Lecture - 13

1.3.1 Comparison of Computational Complexity with Direct Computation :

First we will calculate the computational complexity for direct DFT calculation.

A] For direct computation :

According to the definition of DFT we have,

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

Equation (1) indicates that we have to take multiplication of x (n) and twiddle factor. Then we have to add all the terms. Since twiddle factor is complex; we need to perform complex multiplications and complex additions.

Complex multiplications :

As given by Equation (1), for one value of 'k' multiplication should be performed for all values of 'n'. The range of 'n' is from 0 to N – 1. So for one value of 'k'; N complex multiplications are required. Now the range of k is also from k = 0 to k = N – 1. The total complex multiplications are,

$$\text{Complex multiplications} = N \times N = N^2 \qquad ..(2)$$

Complex additions :

According to Equation (1), for each value of K we need to add the product terms of $x(n) W_N^{kn}$. For example, let us say N = 4.

$$\text{For } k = 0 \Rightarrow X(0) = \sum_{n=0}^{3} x(n) W_4^{0 \times n} = \sum_{n=0}^{3} x(n) W_4^0$$

$$\therefore \quad X(0) = x(0) W_4^0 + x(1) W_4^0 + x(2) W_4^0 + x(3) W_4^0 \qquad ...(3)$$

In Equation (3); four complex multiplications are required and three complex additions are required. Here we have considered N = 4. Thus for each value of 'k'; N complex multiplications are required and 'N – 1' complex additions are required. Now the total values of k and 'N'.

$$\therefore \text{ Complex Additions} = N(N-1) = N^2 - N$$

B] Computational complexity using FFT algorithm :

Firstly we will calculate the computation complexity required for one butterfly. Consider the general structure of butterfly as shown in Fig. G-14.

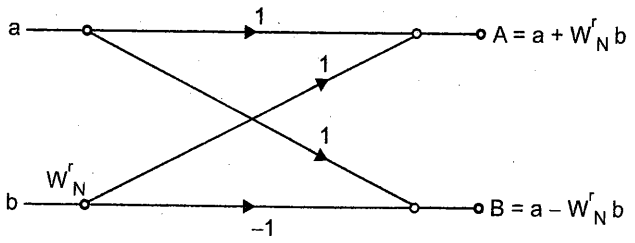

Fig. G-14 : General structure of butterfly

Here 'a' and 'b' are inputs and A and B are outputs of butterfly. The outputs are given by,

$$A = a + W_N^r b \qquad ...(4)$$

$$\text{and} \quad B = a - W_N^r b \qquad ...(5)$$

(1) To calculate any output (A or B), we need to multiple input 'b' by twiddle factor W_N^r. So one complex multiplication is required for one butterfly.

(2) To calculate output 'A', one complex addition is required, while to calculate output 'B' one complex subtraction is required as given by Equation (5). But the computational complexity of addition and subtraction is same. So we can say that for one butterfly two complex additions are required.

(3) As shown in Fig. G-10, for 8 point DFT; 4 butterflies are there at each stage. So for 'N' point DFT, at each stage, $\frac{\text{'N'}}{2}$ butterflies are required.

(4) As shown in Fig. G-10; three stages are required to compute 8-point DFT. In general, for 'N' point DFT, $\log_2 N$ stages are required.

Complex multiplications :

At each stage there are $\frac{N}{2}$ butterflies. Total number of stages are $\log_2 N$. And for each butterfly, one complex multiplication is required.

$$\therefore \text{ Total complex multiplications} = \frac{N}{2} \log_2 N \qquad ...(6)$$

Complex additions :

Total number of stages are $\log_2 N$. At each stage, $\frac{N}{2}$ butterflies are required. And for each butterfly, '2' complex additions are required.

$$\therefore \quad \text{Total complex additions} = 2 \times \frac{N}{2} \log_2 N$$

$$\therefore \text{ Total complex additions} = N \log_2 N$$

Table G-1 shows comparison of direct DFT computation and computation using FFT algorithms.

Table G-1

Number of points 'N'	Direct Computation		Using FFT	
	Complex Multiplications (N^2)	Complex Additions ($N^2 - N$)	Complex Multiplication $\left(\frac{N}{2} \log_2 N\right)$	Complex Addition ($N \log_2 N$)
4	16	12	4	8
8	64	56	12	24
16	256	240	32	64
32	1024	992	40	80
64	4096	4032	96	192
128	16,384	16,256	224	448

The Table G-1 shows that, by the use of FFT algorithms the number of complex multiplications and complex additions are reduced. So there is tremendous improvement in the speed.

