Date ___/___/______

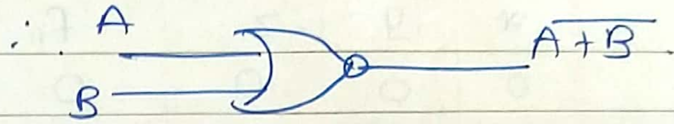
$\Rightarrow [\bar{A} + \bar{A}\cdot\bar{B}]\cdot[\bar{B} + \bar{B}\cdot\bar{C}]$

$\Rightarrow [\bar{A}\cdot(\bar{B}+1)]\cdot[\bar{B}\cdot(\bar{C}+1)]$

$\Rightarrow \bar{A}\cdot\bar{B}$

$\Rightarrow \overline{A+B}$

NOR gate.



* Minterms & Max terms :-

① Binary variable can take value of zero or 1.

② A boolen function is an expression formed with binary variables two binary operators (OR and AND), unary operator NOT, paranthesis and equal sign.
for given value of variable function can be either zero or 1.

eg. $f_1 = xyz'$
$f_2 = (xyz' + yz')$

			Minterms term	Designation	Maxterms	
0	0	0	$x'y'z'$ 000	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$ 001	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$ 010	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$ 011	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$ 100	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$ 101	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

* Conversion from 1 canonical / standard form to another canonical form. (SOP → POS) OR (POS → SOP)
Inter change the symbols [Σ to Π & Π to Σ] and list those no.s missing from the orginal form.

eg

① $f(x, y, z) = \Pi(0, 2, 4, 3)$
$= \Sigma(1, 3, 6, 7)$

② $f(w, x, y, z) = \Sigma m(1, 7)$
$= \Pi M(0, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16)$

Q. I Express boolen function $f = A + B'C$ in a sum of minterms.

→

$A = A(B + B') \cdot (C + C')$
$= ABC + ABC' + AB'C + AB'C'$

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

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

$\Rightarrow ABC + ABC' + AB'C + AB'C' + A'B'C$
$\Rightarrow m_7 + m_6 + m_5 + m_4 + m_1$
$= \Sigma m(7, 6, 5, 4, 1)$

$(AB+CD)$

$(AB+C)(AB+D)$

Date ___/___/______

Q2 Express $f = xy + x'z$ in a product of maxterms.

$\rightarrow$ $f = xy + x'z$.

first convert function into OR terms using ~~xy~~ distributive law.

$f = xy + x'z$ $\qquad$ $xy + xyz + 0 + x'z$

$= (xy + x')(xy + z)$

$= (x + x')(x' + y)(x + z)(y + z)$ $\qquad$ $xy + xyz + x'z$

$= 1 \; (x' + y)(x + z)(y + z)$ $\qquad$ $xy + x'z$

$= \cancel{x'}$

xzz' $\qquad$ $(z + z')$

$x' + y = (x' + y) + zz'$

$= \cancel{x'zz' +} (x' + y + z)(x' + y + z')$

$\Rightarrow M_4 \cdot M_5$.

$\overline{BC}$

$y + x$

$x + z = (x + z) + yy'$

$= (x + z + y)(x + z + y')$

$= M_0 \; M_2$

BC

$y + z = (y + z) + \cancel{x}x'$

$= (x + y + z)(x' + y + z)$

$= M_0 \cdot M_4$.

$f = (x'y)(x + z)(y + z) \Rightarrow M_0 \; M_2 \; M_4 \; M_5$.

$\Rightarrow \prod_M (0, 2, 4, 5)$.

Page No.

Date __/__/__

* Karnaugh Map (K-map)

(simplification of Boolen function by K-Map)

- The map method provides a simple straight forward procedure for minimizing Boolen funn It is diagram made up of square. Each square represent 1 minterms. since any boolen function can be expressed as sum of minterms be follows that a boolen funct is recognised graphically in map from area enclosed by those square whose minterms are in boolen in function.
- In fact, the map present a visiual diagram of all possible
- A function may be expressed in std form
- The by recognising various patterns user can derived alternative algebraic expression for the same function, from which he/she can select the simpleast function.
- We shall assume that simpleast alfebraic expression is anyone in a sum of product or product of sum ie it has minimum no. of literals.
- This expression is not necessisarly unique.

Page No.

TWO

minterms

	0	1
0	$x'y'$ 0	$x'y$ 1
1	xy' 2	xy 3

maxterms

	0	1
0	xy 0	xy' 1
1	$x'y$ 2	xy 3

* Three variable Map :

minterms

x \ yz	y'z' 00	y'z 01	yz 11	yz' 10
x' 0	x'y'z' (0)	x'y'z (1)	x'yz (3)	x'yz' (2)
x 1	xy'z' (4)	xy'z (5)	xyz (7)	xyz' (6)

maxterms

x \ y+z	y+z 0+0	y+z' 0+1	y'+z' 1+1	y'+z 1+0
0 x	x+y+z (0)	x+y+z' (1)	x+y'+z' (3)	x+y'+z (2)
1 x'	x'+y+z (4)	x'+y+z' (5)	x'+y'+z' (7)	x'+y'+z (6)

Q.1 simplify boolen function by k-map.

$f = x'yz + x'yz' + xy'z' + xy'z$

$= m3 + m2 + m4 + m5$

$= \Sigma(3,2,4,5)$.

x \ yz	y'z' 00	y'z 01	yz 11	yz' 10
x' 0	(0)	(1)	1 (3)	1 (2)
x 1	1 (4)	1 (5)	(7)	(6)

→ x'y

xy'

$\Rightarrow xy' + x'z$

Q.2 $f = x'yz + xy'z' + xyz + xyz'$.

