□ **Solution**

To find DTFT of $x(n) = \left(\frac{1}{2}\right)^n u(n)$:

$$X(\omega) \triangleq \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

$$\Rightarrow \quad X(\omega) = \sum_{n=0}^{\infty} \left(\frac{1}{2}\right)^n e^{-j\omega n}$$

Recall the formula: $$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}, \quad |a| < 1$$

Hence, $$X(\omega) = \frac{1}{1 - \frac{1}{2}e^{-j\omega}}$$

To find $X_1(k)$:

$$X_1(k) = X(\omega)\big|_{\omega=\frac{2\pi k}{10}}, \quad 0 \leqslant k \leqslant 9$$

$$\Rightarrow \quad X_1(k) = \frac{1}{1 - \frac{1}{2}e^{-j\frac{2\pi k}{10}}}, \quad 0 \leqslant k \leqslant 9$$

$$= \frac{1}{1 - \frac{1}{2}W_{10}^k}$$

Recall the DFT pair:

$$a^n \underset{N\text{–point}}{\overset{\text{DFT}}{\longleftrightarrow}} \frac{1 - a^N}{1 - aW_N^k}$$

$$\Rightarrow \quad \left(\frac{1}{2}\right)^n \underset{10\text{–point}}{\overset{\text{DFT}}{\longleftrightarrow}} \frac{1 - \left(\frac{1}{2}\right)^{10}}{1 - \frac{1}{2}W_{10}^k}$$

$$\Rightarrow \quad \frac{\left(\frac{1}{2}\right)^n}{1 - \left(\frac{1}{2}\right)^{10}} \underset{10\text{–point}}{\overset{\text{DFT}}{\longleftrightarrow}} \frac{1}{1 - \frac{1}{2}W_{10}^k}$$

Hence the IDFT of $$X_1(k) = \frac{1}{1 - \frac{1}{2}W_{10}^k}$$

is $$x_1(n) = \frac{\left(\frac{1}{2}\right)^n}{1 - \left(\frac{1}{2}\right)^{10}}, \quad 0 \leqslant n \leqslant 9$$

3.12 Fast Fourier Transform

The *fast Fourier transform* (FFT) refers to algorithms that compute the discrete Fourier transform (DFT) in a numerically efficient manner. Many such algorithms are available, however we will present only two such algorithms: decimation-in-time and decimation-in-frequency.

3.12.1 Decimation-in-time FFT

The decimation-in-time algorithm uses the *divide and conquer* approach. In the following presentation, the number of points is assumed as a power of 2, that is, $N = 2^p$. The decimation-in-time approach is one of breaking the N-point transform into two $\frac{N}{2}$-point transforms, then breaking each $\frac{N}{2}$-point transform into two $\frac{N}{4}$-point transforms, and continuing this process until two-point DFTs are obtained. In other words, the N-point DFT is performed as several 2-point DFTs.

Let $x(n)$ represent a sequence of length N, where N is a power of 2. Decimate this sequence into two sequences of length $\frac{N}{2}$, one composed of even-indexed values of $x(n)$ and the other of odd-indexed values of $x(n)$:

Given sequence: $x(0),\ x(1), \cdots\cdots x(N-1)$

Even indexed: $x(0),\ x(2), \cdots\cdots x(N-2)$

Odd indexed: $x(1),\ x(3), \cdots\cdots x(N-1)$

We know that the DFT of an N-point sequence is given by

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

Decomposing the sum into two parts, one for the even and one for the odd-indexed values, we get

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

Letting $n = 2r$ in the first summation and $n = 2r+1$ in the second summation, the above equation can be written as

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

Since, $W_N^{2rk} = e^{-j\frac{2\pi}{N}\times 2rk} = e^{-j\frac{2\pi rk}{N/2}} = W_{N/2}^{rk}$

the above equation may be written as

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

$$= G(k) + W_N^k H(k), \qquad 0 \leqslant k \leqslant \frac{N}{2} - 1 \tag{3.22}$$

where $G(k)$ and $H(k)$ are $\frac{N}{2}$-point DFTs of even-indexed samples and odd-indexed samples, respectively. These DFTs yield

$$G(0), G(1), \cdots G\left(\frac{N}{2}-1\right) \quad \text{and} \quad H(0), H(1), \cdots H\left(\frac{N}{2}-1\right)$$

For the overall N-point DFT, we need $G(k)$ and $H(k)$ for $k = 0, 1, \cdots \frac{N}{2} - 1, \cdots N - 1$. However, $G(k)$ and $H(k)$ are periodic with period $\frac{N}{2}$, and so for $k = \frac{N}{2}, \cdots N - 1$, the values are same as those for $k = 0, 1, \cdots \frac{N}{2} - 1$.

Thus, the equations needed to compute N-point DFT of $x(n)$ are as follows:

$$X(k) = G(k) + W_N^k H(k), \quad 0 \leqslant k \leqslant \frac{N}{2} - 1 \tag{3.23}$$

$$X(k) = G\left(k + \frac{N}{2}\right) + W_N^k H\left(k + \frac{N}{2}\right), \quad \frac{N}{2} \leqslant k \leqslant N - 1 \tag{3.24}$$

The flow diagram for the first stage of the decomposition is shown in Fig. 3.9.

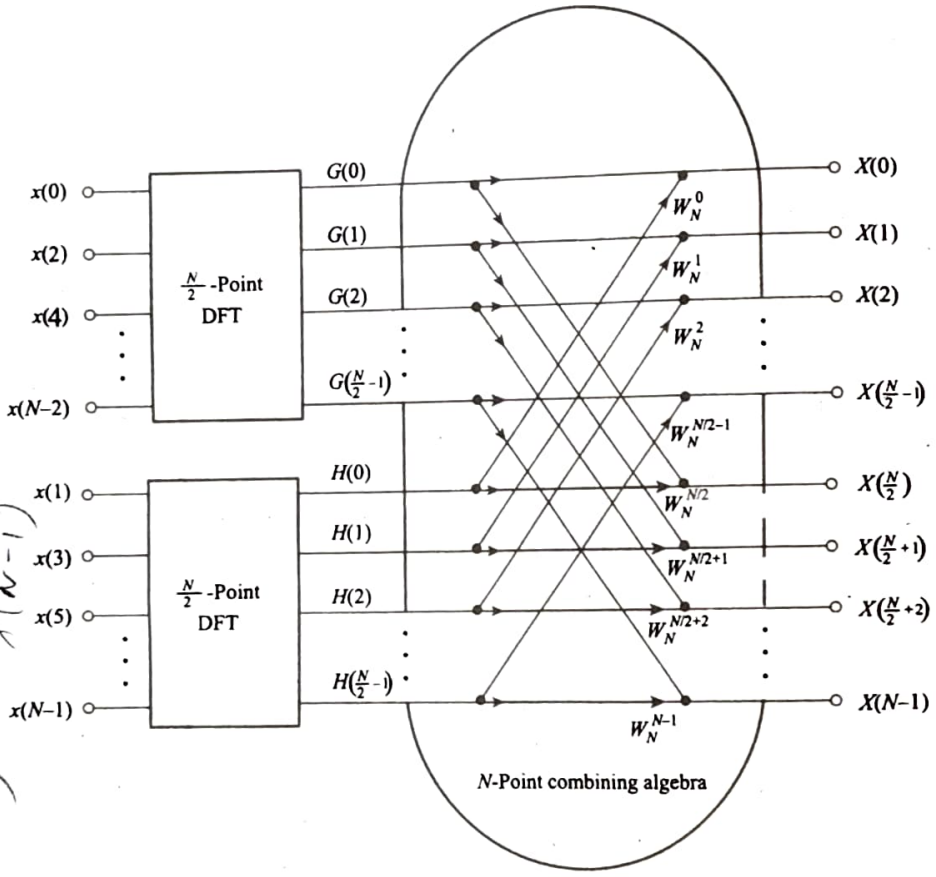


Fig. 3.9 First stage of the decomposition for the decimation-in-time FFT.

The total number of complex multiplications, η_1, required to evaluate the N-point transform with this first decimation is

$$\eta_1 = \left(\frac{N}{2}\right)^2 + \left(\frac{N}{2}\right)^2 + N = N + \frac{N^2}{2}$$

The first term on the right-side of the above equation is the number of complex multiplications for the direct calculation of $\frac{N}{2}$-point DFT of even-numbered samples, the second term is the number of complex multiplications for the direct calculation of $\frac{N}{2}$-point DFT of odd-numbered samples, and the third term is the number of complex multiplications required for the N-point combining algebra.

Each of the $\frac{N}{2}$-point sequences are further decimated into two sequences of length $\frac{N}{4}$ as shown in Fig. 3.10. Then, we get

$$\begin{aligned} G(k) &= \sum_{r=0}^{\frac{N}{2}-1} g(r) W_{N/2}^{kr} \\ &= \sum_{l=0}^{\frac{N}{4}-1} g(2l) W_{N/2}^{2kl} + \sum_{l=0}^{\frac{N}{4}-1} g(2l+1) W_{N/2}^{k(2l+1)} \\ &= \sum_{l=0}^{\frac{N}{4}-1} g(2l) W_{N/4}^{kl} + W_{N/2}^{k} \sum_{l=0}^{\frac{N}{4}-1} g(2l+1) W_{N/4}^{kl} \\ &= A(k) + W_{N/2}^{k} B(k), \quad 0 \leqslant k \leqslant \frac{N}{4} - 1 \end{aligned}$$

since $A(k)$ and $B(k)$ are periodic with a period equal to $\frac{N}{4}$, we may find $\frac{N}{2}$-point transform $G(k)$ using the equation given below.

$$G(k) = \begin{cases} A(k) + W_{N/2}^{k} B(k), & 0 \leqslant k \leqslant \frac{N}{4} - 1 \\ A\left(k + \frac{N}{4}\right) + W_{N/2}^{k} B\left(k + \frac{N}{4}\right), & \frac{N}{4} \leqslant k \leqslant \frac{N}{2} - 1 \end{cases}$$

Similarly,

$$H(k) = \begin{cases} C(k) + W_{N/2}^{k} D(k), & 0 \leqslant k \leqslant \frac{N}{4} - 1 \\ C\left(k + \frac{N}{4}\right) + W_{N/2}^{k} D\left(k + \frac{N}{4}\right), & \frac{N}{4} \leqslant k \leqslant N - 1 \end{cases}$$

Often, in a flow diagram we write W_N^{2k} (as a matter of convenience) instead of $W_{\frac{N}{2}}^{k}$.

The flow diagram after second decimation-in-time is shown in Fig. 3.10. The total number of complex multiplications after second decimation is

$$\eta_2 = 4\left(\frac{N}{4}\right)^2 + 2\left(\frac{N}{2}\right) + N = \frac{N^2}{4} + 2N$$

The first term on the right-side of the above equation accounts for the number of complex multiplications necessary for the direct computation of four $\frac{N}{4}$-point DFTs, while the second and third terms represent the number of complex multiplications needed for combining algebras for the first and second decompositions.

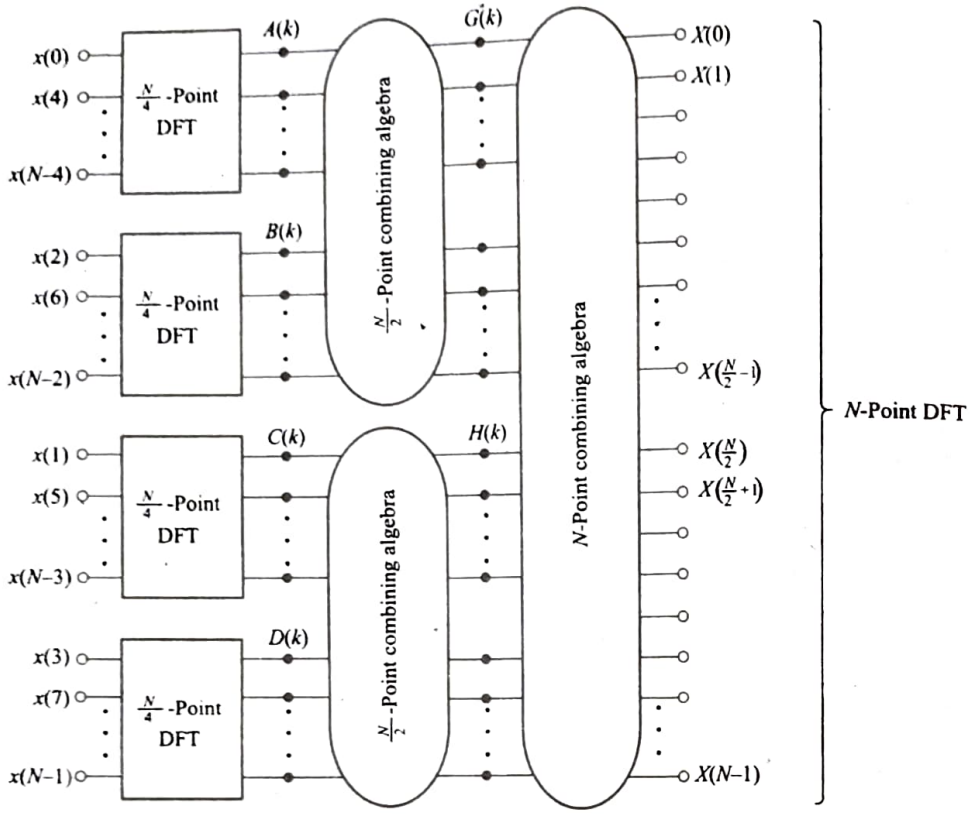


Fig. 3.10 Second stage of the decimation-in-time FFT.

We continue this process of decimation until we end up with 2-point DFTs. This would require $(p-1)$ stages when $N = 2^p$ or $p = \log_2 N$. The total number of complex multiplications then becomes $N \log_2 N$ (there are $\log_2 N$ stages and each stage has N complex multiplications). Thus, the total number of complex multiplications have reduced from N^2 (direct evaluation) to $N \log_2 N$ (after decimation). The final conceptual decomposition for the decimation-in-time FFT is shown in Fig. 3.11.

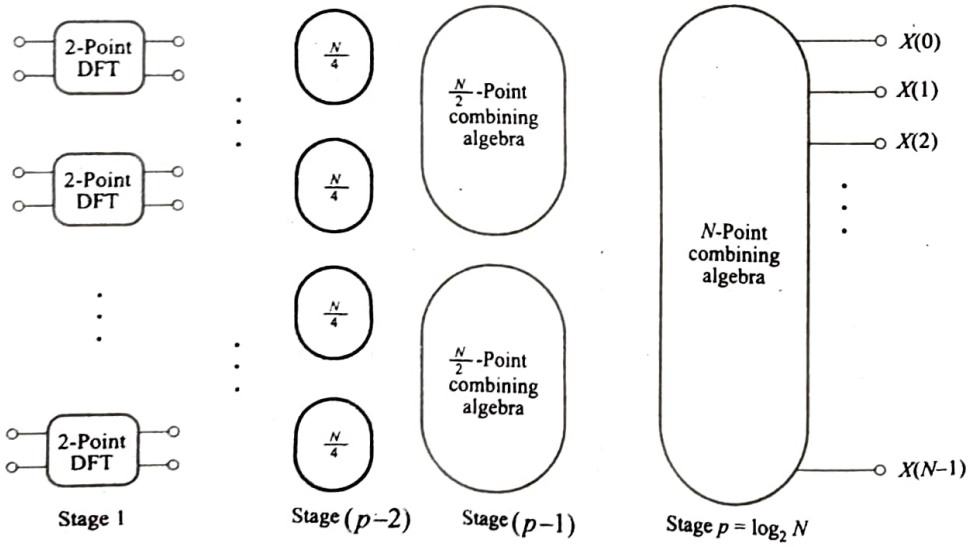


Fig. 3.11 Final conceptual decomposition for computation of N-point DFT using decimation-in-time FFT.

The flow diagram for an 8-point decimation-in-time FFT algorithm is shown in Fig. 3.12.

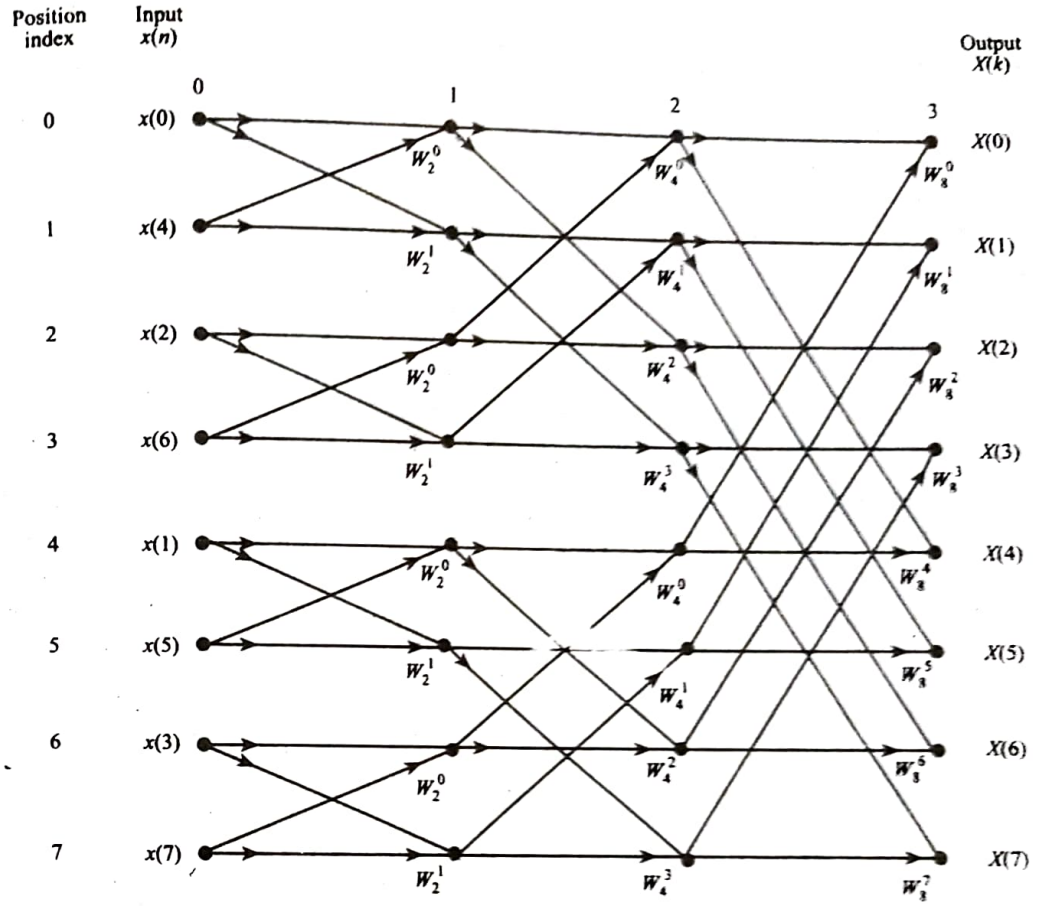


Fig. 3.12 The flow graph for an 8-point decimation-in-time FFT algorithm.

Fig. 3.13 shows the specimen butterfly or the basic building block for decimation-in-time FFT.

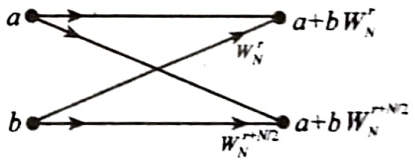


Fig. 3.13 Sample butterfly for decimation-in-time FFT.

The following observations can be made from Fig. 3.12.

1. The input data appears in the bit reversed order:

Position		*Binary equivalent*	*Bit reversed*	*Sequence*
5	$\longrightarrow$	101	101	$x(5)$
6	$\longrightarrow$	110	011	$x(3)$

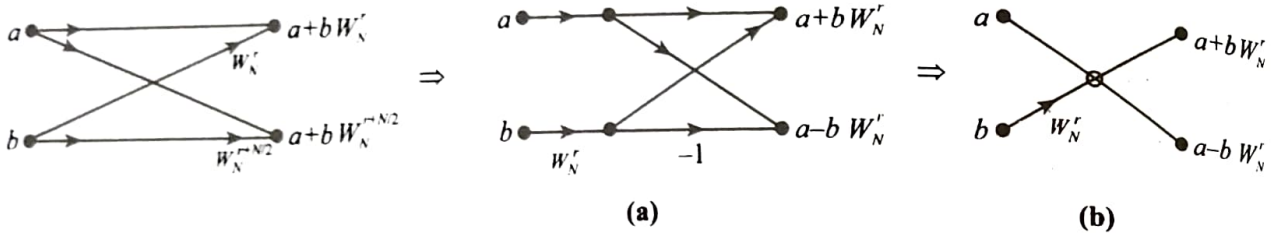
2. The basic computational block in the flow diagram is called a *butterfly*. The power r of W_N is a variable and depends upon the position of the butterfly in the flow diagram.
3. The frequency-domain values are in the normal order.

3.12.2 Further reduction: Cooley-Tukey algorithm

The sample butterfly configuration shown in Fig. 3.13 can be further simplified to reduce the number of complex multiplications per butterfly by one. This is possible due to the fact,

$$W_N^{r+N/2} = e^{-j\frac{2\pi}{N}(r+\frac{N}{2})} = e^{-j\frac{2\pi r}{N}} e^{-j\pi} = -W_N^r$$

Thus, the sample butterfly shown in Fig. 3.13 can be further simplified into a butterfly shown in Fig. 3.14.



(a) (b)

Fig. 3.14 Modified sample butterfies (a) and (b).

The number of complex multiplications required for calculating DIT-FFT algorithm based on Cooley-Tukey is $\frac{N}{2}\log_2 N$. Also, since there are two additions per butterfly, $\frac{N}{2}$ butterflies per stage and $\log_2 N$ stages, the total number of complex additions needed for computing a DFT using DIT-FFT algorithm is $N \log_2 N$.

The reduced complete 8-point DIT-FFT algorithm is shown in Fig. 3.15.

