Design and Simulation of Pipelined 32 Point FFT using Radix 2 Algorithm

Miss. Rutuja R. Taksande[1] Prof. Mangesh N. Thakare[2] Prof. Girish D. Korde[3]
[1]M. Tech. Student [2]Associate Professor [3]Assistant Professor
[1,2,3]Department of Electronics & Telecommunication Engineering
[1,2,3]BDCE, Sewagram, Maharashtra, India

Abstract— The Fast Fourier Transform (FFT) is one of the important operations in field of digital signal processing. Fast Fourier Transform is an algorithm which computes the Discrete Fourier Transform of an input sequence. This paper gives design of the Fast Fourier Transform (FFT) based on Decimation-In-Time (DIT) domain and Radix-2 algorithm .The input of Fast Fourier transform has been given by using a test bench code and output has been displayed using the waveforms son the Xilinx Design Suite 14.5.i.This Paper also includes design, synthesis and simulation of Vedic 32-bit Floating Point Complex Multiplier and 32 point Pipelined FFT. The coding has been done in VHDL whereas design, synthesis and simulation has been done using XILINX ISE 14.5i tool. The delay obtained for Vedic 32-bit Floating Point Complex Multiplier and 32 point Pipelined FFT is 36.356 ns and 24.801ns respectively .The synthesis and simulation results show that the computation for calculating the 32-point Fast Fourier transform is efficient in terms of speed and power.

Key words: Fast Fourier Transform, Floating Point Complex Multiplier, XILINX ISE 14.5i, VHDL

I. INTRODUCTION

This paper presents the design of 32-point FFT architecture using pipeline techniques. The work of the proposed work is concentrated on the design, synthesis and simulation of FFT. This design computes 32-points FFT and all the numbers follow floating point format based on IEEE 754 single precision floating point format .In this project the coding is done in VHDL and the synthesis / simulation is done using Xilinx ISE 14.5i. Discrete Fourier Transform (DFT) plays a vital role in the design and implementation of the discrete-time signal-processing algorithms and systems It also convert the samples in time domain to frequency domain. The Fast Fourier Transform is simply a fast way to calculate the Discrete Fourier Transform. The wide usage of DFT's in Digital Signal Processing applications is the motivation to Implement FFT's. Fourier transform is used to reduce frequency analysis of discrete non periodic signals. The FFT is another method of achieving the same result, but with less overhead involved in the calculations.

Transforms generally convert function from one domain to another domain without loss of information. Fourier Transform converts a function from the time domain to the frequency domain.

The mathematical formula used for the Fourier transform is as follows;

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-iwt}\ dt$$

DFT is similar to samples of the Fourier transform at equal frequencies. Similarly, computation of the N-point DFT corresponds to the computation of N samples of the Fourier transform at N equally spaced frequencies. Considering input x[n] as complex, then N complex multiplications and (N-1) complex additions are required to compute each value of the DFT, if computed directly from the formula given as;

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

$$W_N = e^{-j\frac{2\pi}{N}}$$

For computation of N values, there is requirement of total N^2 complex multiplications and N (N-1) complex additions. Every complex multiplication requires four real multiplications and two real additions and each complex addition requires two real additions. Therefore, a total of $4N^2$real multiplications and N (4N-2) real additions are required. Afterward, these multiplications and additions there should be provision for storing N complex input sequences and also to store N output values. Solution to this is to use Decimation-in-Time FFT radix-2 algorithm. The number of complex multiplications and additions will be reduced to (N/2) $\log_2 N$ and $N\log_2 N$ to compute the DFT of a given complex x[n].

Hence, this proposed work uses Decimation in Time FFT radix-2 algorithm to compute the DFT of a various sequences.[2]

A. Radix-2 DIT FFT Algorithm

The radix-2 algorithms are one of the simplest FFT algorithms. The decimation-in-time(DIT) radix-2 FFT partitions a DFT into two half-length DFTs of the odd-indexed and even indexed time samples. The outputs of these FFTs are again used to compute more than one outputs, it will reduce the total computational cost and total delay of the device. The radix-2 decimation-in-time and decimation-in-frequency both the Fast Fourier transforms (FFTs) are the simplest FFT algorithms. Like all FFTs, these FFTs gain their speed by repeatedly using the results of smaller computations to compute more than one DFT frequency outputs [2].

