12–1 Analog-to-Digital Conversion

In order to process signals using digital techniques, the incoming analog signal must be converted into digital form.

After completing this section, you should be able to

- Explain the basic process of converting an analog signal to digital
- Describe the purpose of the sample-and-hold function
- Define the Nyquist frequency
- Define the reason for *aliasing* and discuss how it is eliminated
- Describe the purpose of an ADC

Sampling and Filtering

An anti-aliasing filter and a sample-and-hold circuit are two functions typically found in a digital signal processing system. The sample-and-hold function does two operations, the first of which is sampling. **Sampling** is the process of taking a sufficient number of discrete values at points on a waveform that will define the shape of the waveform. The more samples you take, the more accurately you can define a waveform. Sampling converts an analog signal into a series of impulses, each representing the amplitude of the signal at a given instant in time. Figure 12–1 illustrates the process of sampling.

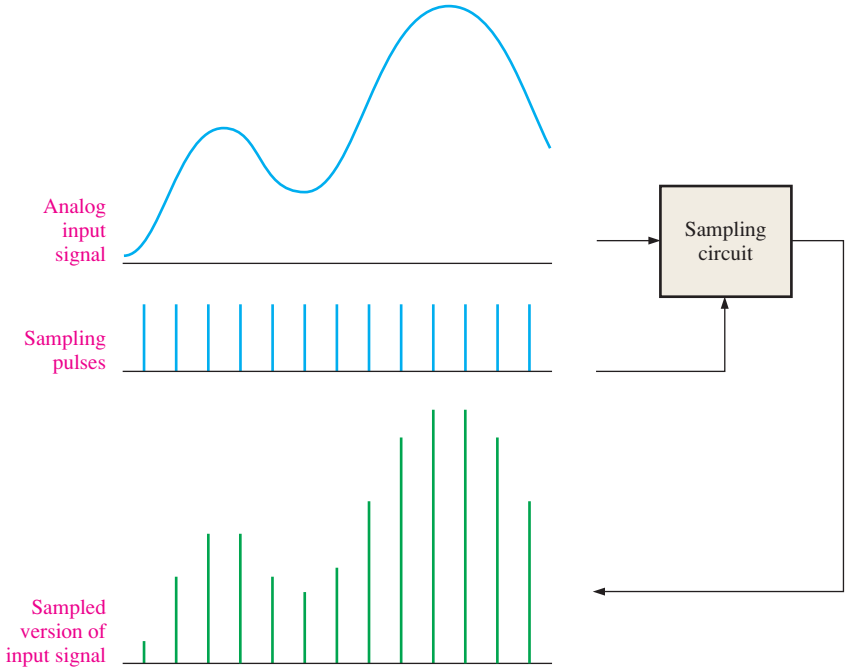


FIGURE 12–1 Illustration of the sampling process.

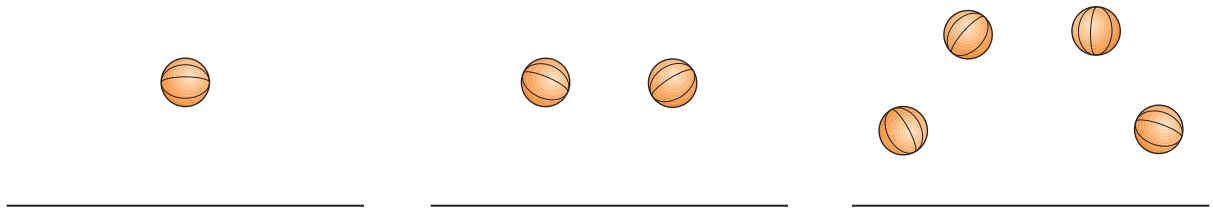
When an analog signal is to be sampled, there are certain criteria that must be met in order to accurately represent the original signal. All analog signals (except a pure sine wave) contain a spectrum of component frequencies. For a pure sine wave, these frequencies appear in multiples called *harmonics*. The harmonics of an analog signal are sine waves of different frequencies and amplitudes. When the harmonics of a given periodic waveform are added, the result is the original signal. Before a signal can be sampled, it must be passed through a low-pass filter (anti-aliasing filter) to eliminate harmonic frequencies above a certain value as determined by the Nyquist frequency.

The Sampling Theorem

Notice in Figure 12–1 that there are two input waveforms. One is the analog signal and the other is the sampling pulse waveform. The sampling theorem states that, in order to represent an analog signal, the sampling frequency, f_{sample}, must be at least twice the highest frequency component $f_{a(max)}$ of the analog signal. Another way to say this is that the highest analog frequency can be no greater than one-half the sampling frequency. The frequency $f_{a(max)}$ is known as the **Nyquist frequency** and is expressed in Equation 12–1. In practice, the sampling frequency should be more than twice the highest analog frequency.

$$f_{sample} > 2f_{a(max)} \qquad \text{Equation 12–1}$$

To intuitively understand the sampling theorem, a simple "bouncing-ball" analogy may be helpful. Although it is not a perfect representation of the sampling of electrical signals, it does serve to illustrate the basic idea. If a ball is photographed (sampled) at one instant during a single bounce, as illustrated in Figure 12–2(a), you cannot tell anything about the path of the ball except that it is off the floor. You can't tell whether it is going up or down or the distance of its bounce. If you take photos at two equally-spaced instants during one bounce, as shown in part (b), you can obtain only a minimum amount of information about its movement and nothing about the distance of the bounce. In this particular case, you know only that the ball has been in the air at the times the two photos were taken and that the maximum height of the bounce is at least equal to the height shown in each photo. If you take four photos, as shown in part (c), then the path that the ball follows during a bounce begins to emerge. The more photos (samples) that you take, the more accurately you can determine the path of the ball as it bounces.



(a) One sample of a ball during a single bounce

(b) Two samples of a ball during a single bounce. This is the absolute minimum required to tell anything about its movement, but generally insufficient to describe its path.

(c) Four samples of a ball during a single bounce form a rough picture of the path of the ball.

FIGURE 12–2 Bouncing ball analogy of sampling theory.

The Need for Filtering

Low-pass filtering is necessary to remove all frequency components (harmonics) of the analog signal that exceed the Nyquist frequency. If there are any frequency components in the analog signal that exceed the Nyquist frequency, an unwanted condition known as **aliasing** will occur. An alias is a signal produced when the sampling frequency is not at least twice the signal frequency. An alias signal has a frequency that is less than the highest frequency in the analog signal being sampled and therefore falls within the spectrum or frequency band of the input analog signal causing distortion. Such a signal is actually "posing" as part of the analog signal when it really isn't, thus the term *alias*.

Another way to view aliasing is by considering that the sampling pulses produce a spectrum of harmonic frequencies above and below the sample frequency, as shown in Figure 12–3. If the analog signal contains frequencies above the Nyquist frequency, these frequencies overlap into the spectrum of the sample waveform as shown and interference occurs. The lower frequency components of the sampling waveform become mixed in with the frequency spectra of the analog waveform, resulting in an aliasing error.

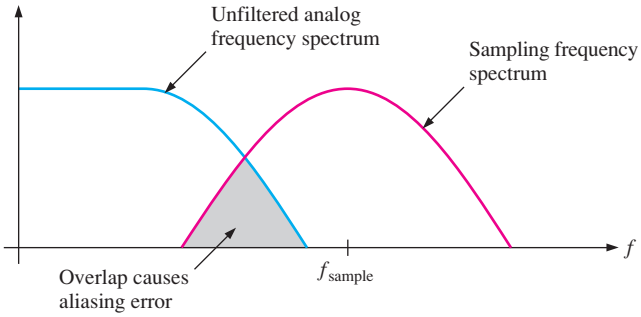


FIGURE 12–3 A basic illustration of the condition $f_{sample} < 2f_{a(max)}$.

A low-pass anti-aliasing filter must be used to limit the frequency spectrum of the analog signal for a given sample frequency. To avoid an aliasing error, the filter must at least eliminate all analog frequencies above the minimum frequency in the sampling spectrum, as illustrated in Figure 12–4. Aliasing can also be avoided by sufficiently increasing the sampling frequency. However, the maximum sampling frequency is usually limited by the performance of the analog-to-digital converter (ADC) that follows it.

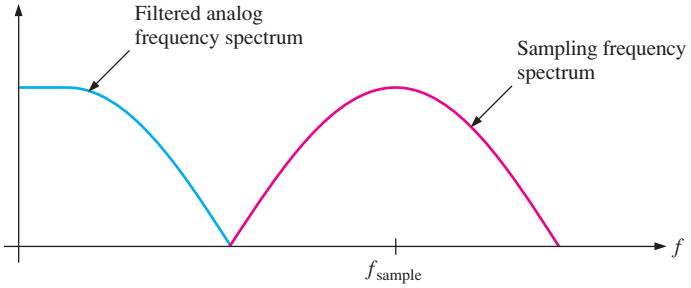


FIGURE 12–4 After low-pass filtering, the frequency spectra of the analog and the sampling signals do not overlap, thus eliminating aliasing error.

An Application

An example of the application of sampling is in digital audio equipment. The sampling rates used are 32 kHz, 44.1 kHz, or 48 kHz (the number of samples per second). The 48 kHz rate is the most common, but the 44.1 kHz rate is used for audio CDs and prerecorded tapes. According to the Nyquist rate, the sampling frequency must be at least twice the audio signal. Therefore, the CD sampling rate of 44.1 kHz captures frequencies up to about 22 kHz, which exceeds the 20 kHz specification that is common for most audio equipment.

Many applications do not require a wide frequency range to obtain reproduced sound that is acceptable. For example, human speech contains some frequencies near 10 kHz and, therefore, requires a sampling rate of at least 20 kHz. However, if only frequencies up to 4 kHz (ideally requiring an 8 kHz minimum sampling rate) are reproduced, voice is very understandable. On the other hand, if a sound signal is not sampled at a high enough rate, the effect of aliasing will become noticeable with background noise and distortion.

Holding the Sampled Value

The holding operation is the second part of the sample-and-hold function. After filtering and sampling, the sampled level must be held constant until the next sample occurs. This is necessary for the ADC to have time to process the sampled value. This sample-and-hold operation results in a "stairstep" waveform that approximates the analog input waveform, as shown in Figure 12–5.

