Scaling Free CORDIC Algorithm Implementation of Sine and Cosine Function

A.S.N Mokhtar, M.B. I Reaz, K. Chellappan and M.A Mohd Ali

***Abstract*—Coordinate Rotation Digital Computer (CORDIC) algorithm is an established method in complex arithmetic function discovery using shift and add operations. An absolute Scaling-free CORDIC algorithm for cosine and sine function computation function has been implemented. A combination of third order approximation Taylor series and leading-one-bit detection algorithm has been adopted in this implementation. Seven (7) iterations were required for a 16-bits iterative architecture implementation of the proposed algorithm. The synthesis result shows that the algorithm is suitable for high-speed computational applications.**

***Index Terms*— Cosine/sine, FPGA, Scaling-free CORDIC algorithm**

I. Introduction

Effective hardware design is critical in circuit implementation practice. The CORDIC algorithm perfectly fits the statement. It offers an efficient and economic hardware implementation operation by exclusively compute addition and shift operations. CORDIC was introduced by Volder [1] in 1959. CORDIC is an iterative algorithm to compute trigonometric and hyperbolic functions. CORDIC has been used regularly in matrix computations [2], signal processing and image processing [3-5], communication [6], robotics and graphics [7], [8], neural network [9], [10] and motor control drive [11], [12].

The improvement of CORDIC algorithm has been a popular study. A lot of development has been established in the area of algorithm design and development of architectures especially for high-performance and low-cost hardware solutions. Pipelined and parallel CORDIC have been recommended for high throughput computations.

One of the first improvement that has been made was a redundant CORDIC which was proposed by Ercegovac and Lang [13].

Manuscript received Mar 6, 2013

A.S.N Mokhtar is with Department of Electrical, Electronics and Systems Engineering, Faculty of Engineering and Built Environment, National University of Malaysia, on leave from Department of Electric and Electronics, Faculty of Engineering, National Defence University of Malaysia, Kuala Lumpur, Malaysia (E-mail: anis@upnm.edu.my)

M. B. I Reaz is with Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, National University of Malaysia, Malaysia. (E-mail: mamun.reaz@gmail.com).

K. Chellappan and M.A Mohd Ali are with Institute of Space Science (ANGKASA), National University of Malaysia, Selangor, Malaysia. (E-mail: kckalai@ukm.my, mama@eng.ukm.my)

Many difference methods have been proposed for the Radix-2 with constant scale factor using signed digit (SD) arithmetic such as double rotation method, correcting rotation method [14], branching method [15], double step branching method [16], and Differential CORDIC [17]. Carry save (CS) arithmetic has been used in radix-2 CORDIC such as low latency redundant [18] and high speed bit level pipelined [19]. The speed of CORDIC algorithm can be improved by reducing the number of iterations, therefore architectures using radix-4 micro-rotations have been developed and reported in the literature by Antelo et al and Lakshmi [20], [21]. However, it requires higher computational time for each iteration and involves larger hardware compared to the radix-2 CORDIC.

Researches in conventional CORDIC improvements continued in two different directions [22], namely high speed performance and scaling factor implementation. The main focus was to have high speed computation with least iterations. The scaling-free by Maharatna et al [23] need multiplication by constant scale-factor and consume more area. The enhanced scale-free CORDIC in Jaimeet al [24] integrates few conventional CORDIC iterations with conventional scaling-free CORDIC iterations for a competent pipelined CORDIC implementation with improved range of convergence (RoC). However, combination of two different types of CORDIC iterations degrades the performance. Virtually modified scaling-free algorithm [23] extends the range of convergence over the entire coordinate space and introduces an adaptive scale factor. The scaling-free CORDIC algorithm by Agrawal et al [25] is completely eliminates the scale factor.

