Discrete Fourier Transform Analysis with Different Window Techniques Algorithm

Bharti Thakur
M.E. Scholar
National Institute of Technical Teachers' Training & Research Chandigarh, India-160019

Rajesh Mehra
Associate Professor
National Institute of Technical Teachers' Training & Research Chandigarh, India-160019

ABSTRACT
While designing the digital circuits in today's world, the most desired factors are high performance, speed and cost. FFT is one of the most efficient ways to meet these requirements. In this paper, authors have discussed the DFT algorithm on periodic waveform using different window techniques using the FFT algorithm. This paper shows that the window techniques reduces the spectral leakage and a higher order DFT can be realized very easily using a lower order FFT. Different window techniques are used here on periodic waveforms and simulation is done using Matlab 2015.

Keywords
Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT), Matlab.

1. INTRODUCTION
In digital world, frequency analysis is the most common and convenient method. A time domain signal is first converted into frequency domain signal and then a frequency analysis is performed on a discrete time signal{x (n)}. In Fourier Transform, a waveform is decomposed into a sum of sinusoids of different frequency components. Fourier Transform converts a signal from one domain to another, mostly from time domain to frequency domain and vice versa. In the digital world, signals are sampled in time domain [2], [3].so we have Discrete Fourier Transform (DFT). DFT is applied to the discrete input signal and we get the signal in frequency domain at the output. When we perform inverse DFT on frequency domain, the resulting signal is in time domain. Thus, the signal when converted into frequency domain will have various components in frequency and can be used to remove certain undesired frequency components. Discrete Fourier Transform is a computationally complex process.

A fast Fourier Transform is used to compute the Discrete Fourier transform (DFT) of a sequence or its inverse as it is a very efficient method to do so. The computational complexity of DFT is O (N^2). The time complexity of the FFT algorithm proposed by Cooley and Tukey was O ($N\log_2 N$), where N is the FFT size. This is a very popular technique in multicarrier modulation scheme and in high speed communication systems [4]. The FFT distinguishes the different frequency sinusoids along with their amplitudes that combine to form an arbitrary waveform.

The DFT and FFT are the most popular and important of all the well-known transforms because they give the most adequate representation in frequency domain and the FFT can be computed very rapidly. Fourier analysis has been in existence since its publication since 1822 by Fourier and has therefore achieved a high degree of familiarity, respectability and development. FFT is widely used in linear filtering, spectral analysis, digital communication, image processing remote sensing, speech processing, geological exploration and secure wireless communication etc.

In this paper, the analysis of DFT with different window techniques using FFT is shown and a comparison is made between these window techniques. The paper is organized as follows: section II discusses the DFT and FFT algorithm implementation, section III will be devoted to FFT architecture. Section IV includes the window based analysis of FFT and the conclusion is given in section V.

2. DFT AND FFT
DFT is used to transform time domain to frequency domain and to do the reverse for the implementation on digital hardware. DFT operates on a finite N point sequence, x (n). The N point DFT of the given sequence is expressed as,

$$X(k) = \sum_{n=0}^{N-1} x(n).e^{-j2\pi nk/N}, k = 0,1,\ldots,N-1 \quad (1)$$

And the corresponding IDFT is,

$$x(n) = \sum_{k=0}^{N-1} X(k).e^{j2\pi nk/N}, n = 0,1,\ldots,N-1 \quad (2)$$

Here x (n) represents the time domain as n represents the discrete time domain index and X (k) represents the frequency domain components where k is the normalized frequency domain index. A more simplified equation can be given as,

$$x(k) = \sum_{n=0}^{N-1} x(n).W_N^{nk}, k = 0,1,\ldots,N-1 \quad (3)$$

Where, $W_N = e^{-j2\pi/N}$ is the Twiddle-Factor. The frequency domain data can be changed into time domain by employing Inverse Discrete Fourier Transform. From the computation of each value of k, it is clear that for the direct computation of X (k), N complex multiplications are needed. This means that, for computing all the N values of DFT it requires N^2 complex multiplications and N^2-N complex additions [4]. Computing DFT is inefficient as this algorithm does not exploit the periodicity and phase symmetry properties of the twiddle factor.