The radix-2 decimation-in-time algorithm rearranges the discrete Fourier transform (DFT) equation into two parts :a sum over the numbered (even) discrete-time in dices *n*=[0,2,4,…,*N*_2] and a sum over the odd-numbered indices *n*=[1,3,5,…,*N*_1].

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

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) e^{-\left(i\frac{2\pi(2n)k}{N}\right)} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) e^{-\left(i\frac{2\pi(2n+1).k}{N}\right)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) e^{-\left(i\frac{2\pi nk}{\frac{N}{2}}\right)} + e^{-\left(i\frac{2\pi k}{N}\right)} \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) e^{-\left(i\frac{2\pi nk}{\frac{N}{2}}\right)}$$

$$DFT\frac{N}{2}[[x(0),x(2),\ldots,x(N-2)]] + W_{N\,DFT}^{K}\frac{x}{2}[(x(1),x(3)\ldots..x(N-1)]$$

This is known as decimation in time since the time samples are rearranged in alternating groups and a radix-2 algorithm. A basic butterfly operation is shown in Figure 2, which requires only N^2 twiddle-factor multiplies per stage.

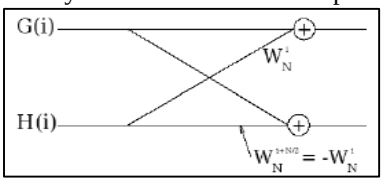

Fig. 1: Basic butterfly computation in the decimation-in-time FFT algorithm. [2]

The radix-2 decimation in time can be applied to the two length N^2 DFTs to save computation .When applied successively until the shorter DFTs reach length-2, the result is then will be radix-2 DIT FFT algorithm

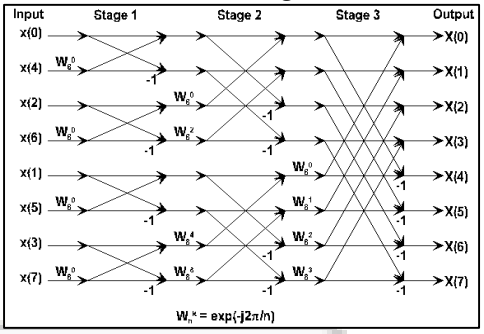

Fig. 2: Radix-2 Decimation-in-Time 8 Point FFT algorithm

II. Designing of Vedic Floating Point Complex Multiplier

The block diagram of 32-Bit Vedic Floating Point Complex Multiplier is shown in the figure below.

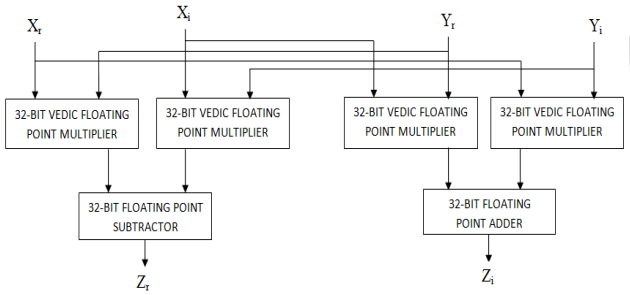

Fig. 3: Block Diagram of 32-Bit Vedic Floating Point Complex Multiplier

For designing of Vedic floating point complex multiplier; floating point adder, floating point subtractor and floating point Vedic multiplier are necessary. These three modules are the essential for the design of Vedic floating point complex multiplier. For the Vedic floating point complex number multiplier design, the most important field is mantissa calculation since the performance of Vedic floating point complex multiplier is depend upon mantissa calculation unit. In this research work we have designed the 32-bit Vedic floating point complex multiplier. Fig.3 shows the block diagram of 32-bit Vedic floating point complex multiplier module which has been designed using four 32-bit Vedic floating point multipliers modules, one 32-bit floating point adder and 32-bit floating point subtractor. Consider the two 32-bit variables X and Y whereas there real and imaginary values are Xr [31:0], Yr [31:0] and Xi [31:0], Yi [31:0] respectively .In the first step multiplication of real number Xr and multiplication of real number Yr is done. In the second step of multiplication of real number Xr and imaginary number of Yi is done. In the third step multiplication of imaginary number Xi and real number of Yr is done and in the last step multiplication of imaginary number Xi and imaginary number of Yi has been done. Finally, Zr and Zi give the output of two complex number multiplication.

Example of 32-bit Vedic floating point complex multiplier based on 32-bit single precision IEEE 754 format requires two variables with sign, exponent and mantissa fields. In this paper we have propose the Multiplication of two, 32-bit floating point number. These numbers are as follows. Xam is the real number of first variable X and Xbm is the imaginary number of first variable X. Similarly Yam is the second real number and Ybm is the second imaginary number.

