Clearly, $x_e(n)$ satisfies the symmetry condition (2.1.24). Similarly, we form an odd signal component $x_o(n)$ according to the relation

$$x_o(n) = \tfrac{1}{2}[x(n) - x(-n)] \tag{2.1.27}$$

Again, it is clear that $x_o(n)$ satisfies (2.1.25); hence it is indeed odd. Now, if we add the two signal components, defined by (2.1.26) and (2.1.27), we obtain $x(n)$, that is,

$$x(n) = x_e(n) + x_o(n) \tag{2.1.28}$$

Thus any arbitrary signal can be expressed as in (2.1.28).

### 2.1.3 Simple Manipulations of Discrete-Time Signals

In this section we consider some simple modifications or manipulations involving the independent variable and the signal amplitude (dependent variable).

**Transformation of the independent variable (time).** A signal $x(n)$ may be shifted in time by replacing the independent variable $n$ by $n - k$, where $k$ is an integer. If $k$ is a positive integer, the time shift results in a delay of the signal by $k$ units of time. If $k$ is a negative integer, the time shift results in an advance of the signal by $|k|$ units in time.

1.Shifting Operation (Delay/Advance)
2.Folding or Reflection
3.Time scaling or Down-sampling

**Example 2.1.2**

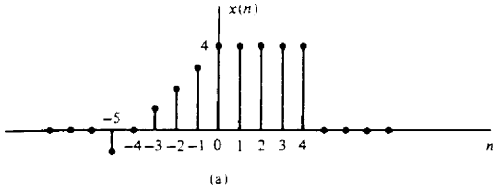
A signal $x(n)$ is graphically illustrated in Fig. 2.9a. Show a graphical representation of the signals $x(n-3)$ and $x(n+2)$.

**Solution** The signal $x(n-3)$ is obtained by delaying $x(n)$ by three units in time. The result is illustrated in Fig. 2.9b. On the other hand, the signal $x(n+2)$ is obtained by advancing $x(n)$ by two units in time. The result is illustrated in Fig. 2.9c. Note that delay corresponds to shifting a signal to the right, whereas advance implies shifting the signal to the left on the time axis.

If the signal $x(n)$ is stored on magnetic tape or on a disk or, perhaps, in the memory of a computer, it is a relatively simple operation to modify the base by introducing a delay or an advance. On the other hand, if the signal is not stored but is being generated by some physical phenomenon in real time, it is not possible to advance the signal in time, since such an operation involves signal samples that have not yet been generated. Whereas it is always possible to insert a delay into signal samples that have already been generated, it is physically impossible to view the future signal samples. Consequently, in real-time signal processing applications, the operation of advancing the time base of the signal is physically unrealizable.

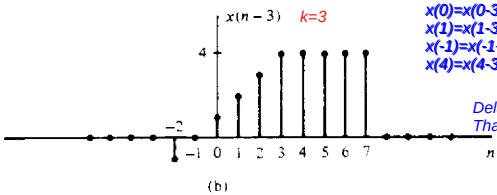
Another useful modification of the time base is to replace the independent variable $n$ by $-n$. The result of this operation is a *folding* or a *reflection* of the signal about the time origin $n = 0$.

A=4,B=2
A-B=/B-A




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

n=-1,x(-1)=3
n=-2,x(-2)=2
n=-3,x(-3)=1
n=-4,x(-4)=0
n=-5,x(-5)=-1




x(0)=x(0-3)=x(-3)=1
x(1)=x(1-3)=x(-2)=2
x(-1)=x(-1-3)=x(-4)=0
x(4)=x(4-3)=x(1)=4

X(n)={-1,0,1,2,3,4,4,4,4,4}

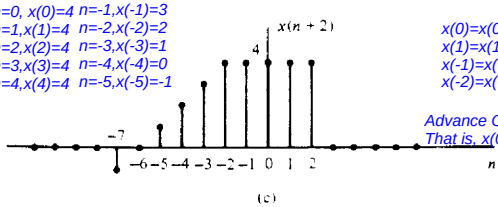
x(n-3)={-1,0,1,2,3,4,4,4,4,4}

Delay Operation. we get previous value.x(n-3) value on x(n).
That is, x(0)=x(-3)

