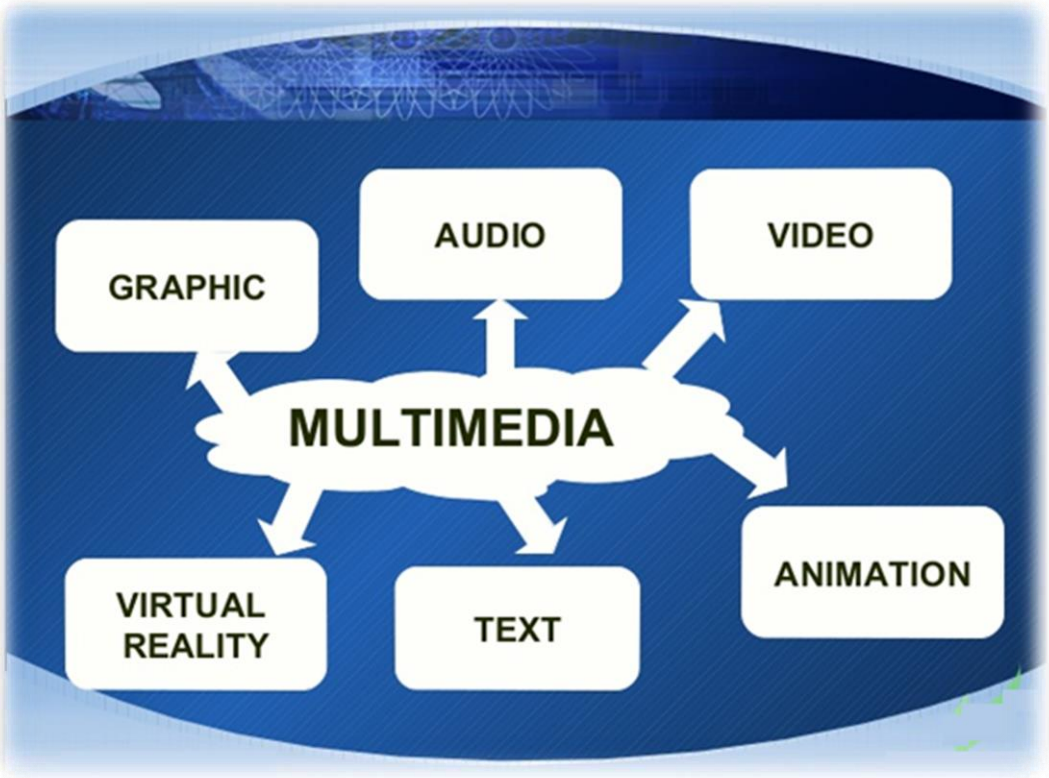

Tikrit University

Computer Science Dept.

Master Degree

Lecture 10

Asst.Prof.Dr.Eng.Zaidoon.T.AL-Qaysi

Fast Foureir Transform

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 $N \times N$ 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.

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

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

Let W_N be the complex valued phase factor, which is an Nth root of unity given by

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

Thus, $X(k)$ becomes

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

Similarly, IDFT is written as

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

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

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

or

$$X(k) = \sum_{n=0}^{N-1} \left[x_R(n) + jx_I(n)\right]\left[\cos\frac{2\pi nk}{N} - j\sin\frac{2\pi nk}{N}\right]$$

Equating the real and imaginary parts of the above two equations for $X(k)$ we have

$$X_R(k) = \sum_{n=0}^{N-1} \left[x_R(n) \cos \frac{2\pi nk}{N} + x_I(n) \sin \frac{2\pi nk}{N} \right]$$

$$X_I(k) = -\sum_{n=0}^{N-1} \left[x_R(n) \sin \frac{2\pi nk}{N} - x_I(n) \cos \frac{2\pi nk}{N} \right]$$

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 $W_N^{k+\frac{N}{2}} = -W_N^k$

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

FFT algorithm exploits the above 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

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 $x(n)$: $x(0), x(1), x(2), \ldots, x\left(\frac{N}{2}-1\right), \ldots, x(N-1)$

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

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

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

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

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

$$X(k) = \sum_{n \text{ even}}^{N-2} x(n)\, W_N^{nk} + \sum_{n \text{ odd}}^{N-1} x(n) W_N^{nk}, \qquad k = 0, 1, 2, \ldots, N-1$$

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,

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

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

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

and $W_N^{(2n+1)k} = W_N^k W_{N/2}^{nk}$, we can write

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

By definition of DFT, the $N/2$-point DFT of $f_1(n)$ and $f_2(n)$ is given by

$$F_1(k) = \sum_{n=0}^{\frac{N}{2}-1} f_1(n) W_{N/2}^{kn} \quad \text{and} \quad F_2(k) = \sum_{n=0}^{\frac{N}{2}-1} f_2(n) W_{N/2}^{kn}$$

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0, 1, 2, \ldots, N-1$$

Butterfly Diagram

In butterfly diagram the left hand side is time and the right hand side is frequency. In decimation in time, because the decomposition is being done in time, the left hand side which is denoted for time is to be written as in even and odd sequence, while right hand side is frequency, it is written in the same sequence i.e. no order is changed. An efficient algorithm for computing the DFT is developed for cases in which the number of samples to be computed is a power of 2 ($N = 2^m$). Where, m is called the stage of the butterfly diagram.

For drawing two point butterfly diagram a power of 2 ($N = 2^m$). Where m is called the stage of the butterfly diagram, it means in drawing two point butterfly 2^1. Only one stage is required. We try to generalize the process. The result of the effort is known as the DIT, radix-2 FFT.

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

For calculating a two-point DFT, it does not need to be decimated in its even and odd components

$$X[k] = \sum_{n=0}^{1} x(n) W_2^{nk}$$
$$X[k] = x(0) W_2^{0k} + x(1) W_2^{1k}$$
$$X[0] = x(0) W_2^{0} + x(1) W_2^{(1)(0)}$$
$$X[1] = x(0) W_2^{0} + x(1) W_2^{1}$$

Because $W_2^{0k} = e^{-j0} = 1$ and $W_2^{1k} = e^{-j\pi k} = (-1)^k$
The following can be

$$X[0] = x(0) + x(1)$$
$$X[1] = x(0) - x(1)$$

The signal flow graph of **Figure 1** illustrates the process for computing the two-point DFT. This signal flow graph is known as a butterfly diagram because of its shape.