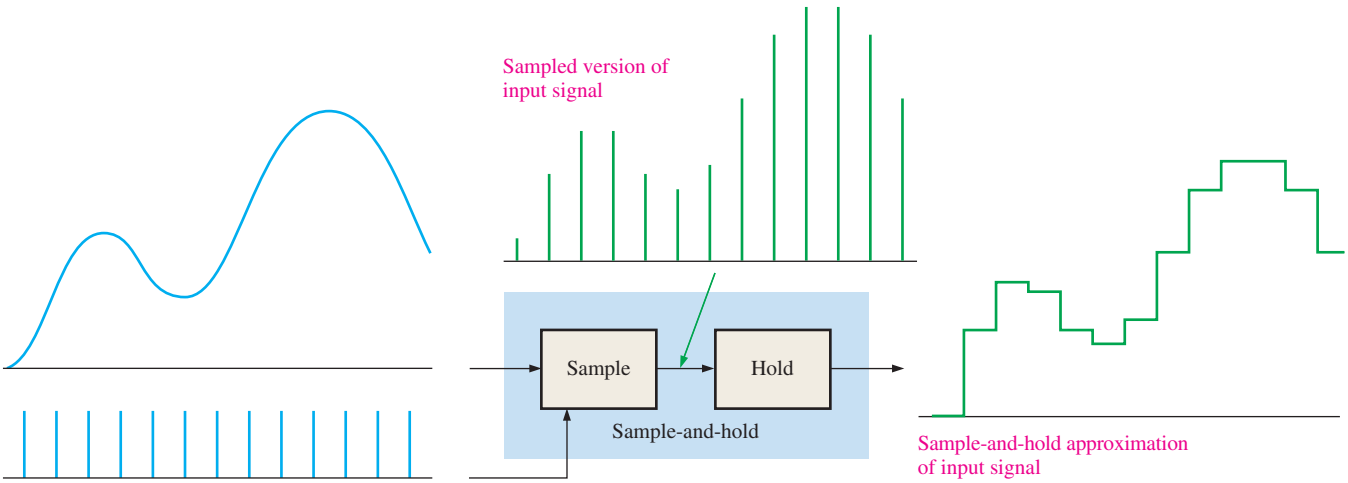


FIGURE 12–5 Illustration of a sample-and-hold operation.

Analog-to-Digital Conversion

Analog-to-digital conversion is the process of converting the output of the sample-and-hold circuit to a series of binary codes that represent the amplitude of the analog input at each of the sample times. The sample-and-hold process keeps the amplitude of the analog input signal constant between sample pulses; therefore, the analog-to-digital conversion can be done using a constant value rather than having the analog signal change during a conversion interval, which is the time between sample pulses. Figure 12–6 illustrates the basic function of an **analog-to-digital converter (ADC)**, which is a circuit that performs analog-to-digital conversion. The sample intervals are indicated by dashed lines.

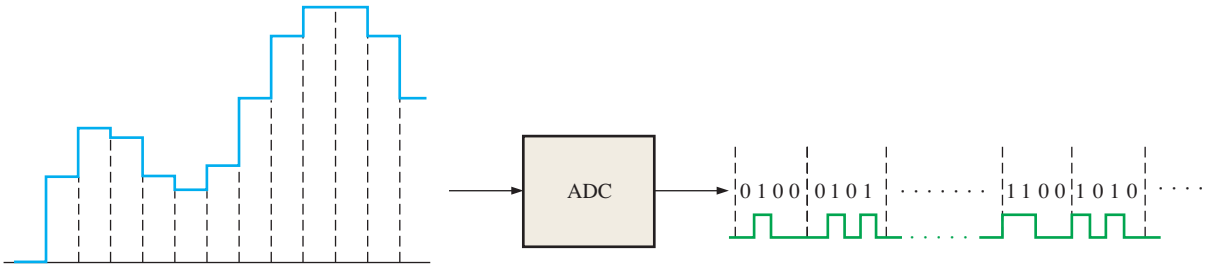


FIGURE 12–6 Basic function of an analog-to-digital converter (ADC) (The binary codes and number of bits are arbitrarily chosen for illustration only). The ADC output waveform that represents the binary codes is also shown.

Quantization

The process of converting an analog value to a code is called **quantization**. During the quantization process, the ADC converts each sampled value of the analog signal to a binary code. The more bits that are used to represent a sampled value, the more accurate is the representation.

To illustrate, let's quantize a reproduction of the analog waveform into four levels (0–3). Two bits are required for four levels. As shown in Figure 12–7, each quantization level is represented by a 2-bit code on the vertical axis, and each sample interval is numbered along the horizontal axis. The sampled data is held for the entire sample period. This data is quantized to the next lower level, as shown in Table 12–1 (for example, compare samples 3 and 4, which are assigned different levels).

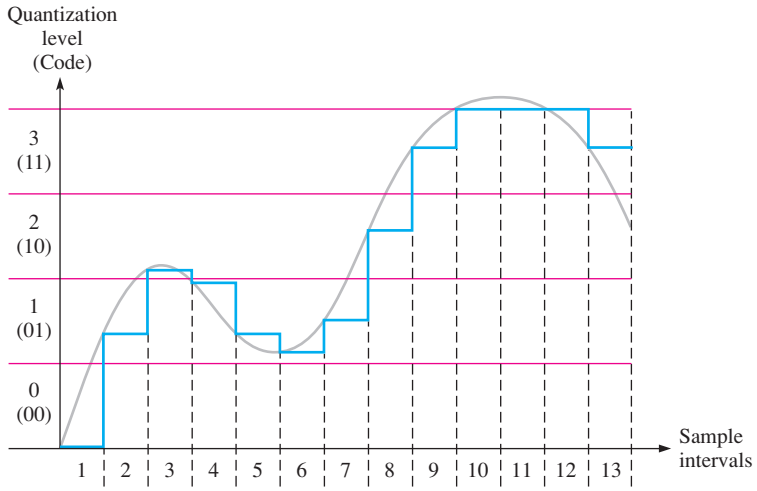


FIGURE 12–7 Sample-and-hold output waveform with four quantization levels. The original analog waveform is shown in light gray for reference.

TABLE 12–1

Two-bit quantization for the waveform in Figure 12–7.

Sample Interval	Quantization Level	Code
1	0	00
2	1	01
3	2	10
4	1	01
5	1	01
6	1	01
7	1	01
8	2	10
9	3	11
10	3	11
11	3	11
12	3	11
13	3	11

If the resulting 2-bit digital codes are used to reconstruct the original waveform, you would get the waveform shown in Figure 12–8. This operation is done by **digital-to-analog converters (DACs)**, which are circuits that perform digital-to-analog conversions. As you can see, quite a bit of accuracy is lost using only two bits to represent the sampled values.

Now, let's see how more bits will improve the accuracy. Figure 12–9 shows the same waveform with sixteen quantization levels (4 bits). The 4-bit quantization process is summarized in Table 12–2.

If the resulting 4-bit digital codes are used to reconstruct the original waveform, you would get the waveform shown in Figure 12–10. As you can see, the result is much more like the original waveform than for the case of four quantization levels in Figure 12–8. This shows that greater accuracy is achieved with more quantization bits. Typical integrated circuit ADCs use from 12 to 24 bits, and the sample-and-hold function is sometimes contained on the ADC chip. Several types of ADCs are introduced in the next section.

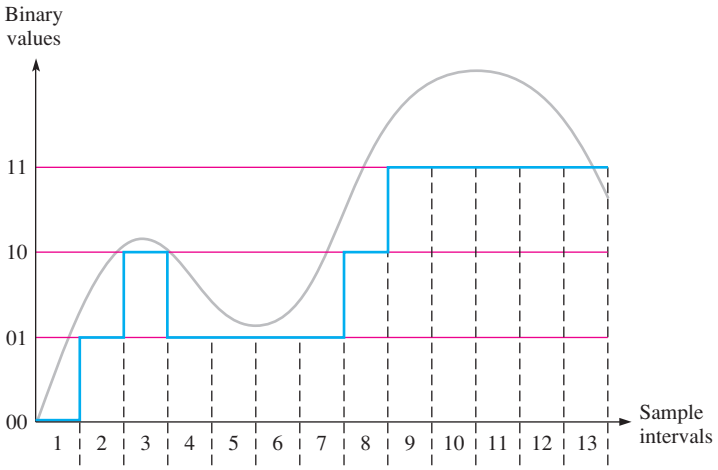


FIGURE 12–8 The reconstructed waveform in Figure 12–7 using four quantization levels (2 bits). The original analog waveform is shown in light gray for reference.

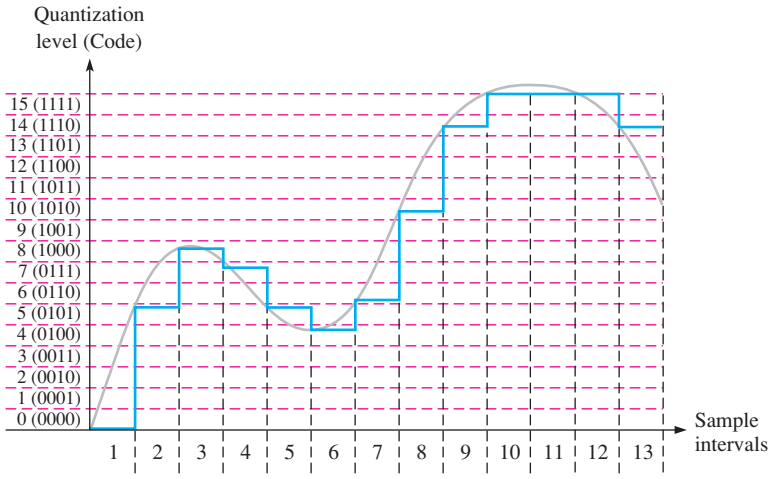


FIGURE 12–9 Sample-and-hold output waveform with sixteen quantization levels. The original analog waveform is shown in light gray for reference.

TABLE 12–2

Four-bit quantization for the waveform in Figure 12–9.

Sample Interval	Quantization Level	Code
1	0	0000
2	5	0101
3	8	1000
4	7	0111
5	5	0101
6	4	0100
7	6	0110
8	10	1010
9	14	1110
10	15	1111
11	15	1111
12	15	1111
13	14	1110

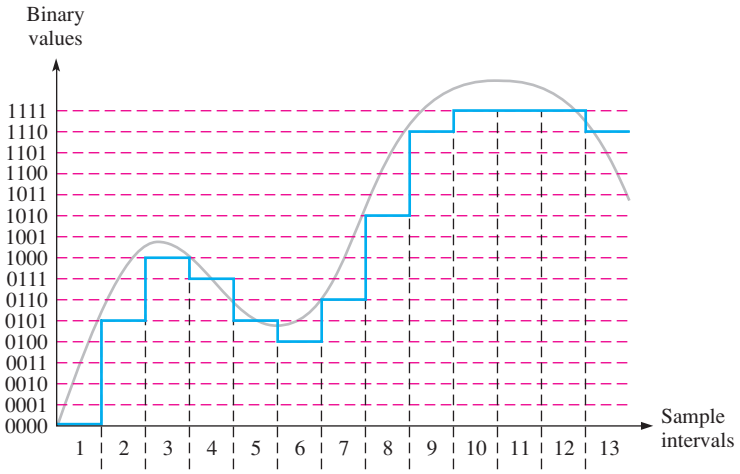


FIGURE 12–10 The reconstructed waveform in Figure 12–9 using sixteen quantization levels (4 bits). The original analog waveform is shown in light gray for reference.

