



Sri
SAI RAM
ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8351

Digital Principles and System Design
(Common to CSE & IT)

UNIT I BOOLEAN ALGEBRA AND LOGIC GATES

1.9 NAND and NOR Implementation

NAND and NOR Implementation

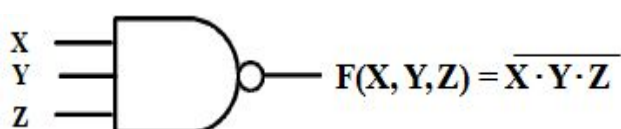
Implementation of Universal Gates

- ❑ The ICs are made up number of logic gates apart from other components.
- ❑ The usage of AND and OR gates cause difficulties and complicates the circuitry. Thus, the methods to reduce the number of logic gates that are employed in a circuit have to be evolved and considered.
- ❑ One such method is implementation of logic expressions using NAND and NOR gates.
- ❑ NAND and NOR can perform functions of all other logic gates and are hence termed as **Universal gates**.

NAND GATES

NAND gate is a universal building block.

- The basic positive logic NAND gate is denoted by the following symbol.

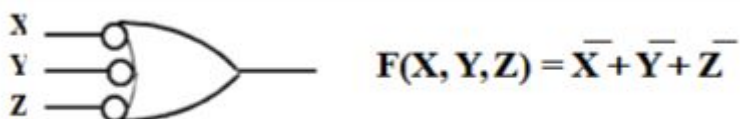


- NAND comes from NOT AND, i. e., the AND function with a NOT applied. The symbol for a NAND gate, an AND-Invert.
- The small circle represents the invert function.

Apply DeMorgan's Law : $\overline{X \cdot Y \cdot Z} = \overline{X} + \overline{Y} + \overline{Z}$

Applying De Morgan's Law gives:

* Invert-OR (NAND)



Realization of Basic gates using NAND gate

Basic Gates	NAND Realization
NOT gate	
AND gate	
OR gate	

Realization of Basic gates using NAND gate

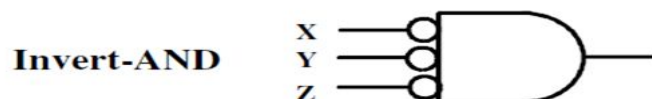
Basic Gates	NAND Realization
NOR gate	
EX-OR gate	
EX-NOR gate	

NOR Gates

The basic positive logic **NOR gate (Not-OR)** is denoted by the following symbol:



This is called the **OR-Invert**, since it is logically an **OR** function followed by an invert. By DeMorgan's Law we have the following **Invert-AND** symbol for a **NOR** gate:

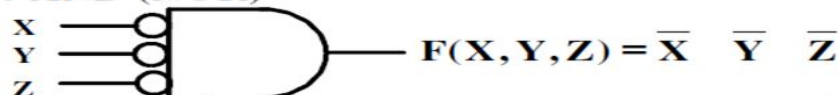


- If we apply DeMorgan's Law we get:

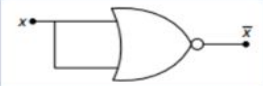
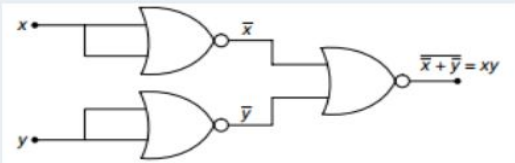
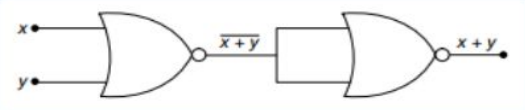
$$\overline{X + Y + Z} = \overline{X} \cdot \overline{Y} \cdot \overline{Z}$$

- Applying DeMorgan's Law gives:

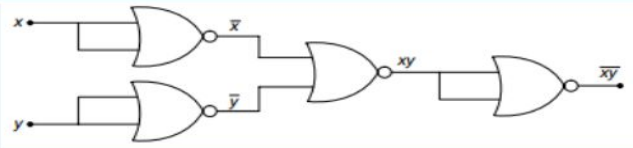
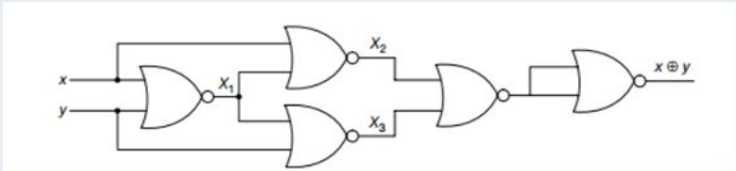
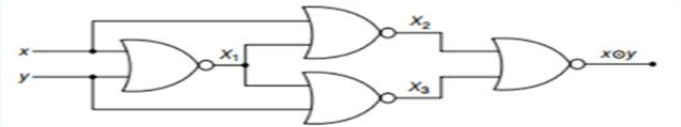
- **Invert-AND (NOR)**

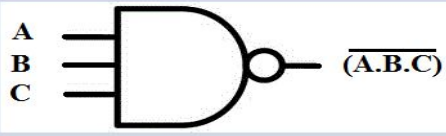
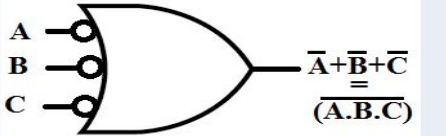
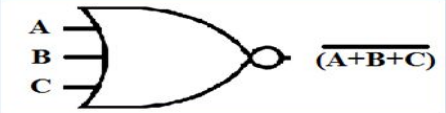
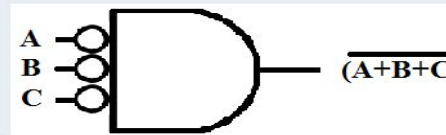


Realization of Basic gates using NOR gate

Basic Gates	NOR Realization
NOT gate	
AND gate	
OR gate	

Realization of Basic gates using NOR gate

Basic Gates	NOR Realization
NAND gate	
EX-OR gate	
EX -NOR gate	

NAND gate in terms of AND-Invert and invert OR	
AND Invert	
Invert OR	
NOR gate in terms of OR-invert and Invert -AND	
OR Invert	
Invert AND	

NAND IMPLEMENTATION

Two Level Implementation

Step 1: Check whether the Boolean function is in sum of product form, otherwise write the Boolean function in sum of product form.

Step 2: At first level, realize each product term with a NAND gate, assuming both normal and inverted inputs are available.

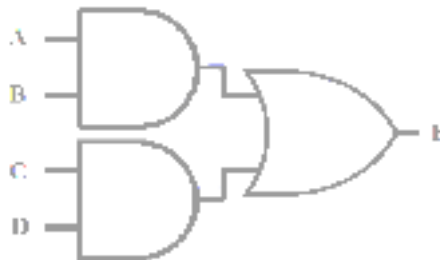
Step 3: At second level, draw a single NAND gate by connecting the outputs of first level gates to the inputs of this gate.

Step 4: A single variable term is complemented and then connected directly to an input to the second level NAND gate.

EXAMPLE 1

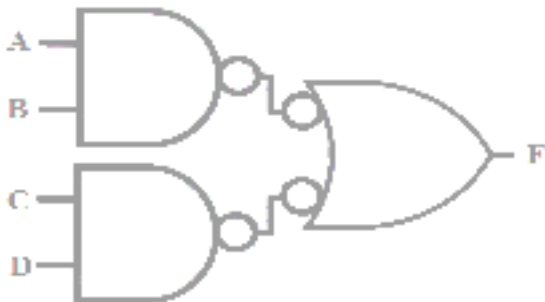
SOP function $F = A B + C D$

This SOP function is in Simplified SOP form and its AND-OR schematic is given below.

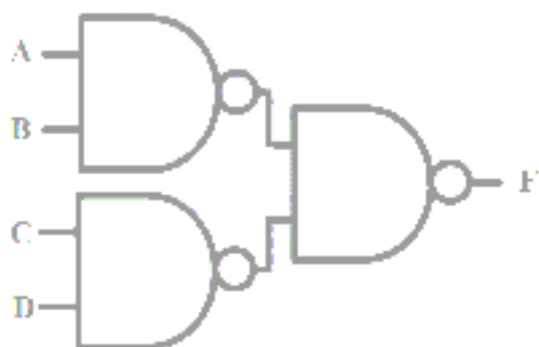


Mixed Notation

- 2nd step is to convert the AND-OR schematic into mixed notation. In mixed notation for NAND gate, AND gate is converted into AND-invert and OR gate is converted into INVERT-OR. Mixed notation design for the above function is given below.



