Connecting an OA with a feedback resistor R as shown in Fig. 12.12b results in an amplifier that acts as an inverting current-to-voltage amplifier. That is, the output voltage V_A is equal to the negative of the input current I multiplied by R. The input impedance to this amplifier is essentially 0 Ω: thus, when it is connected to an R-$2R$ ladder, the connecting point is virtually at ground potential. In this configuration, the R-$2R$ ladder will produce a current output I that is a binary weighted sum of the input digital levels. For instance, the MSB produces a current of $V/2R$. The second MSB produces a current of $V/4V$, and so on. But the OA multiplies these currents by $-R$, and thus V_A is

$$V_A = (-R)\left(\frac{V}{2R} + \frac{V}{4R} + \cdots\right) = -\frac{V}{2} - \frac{V}{4} - \cdots$$

This is exactly the same expression given in Eqs. (12.2) and (12.3) except for the sign. Thus the D/A converter in Fig. 12.12a and b will provide the same output voltage V_A except for sign. In Fig. 12.12a, the R-$2R$ ladder and OA are said to operate in a voltage mode, while the connection in Fig. 12.12b is said to operate in a current mode.

SELF-TEST

3. If the ladder in Example 12.4 is increased to 6 bits, what is the output voltage due to the sixth bit alone?
4. If the ladder in Example 12.4 is increased to 6 bits, what is its full-scale output voltage?

12.3 D/A CONVERTERS

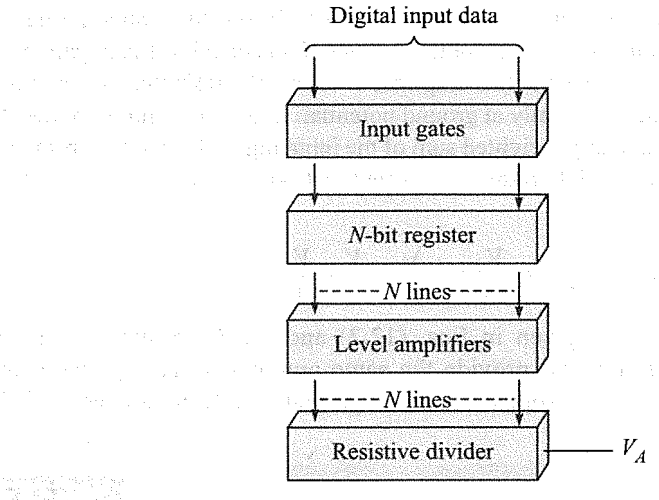
Either the resistive divider or the ladder can be used as the basis for a digital-to-analog (D/A) converter. It is in the resistive network that the actual translation from a digital signal to an analog voltage takes place. There is, however, the need for additional circuitry to complete the design of the D/A converter.

As an integral part of the D/A converter there must be a register that can be used to store the digital information. This register could be any one of the many types discussed in previous chapters. The simplest register is formed by use of *RS* flip-flops, with one flip-flop per bit. There must also be level amplifiers between the register and the resistive network to ensure that the digital signals presented to the network are all of the same level and are constant. Finally, there must be some form of gating on the input of the register such that the flip-flops can be set with the proper infomation from the digital system. A complete D/A converter in block-diagram form is shown in Fig. 12.13a.

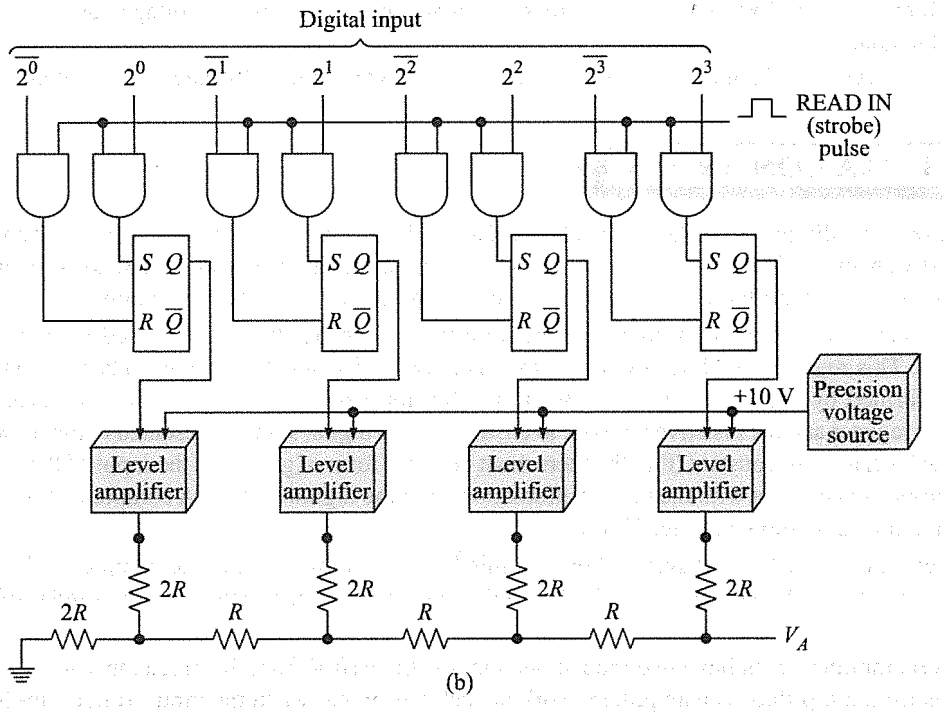
Let us expand on the block diagram shown in this Fig. 12.13a by drawing the complete schematic for a 4-bit D/A converter as shown in Fig. 12.13b. You will recognize that the resistor network used is of the ladder type.

The level amplifiers each have two inputs: one input is the +10 V from the precision voltage source, and the other is from a flip-flop. The amplifiers work in such a way that when the input from a flip-flop is high, the output of the amplifier is at +10 V. When the input from the flip-flop is low, the output is 0 V.

The four flip-flops form the register necessary for storing the digital information. The flip-flop on the right represents the MSB, and the flip-flop on the left represents the LSB. Each flip-flop is a simple *RS* latch and requires a positive level at the R or S input to reset or set it. The gating scheme for entering information into the register is straightforward and should be easy to understand. With this particular gating scheme, the flip-flops need not be reset (or set) each time new information is entered. When the READ IN line goes high, only



(a)



(b)

Fig. 12.13 4-bit D/A converter

one of the two gate outputs connected to each flip-flop is high, and the flip-flop is set or reset accordingly. Thus data are entered into the register each time the READ IN (strobe) pulse occurs. *D* flip-flops could be used in place of the *RS* flip-flops.

Multiple Signals

Quite often it is necessary to decode more than one signal—for example, the X and Y coordinates for a plotting board. In this event, there are two ways in which to decode the signals.

The first and most obvious method is simply to use one D/A converter for each signal. This method, shown in Fig. 12.14a, has the advantage that each signal to be decoded is held in its register and the analog output voltage is then held fixed. The digital input lines are connected in parallel to each converter. The proper converter is then selected for decoding by the select lines.

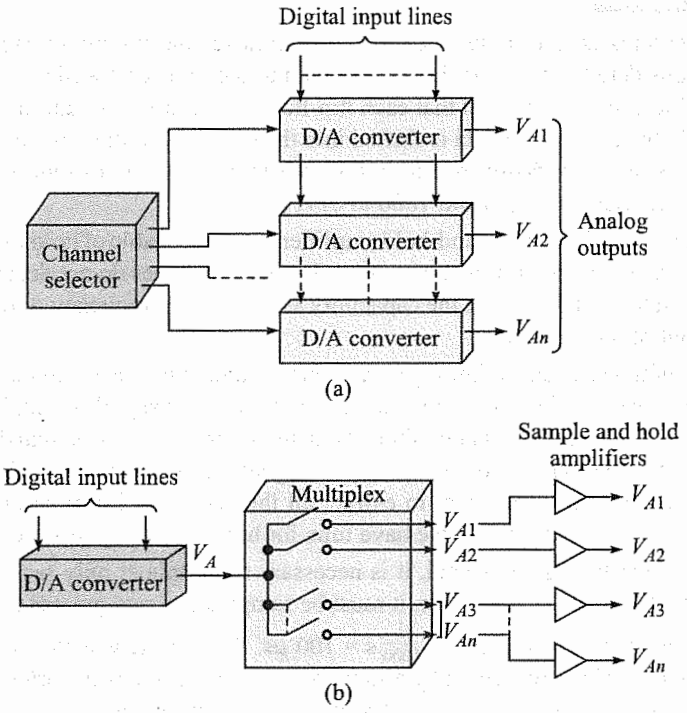


Fig. 12.14 **Decoding a number of signals: (a) Channel selection method, (b) Multiplex method**

The second method involves the use of only one D/A converter and switching its output. This is called *multiplexing*, and such a system is shown in Fig. 12.14b. The disadvantage here is that the analog output signal must be held between sampling periods, and the outputs must therefore be equipped with sample-and-hold amplifiers.

Sample and Hold Circuit

An OA connected as in Fig. 12.15a is a unity-gain noninverting voltage amplifier—that is, $V_o = V_i$. Two such OAs are used with a capacitor in Fig. 12.15b to form a sample-and-hold amplifier. When the switch is closed, the capacitor charges to the D/A converter output voltage. When the switch is opened, the capacitor holds the voltage level until the next sampling time. The operational amplifier provides a large input impedance so as not to discharge the capacitor appreciably and at the same time offers gain to drive external circuits.

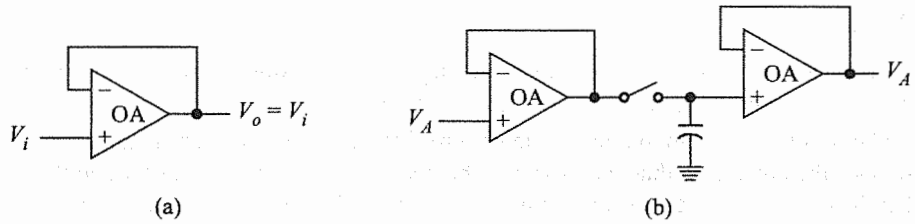


Fig. 12.15 **(a) Unity gain amplifier, (b) Sample-and-hold circuit**

When the D/A converter is used in conjunction with a multiplexer, the maximum rate at which the converter can operate must be considered. Each time data is shifted into the register, transients appear at the output of the converter. This is due mainly to the fact that each flip-flop has different rise and fall times. Thus a settling time must be allowed between the time data is shifted into the register and the time the analog voltage is read out. This settling time is the main factor in determining the maximum rate of multiplexing the output. The worst case is when all bits change (e.g., from 1000 to 0111).

Naturally, the capacitors on the sample-and-hold amplifiers are not capable of holding a voltage indefinitely; therefore, the sampling rate must be sufficient to ensure that these voltages do not decay appreciably between samples. The sampling rate is a function of the capacitors as well as the frequency of the analog signal which is expected at the output of the converter.

At this point, you might be curious to know just how fast a signal must be sampled in order to preserve its integrity. Common sense leads to the conclusion that the more often the signal is sampled, the less the sample degrades between samples. On the other hand, if too few samples are taken, the signal degrades too much (the sample-and-hold capacitors discharge too much), and the signal information is lost. We would like to reduce the sampling rate to the minimum necessary to extract all the necessary information from the signal. The solution to this problem involves more than we have time for here, but the results are easy enough to apply.

First, if the signal in question is sinusoidal, it is necessary to sample at only *twice* the signal frequency. For instance if the signal is a 5-kHz sine wave, it must be sampled at a rate greater than or equal to 10 kHz. In other words, a sample must be taken every $\frac{1}{1000}$ s = 100 μs. What if the waveform is not sinusoidal? Any waveform that is periodic can be represented by a summation of sine and cosine terms, with each succeeding term having a higher frequency. In this case, it will be necessary to sample at a rate equal to twice the highest frequency of interest.

D/A Converter Testing

Two simple but important tests that can be performed to check the proper operation of the D/A converter are the *steady-state accuracy test* and the *monotonicity test.*

The steady-state accuracy test involves setting a known digital number in the input register, measuring the analog output with an accurate meter, and comparing with the theoretical value.

Checking for monotonicity means checking that the output voltage increases regularly as the input digital signal increases. This can be accomplished by using a counter as the digital input signal and observing the analog output on an oscilloscope. For proper monotonicity, the output waveform should be a perfect staircase waveform, as shown in Fig. 12.16. The steps on the staircase waveform must be equally spaced and of the exact same amplitude. Missing steps, steps of different amplitude, or steps in a downward fashion indicate malfunctions.

The monotonicity test does not check the system for accuracy, but if the system passes the test, it is relatively certain that the converter error is less than 1 LSB. Converter accuracy and resolution are the subjects of the next section.

A D/A converter can be regarded as a logic block having numerous digital inputs and a single analog output as seen in Fig. 12.16b. It is interesting to compare this logic block with the potentiometer shown in Fig. 12.16c. The analog output voltage of the D/A converter is controlled by the digital input signals while the analog output voltage of the potentiometer is controlled by mechanical rotation of the potentiometer shaft. Considered in this fashion, it is easy to see how a D/A converter could be used to generate a voltage waveform (sawtooth, triangular, sinusoidal, etc.). It is, in effect, a digitally controlled voltage generator!

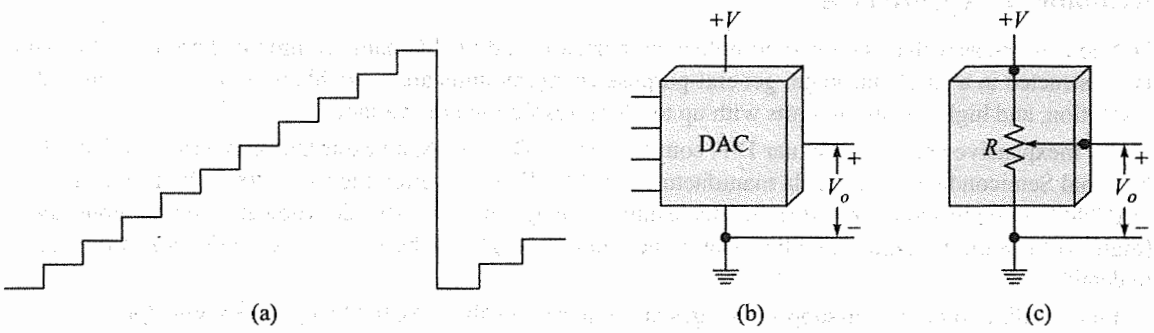


Fig. 12.16 **Correct output voltage waveform for monotonicity test**

Example 12.7 Suppose that in the course of a monotonicity check on the 4-bit converter in Fig. 12.13 the waveform shown in Fig. 12.17 is observed. What is the probable malfunction in the converter?

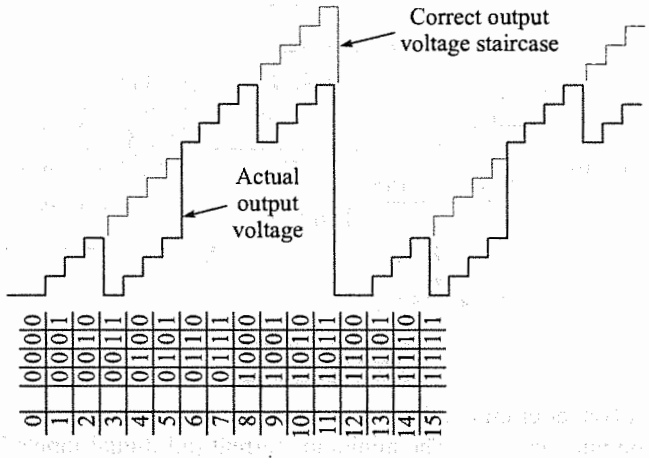


Fig. 12.17 **Irregular output voltage for Example 12.7**

Solution There is obviously some malfunction since the actual output waveform is not continuously increasing as it should be. The actual digital inputs are shown directly below the wave-form. Notice that the converter functions

correctly up to count 3. At count 4, however, the output should be 4 units in amplitude. Instead, it drops to 0. It remains 4 units below the correct level until it reaches count 8. Then, from count 8 to 11, the output level is correct. But again at count 12 the output falls 4 units below the correct level and remains there for the next four levels. If you examine the waveform carefully, you will note that the output is 4 units below normal during the time when the 2^2 bit is supposed to be high. This then suggests that the 2^2 bit is being dropped (i.e., the 2^2 input to the ladder is not being held high). This means that the 2^2-level amplifier is malfunctioning or the 2^2 AND gate is not operating properly. In any case, the monotonicity check has clearly shown that the second MSB is not being used and that the converter is not operating properly.

Available D/A Converters

D/A converters, as well as sample-and-hold amplifiers, are readily obtainable commercial products. Each unit is constructed in a single package; general-purpose economy units are available with 6-, 8-, 10-, and 12-bit resolution, and high-resolution units with up to 16-bit resolution are available.

An inexpensive and very popular D/A converter is the DAC0808, an 8-bit D/A converter available from National Semiconductor. Motorola manufactures an 8-bit D/A converter, the MC1508/1408. In Fig. 12.18, a DAC0808 is connected to provide a full-scale output voltage of $V_o = +10$ Vdc when all 8 digital inputs are 1s (high). If the 8 digital inputs are all 0s (low), the output voltage will be $V_o = 0$ Vdc. Let's look at this circuit in detail.

First of all, two dc power-supply voltages are required for the DAC0808: $V_{CC} = +5$ Vdc and $V_{EE} = -15$ Vdc. The 0.1-μF capacitor is to prevent unwanted circuit oscillations, and to isolate any variations in V_{EE}. Pin 2 is ground (GND), and pin 15 is also referenced to ground through a resistor.

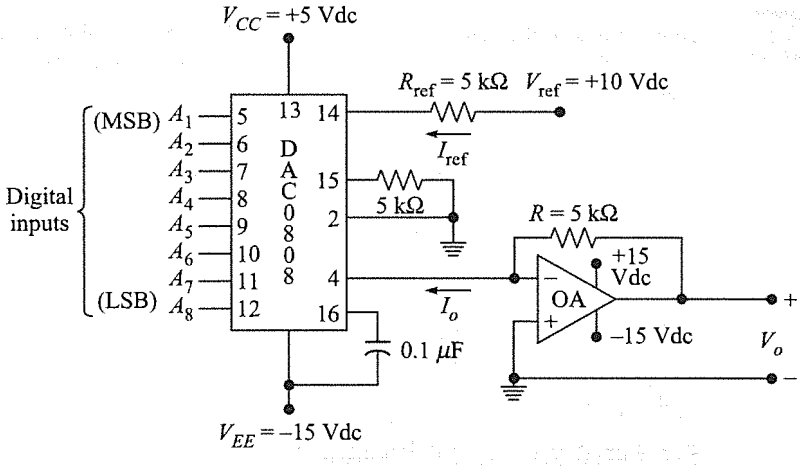


Fig. 12.18

The output of the D/A converter on pin 4 has a very limited voltage range (+0.5 to –0.6 V). Rather, it is designed to provide an output current I_o. The minimum current (all digital inputs low) is 0.0 mA, and the maximum current (all digital inputs high), is I_{ref}. This reference current is established with the resistor at pin 14 and the reference voltage as

$$I_{ref} = V_{ref}/R_{ref} \tag{12.4}$$

The D/A converter output current I_o is given as

$$I_o = I_{\text{ref}}\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \cdots + \frac{A8}{256}\right) \tag{12.5}$$

where $A_1, A_2, A_3, \ldots, A_8$ are the digital input levels (1 or 0).

The OA is connected as a current-to-voltage converter, and the output voltage is given as

$$V_o = I_o \times R \tag{12.6}$$

Substituting Eqs. (12.4) and (12.5) into Eq. (12.6),

$$V_o = V_{\text{ref}} / R_{\text{ref}} \times \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \cdots + \frac{A8}{256}\right) \times R \tag{12.7}$$

If we set the OA feedback resistor R equal to R_{ref}, then

$$V_o = V_{\text{ref}}\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \cdots + \frac{A8}{256}\right) \tag{12.8}$$

Let's try out Eq. (12.8). Suppose all digital inputs are 0s (all low). Then

$$V_o = V_{\text{ref}} \times \left(\frac{0}{2} + \frac{0}{4} + \frac{0}{8} + \cdots + \frac{0}{256}\right)$$
$$= V_{\text{ref}} \times 0 = 0.0 \text{ Vdc}$$

Now, suppose all digital inputs are 1s (all high). Then

$$V_o = V_{\text{ref}} \times \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{256}\right)$$
$$= (V_{\text{ref}}) \times \left(\frac{255}{256}\right) = 0.996 \times V_{\text{ref}}$$

Since V_{ref} in Fig. 12.18 is +10 Vdc, the output voltage is seen to have a range between 0.0 and +9.96 Vdc. It doesn't quite reach +10 Vdc, but this is characteristic of this type of circuit. This circuit is essentially the current-mode operation discussed in the previous section and illustrated in Fig. 12.12b.

Example 12.8 In Fig. 12.16, $A1$ is high, $A2$ is high, $A5$ is high and $A7$ is high. The other digital inputs are all low. What is the output voltage V_o?

Solution

$$V_o = 10 \times \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{32} + \frac{1}{128}\right) = 10 \times 0.789 = 7.89 \text{ V}$$

SELF-TEST

5. What is a monotonicity test?
6. What would be the full-scale output voltage in Fig. 12.18 if V_{ref} were changed to +5 Vdc?

12.4 D/A ACCURACY AND RESOLUTION

Two very important aspects of the D/A converter are the resolution and the accuracy of the conversion. There is a definite distinction between the two, and you should clearly understand the differences.

The accuracy of the D/A converter is primarily a function of the accuracy of the precision resistors used in the ladder and the precision of the reference voltage supply used. Accuracy is a measure of how close the actual output voltage is to the theoretical output value.

For example, suppose that the theoretical output voltage for a particular input should be +10 V. An accuracy of 10 percent means that the actual output voltage must be somewhere between +9 and +11 V. Similarly, if the actual output voltage were somewhere between +9.9 and +10.1 V, this would imply an accuracy of 1 percent.

Resolution, on the other hand, defines the smallest increment in voltage that can be discerned. Resolution is primarily a function of the number of bits in the digital input signal; that is, the smallest increment in output voltage is determined by the LSB.

In a 4-bit system using a ladder, for example, the LSB has a weight of $\frac{1}{16}$. This means that the smallest increment in output voltage is $\frac{1}{16}$ of the input voltage. To make the arithmetic easy, let us assume that this 4-bit system has input voltage levels of +16 V. Since the LSB has a weight of $\frac{1}{16}$, a change in the LSB results in a change of 1 V in the output. Thus the output voltage changes in steps (or increments) of 1 V. The output voltage of this converter is then the staircase shown in Fig. 12.16 and ranges from 0 to +15V in 1-V increments. This converter can be used to represent analog voltages from 0 to +15 V, but it cannot resolve voltages into increments smaller than 1 V. If we desired to produce +4.2 V using this converter, therefore, the actual output voltage would be +4.0 V. Similarly, if we desired a voltage of +7.8 V, the actual output voltage would be +8.0 V. It is clear that this converter is not capable of distinguishing voltages finer than 1 V, which is the resolution of the converter.

If we wanted to represent voltages to a finer resolution, we would have to use a converter with more input bits. As an example, the LSB of a 10-bit converter has a weight of $\frac{1}{1024}$. Thus the smallest incremental change in the output of this converter is approximately $\frac{1}{1000}$ of the full-scale voltage. If this converter has a +10-V full-scale output, the resolution is approximately $+10 \times \frac{1}{1000} = 10$ mV. This converter is then capable of representing voltages to within 10 mV.

Example 12.9 What is the resolution of a 9-bit D/A converter which uses a ladder network? What is this resolution expressed as a percent? If the full-scale output voltage of this converter is +5 V, what is the resolution in volts?

Solution The LSB in a 9-bit system has a weight of $\frac{1}{512}$. Thus this converter has a resolution of 1 part in 512. The resolution expressed as a percentage is $\frac{1}{512} \times 100$ percent $\cong$ 0.2 percent. The voltage resolution is obtained by multiplying the weight of the LSB by the full-scale output voltage. Thus the resolution in volts is $\frac{1}{512} \times 5 \cong 10$ mV.

Example 12.10 How many bits are required at the input of a convener if it is necessary to resolve voltages to 5 mV and the ladder has +10 V full scale?

Solution The LSB of an 11-bit system has a resolution of $\frac{1}{2048}$. This would provide a resolution at the output of $\frac{1}{2048} \times +10 \cong +5$ mV.

It is important to realize that resolution and accuracy in a system should be compatible. For example, in the 4-bit system previously discussed, the resolution was found to be 1 V. Clearly it would be unjustifiable to construct such a system to an accuracy of 0.1 percent. This would mean that the system would be accurate to 16 mV but would be capable of distinguishing only to the nearest 1 V.

Similarly, it would be wasteful to construct the 11-bit system described in Example 12.19 to an accuracy of only 1 percent. This would mean that the output voltage would be accurate only to 100 mV, whereas it is capable of distinguishing to the nearest 5 mV.

SELF-TEST

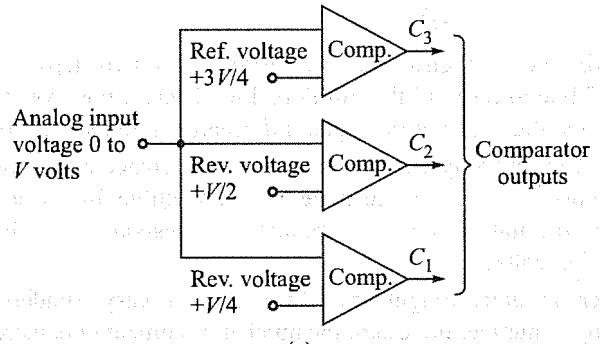
7. What is the resolution of the DAC0808 in Fig. 12.18?

12.5 A/D CONVERTER—SIMULTANEOUS CONVERSION

The process of converting an analog voltage into an equivalent digital signal is known as *analog-to-digital (A/D) conversion*. This operation is somewhat more complicated than the converse operation of D/A conversion. A number of different methods have been developed, the simplest of which is probably the simultaneous method. This is also known as *A/D converter, flash type,* the reason for which will be clear shortly.

The simultaneous method of A/D conversion is based on the use of a number of comparator circuits. One such system using three comparator circuits is shown in Fig. 12.19 below. The analog signal to be digitized serves as one of the inputs to each comparator. The second input is a standard reference voltage. The reference voltages used are $+V/4$, $+V/2$, and $+3V/4$. The system is then capable of accepting an analog input voltage between 0 and $+V$.

If the analog input signal exceeds the reference voltage to any comparator, that comparator turns on. (Let's assume that this means that the output of the comparator goes high.) Now, if all the comparators are off, the analog input signal must be between 0 and $+V/4$. If C_1 is high (comparator C_1 is on) and C_2 and C_3 are low, the input must be between $+V/4$ and $+V/2$ V. If C_1 and C_2 are high while C_3 is low, the input must be between $+V/2$ and $+3V/4$. Finally, if all comparator outputs are high, the input signal must be between $+3V/4$ and $+V$. The comparator output levels for the various ranges of input voltages are summarized in Fig. 12.19.



(a)

Input voltage	Comparator output		
	C_1	C_2	C_3
0 to $+V/4$	Low	Low	Low
$+V/4$ to $+V/2$	High	Low	Low
$+V/2$ to $+3V/4$	High	High	Low
$+3V/4$ to $+V$	High	High	High

(b)

Fig. 12.19 **Simultaneous A/D conversion: (a) Logic diagram, (b) Comparator outputs for input voltage ranges**

Examination of Fig. 12.19 reveals that there are four voltage ranges that can be detected by this converter. Four ranges can be effectively discerned by two binary digits (bits). The three comparator outputs can then be fed into a coding network to provide 2 bits which are equivalent to the input analog voltage. The bits of the coding network can then be entered into a flip-flop register for storage. The complete block diagram for such an A/D converter is shown in Fig. 12.20.

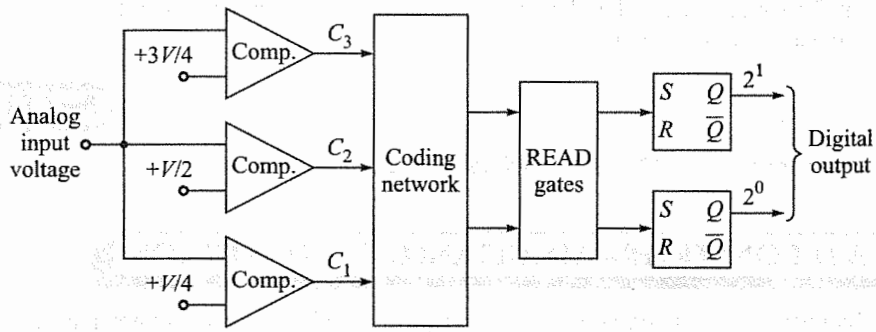


Fig. 12.20 **2-bit simultaneous A/D converter**

In order to gain a clear understanding of the operation of the simultaneous A/D converter, let us investigate the 3-bit converter shown in Fig. 12.21a. Notice that in order to convert the input signal to a digital signal having 3 bits, it is necessary to have seven comparators (this allows a division of the input into eight ranges). For the 2-bit converter, remember that three comparators were necessary for defining four ranges. In general, it can be said that $2^n - 1$ comparators are required to convert to a digital signal that has n bits. Some of the comparators have inverters at their outputs since both C and $\overline{C}$ are needed for the encoding matrix.

The encoding matrix must accept seven input levels and encode them into a 3-bit binary number (having eight possible states). Operation of the encoding matrix can be most easily understood by examination of the table of outputs in Fig. 12.22.

The 2^2 bit is easiest to determine since it must be high (the 2^2 flip-flop must be set) whenever C_4 is high.

The 2^1 line must be high whenever C_2 is high and $\overline{C}_4$ is high, or whenever C_6 is high. In equation form, we can write $2^1 = C_2\overline{C}_4 + C_6$.

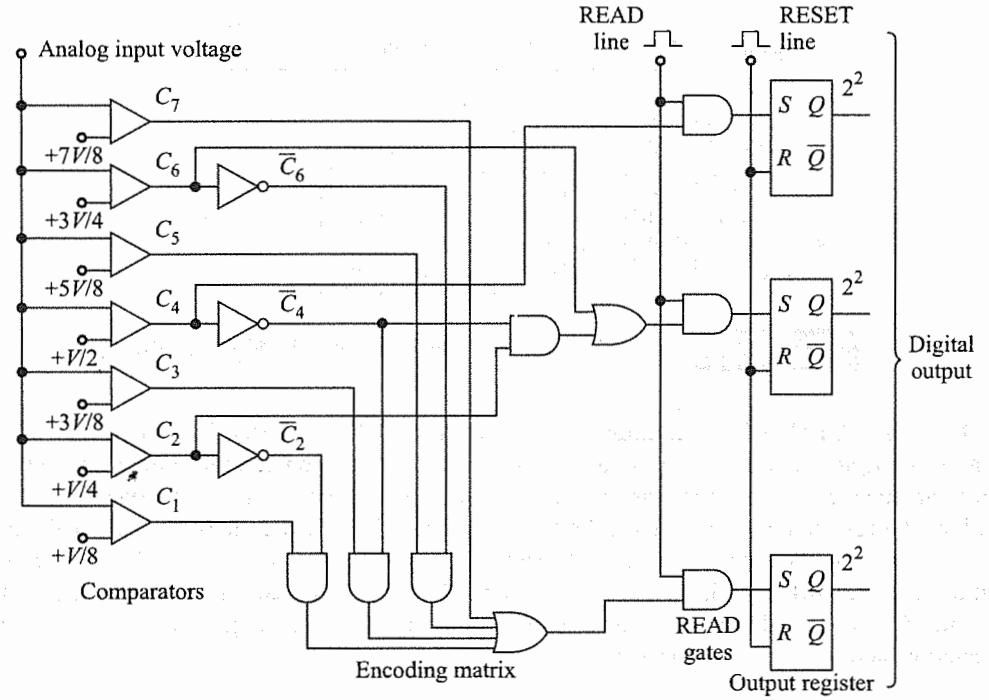
The logic equation for the 2^0 bit can be found in a similar manner; it is

$$2^0 = C_1\overline{C}_2 + C_3\overline{C}_4 + C_5\overline{C}_6 + C_7$$

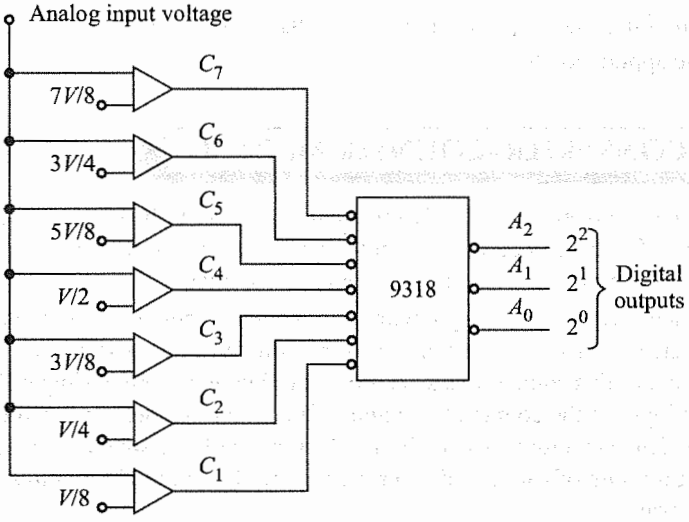
The transfer of data from the encoding matrix into the register must be carried out in two steps. First, a positive reset pulse must appear on the RESET line to reset all the flip-flops low. Then, a positive READ pulse allows the proper READ gates to go high and thus transfer the digital information into the flip-flops.

Interestingly, a convenient application for a 9318 priority encoder is to use it to replace all the digital logic as shown in Fig. 12.21b. Of course, the inputs $C_1, C_2, \ldots, C_7$ must be TTL-compatible. In essence, the output of the 9318 is a digital number that reflects the highest-order zero input; this corresponds to the lowest reference voltage that still exceeds the input analog voltage.

The construction of a simultaneous A/D converter is quite straightforward and relatively easy to understand. However, as the number of bits in the desired digital number increases, the number of comparators increases very rapidly ($2^n - 1$), and the problem soon becomes unmanageable. Even though this method is simple and is capable of extremely fast conversion rates, here are preferable methods for digitizing numbers having more than 3 or 4 bits. Because it is so fast, this type of converter is frequently called a *flash converter*.



(a)



(b)

Fig. 12.21 **3-bit simultaneous A/D converter: (a) Logic diagram, (b) Using a 9318 priority encoder**

Input voltage	Comparator for level							Binary output		
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	2^2	2^1	2^0
0 to $V/8$	Low	Low	Low	Low	Low	Low	Low	0	0	0
$V/8$ to $V/4$	High	Low	Low	Low	Low	Low	Low	0	0	1
$V/4$ to $3V/8$	High	High	Low	Low	Low	Low	Low	0	1	0
$3V/8$ to $V/2$	High	High	High	Low	Low	Low	Low	0	1	1
$V/2$ to $5V/8$	High	High	High	High	Low	Low	Low	1	0	0
$5V/8$ to $3V/4$	High	High	High	High	High	Low	Low	1	0	1
$3V/4$ to $7V/8$	High	High	High	High	High	High	Low	1	1	0
$7V/8$ to V	High	High	High	High	High	High	High	1	1	1

Fig. 12.22 **Logic table tor the converter in Fig. 12.19(a)**

The Motorola MC10319 is an example of an 8-bit flash A/D converter. The input has 256 parallel comparators connected to a precision voltage divider network. The comparator outputs are fed to latches and then to an encoder network that captures the digital signal in Gray code. Gray code is used to ensure that small input errors do not result in large digital signal errors. The Gray code is then decoded into straight binary and presented to the outputs, which are tri-state TTL = compatible. The flash A/D converter is capable of operation with a 25-MHz clock! It comes in a 24-pin DIP and requires two dc supply voltages—typically +5 Vdc and –5 Vdc. Possible applications include radar signal processing, video displays, high-speed instrumentation, and television broadcasting.

SELF-TEST

8. Why is a simultaneous A/D converter called a flash converter?
9. What is one application for a flash converter?

12.6 A/D CONVERTER–COUNTER METHOD

A higher-resolution A/D converter using only one comparator could be constructed if a variable reference voltage were available. This reference voltage could then be applied to the comparator, and when it became equal to the input analog voltage, the conversion would be complete.

To construct such a converter, let us begin with a simple binary counter. The digital output signals will be taken from this counter, and thus we want it to be an n-bit counter, where n is the desired number of bits. Now let us connect the output of this counter to a standard binary ladder to form a simple D/A converter. If a clock is now applied to the input of the counter, the output of the binary ladder is the familiar staircase waveform shown in Fig. 12.16. This waveform is exactly the reference voltage signal we would like to have for the comparator! With a minimum of gating and control circuitry, this simple D/A converter can be changed into the desired A/D converter.

Figure 12.23 shows the block diagram for a counter-type A/D converter. The operation of the counter is as follows. First, the counter is reset to all 0s. Then, when a convert signal appears on the START line, the gate opens and clock pulses are allowed to pass through to the input of the counter. The counter advances through its normal binary count sequence, and the staircase waveform is generated at the output of

the ladder. This waveform is applied to one side of the comparator, and the analog input voltage is applied to the other side. When the reference voltage equals (or exceeds) the input analog voltage, the gate is closed, the counter stops, and the conversion is complete. The number stored in the counter is now the digital equivalent of the analog input voltage.

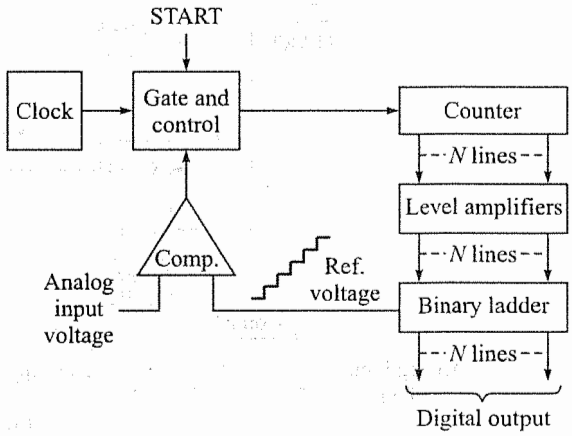


Fig. 12.23 **Counter type A/D converter**

Notice that this converter is composed of a D/A converter (the counter, level amplifiers, and the binary ladder), one comparator, a clock, and the gate and control circuitry. This can really be considered as a closed-loop control system. An error signal is generated at the output of the comparator by taking the difference between the analog input signal and the feedback signal (staircase reference voltage). The error is detected by the control circuit, and the clock is allowed to advance the counter. The counter advances in such a way as to reduce the error signal by increasing the feedback voltage. When the error is reduced to zero, the feedback voltage is equal to the analog input signal, the control circuitry stops the clock from advancing the counter, and the system comes to rest.

The counter-type A/D converter provides a very good method for digitizing to a high resolution. This method is much simpler than the simultaneous method for high resolution, but the conversion time required is longer. Since the counter always begins at zero and counts through its normal binary sequence, as many as 2^n counts may be necessary before conversion is complete. The average conversion time is, of course, $2^n/2$ or 2^{n-1} counts.

The counter advances one count for each cycle of the clock, and the clock therefore determines the conversion rate. Suppose, for example, that we have a 10-bit converter. It requires 1024 clock cycles for a full-scale count. If we are using a 1-MHz clock, the counter advances 1 count every microsecond. Thus, to count full scale requires $1024 \times 10^{-6} = 1.024$ ms. The converter reaches one-half full scale in half this time, or in 0.512 ms. The time required to reach one-half full scale can be considered the *average* conversion time for a large number of conversions.

Example 12.11 Suppose that the converter shown in Fig. 12.23 is an 8-bit converter driven by a 500-kHz clock. Find (a) the maximum conversion time; (b) the average conversion time; (c) the maximum conversion rate.

Solution

(a) An 8-bit converter has a maximum of $2^8 = 256$ counts. With a 500-kHz clock, the counter advances at the rate of 1 count each 2 μs. To advance 256 counts requires $256 \times 2 \times 10^{-6} = 512 \times 10^{-6} = 512$ μs.

(b) The average conversion time is one-half the maximum conversion time. Thus it is $1/2 \times 0.512 \times 10^{-3} = 0.256$ ms.

(c) The maximum conversion rate is determined by the longest conversion time. Since the converter has a maximum conversion time of 0.512 ms, it is capable of making at least $1/(0.512 \times 10^{-3}) \cong 1953$ conversions per second.

Figure 12.24 shows one method of implementing the control circuitry for the converter shown in Fig. 12.23. The waveforms for one conversion are also shown. A conversion is initiated by the receipt of a START signal.

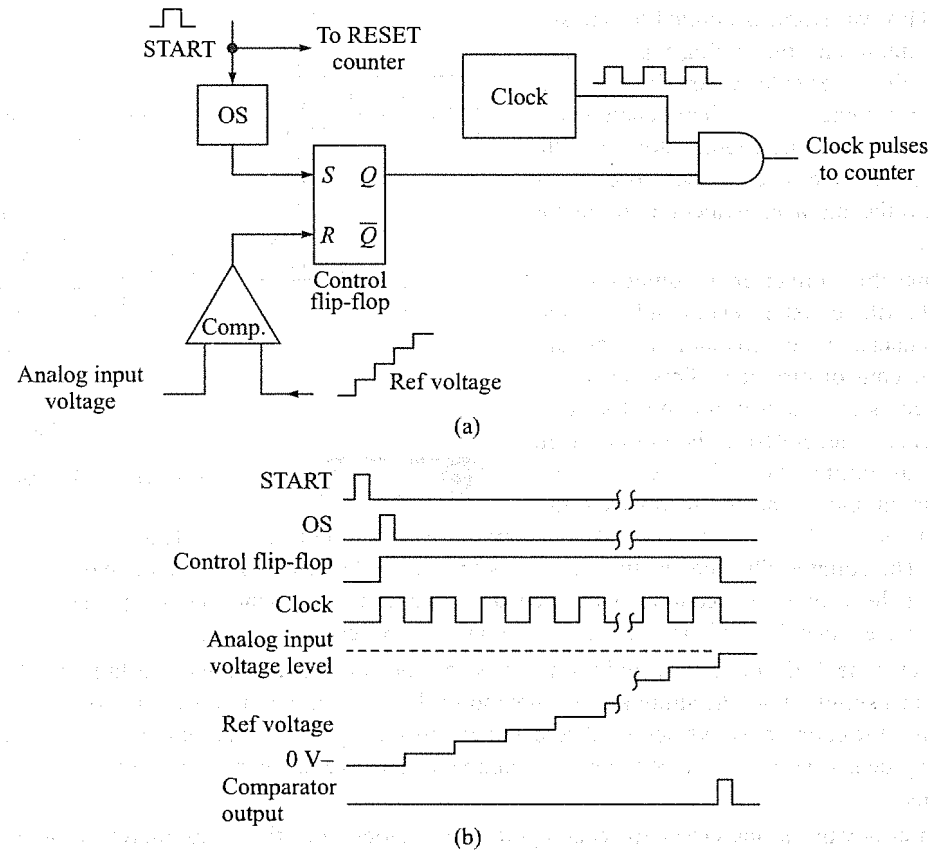


Fig. 12.24 Control of the A/D converter in Fig. 12.21

The positive edge of the START pulse is used to reset all the flip-flops in the counter and to trigger the one-shot. The output of the one-shot sets the control flip-flop, which makes the AND gate true and allows clock pulses to advance the counter.

The delay between the RESET pulse to the flip-flops and the beginning of the clock pulses (ensured by the one-shot) is to ensure that all flip-flops are reset before counting begins. This is a definite attempt to avoid any racing problems.

With the control flip-flop set, the counter advances through its normal count sequence until the staircase voltage from the ladder is equal to the analog input voltage. At this time, the comparator output changes state, generating a positive pulse which resets the control flip-flop. Thus the AND gate is closed and counting ceases. The counter now holds a digital number which is equivalent to the analog input voltage. The converter remains in this state until another conversion signal is received.

If a new start signal is generated immediately after each conversion is completed, the converter will operate at its maximum rate. The converter could then be used to digitize a signal as shown in Fig. 12.25a Notice that the conversion times in digitizing this signal are not constant but depend on the amplitude of the input signal. The analog input signal can be reconstructed from the digital information by drawing straight

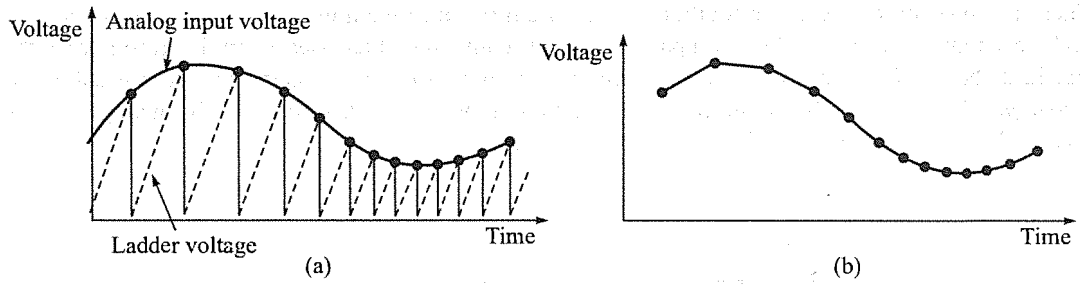


Fig. 12.25 **(a) Digitizing an analog voltage. (b) Reconstructed signal from the digital data.**

lines from each digitized point to the next. Such a reconstruction is shown in Fig. 12.25b; it is, indeed, a reasonable representation of the original input signal. In this case, it is important to note that the conversion times are smaller than the transient time of the input waveform.

On the other hand, if the transient time of the input waveform approaches the conversion time, the reconstructed output signal is not quite so accurate. Such a situation is shown in Fig. 12.26a and b. In this case, the input waveform changes at a rate faster than the converter is capable of recognizing. Thus the need for reducing conversion time is apparent.

SELF-TEST

10. The A/D converter in Fig. 12.23 has 8 bits and is driven by a 2-MHz clock. What is the maximum conversion time?
11. What is the average conversion time for the converter in question 10?

12.7 CONTINUOUS A/D CONVERSION

An obvious method for speeding up the conversion of the signal as shown in Fig. 12.26 is to eliminate the need for resetting the counter each time a conversion is made. If this were done. the counter would not begin at zero each time, but instead would begin at the value of the last converted point. This means that the counter would have to be capable of counting either up or down. This is no problem; we are already familiar with the operation of up-down counters.

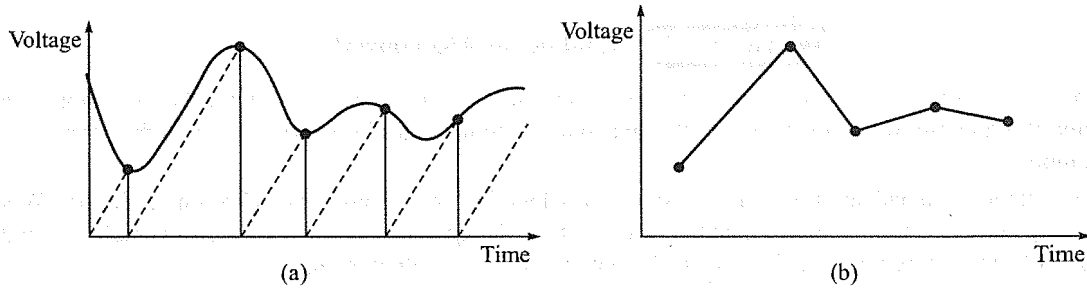


Fig. 12.26 **(a) Digitizing an analog voltage, (b) Reconstructed signal from the digital data**

There is, however, the need for additional logic circuitry, since we must decide whether to count up or down by examining the output of the comparator. An A/D converter which uses an up-down counter is shown in Fig. 12.27 below. This method is known as *continuous conversion*, and thus the converter is called a *continuous-type A/D converter*. Since the converter's digital output always tries to track the analog input to the converter, this is also known as *A/D converter-tracking type*.

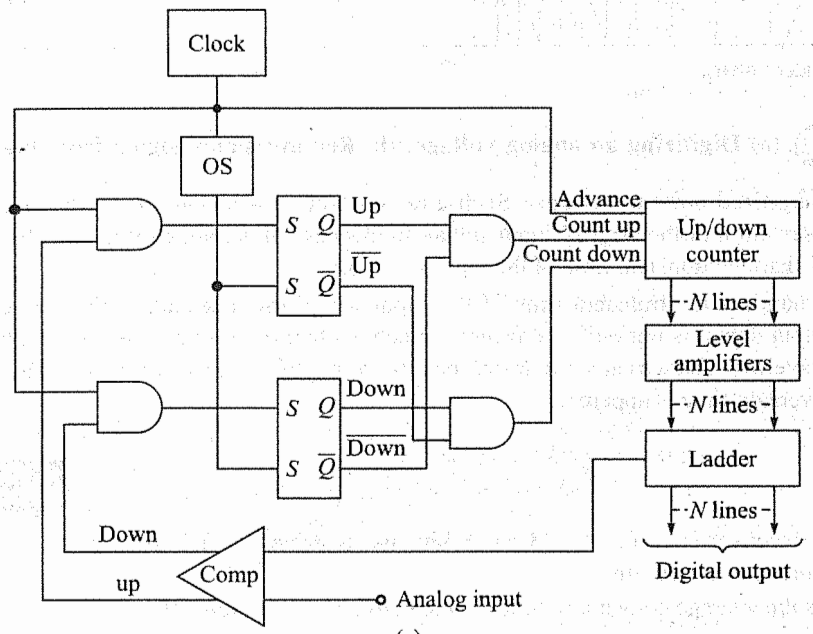


Fig. 12.27 Continuous A/D converter

The D/A portion of this converter is the same as those previously discussed, with the exception of the counter. It is an up-down counter and has the up and down count control lines in addition to the advance line at its input.

The output of the ladder is fed into a comparator which has two outputs instead of one as before. When the analog voltage is more positive than the ladder output, the *up* output of the comparator is high. When the analog voltage is more negative than the ladder output, the *down* output is high.

If the *up* output of the comparator is high, the AND gate at the input of the *up* flip-flop is open, and the first time the clock goes positive, the *up* flip-flop is set. If we assume for the moment that the *down* flip-flop is reset, the AND gate which controls the *count-up* line of the counter will be true and the counter will advance one count. The counter can advance only one count since the output of the one-shot resets both the *up* and the *down* flip-flops just after the clock goes low. This can then be considered as one count-up conversion cycle.

Notice that the AND gate which controls the *count-up* line has inputs of *up* and $\overline{down}$. Similarly, the count-down line AND gate has inputs of *down* and $\overline{up}$. This could be considered an exclusive-OR arrangement and ensures that the count-down and count-up lines cannot both be high at the same time.

As long as the *up* line out of the comparator is high, the converter continues to operate one conversion cycle at a time. At the point where the ladder voltage becomes more positive than the analog input voltage, the *up* line of the comparator goes low and the *down* line goes high. The converter then goes through a count-down conversion cycle. At this point, the ladder voltage is within 1 LSB of the analog voltage, and the converter oscillates about this point. This is not desirable since we want the converter to cease operation and not jump around the final value. The trick here is to adjust the comparator such that its outputs do not change at the same time.

We can accomplish this by adjusting the comparator such that the *up* output will not go high unless the ladder voltage is more than 1/2 LSB below the analog voltage. Similarly, the *down* output will not go high unless the ladder voltage is more than 1/2 LSB above the analog voltage. This is called *centering on the LSB* and provides a digital output which is within 1/2 LSB.

A waveform typical of this type of converter is shown in Fig. 12.28. You can see that this converter is capable of following input voltages that change at a much faster rate.

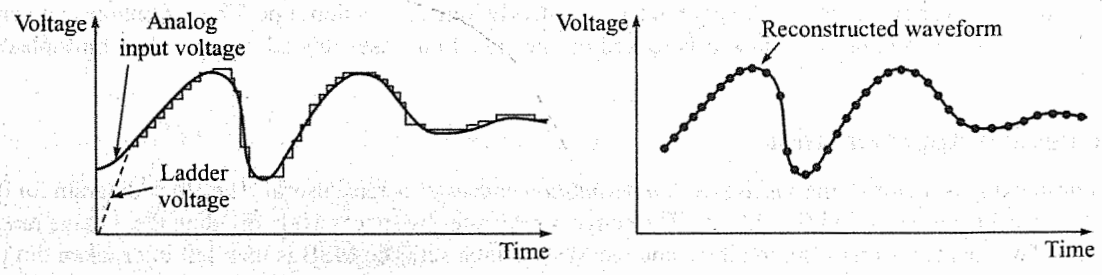


Fig. 12.28 **Continuous A/D conversion**

Example 12.12 Quite often, additional circuitry is added to a continuous converter to ensure that it cannot count off scale in either direction. For example, if the counter contained all 1s, it would be undesirable to allow it to progress through a count-up cycle, since the next count would advance it to all 0s. We would like to design the logic necessary to prevent this.

Solution The two limit points which must be detected are all 1s and all 0s in the counter. Suppose that we construct an AND gate having the 1 sides of all the counter flip-flops as its inputs. The output of this gate will be true whenever the counter contains all 1s. If the gate is then connected to the reset side of the *up* flip-flop, the counter will be unable to count beyond all 1s.

Similarly, we might construct an AND gate in which the inputs are the 0 sides of all the counter flip-flops. The output of this gate can be connected to the reset side of the *down* flip-flop, and the counter will then be unable to count beyond all 0s. The gates are shown in Fig. 12.29.

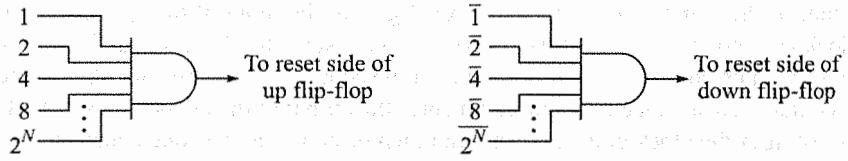


Fig. 12.29 **Count-limiting gates for the converter in Fig. 12.25**

SELF-TEST

12. How does the continuous-type A/D converter differ from the simple counter-type A/D converter?
13. What advantage does the continuous-type A/D converter offer over the counter-type A/D converter?

12.8 A/D TECHNIQUES

There are a variety of other methods for digitizing analog signals—too many to discuss in detail. Nevertheless, we shall take the time to examine two more techniques and the reasons for their importance.

Probably the most important single reason for investigating other methods of conversion is to determine ways to reduce the conversion time. Recall that the simultaneous converter has a very fast conversion time. The counter converter is simple logically but has a relatively long conversion time. The continuous converter has a very fast conversion time once it is locked on the signal but loses this advantage when multiplexing inputs.

Successive Approximation

If multiplexing is required, the *successive-approximation converter* is most useful. The block diagram for this type of converter is shown in Fig. 12.30a. The converter operates by successively dividing the voltage ranges in half. The counter is first reset to all 0s, and the MSB is then set. The MSB is then left in or taken out (by resetting the MSB flip-flop) depending on the output of the comparator. Then the second MSB is set in, and a comparison is made to determine whether to reset the second MSB flip-flop. The process is repeated down to the LSB, and at this time the desired number is in the counter. Since the conversion involves operating on one flip-flop at a time, beginning with the MSB, a ring counter may be used for flip-flop selection.

The successive-approximation method thus is the process of approximating the analog voltage by trying 1 bit at a time beginning with the MSB. The operation is shown in diagram form in Fig. 12.30b. It can be seen from this diagram that each conversion takes the same time and requires one conversion cycle for each bit. Thus the total conversion time is equal to the number of bits, n, times the time required for one conversion cycle. One conversion cycle normally requires one cycle of the clock. As an example, a 10-bit converter operating with a 1-MHz clock has a conversion time of $10 \times 10^{-6} = 10^{-5} = 10\ \mu s$.

When dealing with conversion times this short, it is usually necessary to take into account the other delays in the system (e.g. switching time of the multiplexer, settling time of the ladder network, comparator delay, and settling time).

All the logic blocks inside the dashed line in Fig. 12.30a, or some equivalent arrangement, are frequently constructed on a single MSI chip; this chip is called a *successive-approximation register* (SAR). For example, the Motorola MC6108 shown in Fig. 12.28c is an 8-bit microprocessor-compatible A/D converter that includes an SAR, D/A conversion capabilities, control logic, and buffered digital outputs, in a 28-pin DIP.

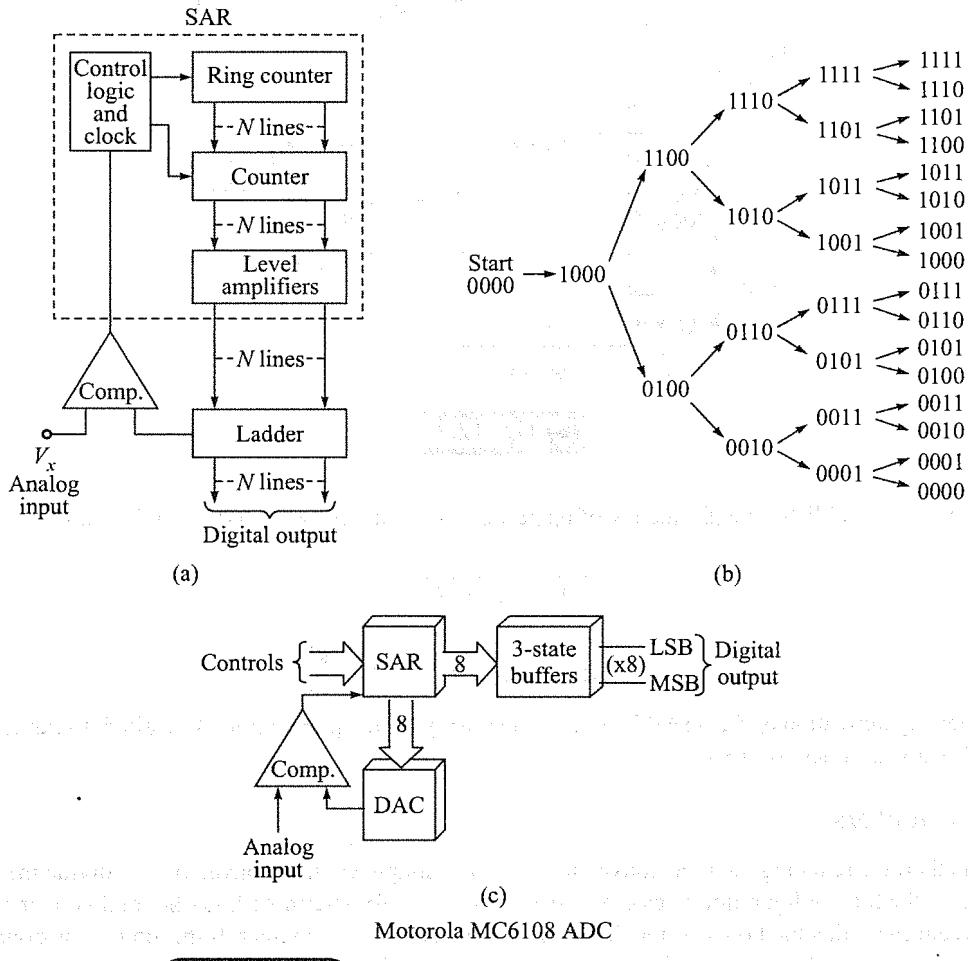


Fig. 12.30 **Successive approximation converter**

The ADC0804

The ADC0804 is an inexpensive and very popular A/D converter which is available from a number of different manufacturers, including National Semiconductor. The ADC0804 is an 8-bit CMOS microprocessor compatible successive-approximation A/D converter that is supplied in a 20-pin DIP. It is capable of digitizing an analog input voltage within the range 0 to +5 Vdc, and it only requires a single dc supply voltage—usually +5 Vdc. The digital outputs are both TTL- and CMOS-compatible.

The block diagram of an ADC0804 is shown in Fig. 12.31. In this case, the controls are wired such that the converter operates continuously. This is the so-called *free-running mode*. The 10-kΩ resistor, along with the

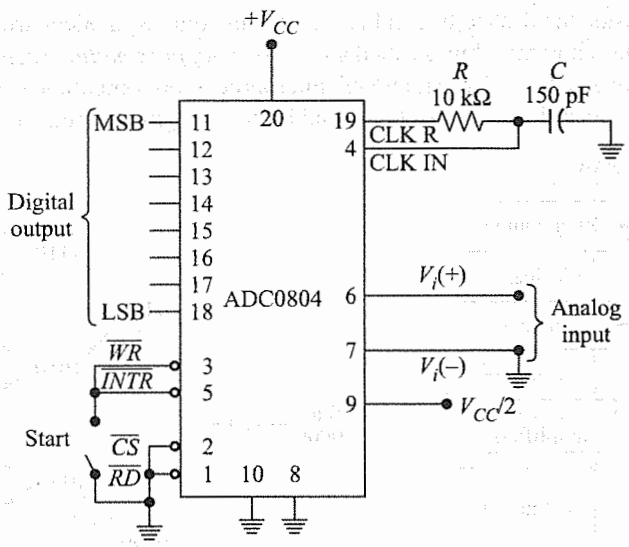


Fig. 12.31

150-pF capacitor, establishes the frequency of operation according to $f \approx 1/1.1(RC)$. In this case,

$$f \approx \frac{1}{1.1 \times (10\ \text{k}\Omega \times 150\ \text{pF})}$$

$$= \frac{1}{1.1 \times (10^4 \times 1.5 \times 10^{-12})} = 607\ \text{kHz}$$

A momentary activation of the START switch is necessary to begin operation. A detailed discussion of the ADC0804 is given in Section 15.4.

Section Counters

Another method for reducing the total conversion time of a simple counter converter is to divide the counter into sections. Such a configuration is called a *section counter*. To determine how the total conversion time might be reduced by this method, assume that we have a standard 8-bit counter. If this counter is divided into two equal counters of 4 bits each, we have a section converter. The converter operates by setting the section containing the four LSBs to all 1s and then advancing the other sections until the ladder voltage exceeds the input voltage. At this point the four LSBs are all reset, and this section of the counter is then advanced until the ladder voltage equals the input voltage.

Notice that a maximum of $2^4 = 16$ counts is required for each section to count full scale. Thus this method requires only $2 \times 2^4 = 2^5 = 32$ counts to reach full scale. This is a considerable reduction over the $2^8 = 256$ counts required for the straight 8-bit counter. There is, of course, some extra time required to set the counters initially and to switch from counter to counter during the conversion. This logical operation time is very small, however, compared with the total time saved by this method.

This type of converter is quite often used for digital voltmeters, since it is very convenient to divide the counters by counts of 10. Each counter is then used to represent one of the digits of the decimal number appearing at the output of the voltmeter. We discuss this subject in detail in the next chapter.

SELF-TEST

14. What does SAR stand for in Fig. 12.30c?
15. What is an ADC0804?

12.9 DUAL-SLOPE A/D CONVERSION

Up to this point, our interest in different methods of A/D conversion has centered on reducing the actual conversion time. If a very short conversion time is not a requirement, there are other methods of A/D conversion that are simpler to implement and much more economical. Basically, these techniques involve comparison of the unknown input voltage with a reference voltage that begins at zero and increases linearly with time. The time required for the reference voltage to increase to the value of the unknown voltage is directly proportional to the magnitude of the unknown voltage, and this time period is measured with a digital counter. This is referred to as a *single-ramp method*, since the reference voltage is sloped like a ramp. A variation on this method involves using an operational amplifier integrating circuit in a dual-ramp configuration. The dual-ramp method is very popular, and widely used in digital voltmeters and digital panel meters. It offers good accuracy, good linearity, and very good noise-rejection characteristics.

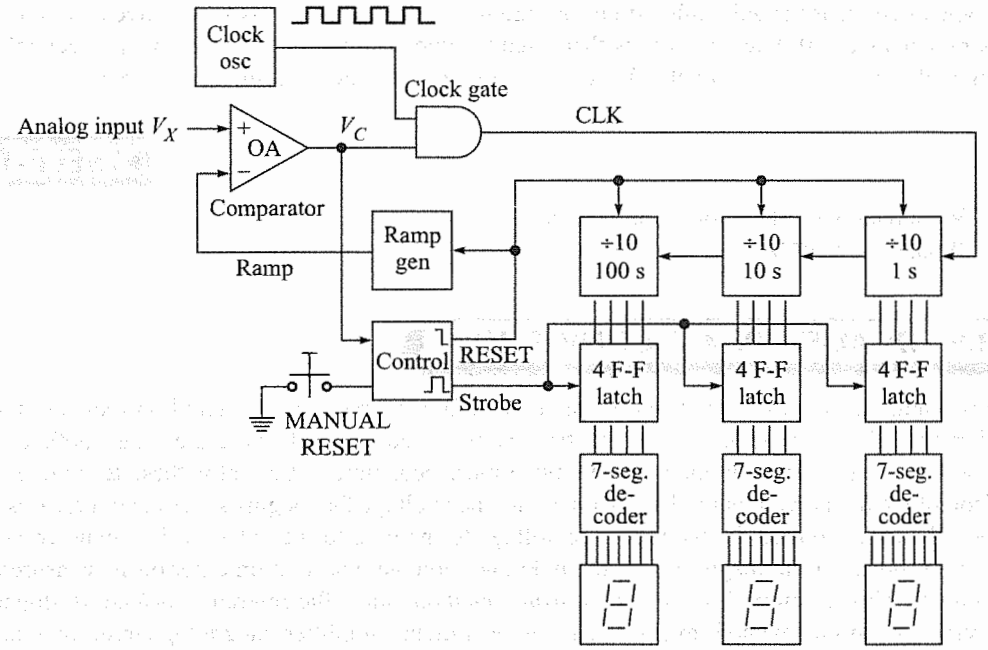
Single-Ramp A/D Converter

Let's take a look at the single-ramp A/D converter in Fig. 12.32. The heart of this converter is the *ramp generator*. This is a circuit that produces an output voltage ramp as shown in Fig. 12.33a. The output voltage begins at zero and increases linearly up to a maximum voltage V_m. It is important that this voltage be a straight line—that is, it must have a constant slope. For instance, if $V_m = 1.0$ Vdc, and it takes 1.0 ms for the ramp to move from 0.0 up to 1.0 V, the slope is 1 V/ms, or 1000 V/s.

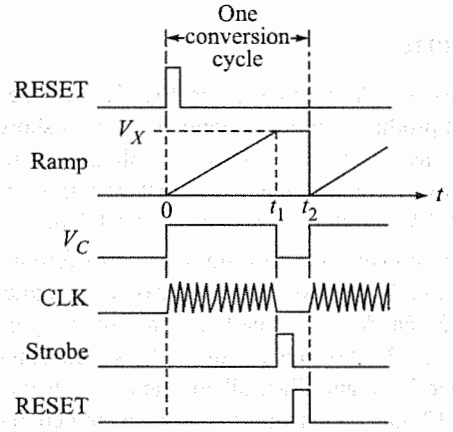
This ramp generator can be constructed in a number of different ways. One way might be to use a D/A converter driven by a simple binary counter. This would generate the staircase waveform previously discussed and shown in Fig. 12.16a. A second method is to use an operational amplifier (OA) connected as an integrator as shown in Fig. 12.33b. For this circuit, if V_i is a constant, the output voltage is given by the relationship $V_o = (V_i/RC)t$. Since V_i, R, and C are all constants, this is the equation of a straight line that has a slope (V_i/RC) as shown in Fig. 12.33a. Now that we have a way to generate a voltage ramp and we understand its characteristics, let's return to the converter in Fig. 12.32.

We assume that the clock is running continuously and that any input voltage V_X that we wish to digitize is positive. If it is not, there are circuits that we can use to adjust for negative input signals. The three decade counters are connected in cascade, and their outputs can be strobed into three 4-flip-flop latch circuits. The latches are then decoded by seven-segment decoders to drive the LED displays as units, tens, and hundreds of counts. We can begin a conversion cycle by depressing the MANUAL RESET switch.

Refer carefully to the logic diagram and the waveforms in Fig. 12.32. MANUAL RESET generates a RESET pulse that clears all the decade counters to 0s and resets the ramp voltage to zero. Since V_X is positive and RAMP begins at zero, the output of the comparator OA, V_c, must be high. This voltage enables the



(a)



(b)

Fig. 12.32 **Single-slope A/D converter**

CLOCK gate allowing the clock, CLK, to be applied to the decade counter. The counter begins counting upward, and the RAMP continues upward until the ramp voltage is equal to the unknown input V_X.

At this point, time t_1, the output of the comparator V_c goes low, thus disabling the CLOCK gate and the counters cease to advance. Simultaneously, this negative transition on V_c generates a STROBE signal in the CONTROL box that shifts the contents of the three decade counters into the three 4-flip-flop latch circuits.

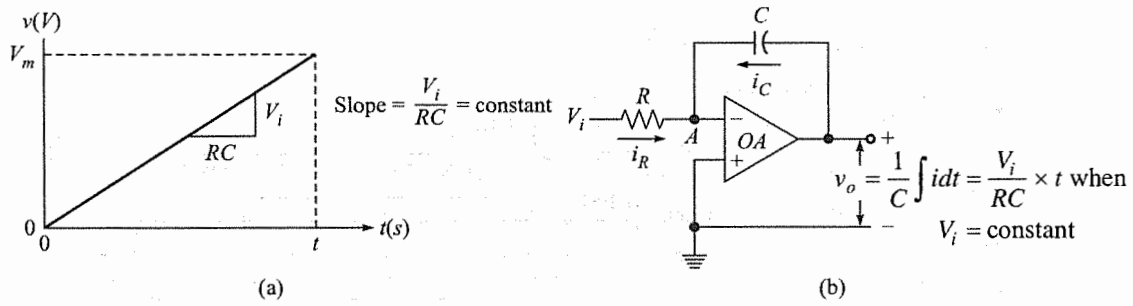


Fig. 12.33 **An integrating circuit**

Shortly thereafter, a reset pulse is generated by the CONTROL box that resets the RAMP and clears the decade counters to 0s, and another conversion cycle begins. In the meantime, the contents of the previous conversion are contained in the latches and are displayed on the seven-segment LEDs.

As a specific example, suppose that the clock in Fig. 12.32 is set at 1.0 MHz and the ramp voltage slope is 1.0 V/ms. Note that the decade counters have the ability to store and display any decimal number from 000 up to 999. From the beginning of a conversion cycle, it will require 999 clock pulses (999 μs) for the counters to advance full scale. During this same time period, the ramp voltage will have increased from 0.0 V up to 999 mV. So, this circuit as it stands will display the value of any input voltage between 0.0 V and 999 mV.

In effect, we have a digital voltmeter! For instance, if $V_X = 345$ mV, it will require 345 clock pulses for the counter to advance from 000 to 345, and during the same time period the ramp will have increased to 345 mV. So, at the end of the conversion cycle, the display output will read 345—we supply the units of millivolts.

One weakness of the single-slope A/D converter is its dependency on an extremely accurate ramp voltage. This in turn is strongly dependent on the values of R and C and variations of these values with time and temperature. The dual-slope A/D converter overcomes these problems.

Dual-Slope A/D Converter

The logic diagram for a basic dual-slope A/D converter is given in Fig. 12.32. With the exception of the ramp generator and the comparator, the circuit is similar to the single-slope A/D converter in Fig. 12.32. In this case, the integrator forms the desired ramp—in fact, two different ramps—as the input is switched first to the unknown input voltage V_X and then to a known reference voltage V_r. Here's how it works.

We begin with the assumptions that the clock is running, and that the input voltage V_X is positive. A conversion cycle begins with the decade counters cleared to all 0s, the ramp reset to 0.0 V, and the input switched to the unknown input voltage V_X. Since V_X is positive, the integrator output V_c will be a negative ramp. The comparator output V_g is thus positive and the clock is allowed to pass through the CLOCK GATE to the counters. We allow the ramp to proceed for a fixed time period t_1, determined by the count detector for time t_1. The actual voltage V_c at the end of the fixed time period t_1 will depend on the unknown input V_X, since we know that $V_c = -(V_x/RC) \times t_1$ for an integrator.

When the counter reaches the fixed count at time t_1, the CONTROL unit generates a pulse to clear the decade counters to all 0s and switch the integrator input to the negative reference voltage V_r. The integrator will now begin to generate a ramp beginning at $-V_c$ and increasing steadily upward until it reaches 0.0 V. All this time, the counter is counting, and the conversion cycle ends when $V_c = 0.0$ V since the CLOCK GATE

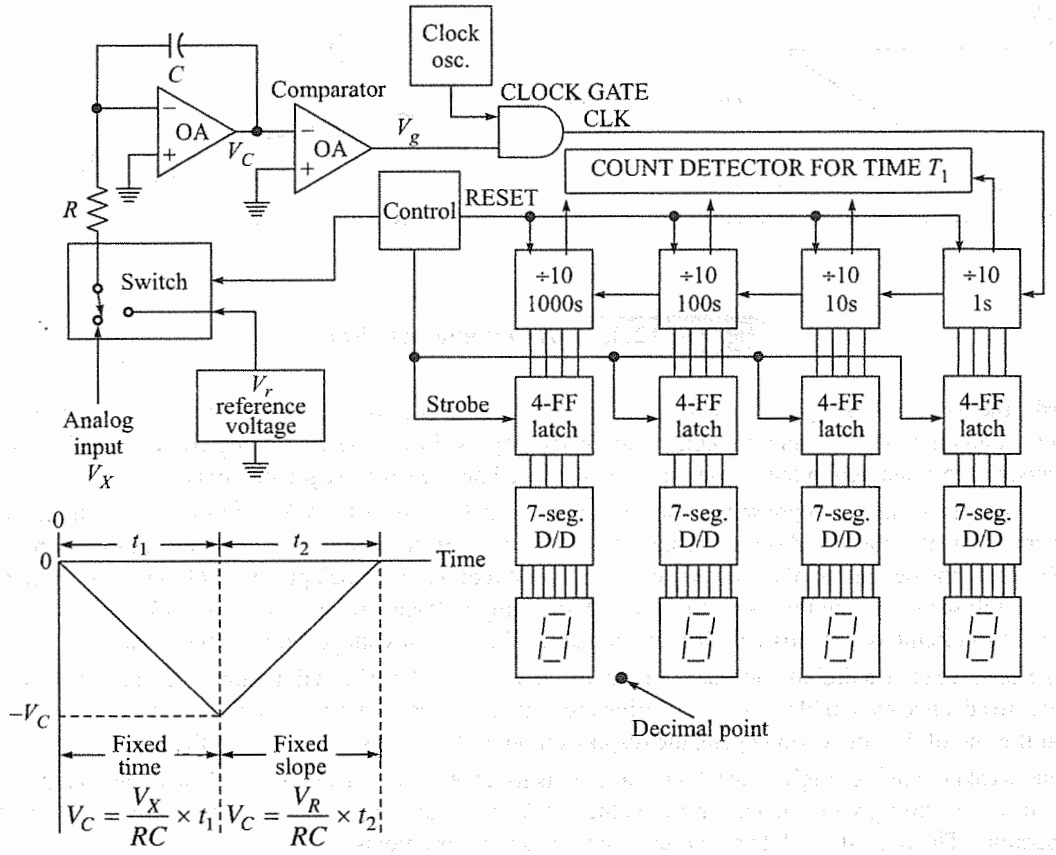


Fig. 12.34 **Dual-slope A/D converter**

is now disabled. The equation for the positive ramp is $V_c = (V_r/RC) \times t_2$. In this case, the slope of this ramp (V_r/RC) is constant, but the time period t_2 is variable.

In fact, since the integrator output voltage begins at 0.0 V, integrates down to $-V_c$, and then integrates back up to 0.0 V, we can equate the two equations given for V_c. That is:

$$\frac{V_X}{RC} \times t_1 = \frac{V_r}{RC} \times t_2$$

The value RC will cancel from both sides, leaving

$$V_X = V_r \times \frac{t_2}{t_1}$$

Since V_r is a known reference voltage and t_1 is a predetermined time, clearly the unknown input voltage is directly proportional to the variable time period t_2. However, this time period is exactly the contents of the decade counters at the end of a conversion cycle! The obvious advantage here is that the RC terms cancel from both sides of the equation above—in other words, this technique is free from the absolute values of either R or C and also from variations in either value.

As a concrete example, let's suppose that the clock in Fig. 12.34 is 1.0 MHz, the reference voltage is −1.0 Vdc, the fixed time period t_1 is 1000 μs, and the RC time constant of the integrator is set at RC = 1.0 ms. During the time period t_1, the integrator voltage V_c, will ramp down to −1.0 Vdc if V_X = 1.0 V. Then, during time t_2, V_c will ramp all the way back up to 0.0 V, and this will require a time of 1000 μs, since the slope of this ramp is fixed at 1.0 V/ms. The output display will now read 1000, and with placement of a decimal as shown, this reads 1.000 V.

Another way of expressing the operation of this A/D converter is to solve the equation $V_X = V_r(t_2/t_1)$ for t_2, since t_2 is the digital readout. Thus $t_2 = (V_X/V_r)t_1$. If the same values as given above are applied, an unknown input voltage V_X = 2.75 V will be digitized and the readout will be $t_2 = (2.75/1.0)1000 = 2750$, or 2.75 V, using the decimal point on the display. Notice that we have used $t_1 = 1000$, the number of clock pulses that occur during the time period t_1. Likewise, t_2 is the number of clock pulses that occur during the time period t_2.

SELF-TEST

16. Is a single-ramp A/D converter slower or faster than a successive-approximation A/D converter?
17. What is the greatest weakness of a single-ramp A/D converter?
18. What advantage does the dual-slope A/D converter offer over the single-ramp A/D converter?

12.10 A/D ACCURACY AND RESOLUTION

Since the A/D converter is a closed-loop system involving both analog and digital systems, the overall accuracy must include errors from both the analog and digital positions. In determining the overall accuracy it is easiest to separate the two sources of error.

If we assume that all components are operating properly, the source of the digital error is simply determined by the resolution of the system. In digitizing an analog voltage, we are trying to represent a continuous analog voltage by an equivalent set of digital numbers. When the digital levels are converted back into analog form by the ladder, the output is the familiar staircase waveform. This waveform is a representation of the input voltage but is certainly not a continuous signal. It is, in fact, a discontinuous signal composed of a number of discrete steps. In trying to reproduce the analog input signal, the best we can do is to get on the step which most nearly equals the input voltage in amplitude.

The simple fact that the ladder voltage has steps in it leads to the digital error in the system. The smallest digital step, or quantum, is due to the LSB and can be made smaller only by increasing the number of bits in the counter. This inherent error is often called the *quantization error* and is commonly ±1 bit. If the comparator is centered, as with the continuous converter, the quantization error can be made ±1/2 LSB.

The main source of analog error in the A/D converter is probably the comparator. Other sources of error are the resistors in the ladder, the reference-voltage supply ripple, and noise. These can, however, usually be made secondary to the sources of error in the comparator.

The sources of error in the comparator are centered around variations in the dc switching point. The dc switching point is the difference between the input voltage levels that cause the output to change state. Variations in switching are due primarily to offset, gain, and linearity of the amplifier used in the comparator. These parameters usually vary slightly with input voltage levels and quite often with temperature. It is these changes which give rise to the analog error in the system.

An important measure of converter performance is given by the differential linearity. *Differential linearity* is a measure of the variation in voltage-step size that causes the converter to change from one state to the next: It is usually expressed as a percent of the average step size. This performance characteristic is also a function of the conversion method and is best for the converters having counters that count continuously. The counter-type and continuous-type converters usually have better differential linearity than do the successive-approximation-type converters. This is true since the ladder voltage is always approaching the analog voltage from the same direction in the one case. In the other case, the ladder voltage is first on one side of the analog voltage and then on the other. The comparator is then being used in both directions, and the net analog error from the comparator is thus greater.

The next logical question that might be asked is: what should be the relative order of magnitudes of the analog and digital errors? As mentioned previously, it would be difficult to justify construction of a 15-bit converter that has an overall error of 1 percent. In general, it is considered good practice to construct converters having analog and digital errors of approximately the same magnitudes. There are many arguments for and against this, and any final argument would have to depend on the situation. As an example, an 8-bit converter would have a quantization error of $\frac{1}{256} \cong 0.4$ percent. It would then seem reasonable to construct this converter to an accuracy of 0.5 percent in an effort to achieve an overall accuracy of 1.0 percent. This might mean constructing the ladder to an accuracy of 0.1 percent, the comparator to an accuracy of 0.2 percent, and so on, since these errors are all accumulative.

Example 12.13 What overall accuracy could one reasonably expect from the construction of a 10-bit A/D converter?

Solution A 10-bit converter has a quantization error of $\frac{1}{1024} \cong 0.1$ percent. If the analog portion can be constructed to an accuracy of 0.1 percent, it would seem reasonable to strive for an overall accuracy of 0.2 percent.

SUMMARY

Digital-to-analog conversion, the process of converting digital input levels into an equivalent analog output voltage, is most easily accomplished by the use of resistance networks. The binary ladder has been found to have definite advantages over the resistance divider. The complete D/A converter consists of a binary ladder (usually) and a flip-flop register to hold the digital input information.

The simultaneous method for A/D conversion is very fast but becomes cumbersome for more than a few bits of resolution. The counter-type A/D converter is somewhat slower but represents a much more reasonable solution for digitizing high-resolution signals. The continuous-converter method, the successive-approximation method, and the section-counter method are all variations of the basic counter-type A/D converter which lead to a much faster conversion time. A dual-slope A/D converter is somewhat slower than the previously discussed methods but offers excellent accuracy in a relatively inexpensive circuit. Dual-slope A/D converters are widely used in digital voltmeters.

The D/A converter and A/D converter logic circuits given in this chapter are all drawn in logic block diagram form and can all be constructed by simply connecting these commercially available logic blocks. For instance, a D/A converter can be constructed by connecting resistors that have values of R and $2R$, or an A/D converter can be constructed by connecting the various inverters, gates, flip-flops, and so on; however, you must realize that these units are now readily available as MSI circuits. The only really practical and economical way to build D/A converters or A/D converters is to make use of these commercially available circuits; this is exactly the subject pursued in the next chapter.

GLOSSARY

- ***A/D conversion*** The process of converting an analog input voltage to a number of equivalent digital output levels.
- ***A/D converter flash type*** Effects fast and simultaneous conversion of analog data to digital through number of comparators.
- ***A/D converter tracking type*** Effects tracking of analog input through its continuous comparison with converter's digital output.
- ***binary equivalent weight*** The value assigned to each bit in a digital number, expressed as a fraction of the total. The values are assigned in binary fashion according to the sequence 1, 2, 4, 8, ... , 2^n, where n is the total number of bits.
- ***D/A conversion*** The process of converting a number of digital input signals to one equivalent analog output voltage.
- ***differential linearity*** A measure of the variation in size of the input voltage to an A/D converter which causes the converter to change from one state to the next.
- ***Millman's theorem*** A theorem from network analysis which states that the voltage at any node in a resistive network is equal to the sum of the currents entering the node divided by the sum of the conductances connected to the node, all determined by assuming that the voltage at the node is zero.
- ***monotonicity*** A consistent increase in output in response to a consistent increase in input (voltage or current).
- ***quantization error*** The error inherent in any digital system due to the size of the LSB.
- ***sample and hold circuit*** Samples analog voltage signal and holds briefly to facilitate analog to digital conversion.
- ***SAR*** Sequential approximation register, used in a sequential A/D converter.

PROBLEMS

Section 12.1

12.1 What is the binary equivalent weight of each bit in a 6-bit resistive divider?

12.2 Draw the schematic for a 6-bit resistive divider.

12.3 Verify the voltage output levels for the network in Fig. 12.4, using Millman's theorem. Draw the equivalent circuits.

12.4 Assume that the divider in Prob. 12.2 has +10 V full-scale output, and find the following:
   a. The change in output voltage due to a change in the LSB
   b. The output voltage for an input of 110110

12.5 A 10-bit resistive divider is constructed such that the current through the LSB resistor is 100 μA. Determine the maximum current that will flow through the MSB resistor.

