PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

UNIT - 03 : Digital Electronics

* Boolean Algebra :- Boolean Algebra & Logic gates.

* Basic theorems and Properties of Boolean algebra

* Boolean functions, Canonical & Standard form.

* Digital logic gates

  → Combinational logic Circuits

  → Sequential logic Circuits : Half adder
  full adder

  → Multiplexer & Demultiplexer

  → Flip Flops : RS, D, T, JK

  → Registers : SISO

  → Counters : 3 bit Synchronous and Asynchro
  -nous counter.

Faculty Signature

Boolean Algebra & logic gates:

* Digital Electronics :- The branch of electronics where the circuits Process the data which contains binary Values 1s and 0s (logical levels)

* Logic 1 is referred as HIGH Voltage / TRUE / ON state
* Logic 0 is referred as LOW Voltage / FALSE / OFF state.
* A Binary digit either "0" (or) "1" is called a "bit"
* Digital information is stored using a Series of binary Values of 1s & 0s.
* All digital Systems stores & process the data in terms of 0s & 1s.

  Ex:- Digital telephones, computers, digital Camera etc

* Boolean algebra is a branch of mathematics which is used to analyse and Simplify logic circuits / digital circuits.
* Digital circuits are constructed by using logic gates.
* Logic gates are building blocks of digital Systems Performs logical functions based on boolean algebra.

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

* Logic gates have one / more inputs but only one output.

* If any logic gate has "n" number of inputs then it can have 2^n possible input states.

* Logic gates are implemented using diodes (or) transistors which acts as electronic switches.

* There are two categories of logic gates
  i) Basic gates : AND, OR, NOT
  ii) Derived gates : NAND, NOR, EXOR & EXNOR gates.

* AND Gate :-

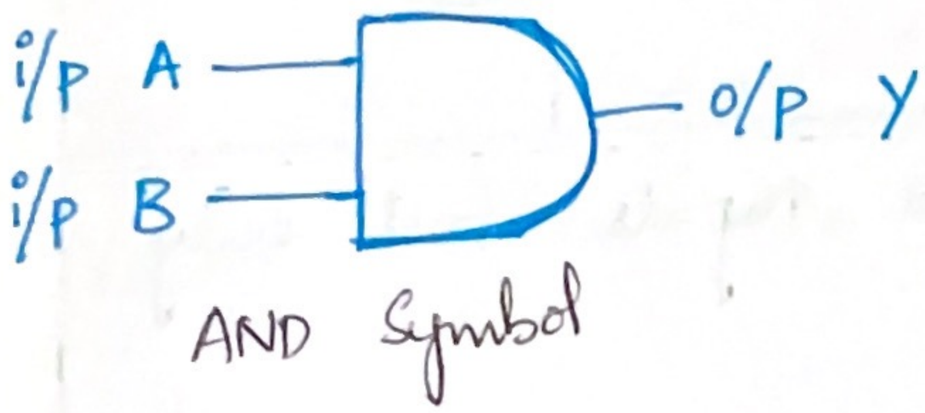
> It is an electronic circuit which gives output as logic 1 (HIGH) iff all inputs are at logic "1". otherwise output is logic "0" (LOW).

> Boolean algebra representation for AND gate is

$Y = A.B$

Faculty Signature

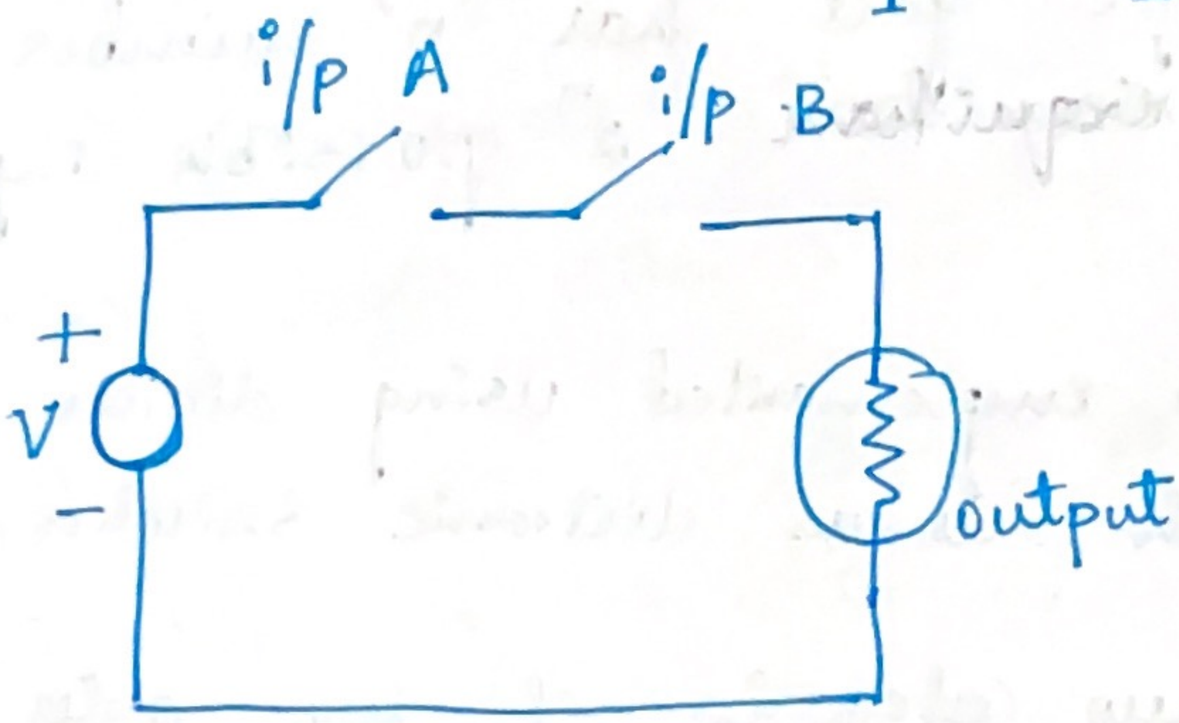
* AND gate is represented as "two switches connected in series".



AND Symbol

TRUTH TABLE

Inputs		output
		Y
A	B	
0	0	0
0	1	0
1	0	0
1	1	1



Circuit

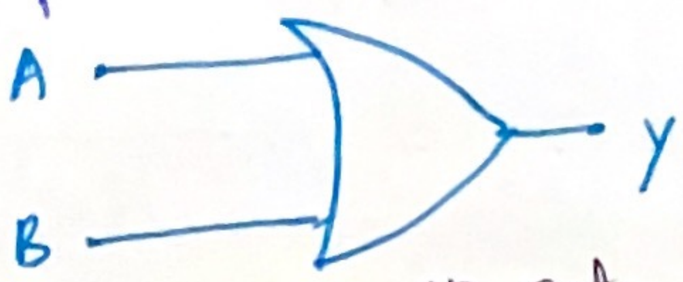
* OR Gate:-

> It is an electronic digital circuit which gives output as 1 when either of the input is logic 1 and output as logic "0" iff all inputs are logic "0".

> Boolean algebra representation for OR gate is

$$Y = A + B$$

> OR gate is represented as two switches connected in parallel

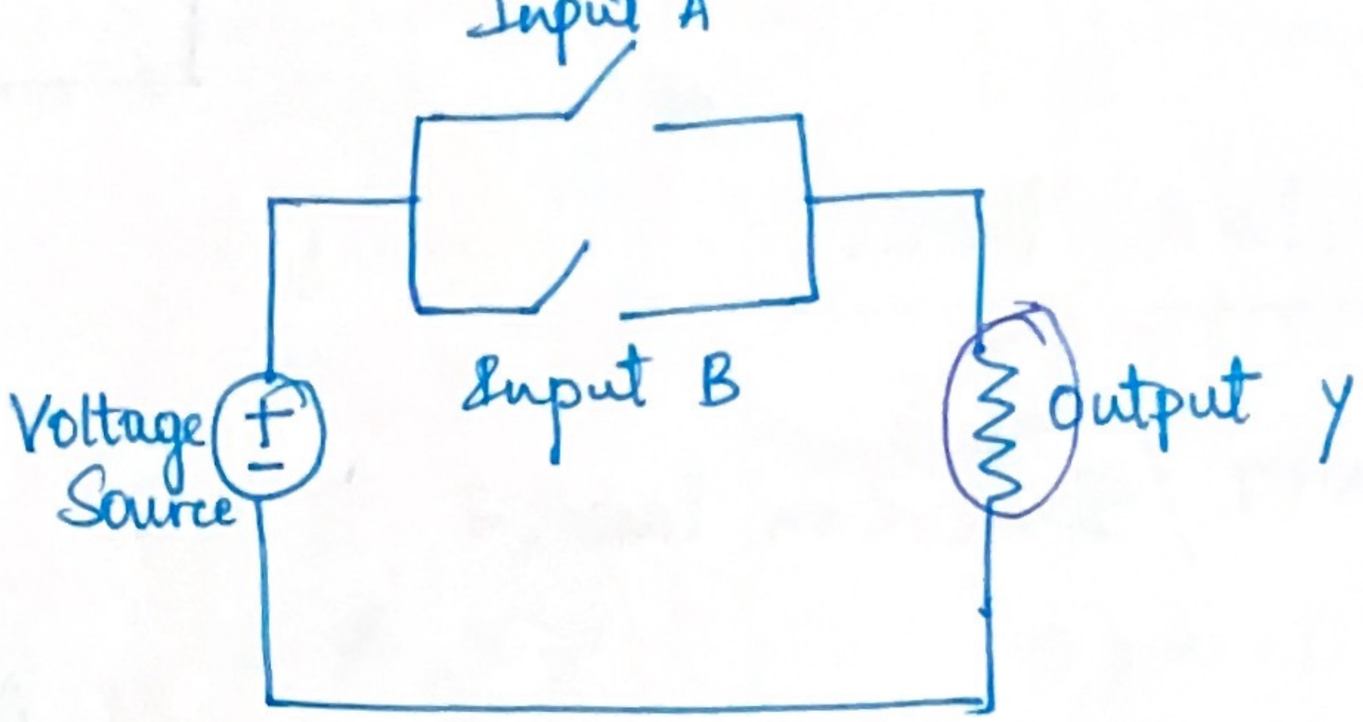


OR gate Symbol

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

TRUTH TABLE

Inputs		output
A	B	Y
0	0	0
0	1	1
0	1	1
1	0	1
1	1	1



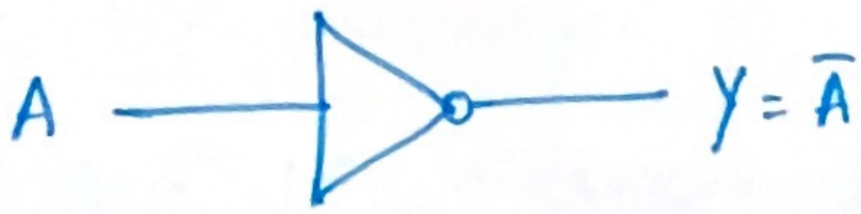
* NOT Gate

> It is a Single input - Single output gate which peforms inversion of the input signal. Hence it is also called as Inverter gate.

> Boolean expression for NOT gate is

$$Y = \bar{A}$$

> NOT Gate is represented by connecting a switch parallel to load.



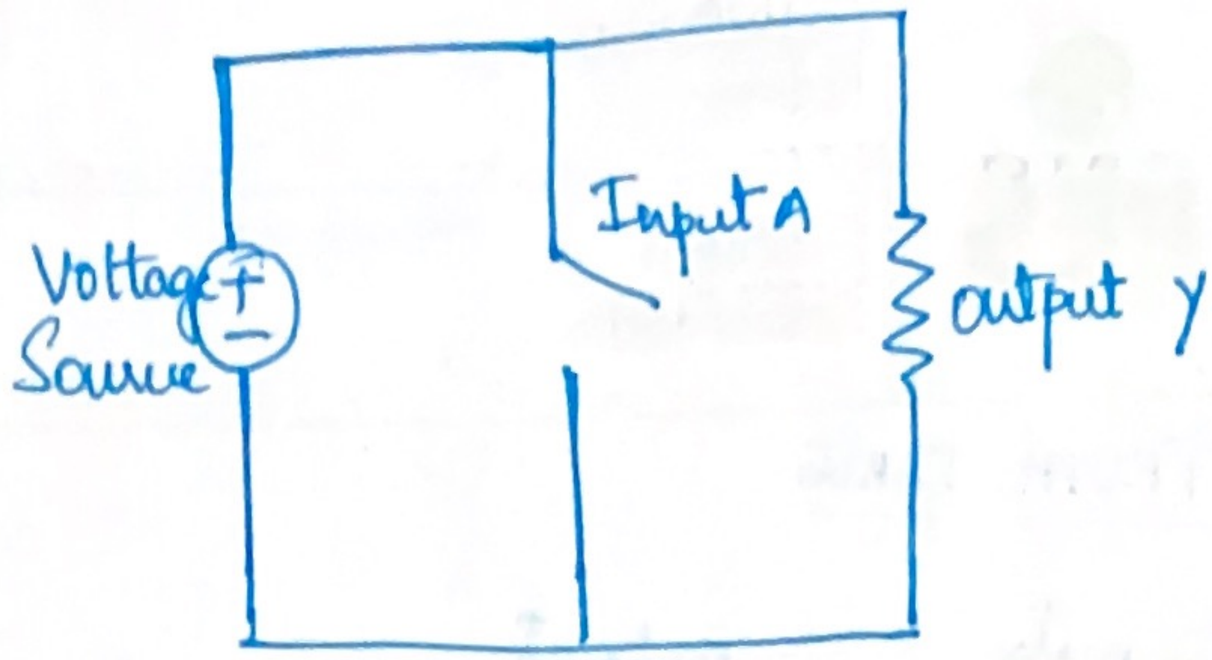
NOT gate Symbol

Faculty Signature

TRUTH TABLE

Input A	output Y
0	1
1	0

CIRCUIT



Boolean laws

1] NOT / Inversion law :-

$\bar{1} = 0$

$\bar{0} = 1$

$\therefore$ If $A = 1 \Rightarrow \bar{A} = 0$

$A = 0 \Rightarrow \bar{A} = 1$ and

$\therefore (A')' = A$

2] AND law :-

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

$A \cdot A' = 0$

3] OR law

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + A' = 1$

4] Principle of duality

In Boolean Algebra, One type of expression can be converted into another type of expression by replacing "0" with "1" and "1" with "0" and also by replacing (+) with (.) and (.) with (+).

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

AND (·)	OR (+)
0 · 0 = 0	0 + 0 = 1
0 · 1 = 0	1 + 0 = 1
1 · 0 = 0	0 + 1 = 1
1 · 1 = 1	0 + 0 = 0

Expression	Dual Expression
$\overline{0} = 1$	$\overline{1} = 0$
0 · 1 = 0	1 + 0 = 1
A · 0 = 0	A + 1 = 1
A · B = B · A	A + B = B + A
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$

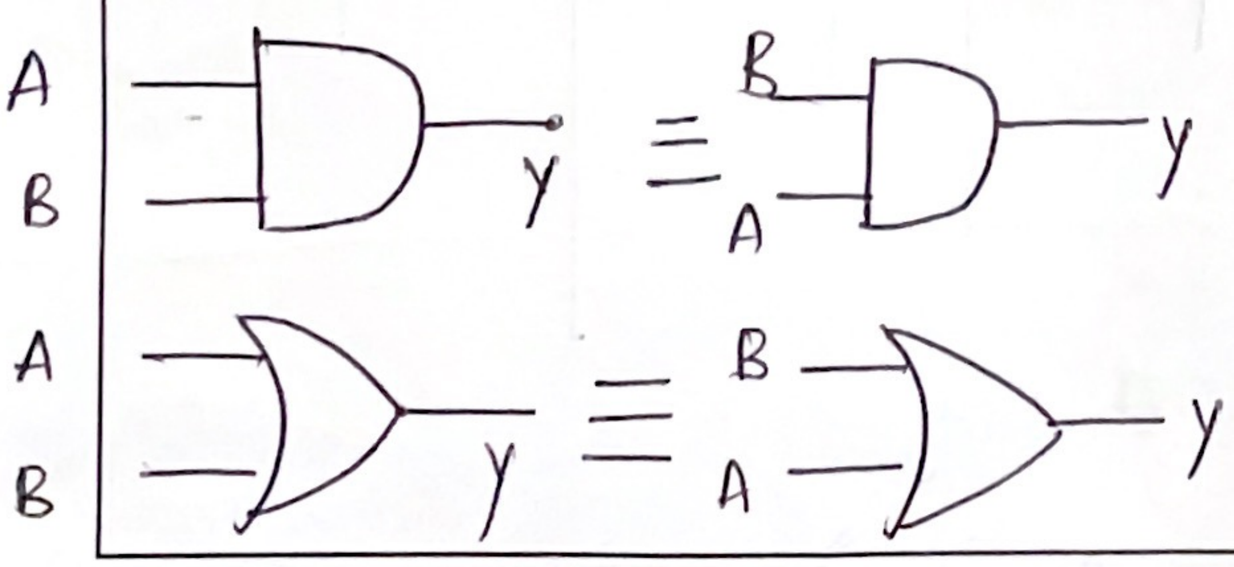
5] Basic Theorem and Properties of Boolean algebra.

