CHAPTER 20

Fast Fourier Transforms

One of the main advantages of working with the DFT is that efficient methods exist for the evaluation of both the DFT and its inverse. These efficient methods are collectively known as "Fast Fourier Transform" or FFT techniques. Operations requiring the DFT are often carried out by implementing the FFT instead, so much so that the FFT forms the cornerstone of most practical and efficient implementations of discrete-time signal processing algorithms. The availability of FFT techniques has enabled the widespread use of discrete-time filtering and processing algorithms in a wide range of consumer and military electronics. There are many variations of the FFT. The purpose of this chapter is to motivate and discuss two of the most popular versions; these versions are sufficient to convey the main concepts underlying FFT algorithms.

20.1 COST OF DFT

In order to appreciate the computational savings that will be afforded by the FFT, let us first examine the computational cost that is involved in evaluating the conventional $N-$point DFT of a sequence. Thus, let $x(n)$ denote a causal sequence of duration N; the samples of $x(n)$ may be real or complex-valued. For generality, we assume the samples of $x(n)$ are complex-valued and evaluate the computational cost accordingly. The $N-$point DFT of $x(n)$ is given by the coefficients:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi k}{N}n}, \quad k = 0, 1, \ldots, N-1 \tag{20.1}$$

The sequence $x(n)$ can be recovered from its DFT through the inverse operation:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi k}{N}n}, \quad n = 0, 1, \ldots, N-1 \tag{20.2}$$

For convenience of notation, we let the complex number W_N denote the following $N-$th root of unity:

$$\boxed{W_N \triangleq e^{-j\frac{2\pi}{N}}} \tag{20.3}$$

This scalar is sometimes called the *twiddle* factor in the FFT literature. Then, evaluating the nk-th power of W_N gives

$$(W_N)^{nk} = e^{-j\frac{2\pi k}{N}n}$$

so that expressions (20.1) and (20.2) for the $N-$point DFT and its inverse can be re-expressed in terms of the factor W_N as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0, 1, \ldots, N-1 \tag{20.4}$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) W_N^{-nk}, \quad n = 0, 1, \ldots, N-1 \tag{20.5}$$

The complex number W_N defined by (20.3) satisfies several easily-verifiable properties such as:

$$\boxed{(W_N)^q = W_{N/q}, \quad (W_N)^{q+\frac{N}{2}} = -(W_N)^q, \quad (W_N)^{qN/2} = (-1)^q} \tag{20.6}$$

for any integer q. We shall exploit these properties for W_N extensively while deriving the FFT methods.

Proof: Indeed, note that

$$(W_N)^q = \left(e^{-j\frac{2\pi}{N}}\right)^q = e^{-j\frac{2\pi q}{N}} = e^{-j\frac{2\pi}{N/q}} = W_{N/q} \tag{20.7}$$

$$(W_N)^{q+\frac{N}{2}} = e^{-j\frac{2\pi}{N}(q+\frac{N}{2})} = e^{-j\frac{2\pi q}{N}} \cdot e^{-j\pi} = -e^{-j\frac{2\pi q}{N}} = -W_N^q \tag{20.8}$$

$$(W_N)^{qN/2} = e^{-j\frac{2\pi}{N} q \frac{N}{2}} = e^{-j\pi q} = (-1)^q \tag{20.9}$$

$\diamond$

Now refer to expression (20.4). The evaluation of each coefficient $X(k)$ involves computing N complex multiplications of the form $x(n)W_N^{nk}$ and $N-1$ complex additions. Each complex addition involves two real additions since

$$(a+jb) + (c+jd) = (a+c) + j(b+d) \tag{20.10}$$

whereas each complex multiplication involves four real multiplications and two real additions:

$$(a+jb) \cdot (c+jd) = (ac-bd) + j(ad+bc) \tag{20.11}$$

It follows that the evaluation of each coefficient $X(k)$ involves $4N$ real multiplications and $4N-2$ real additions, so that the cost of computing all N DFT coefficients $\{X(k)\}$ is

$$\text{computing } N\text{-point DFT requires } \begin{cases} 4N^2 - 2N & \text{real additions} \\ 4N^2 & \text{real multiplications} \end{cases} \tag{20.12}$$

We can also express the same cost in terms of complex operations:

$$\text{computing } N\text{-point DFT requires } \begin{cases} N^2 - N & \text{complex additions} \\ N^2 & \text{complex multiplications} \end{cases} \tag{20.13}$$

It is sufficient for our purposes to work with a rough figure for the computational cost. Thus, we blend the number of complex additions and multiplications and say that the

evaluation of the $N-$point DFT of a sequence through expression (20.4) has complexity of the order of $2N^2$ complex operations, written as:

$$\boxed{N\text{-point DFT requires } O(2N^2) \text{ complex operations}} \tag{20.14}$$

where the notation $O(m)$ means that the cost is of the order of m for large m. A similar complexity figure holds for performing the inverse DFT operation (20.5). If more precise cost figures are desired, then one can fall back on expressions (20.12)–(20.13).

Example 20.1 (Cost of 1024-point DFT)

The evaluation of a 1024-point DFT requires approximately:

$$(1024)^2 = 1,048,576 = \text{complex multiplications} \tag{20.15}$$

$$(1024) \times (1023) = 1,047,552 = \text{complex additions} \tag{20.16}$$

In other words, it is necessary to perform of the order of 2 million complex operations. To have an idea of what this cost entails, consider an integrated circuit processor operating at the rate of 3.5GHz. Assume further that each complex operation requires one clock cycle, which is the inverse of the processor frequency and is approximately 0.286×10^{-9} sec. We then find that the evaluation of the 1024 DFT coefficients would require about 0.6 msec. It is seen that the computational cost of the DFT translates into a demand on the amount of time that is necessary to evaluate its coefficients. ◇

We now move on to derive two popular efficient methods for evaluating the DFT (and its inverse) by resorting to divide-and-conquer strategies. As indicated before, there are several variants of the FFT algorithm. We limit the discussion in this chapter to the so-called radix-2 decimation-in-time and radix-2 decimation-in-frequency versions, which are widely used. Other FFT variants rely on similar divide-and-conquer constructions.

20.2 DECIMATION-IN-TIME FFT

The radix-2 variants of the FFT require the length N to be a power of 2, say, $N = 2^p$, for some positive integer p. This requirement is not restrictive since we can always pad the sequence $x(n)$ with additional trailing zeros in order to extend its length to the nearest power of 2. The padding of zeros does not alter the DTFT, $X(e^{j\omega})$; it therefore does not alter the DFT coefficients, $X(k)$, as well.

20.2.1 Derivation of Algorithm

The decimation-in-time algorithm is based on the idea of splitting the time-domain sequence, $x(n)$, into even and odd-indexed samples. This is in contrast to the decimation-in-frequency algorithm described later, and which relies on splitting the frequency-domain coefficients, $X(k)$, into their even and odd-indexed values.

Now, since N is even, we can split the sequence, $x(n)$, into two smaller sequences of duration $N/2$ each. In one sequence we group the even-indexed samples of $x(n)$ and in

the other sequence we group the odd-indexed samples of $x(n)$, say,

$$x_e(n) = \left\{ \boxed{x(0)}, x(2), x(4), \ldots, x(N-4), x(N-2) \right\} \tag{20.17}$$

$$x_o(n) = \left\{ \boxed{x(1)}, x(3), x(5), \ldots, x(N-3), x(N-1) \right\} \tag{20.18}$$

In (20.17)–(20.18), we are using the box notation to indicate the location of the sample of index $n = 0$ in the sequences $x_e(n)$ and $x_o(n)$. Figure 20.1 illustrates this construction for a particular sequence $x(n)$ of duration $N = 8$.

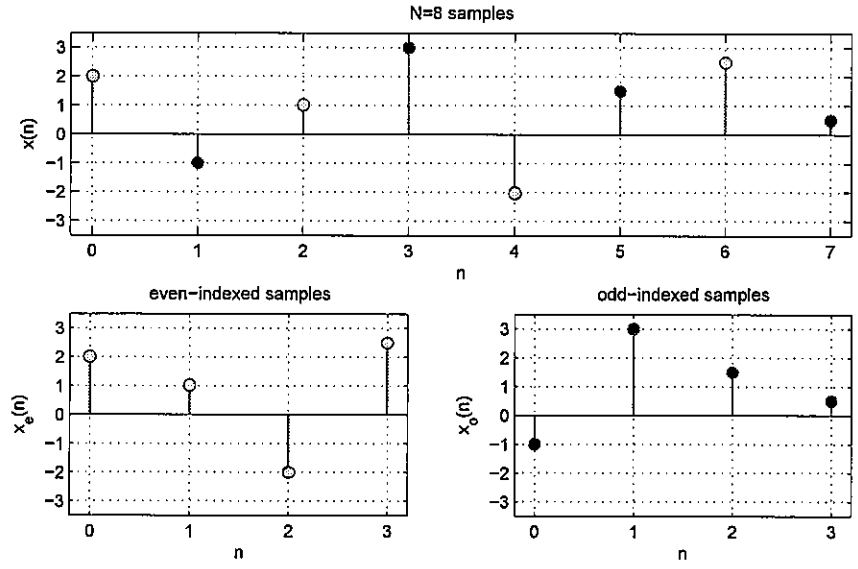

FIGURE 20.1 The sequence $x(n)$ of duration $N = 8$ (top) is decimated into two smaller sequences, $x_e(n)$ and $x_o(n)$, of duration 4 samples each (bottom).

