5

FAST FOURIER TRANSFORM (FFT)

5.1 Introduction

The Fast Fourier Transform (FFT) does not represent a transform different from the DFT but they are special algorithms for speedier implementation of DFT. FFT requires a comparatively smaller number of arithmetic operations such as multiplications and additions than DFT. FFT also requires lesser computational time than DFT. The fundamental principle on which all these algorithms are based upon is that of decomposing the computation of the DFT of a sequence of length into successively smaller DFTs. The way in which this principle is implemented leads to a variety of different algorithms, all with comparable improvements in computational speed. Thus, we can say that DFT plays an important role in several applications of digital signal processing such as linear filtering, correlation analysis and spectrum analysis.

5.2 Fast Fourier Transform Algorithms

Direct computation of the DFT is less efficient because it does not exploit the properties of symmetry and periodicity of the phase factor $W_N = e^{-j2\pi/N}$

$W_N^K = e^{\frac{-j2\pi \cdot K}{N}}$

These properties are :

Symmetry property : $W_N^{K+N/2} = -W_N^K$

Periodicity property : $W_N^{K+N} = W_N^K$

As we already know that all computationally efficient algorithms for DFT are collectively known as FFT Algorithms and these algorithms exploit the above two properties of phase factor, W_N.

5.3 Classification of FFT Algorithms

A) According to the storage of the components of the intermediate vector, FFT algorithms are classified into two groups.

1. In-Place FFT algorithms
2. Natural Input-Output FFT algorithms.

1) In-Place FFT Algorithms. In this FFT algorithm, component of an intermediate vector can be stored at the same place as the corresponding component of the previous vector.

In-place FFT algorithms reduce the memory space requirement.

2) Natural Input-Output FFT Algorithms. In this FFT algorithm, both input and output are in natural order. It means both discrete-time sequence s(n) and its DFT, S(K) are in natural order. This type of algorithm consumes more memory spare for preservation of natural order of s(n) and S(K).

The disadvantage of an In-place FFT algorithm is that the output appears in an unnatural order necessitating proper shuffling of s(n) or S(K).

In-place FFT algorithms are superior to the Natural Input-output FFT algorithms although it needs shuffling of s(n) or S(K). This shuffling operation is known as Scrambling.

The scrambled value of an integer is defined as a new number generated by reversing the order of all bits in the equivalent binary number for that integer.

B) Another classification of FFT algorithms based on Decimation of s(n) or S(K). Decimation means decomposition into decimal parts.

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

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

1) Decimation-in-Time (BIT) FFT Algorithms. In DIT FFT algorithms, the sequence s(n) will be broken up into odd numbered and even numbered subsequences.

2) Decimation-in-Frequency (DIF) FFT Algorithms. In DIF FFT algorithms, the sequence s(n) will be broken up into two equal halves.

Computation reduction factor of FFT algorithms

$$= \frac{\text{Number of computations required for direct DFT}}{\text{Number of computations required for FFT algorithm}}$$

$$= \frac{N^2}{\frac{N}{2}\log_2(N)}$$

5.4 Number of Stages in DFT Computation using FFT Algorithms

Number of stages in DFT computation using FFT algorithms depends upon the total number of points (N) in a given sequence.

For these algorithms, number of points in a discrete-time sequence,

$$N = 2^r \text{ where } r > 0.$$

r is the number of stages required for DFT computation via FFT algorithms.

Let us have a 8-point discrete-time sequence, $N = 8 = 2^3$. It requires three stages for DFT computations.

In Decimation-in-time (DIT) FFT algorithm, input discrete-time sequence s(n) is in Bit-reversed order but output, S(K) is in Natural order for in-place computation. In Decimation-in-frequency (DIF) FFT algorithm, input discrete-time sequence s(n) is in Natural order but its DFT is in Bit-reversed order for in-place computation. For in-place computation smaller memory space is required.

Generally, we use Radix-2 FFT algorithms. In Radix-2 FFT algorithms, original discrete-time sequence, s(n) is divided in two parts and DFT computation is done on each part separately and resultant of each parts added to get the overall discrete-frequency sequence.

In DIT FFT algorithm, original sequence s(n) is divided in even-numbered points and odd-numbered points. But in DIF FFT algorithm, original discrete-time sequence s(n) is divided in two parts as first half and second half. Fig. 5.1 illustrates the number of stages required in Appoint DFT computation via. DIT FFT algorithm (Here N = 8).

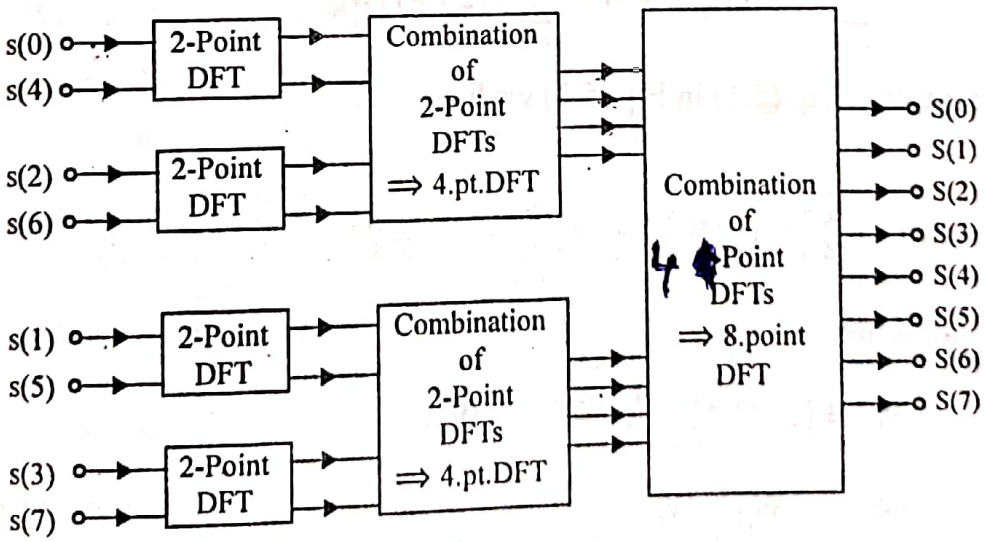


Fig. 5.1 *Three stages in N-point DFT computation via decimation-in-time FFT algorithm (N = 8)*

5.5 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).

$$\text{i.e., } x_e(n) = x(2n) \qquad n = 0, 1 \ldots \frac{N}{2} - 1$$

$$x_0(n) = x(2n+1) \qquad n = 0, 1 \ldots \frac{N}{2} - 1 \qquad \ldots(5.1)$$

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

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} \qquad k = 0, 1, \ldots N-1 \qquad \ldots(5.2)$$

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

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

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

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, W_N^{2nk} + W_N^{nk} \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_N^{2nk} \qquad \ldots(5.3)$$

Substituting Eq. (5.1) in Eq. (5.3) we have

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x_e(n)\, W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_o(n) W_N^{2nk} \qquad \ldots(5.4)$$

we can write

$$W_N^2 = \left(e^{-j2\pi/N}\right)^2 = e^{-j2\pi/N/2} = W_{N/2}$$

$$\text{i.e.,} \qquad W_N^2 = W_{N/2} \qquad \ldots(5.5)$$

Substituting Eq. (5.5) in Eq. (5.4) we get

$$X(k) = \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x_e(n)\, W_{N/2}^{nk}}_{\substack{\text{N/2-point DFT of even} \\ \text{indexed sequence}}} + \underbrace{W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_o(n)\, W_{N/2}^{nk}}_{\substack{\text{N/2-point DFT of odd} \\ \text{indexed sequence}}} \qquad \ldots(5.6)$$

$$= X_e(k) + W_N^k X_o(k) \qquad \ldots(5.7)$$

Each of the sums in Eq. (5.6) 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 are 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. (5.7) to get the N-point DFT of X(k). So the Eq. (5.7) holds good for the values of k = 0, 1... $\frac{N}{2} - 1$.

For $k \geq N/2$

$$W_N^{k+N/2} = -W_N^k \qquad \ldots(5.8)$$

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

$$X(k) = X_e\left(k - \frac{N}{2}\right) - W_N^{k-N/2} X_o\left(k - \frac{N}{2}\right) \text{ for } k = \frac{N}{2}, \frac{N}{2}+1, \ldots N-1 \qquad \ldots(5.9)$$

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

where

$x_e(0) = x(0)$ $\quad x_o(0) = x(1)$

$x_e(1) = x(2)$ $\quad x_o(1) = x(3)$

$x_e(2) = x(4)$ $\quad x_o(2) = x(5)$

$x_e(3) = x(6)$ $\quad x_o(3) = x(7)$

From Eq.(5.7) and Eq. (5.9) we have

$$X(k) = X_e(k) + W_8^k X_o(k) \text{ for } 0 \leq k \leq 3$$

$$= X_e(k-4) - W_8^{k-4} X_o(k-4) \text{ for } 4 \leq k \leq 7 \qquad \ldots(5.10)$$

By substituting different values of k we get

$$X(0) = X_e(0) + W_8^0 X_o(0); \quad X(4) = X_e(0) - W_8^0 X_o(0)$$

$$X(1) = X_e(1) + W_8^1 X_o(1); \quad X(5) = X_e(1) - W_8^1 X_o(1)$$

$$X(2) = X_e(2) + W_8^2 X_o(2); \quad X(6) = X_e(2) - W_8^2 X_o(2)$$

$$X(3) = X_e(3) + W_8^3 X_o(3); \quad X(7) = X_e(3) - W_8^3 X_o(3) \qquad \ldots(5.11)$$

