b) De Morgan's Theorem

1) $\overline{AB} = \overline{A} + \overline{B}$

2) $\overline{A + B} = \overline{A} \cdot \overline{B}$

Problem: Verify all the laws with the help of truth table.

6.5 Introduction to half adder and full adder

Digital computers perform various arithmetic operations. The most basic operation is the addition of two binary digits. The four elementary operations are,

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10_2$

The first three operations produce a sum whose length is one digit, but when the last operation is performed sum is two digits.

The higher significant bit of this result is called CARRY and lower significant bit is called SUM.

The logic circuit which performs this operation is called a half-adder and the circuit which performs addition of three bits is a full-adder.

6.5.1 Half - Adder

The half-adder operation needs two binary inputs, augend and add end bits and two binary outputs sum and carry.

a) Block Schematic of Half Adder

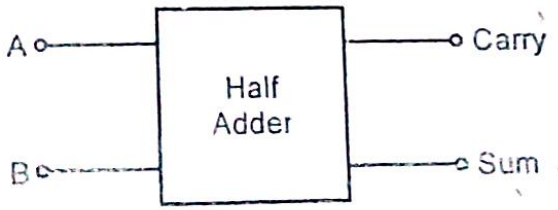


Figure 6.13

b) Truth Table

Inputs		*Outputs*	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

c) Logic Diagram

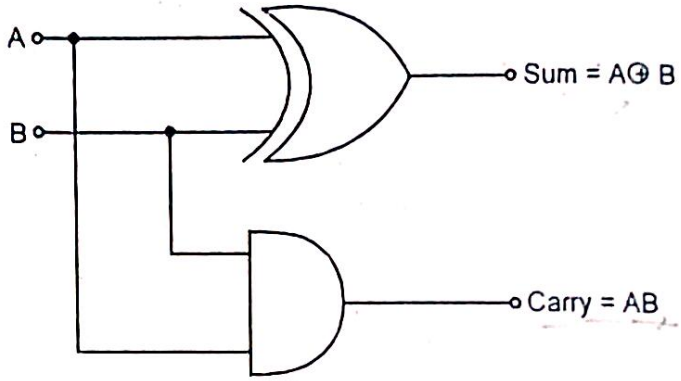


Figure 6.14

6.5.2 Full - Adder

A half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multi bit addition is performed. For this purpose, a full adder is designed.

A full adder is a combinational circuit that performs the arithmetic sum of the input bits and produces a sum output and a carry.

a) Block Schematic of Full Adder

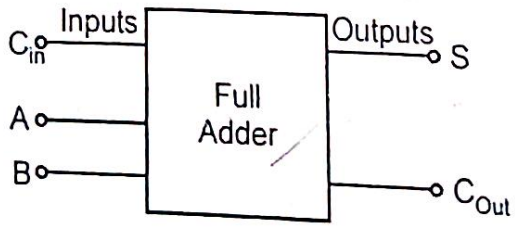


Figure 6.15

b) Truth Table

Inputs			*Outputs*	
Augend bit A	Addent bit B	Carry bit C_{in}	Sum S	Carry output C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

c) Logic Diagram

Figure 6.16 shows logic diagram of full adder.

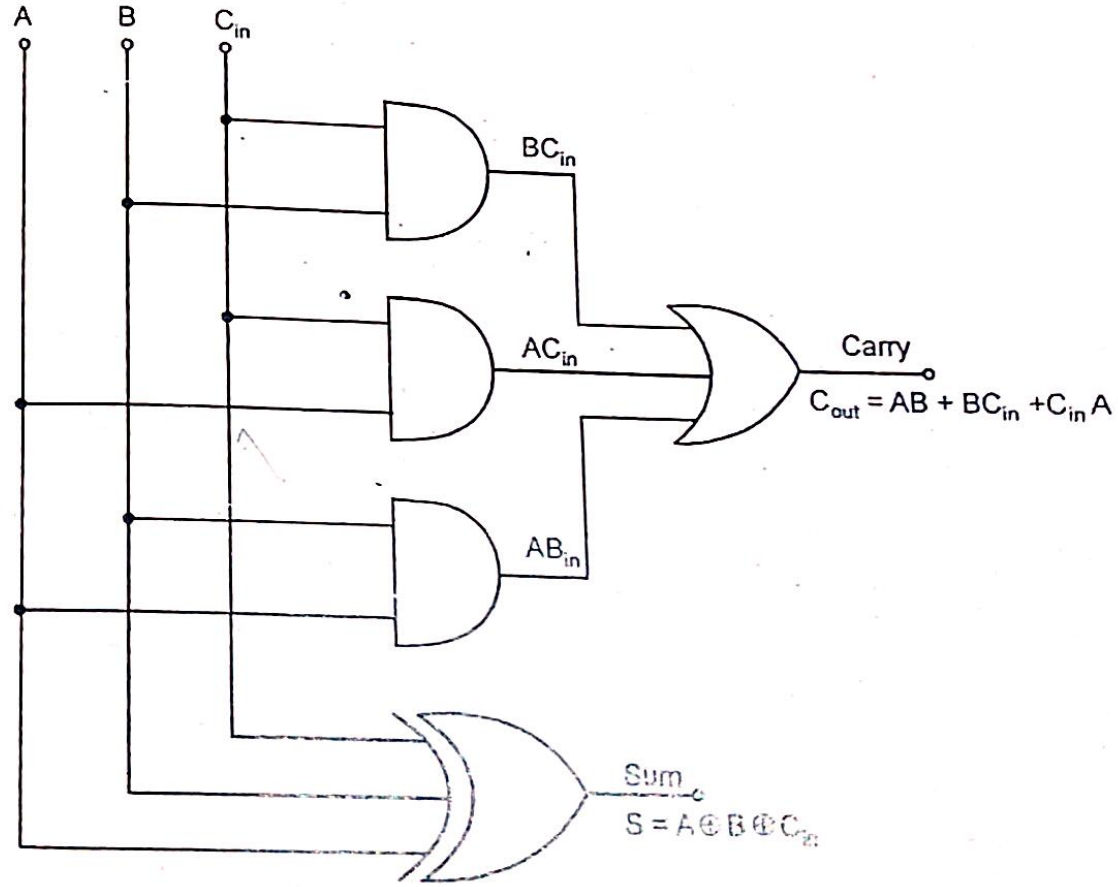


Figure 6.16

The third input, C_{in} represents the carry from the previous lower significant position. The binary variable S gives the value of the LSB of the sum and the binary variable C_{out} gives the output carry.

A full adder can be formed using two half adder circuits and OR gate as shown in figure 6.17.

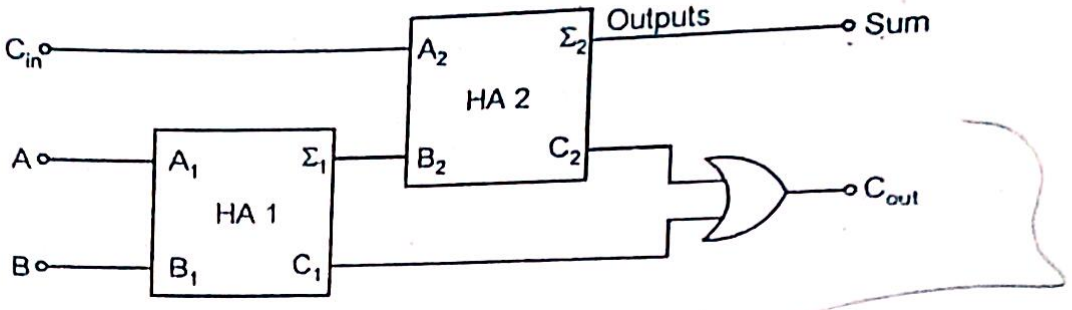


Figure 6.17

6.6 Flip Flops

The output levels of the combinational logic circuits at any instant of time are dependent on the levels present at the inputs at that time. Since combinational logic circuits have no memory, the prior input level conditiions have no effect on the present outputs. Most digital systems are made up of both combinational circuits and memory elements.

Figure 6.18 shows a block diagram of general digital system that combines combinational logic gates with memory devices.

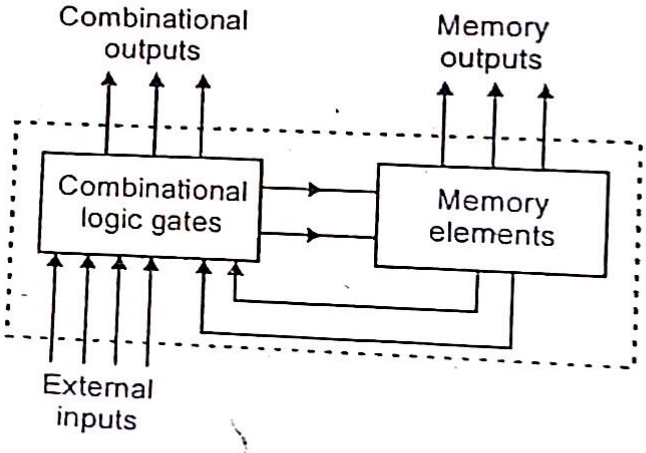


Figure 6.18

The combinational portion accepts logic signals from external inputs and from the outputs of the memory elements. The combinational circuit operates on these inputs and produce various outputs. Some of these outputs are used to determine the binary values to be stored in the memory elements. The outputs of some of the memory elements, in turn, go to the inputs of logic gates in the combinational circuits. This shows that the

4.1 MULTIPLEXERS

Multiplex means *many into one*. A *multiplexer* is a circuit with many inputs but only one output. By applying control signals, we can steer any input to the output. Thus it is also called a *data selector* and control inputs are termed select inputs. Figure 4.1a illustrates the general idea. The circuit has n input signals, m control signals and 1 output signal. Note that, m control signals can select at the most 2^m input signals thus $n \leq 2^m$. The circuit diagram of a 4-to-1 multiplexer is shown in Fig. 4.1c and its truth table in Fig. 4.1b. Depending on control inputs A, B one of the four inputs D_0 to D_3 is steered to output Y.

Let us write the logic equation of this circuit. Clearly, it will give a SOP representation, each AND gate generating a product term, which finally are summed by OR gate. Thus,

$$Y = A'B'.D_0 + A'B.D_1 + AB'.D_2 + AB.D_3$$

If $A = 0$, $B = 0$,

$$Y = 0'0'.D_0 + 0'.0.D_1 + 0.0'.D_2 + 0.0.D_3$$

or,

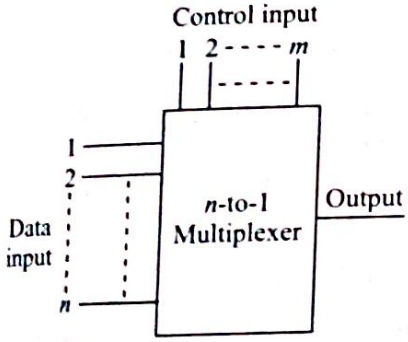
$$Y = 1.1.D_0 + 1.0.D_1 + 0.1.D_2 + 0.0.D_3$$

or,

$$Y = D_0$$

In other words, for $AB = 00$, the first AND gate to which D_0 is connected remains active and equal to D_0 and all other AND gate are inactive with output held at logic 0. Thus, multiplexer output Y is same as D_0. If $D_0 = 0$, $Y = 0$ and if $D_0 = 1$, $Y = 1$.