SECTION 12–1 CHECKUP

Answers are at the end of the chapter.

1. What does sampling mean?
2. Why must you hold a sampled value?
3. If the highest frequency component in an analog signal is 20 kHz, what is the minimum sample frequency?
4. What does quantization mean?
5. What determines the accuracy of the quantization process?

12–2 Methods of Analog-to-Digital Conversion

As you have seen, analog-to-digital conversion is the process by which an analog quantity is converted to digital form. It is necessary when measured quantities must be in digital form for processing or for display or storage. Some common types of analog-to-digital converters (ADCs) are now examined. Two important ADC parameters are *resolution,* which is the number of bits, and *throughput,* which is the sampling rate an ADC can handle in units of samples per second (sps).

After completing this section, you should be able to

- Explain what an operational amplifier is
- Show how the op-amp can be used as an inverting amplifier or a comparator
- Explain how a flash ADC works
- Discuss dual-slope ADCs
- Describe the operation of a successive-approximation ADC
- Describe a delta-sigma ADC
- Discuss testing ADCs for a missing code, incorrect code and offset

A Quick Look at an Operational Amplifier

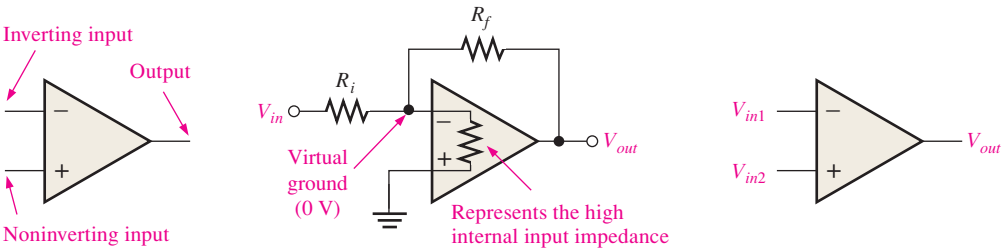
Before getting into analog-to-digital converters (ADCs), let's look briefly at an element that is common to most types of ADCs and digital-to-analog converters (DACs). This element is the operational amplifier, or op-amp for short. This is an abbreviated coverage of the op-amp.

An **op-amp** is a linear amplifier that has two inputs (inverting and noninverting) and one output. It has a very high voltage gain and a very high input impedance, as well as a very low output impedance. The op-amp symbol is shown in Figure 12–11(a). When used as an inverting amplifier, the op-amp is configured as shown in part (b). The feedback resistor, R_f, and the input resistor, R_i, control the voltage gain according to the formula in Equation 12–2, where V_{out}/V_{in} is the closed-loop voltage gain (closed loop refers to the feedback from output to input provided by R_f). The negative sign indicates inversion.

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_i} \qquad \text{Equation 12–2}$$

In the inverting amplifier configuration, the inverting input of the op-amp is approximately at ground potential (0 V) because feedback and the extremely high open-loop gain make the differential voltage between the two inputs extremely small. Since the noninverting input is grounded, the inverting input is at approximately 0 V, which is called *virtual ground.*

When the op-amp is used as a comparator, as shown in Figure 12–11(c), two voltages are applied to the inputs. When these input voltages differ by a very small amount, the op-amp is driven into one of its two saturated output states, either HIGH or LOW, depending on which input voltage is greater.



(a) Op-amp symbol

(b) Op-amp as an inverting amplifier with gain of R_f/R_i

(c) Op-amp as a comparator

FIGURE 12–11 The operational amplifier (op-amp).

Flash (Simultaneous) Analog-to-Digital Converter

The flash method utilizes special high-speed comparators that compare reference voltages with the analog input voltage. When the input voltage exceeds the reference voltage for a given comparator, a HIGH is generated. Figure 12–12 shows a 3-bit converter that uses seven comparator circuits; a comparator is not needed for the all-0s condition. A 4-bit converter of this type requires fifteen comparators. In general, $2^n - 1$ comparators are required for conversion to an n-bit binary code. The number of bits used in an ADC is its **resolution**. The large number of comparators necessary for a reasonable-sized binary number is one of the disadvantages of the **flash ADC**. Its chief advantage is that it provides a fast conversion time because of a high *throughput*, measured in samples per second (sps).

The reference voltage for each comparator is set by the resistive voltage-divider circuit. The output of each comparator is connected to an input of the priority encoder. The encoder is enabled by a pulse on the *EN* input, and a 3-bit code representing the value of the input appears on the encoder's outputs. The binary code is determined by the highest-order input having a HIGH level.

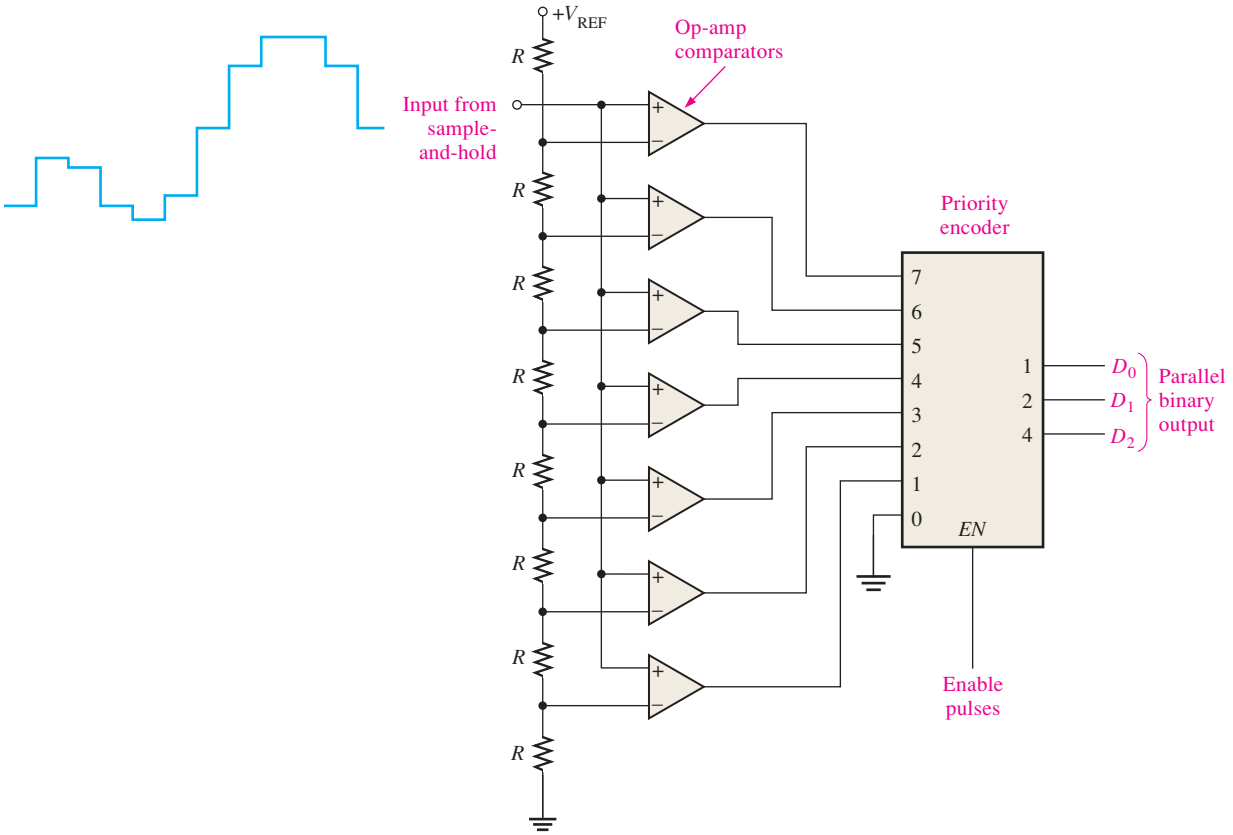


FIGURE 12–12 A 3-bit flash ADC.

The frequency of the enable pulses and the number of bits in the binary code determine the accuracy with which the sequence of binary codes represents the input of the ADC. The signal is sampled each time the enable pulse is active.

EXAMPLE 12–1

Determine the binary code output of the 3-bit flash ADC in Figure 12–12 for the input signal in Figure 12–13 and the encoder enable pulses shown. For this example, $V_{REF} = +8$ V.

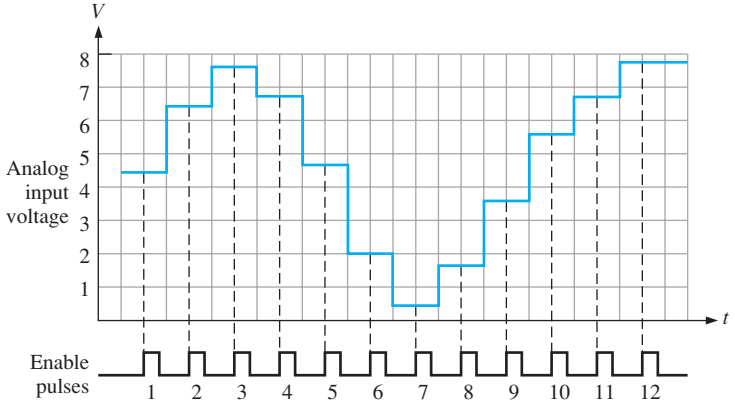


FIGURE 12–13 Sampling of values on a waveform for conversion to binary code.

