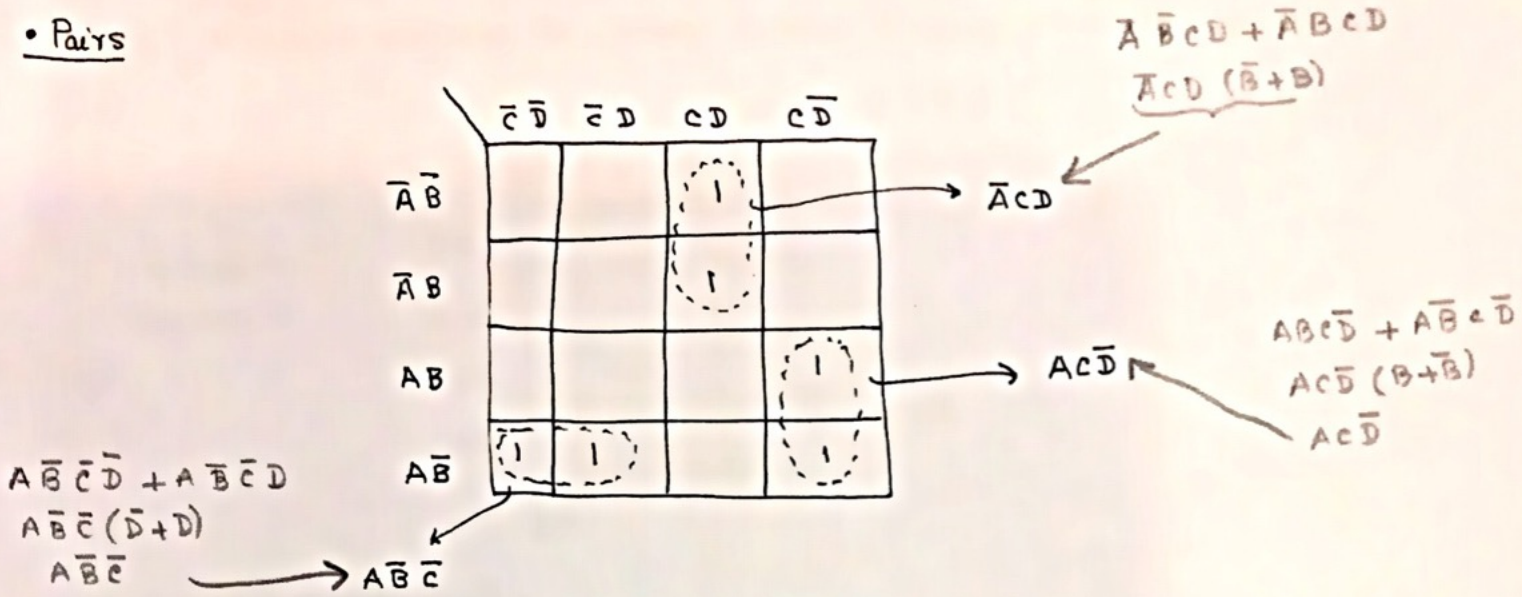
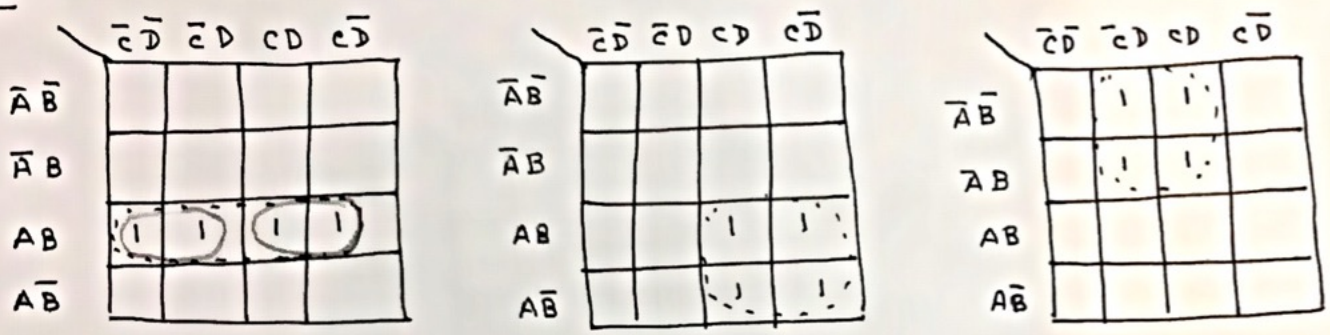
• Pairs



