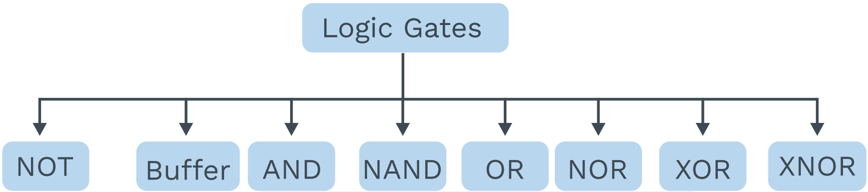
1 Boolean Algebra

1.1 LOGIC GATES

Definition

It is the basic building block to construct any digital circuit.



Note:

We can implement any Boolean function using the above gates.

What are 1 & 0 in Digital Logic?

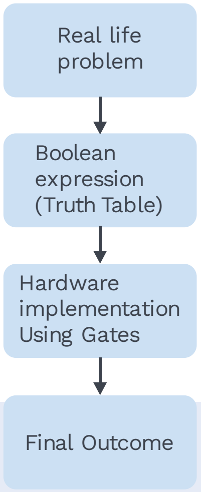
1: True	0: False
1: high	0: low
1: Yes	0: NO

Table 1.1

Note:

Truth table gives value of output for all possible input combination.

How are real-world problems solved using a digital logic system?


Real life problem
Boolean expression (Truth Table)
Hardware implementation Using Gates
Final Outcome

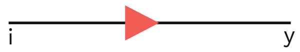
Positive and negative logic:

Definition

In positive logic, 1 represents “high”, and 0 represents “low”.
In negative logic, 1 represents “low”, and 0 represents “high”.

Buffer:

Symbolic representation



y = i [output = input]

Fig. 1.1 Symbolic Representation of Buffer

Truth table	
i	y
0	0
1	1

Table 1.2 Truth Table of Buffer

Note:

A buffer increases the strength of the signal so that it travels for a longer distance

Waveform

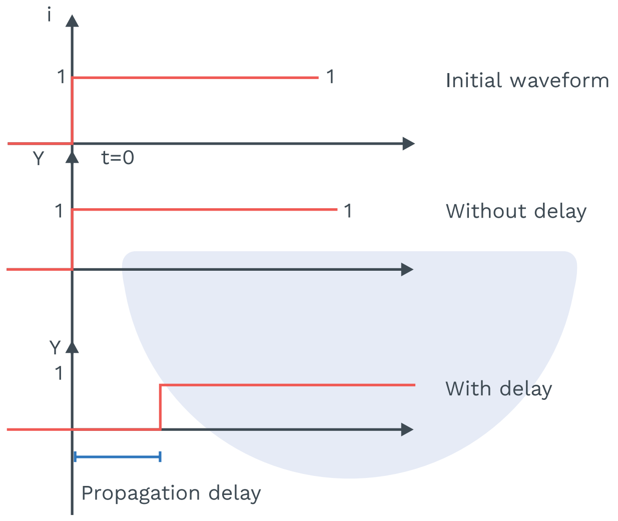


Fig. 1.2 Waveform of Buffer

Propagation Delay is the time taken by a signal to go from input to output of a logic Gate. When the buffer has no delay, it follows the input simply. When the buffer has a propagation delay, then after it senses the input at t = 0, it produces the output after propagation Delay.

Note:

In order to show output with propagation delay we shift output right by t_{pd}.
t_{pd} → propagation delay

Inverter:

Definition

Inverter/NOT gate complements the logic.

Symbolic representation

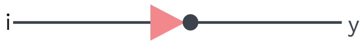


Fig. 1.3 Symbolic Representation of Inverter

Binary logic:

Sample Space, $S = \{0, 1\}$

$0^C = 1$

$1^C = 0$

Grey Matter Alert!

NOT gate and Inverter are same and are used to complement a logic.

Truth table	
i	$y = i^c$
0	1
1	0

Table 1.3 Truth Table of Inverter

Waveform:

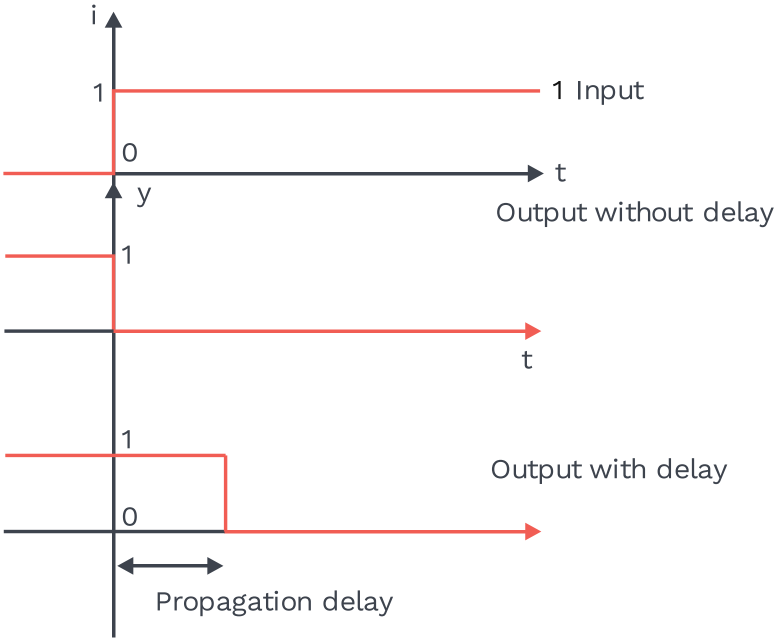


Fig. 1.4 Waveform of NOT Gate

When the NOT Gate has propagation delay, after it senses the input at t=0, it complements the input after the propagation delay.

Cascading of Inverters:

Definition

Whenever o/p of one inverter act as an input to another inverter.

Two inverters in cascade:

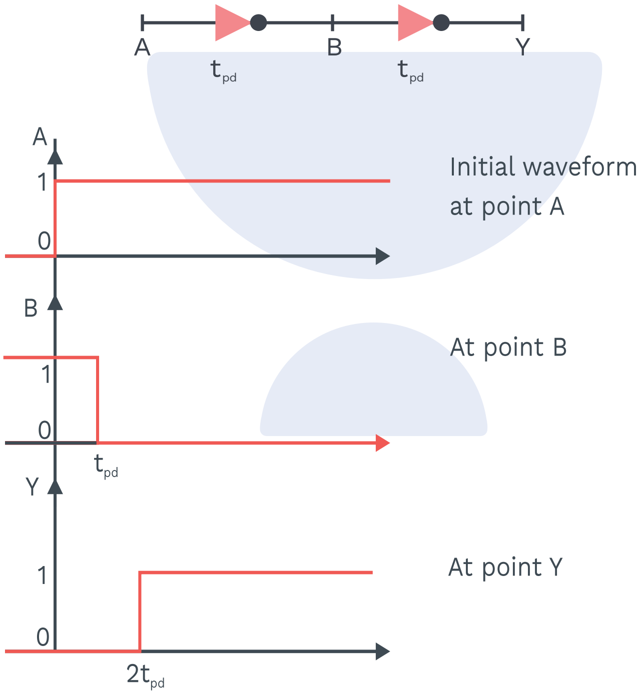


Fig. 1.5 Waveform of two Inverters in Cascade

Note:

Two inverters in cascade behave like a buffer with delay = $2t_{pd}$

Three inverters in cascade:

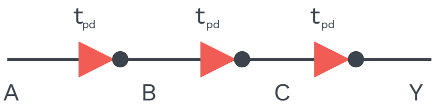


Fig. 1.6 Symbolic Representation of Three Inverters in Cascade

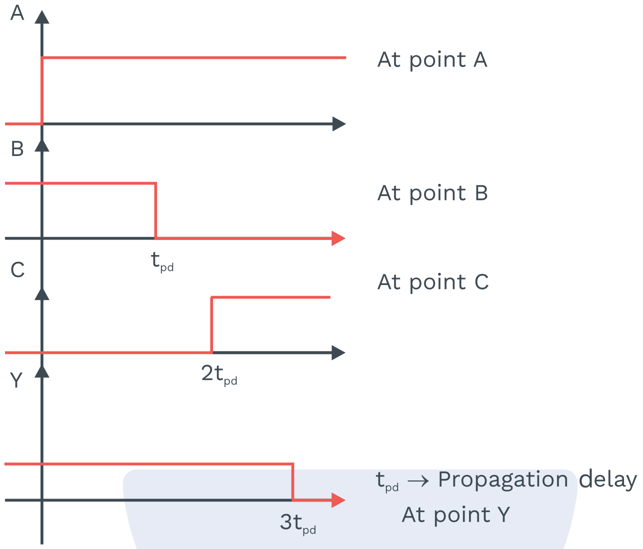


Fig. 1.7 Waveform of Three Inverters in Cascade

Note:

Three inverters in cascade behave like a buffer with delay $3t_{pd}$

NOT gate with feedback:

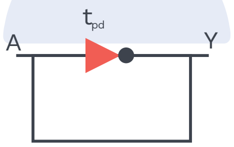


Fig. 1.8 Symbolic Representation of NOT Gate with Feedback

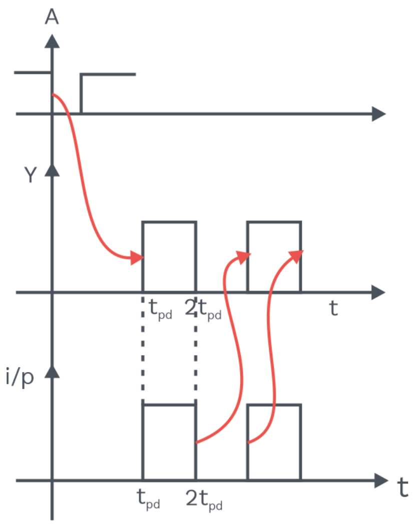


Fig. 1.9 Waveform of NOT Gate with Feedback

- Apply trigger input at ‘A’, which means input is applied for a short duration
- O/p signal is again feedback to the i/p, Arrow in the timing diagram represents input-output effect.
- Whenever input changes, the output will change after t_{pd}.