Solution

The resulting digital output sequence is listed as follows and shown in the waveform diagram of Figure 12–14 in relation to the enable pulses:

100, 110, 111, 110, 100, 010, 000, 001, 011, 101, 110, 111

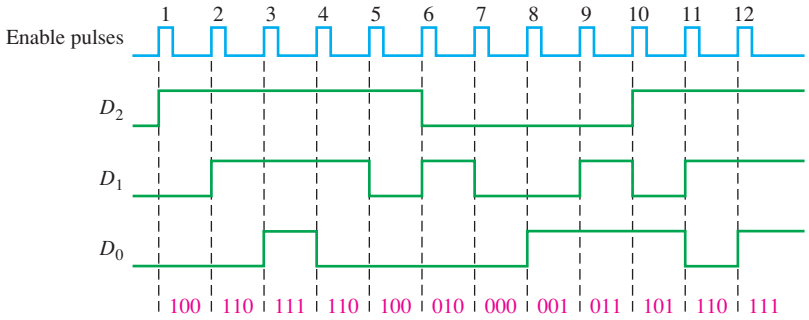


FIGURE 12–14 Resulting digital outputs for sample-and-hold values. Output D_0 is the LSB of the 3-bit binary code.

Related Problem*

If the enable pulse frequency in Figure 12–13 were halved, determine the binary numbers represented by the resulting digital output sequence for 6 pulses. Is any information lost?

*Answers are at the end of the chapter.

Dual-Slope Analog-to-Digital Converter

A dual-slope ADC is common in digital voltmeters and other types of measurement instruments. A ramp generator (integrator) is used to produce the dual-slope characteristic. A block diagram of a dual-slope ADC is shown in Figure 12–15.

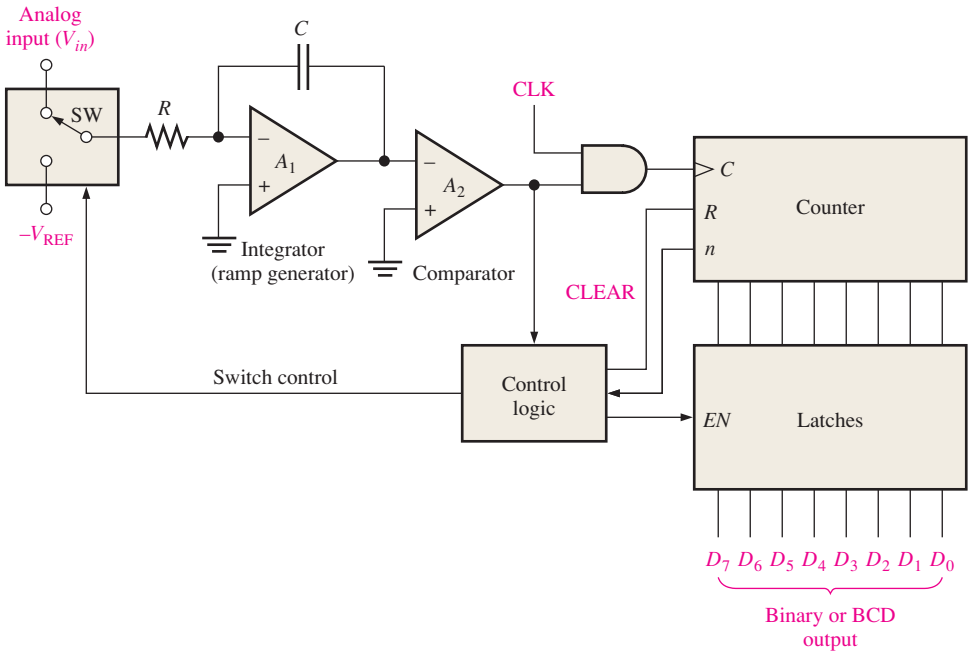
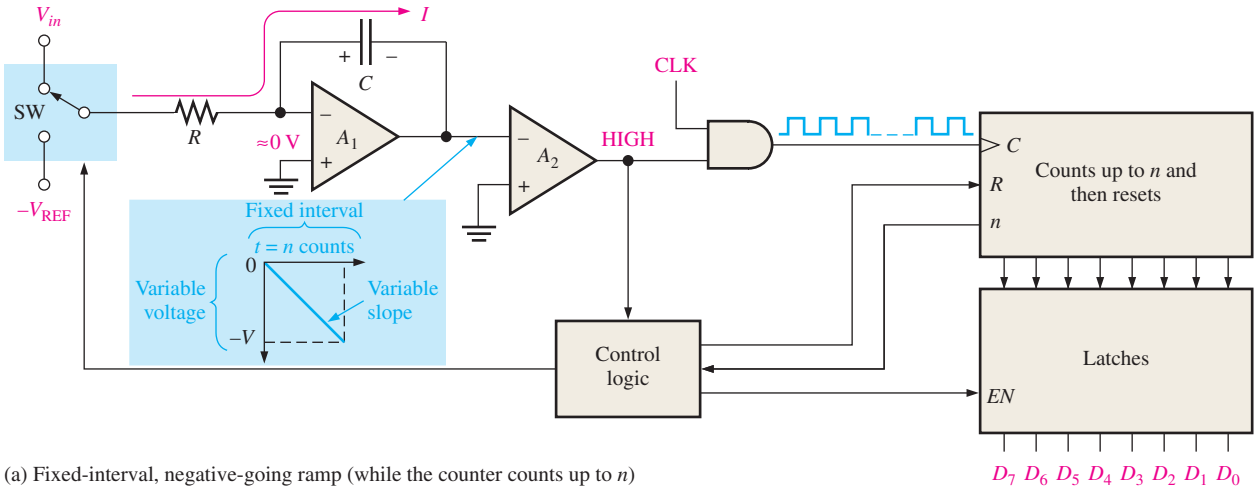
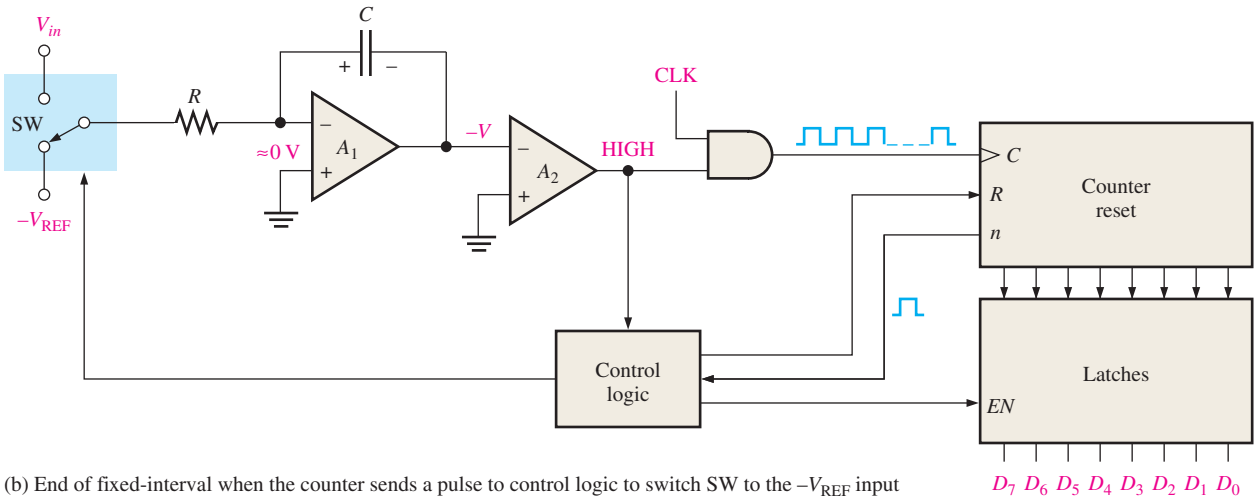


FIGURE 12–15 Basic dual-slope ADC.

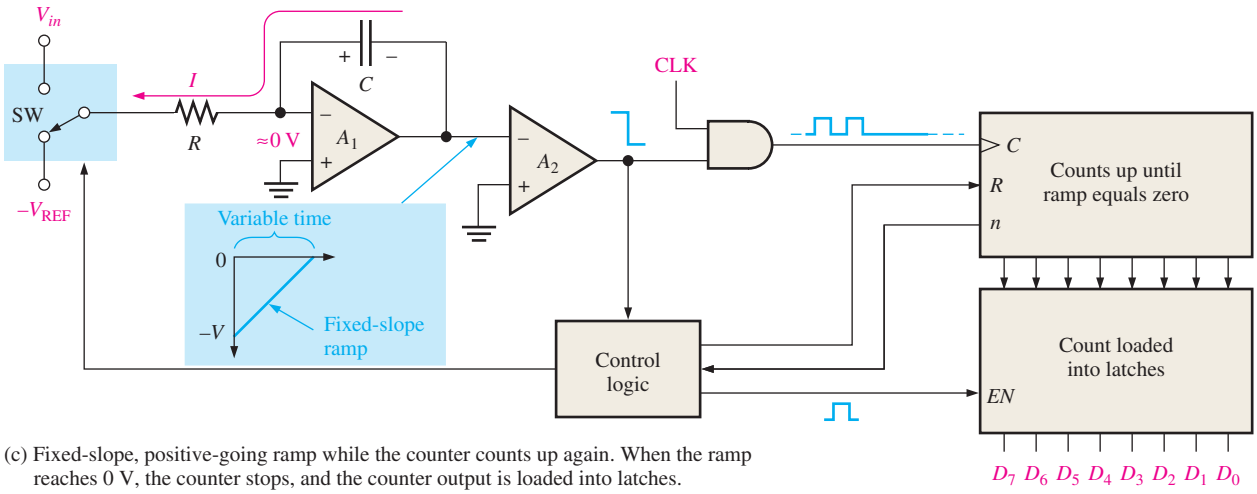
Figure 12–16 illustrates dual-slope conversion. Start by assuming that the counter is reset and the output of the integrator is zero. Now assume that a positive input voltage is applied to the input through the switch (SW) as selected by the control logic. Since the



(a) Fixed-interval, negative-going ramp (while the counter counts up to n)



(b) End of fixed-interval when the counter sends a pulse to control logic to switch SW to the $-V_{REF}$ input



(c) Fixed-slope, positive-going ramp while the counter counts up again. When the ramp reaches 0 V, the counter stops, and the counter output is loaded into latches.

FIGURE 12–16 Illustration of dual-slope conversion.

inverting input of A_1 is at virtual ground, and assuming that V_{in} is constant for a period of time, there will be constant current through the input resistor R and therefore through the capacitor C. Capacitor C will charge linearly because the current is constant, and as a result, there will be a negative-going linear voltage ramp on the output of A_1, as illustrated in Figure 12–16(a).

When the counter reaches a specified count (n), it will be reset (R), and the control logic will switch the negative reference voltage ($-V_{REF}$) to the input of A_1, as shown in Figure 12–16(b). At this point the capacitor is charged to a negative voltage ($-V$) proportional to the input analog voltage.

