SIGNALS AND DIGITAL SIGNAL PROCESSING
BEE502

Module – 3 **Fast-Fourier-Transform (FFT) algorithms**: Direct computation of DFT, need for efficient computation of the DFT (FFT algorithms)., speed improvement factor, Radix-2 FFT algorithm for the computation of DFT and IDFT–decimation-in-time and Decimation-in-frequency algorithms, calculation of DFT when N is not a power of 2	
Bloom's Taxonomy Level	L1 – Remembering, L2 – Understanding, L3 – Applying, L – 4 Analysing,

Fast Fourier Transform

Discrete Fourier Transform

- The DFT pair was given as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} \qquad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j(2\pi/N)kn}$$

- Baseline for computational complexity:
  - Each DFT coefficient requires
    - N complex multiplications
    - N-1 complex additions
  - All N DFT coefficients require
    - N^2 complex multiplications
    - N(N-1) complex additions
- Complexity in terms of real operations
  - $4N^2$ real multiplications
  - 2N(N-1) real additions
- Most fast methods are based on symmetry properties
  - Conjugate symmetry $e^{-j(2\pi/N)k(N-n)} = e^{-j(2\pi/N)kN} e^{-j(2\pi/N)k(-n)} = e^{j(2\pi/N)kn}$
  - Periodicity in n and k $e^{-j(2\pi/N)kn} = e^{-j(2\pi/N)k(n+N)} = e^{j(2\pi/N)(k+N)n}$

Fast Fourier Transform

2.2 INTRODUCTION

The N-point DFT of a sequence $x(n)$ converts the time domain N-point sequence $x(n)$ to a frequency domain N-point sequence $X(k)$. The direct computation of an N-point DFT requires N x N complex multiplications and $N(N-1)$ complex additions. Many methods were developed for reducing the number of calculations involved. The most popular of these is the Fast Fourier Transform (FFT), a method developed by Cooley and Turkey. The FFT may be defined as an algorithm (or a method) for computing the DFT efficiently (with reduced number of calculations). The computational efficiency is achieved by adopting a divide and conquer approach. This approach is based on the decomposition of an N-point DFT into successively smaller DFTs and then combining them to give the total transform. Based on this basic approach, a family of computational algorithms were developed and they are collectively known as FFT algorithms. Basically there are two FFT algorithms; Decimation- in-time (DIT) FFT algorithm and Decimation-in-frequency (DIF) FFT algorithm. In this chapter, we discuss DIT FFT and DIF FFT algorithms and the computation of DFT by these methods.

FAST FOURIER TRANSFORM

The DFT of a sequence $x(n)$ of length N is expressed by a complex-valued sequence $X(k)$ as

$$X(K) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, K = 0, 1, 2, \ldots N-1 \text{ where}$$

Let W_N be the complex valued phase factor, which is an N^{th} root of unity given by

$$W_N = e^{-j2\pi nk/N}$$

Thus,

$X(k)$ becomes,

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

Similarly, IDFT is written as

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

From the above equations for $X(k)$ and $x(n)$, it is clear that for each value of k, the direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and $N-1$ complex additions ($4N-2$ real additions). Therefore, to compute all N values of DFT, N^2 complex multiplications and $N(N-1)$ complex additions are required. In fact the DFT and IDFT involve the same type of computations.

If $x(n)$ is a complex-valued sequence, then the N-point DFT given in equation for $X(k)$ can be expressed as

$$X(k) = X_R(k) + jX_I(k)$$

The direct computation of the DFT needs $2N^2$ evaluations of trigonometric functions, $4N^2$ real multiplications and $4N(N-1)$ real additions. Also this is primarily inefficient as it cannot exploit the symmetry and periodicity properties of the phase factor W_N, which are

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

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

FFT algorithm exploits the two symmetry properties and so is an efficient algorithm for DFT computation.

By adopting a divide and conquer approach, a computationally efficient algorithm can be developed. This approach depends on the decomposition of an N-point DFT into successively smaller size DFTs. An N-point sequence, if N can be expressed as $N = r_1 r_2 r_3, \ldots, r_m$, where $r_1 = r_2 = r_3 = \ldots = r_m$, then $N = r^m$, can be decimated into r-point sequences. For each r-point sequence, r-point DFT can be computed. Hence the DFT is of size r. The number r is called the radix of the FFT algorithm and the number m indicates the number of stages in computation. From the results of r-point DFT, the r^2-point DFTs are computed. From the results of r^2-point DFTs, the r^3-point DFTs are computed and so on, until we get r^m-point DFT. If $r = 2$, it is called radix-2 FFT.

DECIMATION IN TIME (DIT) RADIX-2 FFT

In Decimation in time (DIT) algorithm, the time domain sequence $x(n)$ is decimated and smaller point DFTs are computed and they are combined to get the result of N-point DFT.

In general, we can say that, in DIT algorithm the N-point DFT can be realized from two numbers of $N/2$-point DFTs, the $N/2$-point DFT can be realized from two numbers of $N/4$-point DFTs, and so on.

In DIT radix-2 FFT, the N-point time domain sequence is decimated into 2-point sequences and the 2-point DFT for each decimated sequence is computed. From the results of 2-point DFTs, the 4-point DFTs, from the results of 4-point DFTs, the 8-point DFTs and so on are computed until we get N-point DFT.

For performing radix-2 FFT, the value of r should be such that, $N = 2^m$. Here, the decimation can be performed m times, where $m = \log_2 N$. In direct computation of N-point DFT, the total number of complex additions are $N(N-1)$ and the total number of complex multiplications are N^2. In radix-2 FFT, the total number of complex additions are reduced to $N \log_2 N$ and the total number of complex multiplications are reduced to $(N/2) \log_2 N$.

Let $x(n)$ be an N-sample sequence, where N is a power of 2. Decimate or break this sequence into two sequences $f_1(n)$ and $f_2(n)$ of length $N/2$, one composed of the even indexed values of $x(n)$ and the other of odd indexed values of $x(n)$.

Given sequence $x(n) : x(0), x(1), x(2), \ldots . x(\frac{N}{2}-1) \ldots x(N-1)$

Even indexed sequence $f_1(n) = x(2n) : x(0), x(2), x(4), \ldots \ldots x(N-2)$

Odd indexed sequence $f_2(n) = x(2n+1) : x(1), x(3), x(5), \ldots \ldots x(N-1)$

We know that the transform X(k) of the N-point sequence x(n) is given by

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

Breaking the sum into two parts, one for the even and one for the odd indexed values, gives

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

$$X(K) = \sum_{n=even}^{N/2-1} x(n)W_N^{nk} + W_N^{nk} \sum_{n=odd}^{N-1} x(n)W_N^{nk}$$

When n is replaced by 2n, the even numbered samples are selected and when n is replaced by 2n + 1, the odd numbered samples are selected. Hence,

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

Rearranging each part of X(k) into (N/2)-point transforms using

$$W_N^{2nk} = (W_N^2)^{nk} = \left[e^{-j\frac{2\pi}{N}}\right]^{2nk} = W_{N/2}^{nk} \text{ and } W_N^{(2n+1)k} = (W_N^k)W_{N/2}^{nk}$$

We can write

$$X(K) = \sum_{n=0}^{N/2-1} f_1(n)W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} f_2(n)W_{N/2}^{nk}$$

By definition of DFT, the N/2-point DFT of f1(n) and f2(n) is given by

$$F_1(K) = \sum_{n=0}^{N/2-1} f_1(n)W_{N/2}^{nk} \;\&\; F_2(K) + \sum_{n=0}^{N/2-1} f_2(n)W_{N/2}^{nk}$$

$$X(k) = F_1(K) + W_N^k F_2(K), ...k = 0,1,2,3,....N-1$$

The implementation of this equation for X(k) is shown in the following Figure . This first step in the decomposition breaks the N-point transform into two (N/2)-point transforms and the k WN provides the N-point combining algebra. The DFT of a sequence is periodic with period given by the number of points of DFT. Hence, F1(k) and F2(k) will be periodic with period N/2.

$$F_1(k+N/2) = F_1(K), \&F_2(k+N/2) = F_2(K)$$

$$F_1(k+N/2) = F_1(K), \&F_2(k+N/2) = F_2(K)$$

In addition, the phase factor $W_N^{(k+N/2)} = -(W_N^k)$

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

$$X(K) = F_1(k - N/2) - W_N^k F_2(K - N/2)$$

The implementation using the periodicity property is also shown in following Figure