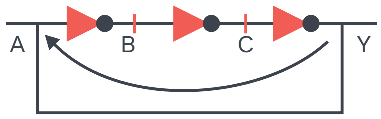


Fig. 1.10 Symbolic Representation of Three Inverters in Cascade with Feedback

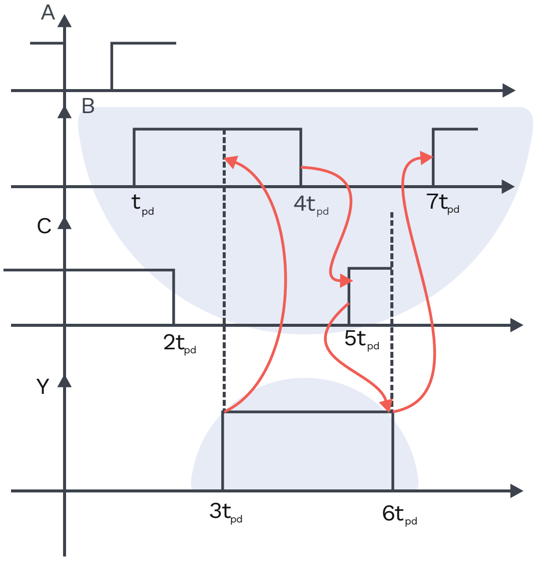
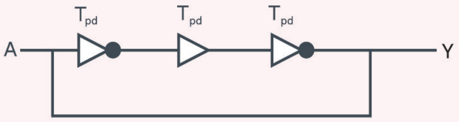


Fig. 1.11 Waveform of Three Inverters in Cascade with Feedback

Note:

Period of square wave (r) = 6tpd
= 2 × 3 × tpd
= 2n tpd
n = number of NOT gates in cascade (odd)

Rack Your Brain



What is the clock period of the output waveform Y?

Note:

Buffer does not impact shape of the Output but provides delay

AND gate:

Definition

Both inputs must be true for the output to be true.

Symbolic representation:

A, B are input, Y is the output

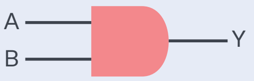


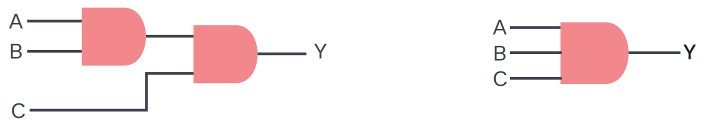
Fig. 1.12 Symbolic Representation of AND Gate

Truth table		
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.4 Truth Table of NAND Gate

O/p = High when both inputs are high, and if any input = low, then output = low

a) Commutative law ⇒ A.B = B.A
b) Associative law ⇒ (AB).C= A.(BC)



(Fan-in=2)

(Fan-in=3)

Fig. 1.13 Symbolic Representation of NAND Gate with 3 i/ps

Note:

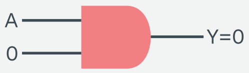
Fan-in defines maximum number of digital inputs that a single logic gate can accept.

Truth table			
A	B	C	Y = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

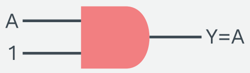
Table 1.5 Truth Table of NAND Gates with 3 i/ps

Grey Matter Alert!

Enable and Disable AND Gate



The input A is not passed to output
∴ it is disable



The input A is passed to output
→ it acts as a buffer
→ it is enabled

Floating input:

- One of the i/p lines is not connected anywhere.

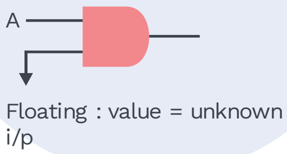


Fig. 1.14 AND Gate with Floating Input

OR gate:

Definition

At least one of the input is High/True for output to be High/True.

Symbolic representation:

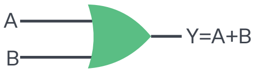


Fig. 1.15 Symbolic Representation of OR Gate

Truth table		
A	B	Y = A + B
0	0	0
0	1	1

1	0	1
1	1	1

Table 1.6 Truth Table of OR Gate

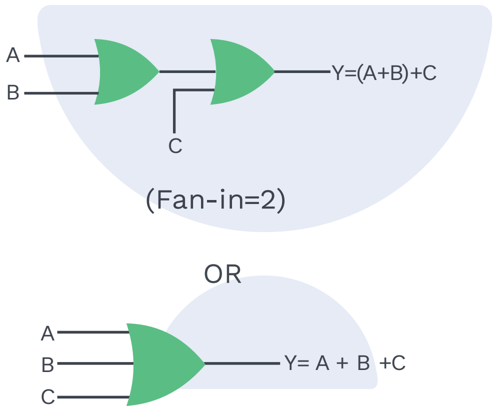
1) **Commutative law:**

   A + B = B + A

   - OR Gate follows commutative law.

2) **Associative law:**

   (A+B) + C = A + (B + C)



(Fan-in=2)

OR

(Fan-in=3)

Fig. 1.16 Truth Table of OR Gate

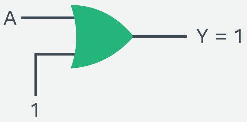
Truth table			
A	B	C	Y = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1

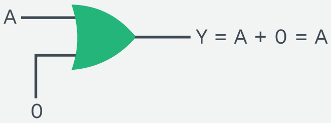
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table 1.7 Truth Table of OR Gate with 3 i/ps

Grey Matter Alert!

Enable & Disable OR gate





→ o/p does not depend on i/p
→ disabled OR gate

→ behaves as a buffer
→ enabled OR gate

Floating i/p:

- Floating i/p is not connected to any value & its value is undetermined.

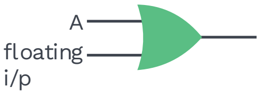


Fig. 1.17 OR Gate with Floating Input

NAND gate:

Definition

It is a combination of invertor and AND gate
NAND = AND output is noted

Symbolic representation:

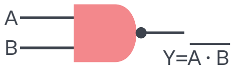


Fig. 1.18 Symbolic Representation of NAND Gate

Truth table		
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 1.8 Truth Table for NAND Gate

Commutative law:

NAND gate follows commutative law.

$$\overline{AB} = \overline{BA}$$

Associative law:

NAND gate does not follow associative law.

$$\overline{\overline{AB} \cdot C} \neq \overline{A \cdot \overline{BC}}$$

NOR gate:

Definition

A HIGH output results if both the inputs to the gate are LOW(0); if one or both input is HIGH(1), a LOW output results.

Symbolic representation:

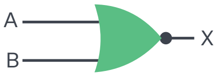


Fig. 1.19 Symbolic Representation of NOR Gate

Truth table		
A	B	$X = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 1.9 Truth Table for NOR Gate

Note:

- NOR gate output is HIGH if both i/ps are low else o/p is LOW
- A NOR gate behaves the same as an AND gate with inverted inputs.
- A NOR gate behaves the same as an OR gate with inverted outputs.

Commutative law

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

NOR gate is commutative.

Associative law

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

$\therefore$ NOT associative.

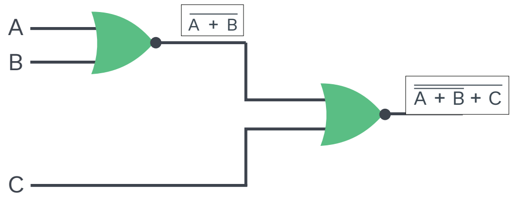
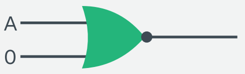


Fig. 1.20

Grey Matter Alert!

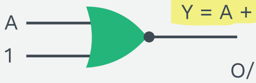
Enable & Disable NOR gate



$Y = \overline{A + 0} = \overline{A}$

O/p depends on input

∴ enabled NOR gate.



$Y = \overline{A + 1} = \overline{1} = 0$

O/p is fixed irrespective of input

∴ Disabled NOR gate.

Floating input:

- Floating input can be made 0 or connected to existing i/p

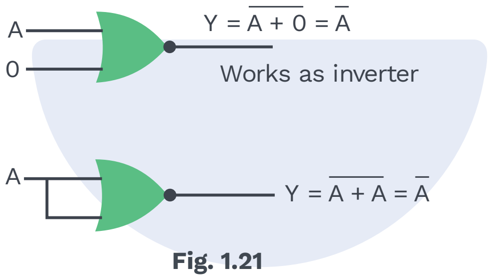

A
0
$Y = \overline{A + 0} = \overline{A}$
Works as inverter
A
$Y = \overline{A + A} = \overline{A}$

Fig. 1.21

Note:

1) $\overline{A + 0} = \overline{A}$

2) $\overline{A + 1} = 0$

3) $\overline{A + A} = \overline{A}$

4) $\overline{A + \overline{A}} = 0$

XOR gate

Definition

XOR gate is a digital logic gate that gives a true output when the number of true or HIGH(1) input is odd.