• Quads



$AB\bar{C} + ABC$

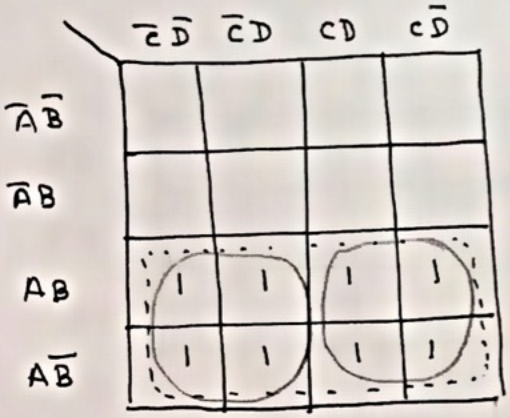
$= AB(\bar{C}+C)$

$= AB$

AC

$\bar{A}D$

• Octets



Sum of two quads

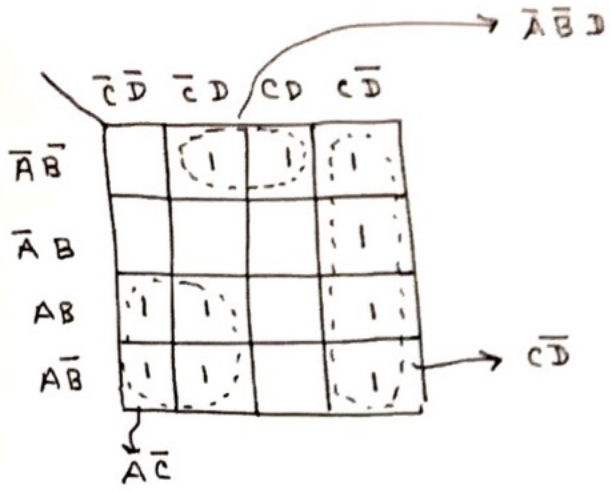
$A\bar{C} + AC$

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

$= A$

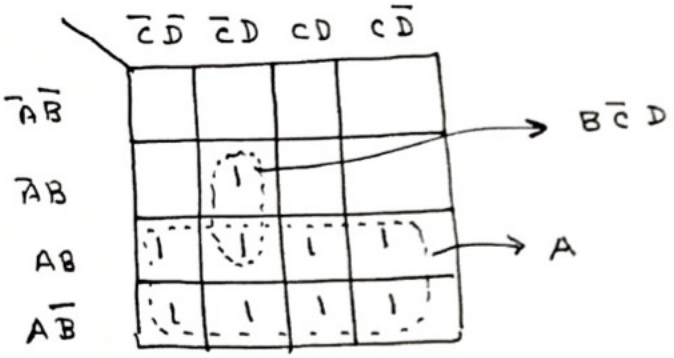
- Example : Obtain Simplified expression for Boolean function Y using Karnaugh's map shown

K-map for function Y



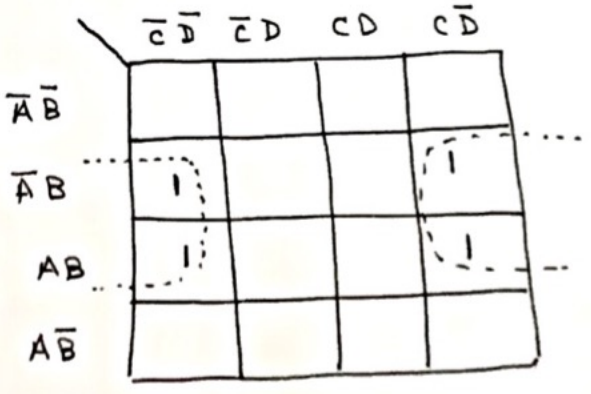
$\therefore Y = \bar{A}\bar{B}D + A\bar{C} + C\bar{D}$

- Overlapping Groups :



$\therefore Y = A + B\bar{C}D$

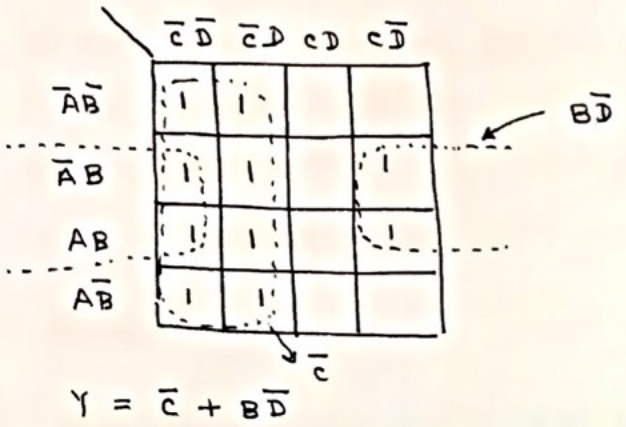
- Rolling the map :



