FPGA Implementation of Adaptive Cordic Algorithm using HDL

Mr.Ashish S.Khachane[1]
ResearchScholar,JDCOEM,Nagpur.
Email Id: ashish.khachane@gmail.com

Dr. Pramod B.Patil [2]
CEO and Principal,CICET, Nagpur.
Email Id: ppamt07@ gmail.com

Prof.Bharti Masram[3]
Research Scholar,JDCOEM, Nagpur.
Email Id: bharatimasram@gmail.com

***Abstract*: In the evaluation of a wide variety of elementary functions, the CORDIC algorithm plays a vital role. It is a simple and well-designed method, but it suffers from long latency. The Adaptive CORDIC Algorithm is able to reduce the number of iterations by more than 30 percent, but its implementation in hardware consumes a large increase in cycle time, to accommodate its complex angle selection function. This restricts its use to those cases where the angle of rotation is fixed and known in advance, so that the angle selection can be performed offline. This paper presents the FPGA implementation of adaptive CORDIC algorithm using HDL with the help of Xilinx14.7 and Quartus9.1 Softwares. The method also has the advantage that all the angle constants are found in parallel, in a single step, by testing only the initial rotation angle, without having to perform successive CORDIC iterations. This dynamic Angle Recoding method can be formulated to use "sections," to limit the number of range Comparators needed, to a reasonable value. There is an increase in the number of adaptive CORDIC iterations needed, but this problem can be mitigated by using a buffer in conjunction with the method of sections.**

***Index Terms— Adaptive* CORDIC Algorithm, Latency, Angle Recoding Method.**

I. INTRODUCTION

The coordinate rotation digital computer (CORDIC) is an efficient iterative algorithm which can be used to compute several elementary functions. It was proposed by Volder in 1959 [1], and later extended by Walther [2] to encompass circular, linear, and hyperbolic coordinate systems. The algorithm is commonly used to evaluate elementary functions as well as perform rotations in a variety of applications. However, it suffers from the problem of long latency. The Angle Recoding method [3] applied to CORDIC reduces the number of iterations, but the implementation of its angle selection function in hardware requires a large increase to the cycle time, thus, reducing its efficiency. This paper presents a simpler angle selection function for the Angle Recoding method which does not require the cycle time to be increased but still achieves the advantages of the reduction in iteration count. The technique is named 'Adaptive CORDIC'.

BASIC OF CORDIC ALGORITHM

CORDIC algorithm is derived from the general equations of vector rotation. Vector rotation can also be used for polar to rectangular and rectangular to polar conversion, for vector magnitude and as a building block in certain transforms such as the DFT and DCT [5]. The CORDIC algorithm provides an iterative method of performing vector rotation by arbitrary angle using shift and adds. The algorithm credited to Volder is derived from the general rotation transform. If a vector V with co-ordinates (x,y) is rotated through an angle Ø then a new vector V' can be obtained with co-ordinates (x',y') where x' and y' can be obtained using x , y and Ø by the following method as shown in Fig.1.

$X=r\cos\theta, y=r\sin\theta$ -------------------------------(1)

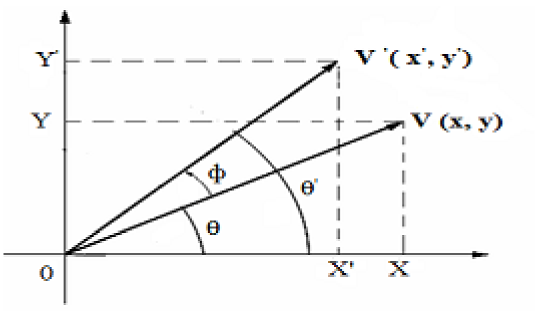


Fig 1 Rotation of a Vector V by the angle Ø

V' came into picture after anticlockwise rotation by an angle Ø. From Fig.1, it can be observed Θ'-θ=Ø
Therefore,

$$V' = \begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} XCOSØ - YSINØ \\ YCOSØ + XSINØ \end{bmatrix} \quad \text{......................(2)}$$

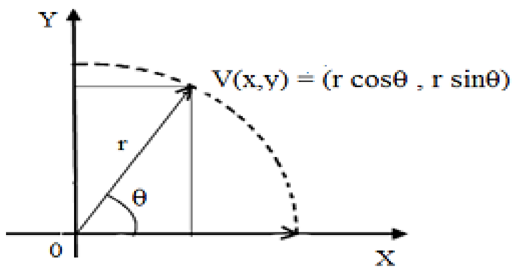


Fig 2 Vector V with magnitude r and phase θ

The Original CORDIC algorithm is specified by the following iterative equations, with N denoting the number of iterations:

(ɣi=0,1,…..N)

$X_{i+1}=X_i-m\sigma_i 2^{-S(m,i)}Yi$

$Y_{i+1}=Y_i+\sigma_i 2^{S(m,i)}Xi$ …………………(3)

$Z_{i+1}=Z_i-\sigma_i\alpha_{m,i}$

The parameter "m" denotes the coordinate system, i.e. circular (m=1), linear (m=0), or hyperbolic (m= -1). Two operational modes are possible, rotation and vectoring, but this paper is only concerned with the rotation mode where the initial position of the vector (X0; Y0) and the angle through which to rotate it (θ) are known, and the final coordinates of the target vector are to be determined. The goal is to reduce the residual angle (Zi) to 0 through the use of appropriate values of the rotation direction . σi=sign(Zi)

The predetermined pseudo rotation angles α(m,i), sometimes known as angle constants or Arc Tangent Radices (ATR), are given by:

$$\alpha_{m,i}=\frac{1}{\sqrt{m}}\tan^{-1}\left(\sqrt{m}*2^{-S(m,i)}\right).$$

II. Literature review of low latency CORDIC techniques

The shift sequences is given by: S(m,i)=0,1,2,3,4,5,………………..N if m=1

=1,2,3,4,5,6,........................N if m=0

=1,2,3,4,5,…………………...;N if m=1;

repeats at $(3^{i+2}-1)/2$

A scaling factorki is associated with the ith iteration step.

$$k_i=\sqrt{1+m\sigma_i^2*2^{-2S(m,i)}}.$$

The scaling factors are usually combined together and applied cumulatively as a total scaling factor K (for the

N iterations) given by:

$$K=\prod_{i=0}^{N}k_i=\prod_{i=0}^{N}\sqrt{1+m\sigma_i^2*2^{-2S(m,i)}}.$$

The total scaling factor may be applied either at the start or at the end of the algorithm. As long as a rotation is performed on every iteration, K is a constant that is determined solely by the number of iterations, N. A fixed value for K eliminates the need for a ROM in which to store the different scaling factors for every possible angle.

The latency of an iterative algorithm like CORDIC is determined by the product of the number of iterations times the cycle time of each iteration. The Original CORDIC algorithm exhibits high and constant latency since the algorithm always requires a fixed number of iterations and there is no facility for early termination. There have been several methods proposed by researchers in an effort to reduce the overall latency of the algorithm. Huand Naganathan [3] propose the Angle Recoding method which can reduce the number of iterations by more than 50 percent—however, the angle selection function that is used is very complex, and in order to avoid impacting the cycle time, it is restricted to applications where the rotation angles are fixed and known a priori, so that the Angle Recoding can be done offline. In [4], an angle selection function inspired by control theory is presented which can be used dynamically with any rotation angle, but the savings obtained in iteration count is not as high as that of the Angle Recoding method.

Another method of reducing the number of iterations is to use a higher radix number system, such as a minimally redundant radix-4 number system [5], [6], where i can take on values off 2; 1;0;1;2g. The number of iterations is reduced, because more bits of the result are produced in each iteration. The cycle time needs to be increased because of the additional effort to identify the correct value of i from among the five possible values. Duprat and Miller [7] take the tack of reducing the cycle time of a CORDIC iteration by using fast adders, based upon the use of redundant arithmetic to express the operands. The method proposed by Phatak [8] is able to execute two consecutive pseudorotations in the same cycle. It does this by examining a sliding window of six digits of the residual angle at a time, to predict the values of i and iþ1. In the Hybrid CORDIC [9] method, if m=ceilðN=3Þ, it is shown that after the first miterations have been executed sequentially, the residual angle that remains can be used to predict the rotation directions of the remaining N-miterations (roughly 2=3 N) in parallel. These iterations are then to be executed using the method shown in [10] or else by using some type of combined or pipelined architecture [11]. The first miterations can be speeded up by using the method proposed by Arbaugh [12] in which the rotation angle is used as an index into a ROM lookup table which provides two multiplication factors. The rotation of the vector is then performed by multiplying these factors by the initial coordinates (x0,y0) of the vector to be rotated, in a standard multiplier.

