Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

AREA EFFIECIENT ALGORITHM FOR THE IMPLEMENTATION OF CONFIGURABLE FFT/IFFT IN FPGA

Arsha P.S.
Department of ECE, Mohandas College of Engineering and Technology

Abstract — The core processing block of an OFDM system are Fast Fourier Transform (FFT) block and the Inverse Fast Fourier Transform (IFFT). The FFT/IFFT functions contain twiddle factor term. These complex functions are combination of sine and cosine terms that generally spread in the channel. The commonly used software solutions for the digital implementation of these functions are table lookup method and polynomial expansions, requiring number of multiplications and additions/subtractions. The size of ROM in the multiplier based implementation for the twiddle factors becomes the matter of concern with larger chip area. CORDIC (Coordinate Rotation Digital Computer) is a method for computing elementary functions using minimal hardware such as shifts adds/subs and compares. CORDIC works by rotating the coordinate system through constant angles until the angle is reduces to zero. The angle offsets are selected such that the operations on X and Y are only shifts and adds. The required Rotation for CORDIC is provided by Rotation factor Module. The Radix 4 FFT is chosen since it has fewer calculation than Radix 2. It is simulated and synthesized using Xilinx ISE design..The performance of the coding is analysed from the result of timing simulation using Xilinx ISE Design Suite 14.5 and Modelsim SE 6.5b.

Index Terms – *Orthogonal Frequency division Multiplexing, CORDIC, FPGA*

I. INTRODUCTION

The Fast Fourier Transform (FFT) Algorithm plays an important role in operation of digital signal processor. In the recent years, FFT and IFFT [3] have been frequently applied in the modern communication systems. The FFT algorithms are based on the fundamental principle of decomposing the computation of the DFTs [3], all with comparable improvements in computational speed. The Fast Fourier Transform [FFT] and Inverse Fast Fourier Transform [IFFT] are the two important key points in the OFDM system.FFT is a fast way to calculate the Discrete Fourier Transform [DFT], which transforms data from time domain to frequency domain where as IFFT [4] transforms data from frequency domain to time domain. FFT analyses an input signal sequence by using decimation-in-frequency (DIF) or decimation-in-time (DIT) decomposition to construct an efficiently computational signal-flow graph.

Multicarrier modulation is a technique used for the data transmission which divides the high bit rate data streams into several parallel low bit data streams, and these low bit data streams are used to modulate many carriers. Orthogonal frequency division multiplexing [OFDM] is one of the multicarrier modulation technologies which are widely used in emerging wired and wireless communication system. OFDM [6] is implementing in many emerging communication protocols due to several advantages over the traditional multiplexing technique frequency division multiplexing [FDM]. OFDM [11] transforms a frequency selective wideband channel into a group of non selective narrowband channels, by preserving orthogonality in the frequency domain, OFDM makes robust against the large delay spreads.

The hardware implementation of FFT/IFFT approaches is a challenging issue where the digital signal processors (DSPs) and field programmable gate array (FPGA) chips are two considering designing

environments for implementing different schemes of FFT approaches. Recently, the FPGA technology is used due to fast progress in very large scale integration (VLSI) technology. The FPGA devices provide complete programmable system on chip environments by incorporating the programmability of programmable logic devices and the architecture of gate arrays.