011 100 111 110

$= m3 + m4 +$

x \ yz	y'z' 00	y'z 01	yz 11	yz' 10
x' 0	(0)	(1)	1 (3)	(2)
x 1	1 (4)	(5)	1 (7)	1 (6)

for 3 & 2 x'y

→ xy

yz

for 4 & 6 xz'

Date ___/___/______

* Four variable map :

minterms

	$w'z'$ 00	$w'z$ 01	wz 11	wz' 10
$x'y'$ 00	$x'y'z'w'z'$ 0	$x'y'w'z$ 1	$x'y'wz$ 3	$x'y'wz'$ 2
$x'y$ 01	$x'yw'z'$ 4	$x'yw'z$ 5	$x'ywz$ 7	$x'ywz'$ 6
xy 11	$xyw'z'$ 12	$xyw'z$ 13	$xywz$ 15	$xywz'$ 14
xy' 10	$xy'w'z'$ 8	$xy'w'z$ 9	$xy'wz$ 11	$xy'wz'$ 10

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

maxterms

	wz 00	wz' 01	$w'z'$ 11	$w'z$ 10
xy 00	$x+y+z+w$ 0	$x+y+w+z'$ 1	$x+y+w'+z'$ 3	$x+y+w'+z$ 2
xy' 01	$x+y'+w+z$ 4	$x+y'+w+z'$ 5	$x+y'+w'+z'$ 7	$x+y'+w'+z$ 6
$x'y'$ 11	$x'+y'+w+z$ 12	$x'+y'+w+z'$ 13	$x'+y'+w'+z'$ 15	$x'+y'+w'+z$ 14
$x'y$ 10	$x'+y+w+z$ 8	$x'+y+w+z'$ 9	$x'+y+w'+z'$ 11	$x'+y+w'+z$ 10

Q-1 Simplify given boolen function by K map.

$F(A, B, C, D) = \Sigma m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

	$z'w'$ 00	$z'w$ 01	zw 11	zw' 10
$x'y'$ 00	1 (0)	1 (1)	(3)	1 (2)
$x'y$ 01	1 (4)	1 (5)	(7)	1 (6)
xy 11	1 (12)	1 (13)	(15)	1 (14)
xy' 10	1 (8)	1 (9)	(11)	(10)

$\rightarrow x'w'$

$\rightarrow yw'$

$\rightarrow z'$

$F = z' + x'w' + yw'.$

$F = A\bar{C} + \bar{B}\bar{D} + \bar{A}D$ —— SOP form

$= (A+B+D)(\bar{A}+\bar{C}+\bar{D})(\bar{A}+\bar{B}+\bar{C})$

$\Rightarrow (A+B+D)\cdot(A+A')$

$\Rightarrow AC+BC+DC+C'A+\bar{C}B+\bar{C}D$

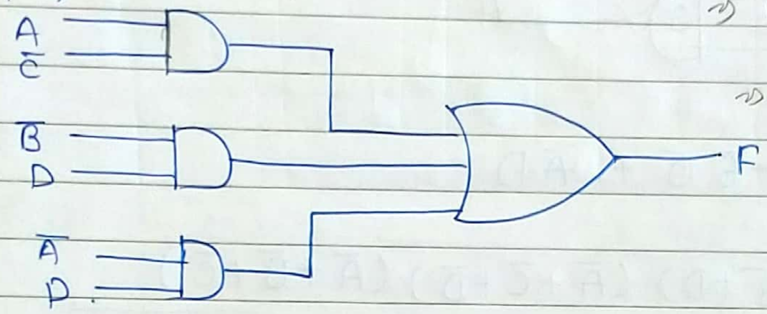
$\Rightarrow C(A+B+D)+C'(A+B+D)$

$\Rightarrow$

* Implementation

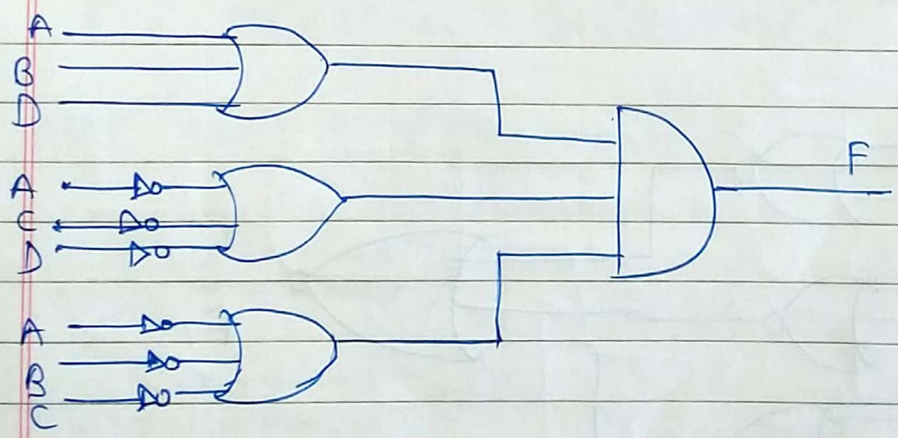
AND, OR, Inverter Logic.

