IJCSC International Journal of Computer Science & Communication Vol. 1, No. 2, July-December 2010, pp. 59-63

Design and Implementation of a Dynamic CORDIC Processor

J.M.Rudagi[1], Basavaraj G. Chougula[2] & S.Subharaman[3]

[1-2]Department of Electronics & Communication Engineering, VTU University, Belgaum, India
[3]Department of Electronics & Communication Engineering, Shivaji University, Kholapur, India
Email: [1]js_itti@yahoo.co.in, [2]basavaraj.chougula@gmail.com

ABSTRACT

This paper presents a Dynamic CORDIC Processor that uses variable radices for trigonometric function computation. The CORDIC algorithm is a well known hardware algorithm for computing various elementary functions. When we use these algorithms in real time signal processing significant reduction in processing latency is required, due to its sequential nature. The basic principle of CORDIC processor presented in this paper is to dynamically change the radix of computation during operation. This yields a reduction in the number of iterations by 34% for 64-bit precision arithmetic.

Keywords: Computer Arithmetic, CORDIC Algorithm, VLSI, Digital Signal Processing

1. INTRODUCTION

Present-day real-time signal processing applications require efficient hardware support not only for multiply and add operation but also for other elementary functions including trigonometric functions. The Coordinate Rotation Digital Computer (CORDIC) algorithm [1],[2] is a known VLSI oriented hardware algorithm for such applications. For real time applications, significant reduction in processing latency is essential. As the sequential nature of the CORDIC algorithm itself: it takes O(n) steps for evaluating a function in n-bit precision.

Recently, there are various proposals for fast CORDIC algorithms. One of the most effective ways to accelerate the CORDIC method is to use redundant number representations such as signed digit (SD) and carry save [4],[5]. Another method for fast CORDIC computation is to introduce parallel prediction of rotation direction [6]-[8]. In general, prediction techniques make possible the fast operation of basic CORDIC iteration step. However, a number of prediction stages must be added among basic iteration stages to use prediction technique. This causes irregular circuit structures that are not suitable for VLSI implementation.

Any approach mentioned above does not mean the reduction in the number of iterations itself. It is likely that the development of high radix algorithm is essential for reducing the number of CORDIC iterations. These algorithms require additional scale correction stages to compensate the length of the rotated vector, which causes severe hardware overhead compared with the radix-2 counterparts.

Addressing these problems, the constant-scale-factor radix 2-4-8 CORDIC algorithm for fast vector rotation was used in [11]. This algorithm employs three different radices 2, 4 and 8 to achieve fast convergence. As a result, number of iterations can be reduced in comparison with the conventional redundant CORDIC algorithm.

2. RADIX2 ALGORITHMS

The name CORDIC stands for Coordinate Rotation Digital Computer. Volder [1] developed the underlying method of computing the rotation of a vector in a Cartesian coordinate system and evaluating the length and angle of a vector. The CORDIC method was later expanded for multiplication, division, logarithm, exponential and hyperbolic functions. The various function computations were summarized into a unified technique in [2].The resulting vector z_n of the rotation of a vector $[x_0, y_0]^T$ by an angle θ in Cartesian coordinates can be computed by the following matrix operation [3]:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (1)$$

Using the identity: $\cos\theta = 1/\sqrt{1+\tan^2\theta}$ and factoring out cos θ equation (1) can be modified as follows:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{\sqrt{1+\tan^2\theta}} \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (2)$$

