FFT(Fast Fourier Transform)

FFT(Fast Fourier Transform)

5.1 INTRODUCTION

As discussed in last chapter the Discrete Fourier Transform (DFT) plays a vital role in several applications of digital signal processing (DSP). These applications may include linear filtering, correlation analysis, and spectrum analysis. A major reason for its importance is the existence of efficient algorithms for computing the discrete Fourier Transform (DFT).

In previous chapter, we have studied how to obtain DFT of a sequence by using direct computation. Basically, the direct computation of DFT requires large number of computations. Hence, more processing time is required.

FFT(Fast Fourier Transform)

For the computation of N-point DFT, N^2 complex multiplications and $N^2 - N$ complex additions are required. If the value of N is large then the number of computations will go into lakhs. This proves inefficiency of direct DFT computation.

FFT(Fast Fourier Transform)

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \le k \le N-1$$

For 1 value of k, multiplication should be done for all values for N(0 to N-1).

Range of K=0 to N-1. So, total **complex multiplication =N*N**

FFT(Fast Fourier Transform)

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \le k \le N-1$$

Complex Addition

N=2, N=3, N=4

FFT(Fast Fourier Transform)

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad 0 \le k \le N-1$$

Complex Addition

FFT(Fast Fourier Transform)

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

Complex Addition:

Similarly for N we need N-1 complex addition and since k has range from 0 to N-1.

Total **Complex Addition=N*(N-1)**

FFT(Fast Fourier Transform)

...by a direct DFT computation.

The primary objective of the present chapter is the description of computationally efficient algorithms for evaluating the DFT. We shall be discussing two different approaches. One is a divide-and-conquer approach in which a DFT of size N, where N is a composite number, is reduced to the computation of smaller DFTs from which the larger DFT is computed. In particular, we present important computational algorithms, called **Fast Fourier Transform (FFT)** algorithms, for computing the DFT when the size N ... of 4. The FFT ...

FFT(Fast Fourier Transform)

DFT and IDFT

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn} \quad 0 \le k \le N-1 \qquad \ldots (5.1)$$

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

Usually, the data sequence $x(n)$ is also assumed to be complex valued. ... (5.2)

Similarly, the IDFT becomes

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W_N^{-nk} \quad 0 \le n \le N-1 \qquad \ldots (5.3)$$

FFT(Fast Fourier Transform)

5.4 DIVIDE-AND-CONQUER APPROACH TO COMPUTATION OF THE DFT

The development of computationally efficient algorithms for the DFT is made possible if we adopt 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 firmly of computationally efficient algorithms known collectively as **FFT algorithms**.

...the computation of an N-point DFT, where N

FFT(Fast Fourier Transform)

5.5 RADIX-2 FFT ALGORITHMS

1. Background Concept

(Sem. Exam, WBTU, Kolkata, 2005–06)

In previous article, we discussed two algorithms for efficient computation of the DFT based on the divide-and-conquer approach. However, such an approach is applicable when the number N of data points is not a prime. In particular, the approach is very efficient when N is highly composite, i.e., when N can be factored as $N = r_1 r_2 r_3 \ldots r_v$ where the $[r_j]$ are prime.

There is an important case in which $r_1 = r_2 = \ldots = r_v \equiv r$, so that $N = r^v$. In such a case, the DFTs are of size r, so that the computation of the N-point DFT has a regular pattern. **The number r is called the radix of the FFT algorithm.**

2. Basic Definition

FFT(Fast Fourier Transform)

2. Basic Definition

Now, let us discuss radix-2 algorithms. These algorithms are the most widely used FFT algorithms.

While calculating DFT, we have discussed that we always calculate N-point DFT. The number N can be factored as,

$$N = r_1, r_2, r_3 \ldots\ldots r_v \qquad \ldots(5.23)$$

Here, every r is a prime,

Now if $r_1 = r_2 = r_3 = \ldots\ldots = r_v = r$

then, we can write,

$$N = r^v \qquad \ldots(5.24)$$

Here, r is called as radix (base) of FFT algorithm and v indicates number of stages in FFT algorithm.

Now, radix means base and if its value is '2' then it is called as **radix-2 FFT algorithm.** Thus, when $r = 2$, equation (5.24) becomes,

$$N = 2^v \qquad \ldots(5.25)$$

Thus, if we are computing 8 point DFT then $N = 8$

so $8 = 2^v$

or $v = 3 \qquad \ldots(5.26)$

Therefore, for 8 point DFT, there are three stages of FFT algorithm

FFT(Fast Fourier Transform)

3. Types

While computing FFT, divide number of input samples by 2, till we reach minimum two samples. Based on this division, there are two algorithms as under :

(*i*) Radix-2 Decimation in time (DIT) algorithm,

(*ii*) Radix-2 Decimation in Frequency (DIF) algorithm.

4. Few Important Properties of Twiddle Factor

Before studying these algorithms, let us derive some important properties of twiddle factor W_N.

We know that the twiddle factor W_N is given by,

$$W_N = e^{\frac{-j2\pi}{N}} \qquad \ldots(5.27)$$

1. $W_N^k = W_N^{k+N}$

Using equation (5.27), we have,

$$W_N^{k+N} = \left[e^{\frac{-j2\pi}{N}} \right]^{k+N} = e^{\frac{-j\,2\pi k}{N}} \cdot e^{-j2\pi}$$

FFT(Fast Fourier Transform)

But, we know that $e^{-j2\pi} = \cos 2\pi - j \sin 2\pi = 1 - j0 = 1$

Therefore, $$W_N^{k+N} = e^{-\frac{j\,2\pi}{N}k}$$

or $$W_N^{k+N} = \left(e^{-j\frac{2\pi}{N}}\right)^k \quad \ldots(5.28)$$

In equation (5.28), the bracket term is W_N

Hence, $$W_N^{k+N} = W_N^k \quad \ldots(5.29)$$

Equation (5.29) indicates that twiddle factor is periodic.

FFT(Fast Fourier Transform)

... indicates that twiddle factor is periodic.

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

Using equation (5.28), we can write,

$$W_N^{k+\frac{N}{2}} = \left[e^{\frac{-j2\pi}{N}}\right]^{\left(k+\frac{N}{2}\right)} = e^{\frac{-j2\pi k}{N}} \cdot e^{\frac{-j2\pi}{N}\cdot\frac{N}{2}}$$

or

$$W_N^{k+\frac{N}{2}} = e^{\frac{-j2\pi k}{N}} \cdot e^{-j\pi} \qquad \ldots(5.30)$$

But $e^{-j\pi} = \cos\pi - j\sin\pi = -1 - 0 = -1$

Therefore, we have

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

or

$$W_N^{k+\frac{N}{2}} = -\left(e^{-j\frac{2\pi}{N}}\right)^k \qquad \ldots(5.31)$$

FFT(Fast Fourier Transform)

In equation (5.31), the bracket term is, W_N.

Thus, we write $$W_N^{k+\frac{N}{2}} = -W_N^k \qquad \ldots(5.32)$$

Equation (5.32) indicates that twiddle factor is symmetric

FFT(Fast Fourier Transform)

3. $W_N^2 = W_{N/2}$

From equation (5.27), we have,

$$W_{N/2} = e^{-\frac{j\,2\pi}{N/2}} = e^{-\frac{j\,2\pi}{N}\cdot 2} = \left[e^{-\frac{j\,2\pi}{N}}\right]^2$$

Therefore, we have $W_{N/2} = W_N^2$

FFT(Fast Fourier Transform)

5.6 RADIX-2 DECIMATION IN TIME (DIT) ALGORITHM (DIT FFT)

(Sem. Exam. JNTU, Hyderabad, 2005-06)

Here, the word 'decimate' means to break into parts. Therefore, DIT indicates dividing (splitting) the sequence in time domain. The different stages of decimation are as under:

FFT(Fast Fourier Transform)

(i) First stage of decimation

Let $x(n)$ be the given input sequence containing N samples. Now, for decimation in time, we shall divide $x(n)$ into even and odd sequences *i.e.*,

$$x(n) = f_1(m) + f_2(m)$$

Here, $f_1(m)$ is even sequence and $f_2(m)$ is odd sequence ...(5.33)

Also, $$f_1(m) = x(2m), \quad m = 0, 1, \ldots \frac{N}{2} - 1$$ ...(5.34)

and $$f_2(m) = x(2m+1), \quad m = 0, 1, \ldots, \frac{N}{2} - 1$$ ...(5.35)

Input sequence $x(n)$ has N samples. Therefore, after decimation, $f_1(m)$ and $f_2(m)$ will consist of $\frac{N}{2}$ samples.