The third step is to convert the AND-INVERT and INVERT-OR symbols into its equivalent NAND gate symbol.



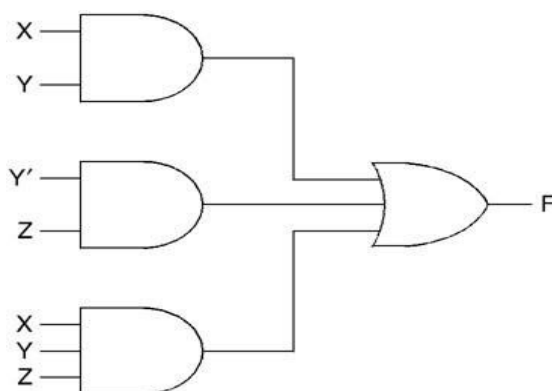
NAND Gate Conversion

EXAMPLE 2

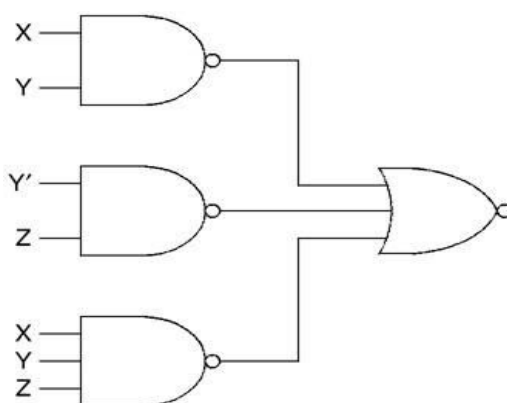
Implement the following SOP function

$$F = XY + Y'Z + XYZ$$

Being an SOP expression, it is implemented in 2-levels using AND-OR network



2-level implementation using AND-OR gates

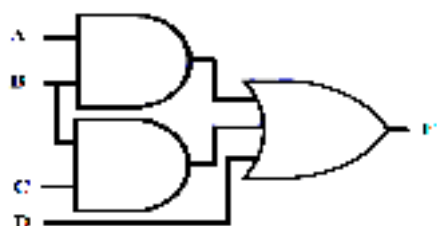


2-level implementation using NAND-NAND network

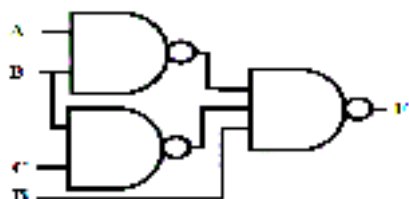
EXAMPLE 3

$$F = A B + B C + D$$

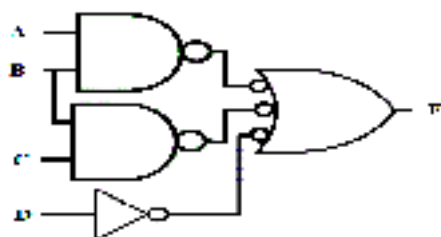
□ This function is in simplified Sum of Product form. First, we need to draw its AND-OR schematic.



□ Now we convert the above-given schematic into mixed notation by converting AND gate into AND - INVERT and OR gate into INVERT-OR.



□ Notice the single input D line to the OR gate. There is one bubble on this line. To compensate this bubble we need to either insert an inverter in this line or complement the input D if available. Then convert AND-INVERT and INVERT-OR symbol into NAND symbol as shown in the figure given below;



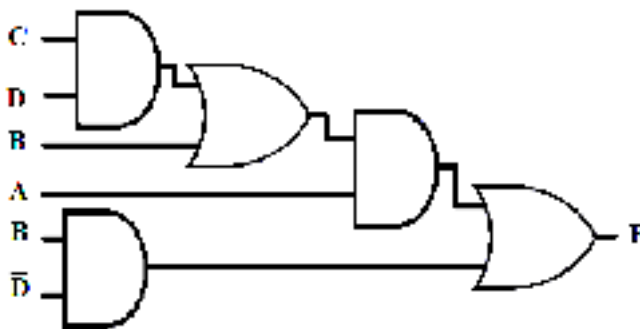
MULTI-LEVEL Implementation using NAND Gate

- Schematic having more than two levels of gates is known as a multi-level schematic.
- We can implement multi-level SOP expression using NAND gate.
- The conversion of multi-level expression into NAND gate has the same method as two-level implementation.
- The multi-level expression can be converted into two-level expression but for the sake of realization, we will implement a multi-level expression.

