Grenze International Journal of Computer Theory and Engineering , Special issue

CORDIC Architectures: A Review

Anu H[1], D Mahesh Kumar[2] and D Jayadevappa[3]
[1]Asst. Prof., Dept. of E&IE, JSS Academy of Technical Education, Bengaluru
shamaiah.anu@gmail.com
[2]Associate Prof., Dept. of E&IE, JSS Academy of Technical Education, Bengaluru
dmkjssate@gmail.com
[3]Prof., Dept. of E&IE, JSS Academy of Technical Education, Bengaluru
devappa22@gmail.com

***Abstract*—From few decades Coordinate Rotation for Digital Computers (CORDIC) algorithm has received attention from academic and industry because of its applications in DSP, biomedical processing, neural networks, MIMO and many more. It is a repetitive algorithm, which involves shift and addition operations, for hardware understanding of basic uncomplicated functions. CORDIC is used as a fundamental block for building a variety of single chip solutions; aspects which are critical and need of consideration are high speed, low power and area, to achieve realistic on the whole performance. In this paper we will discuss about Parallel and Serial implementation, folded and unfolded implementation and IQ-Math and FP-Math CORDIC architectures on FPGA. In-depth evaluation of different architectures are presented, which makes available a first order information to designers who looks out for either advance development of performance or choice of rotational CORDIC for a precise function.**

***Index Terms*— CORDIC Algorithm, FPGA, Serial and Parallel Architecture, Rolling.**

I. INTRODUCTION

The CORDIC was introduced by Jack E. Volder [1] as a computing method, particularly for use in real-time digital computer where the mainstream of computations is trigonometric relations of navigation equations of coordinate transformation. It works on the principle of shift and adds to perform many functions of trigonometric, vector rotations, hyperbolic, logarithmic functions. There are two modes of operation in CORDIC, they are rotation and vector mode. It is mainly used in communication applications for implementation of universal modulator and demodulator. It is also used extensively in filtering, digital signal processing, matrix algebra, image processing etc.
Current applications need accurate and versatile computing requirements. Therefore CORDIC architecture ought to have the capability of delicate and versatile computing like alternative digital systems. CORDIC design cannot be achieved by many digit systems thanks to timing of the system and also the machine load the system required. An architectural design to meet the requirements like CORDIC architecture's has ability to supply accurate results and to work collaterally while it is accomplished on FPGA.
With the quick progress of FPGA performance, its appliance areas will be supplementary prolonged, its description includes low price, versatile programming, micro power, straight forward to transplant, etc. it is acceptable for understanding of CORDIC algorithm system structure in FPGA, the system is additional

suitable, the merchandise style cycle is greatly summarized. The traditional CORDIC algorithm cannot envelop the entire method cycle, additional repetitious times and slow convergence speed.
The CORDIC provides the occasion to determine all the required functions in a fairly simple way, lacking any hardware multiplier. The operation implemented in the algorithm is addition, subtraction, bit shift and lookup table. Because of the straightforwardness of the involved operations the CORDIC algorithm [2], [3], [4] is well matched for VLSI realization also.

A. CORDIC Algorithm

The formula operates in one in every of the 2 modes: Rotation or vectoring. The two modes decide the set of functions may be computed victimization using the algorithm. In Rotation mode, the x and y components of the beginning vector are input plus an angle of rotation. The hardware iteratively computes x and y parts of the vector ones it's been turned by the required angel of rotation. In Vectoring mode, x and y components are input, the magnitude and angle of the original vector are computed. This can be accomplished by rotating the input vector till it's aligned with the coordinate axis. By recording the angle of rotation to realize this alignment, we tend to get the angle of the unique vector. Once the formulae is complete, x-component of the vector is adequate the magnitude of the beginning vector.

$$y1 = \pm x0 = R0\ Sin\ (\theta 0 + \pi/4)$$
$$x1 = \pm y0 = R0\ Cos\ (\theta 0 + \pi/4) \quad (1)$$

