جامعة فلسطين التقنية - خضوري
Palestine Technical University - Kadoorie

Digital Electronics and Logic Design

Combinational Logic Analysis

Dr. Jafar Saifeddin Jallad

Dept. of Electrical Engineering

Palestine Technical University

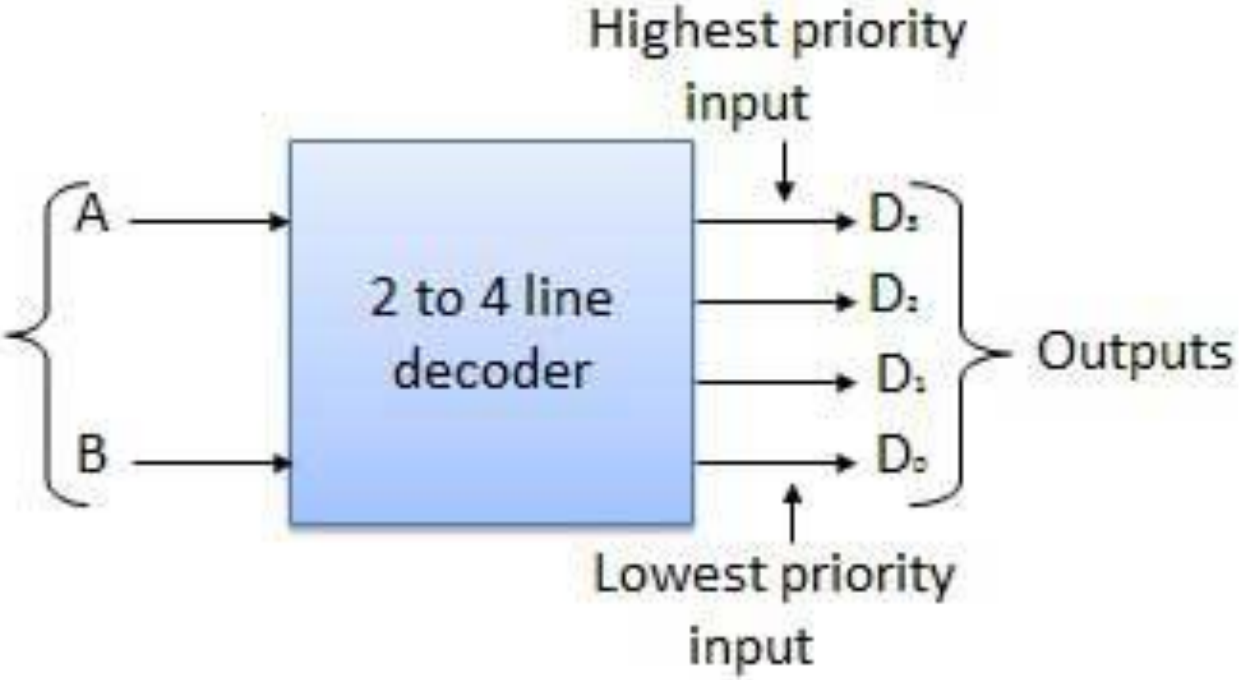
Tulkaram, Palestine

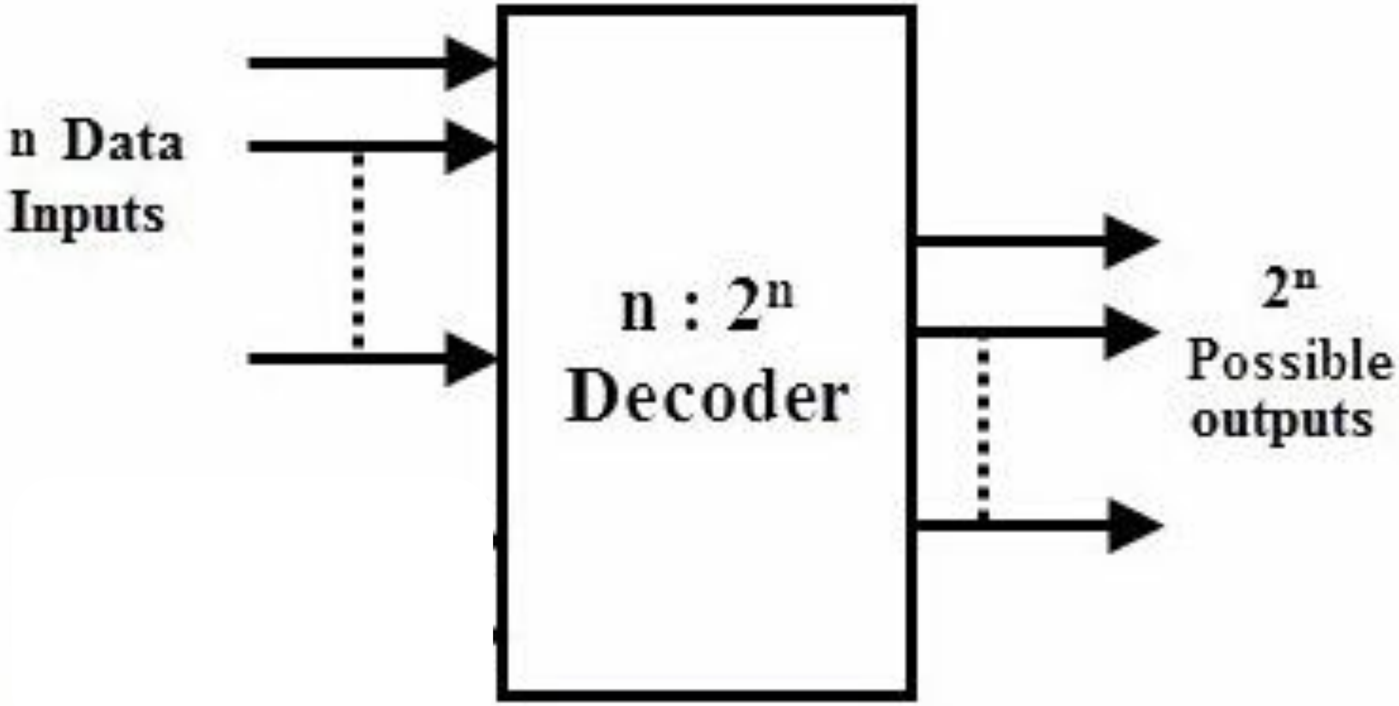
Decoders

Discrete quantities of information are represented in digital systems by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information.

A *decoder* is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit coded information has unused combinations, the decoder may have fewer than 2^n outputs.

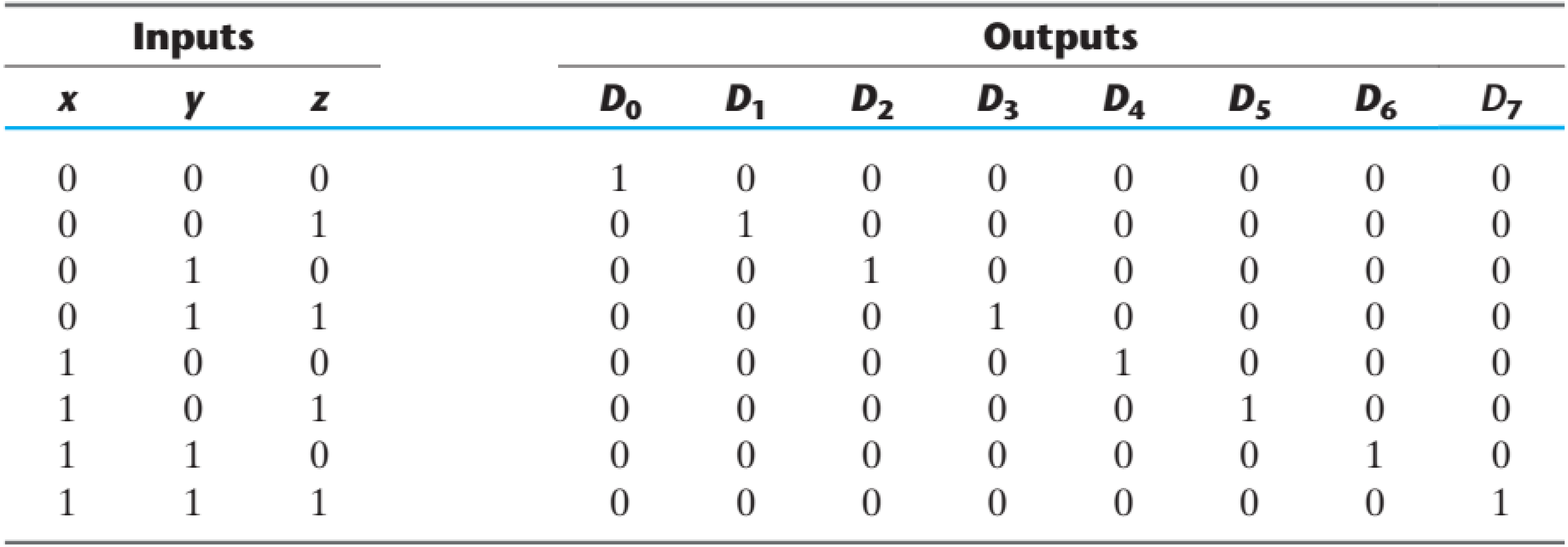
The decoders presented here are called n-to-m-line decoders, where $m \leq 2^n$.





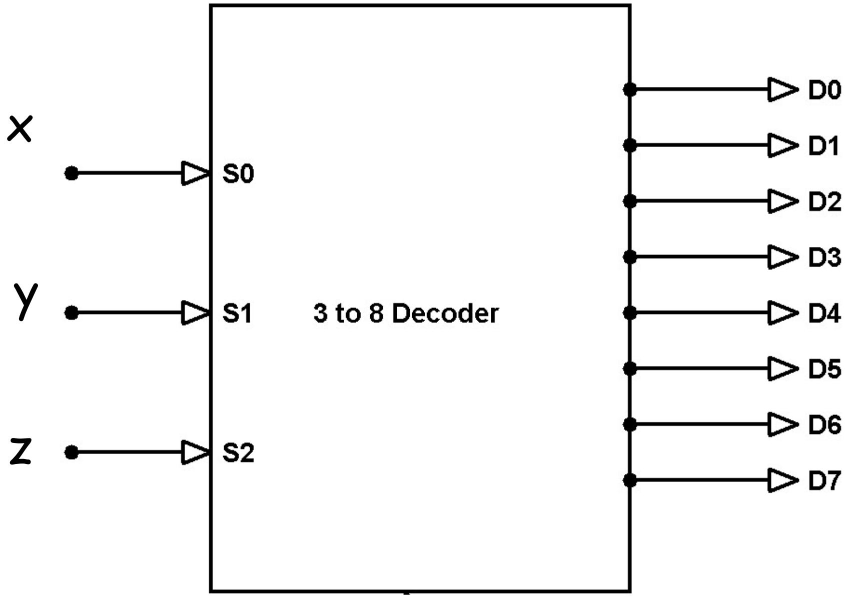
n Data
Inputs
n : 2^n
Decoder
2^n
Possible
outputs

Truth Table of a Three-to-Eight-Line Decoder

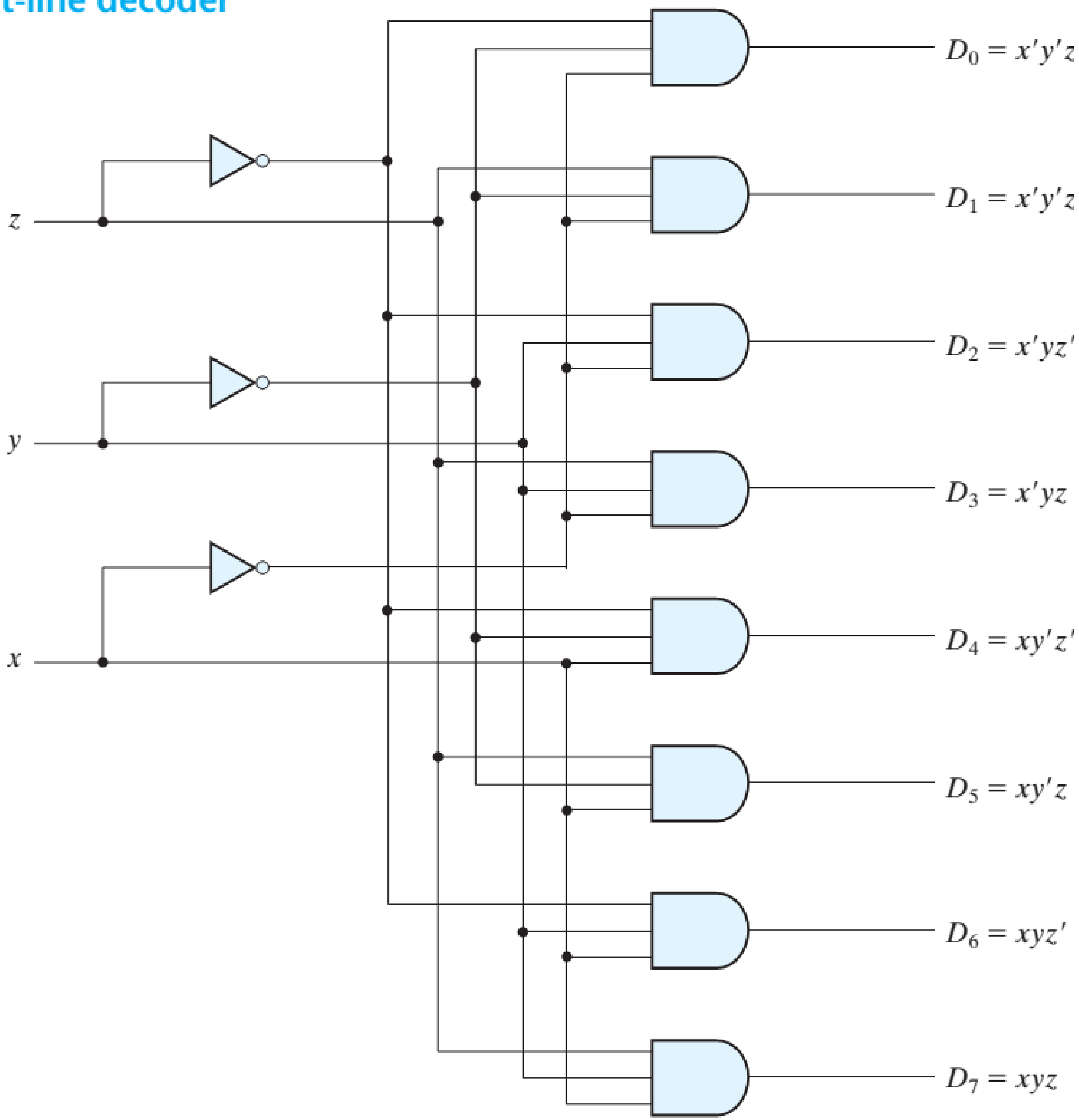


Inputs			Outputs							
x	*y*	*z*	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Active High



