3 Logic Circuit Analysis and Design: Circuit Minimization.

Analyzing Logic Circuits

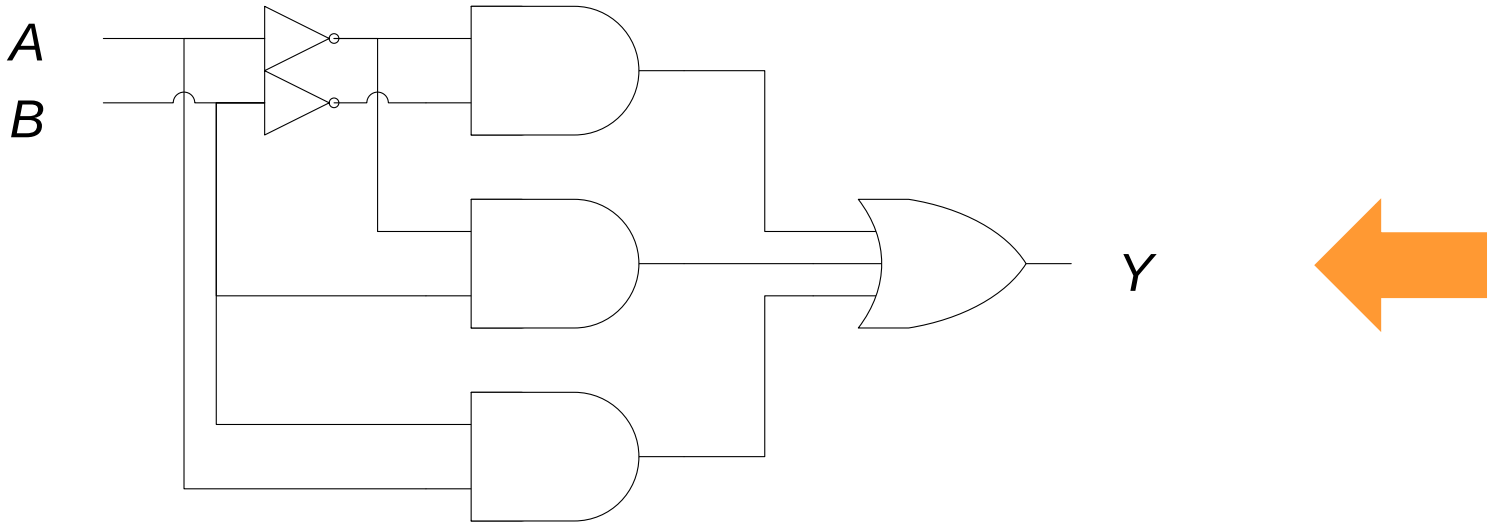
1. From the given logic circuit, write the equation of the circuit;
2. Write the truth table from the equation above;
3. Summary of the operation of the circuit:

- What conditions that provide the output of ‘1’
- What conditions that provide the output of ‘0’

Note:

If the given circuit is connected to another circuit,

we need to know what would happen to that circuit, too.



It can be expressed as: $Y = \overline{A}\ \overline{B} + \overline{A}\ B + A\ B$

A	B	$\overline{A}\ \overline{B}$	$\overline{A}\ B$	$A\ B$	Y
0	0	1	0	0	1
0	1	0	1	0	1
1	0	0	0	0	0
1	1	0	0	1	1

Designing Logic Circuits

From the desired output, write the corresponding Truth Table;

- Assign the number of Inputs and Outputs;
- Find the relationship between Inputs and Outputs for all cases;
- Write the Truth Table.

Write a Boolean Equation from that Truth Table;

- In the configuration of either Sum of Product or Product of Sum.

Designing Logic Circuits

- The Boolean Equation comes from the Truth Table in the configuration of SOP (or POS),

  is **NOT** the shortest equation.

- Sometimes the Boolean Equation could be able to be combined several terms in its equation.

- The better equation need to be shorter,

  or has less terms and less number of variables;

- Consequently, the corresponding circuit would be more economical.

Designing Logic Circuits

If we want to reduce the number of gates in the circuit, we need to minimize the output Equation by

- To have the least number of terms, or
- To have the least number of variables in the terms, or
- To arrange the term(s) in equation to the appropriate format/gate, repeatedly.

So how could we obtain the SHORTER equation?

Designing Logic Circuits

We can minimize a logical Equation by 3 methods:

- Boolean Algebra Method.
- Karnaugh Map Method
- Quine-McCluskey Method

Boolean Algebra

- Use Algebraical Rules to reduce the terms in the logical equations;
- The principle was proposed by George Boole;
- This method is so-called as Boolean Algebra;
  - ➔ The theory of logical relationships;
  - ➔ Using basic operators: NOT, AND (·) and OR (+)
  - ➔ The principle has logical relationships, as follows.

Boolean Algebra

The Boolean algebra consists of basic **11** relationships:

$$\overline{\overline{A}} = A$$

$$A \cdot A = A \qquad A + A = A$$

$$A \cdot 0 = 0 \qquad A \cdot 1 = A$$

$$A + 0 = A \qquad A + 1 = 1$$

$$A + \overline{A} = 1 \qquad A \cdot \overline{A} = 0$$

$$A \cdot B + A \cdot C = A \cdot (B + C) \qquad A + \overline{A} \cdot B = A + B$$

Boolean Algebra

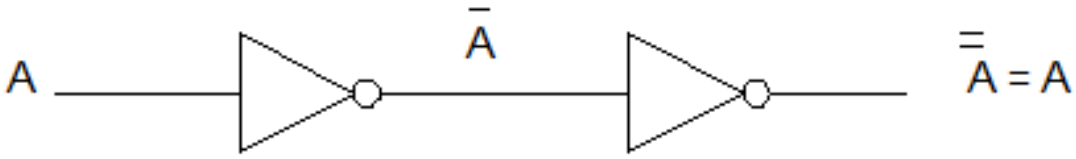
1. Inverse

Give the output with the opposite logic to the input:

If $A = 0$, then $\overline{A} = 1$ and $\overline{\overline{A}} = 0$

And $A = 1$, then $\overline{A} = 0$ and $\overline{\overline{A}} = 1$

Then $\overline{\overline{A}} = A$

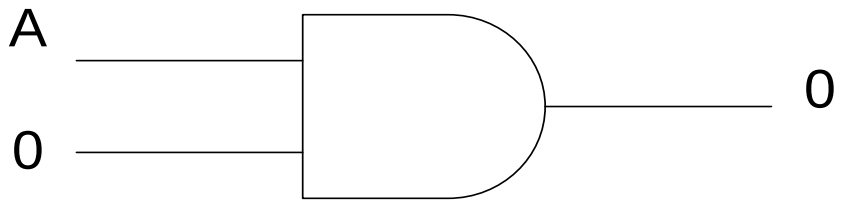


Boolean Algebra

2. AND with 0: $A \cdot 0 = 0$

Put Zero to the input of AND gate, make the output to be Zero.

- Force the output of the AND gate to be Zero.



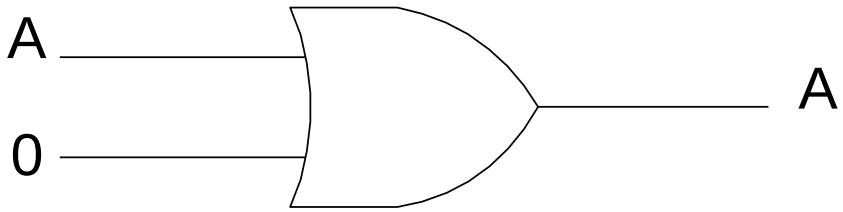
Boolean Algebra

3. OR with 0: $A + 0 = A$

If $A = 1$, then the output will be 1.

And if $A = 0$, then the output will be 0.

- Use to drive the next circuit by the OR gate.



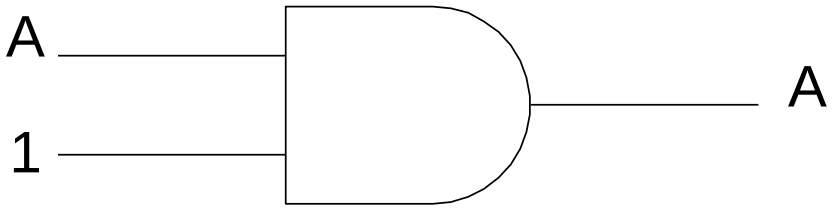
Boolean Algebra

4. AND with 1: $A \cdot 1 = A$

If $A = 1$, then the output will be 1.

And if $A = 0$, then the output will be 0.

- Use to drive the next circuit by the AND gate.

