Logic Gates

NOT Gate (Inverter)

when the input is Low, the output is HIGH, when the input is HIGH, the o/p is Low. Thereby producing an inverted output pulse.

Truth Table

I/P (x)	o/p (x^1)
0	1
1	0

2. AND Gate

For a 2-input AND gate, output X is HIGH only when inputs A and B ~~Low~~ are HIGH, X is Low when either A or B is Low, or when both A and B are Low.

Truth Table

A	B	Y = A·B = AB
0	0	0
0	1	0
1	0	0
1	1	1

A	B	C	Y
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

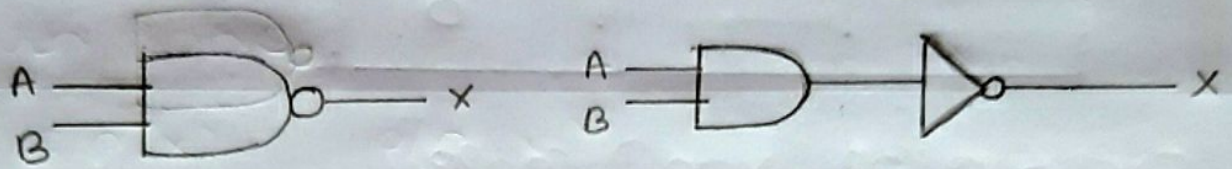
OR Gate

For a 2-input OR gate, output X is HIGH when either input A or input B is HIGH. or when both A and B are HIGH : X is LOW only when both A & B are LOW.

Truth table

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

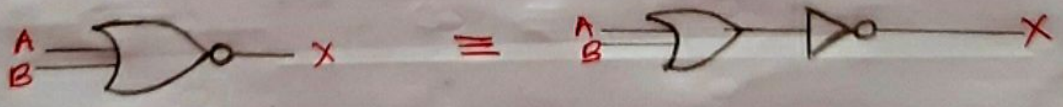
NAND Gate



For a 2-input NAND gate, output X is Low, only when inputs A and B are HIGH, X is HIGH whe either A or B is Low, or when both A and B are Low.

A	B	AB	$X = \overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NOR Gate



For a 2-input NOR gate, output X is Low when either input A or input B is HIGH. or when both A and B are HIGH. X is HIGH only when

both A and B are LOW.

$X = \overline{A+B}$

Truth table

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

Exclusive OR (XOR) Gate

For an exclusive-OR gate, output X is HIGH, when input A is LOW, and input B is HIGH, or when input A is HIGH, and input B is LOW when A & B are both HIGH or both LOW

Truth table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive NOR (XNOR) Gate

For an exclusive-NOR gate, output X is LOW when input A is LOW and input is HIGH, or when A is HIGH and B is LOW, X is HIGH when A and B are both HIGH or both LOW.

Truth table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Laws of boolean algebra

Basic Laws of bodean algebra:

1. commotative laws for addition and moltiplication.
2. Associative claws for addition and moltiplication
3. Distraibotive Law.

* commutative Laws

The commotative claw of addition for two variables is written as,

$A+B = B+A$

The commutative law of multiplication for two variables is,

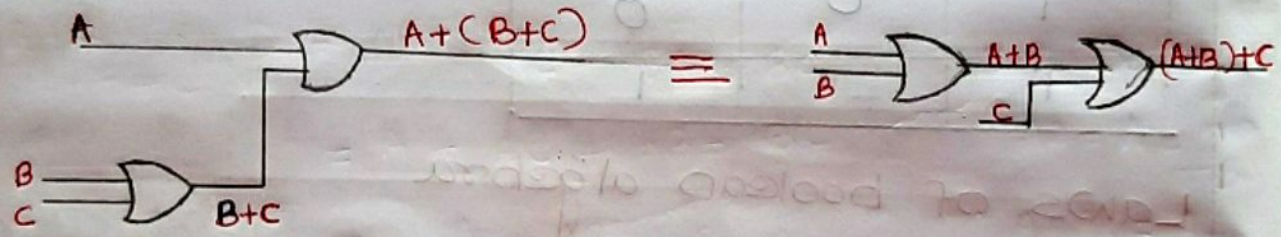
$$AB = BA$$

The law states that the order in which the variables are ANDed makes no difference.

Associative Laws

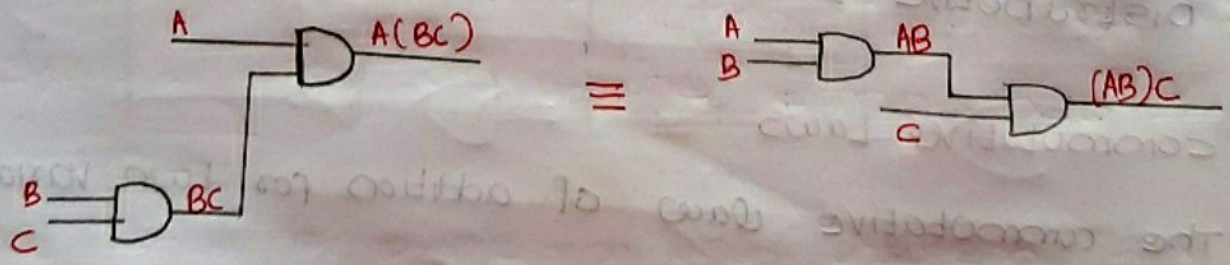
The associative law of addition is written as follows of three variables;

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



The associative law of multiplication is written as follows for three variables.

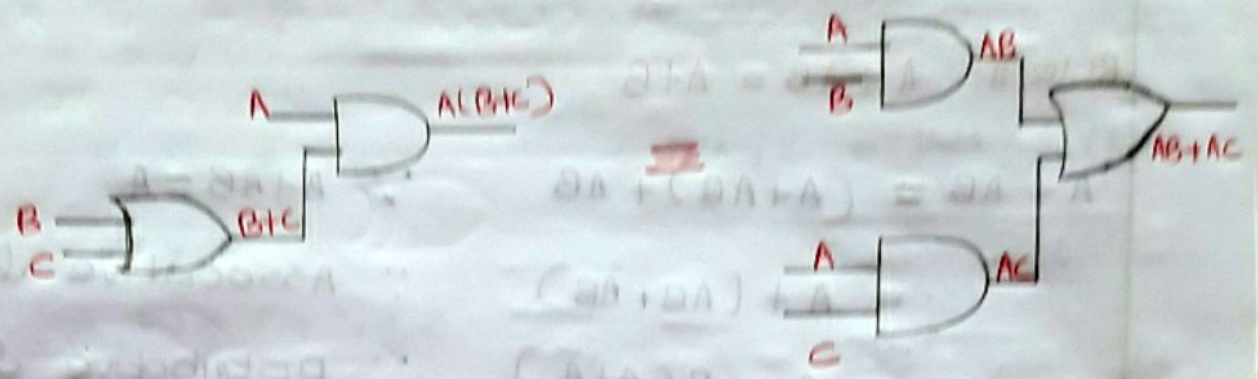
$$A(BC) = (AB)C$$



Distributive law

The distributive law written in three variable is follows,

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



Rules of Boolean Algebra

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \bar{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \bar{A} = 0$
9. $\bar{\bar{A}} = A$
10. $A + AB = A$
11. $A + \bar{A}B = A + B$
12. $(A+B)(A+C) = A + BC$

Rule 1: $A + 0 = A$

Proof,

A	0	A+0
0	0	0
1	0	1

$= A$.

Rule 10: $A + AB = A$

proof,

$A + AB = A(1+B)$ by distributive law

$= A \cdot 1$ by $\because 1+B = 1$

$= A$ $\because A \cdot 1 = A$

Rule 11: $A + \bar{A}B = A+B$

$A + \bar{A}B = (A+AB) + \bar{A}B$ $\because A+AB = A$

$= A + (AB + \bar{A}B)$ $\because$ Associative law

$= A + B(A+\bar{A})$ $\because$ Distributive law

$= A + (B \cdot 1)$ $\because A+\bar{A} = 1$

$= A + B$ $\because B \cdot 1 = B$

Rule 12. $(A+B)(A+C) = A+BC$

$(A+B)(A+C) = AA + AC + BA + BC$ $\because$ Distributive law

$= A + AC + AB + BC$ $\because AA = A$ & $AB = BA$

$= A + AB + BC$ $\because A+AC = A$

$= A+BC$ $\because A+AB = A$

Demorgan's Law

The complement of a product of variables is equal to the sum of the complements of the variables.

The formula for expressing this theorem for 2 variables is, $\overline{xy} = \bar{x} + \bar{y}$

* The complement of a sum of variables is equal to the product of the compliments of the variables.

The formula given by,

$$\overline{X+Y} = \bar{X}\bar{Y}$$

2. $\overline{XYZ} = \bar{X}+\bar{Y}+\bar{Z}$

2. $\overline{X+Y+Z} = \bar{X}\bar{Y}\bar{Z}$

2. $\overline{WXYZ} = \bar{W}+\bar{X}+\bar{Y}+\bar{Z}$

2. $\overline{W+X+Y+Z} = \bar{W}\bar{X}\bar{Y}\bar{Z}$

2. $\overline{ABC + DEF}$

2. $\overline{\bar{W}\bar{X}\bar{Y}\bar{Z}} = W+X+Y+Z$

Q. $\overline{(AB+C)(A+BC)}$

A). $\overline{(AB+C)(A+BC)} = \overline{(AB+C)} + \overline{(A+BC)}$

$= \overline{AB}\,\overline{C} + \overline{A}\,\overline{BC}$

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

Q. $\overline{ABC + DEF}$

A). $\overline{ABC + DEF} = \overline{ABC} \cdot \overline{DEF}$

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

Q. $\overline{A\overline{B} + \overline{C}D + EF}$

A). $\overline{A\overline{B} + \overline{C}D + EF} = \overline{A\overline{B}} \cdot \overline{\overline{C}D} \cdot \overline{EF}$

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

Q. $\overline{(A+B+C)D}$

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

$= \overline{A}\,\overline{B}\,\overline{C} + \overline{D}$

Q. $\overline{A + B\overline{C} + D(E+\overline{F})}$

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

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

Q. $\overline{(A+B)} + \overline{C}$

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

Q. $\overline{(\bar{A}+B)+CD}$

A) $\overline{(\bar{A}+B)+CD} = \overline{(\bar{A}+B)} \cdot \overline{CD}$

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

Q. $\overline{(A+B)\bar{C}\bar{D}+E+\bar{F}}$

A) $= \left(\overline{(A+B)\bar{C}\bar{D}}\right) \cdot \left(\overline{E+\bar{F}}\right)$

$= \left(\overline{(A+B)} + \overline{(\bar{C}\bar{D})}\right) \cdot (\bar{E}+F)$

$= \left[\bar{A}\bar{B} + (C+D)\right](\bar{E}F) = [\bar{A}\bar{B}+C+D]\bar{E}F$

simplification using boolean algebra.

Q. using boolean algebra simplify the following

1. $AB + A(B+C) + B(B+C)$

A) $AB + AB + AC + BB + BC$ ($\because$ distributive law)

$AB + AC + B + BC$ ($\because A+A=0, BB=B$)

$AB + AC + B$ ($\because A + AB = A$)

$BA + AC + B$ ($\because$ commutative law)

$B + BA + AC$ ($\because$ associative law)

$B + AC$ ($AB+B = B$)

2. $[AB'C\,C + BD) + A'B']C$

A) $[A\bar{B}C + A\bar{B}BD + \bar{A}\bar{B}]C$ [$\because$ distributive law]

$[A\bar{B}CC + A\bar{B}BDC + \bar{A}\bar{B}C$ [$\because$ "]

$A\bar{B}C + 0 + \bar{A}\bar{B}C$ [$C.C = C$ & $\bar{B}\cdot\bar{B} = D$]

$A\bar{B}C + \bar{A}\bar{B}C$ ($\because$ distributive law).

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

$\bar{B}C$ ($\because A+\bar{A}=1$)

3 $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$.