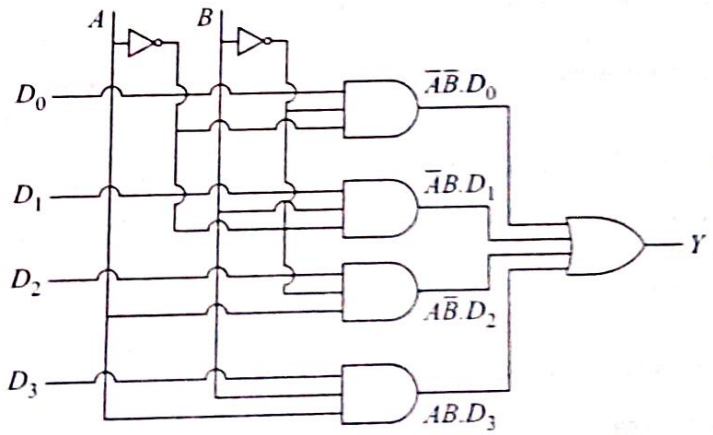
Similarly, for $AB = 01$, second AND gate will be active and all other AND gates remain inactive. Thus, output $Y = D_1$. Following same procedure we can complete the truth table of Fig. 4.1b.



(a)

A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

(b)



(c)

Fig. 4.1 *(a) Multiplexer block diagram. (b) 4-to-1 multiplexer truth table. (c) its logic circuit.*

Now, if we want 5-to-1 multiplexer how many select lines are required? There is no 5^{th} combination possible with two select lines and hence we need a third select input. Note that, with three we can

select up to $2^3 = 8$ data inputs. Commercial multiplexers ICs come in integer power of 2, e.g. 2-to-1, 4-to-1, 8-to-1, 16-to-1 multiplexers. With this background, let us look at a 16-to-1 multiplexer circuit, which may look complex but follows same logic as that of a 4-to-1 multiplexer.

16-to-1 Multiplexer

Figure shows a 16-to-1 multiplexer. The input bits are labeled D_0 to D_{15}. Only one of these is transmitted to the output. Which one depends on the value of $ABCD$, the control input. For instance, when

$$ABCD = 0000$$

the upper AND gate is enabled while all other AND gates are disabled. Therefore, data bit D_0 is transmitted to the output, giving

$$Y = D_0$$

If D_0 is low, Y is low; if D_0 is high, Y is high. The point is that Y depends only on the value of D_0.

If the control nibble (group of 4-bits) is changed to

$$ABCD = 1111$$

all gates are disabled except the bottom AND gate. In this case, D_{15} is the only bit transmitted to the output, and

$$Y = D_{15}$$

As you can see, the control nibble determines which of the input data bits is transmitted to the output. Thus we can write output as

$$Y = A'B'C'D'.D_0 + A'B'C'D.D_1 + A'B'CD'.D_2 + \cdots + ABCD'.D_{14} + ABCD.D_{15}$$

At this point can we answer, how would an 8 to 1 multiplexer circuit look like? First of all we need three select lines for 8 data inputs. And there will be 8 AND gates each one having four inputs; three from select lines and one from data input. The final output is generated from an OR gate which takes input from 8 AND gates. The equation for this can be written as

$$Y = A'B'C'.D_0 + A'B'C.D_1 + A'BC'.D_2 + A'BC.D_3 + AB'C'.D_4 + AB'C.D_5 + ABC'.D_6 + ABC.D_7$$

Thus, for $ABC = 000$, multiplexer output $Y = D_0$; other AND gates and corresponding data inputs D_1 to D_7 remain inactive. Similarly, for $ABC = 001$, multiplexer output $Y = D_1$, for $ABC = 010$, multiplexer output $Y = D_2$ and finally, for $ABC = 111$, multiplexer output $Y = D_7$.

The circuit for 2 to 1 multiplexer is shown in Fig. 4.2a and in Fig. 4.2b we list some of the commercially available multiplexer ICs.

The 74150

Try to visualize the 16-input OR gate of Fig. 4.2 changed to a NOR gate. What effect does this have on the operation of the circuit? Almost none. All that happens is we get the complement of the selected data bit rather than the data bit itself. For instance, when $ABCD = 0111$, the output is

$$Y = \overline{D_7}$$

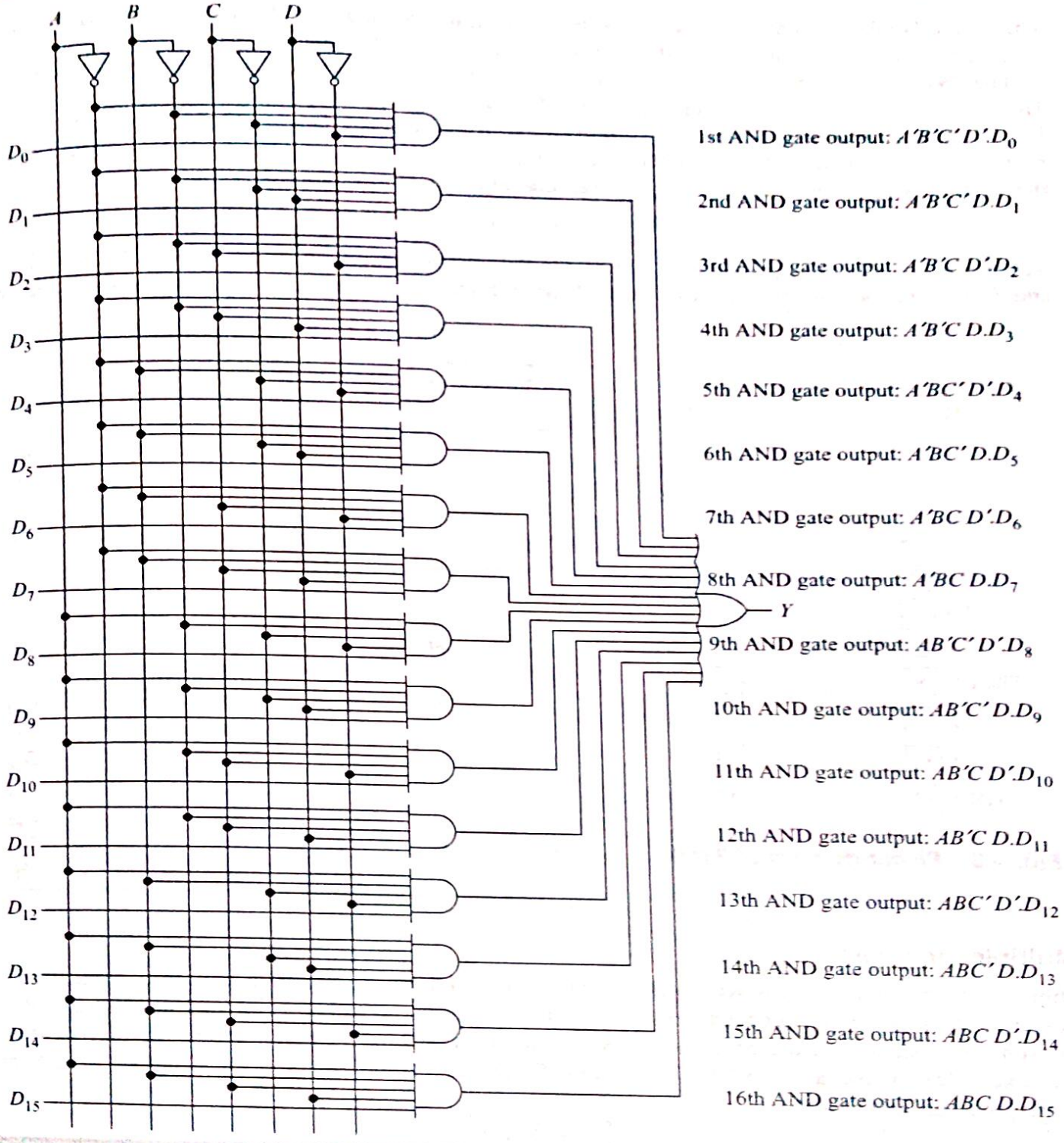


Fig. 4.2 *Sixteen-to-one multiplexer.*

This is the Boolean equation for a typical transistor-transistor logic (TTL) multiplexer because it has an inverter on the output that produces the complement of the selected data bit.

The 74150 is a 16-to-1 TTL multiplexer with the pin diagram shown in Fig. 4.3. Pins 1 to 8 and 16 to 23 are for the input data bits D_0 to D_{15}. Pins 11, 13, 14, and 15 are for the control bits $ABCD$. Pin 10 is the output; and it equals the complement of the selected data bit. Pin 9 is for the STROBE, an input signal that disables or enables the multiplexer. As shown in Table 4.1, a low strobe enables the multiplexer, so that output Y equals the complement of the input data bit:

$$Y = \overline{D}_n$$

where n is the decimal equivalent of $ABCD$. On the other hand, a high strobe disables the multiplexer and forces the output into the high state. With a high strobe, the value of $ABCD$ doesn't matter.

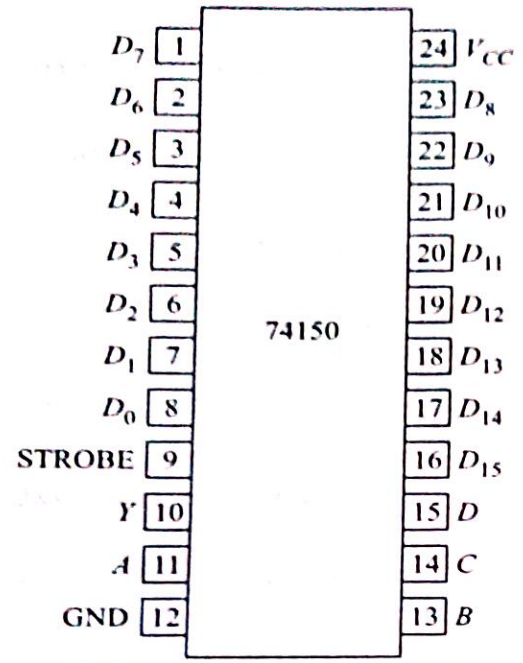


Fig. 4.3 *Pinout diagram of 74150.*

Table 4.1 74150 Truth Table

Strobe	A	B	C	D	Y
L	L	L	L	L	$\overline{D}_0$
L	L	L	L	H	$\overline{D}_1$
L	L	L	H	L	$\overline{D}_2$
L	L	L	H	H	$\overline{D}_3$
L	L	H	L	L	$\overline{D}_4$
L	L	H	L	H	$\overline{D}_5$
L	L	H	H	L	$\overline{D}_6$
L	L	H	H	H	$\overline{D}_7$
L	H	L	L	L	$\overline{D}_8$
L	H	L	L	H	$\overline{D}_9$
L	H	L	H	L	$\overline{D}_{10}$
L	H	L	H	H	$\overline{D}_{11}$
L	H	H	L	L	$\overline{D}_{12}$
L	H	H	L	H	$\overline{D}_{13}$
L	H	H	H	L	$\overline{D}_{14}$
L	H	H	H	H	$\overline{D}_{15}$
H	X	X	X	X	H

Multiplexer Logic

Digital design usually begins with a truth table. The problem is to come up with a logic circuit that has the same truth table. In Chapter 3, you saw two standard methods for implementing a truth table: the sum-of-products and the product-of-sums solutions. The third method is the *multiplexer solution.* For example, to use a 74150 to implement Table 4.2. Complement each Y output to get the corresponding data input:

$$D_0 = \overline{1} = 0$$

$$D_1 = \overline{0} = 1$$

$$D_2 = \overline{1} = 0$$

and so forth, up to

$$D_{15} = \overline{1} = 0$$

Next, wire the data inputs of 74150 as shown in Fig. 4.4, so that they equal the foregoing values. In other words, D_0 is grounded, D_1 is connected to +5 V, D_2 is grounded, and so forth. In each of these cases, the data input is the complement of the desired Y output of Table 4.2.

