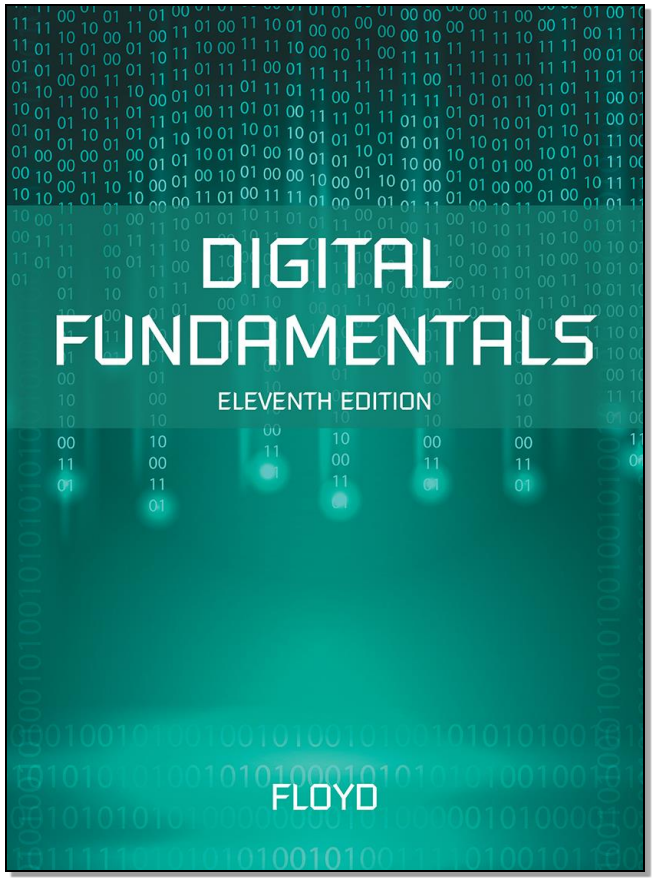
Digital Fundamentals

ELEVENTH EDITION



CHAPTER 6

Functions of Combinational Logic

Logic symbol for a half-adder

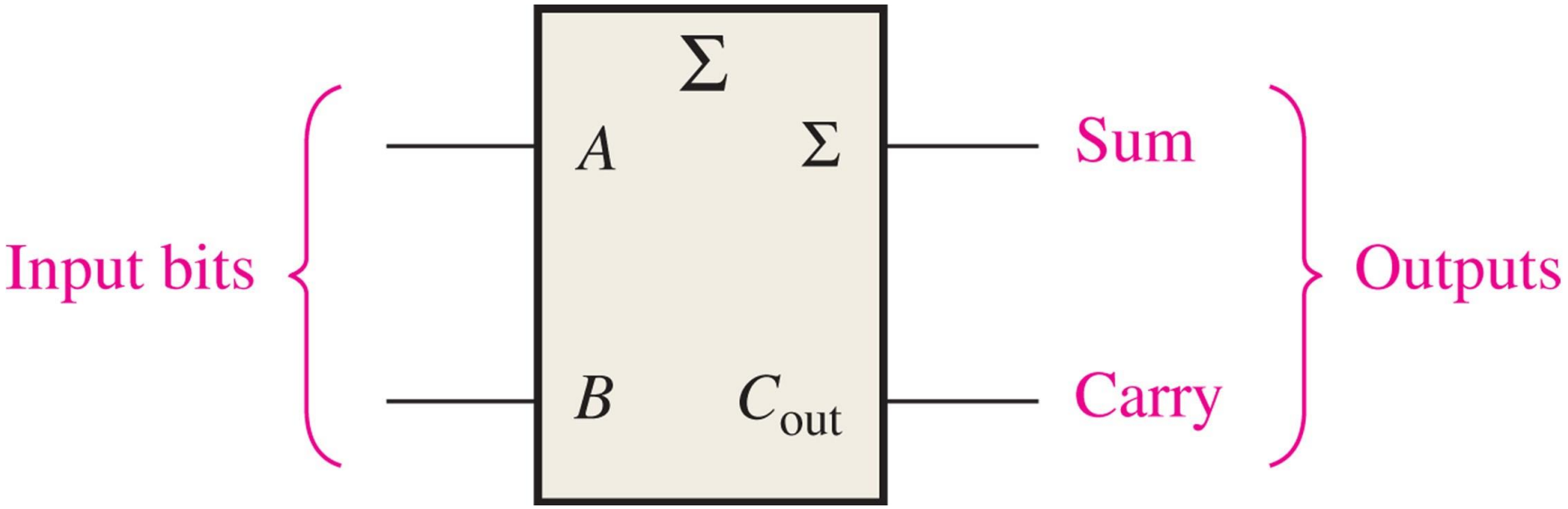


TABLE 6–1

Half-adder truth table.

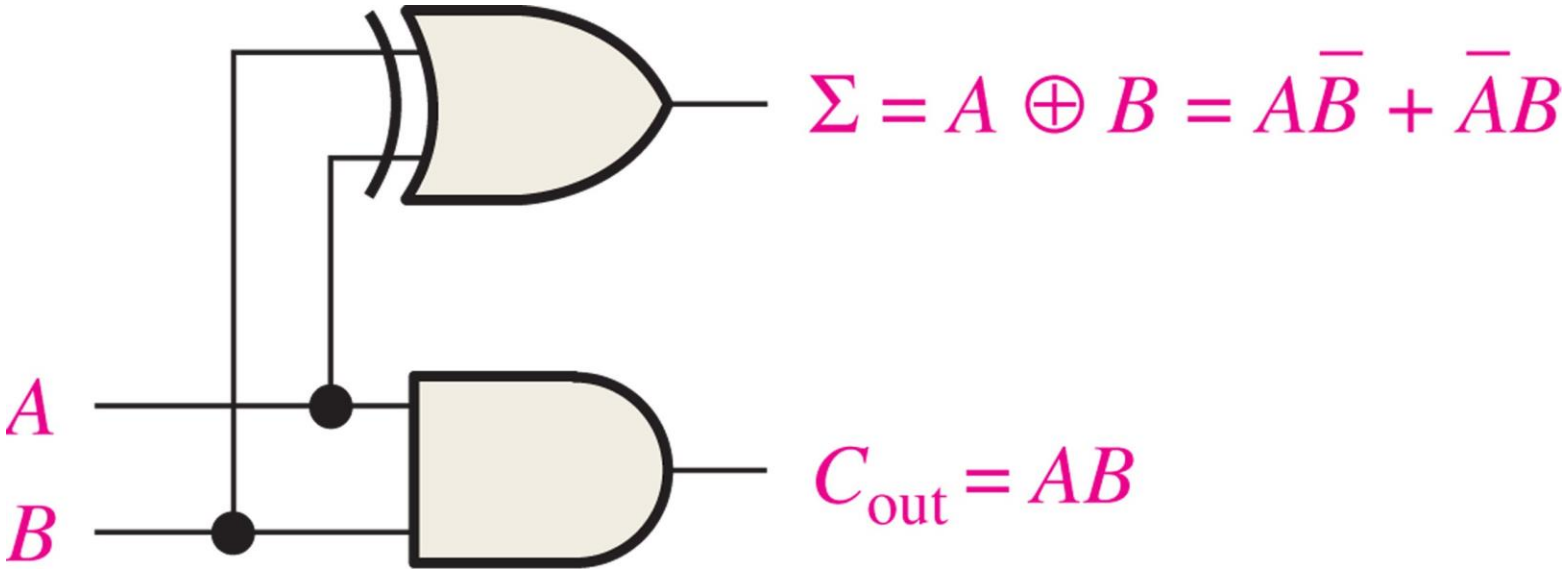
A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum

C_{out} = output carry

A and B = input variables (operands)

FIGURE 6-2 Half-adder logic diagram.



Logic symbol for a full-adder

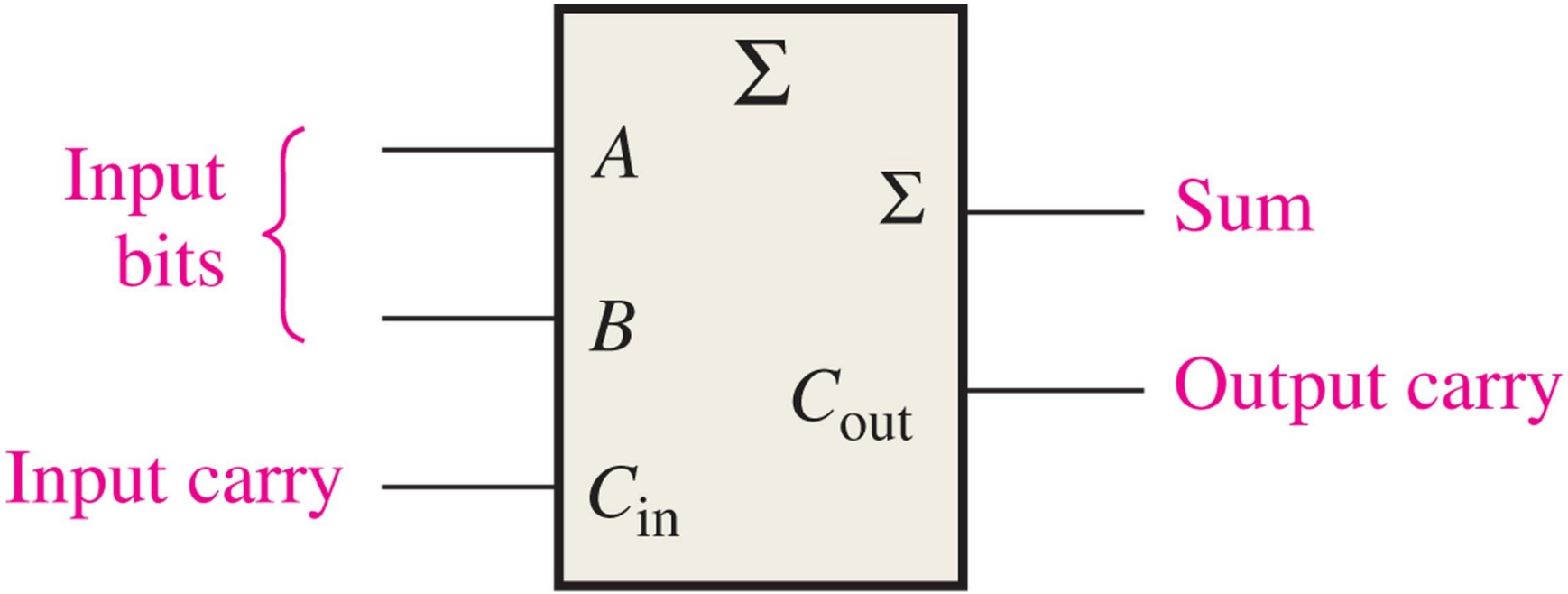


TABLE 6–2

Full-adder truth table.

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

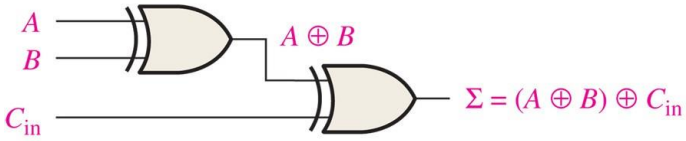
C_{in} = input carry, sometimes designated as CI

C_{out} = output carry, sometimes designated as CO

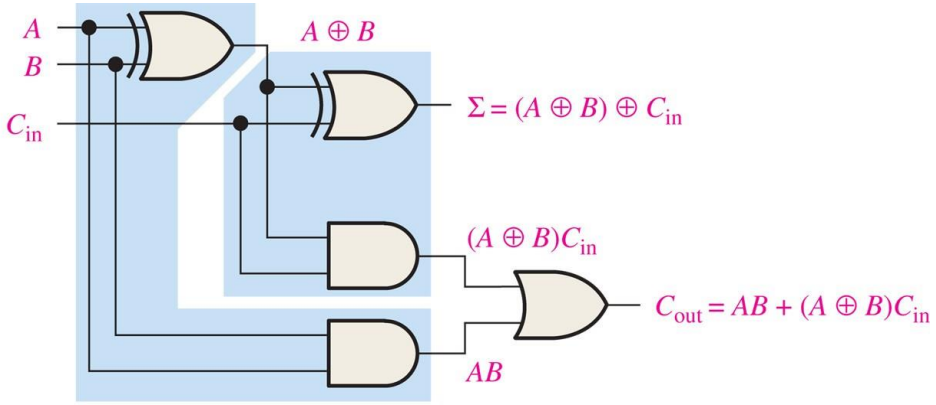
Σ = sum

A and B = input variables (operands)

Full-adder logic

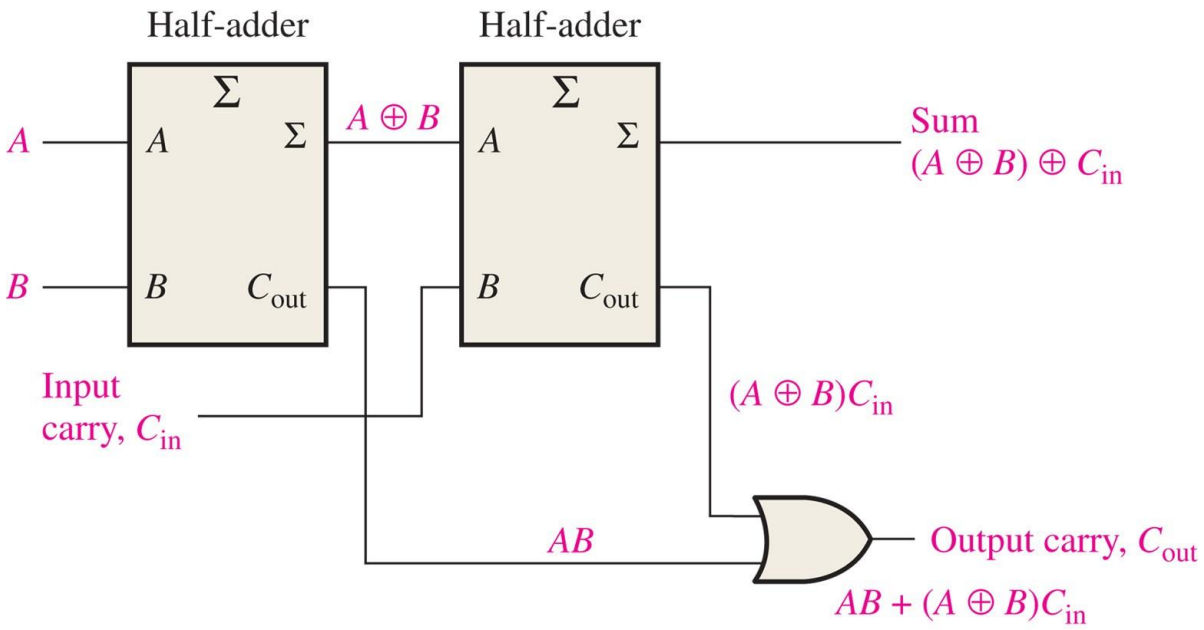


(a) Logic required to form the sum of three bits

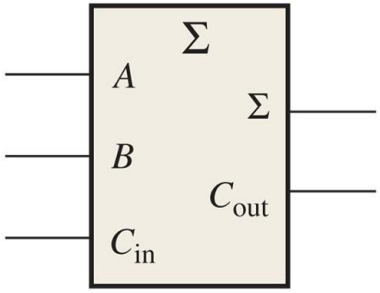


(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)

FIGURE 6-5 Full-adder implemented with half-adders.



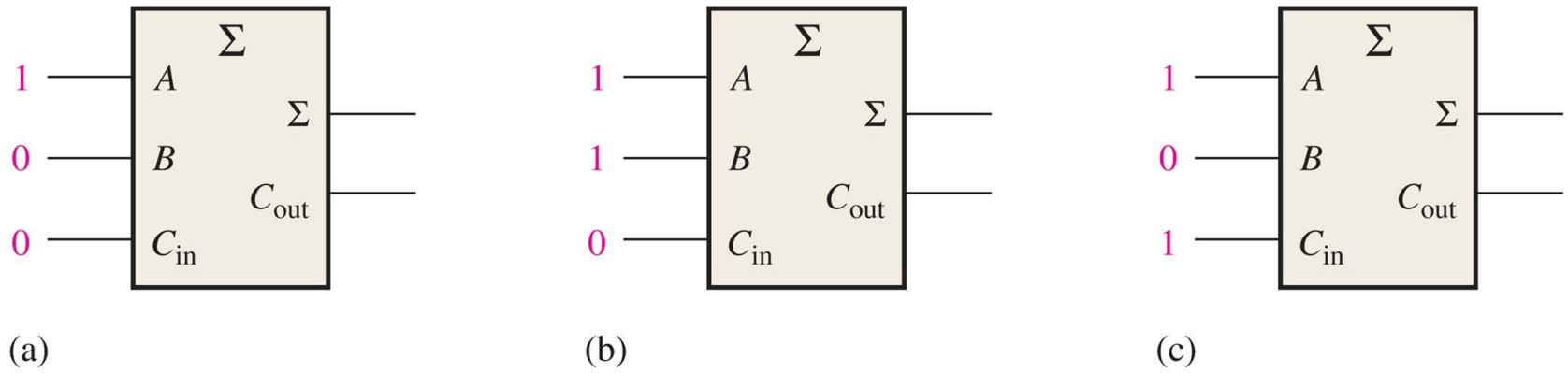
(a) Arrangement of two half-adders to form a full-adder



(b) Full-adder logic symbol

For each of the three full-adders in Figure 6–6, determine the outputs for the inputs shown.

FIGURE 6-6



Solution

(a) The input bits are $A = 1$, $B = 0$, and $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ with no carry}$$

Therefore, $\Sigma = \mathbf{1}$ and $C_{out} = \mathbf{0}$.

(b) The input bits are $A = 1$, $B = 1$, and $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ with a carry of 1}$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

(c) The input bits are $A = 1$, $B = 0$, and $C_{in} = 1$.

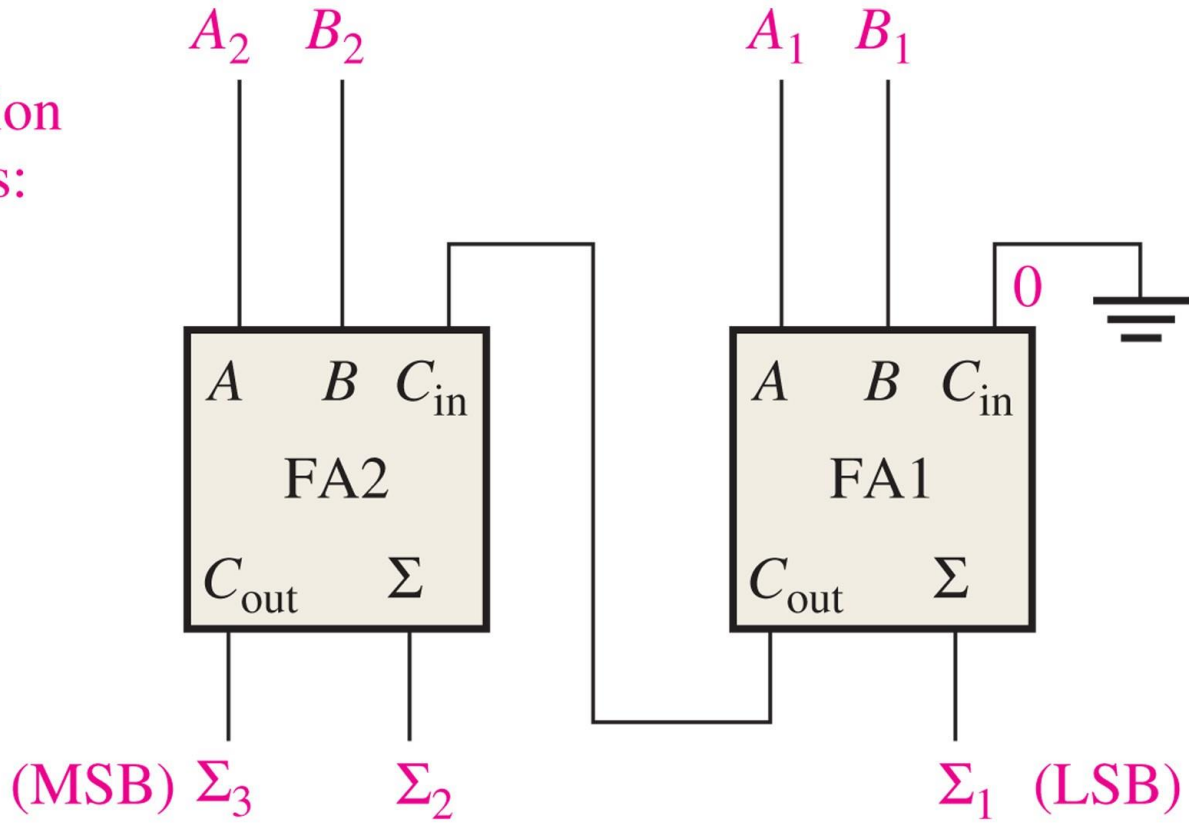
$$1 + 0 + 1 = 0 \text{ with a carry of 1}$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

Block diagram of a basic 2-bit parallel adder using two full-adders

General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$



Determine the sum generated by the 3-bit parallel adder in Figure 6–8 and show the intermediate carries when the binary numbers 101 and 011 are being added.

FIGURE 6-8

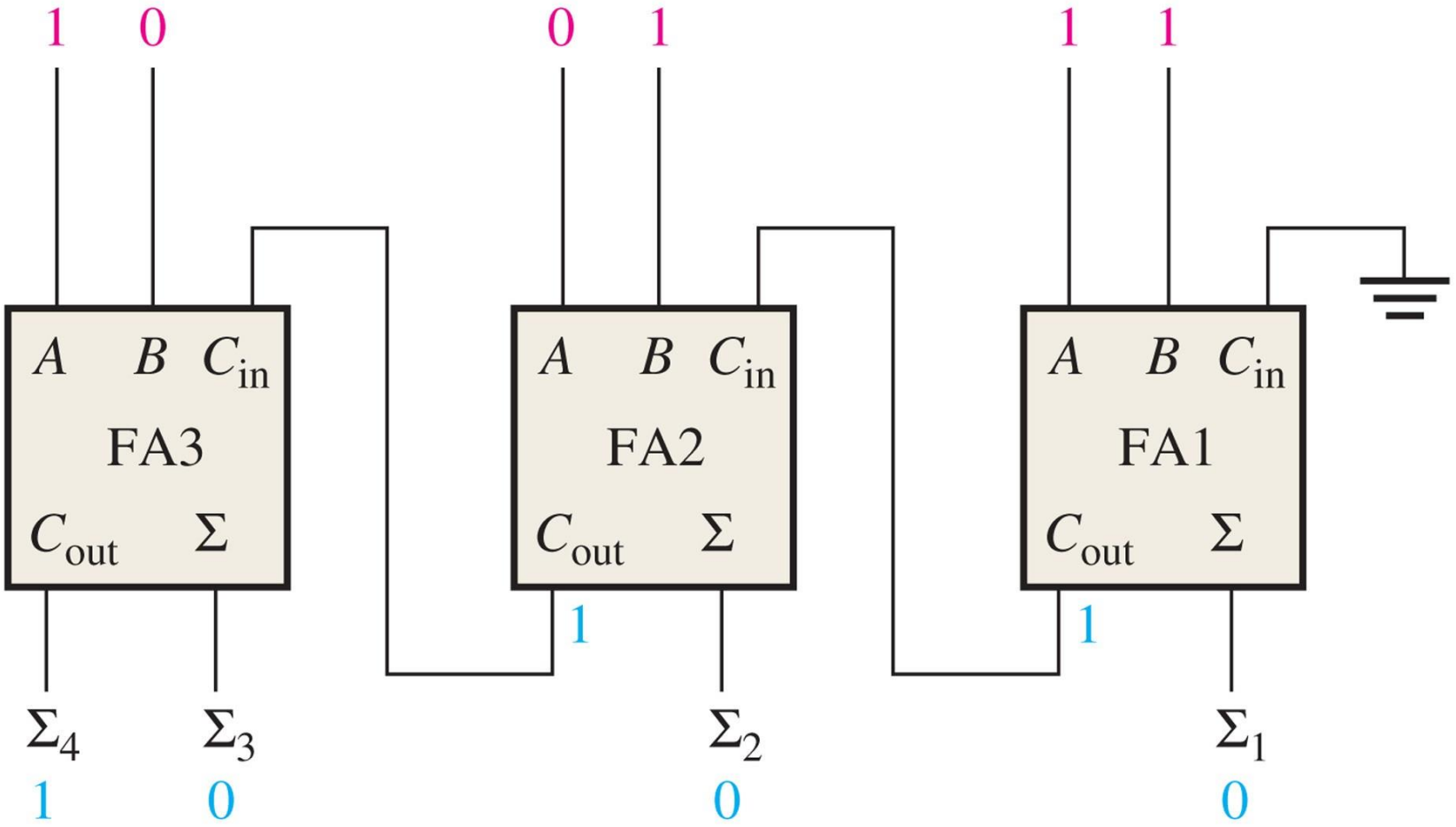
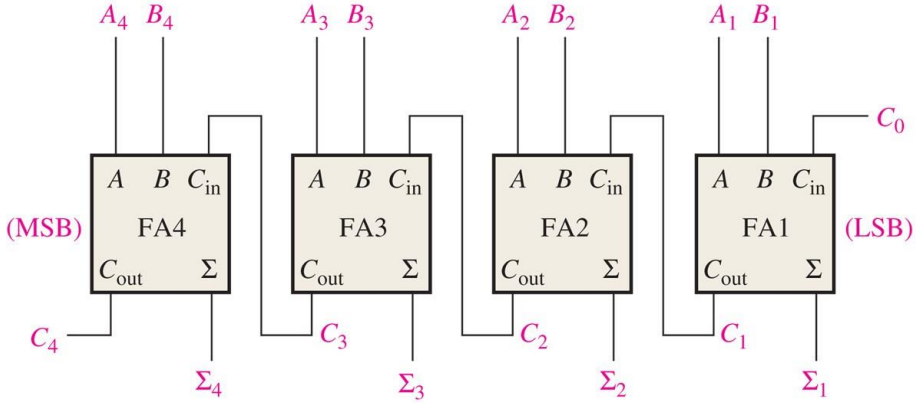
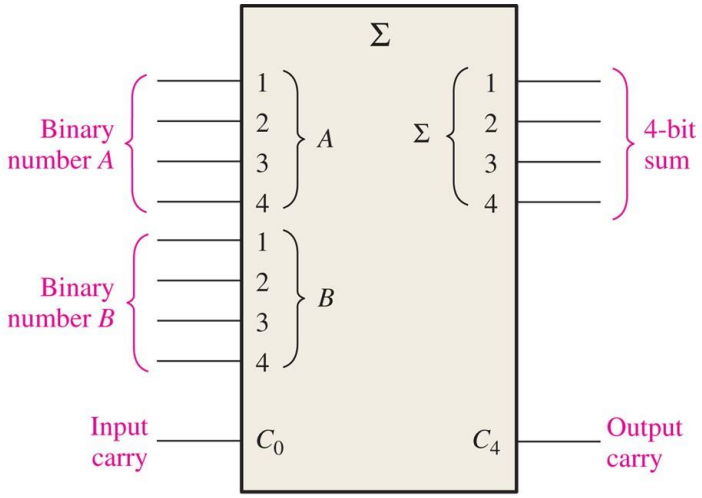


FIGURE 6-9 A 4-bit parallel adder.