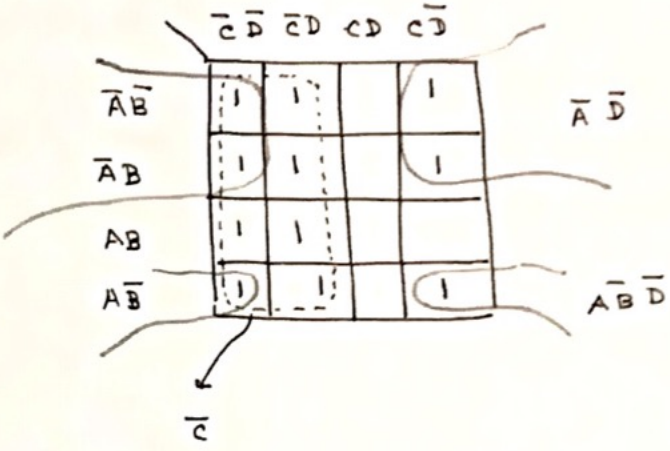
$Y = B\bar{D}$

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

• More examples: Find Y by simplifying the K-maps shown



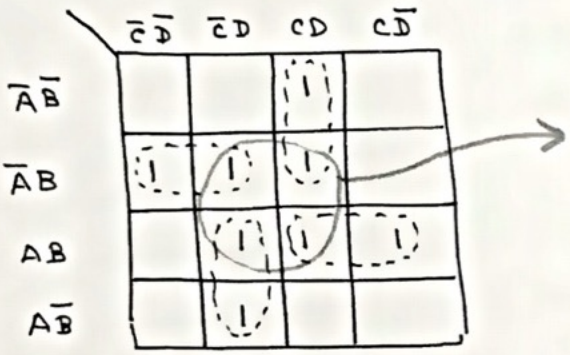
$Y = \bar{C} + B\bar{D}$



$\therefore Y = \bar{C} + \bar{A}\bar{D} + A\bar{B}\bar{D}$

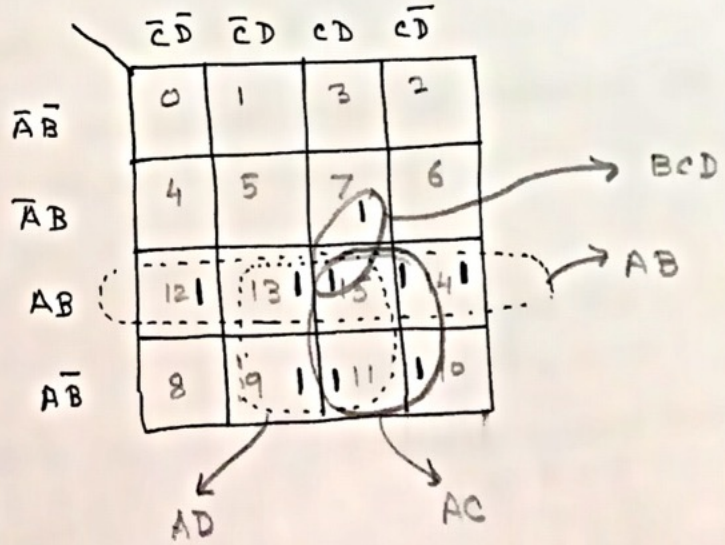
or $Y = \bar{C} + \bar{A}\bar{D} + \bar{B}\bar{D}$

• Eliminating redundant groups:



The quad is redundant as all its 1's have been included in 4 'pairs'.

• Example: Simplify $Y = F(A,B,C,D) = \sum m\,(7,9,10,11,12,13,14,15)$



$Y = AB + AC + AD + BCD$

- **Don't Care conditions :**

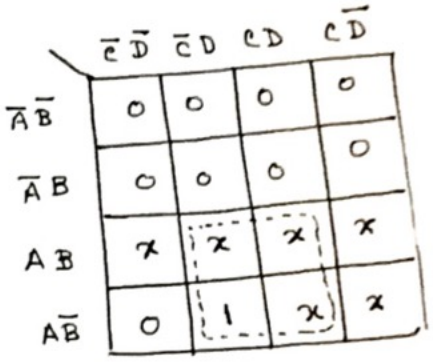
In some digital circuits, certain combinations of inputs are needed or are of no consequence to the desired outputs, these are called dont care conditions and are marked as 'x' in the truth table. This 'x' can be taken as either 0 or 1 depending upon which way it leads to a simplified Boolean function.

