4. The Fast Fourier Transform

4.1 Introduction

In the preceding chapter we have discussed the discrete Fourier transform and the way it can be used to perform linear filtering. In this chapter we study about a set of algorithms known as Fast Fourier Transform (FFT) proposed by Cooley and Tukey in 1965. The fast Fourier transform is a highly efficient procedure for computing the DFT of a finite series and requires less number of computations than that of direct evaluation of DFT. It reduces the computations by taking advantage of the fact that the calculation of the coefficients of the DFT can be carried out *iteratively*. Due to this, FFT computation technique is used in digital spectral analysis, filter simulation, autocorrelation and pattern recognition.

The FFT is based on decomposition and breaking the transform into smaller transforms and combining them to get the total transform. FFT reduces the computation time required to compute a discrete Fourier transform and improves the performance by a factor 100 or more over direct evaluation of the DFT.

4.2 Direct Evaluation of the DFT

The DFT of a sequence can be evaluated using the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \quad 0 \leq k \leq N-1 \tag{4.1}$$

Substituting $W_N = e^{-j2\pi/N}$, we have

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \leq k \leq N-1 \tag{4.2}$$

$$= \sum_{n=0}^{N-1} \{Re[x(n)] + j\ \mathrm{Im}[x(n)]\} \left\{ Re[W_N^{nk}] + j\ \mathrm{Im}[W_N^{nk}] \right\} \tag{4.3}$$

$$= \sum_{n=0}^{N-1} Re[x(n)]Re[W_N^{nk}] - \sum_{n=0}^{N-1} \text{Im}[x(n)]\,\text{Im}[W_N^{nk}]$$
$$+ j\left\{\sum_{n=0}^{N-1} \text{Im}[x(n)]Re[W_N^{nk}] + \sum_{n=0}^{N-1} Re[x(n)]\,\text{Im}[W_N^{nk}]\right\} \quad (4.4)$$

From Eq.(4.3) we can see that to evaluate one value of $X(k)$, the number of complex multiplications required is N. Therefore to evaluate all N value of $X(k)$, the number of complex multiplications required is N^2. In the same way, to evaluate one value of $X(k)$ the number of complex additions required is $(N-1)$. To evaluate all N values of $X(k)$, the total number of complex additions required is $N(N-1)$. From Eq.(4.4), we can observe that the computation of $X(k)$ for each k requires $4N$ real multiplications. Therefore, to evaluate $X(k)$ for all k from 0 to $N-1$ requires $4N^2$ real multiplications.

Each of the four sums of N terms requires $N-1$ real two-input additions, and to combine the sum to get the real part and imaginary part requires two more. Therefore, to evaluate $X(k)$ for each k requires $4(N-1)+2$ real additions. For all values of k a total number of real additions $N(4N-2)$ required for direct evaluation of the DFT. The above results are obtained by assuming the value of W_N^{kn} as always complex, even though for some values of kn, it equals to 1, -1, j or $-j$. The direct evaluation of the DFT is basically inefficient because it does not use the symmetry and periodicity properties of the twiddle factor W_N. These two properties are

$$\text{Symmetry property: } W_N^{k+N/2} = -W_N^k$$
$$\text{Periodicity property: } W_N^{k+N} = W_N^k \quad (4.5)$$

4.3 The Fast Fourier Transform

The fast Fourier transform algorithms exploit the two basic properties of the twiddle factor shown in Eq. (4.5) and reduces the number of complex multiplications required to perform DFT from N^2 to $\frac{N}{2}\log_2 N$. In other words, for $N = 1024$, this implies about 5000 instead of 10^6 multiplications – a reduction factor of 200.

FFT algorithms are based on the fundamental principle of decomposing the computation of discrete Fourier transform of a sequence of length N into successively smaller discrete Fourier transforms. There are basically two classes of FFT algorithms. They are decimation-in-time and decimation-in-frequency. In decimation -in-time, the sequence for which we need the DFT is successivly divided into smaller sequences and the DFTs of these subsequences are combined in a certain pattern to obtain the required DFT of the entire sequence. In the decimation-in-frequency approach, the frequency samples of the DFT are decomposed into smaller and smaller subsequences in a similar manner.

4.4 Decimation-in-time algorithm

This algorithm is also known as Radix-2 DIT FFT algorithm which means the number of output points N can be expressed as a power of 2, that is, $N = 2^M$, where M is an integer.