k=-2

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

n=-1,x(-1)=3
n=-2,x(-2)=2
n=-3,x(-3)=1
n=-4,x(-4)=0
n=-5,x(-5)=-1




x(0)=x(0+2)=x(2)=4
x(1)=x(1+2)=x(3)=4
x(-1)=x(-1+2)=x(1)=4
x(-2)=x(-2+2)=x(0)=4

X(n)={-1,0,1,2,3,4,4,4,4,4}

x(n+2)={-1,0,1,2,3,4,4,4,4,4}

advance

Advance Operation. we get ~~previous~~ value.x(n+2) value on x(n).
That is, x(0)=x(2)

**Figure 2.9** Graphical representation of a signal, and its delayed and advanced versions.

**Example 2.1.3**

Show the graphical representation of the signal $x(-n)$ and $x(-n+2)$, where $x(n)$ is the signal illustrated in Fig. 2.10a.

**Solution** The new signal $y(n) = x(-n)$ is shown in Fig. 2.10b. Note that $y(0) = x(0)$, $y(1) = x(-1)$, $y(2) = x(-2)$, and so on. Also, $y(-1) = x(1)$, $y(-2) = x(2)$, and so on. Therefore, $y(n)$ is simply $x(n)$ reflected or folded about the time origin $n = 0$. The signal $y(n) = x(-n+2)$ is simply $x(-n)$ delayed by two units in time. The resulting signal is illustrated in Fig. 2.10c. A simple way to verify that the result in Fig. 2.10c is correct is to compute samples, such as $y(0) = x(2)$, $y(1) = x(1)$, $y(2) = x(0)$, $y(-1) = x(3)$, and so on.

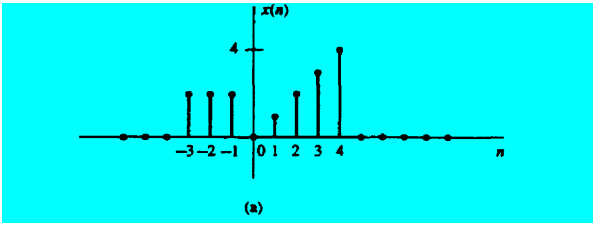
It is important to note that the operations of folding and time delaying (or advancing) a signal are not commutative. If we denote the time-delay operation by TD and the folding operation by FD, we can write

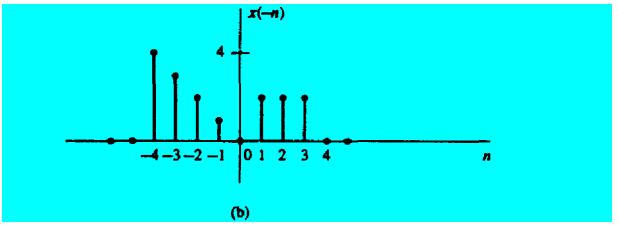
$$\begin{aligned} \mathrm{TD}_k[x(n)] &= x(n-k) \qquad k > 0 \\ \mathrm{FD}[x(n)] &= x(-n) \end{aligned} \tag{2.1.29}$$

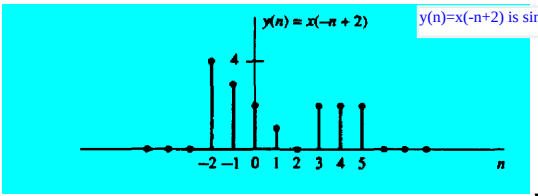
Now

=x(-(n-k))

$$\mathrm{TD}_k\{\mathrm{FD}[x(n)]\} = \mathrm{TD}_k[x(-n)] = x(-n+k) \tag{2.1.30}$$




y(n)=x(-n+2) is simply x(-n) is delayed by two units in time.