Now, according to the definition of DFT, we have

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \quad k = 0, 1, 2, \ldots\ldots N-1$$ ...(5.36)

FFT(Fast Fourier Transform)

Since, we have divided $x(n)$ into two parts, therefore, we can write separate summation for even and odd sequences as under :

$$X(k) = \sum_{n \text{ even}} x(n)\ W_N^{kn} + \sum_{n \text{ odd}} x(n)\ W_N^{kn} \qquad ...(5.37)$$

In the above expression given by equation 5.37 the first summation represents even sequence. So, we shall substitute $n = 2m$ in first summation. The second summation represents odd sequence, so, we shall substitute $n = (2m + 1)$ in second summation. Since, even and odd sequences contain $\frac{N}{2}$ samples each, the limits of summation will be from $m = 0$ to $\frac{N}{2} - 1$.

Therefore, we have

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x(2m)\ W_N^{2km} + \sum_{m=0}^{\frac{N}{2}-1} x(2m+1)\ W_N^{k(2m+1)} \qquad ...(5.38)$$

But, $x(2m)$ is an even sequence, so it is $f_1(m)$ and $x(2m + 1)$ is an odd sequence, so it is $f_2(m)$.

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

or
$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} f_1(m)\ \left(W_N^2\right)^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(m) \left(W_N^2\right)^{km} \qquad ...(5.39)$$

FFT(Fast Fourier Transform)

Now, we have $W_N^2 = W_{N/2}$

Therefore,
$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} f_1(m)\, W_{N/2}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(m)\, W_{N/2}^{km} \qquad \ldots(5.40)$$

Comparing each summation with the definition of DFT, we have

$$X(k) = F_1(k) + W_N^k\, F_2(k), \quad k = 0, 1, \ldots, N-1 \qquad \ldots(5.4)$$

FFT(Fast Fourier Transform)

periodic with period $\frac{N}{2}$.

Using periodicity property of DFT, we have.

$$F_1\left(k+\frac{N}{2}\right) = F_1(k) \qquad \ldots(5.42)$$

and

$$F_2\left(k+\frac{N}{2}\right) = F_2(k) \qquad \ldots(5.43)$$

Replacing k by $k+\frac{N}{2}$ in equation (5.41), we have

$$X\left(k+\frac{N}{2}\right) = F_1\left(k+\frac{N}{2}\right) + W_N^{k+\frac{N}{2}} F_2\left(k+\frac{N}{2}\right) \qquad \ldots.(5.44)$$

Now, we have,

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

FFT(Fast Fourier Transform)

Therefore, $$X\left(k+\frac{N}{2}\right) = F_1\left(k+\frac{N}{2}\right) - W_N^k\, F_2\left(k+\frac{N}{2}\right) \quad \ldots(5.45)$$

Using equations (5.42) and (5.43), we obtain,

$$X\left(k+\frac{N}{2}\right) = F_1(k) - W_N^k\, F_2(k) \quad \ldots(5.46)$$

FFT(Fast Fourier Transform)

(*iv*) Computation of 2-point DFT

According to the basic definition of DFT, we have

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

$$k = 0, 1, \ldots N-1 \qquad \ldots(5.63)$$

Let us use equation (5.63) to compute 2-point DFT. From figure 5.8, let us consider the first block of 2-point DFT. It has been separately drawn as shown in figure 5.8.

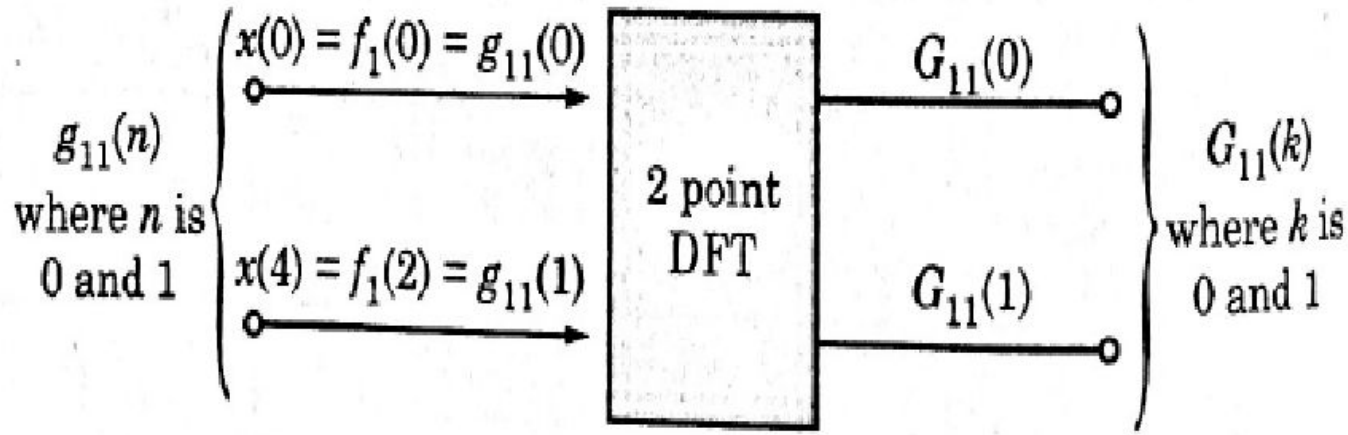


FIGURE 5.8 *Block of 2-point DFT.*

FFT(Fast Fourier Transform)

Here, input sequences are $g_{11}(0)$ and $g_{11}(1)$. We denote it by $g_{11}(n)$; where n varies from 0 to 1.
Now, the output sequences are $G_{11}(0)$ and $G_{11}(1)$. We can denote it by $G_{11}(k)$, where k varies from 0 to 1. Here, $G_{11}(k)$ is DFT of $g_{11}(n)$.

Thus, for $G_{11}(k)$, we can write equation (5.63) as under:

$$G_{11}(k) = \sum_{n=0}^{1} g_{11}(n)\ W_2^{kn}\ ,\ k = 0, 1 \qquad \ldots(5.64)$$

Note that this is 2 point DFT, so we have to substitute $N = 2$
Now, substituting values of k in equation (5.64), we obtain as under :
For $k = 0$, we have

$$G_{11}(0) = \sum_{n=0}^{1} g_{11}(n)\ W_2^0$$

But $$W_2^0 = 1$$

FFT(Fast Fourier Transform)

$$G_{11}(0) = \sum_{n=0}^{1} g_{11}(n)\ W_2^0$$

But $W_2^0 = 1$

Therefore, $G_{11}(0) = \sum_{n=0}^{1} g_{11}(n)$

Expanding the summation, we shall have

$$G_{11}(0) = g_{11}(0) + g_{11}(1)$$

For, $k = 1$, we have

...(5.65)

$$G_{11}(1) = \sum_{n=0}^{1} g_{11}(n)\ W_2^n$$

FFT(Fast Fourier Transform)

Expanding the summation, we get

$$G_{11}(1) = g_{11}(0)\ W_2^0 + g_{11}(1)\ W_2^1 \qquad \ldots(5.66)$$

We have $W_N = e^{-\frac{j2\pi}{N}}$

Therefore, $W_2^1 = \left(e^{-\frac{j2\pi}{2}}\right)^1 = e^{-j\pi} = \cos\pi - j\sin\pi = -1 - j0$

Also, $W_2^1 = -1$

and $W_2^0 = 1$

Substituting these values in equation (5.66), we shall have

$$G_{11}(1) = g_{11}(0) - g_{11}(1) \qquad \ldots(5.67)$$

Using equations (5.65) and (5.67), we can represent the computation of 2-point DFT as shown to figure 5.9. This structure looks like a butterfly. Hence, it is called as **FFT butterfly structure.**