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

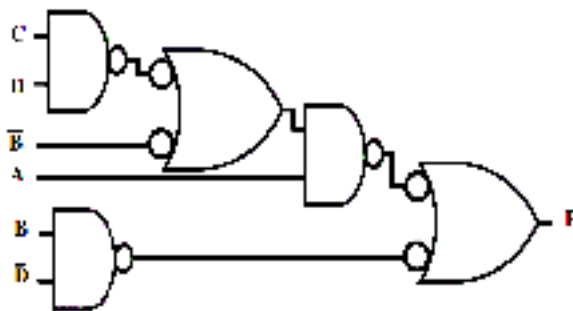
This is a four-level function

First, we will draw its AND-OR schematic

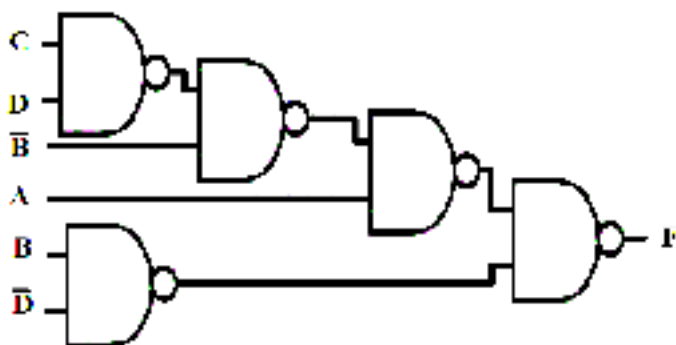


MULTI-LEVEL NAND Gate Implementation

□ Notice the AND-OR pattern. So it can be easily converted into NAND gates. Now we will convert this into mixed notation i.e. AND gate will be converted into AND-INVERT and OR will be converted into INVERT-OR.



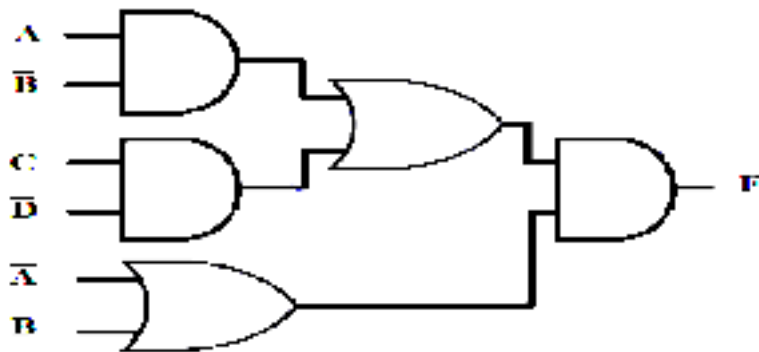
- Remember double bubbles along a single line cancel each other, and a single bubble along a line should be compensated by inserting an inverter in that line.
- Notice the 3rd line of input B, there is a single bubble. To compensate this bubble, either an inverter should be added or the input B should be complimented.
- Then redraw the whole schematic using all NAND gates by replacing AND-INVERT and INVERT-OR with NAND gates .