Example :

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

all 'x' are dont care conditions

K- map



$\therefore Y = AD$

(Otherwise, it would have been $Y = A\bar{B}\bar{C}D$)

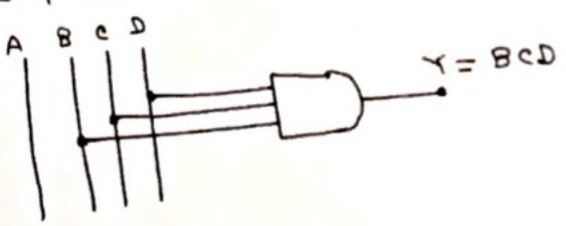
- **Example :** Give the simplest circuit to implement the following function:

$$Y = F(A, B, C, D) = \sum_m (7) + d(10, 11, 12, 13, 14, 15)$$

K- map of function Y

If we combine one 'x' with '1' and make a pair then $Y = BCD$

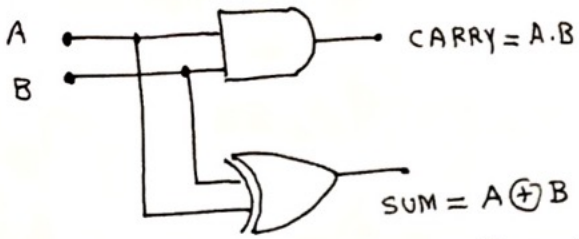
(otherwise Y would have been $\bar{A}BCD$)



10. Half adder

Truth Table

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



$= \bar{A}B + A\bar{B}$ ∴ $SUM = \bar{A}B + A\bar{B} = A \oplus B$

$CARRY = A.B$
(C_o)