Where x0 and y0 represent the input vector associated at the origin with magnitude R_0 and angle θ_0:

$$Xi = Ri\ Cos\ \theta i$$
$$Yi = Ri\ Sin\ \theta i \quad (2)$$

A new angle of rotation α_i is decided such that

$$\alpha i = tan^{-1}(2^{-i}) \quad (3)$$

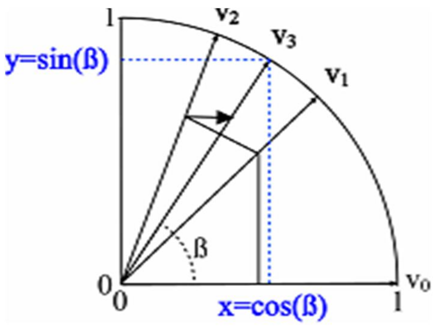


Figure 1. Illustration of CORDIC Algorithm

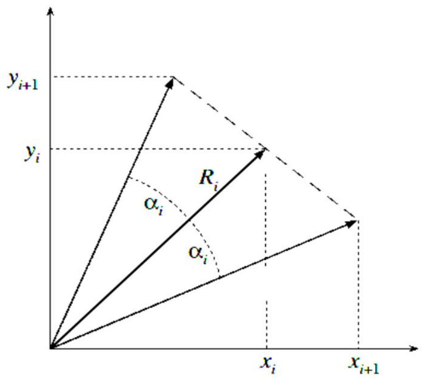


Figure 2. Step I in CORDIC Algorithm

Figure 2 shows what each step in the CORDIC algorithm looks like. At each step, a decision is made whether to rotate the vector by +αi or - αi. The expression for the rotated vector in the (i+1)th step is,

$$X_{i+1}\ \sqrt{1+2^{-2t}}\ \left(Cos\ (\theta i \pm \alpha i)\right)$$
$$Y_{i+1}\ \sqrt{1+2^{-2t}}\ \left(Sin\ (\theta i \pm \alpha i)\right) \qquad (5)$$

When applying the restriction in eqn. 3 the shifting and adding becomes evident.

$$x_{i+1} = 1/k_i\ (x_i - d_i\ 2^{-i} y_i)$$
$$y_{i+1} = 1/k_i\ (y_i + d_i\ 2^{-i} y_i)$$
$$K_i = \sqrt{1+2^{-2t}} \cdot d_i = \pm 1 \qquad (6)$$

This shifted price is then more (or subtracted) to the present price of the part. Volder referred this as operation cross-addition. It’s the cross-addition that permits the formula to be used effectively in digital hardware. As illustrated in fig. 1 all rotation steps have an on a rise within the magnitude of the input vector by an element of √1-2 -2t with every rotation. This error is introduced as a consequence of the algorithm’s derivation from the Givens remodel that rotates a vector by such as angle.

$$X = x\ Cos\ \emptyset - y\ sin\,\emptyset$$
$$Y = y\ Cos\ \emptyset + x\ sin\,\emptyset \qquad (7)$$

These terms can be rearranged by using the fundamental pure mathematics identity, the ordinal rotation may be extended and a general definition derived as

$$Y_{n+1} = \{\sqrt{1+2^{-2(0)}}\ \sqrt{1+2^{-2t(2)}}\ \cdots\cdots\ \sqrt{1+2^{-2n}}\}\ R0\ Sin\ (\theta + d_0\alpha_0 + d_i\alpha_1 + \ldots\ldots d_n\ \alpha_n) \qquad (8)$$

The total increase in magnitude are often specified as Kn = Πn √1+2 $-^{2r}$. This increase should be accounted while performing calculations using this algorithm. The CORDIC design [5], [6] incorporates three accumulator registers. The X and Y registers hold the current x and y parts of the vector because it is being rotated. The angle accumulator (Z) holds the entire rotation quantity completed at the core. The arctangent terms are often stored in a small lookup table. When this is accomplished, only an addition or subtraction is needed to work out the next value in the Z register, since d = ±1.

Abubeker et al. [7] discusses about parallel and serial CORIC architecture. Parallel architecture is implemented which contains of two shift register, a standard register, four adder/subtractions, lookup table and control unit. As shown in fig 3. Serial architecture is implemented with a bit serial arithmetic logic, along with one-bit adders to execute the CORDIC algorithm. This type of architecture requires very less FPGA resources, but will require more time to execute as shown in figure 4.