II. Methodology Architecture

Scaling-free CORDIC was initially aimed to design a scale-free coordinate CORDIC using Taylor series [26]. The scaling-free was first attempted to completely dispose of the scale factor. The sin and cosine were approximated to [25]:

$$\sin\alpha_i = 2^{-i} \qquad \cos\alpha_i = 1 - 2^{-(2i+1)} \tag{1}$$

Taylor series expansion allowed the rotation only in one direction compared to conventional CORDIC [27]. The Taylor series of sine and cosine terms defined as:

$$\sin\alpha_i = \sum_{n=0}^{\infty} (-1)^n \frac{{\alpha_i}^{2n}}{(2n)!} = 1 - \frac{{\alpha_i}^2}{2!} + \frac{{\alpha_i}^4}{4!} - + \ldots. \tag{2}$$

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$\cos\alpha_i = \sum_{n=0}^{\infty}(-1)^n \frac{\alpha_i^{2n+1}}{(2n+1)!} = \alpha_i - \frac{\alpha_i^3}{3!} + \frac{\alpha_i^5}{5!} - +... \qquad (3)$$

For hardware implementation of the series in equation (3) needs to be approximately implying a compromise in accuracy. The suggested algorithm [25] used third order expansion and approximate 3! to 2^2. By this approximation the rest of the Taylor series can be implemented by using shift and add operation. The MSE error in sine and cosine values resulted from the approximation of the factorial was 0.0168% which is insignificant and does not affect the accuracy of the system performance [25].

Design of CORDIC processor is divided in two main parts, the *Arithmetic Calculation Modules* and *Shift Calculation Modules.*

A. Arithmetic Calculation Modules

The expansion of the third order Taylor series approximation proposed by Agrawal et. al. [25], is as in equation (4):

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-\frac{\alpha_{si}^2}{2!} & -\left(\alpha_{si}-\frac{\alpha_{si}^3}{3!}\right) \\ \alpha_{si}-\frac{\alpha_{si}^3}{3!} & 1-\frac{\alpha_{si}^2}{2!} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (4)$$

Assuming $\alpha_i = 2^{-si}$ and approximation of factorial, the above equation can be simplified to

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1-\frac{2^{-2si}}{2} & -\left(2^{-si}-\frac{2^{-3si}}{2^2}\right) \\ 2^{-si}-\frac{2^{-3si}}{2^2} & 1-\frac{2^{-2si}}{2} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} 1-2^{-(2si+1)} & -\left(2^{-si}-2^{-(3si+3)}\right) \\ 2^{-si}-2^{-(3si+3)} & 1-2^{-(2si+1)} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (5)$$

The arithmetic calculation modules were used in implementing the above equation as shown in Fig. 1. 16-bits register is used at the end of the module for synchronization purpose. This module was implemented by using fixed point format. 16-bits used for the x_i and y_i input. The value of s_i obtained from the shift calculation module. The output of this module will be used as input to the next clock cycle, and the iteration continues until the counter is reset to zero.

B. Shift calculation module

The fixed point format of the elementary angle (α_s) has one bit set and represented by $\alpha_{si} = 2^{-si}$. The first one bit number of any input string counting from the Most Significant Bit (MSB), M is used for further calculation of shift iteration (s_i), for a fix word-length (N). s_i of the elementary angle was given by

$$s_i = N - M \qquad (6)$$

In this research the word-length is fixed to 16. The module's input is the angle to be rotated, theta (θ_i). The operation begins to get the first 1 bit from the MSB (M). Fig. 1 is illustrating the shift module algorithm flow:

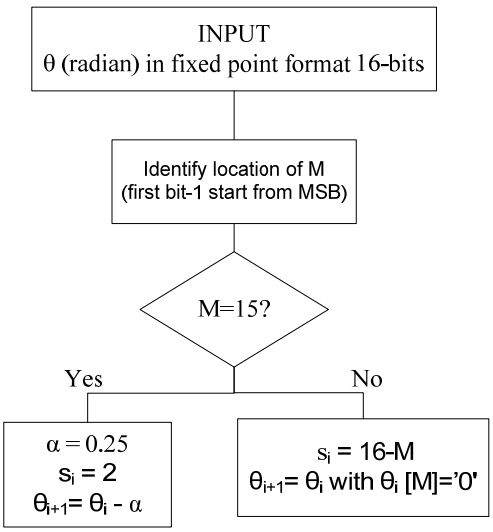

Fig. 1: Flowchart for generating micro-rotation sequence.

The elementary angle (α) is corresponding to the basic shift (s). Basic shift was the first elementary angle for rotation α. The expression for the basic shift is as stated in equation (7):

$$basic-shift, s = \left[\frac{b-\log_2(n+1)!}{(n+1)}\right] \qquad (7)$$

Where b is the wordlength and n is the total number of iterations. Total number of iteration for third order Taylor series is 7, as discussed in [25] and the basic shift for 16-bits word length is $2.854 \approx 2$.