Sections 12.2, 12.3 and 12.4

12.6 What is the full-scale output voltage of a 6-bit binary ladder if 0 = 0 V and 1 = +10 V? Of an 8-bit ladder?

12.7 Find the output voltage of a 6-bit binary ladder with the following inputs:
   a. 101001 b. 111011
   c. 110001

12.8 Check the results of Prob. 11-7 by adding the individual bit contributions.

12.9 What is the resolution of a 12-bit D/A converter which uses a binary ladder? If the full-scale output is +10 V, what is the resolution in volts?

12.10 How many bits are required in a binary ladder to achieve a resolution of 1 mV if full scale is +5 V?

Section 12.5

12.11 How many comparators are required to build a 5-bit simultaneous A/D converter?

12.12 Redesign the encoding matrix and READ gates in Fig. 12.20, using NAND gates.

12.13 Assuming that the input reference voltage is $V = 10.0$ Vdc, determine the digital output of the A/D converter in Fig. 12.21a for an input voltage of:

a. 1.25 V b. 3.33 V
c. 8.05 V

Section 12.6

12.14 Find the following for a 12-bit counter-type A/D converter using a 1-MHz clock:

a. Maximum conversion time
b. Average conversion time
c. Maximum conversion rate

12.15 What clock frequency must be used with a 10-bit counter-type A/D converter if it must be capable of making at least 7000 conversions per second?

12.16 Design additional control circuitry for Fig. 12.24 such that the A/D converter in Fig. 12.23 will continue to make conversions after an initial START pulse is applied.

Sections 12.7 and 12.8

12.17 What is the conversion time of a 12-bit successive-approximation-type A/D converter using a 1-MHz clock?

12.18 What is the conversion time of a 12-bit section-counter-type A/D converter using a 1-MHz clock? The counter is divided into three equal sections.

Section 12.9

12.19 For the integrator in Fig. 12.31, show that the output voltage is given by $V_o = \{V_i/RC)t$, assuming that the input voltage V_i is a constant. [*Hint*: Using Kirchhoff's current law at node A, the resistor current i_R is equal to the capacitor current i_C, but $i_R = V_i/R$ and $i_C = q/t = (V_oC)/t$.]

12.20 Design the control logic for the CONTROL box in Fig. 12.32 to generate the proper control signals shown in that figure.

12.21 Calculate a value for C in Fig. 12.33 to obtain a fixed slope $V_i/(RC) = 1000$ V/s, given $V_i = 1.0$ Vdc and $R = 100$ kΩ.

12.22 Can you design an amplifier such that the output is always positive and is equal to the magnitude of the input voltage? In other words, the input can be either $+V_i$ or $-V_i$ but in either case, the output will be $+V_i$.

12.23 Design the CONTROL logic for the converter in Fig. 12.34.

Section 12.10

12.24 What overall accuracy could you reasonably expect from a 12-bit A/D converter?

12.25 Discuss the overall acceptable accuracy of a 10-bit A/D converter in terms of quantization error, ladder accuracy, comparator accuracy, converter accuracy, and other factors.

Answers to Self-tests

1. 1/63
2. 30/15 = 2 V
3. 0.15625 V
4. 9.84375 V
5. A monotonicity test checks to see that the D/A output voltage increases regularly as the input digital signals increase.
6. +5 Vdc
7. Resolution = 10/256 = 39.06 mV
8. Its conversion time is very fast.
9. Possibilities include radar signal processing, video displays, high-speed instrumentation, and television broadcasting.
10. 128 μs

11. 64 μs
12. The continuous type A/D converter uses an up-down counter.
13. The continuous type A/D converter is faster than the counter-type A/D converter.
14. SAR stands for successive-approximation register.
15. The ADC0804 is an 8-bit CMOS successive-approximation A/D converter.
16. Slower
17. One major weakness of the single-slope A/D converter is that it is extremely sensitive to variations in ramp voltage and hence to errors in ramp voltage.
18. The *RC* time constant cancels out, making the conversion much less sensitive to variations in ramp voltage accuracy.

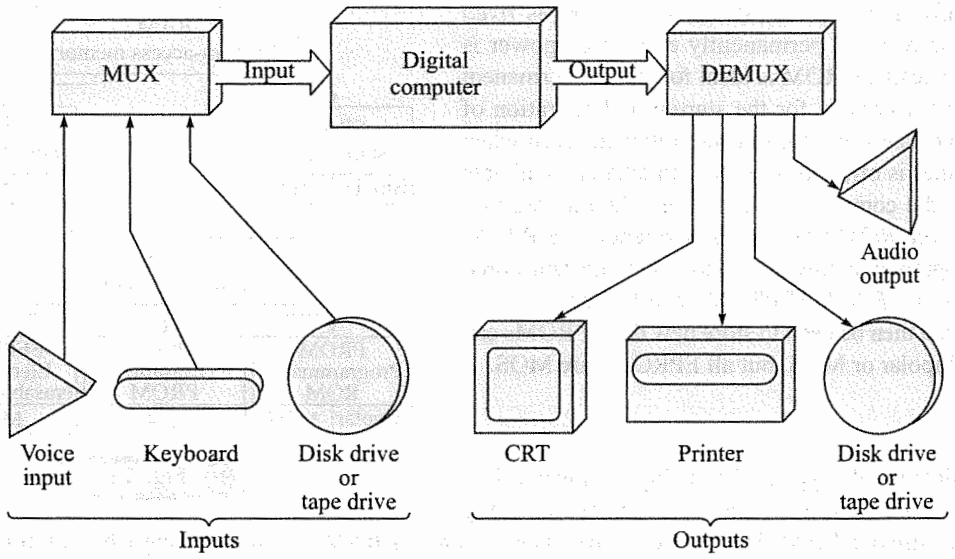
Memory

13

OBJECTIVES

- List the various forms of magnetic and optical memory and explain how each works
- Discuss memory addressing techniques
- Describe ROM, PROM, and EPROM and their characteristics and differences
- Compare the advantages and disadvantages of SRAM and DRAM and be familiar with the basic features of SRAM and DRAM chips
- Describe how content addressable memory works

The ability to store information (to remember) is an important requirement in a digital system. Circuits and/or systems designed specifically for data storage are referred to as memory. In the simplest application, the memory may be a flip-flop, or perhaps a number of flip-flops connected to form a register. In a larger system, such as a microcomputer, the memory may be composed of semiconductor memory chips. Semiconductor memories are composed of bipolar transistors or MOS transistors on an integrated circuit (IC), and are available in two general categories—read-only memory (ROM) and random-access memory (RAM). ROM and RAM memories can be constructed to store impressive amounts of data entirely within a computer system. Both programmed instructions and data are stored in a computer by means of ROM and RAM. But really large amounts of data (such as banking or insurance records) are generally stored using magnetic memory techniques. Magnetic memory includes the recording of digital information on magnetic tape, hard disks, and floppy disks. Magnetic storage systems are quite sophisticated and are usually externally accessed, as shown in Chapter 1 in Fig. 1.28 and repeated here for reference. However, in last two decades there has been tremendous growth in optical memory devices like compact disk, digital versatile disk etc. that gives low cost high capacity alternative storage solution.



13.1 BASIC TERMS AND IDEAS

Semiconductor Memory

Recent advances in semiconductor technology have provided a number of reliable and economical MSI and LSI memory circuits. The typical semiconductor memory consists of a rectangular array of *memory cells*, fabricated on a silicon wafer, and housed in a convenient package, such as a DIP. The basic memory cell is typically a transistor flip-flop or a circuit capable of storing charge and is used to store 1 bit of information. Memories are usually classified as either bipolar, metal oxide semiconductor (MOS), or complementary metal oxide semiconductor (CMOS) according to the type of transistor used to construct the individual memory cells. The total number of cells in a memory determine its *capacity*. For instance, a 1024 bipolar memory chip is a semiconductor memory that has 1024 memory cells, each cell consisting of a flip-flop constructed with the use of bipolar transistors. *Chip* is a term commonly used to refer to a semiconductor memory device. In general, faster operation is obtained with a bipolar memory chip, but greater packing density and thus reduced size and cost, as well as lower power requirements, are characteristics of MOS and CMOS memory chips.

Characteristics

The two general categories of memory, RAM and ROM, can be further divided as illustrated in Fig. 13.1. A dc power supply is required to energize any semiconductor memory chip. Once dc power is applied to a *static RAM (SRAM)*, the SRAM retains stored information indefinitely, without any further action. A *dynamic RAM (DRAM)*, on the other hand, does not retain stored data indefinitely; any stored data must be stored again (refreshed) periodically. Both SRAMs and DRAMs are used to construct the memory inside a microcomputer or minicomputer (see Fig. 1.34 in Chapter 1). DRAMs are used as the bulk of the memory, and high-speed SRAMs are used for a smaller, rapid-access type of memory known as *cache memory*. The cache is used to momentarily store selected data in order to improve computer speed of operation. SRAMs can be either bipolar or MOS, but all DRAMs are MOS.

The information (data) stored in a ROM is *fixed* and will be retained permanently even if dc power is removed. Clearly, a ROM is ideal for storing permanent instructions necessary for the startup and operation of a computer. These instructions are retained, even when the computer is off, and become immediately available each time the computer is turned on. Data stored in a *pogrammable ROM* (PROM) is permanent—a PROM can be programmed only once! However, the data stored in an *erasable PROM* (EPROM) can be "erased"; the EPROM can then be used to store new data. PROMs can be either bipolar or MOS, but all EPROMs are MOS.

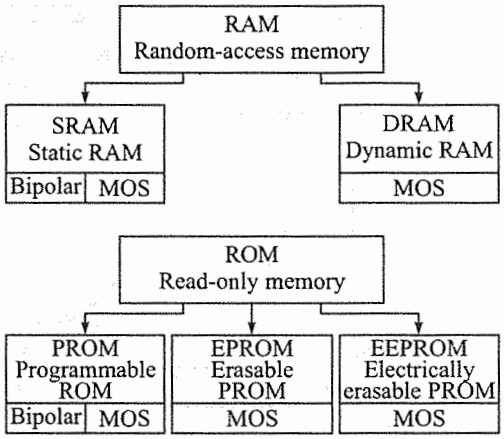


Fig. 13.1

RAM

A block diagram of a typical RAM chip is shown in Fig. 13.2a. An application in which data changes frequently calls for the use of a RAM. The logic circuitry associated with a RAM will allow a single bit of information to be stored in any of the memory cells—this is the write operation. There is also logic circuitry that will detect whether a 0 or a 1 is stored in any particular cell—this is the read operation. The fact that a bit can be written (stored) in any cell or read (detected) from any cell suggests the description *random access*. A control signal, usually called *chip-select* or *chip-enable*, is used to enable or disable the chip. In the read mode, data from the selected memory cells is made available at the output. In the write mode, information at the data input is written into (stored in) the selected cells. The address lines determine the cells written into or read from. Since each cell is a transistor circuit, a loss of dc power means a loss of data—a RAM that has this type of memory cell is said to provide *volatile storage*.

ROM

A typical ROM chip is shown in Fig. 13.2b. An application in which the data does not change dictates the use of a ROM. For instance, a "lookup table" that stores the values of mathematical constants such as trigonometric functions or a fixed program such as that used to find the square root of a number could be stored in a ROM. The content of a ROM is fixed during manufacturing, perhaps by metallization or by the

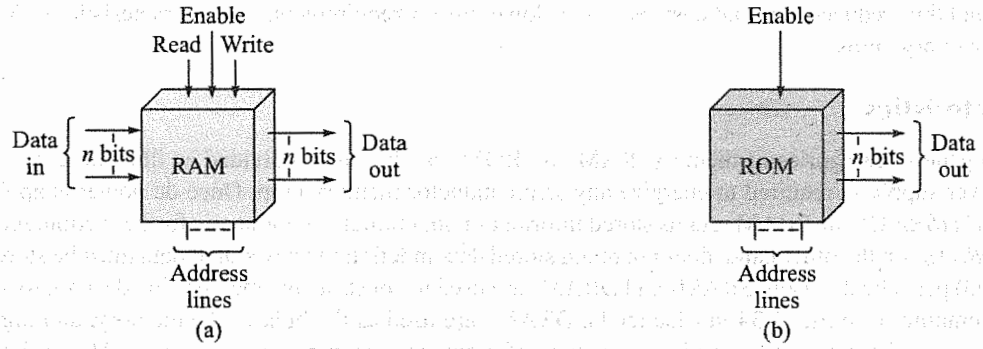


Fig. 13.2

presence or absence of a working transistor in a memory cell, by opening or shorting the gate structure, or by the oxide-layer thickness. A ROM is still random access, since there are logic circuitry and address lines to select any desired cell in the memory. When enabled, data from the selected cells is made available at the output. There is, of course, no write mode. Since data is permanently stored in each cell, a loss of power does not cause a loss of data, and thus a ROM provides *nonvolatile data storage.*

An application in which the data does not change but the required data will not be available until a later time suggests the use of a PROM, where the stored data can be set in the memory by writing into the PROM at the user's convenience. An application in which the data may change from time to time might call for the use of an EPROM.

Example 13.1 State the most likely type of semiconductor memory for each application: (a) main memory in a hand calculator; (b) storing values of logarithms; (c) storing prices of vegetable produce; (d) emergency stop procedures for an industrial mill now in the design stage.

Solution (a) RAM; (b) ROM; (c) EPROM; (d) PROM.

SELF-TEST

1. What is the operational difference between an SRAM and a DRAM?
2. What is an EPROM?
3. What is a cache memory?

13.2 MAGNETIC MEMORY

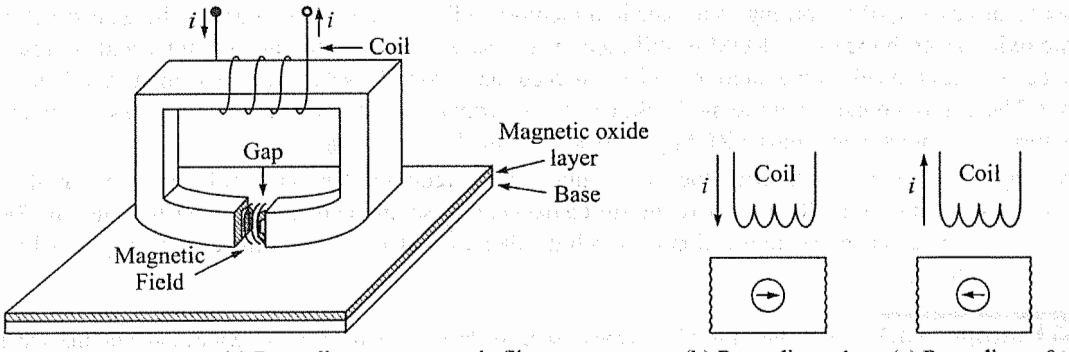
Magnetic tape, floppy disks, and hard disks are all capable of storing large quantities of digital data. A hard disk drive and a floppy disk drive are important components in nearly all microcomputer and minicomputer systems. Large reels of magnetic tape are economical and widely used mass storage components in large computer systems. The basic principle involved in each case is the magnetization of small spots in a thin film of magnetic material.

Magnetic Recording

Magnetic tape is produced by the deposition of a thin film of magnetic material on a long strip of plastic, which is then wound on a reel. Magnetic material deposited on a rigid disk forms the basis of a *hard disk*; the same material on a *semirigid* disk is used to construct a *floppy disk*. Digital information is recorded on any of these surfaces in essentially the same fashion.

A current i in the coil shown in Fig. 13.3a or, the next page will produce a magnetic field across the gap. A portion of this field will extend into the magnetic material below the gap, and the material will be magnetized with a fixed orientation. When the current is removed, a magnetized spot remains, as shown in Fig. 13.3b. Thus, information has been stored. If the current is reversed in direction, a spot will again be magnetized, but with the opposite fixed orientation, as shown in Fig. 13.3c. Clearly this is a binary system, and it can be used to store binary information. For example, one could "define" Part b of Fig. 13.3 as 1 (high) and Part c as 0 (low). Introducing current i to record a 0 or a 1 is *writing* (or recording or storing) data.

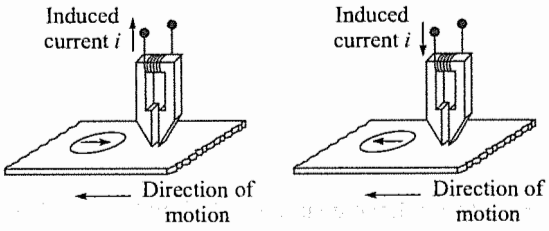
Now if a fixed, magnetized spot with a given orientation is moved past a gap as shown in Fig. 13.4a on the next page, a current with the direction shown will be induced in the coil. But if a magnetized



(a) Recording on a magnetic film (b) Recording a 1 (c) Recording a 0

Fig. 13.3

spot with the opposite orientation is moved past the gap, a current will be induced in the opposite direction, as shown in Fig. 13.4b. Detecting the orientation of the magnetized spot by measuring the induced current is *reading* information (1 or 0).



(a) Reading a 1 (b) Reading a 0

Fig. 13.4

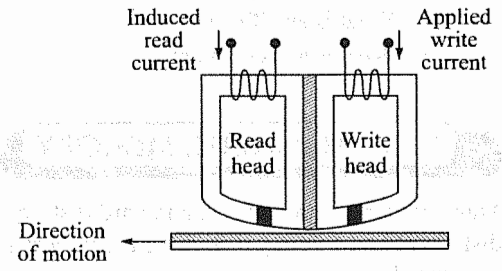
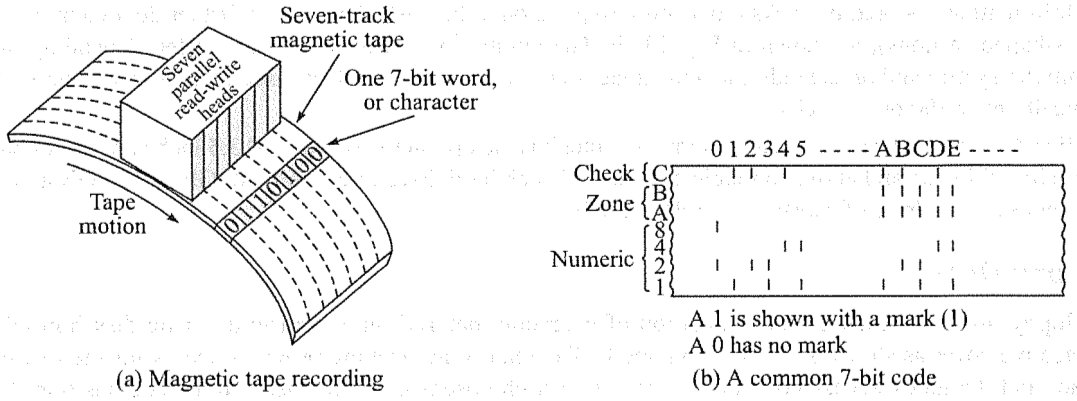


Fig. 13.5 **Dual read-write head**

The same magnetic read-write head in Fig. 13.3a can be used to write digital data or to read digital data. However, the dual read-write head in Fig. 13.5 is more common. Here's why. The tape or disk is moved under the heads in the direction shown. At time t_1, a spot is recorded under the write head. A short time later, at time t_2, this spot passes under the read head. It can then be read out and a check can be made to ensure that the correct data was in fact recorded.

Magnetic Tape

Either seven or nine dual read-write heads are connected in parallel for use with magnetic tape as illustrated in Fig. 13.6a. As the tape moves past the heads, data is read or written, 7 (or 9) bits at a time. In the 7-bit system, alphanumeric information is recorded in coded form, and there is 1 parity bit (even or odd). There are numerous coding schemes, but a portion of a commonly used IBM code is shown in Fig. 13.6b. In the 9-bit system, a data word is composed of 8 bits, and the ninth bit is for parity (either even or odd). Data can be stored in coded form or in straight binary form.



(a) Magnetic tape recording

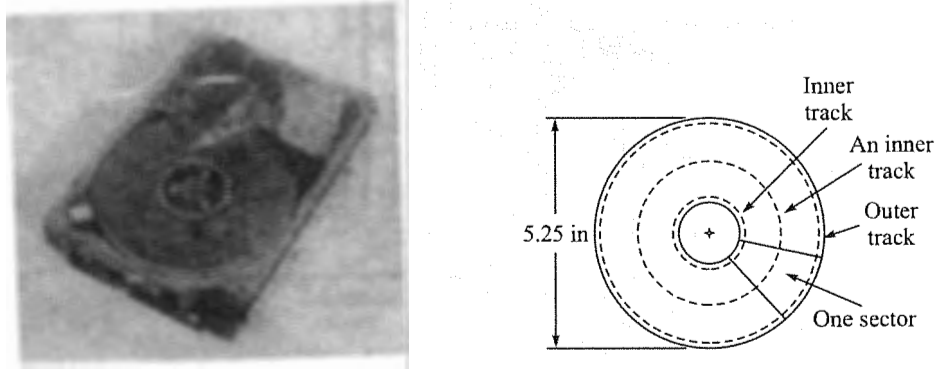
(b) A common 7-bit code

Fig. 13.6

Data storage on a magnetic tape is *sequential*. That is, data is stored one word after another, in sequence. To recover (read) data from the tape requires sequential searching. Clearly, the storage (or recovery) of data in a sequential system such as this requires considerably more time than storage (or recovery) using RAM. Tape is said to have a longer *access time* than RAM. Typical access times are measured in seconds, compared with nanosecond access times for RAMs.

Hard Disks

Magnetic material deposited on a rigid disk (usually aluminum) is the basis for a hard disk system. One or more of these disks are mounted in an enclosure similar to that shown in Fig. 13.7a. The hard disks used in small computer systems are typically 3.5 in. or 5.25 in. in diameter. Hard disk drives with 40 to 400 gigabyte capacities are common in microcomputer systems. The disk is rotated at speeds between 3600 to 7200 rpm and in high end servers up to 15000 rpm resulting in typical access time of 16 ms to 3.6 ms. Because of the relatively short access times and the high storage density, hard disks are widely used in all computer systems.



(a)

(b) Hard disk

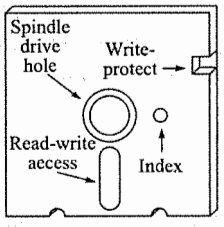
Fig. 13.7 **Hard disk system**

Information is stored in tracks (concentric rings) around the disk. The disk is further divided into sectors (pie-shaped sections), as shown in Fig. 13.7b. The number of tracks and sectors differ, depending on the computer system and on the individual manufacturer. The smaller hard disks used in microcomputer systems typically have 300 or so tracks.

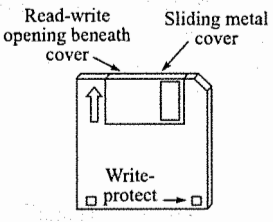
Besides internal Hard Disks, a modern computer has the option to use external 3.5 inch hard drives having capacity of 80 GB and above, portable external 2.5 inch hard drive of capacity 40 GB to 120 GB and palm size pocket hard drive of capacity 2.5 GB or 5 GB.

Floppy Disks

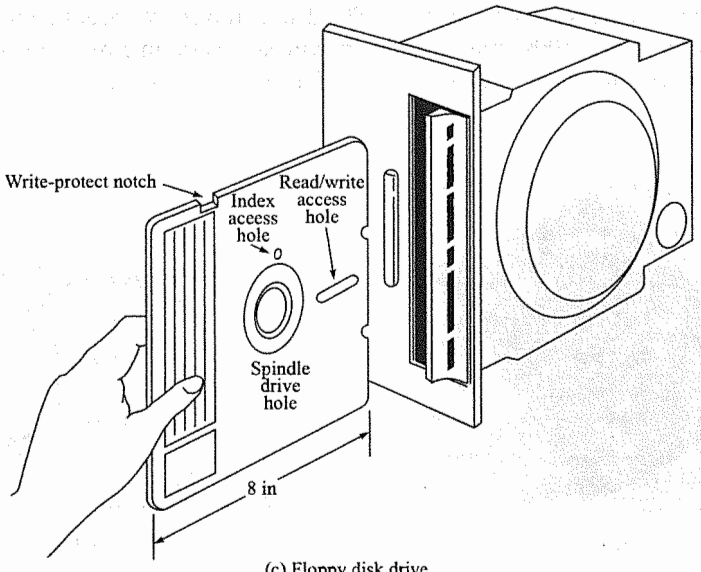
A floppy disk is formed by the deposition of magnetic material on a semirigid plastic disk housed in a protective cover as shown in Figs. 13.8a and b. The read-write opening provides access for the read-write head, and the index access hole allows the use of a photosensor to establish a reference position. When the write-protect notch is covered, data cannot be recorded on the disk, preventing accidental loss of data. Double-sided high-density 5.25-in disks have a capacity of 1.2 MB. Double-sided high-density 3.5-in disks have a capacity of 2.88 MB.



(a) 5.25-in floppy disk (minifloppy)



(b) 3.5-in floppy disk (microfloppy)



(c) Floppy disk drive

Fig. 13.8

As with hard disks, data is stored in tracks and the disk is divided into sectors. In IBM format, 5.25-in disks have 40 tracks per side and 3.5-in disks have 77 tracks per side. The IBM standard for sectors is nine.

The floppy disk is portable, and it must be inserted into a disk drive as shown in Fig. 13.8c. The drive unit consists of a single read-write head, read-write and control electronics, a drive mechanism, and a track-positioning mechanism. The spindle drive rotates the magnetic disk at a speed of 360 rpm. Access time is thus somewhat higher than the hard disk, being about 80 ms on average.

Note that, all the numbers that refer to maximum capacity, speed, given in this section or at other places are improving day by day by rapid technological advancements in this field.

SELF-TEST

4. Does the code in Fig. 13.6b have even or odd parity?
5. Magnetic tape provides inexpensive storage of large quantities of digital data. Why not use it, instead of RAM, in a microcomputer?
6. How can binary information be recorded on magnetic film?

13.3 OPTICAL MEMORY

Introduced in 1982 jointly by Philips and Sony for storing digital audio data, Compact Disk (CD) found its way into computer storage in 1985. There was no looking back since then and today we find different types of CDs flooding the market where binary data is optically coded. The memory capacity of a CD is in the range of 650–700 MB, i.e. nearly 500 times more than 1.44 MB magnetic floppy disk. Both come in movable data storage category with almost same price tag but data integrity in optical disk is maintained over much longer period of time. Its newer variety called Digital Versatile Disk (DVD) can store data from 4.7 GB to 17.1 GB depending on configuration and make. Thus, the growth in optical storage media has been spectacular in last two decades. In this section we'll first discuss how CDs store binary data, what differentiates one type of CD from the other and then we'll look into DVD features.

CD ROM

CD ROM or CD Read Only Memory devices are mass produced in factory using a stamp press technology. CD ROM drives uses LASER (Light Amplification by Stimulated Emission of Radiation) technology to read data from it. A semiconductor LASER generates a high intensity light wave of stable wavelength ≈ 780 nm. A lens system is used to direct the LASER towards the disk over approximately 1 micron diameter spot. Refer to simplified diagram of Fig. 13.9a. The intensity of the reflected light from metallic reflection layer, received by photo sensors gives the information of binary data is stored in CD. There are two different surfaces called *pit* and *land* from which reflection occurs. The pit is approximately 0.12 micron deep compared to land and reflected intensities are about 25% and more than 70% respectively (Fig. 13.9b). Every time laser beam travels from land to pit or pit to land there is a change in intensity of reflected light. This change is read as binary digit 1 and a constant intensity reflected light is interpreted as zero. The pit width is such that there is at least 2 and at most 10 zeroes between every 1. This is achieved by converting every 8-bit byte into a 14-bit value, a process called Eight to Fourteen Modulation (EFM). Such arrangement makes it easy for the read laser to detect bits and also helps in synchronization of internal clock as CDs are essentially self clocking. Data corresponding to a small portion of the track is shown above label in Fig. 13.9a.

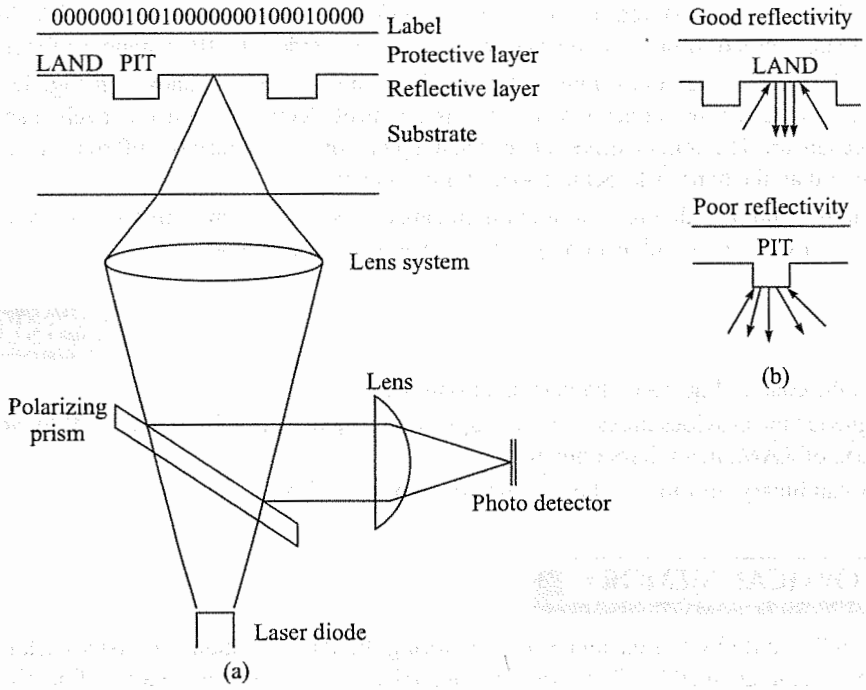


Fig. 13.9 **A compact disk reading system**

A compact disk normally comes in 12 cm diameter. The 1.2 mm thickness has four distinct parts. They are (a) label layer, (b) protective layer, (c) reflective layer, and (d) a transparent substrate layer on which land and pits are formed. The high reliability of CD comes from protection of data on one side by 10–20 microns thick protective lacquer layer and label and on the other side a tough approximately 1.2 mm thick polycarbonate layer. Thus, data integrity is maintained for years against most normal physical abuses and also for the fact that it is not susceptible to magnetic fields or radiations. Note that, small scratches on the surface of CD do not directly erase the data, but create additional areas of light scattering. This can make things difficult for drive's electronic, which is also much less sensitive to radial scratches than to the circumferential ones. The other reason that increases reliability of data stored in a CD is the ability to use efficient error correction codes. The data is stored in the form of a spiral of around 20000 windings totaling approximately 4.5 km of length and contains nearly 2 billion shallow pits on its surface. A macroscopic view of a part of CD is presented in Fig. 13.10.

You definitely have seen some kind of symbol like 48X, 52X etc. written on a CD. What is that? It gives the speed at which data is read from CD where 1X stands for 150 KB/Sec. Earlier versions of CD ROM drive below 12X were built on constant linear velocity (CLV) where motor had variable speed to maintain CLV. Present-day drives are based on constant angular velocity (CAV) where motor rotates at constant speed and this requires less seek time to access data from CD.

CD-R

CD-R or CD-Recordable allows user to write data but once. The CD-R drive has laser unit which uses higher intensity light wave for write operations than read. CD-R disk does not have pits and lands but a photosensitive organic dye (between reflective layer and polycarbonate substrate) that write laser heats

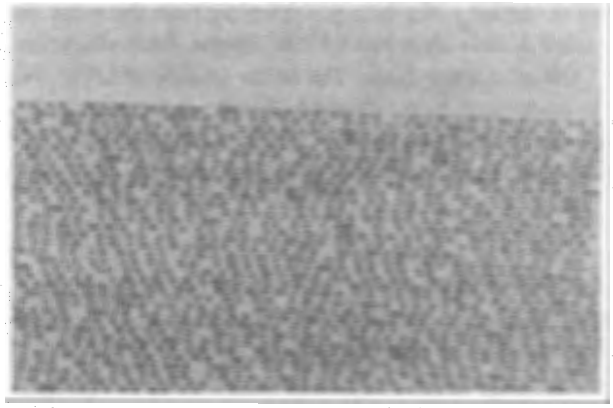


Fig. 13.10 A macroscopic view of a part of compact disk surface

to approximately 250°C. This melts or chemically decomposes the dye to form a depression mark in the recording layer in appropriate places. The places burnt have lower reflectivity of light. Thus read laser gets two different intensities on reflected light while reading the disk, similar to read operation of a CD ROM. Earlier versions of CD-R called WORM, abbreviation of write once read many times required data to be written in only one session or one go. Now it is possible to write data in CD-R in multiple sessions till it is completely filled. The writing speed of CD-R is much slower than the read speed.

CD-RW

CD-RW or CD Read Write, previously known as CD-Erasable gives user facility to write and erase data many times, unlike CD-R. CD-RW uses an active layer of Ag-In-Sb-Te (silver-indium-antimony-tellurium) alloy that has a polycrystalline structure making it reflective (reflectivity 25%). Writing data on disk uses highest power of laser that heats up selected spots to 500°–700°C. At this temperature the chemical structure liquefies losing its polycrystalline structure and on cooling solidifies to an amorphous state that has reduced reflectivity of 15%. The read process is like CD-ROM and CD-R that notes the difference in reflectivity of the reflecting surface.

To reverse the phase or erase data, the laser operates at a lower power setting and heats the active material to nearly 200 °C. This reverses the material from its amorphous to its polycrystalline state and then becomes reflective again. According to manufacturers, in a CD-RW the rewrite operations can be done 1000 times or more. The main drawback of CD-RW is very low reflectivity of the material and the difference between two levels is also not much. This often limits readability of these devices. Note that CD-Recordable drives often come with three different speed ratings, one speed for write-once operations, one for re-write operations, and one for read-only operations. The speeds are typically listed in that order, e.g. 12X/10X/32X. This means CPU and media-permitting, CD drive can write to CD-R disks at 12X speed, write to CD-RW disks at 10X speed and read from CD disks at 32X speed.

DVD

Digital Versatile Disk or Digital Video Disk, popular as DVD resemble compact disk in dimension and look but contains much higher storage space. DVD driver uses smaller wavelength (635 nm or 650 nm) and lower numerical aperture of lens system to read smaller dimension land and pits. Each side can have two layers

from which data is read and in certain disks data is written on both the sides. Single sided, single layer has capacity of 4.7 GB, single sided double layer has 8.5 GB, double sided single layer has 9.4 GB while double sided double layer has 17.1 GB of storage space. The better quality of DVD output compared to CD comes from better channel coding, error correction scheme and of course a higher data transfer rate. Note that in DVD terminology, 1X refers to 1.32 MB/Sec unlike 150 KB/Sec of CD. Like CD, different varieties of DVD like DVD-R, DVD-RW, DVD-RAM are being developed and entering the market.

Handling Tips

Before we complete this section let us provide you some useful tips for handling CDs, the most used movable storage device of these days. Of course, the list is not exhaustive and the requirements come mainly from maintaining a good reflecting surface and physical balance of the CD.

(i) Handle disks by the outer edge or the center hole. Don't touch the surface of the disk to avoid leaving fingerprints and oil behind. Keep the disk free of dirt.

(ii) Clean dirt, smudges, and liquids from disks by wiping with a clean cotton fabric in a straight line radially outward from the center of the disk. Don't wipe in circles. The error correction codes on the disk can handle only small interruptions like scratch that travels across the spiral. You may clean stubborn dirt and foreign substances with 99% isopropyl alcohol or 99% methyl alcohol. First apply the cleaner to a cotton, then rub the cloth across the disk, taking care not to get any fluid on the label side of the disk.

(iii) Label the disk with a non-solvent-based felt-tip permanent marker. Beware of permanent markers that contain strong solvents. The use of adhesive labels is not recommended. If you use a label, don't try to remove or reposition it.

(iv) Never ever bend the disk. Flexing the disk can cause stress patterns to form in the polycarbonate, and if you stretch it far enough the reflective and recording layers get deformed. Store disks vertically. Over a long period, gravity will warp the disk if it's left flat.

(v) Store disks in a cool, dry, dark environment in which the air is clean to avoid corrosion. Keep it away from areas that are excessively hot or damp and also from direct sunlight and other ultraviolet light sources. Do not expose the disk to rapid changes in temperature or humidity.

SELF-TEST

7. What is the wavelength of laser used for reading CD-ROM?
8. What is the reflectivity of CD-ROM and CD-RW surfaces?
9. What is the capacity of a single sided double layer DVD ROM?

13.4 MEMORY ADDRESSING

Cell Selection

Addressing is the process of selecting one of the cells in a memory to be written into or to be read from. In order to facilitate selection, memories are generally arranged by placing cells in a rectangular arrangement of rows and columns as shown in Fig. 13.11a. In this particular case, there are m rows and n columns, for a total of $n \times m$ cells in the memory.

The control circuitry that accompanies the basic memory array is designed such that if one and only one row line is activated and one and only one column line is activated, the memory cell at the intersection of these two lines is selected. For instance, in Fig. 13.11b, if row A is activated and column B is activated, the cell at the intersection of this row and column is selected—that is, it can be read from or written into. For convenience, this cell is then called AB, corresponding to the row and the column selected. This designation is defined as the *address* of the cell. The activation of a line (row or column) is achieved by placing a logic 1 (or perhaps a logic 0) on it.

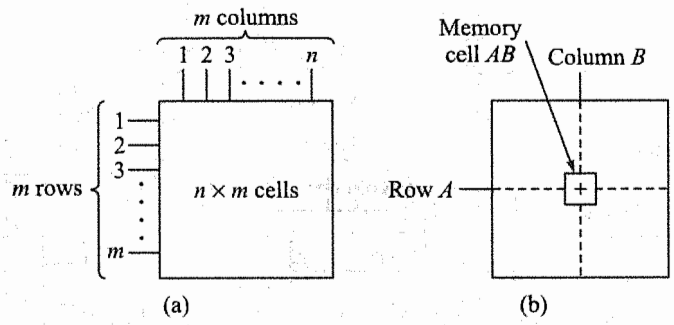


Fig. 13.11 **(a) A rectangular array of $m \times n$ cells, (b) Selecting the cell at memory address AB**

Matrix Addressing

Let's take a little time to consider the various possible configurations for a rectangular array of memory cells. The different rectangular arrays of 16 cells are shown in Fig. 13.12. In each of the five cases given, there are exactly 16 cells. The 16×1 and the 1×16 arrangements in Fig. 13.12a are really equivalent; likewise, the 8×2 and the 2×8 are essentially the same. So, there are really only three different configurations, each of which contain the exact same number of cells.

For any of the three configurations, the selection of a single cell still requires a single row and a single column to define a unique address. In Fig. 13.12a, a total of 17 address lines must be used—16 rows and 1 column, or 1 row and 16 columns. The minimum requirement in either case is really only 16 lines. However, either arrangement in Fig. 13.12b requires only 10 address lines—8 rows and 2 columns, or 2 rows and 8 columns. Clearly the best arrangement is given in Fig. 13.12c, since this configuration only requires 8 address lines—4 rows and 4 columns!

In general, the arrangement that requires the fewest address lines is a square array of n rows and n columns for a total memory capacity of $n \times n = n^2$ cells. It is exactly for this reason that the square configuration is so widely used in industry. This arrangement of n rows and n columns is frequently referred to as *matrix addressing*. In contrast, a single column that has n rows (such as the 16×1 array of cells) is frequently called *linear addressing*, since selection of a cell simply means selection of the corresponding row, and the column is always used.

For instance, a 74S201 is a 256-bit bipolar RAM, arranged in a 256×1 array. The IEEE symbol for the 74S201 ('S201) is given in Fig. 13.13 on the next page. Eight address lines (A_0, A_1, ..., A_7) are required to select one of the 256 cells. There are three chip select lines ($\overline{S}_1$, $\overline{S}_2$ and $\overline{S}_3$), all of which must be low in order to activate (select) the chip. When the $R/\overline{W}$ line is high, the data bit at input D is stored at the selected address. When the $R/\overline{W}$ line is low, the *complement* of the bit at the selected address appears at the $\overline{Q}$ output.

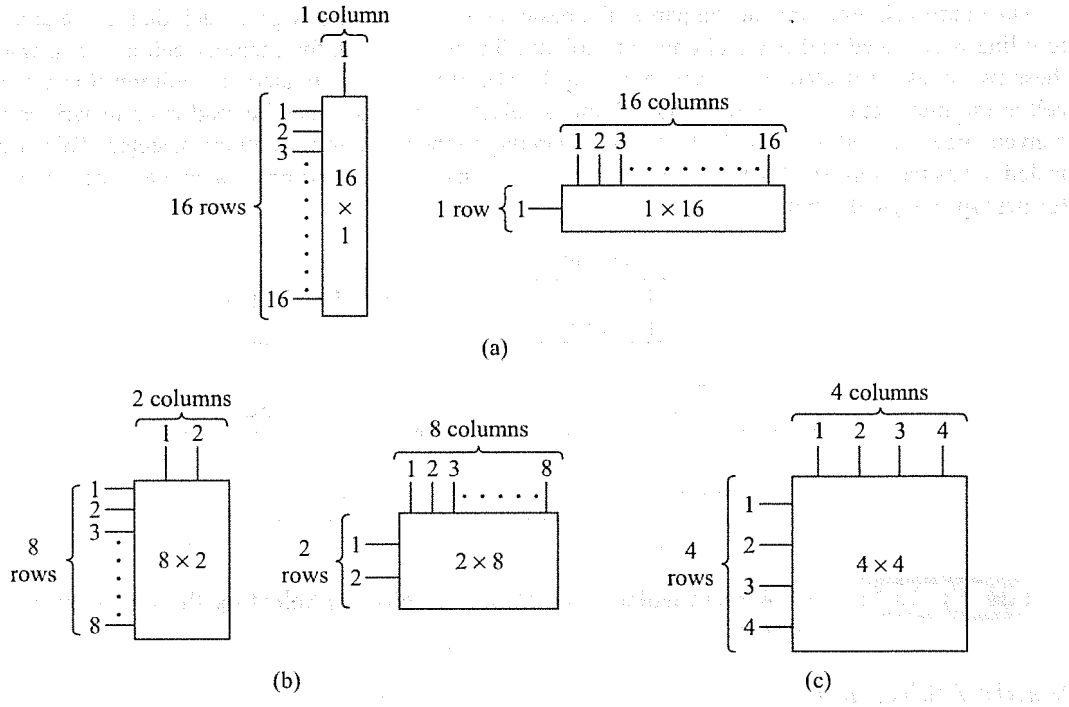


Fig. 13.12

The small triangle (∇) at the $\overline{Q}$ output means that the output is three-state (tri-state). We'll use this chip in Sec. 13.5.

Address Decoding

Take another look at the 4 × 4 memory in Fig. 13.12c. To select a single cell, we must activate one and only one row, and one and only one column. This suggests the use of two 1 of 4 binary to decimal decoders as shown in Fig. 13.14. Consider the selection of the cell at address 43 (row 4 and column 3). If $A_4 = 1$ and $A_3 = 1$, the decoder will hold the row 4 line high while all other row lines will be low. Similarly, if $A_2 = 1$ and $A_1 = 0$, the decoder will hold column 3 high and all other column lines low. Thus an input $A_4A_3A_2A_1 = 1110$ will select cell 43. We can consider A_4A_3 as a row address of 2 bits and A_2A_1 as a column address of 2 bits. Taken together, any cell in the array can be uniquely specified by the 4-bit address $A_4A_3A_2A_1$. As another example, the address $A_4A_3A_2A_1 = 0110$ selects the cell at row 2 and column 3 (address 23).

The address decoders shown in Fig. 13.14 further reduce the number of address lines needed to uniquely locate a memory cell, and they are almost always included on the memory chip. Recall that

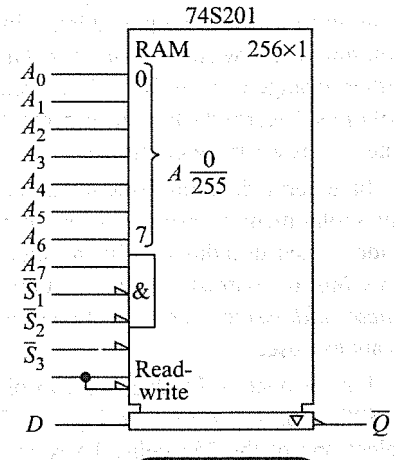


Fig. 13.13

a binary-to-decimal decoder having n binary inputs will select one of 2^n output lines. For instance, a decoder that has 3 binary inputs will have $2^3 = 8$ outputs, or a decoder having 4 inputs will have 16 outputs, and so on.

In general, an address of B bits can be used to define a square memory of 2^B cells, where there are $B/2$ bits for the rows and $B/2$ bits for the columns, as shown in Fig. 13.15. Notice that the total number of address bits B must be an even integer (2,4,6,8, ...). Since the input to each decoder is $B/2$ bits, the output of each decoder must be $2^{B/2}$ lines. So the capacity of the memory must be $2^{B/2} \times 2^{B/2} = 2^B$. For instance, an address of 12 bits can be used in this way for a memory that has $2^{12} = 4096$ bits. There will be 6 address bits providing $2^6 = 64$ rows and likewise 6 address bits providing 64 columns. The memory will then be arranged as a square array of $64 \times 64 = 4096$ memory cells.

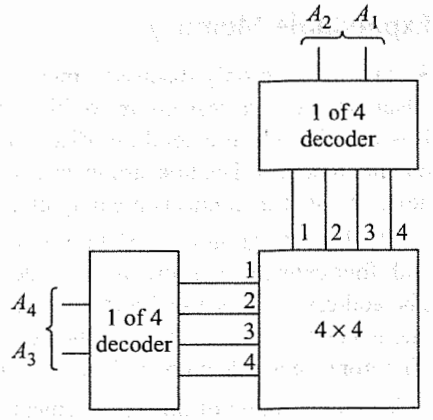


Fig. 13.14

You may have noticed that most commercially available memories have capacities like 1024, 2048, 4096, 16,384, and so on. The reason for this is now clear—all of these numbers are clearly integer powers of 2! Incidentally, a memory having 1024 bits is usually referred to as a 1K memory (1000 bits) simply for convenience. Similarly, a memory advertised as 16K really has 16,384 bits. 4K is really 4096, and so on.

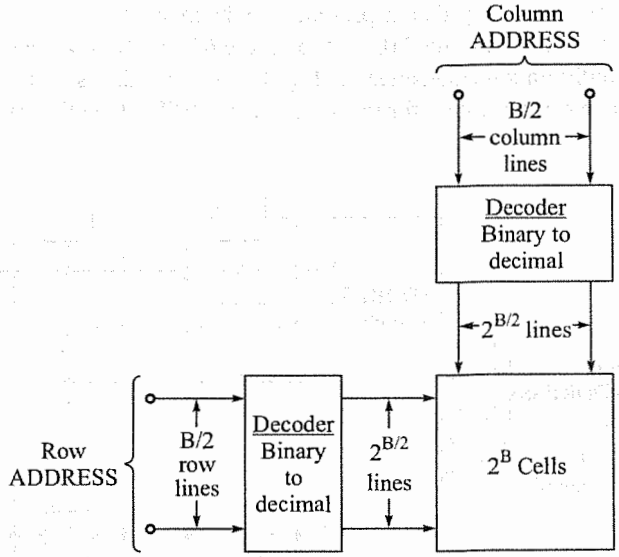


Fig. 13.15

Example 13.2 What would be the structure of the binary address for a memory system having a capacity of 1024 bits?

Solution Since $2^{10} = 1024$, there would have to be 10 bits in the address word. The First 5 bits could be used to designate one of the required 32 rows, and the second 5 bits could be used to designate one of the required 32 columns. Notice that $32 \times 32 = 1024$.

Example 13.3 For the memory system described in the previous example, what is the decimal address for the binary address 10110 01101? What is the address in hexadecimal?

Solution The first 5 bits are the row address. Thus row = 10110 = 22. The second 5 bits are the column address. So, column = 13. The decimal address is thus 22 13. In hexadecimal, this same address is 16 0D.

Expandable Memory

So far, we have only discussed memories that provide access to a single cell or bit at a time. It is often advantageous to access groups of bits—particularly groups of 4 bits (a nibble) and groups of 8 bits (a byte). It is not difficult to extend our discussion here to accommodate such requirements. There are at least two popular methods. The first simply accesses groups of cells on the same memory chip, and we discuss this idea next. The second connects memory chips in parallel, and we consider this technique in a following section.

The logic diagram for a 64-bit (16 × 4) bipolar memory is given in Fig. 13.16. There are 16 rows of cells with four cells in each row; thus the description (16 × 4). Each cell is a bipolar junction transistor flip-flop. The address decoder has 4 address bits and thus 16 select lines—one for each row. In this case, each select line is connected to all four of the cells in a row. So, each select line will now select four cells at a time. Therefore, each select line will select a 4-bit word (a nibble), rather than a single cell.

You might think of this arrangement as a "stack" of sixteen 4-bit registers. This is really a form of linear addressing, since the 4 address bits, when decoded, select one of the sixteen 4-bit registers. In any case, when data is read from this memory it appears at the four data output lines $\overline{D}_1$, $\overline{D}_2$, $\overline{D}_3$, and $\overline{D}_4$ as a 4-bit data word. Similarly, data is presented to the memory for storage as a 4-bit data word at input lines I_1, I_2, I_3, and I_4. The 74S89 and the 74LS189 both are 64-bit (16 × 4) bipolar scratch pad memories arranged in exactly this configuration (look ahead in Fig. 13.22). The idea is easily extended to memories that access a word of 8 bits (a byte) at a time—for instance, the TBP18S030 ROM discussed in the next section.

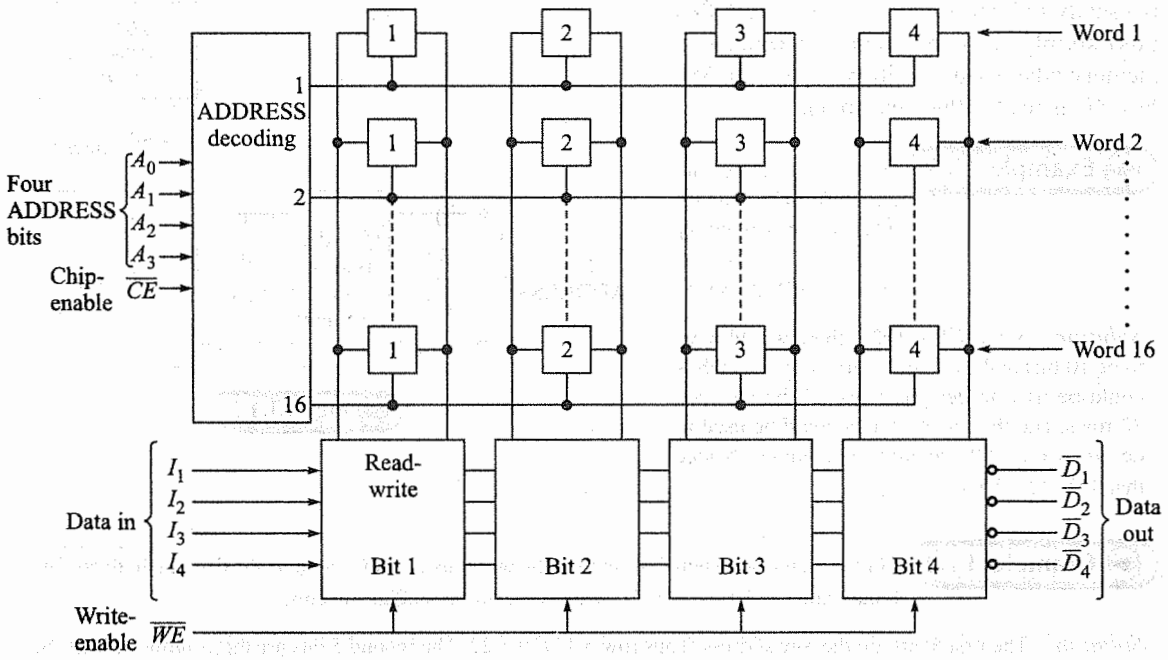


Fig. 13.16 64-bit (16 × 4) memory

SELF-TEST

10. What binary address will select cell 145 (decimal) in the 74S201 in Fig. 13.13?
11. The address applied in Fig. 13.14 is $A_4A_3A_2A_1 = 1010$. What cell is being accessed?

13.5 ROMs, PROMs, AND EPROMs

Having gained an understanding of memory addressing, let's turn our attention to the operation of a ROM. The term ROM is generally reserved for memory chips that are programmed by the manufacturer. Such a chip is said to be *mask-programmable*, in contrast to a PROM, which is said to be *field-programmable*—that is, it can be programmed by the user. EPROMs can be programmed, erased, and programmed again; they are clearly much more versatile chips than PROMs. Refer to Section 4.9 of Chapter 4 for a detailed discussion on its internal circuitry.

Programming

What exactly does programming a ROM, PROM or EPROM involve? It simply involves writing, or storing, a desired pattern of 0s and 1s (data). Each cell in the memory chip can store either a 1 or a 0. As supplied from the manufacturer, most chips have a 0 stored in each cell. The chip is then programmed by entering 1s in the appropriate cells. For instance, the content of every 4-bit word in a 64 × 4 chip is initially 0000. If the desired content of a word is to be 0110, then the two inner bit positions will be altered to 1s during programming.

In the case of a ROM, you must supply the manufacturer with the exact memory contents to be stored in the ROM. The Texas Instruments TMS4732 is a ROM having 4096 eight-bit words (a 4096 × 8 ROM). The logic diagram is given in Fig. 13.17. The 8-bit word length makes this NMOS (*n*-channel MOS) chip ideal for microprocessor applications. Texas Instruments will store user-specified data during manufacturing. The user must supply data storage requirements in accordance with detailed instructions given on the TMS4732 data sheet.

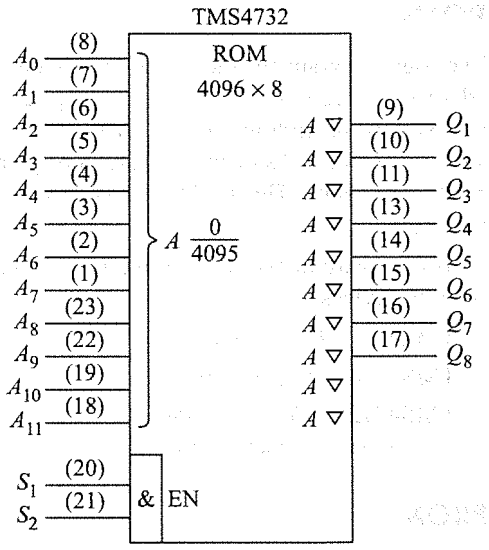


Fig. 13.17

The Texas Instruments TBP18S030 is a bipolar memory chip arranged as thirty-two 8-bit words (256 bits). The logic diagram for this user-programmable PROM chip is given in Fig. 13.18. Basically, the programming is done by applying a current pulse to each output terminal where a logic 1 must appear (be stored). The current pulse will destroy an *existing fuse link*. When the fuse link is present, the transistor circuit in that cell stores a 0. After the fuse link is destroyed, the circuit stores a 1. A typical programming sequence might be:

1. Apply the proper dc power supply voltage(s) to the chip; in the case of the TBP18S030, +5 Vdc. Disable the chip (the enable input is high).

2. Apply the address of the word to be programmed (A_0, A_1, A_2, A_3, A_4). For instance, to programm the word at address 14H (hex 14). apply

$$A_0A_1A_2A_3A_4 = 10100$$

3. To store the word $Q_0Q_1Q_2Q_3Q_4Q_5Q_6Q_7$ = 00101000:
   a. Ground output Q_2 and connect all other outputs to +5 Vdc through a 3.9-kΩ resistor. Raise the +5-Vdc supply to +9.25 Vdc and momentarily enable the chip. This will program a 1 in bit position Q_2.
   b. Repeat (a) for bit position Q_4. This will program a 1 in bit position Q_4.
4. Repeat steps 2 and 3 for each word to be programmed.

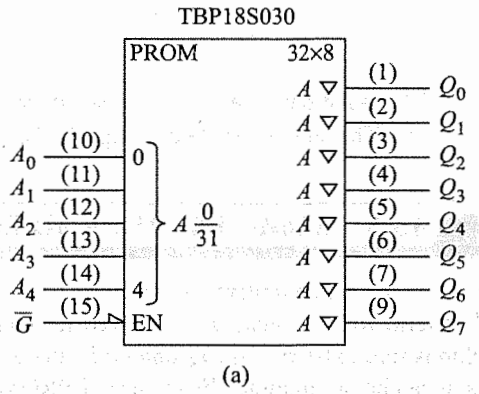


Fig. 13.18 (a) Logic symbol

ROMs

The logic diagram for the Texas Instruments TMS4732, a 4096 × 8 ROM, is given in Fig. 13.17. Twelve address bits are required, A_0, A_1, ... , A_{11} (2^{12} + 4096). There are two chip-enable inputs, S_1 and S_2. Both S_1 and S_2 must be high in order to enable the chip. Each of the eight data output lines is a three-state line (the small ∇ symbol). As mentioned previously, this chip is ideal for microprocessor applications because of the 8-bit word length. This ROM is mask-programmable, and data must be specified for the manufacturer before purchase.

Texas Instruments offers a number of other ROMs with larger memory capacity, all of which are LSI NMOS devices.

TMS4664: 8192 × 8-bit
TMS4764: 8192 × 8 bit
TMS47128: 16,384 × 8-bit
TMS47256: 32,768 × 8-bit

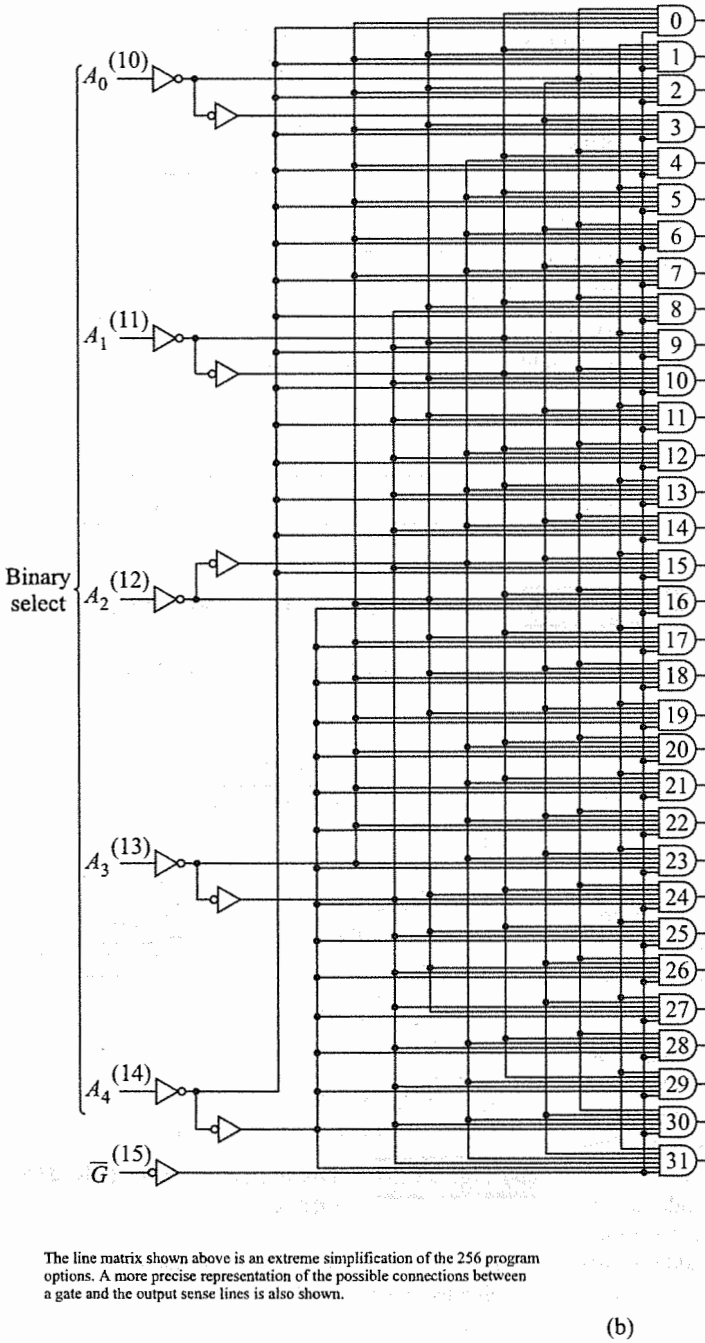
PROMs

The TBP18S030 is a 256-bit (32 × 8) PROM arranged as a stack of thirty-two 8-bit words. The 74S288 is an equivalent designation. As shown in Fig. 13.18, the 5 row address bits are labeled A_0, A_1, A_2, A_3, A_4 and the 8 output bits in a word are labeled Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7.

Input $\overline{G}$ is used to enable or disable the entire set of 32 input decoding gates. When $\overline{G}$ is high, all the address decoding gates are inhibited and the memory chip is disabled, causing the eight output data bit lines to be high. When G is low, the data at the outputs will correspond to the 8-bit word in memory selected by the input address. On most memory chips there is a chip-enable or chip-select input line that performs the same function as $\overline{G}$.

Using the TBP18S030 PROM is relatively simple. First, since the logic circuits are TTL, a supply voltage and ground connections must be made. The data sheet calls for a nominal supply voltage of $+V_{CC}$ = 5.0 Vdc

Functional block diagram and word selection



Word select table

Word	Inputs				
	A_4	A_3	A_2	A_1	A_0
0	L	L	L	L	L
1	L	L	L	L	H
2	L	L	L	H	L
3	L	L	L	H	H
4	L	L	H	L	L
5	L	L	H	L	H
6	L	L	H	H	L
7	L	L	H	H	H
8	L	H	L	L	L
9	L	H	L	L	H
10	L	H	L	H	L
11	L	H	L	H	H
12	L	H	H	L	L
13	L	H	H	L	H
14	L	H	H	H	L
15	L	H	H	H	H
16	H	L	L	L	L
17	H	L	L	L	H
18	H	L	L	H	L
19	H	L	L	H	H
20	H	L	H	L	L
21	H	L	H	L	H
22	H	L	H	H	L
23	H	L	H	H	H
24	H	H	L	L	L
25	H	H	L	L	H
26	H	H	L	H	L
27	H	H	L	H	H
28	H	H	H	L	L
29	H	H	H	L	H
30	H	H	H	H	L
31	H	H	H	H	H

H = high level
L = low level

The line matrix shown above is an extreme simplification of the 256 program options. A more precise representation of the possible connections between a gate and the output sense lines is also shown.

(b)

Fig. 13.18 **(b) TBP18S030 PROM (74S288)**

on pin 16, with ground connected to pin 8. The inputs and outputs are all TTL-compatible. The eight data outputs are three-state (note the symbol ∇ at each output).

Now, all that is required is to apply the correct input address to read a desired 8-bit word and then take the $\overline{G}$ input line (select line) low. The TBP18S030 data sheet states an access time t_p of 12 ns (typical) and 25 ns (maximum). So, an 8-bit data word will be available at the outputs $Q_0 \ldots Q_7$ within 25 ns after the falling edge of $\overline{G}$, as shown in Fig. 13.19a. The address lines should, of course, be stable during the time data is being read out of the memory. There are two output lines in Fig. 13.19a showing that a data line may transition low to high, or high to low. To save time and space, this idea is usually conveyed in a single waveform as seen in Fig. 13.19b—this single waveform is the equivalent of the two output waveforms above it in Fig. 13.19a.

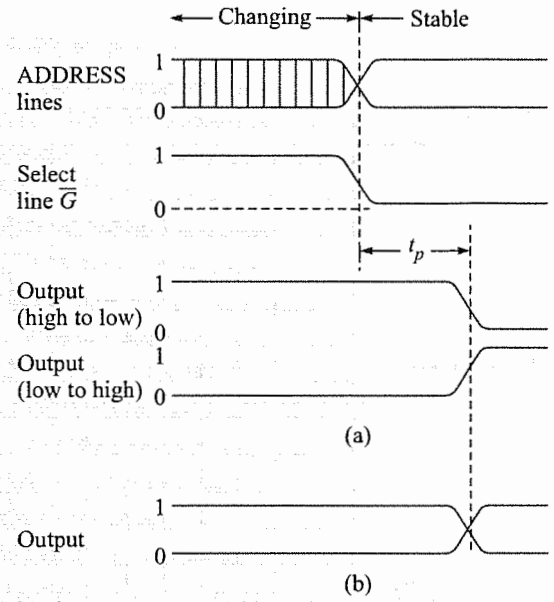


Fig. 13.19 **Access time to for a TPB18S030 PROM**

Example 13.4 The TTL LSI TBP24S10 is advertised as a 1024-bit PROM. Since $2^{10} = 1024$, it would seem to require 10 address bits, but the data sheet shows only 8 bits of address. Can you explain how the memory on this chip must be organized by looking at the logic diagram in Fig. 13.20?

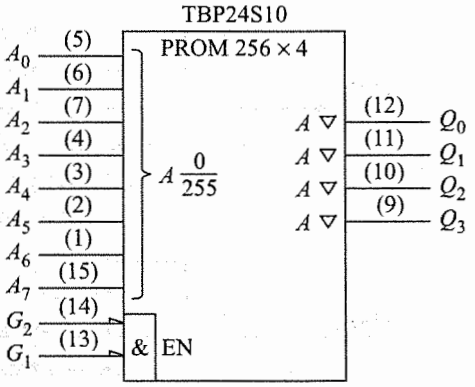


Fig. 13.20 **Texas Instruments TPB24S10 PROM**

Solution There are 4 bits appearing at the output of the chip, so it must be organized as 256 words of 4 bits each ($256 \times 4 = 1024$). The 1024 memory cells are arranged in a square consisting of 32 rows and 32 columns. Five of the address bits (*DEFGH*) are used to select one of the 32 rows ($2^5 = 32$). The 32 columns are divided into eight groups of 4 bits each. So, it is only necessary to select one of the eight groups, and this can be done with three address lines (*ABC*), since $2^3 = 8$. As an example, the address *HGFEDCBA* = 10110 110 will select row 10110 = 22 and the 4 bits (four columns) in group 110 = 6.

EPROMs

One disadvantage of a PROM is that once it is programmed, the contents are stored in that memory chip permanently—it can't be changed; a mistake in programming the chip can't be corrected. The EPROM overcomes this difficulty.

The EPROM has a structure and addressing scheme similar to those of the previously discussed PROM, but it is constructed using MOS devices rather than bipolar devices. Many MOS EPROMs are TTL-compatible, and even the technique used to program the chip is similar to that used with a bipolar memory. The only difference is really the mechanism for permanently storing a 1 or a 0 in an MOS memory cell.

The current pulse used to store a 1 when programming a bipolar PROM is used to destroy ("burn out") a connection on the chip. The same technique is used to program an MOS-type EPROM, but the current pulse is now applied for a period of time (usually a few milliseconds) in order to store a fixed charge in that particular memory cell. This stored charge will cause the cell to store a logic 1.

The interesting thing about this phenomenon is that the charge can be removed (or erased), and the cell will now contain a logic 0! Furthermore, the process can be repeated. The memory cells are "erased" by shining an ultraviolet light through a quartz window onto the top of the chip. The light bleeds off the charge and all cells will now contain 0s. The requirements for programming and erasing an EPROM vary widely from chip to chip, and data sheet information must be consulted in each individual case.

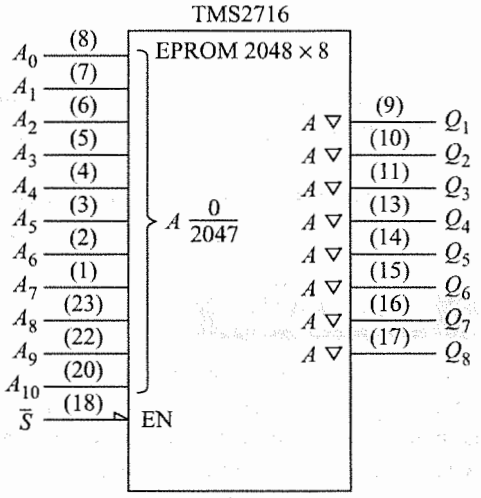


Fig. 13.21 2716 EPROM

The logic diagram for a 2716, a 16K (2K × 8) EPROM, is given in Fig. 13.21. There are actually 2048 words, 8 bits each, for a total storage capacity of 16,384 bits. The chip is completely TTL-compatible and has an access time of less than 450 ns. The 11 address bits will uniquely select one of 2048, 8-bit words (2^{11} = 2048), and the selected word will appear at the data output lines if chip-select is low. The 2732 is a 32K (4K × 8) EPROM that is pin-compatible with the 2716—it simply has twice the memory storage. Likewise, the 2764 is a 64K (8K × 8) EPROM.

As a matter of fact, the logic diagram in Fig. 13.21a is virtually the same for any ROM, PROM, or EPROM. Essentially the only variation is the total number of address inputs to accommodate the number of bits in the cell matrix. As an example, a CMOS EPROM with essentially the same logic diagram is the 27C512. But this chip has 16 address input lines and a 534,288-bit cell matrix, organized as 65,536 words, each 8 bits in length. This is most impressive when you consider that there are over a half-million bits of information stored in a 28-pin package that measures less than 1.5 in. in length and about 0.5 in. in width!

EEPROM, Flash Memory

Electrically Erasable Programmable Read Only Memory (EEPROM) is similar to EPROM as far as writing into memory is considered, i.e. effecting a current pulse to store charge. The erasing, however, is different and is done by removing the charge and sending a pulse of opposite polarity. There are two types of EEPROM—parallel and serial. Parallel EEPROM is faster, costlier and comes in 28xx family. Their pinout and functioning

is similar to 27xx EPROM family. Serial EEPROM is slower, cheaper, uses lower number of pins and comes in 24xx family.

Flash memory is a further advancement on EEPROM. This, too, writes and erases data electrically—can be both parallel and serial type. The number of write/erase cycle is finite and often, there is a separate management scheme to take note of this. There is an internal voltage generation block that takes single voltage supply and generates different voltages required for writing and erasing. Different manufacturers have created different standards for flash memory chip which differ in pinout, memory organization, etc. Intel family chips are 28Fxxx while AMD chips are numbered as 29Fxxx.

SELF-TEST

12. What does it mean to say that a chip is mask-programmable?
13. What is the meaning of the small triangle on each output line of the TMS4732 in Fig. 13.17?
14. The 74S288 in Fig. 13.18 is programmed with 1011 0001 in word 20. The desired content at this word address is 1011 1001. Can this be corrected?

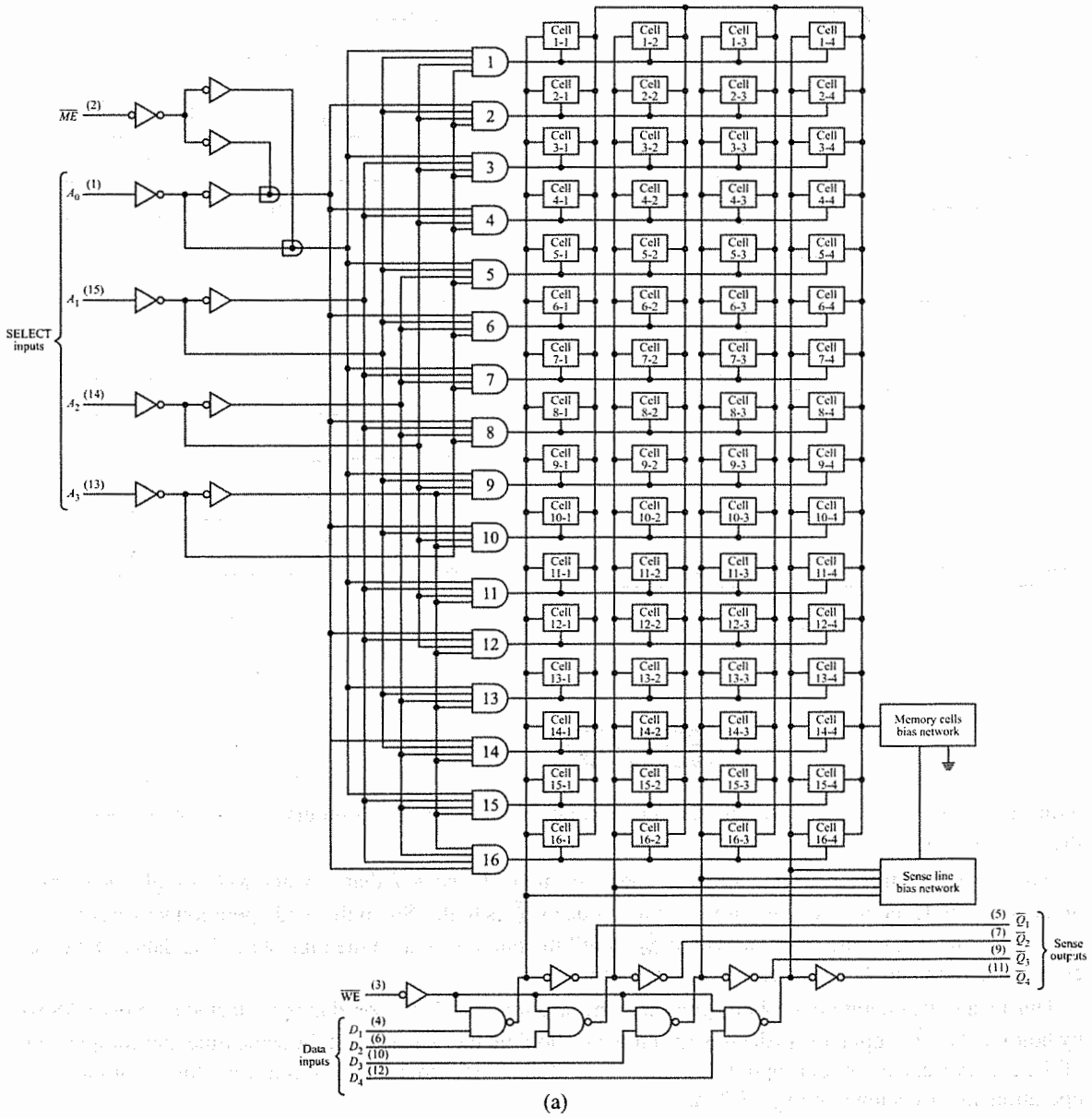
13.6 RAMs

The basic difference between a RAM and a ROM is that data can be written into (stored in) a RAM at any address as often as desired. Naturally data can be read from any address in either a RAM or a ROM, and the addressing and read cycles for both devices are similar. The characteristics of both bipolar and MOS "static" RAMs are discussed in this section.

A static RAM (SRAM) uses a flip-flop as the basic memory cell (either bipolar or MOS) and once a bit is stored in a flip-flop, it will remain there as long as power is available to the chip—essentially forever—thus the term "static." On the other hand, the basic memory cell in a "dynamic" RAM (DRAM) utilizes stored charge in conjunction with an MOS device to store a bit of information. Since this stored charge will not remain for long periods of time, it must periodically be recharged (refreshed), and thus the term "dynamic" RAM. Both static and dynamic RAMs are "volatile" memory storage devices, since a loss of power supply voltages means a loss of stored data. In this section we discuss SRAMs in detail that explains how a RAM unit works and how several RAM chips can be combined together to expand the memory capacity, For this purpose we'll use mostly the TTL devices however a brief discussion of MOS based SRAM and DRAM will also be presented.

The 7489

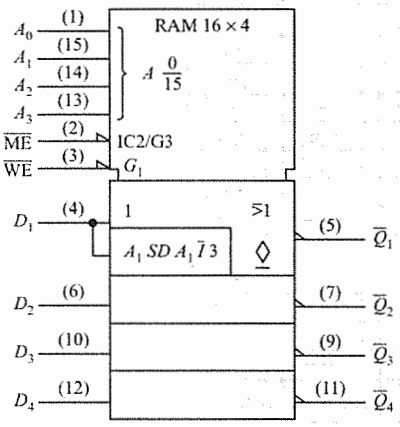
The 7489 shown in Fig. 13.22 is a TTL LSI 64-bit RAM, arranged as 16 words of 4 bits each. Holding the memory-enable ($\overline{\text{ME}}$) input low will enable the chip for either a read or a write operation, and the four data address lines will select which one of the sixteen 4-bit word positions to read from or write into. Then, if the write-enable ($\overline{\text{WE}}$) is held low, the 4 bits present at the data inputs (D_1, D_2, D_3, D_4) will be stored in the selected address. Conversely, if $\overline{\text{WE}}$ is high, the data currently stored in the memory address will be presented to the four data output lines ($\overline{Q}_1$, $\overline{Q}_2$, $\overline{Q}_3$, $\overline{Q}_4$). Incidentally, the outputs are open-collector transistors, and a pull-up resistor from each output up to $+V_{CC}$ is normally required. The operations for this chip are summarized in the truth table in Fig. 13.22.



(a)

Fig. 13.22 7489, 64-bit RAM

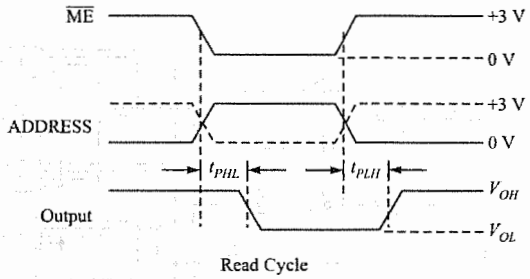
The read operation is no different from that for a ROM. For this chip, simply hold $\overline{\text{ME}}$ low and $\overline{\text{WE}}$ high, and select the desired address. The 4-bit data word then appears at the "sense" outputs. The timing for a read operation is shown by the waveforms in Fig. 13.22d. The propagation delay time t_{PHL} is that period of time from the fall of $\overline{\text{ME}}$ until stable data appears at the outputs—the data sheet gives a maximum value of 50 ns, with 33 ns typical. Naturally the address input lines must be stable during the entire read operation, beginning



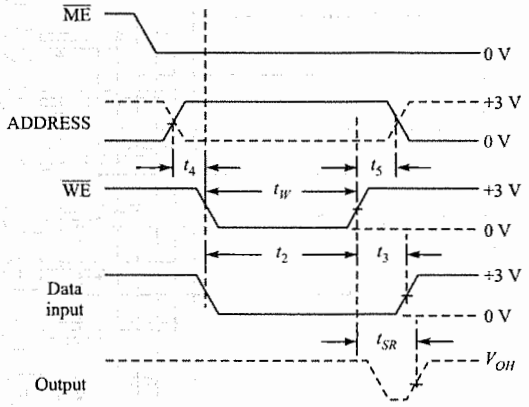
(b)

$\overline{ME}$	$\overline{WE}$	Operation	Condition of outputs
L	L	Write	Complement of data inputs
L	H	Read	Complement of selected word
H	L	Inhibit storage	Complement of data inputs
H	H	Do nothing	High

(c)



Read Cycle



Write cycle

(d)

Fig. 13.22 (Continued)

with the fall of $\overline{ME}$. Notice carefully that the data appearing at the four outputs will be the *complement* of the stored data word!

You will notice from the truth table that when the chip is *deselected*, that is, when $\overline{ME}$ is high, the outputs all go to a high level, provided we are in a read mode ($\overline{WE}$ is high). So, in the read operation waveforms, the time t_{PLH} is the delay time from the rise of $\overline{ME}$ until the outputs assume the high state. The data sheet gives 50 ns maximum and 26 ns typical for this delay time.

During a write operation the 4 bits present at the data inputs will be stored in the selected memory address by holding the $\overline{ME}$ input low (selecting the chip) and holding the $\overline{WE}$ low. At the same time, the complement of the data present at the four input lines will appear at the four output lines. Timing waveforms for the write operation are also shown in Fig. 13.22d.

Let's look carefully at the timing requirements for the write cycle. First, the $\overline{WE}$ must be held low for a minimum period of time in order to store information in the memory cells—this is given as time t_W on the waveforms, and the data sheet calls for 40 ns minimum. Memory-enable selects the chip when low, and is allowed to go low coincident with or before a write operation is called for by $\overline{WE}$ going low.

Next, the data to be written into memory must be stable at the data inputs for a minimum period of time before $\overline{WE}$ and, also for a minimum period of time after $\overline{WE}$. The time period prior to $\overline{WE}$ is called the *data-setup time* t_2. This time is measured from the end of the write-enable signal back to the point where the

data must be stable. The data sheet calls for 40 ns, and in this case it is the same as t_w. Also, the data inputs must be held stable for a period of time after $\overline{WE}$ rises—this is called the *data-hold time* t_3, and the data sheet calls for 5 ns minimum.

The address lines must also be stable for a period of time before as well as after the $\overline{WE}$ signal. The time period before $\overline{WE}$ is called the *address-setup* or *select-setup time* t_4. This time is measured from the fall of $\overline{WE}$ back to the point where the input address lines must be stable; the data sheet calls for 0.0 ns minimum. In other words, the address lines are allowed to become stable coincident with or before $\overline{WE}$ goes low. The address lines must also be stable for a period of time after the rise of $\overline{WE}$; this is called the *address-hold* or *select-hold* time t_5, and the data sheet calls for 5 ns minimum.

Finally, after a write operation, if the chip is deselected ($\overline{ME}$ goes high), the outputs will return to a high state. The maximum time for this to occur is the sense-recovery time t_{SR}, given as 70 ns maximum on the data sheet.

The operation of a 7489 ns straightforward and easy to understand; therefore it is a good chip to study in elementary discussions of RAMs. It can be used to construct memories having larger capacities by connecting chips in parallel, but it's not too practical when we wish to consider memories of 16K, 32K, ... , 256K, 512K, and so on. Nevertheless, the time spent studying this chip is well invested since the fundamentals of addressing and the read and write operations are essentially the same for all static RAMs. So, with these fundamentals in mind, let's take a look at some chips that have more memory capacity.

The 74S201

The block diagram in Fig. 13.23 can be used to describe the operation of most SRAMs. Most of these are constructed with n address lines that will uniquely select only one of the 2^n cells in the memory array—that is, selection is 1 bit at a time. There will be a chip-enable control (CE), a write-enable (WE), and a provision for a single input data bit (D_i) and a single output data bit (D_o).

For instance, the 74S201 in Fig. 13.24 is a 256-bit RAM, organized as 256 words, each 1 bit in length. The 256 cells are arranged in a square array of 16 rows and 16 columns. The 8 address bits ($2^8 = 256$) are divided into 4 bits that are decoded to select one of the 16 rows and 4 bits that are decoded to select one of the 16 columns. There is a single input data bit (D), a single output data bit ($\overline{Q}$), and a read-write line (R/$\overline{W}$). There are three memory-enable inputs ($\overline{S}_1$, $\overline{S}_2$, $\overline{S}_3$), and all three of them must be low to select or enable the chip.

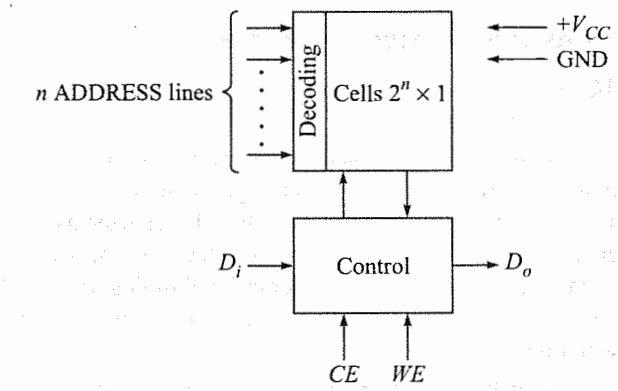
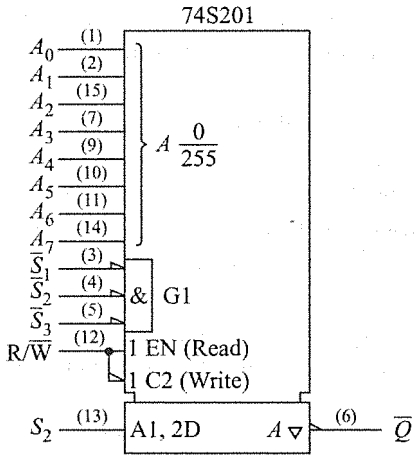


Fig. 13.23 **Generalized block diagram for a static RAM**

The truth table shows that if the chip is enabled, a write cycle is initiated by holding R/$\overline{W}$ low, or a read cycle can be initiated by holding R/$\overline{W}$ high. If any or all of the read-write inputs are high, the chip is inhibited and the output goes to a high-impedance state. Naturally the proper timing must be observed as defined by the timing waveforms; you will see that the timing requirements are very similar to the previously described 7489.



