- Digital Logic Gates :-

- Boolean functions are expressed in terms of AND, OR and NOT operations, it is easier to implement a Boolean function with these types of gates.
- Factors to be weighed in considering the construction of other types of logic gates are ~~(4)~~
  (1) the feasibility and economy of producing the gate with physical components,
  (2) the possibility of extending the gate to more than two inputs,
  (3) the basic properties of the binary operator, such as commu-tativity and and associativity, and
  (4) the ability of the gate to implement Boolean functions alon or in conjuction with other gates.
- ~~(5)~~ AND, OR, NAND, NOR, EX-OR, equivalence, complement and transfer are used as standard gates.

→ There are three types of logic gates
  1) Basic Gates
  2) Universal Gates
  3) Special Gates.

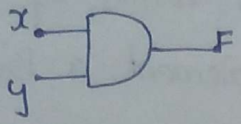
1) Basic Gates
   - AND Gate
   - OR Gate
   - NOT Gate.

2) Universal Gates
   - NAND Gate
   - NOR Gate

3) Special Gates
   - EX-OR Gate
   - EX-NOR Gate.

NAME   Graphic Symbol   Algebraic function   Truth Table

AND

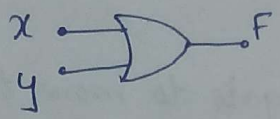


$F = x \cdot y$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

Def:- The output is high when all the inputs are high (or)
The output is low when atleast one input is low.

OR

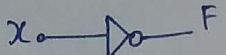


$F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Def:- The output is low when all inputs are low (or)
The output is high when atleast one input is high.

NOT (or) Inverter

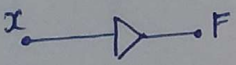


$F = x'$

x	F
0	1
1	0

Def:- The output is complement of input.
The inverter circuit inverts the logic sense of a binary variable, producing the NOT or complement function. The small circle in the output of the graphic symbol of an inverter (referred to as bubble) designates the logic complement.

Buffer



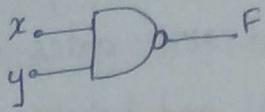
$F = x$

x	F
0	0
1	1

Def:- The triangle symbol by itself designates a buffer circuit. A buffer produces the transfer function, but does not produce a logic operations, since the binary value of the output is equal to the binary value of the input.

Name | Graphic Symbol | Algebraic function | Truth table

NAND

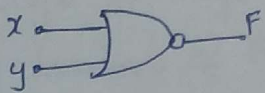


$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

Def:- The NAND function is the complement of the AND function as indicated by a graphic symbol that consists of an AND graphic symbol followed by a small circle.
If all the inputs are high, then output is low (or)
If atleast one input is low, then output is high.

NOR

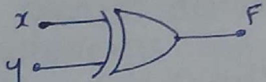


$F = (x+y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

Def:- The NOR function is the complement of the OR function and uses an OR graphic symbol followed by a small circle.
If all the ~~output~~ inputs are low, then the output is high (or)
If atleast one input is high, output is low.

Exclusive-OR (XOR)

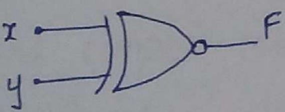


$F = xy' + x'y$
$= x \oplus y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Def: The output is low, when all the inputs are same (or)
The output is high for odd number of ones.
The exclusive-OR gate has a graphic symbol to that of the OR gate, except for the additional curved line on the input side.

Exclusive-NOR (or) Equivalence



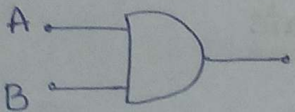
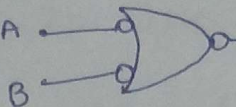
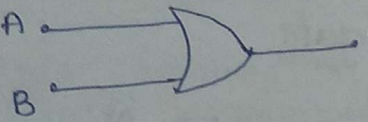
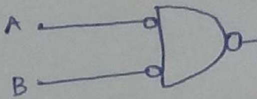
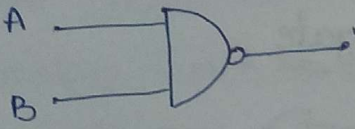
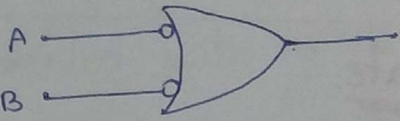
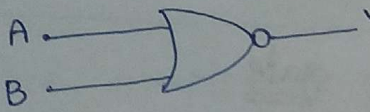
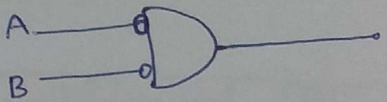
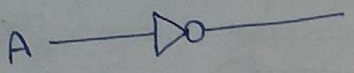
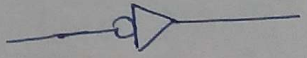
$F = xy + x'y'$
$= (x \oplus y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Def:- The equivalence, or exclusive-NOR gate is the complement of the exclusive-OR, as indicated by the small circle on the output side of the graphic symbol.
The output is high only when all the inputs are same (or) The output is high for even number of ones in the inputs.

NOTE :- NAND and NOR gates are used extensively as standard logic gates and are in fact far more popular than the AND and OR gates. This is because NAND and NOR gates are easily constructed with transistor circuits and because digital circuits can be easily implemented with them.

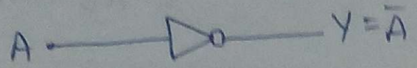
Alternate Logic Gate Representation :-

1.  $Y = AB$	 $Y = \overline{\bar{A} + \bar{B}}$ $= \overline{\overline{AB}} = AB$.
2.  $Y = A+B$	 $Y = \overline{\bar{A}\,\bar{B}}$ $= \overline{\overline{A+B}} = A+B$.
3.  $Y = \overline{AB}$	 $Y = \bar{A} + \bar{B}$ $= \overline{AB}$ – De Morgan
4.  $Y = \overline{A+B}$	 $Y = \bar{A} \cdot \bar{B}$ $= \overline{A+B}$ – De Morgan
5.  $Y = \bar{A}$	 $Y = \bar{A}$

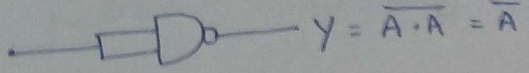
* ~~AND gate~~ is ~~bubble~~.
* AND gate is input bubbled NOR gate.
* OR gate is input bubbled NAND gate.
* NAND gate is input bubbled OR gate.
* NOR gate is input bubbled AND gate.
* NOT gate is input bubbled NOT gate.

