Basic Logic Operations:

- AND $*$
- OR $+$
- NOT (Complement) $-$

- Order of Precedence {BODMAS}
  1. NOT
  2. AND
  3. OR

  – can be modified using parenthesis

$(\quad) + (\underbrace{\quad}_{+})$

Basic Logic Operations:

Table 1.8
Truth Tables of Logical Operations

$n=2$

$2^n = 4$

AND

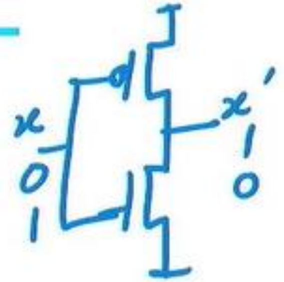
x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

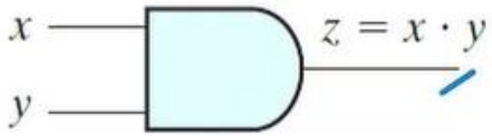
NOT

x	x'
0	1
1	0

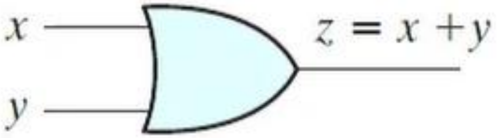


NPTEL

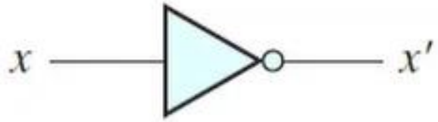
Basic Logic Operations:



(a) Two-input AND gate



(b) Two-input OR gate

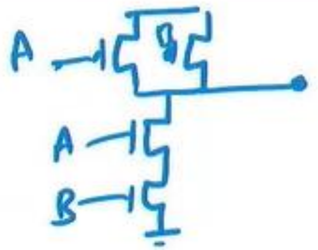


(c) NOT gate or inverter

NPTEL

Additional Logic Operations:

- NAND
  - F = (A . B)'



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

- NOR
  - F = (A + B)'

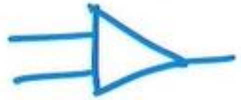
$\overline{(A+B)}$

x	y	o/p
0	0	0
0	1	1
1	0	1
1	1	0

both same

both different.

- XOR
  - Output is 1 iff either input is 1, but not both.