(a) Block diagram



(b) Logic symbol

TABLE 6–3

Truth table for each stage of a 4-bit parallel adder.

C_{n-1}	A_n	B_n	Σ_n	C_n
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

Use the 4-bit parallel adder truth table (Table 6–3) to find the sum and output carry for the addition of the following two 4-bit numbers if the input carry (C_{n-1}) is 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{and} \quad B_4B_3B_2B_1 = 1100$$

Solution

For $n = 1$: $A_1 = 0$, $B_1 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_1 = \mathbf{0} \quad \text{and} \quad C_1 = 0$$

For $n = 2$: $A_2 = 0$, $B_2 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_2 = \mathbf{0} \quad \text{and} \quad C_2 = 0$$

For $n = 3$: $A_3 = 1$, $B_3 = 1$, and $C_{n-1} = 0$. From the 4th row of the table,

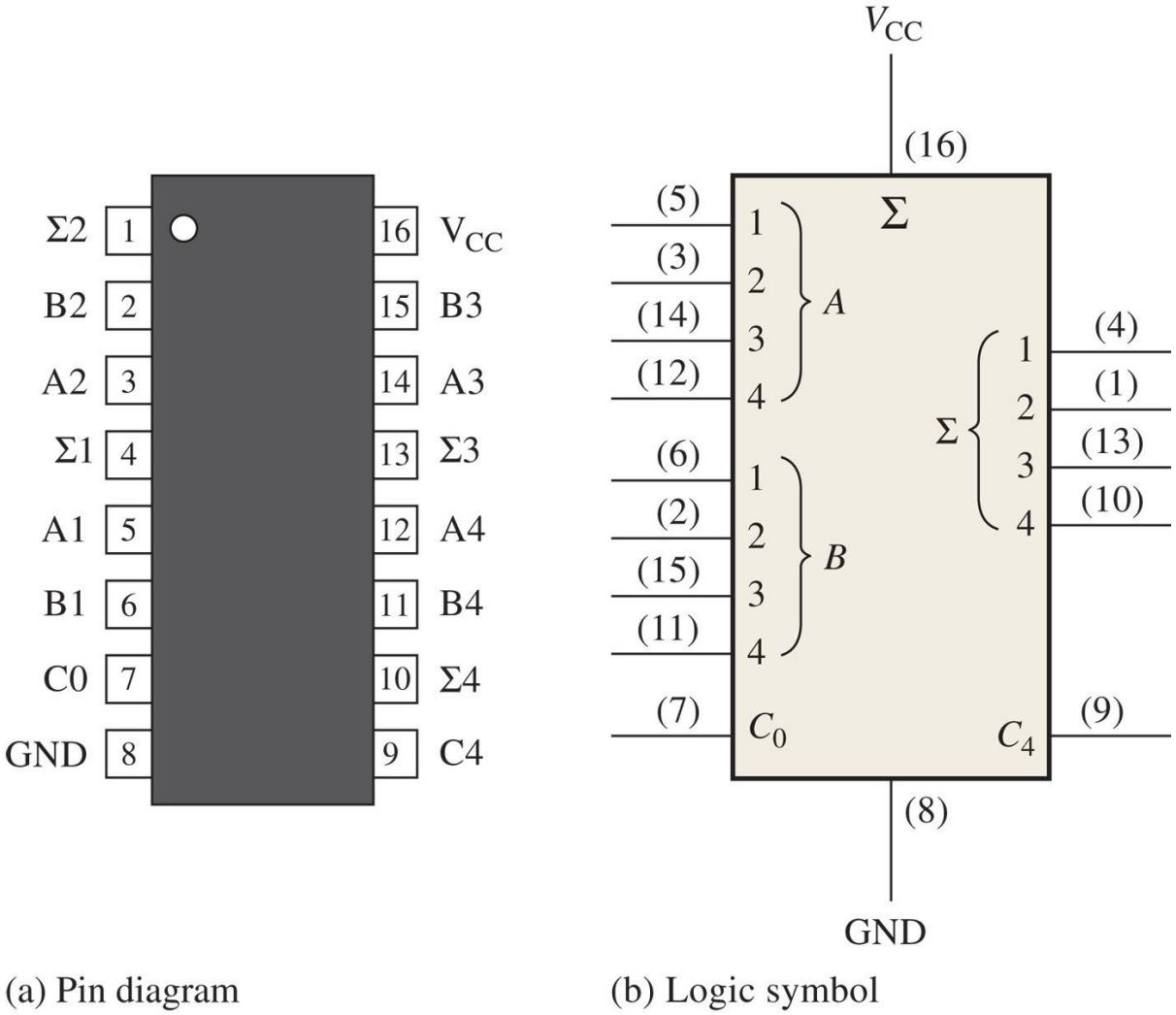
$$\Sigma_3 = \mathbf{0} \quad \text{and} \quad C_3 = 1$$

For $n = 4$: $A_4 = 1$, $B_4 = 1$, and $C_{n-1} = 1$. From the last row of the table,

$$\Sigma_4 = \mathbf{1} \quad \text{and} \quad C_4 = \mathbf{1}$$

C_4 becomes the output carry; the sum of 1100 and 1100 is 11000.

FIGURE 6-10 The 74HC283/74LS283 4-bit parallel adder.



(a) Pin diagram

(b) Logic symbol

FIGURE 6-11 Cascading of two 4-bit adders to form an 8-bit adder.

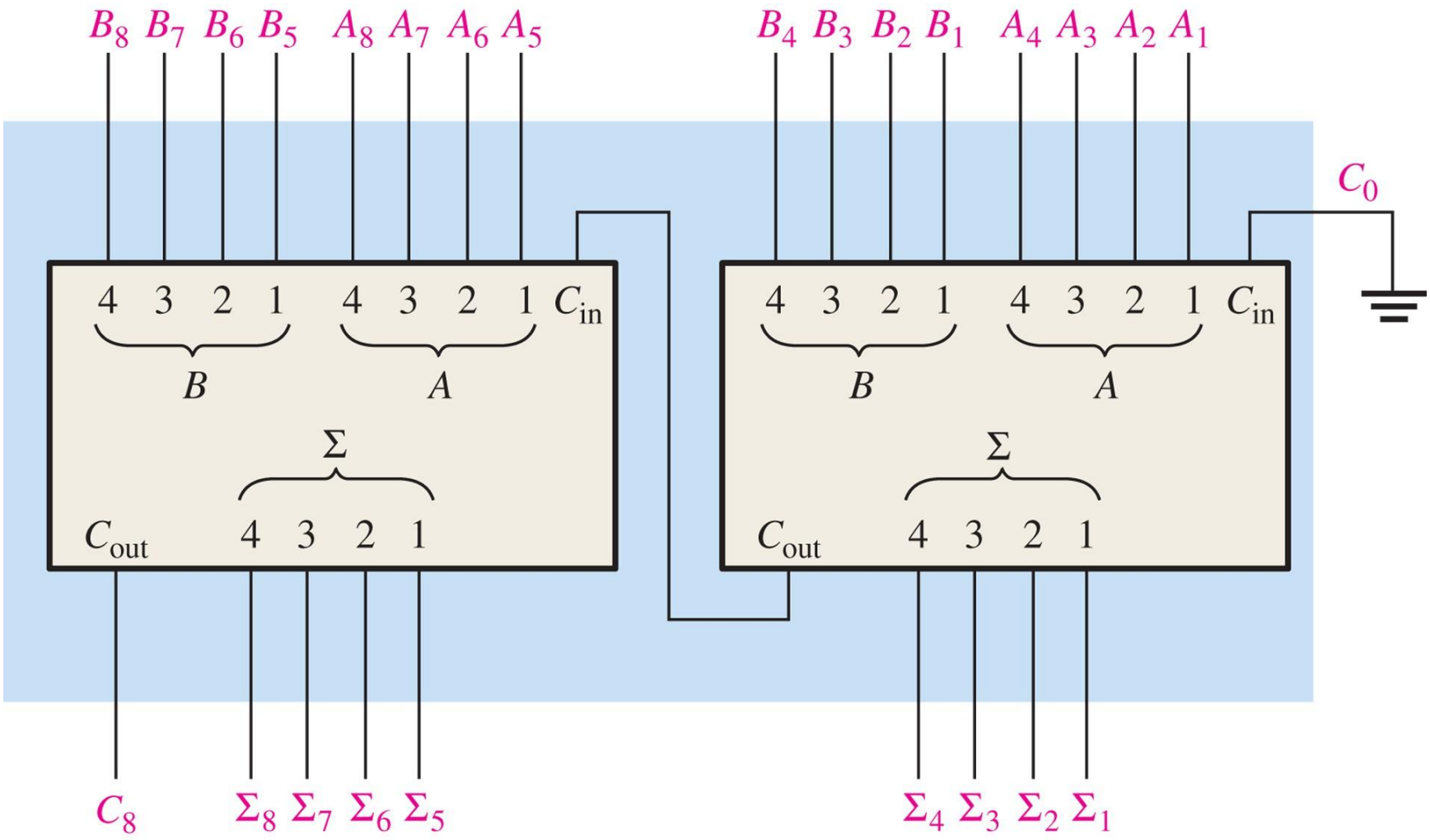
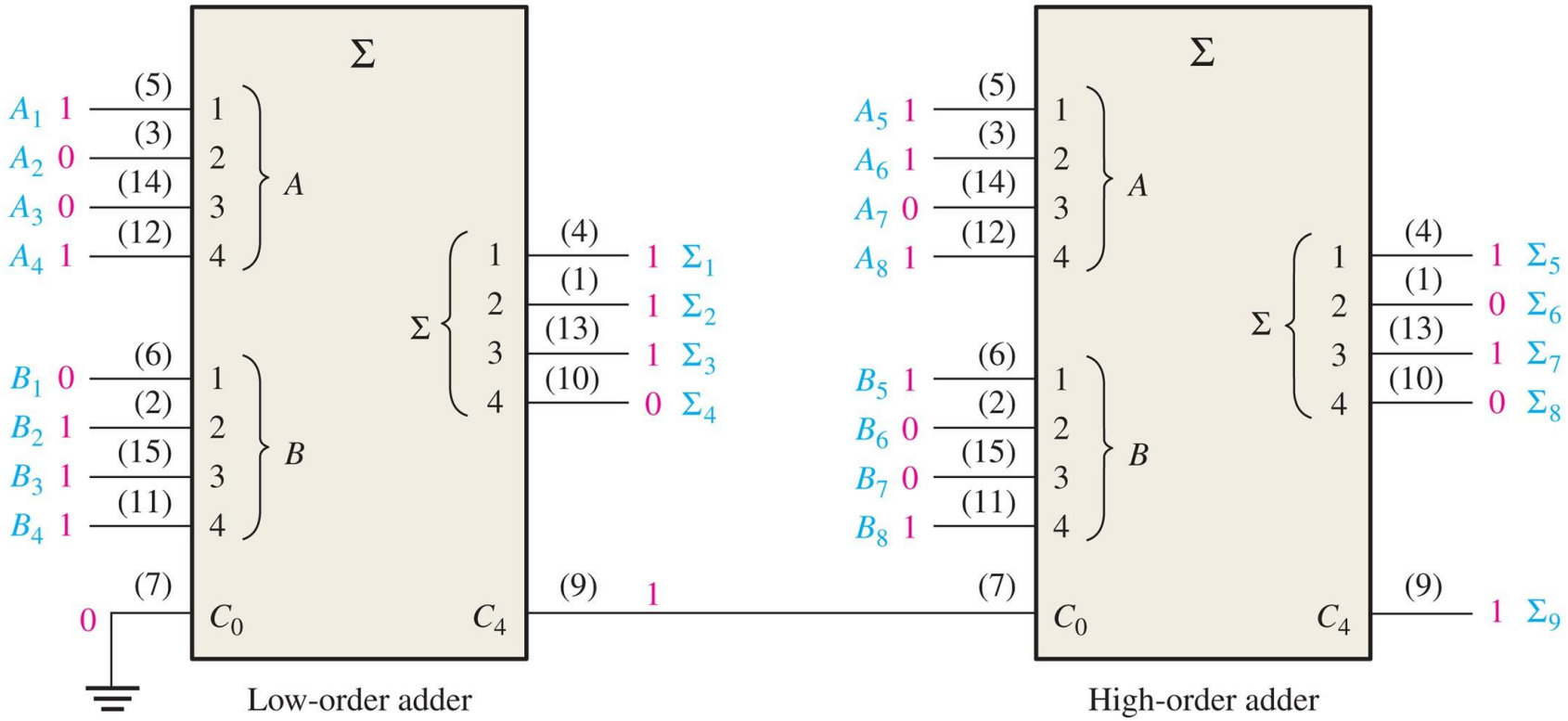


FIGURE 6-12 Two 74HC283 adders connected as an 8-bit parallel adder (pin numbers are in parentheses).



A voting system using full-adders and parallel binary adders

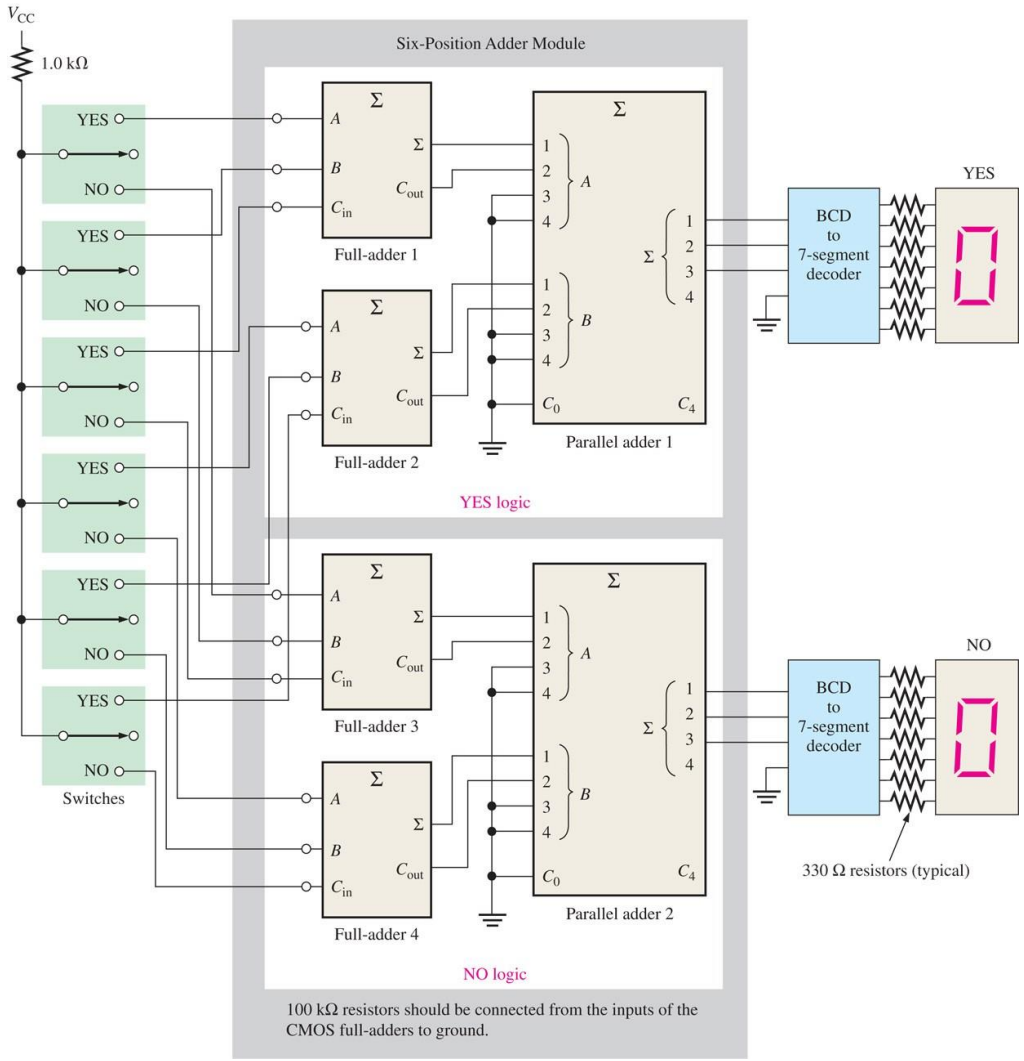


FIGURE 6-14 A 4-bit parallel ripple carry adder showing "worst-case" carry propagation delays.

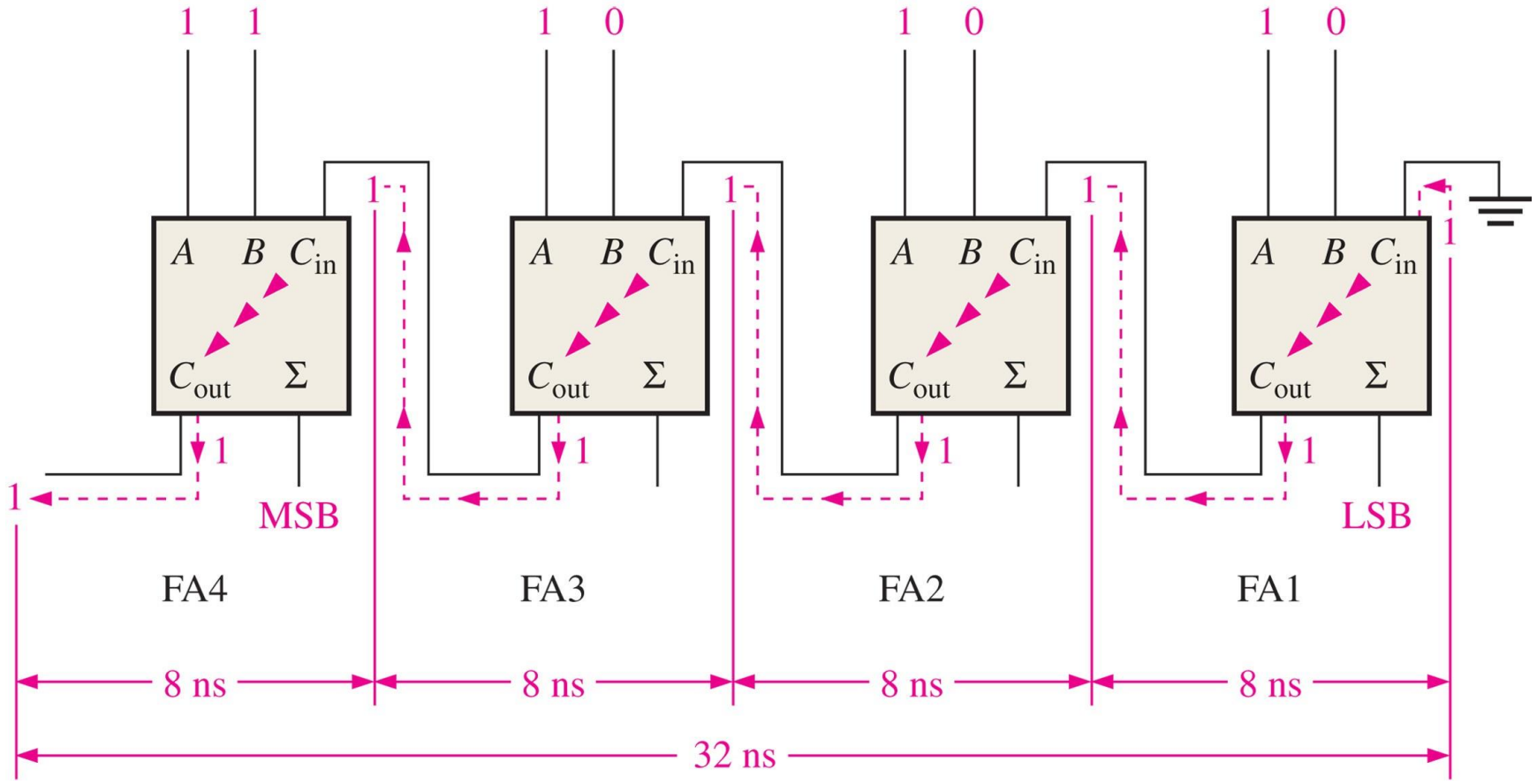


FIGURE 6-15 Illustration of conditions for carry generation and carry propagation.

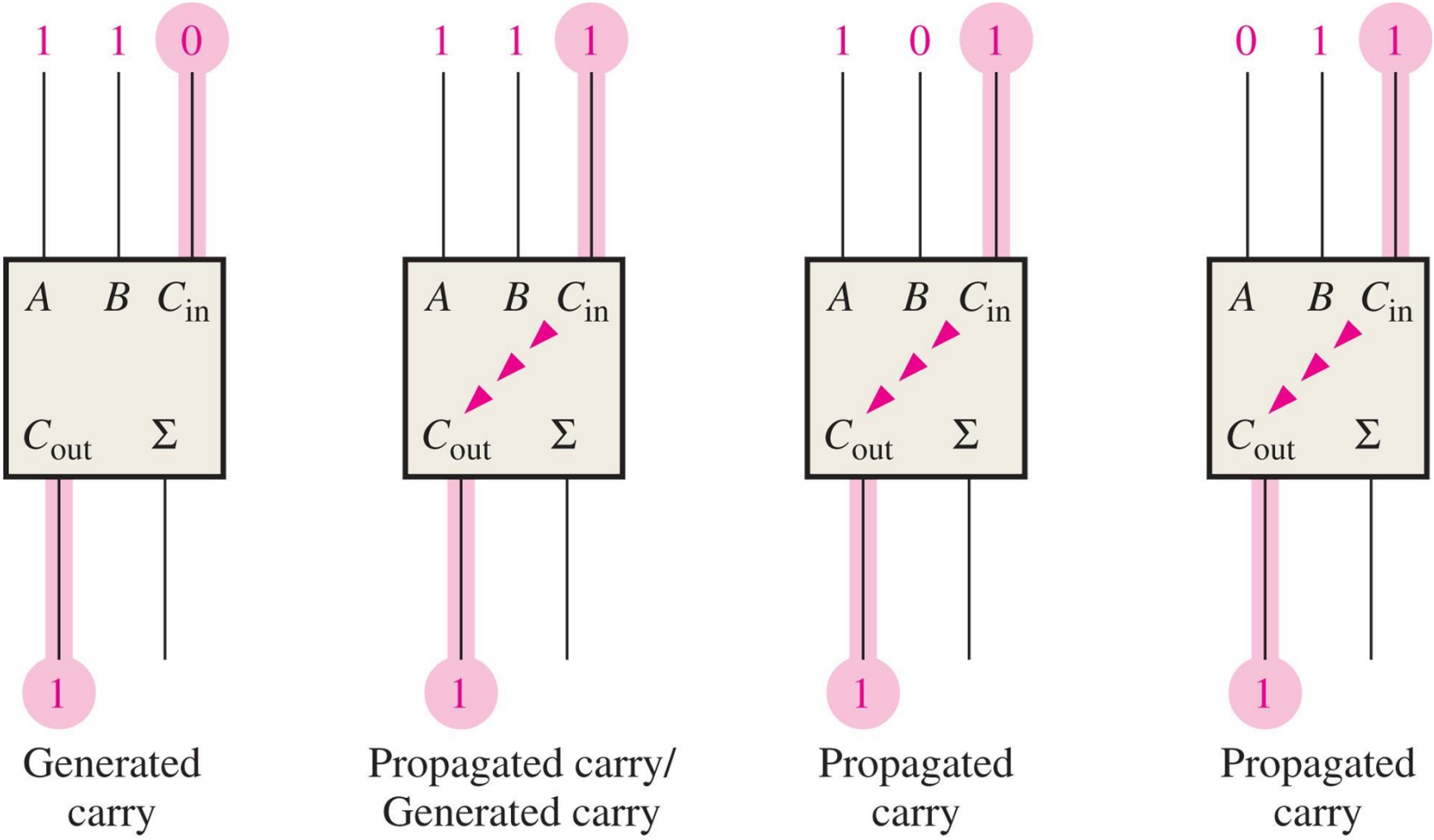


FIGURE 6-16 Carry generation and carry propagation in terms of the input bits to a 4-bit adder.

