# Lecture 15
# Analysis of Combinational Circuits

# Designing Combinational Logic Circuits

- A logic circuit having 3 inputs, A, B, C will have its output HIGH only when a majority of the inputs are HIGH.

Step 1 Set up the truth table

Step 2
Write the AND term for each case where the output is a 1.

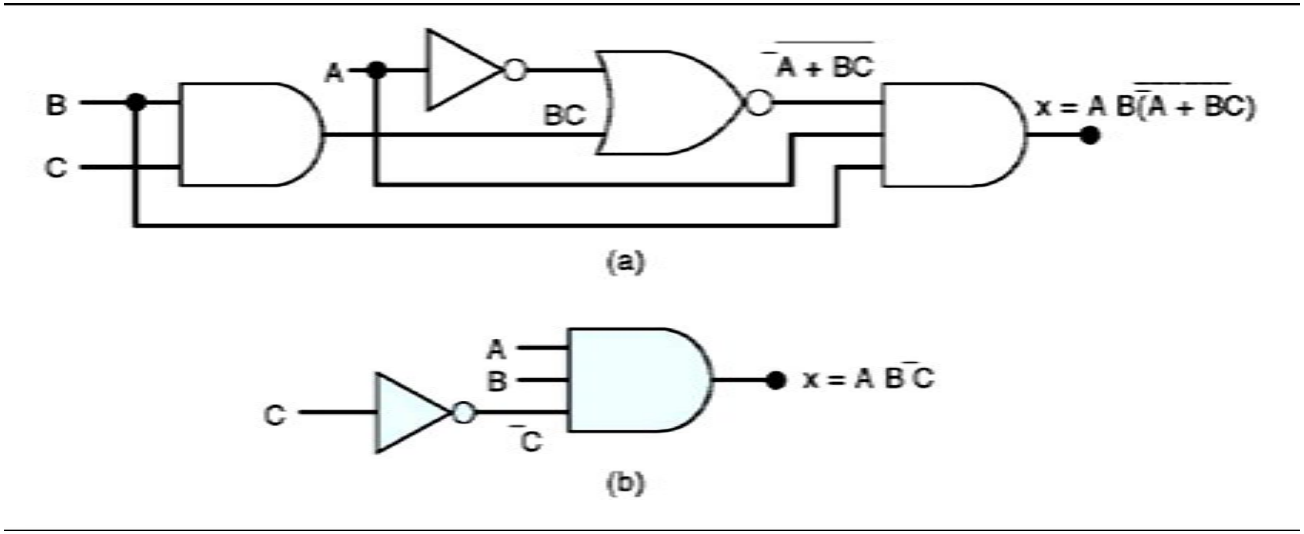
| A | B | C | x | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $\rightarrow \overline{A}BC$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $\rightarrow A\overline{B}C$ |
| 1 | 1 | 0 | 1 | $\rightarrow AB\overline{C}$ |
| 1 | 1 | 1 | 1 | $\rightarrow ABC$ |

# Sum-Of-Products Form

- SOP is useful in simplification and design
- Two or more AND terms OR together
  - –Ex: $ABC+A\overline{B}\overline{C}$
  - –the inversion sign cannot cover more than one variable (ABC)
- Another general form for logic expressions is sometimes used in logic-circuit design. It called product-of-sum (POS)
- Consist 2 or more OR terms that are AND together.
  - Ex: $(A+B\overline{+}C)(A+C)$

# Analysis of Logic Circuits

- First obtain one expression for the circuit, then try to simplify.
- Example:

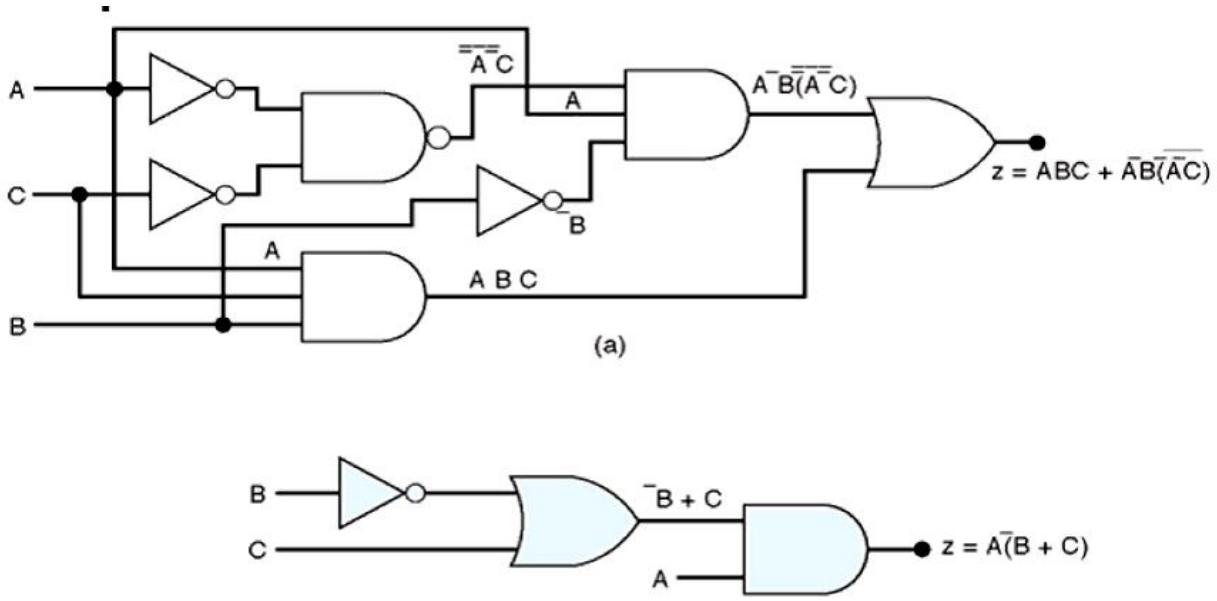



- Two methods for simplifying:
  - Algebraic method (use Boolean algebra theorems)
  - Karnaugh mapping method (systematic, step-by-step approach)

# Algebraic Simplification

1. Put the original expression into SOP form by repeated application of *DeMorgan's theorems*
2. Once in SOP form, check for *common factors* and factor whenever possible.

Example:



Step 3 Write the SOP form the output

Step 4 Simplify the output expression

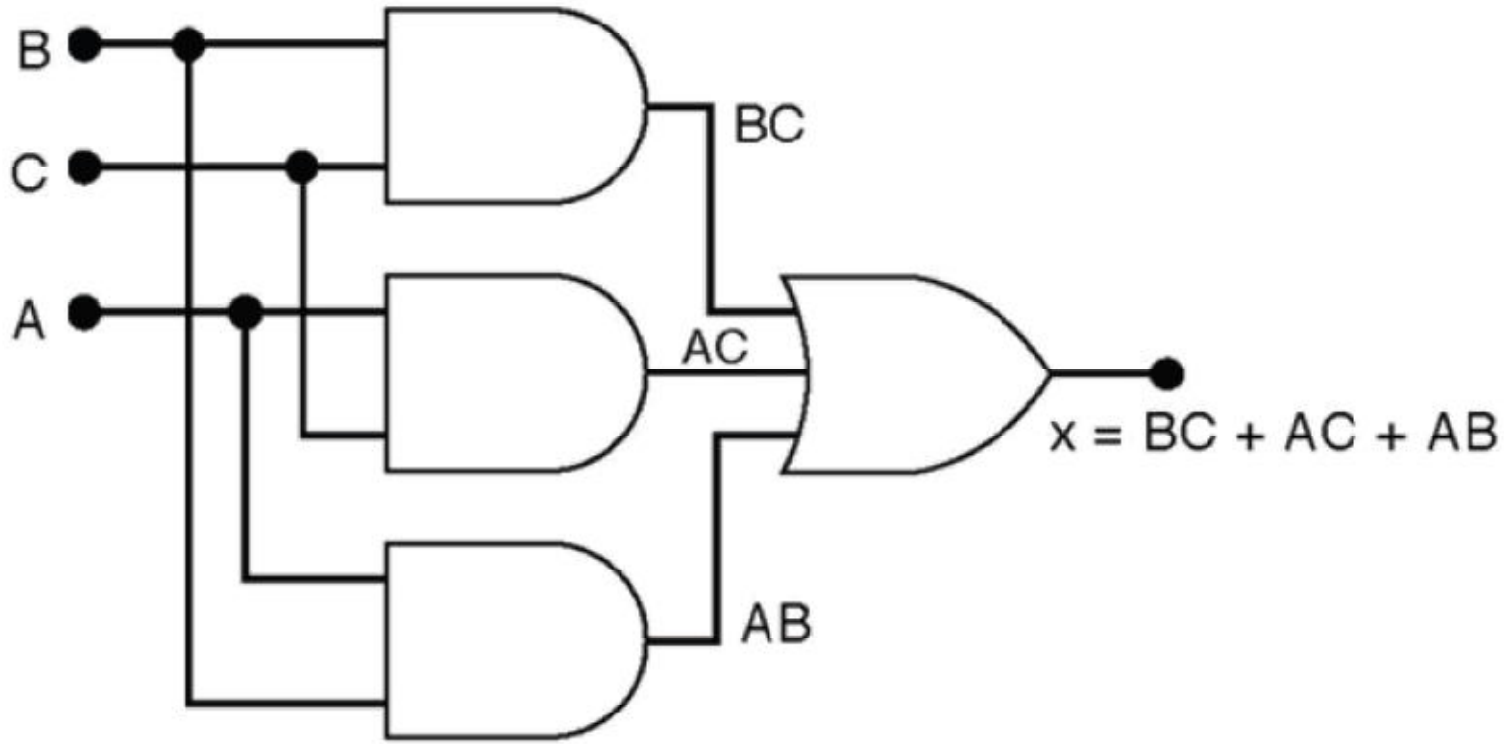
$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$x = \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} + ABC$$

$$= BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

$$= BC + AC + AB$$

Step 5 Implement the circuit



# Karnaugh Map (K-Map) Method

- K Map shows the relationship between inputs & outputs
- Horizontally & vertically adjacent squares differ only in one variable.

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

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

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

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

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

(b)

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

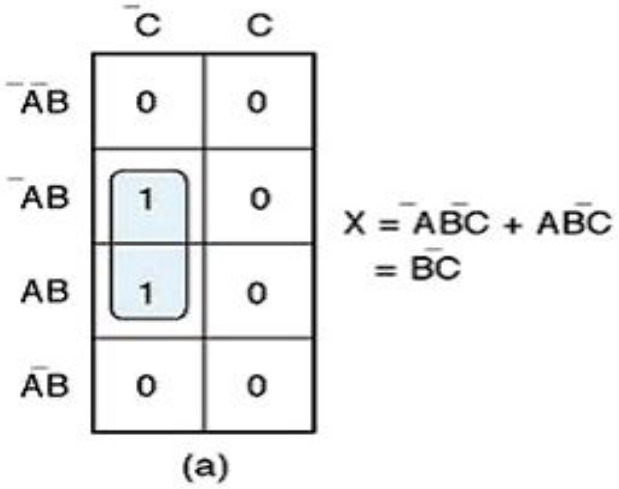
| A | B | C | D | X | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 1 | → $\bar{A}\bar{B}\bar{C}D$ |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 1 | → $\bar{A}B\bar{C}D$ |
| 0 | 1 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 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 | → $AB\bar{C}D$ |
| 1 | 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | 1 | → $ABCD$ |

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

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

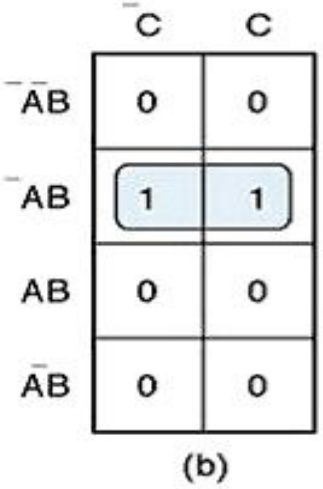
(c)

**Looping is a process combining the squares which contain 1s. The output expression can be simplified by looping.**



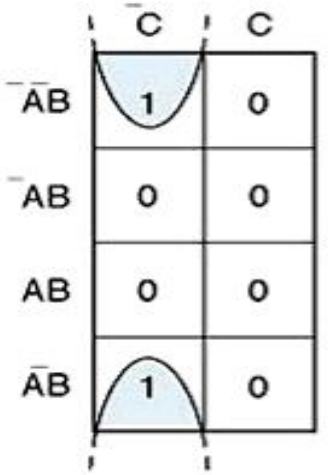

$X = \bar{A}B\bar{C} + AB\bar{C}$
$= B\bar{C}$

(a)



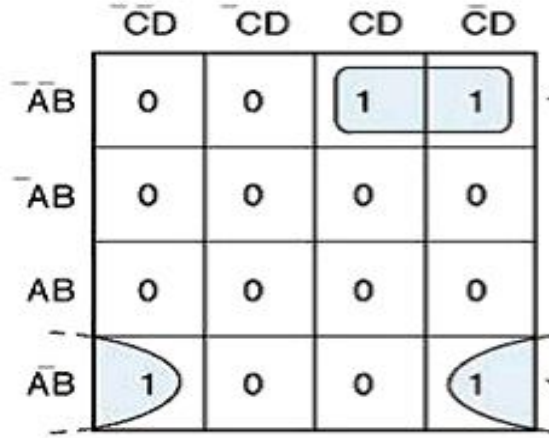

$X = \bar{A}B\bar{C} + \bar{A}BC$
$= \bar{A}B$

(b)




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

(c)

