International Journal of Computer Applications (0975 - 8887)
Volume 63 - No. 14, February 2013

A Novel Scaling free Vectoring CORDIC and its FPGA Implementation

Anita Jain
Research Scholar
Electronics Department
MANIT Bhopal India

Kavita Khare
Associate Professor
Electronics Department
MANIT Bhopal India

Supriya Aggarwal
Asst Professor (contract)
Electronics Department
MANIT Bhopal India

ABSTRACT

This research paper proposes a novel scaling free CORDIC algorithm to operate in vectoring mode which computes absolute magnitude and phase angle of input vector. Using this algorithm, the micro rotation of the vector is unidirectional and totally scaling free. The range of convergence is successfully extended to cover entire coordinate space without increasing any hardware complexity.Further a 16 bit Scaling free vectoring CORDIC architecture based on this proposed algorithm is synthesized on FPGA Xilinx Virtex-5 device using Verilog hardware description language. Synthesized results show throughput every clock cycle with maximum operating frequency of 243.55MHz and demonstrate very low dynamic power consumption.

General Terms:

CORDIC algorithm, Digital signal Processesing,Vector rotation, FPGA.

Keywords:

Vectoring CORDIC, Scaling free, Quadrant Mapping, Sectors, Range of convergence ROC, Pipeline architecture.

1. INTRODUCTION

CORDIC is an acronym for **CO**ordinate **R**otation **Di**gital **C**omputer. It is a simple and hardware-efficient algorithm for the implementation of various trigonometric functions. It uses simple shift, add, subtract and table look-up operations instead of calculus based methods such as polynomial or rational functional approximation used traditionally. The trigonometric functions are used in many applications including real time digital signal processing (DSP), wireless communication, robotics, computer graphics, navigation and astronomy. In the field of DSP, it is used for calculation of various transforms such as fast fourier transform (FFT), discrete sin/cosine transform (DST/DCT), discrete hartley transform (DHT) and Hough transform (HT) etc. In digital communication, it is used to generate signals during modulation and to estimate phase and frequency parameters during demodulation. Its efficiency in Matrix Computation make it attractive choice for application involving QR decomposition, Singular Value Decomposition (SVD), estimation of Eigens values and vectors etc. In antenna systems it is used for estimation of Direction of Arrival (DOA), Multiple Input Multiple Output (MIMO) detectors etc. It is also used in 3D graphics as vector interpolator [3]. CORDIC has found a significant place even in upcoming technologies like Cognitive radio and Software Define Radio (SDR) [8].

The CORDIC algorithm was first proposed by [9] on the basis of Givens Rotation of vectors in two dimensional space. Since then it has been subjected to continuous development in terms of algorithmic change or architecture variations to achieve higher and higher throughput rate, lesser and lesser hardware-complexity and latency.The path of development of CORDIC is beautifully summarized in [6].

The fundamental idea behind the CORDIC is to carry out a sequence of rotations on two-dimensional vectors using a series of specific incremental rotation angles selected such that each is performed by a shift and add operation. It is relatively simple in design and VLSI implementation, as no multipliers are required.

CORDIC Algorithm can operate in two modes namely: rotation and vectoring. In rotation mode, the objective is to rotate a given vector from its initial position to the final position which is at target angle θ, through a series of iterations. The rotation decision at each iteration is made to reduce the magnitude of the residual angle to zero. The rotation trajectory can be linear, circular or hyperbolic depending upon the requirement. In vectoring mode, the aim is to find out magnitude and argument of a vector. The vector is rotated from its initial value to its final value so as to reduce the y component to zero. The magnitude of the vector will get stored in the x component and the angle accrued due to such rotations will be stored in the z register representing its phase.

As indicated by [6], a lot of research work has been done for reducing the complexity and increasing the performance of rotation mode of CORDIC . Various techniques and hardware architectures have been used for the same. Yet there is an ample scope of optimization for vectoring mode. The conventional Vectoring mode CORDIC has a major drawback of generating a bulk scale factor that needs to be compensated using extra circuitry thus increases the hardware cost significantly. This paper proposes a novel algorithm for vectoring mode of CORDIC which is totally scaling free with a provision for skipping iterations not actually needed so as to speed up the operation. Unlike the conventional Vectoring CORDIC, the rotation of vector in the proposed algorithm is always in one direction and has convergence range extended over entire coordinate space. It also has eliminated the need of lookup table and has significantly saved the memory area.