Let $x(n)$ is an N-point sequence, where N is assumed to be a power of 2. Decimate or break this sequence into two sequences of length $N/2$, where one sequence consisting of the even-indexed values of $x(n)$ and the other of odd-indexed values of $x(n)$.

$$\text{That is} \quad x_e(n) = x(2n) \quad n = 0, 1, \ldots, \frac{N}{2} - 1$$
$$x_o(n) = x(2n+1) \quad n = 0, 1, \ldots, \frac{N}{2} - 1 \tag{4.6}$$

The N-point DFT of $x(n)$ can be written as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad k = 0, 1, \ldots, N-1 \tag{4.7}$$

Separating $x(n)$ into even and odd indexed values of $x(n)$,we obtain

$$\begin{aligned} X(k) &= \sum_{\substack{n=0 \\ (\text{even})}}^{N-1} x(n) W_N^{nk} + \sum_{\substack{n=0 \\ (\text{odd})}}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{2nk} \end{aligned} \tag{4.8}$$

Substituting Eq.(4.6) in Eq.(4.8) we have

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_e(n) W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_o(n) W_N^{2nk} \tag{4.9}$$

We can write

$$W_N^2 = \left(e^{-j2\pi/N}\right)^2 = e^{-j2\pi/N/2} = W_{N/2}$$
$$\text{i.e.,} \quad W_N^2 = W_{N/2} \tag{4.10}$$

Substituting Eq.(4.10) in Eq.(4.9) we get

$$X(k) = \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x_e(n) W_{N/2}^{nk}}_{N/2\text{–point DFT of even indexed sequence}} + W_N^k \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x_o(n) W_{N/2}^{nk}}_{N/2\text{–point DFT of odd indexed sequence}} \tag{4.11}$$

$$= X_e(k) + W_N^k X_o(k) \tag{4.12}$$

Each of the sums in Eq.(4.11) is an $\frac{N}{2}$-point DFT, the first sum being the $\frac{N}{2}$-point DFT of the even-indexed sequence and the second being the $\frac{N}{2}$-point DFT of the odd-indexed sequence. Although the index k ranges from $k = 0, 1..., N-1$, each of the sums is computed only for $k = 0, 1..., \frac{N}{2} - 1$, since $X_e(k)$ and $X_o(k)$ are periodic in k with period $\frac{N}{2}$. After the two DFTs are computed, they are combined according to Eq.(4.12) to get the N-point DFT of $X(k)$. So Eq. (4.12) holds for the values of $k = 0, 1, \ldots, \frac{N}{2} - 1$.

For $k \geq N/2$

$$W_N^{k+N/2} = -W_N^k \tag{4.13}$$

Now $X(k)$ for $k \geq N/2$ is given by

$$X(k) = X_e\left(k - \frac{N}{2}\right) - W_N^{k-N/2} X_o\left(k - \frac{N}{2}\right) \text{ for } k = \frac{N}{2}, \frac{N}{2} + 1, \ldots N - 1 \tag{4.14}$$

Let us find the number of complex multiplications and complex additions required to compute Eq.(4.12). For direct evaluation of DFT we know that the number of complex multiplications required is equal to N^2. In the same way to calculate $\frac{N}{2}$ - point DFT of $X_e(k)$ we need $\left(\frac{N}{2}\right)^2$ complex multiplications and to compute $X_o(k)$ we need another $\left(\frac{N}{2}\right)^2$ complex multiplications. That is we require $2\left(\frac{N}{2}\right)^2$ complex multiplications. Then the two $\frac{N}{2}$-point DFTs are combined to get $X(k)$. For this we need N complex multiplications. Thus the total number of complex multiplications required for computing $X(k)$ is

$$\left(\frac{N}{2}\right)^2 + \left(\frac{N}{2}\right)^2 + N = N + \frac{N^2}{2}$$

Similarly the total number of complex additions required is

$$\frac{N}{2}\left(\frac{N}{2} - 1\right) + \frac{N}{2}\left(\frac{N}{2} - 1\right) + N = \frac{N^2}{2}$$

The direct evaluation of $X(k)$ requires N^2 Complex multiplications and $N(N-1)$ complex additions. When we decompose $x(n)$ into two subsequences of length $\frac{N}{2}$ and compute $X(k)$ using Eq.(4.12) the number of computations are reduced by a factor of 2. If we again decompose the sequence $X_e(k)$ and $X_o(k)$ into two subsequences, then the amount of computation again be cut in half.