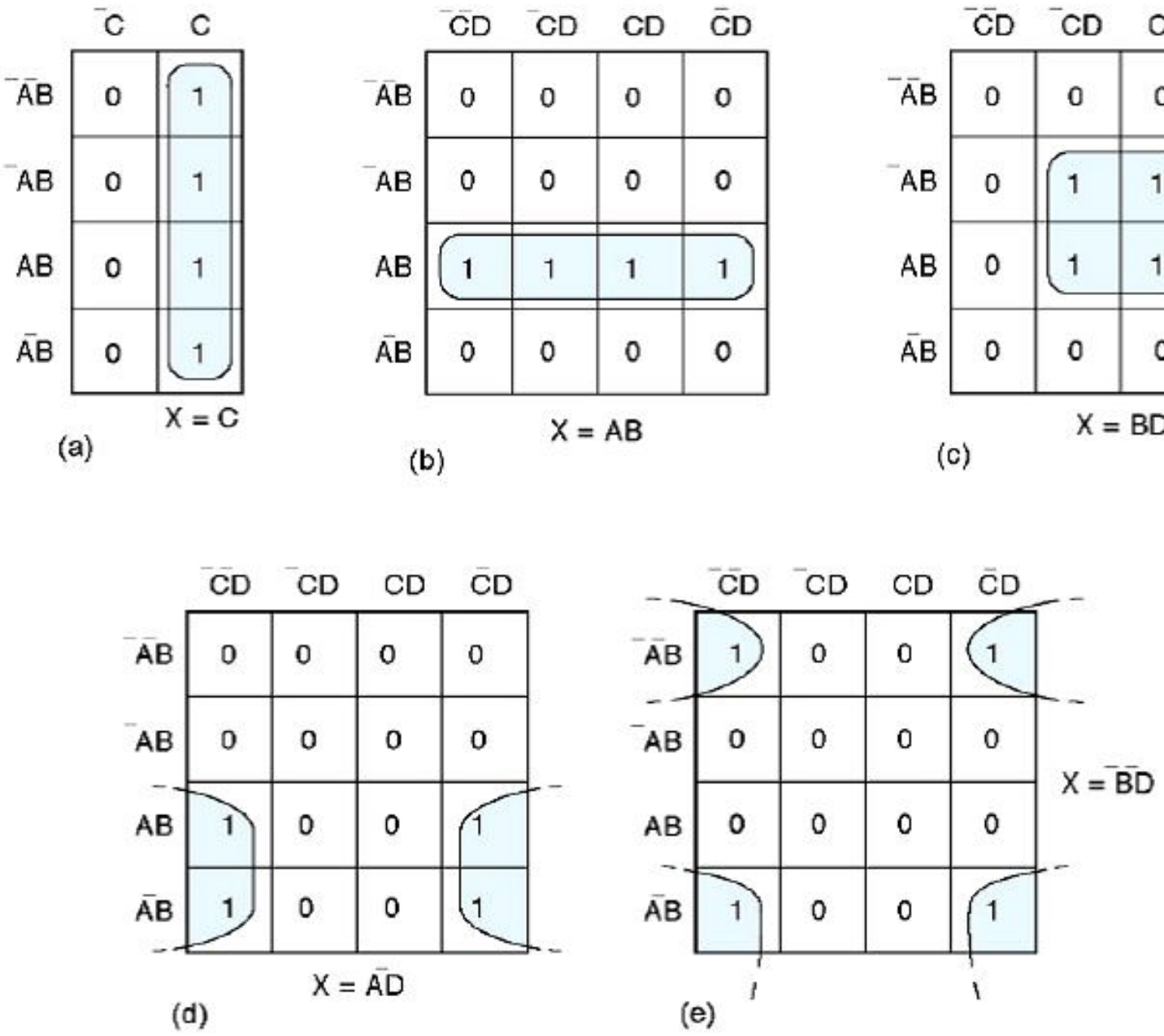


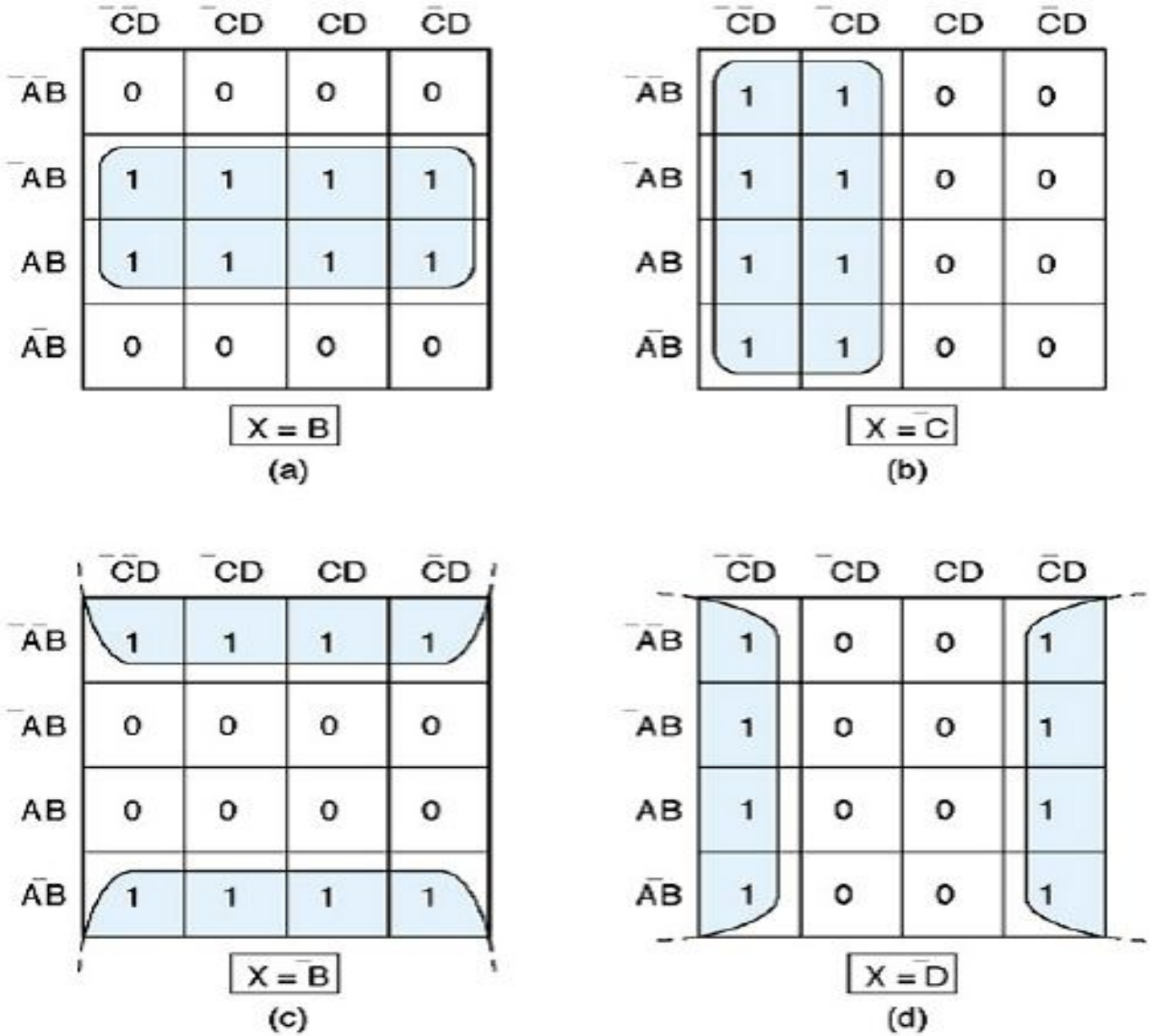

$\bar{A}B\bar{C}$

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

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

(d)


C
C
AB
AB
AB
AB
0
1
X = C
(a)
CD
CD
CD
CD
AB
AB
AB
AB
0
1
X = AB
(b)
CD
CD
CD
CD
AB
AB
AB
AB
0
1
X = BD
(c)
CD
CD
CD
CD
AB
AB
AB
AB
0
1
X = AD
(d)
CD
CD
CD
CD
AB
AB
AB
AB
0
1
X = BD
(e)