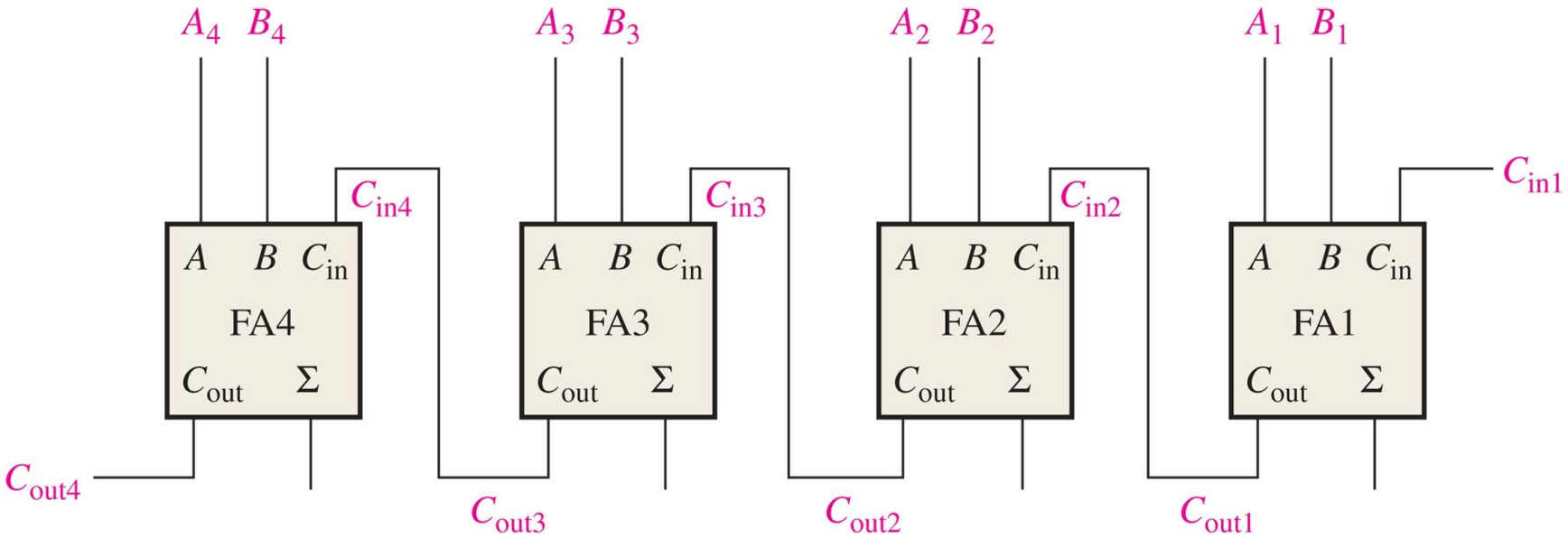
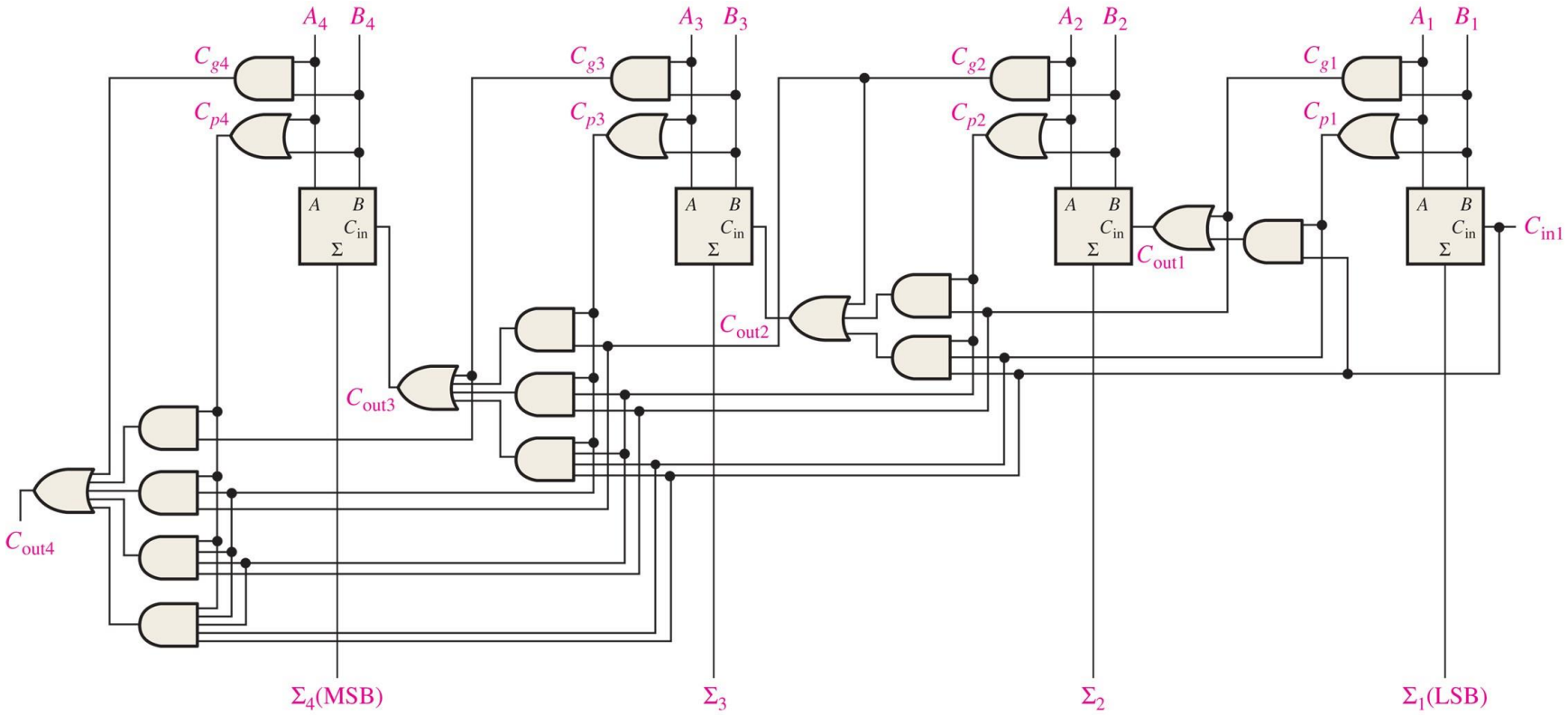


FIGURE 6-17 Logic diagram for a 4-stage look-ahead carry adder.



Comparators

FIGURE 6-18 Basic comparator operation.

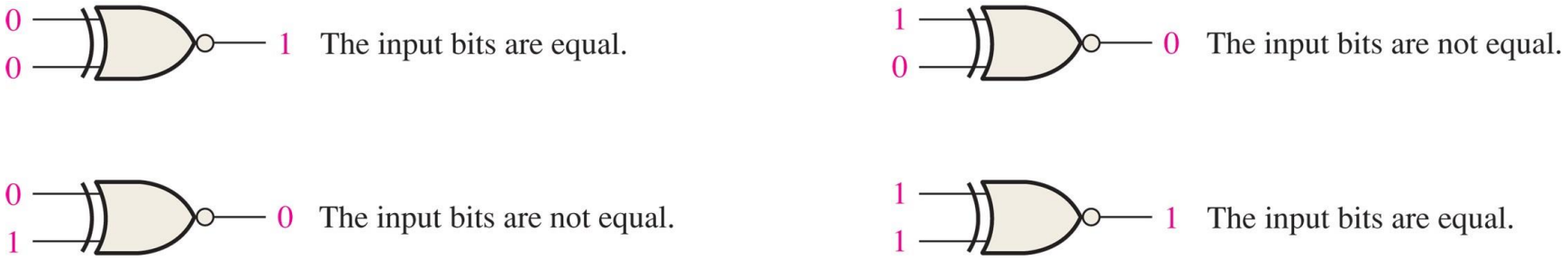
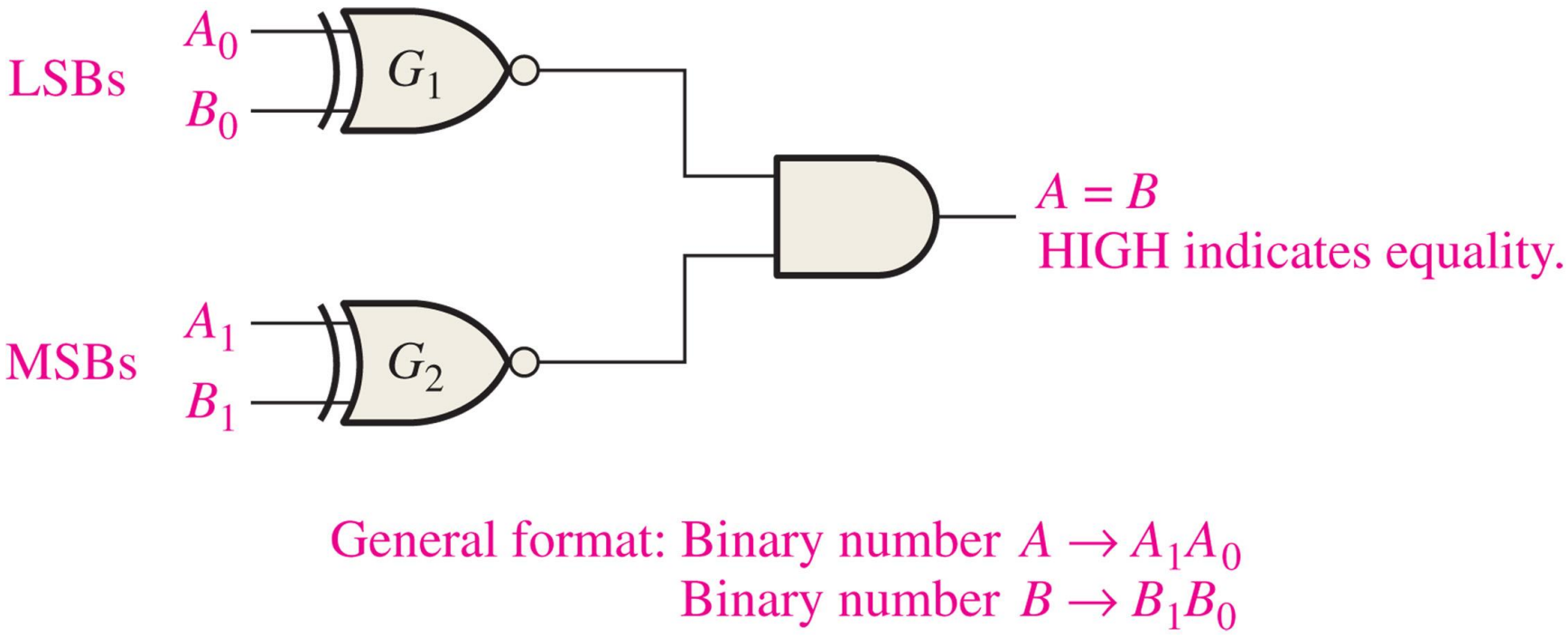


FIGURE 6-19 Logic diagram for equality comparison of two 2-bit numbers.



General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$

Apply each of the following sets of binary numbers to the comparator inputs in Figure 6–20, and determine the output by following the logic levels through the circuit.

(a) 10 and 10 **(b)** 11 and 10

FIGURE 6-20

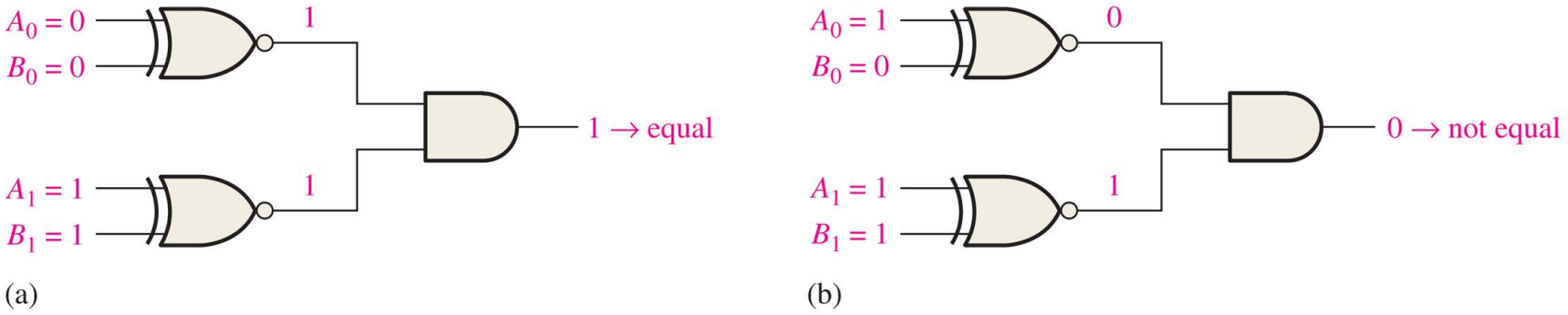


FIGURE 6-21 Logic symbol for a 4-bit comparator with inequality indication.

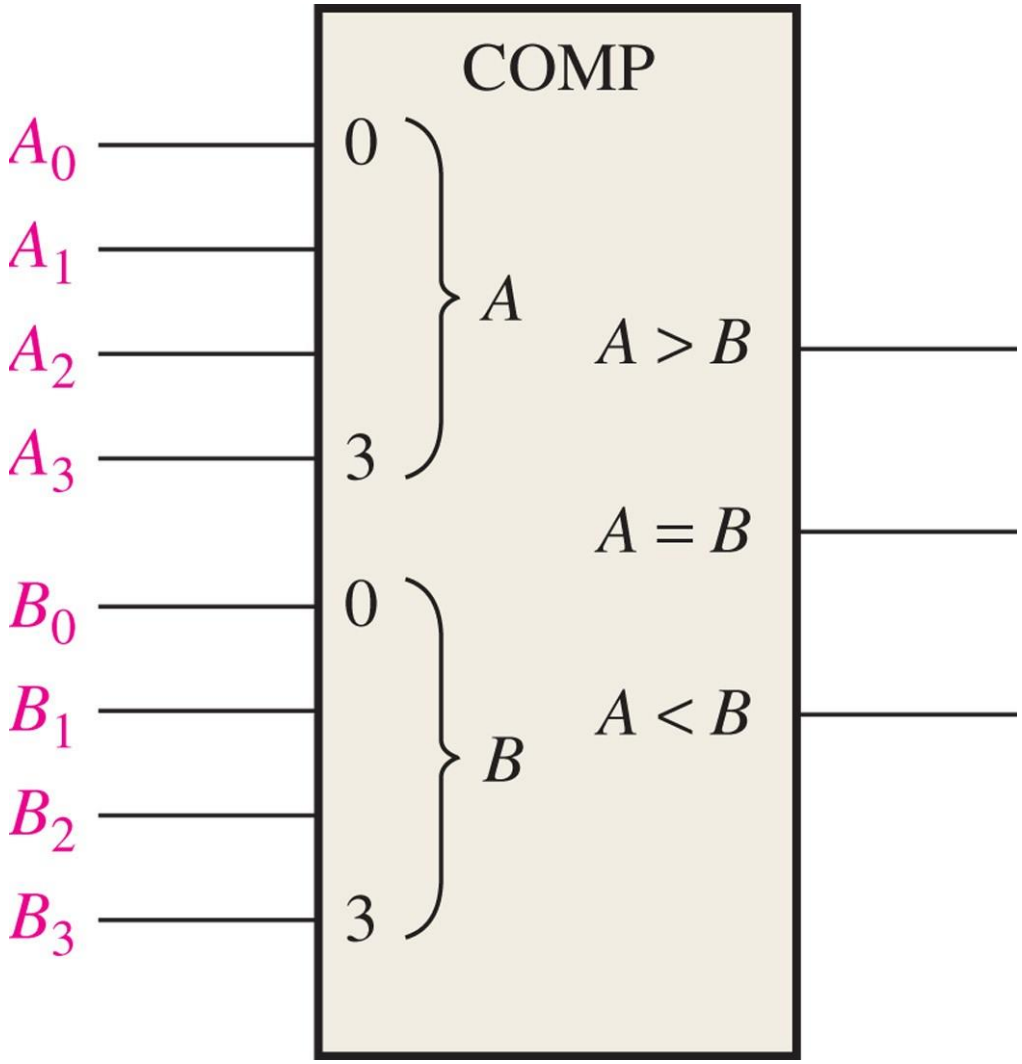
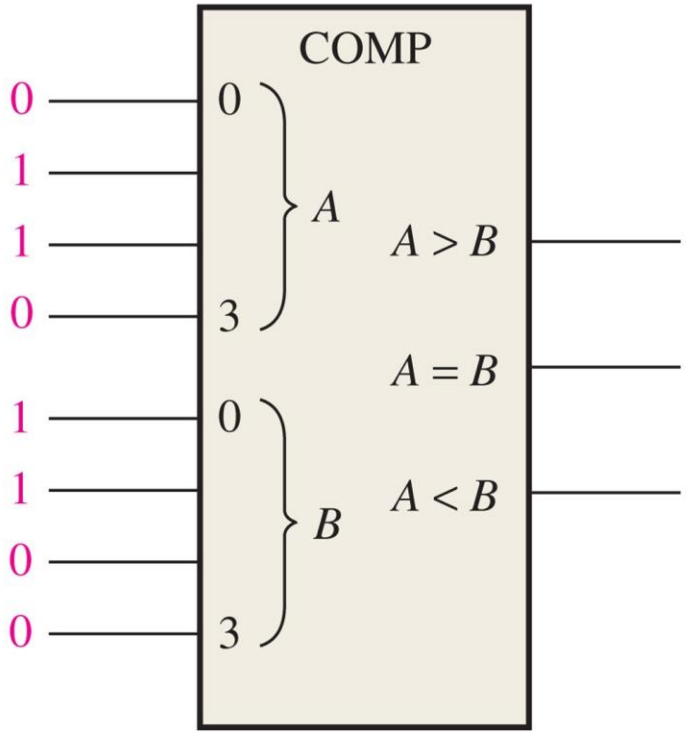


FIGURE 6-22



The number on the A inputs is 0110 and the number on the B inputs is 0011. The A > B output is HIGH and the other outputs are LOW.

FIGURE 6-23 The 74HC85/74LS85 4-bit magnitude comparator.

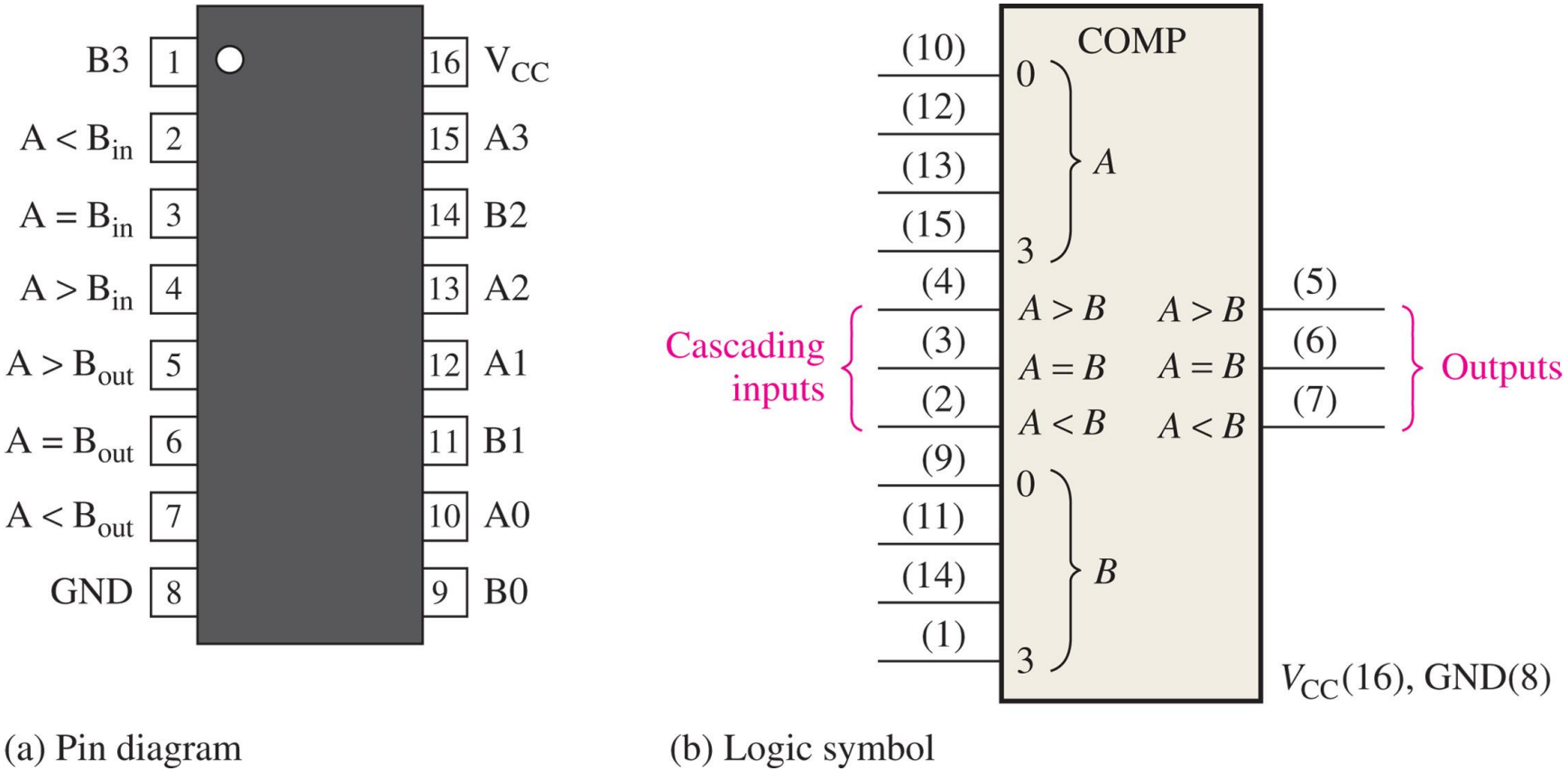
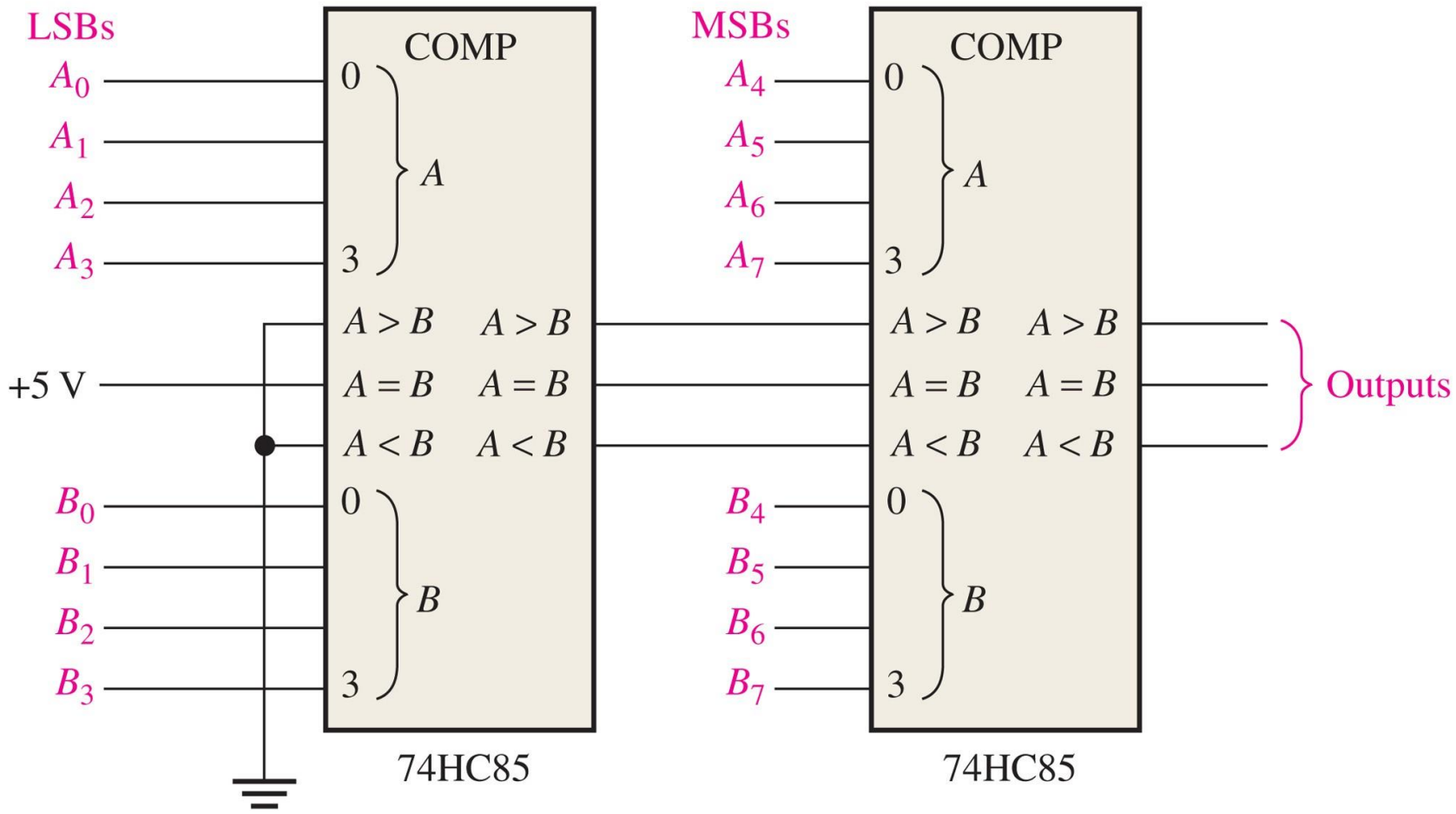


FIGURE 6-25 An 8-bit magnitude comparator using two 74HC85s.



Decoders

FIGURE 6-26 Decoding logic for the binary code 1001 with an active-HIGH output.

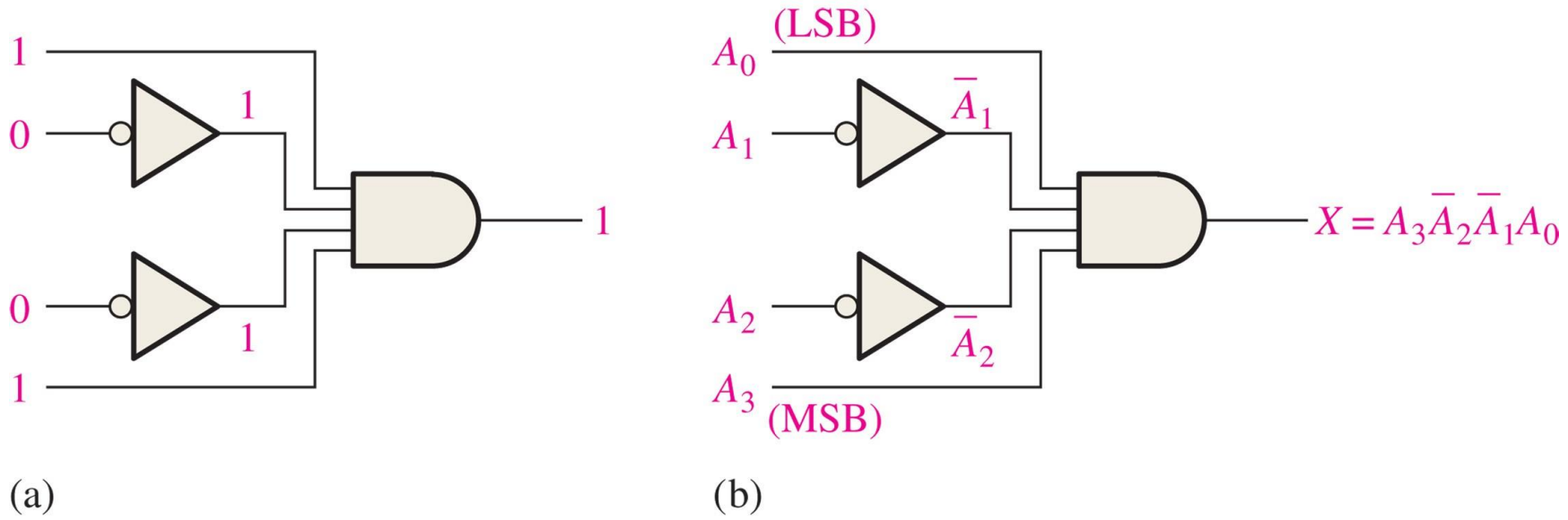


FIGURE 6-27 Decoding logic for producing a HIGH output when 1011 is on the inputs.