Boolean laws and theorems are used to simplify Boolean expression, and reduce the number of logic gates.

i] Commutative law:

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$



Faculty Signature

Proof:

A	B	A·B	B·A	A+B	B+A
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

$C_3 = C_4 \quad \therefore A \cdot B = B \cdot A$

$C_5 = C_6 \quad \therefore A + B = B + A$

ii] Associative law:

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

Proof:

A	B	C	A+B	(A+B)+C	B+C	A+(B+C)
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

From the Principle of duality.

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

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

iii] Distributive law:

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

Proof:

A	B	C	B.C	A+B	A+C	A+B.C	(A+B).(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Faculty Signature

Duality of distributive law:

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

A	B	C	B+C	A.(B+C)	A.B	A.C	(A.B) + (A.C)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

6] De - Morgan's Theorem

(i) The complement of Sum of two Variables is equal to Product of complement of individual Variables.

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

ii) The complement of Product of two Variables is equal to Sum of complement of individual Variables.

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

Prog + output (Hard Copy)

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Proof

A	B	A+B	$\overline{A+B}$	A'	B'	A.B	$\overline{A.B}$	A'+B'	A'.B'
0	0	0	1	1	1	0	1	1	1
0	1	1	0	1	0	0	1	1	0
1	0	1	0	0	1	0	1	1	0
1	1	1	0	0	0	1	0	0	0

7] Absorption theorem :

i) $A + AB = A$

$A + AB$

$= A(1+B)$

$= A(1)$ ∵ OR law

$= A$

$= A$ //

ii] $A(A+B) = A$

$A(A+B)$

$= A.A + A.B$

$= A + AB$ (∵ AND law)

$= A(1+B)$ (∵ OR law)

$= A$ //

iii) $A + \bar{A}B = A + B$

$A + \bar{A}B$

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

$= 1.(A+B)$ (OR law)

$= A+B$ (AND law)

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

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

$= A\bar{A} + AB = 0 + AB$

$= AB$

Faculty Signature

8] Consensus Theorem

Ex:

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

$$LHS = AB + \bar{A}C + BC$$

$$= AB + \bar{A}C \quad (\because BC \text{ is redundant term})$$

Boolean functions, Canonical form & Standard Canonical form

Boolean function:

* It is an algebraic expression consist of binary variables, constants (0 (or) 1) and logical operation symbol.

* Boolean function is evaluated to logic 1 (or) logic 0 for the given value of binary variables.

* Boolean function can be represented in a truth table and it can be implemented as digital circuits constructed using logic gates.

Ex:- $F = A + B'C$

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

If A=0, B=0 and C=1

$F = 0 + 1 \cdot 1 = 1$ //

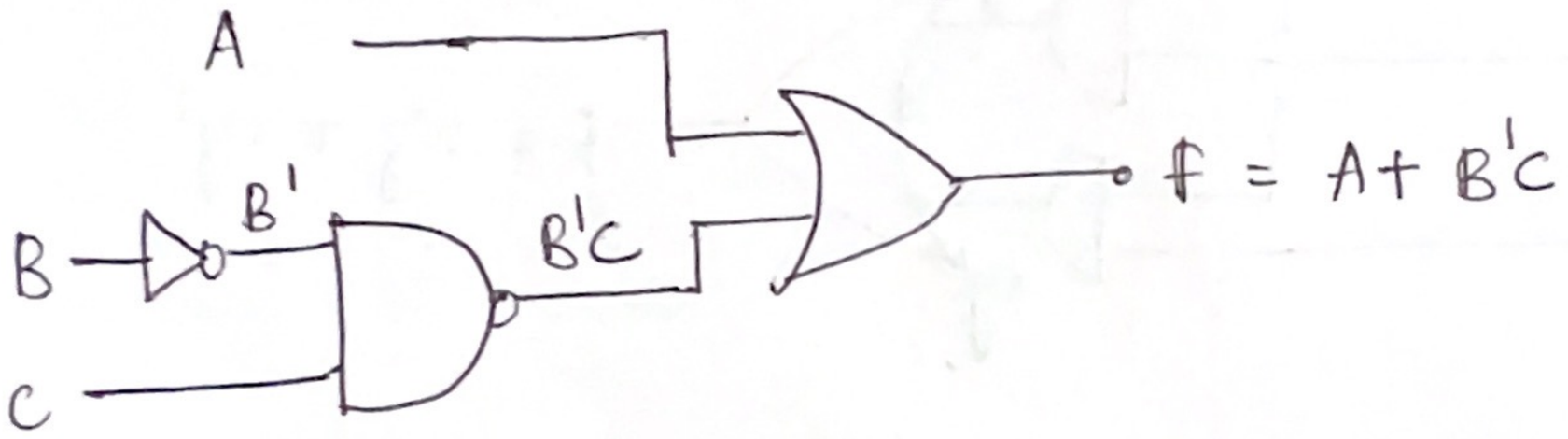
If A=0, B=1 and C=0

$F = 0 + 0 \cdot 0 = 0$

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

∴ f can be either "0" (or) "1" Based on the Value of A, B & C.

The given Boolean expression can be constructed using 1 NOT gate, 1 OR gate and 1 AND gate.

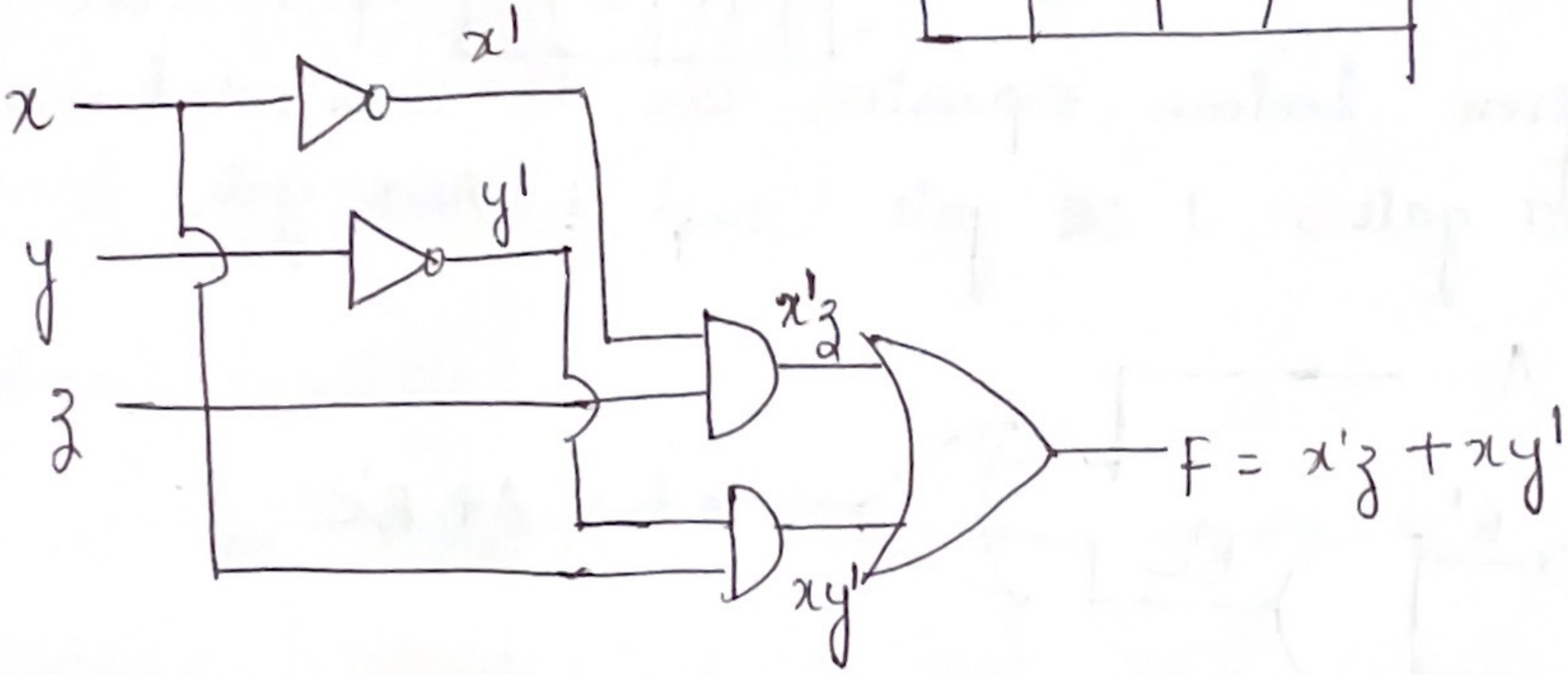


Faculty Signature

Simplify and realize the given Boolean function using basic gates.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$F = x'y'z + x'yz + xy'$$
$$x'z(y'+y) + xy'$$
$$x'z(1) + xy'$$
$$x'z + xy'$$



Sum Of Products (SOP) :-

* SOP is a boolean expression which contains Sum of (ORing of) Prooduct terms (ANDing of one or more literals)

Example $F = y' + xy + x'yz'$

ORing

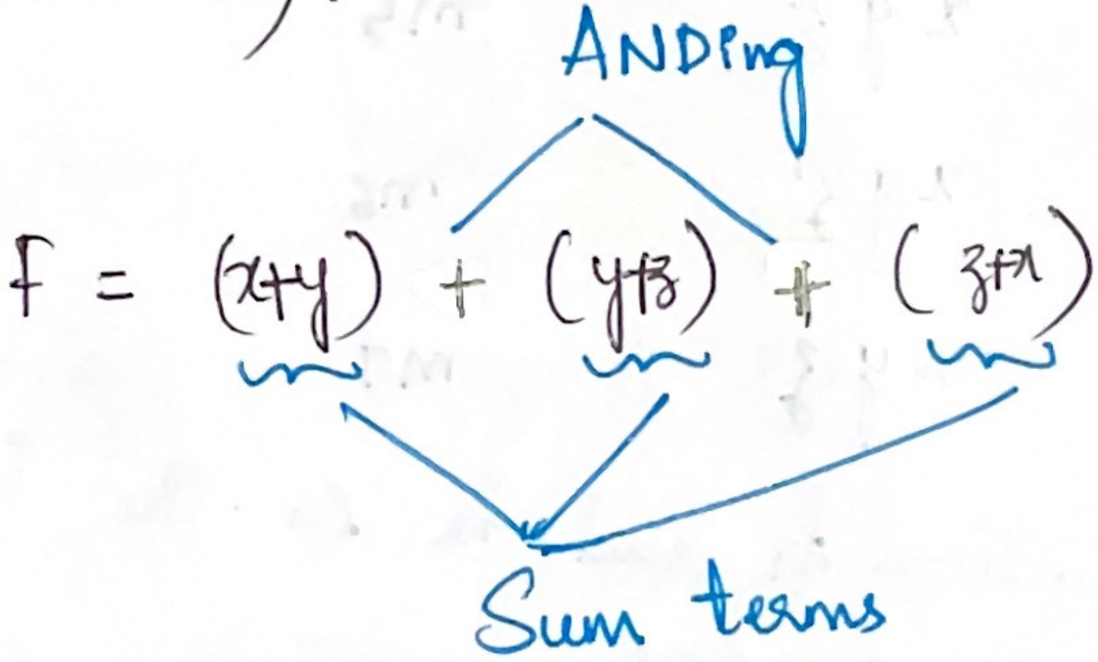
AND terms (or) Product terms

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Product of Sum : POS

* POS is a boolean expression which contains Product of (ANDing) of Sum terms (ORing of one or more literals).

Example:



Canonical SOP form:-

* It is a SOP boolean expression where each product term should contain all the literals either in true (or) complemented form.

* Here each product term is called "Minterm".

* Ex: f = xy + x'z + yz (NOT in Canonical form)

F = xyz + x'y'z + x'yz (In Canonical form because all terms contains all literals in true / Complemented form)

Faculty Signature

Truth table (three Variable function):

Minterm

x	y	z	Term	Designation
0	0	0	$x'y'z'$	m_0
0	0	1	$x'y'z$	m_1
0	1	0	$x'yz'$	m_2
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4
1	0	1	$xy'z'$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

If x is a Variable Variables in true state is written as "x" and Variable in complimated state is written as "x'".

Consider the function f_1 and f_2 in the truth table and Write Canonical form SOP.

x	y	z	f_1	f_2
0	0	0	0 m_0	0 m_0
0	0	1	1 m_1	0 m_1
0	1	0	0 m_2	0 m_2
0	1	1	0 m_3	1 m_3
1	0	0	1 m_4	0 m_4
1	0	1	0 m_5	1 m_5
1	1	0	0 m_6	1 m_6
1	1	1	1 m_7	1 m_7

Canonical SOP form for f_1

$f_1 = m_1 + m_4 + m_7$

$= x'y'z + xy'z' + xyz.$

$f_2 = m_3 + m_5 + m_6 + m_7$

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

PES UNIVERSITY

Experiment Name:

Expt. No.

Problem Statement:

Date

Canonical POS form:-

* It is a POS boolean expression where all sum terms should contains all the literals in its true (or) complimented form.
* Here each sum term is called "Maxterm".
* Ex: $F = (x+y) \cdot (x'+y+z) \cdot (x+z)$ → Not in Canonical POS form.

$F = (x+y+z') \cdot (x'+y'+z) \cdot (x+y+z)$ → In Canonical POS form. because all sum terms contains all the literals in either true/complimented form.

Truth table:

x	y	z	Term	Maxterm Designation
0	0	0	$x+y+z$	M_0
0	0	1	$x+y+z'$	M_1
0	1	0	$x+y'+z$	M_2
0	1	1	$x+y'+z'$	M_3
1	0	0	$x'+y+z$	M_4
1	0	1	$x'+y+z'$	M_5
1	1	0	$x'+y'+z$	M_6
1	1	1	$x'+y'+z'$	M_7

Faculty Signature

Consider the function f_1 and f_2 in the truth table. find Canonical POS form of f_1 and f_2.

x	y	z	f_1	f_2	
0	0	0	0	0	M0
0	0	1	1	0	M1
0	1	0	0	0	M2
0	1	1	0	1	M3
1	0	0	1	0	M4
1	0	1	0	1	M5
1	1	0	0	1	M6
1	1	1	1	1	M7

Canonical POS form for f_1

$f_1 = M0 . M2 . M3 . M5 . M6$

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

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

Canonical POS form of f_2

$f_2 = M0 . M1 . M2 . M4$

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

$(x'+y+z)$

Find the Minterms for the given expression $F = A+Bc$

Note: Convert the given expression into Canonical SOP form

$F = A + BC$

$= A.1.1 + 1.BC$

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

$= (AB + AB')(C+C') + ABC + A'BC$

$= \underline{ABC} + ABC' + AB'C + AB'C' + \underline{ABC} + A'BC$

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

$= m7 + m6 + m5 + m4 + m3$

$F = \sum (3, 4, 5, 6, 7)$ //

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Find the maxterm for the given expression: $F = (x'+y)(x+z)(y+z)$

Convert the given expression into Canonical POS form.

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

$= (x'+y+zz')(x+yy'+z)(xx'+y+z)$

$= (\underbrace{x'+y+zz'}_{A\ BC})(\underbrace{x+z+yy'}_{A\ BC})(\underbrace{y+z+xx'}_{A\ BC})$

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

$\because$ distributive law.

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

$= M_4 \quad M_5 \quad M_0 \quad M_2$

$\therefore F = \Pi M(0, 2, 4, 5)$

Faculty Signature

Other logic gates (Derived gates)

1] XOR Gate :

* It Perfoms modulo sum operation without including carry.

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=0$

T.T

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

* The output of XOR gate is logic 1 if odd number of inputs are at logic 1. and output is logic 0 iff Even no of i/ps are.

For 2 input XOR gate output is logic 1 when Two inputs are unequal. and output is logic 0 if both the inputs are same.

* Boolean Expression for XOR gate.

$Y = A'B + AB'$ (Using Canonical SOP)

$Y = A \oplus B$.

