RESEARCHERID
THOMSON REUTERS

[Mehrotra* *et al.*, 6(5): May, 2017]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IMPLEMENTATION OF FFT ALGORITHM

Mitul Mehrotra*, Geetika Pandey, Mandeep Singh Narula
*Department of Electronics and Communication Engineering Jaypee Institute of Information Technology, Noida, India

DOI: 10.5281/zenodo.573508

ABSTRACT

Fast Fourier Transform (FFT) is a fast and efficient way of computing Discrete Fourier Transform (DFT). FFT is one of the exquisite and ubiquitous operations in the field of digital signal processing. Moreover, it is one of the critical components in Orthogonal Frequency Division Multiplexing (OFDM) [5] systems. FFT can be of two types, namely: Decimation in time (DIT) FFT and Decimation in frequency (DIF) FFT. For most of the real life situations like audio/image/video processing etc., DIT-FFT has an advantage over DIF-FFT since it does not require any output recording. In this paper, an efficient algorithm to compute 8 point FFT has been devised in which a butterfly unit (stage-I) computes the output and then feeds those outputs as inputs to the next butterfly units (stage-II/III) so as to compute the overall FFT.

KEYWORDS: FFT, DFT, Twiddle factor, Butterfly unit, Bit reversal.

INTRODUCTION

FFT is very popular for transforming a signal from time domain to frequency domain and it has quite an interesting history starting from 1805, when Carl Fredrich Gauss tried to determine the orbits of various asteroids from sample locations. He thereby developed the DFT algorithm even before Fourier published his results for the same in 1822. He developed an algorithm similar to that of Cooley [4] and Tukey [4] but Gauss never published his method or algorithm in his lifetime. It took another 160 years until Cooley and Tukey reinvented the FFT. During this period of 160 years from 1805 to 1965, many other scientists invented various efficient algorithms to compute FFT but none of them was as general as that of Gauss's algorithm.

The results of FFT are same as that of DFT; the only difference is that the algorithm is optimised to remove the redundant calculations, the FFT greatly reduces the amount of calculations required and hence reduces the time required for the computation. Functionally, the FFT decomposes the set of data to be transformed into a series of smaller data sets to be transformed. Then it decomposes those smaller sets into even smaller sets. At each stage of processing, the results of the previous stage are combined in a special way. Finally, it calculates the DFT of each small data set. DFT can be computed using the below given formula:

$$X[k] = \sum_{k=0}^{N-1} x_k \; e^{-\left(\frac{2\pi j}{N}\right)nk} \;\; ; \qquad k = 0,1, \ldots . N-1.$$

A major drawback of this DFT algorithm is the computational complexity. For a sequence of length N, it has a complexity given as: $O(N^2)$ hence it is not a very efficient method and here the FFT comes into the picture. FFT significantly reduces the number of computations required for a sequence of length N from $O(N^2)$ to O(N logN) where log is the base-2 logarithm. FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

spectrum. So we split the N-point data sequence into two N/2 point data sequences $f_1(n)$ and $f_2(n)$, corresponding to the even-numbered and odd-numbered samples of x(n) respectively.

$$f_1(n) = x(2n)$$

$$f_2(n) = x(2n+1)\,; \qquad n = 0,1,2,\dots,\frac{N}{2}-1.$$

Here $f_1(n)$ and $f_2(n)$ are obtained by decimating x(n) by a factor of 2, and hence the resulting FFT is called a decimation-in-time (DIT) algorithm. Now the N-point DFT can be expressed as:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \;; \qquad k = 0,1,2,\dots,N-1.$$

$$= \sum_{n\ even}^{N-1} x(n)\, W_N^{kn} + \sum_{n\ odd}^{N-1} x(n)\, W_N^{kn}$$

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

IMPLEMENTATION OF FFT

To compute FFT, we need to break a data set into smaller data sets and then compute the DFT of each small set. Figure 1 describes the implementation of 8-point DFT. We observe that computation is performed in three stages, starting with the computation of four 2-point DFTs, then two 4-point DFTs, and finally one 8-point DFT.