3.12.3 In-place computations

Other than speedy computations, another advantage that naturally comes from this algorithm is the reduction in storage requirement. Referring Fig. 3.15, we find that the algorithm is performed in stages. Each stage involves only $\frac{N}{2}$ butterflies and these butterflies operate on a pair of complex numbers and produce another pair of complex numbers. Once the output pair is calculated, we don't need the input pair anymore. Hence, the output pair can be stored in the same locations as the input pair. Therefore, for an N-point DFT, we need N complex registers or $2N$ real registers for the first stage. The same registers will be used for other stages to follow. This is called *in-place* computations in the DSP literature. If we refer to 8-point FFT, the butterfly computations are performed on the following pairs: $x(0)$ and $x(4)$, $x(2)$ and $x(6)$, $x(1)$ and $x(5)$, and $x(3)$ and $x(7)$. To begin with let us label the input samples as follows:

$$\begin{aligned} X_0(0) &= x(0) & X_0(4) &= x(1) \\ X_0(1) &= x(4) & X_0(5) &= x(5) \\ X_0(2) &= x(2) & X_0(6) &= x(3) \\ X_0(3) &= x(6) & X_0(7) &= x(7) \end{aligned}$$

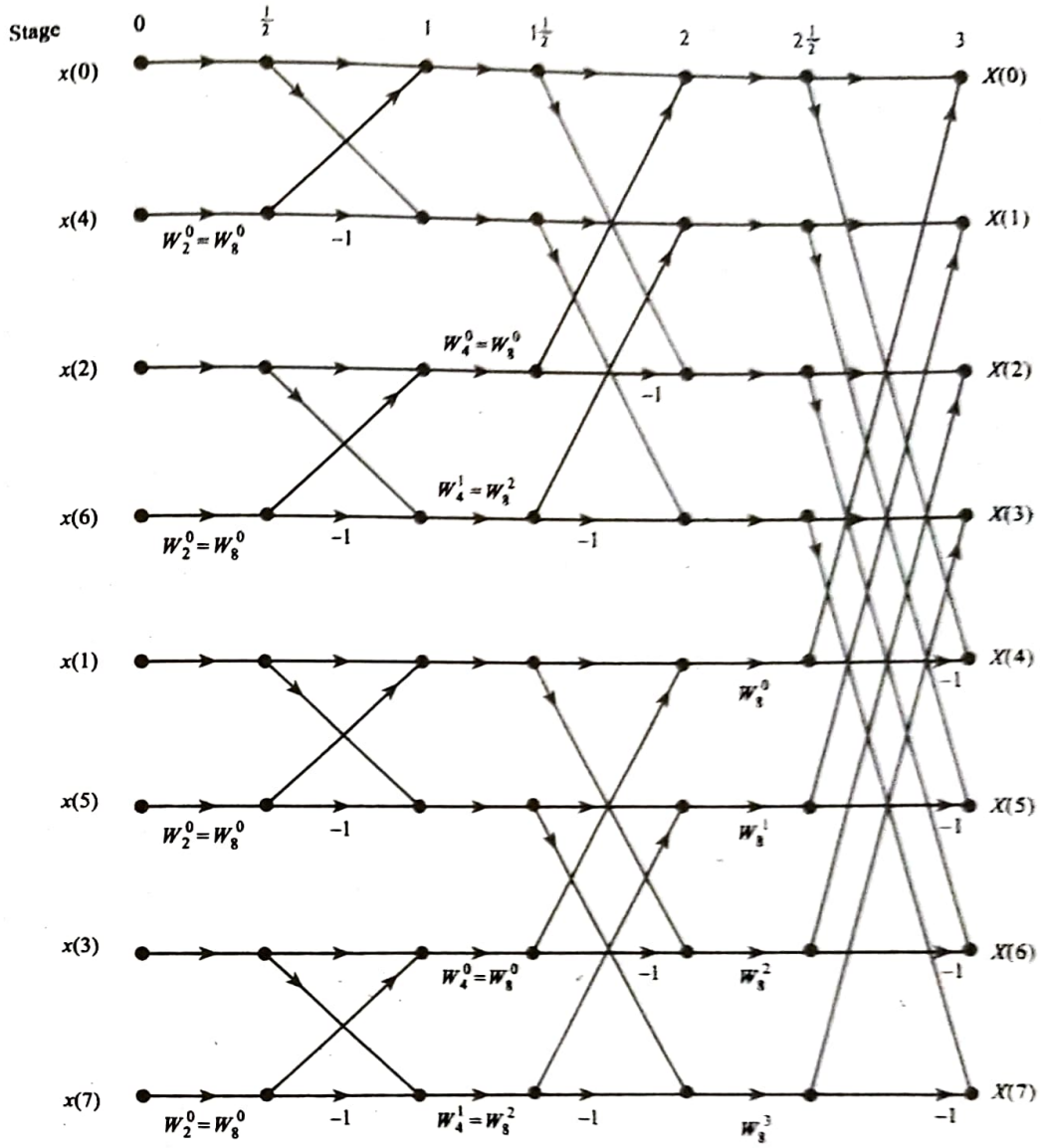


Fig. 3.15 The reduced flow graph for an 8-point DIT FFT algorithm.

Here, $X_m(n)$ refers to the data of the m^{th} stage. The butterflies in the first stage will yield the data $X_1(0), X_1(1) \cdots X_1(7)$. Next, butterfly computations will be performed on the output of the first stage given by the pairs $X_1(0)$ and $X_1(2)$, $X_1(1)$ and $X_1(3)$, $X_1(4)$ and $X_1(6)$, and $X_1(5)$ and $X_1(7)$. The output of the second stage is the data set $X_2(0), X_2(1) \cdots X_2(7)$. The butterflies in the third stage will be computed on the data pairs $X_2(0)$ and $X_2(4)$, $X_2(2)$ and $X_2(6)$, $X_2(1)$ and $X_2(5)$, and $X_2(3)$ and $X_2(7)$ to give the final DFT values.

The scheme of evaluation is pictorially shown in Fig. 3.16. As shown in Fig. 3.16, if we reverse the order of the bits, then we get the location where the data is to be stored. For example, $x(4)$ is represented as $x(100)$. If we reverse the bits, we get $x(001)$, which means that $x(4)$ will be stored in $X_0(1)$. Hence, if the input data samples $x(n)$ are in bit-reversed order, the final DFT is obtained in the natural order. It is easy to show that if the input data samples are stored in the normal order, then the final DFT is obtained in bit-reversed order.

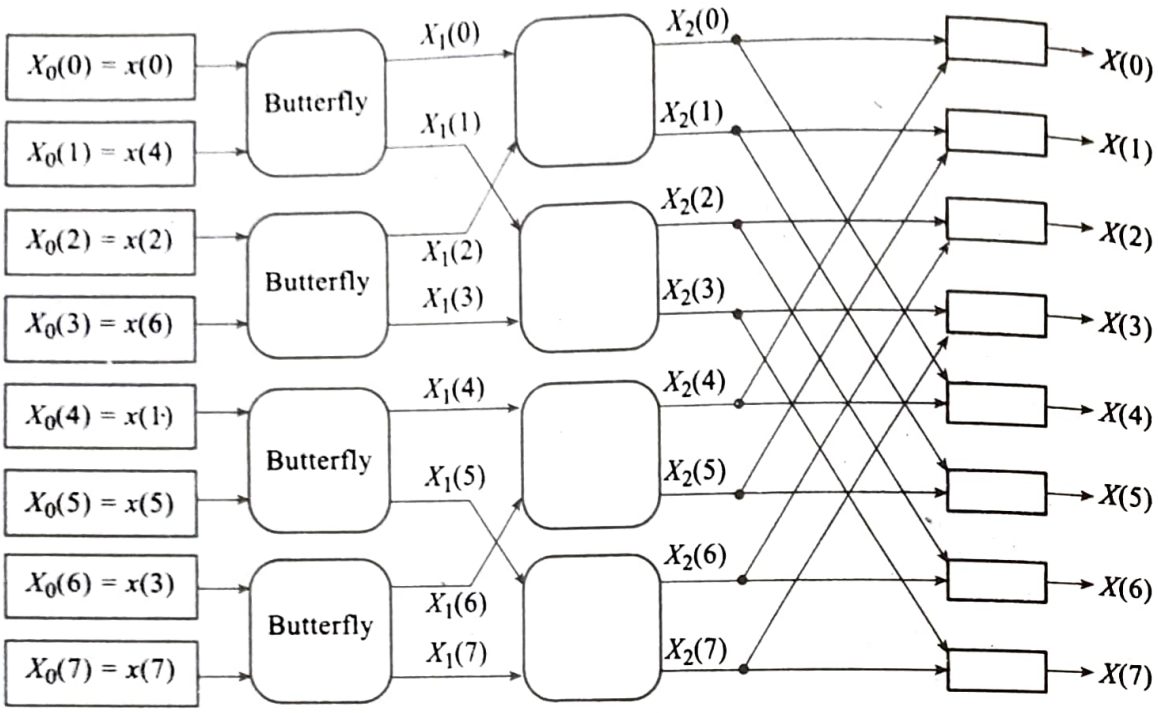


Fig. 3.16 Three-level computation diagram for an 8-point DIT-FFT.

Example 3.50 Find the 8-point DFT of a sequence $x(n) = (1, 1, 1, 1, 0, 0, 0, 0)$ using DIT-FFT radix-2 algorithm. Use the butterfly diagram given in Fig. Ex.3.50.

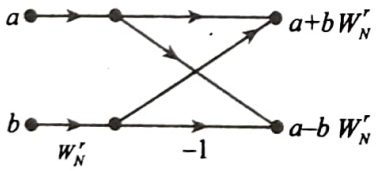


Fig. Ex.3.50 Specified butterfly for Example.3.50.

□ Solution

The scale-factors W_8^r are as follows:

$$W_8^0 = 1 \qquad W_8^4 = -1$$

$$W_8^1 = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \qquad W_8^5 = -\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$$

$$W_8^2 = -j \qquad W_8^6 = +j$$

$$W_8^3 = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \qquad W_8^7 = \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$$

The decimation-in-time FFT algorithms are developed on 2 as the base or radix, and where the number of samples N and base 2 are related by the expression $N = 2^p$. This is the reason for calling the algorithms developed as radix-2 FFT algorithms.

The flow diagram is as shown in Fig. Ex. 3.50(b).

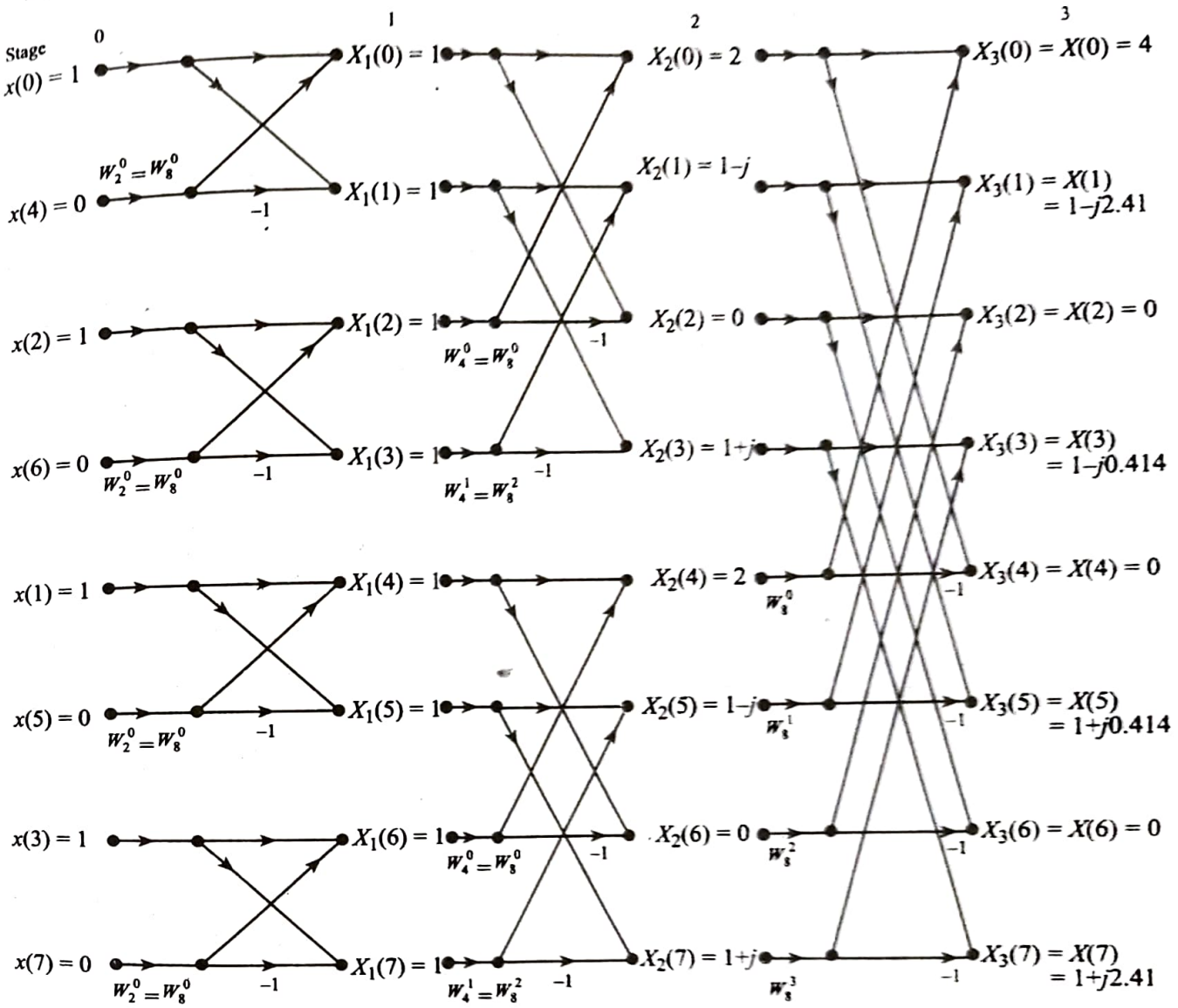


Fig. Ex.3.50(a) The flow graph for 8-point DFT using DIT-FFT algorithm.

To compute the output of first stage:

$$
\begin{aligned}
X_1(0) &= 1 + 0 \times W_8^0 = 1 \\
X_1(1) &= 1 - 0 \times W_8^0 = 1 \\
X_1(2) &= 1 + 0 \times W_8^0 = 1 \\
X_1(3) &= 1 - 0 \times W_8^0 = 1 \\
X_1(4) &= 1 + 0 \times W_8^0 = 1 \\
X_1(5) &= 1 - 0 \times W_8^0 = 1 \\
X_1(6) &= 1 + 0 \times W_8^0 = 1 \\
X_1(7) &= 1 - 0 \times W_8^0 = 1
\end{aligned}
$$

To compute the output of second stage:

$$
\begin{aligned}
X_2(0) &= 1 + W_8^0 \times 1 = 2 \\
X_2(1) &= 1 + 1 \times W_8^2 = 1 - j \\
X_2(2) &= 1 - W_8^0 \times 1 = 0 \\
X_2(3) &= 1 - 1 \times W_8^2 = 1 + j \\
X_2(4) &= 1 + W_8^0 \times 1 = 2 \\
X_2(5) &= 1 + W_8^2 \times 1 = 1 - j \\
X_2(6) &= 1 - W_8^0 \times 1 = 0 \\
X_2(7) &= 1 - W_8^2 \times 1 = 1 + j
\end{aligned}
$$

To compute the output of third stage

$$
\begin{aligned}
X_3(0) &= X(0) = 2 + 2W_8^0 = 4 \\
X_3(1) &= X(1) = 1 - j + (1 - j)W_8^1 = 1 - j2.41 \\
X_3(2) &= X(2) = 0 + 0 \times W_8^2 = 0 \\
X_3(3) &= X(3) = (1 + j) + (1 + j)W_8^3 = 1 - j0.414 \\
X_3(4) &= X(4) = 2 - 2W_8^0 = 0 \\
X_3(5) &= X(5) = (1 - j) - (1 - j)W_8^1 = 1 + j0.414 \\
X_3(6) &= X(6) = 0 - 0 \times W_8^2 = 0 \\
X_3(7) &= X(7) = (1 + j) - (1 + j)W_8^3 = 1 + j2.414
\end{aligned}
$$

Since $x(n)$ is a real sequence, we find that the symmetry property: $X(k) = X^*(8 - k)$ is satisfied.

Example 3.51 Find the 8-point DFT of the sequence,

$$x(n) = (1, 2, 3, 4, 4, 3, 2, 1)$$

using DIT-FFT radix-2 algorithm. The basic computational block known as the butterfly should be as shown in Fig. Ex.3.51.

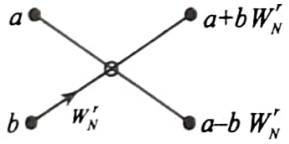


Fig. Ex.3.51 Specified butterfly for Example.3.51.

☐ Solution

The scale-factors W_8^r are as follows:

$$
\begin{aligned}
W_8^0 &= 1 & W_8^4 &= -1 \\
W_8^1 &= \tfrac{1}{\sqrt{2}} - j\tfrac{1}{\sqrt{2}} & W_8^5 &= -\tfrac{1}{\sqrt{2}} + j\tfrac{1}{\sqrt{2}} \\
W_8^2 &= -j & W_8^6 &= +j \\
W_8^3 &= -\tfrac{1}{\sqrt{2}} - j\tfrac{1}{\sqrt{2}} & W_8^7 &= \tfrac{1}{\sqrt{2}} + j\tfrac{1}{\sqrt{2}}
\end{aligned}
$$

Also, recall that:

$$
\begin{aligned}
W_4^0 &= W_8^0 \\
W_4^1 &= W_8^2 \\
W_2^0 &= W_8^0
\end{aligned}
$$

The flow diagram is as shown in Fig. Ex. 3.51(a).

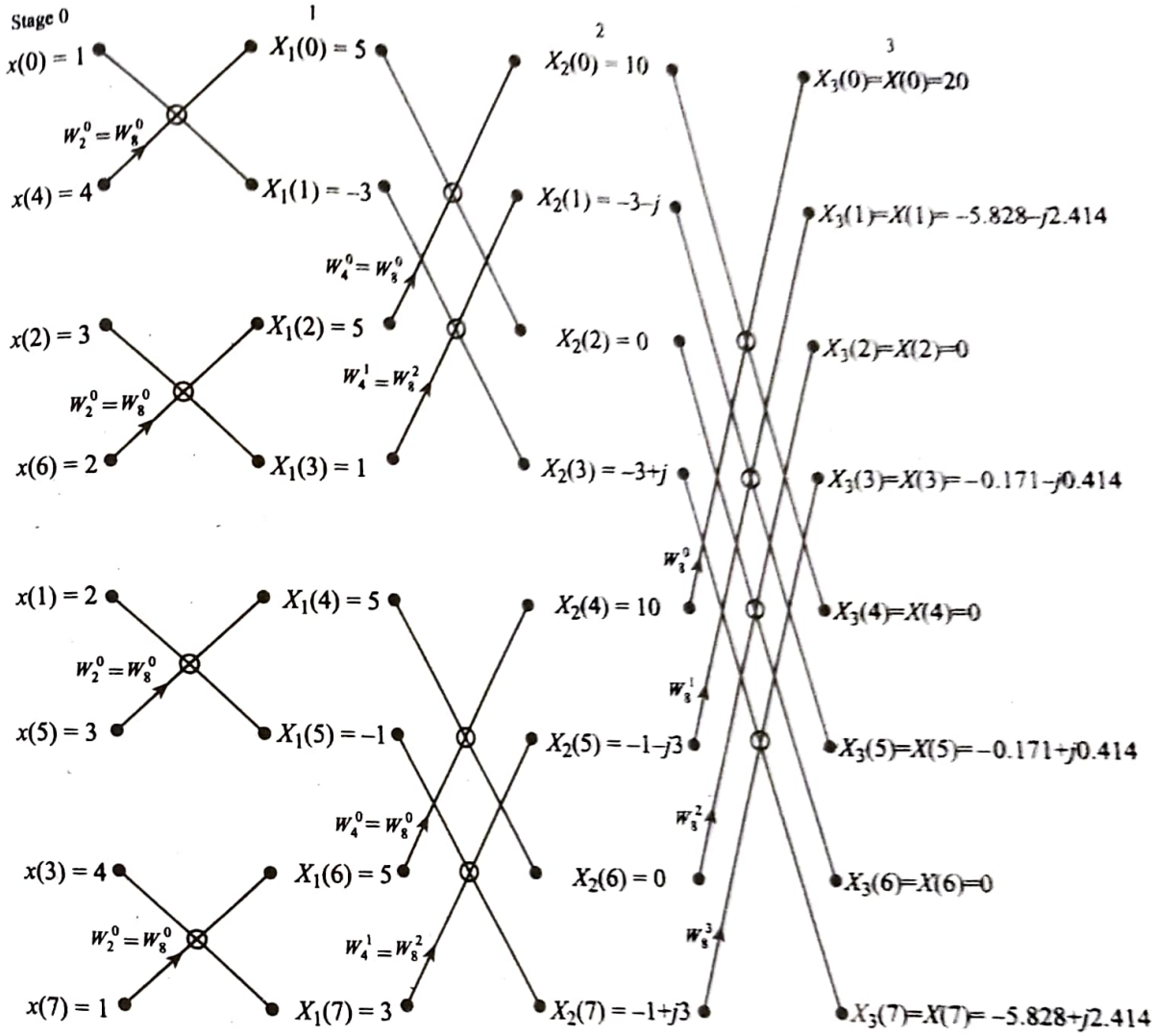


Fig. Ex.3.51(a) Reduced flow graph for an 8-point decimation-in-time FFT.

To compute the output of first stage

$$
\begin{aligned}
X_1(0) &= 1 + 4 \times W_8^0 = 5 \\
X_1(1) &= 1 - 4 \times W_8^0 = -3 \\
X_1(2) &= 3 + 2 \times W_8^0 = 5 \\
X_1(3) &= 3 - 2 \times W_8^0 = 1 \\
X_1(4) &= 2 + 3 \times W_8^0 = 5 \\
X_1(5) &= 2 - 3 \times W_8^0 = -1 \\
X_1(6) &= 4 + 1 \times W_8^0 = 5 \\
X_1(7) &= 4 - 1 \times W_8^0 = 3
\end{aligned}
$$

To compute the output of the second stage

$$
\begin{aligned}
X_2(0) &= 5 + 5 \times W_8^0 = 10 \\
X_2(1) &= -3 + 1 \times W_8^2 = -3 - j \\
X_2(2) &= 5 - 5 \times W_8^0 = 0 \\
X_2(3) &= -3 - 1 \times W_8^2 = -3 + j \\
X_2(4) &= 5 + 5 \times W_8^0 = 10 \\
X_2(5) &= -1 + 3 \times W_8^2 = -1 - j3 \\
X_2(6) &= 5 - 5 \times W_8^0 = 0 \\
X_2(7) &= -1 - 3 \times W_8^2 = -1 + j3
\end{aligned}
$$

To compute the output of the third stage

$$
\begin{aligned}
X_3(0) &= X(0) = 10 + 10 \times W_8^0 = 20 \\
X_3(1) &= X(1) = -3 - j + (-1 - 3j)W_8^1 = -5.828 - j2.414 \\
X_3(2) &= X(2) = 0 + 0 \times W_8^2 = 0 \\
X_3(3) &= X(3) = -3 + j + (-1 + 3j)W_8^3 = -0.171 - j0.414 \\
X_3(4) &= X(4) = 10 - 10 \times W_8^0 = 0 \\
X_3(5) &= X(5) = -3 - j - (-1 - 3j)W_8^1 = -0.171 + j0.414 \\
X_3(6) &= X(6) = 0 - 0 \times W_8^2 = 0 \\
X_3(7) &= X(7) = -3 + j - (-1 + 3j)W_8^3 = -5.828 + j2.414
\end{aligned}
$$

Since $x(n)$ is a real sequence, we find that the symmetry condition: $X(k) = X^*(8-k)$ is satisfied.

Example 3.52 a. Compute the 4-point DFT of the sequence $x(n) = (1, 0, 1, 0)$ using DIT-FFT radix-2 algorithm.

b. Find $x(n)$ for $X(k)$ found in part (a) by two different methods.

□ Solution

The scale-factors are as follows:

$$
\begin{aligned}
& & W_4^0 &= 1, & W_4^1 &= -j \\
\text{Also,} & & W_4^{-0} &= \left(W_4^0\right)^* = 1 \\
\text{and} & & W_4^{-1} &= \left(W_4^1\right)^* = j
\end{aligned}
$$

a. To find 4-point DFT $X(k)$

The signal flow diagram is shown in Fig. Ex.3.52(b). The computational block or the butterfly used is as shown in Fig. Ex.3.52(a), because it is very simple to draw and so is our preferred choice.

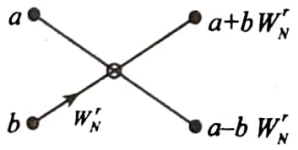


Fig. Ex.3.52(a) Preferred butterfly.

FACTS:

- Since $x(n)$ is a real sequence, the symmetry condition: $X(k) = X^*(N-k)$ must be observed.
- Since $x(n)$ is real and $x(n) = x((-n))_4$, the DFT$\{x(n)\} = X(k)$ must be purely real.