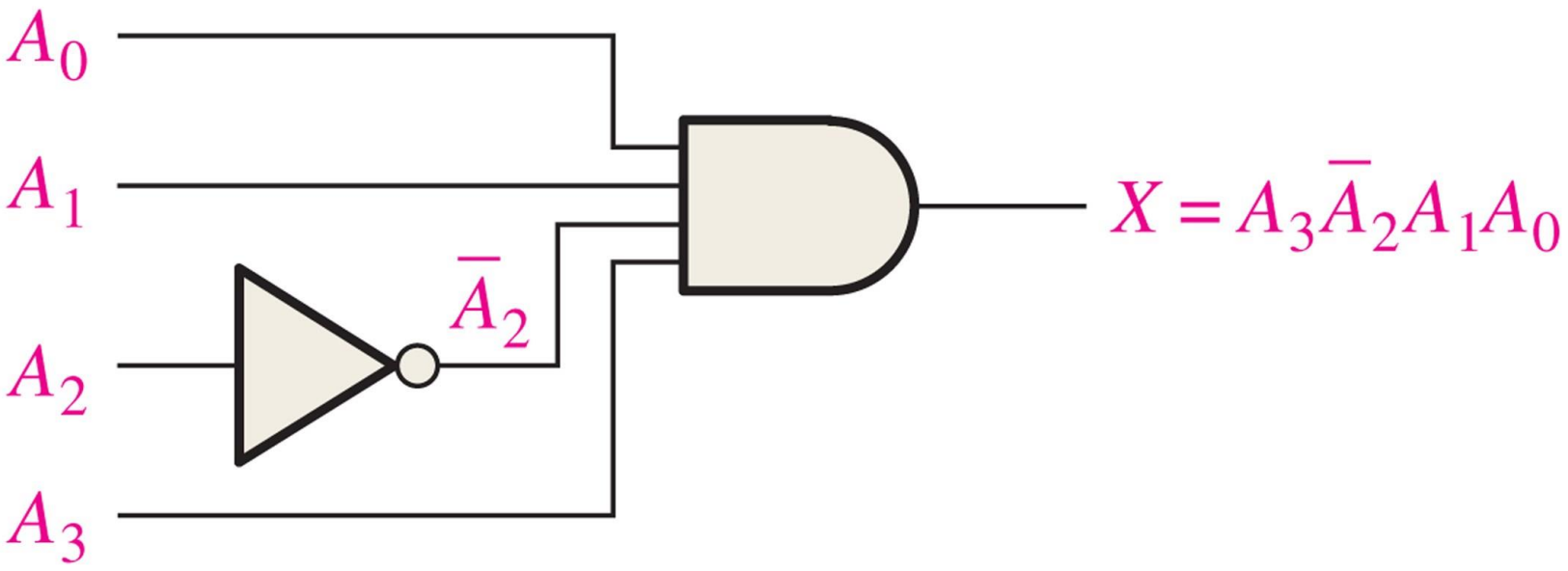


TABLE 6–4

Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with active-LOW outputs.

Decimal Digit	Binary Inputs				Decoding Function	Outputs														
	A_3	A_2	A_1	A_0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

FIGURE 6-28 Logic symbol for a 4-line-to-16-line (1-of-16) decoder.

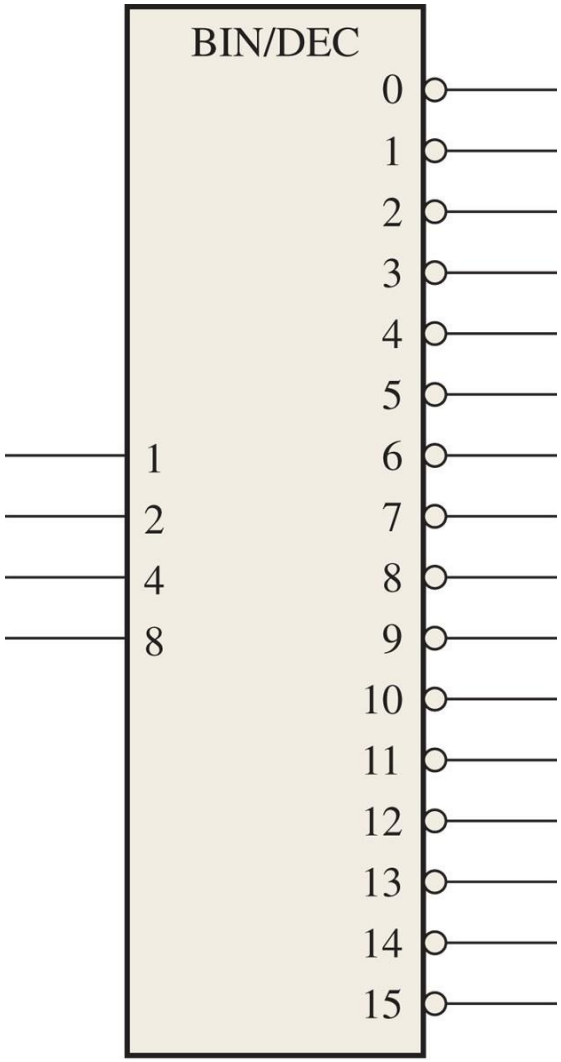


FIGURE 6-29 The 74HC154 1-of-16 decoder.

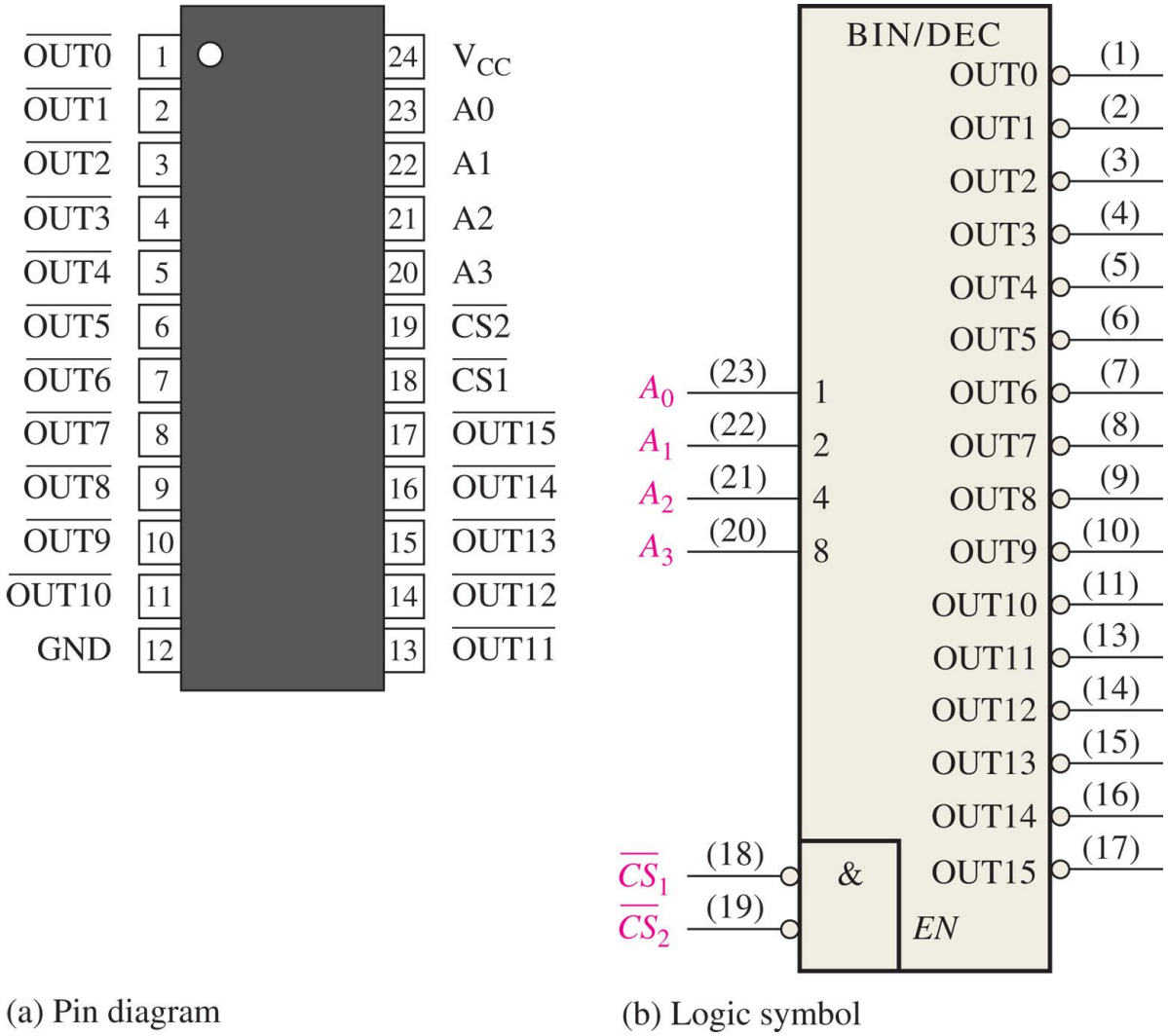


FIGURE 6-30 A 5-bit decoder using 74HC154s.

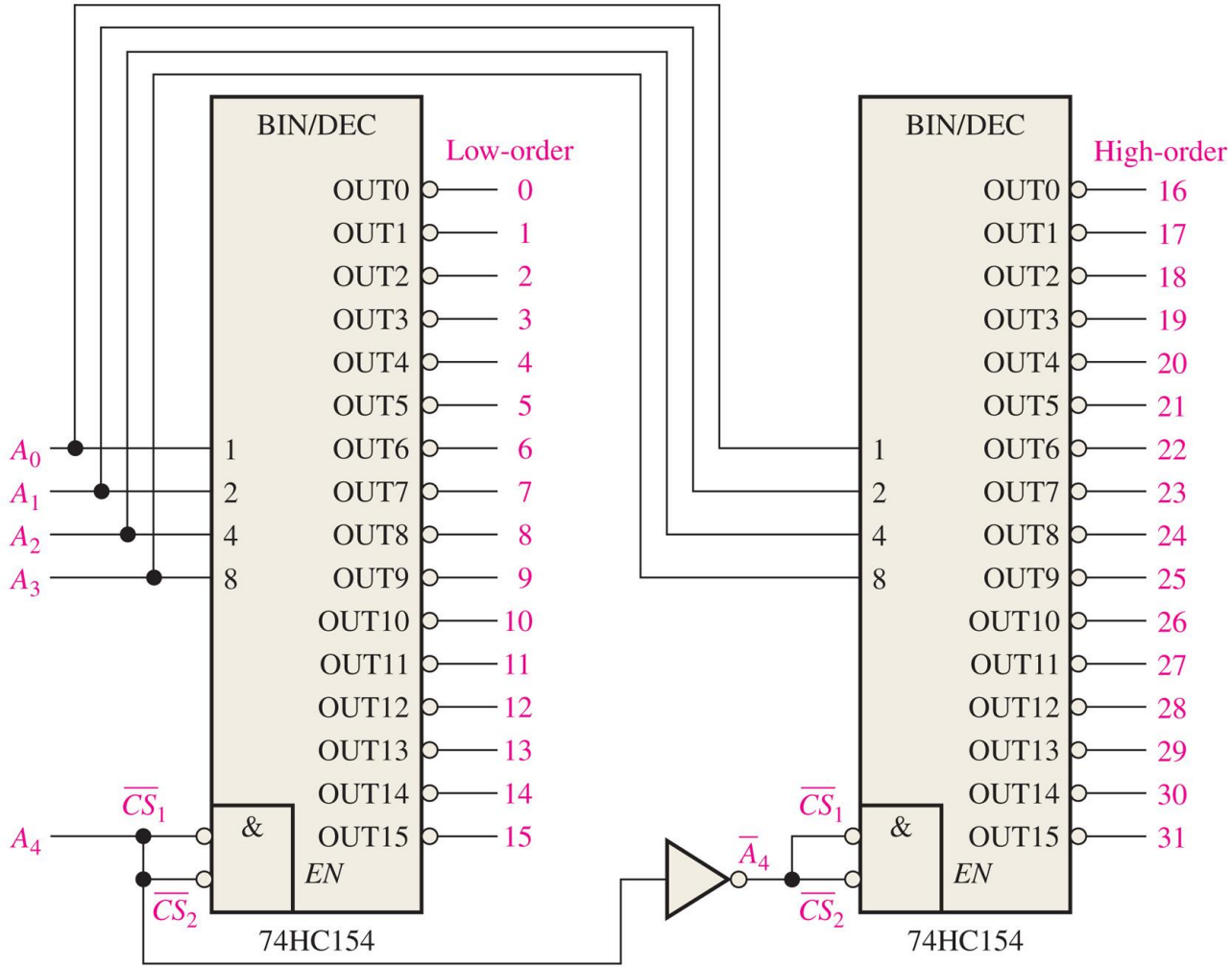


TABLE 6–5

BCD decoding functions.

Decimal Digit	BCD Code A_3	A_2	A_1	A_0	Decoding Function
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

Fixed-Function Device The 74HC42 is a fixed-function IC decoder with four BCD inputs and ten active-LOW decimal outputs. The logic symbol is shown in Figure 6-31

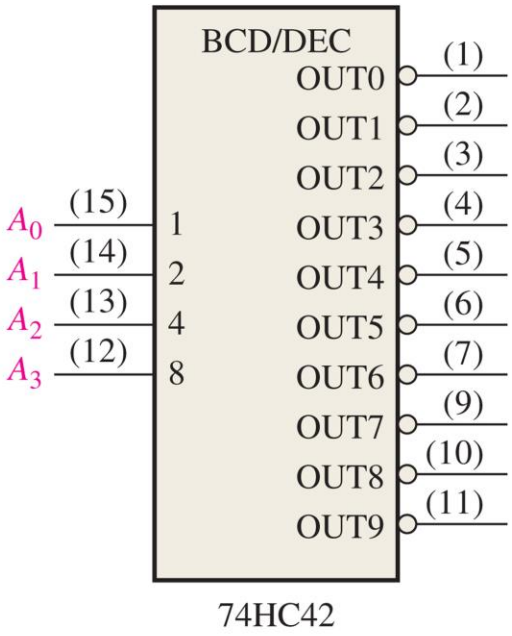


FIGURE 6-31 The 74HC42 BCD-to-decimal decoder.

FIGURE 6-32

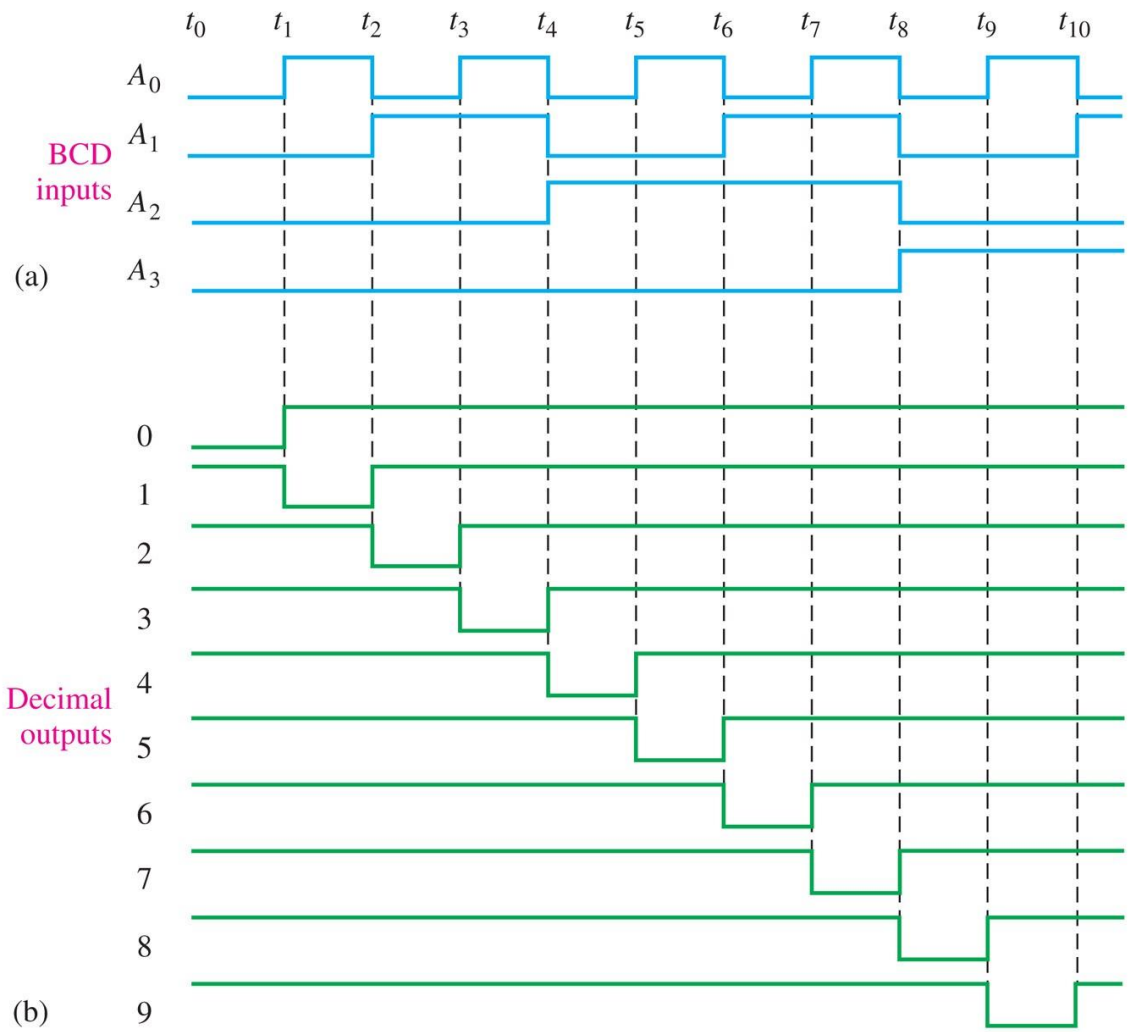


FIGURE 6-33 Logic symbol for a BCD-to-7-segment decoder/driver with active-LOW outputs.

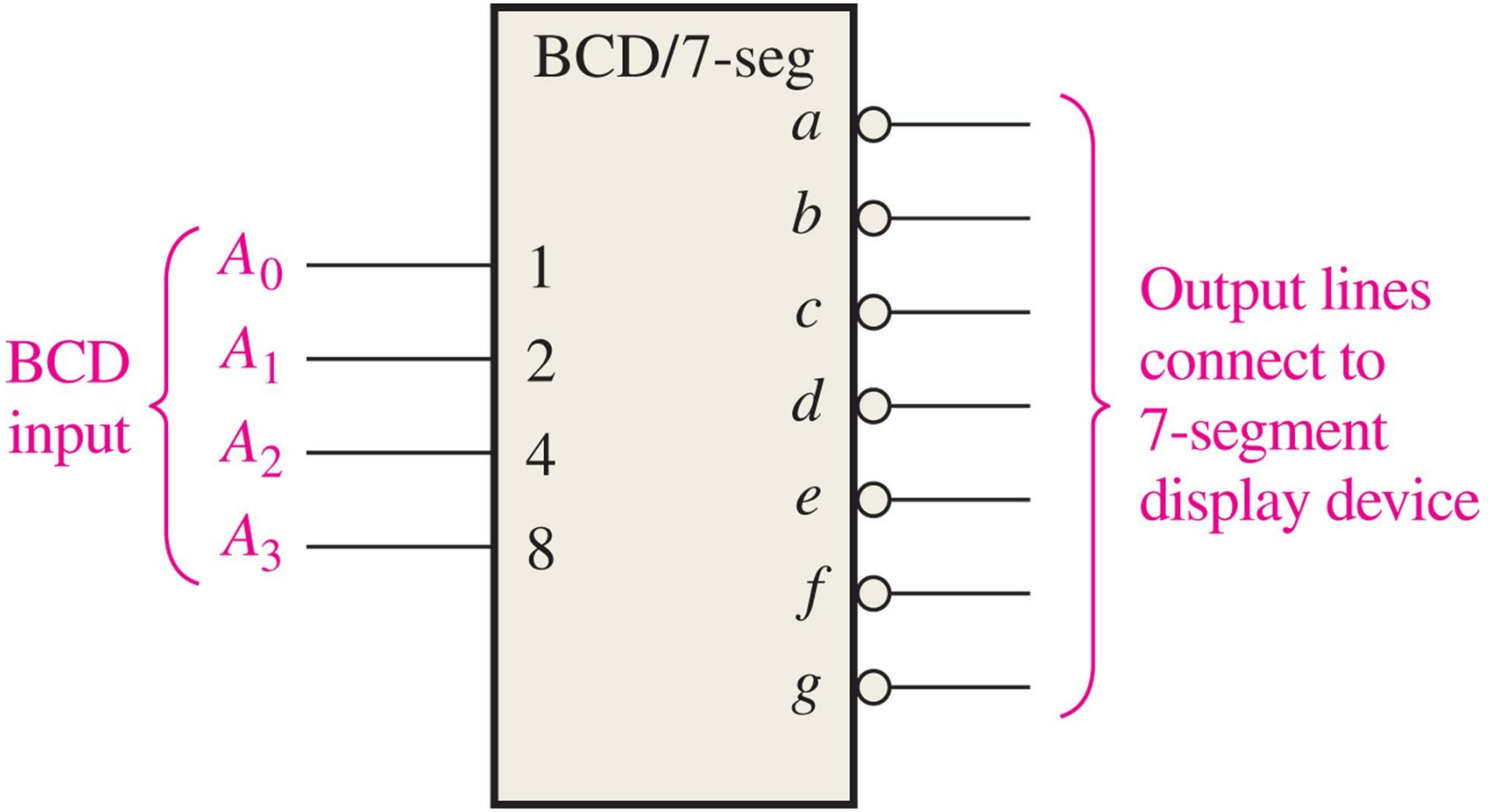


FIGURE 6-34 The 74HC47 BCD-to-7-segment decoder/driver.

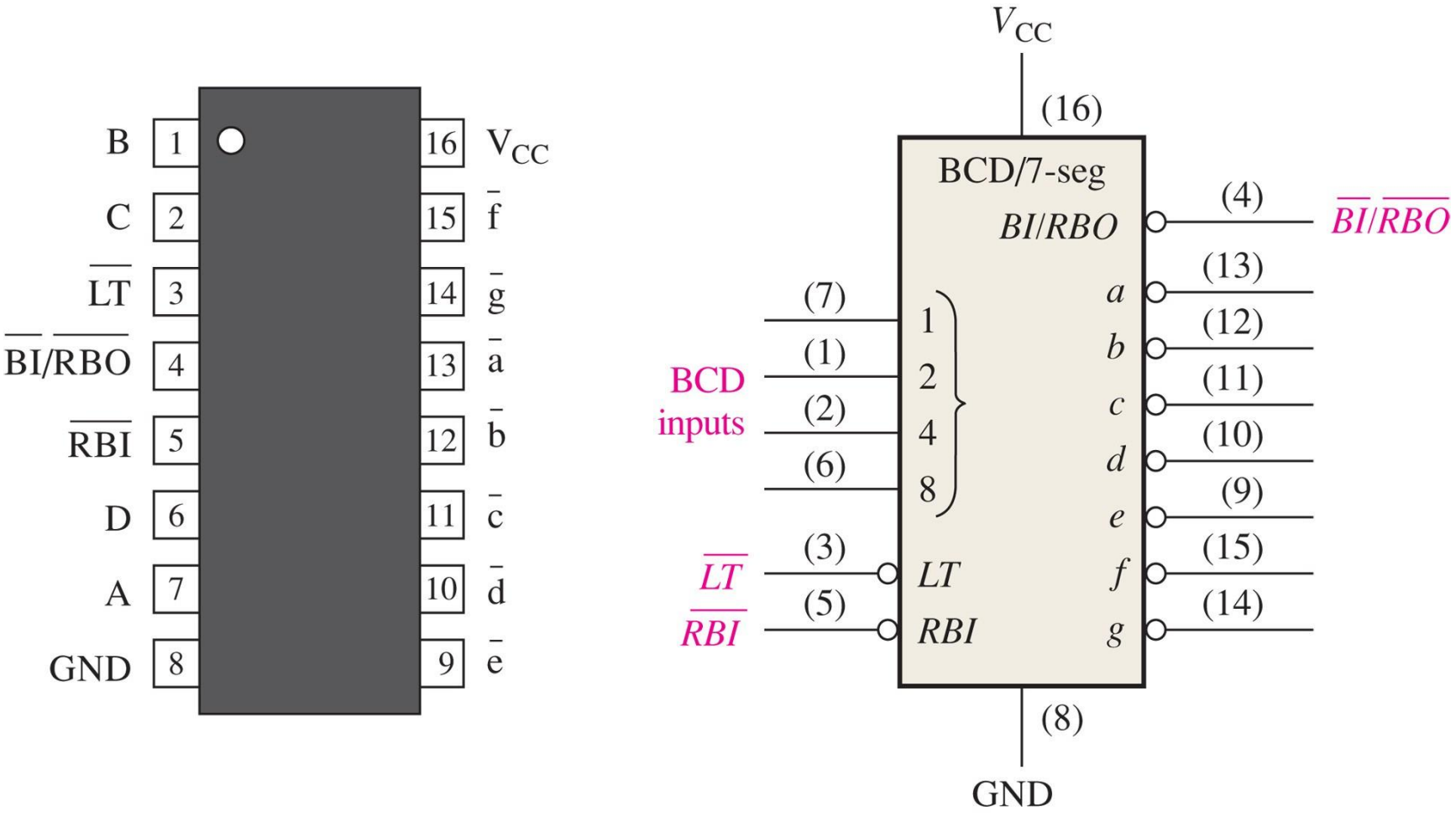
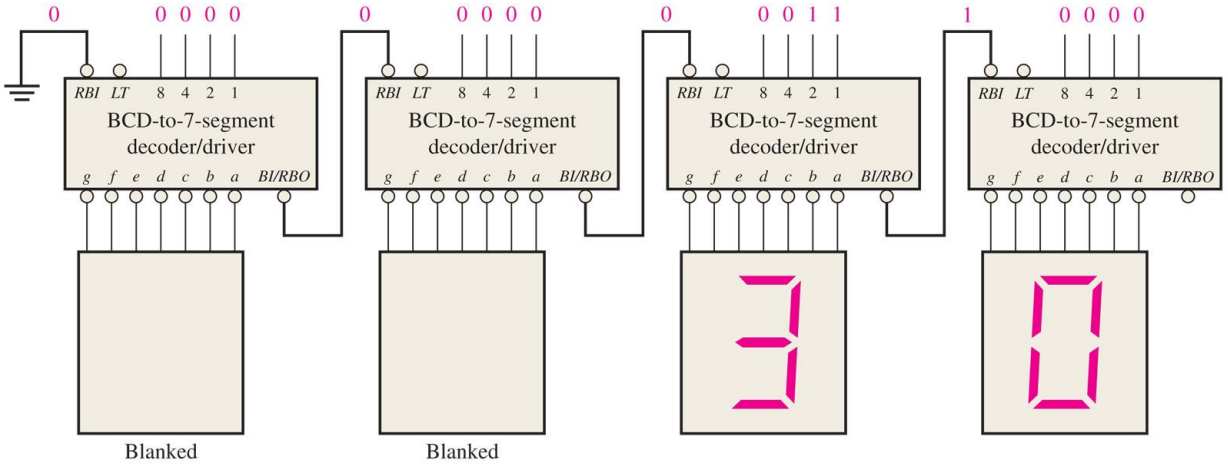
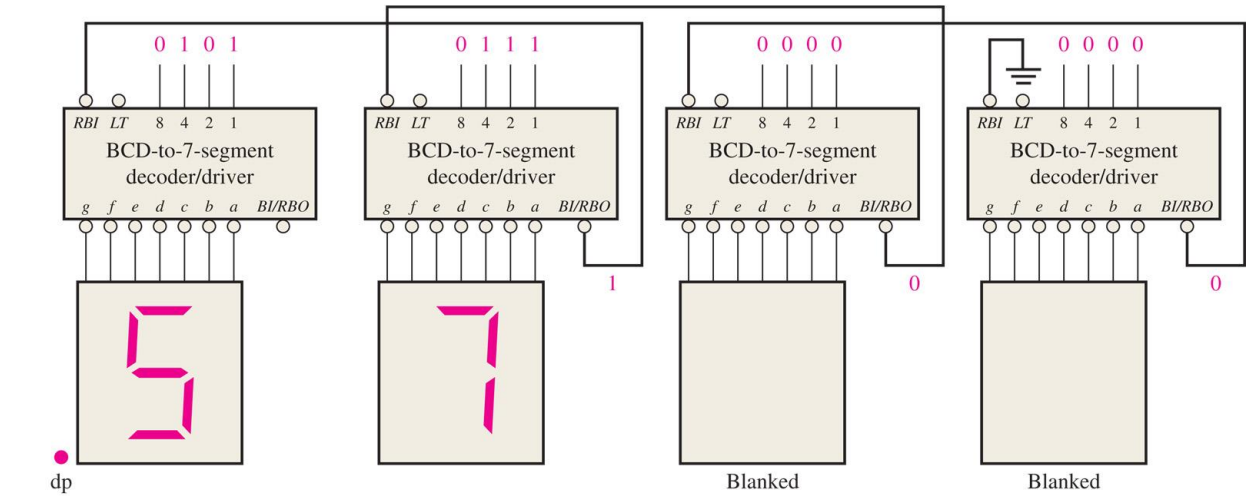


FIGURE 6-35 Examples of zero suppression using a BCD-to-7-segment decoder/driver.