Let $X_e(k)$ and $X_o(k)$ denote the $\frac{N}{2}$–point DFTs of $x_e(n)$ and $x_o(n)$, respectively:

$$\begin{aligned} X_e(k) &= \sum_{n=0}^{\frac{N}{2}-1} x_e(n) W_{N/2}^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{N/2}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{20.19}$$

$$\begin{aligned} X_o(k) &= \sum_{n=0}^{\frac{N}{2}-1} x_o(n) W_{N/2}^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{N/2}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{20.20}$$

Note that we have $N/2$ coefficients $X_e(k)$ and $N/2$ coefficients $X_o(k)$. Although we are limiting the DFT sequences $\{X_e(k), X_o(k)\}$ to the interval $0 \leq k \leq \frac{N}{2} - 1$, these sequences are actually periodic with period $N/2$. Therefore, whenever necessary, the co-

efficients $X_e(k)$ and $X_o(k)$ over the extended interval $0 \leq k \leq N-1$ are given by:

$$\{X_e(k),\ \ k = 0, 1, \ldots, N-1\} \ =$$

$$\left\{ \underbrace{\boxed{X_e(0)}, X_e(1), \ldots, X_e\left(\frac{N}{2}-1\right)}_{\text{one period with } \frac{N}{2} \text{ samples}}, \underbrace{X_e(0), X_e(1), \ldots, X_e\left(\frac{N}{2}-1\right)}_{\text{a second period with } \frac{N}{2} \text{ samples}} \right\} \tag{20.21}$$

and

$$\{X_o(k),\ \ k = 0, 1, \ldots, N-1\} \ =$$

$$\left\{ \underbrace{\boxed{X_o(0)}, X_o(1), \ldots, X_o\left(\frac{N}{2}-1\right)}_{\text{one period with } \frac{N}{2} \text{ samples}}, \underbrace{X_o(0), X_o(1), \ldots, X_o\left(\frac{N}{2}-1\right)}_{\text{a second period with } \frac{N}{2} \text{ samples}} \right\} \tag{20.22}$$

where the box notation is used to indicate the location of the coefficient at bin $k = 0$.

Given the above coefficients $X_e(k)$ and $X_o(k)$ over the extended interval $0 \leq n \leq N-1$, we now verify that the original $N-$point DFT of $x(n)$ can be determined from knowledge of $X_e(k)$ and $X_o(k)$ as follows:

$$\begin{aligned}
X(k) \ &\triangleq \ \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0, 1, \ldots, N-1 \\
&= \ \sum_{n=\text{even}} x(n) W_N^{kn} \ + \ \sum_{n=\text{odd}} x(n) W_N^{kn} \\
&= \ \sum_{m=0}^{\frac{N}{2}-1} x(2m) W_N^{2km} \ + \ \sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_N^{k(2m+1)} \\
&= \ \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x(2m) W_{N/2}^{km}}_{\overset{(20.19)}{=} X_e(k)} \ + \ W_N^k \cdot \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x(2m+1) W_{N/2}^{km}}_{\overset{(20.20)}{=} X_o(k)}
\end{aligned} \tag{20.23}$$

In other words, we find that

$$X(k) = X_e(k) \ + \ W_N^k \cdot X_o(k)\ , \ \ k = 0, 1, \ldots, N-1 \tag{20.24}$$

By further using the identity

$$W_N^{k+\frac{N}{2}} = -W_N^k \tag{20.25}$$

and the fact that $X_e(k)$ and $X_o(k)$ are periodic with period $N/2$, relation (20.24) for $X(k)$ can be rewritten in the equivalent form:

$$\boxed{\begin{cases} X(k) \ &= \ X_e(k) \ + \ W_N^k \cdot X_o(k)\ , \quad 0 \leq k \leq \frac{N}{2}-1 \\ X\left(k+\frac{N}{2}\right) \ &= \ X_e(k) \ - \ W_N^k \cdot X_o(k)\ , \quad 0 \leq k \leq \frac{N}{2}-1 \end{cases}} \tag{20.26}$$

Figure 20.2 illustrates the mapping from the coefficients $\{X_e(k), X_o(k)\}$ to the coefficients $\{X(k)\}$. The structure shown in the figure is known as a *butterfly* section. Result (20.26) therefore establishes that the $N-$point DFT coefficients $X(k)$ can be alternatively computed in the following manner:

(a) We split the original sequence $x(n)$ into two sub-sequences, $x_e(n)$ and $x_o(n)$, of duration $N/2$ each. One sequence contains the even-indexed samples of $x(n)$ and the other sequence contains the odd-indexed samples of $x(n)$.

(b) We determine the $\frac{N}{2}-$point DFTs $X_e(k)$ and $X_o(k)$, $k = 0, 1, \ldots, N/2 - 1$.

(d) We add and subtract the $N/2$ coefficients $\{X_e(k), W_N^k X_o(k)\}$ to generate $X(k)$ in the manner indicated in Fig. 20.2.

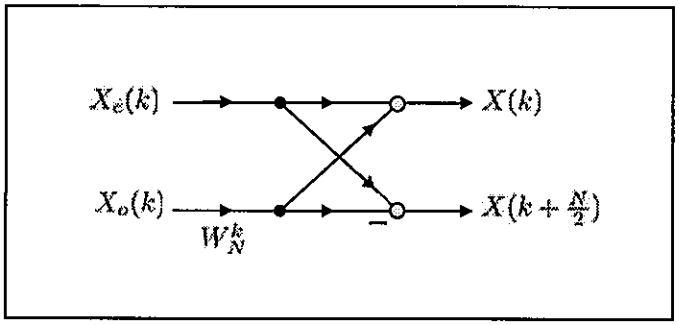

FIGURE 20.2 A flow diagram representation of the mapping (20.26) in terms of a butterfly section, where the bright circles represent adders.

The same decimation procedure can now be applied to the evaluation of the $\frac{N}{2}-$point DFTs $X_e(k)$ and $X_o(k)$ by splitting each of the sequences $x_e(n)$ and $x_o(n)$ into two smaller sequences of length $N/4$ each and computing their respective $\frac{N}{4}-$point DFTs. Specifically, we split $x_e(n)$ into two smaller sequences: one of the sequences contains the even-indexed samples of $x_e(n)$ and the other sequence contains the odd-indexed samples of $x_e(n)$, say,

$$x_{ee}(n) = \Big\{\boxed{x(0)}, x(4), x(8), \ldots, x(N-6), x(N-2)\Big\} \tag{20.27}$$

$$x_{eo}(n) = \Big\{\boxed{x(2)}, x(6), x(10), \ldots, x(N-8), x(N-4)\Big\} \tag{20.28}$$

We then evaluate the $\frac{N}{4}-$point DFTs of $x_{ee}(n)$ and $x_{eo}(n)$, denoted by $X_{ee}(k)$ and $X_{eo}(k)$, respectively, and combine them to obtain the $\frac{N}{2}$-point DFT $X_e(k)$:

$$\boxed{\begin{cases} X_e(k) = X_{ee}(k) + W_{N/2}^k \cdot X_{eo}(k), & 0 \le k \le \frac{N}{4} - 1 \\ X_e\left(k + \frac{N}{4}\right) = X_{ee}(k) - W_{N/2}^k \cdot X_{eo}(k), & 0 \le k \le \frac{N}{4} - 1 \end{cases}} \tag{20.29}$$

Likewise, we split $x_o(n)$ into two smaller sequences: one of the sequences contains the even-indexed samples of $x_o(n)$ and the other sequence contains the odd-indexed samples of $x_o(n)$, say,

$$x_{oe}(n) = \Big\{\boxed{x(1)}, x(5), x(9), \ldots, x(N-5), x(N-1)\Big\} \tag{20.30}$$

$$x_{oo}(n) = \Big\{\boxed{x(3)}, x(7), x(11), \ldots, x(N-7), x(N-3)\Big\} \tag{20.31}$$

We subsequently evaluate the corresponding $\frac{N}{4}$-point DFTs, denoted by $X_{oe}(k)$ and $X_{oo}(k)$, respectively, and combine them to obtain the $\frac{N}{2}-$point DFT $X_o(k)$:

$$\boxed{\begin{cases} X_o(k) &= X_{oe}(k) + W_{N/2}^k \cdot X_{oo}(k), \quad 0 \le k \le \frac{N}{4} - 1 \\ X_o\left(k + \frac{N}{4}\right) &= X_{oe}(k) - W_{N/2}^k \cdot X_{oo}(k), \quad 0 \le k \le \frac{N}{4} - 1 \end{cases}} \tag{20.32}$$

Observe that we now need to evaluate 4 DFTs of order $N/4$ each. The decimation process can be repeated again and applied to each of the sequences $\{x_{ee}(n), x_{eo}(n), x_{oe}(n), x_{oo}(n)\}$. In this way, the $\frac{N}{4}-$point DFTs $\{X_{ee}(k), X_{eo}(k), X_{oe}(k), X_{oo}(k)\}$ would be computed in terms of $\frac{N}{8}-$point DFTs, and so on. Figure 20.3 helps illustrate how the samples of an 8-point sequence, $x(n)$, are decimated into the sequences $x_e(n)$ and $x_o(n)$ and the subsequent sequences $\{x_{ee}(n), x_{eo}(n), x_{oe}(n), x_{oo}(n)\}$ of size 2 samples each.

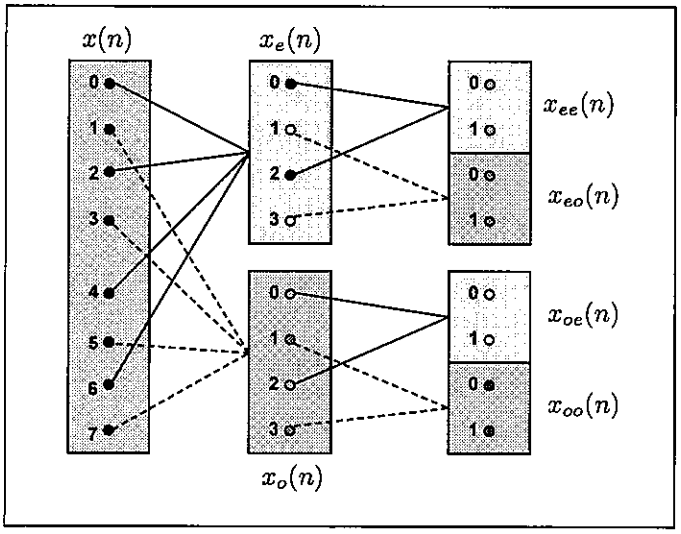

FIGURE 20.3 A representation of the decimation process that takes the samples of an $8-$point sequence $x(n)$ and divides them into smaller sequences $\{x_{ee}(n), x_{eo}(n), x_{oe}(n), x_{oo}(n)\}$ of size 2 samples each. Within each sequence, the even-indexed samples and the odd-indexed samples are denoted by same colored dots. The numbers next to the samples within each box represent the time indices of these samples within that sequence.

