International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 6, No. 3, November 2017, pp. 186~190
ISSN: 2089-4864, DOI: 10.11591/ijres.v6.i3.pp186-190

Optimization of Resource Utilization of Fast Fourier Transform

Subhash Chandra Yadav[1], Pradeep Juneja[2], R. G. Varshney[3]
[1,2]School of Electronics, Graphic Era University, Dehradun, India
[3]School of Applied Sciences, Graphic Era University, Dehradun, India

Article Info

Article history:

Received Aug 02, 2017
Revised Oct 03, 2017
Accepted Oct 17, 2017

Keywords:

Decimation-In-Frequency (DIF)
Decimation-In-Time (DIT)
Discrete Fourier Transform (DFT)
Fast Fourier Transform (FFT)
Field Programmable Gate Array (FPGA)
Very Large Scale Integrated Circuit Hardware Description Language (VHDL)

ABSTRACT

This paper considers the optimization of resource utilization for three FFT algorithms, as it pertains not to the input samples or output modes, but to the twiddle factors that arise in Cooley-Tukey FFT algorithms. Twiddle factors are a set of complex roots of unity, fixed by the transform order for the particular algorithm. This paper shows the comparison between three known FFT algorithms, DIT-FFT, DIF-FFT and GT algorithm. All these algorithms are implemented on FPGA (Spartan-3 XC3S4000l-4fg900) with XILINX 10.1 ISE.

Corresponding Author:

Subhash Chandra Yadav,
School of Electronics, Graphic Era University
566/6, Bell Road, Clement Town, Dehradun 248002, India.
Email: subhash.yadav775@gmail.com

1. INTRODUCTION

There are mainly two reasons to convert a time domain signal into frequency domain signal. First, to decompose a complex signal into simpler parts to facilitate analysis and secondly differential equation, difference equations and convolution operations in the time domain become algebraic operations in the frequency domain. The Fourier series used to convert continuous time and discrete time periodic sequence into frequency domain. The basic representation of periodic signal is the Fourier series, which is a linear weighted sum of related sinusoids or complex exponential [1]. The Fourier transform is a way to decompose a signal into its constituent frequencies and versions it is applied to the aperiodic continues time and discrete time domain signal [2]. DFT is a finite duration discrete frequency sequence which is obtained by sampling one period of Fourier Transform .Sampling is done at 'N' equally spaced points over the period extending from ω_0=0 to ω_0=2π.

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

Where $W_N = e^{-(j2\pi)/N}$ and k=0..1..N-1, W_N is called twiddle Factor, it makes the computation of DFT more easy and fast. The twiddle factor (W_N), describes a "rotating vector", which rotates in increments according to the number of samples N. As 180 degrees out of phase are the negative of each other. So for example, W for N=4 samples, where n=0, 4, 8, etc, are the negative of n=2, 6, 10, etc.

The Butterfly diagram takes advantage of this redundancy and symmetry, which is part of what makes the FFT possible. An FFT is a way to compute the same result more quickly computing a DFT of N points in the naive way, using the definition, takes O(N^2) arithmetical operations, while an FFT can compute the same result in only O(N log N) operations. The difference in speed can be substantial, especially for long data sets where N may be in the thousands or millions—in practice, the computation time can be reduced by several order of magnitude in such cases, and the improvement is roughly proportional to N / log (N). This huge improvement made many DFT-based algorithms practical. The FFT is a fast algorithm to find out the DFT of a sequence, a direct computation of DFT of a sequence required (N^2) complex multiplication and($N^2 - N$) complex addition and if we compute DFT of a sequence by FFT algorithm there required ($\frac{N}{2}\log_2 N$) Complex Multiplication and ($N\log_2 N$) complex addition. In the case of DFT there are N^2 complex multiplication and $N^2 - N$ complex addition are required for N point DFT. But in the case of FFT there are $\frac{N}{2}\, log_2 N$ multiplication and $N\log_2 N$ addition are required.