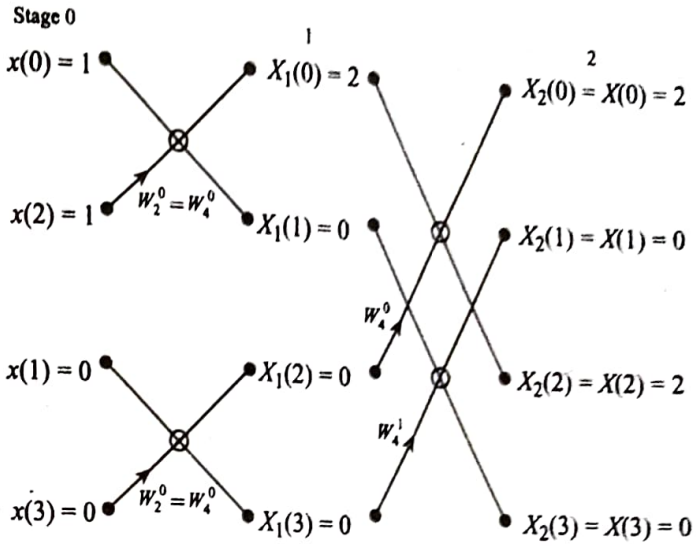


Fig. Ex.3.52(b) Flow graph for computing 4-point DFT.

Hence, $X(k) = (2, 0, 2, 0)$

b. **I method:** We know that

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

The above equation says that DIT-FFT algorithm can be used to find IDFT of $X(k)$ by changing the power of W_N as negative and then dividing the final output by $\frac{1}{N}$.

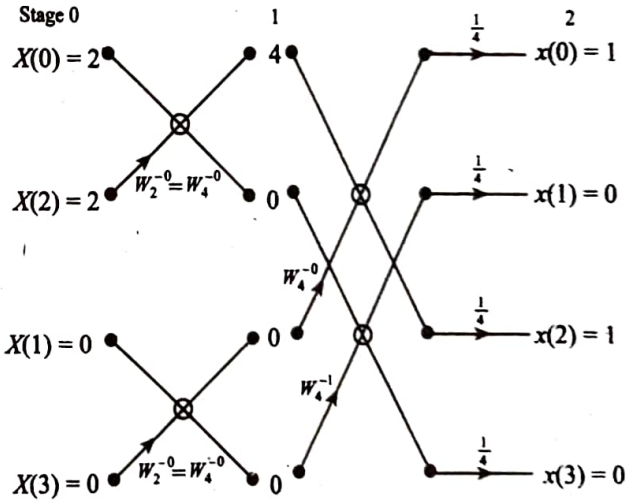


Fig. Ex.3.52(c) Flow graph for computing 4-point IDFT.

Hence, $x(n) = (1, 0, 1, 0)$

II method: We know that

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

Taking conjugates on both the sides, we get

$$\begin{aligned} x^*(n) &= \frac{1}{N}\sum_{k=0}^{N-1} X^*(k) W_N^{+kn} \\ &= \frac{1}{N}\,\text{DFT}\{X^*(k)\} \\ &= \frac{1}{N}\,y(n), \quad \text{where } y(n) = \text{DFT}\{X^*(k)\} \end{aligned}$$

Taking conjugates again, we get

$$x(n) = \frac{1}{N}\,y^*(n)$$

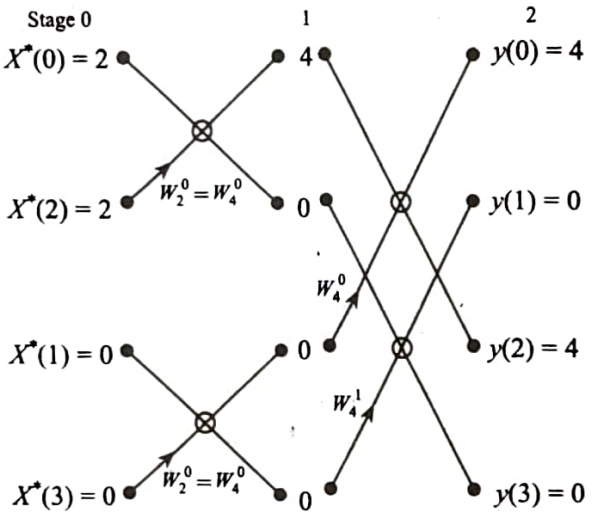


Fig. Ex.3.52(d) Flow graph for computing 4-point DFT of $X(k)$.

n	$x(n) = \frac{1}{4}y^*(n)$
0	$x(0) = \frac{1}{4} \times 4 = 1$
1	$x(1) = \frac{1}{4} \times 0 = 0$
2	$x(2) = \frac{1}{4} \times 4 = 1$
3	$x(3) = \frac{1}{4} \times 0 = 0$

Hence, $x(n) = (1, 0, 1, 0)$

Example 3.53 Given the sequences $x_1(n)$ and $x_2(n)$ below, compute the circular convolution $x_1(n) \circledast_N x_2(n)$ for $N = 4$. Use DIT-FFT algorithm.

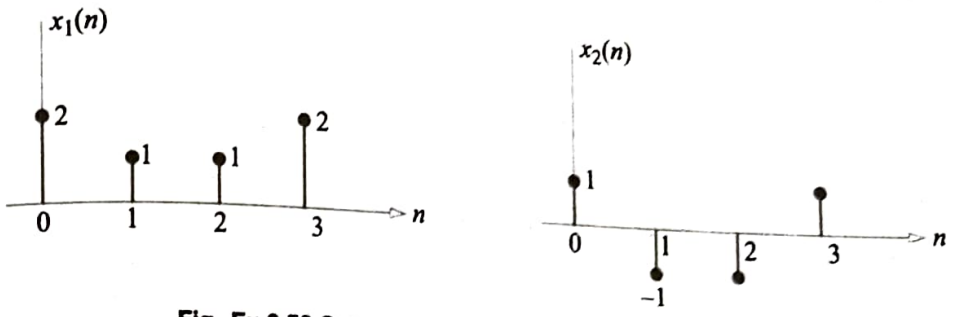


Fig. Ex.3.53 Sequences $x_1(n)$ and $x_2(n)$ for Example.3.53.

Solution

The block diagram used for finding the circular convolution of $x_1(n)$ and $x_2(n)$ using FFTs is shown in Fig. Ex.3.53(a).

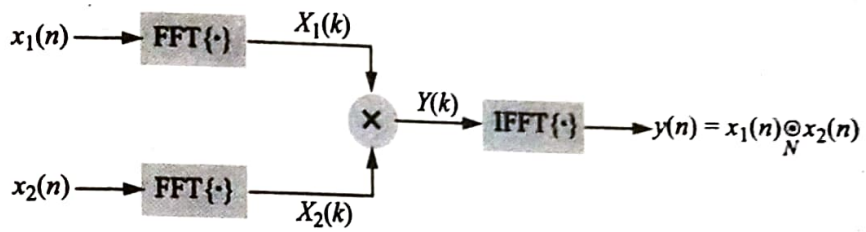


Fig. Ex.3.53(a) Block diagram for computing IDFT.

To find $X_1(k)$:

$$x_1(n) = (2, 1, 1, 2)$$

Let us find $X_1(k)$ using DIT-FFT algorithm. The corresponding signal flow graph is shown in Fig. Ex.3.53(b).

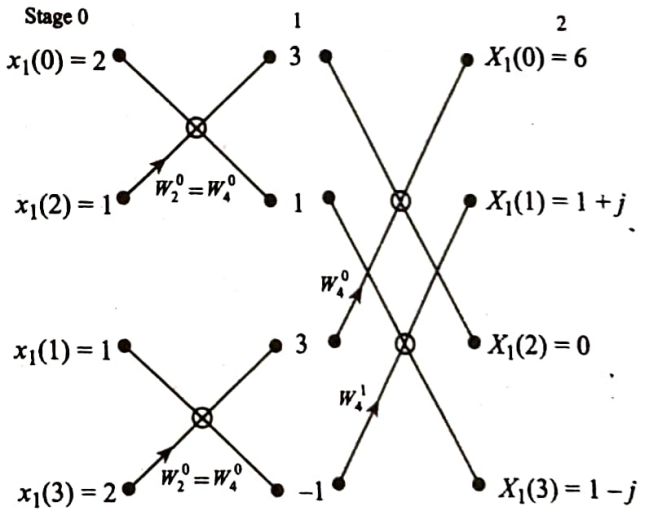


Fig. Ex.3.53(b) Flow graph for computing 4-point DFT.

Hence,

$$X_1(k) = (6, 1 + j, 0, 1 - j)$$

To find $X_2(k)$:

$$x_2(n) = (1, -1, -1, 1)$$

Let us find $X_2(k)$ using DIT-FFT algorithm. The corresponding flow graph is shown in Fig. Ex.3.53(c).

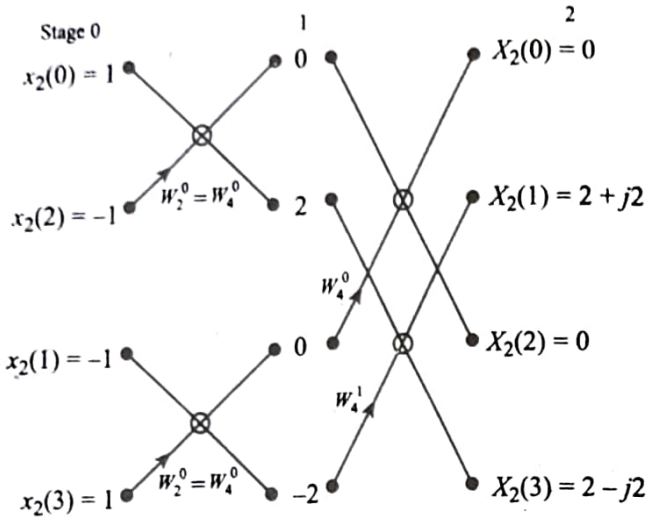


Fig. Ex.3.53(c) Flow graph for computing $X_2(k)$.

Hence, $$X_2(k) = (0, 2 + 2j, 0, 2 - 2j)$$

To find $y(n) = x_1(n) \circledast_4 x_2(n)$

$$\begin{aligned} Y(k) &= X_1(k)X_2(k) \\ &= (6, 1 + j, 0, 1 - j) \times (0, 2 + 2j, 0, 2 - 2j) \\ &= (0, 4j, 0, -4j) \end{aligned}$$

The flow graph for finding IDFT of $Y(k)$ using DIT-FFT algorithm is shown in Fig. Ex.3.53(d).

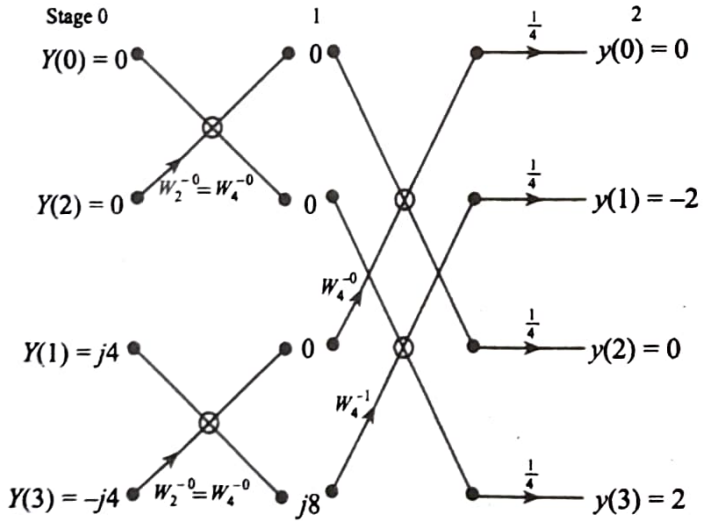


Fig. Ex.3.53(d) Flow graph for computing IDFT of $X_1(k)X_2(k)$.

Thus, we find that

$$y(n) = (0, -2; 0, 2)$$

Example 3.54 Draw the flow diagram for a 16-point radix-2 decimation-in-time FFT algorithm. Label all the multipliers appropriately.

Solution

The flow diagram for a 16-point radix-2 DIT-FFT algorithm is shown below.

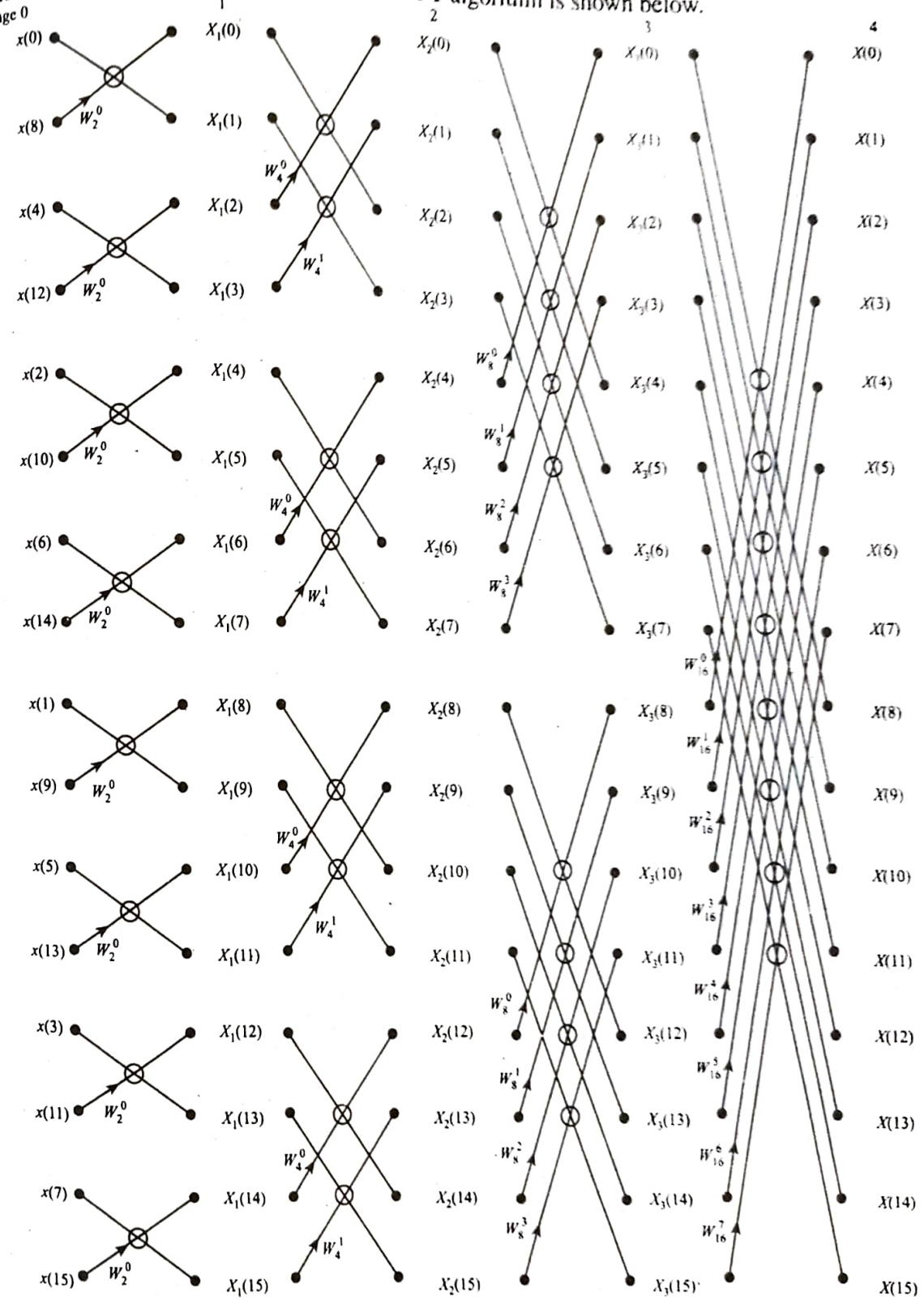


Fig. Ex.3.54 Flow graph for computing 16-point DFT using DIT-FFT algorithm.

3.12.4 Decimation-in-frequency FFT

In this algorithm, the output sequence $X(k)$ is divided into smaller and smaller subsequences in the same manner as in the decimation-in-time algorithm. Here too, we consider N to be a power of 2, so that $N = 2^p$. The goal is to separately evaluate the odd-numbered and the even-numbered frequency samples. We start with the DFT formula:

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

Let us breakup $X(k)$ as follows:

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

Letting $r = n - \frac{N}{2}$ in the second summation, we get

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{r=0}^{\frac{N}{2}-1} x\left(r + \frac{N}{2}\right) W_N^{k(r+N/2)} \tag{3.25}$$

Since r is a dummy variable, it can be replaced by n. Accordingly, the above equation becomes

$$\begin{aligned} X(k) &= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n + \frac{N}{2}\right) W_N^{k(n+N/2)} \qquad (3.26) \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + W_N^{kN/2} \sum_{n=0}^{\frac{N}{2}-1} x\left(n + \frac{N}{2}\right) W_N^{kn} \end{aligned}$$

Notice that $W_N^{kN/2} = (-1)^k$. Hence,

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \tag{3.27}$$

The decimation-in-frequency is now done by taking the odd and even terms of $X(k)$. For even values of k, say $k = 2r$, and $r = 0, 1, \cdots \frac{N}{2} - 1$, equation (3.27) becomes

$$\begin{aligned} X(2r) &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^{2r} x\left(n + \frac{N}{2}\right) \right] W_N^{2rn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N/2}^{rn} \qquad (3.28) \end{aligned}$$

For odd values of k, say $k = 2r + 1$ and $r = 0, 1, \cdots \frac{N}{2} - 1$, equation (3.27) becomes

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

$$= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \, W_N^{2rn} \tag{3.29}$$

Define $$e(n) = x(n) + x\left(n + \frac{N}{2}\right) \tag{3.30}$$

and $$o(n) = \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \tag{3.31}$$

Then, equations (3.28) and (3.29) become

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1} e(n) \, W_{N/2}^{rn} \tag{3.32}$$

$$X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} o(n) \, W_{N/2}^{rn}, \quad r = 0, 1, \cdots \frac{N}{2} - 1 \tag{3.33}$$

From equations (3.30) and (3.31), we see that once $e(n)$ and $o(n)$ are calculated, the even and odd indexed $X(k)$s are found from the $\frac{N}{2}$-point transforms of $e(n)$ and $o(n)$ as shown in Fig. 3.18 for $N = 8$. The basic computational block is a butterfly as shown in Fig. 3.17.

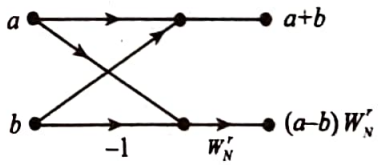


Fig. 3.17 Basic butterfly for decimation-in-frequency FFT.

In a similar manner, each $\frac{N}{2}$-point transform can be decomposed into two $\frac{N}{4}$-point transforms by again decimating the frequency outputs. If this process is continued down to 2-point DFTs, it is easily seen that $\frac{N}{2} \log_2 N$ complex multiplications and $N \log_2 N$ additions are required. Thus, the computational complexity is same as that for the decimation- in-time algorithm. In-place computations can also be done in this method for minimal storage requirements. The complete 8-point decimation-in-frequency FFT flow diagram is shown in Fig. 3.19.

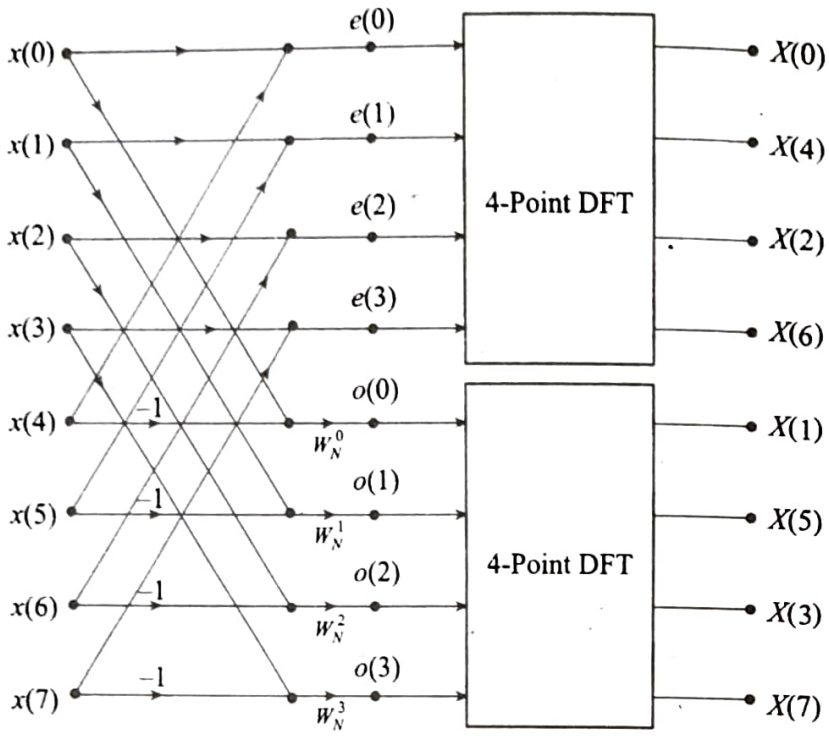


Fig. 3.18 Reduced flow graph for an 8-point decimation-in-frequency FFT.

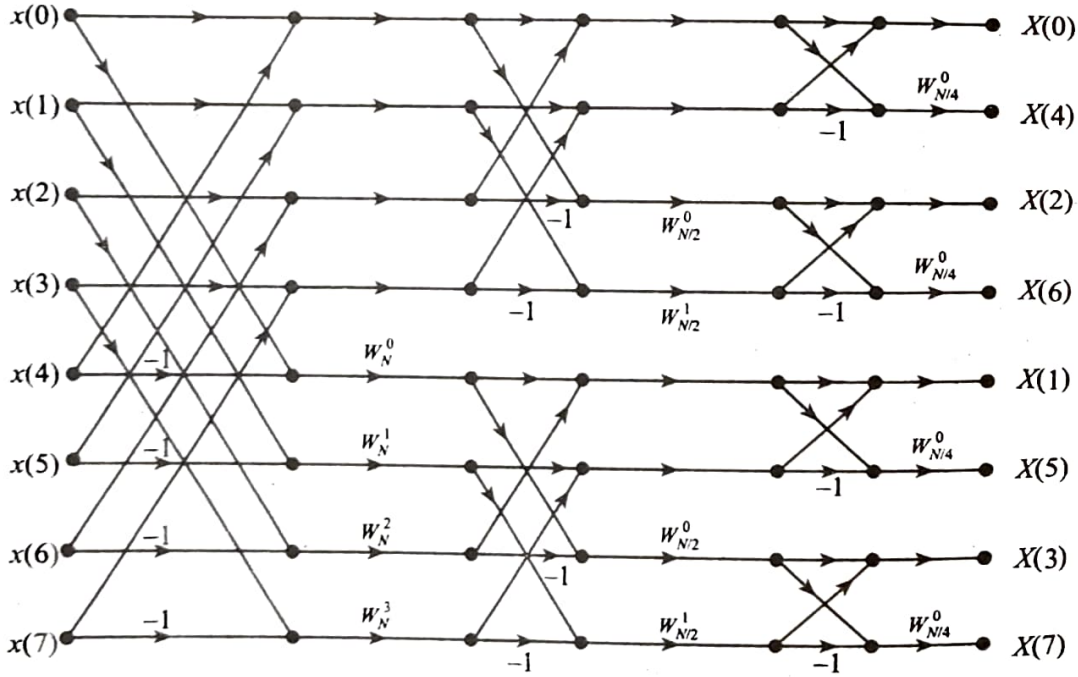


Fig. 3.19 Complete decimation-in-frequency FFT ($N = 8$).

Following observations are made:

i. the input is in the normal order, and

ii. the output is in the bit-reversed order.

We often use, as a matter of convenience, the butterfly shown in Fig. 3.20 as a computational block instead of that shown in Fig. 3.17.

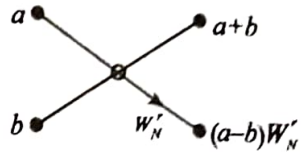


Fig. 3.20 A modified butterfly for DIF-FFT.

Example 3.55 Find the 8-point DFT of a real sequence

$$x(n) = (1, 2, 2, 2, 1, 0, 0, 0)$$

using decimation-in-frequency FFT algorithm.

□ **Solution**

The scaling factors are as follows:

$$W_8^0 = 1, \quad W_8^1 = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}, \quad W_8^2 = -j, \quad W_8^3 = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$$

The basic computational block or the butterfly diagram used is as shown in Fig. Ex.3.55(a).