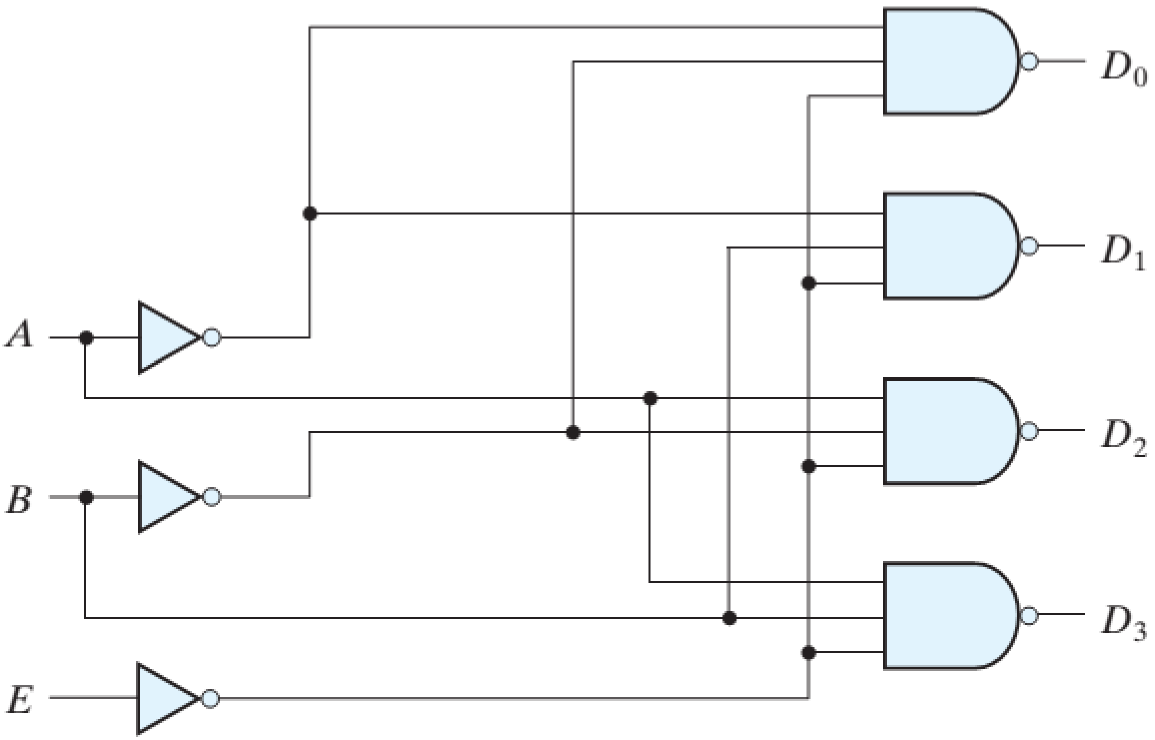
Three-to-eight-line decoder



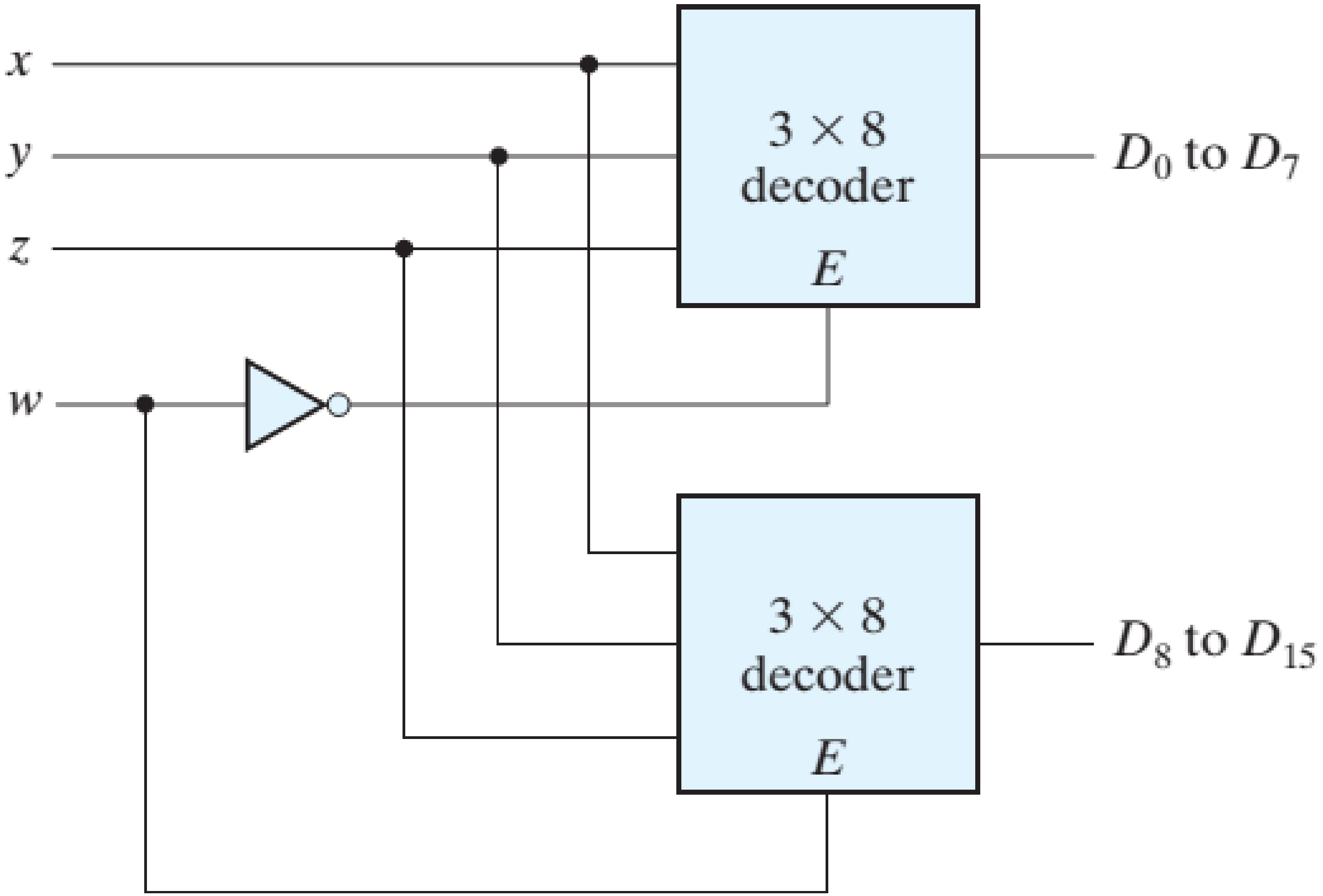
Two-to-four-line decoder with enable input

Active Low

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	1



4×16 decoder constructed with two 3×8 decoders

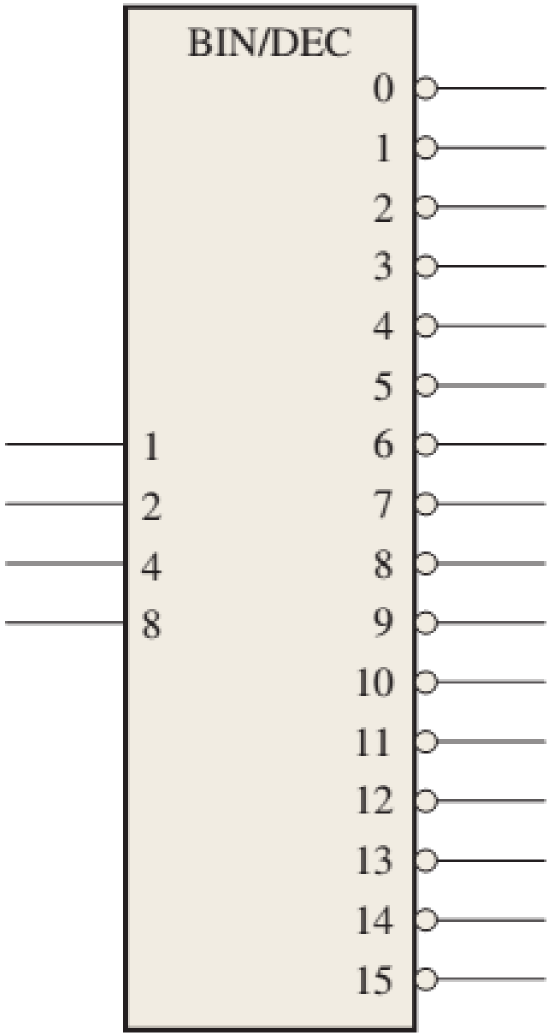


Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with active-LOW outputs.

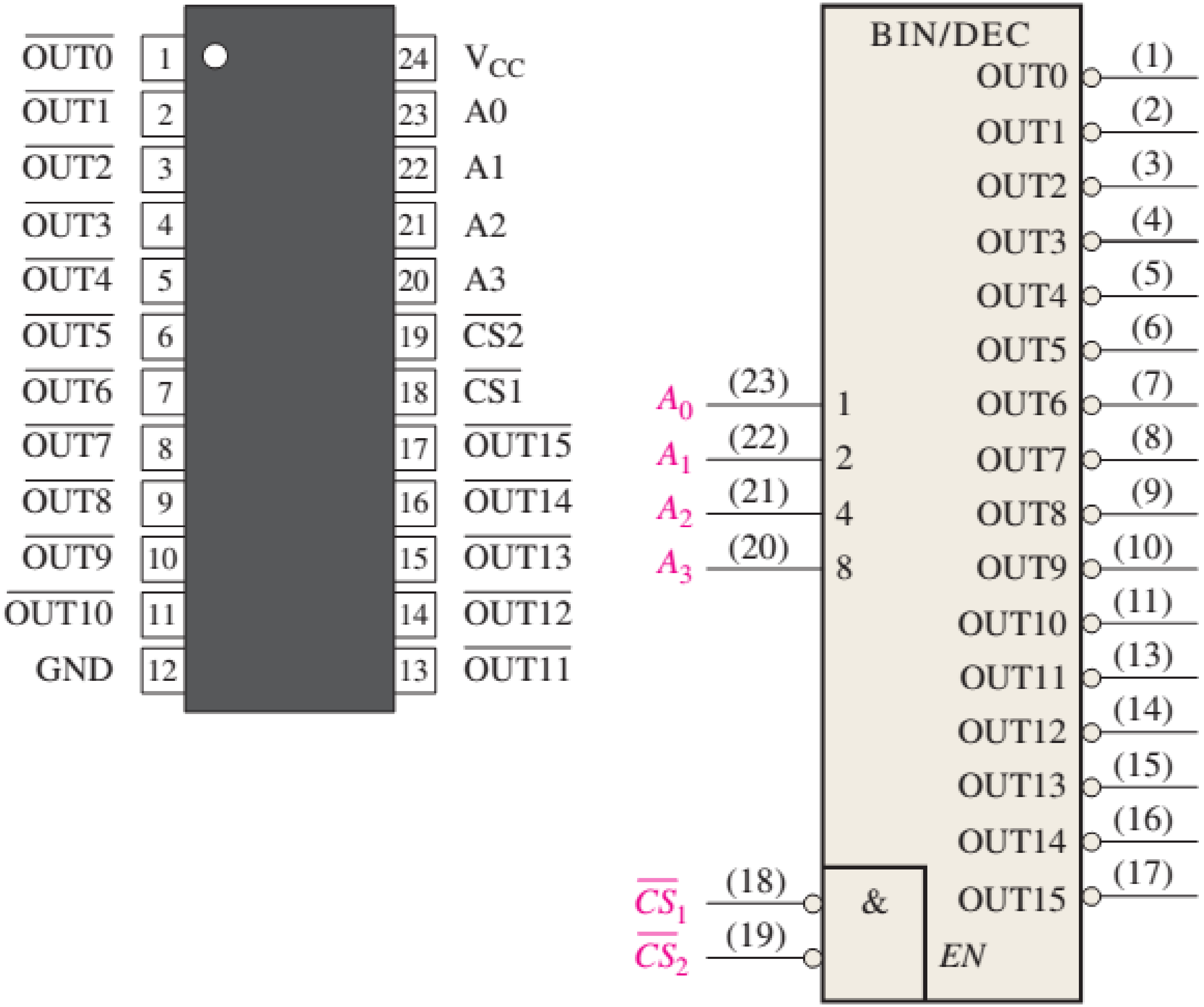
Decimal	Binary Inputs				Decoding	Outputs															
Digit	A_3	A_2	A_1	A_0	Function	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1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

If an active-LOW output is required for each decoded number, the entire decoder can be implemented with NAND gates and inverters. In order to decode each of the sixteen binary codes, sixteen NAND gates are required (AND gates can be used to produce active-HIGH outputs).

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



The 74HC154 1-of-16 decoder.


$\overline{\text{OUT0}}$ 1
$\overline{\text{OUT1}}$ 2
$\overline{\text{OUT2}}$ 3
$\overline{\text{OUT3}}$ 4
$\overline{\text{OUT4}}$ 5
$\overline{\text{OUT5}}$ 6
$\overline{\text{OUT6}}$ 7
$\overline{\text{OUT7}}$ 8
$\overline{\text{OUT8}}$ 9
$\overline{\text{OUT9}}$ 10
$\overline{\text{OUT10}}$ 11
GND 12
24 V_{CC}
23 A0
22 A1
21 A2
20 A3
19 $\overline{\text{CS2}}$
18 $\overline{\text{CS1}}$
17 $\overline{\text{OUT15}}$
16 $\overline{\text{OUT14}}$
15 $\overline{\text{OUT13}}$
14 $\overline{\text{OUT12}}$
13 $\overline{\text{OUT11}}$
BIN/DEC
A_0 (23) 1
A_1 (22) 2
A_2 (21) 4
A_3 (20) 8
$\overline{CS}_1$ (18)
$\overline{CS}_2$ (19)
&
EN
OUT0 (1)
OUT1 (2)
OUT2 (3)
OUT3 (4)
OUT4 (5)
OUT5 (6)
OUT6 (7)
OUT7 (8)
OUT8 (9)
OUT9 (10)
OUT10 (11)
OUT11 (13)
OUT12 (14)
OUT13 (15)
OUT14 (16)
OUT15 (17)

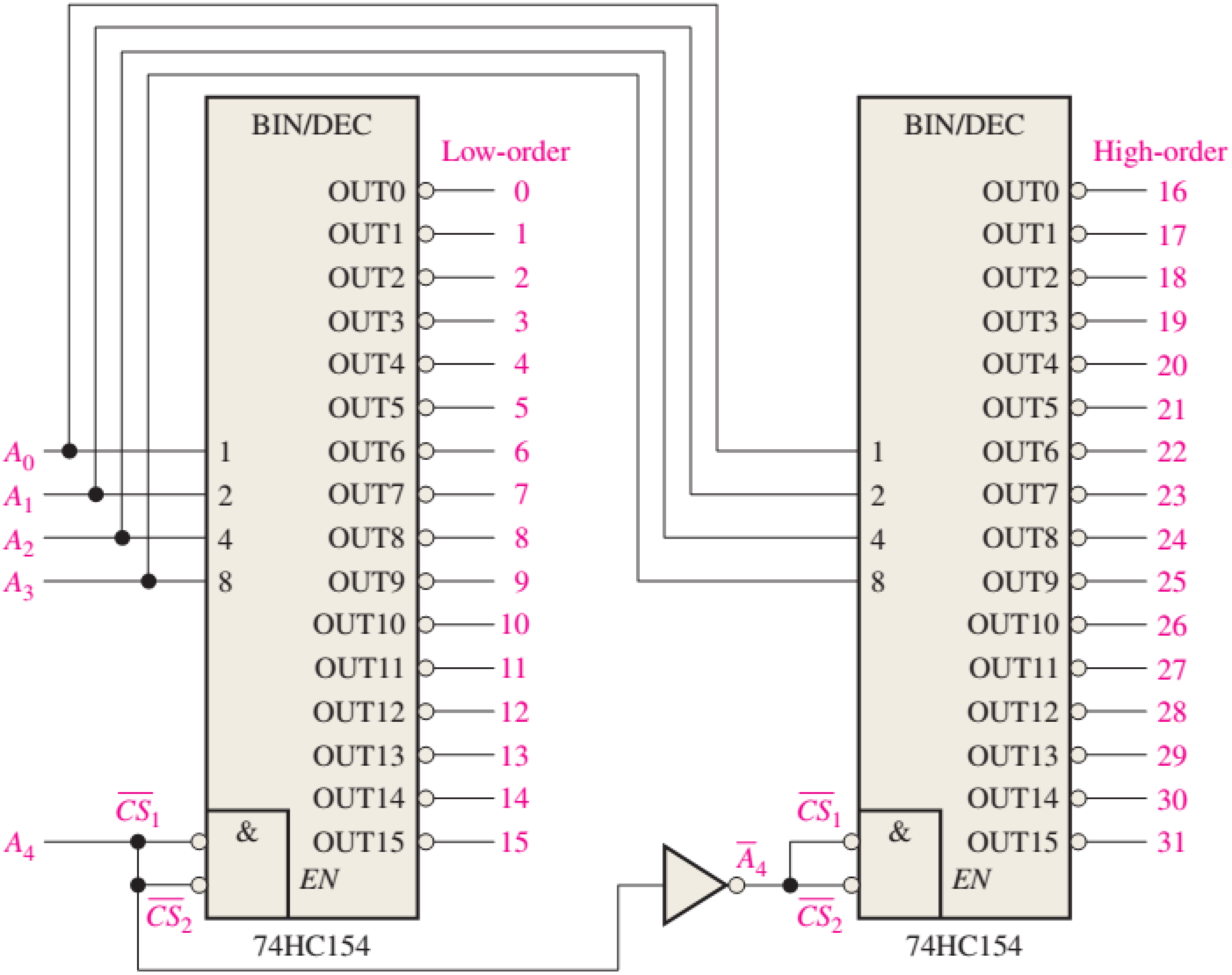
(a) Pin diagram

(b) Logic symbol

EXAMPLE

A certain application requires that a 5-bit number be decoded. Use 74HC154 decoders to implement the logic. The binary number is represented by the format $A_4A_3A_2A_1A_0$.

A 5-bit decoder using 74HC154s.