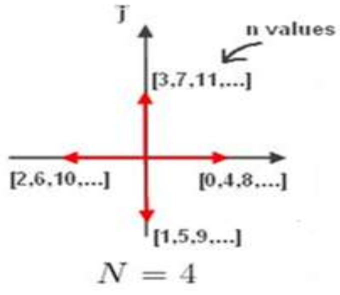


Figure 1. Twiddle factor as a rotating vector

2. DECIMATION IN TIME FFT ALGORITHM

Decimation in time FFT algorithm imply first dividing the input sequence in time domain [3] and then processed for the FFT computation. Let b $x(n)$ e a sequence which we want to convert into frequency domain and $f_1(m)$ and $f_2(m)$ are the even and odd part respectively then the DFT of the sequence $x(n)$ is given by Equation 1 as :

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

The combination of equation (3) ,(4), (5) and (6) gives the flow graph of DIT-FFT as shown in Figure-2(a). By the property of twiddle factor $W_N^2 = W_{N/2}$ and the definition of DFT we can write :

$$X(k)=F_1(k) + W_N^k\, F_2(k) \tag{3}$$

Here $F_1(k)$ is $N/2$ point DFT of $f_1(m)$ and $F_2(k)$ is $N/2$ point DFT of $f_2(m)$.
As N/2 is the period of the sequence $f_1(m)$ and $f_2(m)$ so replacing k by (k+ $\frac{N}{2}$) in Equation 3

$$X(k+\tfrac{N}{2})=F_1(\mathrm{k})-W_N^k\, F_2(k) \tag{4}$$

By the periodic property of DFT and twiddle factor. Again decimating the sequence $F_1(k)$ and $F_2(k)$ we have

$$G_{11}(0)= X(0) + X(2) \tag{5}$$

$$G_{11}(1)= X(0) - X(2) \tag{6}$$

3. DECIMATION IN FREQUENCY FFT ALGORITHM

Decimation in frequency FFT algorithm indicates first computatation of FFT and then dividing the output sequence in even and odd part which is in frequency domain [4]. The flow graph is shown in Figure 2. We can devide equation (1) into two parts as follows:

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{kn}+\sum_{n=N/2}^{N-1}x(n)W_N^{kn} \tag{7}$$

Put n=n+N/2 ,limit will change as
When n=N/2 ⇨ N/2=n +N/2
therefore n=0
When n=N-1 ⇨ N-1=n +N/2
therefore n=N–1–N/2=N/2-1
Putting these values in second summation of above eq. we get :

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{kn}+\sum_{n=0}^{\frac{N}{2}-1}x(n+\frac{N}{2})W_N^{k(n+N/2)}$$