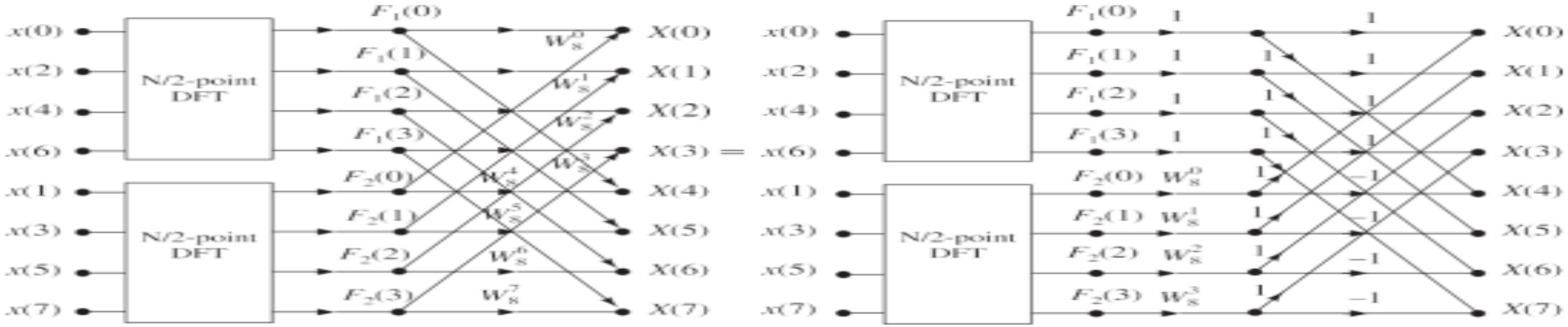


Figure 2.1 Illustration of flow graph of the first stage DIT FFT algorithm for $N = 8$.

Having performed the decimation 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 two $(N/4)$-point sequences and $f_2(n)$ would result in another two $(N/4)$-point sequences.

Decimation-In-Time FFT Algorithms

- Makes use of both symmetry and periodicity
- Consider special case of N an integer power of 2
- Separate x[n] into two sequence of length N/2
  - Even indexed samples in the first sequence
  - Odd indexed samples in the other sequence

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)kn} = \sum_{n\ \text{even}}^{N-1} x[n]e^{-j(2\pi/N)kn} + \sum_{n\ \text{odd}}^{N-1} x[n]e^{-j(2\pi/N)kn}$$

- Substitute variables n=2r for n even and n=2r+1 for odd

$$X[k] = \sum_{r=0}^{N/2-1} x[2r]W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1]W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k]$$

- G[k] and H[k] are the N/2-point DFT's of each subsequence

Decimation In Time

- 8-point DFT example using decimation-in-time
- Two N/2-point DFTs
  - $2(N/2)^2$ complex multiplications
  - $2(N/2)^2$ complex additions
- Combining the DFT outputs
  - N complex multiplications
  - N complex additions
- Total complexity
  - $N^2/2+N$ complex multiplications
  - $N^2/2+N$ complex additions
  - More efficient than direct DFT
- Repeat same process
  - Divide N/2-point DFTs into
  - Two N/4-point DFTs
  - Combine outputs

In-Place Computation

- Decimation-in-time flow graphs require two sets of registers
  - Input and output for each stage
- Note the arrangement of the input indices
  - Bit reversed indexing

$$X_0[0] = x[0] \leftrightarrow X_0[000] = x[000]$$
$$X_0[1] = x[4] \leftrightarrow X_0[001] = x[100]$$
$$X_0[2] = x[2] \leftrightarrow X_0[010] = x[010]$$
$$X_0[3] = x[6] \leftrightarrow X_0[011] = x[110]$$
$$X_0[4] = x[1] \leftrightarrow X_0[100] = x[001]$$
$$X_0[5] = x[5] \leftrightarrow X_0[101] = x[101]$$
$$X_0[6] = x[3] \leftrightarrow X_0[110] = x[011]$$
$$X_0[7] = x[7] \leftrightarrow X_0[111] = x[111]$$

Decimation-In-Frequency FFT Algorithm

- The DFT equation

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

- Split the DFT equation into even and odd frequency indexes

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n2r} = \sum_{n=0}^{N/2-1} x[n] W_N^{n2r} + \sum_{n=N/2}^{N-1} x[n] W_N^{n2r}$$

- Substitute variables to get

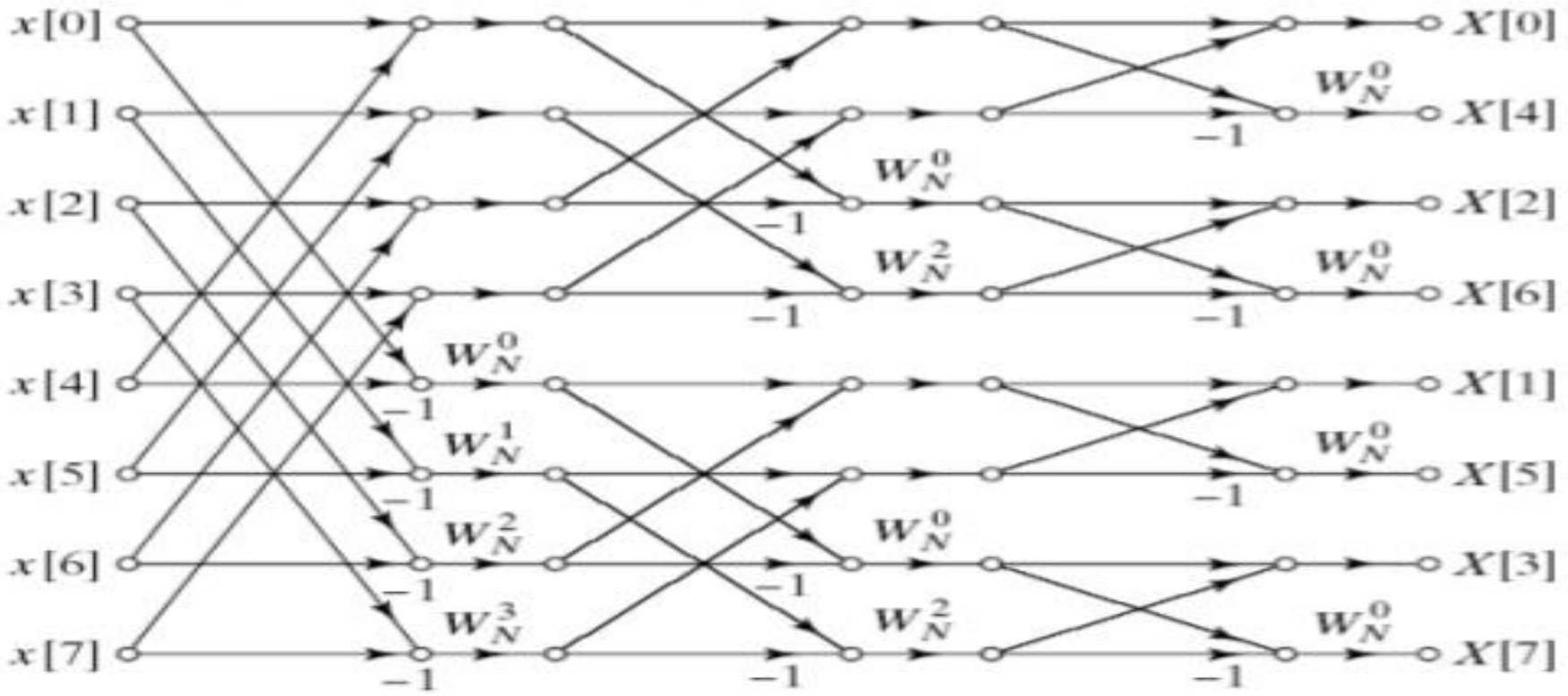
$$X[2r] = \sum_{n=0}^{N/2-1} x[n] W_N^{n2r} + \sum_{n=0}^{N/2-1} x[n+N/2] W_N^{(n+N/2)2r} = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2]) W_{N/2}^{nr}$$

- Similarly for odd-numbered frequencies

$$X[2r+1] = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2]) W_{N/2}^{n(2r+1)}$$

Decimation-In-Frequency FFT Algorithm

- Final flow graph for 8-point decimation in frequency



THE 8-POINT DFT USIKG RADIX-2 DIT FFT

The computation of 8-point DFT using radix-2 FFT involves three stages of computation. Here $N = 8 = 2^3$, therefore, $r = 2$ and $m = 3$. The given 8-point sequence is decimated into four 2-point sequences. For each 2-point sequence, the two point DFT is computed. From the results of four 2-point DFTs, two 4-point DFTs are obtained and from the results of two 4-point DFTs, the 8-point DFT is obtained.