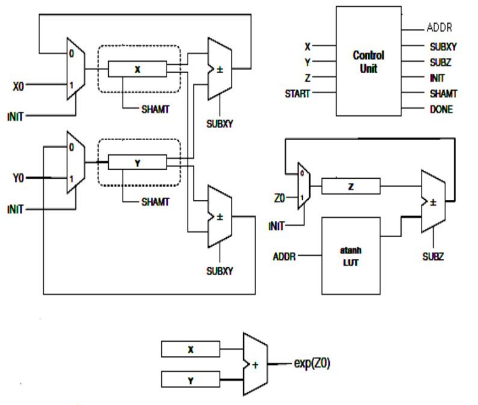


Figure 3. Control unit I

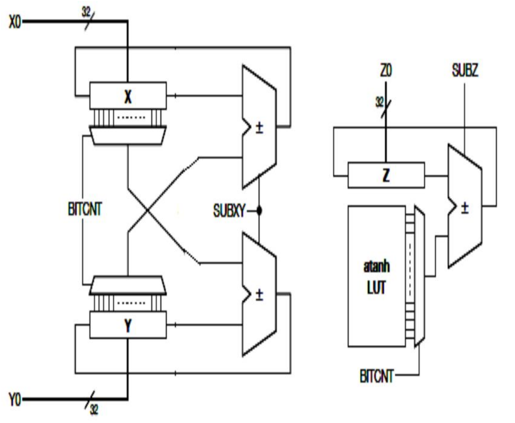
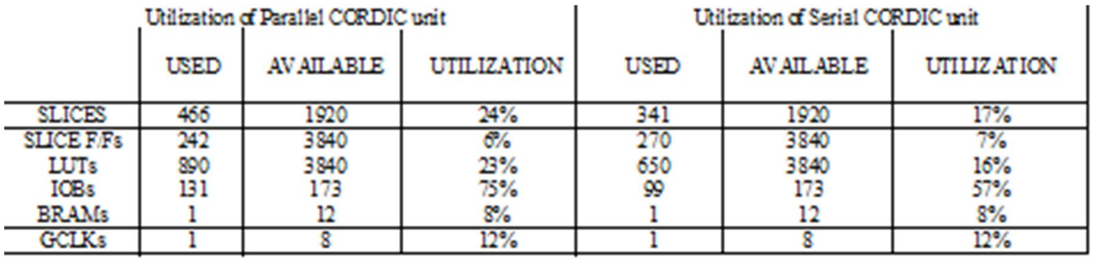


Figure 4. Control unit II

In Parallel architecture of CORDIC algorithm, it will take 32 clock cycles to complete. In theoretical maximum clock frequency of 56.507 MHz, the unit will take 17.697 ns to compute the ultimate value. In serial CORDIC system using 'n' iterations with a word size of w bits, the total execution time of the unit will be w*n+1 clock cycles. Every iteration will require 'w' cycles to complete because the values are passed serially through the adders, and n total repetitions are completed after the algorithm finishes. The added clock cycles are moderately offset by an increase in clock frequency.

TABLE I. UTILIZATION VALUES OF SERIAL AND PARALLEL CORDIC



	Utilization of Parallel CORDIC unit			Utilization of Serial CORDIC unit		
	USED	AVAILABLE	UTILIZATION	USED	AVAILABLE	UTILIZATION
SLICES	466	1920	24%	341	1920	17%
SLICE F/Fs	242	3840	6%	270	3840	7%
LUTs	890	3840	23%	650	3840	16%
IOBs	131	173	75%	99	173	57%
BRAMs	1	12	8%	1	12	8%
GCLKs	1	8	12%	1	8	12%

From the graph it is obvious that parallel structure needs a lot of devices than serial CORDIC structure. So serial CORDIC is chosen in applications where FPGA resources [8] are less utilized as compared to parallel CORDIC. For massive applications where FPGA resources are limited, the serial CORDIC unit offers several advantages over the parallel CORDIC unit. By operating solely on single bits in a serial fashion, the complication of the shift registers is reduced significantly. The size of the adders utilized in the design is also condensed since they only need to handle one bit at a time. Precision and accuracy are not sacrificed to achieve the reduction in resource utilization .On the other hand the added execution time may avoid this design from getting used in additional time-sensitive applications. For those, the parallel design or a quicker table-based approach would be suited.