Now let us take $N = 8$. Then $X_e(k)$ and $X_o(k)$ are 4-point $(N/2)$ DFTs of even-indexed sequence $x_e(n)$ and odd-indexed sequence $x_o(n)$ respectively, where

$$\begin{aligned} x_e(0) &= x(0); \quad x_o(0) = x(1) \\ x_e(1) &= x(2); \quad x_o(1) = x(3) \\ x_e(2) &= x(4); \quad x_o(2) = x(5) \\ x_e(3) &= x(6); \quad x_o(3) = x(7) \end{aligned}$$

From Eq.(4.12) and Eq.(4.14) we have

$$\begin{aligned} X(k) &= X_e(k) + W_8^k X_o(k) \quad \text{for} \quad 0 \le k \le 3 \\ &= X_e(k-4) - W_8^{k-4} X_o(k-4) \quad \text{for} \quad 4 \le k \le 7 \end{aligned} \tag{4.15}$$

By substituting different values of k we get

$$\begin{aligned} X(0) &= X_e(0) + W_8^0 X_o(0); X(4) = X_e(0) - W_8^0 X_o(0) \\ X(1) &= X_e(1) + W_8^1 X_o(1); X(5) = X_e(1) - W_8^1 X_o(1) \\ X(2) &= X_e(2) + W_8^2 X_o(2); X(6) = X_e(2) - W_8^2 X_o(2) \\ X(3) &= X_e(3) + W_8^3 X_o(3); X(7) = X_e(3) - W_8^3 X_o(3) \end{aligned} \tag{4.16}$$

From the above set of equations we can find that $X(0)$ and $X(4)$, $X(1)$ and $X(5)$, $X(2)$ and $X(6)$, $X(3)$ and $X(7)$ have same inputs. $X(0)$ is obtained by multiplying $X_o(0)$ with W_8^0 and adding the product to $X_e(0)$. Similarly $X(4)$ is obtained by multiplying $X_o(0)$ with W_8^0 and subtracting the product from $X_e(0)$. This operation can be represented by a butterfly diagram as shown in Fig. 4.1.

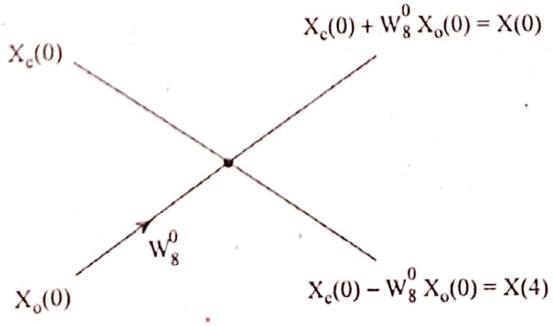

Fig. 4.1 Flow graph of butterfly diagram for Eq.4.16

Now the values $X(k)$ for $k = 1, 2, 3, 4, 5, 6, 7$ can be obtained and an 8-point DFT flowgraph can be constructed from two 4-point DFTs as shown in Fig. 4.2.

From Fig. 4.2 we can find that initially the sequence $x(n)$ is shuffled into even-indexed sequence $x_e(n)$ and odd-indexed sequence $x_o(n)$ and then transformed to give $X_e(k)$ and $X_o(k)$. For $k = 0, 1, 2, 3$ the values $X_e(k)$ and $X_o(k)$ are combined according to Eqs. (4.16) and using butterfly structure shown in Fig. 4.1 the 8-point