A. $\bar{A}BC + \bar{B}\bar{C}(A+\bar{A}) + A\bar{B}C + ABC$ ($\because$

$\bar{A}BC + ABC + \bar{B}\bar{C} + A\bar{B}C$ ($\because A+\bar{A}=1$)

$BC(A+\bar{A}) + \bar{B}\bar{C} + A\bar{B}C$ ($\because$ ") / $\because 1 \cdot BC = BC$

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

$BC + \bar{B}(AC+\bar{C})$

~~$BC + \bar{B}(BC + \bar{B}(AC+\bar{C})$~~

~~$(BC + \bar{B}(A\bar{C}$~~

$BC + \bar{B}(\bar{C}+AC)$

$BC + \bar{B}(\bar{C}+CA)$ $\quad \bar{C}+CA = \bar{C}+\bar{\bar{C}}A$

$(A+\bar{A}B) = A+AB$

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

Standard forms of Boolean expressions

1) SOP form
2) POS form

The - sum-of-products (SOP) form

When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products (SOP), some examples are,

$AB + ABC$

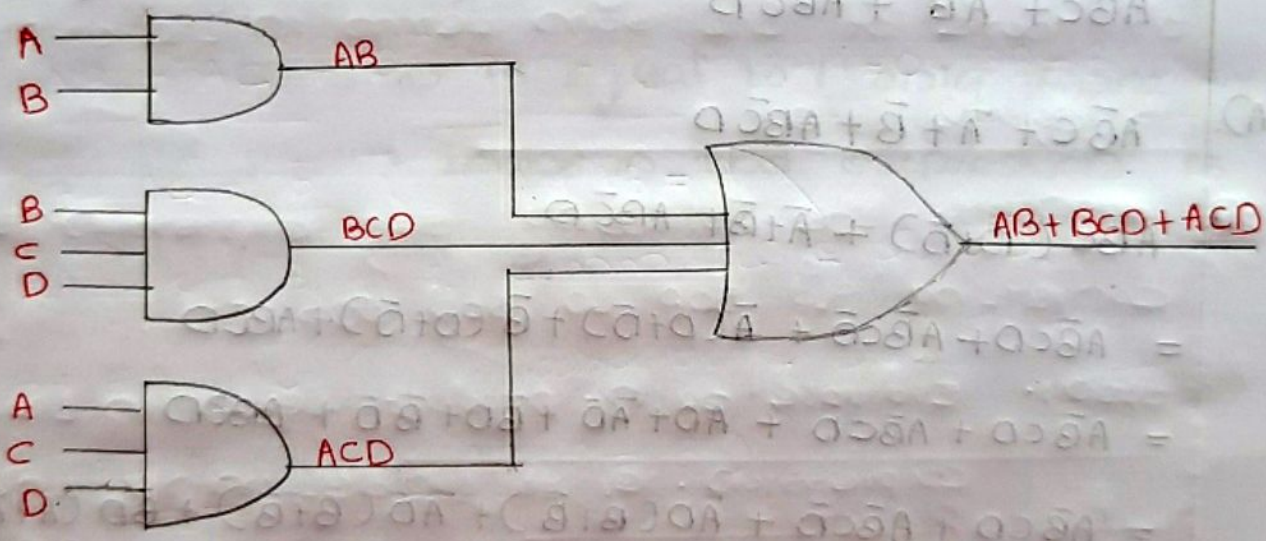
$ABC + CDE + \bar{B}C\bar{D}$

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

Domain of a Boolean expression.

The domain of a general Boolean expression is the set of variables contained in the expression in either complemented or uncomplemented form.

* Implementation of the SOP expression $AB+BCD+ACD$



Conversion of a General Expression to SOP form

1) $A(B+CD) = AB+ACD$

2) $AB + B(CD+EF)$

A) $AB + BCD + BEF$

3) $(A+B)(B+C+D)$

A) $AB + AC + AD + BB + BC + BD$

$= AB + AC + AD + B + BC + BD$

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

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

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

The standard sop Form

A standard sop expression is one in which all the variables in the domain appear in each product term in the expression.

2 Convert the following boolean expression into standard sop form.

$A\bar{B}C + \overline{\bar{A}B} + AB\bar{C}D$

A). $A\bar{B}C + \bar{A} + \bar{B} + AB\bar{C}D$

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

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

$= A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}D + \bar{A}\bar{D} + \bar{B}D + \bar{B}\bar{D} + AB\bar{C}D$

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

$= A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}BD + \bar{A}\bar{B}D + \bar{A}\bar{B}B + \bar{A}B\bar{D} + A\bar{B}D + \bar{A}\bar{B}D + \bar{A}\bar{B}\bar{D} + \bar{A}\bar{B}\bar{D} + AB\bar{C}D$

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

$= A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D} +$

$A\bar{B}\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D$

? $A\bar{B}C + \bar{A}\bar{B} + AB\bar{C}D$

A) $A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + AB\bar{C}D$

$A\bar{B}CD + A\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D$

Binary Representation of a standard product Term

A std product term is equal to 1 for only one combination of variable values.

For eg: the product term $A\bar{B}C\bar{D}$ is equal to 1 when $A=1, B=0, C=1, D=0$

ie, $A\bar{B}C\bar{D} = 1 \cdot \bar{0} \cdot 1 \cdot \bar{0} = 1$

* An sop expression is equal to 1 only if one or more the product terms in the expression is equal to 1.

The product of sums (pos) form

when a or more sum terms are multiplied, the resulting expression is a product-of sums (pos).

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

2. convert the following Boolean expression into std pos form.

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

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

$(A+B)(A+C) = A+BC$

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

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

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

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

Binary Representation of a std sum term

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

$\Rightarrow 0+\bar{1}+0+\bar{1} = 0$

(0101)

* A pos expression is equal to 0 only if one or more of the sum term in the expression is equal to 0.

* [redacted]

Converting Std SOP to Std POS

step 1: Evaluate each product term in the SOP expression. ie, a [redacted] determine the binary numbers that represent the product term.

step 2. Determine all of the binary numbers not included in the evaluation in step 1.

step 3. Write the equivalent sum term for each binary number from step 2 and

express in pos term.

2 convert the following sop expression to an equivalent pos expression.

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

A). $000 + 010 + 011 + 101 + 111$

missing terms

Total 2^3 terms.

$001, 100, 110$

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

converting sop expressions to Truth table format

2 $\bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + ABC$

001 000 111

A	B	C	PDT	O/P
0	0	0	$\bar{A}\bar{B}\bar{C}$	1
0	0	1	$\bar{A}\bar{B}C$	1
0	1	0		0
0	1	1		0
1	0	0		0
1	0	1		0
1	1	0		0
1	1	1	ABC	1

? converting pos expression to truth table format

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

A	B	C	O/P
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

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

A). 010 001 111

A	B	C	O/P
0	0	0	1
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	0

2. $\bar{A}B\bar{C} + ABC$

010 111

A	B	C	O/P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

2. $\bar{A}B\bar{C} + ABC$

010 111

A	B	C	o/p
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

KARNAUGH MAP

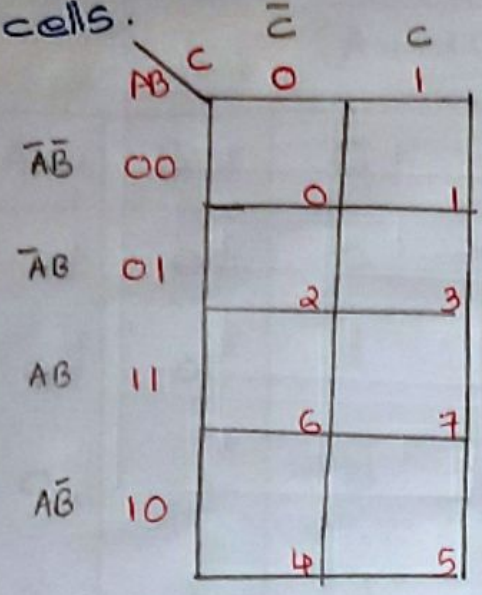
* A karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the minimum expression.

* Karnaugh map is an array of cells in which each cell represents a binary value of the input variables. The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.

* The number of cells in a karnaugh map is equal to the total no. of possible input variable combinations as is the no. of rows in a truth table.

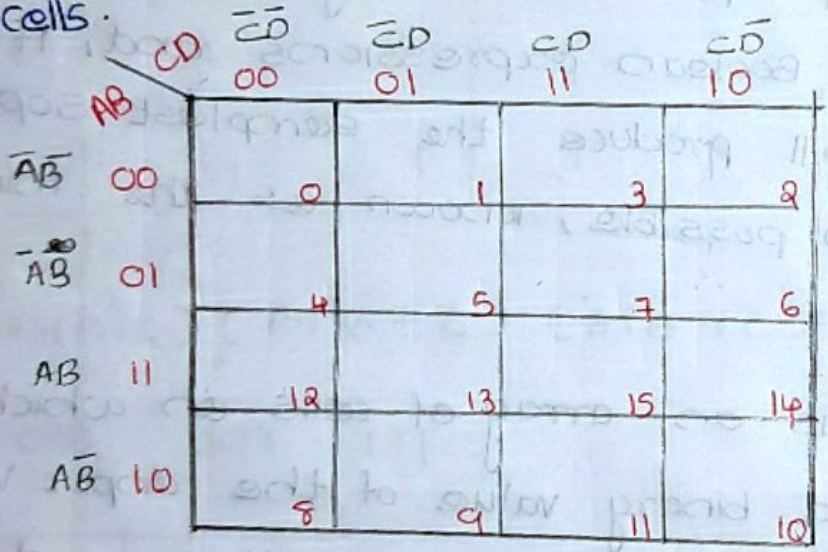
3- variable karnaugh Map

The 3- variable karnaugh map is an array of 8 cells.



4- variable karnaugh map

The 4- variable karnaugh map is an array of 16 cells.



Cell Adjacency

The cells in a K map are arranged so that there is only a single-variable change b/w adjacent cells. In 3-variable map the 010 cell is adjacent to the 000 cell, 110 cell, 011 cell.

Mapping a std SOP expression

step 1: Determine the binary value of each product term in the std SOP expression.

step 2: As each product term is evaluated, place a 1 on the karnaugh map on the cell having the same value as the product term.

? $\bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$

A) $\bar{A}BC = 001$ $AB\bar{C} = 110$

$\bar{A}B\bar{C} = 010$ $ABC = 111$

AB \ C	0	1
00		1
01	1	
11	1	1
10		

? Map the following std SOP expression on a K map

$\bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D + ABCD + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}C\bar{D}$

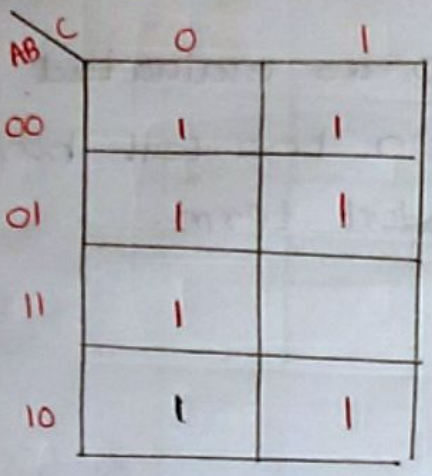
A) $\bar{A}\bar{B}CD = 0011$ $AB\bar{C}D = 1101$ $AB\bar{C}\bar{D} = 1100$ $A\bar{B}C\bar{D} = 1010$

$\bar{A}B\bar{C}\bar{D} = 0100$ $ABCD = 1111$ $\bar{A}\bar{B}\bar{C}D = 0001$

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

Mapping a Non-std SOP expression

1). $\bar{A} + A\bar{B} + AB\bar{C}$

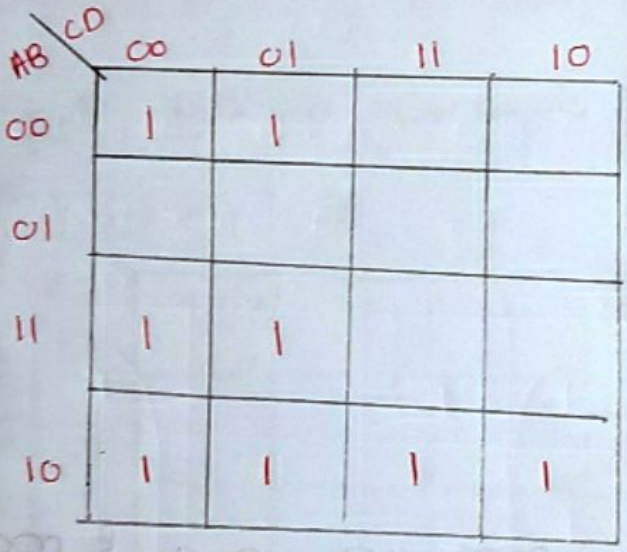


$\bar{A} \Rightarrow 0$
$\Rightarrow 000$
$\Rightarrow 010$
$\Rightarrow 011$
$\Rightarrow 001$

$A\bar{B} \Rightarrow 10$
$\Rightarrow 100$
$\Rightarrow 101$
$\Rightarrow$

$AB\bar{C} \Rightarrow 110$

2). $\bar{A}\bar{B}\bar{C}D + A\bar{B}CD + A\bar{B}C\bar{D} + \bar{B}\bar{C} + A\bar{B} + AB\bar{C}$



$\bar{A}\bar{B}\bar{C}D \rightarrow 0001$

$A\bar{B}CD \rightarrow 1011$

$A\bar{B}C\bar{D} \rightarrow 1010$

$\bar{B}\bar{C} \rightarrow 0000, 1000, 1001, 0001$

$A\bar{B} \rightarrow 1000, 1001, 1010, 1011$

$AB\bar{C} \rightarrow 1100, 1101$

K map simplification of SOP Expression.

After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map.

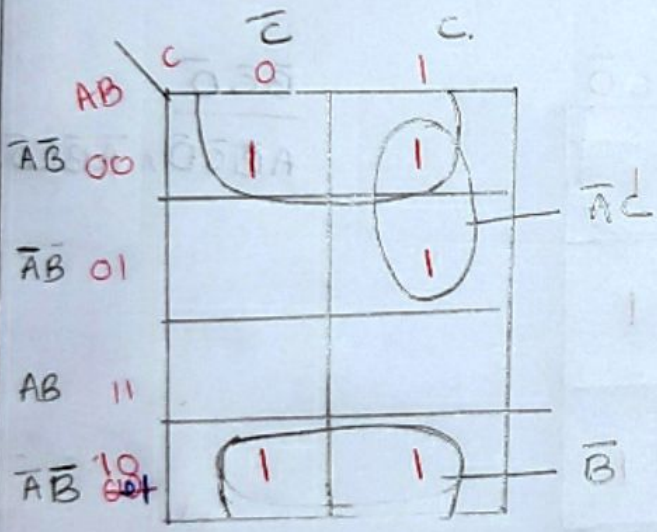
Grouping the 1s.

