Fast Fourier Transform Algorithms : The Fast Fourier Transform

(FFT) does not represent a transform different from the DFT but they are special algorithms for fast implementation of DFT. FFT requires a comparatively smaller number of arithmetic operation such as multiplications and additions than DFT. FFT also requires lesser computation time than direct computation of DFT. Direct computation of the DFT is less efficient because it does not exploit the properties of symmetry and periodicity of the phase factor ($W_N = e^{-j2\pi/N}$).

These properties are

(i) Periodicity property of W_N: $W_N^{K+N} = W_N^K$

(ii) Symmetry property of W_N: $W_N^{K+\frac{N}{2}} = -W_N^K$

The direct computation of DFT does not use these properties of W_N. The FFT algorithms exploit these properties of phase factor, W_N to reduce calculations of DFT.

The FFT algorithms are based on two basic method. The first one is divide and conquer approach. In this method the 'N' point DFT is divided successively to 2-point DFTs to reduce calculations. The second one is based on linear filtering. Based on this method, there as two algorithms such Goertzel algorithm and chirp Z transform algorithm.

On the basis of decimation process, FFT algorithms are of two types:

1. Decimation-in-Time FFT algorithms
2. Decimation-in-Frequency FFT algorithms.

Radix-2 FFT Algorithms: The radix-2 FF algorithms are based on divide and conquer approach. In this approach the N-point DFT is successively decomposed into smaller DFTs. Because of this decomposition, the number of computation are reduced.

Radix-2 DIT-FFT Algorithm: This FFT algorithm was first proposed by Cooley and Tukey in 1960. In DIT FFT algorithms, the sequence $x(n)$ will be broken up into smaller subsequences are called Decimination-in time FFT algorithms.

Let the N-point data sequence $x(n)$ be splitted into two $\frac{N}{2}$ point data sequences $f_1(n)$ and $f_2(n)$, corresponding to the even-numbered and odd-numbered samples of $x(n)$, respectively, that is

$$\left.\begin{aligned} f_1(n) &= x(2n), \quad n=0,1---\frac{N}{2} \\ f_2(n) &= x(2n+1), \quad n=0,1--\frac{N}{2} \end{aligned}\right\} \text{---(1)}$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating $x(n)$ by a factor of 2, and hence the resulting FFT algorithm is called a decimination-in-time algorithm.

Now the N-point DFT can be expressed in terms of the DFTs of the decimated sequences as follows:

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

$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$

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

Putting the value of $W_N^2 = W_{N/2}$ in above equation, we get

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(m) W_{N/2}^{km}$$

$$X(k) = F_1(k) + W_N^k F_2(k), \quad k = 0, 1 \ldots N-1 \quad \text{---(3)}$$

where $F_1(k)$ and $F_2(k)$ are the $\frac{N}{2}$ point DFTs of sequence $f_1(m)$ and $f_2(m)$ respectively.

Since $F_1(k)$ and $F_2(k)$ are periodic, with period $N/2$, we have $F_1(k + N/2) = F_1(k)$ and $F_2(k + \frac{N}{2}) = F_2(k)$. Hence replacing k by $k + \frac{N}{2}$ in equation (3), we get

$$X(k + \tfrac{N}{2}) = F_1(k + \tfrac{N}{2}) + W_N^{k+\frac{N}{2}} F_2(k + \tfrac{N}{2})$$

$$X(k + \tfrac{N}{2}) = F_1(k) - W_N^k F_2(k) \quad \text{---(4)}$$

The N-point DFT $X(k)$ can be combinely obtained from equation (3) & equation (4) by putting $k = 0$ to $\frac{N}{2} - 1$ i.e

$$X(k) = F_1(k) + W_N^k F_2(k), \quad k = 0, 1, \ldots \tfrac{N}{2}-1 \quad \text{---(5)}$$

$$X(k + \tfrac{N}{2}) = F_1(k) - W_N^k F_2(k), \quad k = 0, 1, \ldots \tfrac{N}{2}-1 \quad \text{---(6)}$$

The above two equations shows that N-point DFT can be obtained by two $\frac{N}{2}$ - point DFTs. This computation is shown in Fig. 1.