Let the given 8-sample sequence $x(n)$ be $\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$. The 8-samples should be decimated into sequences of two samples. Before decimation they are arranged in bit reversed order as shown in Table 2.1.

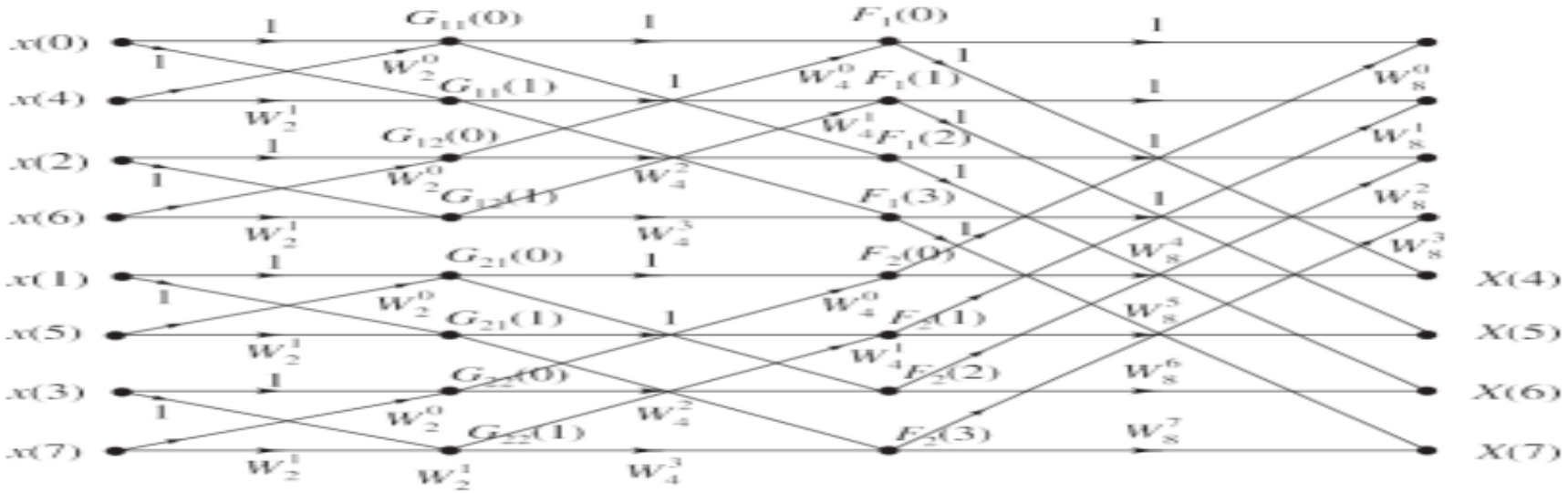
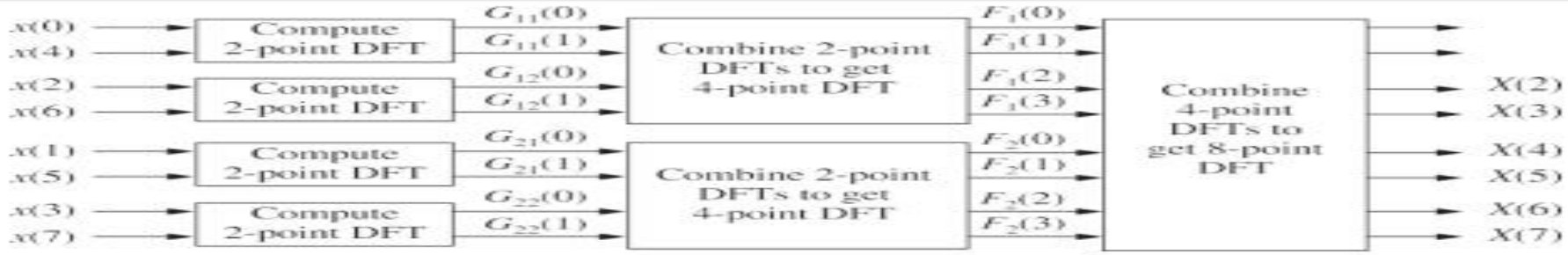


Figure 2.4 Illustration of complete flow graph obtained by combining all the three stages for $N = 8$.



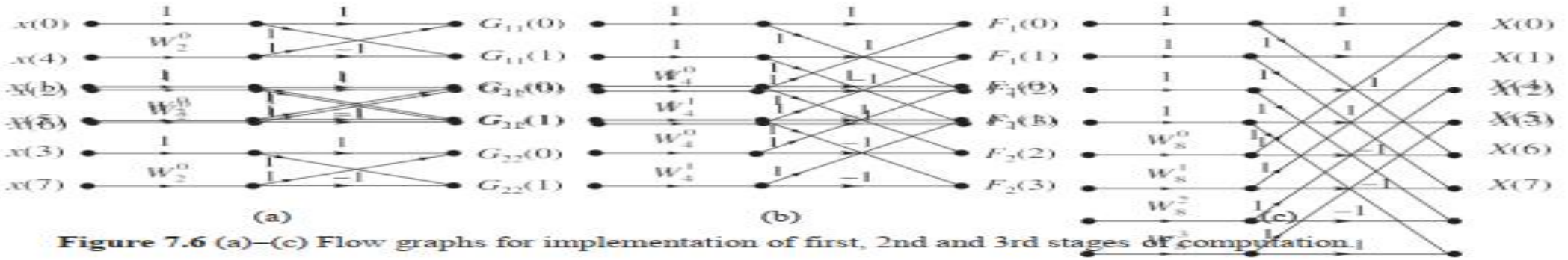


Figure 7.6 (a)–(c) Flow graphs for implementation of first, 2nd and 3rd stages of computation.

Butterfly Diagram

Observing the basic computations performed at each stage, we can arrive at the following conclusions:

(i) In each computation, two complex numbers a and b are considered.
(ii) The complex number b is multiplied by a phase factor W_N^k.
(iii) The product bW_N^k is added to the complex number a to form a new complex number A.
(iv) The product bW_N^k is subtracted from complex number a to form new complex number B.

The above basic computation can be expressed by a signal flow graph shown in Figure 7.7. The signal flow graph is also called butterfly diagram since it resembles a butterfly.

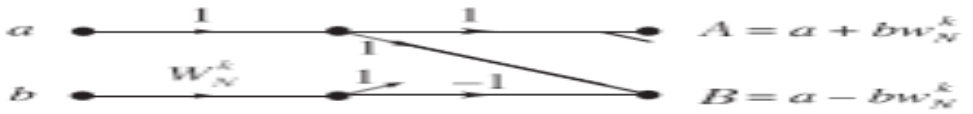


Figure 7.7 Basic butterfFy diagram or fFow graph of radix–2 DI† FF†.

The complete flow graph for 8-point DIT FFT considering periodicity drawn in a way to remember easily is shown in Figure 7.8. In radix-2 FFT, $N/2$ butterflies per stage are required to represent the computational process. In the butterfly diagram for 8-point DFT shown in Figure 7.8, for symmetry, W_2^0, W_4^0 and W_8^0 are shown on the graph eventhough they are unity. The subscript 2 indicates that it is the first stage of computation. Similarly, subscripts 4 and 8 indicate the second and third stages of computation.

$X(0)$

$X(1)$

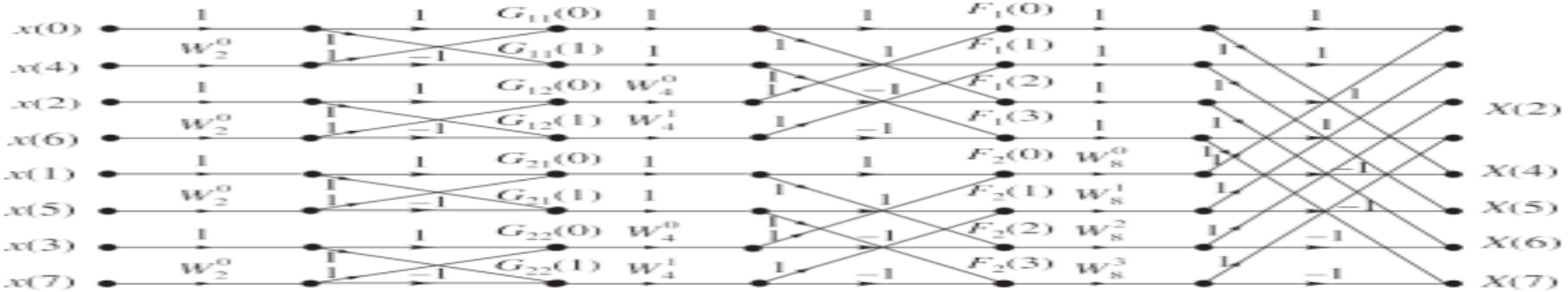


Figure 7.8 †he signal flow graph or butterfly diagram for 8–point radix–2 DIT FFT.

