Journal of Advances in Science and Technology
Vol. 12, Issue No. 25, (Special Issue) December-2016, ISSN 2230-9659

Implementation 64- Point FFT Using CORDIC Algorithm

Yogini D. Borole[1]*, Dr. C. G. Dethe[2]

[1]Department of E&TC, National Institute of Electronics & Information Technology, Aurangabad

[2]Director, UGC -Academic Staff College, Nagpur

Abstract – This electronic document describes the method of implementation of Fast Fourier transform processor: FFT using mixed CORDIC and radix algorithm. The basic result analyzed using this method is power consumption and area. CORDIC is basically DSP application but here it is used for twiddle factor multiplication. Radix 2^5 fft algorithm is used for reduction of complex multiplication. The proposed calculation minimizes the quantity of multipliers. Outline illustrations demonstrate that the FFT planned by the proposed system displays a lower equipment with more quality than existing techniques.

At first FFT usefulness is checked utilizing MATLAB lastly re-enacted and integrated utilizing Xilinx ISE 14.1.Besides CORDIC calculation is executed in both MATLAB and in Xilinx on Virtex-5. The fundamental goal of this work is to get a territory proficient FFT and CORDIC without execution misfortune that could be utilized as a piece of Signal preparing.

Keywords—FFT, Radix 2^5, MDF, CORDIC algorithm

----------♦----------

1. INTRODUCTION

1.1 Overview

A Fast Fourier transform (FFT) is a fast computational algorithm to compute the discrete Fourier transform (DFT) and its inverse. The Fast Fourier Transform does not refer to a new or different type of Fourier transform. It refers to a very efficient algorithm for computing the DFT. The time takes to perform a DFT on a computer depends primarily on the number of multiplications involved O (N2)) while FFT only needs Nlog2(N).The central insight which leads to this algorithm is the realization that a discrete Fourier transform of a sequence of N points can be written in terms of two discrete Fourier transforms of length N/2. Therefore, if N is a power of **2**, it is possible to recursively apply this decomposition.

The main research idea of this work is optimize the design of FFT processor. Area and power reduction is the main objective of this work. So CORDIC algorithm introduced with radix 2^5 FFT algorithm. The Coordinate Rotation Digital Computer (CORDIC) algorithmic, proposed by Jack Voider[1] can be utilized for an extensive variety of capacities including certain trigonometric, hyperbolic, straight and logarithmic capacities. The Coordinate Rotation Digital Computer (CORDIC) algorithmic, proposed by Jack Voider can be utilized for an extensive variety of capacities including certain trigonometric, hyperbolic, straight and logarithmic capacities. CORDIC unit utilizes just moves and add to figure these capacities. The CORDIC algorithm provides an iterative method of performing vector rotation mode, CORDIC is used for converting one vector in rectangular form to another vector in rectangular form. In the vector mode, it converts a vector in rectangular form to polar form. Hence CORDIC is used as an efficient way to realize multiplication-free FFT [2].

The remainder of this project is organized as follows:

Section II and III review the CORDIC algorithm and the general hardware architectures required to implement FFT.

Section IV introduces the proposed radix and CORDIC algorithm for 64 point FFT implementation.

Section V and VI presents the performance evaluation and comparison and conclusion respectively.

II. REVIEW OF CORDIC ALGORITHM

The key concept of CORDIC arithmetic is based on the simple and ancient principles of two-dimensional geometry. But the iterative formulation of a computational algorithm for its implementation was first described in 1959 by Jack E. Volder for the computation of trigonometric functions, multiplication and division [6][7][9]. CORDIC based computing

received increased attentionin1971, when John Walther showed that, by varying a few simple parameters, it could be used as a single algorithm for unified implementation of a wide range of elementary transcendental functions involving logarithms, exponentials, and square roots along with those suggested by Volder [7].

CORDIC is attractive due to the simplicity of its hardware implementation, since the same iterative algorithm could be used for all the above mathematical applications using the basic shift-add operations of the form x ± y2-i.

The conventional method of implementation of rotation transform is represented by the equations.

$$Xout = xin \cos \theta - yin \sin \theta. \quad (1)$$
$$Yout = xin \sin \theta + yin \cos \theta. \quad (2)$$

where (xin ,yin) and (xout ,yout) are the initial and final coordinates of the vector respectively.

The hardware realization of these equations requires four multiplications, two additions/subtractions and accessing the table stored in the memory for trigonometric coefficients. The CORDIC rotator performs 2D rotation using a series of specific incremental rotation angles selected so that each is performed by a shift and add operation iteratively.