Having performed the decimination in time once, we can repeat the process for each of the sequences $f_1(n)$ and $f_2(n)$. Thus $f_1(n)$ would result in the two N/4 -point sequences.

$$v_{11}(n) = f_1(2n), \quad n = 0, 1 - - \frac{N}{4} - 1$$
$$v_{12}(n) = f_1(2n+1), \quad n = 0, 1 - - \frac{N}{4} - 1 \quad \text{---(7)}$$

and $f_2(n)$ would yield

$$v_{21}(n) = f_2(2n), \quad n = 0, 1 - - \frac{N}{4} - 1$$
$$v_{22}(n) = f_2(2n+1), \quad n = 0, 1, - - \frac{N}{4} - 1 \quad \text{---(8)}$$

By computing $\frac{N}{4}$ point DFTs, we would obtain the $\frac{N}{2}$ -point DFTs $F_1(K)$ and $F_2(K)$ from the relations.

$$F_1(K) = V_{11}(K) + W_{N/2}^{K} V_{12}(K), \quad K = 0, 1, - \frac{N}{4} - 1$$
$$F_1(K + \frac{N}{4}) = V_{11}(K) - W_{N/2}^{K} V_{12}(K), \quad K = 0, 1, - \frac{N}{4} - 1 \quad \text{---(9)}$$
$$F_2(K) = V_{21}(K) + W_{N/2}^{K} V_{22}(K), \quad K = 0, 1 - - \frac{N}{4} - 1$$
$$F_2(K + \frac{N}{4}) = V_{21}(K) - W_{N/2}^{K} V_{22}(K), \quad K = 0, 1 - - \frac{N}{4} - 1 \quad \text{---(10)}$$

where the $\{V_{ij}(K)\}$ are the $\frac{N}{4}$ point DFTs of the sequences $\{V_{ij}(n)\}$.