In the Para-CORDIC method [13] the sign of the first m iterations is predicted in parallel using the binary-to-bipolar recoding on the first m bits of the rotation angle. In the P-CORDIC method [14], a preliminary step is used to predict the rotation directions, by adding a constant as well as a variable offset (obtained from a ROM) to the rotation angle, and then using a fast adder to obtain the required binary representation of the rotation directions. The knowledge of the rotation directions is used to perform more than one pseudo rotation in an iteration. In this paper, attention is focused mainly on techniques that reduce the number of iterations, while keeping the cycle time unchanged. An example of the Original CORDIC method is presented in this section. Some of the deficiencies of this method are pointed out, which are then addressed by Control CORDIC [4] and the Angle Recoding method [3].

Original CORDIC—Rotation through 25 Degrees

Consider the rotation of a vector from the x-axis through an angle of, e.g., 25 degrees. Assuming that the rotation is to be accomplished in nine iterations (i.e. ,N=0...8), the set Q of predetermined angle constants that are used is as follows :

Q= $\{45^0, 26.565^0, 14.036^0, 7.125^0, 3.576^0, 1.79^0, 0.895^0, 0.448^0, 0.2238^0\}$

In the Original CORDIC method, the rotation through 25 degrees is carried out by the following sequence of angular steps or pseudorotations, that add up to approximately 25 degrees, as shown on Fig. 1:

$$25^0 = \{+45^0 - 26.565^0 + 14.036^0 - 7.125^0 - 3.576^0 + 1.79^0 + 0.895^0 + 0.448^0 + 0.2238^0\}$$
$$= 25.1268^0.$$

Even though the algorithm as a whole converges, individual intermediate pseudo rotations may diverge, and must be corrected by succeeding pseudo rotations. For example, the iteration from i=2 to i=3is a divergent pseudo rotation

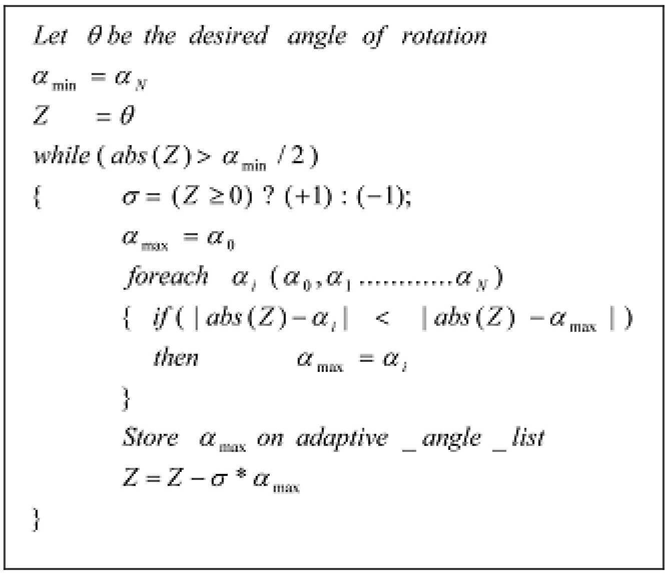
Angle Recoding Method—Rotation through 25 Degrees

The Angle Recoding method proposed by Hu and Naganathan [3] is a more aggressive algorithm. It eliminates divergent pseudo rotations and makes every effort to converge to the final target in the least number of iterations. At every iteration, the angle constant ( ) that will bring the residual angle (Zi) closest to 0 is chosen from the set of available angle constants. For example, rotation through

25 degrees in Fig. 1,

$$25^0 = (26.565^0 - 1.79^0 + 0.2238^0)$$
$$= 24.9988^0.$$

This method results in a reduction of more than 50 percent in the number of iterations needed. Its greatest disadvantage, however, is that the function used to select the angle constant is very complex. Its implementation requires the cycle time to be increased considerably(>2x). It is, therefore, most attractive in static cases where the rotation angleθ is fixed and known a priori. The selection of the angle constants can be done offline, and the angle constants saved in a Look-Up Table.



```
Let $\theta$ be the desired angle of rotation
$\alpha_{\min} = \alpha_N$
$Z = \theta$
while ( $abs(Z) > \alpha_{\min}/2$ )
{        $\sigma = (Z \geq 0)\ ?\ (+1) : (-1);$
         $\alpha_{\max} = \alpha_0$
         foreach $\alpha_i$ ( $\alpha_0, \alpha_1 \ldots\ldots\ldots \alpha_N$ )
         {  if ( $|abs(Z) - \alpha_i| < |abs(Z) - \alpha_{\max}|$ )
              then    $\alpha_{\max} = \alpha_i$
         }
         Store $\alpha_{\max}$ on adaptive_angle_list
         $Z = Z - \sigma * \alpha_{\max}$
}
```

Fig. 3. Angle selection in the Angle Recoding method.