**Figure 2.10** Graphical illustration of the folding and shifting operations.

whereas

Only n is replaced by -n.

$$\text{FD}[\text{TD}_k[x(n)]] = \text{FD}[x(n-k)] = x(-n-k) \tag{2.1.31}$$

Note that because the signs of $n$ and $k$ in $x(n-k)$ and $x(-n+k)$ are different, the result is a shift of the signals $x(n)$ and $x(-n)$ to the right by $k$ samples, corresponding to a time delay.

A third modification of the independent variable involves replacing $n$ by $\mu n$, where $\mu$ is an integer. We refer to this time-base modification as *time scaling* or *down-sampling*.

**Example 2.1.4**

Show the graphical representation of the signal $y(n) = x(2n)$, where $x(n)$ is the signal illustrated in Fig. 2.11a.

**Solution** We note that the signal $y(n)$ is obtained from $x(n)$ by taking every other sample from $x(n)$, starting with $x(0)$. Thus $y(0) = x(0)$, $y(1) = x(2)$, $y(2) = x(4), \ldots$ and $y(-1) = x(-2)$, $y(-2) = x(-4)$, and so on. In other words, we have skipped

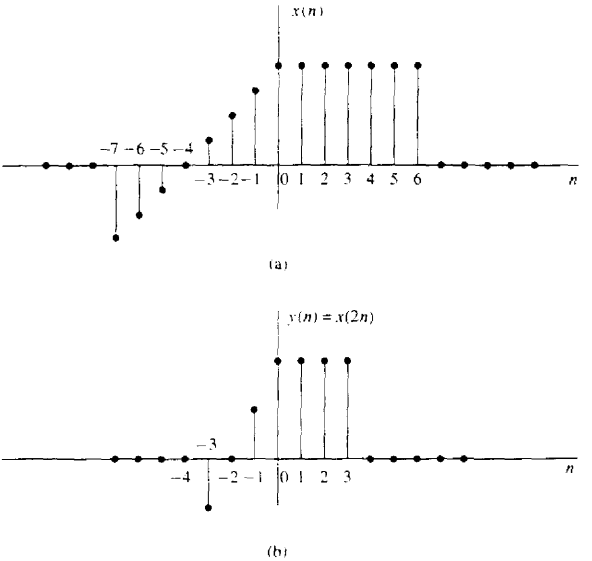



**Figure 2.11** Graphical illustration of down-sampling operation.

the odd-numbered samples in $x(n)$ and retained the even-numbered samples. The resulting signal is illustrated in Fig. 2.11b.

If the signal $x(n)$ was originally obtained by sampling an analog signal $x_a(t)$, then $x(n) = x_a(nT)$, where $T$ is the sampling interval. Now, $y(n) = x(2n) = x_a(2Tn)$. Hence the time-scaling operation described in Example 2.1.4 is equivalent to changing the sampling rate from $1/T$ to $1/2T$, that is, to decreasing the rate by a factor of 2. This is a *downsampling* operation.

**Addition, multiplication, and scaling of sequences.** Amplitude modifications include *addition*, *multiplication*, and *scaling* of discrete-time signals.