1. A group must contain either 1, 2, 4, 8 or 16 cells which are all powers of two.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each cell.
3. Always include the largest possible no. of 1s in a group in accordance with rule 1.

2. Use a kmap to minimize the following std sop expression

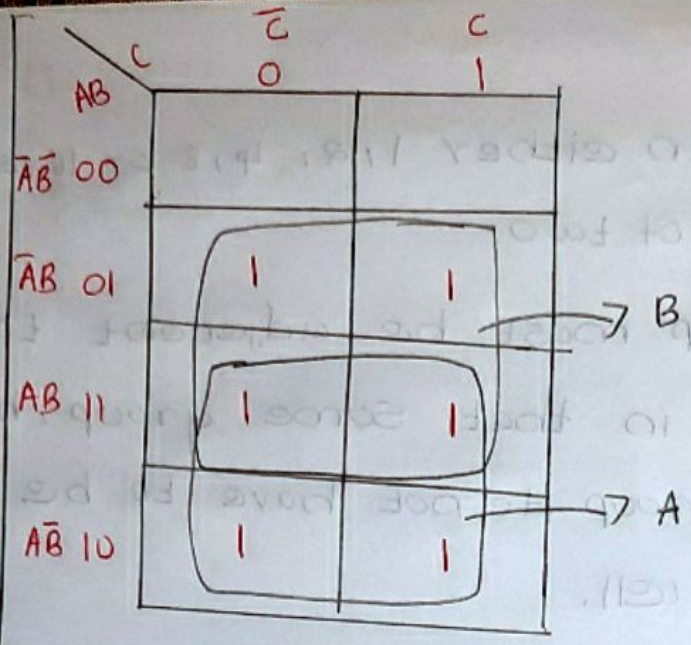
1) $A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

A)



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

2) $A\bar{B}C + AB\bar{C} + \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

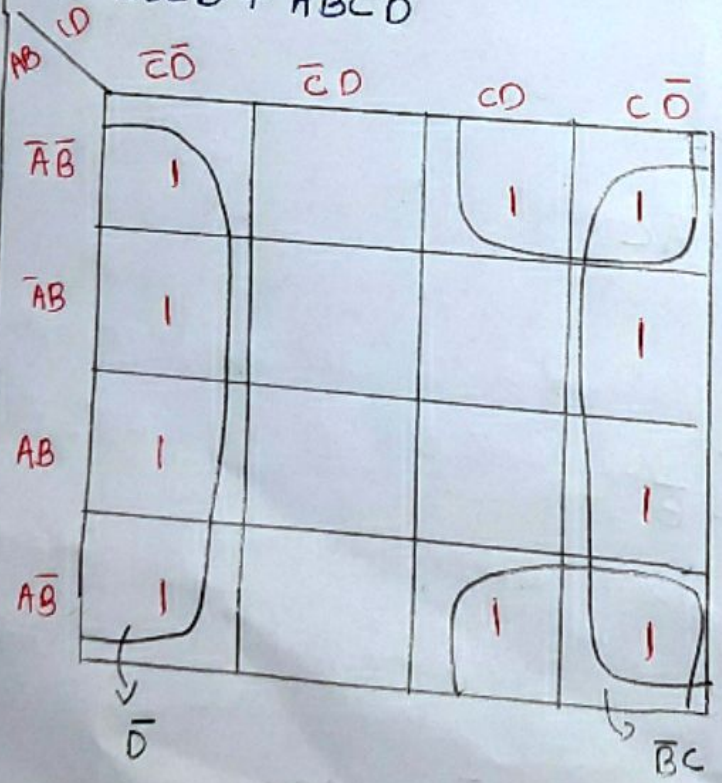


A+B

? use a K map to maximize the following SOP expression.

$\bar{B}C\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}BCD + ABC\bar{D} + A\bar{B}C\bar{D}$

A)



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

$AB\bar{C}\bar{D}$, $\bar{A}\bar{B}C\bar{D}$

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

Mapping a std POS Expression

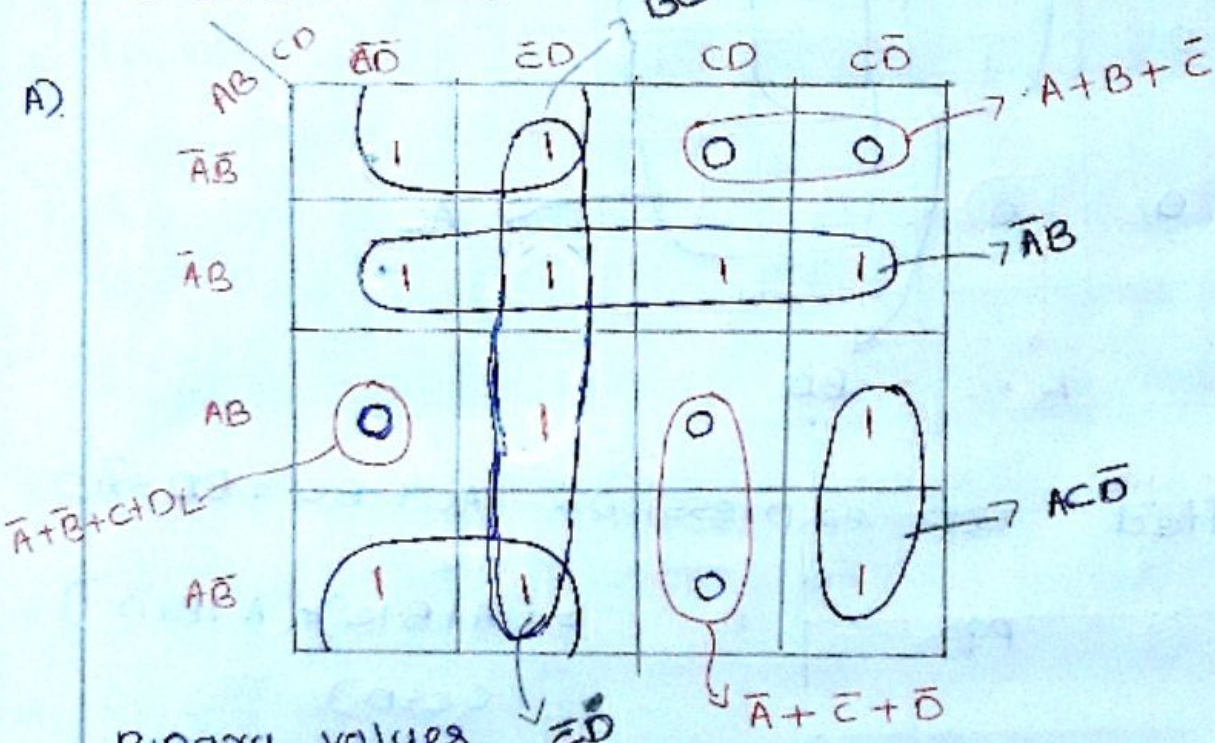
1. Determine the binary value of each sumterm in the std POS expression.
2. As each sum term is evaluated, place a 0 on the K map in the corresponding cell.

2. Map the following std pos expression on K-map.

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

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

A)



Binary values

1100 1011 0010 1111 0011

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

simplified sop expression = $AC\bar{D}+\bar{C}\bar{D}+\bar{B}\bar{C}+\bar{A}B$

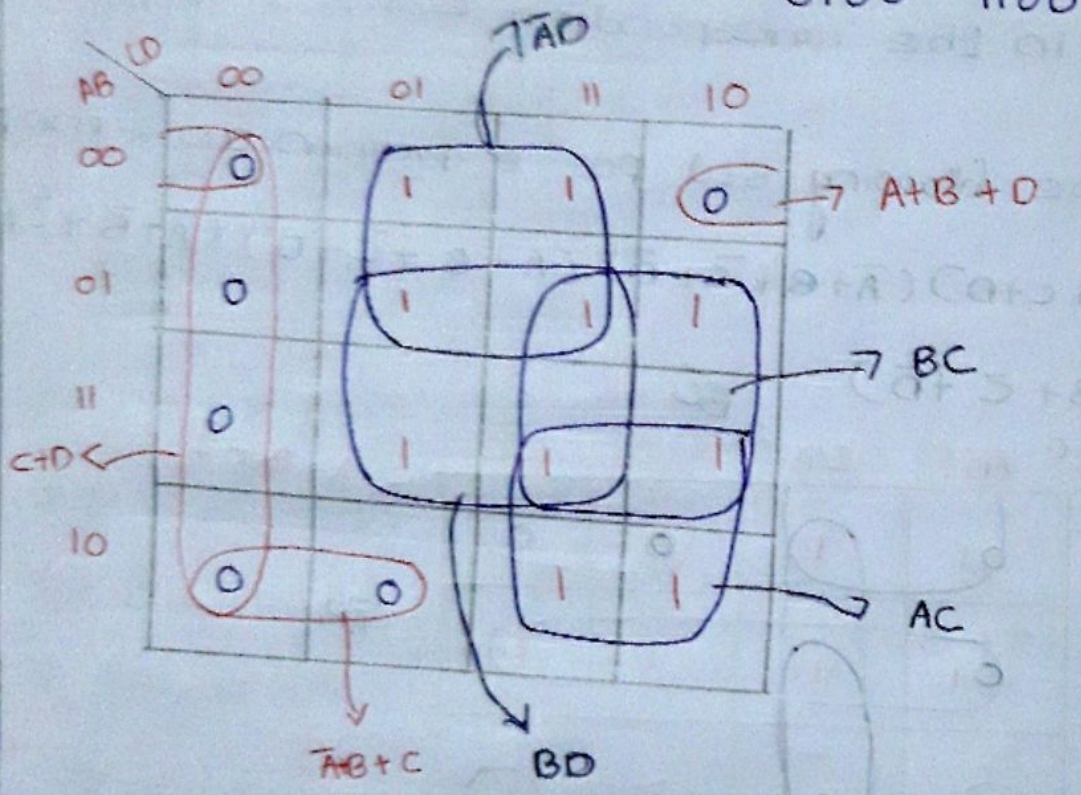
* $\Sigma(0,2,4) = \Pi(1,3,5,6,7)$ (3 variable K map)

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

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

A) Binary values.

1000 0000 0010 1001 0100 1100



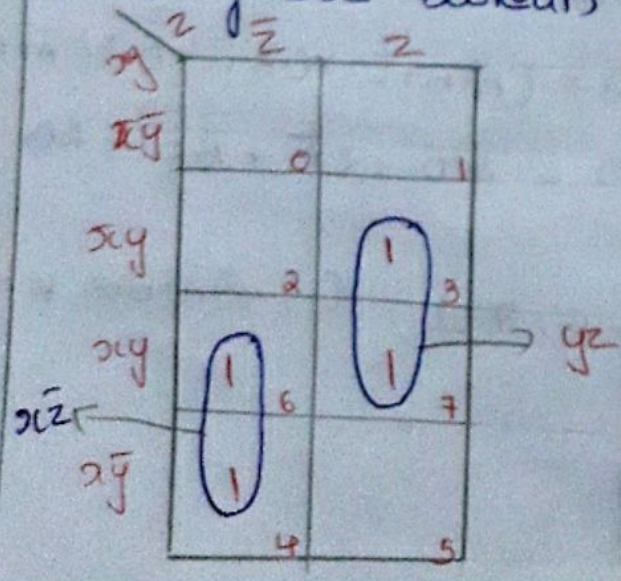
Samplified SOP expression = $AC + BC + B\bar{D} + \bar{A}D$

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

$(C+D)$

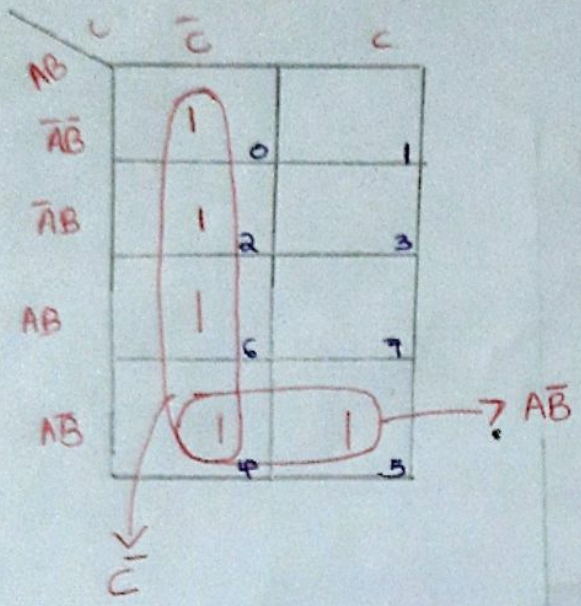
? simplify the boolean expression $f(x,y,z) = \Sigma(3,4,6,7)$

A)



$x\bar{z} + yz$

2 $F(A,B,C) = \Sigma(0,2,4,5,6)$



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

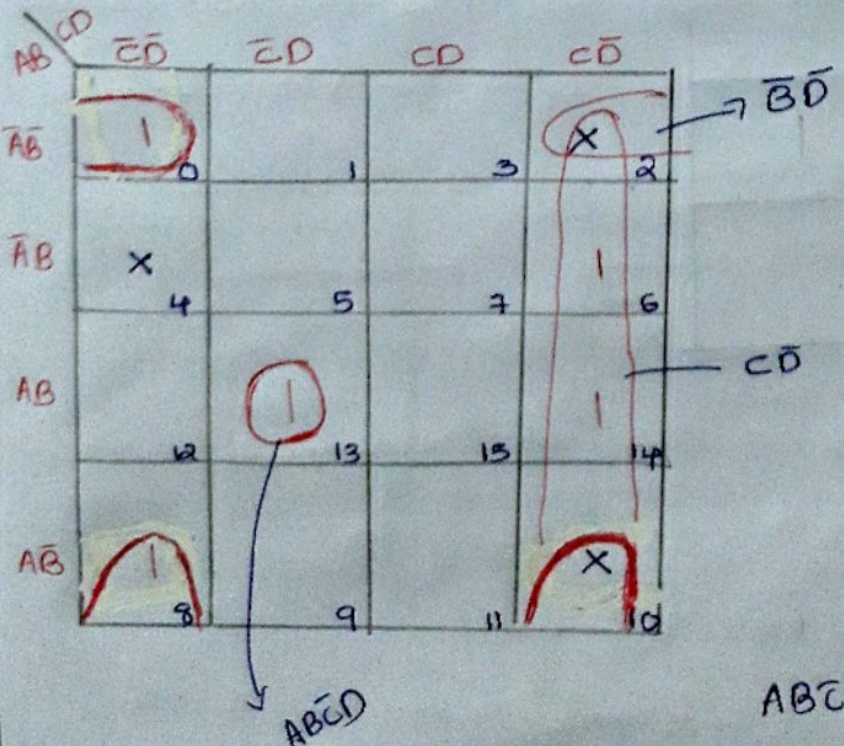
"Don't care" conditions

For each don't care term, an X is placed in the cell. when grouping the 1s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be.