(a) Illustration of leading zero suppression



(b) Illustration of trailing zero suppression

Zero Suppression for a 4-Digit Display

The logic diagram in Figure 6–35(a) illustrates leading zero suppression for a whole number. The highest-order digit position (left-most) is always blanked if a zero code is on its BCD inputs because the RBI of the most-significant decoder is made LOW by connecting it to ground. The RBO of each decoder is connected to the RBI of the next lowest-order decoder so that all zeros to the left of the first nonzero digit are blanked. For example, in part (a) of the figure the two highest-order digits are zeros and therefore are blanked. The remaining two digits, 3 and 0 are displayed.

Encoders

FIGURE 6-36 Logic symbol for a decimal-to-BCD encoder.

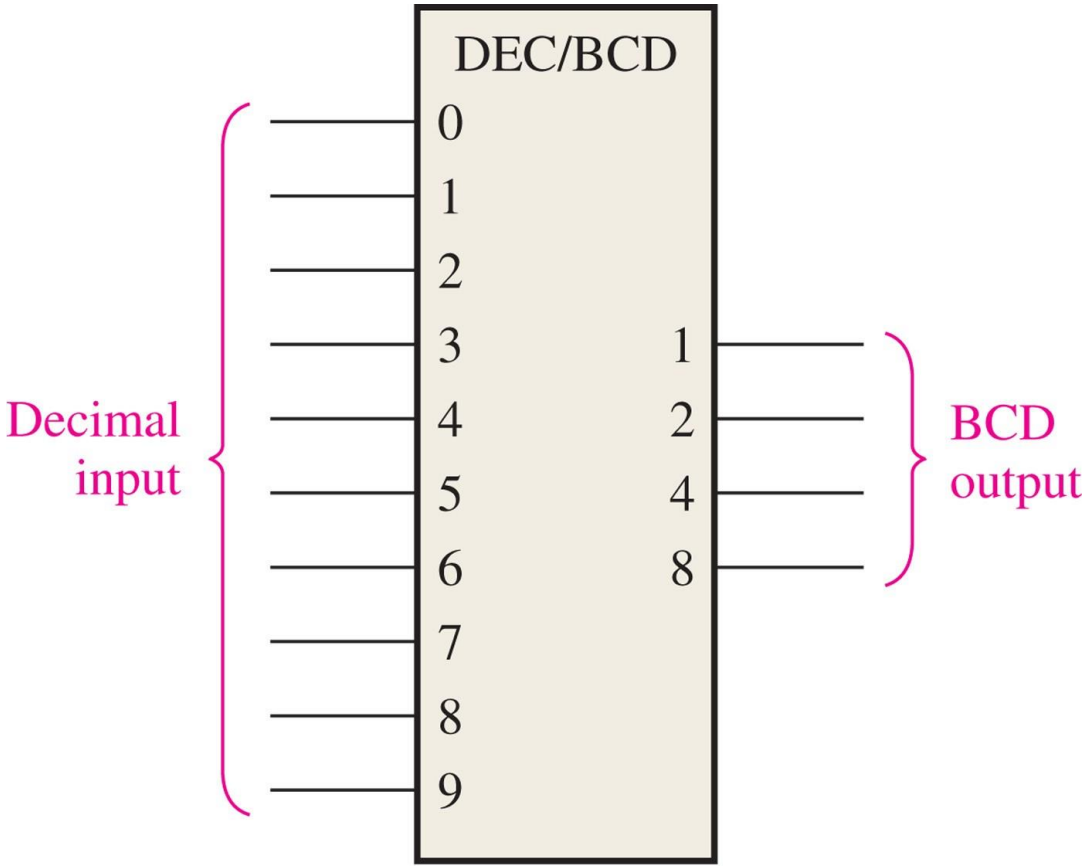
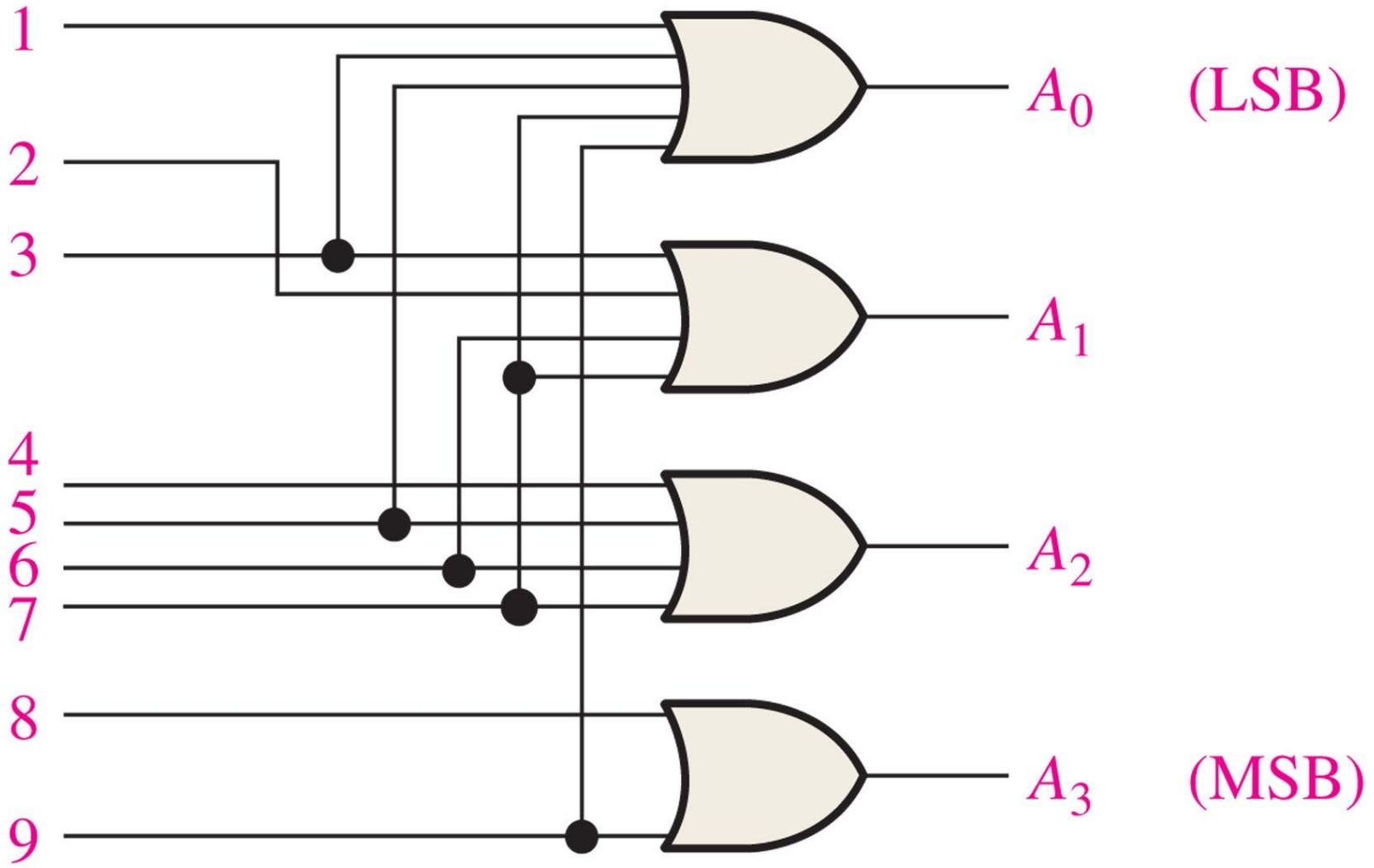


TABLE 6–6

Decimal Digit	BCD Code			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

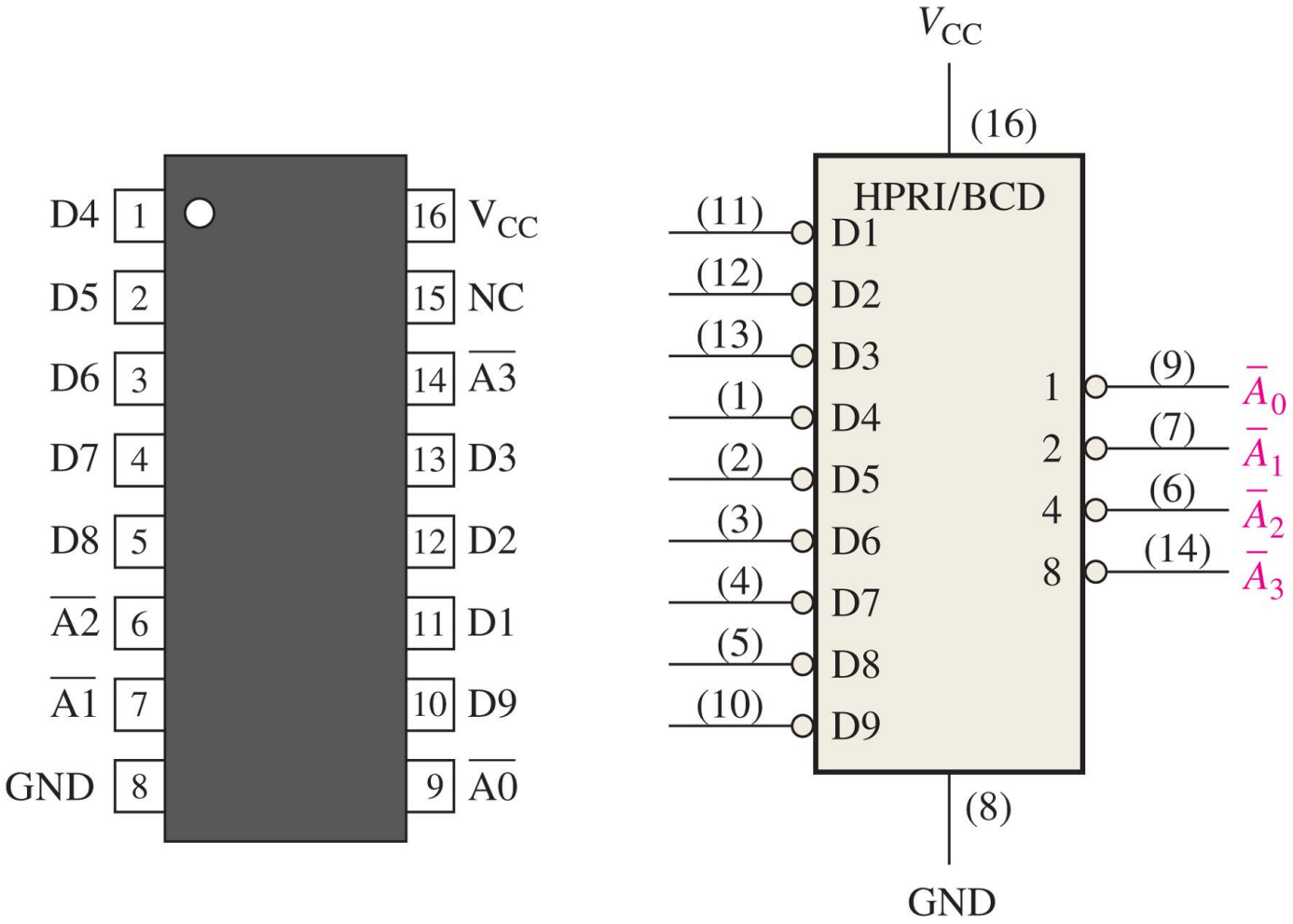
FIGURE 6-37 Basic logic diagram of a decimal-to-BCD encoder. A 0-digit input is not needed because the BCD outputs are all LOW when there are no HIGH inputs.



The Decimal-to-BCD Priority Encoder

This type of encoder performs the same basic encoding function as previously discussed. A priority encoder also offers additional flexibility in that it can be used in applications that require priority detection. The priority function means that the encoder will produce a BCD output corresponding to the highest-order decimal digit input that is active and will ignore any other lower-order active inputs. For instance, if the 6 and the 3 inputs are both active, the BCD output is 0110 (which represents decimal 6).

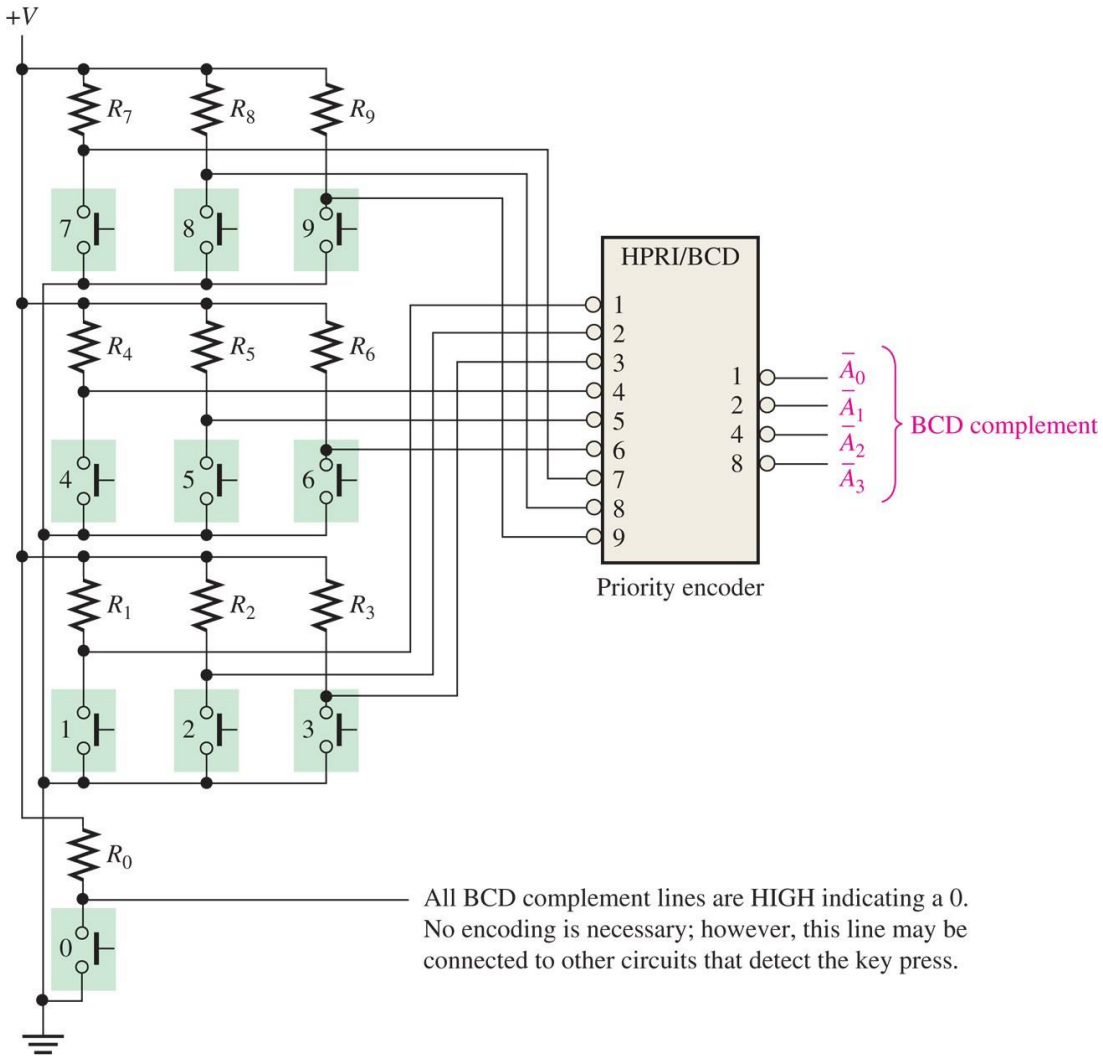
FIGURE 6-38 The 74HC147 decimal-to-BCD encoder (HPRI means highest value input has priority.



(a) Pin diagram

(b) Logic diagram

FIGURE 6-39 A simplified keyboard encoder.



Code Converters

BCD-to-Binary Conversion

One method of BCD-to-binary code conversion uses adder circuits. The basic conversion process is as follows: 1. The value, or weight, of each bit in the BCD number is represented by a binary number. 2. All of the binary representations of the weights of bits that are 1s in the BCD number are added. 3. The result of this addition is the binary equivalent of the BCD number. A more concise statement of this operation is The binary numbers representing the weights of the BCD bits are summed to produce the total binary number

Let's examine an 8-bit BCD code (one that represents a 2-digit decimal number) to understand the relationship between BCD and binary. For instance, you already know that the decimal number 87 can be expressed in BCD as

$$\underbrace{1000}_{8} \quad \underbrace{0111}_{7}$$

The left-most 4-bit group represents 80, and the right-most 4-bit group represents 7. That is, the left-most group has a weight of 10, and the right-most group has a weight of 1. Within each group, the binary weight of each bit is as follows:

	Tens Digit				**Units Digit**			
Weight:	80	40	20	10	8	4	2	1
Bit designation:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

The binary equivalent of each BCD bit is a binary number representing the weight of that bit within the total BCD number. This representation is given in Table 6–7.

TABLE 6–7

Binary representations of BCD bit weights.

BCD Bit	BCD Weight	Binary Representation (MSB) 64	32	16	8	4	2	(LSB) 1
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

Convert the BCD numbers 00100111 (decimal 27) and 10011000 (decimal 98) to binary.

Solution

Write the binary representations of the weights of all 1s appearing in the numbers, and then add them together.

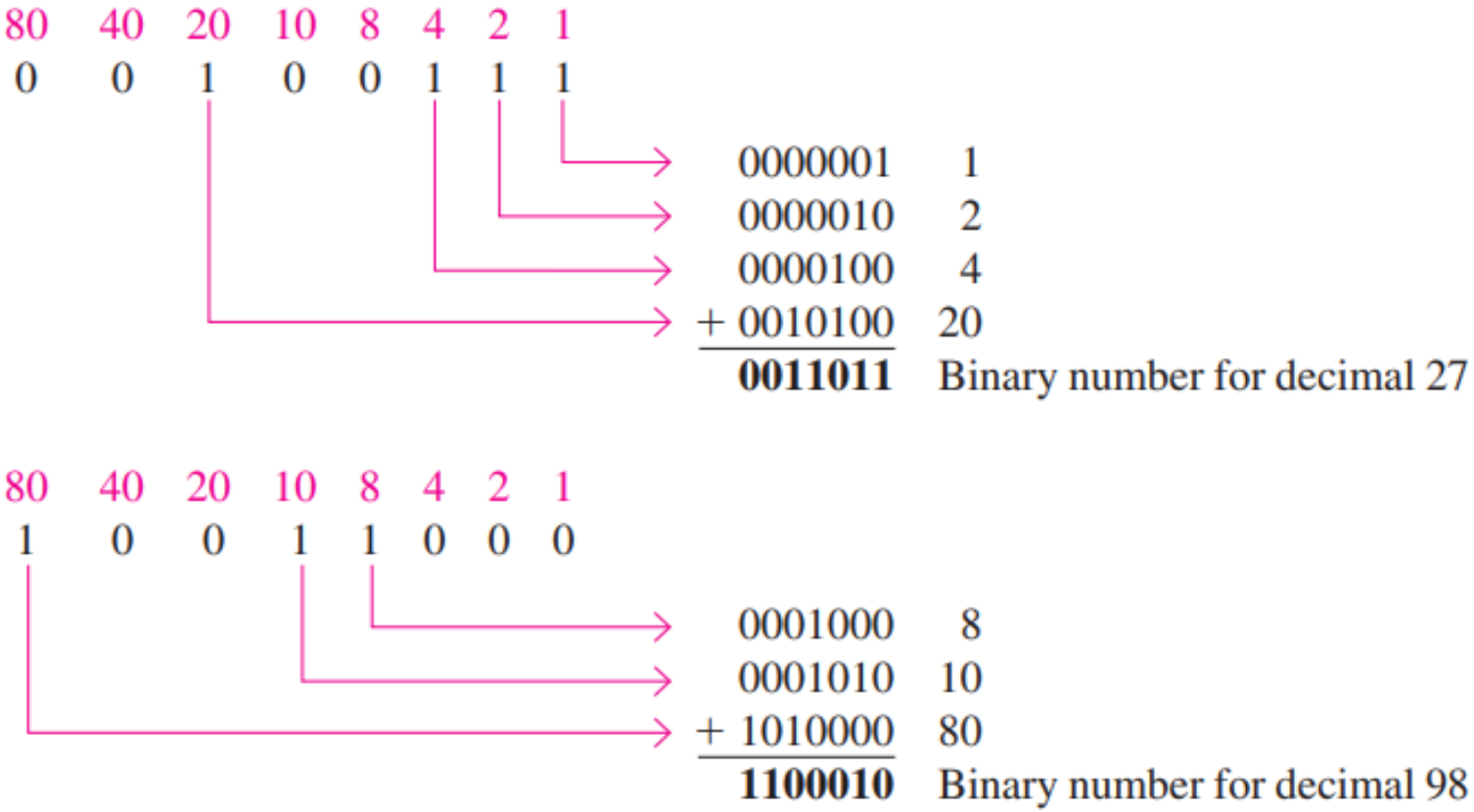


FIGURE 6-40 Four-bit binary-to-Gray conversion logic.

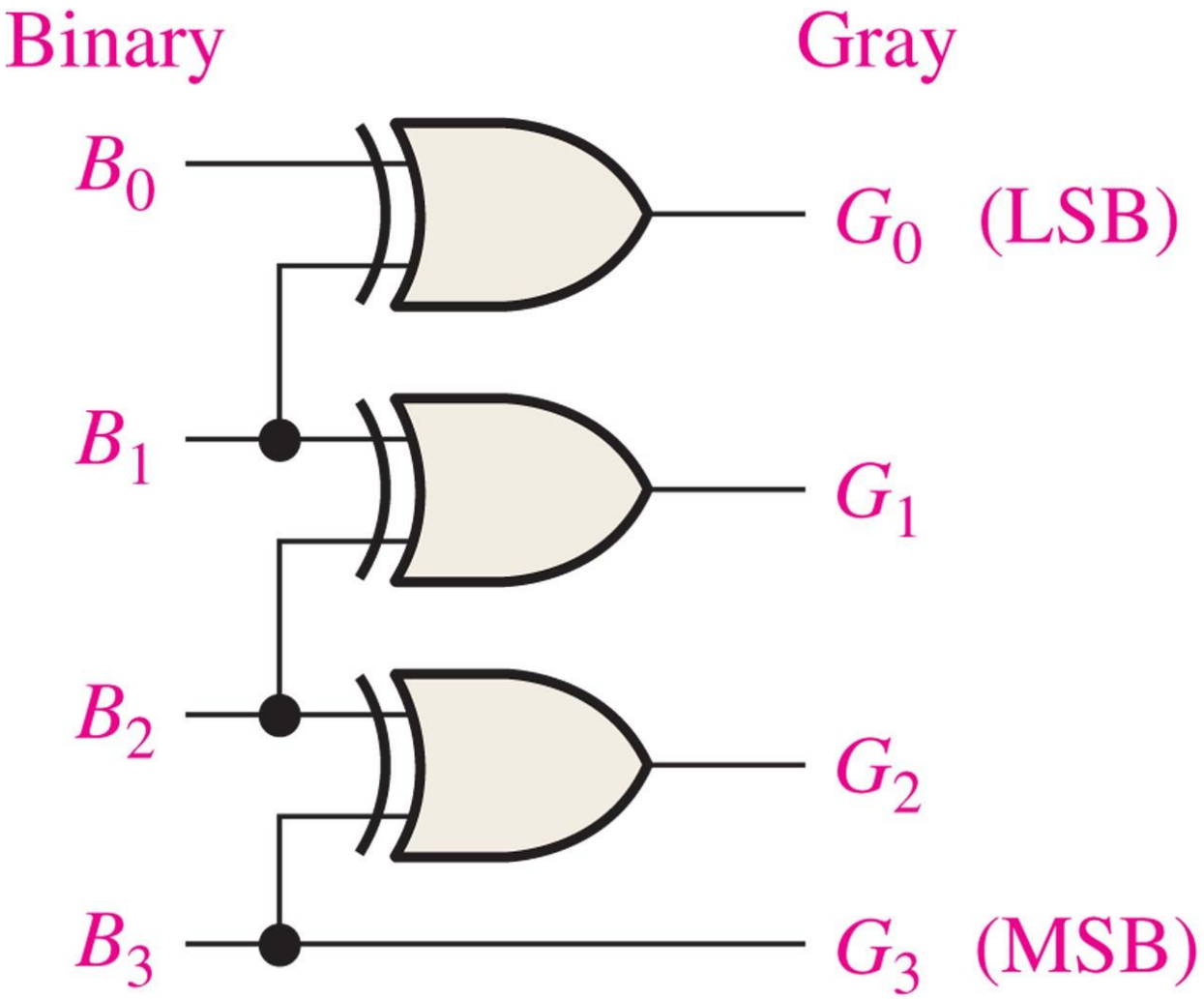
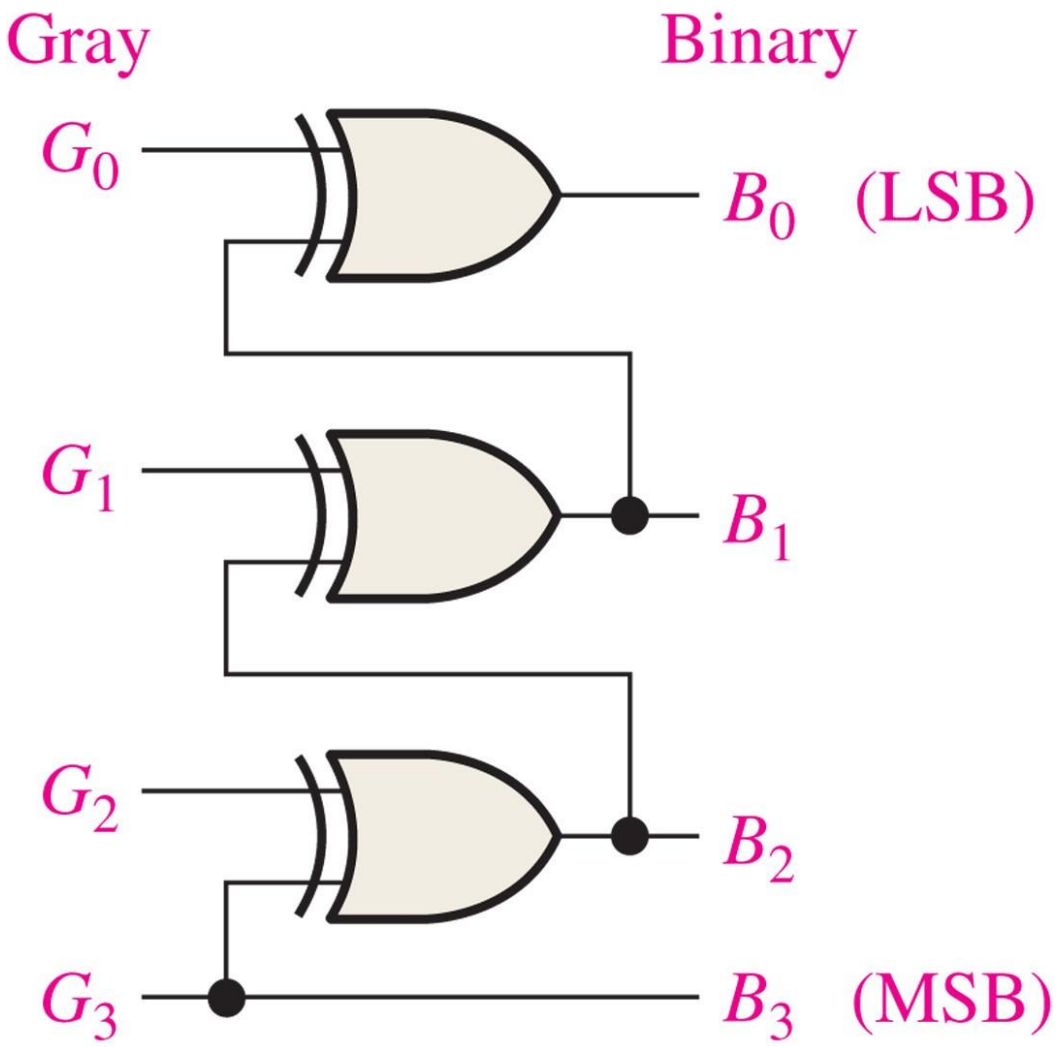


FIGURE 6-41 Four-bit Gray-to-binary conversion logic.