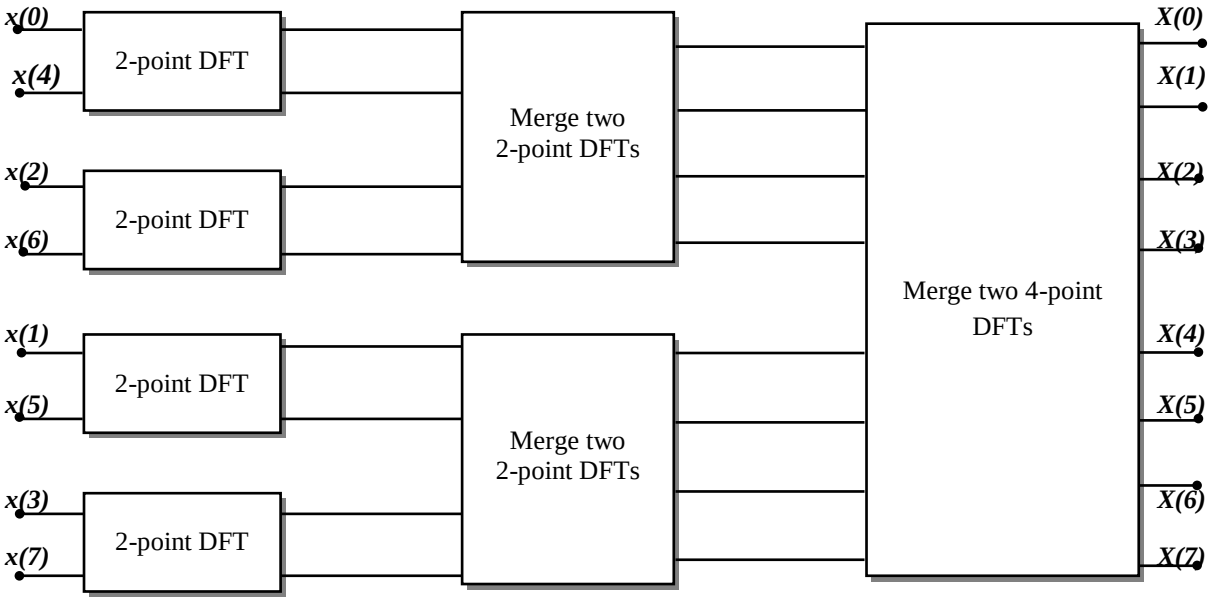

Fig 1: FFT block diagram

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

BUTTERFLY UNIT [7]

In radix-2 Cooley-Tukey algorithm, butterfly is simply a 2-point DFT that takes two inputs and gives two outputs. Butterfly unit is the basic building block for FFT computation. The figure 2 shown below describes the basic butterfly unit used in FFT implementation.

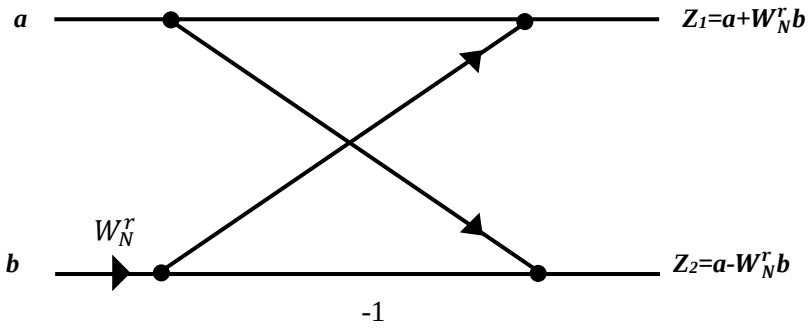

Fig 2: Basic butterfly unit

Implementation of FFT requires the computation of butterfly unit at first, which takes two complex inputs 'a' and 'b' and a twiddle factor 'W'. It generates two complex outputs 'Z_1' and 'Z_2'.