II.EXISTING SYSTEM

There are various algorithms to implement FFT, such as radix-2, radix-4 and split radix with arbitrary sizes. Radix-2 algorithm is the simplest one. The existing system was based on radix-2 algorithm. The so called radix-2[5] is due to its base is equals to 2 and the representation is 2^M where M represents the index/stage and its value is a positive integer. The computation of radix-2 made up of butterflies called Radix-2 butterflies. Depending on the type of decimation in the different domains, it is two types; they are Decimation in Time FFT (DIT-FFT) and Decimation in Frequency FFT (DIF-FFT). For radix-2 there are two inputs and two outputs and the inputs are arranged in bit reversal order, because of saving the memory locations and outputs are in a normal order for DIT-FFT and vice versa for DIF-FFT [7]. The given whole number decides the number of stages as log2N, and each stage consists of blocks it can be given as $N/2^{stage}$ and each block contains butterflies it can be given as $2^{stage-1}$, each stage includes the twiddle factors (W). Each and every stage having complex computations or simply complex multiplications and complex additions. Generally radix-2 having $N/2\log_2 N$ complex multiplications and Nlog2 N complex additions.
Radix-2 algorithm mainly depends on these computations, as number of stages increases proportionally computations are also increases. For example: for 16-point FFT the complex multiplications and additions are 32 and 64 respectively and for 32 point FFT[12] these are 90 and 180 respectively and so on. If number of inputs is more it became impossible to calculate by using R2 algorithm. By this as number of inputs is more, the complexity in calculations also more. FFT [13] is commonly implemented with complex multiplier; a complex multiplier is equivalent to four real multipliers and two real adders, and a ROM to store the twiddle factors. The ROM in this type of implementation takes most of the chip area, consumes more power and degrades the speed because of ROM read operation ROM size increased with large-point FFTs. Hence poor performance of the FFT in terms of power, speed, and area can be seen.

III. PROPOSED SYSTEM

The proposed system is based on radix-4 algorithm. Radix-4 is another FFT algorithm which was surveyed to improve the speed of functioning by reducing the computation; this can be obtained by change the base to 4. For a same number if base increases the power/index will decreases. For radix-4 the number of stages are reduced to 50% since $N=4^3$ ($N=4^M$) i.e. only 3 stages. Radix-4 is having four inputs and four outputs and it follows in-place algorithm. The following will explain the functioning of radix-4 and how the computational complexity is reduced. The radix-4 FFT equation essentially combines two stages of a radix-2 FFT into one, so that half as many stages are required. Since the radix-4 FFT requires fewer stages and butterflies than the radix 2 FFT, the computations of FFT can be further improved In order to speed up the FFT computation we can increase the radix.
CORDIC architecture can be used instead of complex multiplier and ROM based architectures. CORDIC has gained momentum for decades because of its less hardware Complexity in real time applications such as communication systems, signal and image processing. Power consumption, speed and accuracy are issues in this architecture implementation. Coordinate Rotation Digital Computer (CORDIC), a special purpose computer to compute many non-linear and transcendental functions, was

proposed by Volder in 1959. The functions that can be computed using a CORDIC computer include trigonometric, logarithmic, exponential, hyperbolic, multiplication, division, square root, etc. The CORDIC algorithm does not use calculus based methods such as polynomial or rotational function approximation.

The CORDIC algorithm requires only shifts, adders and table lookups, simple integer math. So, it is possible to implement a specialized CORDIC [5] machine, enough to real time calculations, dedicated to that one purpose. It offers opportunity to calculate desired functions in simple and elegant way. Due to the simplicity of the involved operations the CORDIC algorithm is well suited for VLSI implementations. The CORDIC[15] computing technique is used to compute sine and cosine values of the given angle to eliminate the read-only memories (ROM's) used to store the twiddle factors, the proposed architecture applies a ROM-less FFT/IFFT processor, thus consuming lower power. This paper describes how to Design and Implement a configurable data width and a configurable number of sample points FFT and IFFT [16] with efficient CORDIC algorithm to perform the butterfly calculus. Also describes how implement an OFDM Transmitter on FPGA using VHDL for a 31 subcarrier (channels) OFDM system using 64 points radix-4 FFT Frequency decimation, and each channel modulation will use a 4-QAM constellation as an application of the implemented FFT using CORDIC.

IV. FUNCTIONING OF RADIX-4 ALGORITHM

The radix-4 FFT algorithm is more suitable for digital signal processor which has minimal complex computation than radix-2, radix-8 and structural architecture is also more suitable than other radix FFT algorithms. The paper presents efficient implementation of radix 4 Configurable FFT and IFFT with CORDIC algorithm in FPGA [17]. The radix-4 DIF-FFT recursively partitions a DFT into four quarter-length DFTs of groups of every fourth time sample. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix-4 FFTs require only 75% as many complex multiplications as the radix-2 FFTs.