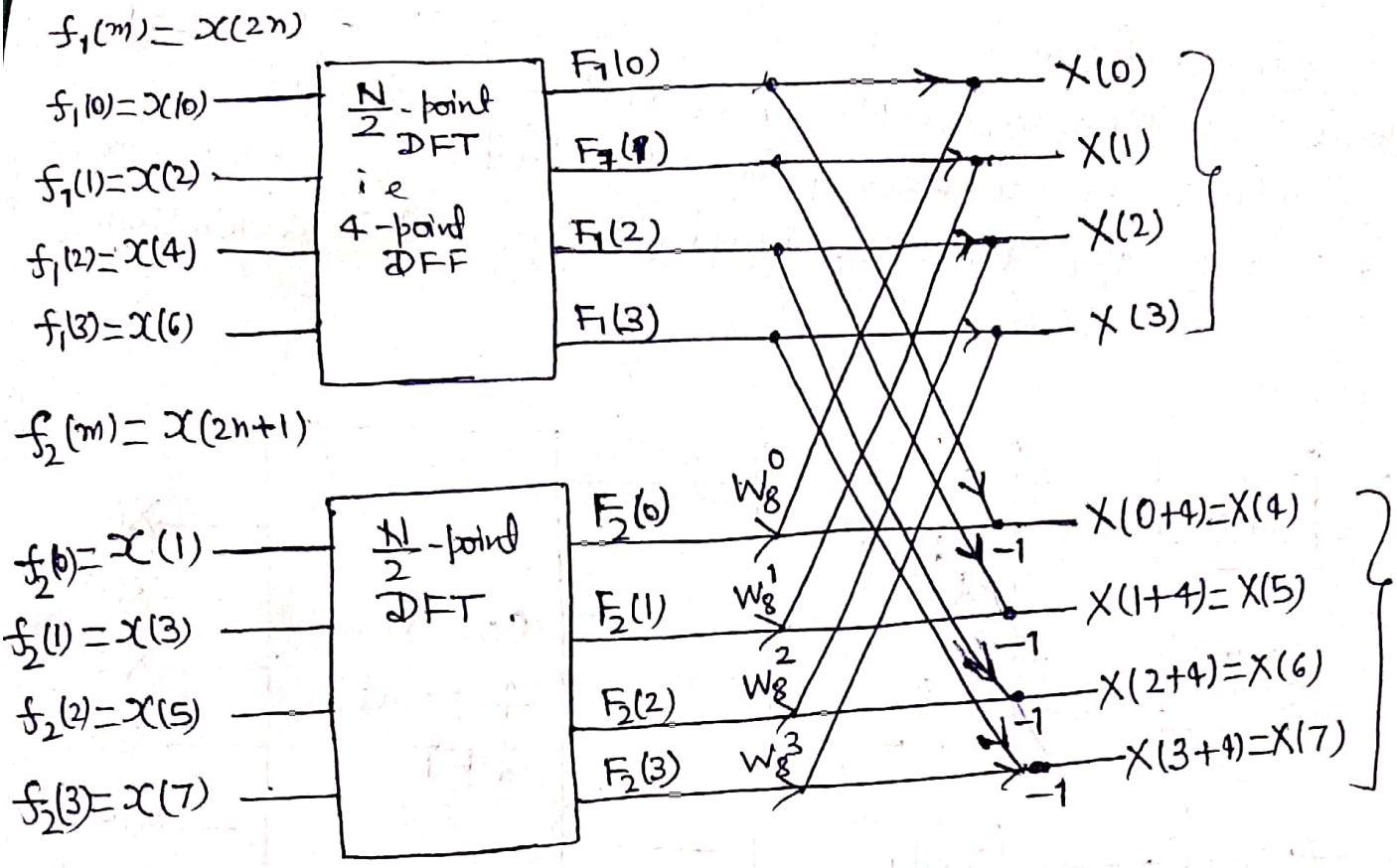

Fig 1: 8-point DFT $X(K)$ is obtained by combining two 4-point DFTs $F_1(K)$ and $F_2(K)$.

Fig. 2 shows the computation of an $N = 8$-point DFT. We observe that the computation is performed in three stages, beginning with the computations of four two point DFTs, then two four-point DFTs, and finally, one eight point DFT. The combination of the smaller DFTs to form the larger DFT is shown in Fig. 3 for $N = 8$.

We observe that the basic computation performed at every stage, as shown in Fig. 3, is to take two complex numbers, say the pair (a, b), multiply b by W_N^r and then add and subtract the product from a to form two new complex numbers (A, B). This basic computation, which is shown in Fig 4, is called a butterfly because the flowgraph resembles a butterfly structure.

In general each butterfly computation involves one complex multiplication and two complex addition.