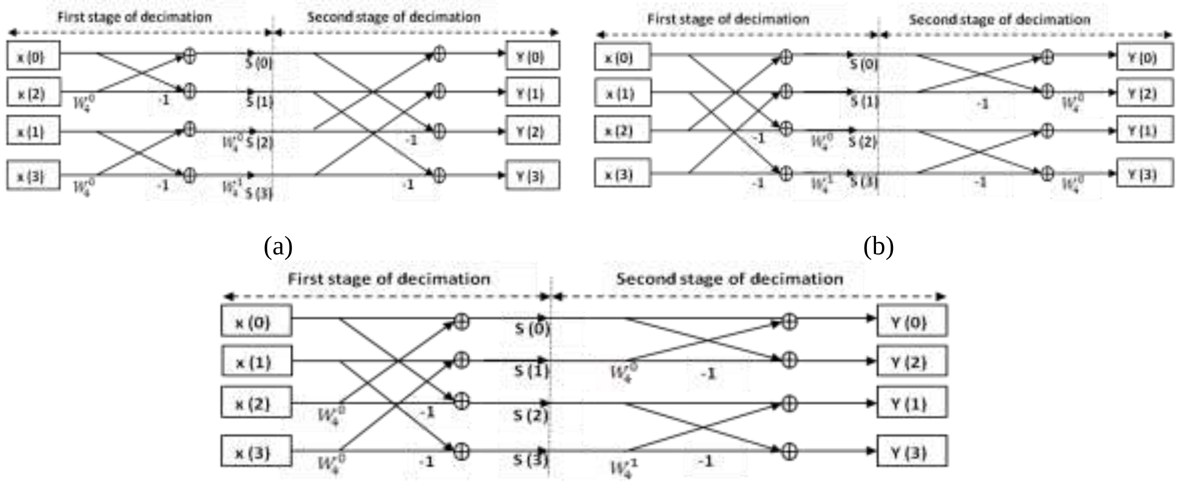
Which can be reduced to

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+(-1)^{2r}$$

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+(-1)^{k}\ x(n+\frac{N}{2})]W_N^{kn} \tag{8}$$

Now if we decimate the equation (8) into even and odd parts then we have

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+(-1)^{2r}\ x(n+\frac{N}{2})]W_N^{2rn} \tag{9}$$



(a) (b)

(c)

Figure 2. (a)flow graph of DIT-FFT(b)flow graph of DIF-FFT(c)flow graph of G-transpose FFT

By putting k=2r+1 in eq. (4.5.16) we will get odd number of sequence

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+(-1)^{2r+1}\ x(n+\frac{N}{2})]W_N^{(2r+1)n} \tag{10}$$

Where r is an integer which varies from 0 to $\frac{N}{2}-1$

As we know (-1)2r=1 and (-1)2r+1=-1
Putting these values in Equation 9 and 10,

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1}[x(n) + x(n+\frac{N}{2})]W_N^{2rn} \tag{11}$$

$$X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1}[x(n) - x(n+\frac{N}{2})]W_{\frac{N}{2}}^{rn}W_N^{n} \tag{12}$$

From Equation 11 and 12

$$g(n)=x(n) + x\left(n+\frac{N}{2}\right) \tag{13}$$

$$h(n)=[\ x(n) - \mathrm{x}\left(\mathrm{n}+\frac{\mathrm{N}}{2}\right)]W_N{}^n \tag{14}$$

Again decimating the sequence $g(n)$ and $h(n)$ we have:

$$\mathrm{X(k)=g(0) + g(1)} \tag{15}$$
$$X(k)=g(0) - g(1)$$

4. G -TRANSPOSE ALGORITHM

G-Transpose algorithm decreases the number of twiddle accesses by an asymptotic factor of log (n) [5]. Idea behind the G-transpose algorithm is to decrease the number of twiddle table accesses by restructuring the DFT. The G-Transpose network applies the twiddle factors to each sub network's inputs. In fact, permuting the G-Transpose network's rows into bit-reversed order while leaving connectivity unchanged (that is, redrawing the G-Transpose network such that the inputs given in the order x (0), x(2), x(1), and x(3)) yields the DIT network.

Thus, a G-Transpose implementation produces the same results in finite precision arithmetic as a DIT implementation but processes input/output data like a DIF implementation. Figure 2(c) shows the flow graph of G^T algorithm. G^T algorithm give an asymptotic reduction in the number of twiddle factor loads required for first stage of decimation which is responsible for minimizing memory size as compare to Cooley-Tukey decimation-in-time and decimation-in-frequency FFT.

5. SIMULATION RESULTS AND DISCUSSION

We investigated G-Transpose, Cooley-Tukey Decimation-in-Time and Decimation-in-Frequency FFT algorithms, We have taken following parameter for simulation of all algorithms Radix=2, N=4 (4 point FFT) Input sequence x(n)={2, 2, 4, 0} Twiddle Factor W_4^0=1 and W_4^1=- j. The result of simulation in MATLAB for all the three algotithm are same and given as {8, -2-2j, 4, -2+2j} as shown in Figure 3, for memory optimization simulation of all Three algorithm is successfully developed using VHDL on XILINX Spartan-3 XC3S4000l-4fg900 FPGA development board. The simulation result for all the three algorithms are same which means the result of FFT is not affected but the resource utilization of all three algorithms is different. Table 1 shows the resource utilization for all three algorithms on FPGA. In the case of G-Transpose algorithm there is a decrease in number of resource utilization. It is because the number of twiddle factor loads required for the first stage of decimation is less in G-transpose as compare to DIT-FFT and DIF-FFT.