Now the capacitor discharges linearly because of the constant current from the $-V_{REF}$, as shown in Figure 12–16(c). This linear discharge produces a positive-going ramp on the A_1 output, starting at $-V$ and having a constant slope that is independent of the charge voltage. As the capacitor discharges, the counter advances from its RESET state. The time it takes the capacitor to discharge to zero depends on the initial voltage $-V$ (proportional to V_{in}) because the discharge rate (slope) is constant. When the integrator (A_1) output voltage reaches zero, the comparator (A_2) switches to the LOW state and disables the clock to the counter. The binary count is latched, thus completing one conversion cycle. The binary count is proportional to V_{in} because the time it takes the capacitor to discharge depends only on $-V$, and the counter records this interval of time.

Successive-Approximation Analog-to-Digital Converter

One of the most widely used methods of analog-to-digital conversion is successive-approximation. It has a much faster conversion time than the dual-slope conversion, but it is slower than the flash method. It also has a fixed conversion time that is the same for any value of the analog input.

Figure 12–17 shows a basic block diagram of a 4-bit successive approximation ADC. It consists of a DAC (DACs are covered in Section 12–3), a successive-approximation register (SAR), and a comparator. The basic operation is as follows: The input bits of the DAC are enabled (made equal to a 1) one at a time, starting with the most significant bit (MSB). As each bit is enabled, the comparator produces an output that indicates whether the input signal voltage is greater or less than the output of the DAC. If the DAC output is greater than the input signal, the comparator's output is LOW, causing the bit in the register to reset. If the output is less than the input signal, the 1 bit is retained in the register. The system does this with the MSB first, then the next most

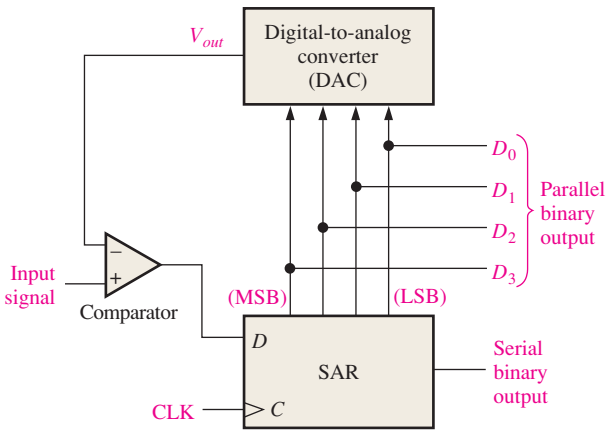


FIGURE 12–17 Successive-approximation ADC.

significant bit, then the next, and so on. After all the bits of the DAC have been tried, the conversion cycle is complete.

In order to better understand the operation of the successive-approximation ADC, let's take a specific example of a 4-bit conversion. Figure 12–18 illustrates the step-by-step conversion of a constant input voltage (5.1 V in this case). Let's assume that the DAC has the following output characteristics: $V_{out} = 8$ V for the 2^3 bit (MSB), $V_{out} = 4$ V for the 2^2 bit, $V_{out} = 2$ V for the 2^1 bit, and $V_{out} = 1$ V for the 2^0 bit (LSB).

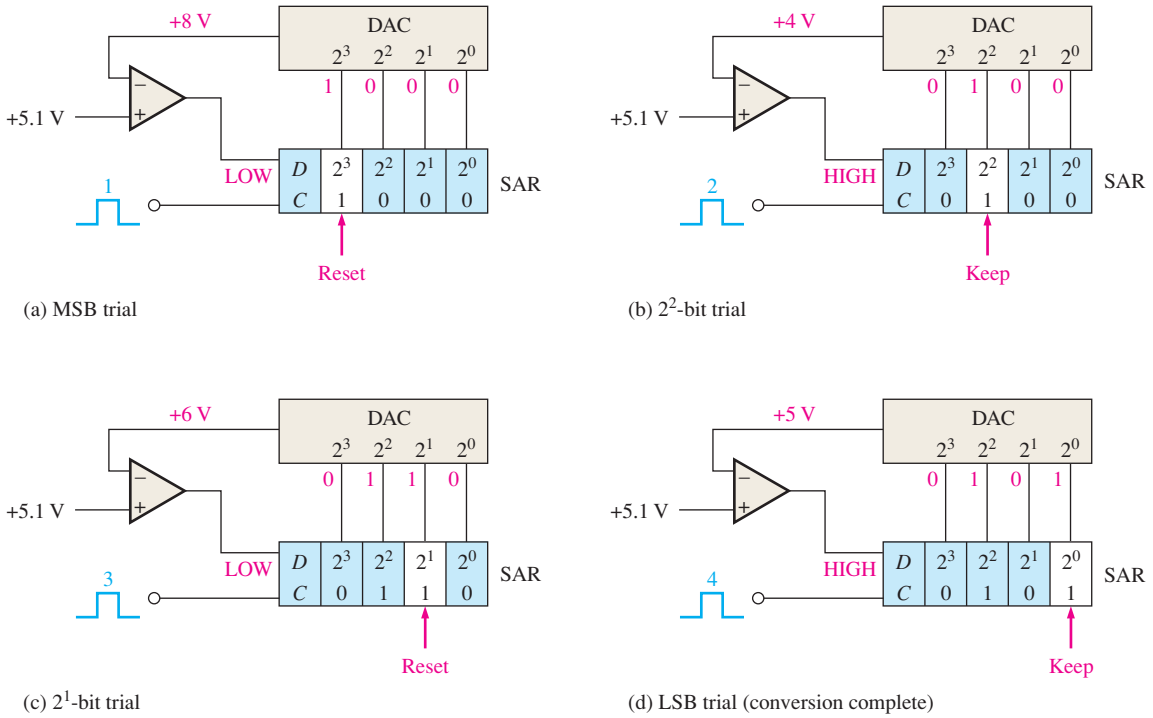


FIGURE 12–18 Illustration of the successive-approximation conversion process.

Figure 12–18(a) shows the first step in the conversion cycle with the MSB = 1. The output of the DAC is 8 V. Since this is greater than the input of 5.1 V, the output of the comparator is LOW, causing the MSB in the SAR to be reset to a 0.

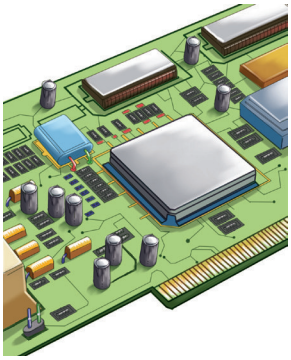
Figure 12–18(b) shows the second step in the conversion cycle with the 2^2 bit equal to a 1. The output of the DAC is 4 V. Since this is less than the input of 5.1 V, the output of the comparator switches to a HIGH, causing this bit to be retained in the SAR.

Figure 12–18(c) shows the third step in the conversion cycle with the 2^1 bit equal to a 1. The output of the DAC is 6 V because there is a 1 on the 2^2 bit input and on the 2^1 bit input; 4 V + 2 V = 6 V. Since this is greater than the input of 5.1 V, the output of the comparator switches to a LOW, causing this bit to be reset to a 0.

Figure 12–18(d) shows the fourth and final step in the conversion cycle with the 2^0 bit equal to a 1. The output of the DAC is 5 V because there is a 1 on the 2^2 bit input and on the 2^0 bit input; 4 V + 1 V = 5 V.

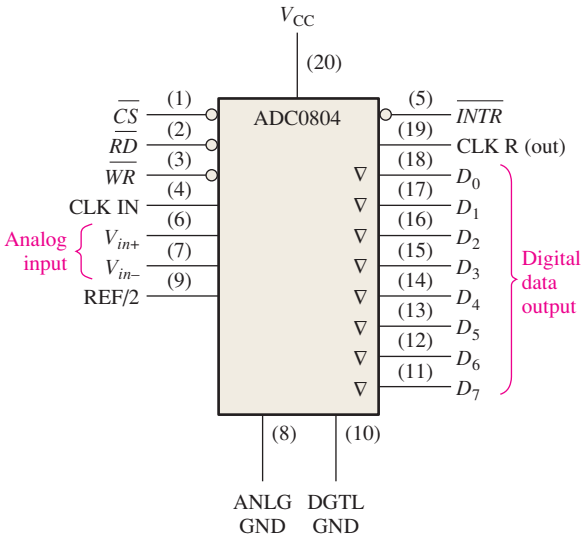
The four bits have all been tried, thus completing the conversion cycle. At this point the binary code in the register is 0101, which is approximately the binary value of the input of 5.1 V. Additional bits will produce an even more accurate result. Another conversion cycle now begins, and the basic process is repeated. The SAR is cleared at the beginning of each cycle.

IMPLEMENTATION: ANALOG-TO-DIGITAL CONVERTER



The ADC0804 is an example of a successive-approximation ADC. A block diagram is shown in Figure 12–19. This device operates from a +5 V supply and has a resolution of eight bits with a conversion time of 100 μs. Also, it has an on-chip clock generator. Optionally, an external clock can be used. The data outputs are tri-state, so they can be interfaced with a microprocessor bus system.

FIGURE 12–19 The ADC0804 analog-to-digital converter.



The basic operation of the device is as follows: The ADC0804 contains the equivalent of a 256-resistor DAC network. The successive-approximation logic sequences the network to match the analog differential input voltage ($V_{in+} - V_{in-}$) with an output from the resistive network. The MSB is tested first. After eight comparisons (sixty-four clock periods), an 8-bit binary code is transferred to output latches, and the interrupt ($\overline{INTR}$) output goes LOW. The device can be operated in a free-running mode by connecting the $\overline{INTR}$ output to the write ($\overline{WR}$) input and holding the conversion start ($\overline{CS}$) LOW. To ensure startup under all conditions, a LOW $\overline{WR}$ input is required during the power-up cycle. Taking $\overline{CS}$ low anytime after that will interrupt the conversion process.

When the $\overline{WR}$ input goes LOW, the internal successive-approximation register (SAR) and the 8-bit shift register are reset. As long as both $\overline{CS}$ and $\overline{WR}$ remain LOW, the ADC remains in a RESET state. Conversion starts one to eight clock periods after $\overline{CS}$ or $\overline{WR}$ makes a LOW-to-HIGH transition.

When a LOW is at both the $\overline{CS}$ and $\overline{RD}$ inputs, the tri-state output latch is enabled and the output code is applied to the D_0–D_7 lines. When either the $\overline{CS}$ or the $\overline{RD}$ input returns to a HIGH, the D_0–D_7 outputs are disabled.