In 1965, Cooley and Tukey gave a new technique that reduced the computational complexity to $N\log_2 N$. Thereafter, various FFT algorithms were developed such as radix-2, radix-4 and split radix algorithm. The structure of these algorithms is a constant butterfly. The approach implies decomposition of an N point DFT into successively smaller size DFTs. The number N can be factorized as,

$$N = r_1, r_2, r_3, r_4, \ldots\ldots, r_v \quad (4)$$

Here every r is prime.

If, $r_1 = r_2 = r_3 = r_4 = r_v = r$ then, $N = r^v$ (5)

Here, r is the radix of FFT algorithm and v indicate the number of stages in the FFT algorithm.

The FFT algorithms make the DFT operations more practical. The fundamental approach of all the FFT algorithms proposed is to make use of the properties of the DFT algorithm. This reduces the computational cost of performing direct DFT on the input sequence. The Twiddle factor is a trigonometric function over discrete points in a two dimensional plane and it has two basic properties as: symmetry and periodicity.

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

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

These properties help eliminating the unnecessary computations. This is a universal algorithm as any factorization of N is possible. FFT algorithm relies on divide and conquer methodology, dividing N coefficients points, yielding N/2 blocks each computing the addition and subtraction of the coefficients scaled by the corresponding twiddle factor called a butterfly for its cross over appearance[5]. These results are then used to compute next state of N/4 blocks which combines the result of 2 previous blocks, thus combining four coefficients at this point. This process is then repeated till we get one main block with the final computation of all the N coefficients.

Let, N= M.T, N= DFT length, k= s.T.t,

n= l+ M .n, where, M and N are integers and S, L€ {0, 1, …., M-1} and t, m € {0, 1,……T-1}. Applying these considerations in equation (3) we get,

$$X(s+T.t) = \sum_{l=0}^{M-1} \sum_{m=0}^{T-1} [x(l+M.m)] W_{MT}^{(l+M.m)(s+Tt)} \quad (6)$$

This equation can be written as,

$$X(s+T.t) = \sum_{l=0}^{M-1} W_M^{lt} \sum_{m=0}^{T-1} [W_{MT}^{ls}\ x(l+Mm) W_T^{ms}] \quad (7)$$

From the above equation, it is clear that in order to realize an N point DFT, first decompose it into one M point and one T point FFT where, N= M.T and then combine then in the end. For example, if we want to perform 64 point FFT, we can go for, M=T=8. Using equation (7), we get:

$$X(s+8) = \sum_{l=0}^{7} W_8^{lt} \sum_{m=0}^{7} [W_{64}^{ls}\ x(l+8m) W_8^{ms}] \quad (8)$$

This equation represents the two dimensional structure of 64 point FFT, represented by 8 point FFT. Thus the performance here will now depend on 8 point FFT performance. Here, the split radix architecture has been taken as it includes a lower number of arithmetic operations.

3. FFT ARCHITECTURE

There are various methods using which 64 point FFT can be realized. One of the realization is shown in the following figure.

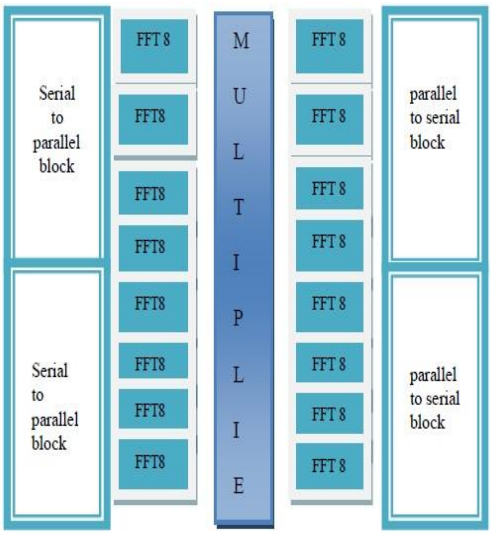

Fig.1 Spatial distribution of FFT Architecture [7]

From the figure it is clear that 64 point FFT is realized on five levels. The first level includes two serial to parallel blocks that are used to store real and imaginary parts of the serial data. The second block consists of eight blocks of 8 point FFT split radix DIT. In the third block, there are 49 complex multiplier blocks used to compute the non-trivial complex multiplication. The fourth block also has eight blocks of 8 point FFT split radix DIT. In the last block, there are two parallel to serial locks giving final data in a serial way. After the first five clock cycles, the 8 point FFT outputs are available and thus the multiplication can be initiated. The block multiplier needs at least two clock cycles to perform 49 complex multiplications [6]. Once the last stage outputs of 8 point FFT transformations occurs, it takes another 5 clock cycles to generate 64 point FFT outputs. This architecture has various advantages like, high speed and low latency but some limitations are also present like, high memory requirements, higher number of complex multipliers and adders thus making it inefficient for low cost FPGAs such as, Sparten3 family.