SOP ⇒



POS ⇒

$f = (A+B+D)(\bar{A}+\bar{C}+\bar{D})(\bar{A}+\bar{B}+\bar{C})$



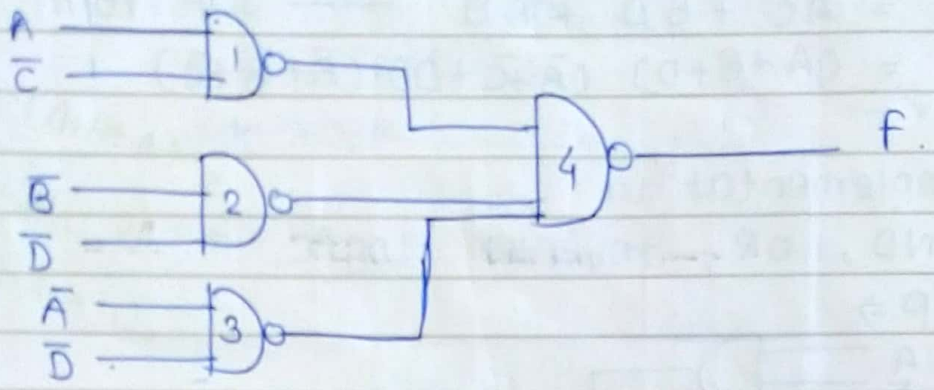
*. Implementation by using universal logic.

$f = A\bar{C} + \bar{B}\bar{D} + \bar{A}D$

$f' = \overline{A\bar{C} + \bar{B}\bar{D} + \bar{A}D}$

$(f') = (\overline{A\bar{C}})\cdot(\overline{\bar{B}\bar{D}})\cdot(\overline{\bar{A}D})$

$(f')' = \overline{(\overline{A\bar{C}})\cdot(\overline{\bar{B}\bar{D}})\cdot(\overline{\bar{A}D})}$



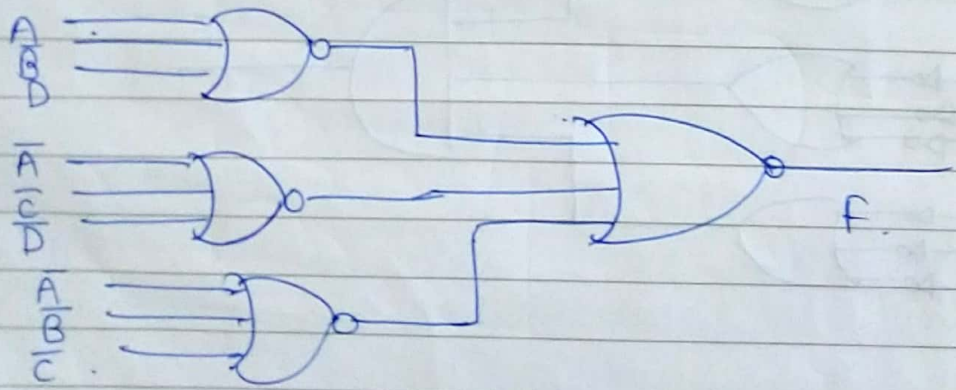
$$F = A\bar{C} + \bar{B}\,\bar{D} + \bar{A}D$$

$$= (A+\bar{B}+D)\,(\bar{A}+\bar{C}+\bar{D})\,(\bar{A}+\bar{B}+\bar{C})$$

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

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

$=$



Q. Reduce expression using K-map
$\pi(2,8,9,10,11,12,14)$ and implement it in AOI logic, NAND, NOR logic.

→

POS

maxterms :

	$w+z$ 00	$w+z'$ 01	$w'+z'$ 11	$w'+z$ 10
$x+y$ 00	0	1	3	0 (2)
$x+y'$ 01	4	5	7	6
$x'+y'$ 11	0 (12)	13	15	0 (14)
$x'+y$ 10	0 (8)	0 (9)	0 (11)	0 (10)

$(w'+z+y)$

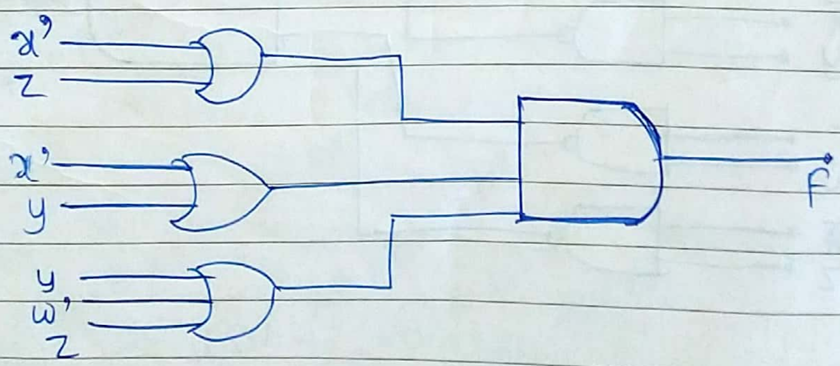
$x'+z$

$x'+y$

$f = (x'+z)\cdot(x'+y)\,(w'+z+y)$

$f = (x'+z)\cdot(x'+y)\cdot(y+w'+z)$

AOI Logic : (maxterms)