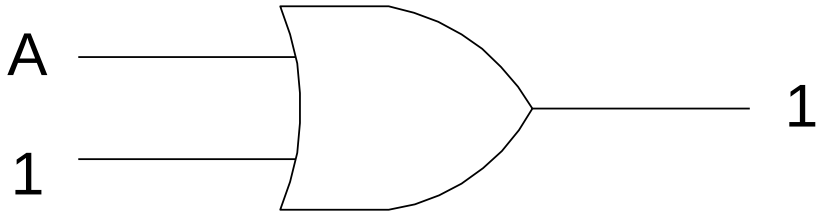


Boolean Algebra

5. OR with 1: $A + 1 = 1$

Put One to the input of OR gate, make the output to One, too.

- Force the output of the OR gate to One/High state.



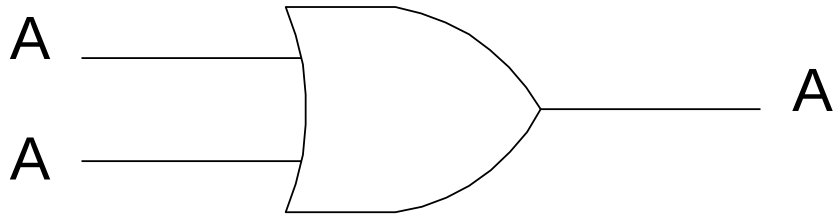
Boolean Algebra

6. OR with itself: $A + A = A$

Tie the signal at the input of the OR gate,

the output will produce that signal.

- Reproduce the same signal at the output by the OR gate.



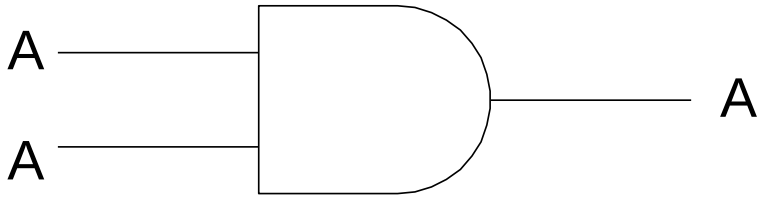
Boolean Algebra

7. AND with itself: $A \cdot A = A$

Tie the signal at the input of the AND gate,

the output will produce that signal.

- Reproduce the same signal at the output by the AND gate.



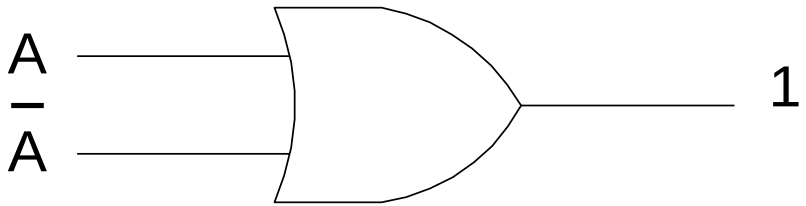
Boolean Algebra

8. OR of the Complement Pair: $A + \overline{A} = 1$

If $A = 1$, so $\overline{A} = 0$ then the output will be 1.

And if $A = 0$, so $\overline{A} = 1$ then the output will be 1.

➢ OR of the Complement Pair always produces 1



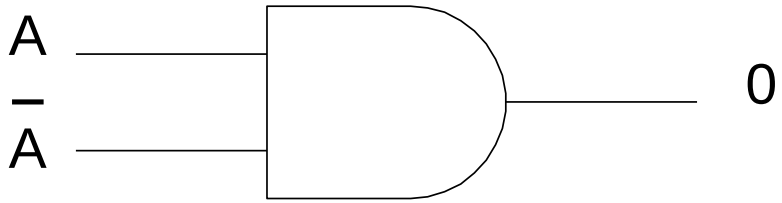
Boolean Algebra

9 AND of the Complement Pair: $A \cdot \overline{A} = 0$

If $A = 1$, so $\overline{A} = 0$ then the output will be 1.

And if $A = 0$, so $\overline{A} = 1$ then the output will be 1.

➢ AND of the Complement Pair always produces 0



Boolean Algebra

10. Association: $A \cdot B + A \cdot C = A \cdot (B + C)$

Proved by the truth table below:

A	*B*	*C*	*AB*	*AC*	***AB+AC***	*B+C*	***A·(B+C)***
0	0	0	0	0	**0**	0	**0**
0	0	1	0	0	**0**	1	**0**
0	1	0	0	0	**0**	1	**0**
0	1	1	0	0	**0**	1	**0**
1	0	0	0	0	**0**	0	**0**
1	0	1	0	1	**1**	1	**1**
1	1	0	1	0	**1**	1	**1**
1	1	1	1	1	**1**	1	**1**

Boolean Algebra

11. $A + \overline{A} \cdot B = A + B$

Proved by the truth table below:

A	B	$\overline{A}$	$\overline{A} \cdot B$	$A + \overline{A} \cdot B$	$A + B$
0	0	1	0	**0**	**0**
0	1	1	1	**1**	**1**
1	0	0	0	**1**	**1**
1	1	0	0	**1**	**1**

De’ Morgan Theory

De Morgan's Theory

- Used to convert the logic terms between **(N)AND** and **(N)OR**;

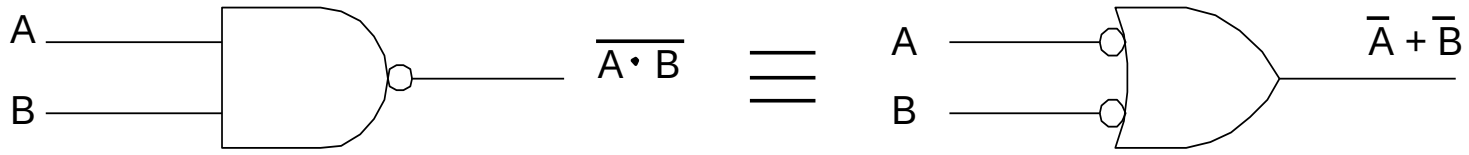
$$\overline{A \cdot B} = \overline{A} + \overline{B} \qquad \text{NAND} \leftrightarrow \text{OR}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B} \qquad \text{NOR} \leftrightarrow \text{AND}$$

De' Morgan Theory

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

A	B	$A \cdot B$	$\overline{A \cdot B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	0	**1**	1	0	**1**
1	0	0	**1**	0	1	**1**
1	1	1	**0**	0	0	**0**

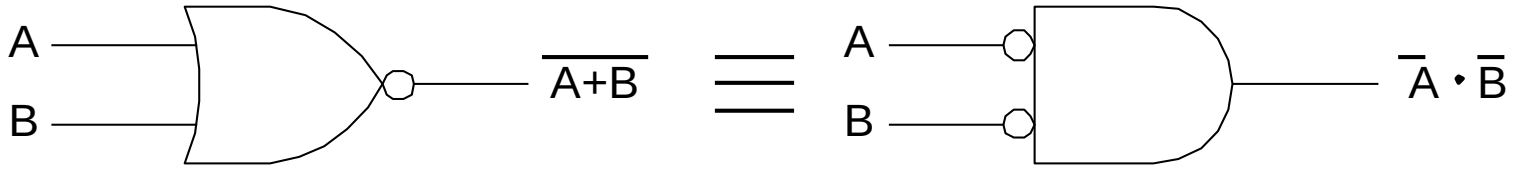


NAND of 2 inputs = OR with NOT inputs

De' Morgan Theory

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

A	B	$A \cdot B$	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	1	0	**0**
1	0	1	**0**	0	1	**0**
1	1	1	**0**	0	0	**0**