In shift calculation module, multiple iteration of shift is performed. Shift calculation module is used to get the shift index (s_i) parameter.

III. Result and Discussion

The architecture was developed by using Verilog hardware description language in Quartus II Altera. All modules were developed as described in the previous section. Fig. 2 illustrates the simulation result of the scaling-free CORDIC calculation. The input theta is 0.61radian (35^o). In 16-bit fix-point format, 0.61radian is equal to 9C61. The values of s_i can be calculated as shown in Table 1. s_i node in Fig. 2 representing the simulated s_i values.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
SEQUENCE OF S_i VALUES

No of iteration	Input θ (radian)	M	S_i=N-M	Output θ_{i+1}
1	9C61	15	2	9C61-4000 =5C61 0101 1100 0110 0001 [5C61]
2	5C61	14	16-14=2	0001 1100 0110 0001 [1C61]
3	1C61	12	16-12=4	0000 1100 0110 0001 [0C61]
4	0C61	11	16-11=5	0000 0100 0110 0001 [0461]
5	0461	10	16-10=6	0000 0000 0110 0001 [0061]
6	0061	6	16-6=10	0000 0000 0010 0001 [0021]
7	0021	5	16-5=11	0000 0000 0000 0001 [0001]

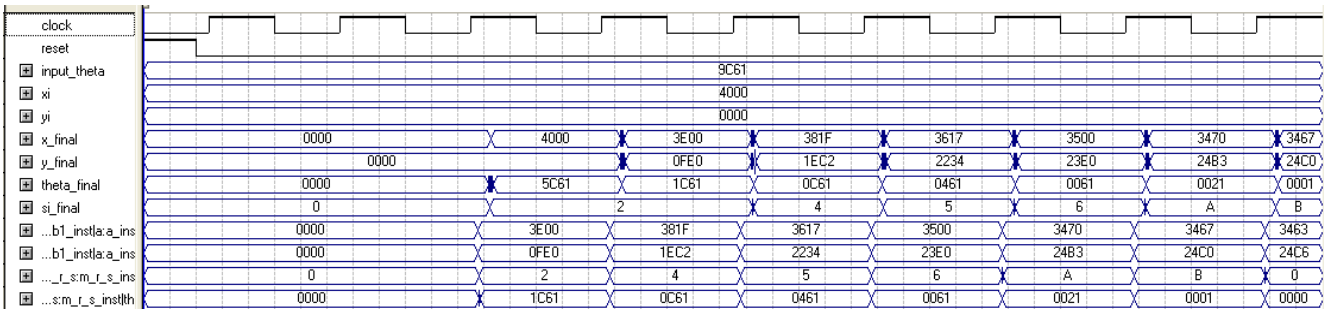

Fig. 2: Simulation result of scaling-free CORDIC

Fig. 2 shows the simulation result of the implemented scaling-free CORDIC. From this figure the calculation output of cos35^o is 3467H which shown as node x_final and sin 35^o is 24C0H as in node y_final. These values need to be converted back in order to get the real value which is equal to 0.819336 for cos35^o and 0.573425 for sin 35^o. From the numerical calculation the value of cos35^o and sin35^o is 0.819152 and 0.573576. The accuracy of the calculation is estimated to be 99.97% for both. From the result, it shows that the scaling-free CORDIC is a reliable calculation for cosine and sine functions which can be implemented in FPGA by only using add and shift functions.

IV. CONCLUSION

Scaling-free CORDIC algorithm provides an iterative calculation using only add and shift function for cosine and sine function calculation in FPGA. The elimination of redundant iterations using leading-one-bit technique improves the number of iteration compared to conventional CORDIC. Less iteration offers faster computation and optimal efficiency. The implementation of this scaling-free CORDIC will improved the performance of the application in term of efficiency and stability.

REFERENCE

[1] Volder J. E, "Cordic Trigonometric Computing Technique." *IRE Trans. Electronics Computer*. vol. EC-8, pp. 330-334, 1959.

[2] A.Despain, "Fourier Transform Computers Using CORDIC Iterations," *IEEE Transactions on Computers.*vol c-23, no 10.1974 pp. 993-1001.

[3] M. Kuhlmann and K. K. Parhi, "A High-speed CORDIC Algorithm and Architecture for DSP Applications." *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE,* vol., no., pp.732,741, 1999

[4] A. Banerjee, A. Sundar Dhar, and S. Banerjee, "FPGA realization of a CORDIC based FFT processor for biomedical signal processing," *Microprocessors and Microsystems*, vol. 25, no. 3, pp. 131–142, May 2001.