(a) Convert the binary number 0101 to Gray code with exclusive-OR gates.

(b) Convert the Gray code 1011 to binary with exclusive-OR gates.

Solution

(a) 0101_2 is 0111 Gray. See Figure 6–42(a).

(b) 1011 Gray is 1101_2. See Figure 6–42(b).

FIGURE 6-42

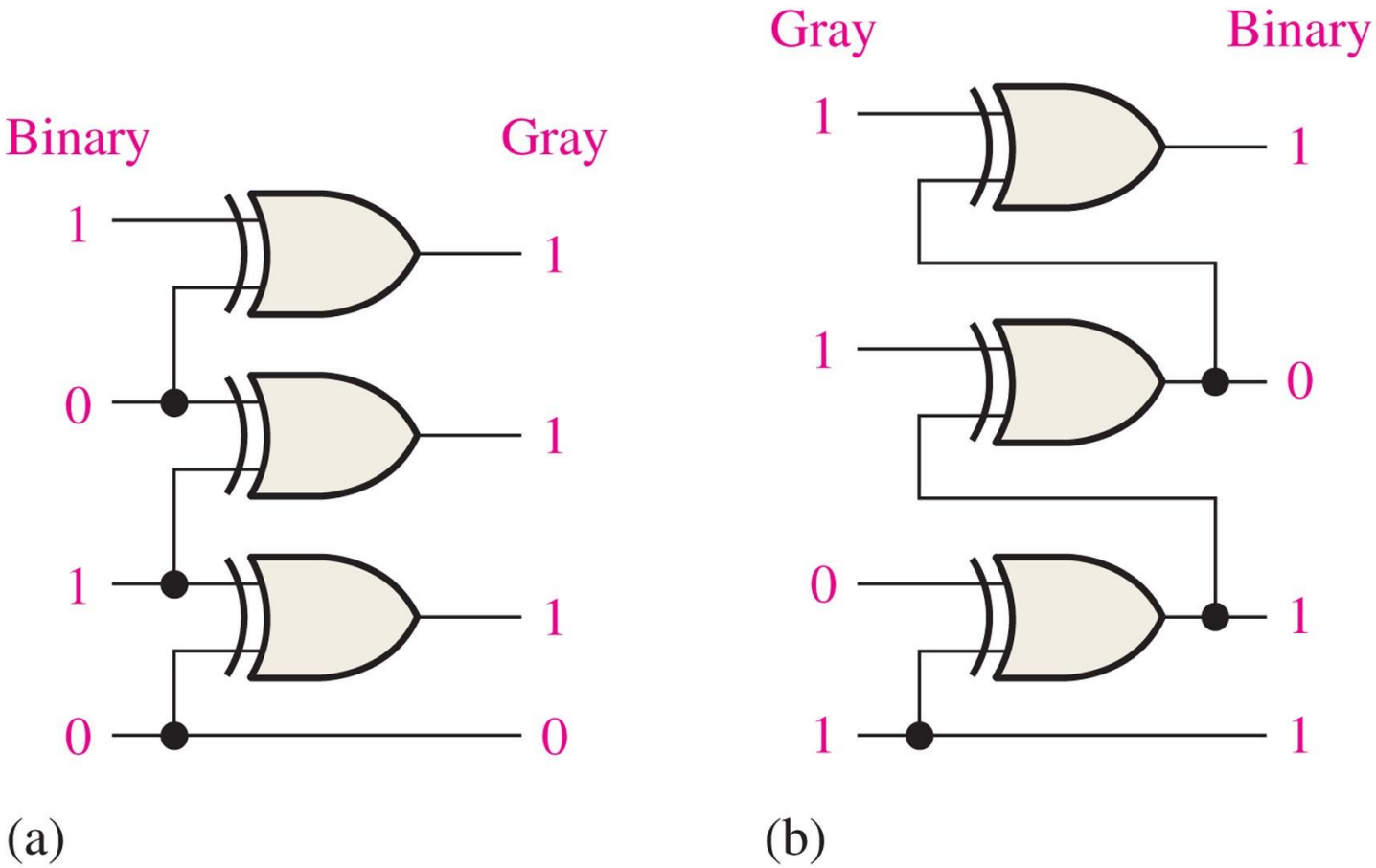


FIGURE 6-43 Logic symbol for a 1-of-4 data selector/multiplexer.

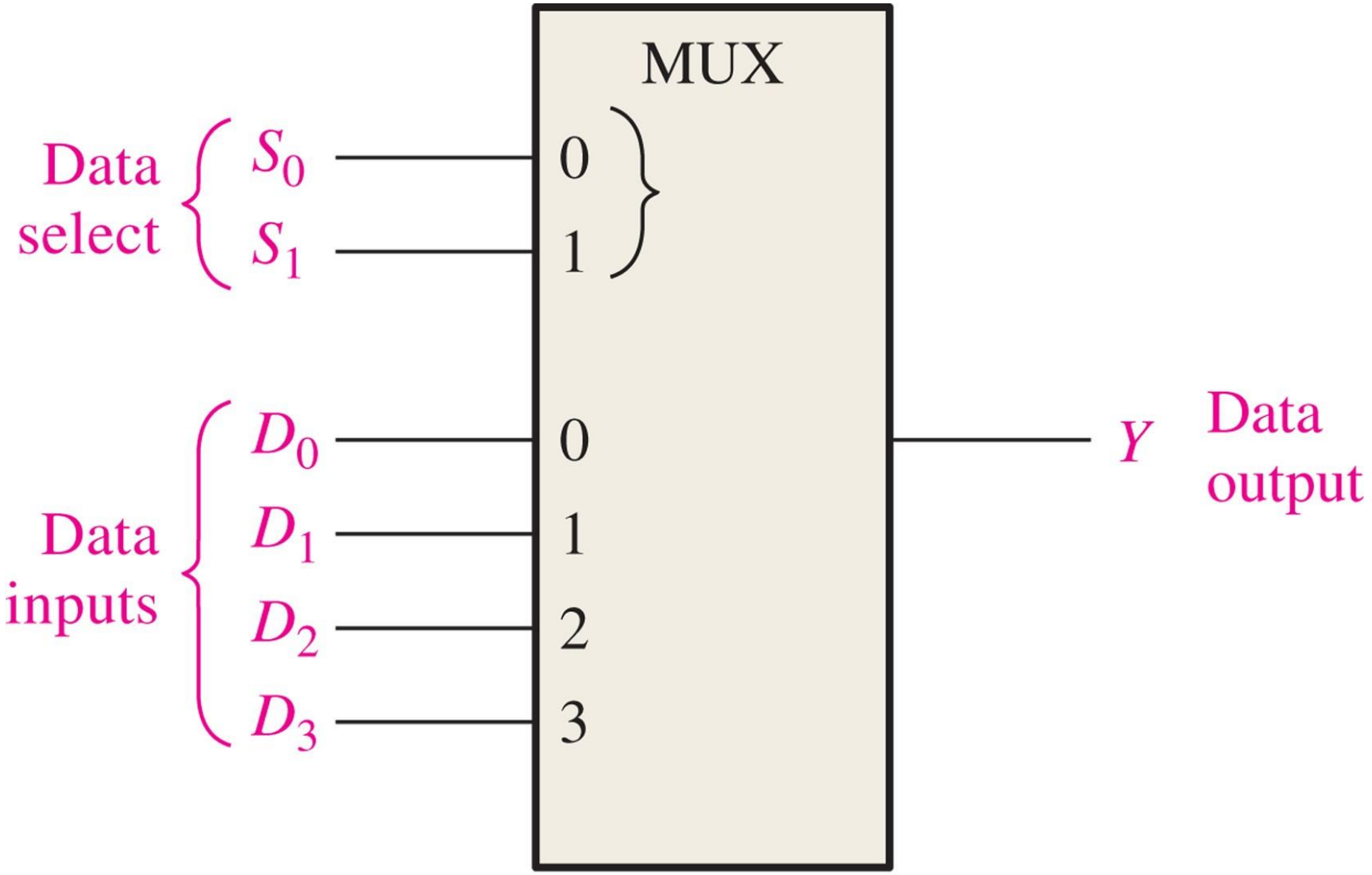


TABLE 6–8

Data selection for a 1-of-4-multiplexer.

Data-Select Inputs		
S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

FIGURE 6-44 Logic diagram for a 4-input multiplexer.

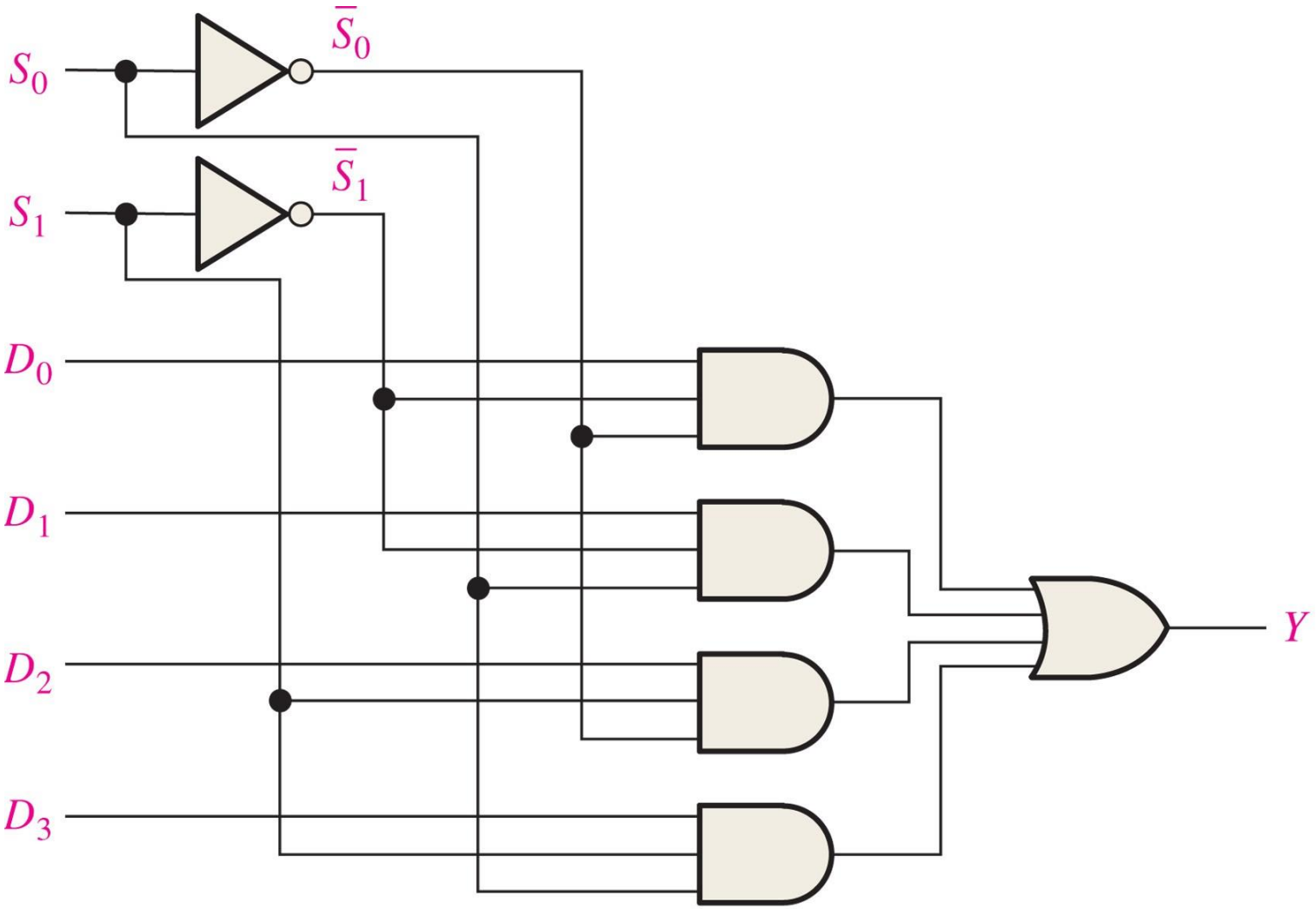


FIGURE 6-45

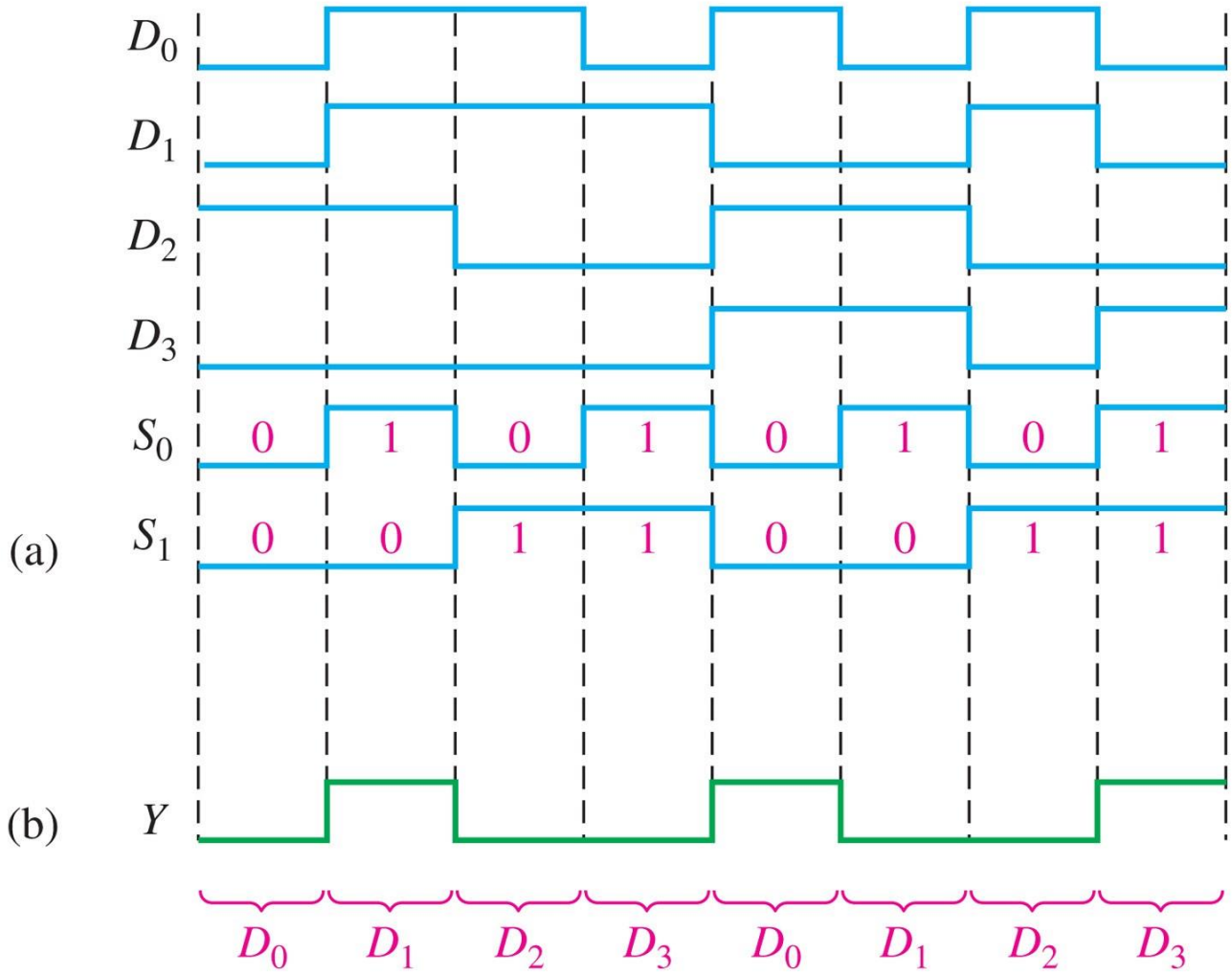
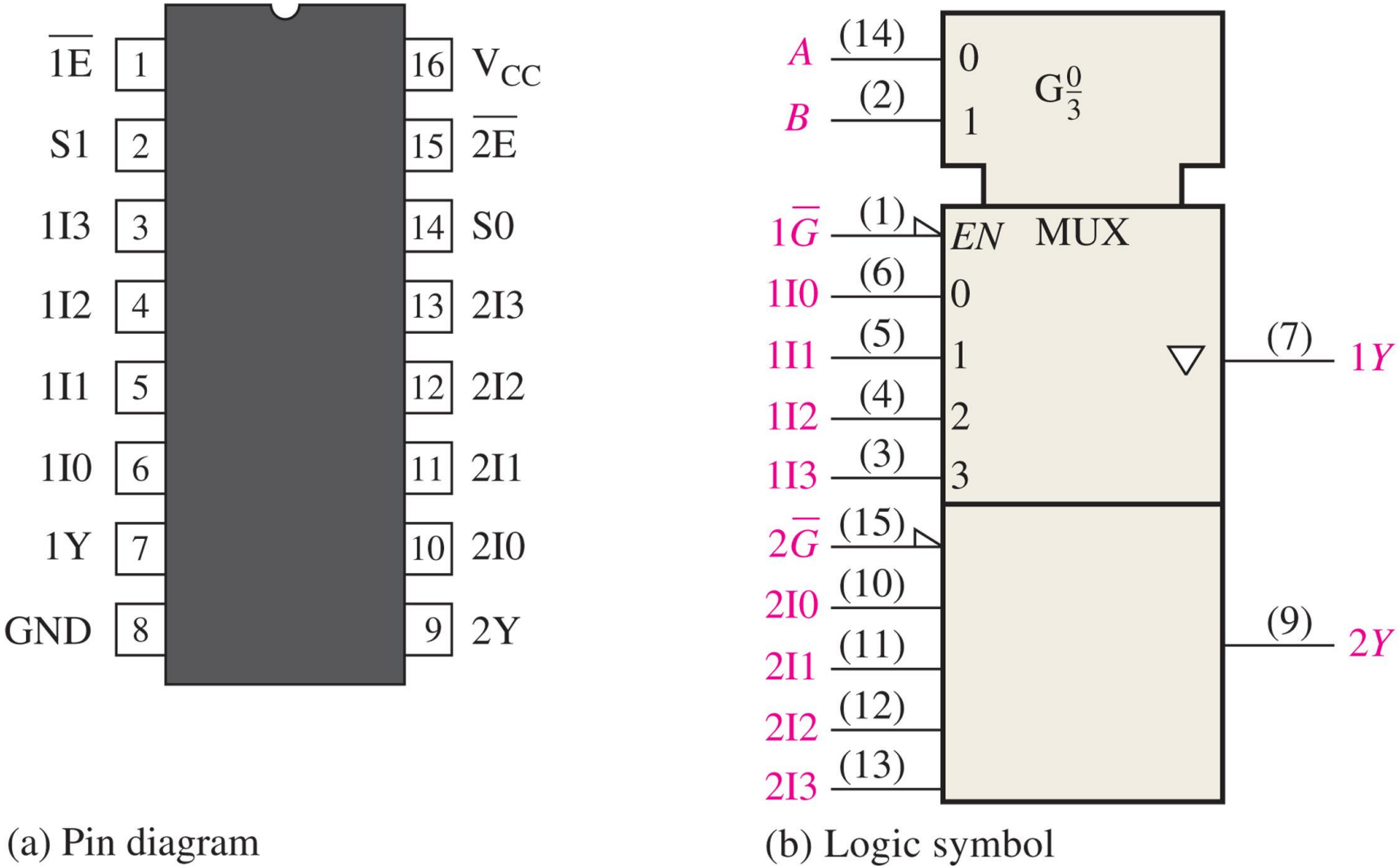


FIGURE 6-46 The 74HC153 dual four-input data selector/multiplexer.



(a) Pin diagram

(b) Logic symbol

FIGURE 6-47 The 74HC151 eight-input data selector/multiplexer.

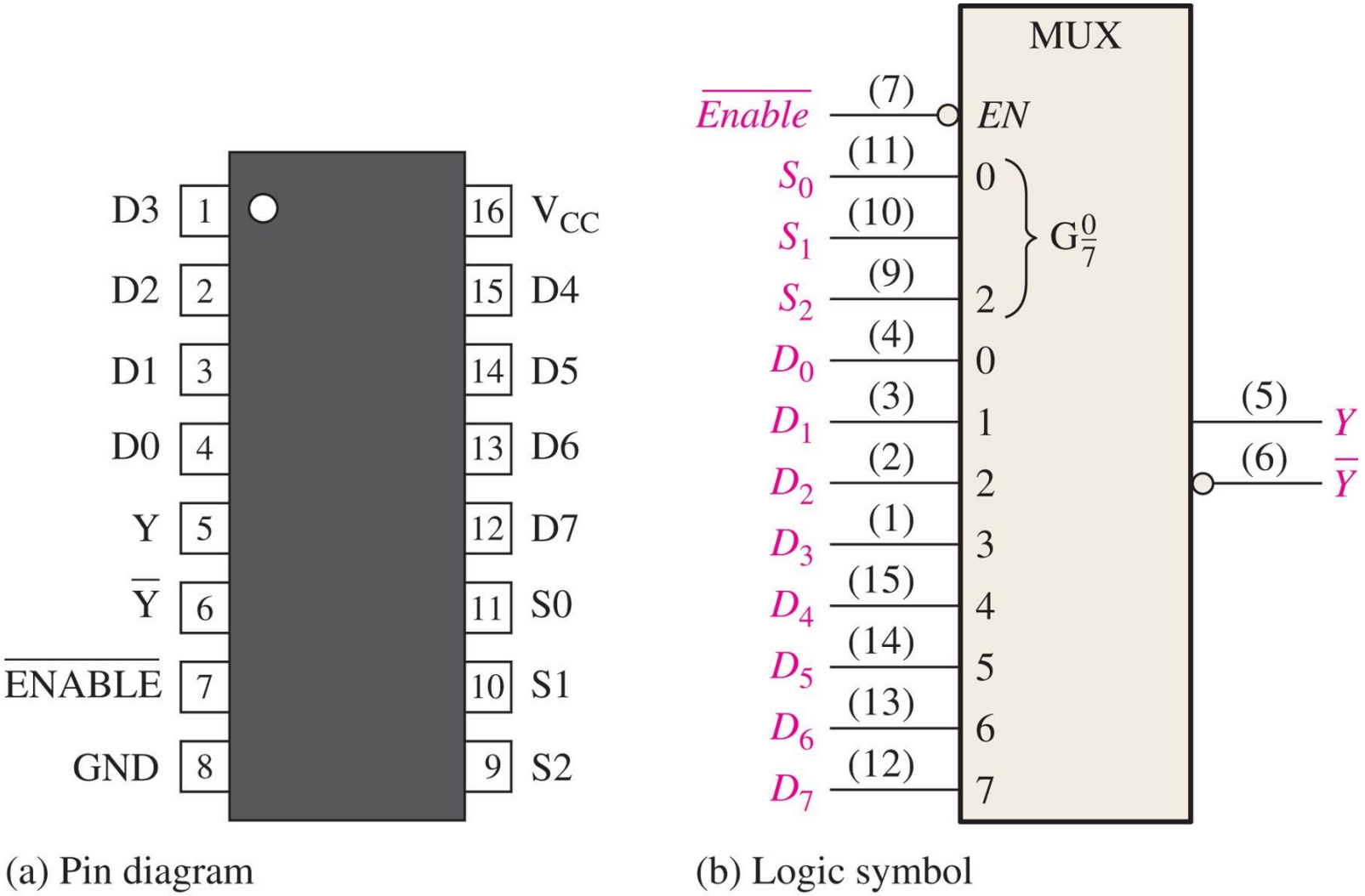
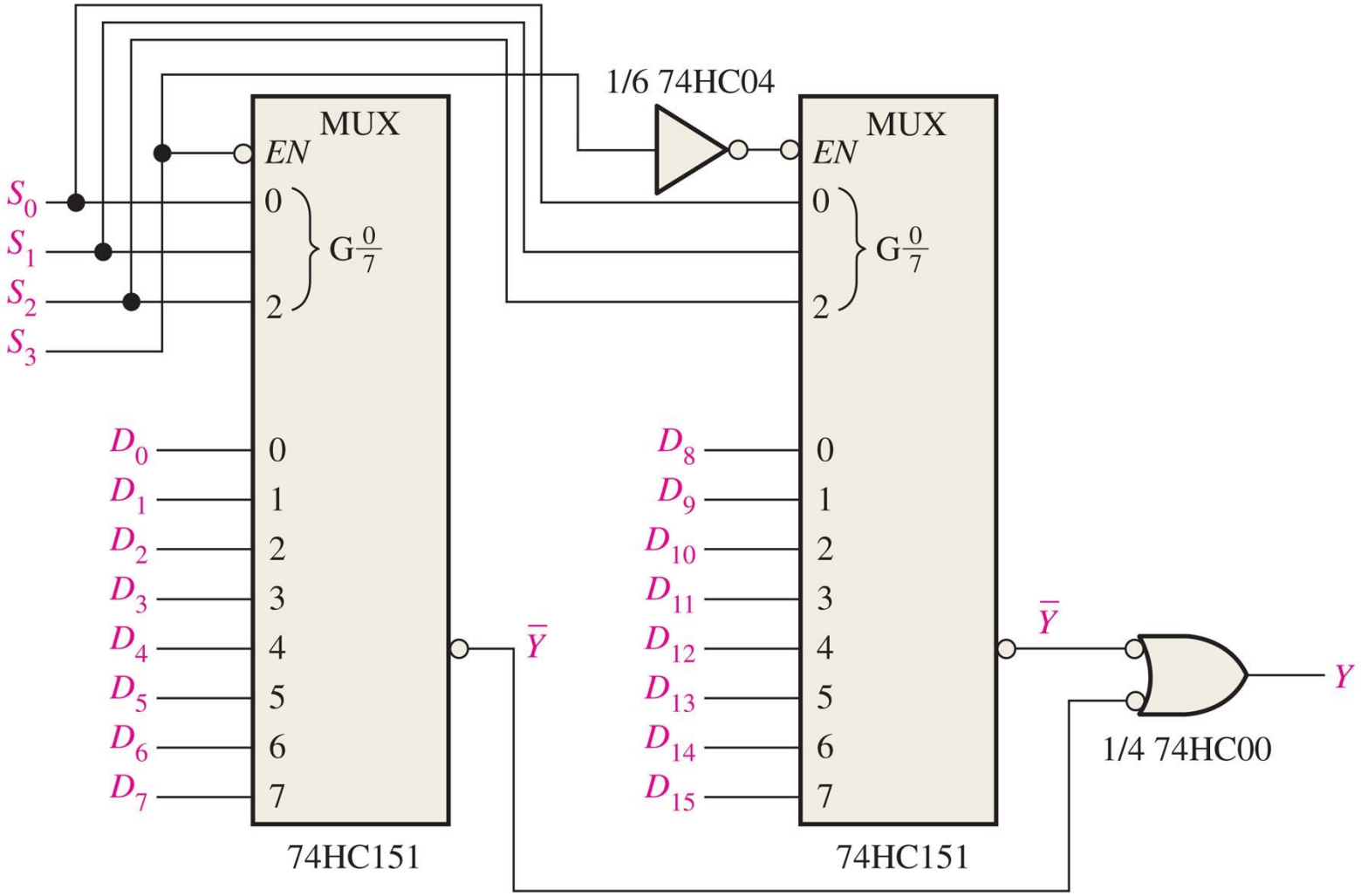


FIGURE 6-48 A 16-input multiplexer.