11. Full-adder

Truth table

A	B	C_o	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

$SUM = \bar{A}\bar{B}C_o + \bar{A}B\bar{C}_o + A\bar{B}\bar{C}_o + ABC_o$

$CARRY = \bar{A}BC_o + A\bar{B}C_o + AB\bar{C}_o + ABC_o$

ex-NOR ex-OR

$SUM = C_o[\bar{A}\bar{B} + AB] + \bar{C}_o[\bar{A}B + A\bar{B}]$ → assume D

$= C_o\bar{D} + \bar{C}_o D$

$= C_o \oplus D$

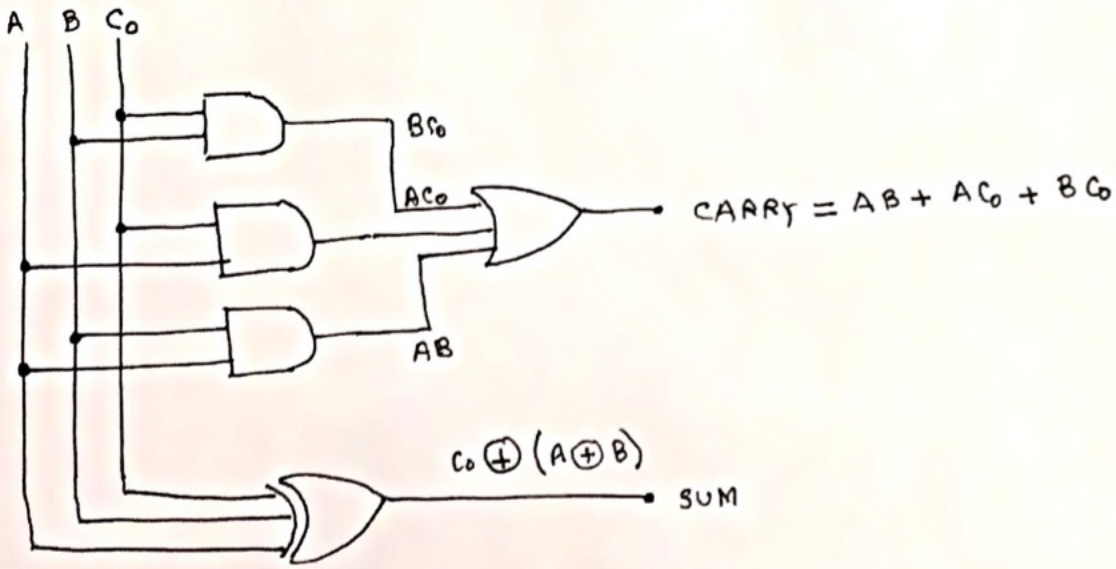
$= C_o \oplus (A \oplus B)$

$CARRY = (\bar{A}B + A\bar{B})C_o + AB(C_o + \bar{C}_o)$

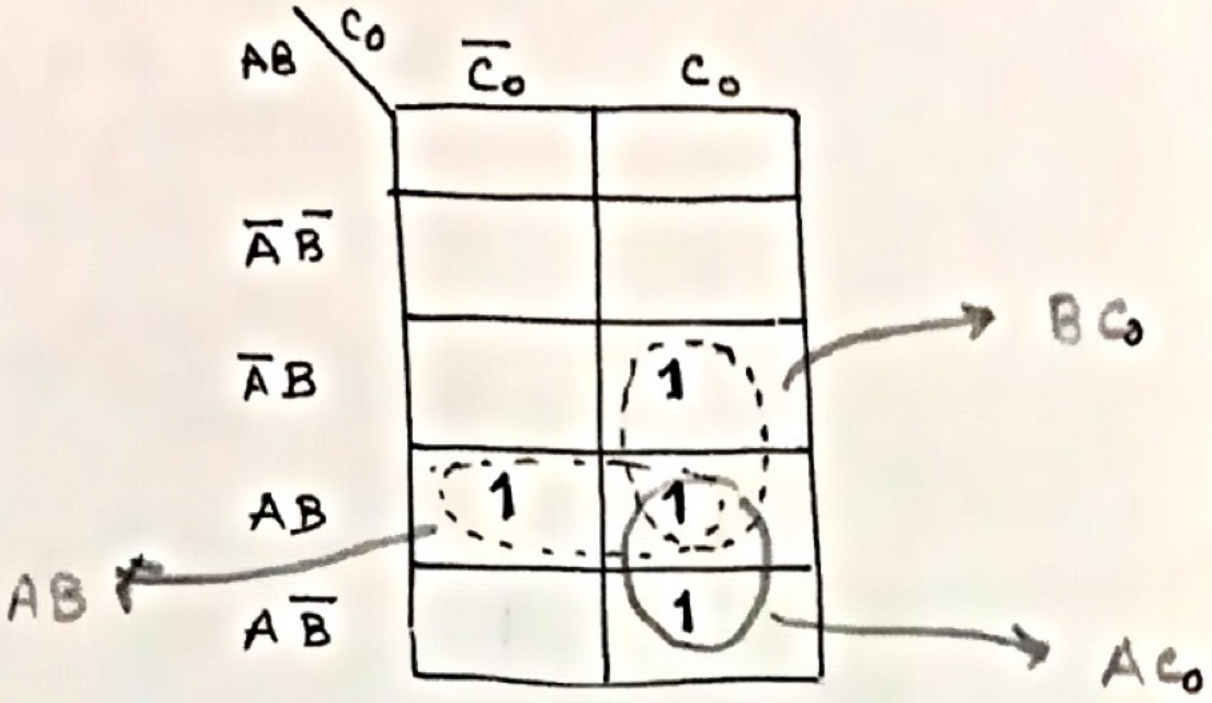
$= \bar{A}BC_o + A\bar{B}C_o + AB$

$= AB + BC_o + AC_o$ (use Karnaugh's map)