Prove that $A'B + AB' = A \oplus B$

$AB' + A'B$

i) $A=0$ & $B=0$

$0.1 + 1.0 = 0+0 = \boxed{0}$

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

ii] A=0, B=1

0.0 + 1.1 = 0 + 1 = [1]

iii] A=1, B=0

1.1 + 0.1 = 1 + 0 = [1]

Hence

$AB' + A'B = A \oplus B$

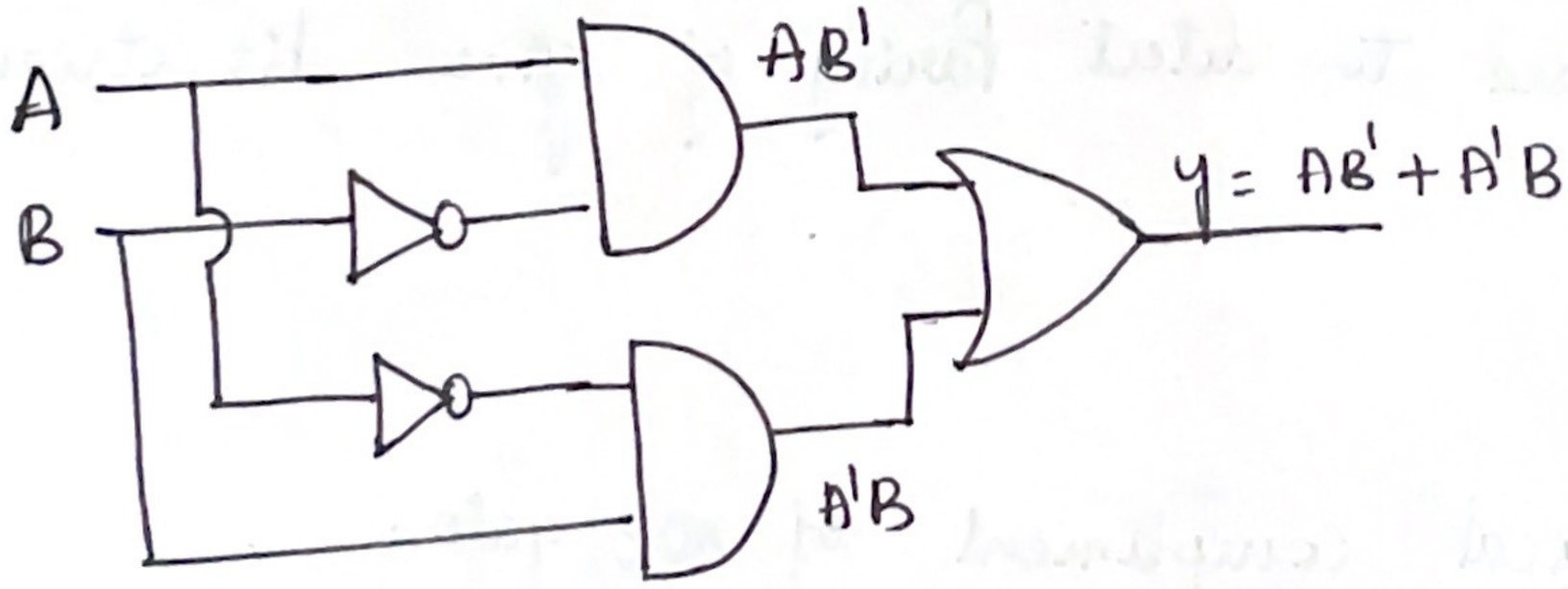
iv] A=1, B=1

1.0 + 0.1 = 0 + 0 = [0]

1.

Implementation of XOR gate using basic gates.

$y = AB' + A'B$.



Symbol for XOR gate

Faculty Signature

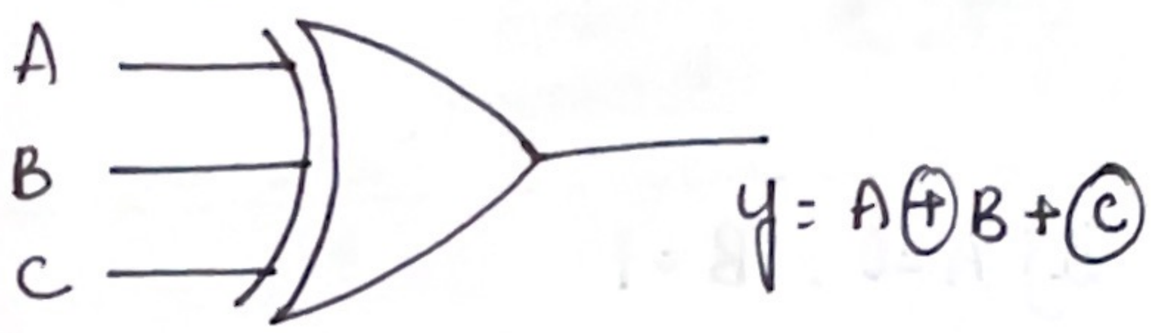
3 input XOR gate

$Y = A \oplus B \oplus C$ → Boolean Expression.

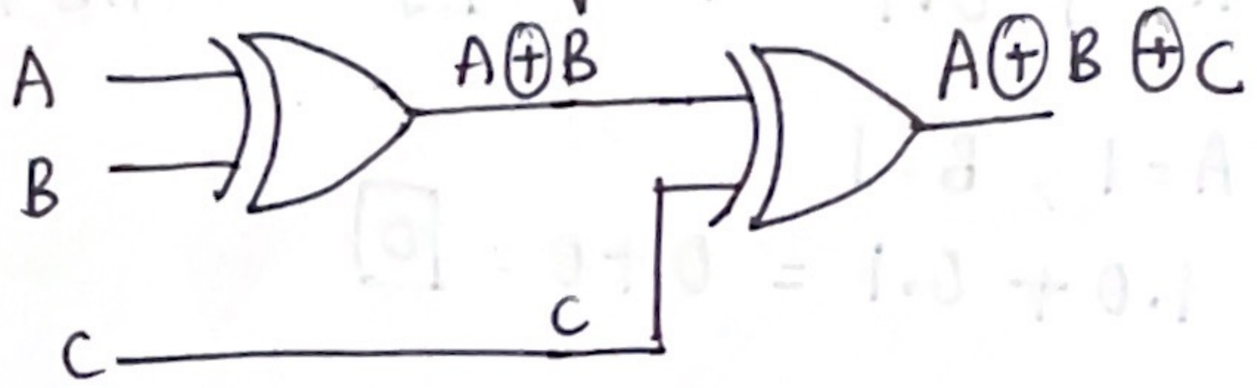
T.T

A	B	C	Y
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

Symbol



3 input XOR gate using 2 input XOR gate.



XOR gate is used to detect Parity of given bit stream & Error detection.

2] XNOR gate:

* It is a logical compliment of XOR gate.

* output of XNOR gate is logic 1 if Even no of inputs are high and is logic 0 if odd no of inputs are high.

* For 2 input XNOR gate output is high if two inputs are same and is low if two inputs are unequal.

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

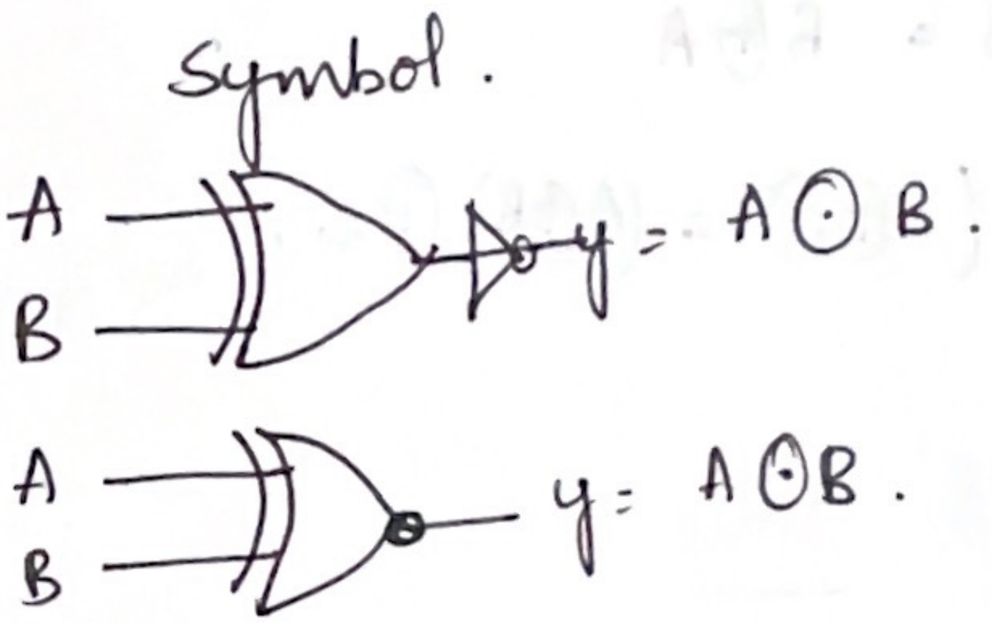
$y = \overline{A \oplus B}$; Boolean Equation . $\boxed{y = A \odot B}$

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

T.T

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

Symbol .



Boolean Equation for XNOR gate

$y = AB' + A'B$

$= (AB')' \cdot (A'B)'$

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

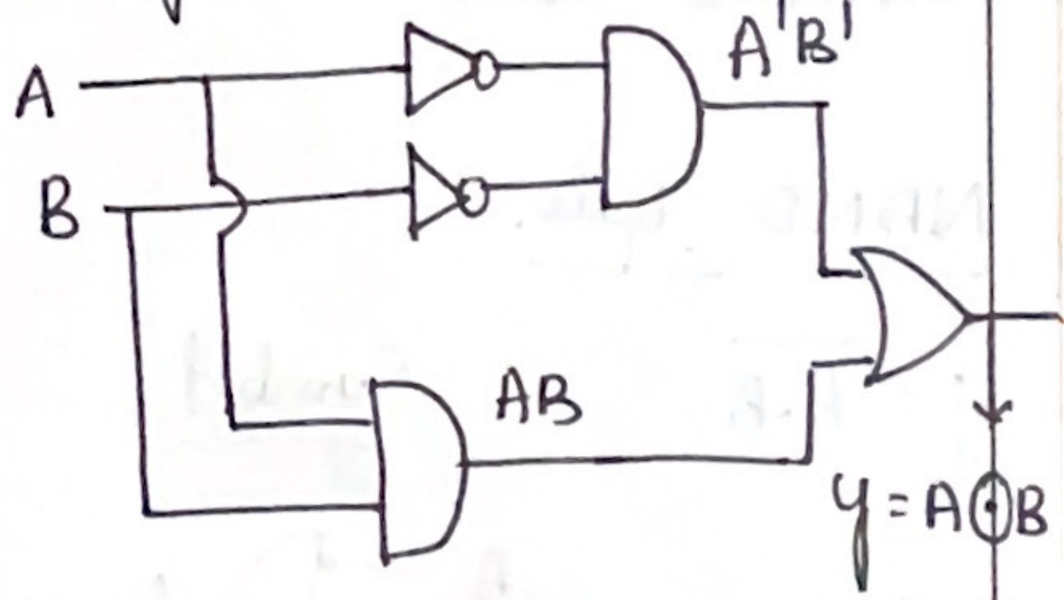
$= (A' + B) \cdot (A + B')$

$= \cancel{A'A}^{0} + A'B' + AB + \cancel{BB}^{\to 0}$

$= A'B' + AB$ //

Implementation of XNOR gate using

$y = A'B' + AB$



* XNOR gate is used in Equality detector .

Faculty Signature

Properties of XOR gate :-

(1) $A \oplus 0 = A$

(2) $A \oplus 1 = A'$

(3) $A \oplus A = 0$

(4) $A \oplus A' = 1$

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

(6) $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

Universal Gate :-

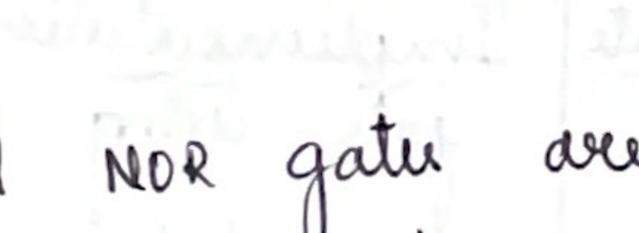
* Any gate can be constructed using either NAND (or) NOR gate are called universal gate.

* NAND and NOR gate are easier to fabricate with electronic components! Hence they are used in all ICs.

NAND Gate:

$y = \overline{A.B}$

Symbol



III

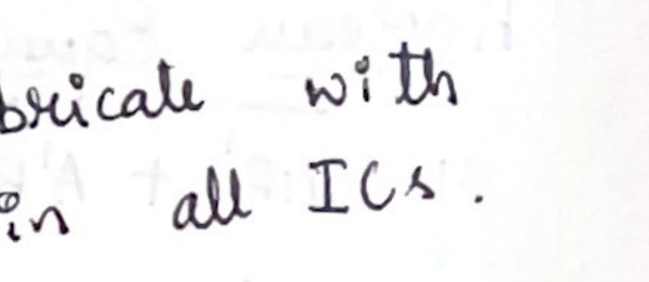
T.T

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate :

$y = \overline{A+B}$

Symbol



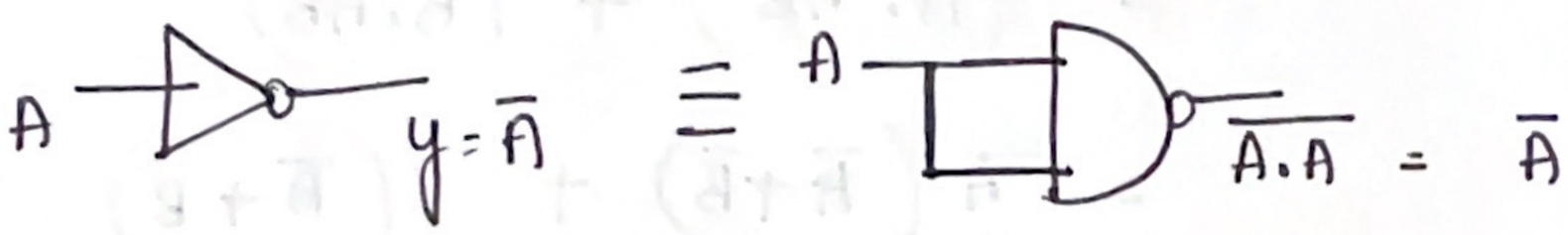
T.T

A.	B.	y
0	0	1
0	1	0
1	0	0
1	1	0

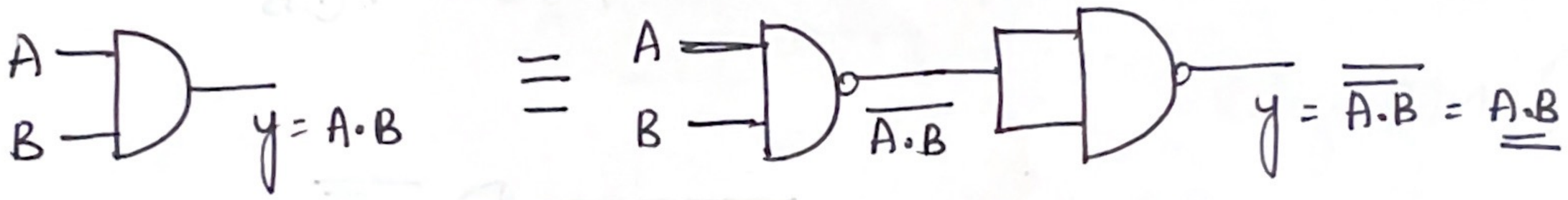
PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Realization of logic gates using NAND gates:

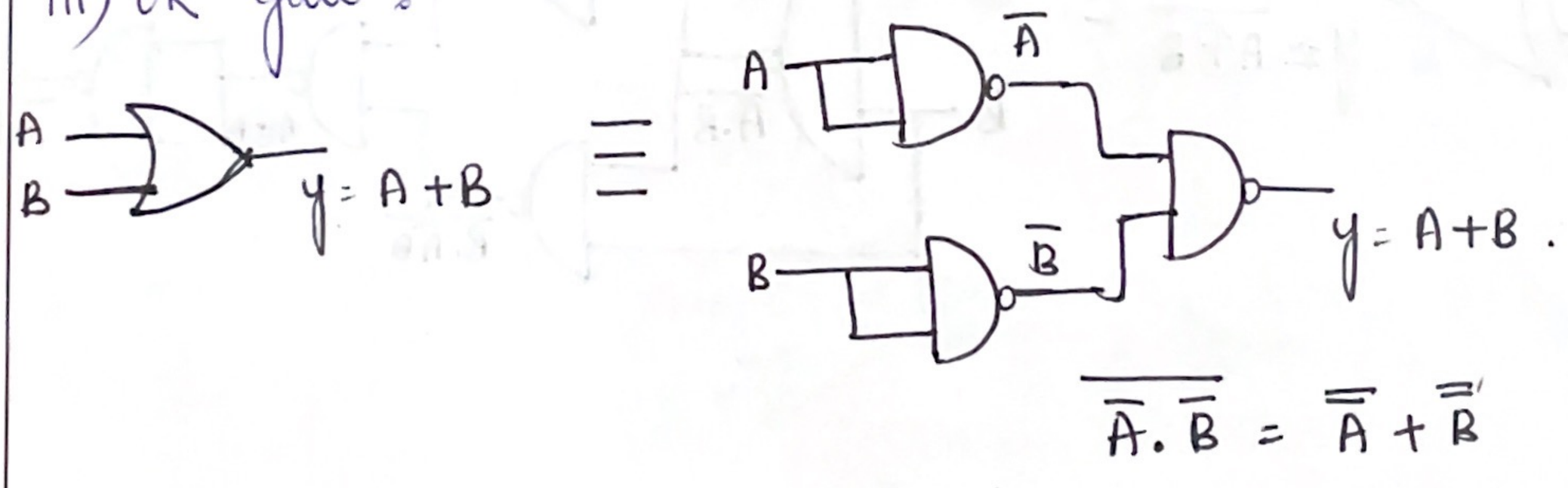
i) NOT gate:



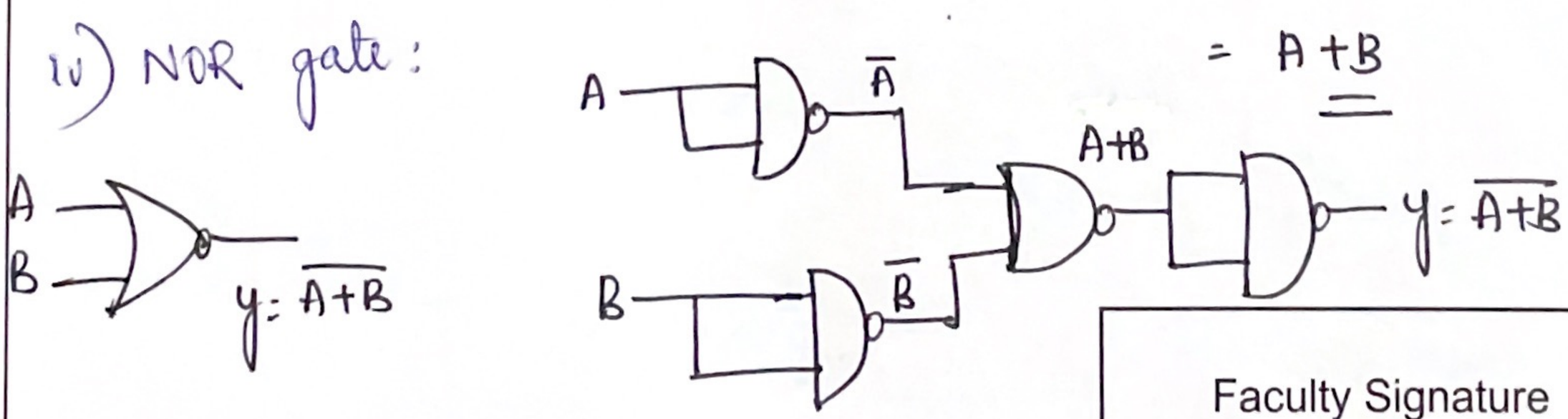
ii) AND gate:



iii) OR gate:

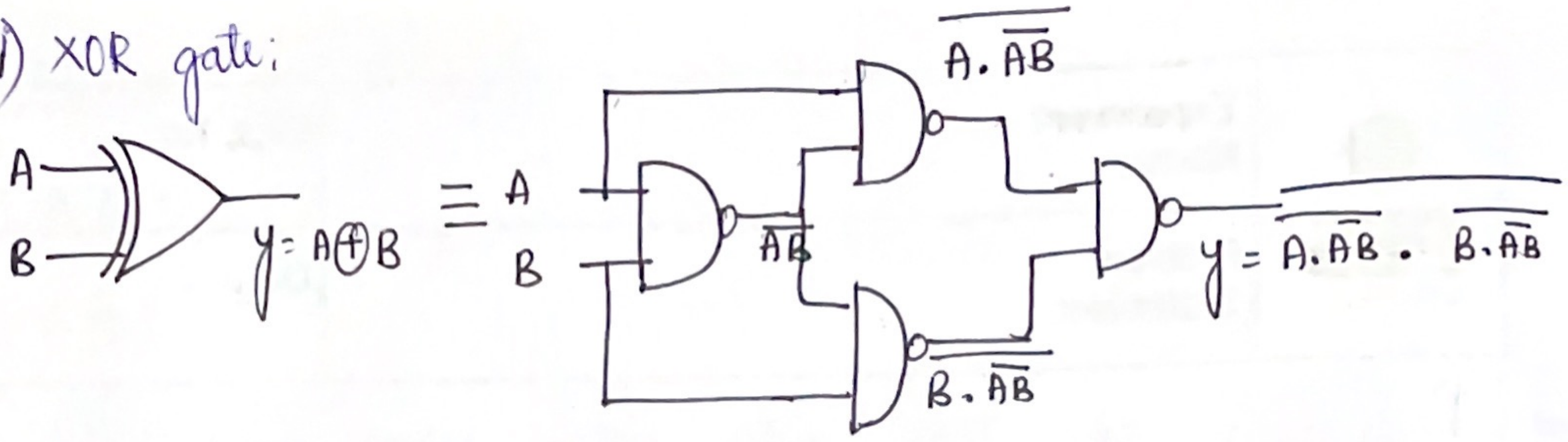


iv) NOR gate:



Faculty Signature

v) XOR gate:



$$y = \overline{\overline{A.\overline{AB}} \cdot \overline{B.\overline{AB}}} = \overline{\overline{A.\overline{AB}}} + \overline{\overline{B.\overline{AB}}}$$

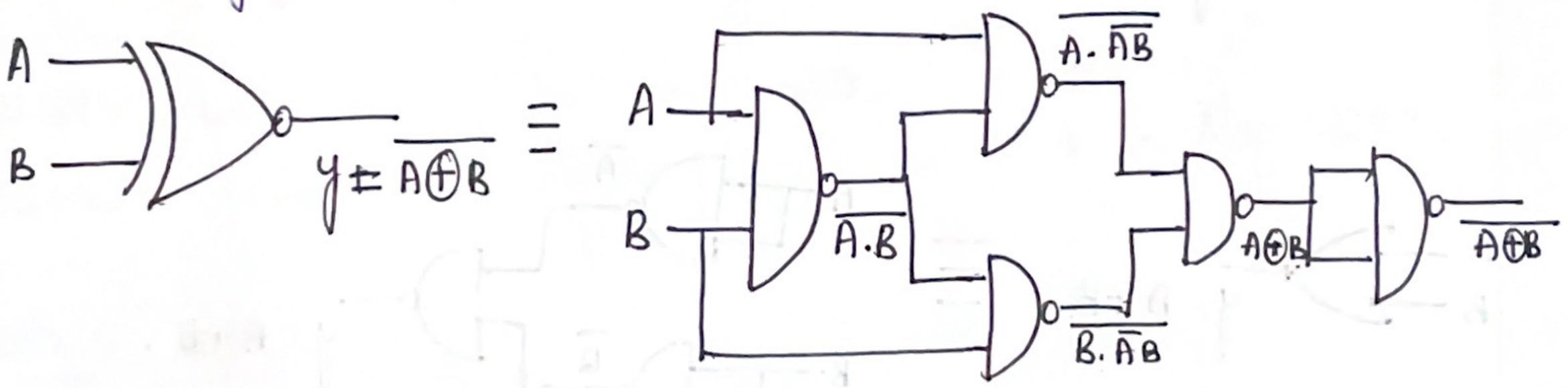
$$= (A.\overline{AB}) + (B.\overline{AB})$$

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

$$= A\overline{A} + A\overline{B} + B\overline{A} + B\overline{B}$$ ($A\overline{A} \to 0$, $B\overline{B} \to 0$)

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

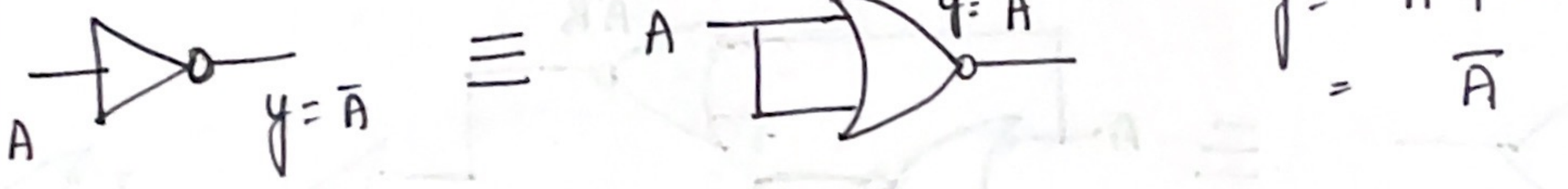
vi) XNOR gate:



PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

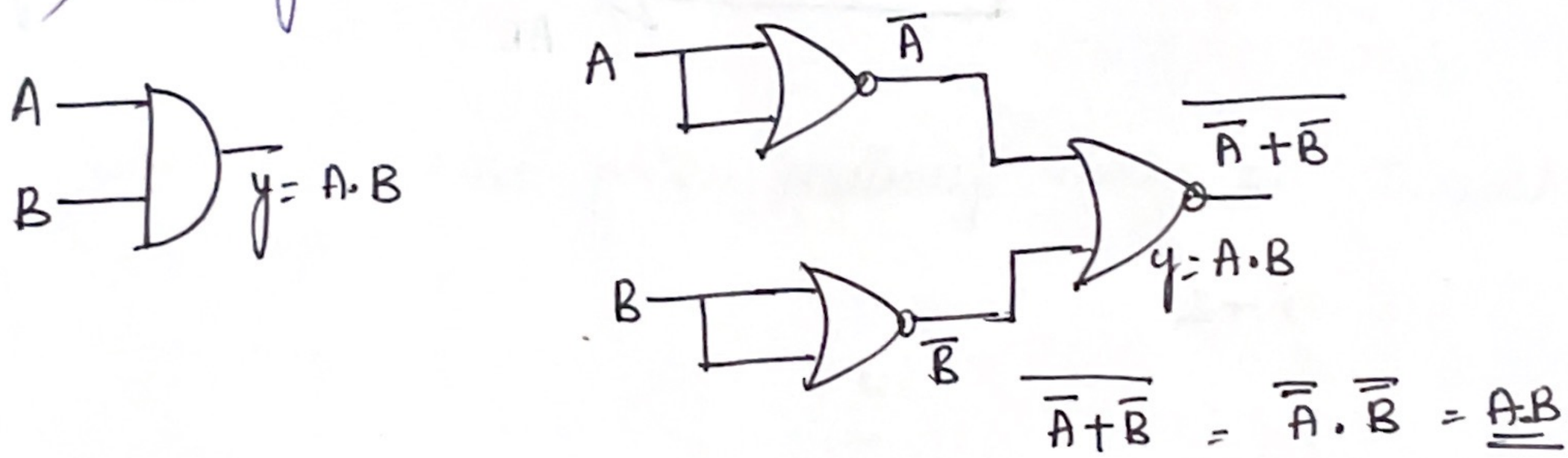
Realization of Logic gate using NOR gate:

i) NOT gate:



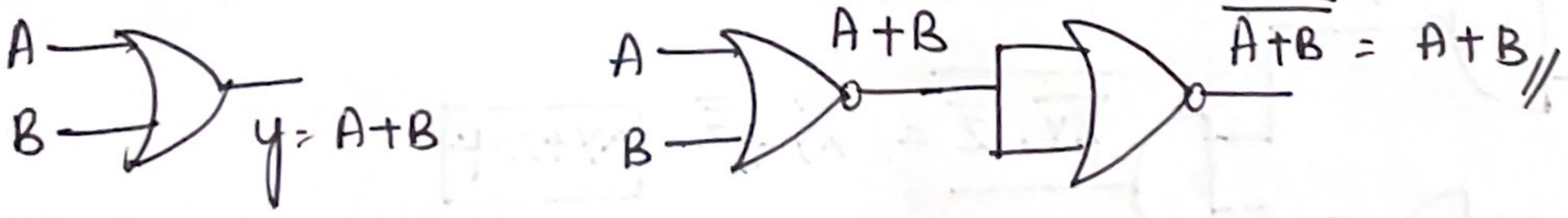
$$y = \overline{A + A} = \bar{A}$$

ii) AND gate:

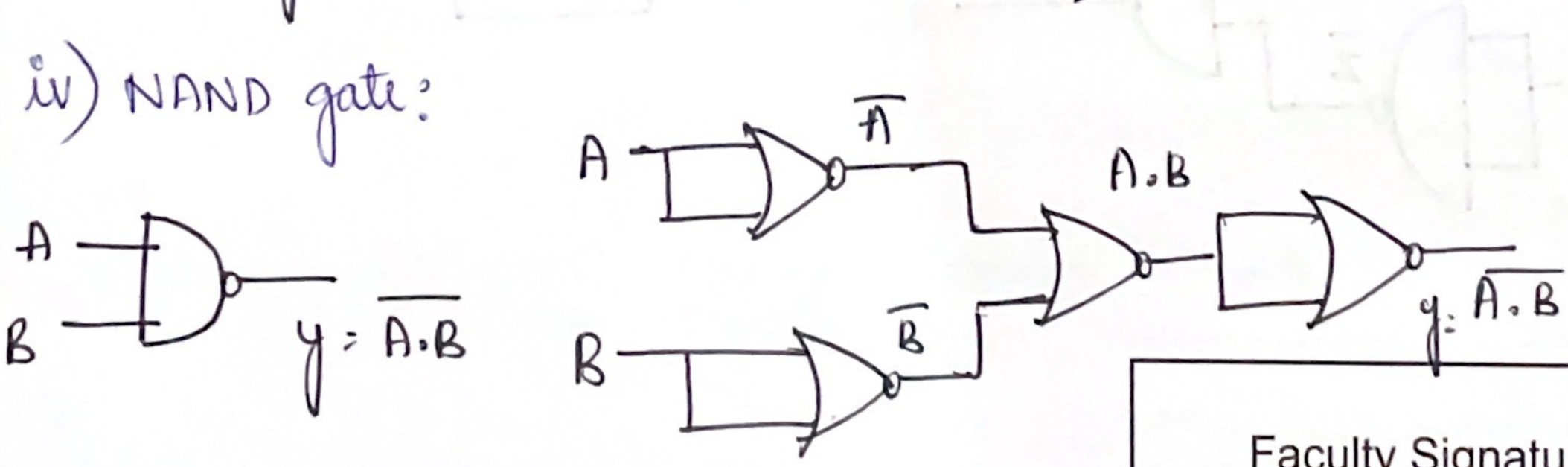


$$\overline{\bar{A} + \bar{B}} = \bar{\bar{A}} \cdot \bar{\bar{B}} = A \cdot B$$

iii) OR gate:

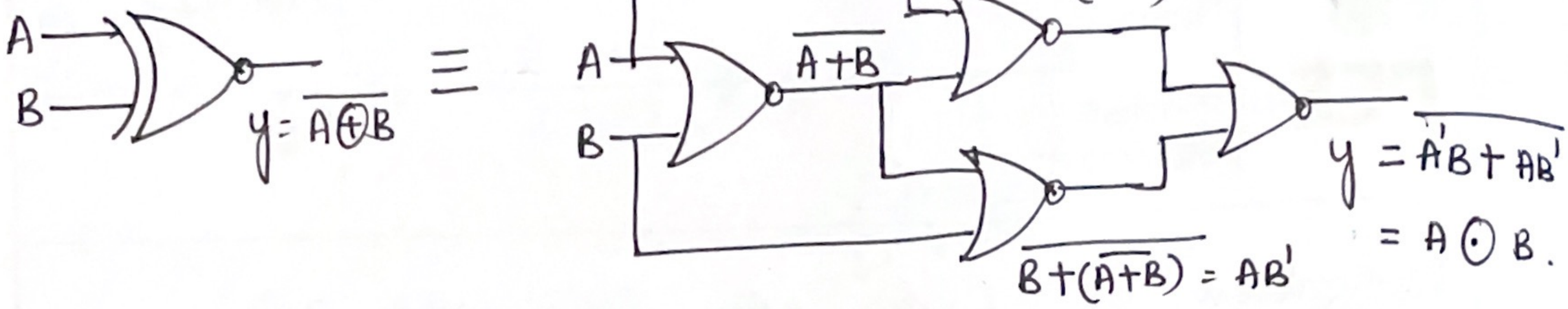


iv) NAND gate:

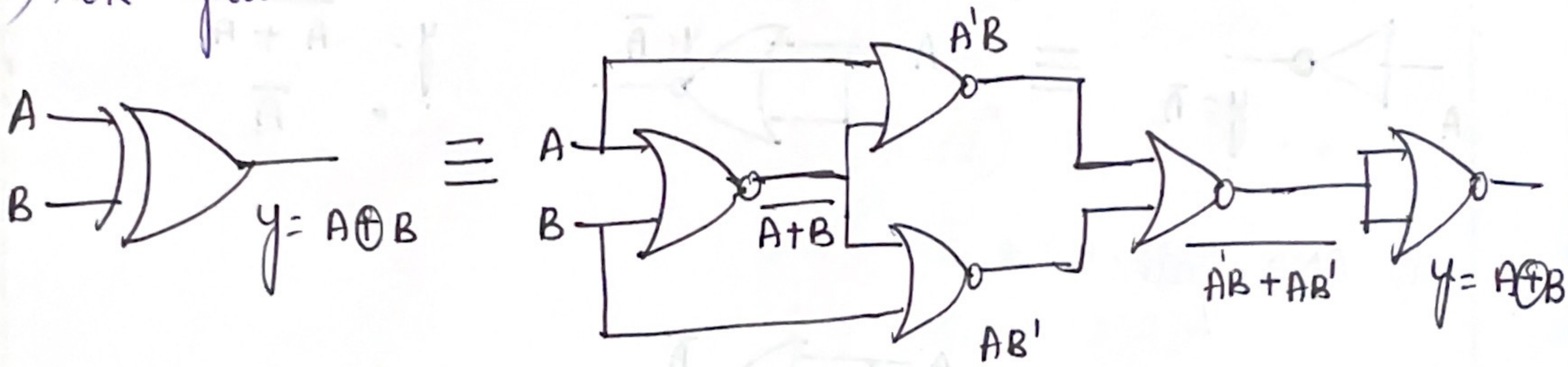


Faculty Signature

v) XNOR gate

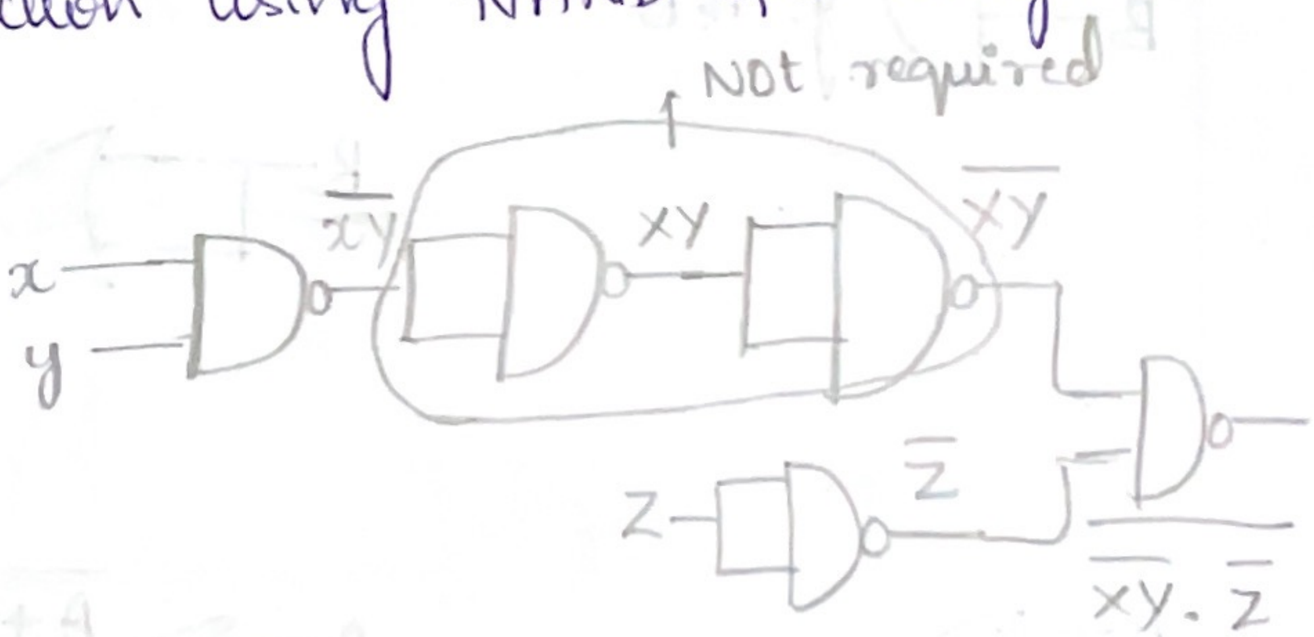


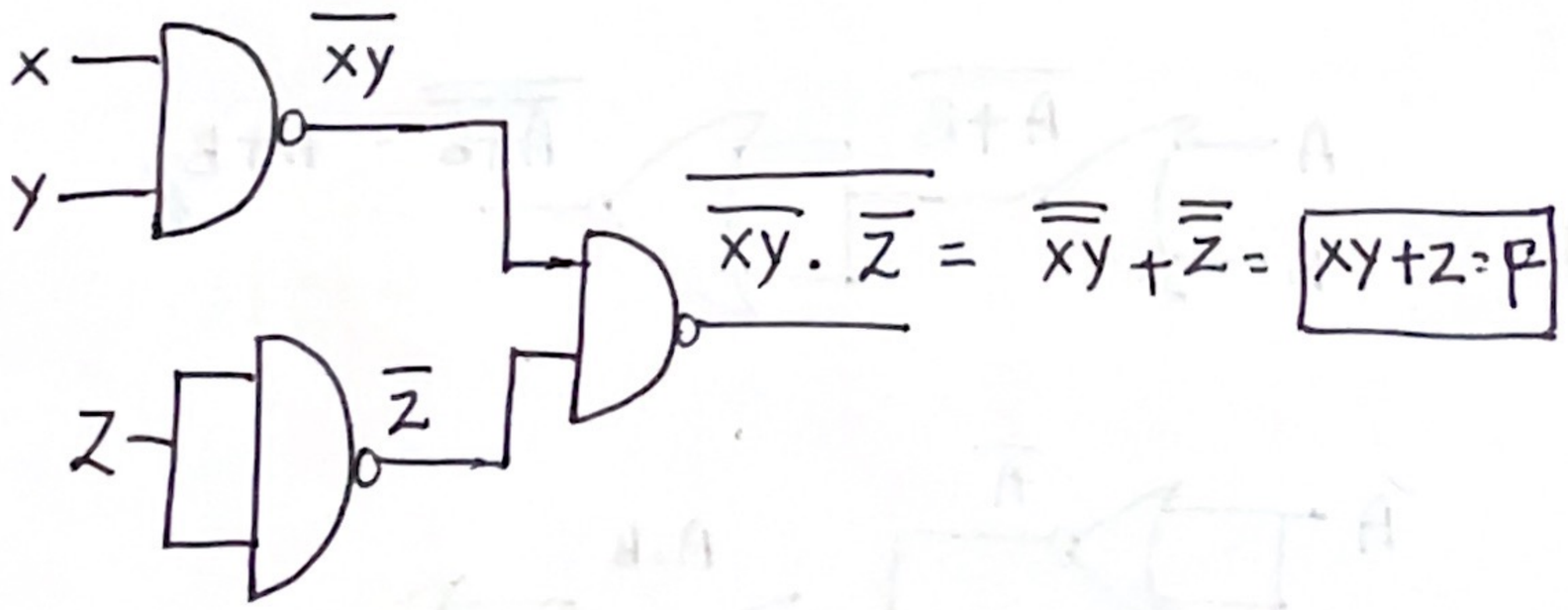
vi) XOR gate



Implement the given function using NAND Gate only.

$F = xy + z$





$\overline{\overline{xy}} + \overline{\overline{z}}$

$xy + z$

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Simplify the given boolean expression and implement using basic gates.

$F = A.B.C + A' + A.B'C$

$= A.B.C + A.B'.C + A'$

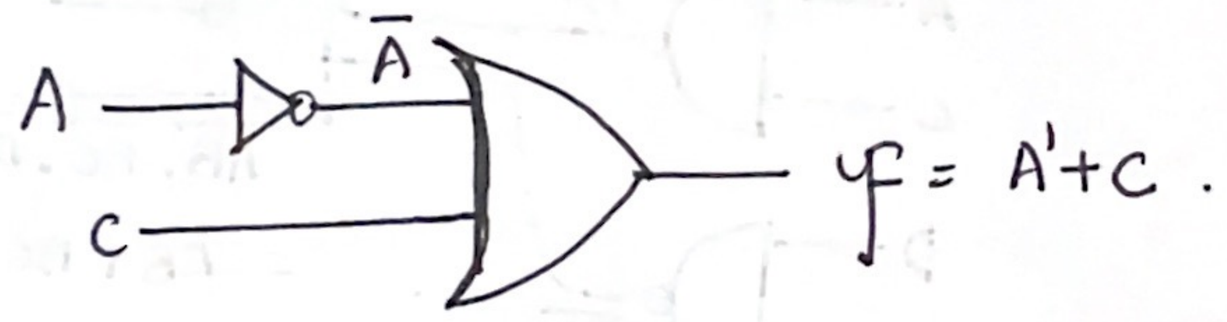
$= Ac[\cancel{B + B'}^{\,1} + A'$

$= AC[1] + A'$

$= A' + AC$ ∵ Distributive law

$= \cancel{(A' + A)}^{\,1}(A' + C)$

$= A' + C$



Simplify and realize the given function using basic gate.

$Y = A'B'C' + AB'C' + A'B' + AC'$

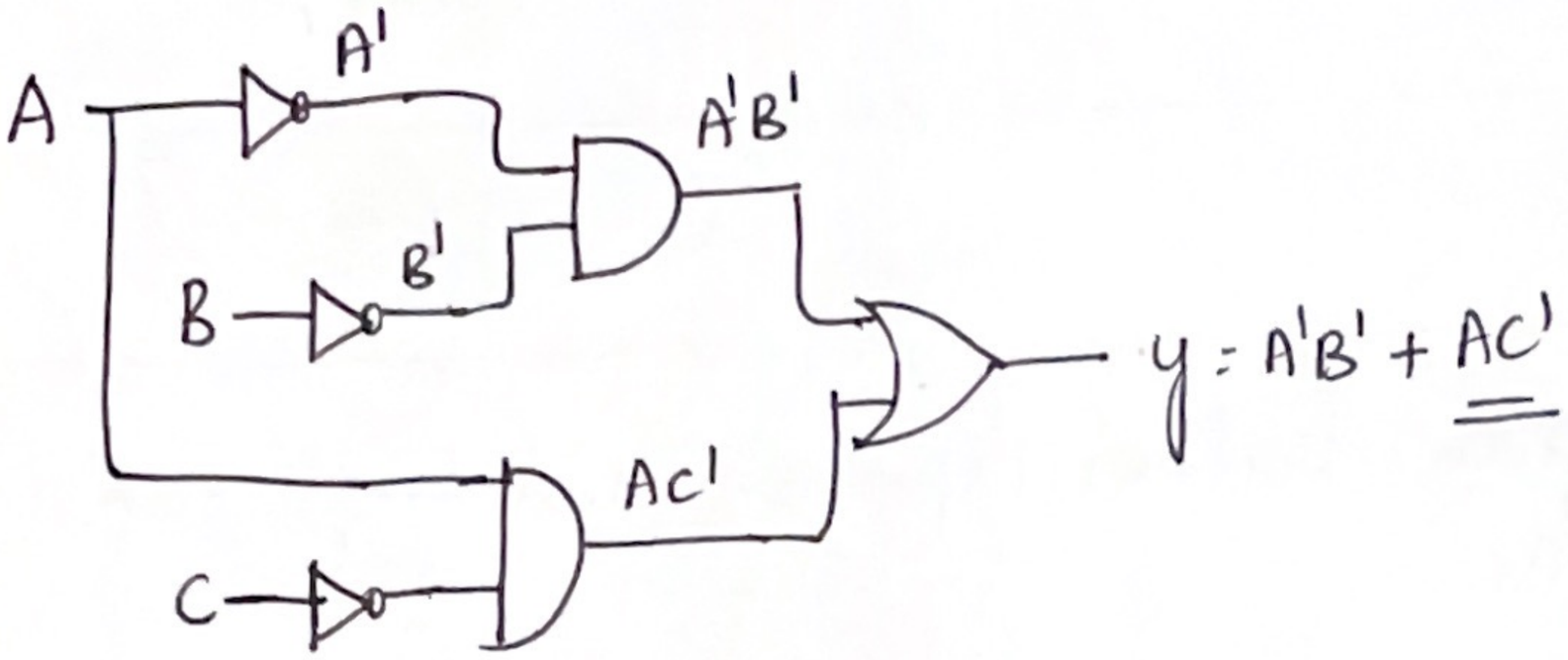
$= B'C'(\cancel{A' + A})^{1} + A'B' + AC'$

$= B'C' + \underline{A}'B' + \underline{A}C'$

$= A'B' + AC'$ (using consensus theorem)
$=$

Faculty Signature

$y = A'B' + AC'$



Simplify and Implement the given function using NAND gate only.

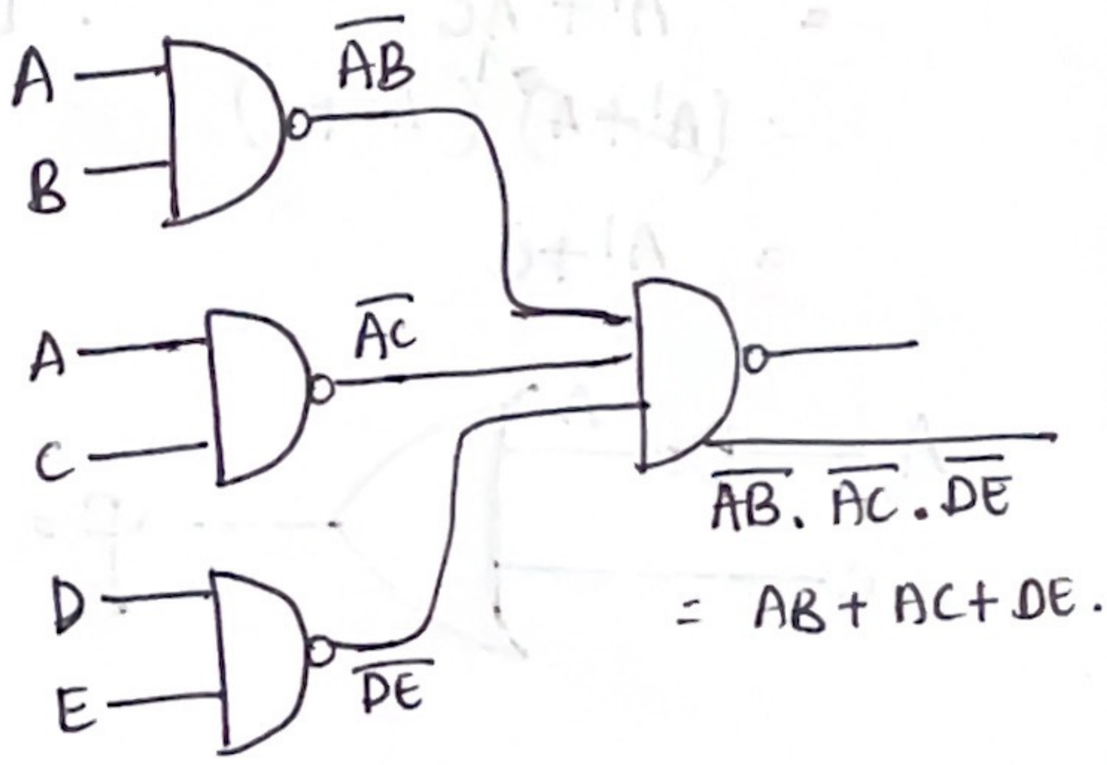
$F = A(B+C) + DE$.

$F = AB + AC + DE$

Apply De-Morgan's law

$$F = \overline{\overline{AB + AC + DE}}$$

$$= \overline{\overline{AB} \cdot \overline{AC} \cdot \overline{DE}}$$



Simplify the given Boolean expression.

$y = A + A'B + A'B'C + A'B'C'D + A'B'C'D'E$.