From the above set of equations we can find that X(0) & X(4), X(1) & X(5), X(2) & X(6), X(3) & 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. 5.2

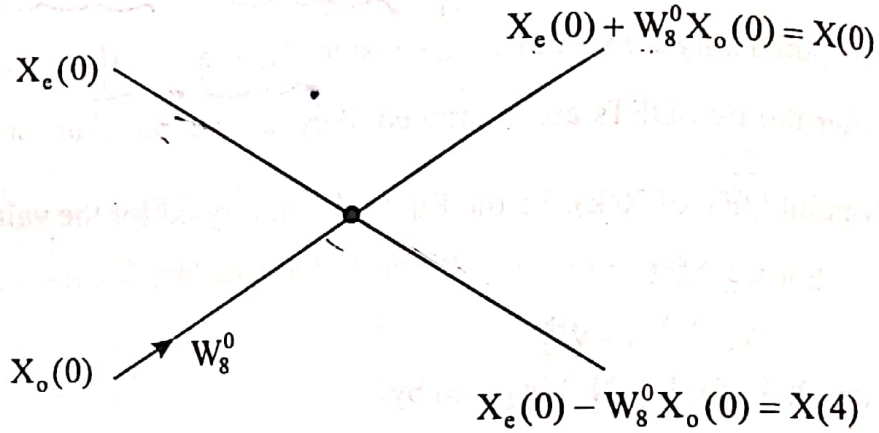


Fig. 5.2 *Flow graph of butterfly diagram for Eq. 5.11*

Now the values X(k) for k = 1, 2, 3, 5, 6, 7 can be obtained and an 8-point DFT flowgraph can be constructed from two 4-point DFTs as shown in Fig. 5.3

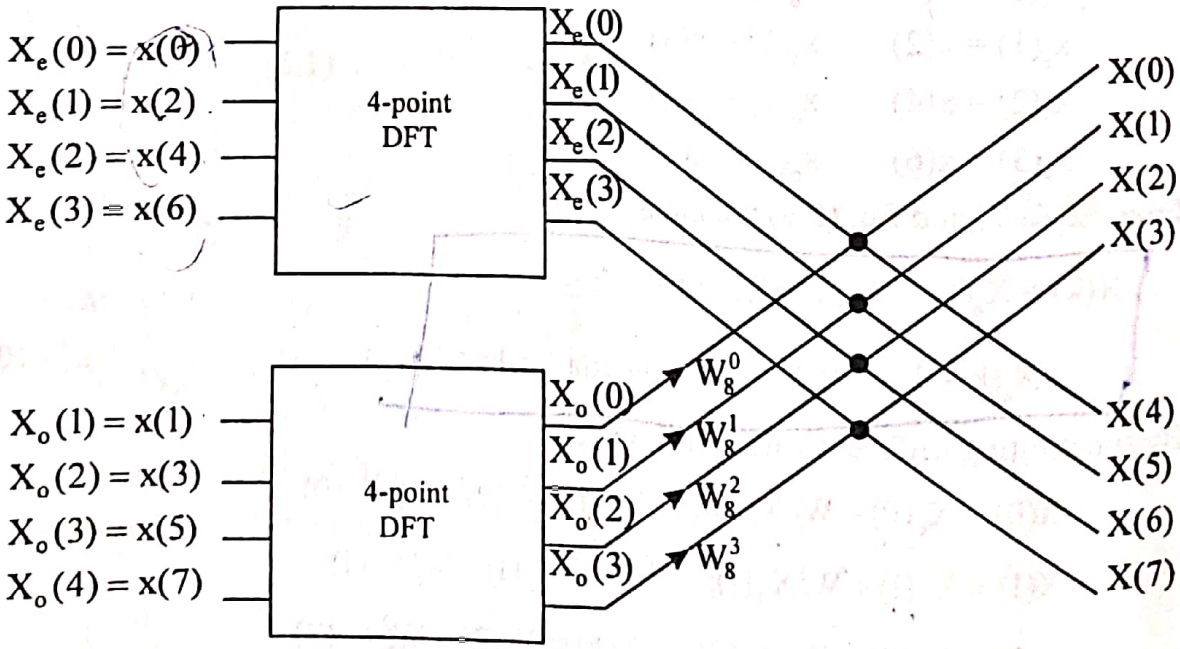


Fig. 5.3 *Construction of an 8-point DFT from two 4 point DFTs*

From the Fig. 5.3 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. (5.11) and using butterfly structure shown in Fig.5.2 the 8-point DFT is obtained. The inputs to the butterfly is separated by $\frac{N}{2}$ samples i.e., 4 samples and the powers of the twiddle factors associated in this set of butterflies are in natural order.

Now we apply the same approach to decompose each of $\frac{N}{2}$ sample DFT. This can be done by dividing the sequence $x_e(n)$ and $x_o(n)$ into two sequences consisting of even and odd members of the sequences. The $\frac{N}{2}$ point DFTs can be expressed as a combination of $\frac{N}{4}$-point DFTs.

i.e. $X_e(k)$ for $0 \le k \le \frac{N}{2}-1$ can be written as

$$X_e(k) = X_{ee}(k) + W_N^{2k} X_{eo}(k) \text{ for } 0 \le k \le \frac{N}{4}-1$$

$$= X_{ee}\left(k-\frac{N}{4}\right) - W_N^{2(k-N/4)} X_{eo}\left(k-\frac{N}{4}\right) \text{for} \frac{N}{4} \le k \le \frac{N}{2}-1 \quad \text{....(5.12)}$$

where $X_{ee}(k)$ is the $\frac{N}{4}$-point DFT of the even members of $x_e(n)$ and $X_{eo}(k)$ is the $\frac{N}{4}$-point of DFT of the odd members of $x_e(n)$.

In the same way

$$X_o(k) = X_{oe}(k) + W_N^{2k} X_{oo}(k) \text{ for } 0 \le k \le \frac{N}{4}-1$$

$$= X_{oe}\left(k-\frac{N}{4}\right) - W_N^{2(k-N/4)} X_{oo}\left(k-\frac{N}{4}\right) \text{for} \frac{N}{4} \le k \le \frac{N}{2}-1 \quad \text{...(5.13)}$$

where $X_{oe}(k)$ is the $\frac{N}{4}$-point DFT of the even members of $x_o(n)$ and $X_{oo}(k)$ is the $\frac{N}{4}$-point DFT of the odd members of $x_o(n)$.

For N = 8

the sequence $x_e(n)$ can be divided into even and odd indexed sequences as

$$x_{ee}(0) = x_e(0); x_{ee}(1) = x_e(2)$$

$$x_{eo}(0) = x_e(1); x_{eo}(1) = x_e(3)$$

Now from Eq. (5.12) we have

$$X_e(0) = X_{ee}(0) + W_8^0 X_{eo}(0)$$

$$X_e(1) = X_{ee}(1) + W_8^2 X_{eo}(1)$$

$$X_e(2) = X_{ee}(0) + W_8^0 X_{eo}(0)$$

$$X_e(3) = X_{ee}(1) + W_8^2 X_{eo}(1) \quad \ldots(5.14)$$

where $X_{ee}(k)$ is the 2 point DFT of even members of $x_e(n)$ and $X_{eo}(k)$ is the 2-point DFT of odd members of $x_e(n)$.

Similarly

the sequence $x_o(n)$ can be divided into even and odd membered sequences as

$$x_{oe}(0) = x_o(0)\, x_{oe}(1) = x_o(2)$$

$$x_{oo}(0) = x_o(1)\, x_{oo}(1) = x_o(3)$$

From the Eq. (5.13) we can obtain

$$X_o(0) = X_{oe}(0) + W_8^0 X_{oo}(0)$$

$$X_o(1) = X_{oe}(1) + W_8^2 X_{oo}(1)$$

$$X_o(2) = X_{oe}(0) + W_8^0 X_{oo}(0)$$

$$X_o(3) = X_{oe}(1) + W_8^2 X_{oo}(1) \quad \ldots(5.15)$$

where

$X_{oe}(k)$ is the 2-point DFT of the even members of $x_o(n)$,

$X_{oo}(k)$ is the 2-point DFT of the odd members of $x_o(n)$.

Fig. 5.4 shows the resulting flow graph when the four-point DFTs of Fig. 5.3 are evaluated as in Eq. (5.14) and Eq. (5.15)

From the Fig. 5.4 we find that the input sequence is again reordered, the input samples to each butterfly are separated by $\frac{N}{4}$ samples i.e., 2 samples and there are two sets of butterflies. In each set of butterflies the twiddle factor exponents are same and separated by two.

