Chapter 4. Combinational Logic

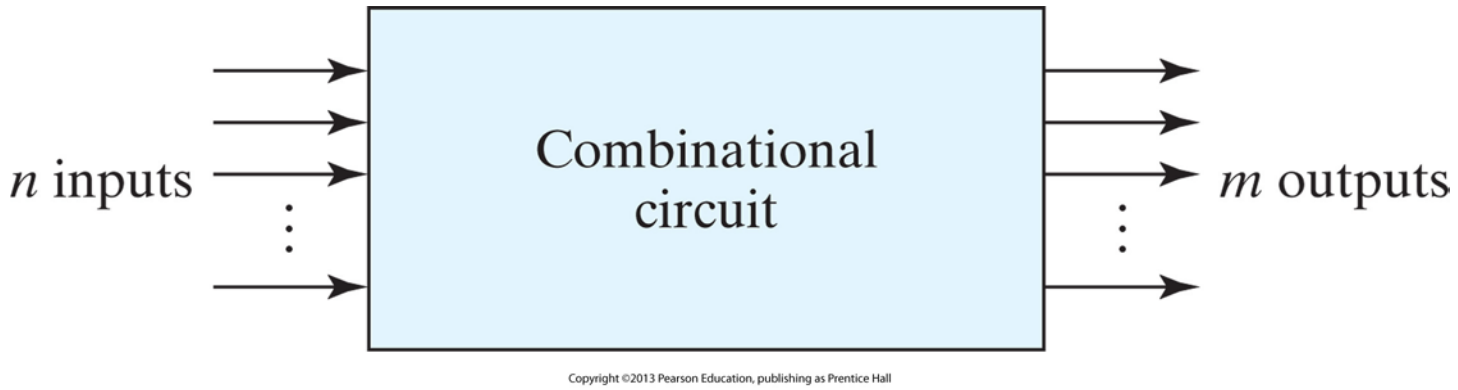
Tong In Oh

4.1 Introduction

- Combinational logic:
  - Logic gates
  - Output determined from only the present combination of inputs
  - Specified by a set of Boolean functions
- Sequential logic:
  - Employ storage elements in addition to logic gates
  - Output determined by present inputs and the state of the storage elements
  - Specified by a time sequence of inputs and internal states

4.2 Combinational Circuits

- Interconnection of logic gates
- React to the input signals and produce the output signal
- Transforming binary information
- For n input variables, m Boolean functions (one for each output variable)



Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.1 Block diagram of combinational circuit

Goals

- To formulate systematic analysis
- Design procedures for combinational circuits
- Provide a useful catalog of elementary functions

- Two tasks
  - Analyze the behavior of a given logic circuit
  - Synthesize a circuit for a given behavior

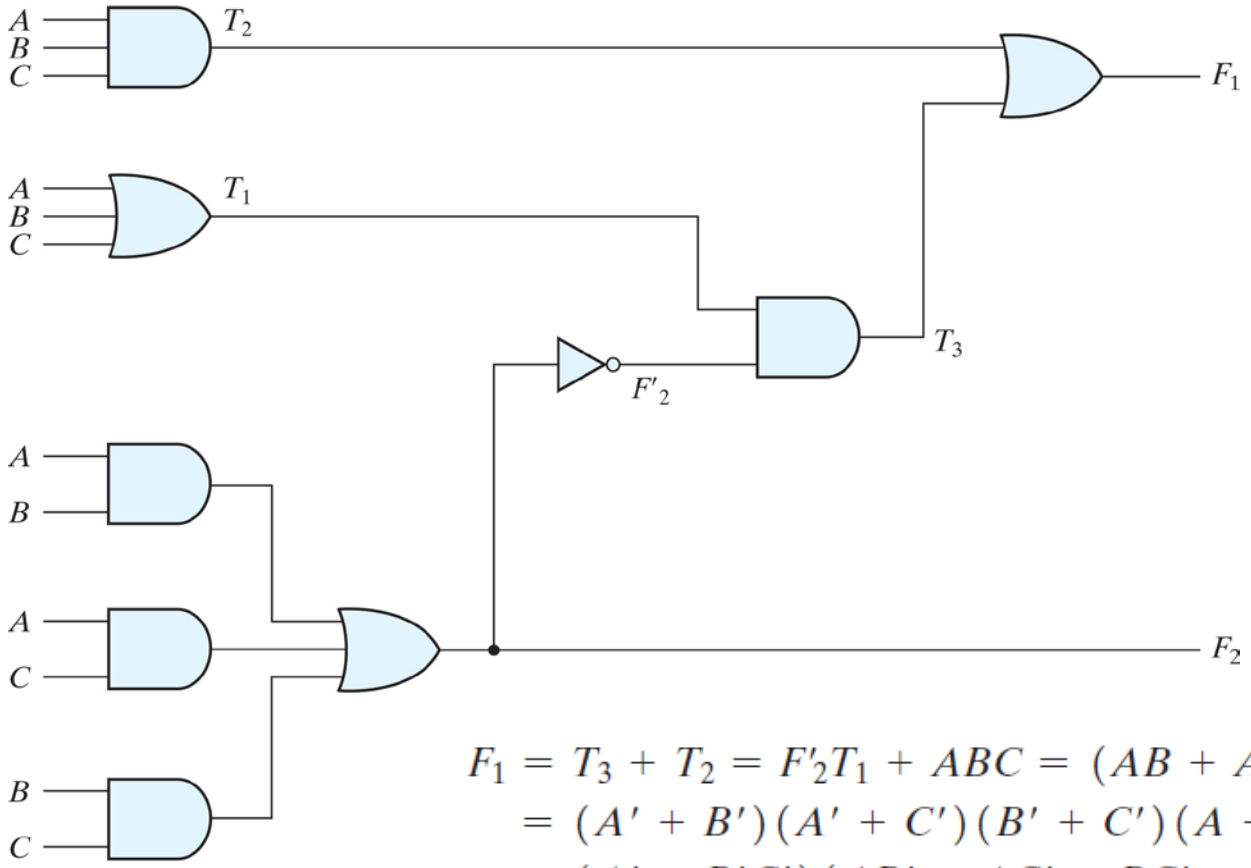
- The most important standard combinational circuits
  - Adders, subtractors, comparators, decoders, encoders, and multiplexers

4.3 Analysis Procedure

- Determine the function implemented in the circuit
- Boolean functions, a truth table, explanation of the circuit operation from the given logic diagram

1. Combinational and not sequential ?
   - No feedback paths or memory elements
2. To obtain the output Boolean functions
   1. Label all gate outputs that are a function of input variables with arbitrary symbols but with meaningful names. Determine the Boolean functions for each gate output.
   2. Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
   3. Repeat the process outlined in step 2 until the outputs of the circuit are obtained.
   4. By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.

- Determine the function from a given logic diagram to a set of Boolean functions or a truth table



$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$
$$T_3 = F'_2 T_1$$
$$F_1 = T_3 + T_2$$

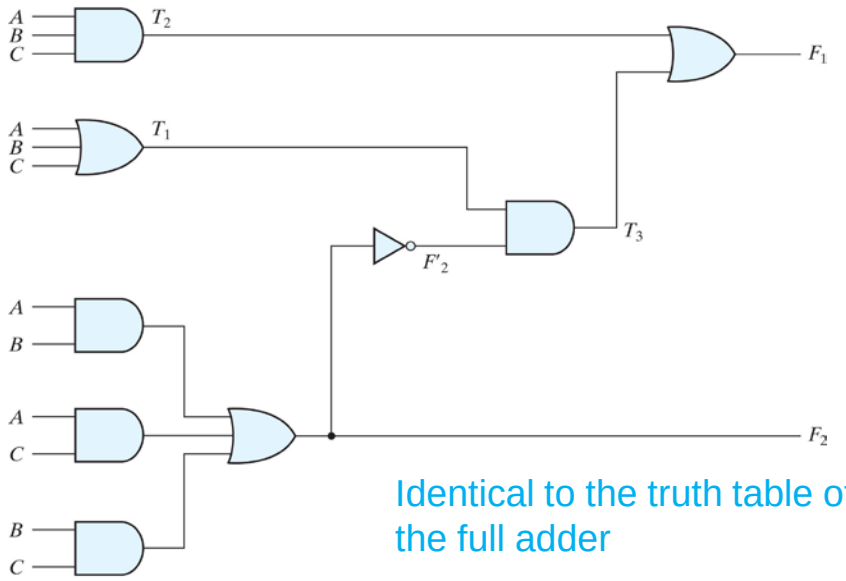
$$\begin{aligned} F_1 = T_3 + T_2 &= F'_2 T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC \end{aligned}$$

FIGURE 4.2 Logic diagram for analysis example

Truth Table

$$F_2 = AB + AC + BC \qquad T_3 = F'_2 T_1$$
$$T_1 = A + B + C \qquad F_1 = T_3 + T_2$$
$$T_2 = ABC$$

- To obtain truth table
  1. Determine the number of input variables in the circuit. List the binary numbers from 0 to (2^n - 1) in a table.
  2. Label the outputs of selected gates with arbitrary symbols.
  3. Obtain the truth table for the outputs of those gates which are a function of the input variables only.
  4. Proceed to obtain the truth table for the outputs of those gates which are a function of previously defined values until the columns for all outputs are determined



Identical to the truth table of the full adder

Table 4.1
Truth Table for the Logic Diagram of Fig. 4.2

A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Copyright ©2012 Pearson Education, publishing as Prentice Hall

4.4 Design Procedure

- Start from the specification of the design objective
- Culminate in a logic circuit diagram or a set of Boolean functions
  1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.
  2. Derive the truth table that defines the required relationship between inputs and outputs.
  3. Obtain the simplified Boolean functions for each output as a function of the input variables.
  4. Draw the logic diagram and verify the correctness of the design (manually or by simulation).
- Interpret verbal specifications correctly in the truth table
- Constraints
  - Number of gates, number of inputs to a gate, propagation time of the signal through the gates, number of interconnections, limitations of the driving capability of each gate, various other criteria

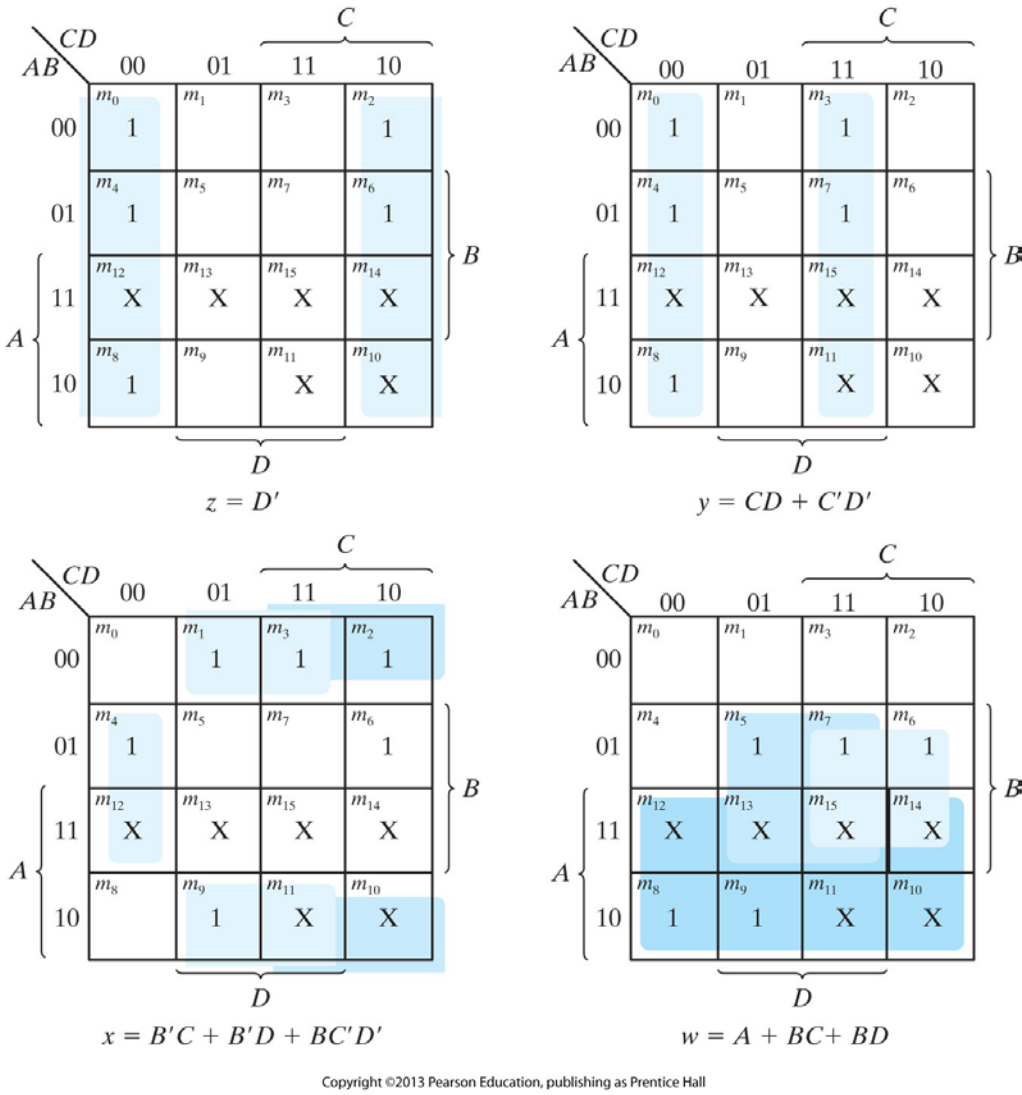
Code Conversion Example

- Code converter: transformation
- Ex. Convert BCD to the excess-3 code for decimal digits
- Four inputs, four outputs
- 10 combinations except of 6 don't care combinations