Representation of all gates using NAND gate :

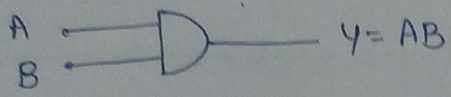
NOT gate



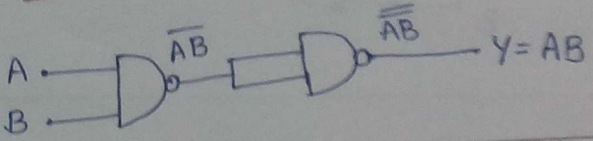
NAND gate



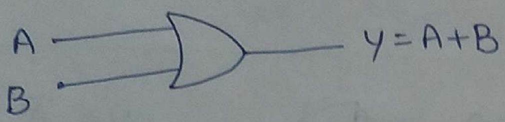
AND gate



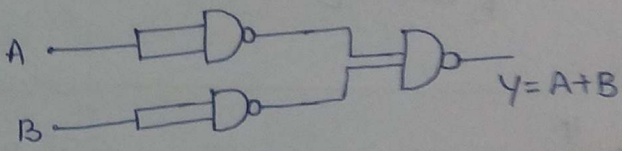
NAND gate



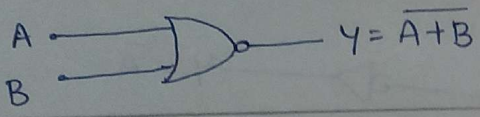
OR gate



NAND gate



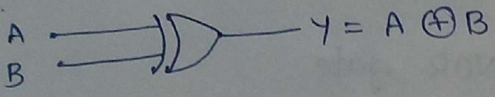
NOR gate

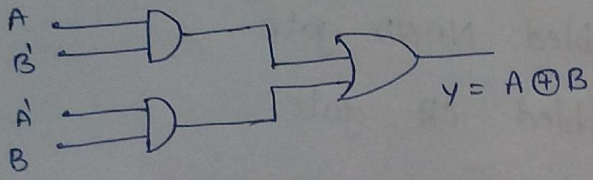


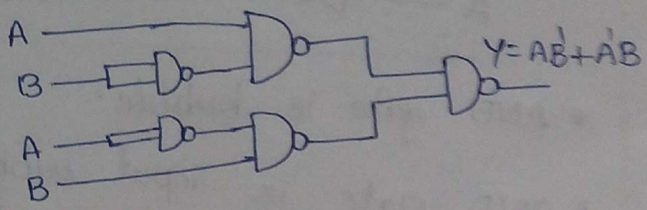
NAND gate

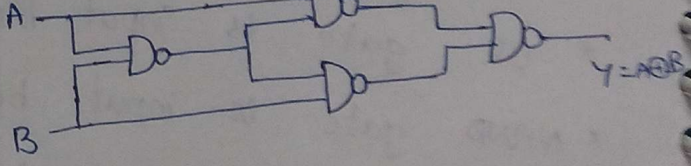
NOR gate cannot be represented in terms of NAND gates.

Ex-OR gate

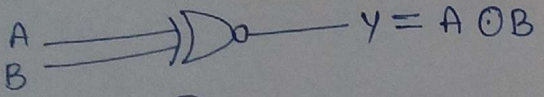


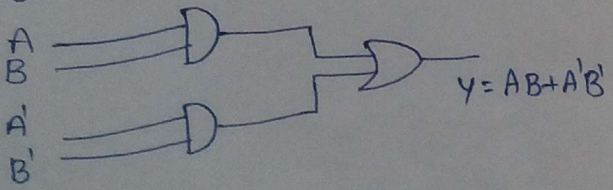


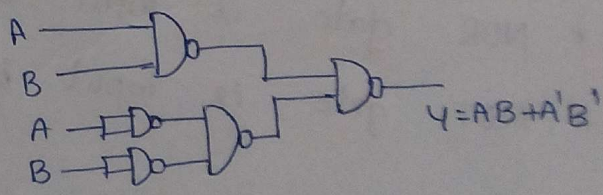


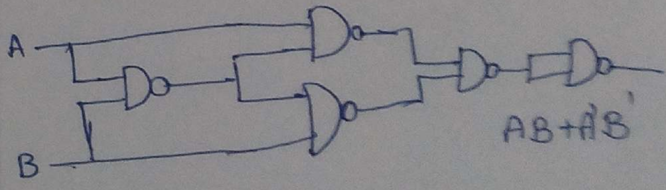


Ex- NOR gate









→ **NAND AND NOR Implementation:-**

- ~~The~~ ~~A~~ Digital circuits are frequently constructed with NAND ~~or~~ NOR gates rather than with AND and OR gates.
- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.

→ **NAND Implementation:-**

- The NAND gate is said to be a "universal gate" because any logic circuit can be implemented with it. To show that any Boolean function can be implemented with NAND gates, the logical operations of AND, OR, and complement can be obtained with NAND gates alone. (This can shown in previous page).

1. The complement operation is obtained from a one-input NAND gate that behaves exactly like an inverter.
2. The AND operation requires two NAND gates. The first produces the NAND operation and the second inverts the logical sense of the signal.
3. The OR operation is achieved through a NAND gate with additional inverters in each input.

* A convenient way to implement a Boolean function with NAND gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND logic.

- The conversion of an algebraic expression from AND, OR, and complement to NAND can be done by simple circuit manipulation techniques that change AND-OR diagrams to NAND diagrams.