The radix-4 decimation-in-time and decimation-in-frequency Fast Fourier transforms (FFTs) gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs. The radix-4 decimation-in-frequency algorithm rearranges the DFT equation into four parts: sums over all groups of every fourth discrete-time index n = [0, 4, 8 ... N - 4], n = [1, 5, 9 ... N - 3], n = [2, 6, 10 ... N - 2] and n = [3, 7, 11 ... N - 1], This works out only when the FFT length is a multiple of four. Just as in the radix-2 DIF FFT, further mathematical manipulation shows that the length-N DFT can be computed as the sum of the outputs of four length-N/4 DFTs, of the even-indexed and odd-indexed discrete-time samples, respectively, where three of them are multiplied by so-called twiddle factors $W_N^k = e^{\left(i\frac{2\pi k}{N}\right)}, W_N^{2k}, and\ W_N^{3k}$. The computational problem for the DFT is to compute the sequence X(K) of N complex -valued numbers given another sequence of data x(n) of length N, according to the formula

$$x(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \qquad 0 \le k \le N-1$$

Where$W_N = e^{\frac{-j2\pi}{N}}$. Sequence *x(n)* is also assumed to be complex valued. Similarly IDFT is given by

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)\, W_N^{-kn} \quad , 0 \le n \le N-1$$

The basic building blocks of all FFT processors are the "Butterfly" which depends on radix of FFT Processor. Radix-4 algorithm is more attractive since it requires less number of multiplication operations for FFT processor [6] which reduces the complexity of computation. Attention is given on improvement of twiddle factor generation in radix-4 FFT design by incorporating the CORDIC [8] for twiddle factor

computation which reduces the complexity of conventional radix-4 architecture which are based on power series method. It involves the decimation of the sequence into four sequences of length $N/4$. The General DFT formulae can be reduced into four smaller DFTs as,

$$X(K) = \sum_{n=0}^{\frac{N}{4}-1} x(n)\, W_N^{kn} + \sum_{n=\frac{N}{4}}^{\frac{N}{2}-1} x(n) W_n^{kn} + \sum_{n=\frac{N}{2}}^{\frac{3N}{4}-1} x(n)\, W_N^{kn} + \sum_{n=\frac{3N}{4}}^{N-1} x(n)\, W_N^{kn}$$

The equation shows that radix-4 Decimation in frequency algorithm is obtained by breaking the N-point DFT. The Basic butterfly diagram of Radix 4 FFT is shown in Fig.1

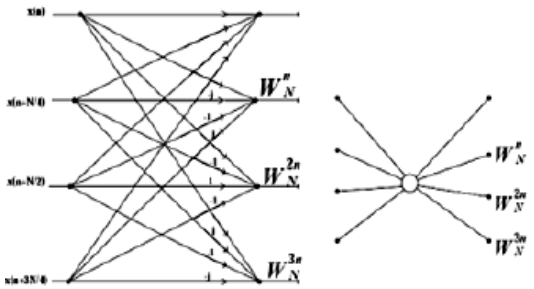

Fig.1: Basic Structure of Radix 4 FFT Butterfly

In all FFT processors, the basic building blocks are the "Butterfly" which depends on radix of FFT Processor. Each butterfly involves three complex multiplications and Complex additions. The same structure used repeatedly as the number of stages and N increases accordingly in FFT and IFFT.

V. CORDIC ARCHITECTURE

A.CORDIC algorithm

CORDIC or Coordinate Rotation Digital Computer is a simple and hardware efficient algorithm for the implementation of various elementary mathematical functions, especially trigonometric. It uses simple shift, add, subtract and table look-up operations to achieve this objective. It is usually implemented in either Rotation mode or Vectoring mode [11]. In either mode, the algorithm is rotation of an angle vector by a definite angle but in variable directions. This fixed rotation in variable direction is implemented through an iterative sequence of addition/subtraction followed by bit-shift operation. The final result is obtained by appropriately scaling the result obtained after successive iterations. Owing to its simplicity the CORDIC algorithm can be easily implemented on a VLSI system. 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 the given angle are computed.