Table 4.2
Truth Table for Code Conversion Example

Input BCD				Output Excess-3 Code			
A	**B**	**C**	**D**	**w**	**x**	**y**	**z**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Copyright ©2012 Pearson Education, publishing as Prentice Hall



Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.3 Maps for BCD-to-excess-3 code converter

$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$

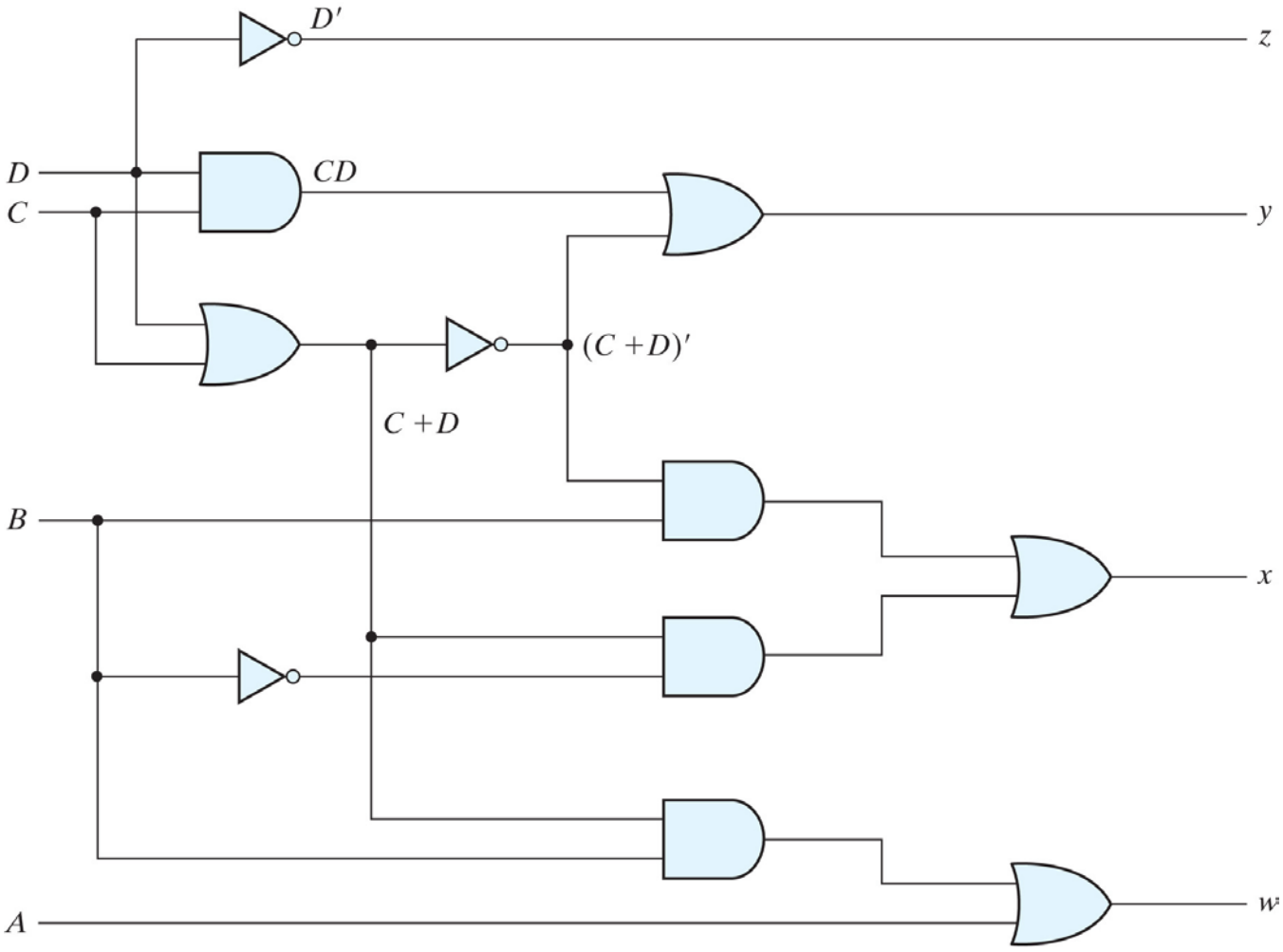


FIGURE 4.4 Logic diagram for BCD-to-excess-3 code converter

Copyright ©2013 Pearson Education, publishing as Prentice Hall

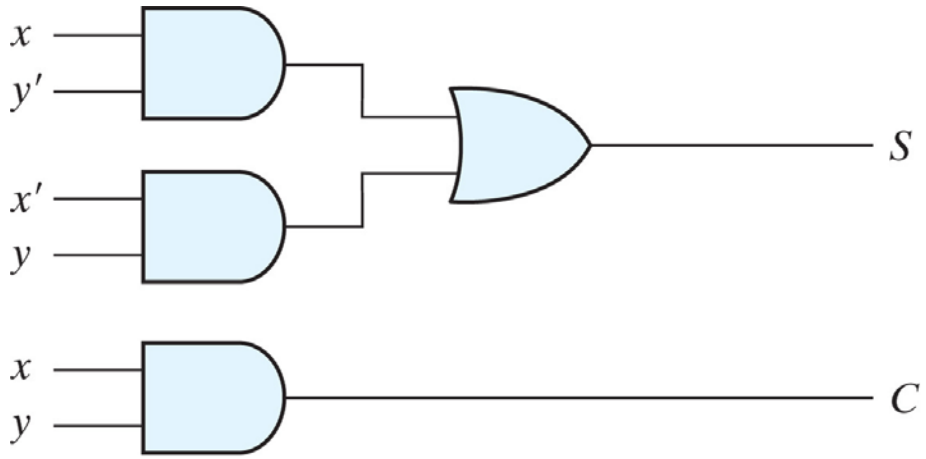
4.5 Binary Adder-Subtractor

- Digital computer:
  - Perform a variety of information-processing tasks
  - Various arithmetic operations
- Addition of two binary digits
  - 0+0=0, 0+1=1, 1+0=1, 1+1=10
- Half adder: perform the addition of two bits
- Full adder: perform the addition of three bits (two significant bits and a previous carry)
- Carry: higher significant bit of binary sum of two digits
- Binary adder-subtractor: combinational circuit performs the arithmetic operation of addition and subtraction with binary numbers
- Hierarchical design
- Half adder → full adder → binary adder → subtractor

Half Adder

- Need two binary inputs (augend, addend) and two binary outputs (sum, carry)

$$S = xy' + x'y$$
$$C = xy$$

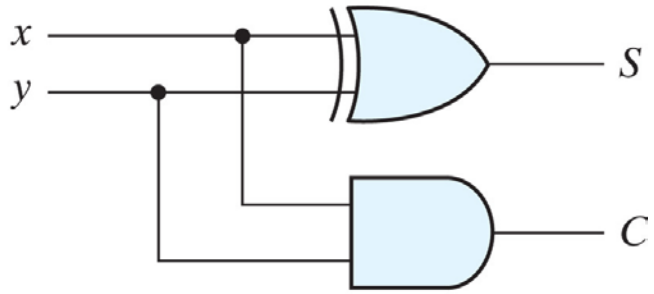


(a) $S = xy' + x'y$
$C = xy$

Table 4.3
Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Copyright ©2012 Pearson Education, publishing as Prentice Hall



(b) $S = x \oplus y$
$C = xy$

Copyright ©2013 Pearson Education, publishing as Prentice Hall

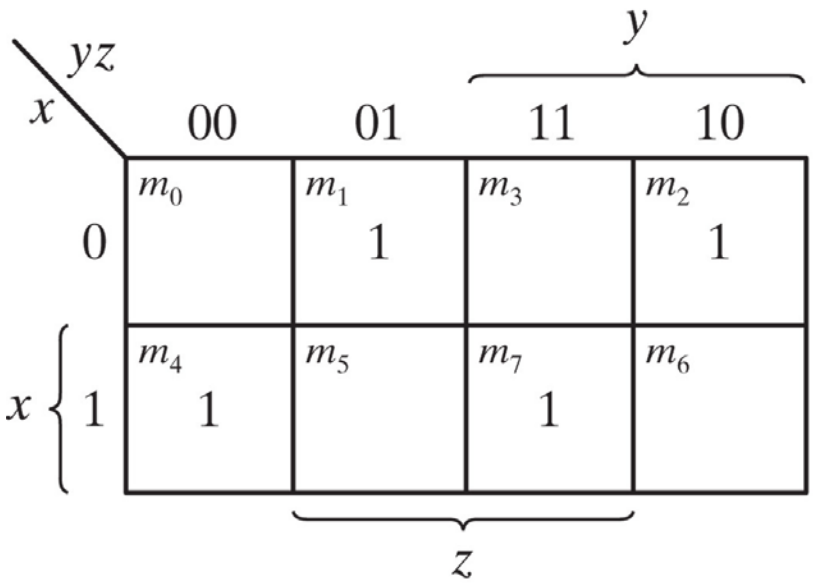
FIGURE 4.5 Implementation of half adder

Full Adder

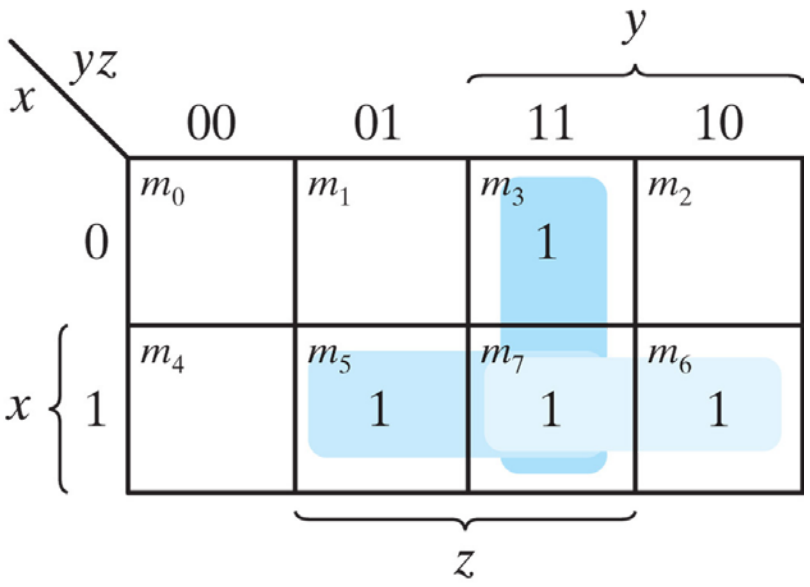
- Process of addition:
  - Bit-by-bit basis, right to left, beginning with the least significant bit
  - Consider the addition at each position and possible carry bit from the previous position
  - Three inputs and two outputs

