



## Simple Computation of DIT FFT

Neha V. Mahajan, Dr. J. S. Chitode

Dept. of Electronics,  
BVDU, College of Engineering, Pune, India

**Abstract**–FFT and IFFT algorithm plays an important role in design of digital signal processing. This paper describes the design of Decimation in Time-Fast Fourier Transform (DIT-FFT). The proposed design is a novel 16 bit word length processor, which is implemented with radix-2, based 8 point FFT. This approach reduces the multiplicative complexity which exists in conventional FFT implementation. For the number representation of FFT fixed point arithmetic has been used. The design is implemented using Verilog HDL language.

**Keywords**- DIT-FFT, Complex multiplication, Verilog, FPGA, Radix-2.

### I. Introduction

FFT and IFFT commonly used algorithm for processing signals. It can be use for WLAN, image process, spectrum measurements, radar and multimedia communication services. Now a days, FFT processor used in wireless communication system should have fast execution and low power consumption. These are the most important constraints of FFT processor. Complex multiplication is main arithmetic operation used in FFT/IFFT blocks. This is the main issue in processor. It is time consuming and it consumes a large chip area and power. When large point FFT is to be design it increases the complexity [3]. To reduce the complexity of the multiplication, there are two methods one simple method is to real and constant multiplications take the place of complex multiplication. The other method is non-trivial complex multiplication is wipe out by the twiddle factors and fulfils the processing with no complex multiplication.

In theoretical explanation of FFT, input given to FFT is normally is in floating point format. For implementing FFT implementation block called floating point arithmetic is not feasible. Creating block of floating point arithmetic is itself being complex. We will omit this and for the number representation a fixed point scheme is used. Floating point arithmetic is not required because order of magnitude of the input and the output of the FFT are similar. By using fixed point we are rounding off the numbers.

The algorithm is divided into time based (DIT) and frequency based (DIF) Fast Fourier algorithms. DIT-FFT orders the data from bit reversal order to normal order, whereas DIF-FFT is converse. DIF-FFT is easier to design than DIT FFT. FFT algorithm can be implemented with radix 2 or radix 4. In our work we have designed it in radix-2 format. The basic idea of these algorithms is to divide the N-point FFT into smaller ones until two point FFT is obtained. Hence the algorithm is called radix-2 algorithm.

The remainder of this paper is organized as follows. Section II explains details of Discrete Fourier transform. Section III describes radix 2 DIT FFT algorithm, fixed point number representation and complex multiplication. Section IV presents simulation result, design summary. Last section concludes the design.

### II. Discrete Fourier Transform

The Fourier transform is mathematical method of changing time representation of signal into frequency representation. It transforms one function from time domain to frequency domain.[2]. The DFT of a input sequence  $x[n]$  can be computed using the formula given by equation 1

$$x(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \leq k \leq N-1 \quad (1)$$

$$W_N = e^{-j2\pi/N}$$

Where, is twiddle factor?  $X(n)$  is the input sequence.  $X(k)$  is the k the harmonic[3].

### IDFT

It is inverse Discrete Fourier Transform. It is exactly reverse of that of DFT. The N-point IDFT with  $\{n \in Z | 0 \leq n \leq N-1\}$  is given by equation 2

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k) e^{j2\pi kn/N} \quad (2)$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} x(k) W_N^{-kn} \quad (3)$$

Equation 3 decomposes into two sums

$$x(n) = \frac{1}{N} \sum_{m=0}^{\frac{N}{2}-1} X(2m) W_N^{-2mn} + \frac{1}{N} \sum_{m=0}^{\frac{N}{2}-1} X(2m+1) W_N^{-n(2m+1)} \quad (4)$$

$$x(n) = \frac{1}{N} \sum_{m=0}^{\frac{N}{2}-1} F1(m) W_N^{-mn} + W_N^{-n} \frac{1}{N} \sum_{m=0}^{\frac{N}{2}-1} F2(m) W_N^{-mn} \quad (5)$$

$$x(n) = f1(n) + W_N^{-n} f2(n) \quad (6)$$

$f1(m)$  and  $f2(m)$  are inverse Fourier transform of  $F1(m)$  and  $F2(m)$ . This structure is recursive. This is well known as IFFT which is presented by Cooley and Tukey in 1965[1][5]. Twiddle factor is a complex multiplicative as root of unity constant in FFT algorithm. In the practical implementation we need to put input as a real value which is in time domain to reduce the complexity. FFT is computational algorithm to implement DFT of samples where  $N$  is length of the samples.

### III. A Radix 2 Dit – Fft Algorithm

Flow graph for DIT- FFT decomposition for 8 point is shown in the fig 1.