III. Adaptive CORDIC Technique By Dynamic Angle Selection

There is a strong motivation to investigate the use of dynamic angle selection techniques for Angle Recoding, because they can reduce the iteration count by more than 50 percent, for any arbitrary rotation angle (not only those that are known a priori). A straightforward implementation of the Angle Recoding method in hardware is presented first. It requires a large increase in cycle time that makes its use in dynamic situations untenable. Then the Parallel Angle Recoding method is presented that performs the angle selection using a much simpler preparatory step, without having to modify the logic used in each iteration (thus requiring no change in the cycle time).

Direct Hardware Implementation Of ADAPTIVE CORDIC Angle Recoding

The Angle Recoding method can be implemented in hardware if all the angle constants (α's) are tested in parallel, rather than serially as in the original algorithm. Fig. 3 shows the original Angle Recoding method and Fig. 4 shows the logic needed to implement it, so that it can handle any rotation angle dynamically. The residual angle Zi is passed to adder-subtractor units that compute the difference quantity (Zi-σiαi) for each angle constant αi. The differences are then compared against each other using a binary-tree like structure, to find the smallest difference. The corresponding angle constant is selected and the index i of the angle constant (satisfying $\tan(\alpha)=2^{-i}$) then determines the shift amount to be used. It is only then that the X and Y coordinates of the vector are updated. The adder-subtracters and binary tree comparison unit that are used in every iteration is on the critical path and greatly increases the cycle time. This mitigates the gains that accrue from the reduction in iteration count, leaving the latency unchanged or even larger than before.

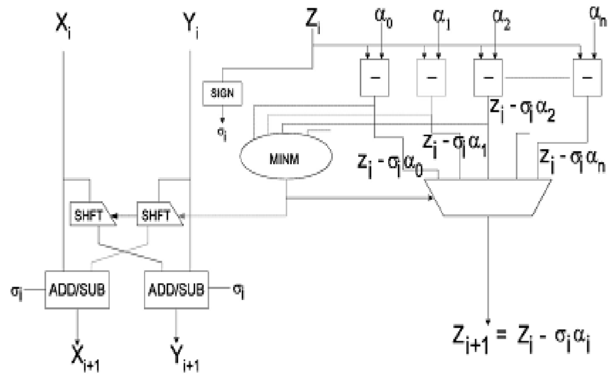
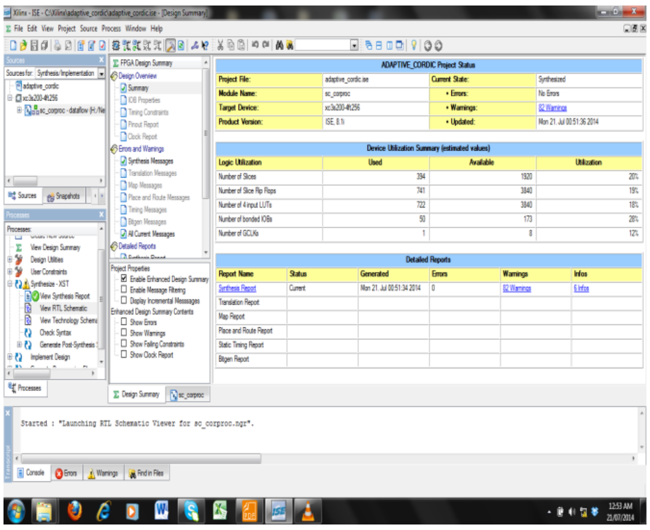


Fig. 4. Hardware implementation of the Angle Recoding method.