- Input 'b' is multiplied with 'W' and then added to 'a' for the output 'Z_1'.
- Input 'b' is multiplied with '-1' and then added to 'a' for the output 'Z_2'.

This basic butterfly unit is replicated four times each in the three stages of the FFT and the final output is generated as shown in the figure 3 below:

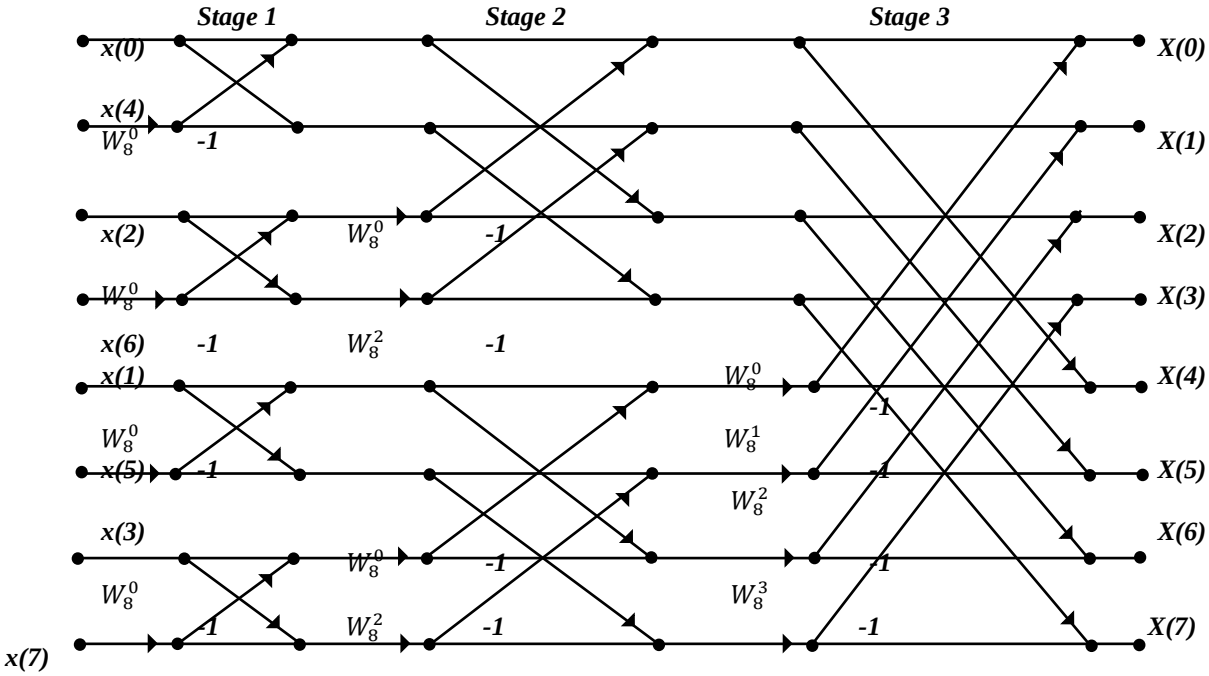

Fig 3: FFT butterfly diagram

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

BIT REVERSAL [8]

In FFT computation, the input bits need to be bit reversed so as to obtain the output bits in the normal order. To achieve this, the input sequence is broken into even and odd parts based on their indexes. Then these even parts are again broken down into even and odd parts. Same is done with odd parts respectively. This procedure is continued until only two points are left. In this paper, the sequence is separated in time domain. For example, if we consider the case where N=8, input sequence is x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7) and after the bit reversal process, the output sequence obtained is x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7) as shown in table 1 below:

Table 1: Bit reversal

Input	Address (binary)	Output (bit reversed)	Address (binary)
x(0)	000	x(0)	000
x(1)	001	x(4)	100
x(2)	010	x(2)	010
x(3)	011	x(6)	110
x(4)	100	x(1)	001
x(5)	101	x(5)	101
x(6)	110	x(3)	011
x(7)	111	x(7)	111