(a) Logic symbol

Write-enable pulse width (minimum)		65 ns
Setup time	ADDRESS before write	65 ns
	Data before end of write	65 ns
	Chip-select before end of write	65 ns
Hold time	ADDRESS after write	0 ns
	Data after write	0 ns
	Chip-select after write	0 ns

(b) Recommended timing

Fig. 13.24

Example 13.5 Using the information in Fig. 13.24, determine how long the address lines for the 74S201 must be held before R/$\overline{W}$ goes low and after R/$\overline{W}$ goes high.

Solution The setup time, address to write-enable, is 0.0 ns. The hold time, address from write-enable, is 0.0 ns. Therefore, the address lines must be stable from the fall of R/$\overline{W}$ until the rise of R/$\overline{W}$.

Formation of Memory Banks

Memory bank is the concept of increasing memory's capacity by connecting more than one memory block in series, parallel or both.

Now that we understand the operation of the 74S201 (abbreviated as '201), it is a simple matter to use multiple '201 chips to construct larger memories. For instance, we can connect four '201 chips in parallel as shown in Fig. 13.25 to construct a RAM organized as 256 words, each 4 bits in length. Connecting eight '201 chips in parallel will form a memory having 8-bit words, and so on. The nice thing about connecting chips in parallel like this is that the control and timing are exactly the same as if there were only a single chip. The only difference is that there are 4 data bits in and 4 data bits out (or 8 in and 8 out), all of which are in parallel with one another.

As a matter of fact, even larger memories can be constructed by connecting basic chips such as the '201 in both series and parallel. For instance, thirty-two '201 chips are connected in a 4×8 matrix in Fig. 13.26 to form a memory having one thousand, twenty-four 8-bit words. This configuration requires a 10-bit address: 2 bits can be used to select one of the four rows of eight '201s, and the remaining 8 bits will be wired in parallel to all the chips; they will work exactly as for the two-hundred fifty-six 4-bit word memory in Fig. 13.25. This concept can be continued, of course, but it becomes somewhat impractical with larger memory requirements, especially since there are MOS chips readily available with greater memory capacity.

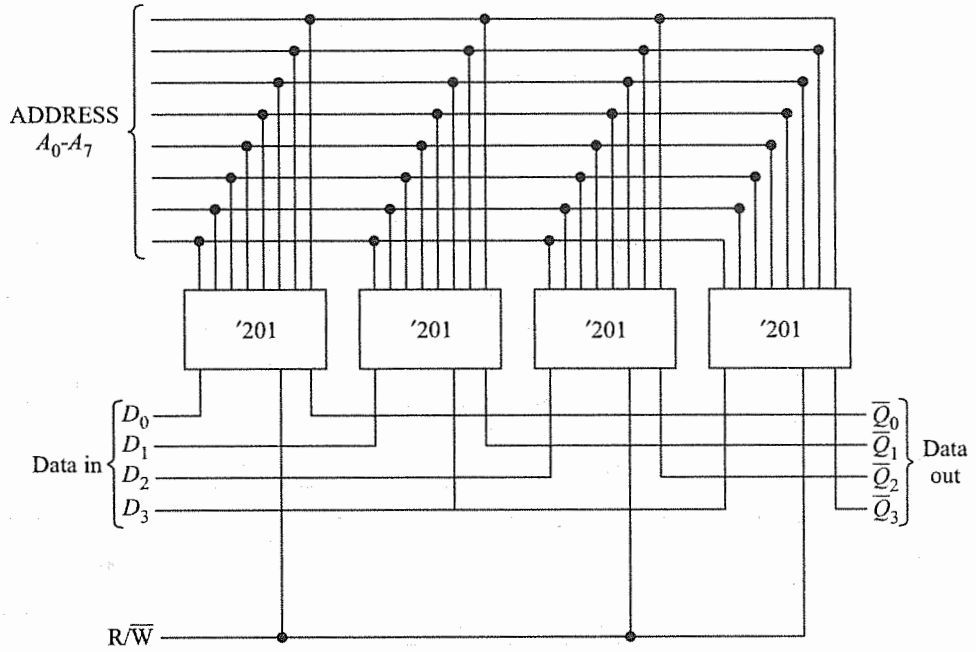


Fig. 13.25 **Four 74S201's arranged as a 1024-bit memory having 256**

SRAMs

A very popular and widely used MOS memory chip is the 2114. This is an SRAM having 4096 bits arranged as 1024 words of 4-bits each. The organization of this chip is quite similar to that of the 7489 shown in Fig. 13.22, but notice that the 2114 is sixteen times larger! Nearly all SRAMs larger than 1024 bits are MOS types.

The basic memory is arranged as 64 rows and 64 columns for a total of $64 \times 64 = 4096$ bits. Six address bits (A_3 through A_8) are used to select one of the 64 rows ($2^6 = 64$). The 64 columns are divided into 16 groups of 4-bit words, and four address bits (A_0, A_1, A_2 and A_9) are used to select one of these 16 groups. A 10-bit address will then select a single 4-bit word from 64 rows and 16 columns, to provide a memory of $64 \times 16 = 1024$ four-bit words.

A basic SRAM cell or latch is shown in Fig. 13.27a. It consists of a back-to-back inverter that latches on to a particular state of logic 0 or 1. The two pass transistors enable writing and reading from two bit lines. Both the transistors are 'on' for both reading and writing when this cell is selected after decoding the address. The bit lines during writing operations write its value into the latch while during reading sense the latch state. A typical SRAM requires six transistors per bit of memory—two pass transistors and two transistors each of the inverter. However, some implementations use only a single transistor per inverter with a total requirement of four transistors per bit.

DRAMs

A typical DRAM is essentially the same as the previously discussed SRAM chip, with the exception of the required refresh cycle. The 4116 is a widely used 16K (16,384 × 1) DRAM available from a number of

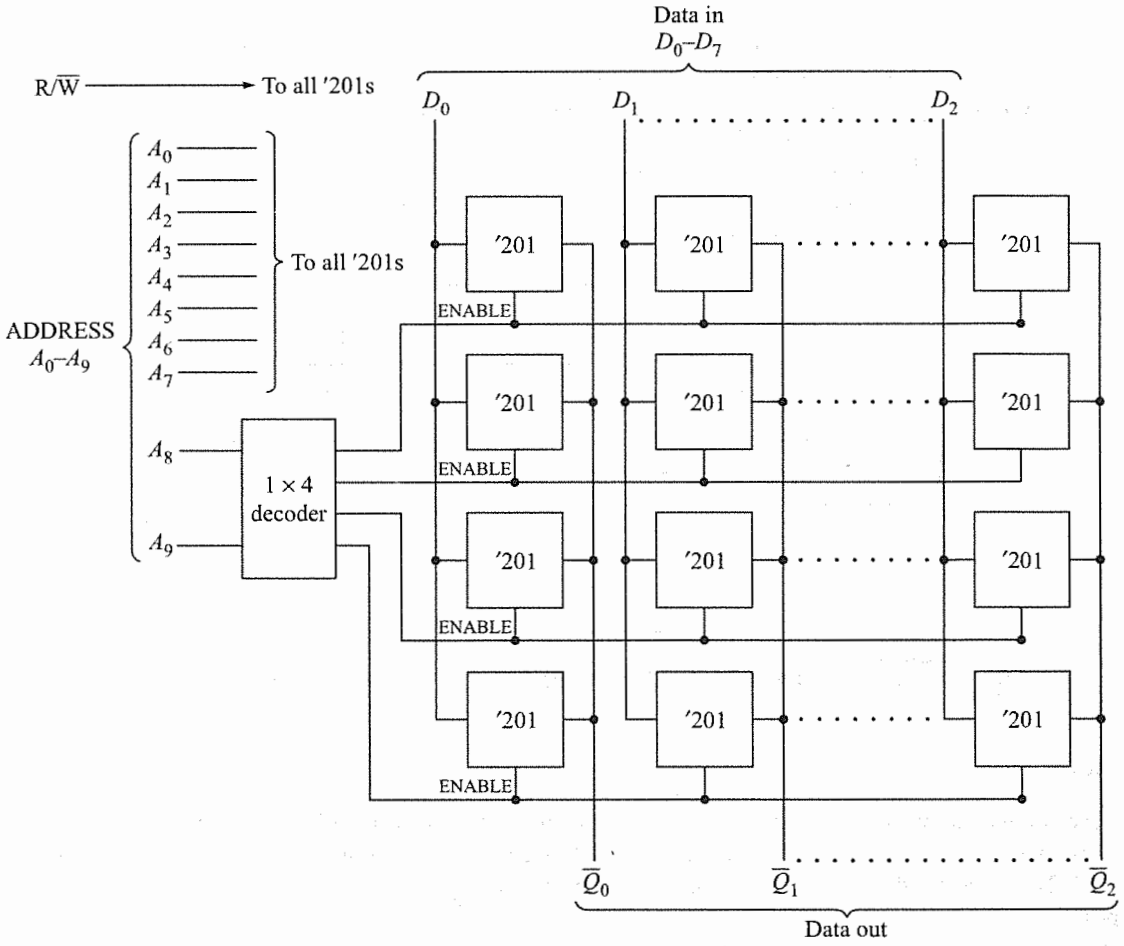


Fig. 13.26

different sources, such as Mostek (MK4116), Motorola (MCM4116), and Texas Instruments (TMS4116). Note that, 4116 must be refreshed at least once in every 2 ms. There is another widely used DRAM, the 4164, organized as a 64,536 × 1 chip. The operation of this chip is quite similar to 4116. The 64K DRAM is available under following part numbers: Texas Instruments TMS4164, Motorola MCM6665, Intel 2164, etc.

A basic DRAM cell is shown in Fig. 13.27b. A capacitor is used as a storage element, as it can store electrical charge but for a limited amount of time. This requires periodic *refresh* to replenish the charge and thus the RAM is always *dynamic*. The cell is selected by turning on the pass transistor. The bit line is used both for writing and reading (sensing). A faster writing ability requires the capacitor to be charged faster which in turn discharges the capacitor quickly requiring quicker refresh cycle. Compared to SRAM, DRAM uses less number of components that require less space, increasing the *packing density*. Also it is much less expensive. But SRAM scores over DRAM on a very important area. DRAM has a very high access time due

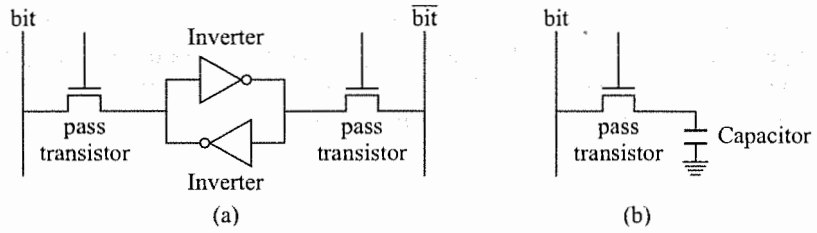


Fig. 13.27 (a) A basic SRAM cell, and (b) A basic DRAM cell

to high latency. This is why even if DRAM is used as a computer's main memory to make it cheap, SRAM is used as a *cache memory* for faster access instruction and data. Cache memory uses *locality* feature where a set of sequential instructions and data are found in contiguous locations in memory. A cache controller brings this memory block from main memory to cache (which is SRAM) for speedier operation of the computer.

13.7 SEQUENTIAL PROGRAMMABLE LOGIC DEVICES

We have discussed programmable devices (like PLA, PAL) for combinatorial circuits. In this section, we discuss similar devices available for sequential logic circuits. Architecture-wise, they additionally have memory elements like flip-flops. The simplest of the three widely used variety is called Simple Programmable Logic Devices (SPLD or simply PLD). Similar technology but using larger number of logic gates, suitable to address more complex sequential logic problems is called Complex Programmable Logic Devices (CPLD). The third type uses slightly different technology but of much higher capacity is known as Field Programmable Gate Array (FPGA). The term, High Capacity Programmable Logic Devices (HCPLD) is also used to refer to CPLD and FPGA together. Note that, Hardware Descriptions Languages (HDL), like Verilog, can be used to program these devices.

PLD

Refer to discussions of Section 4.10 and 4.11 on PAL and PLA and corresponding figures (Fig. 4.44, Fig. 4.47). Note that, output is taken from OR gates following AND plane generating combinatorial logic functions in SOP form. Each OR gate with added circuitry (as shown in Fig. 13.28) that includes a flip-flop, multiplexer and tri-state output forms a *macrocell* of PLD. The flip-flop can store the OR gate output indefinitely and is triggered by a Clock. Multiplexer selects either OR gate or flip-flop output which is also fed back to AND plane for internal use. The output buffer when enabled by *Enable* makes multiplexer output available to external world through output pin, else output pin is held at high impedance state.

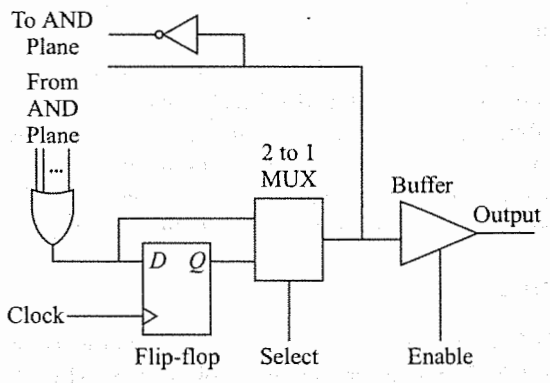


Fig. 13.28 A macrocell of PLD

Each PLD typically has 8–10 macrocells. Advanced Micro Devices (AMD) manufactured SPLDs 16R8 and 22V10 are PAL based. The name "16R8" means that the PAL has a maximum of 16 inputs

(there are 8 dedicated inputs and 8 input/outputs which can be configured either as input or output), and a maximum of 8 outputs. The "R" stands for PAL outputs registered as *D* flip-flop. Similarly, the "22V10" has a maximum of 22 inputs and 10 outputs and "V" stands for versatility of the output. The other manufacturers of popular SPLDs are Altera, Lattice, Cypress and Philips-Signetics.

CPLD

Simple PLDs can handle 10–20 logic equations. Thus, for more complex circuit design one needs to physically connect few such units. This problem is solved by the advent of CPLD, which consists of a number of PLD like blocks (Fig. 13.29). The blocks are interconnected among themselves through programmable switches present in interconnection block. This means it needs two levels of programming: one for programming PLD block the other for programming the switches. Input and output pins of a CPLD chip are routed through I/O blocks. Here, the macrocell has a two input Ex-OR gate after the OR gate. Depending on the value present in the other (Control) input, Ex-OR gate sends complemented or uncomplemented OR output to flip-flop and multiplexer. Commercial CPLDs can have up to 50 PLD blocks. Higher density is not supported by CPLD architecture and FPGAs are used for that. Transistors are used as programmable switches for CPLDs (and also for many SPLDs) by placing it between two wires in a way that facilitates implementation of wired-AND functions. EPROM used as switches does not support in-circuit programming but EEPROM does that. The advantage with them is that both are non-volatile in nature. Re-programmability feature of CPLD is a very useful advantage.

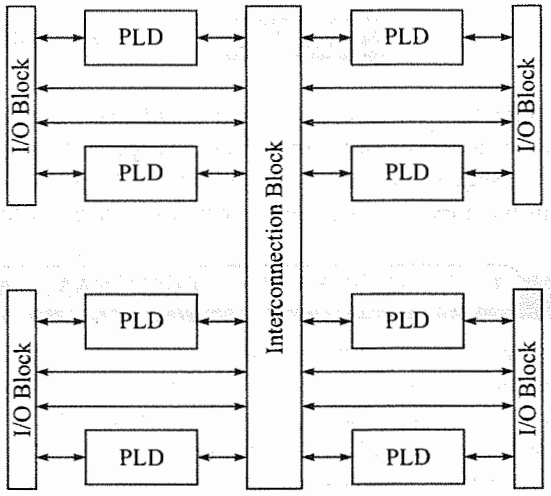


Fig. 13.29 **A typical block diagram representation of CPLD**

The applications of CPLDs can be found in reasonably complex designs, like graphics controller, UARTs, cache control and many others. Circuits that can exploit wide AND/OR gates, and do not need too many flip-flops are suited for CPLD implementation.

AMD offered CPLD family, Mach 1 to Mach 5 comprises multiple PAL-like blocks: Mach 1 and 2 consist of optimized 22V16 PALs, Mach 3 and 4 comprise several optimized 34V16 PALs and Mach 5 is similar but offers enhanced speed performance. Mach chips are based on EEPROM technology, Xilinx offers XC7000 and XC9500 where each chip consists of a collection of SPLD-like blocks with 9 macrocells in each. Altera has developed three families of chips that fit within the CPLD category: MAX 5000, MAX 7000, and MAX 9000. The other manufacturers of CPLD are Lattice, Cypress, etc.

FPGA

FPGA consists of an array of circuit elements called logic blocks, which unlike AND-OR combination of CPLD has programmable look up table (LUT). The look up table can generate any logic combination for the variables involved. A multiplexer based 2-variable look up table is shown in Fig. 13.30a. Based on what value (0 or 1) is stored at input this can generate any of the $2^4 = 16$ possible functions of A, B as

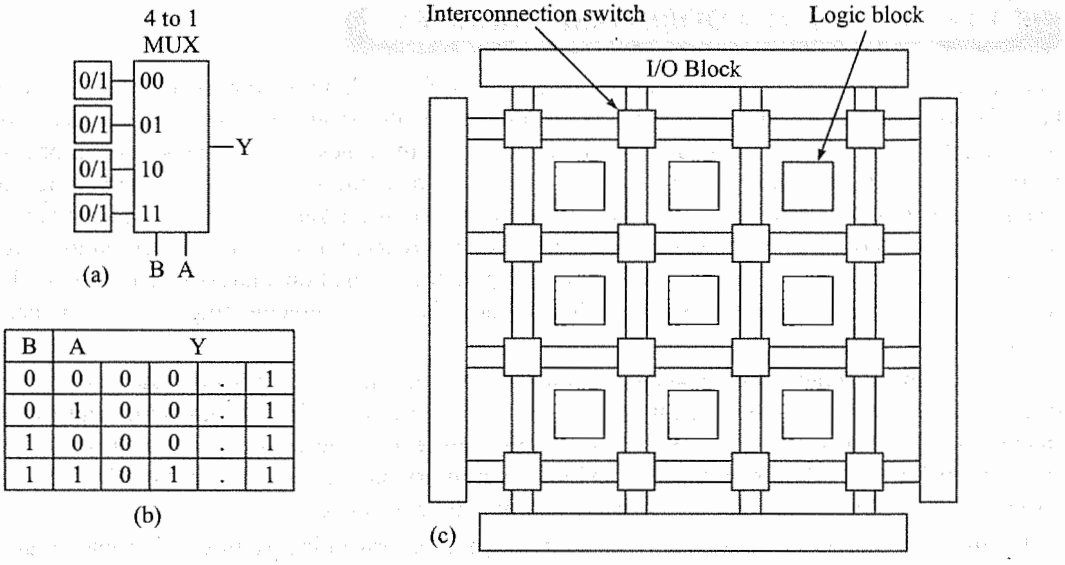


Fig. 13.30 **(a) A two variable programmable look up unit, (b) Generation of any two variable logic Y = *f* (A, B) by placing appropriate combination as input to 4 to 1 Multiplexer, (c) A typical structure of FPGA**

Y = *f* (A, B). Few examples are shown in the table of Fig. 13.30b. A typical FPGA structure is shown in Fig. 13.30c which is a two dimensional array of logic blocks interconnected by horizontal and vertical wires. The interconnection switches in interconnection blocks are either SRAM or antifuse type. Antifuses are modified CMOS based, normally open circuit but provides low resistance when programmed. Antifuses are not reprogrammable and nonvolatile. SRAM switches are reprogrammable but volatile.

FPGAs have gained rapid acceptance and growth because they can be applied to a very wide range of applications like device controllers, communication encoding and filtering, small to medium sized systems with SRAM blocks and many more. The other important applications of FPGAs are prototyping of designs (later to be implemented in custom made integrated circuits) and also for emulation of large hardware systems.

In Xilinx XC4000 SRAM based FPGA, each configurable logic block (also called CLB) can generate logic functions of up to nine inputs and has two flip-flops. Each of the interconnecting horizontal or vertical channel contains some short wire segments that span a single CLB, longer segments that span two CLBs, and very long segments that span the entire length or width of the chip. In this series, XC4003E has 100 CLBs while XC40250XV has as high as 8464 CLBs. Xilinx also offers XC2000, XC3000, XC5000 and XC8100(antifuse). The other manufacturers providing commercial FPGAs are Altera: FLEX8000 and FLEX10000, Actel: Act1, Act2 and Act3, Quicklogic: pASIC, pASIC2, etc. The SRAM based FPGAs, normally comes with EPROMs that stores bit-streams. This gets loaded every time power is switched on and programs the FPGA.

13.8 CONTENT ADDRESSABLE MEMORY

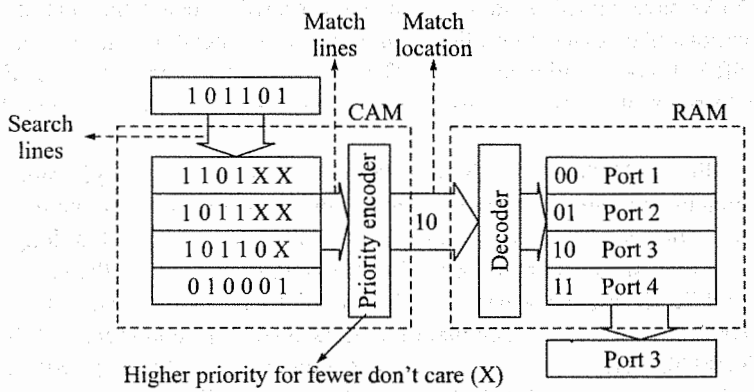
Content addressable memory (CAM) uses a completely different kind of addressing scheme from what has been discussed so far. It is designed to be faster to serve specific applications where speed is an issue. Take for example functioning of a network like the Internet. There, a message such as an e-mail or a web page is transferred by first breaking up the message into small data packets of a few hundred bytes, and then, sending each data packet individually through the network. These packets are routed from the source, through the intermediate nodes of the network (called routers), and reassembled at the destination to reproduce the original message. The function of a router is to compare the destination address of a packet to all possible routes and choose the appropriate one. A CAM is a good choice for implementing this lookup operation due to its fast search capability.

CAM compares input search data against a table of stored data, and returns the address of the matching data. Thus, it makes use of the content or data itself to find specific address by implementing a lookup table function using dedicated comparison circuitry. This is to reduce address decoding delays of conventional RAM by making the address meaningful and not an arbitrary one like RAM. A basic CAM cell serves two basic functions—bit storage, like RAM and in addition, bit comparison.

Let us try to understand how a CAM works from a packet forwarding example of a router. Figure 13.31a shows a simple routing table where output port assignment is shown for a range of destination addresses available from the relevant portion of input data stream. This table is a part of the CAM as shown in Fig. 13.31b. When an input data arrives with destination address 101101, the matching of the content occurs for 2nd and 3rd row, i.e. output port 2 and 3 both are eligible to transmit this data. A priority encoder following this lookup table decides which one is to be selected when a match occurs at more than one place. It follows a specific priority scheme. In this example, higher priority is given for the match that has lower number of don't care (X) states. The logic behind this is to keep a port free or available—to the extent possible—that can handle more number of destination addresses. Thus, match location 10 is the output of the priority encoder. The decoder to RAM takes this 10 as the address and selects port 3 as the selected output port.

Destination address	Output port
1101XX	1
1011XX	2
10110X	3
010001	4

(a)



(b)

Fig. 13.31 **(a) Routing table, (b) CAM implementing address lookup**

Extending the above example, generally speaking, an input word or *tag* from the incoming data is first stored in a *Search Data Register*. Its content, i.e. the tag, is then broadcasted as *search word* over *search lines*. In a typical CAM, there could be 36 to 144 bits in search lines while the table size could be as high as 32K entries (up to 15 bits of address space). Each stored word has a match line and whenever a match with a word occurs, the corresponding match line is activated. A priority encoder selects a match location based on some priority rule when there are more than one match line in activated state. This effectively reduces a larger space of input search word to a smaller space of output match location. The matching logic corresponding to CAM tags could be *binary* or *ternary*. The binary CAM requires exact match of all the binary locations and returns corresponding match lines while ternary CAM in addition, allows matching with don't care (X) bits. As we have seen in the network routing example, ternary CAM is more useful but also more complex to manage. Often, a CAM comes with a *hit* flag to indicate if there is no matching location in CAM.

SELF-TEST

15. What is the organization of the 7489, 64-bit RAM?
16. What is the organization of the 2114 SRAM?
17. What is a DRAM?
18. What is the organization of a 4116 DRAM?
19. How is combinatorial logic generated in FPGA?

SUMMARY

This entire chapter has been devoted to the study of memories. The use of magnetic and optical memory is discussed first. Next we considered the various rectangular arrays of memory cells on a chip and found that a square array containing the same number of rows and columns requires the fewest number of address lines.

Programmable, erasable-programmable, and plain read-only memories (PROMs, EPROMs, and ROMs) are used to store data in appiications where the data changes not at all, or only infrequently. These memory chips are available as either bipolar or MOS, but the MOS devices offer much greater capacity per chip.

Random-access memories (RAMs) are also available as either bipolar or MOS devices and are used to store data that must be readily available and may be changed frequently. The dynamic random-access memory (DRAM) offers the greater advantage of more storage capacity on a chip but has the disadvantage of requiring refreshing. Careful attention to timing requirements is absolutely essential with the use of any memory chip.

The basic memory cell on a bipolar chip is a simple latch using cross-coupled bipolar junction transistors. The same is true for a static MOS or CMOS memory chip, except that the transistors used are MOS or CMOS, respectively. A dynamic memory, on the other hand, uses a capacitor and one or more MOS transistors to store charge and, therefore, a single bit.

We have not undertaken an exhaustive study of all the memory chips available, but the chips discussed in detail are representative of the most popular ones in present use.

GLOSSARY

- ***access time*** In general, the delay time measured fron chip-enable (or address) until valid data appears at the output.
- ***address*** Selection of a cell in a memory array for a read or a write operation.
- ***cache*** Small, fast SRAM, a faster memory as an adjunct to slower main memory.
- ***CAM*** Content Addressable Memory.
- ***capacity*** The total number of bits that can be stored in a memory.
- ***chip*** A semiconductor circuit on a single silicon die.
- ***CD-ROM*** Compact Disk Read Only Memory, a kind of movable optical storage media that has higher capacity compared to magnetic counterpart.
- ***CD-R*** Compact Disk Recordable, a kind of optical memory on which data can be written but once.
- ***CD-RW*** Compact Disk Rewritable, a kind of optical memory on which data can be written and erased many times.
- ***DRAM*** Dynamic RAM.
- ***DVD*** Digital Versatile Disk or Digital Video Disk, a very high density optical memory.
- ***dynamic memory*** A memory whose contents must be restored periodically.
- ***EPROM*** An erasable-programmable read-only memory.
- ***EEPROM*** Electrically Erasable Programmable Read Only Memory.
- ***field-programmable*** Referring to a PROM that can be programmed by the user.
- ***flash memory*** A kind of nonvolatile memory which can be written and erased electrically.
- ***Floppy disk*** A movable low capacity magnetic storage media.
- ***Hard disk*** A high capacity magnetic storage media, integral part of modern computer.
- ***mask-programmable*** Referring to a PROM that can be programmed only by the manufacturer.
- ***matrix addressing*** Selection of a single cell in a rectangular array of cells by choosing the intersection of a single row and a single column.
- ***memory bank*** Connects more than one memory block to increase the capacity of the memory.
- ***memory cell*** The circuit used to store a single bit of information in a semiconductor memory chip.
- ***nonvolatile storage*** A method whereby a loss of power will not result in a loss of stored data.
- ***packing density*** Number of memory bits packed in per unit space.
- ***pass transistor*** A MOS transistor that passes information in either direction when it is turned on.
- ***PROM*** Programmable read-only memory.
- ***RAM*** Random-access memory.
- ***read operation*** The act of detecting the contents of a memory.
- ***refresh cycle*** Periodic refresh of DRAM.
- ***ROM*** read-only memory.
- ***static memory*** A memory capable of storing data indefinitely, provided there is no loss of power.
- ***SRAM*** Static RAM.
- ***volatile storage*** A method of storing information whereby a loss of power will result in a loss of the data stored.
- ***write operation*** The act of storing information in a memory.

PROBLEMS

Section 13.1

13.1 State the most appropriate memory type to use for each of the following:

a. The working memory in a small computer
b. The memory used to store permanent programs in a small computer
c. A memory used to store development programs in a small computer

13.2 Explain the difference between an EPROM and a PROM.

13.3 Explain the term volatile memory.

13.4 Explain why an EPROM is or is not a volatile memory.

13.5 A memory chip has a read and a write input. Is the chip ROM or RAM?

13.6 What is the difference between a memory cell and a memory word?

13.7 Why is a ROM considered nonvolatile memory?

13.8 What is the difference between an SRAM and a DRAM?

Section 13.2

13.9 What is the advantage of using a read-write head in magnetic recording systems?

13.10 Look at the code in Fig. 13.6b and determine the proper recording for each of the following:

a. The decimal number 4
b. The letter D
c. The decimal number 8
d. The decimal number 7

13.11 Why is recording on magnetic tape not considered random access?

13.12 A 2400-ft reel of 1/2-in magnetic tape has a data storage density of 6250, 7-bit characters per inch. Assuming no gaps and no lost space, what is the maximum storage capacity of the tape?

13.13 A 2400-ft reel of 1/2-in magnetic tape has a rewind speed of 300 in/s. How much time is required to rewind the tape from mid-position to its beginning? Neglect start and stop times.

Section 13.3

13.14 Why data integrity of optical memory is better than magnetic memory?

13.15 What is the data transfer rate of a 52X CD-ROM drive?

13.16 Briefly explain Read, Write, Erase process of CD-RW media.

13.17 What is the data transfer rate of 8X DVD-ROM drive?

13.18 Show the different possible rectangular arrangements for a memory that contains 32 memory cells. How many rows and columns for each case?

13.19 How many address lines are required for each case in Prob. 13.14?

13.20 What is the required address $A_4A_3A_2A_1$ to select cell 21 in Fig. 13.10c?

13.21 Determine how many address bits are required for a memory that has the following number of bits:

a. 1024 b. 4098
c. 256 d. 16,384

13.22 A memory chip available from Advanced Micro Devices is the Am9016, advertised as a 16K memory. How many bits of storage are there? How many address lines are required to access one bit at a time?

13.23 What address must be applied to the 74S89 in Fig. 13.20 to select the 4-bit word stored in row 14? Give the address in both binary and hexadecimal.

Section 13.4

13.24 What is the required address in both binary and hexadecimal to select the 8-bit word in row 27 of the TBP18S030 in Fig. 13.16?

13.25 Show a method for scanning the contents of a TBP18S030 beginning with word 1, then word 2, and so on up to word 32, and then repeating. (***Hint***: Try using a five-flip-flop binary counter for the address *ABCDE*, or use a mod-5 counter with decoding gates, or maybe a shift counter, or ...)

13.26 Write a Boolean expression for address row 15 in the TBP18S030 in Fig. 13.16.

13.27 Define the term mask-programmable.

13.28 Draw a set of timing waveforms for a TBP18S030 similar to Fig. 13.17, assuming an access time of 35 ns.

13.29 Redraw Fig. 13.30 and show exactly how to set the switches to program the 8-bit word at address 110 101. Explain exactly what must be done to program the word 1010 0011 at this address. Connecting switch P to an output will program a 1 at that cell.

13.30 In a manufacturing process, the pressure (P) in a pipe is related to the fluid in the pipe (F) according to the relation $P = 3F + 2$. Rather than compute values in real time, it is decided to store precomputed data in a PROM. In this case, F has only integer values between 0 and 4, so the computed values are found as shown in the accompanying table. Here's how the data is stored:

Each integer value of F (0, 1, 2, 3, and 4) represents an address in the PROM. The value of P is stored in binary form at the proper address. For instance, when $F = 2$, $P = 1000$ is stored at row address 2.

a. What is the value of P when $F = 4$?

b. Draw a PROM having 4-bit words, and show how all data are stored.

F	*P*	*P(binary)*
0	2	0010
1	5	0101
2	8	1000
3	11	1011
4	14	1110

13.31 Repeat part (b) of Prob. 13.30 if the relation is changed to $P = 2F + 1$.

13.32 Design a ROM to be used as a look-up table for the relation $L = S^2 - 2S + 3$, where $0 \le S \le 6$, and S has only integer values.

Section 13.5

13.33 Show how to connect 7489s in series to construct a memory that has thirty-two 4-bit words.

13.34 Show how to connect 7489s in parallel to construct a memory that contains sixteen 8-bit words.

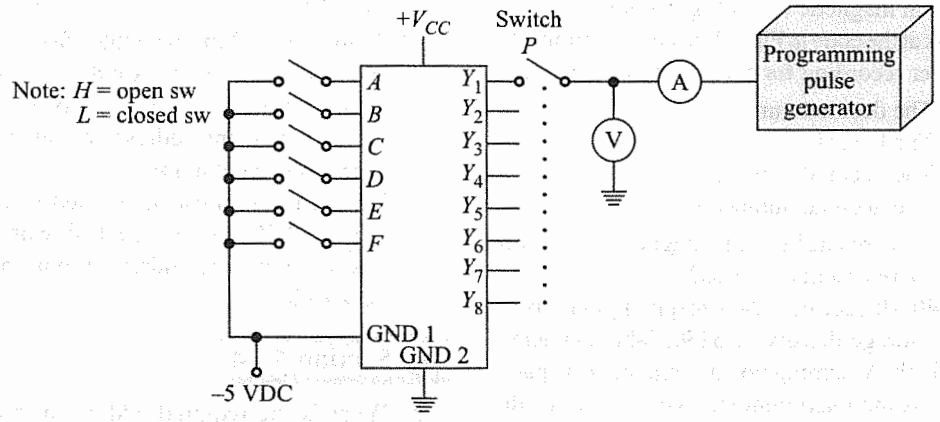


Fig. 13.32

13.35 Design the logic circuits, to provide a read and a write cycle for a 7489.

13.36 Refer to the 74S201 information in Fig. 13.26 and determine the following:

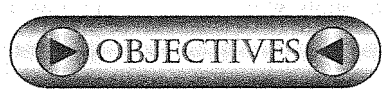
a. Minimum write-enable pulse width
b. Setup time, address to write-enable
c. Hold time, data from write-enable

13.37 Draw the logic diagram for a 256-word 8-bit memory using '201s.

Answers to Self-tests

1. A DRAM must be refreshed periodically.
2. EPROM stands for erasable-programmable read-only memory.
3. Cache memory is a small high-speed SRAM used inside a computer to speed up operation.
4. Even
5. Tape access time is too long!
6. Binary information is recorded on as magnetic film by magnetizing spots with two different orientations.
7. 780 nm.
8. CD-ROM 25% and more than 70%. CD-RW 15% and 25%.
9. 8.5 GB.
10. 145 (decimal) = 1001 0001 = $A_7A_6A_5A_4A_3A_2A_1A_0$
11. The cell at address 22—row 2 and column 2.
12. It refers to a ROM whose contents are established during the manufacturing process.
13. The triangle is the symbol for a three-state output.
14. It can be corrected by simply programming (adding) a 1 at word position Q_3. Note that you can add a 1 by programming (this is destroying a fuse link), but you cannot remove a programmed 1, since this would require replacing a fuse link.
15. Sixteen 4-bit words.
16. 1024, 4-bit words.
17. DRAM stands for dynamic random-access memory.
18. 16,384 × 1 bits
19. Through multiplexer-based look up table.

Digital Integrated Circuits



- **Explain how diodes and transistors can be used as electronic switches**
- **Demonstrate an understanding of TTL devices, their parameters, how to drive them, and how to use them to drive external loads**
- **Be familiar with CMOS-devices and characteristics**
- **Understand TTL-to-CMOS and CMOS-to-TTL interfacing**

In 1964 Texas Instruments introduced transistor-transistor logic (TTL), a widely used family of digital devices. TTL is fast, inexpensive, and easy to use. In this chapter we discuss several types of TTL: standard, high-speed, low-power, Schottky, and low-power Schottky. You will learn about open-collector and tri-state devices because these are used to build buses, the backbone of modern computers and digital systems. Since TTL uses active-low as well as active-high signals, negative logic may be used as well as positive logic. Complementary metal-oxide semiconductor (CMOS) devices are chips that combine *p*-channel and *n*-channel MOSFETs in a push-pull arrangement. Because the input current of a MOSFET is much smaller than that of a bipolar transistor, cascaded CMOS devices have very low power dissipation compared with TTL devices. This low dissipation explains why CMOS circuits are used in battery-powered equipment such as pocket calculators, digital wristwatches, and portable computers.

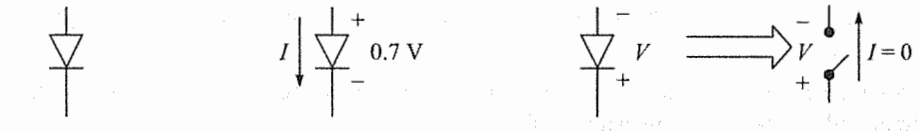
Since a knowledge of the subjects covered here is not prerequisite to any other chapter in this text, the material can be studied in part or in whole, at any time. An understanding of Ohm's law and familiarity with basic dc circuits are the only background needed.

14.1 SWITCHING CIRCUITS

The semiconductor devices used in digital integrated circuits (ICs) include diodes, *bipolar junction transistors (BJTs)* and *metal-oxide-semiconductor field-effect transistors (MOSFETs)*. The most popular transistor-transistor logic (TTL) in use includes the 7400 and the 74LS00 families; resistors, diodes, and BJTs are the elements used to construct these circuits. The 74C00 and the 74HC00 are the most widely used families constructed using MOSFETs. These two families of circuits are referred to as CMOS, since they use two different types of MOSFETs. In Chapter 1, we used the term *electronic switch* (see Fig. 1.7). Virtually all digital ICs in use today are silicon, so let's see how a silicon diode or transistor is used as an electronic switch.

The Semiconductor Diode

The symbol for a semiconductor diode (sometimes called a *pn* junction) is shown in Fig. 14.1a. The diode behaves like a *one-way* switch. That is, it will allow an electric current in one direction but not the other. We will use *conventional current flow* rather than *electron flow*. Figure 14.1b shows the direction of current through a diode—this is the *forward* direction. When conducting current, a silicon (Si) diode will have a nominal voltage of 0.7 V across its terminals as shown in Fig. 14.1b. In this condition, the diode is said to be *forward-biased*. Notice that the triangle in the diode symbol points in the direction of forward current—an easy memory crutch! It is not possible to pass current through the diode in the other direction—the *reverse* direction. When *reverse-biased*, the diode will act as an open switch as illustrated in Fig. 14.1c. To summarize:



(a) Symbol (b) Forward bias (c) Reverse bias

Fig. 14.1 **Semiconductor diode**

1. When forward-biased, the diode conducts current, and the voltage across the diode terminals is about 0.7 Vdc.
2. When reverse-biased, the diode will not conduct current. The voltage across the diode terminals depends on the external circuit.

Example 14.1 For each diode in Fig. 14.2, determine whether the diode is forward- or reverse-biased. Determine the diode current I in each case.

Solution

(a) The current direction is from +5 Vdc to ground, and thus the diode is forward-biased. The voltage across the diode terminals is 0.7 Vdc, and the diode current is found as

$$I = (5 - 0.7)/1\ \text{k}\Omega = 4.3/1\ \text{k}\Omega = 4.3\ \text{mA}$$

(b) The current direction is from +12 Vdc to ground, thus the diode is reverse-biased. The diode current is then $I = 0.0$ mA. There is no voltage across the 10-kΩ resistor, and thus the voltage across the diode terminals is 12 Vdc.

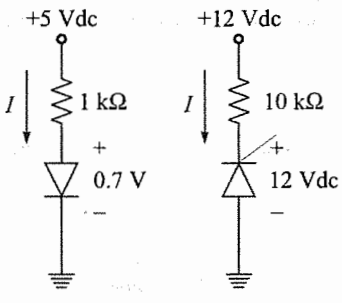
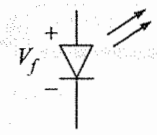


Fig. 14.2

LEDs



(a) Symbol

	Red	Yellow	Green
V_f(V)	1.6	2.2	2.4

(b) Typical forward voltages

Fig. 14.3

The symbol for a light-emitting diode (LED) is shown in Fig. 14.3a. The arrows indicate light emission capability. The operation of an LED is similar to that of an ordinary diode. When forward-biased, it emits light in the visible spectrum and is thus used as an indicator. However, the voltage across the diode terminals when forward-biased (V_f) is somewhat greater than 0.7 Vdc. Typical LED forward voltages given in Fig. 14.3b show that V_f varies with the color of the emitted light. The color of the emitted light depends on the elements added to the semiconductor material during manufacturing.

Example 14.2 The diode in Fig. 14.2a is replaced with a *red* LED. What is the diode current?

Solution The diode is forward-biased, and the voltage across its terminals is about 1.6 Vdc (Fig. 14.3b). The diode current is then

$$I = (5 - 1.6)/1\ \text{k}\Omega = 3.4/1\ \text{k}\Omega = 3.4\ \text{mA}$$

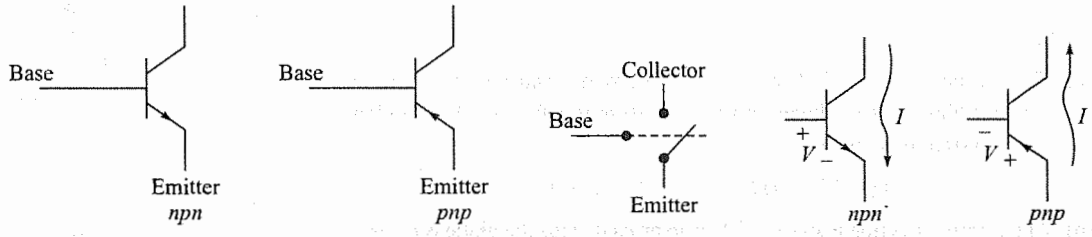
BJTs

The bipolar junction transistor (BJT) is available in two polarities (*npn* and *pnp*), as shown by the symbols in Fig. 14.4a. The BJT terminals are named *collector*, *emitter*, and *base*, as indicated. In Fig. 14.4b the BJT behaves as an electronic switch. The switch is activated by applying a voltage between base and emitter. Here's how it works:

1. The voltage between base and emitter is zero. The switch is open, and no current is allowed between collector and emitter. The transistor is off.
2. A voltage is applied between base and emitter. The switch is closed and a current is allowed between collector and emitter. The transistor is on. The voltage between emitter and collector (across a closed switch) is zero!

Since the BJT is available in two polarities—*npn* and *pnp*—the polarity of the applied base-emitter voltage must be as shown in Fig. 14.4c. For the *npn*, the base must be more positive than the emitter. The opposite is true for the *pnp*. This base-emitter voltage is applied across a forward-biased *pn* junction (a diode) and is thus limited to about 0.7 Vdc. Care must be taken not to exceed 0.7 Vdc, or the BJT may be destroyed.

The current through the *npn* transistor must be from collector to emitter as shown in Fig. 14.4c. For the *pnp*, current must be from emitter to collector. Notice that the current is in the direction of the arrow on the



(a) BJT symbols (b) Electronic switch (c)

Fig. 14.4

emitter—a good memory crutch! These polarities and current directions are important—*you should make every effort to commit them to memory!*

Example 14.3

a. Determine the current I and the voltage V_2 for the circuit in Fig. 14.5a if (i) $V_1 = 0$ Vdc, and (ii) $V_1 = +5$ Vdc.
b. Repeat part (a) for the circuit in Fig. 14.5b.

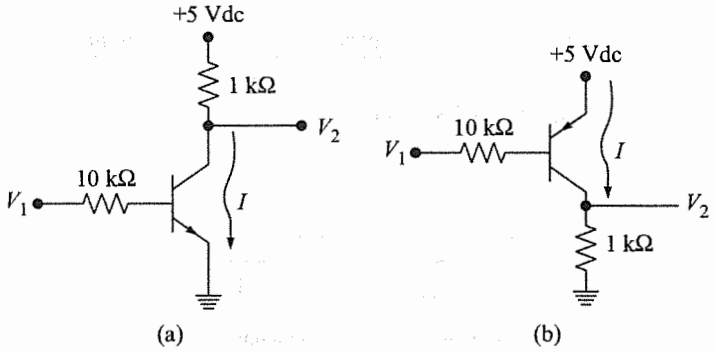


Fig. 14.5

Solution

a. $V_1 = 0$ Vdc. There is no current in the 10-kΩ resistor. Thus the voltage base-emitter is zero. The BJT is off (switch is open). The BJT current and the current in the 1-kΩ resistor is zero. The voltage V_2 is +5 Vdc.
$V_1 = +5$ Vdc. The base is more positive than the emitter—the BJT is on (switch is closed). V_2 is zero. The BJT current is $I = 5$ mA.
b. V_1 is 0 Vdc. The base is more negative than the emitter—the BJT is on (switch closed). V_2 is +5 Vdc. The BJT current is $I = 5$ mA.
V_1 is +5 Vdc. There is no current in the 10-kΩ resistor. The base is at +5 Vdc, and so is the emitter. Thus the voltage base-emitter is zero, and the BJT is off (switch open). The current $I = 0$ mA, and $V_2 = 0$ Vdc.

Let's look carefully at the results from Example 14.3. For both circuits, when $V_1 = 0$ Vdc, $V_2 = +5$ Vdc. Also, when $V_1 = +5$ Vdc, $V_2 = 0$ Vdc. Clearly V_2 is always the *inverse* of V_1—in other words, each circuit in Fig. 14.5 is an inverter! Either of these circuits can be used to implement the basic inverter introduced in Chapter 1 (Fig. 1.10).

MOSFETs

MOSFETs are available in two polarities (n-channel and p-channel) as shown by the symbols in Fig. 14.6a. MOSFETs operate as "depletion" or "enhancement" mode devices; the transistors in Fig. 14.6 are enhancement types. The MOSFET terminals are named *gate, source, drain, and body* as indicated. When the body is connected to the source, as is often the case with ICs, the simplified symbols are used. The MOSFET also behaves as the electronic switch in Fig. 14.6b. The switch is activated by applying a voltage between gate and source. Here's how it works:

1. The voltage between gate and source is zero. The switch is open, and no current is allowed between source and drain. The transistor is off.

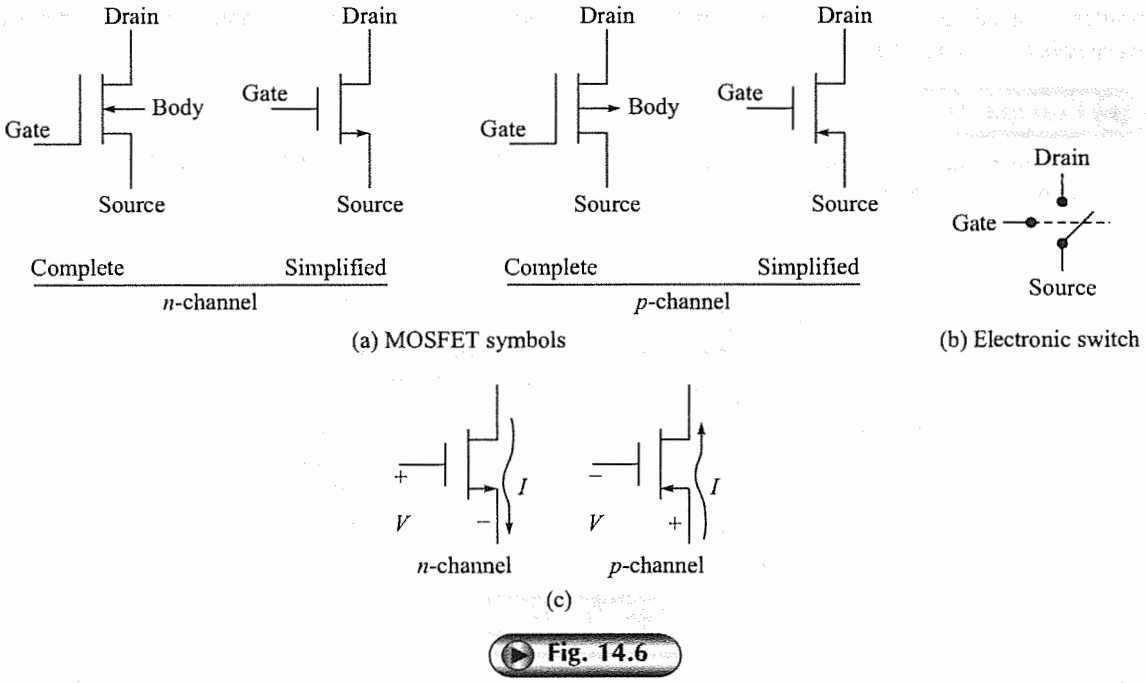


Fig. 14.6

2. A voltage is applied between gate and source. The switch is closed, and a current is allowed between source and drain. The transistor is on. The voltage between source and drain (across a closed switch) is zero!

Since the MOSFET is available in two polarities—*n*-channel and *p*-channel—the polarity of the applied gate-source voltage must be as shown in Fig. 14.6c. For the *n*-channel transistor, the gate must be more positive than the source. The opposite is true for the *p*-channel transistor. The current through the *n*-channel transistor must be from drain to source as shown in Fig. 14.6c. For the *p*-channel transistor, current must be from source to drain. Notice that the current is in the direction of the small arrow on the drain—a good memory crutch! These polarities and current directions are important—*you should make every effort to commit them to memory!*

Example 14.4 An *n*-channel MOSFET can be used to construct a simple inverter as shown in Fig. 14.7. Determine the current I and the output voltage V_2 if (a) $V_1 = 0$ Vdc, and (b) $V_1 = +5$ Vdc.

Solution

(a) $V_1 = 0$ Vdc. There is no current in the 100-kΩ resistor. Thus the gate-source voltage is zero. The MOSFET is off (the switch is open). The MOSFET current and the current in the 10-kΩ resistor are zero. The voltage V_2 is +5 Vdc.

(b) $V_1 = +5$ Vdc. The gate is more positive than the source—the MOSFET is on (the switch is closed). V_2 is zero. The MOSFET current is $I = 0.5$ mA. This circuit could also be used to implement the basic inverter introduced in Chapter 1 (Fig. 1.10).

The small size of a MOSFET on an IC is one of the great advantages of ICs constructed using MOSFETs. The 10-kΩ resistor in Fig. 14.7 requires a large area on an IC compared to a MOSFET. A second MOSFET

can be used in place of this resistor, as shown in Fig. 14.8. In this case, transistor Q_1 has its gate connected directly to its drain. When it is connected in this fashion, its behavior is similar to a resistor but shows little non-linearity compared to passive load, Q_1 is called an *active load*, and this circuit is a simple inverter.

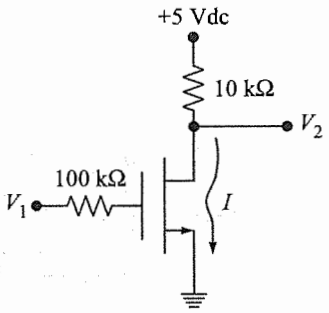


Fig. 14.7

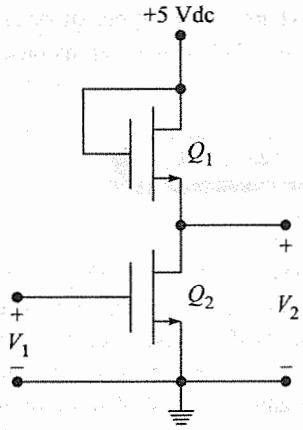


Fig. 14.8 An inverter with active load

Complementary Metal-Oxide-Semiconductor (CMOS) FETs

ICs constructed entirely with *n*-channel MOSFETs are called *NMOS* ICs. ICs constructed entirely with *p*-channel MOSFETs are called *PMOS* ICs. The 74C00 and 74HC00 families are constructed using *both* *n*-channel and *p*-channel MOSFETs. Since *n*-channel and *p*-channel MOSFETs are considered *complementary* devices, these ICs are referred to as *CMOS* ICs.

A CMOS inverter is shown in Fig. 14.9. Ideally, the characteristics of the *n* channel are closely matched with the *p* channel. This circuit is the basis for the 74C00 and 74HC00 families. Here's how it works:

1. $V_1 = 0$ Vdc. Q_n is off and Q_p is on. $V_2 = +5$ Vdc.
2. $V_1 = +5$ Vdc. Q_n is on and Q_p is off. $V_2 = 0$ Vdc.

Note that in the steady state (while not switching), one of the transistors is *always off*. As a result, the current I = 0 mA. When switching between states, both transistors are on for a very short time because of the rise or fall time of V_1. This is the only time the current I is nonzero. This is the reason CMOS is used in applications where dc power supply current must be held to a minimum—watches, pocket calculators, etc. *A word of warning:* If the input V_1 is held at +5 Vdc/2 = 2.5 Vdc, *both* transistors will be on. This is an almost direct short between +5 Vdc and ground, and it won't be long before both transistors expire! So don't impose this condition on a CMOS IC.

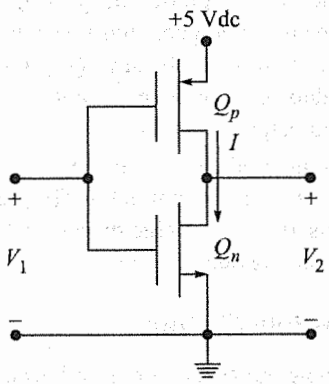


Fig. 14.9 A CMOS inverter

SELF-TEST

1. How does an LED differ from an ordinary silicon diode?
2. What are the two types of BJTs? What is the complement of an *n*-channel MOSFET?
3. An *npn* BJT is on when its base is more (positive, negative) than its emitter.
4. What is an active load in an NMOS IC?

14.2 7400 TTL

Standard TTL

Figure 14.10 shows a TTL NAND gate. The *multiple-emitter* input transistor is typical of the gates and other devices in the 7400 series. Each emitter acts like a diode; therefore, Q_1 and the 4-kΩ resistor act like a 2-input AND gate. The rest of the circuit inverts the signal so that the overall circuit acts like a 2-input NAND gate. The output transistors (Q_3 and Q_4) form a *totem-pole connection* (one *npn* in series with another); this kind of output stage is typical of most TTL devices. With a totem-pole output stage, either the upper or lower transistor is on. When Q_3 is on, the output is high; when Q_4 is on, the output is low.

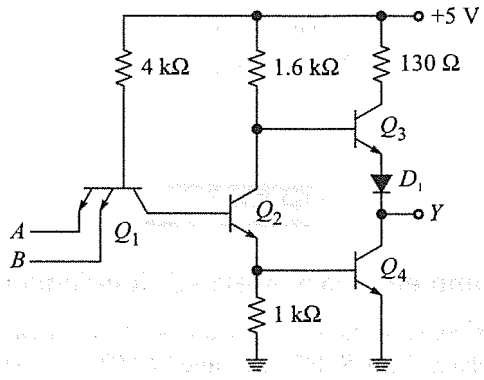


Fig. 14.10 Two-input TTL NAND gate

The input voltages A and B are either low (ideally grounded) or high (ideally +5 V). If A or B is low, the base of Q_1 is pulled down to approximately 0.7 V. This reduces the base voltage of Q_2 to almost zero. Therefore, Q_2 cuts off. With Q_2 open, Q_4 is off, and the Q_3 base is pulled high. The emitter of Q_3 is only 0.7 V below the base, and thus the Y output is pulled up to a high voltage.

On the other hand, when A and B are both high voltages, the emitter diodes of Q_1 stop conducting, and the collector diode goes into forward conduction. This forces Q_2 to turn on. In turn, Q_4 goes on and Q_3 turns off, producing a low output. Table 14.1 summarizes all input and output conditions.

Table 14.1 Two-Input NAND Gate

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Without diode D_1 in the circuit, Q_3 will conduct slightly when the output is low. To prevent this, the diode is inserted; its voltage drop keeps the base-emitter diode of Q_3 reverse-biased. In this way, only Q_4 conducts when the output is low.

Totem-Pole Output

Totem-pole transistors are used because they produce a *low output impedance*. Either Q_3 acts as an emitter follower (high output), or Q_4 is on (low output). When Q_3 is conducting, the output impedance is approximately 70 ohms (Ω); when Q_4 is on, the output impedance is only 12 Ω (this can be calculated from information on

the data sheet). Either way, the output impedance is low. This means the output voltage can change quickly from one state to the other because any stray output capacitance is rapidly charged or discharged through the low output impedance.

Propagation Delay Time and Power Dissipation

Two quantities needed for later discussion are power dissipation and propagation delay time. A standard TTL gate has a power dissipation of about 10 milliwatts (mW). It may vary from this value because of signal levels, tolerances, etc. but on the average it is 10 mW per gate. The *propagation delay time* is the time it takes for the output of a gate to change after the inputs have changed. The propagation delay time of a TTL gate is approximately 10 nanoseconds (ns).

Device Numbers

By varying the design of Fig. 14.10 manufacturers can alter the number of inputs and the logic function. With only few exceptions, the multiple-emitter inputs and the totem-pole outputs are used for different TTL devices. Table 14.2 lists some of the 7400 series TTL gates. For instance, the 7400 is a chip with four 2-input NAND gates in one package. Similarly, the 7402 has four 2-input NOR gates, the 7404 has six inverters, and so on.

Table 14.2 **Standard TTL**

Device Number	*Description*
7400	Quad 2-input NAND gates
7402	Quad 2-input NOR gates
7404	Hex inverter
7408	Quad 2-input AND gates
7410	Triple 3-input NAND gates
7411	Triple 3-input AND gates
7420	Dual 4-input NAND gates
7421	Dual 4-input AND gates
7425	Dual 4-input NOR gates
7427	Triple 3-input NOR gates
7430	8-input NAND gate
7486	Quad 2-input XOR gates

5400 Series

Any device in the 7400 series works over a temperature range of 0 to 70°C and over a supply range of 4.75 to 5.25 V. This is adequate for commercial applications. The 5400 series, developed for the military applications, has the same logic functions as the 7400 series, except that it works over a temperature range of −55 to 125°C and over a supply range of 4.5 to 5.5 V. Although 5400 series devices can replace 7400 series devices, they are rarely used commercially because of their much higher cost.

High-Speed TTL

The circuit of Fig. 14.10 is called *standard TTL*. By decreasing the resistances a manufacturer can lower the internal time constants; this decreases the propagation delay time. The smaller resistances, however, increase the power dissipation. This design variation is known as *high-speed TTL*. Devices of this type are numbered 74H00, 74H01, 74H02, and so on. A high-speed TTL gate has a power dissipating around 22 mW and a propagation delay time of approximately 6 ns.

Low-Power TTL

By increasing the internal resistances a manufacturer can reduce the power dissipation of TTL gates. Devices of this type are called *low-power TTL* and are numbered 74L00, 74L01, 74L02, etc. These devices are

slower than standard TTL because of the larger internal time constants. A low-power TTL gate has a power dissipation of 1 mW and a propagation delay time of about 35 ns.

Schottky TTL

With standard TTL, high-speed TTL, and low-power TTL, the transistors are switched on with excessive current, causing a surplus of carriers to be stored in the base. When you switch a transistor from on to off, you have to wait for the extra carriers to flow out of the base. The delay is known as *saturation delay time*.

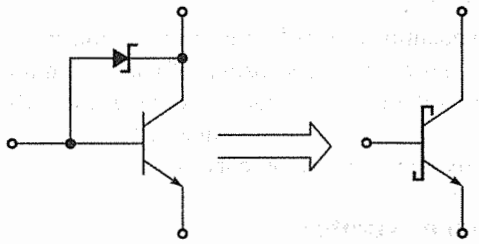


Fig. 14.11 **Schottky diode prevents transistor saturation**

One way to reduce saturation delay time is with *Schottky TTL*. The idea is to fabricate a Schottky diode along with each bipolar transistor of a TTL circuit, as shown in Fig. 14.11. Because the Schottky diode has a forward voltage of only 0.25 to 0.4 V, it prevents the transistor from saturating fully. This virtually eliminates saturation delay time, which means better switching speed. These devices are numbered 74S00, 74S01, 74S02, and so forth.

Schottky TTL devices are very fast, capable of operating reliably at 100 megahertz (MHz). The 74S00 has a power dissipation around 20 mW per gate and a propagation delay time of approximately 3 ns.

Low-Power Schottky TTL

By increasing internal resistances as well as using Schottky diodes, manufacturers have come up with a compromise between low power and high speed: *low-power Schottky TTL*. Devices of this type are numbered 74LS00, 74LS01, 74LS02, etc. A low-power Schottky gate has a power dissipation of around 2 mW and a propagation delay time of approximately 10 ns, as shown in Table 14.3.

The Winner

Low-power Schottky TTL is the best compromise between power dissipation and saturation delay time. In other words, of the five TTL types listed in Table 14.3, low-power Schottky TTL has emerged as the favorite of digital designers. It is used for almost everything. When you must have more output current, you can fall back on standard TTL. Or, if your application requires faster switching speed, then Schottky TTL is useful. Low-power and high-speed TTL are rarely used. if at all.

SELF-TEST

5. Draw the symbol for a Schottky transistor.
6. Which TTL family offers the lowest power and the fastest operation?

14.3 TTL PARAMETERS

7400 series devices are guaranteed to work reliably over a temperature range of 0 to 70°C and over a supply range of 4.75 to 5.25 V. In the discussion that follows, *worst case* means that the *parameters* (input current,

output voltage, and so on) are measured under the worst conditions of temperature and voltage. This means maximum temperature and minimum voltage for some parameters, minimum temperature and maximum voltage for others, or whatever combination produces the worst values.

Floating Inputs

When a TTL input is high (ideally +5 V), the emitter current is approximately zero (Fig. 14.12a). When a TTL input is *floating* (unconnected, as shown in Fig. 14.12b), no emitter current is possible because of the open circuit. Therefore, a floating TTL input is equivalent to a high output. Because of this, you sometimes see unused TTL inputs left unconnected; an open input allows the rest of the gate to function properly.

There is a disadvantage to floating inputs. When you leave an input open, it acts as a small antenna. Therefore, it will pick up stray electromagnetic noise voltages. In some environments, the noise pickup is large enough to cause erratic operation of logic circuits. For this reason, most designers prefer to connect unused TTL inputs to the supply voltage.

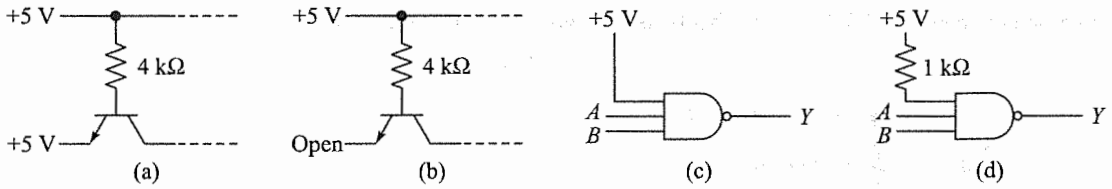


Fig. 14.12 **(a) High input, (b) Open is equivalent to high input, (c) Direct connection to supply voltage, (d) High input through a pull-up resistor**

For instance, Fig. 14.12c shows a 3-input NAND gate. The top input is unused, so it is connected to +5 V. A direct connection like this is all right with most Schottky devices (74S and 74LS) because their inputs can withstand supply overvoltages caused by switching transients. Since the top input is always high, it has no effect on the output. (*Note*: You don't ground the unused TTL input of Fig. 14.12c because then the output would remain stuck high, no matter what the values of A and B.)

Figure 14.12d shows an indirect connection to the supply through a resistor. This type of connection is used with standard, low-power, and high-speed TTL devices (74, 74L, and 74H). These older TTL devices have an absolute maximum input rating of +5.5 V. Beyond this level, the ICs may be damaged. The resistor is called a *pull-up resistor* because it serves to pull the input voltage up to a high. Most transients on the supply voltage are too short to charge the input capacitance through the pull-up resistor. Therefore, the input is protected against temporary overvoltages.

Worst-Case Input Voltages

Figure 14.13a shows a TTL inverter with an input voltage of V_i and an output voltage of V_o. When V_i is 0 V (grounded), it is in the low state and is designated V_{IL}. With TTL devices, we can increase V_{IL} to 0.8 V and still have a low-state input because the output remains in the high state. In other words, the low-state input voltage V_{IL} can have any value from 0 to 0.8 V. TTL data sheets list the worst-case low input as

$$V_{IL,\max} = 0.8 \text{ V}$$

If the input voltage is greater than this, the output state is unpredictable.

However, suppose V_i is 5 V in Fig. 14.13a. This is a high input and can be designated V_{IH}. This voltage can decrease all the way down to 2 V without changing the output state. In other words, the high-stage input V_{IH}

is from 2 to 5 V; any input voltage in this range produces a low output voltage. Data sheets list the worst-case high input as

$$V_{IH,\min} = 2 \text{ V}$$

When the input voltage is less than this, the output state is again unpredictable.

Figure 14.13b summarizes these ideas. As you see, any input voltage less than 0.8 V is a valid low-state input. Any input greater than 2 V is a valid high-state input. Any input between 0.8 and 2 V is indeterminate because there is no guarantee that it will produce the correct output voltage.

Worst-Case Output Voltages

Ideally, the low output state is 0 V, and the high output state is 5 V. We cannot attain these ideal values because of internal voltage drops inside TTL devices. For instance, when the output voltage is low in Fig. 14.13a, Q_4 is saturated and has a small voltage drop across it. With TTL devices, any output voltage from 0 to 0.4 V is considered a low output and is designated V_{OL}. This means the low-state output V_{OL} of a TTL device may have any value between 0 and 0.4 V. Data sheets list the worst-case low output as

$$V_{OL,\max} = 0.4 \text{ V}$$

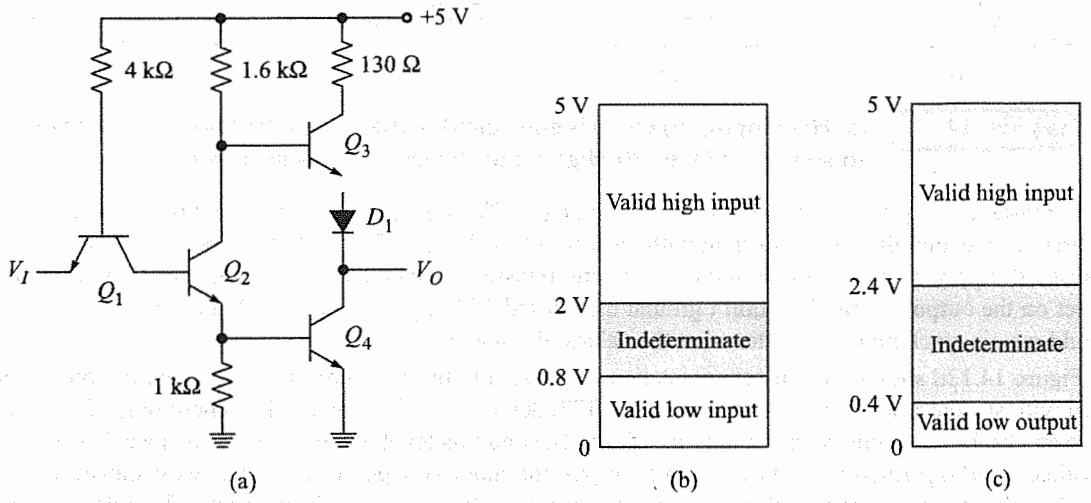


Fig. 14.13 **(a) TTL inverter, (b) TTL input profile, (c) TTL output profile**

When the output is high, Q_3 acts as an emitter follower. Because of the voltage drop across Q_3, D_1, and the 130-Ω resistor, the output voltage will be less than supply voltage. With TTL devices, the high-state output voltage is designated V_{OH}; it has a value between 2.4 and 3.9 V, depending on the supply voltage, temperature, and load. TTL data sheets list the worst-case high output as

$$V_{OH,\min} = 2.4 \text{ V}$$

Figure 14.13c summarizes the output states. As shown, any output voltage less than 0.4 V is a valid low-state output, any output voltage greater than 2.4 V is a valid high-state output, and any output between 0.4 and 2.4 V is indeterminate under worst-case conditions.

Profiles and Windows

The input characteristics of Fig. 14.13b are called the TTL input *profile*. Furthermore, each rectangular area in Fig. 14.13b can be thought of as a *window*. There is a low window (0 to 0.8 V), an indeterminate window (0.8 to 2.0 V), and a high window (2.0 to 5 V).

Similarly, Fig. 14.13c is the TTL output profile. Here you see a low window from 0 to 0.4 V, an indeterminate window from 0.4 to 2.4 V, and a high window from 2.4 to 5 V.

Values to Remember

We have discussed the low and high states for the input and output voltages. Here they are again as a reference for future discussions:

$$V_{IL,\max} = 0.8 \text{ V}$$
$$V_{IH,\min} = 2 \text{ V}$$
$$V_{OL,\max} = 0.4 \text{ V}$$
$$V_{OH,\min} = 2.4 \text{ V}$$

These are the worst-case values shown in Fig. 14.13b and c. On the input side, a voltage has to be less than 0.8 V to qualify as a low-state input, and it must be more than 2 V to be considered a high-state input. On the output side, the voltage has to be less than 0.4 V to be a low-state output and more than 2.4 V to be a high-state output.

Compatibility

TTL devices are *compatible* because the low and high output windows fit inside the low and high input windows. Therefore the output of any TTL device is suitable for driving the input of another TTL device. For instance, Fig. 14.14a shows one TTL device driving another. The first device is called a *driver* and the second a *load*.

Figure 14.14b shows the output stage of the TTL driver connected to the input stage of the TTL load. The driver output is shown in the low state. Since any input less than 0.8 V is a low-state input, the driver output (0 to 0.4 V) is compatible with the load input requirements.

Similarly, Fig. 14.14c shows high TTL output. The driver output (2.4 to 3.9 V) is compatible with the load input requirements (greater than 2 V).

Sourcing and Sinking

When a standard TTL output is low (Fig. 14.14b), an emitter current of approximately 1.6 milliamperes (mA) (worst case) exists in the direction shown. The conventional flow is from the emitter of Q_1 to the collector of Q_4. Because it is saturated, Q_4 acts as a *current sink*; conventional current flows through Q_4 to ground like water flowing down a sink.

However, when the standard TTL output is high (Fig. 14.14c), a reverse emitter current of 40 microamperes (μA) (worst-case) exists in the direction shown. Conventional current flows out of Q_3 to the emitter of Q_1. In this case, Q_3 is acting as a *source*.

Data sheets list the worst-case input currents:

$$I_{IL,\max} = -1.6\text{mA} \quad I_{IH,\max} = 40\ \mu\text{A}$$

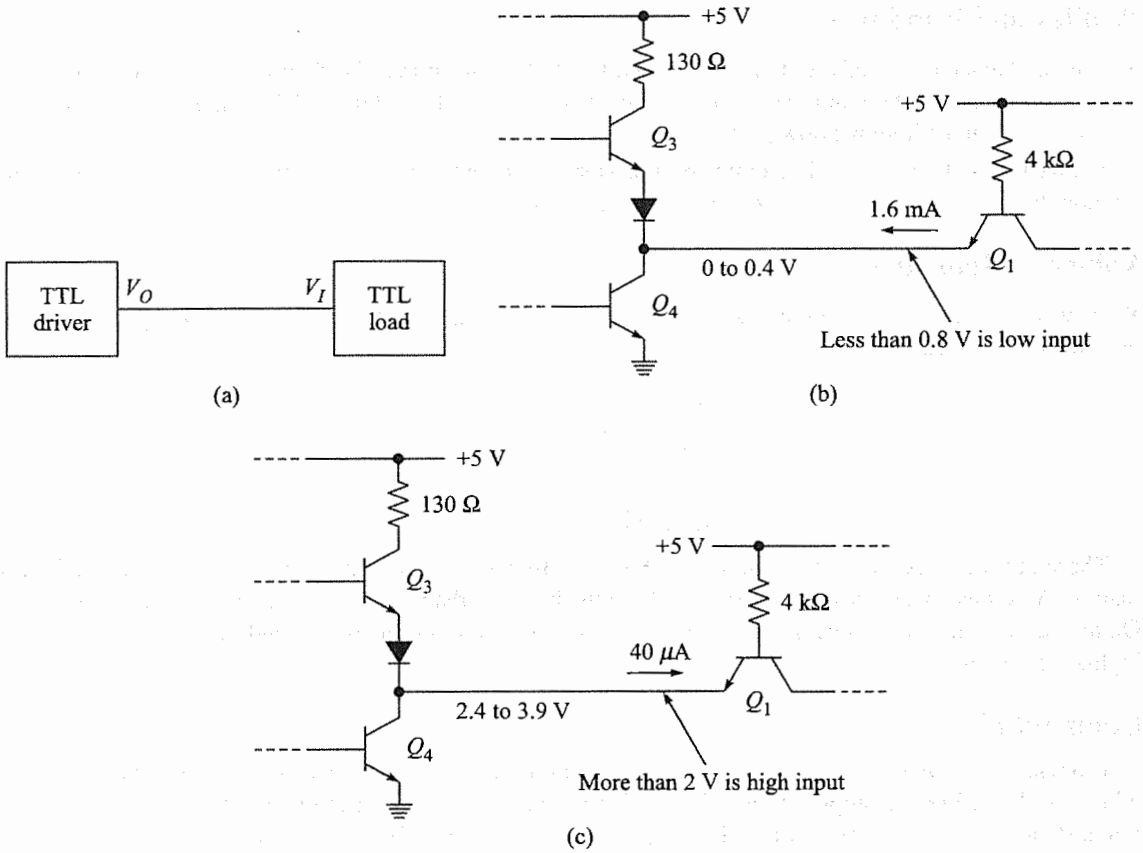


Fig. 14.14 **Sourcing and sinking current**

The minus sign indicates that the conventional current is out of the device; a plus sign means the conventional current is into the device. All data sheets use this notation, so do not be surprised when you see minus currents. The previous data tells us the maximum input current is 1.6 mA (outward) when an input is low and 40 μA (inward) when an input is high.

Noise Immunity

In the worst case, there is a difference of 0.4 V between the driver output voltages and required load input voltages. For instance, the worst-case low values are

$$V_{OL,\max} = 0.4 \text{ V} \quad \text{driver output}$$
$$V_{IL,\max} = 0.8 \text{ V} \quad \text{load input}$$

Similarly, the worst-case high values are

$$V_{OH,\min} = 2.4 \text{ V} \quad \text{driver output}$$
$$V_{IH,\min} = 2 \text{ V} \quad \text{load input}$$

In either case, the difference is 0.4 V. This difference is called *noise immunity*. It represents built-in protection against noise. Note that the concept of noise margin has been introduced in section 1.8 of Chapter 1.

Why do we need protection against noise? The connecting wire between a TTL driver and load is equivalent to a small antenna that picks up stray noise signals. In the worst case, the low input to the TTL load is

$$V_{IL} = V_{OL} + V_{noise} = 0.4\text{ V} + V_{noise}$$

and the high-stage input is

$$V_{IH} = V_{OH} - V_{noise} = 2.4\text{ V} - V_{noise}$$

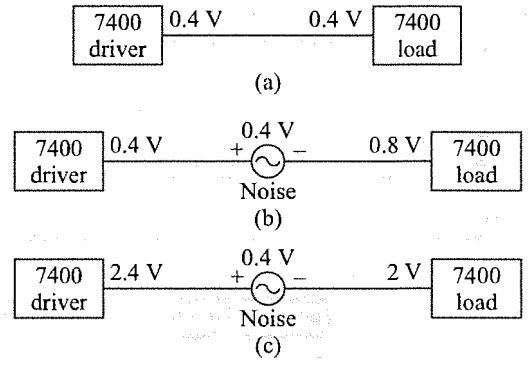


Fig. 14.15 **(a) TTL driver and load, (b) False triggering into high state, (c) False triggering into low state**

In most environments, the induced noise voltage is less than 0.4 V, and we get no false triggering of the TTL load.

For instance, Fig. 14.15a shows a low output from the TTL driver. If no noise voltage is induced on the connecting wire, the input voltage to the TTL load is 0.4 V, as shown. In a noisy environment, however, it is possible to have 0.4 V of induced noise on the connecting wire for either the low state (Fig. 14.15b) or the high state (Fig. 14.15c). Either way, the TTL load has an input that is on the verge of being unpredictable. The slightest additional noise voltage may produce a false change in the output state of the TTL load.

Standard Loading

A TTL device can source current (high output) or sink current (low output). Data sheets of standard TTL devices indicate that any 7400 series device can sink up to 16 mA, designated

$$I_{OL,max} = 16\text{ mA}$$

and can source up to 400 μA, designated

$$I_{OH,max} = -400\ \mu\text{A}$$

(Again, a minus sign means that the conventional current is out of the device, and a plus sign means that it is into the device.) As discussed earlier, the worst-case TTL input currents are

$$I_{IL,max} = -1.6\text{ mA} \quad I_{IH,max} = 40\ \mu\text{A}$$

Since the maximum output currents are 10 times larger than the input currents, we can connect up to 10 TTL emitters to any TTL output.

As an example, Fig. 14.16a shows a low output voltage (worst case). Notice that a single TTL driver is connected to 10 TTL loads (only the input emitters are shown). Here you see the TTL driver sinking 16 mA, the sum of the 10 TTL load currents. In the low state, the output voltage is guaranteed to be 0.4 V or less. If you try connecting more than 10 emitters, the output voltage may rise above 0.4 V under worst-case conditions. If this happens, the low-state operation is no longer reliable. Therefore, 10 TTL loads are the maximum that a manufacturer allows for guaranteed low-state operation.

Figure 14.16b shows a high output voltage (worst case) with the driver sourcing 400 μA for 10 TTL loads of 40 μA each. For this source current, the output voltage is guaranteed to be 2.4 V or greater under worst-case conditions. If you try to connect more than 10 TTL loads, you will exceed $I_{OH,max}$ and high-state operation becomes unreliable.

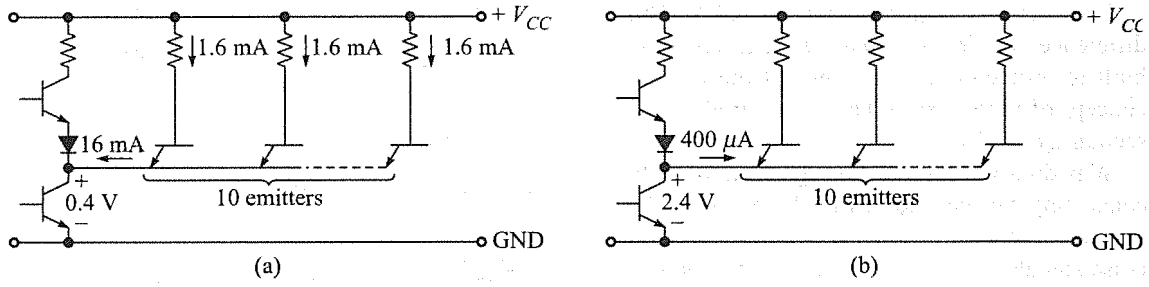


Fig. 14.16 **(a) Low-state fanout, (b) High-state fanout**

Loading Rules

Figure 14.17 shows the output-input profiles for different types of TTL. The output profiles are on the left, and the input profiles are on the right. These profiles are a concise summary of the voltages and currents for each TTL type. Start with the profiles of Fig. 14.17a; these are for standard TTL. On the left, you see the profile of output characteristics. The high output window is from 2.4 to 5 V with up to 400 μA of source current; the low output window is from 0 to 0.4 V with up to 16 mA of sink current. On the right, you see the input profile of a standard TTL device. The high window is from 2 to 5 V with an input current of 40 μA, while the low window is from 0 to 0.8 V with an input current of 1.6 mA.

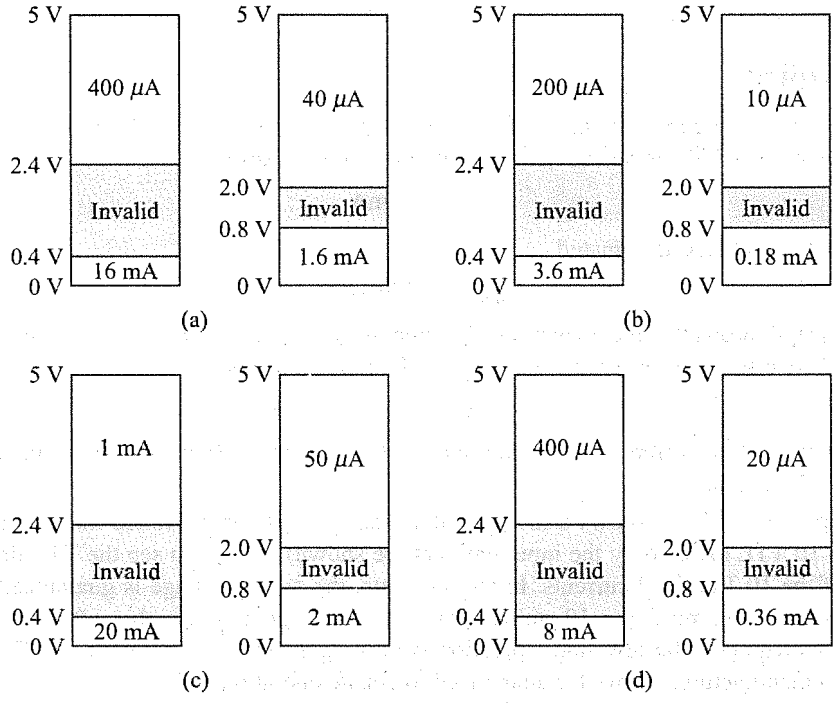


Fig. 14.17 **TTL output-input profiles: (a) Standard TTL, (b) Low-power TTL, (c) Schottky TTL, (d) Low-power Schottky TTL**

Standard TTL devices are compatible because the low and high output windows fit inside the corresponding input window. In other words, 2.4 V is always large enough to be a high input to a TTL load, and 0.4 V is always small enough to be a low input. Furthermore, you can see at a glance that the available source current is 10 times the required high-state input current, and the available sink current is 10 times the required low-state input current. The maximum number of TTL loads that can be reliably driven under worst-case conditions is called the *fanout*. With standard TTL, the fanout is 10 because one TTL driver can drive 10 TTL loads.

The remaining figures all have identical voltage windows. The output states are always 0 to 0.4 and 2.4 to 5 V, while the input states are 0 to 0.8 and 2 to 5 V. For this reason, all the TTL types are compatible; this means you can use one type of TTL as a driver and another type as a load.

The only differences in the TTL types are the currents. You can see in Fig. 14.17a to *d* that the input and output currents differ for each TTL type. For instance, a low-power Schottky TTL driver (see Fig. 14.17d) can source 400 μA and sink 8 mA; a low-power Schottky load requires input currents of 20 μA (high state) and 0.36 mA (low state). These numbers are different from standard TTL (Fig. 14.17a) with its output currents of 400 μA and 16 mA and its input currents of 40 μA and 1.6 mA.

Incidentally, notice that the profiles of high-speed TTL are omitted in Fig. 14.17 because Schottky TTL has replaced high-speed TTL, in virtually all applications. If you need high-speed TTL data, consult manufacturers' catalogs.

By analyzing Fig. 14.17*a* to *d* (plus the data sheets for high-speed TTL), we can calculate the fanout for all possible combinations. Table 14.4 summarizes these fanouts, which are useful if you ever have to mix TTL types.