Table 4.4
Full Adder

x	y	z	C	S
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

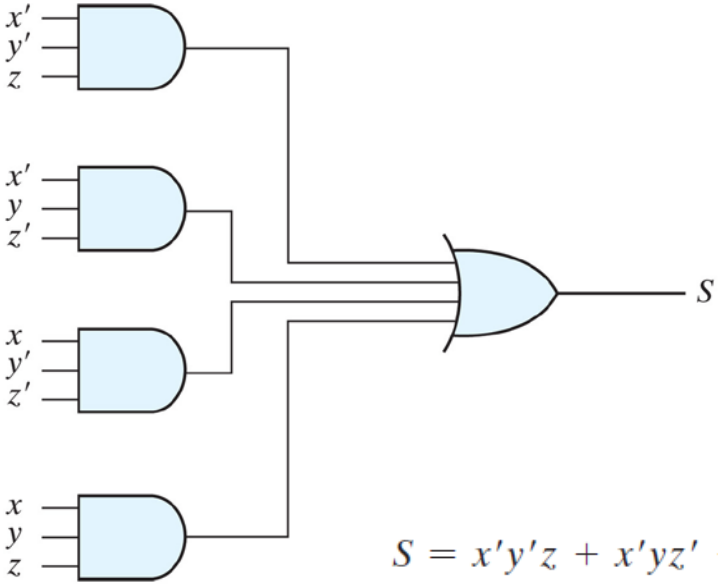


(a) $S = x'y'z + x'yz' + xy'z' + xyz$



(b) $C = xy + xz + yz$

Copyright ©2013 Pearson Education, publishing as Prentice Hall



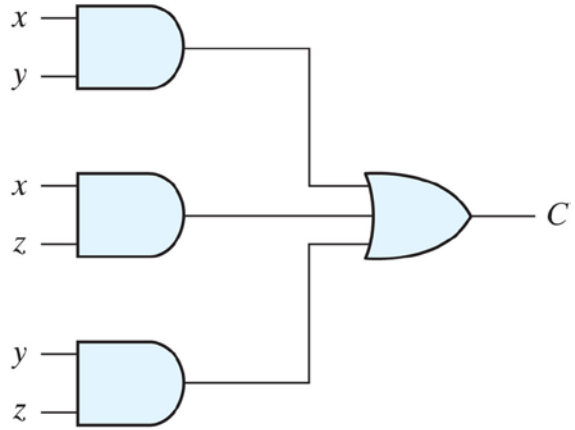
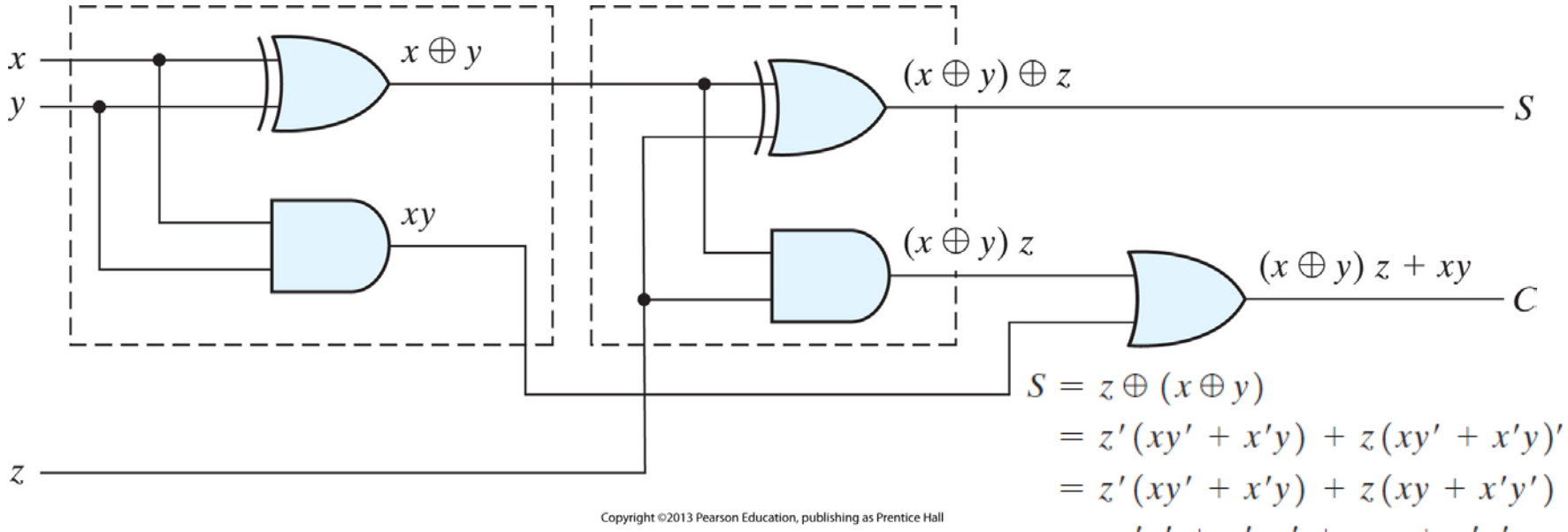


FIGURE 4.7
Implementation of full adder in sum-of-products form

$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$



$$S = z \oplus (x \oplus y)$$
$$= z'(xy' + x'y) + z(xy' + x'y)'$$
$$= z'(xy' + x'y) + z(xy + x'y')$$
$$= xy'z' + x'yz' + xyz + x'y'z$$

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.8 Implementation of full adder with two half adders and an OR gate

Binary Adder

- Produce the arithmetic sum of two binary numbers
- Connected full adders in cascade
- N-bit addition: chain of N full adders or 1 half adder and N-1 full adders

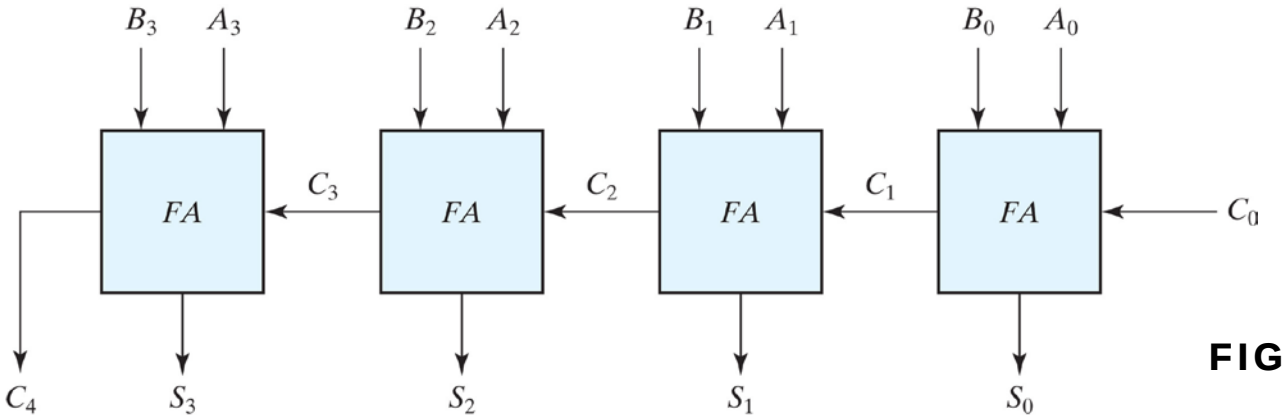


FIGURE 4.9 Four-bit adder

Subscript *i*:	**3**	**2**	**1**	**0**	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

Instead of truth table
9 inputs = $2^9 = 512$ entries,

Using iterative method of cascading a standard function

Carry Propagation

- For N-bit adder, 2n gate levels for carry to propagate from input to output
- Reducing the carry propagation delay time
  - Employ faster gates with reduced delays
  - To increase the complexity of the equipment (in a parallel adder)
    - Principle of carry lookahead logic