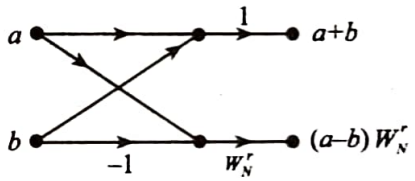


Fig. Ex.3.55(a) Preferred butterfly.

The flow diagram for $N = 8$ is shown in Fig. Ex.3.55(b).

To compute the output of the first stage:

$$\begin{aligned}
X_1(0) &= 1 + 1 = 2 \\
X_1(4) &= 2 + 0 = 2 \\
X_1(2) &= 2 + 0 = 2 \\
X_1(6) &= 2 + 0 = 2 \\
X_1(1) &= (1 - 1)\, W_8^0 = 0 \\
X_1(5) &= (2 - 0)\, W_8^1 = 2\left[\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}\right] = \sqrt{2} - j\sqrt{2} \\
X_1(3) &= (2 - 0)\, W_8^2 = 2(-j) = -j2 \\
X_1(7) &= (2 - 0)\, W_8^3 = 2\left[-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}\right] = -\sqrt{2} - j\sqrt{2}
\end{aligned}$$

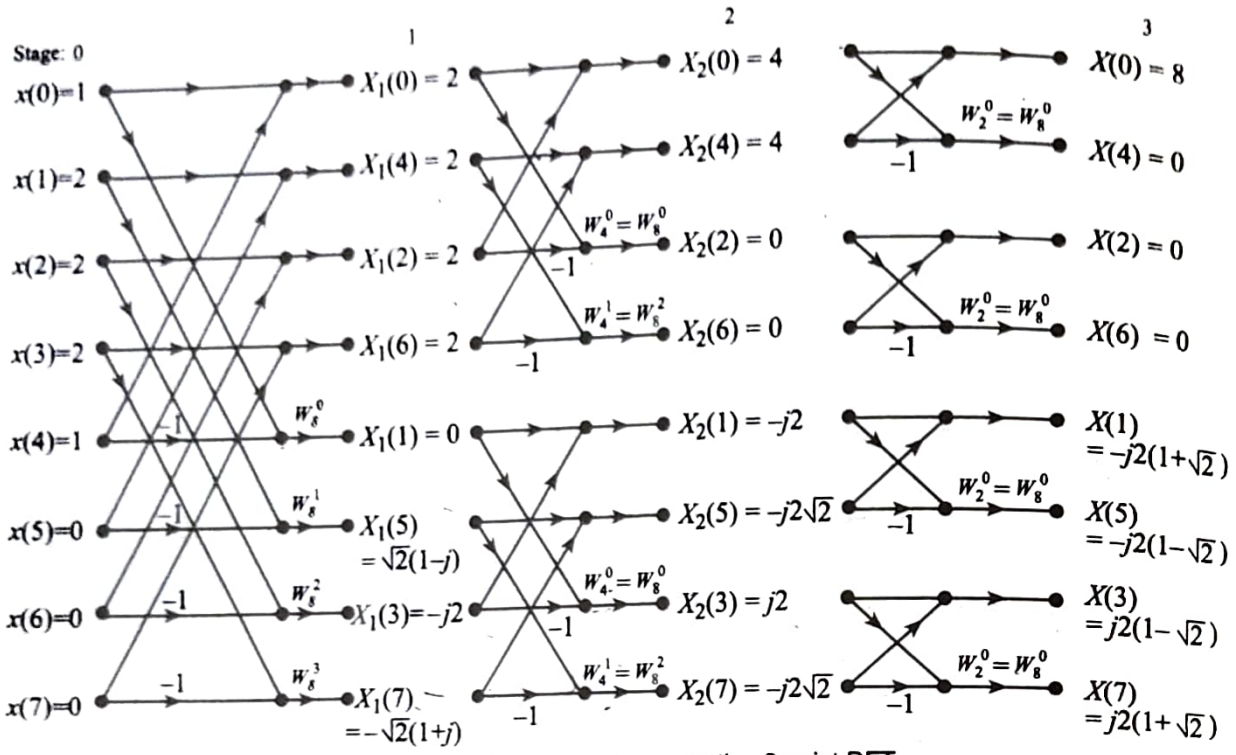


Fig. Ex.3.55(b) Flow graph for computing 8-point DFT.

To compute the output of the second stage:

$$
\begin{aligned}
X_2(0) &= 2+2=4\\
X_2(4) &= 2+2=4\\
X_2(2) &= (2-2)W_8^0=0\\
X_2(6) &= (2-2)W_8^0=0\\
X_2(1) &= (0-j2)=-j2\\
X_2(5) &= \sqrt{2}(1-j)-\sqrt{2}(1+j)=-j2\sqrt{2}\\
X_2(3) &= (0+j2)W_8^0=j2\\
X_2(7) &= \left[\sqrt{2}(1-j)+\sqrt{2}(1+j)\right]W_8^2=-j2\sqrt{2}
\end{aligned}
$$

To compute the output of third stage:

$$
\begin{aligned}
X_3(0) = X(0) &= 4+4=8\\
X_3(4) = X(4) &= (4-4)W_8^0=0\\
X_3(2) = X(2) &= 0+0=0\\
X_3(6) = X(6) &= (0-0)W_8^0=0\\
X_3(1) = X(1) &= -j2-j2\sqrt{2}=-j2\left(1+\sqrt{2}\right)\\
X_3(5) = X(5) &= \left(-j2+j2\sqrt{2}\right)W_8^0=-j2\left(1-\sqrt{2}\right)\\
X_3(3) = X(3) &= j2-j2\sqrt{2}=j2\left(1-\sqrt{2}\right)\\
X_3(7) = X(7) &= \left(j2+j\sqrt{2}\right)W_8^0=j2\left(1+\sqrt{2}\right)
\end{aligned}
$$

Tabulating the result in the normal order, we get

k	$X(k)$	k	$X(k)$
0	8	4	0
1	$-j2(1+\sqrt{2})$	5	$-j2(1-\sqrt{2})$
2	0	6	0
3	$j2(1-\sqrt{2})$	7	$j2(1+\sqrt{2})$

Since $x(n)$ is a real sequence, the symmetry condition, $X(k) = X^*(8-k)$ is observed.

Example 3.56 Find the convolution of $x(n)$ and $h(n)$ shown in Fig. Ex.3.56 below, using

a.. the time-domain convolution operation,

b. the DFT and zero-padding, and

c. the radix-2 FFT and zero-padding.

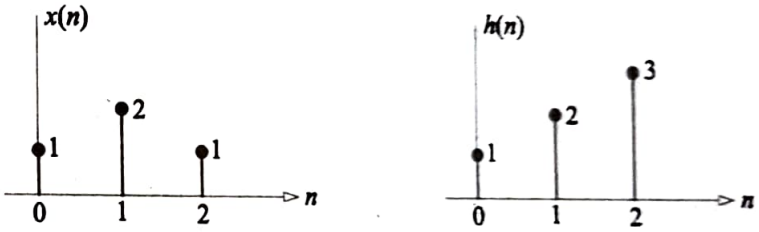


Fig. Ex.3.56 Sequences $x(n)$ and $h(n)$ for Example.3.56.

□ Solution

From Fig. Ex.3.56, we can write

$$x(n) = \underset{\uparrow}{(1, 2, 1)} \quad \text{and} \quad h(n) = \underset{\uparrow}{(1, 2, 3)}$$
$$n=0 \qquad\qquad n=0$$

a.
$$x(n) = \delta(n) + 2\delta(n-1) + \delta(n-2)$$

and
$$h(n) = \delta(n) + 2\delta(n-1) + 3\delta(n-2)$$

Let
$$y(n) = x(n) * h(n)$$

$\Rightarrow$
$$y(n) = [\delta(n) + 2\delta(n-1) + \delta(n-2)] * [\delta(n) + 2\delta(n-1) + 3\delta(n-2)]$$

$\Rightarrow$
$$y(n) = \delta(n) + 4\delta(n-1) + 8\delta(n-2) + 8\delta(n-3) + 3\delta(n-4)$$

$\Rightarrow$
$$y(n) = (1, 4, 8, 8, 3)$$

b. The DFT requires both the sequences to be zero-padded to the convolution length:

$$N = N_1 + N_2 - 1 = 3 + 3 - 1 = 5$$

So, $x_1(n) = (1, 2, 1, 0, 0)$

and $h_1(n) = (1, 2, 3, 0, 0)$

$$\text{DFT}\{x_1(n)\} = X_1(k) = \sum_{n=0}^{4}[1 + 2\delta(n-1) + \delta(n-2)]W_5^{kn}$$

$$= W_5^{kn}\big|_{n=0} + 2W_5^{kn}\big|_{n=1} + W_5^{kn}\big|_{n=2}$$

$$= 1 + 2W_5^k + W_5^{2k}$$

$$\text{DFT}\{h_1(n)\} = H_1(k) = \sum_{n=0}^{4}[\delta(n) + 2\delta(n-1) + 3\delta(n-2)]W_5^{kn}$$

$$= W_5^{kn}\big|_{n=0} + 2W_5^{kn}\big|_{n=1} + 3W_5^{kn}\big|_{n=2}$$

$$= 1 + 2W_5^k + 3W_5^{2k}$$

Let, $y(n) = x_1(n) \circledast_5 h_1(n)$

$\Rightarrow$ $Y(k) = X_1(k)\, H_1(k)$

$\Rightarrow$ $Y(k) = \left[1 + 2W_5^k + W_5^{2k}\right]\left[1 + 2W_5^k + 3W_5^{2k}\right]$

$$= 1 + 2W_5^k + 3W_5^{2k} + 2W_5^k + 4W_5^{2k} + 6W_5^{3k} + W_5^{2k} + 2W_5^{3k} + 3W_5^{4k}$$

$\Rightarrow$ $Y(k) = 1 + 4W_5^k + 8W_5^{2k} + 8W_5^{3k} + 3W_5^{4k}$

Taking IDFT yields $y(n) = (1, 4, 8, 8, 3)$, as before.

c. The radix-2 FFT requires both the sequences to be zero-padded to $N = 8$.

So, $x_2(n) = (1, 2, 1, 0, 0, 0, 0, 0)$

$h_2(n) = (1, 2, 3, 0, 0, 0, 0, 0)$

Let us employ DIF-FFT algorithm to compute $X_2(k)$ and $H_2(k)$. The basic computational block or the **butterfly** is as shown in Fig. Ex.3.56(a).

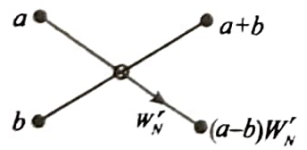


Fig. Ex.3.56(a) Preferred butterfly.

It may be recalled:

(i) $W_8^0 = 1,\ W_8^1 = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}},\ W_8^2 = -j,\ W_8^3 = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

(ii) $W_4^0 = W_2^0 = W_8^0 = 1, \quad W_4^1 = W_8^2 = -j$

To find $X_2(k)$:

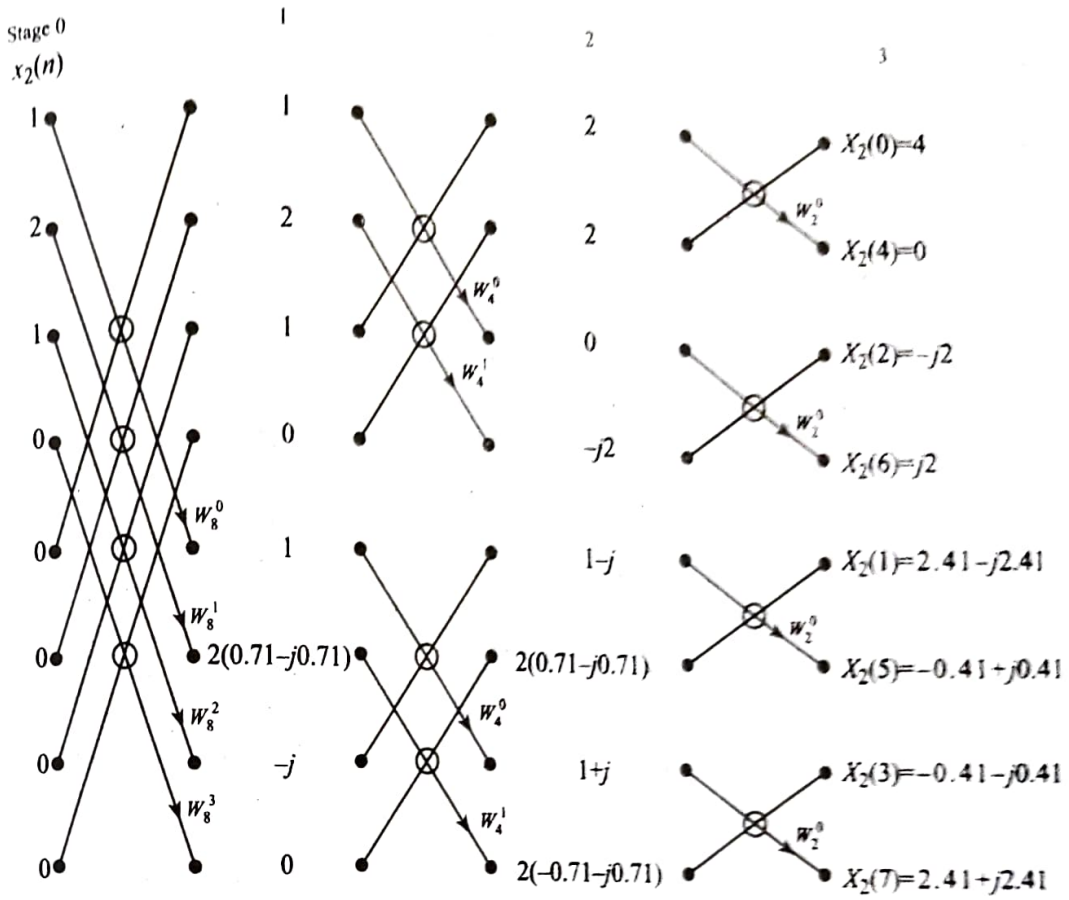


Fig. Ex.3.56(b) Flow diagram for computing 8-point DFT.

Tabulating the results in the normal order, we get

k	$X_2(k)$	k	$X_2(k)$
0	4	4	0
1	$2.41 - j2.41$	5	$-0.41 + j0.41$
2	$-j2$	6	$j2$
3	$-0.41 - j0.41$	7	$2.41 + j2.41$

Since $x_2(n)$ is a real sequence, we find that the symmetry property: $X(k) = X^*(N - k)$ is observed. Furthermore, since $x_2(n)$ is neither odd nor even, we find that $X_2(k)$ is neither purely imaginary nor purely real. It may be noted that the suffix of $X_2(k)$ in the output of stage 3 has nothing to do with the stage.

To find $H_2(k)$

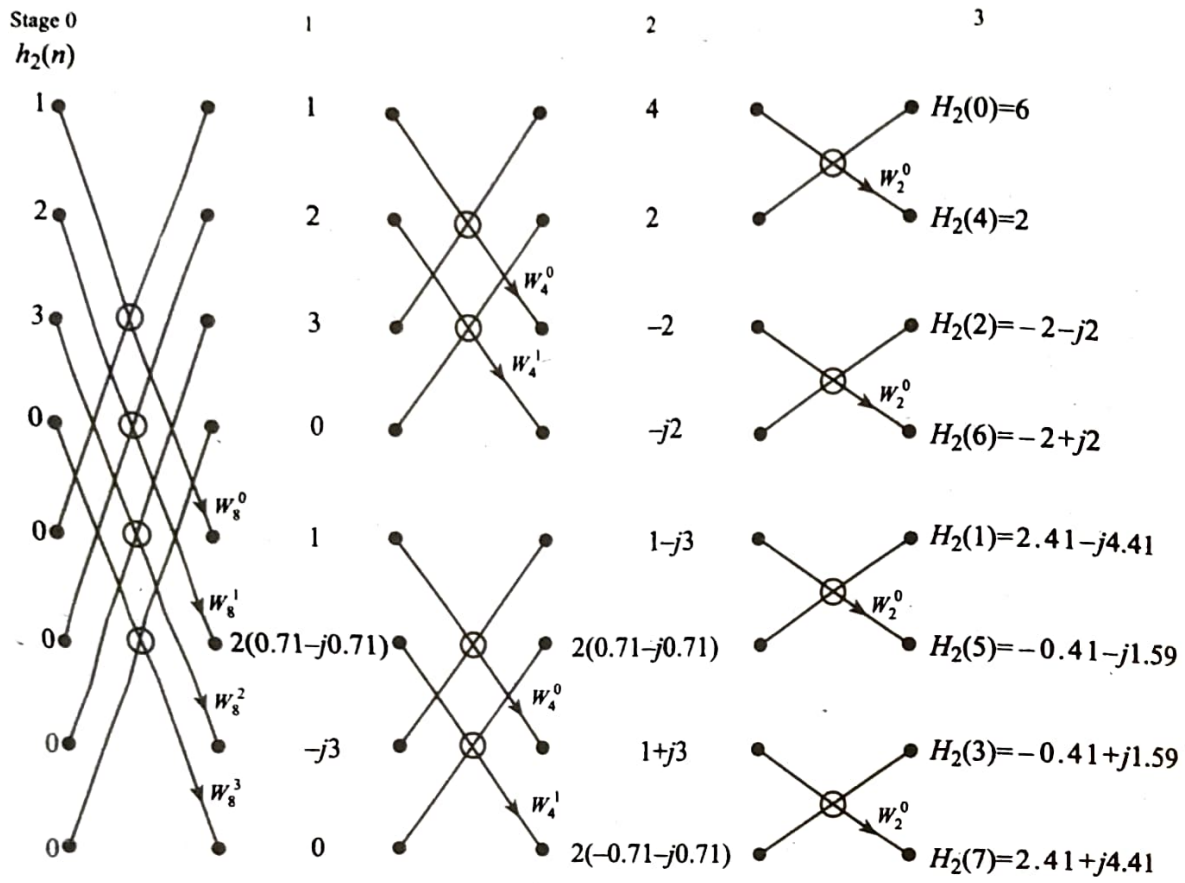


Fig. Ex.3.56(c) Flow diagram for computing 8-point DFT.

Tabulating the results in the normal order, we get

k	$H_2(k)$	k	$H_2(k)$
0	6	4	2
1	$2.41 - j4.41$	5	$-0.41 - j1.59$
2	$-2 - j2$	6	$-2 + j2$
3	$-0.41 + j1.59$	7	$2.41 + j4.41$

Since $h_2(n)$ is a real sequence, the symmetry property: $H_2(k) = H_2^*(8 - k)$ is observed. In addition, since $h_2(n)$ is neither odd nor even, we find that $H_2(k)$ is neither purely imaginary nor purely real. It may be noted that the suffix of $H_2(k)$ in the output of stage 3 has nothing to do with the stage.

To find $Y(k) = X_2(k)\ H_2(k)$:

k	$X_2(k)$	$H_2(k)$	$Y(k) = X_2(k)\ H_2(k)$
0	4	6	24
1	$2.41 - j2.41$	$2.41 - j4.41$	$-4.82 - j16.49$
2	$-j2$	$-2 - j2$	$-4 + j4$
3	$-0.41 - j0.41$	$-0.41 + j1.59$	$0.82 - j0.49$
4	0	2	0
5	$-0.41 + j0.41$	$-0.41 - j1.59$	$0.82 + j0.49$
6	$j2$	$-2 + j2$	$-4 - j4$
7	$2.41 + j2.41$	$2.41 + j4.41$	$-4.82 + j16.49$

To find $y(n)$:

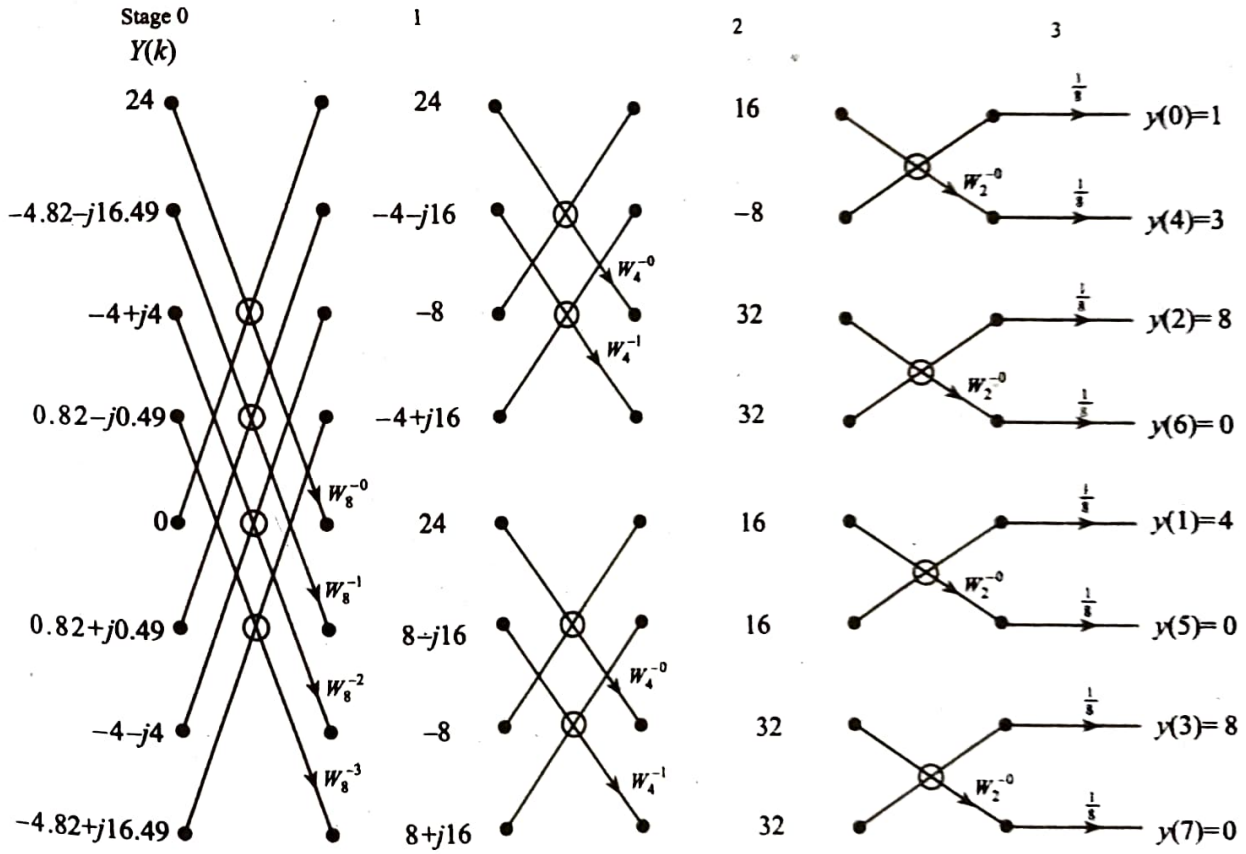


Fig. Ex.3.56(d) Flow diagram for computing 8-point IDFT.