The rest of the paper is structured as follows: Section 2 briefly presents the CORDIC algorithm overview.The proposed algorithm is discussed in Section 3. Section 4 suggests the architec-

ture for the proposed algorithm. Section 5 details the FPGA implementation and comparison and section 6 concludes the paper.

2. CORDIC ALGORITHM OVERVIEW

The basic principle of CORDIC algorithm is to iteratively rotate a vector in a plane by simple shift and add operation to compute either phase and magnitude of a vector or sine and cosine of an angle in circular trajectory. Various other trigonometric, hyperbolic and linear function can also be computed efficiently based on this basic principle.

2.1 Conventional CORDIC algorithm

The conventional CORDIC algorithm [9] is derived from general equation of vector rotation. If a vector V with components (X_i, Y_i) is iteratively rotated through an angle ϕ_i, a new vector V' with components (X_{i+1}, Y_{i+1}) is formed. In matrix form, the value of vector after this microrotation can be represented as :

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} X_i.cos\phi_i - Y_i.sin\phi_i \\ X_i.sin\phi_i + Y_i.cos\phi_i \end{bmatrix} \tag{2.1}$$

rearranged as :

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = cos\phi_i \begin{bmatrix} 1 & -tan\phi_i \\ tan\phi_i & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{2.2}$$

The multiplication by the tangent term can be avoided if the microrotation angle tan ϕ_i is restricted to 2^{-i}, i.e

$$\phi_i = \arctan(2^{-i}) \tag{2.3}$$

In digital hardware 2^{-i} denotes a simple binary shift operation. If the microrotations are performed with variable direction d_i every iteration, the equation 2.2 can be rewritten as :

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = K_i \begin{bmatrix} 1 & -d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{2.4}$$

where $K_i = cos(arctan(2^{-i}))$ and $d_i = \pm 1$. The product of the K_i's represents the K factor or Scaling factor :

$$K = \prod_{i=0}^{w-1} K_i = \prod_{i=0}^{w-1} 1/\sqrt{1+2^{-2i}} \tag{2.5}$$

The direction of each rotation is defined by d_i and the sequence of all d_i's determines the final vector. This yields to a third variable Z_{i+1} as shown in 2.6 which acts like an angle accumulator and keeps track of angle already rotated.

$$Z_{i+1} = Z_i - d_i.\arctan(2^{-i}) \tag{2.6}$$

The input angle of rotation ϕ is achieved through the summation of all microrotations ϕ_i with appropriate direction.

$$\phi = \sum_{i=0}^{w-1} d_i.\arctan(2^{-i}) \tag{2.7}$$

where w is the word length and the values of $\arctan(2^{-i})$ are pre-calculated and stored in a lookup table during implementation.

The conventional CORDIC algorithm can be made to operate in either rotation or vectoring mode depending upon the way of determining the direction of microrotation. In rotation mode the coordinates of the vector and an angle of rotation is given, and the coordinates of the original vector, after rotation through a given angle, are computed. The rotation decision at each iteration is made to reduce the magnitude of the residual angle in the angle accumulator to zero and thus d_i is computed on the basis of sign of Z_i.

$$d_i = \begin{cases} +1 & \text{if } Z_i \geq 0 \\ -1 & \text{if } Z_i < 0 \end{cases} \tag{2.8}$$

Where as, in vectoring mode the coordinates of the vector are given and the magnitude and angular argument of the original vector are computed. In this mode, the y component is minimized to zero and it is the sign of Y_i which decides the direction of microrotation.

$$d_i = -\text{sign}(Y_i) \tag{2.9}$$

In conventional CORDIC algorithm, if some iterations are skipped or repeated to achieve larger or faster convergence range, the scale factor will vary and will require extra circuitry and clock cycle for its compensation. Thus it suffers from limitation of slow speed, small convergence range and bulk scale factor compensation circuitry.

2.2 Scaling free CORDIC Algorithm for Rotation mode

To wipe off the effect of variable scaling factor and associated complex circuitry, the Scaling free CORDIC algorithms were developed. Use of Taylor series approximation of sine and cosine functions form the basis of making scaling free CORDIC. In [4] a modified virtually scaling free algorithm is proposed whereas in [2] enhanced version of modified virtually scaling free CORDIC is suggested using booth recoding and conventional CORDIC to make it scaling free. Recent research in [1] uses third order approximation of the series together with high speed most significant-1 detection scheme.