Read Table 14.3 as follows. The TTL types have been abbreviated; 74 stands for 7400 series (standard), 74H for 74H00 series (high speed), and so forth. Drivers are on the left, and loads are on the right. Pick the driver, pick the load, and read the fanout at the intersection of the two. For instance, the fanout of a standard device (74) driving low-power Schottky devices (74LS) is 20. As another example, the fanout of a low-power device (74L) driving high-speed devices (74H) is only 1.

Table 14.3 Fanouts

TTL Driver	*TTL Load*				
	74	*74H*	*74L*	*74S*	*74LS*
74	10	8	40	8	20
74H	12	10	50	10	25
74L	2	1	20	1	10
74S	12	10	100	10	50
74LS	5	4	40	4	20

SELF-TEST

7. Why should TTL gate inputs never be left floating?
8. What is a TTL input profile?
9. What is the delay time for a 74LS04?

14.4 TTL OVERVIEW

Let's take a look at the logic functions available in the 7400 series. This overview will give you an idea of the variety of gates and circuits found in the TTL family. As a guide, Appendix 3 lists some of the 7400 series devices.

NAND Gates

The NAND gate is the backbone of the 7400 series. All devices in this series are derived from the 2-input NAND gate shown in Fig. 14.10. To produce 3-, 4-, and 8-input NAND gates, the manufacturer uses 3-, 4-, and 8-emitter transistors. Because they are so basic, NAND gates are the least expensive devices in the 7400 series.

NOR Gates

To get other logic functions the manufacturer modifies the basic NAND-gate design. For instance. Fig. 14.18 shows a 2-input NOR gate. Here Q_5 and Q_6 have been added to basic NAND-gate design. Since Q_2 and Q_6 are in parallel, we get the OR function, which is followed by inversion to get the NOR function.

When A and B are both low, the bases of Q_1 and Q_5 are pulled low; this cuts off Q_2 and Q_6. Then Q_3 acts as an emitter-follower, and we get a high output.

If A or B is high, Q_1 and Q_5 are cut off, forcing Q_2 or Q_6 to turn on. When this happens, Q_4 saturates and pulls the output down to a low voltage.

With more transistors, a manufacturer can produce 3- and 4-input NOR gates. (*Note*: A TTL 8-input NOR gate is not available.)

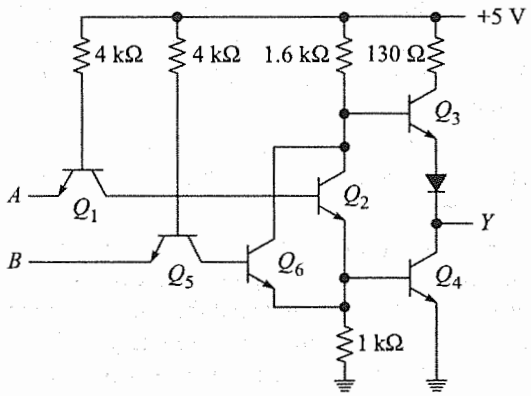


Fig. 14.18 TTL NOR gate

AND and OR Gates

To produce the AND function, another inverting stage is inserted in the basic NAND-gate design. The extra inversion converts the NAND gate to an AND gate. The available TTL AND gates are the 7408 (quad 2-input), 7411 (triple 3-input), and 7421 (dual 4-input).

Similarly, another inverting stage can be inserted in the NOR gate of Fig. 14.18; this converts the NOR gate to an OR gate. The only available TTL OR gate is the 7432 (quad 2-input).

Buffer Drivers

All IC buffer can source and sink more current than a standard TTL gate. As an example, the 7437 is a quad 2-input NAND buffer, meaning four 2-input NAND gates optimized to get high output currents. Each gate has the following worst-case currents:

$$I_{IL} = -1.6 \text{ mA} \qquad I_{IH} = 40\ \mu\text{A}$$
$$I_{OL} = 48 \text{ mA} \qquad I_{OH} = -1.2 \text{ mA}$$

The input currents are the same as those of a 7400 (standard TTL NAND gate), but the output currents are 3 times as high. This means that a 7437 can drive heavier loads. In other words, the fanout of a 7437 is 3 times that of a 7400. Appendix 3 includes several other buffer-drivers.

Hex schmitt-trigger (discussed in Chapter 7) inverter IC 7414 is shown in Fig. 14.19a.

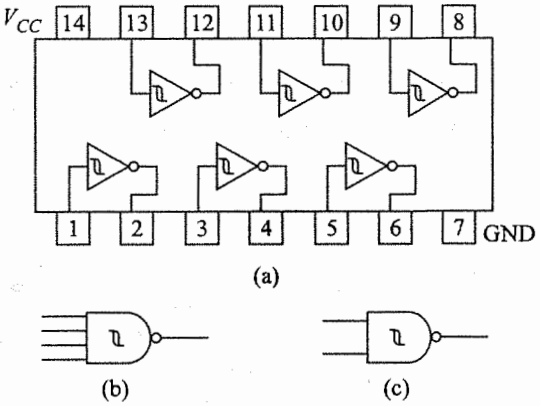


Fig. 14.19 **(a) Hex Schmitt-trigger inverters, (b) 4-input NAND Schmitt trigger, (c) 2-input NAND Schmitt trigger**

AND-or-INVERT Gates

Figure 14.20 shows the schematic diagram of an AND-OR-INVERT circuit. Here, Q_1, Q_2, Q_3 and Q_4 form the basic 2-input NAND gate of the 7400 series. By adding Q_5 and Q_6, we convert the basic NAND gate to an AND-OR-INVERT gate. Both Q_1 and Q_5 act as 2-input AND gates; Q_2 and Q_6 produce ORing and inversion. Because of this, the circuit is logically equivalent to Fig. 14.20b. This circuit represents 2-input, 2-wide AND-OR-INVERT gate. It is 2-input as each AND gate has 2 inputs and it is 2-wide because there are 2 AND gates at input stage. Figure 14.21a shows the schematic diagram of an expandable AND-OR-INVERT gate. As the name suggests, many such gates put together can expand the width at the input side. The difference between this and preceding AND-OR-INVERT gate (Fig. 14.20) is the collector and emitter pin brought outside the package.

Since Q_2 and Q_6 are the key to the ORing operation, we are being given access to the internal ORing function. By connecting other gates to these new inputs, we can expand the width of the AND-OR-INVERT gate.

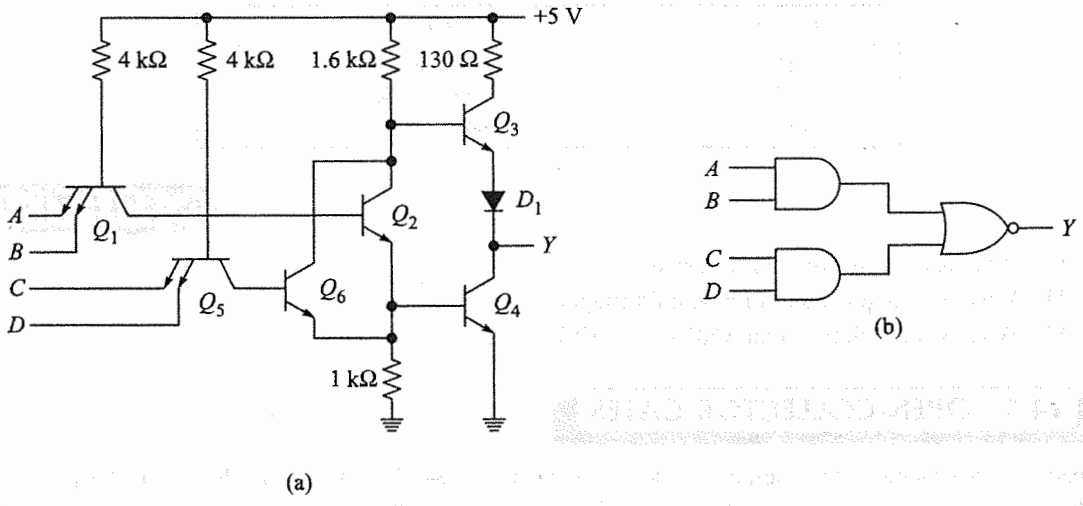


Fig. 14.20 **(a) AND-OR-INVERT schematic diagram, (b) Circuit**

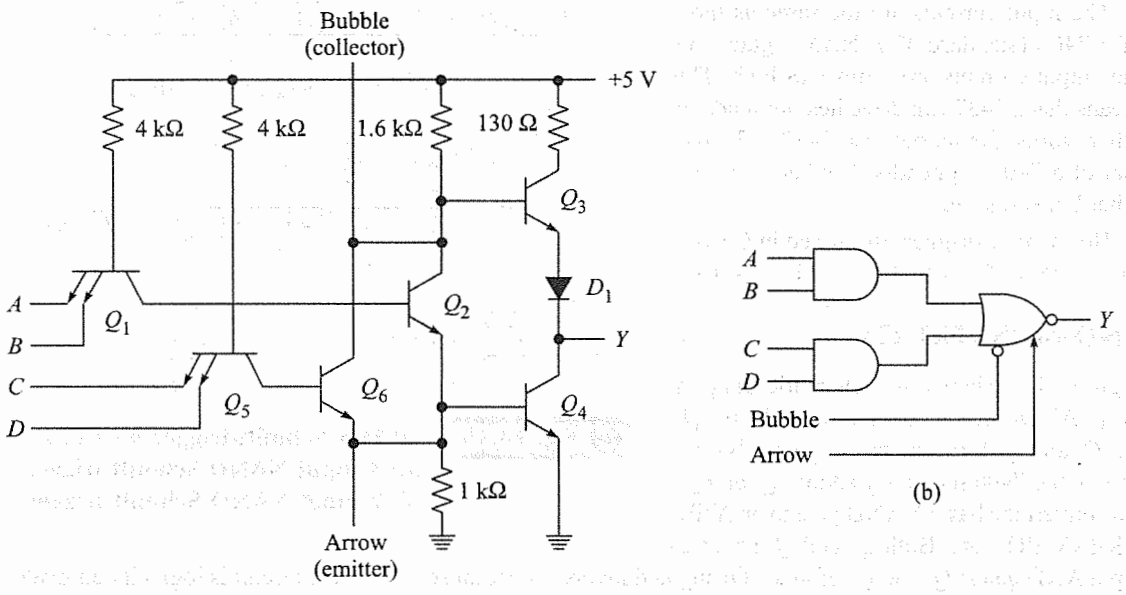


Fig. 14.21 **(a) Expandable AND-OR-INVERT gate, (b) Logic symbol**

Figure 14.21b shows the logic symbol for an expandable AND-OR-INVERT gate. The arrow input represents the emitter and the bubble stands for the collector. Table 14.4 lists the expandable AND-OR-INVERT gates in the 7400 series.

Table 14.4 **Expandable AND–OR–INVERT Gates**

Device	*Description*
7450	Dual 2-input 2-wide
7453	2-input 4-wide
7455	4-input 2-wide

SELF-TEST

10. What is the value of a buffer-driver?
11. What is an application for a Schmitt trigger?
12. What is the "width" of an AND-OR-INVERT gate?

14.5 OPEN-COLLECTOR GATES

Instead of a totem-pole output, some TTL devices have an *open-collector output*. This means they use only the lower transistor of a totem-pole pair. Figure 14.22a shows a 2-input NAND gate with an open-collector output. Because the collector of Q_4 is open, a gate like this will not work properly until you connect an external pull-up resistor, shown in Fig. 14.22b.

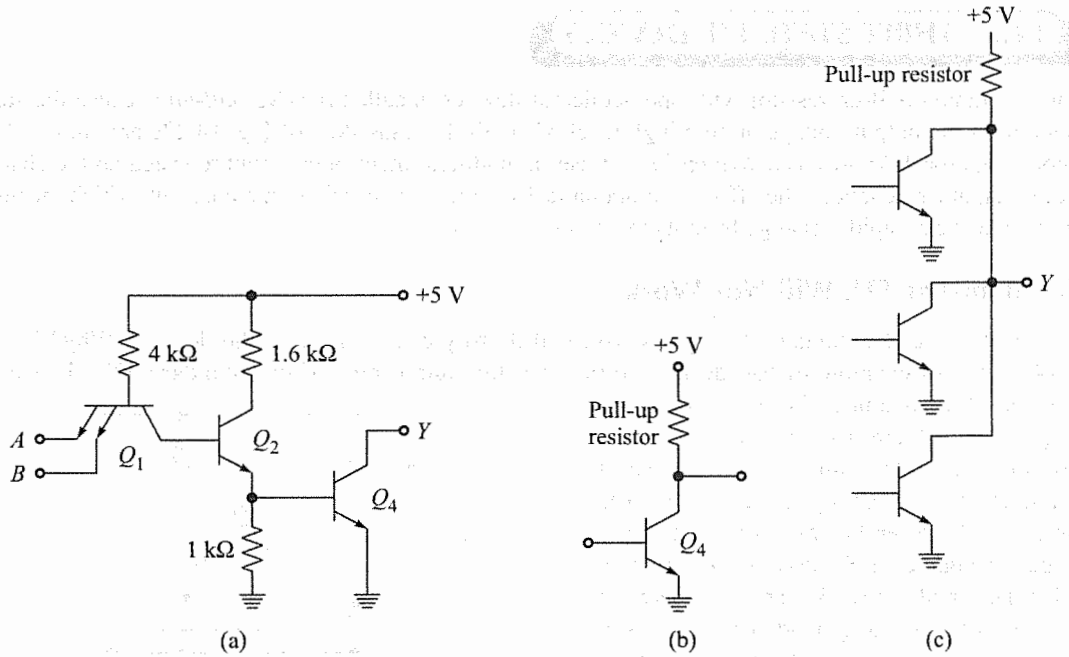


Fig. 14.22 **Open-collector TTL: (a) Circuit, (b) Pull-up resistor, (c) Open-collector outputs connected to a pull-up resistor**

The outputs of open-collector gates can be wired together and connected to a common pull-up resistor. For instance, Fig. 14.22c shows three TTL devices connected to the pull-up resistor. This is known as *wire-OR* (some called it wire-AND). A connection like this has the advantage of combining the output of three devices without using a final OR gate (or AND gate). The combining is done by a direct connection of the three outputs to the lower end of the common pull-up resistor. This is very useful when many devices are wire-ORed together. For instance, in some systems the outputs of 16 open-collector devices are connected to a pull-up resistor.

The big disadvantage of open-collector gates is their slow switching speed. Why is it slow? Because the pull-up resistance is a few kilohms, which results in a relatively long time constant when it is multiplied by the stray output capacitance. The slow switching speed of open-collector TTL devices is worst when the output goes from low to high. Imagine all three transistors going into cutoff in Fig. 14.22c. Then any capacitance across the output has to charge through the pull-up resistor. This charging produces a relatively slow exponential rise between the low and high state.

SELF-TEST

13. What must be connected to the output of an open-collector TTL gate?
14. Open-collector gates have (slower, faster) switching times.

14.6 THREE-STATE TTL DEVICES

Using a common pull-up resistor with open-collector devices is called *passive pull-up* because the supply voltage pulls the output voltage up to a high level when all the transistors of Fig. 14.22c are cut off. There is another approach known as *active pull-up*. It uses a modified totem-pole output to speed up the charging of stray output capacitance. The effect is to dramatically lower the charging time constant, which means the output voltage can rapidly change from its low to its high stage.

Why Standard TTL Will Not Work

If you try to wire-OR standard TTL gates, you will destroy one or more of the devices. Why? Look at Fig. 14.23 for an example of bad design. Notice that the output pins of two standard TTL devices are connected. If the output of the second device is low, Q_4 is on and appears approximately like a short circuit. If, at the same time, the output of the first device is in the high state, then Q_1 acts as an emitter follower that tries to pull the output voltage to a high level. Since Q_1 and Q_4 are both conducting heavily, only 130 Ω remains between the supply voltage and ground. The final result is an excessive current that destroys one of the TTL devices.

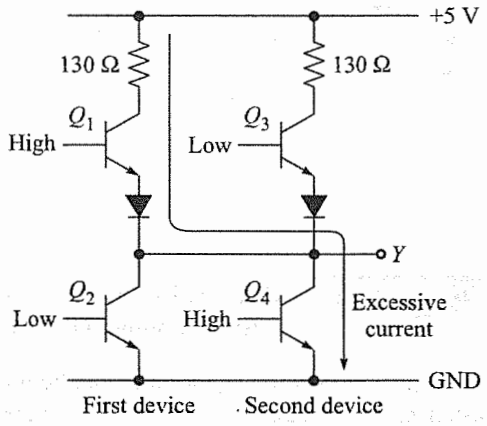


Fig. 14.23 **Direct connection of TTL outputs produces excessive current**

Low DISABLE Input

As you have seen, wire-ORing standard TTL devices will not work because of destructive currents in the output stages. This inability to wire-OR ordinary totem-pole devices is what led to *three-state (tri-state) TTL*, a new breed of totem-pole devices introduced in the early 1970s. With three-state gates, we can connect totem-pole outputs directly without destroying any devices. The reason for wanting to use totem-pole outputs is to avoid the loss of speed that occurs with open-collector devices.

Figure 14.24 shows a simplified drawing for a three-state inverter. When DISABLE is low, the base and collector of Q_6 are pulled low. This cuts off Q_7 and Q_8. Therefore, the second emitter of Q_1 and the cathode of D_1 are floating. For this condition, the rest of the circuit acts as an inverter: a low A input forces Q_2 and Q_5 to cut off, while Q_3 and Q_4 turn on, producing a high output. On the other hand, a high A input forces Q_2 to turn on, which drives Q_5 on and produces a low output. Table 14.5 summarizes the operation for low DISABLE.

Table 14.5 **Three-State Inverter**

Disable	*A*	*Y*
0	0	1
0	1	0
1	*X*	Hi-Z

High DISABLE Input

When DISABLE is high, the base and collector of Q_6 go high, which turns on Q_7 and Q_8. Ideally, the collector of Q_8 is pulled down to ground. This causes the base and collector of Q_1 to go low, cutting off Q_2 and Q_5.

Also Q_3 is off because of the clamping action of D_1. In other words, the base of Q_3 is only 0.7 V above ground, which is insufficient to turn on Q_3 and Q_4.

With both Q_4 and Q_5 off, the Y output is floating. Ideally, this means that the Thévenin impedance looking back into the Y output approaches infinity. Table 14.5 summarizes the action for this high-impedance state. As shown, when DISABLE is high, input A is a don't care because it has no effect on the Y output. Furthermore, because of the high output impedance, the output line appears to be disconnected from the rest of the gate. In effect, the output line is floating.

In conclusion, the output of Fig. 14.24 can be in one of three states: low, high, or floating.

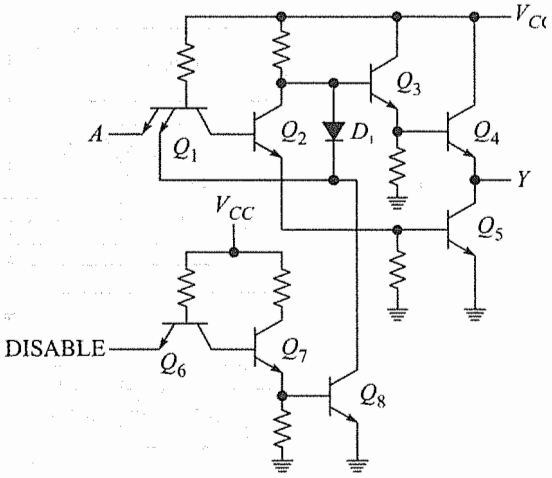


Fig. 14.24 Three-state inverter

Logic Symbol

Figure 14.25a is an equivalent circuit for the three-state inverter. When DISABLE is low, the switch is closed and the circuit acts as an ordinary inverter. When DISABLE is high, the switch is open and the Y output is floating or disconnected.

Figure 14.25b shows the logic symbol for a three-state inverter. When you see this symbol, remember that a low DISABLE results in normal inverter action, but a high DISABLE floats the Y output.

Three-State Buffer

By modifying the design, we can produce a three-state buffer, whose logic symbol is shown in Fig. 14.25c. When DISABLE is low, the circuit acts as a noninverting buffer, so that $Y = A$. But when DISABLE is high, the output floats. The three-state buffer is equivalent to an ordinary switch. When DISABLE is low, the switch is closed. When DISABLE is high, the switch is open.

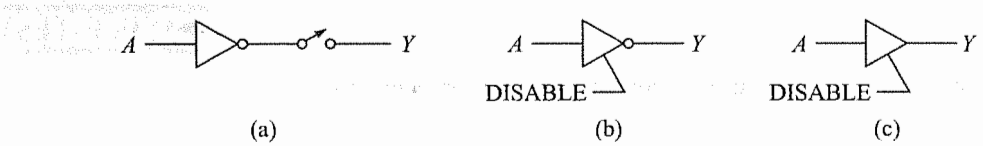


Fig. 14.25 Three-state logic diagrams: (a) Equivalent circuit of inverter, (b) Logic symbol of inverter, (c) Logic symbol of buffer

The 74365 is an example of a commercially available three-state hex noninverting buffer. This IC contains six buffers with three-state outputs. It is ideal for organizing digital components around a *bus*, a group of wires that transmits binary numbers between registers.

Bus Organization

Figure 14.26 shows some registers connected to a common bus. The three-state buffers control the flow of binary data between the registers. For instance, if we want the contents of register A to appear on the bus, all we have to do is make DISABLE low for register A but high for registers B and C. Then all the three-state

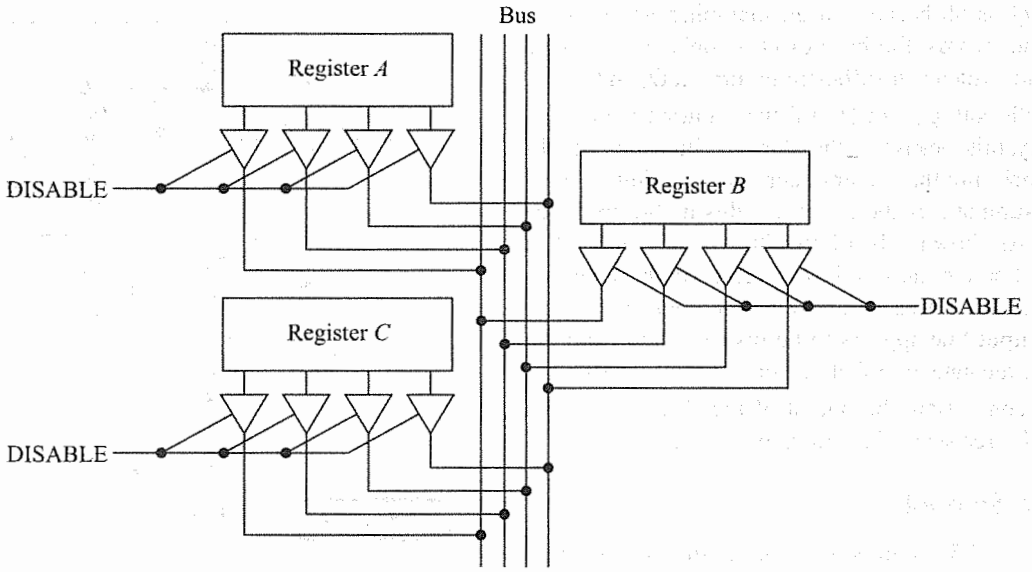


Fig. 14.26 **Three-state bus control**

switches on register A are closed, while all other three-state switches are open. As a result, only the contents of register A appear on the bus.

The idea in any bus-organized system is to make DISABLE high for all registers except the register whose contents are to appear on the bus. In this way, dozens of registers can time-share the same transmission path. Not only does this reduce the amount of wiring, but also it has simplified the architecture and design of computers and other digital systems. Refer to simple computer design discussed in Chapter 16.

SELF-TEST

15. Why are three-state gates used in conjunction with computer buses?

14.7 EXTERNAL DRIVE FOR TTL LOADS

To drive a TTL load with an external source, you need to satisfy the TTL input requirements for voltage and current. For standard TTL in the low state, this means an input voltage between 0 and 0.8 V with a current of approximately 1.6 mA. In the high state, the voltage has to be from 2 to 5 V with a current of approximately 40 μA. Let us take a brief look at some of the ways to drive a TTL load.

Switch Drive

Figure 14.27 shows the preferred method for driving a TTL input from a switch. With the switch open, the input is pulled up to +5 V. In the worst case, only 40 μA of input current exists. Therefore, the voltage

appearing at the input pin is slightly less than the supply voltage because of the small voltage drop across the pull-up resistor:

$$V_i = 5\text{ V} - (40\ \mu\text{A})(1\ \text{k}\Omega) = 4.96\text{ V}$$

This is well above the minimum requirement of 2 V, which is fine because it means that the noise immunity is excellent.

When the switch is closed, the input is pulled down to ground. In the worst case, the input current is 1.6 mA. This sink current creates no problem because it flows through the closed switch to ground. The noise immunity is fine because the input voltage is 0 V, well below the maximum allowable value of 0.8 V.

Size of Pull-Up Resistance

A pull-up resistance of 1 kΩ is nominal. You can use other values. Here are some of the factors to consider when you are selecting a pull-up resistor. In Fig. 14.27, the current drain with a closed switch is

$$I = \frac{5\text{ V}}{1\ \text{k}\Omega} = 5\text{ mA}$$

The smaller the pull-up resistance, the larger the current drain. At some point, too much current drain becomes a problem for the power supply, so you have to use a resistance that is large enough to keep the current drain to tolerable levels.

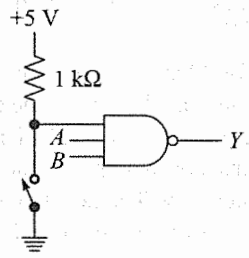


Fig. 14.27 Switch drive for TTL input

On the other hand, too large a pull-up resistance causes speed problems. The worst case occurs when the switch is opened. For instance, if the input capacitance is 10 picofarads (pF) in Fig. 14.27, the time constant is

$$RC = (1\ \text{k}\Omega)(10\text{ pF}) = 10\text{ ns}$$

The larger the pull-up resistance, the larger the time constant. A larger time constant means a slower switching speed because the input capacitance has to charge through the pull-up resistance.

Pull-up resistances between 1 and 10 kΩ are typical. They result in current drains and time constants that are acceptable in most applications.

Transistor Drive

Figure 14.28a shows another way to drive a TTL laid. This time, we are using a transistor switch to control the state of the TTL input. When V_i is low, the transistor is off and is equivalent to an open switch. Then the TTL input is pulled up to +5 V through a resistance of 1 kΩ. When V_i is high, the transistor is on and is equivalent to a closed switch. In this case, it easily sinks the 1.6 mA of input current.

The transistor inverts the control signal V_i. If this is objectionable, you can insert an inverter as shown in Fig. 14.28b. Now, the double inversion produces an in-phase control signal at the TTL input.

Operational Amplifier Drive

Sometimes, you want to use the output of an operational amplifier (OA) to control a TTL input. Because OAs typically use split-supply voltages of +15 and −15 V, you have to be careful how you connect to the TTL

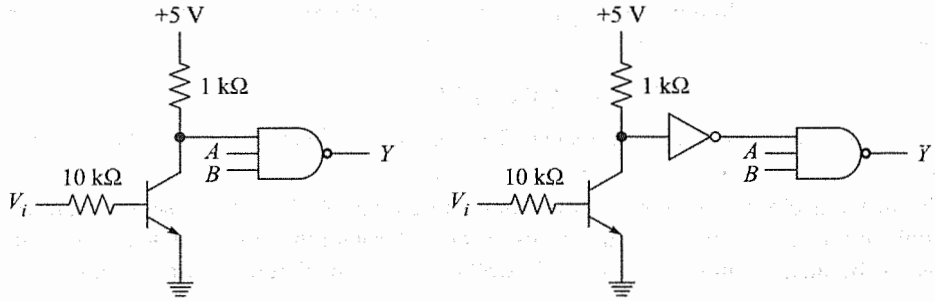


Fig. 14.28 **(a) Transistor drive for TTL input, (b) Inverter eliminates transistor inversion**

load. Figure 14.29 shows one way to use the output of a 741 to control a TTL input. The output of the OA ideally swings from +15 to −15V. The positive swing closes the transistor switch, producing a TTL input of approximately 0 V. The negative swing drives the transistor into cutoff, producing a TTL input of +5V.

Notice the diode in the base circuit. It protects the base against excessive reverse voltage. The data sheet of a 2N3904 indicates an absolute maximum base-emitter voltage rating of

$$V_{BE,\max} = -6 \text{ V}$$

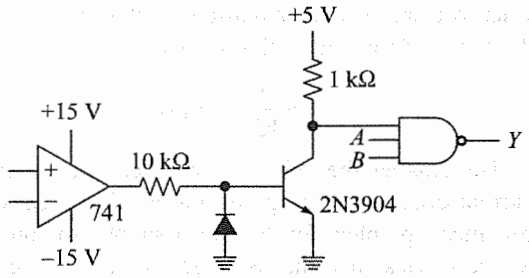


Fig. 14.29 **Op amp and transistor drive for TTL input**

Since the negative output of the OA approaches –15 V, we need to use a protective diode as shown between the base and ground. This diode clamps the base voltage at approximately –0.7 V on the negative swing.

Comparator Drive

Figure 14.30a shows the schematic diagram for a typical comparator, an IC that detects when the input voltage is positive or negative. Notice two things. First, a supply voltage of +15 V is typically used with this kind of device. Second, the comparator has an open-collector output transistor, Q_5. This sink transistor can be connected to any supply voltage.

Figure 14.30b shows how to connect an LM339 (typical comparator) to a TTL load. Because of the open-collector output, we can connect the output pin of the comparator to a supply voltage of +5 V through a pull-up resistance of 1 kΩ. When V_i is positive, the sink transistor goes off and the TTL input is pulled high. When V_i is negative, the sink transistor goes on and the TTL output is pulled low.

SELF-TEST

16. What factors influence the size of the resistor in Fig. 14.29?
17. What is the purpose of the diode in Fig. 14.31?

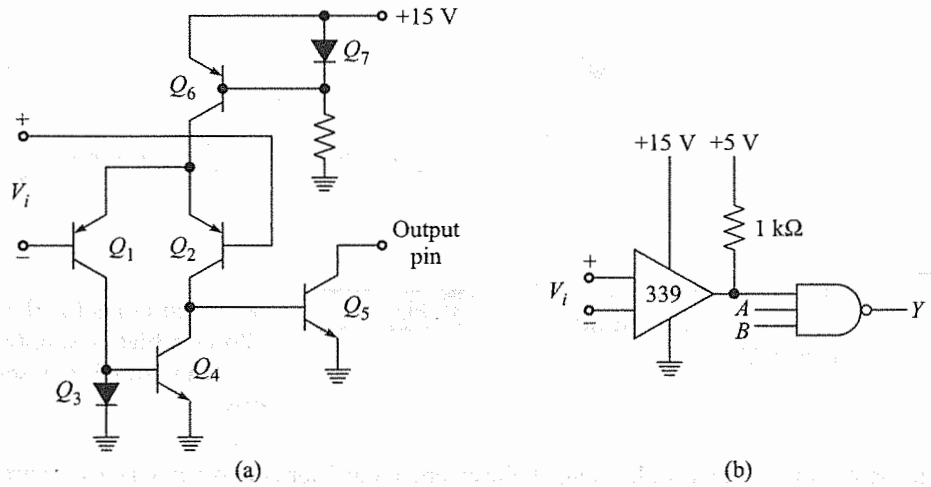


Fig. 14.30 **(a) Schematic diagram of comparator, (b) Interfacing an LM339 to a TTL input**

14.8 TTL DRIVING EXTERNAL LOADS

Because standard TTL can sink up to 16 mA, you can use a TTL driver to control an external load such as a relay, an LED, etc. Figure 14.31a illustrates the idea. When the TTL output is high, there is no load current. But when the TTL output is low, the lower end of R_L is ideally grounded. This sets up a load current of approximately

$$I_L = \frac{5\text{ V}}{R_L}$$

Since standard TTL can sink a maximum of 16 mA, the load resistance is limited to a minimum value of about

$$R_L = \frac{5\text{ V}}{16\text{ mA}} = 312\ \Omega$$

Driving an LED

Figure 14.31b is another example. Here a TTL circuit drives an LED. When the TTL output is high, the LED is dark. When the TTL output is low, the LED lights up. If the LED voltage drop is 2 V, the LED current for a low TTL output is approximately

$$I_L = \frac{5\text{ V} - 2\text{ V}}{270\ \Omega} = 11.1\text{ mA}$$

Supply Voltage Different from +5 V

If you need to use a supply voltage different from +5 V, you can use an open-collector TTL device. For instance, Fig. 14.32a on the next page shows an open-collector gate driving a load resistor that is returned to +15 V. Since an open-collector device can sink a maximum of 16 mA, the minimum load resistance in Fig. 14.32a is slightly less than 1 kΩ.

If you want more than 16 mA of load current, you can use an external transistor, as shown in Fig. 14.32b.

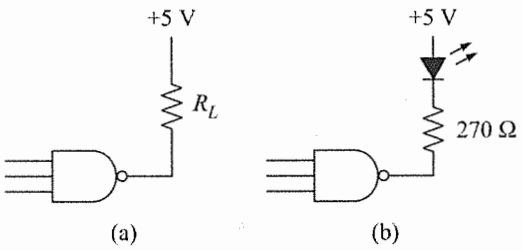


Fig. 14.31 **(a) TTL output drives load resistor, (b) TTL output drives LED**

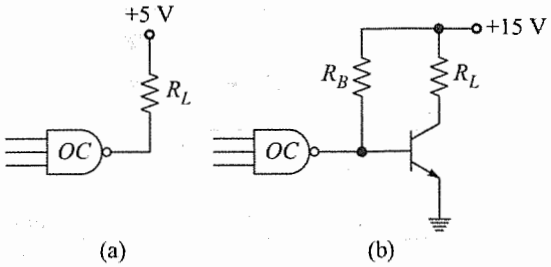


Fig. 14.32 **(a) Open-collector device allows a higher supply voltage, (b) Transistor increases current drive**

When the open-collector device has a low output, the external transistor goes off and the load current is zero. When the device has a high output, the external transistor goes on and the load current is maximum.

SELF-TEST

18. Could the resistor R_L in Fig. 14.34 be replaced with a red LED?

14.9 74C00 CMOS

National Semiconductor Corporation pioneered the 74C00 series, a line of CMOS circuits that are pin-for-pin and function-for-function compatible with TTL devices of similar numbers. For instance, the 74C00 is a quad 2-input NAND gate, the 74C02 is a quad 2-input NOR gate, and so on. This CMOS family contains a variety of small-scale integration (SSI) and medium-scale integration (MSI) chips that allow you to replace many TTL designs by the comparable CMOS designs. This is useful if you are trying to build battery-powered equipment. The 74HC00 series is the high-speed CMOS family.

NAND Gate

Figure 14.33 shows a CMOS NAND gate. The complementary design of input and output stages is typical of CMOS devices. Notice that Q_1 and Q_2 form one complementary connection; Q_3 and Q_4 form another. Visualize these transistors as switches. Then a low A input will close Q_1 and open Q_2; a high A input will open Q_1 and close Q_2. Similarly, a low B input will open Q_3, and close Q_4; a high B input will close Q_3 and open Q_4.

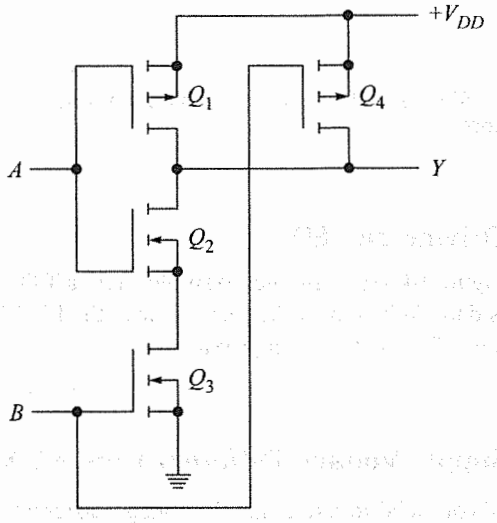


Fig. 14.33 **CMOS NAND gate**

In Fig. 14.33, the Y output is pulled up to the supply voltage when either Q_1 or Q_4 is conducting. The output is pulled down to ground only when Q_2 and Q_3 are conducting. If you keep this in mind, it simplifies the following discussion.

Case 1 Here A is low and B is low. Because A is low, Q_1 is closed. Therefore, Y is pulled high through the small resistance of Q_1.

Case 2 Now A is low and B is high. Since A is still low, Q_1 remains closed and Y stays in the high state.

Case 3 The A input is high and the B is low. Because B is low, Q_4 is closed. This pulls Y up to the supply voltage through the small resistance of Q_4.

Case 4 The A is high, and the B is high. When both inputs are high, Q_2 and Q_3 are closed, pulling the output down to ground.

Table 14.6 summarizes all input-output possibilities. As you can see, this is the truth table of a positive NAND gate. The output is low only when all inputs are high. To produce the positive AND function, we can connect the output of Fig. 14.33 to a CMOS inverter.

Table 14.6 CMOS NAND Gate

A	B	Y
Low	Low	High
Low	High	High
High	Low	High
High	High	Low

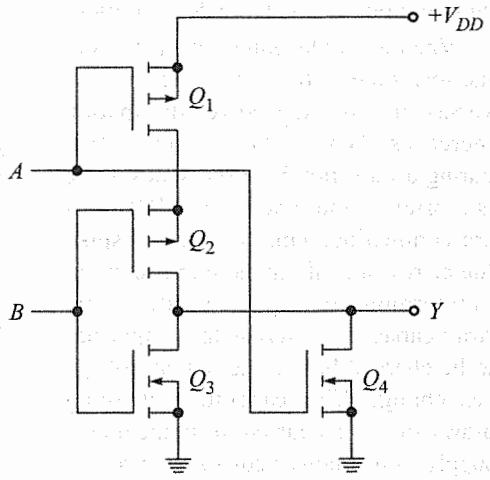


Fig. 14.34 CMOS NOR gate

NOR Gate

Figure 14.34 shows a CMOS NOR gate. The output goes high only when Q_1 and Q_2 are closed. The output goes low if either Q_3 or Q_4 is closed. There are four possible cases:

Case 1 The A is low, and the B is low. For both inputs low, Q_1 and Q_2 are closed. Therefore, Y is pulled high though the small series resistance of Q_1 and Q_2.

Case 2 The A is low, and the B is high. Because B is high, Q_3 is closed, pulling the output down to ground.

Case 3 The A is high, and the B is low. With A high, Q_4 is closed. The closed Q_4 pulls the output low.

Table 14.7

A	B	Y
Low	Low	High
Low	High	Low
High	Low	Low
High	High	Low

Case 4 The A is high, and the B is high. Since A is still high, Q_4 is still closed and the output remains low.

Table 14.7 summarizes these possibilities. As you can see, this is the truth table of a positive NOR gate. The output is low when any input is high.

Propagation Delay Time

A standard CMOS gate has a propagation delay time t_p of approximately 25 to 100 ns, with the exact value depending on the power supply voltage and other factors. As you recall, t_p is the time it takes for the output

of a gate to change after its inputs have changed. When two or more CMOS gates are cascaded, you have to add the propagation delay times to get the total. For instance, if you cascade three CMOS gates each with a t_p of 50 ns, then the total propagation delay time is 150 ns.

Power Dissipation

The *static power dissipation* of a device is its average power dissipation when the output is constant. The static power dissipation of a CMOS gate is in nanowatts. For instance, a 74C00 has a power dissipation of approximately 10 nanowatts (nW) per gate. This dissipation equals the product of supply voltage and leakage current, both of which are dc quantities.

When a CMOS output changes from the low state to the high state (or vice versa), the average power dissipation increases. Why? The reason is that during a transition between states, there is a brief period when both MOSFETs are conducting. This produces a spike (quick rise and fall) in the supply current. Furthermore, during a transition, any stray capacitance across the output has to be charged before the output voltage can change. This capacitive charging draws additional current from the power supply. Since power equals the product of supply voltage and device current, the instantaneous power dissipation increases, which means the average power dissipation is higher.

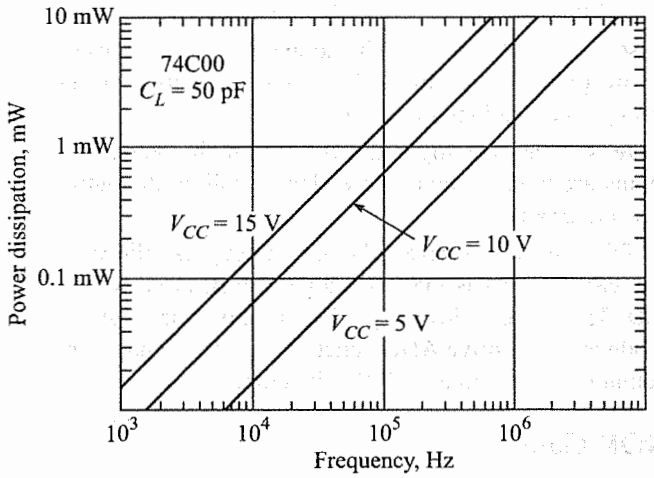


Fig. 14.35 **Active power dissipation of a 74C00**

The average power dissipation of a CMOS device whose output is continuously changing is called the *active power dissipation*. How large is the active power dissipation? This depends on the frequency at which the output is switching states. When the operating frequency increases, the current spikes occur more often and active power dissipation increases. Figure 14.35 shows the active power dissipation of a 74C00 versus frequency for a load capacitance of 50 pF. As you see, the power dissipation per gate increases with frequency and supply voltage. For frequencies in the megahertz region, the gate dissipation approaches or exceeds 10 mW (TTL gate dissipation). For CMOS to have an advantage over TTL, you operate CMOS devices at lower frequencies.

Another way to reduce power dissipation is to decrease the supply voltage. But this has adverse effects because it increases propagation time and decreases noise immunity. Although CMOS devices can work over a range of 3 to 15 V, the best compromise for speed, noise immunity, and overall performance is a supply voltage from 9 to 12 V. From now on, we assume a supply voltage of 10 V, unless otherwise specified.

Incidentally, notice the use of V_{CC} rather than V_{DD} for the supply voltage. This is a carryover from TTL circuits. You will find V_{CC} on the data sheets for 74C00 devices.

54C00 Series

Any device in the 74C00 series works over a temperature range of −40 to +85°C. This is adequate for most commercial applications. The 54C00 series (for military applications) works over a temperature range of −55 to +125°C. Although 54C00 devices can be substituted for 74C00 devices, they are rarely used commercially because of their much higher cost.

74HC00 Devices

The main disadvantage of CMOS devices is their relatively long propagation delay times. This places a limit on the maximum operating frequency of system. The 74HC00 series is a CMOS series of devices that are pin-for-pin and function-for-function compatible with TTL devices. These devices have the advantage of higher speed (less propagation delay time).

74HCT00 Devices

These are also high-speed CMOS circuits designed to be directly compatible with TTL devices. That is, they can be connected *directly* to any TTL circuit. Interfacing TTL and CMOS devices is discussed in Secs. 14.11 and 14.12.

CD4000 Series

RCA was the first to introduce CMOS devices. The original devices were numbered from CD4000 upward. This 4000 series was soon replaced by the 4000A series (called conventional) and the 4000B series (called the buffered type). The 4000A and B series are widely used; they have many functions not available in the 74C00 series. The main disadvantage of 4000 devices is their lack of pin-for-pin and function-for-function compatibility with TTL.

SELF-TEST

19. What kind of transistors are shown in Fig. 14.33?
20. Why is the NOR gate in Fig. 14.34a CMOS device?

14.10 CMOS CHARACTERISTICS

74C00 series devices are guaranteed to work reliably over a temperature range of −40 to +85°C and over a supply range of 3 to 15 V. In the discussion that follows, *worst case* means the parameters are measured under the worst conditions of temperature and voltage.

Floating Inputs

When a TTL input is floating, it is equivalent to a high input. You can use a floating TTL input to simulate a high input; but as already pointed out, it is better to connect unused TTL inputs to the supply voltage. This prevents the floating leads from picking up stray noise in the environment.

If you try to float a CMOS input, however, not only do you set up a possible noise problem, but, much worse, you produce excessive power dissipation. Because of the insulated gates, a floating input allows the gate voltage to drift into the linear region. When this happens, excessive current can flow through push-pull stages.

The absolute rule with CMOS devices, therefore, is to *connect all input pins*. Most of or all the inputs are normally connected to signal lines. If you happen to have an input that is unused, connect it to ground or the supply voltage, whichever prevents a stuck output state. For instance, with a positive NOR gate you should ground an unused input. Why? Because returning the unused NOR input to the supply voltage forces the output into a stuck low state. On the other hand, grounding an unused NOR input allows the other inputs to control the output.

With a positive NAND gate, you should connect an unused input to the supply voltage. If you try grounding an unused NAND input, you disable the gate because its output will stick in the high state. Therefore, the best thing to do with an unused NAND input is to tie it to the supply voltage. A direct connection is all right: CMOS inputs can withstand the full supply voltage.

Easily Damaged

Because of the thin layer of silicon dioxide between the gate and the substrate, CMOS devices have a very high input resistance, approximately infinite. The insulating layer is kept as thin as possible to give the gate more control over the drain current. Because this layer is so thin, it is easily destroyed by excessive gate voltage.

Aside from directly applying an excessive gate voltage, you can destroy the thin insulating layer in more subtle ways. If you remove or insert a CMOS device into circuit while the power is on, transient voltages caused by inductive kickback and other effects may exceed the gate voltage rating. Even picking up a CMOS IC may deposit enough charge to exceed its gate voltage rating.

One way to protect against overvoltages is to include zener diodes across the input. By setting the zener voltage below the breakdown voltage of the insulating layer, manufacturers can prevent the gate voltage from becoming destructively high. Most CMOS ICs include this form of zener protection.

Figure 14.36 shows a typical *transfer characteristic* (input-output graph) of a CMOS inverter. When the input voltage is in the low state, the output voltage is in the high state. As the input voltage increases, the output remains in the high state until a threshold is reached. Somewhere near an input voltage of $V_{CC}/2$, the output will switch to the low state. Then any input voltage greater than $V_{CC}/2$ holds the output in the low state.

This transfer characteristic is an improvement over TTL. Why? Because the indeterminate region is much smaller. As you can see, the input voltage has to be nearly equal to $V_{CC}/2$ before the CMOS output switches states. This implies that the noise immunity of CMOS devices ideally approaches $V_{CC}/2$. Typically, noise immunity is 45 percent of V_{CC}.

Also notice how much better defined the low and high output states are. When CMOS loads are used, the CMOS source and sink transistors have almost no voltage drop because there is almost no input

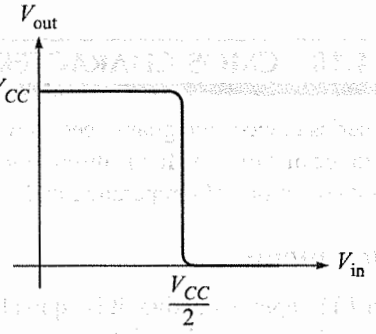


Fig. 14.36 **Typical transfer characteristic of a CMOS gate**

current to a CMOS load. Therefore, the static currents are extremely small. For this reason, the high output voltage is approximately equal to V_{CC}, and the low output voltage is approximately at ground. Stated another way, the logic swing between the low and high output states approximately equals the supply voltage, a considerable advantage for CMOS over TTL.

Compatibility

CMOS devices are compatible with one another because the output of any CMOS device can be used as the input to another CMOS device, as shown in Fig. 14.37a. For instance, Fig. 14.37b shows the output stage of a CMOS driver connected to the input stage of a CMOS load. The supply voltage is +10 V. Ideally, the input switching level is +5 V. Since the CMOS driver has a low output, the CMOS load has a high input.

Similarly, Fig. 14.37c shows a high CMOS driver output. This is more than enough voltage to drive the CMOS load with a high-state input. In fact, the noise immunity typically approaches 4.5 V (from 45 percent of V_{CC}). Any noise picked up on the connecting line between devices would need a peak value of more than 4.5 V to cause unwanted switching action.

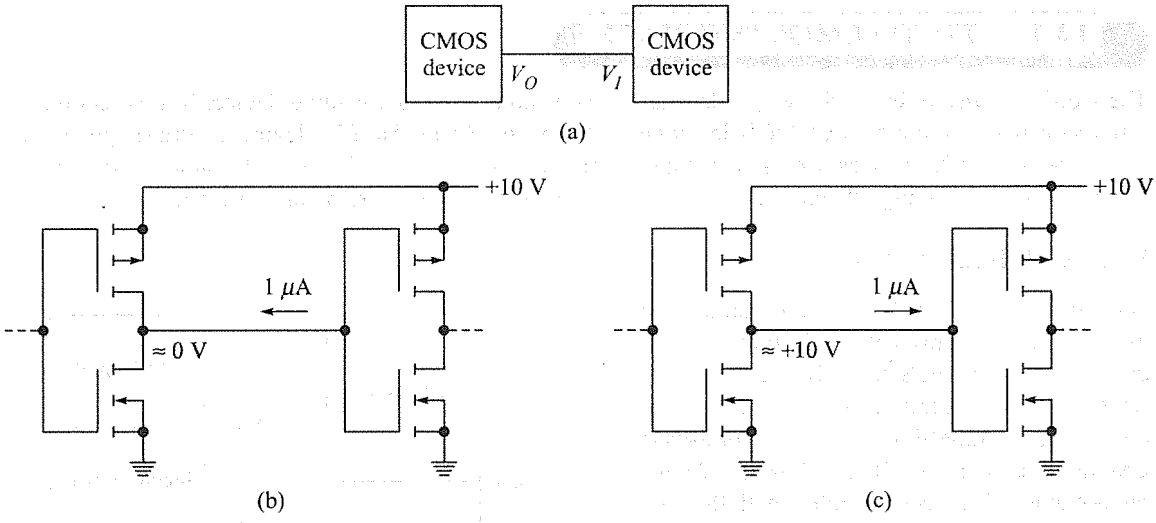


Fig. 14.37 **(a) Output of CMOS device can drive input of another CMOS device, (b) Sink current, (c) Source current**

Sourcing and Sinking

When a standard CMOS driver output is low (Fig. 14.37b), the input current to the CMOS load is only 1 microampere (μA) (worst case shown on data sheet). The input current is so low because of the insulated gates. This means that the CMOS driver has to sink only 1 μA. Similarly, when the driver output is high (Fig. 14.37c), the CMOS driver is sourcing 1 μ.A.

In symbols, here are the worst-case input currents for CMOS devices:

$$I_{IL,\max} = -1\ \mu A \quad I_{IH,\max} = 1\ \mu A$$

We use these values to calculate the fanout.

Fanout

The fanout of CMOS devices depends on the kind of load being driven. In Sec. 14.11, we discuss CMOS devices driving TTL devices. Now we want to concentrate on CMOS driving CMOS. Data sheets for 74C00 series devices give the following output currents for CMOS driving CMOS:

$$I_{OL,\max} = 10\ \mu A \quad I_{OH,\max} = -10\ \mu A$$

Since the worst-case input current of a CMOS device is only 1 μA, a CMOS device can drive up to 10 CMOS loads. Therefore, you can use a fanout of 10 for CMOS-to-CMOS connections. This value is reliable under all operating conditions.

SELF-TEST

21. Why must care be taken when using CMOS devices?
22. What is the transfer characteristic of a CMOS inverter?

14.11 TTL-TO-CMOS INTERFACE

The word *interface* refers to the way a driving device is connected to a loading device. In this section, we discuss methods for interfacing CMOS devices to TTL devices. Recall that TTL devices need a supply voltage of 5 V, while CMOS devices can use any supply voltage from 3 to 15 V. Because the supply requirements differ, several interfacing schemes may be used. Here are a few of the more popular methods.

Supply Voltage at 5 V

One approach to TTL/CMOS interfacing is to use +5 V as the supply voltage for both the TTL driver and the CMOS load. In this case, the worst-case TTL output voltages (Fig. 14.38a) are almost compatible with the worst-case CMOS input voltages (Fig. 14.38b). *Almost*, but not quite. There is no problem with the TTL low-state window (0 to 0.4 V) because it fits inside the CMOS low-state window (0 to 1.5 V). This means the CMOS load always interprets the TTL low-state drive as a low.

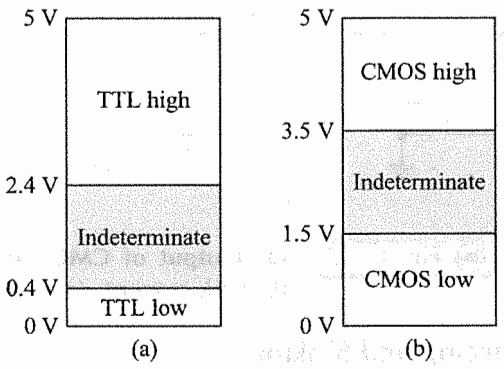


Fig. 14.38 (a) TTL output profile, (b) CMOS input profile

The problem is in the TTL high state, which can be as low as 2.4 V (see Fig. 14.38a). If you try using a TTL high-state output as the input to a CMOS device, you get indeterminate action. The CMOS device needs at least 3.5 V for a high-state input (Fig. 14.38b). Because of this, you cannot get reliable operation by connecting a TTL output directly to a CMOS input. You have to do something extra to make the two different devices compatible.

What do you do? The standard solution is to use a pull-up resistor between the TTL driver and the CMOS load, as shown in Fig. 14.39. What effect does the pull-up resistor have? It has almost no effect on the low

state, but it does raise the high state to approximately +5 V. For instance, when the TTL output is low, the lower end of the 3.3 kΩ is grounded (approximately). Therefore, the TTL driver sinks a current of roughly

$$I = \frac{5\text{ V}}{3.3\text{ k}\Omega} = 1.52\text{ mA}$$

When the TTL output is in the high state, however, the output voltage is pulled up to +5 V. Here is how it happens. As before, the upper totem-pole transistor actively pulls the output up to +2.4 V (worst case). Because of the pull-up resistor, however, the output rises above +2.4 V, which forces the upper totem-pole transistor into cutoff. The pull-up action is now passive because the supply voltage is pulling the output voltage up to +5 V through the pull-up resistor.

The gate capacitance of the CMOS load has to be charged through the pull-up resistor. This slows down the switching action. If speed is important, you can decrease the pull-up resistance. The minimum resistance is determined by the maximum sink current of the TTL device: $I_{OL,\max} = 16$ mA. In the worst case the supply voltage may be as high as 5.25 V, so the minimum resistance is

$$R_{\min} = \frac{5.25\text{ V}}{16\text{ mA}} = 328\ \Omega$$

The nearest standard value is 330 Ω, which you should consider the absolute minimum value for the pull-up resistor. And you would use this only if switching speed were critical. In many applications, a pull-up resistance of 3.3 kΩ is fine.

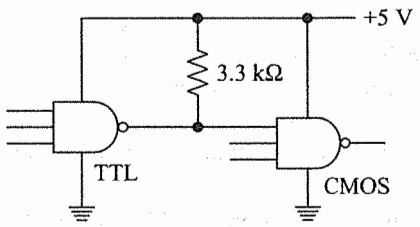


Fig. 14.39 **TTL driver and CMOS load**

Incidentally, the other inputs of the TTL driver and CMOS load (Fig. 14.39) are connected to signal lines not shown. Also, the use of 3-input gates is arbitrary. You can interface gates with any number of inputs. If more than one TTL chip is being interfaced to the CMOS load, connect each TTL driver to a separate pull-up resistor and CMOS input.

Different Supply Voltages

CMOS performance deteriorates at lower voltages because the propagation delay time increases and the noise immunity decreases. Therefore, it is better to run CMOS devices with a supply voltage between 9 and 12 V. One way to use a higher supply voltage is with an open-collector TTL driver (Fig. 14.40). Recall that the output stage of an open collector TTL device consists only of a sink transistor with a floating collector. In Fig. 14.40. this open collector is connected to a supply voltage of +12 V through a pull-up resistance of 6.8 kΩ. Likewise, the CMOS device now has a supply voltage of +12 V.

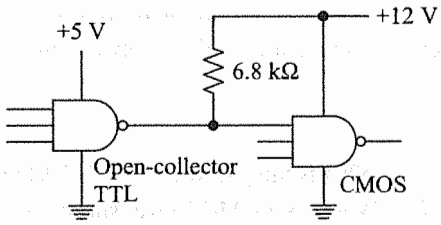


Fig. 14.40 **Open-collector TTL driver allows higher CMOS supply voltage**

When the TTL output is low, we can visualize a ground on the lower end of the pull-up resistor. Therefore, the TTL device has to sink approximately

$$I_{sink} = \frac{12\text{ V}}{6.8\text{ k}\Omega} = 1.76\text{ mA}$$

When the TTL output is high, the open-collector output rises passively to +12 V. In either case, the TTL outputs are compatible with the CMOS input states.

The passive pull-up in Fig. 14.40 produces slower switching action than before. For instance, with a gate input capacitance of 10 pF, the pull-up time constant is

$$RC = (6.8\text{ k}\Omega)(10\text{ pF}) = 68\text{ ns}$$

If this is a problem, reduce the pull-up resistance to its minimum allowable value of

$$R_{min} = \frac{12\text{ V}}{16\text{ mA}} = 750\ \Omega$$

Then the pull-up time constant decreases to

$$RC = (750\ \Omega)\ (10\text{ pF}) = 7.5\text{ ns}$$

CMOS Level Shifter

Figure 14.41 shows a 40109, called a *level shifter*. The input stage of the chip uses a supply voltage of +5 V, while the output stage uses +12 V. In other words, the input stage interfaces with TTL, and the output stage interfaces with CMOS.

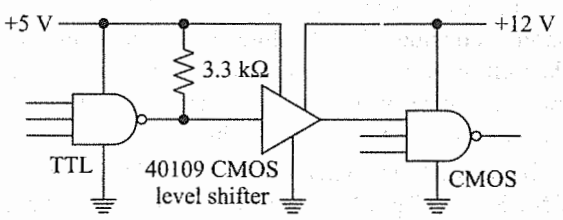


Fig. 14.41 **CMOS level shifter allows the use of 5-V and 12-V supplies**

In Fig. 14.41, a standard TTL device drives the level shifter. This produces active TTL pull-up to at least +2.4 V. Beyond this level, the pull-up resistor takes over and raises the voltage to +5 V, which ensures a valid high-state input to the level shifter. The output side of the level shifter connects to +12V (this can be changed to any voltage from 3 to 15 V). Since the CMOS load runs off of +12 V, it has better propagation delay time and noise immunity.

In summary, TTL has to run off of +5 V, but CMOS does better with a supply voltage of +12 V. This is the reason for using a level shifter between the TTL driver and the CMOS load.

SELF-TEST

23. What is the purpose of the 3.3-kΩ resistor in Fig. 14.41?
24. Where is a CMOS level shifter used?

14.12 CMOS-TO-TTL INTERFACE

In this section, we discuss methods for interfacing CMOS devices to TTL devices, Again, the problem is to shift voltage levels until the CMOS output states fall inside the TTL input windows. Specifically, we have to make sure that the CMOS low-state output is always less than 0.8 V, the maximum allowable TTL low-state input voltage. Also, the CMOS high-state output must always be greater than 2 V, the minimum allowable TTL high-state input voltage.

Supply Voltage at 5 V

One approach is to use +5 V as the supply voltage for the driver and the load, as shown in Fig. 14.42. A direct interface like this forces you to use a low-power Schottky TTL load (or two low-power TTL loads). Why? Because a low-power Schottky device has these worst-case input currents:

$$I_{IL,\max} = -360\ \mu A \quad I_{IH,\max} = 20\ \mu A$$

Data sheets for 74C00 devices list these worst-case output currents for CMOS driving TTL:

$$I_{OL,\max} = 360\ \mu A \quad I_{OH,\max} = -360\ \mu A$$

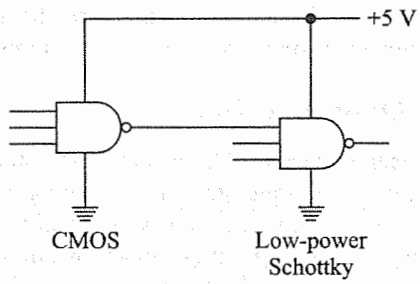


Fig. 14.42 CMOS driver and low-power Schottky TTL load

This tells us that a CMOS drive can sink 360 μA in the low state, exactly the input current for a low-power Schottky TTL devices. On the other hand, the CMOS driver can source 360 μA, which is more than enough to handle the high-state input current (only 20 μA). So the sink current limits the CMOS/74LS fanout to 1.

CMOS can also drive low-power TTL devices. The limiting factor again is the sink current. Low-power TTL has a worst-case low-state input current of 180 μA. Since a CMOS driver can sink 360 μA, it can drive two low-power TTL devices. Briefly stated, the CMOS/74L fanout is 2.

CMOS cannot drive standard TTL directly because the latter requires a low-state input current of −1.6 mA, for too much current for a CMOS device to sink without entering the TTL indeterminate region. The problem is that the sink transistor of a CMOS device is equivalent to a resistance of approximately 1.11 kΩ (worst case). The CMOS output voltage equals the product of 1.6 mA and 1.11 kΩ, which is 1.78 V. This is too large to be low-state TTL input.

Using a CMOS Buffer

Figure 14.43 shows how to get around the fanout limitation just discussed. The CMOS driver now connects directly to a CMOS buffer, a chip with larger output currents. For instance, a 74C902 is a hex buffer, or six CMOS buffers in a single package. Each buffer has these worst-case output currents:

$$I_{OL,\max} = 3.6\ \text{mA} \quad I_{OH,\max} = 800\ \mu A$$

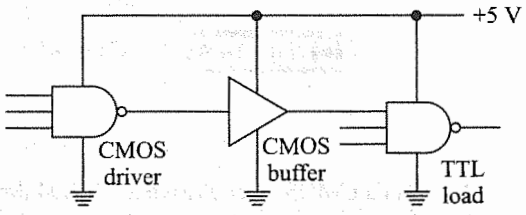


Fig. 14.43 CMOS buffer can drive standard TTL load

Since a standard TTL load has a low-state input current of 1.6 mA and a high-state input current of 40 μA, a 74C902 can drive two standard TTL loads. If you use one-sixth of a 74C902 in Fig. 14.43, the CMOS/TTL fanout is 2. Other available buffers are the CD4049A (inverting), CD4050A (noninverting), 74C901 (inverting), etc.

Different Supply Voltages

CMOS buffers like the 74C902 can use a supply voltage of 3 to 15V and an input voltage of −0.3 to 15 V. The input voltage can be greater than the supply voltage without damaging the device. For instance, you can use a high-state input of +12 V even though the supply voltage is only +5 V.

Figure 14.44 shows how to use the previous idea to our advantage. Here, the supply pin of the CMOS driver is connected to +12 V. On the other hand, the supply pin of the CMOS buffer is connected to +5 V to produce the TTL interface. Therefore, the input to the CMOS buffer will be as much as +12 V, even if its supply voltage is only +5 V. The fanout of this interface is still two standard TTL loads.

Open-Drain Interface

Recall open-collector TTL devices. The output stage consists of a sink transistor with a floating collector. Similar devices exist in the CMOS family. Known as *open-drain devices*, these have an output stage consisting only of a sink MOSFET. An example is the 74C906, a hex open-drain buffer.

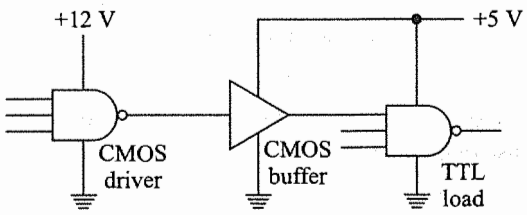


Fig. 14.44 **CMOS driver runs better with 12-V supply**

Figure 14.45 shows how an open-drain CMOS buffer can be used as an interface between a CMOS driver and a TTL load. The supply voltage for most of the buffer is +12 V. The open drain, however, is connected to a supply voltage of +5 V through a pull-up resistance of 3.3 kΩ. This has the advantage that both the CMOS driver and the CMOS buffer run off of +12 V, except for the open-drain output which provides the TTL interface.

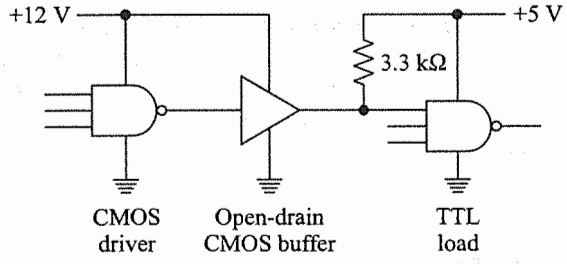


Fig. 14.45 **Open-drain CMOS buffer increases sink current**

SELF-TEST

25. Can a CMOS circuit drive a 74LS04 directly? What about a 7404?
26. CMOS output voltage levels are well within the profile of TTL input voltage levels. Why can't the CMOS drive the TTL directly?

14.13 CURRENT TRACERS

Figure 14.46a shows a solder bridge shorting a node to ground. When you trigger the logic pulser, the logic probe remains dark because the node is stuck in the low state. A logic pulser and logic probe will help you locate stuck nodes, but they cannot tell you the exact location of the short.

Figure 14.46b shows a *current tracer*, a troubleshooting tool than can detect current in a wire or circuit-board trace. Although it touches the wire, the current tracer does not make electric contract. Inside its

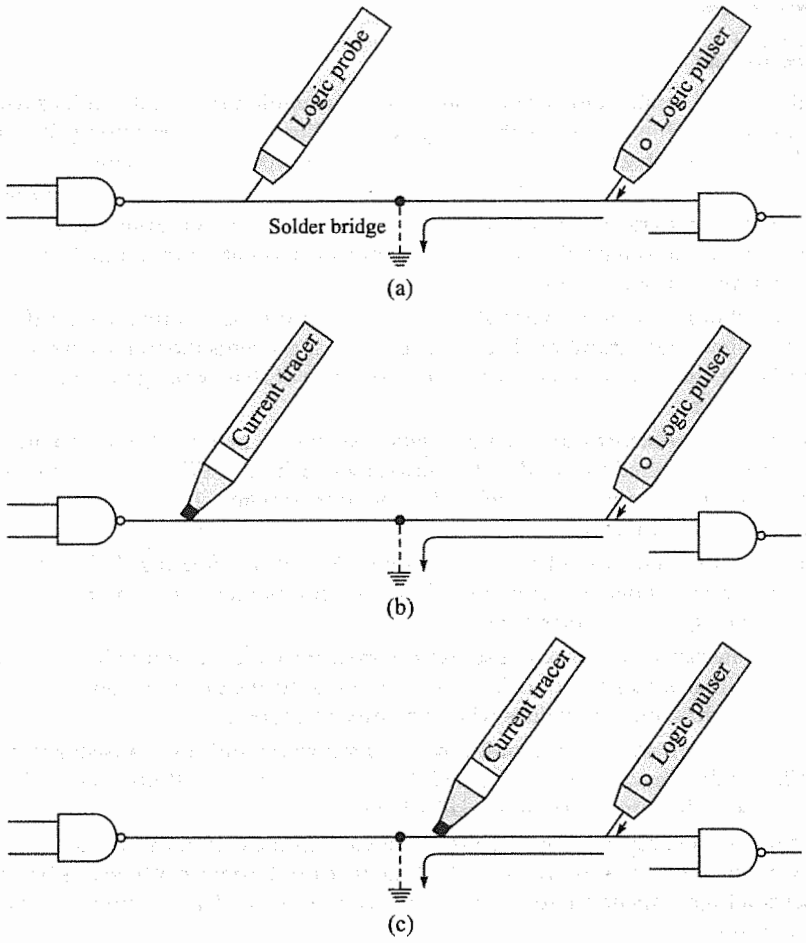


Fig. 14.46 **(a) Solder bridge shorts node to ground, (b) Current tracer will not detect any current, (c) Current tracer will detect current**

blunt insulated tip is a small pick-up coil than can detect the magnetic field around a wire carrying current. Therefore, if there is any current through the wire, the current tracer lights up.

In Fig. 14.46b, each time you trigger the logic pulser, a conventional current flows through its tip to ground along the path shown. The current tracer will not detect this current because it is touching another part of the wire.

If you move the current tracer between the ground and the logic pulser as shown in Fig. 14.46c, the current tracer will light up. The critical position for the current tracer is directly over the short. Move left, and the current tracer goes out. Move right, and it turns on. When this happens, you know the current tracer is directly over the trouble. During troubleshooting, a visual check at this critical location usually reveals the nature of the trouble (a solder bridge, in this discussion).

That's the basic idea behind a current tracer. You use the logic pulser and logic probe to find stuck nodes. Then you use the current tracer to locate the exact position along a trace where the short is.

SUMMARY

A chip is a small piece of semiconductor material with microminiature circuits on its surface. Small-scale integration (SSI) refers to chips with less than 12 gates. Medium-scale integration (MSI) means 12 to 100 gates per chip. Large-scale integration (LSI) refers to more than 100 gates on a chip.

The 7400 series is a line of standard TTL chips. This bipolar family contains a variety of compatible SSI and MSI devices. One way to recognize TTL design is the multiple-emitter input transistors and the totem-pole output transistors. The standard TTL chip has a power dissipation of about 10 mW per gate and a propagation delay time of around 10 ns.

By including a Schottky diode in parallel with the collector-base terminals, manufacturers produce Schottky TTL. This eliminates saturation delay time because it prevents the transistors from saturating on. Numbered from 74S00, these devices have a power dissipation of 20 mW per gate and a propagation delay time of approximately 3 ns.

By increasing internal resistances and including Schottky diodes, manufacturers can produce low-power Schottky TTL devices (numbered from 74LS00). A low-power Schottky TTL gate has a power dissipation of around 2 mW per gate and a propagation delay time of approximately 10 ns. Low-power Schottky TTL is the most widely used of the TTL types.

A floating TTL input is equivalent to a high input. Do not use floating TTL inputs when you are operating in an electrically noisy environment. Floating inputs may pick up enough noise voltage to produce unwanted changes in the output states.

A standard TTL gate can sink 16 mA and source 400 mA. Since the maximum input currents are 1.6 mA (low state) and 40 μA (high state), standard TTL has a fanout of 10, meaning that one standard TTL gate can drive 10 others. Fanout has different values when you mix TTL types.

Open-collector devices have only the pull-down transistor; the pull-up transistor is omitted. Because of this, open-collector devices can be wire—ORed through a common pull-up resistor. This connection is inherently slow because the time constant is relatively long.

Three-state devices have replaced open-collector devices in most applications because they are much faster. These newer devices have a control input that can turn off the device. When this happens, the output floats and presents a high impedance to whatever it is connected to. Three-state devices are widely used for connecting to buses.

A CMOS inverter uses complementary MOSFETs in a push-pull arrangement. The key advantage of CMOS devices is the low power dissipation. The main disadvantage is the slow switching speed.

The 74C00 series is a line of CMOS circuits that are pin-for-pin and function-for-function compatible with TTL devices. The static power dissipation of 74C00 devices is approximately 10 nanowatts (nW) per gate. Active power dissipation is higher because of the current spikes during transitions. Lower supply voltages increase the propagation delay time and noise immunity. Higher supply voltages increase the power dissipation. The best compromise is a supply voltage from 9 to 12 V. The 74HC00 series is a line of high-speed CMOS devices. The CD4000 series is another line of CMOS devices with many functions not available in the 74C00 series.

CMOS devices are guaranteed to work reliably over a temperature range of –40 to +85°C and a supply range of 3 to 15 V. Unused inputs should be returned to the supply voltage or to ground, depending on which connection prevents a stuck output. A floating CMOS input is poor design because it produces large power dissipation. CMOS devices have a fanout of 10 when driving other CMOS devices. By using level shifting, CMOS devices can be interfaced with TTL devices.

The 74HCT00 series is completely TTL-compatible, and special interfacing is not required.

GLOSSARY

- ***active load*** A transistor that acts as a load for another transistor.
- ***active power dissipation*** The power dissipation of a device under switching conditions. It differs from static power dissipation because of the large current spikes during output transitions.
- ***bipolar*** Having two types of charge carriers: free electrons and holes.
- ***bus*** A group of wires that transmits binary data. The data bus of a first-generation microcomputer has eight wires, each carrying 1 bit. This means that the data bus can transmit 1 byte at a time. Typically, the byte represents an instruction or data word that is moved from one register to another.
- ***chip*** A small piece of semiconductor material. Sometimes chip refers to an IC device including its pins.
- ***CMOS inverter*** A push-pull connection of *p*- and *n*-channel MOSFETs.
- ***compatibility*** Ability of the output of one device to drive the input of another device.
- ***interface*** The way a driving device is connected to a loading device. All the circuitry between the output of a device and the input of another device.
- ***fanout*** The maximum number of TTL loads that a TTL device can reliably drive.
- ***low-power Schottky TTL*** A modification of standard TTL in which larger resistances and Schottky diodes are used. The larger resistances decrease the power dissipation, and the Schottky diodes increase the speed.
- ***noise immunity*** The amount of noise voltage that causes unreliable operation. With TTL it is 0.4 V. As long as the noise voltages induced on connecting lines are less than 0.4 V, the TTL devices will work reliably.
- ***saturation delay time*** The time delay encountered when a transistor tries to come out of the saturation region. When the base drive switches from high to low, a transistor cannot instantaneously come out of hard saturation; extra carriers must first flow out of the base region.
- ***Schmitt trigger*** A digital circuit that produces a rectangular output. The input waveform may be sinusoidal, triangular, distorted, and so on. The output is always rectangular.
- ***sink*** A place where something is absorbed. When saturated, the lower transistor in a totem-pole output acts as a current sink because conventional charges flow through the transistor to ground.
- ***source*** The upper transistor of a totem-pole output acts as a source because conventional flow is out of the emitter into the load.
- ***standard TTL*** The basic TTL design. It has a power of dissipation of 10 mW per gate and a propagation delay time of 10 ns.
- ***three-state TTL*** A modified TTL design that allows us to connect outputs directly. Earlier computers used open-collector devices with their buses, but the passive pull-up severely limited the operating speed. By replacing open-collector devices with three-state devices, we can significantly reduce the switching time needed to change from the low state and the output state. The result is faster data changes on the bus, which is equivalent to speeding up the operation of a computer.

PROBLEMS

Section 14.1

14.1 For each assigned circuit in Fig. 14.47, determine the indicated current I and/or voltage V.

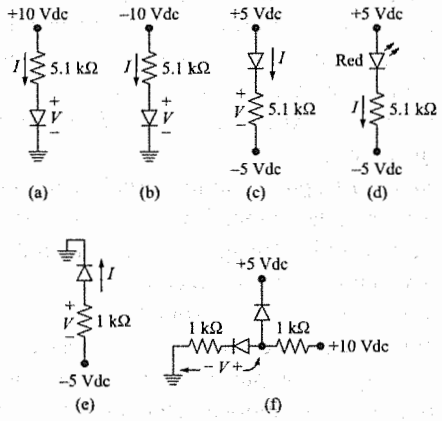


Fig. 14.47

14.2 From memory, draw the symbols for: diode, LED, *npn* BJT, *pnp* BJT.

14.3 Make a truth table for the circuit in Fig. 14.5a.

14.4 From memory, draw the simplified symbols for an *n*-channel MOSFET and a *p*-channel MOSFET.

14.5 For each assigned circuit in Fig. 14.48, determine the indicated current I and/or voltage V.

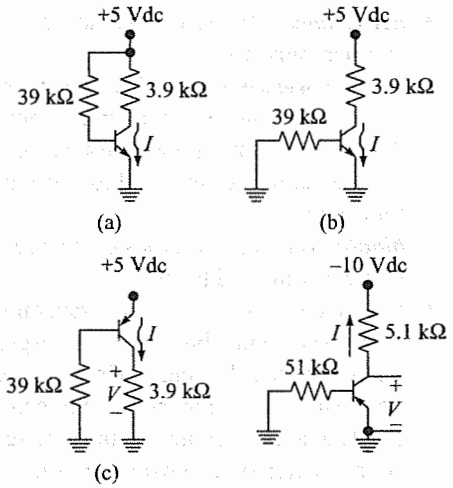


Fig. 14.48

14.6 Draw the circuit for a CMOS inverter.

14.7 For each assigned circuit in Fig. 14.49 on the next page, determine the indicated current I and/or voltage V.

Section 14.2

14.8 Figure 14.10 shows typical resistance values at room temperature. Here A is high, and B is grounded. Allowing 0.7 V for the base-emitter

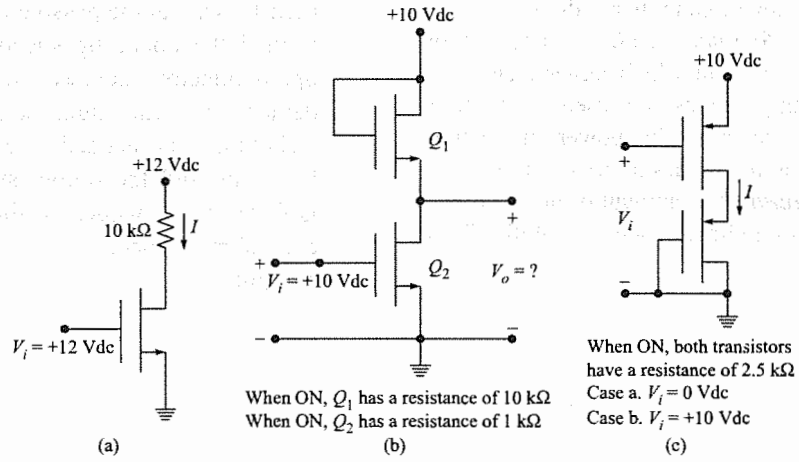


Fig. 14.49

voltage, how much current is there through the 4kΩ?

14.9 Suppose you need a TTL device with a power dissipation of less than 5 mW per gate and a delay time of less than 20 ns. What TTL type would you choose?

14.10 Use the values of Table 14.3 to calculate the total propagation delay time of three cascaded gates for each of the following TTL types:

a. Low-power
b. Low-power Schottky
c. Standard
d. High-speed
e. Schottky

Section 14.3

14.11 What is the fanout of a 74S00 device when it drives low-power TTL loads?

14.12 What is the fanout of a low-power Schottky device driving standard TTL devices?

14.13 What is the fanout of a standard TTL device driving a 74LS device?

14.14 The output of a 74LS04 is connected to the inputs of two 7404s, one 7400, and three 7410s. It seems to malfunction occasionally. What might be the problem?

14.15 What would be a simple "fix" for the circuit in Prob. 14.14?

14.16 A zero-rise-time pulse is applied to the input of a 74LS04. Its output drives a 74LS10. What is the delay time from the rising edge of the input pulse to the rising edge of the 74LS10 output?

Section 14.4

14.17 What is the fanout of a 7437 buffer when it drives standard TTL loads?

14.18 The input to a 7414 hex Schmitt trigger is a 2-V-peak sinewave. Sketch both the input and output voltages.

14.19 What is the output in Fig. 14.20 for these inputs?

a. $ABCD = 0000$
b. $ABCD = 0101$
c. $ABCD = 1100$
d. $ABCD = 1111$

14.20 Is the output Y of Fig. 14.50 low or high for these conditions?

a. Both switches open, A is low
b. Both switches closed, A is high
c. Left switch open, right switch closed, A is low
d. Left switch closed, right switch open, A is high

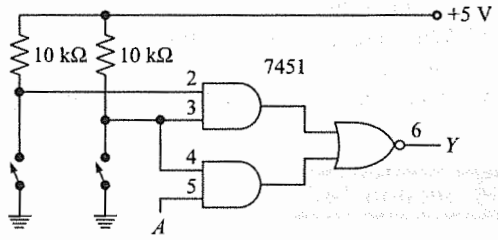


Fig 14.50

14.21 What is the value of Y in Fig. 14.51 for each of these?

a. $ABCD = 0000$
b. $ABCD = 0101$
c. $ABCD = 1000$
d. $ABCD = 1111$

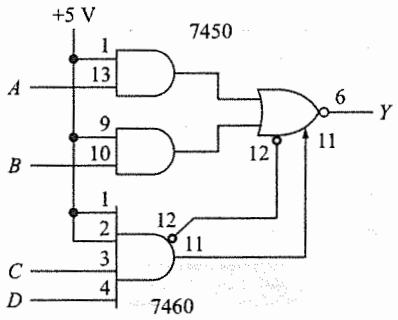


Fig. 14.51

Section 14.5

14.22 In Fig. 14.22a, $I_{OL,max} = 16$ mA. If three open-collector gates like these are wire-ORed together as shown in Fig. 14.22c, what is the minimum value of pull-up resistance needed to avoid destroying any device?

14.23 Suppose the total output capacitance is 20 pF in Fig. 14.22c. If the pull-up resistance equals 3.6 kΩ, what does the charging time constant equal?

Section 14.6

14.24 You want the contents of register *B* to appear on the bus of Fig. 14.28. What are the necessary disable values?

14.25 What are the three output conditions of a three-state gate?

14.26 Draw the logic symbol for a three-state inverter.

Section 14.7

14.27 In Fig. 14.52, what does output *Y* equal when each switch is open? When either switch is closed?

14.28 What is the current drain through the pull-up resistors when both switches are closed in Fig. 14.52? What is the time constant for each input when the switches are open?

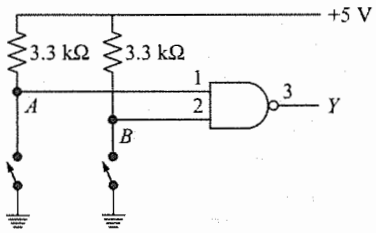


Fig. 14.52

14.29 In Fig. 14.53, what does the output *Y* equal when either switch is open? When both are closed? This is not a preferred method of driving TTL loads. Try to figure out two reasons why this circuit is not as good as the circuit shown in Fig. 14.52.

Section 14.8

14.30 In Fig. 14.54a, the TTL output voltage is 0.4 V, and the LED voltage is 2 V. What is the sink current when the LED is lighted?

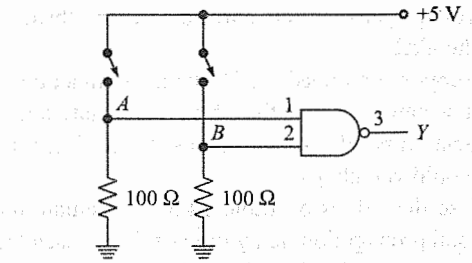


Fig. 14.53

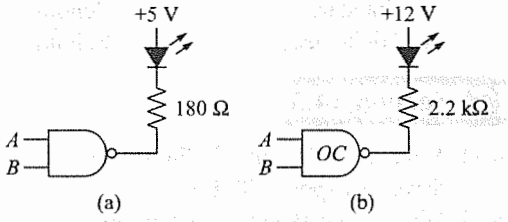


Fig. 14.54

14.31 What is the LED current in Fig. 14.54b if the LED voltage drop is 2 V and the TTL output is high? If the TTL output is 0.4 V, what is the LED current?

14.32 When switch *B* of Fig. 14.55 is closed, is the LED on or off? For this condition, what is the current in the LED?

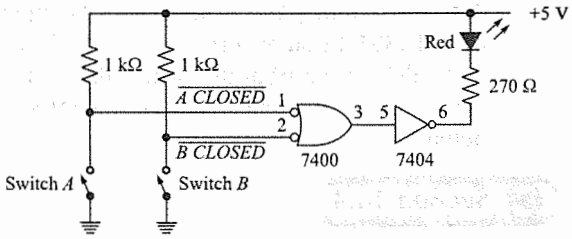


Fig. 14.55

Section 14.9

14.33 Three CMOS devices are cascaded. If each has a propagation delay time of 100 ns, what is the total propagation delay time?

14.34 A 74C00 has a load capacitance of 50 pF. If the supply voltage is 10 V, what is the active

power dissipation per gate at each of the following frequencies?

a. kHz
b. 10 kHz
c. 100 kHz

14.35 Explain the difference between 74C00, 74HC00, and 74HCT00 devices.

14.36 What are CD4000 series ICs?

Section 14.10

14.37 Figure 14.56a shows how to drive a CMOS device from a switch. As you see, the input does not float in either state. Is the output low or high when the switch is open? Is it low or high when the switch is closed?

