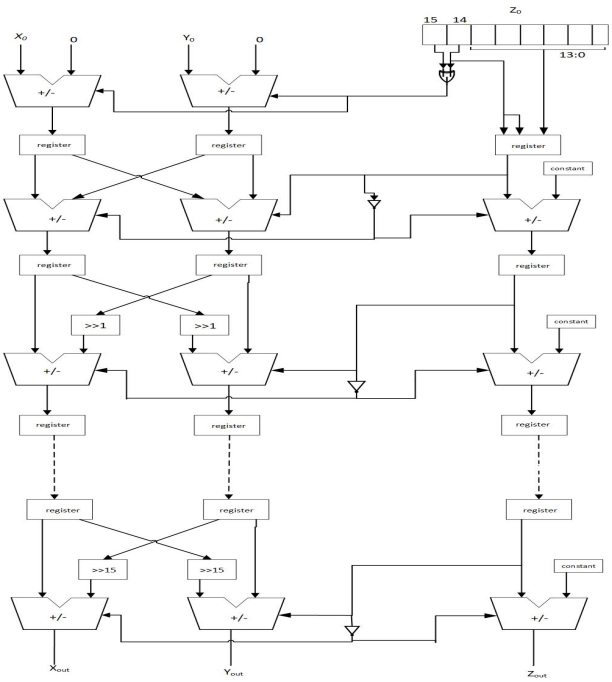

Figure 3: Pipelined CORDIC Architecture

cycle. Hence it has high throughput. The pipelined CORDIC architecture is shown in Fig. 3.

In this architecture, the XOR gate along with the first stage of adder-subtractor will determine whether pre-rotation is required according to the input angle. Then the output is fed to the next successive stage and the result is computed in each stage.

IV. Results and Analysis

This section analyzes both serial and pipelined CORDIC architectures operating in rotation mode. Both the designs are implemented in Xilinx ISE platform with Artix-7, xc7a100tcsg324-1 FPGA. Additionally MATLAB simuation for CORDIC algorithm has been performed to verify the hardware result.

In this work, *x, y, z and* ϕ are expressed with 16 bit precision. Results obtained from ISIM simulator is compared with the values resulted from MATLAB (x_m, y_m and z_m). Table I presents the comparison and shows the difference. The x_{err}, y_{err} and z_{err} represents the difference between them after 16 iterations. This difference can further be minimized with increasing bit precision.

Serial CORDIC design Fig.2 reuses the same building blocks (i.e. MUX, adder/subtractor and registers) to carry out computations for all iterations. Hence, this design have the area advantage over pipelined architecture Fig.3. However, the shifter block shifts the *x* and *y* values by *n* bits at each clock cycle. As *n* changes at each iteration logic behind the shifter changes. This masks over the area advantage obtained by serial design. On the contrary, in the pipelined CORDIC design, register stages are added between each iteration stages. This in turn reduces the critical path and improves the maximum operating frequency. Table II summarizes area utilization for both the designs and compares speed of operation.

Table II: Area Utilization summary and speed comparison

Architecture	LUT	FF	IO	Maximum freq
Serial	207	67	98	253.357MHz
Pipelined	674	739	97	340.414MHz

V. Conclusion

In this paper, VLSI architecture for a serial and a pipelined CORDIC is presented and analyzed. The serial architecture is efficient in terms of area utilization. While speed of computation is of prime importance, the pipelined architecture is faster and more efficient. This design inherits high throughput and less latency for a continuous stream of input.

References

[1] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on electronic computers*, no. 3, pp. 330–334, 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *Proceedings of the May 18-20, 1971, Spring Joint Computer Conference*. ACM, 1971, pp. 379–385.

[3] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "High performance rotation architectures based on the radix-4 cordic algorithm," *IEEE Transactions on Computers*, vol. 46, no. 8, pp. 855–870, 1997.

[4] N. D. Hemkumar and J. R. Cavallaro, "Redundant and on-line cordic for unitary transformations," *IEEE Transactions on Computers*, vol. 43, no. 8, pp. 941–954, 1994.

[5] J. Duprat and J. . Muller, "The cordic algorithm: new results for fast vlsi implementation," *IEEE Transactions on Computers*, vol. 42, no. 2, pp. 168–178, Feb 1993.

[6] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of cordic: Algorithms, architectures, and applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1893–1907, 2009.

[7] O. Mencer, L. Semeria, M. Morf, and J.-M. Delosme, "Application of reconfigurable cordic architectures," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 24, no. 2-3, pp. 211–221, 2000.

[8] B. Lakshmi and A. S. Dhar, "Vlsi architecture for parallel radix-4 cordic," *Microprocessors and Microsystems*, vol. 37, no. 1, pp. 79–86, Feb. 2013.

[9] T. Kulshreshtha and A. S. Dhar, "Cordic-based hann windowed sliding dft architecture for real-time spectrum analysis with bounded error-accumulation," *IET Circuits, Devices Systems*, vol. 11, no. 5, pp. 487–495, 2017.

[10] C.-H. Lin and A.-Y. Wu, "Mixed-scaling-rotation cordic (msr-cordic) algorithm and architecture for high-performance vector rotational dsp applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 11, pp. 2385–2396, 2005.