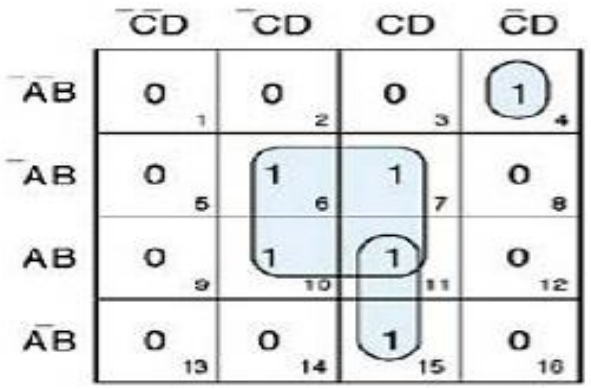

$\bar{C}\bar{D}$ $\bar{C}D$ $CD$ $C\bar{D}$
$\bar{A}\bar{B}$ 0 0 0 0
$\bar{A}B$ 1 1 1 1
$AB$ 1 1 1 1
$A\bar{B}$ 0 0 0 0
$X = B$
(a)
$\bar{C}\bar{D}$ $\bar{C}D$ $CD$ $C\bar{D}$
$\bar{A}\bar{B}$ 1 1 0 0
$\bar{A}B$ 1 1 0 0
$AB$ 1 1 0 0
$A\bar{B}$ 1 1 0 0
$X = \bar{C}$
(b)
$\bar{C}\bar{D}$ $\bar{C}D$ $CD$ $C\bar{D}$
$\bar{A}\bar{B}$ 1 1 1 1
$\bar{A}B$ 0 0 0 0
$AB$ 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
$AB$ 1 0 0 1
$A\bar{B}$ 1 0 0 1
$X = \bar{D}$
(d)

## Rule for loops of any size

When a variable appears in both complemented & uncomplemented form within a loop, that variable is eliminated from the expression. Variables that are the same for all squares of the loop must appear in the final expression.

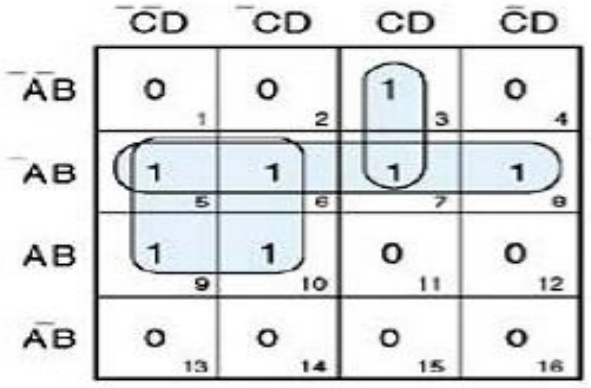
## Complete Simplification Process

1. Construct the K map and place 1s and 0s in the squares according to the truth table.
2. Loop the isolated 1s which are not adjacent to any other 1s. (single loops)
3. Loop any pair which contains a 1 adjacent to only one other 1. (double loops)
4. Loop any octet even if it contains one or more 1s that have already been looped.
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.
6. Loop any pairs necessary to include any 1s that have not yet been looped, making sure to use the minimum number of loops.
7. Form the OR sum of all the terms generated by each loop.



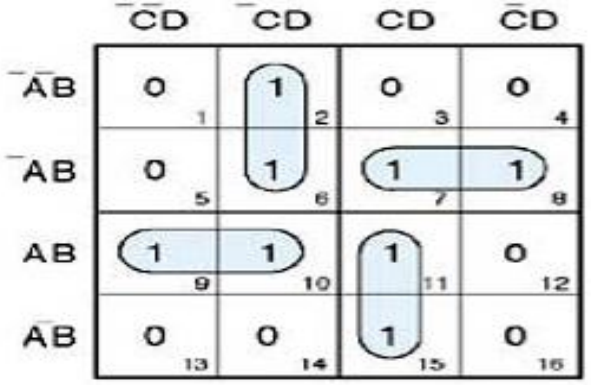

$$X = \underbrace{\bar{A}BC\bar{D}}_{\text{loop 4}} + \underbrace{ACD}_{\text{loop 11, 15}} + \underbrace{BD}_{\text{loop 6, 7, 10, 11}}$$

(a)




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

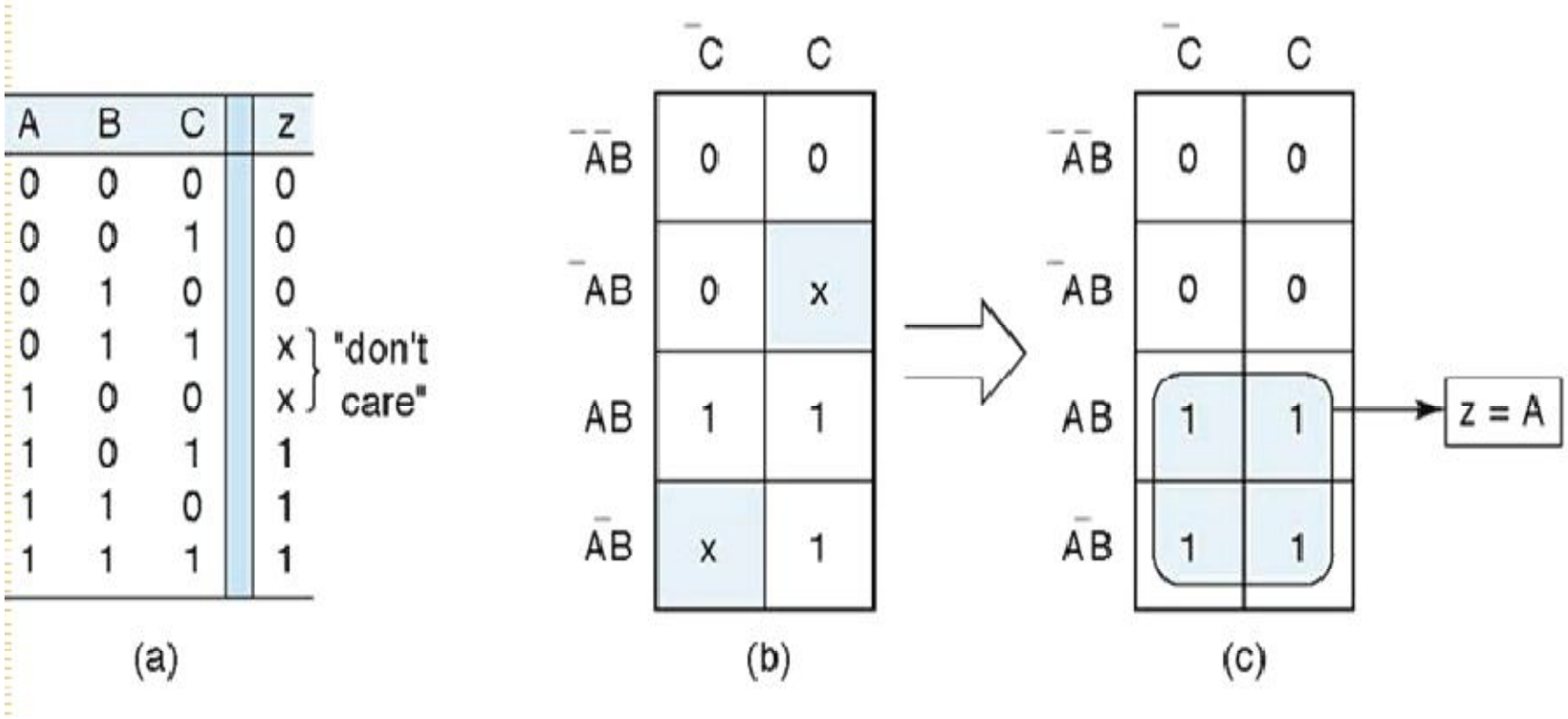
(b)




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

(c)

"Don't-Care" Conditions are certain input conditions for which there are no specified output levels. "Don't-care" conditions should be changed to either 0 or 1 to produce K-map looping that yields the simplest expression.