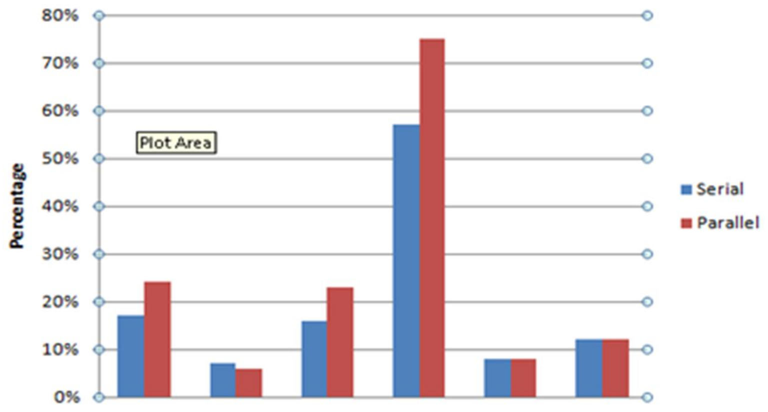


Figure 5. Device utilization summary

Masram et al. [9] discusses about folded word serial and unfolded parallel word design based on CORDIC architecture for hardware realization. In folded word serial architecture shift and add rule is applied, as shown in fig 6, each component consist of a MUX, a register to stock up up the data and an adder-subtractor component. 16 bit of data is given to MUX using 2:1 MUX and output is stored in register. The stored value of register and shift value of adder-subtractor is given as input to it and output is given out as subsequently iteration which is employed as one of the input to MUX. The FSM is utilized to keep track of shifting distances and the ROM addresses. Both adder-subtractor and shift register are employed in n time shared basis. Shifters ought to alter the shift distance with the no. of repetitions necessary for fan in and hence reduces the speed.

Un-folded architecture is used for designing of word-parallel architectures by making use of word serial architecture to increase the throughput or decrease the power consumption of the realization. The iteration method is unfolded for n dispensation elements performing the same iteration. A straight application of the unfolding conversion is to design parallel processing architecture from serial processing architecture, as shown in fig 7. In early stage the values are given to the first stage then output from first stage are inputs of the next stage and so on. The n numbers of repetitions are carried in itself. So shifters are not being changed. It eliminates the usage of ROM because it stores the angle value; in this structural design it is hard-wired. This reduces the area and the need for register is eliminated by applying the values directly. Pre-computed values are also given directly to the adder-subtractor unit.

Different parameters like latency, speed, throughput output and power using Xpower analyzer, is analyzed by collecting the data from simulator database.

The performance investigation of both the designs using CORDIC architecture is carried out. CMOS technologies used in unfolded architecture have higher speed and low power dissipation. Analysis shows that implementation of least number of cycle time n=7.5 reduces not only size but also conjointly improves single byte system performance fairly than using two byte operation on the same chip.

Bureu et al. discusses about different number format to implement CORDIC algorithm. Countless CORDIC architecture have been developed for using in FPGA [10], [11], [12] CORDIC architecture is to be enforced by using "IQ-Math" number system that Texas C28xDSP family firm runs or by means of the IEEE 754 "Floating-Point Numbers" to realize the requirement. The initial stage of dealing out the CORDIC Algorithm on FPGA is to establish the number format to be used by the system.