NOR of 2 inputs = AND with NOT inputs

Minimization by Boolean Algebra

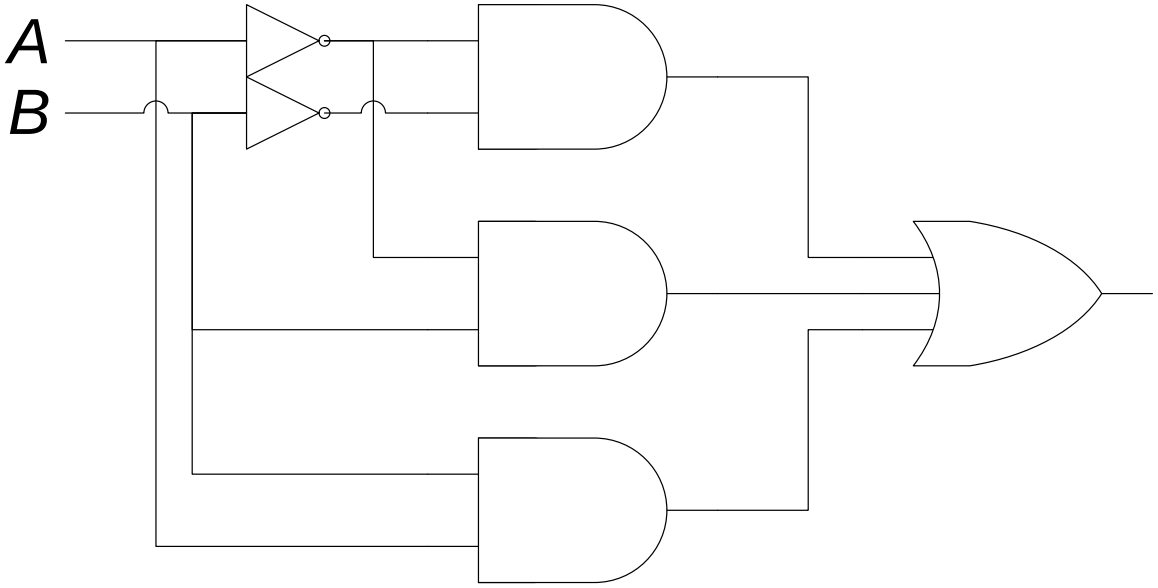
Minimize the obtained equation by the 11 relationships in the Boolean Algebra:

- To get less terms; [or Number of gates]
- To get less operators/variables; [or Number of gates]
- To get the desired gates. [use ONLY NANDs/NORs]

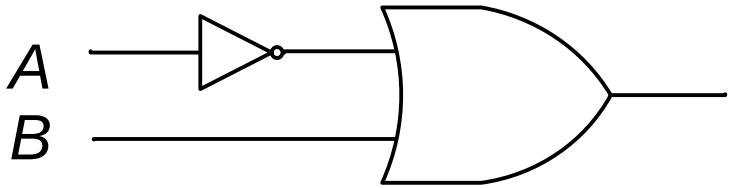
(Using De' Morgan) (Used in IC Design Fabrication)

Minimization by Boolean Algebra

Example: Minimize the circuit below



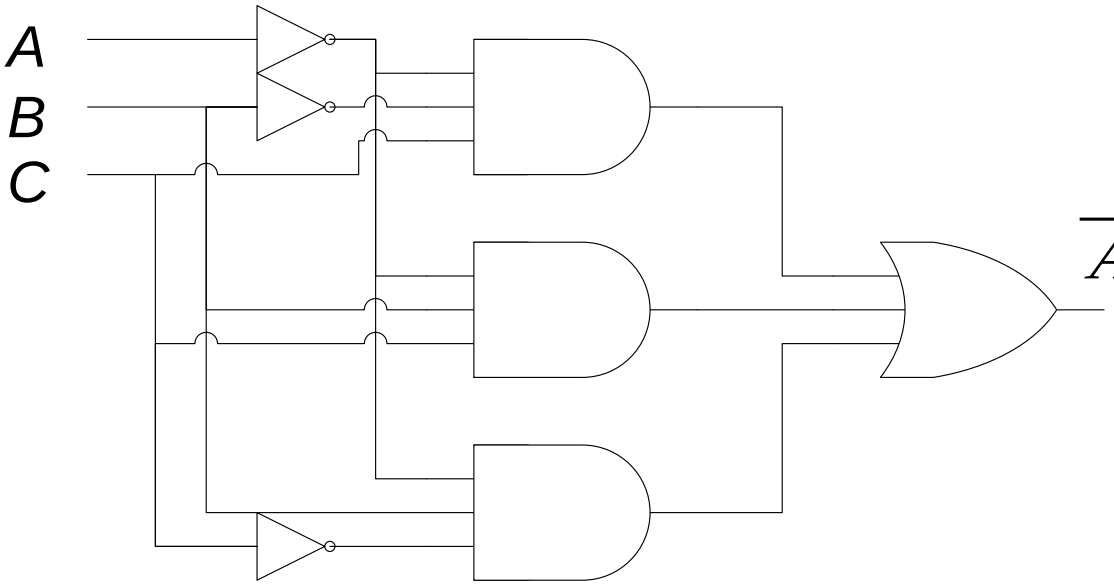
$$Y = \overline{A}\,\overline{B} + \overline{A}\,B + A\,B$$



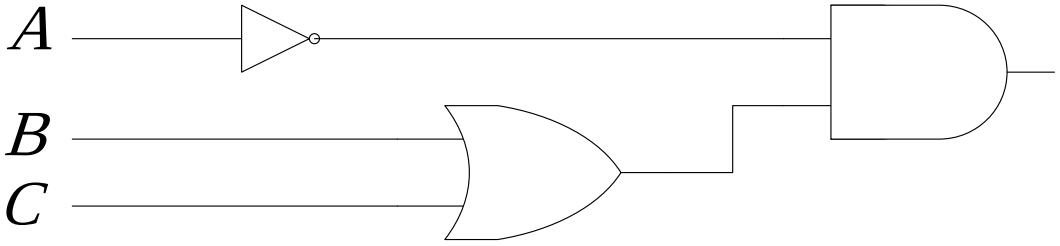
$$Y = \overline{A} + B$$

Minimization by Boolean Algebra

Example: Minimize the circuit below



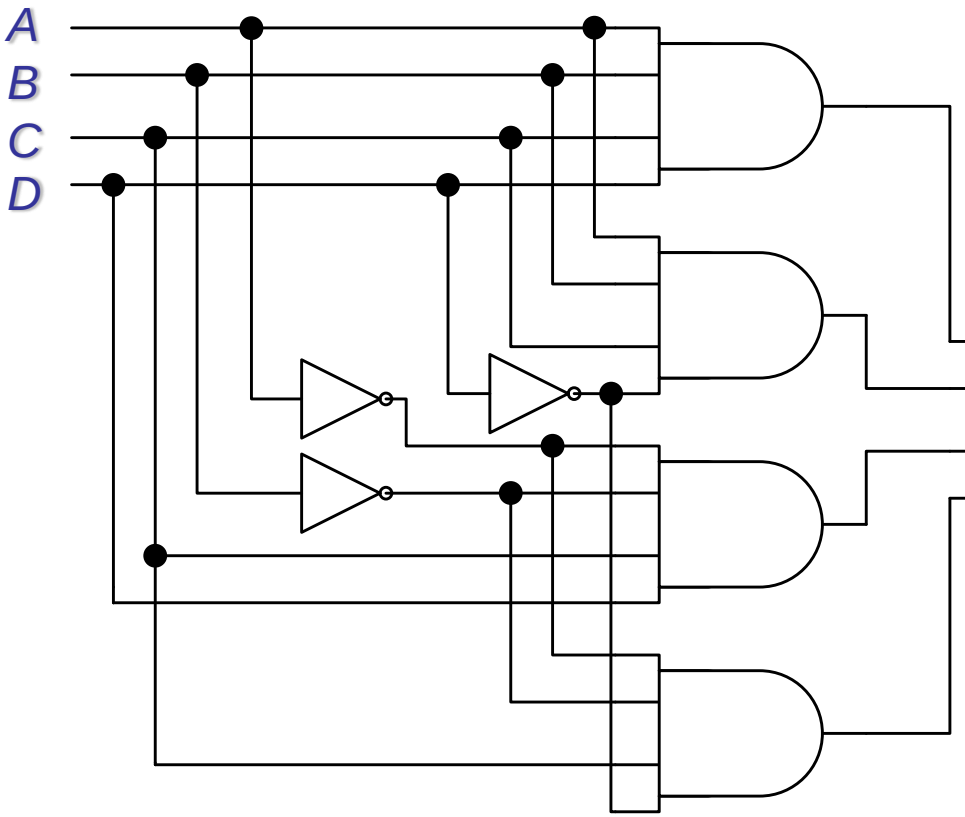
$$\overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C}$$



$$\overline{A}(B + C)$$

Minimization by Boolean Algebra

Example: Minimize the logic circuit



$$Y = ABCD + ABC\overline{D} + \overline{A}\overline{B}CD + \overline{A}\overline{B}C\overline{D}$$

$$= ABC\left(D + \overline{D}\right) + \overline{A}\overline{B}C\left(D + \overline{D}\right)$$

$$= ABC + \overline{A}\overline{B}C$$

$$= \left(AB + \overline{A}\overline{B}\right)C$$

$$= \overline{A \oplus B} \bullet C$$

Produce the Logic Equation from Truth Table

Example: Find the Boolean Eq. from the truth table.

Input			Output
A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Solution: Found that the outputs of '1' occur at *ABC* of 011, 110 and 111.