Figure 4.4 is the multiplexer design solution. It has the same truth table given in Table 4.2. If in doubt, analyze it as follows for each input condition. When $ABCD = 0000$, D_0 is the selected input in Fig. 4.4. Since D_0 is low, Y is high. When $ABCD = 0001$, D_1 is selected. Since D_1 is high, Y is low. If you check the remaining input possibilities, you will see that the circuit has the truth table given in Table 4.2.

Bubbles on Signal Lines

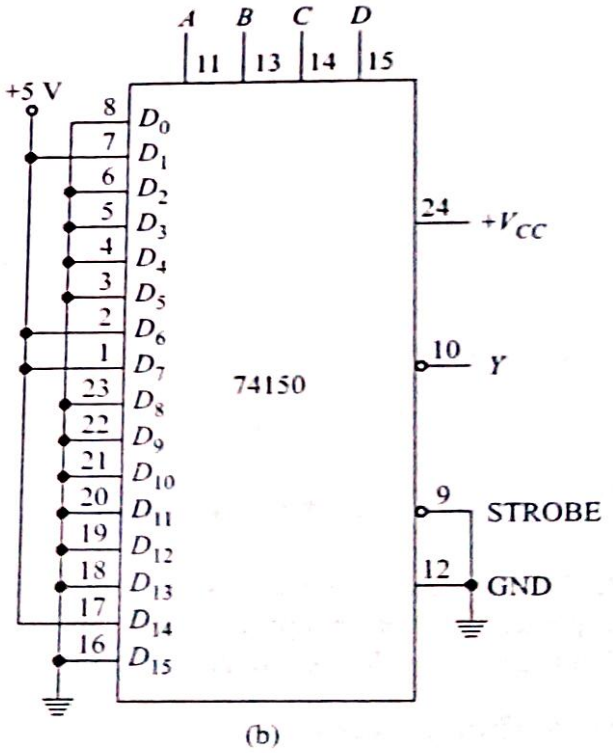
Data sheets often show inversion bubbles on some of the signal lines. For instance, notice the bubble on pin 10, the output of Fig. 4.4. This bubble is a reminder that the output is the complement of the selected data bit.

Also notice the bubble on the STROBE input (pin 9). As discussed earlier, the multiplexer is active (enabled) when the STROBE is low and inactive (disabled) when it is high. Because of this, the STROBE is called an *active-low signal*; it causes something to happen when it is low rather than when it is high. Most schematic diagrams use bubbles to indicate active-low signals. From now on, whenever you see a bubble on an input pin, remember that it means the signal is active-low.

Table 4.2

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

(a)



(b)

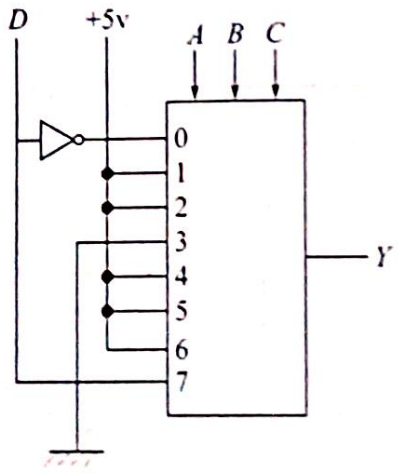
Fig. 4.4 *Using a 74150 for multiplexer logic.*

Universal Logic Circuit

Multiplexer sometimes is called *universal logic circuit* because a 2^n-to-1 multiplexer can be used as a design solution for any n variable truth table. This we have seen for realization of a 4 variable truth table by 16-to-1 multiplexer in Fig. 4.5. Here, we show how this truth table can be realized using an 8-to-1 multiplexer. Let's consider A,B and C variables to be fed as select inputs. The fourth variable D then has to be present as data input. The method is shown in Fig. 4.5a. The first three rows map the truth table in a different way, similar to the procedure we adopted in entered variable map (Section 3.3). We write all the combinations of 3 select inputs in first row along different columns. Now corresponding to each value of 4th variable D, truth table output Y is written in 2nd and 3rd row. The 4th row writes Y as a function of D. In fifth row we assign data input values for 8-to-1 multiplexer simply copying Y values obtained in previous row. This is because for each select variable combination a multiplexer transfers a particular input to its output. In 8-to-1 multiplexer, $ABC = 000$ selects D_0, $ABC = 001$ selects D_1 and so on. The corresponding circuit is shown in Fig. 4.5b.

ABC	000	001	010	011	100	101	110	111
$D = 0$	1	1	1	0	1	1	1	0
$D = 1$	0	1	1	0	1	1	1	1
Y	D'	1	1	0	1	1	1	D
8-to-1 MUX data input	$D_0 = D'$	$D_1 = 1$	$D_2 = 1$	$D_3 = 0$	$D_4 = 1$	$D_5 = 1$	$D_6 = 1$	$D_7 = D$

(a)



(b)

Fig. 4.5 *A four variable truth table realization using 8-to-1 multiplexer.*

Note that, we can choose any of the four variables (A,B,C,D) of truth table to feed as input to 8-to-1 multiplexer but then mapping in first three rows of Fig. 4.5a will change. The rest of the procedure will remain same. We show an alternative to this technique for a new problem in Example 4.2.

Nibble Multiplexers

Sometimes we want to select one of two input nibbles. In this case, we can use a nibble multiplexer like the one shown in Fig. 4.6. The input nibble on the left is $A_3A_2A_1A_0$ and the one on the right

is $B_3B_2B_1B_0$. The control signal labeled *SELECT* determines which input nibble is transmitted to the output. When SELECT is low, the four NAND gates on the left are activated; therefore,

$$Y_3Y_2Y_1Y_0 = A_3A_2A_1A_0$$

When SELECT is high, the four NAND gates on the right are active, and

$$Y_3Y_2Y_1Y_0 = B_3B_2B_1B_0$$

Figure 4.7a on the next page shows the pinout diagram of a 74157, a nibble multiplexer with a SELECT input as previously described. When SELECT is low, the left nibble is steered to the output. When SELECT is high, the right nibble is steered to the output. The 74157 also includes a strobe input. As before, the strobe must be low for the multiplexer to work properly. When the strobe is high, the multiplexer is inoperative.

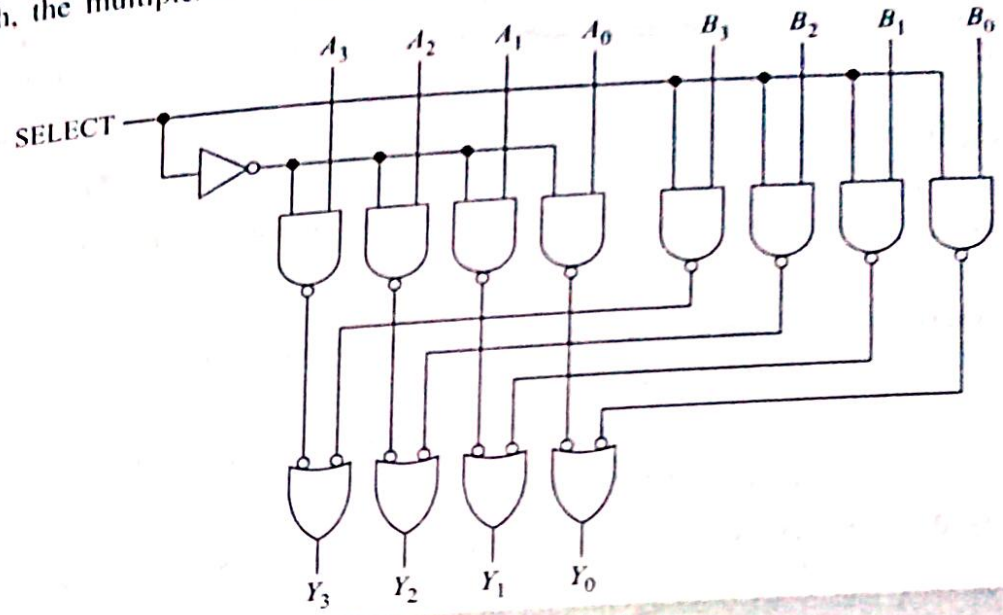


Fig. 4.6 *Nibble multiplexer.*

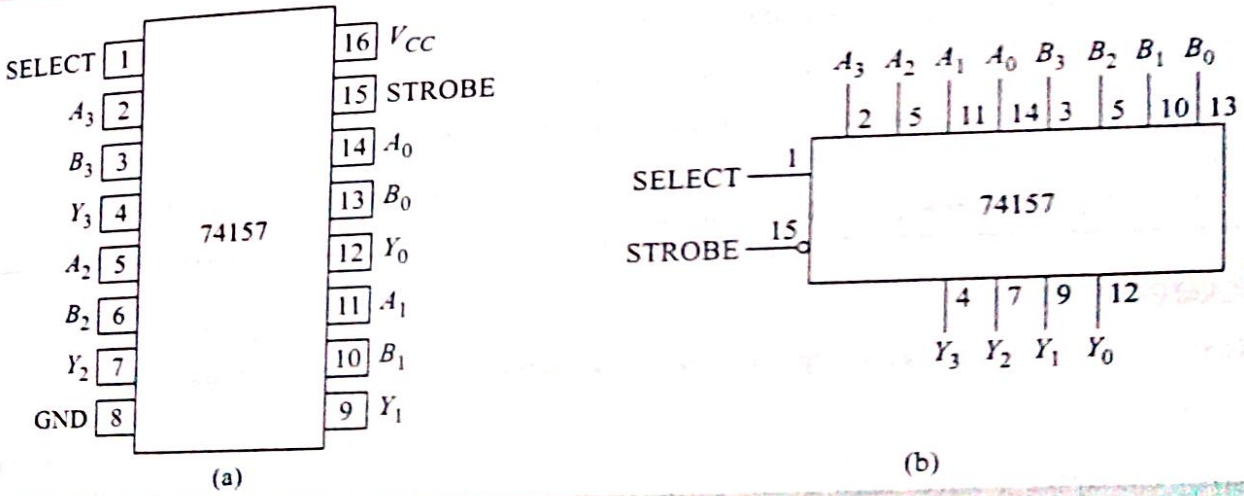


Fig. 4.7 *Pinout diagram of 74157.*

Figure 4.7b shows how to draw a 74157 on a schematic diagram. The bubble on pin 15 tells us that *STROBE* is an active-low input.