Symbolic representation:

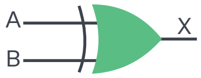


Fig. 1.22 Symbolic Representation of XOR Gate

Truth table		
A	B	X = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.10 Truth Table for XOR Gate

Note:

XOR gate is also called inequality detector

Logic expression:

$X = A\overline{B} + \overline{A}B \xrightarrow{\text{implementation}}$

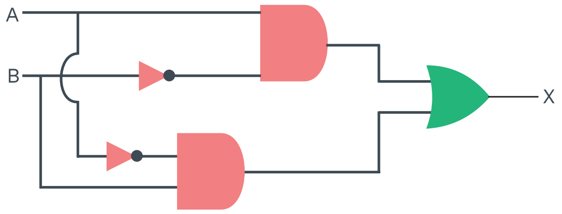


Fig. 1.24 Implementation of XOR Gate Using Basic Gates

Commutative law: A ⊕ B = B ⊕ A
Associative law: A ⊕ (B ⊕ C) = (A ⊕ B) ⊕ C

Properties of XOR gate:

Y = A ⊕ B ⊕ C
o/p = 1 if an odd number of 1’s are present at the input

Truth table			
A	B	C	$Y = A \oplus B \oplus C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 1.11 Truth Table for XOR Gate with 3 i/ps

1) $A \oplus 0 = A\overline{0} + \overline{A}0 = A$(buffer)

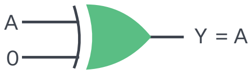


Fig. 1.25

2) $A \oplus 1 = A\overline{1} + \overline{A}1 = \overline{A}$(inverter)

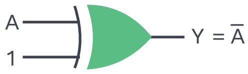


Fig. 1.26

3) $A \oplus A = 0$

4) $A \oplus \overline{A} = 1$

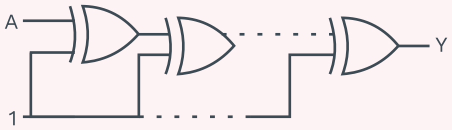
5) if $A \oplus B = C$

Then $A \oplus C = B$

$B \oplus C = A$

Rack Your Brain

Consider a logic circuit with 30 XOR Gates connected as shown below. Find Y



XNOR Gate

Definition

The XNOR gate is a digital logic gate whose function is the logical complement of the Exclusive OR gate.

Symbolic representation:

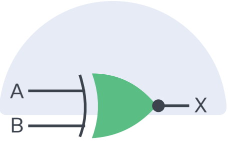


Fig. 1.27 Symbolic Representation of XNOR Gate

Note:

XNOR gate is called equality detector

Truth table		
A	B	$X = A \odot B$
0	0	1
0	1	0

1	0	0
1	1	1

Table 1.12 Truth Table for XNOR Gate

Logic expression:

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

Commutative law:

$A \odot B = B \odot A$

Associative law:

$(A \odot B) \odot C = A \odot (B \odot C)$

Properties:

1) $A \odot 0 = \overline{A}\,\overline{0} + A0$

$= \overline{A} \cdot 1$

$= \overline{A}$ (inverter)

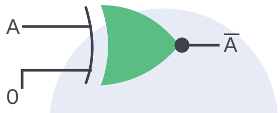


Fig. 1.28 XNOR Gate as Inverter

2) $A \odot 1 = A1 + \overline{A}\,\overline{1}$

$= A + A \cdot 0$

$= A$(Buffer)

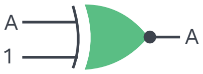


Fig. 1.29 XNOR Gate as Buffer

3) $A \odot A = 1$

4) $A \odot \overline{A} = 0$

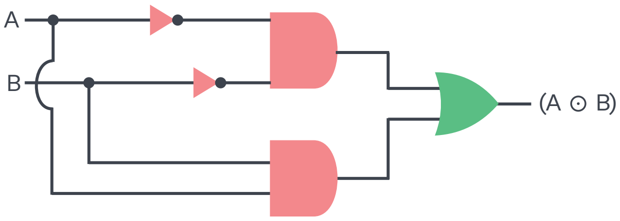


Fig. 1.30 Implementation of XNOR Gate Using Basic Gates

Truth table				
A	B	C	$(A \odot B)$	$Y = A \odot B \odot C$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

Table 1.13 Truth Table for XNOR Gates

Note:

- For odd Number of i/p, XOR = XNOR
- For even Number of i/p, XNOR = $\overline{\text{XOR}}$

1.2 BOOLEAN LAWS

AND law	OR Law	Inversion law
1. $A \cdot 0 = 0$	1. $A + 0 = A$	1. $\overline{\overline{A}} = A$
2. $A \cdot 1 = A$	2. $A + 1 = 1$	
3. $A \cdot A = A$	3. $A + A = A$	
4. $A \cdot \overline{A} = 0$	4. $A + \overline{A} = 1$	

Table 1.14 Laws of Boolean Algebra

Implication and inhibition gate:

Implication gate:

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

if $B = 0, Y = 1$

$B = 1, \; Y = A$ i.e if IMPLY = 1

$Y = A$

Or If ‘B’ is true, then A appears in the output.

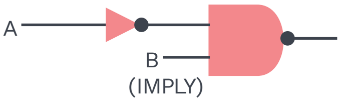


Fig. 1.31 Implication Gate

Inhibition gate:

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

If $B = 1, Y = 0$ i.e A does not reach o/p

$B = 0, Y = A.$

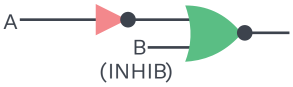


Fig. 1.32 Inhibition Gate

Theorems and properties of Boolean algebra:

1) Idempotent law

- $x.x = x$
- $x + x = x$

2) Associative law

- $(x + y) + z = x + (y + z)$
- $(x.y) . z = x . (y . z)$

3) Complement law

- $x + \overline{x} = 1$
- $x . \overline{x} = 0$

4) Commutative law

- $x + y = y + x$
- $x . y = y . x$

5) Distributive law

- $x . (y + z) = x . y + x . z$
- $x + y . z = (x + y) . (x + z)$

6) Identity law

- $x + 1 = 1$
- $x + 0 = x$
- $x . 0 = 0$
- $x . 1 = x$

Demorgan’s first theorem:

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

Truth table			
A	B	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	1	1

0	1	1	1
1	0	1	1
1	1	0	0

Table 1.15 Truth Table - Demorgan's Law

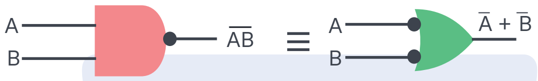


Fig. 1.33 Implementation - Demorgan's Law

NAND gate = Bubbled i/p OR gate

Demorgan's second theorem:

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

Truth table			
A	B	$\overline{A + B}$	$\bar{A} \cdot \bar{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Table 1.16

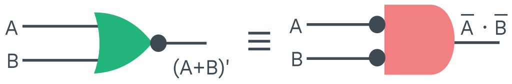


Fig. 1.34

NOR gate = Bubbled i/p AND gate

Transposition theorem:

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

Proof
RHS
= (A + C) (A' + B)
= AA' + AB + A'C + BC
= AB + A'C + BC
= AB + A'C + BC (A + A')
= AB + ABC + A'C + A'BC
= AB (1 + C) + A'C (1 + B)
= AB + A'C = LHS

Consensus theorem/ redundancy theorem:
The Redundancy theorem is used to simplify Boolean expressions. It is also known as consensus theorem.

$$\boxed{AB + A'C + BC = AB + A'C}$$

The consensus of the terms AB and A'C are BC. The conjunctive dual of the above equation is:

$$\boxed{(A + B)(A' + C)(B + C) = (A + B)(A' + C)}$$

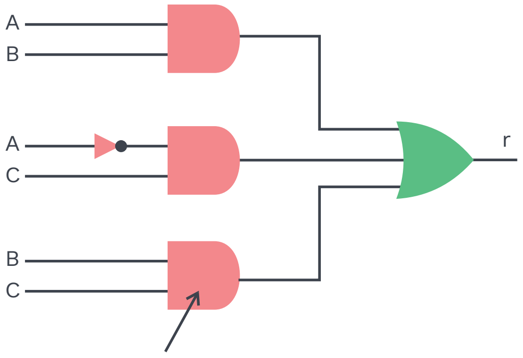


Fig. 1.35

Example $\Rightarrow \overline{B}C + \overline{A}C + B\overline{A}$
$= \overline{B}C + B\overline{A}$

Conditions for applying the redundancy theorem are:

1) Three variables must be present in the Boolean expression.

2) Each variable is repeated twice.

3) One variable in the expression must be present in both complemented and uncomplemented forms.

After applying this theorem, we can only take those terms which contain the complemented variable.

Proof

$Y = AB + A'C + BC$

$Y = AB + A'C + BC(1)$

$Y = AB + A'C + BC\,(A + A')$

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

$Y = AB\,(1 + C) + A'C\,(1 + B)$

$Y = AB + A'C$ (Proved)

Redundant literal rule:

$A + A'B = A + B$ [Literal $\bar{A}$ is redundant & can be removed]

Similarly,

$A\,(A' + B) = AB$

Proof

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

$\Rightarrow A + AB + A'B = A + B\,(A + A')$

$\Rightarrow A + B\,(1) = A + B$

Duality:

Definition

This theorem states that the dual of the Boolean function is obtained by interchanging logical AND operator with logical OR operator and 0's with 1's & vice-versa.

Following Boolean equations are dual to each other:

Group 1		Group 2
$x + 0 = x$	↔	$x . 1 = x$
$x + 1 = 1$	↔	$x . 0 = 0$
$x + x = x$	↔	$x . x = x$
$x + x' = 1$	↔	$x . x' = 0$
$x + y = y + x$	↔	$x . y = y . x$
$x + (y+z) = (x + y)+z$	↔	$x . (yz) = (x . y) . z$
$x(y+z) = xy + xz$	↔	$x + yz = (x + y) (x + z)$

Complementary theorem:

For obtaining complement expression,

1) Change each OR sign by AND sign and vice-versa.
2) Complement any 0 or 1 appearing in the expression.
3) Complement the individual literals.