Let $$y(n) = x_2(n) \circledast h_2(n)$$

$$\Rightarrow \quad y(n) = \text{IDFT}\{X_2(k)\ H_2(k)\}$$

Let us employ DIF-FFT algorithm to find $y(n)$. The scaling factors are as follows:

$$W_8^{-0} = 1$$
$$W_8^{-1} = (W_8^1)^* = \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$$
$$W_8^{-2} = (W_8^2)^* = j$$
$$W_8^{-3} = (W_8^3)^* = -\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$$
$$W_4^{-1} = W_8^{-2}$$

Also,
Refer Fig. Ex.3.56(d), we get

Output of stage 1

$$y_1(0) = 24 + 0 = 24$$
$$y_1(4) = (-4.82 - j16.49) + 0.82 + j0.49 = -4 - j16$$
$$y_1(2) = (-4 + j4) + (-4 - j4) = -8$$
$$y_1(6) = (0.82 - j0.49) + (-4.82 + j16.49) = -4 + j16$$
$$y_1(1) = (24 - 0)W_8^{-0} = 24$$
$$y_1(5) = [(-4.82 - j16.49) - (0.82 + j0.49)]\, W_8^{-1} = 8 - j16$$
$$y_1(3) = [(-4 + j4) - (-4 - j4)]\, W_8^{-2} = -8$$
$$y_1(7) = [(0.82 - j0.49) - (-4.82 + j16.49)]\, W_8^{-3} = 8 + j16$$

Output of stage 2 The final result only is given. The calculations are left as an exercise to the reader.

$$y_2(0) = 16 \qquad y_2(1) = 16$$
$$y_2(4) = -8 \qquad y_2(5) = 16$$
$$y_2(2) = 32 \qquad y_2(3) = 32$$
$$y_2(6) = 32 \qquad y_2(7) = 32$$

Output of stage 3

$$y(0) = y_3(0) = (16 - 8)\frac{1}{8} = 1$$
$$y(4) = y_3(4) = (16 + 8)W_2^{-0} \times \frac{1}{8} = 3$$
$$y(2) = y_3(2) = (32 + 32)\frac{1}{8} = 8$$
$$y(6) = y_3(6) = (32 - 32)W_2^{-0} \times \frac{1}{8} = 0$$
$$y(1) = y_3(1) = (16 + 16)\frac{1}{8} = 4$$
$$y(5) = y_3(5) = (16 - 16)W_2^{-0} \times \frac{1}{8} = 0$$
$$y(3) = y_3(3) = (32 + 32)\frac{1}{8} = 8$$
$$y(7) = y_3(7) = (32 - 32)W_2^{-0} \times \frac{1}{8} = 0$$

That is,

$$y(n) = (1, 4, 8, 8, 3, 0, 0, 0)$$

Note that there are three trailing zeros.

Example 3.57 Find the 4-point circular convolution of $x(n)$ and $h(n)$ given in Fig. Ex.3.57 using radix-2 DIF-FFT algorithm.

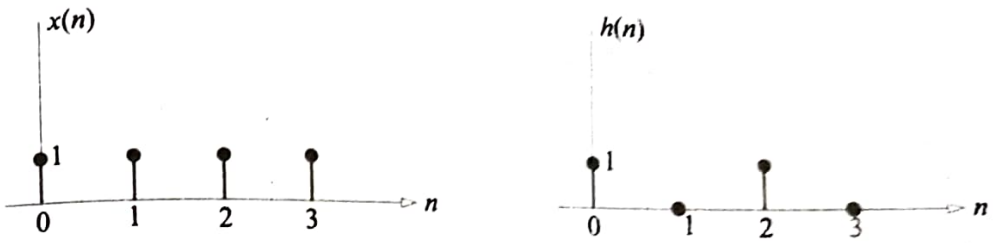


Fig. Ex.3.57 Sequences $x(n)$ and $h(n)$ for Example.3.57.

Solution

Let

$$\begin{aligned} y(n) &= x(n) \circledast_4 h(n) \\ &= \text{IFFT}\{X(k)H(k)\} \end{aligned}$$

To find $X(k)$:

Let the computational block or the butterfly used be of the form as shown in Fig. Ex.3.57(a).

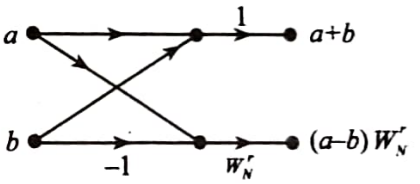


Fig. Ex.3.57(a) Preferred butterfly.

The flow diagram for $N = 4$, DIF-FFT algorithm is shown in Fig. Ex.3.57(b).

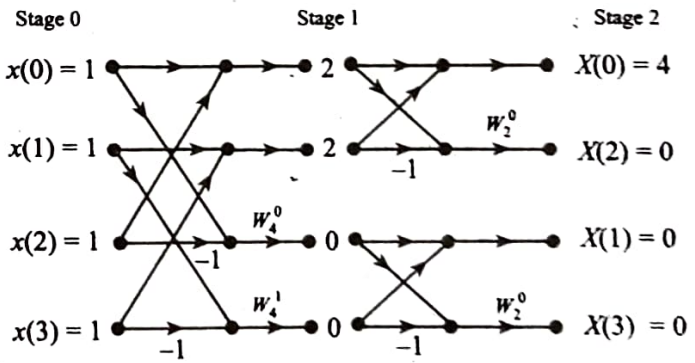


Fig. Ex.3.57(b) Flow diagram for computing 4-point DFT.

Recall: $W_4^0 = 1$, $W_4^1 = -j$ and $W_2^0 = 1$.

The result is tabulated in the normal order as given below:

k	$X(k)$	k	$X(k)$
0	4	2	0
1	0	3	0

or equivalently $X(k) = (4, 0, 0, 0)$.

To find $H(k)$

The flow diagram for $N = 4$ to compute $H(k)$, using DIF-FFT algorithm is shown in Fig. Ex.3.57(c).

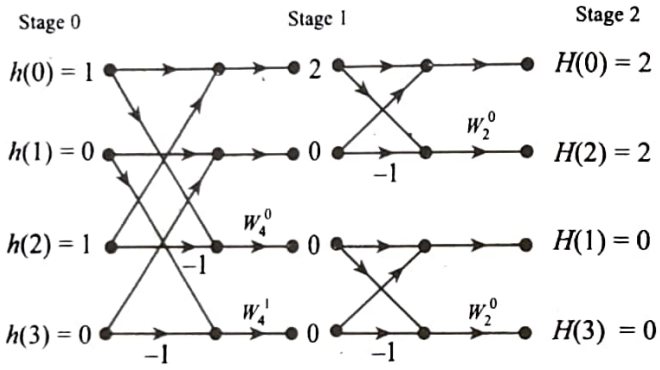


Fig. Ex.3.57(c) Flow diagram for computing 4-point DFT.

The result is tabulated in the normal order as given below.

k	$H(k)$	k	$H(k)$
0	2	2	2
1	0	3	0

or equivalently $H(k) = (2, 0, 2, 0)$.

To find $y(n)$

$$Y(k) = X(k)\, H(k) = (8, 0, 0, 0)$$

IFFT of $Y(k)$ is found using the flow diagram given in Fig. Ex.3.57(d).

Recall: $W_4^{-0} = \left(W_4^0\right)^* = 1$

$W_4^{-1} = \left(W_4^1\right)^* = j$

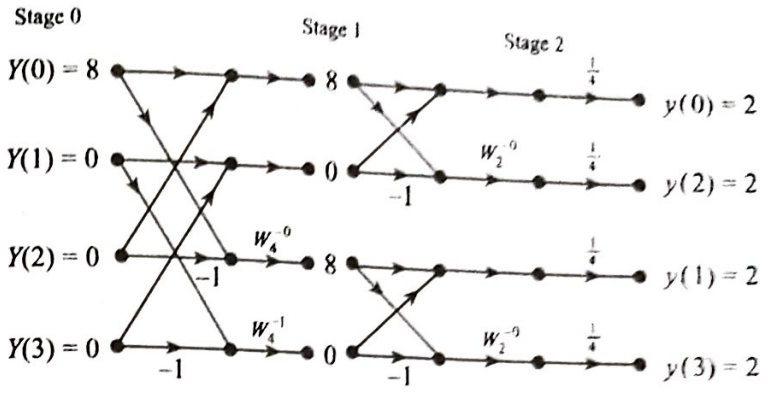


Fig. Ex.3.57(d) Flow diagram for computing 4-point IDFT.

Thus,
$$y(n) = \underset{\uparrow}{(2, 2, 2, 2)}_{n=0}$$

Example 3.58 Find the periodic convolution of $x(n)$ and $h(n)$ shown in Fig. Ex.3.58, using

a. the time-domain convolution operation,

b. the DFT operation. Is this result same as that of part (a)? and

c. the radix-2 FFT and zero-padding. Is this result same as that of part (a)? should it be?

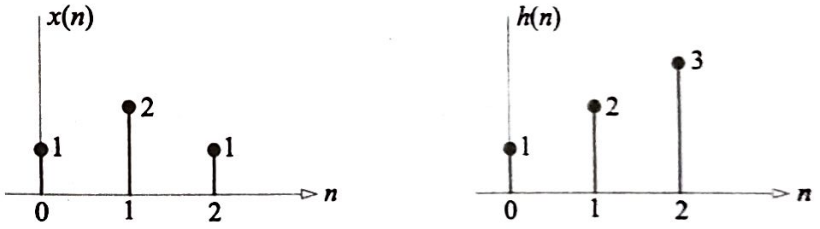


Fig. Ex.3.58 Sequences $x(n)$ and $h(n)$ for Example.3.58.

Solution

a. Let
$$y_l(n) = x(n) * h(n)$$
$$\Rightarrow \quad y_l(n) = [\delta(n) + 2\delta(n-1) + \delta(n-2)] * [\delta(n) + 2\delta(n-1) + 3\delta(n-2)]$$

Recall the property:

$$\delta(n-\alpha) * \delta(n-\beta) = \delta[n-(\alpha+\beta)]$$

Hence,
$$y_l(n) = \delta(n) + 4\delta(n-1) + 8\delta(n-2) + 8\delta(n-3) + 3\delta(n-4)$$
$$\Rightarrow \quad y_l(n) = (1, 4, 8, 8, 3)$$

Since, the circular convolution of $x(n)$ and $h(n)$, namely $y_c(n)$ should have 3 samples, the aliasing takes place in $y_l(n)$. That is, 5 samples of $y_l(n)$ get reduced to 3 samples in $y_c(n)$. By wraparound, we have

$$\begin{aligned} y_c(0) &= y_l(0) + y_l(3) = 1 + 8 = 9 \\ y_c(1) &= y_l(1) + y_l(4) = 4 + 3 = 7 \\ y_c(2) &= y_1(2) = 8 \end{aligned}$$

Thus, $y_c(n) = x(n) \circledast_3 y(n) = (9, 7, 8)$

b. Let $y_c(n) = x(n) \circledast_3 h(n)$

$$\Rightarrow \quad \begin{aligned} Y_c(k) &= X(k)\, H(k), \qquad 0 \leqslant k \leqslant 2 \\ &= \left[1 + 2W_3^k + W_3^{2k}\right]\left[1 + 2W_3^k + 3W_3^{2k}\right] \\ &= 1 + 2W_3^k + 3W_3^{2k} + 2W_3^k + 4W_3^{2k} + 6W_3^{3k} \\ &\quad + W_3^{2k} + 2W_3^{3k} + 3W_3^{4k} \end{aligned}$$

Recall the property: $W_3^{3k} = W_3^{0k}$ and $W_3^{4k} = W_3^k$

Then, $Y_c(k) = 9 + 7W_3^k + 8W_3^{2k}$

Taking 3-point IDFT, we get

$$y_c(n) = (9, 7, 8)$$

c. To compute $X(k)$ and $H(k)$ using DIF-FFT algorithm, let

$$x(n) = (1, 2, 1, 0)$$

and $$h(n) = (1, 2, 3, 0)$$

FACTS:

- Since $h(n)$ and $x(n)$ are both real, their DFTs will observe the symmetry condition:

$$H(k) = H^*(N - k)$$

and $$X(k) = X^*(N - k)$$

- Since $x(n)$ and $h(n)$ are neither odd nor even their DFTs will neither be purely imaginary nor purely real.

To compute $X(k)$:

The butterfly diagram used is as shown in Fig. Ex.3.58(a).

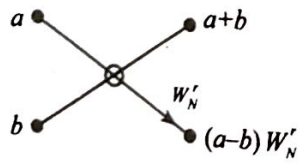


Fig. Ex.3.58(a) Preferred butterfly.

The flow diagram for 4-point DIF-FFT is shown in Fig. Ex.5.58(b).

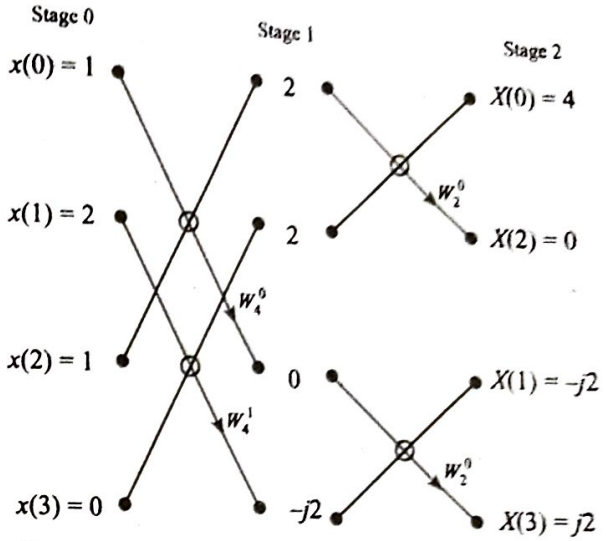


Fig. Ex.3.58(b) Flow diagram for computing 4-point DFT.

Hence, $$X(k) = (4, -2j, 0, 2j)$$

To find $H(k)$:

The 4-point DFT of $h(n)$ is found using the flow diagram as shown in Fig. Ex.3.58(c).

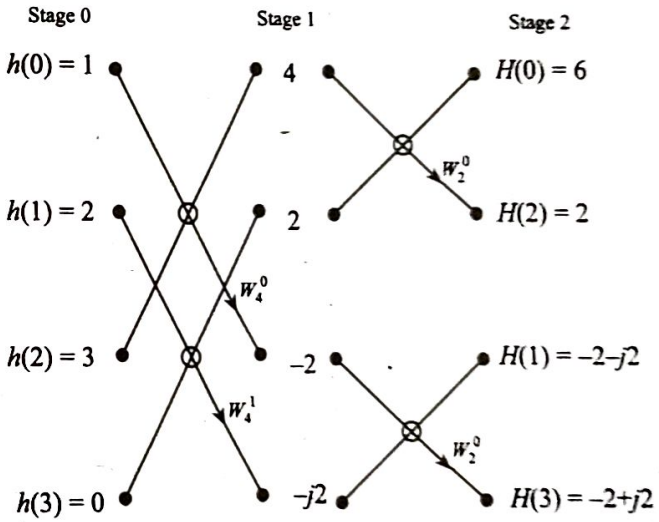


Fig. Ex.3.58(c) Flow diagram for computing 4-point DFT.

Thus, $$H(k) = (6, -2 - 2j, 2, -2 + 2j)$$

To find $y_c(n)$:

$$y_c(n) = x(n) \circledast_4 h(n)$$

In this case, the 4-point circular convolution of $x(n)$ and $h(n)$ is found using the Stockham's method. That is,

$$y_c(n) = \text{IFFT}\{X(k)\ H(k)\} = \text{IFFT}\{Y_c(k)\}$$

where

$$Y_c(k) = X(k)\ H(k)$$
$$= (24, -4 + j4, 0, -4 - j4)$$

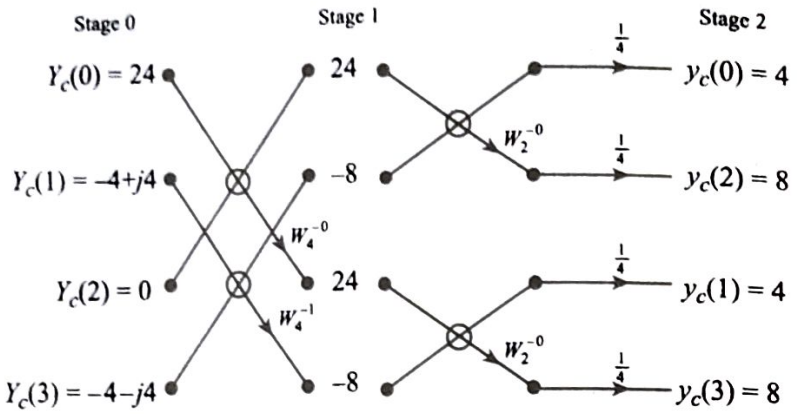


Fig. Ex.3.58(d) Flow diagram for computing 4-point IDFT.

Hence, $$y_c(n) = \underset{\uparrow}{(4, 4, 8, 8)}_{n=0}$$

This result does not match with part (a) and part (b). The reason is that the sequences $x(n)$ and $h(n)$ are assumed to be periodic with a period 4 (not 3).

Example 3.59 Find the 4-point DFT of the sequence, $x(n) = \cos\left(\frac{\pi}{4}n\right)$ using DIF-FFT algorithm.

□ Solution

The scale factors are $W_4^0 = 1$ and $W_4^1 = -j$.

We find that $$x(n) = \underset{\uparrow}{\left(1, \frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}\right)}_{n=0}$$

FACTS:

- Since $x(n)$ is a real sequence, the symmetry condition: $X(k) = X^*(N-k)$ will be observed.
- Also, $X(k)$ will not be purely real or purely imaginary.

Let the butterfly diagram be as shown in Fig. Ex.3.59(a).

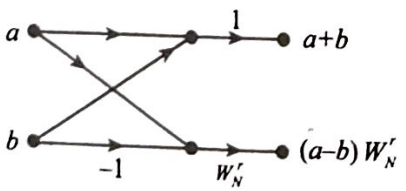


Fig. Ex.3.59(a) Preferred butterfly.

The flow diagram for DIF-FFT algorithm is shown in Fig. Ex.3.59(b).

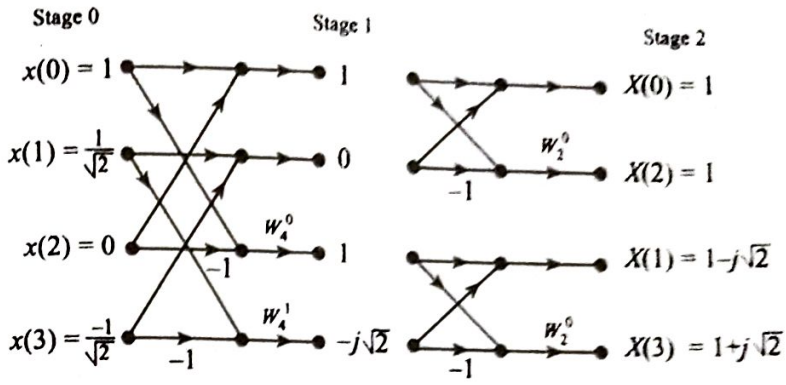


Fig. Ex.3.59(b) Flow diagram for computing 4-point DFT.

Thus, $$X(k) = \left(1, 1 - j\sqrt{2}, 1, 1 + j\sqrt{2}\right)$$

Example 3.60 Find the 4-point real sequence $x(n)$ if its 4-point DFT samples are $X(0) = 6$, $X(1) = -2 + j2$, $X(2) = -2$. Use DIF-FFT algorithm.

Solution

Since $x(n)$ is a real sequence, it has to satisfy the following symmetry condition:

$$X(k) = X^*(4-k)$$

$$\Rightarrow \quad X(3) = X^*(1) = -2 - j2$$

Hence, $$X(k) = (6, -2 + j2, -2, -2 - j2)$$

Since $X(k)$ is neither purely real nor purely imaginary, its inverse DFT, $x(n)$ will neither be even nor odd.

The sample butterfly used in the flow diagram is shown in Fig. Ex.3.60(a).

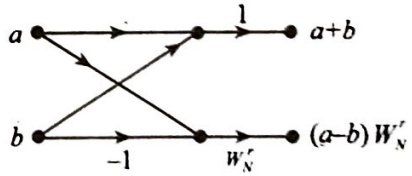


Fig. Ex.3.60(a) Preferred butterfly.

The flow diagram for the computation of 4-point IFFT of $X(k)$ is shown in Fig. Ex.3.60(b).

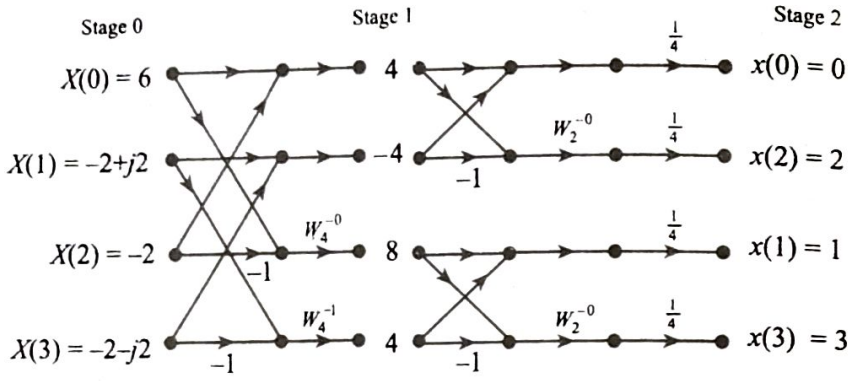


Fig. Ex.3.60(b) Flow diagram for computing 4-point IDFT.

Hence, $x(n) = (0, 1, 2, 3)$
$\uparrow$
$n=0$

3.13 Signal Segmentation (Linear filtering of long data sequences using DFT/FFT)

In this section, we deal with a practical difficulty which often arises, when linear convolution is performed digitally by fast convolution. Our discussion till now assumes that $x(n)$ and $h(n)$ contain roughly the same number of samples, and that they are zero-filled up to the same length of transform. However, in practice we often encounter a situation in which we need to convolve a long input signal with a relatively short impulse response. For example, we may have a signal with several hundreds of samples, but an impulse response with less than fifty samples. In such cases, it is uneconomical in terms of computing time (and storage) to use the same length of transform for both $x(n)$ and $h(n)$. In addition, in real-time applications the use of very long transform lengths for $x(n)$ may give rise to an unacceptable processing delay.

The above mentioned problems may be overcome by segmenting the input signal into sections of manageable length and then performing fast convolution on each section and finally combining the outputs. Two well-known techniques are commonly used: the *overlap-add method* and the *overlap- save method.*

3.13.1 Overlap-add method

Let us consider two sequences, $x(n)$ and $h(n)$ having lengths $K \times N$ and M respectively with K being a very large integer. The sequence $x(n)$ is subdivided into non-overlapping sections each of length N as shown in Fig. 3.20. The objective is to find $y(n) = x(n) * h(n)$.

Procedure for overlap-add method

Step 1: The number of samples M of the system's impulse response $h(n)$ is known, so we decide upon L, the number of points for which the DFTs are to be computed. We choose $L = 2^M$ (a power of 2), so that a radix-2 FFT can be used. The sequence $h(n)$ is zero-padded, so that last $(N - 1)$ values are zeros. The zero-padded impulse response is denoted as $h'(n)$ in the analysis to follow.

Step 2: The input sequence $x(n)$ is sectioned into blocks $x_1(n), x_2(n) \cdots$ of lengths N such that $M + N - 1 = L$, so that the effect of linear convolution will be achieved. Notice that the

last $(M-1)$ values of the sequences $x_1'(n), x_2'(n) \cdots$ are zero values. The DFT $H'(k)$ of the zero-padded sequence $h'(n)$ is computed and the values are stored.

Step 3: The complex product $Y_1'(k) = H'(k)X_1'(k)$ is then computed where $X_1'(k)$ is the DFT of the zero-padded sequence $x_1'(n)$.