POS form

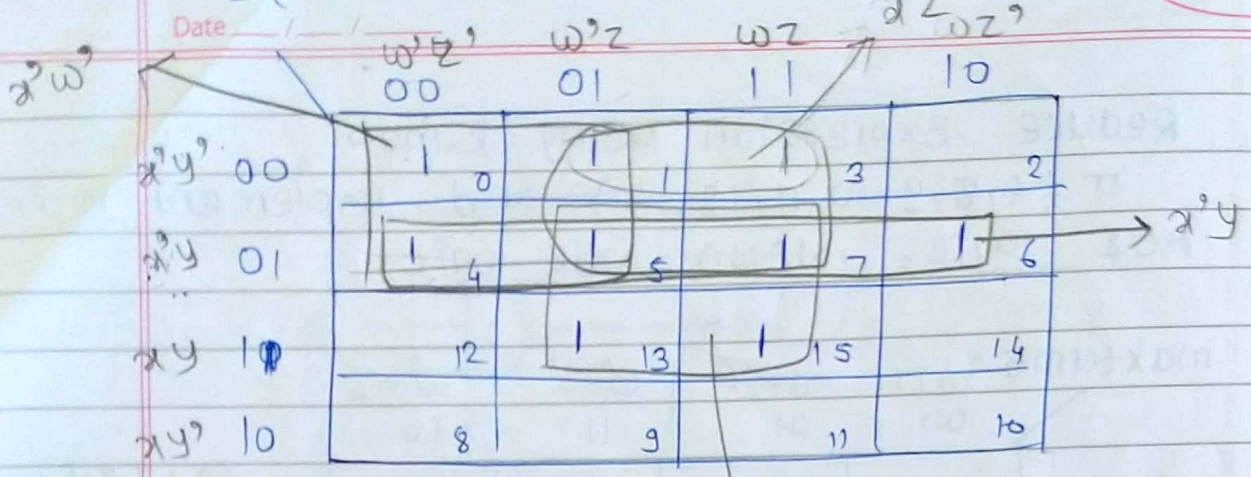
$f = (x'+z)\cdot(x'+y)\cdot(y+w'+z)$

$=$

$(x'+z) \Rightarrow x'+z+yy'+ww'$

$\Rightarrow$

$\Sigma(0,1,3,4,5,6,7,13,15)$



$f = yz + x'y + x'w' + x'z$

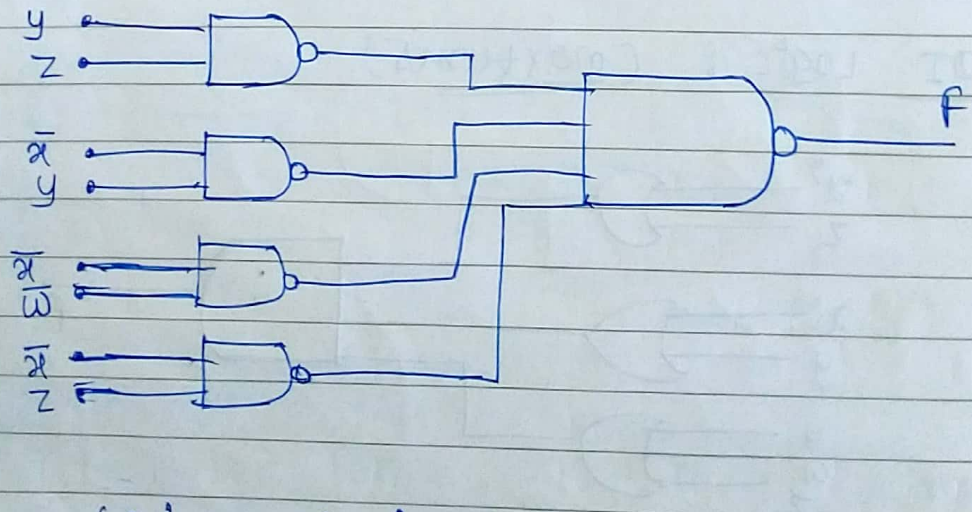
$f = yz + x'y + x'w' + x'z$

$f' = \overline{yz + x'y + x'w' + x'z}$

$\Rightarrow \overline{yz} \cdot \overline{\bar{x}y} \cdot \overline{\bar{x}\bar{w}} \cdot \overline{\bar{x}z}$

$(f')' \Rightarrow \overline{\overline{yz} \cdot \overline{\bar{x}y} \cdot \overline{\bar{x}\bar{w}} \cdot \overline{\bar{x}z}}$

* NAND Implementation



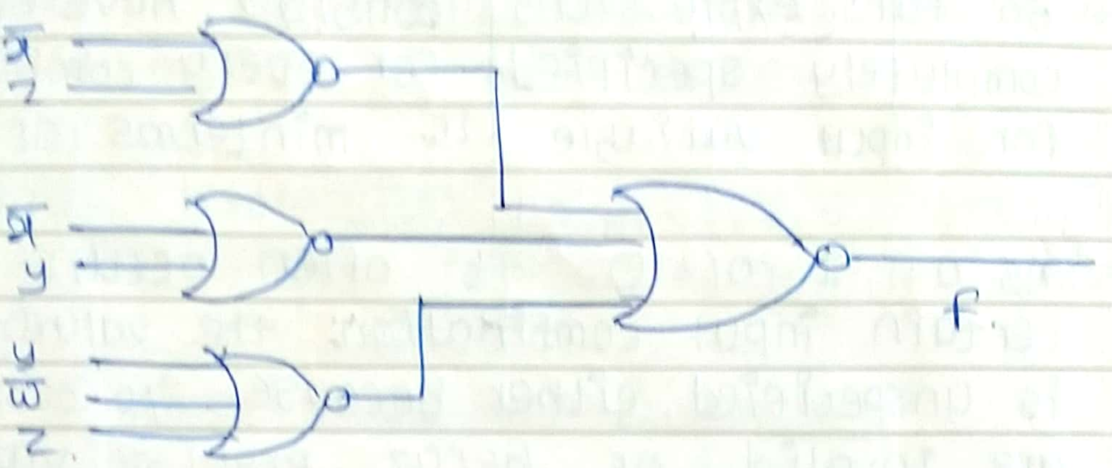
$f = (x'+z) \cdot (x'+y) \cdot (y+w'+z)$

$f' = \overline{(x'+z) \cdot (x'+y) \cdot (y+w'+z)}$

$= \overline{(x'+z)} + \overline{(x'+y)} + \overline{(y+w'+z)}$

$(f')' \Rightarrow \overline{\overline{(\bar{x}+z)} + \overline{(\bar{x}+y)} + \overline{(y+\bar{w}+z)}}$

* NOR implementation:



$A+B+\bar{A}\bar{B}C+C\bar{C}$

$A+B+C(\bar{A}\bar{B}+\bar{C})$

$A\bar{A}+B\bar{B}+A+B+\bar{A}\bar{B}C$

$AA \quad AA$

$A+B+\bar{A}\bar{B}C+AB\cdot\bar{A}\bar{A}$

$\overline{(\bar{A})}+\overline{(\bar{B})} \quad AA+BB+\bar{A}\bar{B}C$

$(A+B)+(\bar{A}\bar{B}C)$

$(A+B+\bar{A}\bar{B})\cdot(A+B+C)$

(

*. Don't care combinations

- So far expressions consider have been completely specified for every combinations for input variable ie minterms or maxterms

- As a 1 or 0 it often occurs that for certain input combination the value of output is unspecified either because i/p combinations are invalid or becaz precise value of o/p is of no consequence.
- The combinations for which values of expression are not specified are called don't care combinations & such expression ie therefore stand incompletely specified.
- The o/p is don't care for this invalid combination

eg

in excess-3 code system binary states are unspecified and never occour.
These are called don't cares
similarly in 8421 code, binary sets from 10 to 15 are invalid or and corresponding o/p are don't care.
- - Don't care terms are denoted by X or ϕ.
- - During process of design using SOP map each don't care is treated as '1' if it is helpful in map reduction otherwise it is treated as zero '0' & left alone.

maxterms = $(B+D)(\bar{A}+B)(\bar{C}+\bar{D})$
min = $B\bar{C} + \bar{B}D + \bar{A}\bar{C}D$

Date ___/___/___

– During process of design using pos form map each don't care is treated as zero if it is useful in map reduction, otherwise it is treated as 1 & left alone.

Q.1 Reduce expression $\Sigma m(1,5,6,12,13,14)$ + $d(2,4)$. and implement it in universal logic.

minterms	C'D' 00	C'D 01	11	10
00	0	1 1	3	X 2
A'B 01	X 4	1 5	7	1 6
AB 11	1 12	1 13	15	1 14
10	8	9	11	10

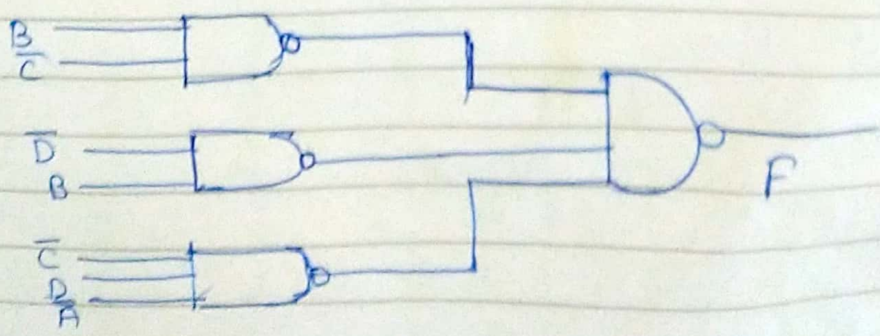
$F = BC' + D'B + C'DA'$

$F' = \overline{BC' + D'B + C'DA'}$

$(F')' = \overline{(B\bar{C})} \cdot \overline{(\bar{D}B)} \cdot \overline{(\bar{C}D\bar{A})}$

$(f')' = \overline{\overline{B\bar{C}} \cdot \overline{\bar{D}B} \cdot \overline{\bar{C}D\bar{A}}}$

using NAND gates



Page No.

$\Sigma m(3,4,7,9,10,11) + d(0,2,13,14,15)$

minterms

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	x		1	x
$\bar{A}B$	1		1	
AB		x	x	x
$A\bar{B}$		1	1	1

$\bar{A}\bar{C}\bar{D}$ AC AD CD

$f = AD + CD + AC + \bar{A}\bar{C}\bar{D}$

maxterms

	$C+D$	$C+\bar{D}$	$\bar{C}+\bar{D}$	$\bar{C}+D$
$A+B$	x	x		x
$A+\bar{B}$		0		0
$\bar{A}+\bar{B}$	0	x	x	x
$\bar{A}+B$	0			

$\bar{B}+\bar{C}+D$

$\bar{A}+C+D$

$\bar{B}+C+\bar{D}$

$\Pi_M = (1,5,6,8,12) + d(0,2,13,14,15)$

Date __/__/____

To clip - Remove
To clamp - shift.

Clipper :

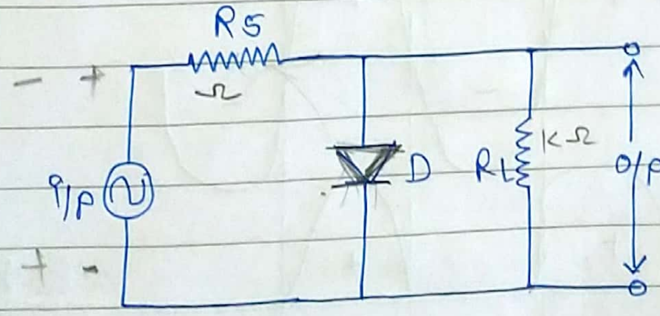
clipper - waveform is shaped by removing some part of input signal is called as clipper.