A 7-Segment Display Multiplexer

Figure 6–49 shows a simplified method of multiplexing BCD numbers to a 7-segment display. In this example, 2-digit numbers are displayed on the 7-segment readout by the use of a single BCD-to-7-segment decoder. This basic method of display multiplexing can be extended to displays with any number of digits. The 74HC157 is a quad 2-input multiplexer.

The basic operation is as follows. Two BCD digits ($A_3A_2A_1A_0$ and $B_3B_2B_1B_0$) are applied to the multiplexer inputs. A square wave is applied to the data-select line, and when it is LOW, the *A* bits ($A_3A_2A_1A_0$) are passed through to the inputs of the 74HC47 BCD-to-7-segment decoder. The LOW on the data-select also puts a LOW on the A_1 input of the 74HC139 2-line-to-4-line decoder, thus activating its 0 output and enabling the *A*-digit display by effectively connecting its common terminal to ground. The *A* digit is now *on* and the *B* digit is *off*.

FIGURE 6-49 Simplified 7-segment display multiplexing logic.

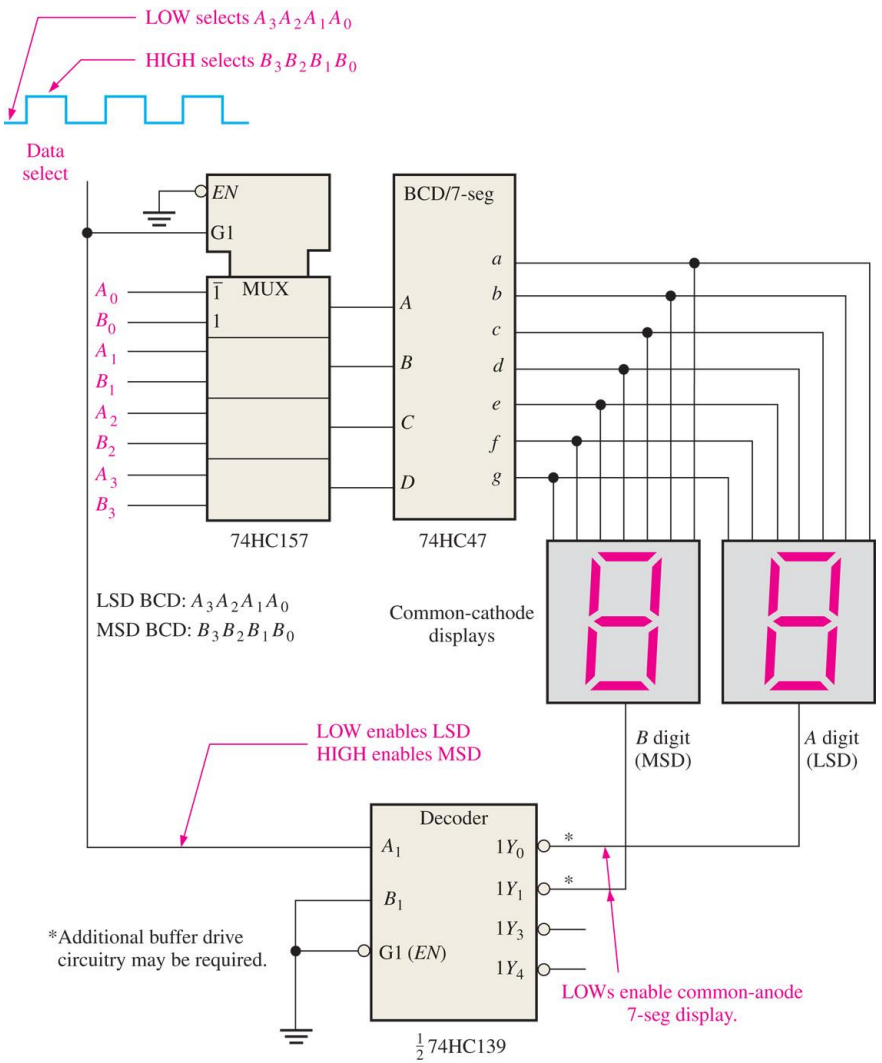


TABLE 6–9

Inputs			Output
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

FIGURE 6-50 Data selector/multiplexer connected as a 3-variable logic function generator.

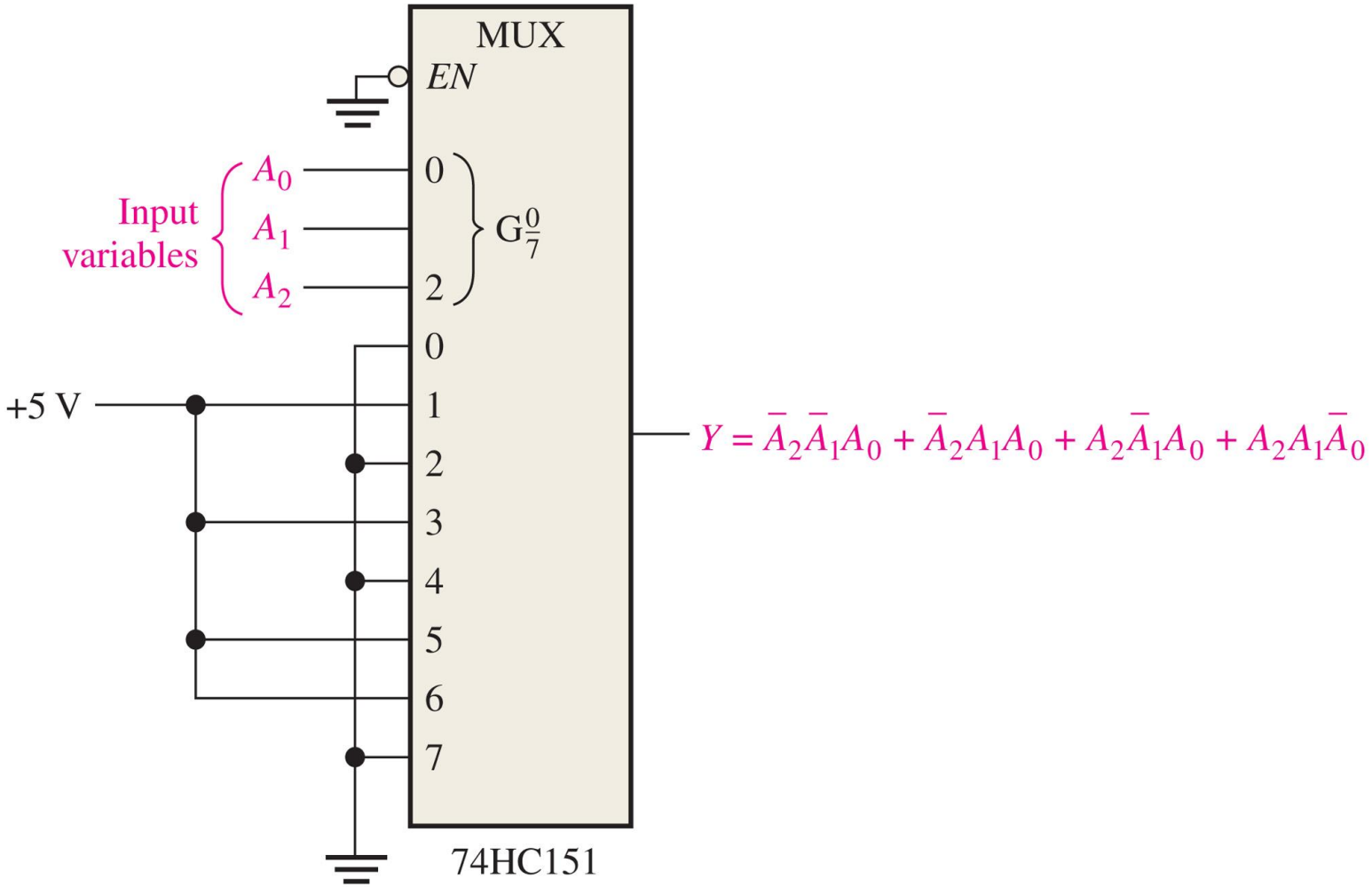
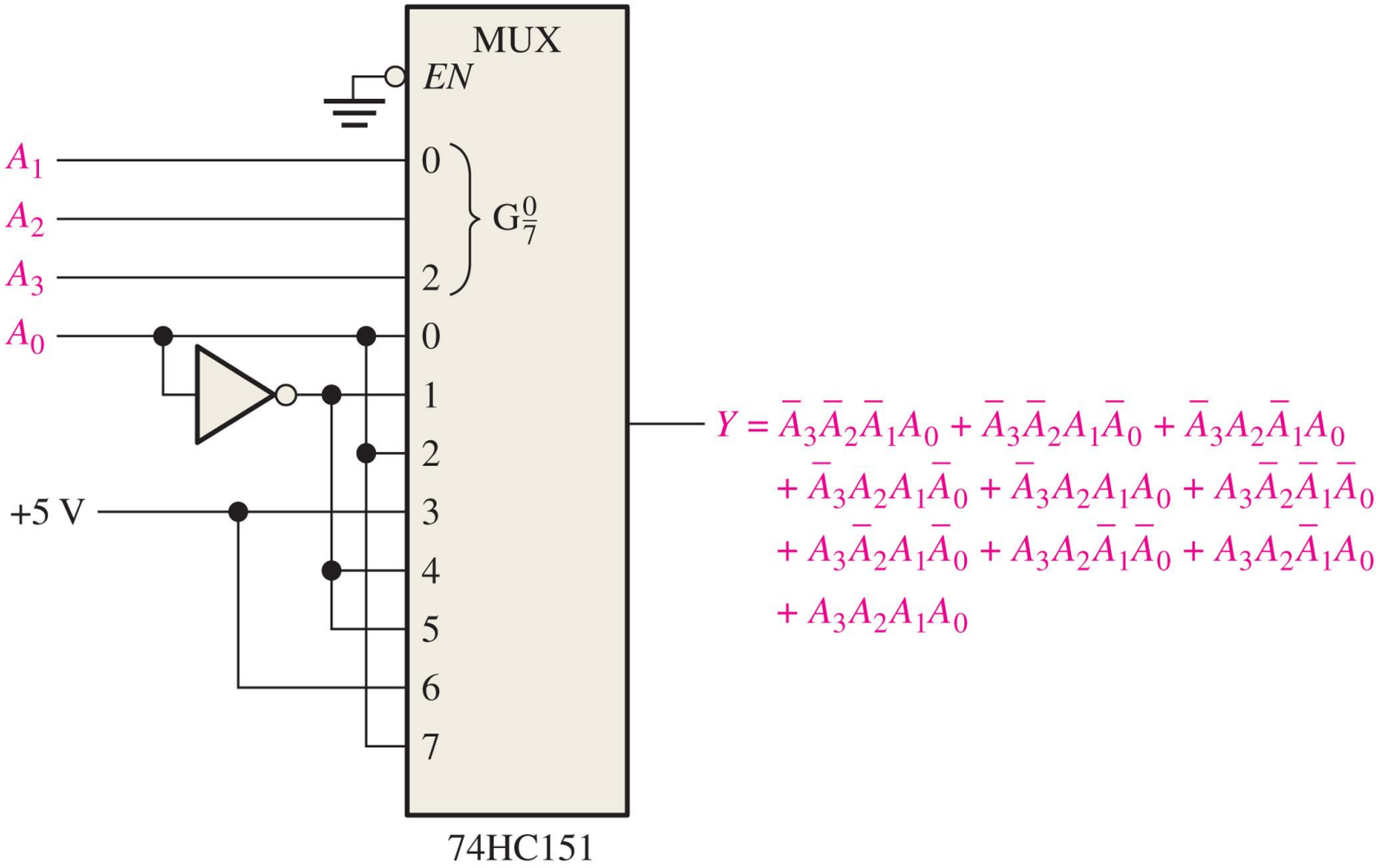


TABLE 6–10

Decimal Digit	Inputs				Output
	A_3	A_2	A_1	A_0	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

FIGURE 6-51 Data selector/multiplexer connected as a 4-variable logic function generator.



Demultiplexers

FIGURE 6-52 A 1-line-to-4-line demultiplexer.

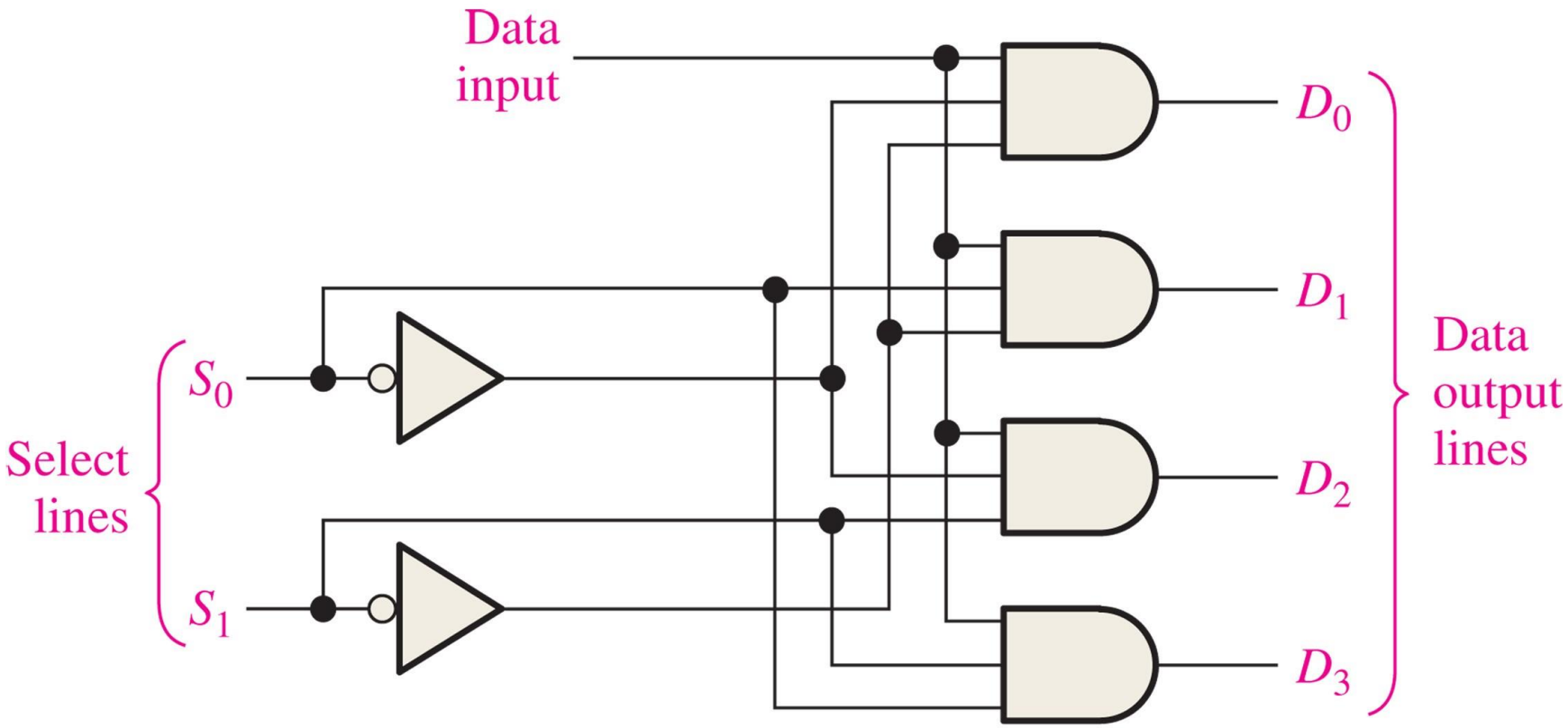


FIGURE 6-53

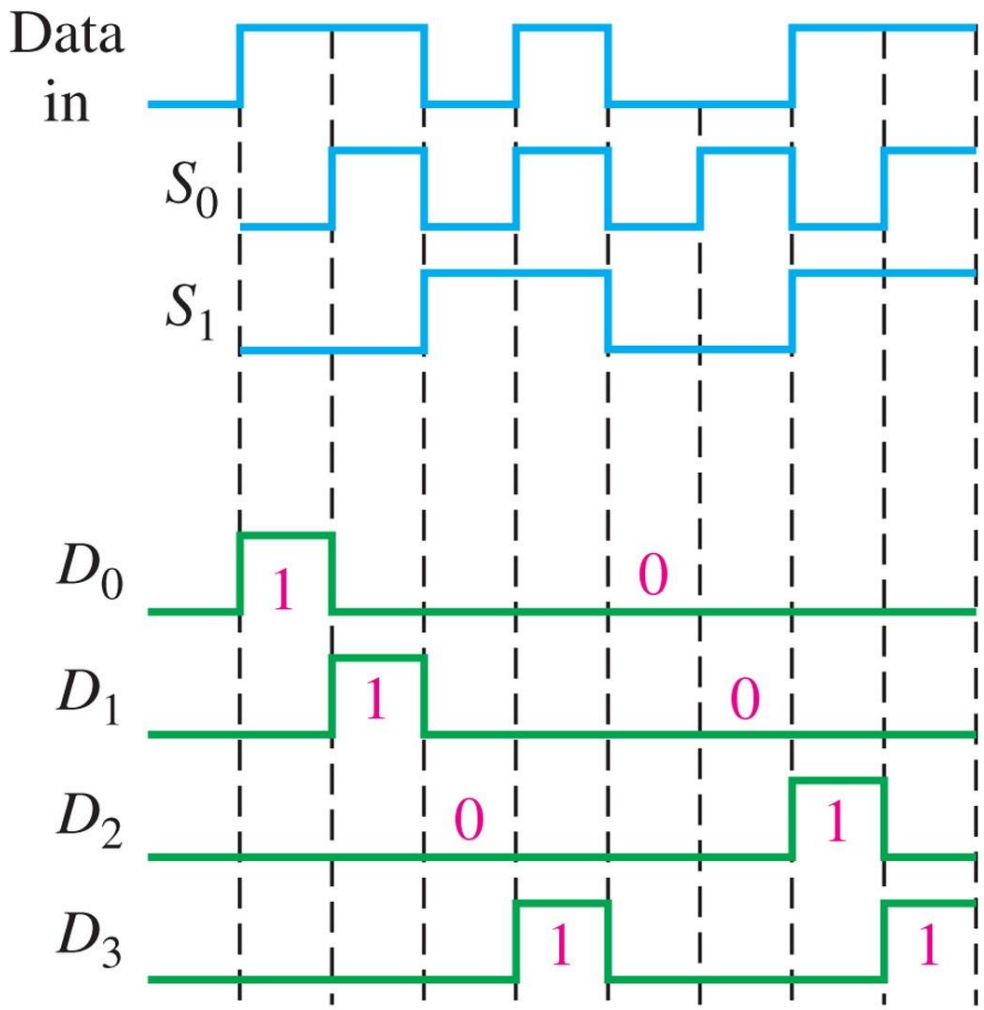


FIGURE 6-54 The decoder used as a demultiplexer.

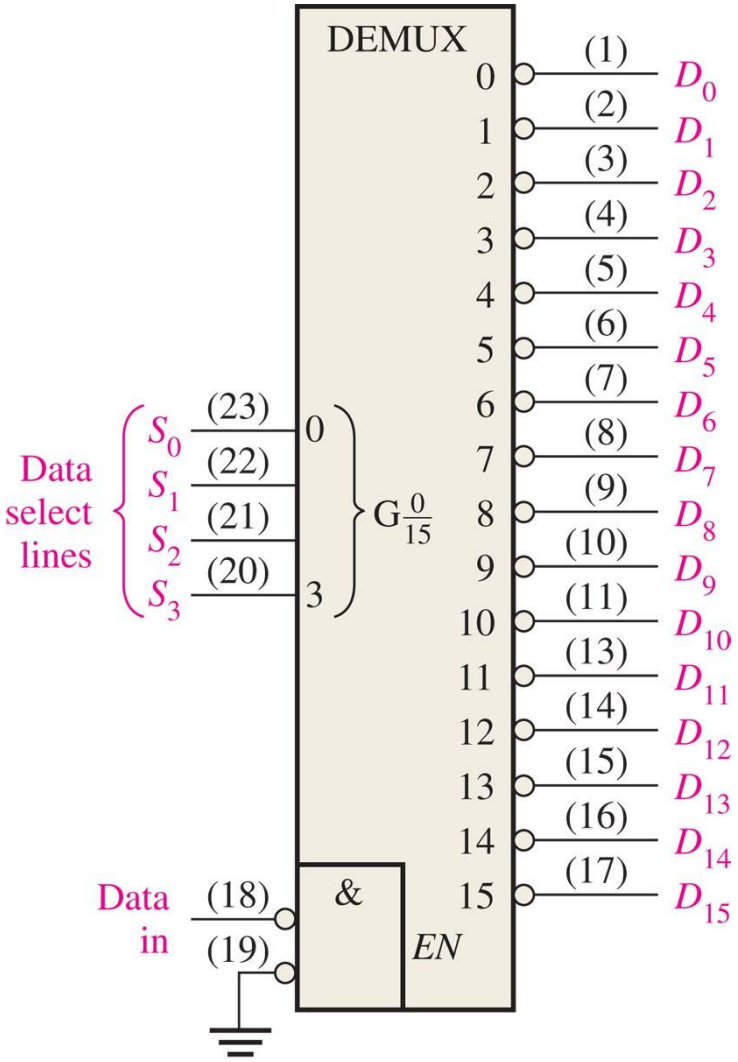
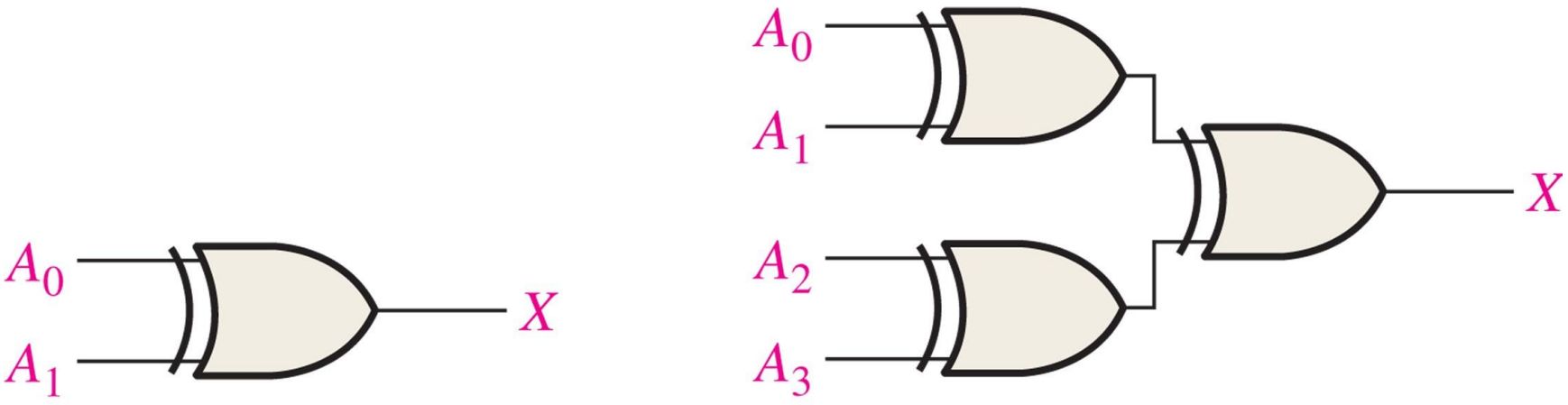


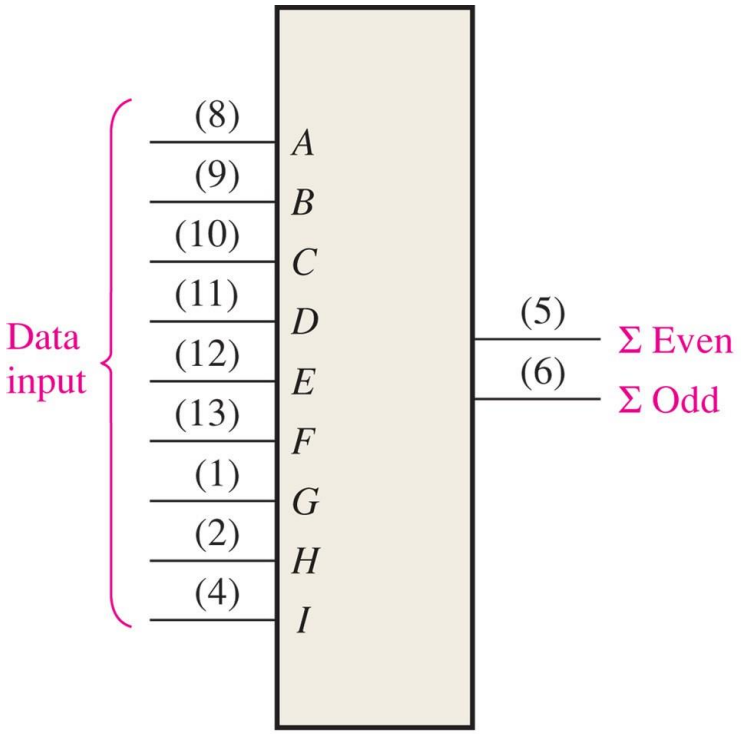
FIGURE 6-55



(a) Summing of two bits

(b) Summing of four bits

FIGURE 6-56 The 74HC280 9-bit parity generator/checker.