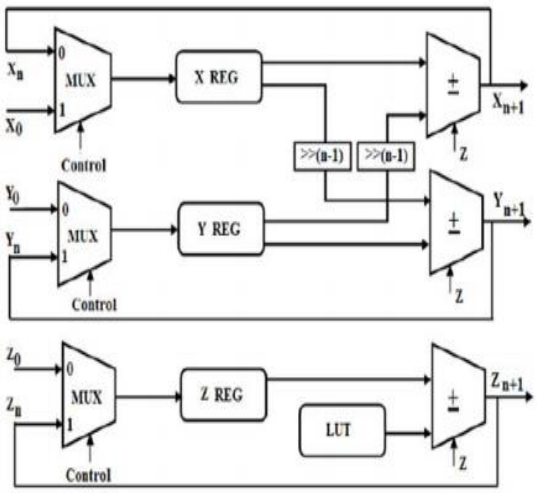

Fig.2: Basic architecture of CORDIC

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 our design is implemented using pipelining, taking care of latency issues. The CORDIC algorithm performs a planar rotation. Graphically, planar rotation means transforming a vector (x_i , y_i) into a new vector (x_j , y_j). A planar rotation for a vector of (X_i , Y_i) is defined as.

$$X_j = X_i Cos\theta - Y_i Sin\theta$$
$$Y_j = Y_i Cos\theta + X_i Sin\theta$$

The complete angle rotation can be executed in several steps, using an iterative process. As the number of iteration increases the accuracy can be improved. A single step is defined by the following equation

$$X_{n+1} = cos\theta_n \, (X_i - Y_i Tan\theta_n)$$
$$Y_{n+1} = Cos\theta_n \, (Y_i + X_i Tan\theta_n)$$

Additional multipliers in implementation of equations can be eliminated by selecting the angle steps such that the tangent of a step is a power of 2. Multiplying or dividing by a power of 2 can be implemented using a simple shift operation. The angle for each step is given by

$\theta_n = Tan^{-1} 2^{-n}$

This results in the following equation for $Tan\theta_n = S_n . 2^{-n}$ where$S_n = [-1 \quad 1]$.

Combining above equations

$$X_{n+1} = Cos\theta_n \, X_n - Y_{.i} S_n . 2^{-n}$$
$$Y_{n+1} = Cos\theta_n \, Y_n + X_i S_n . 2^{-n}$$

Besides for the $Cos(\theta n)$ coefficient, the algorithm has been reduced to a few simple shifts and additions. The coefficient can be eliminated by pre-computing the final result.

The first step is to rewrite the coefficient

$$Cos\ \theta_n = Cos\ (Tan^{-1}(2^{-n}\ )\ ).$$

The second step is to compute above equation for all values of 'n' and multiplying the results, which we will refer to as K.

$$K = \frac{1}{p} = \prod_{n=0}^{\infty} Cos(Tan^{-1}(2^{-n})) = 0.607253$$

K is constant for all initial vectors and for all values of the rotation angle, it is normally referred to as the congregate constant. The derivative P (approx. 1.64676) is defined here because it is also commonly used .We can now formulate the exact calculation the CORDIC performs

$$X_j = K . (X_i Cos\ (Z_i) - YSin_i(Zi))$$

$$Y_j = K.(Y_i cos(Z_i) + X_i Sin(Z_i))$$

Because the coefficient K is pre-computed and taken into account at a later stage, equation may be written as $X_{n+1} = X_n - Y_i S_n . 2^{-n}$

$$Y_{n+1} = Y_n + X_i S_n . 2^{-n}$$

Or as $Z_{n+1} = \theta - \sum_{i=0}^{n} \theta_i$

At this point a new variable called 'Z' is introduced. Z represents the part of the angle θ which has not been rotated yet. For every step of the rotation S_n is computed as a sign of Z_n be $S_n = -1$ if $Z_n \leq 0$ otherwise$S_n = 1$. Combining equations results in a system which reduces the not rotated part of angle θ to zero The Coordinate Rotation without Scaling factor is called pseudo rotation. The absence of scaling factor can affect the results during the rotations but at the end of total computation this factor can be multiplied to balance the effect.