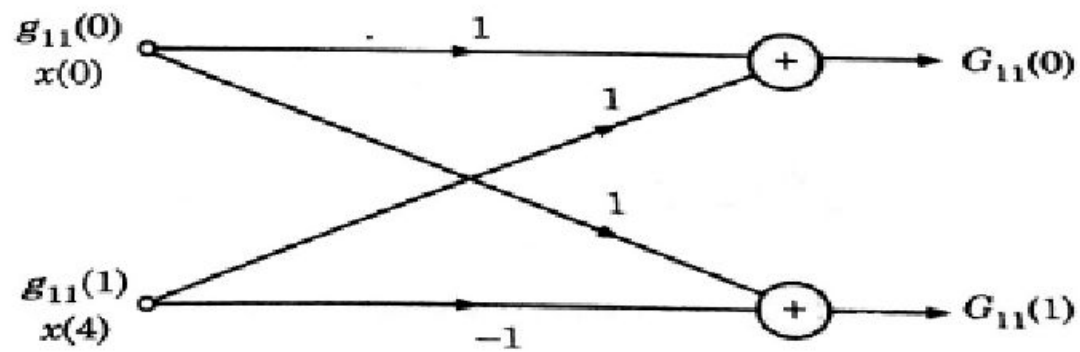


FIGURE 5.9 *FFT butterfly structure.*

FFT(Fast Fourier Transform)

Now, we know that $W_2^0 = 1$. Thus, we can modify equations (5.65) and (5.67) as under :

$$G_{11}(0) = g_{11}(0) + W_2^0\, g_{11}(1) \quad ...(5.68)$$

and $$G_{11}(1) = g_{11}(0) - W_2^0\, g_{11}(1) \quad ...(5.69)$$

The modified butterfly structure is shown in figure 5.10.

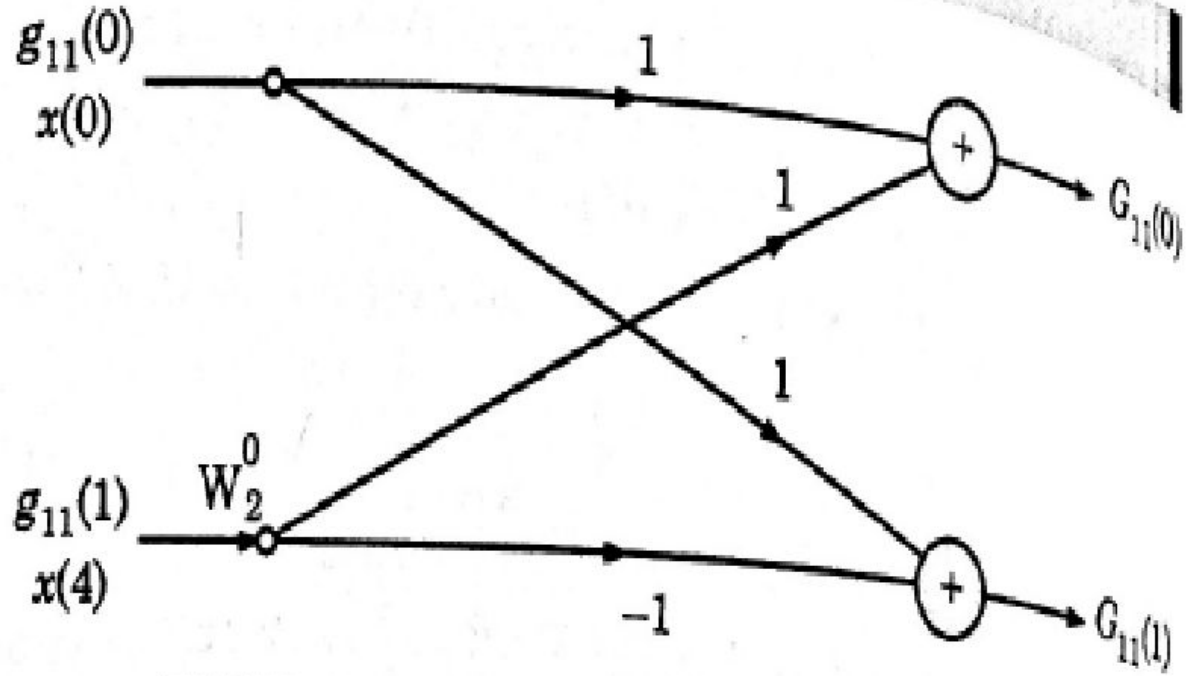


FIGURE 5.10 *Modified butterfly structure.*

Similarly, for other 2-point DFTs, we can draw the butterfly structure.

FFT(Fast Fourier Transform)

EXAMPLE 5.3 Derive DIT FFT flow graph for $N = 4$ and hence find DFT of $x(n) = \{1, 2, 3, 4\}$
(Sem, Exam, PTU, Jalander, 2006–07)

Solution: (i) First stage of decimation.

Let us write the following equations for first stages of decimation:

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

and $$X\left(k + \frac{N}{2}\right) = F_1(k) - W_N^k F_2(k), \quad k = 0, 1, \ldots \frac{N}{2} - 1 \qquad \ldots(ii)$$

Here, since, $N = 4$, therefore, we have

$$X(k) = F_1(k) + W_4^k F_2(k), \quad k = 0, 1 \qquad \ldots(iii)$$

and $$X(k + 2) = F_1(k) - W_4^k F_2(k), \quad k = 0, 1 \qquad \ldots(iv)$$

FFT(Fast Fourier Transform)

Substituting values of k in equation (iii), we obtain

$$X(0) = F_1(0) + W_4^0 F_2(0)$$

and

$$X(1) = F_1(1) + W_4^1 F_2(1)$$

...(v)

Similarly, substituting values of k in equation (iv), we obtain

$$X(2) = F_1(0) - W_4^0 F_2(0)$$

and

$$X(3) = F_1(1) - W_4^1 F_2(1)$$

...(vi)

FFT(Fast Fourier Transform)

This signal flow graph has been shown in figure 5.14 (*a*).

