ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 9, September 2013

Comparative Analysis of FFT Algorithm for Different Window Techniques

Pooja Kataria , Rajesh Mehra

***Abstract*— In this paper authors have discussed about FFT algorithm on periodic waveform using different windows , how window function reduces spectral leakage and how higher order FFT can be realized using lower order FFT has been discussed. The proposed design is focused on trade off between speed and cost as these factors should be taken into consideration when designing signal processing systems on FPGA platforms. 64 point FFT can be realized using 8 point FFT has shown i.e higher order FFT can be realizes using smaller pont FFT. Different windows on periodic waveform has created and simulated using Matlab 2007a.**

***Index Terms*— Fast Fourier Transform(FFT), DFT, FPGA, Orthogonal frequency division multiplexing, Spectral leakage**

I. Introduction

Application using frequency analysis of discrete-time signals in digital signal processor is the most convenient method especially in general-purpose digital computer or specially designed digital hardware. Frequency analysis is performed on a discrete-time signal {x(n)} by converting the time-domain sequence to an equivalent frequency-domain presentation[1]. The essence of the Fourier transform of a waveform is to decompose or separate the waveform into a sum of sinusoids of different frequency. Thus, the pictorial representation of the Fourier transforms is a diagram which displays the amplitude and frequency of each of the determined sinusoids.

DFT is very important technique in modern digital signal processing and telecommunication, especially for applications in orthogonal frequency division multiplexing systems (OFDM). The fast fourier(FFT) is computationally efficient way to implement discrete fourier transform. Discrete fourier transform is computationally intensive and time complexity of O(N^2) where FFT that was proposed by Cooley and Tukey to reduce the complexity to O(Nlog_2N) where N is FFT size. As it is fundamental operation in various popular multicarrier modulation technique, and in high speed communication systems[2]. The Fourier Transform (FFT) identifies or distinguishes the different frequency sinusoids and their respective amplitudes which combine to form an arbitrary waveform. DFT (discrete Fourier Transform) and convolution are two basic and common operations in signal processing.

Pooja kataria, *National Institute Of Technical Teachers, Training and Research, Chandigarh , India , 8968377757. Rajesh Mehra, National Institute Of Technical Teachers, Training and Research.*

In fact, many other algorithms such as filter, spectrum estimation can be transformed into DFT to implement. FFT (Fast Fourier Transform) is the fast algorithm of the DFT (Discrete Fourier Transform [3]. Currently, FFT has been widely used in spectral analysis, matched filtering, digital communications, image processing, speech recognition, radar processing, remote sensing, geological exploration and wireless secure communications and other fields.

Cooley and Tukey proposed Fast Fourier Transform (FFT) to make DFT much easier to implement, thus making DFT can be applied to many related fields. As the faster version of DFT, FFT and its inverse transform IFFT are important analysis methods in digital signal spectrum analysis. But FFT potentially requires multi-cycle processing, and can become a major bottleneck for overall system performance. Therefore enhancing the performance of the FFT component is a key factor in evaluating the overall system performance and it is common to use it as a benchmark for the whole system[4]. The FFT hardware is heavily constrained by the area and power requirement and thus the focus is to minimize these two parameters without sacrificing the performance.

In line with this, designers tend to use pipelined high speed multipliers, but these, in fact, add large area overhead, which in turn lead to more power consumption. For large FFT sizes the major portion of the area for the FFT hardware generally comes from the storage/memory elements for the twiddle factor tables and pipeline registers. The computational units such as complex multipliers and the complex adders also contribute to the area of the FFT significantly [5]. Therefore to reduce the power consumption, comes the need to design the FFT which is not only area efficient but should also operate with an acceptable speed. In recent years, with the rapid development of FPGA (Field Programmable Gate Array) technology, FPGA can perform parallel signal processing, implement pipeline structure and easy to upgrade. So FPGA is very suitable for the realization of FFT algorithm.

