LNMIIT
The LNM Institute of Information Technology
The LNM Institute of Information Technology
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Experiment No. # 08
Radix-2 FFT Algorithm and complexity reduction

---

1) **Aim:**
   a) Radix-2 FFT Algorithm.

2) **Software used:**
   a) MATLAB.

3) **Theory**
   a) J. Proakis and D. Manolakis, Digital signal processing: principles, algorithms, and applications, ser. Prentice-Hall International editions. Prentice Hall, 1996. [Online]. Available:http://books.google.co.in/books?id=sAcfAQAAIAAJ

4) **Procedure FFT Algorithm:Desimation in time**
   The N-point Discrete fourier transform(DFT) calculation generally requires N^2 complex multiplications.(Think of N = 1024). On the other hand Radix-2 FFT algorithm: Desimation in time(DIT) or Decimation in frequency(DIF) are computationally efficient and requires only $\frac{N}{2} \cdot log_2 N$ complex multiplications.
   a) Let's consider a sequence $x[n]$ of length N and calculate it's N point DFT.(For Simplicity $N = 2^V$, V is an integer)

   b) The N length sequence can be divided into two $N/2$ point data sequence $f1[n]$ and $f2[n]$, corresponding to the even numbered and odd numbered samples of $x[n]$ as shown in next step.

   c) The DFT of $x[n]$ is given by
   $$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn} \tag{1}$$
   where k = 0,1,2, ... $\frac{N}{2}$-1 and $W_N = exp(-j * 2 * pi/N)$.

   d) Breaking $x[n]$ into even and odd sequence
   $$X[k] = \sum_{n=even} x[n] \cdot W_N^{kn} + \sum_{n=odd} x[n] \cdot W_N^{kn}$$
   $$X[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m] \cdot W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] \cdot W_N^{(2m+1)k}$$

e) This leads to

$$X[k] = \sum_{m=0}^{\frac{N}{2}-1} f1[m] \cdot W_{N/2}^{mk} + \sum_{m=0}^{\frac{N}{2}-1} f2[m] \cdot W_{N/2}^{mk}$$

since $W_N^2 = W_{N/2}$.

$$X[k] = F_1[k] + F_2[k] \cdot W_N^k \tag{2}$$

for k = 0,1,2....N-1, where $F_1[k]$ and $F_2[k]$ are $N/2$ point DFT sequence of $f_1[m]$ and $f_2[m]$

f) Also by using periodicity of $F_1[k]$ and $F_2[k]$ (means $F_i[k + \frac{N}{2}] = F_i[k]$ for $i = 1, 2$), the above equation reduces to

$$X[k] = F_1[k] + F_2[k] \cdot W_N^k \tag{3}$$

$$X[k] = F_1[k] - F_2[k] \cdot W_N^k \tag{4}$$

for k = 0,1,2,...$\frac{N}{2} - 1$.

g) The above expression shows that direct computation of $F_1[k]$ requires $\frac{N}{2}^2$ complex multiplications, same for $F_2(k)$. Additional $\frac{N}{2}$ for $F_2[k] \cdot W_N^k$. Thus calculating N point DFT of x[n] using above method total requires $Total = (\frac{N}{2})^2 + (\frac{N}{2})^2 + \frac{N}{2}$ complex multiplication.

h) At the same time, Note that the direct calculation of N point DFT of $x[n]$ as per eq.(1) requires total N^2 complex multiplications. So reduction in $Total = N^2 - ((\frac{N}{2})^2 + (\frac{N}{2})^2 + \frac{N}{2})$

i) Again, split $f_1[n]$ and $f_2[n]$ sequence into even and odd sequence and do all the above steps again.

j) Re-do all the above steps untill the final sequence has only one point sequence. At this point the total complex multiplications requeired for N point DFT using Direct method is N^2 and using above stpes is $\frac{N}{2} \cdot log_2 N$.

k) For refrence, an N=8 point Radix-2 FFT algorithm is given in Fig.1. on next page.

5) **Observation:**

a) Simulate N = 2 point fft using above method, and verify the answer with inbuilt fft() function for input sequence x[n] of length 2 or more.

b) Simulate N = 4 point fft using above method, and verify the answer with inbuilt fft() function for input sequence x[n] of length 4 or more.

c) Simulate N = 8 point fft using above method, and verify the answer with inbuilt fft() function for any arbitrary N length sequence x[n].

d) Plot speed factor Vs N, for various N = 2,4,8,16,128 values. Speed factor is given by

$$\text{Speed factor} = \frac{\text{Complex multiplications in Direct-method}}{\text{Complex multiplications in Radix-2 FFT method}}$$

e) Repeat the experiment in simulink.

6) **Conclusion:** Conclude the experiment.

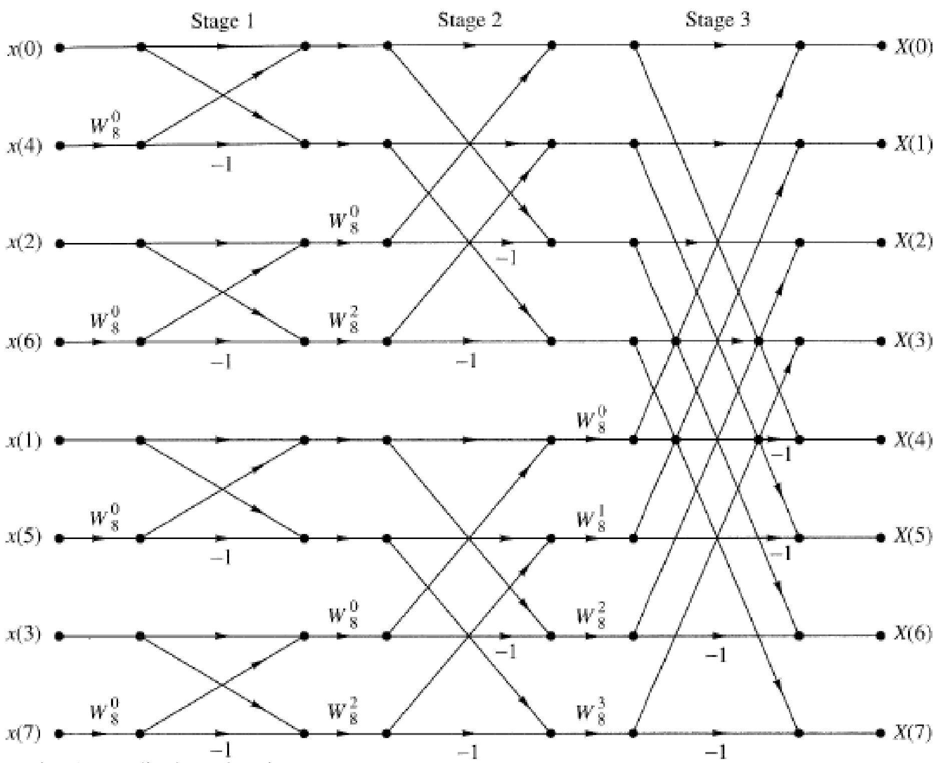

Fig. 1. Radix-2 N=8-point FFT

Well Done