B. Sine-Cosine Generation

CORDIC [4] can be used to compute sine and cosine of an angle with little variation. The angle is given as input. The x and y values are accumulated. After fixed number of iterations the final coordinates of the vector i.e. the x and y values give the values of Cos and Sin respectively Sine and Cosine can be calculated using the first CORDIC scheme which calculates:

$$[X_j , Y_j , Z_j] = [P(X_i CosZ_i - Y_i sinZ_i), P(Y_i cos(Z_i) + X_i SinZ_i , 0]$$

In rotational mode the sine and cosine of the input angle can be computed simultaneously.

$$xi = \frac{1}{P} = \frac{1}{1.6467} = 0.60725$$

$$yi = 0$$

$$zi = \theta$$

Setting the y component of the input vector to zero reduces the rotation mode result to the generation of sine and cosine of the angle θ.The sine and Cosine output can be used as the in phase and quadrature component in Butterfly implementation of FFT and IFFT blocks.

VI. RADIX 4 FFT/IFFT USING CORDIC

For N-point sequence, the radix-4 FFT algorithm consist of taking number of 4 data points at a time from memory, performing the butterfly computation and returning the result to memory. This procedure repeated many times, i.e., ((Nlog4N)/4) times in the computation of N-point data DFT. Therefore, memory requirement is essential factor for FFT processor design. The requirement of memory size is 2N for the input sequence which is complex number and 2N for the output sequence. So the capacity of memory for N-point FFT processor is 4N.

The DFT and IDFT involve basically the same type of computations. Hence the efficient computational algorithms for the DFT [16] can be applied efficiently to IDFT also. The FFT is an efficient class of computational algorithms of the DFT. The input and output are complex data, hence in the FFT algorithm is said to be as complex FFT and complex IFFT.

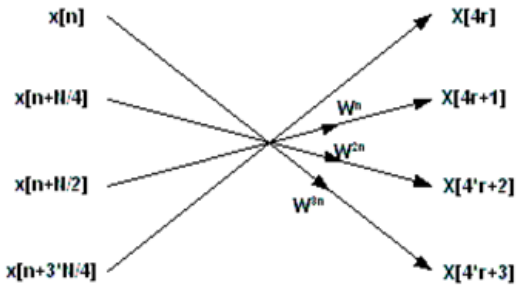

Fig.3: Single Radix 4 Butterfly Diagram

The input real and imaginary data for the arms of the butterfly be

$$x(n) = x_a + j * y_a$$
$$x(n + N/4) = x_b + j * y_b$$
$$x(n + N/2) = x_c + j * y_c$$
$$x(n + 3N\ 4) = x_d + j * y_d$$

where N is length of FFT. The output real and imaginary data for the arms of the butterfly be

$$X(4r) = x_a' + j * y_a'$$
$$X(4r + 1) = x_b' + j * y_b'$$
$$X(4r + 2) = x_c' + j * y_c'$$
$$X(4r + 3) = x_d' + j * y_d'$$

Twiddle factors for radix-4 FFT:

$$W_n = co1 + j * (-si1)$$
$$W_{2n} = co2 + j * (-si2)$$
$$W_{3n} = co3 + j * (-si3)$$

