A PARALLEL AND PIPELINED ARCHITECTURE FOR CORDIC ALGORITHM

Dr. Ellapan V,
Department of Electronics and Communication Engineering,
School of Electrical Engineering and Computing,
Adama Science and Technology University,
Adama, Ethiopia, Africa
ellappan.venugopal@astu.edu.et

Dr. Sam Alaric J,
Department of Electrical and Computer Engineering,
Institute of Technology, Wollega University,
Ethiopia, Africa
samalaric@gmail.com

Abstract: The COordinate Rotation DIgital Computer (CORDIC) algorithm is an efficient algorithm to calculate the iteratively phase and magnitude or the vector rotations in linear, hyperbolic and circular coordinate system. The existing CORDIC method takes less clock frequency with high delay. To overcome this problem, a new version of updated parallel and pipelined architecture is designed without degrading the performance. It provides highest maximum frequency with less delay by splitting the critical path into several smaller delay paths with enhanced circuit processing time. The designed architecture in this study can be used in navigation application. This method is implemented in the Xilinx ISE tool.

Keywords: CORDIC Algorithm, parallel and pipelined method, Xilinx tool.

I. INTRODUCTION

In June 1956, the CORDIC algorithm and a computerized design for executing CORDIC algorithm introduced to Convair management as a technical report. During the preparation of the report, it was realized by VOLDER that the same computerized design could be comfortably alter to generate, logarithmic functions, hyperbolic coordinate rotation and exponential expressions. In [1] different characteristics of CORDIC algorithm are described. Also, it is implemented in the Field Programmable Gate Array (FPGA) processor. In [2] CORDIC rotator algorithm is described by setting the scale factor constant in order to perform the iteration. It offers 50% reduction in iteration.

In [3], a modified CORDIC algorithm with a new attractive Fast Fourier Transform (FFT) is described. It is used in the opposite ends of the computer power spectrum. In [4], the architecture of FPGA implementation and optimization measures is described according to hardware sources, angular coverage and computing precision of the algorithm. The speed and accuracy of this algorithm is high.

In [5], a serial pipelined FFT on FPGA using CORDIC algorithm is described. To enhance the performance of FFT, it utilizes the pipelined structure, dual port structure and radix-2 decimation in time. In [6], three reconfigurable CORDIC designs; CORDIC that works either hyperbolic circular in rotation mode, CORDIC that works both hyperbolic and circular in vectoring mode and CORDIC that work in both hyperbolic and circular in any mode are described.

In [7], a design of direct digital synthesizer utilizing CORDIC algorithm is discussed. It is programmed into FPGA for verification. In [8], an enhanced mixed scaling rotation CORDIC algorithm is described. It offers higher signal to noise ratio performance by amplifying the factor by multiplying the rational sequence to the equivalent signed-power-of-two conditions.

In [9], a reduced memory CORDIC architecture with pipeline is described. It can be used for any radix size FFT and this avoid the storing the angles and twiddle factors. In [10], a complex 128 point FFT processor utilizing rolling back and parallel method is described. This method provides fast speed with low power.

In [11], a power of two point discrete cosine transform based CORDIC algorithm is described. It overcomes the lack of synchronization problem by reusing the uniform processing cell. In [12], a FFT design using radix 2/4/8 with single path delay feedback structure is described. It includes complex multipliers that contains 3 real multiplications and decreased cosine/sine tables.

In this paper, a parallel-pipelined architecture for CORDIC algorithm is presented. The organization of this paper is as follows: The methods and materials used in this study are explained in section 2. Section 3 gives the results and discussion and section 4 describes the conclusion.

II. METHODS AND MATERIALS

The hardware implementation of CORDIC arithmetic is shown in Fig. 1. It consists of three inputs *X, Y* and *Z* and also a look up table to store the values of *tan-12-i* and two shifters to supply the values *2-iX* and *2-iY*. Here all the multiplication operations are converted to simple shift operations. At the start of a calculation of initial values that are fed into the register by the multiplexer wherever the MSB of the stored value within the *Z*-branch determines the operation mode for the adder-subtractor. The signals of *X* and *Y* branch passes the shift units and then they are added to or subtracted from different path of non-shifted signals. Sine and Cosine waveforms are directly given by the CORDIC algorithm which acts as a quadrature phase-to-amplitude converter [13].