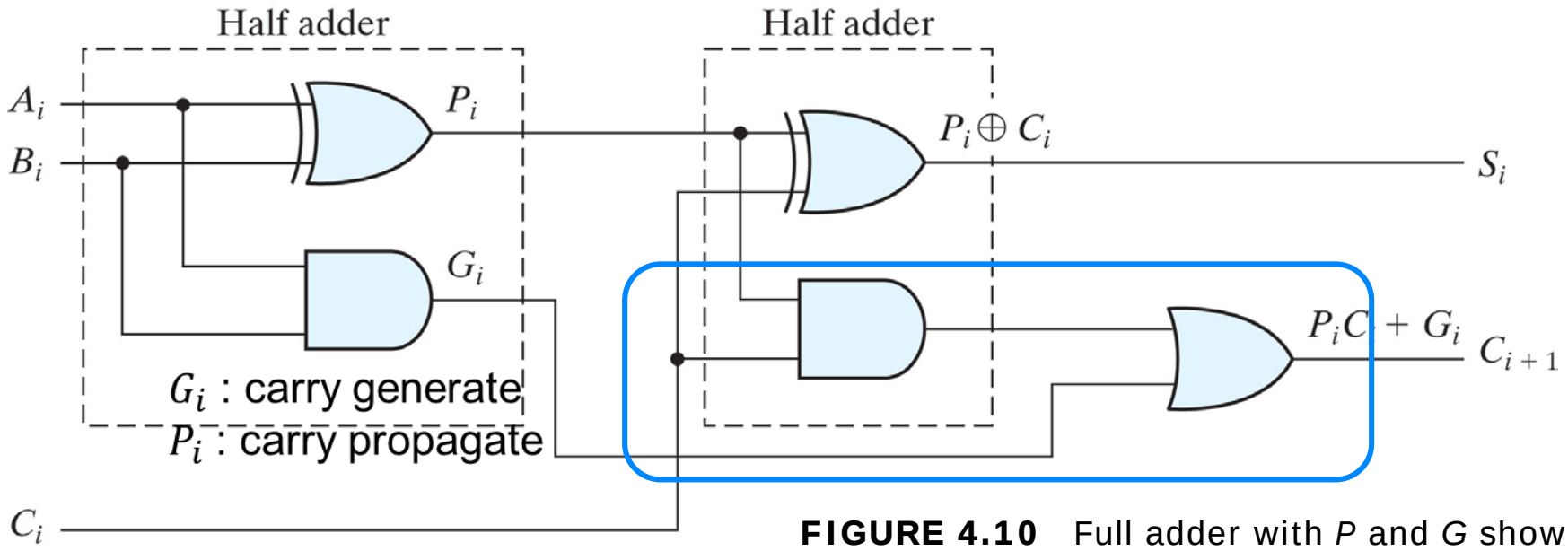


FIGURE 4.10 Full adder with *P* and *G* shown

Copyright ©2013 Pearson Education, publishing as Prentice Hall

- Speed of operation
- Expense of additional complexity

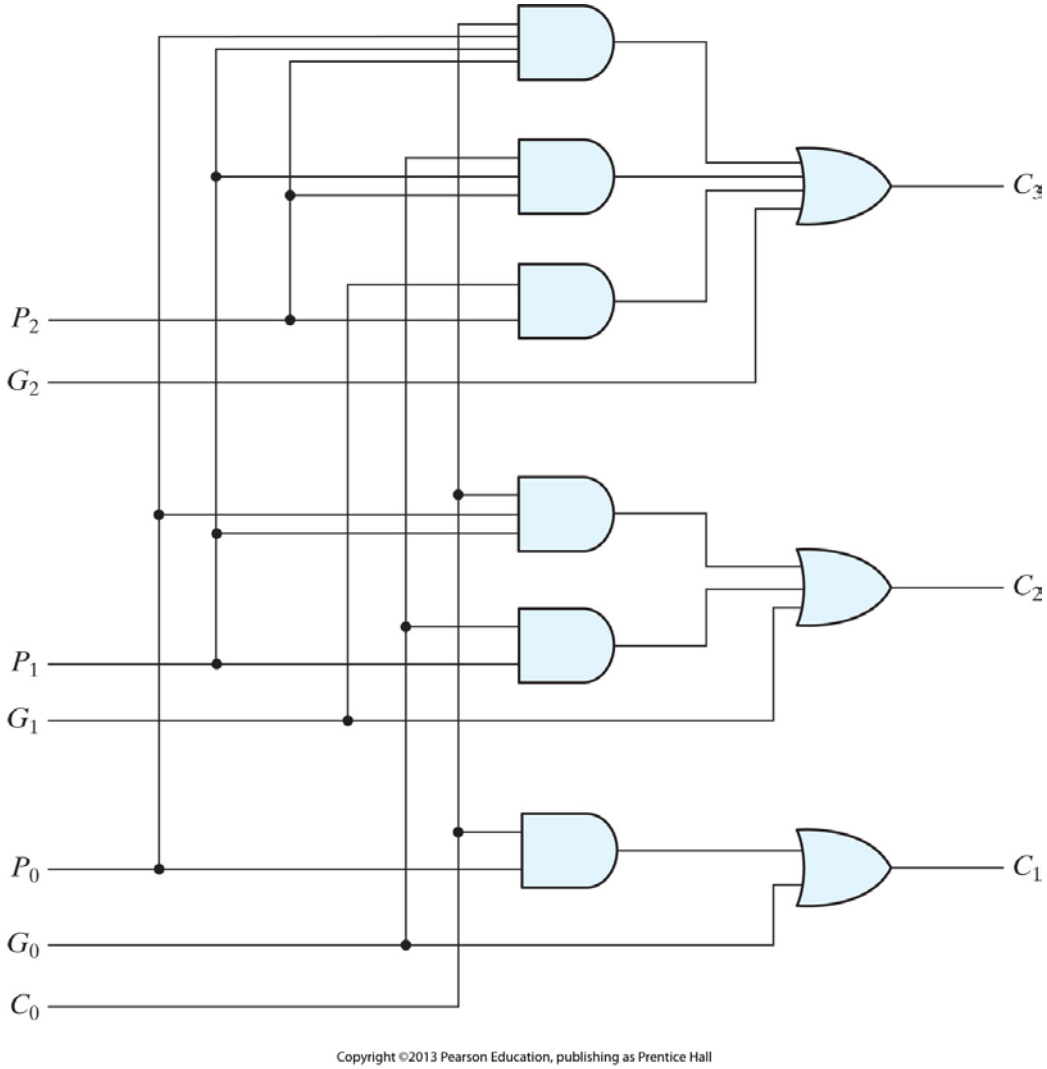


FIGURE 4.11 Logic diagram of carry lookahead generator

$$C_0 = \text{input carry}$$

$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

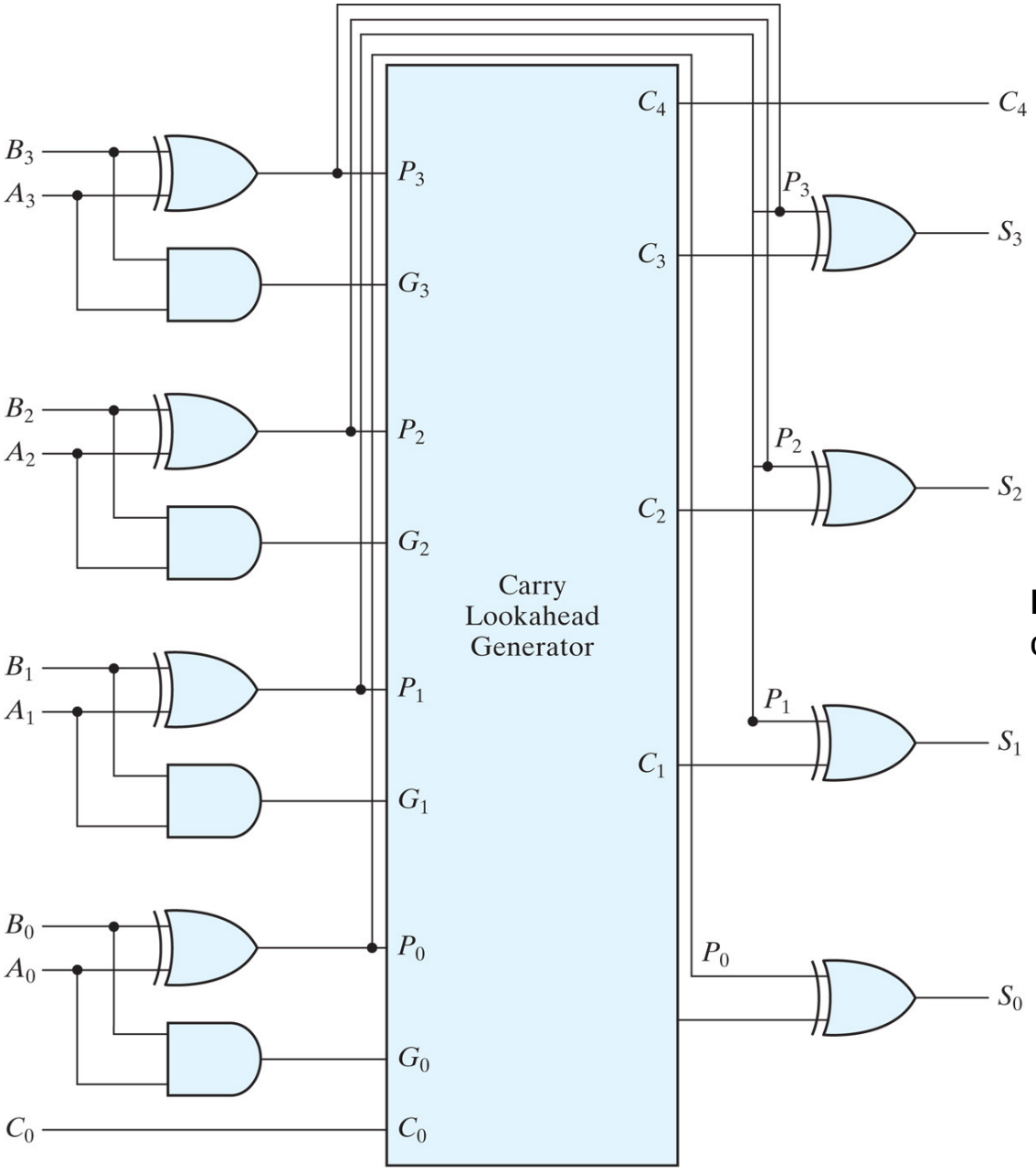
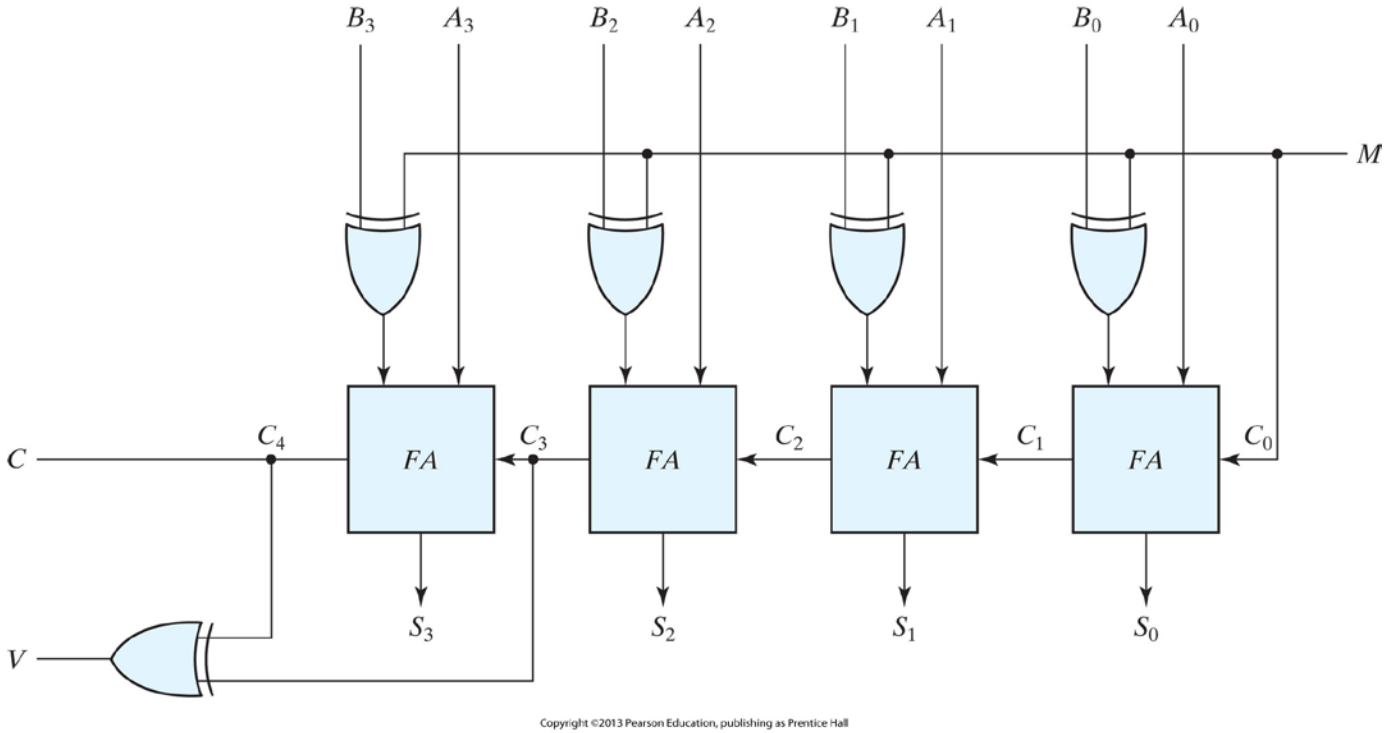


FIGURE 4.12 Four-bit adder with carry lookahead

Copyright ©2013 Pearson Education, publishing as Prentice Hall

Binary Subtractor

- A-B:
  - Adding A to 2's complement of B
  - 2's complement: inverters and sum through the input carry



Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.13 Four-bit adder–subtractor (with overflow detection)

Overflow

- Detect from the end carry out of the most significant position
  - When unsigned numbers
- Consider with the last two carries together
  - When signed numbers (Considering the leftmost sign bit and negative numbers as 2's complement form)
- = If not equal, overflow has occurred

carries:	0 1	carries:	1 0
+70	0 1000110	−70	1 0111010
+80	0 1010000	−80	1 0110000
+150	1 0010110	−150	0 1101010

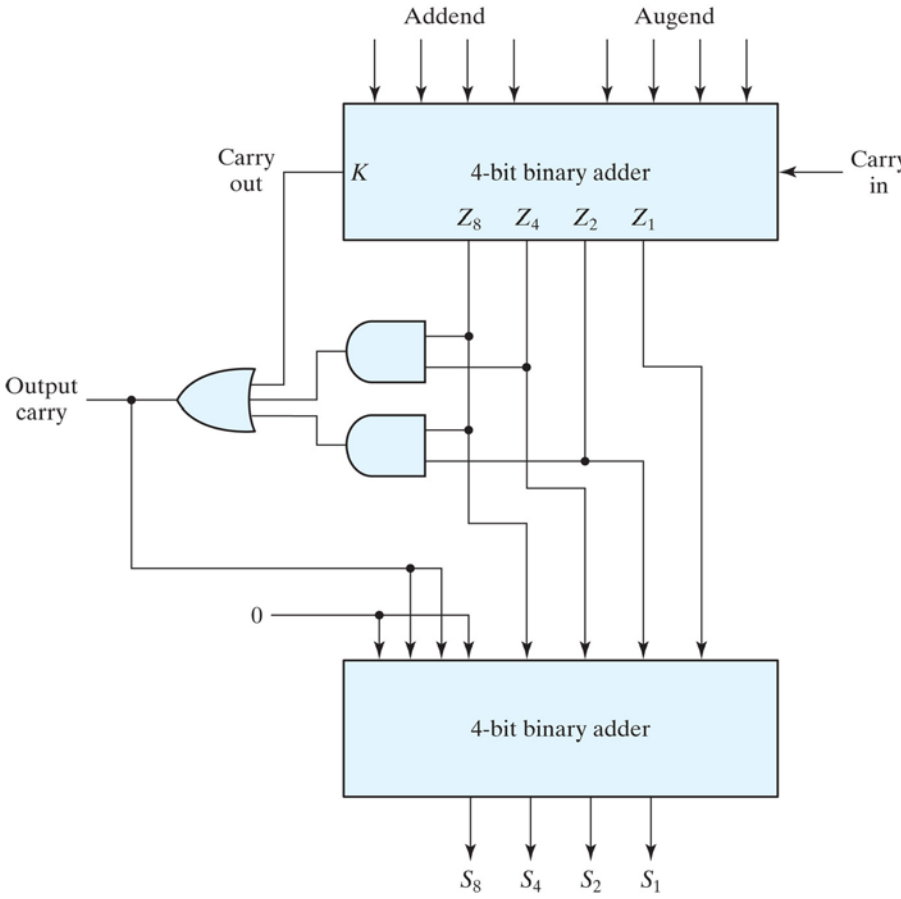
4.6 Decimal Adder

- BCD adder
- 9+9+1 = 19

Table 4.5
Derivation of BCD Adder

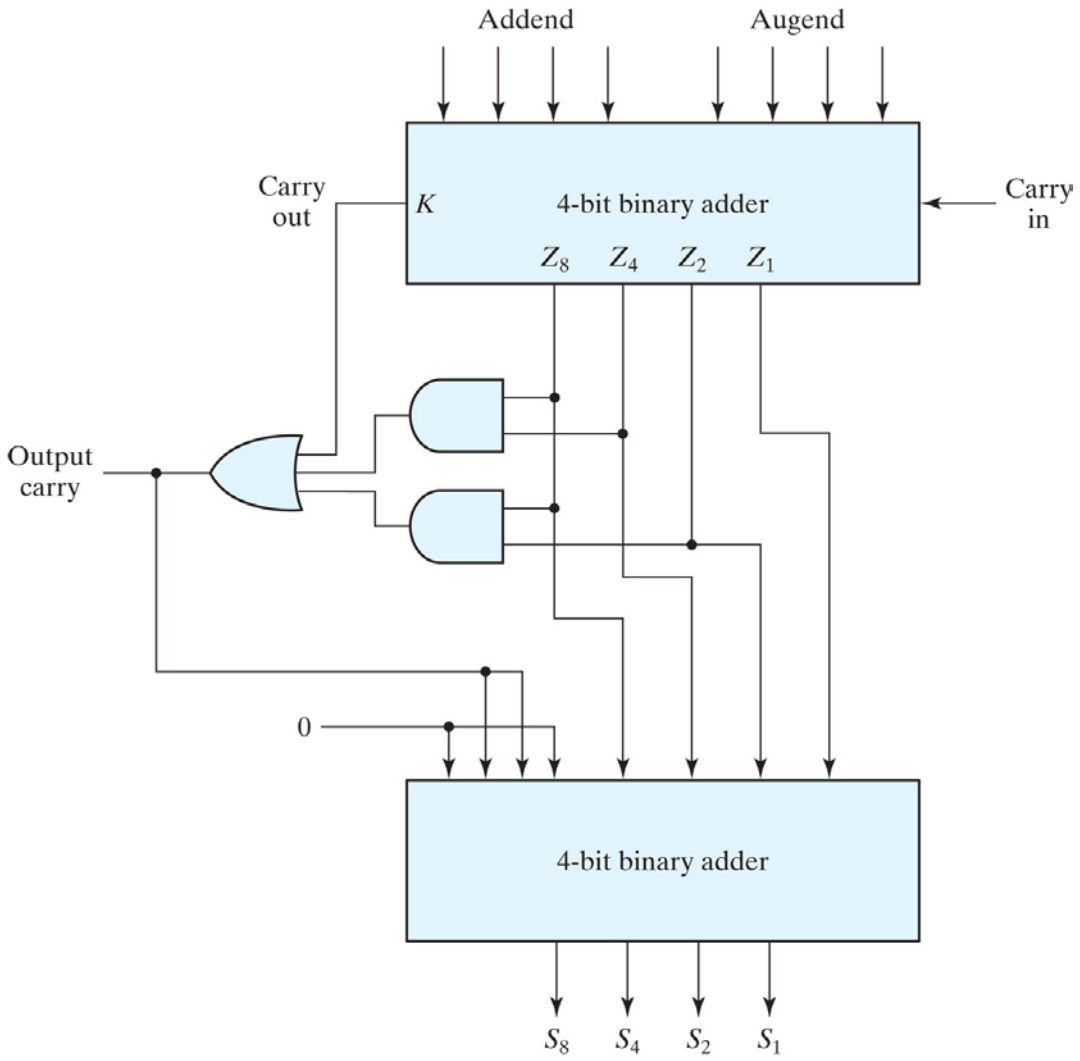
Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Copyright ©2012 Pearson Education, publishing as Prentice Hall