[5] Suchitra Sathyanarayana ,Ravi Kumar Satzoda and Srikanthan Thambipillai, "unified CORDIC based processor for image processing," *15th International Conference on Digital Signal Processing,* vol. 1, no. 2, pp. 343–346, 2007.

[6] K. Maharatna and S. Banerjee, "A VLSI array architecture for Hough transform," *Pattern Recognition*, vol. 34, no. 7, pp. 1503–1512, Jan. 2001.

[7] C. S. Lee and P. Chang, "A maximum pipelined CORDIC architecture for inverse kinematic position computation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 445–458, Oct. 1987.

[8] B. J. Hosticka, "Inverse kinematics computations with modified CORDIC iterations," vol. 143, no. 1, pp. 1–6, 1996.

[9] U. Meyer-Bäse and A. Meyer-Bäse, "A Fast Modified CORDIC—Implementation of Radial Basis Neural Networks," *The Journal of VLSI Signal …*, 1998.

[10] A. Meyer-Bäse, R. Watzel, U. Meyer-Bäse, and S. Foo, "A Parallel Cordic Architecture Dedicated To Compute The Gaussian Potential Function In Neural Networks," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7–8, pp. 595–605, 2003.

[11] A.S.N. Mokhtar, M. B. I. Reaz,and M.A.Mohd. Alauddin, "Inverse Park Transformation Using Cordic and Phase-Locked Loop," pp. 422–431, 2012.

[12] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, and K. Bouallaga, "Fully Integrated FPGA-Based Controller for Synchronous Motor Drive," *IEEE Transactions on Industrial Electronics,* , vol. 56, no. 10, pp. 4006–4017, 2009.

[13] M. D. Ercegovac and T. Lang, "Digital Arithmetic," Elsevier, Amsterdam, The Metherlands,2004.

[14] N. Takagi and T. Asada, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Transactions Computers,* , vol. 40, no. 9, pp. 989–995, 1991.

[15] J. Duprat and J. Muller, "The CORDIC Algorithm : New Results for Fast VLSI Implementation," vol. 42, no. 2, pp. 168–178, 1993.

[16] D. S. Phatak, "Sine and Cosine Generation," *Electrical Engineering*, 1998.

[17] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 307–318, Mar. 1996.

[18] D. Timmermann, H. Hahn, B. J. Hosticka, and S. Member, "Low latency time CORDIC algorithms - Computers, IEEE Transactions on," vol. 41, no. 8, 1992.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[19] Dawid, H.; Meyr, H., "High speed bit-level pipelined architectures for redundant CORDIC implementation," *Application Specific Array Processors, 1992. Proceedings of the International Conference on* , vol., no., pp.358,372, 4-7 Aug 1992

[20] 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.

[21] B. Lakshmi and A. S. Dhar, "Low Latency VLSI Architecture for the Radix-4 CORDIC Algorithm," *Electrical Communication*, no. 210, 2008.

[22] L. Vachhani, S. Member, K. Sridharan, S. Member, and P. K. Meher, "Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation," vol. 56, no. 1, pp. 61–65, 2009.

[23] Maharatna, K.; Banerjee, S.; Grass, E.; Krstic, M.; Troya, A., "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.15, no.11, pp.1463,1474, Nov. 2005

[24] F. J. Jaime, M. A. Sanchez, J. Hormigo, J. Villalba, and E. L. Zapata, "Enhanced Scaling-Free CORDIC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 7, pp. 1654–1662, Jul. 2010.

[25] S. Aggarwal, P. K. Meher, and K. Khare, "Area-Time Efficient Scaling-Free CORDIC Using Generalized Micro-Rotation Selection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 8, pp. 1542–1546, Aug. 2012.

[26] Maharatna, K.; Troya, A.; Banerjee, S.; Grass, E., "Virtually scaling-free adaptive CORDIC rotator," *Computers and Digital Techniques, IEE Proceedings* - , vol.151, no.6, pp.448,456, 18 Nov. 2004.

[27] J. Volder, "The CORDIC trigonometric computing technique," *Electronic Computers, IRE Transactions on*, 1959.

ISBN: 978-988-19252-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)