2 $F(A,B,C,D) = \Sigma(0,6,8,13,14)$

$d(A,B,C,D) = \Sigma(2,4,10)$

A)



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

Mapping Directly from a Truth Table

2

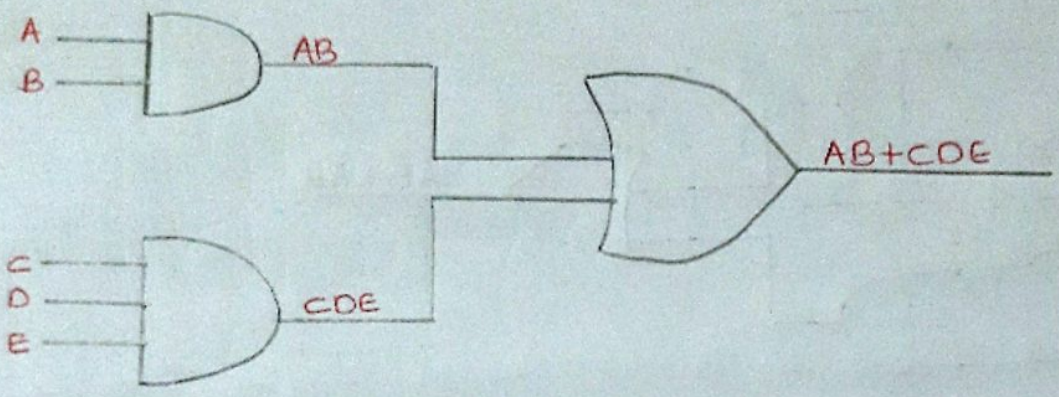
A	B	C	O/P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

A).

AB \ C	0	1
00	1	
01		
11	1	1
10	1	

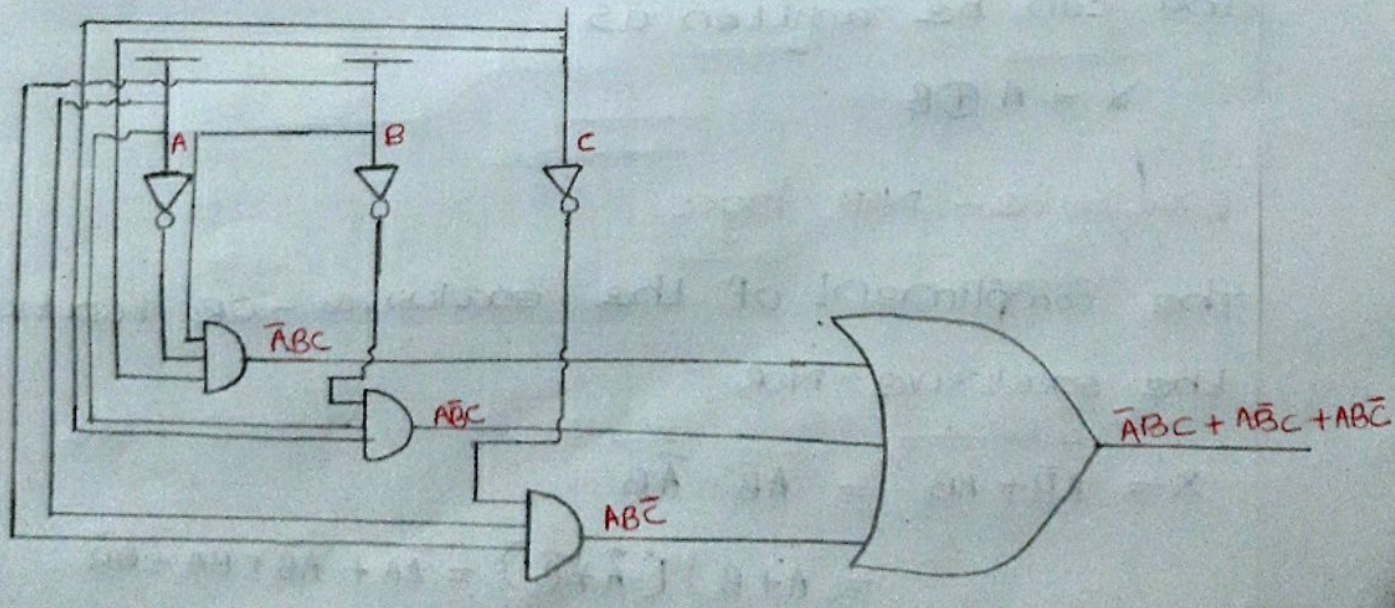
AND-OR Logic

1) AB + CDE



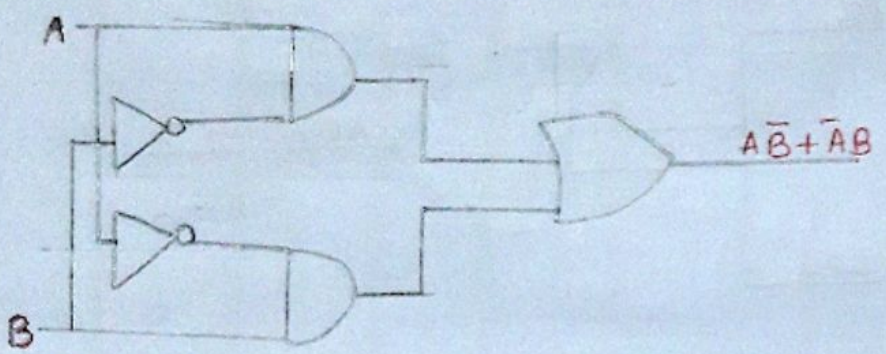
2).

A	B	C	O/P	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	→ $\bar{A}BC$
1	0	0	0	
1	0	1	1	→ $A\bar{B}C$
1	1	0	1	→ $AB\bar{C}$
1	1	1	0	



Exclusive-OR Logic.

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



The output expression for the above circuit is,

$X = A\bar{B} + \bar{A}B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Notice that the output is HIGH only when the 2 inputs are at opposite levels. A special exclusive-OR operator $\oplus$ is often used, so the expression $X = A\bar{B} + \bar{A}B$ can be stated as X is equal to A exclusive-OR B and can be written as

$X = A \oplus B$

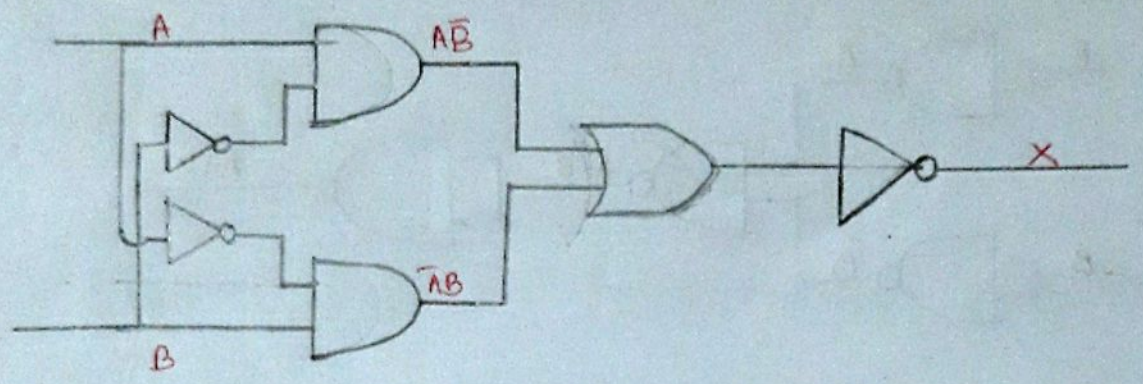
Exclusive-NOR logic

The complement of the exclusive-OR function is the exclusive-NOR.

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

$$= (\bar{A} + B)(A + \bar{B}) = \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B}$$

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

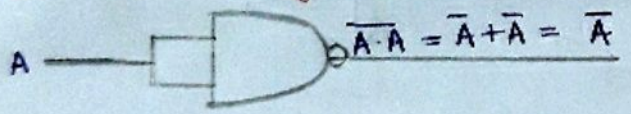


$X = A \odot B$

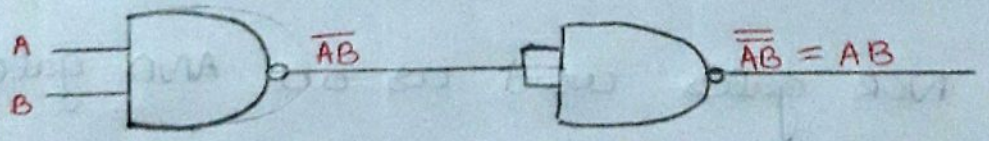
The NAND Gate as a Universal logic element

The NAND gate is a universal gate because it can be used to produce the NOT, the AND, the OR and the NOR functions.

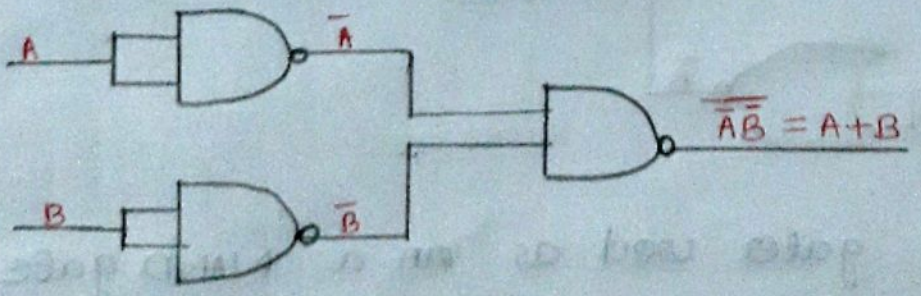
a) One NAND gate used as an inverter



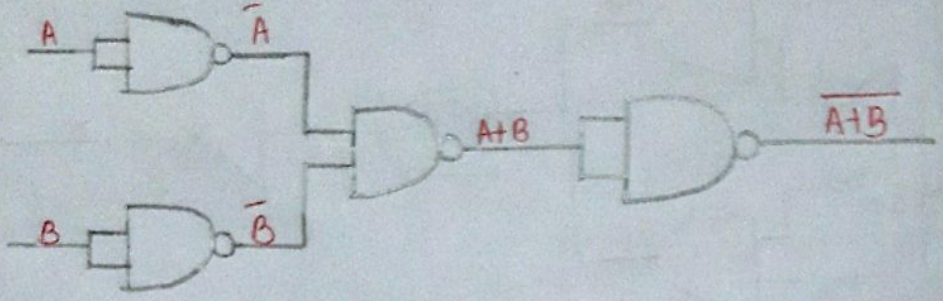
b) Two NAND gates used as AND gate.



c) Three NAND gates used as an OR gate

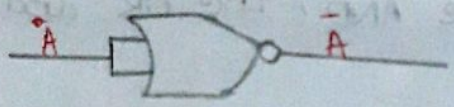


d) Four NAND gates used as a NOR gate

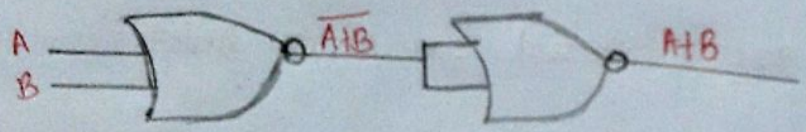


The NOR gate as a Universal Logic Element:

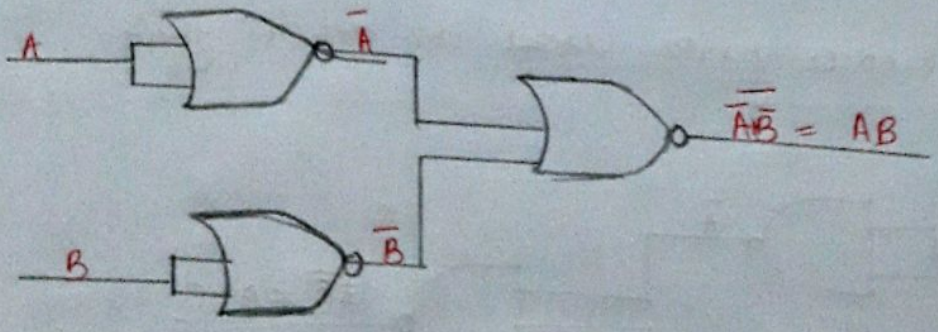
a) One NOR gate used as an inverter



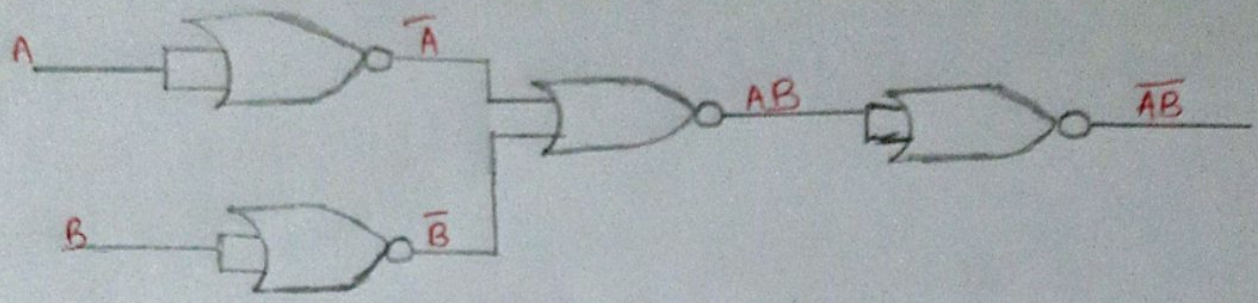
b) Two NOR gates used as an OR gate.



c) Three NOR gates used as an AND gate.

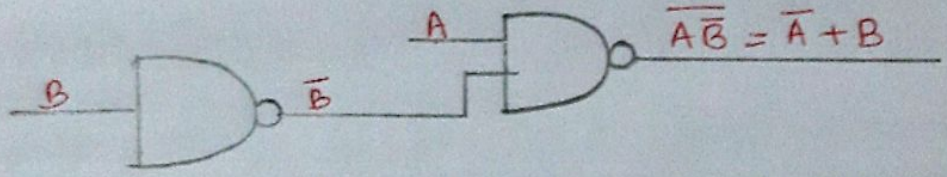


d) four NOR gates used as a NAND gate



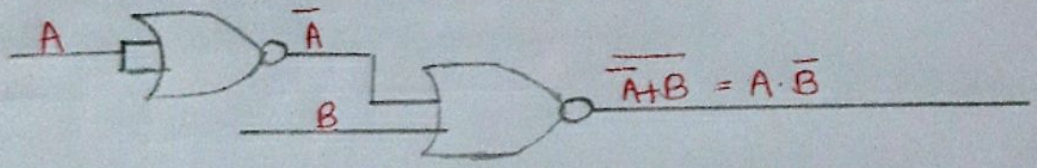
2. Use NAND gates to implement the expression

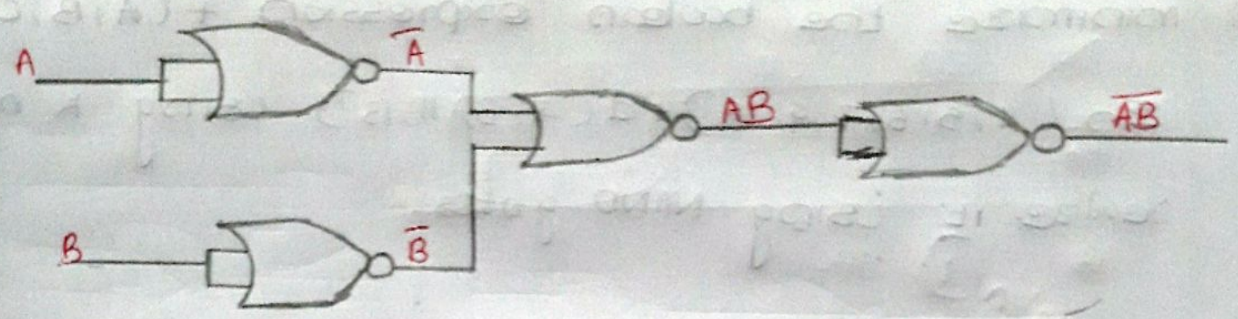
$X = \overline{A} + B$



3. Use NOR gates to implement the expression

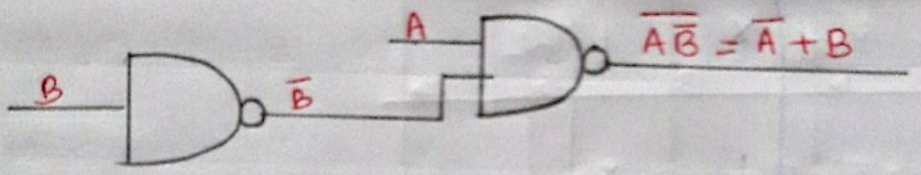
$X = A\overline{B}$





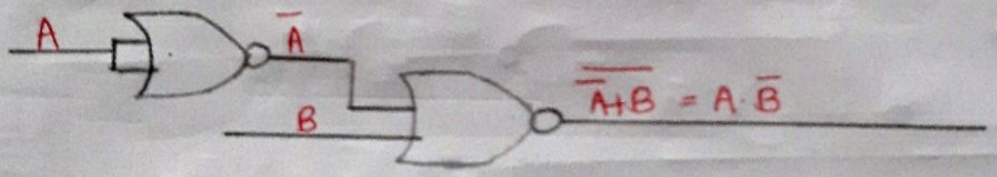
2. Use NAND gates to implement the expression

$X = \overline{A} + B$



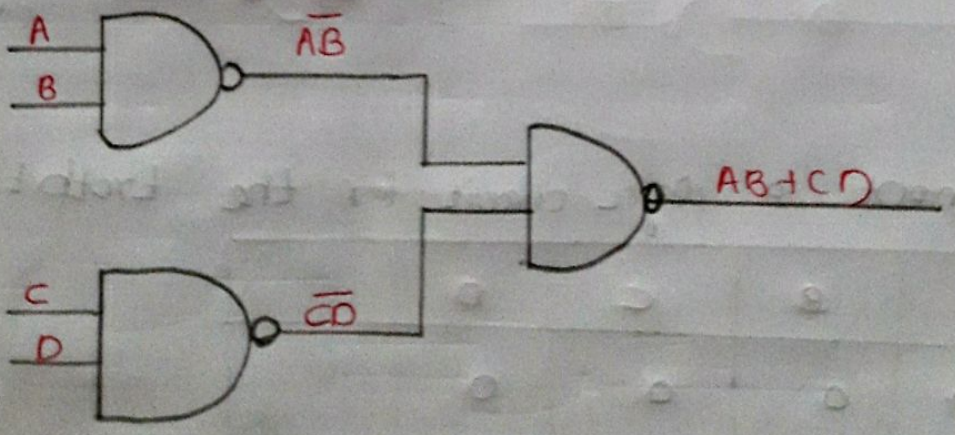
3. Use NOR gates to implement the expression

$X = A\overline{B}$



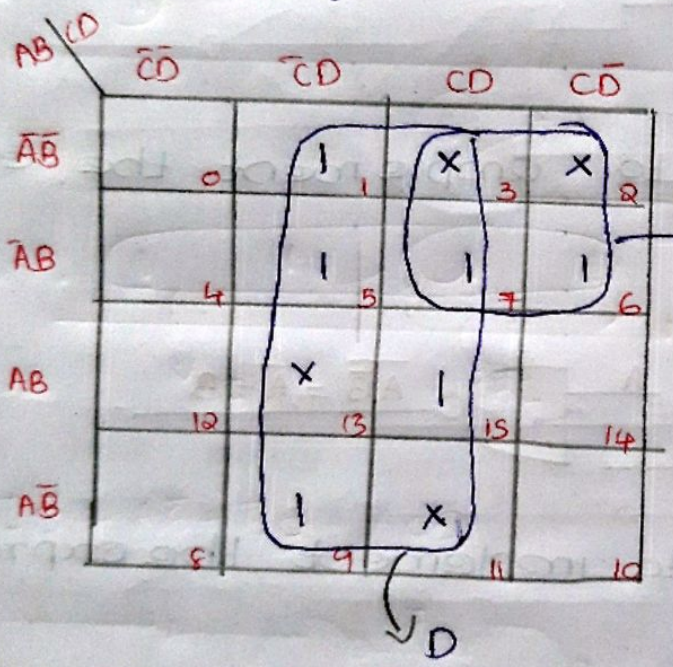
4. Using NAND gate implement the expression.

$X = AB + CD$

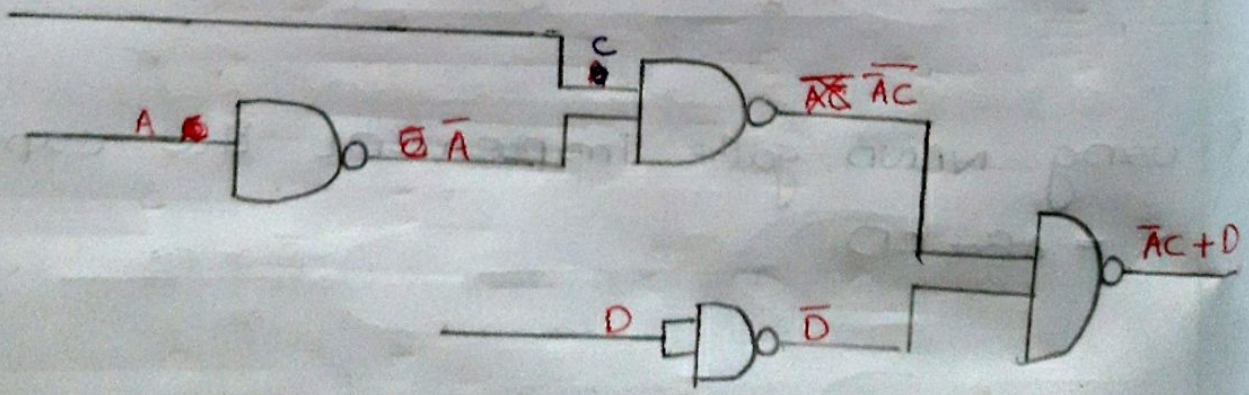


2 minimize the boolean expression $f(A,B,C,D) = \Sigma m(1,5,6,7,9,15) + d(2,3,11,13)$ using K map and realize it using NAND gates.

A)