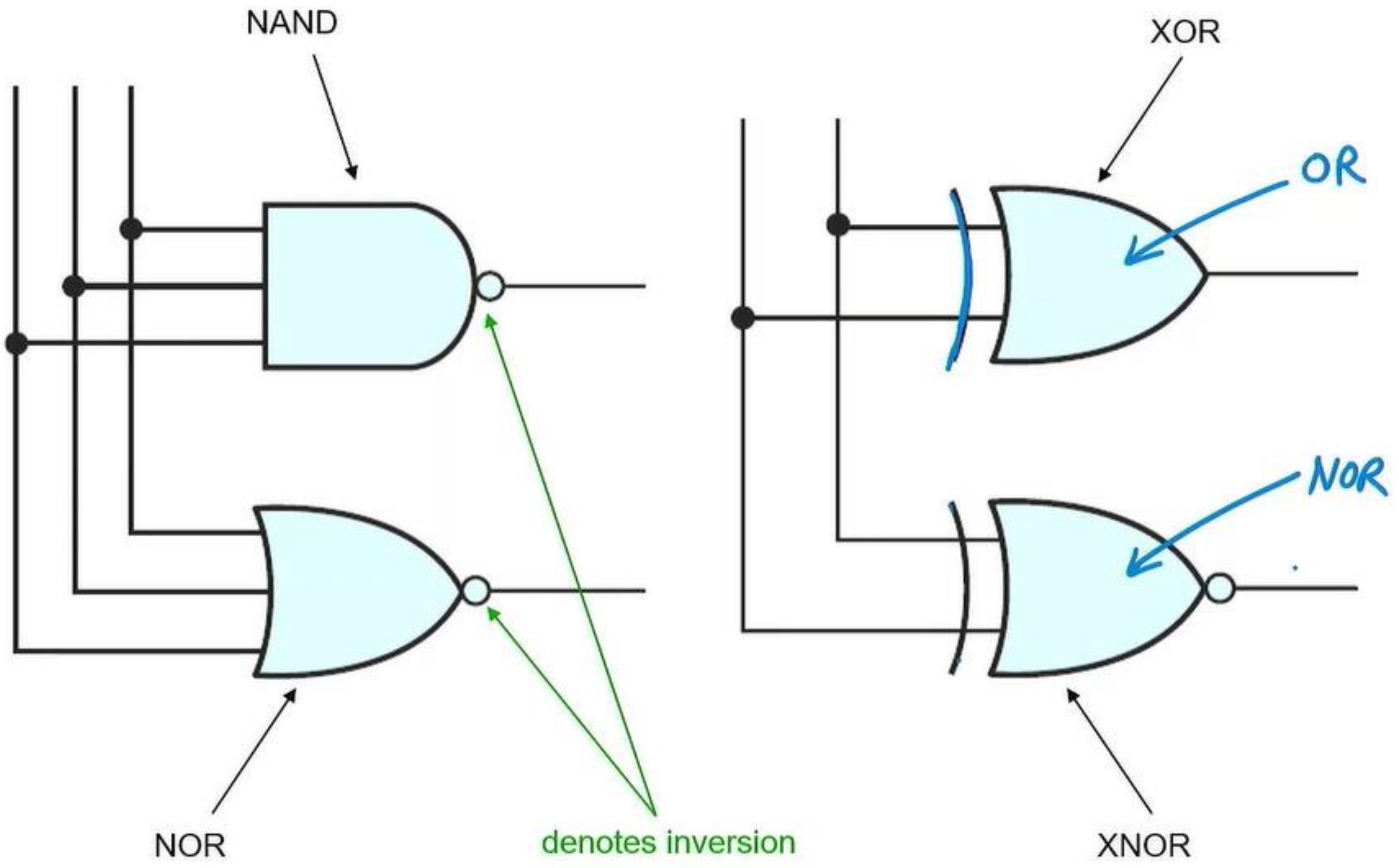


- XNOR (Equivalence)
  - Output is 1 iff both inputs are 1 or both inputs are 0.

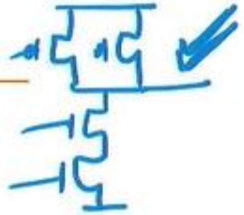
x	y	o/p
0	0	1
0	1	0
1	0	0
1	1	1

NPTEL

Additional Logic Operations:



Truth Tables:



- Used to describe the functional behavior of a Boolean expression and/or Logic circuit.
- Each row in the truth table represents a unique combination of the input variables.
  - For n input variables, there are 2^n rows.
- The output of the logic function is defined for each row.
- Each row is assigned a numerical value, with the rows listed in ascending order.
- The order of the input variables defined in the logic function is important.

	x	y	o/p
1	0	0	1
2	0	1	1
3	1	0	1
4	1	1	[illegible]

2^n unique combination

Truth Tables:

3-input Truth Table $2^3 = 8$ unique combinations.

$A \cdot (B+C)$

F(A,B,C) = Boolean expression

4-input Truth Table $2^4 = 16$ unique combinations.

$A + B(C+D)$

F(A,B,C,D) = Boolean expression

NPTEL

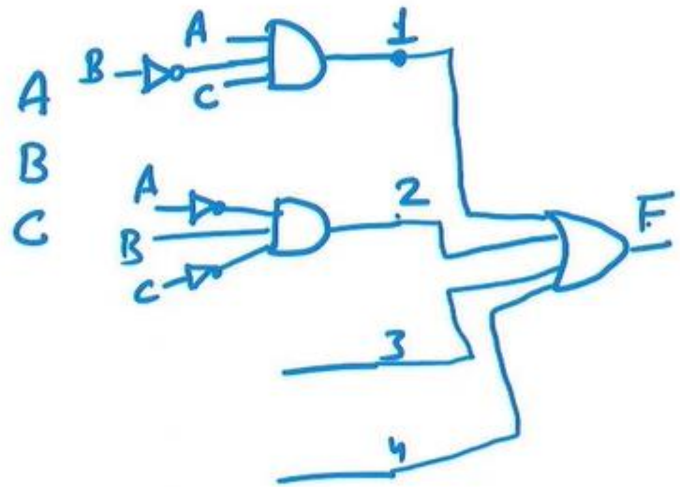
Boolean Expressions:

- Boolean expressions are composed of
  1. Literals – variables and their complements $A, B, \bar{B}$
  2. Logical operations $(A+B)$ $(A \cdot B)$
- Examples
  - F = A.B'.C + A'.B.C' + A.B.C + A'.B'.C' (Literals = A, B, C)