More generally, starting from a sequence $x(n)$ with duration $N = 2^p$, the decimation process is repeated $p = \log_2 N$ times until we reach a final stage with sequences of duration 2 samples each; this final stage would require evaluating $2-$point DFTs only. We encountered $2-$point DFTs earlier in Example 17.7, where it was seen that the $2-$point DFT coefficients are obtained by simply adding and subtracting the samples of the time-domain sequence. This construction is best illustrated by means of an example.

Example 20.2 (8-point DFT via decimation-in-time)

Consider the 8-point sequence:

$$x(n) = \left\{\boxed{x(0)}, x(1), x(2), x(3), x(4), x(5), x(6), x(7)\right\} \tag{20.33}$$

By splitting it into even and odd-indexed samples we obtain the two sub-sequences:

$$x_e(n) = \{x(0), x(2), x(4), x(6)\} \tag{20.34}$$
$$x_o(n) = \{x(1), x(3), x(5), x(7)\} \tag{20.35}$$

By further splitting each sub-sequence into even and odd-indexed samples we obtain the four sub-sequences:

$$x_{ee}(n) = \{x(0), x(4)\} \tag{20.36}$$
$$x_{eo}(n) = \{x(2), x(6)\} \tag{20.37}$$
$$x_{oe}(n) = \{x(1), x(5)\} \tag{20.38}$$
$$x_{oo}(n) = \{x(3), x(7)\} \tag{20.39}$$

We are therefore reduced to computing $2-$point DFTs. Using the result of Example 17.7 we have:

$$X_{ee}(0) = x(0) + x(4) \tag{20.40}$$
$$X_{ee}(1) = x(0) - x(4) \tag{20.41}$$

$$X_{eo}(0) = x(2) + x(6) \tag{20.42}$$
$$X_{eo}(1) = x(2) - x(6) \tag{20.43}$$

$$X_{oe}(0) = x(1) + x(5) \tag{20.44}$$
$$X_{oe}(1) = x(1) - x(5) \tag{20.45}$$

$$X_{oo}(0) = x(3) + x(7) \tag{20.46}$$
$$X_{oo}(1) = x(3) - x(7) \tag{20.47}$$

Observe that eight additions are involved in these calculations. We combine the above $2-$point DFTs to obtain the $4-$point DFTs of the sequences $x_e(n)$ and $x_o(n)$ as follows using (20.29) and (20.32):

$$X_e(0) = X_{ee}(0) + X_{eo}(0) \tag{20.48}$$
$$X_e(1) = X_{ee}(1) + W_4^1 \cdot X_{eo}(1) \tag{20.49}$$
$$X_e(2) = X_{ee}(0) - X_{eo}(0) \tag{20.50}$$
$$X_e(3) = X_{ee}(1) - W_4^1 \cdot X_{eo}(1) \tag{20.51}$$

$$X_o(0) = X_{oe}(0) + X_{oe}(1) \tag{20.52}$$
$$X_o(1) = X_{oe}(0) + W_4^1 \cdot X_{oe}(1) \tag{20.53}$$
$$X_o(2) = X_{oe}(0) - X_{oe}(1) \tag{20.54}$$
$$X_o(3) = X_{oe}(0) - W_4^1 \cdot X_{oe}(1) \tag{20.55}$$

Observe that eight complex additions and four complex multiplications are involved in this second set of calculations. Finally, we combine the above $4-$point DFTs to obtain the desired 8-point DFT

of $x(n)$ using (20.26):

$$
\begin{aligned}
X(0) &= X_e(0) + X_o(0) && (20.56)\\
X(1) &= X_e(1) + W_8^1 \cdot X_o(1) && (20.57)\\
X(2) &= X_e(2) + W_8^2 \cdot X_o(2) && (20.58)\\
X(3) &= X_e(2) + W_8^3 \cdot X_o(3) && (20.59)\\
X(4) &= X_e(0) - X_o(0) && (20.60)\\
X(5) &= X_e(1) - W_8^1 \cdot X_o(1) && (20.61)\\
X(6) &= X_e(2) - W_8^2 \cdot X_o(2) && (20.62)\\
X(7) &= X_e(2) - W_8^3 \cdot X_o(3) && (20.63)
\end{aligned}
$$

We note that eight complex additions and six complex multiplications are involved in this stage of the calculations. Note that 3 stages are needed to arrive at the desired DFT, and that $3 = \log_2(8)$. Figure 20.4 shows a flow diagram representation for the above calculations, where the dark circles represent adders.

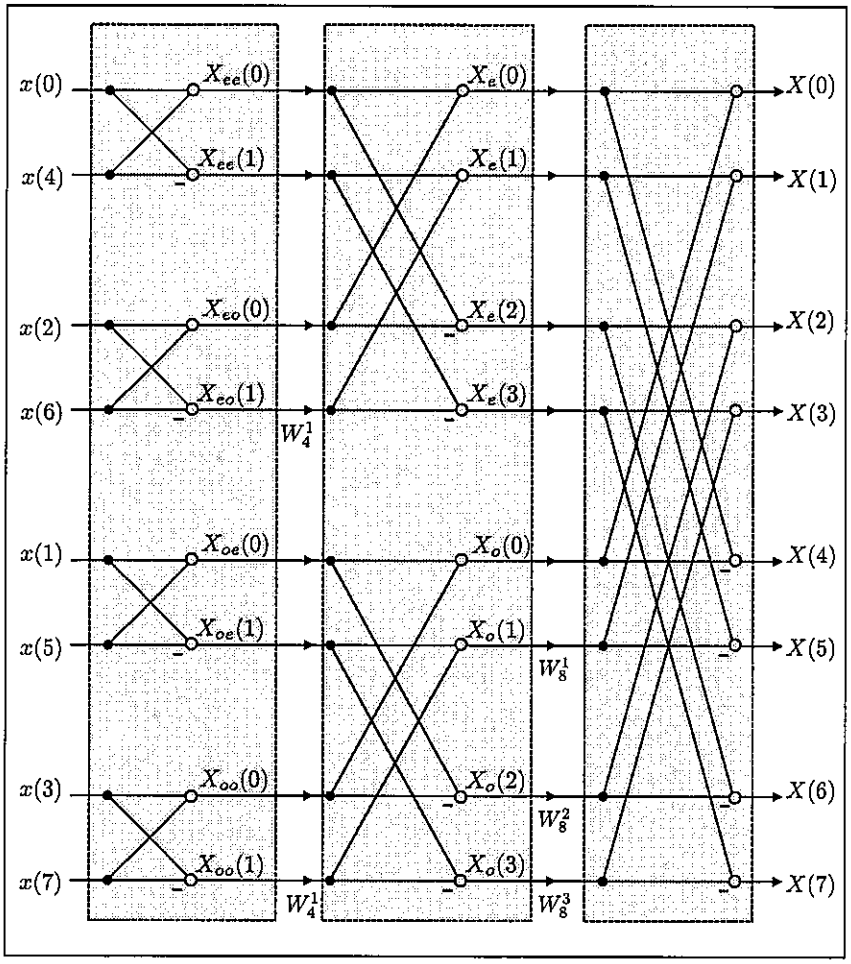

FIGURE 20.4 Flowgraph representation of the 8−point decimation-in-time FFT from Example 20.2. The bright circles in the figure represent adders.

◇

20.2.2 Cost of FFT

Observe from Fig. 20.4 that the basic building block for the implementation of the decimation-in-time FFT is the butterfly section shown separately in Fig. 20.5: its input values are two complex scalars a and b and its output values are two other complex scalars given by

$$A = a + bW \tag{20.64}$$

$$B = a - bW \tag{20.65}$$

where W is a complex scaling coefficient.

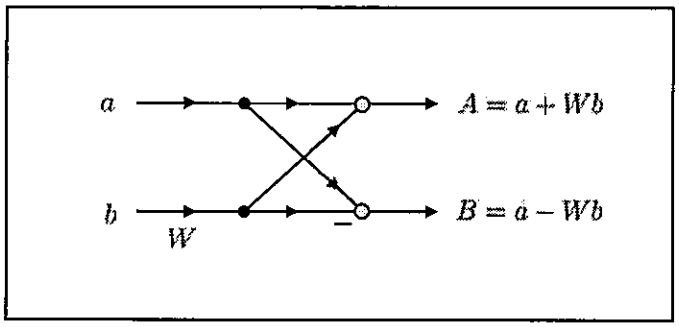

FIGURE 20.5 Elementary butterfly section for decimation-in-time FFT implementations where W is a complex scalar factor.

Each butterfly transformation requires two complex additions and one complex multiplication by W. For simplicity, we combine together the addition and multiplication operations and say that each butterfly transformation requires three complex operations. Using the structure in Fig. 20.4 as a guide, it is straightforward to conclude that, more generally, the implementation of an $N-$point decimation-in-time FFT would require $\log_2 N$ stages (or columns) with $N/2$ butterflies in each stage (or column). Accordingly, the computational complexity of the FFT implementation is roughly

$$3 \times \frac{N}{2} \times \log_2(N) \;=\; 1.5N \log_2(N) \quad \text{complex operations} \tag{20.66}$$

It follows from this analysis that

N-point decimation-in-time FFT requires $O\,(1.5N \log_2 N)$ complex operations (20.67)

In order to illustrate the improvement in computational efficient that results from using the FFT, we use (20.14) and (20.66) to compare in Table 20.1 the computational costs of the FFT and the conventional DFT for several values of N that are powers of 2.

