FPGA Implementation of Efficient Pipelined CORDIC Processor for DSP Applications

Dickson Warepam
Dept. of Instrumentation and Electronics Engineering
Dayananda Sagar College of Engineering
Bangalore, India

U. S. Pavitha
Dept. of Instrumentation and Electronics Engineering
Dayananda Sagar College of Engineering
Bangalore, India

Abstract— **The CORDIC processor plays a vital role in real time Digital signal processing applications where it reduces the complexity in terms of area and speed. The traditional CORDIC Processors are based on basic algorithm where the ranges of angle calculations are limited. In this paper, we modify the existing CORDIC architecture to compute for angles 0 to 360 degrees. The modified CORDIC is designed by using pre-processing block with existing CORDIC block. The comparator is used as pre-processing unit to calculate angles greater than 90° by using trigonometric functions. The output of the comparator is fed to CORDIC unit to obtain the values of Sin θ and Cos θ for angle 0° to 360° and also the CORDIC architecture is modified for pipelining design path. The proposed CORDIC can be used for signal processing applications for angle requirements of 0 to 360 degrees.**

Keywords—CORDIC processor, Pre-processing block, Pipelining design etc.

I. INTRODUCTION

CORDIC is an acronym for Coordinate Rotation Digital computer. CORDIC algorithm is used in various applications such as Fast Fourier transform, SVD, and evaluating trigonometric functions. In digital signal processing, microprocessors are often not fast enough for truly demanding DSP tasks. Algorithm optimized for these microprocessor based systems do not usually map well into hardware. CORDIC algorithm uses reduced hardware architecture and faster computation of the elementary functions by using simple add-shift operations.

CORDIC algorithm was implemented for the first time by Jack E. Volder [1] in 1959. It is an iterative algorithm for the calculation of the rotation of a two dimensional vector in linear, circular, or hyperbolic coordinate systems. The rotation is carried out by means of a sequence of iterations in each of which one rotation over a given elementary angle (micro-rotation) is evaluated by means of addition and shift operations. The rotated vector is scaled by a constant factor that must be compensated. The implementations of the CORDIC algorithm [2] have been carried out on word serial architectures using conventional non redundant arithmetic with radix-2 micro-rotations and fixed point internal format. In order to extend the number of basic functions, Walter proposed a uniform CORDIC arithmetic in 1971[3]. In 2004, Juang and others proposed a modified CORDIC arithmetic with paralleling, it used the methods which were binary-to-bipolar recoding (BBR) and micro-rotation angle recoding (MAR)[4]. The modified CORDIC arithmetic could speed up the iterations and have a higher precision than before. CORDIC arithmetic is an iterative algorithm for the calculation of the rotation of a two-dimensional vector in linear, circular and hyperbolic coordinate systems [5]. Each system has two ways to be done which are the rotation mode and the vectoring mode. In nowadays, the proper control of the tradeoff between latency time and the hardware complexity enables the designer to use higher radix CORDIC unit in place of the conventional radix-2 CORDIC unit in high speed application purposes. A CORDIC architecture that suggests a circuit for scale factor correction is presented in [6]. Some efforts have also been made to implement the CORDIC architectures on an FPGA [7].

By making slight adjustments to the initial conditions and the LUT values, it can be used to efficiently implement trigonometric, hyperbolic, exponential functions, coordinate transformations etc. using the same hardware. Since it uses only shift-add arithmetic, the VLSI implementation of such an algorithm is easily achievable

II. BASIC CORDIC ALGORITHM

The main objective of this algorithm is to eliminate the ROM and a large barrel shifter in the hardware implementation. The CORDIC algorithm operates either in, rotation mode or vectoring mode, following linear, circular or hyperbolic coordinate trajectories. In this paper, we focus on rotation mode CORDIC using circular trajectory.