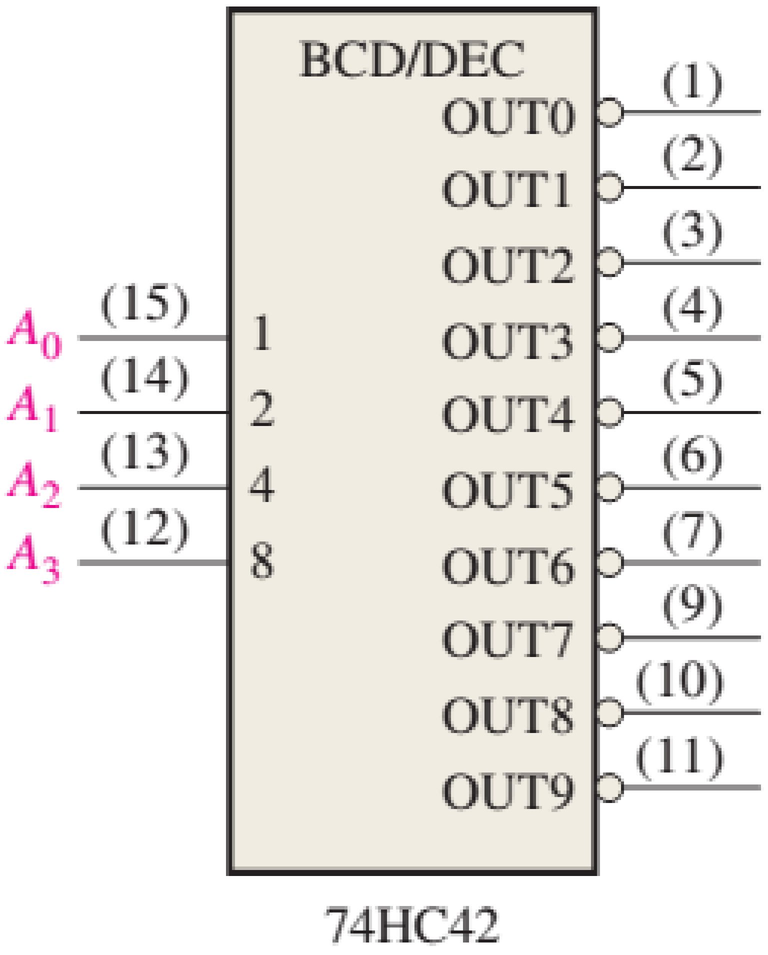
The BCD-to-Decimal Decoder

4-line-to-10-line decoder

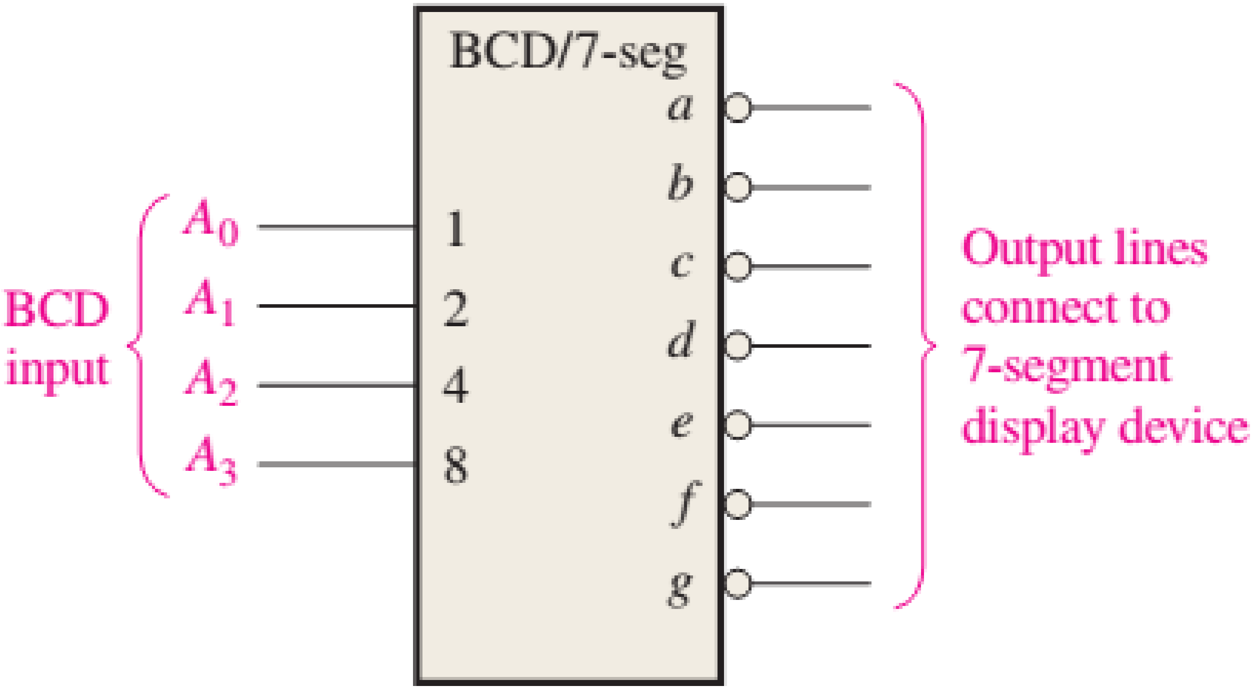
BCD decoding functions.

Decimal Digit	BCD Code				Decoding Function
	A_3	A_2	A_1	A_0	
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	$A_3\overline{A}_2\overline{A}_1A_0$

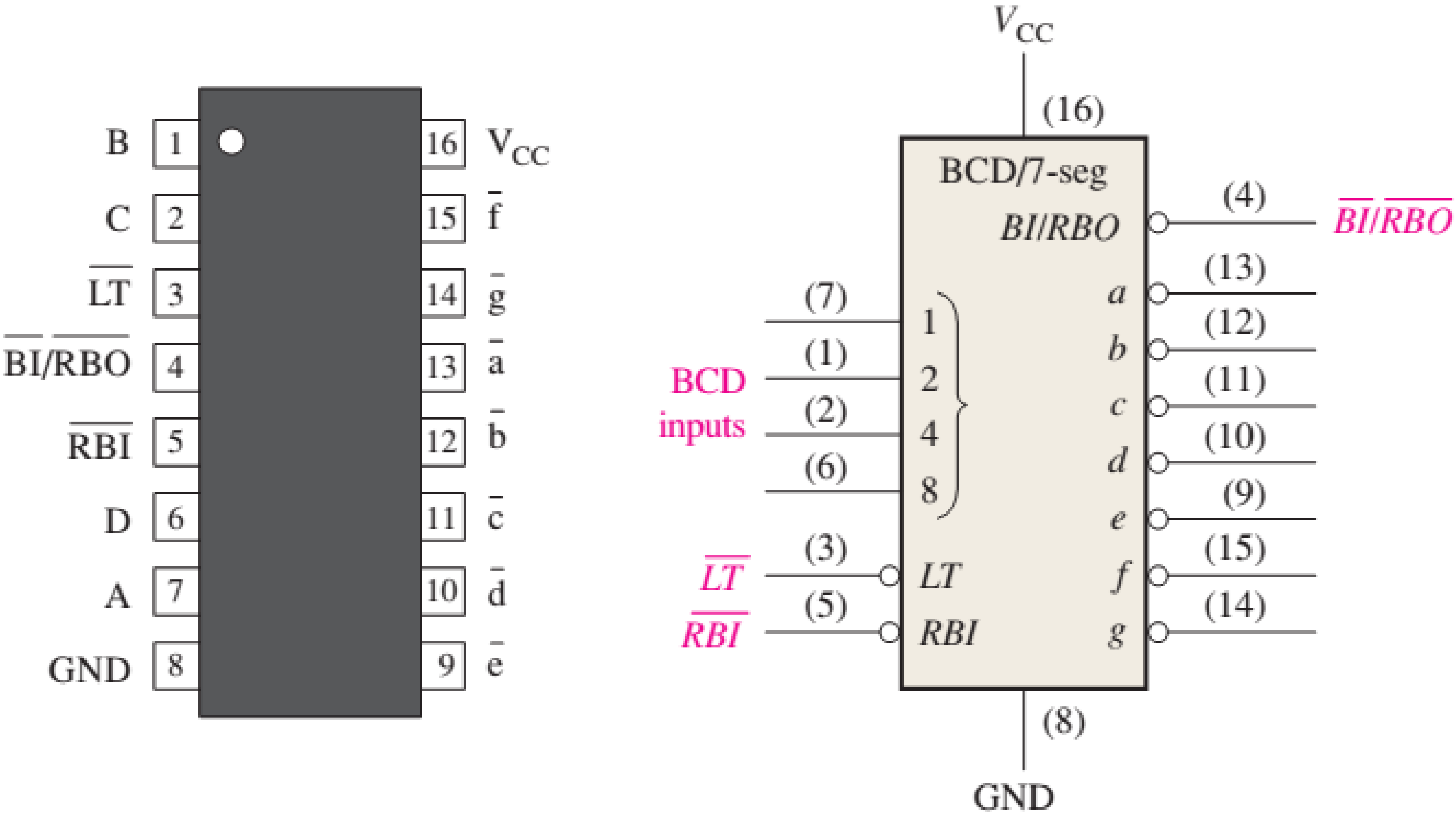
The 74HC42 BCD-to-decimal decoder.



The BCD-to-7-Segment Decoder



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



(a) Pin diagram

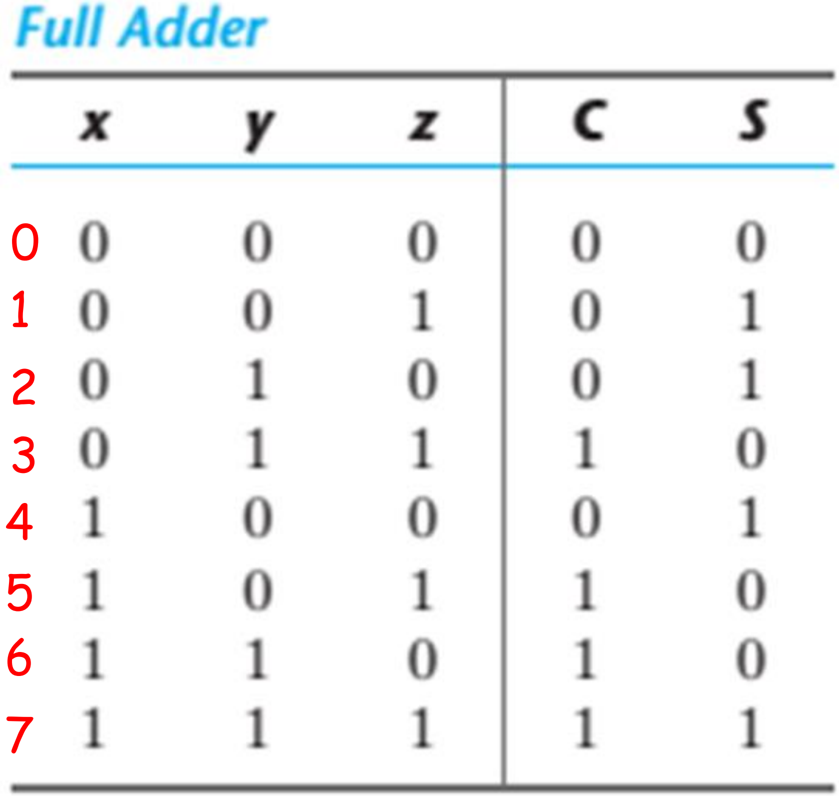
(b) Logic symbol

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

Lamp Test When a LOW is applied to the $\overline{LT}$ input and the $\overline{BI}/\overline{RBO}$ is HIGH, all of the seven segments in the display are turned on. Lamp test is used to verify that no segments are burned out.

Combinational Logic Implementation

Implementation of a full adder with a decoder



Full Adder

	x	y	z	C	S
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

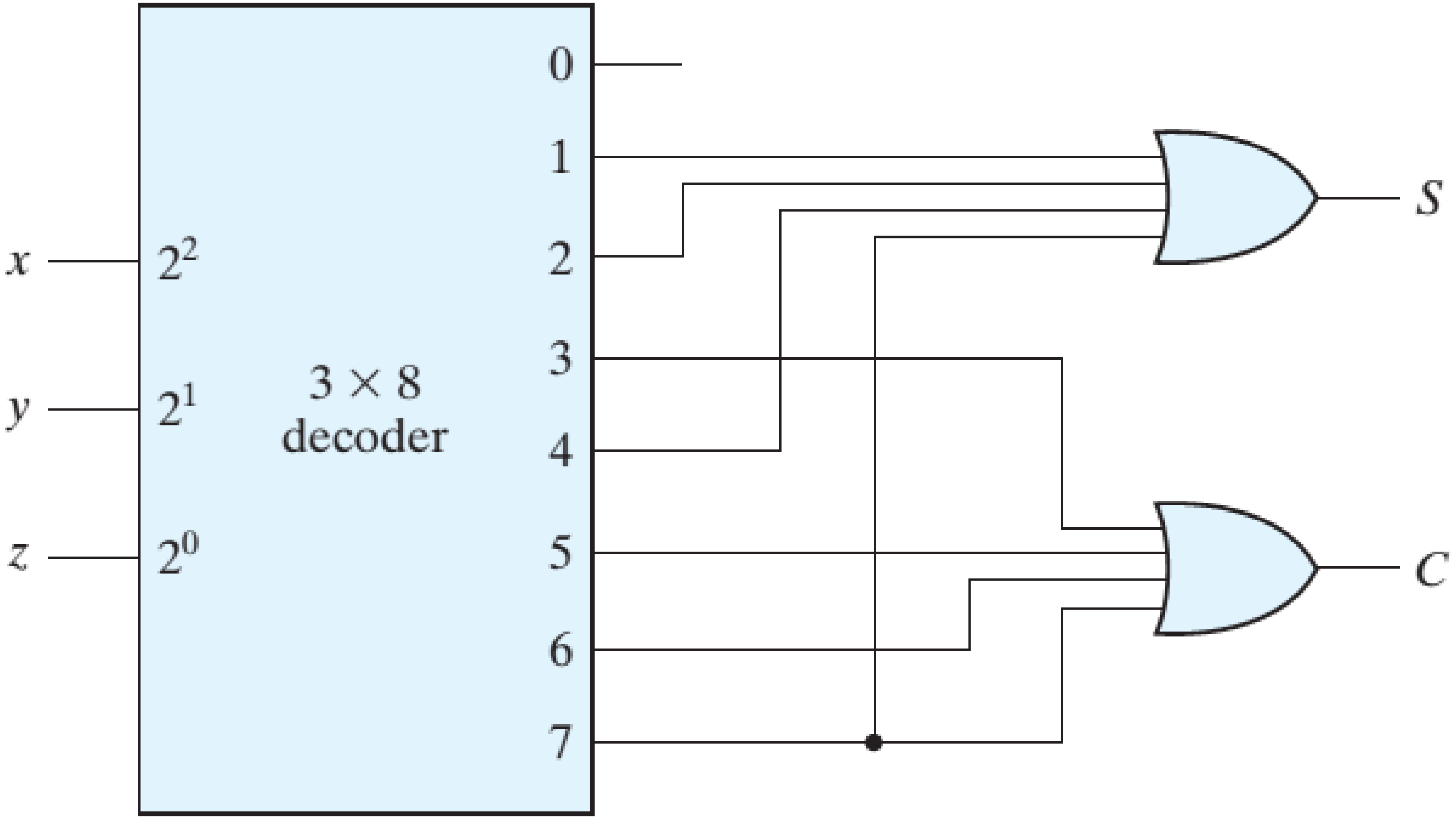
$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

Implementation of a full adder with a decoder

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$

$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$



ENCODERS

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n (or fewer) input lines and n output lines. The output lines, as an aggregate, generate the binary code corresponding to the input value. An example of an encoder is the octal-to-binary encoder

Truth Table of an Octal-to-Binary Encoder

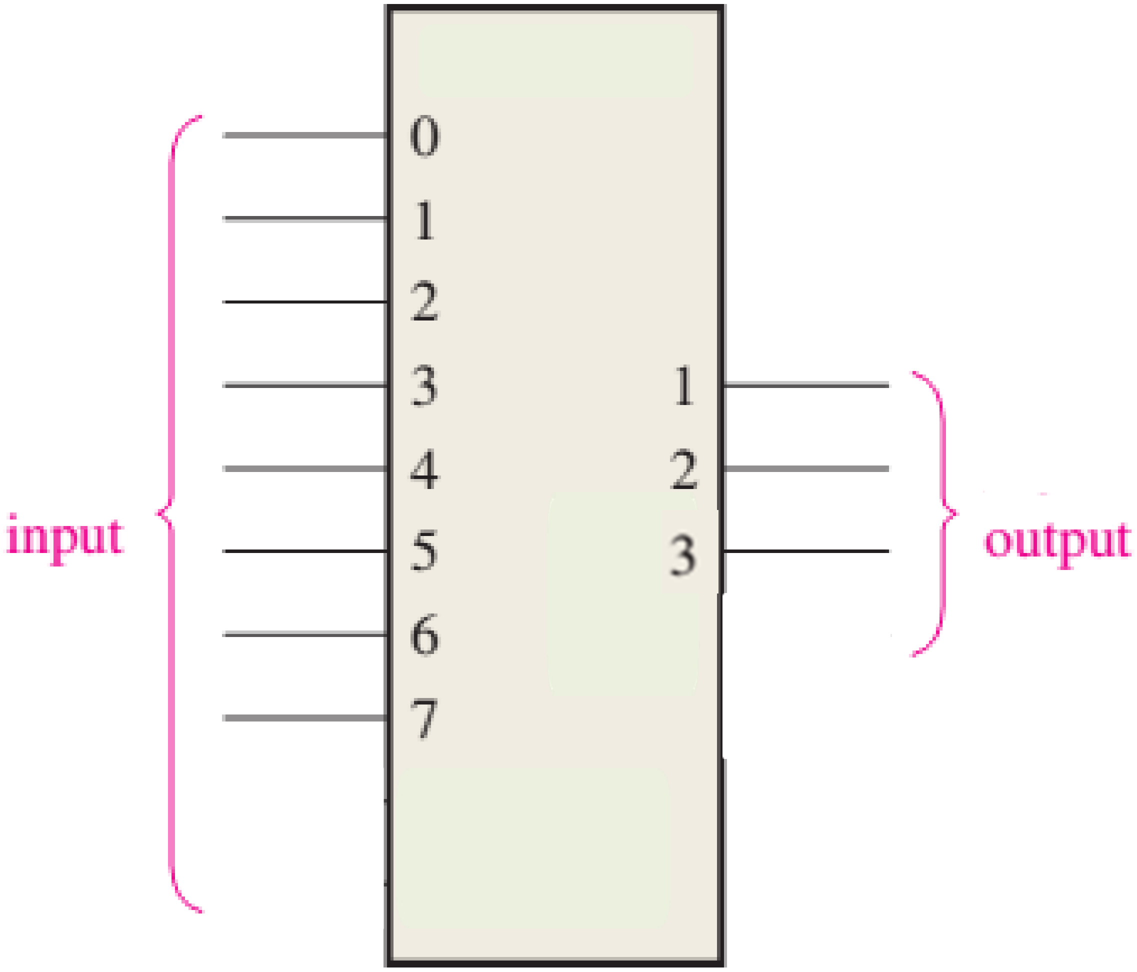
Inputs								**Outputs**		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