1.3.2 In-place Computation to Reduce Memory Size :

Firstly, we will discuss the memory requirement of each butterfly. As shown in Fig. G-11(a); a butterfly calculates the values of 'A' and 'B' for the inputs 'a' and 'b'. Remember that 'a' and 'b' are complex inputs. So two memory locations are required to store any one of the inputs; 'a' or 'b'. One memory location is required to store real part and other memory location is required to store imaginary part. Now, to store both inputs 'a' and 'b'; 2 + 2 = 4 memory locations are required.

Now the outputs are computed as follows :

$$A = a + W_N^r b \quad ...(1)$$

$$\text{and} \quad B = a - W_N^r b \quad ...(2)$$

Thus outputs 'A' and 'B' are calculated by using the values of 'a' and 'b' stored in the memory. Now 'A' and 'B' are also complex numbers; so 2 + 2 = 4 memory locations are required to store both the outputs A and B.

Once the computation of 'A' and 'B' is done then, values of 'a' and 'b' are not required. So instead of storing 'A' and 'B' at other memory locations; these values are stored at the same place where 'a' and 'b' were stored. That means 'A' and 'B' are stored in the place of 'a' and 'b'. This is called as in-place computation. In place computation reduces the memory size.

Memory requirement :

We discuss that 'four' memory locations are required for every butterfly to store input and output values. Now, there are $\frac{N}{2}$ butterflies per stage. Thus for each stage,

Memory locations required to store inputs and outputs $= 4 \times \frac{N}{2} = 2N$

Now as shown in Fig. G-14, one value of twiddle factor is required to compute A and B. To store one value of twiddle factor, one memory location is required for each butterfly. Now there are $\frac{N}{2}$ butterflies at each stage. Thus for each stage,

$$\text{Memory locations required to store twiddle factor} = \frac{N}{2}$$

Thus combined memory required per stage is '$2N + \frac{N}{2}$'. These many number of memory locations are required to store input values, output values and the twiddle factor per stage.

Now the actual computation of N-point FFT is done stage wise. That means computation of one stage is done at a time. So these memory locations can be used for other stages also. This will again reduce the memory size.

$$\therefore \text{ Total memory locations} = 2N + \frac{N}{2}$$

1.4 Radix-2 Decimation In Frequency (DIF) FFT Algorithm :

Decimation in frequency stands for splitting the sequences in terms of frequency. That means we have to split output sequences into smaller subsequences. This decimation is done as follows :

First stage of decimation :

According to the definition of DFT,

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

We can divide the summation into two parts as follows :

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

Now consider the second summation that means, $\sum_{n=\frac{N}{2}}^{N-1} x(n)\, W_N^{kn}$.

Put $n = n + \frac{N}{2}$; the limits will change as follows :

$$\text{when } n = \frac{N}{2} \Rightarrow \frac{N}{2} = n + \frac{N}{2} \quad \therefore n = 0$$

$$\text{and when } n = N-1 \Rightarrow N-1 = n + \frac{N}{2} \quad \therefore n = N-1-\frac{N}{2} = \frac{N}{2}-1$$

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

Putting this value in Equation (2) we get,

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

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) \cdot W_N^{kn} \cdot W_N^{kN/2}$$

$$\therefore X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{kn} \quad ...(3)$$

Now we have, $W_N = e^{-\frac{j2\pi}{N}}$

$$\therefore W_N^{\frac{kN}{2}} = e^{-\frac{j2\pi}{N} \times \frac{kN}{2}} = e^{-j\pi K} = \left(e^{-j\pi}\right)^k$$

$$\therefore W_N^{\frac{kN}{2}} = (\cos\pi - j\sin\pi)^k = (-1 - j0)^k$$

$$\therefore W_N^{\frac{kN}{2}} = (-1)^K$$

Putting this value in Equation (3) we get,

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

$\therefore$ Taking the summation common we get,

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^k x\left(n+\frac{N}{2}\right) \right] W_N^{kn} \quad ...(4)$$

Here we have to split the sequence in terms of frequency. So we will split X (k) in terms of even numbered and odd numbered DFT coefficients. Let X (2r) represents even numbered DFT and X (2r + 1) represent odd numbered DFT.

