Twiddle Factor Angle Generation using Modified Cordic for DSP Applications

P. Rengaprabhu
Associate Professor
Dept. of ECE Oxford Engg. College
Trichy, TN, India

N. Sivarajan
Student (PG)
Dept. of ECE Oxford Engg. College
Trichy, TN, India

B. Keerthana
Assistant Professor
Dept. of ECE ACCET
Karaikudi, TN, India

G. Seetharaman
Professor
Dept. of ECE Oxford Engg. College
Trichy, TN, India

***Abstract*— In this paper an area efficient approach is presented to design and implement a high speed Fast Fourier Transform for DSP applications. The design has been implemented using Modified Coordinate Rotation Digital Computer (CORDIC) algorithm which uses two elementary angles and ROM to store number of microrotation.The proposed design is synthesized and simulated with Quartus II software and implemented on FPGA. Using proposed CORDIC algorithm, enhanced performance is achieved in terms of speed, area and dynamic power constraints.**

Key Words: CORDIC, DSP application, FPGA, FFT, Micro rotation, Twiddle factor

I. INTRODUCTION

CORDIC (COordinate Rotation Digital Computer) which was introduced in 1959 by Jack E. Volder. It is a Hardware Efficient iterative Algorithm for Circular Rotation and to compute trigonometric functions, hyperbolic functions, multiplications, divisions and data type conversions. Compared to other approaches, CORDIC is utilized where hardware multiplier is unavailable. (eg.microcontroller) where the Delay/Hardware cost is comparable to division or square rooting. Two basic CORDIC modes are: the rotation mode and the vectoring mode. For both modes the CORDIC algorithm can be realized as an iterative sequence of additions/subtractions and shift operations, which are rotations by a fixed rotation angle but with variable rotation direction [1]. Due to this simplicity, this algorithm is well suited for VLSI implementation. In this paper modified CORDIC algorithm is used to generate angle for twiddle factor on FFT computation. For FFT applications θ is known in advance which will be used again and again. In these situations, the angle updating equation is evaluated in advance (off line), and the corresponding set of micro rotation is stored; instead of, in the memory. A particular advantage is that there will be no need to implement the angle updating formula, resulting in a cost saving of hardware. This paper is organized as follows: Section II gives the basics of CORDIC algorithm. Section III gives FFT computation and prior work. Section IV explains about the modified CORDIC based algorithm. Section V explains about scaling free CORDIC. Section VI shows the performance of proposed technique and the last section VII concludes the paper followed by references.

II.CORDIC ALGORITHM

The basic CORDIC algorithm can be explained as follows:

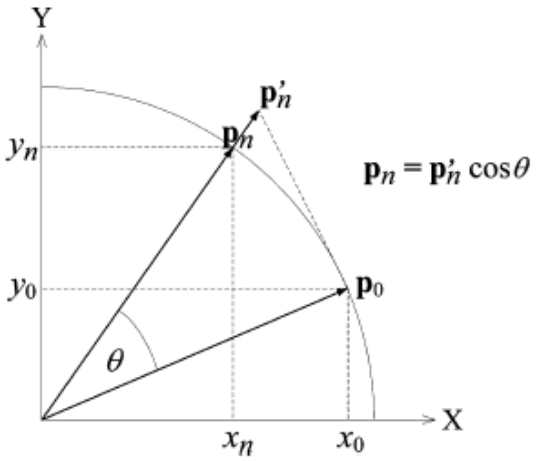

Fig.1 CORDIC rotation

As shown in Fig. 1, the rotation of a vector P0=[x0 y0] through an angle θ, to obtain a rotated vector Pn= [xn yn] could be performed by the matrix product Pn=RP0, where R is the rotation matrix [3].

Each iteration, the matrix product is expressed as $P_{i+1} = RP_i$ and the new coordinate is calculated by:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \text{--- (1)}$$

The rotation matrix will be:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \text{--- (2)}$$

Equation 1 can be rewritten as

$$x_{i+1} = \cos\theta[x - y\tan\theta] \quad \text{--- (3)}$$

$$y_{i+1} = \cos\theta[y + x\tan\theta] \quad \text{--- (4)}$$

It allows only iterative rotations so that

$$\tan\theta = \pm 2^{i}.$$

Then equation 3 and 4 becomes

$$x_{i+1} = \cos(\tan^{-1}(\pm 2^{-i}))[x - yd_i 2^{-i}] \quad \text{--- (5)}$$

$$y_{i+1} = \cos(\tan^{-1}(\pm 2^{-i}))[y + xd_i 2^{-i}] \quad \text{--- (6)}$$

Depending on the direction of rotation, $d_i = \pm 1$, which is determined by

$$z_{i+1} = z_i - d_i \arctan(2^{-i}) \quad \text{--- (7)}$$

Here arctan (2-i) values are pre computed (a0, a1, ...) and it can be scaled to binary number range, e.g. 2π=256.If Zi <0, di =-1.otherwise di =+1; to evaluate Zi, simply use the Zi sign bit (MSB).It will be iteratively repeated for n times to get [xnyn].