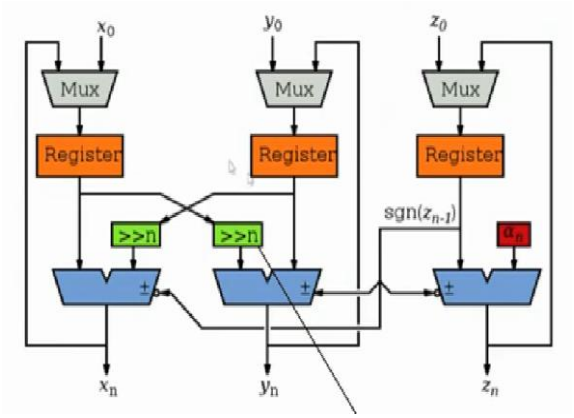

Fig.1. Iterative Architecture of CORDIC

The hardware structure of the parallel CORDIC architecture is shown the Fig. 2. It is a data-driven circuit. The numbers of iterations performed is based on the precision of the bits required. Thus '*n*' number of iterations is required for '*n*' bits of precision. Moreover, '*n-1*' sets of are required for '*n*' bits of precision. So,

3(*n-1*) adder/subtractor circuit, *2 (n-1)* Shifters circuits are required. The various components required in one set or each iteration is:

- 3 Adders/Subtractors
- 2 Shifters
- 1 look-up-table

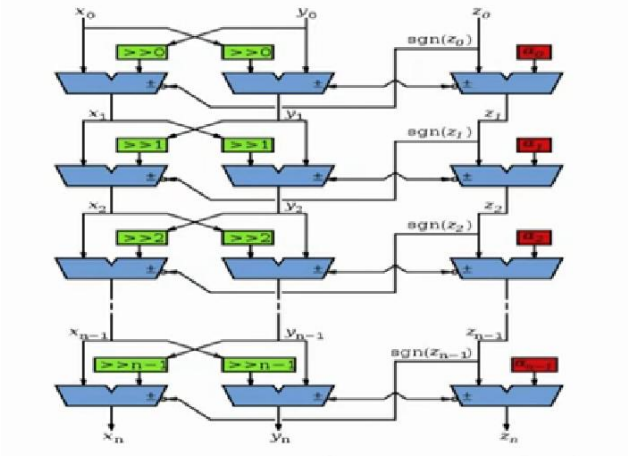

Fig.2 Parallel Architecture of CORDIC

The proposed parallel pipelined architecture is shown in Fig. 3. Previously the circuit was dependent on data-driven property. But now with the presence of register in middle of stages, it has changed to clock driven based circuit. Thus, the stages are now independent and are not adjacent.

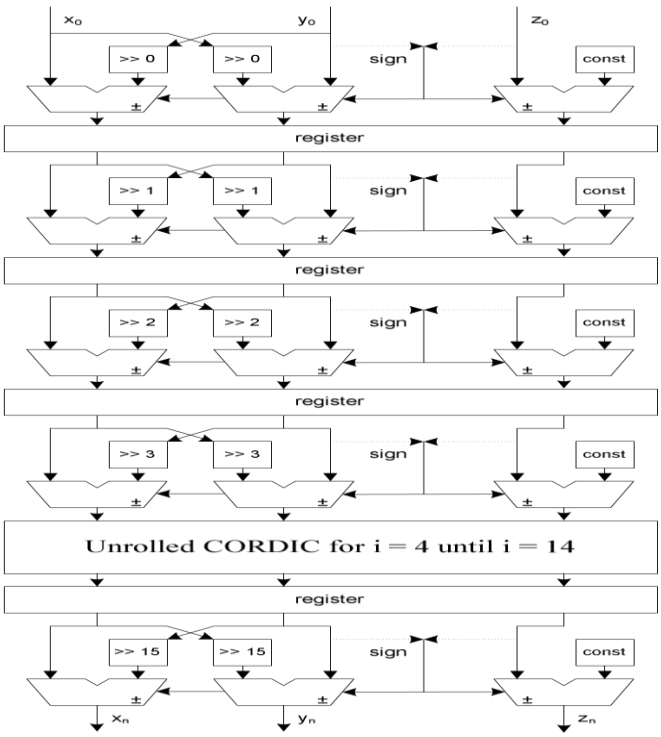

Fig. 3 Parallel- Pipelined Architecture of CORDIC

If the first iteration begins at first clock cycle, the second iteration begins at second clock cycle, then third at third cycle and so on. As a result, the delay gets reduced and the speed of computation increases. But due to the registers the area gets increased. Consider a unit length vector with one end point at vertex and other at $(X,Y)=(1,0)$. If this is rotated by an angle θ, its new point will be $(X,Y)=(Cos\theta,Sin\theta)$ in Fig. 4. Thus, $Cos\theta$ and $Sin\theta$ can be calculated by finding the co-ordinates of the new point [14].