$Y = 011 + 110 + 111$

$Y = min3 + min6 + min7$

$Y = \Sigma min(3, 6, 7) = F(A, B, C)$

$\bar{A}BC$ Which is another way to write the SOP output by minterms.

And produces the output equation:

$AB\bar{C}$

$Y = \bar{A}BC + AB\bar{C} + ABC$

ABC

Example: Design the circuit from the logic equation:

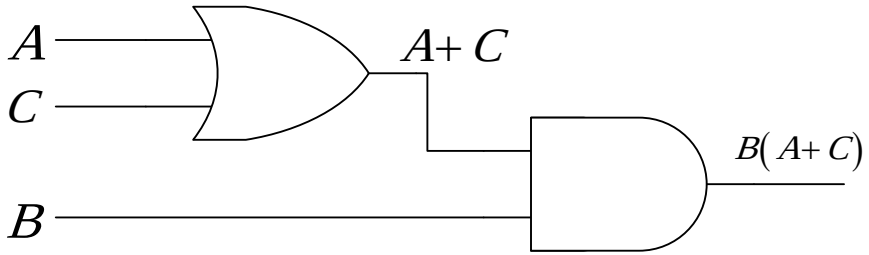
$$Y = \bar{A}BC + AB\bar{C} + ABC$$

$$= \bar{A}BC + AB(\bar{C} + C)$$

$$= \bar{A}BC + AB$$

$$= B(\bar{A}C + A)$$

$$= B(C + A)$$



Example: Design the circuit from the truth table:

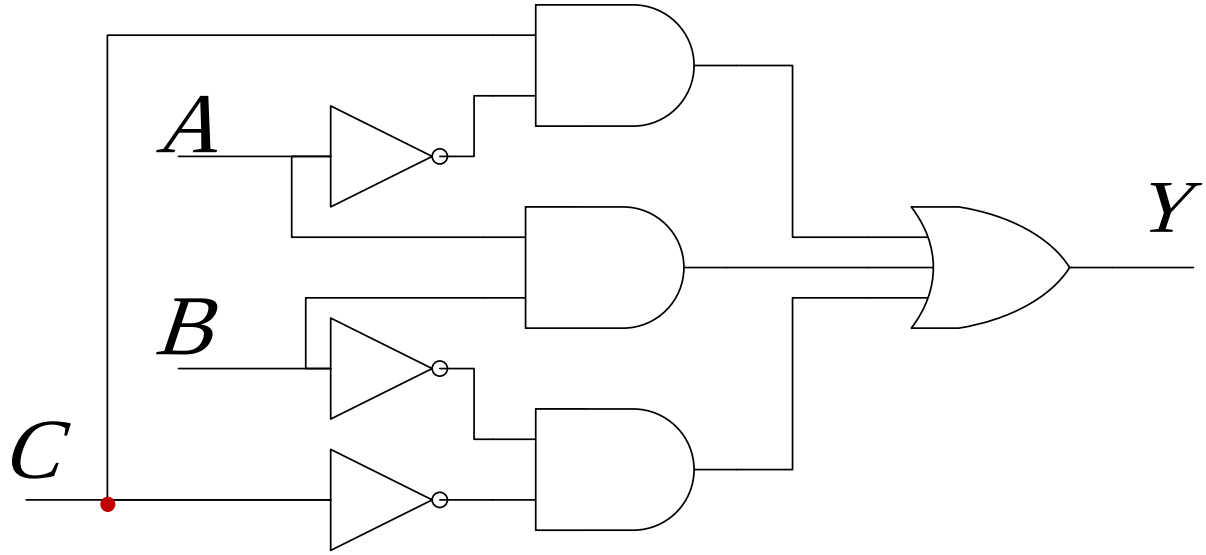
Input			Output	
A	*B*	*C*	*Y*	
0	0	0	1	$\bar{A}\bar{B}\bar{C}$
0	0	1	1	$\bar{A}\bar{B}C$
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	1	$AB\bar{C}$
1	1	1	1	ABC

Example: Design the circuit from the logic equation:

$$Y = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,B\,C$$

$$= \overline{B}\,\overline{C}\left(A + \overline{A}\right) + A\,B\left(C + \overline{C}\right) + \overline{A}\,C\left(B + \overline{B}\right)$$

$$= \overline{B}\,\overline{C} + A\,B + \overline{A}\,C$$

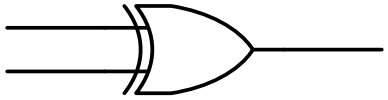


XOR & XNOR Gates using Boolean Algebra.

$$Y_1 = A \oplus B$$
$$= A \bullet \overline{B} + \overline{A} \bullet B$$

$$Y_2 = \overline{A \oplus B}$$
$$= \overline{A \bullet \overline{B} + \overline{A} \bullet B}$$
$$= \overline{(A \bullet \overline{B})} \bullet \overline{(\overline{A} \bullet B)}$$
$$= (\overline{A} + B) \bullet (A + \overline{B})$$
$$= (\overline{A} + B) \bullet A + (\overline{A} + B) \bullet \overline{B}$$
$$= \overline{A}A + AB + \overline{A}\overline{B} + B\overline{B}$$
$$= \overline{A}\overline{B} + AB$$

XOR & XNOR Gates using SOP Form.

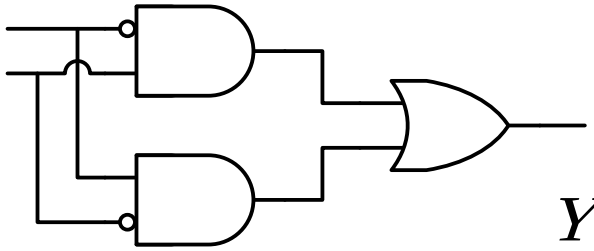


$Y = A \oplus B$

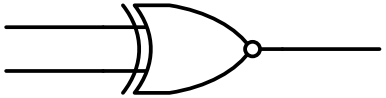
Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$\overline{A} \cdot B$

$A \cdot \overline{B}$



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

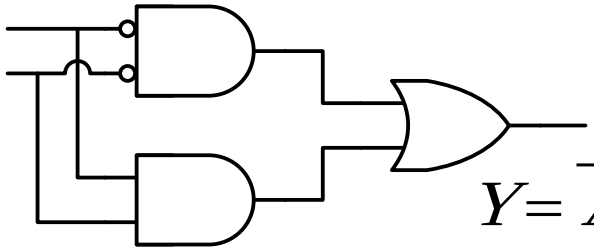


$Y = \overline{A \oplus B}$

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

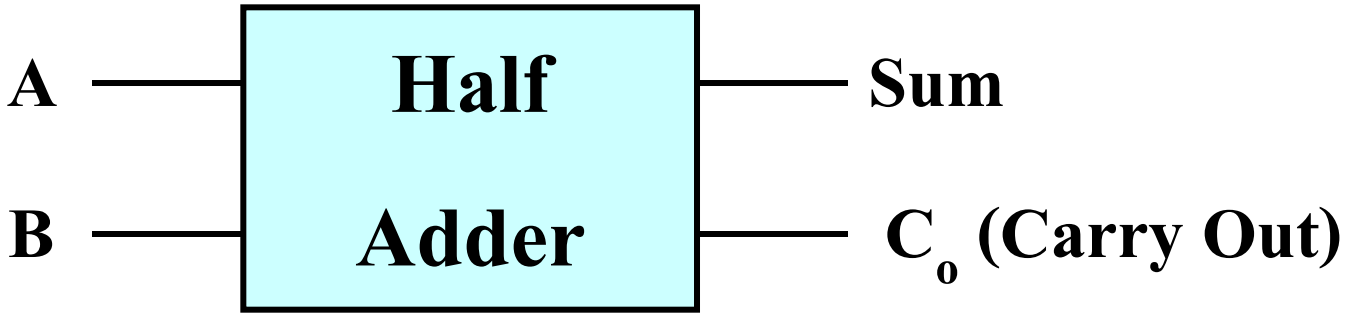
$\overline{A} \cdot \overline{B}$

$A \cdot B$



$Y = \overline{A}\,\overline{B} + A\,B$

Logical Circuit Design: Half-Adder [1]



Input		Output	
A	B	C_O	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$Sum = A \oplus B$$