By factoring out the cosine term in Eq.2, the rotation matrix R can be rewritten as

$$R=[(1+\tan^2\theta)^{-1/2}]\begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad \text{--- (8)}$$

And can be interpreted as a product of a scale-factor K=[(1+tan2θ)-1/2] with a pseudo rotation matrix Rc,given by

$$R_C = \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \quad \text{--- (9)}$$

The pseudo rotation operation rotates the vector P0 by an angle θ and changes its magnitude by a factor K= $\cos\theta$, which is equivalent to cos (tan-1(2-i)) to produce a pseudo-rotated vector Pn=RcP0.

The cosine is symmetric:

$$\cos(\tan^{-1}(2^{-i})) = \cos(\tan^{-1}(-2^{-i})) \quad \text{--- (10)}$$

Then

$$K = \cos(\tan^{-1}(\pm 2^{-i})) = 1/\sqrt{(1+2^{-2i})} \quad \text{-- (11)}$$

We can compute K offline for all n iterations. It approaches 0.6037, if n goes to infinity. In order to compensate the gain, we have to scale the result with the reciprocal value of the gain:

$$A = \prod_n \sqrt{(1+2^{-2i})} \quad \text{--- (12)}$$

We can compute "A" offline for all n iterations. "A" approaches 1.647, if n goes to infinity. The corresponding architecture for this algorithm is shown below:

The key ideas used in CORDIC to achieve simplicity are: (i) decomposition of the rotations into a sequence of elementary rotations through predefined angles and (ii) avoidance of scaling. The purpose of scaling [2] is to make the final coordinate has the same m-norm as the initial coordinate after rotation. A main research issue is to reduce the computation overhead due to the scaling operation.

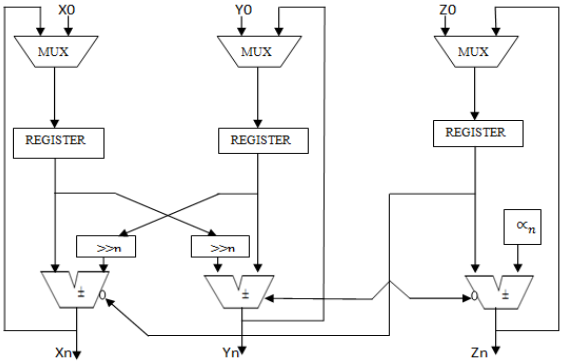

Fig.2 Bit parallel Iterative CORDIC Hardware

III. FAST FOURIER TRANSFORM

An FFT computation is implemented with a complex multiplier but due to demand of higher point FFTs, the size of ROM in this implementation for the twiddle factors becomes the major concern with larger chip area [4, 5]. The aim of this paper is to design an FFT with modified CORDIC algorithms in order to replace complex multipliers through reduced ROM size.

The Fourier transform for N-point is expressed by

$$X(K) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad \text{--- (13)}$$

Where

$W_N = e^{-j2\pi/N}$ is the "twiddle factor". As shown in Eq. 13, the key operation of FFT is "Rotation of input x(n) by an angle corresponds to twiddle factor to get an output X(K)" .It can be related to the CORDIC algorithm without any complex multiplications where old coordinate is rotated by elementary angle corresponds to target angle to get new coordinate. An FFT processor stores the twiddle factors in memory. CORDIC-based FFT doesn't have twiddle factors but stores the rotation angle in a memory bank. For radix-2, N Point, m-bit FFT, mN/2 bits memory needed to store N/2 angles [6]. But the modified CORDIC FFT design needs to store only number of shifts corresponding to micro rotation which is presented in next section where CORDIC uses only two elementary angles [7].

IV. PROPOSED CORDIC ALGORITHM

In FFT, cosine and sine of angles have to be found which lies in the range of 0º to 180. The conventional CORDIC algorithms are used to handle rotations only in the range of angles [-99°, 99°]. Moreover, they are serial in nature and require a ROM to store the lookup table and hardware expensive barrel shifters. So we go for a reduced hardware CORDIC. Our objective is to reduce the complexity which is expected to result in considerable savings in FPGA resources (in comparison to conventional CORDIC). This is designed for rotation mode. The key idea in this is representing all the angles in the range of [0º, 180°] using combinations of two elementary angles 45^0 and 7.125º.