EXAMPLE 4.1

Show how 4-to-1 multiplexer can be obtained using only 2-to-1 multiplexer.

Solution

Logic equation for 2-to-1 Multiplexer: $Y = A'.D_0 + A.D_1$

Logic equation for 4-to-1 Multiplexer: $Y = A'B'.D_0 + A'B.D_1 + AB'.D_2 + AB.D_3$

This can be rewritten as, $Y = A'(B'.D_0 + B.D_1) + A(B'.D_2 + B.D_3)$

Compare this with equation of 2-to-1 multiplexer. We need two 2-to-1 multiplexer to realize two bracketed terms where B serves as select input. The output of these two multiplexers can be sent to a third multiplexer as data inputs where A serves as select input and we get the 4-to-1 multiplexer. Figure 4.8a shows circuit diagram for this.

EXAMPLE 4.2

(a) Realize $Y = A'B + B'C' + ABC$ using an 8-to-1 multiplexer. (b) Can it be realized with a 4-to-1 multiplexer?

Solution

(a) First we express Y as a function of minterms of three variables. Thus

$$Y = A'B + B'C' + ABC$$

$$Y = A'B(C' + C) + B'C'(A' + A) + ABC \text{ [As, } X + X' = 1\text{]}$$

$$Y = A'B'C' + A'BC' + A'BC + AB'C' + ABC$$

Comparing this with equation of 8 to 1 multiplexer, we find by substituting $D_0 = D_2 = D_3 = D_4 = D_7 = 1$ and $D_1 = D_5 = D_6 = 0$ we get given logic relation.

(b) Let variables A and B be used as selector in 4 to 1 multiplexer and C fed as input. The 4-to-1 multiplexer generates 4 minterms for different combinations of AB. We rewrite given logic equation in such a way that all these terms are present in the equation.

$$Y = A'B + B'C' + ABC$$

$$Y = A'B + B'C'(A' + A) + ABC \text{ [As, } X + X' = 1\text{]}$$

$$Y = A'B'.C' + A'B.1 + AB'.C' + AB.C$$

Compare above with equation of a 4-to-1 multiplexer. We see $D_0 = C'$, $D_1 = 1$, $D_2 = C'$ and $D_3 = C$ generate the given logic function.

EXAMPLE 4.3

Design a 32-to-1 multiplexer using two 16-to-1 multiplexers and one 2-to-1 multiplexer.

Solution

The circuit diagram is shown in Fig. 4.8b. A 32-to-1 multiplexer requires $\log_2 32 = 5$ select lines say, *ABCDE*. The lower 4 select lines *BCDE* chose 16-to-1 multiplexer outputs. The 2-to-1 multiplexer chooses one of the output of two 16-to-1 multiplexers depending on what appears in the 5th select line, *A*.

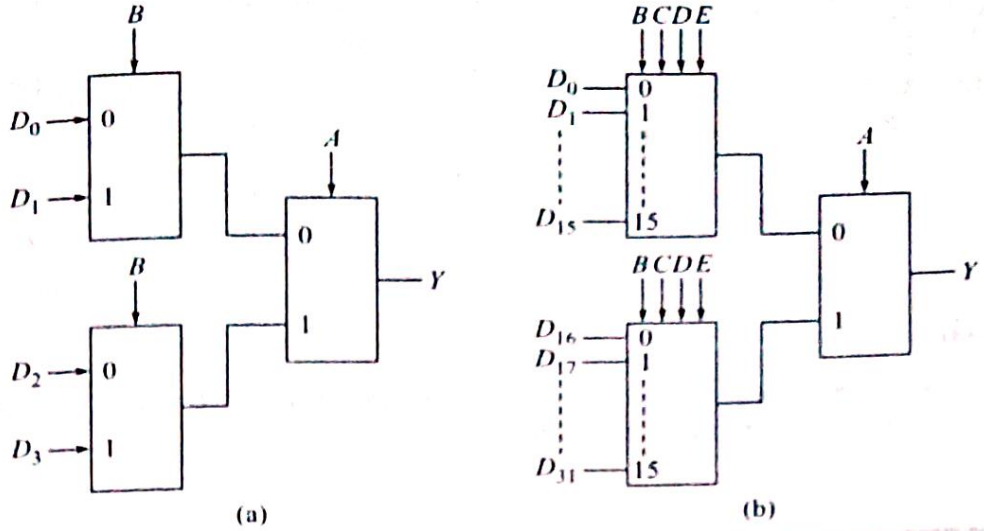


Fig. 4.8 *Realization of higher order multiplexers using lower orders.*

SELF-TEST

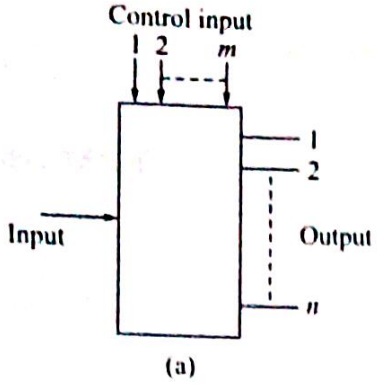
1. A circuit with many inputs but only one output is called a _____.
2. What is the significance of the bubble on pin 10 of the multiplexer in Fig. 4.5?

4.2 DEMULTIPLEXERS

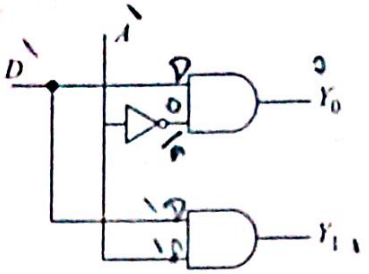
Demultiplex means *one into many*. A *demultiplexer* is a logic circuit with one input and many outputs. By applying control signals, we can steer the input signal to one of the output lines. Figure 4.9a illustrates the general idea. The circuit has 1 input signal, m control or select signals and n output signals where $n \leq 2^m$. Figure 4.9b shows the circuit diagram of a 1-to-2 demultiplexer. Note the similarity of multiplexer and demultiplexer circuits in generating different combinations of control variables through a bank of AND gates. Figure 4.9c lists some of the commercially available demultiplexer ICs. Note that a demultiplexer IC can also behave like a decoder. More about this will be discussed in next section.

1-to-16 Demultiplexer

Figure 4.10 shows a 1-to-16 demultiplexer. The input bit is labeled D. This data bit (D) is transmitted to the data bit of the output lines. But which one? Again, this depends on the value of $ABCD$, the control input. When $ABCD = 0000$, the upper AND gate is enabled while all other AND gates are disabled. Therefore, data bit D is transmitted only to the Y_0 output, giving $Y_0 = D$. If D is low, Y_0 is low. If D is high, Y_0 is high. As you can see, the value of Y_0 depends on the value of D. All other outputs are in the low state. If the control nibble is changed to $ABCD = 1111$, all gates are disabled except the bottom AND gate. Then, D is transmitted only to the Y_{15} output, and $Y_{15} = D$.



(a)



(b)

IC No.	DEMUX Type	Decoder Type
74154	1-to-16	4-to-16
74138	1-to-8	3-to-8
74155	1-to-4	2-to-4

(c)

Fig. 4.9 *(a) Demultiplexer block diagram. (b) Logic circuit of 1-to-2 demultiplexer. (c) Few commercially available ICs.*

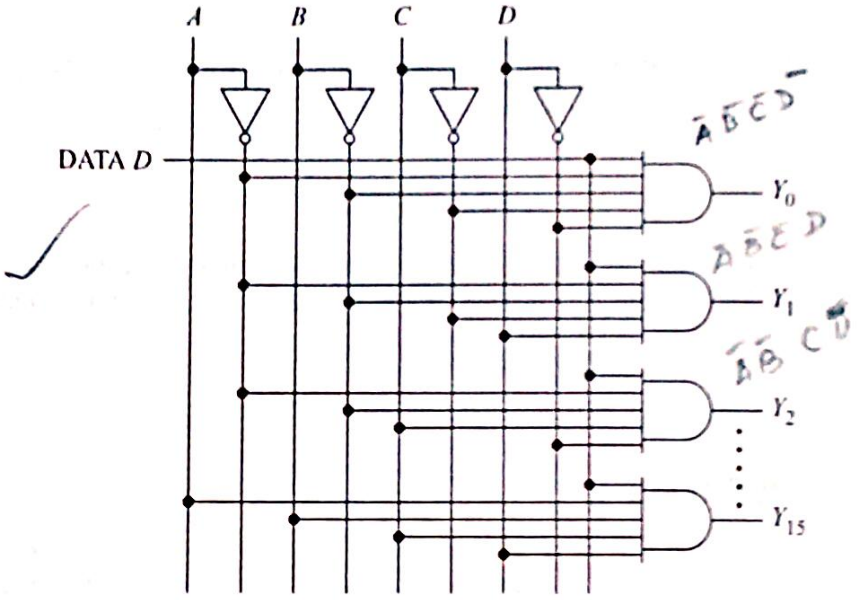


Fig. 4.10 *1-to-16 demultiplexer.*

The 74154

The 74154 is a 1-to-16 demultiplexer with the pin diagram of Fig. 4.11. Pin 18 is for the input DATA *D*, and pins 20 to 23 are for the control bits *ABCD*. Pins 1 to 11 and 13 to 17 are for the output bits Y_0 to Y_{15}. Pin 19 is for the STROBE, again an active-low input. Finally, pin 24 is for V_{CC} and pin 12 for ground.

Table 4.3 shows the truth table of a 74154. First, notice the STROBE input. It must be low to activate the 74154. When the STROBE is low, the control input *ABCD* determines which output lines are low when the DATA input is low. When the DATA input is high, all output lines are high. And, when the STROBE is high, all output lines are high.

Figure 4.12 shows how to draw a 74154 on a schematic diagram. There is one input DATA bit (pin 18) under the control of nibble *ABCD*. The DATA bit is automatically steered to the output line

whose subscript is the decimal equivalent of *ABCD*. Again, the bubble on the STROBE pin indicates an active-low input. Notice that DATA is inverted at the input (the bubble on pin 18) and again on any output (the bubble on each output pin). With this double inversion, DATA passes through the 74154 unchanged.