Z = X * Y. Where, X = Xam + j Xbm, Y = Yam + j Ybm X= $(2.1+ j\ 3.1)_{10}$ Y= $(2.2+ j\ 3.2)_{10}$ the 32 bit IEEE format for these two number are as follows.

Xam=01000000000001100110011001100110
Xbm=01000000010001100110011001100110
Yam=01000000000011001100110011001101
Ybm=01000000010011001100110011001101

The result obtained by multiplication of these two numbers is given by, Z = Zam + j Zbm = -5.3 +j13.54.

Zam=11000000101010011001100110011000
Zbm=01000001010110001010001111010110

III. Experimental Results

A. RTL View

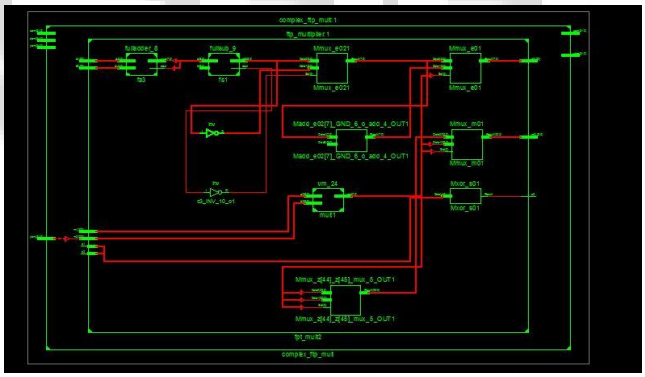

Fig. 3: RTL View of Vedic Floating Point Complex Multiplier

Fig.3: shows Register Transfer Logic view of Vedic Floating Point Complex Multiplier.

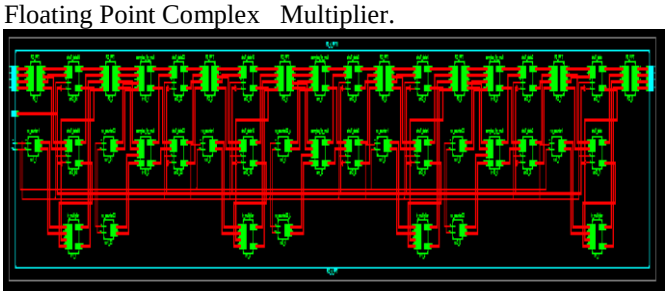

Fig. 4: RTL View of Pipelined 32 point FFT

Fig. 4 shows Register Transfer Logic view of Pipelined 32-point FFT.

B. Simulation Results

Fig.5, Fig.6 Shows the simulation result of Vedic Floating Point Complex Multiplier, pipelined 32 point FFT. Delay

obtained by Vedic floating point complex multiplier is 36.356ns and Delay obtained by pipelined 32 point FFT is 24.801ns.

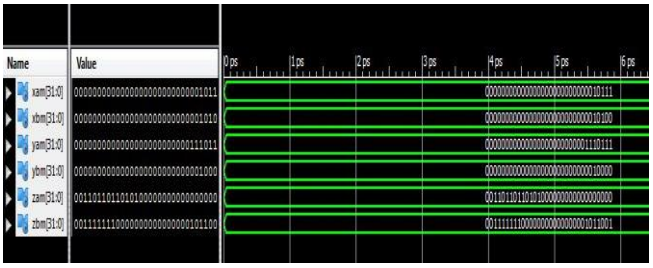

Fig. 5: Simulation result of Vedic Floating Point Complex Multiplier

Fig.5 shows simulation result of Vedic Floating Point Complex Multiplier, in which the two inputs are taken as,

xam=00000000000000000000000000010111
xbm=00000000000000000000000000010100
yam=00000000000000000000000001110111
ybm=00000000000000000000000000010000

are Vedic Floating Point Complex Multiplier of these inputs will give output as

zam=00110110110100000000000000000000
zbm = 00111111100000000000000000000000

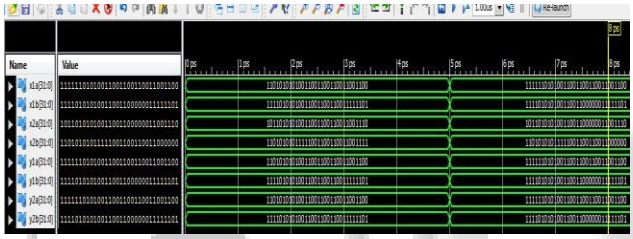

Fig. 6: Simulation result of Pipelined 32 point FFT

Fig. 6 shows simulation result of Pipelined 32 point FFT, in which the two inputs are taken as