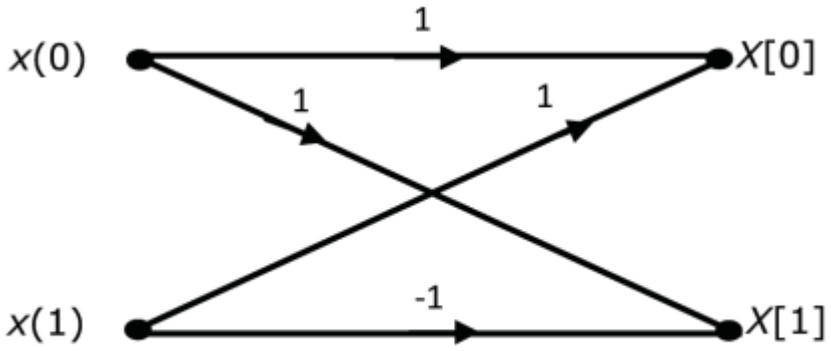

Figure 1: Butterfly diagram for a 2-point DFT.

The four point DFT is carried out from the generalized formula as

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

It is decimated into $N/2$ point even and odd components, we use the concept that DFT is periodic.

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

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

$$X[k] = G(k) + W_N^k \; H(k)$$

Substituting the value of k = 0–3

$$X[0] = G[0] + W_4^0 \; H[0]$$
$$X[1] = G[1] + W_4^1 \; H[1]$$
$$X[2] = G[2] + W_4^2 \; H[2] = G[0] + W_4^2 \; H[0]$$
$$X[3] = G[3] + W_4^3 \; H[3] = G[1] + W_4^3 \; H[1]$$

The Equation can be written for $N/2$ even components of four-point DFT

$$G[k] = \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} = \sum_{n=0}^{1} x(2n) W_2^{nk}$$

$$G[k] = x(0)W_2^{0k} + x(2)W_2^{1k}$$
$$G[0] = x(0)W_2^{0} + x(2)W_2^{0}$$
$$G[1] = x(0)W_2^{0} + x(2)W_2^{1}$$

The Equation can be written for $N/2$ odd components of four-point DFT

$$H[k] = \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{nk} = \sum_{n=0}^{1} x(2n+1)W_2^{nk}$$
$$H[k] = x(1)W_2^{0k} + x(3)W_2^{1k}$$
$$H[0] = x(1)W_2^{0} + x(3)W_2^{0}$$
$$H[1] = x(1)W_2^{0} + x(3)W_2^{1}$$

It is to be noted that four-point DFT can be computed by the generation of two two-point DFTs followed by a re-composition of terms as shown in the signal flow graph of **Figure 2.**

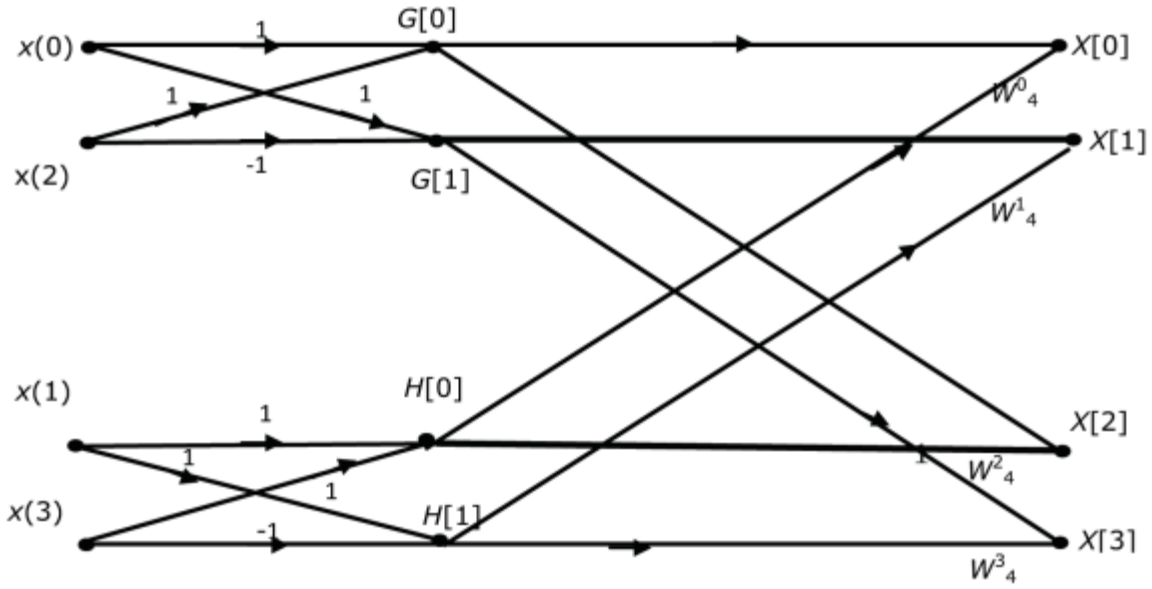

Figure 2: Signal flow graph for a four-point DFT.

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.