Also Field Programmable Gate Arrays (FPGAs) have great potential to substantially accelerate computational intensive algorithms such as FFTs. In conjunction with the enormous logic capacity allowed by today's technologies, makes FPGAs an attractive choice for implementation of complex digital systems. Moreover, due to inclusion of digital signal processing capabilities [5], FPGAs are now expanding their traditional prototyping roles to help offload computationally intensive digital signal processing functions from the processor.

II. Fast Fourier Transform

Method used to transform time domain to frequency domain and reverse for implementation on digital hardware is known as discrete fourier transform. For N point DFT of a complex data sequence x(n) is defined as

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

Where x(n) and X(K) represent time domain and frequency components and $W_N^{kn}=e^{-j2\pi/N}$ is the twiddle factor. The discrete fourier transform of the above equation will represent harmonically related frequency component of x(n). If we go for direct computation for the above equation it will require order of N^2 operations where N is transform size[6]. Whereas in 1965 Cooley and Tukey have found new technique to reduce the computational complexity to Nlog_2N. various FFT algorithms have been developed such as radix 2, radix-4 and split radix algorithm. They have constant butterfly structure.
FFT algorithm relies on divide and conquer methodology dividing N coefficients points into smaller blocks of different sizes. Let us consisder N=M.T, N=DFT length, k=s+T.t and n=l+M.n, where M and N are integers and S, L ∈ {0,1…….M-1} and t,m ∈ $\{0,1 \ldots \ldots T-1\}$ applying these consederations in equation (1) we will obtain

$$X(s+T.t)=\sum_{l=0}^{M-1}\sum_{m=0}^{T-1}[x(l+M.m)W_{MT}^{l+Mm)(s+Tt)} \qquad (2)$$

Finally we can write the equation as the following

$$X(s+T.t)=\sum_{l=0}^{M-1} W_M^{lt} \sum_{m=0}^{T-1}[W_{MT}^{ls}\, x(l+Mm) W_T^{ms}] \qquad (3)$$

It is clear from the above equation that n order to realize N point FFT , it is possible to first decompose it into one M point and one T point FFT where N=M.T and then combining them finally. We can take the example of 64 point to perform 64 point FFT we can go for M=T=8. Then using equation (3)

$$X(s+8t)=\sum_{l=0}^{7} W_8^{lt} \sum_{m=0}^{7}[W_{64}^{ls}\, x(l+8m) W_8^{ms}] \qquad (4)$$

Equation (4) represents two dimensional structure of 8 point FFT representing 64 point FFT. Hence performance of 64 point FFT will now depend upon 8 point performance. Here we have take split radix DIT architecture because of its lower numbers of arithmetic operations.

III. SPATIAL DISTRIBUTION FFT ARCHITECTURE

There are different possible ways by which 64 point FFT can be realized and among various methods one of the realization is represented in signal flow graph of fig.1, here it is clear from the figure that 64 point FFT is composed on five levels[7]. The first level is consists of two serial to parallel blocks used to store real and imaginary part of data presented in a serial way. The second block is composed of eight blocks of 8 point FFT split radix DIT. The third block contains 49 complex multiplier used to compute non trivial complex multiplication. The fourth is similar to second one .In the last we have two parallel to serial locks giving finally data in a serial way. After 5 clock cycles the 8 point FFT outputs are available and multiplication can be started. Block multiplier needs 2 clock cycles to perform 49 complex multiplications[8]. The 64 point FFT outputs are available 5 clock cycle after the last stage of 8 point FFT transformation. The main advantage of this architecture is high speed and low latency, but on the other hand it requires high memory, higher number of complex multipliers and adders making it unsuitable for low cost FPGAs such as Sparten 3 family , whereas Virtex FPGAs comes under high cost FPGAs.