Example 20.3 (Cost of 1024-point FFT)

Let us reconsider the evaluation of the 1024-point DFT from Example 20.1 by using the decimation-in-time FFT implementation. The computational complexity would now be of the order of

$$1.5 \times 1024 \times \log_2(1024) \;=\; 1.5 \times 1024 \times 10 \;=\; 15,360 \;\text{ complex operations} \tag{20.68}$$

Considering again a processor operating at the rate of 3.5GHz and assuming that each complex operation requires one clock cycle, we find that the evaluation of the 1024 DFT coefficients by

TABLE 20.1 Comparison of the computation costs (20.14) and (20.66) for the DFT and FFT implementations in terms of the number of complex operations that are needed.

length N	DFT cost ($2N^2$)	FFT cost ($1.5N \log_2 N$)	speed-up factor
8	128	36	3.6
16	512	96	5.3
32	2,048	240	8.5
64	8,192	576	14.2
128	32,768	1,344	24.4
256	131,072	3,072	42.7
512	524,288	6,912	75.9
1,024	2,097,152	15,360	136.5
2,048	8,388,608	33,792	248.2
4,096	33,554,432	73,728	455.1
8,192	134,217,728	159,744	840.2
32,768	2,147,483,648	737,280	2,912.7
131,072	34,359,738,368	3,342,336	10,280.2
524,288	549,755,813,888	14,942,208	36,792.1
2,097,152	8,796,093,022,208	66,060,288	133,152.5
4,194,304	35,184,372,088,832	138,412,032	254,200.2
8,388,608	140,737,488,355,328	289,406,976	486,296.1
33,554,432	2,251,799,813,685,248	1,258,291,200	1,789,569.7

means of the decimation-in-time FFT procedure would now require about 4.4 μsec; a 136−fold improvement over 0.6 msec.

To appreciate the improvement in computational speed further, assume the samples of $x(n)$ represent the pixels of a 4−megapixel image. That amounts to a total of $N = 4,194,304$ samples. Computing a DFT of this size by means of a 3.5GHz processor would require 2.8 hours, while computing the FFT would require only 0.040 seconds.

◇

20.2.3 Ordering of Samples

We observe from the diagram representation in Fig. 20.4 that the DFT coefficients $X(k)$ appear in their natural order on the right-hand side of the figure, while the samples of the input sequence, $x(n)$, appear shuffled on the left-hand side of the same figure. The order of the input samples can be inferred from the following simple rule. We express the time indices of $x(n)$ in binary format using $p = \log_2 N$ bits, and then reverse the binary representation. For example, for $N = 8$ we obtain the construction shown in Table 20.2.

20.2.4 Evaluating the Inverse DFT

The same decimation-in-time procedure can be used to compute the inverse DFT operation defined by expression (20.5). Comparing the structure of expressions (20.4) and (20.5) for the direct and inverse DFTs, we observe that the only differences are the replacement of the factor W_N^{nk} in (20.4) by its inverse in (20.5) and scaling of the summation by $1/N$. Therefore, the same arguments that were used to derive the decimation-in-time FFT procedure would show that starting from the N−point DFT sequence $X(k)$, the inverse sequence $x(n)$ can be determined by using the same structure shown in Fig. 20.4 for the case $N = 8$ with the following adjustments:

TABLE 20.2 Ordering of the input samples, $x(n)$, for the decimation-in-time FFT.

natural order	binary format	reversed order	after shuffling
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

(a) The coefficients W_N^k in Fig. 20.4 are replaced by their inverse (or complex conjugate) values, W_N^{-k}.

(b) The samples on the left-hand side of the figure would now be the shuffled DFT coefficients $\{X(0), X(4), X(2), X(6), X(1), X(5), X(3), X(7)\}$.

(c) The output signals on the right-hand side of the figure should be scaled by $1/N$, and they would now correspond to the input samples in their natural order, namely, $\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$.

Performing these adjustments leads to the procedure illustrated in Fig. 20.6 for the case $N = 8$.

20.3 DECIMATION-IN-FREQUENCY FFT

We now describe a second efficient implementation of the DFT by relying on decimation in frequency rather than in time. In this variant, the order of the input samples is kept unchanged while the order of the DFT coefficients is shuffled. The algorithm is motivated as follows.

20.3.1 Derivation of Algorithm

Consider again a causal sequence $x(n)$ of duration N. We split $x(n)$ into two sub-sequences, $x_\ell(n)$ and $x_r(n)$, where $x_\ell(n)$ consists of the leading $\frac{N}{2}$ samples of $x(n)$ and $x_r(n)$ consists of the trailing $\frac{N}{2}$ samples of $x(n)$:

$$x_\ell(n) = \left\{ \boxed{x(0)}, x(1), \ldots, x\left(\frac{N}{2} - 1\right) \right\} \tag{20.69}$$

$$x_r(n) = \left\{ \boxed{x\left(\frac{N}{2}\right)}, x\left(\frac{N}{2} + 1\right), \ldots, x(N-1) \right\} \tag{20.70}$$

We can also write

$$\begin{aligned} x_\ell(n) &= x(n), \qquad n = 0, 1, \ldots, \tfrac{N}{2} - 1 \\ x_r(n) &= x\left(n + \tfrac{N}{2}\right), \quad n = 0, 1, \ldots, \tfrac{N}{2} - 1 \end{aligned} \tag{20.71}$$

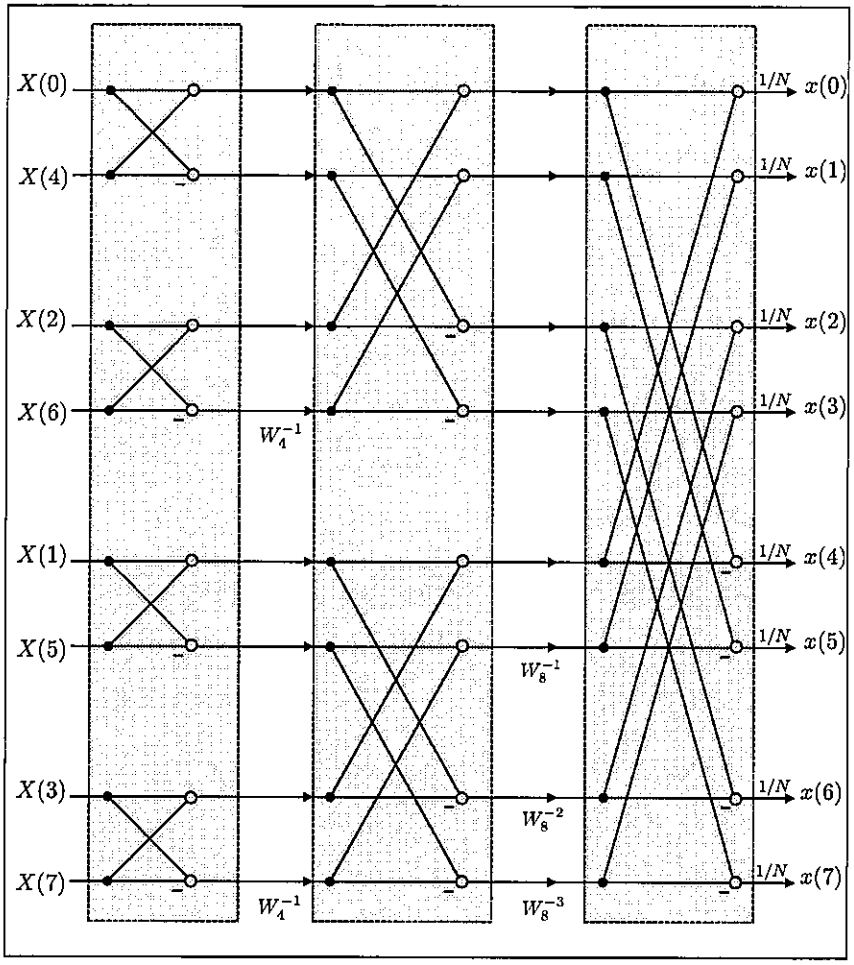

FIGURE 20.6 Flowgraph representation of the 8−point decimation-in-time *inverse* FFT. The dark circles in the figure represent adders.

Figure 20.7 illustrates this construction for a particular sequence $x(n)$ of duration $N = 8$.

We further introduce two sequences $x_1(n)$ and $x_2(n)$, also of duration $N/2$ each, and which are defined in terms of $x_\ell(n)$ and $x_r(n)$ as follows:

$$x_1(n) = x_\ell(n) + x_r(n), \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \tag{20.72}$$

$$x_2(n) = [x_\ell(n) - x_r(n)]W_N^n, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \tag{20.73}$$

That is, $x_1(n)$ is obtained by adding the first $\frac{N}{2}$ samples of $x(n)$ to the trailing $\frac{N}{2}$ samples of $x(n)$. Likewise, $x_2(n)$ is obtained by subtracting the same two sets of samples and scaling the result by the complex factor W_N^n, for each n. Let $X_1(k)$ and $X_2(k)$ denote the $\frac{N}{2}$-point DFTs of the sequences $x_1(n)$ and $x_2(n)$, respectively:

$$X_1(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1(n)W_{N/2}^{kn} \tag{20.74}$$

$$X_2(k) = \sum_{n=0}^{\frac{N}{2}-1} x_2(n)W_{N/2}^{kn} \tag{20.75}$$

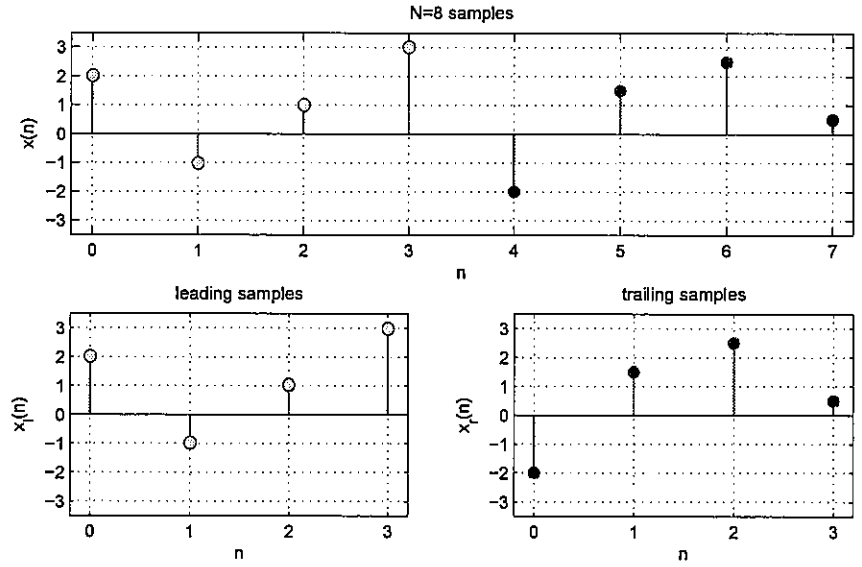

FIGURE 20.7 The sequence $x(n)$ of duration $N = 8$ (top) is decimated into two smaller sequences, $x_\ell(n)$ and $x_r(n)$, of duration 4 samples each (bottom).