In CORDIC computation, the basic idea is to decompose the desired angle into sum of predefined elementary angles such that the rotation can be accomplished with simple shift-and-add operations. For twiddle factor, predefined elementary angles are considered as 45˚ and 7.125˚[8].Then the number of shifts which are stored in ROM are the rotation values corresponding to angles 22.5º, 45º, 67.5º, 90º, 112.5º,135º, 157.5º, 180º. The set of predefined angles for target angle is represented by m0,m1,m2,m3,m4,m5 if six micro rotations are used where m0,m1,m2,m3,m4,m5 are integers which depend on the value of target angle [1]. Instead of storing rotation angles we only need to store values of number of shifts. Then the corresponding rotations provide architecture for 16 point FFT butterfly with merely shifters and adder.

The following table shows micro rotation values corresponding to the above given angles for FFT.

Table.1.Optimized rotation with six micro rotations

angle	m0, c0	m1 ,c1	m2 ,c2	m3,c3	m4,c4	m5 ,c5
22.5	3,1	3,1	3,1	---	---	---
45	0,1	---	---	---	---	---
67.5	0,1	3,1	3,1	3,1	---	---
90	0,1	0,1	---	---	---	---
112.5	0,1	0,1	3,1	3,1	3,1	---
135	0,1	0,1	0,1	---	---	---
157.5	0,1	0,1	0,1	3,1	3,1	3,1

The architecture for modified CORDIC [1] is as shown below in the figure 3.

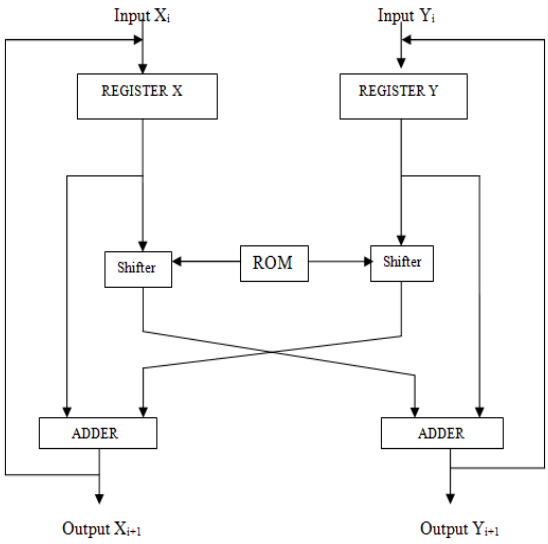

Fig.3.Optimized rotation with micro rotation

Scaling

Other simplifications performed by the Volder's algorithm [2] is the removal of scaling from the iterative micro rotations leads to a pseudo-rotated vector instead of the desired rotated vector .Since the scale-factor does not depend on the direction of micro rotations, the final scale-factor converges to some value depending on number of micro rotation. Therefore, instead of scaling during each micro rotation, the magnitude of final output could be scaled by K.

$$K=\prod_{i=0}^{n} 1/(1+2^{-2i})^{1/2} \quad \text{--- (15)}$$

Where 'n' is the number of micro rotation.

V. SCALING FREE CORDIC

The details of the scaling free CORDIC algorithm are provided in the reference [9, 10]. The working equation of the scaling free CORDIC is given as

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \prod_{i=p}^{b-1} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \begin{bmatrix} 1-2^{-(2i+1)} & 2^{-i} \\ -2^{-i} & 1-2^{-(2i+1)} \end{bmatrix} \quad \text{--(16)}$$

Each of the elementary rotational stages of the scaling free CORDIC costs two adders and two shifters more compared to that of the conventional CORDIC. But it is performed in an alternate cycle to increase the speed. Furthermore, since this formulation completely eliminates the requirement of scale factor compensation circuit, the overall hardware complexity of the scaling free CORDIC is less than the conventional one.