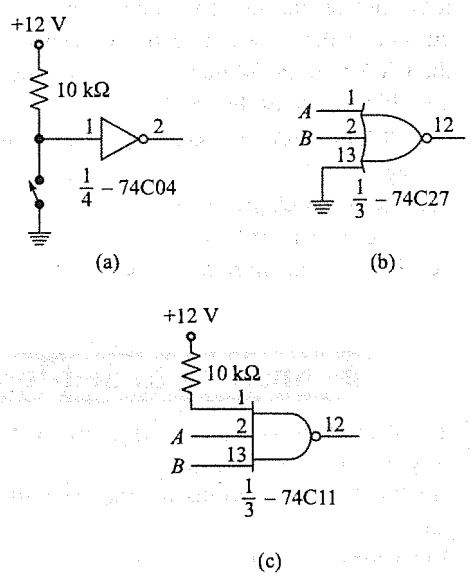


Fig. 14.56

14.38 Pin 13 is an unused input in Fig. 14.56b. As you see, it is grounded. Is the output low or high for each of these conditions?

a. *A* and *B* both low
b. *A* low and *B* high
c. *A* high and *B* low
d. *A* and *B* both high

14.39 If pin 13 is returned to the supply voltage instead of grounded in Fig. 14.56b, the circuit is useless as a NOR gate. Why?

14.40 Pin 1 is an unused input in Fig. 14.56c. As you see, it is returned to the supply voltage through a pull-up resistor. Is the output low or high for each of these conditions?

a. *A* and *B* both low
b. *A* low and *B* high
c. *A* high and *B* low
d. *A* and *B* both high

14.41 If pin 1 is grounded instead of returned to the supply voltage in Fig. 14.56c, the circuit is useless as a NAND gate. Why?

Section 14.11

14.42 If the CMOS input is low in Fig. 14.57, what is the sink current in the TTL driver? If the CMOS input is high, how much voltage drop is there across the 1.5 kΩ if the gate current is 1 μA (worst case)?

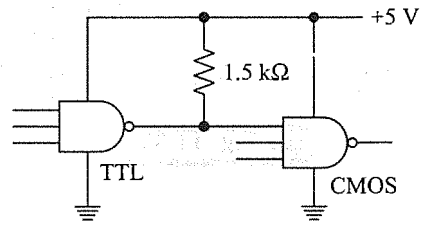


Fig. 14.57

14.43 Ideally, how much current does the open-collector driver of Fig. 14.58 have to sink when its output is low?

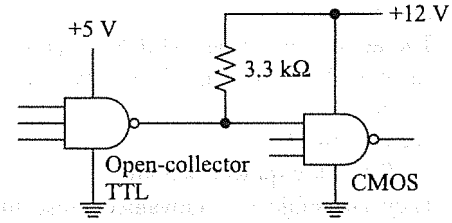


Fig. 14.58

14.44 What is the smallest acceptable value for the 3.3-kΩ resistor in Fig. 14.39 if the TTL device is a 7410 (look at maximum sink current)? What if the 7410 were replaced with a 74LS10?

14.45 The TTL in Fig. 14.40 is a 74LS10. Could a second CMOS circuit be added at the output of the 74LS10 without violating any loading rules? How many could be added?

Section 14.12

14.46 Ideally, what is the sink current I in Fig. 14.59? If the TTL load has a high-state input current of 40 μA, what is the voltage drop across the 2.2 kΩ?

14.47 If the input capacitance of the TTL load is 10 pF, what does the pull-up time constant equal in Fig. 14.59?

14.48 What is the maximum sink current for the 74C906 in Fig. 14.59?

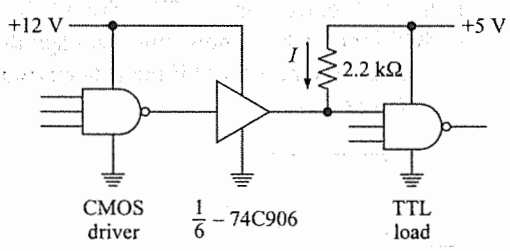


Fig. 14.59

Section 14.13

14.49 In Fig. 14.57, a logic probe indicates that the lower end of the pull-up resistor is stuck in the low state. Using a logic pulser and current tracer, you detect a current in the wire between this resistor and the TTL output. Which of the following is a possible trouble?

a. 1.5 kΩ shorted
b. 1.5 kΩ open
c. Open trace between the TTL output and the resistor
d. TTL sink transistor shorted

14.50 The lower end of the pull-up resistor (Fig. 14.57) is stuck in the high state. With a logic pulser and current tracer, you detect a current in the wire between this resistor and the CMOS input. Which of the following is a possible source of the trouble?

a. CMOS input trace shorted to supply voltage
b. CMOS input grounded
c. TTL output trace open
d. TTL output shorted to ground

Answers to Self-tests

1. The voltage across a forward-biased LED is larger.
2. *npn* and *pnp*. A *p*-channel MOSFET is the complement of an *n*-channel MOSFET.
3. Positive.
4. The active load in an NMOS IC is an *n*-channel MOSFET used in place of a resistor.
5. See Fig. 14.11.
6. 74LS00—low-power Schottky.
7. They are subject to unwanted noise that may switch the circuit.
8. It specifies acceptable high and low input voltage levels.
9. From Table 14.3, it is 10 ns.
10. It can sink or source more current than a standard gate.
11. It will produce an output waveform with very fast rise and fall times.
12. The "width" refers to the number of AND gates.
13. A resistor to +5 Vdc
14. Slower
15. They are used to facilitate connecting two or more gate outputs in parallel.
16. The factors are dc power supply current and switching time.
17. Its purpose is to protect the base-emitter of the transistor from excessive voltage.
18. No, you must also have a resistor in series with the LED to limit current; otherwise, when the transistor is on, the diode and/or the transistor will burn out.

19. Q_1 and Q_4 are p-channel MOSFETs. Q_2 and Q_3 are n-channel MOSFETs.
20. The NOR gate is CMOS because it uses both n- and *p-channel transistors.*
21. The thin oxide layer connected to the gate is easily damaged by static electricity.
22. The transfer characteristic of a CMOS inverter is a plot of input voltage versus output voltage (Fig. 14.38).
23. Its purpose is to raise the minimum TTL high output level above the lowest allowable CMOS high input level.
24. A level shifter is used between a TTL gate driving a CMOS gate; it is used to make their high and low levels compatible.
25. Yes, no
26. The CMOS has current sink and source limitations.

Applications

15

OBJECTIVES

- **Understand the multiplexing techniques used with LED displays**
- **List and describe the main sections of a frequency counter**
- **Explain how a time measurement circuit can be designed**
- **Be familiar with the basic features of the ADC0804 A/D converter**
- **Be familiar with the basic features of the ADC3511 microprocessor compatible A/D converter**
- **Discuss how to construct a digital voltmeter using the National Semi-conductor ADD3501 chip**

This chapter is intended to tie together many of the fundamental ideas presented previously by considering some of the more common digital circuit design encountered in industry. The multiplexing of digital LED displays is considered first since it requires the use of a number of different TTL circuits studied in detail in prior chapters. Digital instruments that can be used to measure time and frequency are considered next, and the concept of display multiplexing is applied here.

A number of applications using the popular ADC0804 are presented. An intergrating-type converter, the microprocessor-compatible ADC3511, is studied in detail. Then a similar converter, the ADC3501, is used to construct a digital voltmeter. In most of the applications considered, specific TTL part numbers have been specified, but in the interest of clarity, detailed designs including pin numbers have not been provided. However, it is a simple matter to consult the appropriate data sheets for this information.

In some cases, a specific part number has not been assigned; an example of this is the 1-MHz clock: oscillator shown in Fig. 15.14, or a divide-by-10 counter in the same figure. In such cases, it is left to you

to select any one of a number of divide-by-10 circuits, or to choose an oscillator circuit such as discussed in a previous chapter, on the basis of availability, cost, ease of use, compatibility with the overall system, and other factors.

15.1 MULTIPLEXING DISPLAYS

The decimal outputs of digital instruments such as digital voltmeters (DVMs) and frequency counters are often displayed using seven-segment indicators. Such indicators are constructed by using a fluorescent bar, a liquid crystal bar, or a LED bar for each segment. LED-type indicators are convenient because they are directly compatible with TTL circuits, do not require the higher voltages used with fluorescents, and are generally brighter than liquid crystals. On the other hand, LEDs do generally require more power than either of the other two types, and *multiplexing* is a technique used to reduce indicator power requirements.

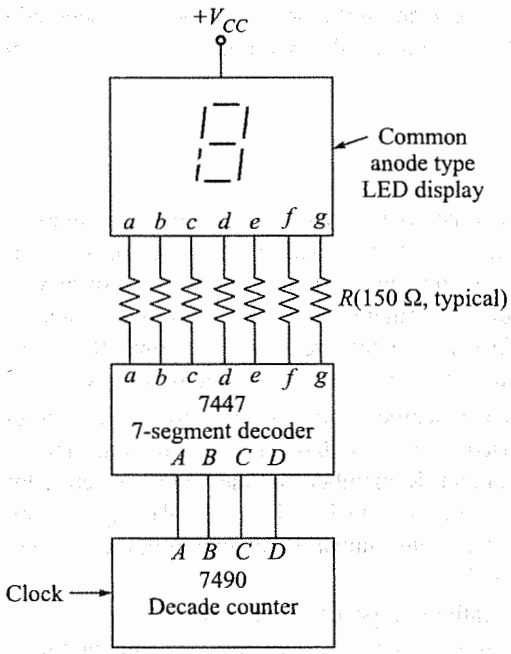
The circuit in Fig. 15.1a is a common-anode LED-type seven-segment indicator used to display a single decimal digit. The 7447 BCD to seven-segment decoder is used to drive the indicator, and the four inputs to the 7447 are the four-flip-flop outputs of the 7490 decade counter. Remember that the 7447 has active low outputs, so the equivalent circuit of an illuminated segment appears as in Fig. 15.lb. A 1-Hz square wave applied at the clock input of the 7490 will cause the counter to count upward, advancing one count each second, and the equivalent decimal number will appear on the display.

A similar single decimal digit display using a common-cathode-type LED indicator is shown in Fig. 15.2a. The seven-segment decoder used here is the 7448; its outputs are active high, and they are intended to drive buffer amplifiers since their output current capabilities are too small to drive LEDs directly. The seven *npn* transistors simply act as switches to connect $+V_{CC}$ to a segment. When an output of the 7448 is high, a transistor is on, and current is supplied to a LED segment. The equivalent circuit for an illuminated segment is shown in Fig. 15.2b. When an output of the 7448 is low, the transistor is off, and there is no segment current and thus no illumination.

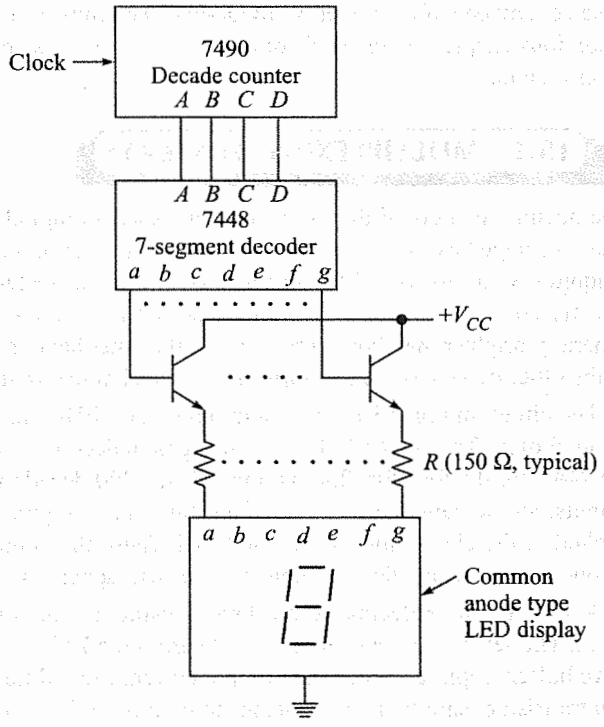
Let's take a look at the power required for the single-digit display in Fig. 15.la. A segment is illuminated whenever an output of the 7447 goes low (essentially to ground). If we assume a 2-Vdc drop across an illuminated segment (LED), a current $I = (5 - 2)/150 = 20$ mA is required to illuminate each segment. The largest current is required when the number 8 is displayed, since this requires all segments to be illuminated. Under this condition, the indicator will require $7 \times 20 = 140$ mA. The 7447 will also require about 64 mA, so a maximum of around 200 mA is required for this single digit display. An analysis of the display circuit in Fig. 15.2 will yield similar results.

A digital instrument that has a four-digit decimal display will require four of the circuits in Fig. 15.1 and thus has a current requirement of $4 \times 200 = 800$ mA. A six-digit instrument would require 1200 mA, or 1.2 A, just for the displays! Clearly these current requirements are much too large for small instruments, but they can be greatly reduced using multiplexing technique.

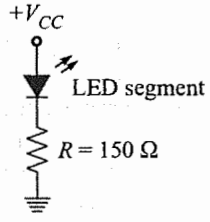
Basically, multiplexing is accomplished by applying current to each display digit in short, repeated pulses rather than continuously. If the pulse repetition rate is sufficiently high, your eye will perceive a steady illumination without any flicker. (For instance, hardly any flicker is noticeable with indicators illuminated using 60 Hz.) The single-digit display in Fig. 15.3a has +5 Vdc (and thus current) applied through a *pnp* transistor that acts as a switch. When DIGIT is high, the transistor (switch) is off, and the indicator current is zero. When DIGIT is low, the transistor is on, and a number is displayed. If the waveform in Fig. 15.3b is used as DIGIT, the transistor will be on and the segment will display a number for only 1 out of every 4 ms. Even though the display is not illuminated for 3 out of 4 ms, the illumination will appear to your eye as if it



(a) Single decimal-digit display

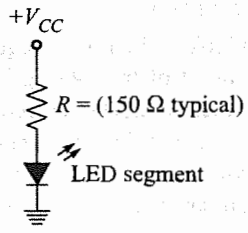


(a)



(b) Equivalent circuit for an illuminated segment

Fig. 15.1



(b) Equivalent circuit for an illuminated segment

Fig. 15.2

were continuous. Since the display is illuminated with a pulse that occurs once every 4 ms, the repetition rate (*RR*) is given as *RR* = 1/0.004 = 250 Hz. As a guideline, any *RR* greater than around 50 or 60 Hz will provide steady illumination without any perceptible flicker. The great advantage here is that this single-digit display requires only one-fourth the current of a continuously illuminated display. This then is the great advantage of multiplexing!

Let's see how to multiplex the four-digit display in Fig. 15.4a. Assume that the four BCD inputs to each digit are unchanging. If the four waveforms in Fig. 15.4b are used as the four DIGIT inputs, each digit will be illuminated for one-fourth of the time and extinguished for three-fourths of the time. Looking at the time line, we see that digit 1 is illuminated during time t_1, digit 2 during time t_2, and so on. Clearly, $t_1 = t_2 = t_3 = t_4$

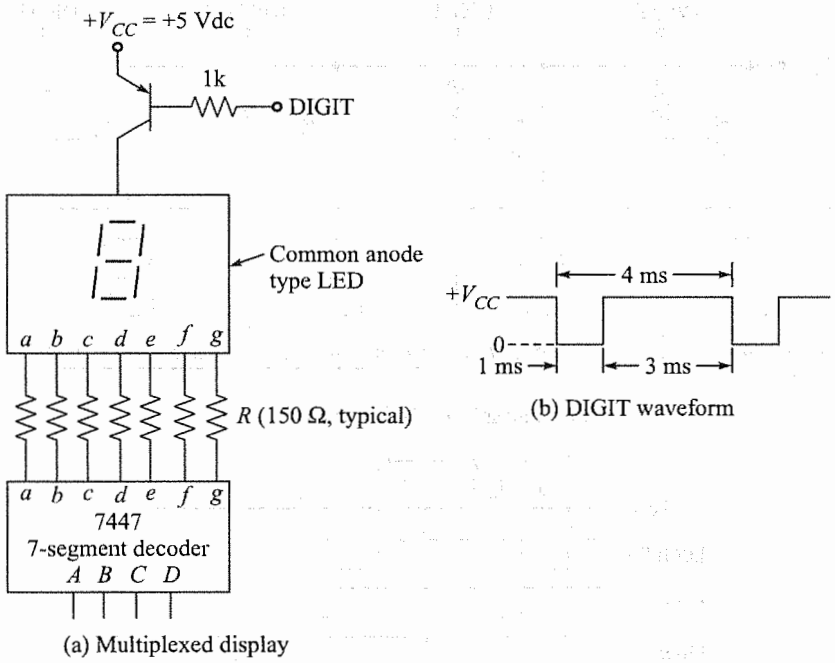


Fig. 15.3

= $T/4$. The repetition rate is given as $RR = 1/T$, and if the rate is sufficient, no flicker will appear. For instance, if t_1 = 1 ms, then T = 4 ms, and RR = 1/0.004 = 250 Hz.

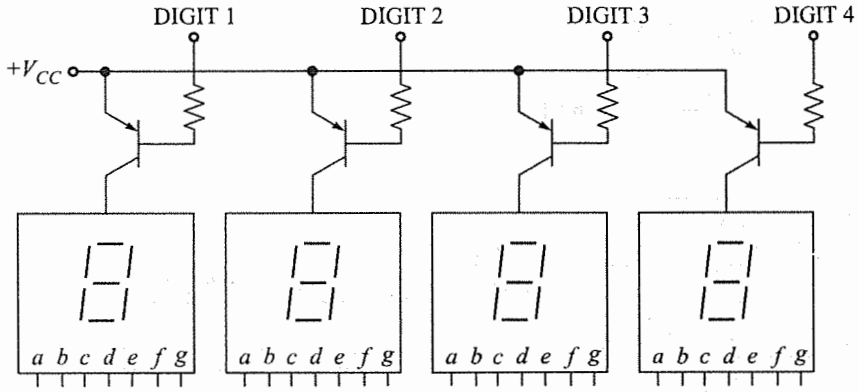
Now, here is an important concept; an illuminated digit requires 200 mA, and since only one digit is illuminated at a time, the current required from the $+V_{CC}$ supply is always 200 mA. Therefore, we are illuminating four indicators but using the current required of only a single indicator. In fact, in multiplexing displays in this way, the power supply current is simply the current required of a single display, no matter how many displays are being multiplexed!

Example 15.1 Explain the timing for a six-digit display that has a repetition rate of 125 Hz.

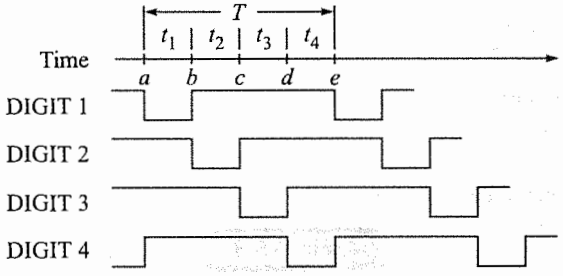
Solution An *RR* of 125 Hz means that all digits must be serviced once eyery $\frac{1}{125}$ = 8 ms. Dividing the time equally among the six digits means that each digit will be on for $\frac{8}{6}$ = 1.33 ms and off for 6.67 ms. Note that as the pulse width is decreased, the display brightness will also decrease. It may thus become necessary to increase the peak current through each segment by reducing the size of the resistors R in Figs. 15.1 and 15.2.

Example 15.2 The circuit in Fig. 15.3 show how to multiplex a common-anode-type display. Show how to multiplex a common-cathode-type display.

Solution The *npn* transistor in Fig. 15.5 is used as a switch between the cathode of the display and ground. When the transistor is on, current is allowed to pass through a segment for illumination. When the transistor is off, no current is allowed, and the segment cannot illuminate. The DIGIT waveform is shown in Fig. 15.5b. Notice that a positive pulse is required to turn the transistor on, and the display will be illuminated for 1 out of every 4 ms.



(a) A multiplexed 4-digit display



(b) Control waveforms

Fig. 15.4

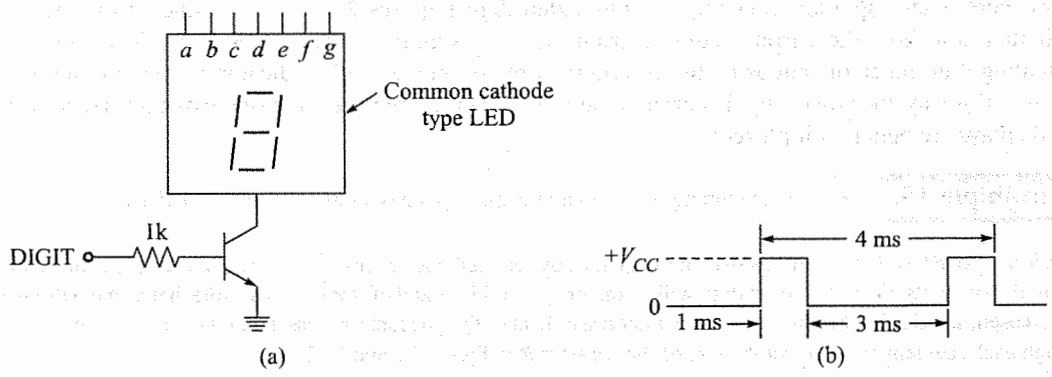


Fig. 15.5

The flip-flop outputs of a 7490 decade counter are used to drive the seven-segment decoder-driver in Figs. 15.1 and 15.2, and as long as the counter is counting, the displays will be changing states. It is often more desirable to periodically strobe the contents of the counter into a four-flip-flop latch and use these latches to drive the seven-segment decoder-driver. Then the four BCD inputs to the decoder-driver as well as the display will be steady all of the time except when new data is being strobed in. The circuit shown

in Fig. 15.6 is a complete single-digit display that will indicate the decimal equivalent of the binary number stored in the 7475 quad *D*-type latch by the positive STROBE pulse. It is also capable of being multiplexed by use of the DIGIT input.

Example 15.3 What are some possible methods for generating the DIGIT control waveforms shown in Fig. 15.4b?

Solution Reflecting back on topics covered in previous chapters, a number of different methods come to mind—for instance:

1. A two-flip-flop counter with four decoding gates
2. A four-flip-flop ring counter
3. A two-flip-flop shift counter with four decoding gates
4. A 1-of-4 low-output multiplexer

Can you think of any others?

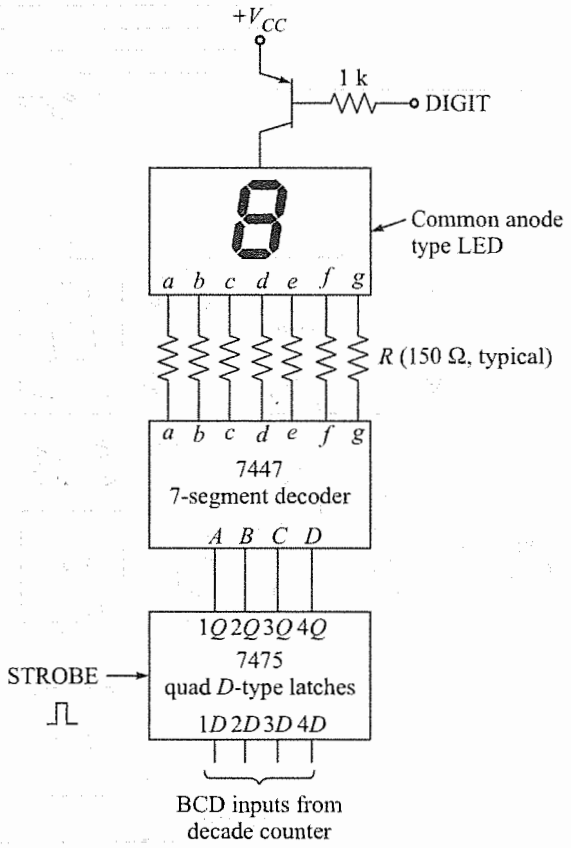


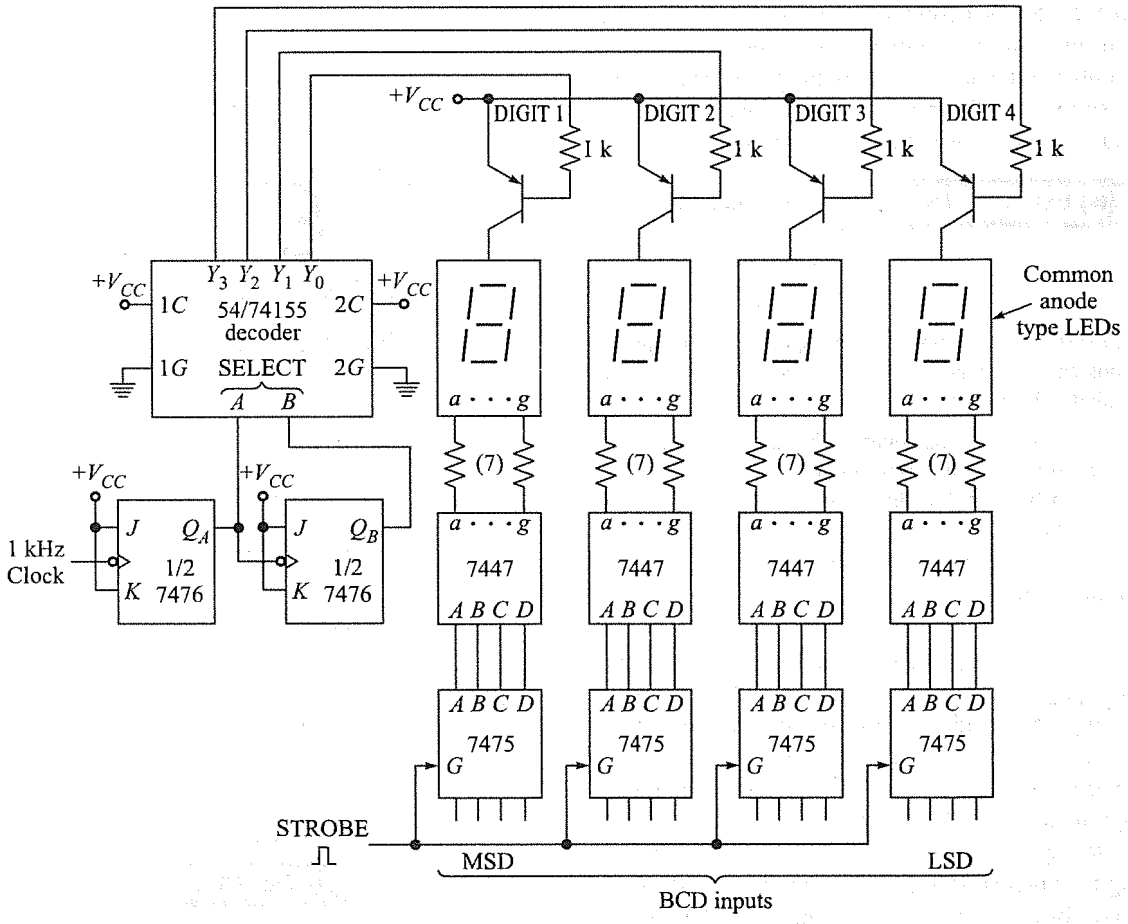
Fig. 15.6

The four-digit display in Fig. 15.7a on the next page uses four of the decimal digit displays in Fig. 15.6, and they are multiplexed to reduce power supply requirements. Notice that DIGIT 1 controls the LED on the left and this is the most-significant digit (MSD). The right display is controlled by DIGIT 4, and this is the least-significant digit counter (LSD). If we assume that the decimal point for this display to be at the right, the LSD is the units digit, and the MSD is the thousands digit. This circuit is capable of displaying decimal numbers from 0000 up to 9999. The 54/74155 is a dual 2-line to 4-line decoder-demultiplexer, and it is driven by a two-flip-flop binary counter called the *multiplexing counter*. As this multiplexing counter progresses through its four states, one and only one of the 54/74155 output lines will go low for each counter state. As a result, the DIGIT control waveforms exactly like those shown in Fig. 15.4b will be developed. You might like to review the operation of decoder-demultiplexers as discussed in Chapter 4.

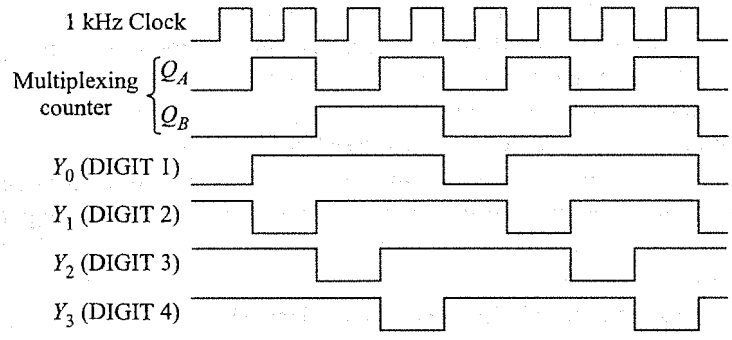
A savings in components as well as power can often be realized if the four inputs (*ABCD*) to the seven-segment decoder in Fig. 15.6 are multiplexed along with the DIGIT control. The four-digit display in Fig. 15.8 uses two 54/74153 dual 4- to 1-line multiplexers to apply the four outputs of each 7475 sequentially to a single seven-segment decoder. Here's how it works.

The BCD input data is stored in four 7475 *D*-type latches labeled 1, 2, 3, and 4. Latch 1 stores the MSD, and latch 4 stores the LSD. The 4-bit binary number representing the MSD is labeled $1_A1_B1_C1_D$. For instance, if the MSD = 7, then $1_A1_B1_C1_D = 0111$.

Each 74153 contains two multiplexers, and the four multiplexers are labeled *A*, *B*, *C*, and *D*. The *A* and *B* SELECT lines of the two multiplexers are connected in parallel and are driven by the multiplexing counter



(a) Four-decimal-digit multiplexed display



(b) Waveforms

Fig. 15.7

(exactly as in Fig. 15.7). When the SELECT inputs are $AB = 00$, the number 1 line of each multiplexer will be connected to its output. So, the multiplexer outputs connected to the 7447 decoder will be $1_A1_B1_C1_D$, which is the binary number for the MSD. This binary number is decoded by the 7447 and applied to all the LED displays in parallel. However, at this same time, DIGIT 1 is selected by the 74155 decoder, so the MSD will be displayed in the leftmost LED display. All the other displays will be turned off.

Now, when the multiplexing counter advances to count $AB = 01$, the number 2 line of each multiplexer will be selected, and the binary number applied to the 7447 will be $2_A2_B2_C2_D$, which is the next MSD (the hundreds digit). The decoded output of the 7447 is again applied to all the displays in parallel, but DIGIT 2 is the only LOW DIGIT line, so the "hundreds" digit is now displayed. (Again, all other displays are turned off during this time.)

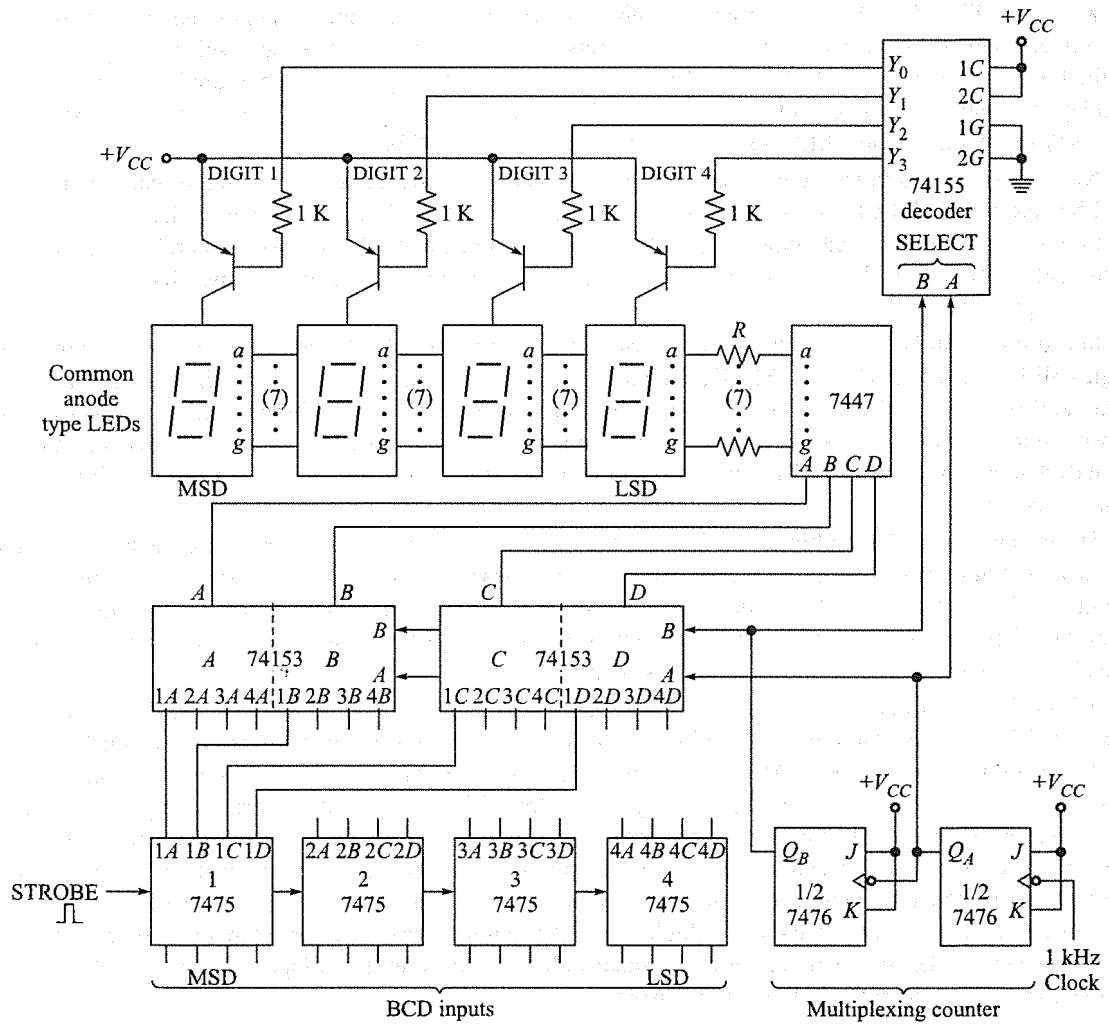
In a similar fashion, the tens digit will be displayed when the SELECT inputs are $AB = 10$, and the units digit will be displayed when the SELECT inputs are $AB = 11$. Notice that only one digit is displayed at a time, and the $RR = 250$ Hz, so no flicker will be apparent. Again, we are illuminating four digits but the power supply current is the same as for a single, continuously illuminated digit. At the same time, there is a modest saving of two chips. The savings in components increases as the number of decimal digits in the display increases.

The techniques used to multiplex the four-digit display in Fig. 15.8 on the next page are easily expanded to displays that have more than four decimal digits. It is necessary only to increase the size of the multiplexing counter and to replace the 74153 multiplexer with one that has a greater number of inputs. It is also a simple matter to alter the design to accommodate common-cathode-type LEDs instead of the common-anode types used here. (See the problems at the end of this chapter.)

All the display circuits discussed here are frequently constructed and used, but you should be aware that there are LSI chips available that have all the multiplexing accomplished on a single chip; examples of this are the National Semiconductor MM74C925, 926, 927, and 928. The MM74C925 shown in Fig. 15.9 is a four-digit counter with multiplexed seven-segment output drivers. The only external components needed are the seven-segment indicators and seven current-limiting resistors. In fact, a four-digit counter is even included on the chip! A positive pulse on the RESET input will reset the 4-bit counter, and then the counter will advance once with each negative transition of CLOCK. A negative pulse on LATCH ENABLE will then latch the contents of the counter into the four 4-bit latches. The four numbers stored are then multiplexed, decoded, and displayed on the four external seven-segment indicators. A simplified diagram is given in Fig. 15.9b. Notice that this is a common-cathode-type display.

15.2 FREQUENCY COUNTERS

A frequency counter is a digital instrument that can be used to measure the frequency of any periodic waveform. The fundamental concepts involved are illustrated in the block diagram in Fig. 15.10. The counter and display unit are exactly as described in Sec. 15.1. A GATE ENABLE signal that has a known period t is generated with a clock oscillator and a divider circuit and is applied to one leg of an AND gate. The unknown signal is applied to the other leg of the AND gate and acts as the clock for the counter. The counter will advance one count for each transition of the unknown signal, and at the end of the known time period, the contents of the counter will equal the number of periods of the unknown signal that have occurred during t. In other words, the counter contents will be proportional to the frequency of the unknown signal. For instance, suppose that the gate signal is exactly 1 s and the unknown input signal is a 750-Hz square wave. At the end of 1 s, the counter will have counted up to 750, which is exactly the frequency of the input signal.

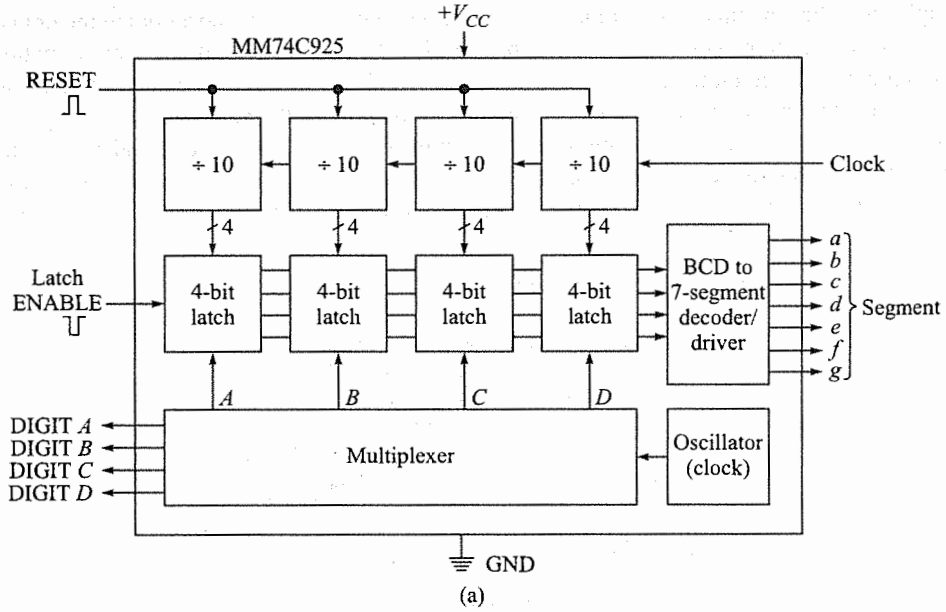


Note: For clarity, only the connections for the *MSD* are shown (1*A*, 1*B*, 1*C*, 1*D*), but the other 12 connections must be made between the input latches and the 74153s.

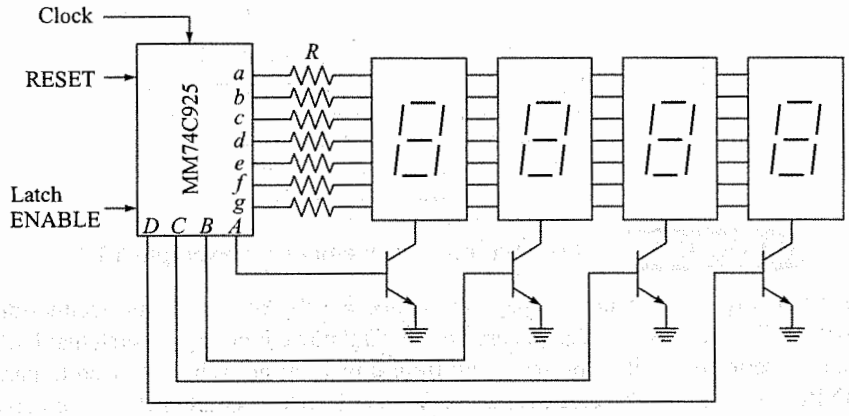
Fig. 15.8

Example 15.4 Suppose that the unknown input signal in Fig. 15.10 is a 7.50-kHz square wave. What will the display indicate if the GATE ENABLE time is $t = 0.1$ s? What if $t = 1$, and then 10 s?

Solution When $t = 0.1$ s, the counter will count up to 7500 (transitions per second) × 0.1 (second) = 750. When $t = 1$ s, the counter will display 7500 (transitions per second) × 1 (second) = 7500. When $t = 10$ s, the counter will display 7500 (transitions per second) × 10 (seconds) = 75,000. For this last case, we would have to have a five-decimal-digit display.



(a)



(b)

Fig. 15.9

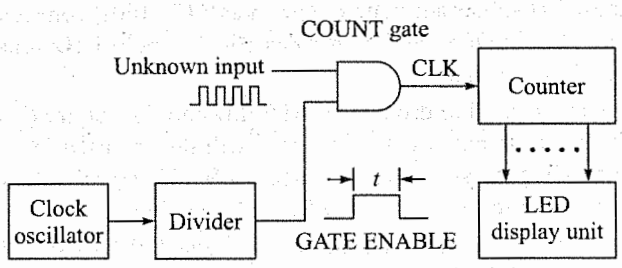


Fig. 15.10 **Basic frequency counter**

In Example 15.4, the contents of the counter are always a number that is proportional to the unknown input frequency. In this case, the proportionality constant is either 10, 1, or $\frac{1}{10}$. So, it is a simple matter to insert a decimal point between the indicators such that the unknown frequency is displayed directly. Figure 15.11 shows how the decimal point moves in a five-decimal-digit display as the gate width is changed. In the top display, the unknown frequency is the display contents multiplied by 10, so the decimal point is moved one place to the right. The middle display provides the actual unknown frequency directly. In the bottom display, the contents must be divided by 10 to obtain the unknown frequency, so the decimal point is moved one place to the left.

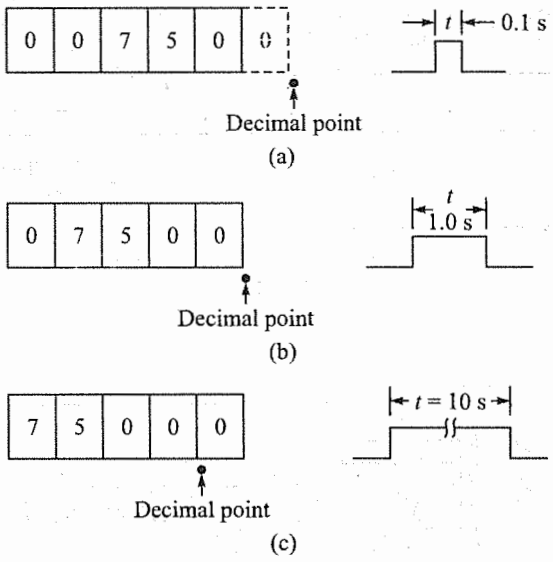


Fig. 15.11 **Decimal point movement for Example 15.4**

The logic diagram in Fig. 15.12 on the next page shows one way to construct a four-decimal-digit frequency counter. The AMPLIFIER block is intended to condition the unknown input signal such that INPUT is a TTL-compatible signal—a series of positive pulses going from 0 to +5 V dc. When allowed to pass through the COUNT gate, INPUT will act as the clock for the COUNTER. The COUNTER can be constructed from four decade counters such as 54/74160s, and it can then be connected to a multiplexed LED DISPLAY such as the one shown in Fig. 15.8. Or, COUNTER and DISPLAY can be combined in a single chip such as the MM74C925 shown in Fig. 15.9.

The DIVIDER is composed of six decade counters (such as 54/74160s) connected in series. Its input is a 100-kHz square wave from OSC CLOCK, and it provides 10-, 1-, and 0.1-Hz square wave outputs that are used to generate the ENABLE-gate signal.

When the 1-Hz square wave is used to drive the GATE flip-flop, its output, Q, is a 0.5-Hz square wave. Output Q will be high for exactly 1s and low for 1s, and it will thus be used for the ENABLE-gate signal. Notice that the 10-Hz signal will generate a 0.1-s gate and the 0.1-Hz signal will generate a 10-s gate. Let's use the waveforms in Fig. 15.12 to see exactly how the circuit functions.

A measurement period begins when the GATE flip-flop is toggled high—labeled *START* on the time line. INPUT now passes through the COUNT gate and advances the COUNTER. (Let's assume that the counter

is initially at 0000.) At the end of the ENABLE-gate time t, the GATE flip-flop toggles low, the COUNTER ceases to advance, and this negative transition of Q triggers the 74121 one-shot. Simultaneously, $\overline{Q}$ goes high, and this will strobe the contents of COUNTER into the DISPLAY latches. There is a propagation delay time of 30 ns minimum through the 74121, and then a negative RESET pulse appears at its output, $\overline{X}$. This propagation delay assures that the contents of COUNTER are strobed into DISPLAY before COUNTER is reset. The RESET pulse from the '121 has an arbitrary width of 1 μs, as set by its R and C timing components. The end of the RESET pulse is the end of one measurement period, labeled END on the time line.

For a 1.0-s gate, the decimal point will be at the right of the units digit, and the counter will be capable of counting up to 9999 full scale, with an accuracy of plus or minus one count (i.e. 1 part in 10^4). With a 10-s gate, the decimal point is between the units and the tens digits, and with a 0.1-s gate, the decimal point is one place to the right of the units digit.

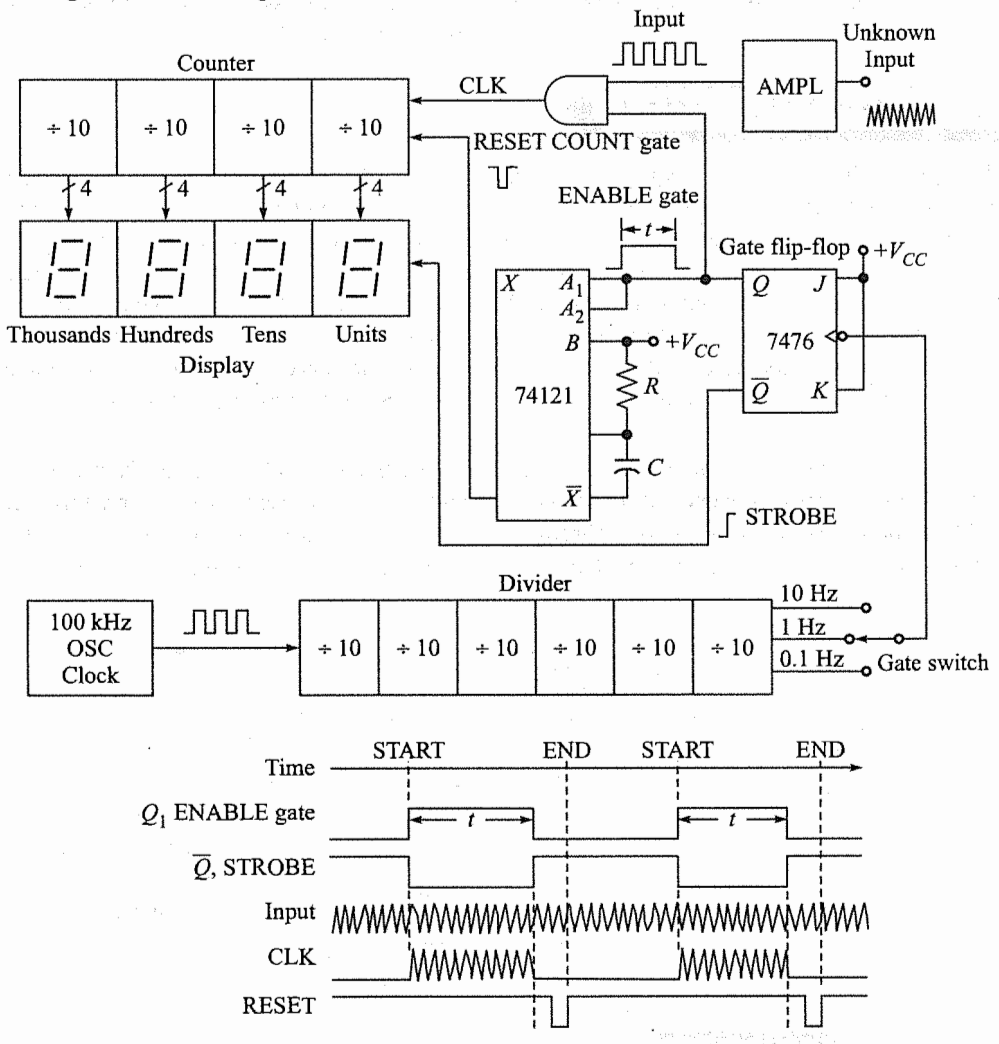


Fig. 15.12 **Four-decimal-digit frequency counter**

The CLOCK oscillator is set at 100 kHz, and this provides an accuracy on the ENABLE-gate time of 1 part in 10^4 with the 0.1-s gate. Thus the accuracy here is compatible with that of the COUNTER.

Example 15.5 Explain what would happen if the instrument in Fig. 15.12 were set on a 1-s gate time and the input signal were 12 kHz.

Solution Assuming that the counter began at 0000, the display would read 200 at the end of the first measurement period. It would then read 400, then 600, and so on at the end of succeeding periods. This is because the counter capacity is exceeded each time, and it simply recycles through 0000.

The design in Fig. 15.12 shows one method for constructing a frequency counter using readily available TTL chips, but you should be aware that there are numerous chips available that have all, or nearly all, of this design on a single chip, for instance, the Intersil ICM7226A. You will be asked to do a complete design of a frequency counter based on Fig. 15.12 in one of the problems at the end of this chapter.

15.3 TIME MEASUREMENT

With only slight modifications, the frequency counter in Fig. 15.10 can be converted into an instrument for measuring time. The logic block diagram in Fig. 15.13 illustrates the fundamental ideas used to construct an instrument that can be used to measure the period of any periodic waveform. The unknown voltage is passed through a conditioning amplifier to produce a periodic waveform that is compatible with TTL circuits and is then applied to a *JK* flip-flop. The output of this flip-flop is used as the ENABLE-gate signal, since it is high for a time t that is exactly equal to the time period of the unknown input voltage. The oscillator and divider provide a series of pulses that are passed through the count gate and serve as the clock for the counter. The contents of the counter and display unit will then be proportional to the time period of the unknown input signal.

For instance, if the unknown input signal is a 5-kHz sine wave and the clock pulses from the divider are 0.1 μs in width and are spaced every 1.0 μs, the counter and display will read 200. Clearly this means 200 μs, since 200 of these 0.1-μs pulses will pass through the COUNT gate during the 200 μs that ENABLE-gate signal is high. Naturally the counter and the display have an accuracy of plus or minus one count.

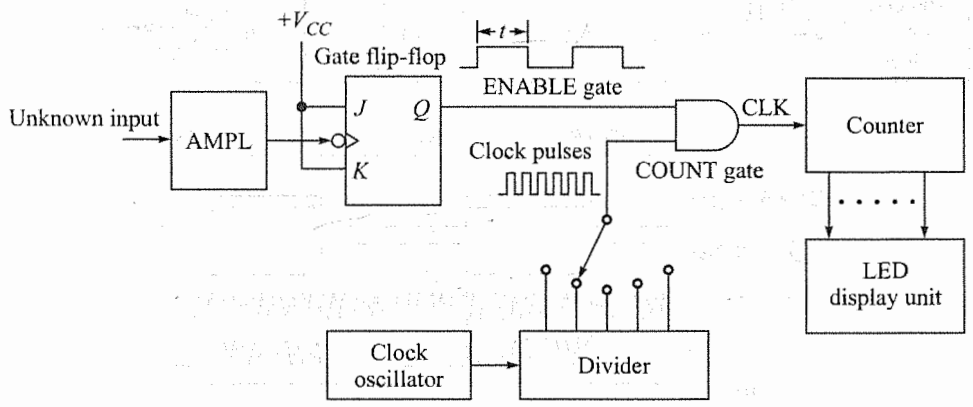


Fig. 15.13 **Instrument to measure time period**

Example 15.6 Suppose that the counter and the display unit in Fig. 15.13 have five-decimal-digit capacity and the divider switch is set to provide a 100-kHz square wave that will be used as clock pulses. What will the display read after one ENABLE-gate time t, if the unknown input is a 200-Hz square wave?

Solution Assume that the counter and the display are initially at 00000. A 200-Hz input signal will produce an ENABLE-gate time of $t = \frac{1}{200} = 5000\ \mu s$. The 100-kHz square wave used as the clock is essentially a series of positive pulses spaced by 10 μs. Therefore, during the gate time t, the counter will advance by, $\frac{5000}{10} = 500$ counts, and this is what will be viewed in the displays. Since each clock pulse represents 10 μs, the display should be read as $500 \times 10 = 5000\ \mu s$—this is the time period of the unknown input.

Example 15.7 Explain the meaning of an accuracy of plus or minus one count applied to the measurement in Example 15.6.

Solution An accuracy of plus or minus one count means that the display could read 499, 500, or 501 after the measurement period. This means that the period as measured could be 4990, 5000, or 5010 μs—in other words, 5000 plus or minus 10 μs. Since a single count represents a clock period of 10 μs, this instrument can be used for measurement only within this limit. For more precise measurement, say, to within 1 μs, the clock pulses would have to be changed from 10- to 1-μs spacing.

The circuit in Fig. 15.14 is a four-decimal-digit instrument for measuring the time period of a periodic waveform. It is essentially the same as the frequency instrument in Fig. 15.12 with only slight modifications. First, the CLOCK has been increased to 1 MHz, and DIVIDER is composed of a buffer amplifier and three decade counters. This will provide clock pulses for COUNTER with 1-, 10-, and 100-μs as well as 1-ms spacing. The unknown input is conditioned by AMPLIFIER and is then applied to the GATE flip-flop to generate the ENABLE-gate signal. STROBE and RESET are generated and applied as before. Notice that a single instrument for measuring both frequency and period could be easily designed by using a 1-MHz clock with a divider that has seven decade counters and some simple mechanical switches.

Example 15.8 Explain the DISPLAY ranges for the four-decimal-digit period measurement instrument in Fig. 15.14.

Solution With CLOCK pulses switched to the 1-μs position, each count of COUNTER represents 1 μs. Therefore, it has a full scale of $9999 \pm 1\ \mu s$. On 10 μs, it has a full scale of $9999 \times 10 = 99{,}990 \pm 10\ \mu s$. Full scale on the 0.1-ms position is $9999 \times 0.1 = 999.9 \pm 0.1$ ms. Full scale on the 1-ms position is 9999 ± 1 ms.

An interesting variation on the instrument in Fig. 15.14 is to use it to measure the time elapsed between two events. There will be two input signals, the first of which sets the ENABLE gate high and begins the count period. The second signal (or event) resets the ENABLE gate low and completes the time period. One method for handling this problem is to use the first event to set a flip-flop and then use the second event to reset it. Of course, both input signals must be first conditioned such that they are TTL-compatible. You are given the opportunity to design such an instrument in one of the problems at the end of the chapter.

15.4 USING THE ADC0804

Stand-Alone Operation

The ADC0804 was briefly introduced in Sec. 12.8. Figure 12.29 shows how to connect the ADC0804 for stand-alone operation, and it is repeated here for convenience. The recommended supply voltage is $V_{CC} = +5$

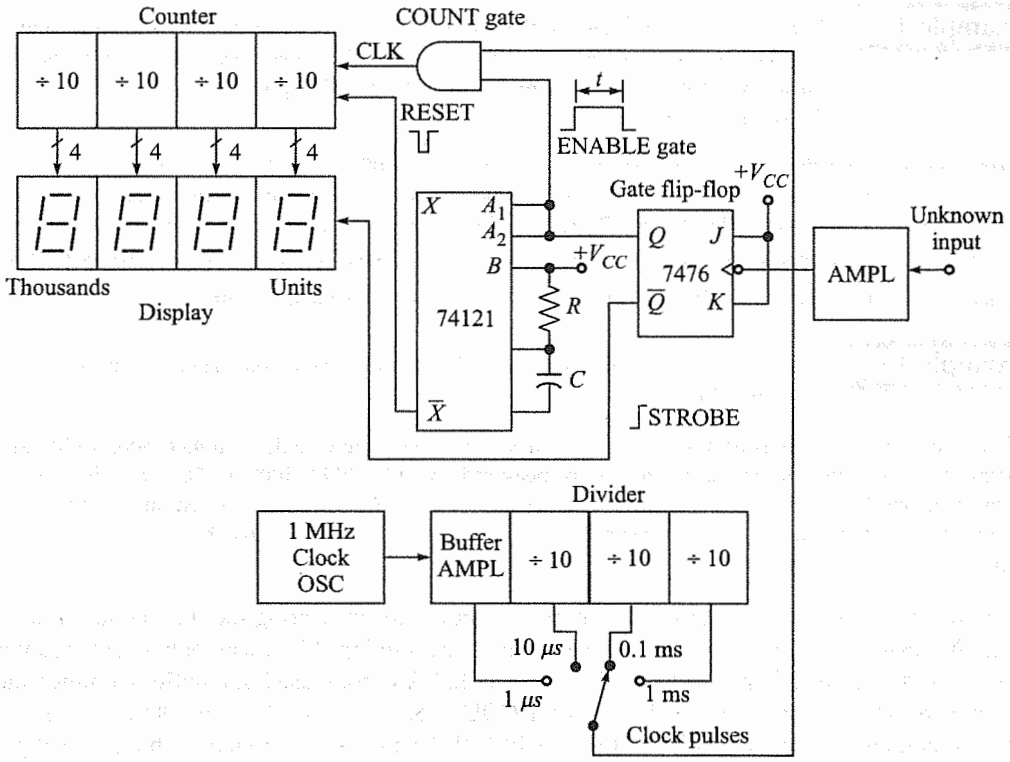


Fig. 15.14 **Four-decimal-digit period measurement instrument**

Vdc. The external resistor R and capacitor C establish the frequency of the internal clock according to

$$f \cong 1/(1.1\,RC) \tag{15.1}$$

Pin 9 is an input for an external reference voltage V_{ref}. If pin 9 is left open, the reference voltage is set internally at $V_{CC}/2$. The analog input voltage is applied between pins 6 and 7. With pin 7 connected to ground, the allowable input voltage range is from 0.0 to +5.0 V.

This ADC is designed for use with the 8080A CPU (central processing unit) chip set, composed of the 8080A microprocessor, the 8228 system controller, and the 8224 clock. It can also be used directly with the 8048 MPU (microprocessor unit). The inputs $\overline{WR}$, $\overline{INTR}$, $\overline{CS}$ and $\overline{RD}$ are microprocessor control signals. In the stand-alone mode of operation, $\overline{WR}$ and $\overline{INTR}$ are connected directly to ground. $\overline{CS}$ and $\overline{RD}$ are momentarily grounded with a push-button switch to initiate a conversion. The converter will digitize the analog voltage present at the input at the instant the push button is depressed. It will then continue to convert additional analog input voltage levels at approximately 100-μs intervals.

The digitized value of an input voltage is presented as an 8-bit binary number on pins 11 through 18, with pin 11 the most significant bit (MSB). An input voltage of 0.0 V has a digitized output of 0000 0000 (OOH). The digital output 1111 1111 (FFH) represents a full-scale input of +5.0 V. The digitized output is accurate to ±1 LSB. Since the full-scale input of 5.0 V is represented by $2^8 = 256$ bits, 1 bit (the LSB) is equivalent to an analog voltage of 5.0 V/256 = 19.53 mV.

Example 15.9 Refer to the ADC0804 in Fig. 12.29, repeated below for your reference.

(a) What is the digital output for an analog input of 2.5 V?
(b) The digital output is 0010 0010 (22H). What is the analog input?

Solution

(a) 2.5 V is one-half full scale. The digital output is then 1000 0000 ±1 bit ($2^7 = 128$). As a check, 128 × 19.53 mV = 2.5 V

(b) $(2^5 + 2^1) \times 19.53$ mV = (32 + 2) × 19.53 = 0.664 V.

Span Adjust

As shown in Fig. 12.29, the ADC0804 functions nicely for analog input voltages between 0.0 and +5.0 V. But what if the input voltage range is only from 0.0 to 2.0 V? In this case, we would like the full-scale input to be 2.0 V rather than 5.0 V. Fortunately, this is quite easy to do with the ADC0804! Simply connect an external reference voltage V_{ref} to pin 9 that is *one-half* the desired full-scale input voltage. Another term for the full-scale input voltage range is *span*. In this case, set $V_{ref} = 2.0$ V/2 = 1.0 Vdc. In general terms,

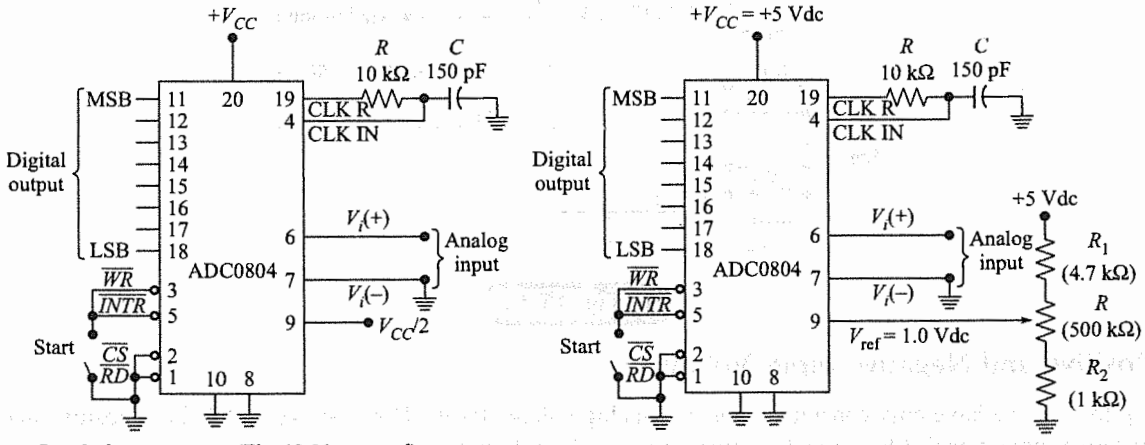
$$V_{ref} = \text{full-scale analog input voltage}/2 = \text{span}/2 \tag{15.2}$$

In Fig. 15.15, a simple resistive divider is used to generate the reference voltage V_{ref}. Here's an expression to use with this divider:

$$V_{ref} = V_{CC} \frac{R_2 + R/2}{R_1 + R_2 + R} \tag{15.3}$$

As an example, let's apply Eq. (15.3) using the circuit values given in Fig. 15.15.

$$V_{ref} = +5 \text{ Vdc} \frac{1\text{ k}\Omega + 0.25\text{ k}\Omega}{1\text{ k}\Omega + 4.7\text{ k}\Omega + 0.5\text{ k}\Omega} = 1.01 \text{ Vdc}$$



Stand-alone operation (Fig. 12.29 repeated)

Fig. 15.15

In this case, $V_{ref} = 1.0$ Vdc. The 500-Ω potentiometer will allow fine adjustment. So the full-scale analog input voltage is then 0.0 to 2.0 V. An input voltage of 2.0 V will convert to a digital output 1111 1111 (FFH). The LSB is equivalent to 2.0 V/256 ≡ 7.8 mV.

Zero Shift

The ADC0804 can also accommodate analog input voltages that are offset from zero. For instance, suppose we wish to digitize an analog signal that is always between the limits +1.5 V and +4.0 V, as illustrated in Fig. 15.16a. The span of this signal is (4.0 – 1.5) V = 2.5 V. So we would use Eq. (15.2) to find

$$V_{ref} = \frac{\text{span}}{2} = \frac{2.5 \text{ volts}}{2} = 1.25 \text{ Vdc}$$

This reference voltage (1.25 Vdc) is applied to pin 9.

Now we connect pin 7 to the *lower limit* of the input voltage. This lower limit is called the *OFFSET*. In general terms,

$$\text{OFFSET at } V_{in} = \text{analog input lower limit} \tag{15.4}$$

In Fig. 15.16b, we have used two voltage dividers, one for $V_{ref} = 1.25$ Vdc and one for $V_{i-} = 1.5$ Vdc. For this circuit an analog input voltage of 1.5 V will be digitized as 0000 0000. An input of 4.0 V will convert to 1111 1111. This LSB is then equivalent to 2.5 V/256 ≅ 9.77 mV.

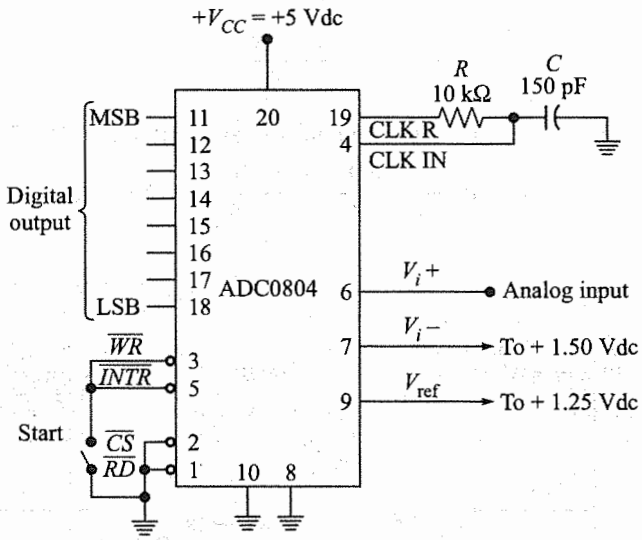


Fig. 15.16

Positive and Negative Input Voltages

Up to now, we have only considered positive analog voltage levels. How can we handle both positive and negative input signals? For instance, suppose we wish to digitize analog voltages that vary from –5 to +5 Vdc. One solution is given in Fig. 15.17. The technique is to use a resistive voltage divider (R and R) at the input pin 6. Pin 7 is connected to ground.

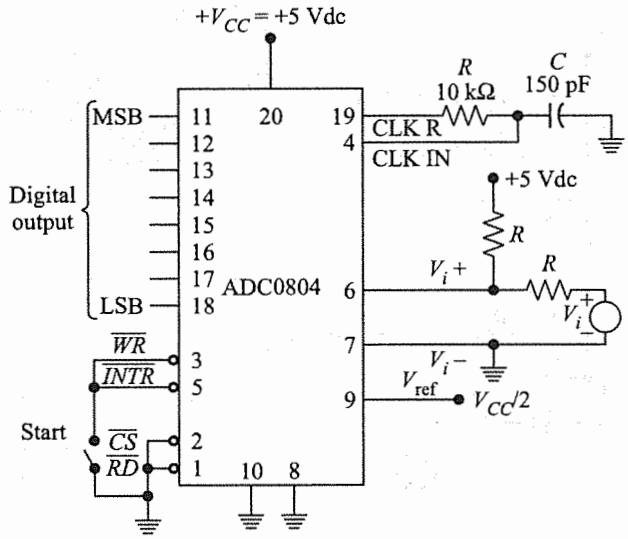


Fig. 15.17

A little thought will reveal the following:

1. $V_i = -5$ V. Then $V_{i+} = 0.0$ V. The digitized output is 0000 0000 (00H).
2. $V_i = 0.0$ V. Then $V_{i+} = +2.5$ V. The digitized output is 1000 0000 (80H). This is mid scale.
3. $V_i = +5$ V. Then $V_{i+} = +5.0$ V. The digitized output is 1111 1111 (FFH). This is full scale.

The span at V_{i+} is clearly 5.0 V. OFFSET is not required, since the voltage at V_{i+} varies between 0.0 V and +5.0 V. Notice that a *negative* input voltage, V_i, always produces a 0 for the MSB of the digital output (with the possible exception of 0.0). A *positive* input voltage, V_i, always produces a 1. for the MSB of the digital output (again, with the possible exception of 0.0). In this case, the LSB is equivalent to 10 V/256 = 39.01 mV.

Testing

When using an ADC0804, it may become necessary to test it for proper operation, for example, before initial installation or perhaps to troubleshoot a suspected malfunction. There are many different testing procedures for A/D converters, some of which are quite complex and computer-controlled. However, a rapid and simple test is to apply a known analog input voltage while monitoring the digital outputs. The test circuit in Fig. 15.18 on the next page can be used for this purpose. Notice that the dc supply voltage has been adjusted carefully to a value $V_{CC} = 5.120$ Vdc. Also, V_{ref} has been set at $V_{CC}/2 = 2.560$ Vdc. These values have been chosen so that the LSB has a weight of 5.120 V/256 = 20 mV. This eliminates any round off error and makes the arithmetic easier!

A checkerboard-type test is used to activate each output. Here's how to do it:

1. Apply an input voltage to produce the digital output 1010 1010 (AAH). The required input is (128 + 32 + 8 + 2) 20 mV = 3.400 V.
2. Apply an input voltage to produce the digital output 0101 0101 (55H). The required input is (64 + 16 + 4 + 1) 20 mV = 1.700 V.

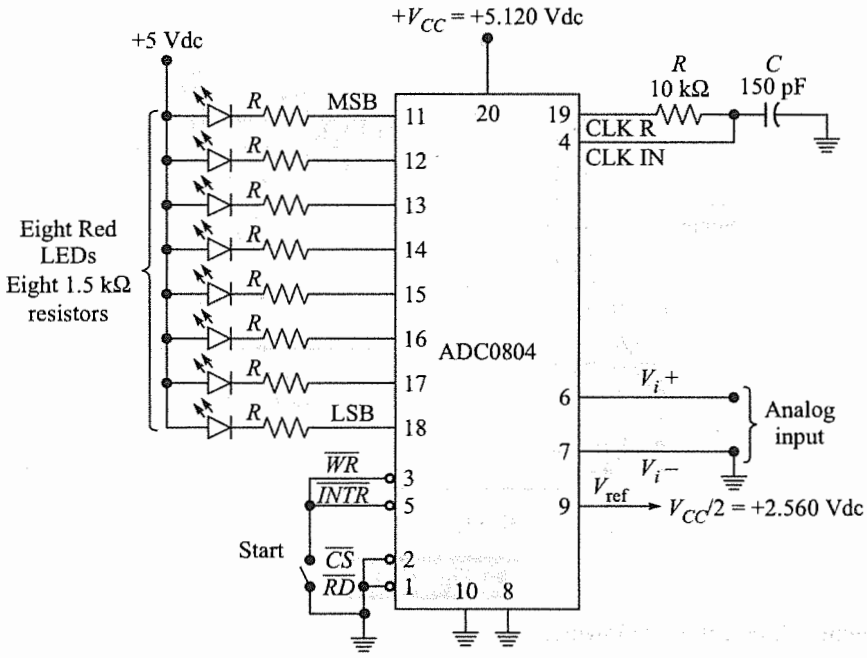
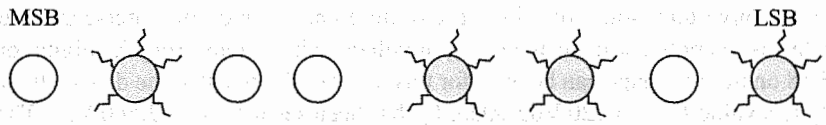


Fig. 15.18

These two tests will activate all eight outputs in both states. This is not a comprehensive test, but it will detect any faults in the outputs, and it will thus give a reasonable degree of confidence in the operation of the A/D converter. Note carefully that a digital output 1 (high) will extinguish the LED. A low ouput (a 0) will illuminate an LED. So,

$$\text{Illuminated LED} \equiv \text{low} \equiv 0$$
$$\text{Extinguished LED} \equiv \text{high} \equiv 1$$

For example, the output 1011 0010 is "seen" as



SELF-TEST

1. Why are the external *R* and *C* in Fig. 12.29 needed?
2. What is the purpose of the START button in Fig. 12.29?
3. The digital output or an ADC0804 is 1100 0011. What is this in hexadecimal?
4. For the ADC0804 what do the terms *span* and *OFFSET* mean?

15.5 MICROPROCESSOR-COMPATIBLE A/D CONVERTERS

A fundamental requirement in many digital data acquisition systems is an A/D converter that is simple, reliable, accurate, inexpensive, and readily usable with a minicomputer or microprocessor. The National Semiconductor ADC3511 is a single-chip A/D converter constructed with CMOS technology that has $3\frac{1}{2}$-digit BCD outputs designed specifically for use with a microprocessor, and it is available for less than $9! The 3511 uses an integrating-type conversion technique and is considerably slower than flash-type or SAR-type A/D converters. It is quite useful in digitizing quantities such as temperature, pressure, or displacement, where fewer than five conversions per second are adequate. The pinout and logic block diagram for an ADC3511 are shown in Fig. 15.19.

Only a single +5-Vdc power supply is required, and the 3511 is completely TTL-compatible. This A/D converter is a very high precision analog device, and great care must be taken to ensure good grounding, power supply regulation, and decoupling. It is important that a single GROUND point be eatablished at pin 13, to eliminate any ground loop currents. Voltage V_{CC} on pin 1 is used to apply +5-Vdc power. A 10-μF 10-Vdc capacitor is connected between pin 2 and GROUND; this capacitor, and the internal 100-Ω resistor shown on the logic block diagram are used to decouple the dc power used for the analog and digital circuits. Voltage V_{SS} on pin 22 should also be connected directly to GROUND.

The conversion rate of the chip is established by a resistor R connected between pins 17 and 18 and a capacitor C connected between pin 17 and GROUND. The clock frequency developed by these two components is given by $f = 0.6/RC$ and should be set between 100 and 640 kHz. This is the clock signal used to advance the internal counters thal finally store the digitized value of the analog input voltage.

The analog signal to be digitized is applied between $+V_i$ and $-V_i$, pins 11 and 10, respectively. Negative signals are handled automatically by the converter through the switching network at the input to the comparator. A conversion is initiated with a low-to-high transition of START CONVERSION on pin 7. The waveform, CONVERSION COMPLETE, on pin 6 will go low at the beginning of a conversion cycle and then return high at the end of a conversion cycle. Connecting pin 7 to $+V_{CC}$ will cause the chip to continuously convert the analog input signal. The using edge of the waveform on pin 6 indicates that new digital information has been transferred to the digit latches and is available for output.

The digitized analog signal is contained in the converter as four BCD digits. The LSD, or units digit, is $D_1C_1B_1A_1$, the tens digit is $D_2C_2B_2A_2$, the hundreds digit is $D_3C_3B_3A_3$, and the MSD is $C_4B_4A_4$. All digits can store the BCD equivalent of decimal 0, 1, 2, ... , 9, except the MSD. The MSD can have values of only decimal 0 or decimal 1. It is for this reason that the 3511 is called a $3\frac{1}{2}$ digit device—the MSD is referred to as a *half-digit*. The 16 × 4 MUX is used to multiplex one digit (4 bits) at a time to the outputs according to the input signals D_0 and D_1 as given in Fig. 15.20. For instance, when $D_1D_0 = 00$, the LSD appears on the four output lines 23, 22, 21, and 20. A low-to-high transition on DIGIT LATCH ENABLE (DLE), pin 19, will latch the inputs D_1D_0, and the selected digit will remain on the four output pins until DLE returns low. The polarity of the digitized input analog signal will also appear on pin 8, SIGN. The 3511 has a full-scale count of 1999, and if this count is exceeded, an overflow condition occurs and the four digit outputs will indicate *EEEE*.

The heart of the analog-to-digital conversion consists of the comparator, the D-type flip-flop, and an RC network that is periodically switched between a reference voltage V_{ref} and ground. When the output of the D-type flip-flop, Q, is high, the transistor designated as SW_1 is on, and the other transistor designated as SW_2 is off. Under this condition, the capacitor C charges through R toward the reference voltage (usually +2.00

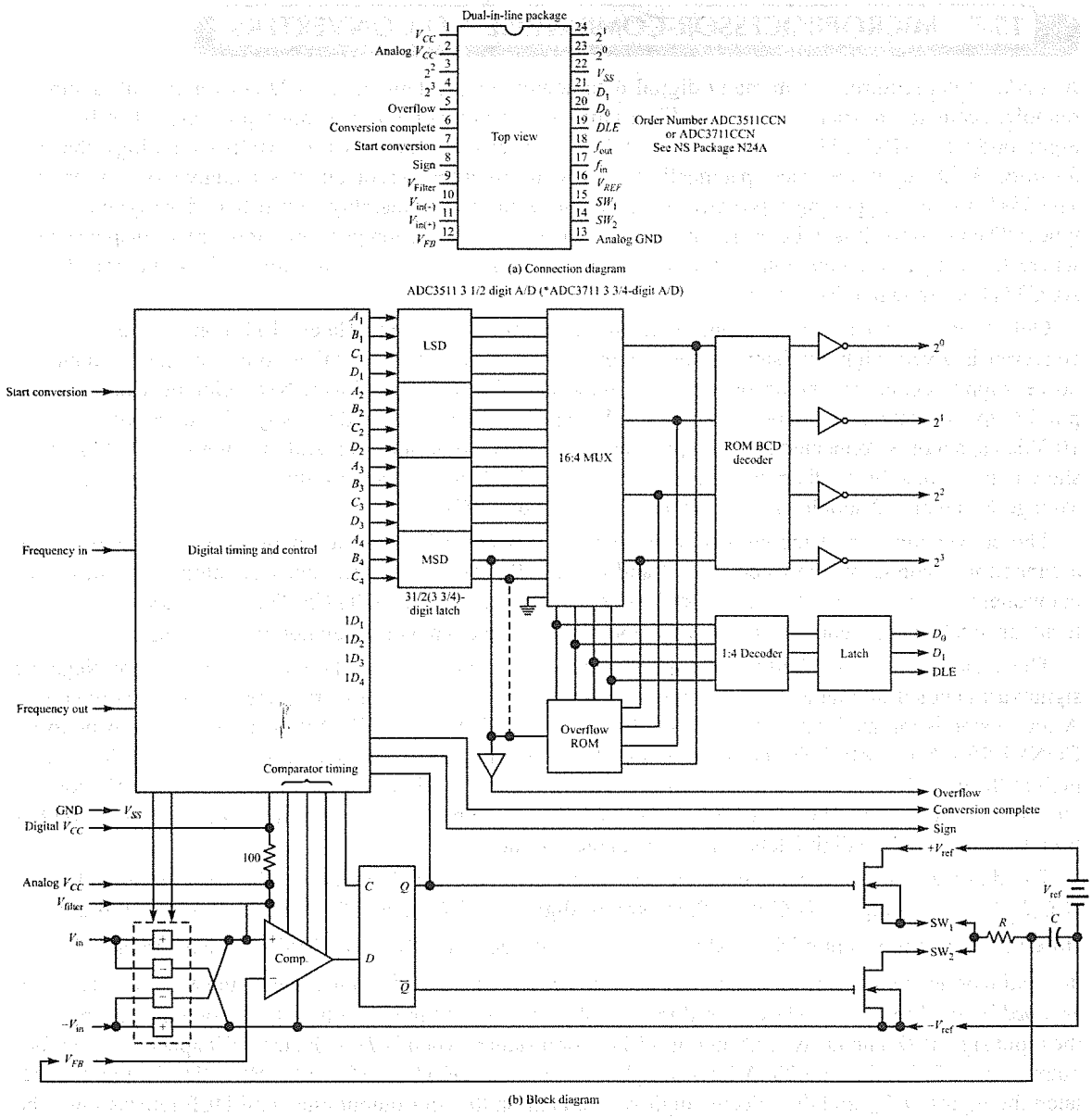


Fig. 15.19 National Semiconductor ADC3511 (3711)

Vdc), and the capacitor voltage V_{fb}, is fed back to the negative input terminal of the comparator. When V_{fb} exceeds the analog input voltage, the comparator output switches low, and the next clock pulse will set Q low. When Q is low, SW_1 is off and SW_2 is on. The capacitor now discharges through resistor R toward 0.0 Vdc, As soon as V_{fb} discharges below the analog input voltage, the comparator output will switch back to a high state, and this process will repeat.

These components form a closed-loop system that will oscillate—that is, a rectangular waveform as shown in Fig. 15.21 will be produced at SW_1 and SW_2 (pins 15 and 14).

The duty cycle of this waveform is given as

$$\text{Duty cycle} = \frac{t_c}{t_c + t_d}$$

and its dc value is given as

$$V_{dc} = V_{ref} \times \text{duty cycle}$$

This dc voltage will appear at V_{fb} and the closed-loop system will adjust itself such that

$$V_i = V_{dc} = V_{ref} \times \text{duty cycle}$$

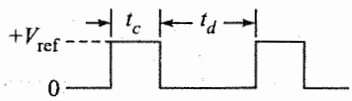
or

$$\frac{V_i}{V_{ref}} = \text{duty cycle} = \frac{t_c}{t_c + t_d}$$

DIGIT SELECT Inputs			
DLE	D_1	D_0	Selected DIGIT
L	*L*	*L*	DIGIT 0 (LSD)
L	*L*	*H*	DIGIT 1
L	*H*	*L*	DIGIT 2
L	*H*	*H*	DIGIT 3 (MSD)
H	*X*	*X*	No change

L = Low logic level
H = High logic level
X = Irrelevent-logic level
The value of the selected digit is presented at the 2^3, 2^2, 2^1, and 2^0 outputs in BCD format.

Fig. 15.20 **ADC 3511 (3711) control levels**



$$\text{Duty cycle} = \frac{t_c}{t_c + t_d}$$

Fig. 15.21 **Waveforms at SW_1 and SW_2 (pins 15 and 14 respectively) for the ADC3511**

The maximum allowable value for the analog input voltage is V_{ref}. When the input is equal to V_{ref}, the duty cycle must be equal to 1.0 ($t_d = 0$) and Q is always high. If the input analog signal is 0.0. the duty cycle must be zero ($t_c = 0$), and Q is always low. For an analog input voltage between 0.0 and $+V_{ref}$, the duty cycle is some value between 0.0 and 1.0.

The waveform Q at the output of the D-type flip-flop has exactly the same duty cycle as V_{fb}, and it is used to gate a counter in the converter. The counter can only advance when Q is high, and the gating is arranged such that for a duty cycle of 1.0, the counter will count full scale (1999), and for a duty cycle of 0, the counter will count 0000. For any duty cycle between 0.0 and 1.0, the counter will count a proportional amount between 0000 and 1999. In fact, the exact COUNT relationship is given as

$$\text{COUNT} = N \times \frac{V_i}{V_{ref}}$$

where N is the full-scale count of 2000.