A. *The CORDIC Algorithm*

As shown in Fig. 1, the rotation of a two-dimensional vector $\mathbf{p_0} = [\mathbf{x_0}\ \mathbf{y_0}]$ through an angle θ, to obtain a rotated vector $\mathbf{p_n} = [\mathbf{x_n}\ \mathbf{y_n}]$ could be performed by the matrix product ,$\boldsymbol{p_n = Rp_0}$, where R is the rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \qquad (1)$$

By factoring out the cosine term in (1), the rotation matrix R can be rewritten as

$$\boldsymbol{R} = [((1+tan^2\theta)^{-\frac{1}{2}}]\begin{bmatrix}1 & -tan\theta\\ tan\theta & 1\end{bmatrix} \qquad (2)$$

and can be interpreted asa product of a scale-factor $\boldsymbol{K}=[\boldsymbol{1}(\boldsymbol{tan^2\theta})^{-\frac{1}{2}}]$ with a pseudo-rotation matrix $\boldsymbol{R}_c$, given by

$$\boldsymbol{Rc} = \begin{bmatrix}1 & -tan\theta\\ tan\theta & 1\end{bmatrix} \qquad (3)$$

The pseudo-rotation operation rotates the vector by an angle θ and changes its magnitude by a factor $\boldsymbol{k = cos\theta}$, to produce a pseudo-rotated vector $\boldsymbol{p'}=\boldsymbol{R}_c\boldsymbol{p}_0$.

To achieve simplicity of hardware realization of the rotation, the key ideas used in CORDIC arithmetic are to (i) decompose the rotations into a sequence of elementary rotations through predefined angles that could be implemented with minimum hardware cost; and (ii) to avoid scaling, that might involve arithmetic operation, such as square-root and division. The second idea is based on the fact the scale-factor contains only the magnitude information but no information about the angle of rotation.

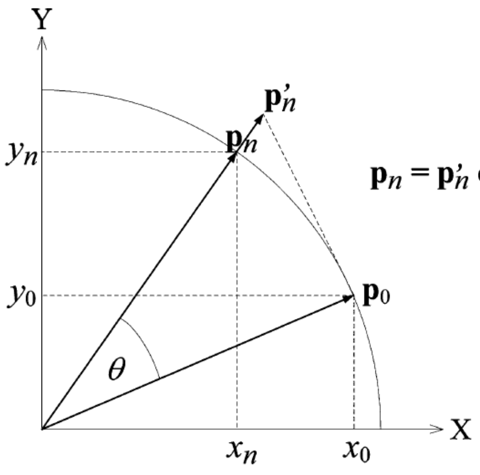

Fig .1. Rotation of vector on a two-dimensional plane

The CORDIC algorithm performs the rotation iteratively by breaking down the angle of rotation into a set of small pre-defined angles, $\boldsymbol{\alpha_i}$= ***arctan(2^{-i}),*** so that $\boldsymbol{tan\alpha_i=2^{-i}}$ could be implemented in hardware by shifting through bit locations. Instead of performing the rotation directly through an angle, CORDIC performs it by a certain number of micro-rotations through angle α_i that satisfies the CORDIC convergence theorem[8]: σ_i $-\sum_{j=i+1}^{n-1}\sigma_i < \alpha_{n-1}, \forall\, i, i = 0,1,\dots,n-2$., where

$$\theta = \sum_{i=0}^{n-1}\sigma_i\alpha_i \qquad (4)$$

$$\sigma_i = \pm 1 \qquad (5)$$

But, the decomposition according to (10) could be used only for -1.74329≤ $\theta \le$ 1.74329 (called the "convergence range")since$\sum_{i=0}^{\infty}(\alpha_i) = 1.73296$.. Therefore, the angular decomposition of (10) is applicable for angles in the first and fourth quadrants. To obtain on-the-fly decomposition of angles into the discrete base α_i, one may otherwise use the non-restoring decomposition [9]

$$\omega_0=0 \text{ and } \omega_{i+1} = \omega_i - \sigma_i.\alpha_i \qquad (6)$$

with σ_i= 1 if and $\omega_i \ge 0$ and σ_i = -1 otherwise, where the rotation matrix for the iteration corresponding to the selected angle is given by

$$R(i) = k_i\begin{bmatrix}1 & -\sigma, 2^{-i}\\ \sigma, 2^{-i} & 1\end{bmatrix} \qquad (7)$$