    1 2 3 4

    literals

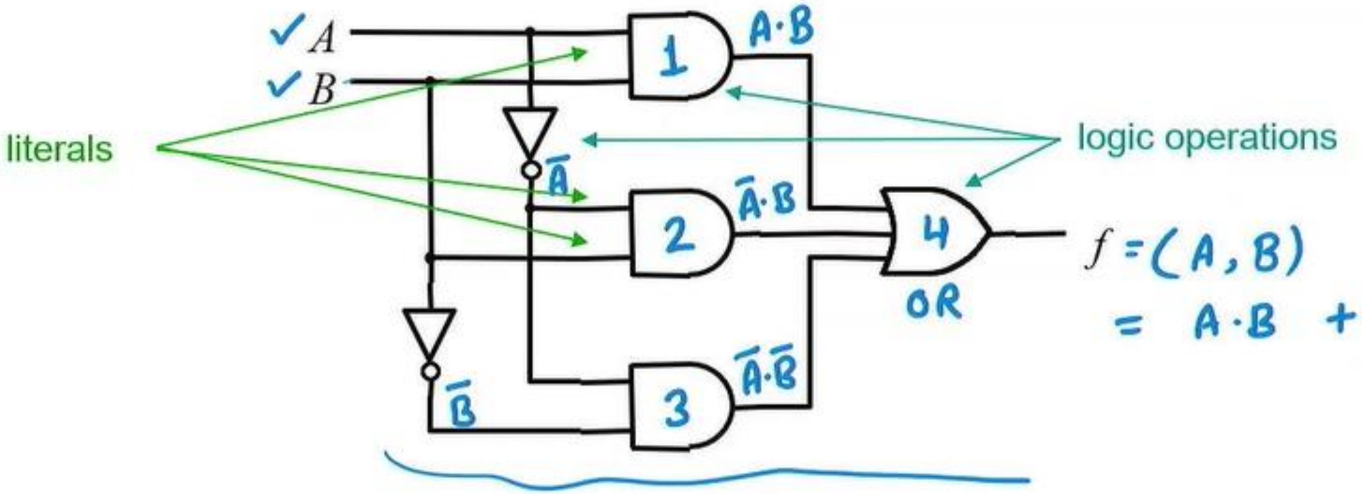
    logic operations
  - F = (A+B+C').(A'+B'+C).(A+B+C)
  - F = A.B'.C' + A.(B.C' + B'.C)



Boolean Expressions:

- Boolean expressions are realized using a network (or combination) of logic gates.
  - Each logic gate implements one of the logic operations in the Boolean expression
  - Each input to a logic gate represents one of the literals in the Boolean expression

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



$f = (A, B)$
$= A \cdot B + \bar{A} \cdot B + \bar{A} \cdot \bar{B}$

Boolean Expressions:

- Boolean expressions are evaluated by
  - Substituting a 0 or 1 for each literal
  - Calculating the logical value of the expression
- A Truth Table specifies the value of the Boolean expression for every combination of the variables in the Boolean expression.
- For an n-variable Boolean expression, the truth table has 2^n rows (one for each combination).

$F = A \cdot B + A\bar{B} + \bar{A}\bar{B}$

A	B	O/P
0	0	( )
0	1	( )
1	0	( )
1	1	( )

4 input → 16 Row
5 i/p → 32 Row
6 i/p → 64 Row
7 i/p → 128 Row

2^n

NPTEL

Boolean Expressions:

- Two Boolean expressions are equivalent if they have the same value for each combination of the variables in the Boolean expression.
  - $F_1 = (A + B)'$ $= \overline{(A+B)}$
  - $F_2 = A'.B'$ $= \overline{A \cdot B}$
- How do you prove that two Boolean expressions are equivalent?
  - Truth table
  - Boolean Algebra

Tr 1 Trial 2

i/p ✓ ✓ ✓ ✓

o/p ✓ ✓ ✓ ✓

NPTEL

Boolean Algebra:

- George Boole developed an algebraic description for processes involving logical thought and reasoning.
  - Became known as Boolean Algebra
- Claude Shannon later demonstrated that Boolean Algebra could be used to describe switching circuits.
  - Switching circuits are circuits built from devices that switch between two states (e.g. 0 and 1).
  - Switching Algebra is a special case of Boolean Algebra in which all variables take on just two distinct values.
- Boolean Algebra is a powerful tool for analyzing and designing logic circuits.

NPTEL

Boolean Expressions:

- Example:

- Using a Truth table, prove that the following two Boolean expressions are equivalent.