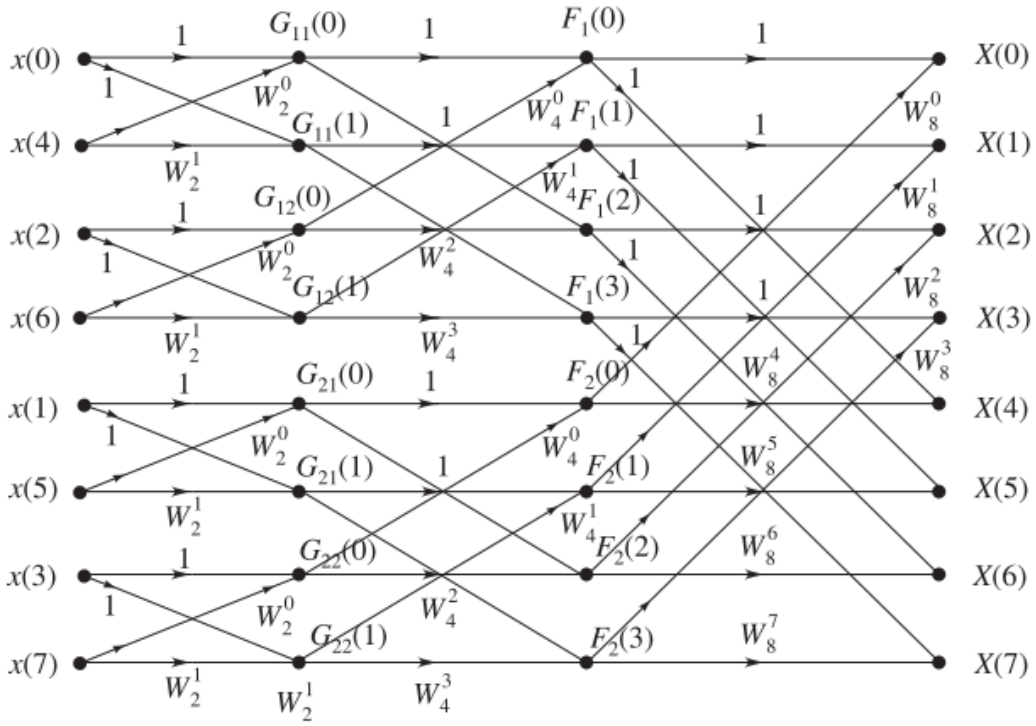

Figure 3: Illustration of complete flow graph obtained by combining all the three stages for $N = 8$.

Table 1: Normal and bit reversed order for $N = 8$.

Normal order		*Bit reversed order*	
$x(0)$	$x(000)$	$x(0)$	$x(000)$
$x(1)$	$x(001)$	$x(4)$	$x(100)$
$x(2)$	$x(010)$	$x(2)$	$x(010)$
$x(3)$	$x(011)$	$x(6)$	$x(110)$
$x(4)$	$x(100)$	$x(1)$	$x(001)$
$x(5)$	$x(101)$	$x(5)$	$x(101)$
$x(6)$	$x(110)$	$x(3)$	$x(011)$
$x(7)$	$x(111)$	$x(7)$	$x(111)$

Using the decimated sequences as input, the 8-point DFT is computed. **Figure 4** shows the three stages of computation of an 8-point DFT.

The computation of 8-point DFT of an 8-point sequence in detail is given below. The 8-point sequence is decimated into 4-point sequences and 2-point sequences as shown below.

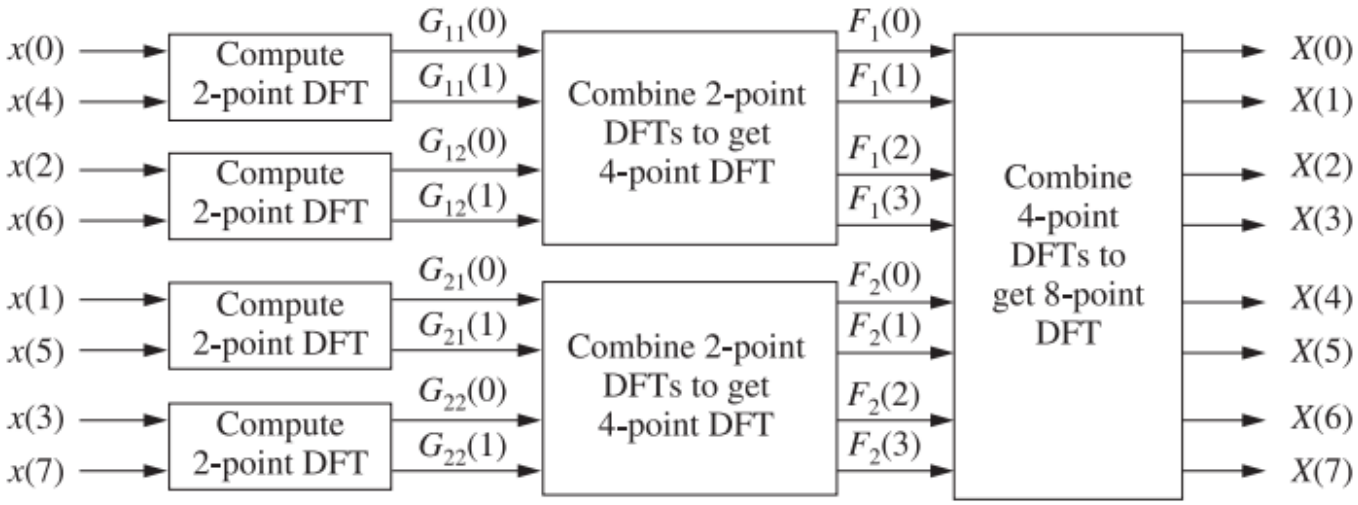

Figure 4: Three stages of computation in 8-point DFT.

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

$f_1(n) = \{x(0), x(2), x(4), x(6)\}$, $f_2(n) = \{x(1), x(3), x(5), x(7)\}$ 4-point sequences obtained from $x(n)$.

$g_{11}(n) = \{x(0), x(4)\}$, $g_{12}(n) = \{x(2), x(6)\}$, 2-point sequences obtained from $f_1(n)$

$g_{21}(n) = \{x(1), x(5)\}$, $g_{22}(n) = \{x(3), x(7)\}$, 2-point sequences obtained from $f_2(n)$

The relations between the samples of various sequences are given below.

$$g_{11}(0) = f_1(0) = x(0) \qquad g_{21}(0) = f_2(0) = x(1)$$
$$g_{11}(1) = f_1(2) = x(4) \qquad g_{21}(1) = f_2(2) = x(5)$$
$$g_{12}(0) = f_1(1) = x(2) \qquad g_{22}(0) = f_2(1) = x(3)$$
$$g_{12}(1) = f_1(3) = x(6) \qquad g_{22}(1) = f_2(3) = x(7)$$

The first stage of computation

In the first stage of computation, 2-point DFTs of the 2-sample sequences are computed.