$= A + A'(B + B'C + B'C'D + B'C'D'E)$

$= A + B + B'C + B'C'D + B'C'D'E$ $\quad \because A + A'B = A + B$ absorption law.

$= A + B + B'(C + C'D + C'D'E)$

$= A + B + C + C'D + C'D'E$

$= A + B + C + C'(D + D'E)$

$= A + B + C + D + D'E = A + B + C + D + E$ //

PES UNIVERSITY	Experiment Name:	Expt. No.
	Problem Statement:	Date

Simplify the given Boolean expression and implement the circuit using basic gates and using NAND gate.

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

$F = AB + AB + AC + B.B + BC$

$= AB + AC + B + BC$

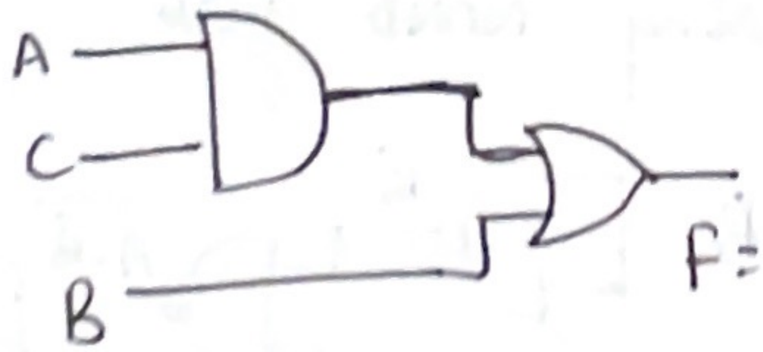
$= AB + AC + B(1+C)$

$= AB + AC + B$

$= B(1+A) + AC$

$$\boxed{F = AC + B}$$

Logic circuit using Basic gate

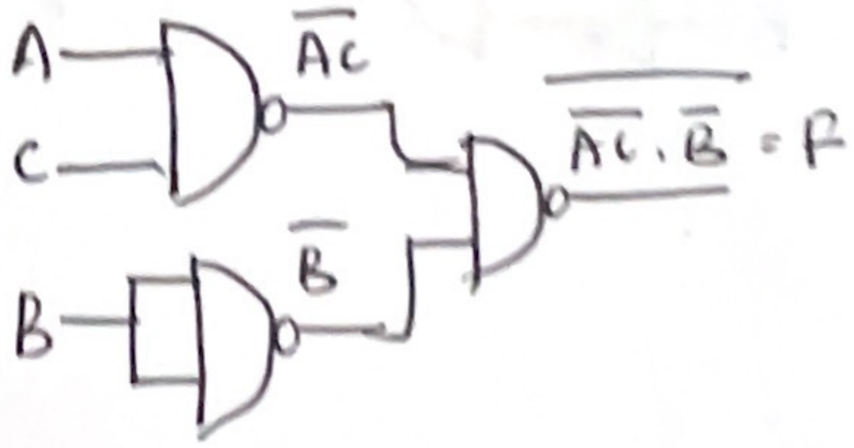


using NAND gate

$F = AC + B$

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

$= \overline{\overline{AC} \cdot \overline{B}}$



Faculty Signature

Simplify and realize the given function using

i) Basic gates

ii) NAND gate.

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

Apply De-Morgan's law.

$$Y = \overline{A'B} \cdot \overline{\overline{A}} \cdot \overline{A.B}$$

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

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

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

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

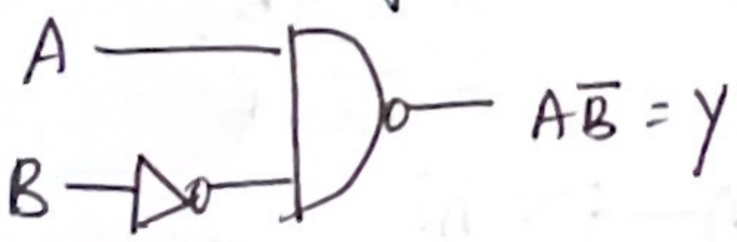
($A.A.\overline{A} \to 0$, $A\overline{B}\,\overline{A} \to 0$)

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

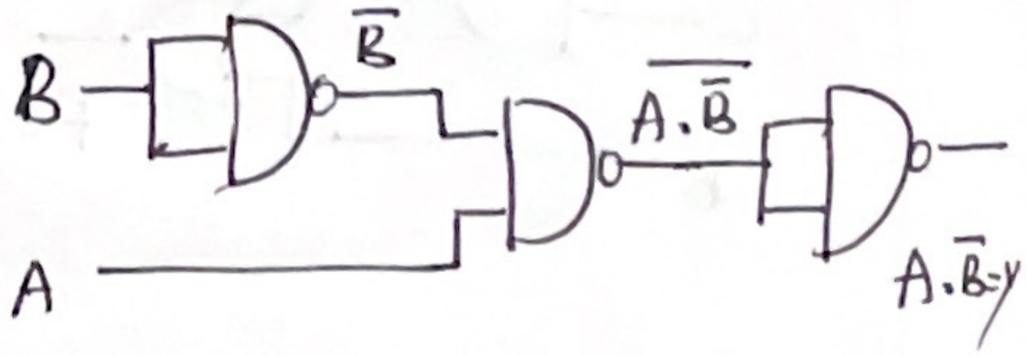
$$= A\overline{B}$$

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

Circuit using basic gates.

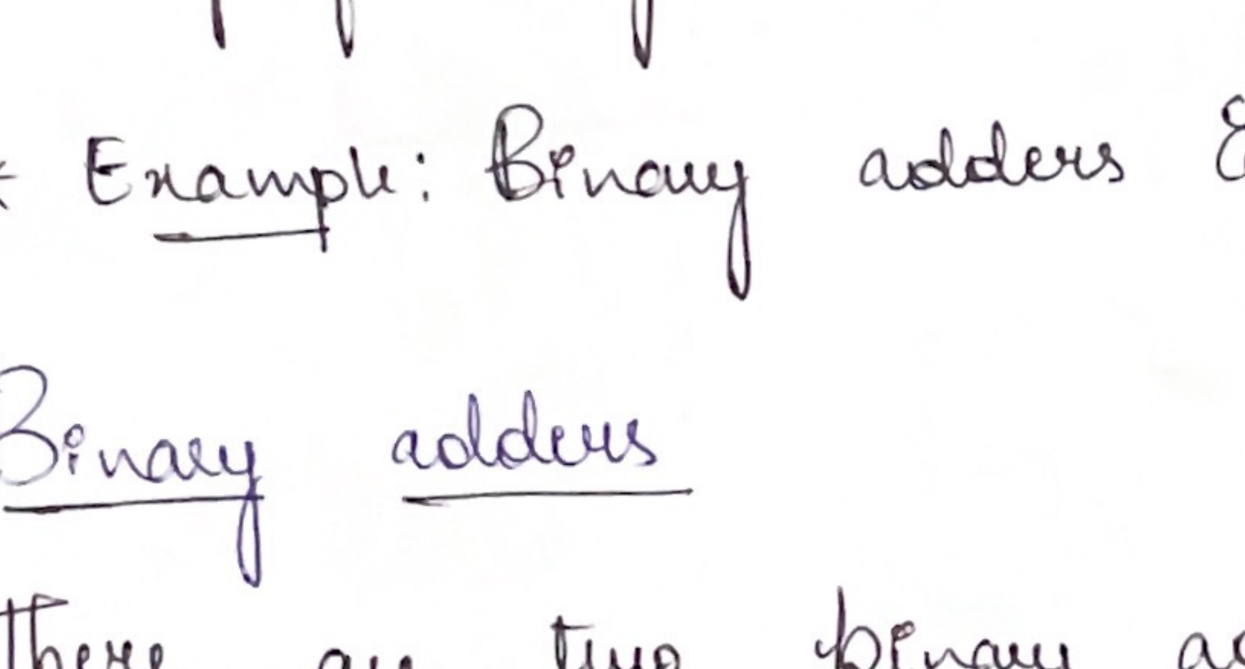


using NAND gate



Combinational logic circuits:

* Combinational logic circuits are digital circuits which are interconnection logic gates whose output depends on current combination of inputs.



* Operation performed by a combinational circuit is specified by a "boolean function".

* Example: Binary adders & Multiplexers etc.

Binary adders

* There are two binary adders
  1] Half adder
  2] Full adder

* A combinational circuit which performs a bit addition is called Half adder.

* A combinational circuit which performs 3 bit addition is called full adder.

* Half adders / full adders are the basic compone -nts of adders such as: ripple carry adder, carry look ahead adder & other fast adders.

Half adder:-

* It Peforms a bit binary addition
* x & y are two binary inputs, Sum (s) Carry (c) are the two binary outputs.

* Boolean Expression for

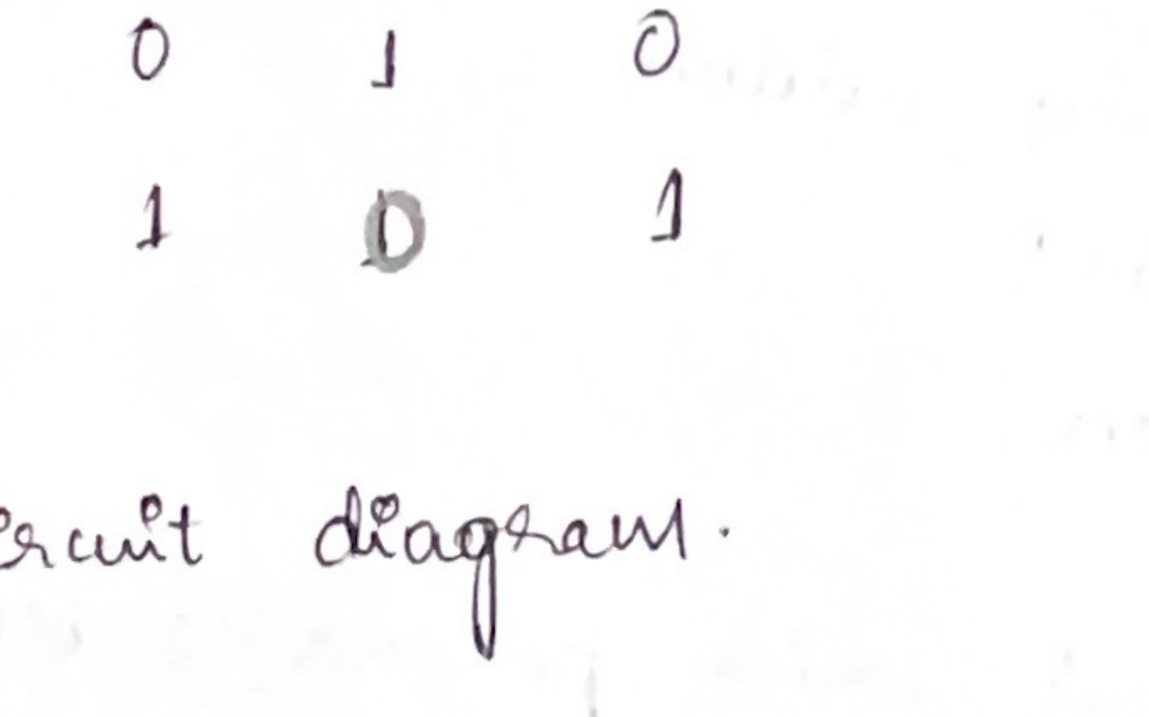
$$\text{Sum}, \quad S = \bar{x}y + x\bar{y} = x \oplus y$$

$$\text{Carry}, \quad C = xy$$

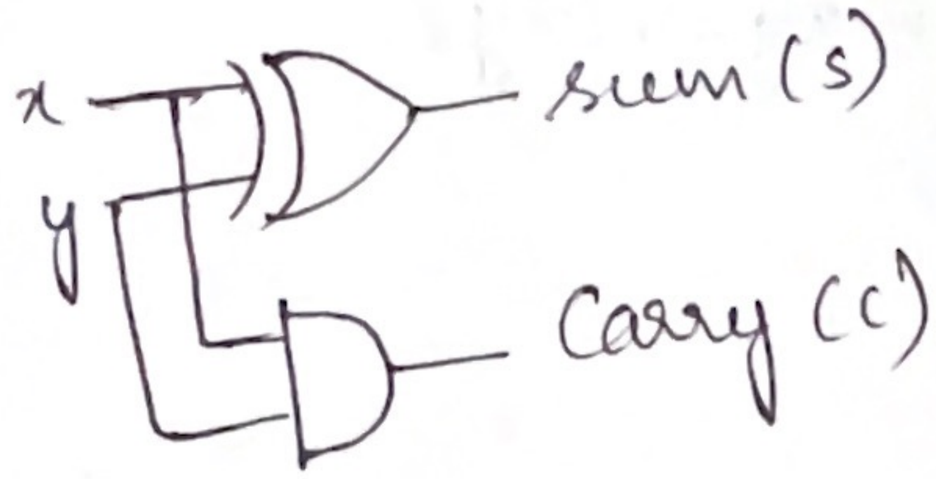
* Truth table.

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

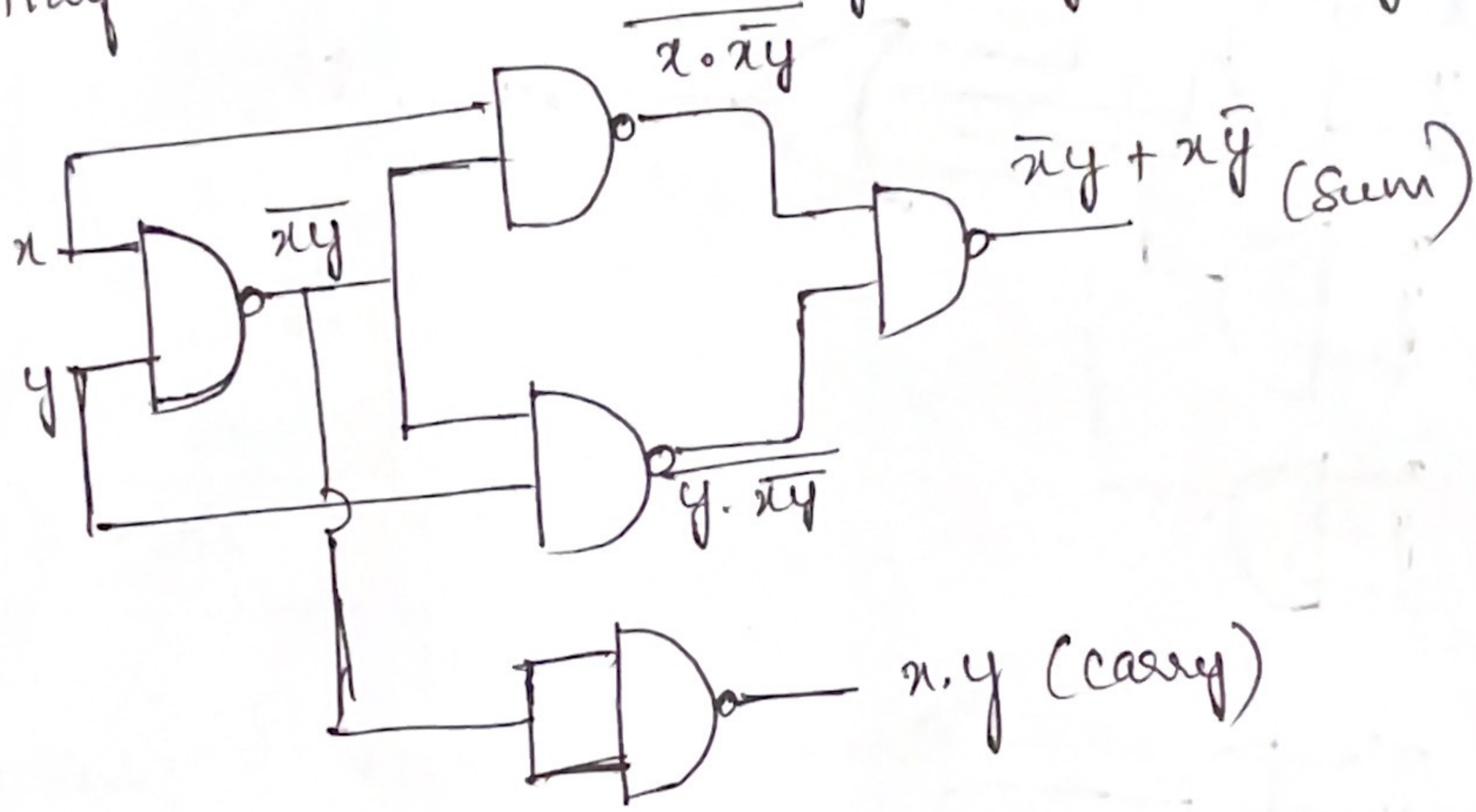
* Circuit diagram.