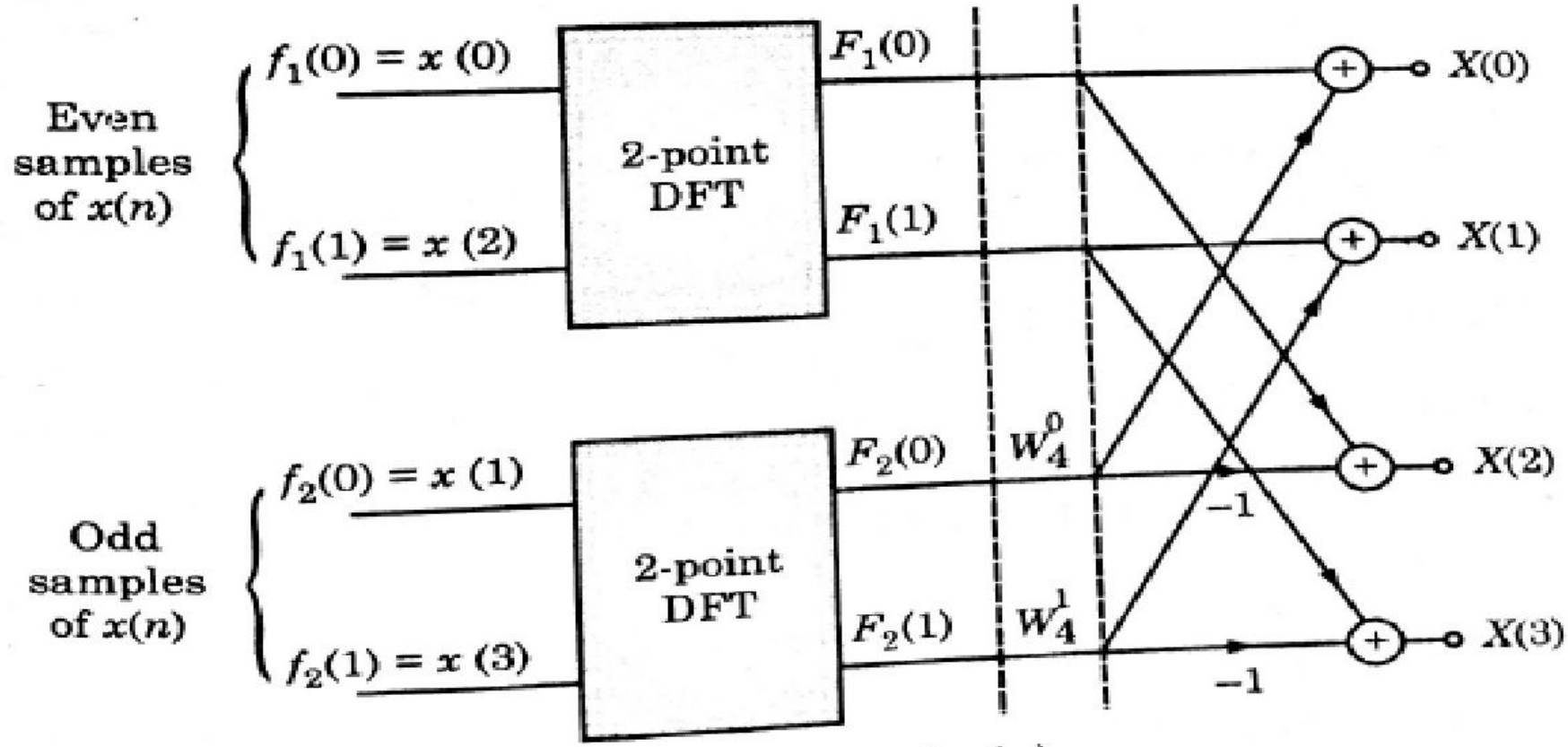
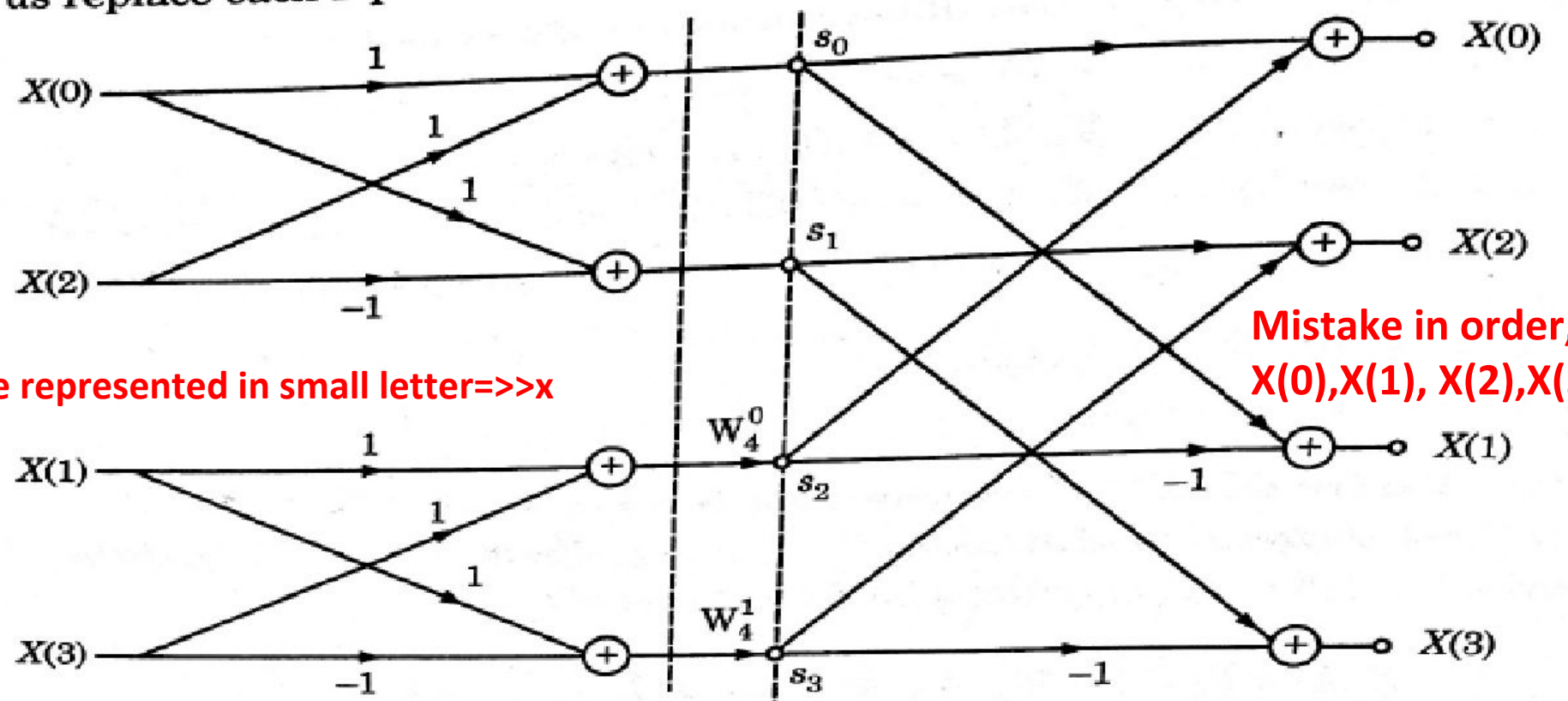


FIGURE 5.14 (*a*)

FFT(Fast Fourier Transform)

Now, let us replace each 2-point DFT by butterfly structure as shown in figure 5.14 (*b*).



Mistake in order, it should be X(0),X(1), X(2),X(3)

X Should be represented in small letter=>>x

FIGURE 5.14 (*b*)

FFT(Fast Fourier Transform)

FIGURE 5.14 *(b)*

The given sequence is

$$x(n) = \{1, 2, 3, 4\}$$

The different values of twiddle factor are as under:

$$W_4^0 = 1$$

$$W_4^1 = e^{-\frac{j2\pi}{4}.1} = e^{-\frac{j\pi}{2}}$$

$$= \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = -j.$$

The output $s(n)$ will be

$$s_0 = x(0) + x(2) = 1 + 3 = 4$$
$$s_1 = x(0) - x(2) = 1 - 3 = -2$$
$$s_2 = [x(1) + x(3)]\ W_4^0 = 2 + 4 = 6$$
$$s_3 = [x(1) - x(3)]\ W_4^1 = (2 - 4) \cdot (-j) = 2j$$

FFT(Fast Fourier Transform)

$s_3 = [x(2) - x(3)]\ W_4 = (2 - 4) \cdot (-j) = 2j$

The final output will be given by

$$X(0) = s_0 + s_2 = 4 + 6 = 10$$
$$X(1) = s_1 + s_3 = -2 + j2$$
$$X(2) = s_0 - s_2 = 4 - 6 = -2$$
$$X(3) = s_1 - s_3 = -2 - j2$$

Thus, we have

$$X(k) = \{X(0), X(1), X(2), X(3)\}$$

Therefore, $X(k) = \{10, -2 + j2, -2, -2 - j2\}$ **Ans.**

FFT(Fast Fourier Transform)

EXAMPLE 5.1 **Determine the 8-point DFT of the following sequence.**

$$x(n) = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0\right\}.$$

Use in-place radix-2 deci-mation in time FFT algorithm.

FFT(Fast Fourier Transform)

Solution : This flow graph has been shown in figure 5.12.