Types of clipper circutary :

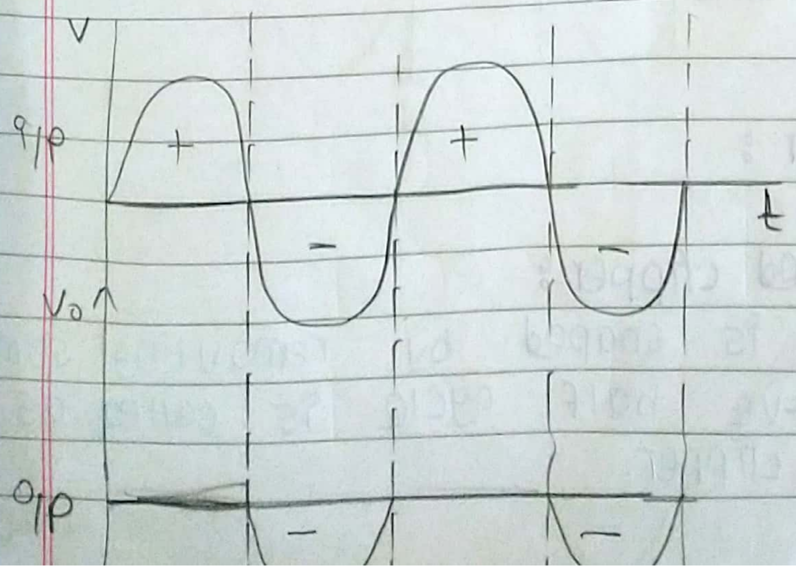
1) positive clipper.
2) negative clipper.
3) ~~combinational clipper.~~
3) Biased clipper
   - positive biased
   - Negative biased
   - combinational clipper.

1) positive clipper :

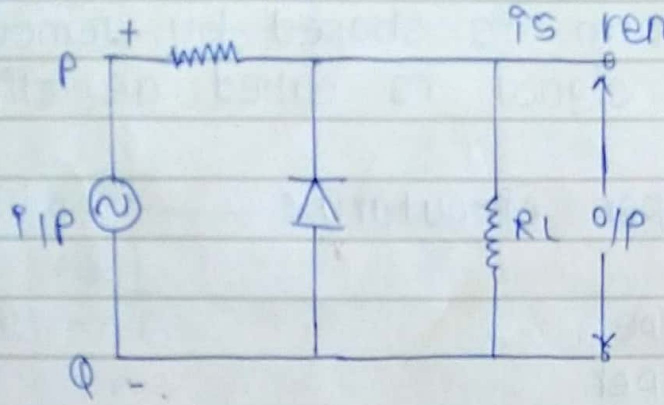
In this circuitary positive half cycle is removed.



Diode
Forward bias
close switch { +ve ⇒ Anode
−ve ⇒ cathode
Reverse bias
+ve ⇒ cathode
−ve ⇒ Anode.

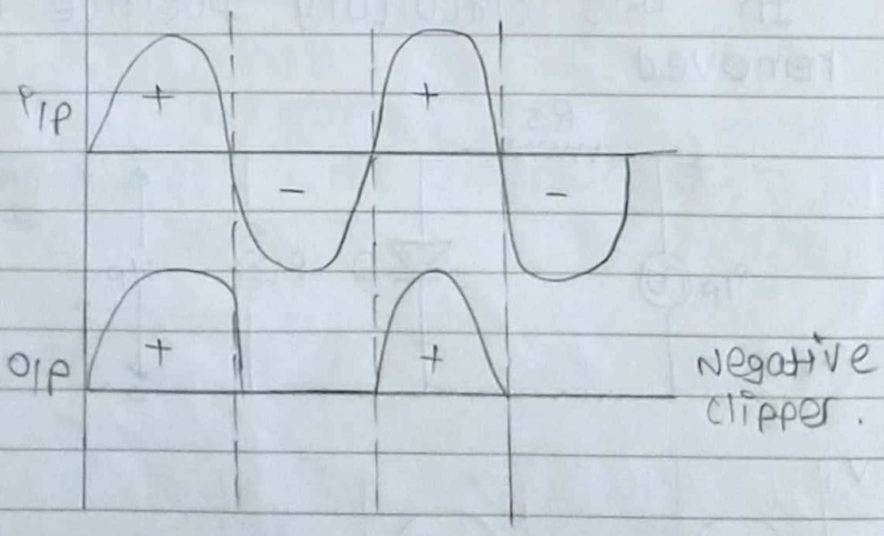


* **Negative clipper :** In this negative half cycle is removed.



1) During +ve half cycle p is +ve and q is -ve so diode is goes in reverse bias so it acts as ~~reverse~~ open switch.
so we get o/p at RL during +ve cycle
2) During -ve half cycle p is -ve & q is +ve diode is in forward bias so acts as close switch.

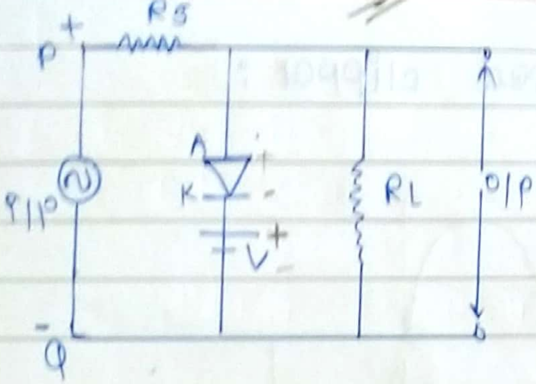
waveform :