RESULTS

Result of FFT for the following input has been verified on MATLAB:
x(n) = (1, 2, 4, 8, 16, 32, 64, 128);
and the output was recorded as:
y(n) = (255, 48+166i, -51+102i, -78+46i, -85, -78-46i, -51-102i, 48-166i);

The waveform of input sequence is shown in figure 5 below and the corresponding output sequence waveform is shown in figure 6 below. Figure 4 displays the verified output on MATLAB software.

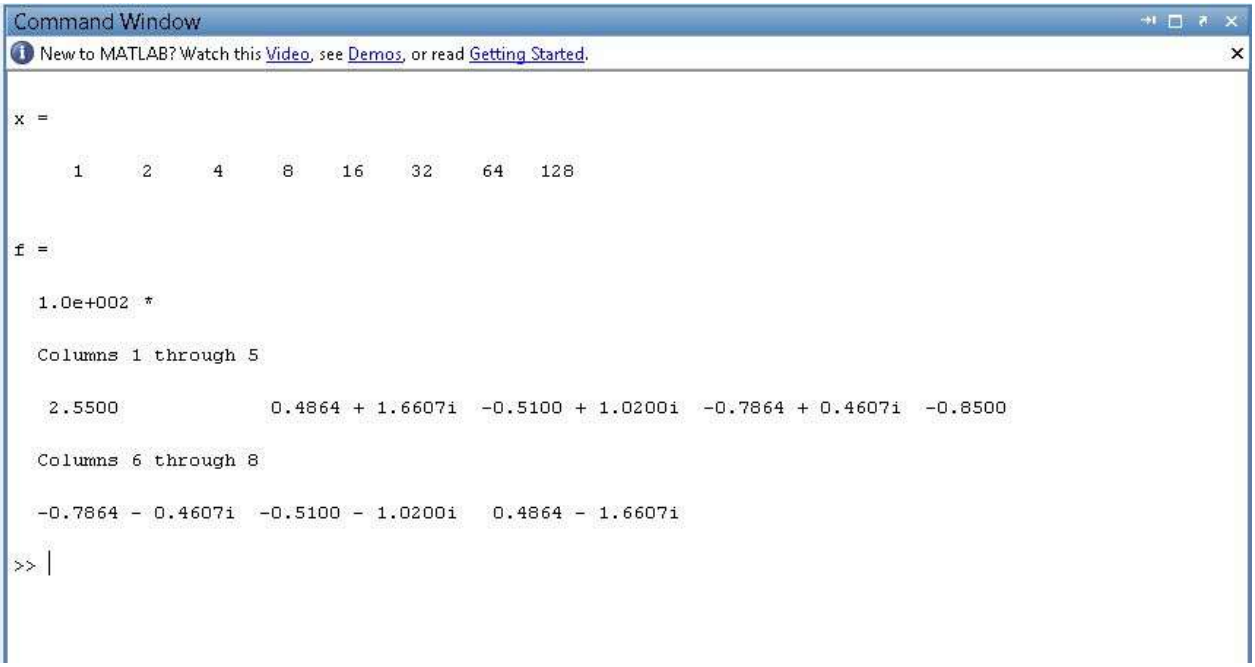

Fig 4: Output verification on MATLAB [9]

