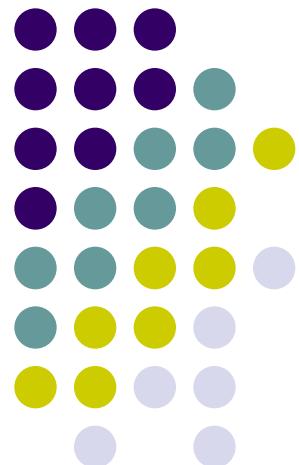
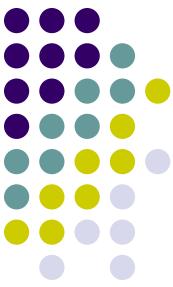


# Digital Systems: Combinational Logic Circuits

---

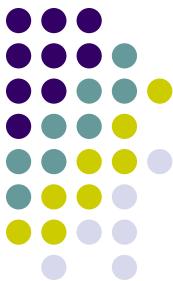
Wen-Hung Liao, Ph.D.





# Objectives

- Convert a logic expression into a sum-of-products expression.
- Perform the necessary steps to reduce a sum-of-products expression to its simplest form.
- Use Boolean algebra and the Karnaugh map as tools to simplify and design logic circuits.
- Explain the operation of both exclusive-OR and exclusive-NOR circuits.
- Design simple logic circuits without the help of a truth table.



# Objectives (cont'd)

- Implement enable circuits.
- Cite the basic characteristics of TTL and CMOS digital ICs.
- Use the basic troubleshooting rules of digital systems.
- Deduce from observed results the faults of malfunctioning combinational logic circuits.
- Describe the fundamental idea of programmable logic devices (PLDs).
- Outline the steps involved in programming a PLD to perform a simple combinational logic function



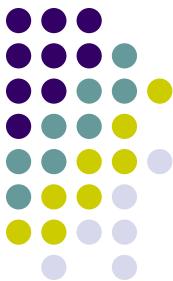
# Combinational Logic Circuits

- The logic level at the output depends on the combination of logic levels present at the inputs.
- A combinational circuit has no memory, so its output depends only on the current value of its inputs.
- We will not spend a great deal of time discussing how to troubleshoot the combinational circuits. (That's what the lab is for.)



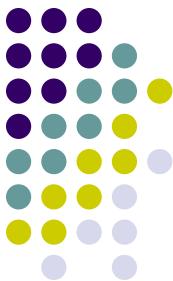
# Sum-of-Products (SOP) Form

- Sum → OR
- Product → AND
- Each of the sum-of-products expression consists of two or more AND terms that are ORed together.
- Examples:  
 $ABC + A'BC'$   
 $AB + A'BC' + C'D' + D$
- Note that one inversion sign cannot cover more than one variable in a term.  $\bar{AB}$  is not allowed.



# Product-of-Sums (POS) Form

- Each of the product-of-sums expression consists of two or more OR terms that are ANDed together.
- Examples:  $(A+B'+C)(A+C)$   
 $(A+B')(C'+D)F$
- Will use sum-of-products form in logic circuit simplification.

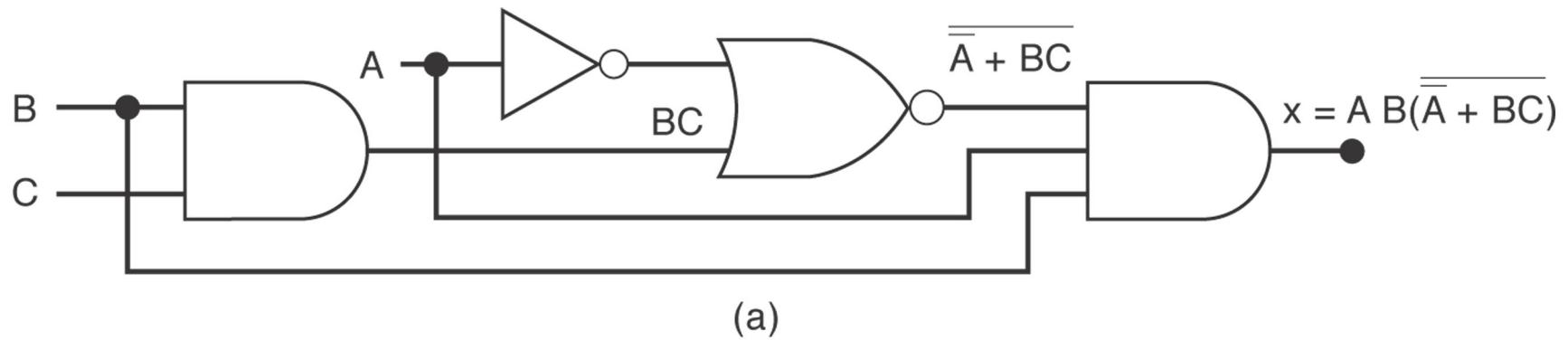


# Simplifying Logic Circuits

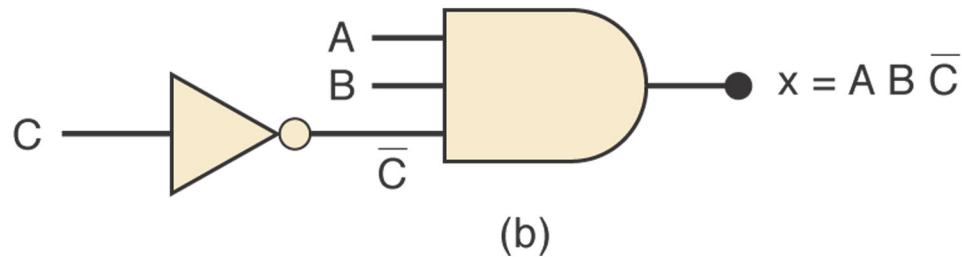
- Goal: reduce the logic circuit expression to a simpler form so that fewer gates and connections are required to build the circuit.
- Example: 4.1(a) and 4.1(b) are equivalent, but 4-1(b) is much simpler.