Corresponding butterfly CFFT equations:

$$x_a' = x_a + x_b + x_c + x_d$$
$$y_a' = y_a + y_b + y_c + y_d$$
$$x_c' = (x_a + y_b - x_c - y_d)co1 + (y_a - x_b - y_c + x_d)(si1)$$
$$y_c' = (y_a - x_b - y_c + x_d)co1 - (x_a + y_b - x_c - y_d)(si1)$$
$$x_b' = (x_a - x_b + x_c - x_d)co2 + (y_a - y_b + y_c - y_d)(si2)$$
$$y_b' = (y_a - y_b + y_c - y_d)co2 - (x_a - x_b + x_c - x_d)(si2)$$
$$x_d' = (x_a - y_b - x_c + y_d)co3 + (y_a + x_b - y_c - x_d)(si3)$$
$$y_d' = (y_a + x_b - y_c - x_d)co3 - (x_a - y_b - x_c + y_d)(si3)$$

CIFFT uses same twiddle factor table as CFFT with modifications in the design equation as shown below

$$x_a' = x_a + x_b + x_c + x_d$$
$$y_a' = y_a + y_b + y_c + y_d$$
$$x_c' = (x_a - y_b - x_c + y_d)co1 - (y_a + x_b - y_c - x_d)(si1)$$
$$y_c' = (y_a + x_b - y_c - x_d)co1 + (x_a - y_b - x_c + y_d)(si1)$$
$$x_b' = (x_a - x_b + x_c - x_d)co2 - (y_a - y_b + y_c - y_d)(si2)$$

$$y'_b = (y_a - y_b + y_c - y_d)co2 + (x_a - x_b + x_c - x_d)(si2)$$
$$x'_d = (x_a + y_b - x_c - y_d)co3 - (y_a - x_b - y_c + x_d)(si3)$$
$$y'_d = (y_a - x_b - y_c = x_d)co3 + (x_a + y_b - x_c - y_d)si3$$

The Computational result of real and imaginary component are stored in separate Memory. The same Butterfly structure can be used repeatedly as number of stages and number of data point varies. The in phase and quadrature Component is provided by the CORDIC module.

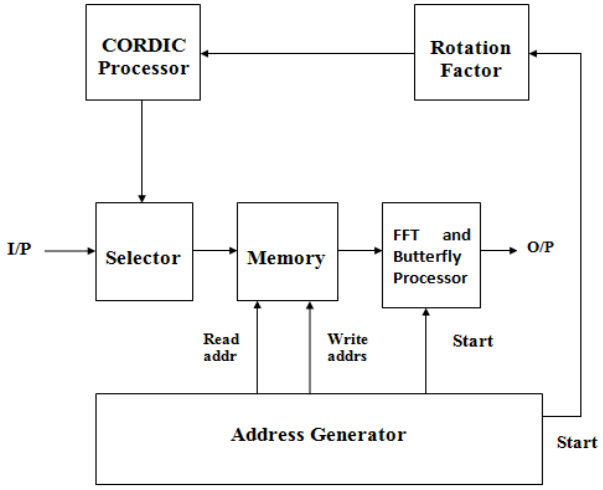

Fig.4: Architectural design

The fig.4 shows the Architectural design of the system. The algorithm of FFT consists of catching the data of the memory (2 for radix-2 and 4 for radix-4) [18], do the butterfly and to return the data made calculations for the memory. The Selector will select the input or data from CORDIC according to the process to be performed in further stages. However, the used element of main storage is the RAM memory, and the access is in a serial way, it is written and read in one address by time. As the amount of RAM memory in FPGA is limited, it is necessary to adapt the system to this situation. Dual Port RAM capable to read and write more than one input and output.

CORDIC generates the Inphase and Quadrature Component Corresponding to the output of rotation Factor module by performing simple shift and add operation. Rotation factor module generates output which is to be given as input to CORDIC inorder to generate the Inphase and quadrature Component corresponding to the stage of simulation of the Butterfly.The address Generator module determine, to which Address we have to write and read. It also provide the start signal to the Butterfly implementaion and Rotation Factor module part so that it can generate corresponding twiddle factor.