In the CORDIC method, the rotation by an angle θ is implemented as an iterative process, consisting of micro-rotations during which the initial vector is rotated by pre-determined step angles α_i. Any angle θ can be represented to certain accuracy by a set of n step angles α_i. Specifying a direction of rotation or sign d_i, the sum of the step angles α_i approximates a given angle θ as follows:

$$\theta = \sum_{i=0}^{n-1} d_i \alpha_i \ , \ d_i \in \{-1, 1\} \quad (3)$$

The sign of the difference between the angle θ and the partial sum of step angles $\theta - \sum_{j=0}^{i-1} d_j \alpha_j$ controls the sign d_i of the step angles α_i. An auxiliary variable z_i is introduced that contains the accumulated partial sum of step angles and is used to determine the sign of the next micro-rotation. To simplify the computation of the matrix product given by (2), the step angles α_i are chosen such that tan α_i represents a series of powers of 2:

$$\tan \alpha_i = 2^{-i}, \quad i = 0, 1, 2, \ldots, n-1 \tag{4}$$

The CORDIC method can be employed in two different modes, known as the "rotation" mode and the "vectoring "mode. In the rotation mode, the co-ordinate components of a vector and an angle of rotation are given and the co-ordinate components of the original vector, after rotation through a given angle, are computed. In the vectoring mode, the co-ordinate components of a vector are given and the magnitude and angular argument of the original vector are computed [Vold59].

The rotation mode of the CORDIC algorithm has three inputs that are initialized to the co-ordinate components of the vector x_0, y_0 and the angle of rotation $z_0 = \theta$ and is described by the following iteration equations:

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \arctan 2^{-i} \end{aligned} \tag{5}$$

$$\text{Where } d_i = \begin{cases} -1 \text{ if } z_i < 0 \\ +1 \text{ if } z_i \geq 0 \end{cases} \text{ and } i = 0, 1, 2, \ldots, n-1 \tag{6}$$

The outputs of the rotation mode x_n, y_n and z_n are given by the following expressions, x_n and y_n being the co-ordinates of the rotated (by the angle θ) vector:

$$\begin{aligned} x_n &= K_n (x_0 \cos z_0 - y_0 \sin z_0) \\ y_n &= K_n (y_0 \cos z_0 + x_0 \sin z_0) \\ z_n &= 0 \end{aligned}$$

$$\text{Where } K_n = \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}} \tag{7}$$

A CORDIC micro-rotation is not a pure rotation but a rotation-extension. The constant K_n, given by (7), is referred to as a scale factor, and represents the increase in magnitude of the vector during the rotation process. When the number of iterations/micro-rotations is fixed the scale factor is a constant approaching the value of 1.647 as the number of iterations goes to infinity.

The elementary functions sine and cosine can be computed using the rotation mode of the CORDIC algorithm if the initial vector is of unit length and is aligned with the abscissa. The computation of sin θ and cos θ is based on equations (5) and (6) with input values $x_0 = 1$, $y_0 = 0$ and $z_0 = \theta$. The outputs after n iterations are as follows:

$$\begin{aligned} x_n &= K_n (x_0 \cos\theta - y_0 \sin\theta) = K_n \cos\theta \\ y_n &= K_n (y_0 \cos\theta + x_0 \sin\theta) = K_n \sin\theta \\ Z_n &= 0 \end{aligned} \tag{8}$$

An additional operation of division is required to obtain the values of sin θ and cos θ from (8) as a result of the increase in magnitude of the vector by the factor K_n during rotation. However, since the scale factor is a constant for a given number of iterations n, the operation of division can be eliminated by setting the magnitude of the initial vector to the reciprocal value of the scale factor, i.e. $x_0 = 1/K_n$.

3. HIGH-RADIX CORDIC ALGORITHMS

In the following, the value of radix 'r' is assumed to be a power of 2, i.e. $r = 2^m$, where 'm' is an integer that is not less than 1.The radix-r CORDIC algorithm is given by the following recursive equations:

$$\begin{aligned} X_i &= (X_{i-1} - d_i . r^{-i} Y_{i-1}) \\ Y_i &= (Y_{i-1} + d_i . r^{-i} X_{i-1}) \end{aligned} \tag{9}$$

$$Z_i = Z_{i-1} - \alpha_i$$

$$\alpha_i = \tan^{-1}(d_i . r^{-i}) \qquad (i = 1, 2 \ldots n),$$

Where, 'i' denote the iteration step, α_i is the angle of i^{th} rotation, and d_i is the i^{th} rotation digit.