# Figure 4.1



(a)



(b)



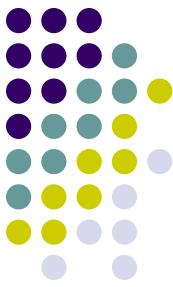
# Circuit Simplification Methods

- Boolean algebra: greatly depends on inspiration and experience.
- Karnaugh map: systematic, step-by-step approach.
- Pros and Cons



# Algebraic Simplification

- Use the Boolean algebra theorems introduced in Chapter 3 to help simplify the expression for a logic circuit.
- Based on experience, often becomes a trial-and-error process.
- No easy way to tell whether a simplified expression is in its simplest form.



# Two Essential Steps

- The original expression is put into the sum-of-products form by repeated application of DeMorgan's theorem and multiplication of terms.
- The product terms are checked for common factors, and factoring is performed whenever possible.



# Examples 4-1 to 4-4

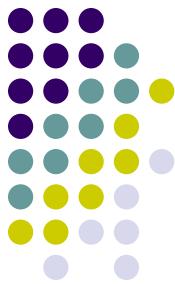
Original	Simplified
$ABC + AB'(A'C)'$	$A(B' + C)$
$ABC + ABC' + AB'C$	$A(B + C)$
$A'C(A'BD)' + A'BC'D' + AB'C$	$B'C + A'D'(B + C)$
$(A' + B)(A + B + D)D'$	$BD'$



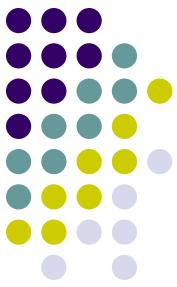
## Examples 4-5, 4-6

- $(A'+B)(A+B')$ : equivalent form  $A'B'+AB$
- $AB'C+A'BD+C'D'$ : cannot be simplified further.

# Designing Combinational Logic Circuits



1. Set up the truth table.
2. Write the AND term for each case where the output is a 1.
3. Write the sum-of-products expression for the output.
4. Simplify the output expression.
5. Implement the circuit for the final expression.

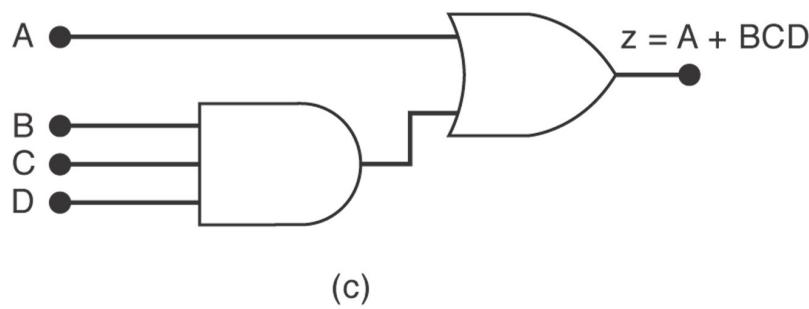
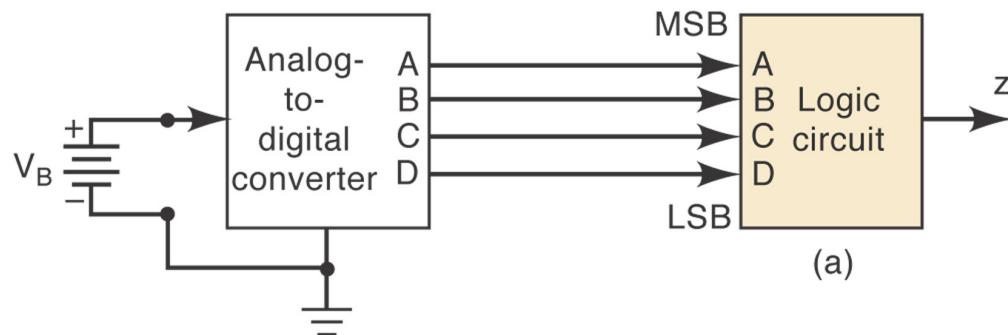


## Example 4-8

- Design a logic circuit that is to produce a HIGH output when the voltage (represented by a four-bit binary number ABCD) is greater than 6V.



# Example 4-8: Answer



	A	B	C	D	z
(0)	0	0	0	0	0
(1)	0	0	0	1	0
(2)	0	0	1	0	0
(3)	0	0	1	1	0
(4)	0	1	0	0	0
(5)	0	1	0	1	0
(6)	0	1	1	0	0
(7)	0	1	1	1	1 → $\bar{A}\bar{B}\bar{C}\bar{D}$
(8)	1	0	0	0	1 → A $\bar{B}\bar{C}\bar{D}$
(9)	1	0	0	1	1 → A $\bar{B}\bar{C}D$
(10)	1	0	1	0	1 → A $\bar{B}CD$
(11)	1	0	1	1	1 → A $\bar{B}C\bar{D}$
(12)	1	1	0	0	1 → ABCD
(13)	1	1	0	1	1 → ABC $\bar{D}$
(14)	1	1	1	0	1 → AB $\bar{C}\bar{D}$
(15)	1	1	1	1	1 → ABCD

(b)