* Half adder circuit can also be written as follows.



* Half adder circuit using only NAND gates.



Full adder :-

* It peforms 3 bit binary addition
* x, y & z are its binary inputs and S & C are its output.
* Truth table

x	y	z	S	C
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

* Boolean Expression.

Sum $S = x'y'z + x'yz' + xy'z' + xyz$

Carry $C = x'yz + xy'z + xyz' + xyz$.

* Carry expression:

$$C = x'yz + xy'z + xyz' + xyz$$

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

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

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

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

$$= x'yz + xy + xz$$

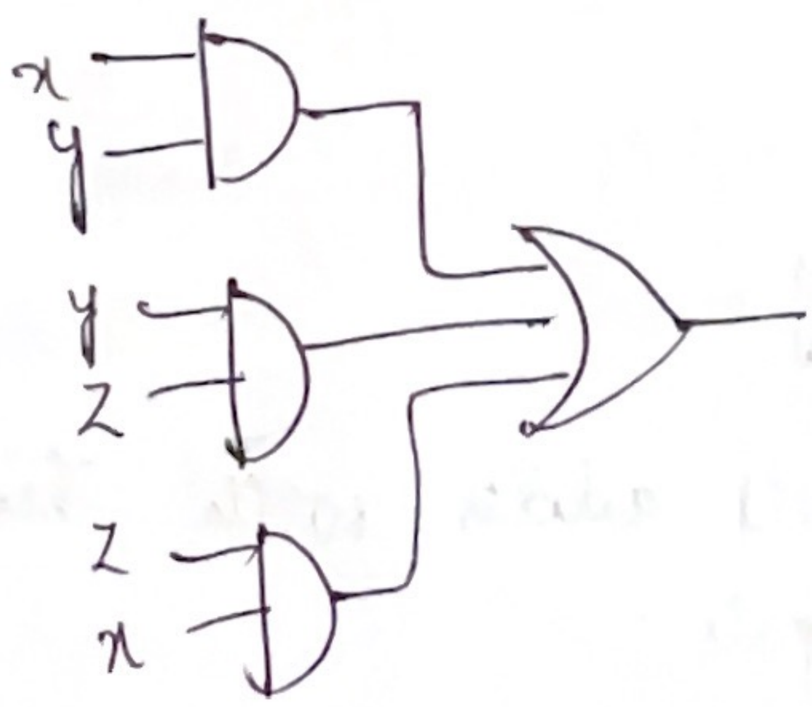
$$= z(x + x'y) + xy$$

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

$$= zx + xy + xy$$

$$\boxed{C = xy + yz + zx}$$

∴ Simplified circuit to generate carry.



* Boolean expression for Sum:

$$S = x'y'z + x'yz' + xy'z' + xyz$$

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

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

$$\boxed{S = x \oplus (y \oplus z)}$$

* Boolean expression for Carry:

$$C = x'yz + xy'z + xyz' + xyz$$

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

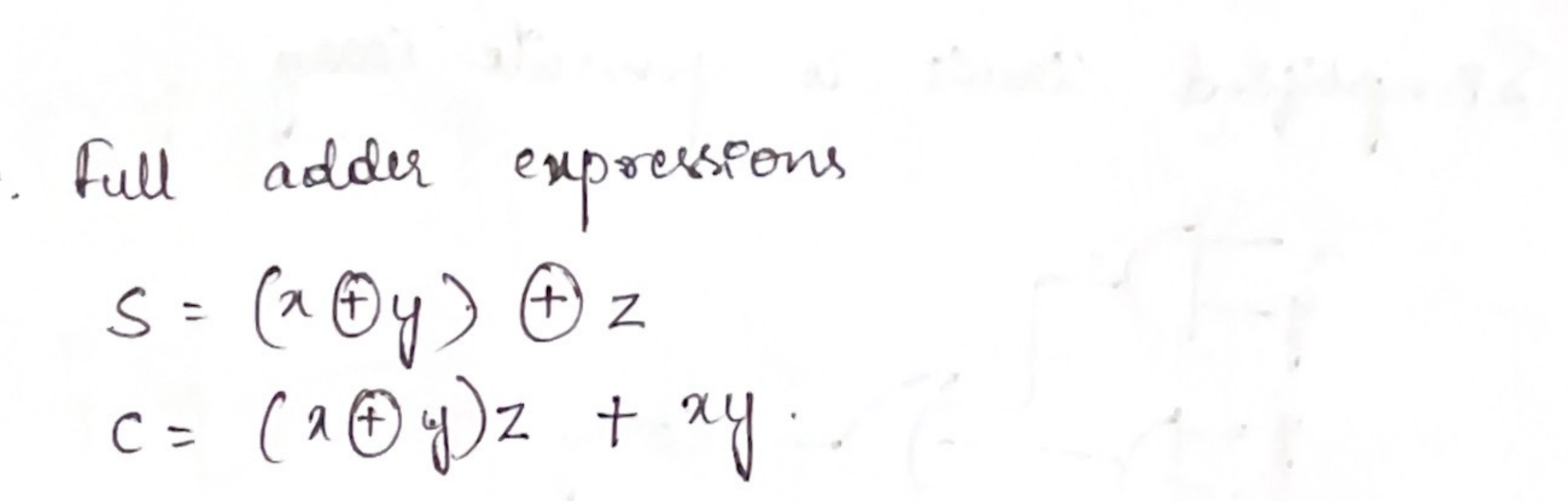
$$\boxed{C = z(x \oplus y) + xy.}$$

* ∴ Full adder expressions

$$S = (x \oplus y) \oplus z$$

$$C = (x \oplus y)z + xy$$

* Implementation of full adder with two half adders and an OR gate.



* Full adder circuit using NAND gates

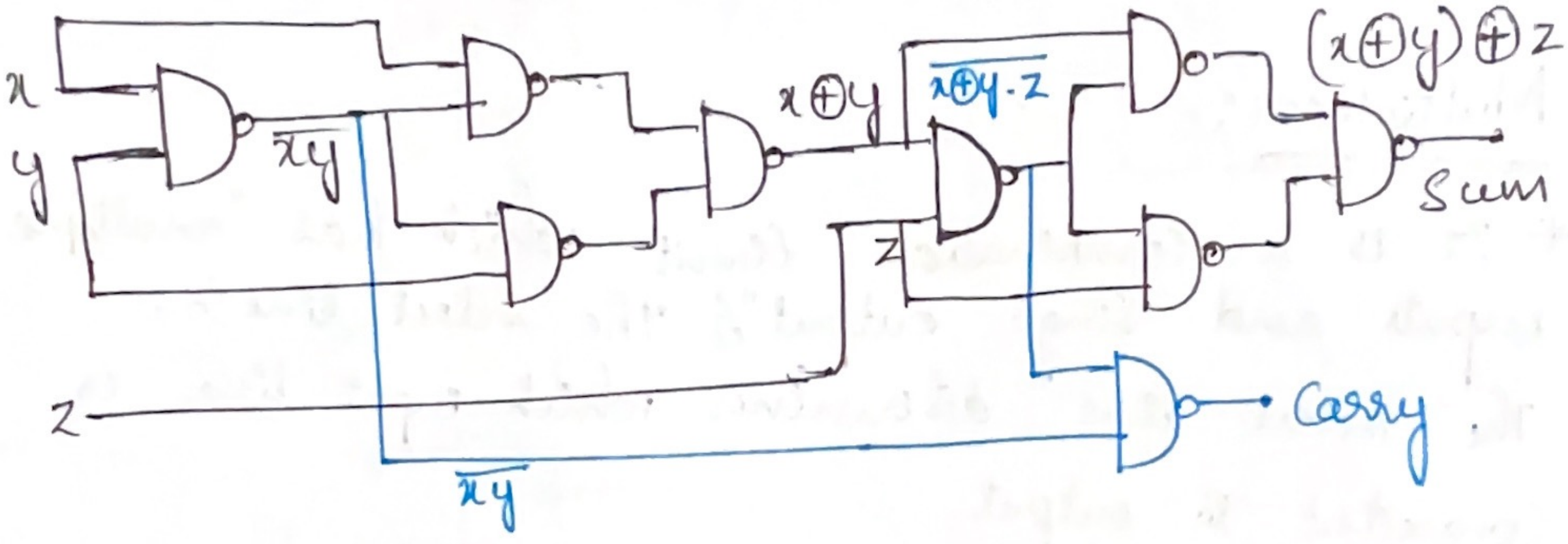
WKT

$$\text{Sum} = (x \oplus y) \oplus z$$

$$\text{Carry} = (x \oplus y)z + xy$$

$$= \overline{\overline{(x \oplus y)z + xy}}$$

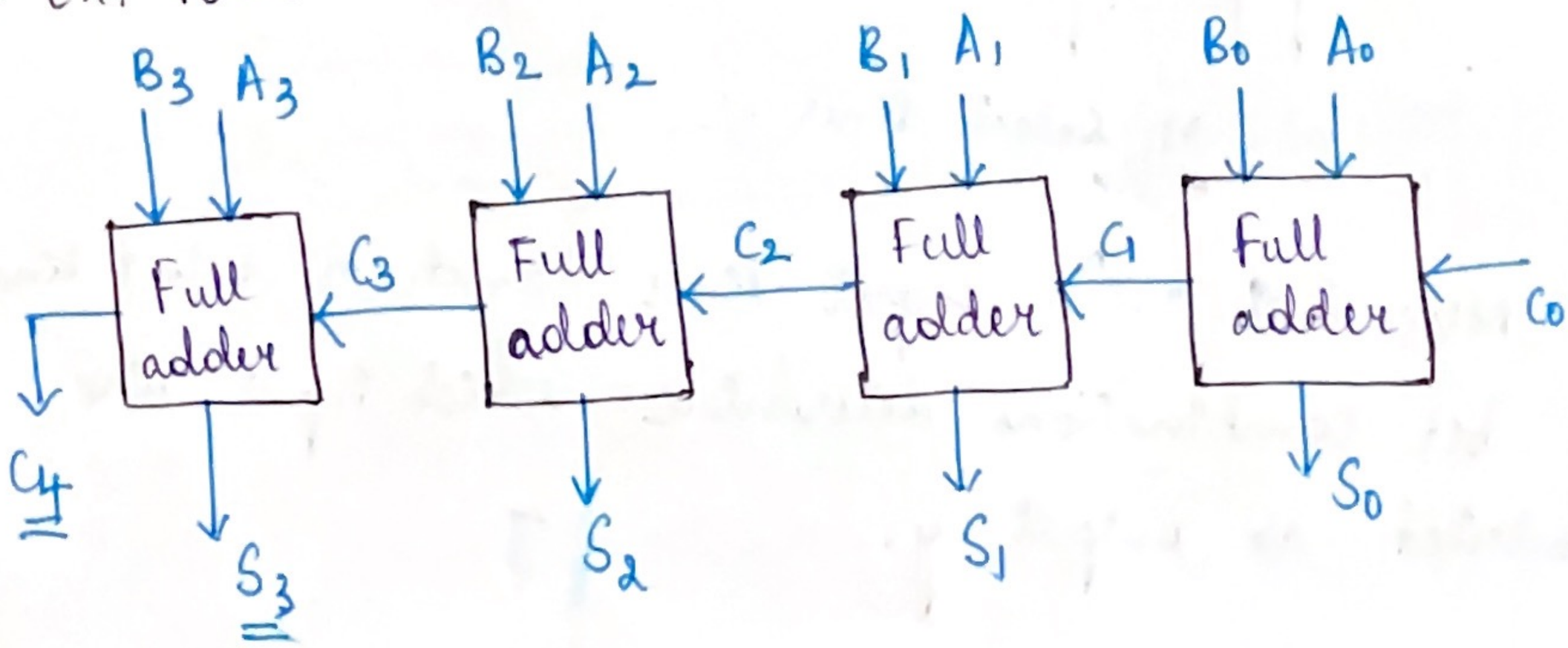
$$= \overline{\overline{(x \oplus y)z} \cdot \overline{xy}}$$



* Ripple Carry Adder *

* Adder circuit can perform upto 3 bit addition. For n bit addition, we have to cascade n adders.

* Ex: four bit adder



* Example:-

C_3 C_2 C_1 C_0

1 1 1 0

A = 1 1 0 1

A_3 A_2 A_1 A_0

B = 0 1 1 1

B_3 B_2 B_1 B_0

---

1 0 1 0 0

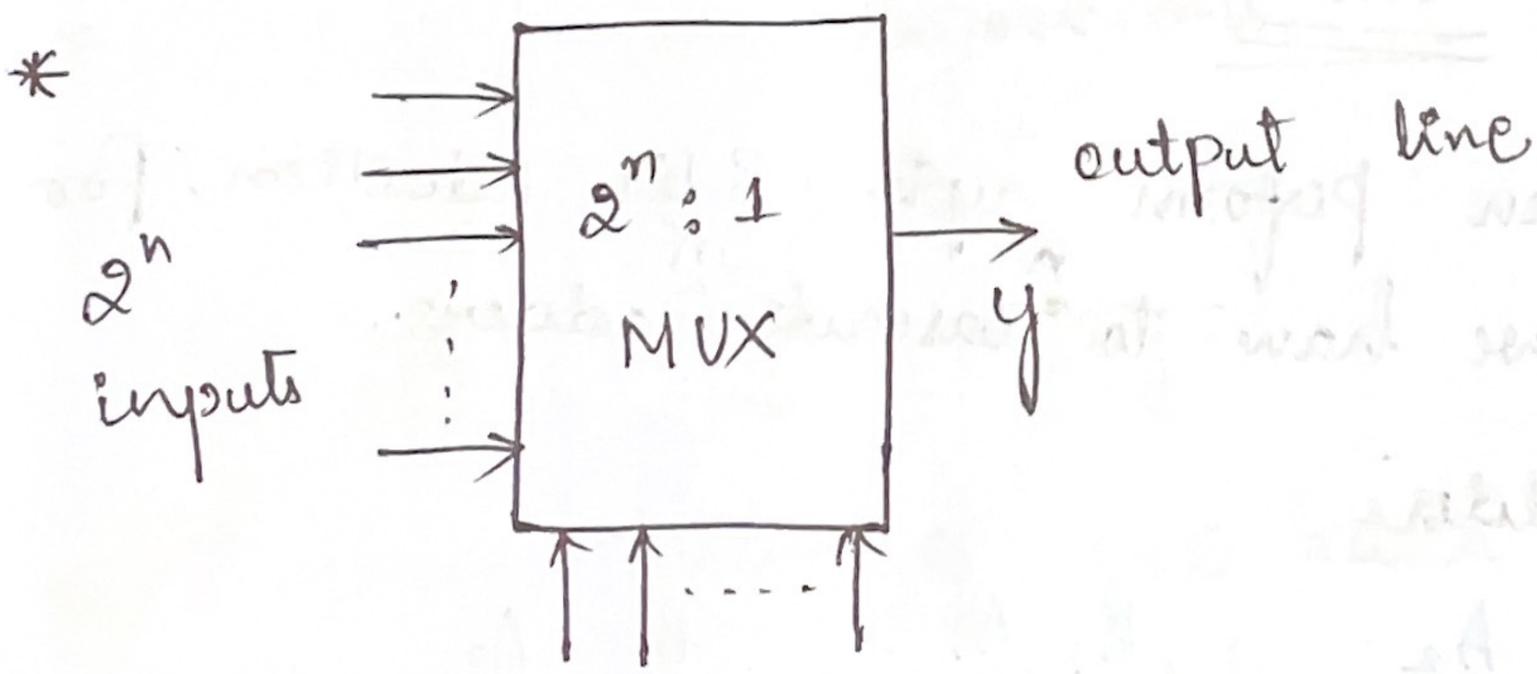
/ S_3 S_2 S_1 S_0

C_4

-: Multiplexer and Demultiplexer :-

Multiplexer :-

* It is a combination circuit which has "multiple inputs and single output" & the select line/s. The select line determines which input line is connected To output.

