Design of 32- Point FFT Algorithm - A Literature Review

Sudhanshu Mohan Khare[1] M. Zahid Alam[2]
[1]P.G. Student [2]Associate Professor
[1,2]Department of Electronics & Communication Engineering
[1,2]Laxmi Naraian College of Technology, Bhopal, India

Abstract— Fast Fourier transform (FFT) is an efficient method to implement discrete Fourier transform (DFT) and DFT is a Fourier Transform which is used for discrete time signals to convert time domain signals into frequency domain. The Fast Fourier Transform is a very important operation in the field of digital signal processing. The paper concentrates on the implementation of the Fast Fourier Transform (FFT) through Decimation-In- Time (DIT) with Radix-2 algorithm. VHDL can be used as design entity and Xilinx Design Suite may be used for synthesis purpose.

Key words: Fast Fourier Transform, Discrete Fourier Transform, Radix, Butterfly, DIT

I. INTRODUCTION

The Discrete Fourier Transform is a very important mathematical tool which is used in discrete-time signal-processing. It is used to convert time domain signals to frequency domain. Since DFT involves a lot of calculations and thus it requires a time efficient algorithm. The Fast Fourier Transform is an efficient algorithm to calculate the Discrete Fourier Transform. This paper presents a design of 32-points FFT by using Decimation in time and radix-2 algorithm. The paper concentrates on the design of FFT which may be implemented on a FPGA kit. To implement this project coding may be done by using VHDL and simulation may be done with the help of Xilinx Design Suite. Transforms generally convert a function from time domain to another domain without any loss of information. Similarly Fourier Transform also converts a function from the time domain to the frequency domain. The DFT of a sequence x[n] of length N is expressed as:

$$X(K) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} \qquad (1.1)$$

Where, k = 0, 1, 2, 3, ……N-1

Let W_N be the complex value phase factor, which is expressed as Nth root of unity and known as Twiddle Factor, it can be written as-

$$W_N = e^{-j2\pi/N} \qquad (1.2)$$

Hence X(k) can be written as:

$$X(K) = \sum_{n=0}^{N-1} x[n]\, W_N^{nk};\ 0 \le n \le N\text{-}1 \qquad (1.3)$$

We can observe that for each value of k, direct computation of X(k) involves N complex multiplications i.e. 4N real multiplications and N-1 complex additions i.e. 4N-2 real additions. Therefore, to compute all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions. Instead of using DFT if we will use Decimation in Time (DIT) FFT with radix-2 algorithm then 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]. Therefore in this project the Decimation in Time FFT with radix-2 algorithm is used to compute the DFT of a sequence.

II. FFT

The Fast Fourier transforms are the efficient algorithms to compute the DFT. FFT algorithms are based on the concept of decomposing the computation of DFT into sequences of smaller DFTs. This operation is useful in many areas but computing it directly from the definition is often very slow to be practical. The FFT is used in many applications where the frequency-domain representation of a signal has to be analyzed. In the field of communications the FFT has gained attention because of its use in orthogonal frequency division multiplexing (OFDM) systems. FFT is the most popular technique of digital spectrum analysis. DFT is one of the fundamental operations in digital signal processing. Computation of DFT with sample input requires complex multiplications. Cooley and Tukey first introduce the concept of FFT to describe a significant computational reduction by making effective use of symmetry and periodicity properties of the twiddle factors. These properties can be described as:

Symmetry property: $W_N^{k+N/2} = -W_N^k$ (2.1)

Periodicity property: $W_N^{k+N} = W_N^k$ (2.2)

The algorithms used for the computation of the DFT are generally known as the FFTs. DFT and FFT are most popular signal processing tools. FFT computes the DFT and provides exactly the same result as obtained by evaluating the DFT definition.

Mathematically FFT can be expressed as:

$$X(K) = \sum_{n=0}^{N-1} x[n]\, W_N^{nk}\ ;\ 0 \le n \le N\text{-}1 \qquad (2.3)$$

Here W_N^{nk} is called twiddle factor and it is given by:

$$W_N^{nk} = e^{-j\left(\frac{2\pi}{N}\right)nk} \qquad (2.4)$$

The FFT algorithms are broadly classified as Decimation In Time (DIT) and Decimation In Frequency (DIF) algorithms. Several algorithms are developed to reduce the computational complexity, which includes Radix-2, Radix-4, Radix-8, Split-Radix etc. All these algorithms are developed on single method, that is, Divide and Conquer method. Fast Fourier Transform (FFT) is based on decomposition and breaking the transform into smaller sequences and at the end again combining into one transform. This paper proposes design of 32 point FFT by using VHDL as a design entity and it is synthesized in Xilinx ISE Design Suite 14.7 version.