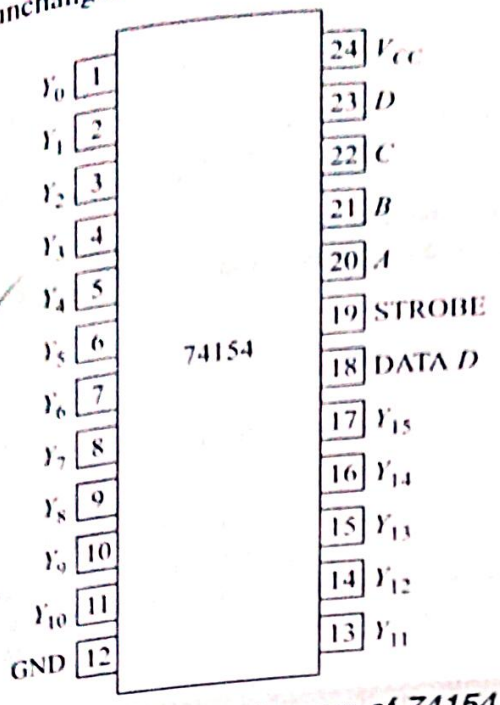


Fig. 4.11 *Pinout diagram of 74154.*

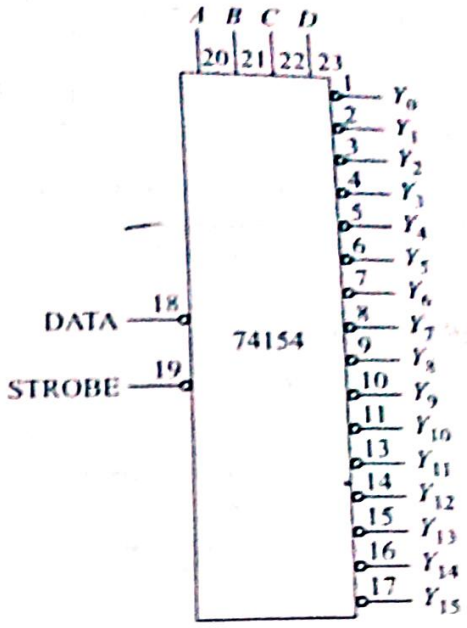


Fig. 4.12 *Logic diagram of 74154.*

Table 4.3 74154 Truth Table

Strobe	Data	A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

EXAMPLE 4.4

In Fig. 4.13a, what does the Y_{12} output equal for each of the following conditions:

a. R is high, T is high, $ABCD = 0110$.
b. R is low, T is high, $ABCD = 1100$.
c. R is high, T is high, $ABCD = 1100$.

Solution

a. Since R and T are both high, the STROBE is low and the 74154 is active. Because $ABCD = 0110$, the input data is steered to the Y_6 output line (pin 7). The Y_{12} output remains in the high state (see Table 4.3).
b. Here, the STROBE is high and the 74154 is inactive. The Y_{12} output is high.
c. With R and T both high, the STROBE is low and the 74154 is active. Since $ABCD = 1100$, the two pulses are steered to the Y_{12} output (pin 14).

EXAMPLE 4.5

Show how two 1-to-16 demultiplexers can be connected to get a 1-to-32 demultiplexer.

Solution

Figure 4.13b shows the circuit diagram. A 1-to-32 demultiplexer has 5 select variables $ABCDE$. Four of them ($BCDE$) are fed to two 1-to-16 demultiplexer. And the fifth (A) is used to select one of these two multiplexer through strobe input. If $A = 0$, the top 714154 is chosen and $BCDE$ directs data to one of the 15 outputs of that IC. If $A = 1$, the bottom IC is chosen and depending on value of $BCDE$ data is directed to one of the 15 outputs this IC.

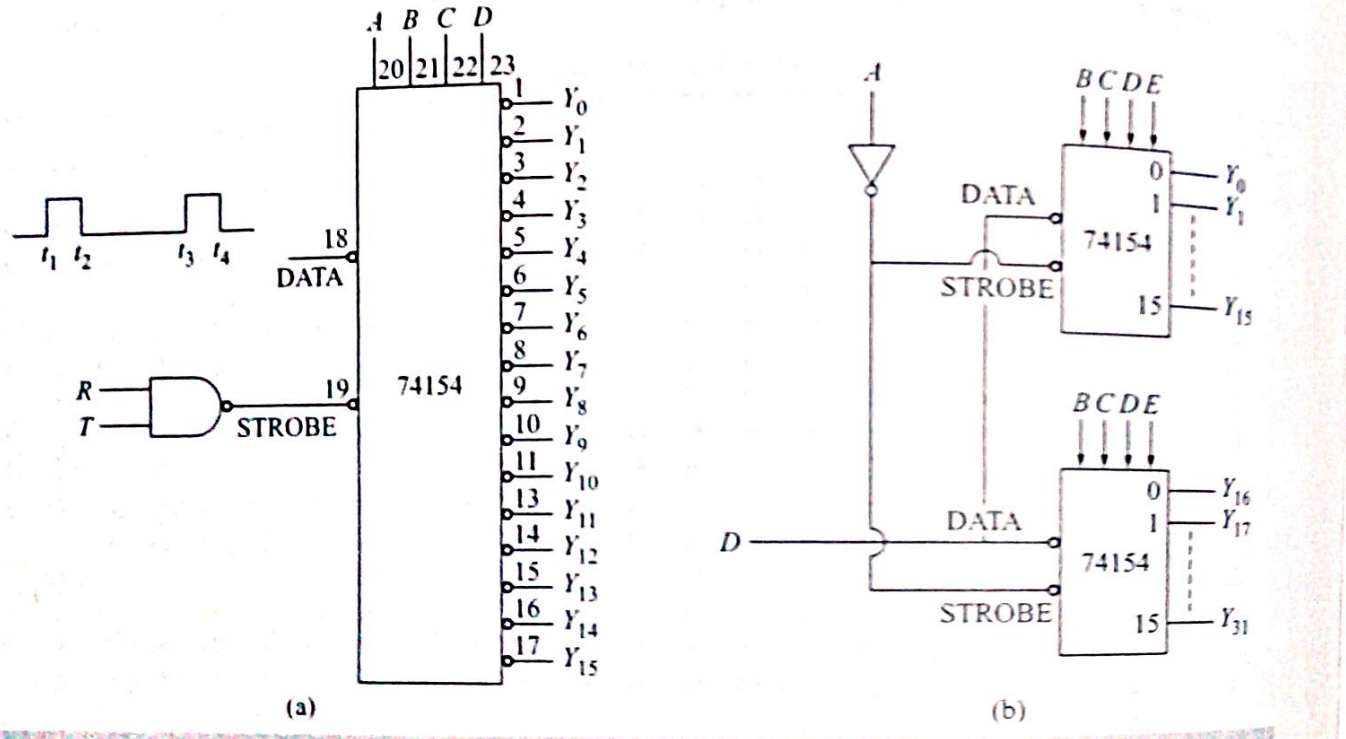
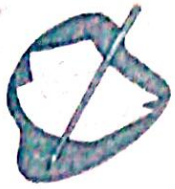


Fig. 4.13

SELF-TEST



3. A logic circuit with one input and many outputs is called a ____.
4. For the 74154 demultiplexer, what must the logic levels *ABCD* be in order to steer the DATA input signal to output line Y_{10}?
5. If *ABCD* = *LHLH*, DATA = *L*, and STROBE = *H*, what will the logic level be at Y_5 on a 74154?

4.3 1-OF-16 DECODER

A *decoder* is similar to a demultiplexer, with one exception—there is no data input. The only inputs are the control bits *ABCD*, which are shown in Fig. 4.14. This logic circuit is called a *1-of-16*

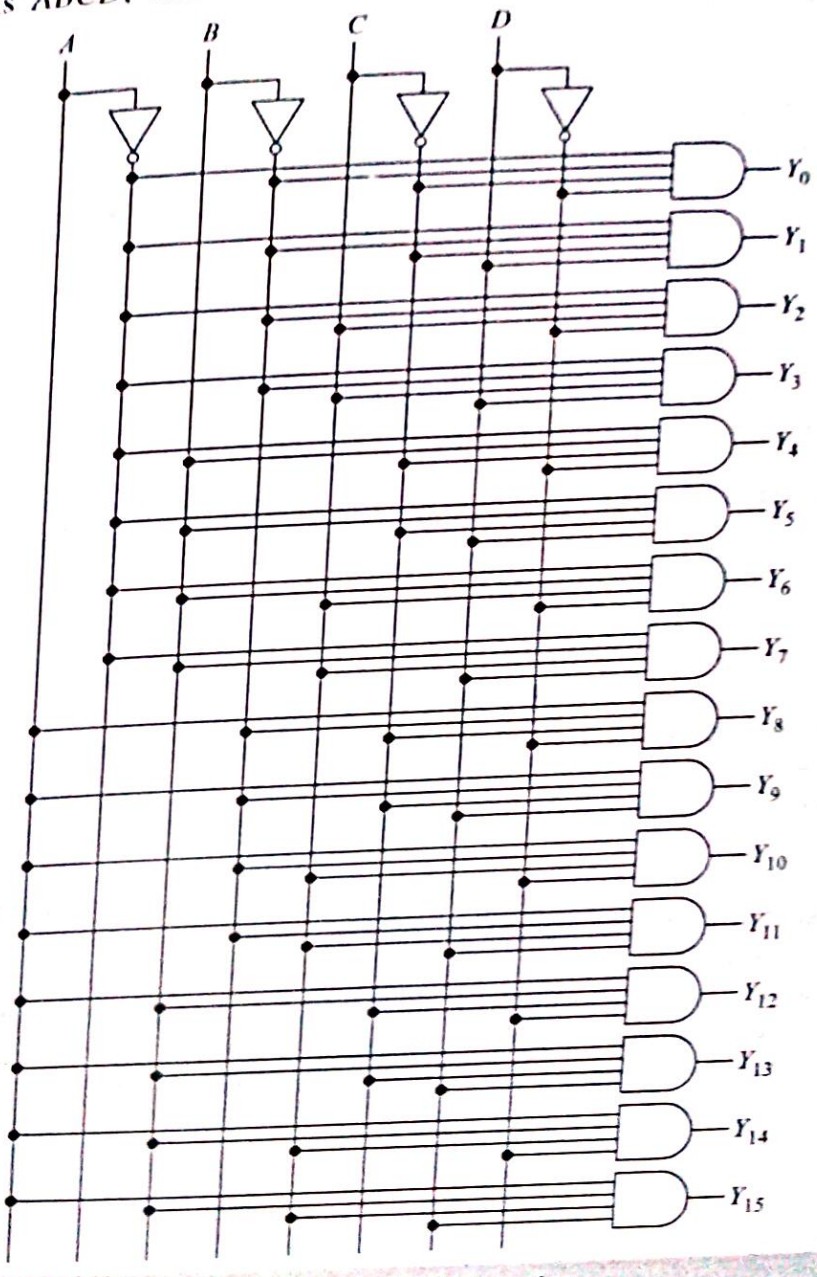


Fig. 4.14 *1-of-16 decoder.*

decoder because only 1 of the 16 output lines is high. For instance, when *ABCD* is 0001, only the Y_1 AND gate has all inputs high; therefore, only the Y_1 output is high. If *ABCD* changes to 0100 only the Y_4 AND gate has all inputs high; as a result, only the Y_4 output goes high.

If you check the other *ABCD* possibilities (0000 to 1111), you will find that the subscript of the high output always equals the decimal equivalent of *ABCD*. For this reason, the circuit is sometimes called a *binary-to-decimal decoder*. Because it has 4 input lines and 16 output lines, the circuit is also known as a *4-line to 16-line decoder*.