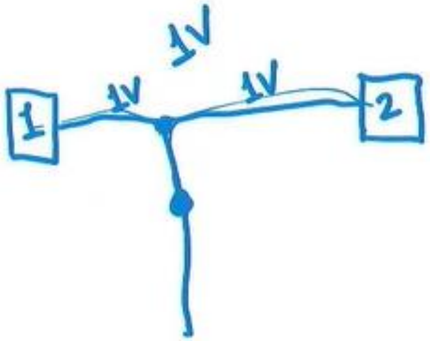
- $F_1 = (A + B)'$
- $F_2 = A'.B'$

Basic Laws and Theorems:

Commutative Law	$A + B = B + A$	$A.B = B.A$
Associative Law	$A + (B + C) = (A + B) + C$	$A . (B . C) = (A . B) . C$
Distributive Law	$A.(B + C) = AB + AC$	$A + (B . C) = (A + B) . (A + C)$
Null Elements	$A + 1 = 1$	$A . 0 = 0$
Identity	$A + 0 = A$	$A . 1 = A$
Idempotence	$A + A = A$	$A . A = A$
Complement	$A + A' = 1$	$A . A' = 0$
Involution	$A'' = A$	
Absorption (Covering)	$A + AB = A$	$A . (A + B) = A$
Simplification	$A + A'B = A + B$	$A . (A' + B) = A . B$
DeMorgan's Rule	$(A + B)' = A'.B'$	$(A . B)' = A' + B'$
Logic Adjacency (Combining)	$AB + AB' = A$	$(A + B) . (A + B') = A$
Consensus	$AB + BC + A'C = AB + A'C$	$(A + B) . (B + C) . (A' + C) = (A + B) . (A' + C)$

NPTEL

Idempotence:



$1 + 1 = 2$

$\boxed{1 + 1 = 1}$

- $A + A = A$
- $F = ABC + ABC' + ABC$
- $F = ABC + ABC'$

- $A \cdot A = A$

$1 \cdot 1 = 1$

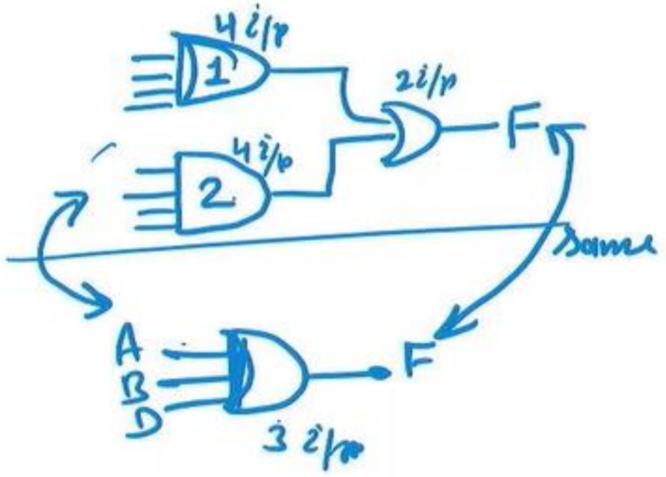
- $G = (A' + B + C').(A + B' + C).(A + B' + C)$
- $G = (A' + B + C') + (A + B' + C)$

Complement:

- $A + A' = 1$
- $F = ABC'D + ABCD$
- $F = ABD.(C' + C)$
- $\boxed{F = ABD}$

$0 + 1 = 1$

$1 + 0 = 1$



- $A . A' = 0$
- $G = (A + B + C + D).(A + B' + C + D)$
- $G = (A + C + D) + (B . B')$
- $\boxed{G = A + C + D}$

Distributive Law:

- $A.(B + C) = AB + AC$
-

- $F = WX.(Y + Z)$
- $F = WXY + WXZ$

- $G = B'.(AC + AD)$
- $G = AB'C + AB'D$

- $H = A.(W'X + WX' + YZ)$
- $H = AW'X + AWX' + AYZ$

- $A + (B.C) = (A + B).(A + C)$

- $F = WX + (Y.Z)$
- $F = (WX + Y).(WX + Z)$

- $G = B' + (A.C.D)$
- $G = (B' + A).(B' + C).(B' + D)$

- $H = A + ((W'X).(WX'))$
- $H = (A + W'X).(A + WX')$

NPTEL

Absorption (Covering):

- $A + AB = A$

$A(\underbrace{1+B}_{\text{always } 1})$
$= A.1$
$= A$

- $A.(A + B) = A$

- $F = A'BC + A'$
  - $F = A'$
- $F = A'.(A' + BC)$
  - $F = A'$

- $G = XYZ + XY'Z + X'Y'Z' + XZ$
  - $G = XYZ + XZ + X'Y'Z'$
  - $G = XZ + X'Y'Z'$
- $G = XZ.(XZ + Y + Y')$
  - $G = XZ.(XZ + Y)$
  - $G = XZ$