See next page K-map



K-MAP for CARRY

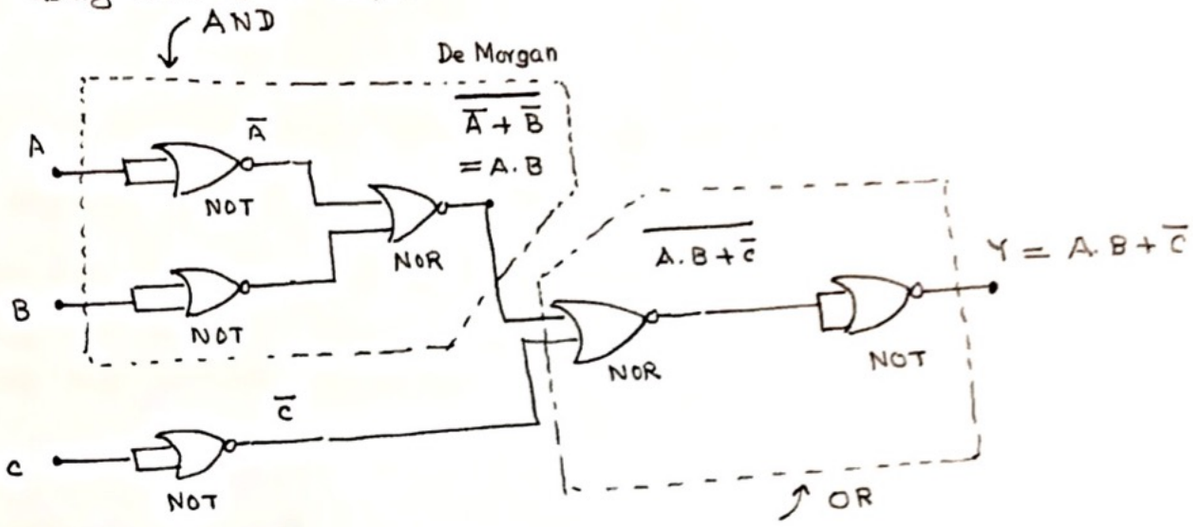


$\therefore$ CARRY $= AB + BC_0 + AC_0$

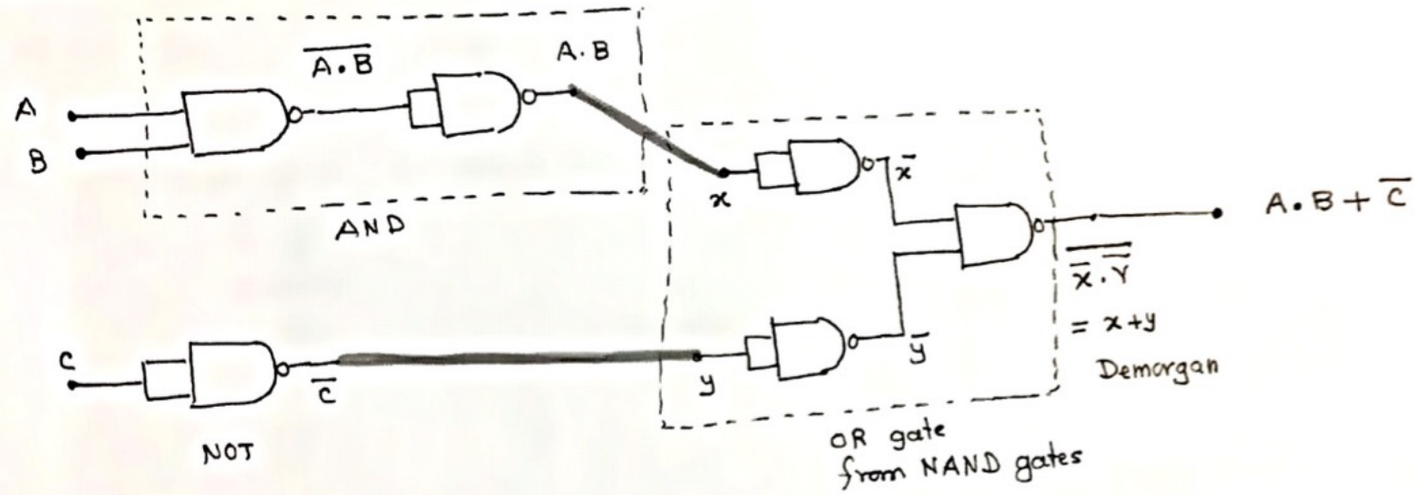
Problem : Realise $Y = A\cdot B + \overline{C}$ using only one types of gates.

WE can use only universal gate ie NOR or NAND to make circuits with only 1 gate

Method 1 Using NOR gates only :



Method 2 Using NAND gates only :

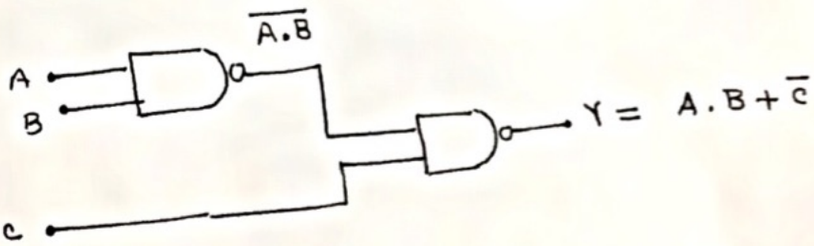


Method 3

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

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

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



Note : The last circuit can be obtained from second one by removing cascades of two-NOT-gates one after the other which are redundant !