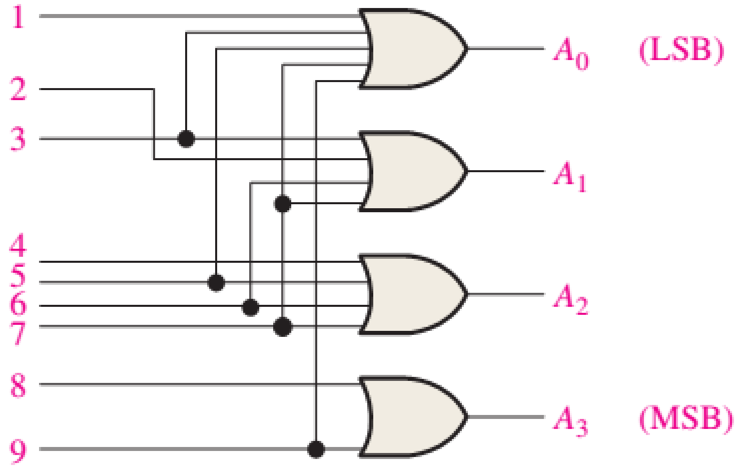
$$x = D_4 + D_5 + D_6 + D_7$$

an Octal-to-Binary Encoder



The Decimal-to-BCD Encoder

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



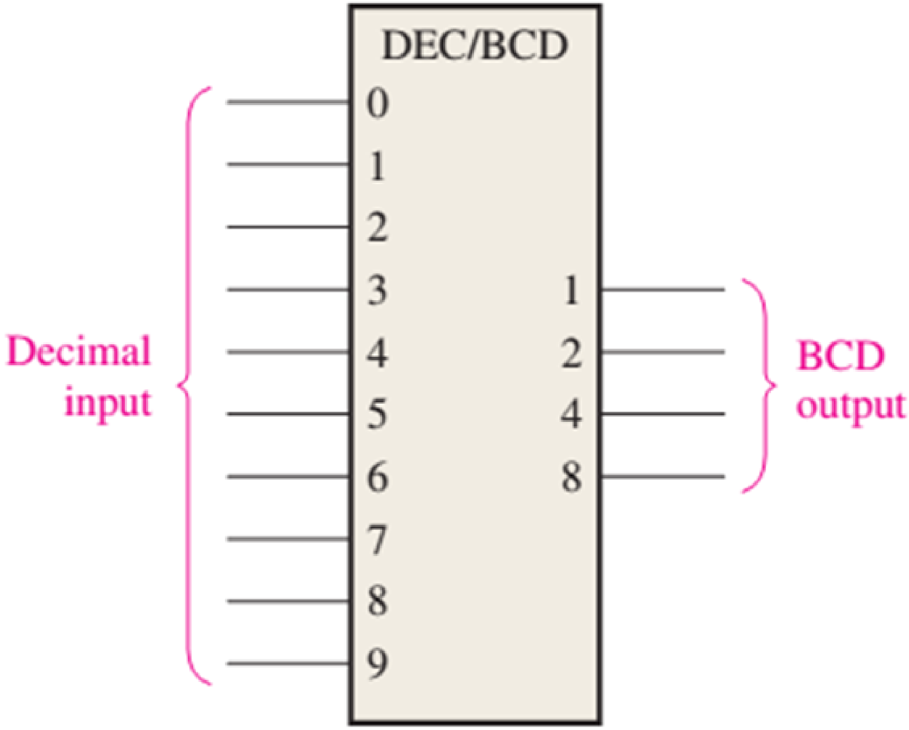
$$A_3 = 8 + 9$$

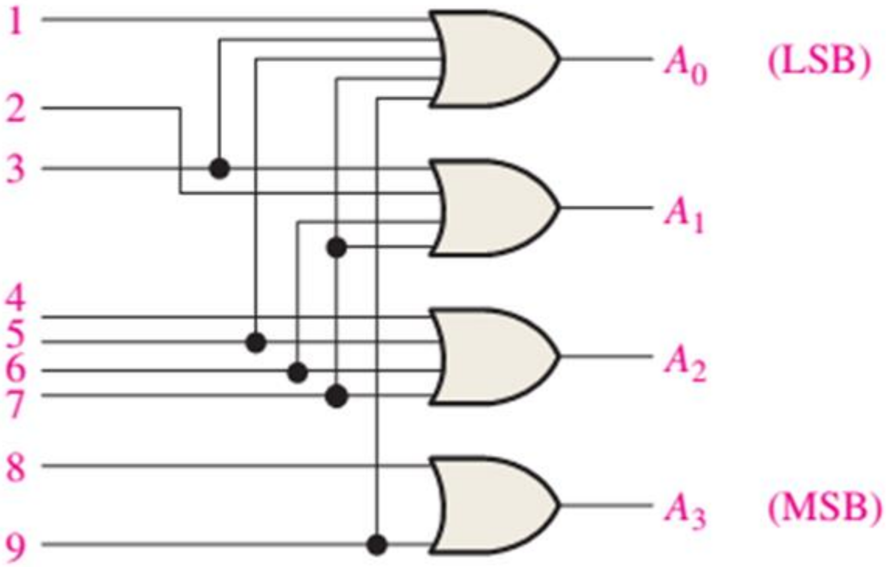
$$A_2 = 4 + 5 + 6 + 7$$

$$A_1 = 2 + 3 + 6 + 7$$

$$A_0 = 1 + 3 + 5 + 7 + 9$$

The Decimal-to-BCD Encoder





$$A_3 = 8 + 9$$

$$A_2 = 4 + 5 + 6 + 7$$

$$A_1 = 2 + 3 + 6 + 7$$

$$A_0 = 1 + 3 + 5 + 7 + 9$$

Priority Encoder

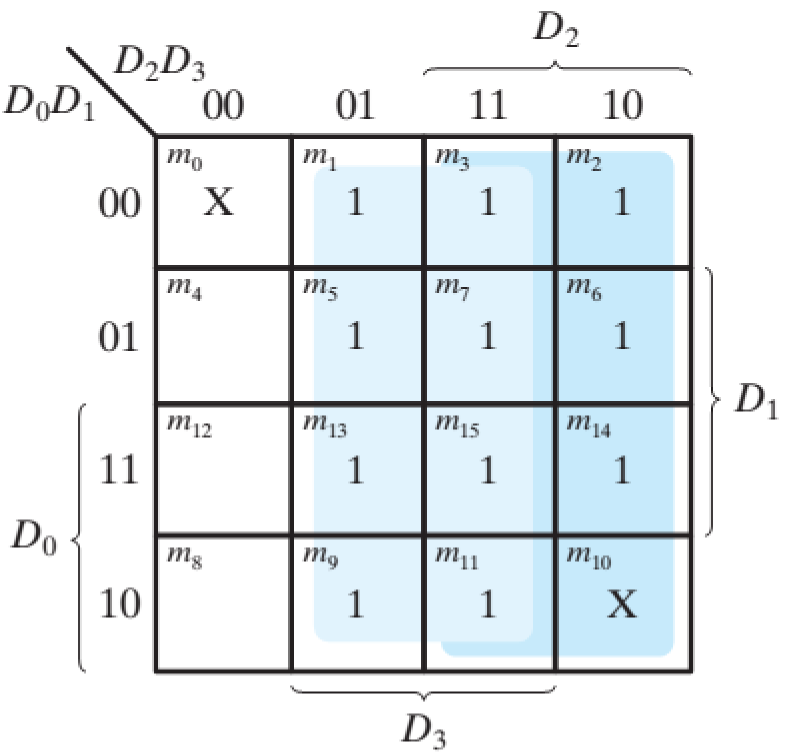
A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. The truth table of a four-input priority encoder is given in Table . In addition to the two outputs *x* and *y*, the circuit has a third output designated by *V*; this is a *valid* bit indicator that is set to 1 when one or more inputs are equal to 1. If all inputs are 0, there is no valid input and *V* is equal to 0.

Four-input priority encoder

Truth Table of a Priority Encoder

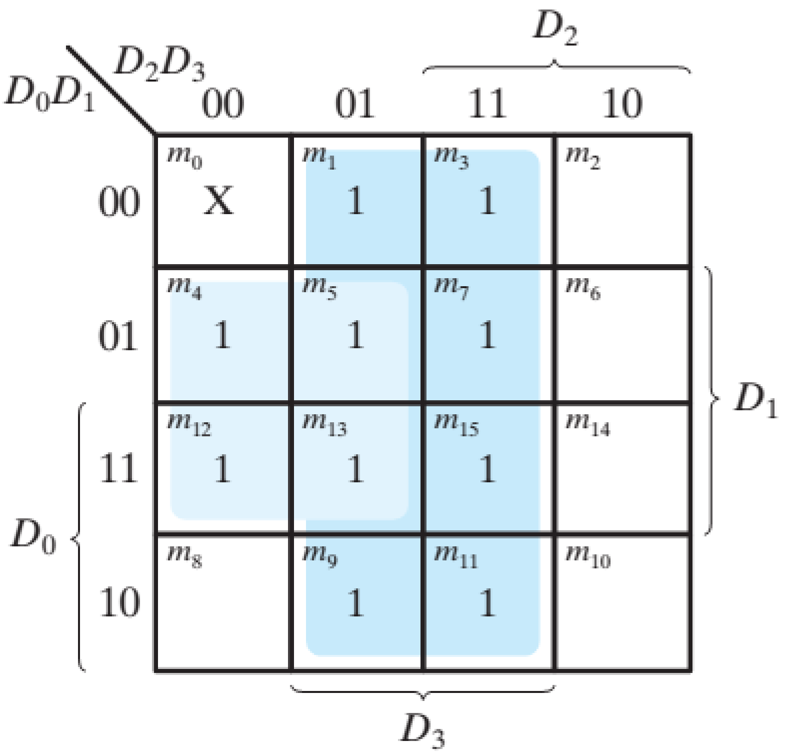
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Four-input priority encoder



$D_0D_1 \backslash D_2D_3$	00	01	11	10
00	m_0 X	m_1 1	m_3 1	m_2 1
01	m_4	m_5 1	m_7 1	m_6 1
11	m_{12}	m_{13} 1	m_{15} 1	m_{14} 1
10	m_8	m_9 1	m_{11} 1	m_{10} X

$x = D_2 + D_3$



$D_0D_1 \backslash D_2D_3$	00	01	11	10
00	m_0 X	m_1 1	m_3 1	m_2
01	m_4 1	m_5 1	m_7 1	m_6
11	m_{12} 1	m_{13} 1	m_{15} 1	m_{14}
10	m_8	m_9 1	m_{11} 1	m_{10}

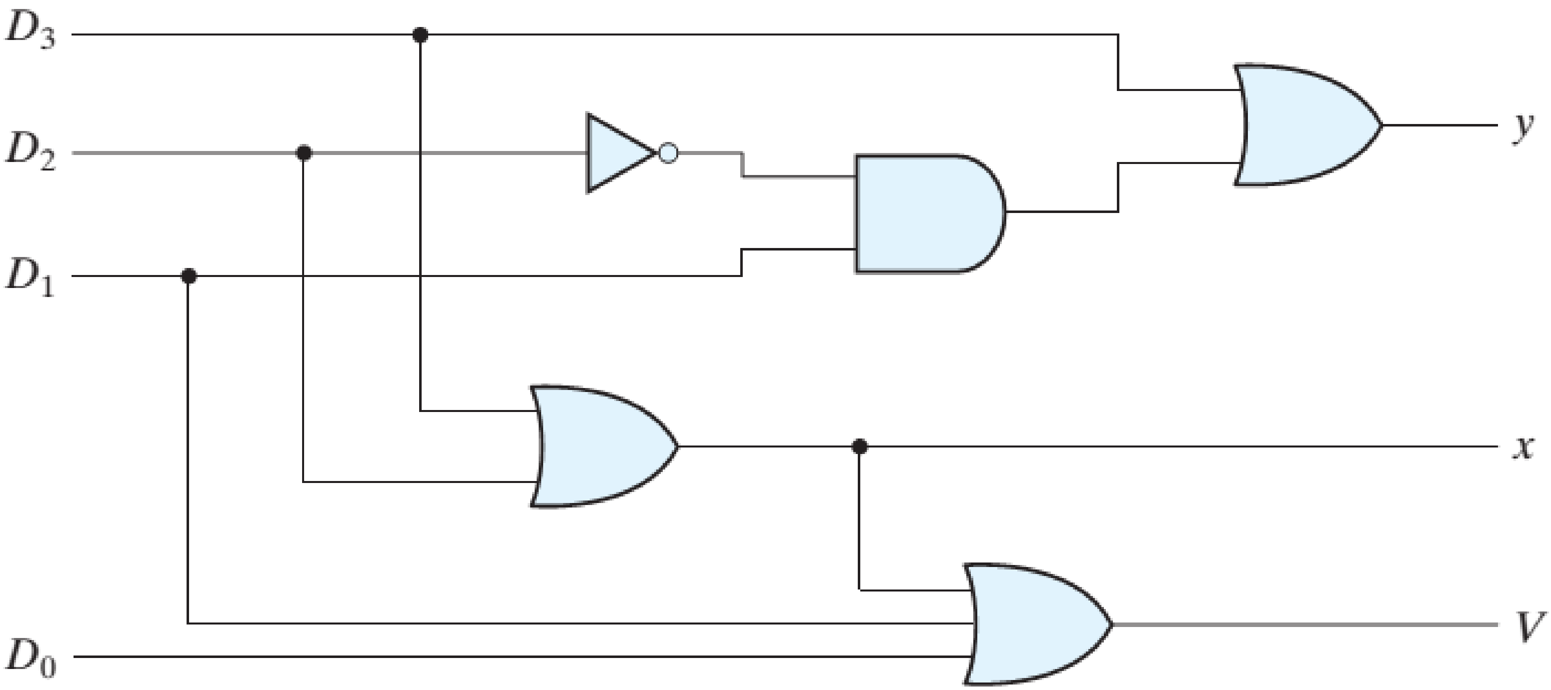
$y = D_3 + D_1D'_2$

$$x = D_2 + D_3$$
$$y = D_3 + D_1\, D'_2$$
$$V = D_0 + D_1 + D_2 + D_3$$

Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Four-input priority encoder



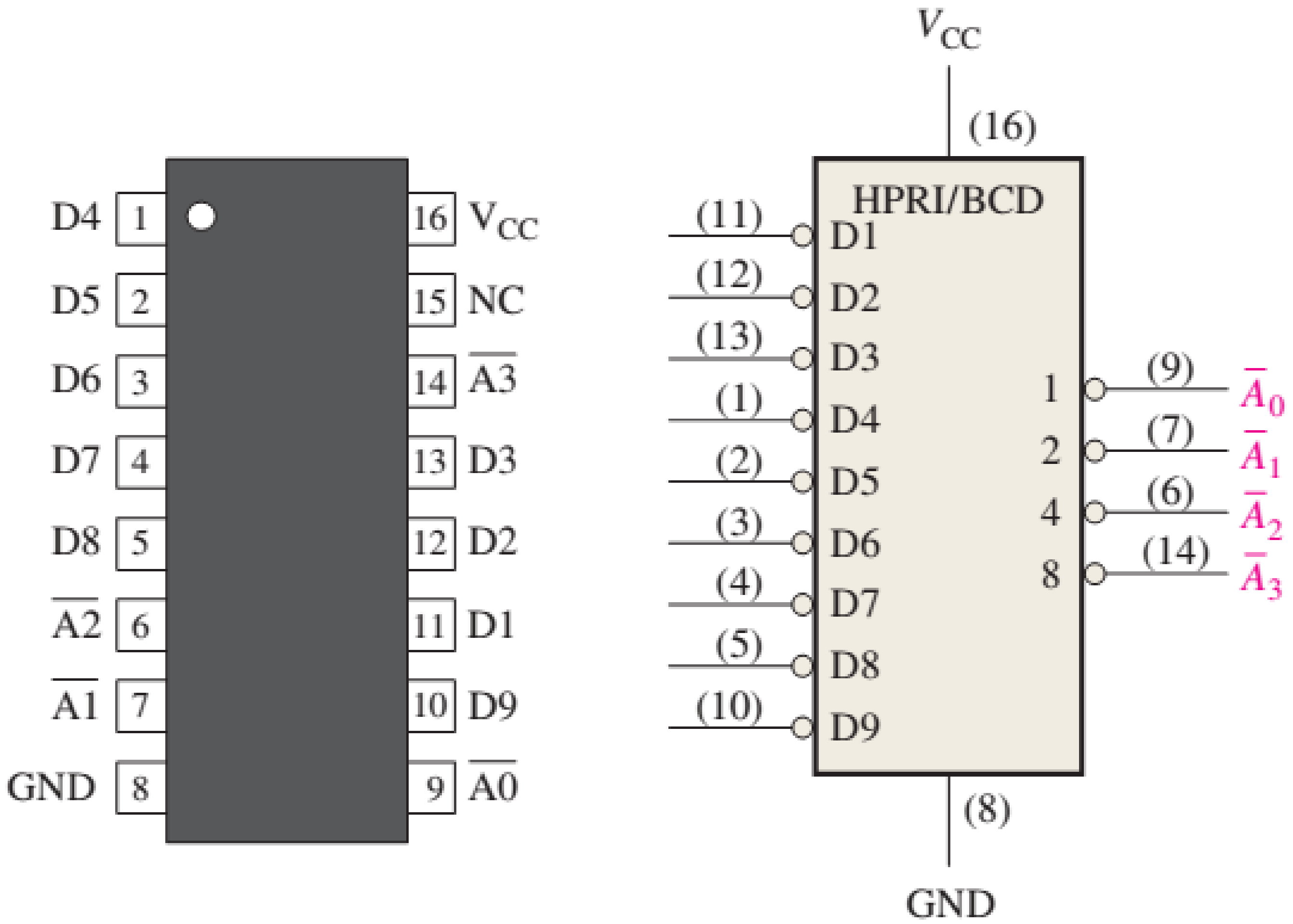
$$x = D_2 + D_3$$

$$y = D_3 + D_1 D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$

DECIMAL-TO-BCD ENCODER

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



(a) Pin diagram

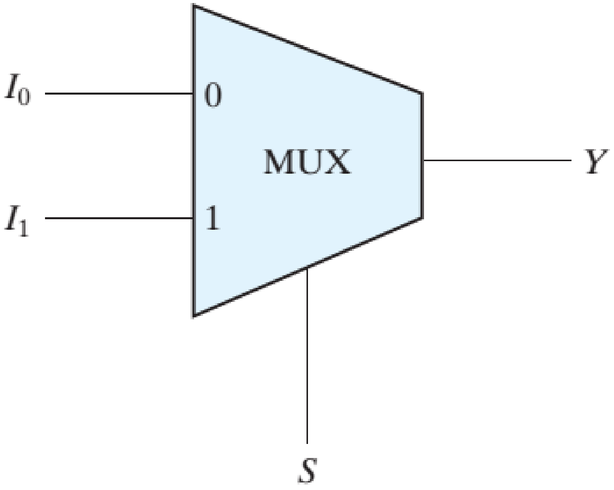
(b) Logic diagram

Multiplexers (Data Selectors)