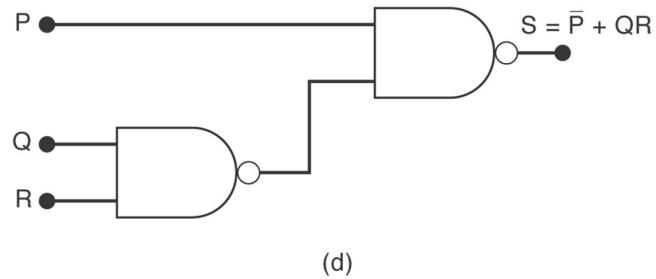
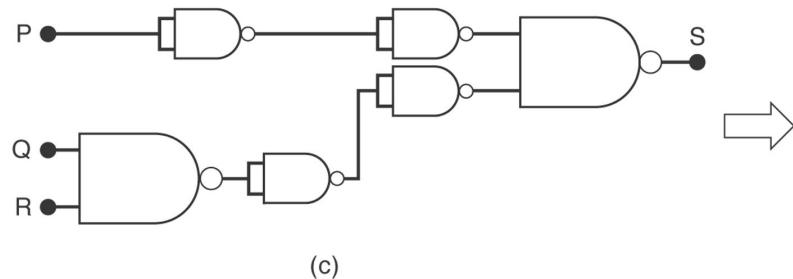
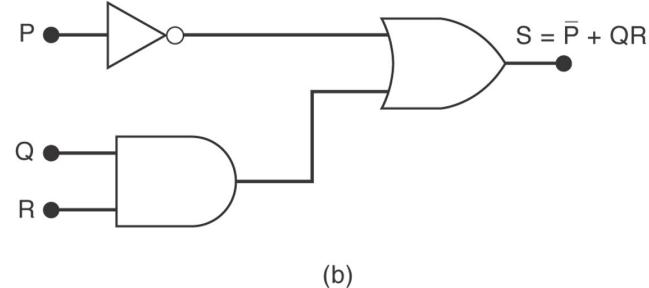
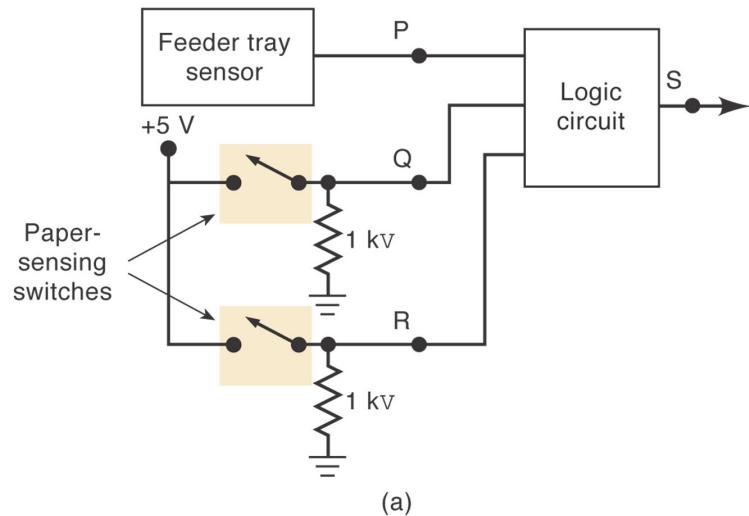


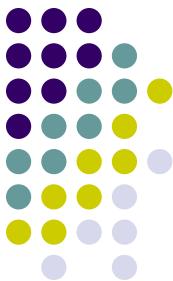
## Example 4-9

- Generate the STOP signal and energize an indicator light whenever either of the following conditions exists: (1) there is no paper in the paper feeder tray; or (2) the two micro-switches in the paper path are activated, indicating a jam.



# Example 4-9: Answer





# Karnaugh Map Method

- A graphical device to simplify a logic expression.
- Will only work on examples with up to 4 input variables.
- From truth table to logic expression to K map.
- Figure 4.11 shows the K map with 2,3 and 4 variables.



# Figure 4.11

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → $AB$

$$\left\{ x = \bar{A}\bar{B} + AB \right\}$$

	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1

(a)

A	B	C	X
0	0	0	1 → $\bar{A}\bar{B}\bar{C}$
0	0	1	1 → $\bar{A}\bar{B}C$
0	1	0	1 → $A\bar{B}\bar{C}$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1 → $ABC\bar{C}$
1	1	1	0

$$\left\{ X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + A\bar{B}\bar{C} \right\}$$

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	0
AB	1	0
A $\bar{B}$	0	0

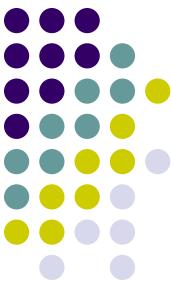
(b)

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1 → $\bar{A}\bar{B}\bar{C}\bar{D}$
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1 → $\bar{A}\bar{B}\bar{C}D$
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1 → $ABC\bar{D}$
1	1	1	0	0
1	1	1	1	1 → $ABCD$

$$\left\{ X = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + ABC\bar{D} + A\bar{B}C\bar{D} \right\}$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$\bar{C}D$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
A $\bar{B}$	0	0	0	0

(c)



# Looping

- The expression for output X can be simplified by properly combining those squares in the K map which contain 1s. The process of combining these 1s is called **looping**.
- Looping groups of two (pairs) → eliminate 1 variable
- Looping groups of four (quads) → eliminate 2 variables
- Looping groups of eight (octets) → eliminate 3 variables
- See Figure 4-12 to 4-14.