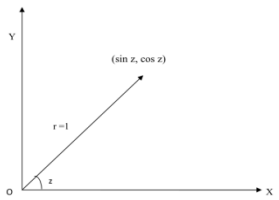

Fig. 4 Vector in Cartesian coordinates

In case of the vector length not equal to unity and it is rotated by an angle θ and the new coordinates of the point $X(i+1),Y(i+1)$ in Fig. 5 of the vector after rotation is given by the Cartesian geometry formulas:

$$X(i+1)=X_i Cos\theta+Y_i Sin\theta \quad (1)$$

$$Y(i+1)=X_i Sin\theta+Y_i Cos\theta \quad (2)$$

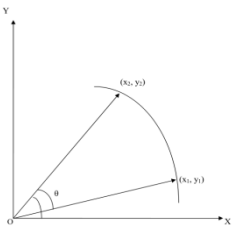

Fig. 5 Rotation of a vector

It is clear in the above equations that the $Cos\theta$ term provides scaling, which means it reduces the magnitude of the vector as $Cos\theta \le 1$. So, by removing the $Cos\theta$ term from the above equations, the magnitude of the vectors is getting increased by the factor $Cos-1\theta$ as shown in Fig. 6.

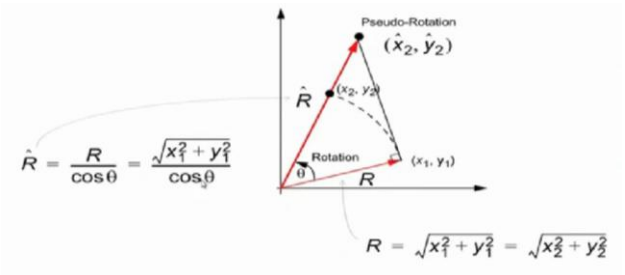

Fig. 6 Pseduo Rotations

With the increase of i values, the values of $Tan\theta_i$ and θ_i goes on decreasing. After each iteration the gets added or subtracted to the angle accumulator. Let z represents the angle accumulator.

$$Z(i+1)=Z_i - d_i\theta_i \tag{3}$$

Thus, the equations become

$$X(i+1)=X(i)-d_i(Y(i)2-i) \tag{4}$$

$$Y(i+1)=Y(i)+d_i(X(i)2-i) \tag{5}$$

Here the di term act as the deciding factor to perform the addition and the subtraction operation in the equations. The value of d_i is nothing but the sign of the $Z(i+1)$.

The different modes of CORDIC algorithms [15] are used to calculate different functions. There are two (2) modes: Rotation and Vector

- ***CORDIC Rotation Mode***

In this, the sign of d_i depends on the sign of the $Z(i+1)$ and this which makes 'z' converge to 0, and it is known as 'rotation mode'.

$$X(i+1)=X(i)-d_i(2-iY(i)) \tag{6}$$

$$Y(i+1)=Y(i)+d_i(2-iX(i)) \tag{7}$$

$$Z(i+1)=Z(i)-d_i\theta \tag{8}$$

In conventional CORDIC these angles are to form all other angles 45, 26.6, 14, 7.1, 3.6, 1.8, .9, 0.4. Taking 30 degree as an example as mentioned in Table 1 and described in Fig. 7.

TABLE 1: Iteration values for rotation mode for 30°

Iterations(i)	d_i	θ_i	Z_i	Y_i	X_i
0	+1	45	+30	0	0.6073
1	-1	26.6	-15	0.6073	0.6073
2	+1	14	+11.6	0.3036	0.9109
3	-1	7.1	-2.4	0.5313	0.8350
4	+1	3.6	+4.7	0.4270	0.9014
5	+1	1.8	+1.1	0.4833	0.8747
6	-1	0.9	0.7	0.5106	0.8596
7	+1	0.4	+0.2	0.4972	0.8676