X1a=11010101010011001100110011001100&
X1b=11110101010011001100110011111101and
X2a=10110101010011001100110011001110 &
X2b=11010101011111001100110011001111

are FFT operation on these inputs will give output as

y1a=11111101010011001100110011001100 &
y1b=11110101010011001100000011111101 and
y2a=11111101010011001100110011001100 &
y2b =11110101010011001100000011111101

IV. CONCLUSION& FUTURE SCOPE

In this paper, we use top-down design method i.e. we will design individual modules then will make amalgamation of these modules to make complete research work. In this paper we have designed Vedic 32-bit Floating Point Complex Multiplier and 32 point Pipelined FFT. VHDL language has been used to describe the system. The 32-bit single precision Vedic 32-bit Floating Point Complex Multiplier and 32 point Pipelined FFT has been designed. The design is verified through exhaustive simulations. The design, synthesis and simulation of Vedic 32-bit Floating Point Complex Multiplier and 32 point Pipelined FFT have been achieved using Xilinx 14.5i ISE tool.

If we suppose to implement this design using FPGA technology it is possible to upgrade the System with new features as per user requirements. The hardware complexity can be reduced and Integration of different circuits in a single chip can be possible on FPGA kit.

REFERENCES

[1] G.Soundarya, V.Jagan Naveen and D.Tirumala. "Implementation of 32 –point FFT processFor OFDM System". © Springer India 2015L.C. Jain et al. (eds.), Computational Intelligence, In Data Mining –volume 2,smart innovation,systems and Technologies 33, DOI 10.1007/978- 81-322-2202_20.

[2] AsmitaHaveliya, "Design and simulation of 32-point FFT use Radix-2 Algorithm for FPGA Implementation" 2012 second International conference on Advanced Computing and Communication Technologies.

[3] T.S.Ghousebhasha and Peela Sabeena Sultan" Design and Simulation of Pipelined Radix -2k Feed-Forward FFT Architectures" International Journal Of Innovative Research In Electrical, Electronics, Instrumentation And Control Engineering Vol. 2, Issue 9, September 2014.

[4] Remya Ramchandran and Vanmathi.k "Simulation of Radix-2 Fast Fourier Transform using Xilinx" Remya Ramachandran et al./International Journal of Computer Science Engineering (IJCSE) ISSN : 2319-7323 Vol. 3 No.02 Mar 2014.

[5] K.Sowjanya, B. LeeleKumari "Design and Performance Analysis of 32 and 64 point FFT Using Radix-2 Algorithm" Proceedings of AECE-IRAJ International Conference, 14th July 2013, Tirupati, India, ISBN: 978-81-927147-9-0.

[6] K. Sowjanya ,Leela Kumari Balivada "Design and Performance Analysis of 32 and 64 Point FFT using Multiple Radix Algorithms" International Journal of Computer Applications (0975 – 8887) Volume78– No.1, September 2013.

[7] Verma, P., Kaur, H., Singh, M.: VHDL implementation of FFT/IFFT blocks for OFDM. In: Proceedings of International, Conference on Advances in Recent in Communication and Computing, PI 978-1-2244-51-4-3, Kerala, pp. 186–188 (2009).

[8] Jinsing, X., Xiaochun, L., Haitao, W., Yujing, B., Decai, Z., Xiaolong, Z., Chaogang, W., Implementation of MB-OFDM transmitter baseband based on FPGA. In: 4th IEEE (ICCSC 2008), 26–28 May 2008.

[9] Li, W., Wanhammar, L.: Complex multiplication reduction in FFT processor. In: SSoCC'02 Falkenberg, Sweden, Mar 2002.

[10] Preeti, G.B., Uma Reddy N.V.: Implementation of area efficient OFDM transceiver on FPGA.Int. J. Soft Comput. Eng. 3(3), 144–147 (2013). ISSN 2231-2307.

[11] Prajapati, K., Sharma, A., Thakor, H.: Implementation of an optimized 8 point FFT inpipeline architecture for FPGA's system, vol. 1, Issue 4, (2014).

[12] U.M. Baese: Digital Signal Processing with FPGA, 3rd edn. Springer, Berlin (2007). VHDL Primer by J.Bhaskar.