For the more general case, we could proceed by decomposing the $\frac{N}{4}$ - point transforms in Eq. (5.12) and Eq. (5.13) into $\frac{N}{8}$ -point transforms and continue until you left with only 2-point transforms. Each decomposition is called a stage, and the total number of stages is given by $M = \log_2 N$. The 8-point DFT requires 3 stages. So far we have seen the decomposition for stage 3 and stage 2. For stage 1 the two point DFT can be easily found by adding and subtracting the input sequences as the twiddle factor associated with first stage is $W_8^0 = 1$, i.e.,

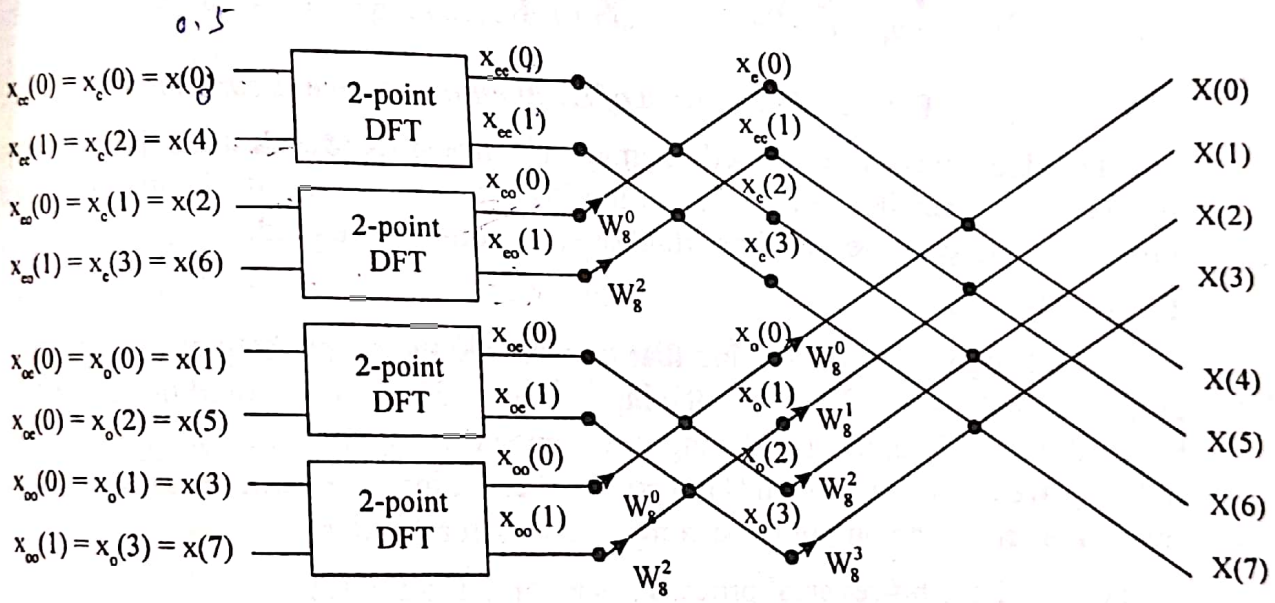


Fig. 5.4 *Construction of 8 point DFT from two 4 point DFTs and 4 point DFT from two point DFTs.*

the first stage involves no multiplication but addition and subtracting. Now we have

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

$$X_{ee}(1) = x_{ee}(0) - x_{ee}(1) = x_e(0) + x_e(2) = x(0) - x(4) \quad \text{....(5.16)}$$

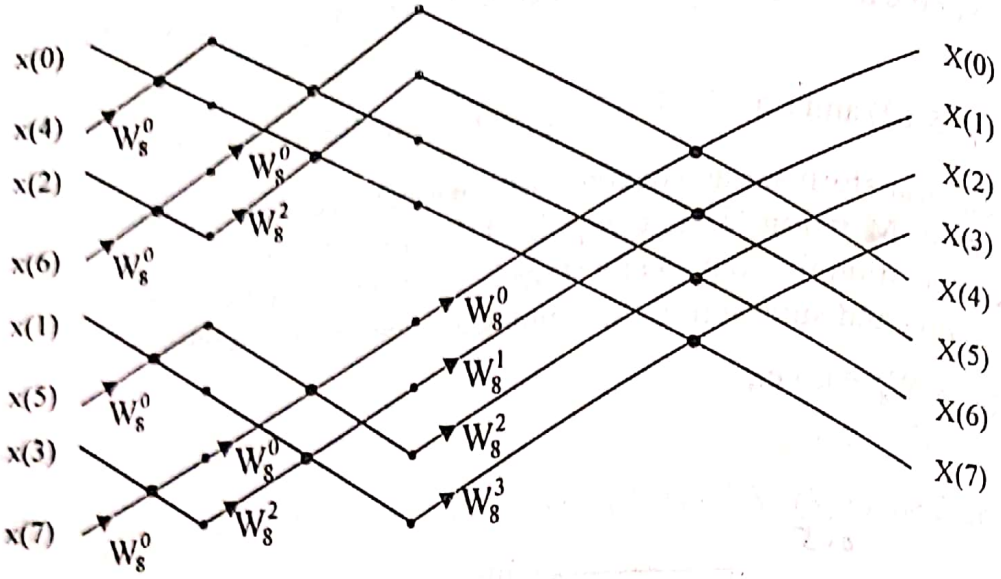


Fig. 5.5. *Flow graph of Decimation-in-time algorithm.*

The algorithm has been called decimation in time since at each stage, the input sequence is divided into smaller sequences i.e. the input sequences are decimated at each stage. From the flow graph several important observations can be made.

1. Bit Reversal

In DIT algorithm we can find that in order for the output sequence to be in natural order (i.e., X(k), k = 0, 1... N – 1) the input sequence had to be stored in a shuffled order. For an 8-point DIT algorithm the input sequence is x(0), x(4), x(2), x(6), x(1), x(5), x(3) and x(7). We can see that when N is a power of 2, the input sequence must be stored in bit-reversal order for the output to be computed in natural order.

For N = 8 the bit-reversal process is shown in table 5.1.

Table 5.1 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

5.5 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.
3. The number of stages in the flowgraph is given by $M = \log_2 N$.
4. Each stage consists of $\frac{N}{2}$ butterflies.
5. Inputs/outputs for each butterfly are separated by 2^{m-1} samples, where m represents the stage index, i.e., for first stage m = 1 and for second stage m = 2 so on.
6. The number of complex multiplications is given by $\frac{N}{2} \log_2 N$.
7. The number of complex additions is given by $N \log_2 N$.
8. The twiddle factor exponents are a function of the stage index m and is given by

$$k = \frac{N t}{2^m} \qquad t = 0, 1, 2, \ldots\ldots 2^{m-1} - 1. \qquad \ldots(5.17)$$

9. The number of sets or sections of butterflies in each stage is given by the formula 2^{M-m}.
10. The exponent repeat factor (ERF), which is the number of times the exponent sequence associated with m is repeated is given by 2^{M-m}.

Table 5.2 Phase Rotation Factors for Quick Computation

Number of points in DFT, N	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
4 No. of stages=2	Twiddle factor not required	W_4^0, W_4^1	–	–	–
8 No. of stages = 3	Twiddle factor not required	W_8^0, W_8^2	W_8^0, W_8^1 W_8^2, W_8^3		
16 No. of stages = 4	Twiddle factor not required	W_{16}^0, W_{16}^4	W_{16}^0, W_{16}^2 W_{16}^4, W_{16}^6 W_{16}^0, W_{16}^5	W_{16}^0, W_{16}^1 W_{16}^2, W_{16}^3 W_{16}^6, W_{16}^7	
32 No. of stages = 5	Twiddle factor not required	W_{32}^0, W_{32}^8	W_{32}^0, W_{32}^4 W_{32}^8, W_{32}^{12}	W_{32}^0, W_{32}^2 W_{32}^{14}	W_{32}^0, W_{32}^1,.. W_{32}^{15}

Example 5.1

Draw the Flow graph of 16-point DIT-FFT.

Solution

1. The number of input Samples, N = 16
2. The input sequence is shuffled through bit-reversal shown in table 5.3 and applied as input to the flow graph.

Table 5.3 Bit-reversal process

Index	Binary Representation	Bit-reversal Order	Bit-reversal Index
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15

3. The number of stages $M = \log_2 16 = 4$.
4. The number of butterflies per stage is $\frac{N}{2} = 8$.
5. The inputs/outputs for each butterfly in stage m is separated by 2^{m-1} samples.

   Stage 1 Inputs/outputs for each butterfly are separated by 1 sample.

   Stage 2 Inputs/outputs for each butterfly are separated by 2 samples.

   Stage 3 Inputs/outputs for each butterfly are separated by 4 samples.

   Stage 4 Inputs/outputs for each butterfly are separated by 8 samples.

6. The number of complex multiplications is given by

$$\frac{N}{2}\log_2 N = 8 \log_2 16 = 32$$

7. The number of complex additions is given by $16\log_2 16 = 64$
8. The number of sets or sections of butterflies in each stage is given by 2^{M-m}
   For Stage 1 the number of sets of butterflies are $2^{4-1} = 8$
   For Stage 2 the number of sets of butterflies are $2^{4-2} = 4$
   For Stage 3 the number of sets of butterflies are $2^{4-3} = 2$
   For Stage 4 there is only one set of butterflies.
9. The twiddle factor exponents for each stage are given by

$$k = \frac{Nt}{2^m} \quad t = 0, 1, 2, \ldots 2^{m-1} - 1.$$

   For Stage 1 the exponent is 0
   For Stage 2 the exponents are 0, 4
   For Stage 3 the exponents are 0, 2, 4, 6
   For Stage 4 the exponents are 0, 1, 2, 3, 4, 5, 6, 7
10. The exponent repeat factor (ERF), which is the number of times the exponent sequence associated with m is repeat is given by 2^{M-m}.
    For stage 1 ERF = 8
    For stage 2 ERF = 4
    For stage 3 ERF = 2
    For stage 4 ERF = 1.

From the steps 8, 9, 10 we can draw the following conclusion.