(a) Traditional logic symbol

Number of Inputs *A–I* that Are High	Outputs Σ Even	Σ Odd
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

(b) Function table

The control logic is developed for a traffic signal at the intersection of a busy main street and a lightly used side street. The system requirements are established, and a general block diagram is developed. Also, a state diagram is introduced to define the sequence of operation. The combinational logic unit of the controller is developed in this chapter, and the remaining units are developed in Chapter 7

Timing Requirements

The green light for the main street will stay on for a minimum of 25 s or as long as there is no vehicle on the side street.

The green light for the side street will stay on until there is no vehicle on the side street up to a maximum of 25 s.

The yellow caution light will stay on for 4 s between changes from green to red on both the main street and the side street.

Defining the Variables

Vs A vehicle is present on the side street.

T_L The 25 s timer (long timer) is on.

Ts The 4 s timer (short timer) is on.

FIGURE 6-63 State diagram for the traffic signal control.

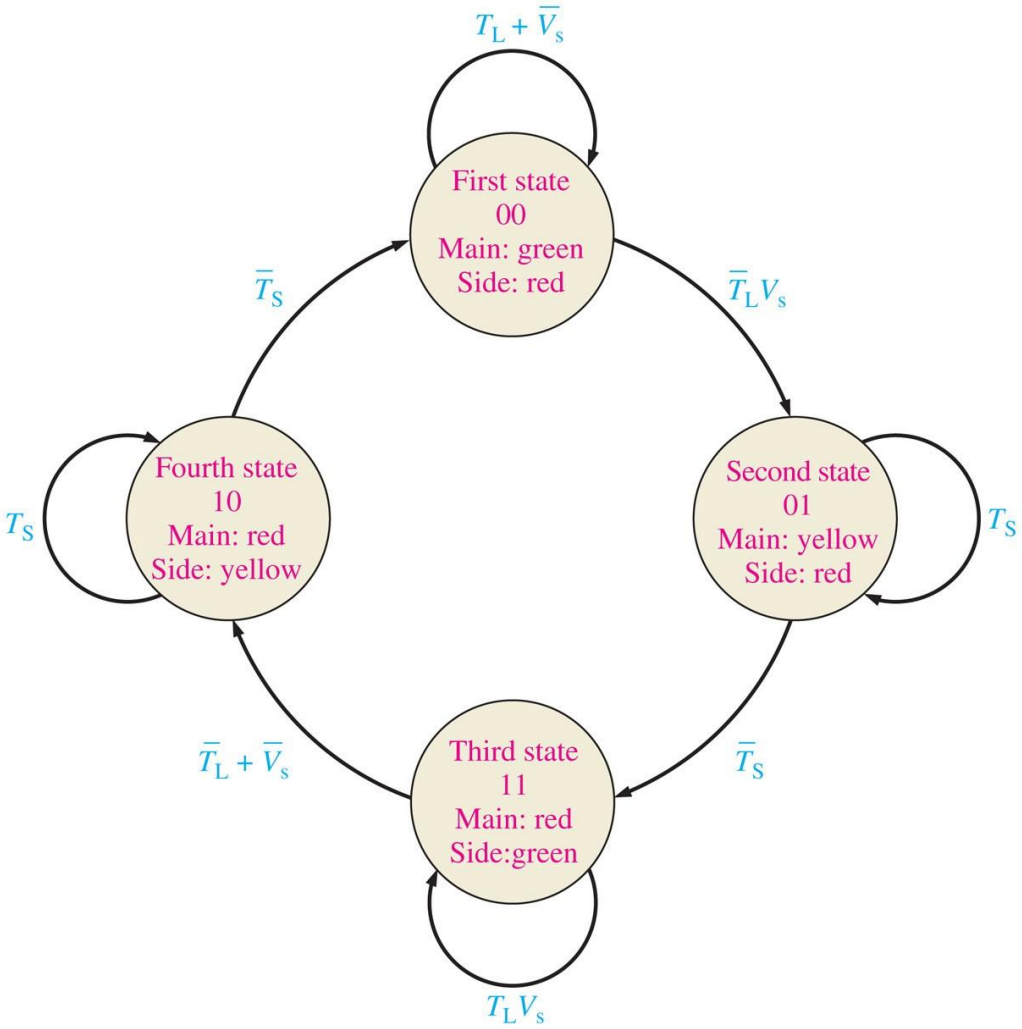


FIGURE 6-64 Block diagram of the traffic signal controller.

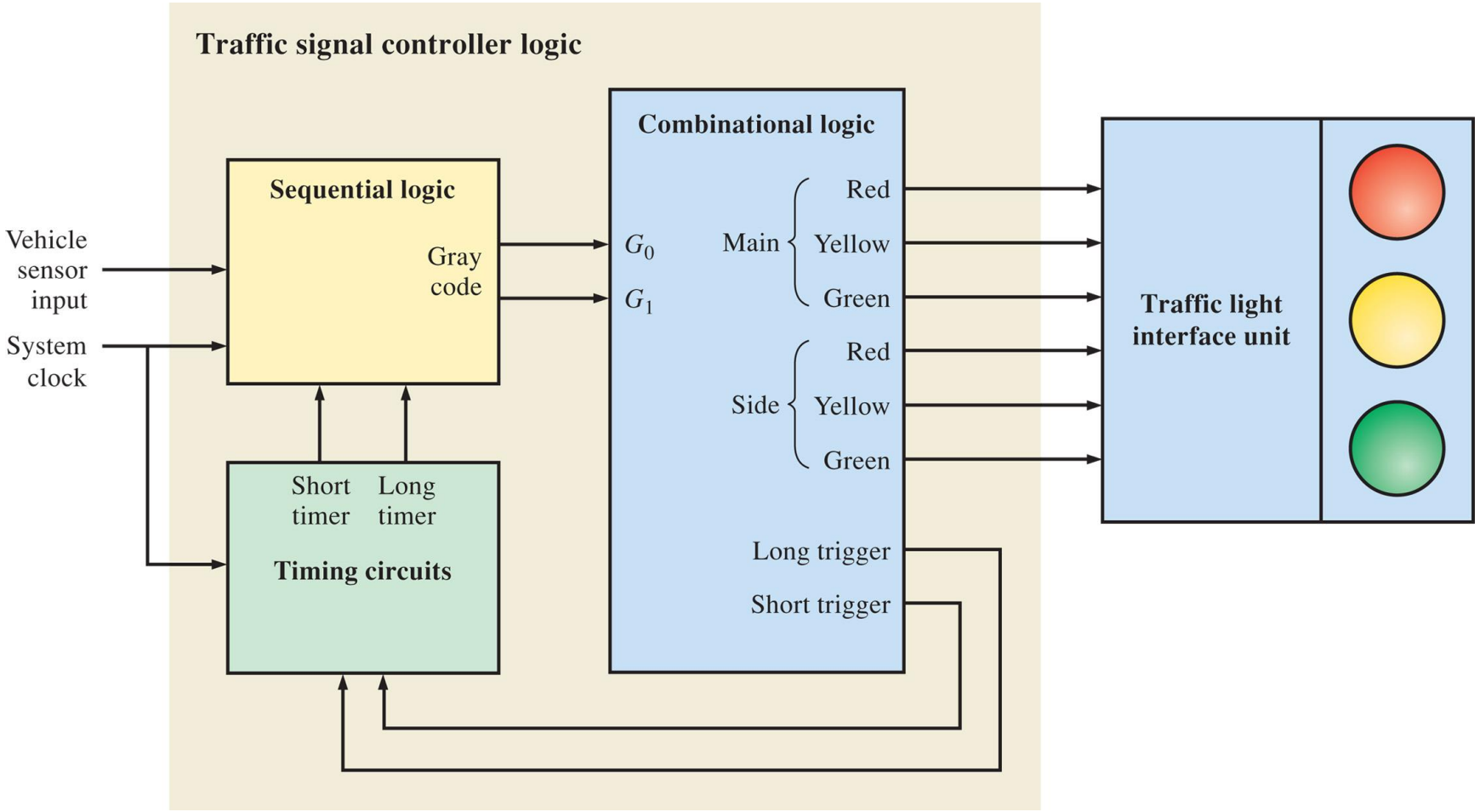


FIGURE 6-65 Block diagram of the combinational logic unit.

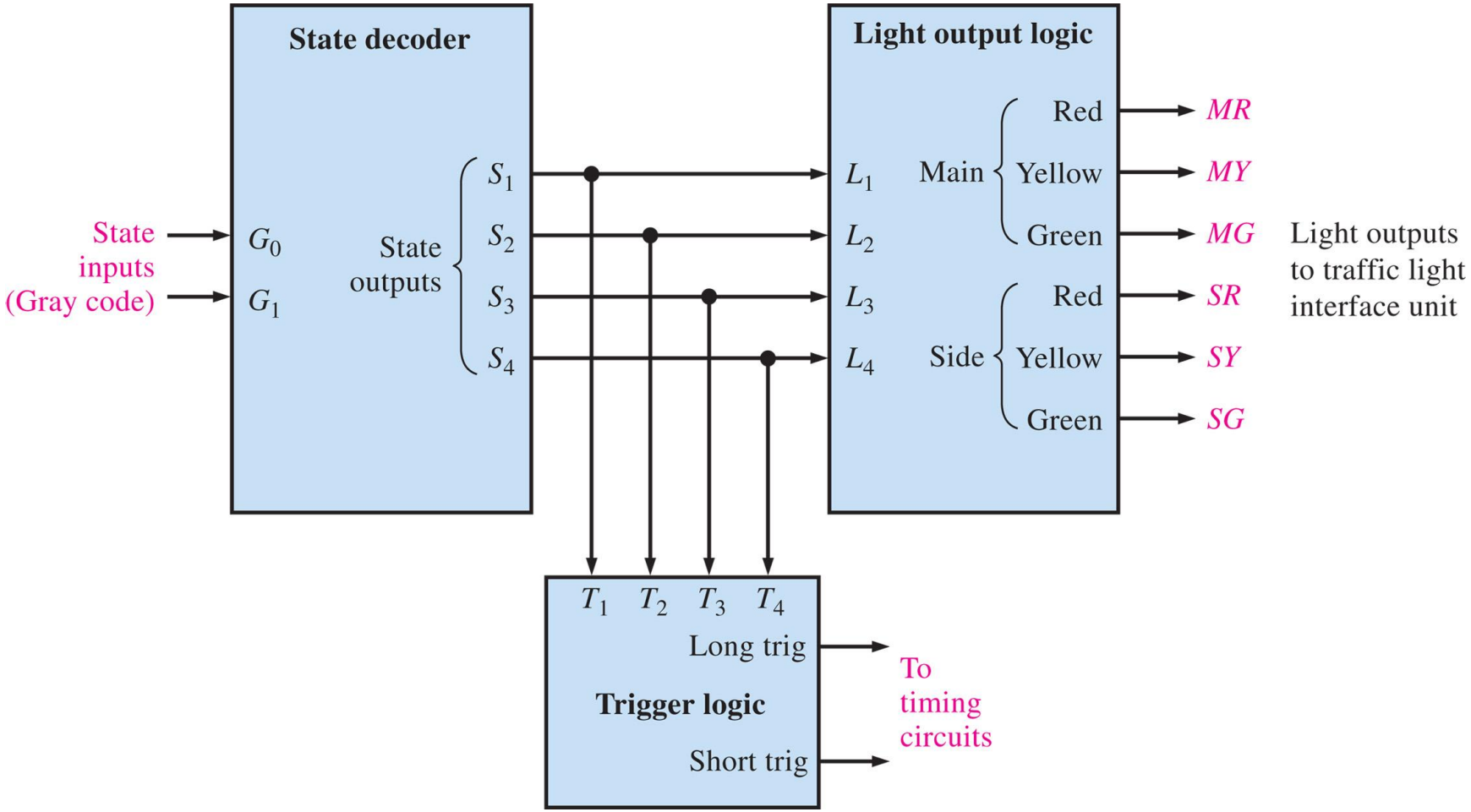


TABLE 6–11

Truth table for the state decoder.

State Inputs (Gray Code)		State Outputs			
G_1	G_0	S_1	S_2	S_3	S_4
0	0	1	0	0	0
0	1	0	1	0	0
1	1	0	0	1	0
1	0	0	0	0	1

FIGURE 6-66 State decoder logic.

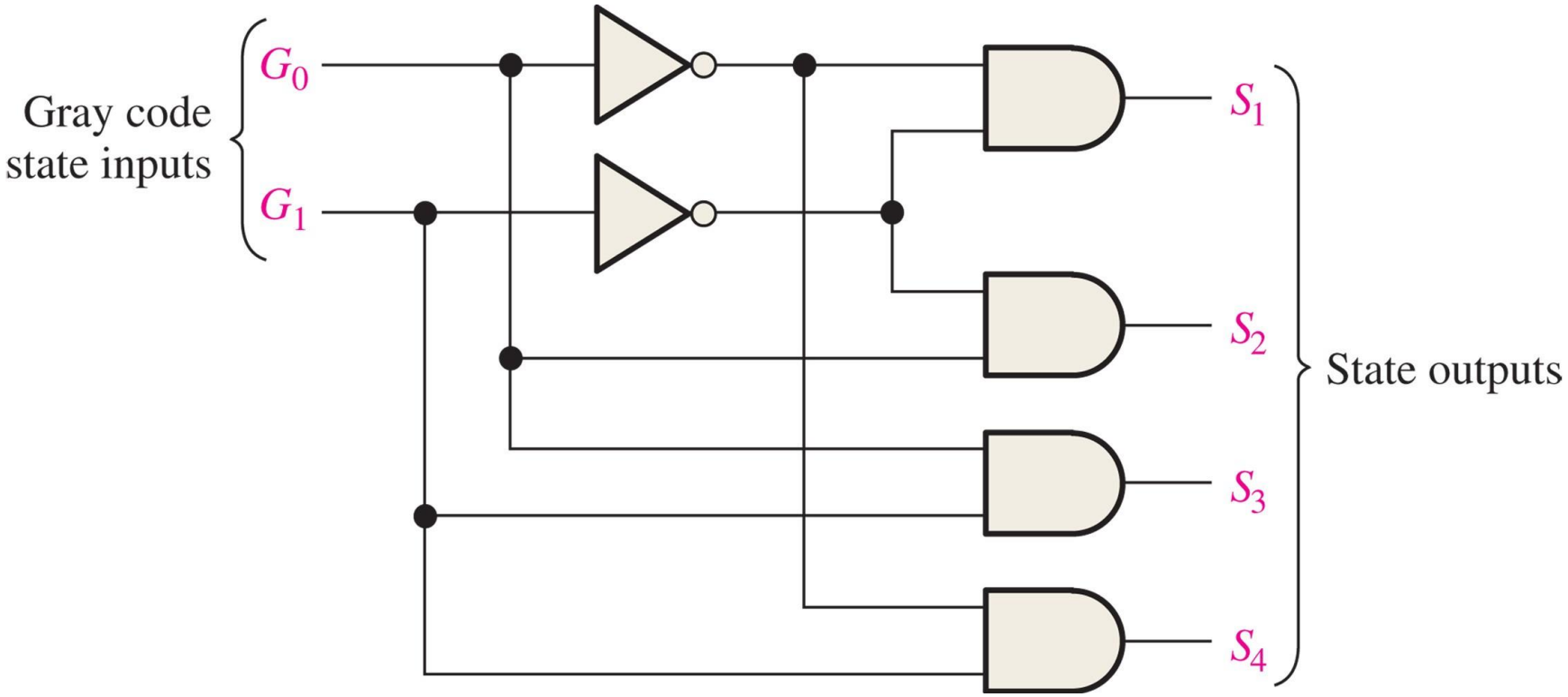


FIGURE 6-67 Light output logic.

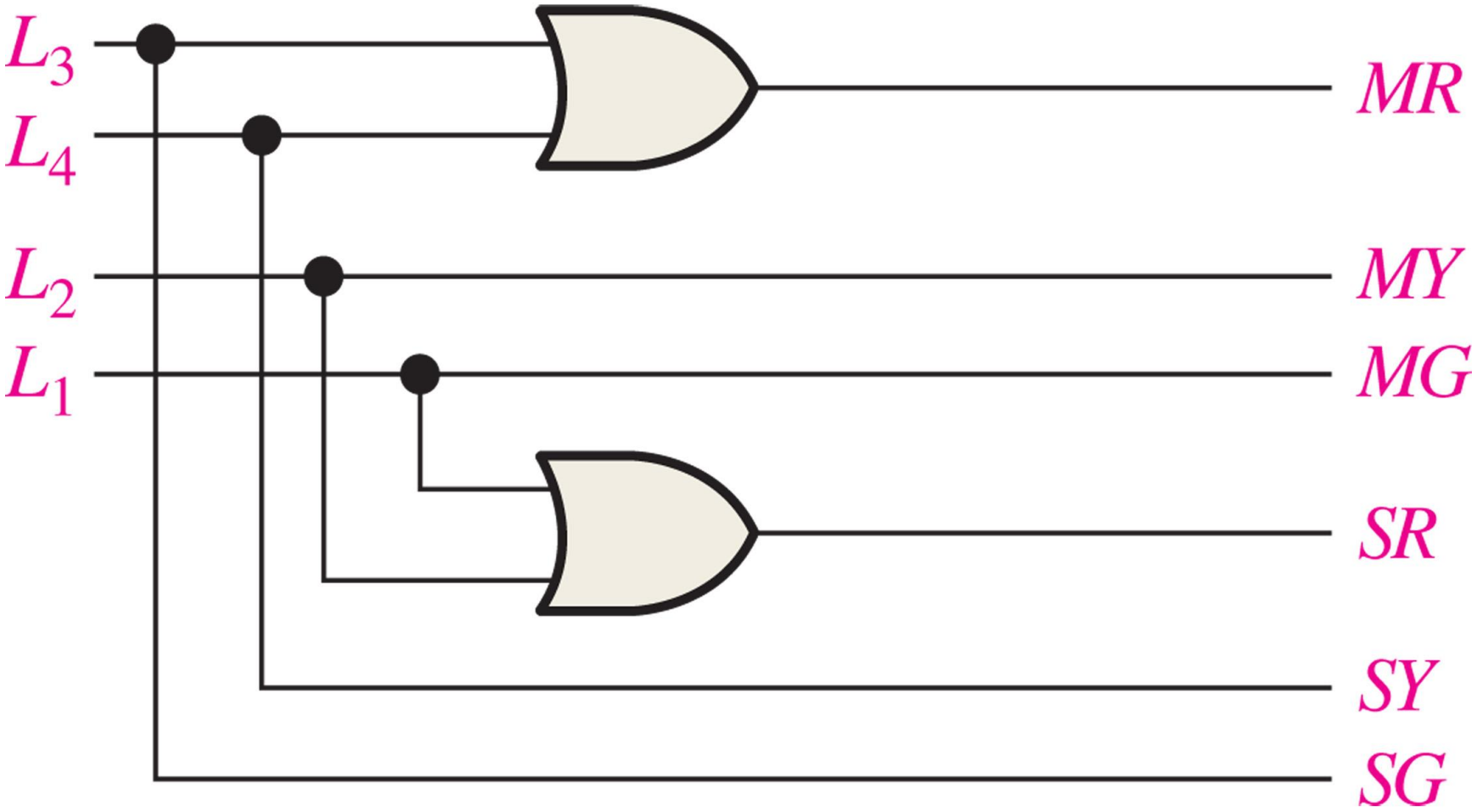


FIGURE 6-68 Trigger logic.

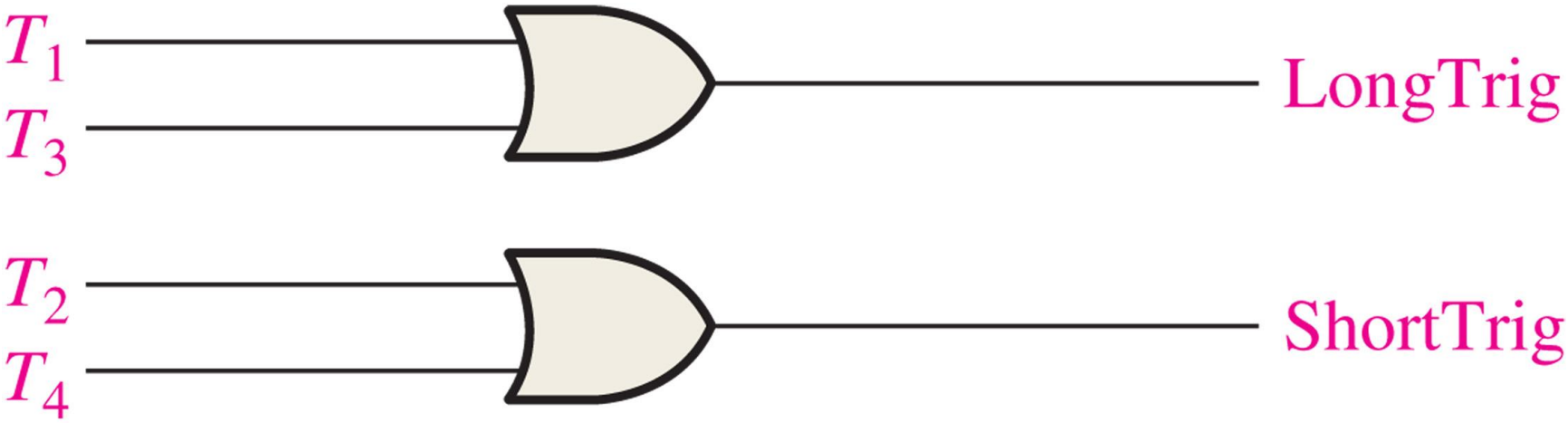


FIGURE 6-71

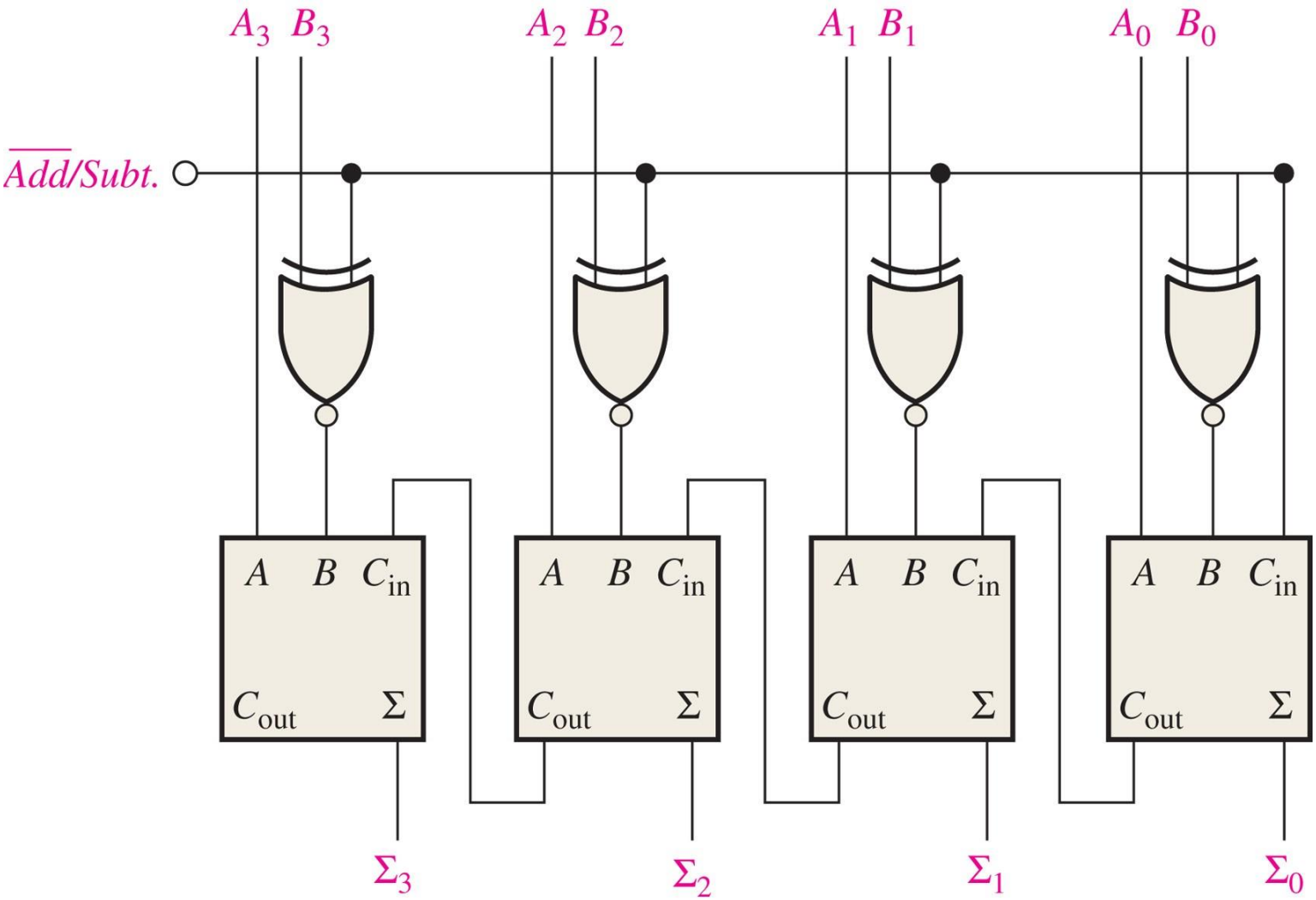


FIGURE 6-94