4. WINDOW BASED ANALYSIS

In digital signal process, the window function is a mathematical function that has zero value outside of some chosen interval. For example, a function that is constant inside the interval and zero everywhere else is called a rectangular window which describes the shape of its graphical representation. Windowing technique is used to shape the time portion of the measurement data and to minimize edge effects that may result in spectral leakage otherwise in FFT spectrum. The window function when used correctly, increases the spectral resolution of frequency domain results. When FFT is used to measure the frequency component of the signal, it is assumed that the analysis is based on a finite set of data. The FFT transform assumes that the finite data set is one period of a periodic signal [7]. For FFT, both the time domain and frequency domain appears to be circular topologies, so the two endpoints of the time waveform are interpreted as though they were connected together [8]. If the measured signal is periodic, and the acquisition time interval is an integer number of periods, the FFT comes out fine as the assumption is matched. However if the measured signal is not an integer number of periods, in that case the sampling record being finite results in truncated waveform having different characteristics as compared to the original continuous time

domain signal. This introduces the sharp transition changes into measured data signal, which results into discontinuities at the end points. These discontinuities appear in the FFT as high frequency components that are not present in the original signal. These frequencies are aliased between 0 and half of the sampling rate as these frequencies are much higher than the Nyquist frequency. The spectrum thus obtained is the smeared version of the actual spectrum of the original signal. This phenomenon is called as spectral leakage due to which the fine spectral lines are spread into wider signals. This effect can be minimized by applying the window techniques to the measured signal in time domain. Windowing reduces the amplitude of these discontinuities at the boundaries of each finite sequence acquired by the digitizer [1]. In windowing, the time record is multiplied by a finite word length window with an amplitude that varies gradually and smoothly towards zero at the edges. Thus, the end points of the waveform meet and, the result is a continuous waveform without any sharp transitions. However an appropriate window function must be applied for a specific application. If the window is not applied correctly, errors are introduced in FFT amplitude that distorts the overall FFT amplitude, frequency and shape of the spectrum

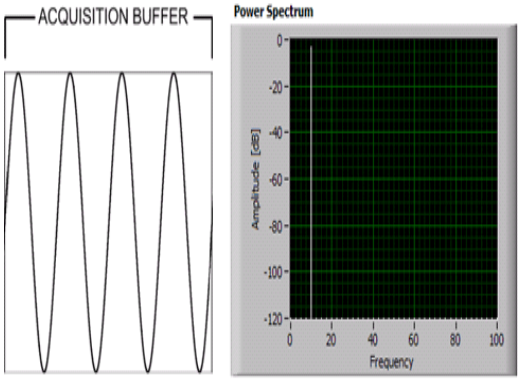

Fig 2. Ideal FFT (measuring an integer number of periods) [7].

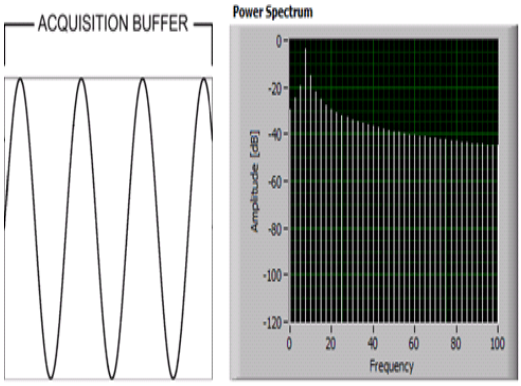

Fig 3. Spectral leakage of FFT (measuring non-integer number of periods) [7].