*Amplitude scaling* of a signal by a constant $A$ is accomplished by multiplying the value of every signal sample by $A$. Consequently, we obtain

$$y(n) = Ax(n) \qquad -\infty < n < \infty$$

The *sum* of two signals $x_1(n)$ and $x_2(n)$ is a signal $y(n)$, whose value at any instant is equal to the sum of the values of these two signals at that instant, that is,

$$y(n) = x_1(n) + x_2(n) \qquad -\infty < n < \infty$$

The *product* of two signals is similarly defined on a sample-to-sample basis as

$$y(n) = x_1(n)x_2(n) \qquad -\infty < n < \infty$$

## 2.2 DISCRETE-TIME SYSTEMS

In many applications of digital signal processing we wish to design a device or an algorithm that performs some prescribed operation on a discrete-time signal. Such a device or algorithm is called a discrete-time system. More specifically, a *discrete-time system* is a device or algorithm that operates on a discrete-time signal, called the *input* or *excitation*, according to some well-defined rule, to produce another discrete-time signal called the *output* or *response* of the system. In general, we view a system as an operation or a set of operations performed on the input signal $x(n)$ to produce the output signal $y(n)$. We say that the input signal $x(n)$ is *transformed* by the system into a signal $y(n)$, and express the general relationship between $x(n)$ and $y(n)$ as

$$y(n) \equiv \mathcal{T}[x(n)] \tag{2.2.1}$$

where the symbol $\mathcal{T}$ denotes the transformation (also called an operator), or processing performed by the system on $x(n)$ to produce $y(n)$. The mathematical relationship in (2.2.1) is depicted graphically in Fig. 2.12.

There are various ways to describe the characteristics of the system and the operation it performs on $x(n)$ to produce $y(n)$. In this chapter we shall be concerned with the time-domain characterization of systems. We shall begin with an input–output description of the system. The input–output description focuses on the behavior at the terminals of the system and ignores the detailed internal construction or realization of the system. Later, in Section 7.5, we introduce the state-space description of a system. In this description we develop mathematical equations that not only describe the input–output behavior of the system but specify its internal behavior and structure.

### 2.2.1 Input–Output Description of Systems

The input–output description of a discrete-time system consists of a mathematical expression or a rule, which explicitly defines the relation between the input and output signals (*input–output relationship*). The exact internal structure of the system is either unknown or ignored. Thus the only way to interact with the system is by using its input and output terminals (i.e., the system is assumed to be a "black box" to the user). To reflect this philosophy, we use the graphical representa-

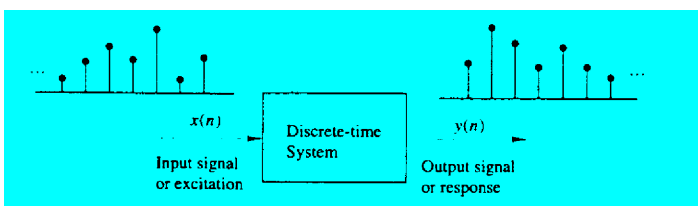



**Figure 2.12** Block diagram representation of a discrete-time system.

tion depicted in Fig. 2.12, and the general input–output relationship in (2.2.1) or, alternatively, the notation

$$x(n) \xrightarrow{\mathcal{T}} y(n) \tag{2.2.2}$$

which simply means that $y(n)$ is the response of the system $\mathcal{T}$ to the excitation $x(n)$. The following examples illustrate several different systems.

**Example 2.2.1**

Determine the response of the following sytems to the input signal

$$x(n) = \begin{cases} |n|, & -3 \le n \le 3 \\ 0, & \text{otherwise} \end{cases}$$

**(a)** $y(n) = x(n)$
**(b)** $y(n) = x(n-1)$
**(c)** $y(n) = x(n+1)$
**(d)** $y(n) = \frac{1}{3}[x(n+1) + x(n) + x(n-1)]$
**(e)** $y(n) = max\{x(n+1), x(n), x(n-1)\}$
**(f)** $y(n) = \sum_{k=-\infty}^{n} x(k) = x(n) + x(n-1) + x(n-2) + \cdots$ (2.2.3)

**Solution** First, we determine explicitly the sample values of the input signal

$$x(n) = \{\ldots, 0, 3, 2, 1, 0, 1, 2, 3, 0, \ldots\}$$
$$\uparrow$$

Next, we determine the output of each system using its input–output relationship.

**(a)** In this case the output is exactly the same as the input signal. Such a system is known as the *identity* system.