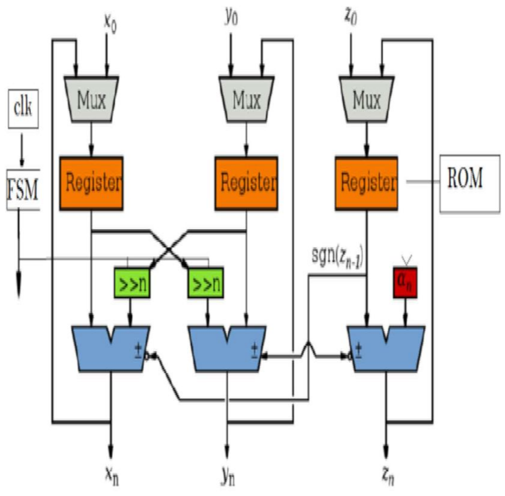


Figure 6. Serial processing unit 1

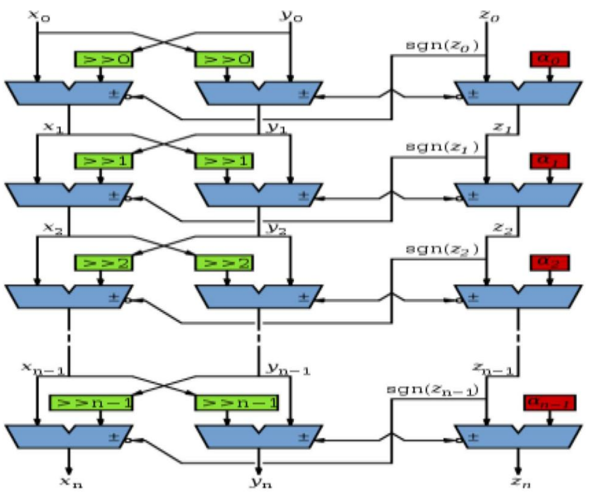


Figure 7. Parallel processing unit 2

TABLE II. SPECIFICATIONS OF THE CORDIC

	Parameter	16 bit CORDIC Architecture	
		Folded word serial design	Unfolded parallel design
Latency	Maximum Combinational block	64.591 ns	23.727ns
Area	No. of Registers	38	48
	NO. of LUTs	174	676
	No. of Logic blocks used	174	676
	No. of LUT flip-flopspairs used	206	685
	No .of bonded TOBs	50	50
Power Consumption	Power (Clock)	2mW	1mW
	Power (Logic)	0	0
	Power (Signals)	0	0
	Power (IOs)	0	0
	Power(leakage/quiescent)	0	0
	Dynamic Power	379 mW	379mW
	Total Power Consumption	381 mW	380mW
Throughput	Max. operating frequency	15.48 MHz	42.15 MHz

Computation of complicated mathematical functions such as CORDIC typically usually requires a bigger range of numbers. It is favored to use floating-point numbers attributable to their vibrant representation. Although they have many benefits, they work terribly slowly than ones in fixed point number. They occupy comparatively larger hardware resources of FPGA.

TABLE III. IEEE STANDARDS DETAILS

	IEEE-754 (FPGA)		IQ-Math(FPGA)		Matlab	
Angle	Cos	sin	cos	sin	cos	sin
61.25	0.4810 14043	0.876713037	0.481013298	0.876713037	0.4810 140708	0.876712864
146.60	- 0.834836006	0.550498723	- 0.834835290	0.550499200	- 0.834836074	0.550498602
230.25	- 0.639433562	- 0.768846452	- 0.639433622	- 0.768845796	- 0.639433495	- 0.768846400
287.21	0.295843660	- 0.955236494	0.295843124	- 0.9552364349	0.295843696	-0.955236352
360.00	0.9999997019	0.000012937	0.999999761	- 0.000013589	0.99999991	0.000012922

According to the results tabulated within the table higher than implementation of a CORDIC algorithm on FPGA using Floating-point number format has produced more delicate results than IQ-Math number format. With the study of IQ-Math, process accuracy is 2.3842E-007 whereas in the study of IEEE-754, process accuracy is 1.4E-45.
Importance of the number formats involves the forefront when algorithms like CORDIC including advance mathematical are being enforced.