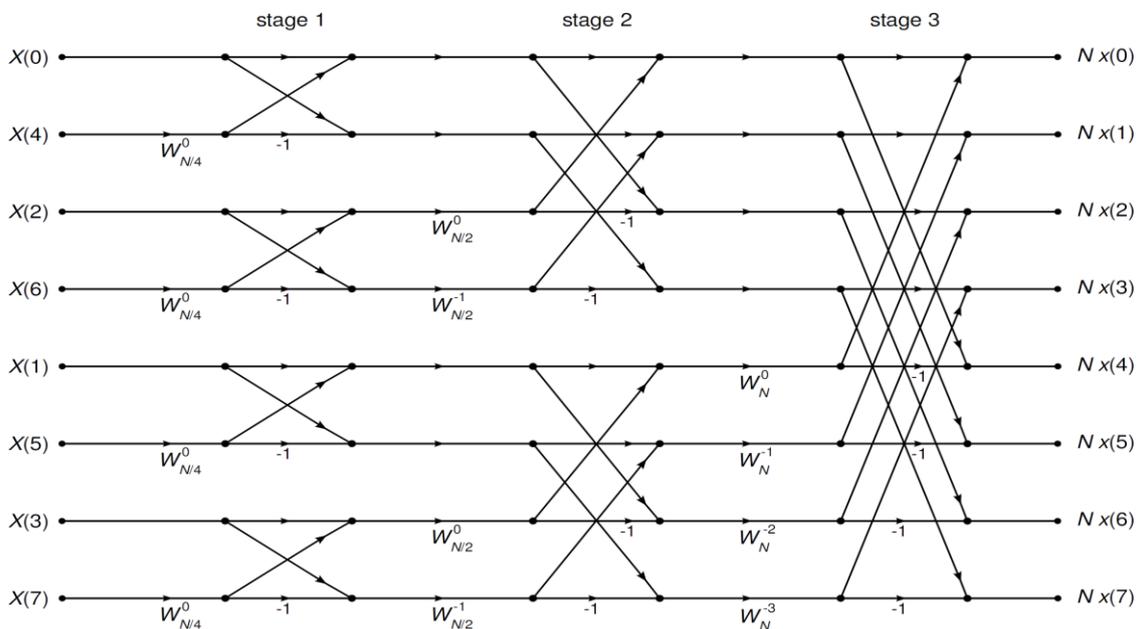


Figure 1: 8 Point DIT-FFT [1]

#### A. Butterfly Unit

The basic module for implementation is butterfly module which is shown in the fig 2.

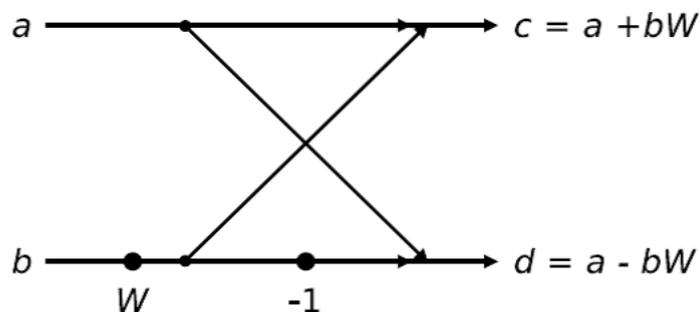


Figure 2 : Butterfly Unit [1]

There are two inputs called  $a$ ,  $b$  and two outputs  $c$ ,  $d$  and twiddle factor  $W$ . Output is as shown in equation (7) and (8).

$$C = a + bW \quad (7)$$

$$D = a - bW \quad (8)$$

With these butterfly unit we can built whole FFT structure[1]. If  $N$  is the input for FFT then stages are required, each stage requires  $N/2$  butterflies. As we can see from above fig that one butterfly unit requires 1 complex multiplier



Design summary for this design is given in figure 5

Device Utilization Summary (estimated values)			[1]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	868	42000	2%
Number of Slice LUTs	1058	21000	5%
Number of fully used LUT-FF pairs	434	1492	29%
Number of bonded IOBs	513	210	244%
Number of BUFG/BUFGCTRLs	1	32	3%

Timing Summary for the same design is given below

Minimum period: 6.528ns (Maximum Frequency: 153.175MHz)

Minimum input arrival time before clock: 1.535ns

Maximum output required time after clock: 0.645ns

## V. Conclusion

In this paper, a novel 8-point DIT-FFT processor is implemented using radix-2. This helps in reducing the complex multiplication. This paper also describes how to avoid floating point arithmetic for implementation of FFT. In future, the work can be extended to the N bit variable input signals. The implemented design can be used as a basic block for further computation. The pipelined architecture can also be added to FFT for providing fast and better performance. The proposed processor can be integrated with other components which can be used as a stand-alone processor for many applications.

## References

1. Michael Bernhard, Joachim Speidel "Implementation of an IFFT for an Optical OFDM Transmitter with 12.1 Gbit/s" Universität Stuttgart, Institut für Nachrichtenübertragung, 70569 Stuttgart.
2. Sneha N.kherde, Meghana Hasamnis, "Efficient Design and Implementation of FFT" Rashtrasant Tukdoji Maharaj Nagpur University, IJEST.
3. Mounir Arioua, Said Belkouch, Mohamed Agdad, "VHDL implementation of an optimized 8-point FFT/IFFT processor in pipeline architecture for OFDM" Morocco. IEEE 978-1-61284-732-0.
4. Kasina Madhusudhana Rao, M.Tech, V.Ravi Tejesvi Asst.Prof, G. Anantha Rao Asst.Prof "Verilog Implementation of 32 Point FFT Using Radix-2 Algorithm on FPGA Technology" IOSR-JECE, Volume 9, Issue 1, Ver. II. e-ISSN: 2278-2834.
5. James W. Cooley, John W. Tukey "An algorithm for machine calculation of complex Fourier series", JSTOR.