The three basic equations of CORDIC algorithm are:

$$Xi+1 = xi - m\, si\, yi\, \rho - Sm,i \quad (3)$$
$$Yi+1 = yi + si\, xi\, \rho - Sm,i \quad (4)$$
$$Zi+1 = zi - si\, am,i \quad (5)$$

Based on the value of **m** the algorithm can operate in one of three configurations:

Linear (m = 0), Circular (m = 1) and Hyperbolic (m = -1). Within each of these configurations the algorithm functions in one of two modes – rotation or vectoring.**σi** represents either clockwise or counter clockwise direction of rotation, **ρ** represents the radix of the number system and the shift sequence Sm, i depends on the coordinate system and the radix of number system. Sm, i affects the convergence of the algorithm. In rotation mode, the input vector is rotated by a specified angle, while in vectoring mode the algorithm rotates the input vector to the x-axis while recording the angle of rotation is required. The value of ai also changes according to the configuration. Depending on the mode of operation z and y are the steering variables in rotation and vectoring mode respectively. The length of the vector increases if required micro rotations are not perfect, so in order to maintain a constant vector length, the obtained results have to be scaled by the scale factor K and it is given by the equation.

$$K = \prod_i K_i \quad (6)$$

III. FFT ALGORITHM

The fast Fourier transform (FFT) is a discrete Fourier transform algorithm which reduces the number of computations needed for points from to, where lg is the base-2 logarithm.

FFTs were first discussed by Cooley and Tukey (1965), although Gauss had actually described the critical factorization step as early as 1805 (Bergland 1969, Strang 1993). [2, 3, 4]. A discrete Fourier transform can be computed using an FFT by means of the Danielson-Lanczos lemma if the number of points is a power of two. If the number of points is not a power of two, a transform can be performed on sets of points corresponding to the prime factors of which is slightly degraded in speed. An efficient real Fourier transform algorithm or a fast Hartley transform (Brace well 1999) gives a further increase in speed by approximately a factor of two. Base-4 and base-8 fast Fourier transforms use optimized code, and can be 20-30% faster than base-2 fast Fourier transforms. prime factorization is slow when the factors are large, but discrete Fourier transforms can be made fast for N=2, 3, 4, 5, 7, 8, 11, 13, and 16 using the Winograd transform algorithm [5][7].

Fast Fourier transform algorithms generally fall into two classes:

a) Decimation in time

b) Decimation in frequency.

The Cooley-Tukey FFT algorithm first rearranges the input elements in bit-reversed order, and then builds the output transform (decimation in time). [10]

The N-point discrete Fourier transform is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0,1,\ldots,N-1) \quad (7)$$

The N-point FFT can be decomposed to repeated micro operations called butterfly operations. When the size of the butterfly is r, the FFT operation is called a radix-r FFT. For FFT hardware realization, if only one butterfly structure is implemented in the chip, this butterfly unit will execute all the calculations recursively. If parallel and pipeline processing techniques are used, an N point radix-r FFT can be executed by N/(r log N) clock cycles. This indicates that a radix-4 FFT can be four times faster than a radix-2 FFT.

Fig. 1 shows the general structure of the 64point radix-4 FFT. For hardware realization of FFT, multi-bank memory and "in place" addressing strategy are often used to speed-up the memory access time and minimize the hardware consumption. For radix-r FFT, r banks of memory are needed to store data, and each memory bank could be two-port memory. With "in-place" strategy, the r outputs of the butterfly can be written back to the same memory locations of the r inputs, and replace the old data. In this case, to realize parallel and pipelined FFT processing, an efficient addressing scheme is needed to avoid the data conflict. A popular addressing scheme for radix-r (r>2) was presented by Johnson, however due to the modulo-r addition, this method is slow and the speed depends on the length of FFT.

$$X(k)=\sum_{n=0}^{N-1}x(n)\cdot e^{-j\left(\frac{2\pi}{N}\right)nk}\ (k=0,1,\ldots,N-1) \qquad (7)$$

Figure 1 RADIX-4 64 points FFT architecture

IV. PROPOSED DESIGN

In this paper, low power techniques are employed for power consumption using reconfigurable complex multiplier. Using

Radix-2^5 algorithm, increase the computational speed, further reduce the chip area by three different processing elements (PE's) were proposed in this radix-2^5 64-point FFT/IFFT processor. The proposed architecture uses CORDIC algorithm to implement butterfly units to generate twiddle factor angle values and to reduce the truncation error. During each iteration of the CORDIC algorithm, the input angle is compared with the constant.