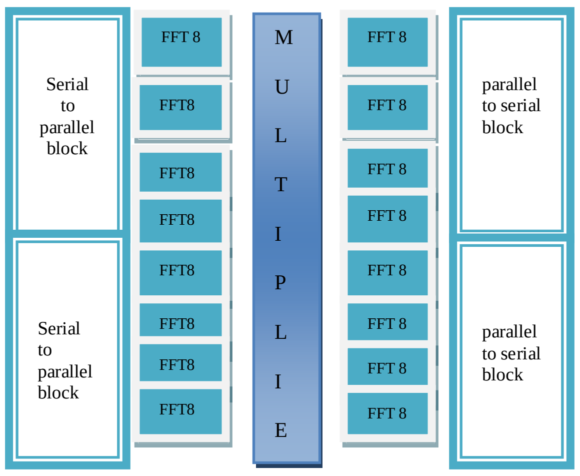

Fig.1 Spatial Distribution FFT Architecture

IV. WINDOW BASED FFT ANALYSIS

Windowing is a technique used to shape the time portion of your measurement data, to minimize edge effects that results in spectral leakage in the FFT spectrum. By using window functions correctly, the spectral resolution of your frequency-domain result will increase. The FFT implements a fourier transform at a discrete set of frequency and from a time domain waveform sampled at discrete times over a finite interval of time because if finite interval, the FFT tends not to be very frequency selective. When we use the FFT to measure the frequency content of data, we will have to base the analysis on a finite set of data.
The FFT transform assume that finite data set is one period of a periodic signal. For the FFT both time domain and frequency domain are circular topologies so the two endpoints of the time waveform are interpreted as though they were connected together[9]. Therefore the finiteness of the sampling record may result in truncated waveform with different spectral characteristics from the original continuous time signal and the finiteness can introduce sharp transition changes into measured data. The sharp transitions are discontinuities. To minimize this effect we can apply a window function to the measured signal in time domain. This will make endpoints meet of the waveform and therefore results in continuous waveform without sharp transitions.

When the signal is periodic and an integer numbers of periods FFT turns out fine and when the number of periods in the acquisition is not an integer, the endpoints are discontinuous the results is the high side lobes seen in the window spectrum plot known as spectral leakage. Leakage results in the signal energy smearing out over a wide frequency range in the FFT when it should be in a narrow frequency range. To correct this problem appropriate windowing functions must be applied. We must choose the appropriate window function for the specific application. When windowing is not applied correctly, then errors may be introduced in the FFT amplitude, frequency or overall shape of the spectrum.

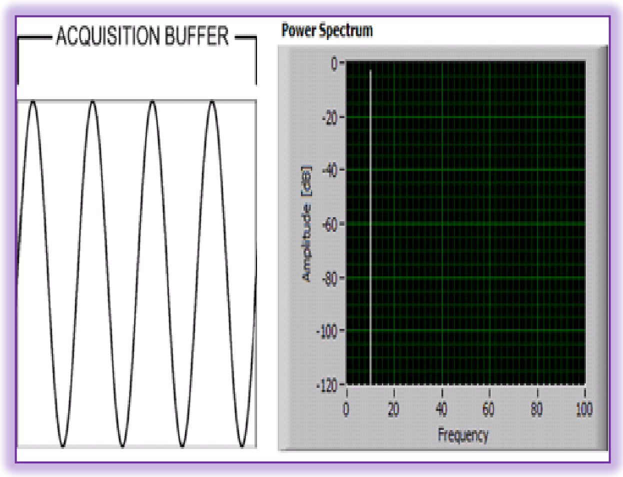

Fig. 2. Measuring an integer number of periods gives ideal FFT

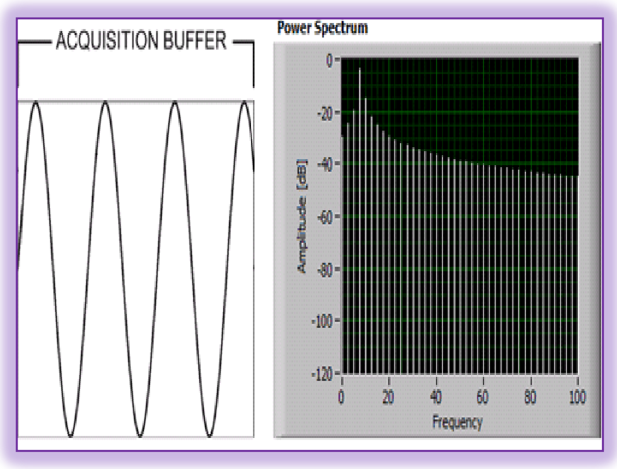

Fig. 3. Measuring a non-integer number of periods adds spectral leakage to the FFT