Sigma-Delta Analog-to-Digital Converter

Sigma-delta is a widely used method of analog-to-digital conversion, particularly in telecommunications using audio signals. The method is based on **delta modulation** where the difference between two successive samples (increase or decrease) is quantized; other ADC methods were based on the absolute value of a sample. Delta modulation is a 1-bit quantization method.

The output of a delta modulator is a single-bit data stream where the relative number of 1s and 0s indicates the level or amplitude of the input signal. The number of 1s over a given number of clock cycles establishes the signal amplitude during that interval. A maximum number of 1s corresponds to the maximum positive input voltage. A number of 1s equal to one-half the

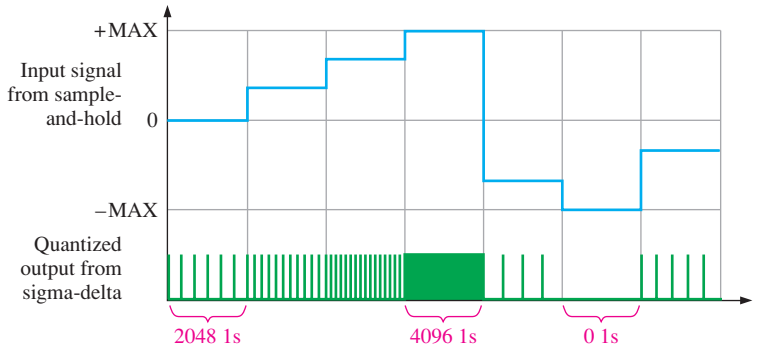


FIGURE 12–20 A simplified illustration of sigma-delta analog-to-digital conversion.

maximum corresponds to an input voltage of zero. No 1s (all 0s) corresponds to the maximum negative input voltage. This is illustrated in a simplified way in Figure 12–20. For example, assume that 4096 1s occur during the interval when the input signal is a positive maximum. Since zero is the midpoint of the dynamic range of the input signal, 2048 1s occur during the interval when the input signal is zero. There are no 1s during the interval when the input signal is a negative maximum. For signal levels in between, the number of 1s is proportional to the level.

The Sigma-Delta ADC Functional Block Diagram

The basic block diagram in Figure 12–21 accomplishes the conversion illustrated in Figure 12–20. The analog input signal and the analog signal from the converted quantized bit stream from the DAC in the feedback loop are applied to the summation (Σ) point. The difference (Δ) signal out of the Σ is integrated, and the 1-bit ADC increases or decreases the number of 1s depending on the difference signal. This action attempts to keep the quantized signal that is fed back equal to the incoming analog signal. The 1-bit quantizer is essentially a comparator followed by a latch.

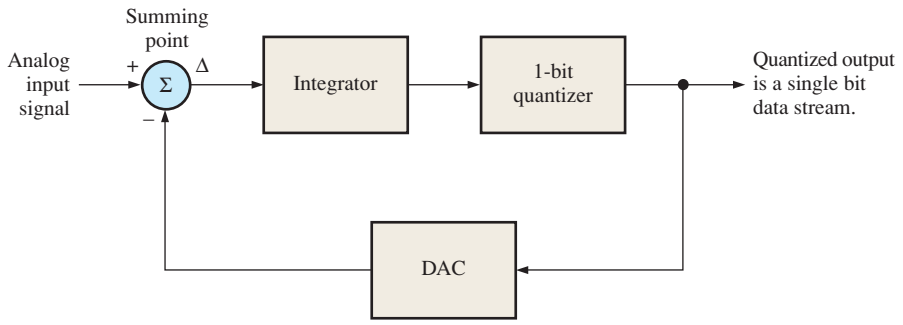


FIGURE 12–21 Partial functional block diagram of a sigma-delta ADC.

To complete the sigma-delta conversion process using one particular approach, the single bit data stream is converted to a series of binary codes, as shown in Figure 12–22. The counter counts the 1s in the quantized data stream for successive intervals. The code in the counter then represents the amplitude of the analog input signal for each interval. These codes are shifted out into the latch for temporary storage. What comes out of the latch is a series of *n*-bit codes, which completely represent the analog signal.

Testing Analog-to-Digital Converters

One method for testing ADCs is shown in Figure 12–23. A DAC is used as part of the test setup to convert the ADC output back to analog form for comparison with the test input.

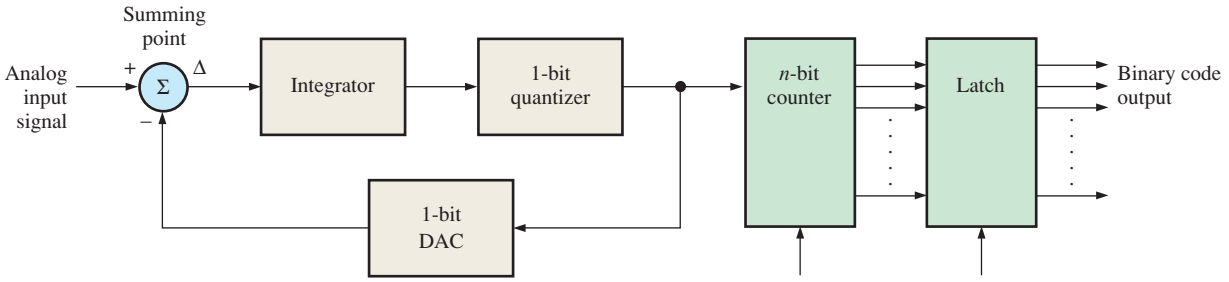


FIGURE 12–22 One type of sigma-delta ADC.

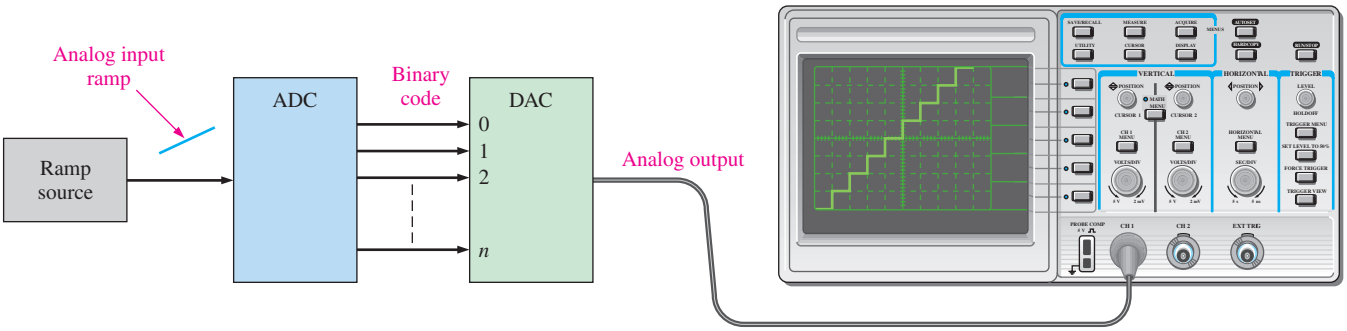


FIGURE 12–23 A method for testing ADCs.

A test input in the form of a linear ramp is applied to the input of the ADC. The resulting binary output sequence is then applied to the DAC test unit and converted to a stairstep ramp. The input and output ramps are compared for any deviation.

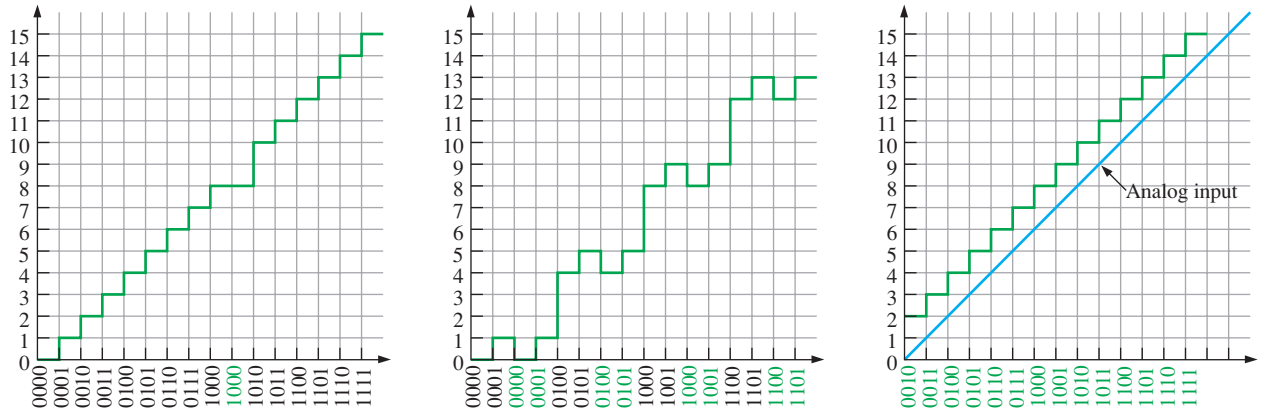
Analog-to-Digital Conversion Errors

Again, a 4-bit conversion is used to illustrate the principles. Let's assume that the test input is an ideal linear ramp.

Missing Code

The stairstep output in Figure 12–24(a) indicates that the binary code 1001 does not appear on the output of the ADC. Notice that the 1000 value stays for two intervals and then the output jumps to the 1010 value.

In a flash ADC, for example, a failure of one of the op-amp comparators can cause a missing-code error.



(a) Missing code (green) (b) Incorrect codes (green) (c) Offset

FIGURE 12–24 Illustrations of analog-to-digital conversion errors.

