CSET105- DIGITAL DESIGN

GATE-LEVEL MINIMIZATION

BENNETT UNIVERSITY

MODULE-3

GATE LEVEL MINIMIZATION

The Map Method - K-map, Product of Sums and Sum of Products Simplification, NAND and NOR Implementation

GATE MINIMIZATION USING BOOLEAN ALGEBRA

SIMPLIFICATION USING BOOLEAN ALGEBRA

- Many times in the application of Boolean algebra, you have to reduce a particular expression to its simplest form or change its form to a more convenient one to implement the expression most efficiently.

- This approach uses the basic laws, rules, and theorems of Boolean algebra to manipulate and simplify an expression.

SIMPLIFICATION USING BOOLEAN ALGEBRA

Using Boolean algebra techniques, simplify this expression:

$$AB + A(B + C) + B(B + C)$$

Solution The following is not necessarily the only approach.

Step 1: Apply the distributive law to the second and third terms in the expression, as follows:

$$AB + AB + AC + BB + BC$$

Step 2: Apply rule 7 ($BB = B$) to the fourth term.

$$AB + AB + AC + B + BC$$

Step 3: Apply rule 5 ($AB + AB = AB$) to the first two terms.

$$AB + AC + B + BC$$

Step 4: Apply rule 10 ($B + BC = B$) to the last two terms.

$$AB + AC + B$$

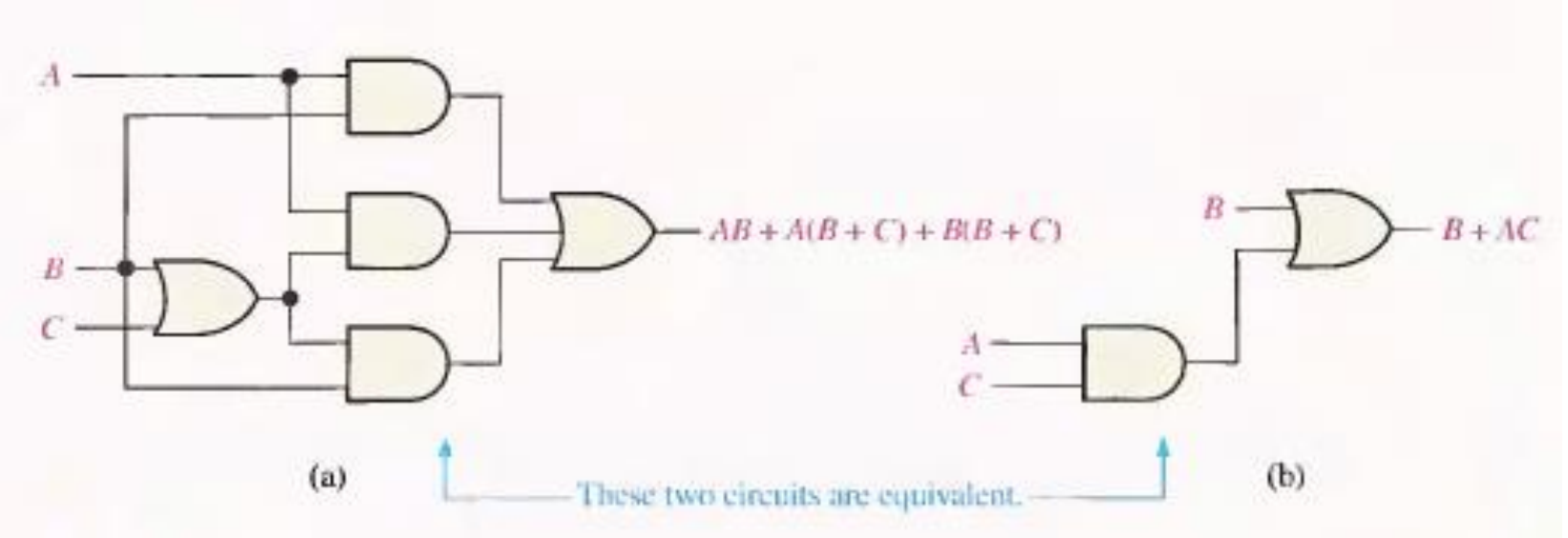
Step 5: Apply rule 10 ($AB + B = B$) to the first and third terms.

$$B + AC$$

At this point the expression is simplified as much as possible. Once you gain experience in applying Boolean algebra, you can often combine many individual steps.

Related Problem Simplify the Boolean expression $A\overline{B} + A(\overline{B + C}) + B(\overline{B + C})$.

SIMPLIFICATION USING BOOLEAN ALGEBRA



1. Simplify the following Boolean expressions if possible:
   (a) $A + AB + A\bar{B}C$ (b) $(\bar{A} + B)C + ABC$ (c) $A\bar{B}C(BD + CDE) + A\bar{C}$
2. Implement each expression in Question 1 as originally stated with the appropriate logic gates. Then implement the simplified expression, and compare the number of gates.

SIMPLIFICATION USING BOOLEAN ALGEBRA

Simplify the following Boolean expression:

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

Solution

Step 1: Apply DeMorgan's theorem to the first term.

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

Step 2: Apply DeMorgan's theorem to each term in parentheses.

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

Step 3: Apply the distributive law to the two terms in parentheses.

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

Step 4: Apply rule 7 ($\overline{A}\,\overline{A} = \overline{A}$) to the first term, and apply rule 10 [$\overline{A}\,\overline{B} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}(1 + C) = \overline{A}\,\overline{B}$] to the third and last terms.

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

Step 5: Apply rule 10 [$\overline{A} + \overline{A}\,\overline{C} = \overline{A}(1 + \overline{C}) = \overline{A}$] to the first and second terms.

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

Step 6: Apply rule 10 [$\overline{A} + \overline{A}\,\overline{B} = \overline{A}(1 + \overline{B}) = \overline{A}$] to the first and second terms.

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

Related Problem Simplify the Boolean expression $\overline{AB} + \overline{AC} + \overline{A}\,\overline{B}\,\overline{C}$.

GATE MINIMIZATION USING K-MAP METHOD

GATE MINIMIZATION USING K-MAP

SIMPLIFICATION USING KARNAUGH MAP

- The Karnaugh map technique provides a systematic method for simplifying and manipulation of Boolean expressions
- A Karnaugh map is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value.
- Karnaugh map is an array of cells in which each cell represents a binary value of the input variables.

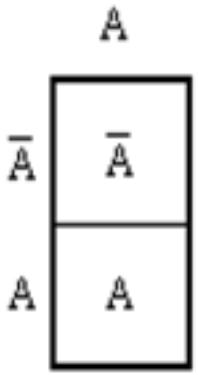
GATE MINIMIZATION USING K-MAP

SIMPLIFICATION USING KARNAUGH MAP

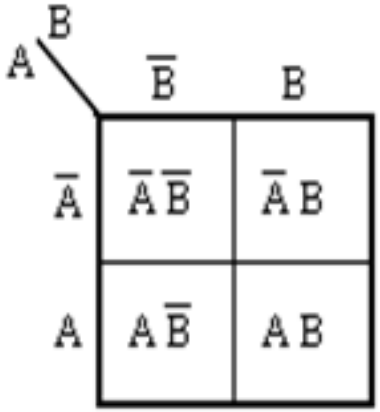
- The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations.
- For *n* variables on a Karnaugh map there are 2^n numbers of squares. For three variables, number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.
- Karnaugh maps can be used for expressions with two, three, four, and five variables.

GATE MINIMIZATION USING K-MAP

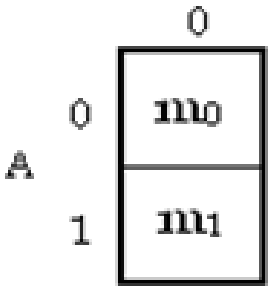
SIMPLIFICATION USING KARNAUGH MAP



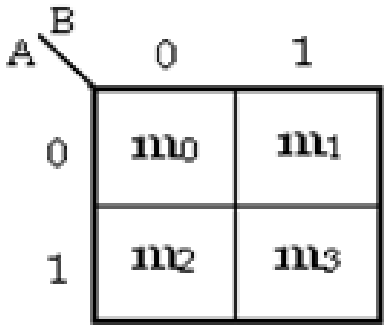
1-Variable map



2-Variable map



1-Variable map



2-Variable map

GATE MINIMIZATION USING K-MAP

3-VARIABLE KARNAUGH MAP

- The 3-variable Karnaugh map is an array of eight cells.
- In this case, A, B, and C are used for the variables although other letters could be used.
- Binary values of A is at the left side (notice the sequence) and the values of BC are across the top.
- The value of a given cell is the binary value of A at the left in the same row combined with the values of B & C at the top in the same column.

GATE MINIMIZATION USING K-MAP

3-VARIABLE KARNAUGH MAP

- Figure shows the standard product terms that are represented by each cell in the Karnaugh map.

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$AB\bar{C}$

3-Variable map

Gray code Sequence

A \ BC	00	01	11	10
0	1110	1111	1113	1112
1	1114	1115	1117	1116

3 -Variable map

GATE MINIMIZATION USING K-MAP

4-VARIABLE KARNAUGH MAP

- The 4-variable Karnaugh map is an array of sixteen cells.
- Binary values of A and B are along the left side and the values of C and D are across the top.
- The value of a given cell is the binary values of A and B at the left in the same row combined with the binary values of C and D at the top in the same column.
- For example, the cell in the upper right corner has a binary value of 0010 and the cell in the lower right corner has a binary value of 1010.

GATE MINIMIZATION USING K-MAP

4-VARIABLE KARNAUGH MAP

- Figure shows the standard product terms that are represented by each cell in the 4-variable Karnaugh map.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

4 -Variable map

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Gray code Sequence (CD)

Gray code Sequence (AB)

4- Variable map

GATE MINIMIZATION USING K-MAP

4-VARIABLE KARNAUGH MAP

- Cells that differ by only one variable are adjacent & Cells with values that differ by more than one variable are not adjacent.
- In the 3-variable map the 010 cell is adjacent to the 000,011 and 110 cell. The 010 cell is not adjacent to the 001, 111, 100, or 101 cell.
- Physically, each cell is adjacent to the cells that are immediately next to it on any of its four sides.
- A cell is not adjacent to the cells that diagonally touch any of its corners.

GATE MINIMIZATION USING K-MAP

4-VARIABLE KARNAUGH MAP

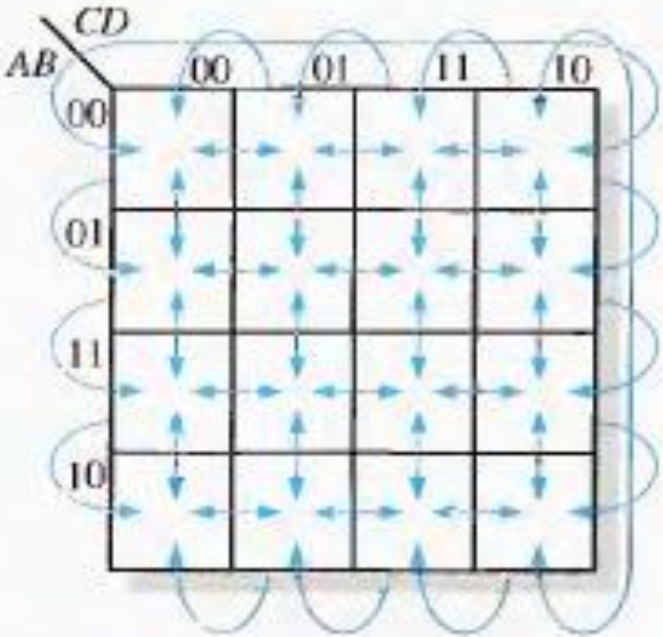


FIGURE 4–23

Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.

When we have a difference of 1 bit b/w two adjacent cells we can easily eliminate Redundant literals from the expression & then we can minimize our function which is the aim of K-Map !

❑ Why k-map follows gray code logic?

In the case of K-Map, we have difference of only 1-bit b/w adjacent cells. Consider the case of first 2 cells: 00 01. In case(SOP) if we have 1 in both these cells then it will produce A'B'+A'B. Which will give A'. Because B+B'=1. To utilize this concept of B+B'=1 easily we have Gray code in K-Map.

GATE MINIMIZATION USING K-MAP

GROUPING IN KARNAUGH MAP

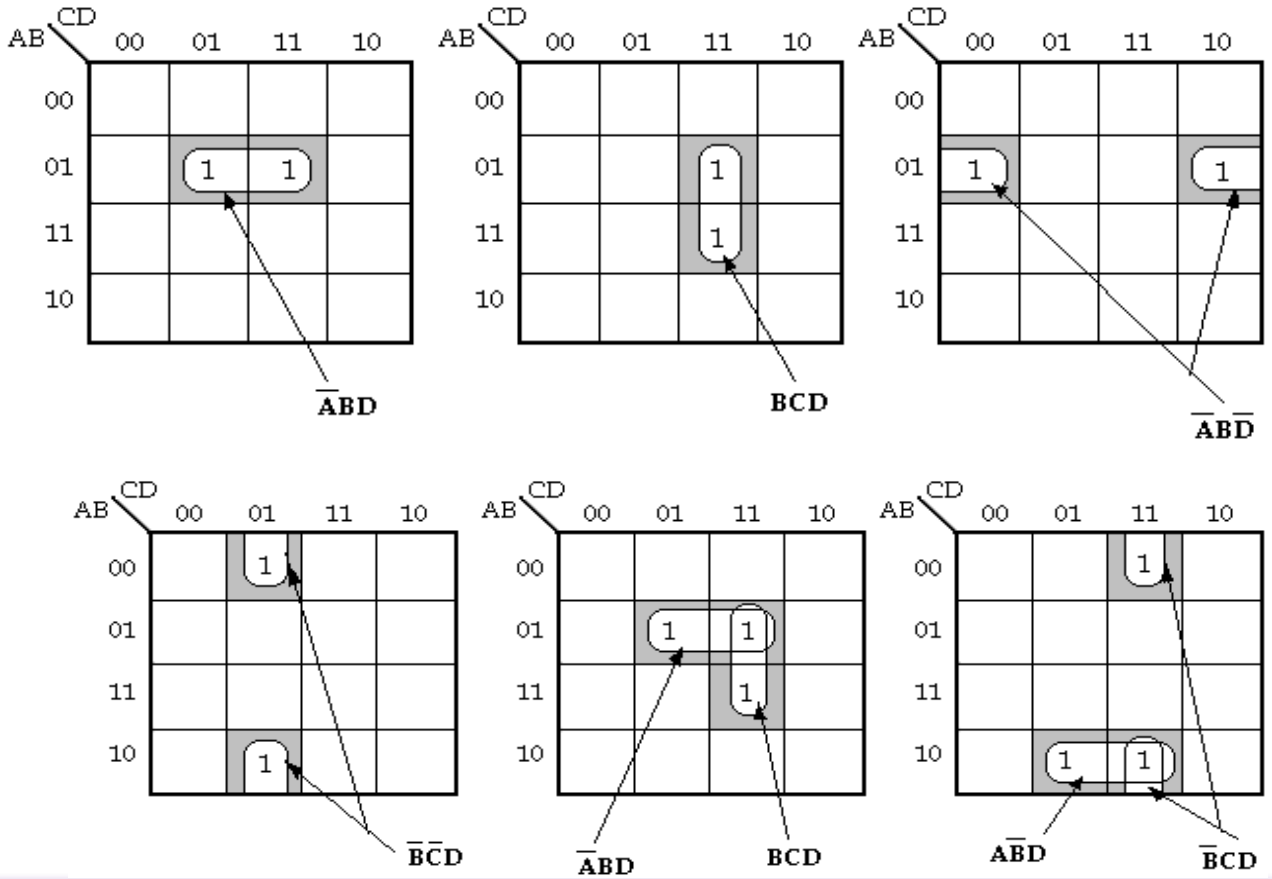
- The power of K-maps is in minimizing the terms, K-maps can be minimized with the help of grouping the terms to form single terms.