Thus putting k = 2r in Equation (4), we will get even numbered sequence.

$$\therefore \quad X(2r) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^{2r} x\left(n+\frac{N}{2}\right) \right] W_N^{2rn} \qquad ...(5)$$

By putting k = 2r + 1, in Equation (4), we will get odd numbered sequence.

$$\therefore \quad X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^{2r+1} x\left(n+\frac{N}{2}\right) \right] W_N^{(2r+1)n} \qquad ...(6)$$

Here 'r' is an integer similar to k and it varies from 0 to $\frac{N}{2}-1$.

$$\therefore \quad (-1)^{2r} = 1 \qquad ...(7)$$

$$\text{and} \quad (-1)^{2r+1} = (-1)^{2r}(-1)^1 = -1 \qquad ...(8)$$

Putting these values in Equations (5) and (6) we get,

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

$$\text{and} \quad X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n+\frac{N}{2}\right) \right] W_N^{(2r+1)n} \qquad ...(10)$$

Now consider the term W_N^{2rn}

$$W_N^{2rn} = \left(W_N^2\right)^{rn}$$

But we have $W_N^2 = W_{N/2}$

$$\therefore \quad W_N^{2rn} = \left(W_{N/2}\right)^{rn} = W_{N/2}^{rn} \qquad ...(11)$$

Now we can write,

$$W_N^{(2r+1)n} = W_N^{2rn} \cdot W_N^n = W_{N/2}^{rn} \cdot W_N^n \qquad ...(12)$$

Putting these values in Equations (9) and (10) we get,

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n+\frac{N}{2}\right) \right] W_{N/2}^{rn} \qquad ...(13)$$

$$\text{and} \quad X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n+\frac{N}{2}\right) \right] W_{N/2}^{rn} \cdot W_N^n \qquad \ldots(14)$$

$$\text{Now let,} \quad g(n) = x(n) + x\left(n+\frac{N}{2}\right) \qquad \ldots(15)$$

$$\text{and} \quad h(n) = \left[x(n) - x\left(n+\frac{N}{2}\right) \right] W_N^n \qquad \ldots(16)$$

Putting these values in Equations (13) and (14) we get,

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1} g(n)\, W_{N/2}^{rn} \qquad \ldots(17)$$

$$\text{and} \quad X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} h(n)\, W_{N/2}^{rn} \qquad \ldots(18)$$

Note that at this stage we have decimated the sequence of 'N' point DFT into two $\frac{N}{2}$ point DFTs given by Equations (17) and (18). Let us consider an example of 8-point DFT. That means N = 8. So combining Equations (17) and (18) (that means $\frac{N}{2}$ = 4) we can obtain N (8 point) point DFT. This is first stage of decimation. Note that Equation (17) indicates $4\left(\frac{N}{2}\right)$ point DFT of g (n) and Equation (18) indicates $4\left(\frac{N}{2}\right)$ point DFT of h (n). For 8-point DFT Equation (15) becomes,

$$g(n) = x(n) + x(n+4) \qquad \ldots(19)$$

Here we are computing '4' point DFT. So range of 'n' is n = 0 to n = 3. Putting these values in Equation (19) we get,

$$\left.\begin{aligned} &\text{For } n = 0 \Rightarrow g(0) = x(0) + x(4) \\ &\text{For } n = 1 \Rightarrow g(1) = x(1) + x(5) \\ &\text{For } n = 2 \Rightarrow g(2) = x(2) + x(6) \\ &\text{For } n = 3 \Rightarrow g(3) = x(3) + x(7) \end{aligned}\right\} \qquad \ldots(20)$$

Similarly for 8 point DFT Equation (16) becomes,

$$h(n) = [x(n) - x(n+4)]\, W_8^n \qquad \ldots(21)$$

$$\text{For } n = 0 \Rightarrow h(0) = [x(0) - x(4)]\, W_8^0$$
$$\text{For } n = 1 \Rightarrow h(1) = [x(1) - x(5)]\, W_8^1$$
$$\text{For } n = 2 \Rightarrow h(2) = [x(2) - x(6)]\, W_8^2$$
$$\text{For } n = 3 \Rightarrow h(3) = [x(3) - x(7)]\, W_8^3 \qquad \text{...(22)}$$

Using Equations (20) and (22), and Equations (17) and (18) we can draw the flow graph of first stage of decimation as shown in Fig. G-15.

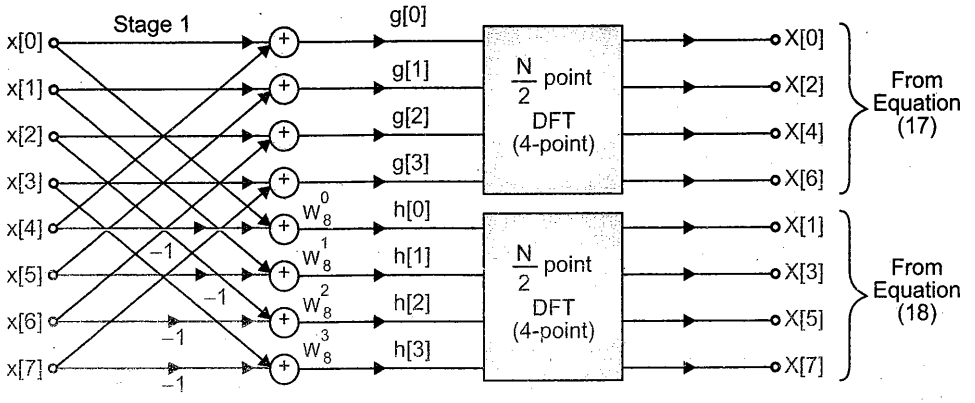

Fig. G-15 : First stage of decimation

Second stage of decimation :