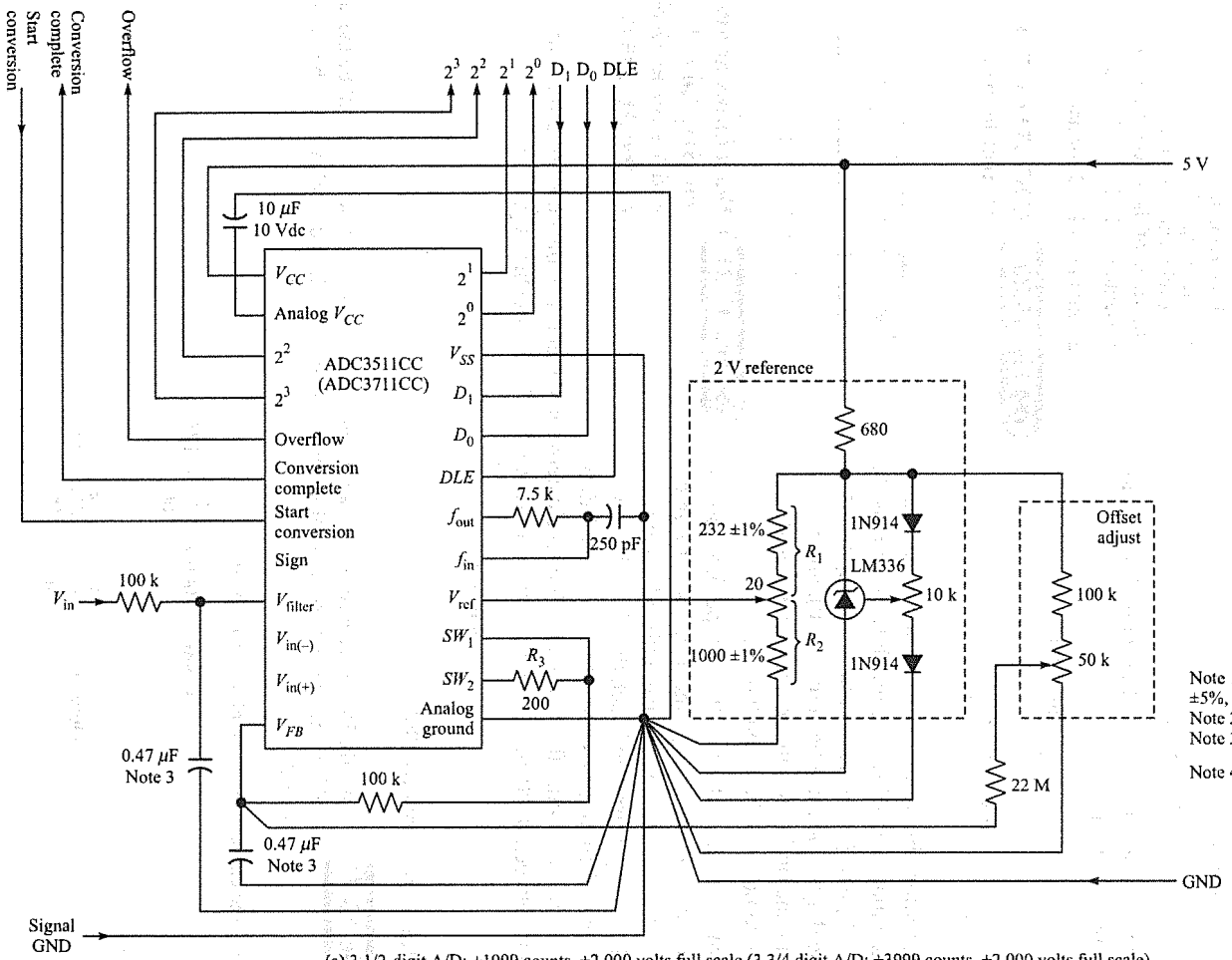
Example 15.10 An ADC3511 is connected with a reference voltage of +2.0 Vdc. What will be the count held in the counter for an analog input voltage of 1.25 Vdc? What must be the duty cycle?

Solution Using the expression given above, we obtain

$$\text{Count} = 2000 \times \frac{1.25}{2.00} = 1250$$

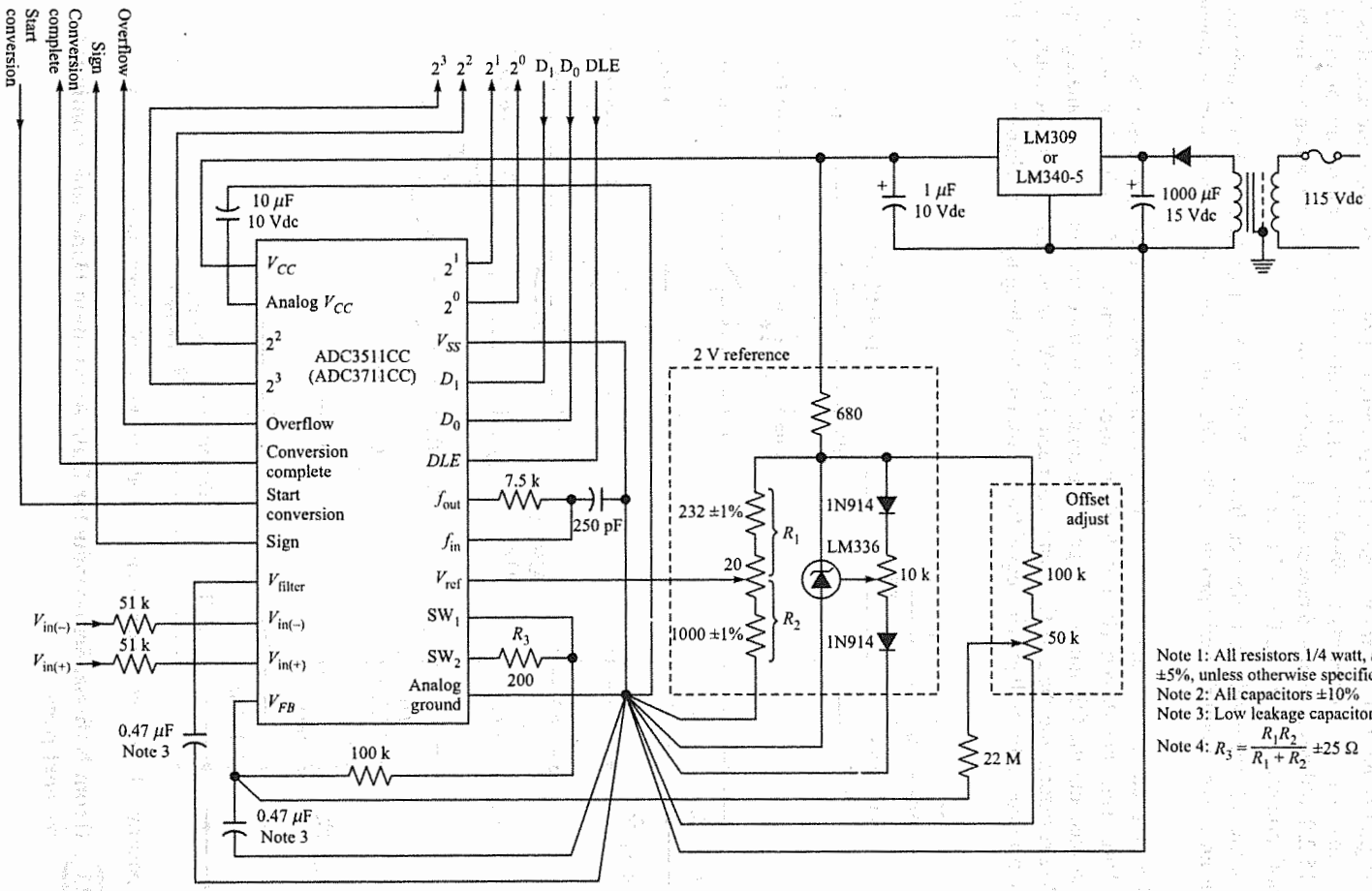
The duty cycle must be

$$\text{Duty cycle} = \frac{V_i}{V_{ref}} = \frac{1.25}{2.00} = 0.625$$



(a) 3 1/2-digit A/D; +1999 counts, +2.000 volts full scale (3 3/4 digit A/D; +3999 counts, +2.000 volts full scale)

Fig. 15.22 **(From National Semiconductor Data Acquisition Handbook; continued on next page)**



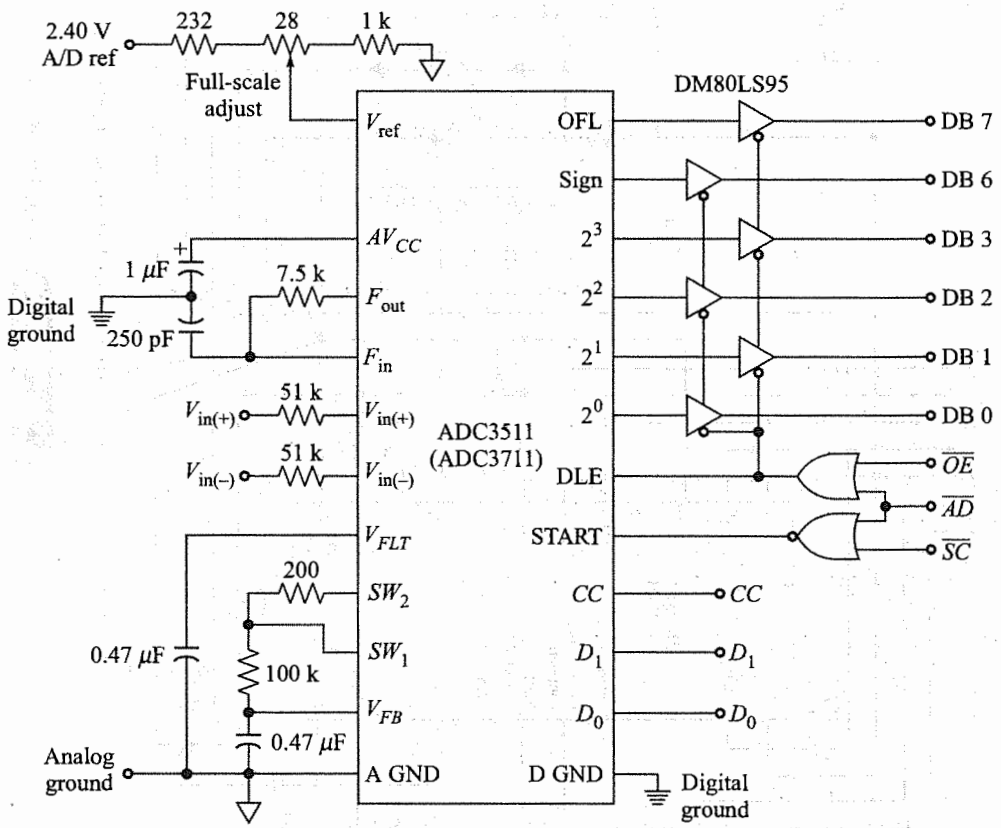
(b) 3 1/2-digit A/D; ±1999 counts, ±2.000 volts full scale (3 3/4 digit A/D; ±3999 counts, ±2.000 volts full scale)

Fig. 15.22 **(Continued)**

The circuit in Fig. 15.22a shows an ADC3511 (or an ADC3711) connected to convert 0.0 to +2.00 Vdc into an equivalent digital signal in BCD form. The 3511 converts to 1999 counts full scale and thus has a 1-bit resolution of 1 mV. The 3711 converts to 3999 counts full scale and has a 1-bit resolution of 0.5 mV. The circuit in Fig. 15.22b utilizes an isolated power supply such that the converter can automatically handle input voltages of both polarities—from +2.0 to −2.0 Vdc.

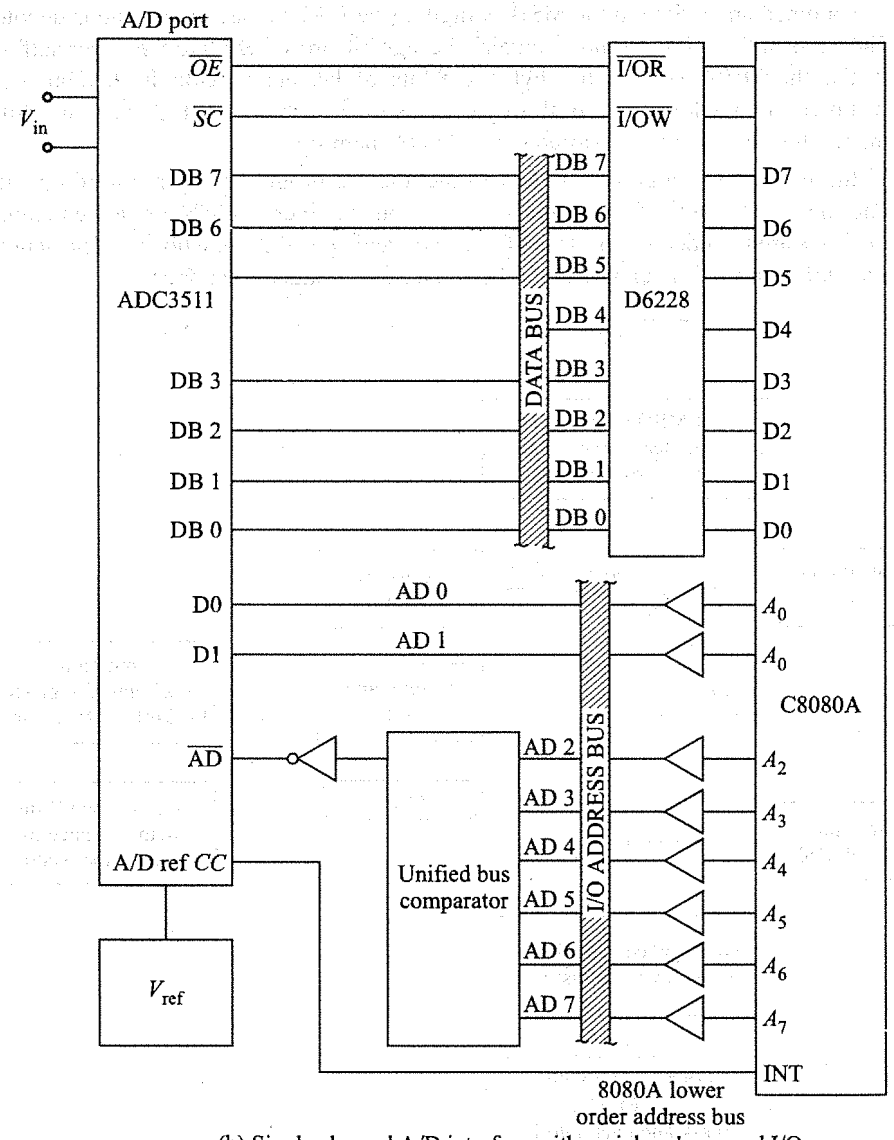
For both circuits, the reference voltage is derived from a National Semiconductor LM336, indicated by dotted lines. This is an active circuit that will provide 2.000 Vdc with a very low thermal drift of around 20 ppm/° C.

A complete circuit used to interface the ADC3511 with an 8080A microprocessor is shown in Fig. 15.23 below. Three-state bus drivers (DM80LS95) are used between the 3511 digital outputs and the microprocessor data bus, and the OR-gate–NOR-gate combination is used for control. The analog input is balanced with 51-kΩ resistors, and the 200-Ω resistor connected to SW_1 is chosen to equal the source resistance of the voltage reference; this will provide equal time constants for charging or discharging the 0.47-μF capacitor.



(a) Dual polarity A/D requires that inputs are isolated from the supply. Input range is ± 1.999 V

Fig. 15.23 **(From National Semiconductor Data Acquisition Handbook; continued on next page)**



(b) Single channel A/D interface with peripheral mapped I/O

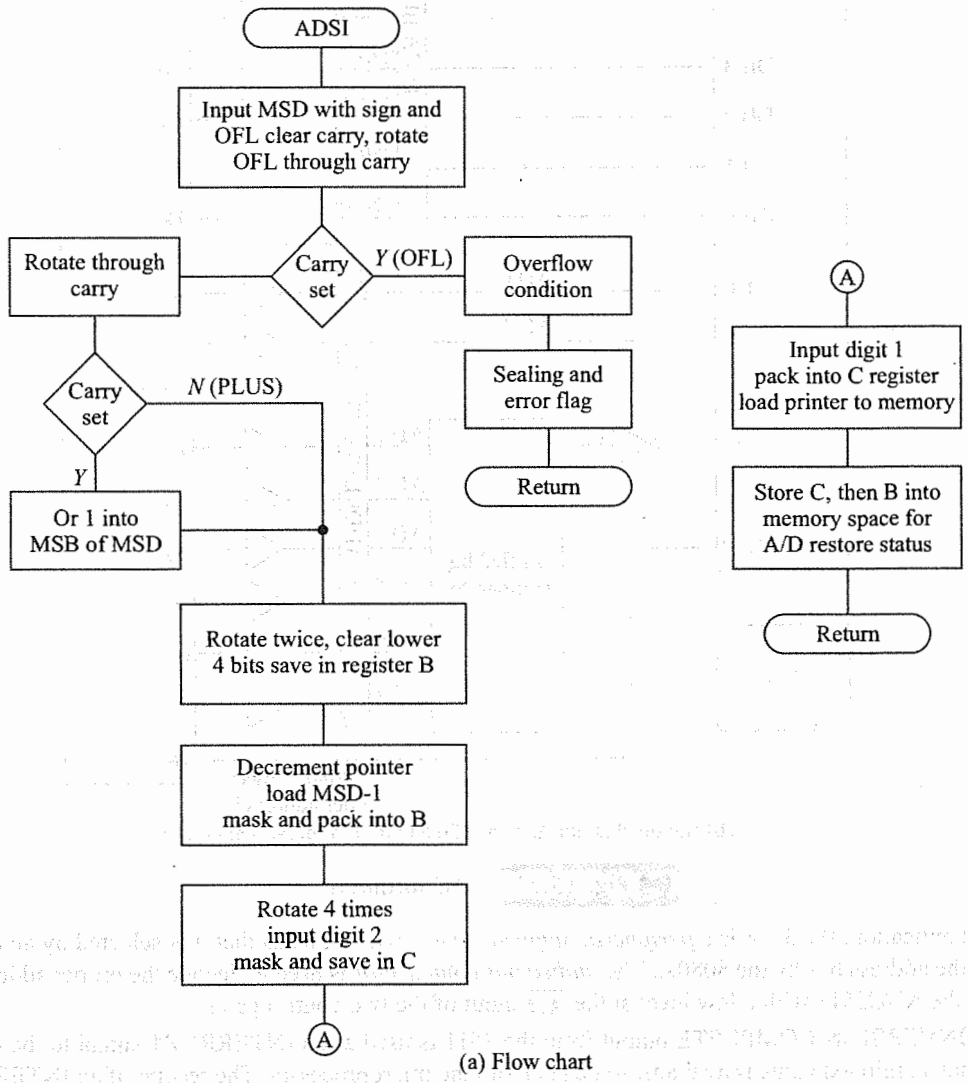
Fig. 15.23 **(Continued)**

In this application, the 3511 is a *peripheral mapped device*, which means that it is selected by an address placed on the address bus by the 8080A. The *unified bus comparator* is used to decode the proper address bits and select the ADC3511 with a low level at the $\overline{AD}$ input of the two control gates.

The CONVERSION COMPLETE output from the 3511 is used as an INTERRUPT signal to the 8080A, telling it that a digitized value is available to be read into the microprocessor. The receipt of an INTERRUPT signal causes the 8080A to read in the MSD (4 bits), the overflow (OFL), and the SIGN. If an overflow condition exists (OFL is high), an error signal is generated and the 8080A returns to its prior duties. Otherwise,

the SIGN bit is examined and stored in the MSB of digit 4 (the LSD); a negative value is denoted by a 1 in this position. The 4 bits of the LSD, that now contains the sign bit, are shifted into the upper half of the 8080A data byte. (Note that the 8080A works with 1 byte, i.e. 8 bits, of data at a time on the data bus.) The 4 bits of digit 3 are then shifted into the lower half of this byte. In a similar fashion, digits 2 and 1 are shifted into the second byte, and the four digits are now stored in the 8080A memory.

It is beyond the intent and scope of this text to include the programming required on the 8080A to interface with the ADC3511, but the flow chart and service routine given in the National Semiconductor *Data Acquisition Handbook* are included in Fig. 15.24 for the convenience of those who might presently utilize the circuit. Additional information is available in the National Semiconductor handbook.



(a) Flow chart

Fig. 15.24 **(From National Semiconductor Data Acquisition Handbook; continued on next page)**

Label	Opcode	Operand	Comment
ADIS:	PUSH	PSW	:A/D interrupt service
	PUSH	H	:save
	PUSH	B	current status
	IN	ADD 4	:input A/D digit 4
	IN	ADD 4	:delay
	ORA		:RESET carry
	RAL		:rotate OFL through carry
	JC	OFL	:overflow condition
	RAL		:rotate sign through carry
	JC	PLUS	:positive input
	ORI	20H	:OR 1 into MSB negative input
PLUS:	RAL		:shift
	RAL		:into position
	ANI	FO	:make lower bits
	MOV	BA	:save in B
	IN	ADD 3	:input digit 3
	IN	ADD 3	:delay
	ANI	OF	:mask higher bits
	OR	B	:pack into B
	MOV	B, A	:save in B
	IN	ADD 2	:input digit 2
	IN	ADD 2	:delay
	RAL		:rotate
	RAL		:into
	RAL		:upper
	RAL		:4 bits
	ANI	FO	:mask lower bits
	MOV	C, A	:save in C
	IN	ADD 1	:in digit 1
	IN	ADD 1	:delay
	ANI	OF	:mask upper bits
	OR	C	:pack
	MOV	C, A	:save in C
	LXI	H, ADMS	:load printer to A/D memory space
	MOV	M, C	:save C in memory
	INX	H	:point next
	MOV	M, B	:save B in memory
	OUT	ADD 1	:start new conversion
	POP	B	:restore
	POP	H	:previous
	POP	PSW	:status
	EI		:ENABLE interrupts
	RET		:return to main program

(b) Routine 1, single channel interrupt service routine

Fig. 15.24 (Continued)

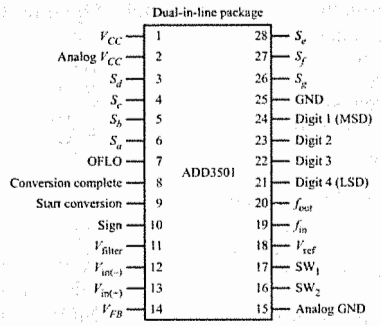
Example 15.11 Explain why it is acceptable to place the sign bit of a voltage digitized by the ADC3511 (or 3711) in the MSB of the MSD.

Solution The full-scale count for the 3511 is 1999 and for the 3711, is 3999. So, the largest value possible for the MSD in either case is 3 = 0011. Clearly the MSB is not needed for the magnitude of the MSD. It is thus convenient to specify a positive number when this bit is a 0 and a negative number when this bit is a 1.

15.6 DIGITAL VOLTMETERS

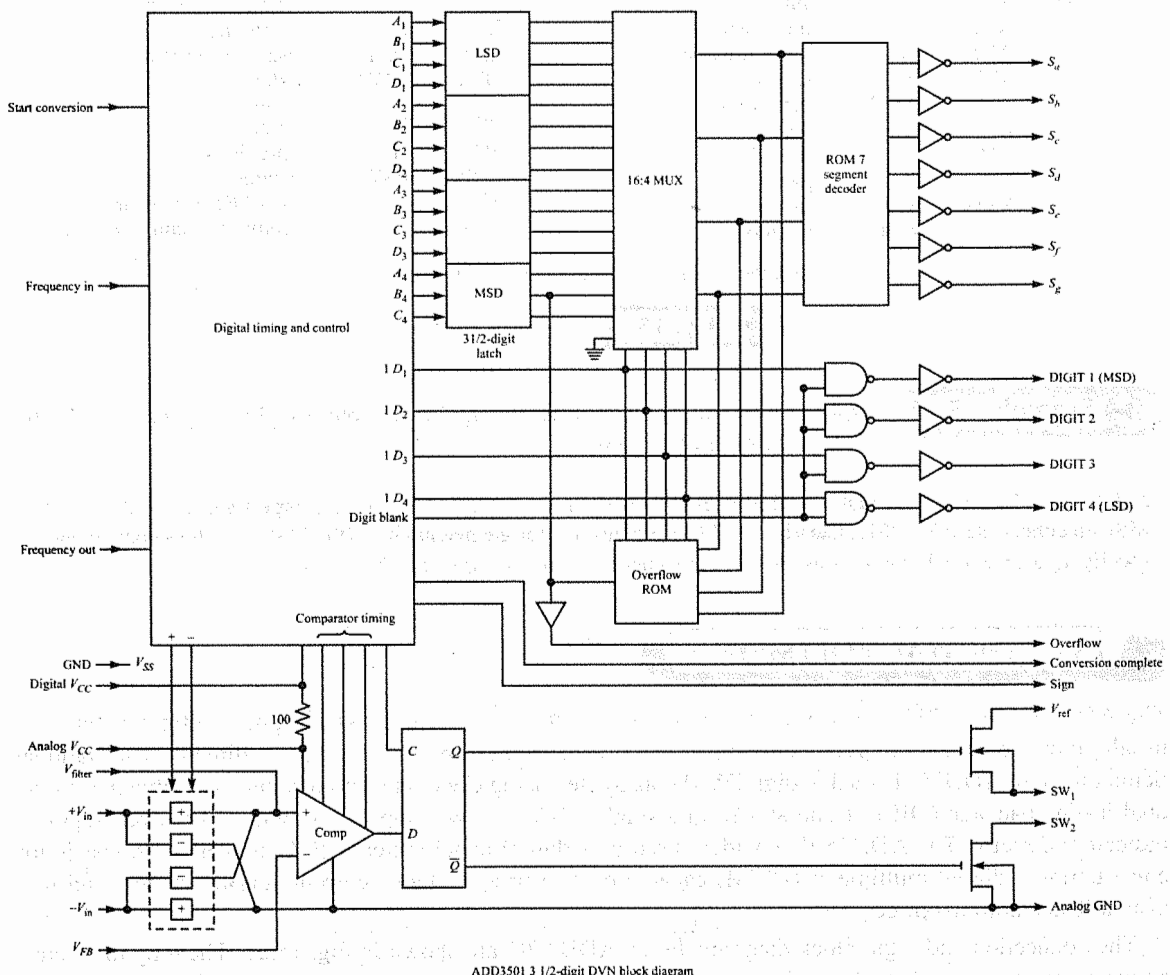
The ADC3511 (or 3711) discussed in the previous section can be used as a digital voltmeter, but it is usually more convenient to have a circuit that will drive seven-segment LED displays directly. The National Semiconductor ADD3501 is a $3\frac{1}{2}$-digit DVM constructed using CMOS technology and available in a single dual in-line package (DIP). It operates from a single +5-Vdc power supply and will drive seven-segment indicators directly. The ADD3501 is widely used as a digital panel meter (DPM) as well as the basis for constructing a digital multimeter (DMM) capable of measuring voltage, current, and resistance, and it is available at a nominal price.

The connection and logic block diagrams for an ADD3501 are shown in Fig. 15.25. The only difference between this device and the ADC3511 are the outputs. There are seven segment outputs , S_a, S_b, ... , S_g, and the four digit outputs, DIGIT 1, ... , DIGIT 4. These outputs are fully multiplexed and are designed to drive a



Order Number ADD3501CCN See NS Package N28A

Block diagram



ADD3501 3 1/2-digit DVN block diagram

Fig. 15.25 National Semiconductor ADC3501

common-cathode-type LED display directly. All the other inputs and controls are identical to the previously discussed ADC3511. The 3501 has a full-scale count of 1999 for a full-scale analog input voltage of +2.00 Vdc. A resolution of 1 bit thus corresponds to 1 mV of input voltage.

The circuit shown in Fig. 15.26 on next page shows how to use an ADD3501 as a digital voltmeter that has a full-scale analog input voltage of +2.00 Vdc. The LM309 is a voltage regulator used to reduce jitter problems caused by switching. The NSB5388 is a $3\frac{1}{2}$-digit 0.5-in common-cathode LED display. The LM336 is an active circuit which is used to provide the 2.00-Vdc reference voltage. When using this configuration, it is important to keep all ground leads connected to a single, central point as shown in Fig. 15.26, and care must be taken to prevent high currents from flowing in the analog V_{CC} and ground wires. National Semiconductor has carefully designed the circuit to synchronize the multiplexing and the A/D conversion operations in an effort to eliminate switching noise due to power supply transients.

Example 15.12 What is the purpose of the 7.5-kΩ resistor and the 250-pF capacitor connected to pins 19 and 20 of the ADD3501 in Fig. 15.26?

Solution These two components establish the internal oscillator frequency used as the clock frequency in the converter according to the relationship $f_i = 0.6/RC$. In this case, $f_i = 320$ kHz.

The DVM in Fig. 15.27 on page 589 is modified slightly in order to accommodate analog input voltages of either polarity, and also of different magnitudes. Power for the circuit is obtained from the 115-Vac power line through an isolation transformer, and the analog input is now applied at $V_{in}(+)$ and $V_{in}(-)$.

Scaling the analog input voltage for different ranges is accomplished by changing the feedback resistor between SW_1 on pin 17 and V_{FB} on pin 14, or using a simple resistance divider across the analog input. First look at the 2.00-Vdc range, since this is the normal full-scale range for the 3501. In this position, the range switch connects 100 kΩ as the feedback resistor, and the analog input goes directly to pin 13 [V_{i+}]. Also notice that the decimal point is between the 1 and the 8, giving 1.999 Vdc as a full-scale reading.

On the 0.2-Vdc scale, the range switch still applies the analog input voltage directly to pin 13. but the reference voltage at SW_1 is reduced by a factor of 10 by a resistive voltage divider before being used as a feedback voltage. The resistive divider is composed of a 90-kΩ resistor R_1, and a 10-kΩ resistor R_2. The voltage developed at the node connecting these two resistors is 0.1 V_{ref}, and so the full-scale voltage is also reduced by a factor of 10. The 90-kΩ resistor R_3 is used to keep the charging time constant essentially the same on all ranges. The time constant is given as $RC = 100\ \text{k}\Omega \times 0.47\ \mu\text{F}$. Notice that the decimal-point position has moved to pin 7 on the NSB5388 to give a full-scale reading of 199.9 mV.

On the 200-Vdc position, the range switch puts back the original feedback resistor, but the analog input voltage is reduced by a factor of 100 with a resistive voltage divider composed of 9.9-MΩ and a 100-kΩ resistor. The analog input to the 3501 is thus still 2.00 Vdc full scale even though the actual input signal is 200 Vdc full scale. The decimal point will be placed on pin 7 of the NSB5388 to give a full-scale reading of 199.9 Vdc.

On the 20-Vdc full-scale position, the range switch still uses the input voltage divider to reduce the input signal by a factor of 100, but the feedback resistor is also used to effectively increase the full scale by a factor of 10. The net result is that the 3501 will count full scale when the analog input voltage is 20 Vdc. Notice that the decimal point is now applied to pin 6 of the NSB5388 to give a full-scale reading of 19.99 Vdc.

The circuit shown in Fig. 15.28 on page 590 is a complete DMM taken from the National Semiconductor *Data Acquisition Handbook*. It utilizes the ADD3501 and is capable of measuring both dc and ac currents and voltages as well as resistances. The ranges and accuracies of the instrument are given in Fig. 15.29.

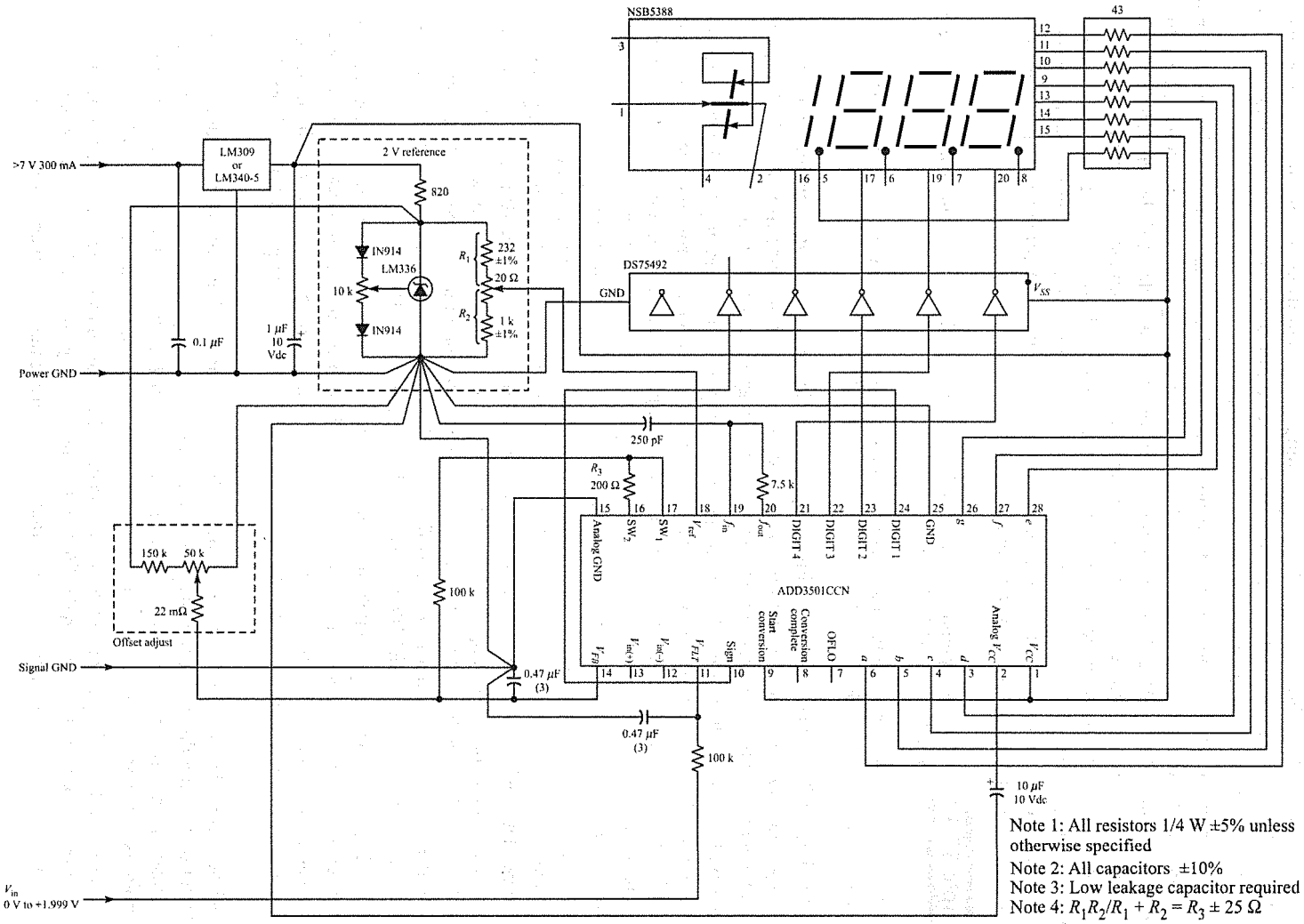


Fig. 15.26 $3\frac{1}{2}$ digit DPM, +1.999 Vdc full scale, (National Semiconductor)

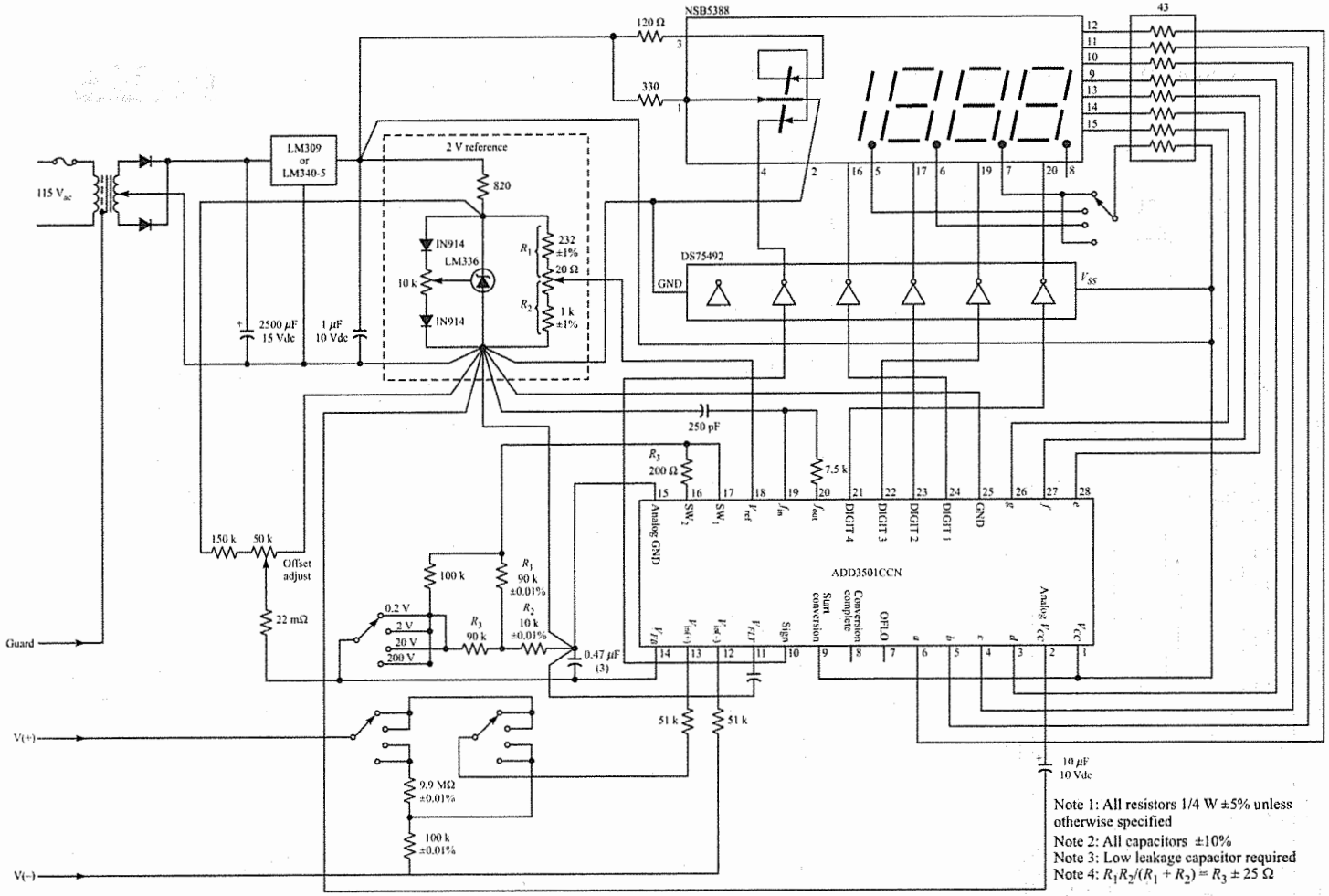


Fig. 15.27 $3\frac{1}{2}$ digit DVM, Four decade; ±0.2, +0.2, ±20, and ±200 Vdc (National Semiconductor)

Technical specifications

DC volts	< ± 1% accuracy
ranges	2 V, 20 V, 200 V, 2 kV
input impedance	2 V range, > 10 MΩ 20 V to 2 kV range, 10 MΩ
AC RMS volts	< ± 1% accuracy
ranges	2 V, 20 V, 200 V, 2 kV (40 to 5 kHz sinewave)
DC amps	< ± 1% accuracy
ranges	200 μA, 2 mA, 20 mA, 200mA, 2A
Ohms	< ± 1% accuracy
ranges	200 Ω, 2kΩ, 20kΩ, 200kΩ, 2 MΩ

Note 1: All V_{cc} connections should use a single V_{cc} point and all ground/analog ground connection should use a single ground/analog ground point.
Note 2: All resistors are 1/4 watt unless otherwise specified
Note 3: All capacitors are ±10%
Note 4: All op amps have a 0.1 μF capacitor connected across the V+ and V– supplier
Note 5: All diodes are 1N914

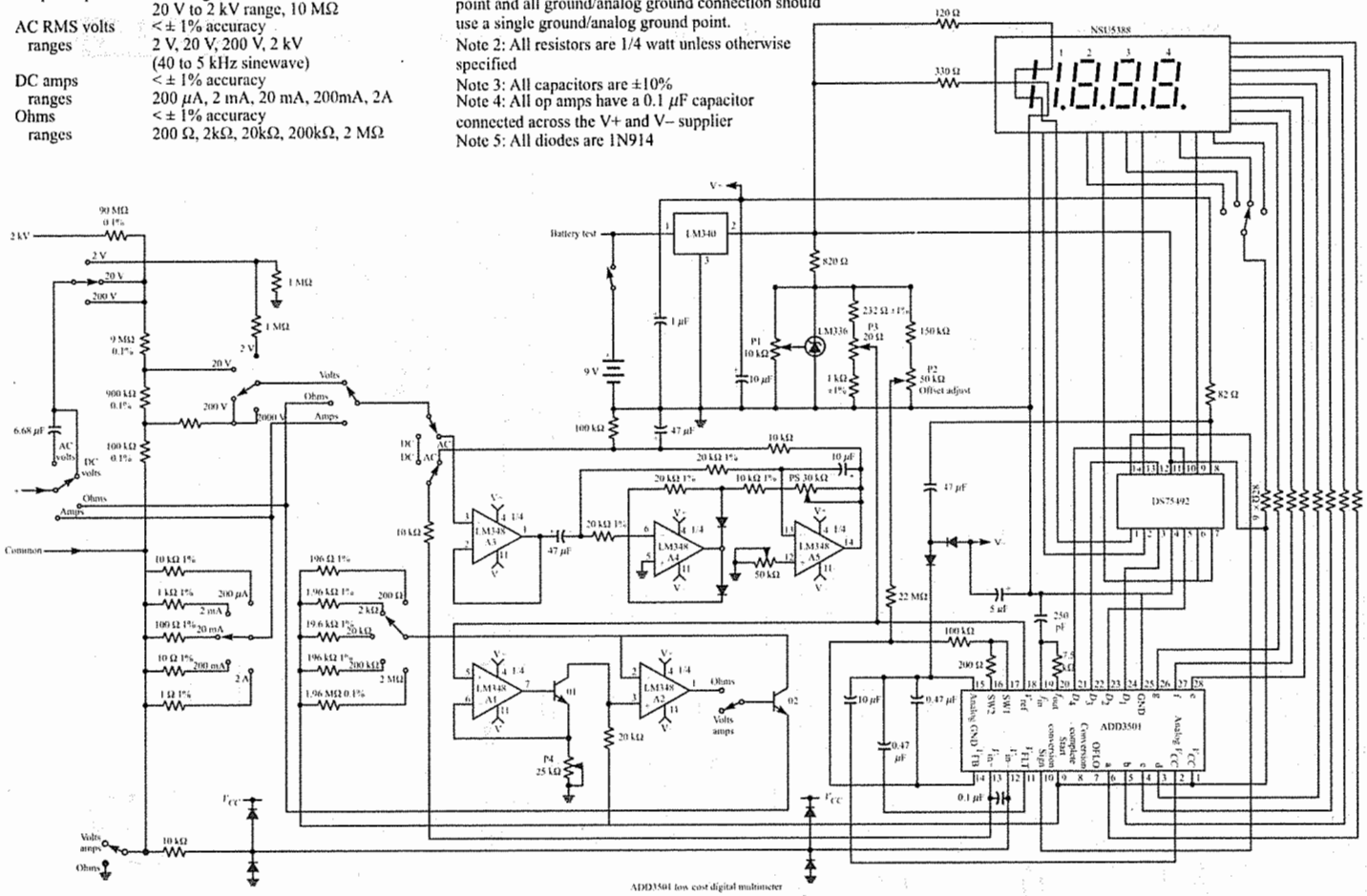


Fig. 15.28 A low-cost DMM using the ADD 3501 (National Semiconductor Data Acquisition Handbook)

The different dc and ac voltage ranges are accommodated by a resistive voltage divider at the analog input. Alternating-current voltages are measured by using the three operational amplifiers A_3, A_4, and A_5 to develop a dc voltage that is proportional to the root-mean-square (RMS) value of the ac input voltage.

Measurement mode	Range					Frequency response	Accuracy	Overrange display
	0.2	2.0	20	200	2000			
DC volts	–	V	V	V	V	–	≤ 1% FS	± OFLO
AC volts	–	V_{RMS}	V_{RMS}	V_{RMS}	V_{RMS}	40 Hz to 5 kHz	≤ 1% FS	+ OFLO
DC amps	mA	mA	mA	mA	mA	–	≤ 1% FS	± OFLO
AC amps	mARMS	mARMS	mARMS	mARMS	mARMS	40 Hz to 5 kHz	≤ 1% FS	+ OFLO
Ohms	kΩ	kΩ	kΩ	kΩ	kΩ	–	≤ 1% FS	+ OFLO

Fig. 15.29 Performance of the DMM in Fig. 15.28

A series of current-sensing resistors are used to measure either dc or ac current. The current to be measured is passed through one of the sensing resistors, and the DMM digitizes the voltage developed across the resistor.

The DMM measures resistance by applying a known current from an internal current source (operational amplifiers A_1 and A_2) to the unknown resistance and then digitizing the resulting voltage developed.

For those interested in pursuing this subject, complete details for the construction and calibration of this DMM are given in the *National Semiconductor Data Acquisition Handbook.*

SUMMARY

The primary objective of this chapter is to demonstrate the use of many of the most fundamental principles discussed throughout the text by considering some of the more common digital circuit configurations encountered in industry. Multiplexing of LED displays, time and frequency measurement, and use of digital voltmeters of all types are widely used throughout industry. Although our coverage is by no means comprehensive, it will serve as an excellent introduction to industrial practices.

The problems at the end of this chapter will also provide a good transition into industry. They are in general longer than previously assigned problems. All the necessary information required to work a given problem may not be given—this is intentional since it will require you to seek information from industrial data sheets. However, the problems are more of a design nature, and usually deal with a practical, functional circuit that can be used to accomplish a given task; as such, they are much more interesting and satisfying to solve.

PROBLEMS

In order to solve some of these problems, you may have to consult product data sheets that are not included in this text. It is intended that you discover a source for such information.

15.1 Pick one of the solutions suggested in Example 15.3 and do a detailed design, including part numbers and pin numbers.

15.2 Design a four-decimal-digit multiplexed display like the one in Fig. 15.7, but use common-cathode-type LEDs. Use a basic circuit like the one in Fig. 15.2, but you will now need to generate DIGIT waveforms that have positive pulses.

15.3 How often is each digit in Fig. 15.7 serviced, and for what period of time is it illuminated? Extinguished?

15.4 Design a multiplexed display like the one in Fig. 15.8 having eight decimal digits. Use a three-flip-flop multiplexing counter and four 74151 multiplexers.

15.5 Specify a ROM that could be used in place of the 7447 in Fig. 15.8. Draw a circuit, showing exactly how to connect it.

15.6 Design a four-decimal-digit display using 54/74143 and common-anode LEDs.

15.7 Using Fig. 15.12 as a pattern, design a four-digit frequency counter using 54/74143s and 54/74160s. Use a 1.0-MHz clock, and provide 0.1-, 1.0-, and 10.0-s gates. Specify the frequency range for each gate.

15.8 Following Fig. 15.12 as a guide, design a four-digit frequency counter using National Semiconductor MM74C925.

15.9 Design a circuit to measure "elapsed time" between two events in time—for instance, the time difference between a pulse occurring on one signal followed by a pulse occurring on another signal. Use as much of Fig. 15.14 as possible, but consider using a set-reset flip-flop in conjunction with the two input signals.

15.10 Combine the circuits in Figs. 15.12 and 15.14 into a single instrument. Use a 1.0-MHz clock and seven decade counters, define the scales and readouts carefully.

15.11 What is the internal clock frequency of the ADC0804 in Fig. 12.29 if the capacitor C is changed to 100 pF?

15.12 In stand-alone operation, how often does the ADC0804 do an A/D conversion?

15.13 Assuming that V_{CC} = +5.0 Vdc, determine the digital outputs of an ADC0804 for analog inputs of:

a. 1.25 V, b. 1.0 V

c. 4.4 V

15.14 Assuming that V_{CC} = +5.0 Vdc, in Fig. 12.29, determine the analog input voltages that will produce a digital output of:

a. 1000 1100 b. 25H

c. 0001 1000?

15.15 The ADC0804 in Fig. 12.29 is to be used with an analog signal that varies between 0.0 and +3.3 V. Determine a new value for V_{ref}.

15.16 The ADC0804 in Fig. 15.16b is to be used with an analog signal that varies between +2.2 and +3.3 V. Determine a new value for V_{ref} and V_{i-}.

15.17 Use the ADC0804 and design a stand-alone circuit to digitize an analog voltage that ranges between +0.25 and +5.0 V.

15.18 Use the ADC0804 and design a stand-alone circuit to digitize an analog voltage that ranges between –2.5 and +2.5 V.

15.19 Design a resistive voltage divider to use with the ADC3511 such that it will digitize an analog input voltage of 20 Vdc as full-scale voltage input. What is the resolution in millivolts for this design?

15.20 Design a voltage divider such that the DVM in Fig. 15.27 will measure full-scale voltages of 2.0, 20.0 and 200.0 Vdc without changing the feedback resistor. Leave the feedback resistor at 100 kΩ. Draw the complete design. Is it possible to achieve a full scale of 0.2 Vdc for this circuit without changing the feedback resistor?

Answers to Self-tests

1. R and C are needed to set the internal clock frequency.
2. Depressing the START button begins the A/D conversion process.
3. C3H
4. Span is the range of input voltage. OFFSET is the lowest value of analog input voltage.

A Simple Computer Design

16

OBJECTIVES

- Determine hardware requirement in design of a simple computer
- Discuss use of Register Transfer Language in computer design
- Design control unit of a simple computer
- Discuss how to program the simple computer in solving various problems

In this chapter, we demonstrate how the knowledge that you gathered in this book can take you to the next higher level, where you can start designing a digital computer. A digital computer is capable of computation and taking decision based on binary coded instructions stored inside it. The central processing unit (CPU), also known as the brain of the computer sequentially fetches these instructions, decodes it and then executes it by performing some action through available hardware. In this chapter, we'll design a simple computer, which has a limited instruction set but is capable enough to solve variety of arithmetic and logic problems. The technique you learn in developing this simple machine will be useful when you go for a full-fledged computer design in some higher-level courses.

We begin the chapter by defining a small problem, which our simple computer should be able to solve. Next, we spell out different hardware components required as building blocks. We'll also discuss a simple hardware operation description language, called Register Transfer Language (RTL) useful for state machine design. Through RTL we'll describe all the operations of our simple computer. Then we'll design the control unit that will coordinate all these operations. Finally, we will discuss how to program this simple computer to solve the problem we started with and many other arithmetic and logic problems.

16.1 BUILDING BLOCKS

In Section 1.6 of Chapter 1, we have broadly seen the kind of components required for designing a computer. In this chapter, we address how to design central processing unit of a simple computer that interacts with a small memory module. Before we proceed further let us define a problem that our computer is supposed to solve. This is not the only problem it can handle. Depending on how we program it, we will be able to solve different arithmetic and logic problems and that is shown towards the end of this chapter through examples. The purpose of defining a problem is to choose specific hardware components that will serve as building block of our simple computer.

The Problem

Add 10 numbers stored in consecutive locations of memory. Subtract from this total a number, stored in 11th location of memory. Multiply this result with 2 and store it in 12th location of memory. All the numbers brought from memory lie between 0 and 9.

Memory

Since, the problem says the numbers or data to be fetched from memory and we also know that programs, i.e. binary coded instructions are also stored in memory, let us divide the memory used in our computer in two parts. One part stores the program or series of instructions the computer executes sequentially and this is known as *program memory*. The other part houses data that program uses and is also used for storing result. This is called *data memory*. From the given problem we find, we need 12 memory locations for data storage. We expect our computer won't need more than 20 instructions to complete the given task hence, a memory with 32 locations (integer power of 2) can be selected for our computer.

Now we try to decide how many bits of information we store in each address location. Usually, bits in memory locations are stored in multiple of 8 called *byte*. Let's see if our job can be done with 8 bits. Each memory location stores data between 0 and 9 on which program operates and thus require only 4 bits. The final result at most can be $10 \times 9 \times 2 = 180$ which requires 8-bit for representation. So the data memory can be of 8-bits with which we can represent decimal number up to $2^8 - 1 = 255$.

Let's now see the requirement of program memory. There, in each location, certain number of bits are allocated that defines the instruction to be executed. This is called operation code or in short, *opcode*. The rest of the bits can be used for referring the memory location from which data is to be brought or stored, if required by the instruction. Since, 32 memory locations require $\log_2 32 = 5$ bits for memory referencing we'll have $8 - 5 = 3$ bits for opcode specification giving $2^3 = 8$ different opcodes (Fig. 16.1). We'll see that 8 instructions are sufficient for the given kind of task in our limited ability computer. Hence, one important hardware component of our computer gets decided. The memory to be used is of size 32×8.

The above mode of addressing memory for data is called *direct addressing*. If the address mentioned in the instruction contains address from which actual data is to be brought it is called *indirect addressing*. If after opcode, in place of address actual data is made available, it is called *immediate addressing*. Note that, in immediate addressing data cannot be more than 5 bits as 3 bits gets used in opcode. Also note that the instruction like this is called single byte instruction. If an instruction requires 2 bytes to be fetched from program memory it is called 2-

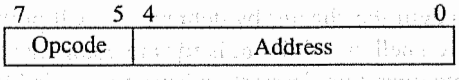


Fig. 16.1 **Three Most Significant Bits (MSB) are opcode and five Least Significant Bits (LSB) are address**

byte instruction. Obviously, in 2-byte instruction number of opcodes or memory addressing capability can be more than a single byte instruction.

Register Array

The computer needs a set of registers to perform its operation. Let us define them and assign task to each one of them for our simple computer. Note that, we are using a 32×8 memory module.

Memory Address Register (*MAR*) is a 5-bit register that stores the address of the memory location referred in a particular instruction. The output of this is fed to a 5-to-32 address decoder. Each output of the decoder points to a location in the memory. All memory referenced instruction loads memory address in *MAR*.

Memory Data Register (*MDR*) is an 8-bit register that stores the memory output when a memory read operation is performed. During memory write operation it stores the value that gets written to the memory. Thus it can also be called a memory buffer. In arithmetic or logic operation when more than one operand is required by ALU, one operand in our simple machine comes from *MDR*.

Program Counter (*PC*) is a 5-bit counter that stores the address of the memory location from which next instruction is fetched. At power on, our machine *PC* is reset so that its content is all zero. Thus location 00000 has to be a part of program memory and this is also the starting address from which program execution begins. Since, in our simple machine all the instructions are single byte instruction, every time an opcode is fetched we'll increment *PC* by one, and thus *PC* will point to location of the next opcode.

Instruction Register (*IR*) is a 3-bit register, which retains the opcode till it is properly executed in one or more clock cycles. Since all memory read and write operations are done through *MDR*, after an instruction is read from memory, 3 MSB that contains the opcode are transferred to *IR*.

Accumulator (*ACC*) is a multi-purpose register that always stores one operand of an arithmetic or logic operation. The result of this operation, i.e. ALU output is also stored in *ACC*. Functions like shifting of bits to left or right are also carried on *ACC*. Thus, in our simple computer *ACC* is a shift register with parallel load facility.

Timing Counter (*TC*) is a synchronous parallel load counter that stores and updates the timing information. The timing counter output is decoded to generate different timing signal, which in turn triggers different events in execution of an instruction. The counter is reset synchronously with clock once an instruction is fully executed. If an instruction is conceived as a *macro operation* then series of sequential steps necessary to carry out the instruction in the computer is called *micro operations*. In our simple computer, we are not expecting more than 8 micro operations for any macro-operations and hence a 3-bit counter is sufficient. Later if we see, we need more than 8 micro operations we'll change it to a 4-bit counter. Note that, a *master clock* (also called system clock) to which all the state changes of the computer are synchronized, triggers this counter. Also note, *TC* has power on reset facility, i.e. when the computer is switched on it stores 000.

Start/Stop Flag (*S*) is a flip-flop which when set, stops execution of the program. This we do in our simple computer by inhibiting the master clock. Like program counter, this also has a power-on-reset facility so that when the computer is switched on the master clock is not inhibited.

Other Important Hardware

Arithmetic Logic Unit (ALU) is a versatile combinatorial circuit that can perform a large range of arithmetic and logic operations. Since the data is 8-bit long, we use an 8-bit ALU. The control input value decides the function ALU executes at a particular time. ALU can accept up to two operands at a time, one from *ACC* and the other from *MDR*. The ALU output is stored in *ACC*. If addition operation generates a carry output from ALU, that can be stored in a flip-flop, often called carry flag (*CY*). Since, in our problem numbers are small in

magnitude the 8-bit ALU doesn't generate carry output and we don't need *CY* flag for our simple computer. Note that, ALU cannot perform multiply and division operation for which we use special hardware or some indirect technique.

Instruction Decoder (ID) is a 3-to-8 decoder, which takes input from *IR* and thereby decodes the opcode. In our simple computer there are 8 different opcodes, each one making one of the decoder output (D_0, D_1,..., D_7) high. This in turn initiates specific micro operations necessary to execute that opcode in subsequent clock cycles.

Timing Sequence Decoder (TSD) is again a 3-to-8 decoder that takes input from *TC* and provides necessary timing information in the form of decoded output (T_0, T_1,..., T_7) for a micro operation to be executed.

BUS is a group of wires that serve as a shared common path for data transfer of all the devices connected to it. With this, we do not need a separate device to device connection which increases the number of wiring specially when large number of devices are used in a system. Since, the largest group of binary data that is transferred in our computer is 8-bit, the bus used is an 8-bit bus.

BUS Selector (BS) is a multiplexer, which decides which one of all the connected devices is in transmission mode, i.e. has placed data in the BUS. Note that, if more than one device try to send data simultaneously, there will be a conflict producing erroneous result. However, in our computer we may allow more than one device connected to BUS to receive data from BUS. We'll see shortly that only *PC*, *ACC*, *MDR* and ALU want to transmit or place data on the BUS. *MAR* and *IR* only receive data and other hardware give control signal and don't do data transfer. Thus, BS has to select one of the four devices and uses eight (each one for one bit) 4-to-1 multiplexer type of device. We can also use tri-stated output for bus connection (Section 14.6 of Chapter 14) that will reduce the current loading on the device when it is not selected.

From this discussion we can draw the *data path* of our simple computer as shown in Fig. 16.2. Here, by data we mean address, opcode as well as operand and they move from/to memory, register, ALU, etc. Of course, we need another set of path to send control signal to various hardware to carry out microoperations. This is called *control path* and we'll design it when we define the instruction set for our computer.

Generally speaking, *address bus* is the group of wires that transfer address information, *data bus* is another group that transfers data and *control bus* transfers control information. Often, address information and data are transferred through a common bus and a control logic decides which is to be transferred and when. You might have noticed that in our simple computer design, we have used a common address and data bus. More about control bus will appear in Section 16.4 where we discuss the design of control unit.

Find in Fig. 16.2 the direction of arrow that shows the direction of data flow. Note that, *IR* and *MAR* can only receive data from BUS; *PC* can only send data by BUS; *ACC* and *MDR* can do both;

Memory data transfer takes place only via *MDR* and operands of ALU come from *ACC* and *MDR* and result is sent via BUS.

Example 16.1 In a particular configuration each memory location contains 16-bit data. In program memory, if 4 MSB contains opcode and rest contains address of memory locations give (a) Number of opcodes (b) Size of memory (c) Size of *PC*, *IR*, *ACC*, *MAR* and *MDR*.

Solution

(a) Number of opcodes = 2^4 = 16 (Maximum)

(b) Number of address bits = 16 – 4 = 12. No. of memory locations = $2^{12} = 2^2 \cdot 2^{10}$ = 4K. So size of memory is 4K × 16.

(c) Size of *PC* and *MAR* = No. of address bits = 12. Size of *IR* = Size of opcode = 4. Size of *ACC* and *MDR* = No. of data bits = 16

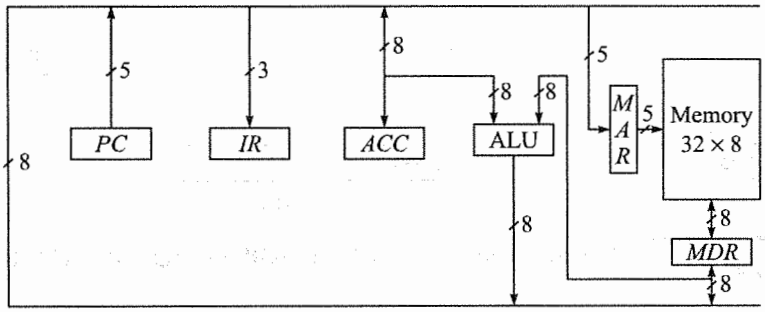


Fig. 16.2 **Data path of the simple computer**

SELF-TEST

1. What is the highest integer in decimal that we can store in 16-bit data field?
2. What is an opcode?
3. What is the function of program counter?
4. What is indirect addressing?

16.2 REGISTER TRANSFER LANGUAGE

Before we go for design of control path and the control unit as a whole we have to define macro operations and then we need to break up each macro operation in series of micro operations at register level. Register Transfer Language (RTL) gives a simple tool through which these micro operations can be expressed and then control unit can be designed from that. The basic structure of this language is

$$X : A \leftarrow B$$

This means, if condition X is TRUE, i.e. $X = 1$ then content of register B is transferred to register A. X can be a single logic variable or a logic expression like $xy \equiv x\&y$, $x + y \equiv x \mid y$, etc. In RTL we distinguish logic operation 'OR' from arithmetic operation 'addition' by assigning symbols ' | ' and '+' respectively. The logic AND is expressed by symbol '&'. However, if the '+' sign appears left to ':' in an RTL statement it means logical OR and '.' refers to logical AND. This is so because to left of ':' only logical operators can reside. Often AND, OR, NOT are expressed by '$\wedge$', '$\vee$', '~' respectively. Also note, this register transfer destroys the previous content of A but not that of B. Both the register A and B now have the same value. If register transfer takes via BUS

$$A \leftarrow B \equiv BUS \leftarrow B, \quad A \leftarrow BUS$$

Since, *BUS* is not a register but a group of wire this means B getting access to *BUS* through BUS selector (BS) and the whole event takes place in one clock cycle. Figure 16.3 pictorially depicts register transfer without and with BUS.

To write anything to memory, in our simple computer we have to place the address information in *MAR* and the data to be written in *MDR*. Thus, memory write operation in RTL is expressed as

$$X : \mathrm{M}[MAR] \leftarrow MDR$$

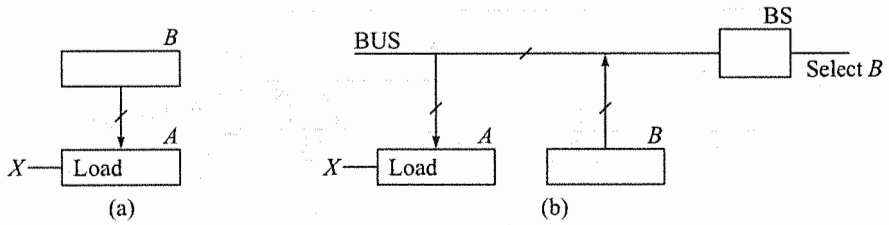


Fig. 16.3 **Register transfer A → B: (a) without BUS, (b) with BUS**

Similarly, memory read operation is also done through *MAR* and *MDR* and RTL expression is

$$X: MDR \leftarrow \mathrm{M}[MAR]$$

If certain bits of a register are to be addressed we use RTL as follows:

$$X: IR \leftarrow MDR[7{:}5]$$

The statement above refers to transfer of three most significant bits of *MDR* to *IR*, a 3-bit register when $X = 1$.

The arithmetic and logic operations of ALU that bring operands from *ACC* and *MDR* and store the result in *ACC* can be expressed in RTL in the following way

$X: ACC \leftarrow ACC \,\&\, MDR$	[logic AND]
$X: ACC \leftarrow ACC \mid MDR$	[logic OR]
$X: ACC \leftarrow ACC \oplus MDR$	[logic EX-OR]
$X: ACC \leftarrow ACC'$	[logic NOT]
$X: ACC \leftarrow ACC + MDR$	[arithmetic addition]
$X: ACC \leftarrow ACC - MDR$	[arithmetic subtraction]
$X: ACC \leftarrow ACC + 1$	[increment by 1]
etc.	

And finally if data is to be shifted in a register say by 1 bit to left we can write

$$X: ACC[7{:}1] \leftarrow ACC[6{:}0], ACC[0] \leftarrow 0$$

If such left shift occurs through carry the statement will be

$$X: ACC[7{:}1] \leftarrow ACC[6{:}0], ACC[0] \leftarrow CY$$

Normally, we come across these four kinds of micro operations namely (i) Inter-Register transfer (ii) Arithmetic operation (iii) Logic operation and (iv) Shift operation. Note that, left shift without carry can also be obtained by addition operation as shown in Example 6.14 of Chapter 6. Figure 16.4 shows how addition operation $A \leftarrow A + B$ takes place through ALU and BUS.

Note that, *TC* and *PC* can increment by 1 without taking help of ALU as they are designed to be parallel load up counters. For more complex processor unit where 2 byte, 3 byte instructions are possible we can have an adder unit accessible by *PC*.

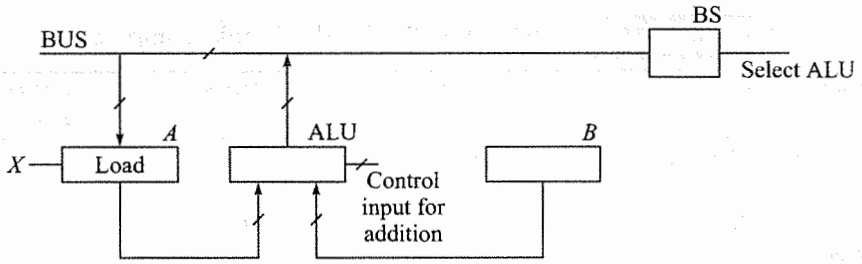


Fig. 16.4 **The micro operation $A \rightarrow A + B$**

Example 16.2 Explain what the following RTL statements perform

$$T_1 : MDR \leftarrow ACC$$
$$T_2 : ACC \leftarrow ACC'$$
$$T_3 : ACC \leftarrow ACC \ \& \ MDR$$

Solution The first statement says if $T_1 = 1$, content of *ACC* is transferred to *MDR*. The second statement says if $T_2 = 1$, content of *ACC* is complemented. The third statement says if $T_3 = 1$, bit-wise AND operation is performed on *ACC* and *MDR* and the result is stored in *ACC*. Since content of *MDR* and *ACC* were complement of one another before this statement is executed, by AND operation all the bits of *ACC* become zero, i.e. *ACC* is reset by these three statements irrespective of its initial content.

Note that, T_1, T_2 and T_3 can be output of a timing sequencer, which become active one after another in consecutive clock cycles. This way, *ACC* can be cleared in three clock cycles by above RTL statements.

SELF-TEST

5. What is RTL?
6. What is to be changed in Fig. 16.4 to perform $A \leftarrow A \ \& \ B$?

16.3 EXECUTION OF INSTRUCTIONS, MACRO AND MICRO OPERATIONS

In a computer, *execution of instructions* is carried through macro operations which again can be subdivided into micro operations. In this section, we first define the macro operations that we want to be executed in the computer we are designing. Next, we'll discuss micro operations necessary to execute each macro operation and it will be expressed through RTL. Remember that we have assigned only 3-bits as opcode and hence we can define $2^3 = 8$ instructions or macro operations with them. Table 16.1 lists all the instructions, corresponding mnemonics (easy to remember short forms), opcodes and 3-to-8 decoder (ID) output when *IR* is loaded with this opcode.

Table 16.1 Instruction Set for the Simple Computer

Macro operation performed	*Instruction mnemonic*	*Opcode*	*Instruction decoder (ID) output activated*
Load data from a specified memory location to *ACC*	LDA	0 0 0	D_0
Store *ACC* data in a specified memory location	STA	0 0 1	D_1
Halts execution of the program	HLT	0 1 0	D_2
Perform bitwise AND operation of *ACC* with data of a specified memory location and store result in *ACC*	AND	0 1 1	D_3
Perform bitwise NOT operation of *ACC*	NOT	1 0 0	D_4
Perform 1-bit left shift of *ACC* with $ACC[0] \leftarrow 0$	SHL	1 0 1	D_5
Perform addition operation of *ACC* with data of a specified memory location and store result in *ACC*	ADD	1 1 0	D_6
Subtract from *ACC*, data of a specified memory location and store result in *ACC*	SUB	1 1 1	D_7

Instruction Cycles

To carry out each instruction or macro operation the computer has to go through three distinct phases or cycles. In fetch cycle it brings the instruction or opcode from the program memory. In decoding phase it decodes the opcode and finally the execution is done in execute cycle. These cycles together known as *instruction cycle* are again repeated for next instruction. It is understandable that fetch and decode phase will be same for all instructions in our simple computer as we have only single byte directly addressed instructions. However, the execution cycle will be different for different instructions depending on the tasks the instruction wants to perform.

Fetch Cycle

An instruction cycle begins with *fetch cycle* when *TC* is reset to 0. Then, only T_0 output of TSD will be high and rest low. As told before *PC* contains the address of the location from which next instruction is to be fetched, content of *PC* is loaded into *MAR* in T_0.

At the next trigger of master clock *TC* is incremented by 1 so that T_1 becomes high and other outputs of TSD are low. In this clock cycle, content of memory from location specified by *MAR* (through 5-to-32 address decoder attached to memory) is loaded to *MDR*. *PC* now can be incremented to point to address of next location in program memory, which stores next instruction.

In the next clock cycle *TC* generates $T_2 = 1$ when opcode from 3 MSB of *MDR* is transferred to *IR* and 5 LSB to *MAR*. Content of *IR* is used for decoding opcode in decode phase. Content of *MAR* will be useful in execute phase if the opcode makes some memory reference, the address of which remain available at *MAR*. In RTL the above operations can be represented as

$$T_0 : MAR \leftarrow PC$$
$$T_1 : MDR \leftarrow \mathrm{M}[MAR],\ PC \leftarrow PC+1$$
$$T_2 : IR \leftarrow MDR[7{:}5],\ MAR \leftarrow MDR[4{:}0]$$

Decode Cycle

In *decode cycle* we decode the opcode fetched from program memory. Since at T_2, register *IR* is loaded with opcode and 3-to-8 decoder (ID) that decodes the opcode is a combinatorial circuit, we finish decoding in T_2 itself. In RTL we express it as

$$T_2 : D_0 \ldots D_7 \leftarrow \mathrm{DECODE}\ (IR)$$

Often, the 3rd statement of previously mentioned fetch cycle that loads *IR* with new opcode is considered a part of decode cycle or fetch-decode together is called fetch cycle.

Execute Cycle

Micro operations for each instruction are different and we list them first and then give the explanation.

LDA	$D_0T_3 : MDR \leftarrow \mathrm{M}[MAR]$
	$D_0T_4 : ACC \leftarrow MDR,\ TC \leftarrow 0$
STA	$D_1T_3 : MDR \leftarrow ACC$
	$D_1T_4 : \mathrm{M}[MAR] \leftarrow MDR,\ TC \leftarrow 0$
HLT	$D_2T_3 : S \leftarrow 1,\ TC \leftarrow 0$
AND	$D_3T_3 : MDR \leftarrow \mathrm{M}[MAR]$
	$D_3T_4 : ACC \leftarrow ACC\ \&\ MDR\ \ TC \leftarrow 0$
NOT	$D_4T_3 : ACC \leftarrow ACC',\ TC \leftarrow 0$
SHL	$D_5T_3 : ACC[7{:}1] \leftarrow ACC[6{:}0], ACC[0] \leftarrow 0, TC \leftarrow 0$
ADD	$D_6T_3 : MDR \leftarrow \mathrm{M}[MAR]$
	$D_6T_4 : ACC \leftarrow ACC + MDR,\ TC \leftarrow 0$
SUB	$D_7T_3 : MDR \leftarrow \mathrm{M}[MAR]$
	$D_7T_4 : ACC \leftarrow ACC - MDR, TC \leftarrow 0$

A quick overview of the above list shows, at the completion of each instruction cycle (fetch-decode-execute) *TC* is reset by which the computer goes to T_0 state and fetch cycle for next instruction begins. Note that, a detailed discussion on execution of the program at register level for every clock trigger appears in Section 16.5.

In operations like LDA, AND, ADD, SUB data is brought from memory, address of which is available in *MAR*. In executing STA the *MAR* content denotes the location where data is to be stored in memory.

Macro operations AND, NOT, ADD, SUB use ALU. When HLT is executed *S* flag is set which stops execution of the program. This flag is cleared through power-on-reset.

Now let's pick up one macro operation (say, LDA) and see how it gets executed through its constituent micro operations. From Table 16.1 we find instruction LDA transfers content of a specified memory location to *ACC*. If the opcode fetched in fetch cycle is 000 it refers to LDA operation. In decode phase, opcode 000 makes $D_0 = 1$ and the other outputs of ID are all zero. This is so till *IR* is refreshed or receives another opcode in the next fetch cycle, state T_2. Till then D_0 gives output 1.

The computer enters execution phase at state T_3 ($T_3 = 1$). Now as $D_0 = 1$, condition $D_0T_3 = 1$ and data is read from memory and loaded in *MDR*. Note that, the address of memory location from which data is to be brought was made available to *MAR* in state T_2. Also note that, memory content cannot directly be loaded into *ACC* (refer to data path shown in Fig. 16.8) and is to be done through *MDR*. In next clock cycle, i.e. when $D_0T_4 = 1$ the content of *MDR* is transferred to *ACC* via BUS and the macro operation is complete. We reset *TC* and let the computer begin a new instruction cycle. This analysis can be extended to explain execution of other instructions.

At this point we make an important observation that all the instruction executions are completed within 5 clock cycles (T_0 to T_4) and hence a 3-bit counter, which can count up to 8 is sufficient as *TC* in our simple computer.

SELF-TEST

7. What is a fetch cycle?
8. Why *TC* is reset every time an instruction is executed?

16.4 DESIGN OF CONTROL UNIT

The control unit is primarily a combinatorial circuit that supplies necessary controls inputs to all the important hardware elements of the computer. This takes timing information from computer master clock and is thus responsible for providing necessary *timing and control* information. The path through which these signals travel to reach different parts of a computer is called *control path*. Often we assign a group of wires, called *control bus* as shared path for this. The control logic is arrived at from (i) basic computer architecture we have adopted in the beginning, (ii) conditions appearing at left hand side of symbol ':' in RTL statements for our simple computer, given in previous section, and (iii) certain other issues, e.g. power-on-reset, control variables need to be activated for intended operation of a particular hardware, etc.

Loading Registers

Let us first see when parallel load control of *IR* is to be activated. We find from discussion of previous section, only during T_2 it is loaded. So TSD (Timing counter decoder) output T_2 can be directly connected as parallel load control input of *IR*. Every time T_2 is active this loads three MSBs of BUS (data path is such, refer to Fig. 16.2), which at that time holds *MDR* value, into *IR*. Obviously, at that time BUS selector (BS) should place content of *MDR* into BUS. This we'll discuss while designing control for BS.

What happens if we allow loading of *IR* say, in every clock cycle instead of above? Whenever there is some data made available in BUS by any hardware 3 MSB of that will be loaded into *IR*; ID (decoder) will immediately change and execution corresponding to a different opcode, not the intended one, may begin. You

can understand it'll be all chaos without any sense. Thus we return sanity to our simple machine by loading *IR* only when opcode is fetched, i.e. in T_2 and we can write logic relation

$$\text{LOAD}_{IR} = T_2$$

We see, *MDR* is loaded during T_1, D_0T_3, D_1T_3, D_3T_3, D_6T_3, D_7T_3 and corresponding condition is

$$\text{LOAD}_{MDR} = T_1 + (D_0 + D_1 + D_3 + D_6 + D_7)T_3$$

Proceeding in same manner we can write, $\text{LOAD}_{MAR} = T_0 + T_2$ and

$$\text{LOAD}_{ACC} = D_4T_3 + (D_0 + D_3 + D_6 + D_7)T_4$$

Memory Read/Write

Memory read signal is invoked by: $\text{READ}_{\text{M}} = T_1 + (D_0 + D_3 + D_6 + D_7)T_3$

Memory write signal is invoked by: $\text{WRITE}_{\text{M}} = D_1T_4$

ALU Control

Control variables of ALU activated for addition: $\text{ALU}_{\text{ADD}} = D_6T_4$

Control variables of ALU activated for subtraction: $\text{ALU}_{\text{SUB}} = D_7T_4$

Control variables of ALU activated for logic AND: $\text{ALU}_{\text{AND}} = D_3T_4$

Control variables of ALU activated for logic NOT: $\text{ALU}_{\text{NOT}} = D_4T_3$

BUS Controller

BUS controller gives access to *ACC* by $\text{BUS}_{ACC} = D_1T_3$,

to *PC* by $\text{BUS}_{PC} = T_0$,

to *MDR* by $\text{BUS}_{MDR} = T_2 + D_0T_4$

and to ALU by $\text{BUS}_{\text{ALU}} = D_4T_3 + (D_3 + D_6 + D_7)T_4$

Thus, selection inputs of eight 4-to-1 multiplexers that places data from one of these four devices *ACC*, *PC*, *MDR* and *ALU* on BUS should become active when corresponding conditions mentioned by above logic equations are met.

Other Control Signal

The condition for setting START/STOP flag *S* is: $\text{SET}_S = D_2T_3$ [*S* is power on reset]

The condition for shift left operation of *ACC* is: $\text{SHIFT\_LEFT}_{ACC} = D_5T_3$

The signal that triggers increment of *PC*: $\text{INCREMENT}_{PC} = T_1$

Timing counter *TC* is synchronously reset by: $\text{RESET}_{TC} = (D_2 + D_4 + D_5)\,T_3 + (D_0 + D_1 + D_3 + D_6 + D_7)\,T_4$

Finally, the master clock remains enabled if flag *S* is not set. Thus $\text{ENABLE}_{CLOCK} = S'$

Based on these equations the control unit of our simple computer can be made. We show the control circuit of *ACC*, *TC* and *TSD*, BS in following three examples. Refer to problems of Section 4 of this chapter for more circuits. Together they make the control unit of our simple computer.

Example 16.3 Show using circuit diagrams the control inputs to *ACC*.

Solution The parallel load shift register *ACC* in the simple computer designed shifts data to left while serial data in is 0 (GND). It also loads parallel data from BUS. The conditions for these two operations are shown above in the form of logic equations. The corresponding diagram is shown in Fig. 16.5.

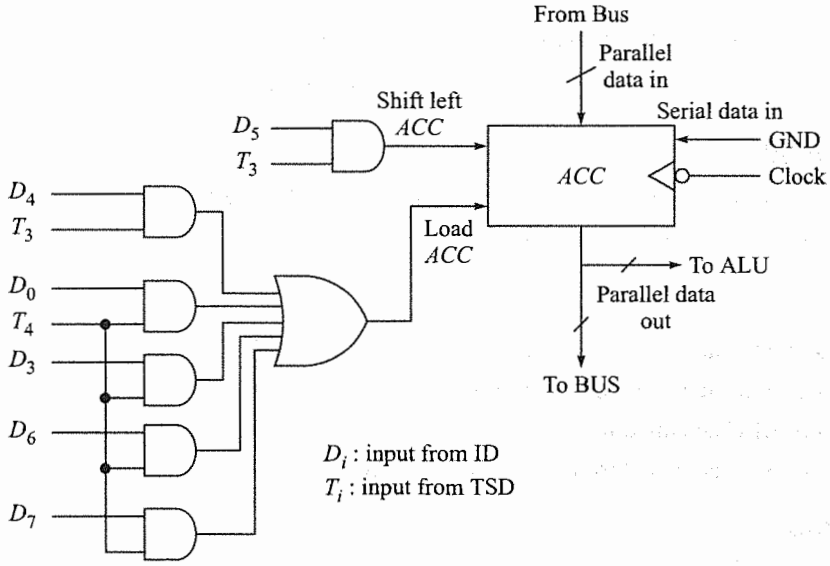


Fig. 16.5 Control for ACC

Example 16.4 Show using diagrams control inputs to *TC* and its connection to TSD.

Solution The timing counter *TC* is a mod-8 up counter with parallel load facility. When $RESET_{TC}$ is activated according to control logic discussed in this section, 000 is synchronously loaded and up count resumes. The required circuit diagram is shown in Fig. 16.6.

Example 16.5 Show using diagram how bus controller works.

Solution The controller developed from control equations discussed in this section is shown in Fig. 16.7. This is developed on multiplexer logic like Fig. 4.5 of Chapter 5. A tri-state bus control can also be designed similar to diagram shown in Fig. 14.26 of Chapter 14. There the *DISABLE* control input will be fed by complement of respective BUS activate signal, i.e. complement of BUS_{ACC}, BUS_{PC} etc.

Note that, *PC* does not access bit 5 to 7 of BUS as it has only 5 bit binary information that is transferred to *MAR* via bit 0 to 4 of the BUS. Hence, for 3 MSB there is one AND gate less and the OR gate is of 3 input.

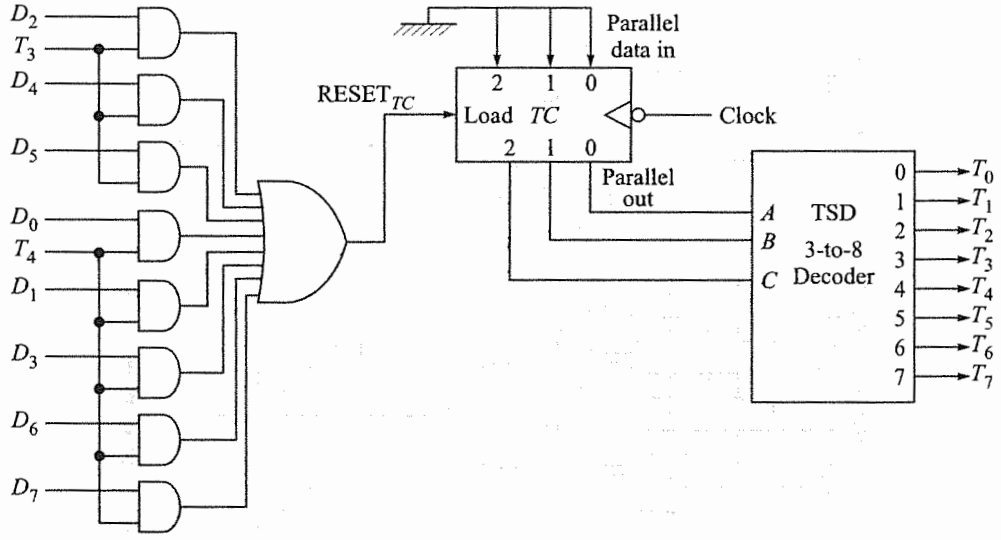


Fig. 16.6 **Control of Timing Counter, TC and its connection with Timing Sequence Detector, TSD**

SELF-TEST

9. How long instruction decoder outputs $D_7 \ldots D_0$ remain constant?
10. What are the instructions of this simple computer in which ALU places data on BUS?
11. Which instruction sets flag *F*?

16.5 PROGRAMMING COMPUTER

Now that our simple computer is ready with hardware and instruction sets let us see what computer program can solve the problem with which we started designing our simple machine. In Table 16.2 we present the program in mnemonics along with comments on job done by each instruction. Program in binary code as exists in 32 × 8 memory module will be shown after that.

Thus we need 14 instructions (Table 16.2), all single byte to solve the problem in our simple computer. We need 12 memory locations for storing numbers. So 14 + 12 = 26 bytes of our 32 byte memory are used for this problem. For bigger sized problems we need bigger memory and for more complex problem additional instruction sets and, of course, more complex computer architecture.