- When forming groups of squares, observe the following:
  1. Every square containing 1 must be considered at least once.
  2. The number of 1's in a group must be equal to 2^n , i.e. 2,4,8.
  3. A group must be as large as possible.
  4. A square containing 1 can be included in as many groups as desired.
  5. If a square containing 1 cannot be placed in a group, then leave it out to include in final expression.
  6. The simplified logic expression from a K-map is not always unique.
  7. Groupings can be made in different ways.

GATE MINIMIZATION USING K-MAP

TYPES OF GROUPING IN KARNAUGH MAP

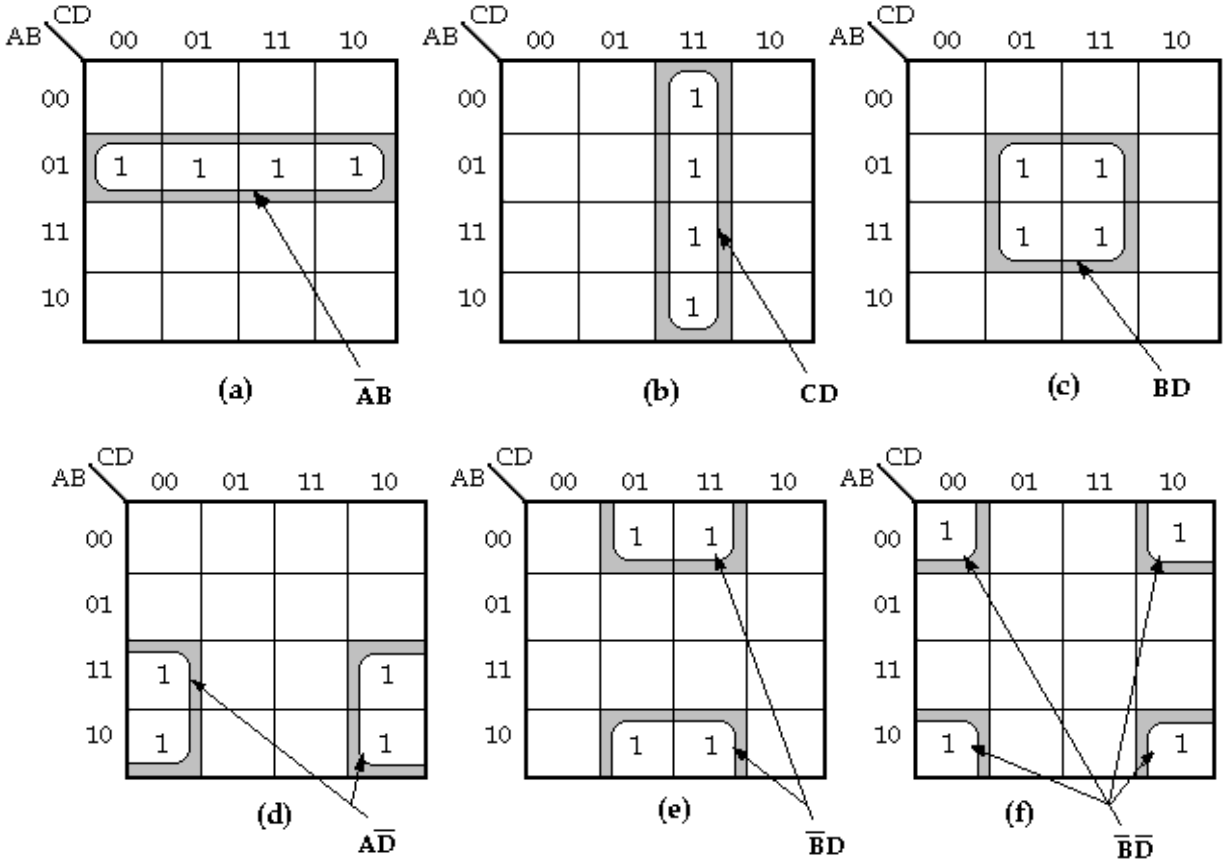
- **PAIR:** Grouping of two 1's together and it is represented with three variables



GATE MINIMIZATION USING K-MAP

TYPES OF GROUPING IN KARNAUGH MAP

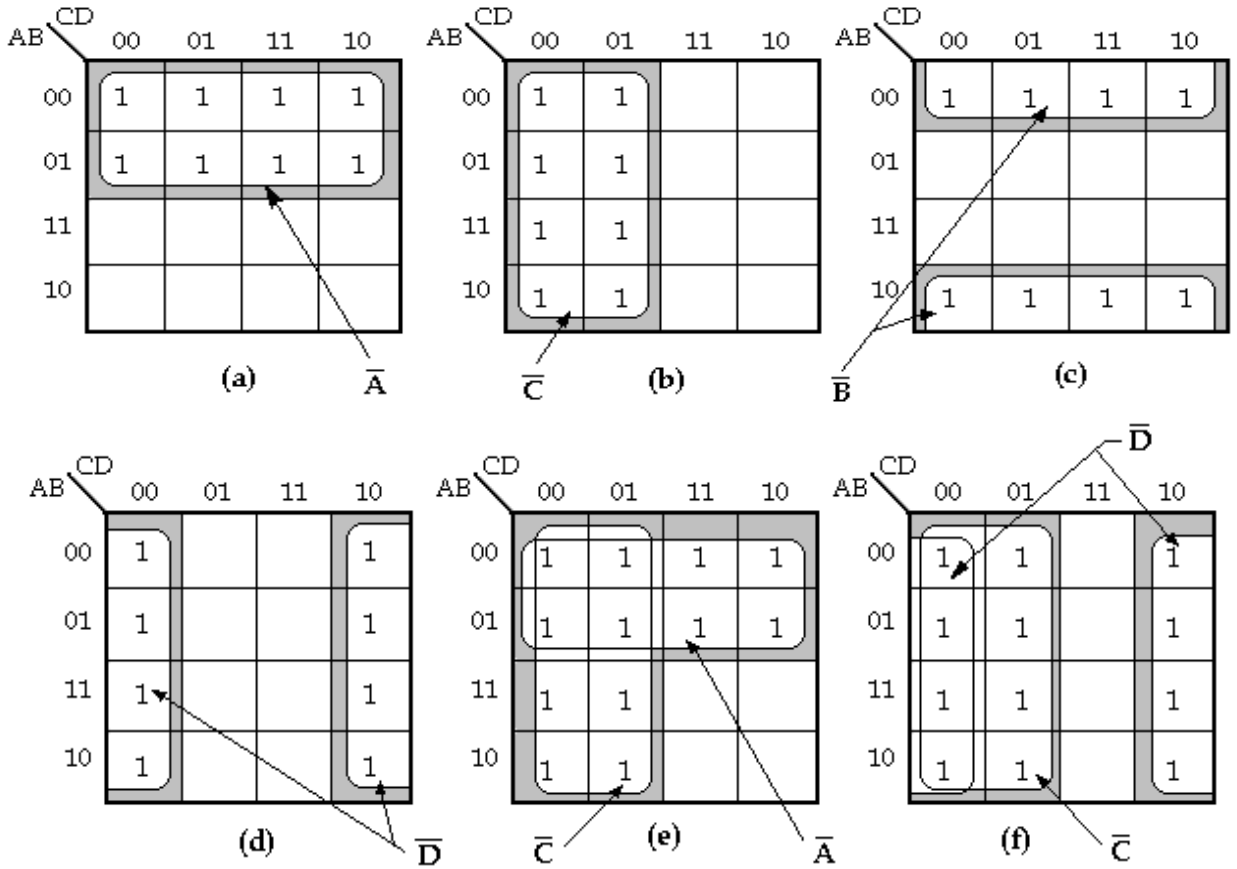
- **QUAD:** Grouping of four adjacent 1's and it results two variables



GATE MINIMIZATION USING K-MAP

TYPES OF GROUPING IN KARNAUGH MAP

- **OCTET:** If eight adjacent 1's are combined; and represented with by one variable.



GATE MINIMIZATION USING K-MAP

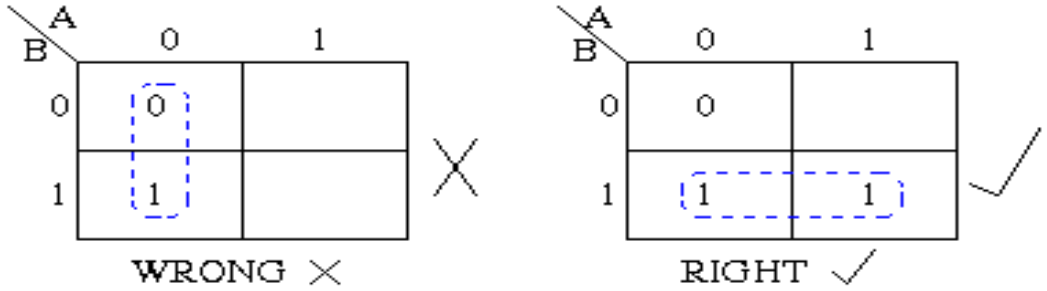
RULES FOR KARNAUGH MAP MINIMIZATION

1. No zeros allowed in grouping.
2. No diagonals grouping.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every 1's must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

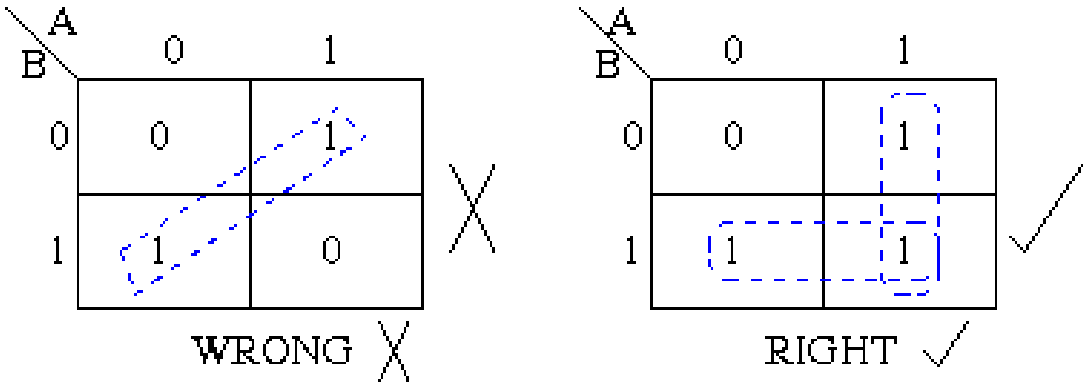
GATE MINIMIZATION USING K-MAP

RULES FOR KARNAUGH MAP MINIMIZATION

1. Groups may not include any cell containing a zero

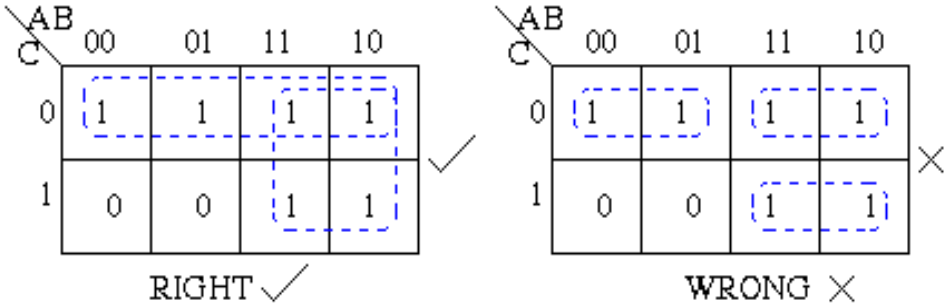


2. Groups may be horizontal or vertical, but not diagonal



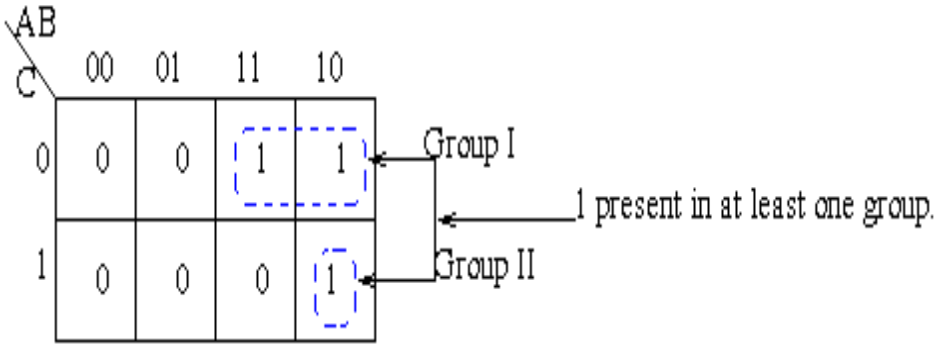
GATE MINIMIZATION USING K-MAP

4. Each group should be as large as possible.



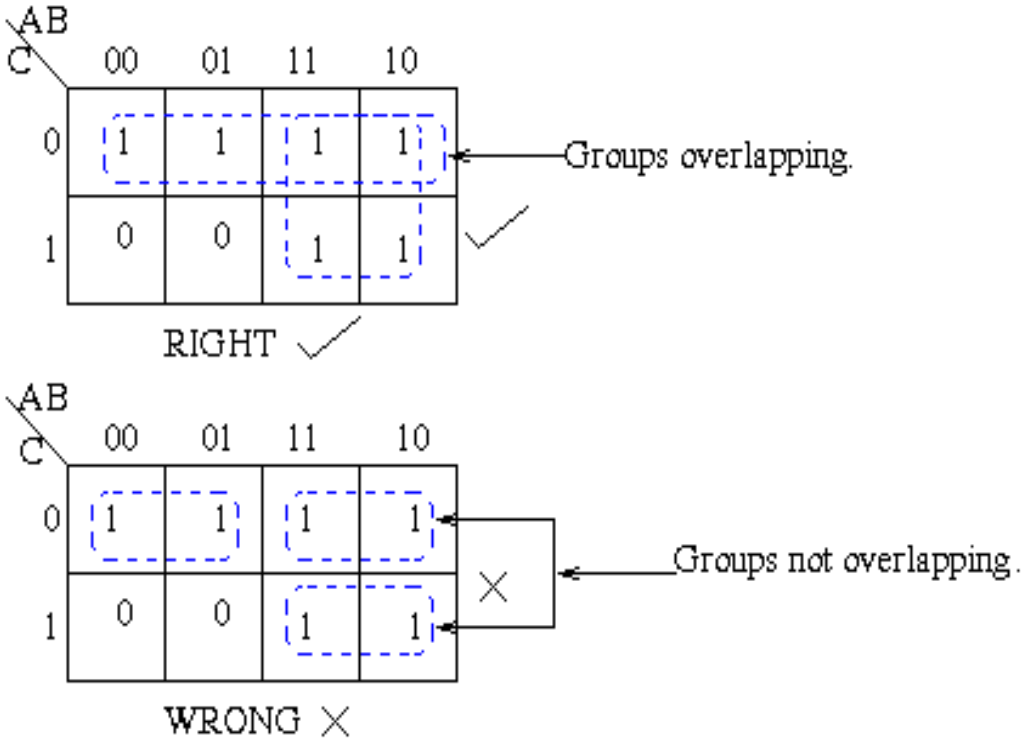
(Note that no Boolean laws broken, but not sufficiently minimal)