Normally, you would not build a decoder with separate inverters and AND gates as shown in Fig. 4.14. Instead, you would use an IC such as the 74154. The 74154 is called a *decoder-demultiplexer*, because it can be used either as a decoder or as a demultiplexer.

You saw how to use a 74154 as a demultiplexer in Sec. 4.2. To use this same IC as a decoder, all you have to do is ground the DATA and STROBE inputs as shown in Fig. 4.15. Then, the selected output line is in the low state (see Table 4.3). This is why bubbles are shown on the output lines. They remind us that the output line is low when it is active or selected. For instance, if the binary input is

$$ABCD = 0111$$

then the Y_7 output is low, while all other-outputs are high.

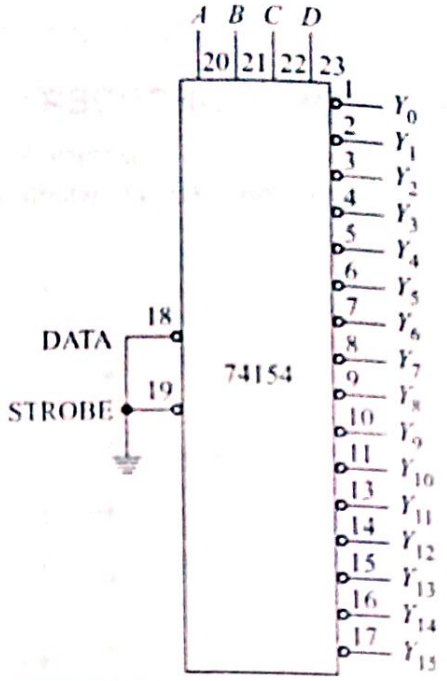


Fig. 4.15 *Using 74154 as decoder.*

EXAMPLE 4.6

Figure 4.16 illustrates *chip expansion*. We have expanded two 74154s to get a 1-of-32 decoder. Here is the way the circuit works. Bit *X* drives the first 74154, and the complement of *X* drives the second 74154. When *X* is low, the first 74154 is active and the second is inactive. The *ABCD* input drives both decoders but only the first is active; therefore, only one output line on the first decoder is in the low state.

On the other hand, when *X* is high, the first 74154 is disabled and the second one is enabled. This means the *ABCD* input is decoded into a low output from the second decoder. In effect, the circuit of Fig. 4.16 acts like a 1-of-32 decoder.

In Fig. 4.16, all output lines are high, except the decoded output line. The bubble on each output line tells anyone looking at the schematic diagram that the active output line is in the low state rather than the high state. Similarly, the bubbles on the STROBE and DATA inputs of each 74154 indicate active-low inputs.

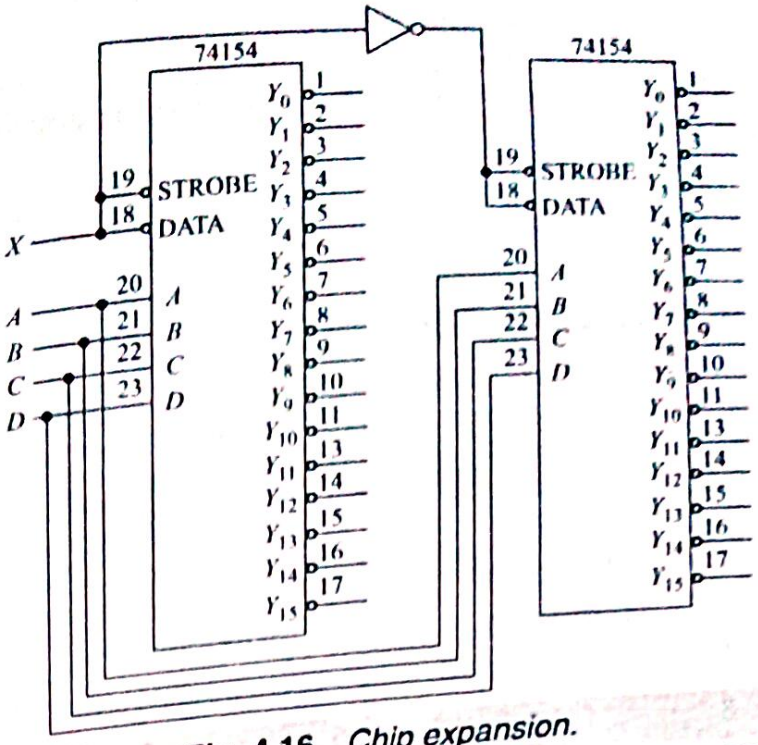


Fig. 4.16 *Chip expansion.*

EXAMPLE 4.7

Show how using a 3-to-8 decoder and multi-input OR gates following Boolean expressions can be realized simultaneously.

$F_1(A, B, C) = \Sigma m(0, 4, 6)$; $F_2(A, B, C) = \Sigma m(0, 5)$; $F_2(A, B, C) = \Sigma m(1, 2, 3, 7)$

Solution

Since at the decoder output we get all the minterms we use them as shown in Fig. 4.17 to get the required Boolean functions.

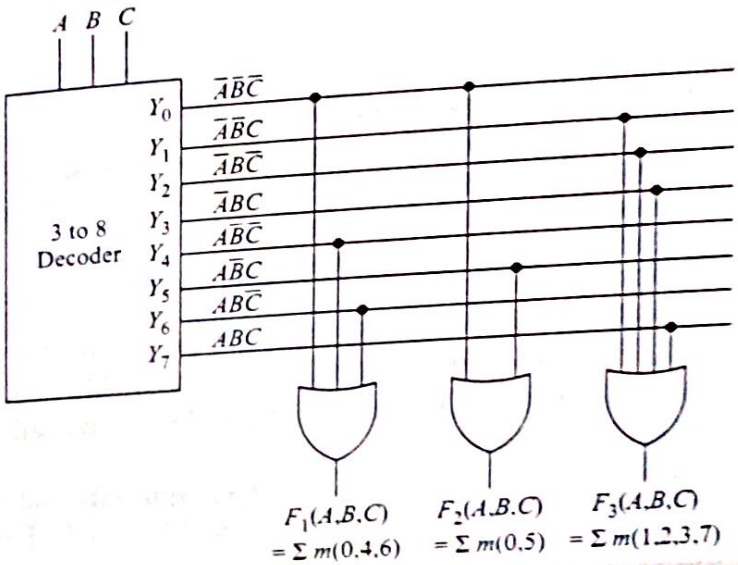


Fig. 4.17 *Solution for Example 4.7.*

SELF-TEST

6. What is the significance of the bubbles on the outputs of the 74154 in Fig. 4.15?
7. In Fig. 4.15, *ABCD* = *HLHL*. What are the logic levels of the outputs?

4.4 BCD-TO-DECIMAL DECODERS

BCD is an abbreviation for *binary-coded decimal*. The BCD code expresses each digit in a decimal number by its nibble equivalent. For instance, decimal number 429 is changed to its BCD form as follows:

4	2	9
↓	↓	↓
0100	0010	1001

To anyone using the BCD code, 0100 0010 1001 is equivalent to 429.

As another example, here is how to convert the decimal number 8963 to its BCD form:

8	9	6	3
↓	↓	↓	↓
1000	1001	0110	0011

Again, we have changed each decimal digit to its binary equivalent.

Some early computers processed BCD numbers. This means that the decimal numbers were changed into BCD numbers, which the computer then added, subtracted, etc. The final answer was converted from BCD back to decimal numbers.

Here is an example of how to convert from the BCD form back to the decimal number:

0101	0111	1000
↓	↓	↓
5	7	8

As you can see, 578 is the decimal equivalent of 0101 0111 1000.

One final point should be considered. Notice that BCD digits are from 0000 to 1001. All combinations above this (1010 to 1111) cannot exist in the BCD code because the highest decimal digit being coded is 9.

BCD-to-Decimal Decoder

The circuit of Fig. 4.18 is called a *1-of-10 decoder* because only 1 of the 10 output lines is high. For instance, when *ABCD* is 0011, only the Y_3 AND gate has all high inputs; therefore, only the Y_3 output is high. If *ABCD* changes to 1000, only the Y_8 AND gate has all high inputs; as a result, only the Y_8 output goes high.

If you check the other *ABCD* possibilities (0000 to 1001), you will find that the subscript of the high output always equals the decimal equivalent of the input BCD digit. For this reason, the circuit is also called a *BCD-to-decimal converter*.

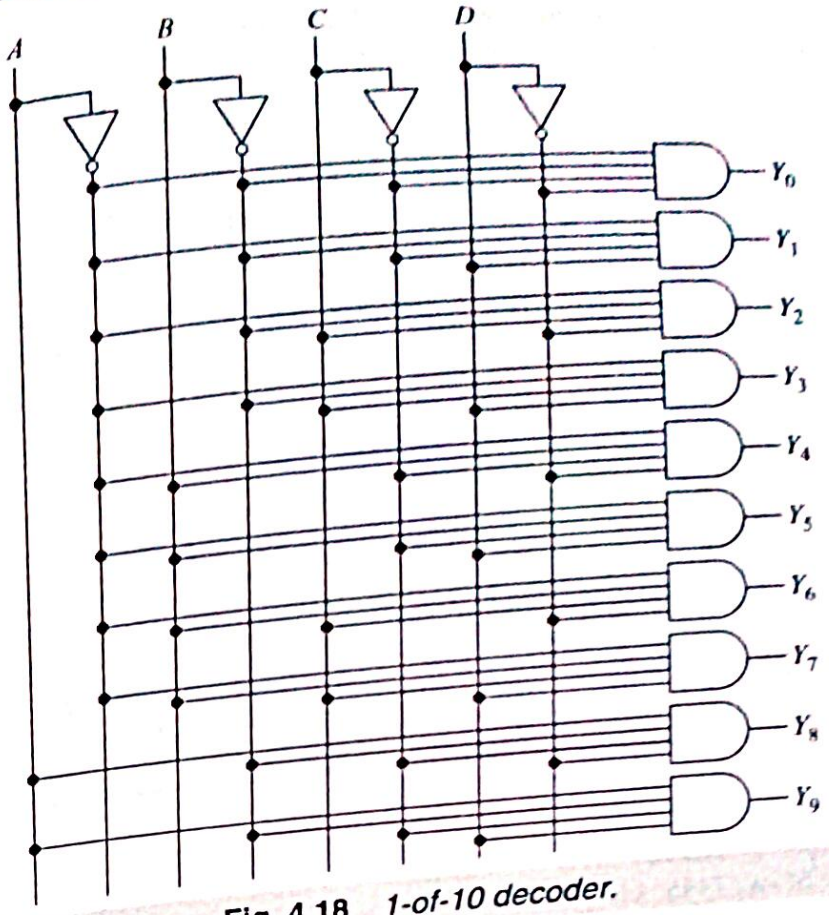


Fig. 4.18 *1-of-10 decoder.*

The 7445

Typically, you would not build a decoder with separate inverters and AND gates, as shown in Fig. 4.18. Instead, you would use a TTL IC like the 7445 of Fig. 4.19. Pin 16 connects to the supply voltage V_{CC} and pin 8 is grounded. Pins 12 to 15 are for the BCD input (*ABCD*), while pins 1 to 7 and 9 to 11 are for the outputs. This IC is functionally equivalent to the one in Fig. 4.18, except that the active output line is in the low state. All other output lines are in the high state, as shown in Table 4.4. Notice that an invalid BCD input (1010 to 1111) forces all output lines into the high state.