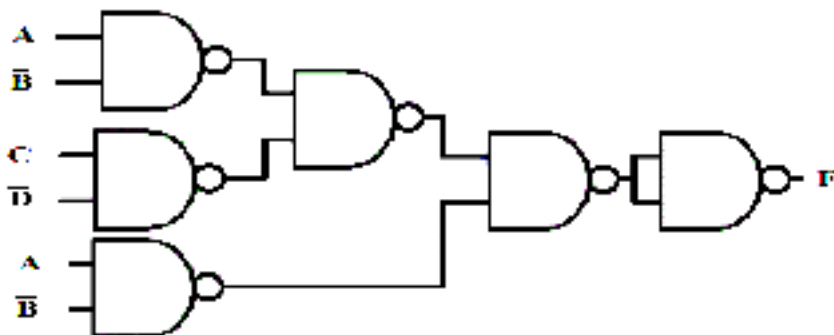
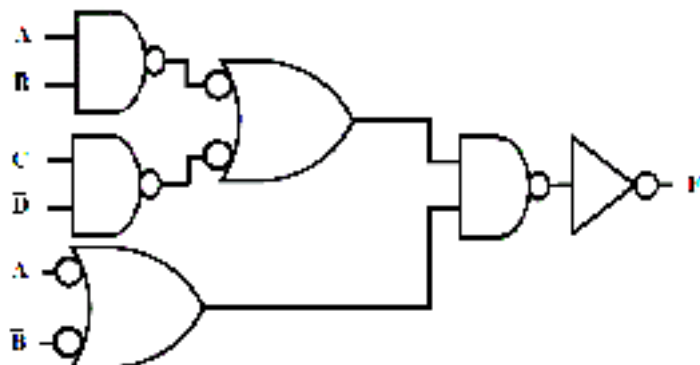
Three-level Implementation & Example using NAND

Three-level Implementation & Example using NAND Gate

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



Three level implementation



NOR – NOR realization

Step 1: Write the expression in POS form.

Step 2: For each sum term draw one NOR gate at the first level.

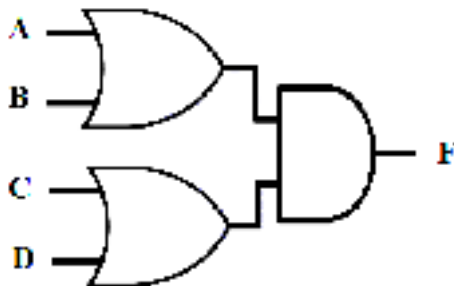
Step 3: Draw a single NOR gate at the 2nd level, with inputs coming from outputs of first level gates.

Step 4: A term with a single literal requires an inverter in the first level.
If single literal is complemented, it can be connected directly to an input of the II level NOR gate.

EXAMPLE 1

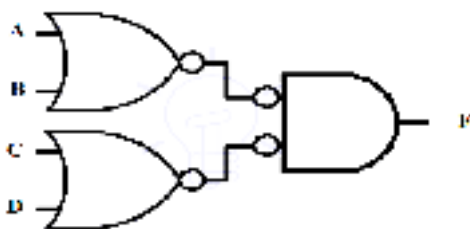
$$F = (A + B)(C + D)$$

Draw its schematic using AND-OR NOT gates as shown in the figure given below.



Mixed Notation

Next step is to draw the above-mentioned schematic using OR-Invert and Invert-AND gates. OR-Invert should replace OR gates and invert-AND replaces AND gates. This schematic is said to be in mixed notation and its schematic is given below.

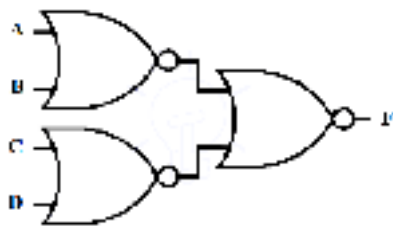


A bubble means complement. Two bubbles along a line mean double complementation and they cancel each other. However, a single bubble

along a line should be compensated by inserting an Inverter in that line or if it is an input line then you can also feed a complemented input if available.

NOR Gate Conversion

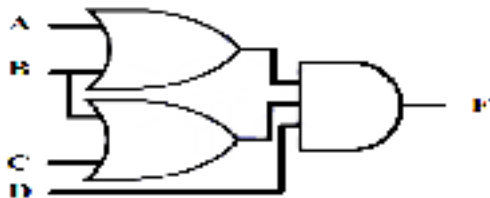
The last step is to redraw the whole schematic replacing OR-Invert and Invert-AND gate symbol by NOR gate symbol because OR-Invert and Invert-AND are equivalent to NOR gate. The final schematic is shown in the figure given below.