Here, $s_1(n)$ represents output of stage 1 and $s_2(n)$ represents output of stage 2. The different values of twiddle factor are

$$W_8^0 = e^0 = 1$$

$$W_8^1 = e^{-j\frac{\pi}{4}} = 0.707 - j\,0.707$$

$$W_8^2 = e^{-j\frac{\pi}{2}} = -j$$

$$W_8^3 = -0.707 - j\,0.707$$

FFT(Fast Fourier Transform)

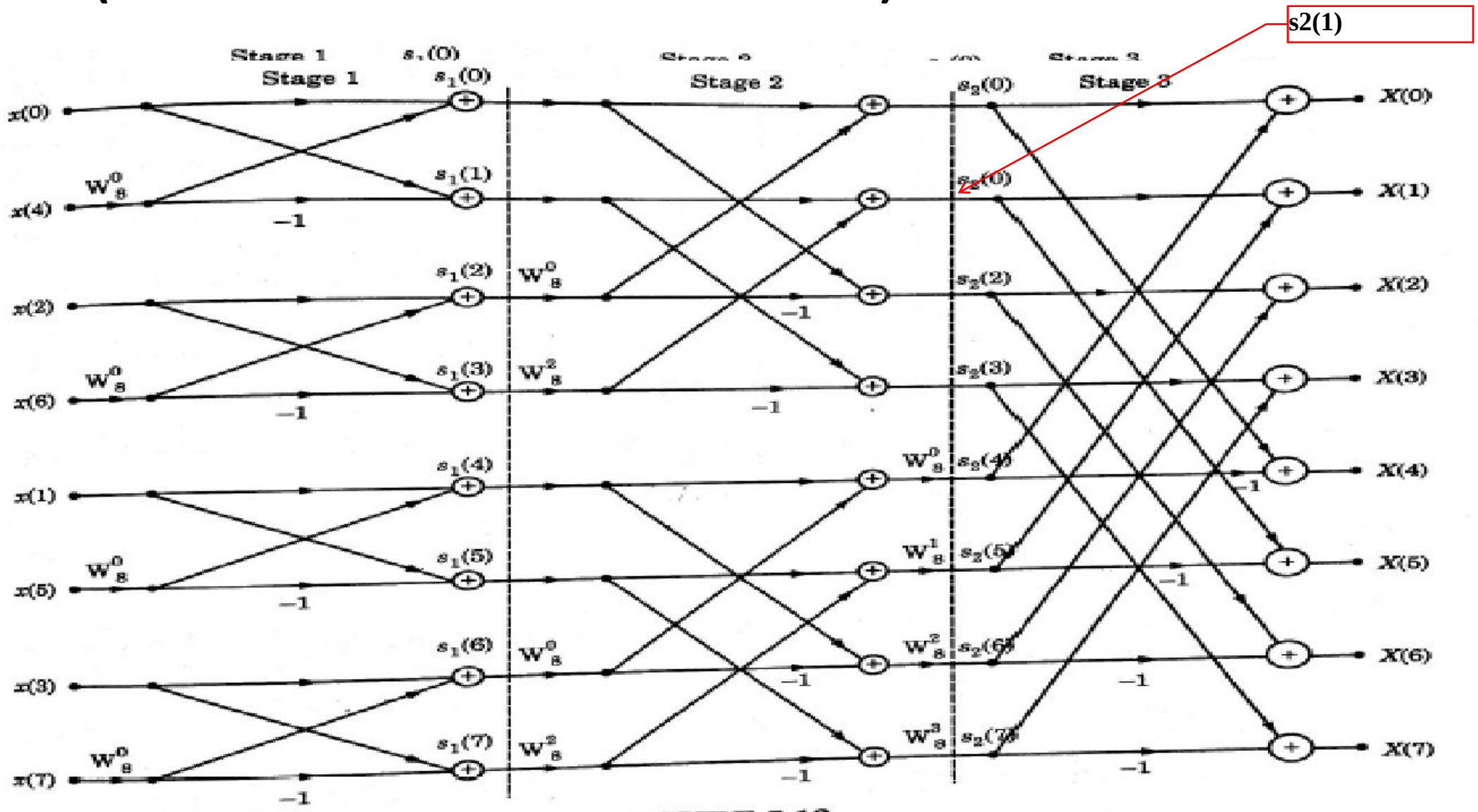
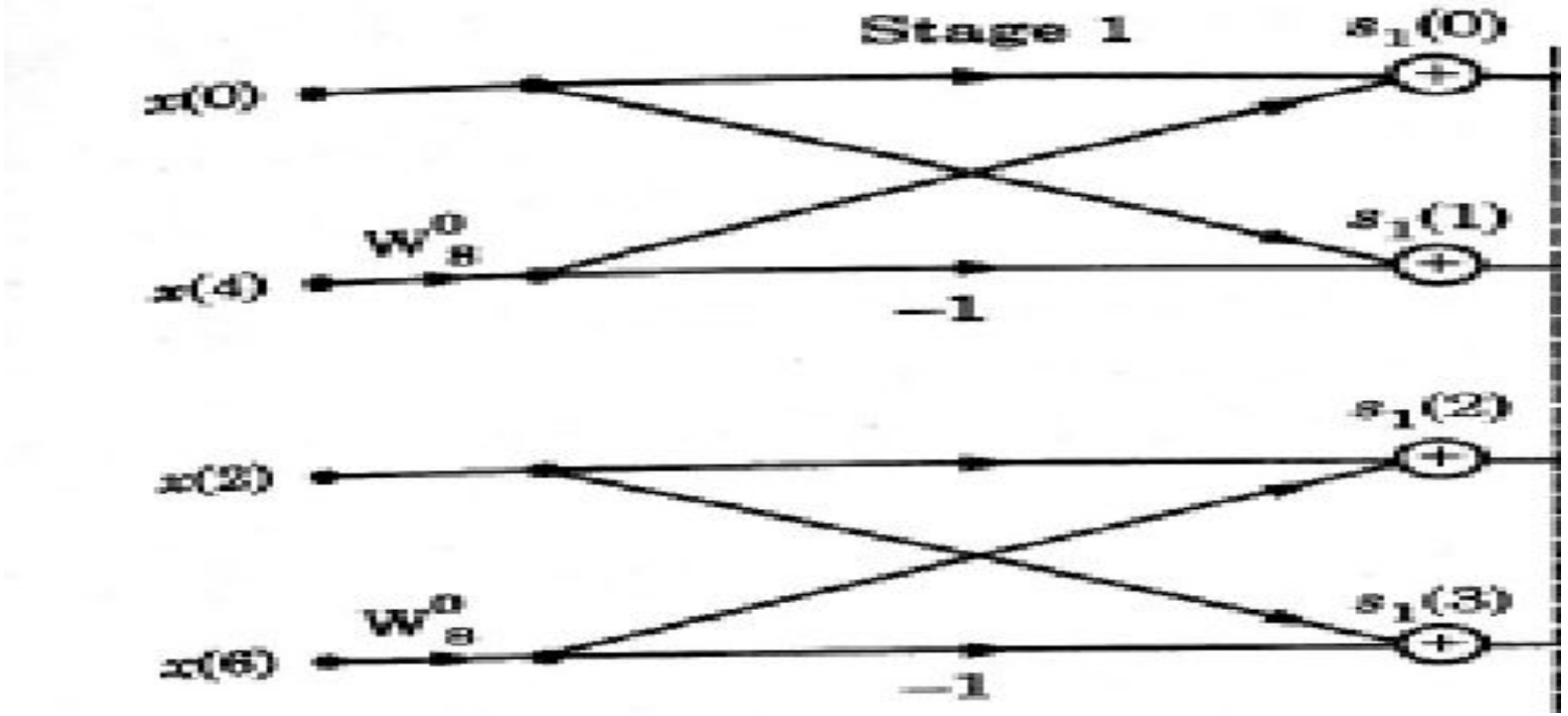
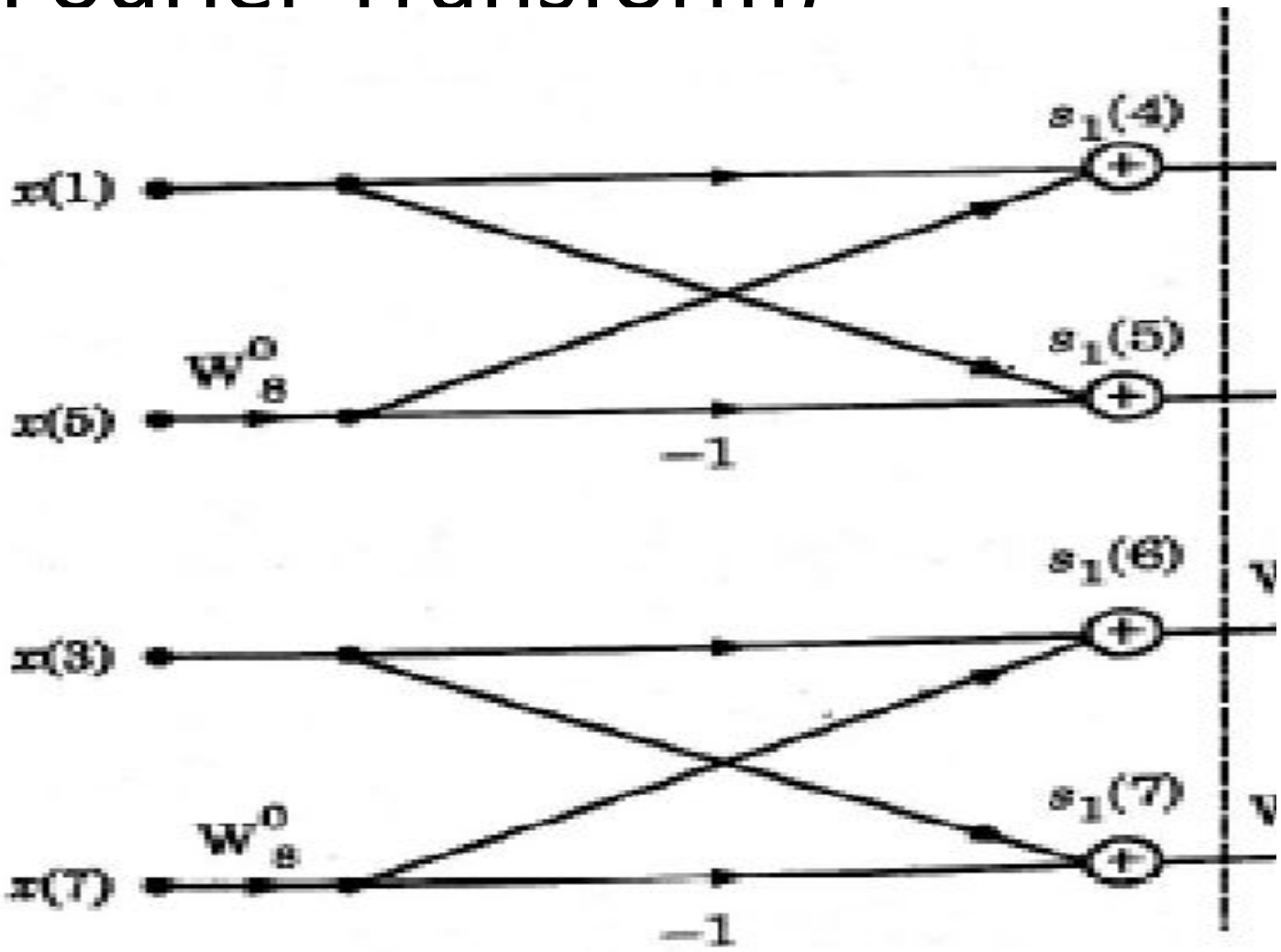