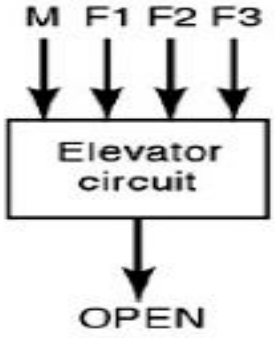
# Filling K-Map from Output Expression

When the desired output is presented as a Boolean expression instead of a truth table, the K map can be filled by using the following steps:

1. Get the expression into SOP form if it is not already so.
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 variables. Place a 0 in all other squares.

- Don’t care condition can come about for several reasons:
  - In some situations certain input combination can never occur and so there is no specified output for these condition.
- Whenever don’t care conditions occur, we must decide which x to change to 0 and which to 1 to produce the best K-map looping (i.e the simplest expression)

# Example




(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)

| | $\overline{F2}\,\overline{F3}$ | $\overline{F2}F3$ | $F2F3$ | $F2\overline{F3}$ |
|---|---|---|---|---|
| $\overline{M}\,\overline{F1}$ | 0 | 1 | X | 1 |
| $\overline{M}F1$ | 1 | X | X | X |
| $MF1$ | 0 | X | X | X |
| $M\overline{F1}$ | 0 | 0 | X | 0 |

(c)

| | $\overline{F2}\,\overline{F3}$ | $\overline{F2}F3$ | $F2F3$ | $F2\overline{F3}$ |
|---|---|---|---|---|
| $\overline{M}\,\overline{F1}$ | 0 | 1 | 1 | 1 |
| $\overline{M}F1$ | 1 | 1 | 1 | 1 |
| $MF1$ | 0 | 0 | 0 | 0 |
| $M\overline{F1}$ | 0 | 0 | 0 | 0 |

$OPEN = \overline{M}\,(F1 + F2 + F3)$

(d)

# Example

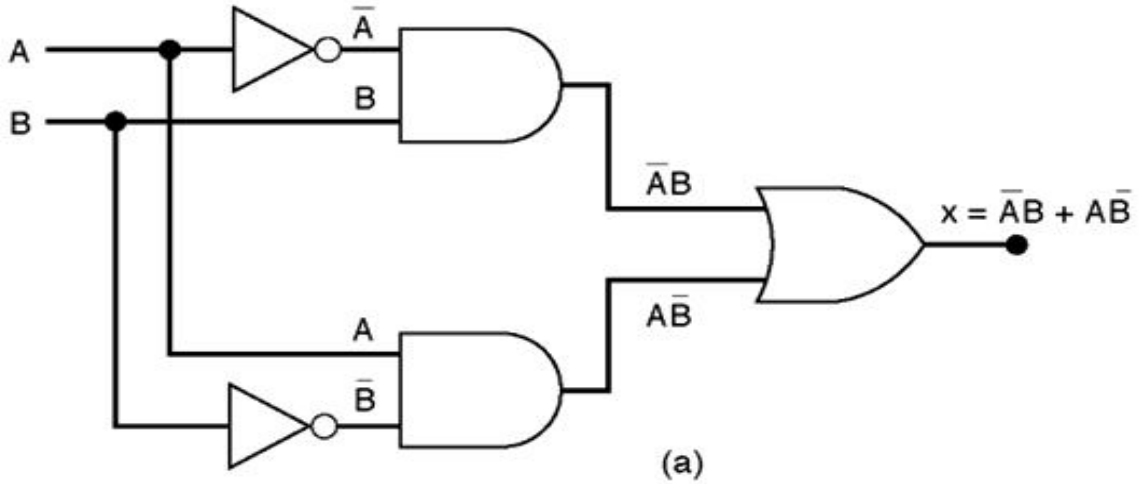
A BCD counter produces a four bit output representing the BCD code for the number of pulses hat have been applied to the counter input. For example, after 4 pulses have occurred, the counter outputs are DCBA= 01002 = 410. The counter resets to 0000 on the tenth pulse and starts counting over again. In other words, the DCBA output will never represent a number greater than 10012=910. Design the logic circuit that produces a HIGH output whenever the count is 2,3, or 9. Use K mapping and take advantage of the don’t care conditions.

## Summary

- Compared to the algebraic method, the K-map process is a more orderly process requiring fewer steps and always producing a minimum expression.
- For the circuits with large numbers of inputs (larger than four), other more complex techniques are used.

# Exclusive-OR and Exclusive-NOR Circuits

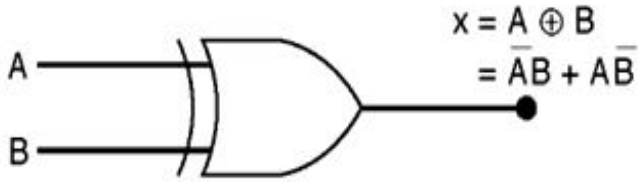
Exclusive-OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.



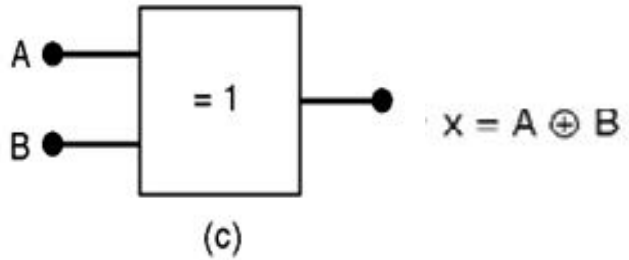

| A | B | x |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(a)