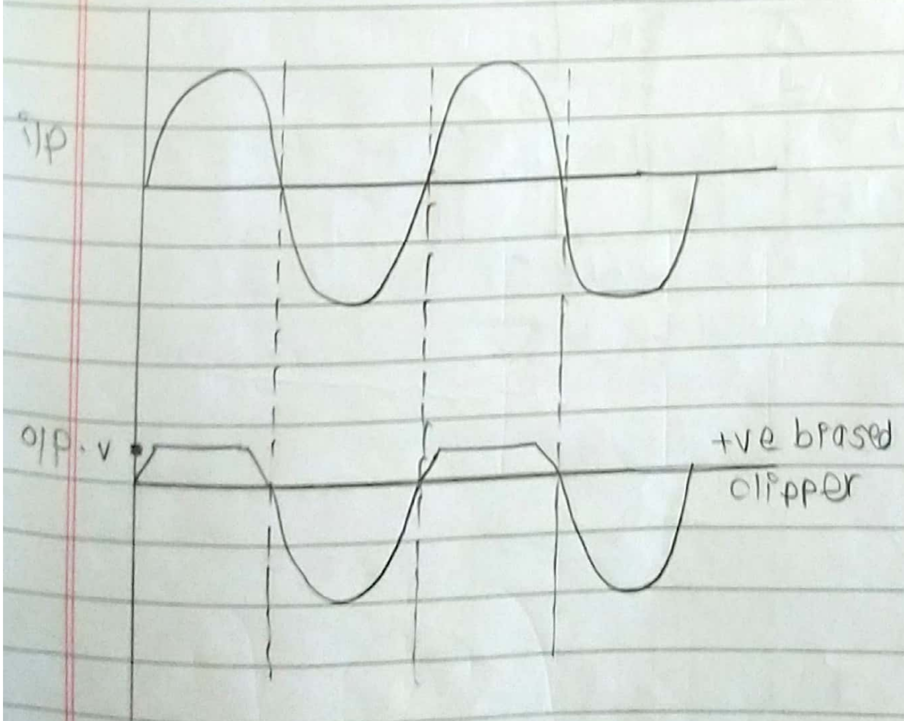
* **Biased clipper :**

1) **Positive biased clipper :**
waveform is shaped by removing some part of +ve half cycle is called as +ve biased clipper.

If battery is  then it acts as positive clipper

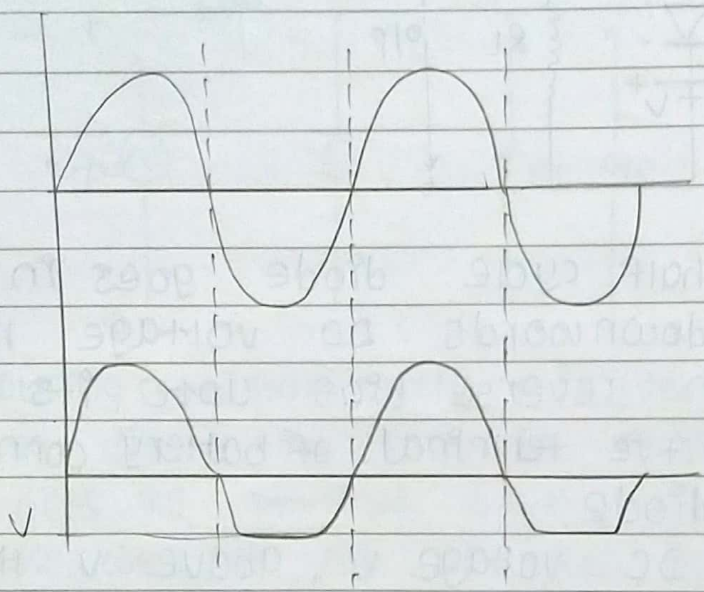


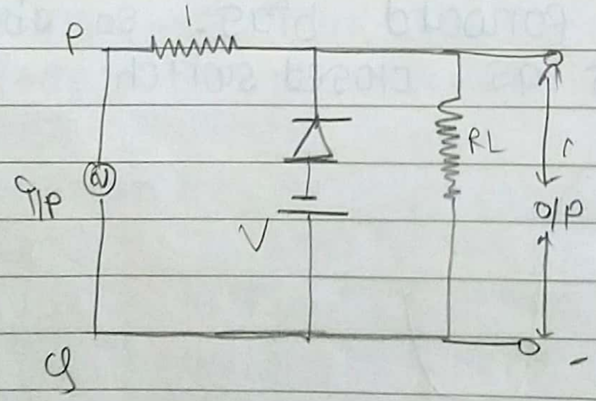
- During +ve half cycle diode goes in forward bias. but downwords DC voltage revered (V) the diode in reverse bias upto its voltage (offset open switch) th because +ve terminal of battery connected to cathode of diode.
After this DC voltage V, above V this diode goes in forward bias. so above voltage V then it acts as closed switch.



Date ___/___/___

2) Negative biased clipper :





* ~~Clampper circuit~~ :

~~To clamp~~ ⇒ ~~To shift~~.

* Steps of Designing.

1) Identify no. of inputs and no. of outputs
2) prepare truth table
3) obtain boolean expression.
4) Reduce expression | boolen function by K-map if necessary
5) Implement reduced boolean function or expression by using logic gates according to required logic. (AOI, NAND, NOR].

Q.1 Construct hardware circuit using only NAND gates for

Y = F (P, Q, R) where Y is true (high | 1), if, two or more inputs of P | Q | R are true (high | 1)

→ 1) 3 inputs & 1 o/p.

2)

P	Q	R	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	$\boxed{1}$
1	0	0	0
1	0	1	$\boxed{1}$
1	1	0	$\boxed{1}$
1	1	1	$\boxed{1}$

3) obtain boolean function by truth table

$$f = P'QR + PQ'R + PQR' + PQR.$$

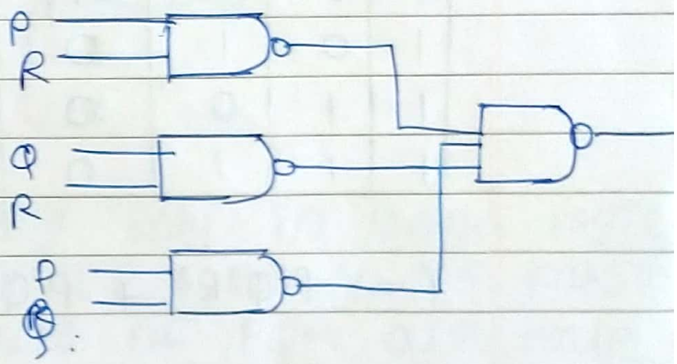
4) Reduce above function by K-map.