III. DIT RADIX – 2 FFT

The implementation of FFT can be done in Decimation-In-Time FFT and Decimation-In-Frequency FFT algorithm. Both the algorithms have same computational complexity but differ in input and output computational arrangement. The name Radix-2 is called due to its base is equals to 2 and the representation is 2M, here M represents the index/stage and its value is always equals to a positive integer. In Radix-2 DIT-FFT algorithm the sequence is split into two

sequences consisting of even numbered values and odd numbered values of the input sequence x(n). The Radix-2 DIT-FFT is derived by rewriting the equation as:

$$=\sum_{n(even)} x(n)W_N^{kn}+\sum_{n(odd)} x(n)W_N^{kn}$$

$$=\sum_{m=0}^{\frac{N}{2}-1} x(2m)W_N^{2mk}+\sum_{m=0}^{\frac{N}{2}-1} x(2m+1)W_N^{(2m+1)k} \quad (3.1)$$

Butterfly is a portion of the computation of FFT that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa i.e. breaking a larger DFT up into sub transforms. The name "butterfly" arises from the shape of the data-flow diagram in the radix-2 case, as described in the diagram given below. The same diagram can also be found in the Viterbi algorithm which is used to find the most likely sequence of hidden states. Generally the term "butterfly" appears in the context of the Cooley–Tukey DIT-FFT algorithm, which recursively breaks down a DFT of composite size n = r^m into r smaller transforms of size m where r is known as the "radix" of the transform. These smaller DFTs are then combined with the help of size-r butterflies, which themselves are DFTs of size r and it is performed m times on corresponding outputs of the sub-transforms which is pre-multiplied by roots of unity which is also known as twiddle factors. This is the method of "decimation in time (DIT) FFT".

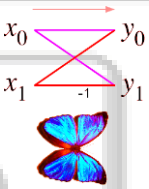

Fig.1: Basic Butterfly Operation

Butterfly is simply a DFT of size-2 that takes two inputs (*x*0, *x*1); where x_0 and x_1 are corresponding outputs of the two sub-transforms and it provides two outputs (*y*0, *y*1) by the formula:

$$y_0 = x_0 + x_1 w_n^k$$
$$y_1 = x_0 - x_1 w_n^k \quad (3.2)$$

Where ***k*** is an integer which depends on the part of the transform being computed. The radix-2 decimation in time can be applied recursively to the two length *N*/2 DFTs to save computation time and complexity. When successively applied in DIT-FFT until the shorter and shorter DFTs reach length-2, the result is the radix-2 DIT FFT algorithm.

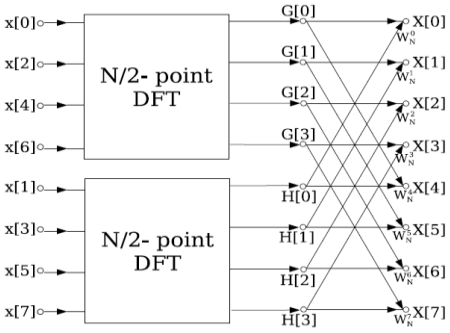

Fig. 2: Block Diagram of 8 point DIT - FFT

The radix-2 algorithm is the simplest FFT algorithm. The decimation-in-time (DIT) radix-2 FFT recursively partitions a DFT into two half-length DFTs of the even-sequence and odd-sequence time samples. The outputs of the shorter FFTs are reused to compute many outputs, thus it is reducing the total computational cost. The radix-2 decimation-in-time and decimation-in-frequency fast Fourier transforms (FFTs) are the two simplest FFT algorithms. Like all FFTs, they achieve their speed by reusing the results of smaller blocks.