XOR gate symbols

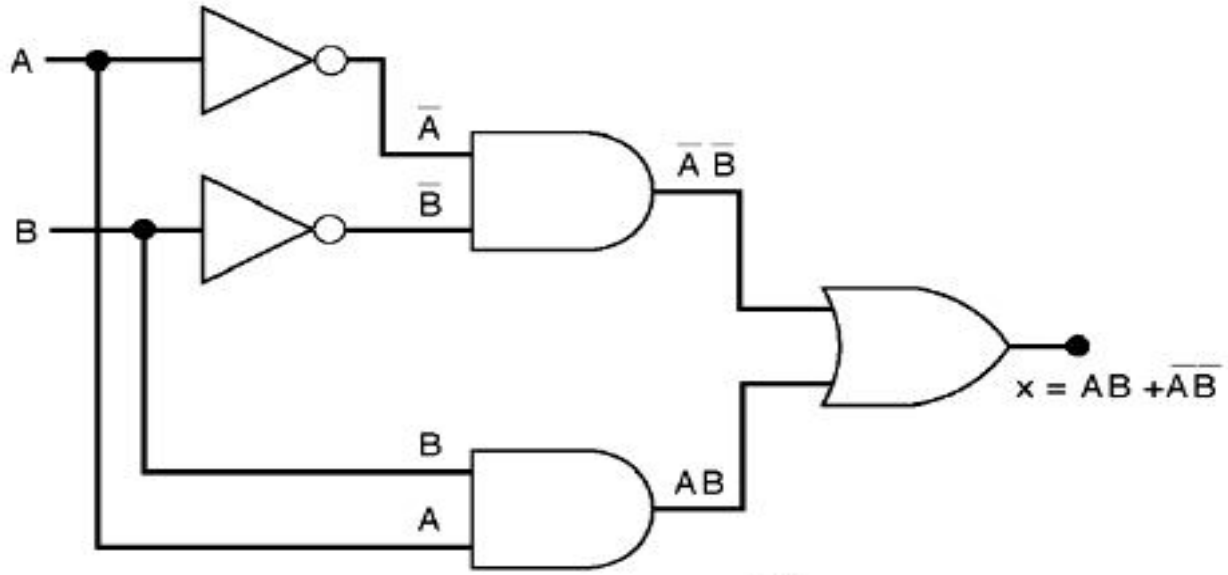



(b)




(c)

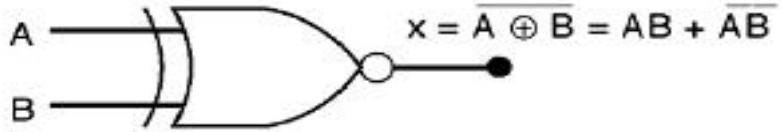
Exclusive-NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.



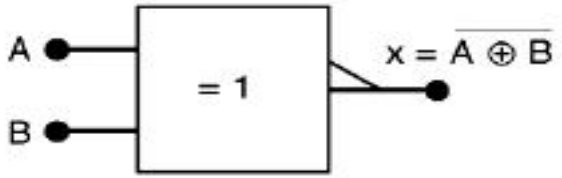

(a)

| A | B | x |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

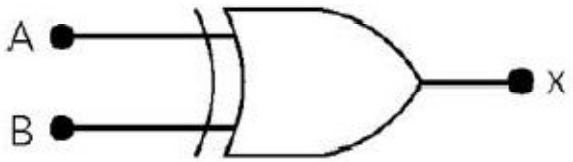
XNOR gate symbols

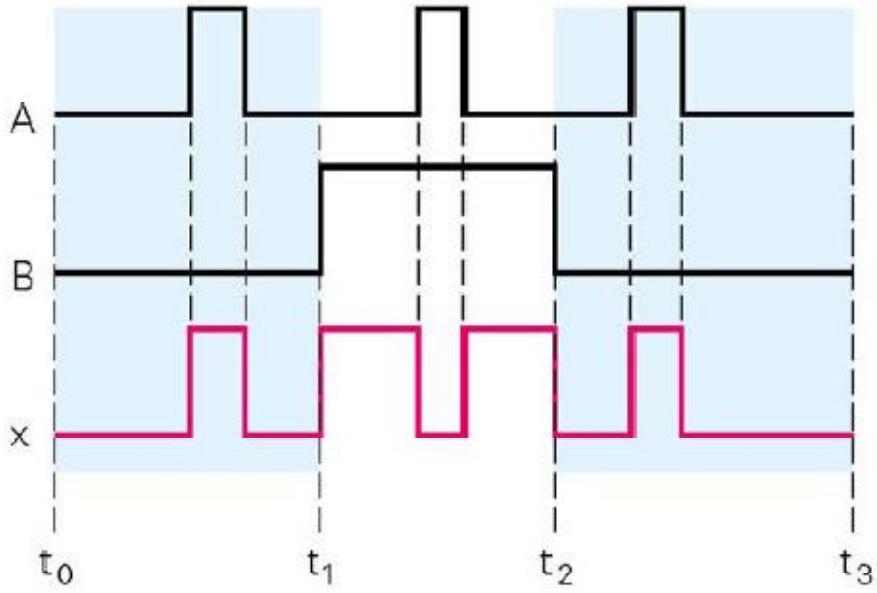



(b)

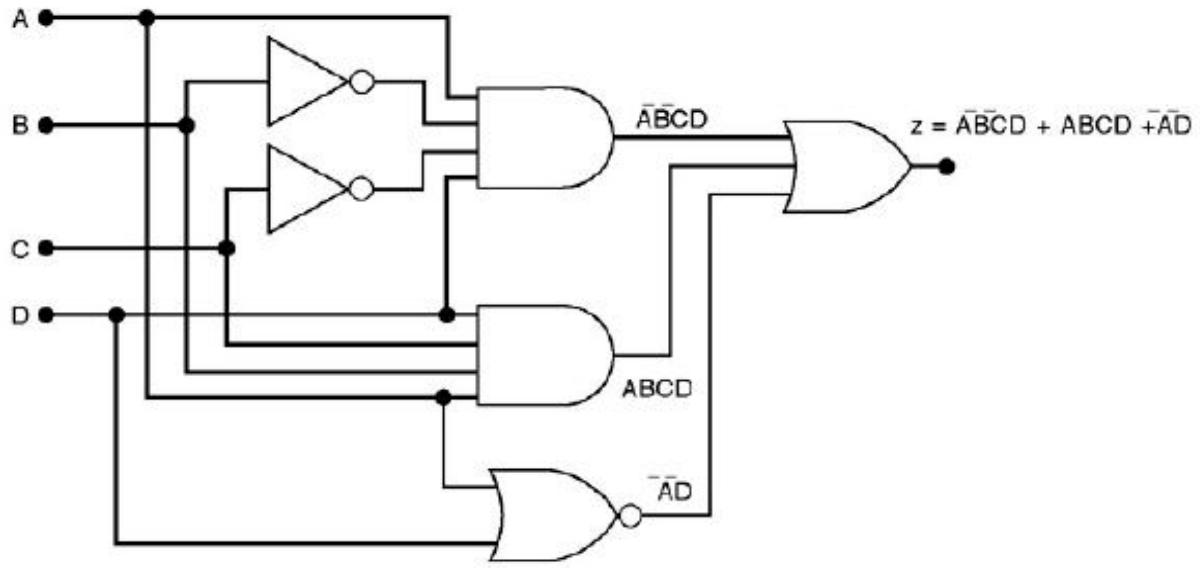



(c)