- To facilitate the conversion to NAND logic, it is convenient to define an alternative graphic symbol for the gate.
- Two equivalent graphic symbols for the NAND gate are shown in below

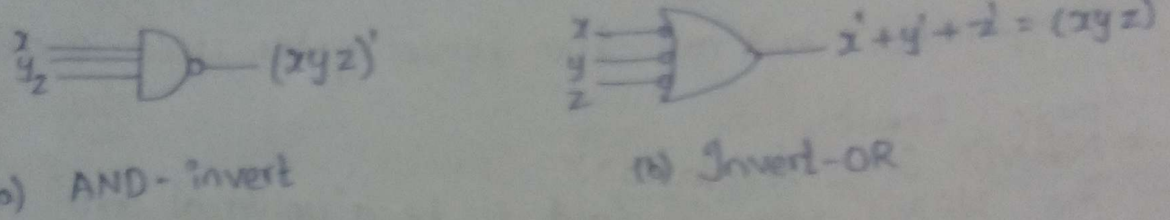


fig: Two graphic symbols for a three-input NAND gate

- The AND-invert symbol consists of an AND graphic symbol followed by a small circle negation indicator referred to as a bubble.

* - Alternatively, it is possible to represent a NAND gate by an OR graphic symbol that is preceded by a bubble in each input.
- The invert-OR symbol ~~of~~ for the NAND gate follows DeMorgan's theorem and the convention that the negation indicator (bubble) denotes complementation.
- The two graphic symbols representations are useful in the analysis and design of NAND circuits
- When both symbols are mixed in the same diagram, the circuit is said to be in mixed notation.

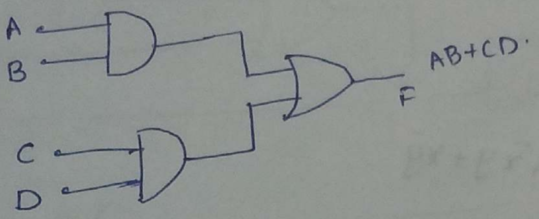
→ Two-level implementation:-

* The implementation of Boolean functions with NAND gates requires that the functions be in sum-of-products form.

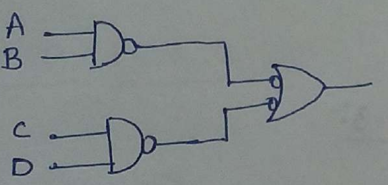
NAND Implementation Procedure :-

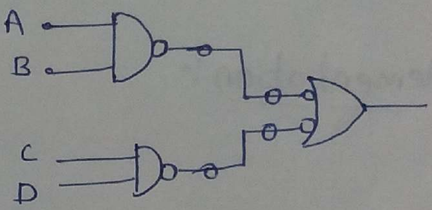
1. Add bubble to input of OR gate
2. Add bubble to output of AND gate
3. Add bubble where bubble is added.
4. Double inversion is eliminated.
5. Replace the input bubbled OR gate by NAND gate.

Example :- F = AB+CD (Two-level Implementation).

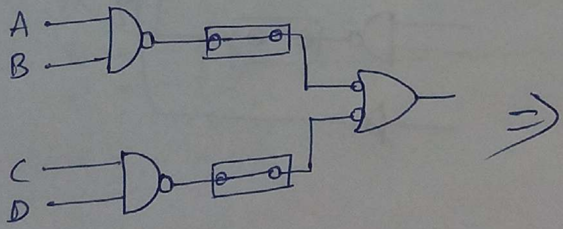


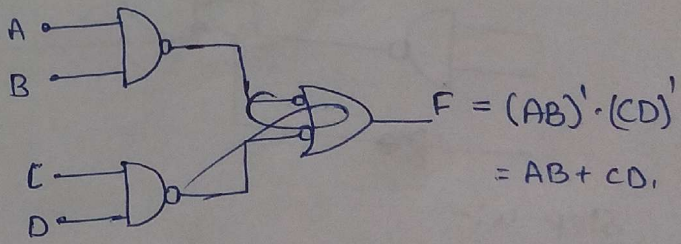
NAND Implementation (step 1 & 2) step 3 :-





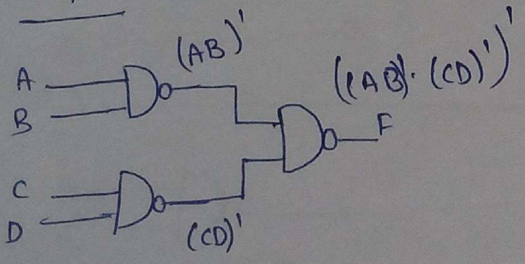
step 4 :-





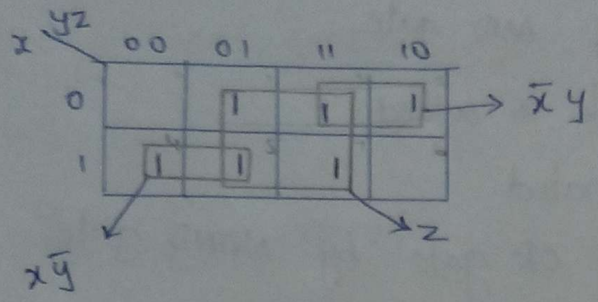
$F = (AB)' \cdot (CD)'$
$= AB + CD.$

step 5 :-



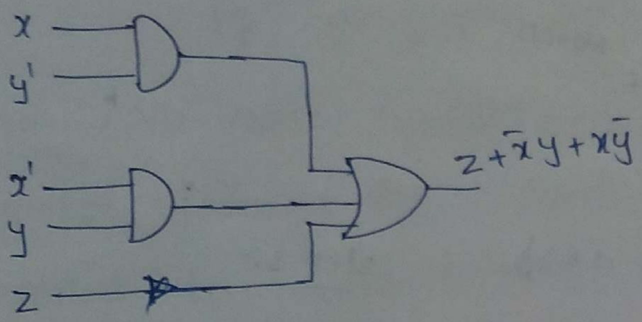
→ Implement the following Boolean function with NAND gates:

$F(x,y,z) = (1,2,3,4,5,7)$



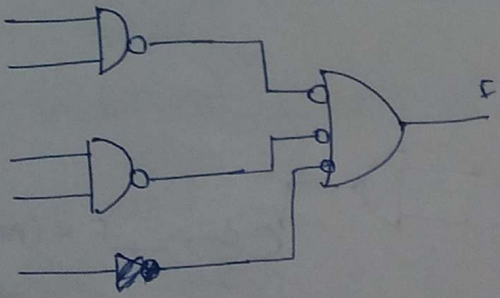
$F = z + \bar{x}y + x\bar{y}$

→

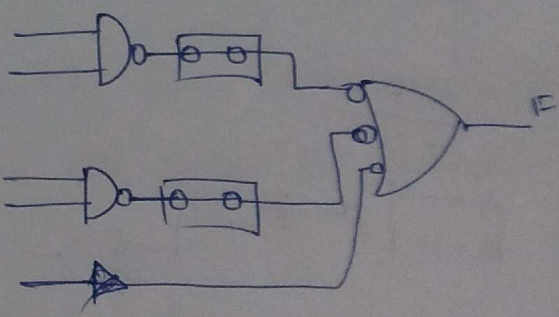


NAND Implementation:-

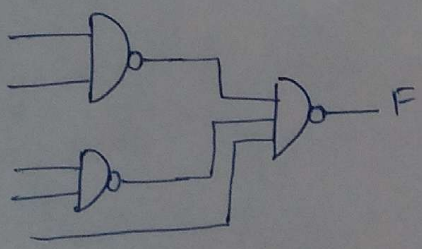
step 1 & 2



step 3:-



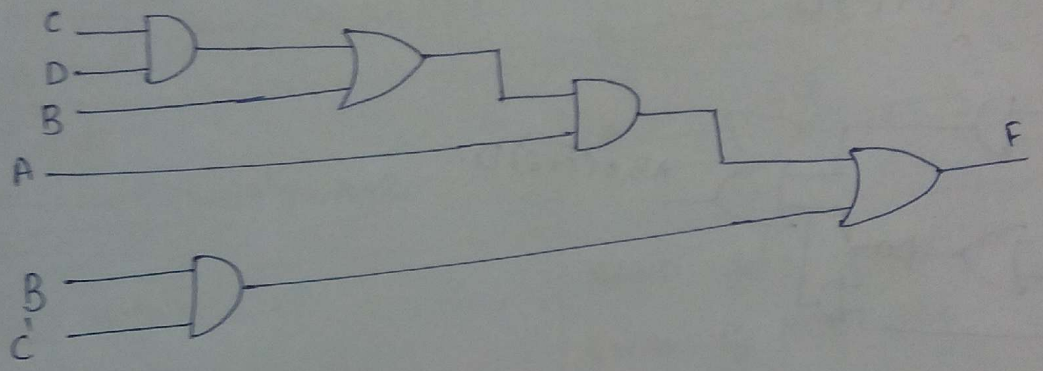
step 4:-



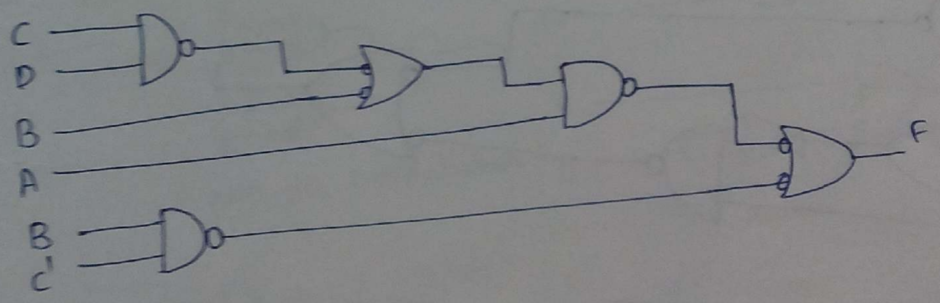
Multilevel NAND Implementation:-

- The most common procedure in terms of AND, OR and complement operations. The function can then be implemented with AND and OR gates. After that, if necessary, it can be converted into an all-NAND circuit.