We now verify that the desired $N-$point DFT of $x(n)$ can be expressed in terms of the $\frac{N}{2}-$point DFTs, $X_1(k)$ and $X_2(k)$. Indeed, note that:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, \ldots, N-1 \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_\ell(n) W_N^{kn} + W_N^{Nk/2} \cdot \left(\sum_{n=0}^{\frac{N}{2}-1} x_r(n) W_N^{kn} \right) \end{aligned} \tag{20.76}$$

Using the identity $W_N^{Nk/2} = (-1)^k$, we obtain the equivalent expression

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} \left[x_\ell(n) + (-1)^k x_r(n) \right] W_N^{kn}, \quad k = 0, 1, \ldots, N-1 \tag{20.77}$$

We can now split $X(k)$ into even and odd-indexed samples and note that these samples coincide with the DFT coefficients $X_1(k)$ and $X_2(k)$ introduced earlier in (20.74)–(20.75):

$$\begin{aligned} X(2k) &= \sum_{n=0}^{\frac{N}{2}-1} [x_\ell(n) + x_r(n)]\, W_N^{2kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{N/2}^{kn} \\ &\overset{(20.74)}{=} X_1(k), \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{20.78}$$

$$\begin{aligned} X(2k+1) &= \sum_{n=0}^{\frac{N}{2}-1} [x_\ell(n) - x_r(n)]\, W_N^{(2k+1)n} \\ &= \sum_{n=0}^{\frac{N}{2}-1} [x_\ell(n) - x_r(n)]\, W_N^n W_{N/2}^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_{N/2}^{kn} \\ &\overset{(20.75)}{=} X_2(k), \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \end{aligned} \tag{20.79}$$

In summary, we have shown the following so far:

(a) Starting with a sequence $x(n)$, we split it into two sequences of duration $N/2$ each. The first sequence, $x_\ell(n)$, contains the leading $N/2$ entries of $x(n)$ and the second sequence, $x_r(n)$, contains the trailing $N/2$ entries of $x(n)$.

(b) We then add the sequences $x_\ell(n)$ and $x_r(n)$ to generate $x_1(n)$. We also subtract the sequences and multiply each difference by W_N^n to generate the sequence $x_2(n)$.

(c) We then evaluate the $\frac{N}{2}$-point DFTs of $x_1(n)$ and $x_2(n)$. The DFT $X_1(k)$ provides the even-indexed coefficients of the desired $N-$point DFT, $X(k)$, while the DFT $X_2(k)$ provides the odd-indexed coefficients of $X(k)$.

The same decimation procedure can now be applied to the evaluation of $\frac{N}{2}-$point DFTs $X_1(k)$ and $X_2(k)$ by splitting each of the sequences $x_1(n)$ and $x_2(n)$ into two smaller sequences of duration $N/4$ each and computing their respective $\frac{N}{4}-$point DFTs. Specifically, we split $x_1(n)$ into two sequences: one sequence contains the leading $N/4$ entries of $x_1(n)$ and the other sequence contains the trailing $N/4$ entries of $x_1(n)$, say,

$$x_{1\ell}(n) = \left\{ \boxed{x_1(0)}, x_1(1), x_1(2), \ldots, x_1\left(\frac{N}{4} - 1\right) \right\} \tag{20.80}$$

$$x_{1r}(n) = \left\{ \boxed{x_1\left(\tfrac{N}{4}\right)}, x_1\left(\frac{N}{4} + 1\right), x_1\left(\frac{N}{4} + 2\right), \ldots, x_1\left(\frac{N}{2} - 1\right) \right\} \tag{20.81}$$

We then construct the sequences:

$$x_{11}(n) = x_{1\ell}(n) + x_{1r}(n), \qquad n = 0, 1, \ldots, \frac{N}{4} - 1 \tag{20.82}$$

$$\tag{20.83}$$

$$x_{12}(n) = [x_{1\ell}(n) - x_{1r}(n)]\, W_{N/2}^{n}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1 \tag{20.84}$$

and evaluate the $\frac{N}{4}-$point DFTs of $x_{11}(n)$ and $x_{12}(n)$, denoted by $X_{11}(k)$ and $X_{12}(k)$, respectively. The DFT $X_{11}(k)$ provides the even-indexed coefficients of $X_1(k)$ and the DFT $X_{12}(k)$ provides the odd-indexed coefficients of $X_1(k)$:

$$X_1(2k) = X_{11}(k), \quad 0 \leq k \leq \frac{N}{4} - 1 \tag{20.85}$$

$$X_1(2k+1) = X_{12}(k), \quad 0 \leq k \leq \frac{N}{4} - 1 \tag{20.86}$$

Likewise, we split $x_2(n)$ into two smaller sequences: one sequence contains the leading $N/4$ entries of $x_2(n)$ and the other sequence contains the trailing $N/4$ entries of $x_2(n)$, say,

$$x_{2\ell}(n) = \left\{ \boxed{x_2(0)}, x_2(1), x_2(2), \ldots, x_2\left(\frac{N}{4} - 1\right) \right\} \tag{20.87}$$

$$x_{2r}(n) = \left\{ \boxed{x_2\left(\tfrac{N}{4}\right)}, x_2\left(\frac{N}{4} + 1\right), x_2\left(\frac{N}{4} + 2\right), \ldots, x\left(\frac{N}{2} - 1\right) \right\} \tag{20.88}$$

We then construct the sequences:

$$\begin{aligned} x_{21}(n) &= x_{2\ell}(n) + x_{2r}(n), && n = 0, 1, \ldots, \tfrac{N}{4} - 1 \\ x_{22}(n) &= [x_{2\ell}(n) - x_{2r}(n)]\, W_{N/2}^{n}, && n = 0, 1, \ldots, \tfrac{N}{4} - 1 \end{aligned} \tag{20.89}$$

and evaluate the $\frac{N}{4}-$point DFTs of $x_{21}(n)$ and $x_{22}(n)$, denoted by $X_{21}(k)$ and $X22(k)$, respectively. The DFT $X_{21}(k)$ provides the even-indexed coefficients of $X_2(k)$ and the DFT $X_{22}(k)$ provides the odd-indexed coefficients of $X_2(k)$:

$$X_2(2k) = X_{21}(k), \quad 0 \leq k \leq \frac{N}{4} - 1 \tag{20.90}$$

$$X_2(2k+1) = X_{22}(k), \quad 0 \leq k \leq \frac{N}{4} - 1 \tag{20.91}$$

The decimation procedure can be repeated and applied to each of the earlier sequences $\{x_{11}(n), x_{12}(n), x_{21}(n), x_{22}(n)\}$. In this way, their respective $\frac{N}{4}-$point DFTs represented by $\{X_{11}(k), X_{12}(k), X_{21}(k), X_{22}(k)\}$ would instead be computed in terms of smaller $\frac{N}{8}-$point DFTs, and so on. The entire process will again require $\log_2 N$ stages of decimation with computational complexity given roughly by:

$$\boxed{N\text{-point decimation-in-frequency FFT requires } O\left(1.5N \log_2 N\right) \text{ complex operations}} \tag{20.92}$$

The above procedure is best illustrated by means of an example.

Example 20.4 (8-point DFT via decimation-in-frequency)

Consider again an 8-point sequence:

$$x(n) = \left\{\boxed{x(0)}, x(1), x(2), x(3), x(4), x(5), x(6), x(7)\right\} \tag{20.93}$$

We first determine the sub-sequences $\{x_1(n), x_2(n)\}$ of duration 4 samples each:

$$\begin{aligned}
x_1(0) &= x(0) + x(4) && (20.94)\\
x_1(1) &= x(1) + x(5) && (20.95)\\
x_1(2) &= x(2) + x(6) && (20.96)\\
x_1(3) &= x(3) + x(7) && (20.97)
\end{aligned}$$

$$\begin{aligned}
x_2(0) &= x(0) - x(4) && (20.98)\\
x_2(1) &= [x(1) - x(5)] \cdot W_8^1 && (20.99)\\
x_2(2) &= [x(2) - x(6)] \cdot W_8^2 && (20.100)\\
x_2(3) &= [x(3) - x(7)] \cdot W_8^3 && (20.101)
\end{aligned}$$

We then determine the sub-sequences $\{x_{11}(n), x_{12}(n), x_{21}(n), x_{22}(n)\}$ of duration 2 samples each:

$$\begin{aligned}
x_{11}(0) &= x_1(0) + x_1(2) && (20.102)\\
x_{11}(1) &= x_1(1) + x_1(3) && (20.103)
\end{aligned}$$

$$\begin{aligned}
x_{12}(0) &= x_1(0) - x_1(2) && (20.104)\\
x_{12}(1) &= [x_1(1) - x_1(3)] \cdot W_4^1 && (20.105)
\end{aligned}$$

$$\begin{aligned}
x_{21}(0) &= x_2(0) + x_2(2) && (20.106)\\
x_{21}(1) &= x_2(1) + x_2(3) && (20.107)
\end{aligned}$$

$$\begin{aligned}
x_{22}(0) &= x_2(0) - x_2(2) && (20.108)\\
x_{22}(1) &= [x_2(1) - x_2(3)] \cdot W_4^1 && (20.109)
\end{aligned}$$