**(b)** This system simply delays the input by one sample. Thus its output is given by

$$x(n) = \{\ldots, 0, 3, 2, 1, 0, 1, 2, 3, 0, \ldots\}$$
$$\uparrow$$

**(c)** In this case the system "advances" the input one sample into the future. For example, the value of the output at time $n = 0$ is $y(0) = x(1)$. The response of this system to the given input is

$$x(n) = \{\ldots, 0, 3, 2, 1, 0, 1, 2, 3, 0, \ldots\}$$
$$\uparrow$$

**(d)** The output of this system at any time is the mean value of the present, the immediate past, and the immediate future samples. For example, the output at time $n = 0$ is

$$y(0) = \frac{1}{3}[x(-1) + x(0) + x(1)] = \frac{1}{3}[1 + 0 + 1] = \frac{2}{3}$$

Repeating this computation for every value of n, we obtain the output signal

$$y(n) = \{\ldots, 0, 1, \tfrac{5}{3}, 2, 1, \tfrac{2}{3}, 1, 2, \tfrac{5}{3}, 1, 0, \ldots\}$$
$$\uparrow$$

**(e)** This system selects as its output at time $n$ the maximum value of the three input samples $x(n-1)$, $x(n)$, and $x(n+1)$. Thus the response of this system to the input signal $x(n)$ is

$$y(n) = \{0, 3, 3, 3, 2, 1, 2, 3, 3, 3, 0, \ldots\}$$
$$\uparrow$$

**(f)** This system is basically an *accumulator* that computes the running sum of all the past input values up to present time. The response of this system to the given input is

$$x(n) = \{\ldots, 0, 3, 2, 1, 0, 1, 2, 3, 0, \ldots\}$$
$$\uparrow$$

(f) $y(n) = \sum_{k=-\infty}^{n} x(k) = x(n) + x(n-1) + x(n-2) + \cdots$

$$y(n) = \{\ldots, 0, 3, 5, 6, 6, 7, 9, 12, 0, \ldots\}$$
$$\uparrow$$

We observe that for several of the systems considered in Example 2.2.1 the output at time $n = n_0$ depends not only on the value of the input at $n = n_0$ [i.e., $x(n_0)$], but also on the values of the input applied to the system before and after $n = n_0$. Consider, for instance, the accumulator in the example. We see that the output at time $n = n_0$ depends not only on the input at time $n = n_0$, but also on $x(n)$ at times $n = n_0 - 1$, $n_0 - 2$, and so on. By a simple algebraic manipulation the input-output relation of the accumulator can be written as

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{n} x(k) = \sum_{k=-\infty}^{n-1} x(k) + x(n) \\ &= y(n-1) + x(n) \end{aligned} \tag{2.2.4}$$

which justifies the term *accumulator*. Indeed, the system computes the current value of the output by adding (accumulating) the current value of the input to the previous output value.

There are some interesting conclusions that can be drawn by taking a close look into this apparently simple system. Suppose that we are given the input signal $x(n)$ for $n \geq n_0$, and we wish to determine the output $y(n)$ of this system for $n \geq n_0$. For $n = n_0, n_0 + 1, \ldots$, (2.2.4) gives

$$y(n_0) = y(n_0 - 1) + x(n_0)$$
$$y(n_0 + 1) = y(n_0) + x(n_0 + 1)$$

and so on. Note that we have a problem in computing $y(n_0)$, since it depends on $y(n_0 - 1)$. However,

$$y(n_0 - 1) = \sum_{k=-\infty}^{n_0-1} x(k)$$

that is, $y(n_0 - 1)$ "summarizes" the effect on the system from all the inputs which had been applied to the system before time $n_0$. Thus the response of the system for $n \geq n_0$ to the input $x(n)$ that is applied at time $n_0$ is the combined result of this input and all inputs that had been applied previously to the system. Consequently, $y(n)$, $n \geq n_0$ is not uniquely determined by the input $x(n)$ for $n \geq n_0$.