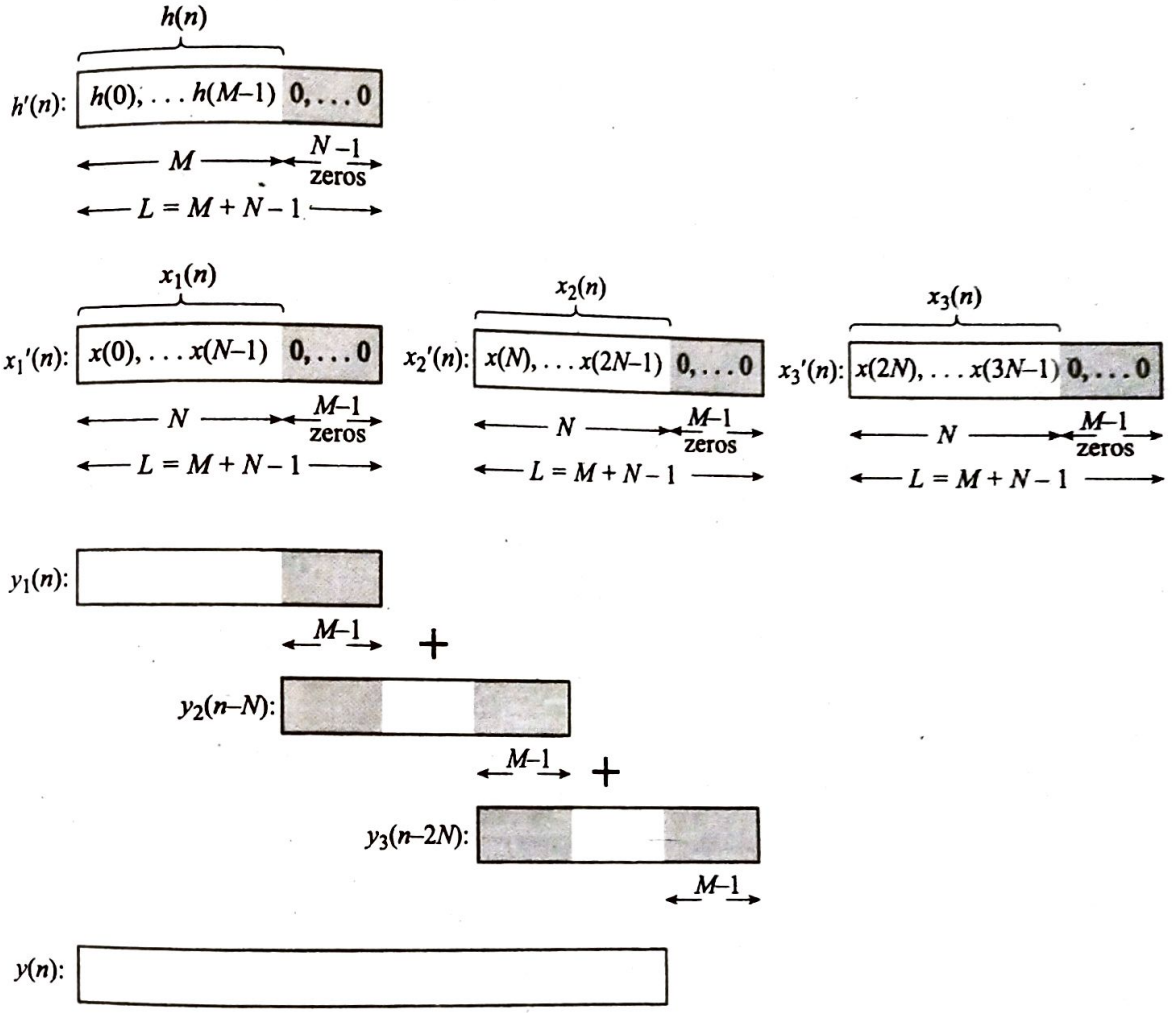


Fig. 3.21 Illustration of overlap-add fast convolution.

Step 4: The IDFT of $Y_1'(k)$ is then computed giving $y_1'(n) = y_1(n)$, for $n = 0, 1, \cdots L-1$.

Step 5: When the input data are available for the next section of L-points, these data are used to compute $M + N - 1 = L$-point DFT of $x_2'(n)$. For doing this, we have to reindex the points in $x_2'(n)$ from $n = 0$ to $M + N - 2 = L - 1$. Steps 3 and 4 are repeated with $X_1'(k)$ replaced by $X_2'(k)$ to obtain $y_2'(n) = y_2(n)$ for $0 \leqslant n \leqslant L-1$.

Step 6: As a next step, the sequence $y_2(n)$ is time shifted to produce $y_2(n-N)$. The last $(M-1)$ points of $y_1(n)$ and the first $(M-1)$ points of $y_2(n-N)$ are overlapped and added as shown in Fig. 3.21.

Step 7: Repeat steps 5 and 6 until the end of the input data is reached. Fig. 3.21 illustrates the various steps involved in the overlap-add technique.

Thus, $$y(n) = y_1(n) + y_2(n-N) + y_3(n-2N) + \cdots$$

Example 3.61 Perform $x(n) * h(n)$, $0 \leqslant n \leqslant 11$ for the sequences $x(n)$ and $h(n)$ given below, using overlap-add fast convolution technique.

$$h(n) = (1,1,1)$$

and

$$x(n) = (1,2,0,-3,4,2,-1,1,-2,3,2,1,-3)$$

□ **Solution**

Since $M = 3$, we choose the transform length for DFT and IDFT computations as $L = 2^M = 2^3 = 8$. Since $L = M + N - 1$, we get $N = 6$.

Step 1: To find $y_1'(n) = y_1(n)$:

$$\text{Here,} \quad x_1'(n) = \left(1,2,0,-3,4,2,\overbrace{\boxed{0,\ 0}}^{(M-1)\text{ zeros}}\right)$$

$$h'(n) = \left(1,1,1,\overbrace{\boxed{0,0,0,0,0}}^{(N-1)\text{ zeros}}\right)$$

$$y_1(n) = x_1'(n) \circledast_8 h'(n)$$

$$\Rightarrow \quad Y_1(k) = X_1'(k)\, H'(k)$$

$$= \left(1 + 2W_8^k - 3W_8^{3k} + 4W_8^{4k} + 2W_8^{5k}\right)\left(1 + W_8^k + W_8^{2k}\right)$$

$$= 1 + 3W_8^k + 3W_8^{2k} - W_8^{3k} + W_8^{4k} + 3W_8^{5k} + 6W_8^{6k} + 2W_8^{7k}$$

Recall the transform pair: $\text{DFT}\{\delta(n - n_0)\} = W_N^{kn_0}$

Thus,

$$y_1(n) = 1 + 3\delta(n-1) + 3\delta(n-2) - \delta(n-3) + \delta(n-4) + 3\delta(n-5) + 6\delta(n-6) + 2\delta(n-7)$$

$$\Rightarrow \quad y_1(n) = (1,3,3,-1,1,3,6,2)$$

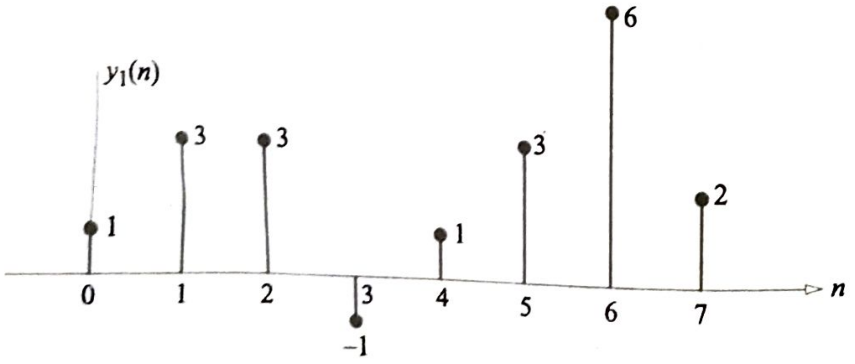


Fig. Ex.3.61(a) Linear convolution of $x_1'(n)$ and $h(n)$ (through circular convolution).

Step 2: To find $y_2'(n) = y_2(n)$:

Here, we re-index the points in $x_2'(n)$ from $n = 0$ to 7.

$$x_2'(n) = \Big(-1, 1, -2, 3, 2, 1, \underbrace{\boxed{0, 0}}_{(M-1)\text{ zeros}}\Big)$$

$$h'(n) = \Big(1, 1, 1, \underbrace{\boxed{0, 0, 0, 0, 0}}_{(N-1)\text{ zeros}}\Big)$$

$$\begin{aligned} Y_2(k) &= \text{DFT}\{y_2(n)\} \\ &= \text{DFT}\{x_2'(n) \circledast_8 h'(n)\} \\ &= X_2'(k)H'(k) \\ &= \left(-1 + W_8^k - 2W_8^{2k} + 3W_8^{3k} + 2W_8^{4k} + W_8^{5k}\right)\left(1 + W_8^k + W_8^{2k}\right) \\ &= -1 - 2W_8^{2k} + 2W_8^{3k} + 3W_8^{4k} + 6W_8^{5k} + 3W_8^{6k} + W_8^{7k} \end{aligned}$$

Recall the transform pair: $\text{DFT}\{\delta(n - n_0)\} = W_N^{kn_0}$

Thus, $y_2(n) = -1 - 2\delta(n-2) + 2\delta(n-3) + 3\delta(n-4) + 6\delta(n-5) + 3\delta(n-6) + \delta(n-7)$

Equivalently, $y_2(n) = (-1, 0, -2, 2, 3, 6, 3, 1)$

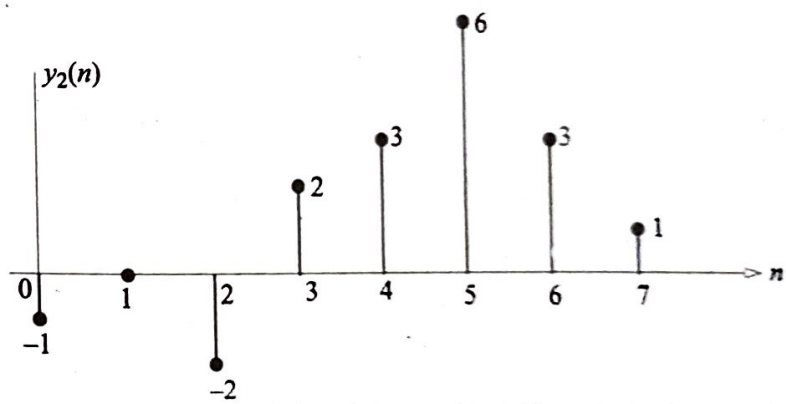


Fig. Ex.3.61(b) Linear convolution of $x_2'(n)$ and $h(n)$ (through circular convolution).

Step 3: To find $y(n) = x(n) * h(n)$:

Formula used: $y(n) = y_1(n) + y_2(n - N) + y_3(n - 2N) + \cdots$

For the present context: $y(n) = y_1(n) + y_2(n - N)$

$= y_1(n) + y_2(n - 6)$

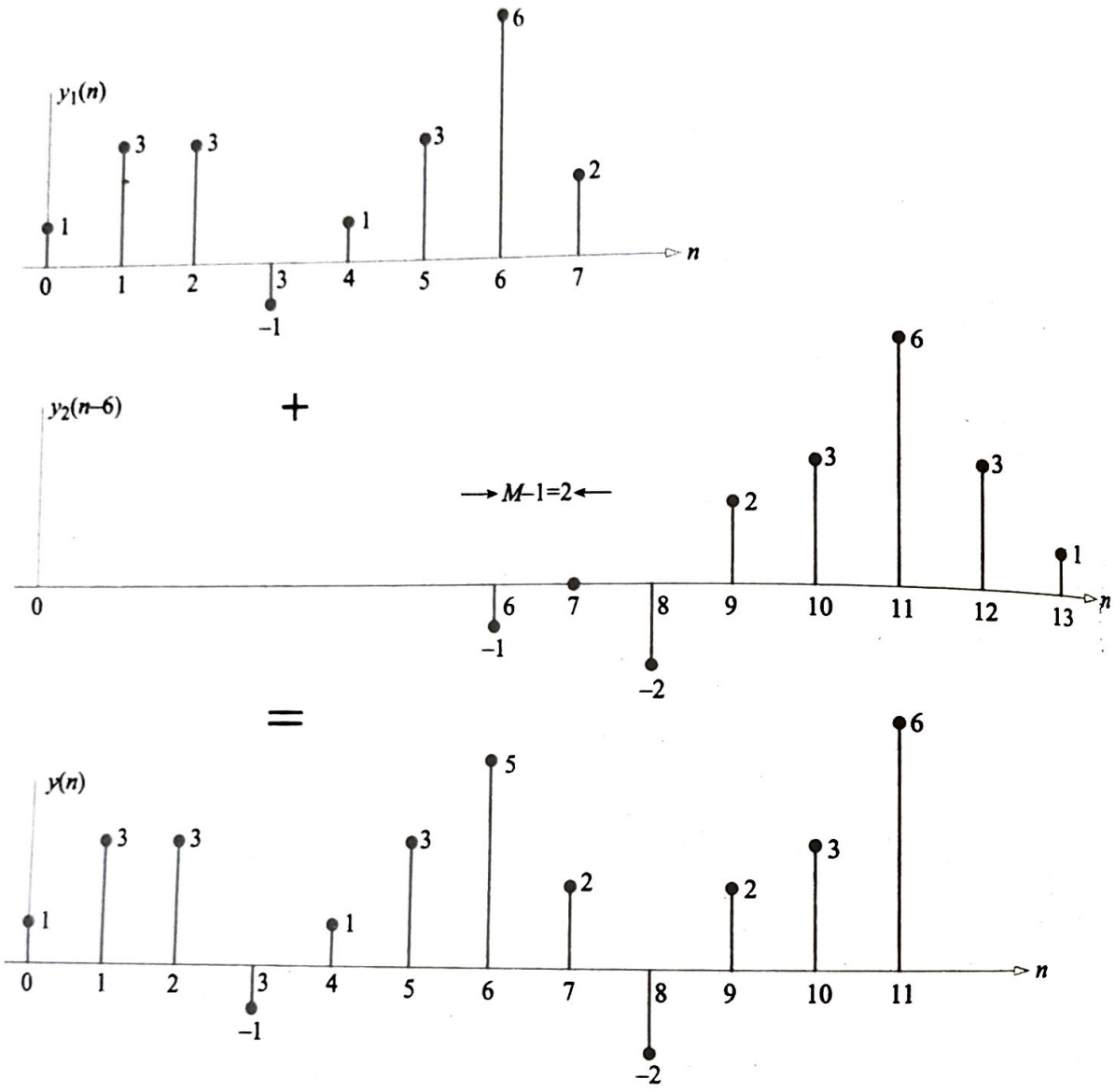


Fig. Ex.3.61(c) The resultant linear convolution.

Thus, $$y(n) = (1, 3, 3, -1, 1, 3, 5, 2, -2, 2, 3, 6)$$
$$\uparrow$$

It may be noted that last two points of $y_2(n-6)$ must be overlapped with $y_3(n-12)$ and added. Since $y_3(n)$ is not calculated, the samples beyond $n = 11$ cannot be computed.

Cross-check:

The veracity of overlap-add convolution technique is now checked by finding the linear convolution of the sequences $x(n)$ and $h(n)$.

Given $$h(n) = (1, 1, 1)$$

and $$x(n) = (1, 2, 0, -3, 4, 2, -1, 1, -2, 3, 2, 1, -3)$$

Writing $h(n)$ and $x(n)$ as the sum of delayed impulses, we get

$$h(n) = \delta(n) + \delta(n-1) + \delta(n-2)$$

and

$$\begin{aligned} x(n) = {} & \delta(n) + 2\delta(n-1) - 3\delta(n-3) + 4\delta(n-4) + 2\delta(n-5) \\ & -\delta(n-6) + \delta(n-7) - 2\delta(n-8) + 3\delta(n-9) \\ & +2\delta(n-10) + \delta(n-11) - 3\delta(n-12) \end{aligned}$$

Recall the property: $\delta(n-\alpha) * \delta(n-\beta) = \delta[n-(\alpha+\beta)]$

Let us perform linear convolution of $x(n)$ and $h(n)$ and then find $y(n)$ for $0 \leqslant n \leqslant 11$.

$$\begin{aligned} y(n) &= h(n) * x(n) \\ &= [\delta(n) + \delta(n-1) + \delta(n-2)] * [\delta(n) + 2\delta(n-1) \\ &\quad -3\delta(n-3) + 4\delta(n-4) + 2\delta(n-5) - \delta(n-6) + \delta(n-7) \\ &\quad -2\delta(n-8) + 3\delta(n-9) + 2\delta(n-10) + \delta(n-11)] \\ &= \delta(n) + 2\delta(n-1) - 3\delta(n-3) + 4\delta(n-4) + 2\delta(n-5) \\ &\quad -\delta(n-6) + \delta(n-7) - 2\delta(n-8) + 3\delta(n-9) + 2\delta(n-10) \\ &\quad +\delta(n-11) + \delta(n-1) + 2\delta(n-2) - 3\delta(n-4) + 4\delta(n-5) + 2\delta(n-6) \\ &\quad -\delta(n-7) + \delta(n-8) - 2\delta(n-9) + 3\delta(n-10) + 2\delta(n-11) \\ &\quad +\delta(n-12) + \delta(n-2) + 2\delta(n-3) - 3\delta(n-5) + 4\delta(n-6) \\ &\quad +2\delta(n-7) - \delta(n-8) + \delta(n-9) - 2\delta(n-10) + 3\delta(n-11) \\ &\quad +2\delta(n-12) + \delta(n-13) \\ &= \delta(n) + 3\delta(n-1) + 3\delta(n-2) - \delta(n-3) + \delta(n-4) \\ &\quad +3\delta(n-5) + 5\delta(n-6) + 2\delta(n-7) - 2\delta(n-8) + 2\delta(n-9) \\ &\quad +3\delta(n-10) + 6\delta(n-11), \qquad 0 \leqslant n \leqslant 11 \end{aligned}$$

Equivalently, we may write

$$y(n) = (1, 3, 3, -1, 1, 3, 5, 2, -2, 2, 3, 6)$$

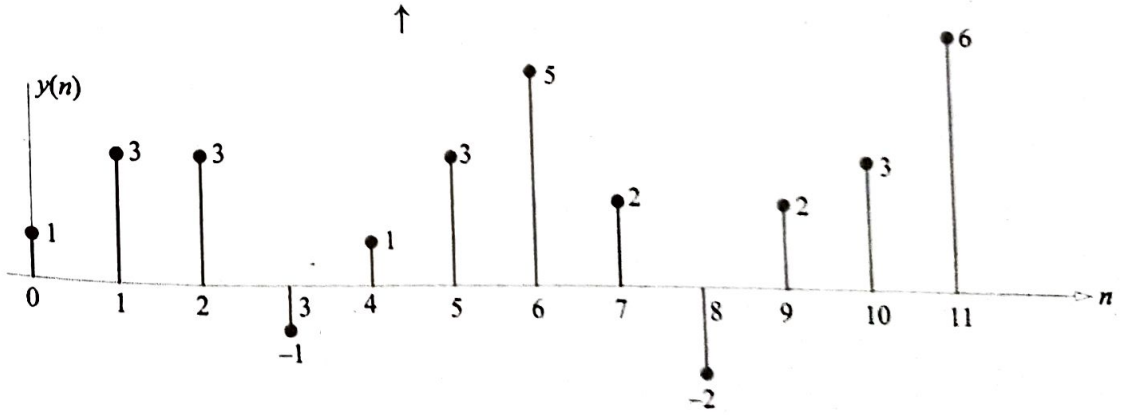


Fig. Ex.3.61(d) Resulant linear convolution.

3.13.2 Overlap-save method

If a length-M sequence is circularly convoluted with a length-N sequence, where $N > M$, the last $N - M + 1$ samples of the result correspond to linear convolution, whereas the first $(M - 1)$ samples do not.

Let $$x(n) = (x(0) \cdots x(N-1), x(N) \cdots x(2N-1), \cdots)$$
$$\longleftarrow x_1(n) \longrightarrow \quad \longleftarrow \quad x_2(n) \quad \longrightarrow$$

and $$h(n) = (h(0), h(1) \cdots h(M-1))$$

Procedure:

Step 1: The number of samples of M of the system's impulse response $h(n)$ is known, so we decide upon L, the number of points for which DFTs are to be computed. We choose $L = 2^M$ (a power of 2), so that a radix-2 FFT can be used. The sequence $h(n)$ is zero-padded, so that the last $(N-1)$ values are zeros. The zero-padded impulse response is denoted as $h'(n)$ in the analysis to follow.

Step 2: The input sequence $x(n)$ is sectioned into blocks $x_1(n), x_2(n) \cdots$ of lengths N such that $M + N - 1 = L$, so that the effect of linear convolution will be achieved. Notice that the first $(M-1)$ values of the sequence $x_1'(n)$ are zero values. However, the first $(M-1)$ values of the sequences $x_2'(n), x_3'(n) \cdots$ begin with last $(M-1)$ values of the previous sequences.

Step 3: The complex product $Y_1'(k) = X_1'(k)\ H'(k)$ is then computed, where $X_1'(k)$ is the L-point DFT of the sequence $x_1'(n)$ and $H'(k)$ is the L-point DFT of the sequence $h'(n)$. Remember to re-index the points in $x_1'(n)$ from $n = 0$ to $L - 1$.

Step 4: The IDFT of $Y_1'(k)$ is then computed giving $y_1'(n) = y_1(n)$ for $n = 0, 1, \cdots L-1$.

Step 5: Steps 3 and 4 are repeated with $X_1'(k)$ replaced by $X_2'(k), X_3'(k) \cdots$. Thus, $y_2(n), y_3(n) \cdots$ are found for $n = 0, 1, \cdots L-1$. Remember to re-index the points in the sequences $x_2'(n), x_3'(n) \cdots$ from $n = 0$ to $M + N - 2 = L - 1$.

Step 6: The first $(M-1)$ samples of $y_1(n), y_2(n), \cdots$ are neglected. Then, $y_1(n), y_2(n) \cdots$ are re-indexed and final convolution output is obtained by gluing $y_1(n), y_2(n) \cdots$ as shown in Fig. 3.22.

That is, $$y(n) = (y_1(n),\ y_2(n), \cdots)$$
$$0 \leqslant n \leqslant N-1 \quad \text{—}\uparrow \qquad \uparrow\text{—} \quad N \leqslant n \leqslant 2N-1$$

The final convolution output is obtained after discarding $(M-1)$ samples at the beginning of each block and the last $(M-1)$ samples of $y_m(n)$ are being saved to replace the $(M-1)$ discarded samples of $y_{m+1}(n)$. This justifies the name overlap and save.

Fig. 3.22 illustrates in detail, the various steps involved in the overlap-save technique.