We evaluate the 2−point DFTs of the sequences $\{x_{11}(n), x_{12}(n), x_{21}(n), x_{22}(n)\}$ by using the result of Example 17.7

$$\begin{aligned}
X_{11}(0) &= x_{11}(0) + x_{11}(1) && (20.110)\\
X_{11}(1) &= x_{11}(0) - x_{11}(1) && (20.111)
\end{aligned}$$

$$\begin{aligned}
X_{12}(0) &= x_{12}(0) + x_{12}(1) && (20.112)\\
X_{12}(1) &= x_{12}(0) - x_{12}(1) && (20.113)
\end{aligned}$$

$$\begin{aligned}
X_{21}(0) &= x_{21}(0) + x_{21}(1) && (20.114)\\
X_{21}(1) &= x_{21}(0) - x_{21}(1) && (20.115)
\end{aligned}$$

$$\begin{aligned}
X_{22}(0) &= x_{22}(0) + x_{22}(1) && (20.116)\\
X_{22}(1) &= x_{22}(0) - x_{22}(1) && (20.117)
\end{aligned}$$

and then map these coefficients into the corresponding coefficients $X(k)$:

$$X(0) = X_{11}(0) \tag{20.118}$$
$$X(4) = X_{11}(1) \tag{20.119}$$

$$X(2) = X_{12}(0) \tag{20.120}$$
$$X(6) = X_{12}(1) \tag{20.121}$$

$$X(1) = X_{21}(0) \tag{20.122}$$
$$X(5) = X_{21}(1) \tag{20.123}$$

$$X(3) = X_{22}(0) \tag{20.124}$$
$$X(7) = X_{22}(1) \tag{20.125}$$

Figure 20.8 shows a flow diagram representation for the above calculations, where the dark circles represent adders.

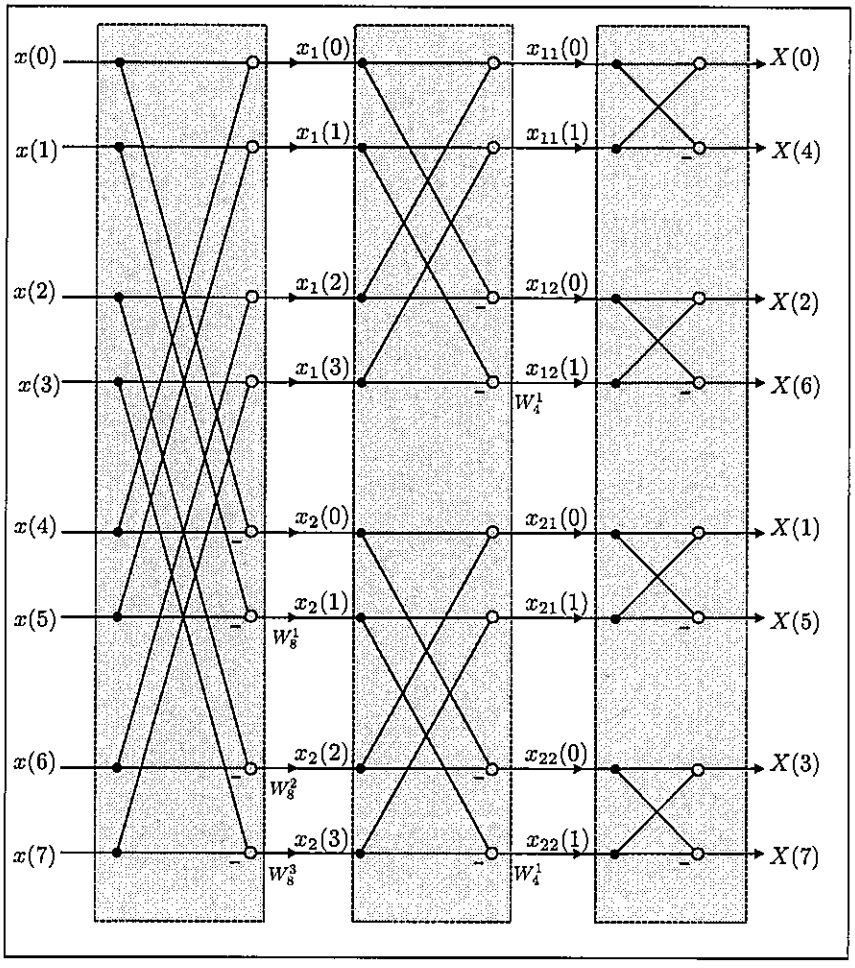

FIGURE 20.8 Flowgraph representation of the 8−point decimation-in-frequency FFT. The dark circles in the figure represent adders.

20.3.2 Cost of FFT

We observe from Fig. 20.8 that the basic building block for the implementation of decimation-in-frequency FFT is again a butterfly section; albeit one with the transformation shown in Fig. 20.9: its input values are two complex scalars a and b and its output values are again two complex scalars given by:

$$A = a + b \tag{20.126}$$

$$B = (a - b)W \tag{20.127}$$

where W is a complex scaling coefficient. Each butterfly transformation requires two complex additions and one complex multiplication, so we say that it requires roughly 3 complex operations. Guided by Fig. 20.8 we observe further that, more generally, the implementation of an $N-$point decimation-in-frequency FFT requires $\log_2 N$ stages (or columns) with $N/2$ butterflies in each stage (or column). Accordingly, the computational complexity of this FFT implementation is again

N-point decimation-in-frequency FFT requires $O\left(1.5N \log_2 N\right)$ complex operations

(20.128)

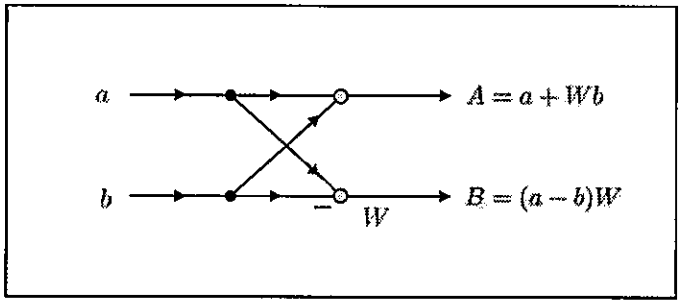

FIGURE 20.9 Elementary butterfly section for decimation-in-frequency FFT implementations where W is a complex scalar factor. The bright circles denote adders.

20.3.3 Evaluating the Inverse DFT

The same decimation-in-frequency procedure can be used to compute the inverse DFT operation defined by expression (20.5) with the following adjustments to the structure shown in Fig. 20.8:

(a) The coefficients W_N^k in Fig. 20.8 are replaced by their inverses (or complex conjugate) values, W_N^{-k}.

(b) The samples on the left-hand side of the figure would be the DFT coefficients in their natural order, $\{X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7)\}$.

(c) The output signals on the right-hand side of the figure should be scaled by $1/N$ and they would then coincide with the input samples appearing in a reshuffled order, namely, $\{x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)\}$.

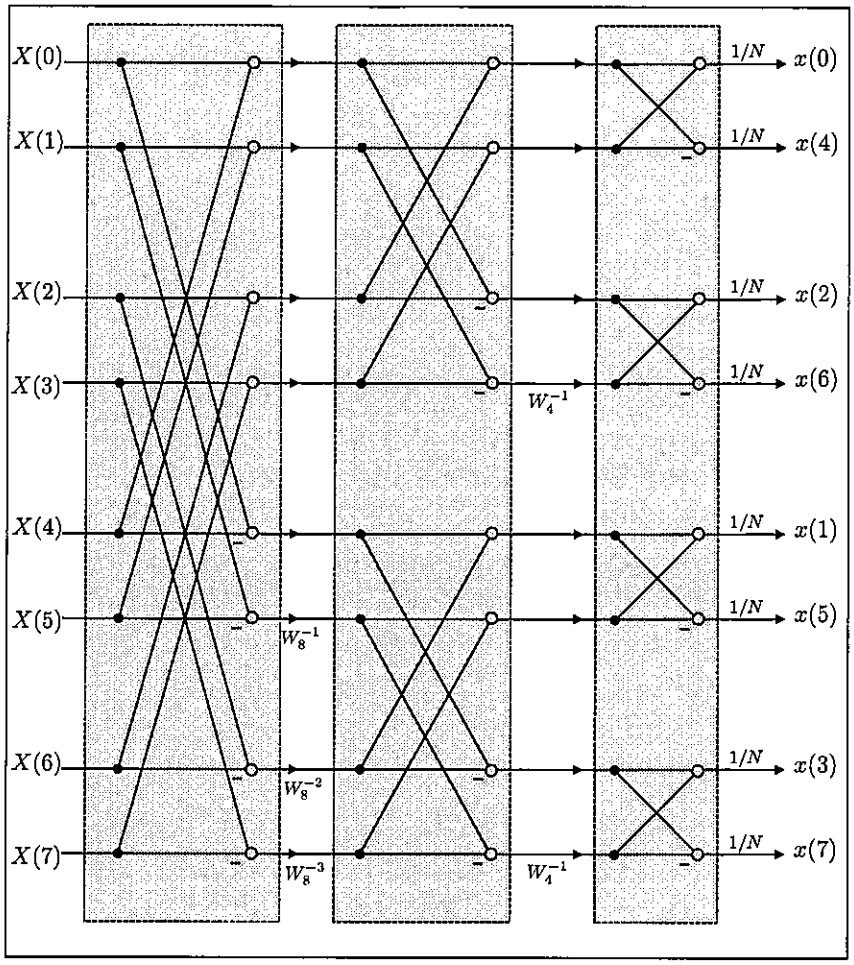

FIGURE 20.10 Flowgraph representation of the 8−point decimation-in-frequency inverse FFT. The dark circles in the figure represent adders.

Performing these adjustments leads to Fig. 20.10 for the case $N = 8$.

20.3.4 Relation between Decimation in Time and Frequency

By comparing the flowgraphs of Figs. 20.4 and 20.10 for the *direct* decimation-in-time FFT and the *inverse* decimation-in-frequency FFT, we observe a clear relation between both diagrams. If we start with the flowgraph of a *direct* decimation-in-time FFT and perform the following sequence of adjustments, we arrive at the implementation for the *inverse* decimation-in-frequency FFT:

(a) We invert the direction of flow and exchange the input and output nodes.