In the first stage of decimation we have used 4-point DFT. We can further decimate the sequence by using 2-point DFT. The second stage of decimation is shown in Fig. G-16.

This is similar to DIT-FFT.

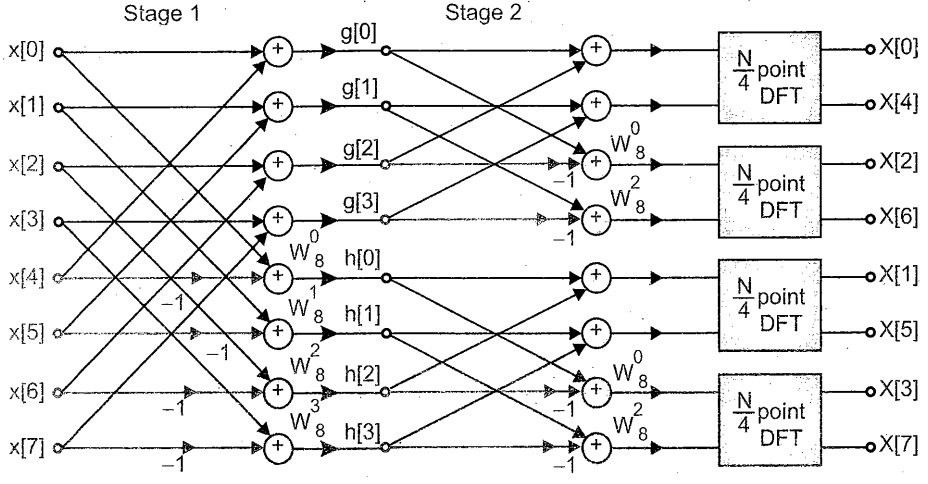

Fig. G-16 : Second stage of decimation

Third stage of decimation :

In the second stage of decimation we have used 2-point DFT. So further decimation is not possible. Now we will use a butterfly structure to obtain 2-point DFT. This butterfly is same as shown in Fig. G-14. Thus the total flow graph for 8 point DIF-FFT is shown in Fig. G-17.

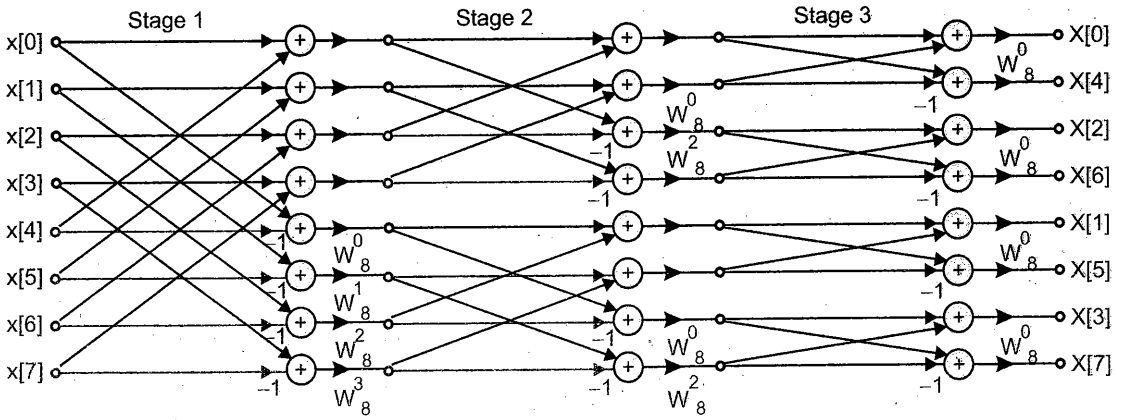

Fig. G-17 : Total flow graph for 8-point DIF-FFT

This flow graph is similar to the flow graph of DIF-FFT but it is in the opposite direction. Note that here input x (n) is in sequence but output is shuffled. Similar to DIT-FFT there are $\log_2 N = \log_2 8 = 3$ stages.

The computational complexity and the memory requirement is same as that of DIT-FFT.