Ki=($1/\sqrt{1+2^{-2i}}$) being the scale-factor, and the pseudo-rotation matrix

$$R_c(i) = \begin{bmatrix}1 & -\sigma, 2^{-i}\\ \sigma, 2^{-i} & 1\end{bmatrix} \qquad (8)$$

Note that the pseudo-rotation matrix $\mathbf{R_c}$(i) for the *i*th iteration alters the magnitude of the rotated vector by a scale-factor Ki = ($1/\sqrt{1+2^{-2i}}$) during the *i* th micro-rotation, which is independent of the value of σ_i(direction of micro-rotation) used in the angle decomposition.

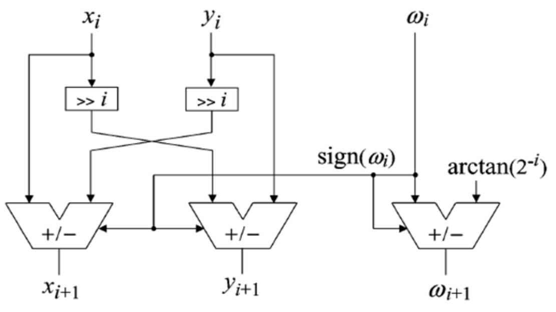

Fig .2. Hardware implementation of a CORDIC

III. MODIFIED CORDIC ALGORITHM

The modified CORDIC is designed to compute angles from 0° to 360° by using pre-processing block and existing CORDIC block as shown in the Fig. 1a The comparator is used as pre-processing unit for angles greater than 90° by using trigonometric functions as shown in Fig. 1(b). The output of the comparator is fed to CORDIC unit to obtain the values of Sinθ and Cosθ for angle 0° to 360°. The parallel CORDIC architecture used in modified CORDIC unit is shown in Fig. 2.

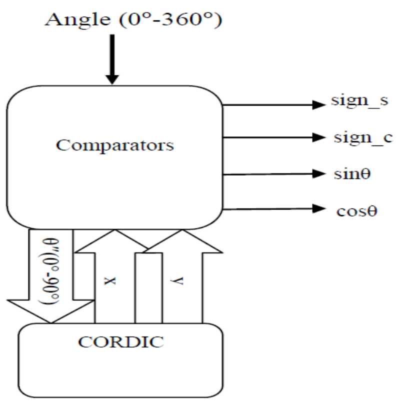

Fig.3. Modified CORDIC

As to operate from 0 to ± 360^0, the comparator output is given from 0 to 90^0

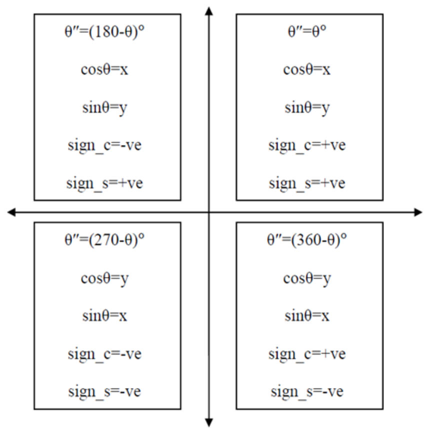

Fig.4. Pre-processing Unit

IV. HARDWARE IMPLEMENTATION

In this type of architecture, all the iterations take place in a single clock cycle. The architecture is shown below figure 5.