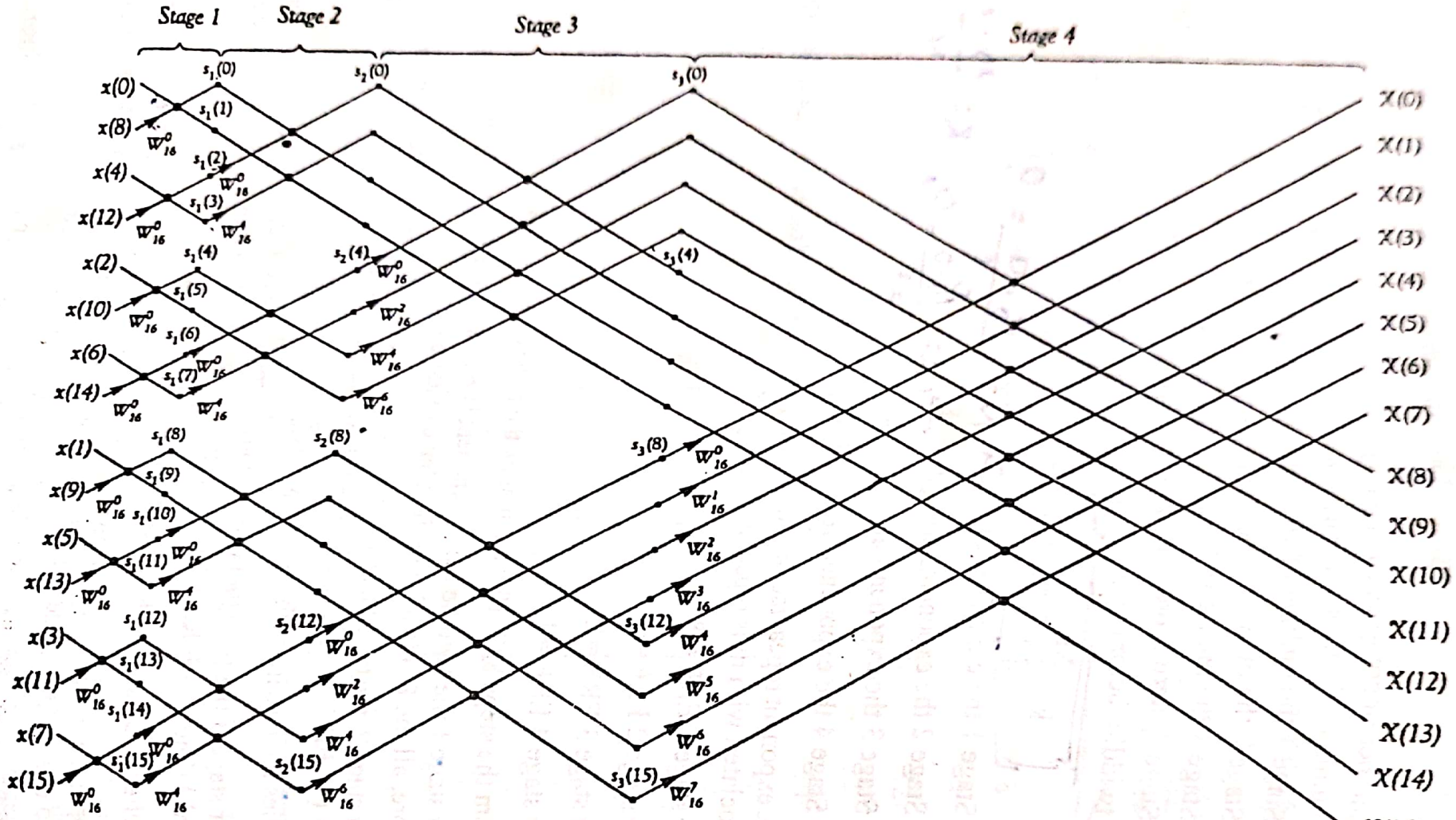
For stage 1 the twiddle factor exponent is zero and is repeated 8 times. ($\because$ ERF = 8). Therefore, all the 8 sets of butterflies have twiddle facts

For stage 2, the twiddle factor exponents sequence is 0, 4 and this sequence is repeated 4 times ($\because$ ERF = 4), i.e., all the 4 sets of butterflies where each set consists of two butterflies have twiddle factors as W_{16}^0, W_{16}^4.

For stage 3 the twiddle factor exponents sequence is 0, 2, 4, 6 and this sequence is repeated 2 times ($\because$ E.R.F = 2), i.e., all the two sets of butterflies where each set consists of 4 butterflies have twiddle factors as $W_{16}^0, W_{16}^2, W_{16}^4, W_{16}^6$.

For stage 4 the twiddle factor exponents sequence is 0, 1, 2, 3, 4, 5, 6, 7 and ERF is equal to one. In this stage the only set of butterflies which consists of 8 butterflies have twiddle factor as $W_{16}^0, W_{16}^1, W_{16}^2, W_{16}^3, W_{16}^4, W_{16}^5, W_{16}^6, W_{16}^7$

Using the above steps the complete flowgraph of 16 point DFT using D1T algorithm is drawn as shown in Fig. 5.6.



Fog 5.6 : *Flow Graph of 16-point DIT-FFT algorithm*

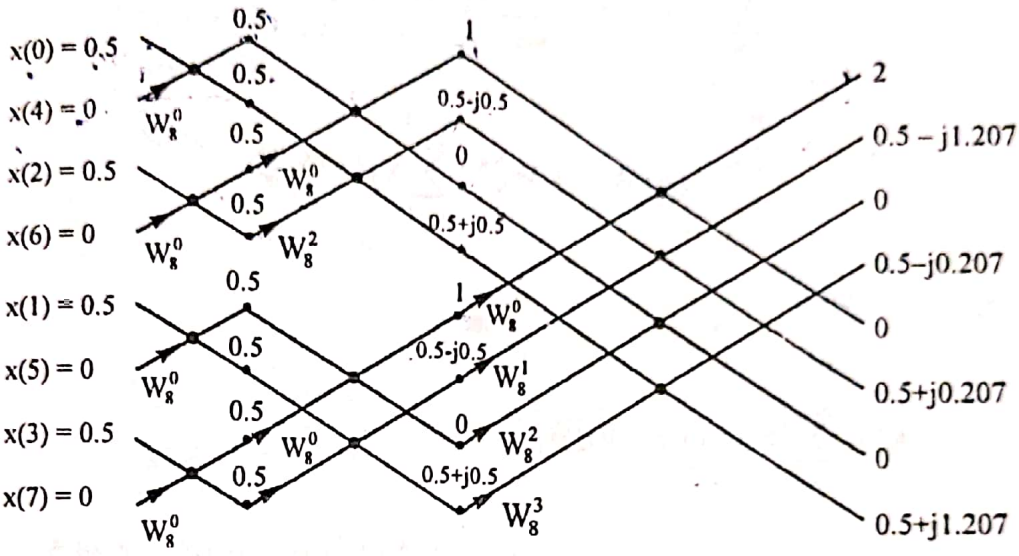
Example 5.2

Compute the eight-point DFT of the sequences x(n) = {0.5, 0.5, 0.5, 0, 0, 0} using the radix-2 DIT algorithm.

Solution

The twiddle factors are

$W^0 = 1$; $W^1 = 0.707 - j0.707$; $W^2 = -j$; $W^3 = -0.707 - j0.707$



X(k) = {2, 0.5 – j 1.207, 0, 0.5 – j 0.207, 0, 05. + j 0.207, 0, 0.5 + j 1.207 }

5.6 Decimation-in-frequency algorithm

DIT algorithm is based on the decomposition of the DFT computation by forming smaller and smaller subsequences of the sequence x(n). In DIP algorithm the output sequence X(k) is divided into smaller and smaller subsequences. In this algorithm the input sequence x(n) is partitioned into two sequences each of length $\frac{N}{2}$ samples. The first sequence $x_1(n)$ consists of first $\frac{N}{2}$ samples of x(n) and the second sequence $x_2(n)$ consists of the last $\frac{N}{2}$ samples of x(n) i.e.,

$$x_1(n) = x(n),\ n = 0, 1, 2, \ldots N/2 - 1 \quad \ldots(5.18)$$

$$x_2(n) = x(n + N/2)\ n = 0, 1, 2, \ldots N/2 - 1 \quad \ldots(5.19)$$

i.e., If N = 8 the first sequence $x_1(n)$ has values for $0 \le n \le 3$ and $x_2(n)$ has values for $4 \le n \le 7$.

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

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

$$= \sum_{n=0}^{\frac{N}{2}-1} x_1(n)\, W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\, W_N^{(n+N/2)k}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x_1(n)\, W_N^{nk} + W_N^{Nk/2} \sum_{n=0}^{\frac{N}{2}-1} x_2(n)\, W_N^{nk}$$

when k is even $e^{-j\pi k} = 1$

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)] W_N^{2nk}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)] W_{N/2}^{nk} \qquad \ldots(5.20)$$

$(\because\ W_N^2 = W_{N/2})$

Eq. (5.20) is the $\frac{N}{2}$-point DFT of the $\frac{N}{2}$-point sequence obtained by adding first half and the last half of the input sequence.

when k is odd $e^{-j\pi k} = -1$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)] W_N^{(2k+1)n}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} [x_1(n) + x_2(n)] W_N^{2kn} W_{N/2}^{nk} \qquad \ldots.(5.21)$$

Eq. (5.21) is the $\frac{N}{2}$-point of DFT of the sequence obtained by subtracting the second half of the input sequence from the first half and multiplying the resulting sequence by W_N^n

Eq. (5.20) and Eq. (5.21) show that the even and odd samples of the DFT N can be obtained from the $\frac{N}{2}$-point DFTs of /(n) and g(n) respectively

where $f(n) = x_1(n) + x_2(n)$ $\quad n = 0, 1 \ldots \frac{N}{2} - 1$

$$g(n) = [x_1(n) - x_2(n)]\ W_N^n \qquad n = 0, 1, \ldots \frac{N}{2} - 1 \qquad \ldots (5.22)$$

The Eq. (5.22) can be represented by a butterfly as shown in Fig. 5.7. This is the basic operation of DIF algorithm.