# Figure 4-12: Looping Pairs

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	1	0
AB	1	0
A $\bar{B}$	0	0

(a)

$$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} \\ = B\bar{C}$$

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	1	1
AB	0	0
A $\bar{B}$	0	0

(b)

$$X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C \\ = \bar{A}\bar{B}$$

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	0
$\bar{A}B$	0	0
AB	0	0
A $\bar{B}$	1	0

(c)

$$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} = \bar{B}\bar{C}$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	1
$\bar{A}B$	0	0	0	0
AB	0	0	0	0
A $\bar{B}$	1	0	0	1

(d)

$\bar{A}\bar{B}\bar{C}$

$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} \\ + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} \\ = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{D}$$

$A\bar{B}\bar{D}$



# Figure 4-13: Looping Quads

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	0	1
AB	0	1
$A\bar{B}$	0	1

$X = C$

(a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	0	0
AB	1	1	1	1
$A\bar{B}$	0	0	0	0

$X = AB$

(b)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	1	1	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

$X = BD$

(c)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	0	0
AB	1	0	0	1
$A\bar{B}$	1	0	0	1

$X = A\bar{D}$

(d)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	0	0	0
AB	0	0	0	0
$A\bar{B}$	1	0	0	1

$X = \bar{B}\bar{D}$

(e)



# Figure 4-14: Looping Octets

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	1	1	1	1
$A\bar{B}$	1	1	1	1
$A\bar{B}$	0	0	0	0

$X = B$

(a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	0	0
$\bar{A}B$	1	1	0	0
$A\bar{B}$	1	1	0	0
$A\bar{B}$	1	1	0	0

$X = \bar{C}$

(b)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	0	0	0	0
$A\bar{B}$	0	0	0	0
$A\bar{B}$	1	1	1	1

$X = \bar{B}$

(c)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	1	0	0	1
$A\bar{B}$	1	0	0	1
$A\bar{B}$	1	0	0	1

$X = \bar{D}$

(d)

# Complete Simplification Process



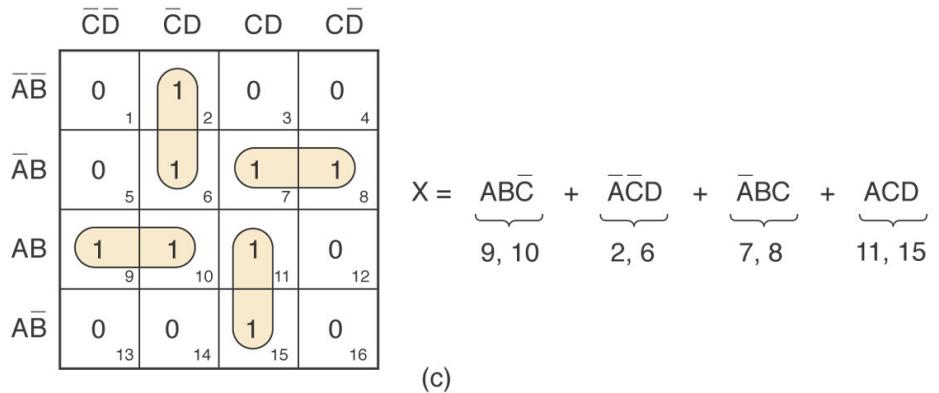
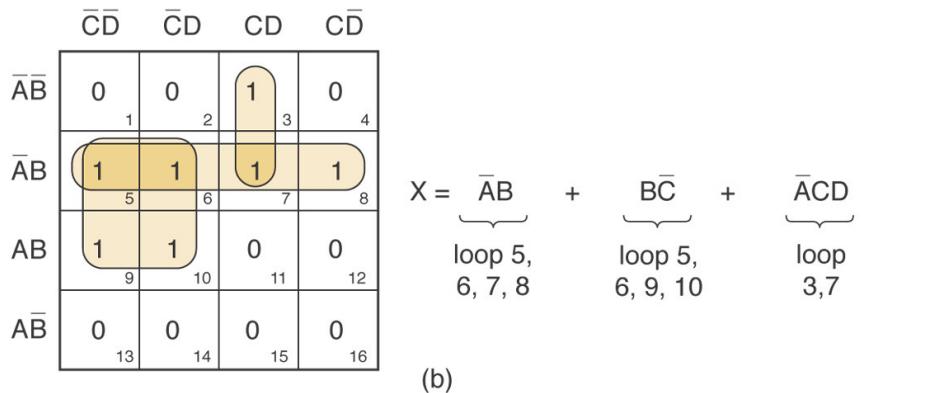
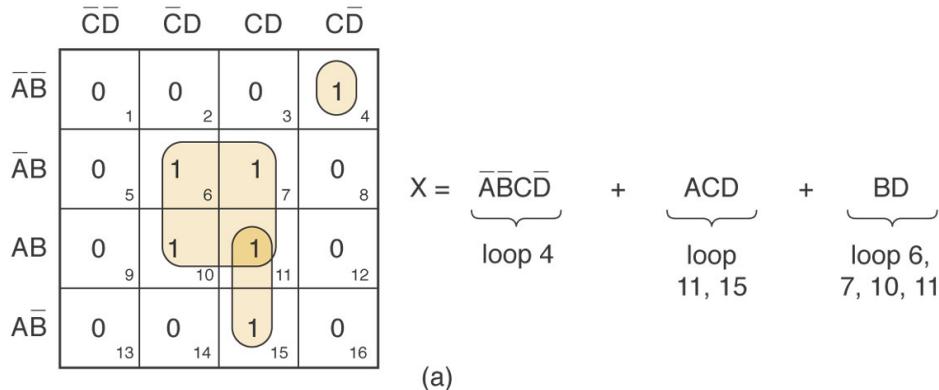
- Step 1: Construct the K map and places 1s in those squares corresponding to the 1s in the truth table. Places 0s in the other squares.
- Step 2: Examine the map for adjacent 1s and loop those 1s which are not adjacent to any other 1s. (isolated 1s)
- Step 3: Look for those 1s which are adjacent to only one other 1. Loop any pair containing such a 1.
- Step 4: Loop any octet even when it contains some 1s that have already been looped.