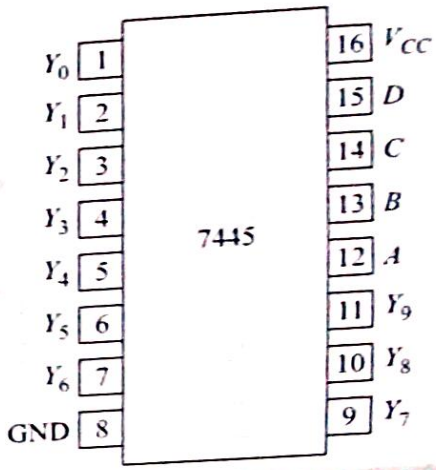


Fig. 4.19 *Pinout diagram of 7445.*

Table 4.4 7445 Truth Table

No.	Inputs A	B	C	D	Outputs Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

EXAMPLE 4.8

The decoded outputs of a 7445 can be connected to light-emitting diodes (LEDs), as shown in Fig. 4.20. If each resistance is 1 kΩ and each LED has a forward voltage drop of 2 V, how much current is there through a LED when it is conducting? (See Chap. 13 for a discussion of LEDs.)

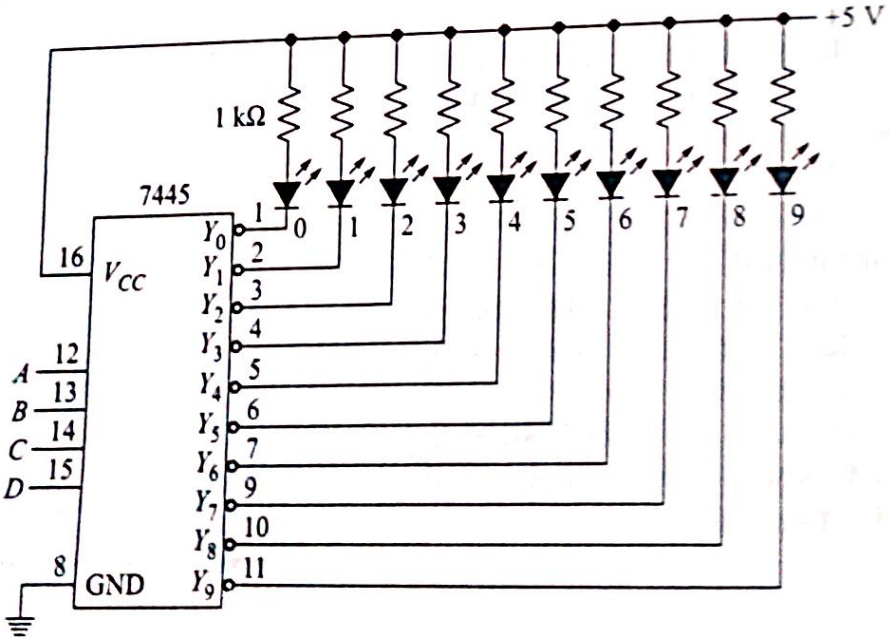


Fig. 4.20 *Circuit for Example 4.7.*

Solution

When an output is in the low state, you can approximate the output voltage as zero. Therefore, the current through a LED is

$$I = \frac{5\text{ V} - 2\text{ V}}{1\text{ k}\Omega} = 3\text{ mA}$$

EXAMPLE 4.9

The LEDs of Fig. 4.20 are numbered 0 through 9. Which of the LEDs is lit for each of the following conditions:

a. $ABCD = 0101$.
b. $ABCD = 1001$.
c. $ABCD = 1100$.

Solution

a. When $ABCD = 0101$, the decoded output line is Y_5. Since Y_5 is approximately grounded, LED 5 lights up. All other LEDs remain off because the other outputs are high.
b. When $ABCD = 1001$, LED 9 is on.
c. $ABCD = 1100$ is an invalid input. Therefore, none of the LEDs is on because all output lines are high (see Table 4.4).

SELF-TEST

8. What does the abbreviation BCD stand for?
9. What is a LED?

4.5 SEVEN-SEGMENT DECODERS

A LED emits radiation when forward-biased. Why? Because free electrons recombine with holes near the junction. As the free electrons fall from a higher energy level to a lower one, they give up energy in the form of heat and light. By using elements like gallium, arsenic, and phosphorus, a manufacturer can produce LEDs that emit red, green, yellow, blue, orange and infrared (invisible) light. LEDs that produce visible radiation are useful in test instruments, pocket calculators, etc.

Seven-Segment Indicator

Figure 4.21a shows a *seven-segment indicator*, i.e. seven LEDs labeled *a* through *g*. By forward-biasing different LEDs, we can display the digits 0 through 9 (see Fig. 4.21b). For instance, to display a 0, we need to light up segments *a*, *b*, *c*, *d*, *e*, and *f*. To light up a 5, we need segments *a*, *c*, *d*, *f*, and *g*.

Seven-segment indicators may be the common-anode type where all anodes are connected together (Fig. 4.22a) or the common-cathode type where all cathodes are connected together (Fig. 4.22b). With the common-anode type of Fig. 4.22a, you have to connect a current-limiting resistor between each LED and ground. The size of this resistor determines how much current flows through the

LED. The typical LED current is between 1 and 50 mA. The common-cathode type of Fig. 4.22b uses a current-limiting resistor between each LED and $+V_{CC}$.

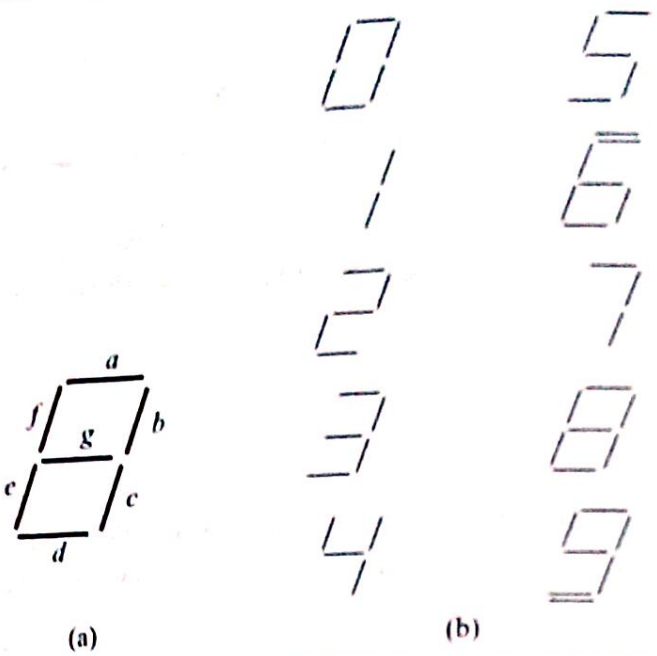


Fig. 4.21 *Seven-segment indicator.*

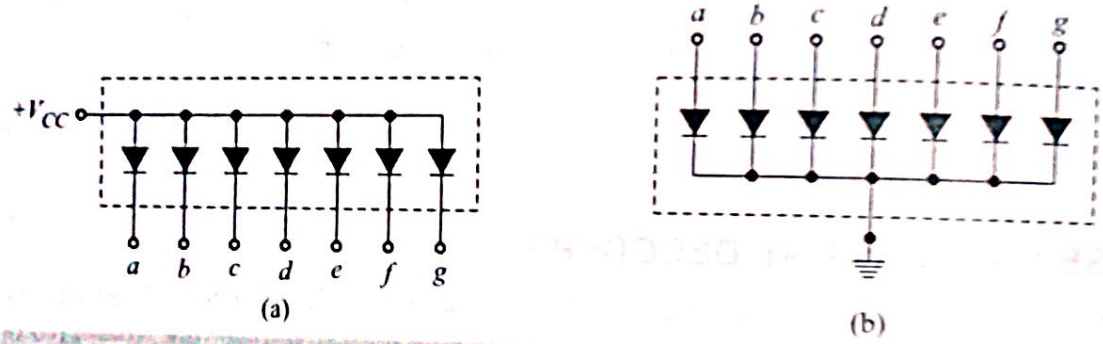


Fig. 4.22 *(a) Common-anode type. (b) Common-cathode type.*

The 7446

A seven-segment *decoder-driver* is an IC decoder that can be used to drive a seven-segment indicator. There are two types of decoder-drivers, corresponding to the common-anode and common-cathode indicators. Each decoder-driver has 4 input pins (the BCD input) and 7 output pins (the *a* through *g* segments).

Figure 4.23a shows a 7446 driving a common-anode indicator. Logic circuits inside the 7446 convert the BCD input to the required output. For instance, if the BCD input is 0111, the internal logic (not shown) of the 7446 will force LEDs *a*, *b*, and *c* to conduct. As a result, digit 7 will appear on the seven-segment indicator.

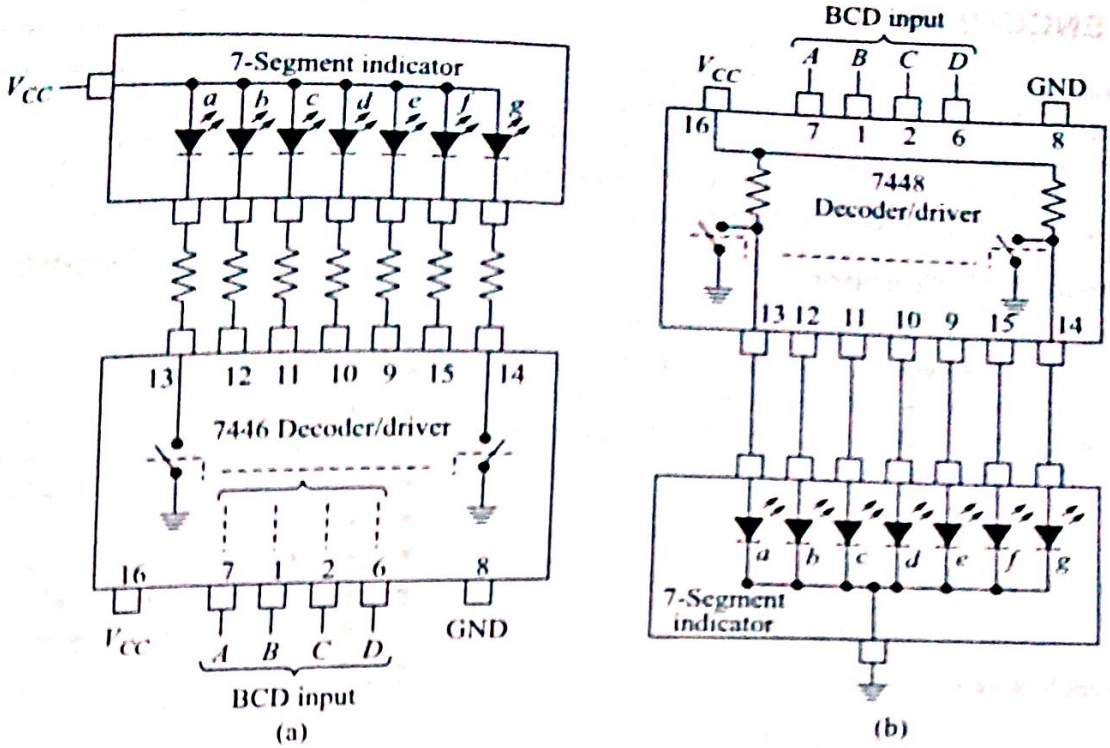


Fig. 4.23 *(a) 7446 decoder-driver. (b) 7448 decoder-driver.*

Notice the current-limiting resistors between the seven-segment indicator and the 7446 of Fig. 4.23a. You have to connect these external resistors to limit the current in each segment to a safe value between 1 and 50 mA, depending on how bright you want the display to be.