- $H = D + DE + DEF$
  - $H = D$
- $H = D.(D + E + EF)$
  - $H = D$

NPTEL

Simplification:

- $A + A'B = A + B$

- $F = (XY + Z).(Y'W + Z'V') + (XY + Z)'$
- $F = Y'W + Z'V' + (XY + Z)'$

- $A.(A' + B) = A . B$

- $G = (X + Y).((X + Y)' + (WZ))$
- $G = (X + Y) . WZ$

Logic Adjacency (Combining):

- $A.B + A.B' = A$

$A(B + B') = A \cdot 1$
$= A$

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

- $\underbrace{(A + B).(A + B') = A}$

- $G = (XY + X'Z').(XY + (X'Z')')$
  - $G = XY$

NPTEL

DeMorgan's Law:

- Can be stated as follows:
  - The complement of the product (AND) is the sum (OR) of the complements.
    - $\boxed{(X.Y)' = X' + Y'}$
  - The complement of the sum (OR) is the product (AND) of the complements.
    - $(X + Y)' = X' . Y'$
- Easily generalized to n variables.
- Can be proven using a Truth table

$\overline{x \cdot y} = \bar{x} + \bar{y}$

$\overline{(x + y)} = \bar{x} \cdot \bar{y}$

$\overline{xy \cdot z} \Rightarrow \overline{xy} + \bar{z}$

NPTEL

DeMorgan's Law:

(X . Y)' = X' + Y'

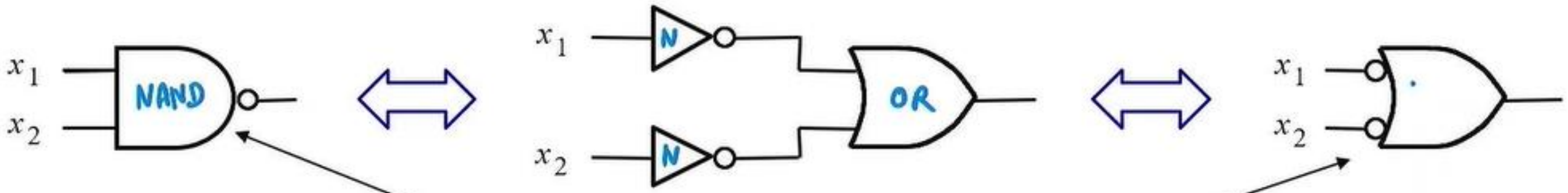
x	y	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

LHS RHS

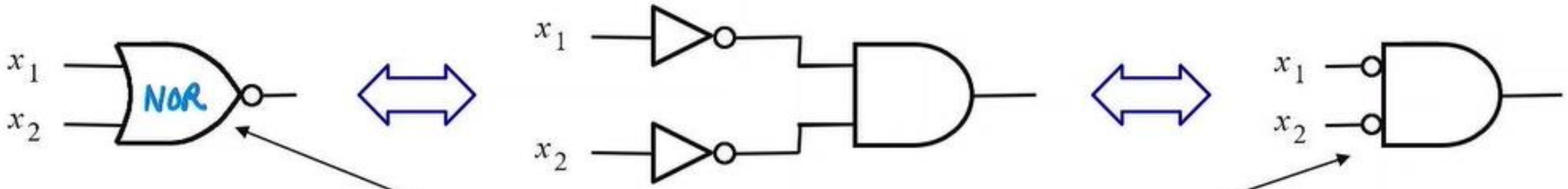
LHS RHS

NPTEL

DeMorgan's Law:



(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$



(b) $\overline{x_1 + x_2} = \bar{x}_1 \cdot \bar{x}_2$

Importance of Boolean Algebra:

- Boolean Algebra is used to simplify Boolean expressions.
  - Through application of the Laws and Theorems discussed

Law's and theorems

- Simpler expressions lead to simpler circuit realization, which, generally, reduces cost, area requirements, and power consumption.
- The objective of the digital circuit designer is to design and realize optimal digital circuits.

NPTEL

Importance of Boolean Algebra:

- Justification for simplifying Boolean expressions:
  - Reduces the cost associated with realizing the expression using logic gates.
  - Reduces the area (i.e. silicon) required to fabricate the switching function.
  - Reduces the power consumption of the circuit.
- In general, there is no easy way to determine when a Boolean expression has been simplified to a minimum number of terms or minimum number of literals.
  - No unique solution