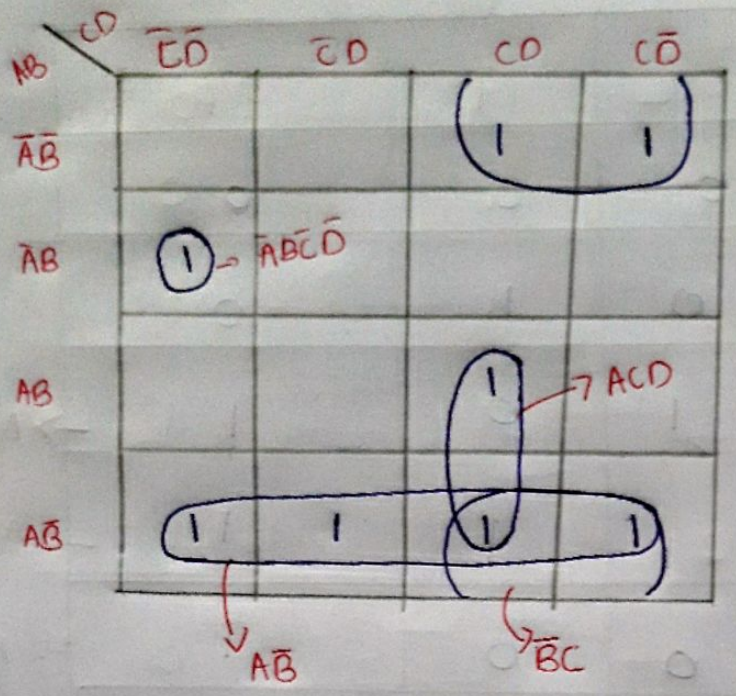
$\bar{A}C + D$



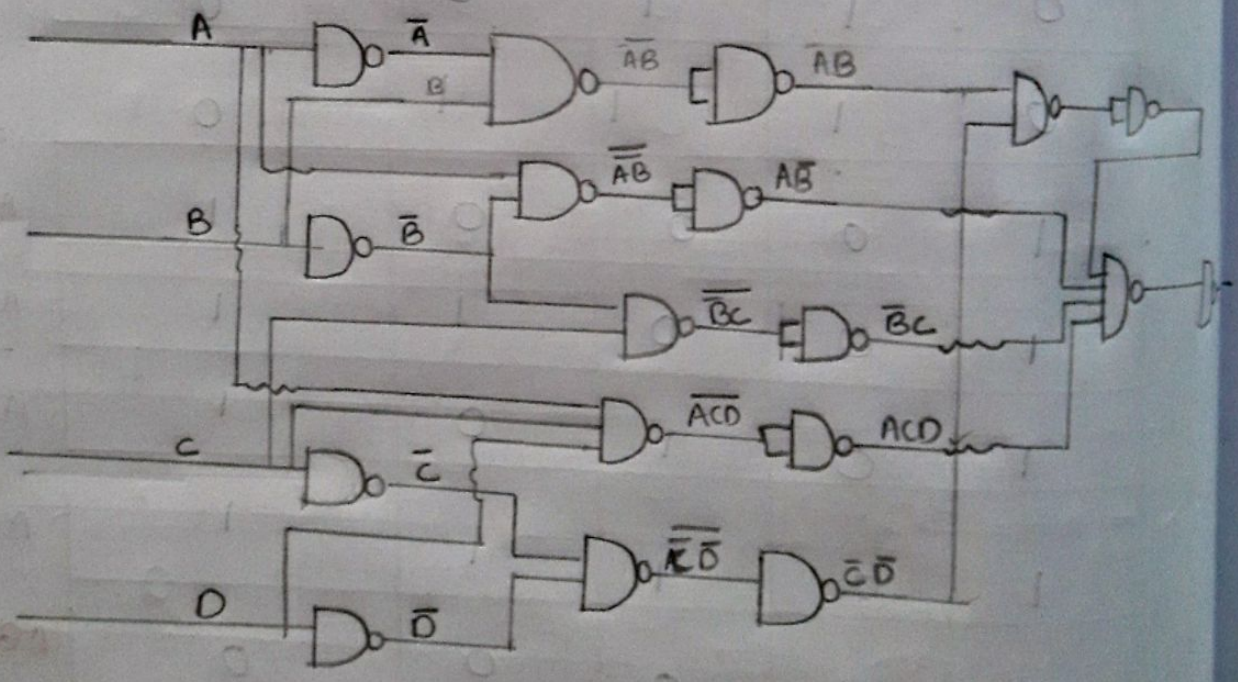
3 implement a [illegible] for the truth table

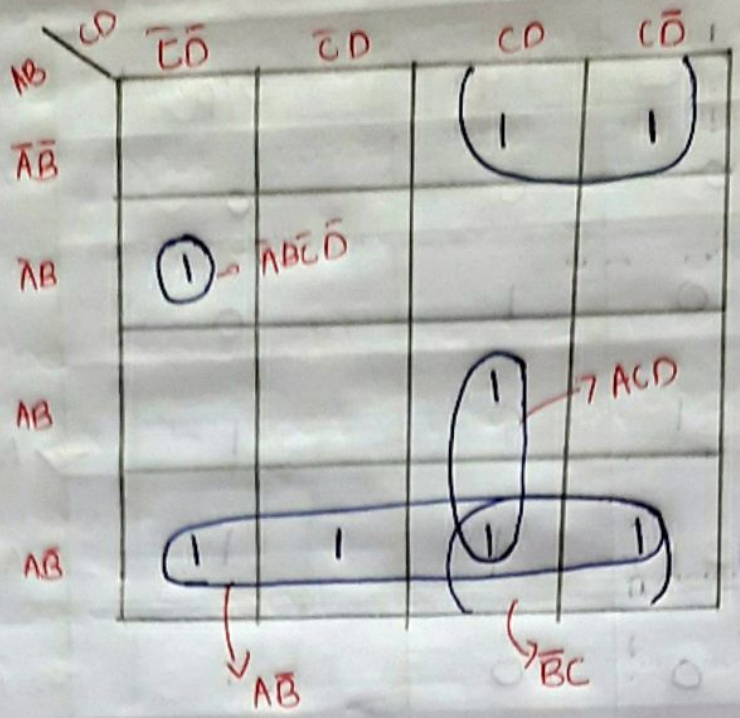
A	B	C	D	O/P	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	→ $\bar{A}B\bar{C}\bar{D}$
0	0	1	1	1	→ $\bar{A}\bar{B}CD$
0	1	0	0	1	→ $\bar{A}B\bar{C}\bar{D}$
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	1	→ $A\bar{B}\bar{C}\bar{D}$
1	0	0	1	1	→ $A\bar{B}\bar{C}D$
1	0	1	0	1	→ $A\bar{B}C\bar{D}$
1	0	1	1	1	→ $A\bar{B}CD$
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	→ $ABCD$

A) Sop expression → $\bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}CD + ABCD$

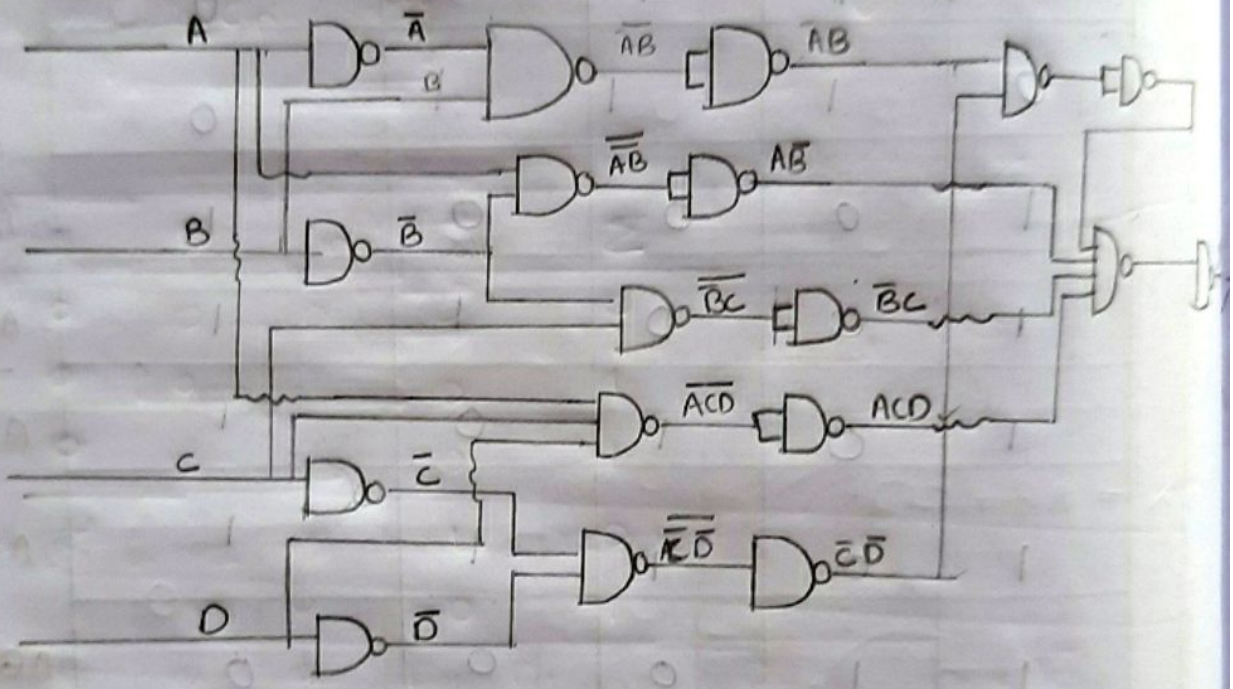


$\bar{A}B\bar{C}\bar{D} + A\bar{B} + \bar{B}C + ACD$





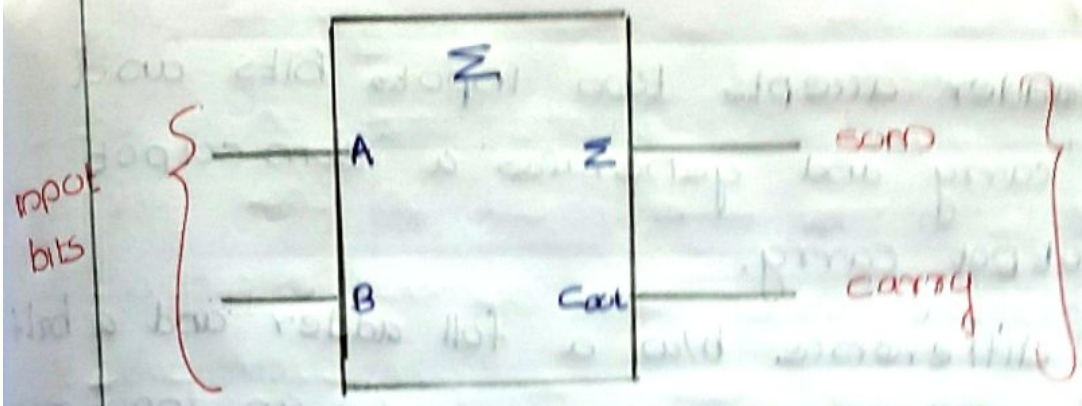
$\bar{A}B\bar{C}\bar{D} + A\bar{B} + \bar{B}C + ACD$



Half adder

The half-adder accepts two binary digits on its input and produces two binary digits on its outputs, a sum bit and a carry bit.

Logic symbol for a half-adder is given by,



A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

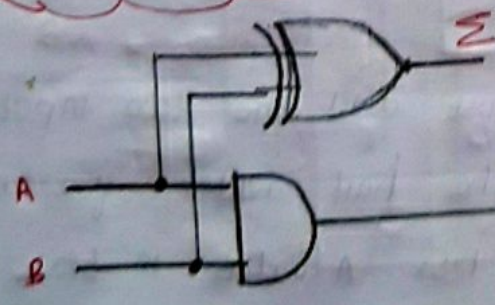
* Notice that the output carry (C_{out}) is a 1 only when both A and B are 1s: ∴, C_{out} can be expressed as the AND of the input variables.

$$C_{out} = AB$$

* Now observe that the sum output (Σ) is a 1 only if the input variables, A and B are not equal. The sum can therefore be expressed as the exclusive-OR of the input variables.

$$\Sigma = A \oplus B$$

$$\Sigma = A \oplus B = A\bar{B} + \bar{A}B$$

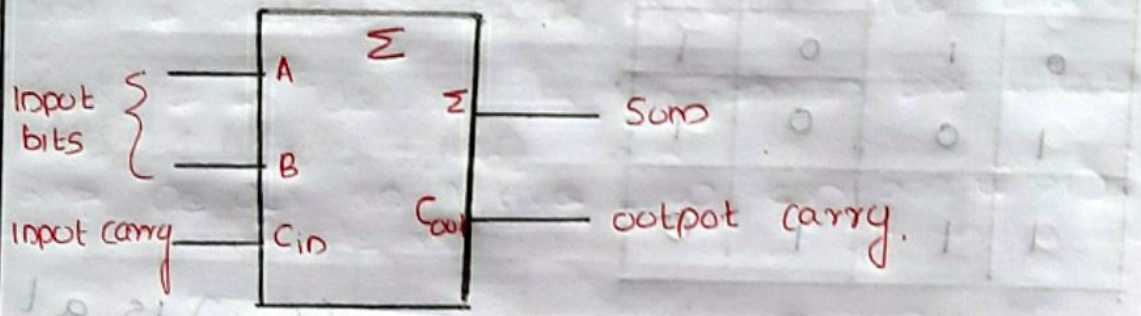


Full - Adder

The full - adder accepts two inputs bits and an input carry and generates a sum output and an output carry.

* The basic difference b/w a full adder and a half adder is that the full - adder accepts an input carry.