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

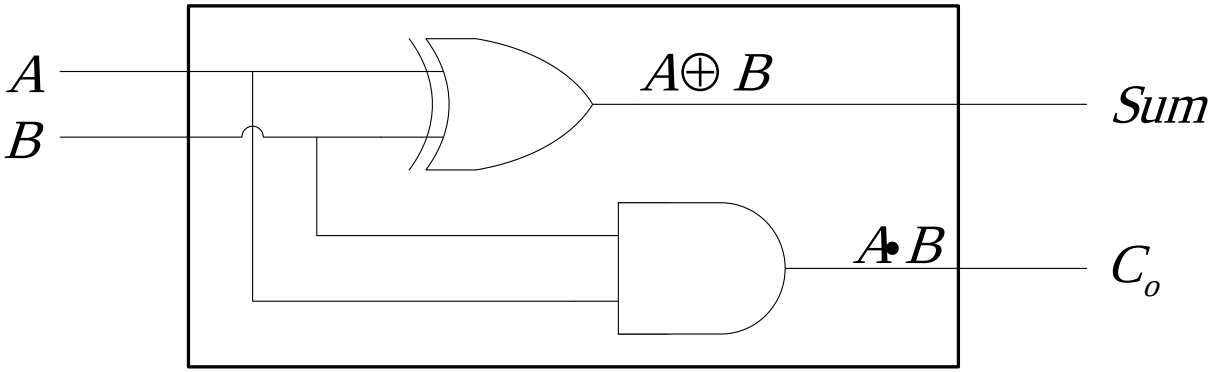
$$= \overline{\overline{A}\,\overline{B} + AB}$$

$$= \overline{\overline{\overline{A}\,\overline{B}} \cdot \overline{AB}}$$

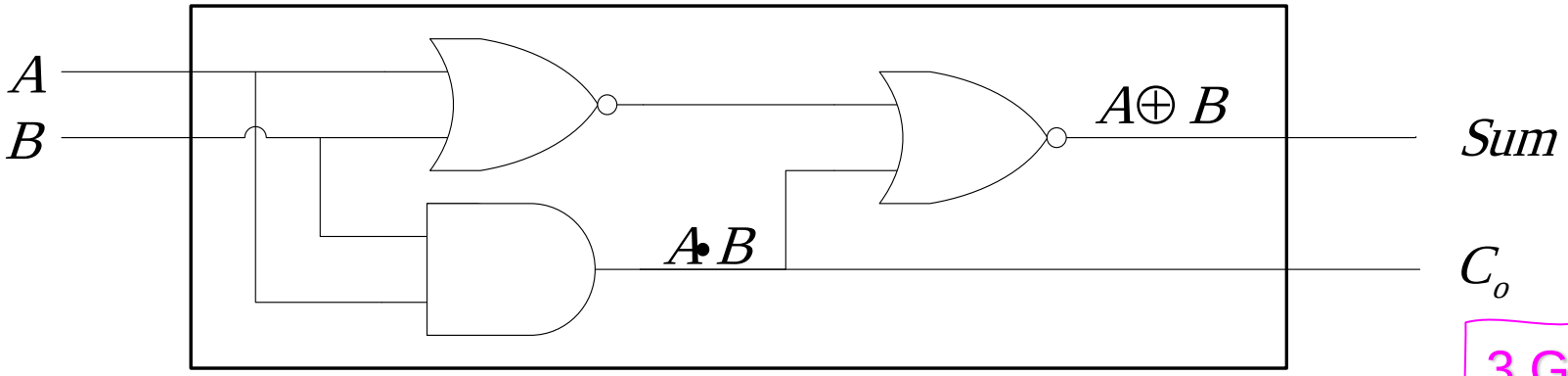
$$= \overline{\overline{A} \cdot \overline{B} \cdot \overline{AB}}$$

$$C_o = A \cdot B$$

Logical Circuit Design: Half-Adder [2]



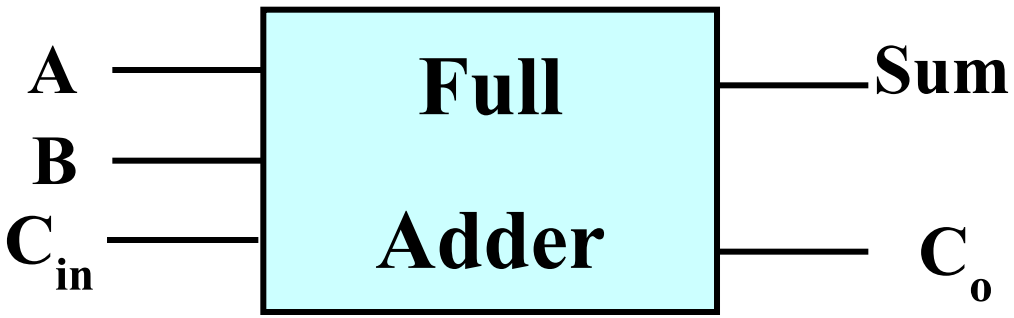
6 Gates



3 Gates

Logical Circuit Design: Full Adder [1]

Input			Output	
C_{in}	A	B	C_o	Sum
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



Cin
A
\+ B
Co Sum

Logical Circuit Design: Full Adder [2]

$$Sum = A\overline{B}\,\overline{C_{in}} + \overline{A}B\,\overline{C_{in}} + \overline{A}\,\overline{B}C_{in} + ABC_{in}$$

$$= \left(A\overline{B} + \overline{A}B\right)\overline{C_{in}} + \left(\overline{A}\,\overline{B} + AB\right)C_{in}$$

$$= \left(A \oplus B\right)\overline{C_{in}} + \left(\overline{A \oplus B}\right)C_{in}$$

$$= (A \oplus B) \oplus C_{in}$$

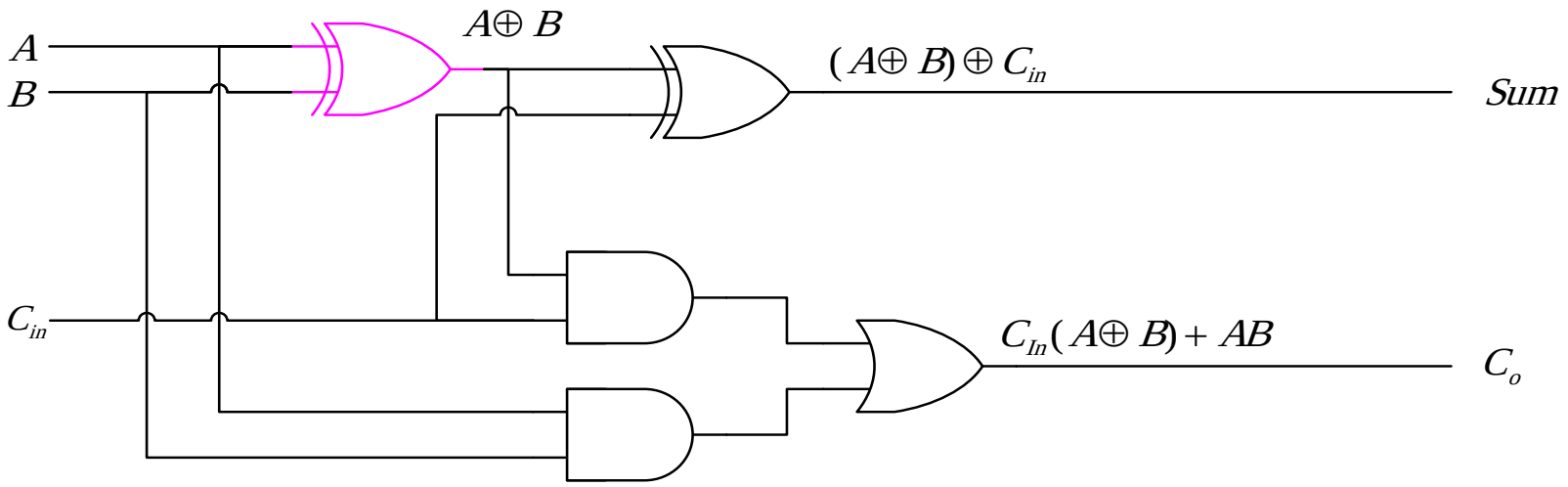
$$C_O = \overline{A}BC_{In} + A\overline{B}C_{In} + AB\overline{C_{In}} + ABC_{In}$$

$$= C_{In}(\overline{A}B + A\overline{B}) + AB(\overline{C_{In}} + C_{In})$$

$$= C_{In}(A \oplus B) + AB$$

Logical Circuit Design: Full Adder [3]