(b) We invert the phase factors, W_N^k, and replace them by W_N^{-k}.

(c) We scale the output signals by $1/N$.

Likewise, by comparing the flowgraphs of Figs. 20.6 and 20.8 for the *inverse* decimation-in-time FFT and the *direct* decimation-in-frequency FFT, we observe a similar relation between both diagrams. If we start with the flowgraph of a *direct* decimation-in-frequency FFT and perform the same sequence of operations as above, we arrive at the implementa-

tion for the *inverse* decimation-in-time FFT.

20.4 APPLICATION: CELESTIAL ORBITS AND ASTRONOMY

In this section we consider the problem of fitting astronomical data into trigonometric polynomials. The application is meant to shed some light on the historical origin of the FFT procedure.

The Fast Fourier Transform can be traced back to unpublished notes by the German mathematician **Carl F. Gauss (1777-1855)**. Around 1805, Gauss became interested in the problem of fitting observed astronomical data into a trigonometric polynomial. In much the same way as we motivated the FFT procedure in the previous sections, Gauss employed a divide-and-conquer strategy to reduce the original problem into smaller problems, which could then be combined to yield the desired solution. Gauss' motivation was computational efficiency. Unfortunately, Gauss did not publish his 1805 notes and his contribution remained largely unknown. Almost 160 years later, the mathematical foundations for the FFT procedure were re-discovered by Cooley and Tukey in 1965. Their contribution led to an explosion of interest in the FFT and to the application of discrete-time processing techniques in a wide range of products in consumer and military electronics.

Celestial Coordinates

The location of a celestial object in the sky can be described in terms of its ascension and declination coordinates, which are defined as follows. The celestial equator is taken to be the projection of the Earth's equator onto the sky — see Fig. 20.11. The Earth orbits the Sun once every 365 days. Viewed from the Earth, the Sun appears to move along an imaginary trajectory called the ecliptic. For half of the time, the Sun is above the celestial equator and for the other half it is below the celestial equator. The point at which the ecliptic trajectory crosses the celestial equator from South to North is called the Vernal Equinox (the other point is called the Autumnal Equinox). The ascension coordinate of a celestial body is an angle that measures how far the object is to the east or west of the Vernal Equinox. Similarly, the declination coordinate of the celestial body is an angle that measures how far the object is to the north or south of the celestial equator. Since the sky appears to turn around 360 degrees every 24 hours, then every 15 degrees of ascension correspond to a lapse of one hour duration.

Data Fitting

Now assume that, in a manner similar to what Gauss did back in the early 1800s, we collect the declination measures of a body in the sky every two hours over a period of one day. This means that we measure the object's declination at ascension intervals of 30 degrees. We denote the twelve ascension sample values (or "instants") by:

$$\theta_n = 0^o,\ 30^o,\ 60^o,\ 90^o,\ 120^o,\ 150^o,\ 180^o,\ 210^o,\ 240^o,\ 270^o,\ 300^o,\ 330^o \quad (20.129)$$

or, equivalently, in radians,

$$\theta_n = \frac{2\pi n}{N}, \quad n = 0, 1, 2, \ldots, N-1 \quad \text{(in radians)} \qquad (20.130)$$

where $N = 12$. We denote the declination measure corresponding to θ_n by $x_p(n)$. These measurements exhibit periodicity and repeat themselves every $N = 12$ samples. For this

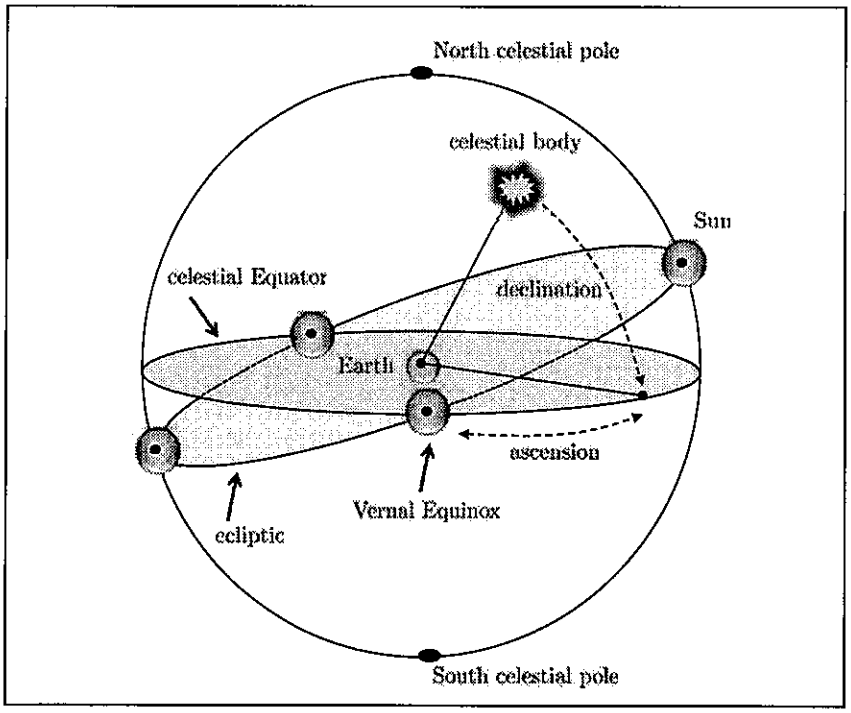

FIGURE 20.11 Ascension and declination coordinates of celestial bodies in a celestial coordinate system with the Earth located at the center of the celestial equator.

reason, it is common in astronomical studies to attempt to fit the measured declinations $\{x_p(n)\}$ into a trigonometric polynomial, say, of the form:

$$x_p(n) \approx \frac{A(0)}{2} + \sum_{k=1}^{N-1} A(k) \cos\left(\frac{2\pi kn}{N}\right) + \sum_{k=1}^{N-1} B(k) \sin\left(\frac{2\pi kn}{N}\right) \tag{20.131}$$

Such (Fourier series) representations are useful because, once determined, they can be used to estimate (or predict) the location of the celestial object for time instants (i.e., ascensions) for which measurements are not available.

We explained earlier in Sec. 17.6 how the real coefficients $\{A(k), B(k)\}$ can be determined through expressions (17.126)–(17.128), namely,

$$A(0) = \frac{2}{N} \sum_{n=0}^{N-1} x_p(n) \tag{20.132}$$

$$A(k) = \frac{1}{N} \sum_{n=0}^{N-1} x_p(n) \cos\left(\frac{2\pi kn}{N}\right), \; k = 1, 2, \ldots, N-1 \tag{20.133}$$

$$B(k) = \frac{1}{N} \sum_{n=0}^{N-1} x_p(n) \sin\left(\frac{2\pi kn}{N}\right), \; k = 1, 2, \ldots, N-1 \tag{20.134}$$

We also explained in that section that the problem of determining these coefficients is equivalent to the problem of determining the $N-$point DFT coefficients corresponding the samples of $x_p(n)$ over the period $0 \leq n \leq N-1$:

$$X(k) = \sum_{n=0}^{N-1} x_p(n) e^{-j\frac{2\pi k}{N}n}, \quad k = 0, 1, \ldots, N-1 \tag{20.135}$$

with the relation between the coefficients $\{X(k)\}$ and $\{A(k), B(k)\}$ given by

$$\boxed{\begin{cases} \frac{1}{N}X(0) \triangleq \frac{1}{2}A(0) \\ \\ \frac{1}{N}X(k) \triangleq A(k) - jB(k) \end{cases}} \tag{20.136}$$

Gauss devised a divide-and-conquer strategy to determine the coefficients $\{A(k), B(k)\}$ $(k = 0, 1, \ldots, N-1)$ by splitting the problem into two smaller problems of size $N/2$ samples each, and then splitting each of these into smaller problems of size $N/4$ samples each, and so forth, in a manner that is similar to the construction that we employed in this chapter to arrive at the FFT procedure for computing the $\{X(k)\}$.

Practice Questions:

1. Start from (20.131) and use trigonometric identities to establish the validity of expressions (20.132)– (20.134) for the coefficients $\{A(k), B(k)\}$.
2. Assume N is a power of 2 and split the sequence $x(n)$ into two smaller sequences of duration $N/2$ each. In one sequence we group the even-indexed samples of $x(n)$ and in the other sequence we group the odd-indexed samples of $x(n)$, as we did earlier in (20.17)–(20.18). Let $\{A_e(k), B_e(k)\}$ and $\{A_o(k), B_o(k)\}$ denote the $\frac{N}{2}$−point Fourier series coefficients of $x_e(n)$ and $x_o(n)$, respectively. Express the coefficients $\{A(k), B(k)\}$ in terms of the coefficients $\{A_e(k), B_e(k), A_o(k), B_o(k)\}$.

$\diamond$

20.5 REFERENCES AND COMMENTARIES

The modern version of the Fast Fourier Transform was derived by Cooley and Tukey (1965) in an influential short article that has since then revolutionized the field of discrete-time signal processing. The FFT is nowadays the preferred method for implementing direct and inverse DFT computations, so much so that FFT processors have become common components in many DSP devices in modern electronics.

