International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 11, November - 2013

A Verilog Implementation of Fixed Point Cordic Processor

Ashish Gambhir, Susmita Samanta, Sunil Kumar
Department of ECE, Dronacharya College of Engineering, Gurgaon

Abstract - **There are two types of representations for real numbers that is fixed point and floating point. This paper compares the original CORDIC for sine-cosine generation on the basis of their area for 16-bit, 24-bit and 32-bit fixed point numbers. The advantage of floating-point representation over fixed-point (and integer) representation is that it can support a much wider range of values and precision. A high speed Original CORDIC for sine cosine generation for 24-bit, 28-bit and 32-bit (single precision IEEE 754) floating point numbers is also synthesized and the results have been compared. It is shown that there is 2%, 3% and 5% utilization of slice registers for 16-bit, 24-bit and 32-bit fixed point CORDIC.**

Keywords-Coordinate Rotation Digital Computer(CORDIC), Fixed point, Floating Point.

I. INTRODUCTION

CORDIC is an acronym for COrdinate Rotation Digital Computer. It is a class of shift and add algorithms for rotating vectors in a plane, which is usually used for elegant computation of several transcendental functions such as trigonometric functions, multiplication, division and conversion between binary and mixed radix number systems of DSP applications, such as Fourier Transform. Two another functions are the absolute magnitude of a vector and the corresponding phase angle (arctangent computation). These functions can be evaluated using the CORDIC in its angle accumulation or vectoring mode. On VLSI implementation level, the area also becomes quite important as more area means more system cost. In this paper, area efficient CORDIC algorithm is implemented for calculations of trigonometric functions. Verilog HDL is used to implement technology-independent design. There are two types of representations for real numbers that is fixed point and floating point. The comparison of original CORDIC for sine-cosine generation on the basis of their area for 16-bit, 24-bit and 32-bit fixed point numbers have been synthesized and discussed. A high speed Original CORDIC for sine cosine generation for 24-bit, 28-bit and 32-bit (single precision IEEE 754) floating point numbers is also synthesized. The rest of the paper is structured as follows: Section II describes the theory of the CORDIC with its application areas, conventional CORDIC, and Section III describes the fixed point CORDIC algorithms. Section IV explains the two types of representations for real numbers. Section V compares the results obtained in terms of delay and hardware utilization when the number is represented in 16-bit, 24-bit and 32-bit fixed point format. Section VI gives the conclusion for the paper.

II. COORDINATE ROTATION DIGITAL COMPUTER(CORDIC)

A. *Introduction*

The Coordinate Rotation DIgital Computer (CORDIC) algorithm [1], [2] has been used for many years for efficient implementation of vector rotation operations in hardware. It is executed merely by table look-up, shift, and addition operations. Thus, the corresponding hardware can be implemented in very economic fashion. Subsequently, it has been applied for many performance demanding applications in digital signal processing (DSP), image processing, and video technology like Fast Fourier Transform (FFT) [3], [4], Discrete Hartley Transform (DHT) [4], [5], Discrete Cosine Transform (DCT) [4], [6], [14] Discrete Sine Transform (DST)[4], Hough Transform (HT) [7]–[9], [12], graphics application [10], [11], and motion vector estimation[12].

In essence, a CORDIC can be operated in two different modes: the rotation and the vectoring mode. In the former mode of operation, given a vector with initial coordinate(x0, y0) and a target rotation angle(z0), the objective is to compute the final coordinate(x1,y1) through a series of backward and forward rotation of the vector in an iterative manner. In the vectoring mode, the objective is to compute the magnitude and the phase angle of a vector given its initial and final coordinates. Table I shows the different modes of CORDIC operations in different coordinate systems where K_h and K_c are two constants known as scale factors for the hyperbolic and circular coordinate systems, respectively. However, despite its attractiveness, the conventional CORDIC algorithm has some drawbacks, such as slow speed, requirement of compensation of a bulk scale factor, and limited convergence range.

Mode of Operation	$y \rightarrow 0$ (Vectoring)	$z \rightarrow 0$ (Rotation)
Hyperbolic	$x1 = K_h\sqrt{x0^2 - y0^2}$ $z1 = z0 + \tanh^{-1}(y0/x0)$ $\lvert\tanh^{-1}(y0/x0)\rvert <= 1.1182$	$x1 = K_h[x0\cosh(z0) + y0\sinh(z0)]$ $y1 = K_h[x0\sinh(z0) + y0\cosh(z0)]$ $\lvert z0\rvert <= 1.1182$
Linear	$x1 = x0$ $z1 = z0 + (y0/x0)$ $\lvert(y0/x0)\rvert <= 1$	$x1 = x0$ $y1 = y0 + (x0z0)$ $\lvert z0\rvert <= 1$
Circular	$x1 = K_c\sqrt{x0^2 + y0^2}$ $z1 = z0 + \tanh^{-1}(y0/x0)$ $\lvert\tanh^{-1}(y0/x0)\rvert <= 1.7433(99.9^o)$	$x1 = K_c[x0\cosh(z0) - y0\sinh(z0)]$ $y1 = K_c[x0\sinh(z0) -$

		y0cosh(z0)] |z0| <= 1.7433(99.9°)