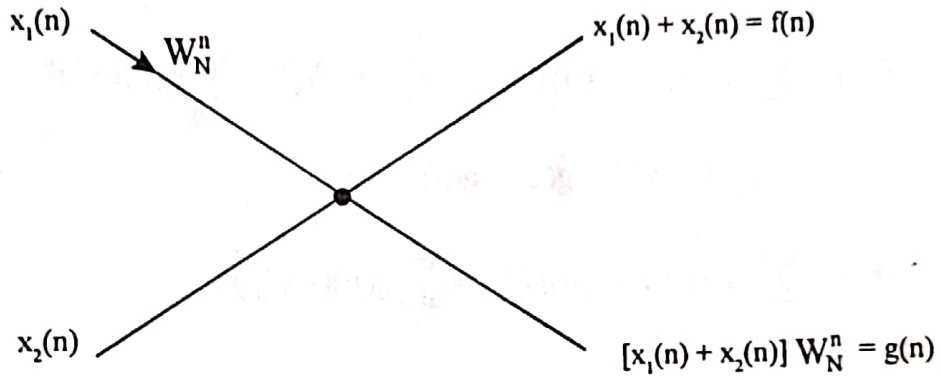


Fig 5.7 *Flow graph of basic butterfly diagram for DIF algorithm*

From Eq. (5.20), for N = 8, we have

$$X(0) = \sum_{n=0}^{3}[x_1(n) + x_2(n)] = \sum_{n=0}^{3} f(n) = f(0) + f(1) + f(2) + f(3) \qquad \ldots (5.23)$$

$$X(2) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{2n} = \sum_{n=0}^{3} f(n)W_8^{2n}$$

$$= f(0) + f(1)W_8^2 - f(2) - f(3)W_8^2$$

$$W_8^4 = (e^{-j2\pi/8})^4 = e^{j\pi} = -1$$
$$W_8^8 = (e^{j2\pi/8})^8 = e^{j2\pi} = 1$$

$$X(4) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{4n} = \sum_{n=0}^{3} f(n)W_8^{4n} = \sum_{n=0}^{3} f(n)(-1)^n$$

$$= f(0) - f(1) + f(2) - f(3) \qquad \ldots . (5.24)$$

$$X(6) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{6n} = \sum_{n=0}^{3} f(n)(-W_8^2)^n$$

$$= f(0) - f(1)\ W_8^2 - f(2) + f(3)\ W_8^2 \qquad \ldots(5.25)$$

From Eq. (5.21) we have

$$X(1) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^n = \sum_{n=0}^{3} g(n) = g(0) + g(1) + g(2) + g(3) \qquad \ldots(5.26)$$

$$X(3) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{3n} = \sum_{n=0}^{3} g(n) W_8^{2n}$$

$$= g(0) + g(1)\ W_8^2 - g(2) - g(3)\ W_8^2 \qquad \ldots(5.27)$$

$$X(5) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{5n} = \sum_{n=0}^{3} g(n) W_8^{4n} = \sum_{n=0}^{3} g(n)(-1)^n$$

$$= g(0) - g(1) + g(2) - g(3) \qquad \ldots(5.28)$$

$$X(7) = \sum_{n=0}^{3}[x_1(n) + x_2(n)]W_8^{7n} = \sum_{n=0}^{3} g(n)(-W_8^2)^n$$

$$= g(0) - g(1)\ W_8^2 - g(2) + g(3)\ W_8^2 \qquad \ldots(5.29)$$

We have seen that the even-valued samples of X(k) can be obtained from the 4-point DFT of the sequence f(n) where.

$$f(n) = x_1(n) + x_2(n) \qquad n = 0, 1 \ldots \frac{N}{2} - 1$$

i.e.,

$$f(0) = x_1(0) + x_2(0)$$
$$f(1) = x_1(1) + x_2(1)$$
$$f(2) = x_1(2) + x_2(2)$$
$$f(3) = x_1(3) + x_2(3) \qquad \ldots(5.30)$$

The odd-valued samples of X(k) can be obtained from the 4-point DFT of the sequence g(n) where $g(n) = [x_1(n) - x_2(n)]\ W_8^n$

i.e.,

$$g(0) = [x_2(0) - x_2(0)]\ W_8^0$$
$$g(1) = [x_1(1) - x_2(1)]\ W_8^1$$
$$g(2) = [x_1(2) - x_2(2)]\ W_8^2$$
$$g(3) = [x_1(3) - x_2(3)]\ W_8^3 \qquad \ldots(5.31)$$

Using the above information and the butterfly structure shown in Fig. 5.7 we can draw the flow graph of 8-point DFT shown in Fig. 5.8.

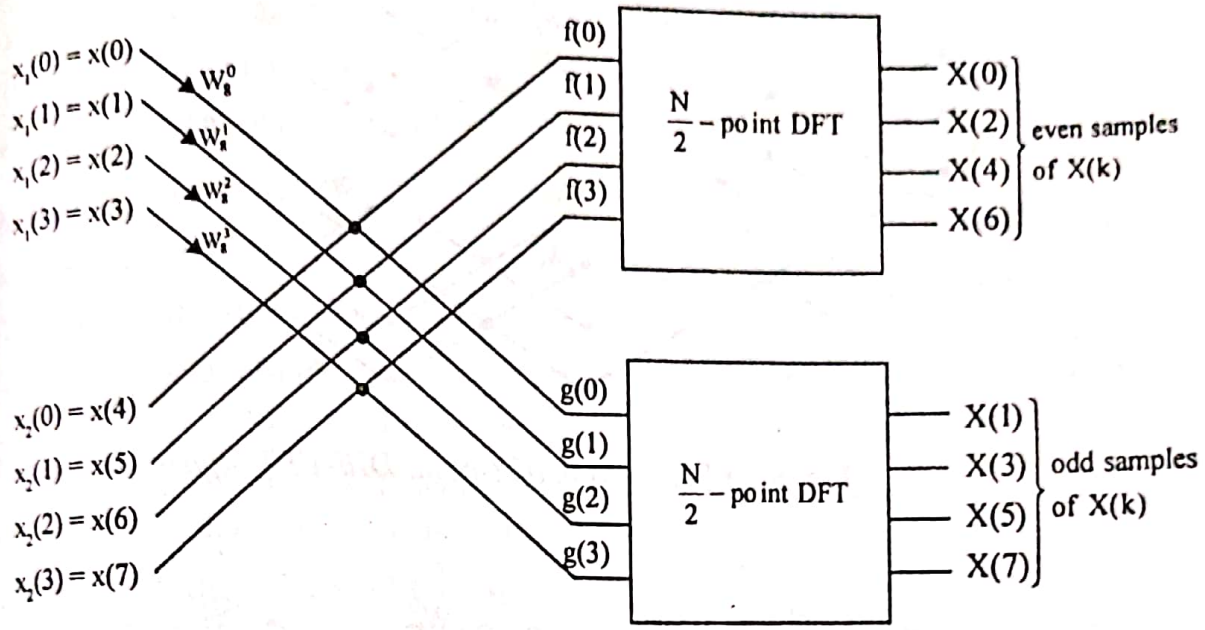


Fig. 5.8 *Reduction of an 8 point DFT to two 4 point DFTs by decimation in frequency*

Now each $\frac{N}{2}$-point DFT can be computed by combining the first half and the last half of the input points for each of the $\frac{N}{2}$-point DFTs and then computing $\frac{N}{4}$-point DFTs. For the 8-point DFT example the resultant flow graph is shown in Fig.5.9

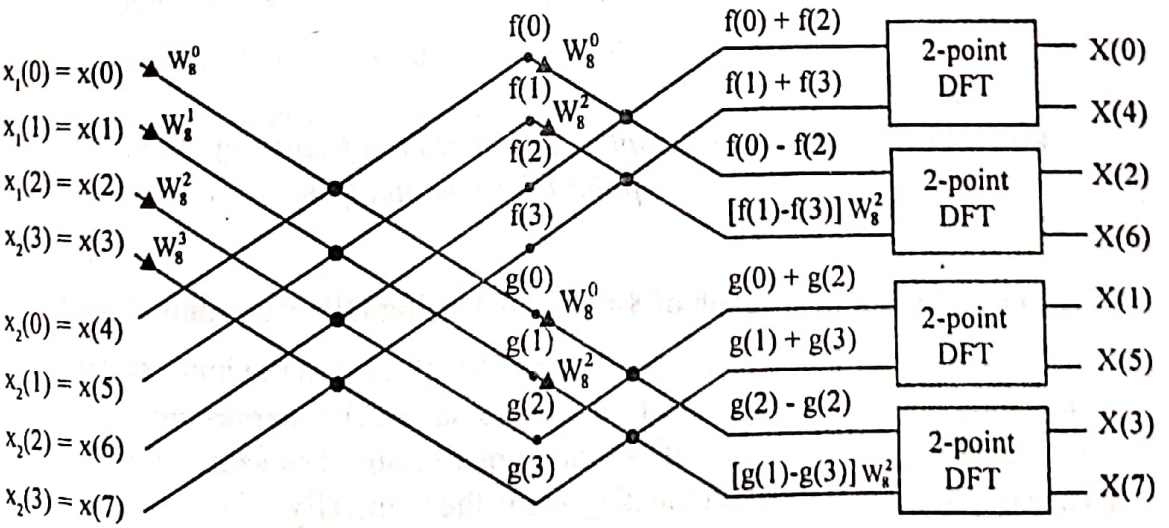


Fig. 5.9 *Flow graph of decimation in frequency decomposition of an 8-point DFT into four 2-point DFT computations*

The 2-point DFT can be found by adding & subtracting the input points. The Fig.5.9 can be further reduced as in Fig. 5.10.