DECIMATION IN FREQUENCY (DIF) RADIX-2 FFT

In decimation in frequency algorithm, the frequency domain sequence $X(k)$ is decimated. In this algorithm, the N-point time domain sequence is converted to two numbers of $N/2$-point sequences. Then each $N/2$-point sequence is converted to two numbers of $N/4$-point sequences. This process is continued until we get $N/2$ numbers of 2-point sequences. Finally, the 2-point DFT of each 2-point sequence is computed. The 2-point DFTs of $N/2$ numbers of 2-point sequences will give N-samples, which is the N-point DFT of the time domain sequence. Here the equations for $N/2$-point sequences, $N/4$-point sequences, etc., are obtained by decimation of frequency domain sequences. Hence this method is called DIF.

To derive the decimation-in-frequency form of the FFT algorithm for N, a power of 2, we can first divide the given input sequence $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)$ into the first half and last half of the points so that its DFT $X(k)$ is

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

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

It is important to observe that while the above equation for $X(k)$ contains two summations over $N/2$-points, each of these summations is not an $N/2$-point DFT, since W_N^{nk} rather than $W^{nkN/2}$

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

$$= \sum_{n=0}^{N/2-1} \left[x(n) W_N^{nk} + (-1)^{nk} x(n+\frac{N}{2}) W_N^{nk} \right]$$

$$= \sum_{n=0}^{N/2-1} \left[x(n) + (-1)^{nk} x(n+\frac{N}{2}) \right] W_N^{nk}$$

Let us split $X(k)$ into even and odd numbered samples. For even values of k, the $X(k)$ can be written as

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

For odd values of k, the X(k) can be written as

$$X(2K+1) = \sum_{n=0}^{N/2-1} \left[x(n) + (-1)^{2k+1} x(n + \frac{N}{2}) W_N^{(2k+1)n} \right]$$
$$= \sum_{n=0}^{N/2-1} \left[x(n) - x(n + \frac{N}{2}) \right] W_N^{nk} W_{N/2}^{nk}$$

The above equations for $X(2k)$ and $X(2k + 1)$ can be recognized as $N/2$-point DFTs. $X(2k)$ is the DFT of the sum of first half and last half of the input sequence, i.e. of $\{x(n) + x(n + N/2)\}$ and $X(2k + 1)$ is the DFT of the product W_N^n with the difference of first half and last half of the input, i.e. of $\{x(n) - x(n + N/2)\} W_N^n$.
If we define new time domain sequences, $u_1(n)$ and $u_2(n)$ consisting of $N/2$-samples, such that

$$u_1(n) = x(n) + x\left(n + \frac{N}{2}\right); \qquad \text{for } n = 0, 1, 2, \ldots, \frac{N}{2} - 1$$

and $$u_2(n) = \left[x(n) - x\left(n + \frac{N}{2}\right)\right] W_N^n ; \quad \text{for } n = 0, 1, 2, \ldots, \frac{N}{2} - 1$$

then the DFTs $U_1(k) = X(2k)$ and $U_2(k) = X(2k + 1)$ can be computed by first forming the sequences $u_1(n)$ and $u_2(n)$, then computing the $N/2$-point DFTs of these two sequences to obtain the even numbered output points and odd numbered output points respectively. The procedure suggested above is illustrated in Figure 7.9 for the case of an 8-point sequence.

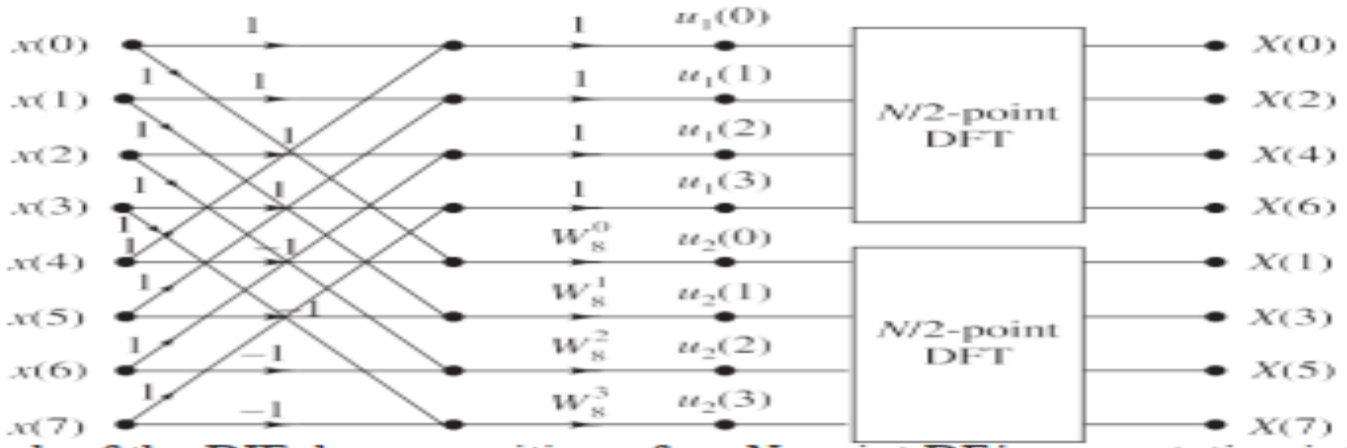


Figure 7.9 FFow graph of the DIF decomposition of an N–point DF† computation into two $N/2$–point DF† computations $N = 8$.

into two numbers of $N/4$-point sequences and four numbers of new $N/4$-point sequences can be obtained from them.

Let the new sequences be $v_{11}(n)$, $v_{12}(n)$, $v_{21}(n)$, $v_{22}(n)$. On similar lines as discussed above, we can get

$$v_{11}(n) = u_1(n) + u_1(n+2); \quad \text{for } n = 0, 1, 2, \ldots, \frac{N}{4} - 1$$

$$v_{12}(n) = [u_1(n) - u_1(n+2)]W_{N/2}^n; \quad \text{for } n = 0, 1, 2, \ldots, \frac{N}{4} - 1$$

$$v_{21}(n) = u_2(n) + u_2(n+2); \quad \text{for } n = 0, 1, 2, \ldots, \frac{N}{4} - 1$$

$$v_{22}(n) = [u_2(n) - u_2(n+2)]W_{N/2}^n; \quad \text{for } n = 0, 1, 2, \ldots, \frac{N}{4} - 1$$

This process is continued till we get only 2-point sequences. The DFT of those 2-point sequences is the DFT of $x(n)$, i.e. $X(k)$ in bit reversed order.

The third stage of computation for $N = 8$ is shown in Figure 7.11.

The entire process of decimation involves m stages of decimation where $m = \log_2 N$. The computation of the N-point DFT via the DIF FFT algorithm requires $(N/2)\log_2 N$ complex multiplications and $(N - 1)\log_2 N$ complex additions (i.e. total number of computations remains same in both DIF and DIT).

Observing the basic calculations, each stage involves $N/2$ butterflies of the type shown in Figure 7.12.

The butterfly computation involves the following operations:

(i) In each computation two complex numbers a and b are considered.
(ii) The sum of the two complex numbers is computed which forms a new complex number A.
(iii) Subtract the complex number b from a to get the term $(a - b)$. The difference term $(a - b)$ is multiplied with the phase factor or twiddle factor W_N^n to form a new complex number B.

Figure 7.12 Basic butterFy diagram for DIF FI†.

The signal flow graph or butterfly diagram of all the three stages together is shown in Figure 7.13.

TKE 8-POINT DFT USIKG RADIX-2 DIF FFT

The DIF computations for an 8-sample sequence are given below in detail.

Let $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$ be the given 8-sample sequence.

First stage of COMPUTATION

In the first stage of computation, two numbers of 4-point sequences $u_1(n)$ and $u_2(n)$ are obtained from the given 8-point sequence $x(n)$ as shown below.