5. Each cell containing a *one* must be in at least one group.



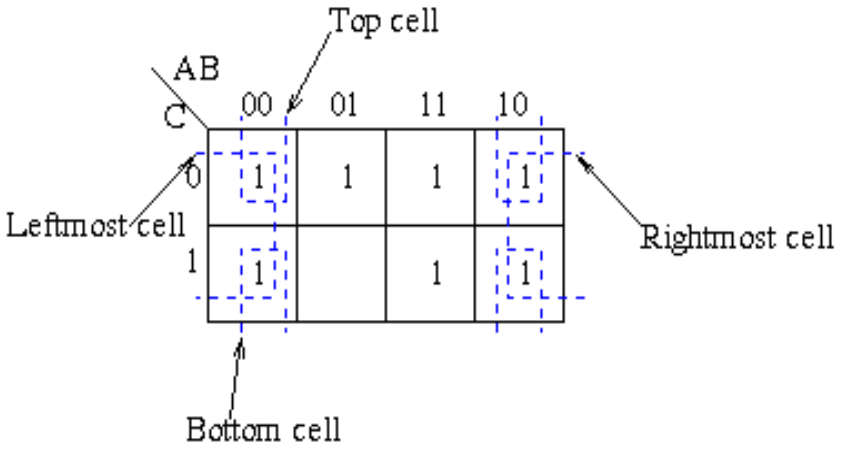
GATE MINIMIZATION USING K-MAP

6. Groups may overlap.

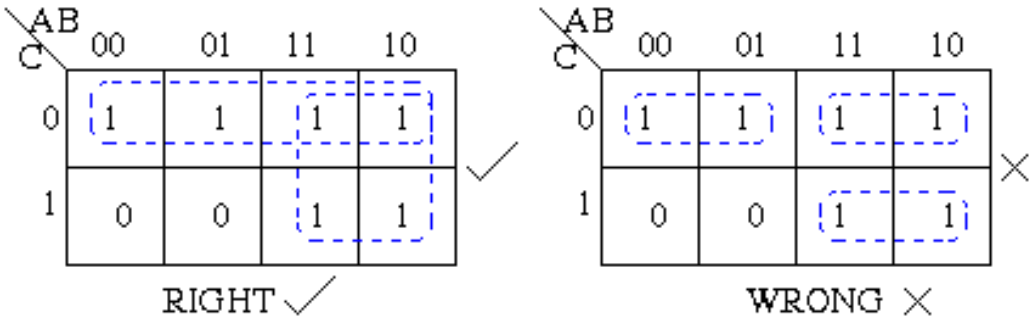


GATE MINIMIZATION USING K-MAP

7. Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



8. There should be as few groups as possible, as long as this does not contradict any of the previous rules.



GATE MINIMIZATION USING K-MAP

PROCEDURE FOR K- MAP SOP MINIMIZATION

The generalized procedure to simplify Boolean expressions as follows:

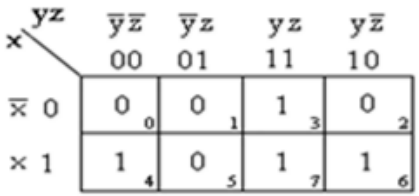
1. Plot the K-map and place 1's in those cells corresponding to the 1's in the sum of product expression. Place 0's in the other cells.
2. Check the K-map for adjacent 1's and encircle those 1's which are not adjacent to any other 1's. These are called isolated 1's.
3. Check for those 1's which are adjacent to only one other 1 and encircle such pairs.
4. Check for quads and octets of adjacent 1's even if it contains some 1's that have already been encircled. While doing this make sure that there are minimum number of groups.
5. Form the simplified expression by summing product terms of all the groups.

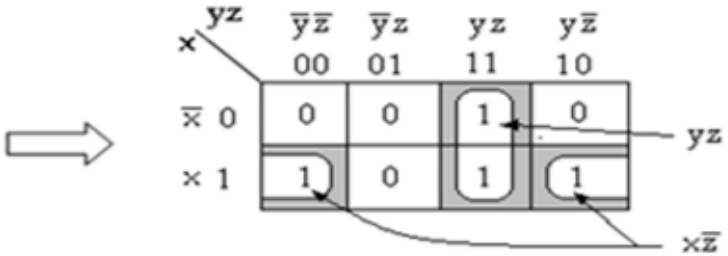
GATE MINIMIZATION USING K-MAP

3-Variable K-Maps Simplification

1. Simplify the Boolean expression,
   $F(x, y, z) = \sum m\,(3, 4, 6, 7)$.

Soln:

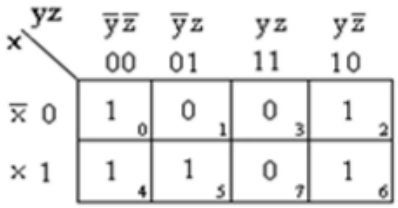




$F = yz + xz'$

2. $F(x, y, z) = \sum m\,(0, 2, 4, 5, 6)$.

Soln:



GATE MINIMIZATION USING K-MAP

3-Variable K-Maps Simplification

3. $F = A'C + A'B + AB'C + BC$

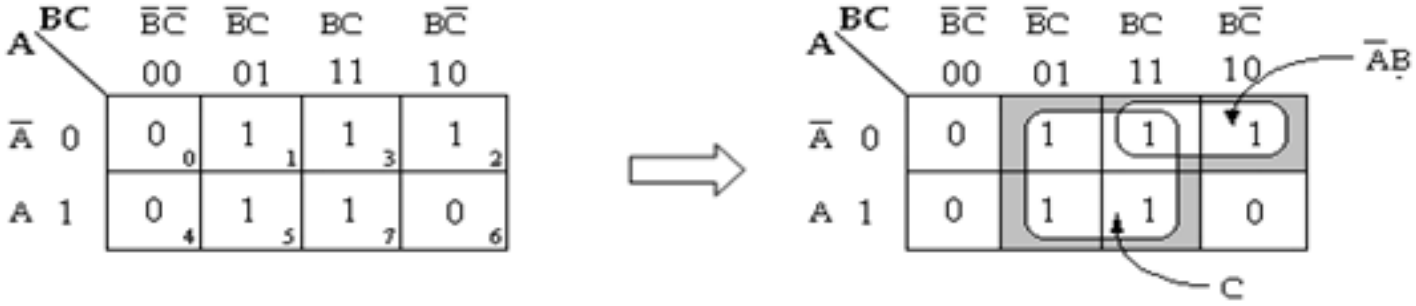
$= A'C(B + B') + A'B(C + C') + AB'C + BC(A + A')$

$= \underline{A'BC} + A'B'C + \underline{A'BC} + A'BC' + AB'C + ABC + \underline{A'BC}$

$= A'BC + A'B'C + A'BC' + AB'C + ABC$

$= m_3 + m_1 + m_2 + m_5 + m_7$

$= \sum m(1, 2, 3, 5, 7)$



$F = C + A'B$

GATE MINIMIZATION USING K-MAP

K- MAP SOP MINIMIZATION

Design an car driver alert system based on following condition. The buzzer (B=1) will alert the driver whenever any one of the following condition is satisfied.

(i). When engine is ON (E=1) and any one of the door is opened (D=1)
(ii).when fuel level is low (F=1)
(iii). When car exceed speed limit (S=1)

GATE MINIMIZATION USING K-MAP

PROCEDURE FOR K- MAP SOP MINIMIZATION

Truth table

E	D	F	S	B
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

E	D	F	S	B
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

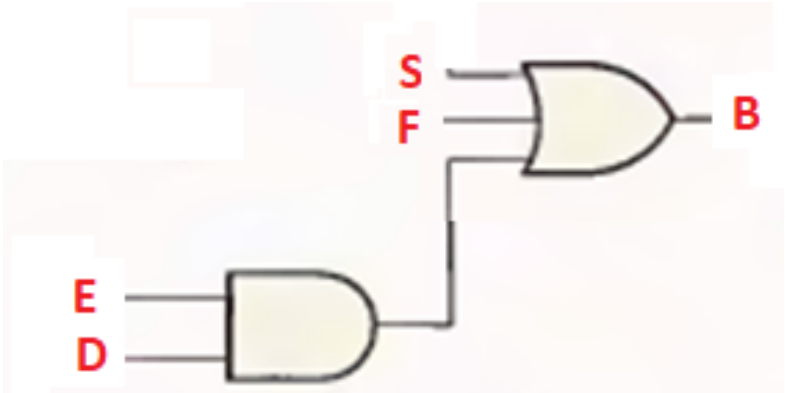
GATE MINIMIZATION USING K-MAP

PROCEDURE FOR K- MAP SOP MINIMIZATION

K-MAP

	F'S'	F'S	FS	FS'
E'D'	0	1	1	1
E'D	0	1	1	1
ED	1	1	1	1
ED'	0	1	1	1

	F'S'	F'S	FS	FS'
E'D'	0	1	1	1
E'D	0	1	1	1
ED	1	1	1	1
ED'	0	1	1	1



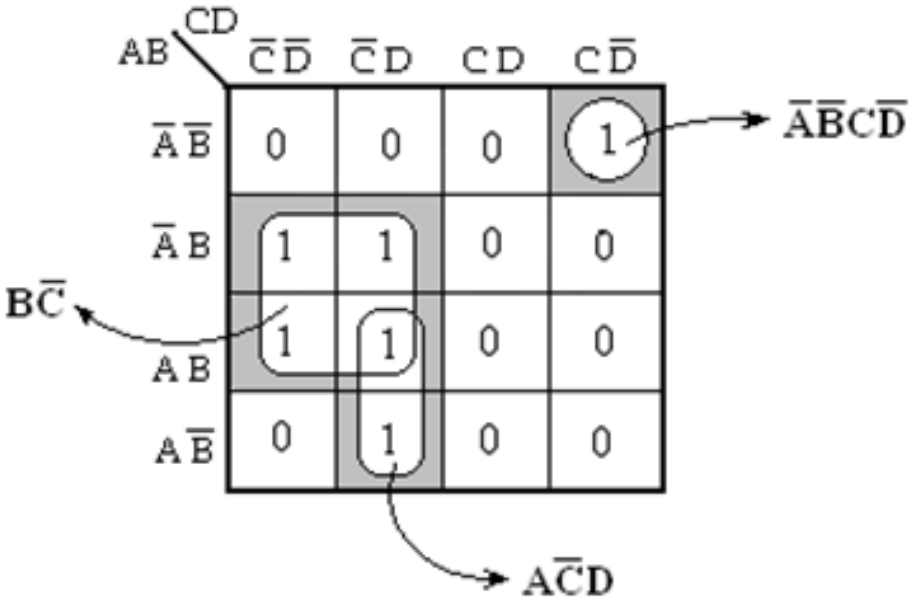
B=F+S+ED

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS - EXAMPLE

Simplify the Boolean expression,

$Y = A'BC'D' + A'BC'D + ABC'D' + ABC'D + AB'C'D + A'B'CD'$



Therefore, $Y = A'B'CD' + AC'D + BC'$

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS - EXAMPLE

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

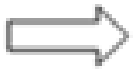
$$= A'B'C'(D + D') + B'CD'(A + A') + A'BCD' + AB'C'(D + D')$$

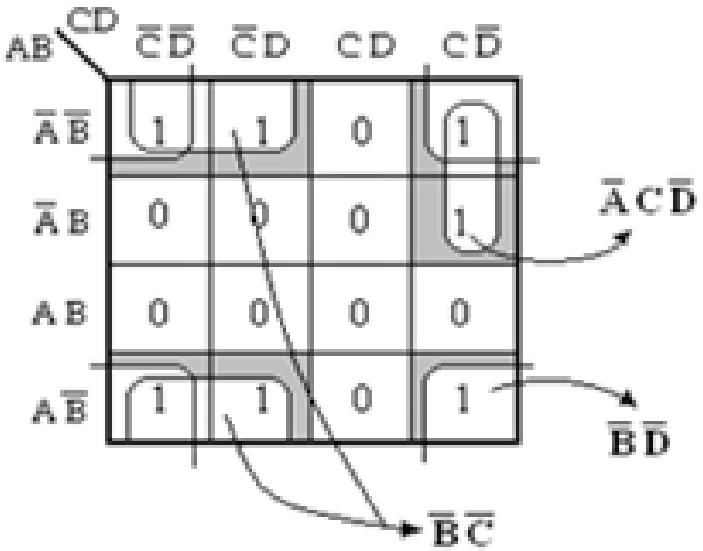
$$= A'B'C'D + A'B'C'D' + AB'CD' + A'B'CD' + A'BCD' + AB'C'D + AB'C'D'$$

$$= m_1 + m_0 + m_{10} + m_2 + m_6 + m_9 + m_8$$

$$= \sum m\,(0, 1, 2, 6, 8, 9, 10)$$

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	0 (3)	1 (2)
01	0 (4)	0 (5)	0 (7)	1 (6)
11	0 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	0 (11)	1 (10)





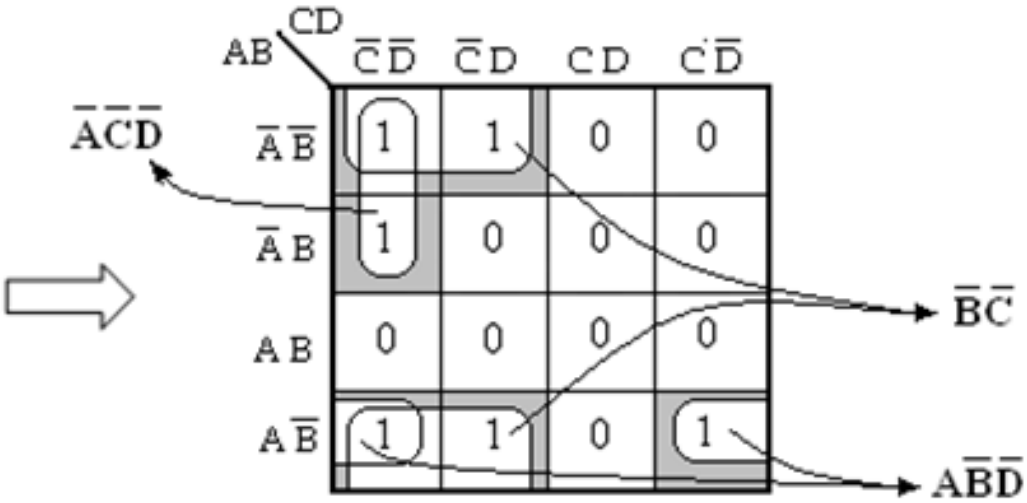
Therefore, $F = B'D' + B'C' + A'CD'$.

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS - EXAMPLE

$F(A, B, C, D) = \sum m\,(0, 1, 4, 8, 9, 10)$

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	0 (3)	0 (2)
01	1 (4)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	0 (11)	1 (10)



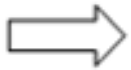
Therefore, F= A′C′D′+ AB′D′+ B′C′.