IV. Result



ADAPTIVE_CORDIC Project Status			
Project File:	adaptive_cordic.ise	Current State:	Synthesized
Module Name:	sc_corproc	• Errors:	No Errors
Target Device:	xc3s200-4ft256	• Warnings:	82 Warnings
Product Version:	ISE, 8.1i	• Updated:	Mon 21. Jul 00:51:36 2014

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	394	1920	20%
Number of Slice Flip Flops	741	3840	19%
Number of 4 input LUTs	722	3840	18%
Number of bonded IOBs	50	173	28%
Number of GCLKs	1	8	12%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon 21. Jul 00:51:34 2014	0	82 Warnings	6 Infos
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

```
Started : "Launching RTL Schematic Viewer for sc_corproc.ngr".
```

Fig. 5. Design Flow Summary

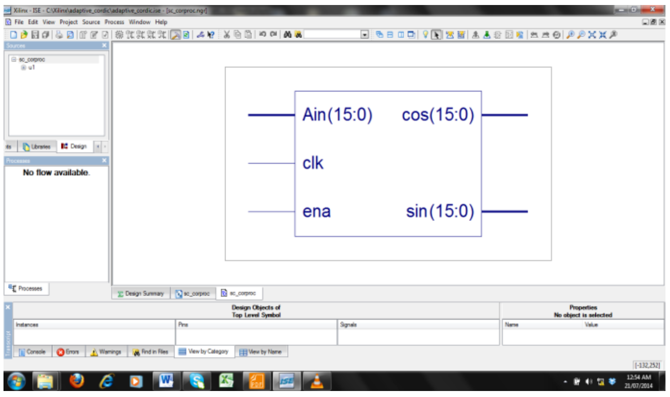


Fig. 6 Folded RTL View Of Adaptive Cordic Technique

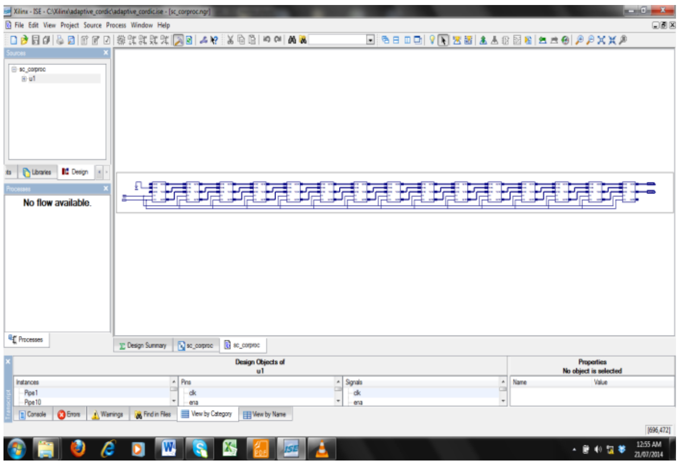


Fig. 7 Unfoled RTL View Of Adaptive Cordic Technique

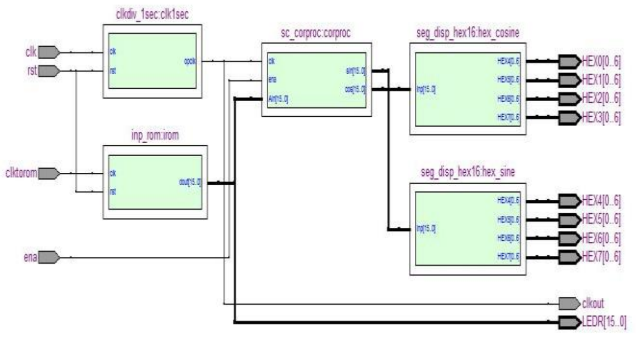
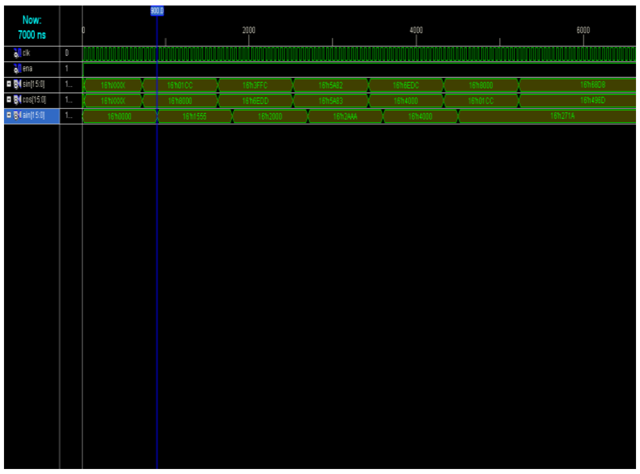