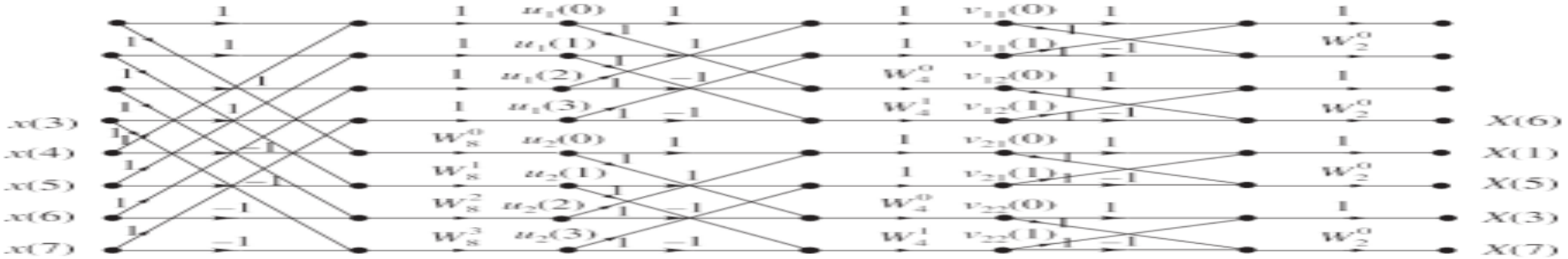


Figure 7.13 SignaF fFow graph or butterfFy diagram for the 8–point radix–2 DIF FF† aFgorithm.

EXAMPLE 1 Draw the butterfly line diagram for 8-point FFT calculation and briefly explain. Use decimation-in-time algorithm.

Solution: The butterfly line diagram for 8-point DIT FFT algorithm is shown in following Figure

Solution: For 8-point DIT FFT

1. The input sequence $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$.
2. bit reversed order, of input as i.e. as $x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$. Since $N = 2^m = 2^3$, the 8-point DFT computation
3. Radix-2 FFT involves 3 stages of computation, each stage involving 4 butterflies. The output $X(k)$ will be in normal order.
4. In the first stage, four 2-point DFTs are computed. In the second stage they are combined into two 4-point DFTs. In the third stage, the two 4-point DFTs are combined into one 8-point DFT.
5. The 8-point FFT calculation requires $8 \log_2 8 = 24$ complex additions and $(8/2) \log_2 8 = 12$ complex multiplications.

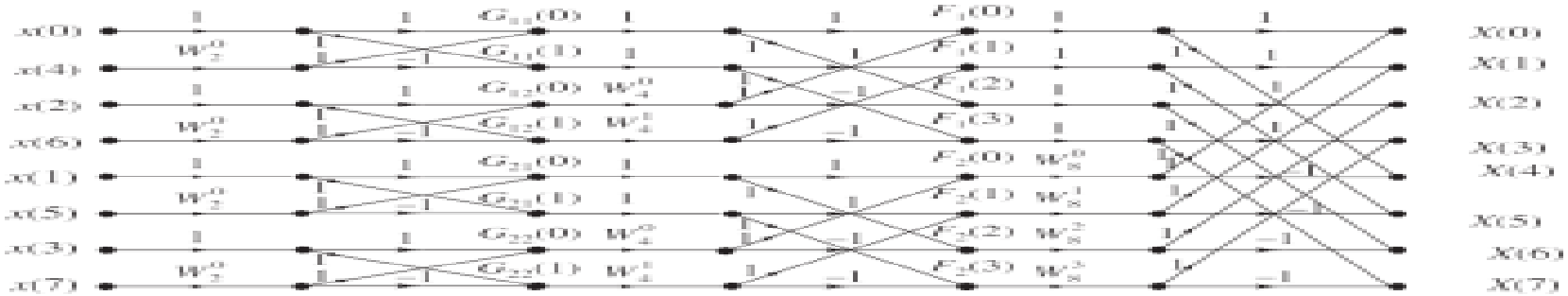


Figure : Butterfly Fine diagram for 8–point DIT FFT algorithm for $N = 8$.

EXAMPLE 2 Implement the decimation-in-frequency FFT algorithm of N-point DFT where $N = 8$. Also explain the steps involved in this algorithm.

Solution: The 8-point radix-2 DIF FFT algorithm

1. It involves 3 stages of computation. The input to the first stage is the input time sequence $x(n)$ in normal order. The output of first stage is the input to the second stage and the output of second stage is the input to the third stage. The output of third stage is the 8-point DFT in bit reversed order.
2. In DIF algorithm, the frequency domain sequence $X(k)$ is decimated.
3. In this algorithm, the N-point time domain sequence is converted to two numbers of $N/2$-point sequences. Then each $N/2$-point sequence is converted to two numbers of $N/4$-point sequences. Thus, we get 4 numbers of $N/4$, i.e. 2-point sequences.
4. Finally, the 2-point DFT of each 2-point sequence is computed. The 2-point DFTs of $N/2$ number of 2-point sequences will give N-samples which is the N-point DFT of the time domain sequence. The implementation of the 8-point radix-2 DIF FFT algorithm is shown in Figure 7.16.

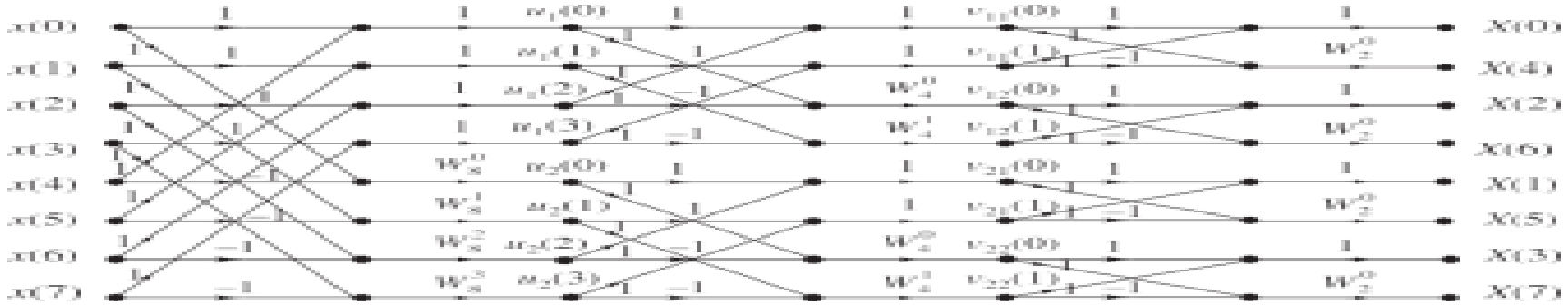


Figure 7.16 Butterfly Fine diagram for 8–point radix–2 DIF FFT algorithm.

EXAMPLE 7.4 What is FFT? Calculate the number of multiplications needed in the calculation of DFT using FFT algorithm with 32-point sequence.

Solution: The FFT, i.e. Fast Fourier transform is a method (or algorithm) for computing the DFT with reduced number of calculations. The computational efficiency is achieved by adopting a divide and conquer approach. This approach is based on the decomposition of an N-point DFT into successively smaller DFTs. This basic approach leads to a family of efficient computational algorithms known as FFT algorithms. Basically there are two FFT algorithms. (i) DIT FFT algorithm and (ii) DIF FFT algorithm. If the length of the sequence $N = 2^m$, 2 indicates the radix and m indicates the number of stages in the computation. In radix-2 FFT, the N-point sequence is decimated into two $N/2$-point sequences, each $N/2$-point sequence is decimated into two $N/4$-point sequences and so on till we get two point sequences. The DFTs of two point sequences are computed and DFTs of two 2-point sequences are combined into DFT of one 4-point sequence, DFTs of two 4-point sequences are combined into DFT of one 8-point sequence and so on till we get the N-point DFT.

The number of multiplications needed in the computation of DFT using FFT algorithm with $N = 32$-point sequence is $= \frac{N}{2}\log_2 N = \frac{32}{2}\log_2 2^5 = 80$.

The number of complex additions $= N\log_2 N = 32\log_2 32 = 32\log_2 2^5 = 160$

EXAMPLE 7.5 Explain the inverse FFT algorithm to compute inverse DFT of a 8-point DFT. Draw the flow graph for the same.

Solution: The IDFT of an 8-point sequence $\{X(k), k = 0, 1, 2, \ldots, 7\}$ is defined as

$$x(n) = \frac{1}{8}\sum_{k=0}^{7} X(k)\, W_8^{-nk}, \quad n = 0, 1, 2, \ldots, 7$$

Taking the conjugate of the above equation for $x(n)$, we have

$$x^*(n) = \frac{1}{8}\left[\sum_{k=0}^{7} X^*(k) W_8^{nk}\right]$$

Taking the conjugate of the above equation for $x^*(n)$ we have

$$x(n) = \frac{1}{8}\left[\sum_{k=0}^{7} X^*(k) W_8^{nk}\right]^*$$

The term inside the square brackets in the RHS of the above expression for $x(n)$ is the 8-point DFT of $X^*(k)$. Hence, in order to compute the IDFT of $X(k)$ the following procedure can be followed:

1. Given $X(k)$, take conjugate of $X(k)$ i.e. determine $X^*(k)$.
2. Compute the DFT of $X^*(k)$ using radix-2 DIT or DIF FFT. [This gives $8x^*(n)$]

1. Take conjugate of output sequence of FFT. This gives $8x(n)$.
2. Divide the sequence obtained in step 3 by 8. The resultant sequence is $x(n)$.