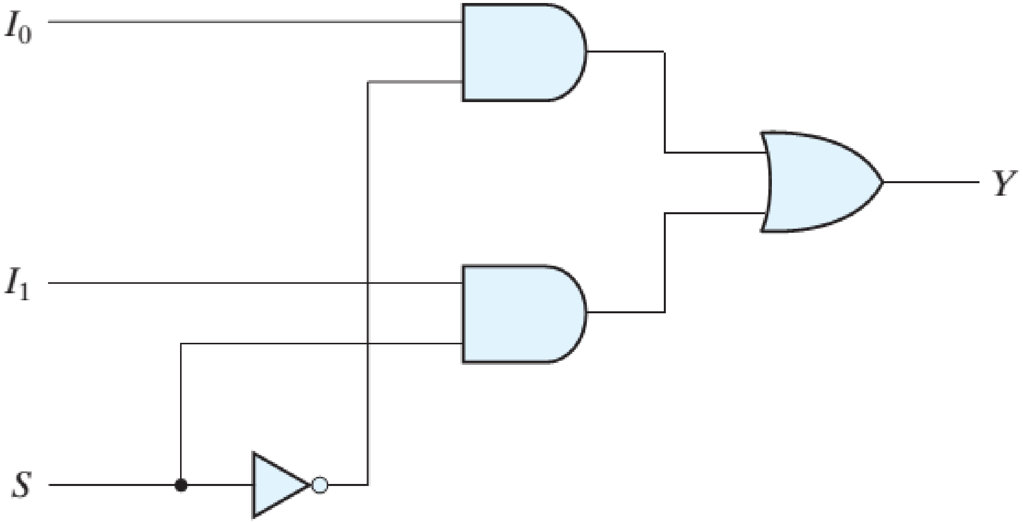
A **multiplexer (MUX)** is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.

many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

Two-to-one-line multiplexer



(b) Block diagram

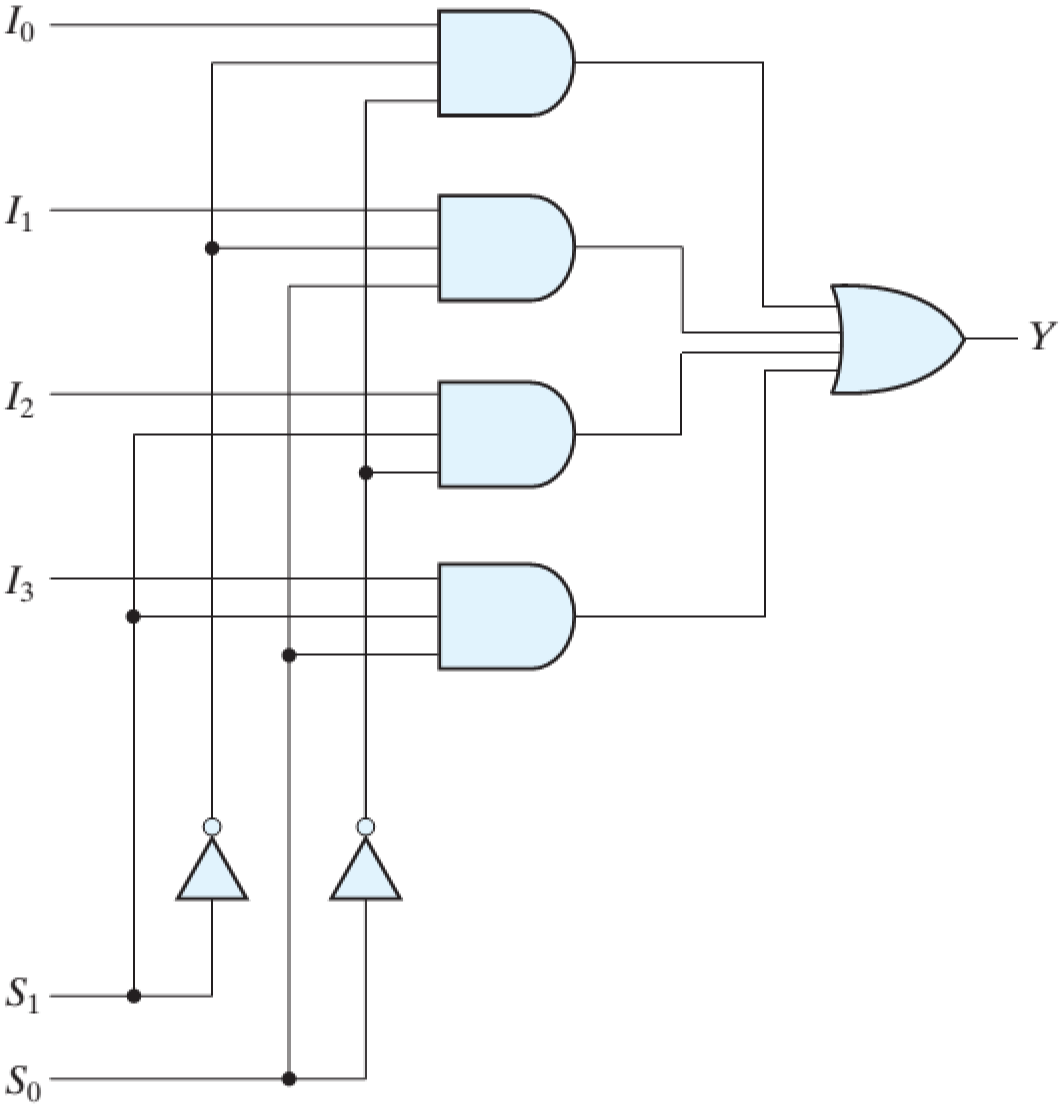


(a) Logic diagram

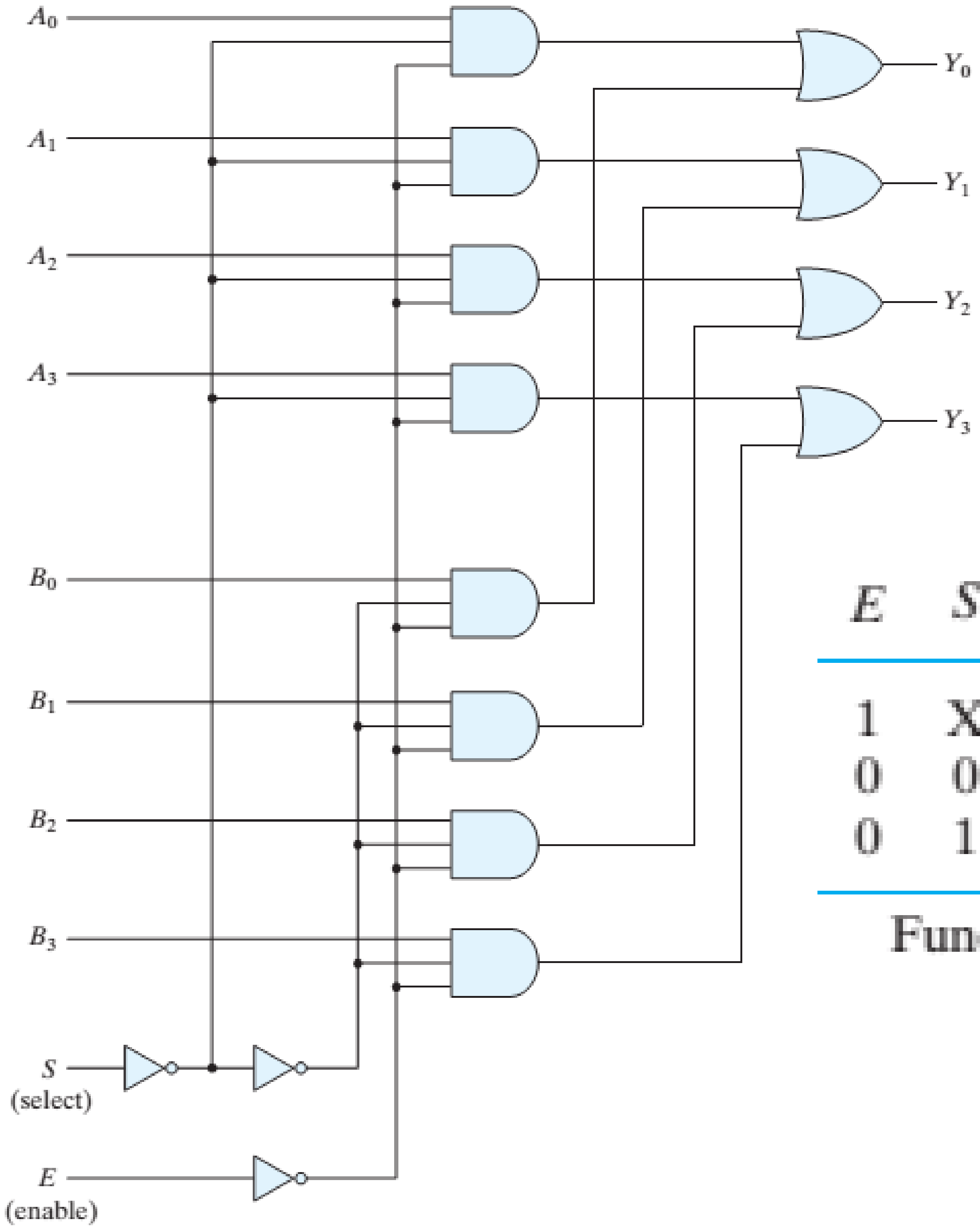
Four-to-one-line multiplexer

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table



(a) Logic diagram

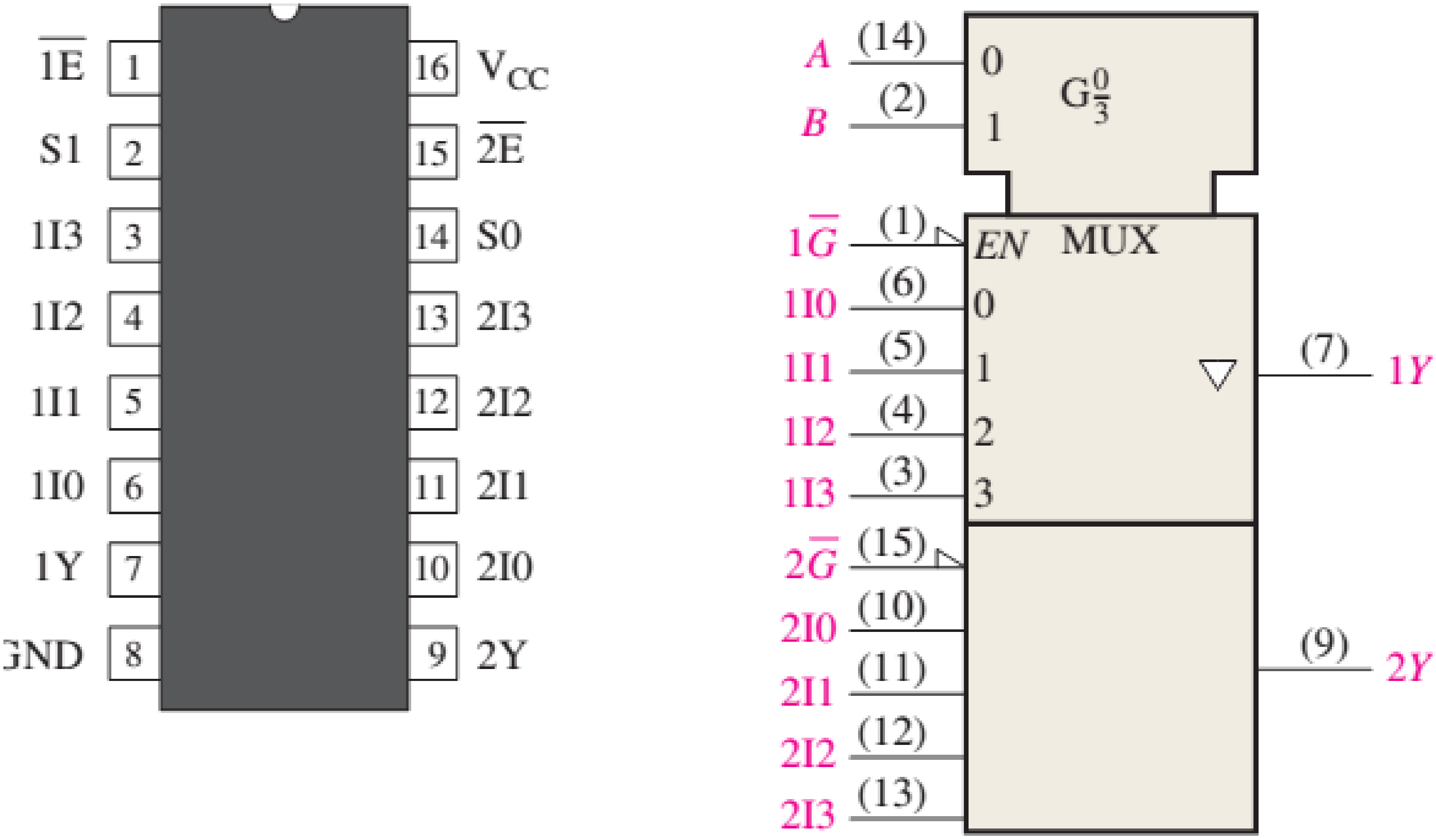


E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Function table

Quadruple two-to-one-line multiplexer

The 74HC153 is a dual four-input data selector/multiplexer.



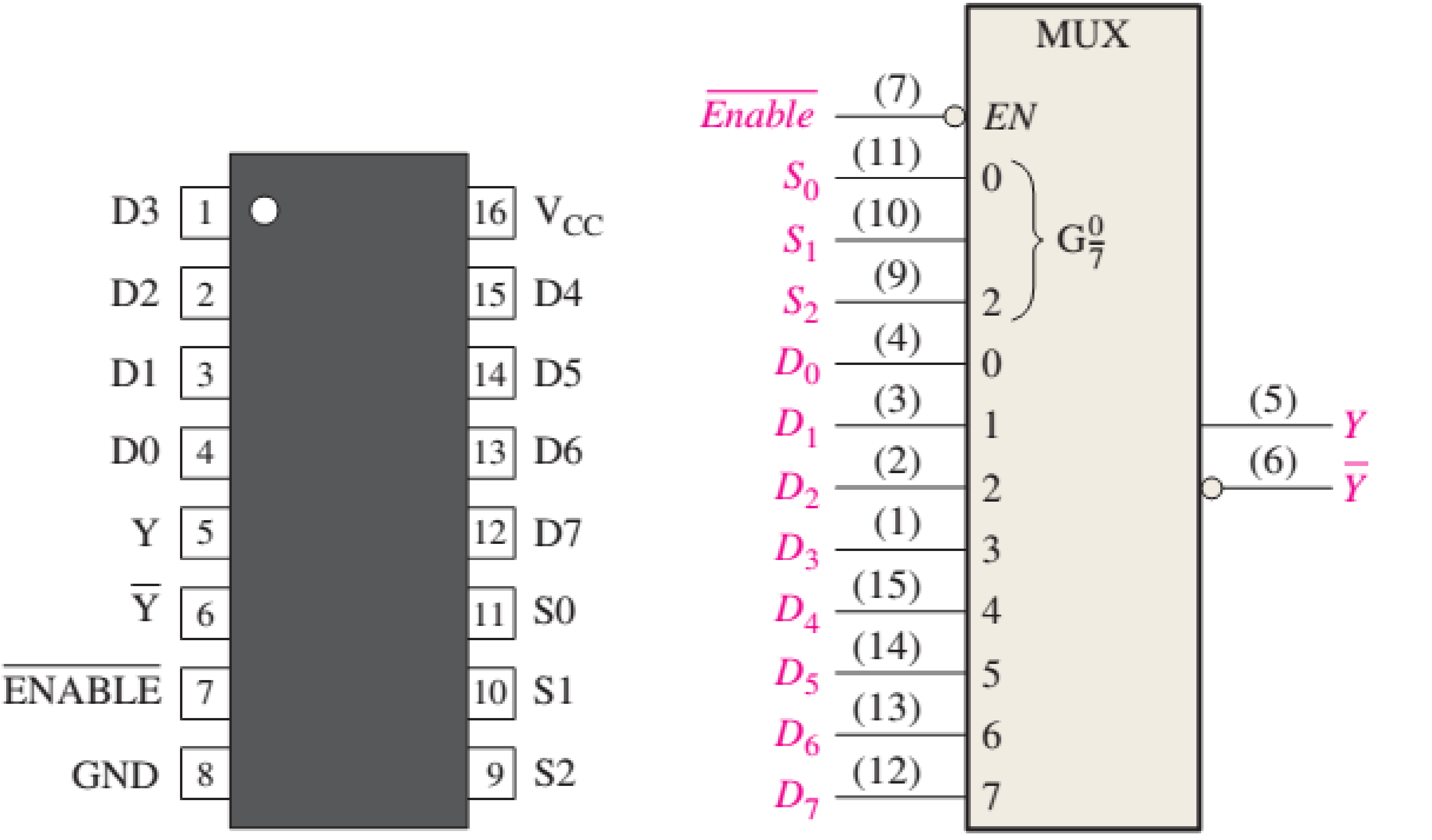
(a) Pin diagram

(b) Logic symbol

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

EIGHT-INPUT DATA SELECTOR/MULTIPLEXER

Fixed-Function Device The 74HC151 has eight data inputs (D_0–D_7) and, therefore, three data-select or address input lines (S_0–S_2). Three bits are required to select any one of the eight data inputs ($2^3 = 8$). A LOW on the $\overline{Enable}$ input allows the selected input data to pass through to the output. Notice that the data output and its complement are both available. The pin diagram is shown in Figure and the ANSI/IEEE logic symbol