Method 4 :

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

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

Circuit resulting from this expression would be same as in method 1 .

13. Sequential Circuits : Flip flops (FF)

- A flip flop is a circuit called 'bistable' (having two stable states)
- A flip flop has memory, storing one bit information

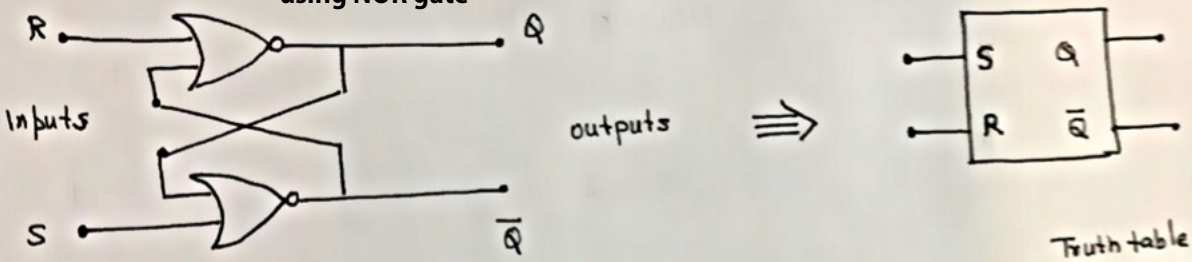
  When flip flop has its output as HIGH ($\approx +V_{cc} = +5\,V$) it stores '1'

  When flip flop has its output as LOW ($\approx$ 0 volts), it stores '0'.
- A flip flop has 'feedback' connections.

NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

- **SR flip-flop :** **This is a SR latch(Basic storage element) using NOR gate**



('R' stands for reset; 'S' stands for setting)

↓ making output '0'

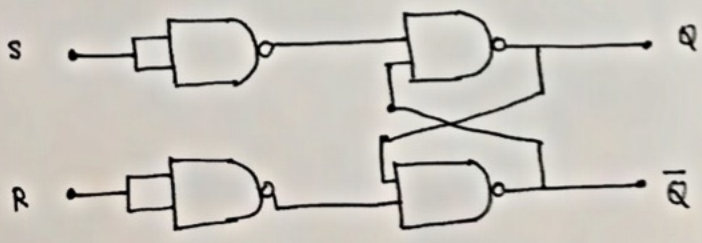
↓ making output '1'

Truth table

R	S	Q_{n+1}	
0	0	Q_n	previous state
0	1	1	'set' to '1'
1	0	0	'reset' to '0'
1	1	? (forbidden)	indeterminate

next state

Same circuit can also be realised by NAND gates :



NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

This circuit too is characterised by the same truth table. (verify yourself!)

R	S	Q_n	$\overline{Q_n}$	Q_{n+1}	$\overline{Q_{n+1}}$	
0	0	0	1	0	1	$Q_{n+1} = Q_n$
0	0	1	0	1	0	
1	0	0	1			$Q_{n+1} = 0$
1	0	1	0			
0	1	0	1			$Q_{n+1} = 1$
0	1	1	1			
1	1	0	1			Q_{n+1} and $\overline{Q_{n+1}}$ don't remain complimentary
1	1	1	0			

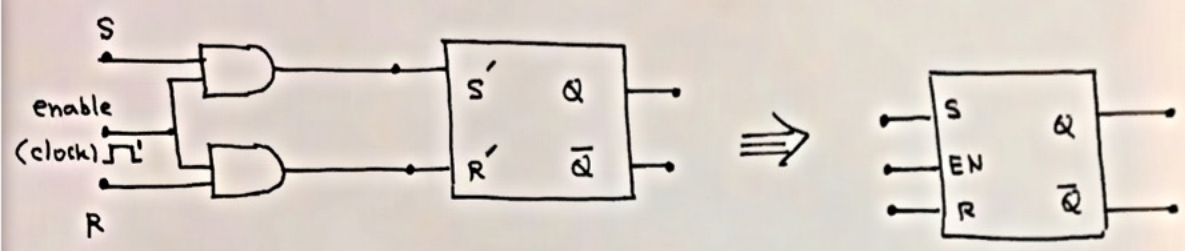
(Not in order !)

AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

If A = 1 (fixed)
then
Y = A · B = B

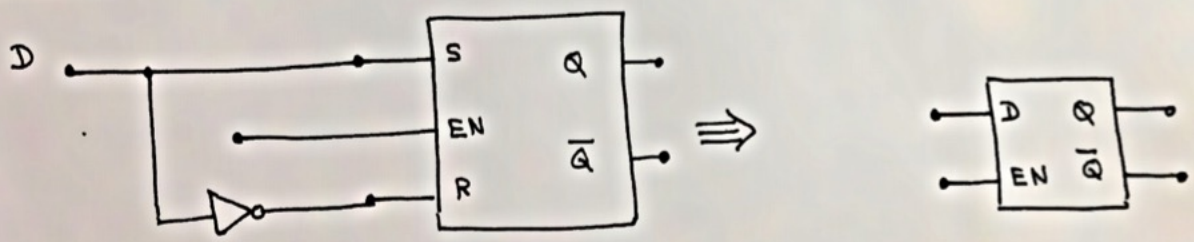
- **Clocked SR flip flop: (with enable)**



Truth table

EN	S	R	Q_{n+1}	
1	0	0	Q_n	no change (previous state continues)
1	1	0	1	
1	0	1	0	
1	1	1	?	forbidden
0	x	x	Q_n	no change (previous state continues)

- **D flip flop:**



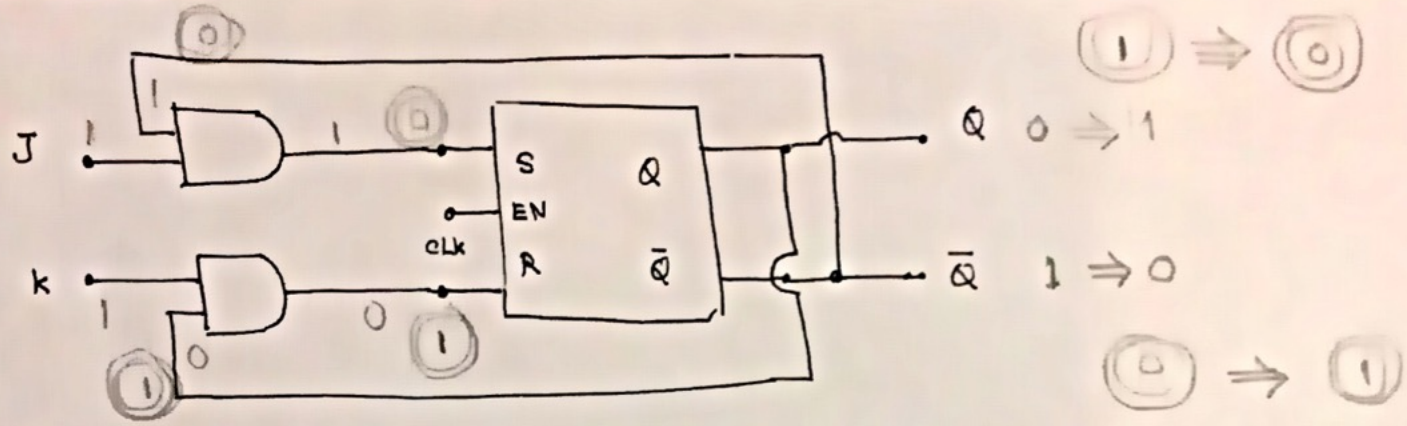
↓D

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	? ?

Truth Table

EN	D	Q_{n+1}	
0	x	Q_n	last state
1	0	0	reset
1	1	1	set

• Jk Flip flop :



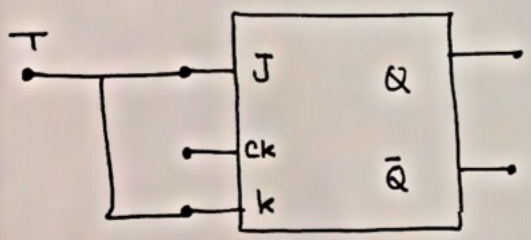
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

J	K	Q_n	Q_{n+1}
1	1	0	1
1	1	1	0

$\therefore Q_{n+1} = \bar{Q_n}$

$\therefore Q_{n+1} = \bar{Q_n}$

• Toggle Flip flop (T flip-flop)



ck = 1

T	Q_{n+1}
0	Q_n
1	$\bar{Q_n}$

On appearance of each '1' at the input, the output would 'toggle' (becoming '1' from '0' or becoming '0' from '1').