# Complete Simplification Process



- Step 5: Loop any quad that contains one or more 1s that have not already been looped, making sure to use the minimum number of loops.
- Step 6: Loop any pairs necessary to include any 1s have not already been looped, making sure to use the minimum number of loops.
- Step 7: Form the ORed sum of all the terms generated by each loop.

# Examples



$$X = \underbrace{\bar{A}\bar{B}\bar{C}\bar{D}}_{\text{loop 4}} + \underbrace{ACD}_{\text{loop 11, 15}} + \underbrace{BD}_{\text{loop 6, 7, 10, 11}}$$

(a)

$$X = \underbrace{\bar{A}B}_{\text{loop 5, 6, 7, 8}} + \underbrace{B\bar{C}}_{\text{loop 5, 6, 9, 10}} + \underbrace{\bar{A}CD}_{\text{loop 3, 7}}$$

(b)

$$X = \underbrace{ABC}_{9, 10} + \underbrace{\bar{A}\bar{C}D}_{2, 6} + \underbrace{\bar{A}\bar{B}C}_{7, 8} + \underbrace{ACD}_{11, 15}$$



# More Examples

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	1	1
$A\bar{B}$	0	0	0	1
$A\bar{B}$	1	1	0	1

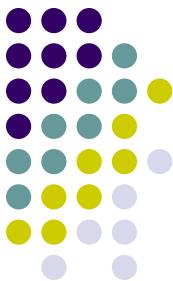
$$X = \bar{A}\bar{C}D + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ACD$$

(a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	1	1
$A\bar{B}$	0	0	0	1
$A\bar{B}$	1	1	0	1

$$X = \bar{A}BD + BCD + \bar{B}\bar{C}D + A\bar{B}\bar{D}$$

(b)



# Filling K Map from Output Expression

- What to do when the desired output is presented as a Boolean expression instead of a truth table?
- Step 1: Convert the expression into SOP form.
- Step 2: For each product term in the SOP expression, place a 1 in each K-map square whose label contains the same combination of input values. Place a 0 in other squares.
- Example 4-14:  $y=C'(A'B'D'+D)+AB'C+D'$



## Example 4-14

	$\bar{C}\bar{D}$	$\bar{C}D$	$CD$	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	0	1
$\bar{A}B$	1	1	0	1
$A\bar{B}$	1	1	0	1
$AB$	1	1	1	1

$$y = A\bar{B} + \bar{C} + \bar{D}$$



# Don't-Care Conditions

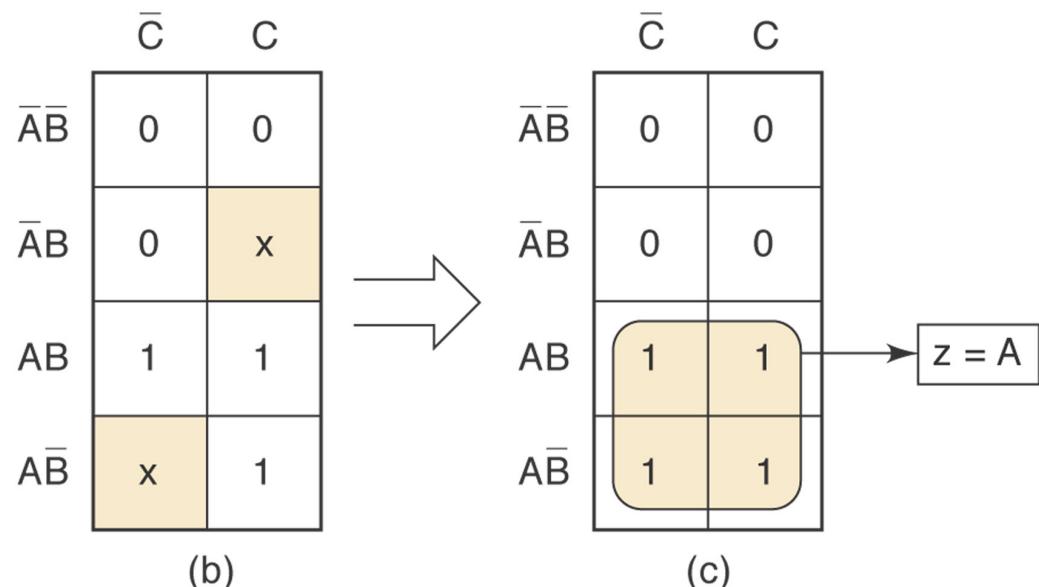
- Some logic circuits can be designed so that there are certain input conditions for which there are no specified output levels.
- A circuit designer is free to make the output for any don't care condition either a 0 or a 1 in order to produce the simplest output expression.
- Figures 4-18,19.