From the Figure 2 and 3 we can see the difference between the integer and non integer numbers of periods, how the non-integer adds to the spectral leakage.Since most signals are not periodic in the predefined data block time periods, a window must be applied to correct for leakage. A window is shaped so that it is exactly zero at the beginning and end of the data block and has some special shape in between. This function is then multiplied with the time data block forcing the signal to be periodic. A special weighting factor must also be applied so that the correct FFT signal amplitude level is recovered after the windowing. Figure 4 and 5 shows the effect of applying a Hanning weighting and Triangular weigh function in the time domain and its resultant frequency domain plot.

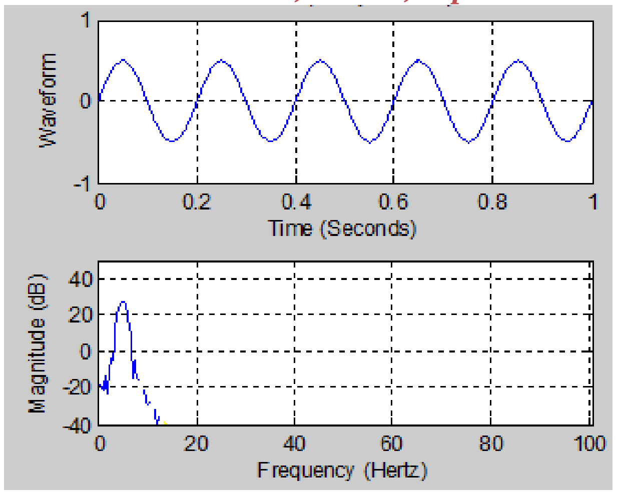

Fig.4 FFT using Hanning Window

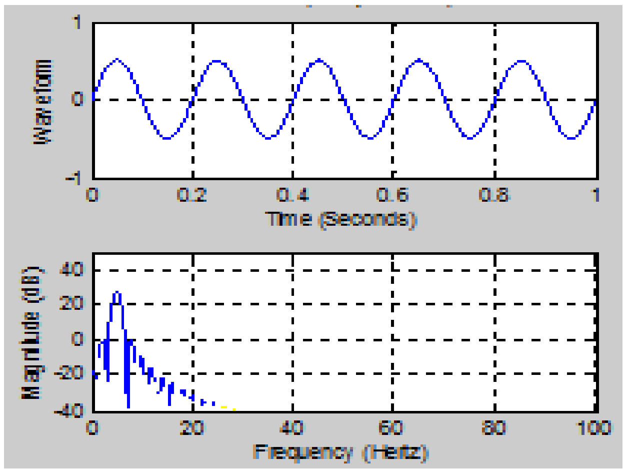

Fig.5 FFT using Triangular Window

The hanning window forces the amplitude of time record to zero at both the beginning and at the end of sample interval. But by this it adds distortion to the wave form being analyzed in the form of amplitude modulation; i.e., the variation in amplitude of the signal over the time record. Amplitude Modulation in a wave form results in sidebands in its spectrum, and in the case of the Hanning window, these side lobes, effectively reduce the frequency resolution of the analyzer. The Hanning window should , therefore, always be used with continuous signals, but must never be used with transients. The reason is that the window shape will distort the shape of the transient, and the frequency and phase content of a transient is intimately connected with its shape.