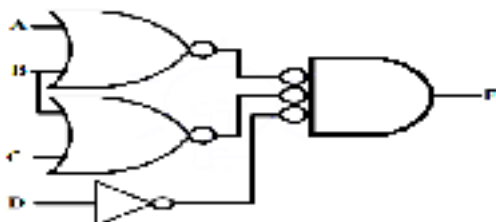
EXAMPLE 2

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

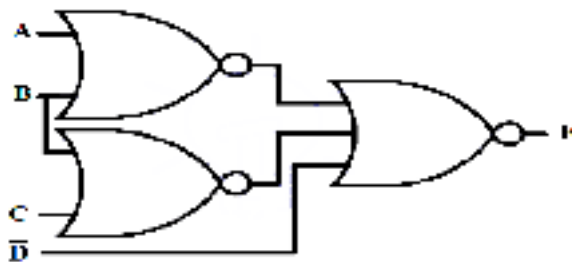
1. This function is in simplified Product of Sum form. First, we need to draw its OR-AND schematic.



2. Now we convert the above-given schematic into mixed notation by converting OR gate into OR-INVERT and AND gate into INVERT-AND.



- Now replace every OR-Invert and Invert-AND with NOR gate as shown in the figure given below.

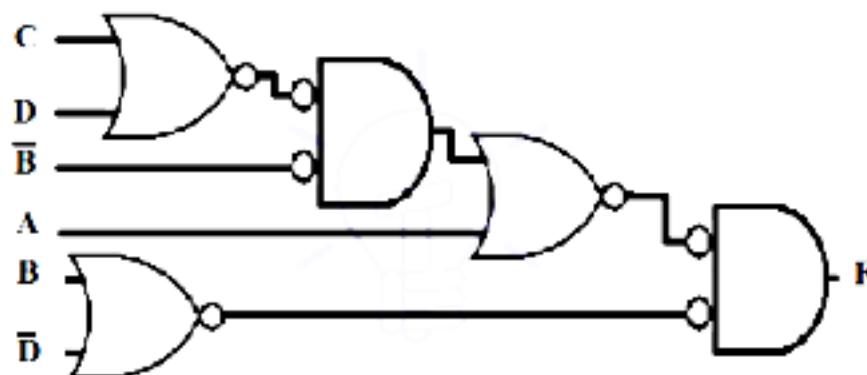
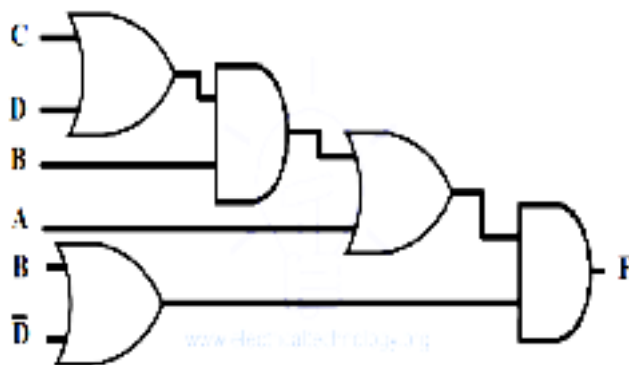


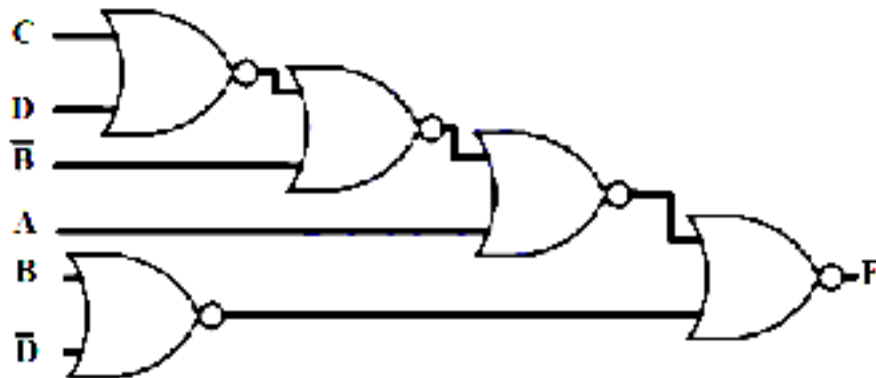
Multi-level Implementation using NOR Gate

Schematic having more than two levels of gates is known as a multi-level schematic.

EXAMPLE 1

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





EXAMPLE 2 (Three Level)

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