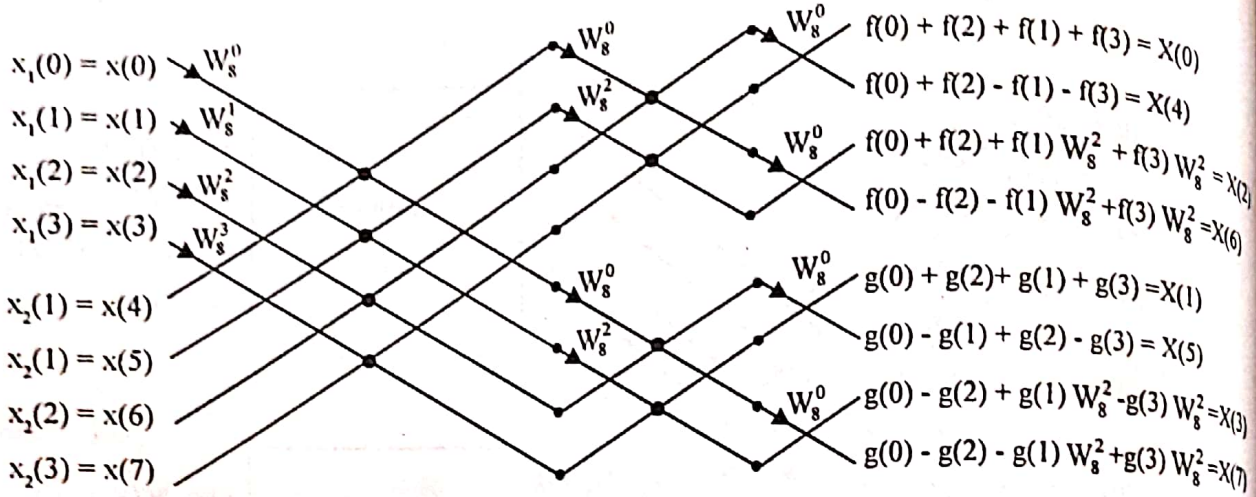


Fig. 5.10 *Flow graph of 8-point DIF-FFT algorithm*

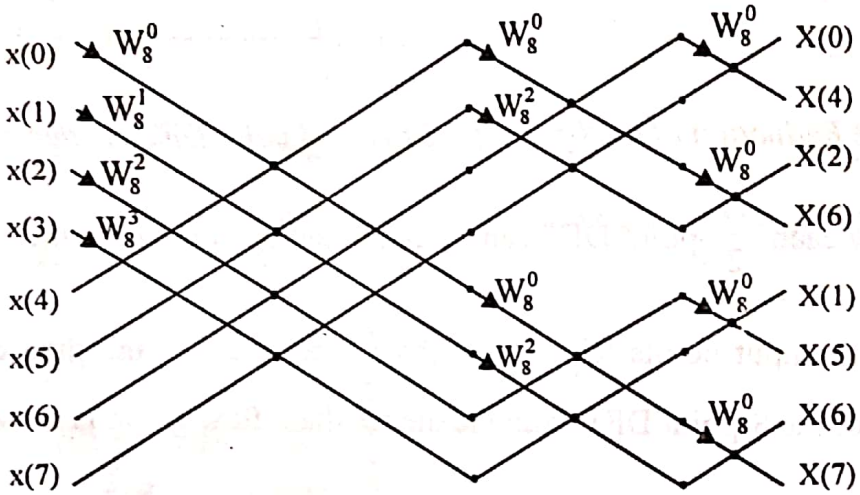


Fig. 5.11 *Flow graph of Complete decimation in frequency decomposition of an 8 point DFT computation*

The complete flow graph of 8-point DFT using DIF algorithm is shown in Fig. 5.11.

From the Fig. 5.11 we observe that for DIF algorithm the input sequence is in natural order, while the output sequence is in bit reversal order, whereas the reverse is true for the DIT algorithm. The number of computations required is same as DIT algorithm. The basic computational block in the diagram is the "butterfly" shown in Fig. 5.12.

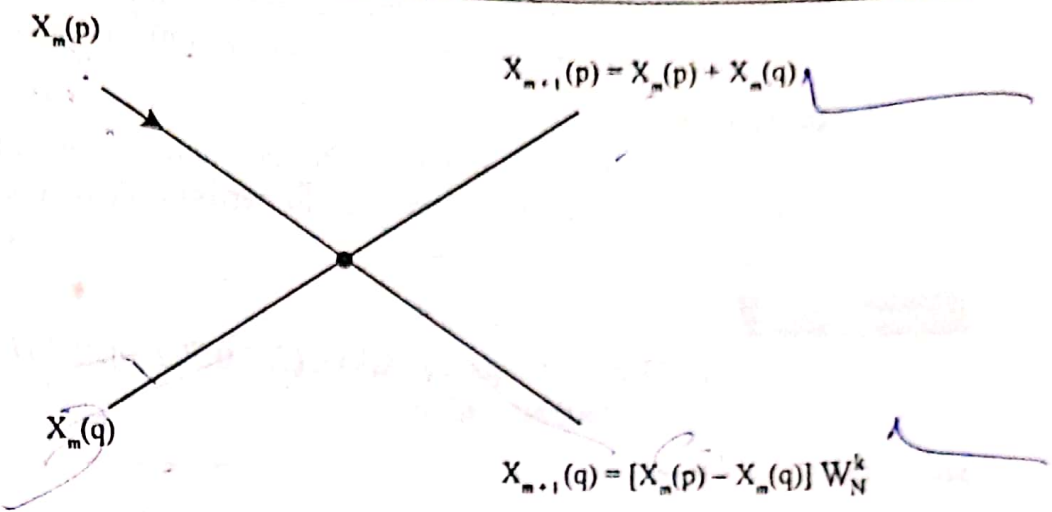


Fig. 5.12 *Basic Computational diagram for DIF-FFT*

Like DIT algorithm, DIF algorithm also in-place algorithm where the same locations are use to store both the input and output sequences.

5.7 Steps for Radix - 2 DIF-FFT algorithm

1. The number of input samples $N = 2$, where, M is number of stages.
2. The input sequence is in natural order.
3. The number of stages in the flow graph is given by $M = \log_2 N$.
4. Each stage consists of $\frac{N}{2}$ butterflies.
5. Inputs/outputs for each butterfly are separated by 2^{M-m} samples, where m represents the stage index i.e., for first stage m = 1 and for second stage m = 2 so on.
6. The number of complex multiplications is given by
7. The number of complex additions is given by $N \log_2 N$.
8. The twiddle -factor exponents are a function of the stage index m and is given by

$$k = \frac{Nt}{2^{M-m+1}} \quad t = 0, 1, 2, \ldots 2^{M-m} \qquad \ldots (5.32)$$

9. The number of sets or sections of butterflies in each stage is given by the formula 2^{m-1}.
10. The exponent repeat factor (ERF), which is the number of times the exponent sequence associated with m is repeated is given by 2

5.8 Differences and similarities between DIT and DIF algorithms

Differences

1. For decimation-in-time (DIT), the input is bit-reversed while the output is in natural order. Whereas, for decimation-in-frequency the input is in natural order while the output is bit reversed order.

2. The DIF butterfly is slightly different from the DIT wherein DIF the complex multiplication takes place after the add-subtract operation.

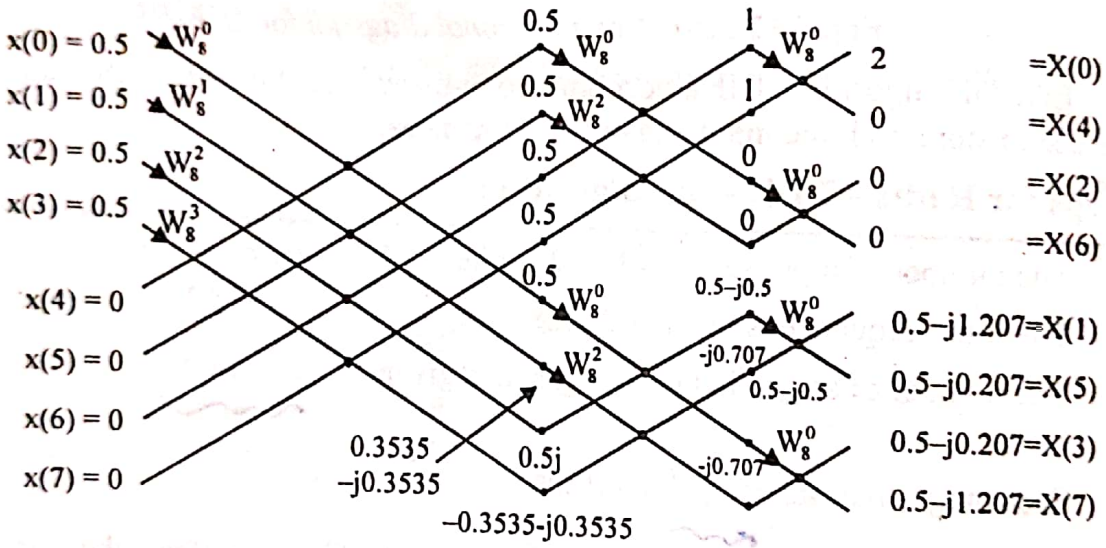
Similarities

Both algorithms require N $\log_2$ N operations to compute the DFT. Both algorithms can be done in-place and both need to perform bit reversal at some place during the computation.