Now let us see how program and data remain stored in memory in binary numbers. We know that due to power-on-reset *PC* is always initialized with 00000, the first location of the memory (Refer Table 16.3) where first instruction of the program is to be stored. We use first 14 locations (address 00000 to 01101) of memory to store instructions. If we store data used in the program, i.e. 11 numbers in next consecutive locations then addresses 01110 to 11000 get filled. The location 11001, i.e. 26th location of memory can be used to store the result. Note that, multiplication is achieved by left shifting *ACC* and thus we don't need to store any multiplicand for that. If the 10 numbers to be added are say, 5, 2, 1, 3, 8, 6, 5, 2, 7, 4 and the number

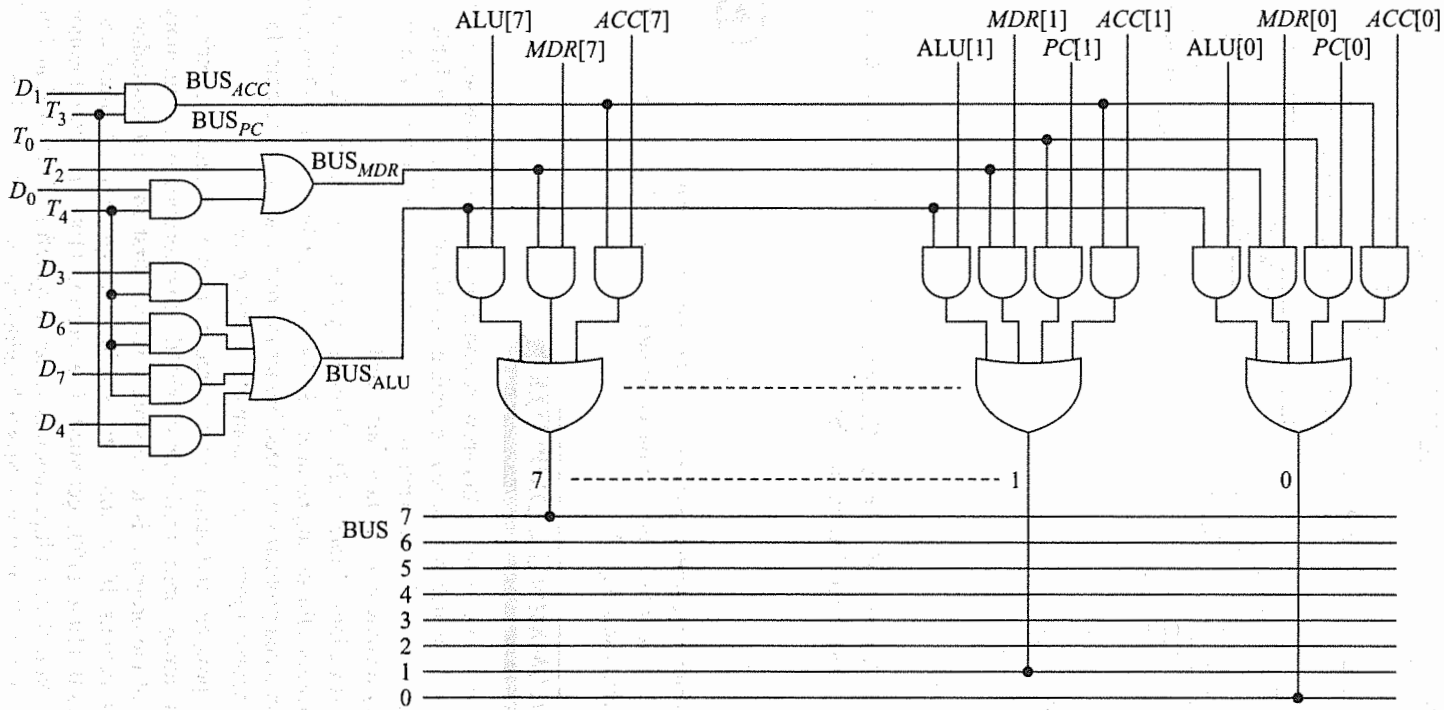


Fig. 16.7 Control over bus by different devices using multiplexer logic

Table 16.2 Program to Solve Given Problem with Comments

Instruction Number	*Instruction pnemonic*	*Comment*
1	LDA *addr*1	loads 1st number to *ACC*, the address of which follows opcode
2	ADD *addr*2	fetches 2nd number from memory and adds to 1st, stores the sum in *ACC*
3	ADD *addr*3	similarly adds 3rd number
4	ADD *addr*4	adds 4th number
5	ADD *addr*5	adds 5th number
6	ADD *addr*6	adds 6th number
7	ADD *addr*7	adds 7th number
8	ADD *addr*8	adds 8th number
9	ADD *addr*9	adds 9th number
10	ADD *addr*10	adds 10th number, now sum of 10 numbers remain available in *ACC*
11	SUB *addr*11	fetches 11th no. from memory, subtracts it from sum of 10 nos., stores result in *ACC*
12	SHL	shifts *ACC* to left by 1 bit, equivalent to multiplication by 2
13	STA *addr*12	stores content of *ACC* in memory in the address available after opcode
14	HTL	halts the computer

subtracted is say, 9 then we can fill up first 25 locations of memory as shown in Fig. 16.14. The 26th location, before the program is run, may contain anything but after the program is run will contain the end result, i.e. 68 expressed in binary. Memory content is often shown in hexadecimal instead of binary. Refer to Problems 16.19 and 16.20 for this.

Program Execution

Now let us see how the program gets executed in first few instruction cycles. We note the change in the value of the registers along with ID and TSD in each clock cycle since the program begins. Table 16.4 shows sequential progress of our simple computer with every trigger of system clock. As told before *PC*, *TC* and *S* are power on reset. They all contain zero in the beginning when the computer is switched on.

In first clock cycle, the machine is in T_0 state given by TSD that decodes *TC*. At T_0, content of *PC* that contains the starting address of the program is copied to *MAR*. Corresponding micro operation is shown in rightmost column of Table 16.4. *TC* is incremented by 1.

In next clock cycle, *TC* and *PC* are incremented by 1, data from memory is loaded to *MDR* that contains the first instruction.

In 3rd clock cycle *TC* is incremented, *IR* gets the opcode and *MAR* gets address for first data. Note that decoding of *IR* is also done in same clock cycle that makes D_0 high, as opcode is 000 (LDA). This completes the fetch cycle, which is common for all instructions.

In executing LDA instruction the first state is T_3 state. Here, data from 15th location of Memory (*MAR* = 01110) which contains 00000101, decimal equivalent of 5 is loaded to *MDR*. In T_4 state this data is transferred to *ACC* and macro operation LDA is fully executed. This completes the first instruction cycle. Note that timing counter (*TC*) is to be reset after execution of data transfer from *MDR* to *ACC* and that begins the next instruction fetch.

Table 16.3 Program and Data Section of the Memory

Memory location number	*Memory address in binary*	*Memory content*	*Comment*
1	00000	00001110	Program section begins. Loads 1st no. from location 01110 to *ACC*. 3MSB 000: Load
2	00001	11001111	3MSB110: ADD, 5LSB 01111: Address of 2nd operand
3	00010	11010000	.
4	00011	11010001	.
5	00100	11010010	.
6	00101	11010011	First 14 locations, i.e. memory address 00000 to 01101 contain instructions.
7	00110	11010100	Here, three MSBs always refer to opcode. Five LSBs refer to memory
8	00111	11010101	address for instructions LDA, ADD, SUB, STA. For instructions
9	01000	11010110	SHL and HLT, five LSBs can be anything as they are not referred
10	01001	11010111	anywhere.
11	01010	11111000	.
12	01011	10100000	.
13	01100	00111001	.
14	01101	01000000	Halts computer. Program section ends.
15	01110	00000101	The data section starts. Stores 1st number, 5 expressed in binary
16	01111	00000010	2nd no. 2 in binary
17	10000	00000001	.
18	10001	00000011	.
19	10010	00001000	.
20	10011	00000110	.
21	10100	00000101	.
22	10101	00000010	.
23	10110	00000111	.
24	10111	00000100	.
25	11000	00001001	Stores 11th number, 9 that is subtracted from the sum of 10 nos.
26	11001	xxxxxxxx	After the program is run it becomes 01000100, i.e. 68 in decimal.
27	11010	xxxxxxxx	UNUSED
28	11011	xxxxxxxx	UNUSED
29	11100	xxxxxxxx	UNUSED
30	11101	xxxxxxxx	UNUSED
31	11110	xxxxxxxx	UNUSED
32	11111	xxxxxxxx	UNUSED

The fetch cycle is repeated in clock cycle 6 to 8. Since the instruction fetched is ADD (opcode 110) corresponding micro operations are performed in clock cycles 9 and 10 followed by next instruction fetch, starting again at 11th clock cycle. This continues till we reach 14th instruction HLT which when executed, sets *S* flag. This inhibits the system clock output in our design; thus content of all registers and memory will remain unchanged after that till the computer is switched off.

Table 16.4 Execution of the Program at Register Level

Clock Cycle	*TC$*	*TSD*	*PC$*	*MAR*	*MDR*	*IR*	*ID*	*ACC*	*S*	*Micro operation performed after clock trigger*
1	000	T_0	00000	00000	xxxxxxxx	xxx	x	xxxxxxxx	0	*MAR* ← *PC*
2	001	T_1	00000	00000	00001110	xxx	x	xxxxxxxx	0	*MDR* ← M[*MAR*], *PC*←*PC*+1
3	010	T_2	00001	01110	00001110	000	D_0	xxxxxxxx	0	*IR*←*MDR*[7:5], *MAR*←*MDR*[4:0]
4	011	T_3	00001	01110	00000101	000	D_0	xxxxxxxx	0	*MDR* ← M[*MAR*]
5	100	T_4	00001	01110	00000101	000	D_0	00000101	0	*ACC* ← *MDR*, *TC* ← 0
6	000	T_0	00001	00001	00000101	000	D_0	00000101	0	*MAR* ← *PC*
7	001	T_1	00001	00001	11001111	000	D_0	00000101	0	*MDR* ← M[*MAR*], *PC*←*PC*+1
8	010	T_2	00010	01111	11001111	110	D_6	00000101	0	*IR* ← *MDR*[7:5], *MAR* ←*MDR*[4:0]
9	011	T_3	00010	01111	00000010	110	D_6	00000101	0	*MDR* ← M[*MAR*]
10	100	T_4	00010	01111	00000010	110	D_6	00000111	0	*ACC* ← *ACC* + *MDR*, *TC*←0
11	000	T_0	00010	00010	00000010	110	D_6	00000111	0	*MAR* ← *PC*
12	001	T_1	00010	00010	11010000	110	D_6	00000111	0	*MDR* ← M[*MAR*], *PC*←*PC*+1
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...

ǂ*PC*, *TC* (also *TSD*) values shown are the ones before clock trigger while for other registers this is what appears after clock trigger (taking care from *TC* and *PC*).

Concluding Remark

Before we conclude our computer design exercise let us see what we have achieved and what more is needed to make this computer fully functional. We have designed a simple processor comprising register arrays, flag, small memory, BUS and control unit. In short, we have designed a *central processing unit* (CPU) that connects to a small memory module and is able to execute programs built on a small instruction set.

What we have not discussed is how data is entered into computer from an external device say, keypad and also how it displays data in some output device say, a monitor. We also have not discussed interesting and important issues like

(i) handling of jump instructions (CPU jumps to an address to fetch an instruction),
(ii) use of subroutines (program under program that is used and called many times),
(iii) memory management (slow-fast, addressing mode details),
(iv) interrupt handling (devices of different priorities asking attention and service of CPU),
(v) pipelined CPU (doing jobs in parallel if there is no conflict to enhance computer speed, e.g. execution phase of present instruction in parallel with fetch of next instruction), so on and so forth. These are covered in detail in titles related to modern computer design courses and an interested reader can refer to the same.

We shall conclude this chapter by revisiting *computer architecture* introduced in Section 1.6, carrying forward the discussions of this chapter. Figure 16.8 represents a basic 8-bit computer. The 8-bit data bus is bidirectional in nature i.e. CPU is capable of both reading and writing from/to a location defined by 16-bit address which is a total of $2^{16} = 64$K locations. Individual RAM and ROM are of size 16K and this requires 14 bits for addressing. The two MSBs are sent to a 2 to 4 address decoder which generates four Chip Select (CS)

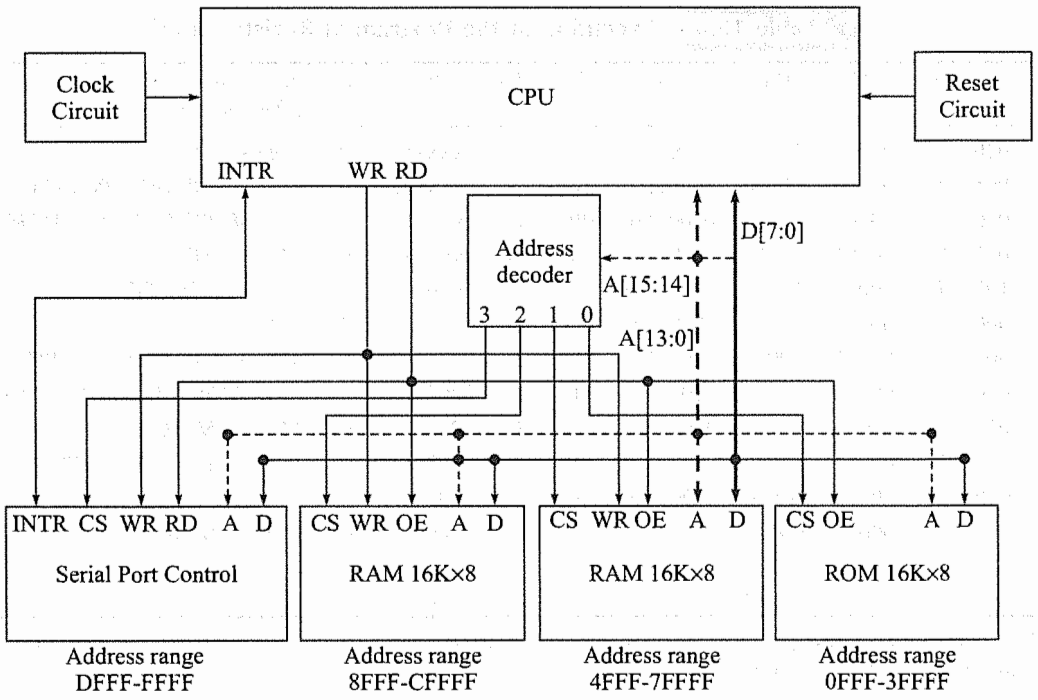


Fig. 16.8 **Basic architecture of an 8-bit computer**

signals, connected to each of the memory and output module generating unique address ranges as specified in the bottom of the figure. The calculation is as follows. For the ROM, A[15:14] is always 00 and thus all possible values of A[13:0] generate address ranges 0000 0000 0000 0000 to 0011 1111 1111 1111, i.e. 0000 to 3FFF in hex. Similarly, for the first RAM block, A[15:14] is always 01 and thus all possible values of A[13:0] generate address ranges 0100 0000 0000 0000 to 0111 1111 1111 1111, i.e. 4000 to 7FFF etc. CPU read is enabled by activating the control signal, RD (Read). This, in turn, requests outputs of the devices from which data is to be read to be enabled through OE (Output Enable) or through RD, if it is a serial input-output port, following which CPU takes the value from data bus. The control signal WR (Write) is activated to enable CPU writing to devices. Note that WR and RD should not be activated simultaneously. The timing of these control signals are also important so that data, chip select and address are properly stabilized to avoid false reading and writing operations. The ROM is not writable and usually contains sequence of instructions required for booting. This is usually used during *power on* of the computer and also in between, if the computer is asked to stop all operations and start afresh. The other time a computer may be asked to stop its usual fetch-decode-execute operations, but only temporarily, is when an *interrupt* is invoked. Then the computer's present state is stored in a designated memory space called *stack*. The computer comes back to its usual operating state once the interrupt is served usually through a *interrupt service routine* (ISR). There could be both software and hardware interrupts. The serial port control block shows how a hardware interrupt can ask service from CPU by activating INTR. Note that the *maskable interrupts* can be masked (disabled) by writing into a control register while *non-maskable interrupts* cannot be disabled. Reset is a non-maskable interrupt and care should be taken in the design of a computer so that corresponding ISR is in place before an interrupt is invoked.

Example 16.6 How many clock cycles are needed to execute the program shown in Table 16.2?

Solution The calculation of clock cycles is as follows.

One LDA	:	5
Nine ADD	:	$9 \times 5 = 45$
One SUB	:	5
One STA	:	5
One SHL	:	4
One HLT	:	4
TOTAL =		68 clock cycles

Example 16.7 Write a program for this computer that adds two positive integers, available in memory locations *addr*1 and *addr*2, multiplies the sum by 5 and stores final result in location *addr*3. Consider, the numbers are small enough not to cause any overflow, i.e. data at every stage require less than 8-bits for its representation.

Solution Addition of two numbers is straightforward and can be done using LDA and ADD instructions as done before. For multiplication with 5 we have to use an indirect technique. Two left shift give multiplication by 4 and one more addition will make it multiplication by 5. Alternatively, 5 ADD operations will also give multiplication by 5. The program can be written as follows.

```
LDA addr1
ADD addr2
STA addr3
SHL
SHL
ADD addr3
STA addr3
HLT
```

Note that, we have used *addr*3 as intermediate storage of addition result. Since in the computer designed there is no instruction to place data on a register from *ACC* (and also retrieve the same) we had to use memory. Storing intermediate results in registers speeds up the process but here we are limited by the architecture and instruction set available. Also note, we could have used any other available memory location for intermediate storage.

Example 16.8 Write a program for this computer that performs bit-wise Ex-OR operation on two numbers available in memory locations *addr*1 and *addr*2. The result is to be stored in location *addr*3.

Solution Our designed computer can perform only two kinds of logic operations AND and NOT. Therefore, we break Ex-OR logic of two numbers, say A and B in such a way that there is only AND and NOT operator.

$$Y = A \oplus B = AB' + A'B = ((AB')'.(A'B)')' \quad \text{[From DeMorgan's Theorem]}$$

Thus the program can be written as shown next. The logic operation performed by each instruction is shown as comment after semicolon.

```
LDA     addr1       ; A
NOT                 ; A'
AND     addr2       ; A'B
NOT                 ; (A'B)'
STA     addr3
LDA     addr2       ; B
NOT                 ; B'
AND     addr1       ; AB'
NOT                 ; (AB')'
AND     addr3       ; (AB')'.(A'B)'
NOT                 ; ((AB')'.(A'B)')'
STA     addr3
HLT
```

SELF-TEST

12. What happens to the program shown in Table 16.2 or Table 16.3 if HLT instruction is not provided?

SUMMARY

A computer stores program or binary coded instructions in its program memory. The centrai processing unit comprising set of registers and a control unit sequentially fetches this program, decodes it and executes the same. To accomplish this, an instruction, also called macro operation is broken into series of micro operations. Register Transfer Language is a very convenient tool to express each of these micro operations. A simple computer is designed in this chapter that has eight instructions and can perform logic operations like AND, NOT and arithmetic operations like addition and subtraction. It can also load data from memory and store data in memory. The data path and control unit of the computer is designed using hardware discussed in earlier chapters of the book. The programming technique for this computer for various arithmetic and logic problems is also demonstrated.

GLOSSARY

- ***accumulator*** A multipurpose register that stores one operand of all arithmetic and logic operations and also for memory referenced data transfers.
- ***address bus*** Group of wires that transfer address information.
- ***arithmetic logic unit*** A combinatorial circuit that can perform various types of arithmetic and logic functions decided by a set of selection inputs.
- ***bus*** A group of wire providing shared common path between number of devices.
- ***central processing unit*** The brain of computer that controls the operations of a computer.
- ***computer architecture*** Organization of a digital computer.

- ***control bus*** Group of wires that transfer control information.
- ***control path*** The path through which control signals travel to different devices and make them perform their assigned tasks.
- ***data bus*** Group of wires that transfer data.
- ***data memory*** The part of memory that contains data.
- ***data path*** The path through which data moves from one device to another in a computer.
- ***flag*** A single flip-flop that stores binary outcome of a certain operation.
- ***instruction register*** A register that contains the opcode or binary code of an instruction.
- ***interrupt*** An event that asks computer's immediate attention.
- ***interrupt service routine*** A set of computer instructions that serves an interrupt.
- ***macro operation*** An instruction that a computer executes in a complete instruction cycle, consists of series of micro operations.
- ***maskable interrupt*** Interrupt that can be disabled.
- ***memory address register*** A register that contains address of memory location for all memory referenced instructions.
- ***memory data register*** A register that acts as buffer between memory and rest of the circuit, storing data that moves to and from memory.
- ***micro operation*** The basic operation, a computer performs at register level.
- ***non-maskable interrupt*** Interrupt that cannot be disabled.
- ***opcode*** The binary code of an instruction.
- ***program*** Series of instructions that accomplishes a task in a computer.
- ***program counter*** A register that stores address of next instruction.
- ***program memory*** The part of memory that contains instruction.
- ***register transfer language*** A language, which expresses register transfer and condition for that.
- ***stack*** A memory block usually used for storing a computer's present state when interrupt is invoked.
- ***system clock*** The clock providing basic unit of clock cycle from which trigger of all sequential operations are derived.

PROBLEMS

Section 16.1

16.1 For memory configured as in Fig. 16.2, if immediate addressing is allowed what is the maximum value of number (in decimal) that can be loaded through instruction fetch?

16.2 What is the minimum size required for *MAR* if memory addressed has size 1K × 16?

16.3 For a more complex computer design, 75 different instructions are required. What size of *IR* would you likely choose?

16.4 Draw data path of the computer described next. The computer in addition to what is described in Section 16.2 has two more registers *P* and *Q*, which can transfer data to/from *ACC* via BUS. It also has a *CY* flag that stores ALU overflow and a *Z* flag that is set when all the bits of *ACC* are zero.

Section 16.2

16.5 What does the following statement mean $(X + Y) : A \leftarrow B$

16.6 Explain the meaning of $XY : A \leftarrow B$

16.7 Give the final content of *ACC* when following statements are executed

$$T_1 : ACC \leftarrow ACC \oplus MDR$$
$$T_2 : ACC \leftarrow ACC'$$

16.8 State what the following statement performs for the computer described in problem 16.8

$$T_1 : ACC \leftarrow ACC + MDR$$

$CY \& T_2 : P \leftarrow ACC$

$CY' \& T_2 : Q \leftarrow ACC$

Section 16.3

16.9 Show how shift left operation with carry is executed.

16.10 Show how shift right operation with carry can be executed.

16.11 Consider the first instruction of the simple computer is replaced by MVI that moves immediate data (immediate addressing) to *ACC*. Write micro operations for this instruction. What change in hardware is required for this?

16.12 Consider the first instruction of the simple computer is replaced by LDI that moves indirect data (indirect addressing) to *ACC*. Write micro operations for this instruction. Does it require any change in hardware?

Section 16.4

16.13 What does $\text{LOAD}_{MAR} = T_0 + T_2$ mean?

16.14 Explain if there will be any problem if by mistake the control unit is developed on logic equation $\text{LOAD}_{MAR} = T_0 + T_2 + T_4$.

16.15 Show using diagrams control inputs to ALU. Consider IC 74181 (Section 6.10, Chapter 6) is used as ALU.

16.16 Show using diagrams control inputs to Memory.

Section 16.5

16.17 How many clock cycles are required to execute program given for Example 16.7?

16.18 How many clock cycles are required to execute program given for Example 16.8?

16.19 If the two numbers used in Example 16.7 are 5 and 8, and data section immediately follows program section show the memory values in binary after the program is executed? How will it be represented in hexadecimal?

16.20 If the two numbers used in Example 16.8 are $F2_{16}$ and $D6_{16}$ and data section immediately follows program section show the memory values in binary after the program is executed?

16.21 Write a program that compares two binary data located in *addr*1 and *addr*2 of memory by making all the bits of *addr*3 one if two numbers are exactly equal.

16.22 Write a program that executes following where Data_{addr} refers to data corresponding to address *addr*.

$$\text{Data}_{addr7} = (\text{Data}_{addr1} + \text{Data}_{addr2} + \text{Data}_{addr3} + \text{Data}_{addr4}) \times 3 - (\text{Data}_{addr5} + \text{Data}_{addr6}) \times 2$$

Answers to Self-tests

1. $2^{16} - 1 = 65535$
2. A computer operation coded in a group of binary digits.
3. To store address from which next instruction is fetched.
4. The instruction fetches address of location in which address for operand exists.
5. Register Transfer Language.
6. Control input to ALU.
7. A part of instruction cycle that fetches instruction from memory that after decoding gets executed in execute cycle.
8. Resetting *TC* a new instruction cycle can begin.
9. Till it is loaded in next fetch cycle.
10. ADD, SUB, AND, NOT.
11. HLT.
12. It goes on executing by loading next content of memory which houses first number and since three MSB are 000 opcode decodes it as an LDA instruction.

Appendix 1:
Binary-Hexadecimal-Decimal Equivalents

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
0000 0000	00	0	0
0000 0001	01	256	1
0000 0010	02	512	2
0000 0011	03	768	3
0000 0100	04	1,024	4
0000 0101	05	1,280	5
0000 0110	06	1,536	6
0000 0111	07	1,792	7
0000 1000	08	2,048	8
0000 1001	09	2,304	9
0000 1010	0A	2560	10
0000 1011	0B	2,816	11
0000 1100	0C	3,072	12
0000 1101	0D	3,328	13
0000 1110	0E	3,584	14
0000 1111	0F	3,840	15
0001 0000	10	4,096	16
0001 0001	11	4,352	17
0001 0010	12	4,608	18
0001 0011	13	4,864	19
0001 0100	14	5,120	20
0001 0101	15	5,376	21
0001 0110	16	5,632	22
0001 0111	17	5,888	23
0001 1000	18	6,144	24
0001 1001	19	6,400	25
0001 1010	1A	6,656	26
0001 1011	1B	6,912	27
0001 1100	1C	7,168	28
0001 1101	1D	7,424	29
0001 1110	1E	7,680	30

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
0001 1111	1F	7,936	31
0010 0000	20	8,192	32
0010 0001	21	8,448	33
0010 0010	22	8,704	34
0010 0011	23	8,960	35
0010 0100	24	9,216	36
0010 0101	25	9,472	37
0010 0110	26	9,728	38
0010 0111	27	9,984	39
0010 1000	28	10,240	40
0010 1001	29	10,496	41
0010 1010	2A	10,752	42
0010 1011	2B	11,008	43
0010 1100	2C	11,264	44
0010 1101	2D	11,520	45
0010 1110	2E	11,776	46
0010 1111	2F	12,032	47
0011 0000	30	12,288	48
0011 0001	31	12,544	49
0011 0010	32	12,800	50
0011 0011	33	13,056	51
0011 0100	34	13,312	52
0011 0101	35	13,568	53
0011 0110	36	13,824	54
0011 0111	37	14,080	55
0011 1000	38	14,336	56
0011 1001	39	14,592	57
0011 1010	3A	14,848	58
0011 1011	3B	15,104	59
0011 1100	3C	15,360	60
0011 1101	3D	15,616	61
0011 1110	3E	15,872	62
0011 1111	3F	16,128	63
0100 0000	40	16,384	64
0100 0001	41	16,640	65
0100 0010	42	16,896	66
0100 0011	43	17,152	67
0100 0100	44	17,408	68
0100 0101	45	17,664	69
0100 0110	46	17,920	70
0100 0111	47	18,176	71
0100 1000	48	18,432	72
0100 1001	49	18,688	73
0100 1010	4A	18,944	74
0100 1011	4B	19,200	75

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
0100 1100	4C	19,456	76
0100 1101	4D	19,712	77
0100 1110	4E	19,968	78
0100 1111	4F	20,224	79
0101 0000	50	20,480	80
0101 0001	51	20,736	81
0101 0010	52	20,992	82
0101 0011	53	21,248	83
0101 0100	54	21,504	84
0101 0101	55	21,760	85
0101 0110	56	22,016	86
0101 0111	57	22,272	87
0101 1000	58	22,528	88
0101 1001	59	22,784	89
0101 1010	5A	23,040	90
0101 1011	5B	23,296	91
0101 1100	5C	23,552	92
0101 1101	5D	23,808	93
0101 1110	5E	24,064	94
0101 1111	5F	24,320	95
0110 0000	60	24,576	96
0110 0001	61	24,832	97
0110 0010	62	25,088	98
0110 0011	63	25,344	99
0110 0100	64	25,600	100
0110 0101	65	25,856	101
0110 0110	66	26,112	102
0110 0111	67	26,368	103
0110 1000	68	26,624	104
0110 1001	69	26,880	105
0110 1010	6A	27,136	106
0110 1011	6B	27,392	107
0110 1100	6C	27,648	108
0110 1101	6D	27,904	109
0110 1110	6E	28,160	110
0110 1111	6F	28,416	111
0111 0000	70	28,672	112
0111 0001	71	28,928	113
0111 0010	72	29,184	114
0111 0011	73	29,440	115
0111 0100	74	29,696	116
0111 0101	75	29,952	117
0111 0110	76	30,208	118
0111 0111	77	30,464	119
0111 1000	78	30,720	120

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
0111 1001	79	30,976	121
0111 1010	7A	31,232	122
0111 1011	7B	31,488	123
0111 1100	7C	31,744	124
0111 1101	7D	32,000	125
0111 1110	7E	32,256	126
0111 1111	7F	32,512	127
1000 0000	80	32,768	128
1000 0001	81	33,024	129
1000 0010	82	33,280	130
1000 0011	83	33,536	131
1000 0100	84	33,792	132
1000 0101	85	34,048	133
1000 0110	86	34,304	134
1000 0111	87	34,560	135
1000 1000	88	34,816	136
1000 1001	89	35,072	137
1000 1010	8A	35,328	138
1000 1011	8B	35,584	139
1000 1100	8C	35,840	140
1000 1101	8D	36,096	141
1000 1110	8E	36,352	142
1000 1111	8F	36,608	143
1001 0000	90	36,864	144
1001 0001	91	37,120	145
1001 0010	92	37,376	146
1001 0011	93	37,632	147
1001 0100	94	37,888	148
1001 0101	95	38,144	149
1001 0110	96	38,400	150
1001 0111	97	38,656	151
1001 1000	98	38,912	152
1001 1001	99	39,168	153
1001 1010	9A	39,424	154
1001 1011	9B	39,680	155
1001 1100	9C	39,936	156
1001 1101	9D	40,192	157
1001 1110	9E	40,448	158
1001 1111	9F	40,704	159
1010 0000	A0	40,960	160
1010 0001	A1	41,216	161
1010 0010	A2	41,472	162
1010 0011	A3	41,728	163
1010 0100	A4	41,984	164
1010 0101	A5	42,240	165

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
1010 0110	A6	42,496	166
1010 0111	A7	42,752	167
1010 1000	A8	43,008	168
1010 1001	A9	43,264	169
1010 1010	AA	43,520	170
1010 1011	AB	43,776	171
1010 1100	AC	44,032	172
1010 1101	AD	44,288	173
1010 1110	AE	44,544	174
1010 1111	AF	44,800	175
1011 0000	B0	45,056	176
1011 0001	B1	45,312	177
1011 0010	B2	45,568	178
1011 0011	B3	45,824	179
1011 0100	B4	46,080	180
1011 0101	B5	46,336	181
1011 0110	B6	46,592	182
1011 0111	B7	46,848	183
1011 1000	B8	47,104	184
1011 1001	B9	47,360	185
1011 1010	BA	47,616	186
1011 1011	BB	47,872	187
1011 1100	BC	48,128	188
1011 1101	BD	48,384	189
1011 1110	BE	48,640	190
1011 1111	BF	48,896	191
1100 0000	C0	49,152	192
1100 0001	C1	49,408	193
1100 0010	C2	49,664	194
1100 0011	C3	49,920	195
1100 0100	C4	50,176	196
1100 0101	C5	50,432	197
1100 0110	C6	50,688	198
1100 0111	C7	50,944	199
1100 1000	C8	51,200	200
1100 1001	C9	51,456	201
1100 1010	CA	51,712	202
1100 1011	CB	51,968	203
1100 1100	CC	52,224	204
1100 1101	CD	52,480	205
1100 1110	CE	52,736	206
1100 1111	CF	52,992	207
1101 0000	D0	53,248	208
1101 0001	D1	53,504	209
1101 0010	D2	53,760	210

Binary	*Hexadecimal*	*Upper Byte*	*Lower Byte*
1101 0011	D3	54,016	211
1101 0100	D4	54,272	212
1101 0101	D5	54,528	213
1101 0110	D6	54,784	214
1101 0111	D7	55,040	215
1101 1000	D8	55,296	216
1101 1001	D9	55,552	217
1101 1010	DA	55,808	218
1101 1011	DB	56,064	219
1101 1100	DC	56,320	220
1101 1101	DD	56,576	221
1101 1110	DE	56,832	222
1101 1111	DF	57,088	223
1110 0000	E0	57,344	224
1110 0001	E1	57,600	225
1110 0010	E2	57,856	226
1110 0011	E3	58,112	227
1110 0100	E4	58,368	228
1110 0101	E5	58,624	229
1110 0110	E6	58,880	230
1110 0111	E7	59,136	231
1110 1000	E8	59,392	232
1110 1001	E9	59,648	233
1110 1010	EA	59,904	234
1110 1011	EB	60,160	235
1110 1100	EC	60,416	236
1110 1101	ED	60,672	237
1110 1110	EE	60,928	238
1110 1111	EF	61,184	239
1111 0000	F0	61,440	240
1111 0001	F1	61,696	241
1111 0010	F2	61,952	242
1111 0011	F3	62,208	243
1111 0100	F4	62,464	244
1111 0101	F5	62,720	245
1111 0110	F6	62,976	246
1111 0111	F7	63,232	247
1111 1000	F8	63,488	248
1111 1001	F9	63,744	249
1111 1010	FA	64,000	250
1111 1011	FB	64,256	251
1111 1100	FC	64,512	252
1111 1101	FD	64,768	253
1111 1110	FE	65,024	254
1111 1111	FF	65,280	255

Appendix 2:
2's Complement Representation

Positive			Negative		
Decimal	*Hexadecimal*	*Binary*	*Binary*	*Hexadecimal*	*Decimal*
0	00H	0000 0000	0000 0000	00H	−0
1	01H	0000 0001	1111 1111	FFH	−1
2	02H	0000 0010	1111 1110	FEH	−2
3	03H	0000 0011	1111 1101	FDH	−3
4	04H	0000 0100	1111 1100	FCH	−4
5	05H	0000 0101	1111 1011	FBH	−5
6	06H	0000 0110	1111 1010	FAH	−6
7	07H	0000 0111	1111 1001	F9H	−7
8	08H	0000 1000	1111 1000	F8H	−8
9	09H	0000 1001	1111 0111	F7H	−9
10	0AH	0000 1010	1111 0110	F6H	−10
11	0BH	0000 1011	1111 0101	F5H	−11
12	0CH	0000 1100	1111 0100	F4H	−12
13	0DH	0000 1101	1111 0011	F3H	−13
14	0EH	0000 1110	1111 0010	F2H	−14
15	0FH	0000 1111	1111 0001	F1H	−15
16	10H	0001 0000	1111 0000	F0H	−16
17	11H	0001 0001	1110 1111	EFH	−17
18	12H	0001 0010	1110 1110	EEH	−18
19	13H	0001 0011	1110 1101	EDH	−19
20	14H	0001 0100	1110 1100	ECH	−20
21	15H	0001 0101	1110 1011	EBH	−21
22	16H	0001 0110	1110 1010	EAH	−22
23	17H	0001 0111	1110 1001	E9H	−23
24	18H	0001 1000	1110 1000	E8H	−24
25	19H	0001 1001	1110 0111	E7H	−25
26	1AH	0001 1010	1110 0110	E6H	−26
27	1BH	0001 1011	1110 0101	E5H	−27

Positive			Negative		
Decimal	*Hexadecimal*	*Binary*	*Binary*	*Hexadecimal*	*Decimal*
28	1CH	0001 1100	1110 0100	E4H	−28
29	1DH	0001 1101	1110 0011	E3H	−29
30	1EH	0001 1110	1110 0010	E2H	−30
31	1FH	0001 1111	1110 0001	E1H	−31
32	20H	0010 0000	1110 0000	E0H	−32
33	21H	0010 0001	1101 1111	DFH	−33
34	22H	0010 0010	1101 1110	DEH	−34
35	23H	0010 0011	1101 1101	DDH	−35
36	24H	0010 0100	1101 1100	DCH	−36
37	25H	0010 0101	1101 1011	DBH	−37
38	26H	0010 0110	1101 1010	DAH	−38
39	27H	0010 0111	1101 1001	D9H	−39
40	28H	0010 1000	1101 1000	D8H	−40
41	29H	0010 1001	1101 0111	D7H	−41
42	2AH	0010 1010	1101 0110	D6H	−42
43	2BH	0010 1011	1101 0101	D5H	−43
44	2CH	0010 1100	1101 0100	D4H	−44
45	2DH	0010 1101	1101 0011	D3H	−45
46	2EH	0010 1110	1101 0010	D2H	−46
47	2FH	0010 1111	1101 0001	D1H	−47
48	30H	0011 0000	1101 0000	D0H	−48
49	31H	0011 0001	1100 1111	CFH	−49
50	32H	0011 0010	1100 1110	CEH	−50
51	33H	0011 0011	1100 1101	CDH	−51
52	34H	0011 0100	1100 1100	CCH	−52
53	35H	0011 0101	1100 1011	CBH	−53
54	36H	0011 0110	1100 1010	CAH	−54
55	37H	0011 0111	1100 1001	C9H	−55
56	38H	0011 1000	1100 1000	C8H	−56
57	39H	0011 1001	1100 0111	C7H	−57
58	3AH	0011 1010	1100 0110	C6H	−58
59	3BH	0011 1011	1100 0101	C5H	−59
60	3CH	0011 1100	1100 0100	C4H	−60
61	3DH	0011 1101	1100 0011	C3H	−61
62	3EH	0011 1110	1100 0010	C2H	−62
63	3FH	0011 1111	1100 0001	C1H	−63
64	40H	0100 0000	1100 0000	C0H	−64
65	41H	0100 0001	1011 1111	BFH	−65
66	42H	0100 0010	1011 1110	BEH	−66
67	43H	0100 0011	1011 1101	BDH	−67
68	44H	0100 0100	1011 1100	BCH	−68
69	45H	0100 0101	1011 1011	BBH	−69

Positive			Negative		
Decimal	*Hexadecimal*	*Binary*	*Binary*	*Hexadecimal*	*Decimal*
70	46H	0100 0110	1011 1010	BAH	−70
71	47H	0100 0111	1011 1001	B9H	−71
72	48H	0100 1000	1011 1000	B8H	−72
73	49H	0100 1001	1011 0111	B7H	−73
74	4AH	0100 1010	1011 0110	B6H	−74
75	4BH	0100 1011	1011 0101	B5H	−75
76	4CH	0100 1100	1011 0100	B4H	−76
77	4DH	0100 1101	1011 0011	B3H	−77
78	4EH	0100 1110	1011 0010	B2H	−78
79	4FH	0100 1111	1011 0001	B1H	−79
80	50H	0101 0000	1011 0000	B0H	−80
81	51H	0101 0001	1010 1111	AFH	−81
82	52H	0101 0010	1010 1110	AEH	−82
83	53H	0101 0011	1010 1101	ADH	−83
84	54H	0101 0100	1010 1100	ACH	−84
85	55H	0101 0101	1010 1011	ABH	−85
86	56H	0101 0110	1010 1010	AAH	−86
87	57H	0101 0111	1010 1001	A9H	−87
88	58H	0101 1000	1010 1000	A8H	−88
89	59H	0101 1001	1010 0111	A7H	−89
90	5AH	0101 1010	1010 0110	A6H	−90
91	5BH	0101 1011	1010 0101	A5H	−91
92	5CH	0101 1100	1010 0100	A4H	−92
93	5DH	0101 1101	1010 0011	A3H	−93
94	5EH	0101 1110	1010 0010	A2H	−94
95	5FH	0101 1111	1010 0001	A1H	−95
96	60H	0110 0000	1010 0000	A0H	−96
97	61H	0110 0001	1001 1111	9FH	−97
98	62H	0110 0010	1001 1110	9EH	−98
99	63H	0110 0011	1001 1101	9DH	−99
100	64H	0110 0100	1001 1100	9CH	−100
101	65H	0110 0101	1001 1011	9BH	−101
102	66H	0110 0110	1001 1010	9AH	−102
103	67H	0110 0111	1001 1001	99H	−103
104	68H	0110 1000	1001 1000	98H	−104
105	69H	0110 1001	1001 0111	97H	−105
106	6AH	0110 1010	1001 0110	96H	−106
107	6BH	0110 1011	1001 0101	95H	−107
108	6CH	0110 1100	1001 0100	94H	−108
109	6DH	0110 1101	1001 0011	93H	−109
110	6EH	0110 1110	1001 0010	92H	−110
111	6FH	0110 1111	1001 0001	91H	−111

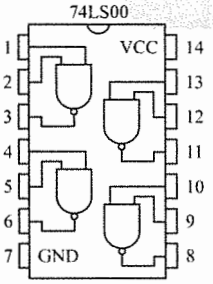
Positive			Negative		
Decimal	*Hexadecimal*	*Binary*	*Binary*	*Hexadecimal*	*Decimal*
112	70H	0111 0000	1001 0000	90H	−112
113	71H	0111 0001	1000 1111	8FH	−113
114	72H	0111 0010	1000 1110	8EH	−114
115	73H	0111 0011	1000 1101	8DH	−115
116	74H	0111 0100	1000 1100	8CH	−116
117	75H	0111 0101	1000 1011	8BH	−117
118	76H	0111 0110	1000 1010	8AH	−118
119	77H	0111 0111	1000 1001	89H	−119
120	78H	0111 1000	1000 1000	88H	−120
121	79H	0111 1001	1000 0111	87H	−121
122	7AH	0111 1010	1000 0110	86H	−122
123	7BH	0111 1011	1000 0101	85H	−123
124	7CH	0111 1100	1000 0100	84H	−124
125	7DH	0111 1101	1000 0011	83H	−125
126	7EH	0111 1110	1000 0010	82H	−126
127	7FH	0111 1111	1000 0001	81H	−127
128	—	—	1000 0000	80H	−128

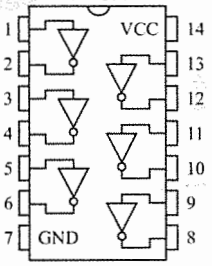
Appendix 3:
TTL Devices

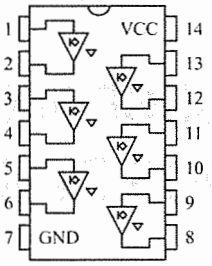
Number	*Function*	*Number*	*Function*
7400	Quad 2-input NAND gates	7441	BCD-to-decimal decoder-Nixie driver
7401	Quad 2-input NAND gates (open collector)	7442	BCD-to-decimal decoder
7402	Quad 2-input NOR gates	7443	Excess 3-to-decimal decoder
7403	Quad 2-input NOR gates (open collector)	7444	Excess Gray-to-decimal
7404	Hex inverters	7445	BCD-to-decimal decoder-driver
7405	Hex inverters (open collector)	7446	BCD-to-seven segment decoder-drivers
7406	Hex inverter buffer-driver		(30-V output)
7407	Hex buffer-drivers	7447	BCD-to-seven segment decoder-drivers
7408	Quad 2-input AND gates		(15-V output)
7409	Quad 2-input AND gates (open collector)	7448	BCD-to-seven segment decoder-drivers
7410	Triple 3-input NAND gates	7450	Expandable dual 2-input 2-wide AND-
7411	Triple 3-input AND gates		OR-INVERT gates
7412	Triple 3-input NAND gates (open collector)	7451	Dual 2-input 2-wide AND-OR-INVERT gates
7413	Dual Schmitt triggers	7452	Expandable 2-input 4-wide AND-OR gates
7414	Hex Schmitt triggers	7453	Expandable 2-input 4-wide AND-OR-INVERT
7416	Hex inverter buffer-drivers		gates
7417	Hex buffer-drivers	7454	2-input 4-wide AND-OR-INVERT gates
7420	Dual 4-input NAND gates	7455	Expandable 4-input 2-wide AND-OR-INVERT
7421	Dual 4-input AND gates		gates
7422	Dual 4-input NAND gates (open collector)	7459	Dual 2-3 input 2-wide AND-OR-INVERT
7423	Expandable dual 4-input NOR gates		gates
7425	Dual 4-input NOR gates	7460	Dual 4-input expanders
7426	Quad 2-input TTL-MOS interface NAND gates	7461	Triple 3-input expanders
7427	Triple 3-input NOR gates	7462	2-2-3-3 input 4-wide expanders
7428	Quad 2-input NOR buffer	7464	2-2-3-4 input 4-wide AND-OR-INVERT gates
7430	8-input NAND gate	7465	4-wide AND-OR-INVERT gates (open collector)
7432	Quad 2-input OR gates	7470	Edge-triggered *JK* flip-flop
7437	Quad 2-input NAND buffers	7472	*JK* master-slave flip-flop
7438	Quad 2-input NAND buffers (open collector)	7473	Dual *JK* master-slave flip-flop
7439	Quad 2-input NAND buffers (open collector)	7474	Dual *D* flip-flop
7440	Dual 4-input NAND buffers	7475	Quad latch

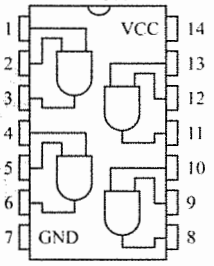
Number	Function	Number	Function
7476	Dual *JK* master-slave flip-flop	74162	Synchronous 4-bit counter
7480	Gates full adder	74163	Synchronous 4-bit counter
7482	2-bit binary full adder	74164	8-bit serial shift register
7483	4-bit binary full adder	74165	Parallel-load 8-bit serial shift register
7485	4-bit magnitude comparator	74166	8-bit shift register
7486	Quad EXCLUSIVE-OR gate	74173	4-bit three-state register
7489	64-bit random-access read-write memory	74174	Hex *F* flip-flop with clear
7490	Decade counter	74175	Quad *D* flip-flop with clear
7491	8-bit shift register	74176	35-MHz presettable decade counter
7492	Divide-by-12 counter	74177	35-MHz presettable binary counter
7493	4-bit binary counter	74179	4-bit parallel-access shift register
7494	4-bit shift register	74180	8-bit odd-even parity generator-checker
7495	4-bit right-shift–left-shift register	74181	Arithmetic-logic unit
7496	5-bit parallel-in–parallel-out shift register	74182	Look-ahead carry generator
74100	4-bit bistable latch	74184	BCD-to-binary converter
74104	*JK* master-slave flip-flop	74185	Binary-to-BCD converter
74105	*JK* master-slave flip-flop	74189	Three-state 64-bit random-access memory
74107	Dual *JK* master-slave flip-flop	74190	Up-down decade counter
74109	Dual *JK* positive-edge-triggered flip-flop	74191	Synchronous binary up-down counter
74116	Dual 4-bit latches with clear	74192	Binary up-down counter
74121	Monostable multivibrator	74193	Binary up-down counter
74122	Monostable multivibrator with clear	74194	4-bit directional shift register
74123	Monostable multivibrator	74195	4-bit parallel-access shift register
74125	Three-state quad bus buffer	74196	Presettable decade counter
74126	Three-state quad bus buffer	74197	Presettable binary counter
74132	Quad Schmitt trigger	74198	8-bit shift register
74136	Quad 2-input EXCLUSIVE-OR gate	74199	8-bit shift register
74141	BCD-to-decimal decoder-driver	74221	Dual one-shot Schmitt trigger
74142	BCD counter-latch-driver	74251	Three-state 8-channel multiplexer
74145	BCD-to-decimal decoder-driver	74259	8-bit addressable latch
74147	10/4 priority encoder	74276	Quad *JK* flip-flop
74148	Priority encoder	74279	Quad debouncer
74150	16-line-to-1-line multiplexer	74283	4-bit binary full adder with fast carry
74151	8-Channel digital multiplexer	74284	Three-state 4-bit multiplexer
74152	8-Channel data selector-multiplexer	74285	Three-state 4-bit multiplexer
74153	Dual 4/1 multiplexer	74365	Three-state hex buffers
74154	4-line-to-16-line decoder-demultiplexer	74366	Three-state hex buffers
74155	Dual 2/4 demultiplexer	74367	Three-state hex buffers
74156	Dual 2/4 demultiplexer	74368	Three-state hex buffers
74157	Quad 2/1 data selector	74390	Individual clocks with flip-flops
74160	Decade counter with asynchronous clear	74393	Dual 4-bit binary counter
74161	Synchronous 4-bit counter		

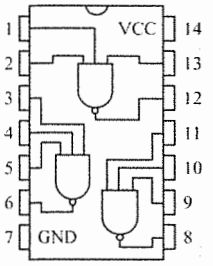
TTL CIRCUITS

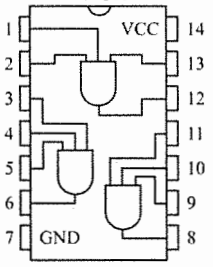
74LS00


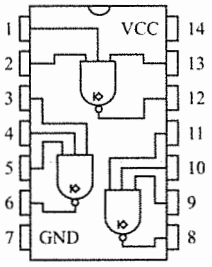
74LS04


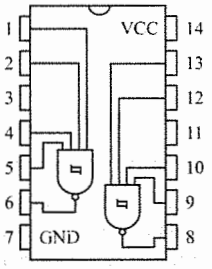
7407/7417


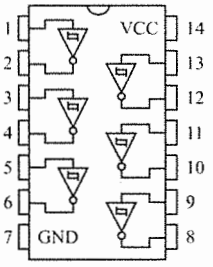
74LS08


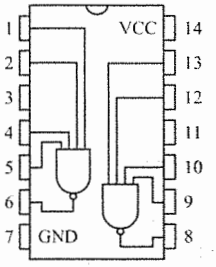
74LS10


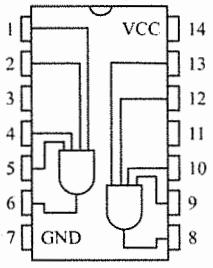
74LS11


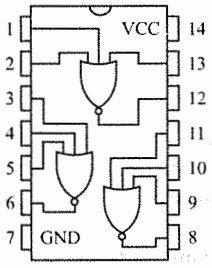
74LS12


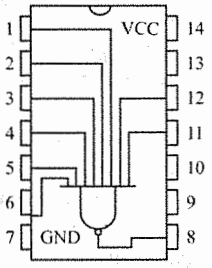
74LS13


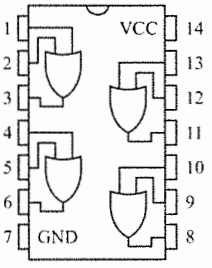
74LS14


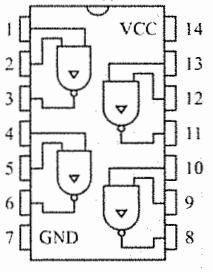
74LS20


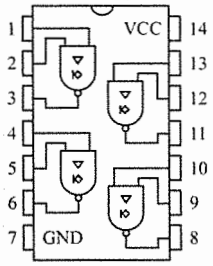
74LS21


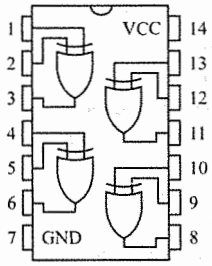
74LS27


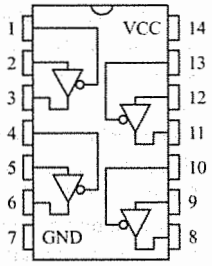
74LS30


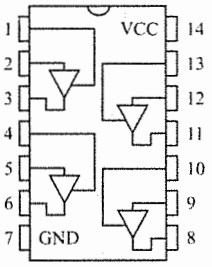
74LS32


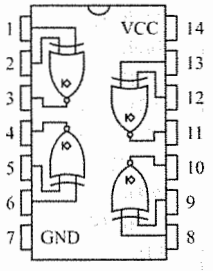
74LS37


74LS38


74LS86


74LS125


74LS126


74LS266


Appendix 4:
CMOS Devices

74HC00 Series

Part No.	*Pins*	*Function*
74HC00	14	Quad 2-input NAND gate
74HC02	14	Quad 2-input NOR gate
74HC04	14	Hex inverter (buffered)
74HC08	14	Quad 2-input AND gate
74HC14	14	Hex inverting Schmitt trigger
74HC20	14	Dual 4-input NAND gate
74HC30	14	8-input NAND gate
74HC32	14	Quad 2-input OR gate
74HC42	16	BCD-to-decimal decoder
74HC74	14	Dual *D* flip-flop with preset and clear
74HC85	16	4-bit magnitude comparator
74HC123	16	Dual monostable multivibrator
74HC132	14	Quad 2-input NAND Schmitt trigger
74HC138	16	3- to 8-line decoder
74HC139	16	Expandable dual 2- to 4-line decoder
74HC154	S-24	4- to 16-line decoder (use 24SLP socket)
74HC161	16	Synchronous binary counter
74HC163	16	Synchronous binary counter
74HC164	14	8-bit serial in–parallel out shift register
74HC165	16	8-bit parallel in–serial out shift register
74HC174	16	Hex *D* Flip-Flop with clear
74HC175	16	Quad *D* type flip-flop with clear
74HC191	16	Up-down binary counter
74HC192	16	Synchronous decade up-down counter
74HC193	16	Synchronous binary up-down counter
74HC221	16	Dual monostable multivibrator
74HC240	20	Inverting octal tri-state buffer
74HC244	20	Octal tri-state buffer

74HC00 Series

Part No.	*Pins*	*Function*
74HC245	20	Octal tri-state transceiver
74HC257	16	Quad 2-channel tri-state multiplexer
74HC273	20	Octal *D* flip-flop
74HC367	16	Tri-state hex buffer
74HC373	20	Tri-state octal *D*-type latch
74HC374	20	Tri-state octal *D*-type flip-flop
74HC390	16	Dual 4-bit decade counter
74HC393	14	Dual 4-bit binary counter
74HC541	20	Octal buffer–line driver (tri-state)
74HC573	20	Tri-state octal *D*-type latch
74HC574	20	Tri-state octal *D*-type flip-flop
74HC595	16	8-bit serial to-parallel shift register latch
74HC688	20	8-bit magnitude comparator (equality detector)
74HC942	20	Full duplex low-speed 300-baud modem chip
74HC943	20	Full duplex 300-baud modem chip
74HC4017	16	Decade counter-divider with 10 decoded outputs
74HC4020	16	14-stage binary counter
74HC4040	16	12-stage binary counter
74HC4046	16	CMOS phase-lock loop
74HC4060	16	14-stage binary counter
74HC4066	14	Quad analog switch
74HC4514	S-24	4- to 16-line decoder with latch (use 24SLP socket)
74HC4538	16	Dual retriggerable monostable multivibrator

Appendix 5:
Codes

Table A5.1

Decimal	*BCD*	*Binary*
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	0001 0000	1010
11	0001 0001	1011
12	0001 0010	1100
13	0001 0011	1101
...	.....	...
98	1001 1000	1100010
99	1001 1001	1100011
100	0001 0000 0000	1100100
101	0001 0000 0001	1100101
102	0001 0000 0010	1100110
...	.....	...
578	0101 0111 1000	1001000010
...	.....	.....

Table A5.2 4-Bit BCD Codes

Decimal	*7421*	*6311*	*5421*	*5311*	*5211*
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001
2	0010	0011	0010	0011	0011
3	0011	0100	0011	0100	0101
4	0100	0101	0100	0101	0111
5	0101	0111	1000	1000	1000
6	0110	1000	1001	1001	1001
7	1000	1001	1010	1011	1011
8	1001	1011	1011	1100	1101
9	1010	1100	1100	1101	1111

Table A5.3 More 4-Bit BCD Codes

Decimal	*4221*	*3321*	*2421*	*84$\bar{2}\bar{1}$*	*74$\bar{2}\bar{1}$*
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0111	0111
2	0010	0010	0010	0110	0110
3	0011	0011	0011	0101	0101
4	1000	0101	0100	0100	0100
5	0111	1010	1011	1011	1010
6	1100	1100	1100	1010	1001
7	1101	1101	1101	1001	1000
8	1110	1110	1110	1000	1111
9	1111	1111	1111	1111	1110

Table A5.4 5-Bit BCD Codes

Decimal	*2-out-of-5*	*63210*	*Shift-Counter*	*86421*	*51111*
0	00011	00110	00000	00000	00000
1	00101	00011	00001	00001	00001
2	00110	00101	00011	00010	00011
3	01001	01001	00111	00011	00111
4	01010	01010	01111	00100	01111
5	01100	01100	11111	00101	10000
6	10001	10001	11110	01000	11000
7	10010	10010	11100	01001	11100
8	10100	10100	11000	10000	11110
9	11000	11000	10000	10001	11111

Table A5.5 More than 5-Bit BCD Codes

Decimal	*50 43210*	*543210*	*9876543210*
0	01 00001	000001	0000000001
1	01 00010	000010	0000000010
2	01 00100	000100	0000000100
3	01 01000	001000	0000001000
4	01 10000	010000	0000010000
5	10 00001	100001	0000100000
6	10 00010	100010	0001000000
7	10 00100	100100	0010000000
8	10 01000	101000	0100000000
9	10 10000	110000	1000000000

Table A5.6 Excess-3 Code

Decimal	*BCD*	*Excess-3*
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	0001	1100

Table A5.7 Gray Code

Decimal	*Gray Code*	*Binary*
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111
...	...	...

Appendix 6:
BCD Codes

Hollerith Code

Punched cards are rarely, if ever, used today, but the *Hollerith code* is included here for historical and reference purposes. In this code the numbers 0 through 9 are represented by a single punch in a vertical column. For example, a hole punched in the fifth row of column 12 represents a 5 in that column. The letters of the alphabet are represented by two punches in any one column. The letters A and I are represented by a zone punch in row 12 and a punch in rows 1 through 9. The letters J through R are represented by a zone punch in row 11 and a punch in rows I through 9. The letters S through Z are represented by a zone punch in row 0 and a punch in rows 2 through 9. Thus, any of the 10 decimal digits and any of the 26 letters of the alphabet can be represented in a binary fashion by punching the proper holes in the card. In addition, a number of special characters can be represented by punching combinations of holes in a column which are not used for the numbers or letters of the alphabet.

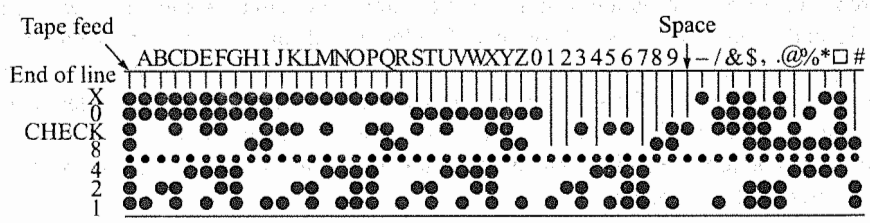
Fig. A6.1 Standard punched card using Hollerith code

An easy device for remembering the alphabetic characters is the phrase "JR is 11." Notice that the letters J through R have an 11 punch, those before have a 12 punch, and those after have a 0 punch. It is also necessary to remember that S begins on a 2 and not a 1.

Eight-Hole Code

Again, this information is for historical and reference purposes. There are a number of codes for punching data in paper tape, but one of the most widely used is the *eight-hole code*. Holes, representing data, are punched in eight parallel channels which run the length of the tape. (The channels are labeled 1, 2, 4, 8, parity, 0, *X*, and end of line.) Each character—numeric, alphabetic, or special—occupies one column of eight positions across the width of the tape.

Numbers are represented by punches in one or more channels labeled 0, 1, 2, 4, and 8, and each number is the sum of the punch positions. For example, 0 is represented by a single punch in the 0 channel; 1 is represented by a single punch in the 1 channel; 2 is a single punch in channel 2; 3 is a punch in channel 1 and a punch in channel 2, etc. Alphabetic characters are represented by a combination of punches in channels *X*, 0, 1, 2, 4, and 8. Channels *X* and 0 are used much as the zone punches in punched cards. For example, the letter *A* is designated by punches in channels *X*, 0, and 1. The special characters are represented by combinations of punches in all channels which are not used to designate either numbers or letters. A punch in the end-of-line channel signifies the end of a block of information, or the end of record. This is the only time a punch appears in this channel.



As a means of checking the validity of the information punched on the tape, the parity channel is used to ensure that each character is represented by an *odd* number of holes. For example, the letter *C* is represented by punches in channels *X*, 0, 1, and 2. Since an odd number of holes is required for each character, the code for the letter C also has a punch in the parity channel, and thus a total of five punches is used for this letter.

Universal Product Code (UPC)

The Universal Product Code (UPC) symbol in Fig. A6.3 is an example of a machine-readable label that appears on virtually every kind of retail grocery product. It is the result of an industrywide attempt to improve productivity through the use of automatic checkstand equipment. The standard symbol consists of a number of parallel light and dark bars of variable widths.

The symbol is designed around a 10-digit numbering system, 5 digits being assigned as an identification number for each manufacturer and the remaining 5 digits being used to identify a specific product, e.g. creamed corn, pea soup, or catsup. Each symbol can be read by a fixed-position scanner, as on a conveyor belt, or by a hand-held wand. The code numbers are printed on each symbol under the bars as a convenience in the event of equipment failure.

The American Standard Code for Information Exchange*								
	000	001	010	011	100	101	110	111
0000	*NULL*	① DC_0	*b*	0	@	*P*		
0001	*SOM*	DC_1	!	1	*A*	*Q*		
0010	*EOA*	DC_2	"	2	*B*	*R*		
0011	*EOM*	DC_3	#	3	*C*	*S*		
0100	*EOT*	DC_4 (Stop)	$	4	*D*	*T*		
0101	*WRU*	ERR	%	5	*E*	*U*		
0110	*RU*	*SYNC*	&	6	*F*	*V*		
0111	*BELL*	*LEM*	'	7	*G*	*W*		
1000	FE_0	S_0	(	8	*H*	*X*	Unassigned	
1001	*HT* / *SK*	S_1	)	9	*I*	*Y*		
1010	*LF*	S_2	*	:	*J*	*Z*		
1011	V_{TAB}	S_3	+	;	*K*	[		
1100	*FF*	S_4	,	<	*L*	\		ACK
1101	*CR*	S_5	–	=	*M*	]		②
1110	*SO*	S_6	*	>	*N*	↑		*ESC*
1111	*SI*	S_7	/	?	*O*	←		*DEL*

Example: 100 0001 = *A*

b_1----------b_1

The abbreviations used in the figure mean:

NULL	Null idle	*CR*	Carriage return
SOM	Start of message	*SO*	Shift out
EOA	End of address	*SI*	Shift in
EOM	End of message	DC_0	Device control 1
			Reserved for data
			Link escape
EOT	End of transmission	DC_1—DC_2	Device control
WRU	"Who are you?"	*ERR*	Error
RU	"Are you . . . ?"	*SYNC*	Synchronous idle
BELL	Audible signal	*LEM*	Logical end of media
FE	Format effector	SO_0—SO_7	Separator (information)
HT	Horizontal tabulation		Word separator (blank, normally nonprinting)
SK	Skip (punched card)	*ACK*	Acknowledge
LF	Line feed	2	Unassigned control
V/TAB	Vertical tabulation	*ESC*	Escape
FF	Form feed	*DEL*	Delete idle

*Reprinted from *Digital Computer Fundamentals* by Thomas C. Bartee. Copyright 1960, 1966 by McGraw-Hill, Inc. Used with permission of McGraw-Hill Book Company.

Fig. A6.2 **American Standard Code for Information Exchange**

Specifications

Each 10-digit symbol is rectangular in shape and consists of exactly 30 dark and 29 light vertical bars, as seen in Fig. A6.4. Each digit is represented by two dark bars and two light spaces. To account for the variable widths of the bars, each digit or character is broken down into seven modules. A module can be either dark or light, and each dark bar is made up of 1, 2, 3, or 4 dark modules. An example is shown in Fig. A6.5. Notice that each digit or character has exactly seven modules, and each digit has two dark bars and two light spaces. Dark modules are 1s while light modules are 0s.

Characters are encoded differently at the left and at the right of center. A left-side character begins with a light space and ends with a dark bar and always consists of either three or five dark modules (odd parity). A right-side character begins with a dark bar and ends with a light space and always has either two or four dark modules (even parity). The encoding for each character is summarized in Fig. A6.6.

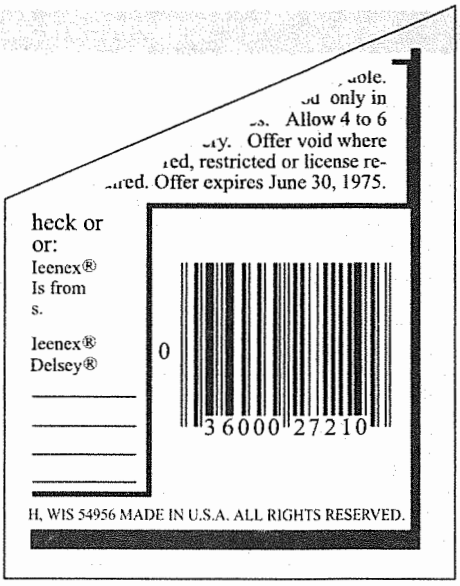


Fig. A6.3 **UPC symbol from box of Kleenex tissues. Registered trademark of Kimberly-Clark Corp., Neenah, Wis**

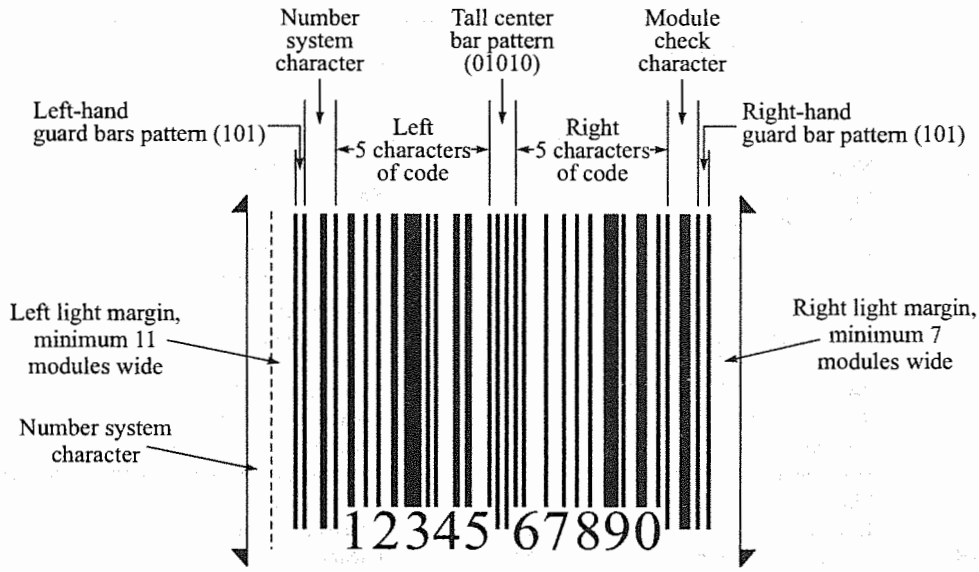


Fig. A6.4 **UPC standard symbol**

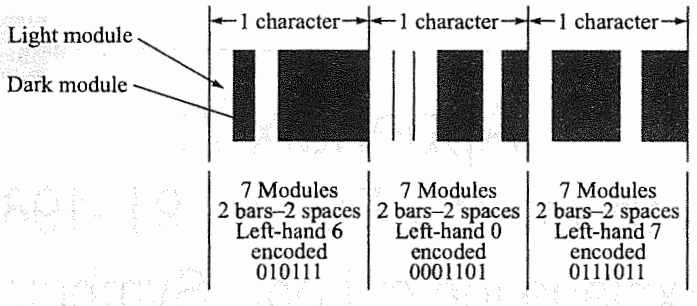


Fig. A6.5 UPC character construction

Decimal number	Left characters (odd parity)	Right characters (even parity)
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	0011100
5	0110001	0001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100

Fig. A6.6 UPC character encoding

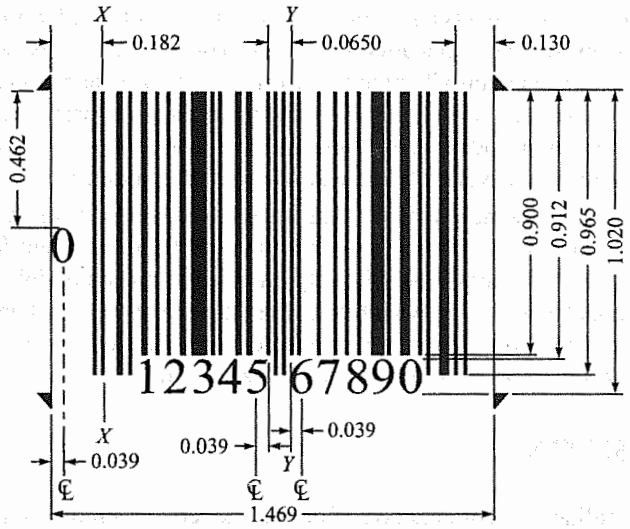


Fig. A6.7 UPC symbol dimensions

Appendix 7: Overview of IEEE Std. 91–1984, Explanation of Logic Symbols*

INTRODUCTION

The International Electrotechnical Commission (IEC) has been developing a very powerful symbolic language that can show the relationship of each input of a digital logic circuit to each output without showing explicitly the internal logic. At the heart of the system is dependency notation.

The system was introduced in the United States in a rudimentary form in IEEE/ANSI Standard Y32.14-1973. Lacking at that time a complete development of dependency notation, it offered little more than a substitution of rectangular shapes for the familiar distinctive shapes for representing the basic functions of AND, OR, negation, etc. This is no longer the case.

Internationally, Working Group 2 of IEC Technical Committee TC-3 has prepared a new document (Publication 617-12) that consolidates the original work started in the mid-1960s and published in 1972 (Publication 117-15) and the amendments and supplements that have followed. Similarly, for the USA, IEEE Committee SCC 11.9 has revised the publication IEEE Std 91/ANSI Y32.14. Now numbered simply IEEE Std 91-1984, the IEEE standard contains all of the IEC work that has been approved, and also a small amount of material still under international consideration. Texas Instruments is participating in the work of both organizations and this document introduces new logic symbols in accordance with the new standards. When changes are made as the standards develop, future editions will take those changes into account.

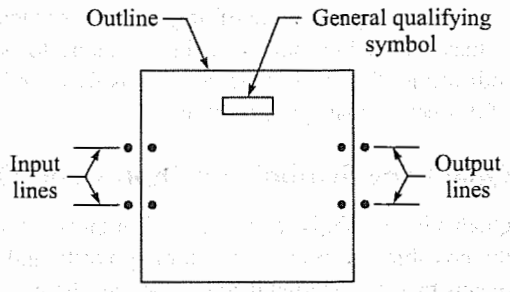
The following explanation of the new symbolic language is necessarily brief and greatly condensed from what the standards publications will contain. This is not intended to be sufficient for those people who will be developing symbols for new devices. It is primarily intended to make possible the understanding of the symbols used in this data book and is somewhat briefer than the explanation that appears in several of Texas Instruments data books on digital logic. However, it includes a new section that explains several symbols for actual devices in detail. This has proven to be a powerful learning aid.

SYMBOL COMPOSITION

A symbol comprises an outline or a combination of outlines together with one or more qualifying symbols. The shape of the symbols is not significant. As shown in Fig. A7.1, general qualifying symbols are used to

*Courtesy of Texas Instruments Incorporated.

tell exactly what logical operation is performed by the elements. Table A7.1 shows general qualifying symbols defined in the new standards. Input lines are placed on the left and output lines are placed on the right. When an exception is made to that convention, the direction of signal flow is indicated by an arrow.



*Possible positions for qualifying symbols relating to inputs and outputs

Fig. A7.1 Symbol composition

QUALIFYING SYMBOLS*

General Qualifying Symbols

Table A7.1 shows general qualifying symbols defined by IEEE Standard 91. These characters are placed near the top center or the geometric center of a symbol or symbol element to define the basic function of the device represented by the symbol or of the element.

Table A7.1 General Qualifying Symbols

Symbol	Description
&	AND gate or function.
≥ 1	OR gate or function. The symbol was chosen to indicate that at least one active input is needed to activate the output.
= 1	Exclusive OR. One and only one input must be active to activate the output.
1	The one input must be active.
▷ or ◁	A buffer or element with more than usual output capability (symbol is oriented in the direction of signal flow).
⊓⊔	Schmitt trigger; element with hysteresis.
X/Y	Coder, code converter, level converter.
	The following are examples of subsets of this general class of qualifying symbol:
	BCD/7-SEG — BCD to 7-segment display driver.
	TTL/MOS — TTL to MOS level converter.
	CMOS/PLASMA DISP — Plasma-display driver with CMOS-compatible inputs.
	MOS/LED — Light-emitting-diode driver with MOS-compatible inputs.
	CMOS/VAC FLUOR DISP — Vacuum-fluorescent display driver with CMOS-compatible inputs.
	CMOS/EL DISP — Electroluminescent display driver with CMOS-compatible inputs.
	TTL/GAS DISCH DISPLAY — Gas-discharge display driver with TTL-compatible inputs.
SRGm	Shift register. m is the number of bits

* Symbols on pages 639 through 641 can be found in Texas Instruments, *High-Speed CMOS Logic Data Book*, pp. 6-5 to 6-10.

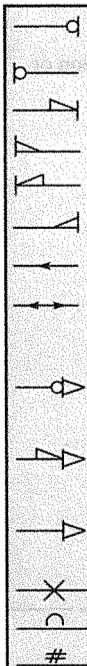
X/Y is the general qualifying symbol for identifying coders, code converters, and level converters. X and Y may be used in their own right to stand for some code or either or both may be replaced by some other indication of the code or level such as BCD or TTL. As might be expected, interface circuits often make use of this set of qualifying symbols.

Qualifying Symbols for Inputs and Outputs

Qualifying symbols for inputs and outputs are shown in Table A7.2 and will be familiar to most users with the possible exception of the logic polarity and analog signal indicators. The older logic negation indicator means that the external 0 state produces the internal 1 state. The internal 1 state means the active state. Logic negation may be used in pure logic diagrams; in order to tie the external 1 and 0 logic states to the levels H (high) and L (low), a statement of whether positive logic (1 = H, 0 = L) or negative logic (1 = L, 0 = H) is being used is required or must be assumed. Logic polarity indicators eliminate the need for calling out the logic convention and are used in the symbology for actual devices. The presence of the triangle polarity indicator indicates that the L logic level will produce the internal 1 state (the active state) or that, in the case of an output, the internal 1 state will produce the external L level. Note how the active direction of transition for a dynamic input is indicated in positive logic, negative logic, and with polarity indication.

When nonstandardized information is shown inside an outline, it is usually enclosed in square brackets [like these]. The square brackets are omitted when associated with a nonlogic input, which is indicated by an X superimposed on the connection line outside the symbol.

Table A7.2 **Qualifying Symbols for Inputs and Outputs**



Logic negation at input. External 0 produces internal 1.

Logic negation at output. Internal 1 produces external 0.

Active-low input. Equivalent to ——◁ in positive logic.

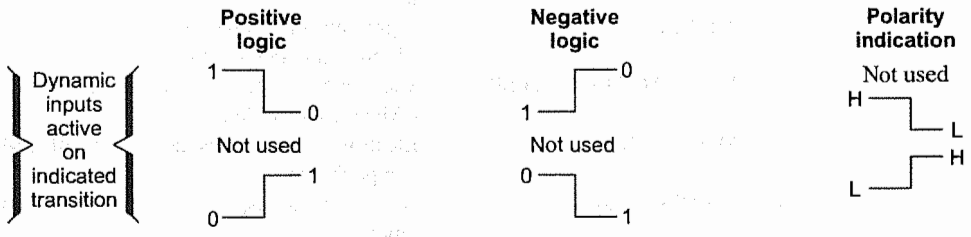
Active-low output. Equivalent to ▷—— in positive logic.

Active-low input in the case of right-to-left signal flow.

Active-low output in the case of right-to-left signal flow.

Signal flow from right to left. If not otherwise indicated, signal flow is from left to right.

Bidirectional signal flow.



Nonlogic connection. A label inside the symbol will usually define the nature of this pin.

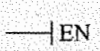
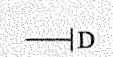
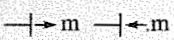
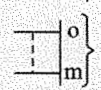
Input for analog signals (on a digital symbol).

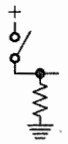
Input for digital signals (on an analog symbol).

Symbols Inside the Outline

Table A7.3 shows some symbols used inside the outline. Note particularly that open-collector (open-drain), open-emitter (open-source), and three-state outputs have distinctive symbols. Also note that an EN input affects all of the outputs of the element and has no effect on inputs. An EN input affects all the external outputs of the element in which it is placed, plus the external outputs of any elements shown to be influenced by that element. It has no effect on inputs. When an enable input affects only certain outputs, affects outputs located outside the indicated influence of the element in which the enable input is placed, and/or affects one or more inputs, a form of dependency notation will indicate this. The effects of the EN input on the various types of outputs are shown.

Table A7.3 Symbols Inside the Outline

	Bi-threshold input (input with hysteresis).
	*npn* open-collector or similar output that can supply a relatively low-impedance L level when not turned off. Requires external pull-up. Capable of positive-logic wired-AND connection.
	Passive-pull-up output is similar to *npn* open-collector output but is supplemented with a built-in passive pull-up.
	*npn* open-emitter or similar output that can supply a relatively low-impedance H level when not turned off. Requires external pull-down. Capable of positive-logic wired-OR connection.
	Passive-pull-down output is similar to *npn* open-emitter output but is supplemented with a built-in passive pull-down.
	3-state output.
	Output with more than usual output capability (symbol is oriented in the direction of signal flow).
	Enable input When at its internal 1-state, all outputs are enabled. When at its internal 0-state, open-collector, open-emitter, and three-state outputs are at external high-impedance state, and all other outputs (i.e., totem-poles) are at the internal 0-state.
J, K, R, S, T	Usual meanings associated with flip-flops (e.g., R = reset, T = toggle).
	Data input to a storage element equivalent to: 
	Shift right (left) inputs, m = 1, 2, 3, etc. If m = 1, it is usually not shown.
	Binary grouping, m is highest power of 2. Produces a number equal to the sum of the weights of the active inputs.
	Input line grouping . . . indicates two or more terminals used to implement a single logic input, e.g., differential inputs.

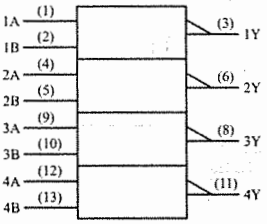


It is particularly important to note that a D input is always the data input of a storage element. At its internal 1 state, the D input sets the storage element to its 1 state, and at its internal 0 state it resets the storage element to its 0 state.

The binary grouping symbol is explained more fully in a later section. Binary-weighted inputs are arranged in order and the binary weights of the least-significant and the most-significant lines are indicated by numbers. In this document weights of input and output lines will be represented by powers of 2 usually only when the binary grouping symbol is used, otherwise decimal numbers will be used. The grouped inputs generate an internal number on which a mathematical function can be performed or that can be an identifying number for dependency notation. This number is the sum of the weights (1, 2, 4 ... , $2n$) of those inputs standing at their 1 states. A frequent use is in addresses for memories.

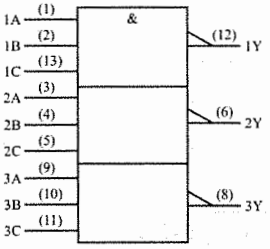
Reversed in direction, the binary grouping symbol can be used with outputs. The concept is analogous to that for the inputs, and the weighted outputs will indicate the internal number assumed to be developed within the circuit.