Incorrect Code

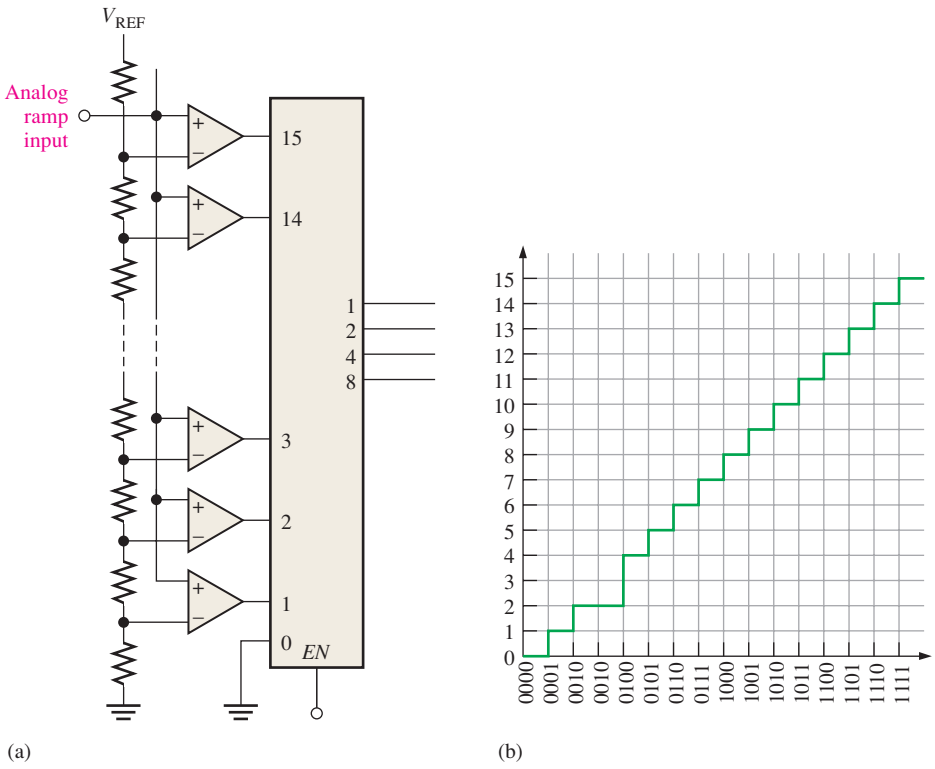
The stairstep output in Figure 12–24(b) indicates that several of the binary code words coming out of the ADC are incorrect. Analysis indicates that the 2^1-bit line is stuck in the LOW (0) state in this particular case.

Offset

Offset conditions are shown in 12–24(c). In this situation the ADC interprets the analog input voltage as greater than its actual value.

EXAMPLE 12–2

A 4-bit flash ADC is shown in Figure 12–25(a). It is tested with a setup like the one in Figure 12–23. The resulting reconstructed analog output is shown in Figure 12–25(b). Identify the problem and the most probable fault.



(a) (b)

FIGURE 12–25

Solution

The binary code 0011 is missing from the ADC output, as indicated by the missing step. Most likely, the output of comparator 3 is stuck in its inactive state (LOW).

Related Problem

Reconstruct the analog output in a test setup like in Figure 12–23 if the ADC in Figure 12–25(a) has comparator 8 stuck in the HIGH output state.

SECTION 12–2 CHECKUP

1. What is the fastest method of analog-to-digital conversion?
2. Which analog-to-digital conversion method produces a single-bit data stream?
3. Does the successive-approximation converter have a fixed conversion time?
4. Name two types of output errors in an ADC.

12–3 Methods of Digital-to-Analog Conversion

Digital-to-analog conversion is an important part of a digital processing system. Once the digital data has been processed, it is converted back to analog form. In this section, we will examine the theory of operation of two basic types of digital-to-analog converters (DACs) and learn about their performance characteristics.

After completing this section, you should be able to

- Explain the operation of a binary-weighted-input DAC
- Explain the operation of an *R*/2*R* ladder DAC
- Discuss resolution, accuracy, linearity, monotonicity, and settling time in a DAC
- Discuss the testing of DACs for nonmonotonicity, differential nonlinearity, low or high gain, and offset error

Binary-Weighted-Input Digital-to-Analog Converter

One method of digital-to-analog conversion uses a resistor network with resistance values that represent the binary weights of the input bits of the digital code. Figure 12–26 shows a 4-bit DAC of this type. Each of the input resistors will either have current or have no current, depending on the input voltage level. If the input voltage is zero (binary 0), the current is also zero. If the input voltage is HIGH (binary 1), the amount of current depends on the input resistor value and is different for each input resistor, as indicated in the figure.

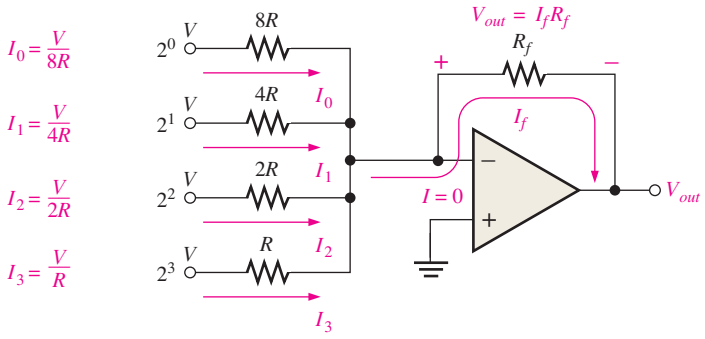


FIGURE 12–26 A 4-bit DAC with binary-weighted inputs.

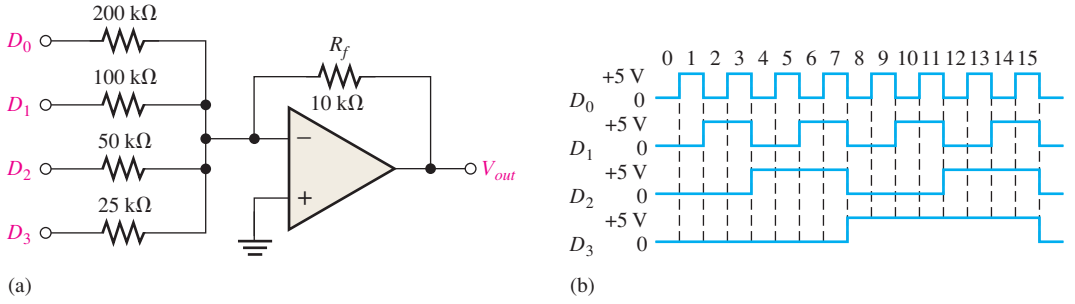
Since there is practically no current into the op-amp inverting (−) input, all of the input currents sum together and go through R_f. Since the inverting input is at 0 V (virtual ground), the drop across R_f is equal to the output voltage, so $V_{out} = I_f R_f$.

The values of the input resistors are chosen to be inversely proportional to the binary weights of the corresponding input bits. The lowest-value resistor (R) corresponds to the highest binary-weighted input (2^3). The other resistors are multiples of R (that is, $2R$, $4R$, and $8R$) and correspond to the binary weights 2^2, 2^1, and 2^0, respectively. The input currents are also proportional to the binary weights. Thus, the output voltage is proportional to the sum of the binary weights because the sum of the input currents is through R_f.

Disadvantages of this type of DAC are the number of different resistor values and the fact that the voltage levels must be exactly the same for all inputs. For example, an 8-bit converter requires eight resistors, ranging from some value of R to $128R$ in binary-weighted steps. This range of resistors requires tolerances of one part in 255 (less than 0.5%) to accurately convert the input, making this type of DAC very difficult to mass-produce.

EXAMPLE 12–3

Determine the output of the DAC in Figure 12–27(a) if the waveforms representing a sequence of 4-bit numbers in Figure 12–27(b) are applied to the inputs. Input D_0 is the least significant bit (LSB).



(a) (b)

FIGURE 12–27

Solution

First, determine the current for each of the weighted inputs. Since the inverting (−) input of the op-amp is at 0 V (virtual ground) and a binary 1 corresponds to +5 V, the current through any of the input resistors is 5 V divided by the resistance value.

$$I_0 = \frac{5\text{ V}}{200\text{ k}\Omega} = 0.025\text{ mA}$$

$$I_1 = \frac{5\text{ V}}{100\text{ k}\Omega} = 0.05\text{ mA}$$

$$I_2 = \frac{5\text{ V}}{50\text{ k}\Omega} = 0.1\text{ mA}$$

$$I_3 = \frac{5\text{ V}}{25\text{ k}\Omega} = 0.2\text{ mA}$$

Almost no current goes into the inverting op-amp input because of its extremely high impedance. Therefore, assume that all of the current goes through the feedback resistor R_f. Since one end of R_f is at 0 V (virtual ground), the drop across R_f equals the output voltage, which is negative with respect to virtual ground.

$$V_{out(D0)} = (10\text{ k}\Omega)(-0.025\text{ mA}) = -0.25\text{ V}$$
$$V_{out(D1)} = (10\text{ k}\Omega)(-0.05\text{ mA}) = -0.5\text{ V}$$
$$V_{out(D2)} = (10\text{ k}\Omega)(-0.1\text{ mA}) = -1\text{ V}$$
$$V_{out(D3)} = (10\text{ k}\Omega)(-0.2\text{ mA}) = -2\text{ V}$$

From Figure 12–27(b), the first binary input code is 0000, which produces an output voltage of 0 V. The next input code is 0001, which produces an output voltage of −0.25 V. The next code is 0010, which produces an output voltage of −0.5 V. The next code is 0011, which produces an output voltage of −0.25 V + −0.5 V = −0.75 V. Each successive binary code increases the output voltage by −0.25 V, so for this particular straight binary sequence on the inputs, the output is a stairstep waveform going from 0 V to −3.75 V in −0.25 V steps. This is shown in Figure 12–28.

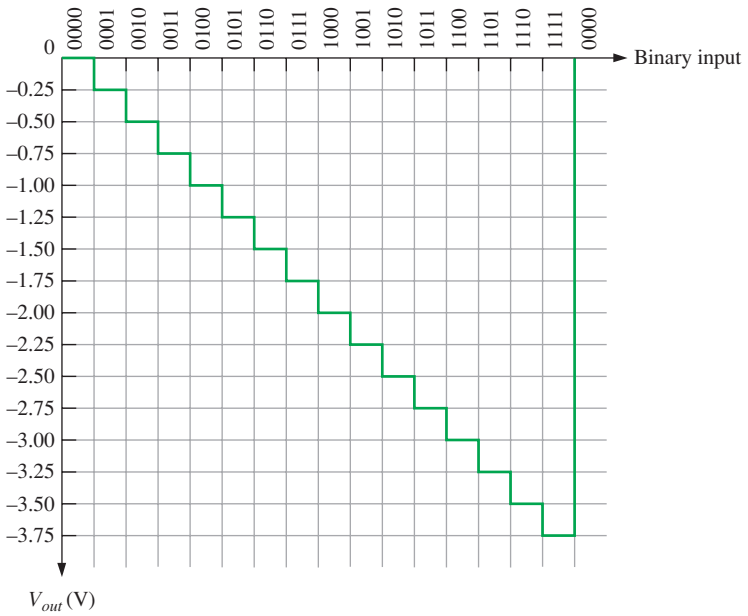


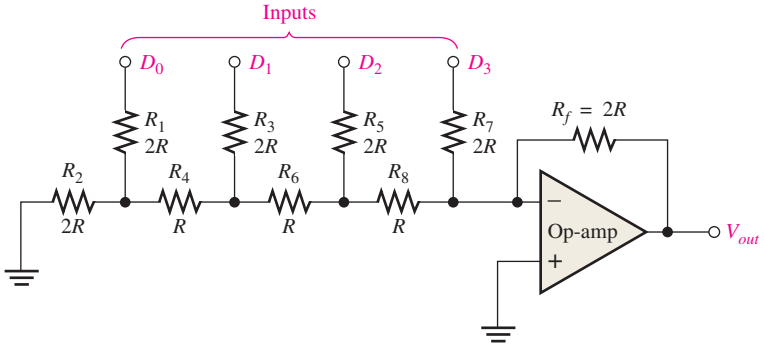
FIGURE 12–28 Output of the DAC in Figure 12–27.