The Taylor series expansion of sine and cosine of an angle α is given by:

$$\begin{aligned} \sin\alpha &= \alpha - (3!)^{-1}.\alpha^3 + (5!)^{-1}.\alpha^5 + ... \\ \cos\alpha &= 1 - (2!)^{-1}.\alpha^2 + (4!)^{-1}.\alpha^4 + ... \end{aligned} \tag{2.10}$$

First order approximation for sin and cosine series was used in [4] during the implementation, reducing the series as:

$$\begin{aligned} \sin\alpha &\cong \alpha \\ \cos\alpha &\cong 1 - 2^{-(2i+1)} \end{aligned} \tag{2.11}$$

But, this approximation imposes a restriction on the allowed values of iterations i as:

$$\lfloor (w - 2.585)/3 \rfloor \leq i \leq w - 1 \tag{2.12}$$

for 16 bit data, the value of i comes out to be 4.

There is no complete scaling free version of vectoring CORDIC available as yet. However attempts were made in designing virtually constant scaling factor vectoring CORDIC in [5]and [7]. But it still requires circuitry for fixed scale factor multiplication.

3. PROPOSED ALGORITHM FOR SCALING FREE VECTORING CORDIC

The proposed algorithm for scaling free vectoring CORDIC uses third order approximation of Taylor series and divided the coordinate space into eight equal sectors, each of 45 degrees as shown in figure 1 based on minimum value of i as 2. With this division the range of convergence is (0,π/4). To cover entire coordinate space i.e for the vectors lying in other sectors and quadrants, quadrant mapping is done. Vectors lying in other quadrants are mapped to first sector of first quadrant. This is done by pre rotating them in the clockwise direction. The

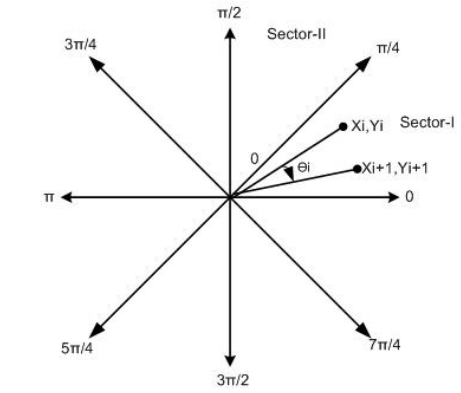

Fig. 1. Division of Quadrants and basic Rotation of vector

Table 1. Quadrant Demapping.

MSB's	Quadrant	Sector	Final angle Θf
00	I	I	θi
		II	(90-θi)
01	II	I	(180-θi)
		II	(90+θi)
10	III	I	(180+θi)
		II	(270-θi)
11	IV	I	(360-θi)
		II	(270+θi)

direction of all microrotation is clockwise. The pre rotation operation also ensures appropriate modification of x and y coordinates. Following pseudo- code for pre rotation of the input vector is compiled as:

Pseudo code for Quadrant Mapping
Input xin,yin;
output xi,yi;
Begin

if $|xin| > |yin|$ **then**
$xin = |xin|$;
$yin = |yin|$;
$Sector = I$;

else
$xin = |yin|$;
$yin = |xin|$;
$Sector = II$;
end if
End

The magnitude of x and y coordinates determines the sector of the vector whereas their signs indicates the respective quadrant.

Quadrant demapping is performed to reverse the effect of pre rotation and finally to get the phase angle of the vector. Rule for demapping is listed in table 1.

The essentials steps of the algorithm can be summarized as follows:

(1) Input the x and y coordinates of the vector. Depending on the magnitude and sign of the coordinates, determine the sector and quadrant of the vector.

(2) Map the vector to the sector I of the first quadrant as per the pseudocode.

(3) Compute the next value of the vector by rotating it in clockwise direction using basic shift and add operation.

(4) Store 0 in the angle accumulation register if sign of y vector changes, skip that particular iteration or else store 1 in the register and go for next iteration.

(5) Stop the process of microrotation after performing iterations designed to reduce y coordinate to zero.

(6) Demapped the value stored in angle accumulation register to get final phase angle.

(7) Get the value of absolute magnitude from the X register.

Flow chart for the above algorithm can be graphically represented as figure 2.