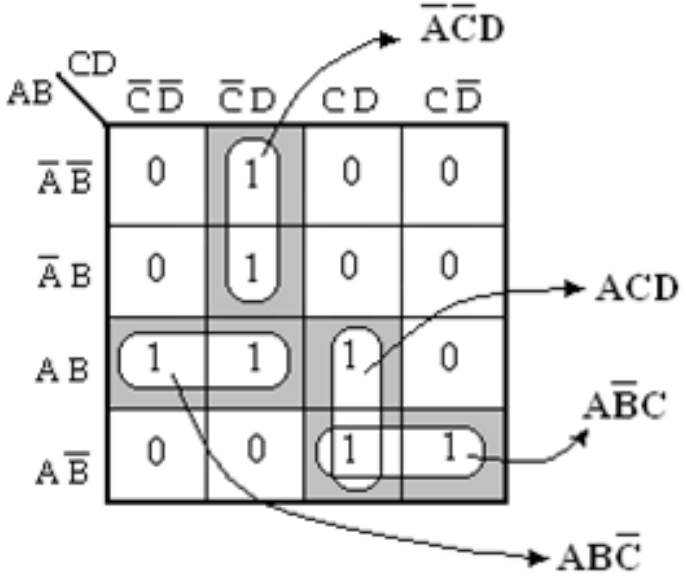
GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS - EXAMPLE

$Y = \sum m\,(1, 5, 10, 11, 12, 13, 15)$

AB \ CD	00	01	11	10
00	0 (0)	1 (1)	0 (3)	0 (2)
01	0 (4)	1 (5)	0 (7)	0 (6)
11	1 (12)	1 (13)	1 (15)	0 (14)
10	0 (8)	0 (9)	1 (11)	1 (10)





Therefore, $Y = A'C'D + ABC' + ACD + AB'C$.

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS – WITH DON'T CARES

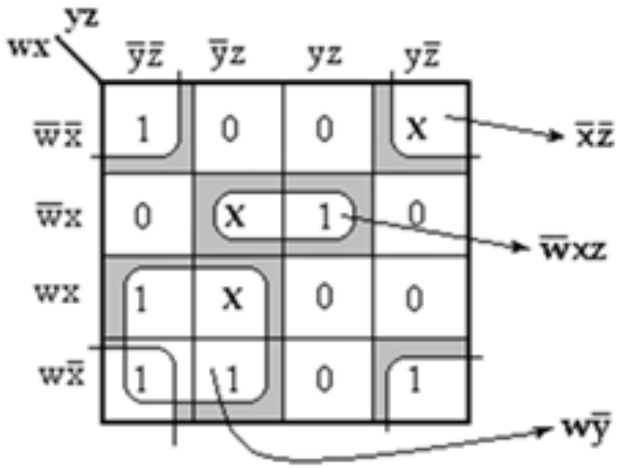
- A don't care minterm is a combination of variables whose logical value is not specified.
- When choosing adjacent squares to simplify the function in a map, the don't care minterm may be assumed to be either 0 or 1.
- When simplifying the function, we can choose to include each don't care minterm with either the 1's or the 0's, depending on which combination gives the simplest expression.

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS – WITH DON'T CARES

$$F(w, x, y, z) = \sum m(0, 7, 8, 9, 10, 12) + \sum d(2, 5, 13)$$

wx \ yz	00	01	11	10
00	1 (0)	0 (1)	0 (3)	X (2)
01	0 (4)	X (5)	1 (7)	0 (6)
11	1 (12)	X (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	0 (11)	1 (10)



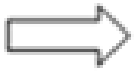
$$F(w, x, y, z) = w'xz + wy' + x'z'.$$

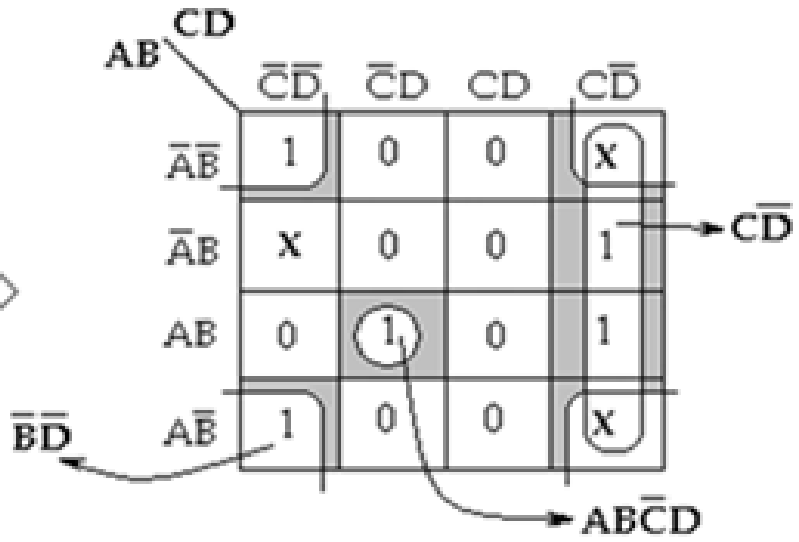
GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF SOP FORMS – WITH DON'T CARES

$$F(A, B, C, D) = \sum m\,(0, 6, 8, 13, 14) + \sum d\,(2, 4, 10)$$

AB \ CD	00	01	11	10
00	1 (0)	0 (1)	0 (3)	X (2)
01	X (4)	0 (5)	0 (7)	1 (6)
11	0 (12)	1 (13)	0 (15)	1 (14)
10	1 (8)	0 (9)	0 (11)	X (10)





$$F(A, B, C, D) = CD' + B'D' + A\,B\,C'D$$

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF POS FORMS

- For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression & the cells that do not have a 0 are the cells for which the expression is 1.

- Steps for the POS K-mapping process.

  Step 1: Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.

  Step 2: As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.

  Step3: Other procedure are similar to SOP K-map method

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF POS FORMS - EXAMPLE

EXAMPLE: *Simplify the expression*

$$F(W,X,Y,Z) = \Pi\ (0,\ 1,\ 4,\ 5,\ 6,\ 8,\ 9,\ 12,\ 13,\ 14)$$

STEP 1: Express given values in terms their maxterms

$$F(A,B,C,D) = (M0.M1.M4.M5.M6.M8.M9.M12.M13.M14)$$

STEP 2: Draw the four variable K-Map structure

	C+D	C+D'	C'+D'	C'+D
A+B	M0	M1	M3	M2
A+B'	M4	M5	M7	M6
A'+B'	M12	M13	M15	M14
A'+B	M8	M9	M11	M10

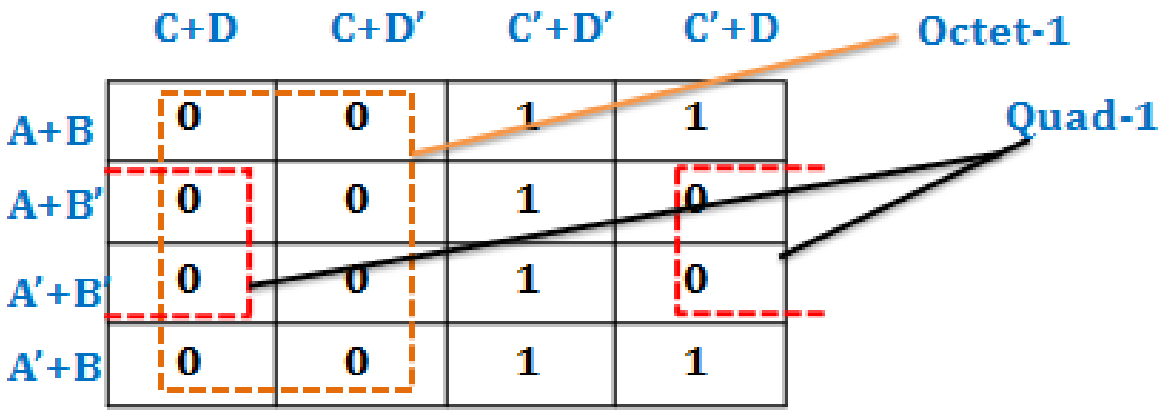
GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF POS FORMS - EXAMPLE

STEP 3: Plot the k-map for the given maxterms. Put 0's for the given maxterm values and 1's in other places

(M0.M1.M4.M5.M6.M8.M9.M12.M13.M14)

STEP 4: Do the grouping operation



STEP 5: Find the simplified boolean expressions from the grouping

F = Octet-1 . quad-1

= (C) . (B'+D)

GATE MINIMIZATION USING K-MAP

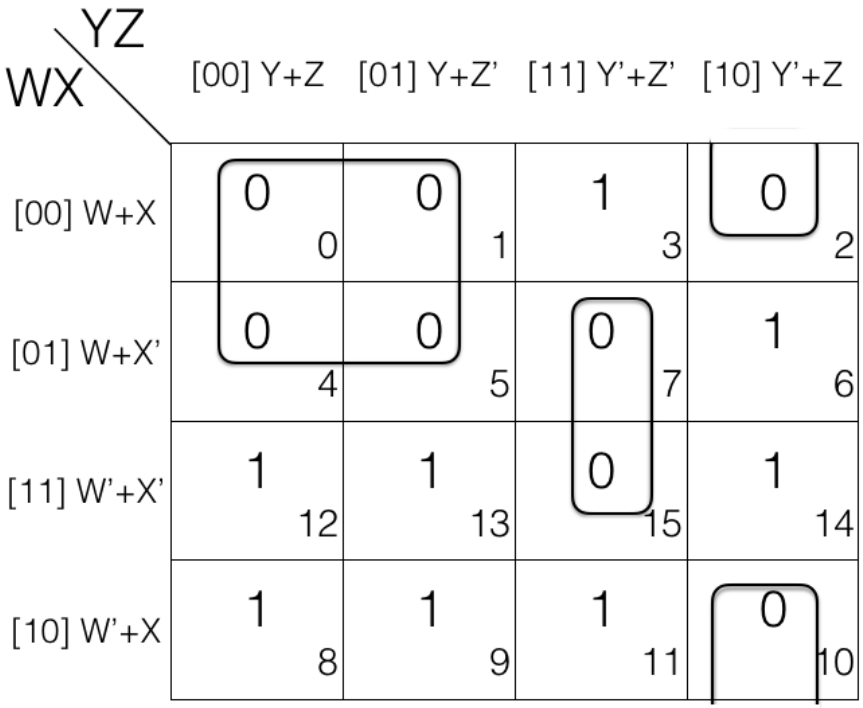
K-MAP SIMPLIFICATION OF POS FORMS

- Reduce F(W,X,Y,Z) = ∏(0,1,2,4,5,7,10,15) using K-map

WX \ YZ	[00] Y+Z	[01] Y+Z'	[11] Y'+Z'	[10] Y'+Z
[00] W+X	0 (0)	0 (1)	1 (3)	0 (2)
[01] W+X'	0 (4)	0 (5)	0 (7)	1 (6)
[11] W'+X'	1 (12)	1 (13)	0 (15)	1 (14)
[10] W'+X	1 (8)	1 (9)	1 (11)	0 (10)

GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF POS FORMS



F = (W+Y) . (X'+Y'+Z') . (X+Y'+Z)

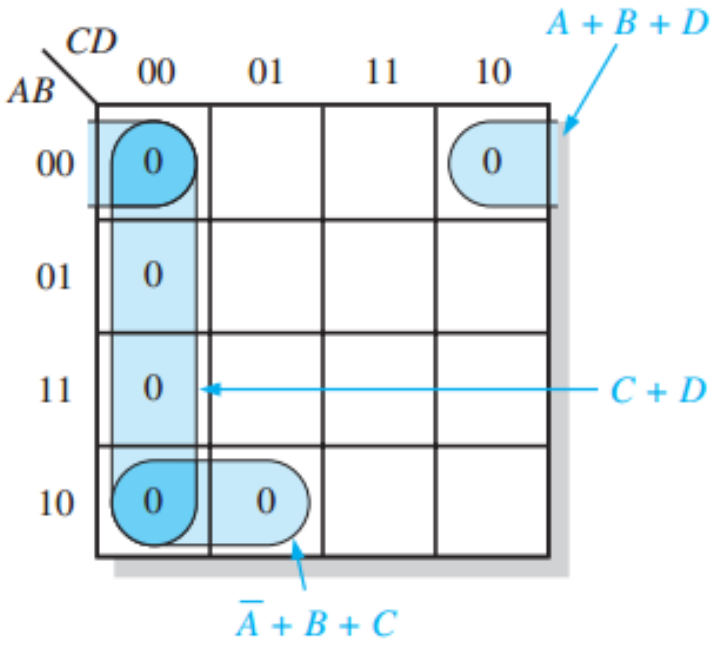
GATE MINIMIZATION USING K-MAP

K-MAP SIMPLIFICATION OF POS FORMS

- Use a Karnaugh map to minimize the following POS expression:

  (B+ C+D)(A+B+C'+D)(A'+B+C+D')(A+B'+C+D)(A'+B'+ C+D)

The first term must be expanded into A' + B + C + D and A + B + C + D to get a standard POS expression, which is then mapped;



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

GATE MINIMIZATION USING K-MAP

KARNAUGH MAP MINIMIZATION - EXERCISE

Using a Karnaugh map, simplify the following functions and implement them with basic gates.

(a) F (A, B, C, D) = Σm (0, 2, 3, 6, 7, 8, 10, 11, 12, 15)
(b) F (A, B, C, D) = Σm (0, 2, 3, 5, 7, 8, 13) + d (1, 6, 12)
(c) F (A, B, C, D) = Σm (1, 7, 9, 10, 12, 13, 14, 15) + d (4, 5, 8)
(d) F (A, B, C, D) = πM (0, 8, 10, 11, 14) + d (6)
(e) F (A, B, C, D) = π M(2, 8, 11, 15) + d (3, 12, 14)
(f) F (W, X, Y, Z) = πM (0, 2, 6, 11, 13, 15) + d (1, 9, 10, 14)

Prepare a Karnaugh map for the following functions.

(a) F = ABC + A'BC + B'C'
(b) F = A + B + C'
(c) Y = AB + B'CD

Using the Karnaugh map method, simplify the following functions, obtain their sum of the products form, and product of the sums form. Realize them with basic gates.