Table 1 : Functionality of the Generalized CORDIC

B. Conventional CORDIC

The rotation of a vector $[x0\ y0]^T$ in the Cartesian coordinate system can be described as (considering clockwise rotation)

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x0 \\ y0 \end{bmatrix} \quad (1)$$

Where $[x1\ y1]^T$ is the final vector and Ɵ is the target angle of rotation. In the CORDIC algorithm [13], Ɵ is expressed as the summation of a decreasing sequence of elementary angles α_i so that

$$\Theta = \sum_{i=0}^{b-1} \sigma_i \alpha_i \quad (2)$$

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

Where b is the word length of the machine in which the operation is to be implemented and $\sigma_i \epsilon \{1, -1\}$ is known as the direction of vector rotation for the *i*th iteration. Substituting (2) into (1) and using (3), one may write

$$\begin{bmatrix} x1 \\ y1 \end{bmatrix} = \prod_{i=0}^{b-1} \cos\alpha_i \begin{bmatrix} 1 & \sigma_i 2^{-i} \\ -\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (4)$$

and

$$\sigma_i = \mathrm{Sign}\,[\theta - \sum_{r=0}^{i-1} \alpha_r] \quad (5)$$

$$z_{i+1} = z_i + \sigma_i 2^{-i} \quad (6)$$

Equations (4)–(6) are the basic working equations of the CORDIC rotator operation where $[x_i\ y_i]^T$ and z_i are the intermediate result vector and the residual angle, respectively, at the beginning of the *i*th iteration step. From the hardware implementation point of view, this vector rotation is nothing but a sequence of shift-and-add operations. However, the final result requires a scaling by a factor $\prod_{i=0}^{b-1} \cos\alpha_i$ (K_c in Table I). The scale factor remains a machine constant as long as the index runs through all of the values from 0 to b-1, i.e., when all of the allowed iteration steps are executed. However, if i changes in a different manner, i.e., if some of the allowed iterations are bypassed or repeated in order to achieve a faster convergence rate or a larger convergence range, the scale factor will not remain constant and, for its compensation, one requires extra hardware and comparable post processing cycles.

III. FIXED POINT CORDIC ALGORITHM

Fixed-point Fast Fourier Transform (FFT) units are widely used in digital communication systems. The twiddle multipliers required for realizing large FFTs are typically implemented with the Coordinate Rotation Digital Computer (CORDIC) algorithm to restrict memory requirements. Recent approaches aiming to optimize the bit widths of FFT units while satisfying a given maximum bound on Mean-Square- Error (MSE) mostly focus on the architectures with integer multipliers. They ignore the quantization error of coefficients, disabling them to analyze the exact error defined as the difference between the fixed-point circuit and the reference floating-point model. Radecka et.al. presents an efficient analysis of MSE as well as an optimization algorithm for CORDIC based FFT units, which is applicable to other Linear-Time-Invariant (LTI) circuits as well [13].

IV. NUMBER FORMAT

A number format in computer is the internal representation of numeric values in digital computer hardware and software. Normally, numeric values are stored as groupings of bits, named for the number of bits that compose them. In real life, we deal with real numbers that is numbers with fractional part. In most modern computer we have hardware support for fixed point numbers and floating point numbers for representing real numbers.

A. Fixed point number representation:

Fixed point formatting is useful to represents fractions in binary. In fixed point representation every word has the same number of digits and the binary point is always fixed at the same position. By implementing algorithms using fixed point mathematics a significant improvement in execution speed can be observed because of inherent integer math hardware support in a large number of processor as well as the reduced software complexity for emulated integer multiply and divide. This speed improvement does come at the cost of reduced range and accuracy of the algorithm variables.

Qm.n format: m bit for whole part, n bit for fractional part.

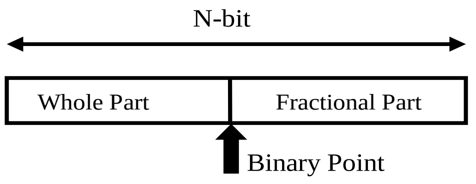


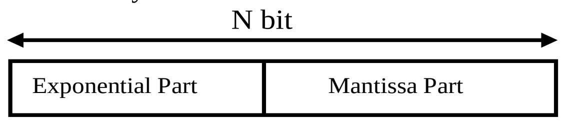
Figure 1: Fixed Point Number Representation

B. Floating point number representation

In computing, floating point describes a method of representing real numbers in a way that can support a wide range of values. Numbers are, in general, represented approximately to a fixed number of significant digits and scaled using an exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

$$\text{Significant digits} \times \text{base}^{\text{exponential}}$$

The term floating point refers to the fact that the radix point (decimal point, or, more commonly in computers, binary point) can "float" that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation. The advantage of floating-point representation is that it can support a much wider range of values and are very accurate.