Let $G_{11}(k) = \text{DFT}\{g_{11}(n)\}$. The 2-point DFT of $g_{11}(n)$ is given by

$$G_{11}(k) = \sum_{n=0}^{1} g_{11}(n) W_{N/4}^{nk}; \quad \text{for } k = 0, 1$$

$$\therefore \quad G_{11}(0) = g_{11}(0) W_2^0 + g_{11}(1) W_2^0 = g_{11}(0) + g_{11}(1) = x(0) + x(4) = x(0) + x(4) W_2^0$$

$$G_{11}(1) = g_{11}(0) W_2^0 + g_{11}(1) W_2^1 = g_{11}(0) - g_{11}(1) = x(0) - x(4) = x(0) - x(4) W_2^0$$

Let $G_{12}(k) = \text{DFT}\{g_{12}(n)\}$. The 2-point DFT of $g_{12}(n)$ is given by

$$G_{12}(k) = \sum_{n=0}^{1} g_{12}(n) W_{N/4}^{nk}; \quad \text{for } k = 0, 1$$

$$\therefore \quad G_{12}(0) = g_{12}(0) W_2^0 + g_{12}(1) W_2^0 = g_{12}(0) + g_{12}(1) = x(2) + x(6) = x(2) + x(6) W_2^0$$

$$G_{12}(1) = g_{12}(0) W_2^0 + g_{12}(1) W_2^1 = g_{12}(0) - g_{12}(1) = x(2) - x(6) = x(2) - x(6) W_2^0$$

Let $G_{21}(k) = \text{DFT}\{g_{21}(n)\}$. The 2-point DFT of $g_{21}(n)$ is given by

$$G_{21}(k) = \sum_{n=0}^{1} g_{21}(n) W_{N/4}^{nk}; \quad \text{for } k = 0, 1$$

$$\therefore \quad G_{21}(0) = g_{21}(0)W_2^0 + g_{21}(1)W_2^0 = g_{21}(0) + g_{21}(1) = x(1) + x(5) = x(1) + x(5)W_2^0$$

$$G_{21}(1) = g_{21}(0)W_2^0 + g_{21}(1)W_2^1 = g_{21}(0) - g_{21}(1) = x(1) - x(5) = x(1) - x(5)W_2^0$$

Let $G_{22}(k) = \text{DFT}\{g_{22}(n)\}$. The 2-point DFT of $g_{22}(n)$ is given by

$$G_{22}(k) = \sum_{n=0}^{1} g_{22}(n) W_{N/4}^{nk}; \quad \text{for } k = 0, 1$$

$$\therefore \quad G_{22}(0) = g_{22}(0)W_2^0 + g_{22}(1)W_2^0 = g_{22}(0) + g_{22}(1) = x(3) + x(7) = x(3) + x(7)W_2^0$$

$$G_{22}(1) = g_{22}(0)W_2^0 + g_{22}(1)W_2^1 = g_{22}(0) - g_{22}(1) = x(3) - x(7) = x(3) - x(7)W_2^0$$

The computation of first stage is shown in **Figure 5(a).**

The second stage of computation

In the second stage of computations, 4-point DFTs are computed using the 2-point DFTs obtained in stage one as inputs.

Let $F_1(k) = \text{DFT}\{f_1(n)\}$. The 4-point DFT of $f_1(n)$ can be computed using the equation

$$F_1(k) = G_{11}(k) + W_{N/2}^k G_{12}(k); \quad \text{for } k = 0, 1, 2, 3$$

$$\therefore \quad F_1(0) = G_{11}(0) + W_4^0 G_{12}(0) = G_{11}(0) + G_{12}(0)W_4^0$$

$$F_1(1) = G_{11}(1) + W_4^1 G_{12}(1) = G_{11}(1) + G_{12}(1)W_4^1$$

$$F_1(2) = G_{11}(2) + W_4^2 G_{12}(2) = G_{11}(0) - G_{12}(0)W_4^0$$

$$F_1(3) = G_{11}(3) + W_4^3 G_{12}(3) = G_{11}(1) - G_{12}(1)W_4^1$$

Let $F_2(k) = \text{DFT}\{f_2(n)\}$. The 4-point DFT of $f_2(n)$ can be computed using the equation

$$F_2(k) = G_{21}(k) + W_4^k G_{22}(k); \quad \text{for } k = 0, 1, 2, 3$$

$$\therefore \quad F_2(0) = G_{21}(0) + W_4^0 G_{22}(0) = G_{21}(0) + G_{22}(0)W_4^0$$

$$F_2(1) = G_{21}(1) + W_4^1 G_{22}(1) = G_{21}(1) + G_{22}(1)W_4^1$$

$$F_2(2) = G_{21}(2) + W_4^2 G_{22}(2) = G_{21}(0) - G_{22}(0)W_4^0$$

$$F_2(3) = G_{21}(3) + W_4^3 G_{22}(3) = G_{21}(1) - G_{22}(1)W_4^1$$

Here $G_{11}(k)$ and $G_{12}(k)$ are periodic with periodicity of 2.

i.e. $\quad G_{11}(k+2) = G_{11}(k) \quad \text{and} \quad G_{12}(k+2) = G_{12}(k)$

Here $G_{21}(k)$ and $G_{22}(k)$ are periodic with periodicity of 2.

i.e. $G_{21}(k+2) = G_{21}(k)$ and $G_{22}(k+2) = G_{22}(k)$

The computation of second stage is shown in **Figure 5(b).**

The third stage of computation

In the third stage of computations, the 8-point DFT is computed using the 4-point DFTs obtained in second stage as inputs.

Let $X(k) = \text{DFT}\{x(n)\}$. The 8-point DFT of $x(n)$ can be computed using the equation.

$$X(k) = F_1(k) + W_8^k F_2(k); \quad \text{for } k = 0, 1, 2, 3, 4, 5, 6, 7$$

$\therefore$

$$X(0) = F_1(0) + W_8^0 F_2(0)$$

$$X(1) = F_1(1) + W_8^1 F_2(1)$$

$$X(2) = F_1(2) + W_8^2 F_2(2)$$

$$X(3) = F_1(3) + W_8^3 F_2(3)$$

$$X(4) = F_1(4) + W_8^4 F_2(4) = F_1(0) - F_2(0)W_8^0$$

$$X(5) = F_1(5) + W_8^5 F_2(5) = F_1(1) - F_2(1)W_8^1$$

$$X(6) = F_1(6) + W_8^6 F_2(6) = F_1(2) - F_2(2)W_8^2$$

$$X(7) = F_1(7) + W_8^7 F_2(7) = F_1(3) - F_2(3)W_8^3$$

Here $F_1(k)$ and $F_2(k)$ are periodic with periodicity of 4.

i.e. $F_1(k+4) = F_1(k)$ and $F_2(k+4) = F_2(k)$

The computation of third stage is shown in Figure 7.6(c).