Appendix 8:
Pinout Diagrams



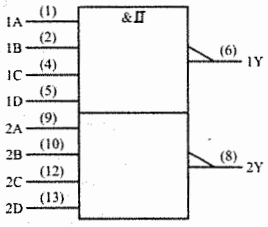
Positive logic: $Y = \overline{AB}$

00



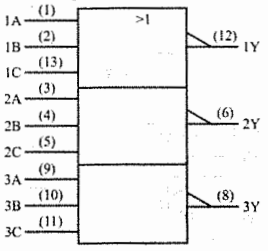
Positive logic: $Y = \overline{ABC}$

10



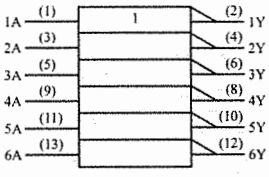
Positive logic: $Y = \overline{ABCD}$

13



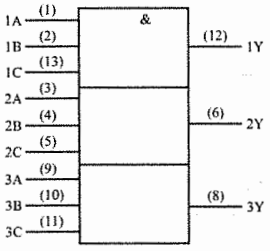
Positive logic: $Y = \overline{A+B+C}$

27



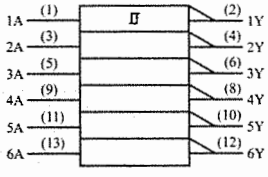
Positive logic: $Y = \overline{A}$

04



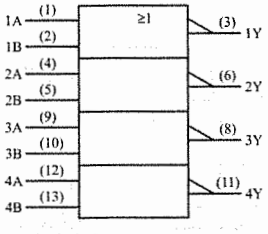
Positive logic: Y = ABC

11



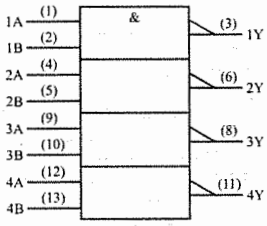
Positive logic: $Y = \overline{A}$

14



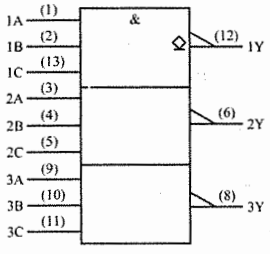
Positive logic: Y = A+B

32



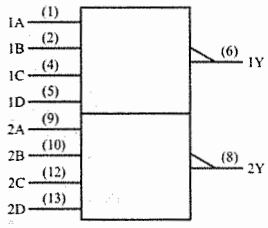
Positive logic: Y = AB

08



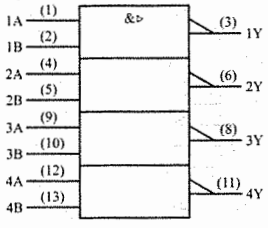
Positive logic: $Y = \overline{ABC}$

12



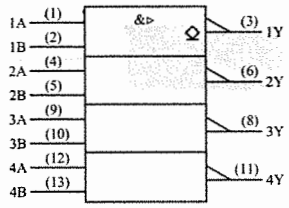
Positive logic: $Y = \overline{ABCD}$

20



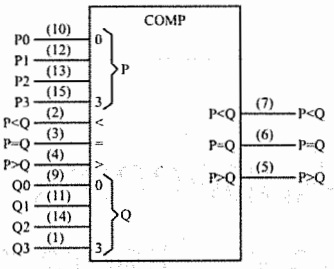
Positive logic: $Y = \overline{AB}$

37

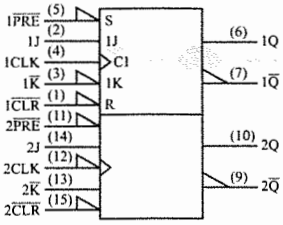


Positive logic: $Y = \overline{AB}$

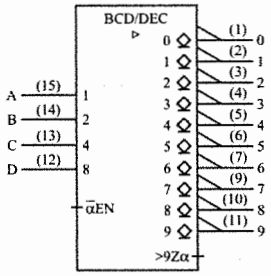
38



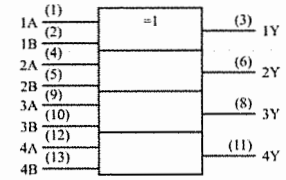
85



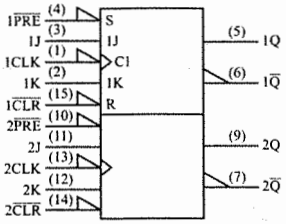
109



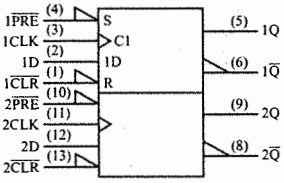
45



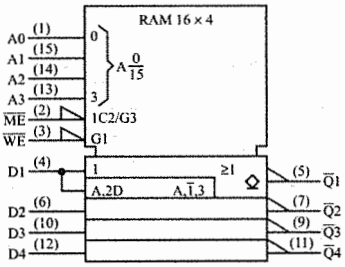
86



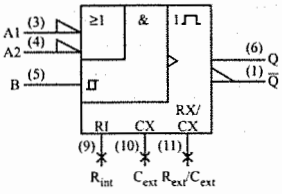
112



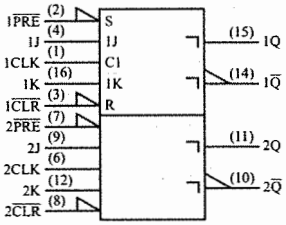
74



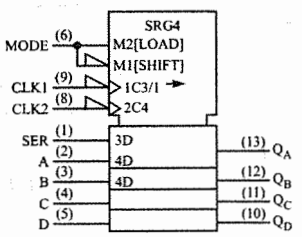
89



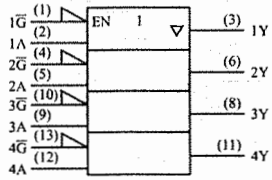
121



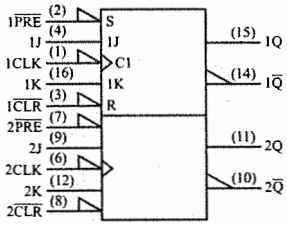
76



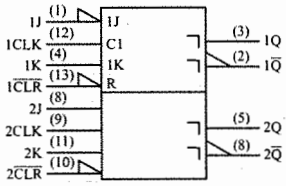
95



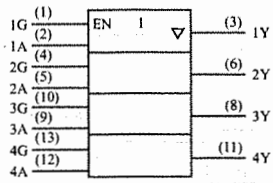
125



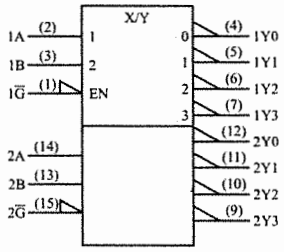
LS76A



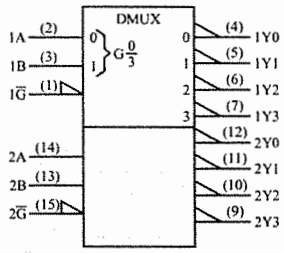
107



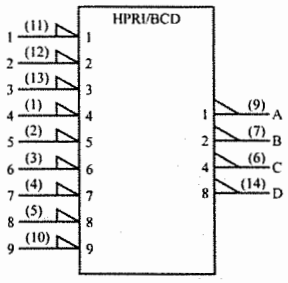
126



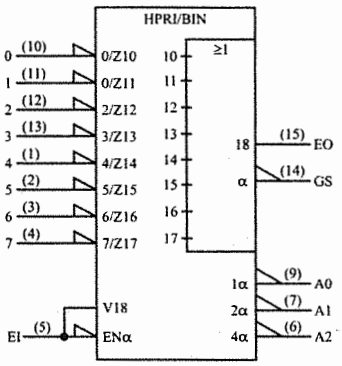
OR



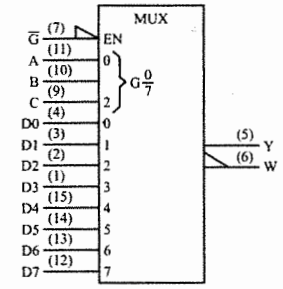
139



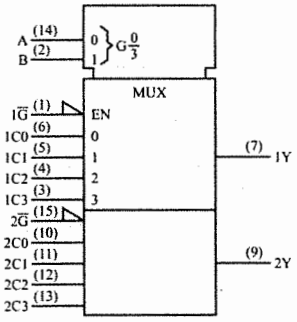
147



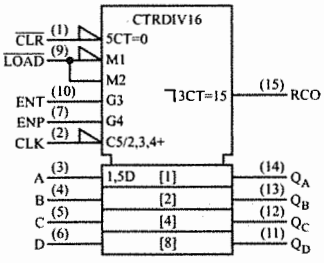
148



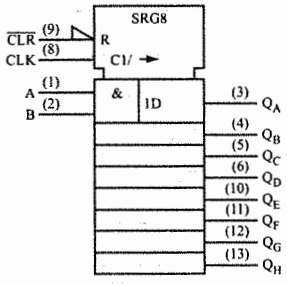
151



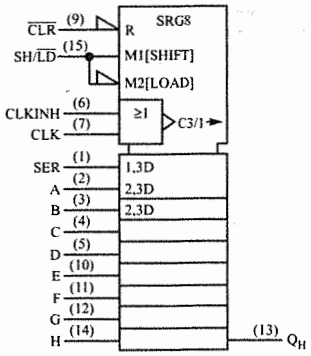
153



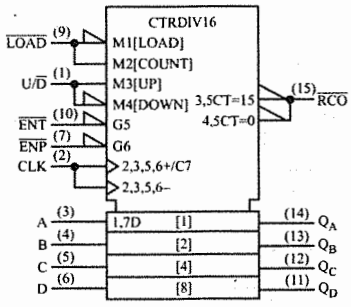
163



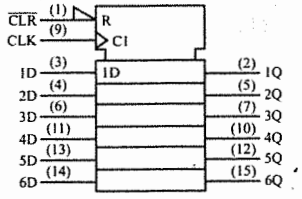
164



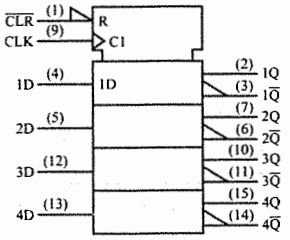
166



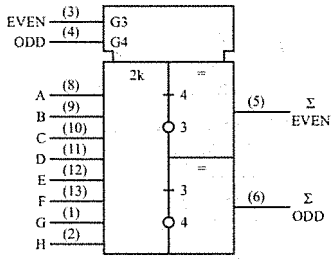
169



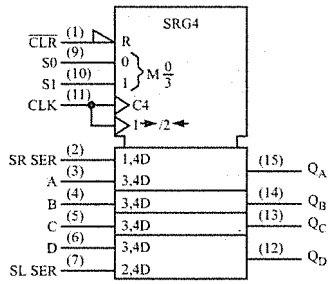
174



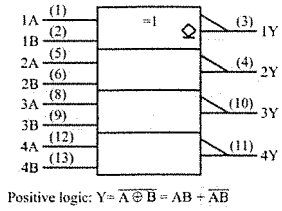
175



180

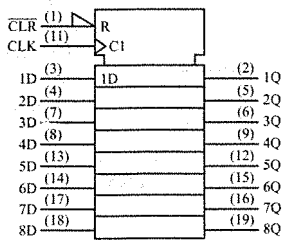


194

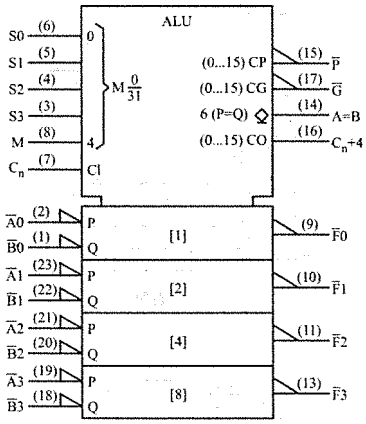


Positive logic: $Y = \overline{A \oplus B} = AB + \overline{A}\,\overline{B}$

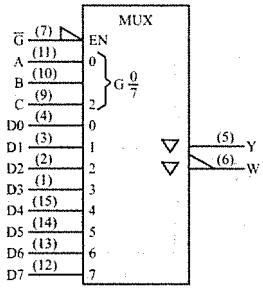
266



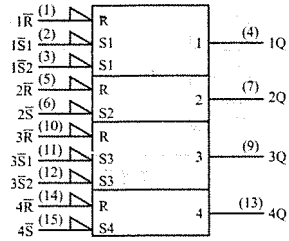
273



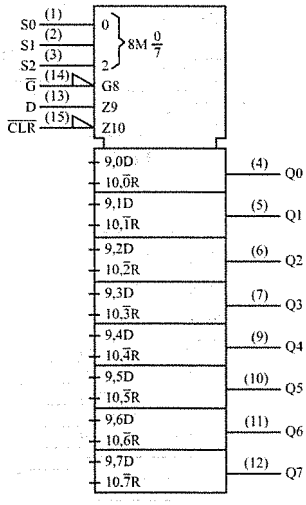
181



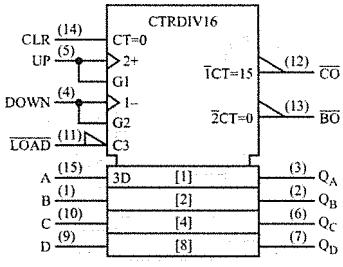
251



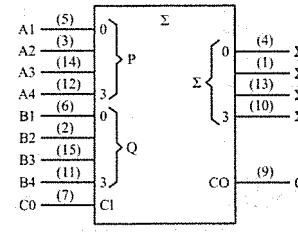
279



259



193



283

Appendix 9:
Answers to Selected Odd-Numbered Problems

Chapter 1

1.1 See Sec. 1.1.

1.3

Lamps *	0	1	0	1	0	1	0	1
*	0	0	1	1	0	0	1	1
*	0	0	0	0	1	1	1	1
No.	0	1	2	3	4	5	6	7

1.5 See Fig. 1.4c.

1.7 $t_H = 100\ \mu s$

1.9 The nonideal waveform is nearly triangular in shape!

1.11

V_1	*L*	*H*	*L*	*H*	*L*	*H*	*L*	*H*
V_2	*L*	*L*	*H*	*H*	*L*	*L*	*H*	*H*
V_3	*L*	*L*	*L*	*L*	*H*	*H*	*H*	*H*
V_0	*L*	*H*	*H*	*H*	*H*	*H*	*H*	*H*

1.13 G must be high to have a signal at V_o. $G = H$ and $V_i = L$.

1.15 Simply connect the output of the first 4-bit register to the input of the second 4-bit register.

1.17 32 connections. 16 connections. True parallel shifting requires a single operation of 250 ns. Shifting 16 bits twice requires 500 ns.

1.19 Nine; seven

1.21 $F = 0100$; CARRY OUT = 0

1.23 Only line 9 is high.

1.25 Handshaking is a request to transfer data into or out of the computer. It is a request to transfer data, followed by an acknowledge, allowing data transfer to begin.

1.27 Because data to be operated on is taken from memory into the CPU and results are moved back to memory for storage.

1.29 54LSXX

1.31 2.5 W

1.33 See Figs. 1.36, 1.37, and 1.38.

Chapter 2

2.1 Low; high

2.3 The truth table is:

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

This is the truth table of a 3-input OR gate. Therefore, a cascade of two 2-input OR gates is equivalent to a 3-input OR gate.

2.5 The truth table is

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

2.7 The truth table is

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

2.9 The truth table is

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

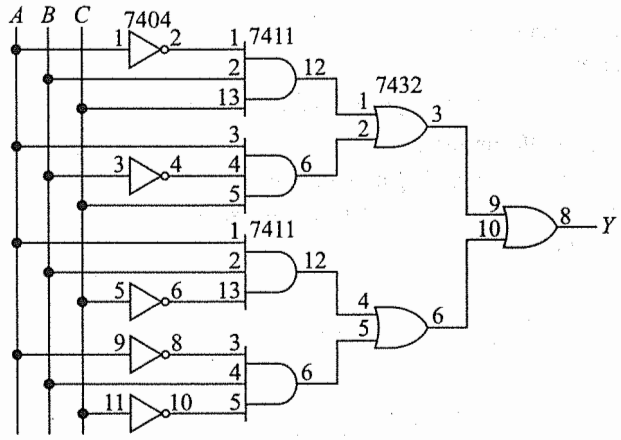
2.11 The Boolean equations are

$$Y = A + B + C$$
$$Y = \overline{(A + B)}$$
$$Y = \overline{(A + B + C)}$$

2.13 The Boolean equations are

$$Y = ABC$$
$$Y = \overline{AB}$$
$$Y = \overline{ABC}$$

2.15 The logic circuit.



2.17 Here is a summary of the truth table. Y equals 1 when $ABCD = 0000$; Y equals 0 for all other $ABCD$ inputs. There are 16 $ABCD$ inputs, starting with 0000 and ending with 1111.

2.19 d

2.21 $Y = \overline{A_0A_1A_2A_3A_4A_5A_6A_7}$

2.23 a

2.25 a. Low b. High c. High d. Low

2.27 a. 1 b. 0 c. 0 d. 0

2.29 Active-low: b., c., d., and g.; active-high: a., e. and f.

Chapter 3

3.1 Draw an AND-OR circuit with two AND gates and one OR gate. The upper AND gate has inputs of A, $\overline{B}$, and C. The lower AND gate has inputs of A, B, and C. The simplified logic circuit is an AND gate with inputs of A and C.

3.3 The lower input gate

3.5 d

3.7 $Y = \overline{A}\,\overline{C}D + A\overline{B}C + AB\overline{C}$, which means an AND-OR circuit that ORs the foregoing logical products.

3.9 $Y = \overline{A}B + A\overline{B}$, which implies an AND-OR circuit that ORs the foregoing products.

3.11 Figure (a) shows the Karnaugh map.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	1	1
$\overline{A}B$	0	0	1	0
AB	0	1	0	0
$A\overline{B}$	1	1	0	1

(a)

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	1	1
$\overline{A}B$	0	0	1	0
AB	0	1	0	0
$A\overline{B}$	1	1	0	1

(b)

3.13 Fig. 3.16a

3.15 Figure (b) shows the Karnaugh map.

3.17 The simplified equation is

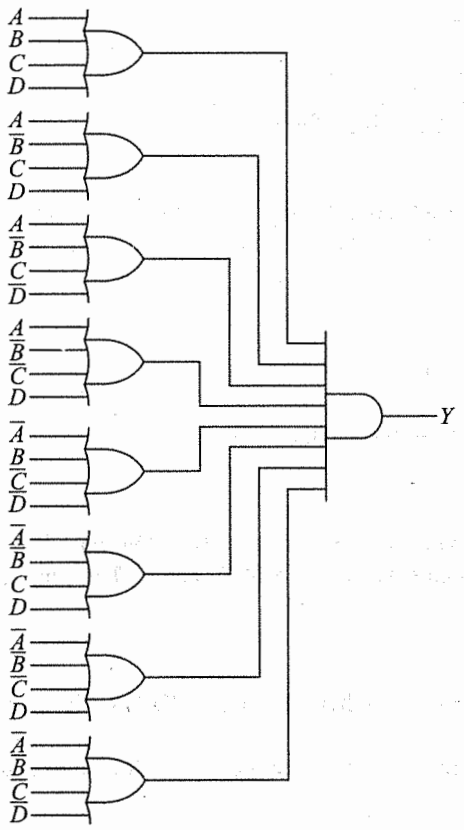
$$Y = \overline{A}\,\overline{B}D + \overline{A}CD + A\overline{B}\,\overline{C} + A\overline{C}D + \overline{B}C\overline{D}$$

The corresponding AND-OR circuit has five AND gates driving an OR gate.

3.19 The simplified logic circuit is an AND gate with inputs of $\overline{A}$, $\overline{C}$, and D.

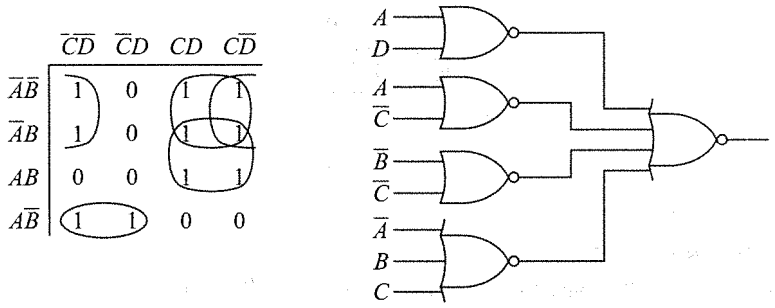
3.21 $Y = AB + AC$; use an AND-OR circuit to produce this equation.

3.23 The unsimplified logic circuit.



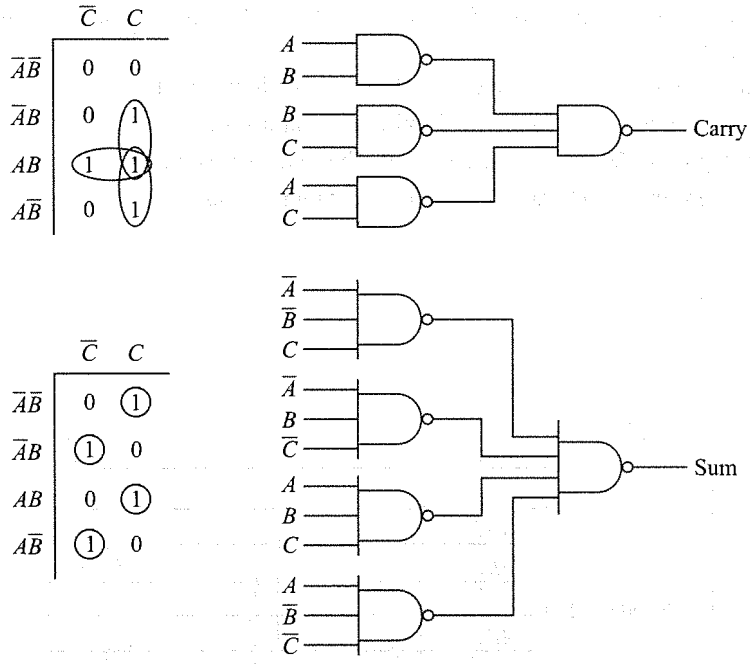
3.25 $Y = F(A, B, C, D) = \Sigma\, m(1, 2, 8, 9, 10, 12, 13, 14)$

3.27 The map and the circuit.



3.29 The Y waveform is low between 0 and 7. Then, it is high between 7 and 16.

3.31 The simplified NAND-NAND circuits.



3.33

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	0	1
$\overline{A}B$	0	0	0	0
AB	1	1	0	1
$A\overline{B}$	1	1	0	1

$Y = (A + \overline{B})(\overline{C} + \overline{D})(A + C + D)$

3.35 $Y = AC' + AD' + B'C'D + B'CD'$

3.37 SOP: $Y = A'B' + AC' + B'C'$ and POS: $Y = (A + B')(A' + C')(B' + C')$

Chapter 4

4.1 Y equals D_9.

4.3 Connect the data inputs as follows: +5 V—D_0, D_4, D_5, D_6, D_{11}, D_{12}, D_{14}, and D_{15}; ground—D_1, D_2, D_3, D_7, D_8, D_9, D_{10}, and D_{13}.

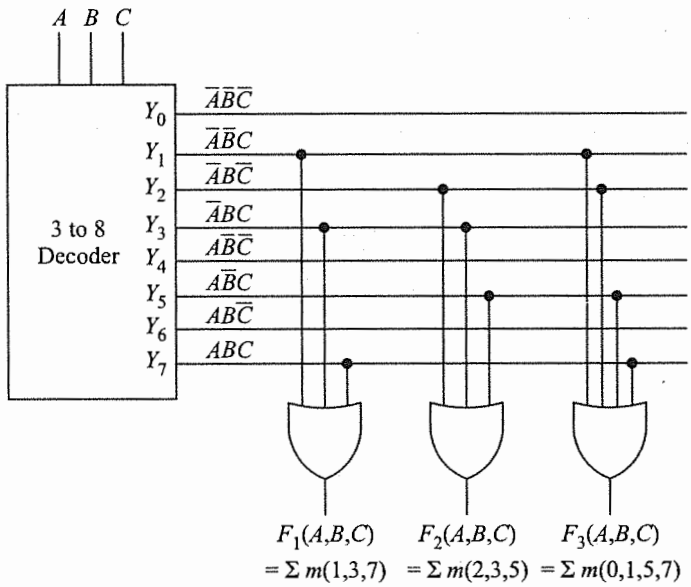
4.5 Y_3 multiplexer: ground D_1, D_6, D_7, and D_{14}; all other data inputs high.
Y_2 multiplexer: ground D_3, D_8, and D_{13}, all other data inputs high.
Y_1 multiplexer: ground D_0, D_1, D_{14}, and D_{15}, all other data inputs high.
Y_0 multiplexer: ground D_8, D_9, and D_{13}, all other data inputs high.

4.7 None; Y_5

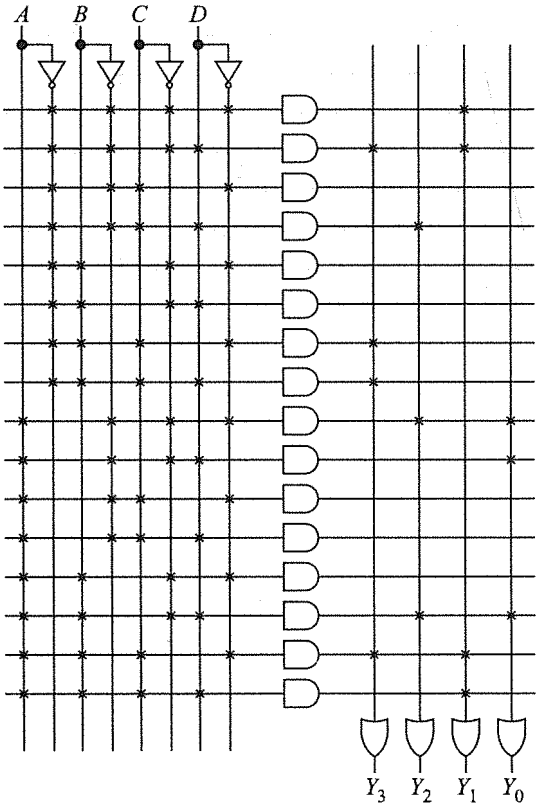
4.9 c

4.11 The chip on the right; Y_6

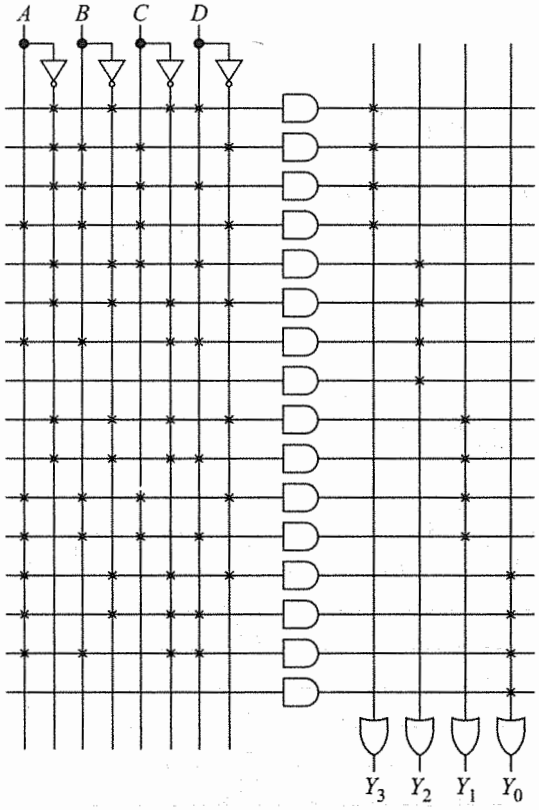
4.13



4.15 a. 67 b. 813 c. 7259
4.17 Y_7
4.19 c
4.21 Approximately 3 mA
4.23 Pin 5; 0111
4.25 a. 0 b. 1 c. 0 d. 1
4.27 a. 0 b. 1 c. 0
4.29 $(X > Y) = G_3 + E_3G_2 + E_3E_2G_1 + E_3E_2E_1G_0$
4.31 Ground pin 4 (after disconnecting from +5 V) and connect pin 3 to +5 V (after disconnecting from ground).
4.33 256
4.35 1100
4.37 The PROM.



4.39 The PAL circuit.



4.41 3; 9; 15 (illegal)

Chapter 5

5.1 01110000

5.3 a. 1 b. 2 c. 3 d. 4

5.5 a. 188 b. 255

5.7 131,072

5.9 100001100

5.11 1010010100000

5.13 011 010 101.111 011 110

5.15 504.771

5.17 a. 257 b. 15.331 c. 123.55

5.19 a. 1110 0101 b. 1011 0100 1101 c. 0111 1010 1111 0100

5.21 12,121

5.23 a. 0000 b. 0100 c. 1010 d. 1111

5.25 a. 011 0111 b. 101 0111 c. 110 0110 d. 111 1001

5.27

Address	Alphanumeric	Hex contents
2000	G	C7
2001	O	4F
2002	O	4F
2003	D	C4
2004	B	C2
2005	Y	D9
2006	E	45

5.29 3694
5.31 0001 0001
5.33 a, b, and d
5.35 e
5.37 11100001111
5.39 Five for both (a) and (b)

Chapter 6

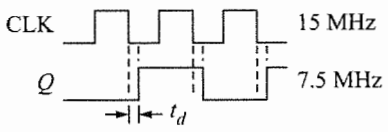
6.1 a. 12_8 b. 13_8 c. 10_{16} d. 17_{16}
6.3 0000 0101 0000 1000
6.5 0001 1000
6.7 a. 0001 0111 b. 0111 1011 c. 1011 1000 d. 1110 1011
6.9 a. DCH b. BAH c. 36H d. O2H
6.11 a. 0100 1110 b. 1110 1001 c. 1010 0110 d. 1000 0111
6.13

a.	b.	c.
0010 1101	0101 1001	0100 0011
+ 0011 1000	+ 1101 1110	+ 1001 1110
0110 0101	0011 0111	1110 0001

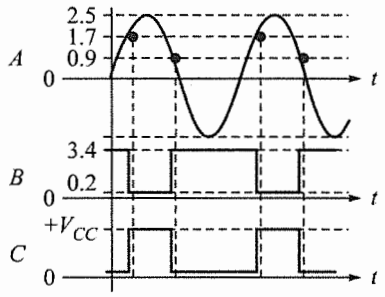
6.15 02CBH, 0000 0010 1100 1011
6.17 Binary 0101 0011, or decimal 83
6.19 $G_0 = 1.1 = 1$, $G_1 = 1.0 = 0$, $G_2 = 0.0 = 0$, $G_3 = 1.1 = 1$.
$P_0 = 1 + 1 = 1$, $P_1 = 1 + 0 = 1$, $P_2 = 0 + 0 = 0$, $P_3 = 1 + 1 = 1$ and $C_{-1} = 0$.
Substituting these in corresponding equations $C_0 = 1$, $C_1 = 1$, $C_2 = 0$, $C_3 = 1$.
Using $S_i = G_i \oplus P_i \oplus C_{i-1}$ $S_0 = 0$, $S_1 = 0$, $S_2 = 1$, $S_3 = 0$.
Final result : $C_3S_3S_2S_1S_0 = 10100$
6.21 Substitute $M = 0$ and $S_3..S_0 = 0110$, $C_{in} = 0$, $A = 1101$ and $B = 0111$.

Chapter 7

7.1 a. 100 ns b. 167 ns c. 1.33 µs
7.3 13.3 MHz
7.5 0.45/4.05
7.7 3.5 MHz plus or minus 28 Hz
7.9



7.11



7.15 48 kHz. $t_1 = 13\ \mu s$, $t_2 = 7.8\ \mu s$

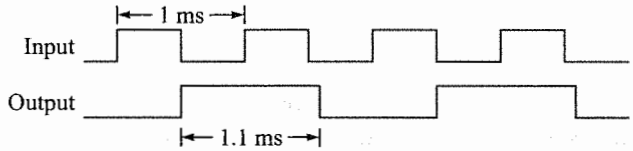
7.17 33.3 percent, 37.5 percent

7.19 $R_A + R_B = 15\ k\Omega$. $R_A = 3.75\ k\Omega$, $R_B = 11.25\ k\Omega$

7.21 3.88 ms

7.23 0.136 μF

7.25



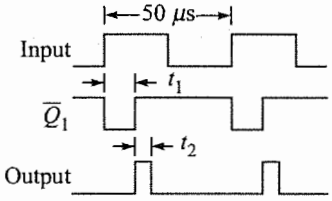
7.27 Connect as in Example 7.8. 21.3 nF.

7.29 a. b.

7.31 Connect $\overline{A}_1$ to GND and apply input to B_2. $C = 44.6$ nF.

7.33 Same as Prob. 7.29.

7.35



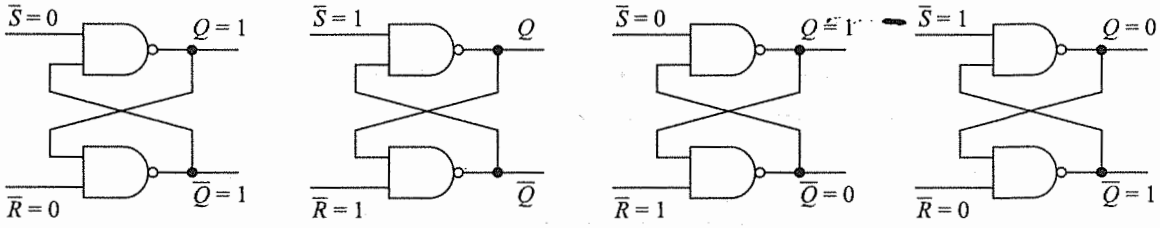
7.37 Let $R_1 = R_2 = 1\ k\Omega$.

a. $t_1 = 2.5\ \mu s$, $t_2 = 7.5\ \mu s$, $C_1 = 7500$ pF, $C_2 = 22{,}500$ pF.

b. $t_1 = t_2 = 1\ \mu s$, $C_1 = C_2 = 3000$ pF.

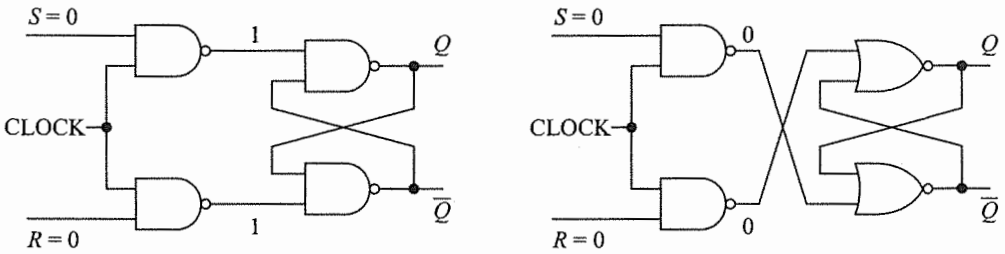
Chapter 8

8.3



8.5 a. C b. G

8.7 When the clock is low, the flip-flop is insensitive to levels on either R or S input. (Only first case is shown here.)



8.9 The R and S inputs do not need to be held static while the clock is high.

8.11 Use negative-edge-triggering.

t_0: S is low, R is high

t_1: S is high, R is low

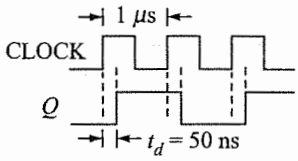
t_2 and after: S is low, R is high.

After t_2, both R and S can be low.

8.13 Low

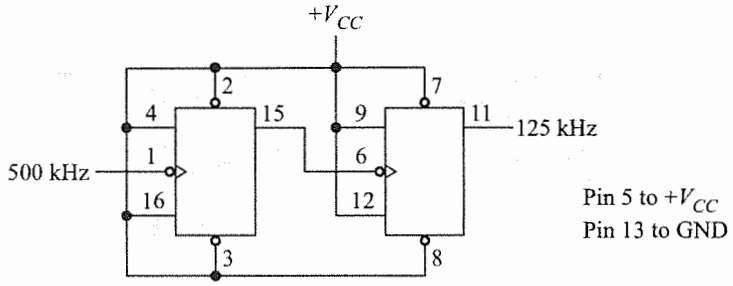
8.15 a. 5 ns b. 10 ns c. 15 ns

8.17



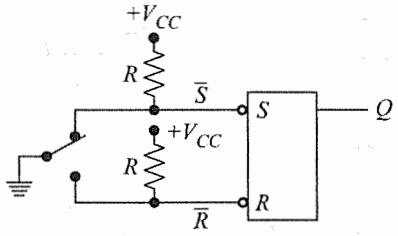
8.19 Clock period = 1 μs. Period of Q = 2 μs (f = 500 kHz)

8.21



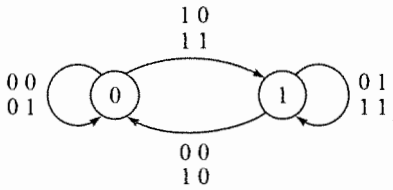
8.23 The pulse symbol shows that the flip-flop is pulse-triggered.

8.25



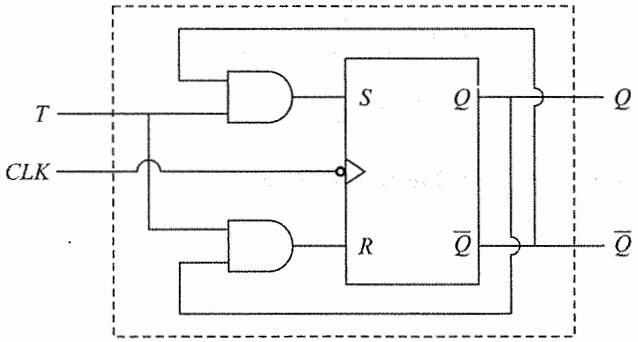
8.27 (a) $Q_{n+1} = BQ_n + AQ_n'$

(b)



8.29 This is a modulo-3 counter with state sequence 00→01→10→00... and corresponding output, Y changes as 1→0→0→1...

8.31

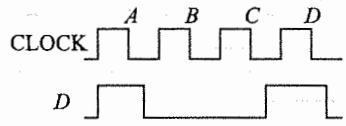


Chapter 9

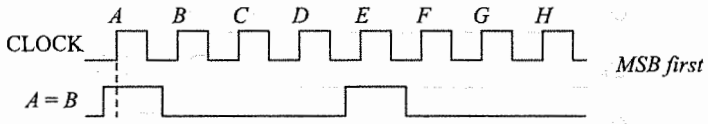
9.1 a. 6 b. 6 c. 4

9.3 See Fig. 9.1

9.5



9.7

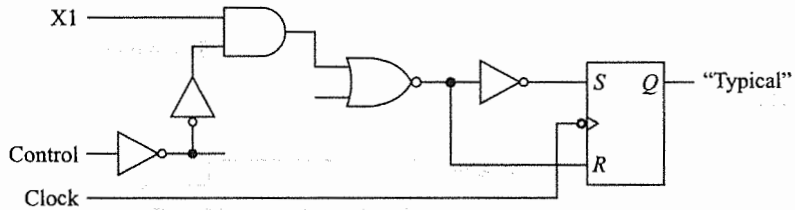


9.9 a. 8 μs b. 1.6 μs

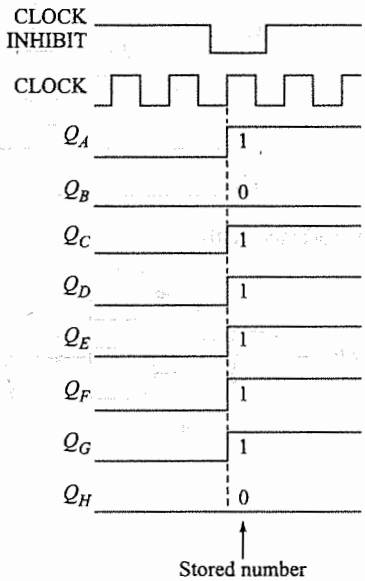
9.11 16.7 MHz

9.13 a. $R = 1, S = 0, Q = 0$ b. $R = 0, S = 1, Q = 1$

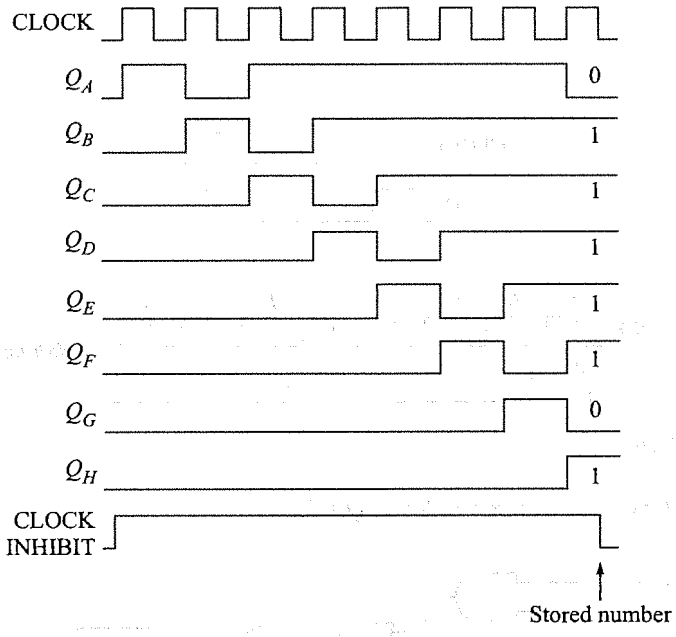
9.15



9.17 MSB first, shift/load is low, *ABCD EFGH* = 1011 1110.

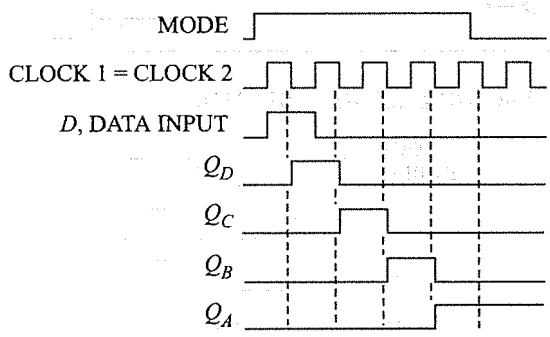


MSB first, shift/load is high.

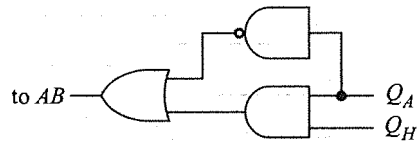


9.19 Same as 9.15.

9.21

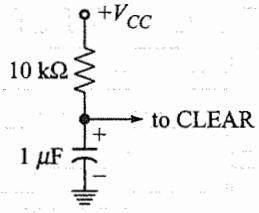


9.23 For alternate 1s and 0s, replace feedback with:



(a)

Include a power-on-CLEAR circuit like:



(b)

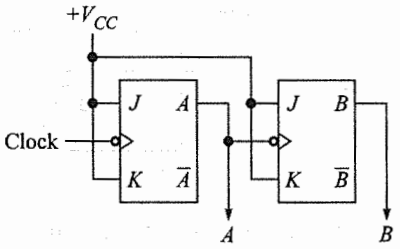
This will CLEAR all flip-flops to zeros when power is first applied.

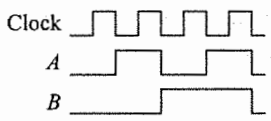
9.25 Decoder output $Y = R'S'$

Clock	*Serial in* = T'	Q	R	S	T	$Y = R'S'$
0	0	1	0	0	1	1
1	1	0	1	0	0	0
2	1	1	0	1	0	0
3	0	1	1	0	1	0
4	1	0	1	1	0	0
5	0	1	0	1	1	0
6	0	0	1	0	1	0
7	1	0	0	1	0	0
8	0	1	0	0	1	1
9	1	0	1	0	0	0
						repeats

Chapter 10

10.1





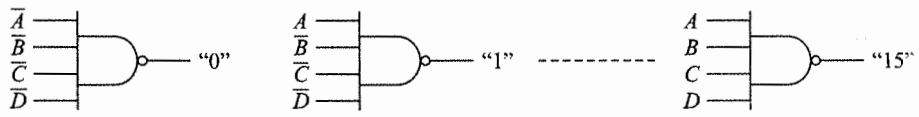
Clock	B	A
0	0	0
1	0	1
2	1	0
3	1	1
0	0	0

10.3 4 MHz

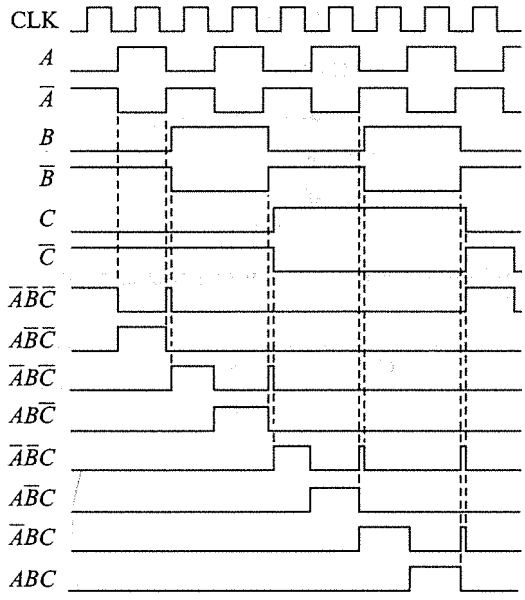
10.5 Difficult to get clock and first few flip-flop outputs on the same page with the last few flip-flop outputs.

10.7 Same as Fig. 10.3 where period of QB is 2 μs.

10.9 Sixteen 4-input NAND-gates with inputs, $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$, $A\overline{B}\,\overline{C}\,\overline{D}$, $\overline{A}B\overline{C}\,\overline{D}$, $AB\overline{C}\,\overline{D}$, $\overline{A}\,\overline{B}C\overline{D}$, $A\overline{B}C\overline{D}$, $\overline{A}BC\overline{D}$, $ABC\overline{D}$, $\overline{A}\,\overline{B}\,\overline{C}D$, $A\overline{B}\,\overline{C}D$, $\overline{A}B\overline{C}D$, $AB\overline{C}D$, $\overline{A}\,\overline{B}CD$, $A\overline{B}CD$, $\overline{A}BCD$, and $ABCD$.



10.11

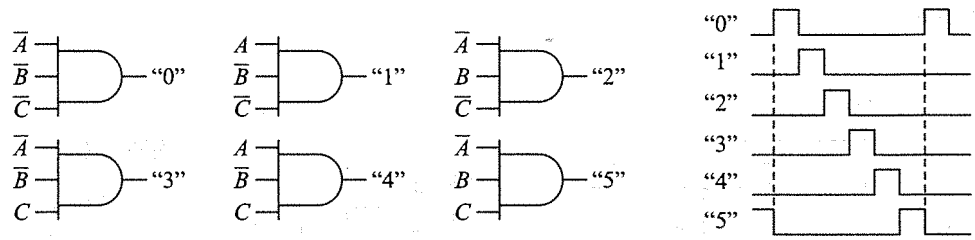


10.13 Same as Fig. 10.12c, except transitions occur on low-to-high clock.

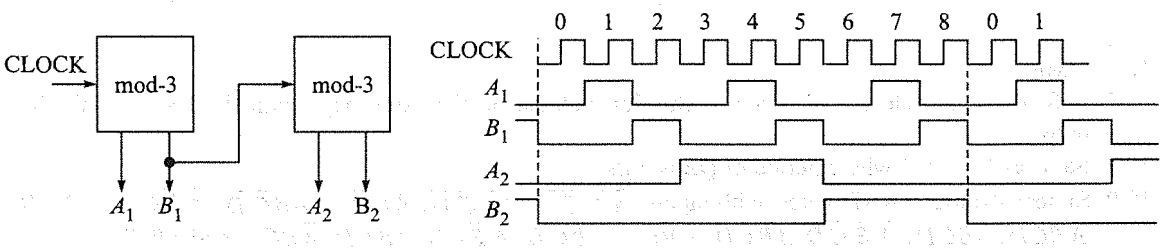
10.15 As in Fig. 10.15.

10.17 a. 3 b. 4 c. 4 d. 5 e. 5

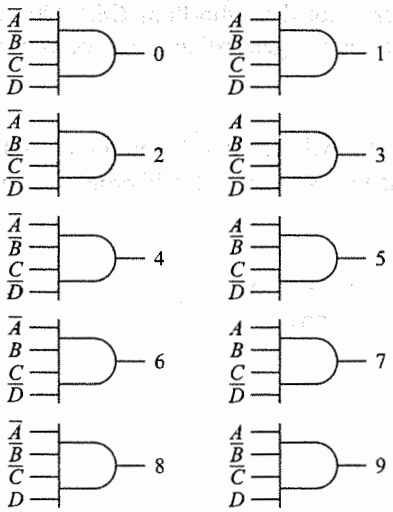
10.19



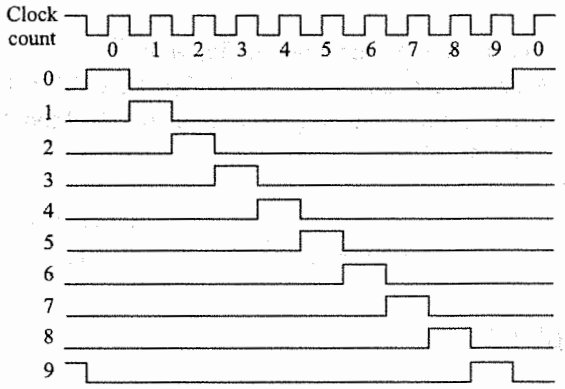
10.21



10.23

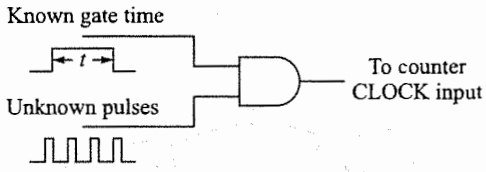


Decoding gates

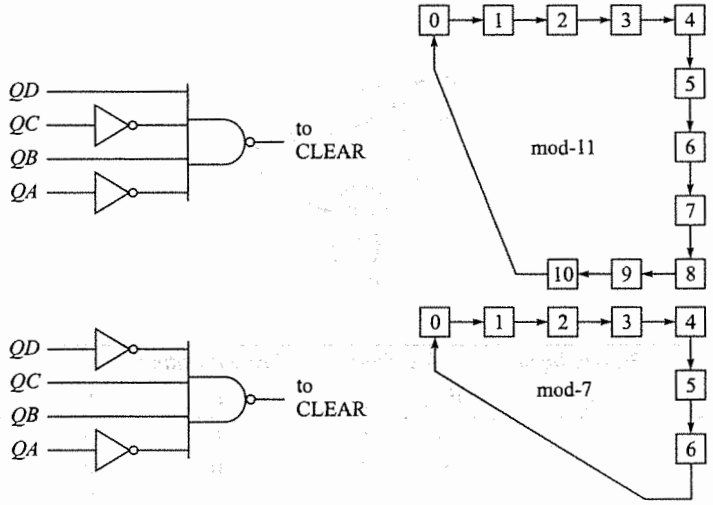


Decode-counter outputs

10.25



10.27



10.29 Like Prob. 10.27.

10.31 Draw the circuit from design equations $D_A = A \otimes B$, $D_B = A' + B$

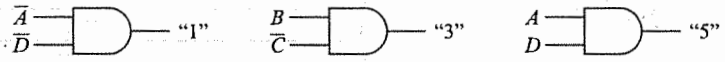
10.33 Draw the circuit from design equations $J_A = B + C'$, $K_A = B' + C'$; $J_B = A$, $K_B = A$; $J_C = AB$, $K_C = A'B'$

10.35 Design a modulo-6 counter (Section 10.7) with state sequence for three flip-flops CBA 000 → 001 → 010 → 011 → 100 → 101 → 000... Then draw circuit for output generating sequence as $Y = C + A'B'$

10.39 Reconnect the J input on flip-flop A to $\overline{D}$.

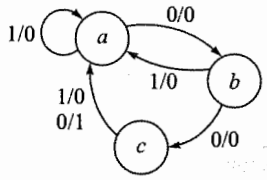
10.41 Mod-5 illegal states are 2(010), 5(101), and 7(111). AND gate will detect $\overline{A}B$ and force counter to $CB\overline{A}$. Count 7 will progress to count 6(110). Mod-3 illegal state is 3(11), which will progress naturally to count 2(10).

10.43

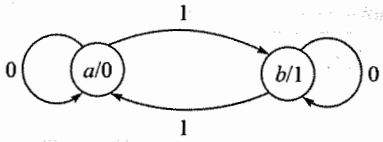


Chapter 11

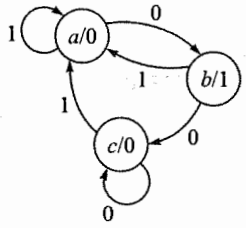
11.1



11.3



11.5



11.7

Present State		*Present Input*	*Next State*		*Present Output*				
B_n	A_n	X_n	B_{n+1}	A_{n+1}	Y_n	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	×
0	1	0	0	0	0	0	×	×	1
0	1	1	1	1	0	1	×	×	0
1	1	0	0	0	1	×	1	×	1
1	1	1	1	1	0	×	0	×	0

11.9

X \ B_nA_n	00	01	11	10
0	0	0	×	×
1	0	1	×	×

$J_B = XA_n$

X \ B_nA_n	00	01	11	10
0	×	×	1	×
1	×	×	0	×

$K_B = \overline{X}$

X \ B_nA_n	00	01	11	10
0	0	×	×	×
1	1	×	×	×

$J_A = X$

X \ B_nA_n	00	01	11	10
0	×	1	1	×
1	×	0	0	×

$K_A = \overline{X}$

X \ B_nA_n	00	01	11	10
0	0	0	1	×
1	0	0	0	×

$Y = XB_n$

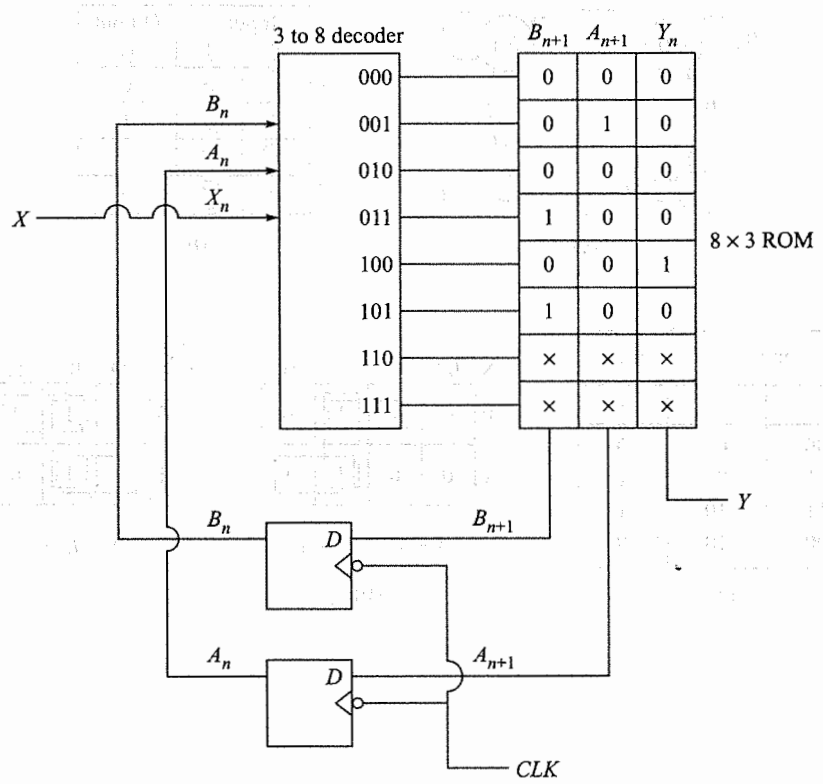
Compared to solution given in Section 11.4, this requires one AND gates less for J_A input.

11.11 Two for Mealy and three for Moore circuit.

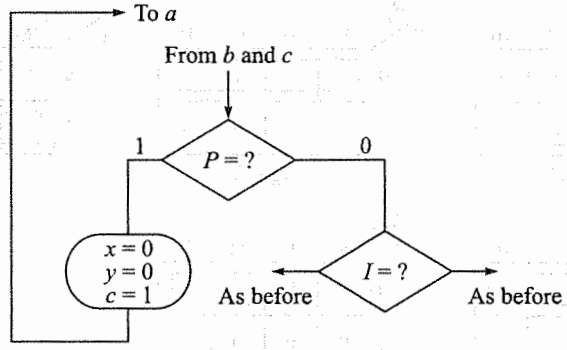
11.13 Two flip-flops (B and A) are required. Three states assigned as (BA) 00, 01, 11 representing no, 1st, 2nd bit detection respectively. Then,

$$D_B = X'B'A \qquad D_A = X'B' \qquad \text{Output } Y = X'BA$$

11.15



11.17



11.19 Three.

11.21 Current output is same as previous output fed back at the input side.

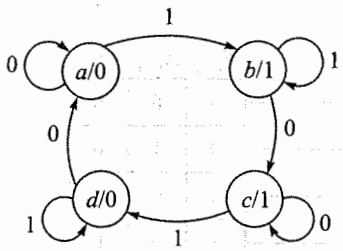
11.23 Stable states $Axy = 010, 100, 101, 111$.

11.25 $ABC = 110, 101, 011$.

11.27 When the circuit moves between two transient states following a particular input transition.

11.29 10110, 11010, 00010.

11.31



(a)

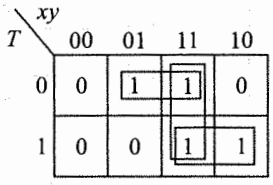
	Input T		Output
	0	1	Q
a	(a)	b	0
b	c	(b)	1
c	(c)	d	1
d	a	(d)	0

(b)

11.33

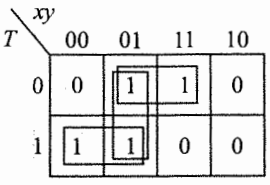
	Input T		Output
xy	0	1	Z
00	00	01	0
01	11	01	1
11	11	10	1
10	00	10	0

(a)



$X = \bar{T}y + Tx + xy$

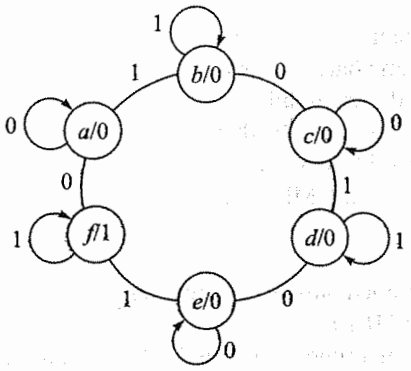
(b)



$Y = T\bar{x} + \bar{T}y + \bar{x}y$

(c)

11.35



11.37

Present state	Input A		Output
(xy)	0	1	Z
a (10)	(a)	b	1
b (00)	–	c	1
c (01)	–	d	0
d (11)	a	(d)	0

Chapter 12

12.1 $\frac{1}{63}, \frac{2}{63}, \frac{4}{63}, \frac{8}{63}, \frac{16}{63}, \frac{32}{63}$

12.5 51.2 mA

12.7 a. 0.641 V b. 0.923 V c. 0.766 V

12.9 1 part in 4096; 2.44 mV

12.11 31

12.13 $A_2A_1A_0$ = a. 001 b. 010 c. 110

12.15 7 MHz

12.17 12 μs, not counting delay and control times.

12.19 $i_R = i_C$, $V_i/R = V_oC/t$, $\therefore V_o = \frac{V_i}{RC} \times t$

12.21 0.01 μF

12.25 They should all be comparable

Chapter 13

13.1 a. RAM b. ROM c. EPROM

13.3 Loss of power results in loss of memory.
13.5 RAM
13.7 ROM data storage is permanent.
13.9 For accuracy check. Read immediately after write.
13.11 Data is recorded and retrieved sequentially.
13.13 One-half tape length = 14,400 in; 14,400/300 = 48 s.
13.15 Maximum 52 × 150 KB/Sec = 7.8 MB/Sec.
13.17 Maximum 8 × 1.32 MB/Sec = 10.56 MB/Sec.
13.19 *DCBA* = 1101
13.21 Five flip-flop binary counter.
13.23 A PROM that can be programmed only by the supplier.
13.25 a. Apply address *FEDCBA* = 110101.
b. Apply a current pulse, one at a time to outputs Y_1, Y_2, Y_6, and Y_8.
13.27 $P = 2F + 1$

F	*P*	*P(Binary)*			
0	1	0	0	0	1
1	3	0	0	1	1
2	5	0	1	0	1
3	7	0	1	1	1
4	9	1	0	0	1

13.29 Two identical chips connected together as follows:
a. Select inputs (*A* to *A*, *B* to *B*, *C* to *C*, *D* to *D*)
b. Data inputs (D_1 to D_1, D_2 to D_2, D_3 to D_3, D_4 to D_4)
c. Sense outputs (S_1 to S_1, S_2 to S_2, S_3 to S_3, S_4 to S_4)
Now, *ME* and *WE* are used to select one chip or the other.
13.31 You must accomplish the following:
READ: a. *ME* goes low for a given time period.
b. The SELECT inputs (address) must be stable while *ME* is low.
Then DATA is valid at the outputs for the time shown in Fig. 13.22d.
WRITE: a. *ME* must be low for a given time period.
b. The select inputs (address) and the data inputs must be stable while *ME* is low.
c. *WE* must go low for a time t_w as in Fig. 13.22c.
13.33 Draw two circuits exactly like Fig. 13.25, one below the other. Now, connect:
a. The ADDRESS lines in parallel
b. The two *ME* lines together
c. The two *WE* lines together
The four DATA IN and DATA OUT lines from the upper circuit can be considered the 4 LSBs, and those from the lower circuit are the 4 MSBs.
13.35 150 ns, 100 ns
13.37 +5 Vdc, −5 Vdc, +12 Vdc
13.39 False
13.41 RAM chips

Chapter 14

14.1 a. 1.82 mA, 0.7 V b. 0 mA, –10 V c. 1.82 mA, 9.3 V
d. 1.65 mA e. 0 mA, 0 V f. 5.35 V

14.3

V_1	V_2
0	+5
+5	0

14.5 a. 1.28 mA b. 0 mA c. 1.28 mA d. –10 Vdc
14.7 a. 1.2 mA b. 0.91 V c. Either case: $I = 0$ mA
14.9 Low-power Schottky; see Table 14.3.
14.11 100
14.13 20
14.15 Change 74LS04 to 7404.
14.17 30
14.19 a. 1 b. 1 c. 0 d. 0
14.21 a. 1 b. 0 c. 0 d. 0
14.23 Time constant = $RC = (3.6\ \text{k}\Omega)(20\ \text{pF}) = 72$ ns
14.25 High, low, high impedance (open)
14.27 Low; high
14.29 High; low
14.31 With high TTL output, $I \approx 0$, ideally with TTL output 0.4 V,

$$t = \frac{12\ \text{V} - 2\ \text{V} - 0.4\ \text{V}}{2.2\ \text{k}\Omega} = 4.36\ \text{mA}$$

14.33 300 ns
14.37 Low; high
14.39 Connecting pin 13 to the supply voltage will produce a permanent high input. This will force the output to stick in the low state, regardless of what values A and B have.
14.41 Grounding pin 1 will force the output to remain permanently in the high state, no matter what the values of A and B.
14.43 3.6 mA
14.45 Yes; 4 maximum
14.47 Time constant = $RC = (2.2\ \text{k}\Omega)(10\ \text{pF}) = 22$ ns
14.49 Choice d, shorted sink transistor

Chapter 15

Solutions for the problems at the end of this chapter are not unique—that is, there are many different designs that will satisfy each requirement. Nevertheless, typical designs for each problem can be found in the literature and are therefore not included here. It is intended that you search application notes and other publications supplied by manufacturers in order to satisfy a particular design requirement. This will provide the opportunity to see numerous different applications, and at the same time challenge you to improve on existing logic configurations. Here are some suggested references:

Intersil, Inc., Cupertino, Calif.: *Data Sheets and Application Notes.*
Motorola Semiconductor Products, Inc., Technical Information Center, Phoenix, Ariz.: *Linear Integrated Circuits Data Book*, 1979.

National Semiconductor Corporation, Santa Clara, Calif.: *Data Acquisition Handbook*, 1978; *Linear Applications Handbook*, 1980.
Texas Instruments, Data Book Marketing, P.O. Box 225558, Dallas, TX 75222-5558: *Interface Circuits Data Book*; *Linear Circuits Data Book* (3 volumes); *Optoelectronics and Image Sensing Data Book; TTL Logic Data Book.*

15.11 909 kHz

15.13 a. 0100 0000 b. 0011 0011 c. 1100 0001

15.15 +1.65 Vdc

15.17 $V_{i-} = +0.25$ Vdc; $V_{ref} = +2.375$ Vdc

Chapter 16

16.1 Five bits are available for data. Maximum number that can be loaded is $2^5-1 = 31$.

16.3 6-bit opcode gives $2^6 = 64$ different instructions and 7-bit opcode gives up to $2^7 = 128$ instructions. Thus we have to assign 7-bit for opcode and in turn need a 7-bit register for *IR*.

16.5 *B* is copied to *A* when any of *X* and *Y* is true.

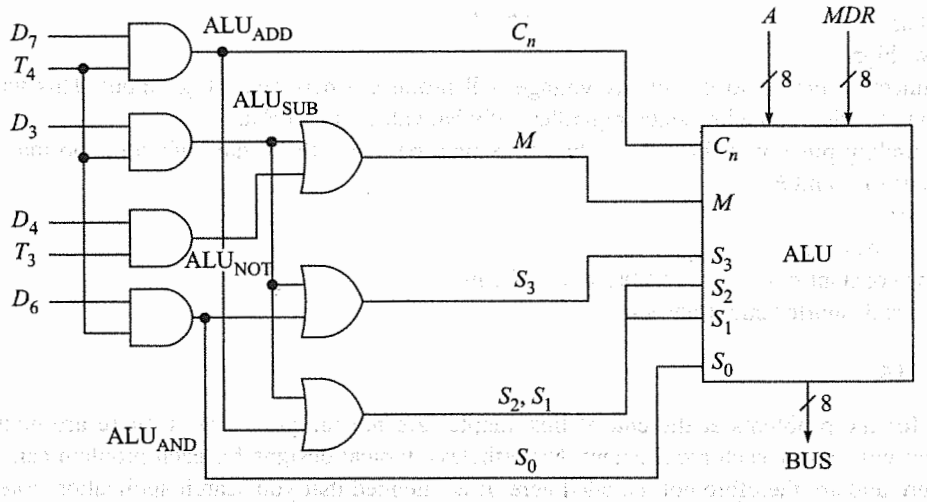
16.7 The bit positions where *ACC* and *MDR* were same will have 1 and rest 0.

16.9 $ACC[7:1] \leftarrow ACC[6:0], ACC[0] \leftarrow CY$

16.11 $D_0T_3 : ACC[4:0] \leftarrow MDR[4:0], ACC[7:5] \leftarrow 0, TC \leftarrow 0$
Parallel load should allow 0 to enter 3 MSB of *ACC* which otherwise receives all 8 bits from BUS. Three 2-to-1 multiplexer with D_0T_3 as selection can be used for this so that $D_0T_3 = 1$ selects 0 and $D_0T_3 = 0$ selects BUS.

16.13 Register *MAR* will be loaded with data available on BUS when TSD generates T_0 or T_2.

16.15



16.17 37 clock cycles.

16.19

Memory address in binary	*Memory content in binary*	*Memory content in hexadecimal*
00000	00001000	08
00001	11001001	C9
00010	00101010	2A
00011	10100000	A0
00100	10100000	A0
00101	11001010	CA
00110	00101010	2A
00111	01000000	40
01000	00000101	05
01001	00001000	08
01010	01000001	41
01011	UNUSED	UNUSED
....	.....	.....
11111	UNUSED	UNUSED

16.21

```
LDA     addr1
NOT
AND     addr2
NOT
STA     addr3
LDA     addr2
NOT
AND     addr1
NOT
AND     addr3
STA     addr3
HLT
```

Index

1'S complement 216
10H8 156
14L4 156
16H2 156
16R8 503
2'S complement arithmetic 220
2'S complement representation 216
22V10 503, 504
24Xx family 496
2716 153
2732 153
2732 495
27XX EPROM 496
28FXXX 496
29FXXX 496
3711 578, 579, 582, 585
4116 501, 502, 507
4164 502
4-Bit counters 368, 369
54/74153 563
54/74155 563
54/74160 368, 369, 373, 382
54/74160 and the 54/74162 369, 373
54/74161 and the 54/74163 368
54/74162, 54/74190, and 54/74192 368
54/74164 310, 314, 315, 321, 325, 331, 332, 337
54/74165 310, 337, 338
54/74166 316, 317, 319, 320, 337
54/74174 320
54/74175 283
54/74176 368
54/74190 and the 54/74192 369
54/74191 356, 357, 368
54/74191 and the 54/74193 368
54/74193 352, 354, 356, 368, 374, 375
54/74198 310, 321
54/7427 273
54/7492A 362, 363
54/7493A 343
54/7495A 321, 322, 323, 337
54/74LS91 309, 336
5400 series 519
54C00 series 541
54LS109 276
54LS11, 348
54LS74 276
555 Timer 253, 256
7400 513, 518, 519, 520, 525, 527, 528, 529, 530, 550, 553
7400 TTL 518
7404 249, 251
74121 569
74121 Nonretriggerable 258, 259
74123 Retriggerable monostables 258

74147 139-141, 169
74150 120, 122, 123, 164
74154 127-133
74155 563, 565
74180 145, 146
74181 231, 234-237, 242
74187 151
74194 308, 324-326
74283 231, 233
74284 237
74285 237
74299 325
7445 134, 135
7446 137, 138
7447 559, 565, 592
7447 Bcd to seven-segment decoder 559
7448 138
7448 559
7450 530
7453 530
7455 530
7473, 7476, and 7478 288
7475 278
7475 563
7476 288, 306, 307
7483 231, 243
7485 147, 148, 161, 162, 167
7488 151
7489 496, 497, 499, 501, 507, 511
7490 367, 382, 390
7490 Decade counter 559, 562
74C00 513, 517, 538, 540, 541, 544, 547, 550, 554, 555
74C00 CMOS 538
74C02 538
74C906 548, 556
74H00 519, 527
74HC00 513, 517, 538, 541, 550, 555
74HC00 devices 541
74L00 519
74LS00 513, 520, 550, 556
74LS189 490
74LS279 274, 275
74LS73a 288
74LS76a 288
74LS78a 288
74LS91 308, 309, 312-314, 336
74S00 520, 550, 553
74S188 152
74S201 487, 491, 499-501, 511
74S287 152
74S288 492, 493, 496
74S370 151
74S472 152
74S89 490, 509

A
A/D accuracy and resolution 471
A/D converter 438, 455-462, 464, 465, 467-475
A/D converter–counter method 458
A/D converter-tracking type 462
Access time 481, 494, 495, 502, 508, 510, 511
Accumulator 595
Act1 505
Act2 505
Act3 505
Active load 517, 518, 551, 556
Active
  low 17
  power dissipation 540, 551, 554
  pull-up 532
  low signals 60, 68
ADC0804 465-467, 475, 558, 571, 573-576, 592
ADC3511 558, 577-579, 582-585, 587, 592
ADC3711 582
ADD3501 558, 585, 587
Adder-subtracter 228
Addition 19
Address 478, 479, 487-492, 494-501, 503, 506-511
Address bus 596, 612
Address decoding 488
Addresses 148, 149, 151, 164
Address-hold 499
Address-setup or select-setup time 499

ALU 595, 596, 598, 602, 603, 605, 613, 614
Analog 1-3, 35, 36
Analog signals 3, 4
And and OR gates 528
AND gate 1, 11-13, 27, 29, 31, 35, 37, 38
And gates 44, 45
And-or 40, 46-48, 51, 55-59, 68, 69
And-or-invert 40, 51, 57, 58, 59
And-or-invert gates 40, 57, 58, 529
Arithmetic logic unit 18, 19, 235, 595
ASCII code 190
Assertion-level logic 40, 60, 61, 68, 72
Associative laws 75
Astable multivibrator 253
Asynchronous
  counters 342
  inputs 282, 299
  operation 245
  sequential circuit 413, 414
  sequential logic 392, 420, 422, 423, 434, 435

B

Base 514, 515, 518, 520, 532, 533, 536, 550, 551, 552, 556
BCD-2421 178, 179
BCD-8421 178, 179
BCD-to-decimal decoder 134
Behavioral modeling 109
Bidirectional data bus 15
Binary
  addition 207
  digit 14, 35
  equivalent weight 439, 473
  ladder 438, 442-444, 458, 459, 472, 473
  multiplication and division 237
  number system 3, 171, 179, 199
  subtraction 211
  weights 173
Binary-coded decimal 133, 163, 169
Binary-to-decimal conversion 173
Binary-to-hexadecimal conversion 184
Binary-to-octal conversion 182
Bipolar 8, 28, 35, 36
Bipolar junction transistors (bjts) 513, 514
Bistable 271, 278, 279, 303, 304
Bit 14, 15, 17, 19-21, 35-38, 172, 173, 176, 178, 179, 184, 190-192, 194, 196-200, 202, 203, 205
Boolean algebra 41, 68, 74-76, 80, 93, 98, 100, 110-113
Bubbled and gate 49, 53
Bubbled or gate 53
Buffer 1, 8-12, 35
  amplifier 9
  drivers 528
Bus 82, 83, 99, 596-598, 602-605, 609, 613
  organization 533
  selector 596
Byte 172, 173, 186, 187, 189, 196, 200, 594, 595, 598, 600, 605

C

Cache memory 477, 479, 503
Canonical
  product form 96
  sum form 82
Carry 226-231, 243
  flag 213, 214, 595
  look ahead adder 232
CD ROM 483-485
CD-R 484, 485, 508
CD-RW 485, 486, 508, 509, 511
Cell selection 486
Centering on the LSB 463
Central processing unit (CPU) 23, 593, 594, 609, 612
Characteristic equations 290
Checksum code 196
Chip 26-28, 32, 35, 477
  expansion 132
  enable 478, 492, 499, 508
  select 478, 492, 495, 609
  select or chip-enable 478
Circuit excitation table 396
Clear 282-284, 306, 307
Clock 6, 7, 17, 18, 21, 24-26, 35, 38
Clock
  cycle time 245, 247, 266
  oscillator 565

signal 7, 17, 18, 24
Clocked
D flip-flops 277
RS flip-flops 276
CMOS 29, 30, 32-35, 39, 512, 513, 517, 538-548, 550-552, 554-557
CMOS
buffer 547
characteristics 541
level shifter 546
load 543-546
logic levels 33
CMOS-to-TTL interface 546
Collector 512, 514, 518, 523, 529-532, 536-538, 545, 546, 548, 550, 551, 553, 555
Combining rules 78
Commutative laws 75
Compact disk (CD) 483
Comparator drive 536
Comparison 20
Compatibility 523, 543
Complementary metal-oxide semiconductor (CMOS) 512
Computer architecture 602, 605, 609, 612
Consensus theorem 78, 112
Contact bounce 270, 289, 290, 305, 307
Content addressable memory (CAM) 506
Continuous A/D conversion 461
Continuous-type A/D converter 462, 464
Control
bus 596, 602, 613
path 596, 597, 602, 613
unit 602
Controlled inverter 228-231, 241, 242
Controller 25, 26
Conversion between canonical forms 97
Conversion of flip-flops 296
Conversion of models 395
Count-down
line 463
mode 342, 345, 346, 348, 351, 352, 355-357, 368, 369, 388, 389
Counter 6, 17, 18, 21, 35, 37
design 376, 378, 384
modulus 357
Count-up
line 463
mode 341, 343, 348, 350, 351, 355, 356, 368, 369, 389
Covering rule 78
CPLD 503, 504
Critical race 417
Current
sink 4, 35
source 4, 35
tracer 548, 549, 556
Cycle redundancy code 196, 205

D

D/A accuracy and resolution 454
D/A converter 438, 447-454, 458, 459, 467, 472, 473
D/A converters 447
DAC0808 452, 455
Data 2, 14, 15, 16, 18-20, 22-25, 30, 32, 35-39
bus 15, 24, 596, 609, 610, 613
memory 594, 613
path 596, 602, 612, 613
selector 118, 163
Dataflow modeling 108
Data-hold time 499
Data-setup time 498
De morgan's first theorem 49
De morgan's second theorem 54
De morgan's theorems 77
Debounce circuit 290
Decade counter 365, 369
Decimal-to-BCD encoder 139, 163
Decimal-to-binary conversion 176
Decimal-to-hexadecimal conversion 185
Decimal-to-octal conversion 181
Decode cycle 601
Decoder 21, 35, 37, 118, 127, 130-135, 137, 138, 149, 158, 160-163, 165, 169, 488-490, 506
Decoding gate 346, 349, 350, 357, 388
Delay time 497, 498, 508

Demorgan's theorem 611
Demultiplexer 20, 21, 35, 118, 127-131, 160, 163
Describe characteristic equations of flip-flops and analysis techniques of sequential circuit 270
Describe excitation table of flip-flops and explain conversion of flip-flops as synthesis example 270
Differential linearity 472
Digital 1-15, 17-30, 32, 34-38
Digital
  computer 1, 14, 22-24, 26, 35, 36
  panel meter 585
  signals 3
  versatile disk (DVD) 483
  voltmeters 585
Diode 513
Direct
  addressing 594
  memory access (DMA) 24
Distributive law 76, 79, 80, 110
Divider circuit 565
Don't-care condition 93, 95, 112, 117
Down counter. 345, 351, 356, 386
Dram cell 502, 503
Drams 477, 501
Dual-inline package (DIP) 27
Duality 77, 100
Duality theorem 77, 78, 100
Dual-slope A/D conversion 467, 469
Duty cycle 7, 35, 36, 38
DVD 483, 485, 486, 508, 509
Dynamic
  hazard 107
  input indicator 248, 249, 266, 269
  ram (dram) 477, 496

E

Ebcdic 191
Edge-triggered D flip-flops 281
Edge-triggered flip-flop 279, 285, 304, 305, 307
Edge-triggered JK flip-flops 283
Edge-triggered RS flip-flops 279
Eprom, flash memory 495
Eight to fourteen modulation (EFM) 483
Emitter 514, 515, 518, 519, 521-523, 528-530, 532, 536, 550-552, 556
Emitter-coupled logic 28, 29
Encoder 20, 21, 35, 118, 138-141, 163, 169
Encoding device 438
Entered variable map 85, 86, 92, 104
Eproms 478, 491, 495, 505, 507
Erasable PROM (EPROM) 153, 478
Error detection and correction 171, 196, 198
Essential prime implicants 102-104, 111
Even parity 143
Excess-3 code 171, 192, 199, 202
Excitation table 292
Exclusive-or gate 118, 141-144, 163, 170
Execute cycle 601
Execution of instructions 599
Expandable and-or-invert 530
Expandable and-or-invert gate 58, 59
Expandable memory 490
Expander 58, 59

F

Fall time 6, 35, 36, 246, 266
Falling edge 245, 247, 248
Fan-in 233
Fanout 33, 34, 526, 527, 529, 543, 544, 547, 548, 550, 551, 553
Fast adder 232
Fetch cycle 600
Field-programmable 157, 491, 508
Finite state machine 291, 292
Flash converter 456, 458
FLEX10000 505
FLEX8000 505
Flip-flop 13, 14, 17, 18, 35, 37, 270-299, 301-307
Floating inputs 521, 541
Floppy disk 479, 482, 483
Forbidden region 32, 38
Forward-biased 513, 514, 556
FPGA 503-505, 507
Fractions 174, 177, 181
Free-running mode. 465
Frequency 6, 7, 24, 38

Frequency counters 565
Frequency stability 246, 265, 266
Full-adder 226, 227
Fundamental mode 413, 416
Fuse link 491, 511
Fusible link 152, 153

G
Gate 40-61, 63-73
Gate, source, drain, and body 515
Glitch 348-350, 388
Glitches 262, 269
Gray code 193

H
Half-adder 226
Hamming code 197
Handshaking 25, 35, 37
Hard disk 16, 479, 481-483
Hardware description language 62
Hazard covers 105
Hazards 74, 105, 107, 112, 418
HDL 40, 61-63, 108-110, 237, 298, 333, 423, 425, 503
Hexadecimal numbers 183, 199
Hexadecimal-to-binary conversion 184
Hexadecimal-to-decimal conversion 185
High-speed ttl 519
Hit flag 507
Hold time 285
Hysteresis 251, 253, 265

I
IC 74181 234-237, 242
IC 74182 234
IC 74283 233
IC families 28
IEEE 30-32, 38
IEEE symbol 30, 31
Immediate addressing 594, 613, 614
Implication table method 411
Indirect addressing 594, 597, 614
Inhibit mode, 355
Instruction
  cycles 600
  decoder 596
  register 595
Integrated circuit (IC) 1, 8
Intel 2164 502
Interfacing 30, 34
Interrupt 609, 610, 613
Interrupt service routine (ISR) 610
Inversion 10, 18
Inverter 1, 10, 27, 29-32, 35, 37, 38
Inverter (not gate) 41
Irregular counter 377-379

J
Johnson counter 328, 339, 340

K
Karnaugh map 74, 84-87, 89, 90, 92-95, 98, 100-105, 111-114, 116, 117, 290, 291, 297

L
Ladder, 442, 447, 454, 471
Large-scale integration (LSI) IC 28
Laser (light amplification by stimulated emission of radiation) 483
Latch 271, 274-276, 278, 279, 281, 286, 290, 298, 303-305, 307
Least-significant bit (LSB) 209
Led 514, 518, 537, 538, 552, 554, 556
Light-emitting 3
Linear
  addressing 487, 490
  operation 2
LM336 582, 587
Locality 503
Lock out 286, 287
Logic
  circuits 40, 41, 50, 51, 60, 68
  clip 80, 112, 113
  probe 158, 159, 163, 165, 167
  symbols 30
Logical operations 3, 24, 29

Look-up table 148
Low-power schottky TTL 520
Low-power TTL 519
LSB, 439, 441, 464

M

Mach chips 504
Macro operations 595, 597, 599, 600, 602, 607, 612, 613
Macrocell 503, 504
Magnetic
  memory 479
  tape 479, 480, 483
Magnitude comparator 146, 147, 163
Mask 151, 163, 164
Maskable interrupts 610
Mask-programmable 491, 492, 496, 508, 510
Master clock 595, 600, 602, 603
Master-slave flip-flop 270, 288, 289
Matrix addressing 487
Max 5000 504
Max 7000 504
Max 9000 504
MC10319 458
MC1508/1408 452
MC6108 465
MCM6665 502
Mealy model 392-398, 400-405, 407-409, 416, 424, 432, 433
Medium-scale integration (MSI) ICS 28
Memory 13, 16, 17, 24-26, 28, 35, 36, 38, 39, 270, 271, 291, 293, 303
Memory address register 595
Memory addressing 486
Memory bank 500
Memory cells 477, 478, 487, 489, 494, 495, 498, 507, 509
Memory data register 595
Memory-enable 496, 499
Metal-oxide-semiconductor (MOS) 28
Metal-oxide-semiconductor field-effect transistors (MOSFETs) 513
Micro operations 595-599, 602, 608, 612-614
Microprocessor 14, 15, 17, 25, 26, 28, 36-38
Millman's theorem 440, 441, 473
Minterms 81, 82, 84, 92, 103
MM74C925 565, 568, 592
Mod-10 counter 365
Mod-3 counter 360
Mod-5 counter 363
Module body 63
Modulo-6 counter, 376
Modulus 341, 343, 357, 359-363, 365, 368, 369, 376, 387, 388
Monostable 244, 253, 256-262, 264-268
Monostable multivibrator 256
Monotonicity test. 450
Moore model 392-399, 401-404, 408, 409, 419, 421, 432, 433, 436
MOSFETs 512, 513, 515-517, 540, 550, 551, 557
Most-significant bit (MSB) 209
MSB 439, 441-444, 446, 447, 452, 464, 473
MSBs 602, 608, 609
Multiplexer 20, 21, 36, 118-127, 129, 154, 159, 160, 163, 164, 169
Multiplexing displays 559

N

Nand gates 53, 55, 528, 538
Nand-gate latch 274
Nand-nand network 55
N-channel transistor 516
Negation 10, 218, 222
Negative logic 3, 4, 36, 40, 41, 59, 60, 61, 68
Negative transition 245
Negative-edge-triggered 245, 248, 249, 256
Negative-edge-triggered 280
Nibble 172, 173, 200
  multiplexers 124
NMOs 29
Noise
  immunity 524, 525, 535, 540, 542, 543, 545, 546, 550, 551
  margin 2, 34, 36
  voltage 34
Noisy signals 251
Nonlinear operation 2

Non-maskable interrupts 610
Nonretriggerable 244, 258, 259, 261
Nonvolatile data storage 479
NOR gates 48, 528
NOR-gate latch 271
NOR-nor network 51
NOT circuit 10
NPN transistor 514
NSB5388 587

O
Octal number system 179, 180, 183, 199
Octal-to-binary conversion 181
Octal-to-decimal conversion 180
Octet 88-91, 112
Odd parity 143
On-chip decoding 149, 150
Opcode 594-597, 599-603, 607, 608, 613, 614
Open circuit 9
Open-collector gates 530
Open-collector output 530, 536, 546
Open-drain devices 548
Operational amplifier 446, 449, 467
Operational amplifier drive 535
Optical memory 16, 483
OR gate 1, 12, 13, 29, 31, 32, 35, 37, 38
OR gates 42, 43
OR-and 46, 47, 51, 68, 69
Oscillation 417
Overflow 213-215, 222, 224, 225, 240-243

P
Packing density 477, 502, 508
Parallel register 14
Parallel adder 232-234, 242
Parallel in–parallel out 320
Parallel in–serial out 316
Parallel shifting 309
Parity bit 191
Parity checker 144
Parity code 196
Parity generator 144, 145
Pasic 505
pASIC2 505
Passive pull-up 532, 546, 551
P-channel transistor 516
Period 2, 6, 7, 35
PLD 503, 504
PMOs 29
PNP 514, 552, 556
Pogrammable rom 478
Port 16, 17, 20, 36-38
Ports 62
Positive and negative gates 59
Positive logic 3, 4, 41, 59, 68, 41, 59, 68, 73
Positive transition 245, 262, 264
Positive-edge-triggered 245, 247-249, 279-285, 305
Postponed output 288
Power dissipation 519, 540
Preset 282, 284, 306, 307
Presettable counters 368
Prime implicants 102-104, 111
Primitive table 420
Primitives 63
Priority encoder 140, 141
Product-of-sums equations 47
Product-of-sums method 95
Program 24, 26, 36
  counter 595
  execution 607
  memory 594-596, 600, 601, 612, 613
Programmable array logic (PAL) 154
  logic arrays (PLAS) 156
  rom (PROM) 152
  sequence detector 330
Programming computer 605
PROMs 478, 491, 492, 507
Propagation delay 244, 247, 285, 303, 305, 519, 539
Propagation delay time 497, 519, 539
Pull-up resistor 521, 530-532, 535, 544-546, 550, 555, 556
Pulse mode 413

Pulse-forming circuits 247, 262
Pulse-triggered 288, 289, 305, 307

Q
Quad 31, 32, 80, 88-91, 93, 94, 112
Quantization error 471-474
Quine-McClusky method 102

R
Ram 476-479, 481, 483, 486, 487, 496, 497, 499, 500, 502, 506-509, 609, 610
Ramp generator 467
RAMs 481, 496, 499, 507
Random access 478, 479, 509
Read operation 478, 485, 497, 498, 508
Read-only memory 148, 163, 164, 170, 392, 400, 432
Read-write head 480, 482, 483, 509
Redundant group 91, 92, 112
Refresh cycle 501, 502, 508
Register
  array 595
  transfer language 597
Registers 14
Relay 8
Resistive divider 440
Retriggerable 244, 258-261, 265
Reverse-biased 513, 518
Ring counter 325, 327-329, 332-334, 336, 337
Ripple carry addition 232, 238
Ripple counter 341-349, 351, 388-390
Rise time 6, 35, 36, 246, 266, 269
Rising edge 245
ROM 118, 148-153, 163, 164, 167, 170, 476-479, 483-486, 490-492, 495-497, 508-511, 609, 610
ROMs 491, 492, 507
Row elimination method 410
RS flip-flops 271, 279
RTL 593, 597-599, 601, 602

S
Sample and hold circuit 449
Sampled 3
Saturation delay time 520, 550, 551
Schmitt trigger 244, 250-253, 261, 265-267
Schmitt triggers 244, 253, 269
Schmitt-trigger inverter 251
Schmitt-trigger NAND gate 251
Schottky transistor 29, 36
Schottky TTL 520
Search
  data register 507
  lines 507
  word 507
Section counters 466
Select-hold time 499
Semitransparent 279
Sequence
  detector 329
  generator 329
Sequential circuits 293
Serial
  adder 330
  addition 232, 241
Serial in–parallel out 313
Serial in–serial out 310
Serial
  register 15-17, 37, 38
  shifting 309
Settling time 341
Setup time 285, 286, 303, 305
Seven-segment decoder-driver 137
Seven-segment decoders 136
Shift register 308-316, 319-321, 323-326, 328-331, 333-340
Sign-magnitude numbers 214
Simulation 62, 64-66
Simultaneous A/D converter 456-458, 474
Single-ramp A/D converter 467
Small-scale integration (SSI) IC 28
Solder bridge 159
Sourcing and sinking 523, 543
Srams 477, 496, 499, 501
Stable 393, 406, 414-418, 420, 421, 432, 434, 437
Start/stop flag 595
State 270-274, 276-289, 291, 292, 294-297, 301, 303, 306, 307