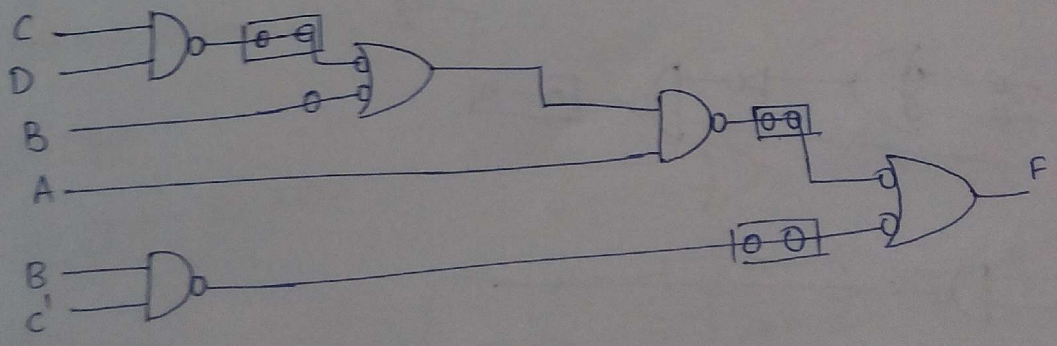
Example:- $F = A(CD+B) + BC'$



step 1 & 2

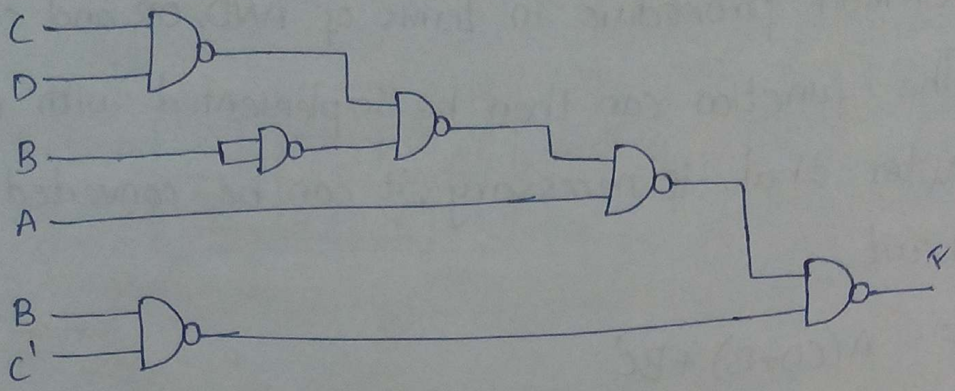


step 3:-

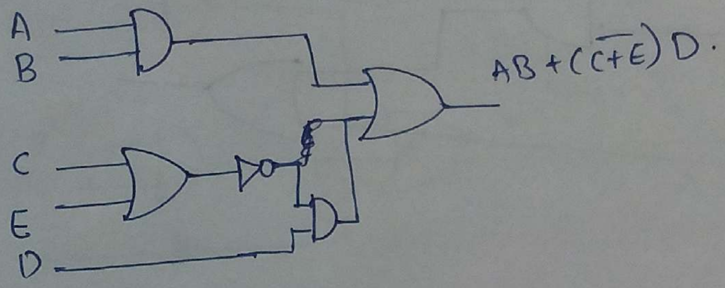


~~step~~

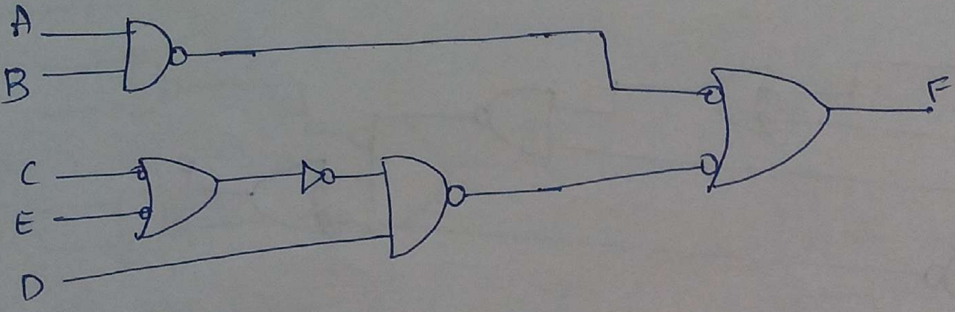
Step 4:

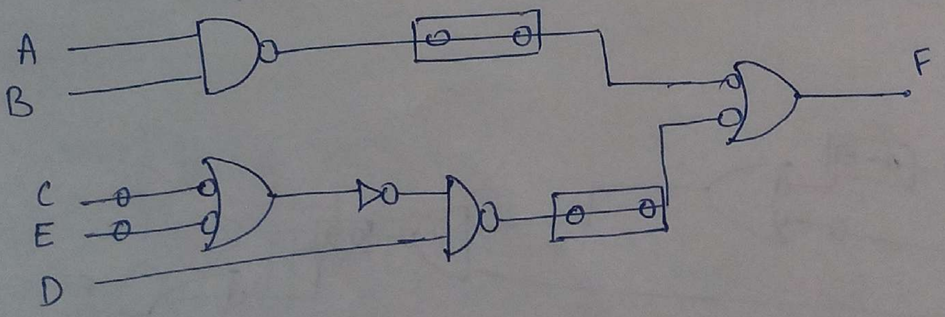


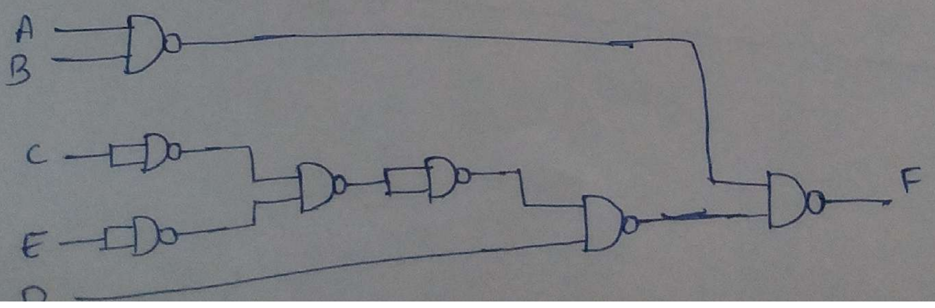
→ $Y = AB + (\overline{C+E})D.$



NAND Implementation

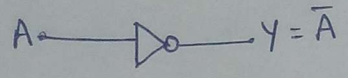


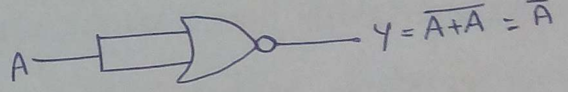




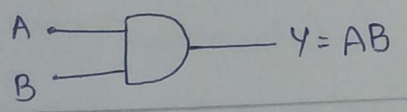
Representation of all gates using NOR gate :-