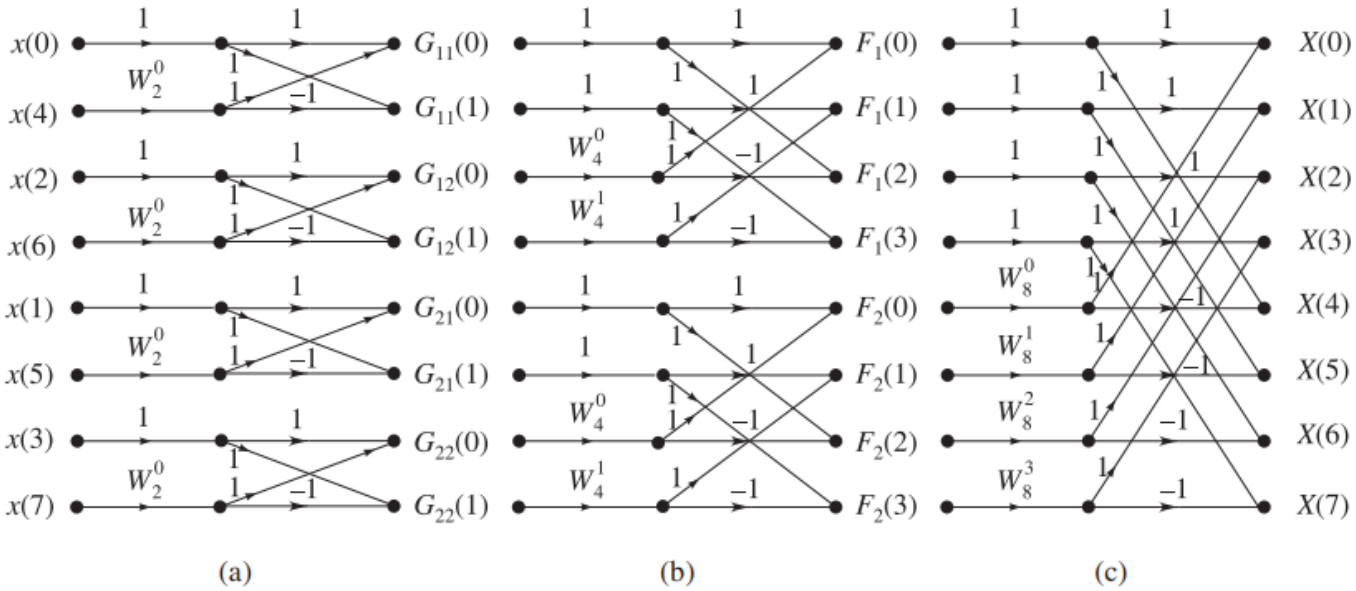

Figure 5: (a)–(c) Flow graphs for implementation of 1st, 2nd and 3rd stages of computation.

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 **Figure 6.** For 8-point DIT FFT, the input sequence $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$, must be fed in bit reversed order, 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 using radix-2 FFT involves 3 stages of computation, each stage involving 4 butterflies. The output $X(k)$ will be in normal order. 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. The 8-point FFT calculation requires $8 \log_2 8 = 24$ complex additions and $(8/2) \log_2 8 = 12$ complex multiplications.

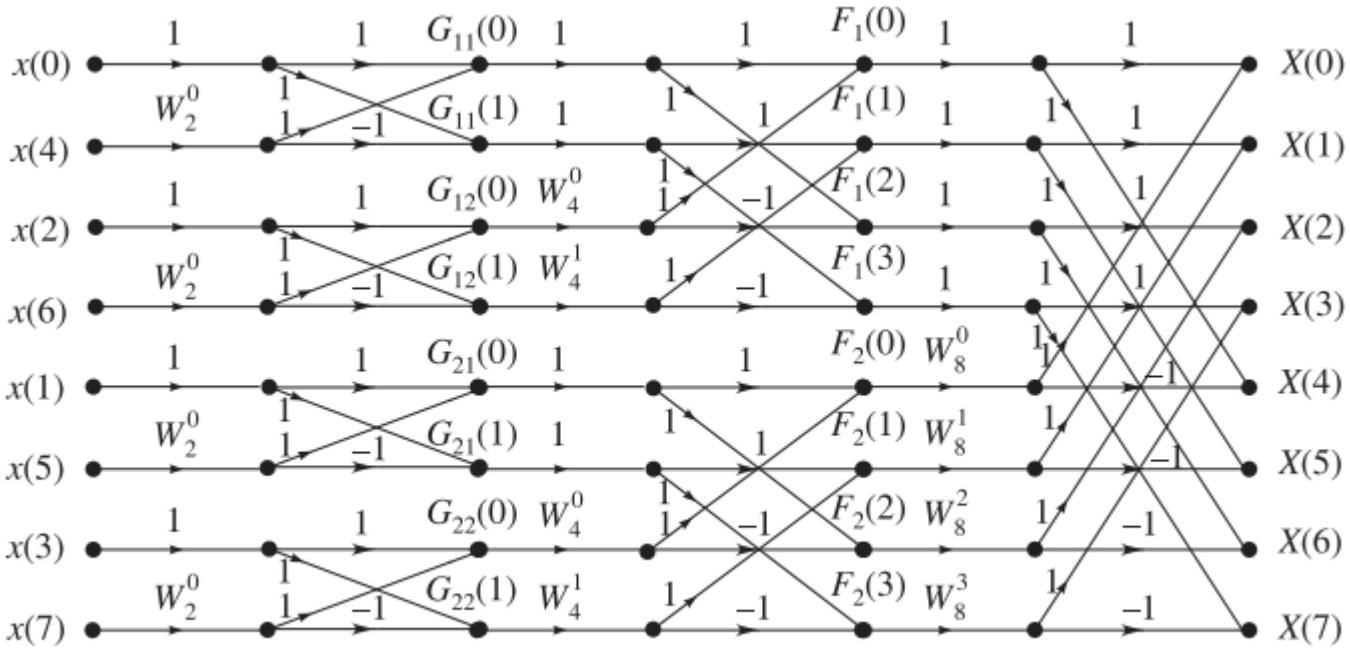

Figure 6: Butterfly line diagram for 8-point DIT FFT algorithm for $N = 8$.

Example 2 (a) Implement the decimation-in-time FFT algorithm for $N = 16$, (b) In the above question how many non-trivial multiplications are required?

Solution: (a) The butterfly line diagram showing the implementation of 16-point radix-2 DIT FFT algorithm for $N = 16$ is shown in **Figure 7.** For DIT FFT, the given 16-point sequence $x(n)$ is to be fed in bit reversed order as

$$x_r(n) = \{x(0), x(8), x(4), x(12), x(2), x(10), x(6), x(14), x(1), x(9), x(5), x(13), x(3), x(11), x(7), x(15)\}$$

Since $N = 16 = 2^4$, the radix-2 DIT FFT computation involves 4 stages. The output is in normal order.