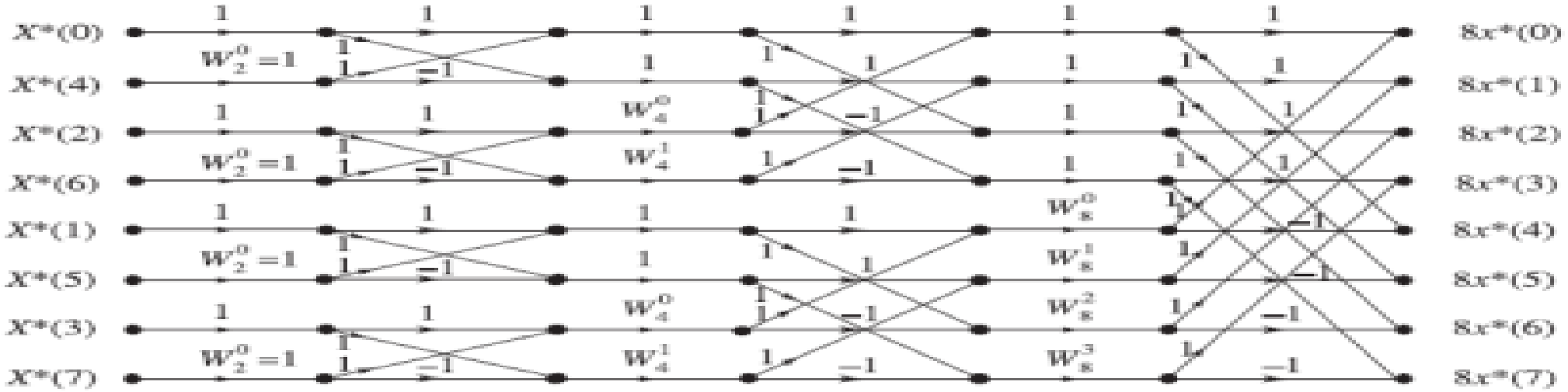


Figure 7.18 Computation of 8–point DF† of $X^*(k)$ by radix–2, DI† FF†.

From Figure 7.18, we get the 8-point DFT of $X^*(k)$ by DIT FFT as

$$8x^*(n) = \{8x^*(0), 8x^*(1), 8x^*(2), 8x^*(3), 8x^*(4), 8x^*(5), 8x^*(6), 8x^*(7)\}$$

$$x(n) = \frac{1}{8}\{8x^*(0), 8x^*(1), 8x^*(2), 8x^*(3), 8x^*(4), 8x^*(5), 8x^*(6), 8x^*(7)\}^*$$

EXAMPLE 7.11 Compute the DFT of the sequence $x(n) = \{1, 0, 0, 0, 0, 0, 0, 0\}$ (a) directly, (b) by FFT.

Solution: (a) Direct computation of DFT

The given sequence is $x(n) = \{1, 0, 0, 0, 0, 0, 0, 0\}$. We have to compute 8-point DFT. So $N = 8$.

$$\text{DFT}\{x(n)\} = X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2}{N}nk} = \sum_{n=0}^{N-1} x(n)\, W_N^{nk} = \sum_{n=0}^{7} x(n)\, W_8^{nk}$$

$$= x(0)W_8^0 + x(1)W_8^1 + x(2)W_8^2 + x(3)W_8^3 + x(4)W_8^4 + x(5)W_8^5 + x(6)W_8^6 + x(7)W_8^7$$

$$= (1)(1) + (0)(W_8^1) + (0)W_8^2 + (0)W_8^3 + (0)W_8^4 + (0)W_8^5 + (0)W_8^6 + (0)W_8^7 = 1$$

$X(k) = 1$ for all k

$X(0) = 1, X(1) = 1, X(2) = 1, X(3) = 1, X(4) = 1, X(5) = 1, X(6) = 1, X(7) = 1$

$X(k) = \{1, 1, 1, 1, 1, 1, 1, 1\}$

(b) Computation by FFT. Here $N = 8 = 2^3$

The computation of 8-point DFT of $x(n) = \{1, 0, 0, 0, 0, 0, 0, 0\}$ by radix-2 DIT FFT algorithm is shown in Figure 7.31. $x(n)$ in bit reverse order is

$$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$$

$$= \{1, 0, 0, 0, 0, 0, 0, 0\}$$

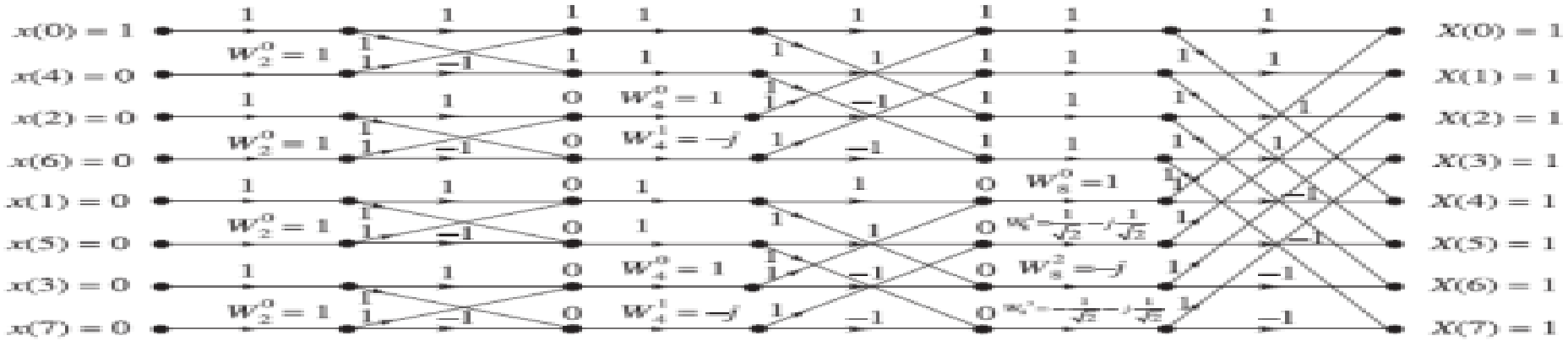
For DIT FFT input is in bit reversed order and output is in normal order.

From Figure 7.31, the 8-point DFT of the given $x(n)$ is $X(k) = \{1, 1, 1, 1, 1, 1, 1, 1\}$

EXAMPLE 7.12 An 8-point sequence is given by $x(n) = \{2, 2, 2, 2, 1, 1, 1, 1\}$. Compute the 8-point DFT of $x(n)$ by

(a) Radix-2 DIT FFT algorithm
(b) Radix-2 DIF FFT algorithm

Also sketch the magnitude and phase spectrum.



Solution: (a) 8-point DFT by Radix-2 DIT FFT algorithm

The given sequence is $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$

$= \{2, 2, 2, 2, 1, 1, 1, 1\}$

The given sequence in bit reversed order is

$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$

$= \{2, 1, 2, 1, 2, 1, 2, 1\}$

For DIT FFT, the input is in bit reversed order and the output is in normal order. The computation of 8-point DFT of $x(n)$, i.e. $X(k)$ by Radix-2 DIT FFT algorithm is shown in Figure 7.32.

From Figure 7.32, we get the 8-point DFT of $x(n)$ as

$$X(k) = \{12, 1 - j2.414, 0, 1 - j0.414, 0, 1 + j0.414, 0, 1 + j2.414\}$$

(b) 8-point DFT by radix-2 DIF FFT algorithm

For DIF FFT, the input is in normal order and the output is in bit reversed order. The computation of DFT by radix-2 DIF FFT algorithm is shown in Figure 7.33.