Example 5.3

Compute IDFT of the sequence X(k) = (7, – 0.707, –j, 0.707, 1,0.707 + j 0.707, j, – 0.707 + j 0.707} using DIF algorithm.

Solution



X(k) = {2, 0.5 – j 1.207, 0, 0.5 – j 0.207, 0, 0.5 + j 0.207, 0, 0.5 + j 1.207}

Example 5.4

Compute 4-point DFT of a sequence x(n) = {0, 1, 2, 3} using DIT, DIF algorithm.

Solution

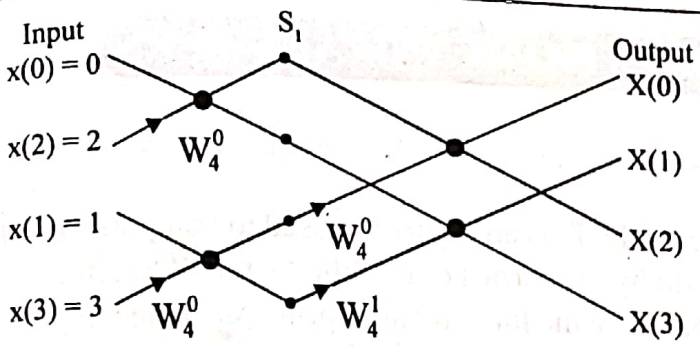
DIT algorithm

Twiddle factors associated with butterflies are

$W_4^0 = 1$; $W_4^1 = e^{-2j\pi/4} = -j$

Bit reversal of input is given by

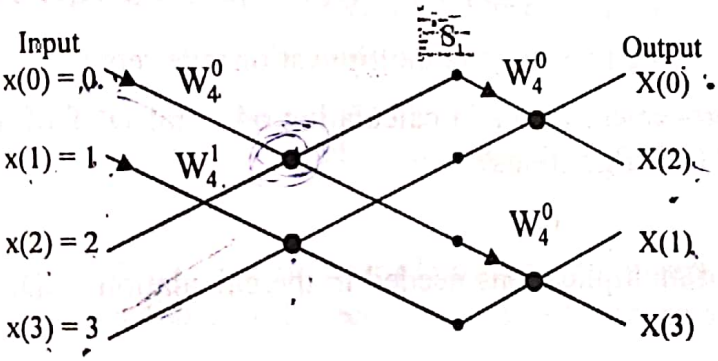
Input index	Binary index	Bit-reversal	Bit-reversal index
0	00	00	0
1	01	10	2
2	10	01	1
3	11	11	3



Input	S_1	Output
0	$0 + 2 = 2$	$2 + 4 = 6$
2	$0 - 2 = -2$	$-2 + (-j)(-2) = -2 + 2j$
1	$1 + 3 = 4$	$2 - 4 = -2$
3	$1 - 3 = -2$	$-2 - (-j)(-2) = -2 - 2j$

$$X(k) = \{6, -2 + 2j, -2, -2 - 2j\}$$

DIF



Input	S_1	Output
0	$0 + 2 = 2$	$2 + 4 = 6$
1	$1 + 3 = 4$	$2 - 4 = -2$
1	$0 - 2 = -2$	$-2 + 2j$
3	$(1 - 3)(-j) = 2j$	$-2 - 2j$

$$X(k) = \{6, -2 + 2j, -2, -2 - 2j\}$$

t=2 means t=0, 1, 2 (In DIT)

t=2 means t=0, ends=2 (In DIF)

QUESTIONS AND ANSWERS

Q.1 What is FFT?

Ans The fast Fourier transform (FFT) is an algorithm used to compute the DFT. It makes use of the symmetry and periodicity properties of twiddle factor W to effectively reduce the DFT computation time. It is based on the fundamental' principle of decomposing the computation of DFT of a sequence of length N into successively smaller discrete Fourier transforms. The FFT algorithm provides speed-increase factors, when compared with direct computation of the DFT, of; approximately 64 and 205 for 256-point and 1024-point transforms, respectively,

Q.2 Why FFT is needed?

Ans The direct evaluation of DFT using the formula $X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N}$ requires N^2 complex multiplications and N(N — 1) complex additions. Thus for reasonably large values of N (in the order of 1000) direct evaluation of the DFT requires an inordinate amount of computation. By using FFT algorithms the number of computations can be reduced. For example, for an N-point DFT, the number of complex multiplications required using FFT is $\frac{N}{2}\log_2 N$. If N = 16, the number of complex multiplications required for direct evaluation of DFT is 256, whereas using FFT only 32 multiplications are required.

Q.3 What is the speed improvement factor in calculating 64-point DFT of a sequence using direct computation and FFT algorithms?

or

Calculate the number of multiplications needed in the calculation of DFT and FFT with 64-point sequence.

Ans The number of complex multiplications required using direct computation is

$$N^2 = 64^2 = 4096$$

The number of complex multiplications required using FFT is N

$$\frac{N}{2}\log_2 N = \frac{64}{2}\log_2 64 = 192.$$

$$\text{Speed improvement factor} = \frac{4096}{192} = 21.33$$

Q.4 What is the main advantage of FFT?

Ans FFT reduces the computation time required to compute discrete Fourier transform.

Q.5 Calculate the number of multiplications needed in the calculation of DFT using FFT algorithm with 32-point sequence.

Ans For N-point DFT the number of complex multiplications needed using FFT algorithm is $\frac{N}{2}\log_2 N$.

For N = 32, the number of complex multiplications is equal to

$$\frac{32}{2}\log_2 32 = 16 \times 5 = 80.$$

Q.6 How many multiplications and additions are required to compute N-point DFT using radix-2 FFT ?

Ans The number of multiplications and additions required to compute N-point DFT using radix-2 FFT are $N \log_2 N$ and $\frac{N}{2} \log_2 N$ respectively.

Q.7 What is meant by radix-2 FFT?

Ans The FFT algorithm is most efficient in calculating N-point DFT. If the number of output points N can be expressed as a power of 2, that is, $N = 2^M$, where M is an integer, then this algorithm is known as radix-2 FFT algorithm,

Q.8 What are the differences and similarities between DIP and DIT algorithms?

Ans ***Differences***

1. For DIT, the input is bit reversed while the output is in natural order, whereas for DIF, the input is in natural order while the output is bit reversed.
2. The DIF butterfly is slightly different from the DIT butterfly, the difference being that the complex multiplication takes place after the add-subtract operation in DIP.

Similarities

Both algorithms require same number of operations to compute the DFT. Both algorithms can be done in-place and both need to perform bit reversal at. some place during the computation.

Q.9 What is the basic operation of the DIT algorithm?

Ans The basic operation of the DIT algorithm is the so called butterfly in which two inputs $X_m(p)$ and $X_m(q)$ are combined to give the outputs $X_{m+1}(p)$ and $X_{m+1}(q)$ via. the operation

$$X_{m+1}(p) = X_m(p) + W_N^k X_m(q)$$

$$X_{m+1}(q) = X_m(p) - W_N^k X_m(q)$$

where W_N^k is twiddle factor.

Q.10 What is the basic operation of the DIF algorithms?

Ans The basic operation of the DIF algorithm is the so called butterfly in which two inputs

$X_m(p)$ and $X_m(q)$ are combined to give the outputs $X_{m+1}(p)$ and $X_{m+1}(q)$ via the operation

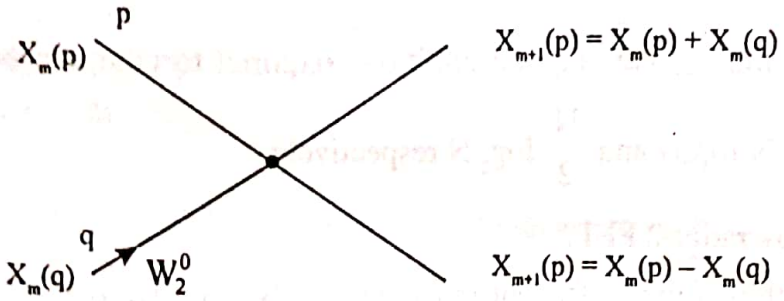
$$X_{m+1}(p) = X_m(p) + X_m(q)$$

$$X_{m+1}(q) = [X_m(p) - X_m(q)]\, W_N^k$$

where W_N^k is twiddle factor.

Q.11 Draw the flow graph of a two-point DFT for a decimation-in-time decomposition.

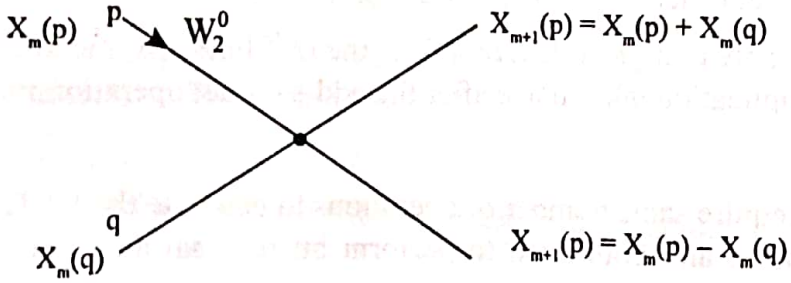
Ans The flow graph of a two-point DFT for a decimation-in-time algorithm is



where $X_m(p)$ and $X_m(q)$ are inputs to the butterfly, $X_{m+1}(p)$ and $X_{m+1}(q)$ are outputs of the butterfly. The nodes p and q represents memory locations.