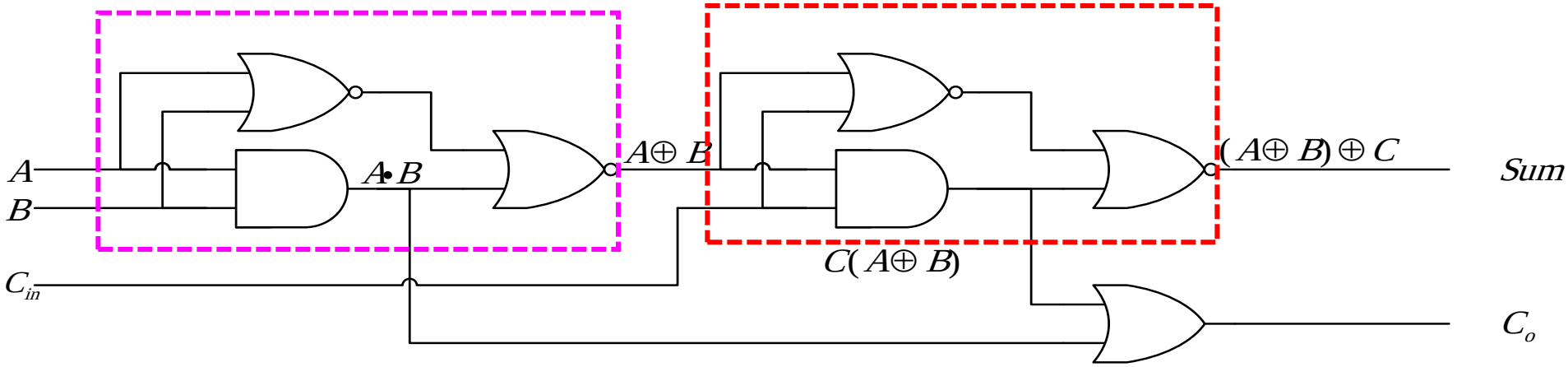
$$Sum = (A \oplus B) \oplus C_{in} \qquad C_O = C_{In}(A \oplus B) + AB$$



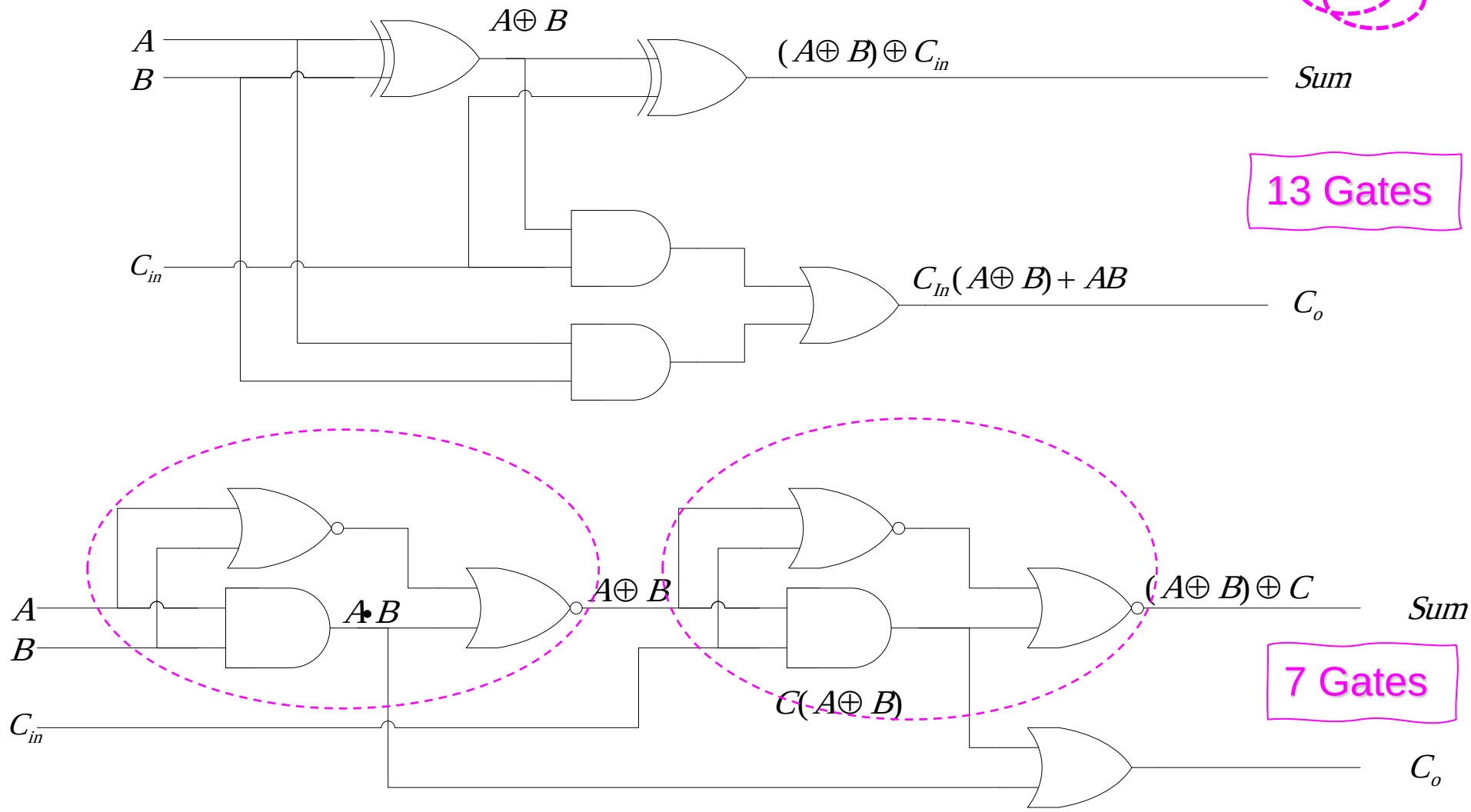
Logical Circuit Design: Full Adder [4]

$$Sum = (A \oplus B) \oplus C_{in}$$

$$C_O = C_{In}(A \oplus B) + AB$$



Logical Circuit Design: Full Adder [5]



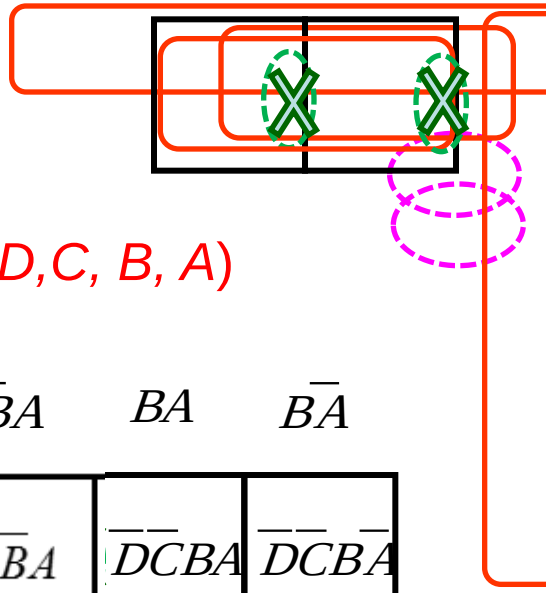
Boolean Algebra Usage: Summary

- The Boolean Algebra can be used to reduce the terms in the output equation,
- But we have to recall all rules in the theory;
- This requires some practice and experience to manage the terms in the equation;
- So, the method is suitable for the system with **several** inputs.
- For the system with **more** inputs, we would rather use:
  - Karnaugh Map
  - Quine-Mc Cluskey Method.

Karnaugh Map

- Karnaugh's Map is a Table of 2^m x 2^n,
- where $m + n$ = total number of the variables.
- The column and row are in the sequence of
  the grey codes: 000, 001, 011, 010,…
- First, we put the '1' marks on the diagram corresponds to the individual minterms, and the '0' marks for the rest.
- Typically, total number of variables is not more than 4,
- Otherwise, the diagram would get too big to handle, more complicated to consider.