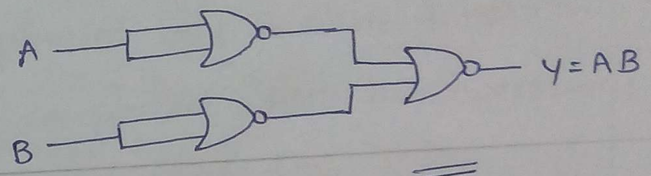
NOT gate :-



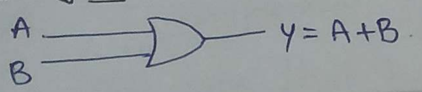


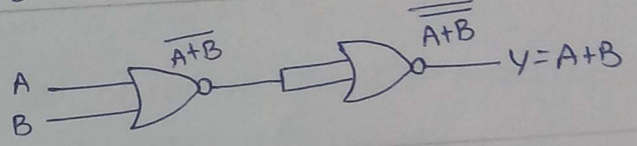
AND gate :-



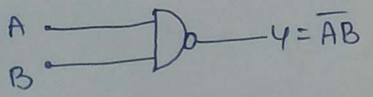


OR gate :-



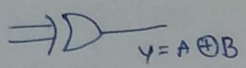


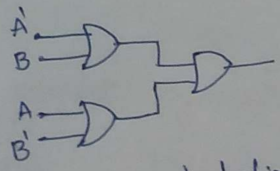
NAND gate :-



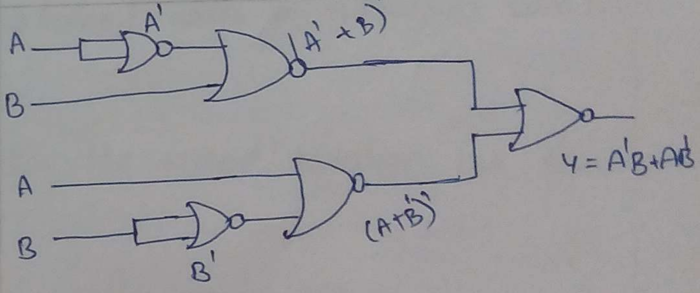
NAND gate cannot be represented in terms of NOR gate.

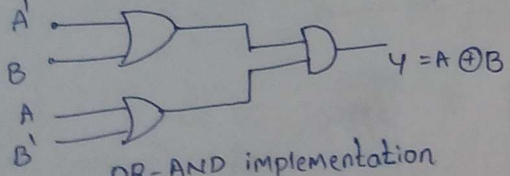
EX-OR gate :-





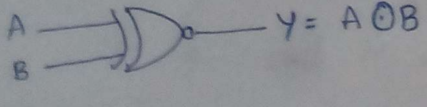
AND-OR implentation

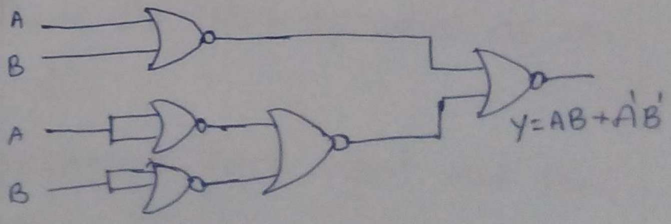


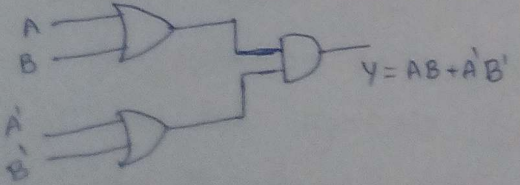


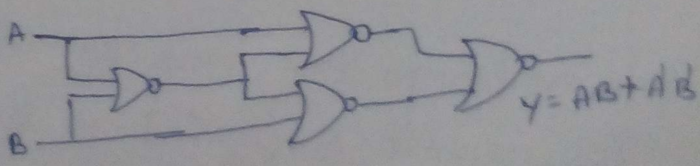
OR-AND implementation

EX-NOR gate :-







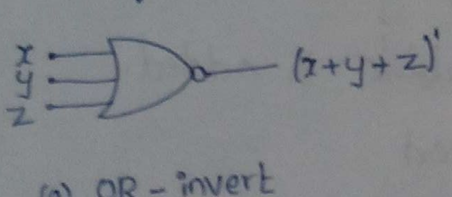


NOR Implementation :-

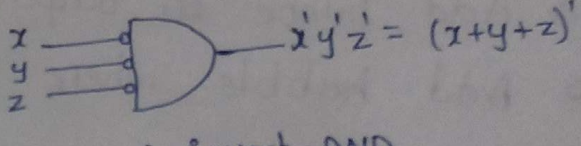
- The NOR operation is the dual of the NAND operation.
- Therefore, all procedures and rules for NOR logic are the duals of the corresponding procedures and rules developed for NAND logic. The NOR gate is another universal gate that can be used to implement any Boolean function.
- The implementation of the complement, AND, OR operations with NOR gates is shown in above fig.

1. The complement operation is obtained from a one-input NOR gate that behaves exactly like an inverter.
2. The AND operation is obtained with a NOR gate that has inverters in each input.
3. The OR operation is obtained ~~with a~~ requires two NOR gates.

* The two graphic symbols for the mixed notation are shown in below figure.



(a) OR - invert



(b) invert - AND

fig:- Two graphic symbols for the NOR gate.

- (a) The OR-invert symbol defines the NOR operation as an OR followed by a complement.
- (b) The invert-AND symbol complements each input and then performs an AND operation.
- The two symbols designate the same NOR operation and are logically identical because of DeMorgan's theorem.

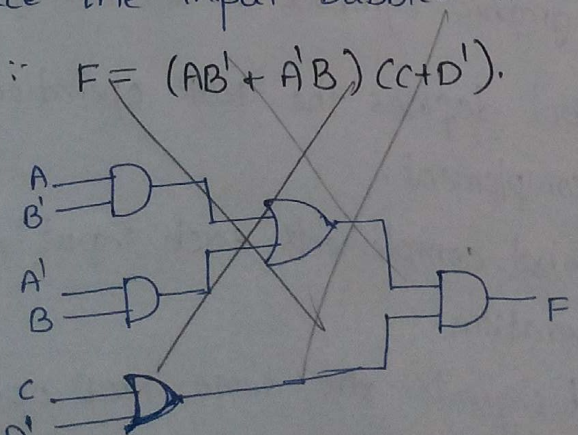
→ Two-level implementation

- A two-level implementation with NOR gates requires that function be simplified into product-of-sums form.
- Product-of-Sums expression is obtained from the map by combining the 0's and complementing.
- A product-of-sums expression is implemented with a first level of OR gates that produce the sum terms followed by a second-level AND gate to produce the product.
- The transformation from the OR-AND diagram to a NOR diagram is acheived by changing the OR gates to NOR gates with OR-invert graphic symbol and the AND gate to a NOR gate with an invert-AND graphic symbol. A single literal term going into the second-level gate must be complemented.