30 = 45 -26.6+14 -7.1+3.6+1.8 -0.9+0.4
After n iterations,

$$X(n)=K(X\,Cos Z - Y\,Sin Z) \tag{9}$$

$$Y(n)K(Y\,Cos Z + X\,Sin Z) \tag{10}$$

$$Z(n)=0 \tag{11}$$

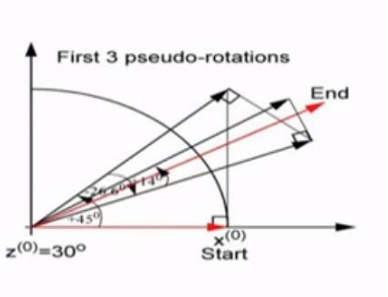

Fig.7 Rotations for 30 degrees

To avoid the storing of constant value of scaling constant in order to save area, we start with values as *X = 1/K [X = 0. 6072529 and Y = 0]*.For result with '*n*' bits of precision, 'n' CORDIC iterations are necessary [16]. As *Z(n)* tends to '0', *X(n)* and *Y(n)* tends to *CosZ* and *SinZ* . Range of angles covered is -99.7° ≤ z ≤ 99.7°.where 99.7° is total sum of all the angles in look-up-table.

- ***CORDIC Vectoring Mode***

The CORDIC equation becomes:

$$X(n)=K(X_2+Y_2)/2 \tag{12}$$

$$Y(n)=0 \tag{13}$$

$$Z(n)=Z+Tan-1\left(Y/X\right) \tag{14}$$

Figure 8 describes the vectoring mode of a point inclined at 30 degree and the iterative values of the process are mentioned in Table 2.

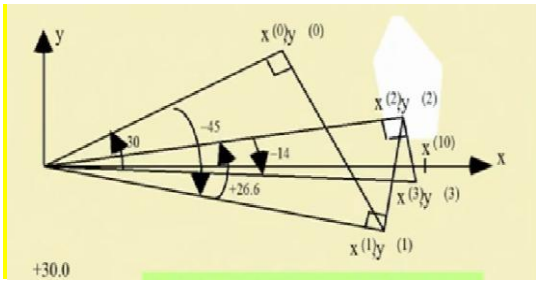

Fig.8 Vectoring mode for 30 degrees

TABLE 2. Iteration values in Vectoring mode

Iterations (i)	Z_i	Θ_i	Y_i
0	0	45	2
1	45	26.6	1
2	71.6	14	-0.5
3	57.6	7.1	0.375
4	64.7	3.6	-0.078
5	61.1	1.8	0.151
6	62.9	0.9	0.039
7	63.8	0.4	-0.019
8	63.4	0.2	0.009

To calculate $Tan-1Y$, at the beginning we take *X = 1 and Z = 0*

However, one can take advantage of below formula to limit the range of fixed-point numbers encountered.

$$Tan-1\left(1/Y\right)=\pi/2-Tan-1Y \tag{15}$$

III. RESULTS AND DISCUSSION

For verifying the results, a simulation in Xilinx of the CORDIC algorithm is realized and the output is shown in Fig. 9. In VHDL, all these implementations are designed with the help of ISE environment and ISIM simulator. The earlier circuits are synthesized by Xilinx spartan XC5VTX240T device.

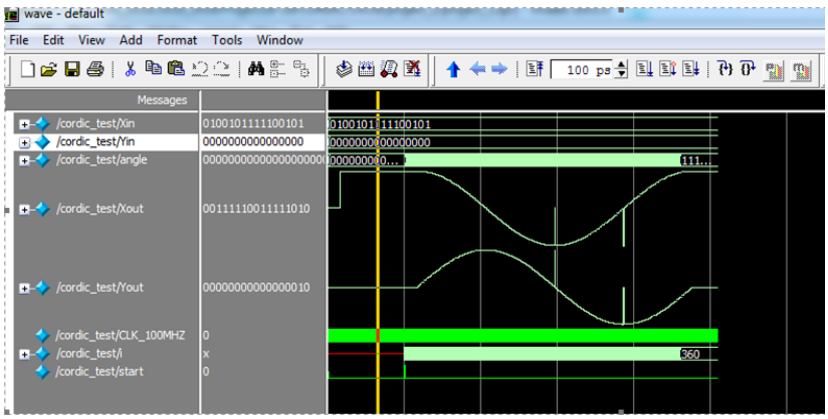

Fig. 9 Simulation Output

The generation of input vector is done in such a way that all the four quadrants gets covered and standardized to have a magnitude equal to 1. For 16-bit input, *A (1,6)* is the format of fixed point. As the CORDIC convergence range is restricted to $\left[-\pi/2,\pi/2\right]$, each input vector has to be rotated by an angle of $\pm\pi/2$, moving every vector to the forth and first quadrant such that the range can be increased. Table 3 gives the results of comparison among the three methods-sequential, parallel, pipelined and parallel methods.