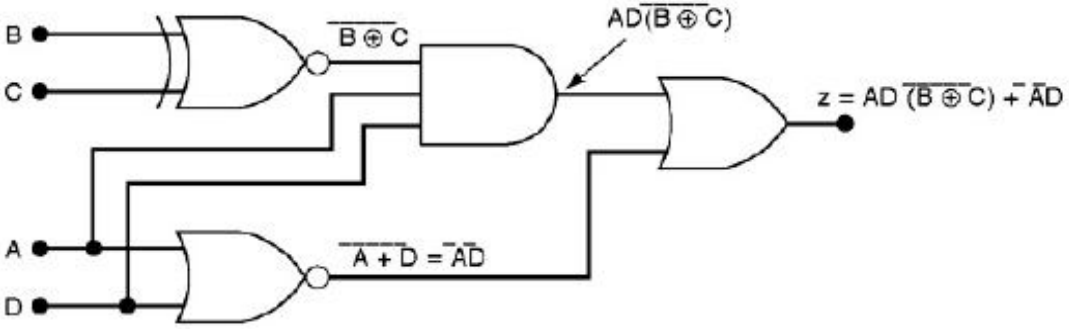

A
B
x


A
B
x
$t_0$
$t_1$
$t_2$
$t_3$

# XNOR gate may be used to simplify circuit implementation.

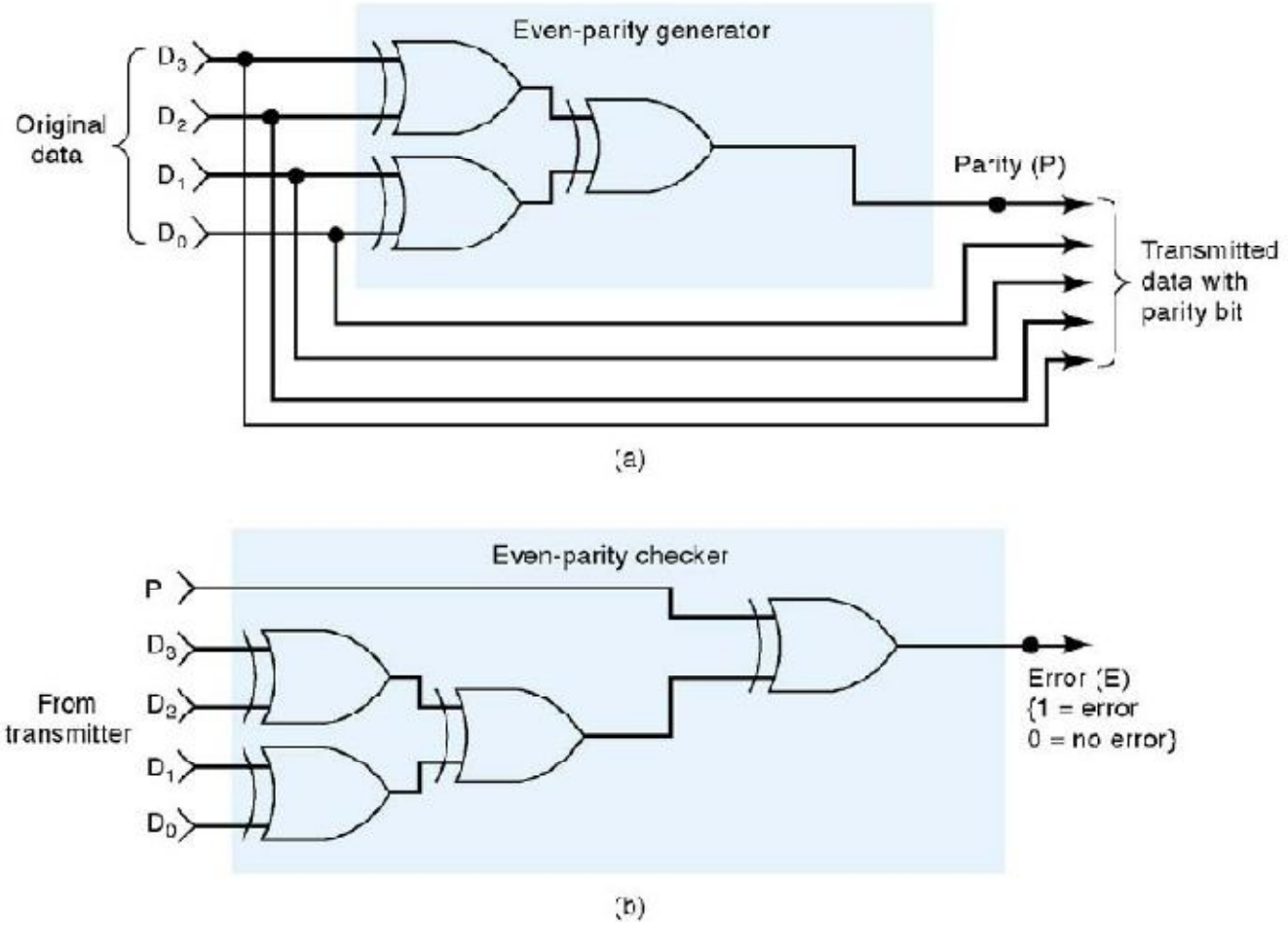



(a)



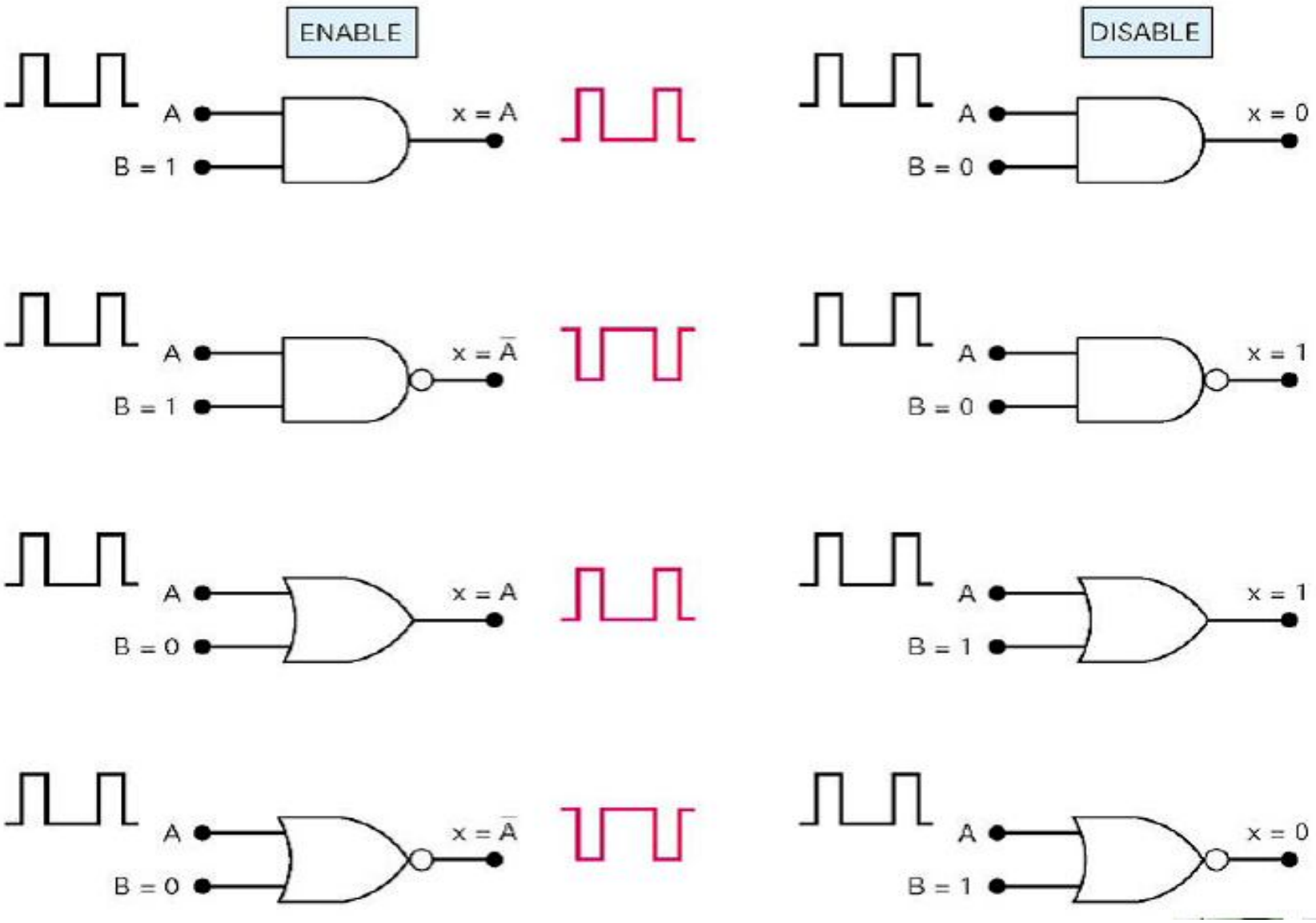

(b)