# Figure 4-18

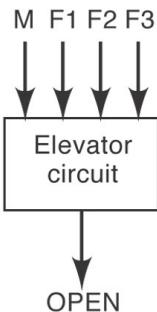
A	B	C	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x } "don't
1	0	0	x } care"
1	0	1	1
1	1	0	1
1	1	1	1

(a)





# Figure 4-19



(a)

M	F1	F2	F3	OPEN
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(b)

	$\bar{F}_2\bar{F}_3$	$\bar{F}_2F_3$	$F_2F_3$	$F_2\bar{F}_3$
$\bar{M}\bar{F}_1$	0	1	X	1
$\bar{M}F_1$	1	X	X	X
$M\bar{F}_1$	0	X	X	X
$M\bar{F}_1$	0	0	X	0

(c)

	$\bar{F}_2\bar{F}_3$	$\bar{F}_2F_3$	$F_2F_3$	$F_2\bar{F}_3$
$\bar{M}\bar{F}_1$	0	1	1	1
$\bar{M}F_1$	1	1	1	1
$M\bar{F}_1$	0	0	0	0
$M\bar{F}_1$	0	0	0	0

(d)

$$OPEN = \bar{M} (F_1 + F_2 + F_3)$$

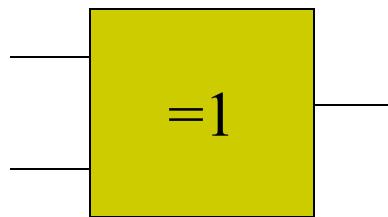


# Exclusive-OR

- Exclusive-OR (XOR)

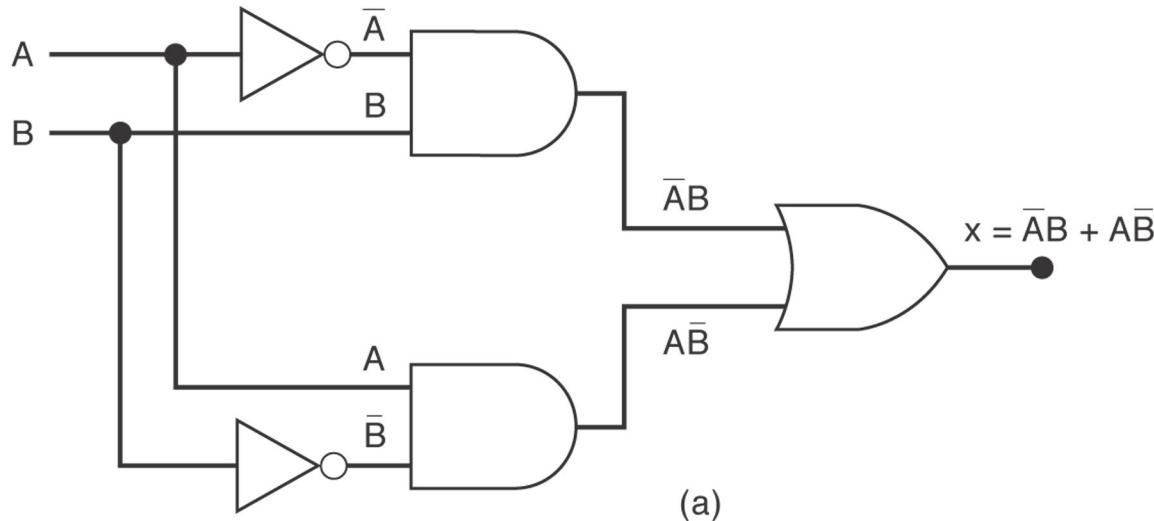
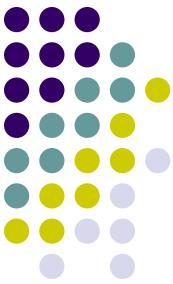
$$x = A'B + AB'$$

- Timing diagram



XOR		
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

# XOR



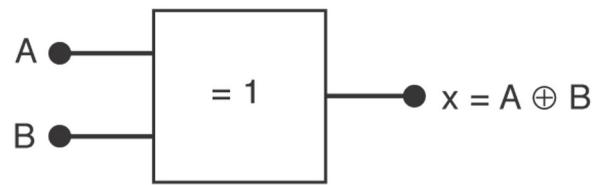
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

(a)

XOR gate symbols



(b)



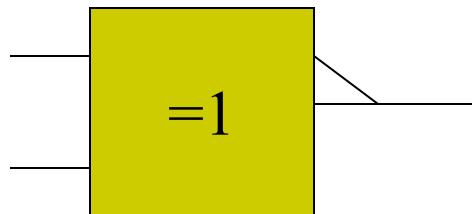
(c)



# Exclusive-NOR

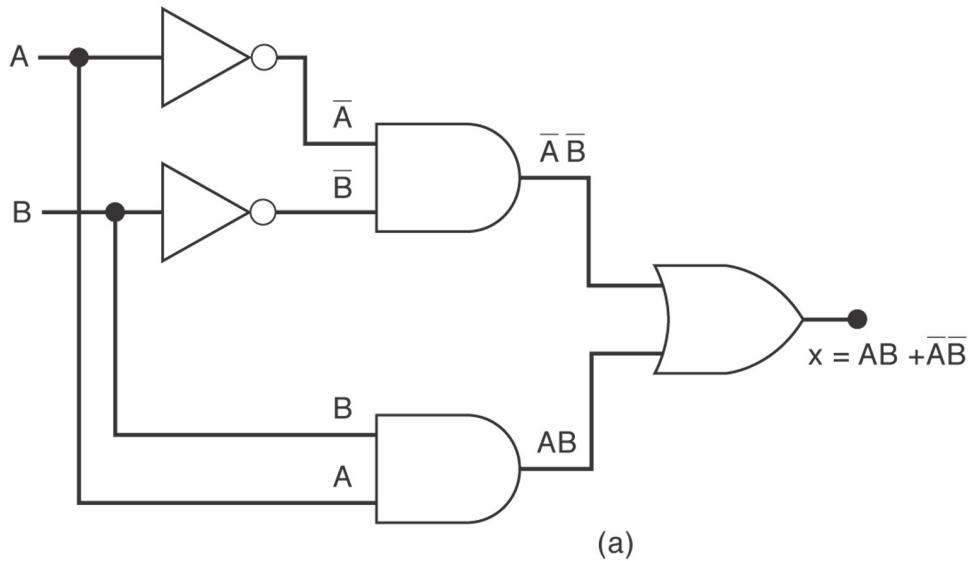
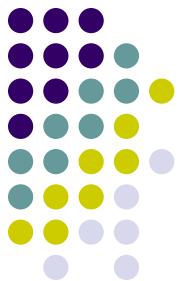
- Exclusive-NOR (XNOR)