FIGURE 5.12.

FFT(Fast Fourier Transform)



FFT(Fast Fourier Transform)



FFT(Fast Fourier Transform)

Output of stage 1

$$s_1(0) = x(0) + W_8^0\, x(4) = \frac{1}{2} + 1(0) = \frac{1}{2}$$

$$s_1(1) = x(0) - W_8^0\, x(4) = \frac{1}{2} - 1(0) = \frac{1}{2}$$

$$s_1(2) = x(2) + W_8^0\, x(6) = \frac{1}{2} + 1(0) = \frac{1}{2}$$

$$s_1(3) = x(2) - W_8^0\, x(6) = \frac{1}{2} - 1(0) = \frac{1}{2}$$

FFT(Fast Fourier Transform)

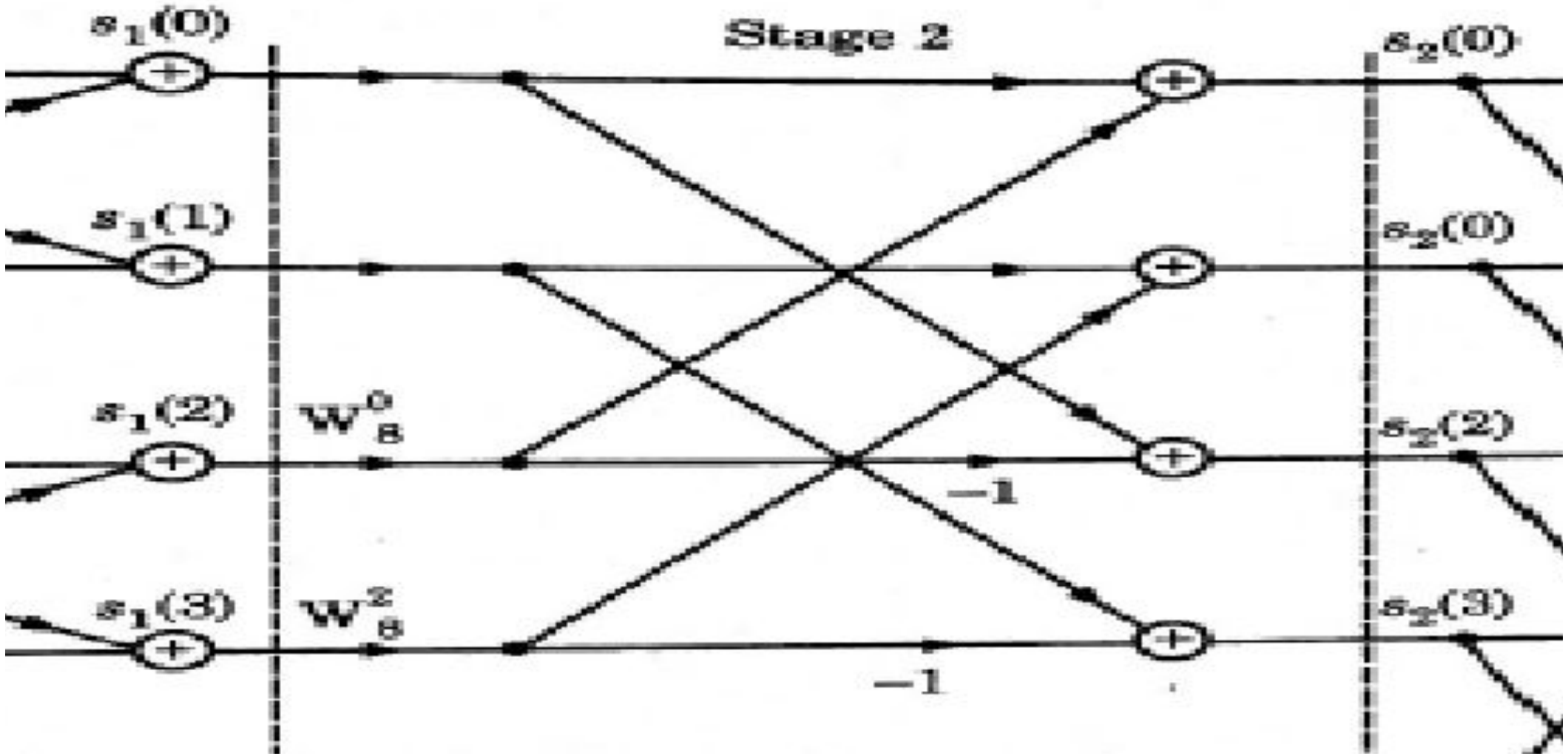
$$s_1(4) = x(1) + W_8^0\, x(5) = \frac{1}{2} + 1 \cdot (0) = \frac{1}{2}$$

$$s_1(5) = x(1) - W_8^0\, x(5) = \frac{1}{2} - 1 \cdot (0) = \frac{1}{2}$$

$$s_1(6) = x(3) + W_8^0\, x(7) = \frac{1}{2} + 1 \cdot (0) = \frac{1}{2}$$

$$s_1(7) = x(3) - W_8^0\, x(7) = \frac{1}{2} - 1 \cdot (0) = \frac{1}{2}$$

FFT(Fast Fourier Transform)



FFT(Fast Fourier Transform)