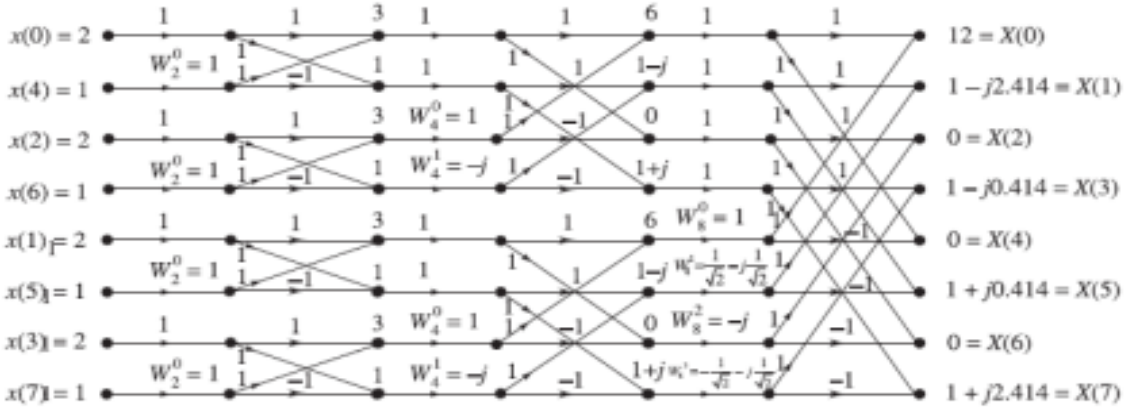


Figure 7.33 Computation of 8–point DF↑ of $x(n)$ by radix–2 DIF FF↑ aFgorithm.

From Figure 7.33, we observe that the 8-point DFT in bit reversed order is

$$X_r(k) = \{X(0), X(4), X(2), X(6), X(1), X(5), X(3), X(7)\}$$
$$= \{12, 0, 0, 0, 1 - j2.414, 1 + j0.414, 1 - j0.414, 1 + j2.414\}$$

The 8-point DFT in normal order is

$$X(k) = \{X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7)\}$$
$$= \{12, 1 - j2.414, 0, 1 - j0.414, 0, 1 + j0.414, 0, 1 + j2.414\}$$

$$X(k) = \{12, 1 - j2.414, 0, 1 - j0.414, 0, 1 + j0.414, 0, 1 + j2.414\}$$
$$= \{12 \angle 0°, 2.61 \angle -67.5°, 0 \angle 0°, 1.08 \angle -22.5°, 0 \angle 0°, 1.08 \angle 22.5°, 0 \angle 0°, 2.61 \angle 67.5°\}$$
$$= \{12 \angle 0, 2.61 \angle -0.37, 0 \angle 0, 1.08 \angle -0.12, 0 \angle 0, 1.08 \angle 0.12, 0 \angle 0, 2.61 \angle 0.37\}$$

$$|X(k)| = \{12, 2.61, 0, 1.08, 0, 1.08, 0, 2.61\}$$

$$\angle X(k) = \{0, -0.37, 0, -0.12, 0, 0.12, 0, 0.37\}$$

The magnitude and phase spectrum are shown in Figures 7.34(a) and (b).

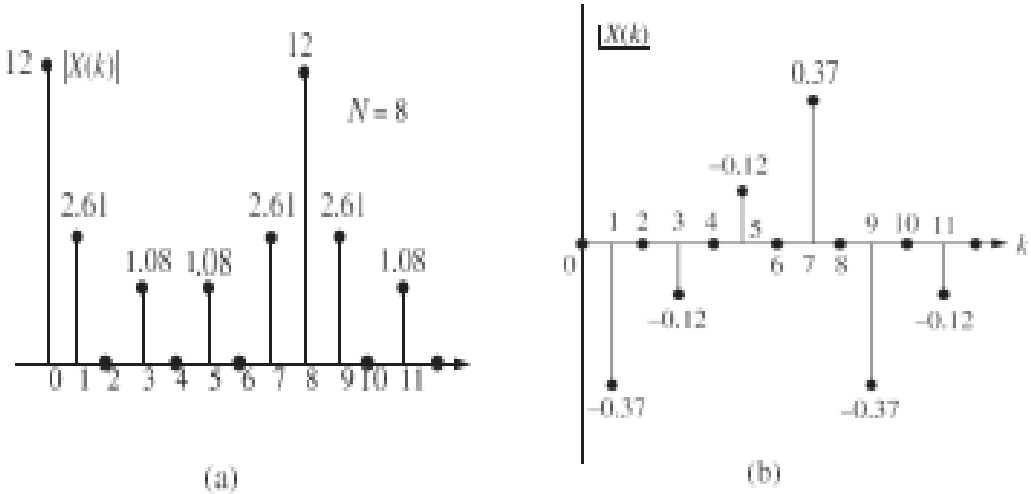


Figure 7.34 (a) Magnitude spectrum, (b) Phase spectrum.

EXAMPLE 7.13 Find the 8-point DFT by radix-2 DIT FFT algorithm.

$$x(n) = \{2, 1, 2, 1, 2, 1, 2, 1\}$$

Solution: The given sequence is $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$

$$= \{2, 1, 2, 1, 2, 1, 2, 1\}$$

For DIT FFT computation, the input sequence must be in bit reversed order and the output sequence will be in normal order.

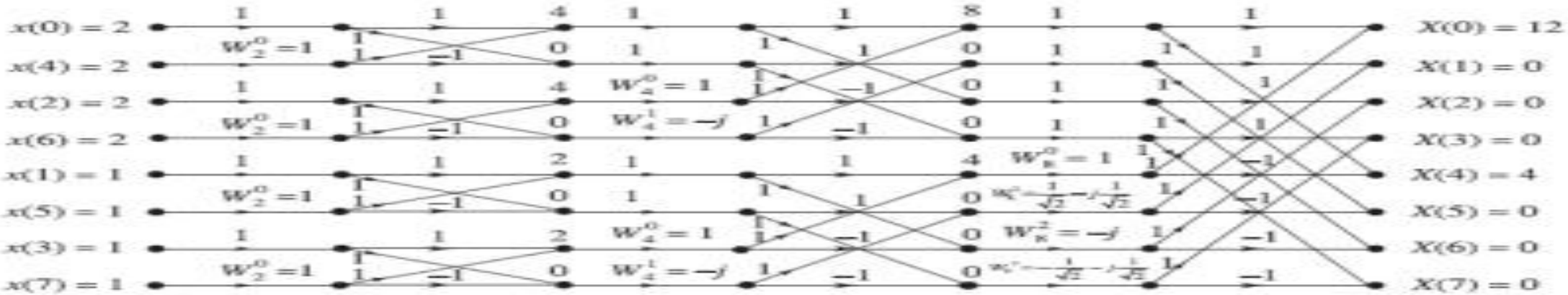
$x(n)$ in bit reverse order is

$$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$$

$$= \{2, 2, 2, 2, 1, 1, 1, 1\}$$

The computation of 8-point DFT of $x(n)$ by radix-2 DIT FFT algorithm is shown in Figure 7.35.

From Figure 7.35, we get the 8-point DFT of $x(n)$ as $X(k) = \{12, 0, 0, 0, 4, 0, 0, 0\}$



EXAMPLE 7.14 Compute the DFT for the sequence $x(n) = \{1, 1, 1, 1, 1, 1, 1, 1\}$.

Solution: The given sequence is $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$
$= \{1, 1, 1, 1, 1, 1, 1, 1\}$

The computation of 8-point DFT of $x(n)$, i.e. $X(k)$ by radix-2, DIT FFT algorithm is shown in Figure 7.36.

The given sequence in bit reversed order is

$$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$$
$$= \{1, 1, 1, 1, 1, 1, 1, 1\}$$

For DIT FFT, the input is in bit reversed order and output is in normal order.

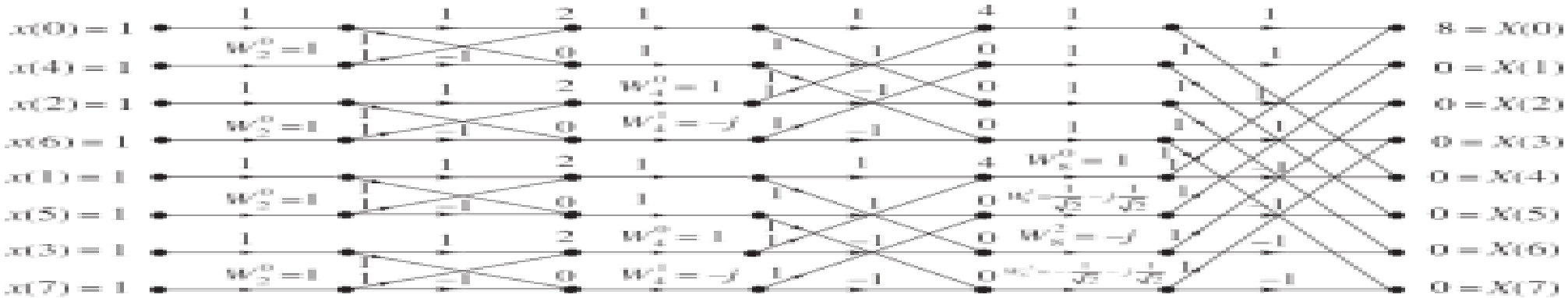


Figure 7.36 Computation of 8–point DFT of $x(n)$ by radix–2, DIT FFT.