$$x = (A'B + AB')'$$



XNOR		
A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

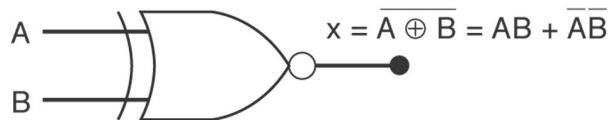
# XNOR



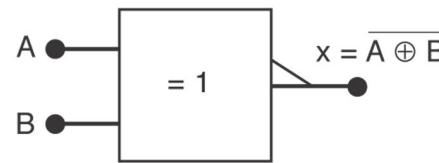
A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

(a)

XNOR gate symbols



(b)

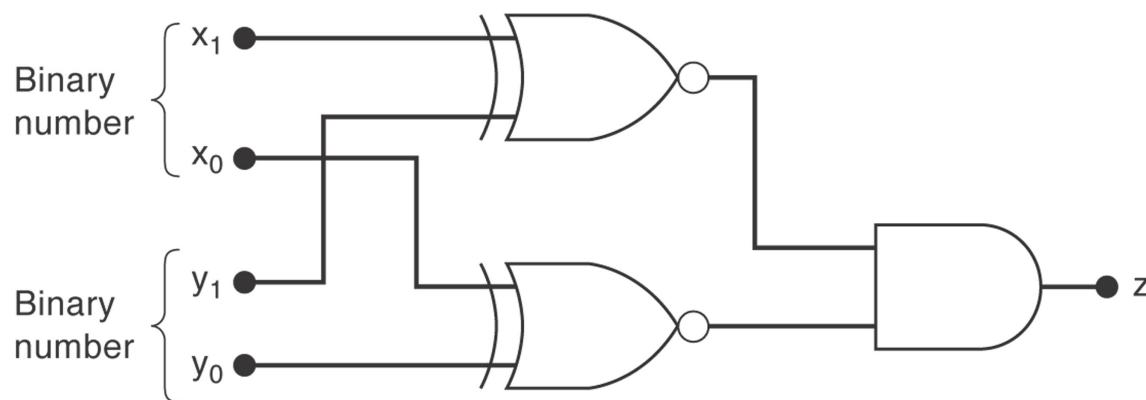


(c)

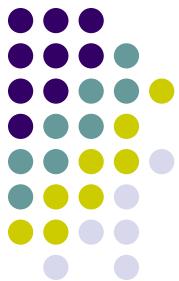


## Example 4-17

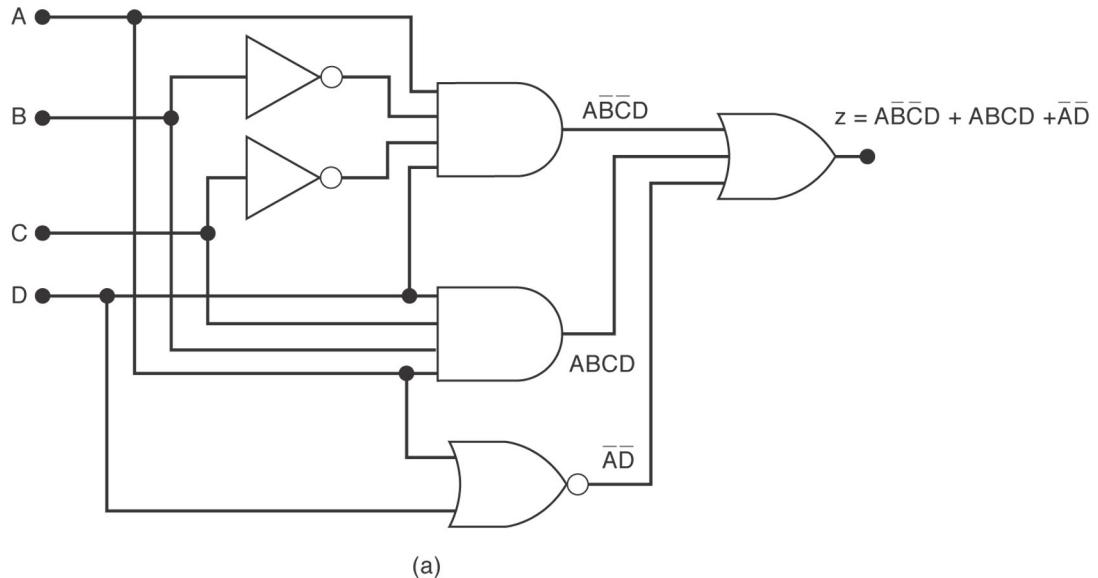
- Design a logic circuit, using  $x_1$ ,  $x_0$ ,  $y_1$  and  $y_0$  inputs, whose output will be HIGH only when the two binary numbers  $x_1x_0$  and  $y_1y_0$  are equal.
- Hint: use XNOR gates ([Figure 4-23](#))