Q.12 Draw the flow graph of a two-point radix-2 DIF-FFT.

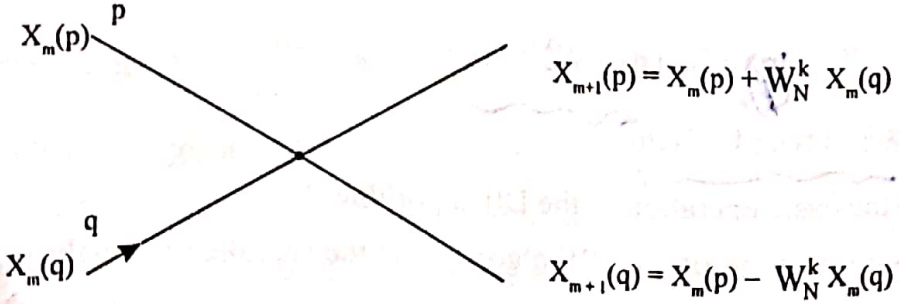
Ans The flow graph of a two-point DFT for a decimation-in-time frequency algorithm is



where $X_m(p)$ and $X_m(q)$ are inputs, $X_{m+1}(p)$ and $X_{m+1}(q)$ are outputs of the butterfly. The nodes p and q represents memory locations.

Q.13 Draw the basic butterfly diagram for DIT algorithm.

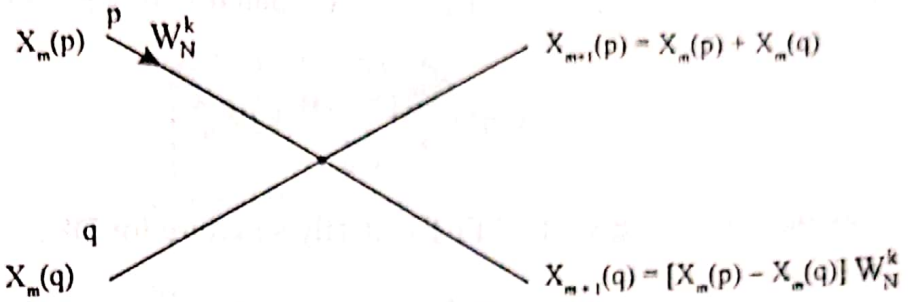
Ans The basic butterfly diagram for DIT algorithm is



where $X_m(p)$ and $X_m(q)$ are inputs to the butterfly, $X_{m+1}(p)$ and $X_{m+1}(q)$ are outputs of the butterfly. The nodes p and q represents memory locations.

Q.14 Draw the basic butterfly diagram for DIP algorithm.

Ans The basic butterfly diagram for DIP algorithm is



where $X_m(p)$ and $X_m(q)$ are inputs, $X_{m+1}(p)$ and $X_{m+1}(q)$ are outputs of the butterfly. The nodes p and q represents memory locations.

Q.15 What is meant by 'm-place' in DIT and DIF algorithms?

Ans The basic butterfly diagrams used in DIT and DIF algorithms are shown in Fig. 1 and Fig. 2 respectively.

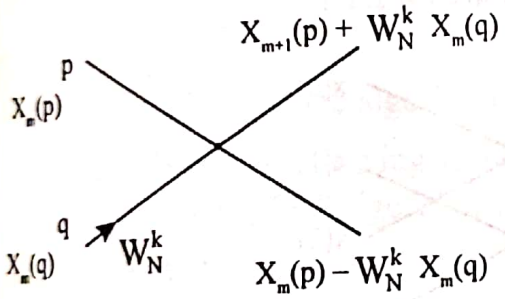


Fig.1

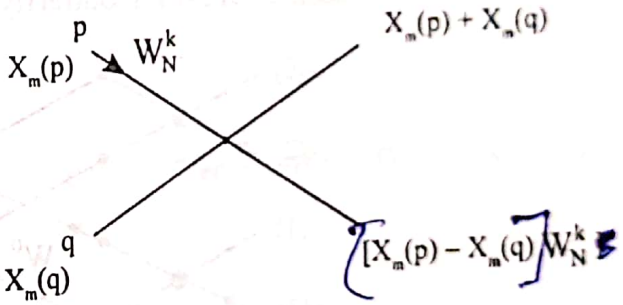


Fig.2

In the Fig. 1 two lines emerging from two nodes cross each other and connected to two nodes on the right hand side. These nodes represents memory locations. At the input nodes $X_m(p)$ and $X_m(q)$, the inputs are stored. After the outputs $X_{m+1}(p)$ and $X_{m+1}(q)$ are calculated, the same memory location is used to store the new values in place of the input values. An algorithm that use the, same location to store both the input and output sequences is called an 'in-place' algorithm.

Q.16 How we can calculate IDFT using FFT algorithm ?

Ans The inverse DFT of an N-point sequence X(k); k = 0, 1... N – 1 is defined as

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

If we take complex conjugate and multiply by N, we get

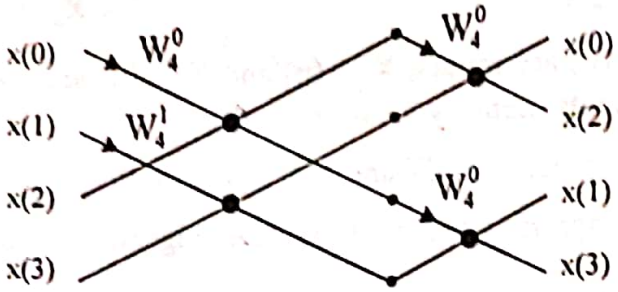
$$Nx^*(n) = \sum_{k=0}^{N-1} X^*(k) W_N^{nk} \quad \ldots (2)$$

The right hand side of the above equation is DFT of the sequence $X^*(k)$ and may be computed using any FFT algorithm. The desired output sequence x(n) can then be obtained by complex conjugating the DFT of Eq. (2) and dividing by N to give.

$$x(n) = \frac{1}{N}\left[\sum_{k=0}^{N-1} X^*(k) W_N^{nk}\right]^*$$

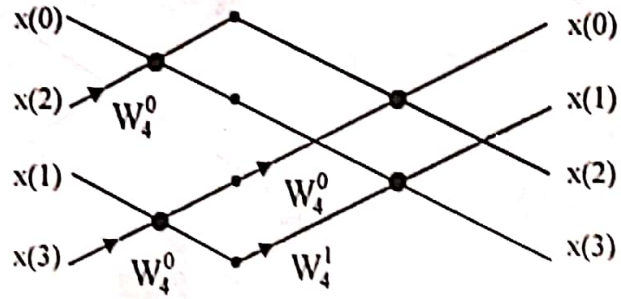
Q.17 Draw the 4-point radix 2 DIF-FFT butterfly structure for DFT.

Ans



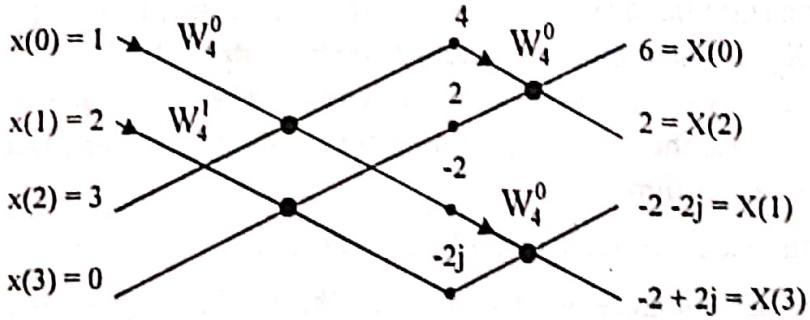
Q.18 Draw the 4-point radix-2 DIT-FFT butterfly structure for DFT.

Ans



Q.19 Find DFT of the sequence x(n) = {1, 2, 3, 0} using DIF algorithm.

Ans



The twiddle factors are $W_4^0 = 1$; $W_4^1 = e^{-j2\pi/4} = -j$

$X(k) = \{6, -2, -2j, 2, -2 + 2j\}$

Q.20 What are the applications of FFT Algorithms ?

Ans The applications of FFT algorithms includes.

(i) Linear filtering, (ii) Correlation, (iii) Spectrum analysis.

EXERCISE

1. Write the equations and draw the signal flow graph for the decimation in frequency algorithm for N =4.
2. Draw the signal flow graph of decimation-in-time algorithm for N = 8.
3. Compute the DFT for N = 4 if

$$x(n) = 1 \qquad 0 \le n \le 3$$

using the decimation-in-frequency algorithm.

4. Compute the DFT of the sequence for N = 4 if

$$x(n) = \sin\frac{n\pi}{2}$$

using decimation-in-time algorithm.

5. Find the DFT of the following sequences using decimation-in-time (DIT) and decimation-in-frequency (DIF) FFT algorithms.

(a) s(n) = {1, 1, 1, 1, 1, 1, 1, 1}

(b) s(n) = {1, 0, 0, 0, 1, 1, 1, 0}

(c) s(n) = {1, 0, 0, 1, –1, 1}

(d) s(n) = {1, 1, 1, 1, 0, 0, 0, 0}

□□□