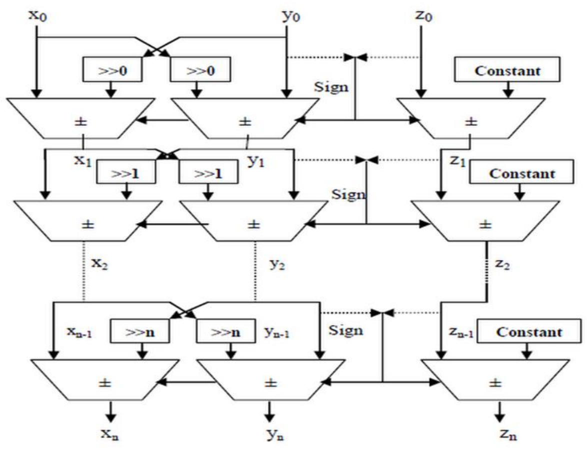

Fig .5. Parallel CORDIC Architecture

It has considerable delay, but processing time is reduced as compared to the iterative process. Shifters are of fixed size and so can be implemented in the wiring. Constants can be hardwired instead of requiring storage space.

The basic CORDIC equations are given in equation $x_{i+1} = x_i \pm y_i.d_i.2^{-i}$ (9)

$$y_{i+1} = y_i \pm x.d_i.2^{-i} \quad (10)$$

$$z_{i+1} = z_i - d_i.\tan^{-1}(2^{-i}) \quad (11)$$

Where $d_i = -1$ if z_i is negative (-ve)

$d_i = +1$ if z_i is positive (+ve)

K~1.6467605 is called *scaling factor*.

The equation (2) is modified by eliminating 2^{-i} term using equation (3) which reduces the hardware requirements.

$$x_{i+1} = x_i \pm y_i.d_i.(\gg i) \quad (12)$$
$$y_{i+1} = y_i \pm x.d_i.(\gg i) \quad (13)$$

V. FPGA IMPLEMENTATION AND COMPARISON

The proposed hardware architectures is being coded and synthesized using Xilinx ISE 14.5. On comparing the proposed method with the existing techniques, the proposed technique has certain advantage and in some area it is more efficient. The comparison is shown in the following table

TABLE I. COMPARISION WITH EXISTING TECHNIQUE

	Existing algorithm	Proposed algorithm
Range of angle	-99.9° to +99.9°	0 to ±360
Radix	4	2
frequency	155.174 MHz	>300 MHz
Error	10%	3-6%

VI. CONCLUSION

The CORDIC algorithm is a powerful and widely used tool for DSP applications. So, the implementation of DSP using CORDIC algorithm on FPGA is the need of the day as the FPGAs can give enhanced speed at low cost with a lot of flexibility. This is due to the fact that the hardware implementation of a lot of multipliers can be done on FPGA. The operation with angle from 0 to ± 360 degree is possible.

REFERENCES

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

[2] J.R. Cavallaro and F.T. Luk, "CORDIC Arithmetic for an SVD Processor," J. Parallel and Distributed Computing, No. 5, pp. 271-290,1988.

[3] Walther J S. "A unified algorithm for elementary functions" Spring Joint Computer Conference, New York, USA. 1971, pp.297–385.

[4] Juang T B, Hsaio S F, Tsai M Y. Para-CORDIC: Parallel CORDIC rotation algorithm. Circuits and Systems: Regular Papers, 2004, 51(8): 1515–1524.

[5] Kaushik B, Rakesh B., "Architectural design and FPGA implementation of radix-4 CORDIC processor", Microprocessors and Microsystems, 2010: 96–101.

[6] M.G.B. Sumanasena, "A scale factor correction scheme for the CORDIC algorithm", IEEE Transactions on Computers, Vol. 57, pp. 1148–1152, 2008.

[7] F.Angarita, A. Perez-Pascual, T. Sansaloni, and J. Valls. “Efficient FPGA implementation of CORDIC algorithm for circular and linear coordinates”. In Proc. 2005 IEEE Int. Conf. Field Prog. Logic Appl., pp. 535–538, 2005.

[8] J. S. Walther, “A unified algorithm for elementary functions,” in *Proc. 38th Spring Joint Computer Conf.*, Atlantic City, NJ, 1971, pp. 379–385.

[9] 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.

[10] J. S.Walther, “The story of unified CORDIC,” *J. VLSI Signal Process.*, vol. 25, no. 2, pp. 107–112, June 2000