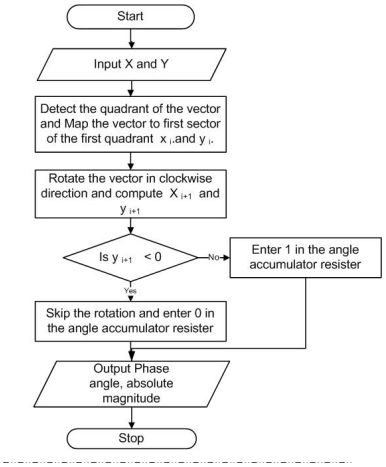

Fig. 2. Flow chart of proposed Algorithm

4. ARCHITECTURE OF THE PROPOSED SCALING-FREE VECTORING CORDIC

Architecture of the proposed Scaling-free vectoring CORDIC is shown in the block diagram of figure 3. Input to the CORDIC module consists of x and y coordinate of the vector whose magnitude and phase angle is to be determined. These inputs are stored in the X and Y registers. Fixed point 16 bit representation is used for X and Y inputs. The angle accumulator register is initialized to zero in the beginning. The inputs X and Y are given to quadrant mapping block which not only determines the quadrant and sector of the given combination but also maps it to the first sector of the first quadrant. This transformed vector is applied to basic CORDIC pipeline as shown in figure 4.

This CORDIC pipeline rotates the vector in clockwise direction and compute its new value. This computation is simple shift and add operation performed for particular value of iteration index i. Depending upon the sign of computed y vector, a multiplexer is used to either accept the iteration or to skip it. If the iteration is skipped 0 is stored in angle accumulator register or else 1 is entered.

Processing of the vector from the pipeline stages results in:

(1) X register holding the value of magnitude of the vector, which does not require any further processing, as scale factor is one i.e it is completely scaling free vectoring CORDIC.

(2) Zero value of Y register.

(3) angle accumulator register value which needs to be modified as per the quadrant demapper to get the final phase angle.

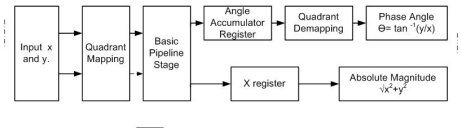

Fig. 3. Block diagram of Scaling-free Vectoring CORDIC

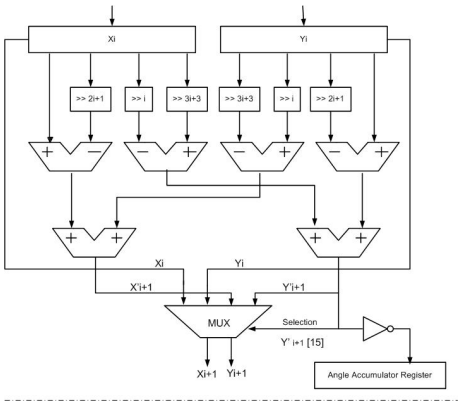

Fig. 4. Basic CORDIC Pipeline Architecture (for i= 2 to 6)

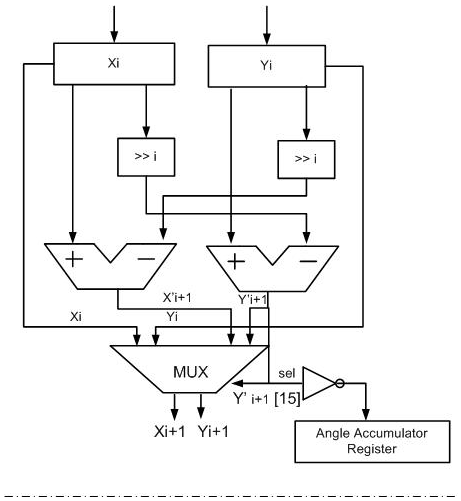

Fig. 5. Basic CORDIC Pipeline Architecture (for i= 7 to 14)

4.1 Basic CORDIC Pipeline Architecture

Basic CORDIC pipeline consists of thirteen stages from i = 2 to 14.

Stages from i = 2 to 6 require six adders as shown in figure 4 whereas for stages i = 7 to 14 ,the requirement of adders reduces to only two, as shown in figure 5 . Shifters for each stage are simple wired connection and does not add to the complexity of the circuit.

5. FPGA IMPLEMENTATION OF THE VECTORING CORDIC

Implementation of the proposed architecture is carried out in Xilinx ISE9.2i targeting virtex5 device using Verilog hardware description language. The output data rate is one set of data per clock period except for the first data set which is obtained after completely filling the pipeline.

The power dissipation of the proposed design for different clock frequencies is computed using XPower tool of Xilinx and is plotted in fig 6. Total power dissipation is 264.85 mW.Maximum operating frequency is found to be 243.55 MHz with maximum output required time after clock is 5.513ns.