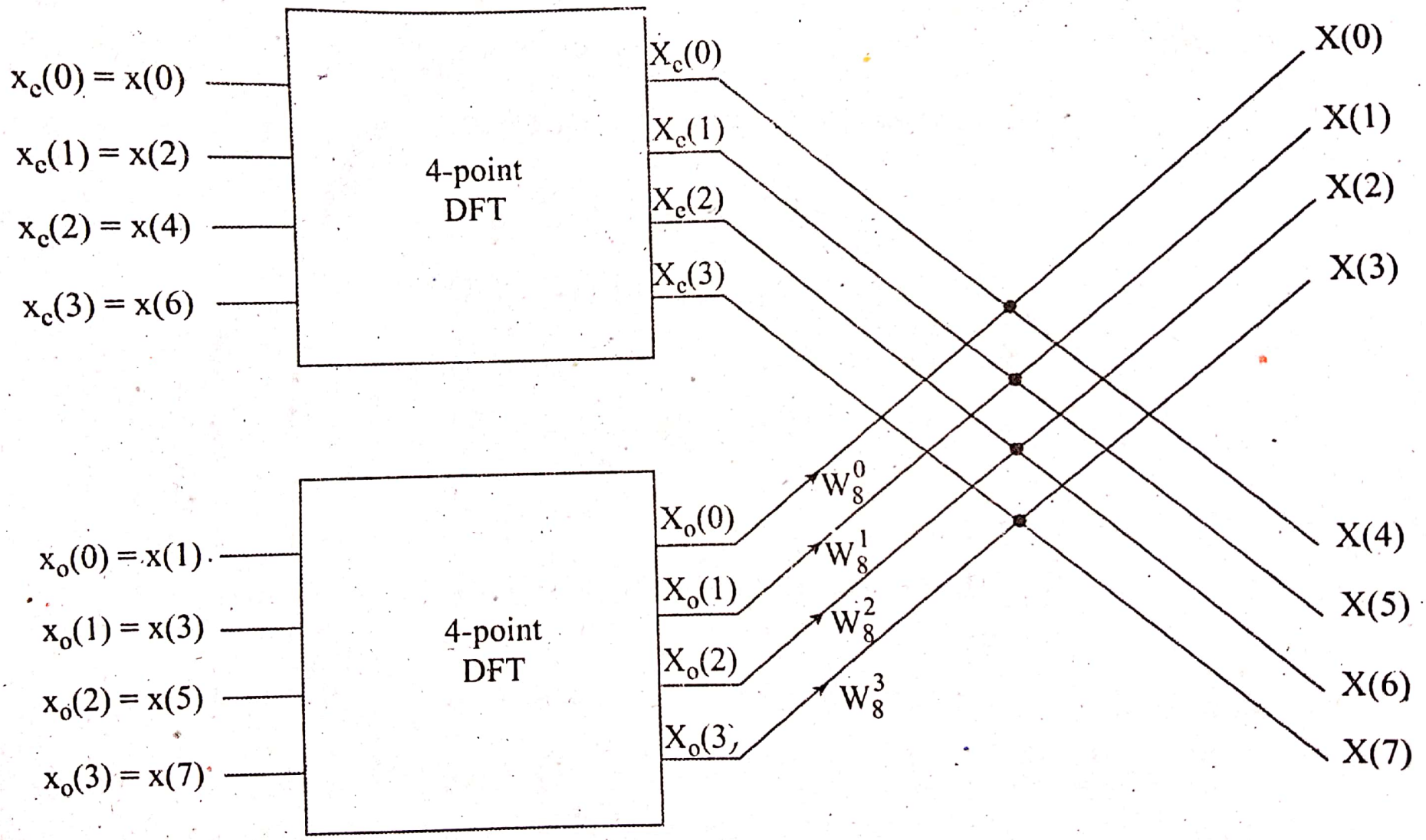

Fig. 4.2 Construction of an 8-point DFT from two 4 point DFTs

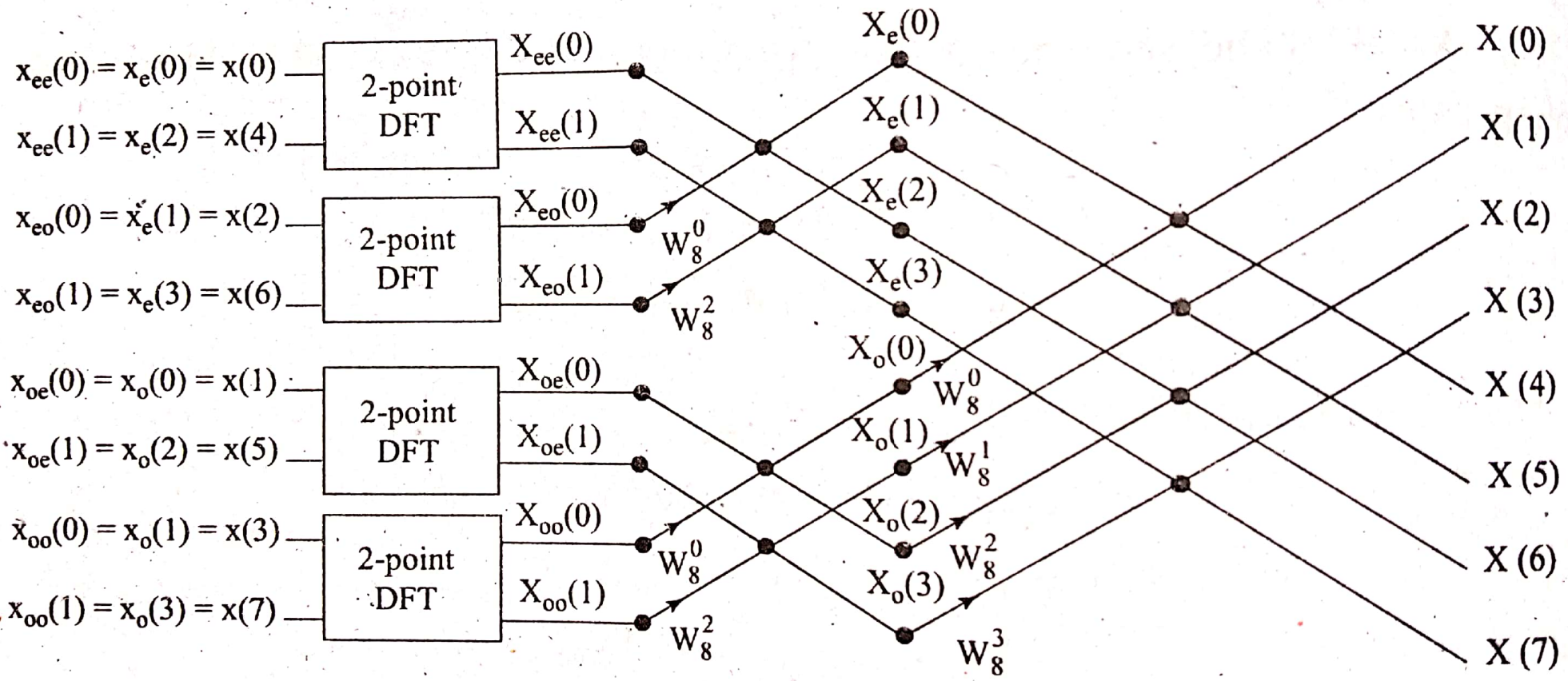

Fig. 4.3 Construction of 8-point DFT from two 4-point DFTs and 4-point DFT from two 2-point DFTs.

half of the multiplications are not really needed. Therefore

$$X_{ee}(0) = x_{ee}(0) + x_{ee}(1) = x_e(0) + x_e(2) = x(0) + x(4)$$
$$X_{ee}(1) = x_{ee}(0) - x_{ee}(1) = x_e(0) - x_e(2) = x(0) - x(4)$$