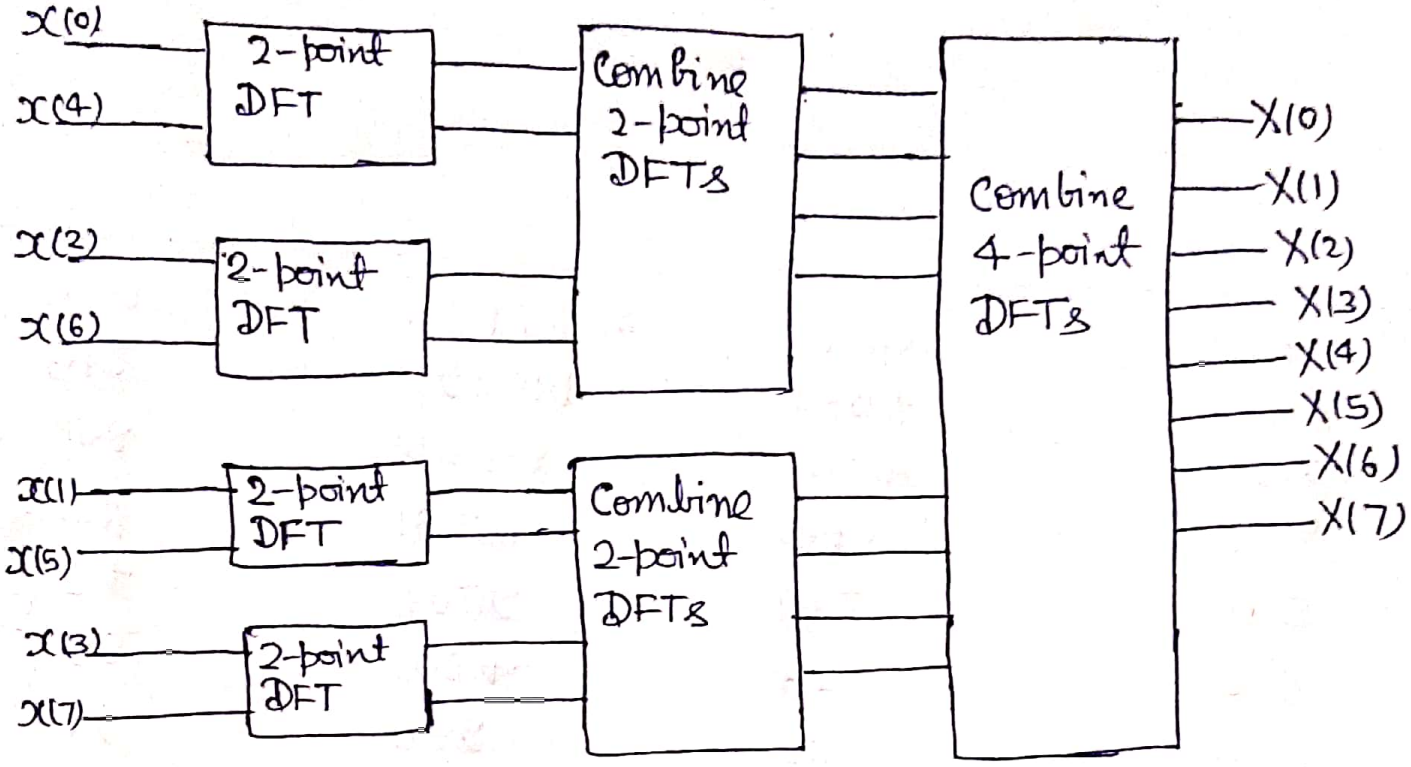

Fig. 2 Three stages in the computation of an N = 8-point DFT.