Copyright ©2013 Pearson Education, publishing as Prentice Hall

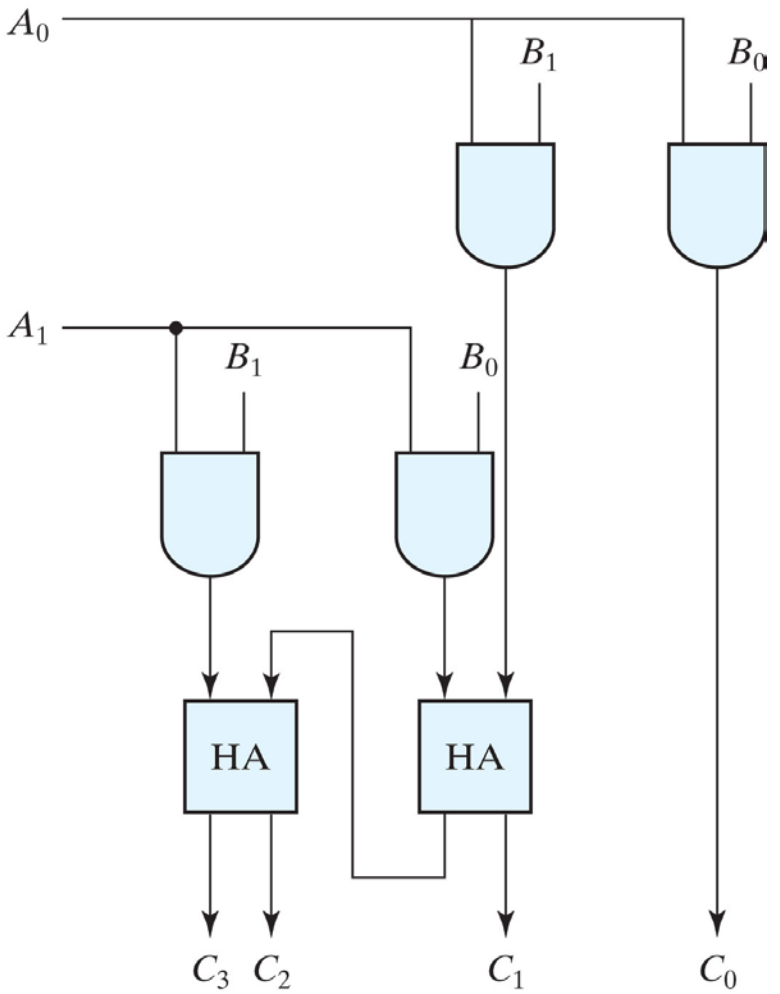
Block diagram of a BCD adder



Copyright ©2013 Pearson Education, publishing as Prentice Hall

4.7 Binary Multiplier

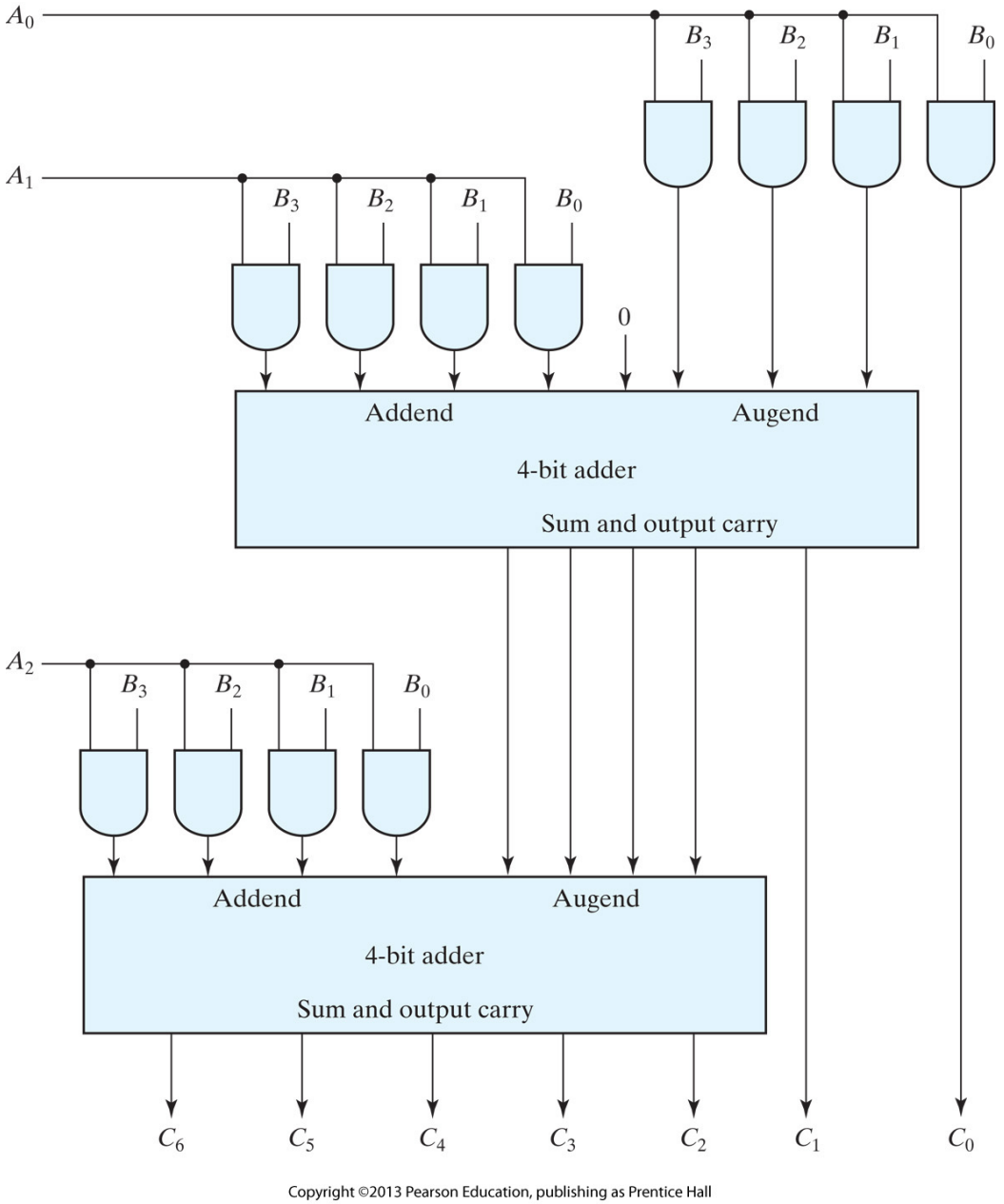
$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



Copyright ©2013 Pearson Education, publishing as Prentice Hall

J multiplier bits
K multiplicand bits
(JXK) AND gates
(J-1) K-bit adders
Result: product of (J+K) bits

FIGURE 4.15 Two-bit by two-bit binary multiplier

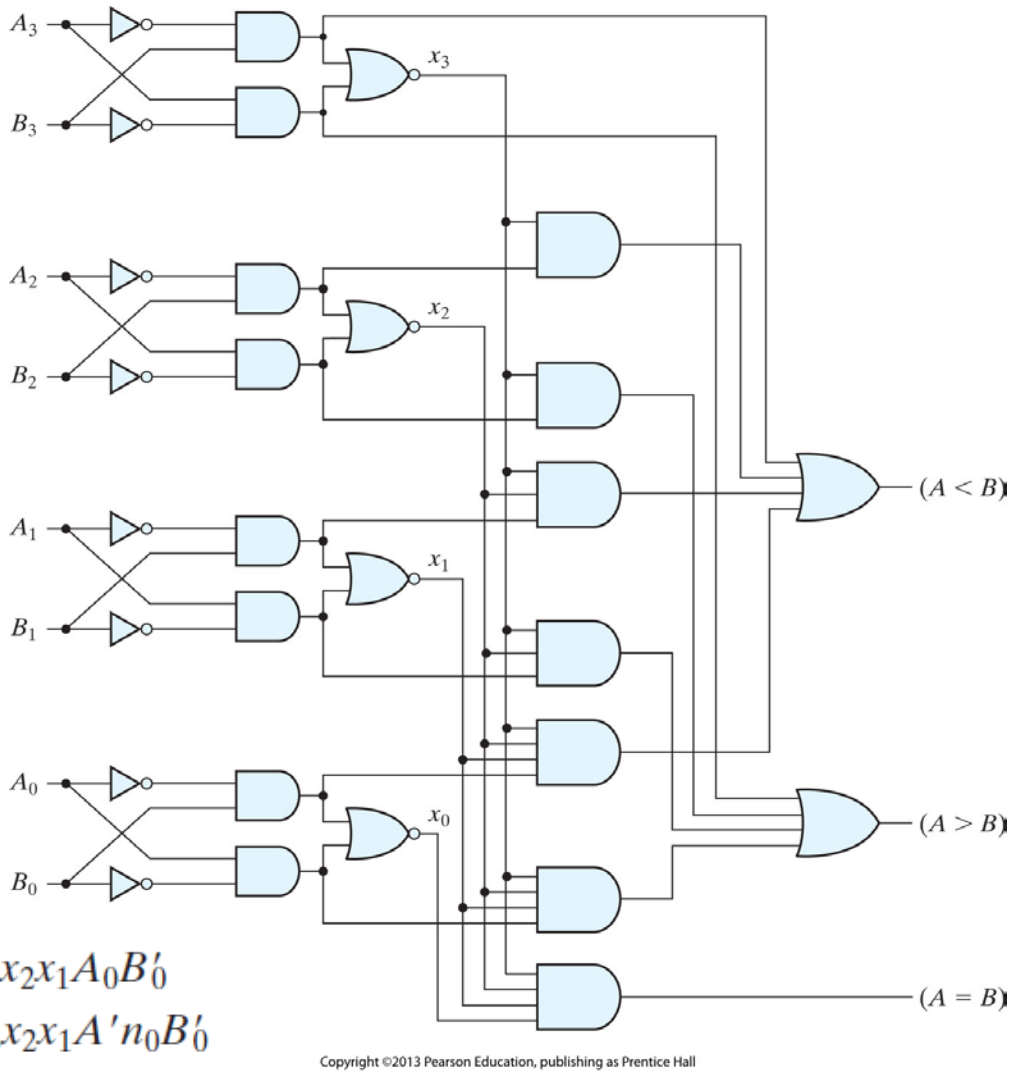


Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.16 Four-bit by three-bit binary multiplier

4.8 Magnitude Comparator

Regularity
→ Algorithm



Copyright ©2013 Pearson Education, publishing as Prentice Hall

$$A = A_3\,A_2\,A_1\,A_0$$
$$B = B_3\,B_2\,B_1\,B_0$$
$$x_i = A_iB_i + A_i'B_i' \quad \text{for } i = 0, 1, 2, 3$$
$$(A = B) = x_3x_2x_1x_0$$
$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$
$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1' + x_3x_2x_1A'n_0B_0'$$

FIGURE 4.17 Four-bit magnitude comparator

4.9 Decoders

- Convert binary information from n input lines to a maximum of 2^n unique output lines
- 2^n (or fewer) minterms of n input variables
- Generate a unique output
- N-to-M line decoder
- Code converter