The various windows that are widely used are Rectangular Window, Triangular window, Hanning Window, Hamming Window. **Fig. 2** shows that the ideal integer number of periods gives an ideal FFT. **Fig. 3.** Shows that the non-integer number of periods leads to spectral leakage. Most of the times, in the predefined data block time periods the signals are not periodic, so a window must be applied to avoid the leakage. These functions simply "tails off" the signal at both ends, thus reducing the DFT leakage. Each point in the signal is multiplied by a window scaling factor before calculating the DFT. While applying the window, it should be zero at the beginning and end of the data block and has some shape in between. This function when multiplied with the time data block, forces the signal to be periodic. A weighting factor is also applied so that the correct FFT signal amplitude is recovered after the windowing.

In the following figures, the discrete samples of a periodic waveform are shown in the upper plot and the absolute values of their Discrete Fourier Transform (DFT) obtained using a Fast Fourier Transform (FFT) are shown in the lower plot. The frequencies shown are from 0 to 100 hertz, sampling rate

is 200 Hz, which means that the Nyquist Frequency is 100 Hz. The DFT at frequencies above Nyquist frequency is same as the DFT at lower frequencies.

The effect of Rectangular Window on sinusoidal waveform is shown in **Fig.4.** The Rectangular Window is a good choice while dealing with the transient events such as hammer excitation. It is actually not a window at all, as the transient events starts and ends at zero amplitude i.e. within the finite word length of the record. Using the Rectangular window function only truncates the signal within a finite word length. Using DFT with rectangular window is generally not recommended due to high side lobes.

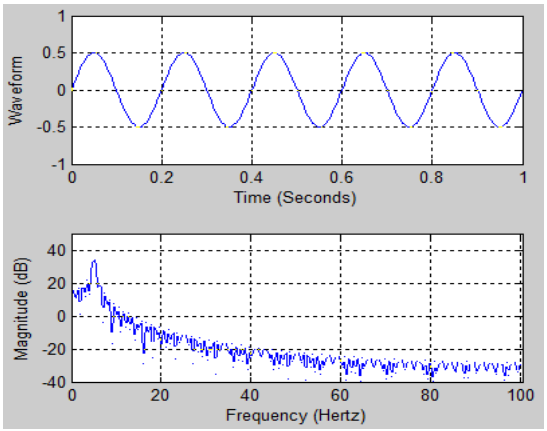

Fig.4. DFT with Rectangular Window

From fig.4 it is clear that the graph extends to a wide frequency range for the rectangular window and it gives information about every component of frequency within the range. Also, as it covers the whole event uniformly, no amplitude corrections are needed which ensures great fidelity.

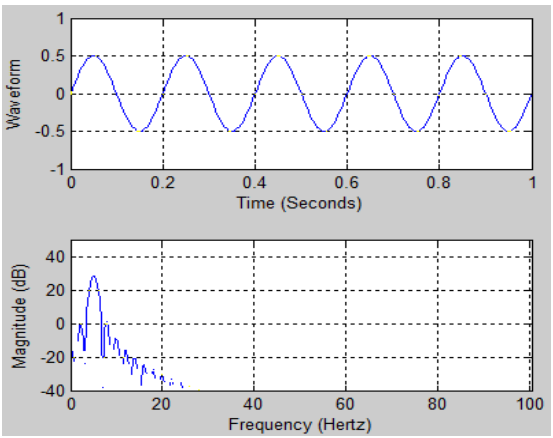

Fig.5. DFT using Triangular Window

The amplitude of time record is forced to zero both at the beginning and at the end of sample interval in Hanning Window shown in **Fig6**. This leads to variations in the amplitude of the signal being analysed over the time record i.e. distortion in the form of amplitude modulation. Amplitude modulation in the waveform leads to sidebands in its spectrum.

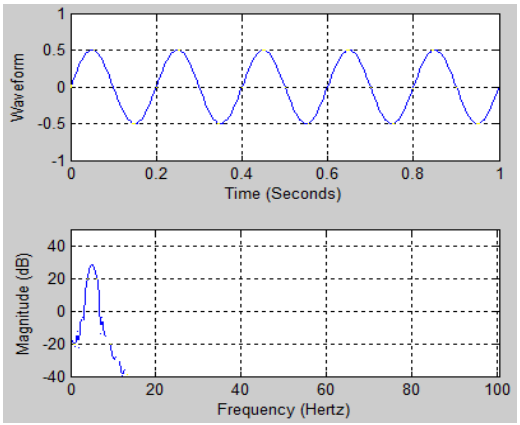

Fig.6. DFT using Hanning Window

In the case of Hanning Window, the side lobes reduces the frequency resolution of the analyzer effectively. Therefore, the Hanning Window should always be used for the continuous signal, not with the transients.