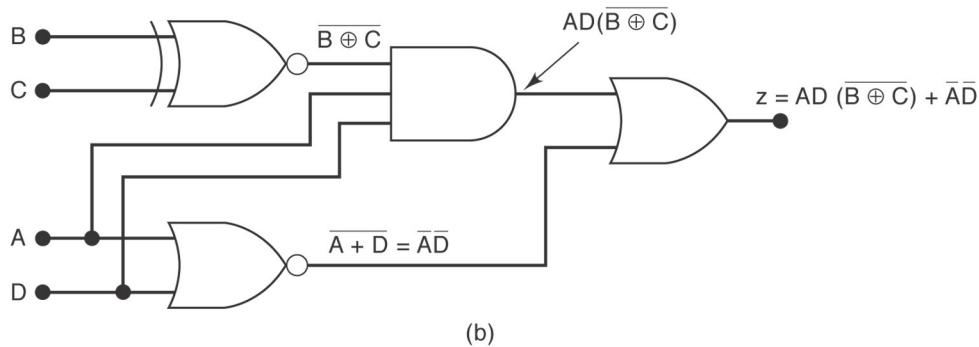
# Using XNOR to Simplify Circuit Implementation



- Example 4-18



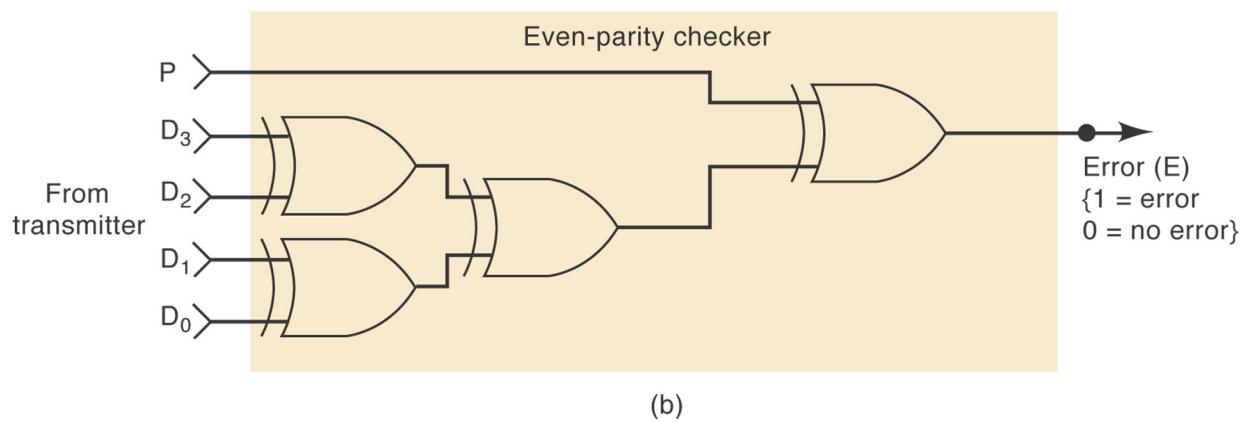
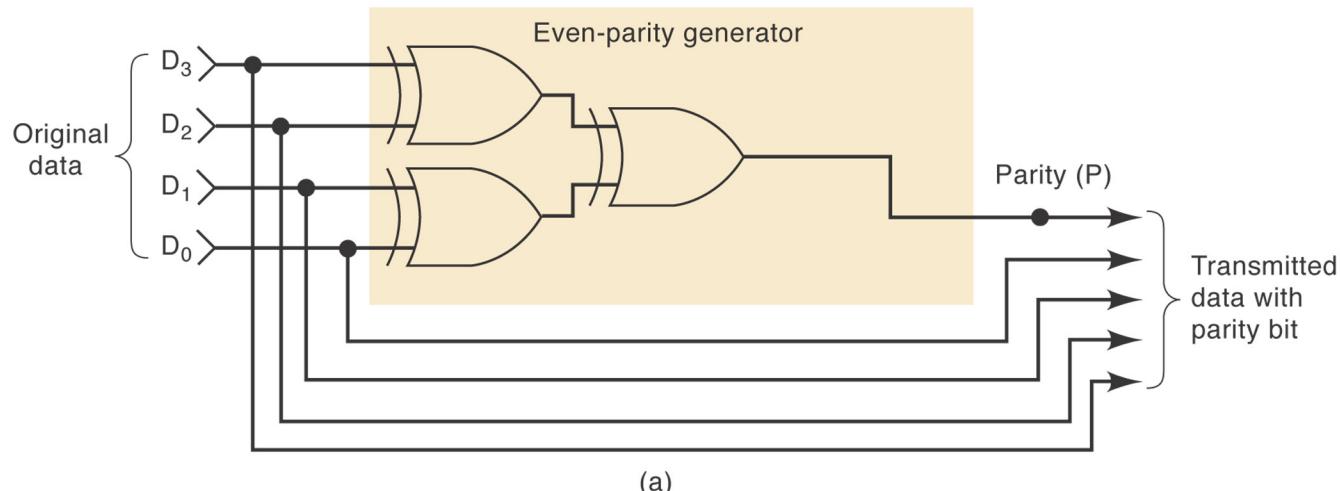
(a)

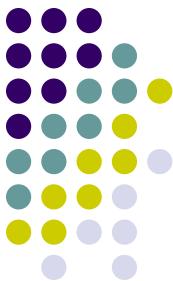


(b)



# Parity Generator and Checker





# Enable/Disable Circuits

- Each of the basic logic gates can be used to control the passage of an input logic signal through to the output.
- A: input, B: control (Figure 4-26)
- The logic level at the control input determines whether the input signal is enabled to reach the output or disabled from reaching the output.



# Figure 4-26