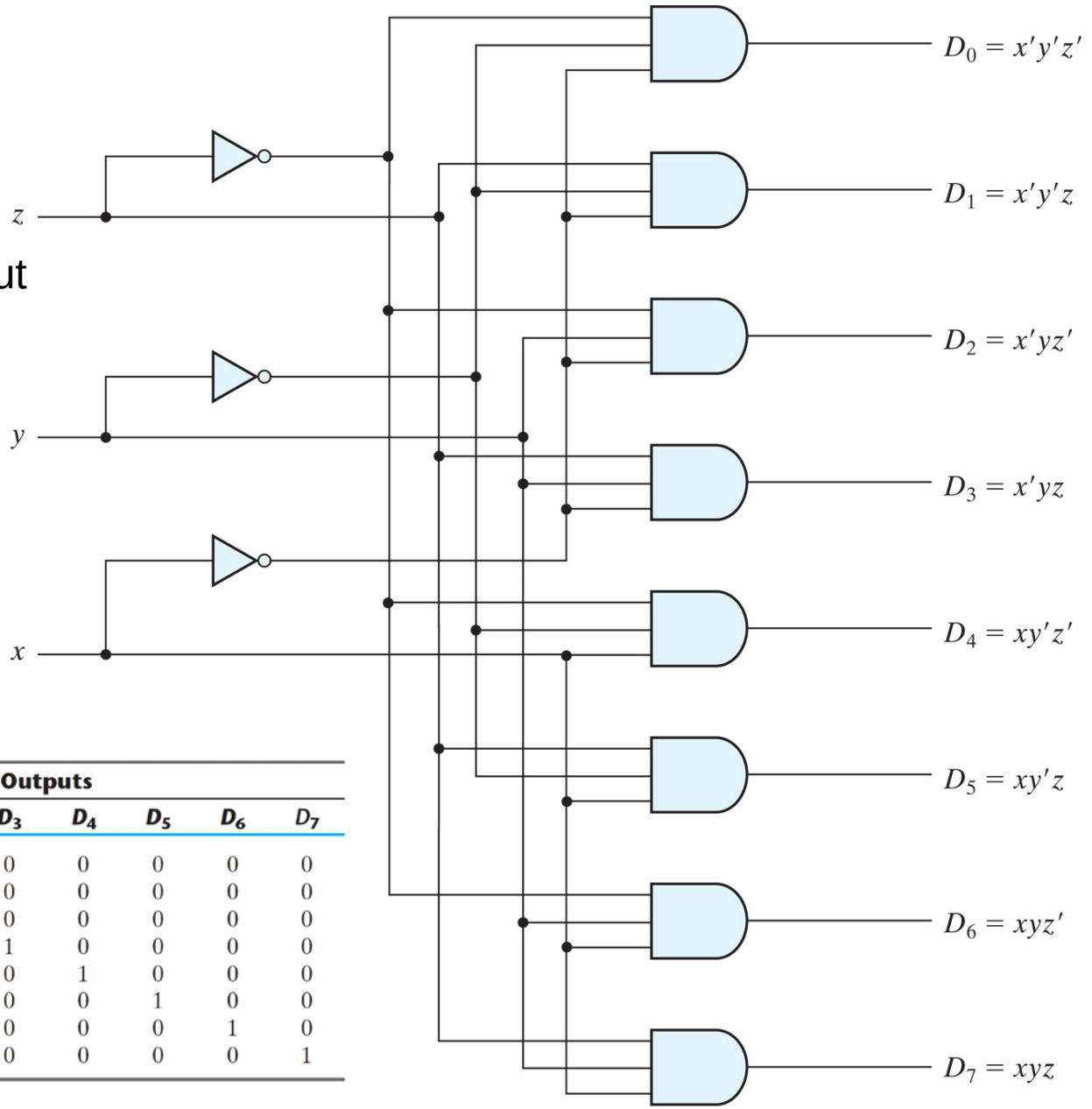


Table 4.6
Truth Table of a Three-to-Eight-Line Decoder

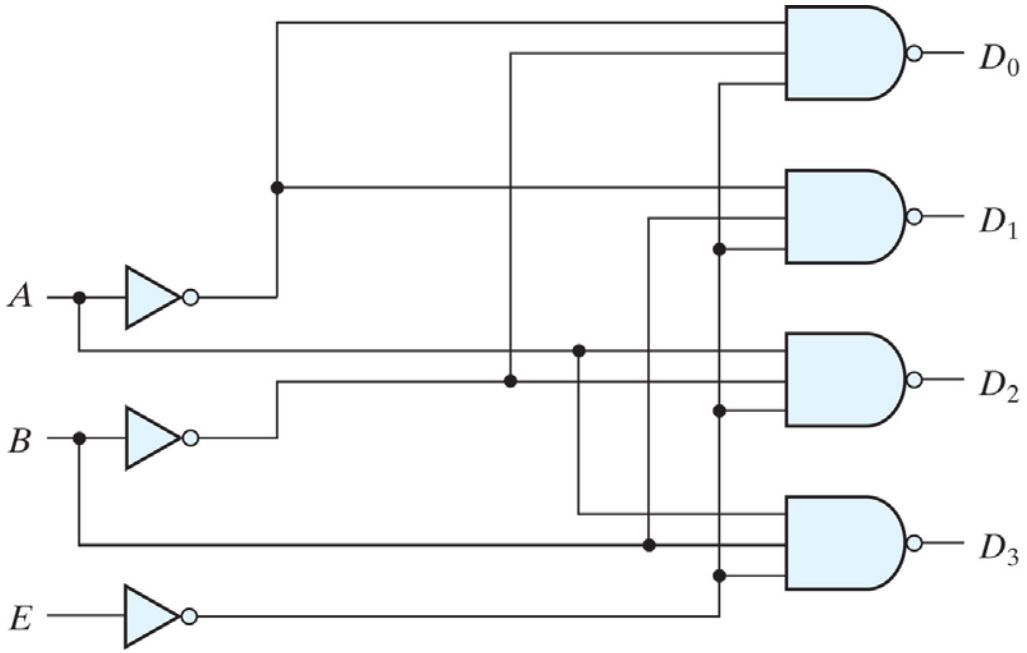
Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Copyright ©2012 Pearson Education, publishing as Prentice Hall

Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.18 Three-to-eight-line decoder

Decoder-Demultiplexer



(a) Logic diagram

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	0

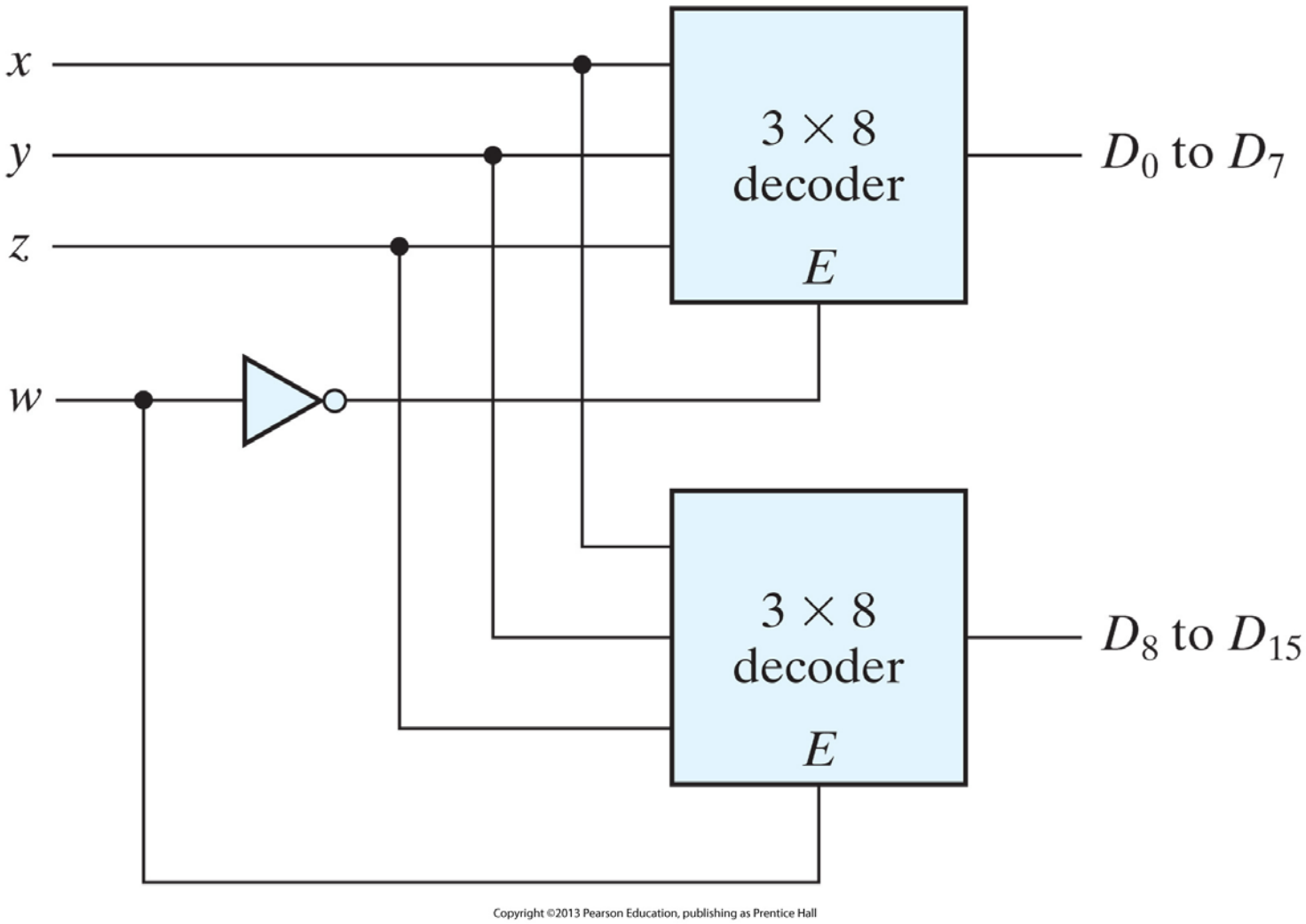
(b) Truth table

Copyright ©2013 Pearson Education, publishing as Prentice Hall

A decoder with enable input can function as a demultiplexer
Ex) one to four line demultiplexer (E: data input line, A and B: selection inputs)

FIGURE 4.19 Two-to-four-line decoder with enable input

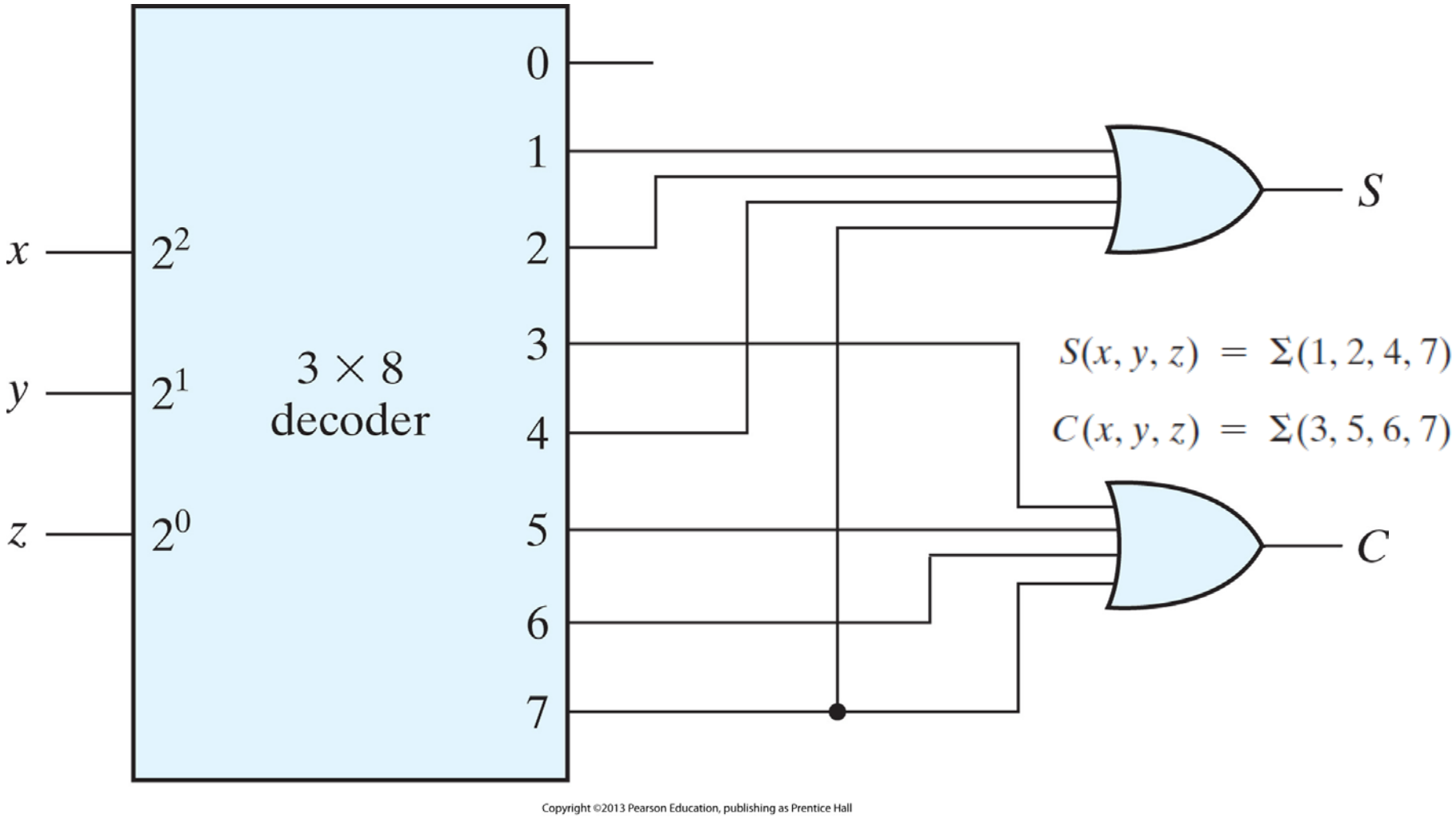
Larger Decoder Circuit



Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.20 4 × 16 decoder constructed with two 3 × 8 decoders

Combinational Logic Implementation



Copyright ©2013 Pearson Education, publishing as Prentice Hall

(a) $S = x'y'z + x'yz' + xy'z' + xyz$ (b) $C = xy + xz + yz$

FIGURE 4.21 Implementation of a full adder with a decoder

Encoders

- 2^n (or fewer) input lines and n output lines
- Assume only one input has a value of 1 at any given time
- Must establish an input priority
- Ambiguities of all 0's inputs