The proposed architecture consists of a butterfly units, complex Booth multipliers, complex constant multipliers, first-in first-out (FIFO), control unit and multidata scaling blocks. The butterfly units perform complex addition and subtraction of two input data x[*n*] and x[*n* + *N*/2]. Each input signals in the butterfly units come from previous stage and the FIFO, respectively. The butterfly unit 1 (BU1) conducts to only complex addition and subtraction. But, the butterfly unit 2 (BU2) includes twiddle factor *W*4 multiplication utilizing any multiplexers.

The twiddle factor multiplication for FFT computation is conducted by fixed-width complex multipliers.

The complex multiplication needs a look-up table (LUT) using read-only memory (ROM) to store the twiddle factor values. To reduce the critical path delay, the pipelined complex Booth multiplier was used in this FFT processor. Furthermore, the error compensation technique for the fixed-width multiplication was applied to reduce quantization error. The twiddle factor multiplication is conducted using fixed width complex multipliers. The twiddle factor values stored in the read-only memory (ROM) are used as the multiplicand in the complex Booth multiplier. The modified Booth algorithm is used widely for high speed multiplications. Since the maximum clock rate of the FFT processor depends on the critical path of the complex Booth multiplier, three level pipelined complex Booth multiplier is used for high-speed operation. Because quantization errors affect the signal-to-noise ratio (SNR) performance of the system, an error compensation method [9] is used to reduce the quantization error. The proposed FFT processor uses constant multipliers based on the canonical signed digit (CSD) representation for the complex multiplication arithmetic in stages 2, 3, and 7.

The twiddle factor W8 has only one coefficient, but twiddle factors W16 and W32 have three and seven coefficients, respectively. Mostly the existing research is using complex Booth multipliers for the twiddle factor W32 multiplication. However, in our design, the complex CSD constant multiplier has been used for the twiddle factor W32 multiplication.

Also, the common sub-expressions sharing (CSS) technique reduces the hardware complexity of the complex CSD constant multipliers [10]. The proposed FFT processor applied CSD constant multiplier instead of complex Booth multiplier at several stages.

The radix 2^5 algorithm has same butterfly structure as radix 2, only the change is twiddle factor is available or each stage. The radix 2^5 algorithm can be expressed as follows.

$$n = (N/2\ n1+N/4\ n2+N/8\ n3+N/16\ n4+N/32\ n5+n6)\ N$$

$$n1, n2, n3, n4, n5 = 0, 1 \quad n6=0, \ldots, N/32-1$$

$$k = (k1+2k2+4k3+8k4+16k5+32k6)\ N$$

$$X\ (k1+2k2+4k3+8k4+16k5+32k6)$$

$$=\sum_{n_6=0}^{\frac{N}{32}-1}\sum_{n_5=0}^{1}\sum_{n_4=0}^{1}\sum_{n_3=0}^{1}\sum_{n_2=0}^{1}\sum_{n_1=0}^{1}\times x\left(\frac{N}{2}n_1+\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+\frac{N}{32}n_5+n_6\right)W_N^{n_6}$$

The twiddal factor can be expressed as follows:

$$W_N^{\left(\frac{N}{2}n_1+\frac{N}{4}n_2+\frac{N}{8}n_3+\frac{N}{16}n_4+\frac{N}{32}n_5+n_6\right)(k_1+2k_2+4k_3+8k_4+16k_5+32k_6)}$$

$$=\underbrace{(-1)^{n_1k_1}}_{Stage1BU}\overbrace{(-j)^{n_2k_1}}^{Stage1TF}\underbrace{(-1)^{n_2k_2}}_{Stage2BU}\overbrace{W_8^{n_3(k_1+2k_2)}}^{Stage2TF}$$

$$\times\underbrace{(-1)^{n_3k_3}}_{Stage3BU}\overbrace{W_{16}^{(2n_4+n_5)(k_1+2k_2+4k_3)}}^{Stage3TF}\underbrace{(-1)^{n_4k_4}}_{Stage4BU}\overbrace{(-j)^{n_5k_4}}^{Stage4TF}$$

$$\times\underbrace{(-1)^{n_5k_5}}_{Stage5BU}\overbrace{W_N^{n_6(k_1+2k_2+4k_3+8k_4+16k_5)}}^{Stage5TF}W_{32}^{n_6k_6} \quad (8)$$

twiddle factor generation implemented using CORDIC algorithm as shown in figure 2.