(a) F (W, X, Y, Z) = Σ (1, 3, 4, 5, 6, 7, 9, 12, 13)
(b) F (W, X, Y, Z) = Σ (1, 5, 6, 7, 11, 12, 13, 15)

NAND & NOR IMPLEMENTATION

NAND & NOR IMPLEMENTATION

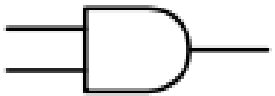
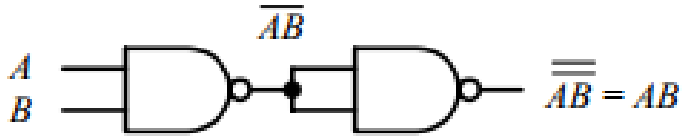
UNIVERSAL LOGIC GATES

- OR, AND and NOT gates are the three basic logic gates as they together can be used to construct the logic circuit for any given Boolean expression.
- The combination of NAND gates or a combination of NOR gates can be used to perform functions of any of the basic logic gates.
- And also, NAND and NOR gates are referred to as universal gates because each alone can be combined together with itself to form all other possible logic gates

NAND & NOR IMPLEMENTATION

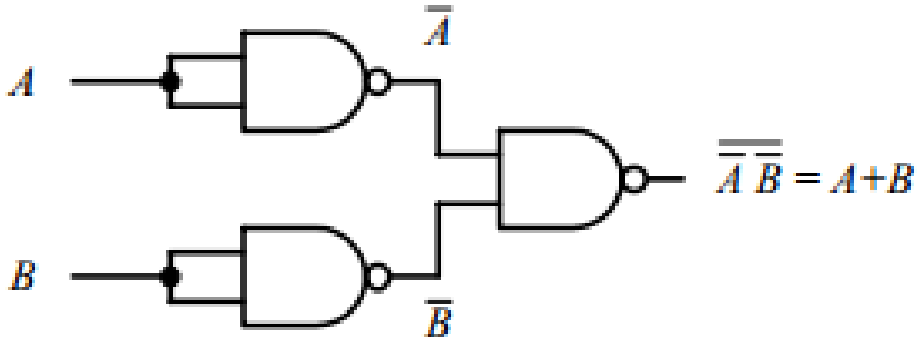
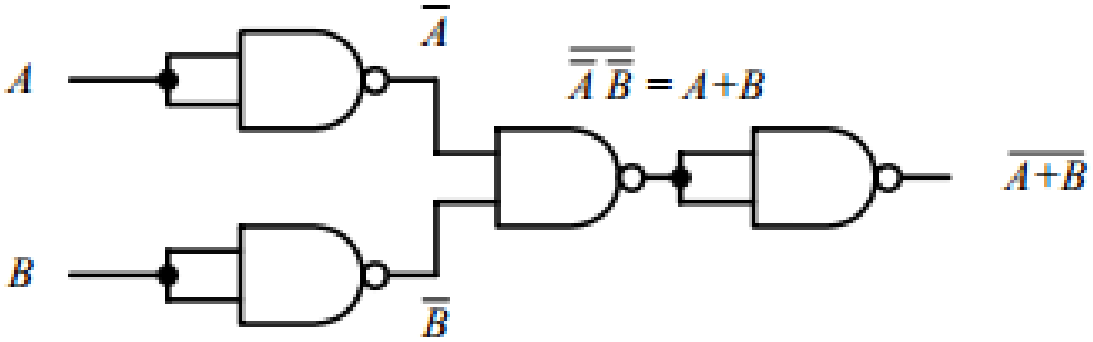
NAND GATE AS UNIVERSAL LOGIC GATE

- The NAND gate is a universal gate because it can be used to produce any of the other logic gates function.

Logic gate	NAND equivalent circuit
NOT 	
AND 	

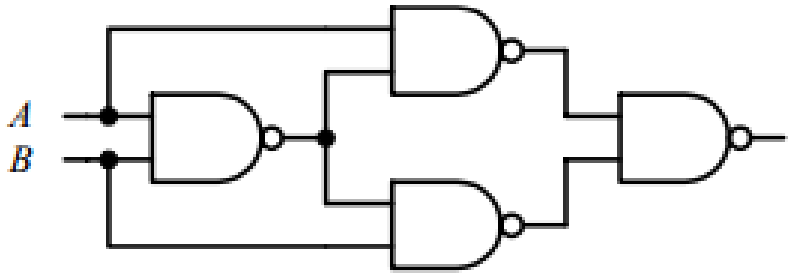
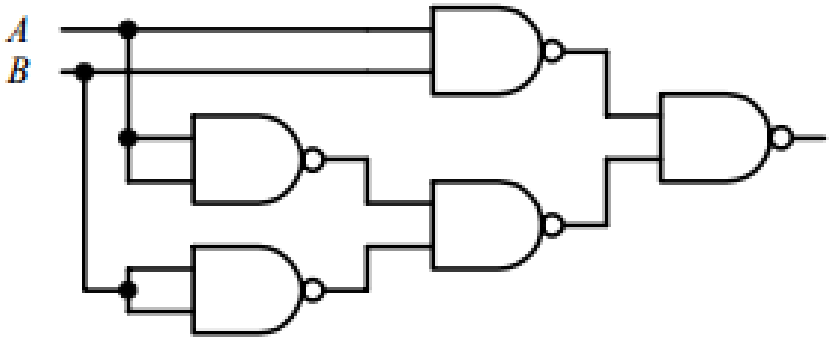
NAND & NOR IMPLEMENTATION

NAND GATE AS UNIVERSAL LOGIC GATE

OR 	
NOR 	

NAND & NOR IMPLEMENTATION

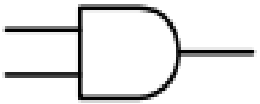
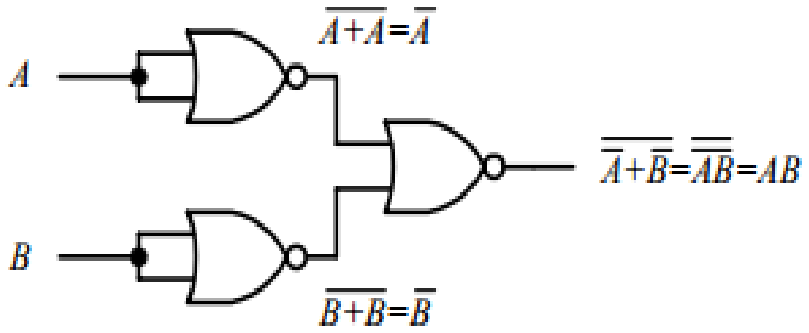
NAND GATE AS UNIVERSAL LOGIC GATE

XOR 	
XNOR 	

NAND & NOR IMPLEMENTATION

NOR GATE AS UNIVERSAL LOGIC GATE

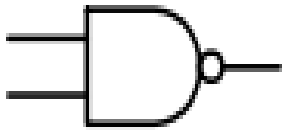
- Like the NAND gate, the NOR gate can be used to produce any logic function.

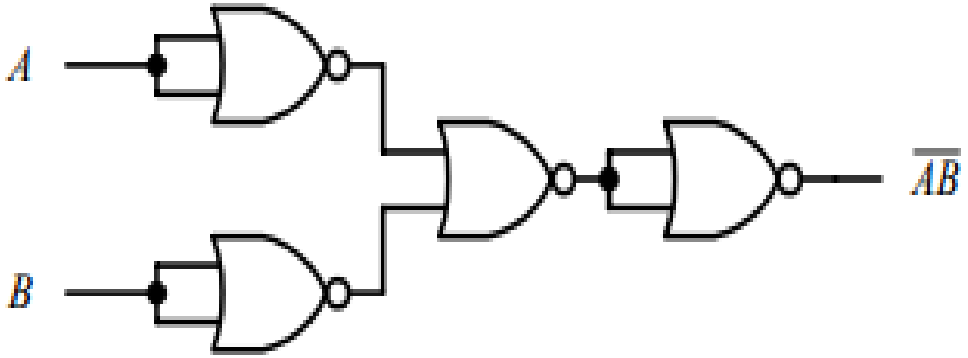
Logic gate	NOR equivalent circuit
NOT 	
AND 	

NAND & NOR IMPLEMENTATION

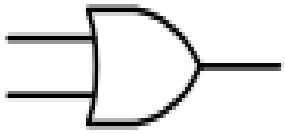
NOR GATE AS UNIVERSAL LOGIC GATE

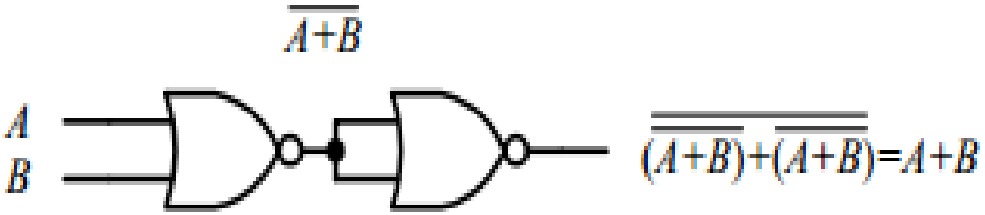
NAND





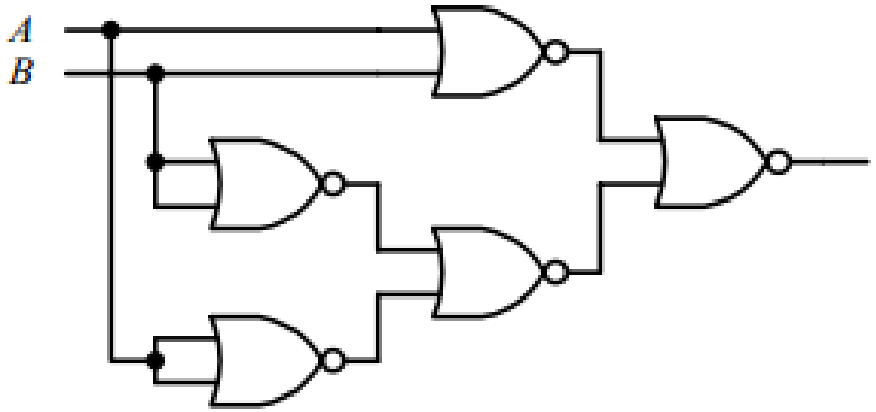
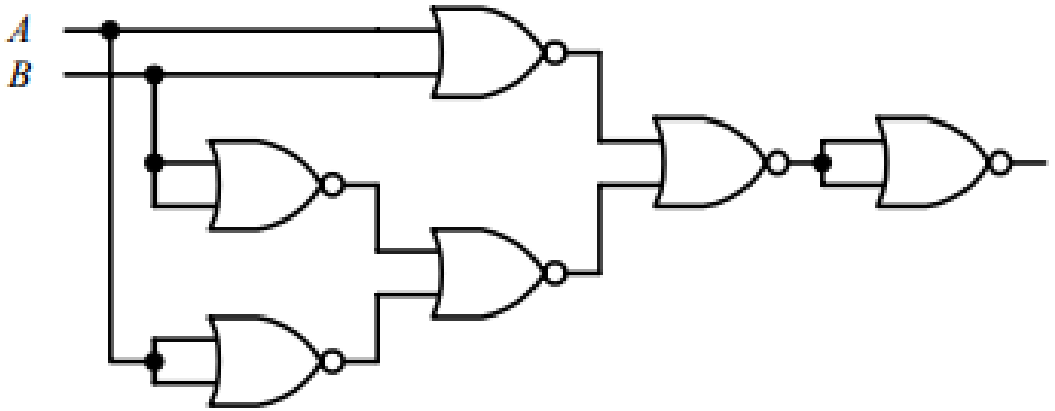
OR





NAND & NOR IMPLEMENTATION

NOR GATE AS UNIVERSAL LOGIC GATE

XOR 	
XNOR 	

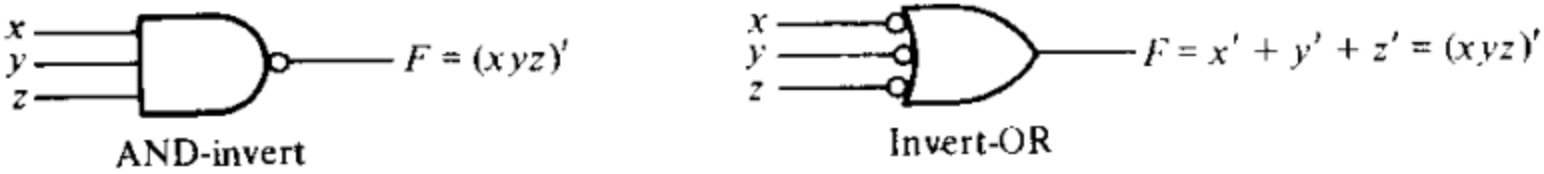
NAND & NOR IMPLEMENTATION

NAND & NOR IMPLEMETATION

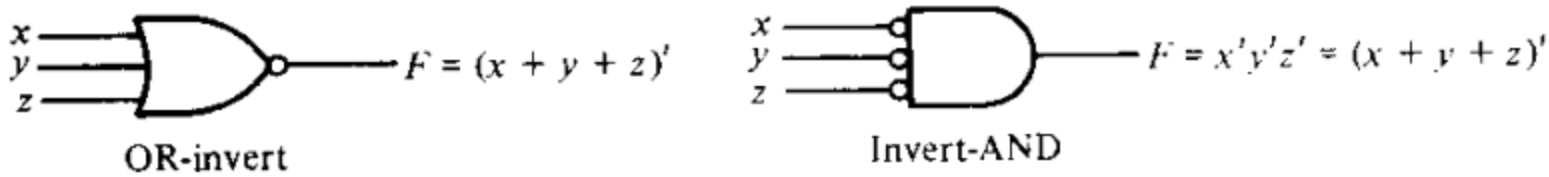
- Digital circuits are more frequently constructed using NAND and NOR gates than with AND and OR gates
- NAND and NOR gates are easier to fabricated with other electronics components and these are the basic gates used in most digital logic IC families
- To ease the conversion of NAND, NOR logic it is convenient to define other graphical symbol for these gates

NAND & NOR IMPLEMENTATION

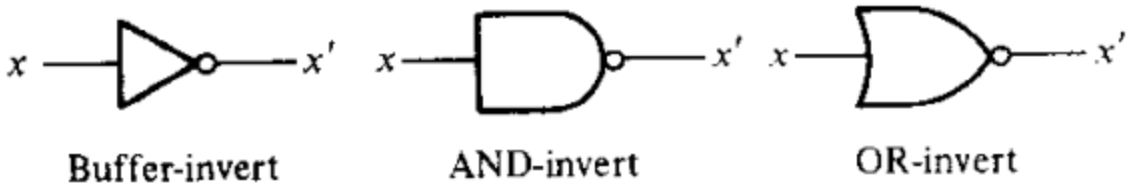
NAND & NOR IMPLEMETATION



(a) Two graphic symbols for NAND gate.



(b) Two graphic symbols for NOR gate.



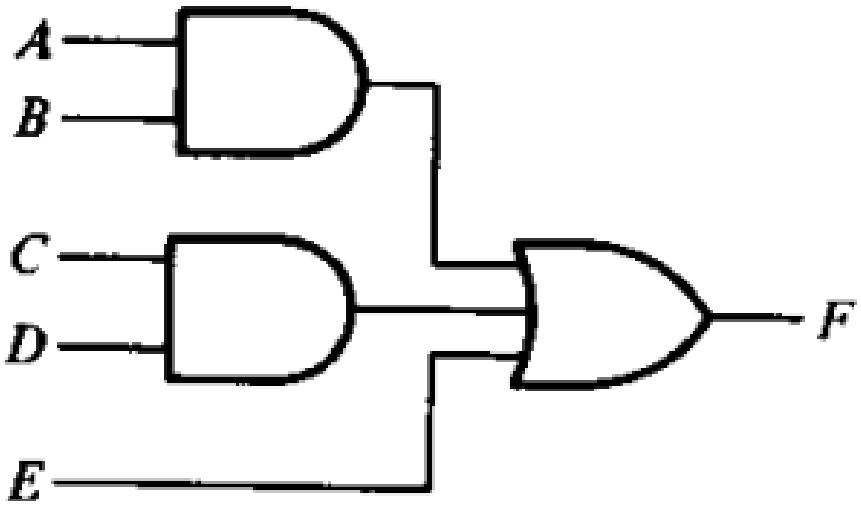
(c) Three graphic symbols for inverter.

NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION

- Implement the following logical expression into an equivalent NAND gate based logic circuit

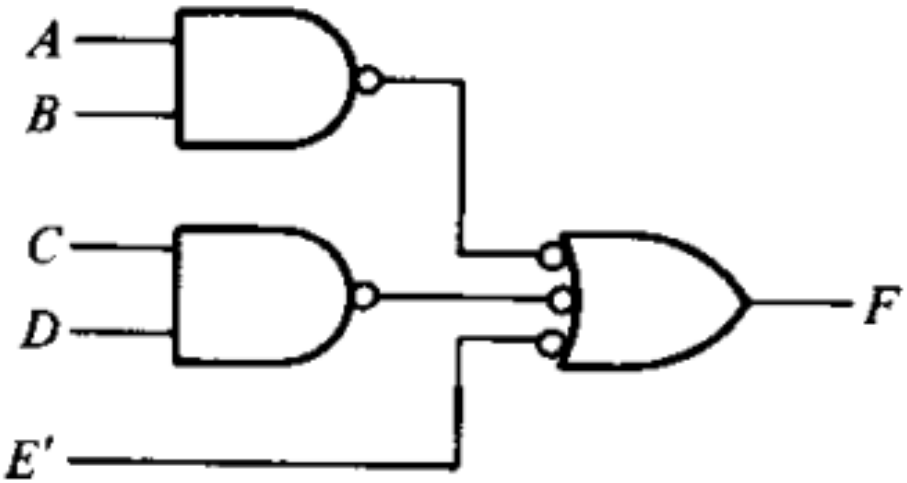
$$F = AB + CD + E$$



(a) AND-OR

NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION

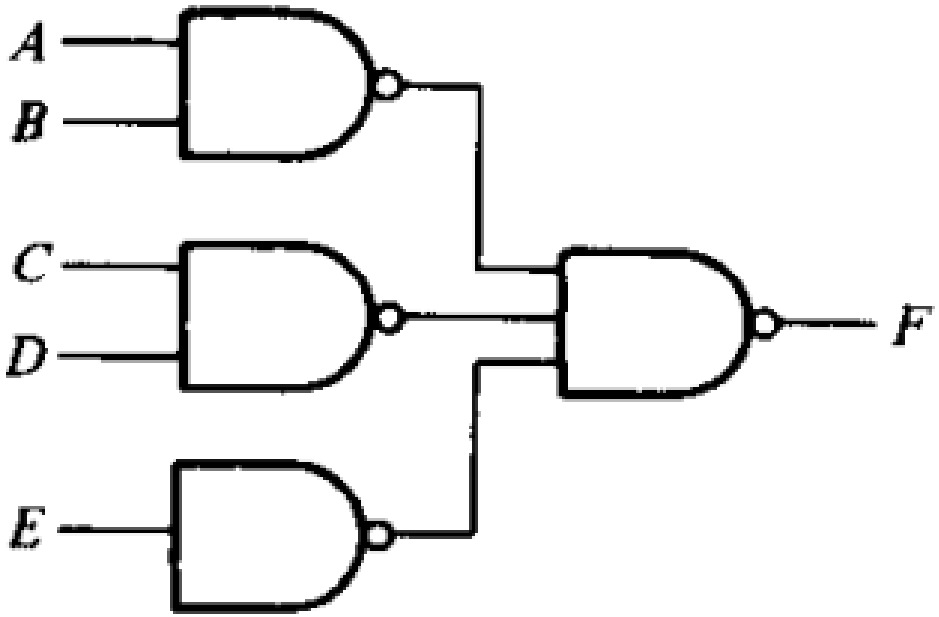


(b) NAND-NAND

AND is gate is replaced by NAND gate and OR gate is replaced by NAND gate with inverted OR gate

NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION



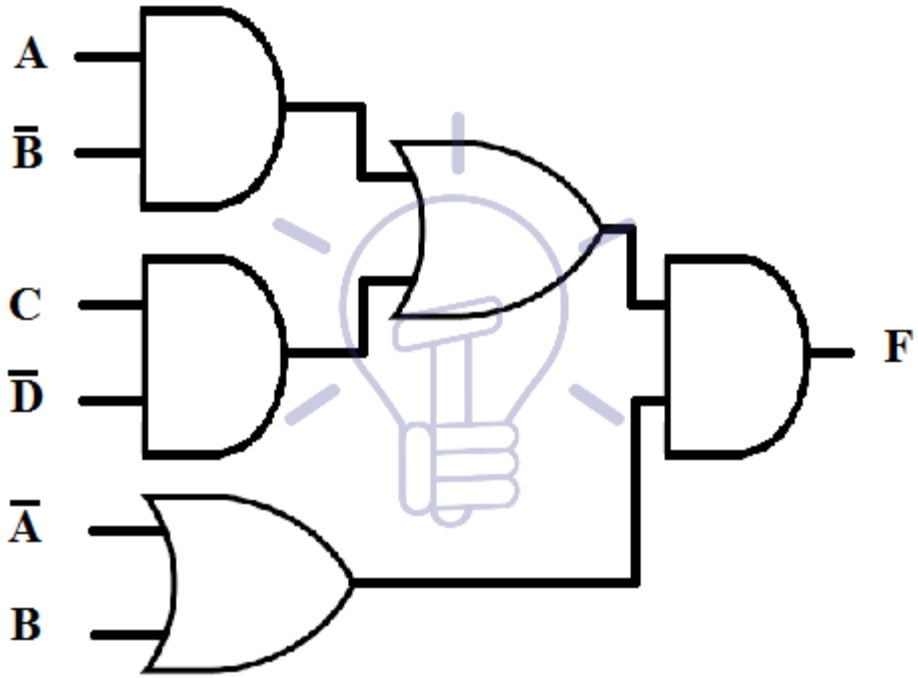
(c) NAND-NAND

$$F = [(AB)' \cdot (CD)' \cdot E']' = AB + CD + E$$

NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION

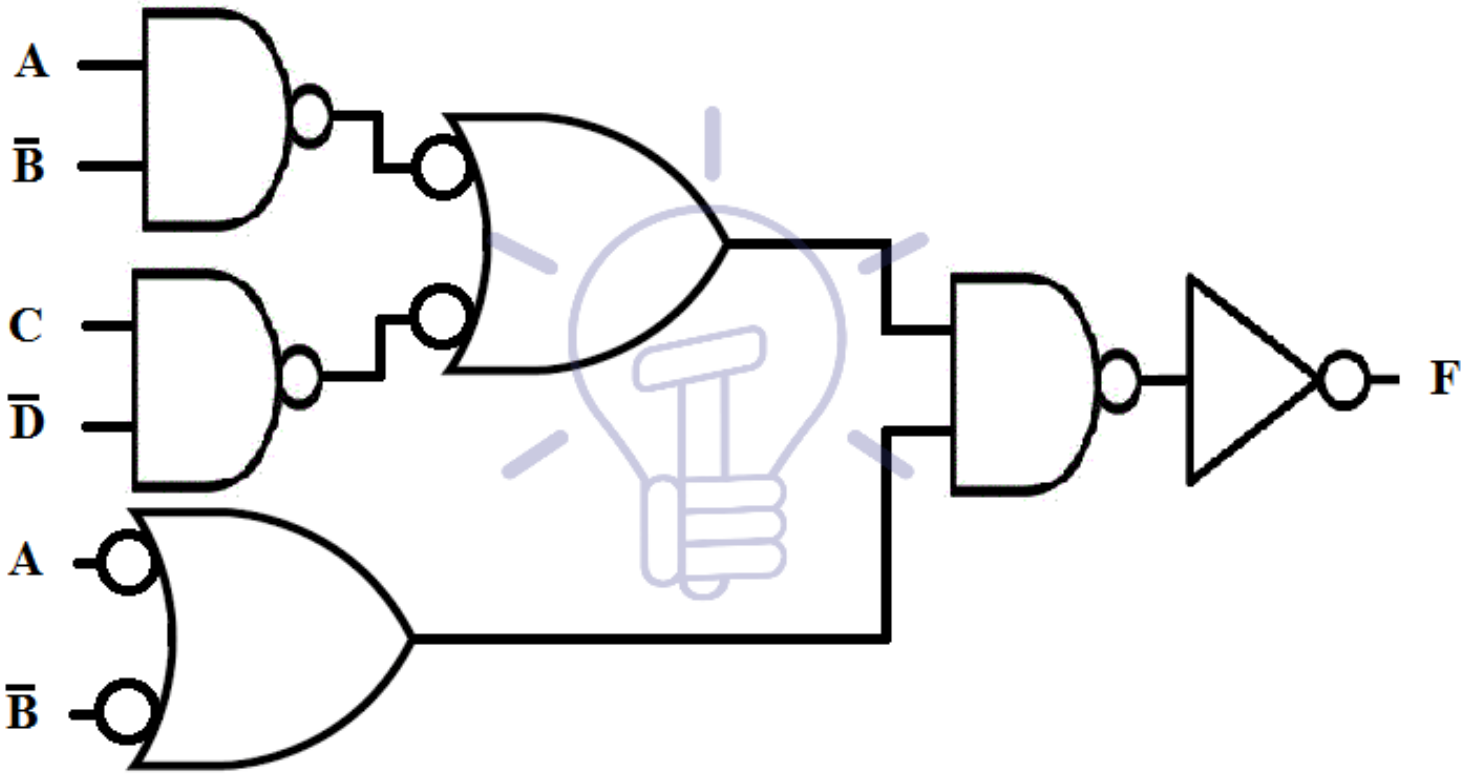
$$F = (AB'+CD') (A'+B)$$



Three-level implementation

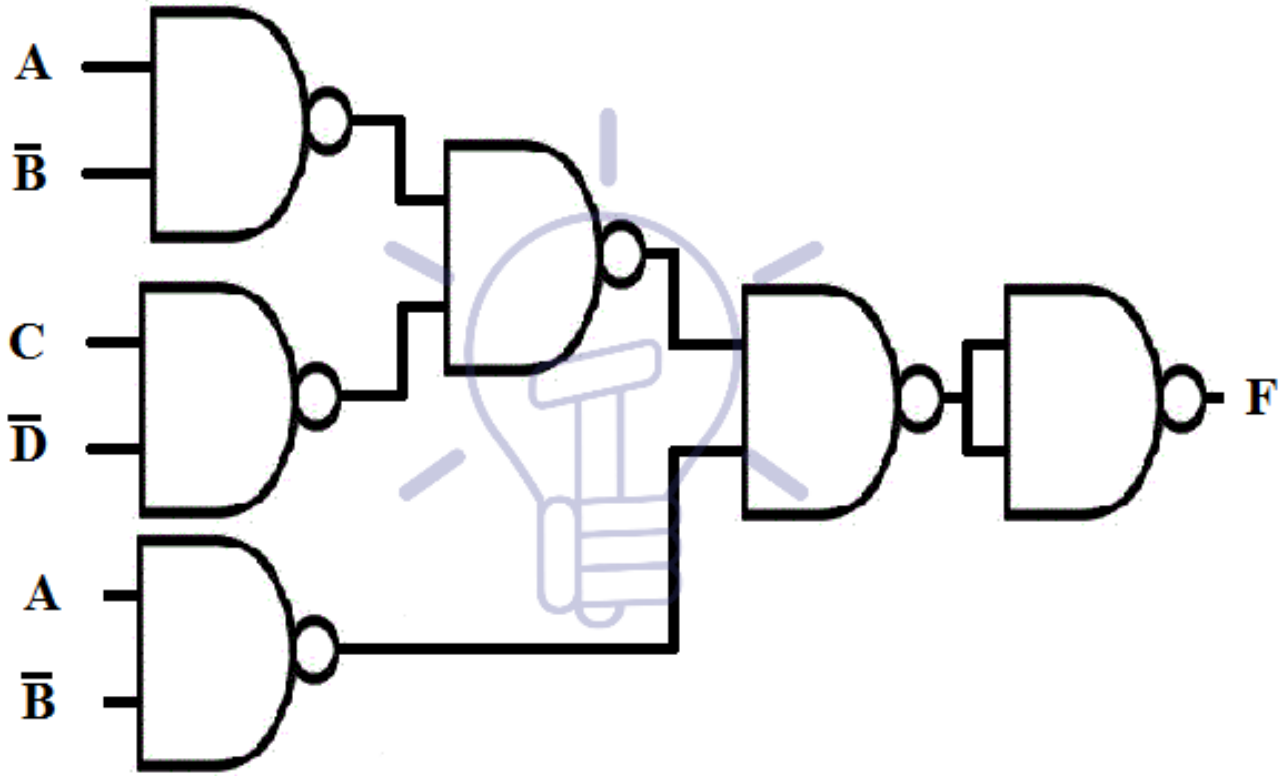
NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION



NAND & NOR IMPLEMENTATION

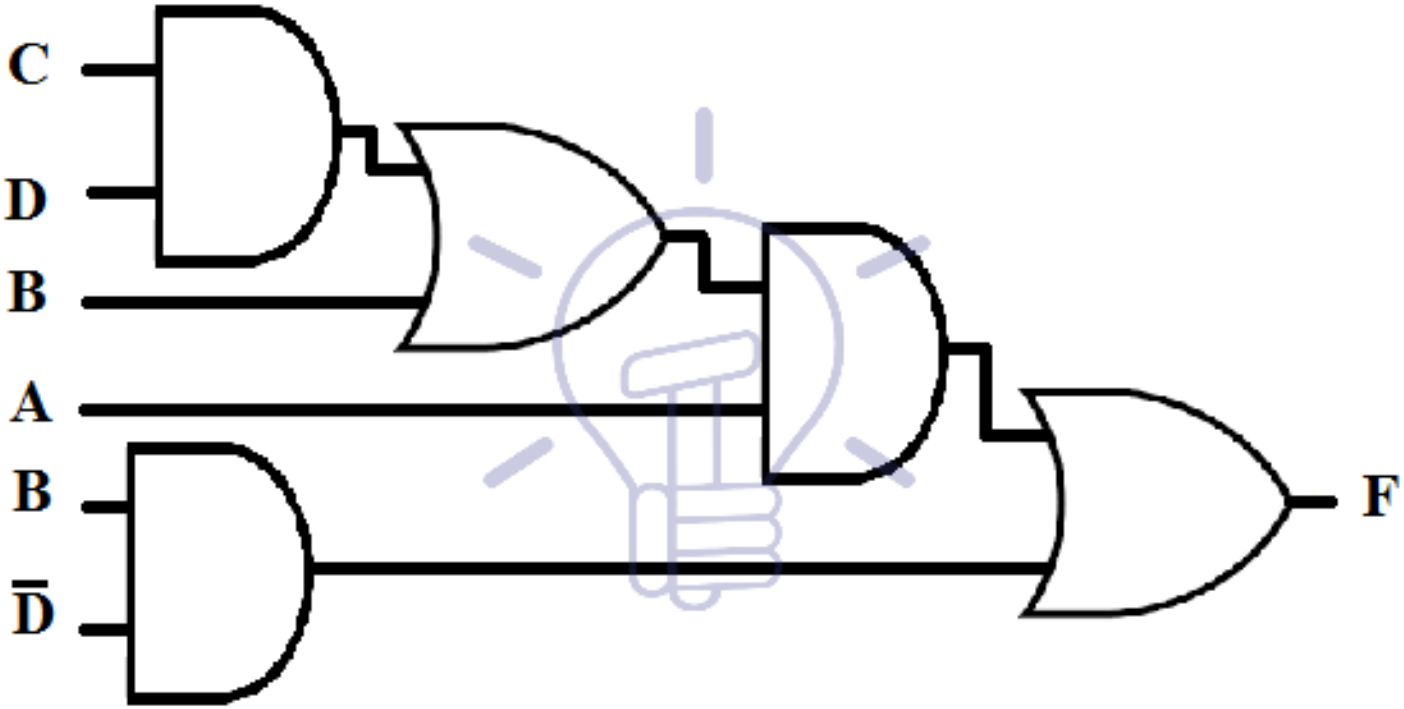
NAND IMPLEMETATION



NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION - EXAMPLES

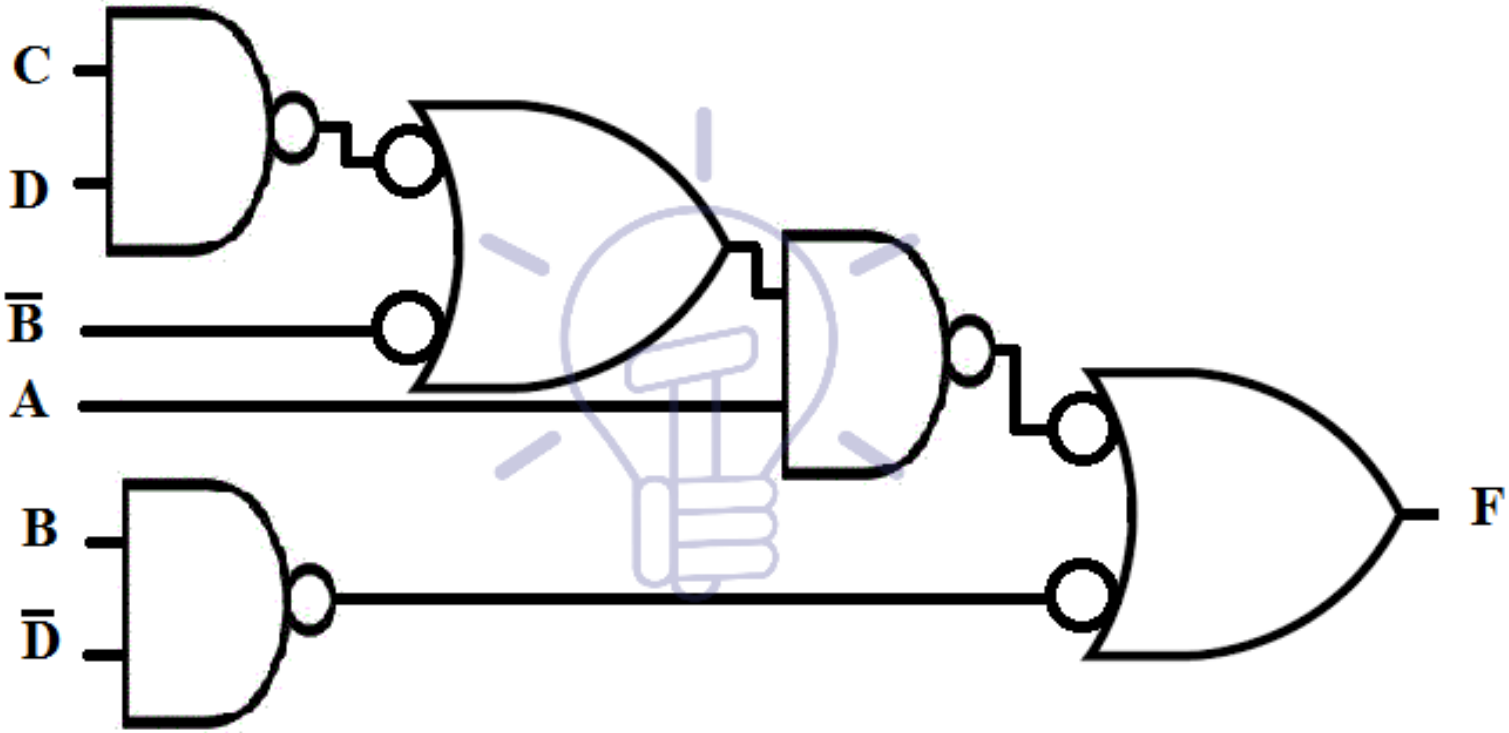
$$F = A (B + CD) + BD'$$



MULTI-LEVEL NAND Gate Implementation

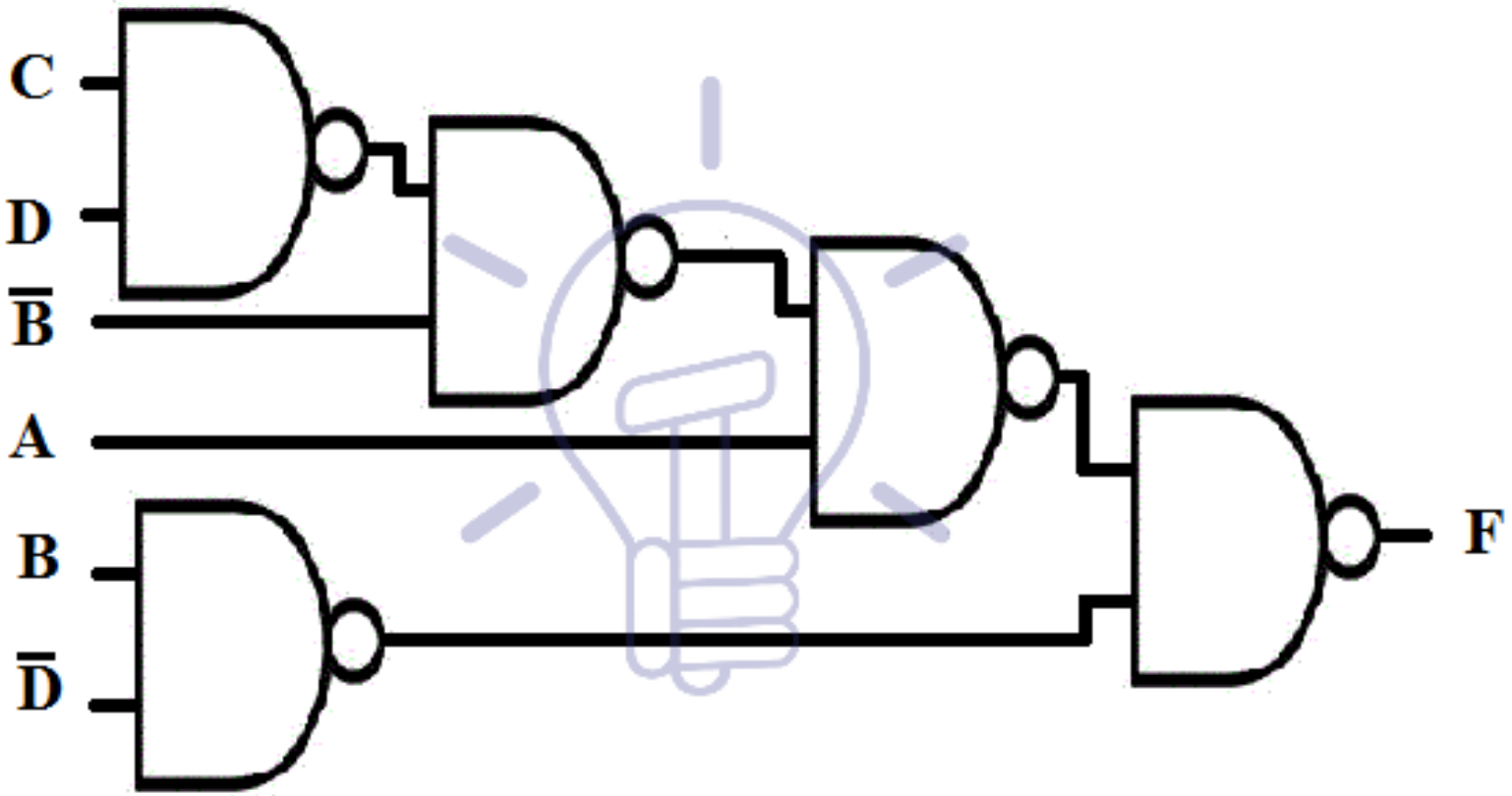
NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION - EXAMPLES



NAND & NOR IMPLEMENTATION

NAND IMPLEMETATION - EXAMPLES



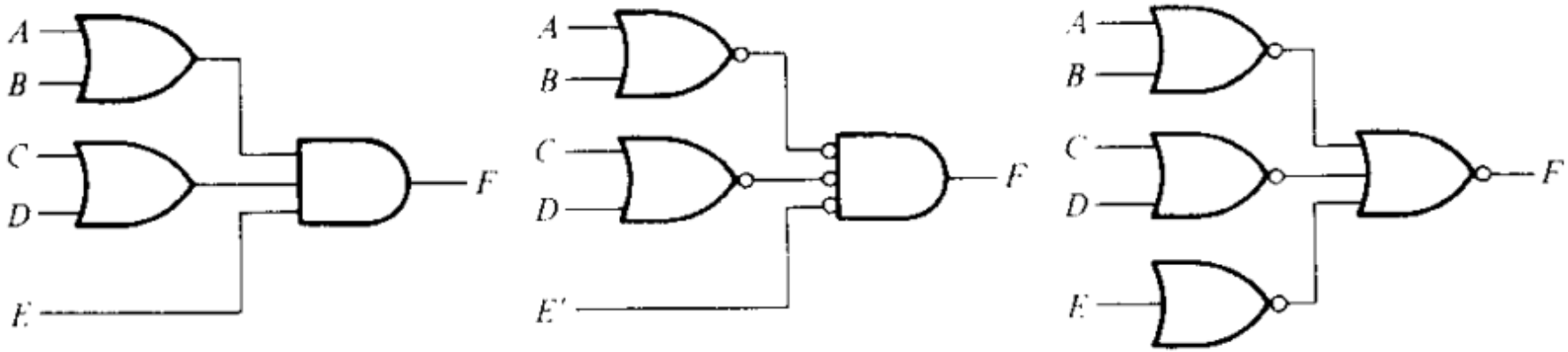
NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION

- NOR function is a dual of NAND function, so all procedures and rules followed for NOR will be dual of NAND logic realization
- Implementation of boolean function with NOR gates requires that the function to be simplified in POS forms
- The POS specifies a group of OR gates for the sum terms, followed by AND gates to produce the product
- So, it transfers the OR-AND logic circuits into the NOR-NOR logic circuits.

NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION

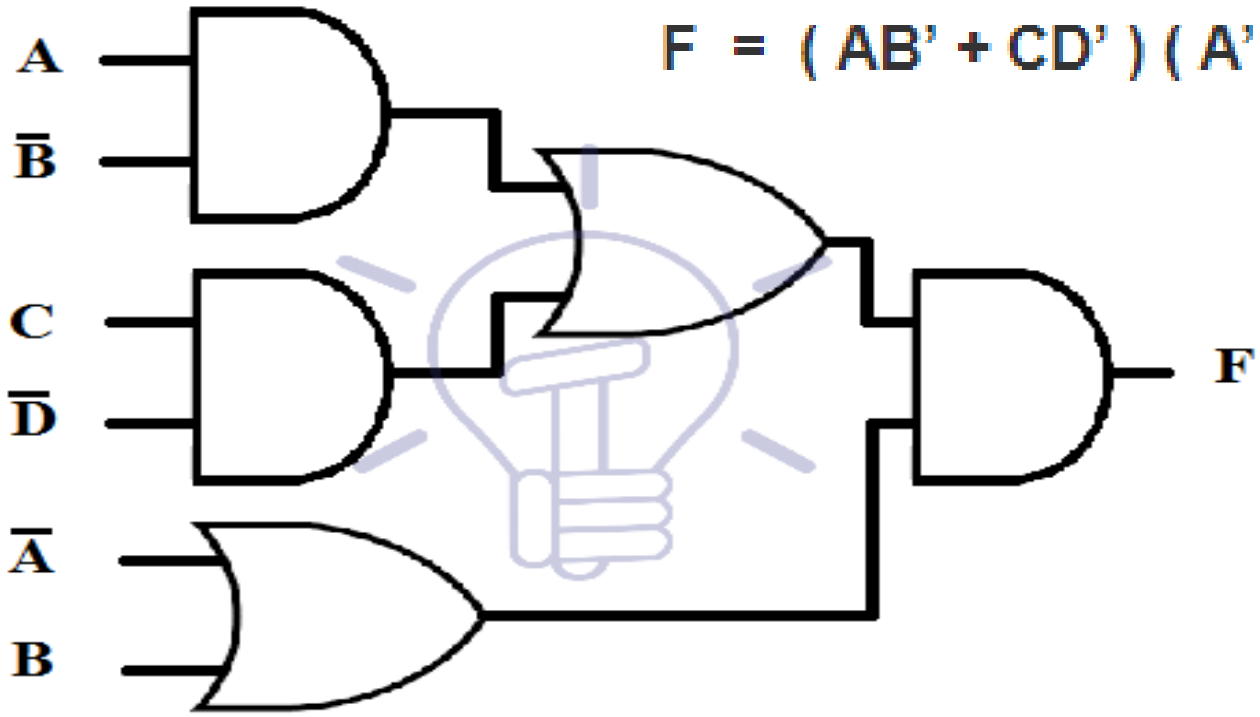


(a) OR-AND (b) NOR-NOR (c) NOR-NOR

$$F = (A + B)(C + D)E$$

NAND & NOR IMPLEMENTATION

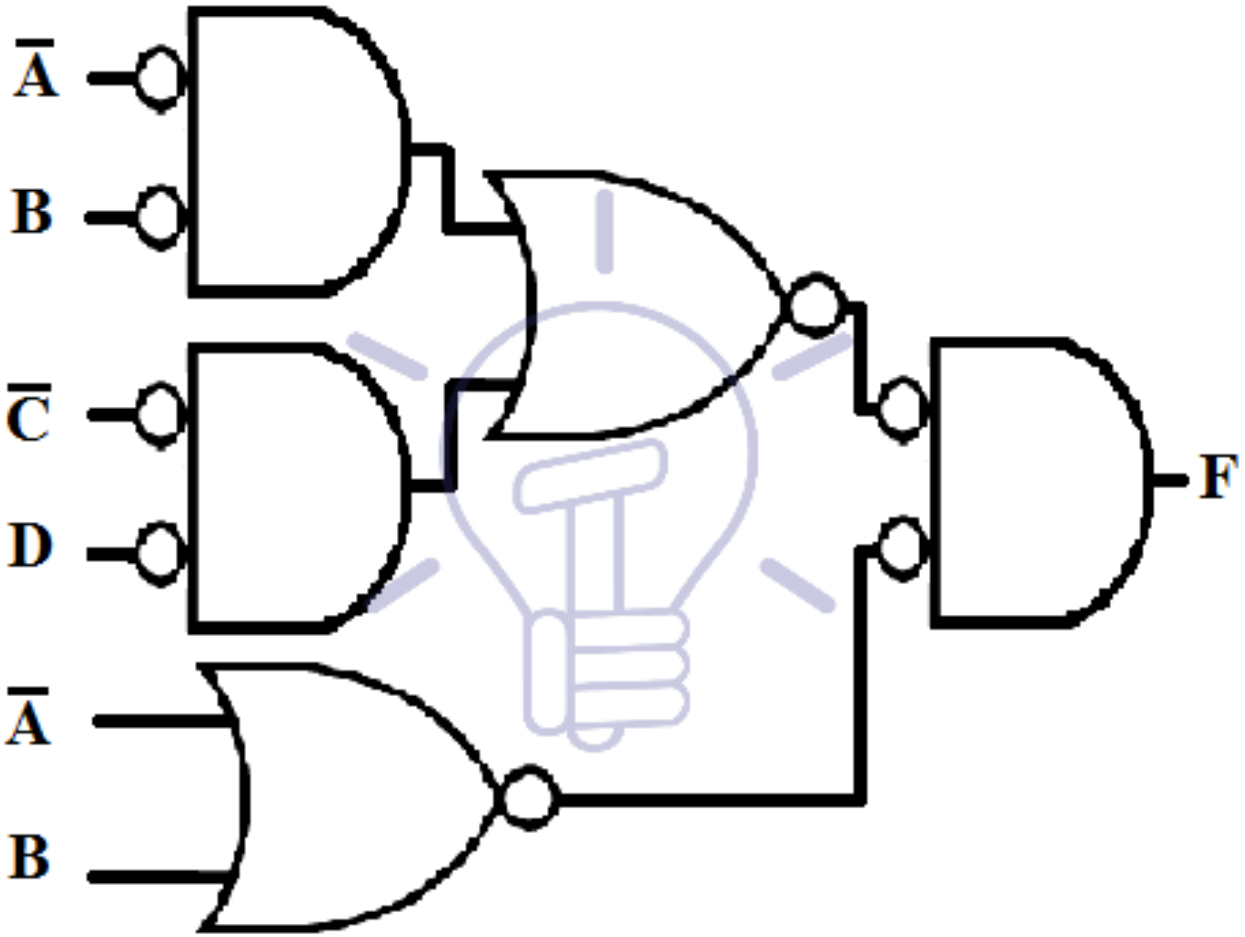
NOR IMPLEMETATION – Other Examples



3-Level NOR Gate Implementation

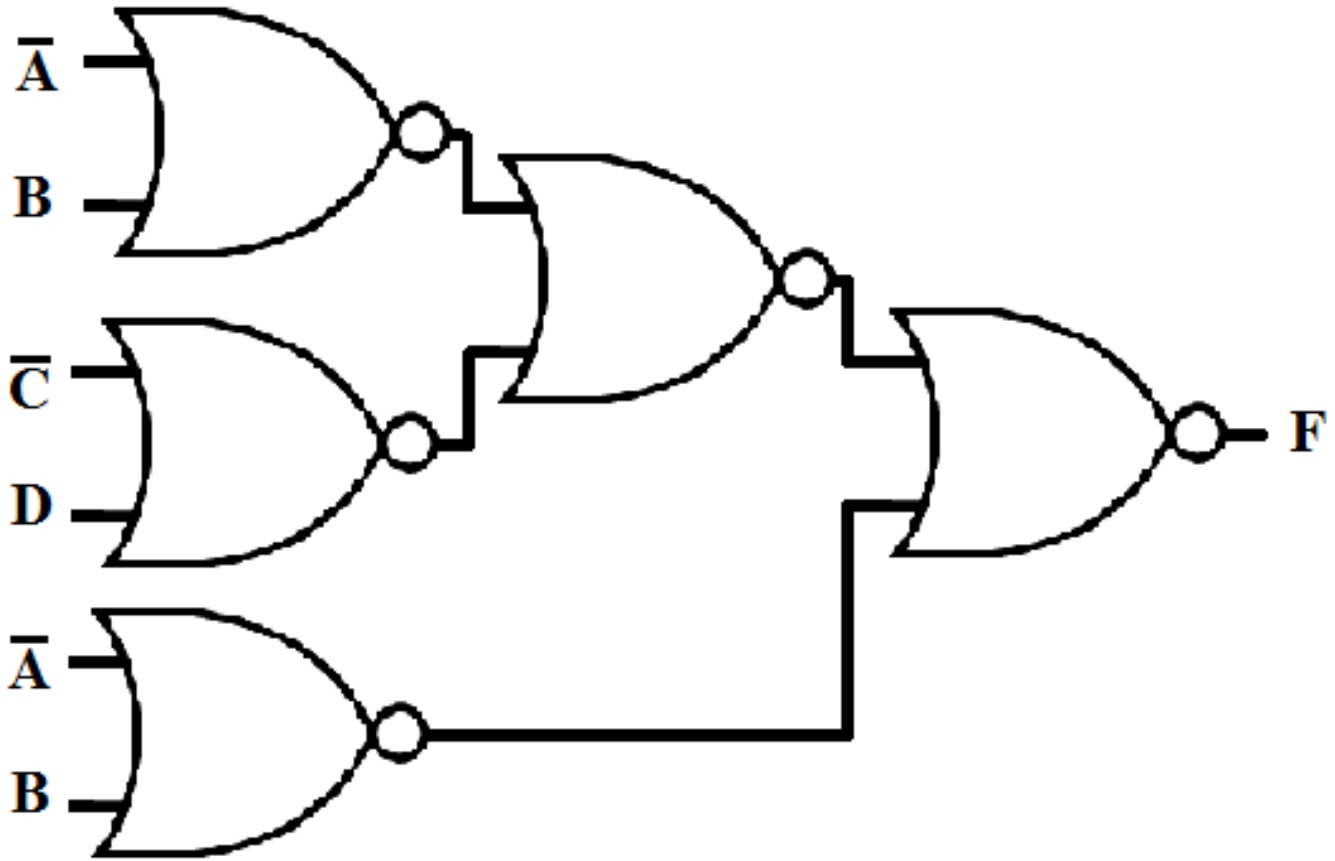
NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION – Other Examples



NAND & NOR IMPLEMENTATION

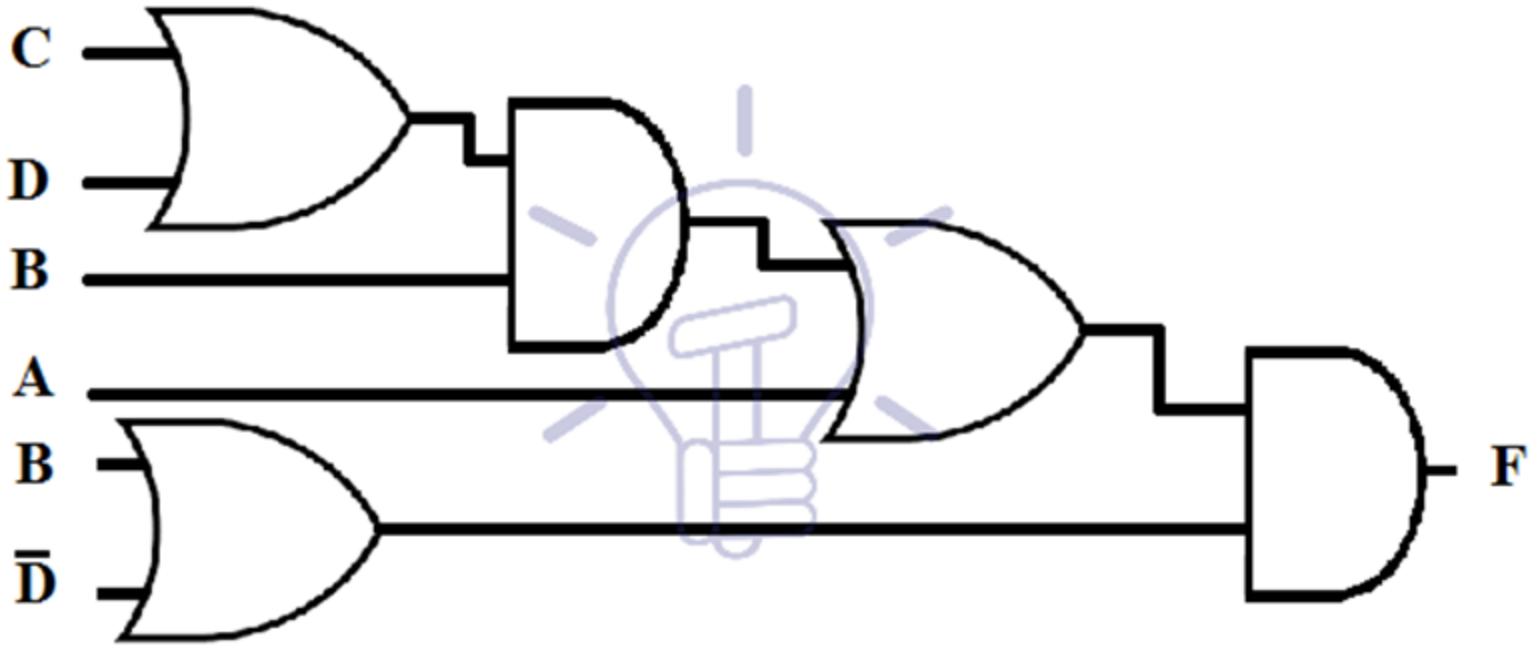
NOR IMPLEMETATION – Other Examples



NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION – Other Examples

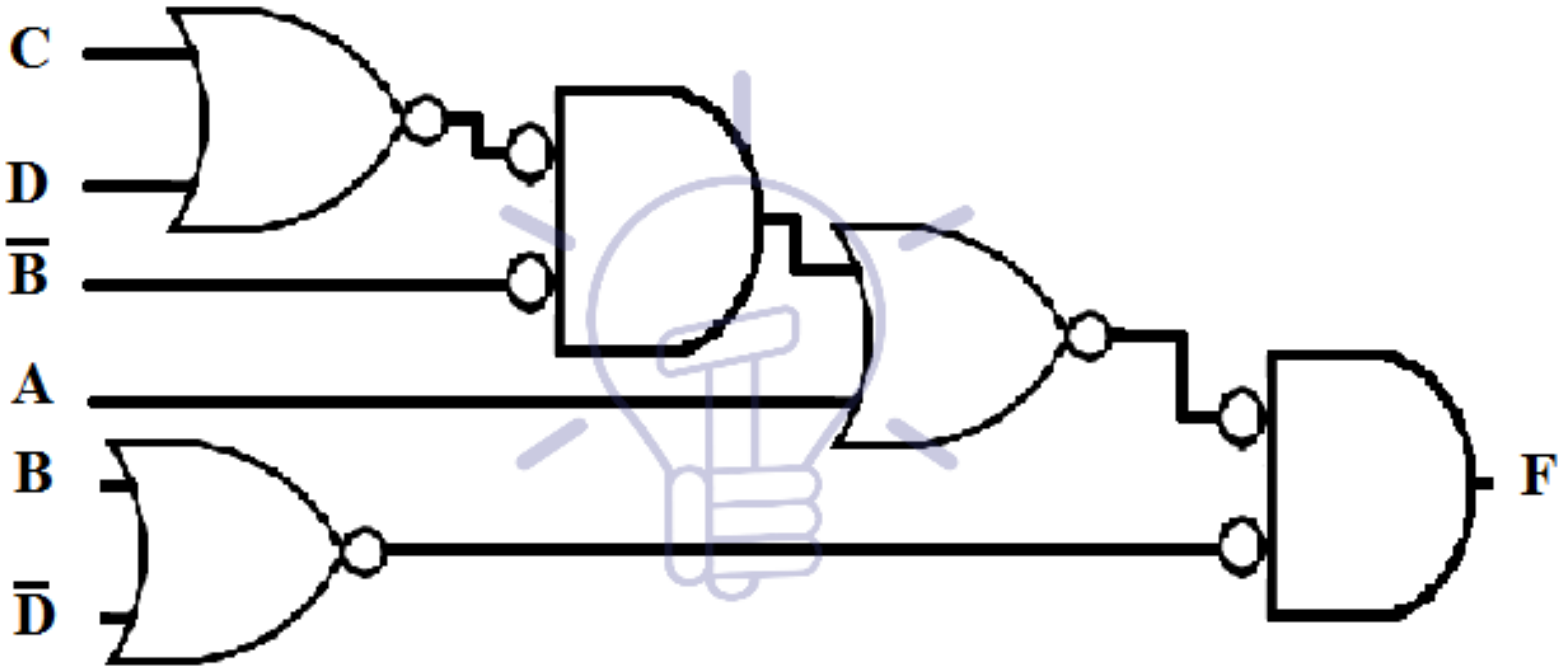
$$F = (A + B (C + D)) (B + D')$$



Multi-level NOR Gate Implementation

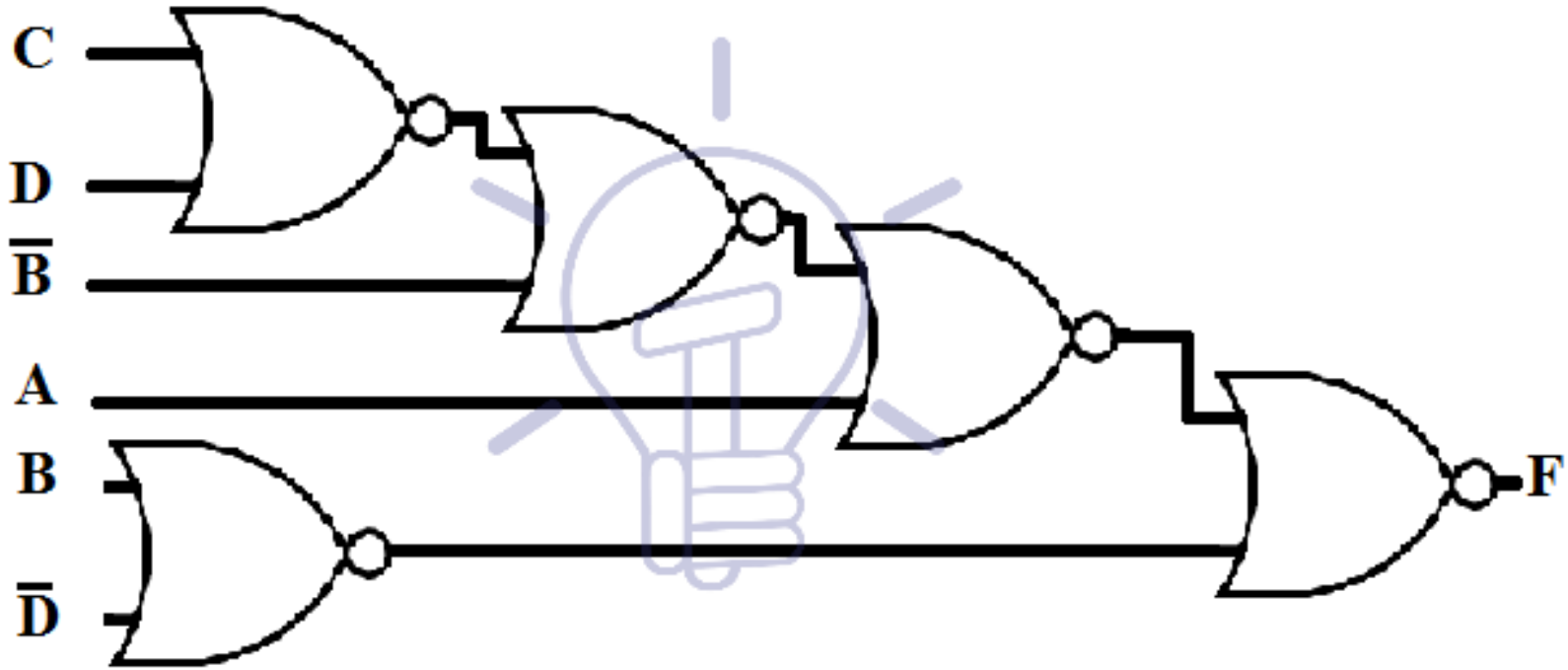
NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION – Other Examples



NAND & NOR IMPLEMENTATION

NOR IMPLEMETATION – Other Examples



THANK YOU