* Logic symbol for full adder,



A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

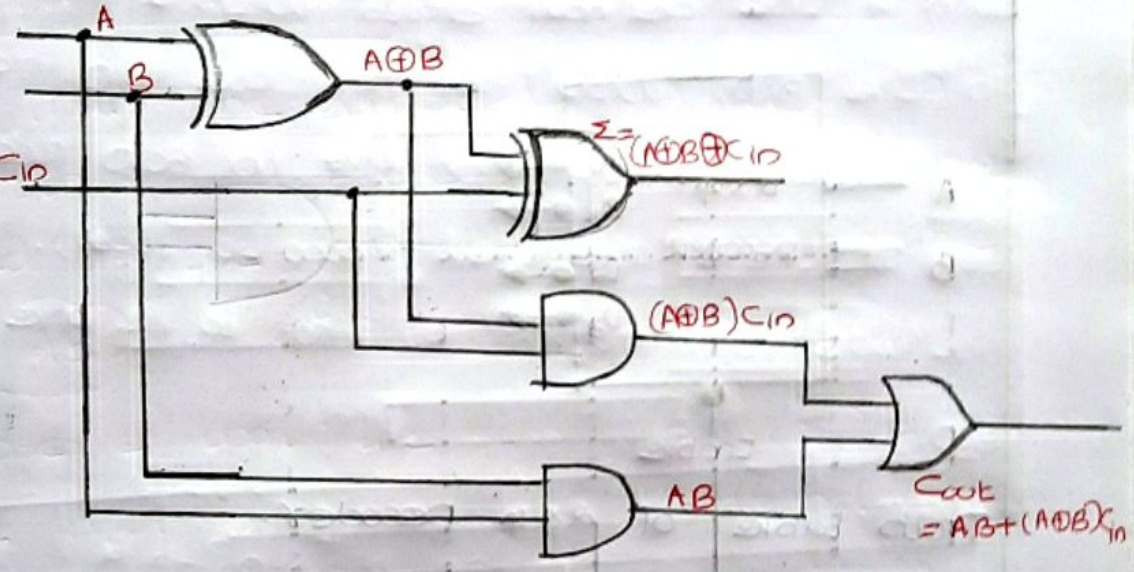
The ~~full~~ a full - adder must add the two input bits and the input carry. From the half - adder you know that the sum of the input bits A and B is the exclusive-OR of those two variables, $A \oplus B$, for the input carry (C_{in}) to be added to the input bits, it must

be exclusive ORed with A⊕B. Therefore,
The equation for the sum output of the full adder is,

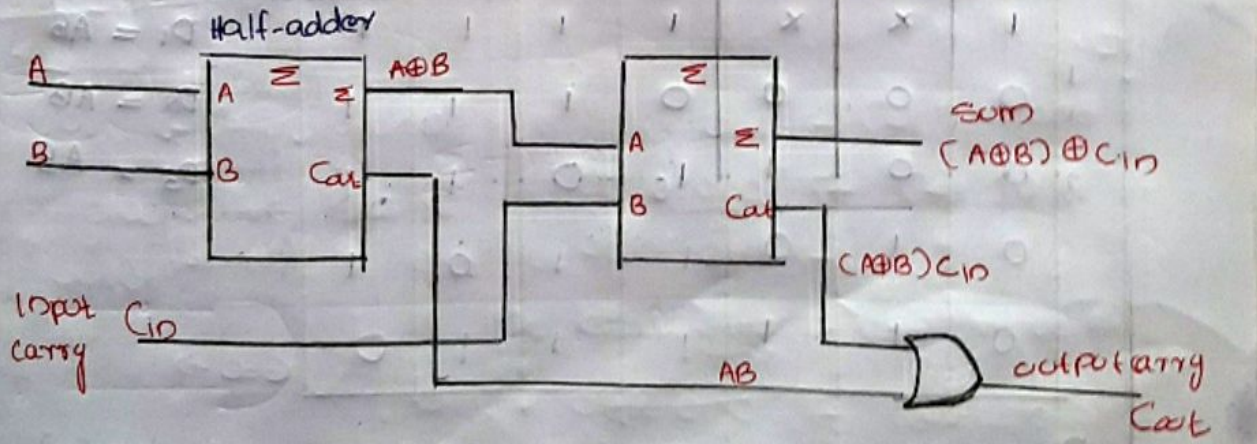
$$\Sigma = (A \oplus B) \oplus C_{in}$$

* The output carry is a 1 when both inputs to the first XOR gate are 1s or when both inputs to the second XOR gate are 1s.
The equation for the output carry of the full adder is,

$$C_{out} = AB + (A \oplus B) C_{in}$$



Logic circuit for a full-adder.



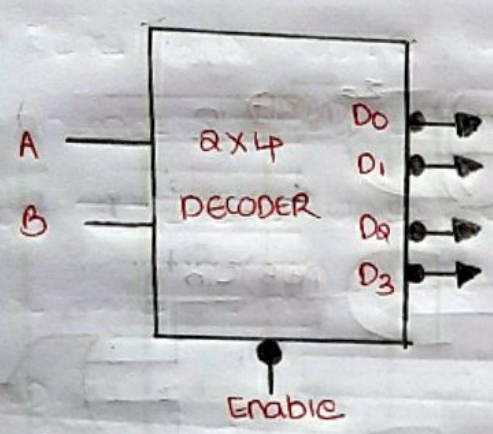
Decoders

Decoder is a combinational circuit that has 'n' input lines, and maximum of 2^n output lines. on of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code.

* 2 to 4 Decoder

Let 2 to 4 decoder has two inputs A and B and 4 outputs D_3, D_2, D_1 & D_0. The block diagram of 2 to 4 decoder is shown in the following figure.

2x4 Decoder an Active Low configuration.



Truth table of 2x4 Decoder.

E	A	B	D_0	D_1	D_2	D_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

D_3 D_2 D_1 D_0 →

$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

$D_2 = A\bar{B}$

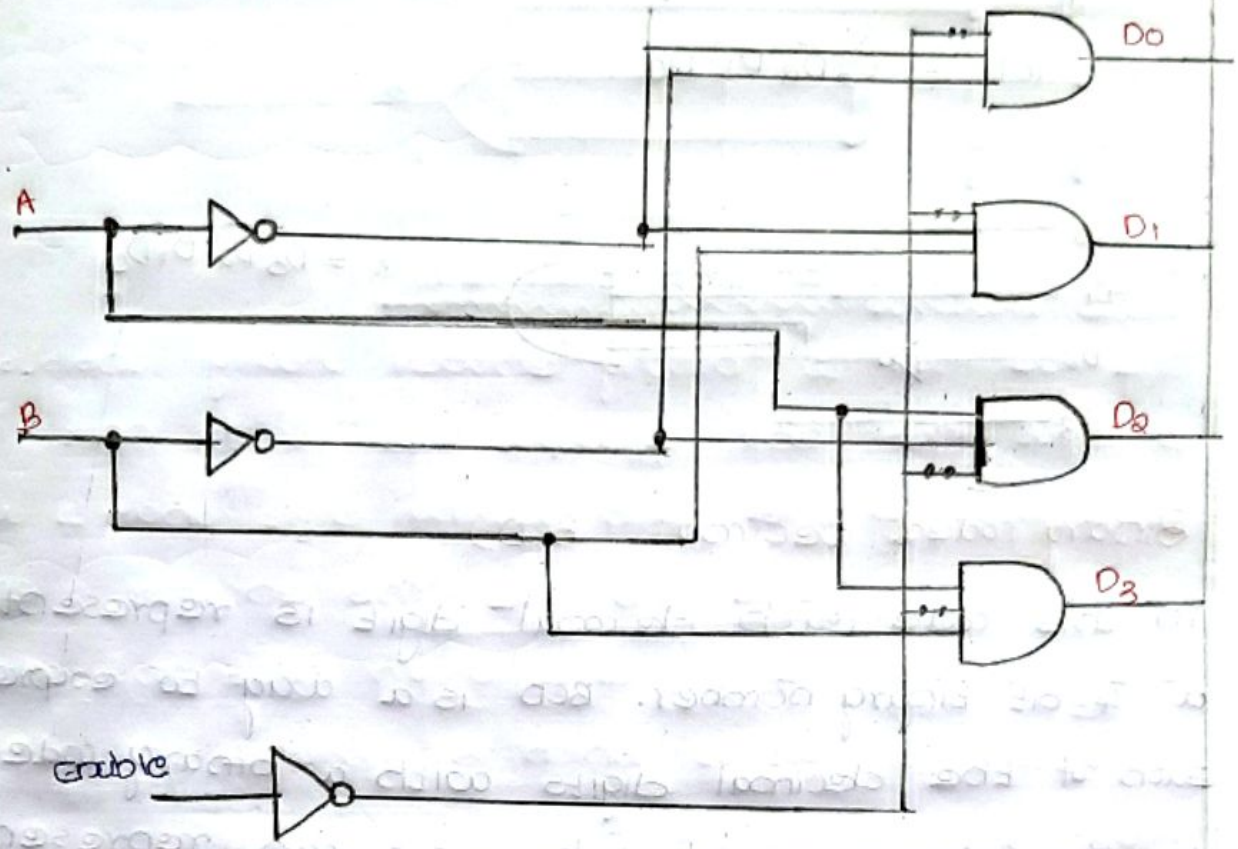
$D_3 = AB$

E=0 →

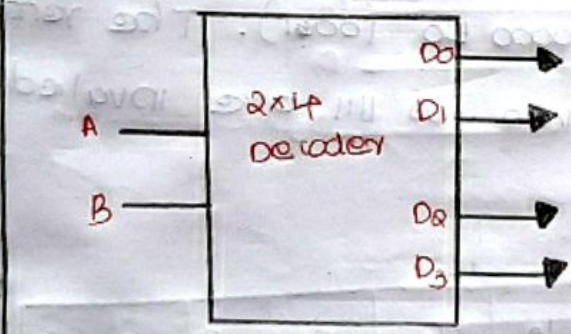
D_0 = one of these four outputs will be '0' for each combination of inputs when enable, E is 0.

2×4 Decoder in Active Low configuration

gate Diagram



2×4 Decoder in Active High configuration.



circuit diagram

Enable

Truth table of 2×4 DECODER :-

E	A	B	D_0	D_1	D_2	D_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

D_0 D_1 D_2 D_3

$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

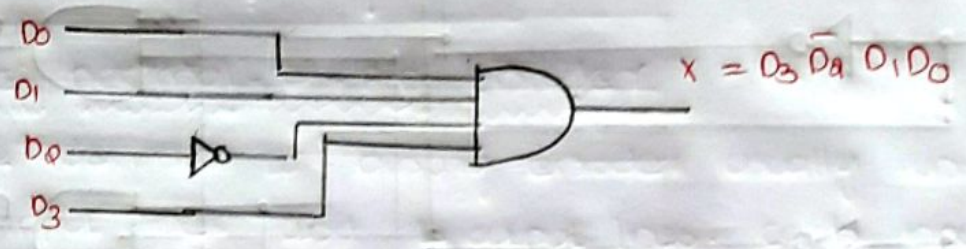
$D_2 = A\bar{B}$

$D_3 = AB$

one of these four outputs will be '1' for each combination of inputs when Enable, E is 1.

2. Determine the logic required to decode the binary number 1011 by producing a HIGH level on output

A) $X = 1011 = D_3 \bar{D}_2 D_1 D_0$



Binary coded decimal (BCD)

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent 16 numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations ie, 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Eg:

Binary (14) → 1110

BCD (14) → 00010100

→ 10100

Binary (127) → 01111111

BCD (127) → 000100100111

→ 100100111

binary (89) → 1011001

BCD (89) → 01011001

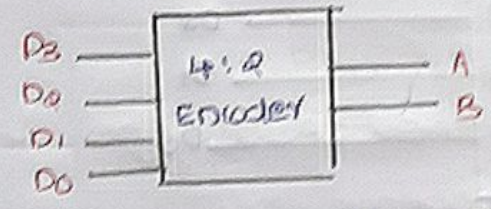
→ 1011001

Encoder

A encoder is a combinational logic circuit that essentially performs a "reverse" decoder function. It has maximum of 2^n input lines and 'n' output lines, hence it encodes the information from 2^n inputs onto an n-bit code. It will produce a binary code equivalent to the input, which is active High

4:2 Encoder

The 4 to 2 Encoder consists of four inputs D_3, D_2, D_1 & D_0 and 2 outputs A and B. At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.



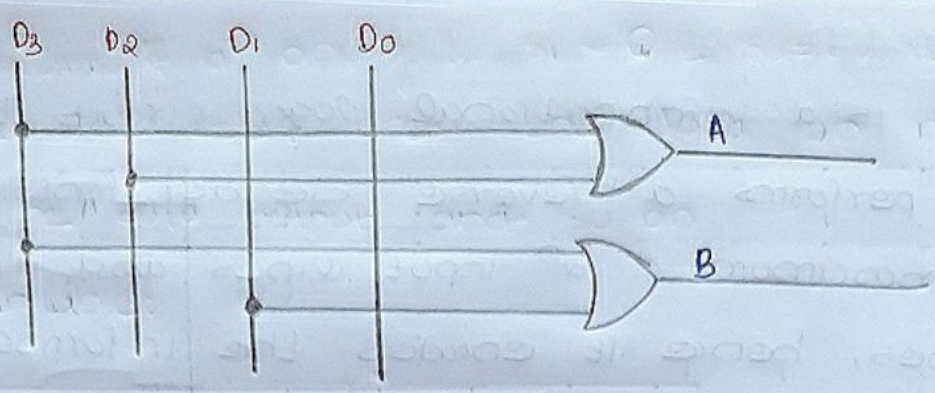
Logic symbol of 4:2 encoder

Truth table.

D_3	D_2	D_1	D_0	A	B
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

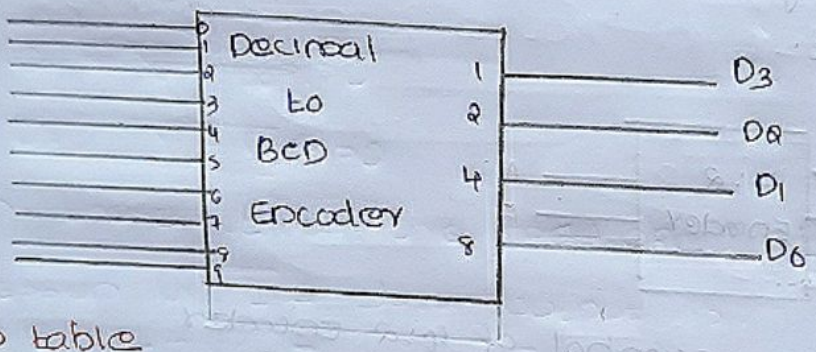
$A = D_3 + D_2$

$B = D_3 + D_1$



The Decimal-to-BCD Encoder

The decimal to binary encoder usually consists of 10 input lines and 4 output lines. Each input line corresponds to the each decimal digit and 4 outputs corresponds to the BCD code. This encoder accepts the decoded decimal data as an input and encodes it to the BCD output which is available on the output lines.