(b) The number of non-trivial multiplications required in the implementation of DIT FFT for $N = 16 = 2^4$ is $N/2 \log_2 N = 16/2 \log_2 16 = 8 \log_2 2^4 = 8 \times 4 = 32$. The number of complex additions are $N \log_2 N = 16 \log_2 16 = 16 \log_2 2^4 = 16 \times 4 = 64$.

Number of multiplications required by direct computation $= N^2 = 16^2 = 256$

Number of complex additions required by direct computation $= N(N-1) = 16 \times 15 = 240$.

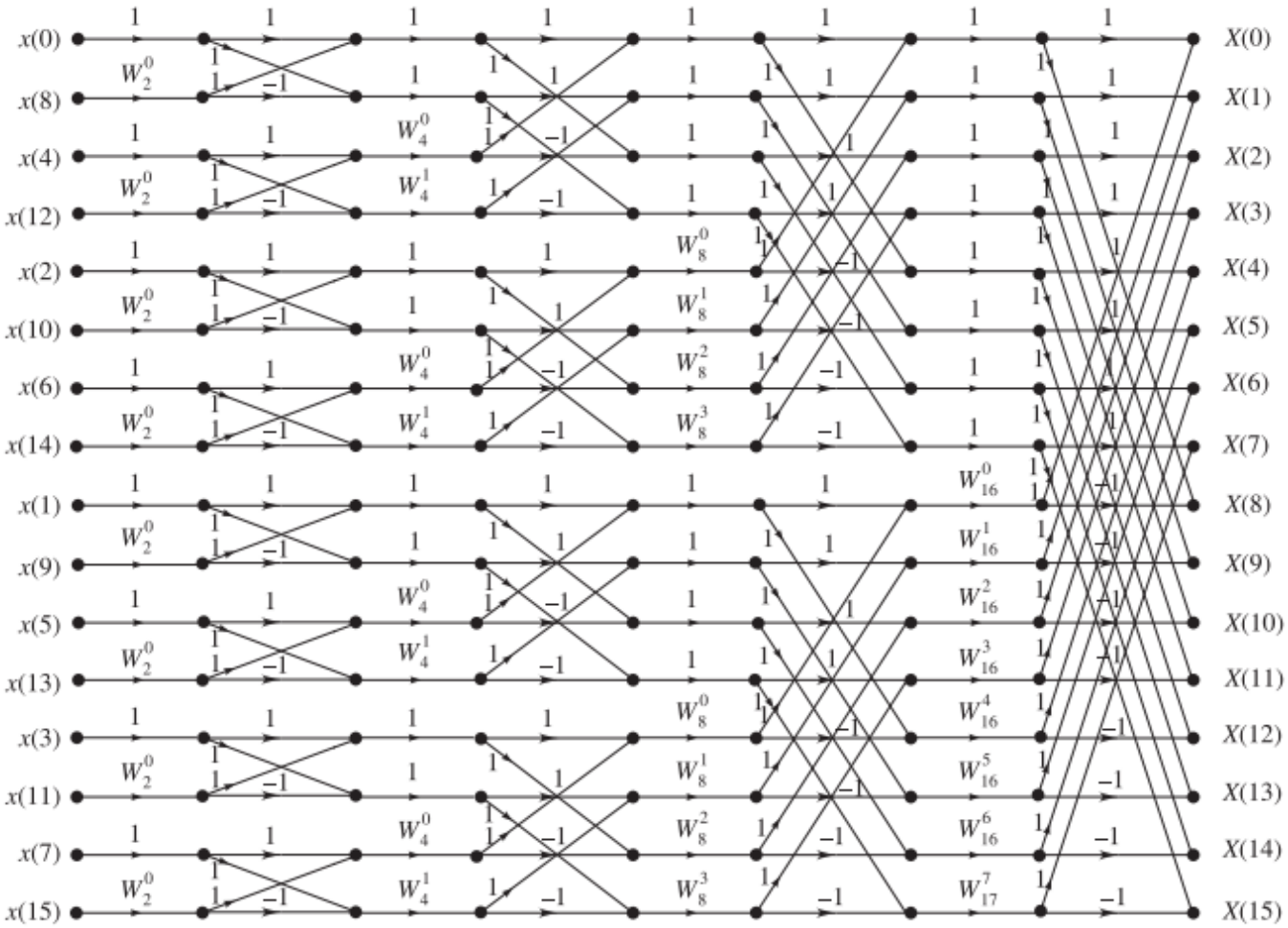

Figure 7: Butterfly line diagram for DIT FFT algorithm for $N = 16$.

Example 3 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 $= N\log_2 N = 32\log_2 32 = 32\log_2 2^5 = 160$

Computation of IDFT through FFT

The IDFT of an N-point sequence $\{X(k)\}$; $k = 0, 1, ..., N-1$ is defined as

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

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

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

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

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

The term inside the square brackets in the above equation for $x(n)$ is same as the DFT computation of a sequence $X^*(k)$ and may be computed using any FFT algorithm. So we can say that the IDFT of $X(k)$ can be obtained by finding the DFT of $X^*(k)$, taking the conjugate of that DFT and dividing by N. Hence, to compute the IDFT of $X(k)$ the following procedure can be followed

1. Take conjugate of $X(k)$, i.e. determine $X^*(k)$.
2. Compute the N-point DFT of $X^*(k)$ using radix-2 FFT.
3. Take conjugate of the output sequence of FFT.
4. Divide the sequence obtained in step-3 by N.

The resultant sequence is $x(n)$.

Thus, a single FFT algorithm serves the evaluation of both direct and inverse DFTs.