VIII. SIMULATION RESULTS

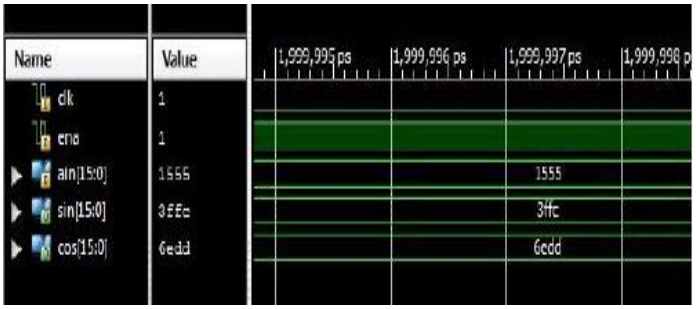

Fig.5: Sine and Cosine value of angle 30

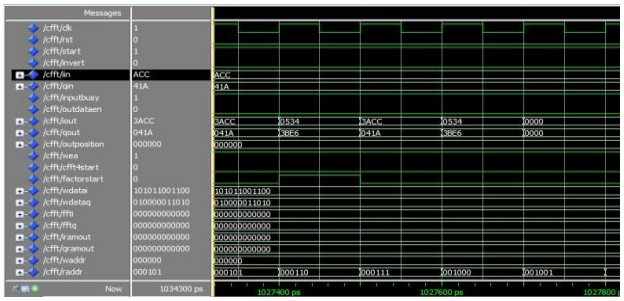

Fig.6: Output of FFT

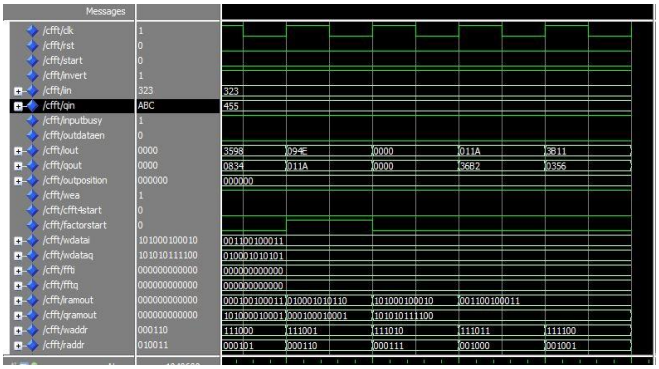

Fig.7: Output of IFFT

IX. CONCLUSION

FFT calculation has wider application in DSP. Existing method requires large ROM for storing many twiddle factor. Implementation of a CORDIC-based processor on FPGA gives us a powerful mechanism of implementing complex computations on a platform that provides a lot of resources and flexibility at a relatively lesser cost. Further, since the algorithm is simple and efficient the design and VLSI implementation of a CORDIC based processor is easily achievable. Sine and Cosine value is generated by using CORDIC algorithm. CORDIC processor are employed for twiddle factor multiplications for improving the accuracy, reducing the complexity and efficient utilization of available memory. The twiddle Factors generated is used for the implementation of FFT and IFFT for implementation of OFDM Transmitter

X.ACKNOWLEDGMENT

The author thankfully acknowledges Mrs.Devika R.G., Assistant Professor, M-CET, without whose guidance and supervision, this work would not have been possible. I would like to express my sincere gratitude to Director Dr. Ashalatha Thampuran in providing me with necessary requirements to help us to finish the work in time. I would also like to extend my whole hearted gratitude to the Head of the Department of Electronics and Communication Dr. R.Ibrahimkutty and the PG coordinator Prof. Ajith Chandran M.C. who were always ready to help me with ideas and suggestions for rectifying the mistakes that crept up time to time during the completion of this venture. I would also like to thank my friends and last but not the least the staff of ECE department for their whole hearted support and encouragement. Above all, I am thankful to the GOD ALMIGHTY!!