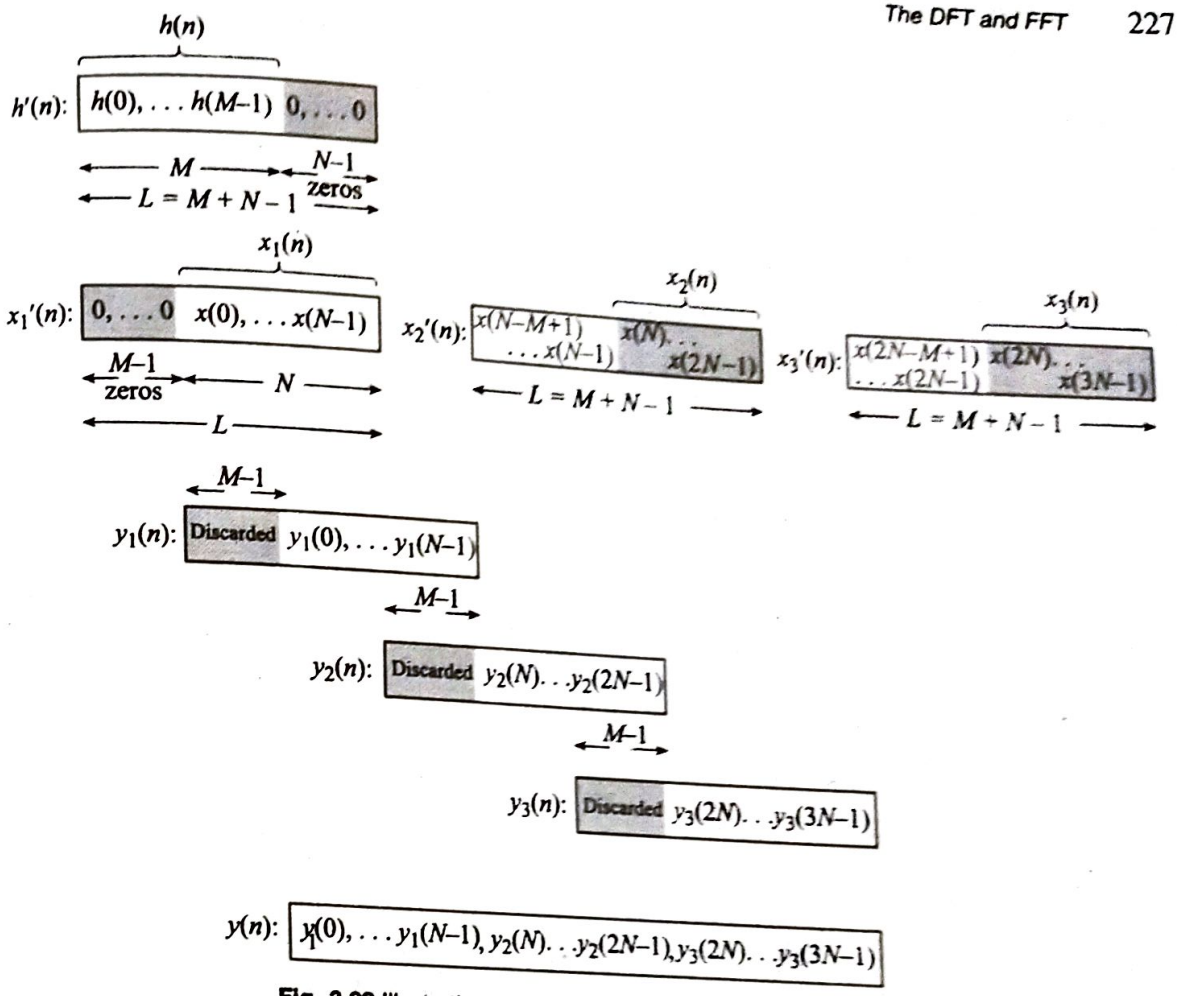


Fig. 3.22 Illustration of overlap-save fast convolution technique.

Example 3.62 Repeat the Example 3.61 using overlap-save technique.

□ Solution

Since $M = 3$, we choose the transform length for DFT/IDFT operation as $L = 2^M = 8$. Since $L = M + N - 1$, we get $N = 6$.

Step 1: To compute $y_1'(n) = y_1(n)$

Here, $$x_1'(n) = \left(\overset{(M-1)\text{ zeros}}{\overset{\downarrow}{\boxed{0, 0,}}} \overset{x_1(n)}{\overset{\downarrow}{\boxed{1, 2, 0, -3, 4, 2}}}\right)$$

It may be noted that we have re-indexed the values of n for $x_1'(n)$ from $n = 0$ to $L - 1$.

$$\text{Also,} \qquad h'(n) = \Big(1, 1, 1, \underbrace{\boxed{0, 0, 0, 0, 0}}_{(N-1)\text{ zeros}}\Big)$$

Let us now find $y_1'(n) = y_1(n)$.

That is,
$$\begin{aligned} y_1(n) &= x_1'(n) \circledast_8 h'(n), \quad 0 \leqslant n \leqslant L-1 = 7 \\ \Rightarrow \quad Y_1(k) &= X_1'(k)\, H'(k) \\ \Rightarrow \quad Y_1(k) &= \left(W_8^{2k} + 2W_8^{3k} - 3W_8^{5k} + 4W_8^{6k} + 2W_8^{7k}\right)\left(1 + W_8^{k} + W_8^{2k}\right) \\ &= W_8^{2k} + 2W_8^{3k} - 3W_8^{5k} + 4W_8^{6k} + 2W_8^{7k} \\ &\quad + W_8^{3k} + 2W_8^{4k} - 3W_8^{6k} + 4W_8^{7k} + 2W_8^{8k} \\ &\quad + W_8^{4k} + 2W_8^{5k} - 3W_8^{7k} + 4W_8^{8k} + 2W_8^{9k} \end{aligned}$$

Recall: $W_8^{8k} = W_8^{0k}$ and $W_8^{9k} = W_8^{k}$

$$\Rightarrow \quad Y_1(k) = 6 + 2W_8^{k} + W_8^{2k} + 3W_8^{3k} + 3W_8^{4k} - W_8^{5k} + W_8^{6k} + 3W_8^{7k}$$

Recall the transform pair: $\text{DFT}\{\delta(n - n_0)\} = W_N^{kn_0}$

Thus, $y_1(n) = 6+2\delta(n-1)+\delta(n-2)+3\delta(n-3)+3\delta(n-4)-\delta(n-5)+\delta(n-6)+3\delta(n-7)$

Equivalently, we may write,

$$y_1(n) = (6, 2, 1, 3, 3, -1, 1, 3)$$

Discarding the first two values and re-indexing $y_1(n)$, we get

$$y_1(n) = \underset{\substack{\uparrow \\ n=0}}{(1, 3, 3, -1, 1, 3)}$$

Step 2: To compute $y_2(n) = y_2'(n)$

$$\text{Here,} \qquad x_2'(n) = \Big(\underbrace{\boxed{4, 2}}_{\substack{\text{Last } (M-1) \\ \text{values of } x_1(n)}}, \underbrace{\boxed{-1, 1, -2, 3, 2, 1}}_{x_2(n)}\Big)$$

It may again be noted that we have re-indexed the values of n for $x_2'(n)$ from $n = 0$ to 7.

$$\text{Also, we have} \qquad h'(n) = \Big(1, 1, 1, \underbrace{\boxed{0, 0, 0, 0, 0}}_{(N-1)\text{ zeros}}\Big).$$

Hence,

$$\begin{aligned} & y_2(n) = x_2'(n) \circledast_8 h'(n), \quad 0 \leqslant n \leqslant L-1=7 \\ \Rightarrow \quad & Y_2(k) = X_2'(k)\, H'(k) \\ & = \left(4+2W_8^k - W_8^{2k} + W_8^{3k} - 2W_8^{4k} + 3W_8^{5k} \right. \\ & \quad \left. +2W_8^{6k} + W_8^{7k}\right)\left(1+W_8^k + W_8^{2k}\right) \\ & = 4+2W_8^k - W_8^{2k} + W_8^{3k} - 2W_8^{4k} + 3W_8^{5k} + 2W_8^{6k} \\ & \quad W_8^{7k} + 4W_8^k + 2W_8^{2k} - W_8^{3k} + W_8^{4k} - 2W_8^{5k} \\ & \quad +3W_8^{6k} + 2W_8^{7k} + W_8^{8k} + 4W_8^{2k} + 2W_8^{3k} - W_8^{4k} + W_8^{5k} \\ & \quad -2W_8^{6k} + 3W_8^{7k} + 2W_8^{8k} + W_8^{9k} \\ & = 7+7W_8^k + 5W_8^{2k} + 2W_8^{3k} - 2W_8^{4k} + 2W_8^{5k} \\ & \quad +3W_8^{6k} + 6W_8^{7k} \end{aligned}$$

Taking IDFT, we get

$$\begin{aligned} & y_2(n) = 7+7\delta(n-1)+5\delta(n-2)+2\delta(n-3)-2\delta(n-4) \\ & \quad +2\delta(n-5)+3\delta(n-6)+6\delta(n-7) \\ \Rightarrow \quad & y_2(n) = (7,7,5,2,-2,2,3,6) \end{aligned}$$

Discarding the first two values and re-indexing $y_2(n)$, we get

$$y_2(n) = (5, 2, -2, 2, 3, 6)$$
$$\uparrow$$
$$n=N=6$$

Step 3: Finally, we glue $y_1(n)$ and $y_2(n)$ to obtain

$$\begin{aligned} y(n) &= (y_1(n), y_2(n)) \\ &= (1, 3, 3, -1, 1, 3, 5, 2, -2, 2, 3, 6) \\ &\quad \uparrow \qquad\qquad\qquad\qquad \uparrow \\ &\quad n=0 \qquad\qquad\qquad\qquad n=6 \end{aligned}$$

We wish to inform the reader that in practice, circular convolution is performed digitally by using FFT algorithms.

3.14 Computing the DFT Using Linear Filtering

In the preceeding sections, we have studied direct DFT computation, the decimation-in-time and decimation-in-frequency FFT algorithms. In this section, we will study two linear filtering approaches by which DFT can also be computed. They are Goertzel algorithm and the chirp-$\mathcal{Z}$ transform. The Goertzel algorithm is computationally more efficient than the FFT when a relatively small number of values of the DFT are desired.

We know that DFT is a special case of the $\mathcal{Z}$-transform evaluated on the unit circle. Some applications, demand the $\mathcal{Z}$-transform to be evaluated at points other than the unit circle. The chirp-$\mathcal{Z}$ transform is a well documented linear filtering approach that allows us to compute the $\mathcal{Z}$-transform of a finite length sequence on a different variety of contours in the z-plane.

3.14.1 The Chirp-$\mathcal{Z}$ transform (CZT)

The chirp-$\mathcal{Z}$ transform is used for evaluating $\mathcal{Z}$-transform of a sequence of M points in the z-plane which lie on circular or spiral contours beginning at any arbitrary point in the z-plane. To derive chirp-$\mathcal{Z}$ transform, consider the $\mathcal{Z}$-transform of the sequence $x(n)$, which for discrete set of values of z_k can be written as

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n}, \quad k = 0, 1 \cdots M-1 \tag{3.34}$$

Equation (3.34) reduces to DFT if

$$z_k = e^{j\frac{2\pi k}{N}} = W_N^{-k}, \quad k = 0, 1, \cdots N-1$$

Let z_k be a point on the spiral centered about the origin. This can be accomplished by writing z_k as

$$z_k = A\, B^{-k} \tag{3.35}$$

where $A = A_0\, e^{j\theta_0}$ is the point at which the spiral starts and $B = B_0 e^{-j\phi_0}$ determines the rate of the spiral and the angular separation of the points z_k.

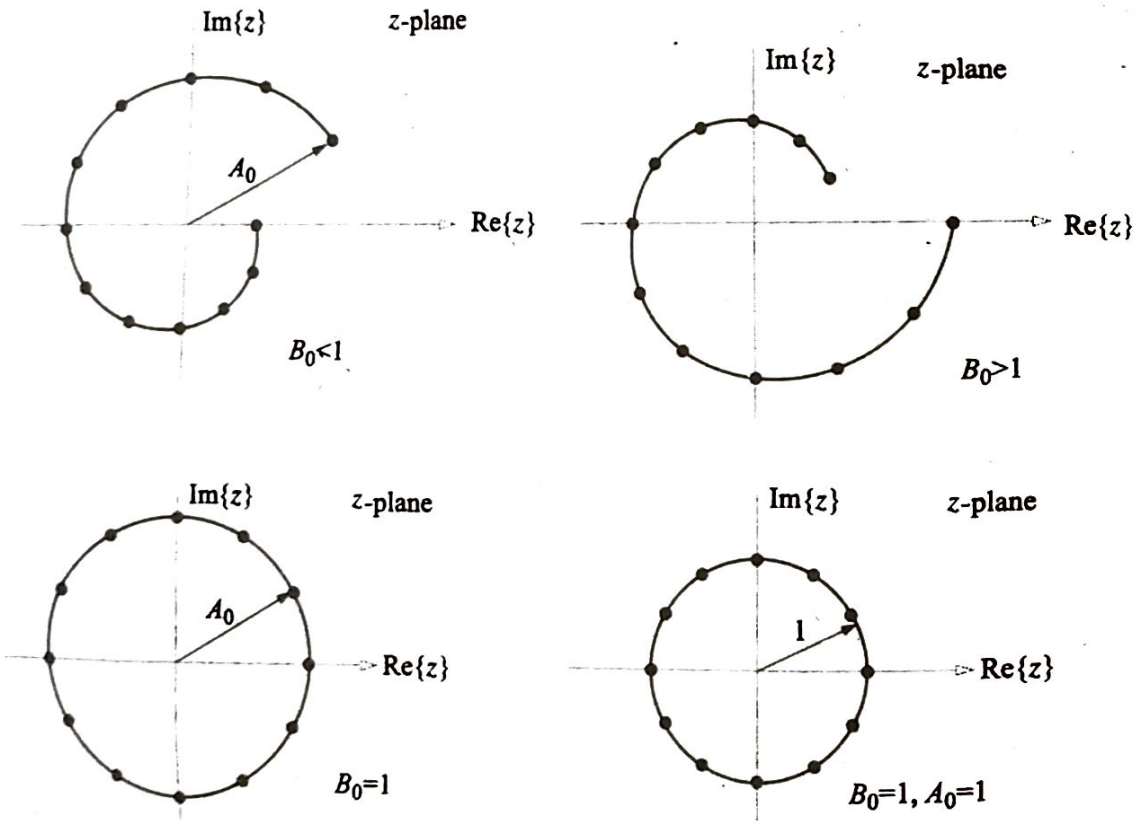


Fig. 3.23 Sample contours for the CZT.

Fig. 3.23 shows typical cases for $B_0 < 1$, for which the sequence of points $\{z_k\}$ spirals towards the origin, and for $B_0 > 1$, the sequence of points $\{z_k\}$ spirals away from the origin. If $|B_0| = 1$, the sequence $\{z_k\}$ lies on a circle of radius A_0. The special case of $A = 1$, $B = e^{-j\frac{2\pi}{N}}$, and $M = N$ corresponds to the DFT, and the contour is the unit circle.

Substituting equation (3.35) in equation (3.34) we get

$$X(z_k) = \sum_{n=0}^{N-1} x(n) A^{-n} B^{kn}, \qquad 0 \leqslant k \leqslant M-1 \tag{3.36}$$

The next step in the derivation is to make the observation

$$nk = \frac{1}{2}\left[n^2 + k^2 - (n-k)^2\right] \tag{3.37}$$

Substituting equation (3.37) in equation (3.36), we get

$$X(z_k) = \sum_{n=0}^{N-1} x(n) A^{-n} B^{\frac{n^2}{2}} B^{\frac{k^2}{2}} B^{\frac{-(n-k)^2}{2}} \tag{3.38}$$

Let us define two functions:

$$g(n) = x(n) A^{-n} B^{\frac{n^2}{2}}$$

and

$$h(n) = B^{\frac{-n^2}{2}}$$

Making use of the above functions in equation (3.38), we get

$$X(z_k) = \sum_{n=0}^{N-1} g(n) \frac{1}{h(k)} h(n-k), \quad 0 \leqslant k \leqslant M-1 \tag{3.39}$$

In preparation for interpreting equation (3.39) as the output of an LTI system, we obtain more familiar notation by replacing k by n and n by k in equation (3.39):

$$X(z_n) = \sum_{k=0}^{N-1} g(k) \frac{1}{h(n)} h(k-n), \quad 0 \leqslant n \leqslant M-1$$

Also $h(k-n) = h(n-k)$. Thus the above summation is recognized as the convolution of the sequences $x(n)$ and $h(n)$. Hence,

$$X(z_n) = \frac{1}{h(n)} [g(n) * h(n)] \tag{3.40}$$

The computation of equation (3.40) is the chirp-$\mathcal{Z}$ transform algorithm, which is illustrated by the block diagram of Fig. 3.24. In otherwords, equation (3.40) forms the basis for developing the block diagram for chirp-$\mathcal{Z}$ transformer.

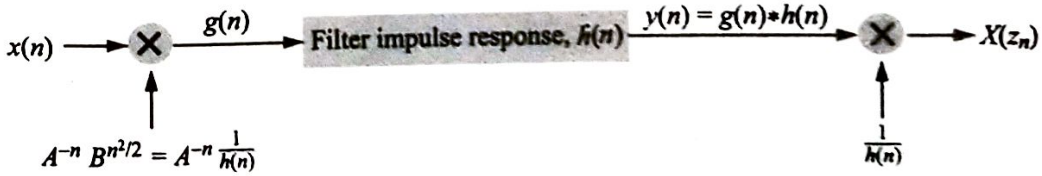


Fig. 3.24 Block diagram representation for chirp-$\mathcal{Z}$ transform.

If A and B_0 are unity, the sequence $h(n)$ is given by

$$h(n) = \left[e^{-j\phi_0}\right]^{-\frac{n^2}{2}} = e^{j\frac{n^2\phi_0}{2}} = e^{j\omega n},$$

where $\omega = \frac{n\phi_0}{2}$.

The above sequence can be thought of as a complex exponential sequence with a linearly increasing frequency. Such signals are called *chirp signals* in radar systems, hence the origin of the name.

All operations illustrated in Fig. 3.24 could, of course, be carried out digitally. However, the convolution operation for the chirp-$\mathcal{Z}$ transform can be implemented by means of *charge transfer devices* (CTDs), and such chirp-$\mathcal{Z}$ transformers are available commercially. The major advantage of a CTD-implemented chirp-$\mathcal{Z}$ transformer over the equivalent digital implementation appears to be the cost. However, CTD implementation is not 100% efficient.

The chirp-$\mathcal{Z}$ transform algorithm can be used for the enhancement of poles in spectral analysis, high-resolution analysis, interpolation of data in the time-domain for a change in the sampling rate and many other applications.

3.14.2 Goertzel algorithm

The Goertzel algorithm, is actually a linear filtering technique that is computationally quite efficient when the DFT must be computed for a relatively small number of frequency points.

The Goertzel algorithm takes the advantage of the periodicity of the sequence W_N^k in order to reduce computations. The goal is to express the basic DFT formula as a convolution of two sequences so that a linear filter can be realized. Since $W_N^{-kN} = 1$, we multiply the DFT formula by this factor to get

$$\begin{aligned} X(k) &= \sum_{m=0}^{N-1} x(m) W_N^{km} W_N^{-kN} \\ &= \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)} \end{aligned}$$

Since, $x(m)$ is a finite-length sequence in the interval: $0 \leqslant m \leqslant N-1$, we can change the limits in the above summation as

$$X(k) = \sum_{m=-\infty}^{\infty} x(m) W_N^{-k(N-m)}$$

Let us define the sequence:

$$y_k(n) = \sum_{m=-\infty}^{\infty} x(m) W_N^{-(n-m)k} \tag{3.41}$$

so that $$X(k) = y_k(n)\big|_{n=N}$$

It may be noted that equation (3.41) is the convolution sum of sequences $x(n)$ and W_N^{-kn}.

That is, $$y_k(n) = x(n) * W_N^{-kn} \tag{3.42}$$

The above equation can be realized by making $x(n)$ as the input of a linear filter having the impulse response $h_k(n) = W_N^{-kn}$ and then taking $y_k(n)$ as the output of the filter. This aspect is illustrated in Fig. 3.25.

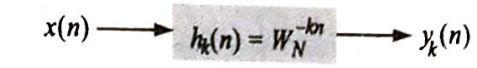


Fig. 3.25 Realization of convolution operation by a linear filter (Goertzel filter).

The transfer function of this linear filter known as *Goertzel filter* is found by taking the $\mathcal{Z}$-transform of $h_k(n)$, which gives

$$\begin{aligned} H_k(z) &= \sum_{n=0}^{\infty} h_k(n) z^{-n} \\ &= \sum_{n=0}^{\infty} \left(W_N^{-k} z^{-1}\right)^n \end{aligned} \tag{3.43}$$

The lower limit of summation is zero, since the filter is causal in nature.
Recall the infinite summation formula:

$$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}, \quad |a| < 1$$

Applying the above infinite summation formula to equation (3.43) gives

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} \tag{3.44}$$

$$\Rightarrow \quad \frac{Y_k(z)}{X(z)} = \frac{1}{1 - W_N^{-k} z^{-1}}$$

Cross-multiplying and taking inverse $\mathcal{Z}$-transform on both the sides, we get

$$y_k(n) - W_N^{-k} y_k(n-1) = x(n) \tag{3.45}$$

The above equation is represented by a block diagram as shown in Fig. 3.26.

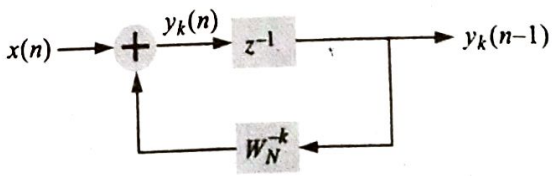


Fig. 3.26 First-order Goertzel filter.

The first-order filter requires N complex multiplications to compute the output at the N^{th} sample.

The complex multiplications inherent in equation (3.45) can be avoided by combining the complex-conjugate poles. Multiplying and dividing the right-hand side of equation (3.44) by $1 - W_N^k z^{-1}$, we get

$$\begin{aligned} H_k(z) &= \frac{1 - W_N^k z^{-1}}{\left(1 - W_N^{-k} z^{-1}\right)\left(1 - W_N^k z^{-1}\right)} \\ &= \frac{1 - W_N^k z^{-1}}{1 - W_N^k z^{-1} - W_N^{-k} z^{-1} + z^{-2}} \\ &= \frac{1 - W_N^k z^{-1}}{1 - e^{-j\frac{2\pi}{N}k} z^{-1} - e^{j\frac{2\pi}{N}k} z^{-1} + z^{-2}} \\ \Rightarrow \quad H_k(z) &= \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}} = H_1(z) H_2(z) \end{aligned}$$

where $H_1(z)$ contains only poles of $H_k(z)$ and $H_2(z)$ contains only zeros of $H_k(z)$.

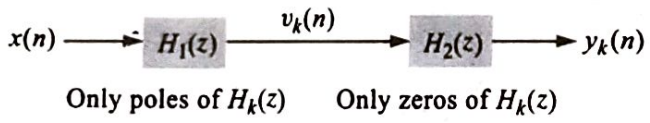


Fig. 3.27 Decomposition of $H_k(z)$ into $H_1(z)$ and $H_2(z)$.

From Fig. 3.27, we can write

$$\begin{aligned} V_k(z) &= X(z) H_1(z) \\ \Rightarrow \quad V_k(z) &= \frac{1}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}} X(z) \end{aligned}$$

Cross-multiplying and taking inverse $\mathcal{Z}$-transform on both the sides, we get

$$v_k(n) = 2\cos\left(\frac{2\pi k}{N}\right) v_k(n-1) - v_k(n-2) + x(n) \tag{3.46}$$

Again from Fig. 3.27, we can write

$$\begin{aligned} Y_k(z) &= H_2(z) V_k(z) \\ \Rightarrow \quad Y_k(z) &= \left(1 - W_N^k z^{-1}\right) V_k(z) \end{aligned}$$

Hence,

$$y_k(n) = v_k(n) - W_N^k v_k(n-1) \tag{3.47}$$

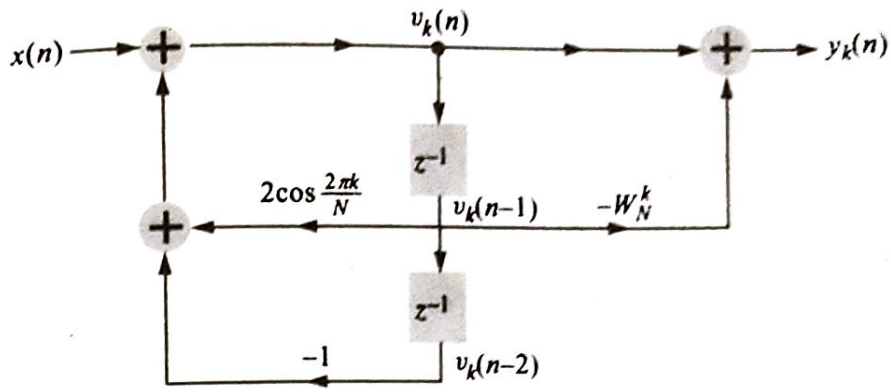


Fig. 3.28 Direct form-II realization of Goertzel filter.

The initial conditions $v_k(-1)$ and $v_k(-2)$ are assumed to be zero. The direct form-II realization of second-order Goertzel filter is shown in Fig. 3.28. The recursive equation (3.46) is iterated for $n = 0, 1, \cdots N$, but equation (3.47) is computed only once at time $n = N$. Each iteration needs one real multiplication and two additions. Finally, for a real input sequence $x(n)$, this algorithm needs $N + 1$ real multiplications to give not only $X(k)$, but also due to symmetry, the values of $X(N - k)$.