Example ⇒ Complement of $A(B+C) = A'+(B'\cdot C') = (A'+B')(A'+C')$

1.3 BOOLEAN FUNCTION

Definition

A Boolean function is described by an algebraic expression consisting of binary variables, the constants 0 and 1 and logical symbols '+' and ' . '. For a given set of input, a Boolean function can have value of '0' or '1'

A Boolean function can be represented in canonical form:

i) POS (product of sum)
ii) SOP (sum of product)

Example

$f = a + bc$

Truth table

a	b	c	f

0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table 1.17 Truth Table of 'f'

Rack Your Brain

How many m-ary functions are possible with 'n' K-ary variables?

SOLVED EXAMPLES

Q1 **f (p, q, r) = p'q + pqr + r', Represent it in canonical SOP form**

Sol: f (p, q, r) = p' q + pqr + r'
= p'q (r + r') + pqr + r’ ' (p + p') (q + q')
= p'qr + p'qr' + pqr + pqr' + p'qr' + p'q'r' + pq'r'
= p'qr + pqr + pqr' + pq'r' + p'q'r' + p'qr'

Functional properties:

- The canonical sum of product or product of sum form of a switching function is UNIQUE.
- The switching function $f_1(x_1.......x_n)$ and $f_2(x_1,x_2.....x_n)$ are said to be logically equivalent if both functions have same value for each and every combination of x_1, x_2....... x_n.
- Two switching functions are equivalent, if their canonical SOP and canonical POS expressions are equal.

Note:

For a switching function, there can be many expressions which will define the same function, but when we convert them to their corresponding SOP and POS form, it is UNIQUE.

Neutral function:

Number of minterms = Number of maxterms

Example

For 2 variable Boolean function number of neutral functions possible = $^4C_2$

A	B	f_1	f_2
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Table 1.20 Neutral Functions with 2 Variables

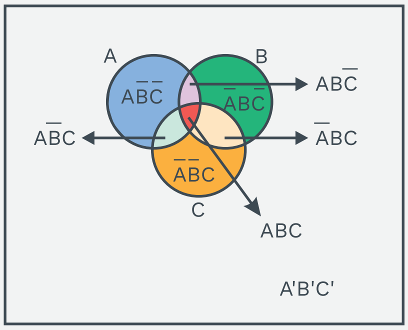
For 'n' variable function:

Number of neutral function $= {}^{2^n}C_{\frac{2^n}{2}}$

$= {}^{2^n}C_{2^{n-1}}$

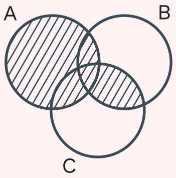
Grey Matter Alert!

Venn diagram representation:



Rack Your Brain

Find the Boolean expression



Simplifications of Boolean functions:

1) Simplification using algebraic functions:

SOLVED EXAMPLES

Q2 **F(X,Y,Z)=(X+Y)(X+Z)**

Sol: F(X,Y,Z) = (X+Y)(X+Z)

= X.X + X.Z + Y.X + Y.Z

= X + X.Z + X.Y + Y.Z

= X (1 + Z) + X.Y + Y.Z

= X + X.Y + Y.Z

$= X(1 + Y) + Y.Z$

$= X + Y.Z$ (minimized form)

2) Simplification using K-map: (Discussed later)

Complement of Boolean function:

We can find the complement of a function using given 2 rules in Demorgan’s law:

1) Change the OR Gates with AND Gates or change the AND Gates with OR Gates

2) Change each literal of the function with its complement.

Example $F = pq' + p'q$

$F' = (p' + q) \cdot (p + q')$

$= p'q' + pq$ $[\because p \cdot p' = 0]$

1.4 STANDARD SOP AND POS FORMS

The main advantage of standard forms is to minimize the number of inputs to logic gates. Sometimes there will be a reduction in the total number of logic Gates required to represent a Boolean function.

1) Standard SOP form

2) Standard POS form

Note:

- Canonical form is unsimplified
- Standard form is simplified

Standard sum of product (SOP):

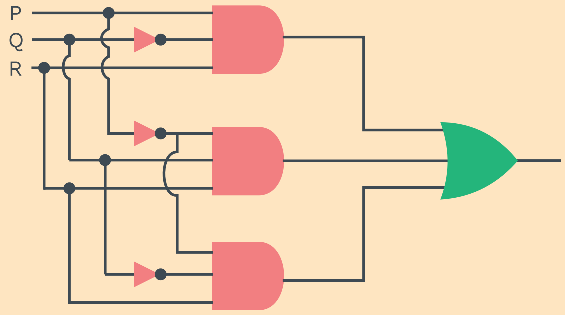
Definition

Standard SOP form means standard sum of products form. In this form, each product term need not contain all literals. So product terms may/may not be minterms.

∴ Standard SOP form is the simplified form of canonical SOP form.

SOLVED EXAMPLES

Q3 **Implement the function given below using SOP and POS forms:**

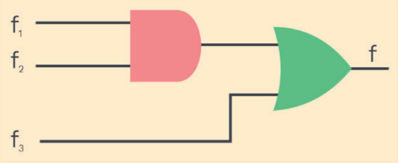


Sol: $F = P\overline{Q}R + \overline{P}QR + \overline{P}\,\overline{Q}R$

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

$= \pi\ M\ (0, 2, 4, 6, 7)$

Q4 **Consider the three 4 variable functions f_1, f_2, f_3.**
Given, f_1, f_3 and f in canonical SOP (m decimal) for the circuit
$f_1 = \Sigma m\ (8, 6, 7, 4, 5)$
$f_3 = \Sigma m\ (1, 6, 15)$
$f = \Sigma m\ (6, 1, 8, 15)$

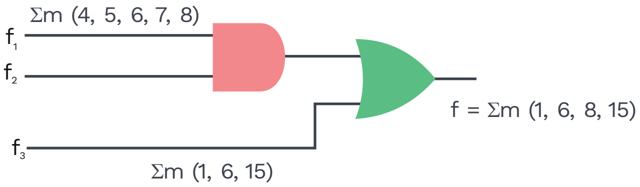


Then f_2 is -

1) $\Sigma m\ (4, 6)$ **2) $\Sigma m\ (4, 8)$**

3) $\Sigma m\ (6, 8)$ **4) $\Sigma m\ (4, 6, 8)$**

Sol: AND: Intersection of minterms
i.e. common minterm between 2 function
OR: Union of minterms



$f = (f_1 \wedge f_2) \vee f_3$

f_2 must be $\sum m\ (6, 8)$

Option (3)

1.5 CANONICAL FORM

Minterms and maxterms:

Minterms:

Definition

A minterm of n variables is a product of variables in which each variable appears exactly once in true or complemented form.

Each minterm = 1 for only one combination of values of the variables.
= 0, otherwise

A	B	C	Minterm
0	0	0	$\bar{A}\bar{B}\bar{C}$ m_0
0	0	1	$\bar{A}\bar{B}C$ m_1
0	1	0	$\bar{A}B\bar{C}$ m_2
0	1	1	$\bar{A}BC$ m_3
1	0	0	$A\bar{B}\bar{C}$ m_4
1	0	1	$A\bar{B}C$ m_5
1	1	0	$AB\bar{C}$ m_6
1	1	1	ABC m_7

Table 1.21 Minterms of 3 variables

Note:

For a particular combination of input values, if any input variable = 0, take complement else use as it is & take product of all such variables.

Maxterms:

Definition

A maxterm of n variables is sum of variables in which each variable appears exactly once in true or complemented form. Each maxterm = 0 for only one combination of values of the variables.

= 1, otherwise

Note:

For a particular combination of input values, if any input variable = 1, take complement else use as it is and take sum of all such variables.

Canonical SOP and POS forms:

Definition

If there are 'n' input variables, then there will be 2^n possible combinations with zeros and ones. So value of each output variable depends on combination of input variable, So each output variable will have “1” for some combination of input variables and 'O' for some other combination of input variables.

Therefore each output variable can be expressed in following ways:

- Canonical SOP form ⟶ minterms: product of variables
- Canonical POS form ⟶ maxterm: sum of variables

Canonical SOP

Canonical SOP form means the canonical sum of product form. In this form, each product term contains all variables or literals. Hence canonical SOP is called the sum of minterms form.

Example

Input			Output
x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1 m_3
1	0	0	0
1	0	1	1 m_5
1	1	0	1 m_6
1	1	1	1 m_7

Table 1.22 Table Showing Minterms

$f = m_3 + m_5 + m_6 + m_7$
$f = \sum m\ (3, 5, 6, 7)$

The Boolean function of the output f is x'yz + xy'z + xyz' + xyz

Canonical POS:

Definition

Canonical POS form means canonical product of sum from. In this form, each sum term contains all literals. So these Sum terms are the Maxterms. First, identify the Max terms for which the output variable is zero and then perform logical AND of those MAX-terms to get Boolean expression function corresponding to that output variable.

Input			**Output**
p	q	r	f
0	0	0	0 M_0
0	0	1	0 M_1
0	1	0	0 M_2
0	1	1	1
1	0	0	0 M_4
1	0	1	1
1	1	0	1
1	1	1	1

Table 1.23 Table Showing Maxterms

$f = M_0. M_1. M_2. M_4$

$f = \pi M (0, 1, 2, 4)$

$f = (p + q + r) (p + q + r') (p + q' + r) (p' + q + r)$

Conversion between canonical forms:

Let's look at an example:

$$F = ABC + AB\overline{C} + A\overline{B}C + \overline{A}BC$$

In terms of minterms the functions can be written as:

$F = m_3 + m_5 + m_6 + m_7 = \sum m\,(3, 5, 6, 7)$ Sum of minterms

In terms of maxterms the function can be written as:

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

$= \pi\ M\ (0, 1, 2, 4) \leftarrow$ Consider those maxterms which are not included in minterms

Product of Maxterms.