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

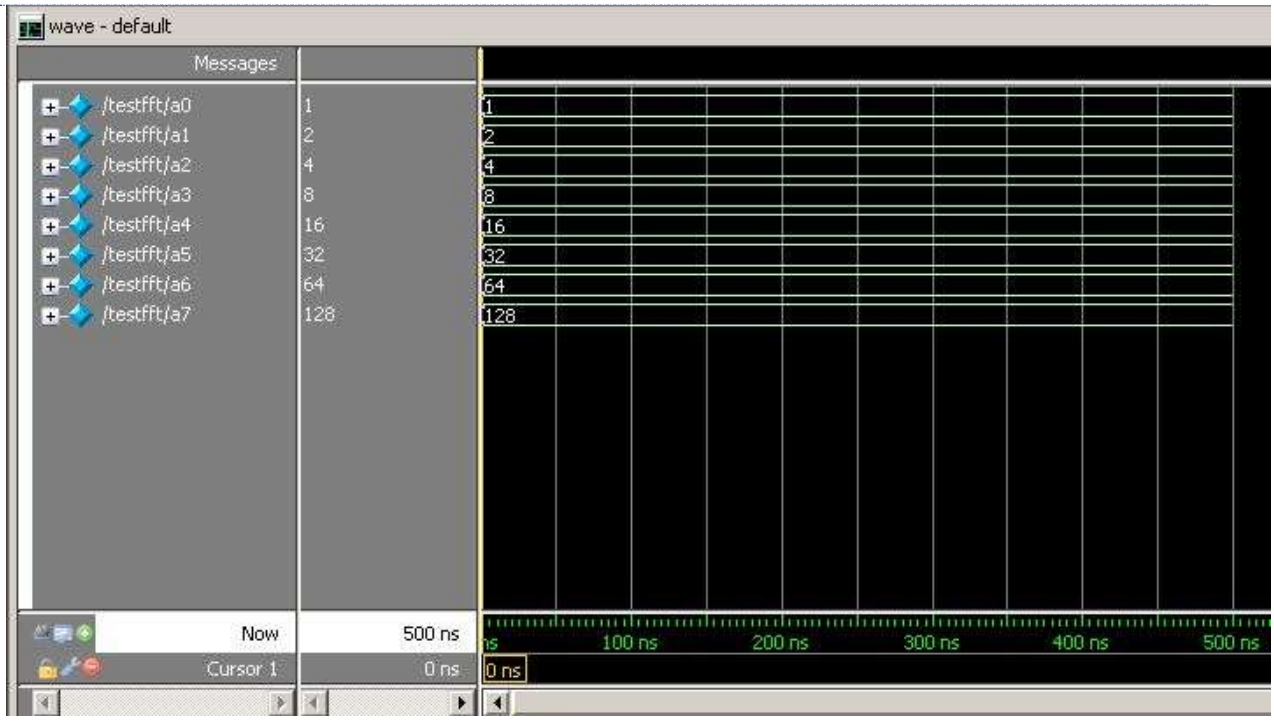

Fig 5: Input sequence

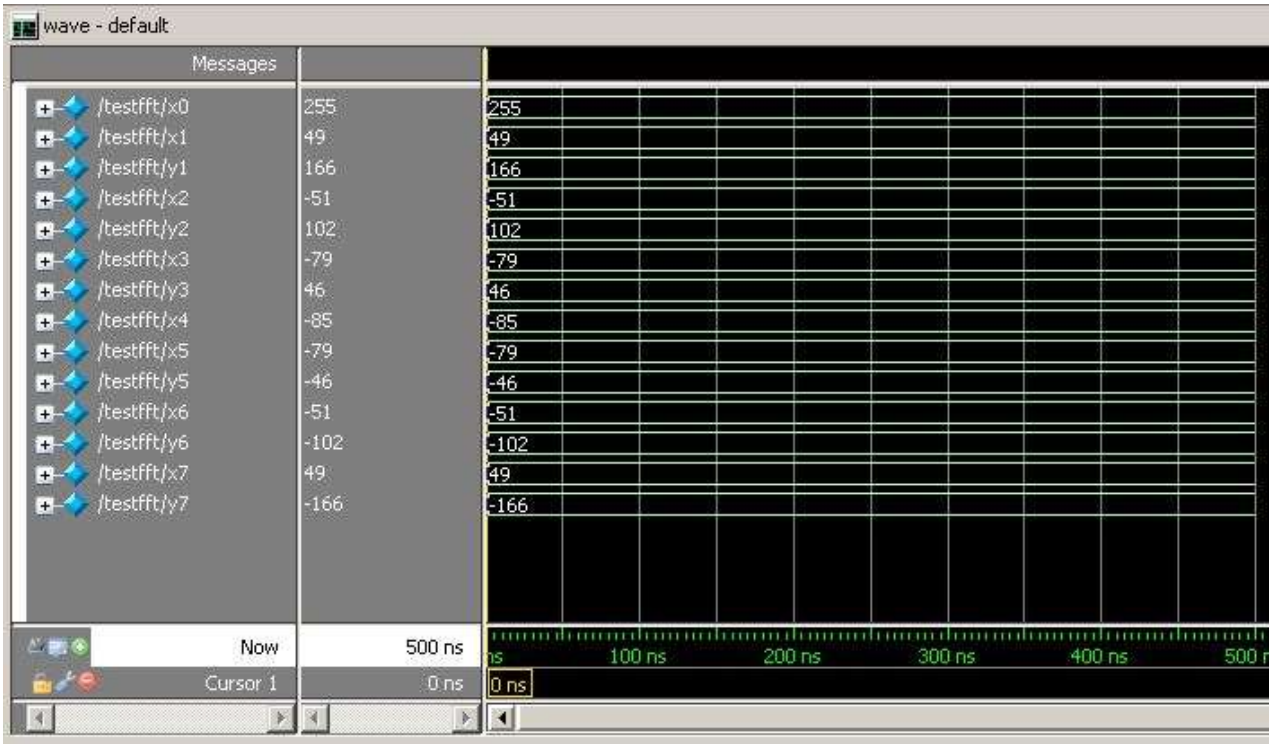

Fig 6: Output sequence

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

CONCLUSION

The Fast Fourier Transform (FFT) is simply a professional method to compute the Discrete Fourier Transform (DFT). Use of FFT reduces the complexity and the time required for the computation of DFT. A verilog implementation of floating point FFT with bit reversal has been generated using single precision floating point number IEEE 754 standard.

REFERENCES

[1] Swartzlander, E.E and Saleh, H.H.,2012. FFT implementation with fused floating-point operations. IEEE transactions on computers, 61(2), pp.284-288.
[2] Nussbaumer, H.J., 2012. *Fast Fourier transform and convolution algorithms* (Vol. 2). Springer Science & Business Media.
[3] Chen, Y.H. and Chang, T.Y., 2012. A high-accuracy adaptive conditional-probability estimator for fixed-width Booth multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *59*(3), pp.594-603.