The effect of using Hamming Window is shown in Fig.7. The Hamming Window is a modification over Hanning Window The shape of the Hamming Window resembles that of the cosine wave. Hamming Window and Hanning Window are almost similar. The only difference between the two is that, at the edges the Hamming Window does not get as close to zero as does Hanning window.

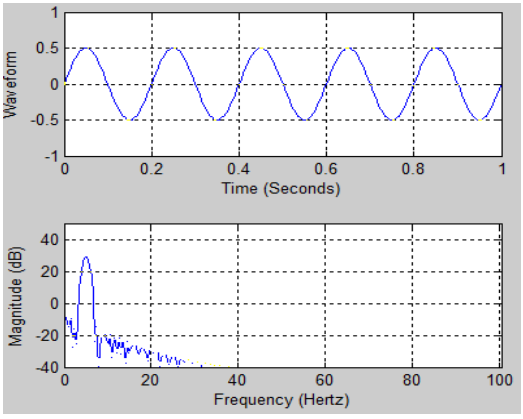

Fig.7. DFT using Hamming Window

5. CONCLUSION

This paper presents the DFT algorithm analysis on periodic waveform using different window techniques using FFT, effect of integer and non-integer periods of the waveform, how spectral leakage is reduced using different window techniques. Various window techniques are used such as Triangular Window, Rectangular Window, Hanning window and Hamming Window. Each window has its own unique advantages and disadvantages. According to the kind of signal being analyzed a specific window function should be selected. It interprets the effects of leakage and the role of windowing techniques to show the frequency domain measurements. Matlab has been used to simulate the different windows.

6. ACKNOWLEDGMENTS

The author would like to thank The Director and Head of Electronics and Communication Engineering Department, National Institute of Technical Teachers' Training & Research, Chandigarh India, for their constant inspiration, support and helpful suggestions, through-out this research work.

7. REFERENCES

[1] Rajesh Mehra, Pooja Kataria, 'FPGA Based Area Efficient 64-point FFT Using MAC Algorithm', International Journal of Electrical, Electronics and Telecommunication Engineering, vol.44, Issue.2 pp.1330-1337, ISSN: 2051-3240, october2013.

[2] Yazan Samir, Rozita Teymourzadeh "The Effect of the Digit Slicing Architecture on the FFT Butterfly" International Conference on Information Science, Signal Processing and Their Applications (ISSPA), pp.802-805, IEEE 2010.

[3] Xin Xiao, Erdal Oruklu and Jafar Saniie "Reduced Memory Architecture for CORDIC-based FFT" IEEE International Conference on Electronic Information Technology, pp.345-350, IEEE 2010.

[4] Yousri Ouerhani, Maher Jridi "Implementation Techniques of High-Order FFT into Low-Cost FPGA" International Midwest Symposium on Circuits and Systems (MWSCAS), pp.1-4, IEEE 2011.

[5] Earl E. Swartzlander, Hani H.M. Saleh "FFT Implementation with Fused Floating-Point Operations" IEEE Transaction On Computers, Volume No.61, Issue No.2, pp.312-317, February 2012.

[6] Muhammad Ahsan, Ehtsham Elahi, Waqas Ahmad Farooqi "Superscalar Power Efficient Fast Fourier Transform FFT Architecture" IEEE Transaction on Computers, Volume No.24, Issue No.31, pp.414-426, June 2011.

[7] Pooja Kataria, Rajesh Mehra, "Comparative Analysis of FFT Algorithm for different window techniques " International Journal of Science, Engineering and Technology Research, volume 2, Issue 9, pp-1691-1695, September 2013

[8] Earl E. Swartzlander, Hani H.M. Saleh "FFT Implementation with Fused Floating-Point Operations" IEEE Transaction On Computers, Volume No.61, Issue No.2, pp.312-317, February 2012.

[9] Mathworks, "Users Guide Filter Design Toolbox 4", 2007.