REFERENCES

[1] Amaresh Kumar, U.N. Tripathi, Roopak Kumar Verma, Manish Mishra,"64 Point Radix-4 FFT Butterfly Realization using FPGA"International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 4, October 2014
[2] Pramod Kumar Meher, Sang Yoon Park.' CORDIC Designs for Fixed Angle of Rotation'; IEEE Transactions On Very Large Scale Integration (VLSI) Systems, VOL. 21, NO. 2, February 2013
[3] Srinivasa Murthy H N, Roopa M, 'FPGA Implementation of Sine and Cosine Generators', International Journal of Innovative Technology and Exploring Engineering(IJITEE) ISSN, Volume-1, Issue-6, November 2012
[4] Rohit Shukla and Kailash Chandra Ray,' Low Latency Hybrid CORDIC Algorithm', IEEE Transactions On Computers, May 2012
[5] M. Mohamed Ismail, Dr. M.J.S Rangachar and Dr.Ch. D.V. Paradesi Rao "VLSI Implementation of OFDM using Efficient Mixed-Radix 8-2 FFT Algorithm with bit reversal for the output sequences". International Journal of Electronics and Communication Engineering. Volume 5, Number 4 (2012)
[6] Chu Yu, Chen-Hen Sung, Chien-Hung Kuo, and Mao-Hsu Yen,and Sao-Jie Chen, "Design and Implementation of a Low-Power OFDM Receiver for Wireless Communications", IEEE Transactions on Consumer Electronics, Vol. 58, No. 3, August 2012.
[7] Arioua M., Belkouch S., Agdad M., Hassani M. M., "VHDL implementation of an optimized 8-point FFT/IFFT processor in pipeline architecture for OFDM systems," International Conference on Multimedia Computing and Systems (ICMCS), pp.1-5, April 2011
[8] Manohar Ayinala, Michael Brown, and Keshab K. Parhi, "Pipelined Parallel FFT Architectures via Folding Transformation," IEEE Trans. VLSI Syst, Jun. 2012
[9] John G. Proakis, Dimitris G. Manolokis "Digital signal processing" Pearson prentice Hall, Inc 2008
[10] N. Mahdavi, R. Teymourzadeh, IEEE Student Member, Masuri Bin Othman, "VLSI Implementation of High Speed and High Resolution FFT Algorithm Based on Radix 2 for DSP Application," The 5th Student Conference on Research and Development, pp. 1-4,11-12 Dec. 2007
[11] Young-jin Moon, Young-il Kim, "A Mixed-Radix 4-2 Butterfly with Simple Bit Revering for Ordering the Output Sequences", ISBN 89-551 9-1 29-4, Feb.2006.
[12] J. C. Kuo et al., "VLSI design of a variable-length FFT/IFFT processor for OFDMbased communication systems," EURASIP J. Appl. Signal Process., no. 13,Dec. 2003
[13] S. Y. Park and N. I. Cho, "Fixed-point error analysis of CORDIC processor based on the variance propagation formula," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 51, no. 3, pp. 573-584, Mar. 2004.
[14] G. Lijun and K. K. Parhi, "Hierarchical pipelining and folding of QRD-RLS adaptive filters and its application to digital beamforming," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 47, no. 12, pp. 1503-1519, Dec. 2001
[15] A. Y.Wu and C. S.Wu, "A unified view for vector rotational CORDIC algorithms and architectures based on angle quantization approach," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 49;Oct. 2002
[16] C. S. Wu and A. Y. Wu, "Modified vector rotational CORDIC (MVRCORDIC) algorithm and architecture," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 48, no. 6, pp. 548-561, Jun. 2001.
[17] M. Jun, K. K. Parhi, and E. F. Deprettere, "Annihilation-reordering lookahead pipelined CORDIC-based RLS adaptive filters and their application to adaptive beam forming," IEEE Trans. Signal Process., vol. 48, Aug. 2000.
[18] Shousheng. He and Mats Torkelson, "Design and Implementation of a 1024-point Pipeline FFT Processor", IEEE Custom Integrated Circuits Conference, May. 1998, pp. 131-134