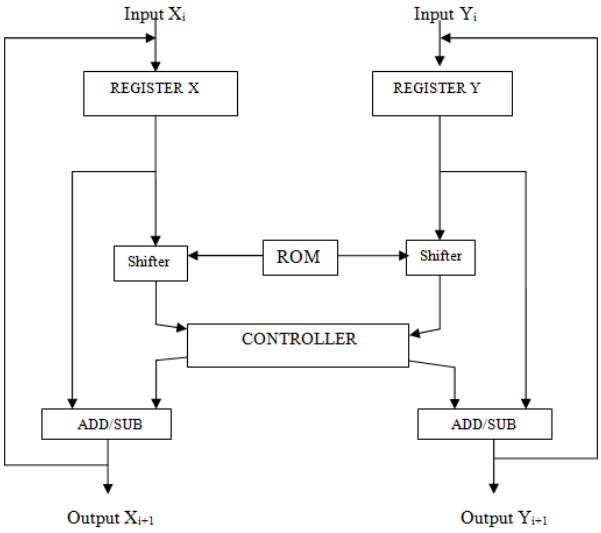

Fig.4.Scaling free CORDIC

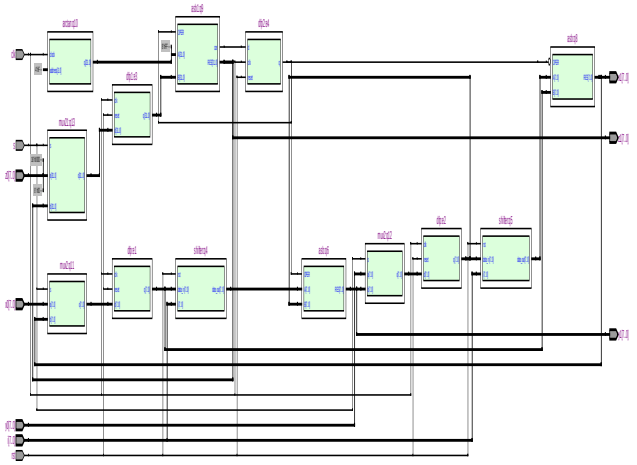

Fig.6.RTL View for conventional CORDIC

VI. SYNTHESIS AND SIMULATION RESULTS

Figure 5, 6 and 7 shows the RTL view for the proposed architecture, conventional CORDIC and scaling free CORDIC respectively. Figure 8 shows the simulation result of proposed architecture followed by scaling free CORDIC simulation result.

Table.2.Result Analysis

Design	Area	Speed
Conventional CORDIC	192 (LEs)	170.24MHz
Scaling free CORDIC	99 (LEs)	174.03MHz
Modified CORDIC (Proposed)	82 (LEs)	185.94MHz