A better choice when working with transients events such as hammer excitation is the rectangular window, a rectangular window is actually not a window at all which makes sense for this type of signal as the transient event typically starts and ends at zero amplitude within the finite word length of the record. Applying a rectangular window is equivalent to not using any window because the rectangular function just truncates the signal to within a finite time interval. Because of the high side lobes, using the FFT with a rectangular window function (or no window function) is normally not recommended. Fig.6 and 7 shows the effect of rectangular and Kaiser window on sine waveform. Frequencies from 0 to 100 Hz are displayed , the sampling rate is 200 Hz which implies that nyquist frequency is 100 Hz.

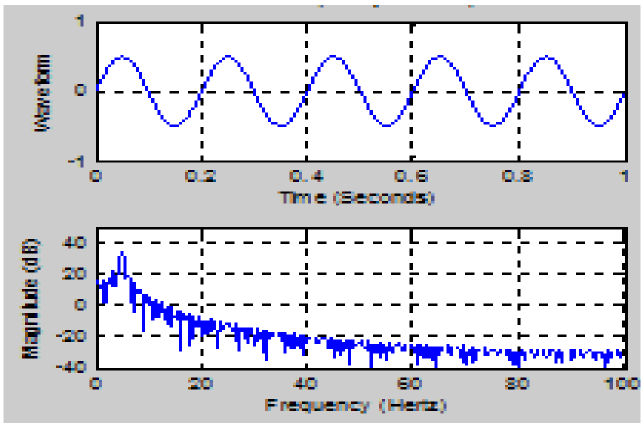

Fig.6 FFT Transform using Rectangle Window

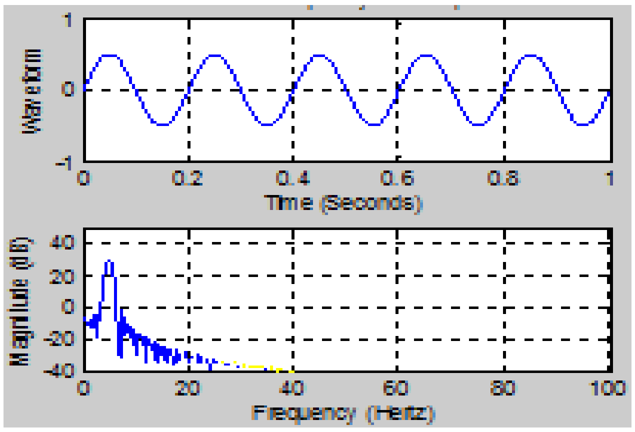

Fig.7 FFT Transform using Kaiser Window

It is clear from the fig.6 that for the rectangle window the frequency graph extends to very wide range and it will give information about each and every frequency component of the signal. Also no amplitude corrections are necessary with the rectangular window, since it weights the whole event uniformly, thus ensuring good fidelity. Whereas this is not in case of Kaiser window , as can be seen from fig.7, it gives information only for that frequency components which are near to peak lobe and not giving other frequency components details.

But Kaiser window allows control of the trade-off between the main-lobe width and the highest side-lobe level. If we want less main-lobe width we will get higher side-lobe level and vice versa. Since control of this trade-off is valuable, the Kaiser window is a good general-purpose choice. Basically the Kaiser-Bessel window is a flexible smoothing window whose shape you can modify by adjusting the beta parameter. Thus, depending on the application, we can change the shape of the window to control the amount of spectral leakage. The Kaiser-Bessel window is useful for detecting two signals of almost the same frequency but with significantly different amplitudes

Figure 8 and 9 shows the effect of applying a Hamming weighting and chebyshev weigh function in the time domain and its resultant frequency domain plot. The Hamming window is a modified version of the Hanning window. The shape of the Hamming window is similar to that of a cosine wave. The Hanning and Hamming windows are similar, as can be seen fig.4 and 8. However the Hamming window does not get as close to zero near the edges as does the Hanning window. Dolph-Chebyshev window that has the narrowest main lobe width for a given maximum side lobe depth.