(a) Pin diagram

(b) Logic symbol

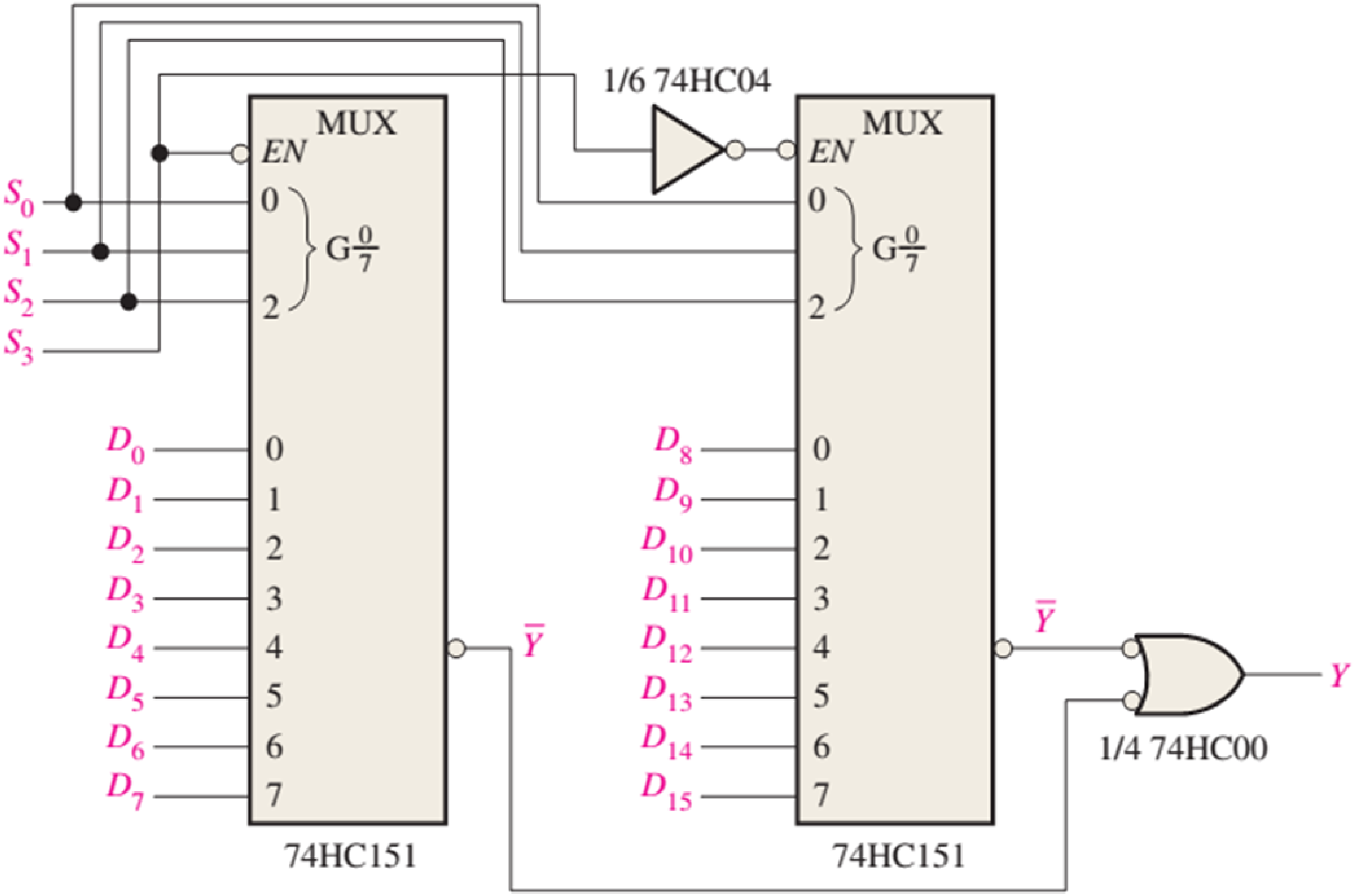
EXAMPLE

Use 74HC151s and any other logic necessary to multiplex 16 data lines onto a single data-output line.

Solution

An expansion of two 74HC151s is shown in Figure 6–48. Four bits are required to select one of 16 data inputs ($2^4 = 16$). In this application the $\overline{Enable}$ input is used as the most significant data-select bit. When the MSB in the data-select code is LOW, the left 74HC151 is enabled, and one of the data inputs (D_0 through D_7) is selected by the other three data-select bits. When the data-select MSB is HIGH, the right 74HC151 is enabled, and one of the data inputs (D_8 through D_{15}) is selected. The selected input data are then passed through to the negative-OR gate and onto the single output line.

A 16-input multiplexer.



Implementing a Boolean function with a multiplexer

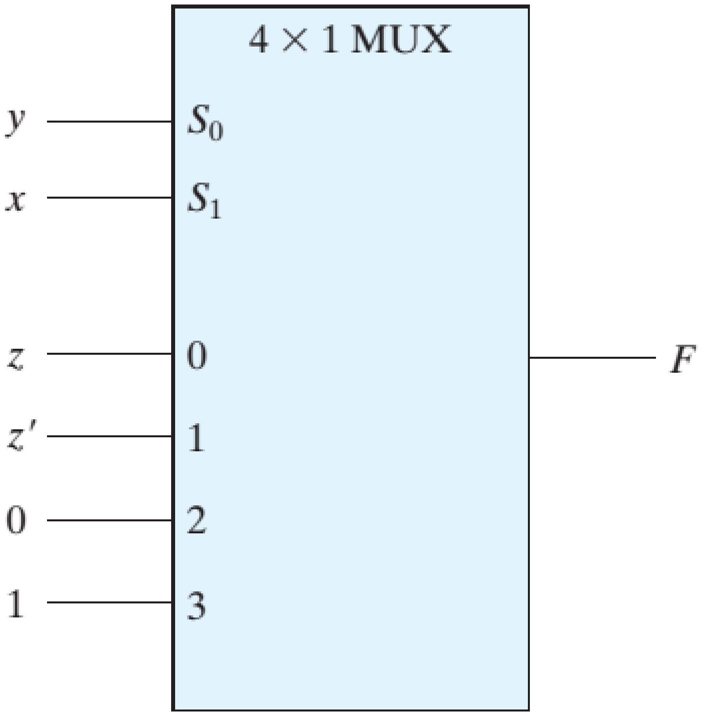
consider the Boolean function

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

This function of three variables can be implemented with a four-to-one-line multiplexer

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



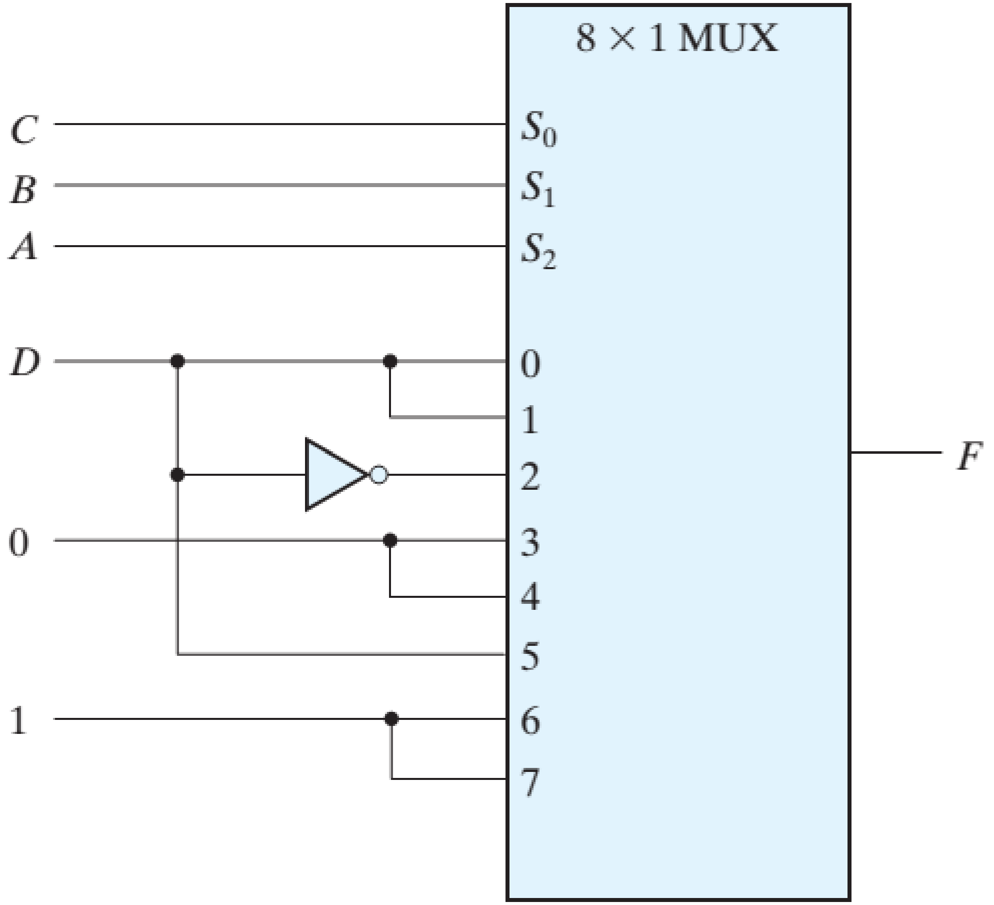
(b) Multiplexer implementation

Implementing a four-input function with a multiplexer

a second example, consider the implementation of the Boolean function

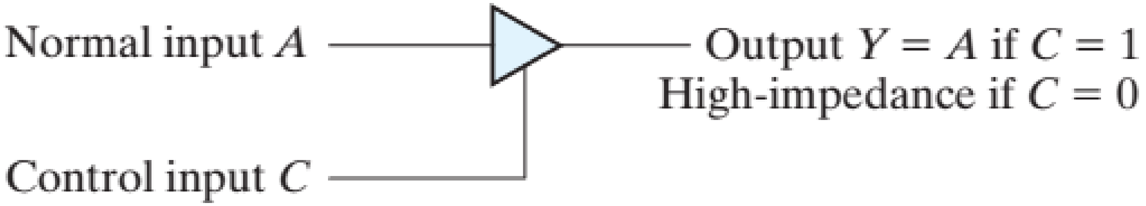
$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$$

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



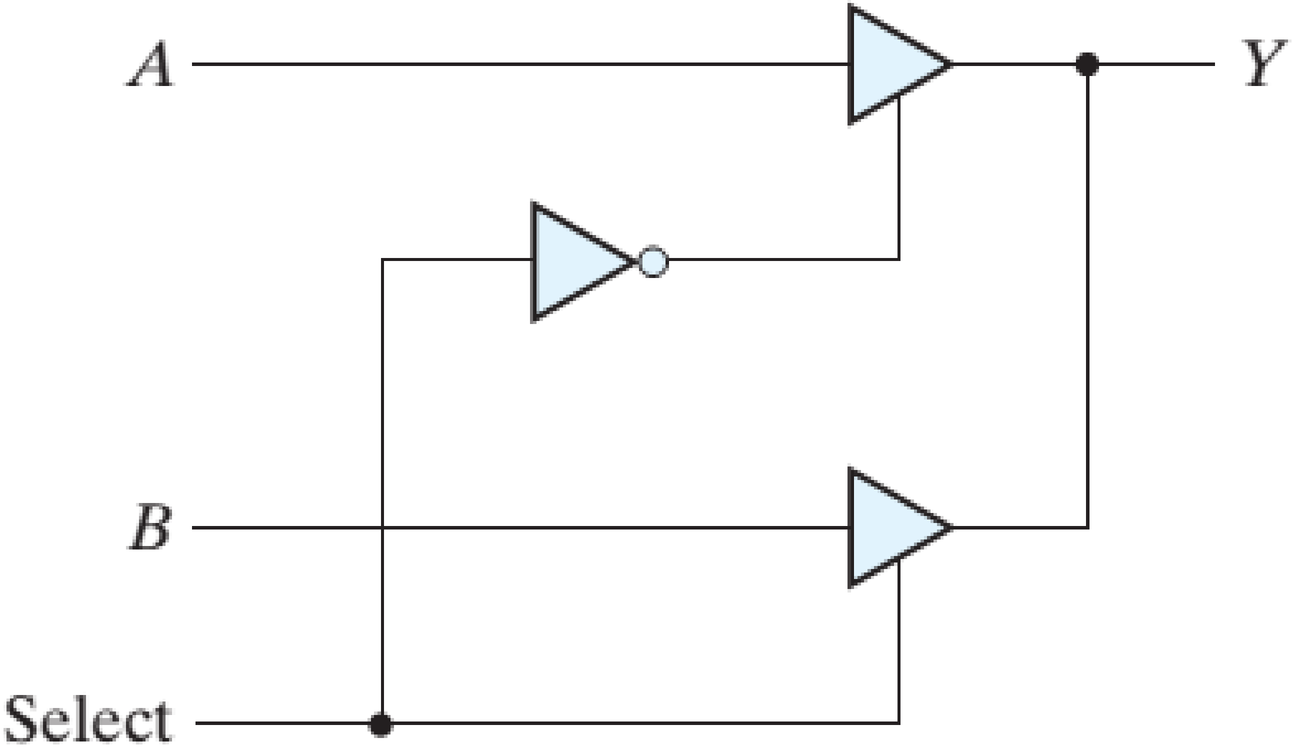
Three-State Gates

A multiplexer can be constructed with three-state gates—digital circuits that exhibit three states. Two of the states are signals equivalent to logic 1 and logic 0 as in a conventional gate. The third state is a *high-impedance* state in which (1) the logic behaves like an open circuit, which means that the output appears to be disconnected, (2) the circuit has no logic significance, and (3) the circuit connected to the output of the three-state gate is not affected by the inputs to the gate. Three-state gates may perform any conventional logic, such as AND or NAND. However, the one most commonly used is the buffer gate.