V. RESULTS

With the angle assumed to be 60^o and the clock signal applied, the results for 16-bit, 24-bit and 32-bit fixed point representation are found as shown in Table 2. 60^o in binary is

Figure 2: Floating Point Number Representation

0001111000000000 for 16-bit fixed point representation, 000111100000000000000000 for 24-bit fixed point representation and 00011110000000000000000000000000 for 32-bit fixed point representation. The design was synthesized into a Xilinx VIRTEX 5 family FPGA.

As can been seen in the Table 2, as we increase the no. of bits the delay keeps on increasing. On the other hand, there is marginal increase in utilization in terms of no. of slice registers.

Table 3: Comparison between 16-bit, 24-bit and 32-bit fixed point CORDIC

	16-bit		**24-bit**		**32-bit**	
	Total	**Utilization**	**Total**	**Utilization**	**Total**	**Utilization**
No. of slice registers	5106/ 207360	2%	6981/ 207360	3%	9532/ 207360	5%
No. of slice LUTs	2916/ 207360	1%	5102/ 207360	2%	11012/ 207360	4%
No. of bonded IOBs	53/1200	4%	67/1200	5%	83/1200	7%
Delay	2.01ns		2.34ns		2.59ns	
Frequency	497.51 MHz		427.35 MHz		386.10 MHz	

VI. CONCLUSION

In this paper, we implemented the conventional CORDIC algorithm for computing the sine and cosine values in fixed point number format. We used 16-bit, 24-bit and 32-bit fixed point representation representation. The comparison in table 2 suggests that as we increase the no. of bits for the number representation, the delay increases. In terms of hardware utilization, there is marginal increase in no. of slices for fixed point representation.

References

[1] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. EC-8, no. 3, pp. 330–334, Sep. 1959.
[2] J. S. Walther, "A unified algorithm for elementary functions," in Proc. Joint Spring Comput. Conf., vol. 38, Jul. 1971, pp. 379–385.
[3] E. F. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC architectures for fast VLSI filtering and array processing," in Proc. ICASSP, 1984, pp. 41 A 6.1–41 A. 6.5.
[4] K. Maharatna, A. S. Dhar, and S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST," Signal Process., vol. 81, pp. 1813–1822, 2001.
[5] E. L. Zapata and F. Arguello, "A VLSI constant geometry architecture for the Hartley and Fourier transform," IEEE Trans. Parallel Distrib. Syst., vol. 3, no. 1, pp. 58–770, Jan. 1992.
[6] M. C. Mandal, A. S. Dhar, and S. Banerjee, "Multiplierless array architecture for computing discrete cosine transform," Computers Elect. Eng., vol. 21, no. 4, pp. 327–333, 1994.
[7] J. D. Bruguera, N. Guil, T. Lang, J. Villalba, and E. L. Zapata, "CORDIC-based parallel/pipelined architecture for hough transform," J. VLSI Signal Process., vol. 12, pp. 207–221, 1996.
[8] D. Timmermann, H. Hann, and B. J. Hosticka, "Hough transform using CORDIC method," Electron. Lett., vol. 25, no. 3, pp. 205–206, 1989.
[9] K. Maharatna and S. Banerjee, "A VLSI array architecture for hough transform," Pattern Recognit., vol. 34, pp. 1503–1512, 2001.
[10] H. Yoshimura, T. Nakanishi, and Y. Yamaguchi, "A 50 MHz CMOS geometrical mapping processor," IEEE Trans. Circuits Syst., vol. 36, no. 10, pp. 1360–1363, Oct. 1989.
[11] K. Maharatna and S. Banerjee, "CORDIC based array architecture for affine transformation of images," in Proc. Int.

Conf. Communications, Computers and Devices, vol. II, Kharagpur, India, Dec. 2000, pp. 645–648.

[12] H. L. Li and C. Charkrabarti, “Hardware design of a 2-D motion estimation system based on hough transform,” IEEE Trans. Circuits Syst. II, vol. 45, no. 1, pp. 80–95, Jan. 1998.

[13] A. Despain, “Fourier Transform Computers Using CORDIC Iterations,” IEEE Transactions on Computers, vol. 23, pp. 993-1001, 1974.

[14] Liyi Xiao and Hai Huang, “A Novel CORDIC based unified architecture for DCT and IDCT”, International Conference on Optoelectronics and Microelectronics (ICOM), pp. 496-500, 2012