In the high-radix CORDIC rotation, the i^{th} scale factor K_i given by,

$$K_i = \Pi\, k_i = \Pi \text{ sqrt } (1 + d_i^2\, r^{-2i})$$

Where k_i = sqrt $(1 + d_i^2\, r^{-2i})$ $i = 1, 2 \ldots . n$.

It varies depending on the choice of rotation digit d_i. This variable scale factor causes significant overhead. Addressing this issue, On-the-fly scale correction method based on the Taylor expansion is used. The scale factor of each iteration step is forced to 1.This is done by scaling the ith vector (Xi, Yi) by the factor of $1/K_i$ at every step. This can be written as,

$$\begin{aligned} X_i &= (X_{i-1} - d_i . r^{-i} Y_{i-1}) / K_i \\ Y_i &= (Y_{i-1} + d_i . r^{-i} X_{i-1}) / K_i \end{aligned} \tag{10}$$

Using Taylor expansion of $1/K_i$, we can rewrite these equations as,

$$X_i = X_{i-1} - d_i . r^{-i} Y_{i-1} - \tfrac{1}{2} . d_i^2 r^{-2i} X_{i-1} + \tfrac{1}{2} . d_i^3 r^{-3i} Y_{i-1} + 3/8 . d_i^4 r^{-4i} X_{i-1} - \ldots . \tag{11}$$

$Y_i = Y_{i-1} + d_i.r^{-i}X_{i-1} - \frac{1}{2}.d_i^2 r^{-2i}Y_{i-1} - \frac{1}{2}.d_i^3 r^{-3i}X_{i-1} + 3/8.d_i^4 r^{-4i}Y_{i-1} - \ldots..$

Consider the approximation of the above eqn. (6) and (7) with a small number of terms. In the case of radix-4 CORDIC with 64-bit precision of 2^{-64}, the 4th term becomes below the precision of 2^{-64} after 12 iterations. Similarly, the 3rd term becomes negligible after 18 iterations. As number of iterations increases, the number of negligible terms increases. Accordingly, the scale correction is simplified. Table 1.summarizes the step at which the higher order terms become negligible in radix-4 and radix-8 CORDIC algorithm with 64-bit precision. So, the problem is to reduce the number of iterations with minimum scale-correction overhead.

Table1
Iteration Step After Which the Term is Negligible in 64-bit Precision

Radix	2nd term	3rd term	4th term	5th term
4	Step33	Step18	Step12	Step9
8	Step23	Step12	Step 8	Step6

4. RADIX 2-4-8 ALGORITHMS

On the basis of above analysis, we propose the Radix 2-4-8 algorithm. This changes the radix value 'r' of CORDIC Computation during its execution. This yields a reduction in latency, while keeping the cost of scale correction sufficiently low. The computation flow of the radix 2-4-8 CORDIC is shown in Fig.1.