The 7448

Figure 4.23b is the alternative decoding approach. Here, a 7448 drives a common-cathode indicator. Again, internal logic converts the BCD input to the required output. For example, when a BCD input of 0100 is used, the internal logic forces LEDs *b*, *c*, *f*, and *g* to conduct. The seven-segment indicator then displays a 4. Unlike the 7446 that requires external current-limiting resistors, the 7448 has its own current-limiting resistors on the chip. A switch symbol is used to illustrate operation of the 7446 and 7448 in Fig. 4.23. Switching in the actual IC is of course accomplished using bipolar junction transistors (BJTs).

SELF-TEST

10. Sketch the segments in a seven-segment indicator.
11. Each segment of a seven-segment indicator is what type of device?

4.6 ENCODERS

An encoder converts an active input signal into a coded output signal. Figure 4.24 illustrates the general idea. There are n input lines, only one of which is active. Internal logic within the encoder converts this active input to a coded binary output with m bits.

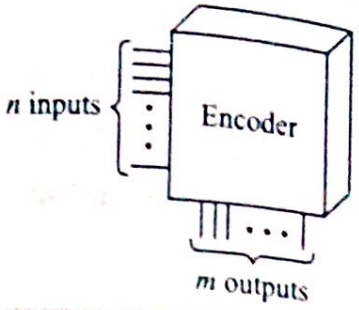


Fig. 4.24 Encoder.

Decimal-to-BCD Encoder

Figure 4.25 shows a common type of encoder—*the decimal-to-BCD encoder*. The switches are push-button switches like those of a pocket calculator. When button 3 is pressed, the C and D OR gates have high inputs: therefore, the output is

$$ABCD = 0011$$

If button 5 is pressed, the output becomes

$$ABCD = 0101$$

When switch 9 is pressed,

$$ABCD = 1001$$

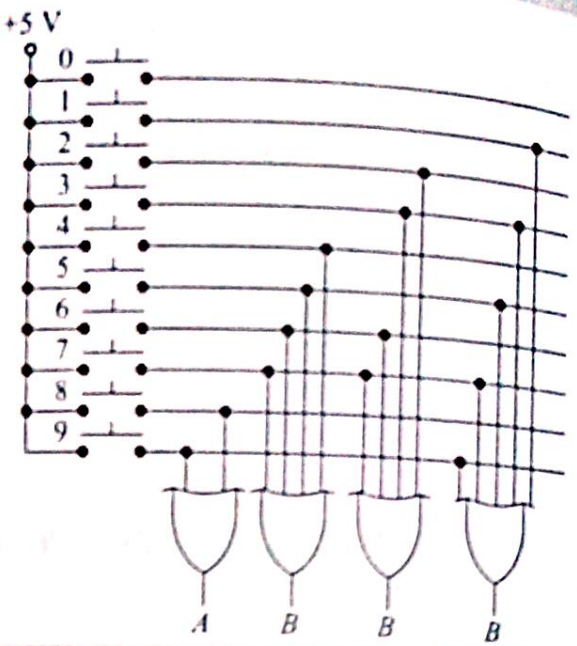


Fig. 4.25 Decimal-to-BCD encoder.

THE 74147

Figure 4.26a is the pinout diagram for a 74147, a decimal-to-BCD encoder. The decimal input, X_1 to X_9, connect to pins 1 to 5, and 10 to 13. The BCD output comes from pins 14, 6, 7, and 9. Pin 16 is for the supply voltage, and pin 8 is grounded. The label NC on pin 15 means *no connection* (the pin is not used).

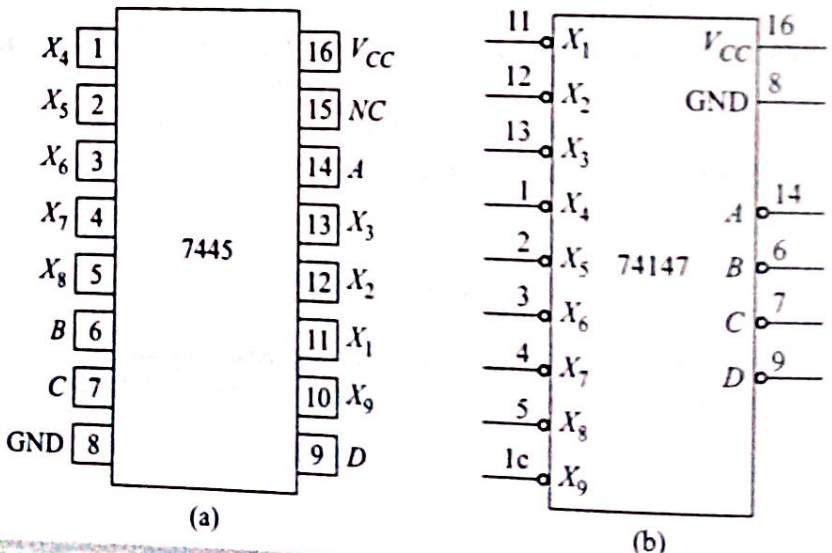


Fig. 4.26 (a) Pinout diagram of 74147. (b) Logic diagram.

Table 4.5 74147 Truth Table

Inputs									Outputs			
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	A	B	C	D
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	X	H	H	H	H	H	H	H	H	H	L	H
X	L	H	H	H	H	H	H	H	H	H	H	L
L	H	H	H	H	H	H	H	H				

Figure 4.26b shows how to draw a 74147 on a schematic diagram. As usual, the bubbles indicate active-low inputs and outputs. Table 4.5 is the truth table of a 74147. Notice the following. When all X inputs are high, all outputs are high. When X_9 is low, the $ABCD$ output is $LHHL$ (equivalent to 9 if you complement the bits). When X_8 is the only low input, $ABCD$ is $LHHH$ (equivalent to 8 if the bits are complemented). When X_7 is the only low input, $ABCD$ becomes $HLLL$ (equivalent to 7 if the bits are complemented). Continue like this through the rest of the truth table and you can see that an active-low decimal input is being converted to a complemented BCD output.

Incidentally, the 74147 is called a *priority encoder* because it gives priority to the highest-order input. You can see this by looking at Table 4.5. If all inputs X_1 through X_9 are low, the highest of these, X_9, is encoded to get an output of $LHHL$. In other words, X_9 has priority over all others. When X_9 is high, X_8 is next in line of priority and gets encoded if it is low. Working your way through Table 4.5, you can see that the highest active-low from X_9 to X_0 has priority and will control the encoding.

EXAMPLE 4.10

What is the $ABCD$ output of Fig. 4.27 when button 6 is pressed?

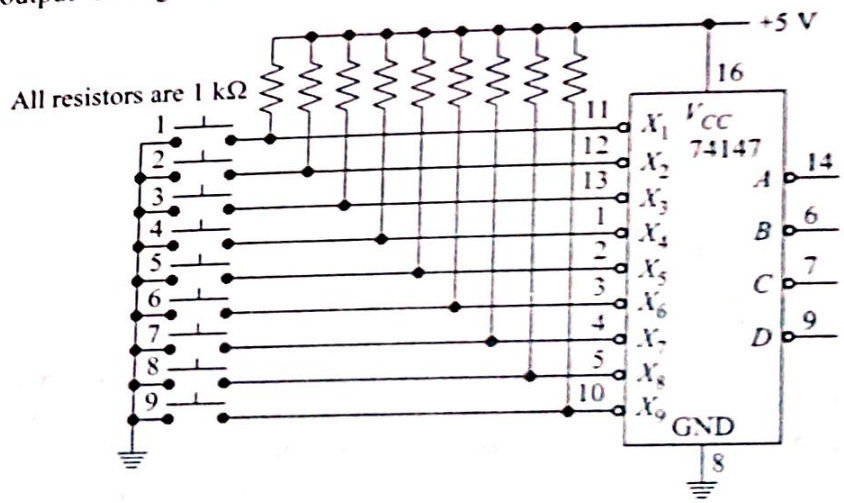


Fig. 4.27

Solution

When all switches are open, the X_1 to X_9 inputs are pulled up to the high state (+5 V). A glance at Table 4.5 indicates that the *ABCD* output is *HHHH* at this time.

When switch 6 is pressed, the X_6 input is grounded. Therefore, all X inputs are high, except for X_6. Table 4.5 indicates that the *ABCD* output is *HLLH*, which is equivalent to 6 when the output bits are complemented.

EXAMPLE 4.11

Design a priority encoder the truth table of which is shown in Fig. 4.28a. The order of priority for three inputs is $X_1 > X_2 > X_3$. However, if the encoder is not enabled by S or all the inputs are inactive the output $AB = 00$.

Solution

Figure 4.28b and Fig. 4.28c show the Karnaugh map for output A and B respectively. Note that, we have used a different notation for input variables in these maps. Compare this with notations presented in previous chapters. You will find a variable with prime is presented by 0 and if it is not primed is represented by 1. Then taking groups of 1s we get the design equations as shown in the figure. The logic circuits for output A and B can be directly drawn from these equations.

Intput				Output	
S	X_1	X_2	X_3	A	B
0	×	×	×	0	0
1	1	×	×	0	1
1	0	1	×	1	0
1	0	0	1	1	1
1	0	0	0	0	0

(a)

X_2X_3 \ SX_1	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	0	1
10	0	0	0	1

$A = S\bar{X}_1X_3 + S\bar{X}_1X_2$

(b)

X_2X_3 \ SX_1	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	0	0	1	0
10	0	0	1	0

$B = SX_1 + S\bar{X}_2X_3$

(c)

Fig. 4.28 *Design of a priority encoder.*

SELF-TEST

12. What is an encoder?
13. What type of encoder is the TTL 74147?

4.7 EXCLUSIVE-OR GATES

The *exclusive-OR gate* has a high output only when an odd number of inputs is high. Figure 4.29 shows how to build an exclusive-OR gate. The upper AND gate forms the product $\bar{A}B$, while the lower one produces $A\bar{B}$. Therefore, the output of the OR gate is

$$Y = \bar{A}B + A\bar{B}$$