The additional information required to determine $y(n)$ for $n \geq n_0$ is the *initial condition* $y(n_0 - 1)$. This value summarizes the effect of all previous inputs to the system. Thus the initial condition $y(n_0 - 1)$ together with the input sequence $x(n)$ for $n \geq n_0$ uniquely determine the output sequence $y(n)$ for $n \geq n_0$.

If the accumulator had no excitation prior to $n_0$, the initial condition is $y(n_0 - 1) = 0$. In such a case we say that the system is *initially relaxed*. Since $y(n_0 - 1) = 0$, the output sequence $y(n)$ depends only on the input sequence $x(n)$ for $n \geq n_0$.

It is customary to assume that every system is relaxed at $n = -\infty$. In this case, if an input $x(n)$ is applied at $n = -\infty$, the corresponding output $y(n)$ is *solely* and *uniquely* determined by the given input.

**Example 2.2.2**

The accumulator described by (2.2.3) is excited by the sequence $x(n) = nu(n)$. Determine its output under the condition that:

**(a)** It is initially relaxed [i.e., $y(-1) = 0$].

**(b)** Initially, $y(-1) = 1$.

**Solution** The output of the system is defined as

$$y(n) = \sum_{k=-\infty}^{n} x(k) = \sum_{k=-\infty}^{-1} x(k) + \sum_{k=0}^{n} x(k)$$

$$= y(-1) + \sum_{k=0}^{n} x(k)$$

But

$$\sum_{k=0}^{n} x(k) = \frac{n(n+1)}{2}$$

**(a)** If the system is initially relaxed, $y(-1) = 0$ and hence

$$y(n) = \frac{n(n+1)}{2} \qquad n \geq 0$$

**(b)** On the other hand, if the initial condition is $y(-1) = 1$, then

$$y(n) = 1 + \frac{n(n+1)}{2} = \frac{n^2 + n + 2}{2} \qquad n \geq 0$$

### 2.2.2 Block Diagram Representation of Discrete-Time Systems

It is useful at this point to introduce a block diagram representation of discrete-time systems. For this purpose we need to define some basic building blocks that can be interconnected to form complex systems.

**An adder.** Figure 2.13 illustrates a system (adder) that performs the addition of two signal sequences to form another (the sum) sequence, which we denote

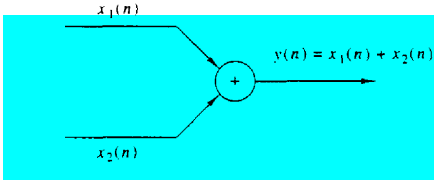



**Figure 2.13** Graphical representation of an adder.

as $y(n)$. Note that it is not necessary to store either one of the sequences in order to perform the addition. In other words, the addition operation is *memoryless*.

**A constant multiplier.** This operation is depicted by Fig. 2.14, and simply represents applying a scale factor on the input $x(n)$. Note that this operation is also memoryless.

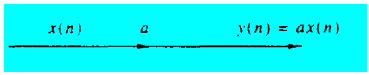



**Figure 2.14** Graphical representation of a constant multiplier.

**A signal multiplier.** Figure 2.15 illustrates the multiplication of two signal sequences to form another (the product) sequence, denoted in the figure as $y(n)$. As in the preceding two cases, we can view the multiplication operation as memoryless.

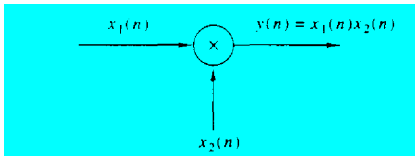



**Figure 2.15** Graphical representation of a signal multiplier.

**A unit delay element.** The unit delay is a special system that simply delays the signal passing through it by one sample. Figure 2.16 illustrates such a system. If the input signal is $x(n)$, the output is $x(n-1)$. In fact, the sample $x(n-1)$ is stored in memory at time $n-1$ and it is recalled from memory at time $n$ to form

$$y(n) = x(n-1)$$

Thus this basic building block requires memory. The use of the symbol $z^{-1}$ to denote the unit of delay will become apparent when we discuss the $z$-transform in Chapter 3.