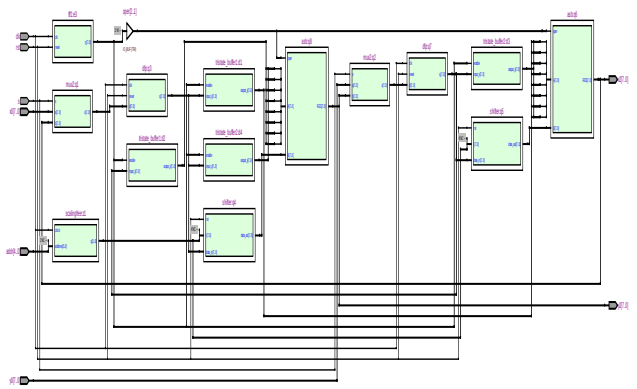

Fig.7.RTL View for scaling free CORDIC

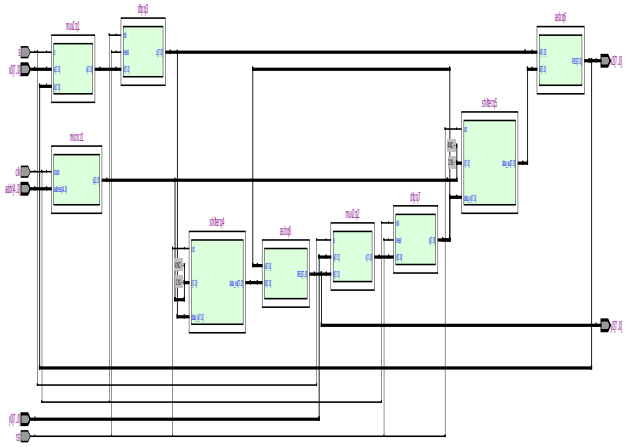

Fig.5.RTL View for proposed architecture

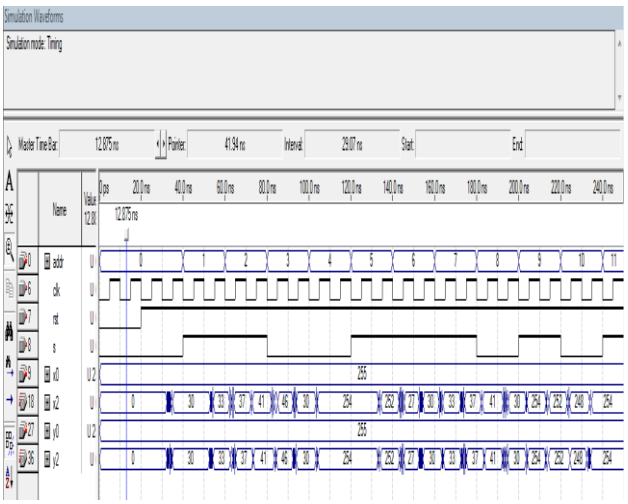

Fig.8 Simulation waveform for proposed architecture

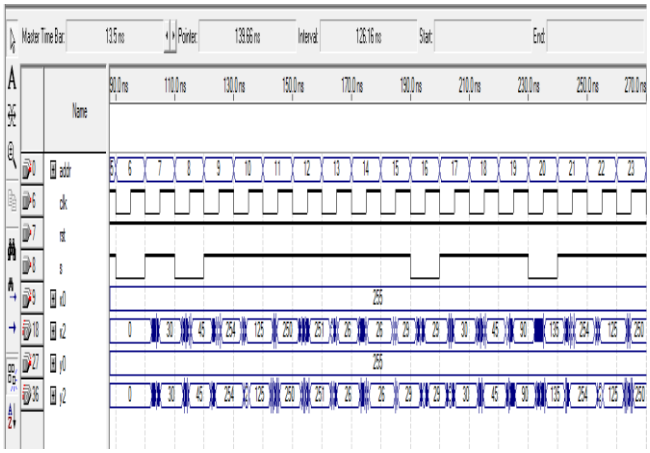

Fig.9 Simulation waveform for scaling free CORDIC

VII. CONCLUSION

In this paper, a modified CORDIC algorithm based design of twiddle factor angle generation for Fast Fourier Transform in DSP is presented. From the table 2, it is found that the Modified CORDIC algorithm based design shows better results with 2.34% lesser area and 1.09 times faster in FFT calculation than conventional one. Although it is used only for fixed angle, it is suitable for scaling free CORDIC by approximating sin ϕi=2-i and cos ϕi=1-2-i. The proposed design operates at a maximum frequency of 185.94 MHz for micro rotation along with efficient speed and area utilization than the conventional CORDIC which requires separate scaling architecture to provide cost effective solution. With reduced ROM size, dynamic power dissipation is reduced without delay penalties.

REFERENCES

[1] Pramod Kumar Meher, SeniorMember, IEEE, and Sang Yoon Park, Member,"CORDIC Designs for fixed angle of rotation" IEEE transactions on very large scale integration (VLSI) systems, vol. 21, no. 2, February 2013.

[2] J.E.Volder," The CORDIC trigonometric computing technique," IRE Transactions on Electronic Computers, vol. EC-8, pp. 330–334, Sep.1959.

[3] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, 50 Years of CORDIC: Algorithms, Architectures, and Applications in IEEE transactions on circuits and systems—I: regular papers, vol. 56, no. 9, September 2009.

[4] J. S. Walther, "A unified algorithm for elementary functions," in Proceedings 38th Spring Joint Computer Conference, Atlantic City, New Jersey, 1971, pp. 379–385.

[5] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing, "IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35,Jul. 1992.

[6] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture," IEEE Trans. Circuits Syst. for Video Technol., vol. 15, no.11, pp. 1463–1474, Nov. 2005.

[7] S.Karthick, P.Priya, S.Valarmathy, "CORDIC based fft for signal processing System ", International journal of advanced research in electrical, electronics and instrumentation engineering vol. 1, issue 6, December 2012.

[8] A.A.Mushina, T.Y.Siby, B.D.Parameshachari, V.R.Athira," Reduced hardware cordic based 16 point fft processor ", International journal of advanced research in computer science, volume 4, no. 10, September-October 2013.

[9] E. Grass, B. Sarker and K. Maharatna, "A Dual Mode Synchronous/Asynchronous CORDIC Processor", Proc.8th IEEE International Symposium on Asynchronous Circuits and Systems, pp. 76 – 83, Manchester, U. K., April 2002.

[10] K Maharatna, A. S. Dhars and Swapna Banerjee, "A VLSI Array Architecture for Realization of DFT, DHT, DCT and DST", J. Signal Processing, vol. 81, pp. 1813 – 1822, 2001.