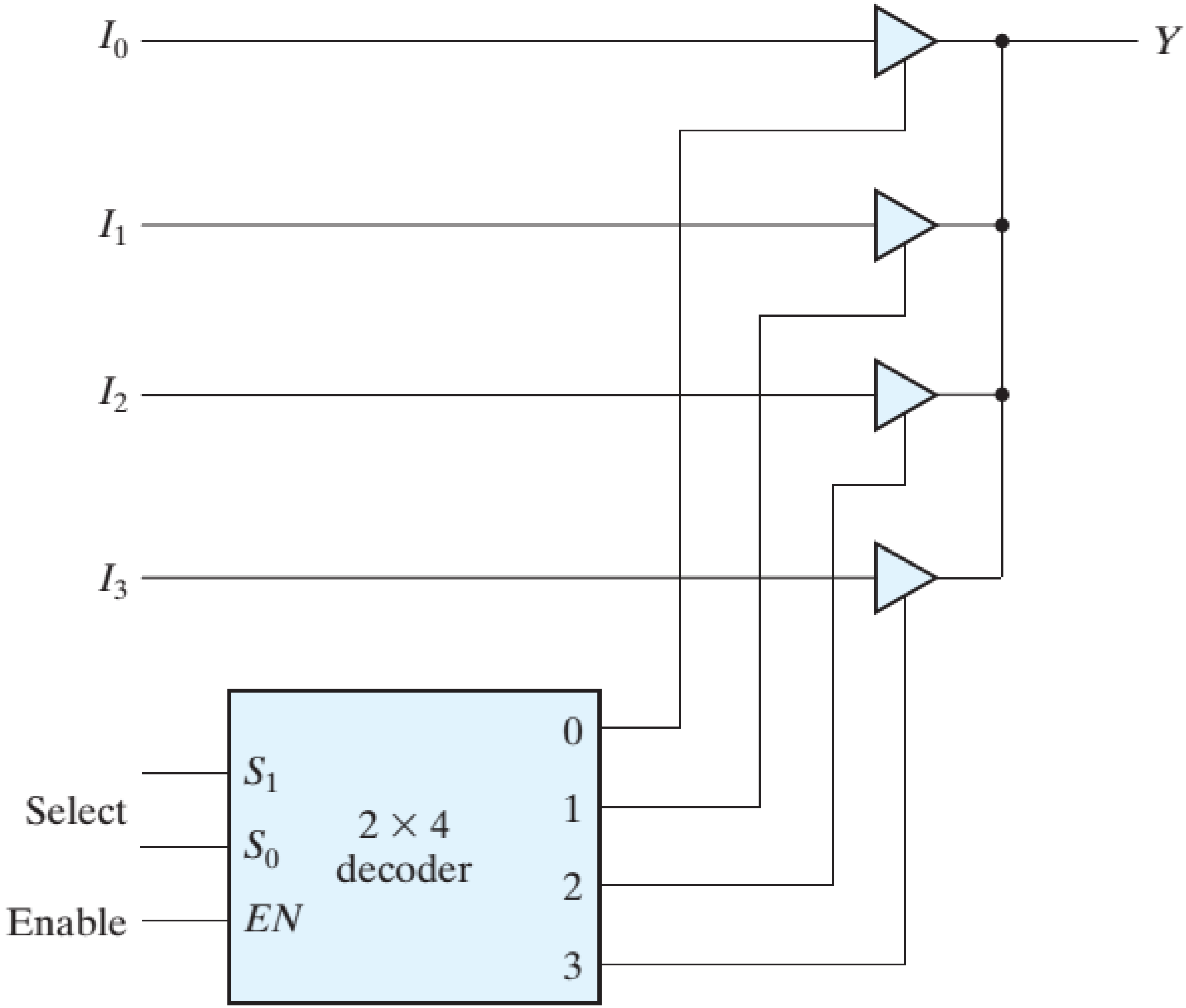
Graphic symbol for a three-state buffer

Multiplexers with three-state gates



(a) 2-to-1-line mux

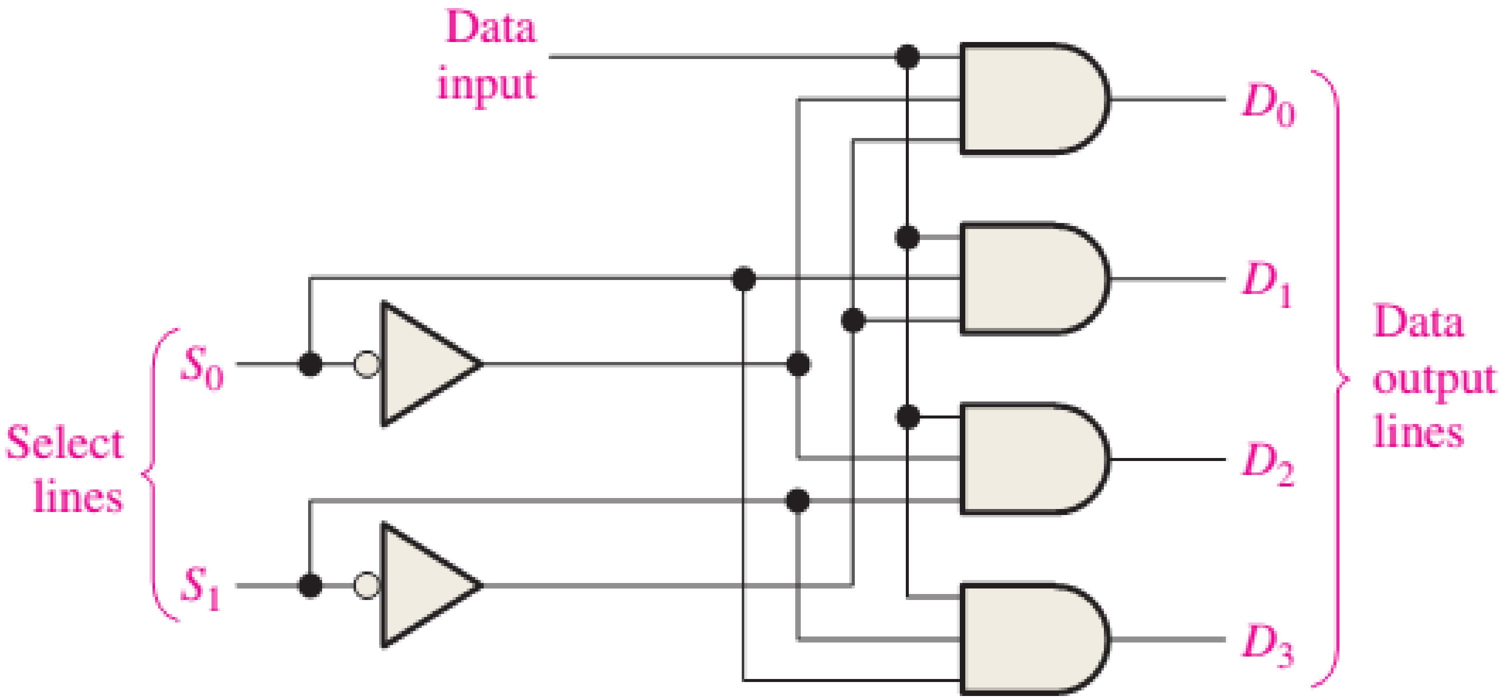
Multiplexers with three-state gates



(b) 4-to-1-line mux

Demultiplexers

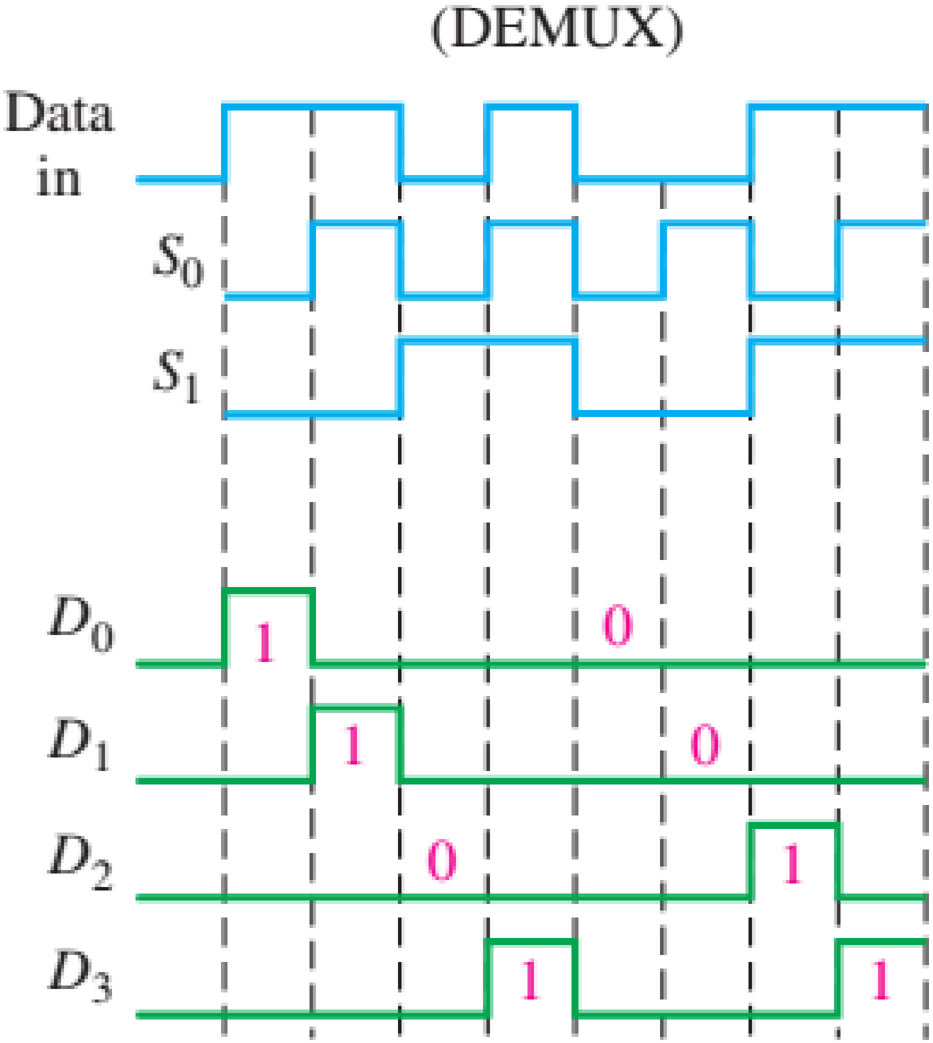
A **demultiplexer (DEMUX)** basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. As you will learn, decoders can also be used as demultiplexers.

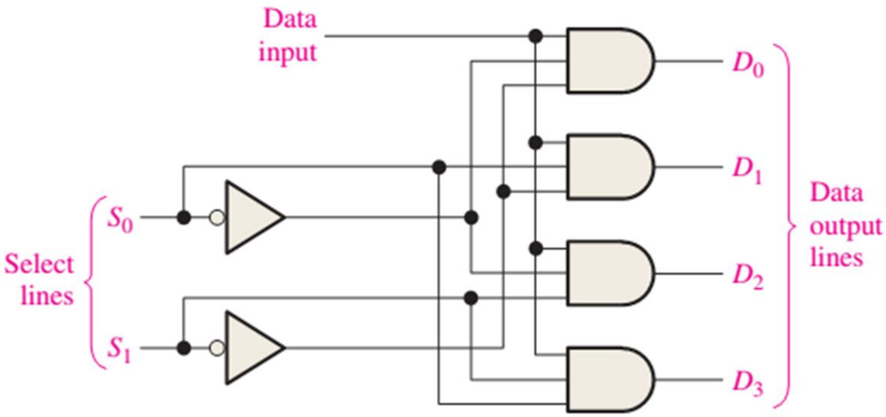


A 1-line-to-4-line demultiplexer.

In a demultiplexer, data are switched from one line to several lines.

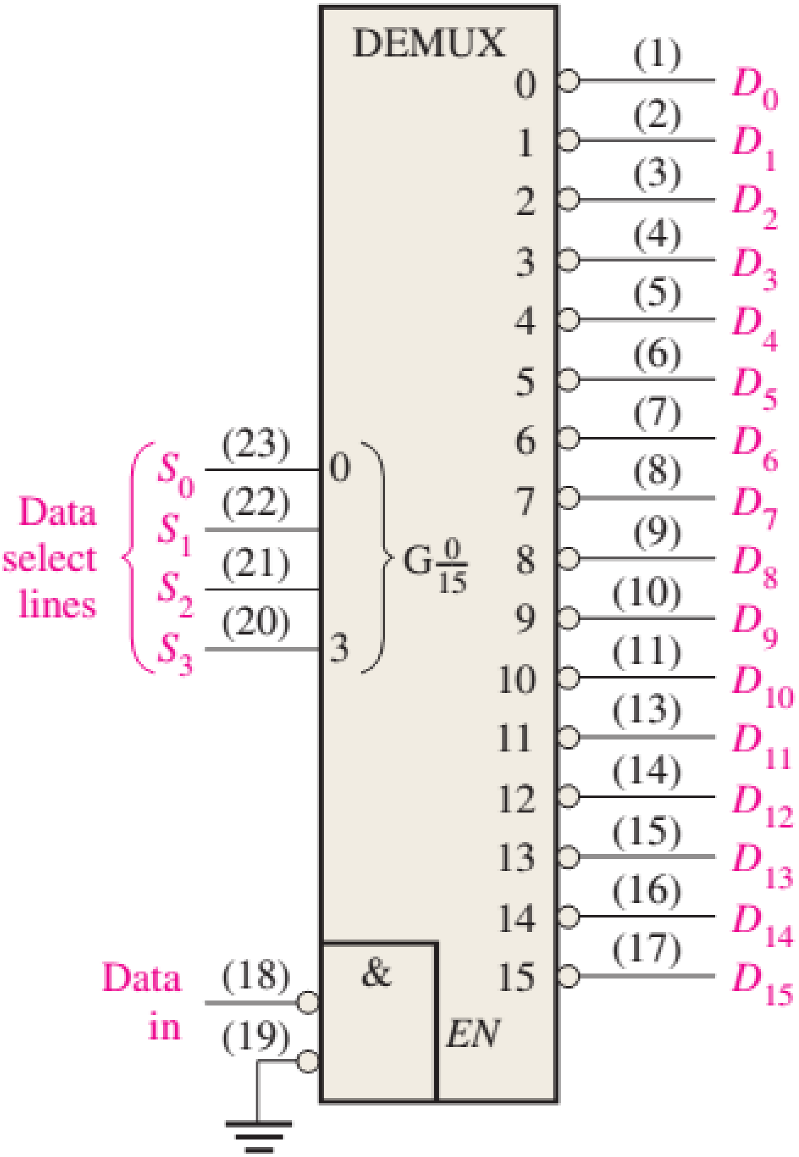
Demultiplexers





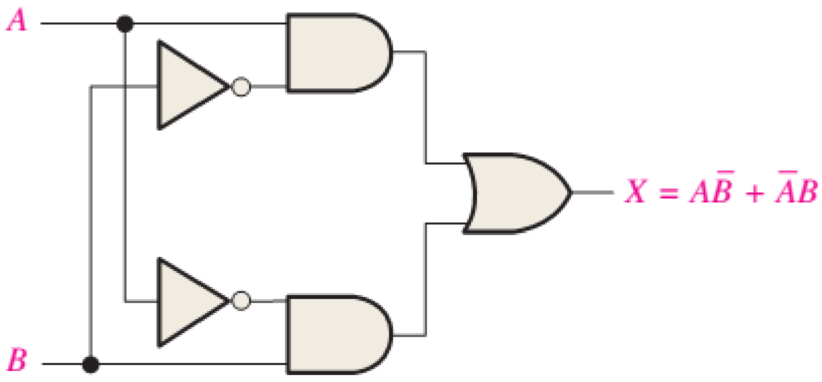
A 1-line-to-4-line demultiplexer.

4-Line-to-16-Line Decoder as a Demultiplexer

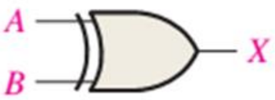


The decoder used as a demultiplexer.

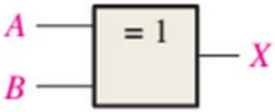
Exclusive-OR Logic



(a) Logic diagram



(b) ANSI distinctive shape symbol



(c) ANSI rectangular outline symbol

$$X = A \oplus B$$

$$X = A\overline{B} + \overline{A}B$$

Truth table for an exclusive-OR.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$x \oplus 0 = x$$

$$x \oplus 1 = x'$$

$$x \oplus x = 0$$

$$x \oplus x' = 1$$

$$x \oplus y' = x' \oplus y = (x \oplus y)'$$

$$A \oplus B = B \oplus A$$

$$X \oplus 0 = X$$

$$X \oplus 1 = \overline{X}$$

$$X \oplus X = 0$$

$$X \oplus \overline{X} = 1$$

$$\overline{X} \oplus \overline{X} = 0$$

$$\overline{X} \oplus \overline{X} = X \oplus X$$

$$A \oplus AB = A\overline{B}$$

$$A \oplus A\overline{B} = AB$$

$$A \oplus \overline{A}B = A + B$$

$$(A \oplus B).(A \oplus C) = \overline{A}BC + A\overline{B}\,\overline{C}$$

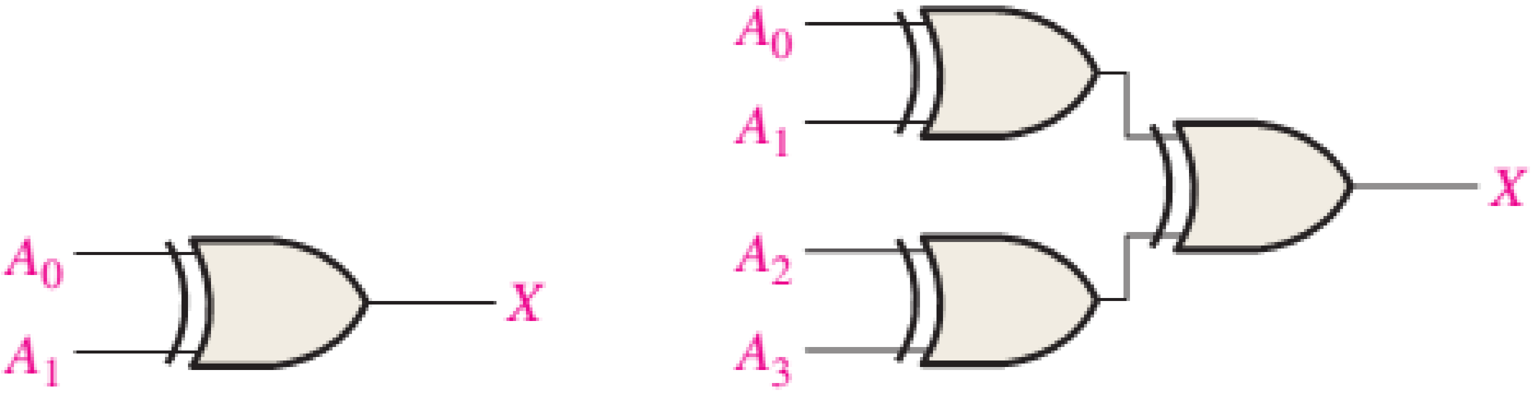
$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

Parity Generators/Checkers

A parity bit indicates if the number of 1s in a code is even or odd for the purpose of error detection.

The sum (disregarding carries) of an even number of 1s is always 0, and the sum of an odd number of 1s is always 1.

Therefore, to determine if a given code has **even parity** or **odd parity,** all the bits in that code are summed. As you know, the modulo-2 sum of two bits can be generated by an exclusive-OR gate, as shown in Figure 6–55(a); the modulo-2 sum of four bits can be formed by three exclusive-OR gates connected as shown in Figure 6–55(b); and so on. When the number of 1s on the inputs is even, the output *X* is 0 (LOW). When the number of 1s is odd, the output *X* is 1 (HIGH).



(a) Summing of two bits

(b) Summing of four bits

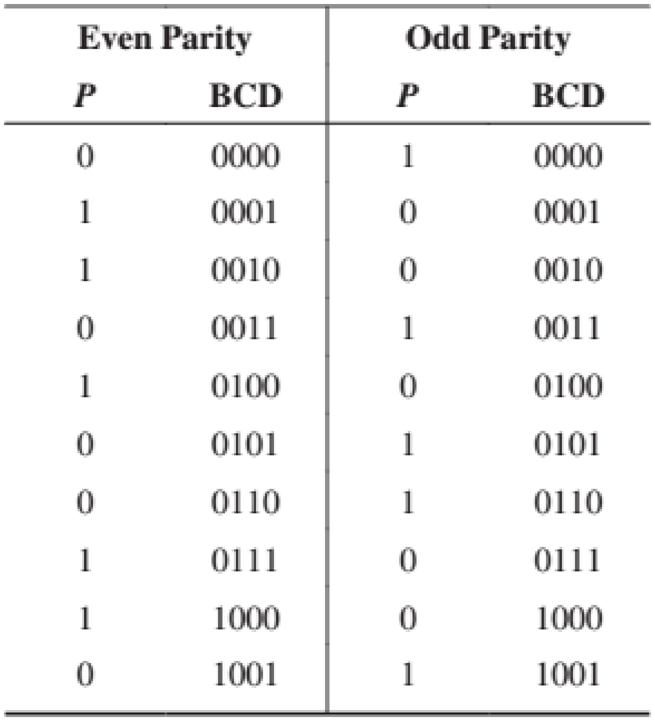
Parity Method for Error Detection

A parity bit tells if the number of 1s is odd or even.