Conversion In Standard Form:

SOLVED EXAMPLES

Q5 **Convert the following Boolean function into standard SOP form:**
$f = A'\,B\,C + A\,B'\,C + A\,B\,C' + A\,B\,C$

Sol:

$$
\begin{aligned}
f &= A'\,B\,C + A\,B'\,C + A\,B\,C' + A\,B\,C \\
&= A'\,B\,C + A\,B'\,C + AB \\
&= B\,[A + A'\,C] + A\,B'\,C \\
&= B\,[A + C] + A\,B'\,C \\
&= A\,B + B\,C + A\,B'\,C \\
&= A\,[B + B'\,C] + B\,C \\
&= A\,(B + C) + B\,C \\
&= A\,B + A\,C + B\,C
\end{aligned}
$$

Q6 **Convert the following Boolean function into standard POS form:**
$f = (X + Y + Z)\,(X + Y + Z')\,(X + Y' + Z)\,(X' + Y + Z)$

Sol:

$$
\begin{aligned}
f &= (X + Y + Z)\cdot(X + Y + Z')\,(X + Y' + Z)\,(X' + Y + Z) \\
f &= (X + Y + Z)\,(X + Y + Z')\,(X + Y + Z)\,(X + Y' + Z)\,(X + Y + Z)\,(X' + Y + Z) \\
f &= (X + Y + Z\,Z')\,(X + Z + Y\,Y')\,(Y + Z + X\,X') \\
f &= (X + Y)\,(X + Z)\,(Y + Z)
\end{aligned}
$$

using $(A+B)\cdot(A+C)= A + BC]$

1.6 MINIMIZATION OF BOOLEAN FUNCTION

Logic minimization – using K-map:

Definition

K-Map is a graphical method which consists of 2^n cells for 'n' variables. The adjacent cells are differed by a single bit position.

Note:

K-map method is most suitable for minimizing the Boolean Function of 2 variables to 5 variable.

Grouping of K-map variables:

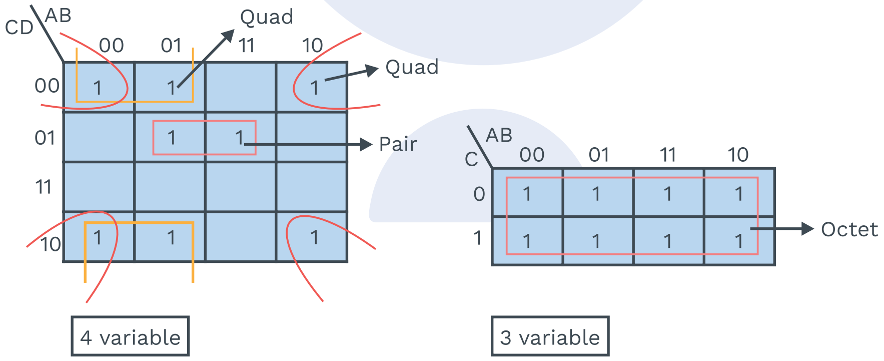


Fig. 1.36 Grouping of K-map Variables

- Groups can overlap.
- The number of literals in a group must be equal to the power of 2, such as 1,2,4,8 etc.
- Groups can wrap around. The squares at the corners (which are at the end of a column or row) should be considered as adjacent squares.
- The grouping of K-map variables can be done in different ways, So obtained simplified equations need not be UNIQUE always.
- The boolean equation must be in canonical form (SOP/POS) to draw the K–map.
- If we group 2^K cells together, then K variables are eliminated.

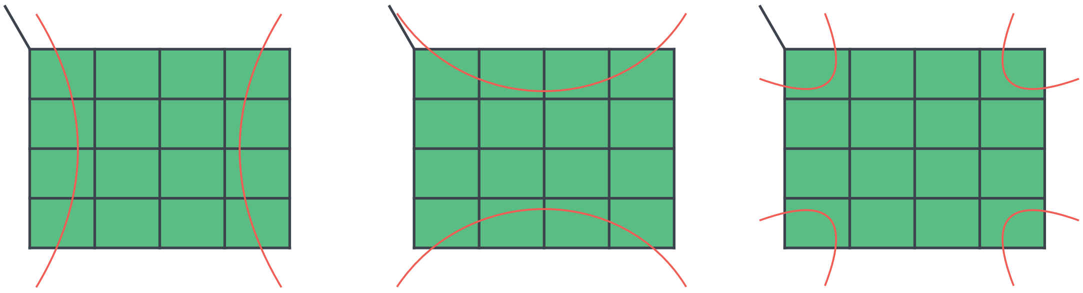


Fig. 1

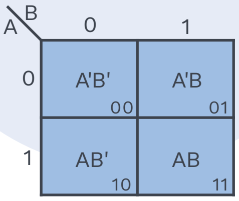
Fig. 2

Fig. 3

Fig. 1.37 Wrapping Around K-map

2) Variable K-maps:

The possible minterms with 2 variables (A and B) are AB, AB’, A’B, A’B’



A \ B	0	1
0	A'B' 00	A'B 01
1	AB' 10	AB 11

Fig. 1.38

Example

A	B	F
0	0	1
0	1	1
1	0	0
1	1	0

Table 1.24

- For SOP, consider minterms, where F = 1.

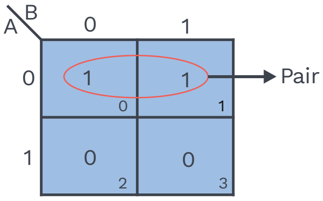


Fig. 1.39

In the pair, ‘B’ is changing from 0 to 1.

∴ $\boxed{F = \overline{A}}$

- For POS consider maxterms, where F = 0

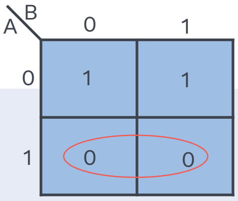


Fig. 1.40

In the pair, ‘B’ is changing from 0 to 1.

∴ $\boxed{F = \overline{A}}$ (maxterm)

3-Variable K-maps: 8 Possible minterms are:

A	B	C	O/P function
0	0	0	A'B'C'
0	0	1	A'B'C
0	1	0	A'BC'
0	1	1	A'BC
1	0	0	AB'C'
1	0	1	AB'C
1	1	0	ABC'
1	1	1	ABC

Table 1.25

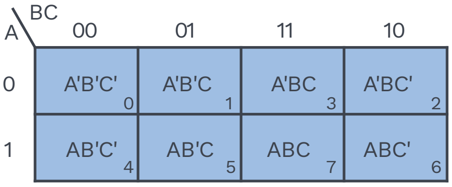


Fig. 1.41

Example

F = $\sum$m (5, 1, 0, 2, 4)

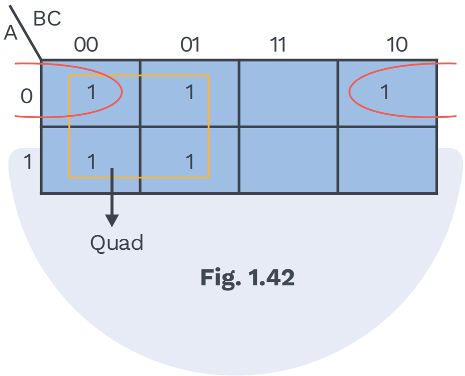


Fig. 1.42

$\boxed{F = \overline{B} + \overline{A}\overline{C}}$

Example

F = π M (3, 6, 7)

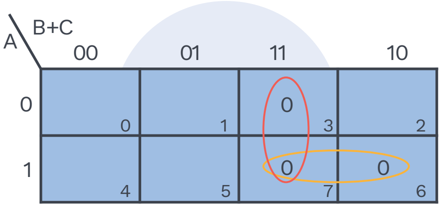


Fig. 1.43

$\boxed{F = (\overline{B} + \overline{C})(\overline{B} + \overline{A})}$

4 variable K-map:

F (y, z, w, x) = $\sum$m (4, 1, 5, 7, 13, 9, 14, 15)

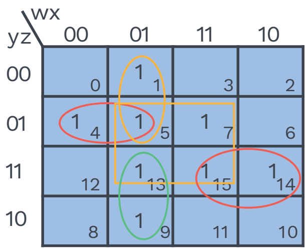


Fig. 1.44

F = w'y'z + w'xy' + w'xy + wyz + xz (14 literals)

But this is not a minimized expression; we need to do the grouping carefully so that the biggest group ("octet" then "Quad" then "pair") gets the first priority.

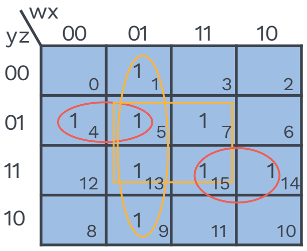


Fig. 1.45 4-Variable K-Map

$f = w'y'z + wyz + w'x + xz$ (10 literals) (minimum)

1.1.2 Implicants, prime implicants and essential prime implicants:

Definition

If 'f' covers 'g' then 'g' is said to imply 'f'. This is denoted by $g \rightarrow f$ ['g' is a implicant of 'f']

Ex. = $f(x, y, z) = xy + z$

x	y	z	$f_1 = xy$	$f_2 = z$	$f = xy + z$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	1	1

1	1	0	1	0	1
1	1	1	1	1	1

Table 1.26 Truth Table of 'f'

∵ f is covering both f_1 and f_2

if f_1 or f_2 is 1, f is 1

∴ f_1 and f_2 are implicants of f.

$f_1 \rightarrow f$

$f_2 \rightarrow f$

Note:

In an SOP expression, each product term is implicant of the original function.