Fig. 8 SRAM Object File Of Adaptive Cordic Architecture



V. Conclusion

The Angle Recoding method in CORDIC, reduces the number of iterations needed by more than 50 percent, but its use is restricted to applications where the rotation angle is known a priori. In order to apply the Angle Recoding method to arbitrary rotation angle, the angle selection component of the method must be simplified so that it can be implemented without requiring a large increase in cycle time. This paper has presented the Parallel Angle Recoding method that provides a simple way to identify the angle constants used in the Angle Recoding of arbitrary rotation angles. Several contiguous ranges are identified that are associated with each angle constant. By comparing the incoming rotation angle against these ranges, all the angle constants used in the Angle Recoding can be identified in a single step. The number of range comparators needed can be reduced by using the method of sections, but it does add some overhead cycles to the Adaptive CORDIC iteration count. The problem can be alleviated considerably by using a buffer in conjunction with sections. The buffer reduces the number of overhead cycles by allowing independent sections drawn from different rotation angles to follow each other through the pipeline. It also allows the compensation of the scaling factor to be carried out in parallel with the CORDIC iterations. The method can reduce the number of iterations needed, without requiring any increase in cycle time, it does, however, incur the overhead of the comparison logic.

VI. Acknowledgements

The authors would like to thank the anonymous referees for their constructive comments.

VII. References

[1] J.E. Volder, “The CORDIC Trigonometric Computing Technique, “IRE Trans. Electronic Computers,vol. EC-8, pp. 330-334, 1959.530 IEEE TRANSACTIONS ON COMPUTERS, VOL. 59, NO. 4, APRIL 2010Fig. 18. Latency as a function of section count, forN=8.Fig. 19. Latency as a function of section count, orN=16.Fig. 20. Latency as a function of section count, forN¼24.Fig. 17. Pipeline stall in cycle 6 because no angle constants werechosen from sectionSw.

[2] J.S. Walther, “A Unified Algorithm for Elementary Functions,”Proc. Spring Joint Computer Conf.,vol. 38, pp. 379-385, 1971.

[3] Y.H. Hu and S. Naganathan, “An Angle Recoding Method forCORDIC Algorithm Implementation,”IEEE Trans. Computers,vol. 42, no. 1, pp. 99-102, Jan. 1993.

[4] S. Wang and E.E. Swartzlander, Jr., “Critically Damped CORDIC Algorithm,”Proc. the 37th Midwest Symp. Circuits and Systems,pp. 236-239, Aug. 1994.

[5] E. Antelo, J. Villalba, J.D. Bruguera, and E.L. Zapata, “High Performance Rotation Architectures Based on the Radix-4 CORDIC Algorithm,”IEEE Trans. Computers, vol. 46, no. 8,pp. 855-870, Aug. 1997.

[6] C. Li and S.G. Chen, “A Radix-4 Redundant CORDIC Algorithm with Fast on-Line Variable Scale Factor Compensation,”Proc. Int’l Symp. Circuits and Systems,pp. 639-642, June 1997.

[7] J. Duprat and J. Muller, “The CORDIC Algorithm: New Results for Fast VLSI Implementation,” IEEE Trans. Computers, vol. 42, no. 2, pp. 168-178, Feb. 1993.

[8] D.S. Phatak, “Double Step Branching CORDIC: A New Algorithm for Fast Sine and Cosine Generation,”IEEE Trans. Computers,vol. 47, no. 5, pp. 587-602, May 1998.

[9] S. Wang, V. Piuri, and E.E. Swartzlander, Jr., “Hybrid CORDIC Algorithms,”IEEE Trans. Computers,vol. 46, no. 11, pp. 1202-1207,Nov. 1997.

[10] D. Timmermann, H. Hahn, and B.J. Hosticka, “Low Latency Time CORDIC Algorithms,”IEEE Trans. Computers,vol. 41, no. 8, pp. 1010-1015, Aug. 1992.

[11] S. Wang, V. Piuri, and E.E. Swartzlander, Jr., “A Unified View of CORDIC Processor Design,”Proc. Midwest Symp. Circuits and Systems,1996.

[12] J.A. Arbaugh, “Table Look-Up CORDIC : Effective Rotations Through Angle Partitioning,” PhD dissertation, Univ. of Texas at Austin, 2004.