[4] Puschel, M. and Moura, J.M., 2008. Algebraic signal processing theory: Cooley–Tukey type algorithms for DCTs and DSTs. *IEEE Transactions on Signal Processing*, *56*(4), pp.1502-1521.
[5] Lin, Y.W. and Lee, C.Y., 2007. Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(4), pp.807-815.
[6] Frigo, M. and Johnson, S.G., 2005. The design and implementation of FFTW3. *Proceedings of the IEEE*, *93*(2), pp.216-231.
[7] Yeh, W.C. and Jen, C.W., 2003. High-speed and low-power split-radix FFT. *IEEE Transactions on Signal Processing*, *51*(3), pp.864-874.
[8] Rubio, M., Gómez, P. and Drouiche, K., 2002. A new superfast bit reversal algorithm. *International Journal of Adaptive Control and Signal Processing*, *16*(10), pp.703-707.
[9] Trefethen, L.N., 2000. *Spectral methods in MATLAB*. Society for Industrial and Applied Mathematics.

CITE AN ARTICLE:

Mehrotra, Mitul , Geetika Pandey, and Mandeep Singh Narula. "IMPLEMENTATION OF FFT ALGORITHM ." *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY* 6.5 (2017): 206-11. Web. 10 May 2017. <http://www.ijesrt.com/issues%20pdf%20file/Archive-2017/May-2017/28.pdf>.