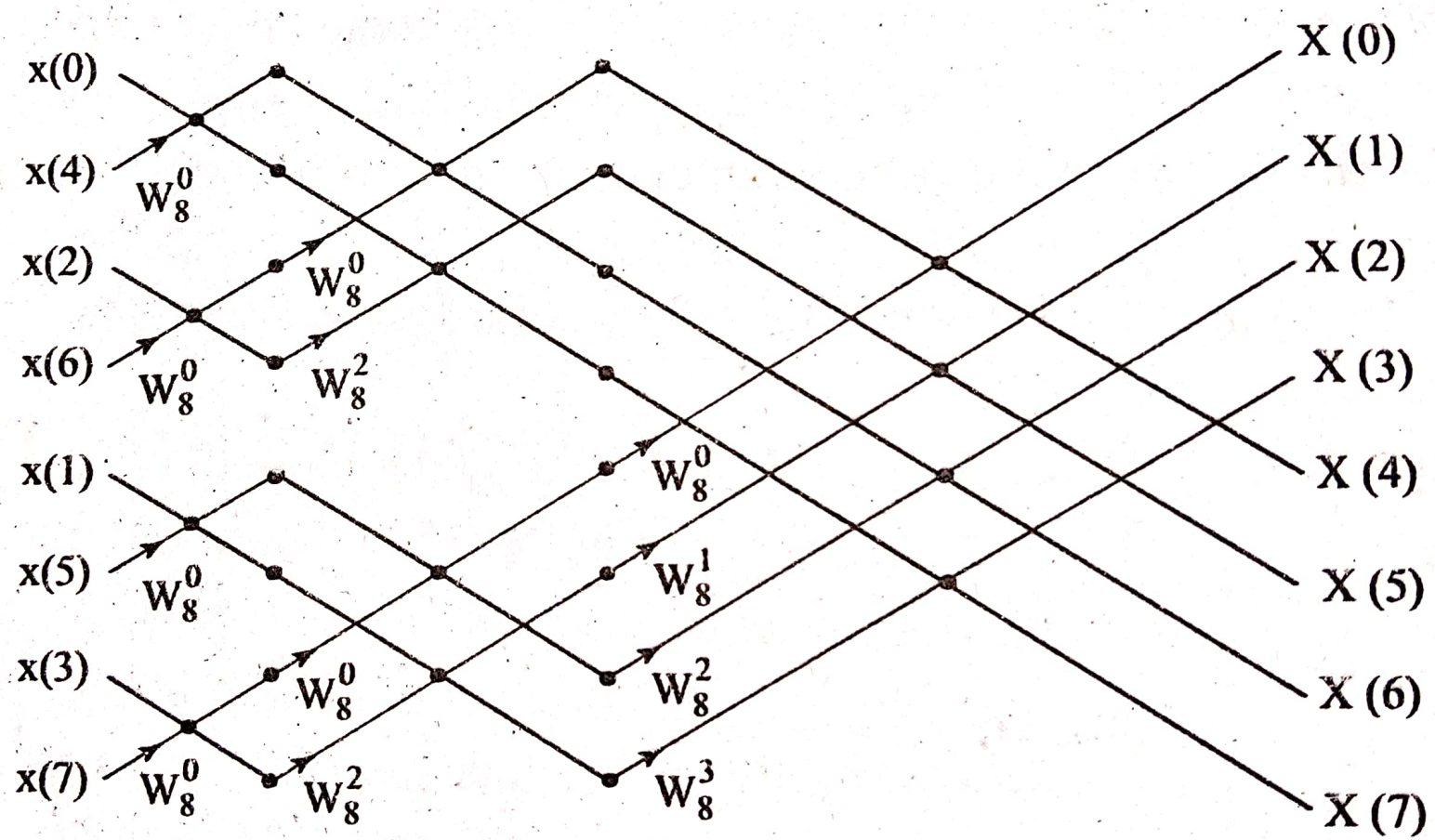

Fig. 4.4 Flow graph of Decimation-in-time algorithm

Table 4.1 Bit-reversal process for $N = 8$

Input sample index	Binary representation	Bit reversed binary	Bit reversed sample index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

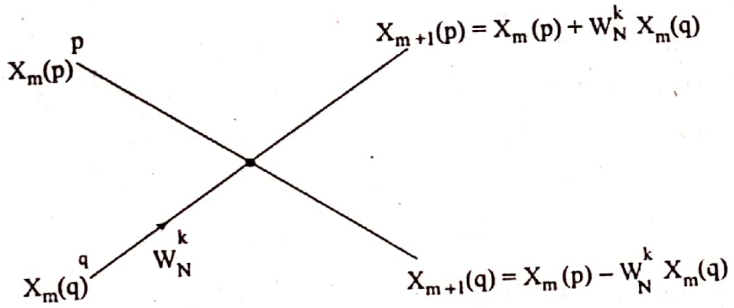

Fig. 4.5 Flow graph of Basic butterfly diagram for DIT algorithm

The FFT algorithm reduces the number of computations. The total number of complex multiplications required for calculating DIT-FFT $= \frac{N}{2} \log_2 N$ and the total number of complex additions for evaluating a DFT using DIT-FFT is $N \log_2 N$.

A comparison of the number of complex multiplications required for direct evaluation of the DFT and the number needed for FFT is given in table 4.2. From the table 4.2 we can find that the FFT algorithm is greater than 100 times faster than the direct evaluation for a 512-point DFT.

Table 4.2 Comparison of number of complex multiplications for the direct evaluation of the DFT versus the FFT algorithm

Number of Stages M	Number of Points N	Number of Complex Multiplications Using		Speed improvement Factor $\frac{N^2}{(N/2)\log_2 N}$
		Direct evaluation N^2	FFT algorithm $(N/2)\log_2 N$	
2	4	16	4	4
3	8	64	12	5.333
4	16	256	32	8
5	32	1,024	80	12.8
6	64	4,096	192	21.33
7	128	16,384	448	36.57
8	256	65,536	1024	64
9	512	2,62,144	2,304	113.77
10	1,024	10,48,576	5,120	204.8

4.5 Summary of Steps of radix - 2 DIT-FFT algorithm

1. The number of input samples $N = 2^M$, where, M is an integer.
2. The input sequence is shuffled through bit-reversal.