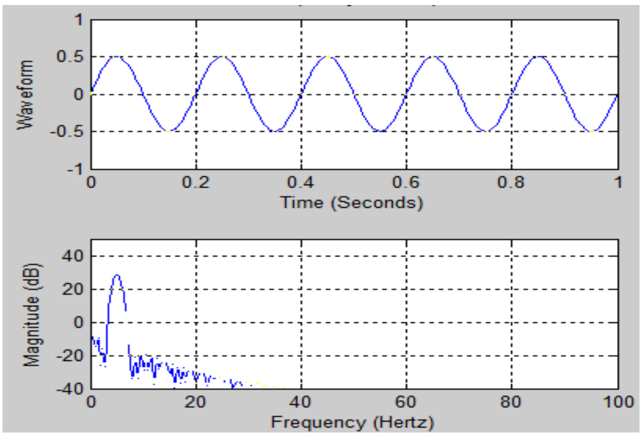

Fig.8 FFT Transform using Hamming Window

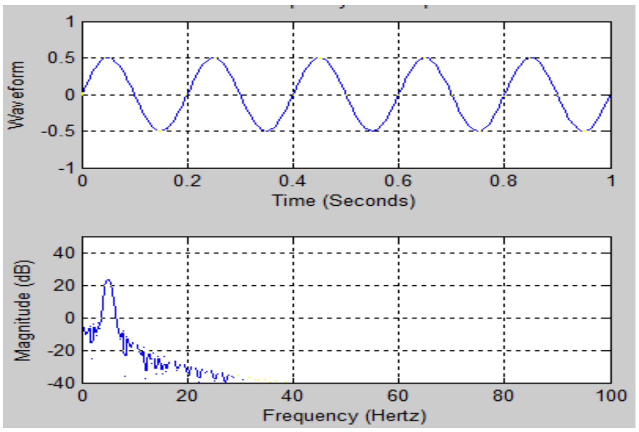

Fig.9 FFT Transform using Chebyshev Window

Hence we can say that when the FFT is used, attention should be paid to leakage, which is caused by the FFT's assumption that the input signal repeats periodically and that the periodic length is equal to the length of the actual input. However, if the true signal is not periodic or if the assumed periodic length is not correct, leakage will occur. This will cause both the amplitude and position of a frequency measurement to be inaccurate. Origin supports the use of window functions to mitigate leakage.

V. CONCLUSION

This paper has presented the FFT algorithm on periodic waveform using different windows , how using different windows spectral leakage can be reduced, effect of integer and non integer period of waveform. Several window functions are supported, including Triangular, Hanning, rectangular, hamming , chebyshev and kaiser, each of which has its own unique advantages and disadvantages. A specific window function should be selected according to the kind of signals being analyzed. Also we must understand the effects of leakage and know the tradeoffs and advantages of the various windowing functions to accurately interpret frequency domain measurements. How higher order FFT can be realized using lower order FFT has also been discussed. 64 point FFT has taken as an example and how it can be designed using 8 point FFT.

REFERENCES

[1] 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.

[2] 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.

[3] Shakeel S. Abdulla, Haewoon Nam, Mark McDermot "A High Throughput FFT Processor With No Multipliers" International Conference on Computer Design(ICCD), pp.485-495, IEEE 2009.

[4] Mao-Hsu Yen, Pao-Ann Hsiung, Sao-Jie Chen " A Low Power 64-Point Pipeline FFT/IFFT Processor" IEEE Transactions on Consumer Electronics, Volume No. 57, Issue No. 1, pp. 40-45, February 2011.

[5] Ahmad A. Al Sallab, Dr. Hossam Fahmy, Prof. Dr. Mohsen Rashwan "Optimized Hardware Implementation Of FFT Processor" International Conference on Circuits and Systems, pp.208-213, IEEE 2009.

[6] 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.

[7] Zou Wen, Qiu Zhongpan, Song Zhijun "FPGA Implementation of efficient FFT algorithm based on complex sequence" International Conference on Electronic Computers, pp.614-617, IEEE 2010.

[8] 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.

[9] 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.

[10] S. Winograd "On computing the discrete Fourier transform" Math Computation, Volume No. 32, Issue No.1, pp. 175–199, January 2008.