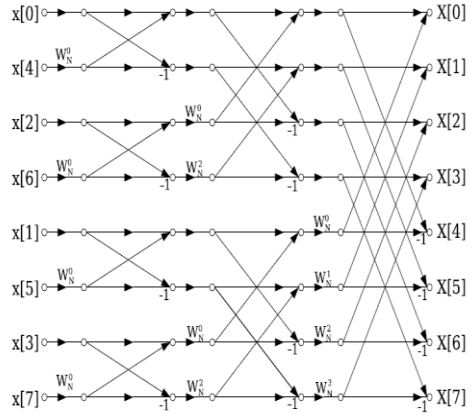

Fig. 3: Radix–2 DIT FFT of an 8 point sequence

In case of 32-point Radix-2 DIT-FFT 32 input butterfly diagram has 64 2-input butterflies and thus 64*2 = 128 multiplies. N Log N = 32 Log (32) = 128 where a straight DFT has N*N multiplies, or 32*32 = 1024 multiplies.

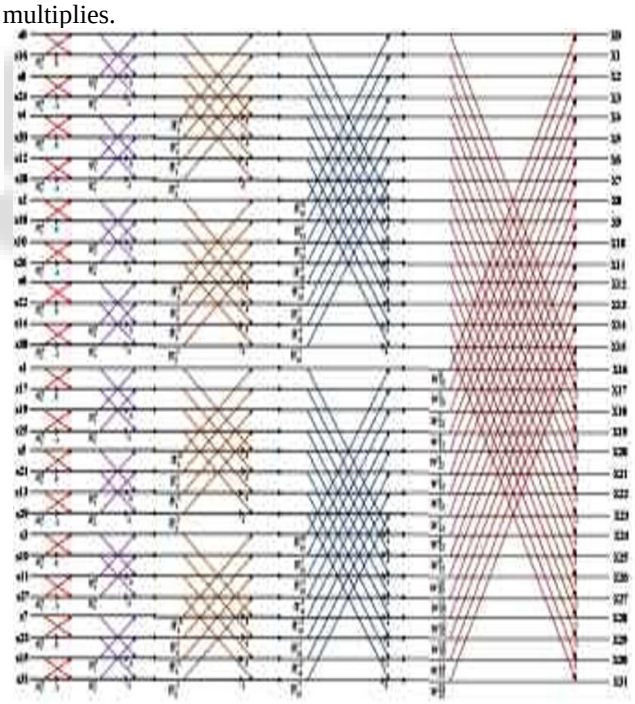

Fig. 4: 32 point DIT FFT with Radix-2 Algorithm

IV. CONCLUSION

On the basis of above discussion we can conclude that a 32 point FFT can be implemented via DIT-FFT with Radix-2 by using 16 butterflies per stage and 5 such stages are required to implement a 32 point FFT. Hence we required only 80 butterfly operations to implement a 32 point FFT. The implementation of FFT can be divided into 3 stages. First and second stage of FFT requires adders, subtractions and registers which are used to implement the equation of butterfly operation. In the third stage we need multiplier of

core generator. Purpose of this paper has been to develop an understanding of the basic principle in FFT implementation which is required in the VHDL based design used for FPGA implementation.

REFERENCES

[1] Asmita Haveliya, "Design and simulation of 32 point FFT using Radix 2 Algorithm For FPGA Implementation, 2012 Second International Conference on Advanced Computing & Communication Technologies.
[2] Remya Ramachandran Department of EEE Hindusthan College of Engineering and Technology, "Simulation of radix-2 fast fourier transform using xilinx", Remya Ramachandran et al. / International Journal of Computer Science Engineering (IJCSE)
[3] T.S. Ghouse basha1, Peerla Sabeena sulthana, "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] Afreen Fatima, "Designing and Simulation of 32 Point Fft Using Radix-2 Algorithm for Fpga", IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 9, Issue 1 Ver. III (Jan. 2014), PP 42-50 www.iosrjournals.org
[5] LEELE KUMARI, "Design and performance analysis of 32 and 64 point fft using radix-2 algorithm", 1K. SOWJANYA, 2B. Proceedings of AECE-IRAJ International Conference, 14th July 2013, Tirupati, India, ISBN: 978-81-927147-9-0
[6] Alan V. Oppenheim, Ronald W. Schafer with John R. Buck, Discrete Time Signal Processing, Second Edition.