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

Table 4.7
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

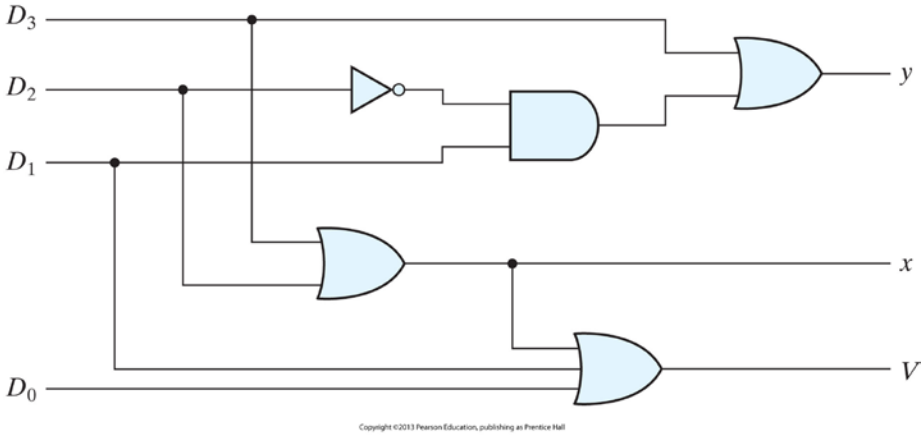
Copyright ©2012 Pearson Education, publishing as Prentice Hall

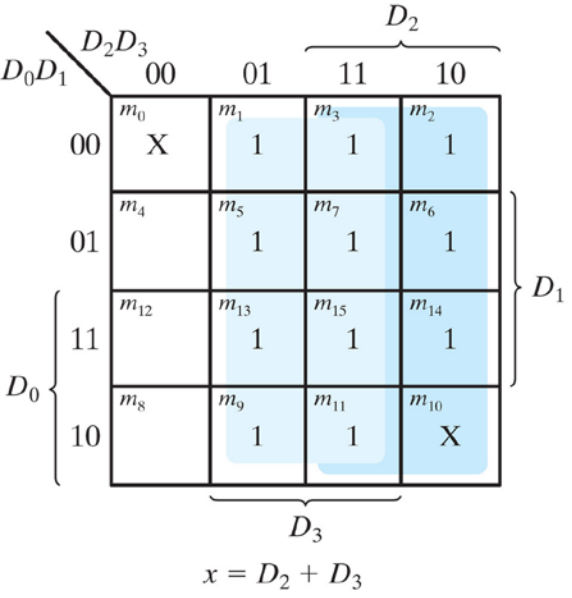
Priority Encoder

V: valid bit indicator

Table 4.8
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





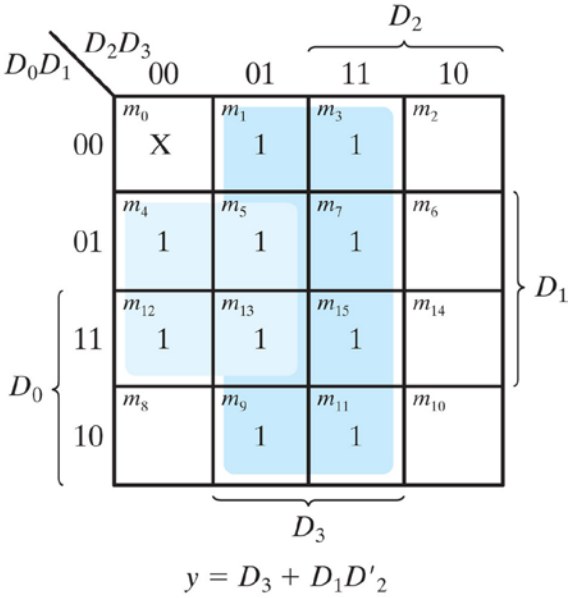


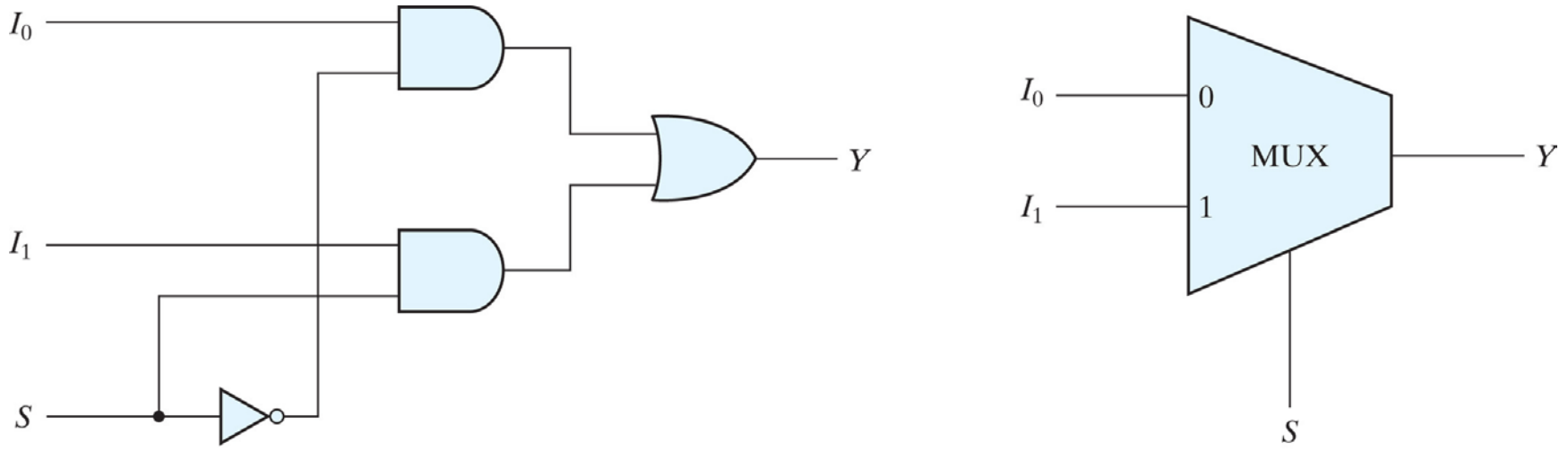
FIGURE 4.22 Maps for a priority encoder

FIGURE 4.23 Four-input priority encoder

$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$

Multiplexers (MUX)

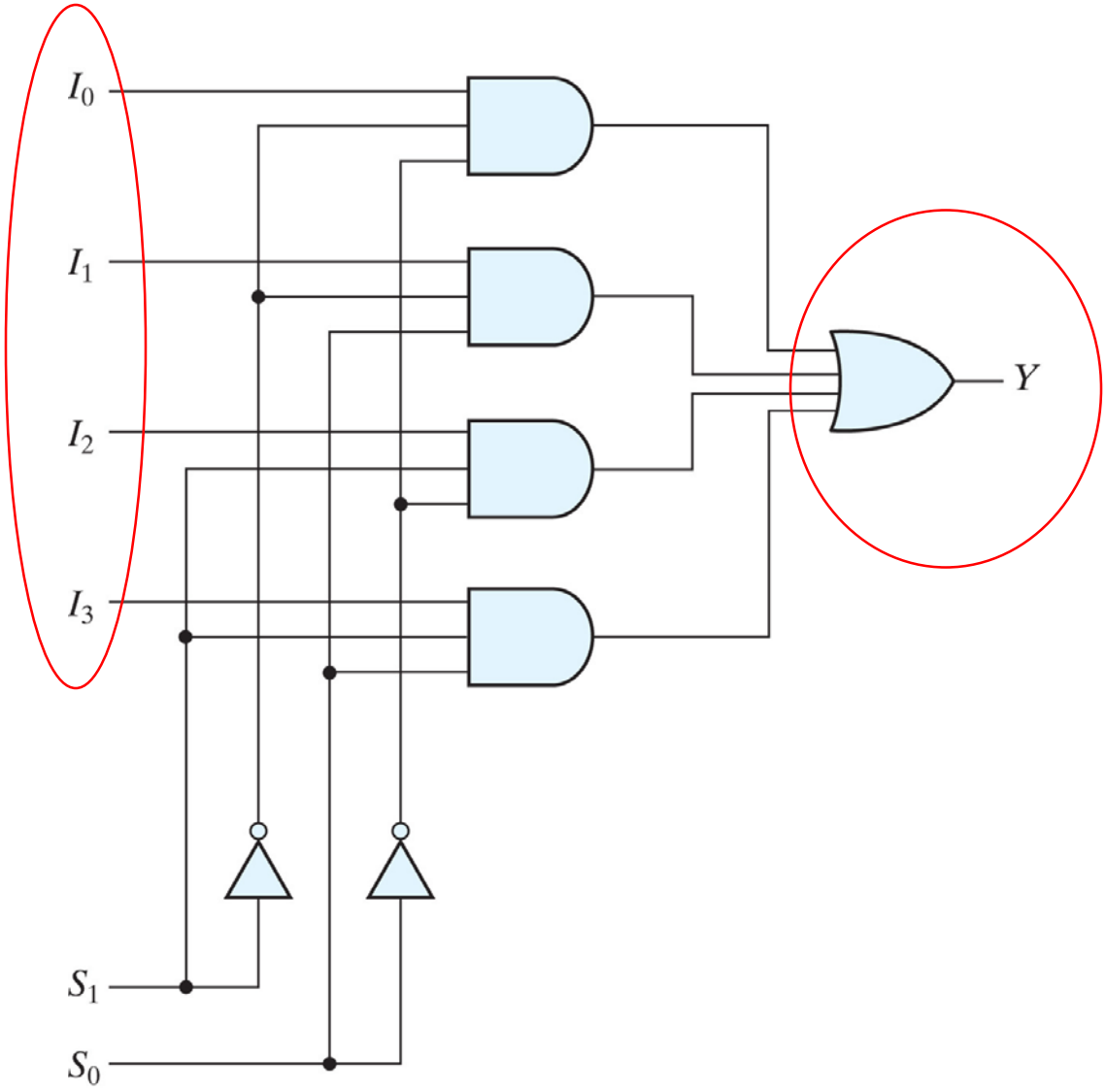
- Select binary information from one of many input lines and directs it to a single output line
- 2^n input lines and n selection lines, a common destination
- Electronic switch that selects one of two sources
- Data selector (providing a path from the selected input to the output)



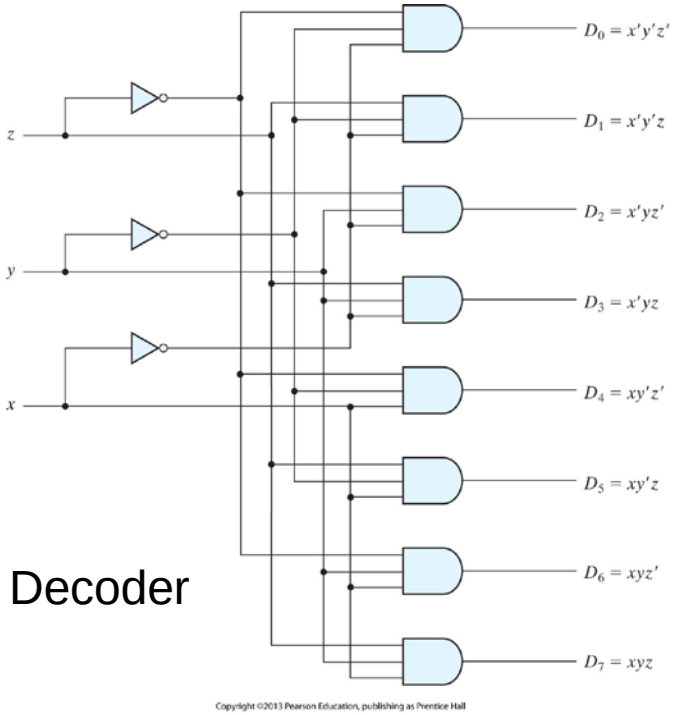
(a) Logic diagram (b) Block diagram

Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.24 Two-to-one-line multiplexer