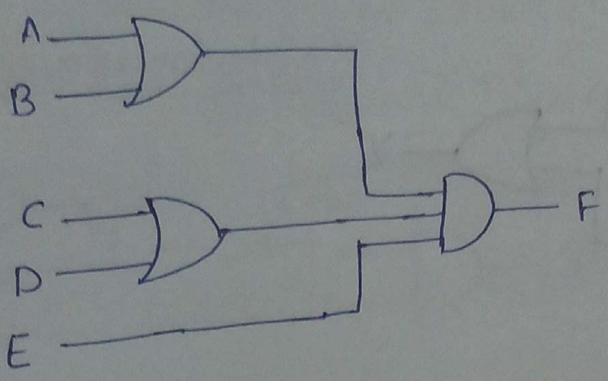
→ NOR Implementation Procedure:

1. Add bubble to input of AND gate.
2. Add bubble to output of OR gate
3. Add bubble where the bubble is added.
4. Double inversion is eliminated
5. Replace the input bubbled AND gate by NOR gate.

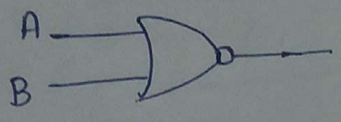
Example: $F = (AB' + A'B)(C+D')$.

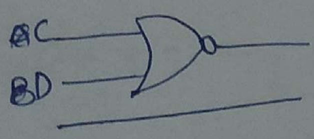


Example: $F = (A+B)(C+D)E$.