Many systems use a parity bit as a means for bit **error detection**. Any group of bits contain either an even or an odd number of 1s. A parity bit is attached to a group of bits to make the total number of 1s in a group always even or always odd. An even parity bit makes the total number of 1s even, and an odd parity bit makes the total odd.

A given system operates with even or odd **parity**, but not both. For instance, if a system operates with even parity, a check is made on each group of bits received to make sure the total number of 1s in that group is even. If there is an odd number of 1s, an error has occurred.

The BCD code with parity bits.



Even Parity		Odd Parity	
P	**BCD**	***P***	**BCD**
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

Even Parity =0 ====> No. Of Ones is EVEN

Even Parity =1 ====> No. Of Ones is Odd

Odd Parity =1 ====> No. Of Ones is EVEN

Odd Parity =0 ====> No. Of Ones is Odd

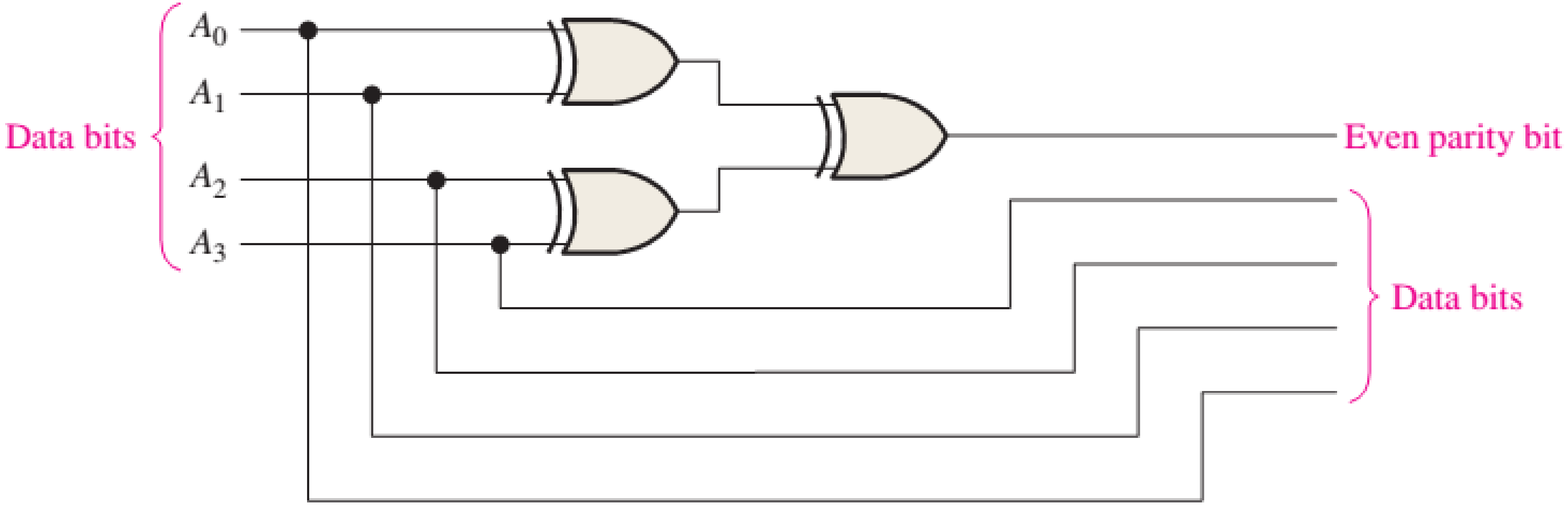
The BCD code with parity bits.

Even Parity		Odd Parity	
P	**BCD**	***P***	**BCD**
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

The parity bit can be attached to the code at either the beginning or the end, depending on system design. Notice that the total number of 1s, including the parity bit, is always even for even parity and always odd for odd parity.

EXAMPLE

Use exclusive-OR gates to implement an even-parity code generator for an original 4-bit code.



Even-parity generator.

Even Parity =0 ====> No. Of Ones is EVEN

Even Parity =1 ====> No. Of Ones is Odd

Parity Generation and Checking

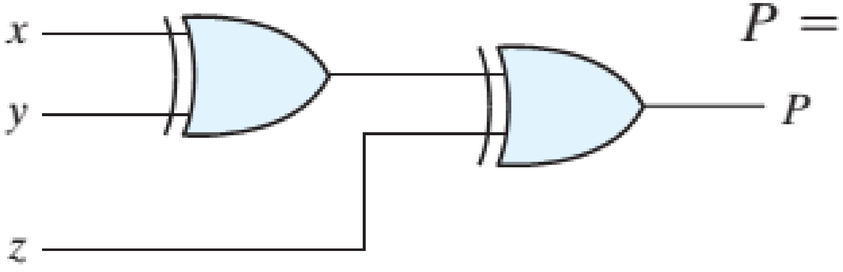
Even Parity =0 ====> No. Of Ones is EVEN

Even Parity =1 ====> No. Of Ones is Odd

Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	y	z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

. For even parity, the bit P must be generated to make the total number of 1's (including P) even.

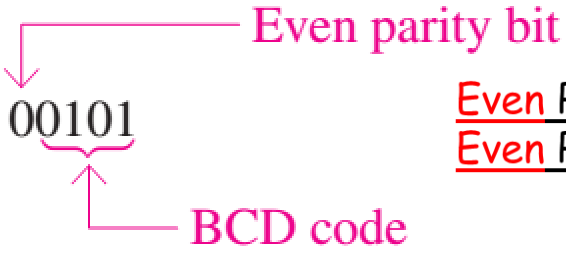


$$P = x \oplus y \oplus z$$

(a) 3-bit even parity generator

Detecting an Error

A parity bit provides for the detection of a single bit error (or any odd number of errors, which is very unlikely) but cannot check for two errors in one group. For instance, let's assume that we wish to transmit the BCD code 0101. (Parity can be used with any number of bits; we are using four for illustration.) The total code transmitted, including the even parity bit, is



Even parity bit

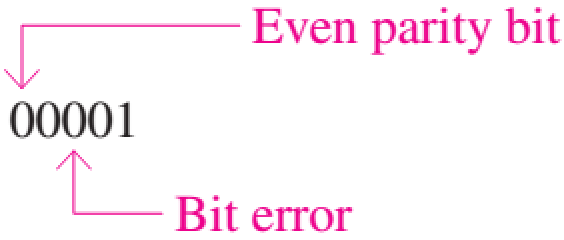
00101

BCD code

Even Parity =0 ====> No. Of Ones is EVEN

Even Parity =1 ====> No. Of Ones is Odd

Now let's assume that an error occurs in the third bit from the left (the 1 becomes a 0).



Even parity bit

00001

Bit error

When this code is received, the parity check circuitry determines that there is only a single 1 (odd number), when there should be an even number of 1s. Because an even number of 1s does not appear in the code when it is received, an error is indicated.

An odd parity bit also provides in a similar manner for the detection of a single error in a given group of bits.

Even Parity =0 ====> No. Of Ones is EVEN

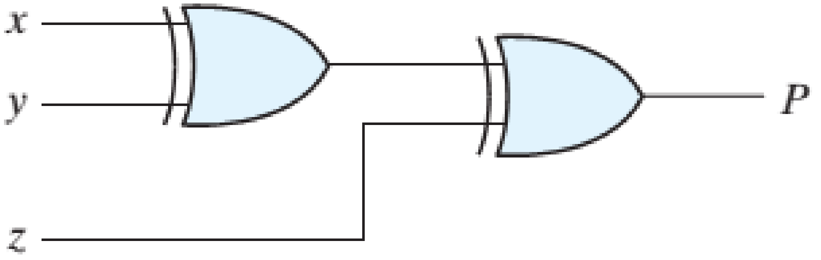
Even Parity =1 ====> No. Of Ones is Odd

Even-Parity-Checker Truth Table

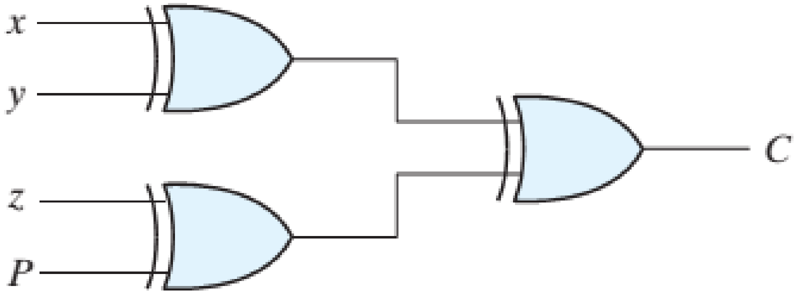
C, will be equal to 1 if an error occurs

Four Bits Received				Parity Error Check
x	***y***	***z***	***P***	***C***
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$$C = x \oplus y \oplus z \oplus P$$



(a) 3-bit even parity generator



(b) 4-bit even parity checker

EXAMPLE

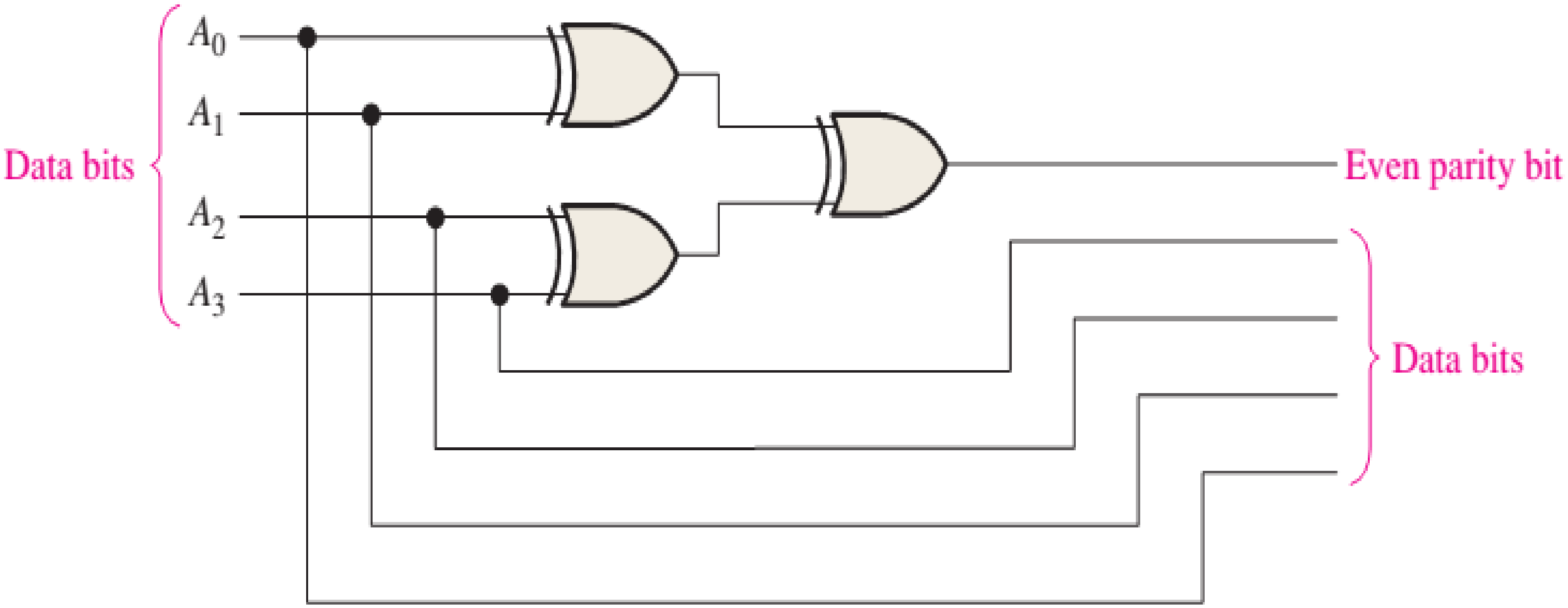
Use exlusive-OR gates to implement an even-parity checker for the 5-bit code generated by the circuit in Example

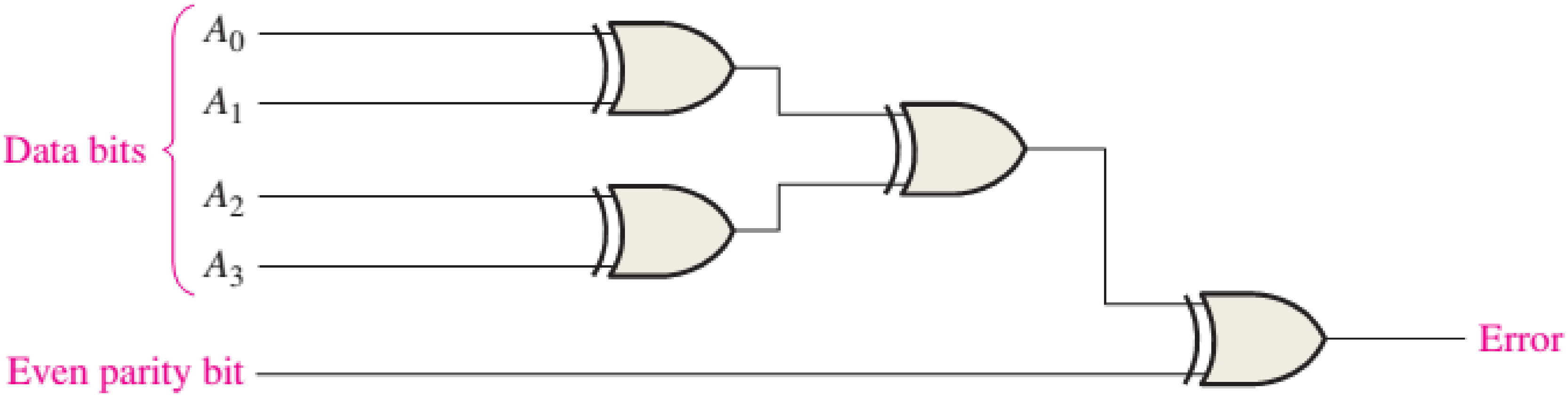
Solution

The circuit in Figure produces a 1 output when there is an error in the five-bit code and a 0 when there is no error.

Even Parity =0 ====> No. Of Ones is EVEN

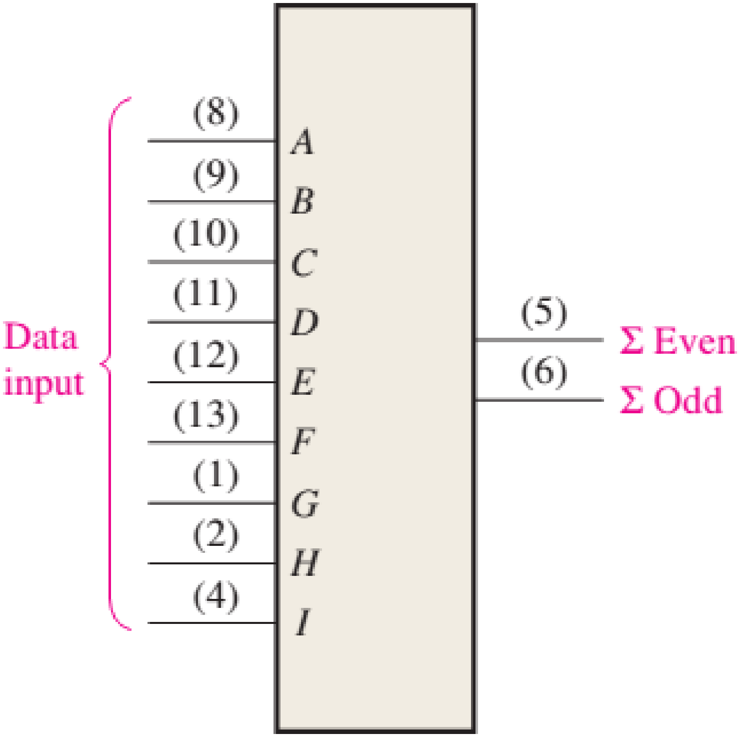
Even Parity =1 ====> No. Of Ones is Odd


Data bits
A_0
A_1
A_2
A_3
Even parity bit
Data bits


Data bits
A_0
A_1
A_2
A_3
Even parity bit
Error

9-BIT PARITY GENERATOR/CHECKER

Fixed-Function Device The logic symbol and function table for a 74HC280 are shown in Figure 6–56. This particular device can be used to check for odd or even parity on a 9-bit code (eight data bits and one parity bit), or it can be used to generate a parity bit for a binary code with up to nine bits. The inputs are *A* through *I*; when there is an even number of 1s on the inputs, the Σ Even output is HIGH and the Σ Odd output is LOW.



(a) Traditional logic symbol

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

(b) Function table

The 74HC280 9-bit parity generator/checker.

Parity Checker When this device is used as an even parity checker, the number of input bits should always be even; and when a parity error occurs, the Σ Even output goes LOW and the Σ Odd output goes HIGH. When it is used as an odd parity checker, the number of input bits should always be odd; and when a parity error occurs, the Σ Odd output goes LOW and the Σ Even output goes HIGH.

Parity Generator If this device is used as an even parity generator, the parity bit is taken at the Σ Odd output because this output is a 0 if there is an even number of input bits and it is a 1 if there is an odd number. When used as an odd parity generator, the parity bit is taken at the Σ Even output because it is a 0 when the number of inputs bits is odd.

Thank you!