Example 4 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 $\{X(k), k = 0, 1, 2, ..., 7\}$ is defined as

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

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

The flow graph for computation of N = 8-point IDFT using DIT FFT algorithm is shown in **Figure 8.**

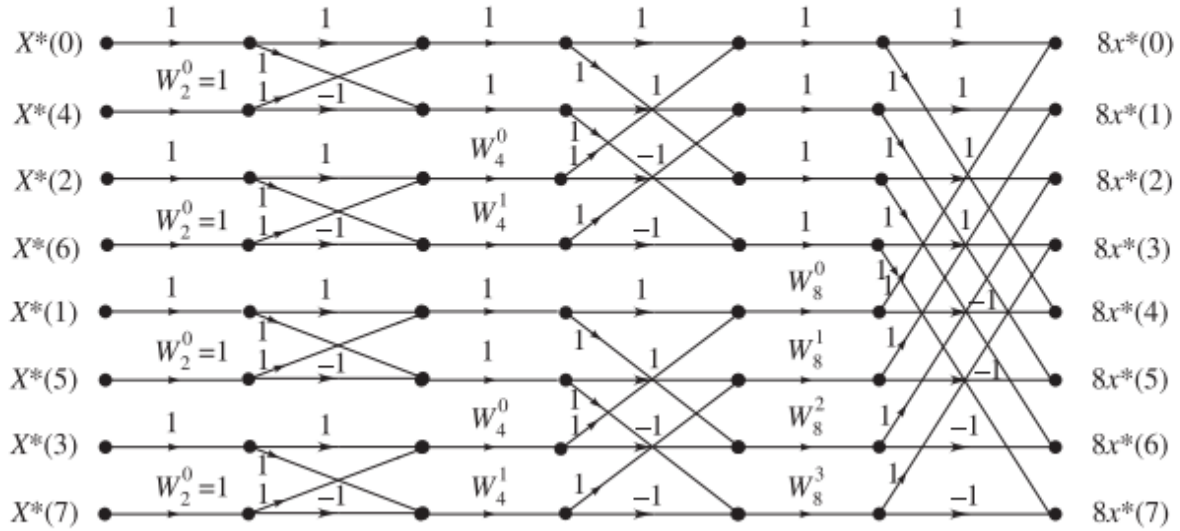

Figure 8: Computation of 8-point DFT of $X^*(k)$ by radix-2, DIT FFT.

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

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

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

Example 5 Compute the DFT of the square wave sequence

$$x(n) = \begin{cases} 1, & 0 \le n \le \left(\frac{N}{2} - 1\right) \\ -1, & \frac{N}{2} \le n \le N \end{cases}$$

where N is even.

Solution: It is given that N is even, but the value of N is not given. Let us take $N = 4$.

$$\therefore \qquad x(n) = \{1, 1, -1, -1\}$$

Let us calculate $X(k)$ using 4-point radix-2 DIT FFT algorithm as shown in **Figure 9.** For DIT FFT, the input is in bit reversed order and output is in normal order. $x(n)$ in bit reversed order is $x_r(n) = \{1, -1, 1, -1\}$.

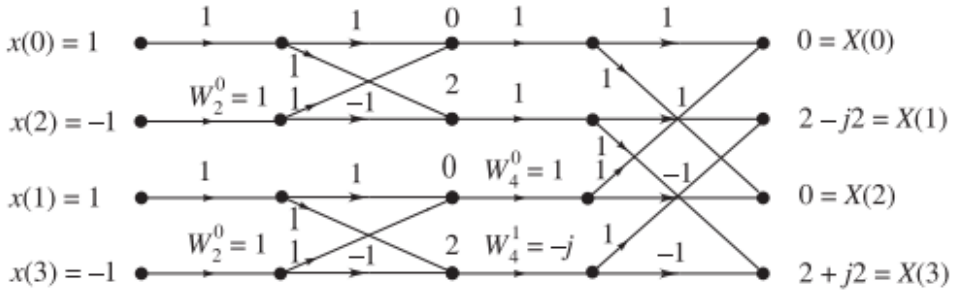

Figure 9: 4-point DFT of x(*n*) by radix-2 DIT FFT.

From **Figure 9** , we have $X(k) = \{0, 2 - j2, 0, 2 + j2\}$.

Example 6 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 10.**

From **Figure 10** , we get the 8-point DFT of $x(n)$ as

$$X(k) = \{12, 0, 0, 0, 4, 0, 0, 0\}$$

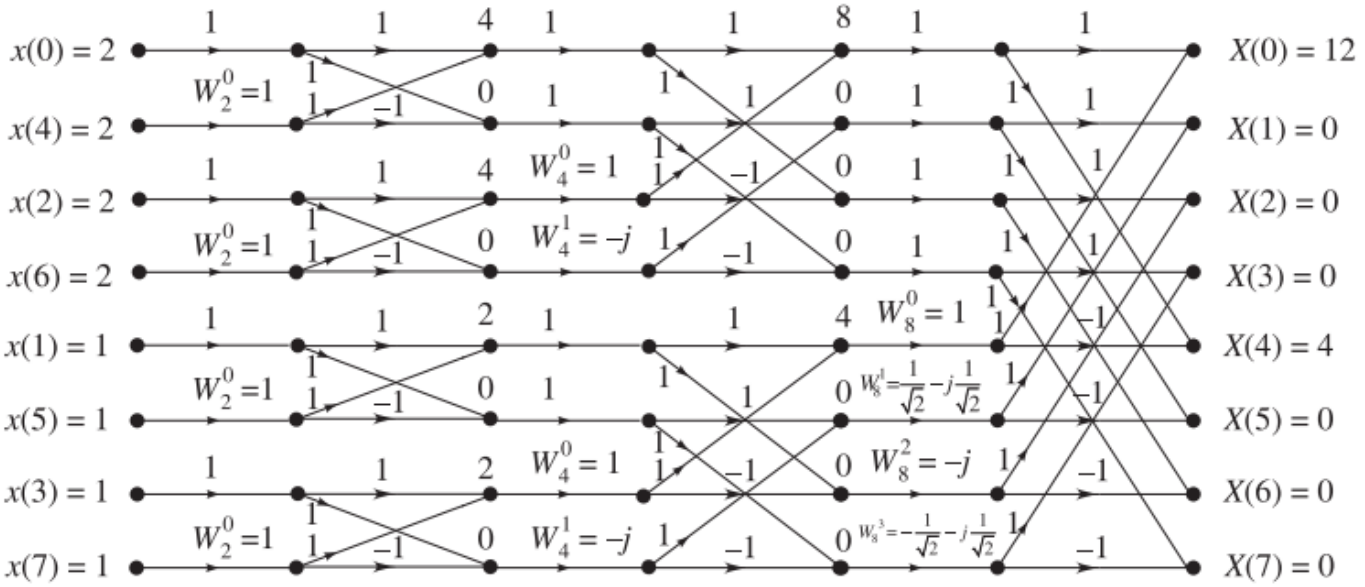

Figure 10: Computation of 8-point DFT of x(*n*) by radix-2, DIT FFT.