Truth table

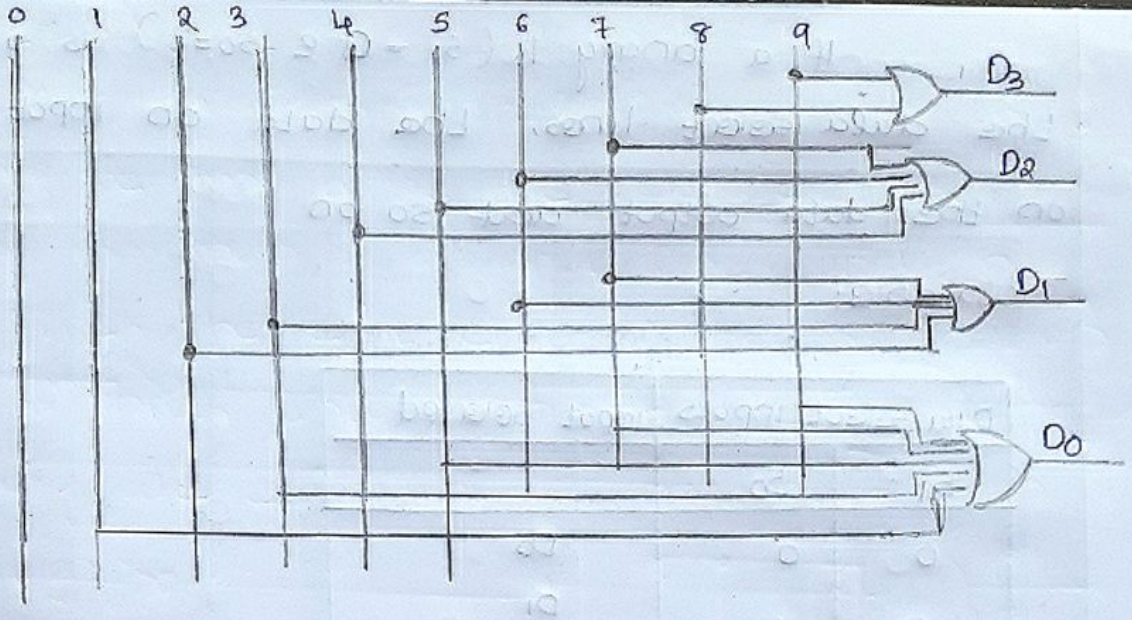
Decimal Digit	D_3	D_2	D_1	D_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

$D_3 = 8 + 9$

$D_2 = 4 + 5 + 6 + 7$

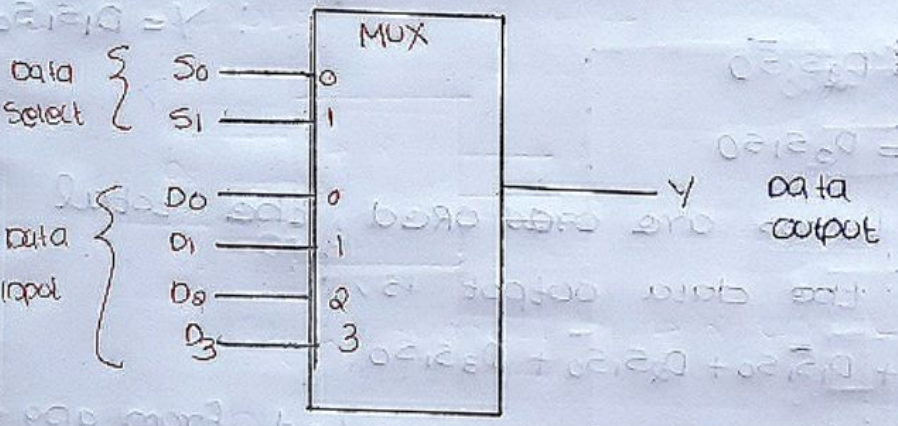
$D_1 = 2 + 3 + 6 + 7$

$D_0 = 1 + 3 + 5 + 7 + 9$



Multiplexers (Data selectors)

Multiplexer is a data selector which takes several inputs and gives a single output. In multiplexer are have 2^n input lines and 1 output lines where n is the no. of selection lines.



Logic symbol for a 4-input Multiplexer.

Notice that there are two data-select lines bcoz with two select bits, any one of the 4 data-input lines can be selected.

A 2 bit code on the data-select (S) inputs will allow the data on the selected data input to pass through to the data output.

If a binary 0 ($S_1 = 0$ & $S_0 = 0$) is applied to the data-select lines, the data on input D_0 appear on the data-output line.

If a binary 1 ($S_1 = 0$ & $S_0 = 1$) is applied to the data select lines, the data on input D_1 appears on the data output and so on

Truth table:

Data select inputs		Input selected
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

The data output is equal to D_0 only if $S_1 = 0$ & $S_0 = 0$: $Y = D_0 \bar{S}_1 \bar{S}_0$

The data output is equal to D_0 only if $S_1 = 0$ & $S_0 = 1$: $Y = D_1 \bar{S}_1 S_0$

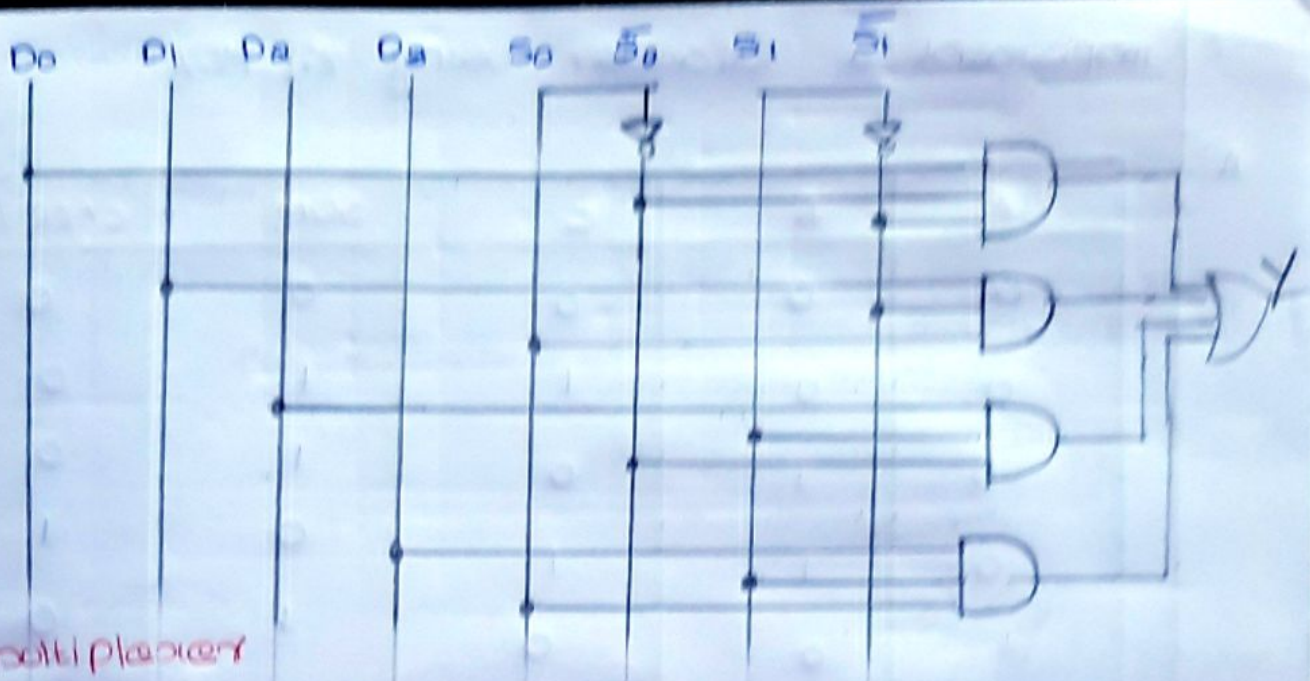
D_2, $Y = D_2 S_1 \bar{S}_0$

D_3, $Y = D_3 S_1 S_0$

when these terms are ORed, the total expression for the data output is,

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0$$

Because data can be selected from any one of the input lines, this circuit is also referred to as a data selector.

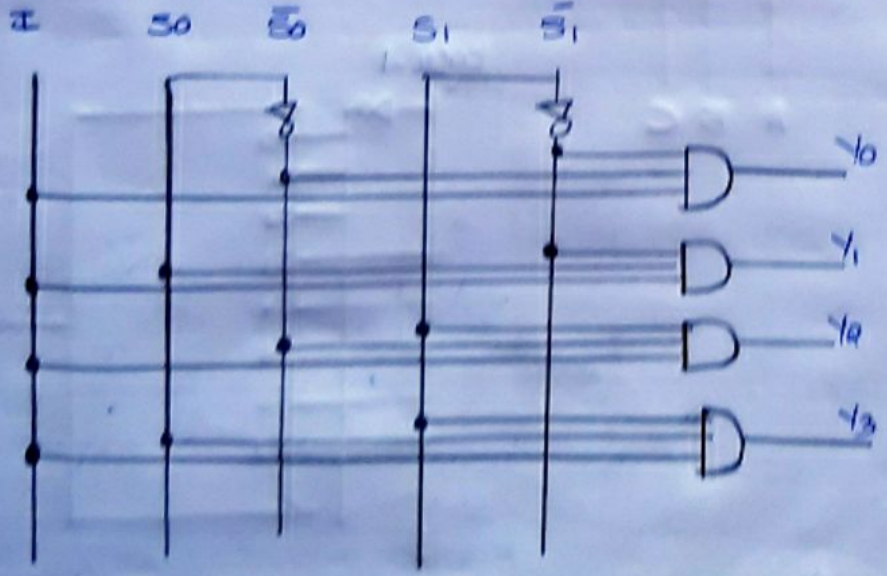


Demultiplexer

A demultiplexer (DEMUX) basically reverse the multiplexing function. It takes digital information from one line and distributes it to a given no. of output lines. For this reason, the demultiplexer is also known as a data distributor.

S_1	S_0	y_0	y_1	y_2	y_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

$Y_0 = \bar{S}_1\bar{S}_0 I$, $Y_1 = \bar{S}_1 S_0 I$, $Y_2 = S_1\bar{S}_0 I$, $Y_3 = S_1 S_0 I$

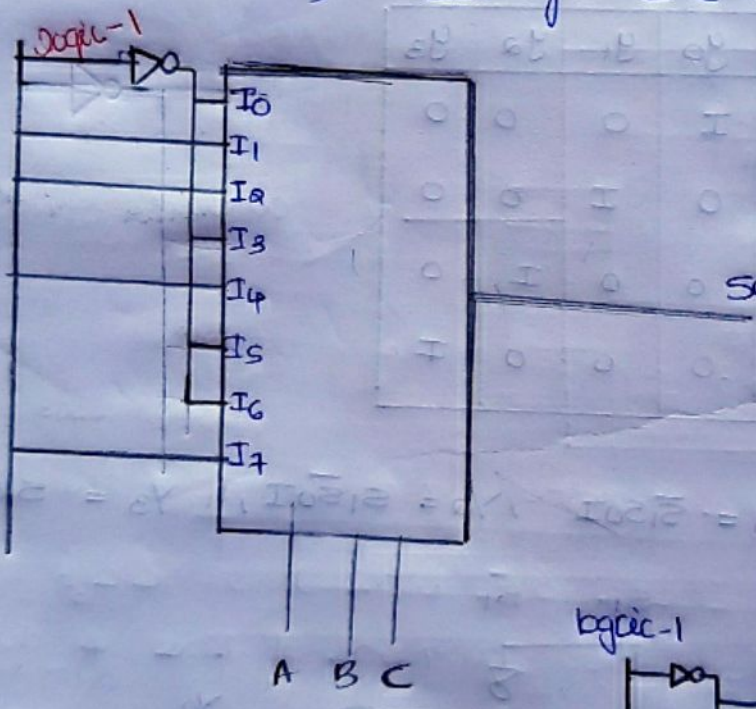


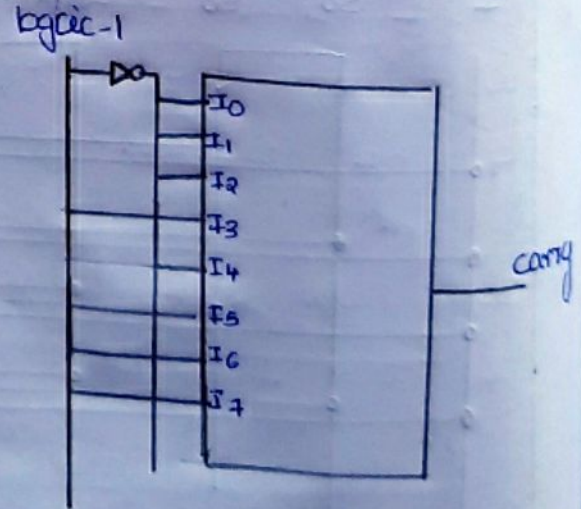
2. Implement a full adder using 8:1 MUX

A).

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$SUM = \Sigma(1, 2, 4, 7)$ $carry = \Sigma(3, 5, 6, 7)$





2 Represent +45, -45 in 1's complement and 2's complement form

A). Binary (45) = 101101

8 bit representation (45) = 00101101

1's complement (45) = 00101101

2's complement (45) = 00101101

2's complement (-45) = 11010010

2's complement (-45) = 11010011

2	45
2	22 - 1
2	11 - 0
2	5 - 1
2	2 - 1
	1 - 0