Table 1. Simulation Results for Three Algorithms in XILINX Spartan-3 XC3S4000l-4fg900 FPGA

Resource Utilization	Algorithm	Used
Occupied Slices	G-Transpose	304
	DIT-FFT	352
	DIF-FFT	352
Total Number of 4 input LUTs	G-Transpose	608
	DIT-FFT	704
	DIF-FFT	704
Number used as a route-thru	G-Transpose	3
	DIT-FFT	6
	DIF-FFT	6

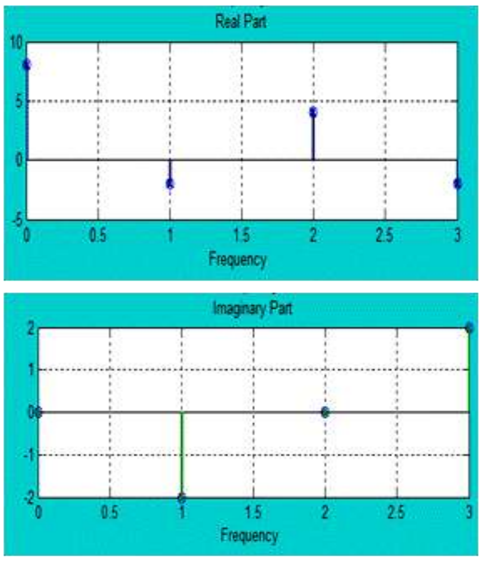
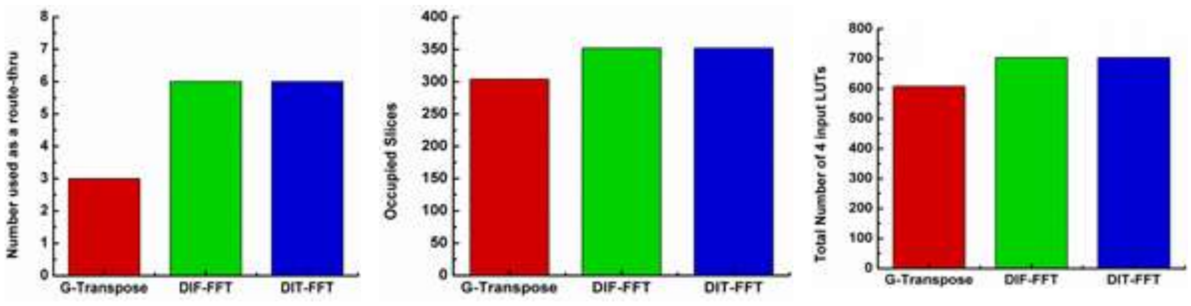


Figure 3. Simulation result of G^T, DIT-FFT, DIF-FFT algorithm (all the results are same)

The graph below shows the resource utilization of FPGA for three algorithms



(a) (b) (c)

Figure 4. Device utilization comparison for three algorithm (a) Number used as route thru,(b) Occupied slices (c) Total number of 4 input LUTs

6. CONCLUSION

We investigate three algorithms on FPGA using VHDL and observed that the number of complex addition and multiplication are same for all the three algorithm but the resource utilization of the FPGA for G-Transpose algorithm is less as compare to DIT-FFT and DIF-FFT which leads to reduce the memory.

REFERENCES

[1] Vretblad, Anders, Fourier Analysis and Its Applications.Springer,2003
[2] Proakis, J. G. " Digital Signal Processing, Principles, algorithms and applications" Prentice Hall, Inc., 1996.
[3] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301,1965.
[4] S. Salivahnan, A. vallavraj, "*Digital Signal Processing"* Tata McGraw-Hill 2000.
[5] Kevin J. Bowers, Ross A. Lippert, Ron O. Dror and David E. Shaw "*Improved Twiddle Access for Fast Fourier Transforms*" IEEE transactions on signal processing, vol. 58, no. 3,pp. 1122-1130, March 2010.