We reiterate the fact that the Goertzel algorithm is particularly attractive when the DFT is to be evaluated at a relatively small number of values compared to $\log_2 N$. Otherwise, FFT algorithm is preferred.

3.15 Frequency Analysis of Signals Using DFT

One of the main uses of the DFT is to find the frequency spectrum of an infinite duration signal when it is observed for only a finite duration. The act of truncating an infinite length duration signal to a finite length duration (the duration being the period of observation) is called *windowing*. Let $x(n)$ be an infinite duration signal, $w(n)$ be a window function and $x_w(n)$ be a windowed sequence. The simplest possible window function is the rectangular window function and is defined below:

$$w(n) = \begin{cases} 1, & 0 \leqslant n \leqslant N-1 \\ 0, & \text{otherwise} \end{cases}$$

The DTFT of the windowed signal, $x_w(n) = x(n)w(n)$ is

$$\begin{aligned} X_w(\omega) &= \text{DTFT}\{w(n)x(n)\} \\ &= \sum_{n=-\infty}^{\infty} x(n)w(n)e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} x(n)e^{-j\omega n} \end{aligned}$$

It may be noted that we can relate the right-hand side of the above equation to the DFT of the finite sequence $x(0), \cdots x(N-1)$ as

$$X(k) = X_w(\omega)\big|_{\omega=\frac{2\pi k}{N}}, \quad k = 0, 1, \cdots N-1$$

In other words, we sample $X_w(\omega)$ at discrete frequencies $\omega = \frac{2\pi k}{N}$ to get $X(k)$. Let us now proceed to find the effect of windowing operation on the frequency. To keep things simple, let us assume $x(n)$ to be a sinusoidal signal defined as $x(n) = 2\cos\omega_0 n$, for $-\infty < n < \infty$. Then, the DTFT of this signal windowed by $w(n)$ is

$$\begin{aligned} X_w(\omega) &= \text{DTFT}\{x(n)w(n)\} \\ &= \sum_{n=-\infty}^{\infty} x(n)w(n)e^{-j\omega n} \end{aligned}$$

Since $x(n) = e^{j\omega_0 n} + e^{-j\omega_0 n}$, we get

$$\begin{aligned} X_w(\omega) &= \sum_{n=-\infty}^{\infty} w(n)e^{-j(\omega-\omega_0)n} + \sum_{n=-\infty}^{\infty} w(n)e^{-j(\omega+\omega_0)n} \\ &= W(\omega-\omega_0) + W(\omega+\omega_0) \end{aligned}$$

where $W(\omega)$ is the DTFT of the rectangular window function and is found as follows:

$$\begin{aligned} W(\omega) &= \text{DTFT}\{w(n)\} \\ &= \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n} = \sum_{n=0}^{N-1} e^{-j\omega n} \end{aligned}$$

Recall the formula: $\sum_{n=0}^{N-1} a^n = \dfrac{1-a^N}{1-a}, \quad a \neq 1.$

Applying the above formula, we get

$$\begin{aligned} W(\omega) &= \frac{1-e^{-j\omega N}}{1-e^{-j\omega}} \\ &= e^{-j\omega\frac{N}{2}} \frac{\left[e^{j\omega\frac{N}{2}} - e^{-j\omega\frac{N}{2}}\right]}{e^{\frac{-j\omega}{2}}\left[e^{\frac{j\omega}{2}} - e^{\frac{-j\omega}{2}}\right]} \\ &= e^{-j\omega\frac{(N-1)}{2}} \frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} \end{aligned}$$

$$\begin{aligned} \text{Hence,}\quad X_w(\omega) &= |W(\omega-\omega_0) + W(\omega+\omega_0)| \\ &= \left| \frac{\sin\left[(\omega-\omega_0)\frac{N}{2}\right]}{\sin\left[(\omega-\omega_0)\frac{1}{2}\right]} e^{-j(\omega-\omega_0)\left(\frac{N-1}{2}\right)} + \frac{\sin\left[(\omega+\omega_0)\frac{N}{2}\right]}{\sin\left[(\omega+\omega_0)\frac{1}{2}\right]} e^{-j(\omega+\omega_0)\left(\frac{N-1}{2}\right)} \right| \end{aligned}$$

Fig. 3.29 compares the magnitude frequency spectra of $x(n)$ and $x_w(n)$. Withouting windowing, the sinusoidal sequence $x(n)$ is perfectly localized in frequency, since

$$\text{DFT}\{2\cos\omega_0 n\} = 2\pi\delta(\omega-\omega_0) + 2\pi\delta(\omega+\omega_0)$$

However, the spectrum of the windowed sequence, $x_w(n)$ is not perfectly localized any more. The energy smears around the frequency ω_0 within the major lobe and trickles to other frequencies by the sidelobes of the sinc function. This degradation can be controlled by minimizing the width of major lobe and by reducing the maximum amplitude of the first sidelobe.

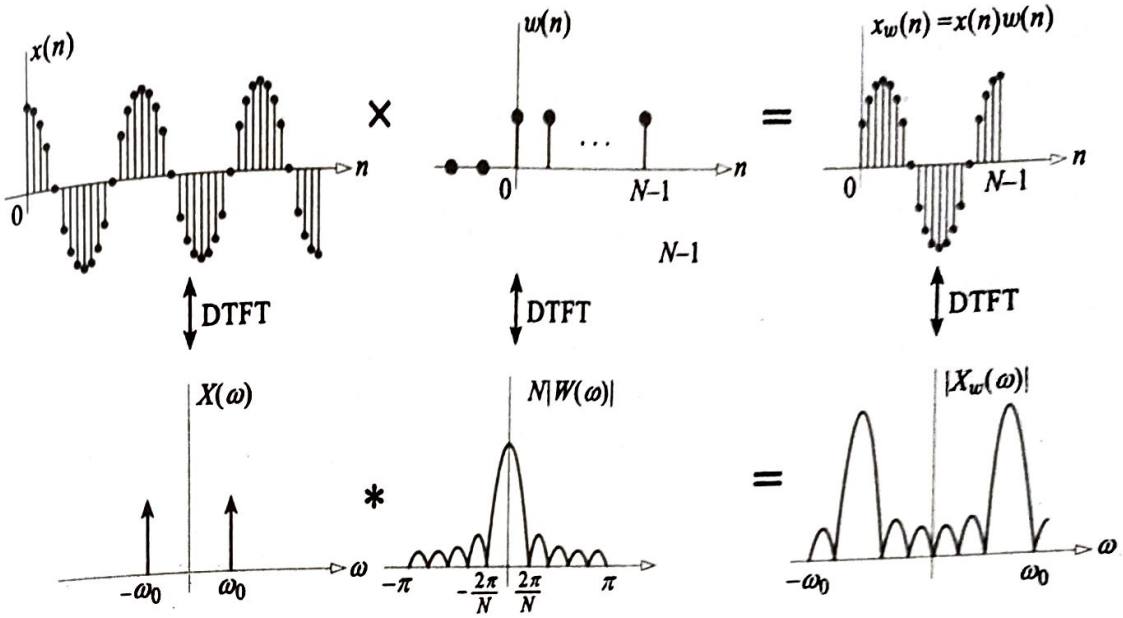


Fig. 3.29 Spectra of $x(n)$ and $x_w(n)$.

Adding to the above fact that by the DFT, we compute $X_w(\omega)$ at discrete frequencies $\omega = \frac{2\pi k}{N}$ as shown in Fig. 3.30, and we realize two immediate effects of windowing: Loss of resolution and leakage to other frequencies.

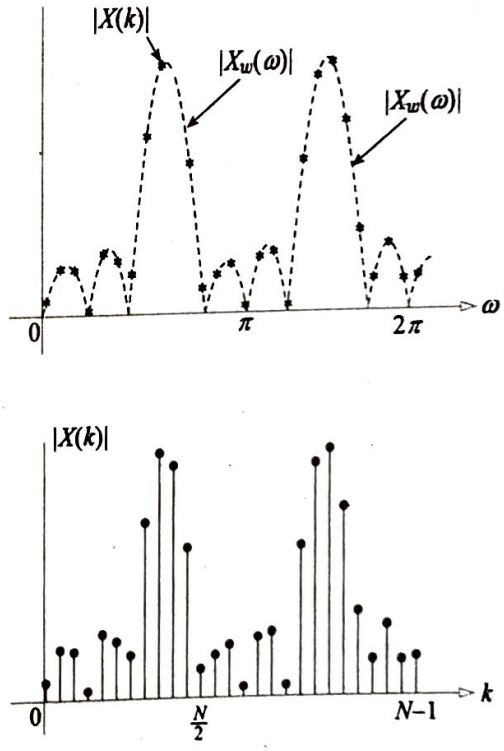


Fig. 3.30 DFT obtained by sampling $|X_w(\omega)|$.

Loss of resolution

IF the DFT $X(k)$ has a peak at index k_0, the actual frequency, ω_0 can be anywhere in the range:

$$(k_0 - 1)\frac{2\pi}{N} < \omega_0 < (k_0 + 1)\frac{2\pi}{N}$$

where N, we recall, is the data length in terms of number of samples.

Leakage to other frequencies

The sidelobes of $W(\omega)$ also have an effect on the frequency resolution. The maximum magnitude of sidelobes does not decrease as $N \to \infty$ and therefore adding data points is not of any help. The only alternative is to use different windows. In any case, given a fixed set of data points, we have to accept a trade-off between frequency resolution (width of the mainlobe of $W(\omega)$) and sidelobes.

Honing Your Skills

HS-3.1 Find the circular convolution of the sequences

$$x(n) = (2, 3, -1, 5), \quad h(n) = (1, 4, -2, -3)$$

using matrix method. Verify the result using DFT.

□ **Solution**

This problem illustrates one of the several ways to implement circular convolution on a digital computer.

Let

$$\begin{aligned} y(n) &= x(n) \circledast_N h(n) \\ y(n) &\triangleq \sum_{m=0}^{N-1} x(m)h((n-m))_N, \quad 0 \leqslant n \leqslant N-1 \end{aligned}$$

For each value of n, we get a linear equation and thus there are a total of N linear equations. Putting these equations in the matrix form, we get

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-2) \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(N-1) & \cdots & h(2) & h(1) \\ h(1) & h(0) & \cdots & h(3) & h(2) \\ \vdots & \vdots & & \vdots & \vdots \\ h(N-2) & h(N-3) & \cdots & h(0) & h(N-1) \\ h(N-1) & h(N-2) & \cdots & h(1) & h(0) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-2) \\ x(N-1) \end{bmatrix}$$

Observe the elements of the sequence $h(n)$ in the above matrix. A matrix having such a structure is called circulant, because its rows are consecutive circular shifts of the first row.

For the present problem, $N = 4$. Consequently, we get

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 1 & -3 & -2 & 4 \\ 4 & 1 & -3 & -2 \\ -2 & 4 & 1 & -3 \\ -3 & -2 & 4 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} 15 \\ 4 \\ -8 \\ -11 \end{bmatrix}$$

$$\Rightarrow \qquad y(n) = (15, 4, -8, -11)$$

Verification

$$\begin{aligned} X(k) &= 2 + 3W_4^k - W_4^{2k} + 5W_4^{3k} \\ H(k) &= 1 + 4W_4^k - 2W_4^{2k} - 3W_4^{3k} \end{aligned}$$

Hence,

$$\begin{aligned} Y(k) &= X(k)H(k) \\ &= 15 + 4W_4^k - 8W_4^{2k} - 11W_4^{3k} \end{aligned}$$

Taking IDFT of $Y(k)$, we get

$$\begin{aligned} y(n) &= 15\delta(n) + 4\delta(n-1) - 8\delta(n-2) - 11\delta(n-3) \\ \Rightarrow \qquad y(n) &= (15, 4, -8, -11). \end{aligned}$$

HS- 3.2 Fig. HS.3.2 shows the flow graph representation of a DIT-FFT algorithm for $N = 8$. The dashed line shows a path from sample $x(6)$ to DFT sample $X(7)$.

a. What is the gain along the path that is shown by dashed lines in Fig. HS.3.2 ?

b. How many paths in the flow graph begin at $x(6)$ and end at $X(7)$? Is this true in general; that is, how many paths are there between each input sample and each output sample?

c. Let us consider the DFT sample $X(2)$. By tracing paths in the flow graph, show that each input sample contributes the proper amount to the output DFT sample; that is, verify that

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

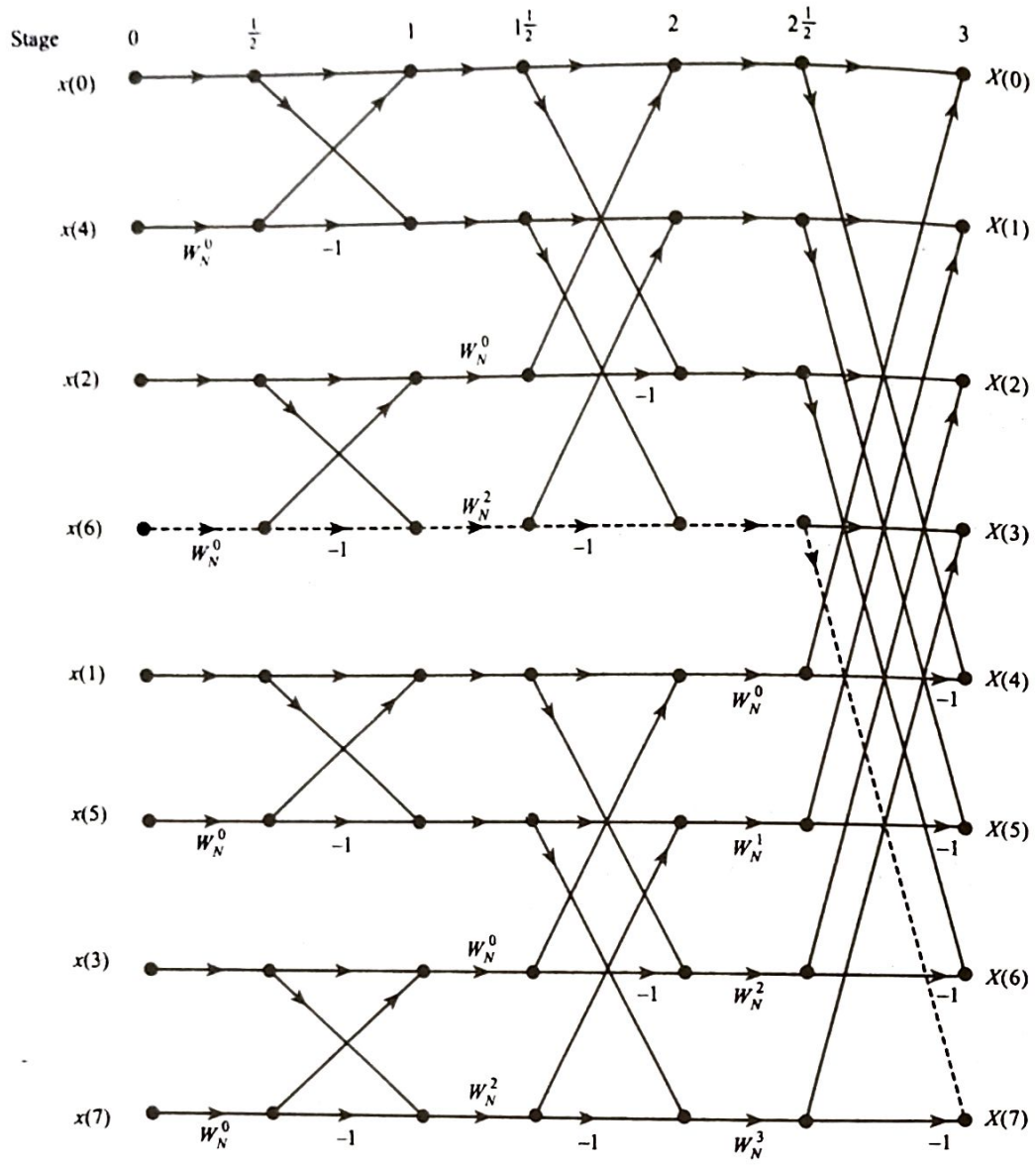


Fig. HS.3.2

□ Solution

a. Gain = $W_N^0 \times -1 \times W_N^2 \times -1 \times 1 \times 1 = W_N^2$

b. In general, there is only one path between each input sample and output sample.

c.

$x(0)$ to $X(2)$: gain is 1

$x(1)$ to $X(2)$: gain is W_N^2

$x(2)$ to $X(2)$: gain is $-W_N^0 = -1$

$x(3)$ to $X(2)$: gain is $-W_N^0 \times W_N^2 = -W_N^2$

$x(4)$ to $X(2)$: gain is $W_N^0 = 1$

$x(5)$ to $X(2)$: gain is $W_N^0 \times W_N^2 = W_N^2$

$x(6)$ to $X(2)$: gain is $-W_N^0 \times W_N^0 = -1$

$x(7)$ to $X(2)$: gain is $-W_N^2$

The above mentioned gains can be checked by evaluating $X(2)$ and they are the coefficients of $x(n)$, for $0 \leqslant n \leqslant 7$.

$$\begin{aligned} X(2) &= \sum_{n=0}^{7} x(n) W_N^{2n} \\ &= x(0) + x(1)W_N^2 + x(2)W_N^4 + x(3)W_N^6 + x(4)W_N^8 \\ &\quad + x(5)W_N^{10} + x(6)W_N^{12} + x(7)W_N^{14} \end{aligned}$$

Since $N = 8$, we have

$$W_N^4 = -1, \quad W_N^6 = -W_N^2, \quad W_N^8 = 1$$
$$W_N^{10} = W_N^2, \quad W_N^{12} = -1, \quad W_N^{14} = -W_N^2$$

Hence,
$$\begin{aligned} X(2) &= x(0) + x(1)W_N^2 + x(2)[-1] + x(3)\left[-W_N^2\right] + x(4)[1] \\ &\quad + x(5)W_N^2 + x(6)[-1] + x(7)\left[-W_N^2\right] \end{aligned}$$

This verifies the various gains already computed.

HS-3.3 A designer is having a number of 8-point FFT chips. Show explicitly how he should interconnect three chips in order to compute a 24-point DFT.

□ Solution

Let us create three subsequences of length-8 each.

$$\begin{aligned} \text{Then,} \quad Y(k) &= \sum_{n=0,3,6,\cdots}^{21} y(n)W_N^{kn} + \sum_{n=1,4,7,\cdots}^{22} y(n)W_N^{kn} + \sum_{n=2,5,8,\cdots}^{23} y(n)W_N^{kn} \\ &= \sum_{i=0}^{7} y(3i)W_N^{k(3i)} + \sum_{i=0}^{7} y(3i+1)W_N^{k(3i+1)} + \sum_{i=0}^{7} y(3i+2)W_N^{k(3i+2)} \\ &= \sum_{i=0}^{7} y(3i)W_{\frac{N}{3}}^{ki} + \sum_{i=0}^{7} y(3i+1)W_{\frac{N}{3}}^{ki}W_N^k + \sum_{i=0}^{7} y(3i+2)W_{\frac{N}{3}}^{ki}W_N^{2k} \\ &= Y_1(k) + W_N^k Y_2(k) + W_N^{2k} Y_3(k), \quad 0 \leqslant k \leqslant 23 \end{aligned}$$

Where $Y_1(k)$, $Y_2(k)$ and $Y_3(k)$ are the 8-point DFTs of the subsequences. The above equation gives the algebra for combining $Y_1(k)$, $Y_2(k)$ and $Y_3(k)$. Also, $Y_1(k)$, $Y_2(k)$ and $Y_3(k)$ are periodic with a period equal to 8.

HS-3.4 Find $X(2)$. Given $x(n) = (1, 0, 1, 0)$. Use Goertzel algorithm.

☐ **Solution**

According to Goertzel algorithm,

$$X(k) = y_k(n)|_{n=N}$$

where $y_k(n)$ is found using the following recursive formula.

$$y_k(n) - W_N^{-k} y_k(n-1) = x(n).$$

Since, $k = 2$ and $N = 4$, we get

$$X(2) = y_2(4)$$

and

$$y_2(n) - W_4^{-2} y_2(n-1) = x(n).$$

Since $W_4^{-2} = -1$, the recursive equation becomes

$$y_2(n) + y_2(n-1) = x(n)$$

***Case* (i): $n = 0$**

$$y_2(0) + y_2(-1) = x(0)$$

Assuming $y_2(-1) = 0$, we get

$$y_2(0) + 0 = 1 \quad \Rightarrow \quad y_2(0) = 1$$

***Case* (ii): $n = 1$**

$$y_2(1) + y_2(0) = x(1)$$

$$\Rightarrow \quad y_2(1) + 1 = 0 \quad \Rightarrow \quad y_2(1) = -1$$

***Case* (iii): $n = 2$**

$$y_2(2) + y_2(1) = x(2)$$

$$\Rightarrow \quad y_2(2) - 1 = 1 \quad \Rightarrow \quad y_2(2) = 2$$

***Case* (iv): $n = 3$**

$$y_2(3) + y_2(2) = x(3)$$

$$\Rightarrow \quad y_2(3) + 2 = 0 \quad \Rightarrow \quad y_2(3) = -2$$

***Case* (v): $n = 4$**

$$y_2(4) + y_2(3) = x(4)$$

$$\Rightarrow \quad y_2(4) - 2 = 0 \quad \Rightarrow \quad y_2(4) = 2$$

Hence,

$$X(2) = y_2(4) = 2.$$

HS-3.5 Assume that a complex multiplication takes 1 μs and that the amount of time taken to compute N-point DFT is determined by the amount of time it takes to perform all of the multiplications.

a. How much time it takes to compute a 64-point DFT directly?

b. How much time is required if an FFT is used?

c. Repeat parts (a) and (b) for a 256-point DFT.

Solution

a. The DFT $X(k)$ of an N-point sequence $x(n)$ is defined as follows.

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

Including multiplication of $x(n)$ by ± 1 as complex multiplications, computing an N-point DFT directly requires N^2 complex multiplications. Since, one complex multiplication requires 1 μs, the direct evaluation of a 64-point DFT requires

$$t_{\text{DFT}} = (64)^2 \times 10^{-6} = 4.096 \times 10^{-3} \text{ seconds}$$

b. With a radix-2 FFT, the number of complex multiplications equals $\frac{N}{2} \log_2 N$. Hence, the amount of time to compute a 64-point DFT using an FFT algorithm is

$$t_{\text{FFT}} = \frac{64}{2} \times \log_2 64 \times 1 \times 10^{-6} = 192 \times 10^{-6} \text{ seconds}$$

c. If the length of the DFT is increased by a factor of 4 to $N = 256$, the number of complex multiplications necessary to compute DFT directly increases by a factor of 16. Hence, the time required to evaluate the DFT directly is

$$t_{\text{DFT}} = 16 \times 4.096 \times 10^{-3} = 0.065 \text{ seconds}$$

On the otherhand, if an FFT is used, the time taken to evaluate the number of complex multiplications is

$$t_{\text{FFT}} = \frac{256}{2} \times \log_2 256 \times 1 \times 10^{-6} = 1.024 \times 10^{-3} \text{ seconds}$$

HS-3.6 We wish to compute M-point CZT samples $X(z_k)$, $k = 0, 1, \cdots M-1$, of a length-N sequence according to

$$X(z_k) = \sum_{n=0}^{N-1} x(n) A^{-n} B^{kn}, \quad 0 \leqslant k \leqslant M-1$$

where $z_k = AB^{-k}$ with $A = A_0 e^{j\theta_0}$ and $B = B_0 e^{-j\phi_0}$.

What are the values of A_0, θ_0, B_0 and ϕ_0 if the CZT needs to be calculated at points $\{z_k\}$ on the real axis in the z-plane such that $z_k = \alpha^k$, $0 \leqslant k \leqslant M-1$, for α real and $\alpha \neq \pm 1$?