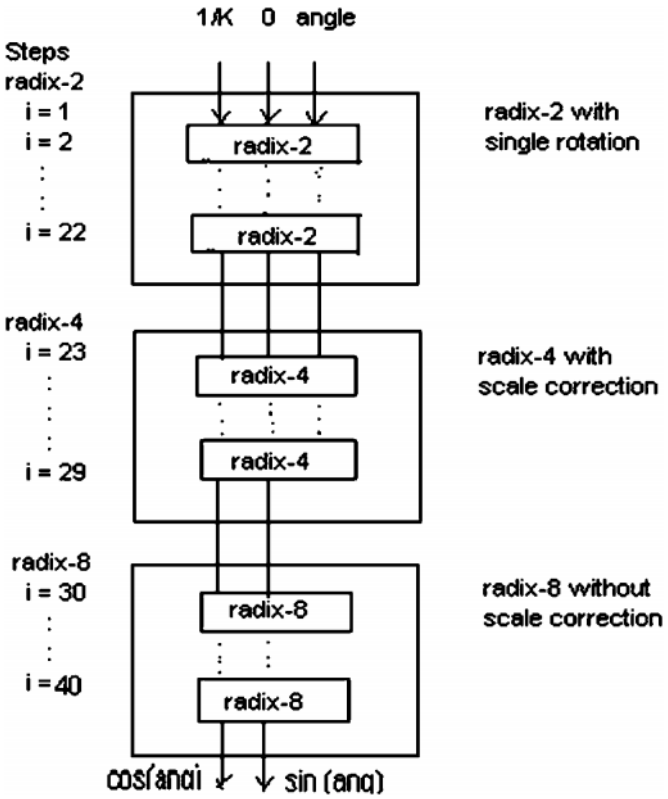


Fig.1: Computation Flow of the Radix 2-4-8 CORDIC

The summaries of Radix 2-4-8 CORDIC algorithm are:

1. Radix-2 CORDIC with single rotation method for steps $1 \le i \le 22$.

   Select $d_i = \{+1, -1\}$ according to Z_{i-1}

   Coordinate calculation:

   $X_i = (X_{i-1} - d_i.2^{-i}Y_{i-1})$

   $Y_i = (Y_{i-1} + d_i.2^{-i}X_{i-1})$

   Angle calculation:

   $Z_i = Z_{i-1} - \alpha_i$

   $\alpha_i = \tan^{-1}(d_i.r^{-i})$ $(i = 1, 2\ldots n)$,

2. Radix-4 CORDIC with scale correction for steps $23 \le i \le 29$.

   $j = i - 11$;

   Select $d_i = \{0, \pm 2, \pm 4\}$

   Coordinate calculation:

   $X_i = X_{i-1} - d_i.4^{-j}Y_{i-1} - \frac{1}{2}.d_i^2 4^{-2j}X_{i-1}$

   $Y_i = Y_{i-1} + d_i.4^{-j}X_{i-1} - \frac{1}{2}.d_i^2 4^{-2j}Y_{i-1}$

3. Radix-8 CORDIC without scale correction for steps $30 \le i \le 40$.

   $k = i - 18$;

   Select d_i from {0, ±1,±2… ±5}

   Coordinate calculation:

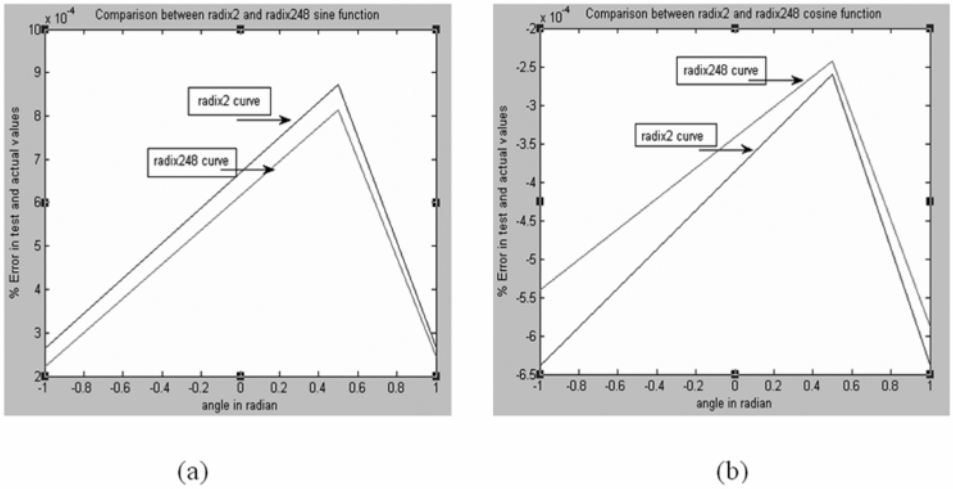
   $X_i = X_{i-1} - d_i.8^{-k}Y_{i-1}$

   $Y_i = Y_{i-1} + d_i.8^{-k}X_{i-1}$

End.

5. RESULTS AND DISCUSSION

The proposed Dynamic CORDIC Processor generates sine and cosine of angle after 40 iterations. The algorithm used approximates the sine and cosine of angle $-\pi/2 < (ang) < \pi/2$. The Xilinx simulation results are hexadecimal or binary values. The trigonometric functions are manipulated with MATLAB version 7.0.0.19920(R14). The radix2 and radix248 simulation results are compared. The percentage error in actual and test results of sine and cosine of angle (rad.) are given below.