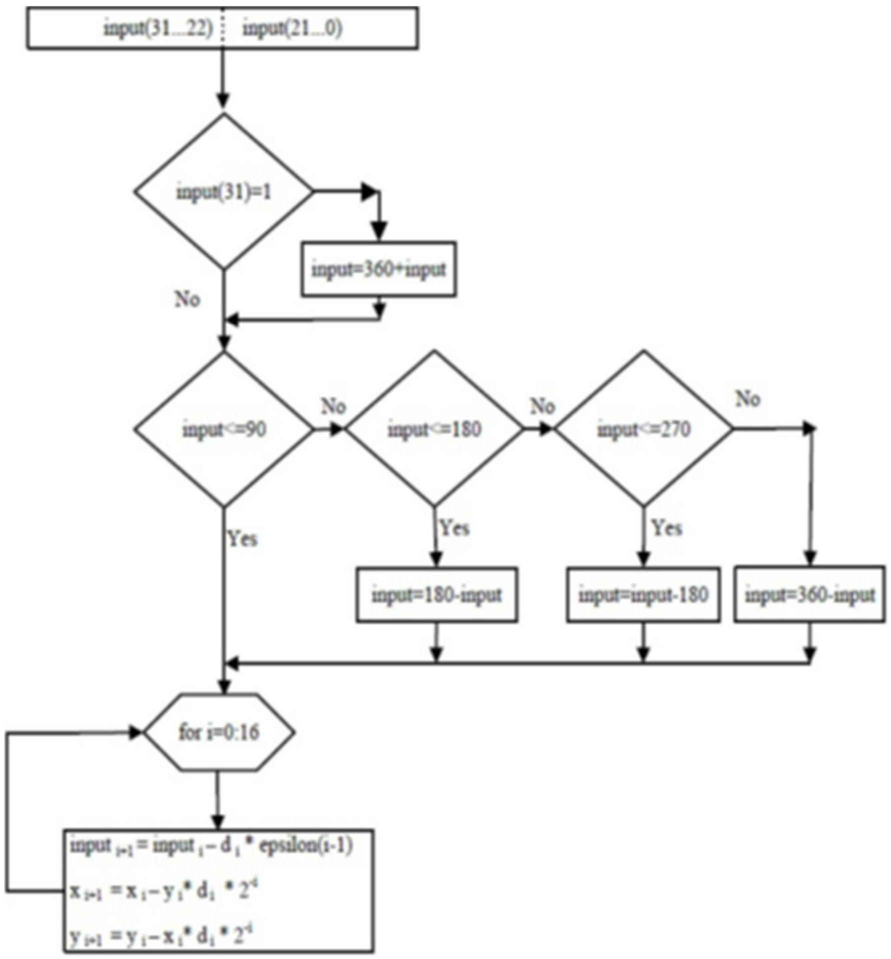


Fig 8, is the flow chart for IQ math number format used to implement CORDIC algorithm on FPGA

II. CONCLUSION

Serial CORDIC architecture is preferred and has more advantages than parallel CORDIC architecture in FPGA. Execution of a CORDIC based processor on FPGA gives us authoritative mechanism of implementing complicated computations on a podium that has a lot of property and suppleness at a comparatively lesser cost. Masram's CORDIC algorithm is implemented on FPGA based on two different platforms, i.e. folded serial design and unfolded parallel design. From the results obtained it is clear that unfolded parallel deign has less latency, occupies less space, power consumed by logical elements are less and has an operating frequency of 42.15 MHZ. Also, FP-CORDIC and IQ-CORDIC architectures have been compared supported by implementation time, method accurateness and hardware performance criteria on FPGA.

REFERENCES

[1] J. Volder, "The CORDIC computing technique, "IRE Trans. Electronic Computers, Vol. EC-8, pp. 330-334, Sept. 1959.

[2] S. Sahin, S. Dikmee, A. Kavak, "Which Number Format to Use for Baseband Wimax Modem Implementation on an FPGA ",Communication and Applications Conference, SIU, pp. 1-4,2008

[3] M. Wahab and D. Puckey, "FPGA-Based DSP Systems," Abindon EE & CS books, 2nd ed. , W. R. Moore and W. Luk, 1994.

[4] R. Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," ACM 0-89791-978-5/98/01, pp. 191-200, 1998.

[5] A. Sahin, A. Kavak, Y. Becerikli, H. E. Demiray, "Implementation of floating point arithmetic using an FPGA," Mathematical Methods in Engineering, pp. 445-453, 2007.

[6] M. A. Avuslu, C. Karakuzu, S. Sahin, M. Yakut, "Neural network training based on FPGA with floating point number format and it's perfonnance," Neural Comput & Applic, pp. 195-202,2011.

[7] Abubeker K. M, Sabana Backer and Abey Mathew Varghes, "Serial and Parallel Implementation of CORDIC Architecture: A Comparative Approach", 2013.

[8] Keshab K.Parhi ,"VLSI Digital Signal Processing systems "Willey Student edition "University of Minnesota.

[9] B. Y .Masram, P. T.Karule, "High Performance Analysis of a CORDIC Architectures based on FPGA: A comparative approach", 2014.

[10] Mehmet Ali AL Tuncu,Suhap Sahin, "FPGA Based Implementation of CORDIC using different number format", 2013

[11] J. Walther, "A unified algorithm for elementary functions," Proc. AFIPS Spring Joint Computing Conf., vol. 38, pp. 379-385, 1971.

[12] Ray Andraka. "A survey of CORDIC algorithms for FPGA based computers". Proceedings of the 1998 ACM/SIGDA sixth International symposium on FPGA, pp 192-200, New York, NY, USA, 1998. ACM Press.