Prime implicant:

Definition

An implicant 'P' of a function 'f' is said to be prime implicant if:

i) P is any product term (i.e. subcube)

ii) Subcube P can not be part of any bigger subcubes completely

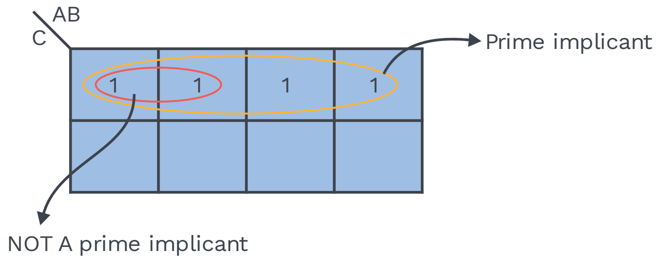


Fig. 1.46 3-Variable K-Map

(∵ it is a part of the bigger subcube completely)

Essential prime implicants:

Definition

A Prime implicant 'P' of a function 'f' is said to be an essential prime implicant if it covers at least one minterm of 'f', which is not covered by any other prime implicants.

Example

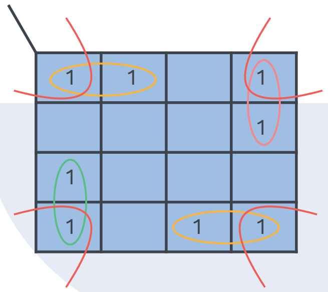
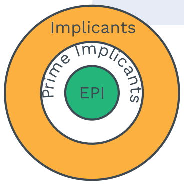


Fig. 1.47 K-Map Showing Prime Implicants and Essential Prime Implicants

5 Prime implicants
4 Essential prime implicants



EPI represents essential prime implicants

Fig. 1.48

Rack Your Brain

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

Find number of prime implicants and essential prime implicants.

Note:

In cyclic K map

- Every minterms is covered by atleast 2 prime implicants.
- All prime implicants are of the same size.

Procedure for obtaining minimal SOP:

1) Determine all essential prime implicants and include them in the minimal SOP.

2) Remove all those prime implicants which are covered by essential prime implicants.

3) If the set determined in step 1 covers all the minterms of ‘f’ then it is a unique minimal expression; otherwise, select the additional prime implicants so that the function ‘f’ is covered completely and the total number and size of prime implicants added are minimal.

Note:

Prime implicants will be chosen such that literals in the term are minimum.

Example

F(yzwx) = $\sum$m (4, 3, 5, 7, 13, 9, 14, 15)

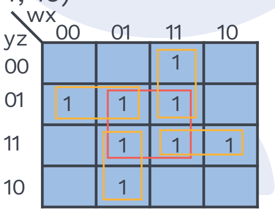


Fig. 1.48

5 Prime implicants

4 Essential prime implicants

f = w'y'z + wxy' + wyz + w'xy

Prime implicant chart:

Prime implicants ↓ / Minterms →	✓ 3	✓ 4	✓ 5	✓ 7	✓ 9	✓ 13	✓ 14	✓ 15
✓ (EPI) $\overline{w}\,\overline{y}\,z$		1	1					
✓ (EPI) $w\,x\,\overline{y}$	1			1				
✓ (EPI) $w\,y\,z$							1	1
✓ (EPI) $\overline{w}\,x\,y$					1	1		
x z Redundant			1	1		1		1

Fig. 1.49 Prime Implicant Chart

- All essential prime implicants are included in SOP (Find essential prime implicants from k-map).
- Include the prime implicants in the minimal expression which is covering maximum minterms.
- All minterms must be covered.
- 'xz' is redundant prime implicant since after including all the essential Prime implicants, all of the minterms get covered.

Note:

Redundant Prime Implicants are never included in minimal SOP.

Don't care conditions:

Definition

- A function is said to be completely specified if it is given 0 or 1 for every combination of i/p variables.
  There are some function which are not completely specified.
- Combination for which value of function is not fully specified are called don't care combinations.
- Since each don't care combination represent two values {0, 1}, an incompletely specified function containing K don't care combinations, corresponds to a class of 2^k distinct function.

Example

a	b	f	f_1	f_2	f_3	f_4
0	0	1	1	1	1	1
0	1	1	1	1	1	1
0	0	Ø	0	0	1	1
1	1	Ø	0	1	0	1

Table 1.27 Table Showing Don't Cares

Ø ← don't care

Note:

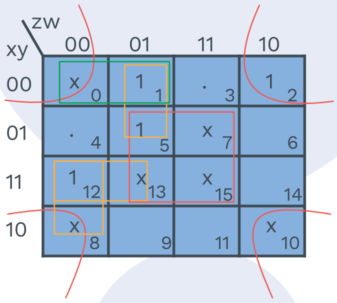
We can assign '0' or '1' to a don't care Ø in such a way to increase/decrease the size of a prime implicant.

Understanding through examples:

SOLVED EXAMPLES

Q7 **F = S m (2, 1, 5, 12) + S d (0, 7, 8, 10, 13, 15)**
Find minimized Boolean expression

Sol:



xy \ zw	00	01	11	10
00	x 0	1 1	. 3	1 2
01	. 4	1 5	x 7	6
11	1 12	x 13	x 15	14
10	x 8	9	11	x 10

x ← don't care
6 prime implicant

Note:

Here, EPIs do not include all the minterms so, minimal SOP is not UNIQUE here.

2 Possibilities exist:

1) 2 QUAD, 2 Pair $\Big\}$ $F = yw + \overline{y}\overline{w} + \overline{x}\,\overline{y}\,\overline{z} + x\overline{z}\overline{w}$ (10 literals)

(NOT Minimal)

2) 1 QUAD, 2 Pair $\Big\}$ $F = \overline{y}\overline{w} + \overline{x}\overline{z}w + x\overline{z}\overline{w}$ (8 literals)

(Minimal)

Let's look at how to find the number of minimal expressions.

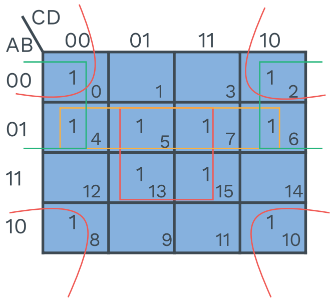


Fig. 1.50

Minterms →	✓ 0	✓ 2	4	✓ 5	6	✓ 7	✓ 8	✓ 10	✓ 13	✓ 15
PI. ↓										
$\overline{B}\,\overline{D}$ (EPI)	1	1					1	1		
B D (EPI)				1		1			1	1
A' B			1	1	1	1				
A' D'	1	1	1		1					

Both these prime-implicants are covering the left over minterms (m_4, m_6) we can include either of them in the final expression

Fig. 1.51 Prime Implicant Chart

Final SOP ~ $\boxed{\overline{B}\,\overline{D} + BD + \overline{A}\,B}$

OR

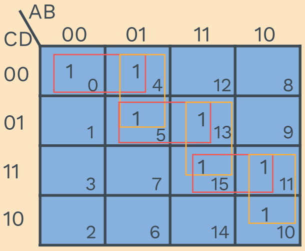
$\boxed{\overline{B}\,\overline{D} + BD + \overline{A}\,\overline{D}}$

Note:

Both minimal SOP and minimal POS will be independent of the same number of variables if there are no don't cares.

SOLVED EXAMPLES

Q8 **Find number of Minimal SOP in the following K-map**



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

Sol: Prime implicant chart

		✓ 0	✓ 4	5	13	15	✓ 11	✓ 10
(EPI)	$P = \overline{A}\,\overline{C}\,\overline{D}$	1	1					
	$Q = \overline{A}\,B\,\overline{C}$			1				
	$R = B\,\overline{C}\,D$			1	1			
	$S = A\,B\,D$				1	1		
	$T = A\,C\,D$					1	1	
(EPI)	$V = A\,\overline{B}\,C$						1	1

Let's reduce the prime implicant chart (include all the EPIs in the Minimal SOP and removing the all minterms associated with the EPIs)

	5	13	15
Q	1		
R	1	1	
S		1	1
T			1

Minimal SOP →
$$\left.\begin{array}{c} P + V + R + S \\ \text{OR} \\ P + V + R + T \\ \text{OR} \\ P + V + Q + S \end{array}\right\} \text{3 Minimal SOP}$$

Variable entrant map: (VEM)

Definition

A variable entrant map (VEM) is a K-map in which the size of the map is reduced by removing one or more variables from the specification of the map cell locations

$f(x, y, z) = \sum(1, 2, 3, 5, 6)$

X	Y	Z	f	
0	0	0	0	} z
0	0	1	1	
0	1	0	1	} 1
0	1	1	1	
1	0	0	0	} z
1	0	1	1	
1	1	0	1	} $\bar{z}$
1	1	1	0	

$\xrightarrow{\text{VEM}}$

X	Y	f
0	0	z
0	1	1
1	0	z
1	1	$\bar{z}$

Table 1.28 Table Showing VEM

Minimization procedure for VEM:

- Set all the variables in the cell as ‘0’ and obtain the SOP expression.
- Make one variable in the cell as ‘1’ and obtain SOP by making earlier minterms as don’t cares (Ø) and multiply the obtained SOP expression with the concerned variable.
- Repeat step 2 until all the variables in the cell are covered.
- SOP of VEM is obtained by performing ‘OR’ operation with previously obtained SOP expression.