Table 2. Experimental Results.

CORDIC	Number of full adders	Number of registers	Scaling Factor
Conventional	768	768	Fixed
Dawid	512	1280	Yes
Maharatna	816	553	$1,1/\sqrt{2}$
Proposed	832	462	NO

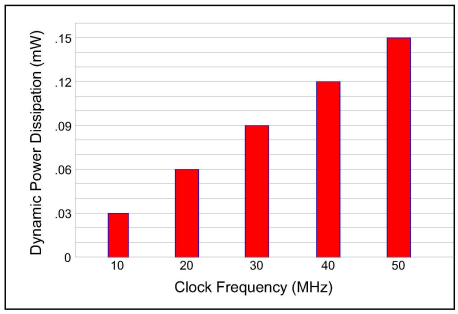

Fig. 6. Dynamic Power Dissipation for the proposed CORDIC

Hardware Utilization on the target device is as shown in table 3 Functional simulation of the proposed design is successfully carried out using test bench waveform option of the Xilinx tool.The simulation results are shown in fig 7 . The simulation values are verified mathematically as well.

Table 3. Hardware Utilization

Device Parameter	No Usage	No Available	% Utilization
No of Slice Registers	462	19200	2%
No of Slice LUTs	1482	19200	7%
No of bonded IOB	66	400	16%
No of BUFG	1	32	3%

6. CONCLUSION

The proposed algorithm successfully realizes completely scale-free vectoring CORDIC . It does not require any complex pre and post processing circuitry. The approximation used here for sin and cosine series has not only increased the accuracy of the processor but also expanded the range of convergence to complete coordinate space. It has also completely eliminated the use of lookup tables for implementation of CORDIC. The hardware requirement is also less as compared to other designs.

7. REFERENCES

[1] Supriya Aggarwal, Pramod K. Meher, and Kavita Khare. Area-time efficient scaling-free cordic using generalized micro-rotation selection. *IEEE Transactions on VLSI Systems*, 20(8):1542–1546, Aug 2012.

[2] F. J. Jaime, M. A. Sanchez, J. Hormigo, J. Villalba, and E. L.Zapata. Enhanced scaling-free cordic. *IEEE Transactions on Circuits and Systems*, 57(7):16541662, 2010.

[3] T. Lang and E. Antelo. High-throughput cordic-based geometry operations for 3d computer graphics. *IEEE Transactions on Electronic and Computers*, 54(3):347–361, Mar 2005.

[4] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya. Modified virtually scaling-free adaptive cordic rotator algorithm and architecture. *IEEE Trans. Circuits Syst. Video Technol.*, 11(11):1463–1474, Nov 2005.

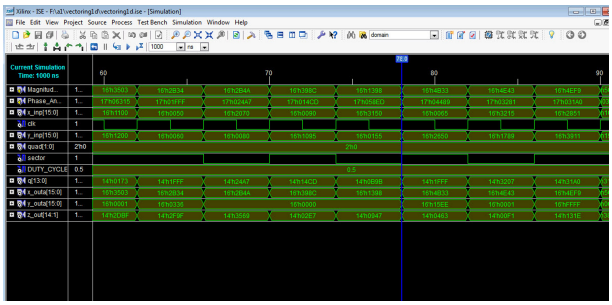

Fig. 7. Simulation Waveform of the proposed CORDIC

[5] K. Maharatna, A. Troya, M. Krstic, E. Grass, and U. Jagdhold. A cordic like processor for computation of arctangent and absolute magnitude of a vector. In *Proc. ISCAS*, volume 2, pages 713–16, 2004.

[6] Pramod Kumar Meher, Javier Walls, Tso-Bing Juang, K. Sridharan, and Koushik Maharatna. 50 years of cordic : Algorithms and architectures and applications. *IEEE Transactions on Circuits and Systems I*, 56(9):1893–1907, Sept 2009.

[7] R. Stapenhurst, K. Maharatna, J. Mathew, J.L. Nunez-Yanez, and D.K. Pradhan. A cordic like processor for computation of arctangent and absolute magnitude of a vector. In *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2007*, pages 3002–3005, May 2007.

[8] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar. The use of cordic in software defined radios: A tutorial. *IEEE Communication Mag.*, 44(9), 2006.

[9] J. E. Volder. The cordic trigonometric computing technique. *IEEE Transactions on Electronic and Computers*, 8:330–334, 1959.