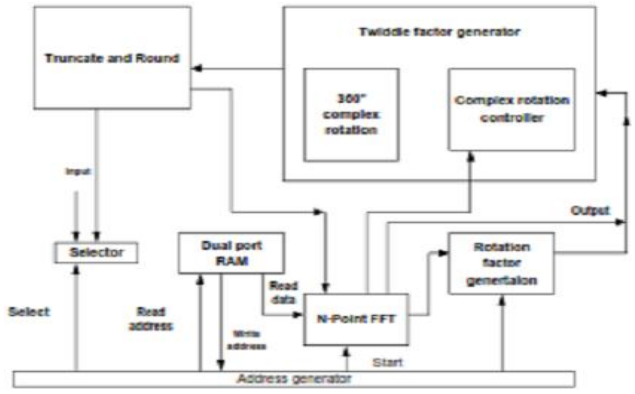

Figure 2.CORDIC algorithm structure for butterfly unit

V. VHDL AND MATLAB SIMULATION RESULTS:

64 point FFT Real and imaginary outputs are shown in figures which are compare using matlab and FPGA.

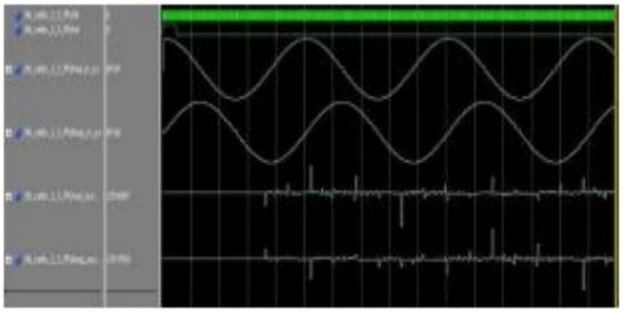

Figure 3.Real and imaginary outputs for 64 point FFT

VI. CONCLUSIONS

In this paper Radix-2^5 FFT processor architecture is studied. For generation of twiddle factor CORDIC algorithm is used. Various parts of FFT architecture such as Butterfly unit, Control Unit, Delay-Feedback model are discussed. In the next phase of this paper actual Implementation of FFT processor on FPGA will be done using VHDL.

Proposed research work emphases on the use of techniques to reduce the computational complexity of processor design and the algorithm used which result into improvement of the design significantly.

ACKNOWLEDGEMENT

The author would like to thank National Institute of Electronics & Information Technology, Aurangabad Research lab for supporting this work. Also very thankful to Dr. C. G. Dethe for their valuable guidance.

REFERENCES

Taesang Cho and Hannho lee, —A High- Speed Low Complexity Modified Radix- 2^5 FFT Processor for High Rate WPAN Application‖ IEEE Trans. VLSI SYSTEMS, vol.21, no.1, JANUARY 2013.

J. Lee and H. Lee, A high-speed two parallel FFT/IFFT processor for OFDM systems, IEICE Trans. Fundam., vol. E91-A, no. 4, pp. 1206– 1211, Apr. 2008.

Y. Lin, H. Liu, and C. Lee, —A 1 -GS/s FFT/IFFT processor for UWB applications,‖ IEEE J. Solid-State Circuits, vol. 40, no. 8, pp.1726–1735, Aug. 2005.

Y. Chen, Y. Tsao, Y. Wei, C. Lin, and C. Lee, —An indexed-scaling pipelined FFT processor for OFDM-based WPAN applications,‖ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 2, pp. 146–150, Feb. 2008.

M. Shin and H. Lee, —A high-speed four-parallel FFT processor for UWB applications, in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), 2008, pp. 960–963.

S. Tang, J. Tsai, and T. Chang, —A 2.4- GS/s FFT processor for OFDM based WPAN applications, IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 6, pp. 451–455, Jun. 2010.

Huang and S. Chen, —A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs),‖ in Proc. Int. Conf. Green Circuits Syst. (ICGCS), 2010, pp. 9–13.

T. Cho, H. Lee, J. Park, and C. Park, —A high-speed low-complexity modified radix-2^5 FFT processor for gigabit WPAN applications, in

Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), 2011, pp. 1259– 1262.

A. Cortes, I. Velez, and J. F. Sevillano, Radix FFTs: Matrical representation and SDC/SDF pipeline implementation,‖ IEEE Trans. Signal Process., vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

K. Cho, K. Lee, J. Chung, and K. Parhi, —Design of low-error fixed width modified Booth multiplier, IEEE Trans.Very Large Scale Integer. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

Corresponding Author

Yogini D. Borole*

Department of E&TC, National Institute of Electronics & Information Technology, Aurangabad

E-Mail – yoginiborole@gmail.com