# Parity Generator and Checker



- A transmitter can attach a parity bit to a set of data bits before transmitting the data bits to a receiver. The receiver will detect any single bit errors that may have occurred during the transmission.
- In figure (a) the set of data to be transmitted is applied to the parity-generator circuit, which produces the even-parity bit, P, at its output. This parity bit is transmitted to the receiver along with the original data bits, making a total of five bits.
- In figure (b) these five bits (data+parity) enter the receiver's parity-checker circuit, which produces an error output, E that indicates whether or not a single-bit error has occurred.

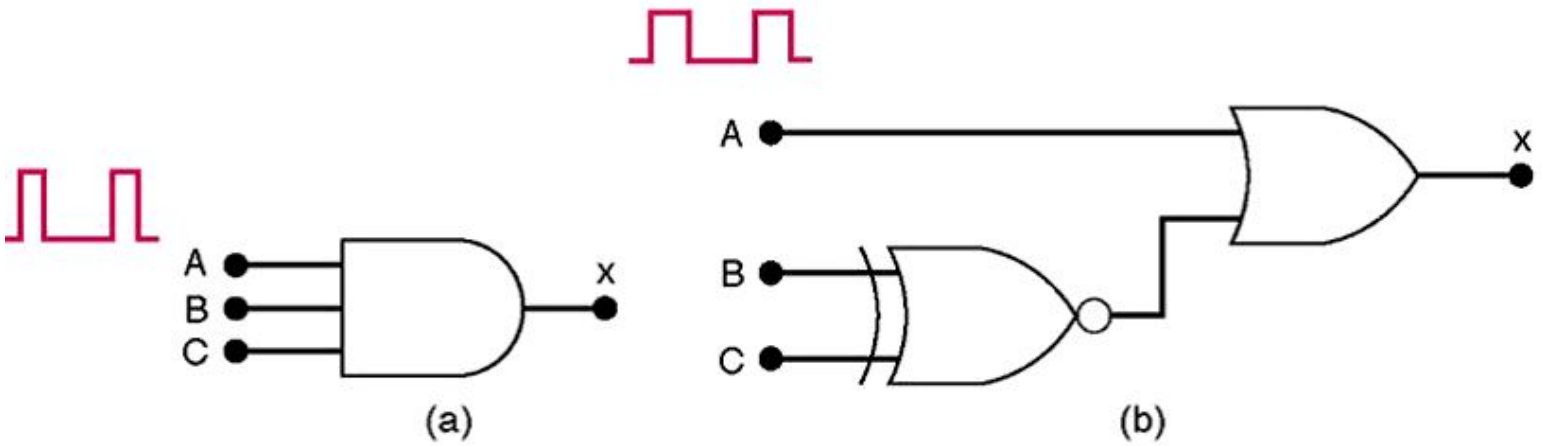
# Enable/Disable Circuits



# Enable/Disable Circuits cont.

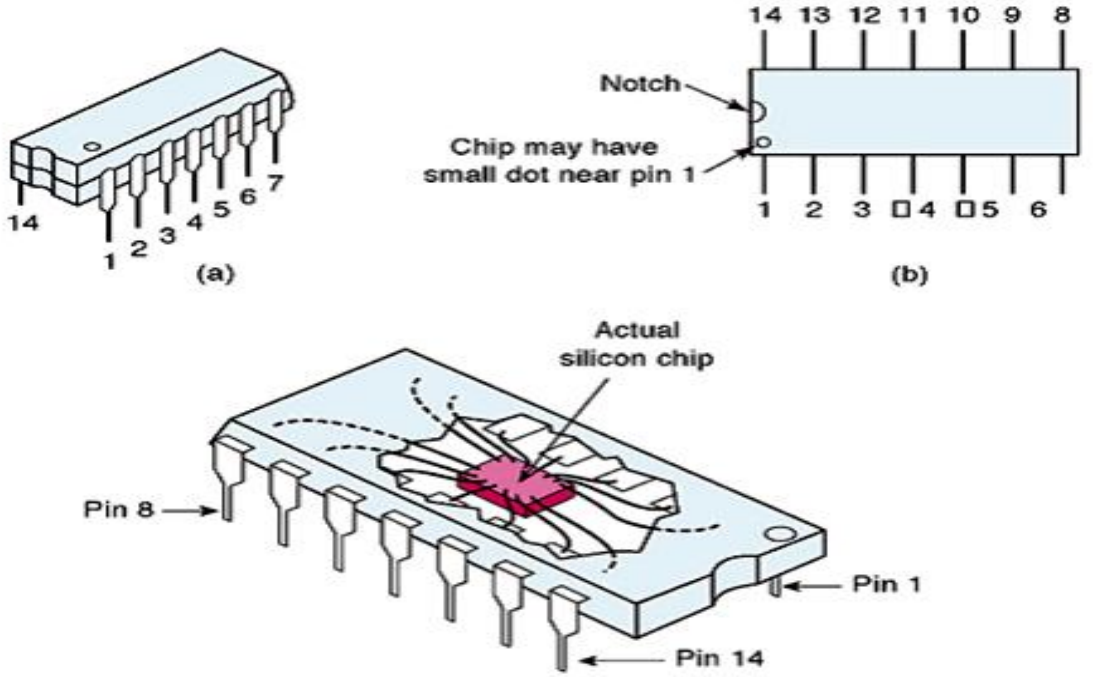
Ex. 1(Fig.a): Design a logic circuit that will allow a signal to pass to the output only when control inputs B and C are both HIGH; otherwise, the output will stay LOW.

Ex. 2(Fig.b): Design a logic circuit that will allow a signal to pass to the output only when one, but not both, of the control inputs are HIGH; otherwise, the output will stay LOW.



# Basic Characteristics of Digital ICs

- Digital ICs (chips): a collection of resistors, diodes and transistors fabricated on a single piece of semiconductor materials called substrate.
- Dual-in-line package (DIP) is a common type of packages. It contains two parallel rows of pins.



- Digital ICs are often categorized according to their circuit complexity as measured by the number of equivalent logic gates on the substrates. 6 levels of complexity:

  SSI, MSI, LSI,VLSI, ULSI,GSI.

- SSI – having a small number of gates

# Assignment - 15

Design a logic circuit that controls an elevator door in a three-story building. The circuit has 4 inputs. M is a logic signal that indicates when the elevator is moving (M=1) or stop (M=1).F1, F2, F3 are floor indicator signals

that are normally LOW and they go HIGH only when the elevator is positioned at the level of that particular floor. For example, when the elevator is lined up level with the second floor, F2=1 and F1=F3=0. The circuit output is the open signal which is normally LOW and is to go HIGH when the elevator door is to be opened.