Karnaugh Map



K-Map Plot:

$Y = F(B, A)$

B \ A	$\overline{A}$	A
$\overline{B}$	$\overline{B}\overline{A}$	$\overline{B}A$
B	$B\overline{A}$	BA

$Y = F(C, B, A)$

C \ BA	$\overline{B}\overline{A}$	$\overline{B}A$	BA	$B\overline{A}$
$\overline{C}$	$\overline{C}\overline{B}\overline{A}$	$\overline{C}\overline{B}A$	$\overline{C}BA$	$\overline{C}B\overline{A}$
C	$C\overline{B}\overline{A}$	$C\overline{B}A$	CBA	$CB\overline{A}$

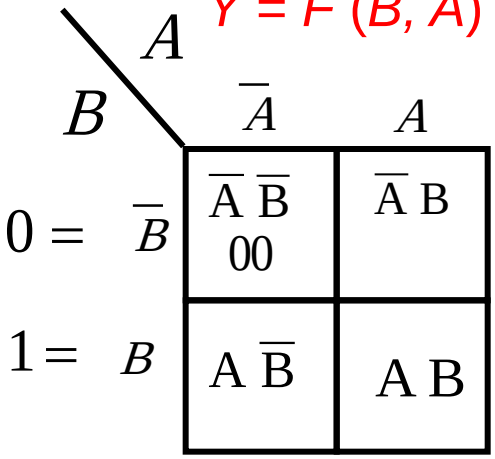
$Y = F(D, C, B, A)$

DC \ BA	$\overline{B}\overline{A}$	$\overline{B}A$	BA	$B\overline{A}$
$\overline{D}\overline{C}$	$\overline{C}\overline{B}\overline{A}$	$\overline{C}\overline{B}A$	$\overline{D}\overline{C}BA$	$\overline{D}\overline{C}B\overline{A}$
$\overline{D}C$	$\overline{D}C\overline{B}\overline{A}$	$\overline{D}C\overline{B}A$	$\overline{D}CBA$	$\overline{D}CB\overline{A}$
DC	$DC\overline{B}\overline{A}$	$DC\overline{B}A$	$DCBA$	$DCB\overline{A}$
$D\overline{C}$	$D\overline{C}\overline{B}\overline{A}$	$D\overline{C}\overline{B}A$	$D\overline{C}BA$	$D\overline{C}B\overline{A}$

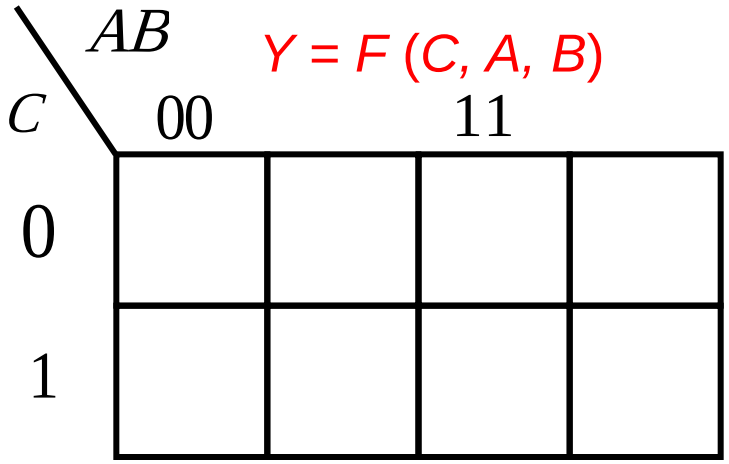
Karnaugh Map

K-Map Plot:

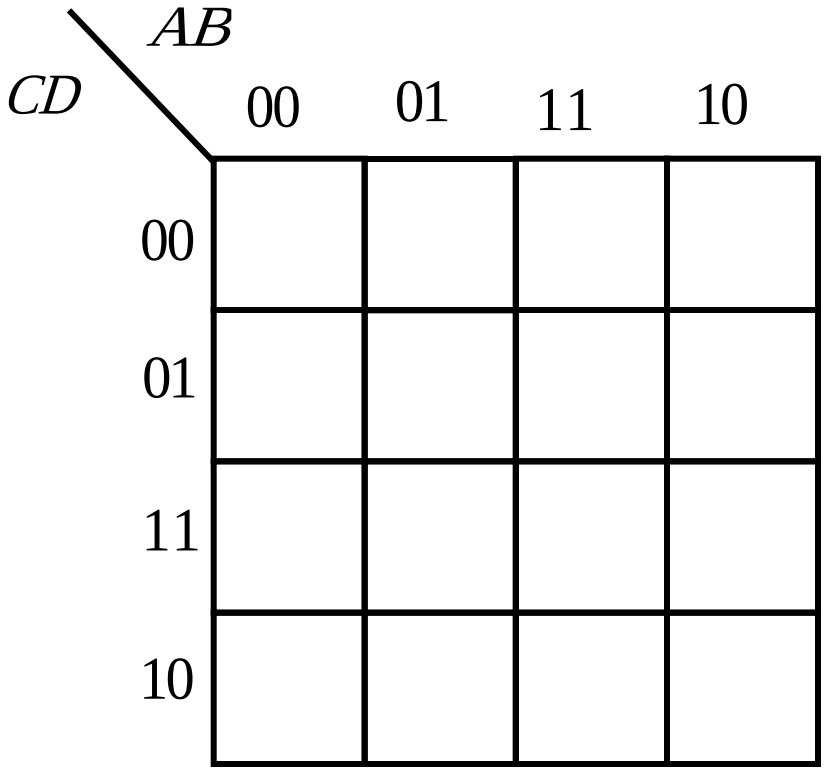
Y = F (B, A)



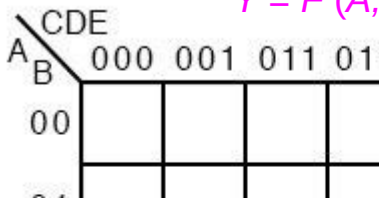
Y = F (C, A, B)



Y = F (C, D, A, B)



Y = F (A,



Karnaugh Map: How to use it

- Fill the outputs (0/1) into the condition corresponding to the individual minterms in the Karnaugh's diagram;
- Marks (encircle) all the outputs of '1';
- Try to make a group the output of '1' with the neighboring output(s) of '1' to make a bigger group of 2, 4, 8, …, 2^n by drawing a bigger circle around either in line or in square.
- In the individual group, take away the variable that has changed its logic, namely, we can reduce that variable from the minterms;
- Finally, collect/sum all the outcome products as the output equation.

Karnaugh Map : Example 1

Input			Output
A	*B*	*C*	*Y*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

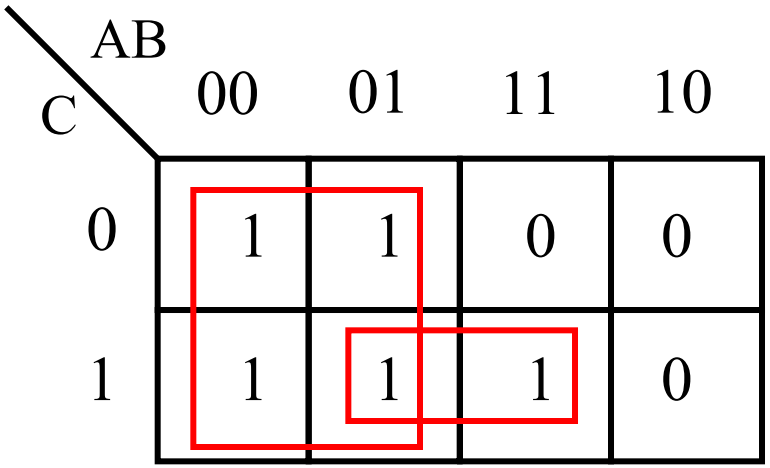
$Y = F(A, B, C) = \Sigma Min(0,1, 2, 3, 7)$
$= \Sigma m(0,1, 2, 3, 7)$

C \ AB	00	01	11	10
0	1	1	0	0
1	1	1	1	0

A \ BC	00	01	11	10
0				
1				

Karnaugh Map : Example 1

Input			Output
A	*B*	*C*	*Y*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

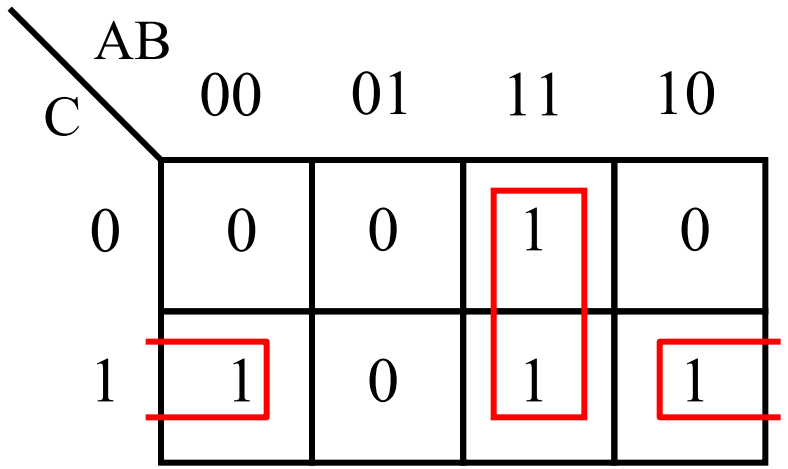


$$Y = \overline{A} + BC$$

Karnaugh Map : Example 2

Input			Output
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$Y = F(A, B, C) = \Sigma Min(1, 5, 6, 7)$