(a) Logic diagram



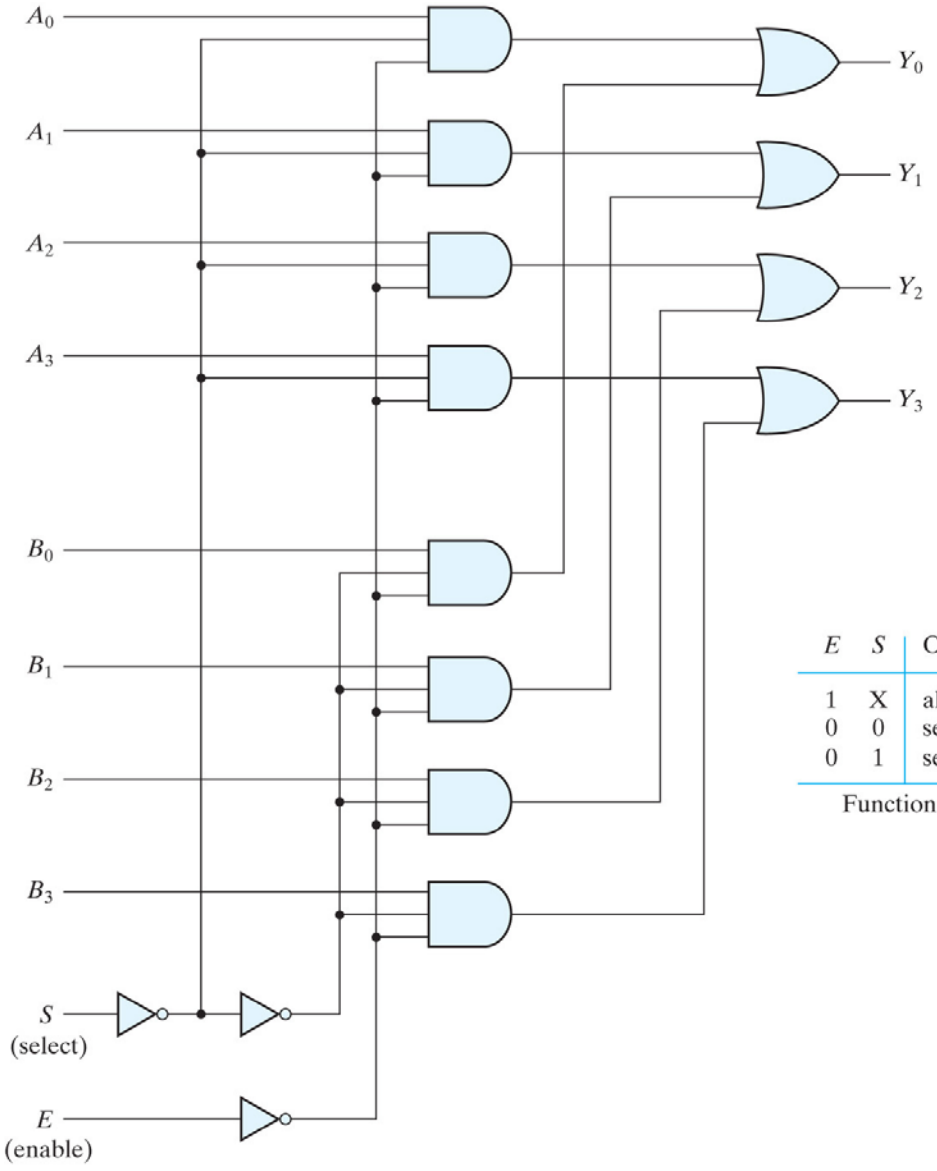
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.25 Four-to-one-line multiplexer

Multiple-bit Selection Logic



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

Function table

FIGURE 4.26 Quadruple two-to-one-line multiplexer

Copyright ©2013 Pearson Education, publishing as Prentice Hall

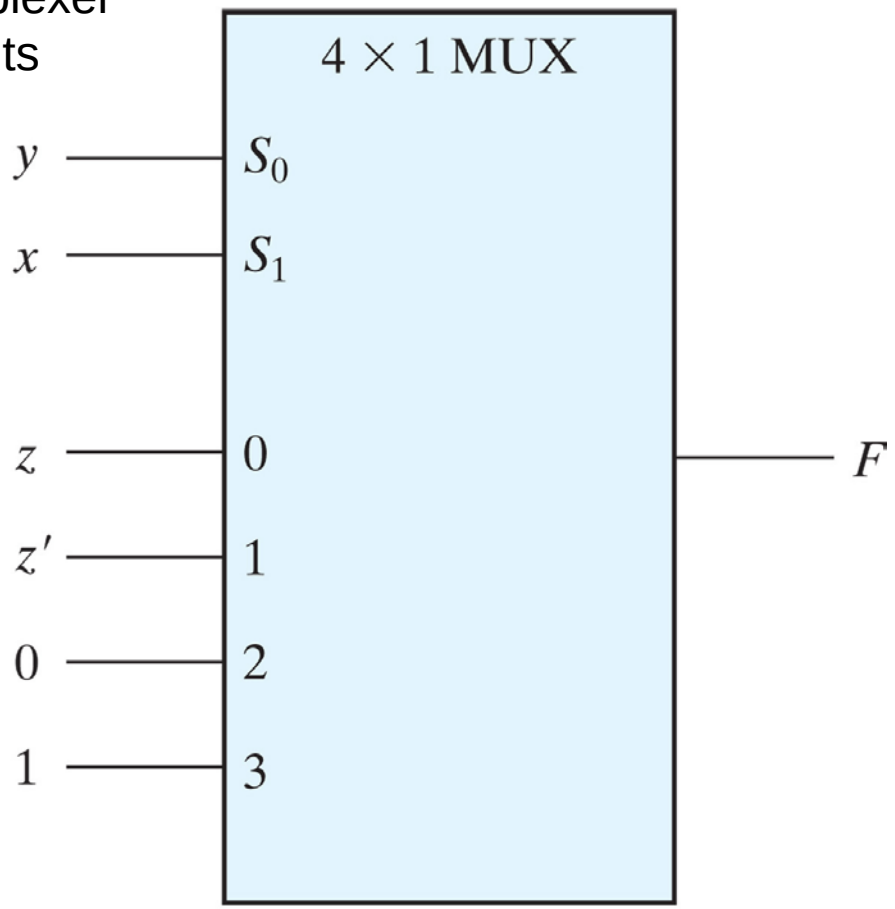
Boolean Function Implementation

Bolean function of n variables with a multiplexer with n-1 selection inputs and 2^{n-1} data inputs

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

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



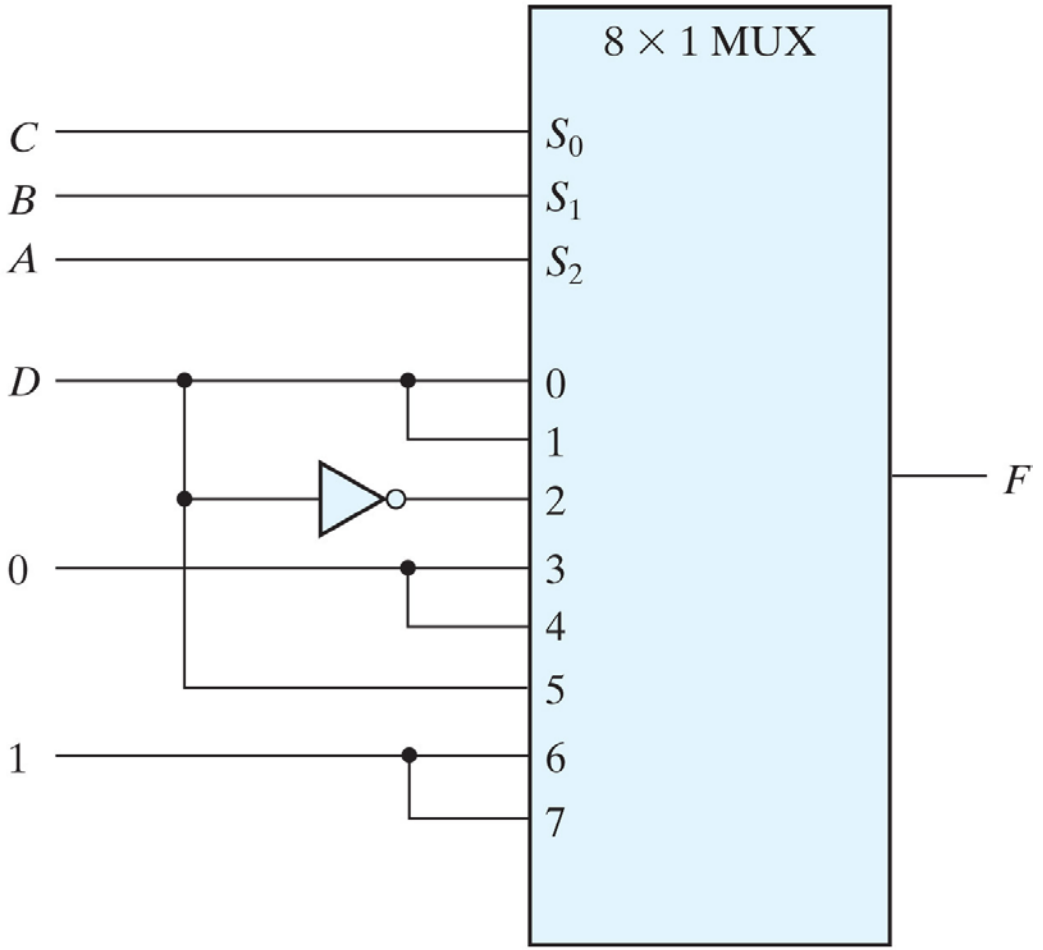
(b) Multiplexer implementation

Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.27 Implementing a Boolean function with a multiplexer

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	

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



Copyright ©2013 Pearson Education, publishing as Prentice Hall

FIGURE 4.28 Implementing a four-input function with a multiplexer

Three-State Gates

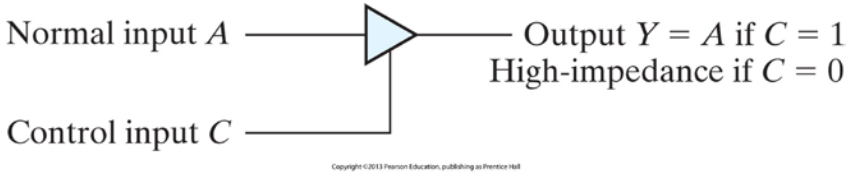
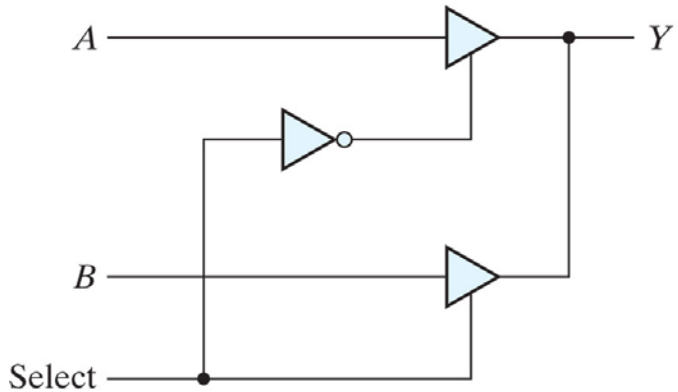


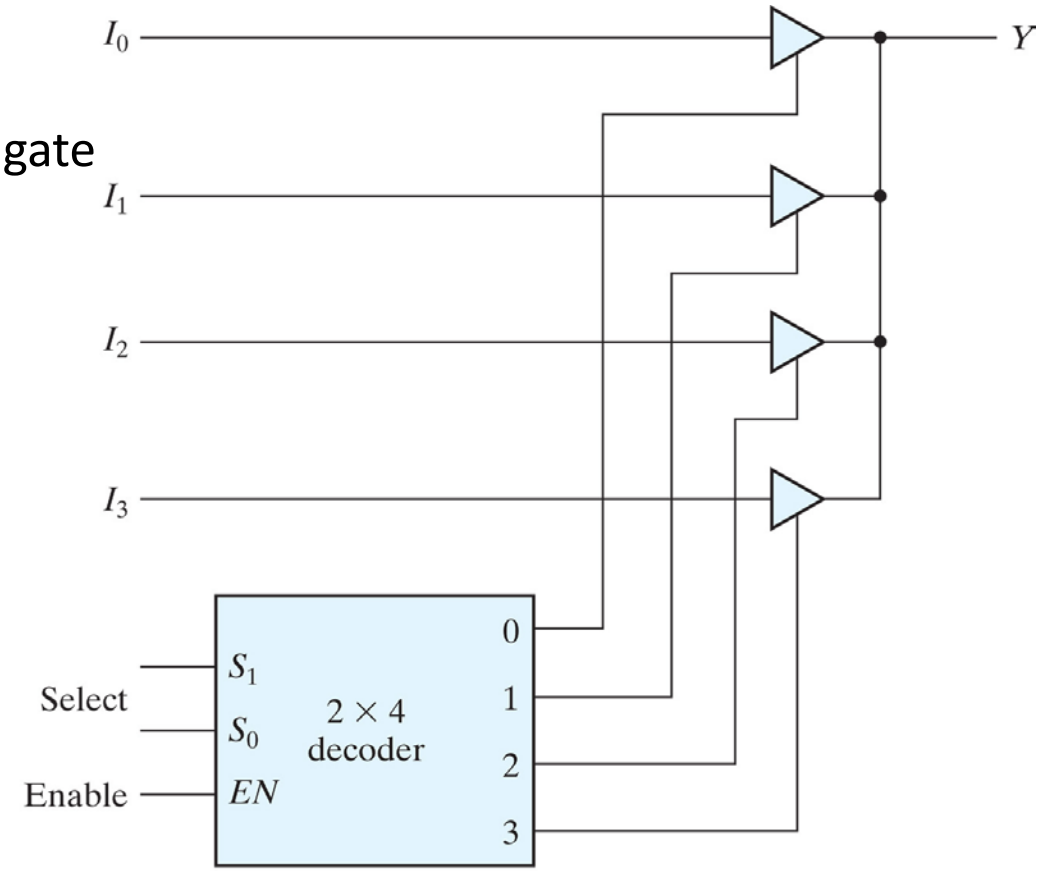
FIGURE 4.29 Graphic symbol for a three-state buffer

- A multiplexer constructed with three-state gates
- The third state: high impedance state
  - Behave like an open circuit
  - No logic significance
  - Not affected by the inputs to the gate
  - Buffer gate

FIGURE 4.30 Multiplexers with three-state gates



(a) 2-to-1-line mux



(b) 4-to-1-line mux

Copyright ©2013 Pearson Education, publishing as Prentice Hall