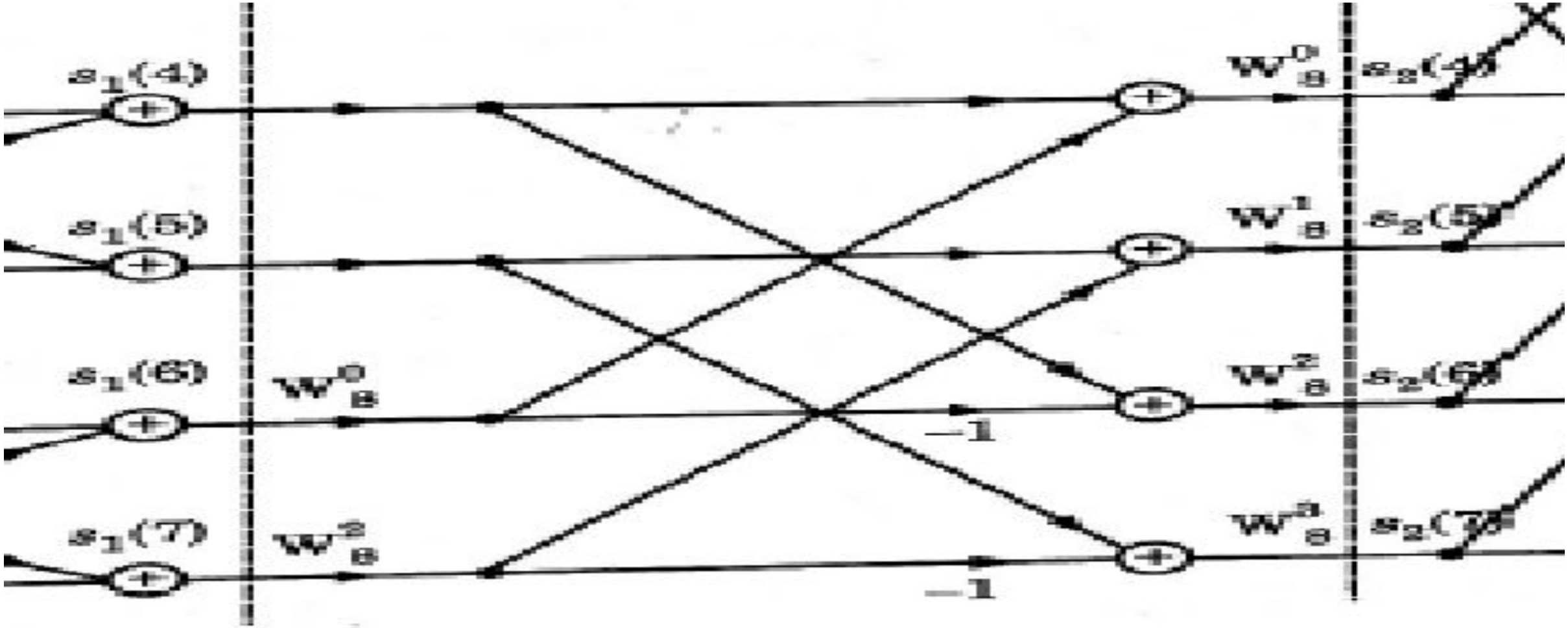


FIGURE 5.12.

FFT(Fast Fourier Transform)

Output of stage 2

$$s_2(0) = s_1(0) + W_8^0 s_1(2) = \frac{1}{2} + 1 \cdot \left(\frac{1}{2}\right) = 1$$

$$s_2(1) = s_1(1) + W_8^2 s_1(3) = \frac{1}{2} - j\,\frac{1}{2}$$

$$s_2(2) = s_1(0) - W_8^0 s_1(2) = \frac{1}{2} - \frac{1}{2} = 0$$

$$s_2(3) = s_1(1) - W_8^2 s_1(3) = \frac{1}{2} + j\,\frac{1}{2}$$

$$s_2(4) = s_1(4) + W_8^0 s_1(6) = \frac{1}{2} + \frac{1}{2} = 1$$

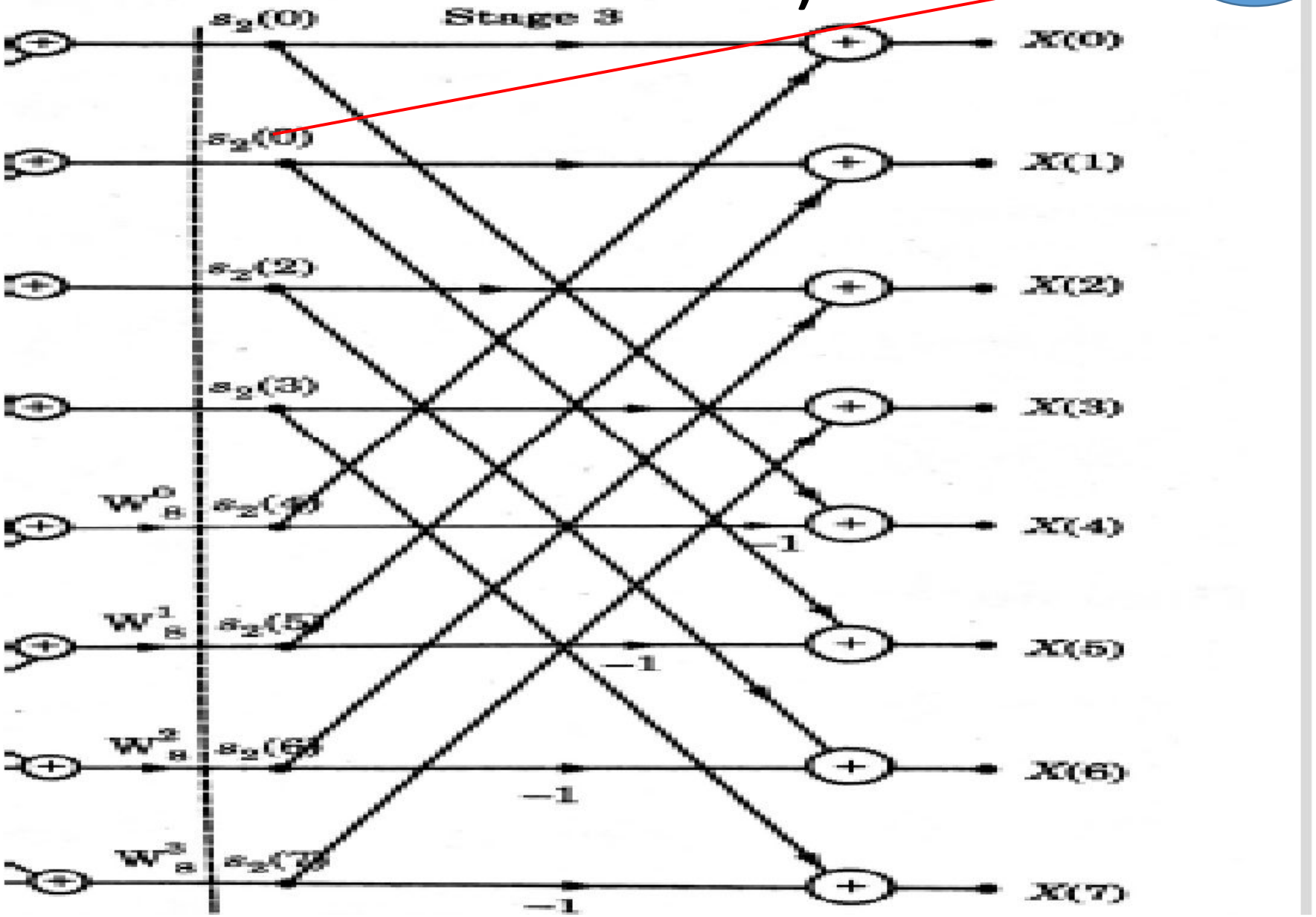
$$s_2(5) = s_1(5) + W_8^2 s_1(7) = \frac{1}{2} - j\,\frac{1}{2}$$

$$s_2(6) = s_1(4) - W_8^0 s_1(6) = \frac{1}{2} - \frac{1}{2} = 0$$

$$s_2(7) = s_1(5) - W_8^2 s_1(7) = \frac{1}{2} + j\,\frac{1}{2}$$

FFT(Fast Fourier Transform)

S2(1)



FFT(Fast Fourier Transform)