From Figure 7.36, we get the 8-point DFT of $x(n)$ as $X(k) = \{8, 0, 0, 0, 0, 0, 0, 0\}$.

EXAMPLE 7.15 Given a sequence $x(n) = \{1, 2, 3, 4, 4, 3, 2, 1\}$, determine $X(k)$ using DIT FFT algorithm.

Solution: The given sequence is $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$
$= \{1, 2, 3, 4, 4, 3, 2, 1\}$

The computation of 8-point DFT of $x(n)$, i.e. $X(k)$ by radix-2, DIT FFT algorithm is shown in Figure 7.37. For DIT FFT, the input is in bit reversed order and the output is in normal order.

The given sequence in bit reverse order is

$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\} = \{1, 4, 3, 2, 2, 3, 4, 1\}$

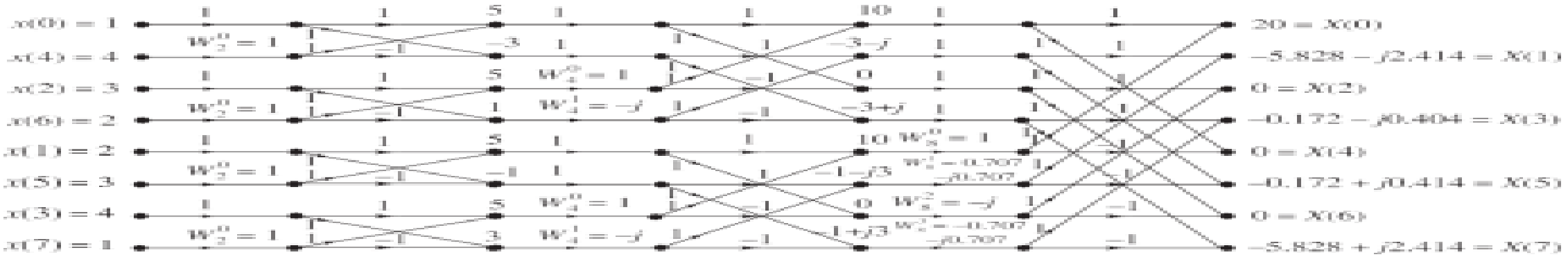


Figure 7.37 Computation of 8–point DFT of $x(n)$ by radix–2, DIT FFT.

From Figure 7.37, we get the 8-point DFT of $x(n)$ as

$X(k) = \{20, -5.828 - j2.414, 0, -0.172 - j0.414, 0, -0.172 + j0.414, 0, -5.828 + j2.414\}$

EXAMPLE 7.16 Given a sequence $x(n) = \{0, 1, 2, 3, 4, 5, 6, 7\}$, determine $X(k)$ using DIT FFT algorithm.

Solution: The given sequence is $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$

$= \{0, 1, 2, 3, 4, 5, 6, 7\}$

The computation of 8-point DFT of $x(n)$, i.e. $X(k)$ by radix-2, DIT FFT algorithm is shown in Figure 7.38. For DIT FFT, the input is in bit reversed order and output is in normal order.

The given sequence in bit reverse order is

$x_r(n) = \{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$

$= \{0, 4, 2, 6, 1, 5, 3, 7\}$

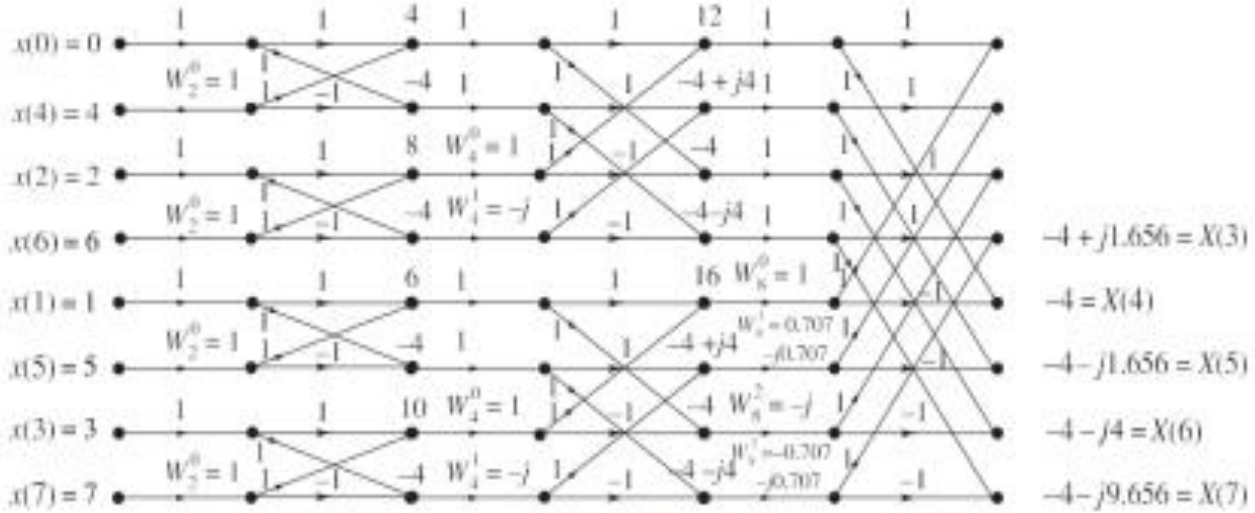


Figure 7.38 Computation of 8–point DF↑ of $x(n)$ by radix–2, DI↑ FF↑.

From Figure 7.38, we get the 8-point DFT of $x(n)$ as

$X(k) = \{28, \square\ 4 + j9.656, \square\ 4 + \square\ 4 + \square\ \square\ 4 \square\ j1.656, \square\ 4 \square\ j4, \square\ j9.656\}$
$j4, \quad j1.656, \quad 4, \quad \square\ 4$

$8x^*(n)$, taking the conjugate of that to get $8x(n)$ and then dividing the result by 8 to get $x(n)$. For DIF algorithm, input $X^*(k)$ must be in normal order. The output will be in bit reversed order for the given $X(k)$.

$$X^*(k) = \{4,\ 1 + j2.414,\ 0,\ 1 + j0.414,\ 0,\ 1 - j0.414,\ 0,\ 1 - j2.414\}$$

The DFT of $X^*(k)$ using radix-2, DIF FFT algorithm is computed as shown in Figure 7.42.

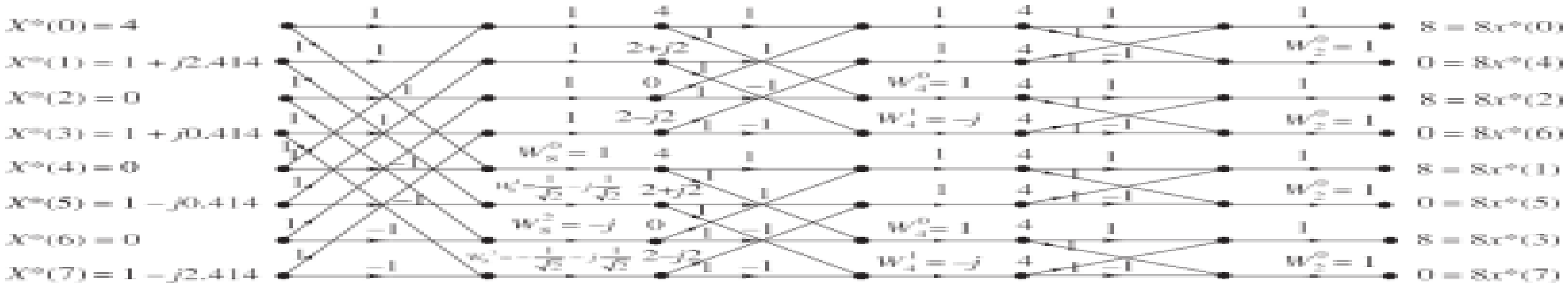


Figure 7.42 Computation of 8–point DFT of $X^*(k)$ by radix–2 DIF FFT.

From the DIF FFT algorithm of Figure 7.42, we get

$$8x_r^*(n) = \{8,\ 0,\ 8,\ 0,\ 8,\ 0,\ 8,\ 0\}$$

$$8x_r(n) = \{8,\ 0,\ 8,\ 0,\ 8,\ 0,\ 8,\ 0\}^* = \{8,\ 0,\ 8,\ 0,\ 8,\ 0,\ 8,\ 0\}$$

$$x(n) = \frac{1}{8}\{8,\ 8,\ 8,\ 8,\ 0,\ 0,\ 0,\ 0\} = \{1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0\}$$