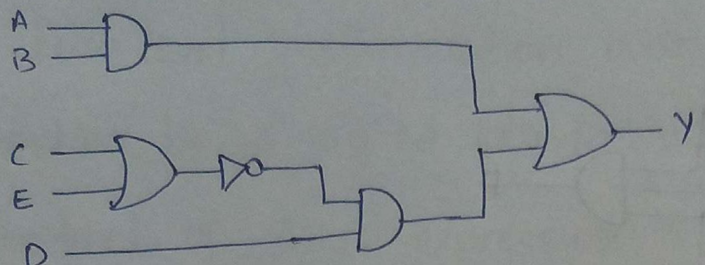


Step 1 & 2:-

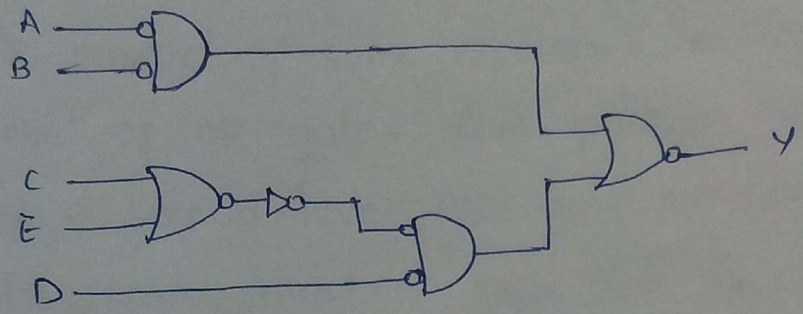




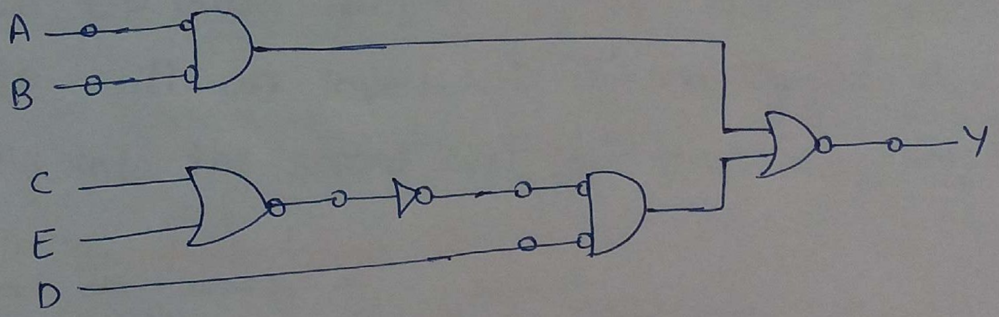
- $Y = AB + (\overline{C+E})D.$



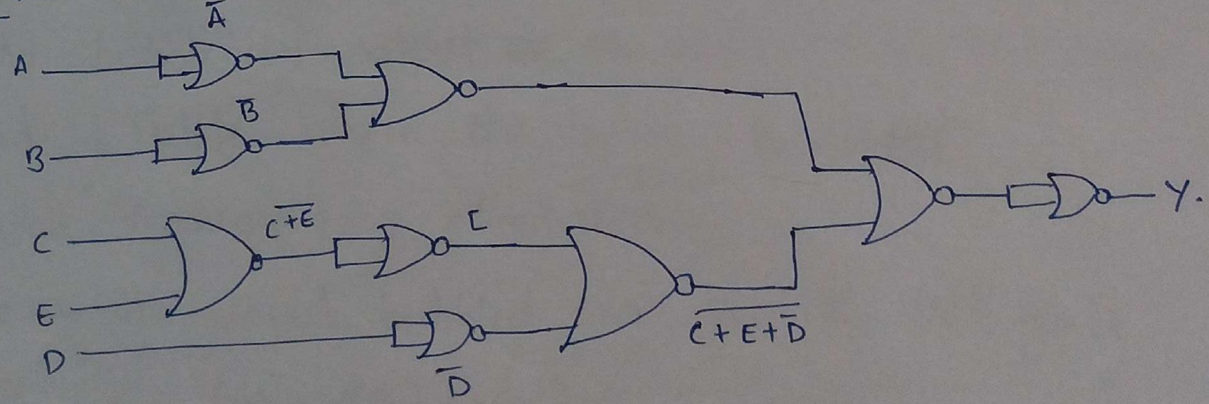
step 1 & 2 :-



step 3 :-

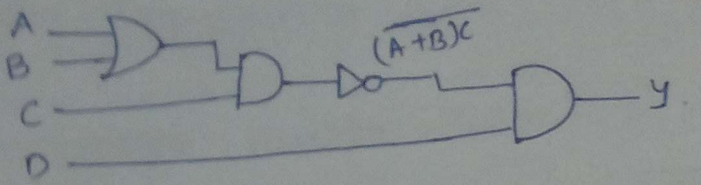


step 4 & 5

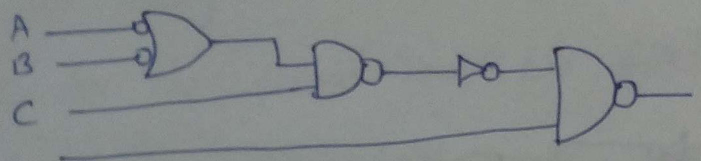


A $Y = [\overline{(A+B)C}]D.$

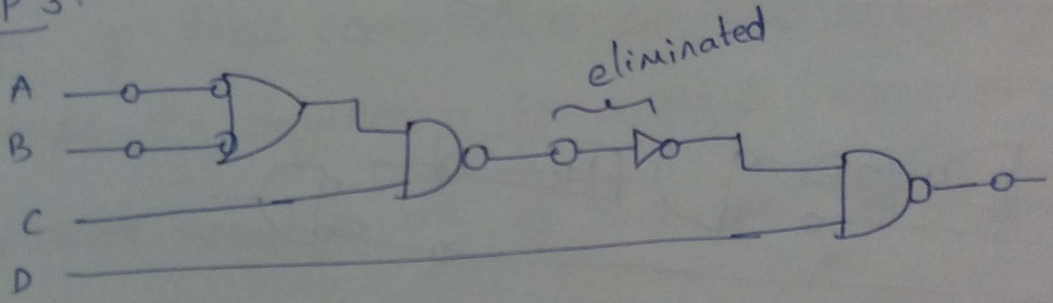
NAND Implementation:-



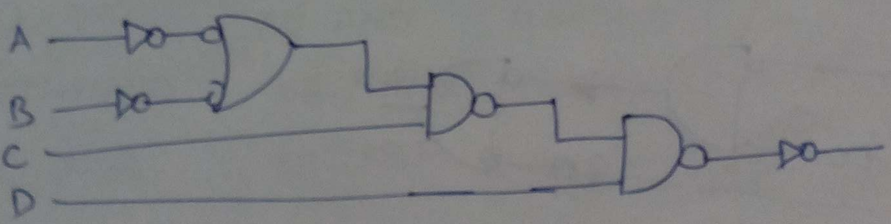
step 1 & 2



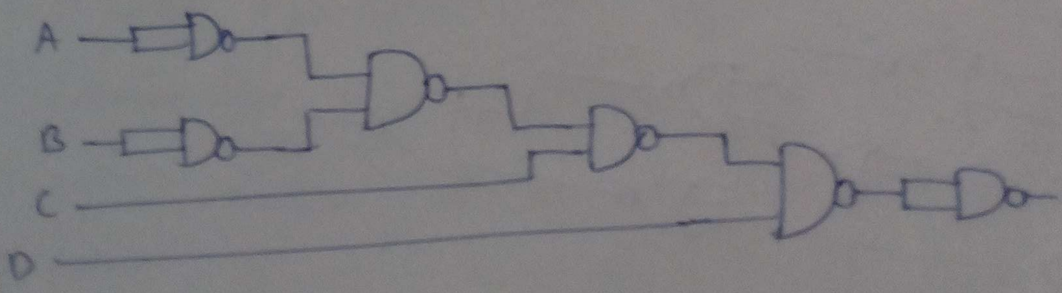
step 3:-



step 4:-



step 5:-

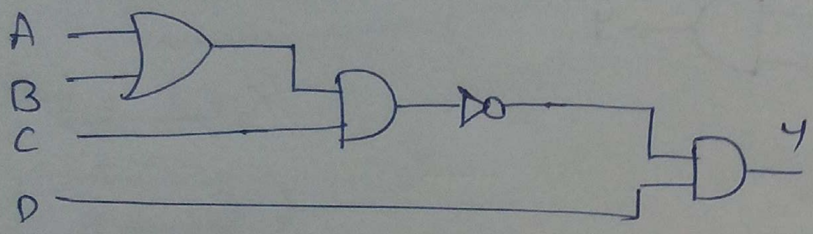


Scanned by CamScanner

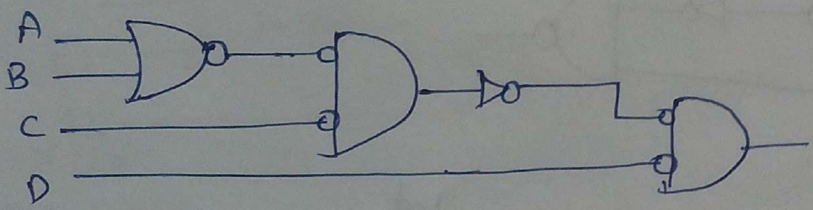
NOR Implementation

$Y = \left[\overline{(A+B)C} \right] D.$

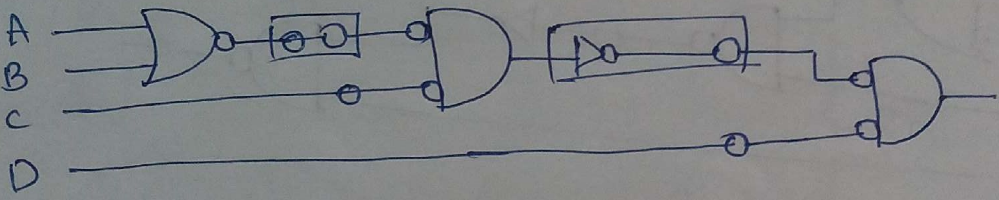
step 1:-



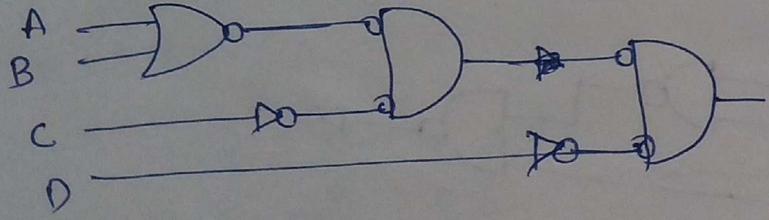
step 1 & 2:-



step 2:-



step 4:-



step 5:-