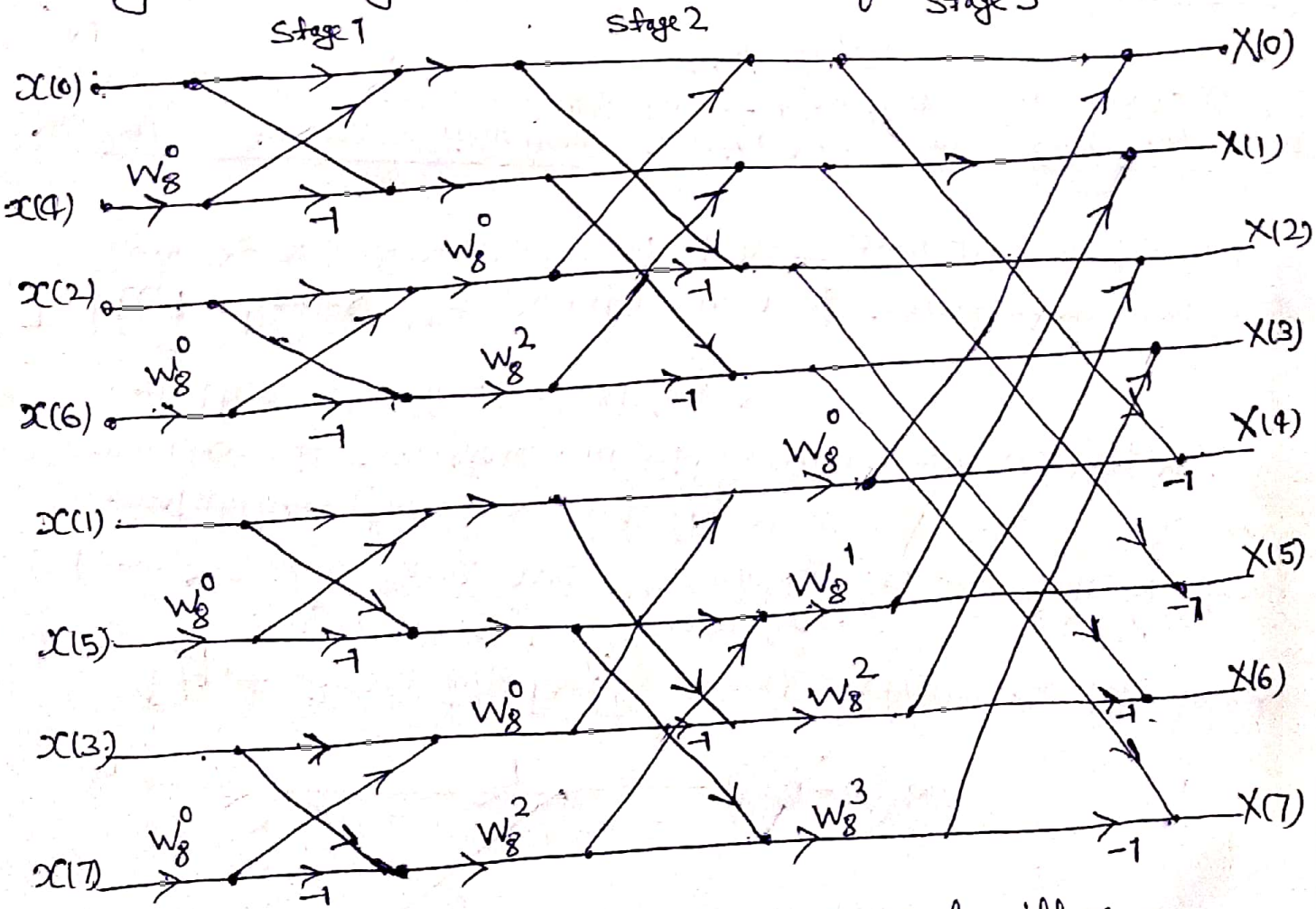

Fig. 3. 8-point decimation in time FFT algorithm.

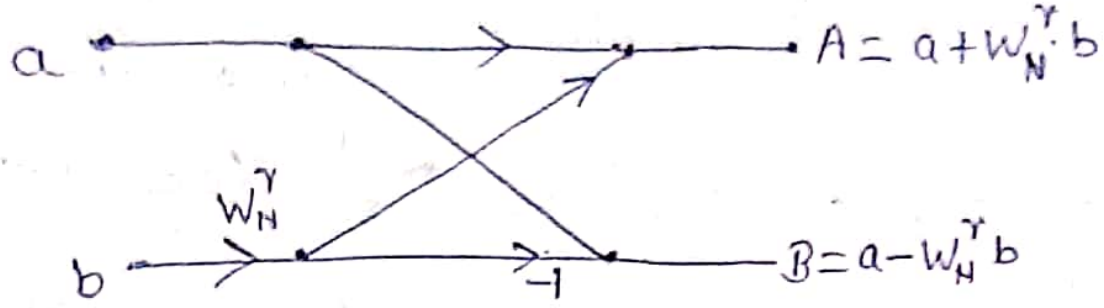

Fig. 4 Basic butterfly computation in the decimation in-time FFT algorithm.

For $N = 2^v$, there are $N/2$ butterflies for stage of the computation process and $\log_2 N$ stages. Hence, the total number of complex multiplications is $(N/2)\log_2 N$ and complex additions is $N \log_2 N$.

Table 1 shows the comparsion of computation complexity for the Direct computation of DFT and DIT FFT algorithm.

Number of Points N	Complex Multiplications in Direct computation N^2	Complex Multiplications in FFT Algorithm $(N/2)\log_2 N$	Speed Improvement Factor $\left(\frac{N^2}{\frac{N}{2}\log_2 N}\right)$
8	64	12	5.3
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.3
128	16,384	448	36.6
256	65,536	1024	64.0
512	262,144	2304	113.8
1024	1048,576	5120	204.8