As already hinted in Sec. 20.4, it turns out that the FFT has a much longer and richer history. It is now widely recognized that the same algorithm by Cooley and Tukey (1965) was actually proposed 160 years earlier in 1805 in unpublished notes by the prodigious German mathematician **Carl F. Gauss (1777-1855)**; the notes appeared later in the collected works Gauss (1866). As we explained in Sec. 20.4, around 1805, Gauss became interested in the problem of fitting astronomical data into trigonometric polynomials and in determining the Fourier series coefficients in a computationally efficient manner. We already commented earlier in the concluding remarks of Chapter 17 on Gauss' contribution to the development of the DFT itself and on how his work on trigonometric interpolation problems of the form (17.152), involving sine and cosine factors, led to expressions (17.166)–(17.168) for computing the interpolating coefficients. His expressions correspond to an equivalent representation for the expression we use nowadays to evaluate the (complex-valued) DFT coefficients $\{X(k)\}$ of a sequence $\{x(n)\}$. Gauss went further in his studies on celestial mechanics. He employed a divide-and-conquer strategy to reduce the problem of computing the Fourier series coefficients into problems of smaller size, which could then be combined to yield the desired solution. Gauss' method turns out to be equivalent to the decimation-in-frequency FFT method that we described in Sec. 20.3. Unfortunately, Gauss did not publish his 1805 notes and his contribution remained largely unknown until the rediscovery of the FFT procedure by Colley and Tukey almost 160 years later. An insightful historical account on the FFT and its origin appears in the article by Heideman, Johnson, and Burrus (1984). References to the earlier 1805 contribution of

Gauss appear in the works by Burkhardt (1904) and Goldstine (1977). It is noteworthy to remark that Gauss' 1805 efficient algorithm for computing the Fourier series coefficients (or, equivalently, the DFT coefficients) predates the introduction of the Fourier series formalism in 1807 by the French mathematician **Jean-Baptiste Joseph Fourier (1768–1830)**.

We limited our discussion in the chapter to a widely used class of FFT algorithms, namely, the class of radix-2 implementations. In these implementations, the length N of the sequence is assumed to be a power of 2, and the sizes of the subsequent sequences are successively halved. There are many other variations of fast Fourier transforms. For example, the duration N can instead be factored as a product of the form $N = N_1 N_2$, and the original sequence of duration N can then be subdivided into N_1 sequences of size N_2 each. The DFT coefficients of these shorter sequences can be combined to arrive at the DFT coefficients of the original sequence. Actually, the algorithm derived by Gauss in 1805 and by Cooley and Tukey (1965) was of this more general form. When $N_1 = 2$, we recover the radix-2 implementations discussed in the body of the chapter.

There is another FFT implementation when N_1 and N_2 are relatively prime numbers; this fact can be exploited to arrive at an algorithm that does not require the use of the twiddle factor, W_N. This latter implementation is known as the Prime Factor Algorithm (PFA), and it was proposed by Good (1958) a few years before Cooley and Tukey's 1965 work. Another FFT implementation is Rader's algorithm for prime lengths N; the algorithm exploits the fact that N is prime to express the DFT as a cyclic convolution of two sequences. The convolution can then be evaluated, say, by conventional FFT implementations. This algorithm was proposed in Rader (1968) and it was later extended by Winograd (1976,1978) to lengths N that are powers of a prime factor, say, $N = P^\ell$, for some prime number P. A related algorithm is Bluestein's algorithm. We explained in the concluding remarks of Chapter 18 that Bluestein (1968) expressed the DFT in a convolution form as well, as shown by expressions (18.247)–(18.250); these expressions are valid for any length N (they do not require N to be prime). The convolution can then be evaluated by conventional FFT implementations. For further information on the FFT and its variations, the reader may refer to several references on the topic including Brigham (1988), Chu and George (1999), Duhamel and Vetterli (1990), Mitra (2005), Rao, Kim, and Hwang (2010), Oppenheim, Schafer, and Buck (2009), Proakis and Manolakis (2006), Smith (2007), Walker (1996), and Zonst (2003).

20.6 PROBLEMS

P 20.1 Let $x(n) = \{\boxed{1+j},\ -1,\ e^{j\frac{\pi}{3}},\ 2\}$.

(a) Determine the 4−point DFT of $x(n)$ using the definition (20.1).

(b) How many real additions and real multiplications are needed to evaluate the DFT of part (a)? Count additions and multiplications separately.

(c) Draw a decimation-in-time FFT structure to evaluate $X(k)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

(d) Draw a decimation-in-frequency FFT structure to evaluate $X(k)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

P 20.2 Let $x(n) = \{\boxed{-1},\ 0,\ 1-j,\ e^{-j\frac{\pi}{2}},\ 2\}$.

(a) Determine the 8−point DFT of $x(n)$ using the definition (20.1).

(b) How many real additions and real multiplications are needed to evaluate the DFT of part (a)? Count additions and multiplications separately.

(c) Draw a decimation-in-time FFT structure to evaluate $X(k)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

(d) Draw a decimation-in-frequency FFT structure to evaluate $X(k)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

P 20.3 Let $X(k) = \{\boxed{1+j},\ -1,\ 1,\ 1-j\}$.

(a) Determine the inverse 4−point DFT $x(n)$ using the definition (20.2).

(b) How many real additions and real multiplications are needed to evaluate the DFT of part (a)? Count additions and multiplications separately.

(c) Draw an inverse decimation-in-time FFT structure to evaluate $x(n)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

(d) Draw an inverse decimation-in-frequency FFT structure to evaluate the $x(n)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

P 20.4 Let $X(k) = \{\boxed{-1},\ j,\ -j, 0,\ 1\}$.

(a) Determine the inverse 8−point DFT $x(n)$ using the definition (20.2).

(b) How many real additions and real multiplications are needed to evaluate the DFT of part (a)? Count additions and multiplications separately.

(c) Draw an inverse decimation-in-time FFT structure to evaluate $x(n)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

(d) Draw an inverse decimation-in-frequency FFT structure to evaluate $x(n)$. How many real additions and real multiplications are needed to evaluate the DFT coefficients in this structure?

P 20.5 Consider the two sequences

$$x_1(n) = \delta(n) + \frac{1}{2}\delta(n-1) \quad \text{and} \quad x_2(n) = \frac{1}{2}\delta(n) + \delta(n-2)$$

Explain how you would evaluate the linear convolution of $x_1(n)$ and $x_2(n)$ by employing a decimation-in-time FFT. Draw a complete diagram with all necessary weights and input and output signals clearly indicated. The output nodes of your diagram should be the samples of the linear convolution sequence.

P 20.6 The samples of a finite-duration sequence $h(n)$ are nonzero only over $0 \leq n \leq 2$ and they are equal to the samples of the periodic sequence

$$h_p(n) = \sum_{k=-\infty}^{\infty} \left(\frac{1}{2}\right)^{n-5k} u(n-5k)$$

over the same interval of time.

(a) Determine the samples of $h(n)$.

(b) Draw the diagram of the smallest radix-2 decimation-in-time FFT that computes a DFT of $h(n)$.

(c) Draw a radix-2 decimation-in-frequency FFT that computes the 4−point DFT of the sequence

$$x(n) = \delta(n) - \delta(n-2)$$

(d) Explain how you would combine the FFT diagrams of parts (b) and (c) in order to obtain at the output nodes the values of the circular convolution

$$y(n) = \{\boxed{1}, 0, -1, 0\} \circ \{h(n), 0\}$$

Show all connections. Determine also the sequence $y(n)$.

P 20.7 Consider the problem of finding the circular convolution of two complex sequences, each having 2^m entries.

(a) How many complex multiplications and additions are required to directly convolve the sequences without using any DFTs?

(b) Assuming that a real addition and a real multiplication take about the same computation time, for what values of m, less computation time will be required to perform the convolution as

above, rather than computing the FFT of each sequence, multiplying the DFT coefficients, and performing an inverse FFT?

P 20.8 Let $x(n)$ and $h(n)$ denote two real-valued causal sequences of duration 1024 and 128 samples, respectively. Let $y(n)$ denote the linear convolution of $x(n)$ and $y(n)$.

(a) How many real additions and multiplications are required to evaluate the samples of $y(n)$ directly from the definition of the linear convolution.

(b) How many real additions and multiplications are required to evaluate the samples of $y(n)$ by using the overlap-add method of Sec. 19.3, where the required smaller linear convolutions are computed again from the definition.

(c) How many real additions and multiplications are required to evaluate the samples of $y(n)$ by using the overlap-save method of Sec. 19.3 where the required circular convolutions are computed directly from the definition.

(d) How many real additions and multiplications are required to evaluate the samples of $y(n)$ by using the overlap-save method of Sec. 19.3 where the required circular convolutions are computed by means of decimation-in-time FFTs.

P 20.9 Express the DFT of the $9-$point sequence $\{x(0), x(1), \ldots, x(8)\}$ in terms of the DFT's of the $3-$point sequences:

$$\{x(0), x(3), x(6)\}, \quad \{x(1), x(4), x(7)\}, \quad \{x(2), x(5), x(8)\}$$

Draw a flowgraph diagram for this method of computing a $9-$point DFT from $3-$point DFTs.

P 20.10 Let $x(n)$ be a causal sequence of duration $2N$, i.e., its nonzero samples occur over $0 \leq n \leq 2N-1$. How would you use an $N-$point decimation-in-time FFT algorithm to evaluate the $2N-$point DFT of $x(n)$?

P 20.11 A sequence $x(n)$ has nonzero samples over the interval $\frac{N}{2} \leq n \leq N-1$. Explain how you would evaluate the $N-$point DFT of $x(n)$ by using an $\frac{N}{2}-$point FFT algorithm?

P 20.12 A sequence $x(n)$ has duration N and its odd-indexed samples are all zero. How would you evaluate the $N-$point DFT of $x(n)$ by using an $\frac{N}{2}-$point FFT algorithm?

P 20.13 Consider an arbitrary signal $x(n)$ defined in the interval $0 \leq n \leq N-1$, and its $N-$point DFT coefficients $X(k)$, $0 \leq k \leq N-1$.

(a) Draw a diagram of a 4-point decimation-in-time FFT. Your diagram should use multipliers and adders only, and should clearly indicate how the samples of $x(n)$ are transformed into the values of $X(k)$.

(b) Show that by appropriately reordering the samples $X(k)$, and by calculating the DFT of the result, you can obtain the signal $Nx(n)$. In other words, show that

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

where $X'(k)$ is a reordered (shuffled) version of $X(k)$. Find the reordered sequence.

(c) Use the results of parts (a) and (b) to construct a 4-point inverse DFT block that uses a shuffler, followed by the FFT block of part (a), followed by scaling multipliers. Draw a diagram of the shuffler block.