Example 7 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 11** . For DIT FFT, the input is in bit reversed order and output is in normal order. The given sequence in bit reverse order is

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

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

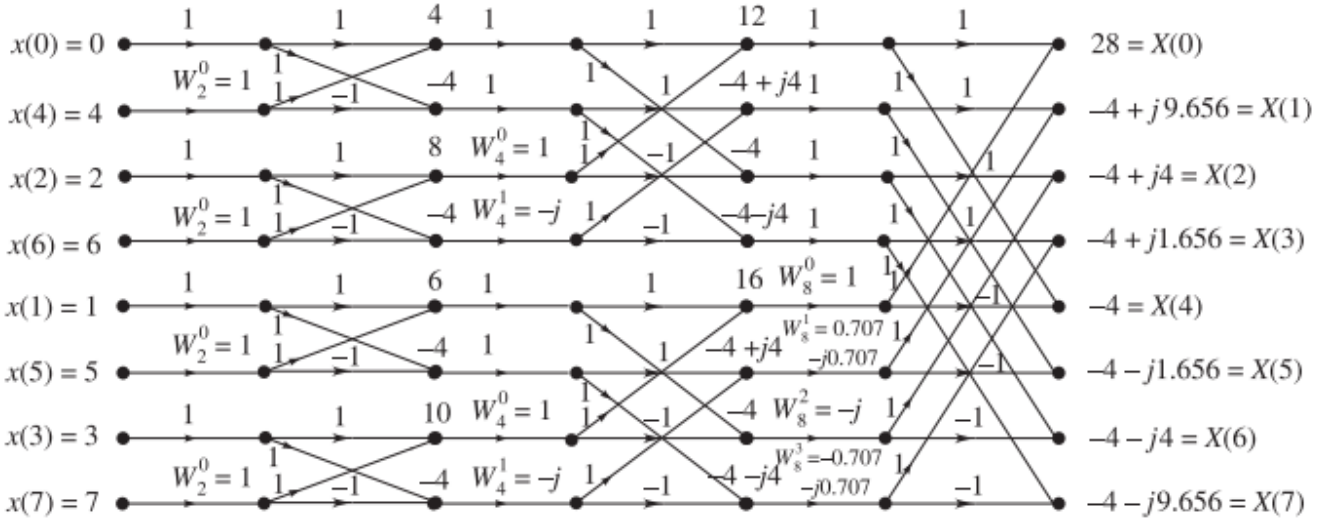

Figure 11: Computation of 8-point DFT of x(*n*) by radix-2, DIT FFT.

From Figure 11 , we get the 8-point DFT of $x(n)$ as

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

Example 8 Compute the IDFT of the sequence

$$X(k) = \{7,\ -0.707 - j0.707, -j, 0.707 - j0.707, 1, 0.707 + j0.707, j,\ -0.707 + j0.707\}$$

using DIT algorithm.

Solution: The IDFT $x(n)$ of the given sequence $X(k)$ can be obtained by finding $X^*(k)$, the conjugate of $X(k)$, finding the 8-point DFT of $X^*(k)$ using radix-2 DIT FFT algorithm to get $8x^*(n)$, taking the conjugate of that to get $8x(n)$ and then dividing by 8 to get $x(n)$. For DIT FFT, the input $X^*(k)$ must be in bit reverse order. The output $8x^*(n)$ will be in normal order. For the given $X(k)$.

$$X^*(k) = \{7, -0.707 + j0.707, j, 0.707 + j0.707, 1, 0.707 - j0.707,\ -j, -0.707 - j0.707\}$$

$X^*(k)$ in bit reverse order is

$$X_r^*(k) = \{7, 1, j, -j, -0.707 + j0.707, 0.707\ - j0.707, 0.707 + j0.707,\ -0.707 - j0.707\}$$

The 8-point DFT of $X^*(k)$ using radix-2, DIT FFT algorithm is computed as shown in **Figure 12.**

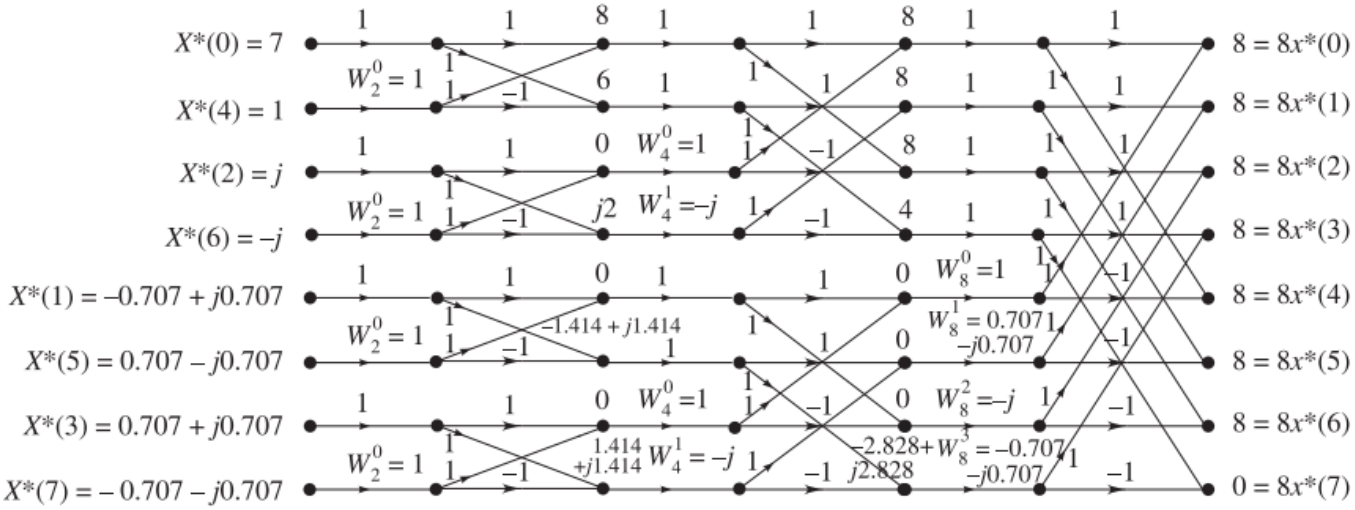

Figure 12: Computation of 8-point DFT of $X^*(k)$ by radix-2, DIT FFT.

From the DIT FFT algorithm of **Figure 12** , we have

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

$$\therefore \quad 8x(n) = \{8, 8, 8, 8, 8, 8, 8, 0\}$$

$$\therefore \quad x(n) = \{1, 1, 1, 1, 1, 1, 1, 0\}$$