SOLVED EXAMPLES

Q9 **Find the minimized Boolean expression for the following VEM**

Z \ XY	00	01	11	10
0	D	1	D'	D'
1	D	1	0	Ø

Sol: **Step 1**

Z \ XY	00	01	11	10
0	0	1	0	0
1	0	1	0	Ø

SOP = X'Y

Step 2

For D :

Z \ XY	00	01	11	10
0	1	Ø	0	0
1	1	Ø	0	Ø

SOP = X'

For D' :

Z \ XY	00	01	11	10
0	0	Ø	1	1
1	0	Ø	0	Ø

SOP = XZ'

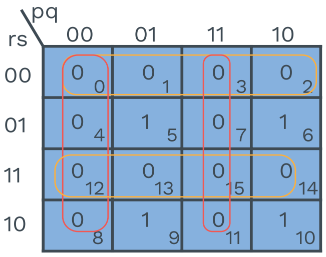
Step 3

$\boxed{X'Y + X'D + XZ'D'}$

Product of Sum Simplification

Q10 **f (p,q,r,s) = π M (0,1,2,3,4,7,8,11,12,13,14,15) Find Minimal**

Sol:



rs \ pq	00	01	11	10
00	0_{0}	0_{1}	0_{3}	0_{2}
01	0_{4}	1_{5}	0_{7}	1_{6}
11	0_{12}	0_{13}	0_{15}	0_{14}
10	0_{8}	1_{9}	0_{11}	1_{10}

4 Essential Prime Implicants

Minimal POS = $(r+s)(p+q)(\overline{r}+\overline{s})(\overline{p}+\overline{q})$ ← [8 literals]

Note:

Minimal expression = All essential prime implicants
+
Prime implicants needed to cover remaining minterms,if any.

Previous Years' Question

The literal count of a Boolean expression is the sum of the number of times each literal appears in the expression. For example, the literal count of (xy + xz') is 4. What are the minimum possible literal counts of the product-of-sum and sum-ofproduct representations respectively of the function given by the following Karnaugh map? Here, X denotes "don't care"

xy \ zw	00	01	11	10
00	X	1	0	1
01	0	1	X	0
11	1	X	X	0
10	X	0	0	X

1) (11, 9)
2) (9, 13)
3) (9, 10)
4) (11, 11)

Sol: 3) **(GATE-2003)**

1.7 NAND AND NOR

Universal gates:

Definition

- A universal gate can implement any Boolean function without using gates of any other type. NAND and NOR are the universal gates.
- In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all ic digital logic families.
- In fact, AND gate is implemented as a NAND gate followed by an inverter. OR gate is implemented as a NOR gate followed by an inverter.

1.7 IMPLEMENTATION OF BOOLEAN FUNCTIONS WITH NAND GATES

NAND Gate is a Universal Gate

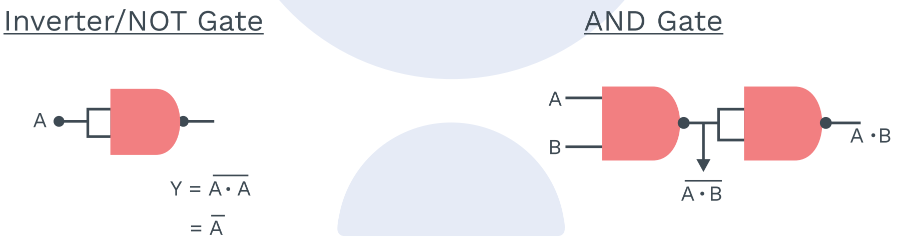


Fig. 1.50 Implementation Using NAND Gate

OR gate:

$$Y = A + B = \overline{\overline{A} \cdot \overline{B}} \qquad \text{[From Demorgan's law]}$$

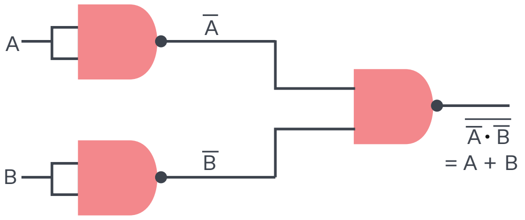


Fig. 1.51 Implementation Using NAND Gate

NOR gate:

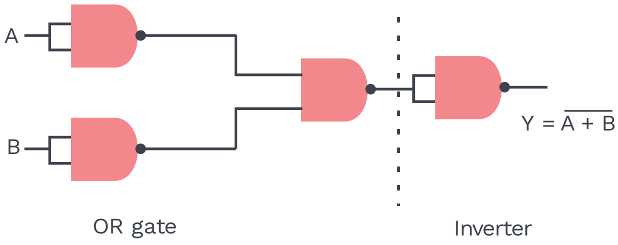


Fig. 1.52 Implementation Using NAND Gate

X-OR:

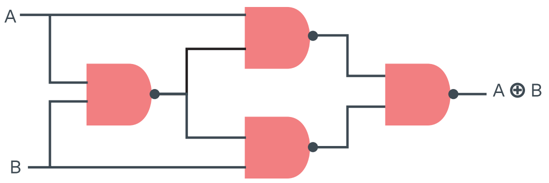


Fig. 1.53 Implementation of X-OR Gate using NAND Gate

4 NAND gates [minimum] required

XNOR:

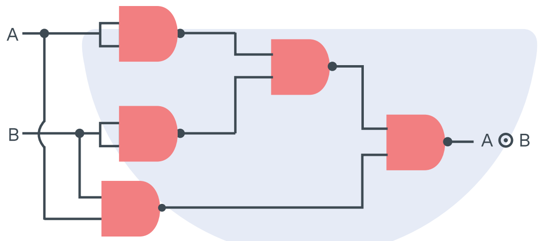


Fig. 1.54 Implementation of X-NOR Gate using NAND Gate

5 NAND Gates (minimum)

Implementation of Boolean functions with NOR gates:

NOR gate is a universal gate

Inverter/Not gate

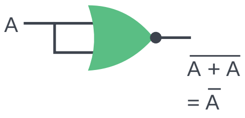


Fig. 1.55 Implementation of NOT Gate Using NOR Gate

AND gate

$y = A \cdot B = \overline{\overline{A} + \overline{B}}$ [By Demorgan's law]

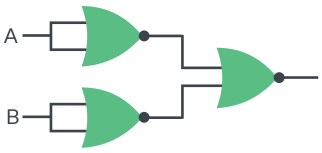


Fig. 1.56 Implementation of AND Gate Using NOR Gate

OR gate

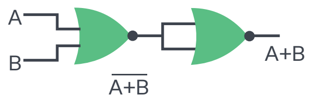


Fig. 1.57 Implementation of OR Gate Using NOR Gate

NAND gate

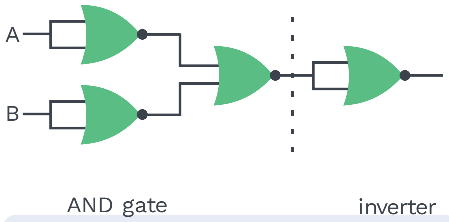


Fig. 1.58 Implementation of NAND Gate Using NOR Gate

X-OR

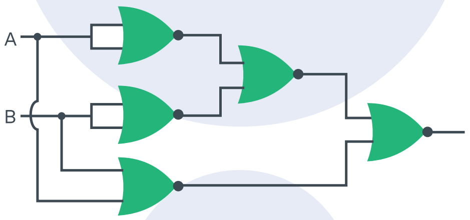


Fig. 1.59 Implementation of XOR Gate using NOR Gate

XOR = XNOR + Inverter

5 NOR gates [minimum]

XNOR gate

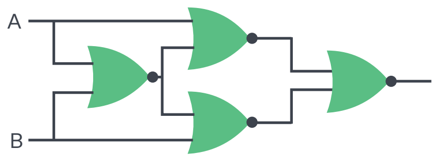


Fig. 1.60 Implementation of XNOR Gate Using NOR Gate

4 NOR gates [minimum] required

Note:

	Min NAND gates	Min NOR gates
X-OR	4	5
X-NOR	5	4

- Bubbled i/p OR gate = NAND gate
- Bubbled i/p AND gate = NOR gate.

Example $\boxed{Y = AB + CD}$

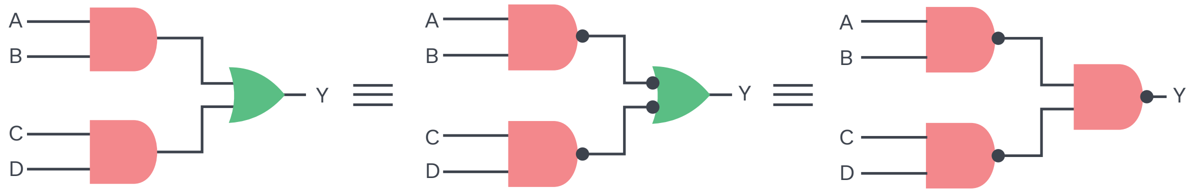


Fig. 1.61 Boolean Logic Circuit Implementing Function 'Y'

Example $\boxed{Y = (A + B).(C + D)}$

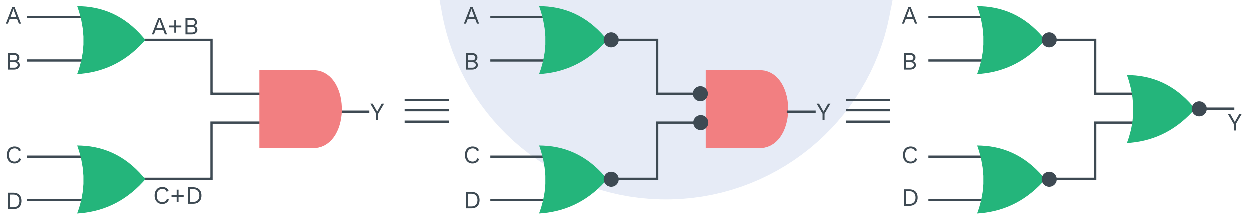


Fig. 1.62 Boolean Logic Circuit Implementing Function 'Y'

Note:

If the bubble is placed one after the other, it cancels each other's effect

SWITCH REPRESENTATION

Switch representation of the AND function:

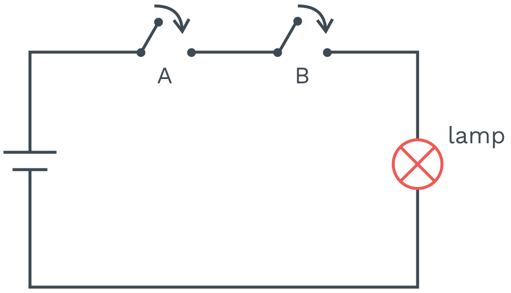


Fig. 1.63 Circuit Diagram Representing AND Function

Switch A – open = "0" closed = "1"

Switch B – open = "0" closed = "1"

Both switches must be ON for the lamp to be ON, if A = high, B = high, Y = high.