$$Y = AB + \overline{B}C$$

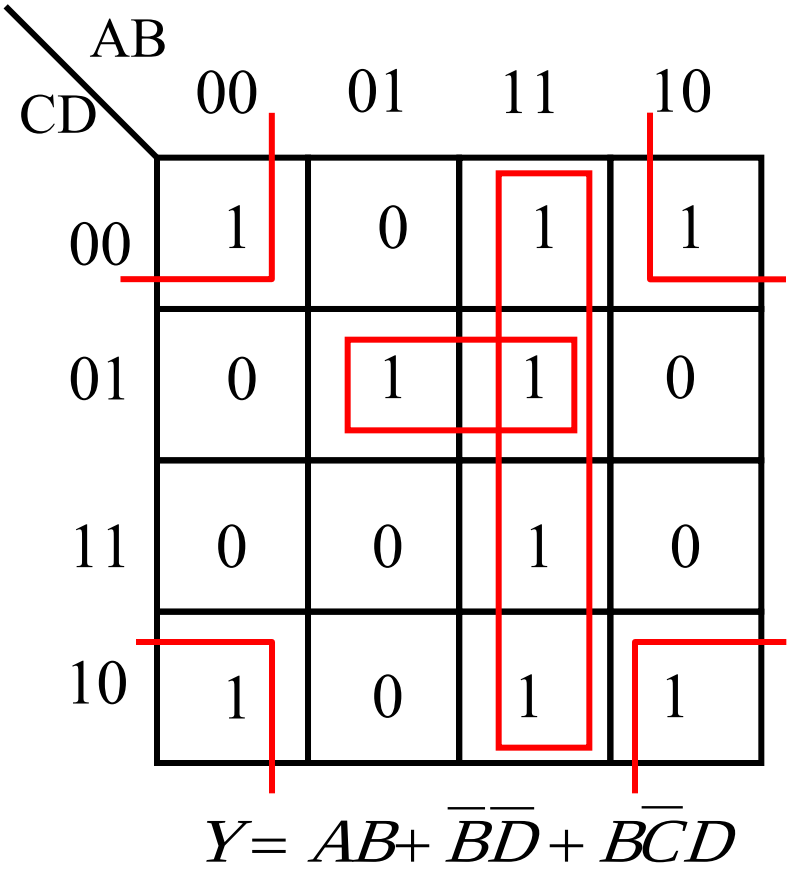
A \ BC	00	01	11	10
0				
1				

Karnaugh Map : Example 3

Input				Output
A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$Y = F(A, B, C, D)$

$= \Sigma Min(0,2,5,8,10,12,13,14,15)$



$Y = AB + \overline{B}\overline{D} + B\overline{C}D$

Karnaugh Map : Example 4

Simplify (minimize) this Boolean equation

$Y = F(A, B, C, D) = \Sigma m(2, 3, 6, 10, 11)$ using the Karnaugh Map.

Solution

The first step is to change the Boolean equation to the

Sum of Product form:

$$Y = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D}$$

Then draw the Karnaugh Map as follows:

Karnaugh Map : Example 4

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	0	0	0	0
10	0	0	1	1

Then gets the Answer: $Y = \bar{B}C + \bar{A}C\bar{D}$

Karnaugh Map : Example 5

Determine the product terms for each of the Karnaugh maps in Figure 4–32 and write the resulting minimum SOP expression.

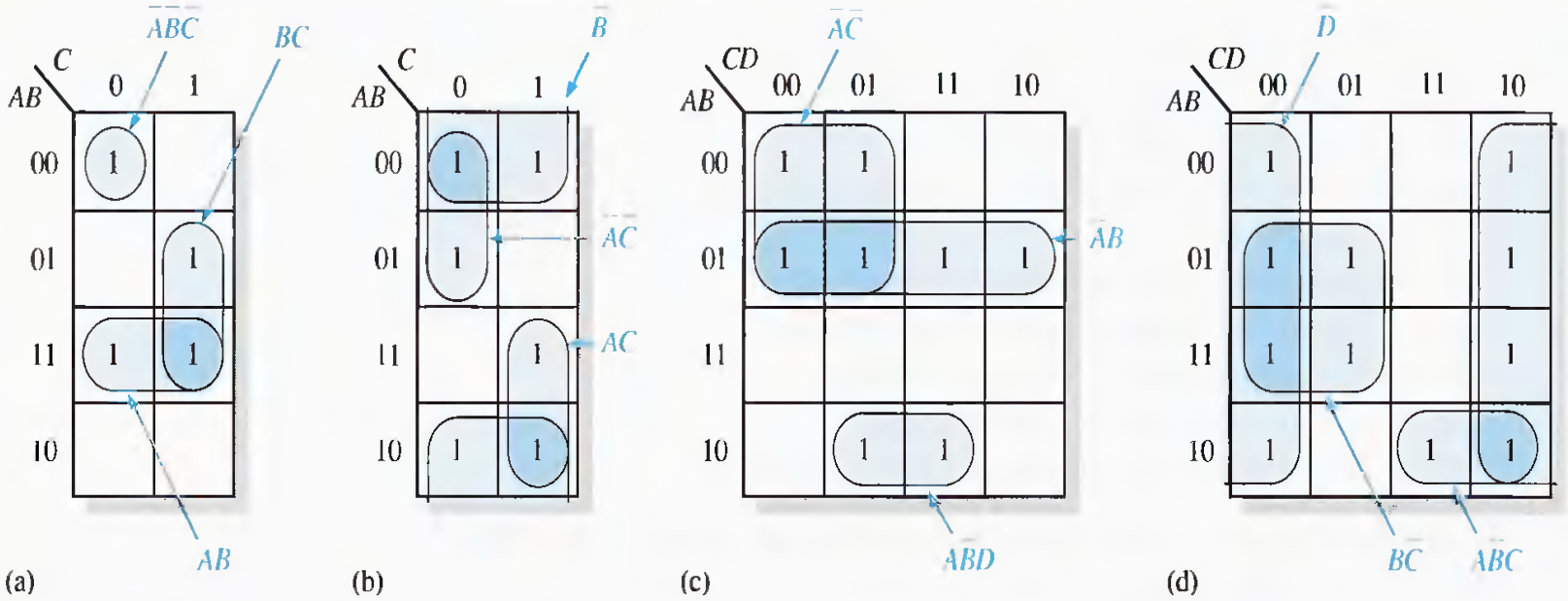


FIGURE 4–32

Solution The resulting minimum product term for each group is shown in Figure 4–32. The minimum SOP expressions for each of the Karnaugh maps in the figure are

(a) $AB + BC + \overline{A}\,\overline{B}\,\overline{C}$ (b) $\overline{B} + \overline{A}\,\overline{C} + AC$

(c) $\overline{A}B + \overline{A}\,\overline{C} + A\overline{B}D$ (d) $\overline{D} + A\overline{B}C + B\overline{C}$

Karnaugh Map : Don't Care Case

Don 't Care

- The case that can be ignored its logic value;
- Expressed by the symbol *d*, (x, *, …);
- Therefore, *d* could be any logic either 0 or 1;
- Help to simplify the output equation since it can be any logic:

  Typically, Set *d* = 1 for SOP form;

  Set *d* = 0 for POS form.

Minimize the case with Don't Care:

Set logic of the Don't Care that helps for Output Minimization;

- All '1' state need to be used;
- But Just use ONLY the Don't Cares as necessary.

Karnaugh Map: Don't Care Case

Use the Karnaugh Map to design circuits for the truth table below.

Input N	Input				Output
	A	*B*	*C*	*D*	*Y*
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	d
11	1	0	1	1	0
12	1	1	0	0	d
13	1	1	0	1	d
14	1	1	1	0	d
15	1	1	1	1	1

$Y = F(A, B, C, D)$

$= \Sigma m(\quad)$

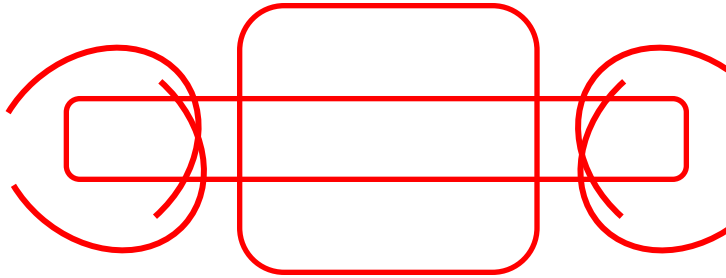
$+ \Sigma d(\quad)$

$Y = \Sigma m(0, 2, 5, 7, 8, 15) + \Sigma d(10, 12, 13, 14)$

Karnaugh Map: Don't Care Case

$Y = F(A, B, C, D) = \Sigma m(0, 2, 5, 7, 8, 15) + \Sigma d(10, 12, 13, 14)$

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	d	d	1	d
10	1	0	0	d



$$Y = BD + \overline{B}\,\overline{D}$$