Final output

$$X(0) = s_2(0) + W_8^0 \; s_2(4) = 1 + 1 = 2$$

$$X(1) = s_2(1) + W_8^1 \; s_2(5) = \left(\frac{1}{2} - j\frac{1}{2}\right) + (0.707 - \text{j } 0.707)\left(\frac{1}{2} - j\frac{1}{2}\right)$$

$$X(1) = 0.5 - j\,1.207$$

$$X(2) = s_2(2) + W_8^2 \; s_2(6) = 0 + (-j)(0) = 0$$

$$X(3) = s_2(3) + W_8^3 \; s_2(7) = \left(\frac{1}{2} + j\frac{1}{2}\right) + (-0.707 - j\,0.707)\left(\frac{1}{2} + j\frac{1}{2}\right)$$

$$X(3) = \left(\frac{1}{2} + j\frac{1}{2}\right) + (0 - j\,0.707) = 0.5 - j\,0.207$$

$$X(4) = s_2(0) - W_8^0 \; s_2(4) = 1 - 1.1 = 0$$

$$X(5) = s_2(1) - W_8^1 \; s_2(5)$$

$$X(5) = \left(\frac{1}{2} - j\frac{1}{2}\right) - (0.707 - j\,0.707)\left(\frac{1}{2} - j\frac{1}{2}\right)$$

or

$$X(5) = \left(\frac{1}{2} - j\frac{1}{2}\right) - (-0.707\,j)$$

FFT(Fast Fourier Transform)

$$X(5) = 0.5 + j\,0.207$$

Also,

$$X(6) = s_2(2) - W_8^2\, s_2(6) = 0 + j\,.\,(0) = 0$$

$$X(7) = s_2(3) - W_8^3\ \ s_2(7) = \left(\frac{1}{2} + j\frac{1}{2}\right) - (-0.707 - \mathrm{j}\,0.707)\left(\frac{1}{2} + j\frac{1}{2}\right)$$

$$X(7) = \left(\frac{1}{2} + j\frac{1}{2}\right) + 0.707\,j = 0.5 + j\,1.21$$

Thus, we have

$$X(k) = \{X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7)\}$$

or

$$X(k) = \{2,\ 0.5 - j\,1.207,\ 0,\ 0.5 - j\,0.207,\ 0,\ 0.5 + j\,0.207,\ 0,\ 0.5 + j\,1.21\}$$ **Ans.**

FFT(Fast Fourier Transform)

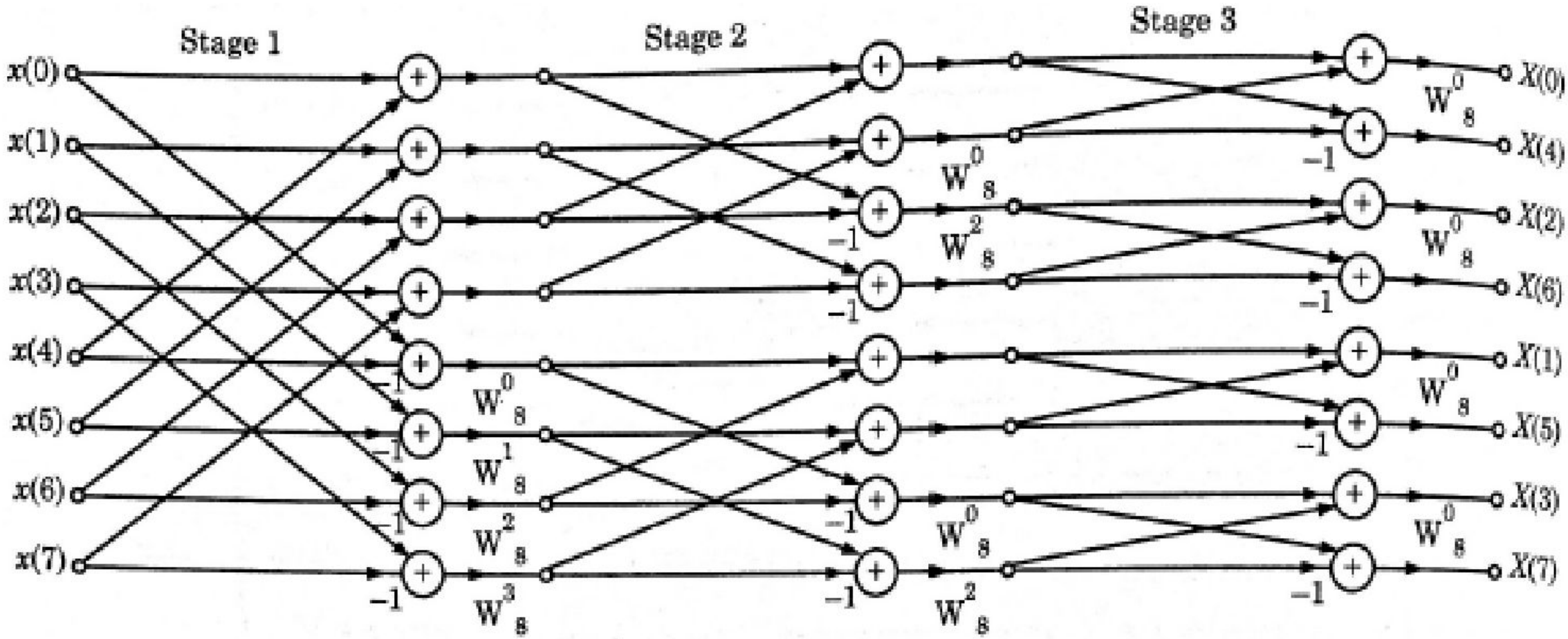


FIGURE 5.19 *Total flow graph for 8-point DIF-FFT*

FFT(Fast Fourier Transform)

Computational Complexity using FFT Algorithm

Firstly, let us calculate the computation complexity required for one butterfly. Let us consider the general structure of butterfly as shown in figure 5.16.

Here, a and b are inputs and A and B are outputs of butterfly. The outputs are given by,

$$A = a + W_N^r\ b \qquad ...(5.73)$$

and

$$B = a - W_N^r\ b \qquad ...(5.74)$$

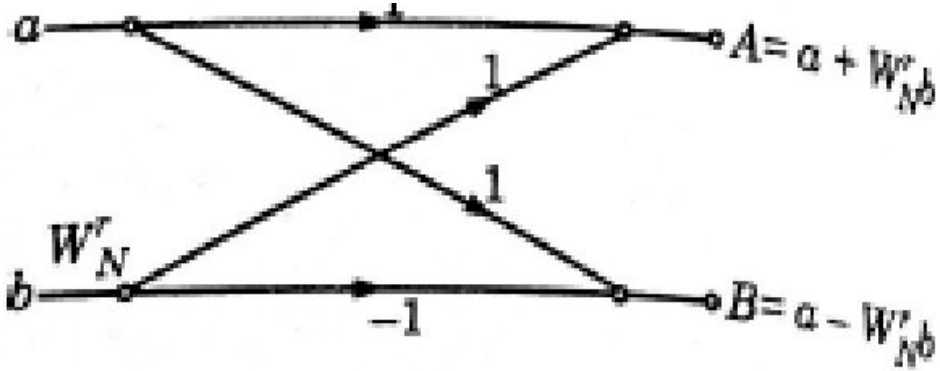


FIGURE 5.16 *General structure of butterfly.*

(*i*) To calculate any output (A or B), we require to multiply input b by twiddle factor W_n^r. Therefore, **one complex multiplication** is required for one butterfly.

(*ii*) To collect output A, one complex addition is required, while to calculate output B, complex subtraction is required as given by equation (5.74). But, the computational complexity of addition and subtraction is same. Therefore, we can say that for one butterfly **two complex additions** are required.

FFT(Fast Fourier Transform)

By analysing, different figures:
For N point DFT, at each stage, N/2 butterflies are required.

FFT(Fast Fourier Transform)

Complex multiplications

At each stage, there are $\frac{N}{2}$ butterflies. Total number of stages are $\log_2 N$. Also, for each butterfly, one complex multiplication is required.

Therefore,

$$\text{Total complex multiplications} = \frac{N}{2} \log_2 N \qquad \ldots(5.75)$$

FFT(Fast Fourier Transform)

Complex additions

Total number of stages are $\log_2 N$. At each stage, $\frac{N}{2}$ butterflies are required. Also, for each butterfly, 2 complex additions are required

Therefore, total complex additions $= 2 \times \frac{N}{2} \log_2 N = N \log_2 N$

FFT(Fast Fourier Transform)

Number of points N	Direct Computation Complex Multiplications N^2	Complex Additions $N(N-1)$	Using FFT Complex Multiplications $\frac{N}{2}\log_2 N$	Complex Additions $N\log_2 N$
4	16	12	4	8
8	64	56	12	24
16	256	240	32	64

Thus Computational complexity of DFT is tremendous than FFT.