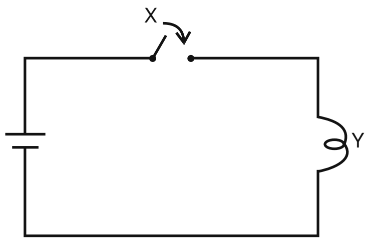


Fig. 1.64 Circuit Diagram Representing Buffer

Y = X, switch X is ON, lamp turns ON (Buffer)

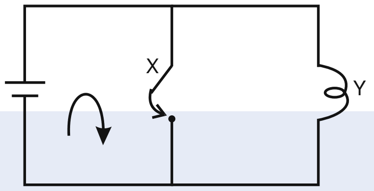
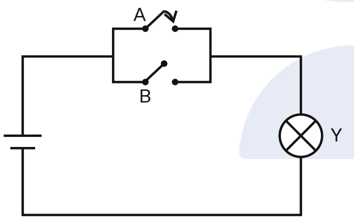


Fig. 1.65 Circuit Diagram Representing NOT Function

Switch X is ON, lamp is OFF

$Y = \overline{X}$ (Inverter)



If A is ON, lamp ON
if B is ON, lamp ON

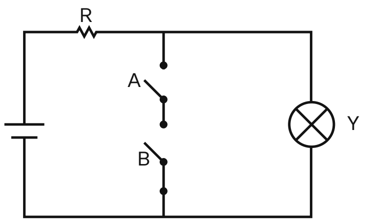
Y = A + B (OR)

Fig. 1.66 Circuit Diagram Representing OR Function

Rack Your Brain

A bulb that can be turned on by 2 switches (2-way switch) implements which gate?

Switch representation of the NAND function:



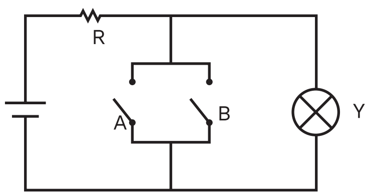
Lamp ON = "1"
Lamp OFF = "0"

Fig. 1.67 Circuit Diagram Representing NAND Function

If A & B are high, then the bulb gets shorted, lamp = OFF

$\boxed{Y = \overline{AB}}$

Switch representation of the NOR function:



If A = ON, lamp = OFF
B = ON, lamp = OFF

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

Fig. 1.68 Circuit Diagram Representing NOR Function

Switch representation of XOR function:

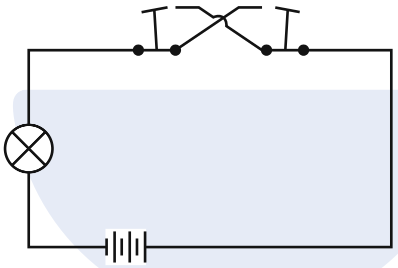


Fig. 1.69 Circuit Diagram Representing XOR Function

If both are up or both are down, the current can not flow
If one is up and another down, then current can flow.
∴ Current flows for unequal i/p
⇒ XOR gate

Switch representation of the XNOR function:

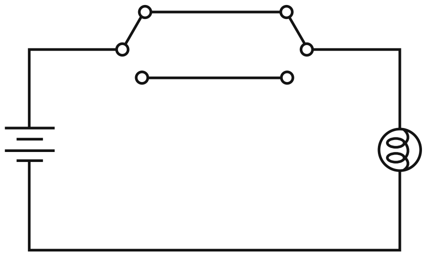


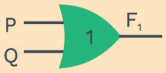
Fig. 1.70 Circuit Diagram Representing XNOR Function

If both switches are up, current flows
If both switches are down, current flows
∴ equality detector : XNOR gate

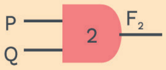
SOLVED EXAMPLES

Q11 **A universal gate can implement any Boolean function by connecting the sufficient number of them appropriately.**

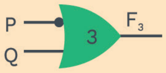
$F_1 = P + Q$



$F_2 = P \cdot Q$



$F_3 = \overline{P} + Q$



Which of the following is true?

1) Gate 1 is universal gate

2) Gate 2 is the universal gate

3) Gate 3 is universal gate

4) None

Sol: We already know, NOR and NAND universal gates.

$F_3 = \overline{P} + Q$

if $Q = 0$

$F_3 = \overline{P}$ [complement is generated]

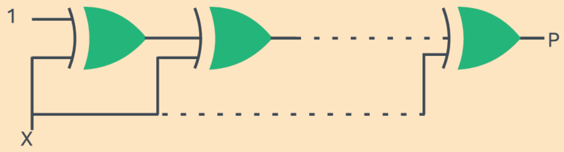
$F_3(P, Q) = \overline{P} + Q$ [given]

$F_3(\overline{P}, Q) = P + Q$ [OR is generated]

"OR" followed by complement is NOR, which is universal gate.

$\boxed{\text{Ans.} - \text{Option}(3)}$

Q12 **Below digital circuit consists of 20 XOR gates in cascade.**



output P is equal to:

1) 0 **2) 1**

3) X **4) $\overline{X}$**

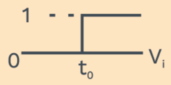
Sol: $X \oplus 1 = \overline{X}$

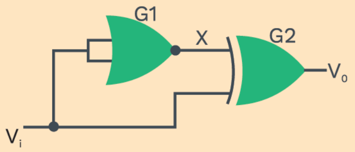
$\overline{X} \oplus X = 1$

After 2 XOR Gate, o/p is 1, So after 20 XOR gates also, the output is going to be 1.

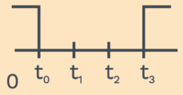
$\boxed{P = 1}$ $\boxed{\text{Ans.} - (2)}$

Q13 **The gates G_1 and G_2 have propagation delay 10 nsec and 20 nsec respectively. If input V_i makes an abrupt change from logic 0 to 1 at $t = t_0$ then o/p wave form V_0 is:**

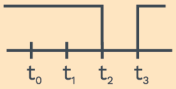




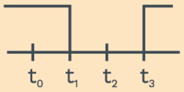
1)



2)

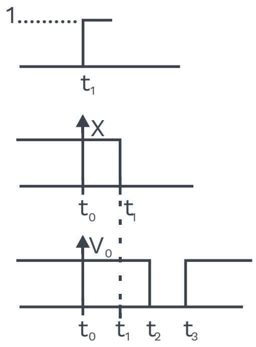


3)



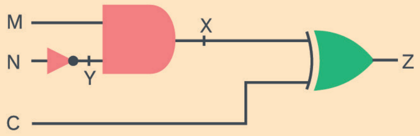
4) None

Sol:

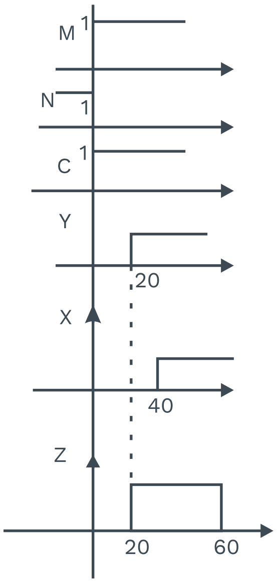


option 2)

Q14 **All logic gates shown in the figure below have a propagation delay of 20 ns. Let M = C = 0 and N = 1 until t = 0. At t = 0, all the inputs flip and remain in that state. For t > 0, output is high (1) for a duration (in ns) of ________.**



Sol:



Ans. = 40

Chapter Summary

- AND - $A.B$
  OR - $A + B$
  NOT - $\bar{A}$
  NAND - $\overline{A.B}$
  NOR - $\overline{A + B}$
  XOR - $A\bar{B} + B\bar{A}$
  XNOR - $AB + \bar{A}\bar{B}$
- Associative law for XOR – $(A \oplus B) \oplus C = A \oplus (B \oplus C)$
  Commutative law for XOR – $A \oplus B = B \oplus A$
- AND law : $A.0 = 0$
  $A.1 = A$
  OR law : $A + 0 = A$
  $A + 1 = 1$
- Inversion law : $\bar{\bar{A}} = A$
- $A \oplus B \oplus C = A \odot B \odot C$
  $A \oplus B = \overline{A \odot B}$
- Demorgan's law:
  i) $\overline{A + B} = \bar{A}.\bar{B}$
  ii) $\overline{A.B} = \bar{A} + \bar{B}$
- Transposition law : $AB + A'C = (A + C)(A' + B)$
- Consensus theorem : $A'B + AC + BC = A'B + AC$
- Duality theorem: $A + BC \xrightarrow{\text{dual}} A(B + C)$
- Complementary theorem: $A(B+C) \xrightarrow{\text{comp.}} A' + (B'.C')$
- Standard SOP : each product term need not contain all literals.
- Canonical SOP : each product term must contain all literals.
- Minterm : Product of all variables where each variable appears only once either in true of complemented form
- Maxterm : Sum of variables, in which all variables appear once either in true or complemented form.
- Minimization of Boolean Expression through :
  K-map (2, 3, 4 variable maps)
- NAND and NOR are universal gates.
- Don't care conditions: It represents class of distinct functions.