Related Problem

Reverse the input waveforms to the DAC in Figure 12–27 (D_3 to D_0, D_2 to D_1, D_1 to D_2, D_0 to D_3) and determine the output.

The *R*/2*R* Ladder Digital-to-Analog Converter

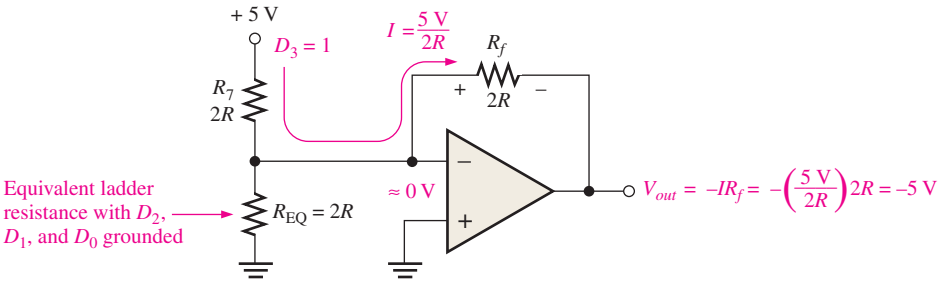
Another method of digital-to-analog conversion is the *R*/2*R* ladder, as shown in Figure 12–29 for four bits. It overcomes one of the problems in the binary-weighted-input DAC in that it requires only two resistor values.



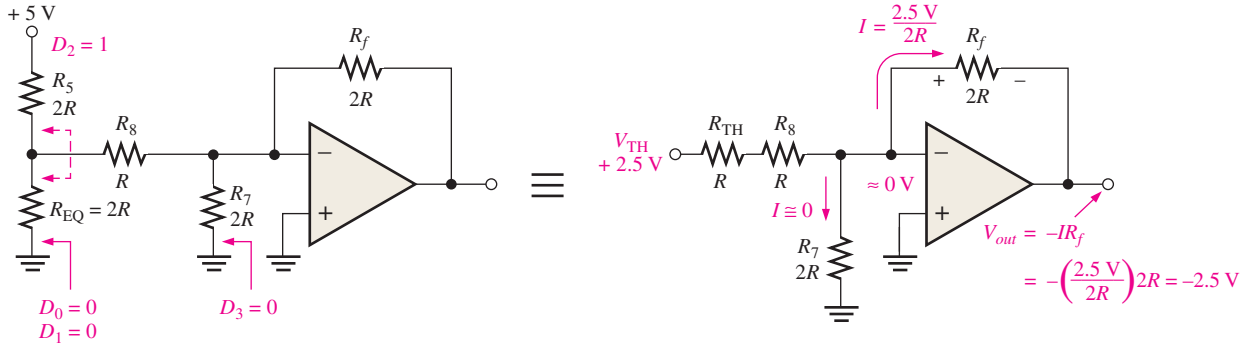
fg12_02900

FIGURE 12–29 An *R*/2*R* ladder DAC.

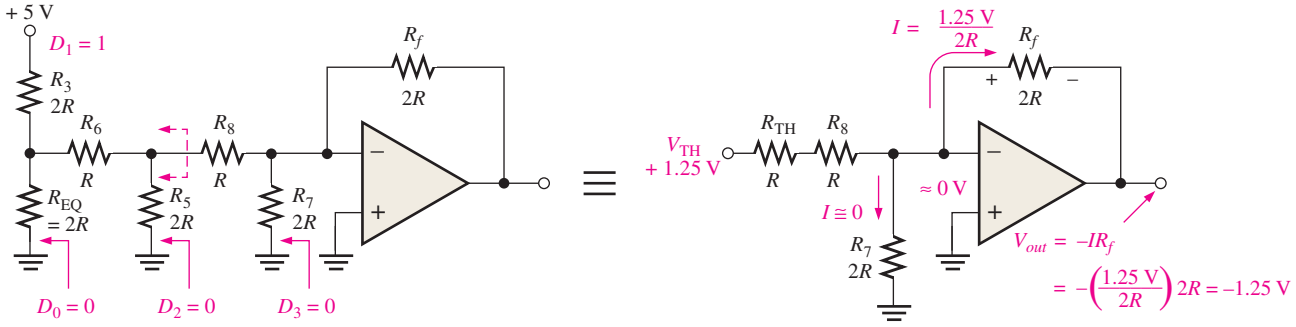
Start by assuming that the D_3 input is HIGH (+5 V) and the others are LOW (ground, 0 V). This condition represents the binary number 1000. A circuit analysis will show that this reduces to the equivalent form shown in Figure 12–30(a). Essentially no current goes



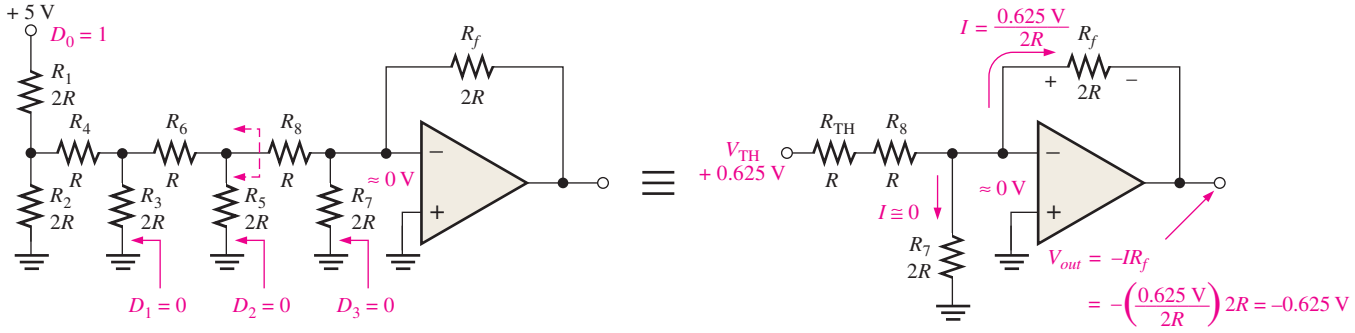
(a) Equivalent circuit for $D_3 = 1$, $D_2 = 0$, $D_1 = 0$, $D_0 = 0$



(b) Equivalent circuit for $D_3 = 0$, $D_2 = 1$, $D_1 = 0$, $D_0 = 0$



(c) Equivalent circuit for $D_3 = 0$, $D_2 = 0$, $D_1 = 1$, $D_0 = 0$



(d) Equivalent circuit for $D_3 = 0$, $D_2 = 0$, $D_1 = 0$, $D_0 = 1$

FIGURE 12–30 Analysis of the *R*/2*R* ladder DAC.

through the $2R$ equivalent resistance because the inverting input is at virtual ground. Thus, all of the current ($I = 5\text{ V}/2R$) through R_7 also goes through R_f, and the output voltage is -5 V. The operational amplifier keeps the inverting ($-$) input near zero volts ($\approx$0 V) because of negative feedback. Therefore, all current goes through R_f rather than into the inverting input.

Figure 12–30(b) shows the equivalent circuit when the D_2 input is at +5 V and the others are at ground. This condition represents 0100. If we thevenize* looking from R_8, we get 2.5 V in series with R, as shown. This results in a current through R_f of $I = 2.5\text{ V}/2R$, which gives an output voltage of −2.5 V. Keep in mind that there is no current into the op-amp inverting input and that there is no current through the equivalent resistance to ground because it has 0 V across it, due to the virtual ground.

Figure 12–30(c) shows the equivalent circuit when the D_1 input is at +5 V and the others are at ground. This condition represents 0010. Again thevenizing looking from R_8, you get 1.25 V in series with R as shown. This results in a current through R_f of $I = 1.25\text{ V}/2R$, which gives an output voltage of −1.25 V.

In part (d) of Figure 12–30, the equivalent circuit representing the case where D_0 is at +5 V and the other inputs are at ground is shown. This condition represents 0001. Thevenizing from R_8 gives an equivalent of 0.625 V in series with R as shown. The resulting current through R_f is $I = 0.625\text{ V}/2R$, which gives an output voltage of −0.625 V.

Notice that each successively lower-weighted input produces an output voltage that is halved, so that the output voltage is proportional to the binary weight of the input bits.

Performance Characteristics of Digital-to-Analog Converters

The performance characteristics of a DAC include resolution, accuracy, linearity, monotonicity, and settling time, each of which is discussed in the following list:

- *Resolution.* The resolution of a DAC is the reciprocal of the number of discrete steps in the output. This, of course, is dependent on the number of input bits. For example, a 4-bit DAC has a resolution of one part in $2^4 - 1$ (one part in fifteen). Expressed as a percentage, this is $(1/15)100 = 6.67\%$. The total number of discrete steps equals $2^n - 1$, where n is the number of bits. Resolution can also be expressed as the number of bits that are converted.
- *Accuracy.* Accuracy is derived from a comparison of the actual output of a DAC with the expected output. It is expressed as a percentage of a full-scale, or maximum, output voltage. For example, if a converter has a full-scale output of 10 V and the accuracy is ±0.1%, then the maximum error for any output voltage is $(10\text{ V})(0.001) = 10\text{ mV}$. Ideally, the accuracy should be no worse than ±1/2 of a least significant bit. For an 8-bit converter, the least significant bit is 0.39% of full scale. The accuracy should be approximately ±0.2%.
- *Linearity.* A linear error is a deviation from the ideal straight-line output of a DAC. A special case is an offset error, which is the amount of output voltage when the input bits are all zeros.
- *Monotonicity.* A DAC is **monotonic** if it does not take any reverse steps when it is sequenced over its entire range of input bits.
- *Settling time.* Settling time is normally defined as the time it takes a DAC to settle within ±1/2 LSB of its final value when a change occurs in the input code.

EXAMPLE 12–4

Determine the resolution, expressed as a percentage, of the following:

(a) an 8-bit DAC

(b) a 12-bit DAC

*Thevenin's theorem states that any circuit can be reduced to an equivalent voltage source in series with an equivalent resistance.