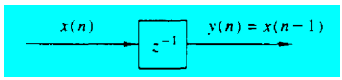



**Figure 2.16** Graphical representation of the unit delay element.

**A unit advance element.** In contrast to the unit delay, a unit advance moves the input $x(n)$ ahead by one sample in time to yield $x(n+1)$. Figure 2.17 illustrates this operation, with the operator $z$ being used to denote the unit advance.

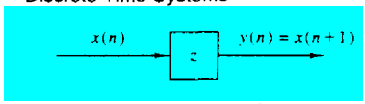



**Figure 2.17** Graphical representation of the unit advance element.

We observe that any such advance is physically impossible in real time, since, in fact, it involves looking into the future of the signal. On the other hand, if we store the signal in the memory of the computer, we can recall any sample at any time. In such a nonreal-time application, it is possible to advance the signal $x(n)$ in time.

**Example 2.2.3**

Using basic building blocks introduced above, sketch the block diagram representation of the discrete-time system described by the input–output relation.

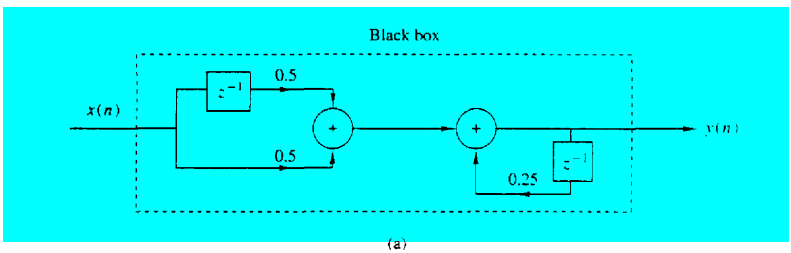
$$y(n) = \tfrac{1}{4}y(n-1) + \tfrac{1}{2}x(n) + \tfrac{1}{2}x(n-1) \tag{2.2.5}$$

where $x(n)$ is the input and $y(n)$ is the output of the system.

**Solution** According to (2.2.5), the output $y(n)$ is obtained by multiplying the input $x(n)$ by 0.5, multiplying the previous input $x(n-1)$ by 0.5, adding the two products, and then adding the previous output $y(n-1)$ multiplied by $\frac{1}{4}$. Figure 2.18a illustrates this block diagram realization of the system. A simple rearrangement of (2.2.5), namely,

$$y(n) = \tfrac{1}{4}y(n-1) + \tfrac{1}{2}[x(n) + x(n-1)] \tag{2.2.6}$$

leads to the block diagram realization shown in Fig. 2.18b. Note that if we treat "the system" from the "viewpoint" of an input–output or an external description, we are not concerned about how the system is realized. On the other hand, if we adopt an



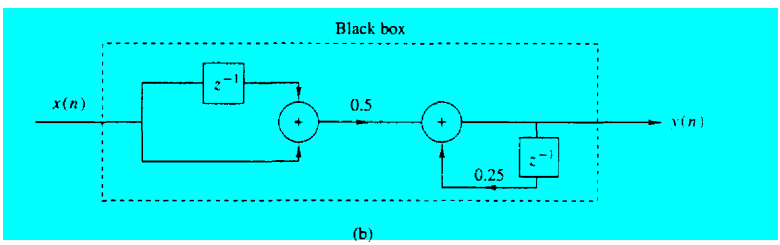




**Figure 2.18** Block diagram realizations of the system $y(n) = 0.25y(n-1) + 0.5x(n) + 0.5x(n-1)$.