(a) (b)

Fig. 2: Comparison between Radix2 and Radix248 for (a) sine; (b) cosine Function

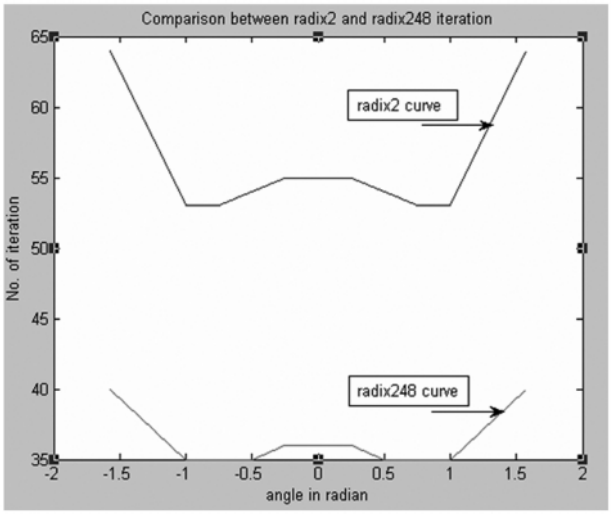


Fig.3: Comparision between Radix2 and Radix248 Processing Iteration

Fig.3 shows the proposed dynamic processor takes 35 iterations compared to radix2 unit, which took 53 iterations for angle 1rad. This shows a reduction in no. of iteration by 34%.

6. CONCLUSION

The analysis of proposed Dynamic CORDIC Processor have shown that the use of higher radices 2, 4 and 8 make possible reduction in number of iterations by 34%. The number of iterations reduced means total computation time is also reduced compared to conventional radix-2 CORDIC.

REFERENCES

[1] J.E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Trans. Electron. Compute*, **EC-8**, pp.330-

334. 1959.

[2] J.S. Walther, "A Unified Algorithm for Elementary Functions", *Spring Jo Computer Conf.*, pp.379-385, 1971.

[3] P.Pirsch. *Architectures for Digital Signal Processing*, John Wiley & Sons, 1998.

[4] M.D.Ercegovac, T.Lang, "Redundant and On-line CORDIC: Apprication to Matrix Triangularization and SVD", *IEEE Trans,. Comput*, **39**, No.6, pp.725-740, June 1990.

[5] N.Takagi, T.Asada, S.Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation, *IEEE Trans.Comput.*, **40**, No.9, pp.989-995, Sept.1991.

[6] P.W.Baker, "Suggestion for a Fast Binary Sine/Cosine Generator", *IEEE Trans.Comput.*, **C-25**, pp.1134-1136, Nov.1976.

[7] D.Timmermann, H.Hahn, and B.J.Hosticka, "Low Latency Time CORDIC Algorithms", *IEEE Trans.Comput.*, **41**, No.8, pp.1010-1015, Aug.1992.

[8] E.Antelo, J.D.Bruguera, J.Villalba, and E.L.Zapata, "Redundant CORDIC Rotator based on Parallel Prediction", *Proc.12th IEEE Symp. Computer Arithmetic*, pp.172-179, July 1995.

[9] R.Hamill and J.V.McCanny, "Constant Scale Factor On-line CORDIC Algorithm in the Circular Coordinate System", *VLSI Signal Processing VIII*, pp.562-571, Oct.1995.

[10] E.Antelo, J.D.Bruguera, and E.L.Zapata, "Unified Mixed Radix 2-4 Redundant CORDIC Processors", *IEEE Trans.Comput.*, **45**, No.9, pp.1068-1073, Sept.1996.

[11] T.Aoki, H.Nogi, and T.Higuchi, "High-radix CORDIC Algorithm for VLSI Signal Processing", *Proc.1997 IEEE Workshop on Signal Processing Systems*, pp.183-192, Nov.1997.