P \ QR	Q'R' 00	Q'R 01	QR 11	QR' 10
P' 0	0	1	1 (3)	2
P 1	4	1 (5)	1 (7)	1 (6)

→ PQ

PR QR

NAND Logic.



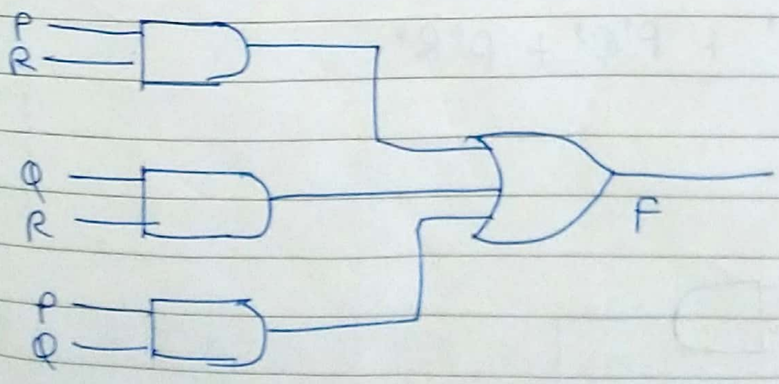
$$f = PR + QR + PQ$$

$$f' = \overline{PR + QR + PQ}$$

$$f' = \overline{PR} \cdot \overline{QR} \cdot \overline{PQ}$$

$$f = \overline{\overline{PR} \cdot \overline{QR} \cdot \overline{PQ}}$$

AOI Logic.



Q.2 construct hardware circuit using NAND logic as well as AOI logis for $Y = f(P, Q, R)$ where Y is true if two or more inputs of P, Q, R are false.

P	Q	R	Y
0	0	0	$\boxed{1}$
0	0	1	$\boxed{1}$
0	1	0	$\boxed{1}$
0	1	1	0
1	0	0	$\boxed{1}$
1	0	1	0
1	1	0	0
1	1	1	0

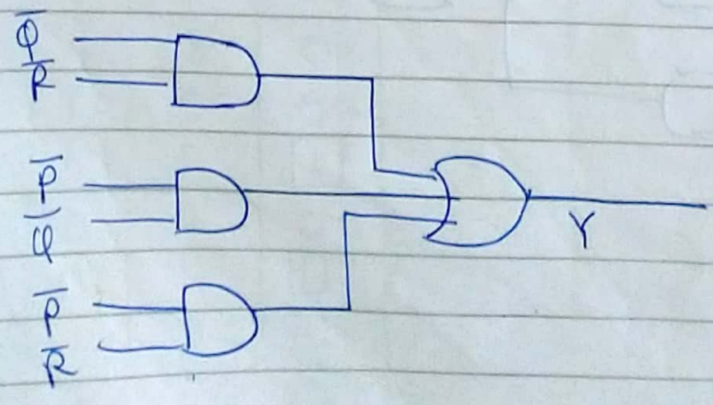
$$Y = P'Q'R' + P'Q'R + P'QR' + PQ'R'$$

	Q'R'	Q'R	QR	QR'
P' 0	1 (0)	1	(3)	1 (2)
P 1	1 (4)	(5)	(7)	0 (6)

$$f = Q'R' + P'Q' + P'R'$$

$$f = Q'R' + P'Q' + P'R'$$

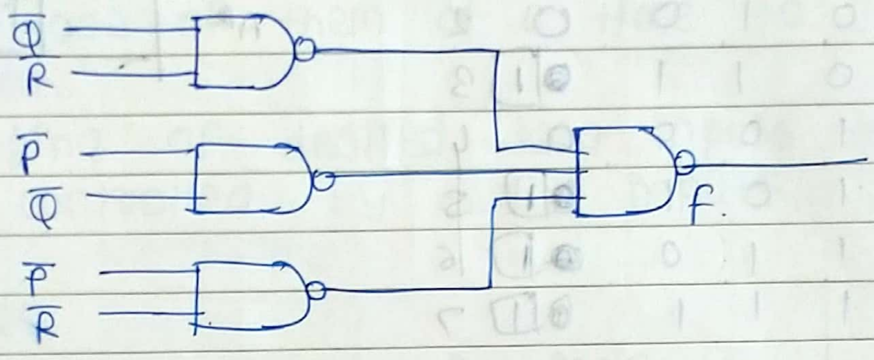
AOI logic



$f' = \overline{Q'R'} \cdot \overline{\bar{P}\bar{Q}} \cdot (\overline{\bar{P}\bar{R}})$

$f = \overline{\overline{\bar{Q}\bar{R}} \cdot \overline{\bar{P}\bar{Q}} \cdot (\overline{\bar{P}\bar{R}})}$

NAND implementation



Q.3 construct hardware using only in Nand logic & AOI logic for Y = f(P, Q, R) Y is false if two or more inputs of PQR are true.

→

P	Q	R	Y
0	0	0	1
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	0

$Y = P'Q'R' + P'Q'R + P'QR' + PQ'R'$

1 ₀	1 ₁	₃	1 ₂
1 ₄	₅	₇	₆

Q.3 Construct hardware ckt fn NAND & AOI logic
Y(P,Q,R) Y is false if two or more inputs
of P, Q, R are false

P	Q	R	Y	
0	0	0	0	m0
0	0	1	0	m1
0	1	0	0	2
0	1	1	$\boxed{1}$	3
1	0	0	0	4
1	0	1	$\boxed{1}$	5
1	1	0	$\boxed{1}$	6
1	1	1	$\boxed{1}$	7

	B'C'	B'C	BC	BC'
A'			1	
A		1	1	1