TABLE 3. Comparison of 3 Architectures

S.NO.	METHOD	SLICES	LUT	FLIP FLOP	MAX CLOCK FREQUENCY
1.	**Sequential**	957	1809	952	362.588
2.	**Parallel**	964	905	884	448.762
3.	**Parallel- Pipelined**	977	902	873	464.563

It is inferred from Table 3 that the parallel-pipelined method has the highest maximum-clock frequency which is 464.563 (approx.). The length of the critical path of the circuit is reduced with the introduction of pipelining. Thus, this method has the lowest delay when compared to all three.

IV. CONCLUSION

In this paper, a parallel-pipelined architecture for CORDIC algorithm is discussed. Though there is an increase in area with the addition of registers, delay is reduced drastically. When compared to existing methods the parallel-pipelined architecture has the highest max-clock frequency. Since real time data acquisition is the need of the hour, this method has enormous scope in real time processing. It can be used in navigation applications, radar signal processors and unmanned aerial vehicle (UAV's) that require high computational speed. CORDIC is definitely a light at the end of the tunnel because they are used in super computers, which is an evolving technology.

REFERENCES

[1]. R. Andraka, "A survey of CORDIC algorithms for FPGA based computers", ACM/SIGDA sixth international symposium on Field programmable gate arrays, 1998, pp. 191-200.

[2]. K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture", IEEE Transactions on circuits and systems for video technology, Vol. 15, No. 11, 2005, pp.1463-74.

[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, 1997, pp. 855-70.

[4]. J. Li, J. Fang, B. Li and Y. Zhao, "Study of CORDIC algorithm based on FPGA", IEEE Chinese Control and Decision Conference, 2016, pp. 4338-4343.

[5]. A. Tang, L. Yu, F. Han and Z. Zhang, "CORDIC-based FFT real-time processing design and FPGA implementation", IEEE International Colloquium on Signal Processing & Its Applications, 2016, pp. 233-236.

[6]. S. Aggarwal, P.K. Meher and K. Khare, "Concept, design and implementation of reconfigurable CORDIC", IEEE Transactions on Very Large Scale Integration Systems, Vol. 24, No. 4, 2015, pp. 1588-92.

[7]. X. Zhang, X. Zhao and B. Zhou, "The Design of Direct Digital Synthesizer Based On Cordic Algorithm and FPGA Implementation", International Conference on Computer Engineering, Information Science & Application Technology, 2016, pp. 98-103.

[8]. J.M. Mehta and P. Trivedi, "An enhanced mixed-scaling-rotation CORDIC algorithm with weighted amplifying factor", IEEE International Conference on Digital Signal Processing, 2016, pp. 527-531.

[9]. X. Xiao, E. Oruklu and J. Saniie, "Reduced memory architecture for CORDIC-based FFT", IEEE International Symposium on Circuits and Systems, 2010, pp. 2690-2693.

[10]. G. Zhang and F. Chen, "Parallel FFT with CORDIC for ultra wide band", IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol. 2, 2004, pp. 1173-1177.

[11]. H. Huang and L. Xiao, "CORDIC based fast algorithm for power-of-two point DCT and its efficient VLSI implementation", Microelectronics Journal, Vol. 45, No. 11, 2014, pp. 1480-1488.

[12]. Y.T. Lin, P.Y. Tsai and T.D. Chiueh, "Low-power variable-length fast Fourier transform processor", IEEE Proceedings-Computers and Digital Techniques, Vol. 152, No. 4, 2005, pp. 499-506.

[13]. A.P. Renardy, N. Ahmadi, A.A. Fadila, N. Shidqi and T. Adiono, "FPGA implementation of CORDIC algorithms for sine and cosine generator", IEEE International Conference on Electrical Engineering and Informatics, 2015, pp. 1-6.

[14]. W. Cui, H. Chen and Y. Han, "VLSI implementation of universal random number generator", Asia-Pacific Conference on Circuits and Systems, Vol. 1, 2002, pp. 465-470.

[15]. J.M. Muller, "The CORDIC Algorithm", Elementary Functions, 2016, pp. 165-184.

[16]. N. Das, S. Jena and S.K. Panda, "FPGA implementation of Angle Generator for CORDIC Based High pass FIR Filter Design", IOSR Journal of Electronics and Communication Engineering, 2016, pp. 1-11.