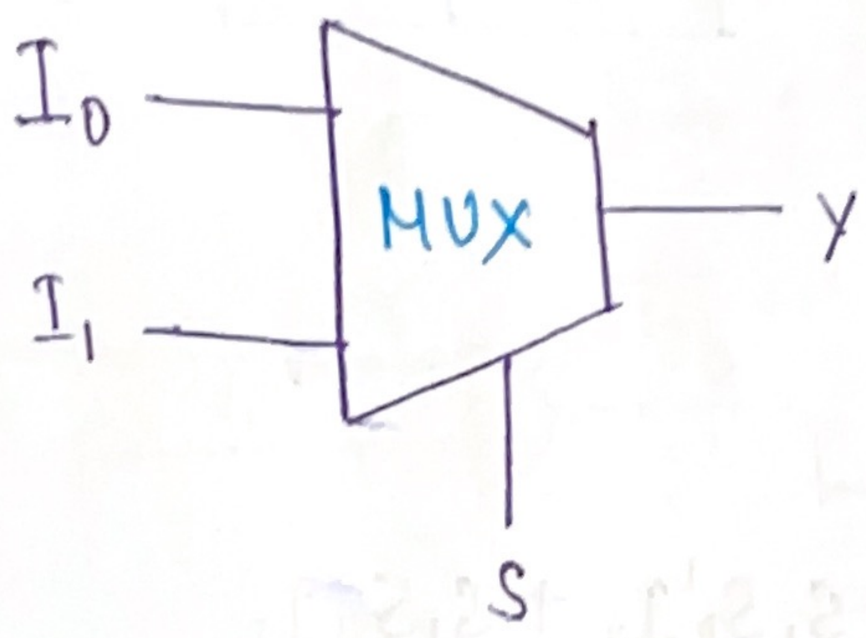


* A MUX has 2^n input lines and n select lines whose bit combinations determine which input line is seleted as output y.

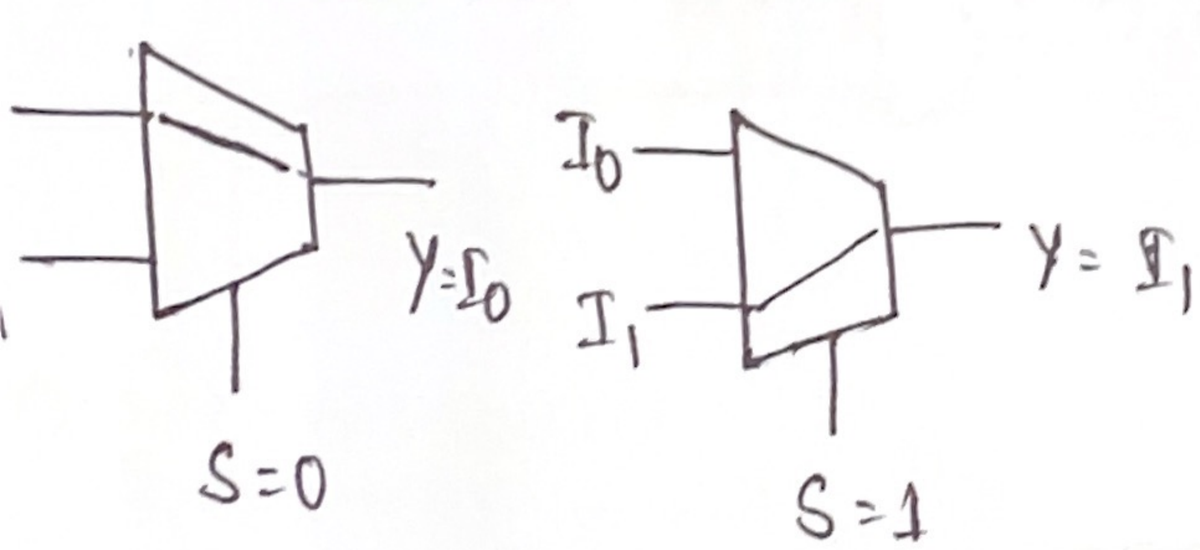
* Application of Mux is mainly in data Compression.

* The function of demultiplexer is to inverse the function of multiplexer.

* 2:1 MUX (Two to one line multiplexer)

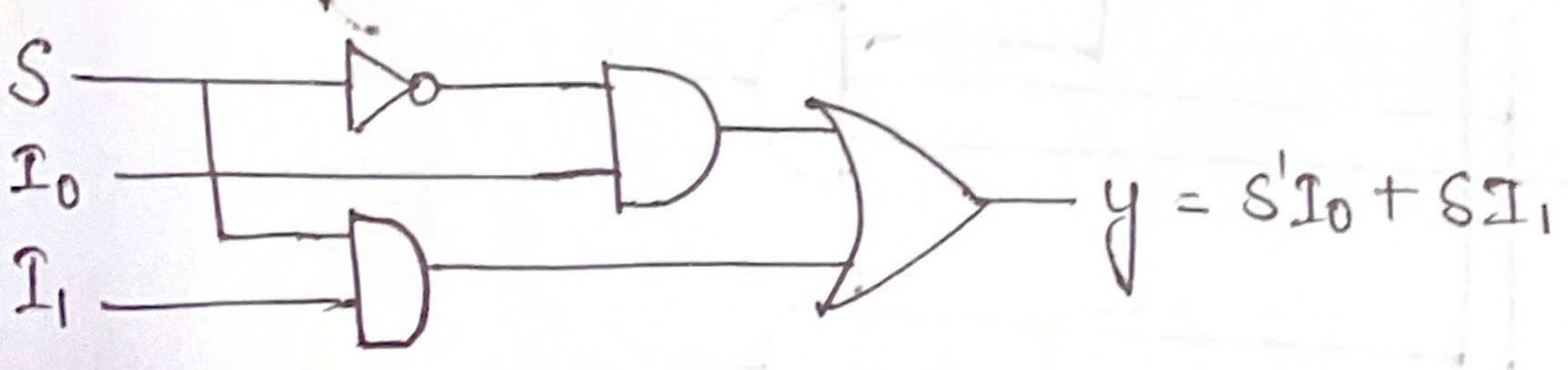


S	Y
0	I_0
1	I_1

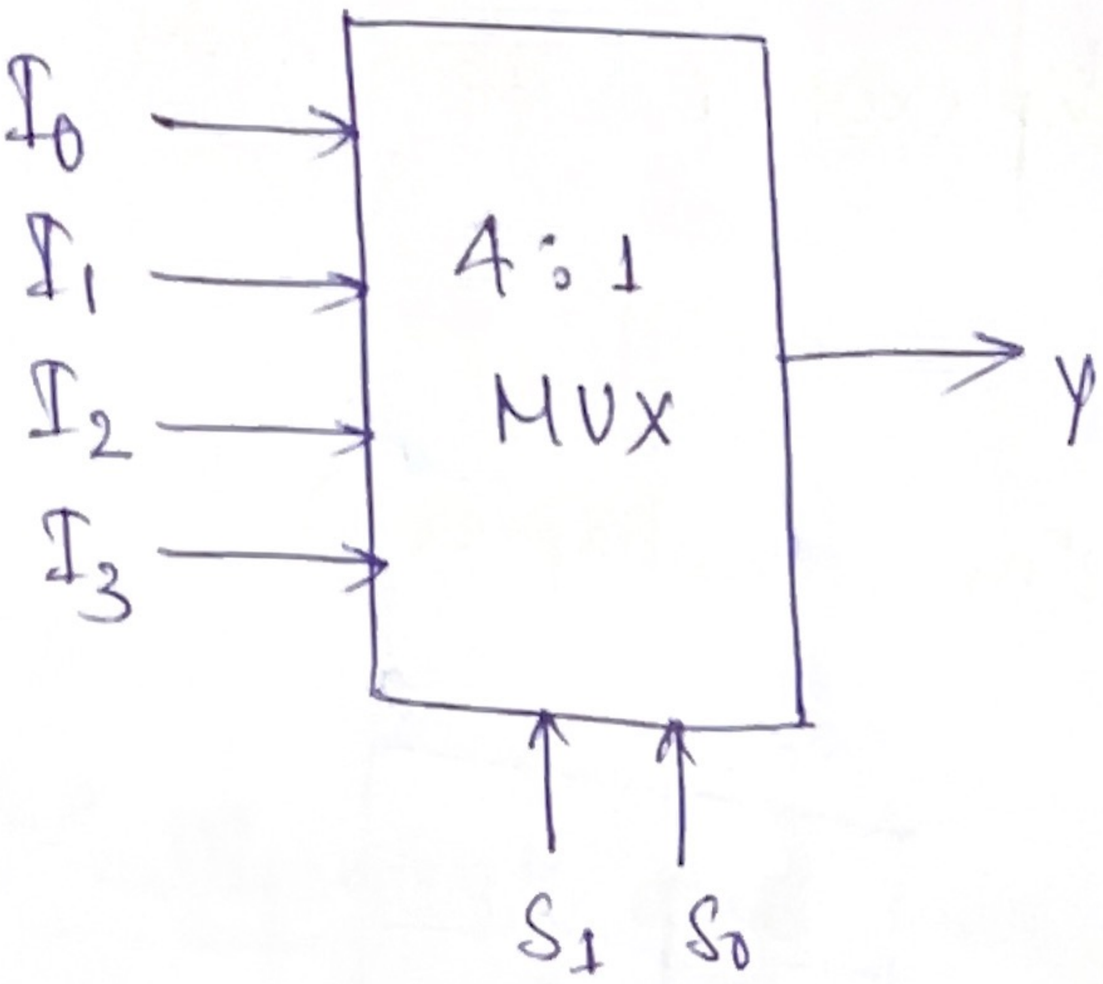


* Boolean Expression : $S'I_0 + SI_1$

* Circuit diagram.



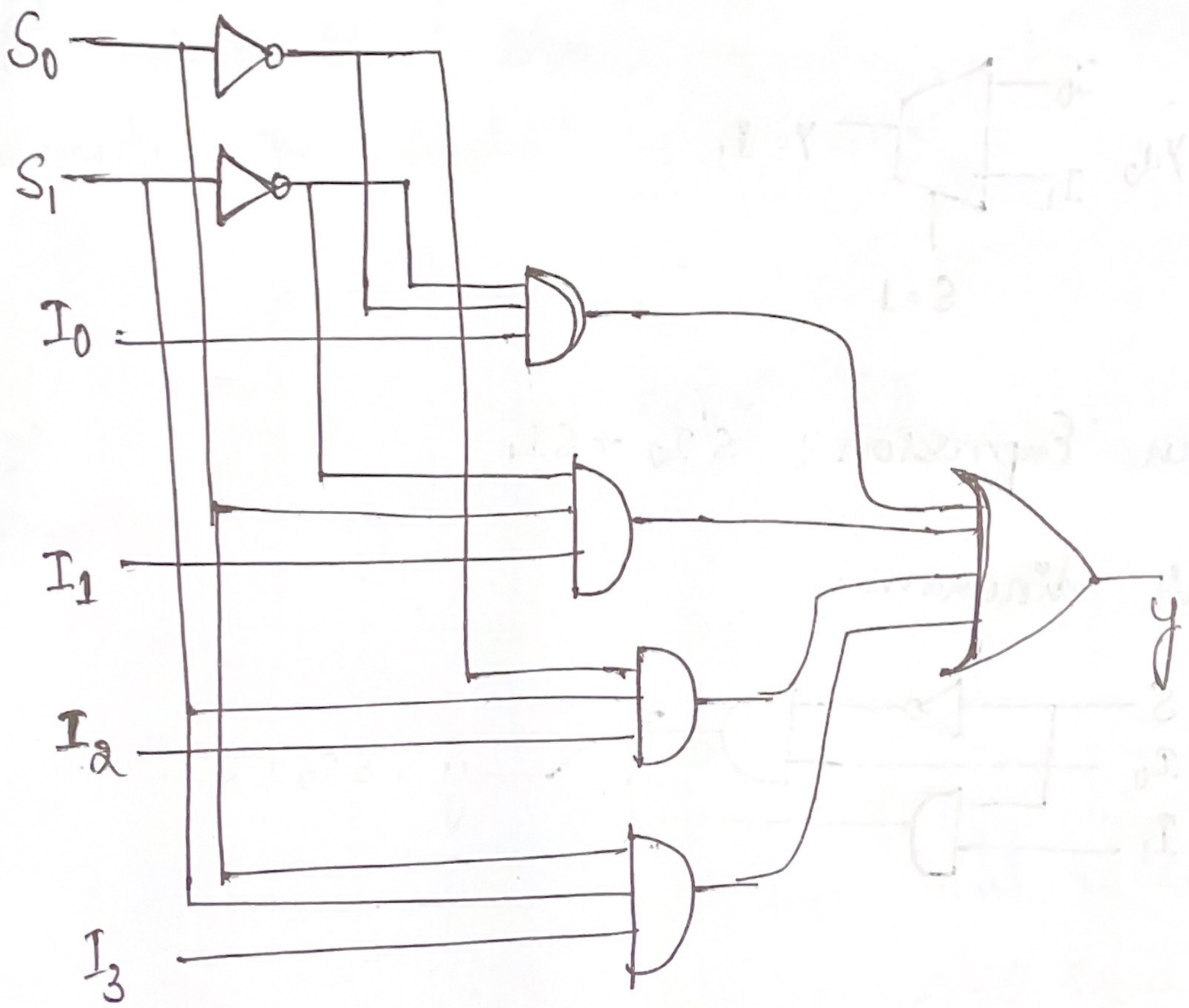
* 4:1 MUX (four to one line multiplexer)



S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

* Boolean expression

$$y = S_0' S_1' I_0 + S_1' S_0 I_1 + S_1 S_0' I_2 + S_1 S_0 I_3$$



* 2:1 MUX using NAND gates only.

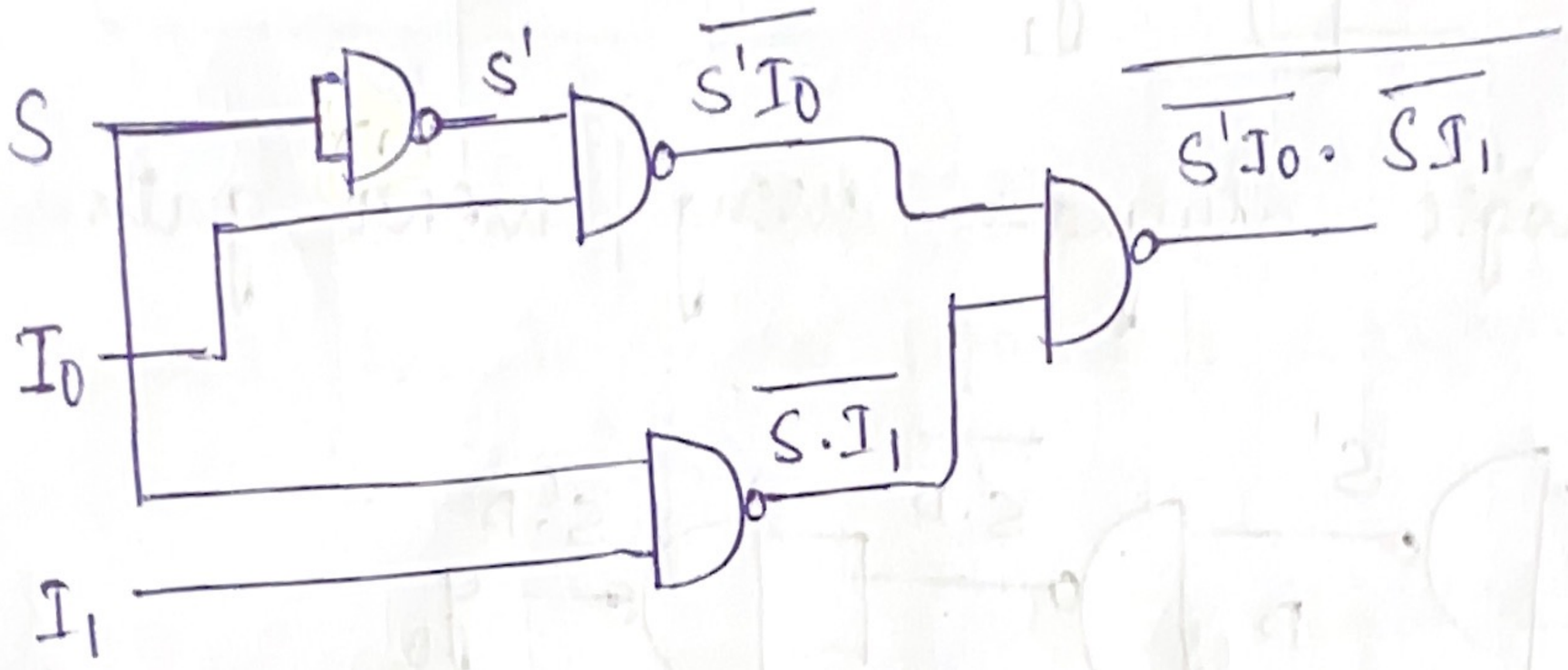
WKT

Boolean expression for 2:1 MUX is.

$$y = S'I_0 + SI_1$$

$$y = \overline{\overline{S'I_0 + SI_1}}$$

$$y = \overline{\overline{S'I_0} \cdot \overline{S.I_1}}$$

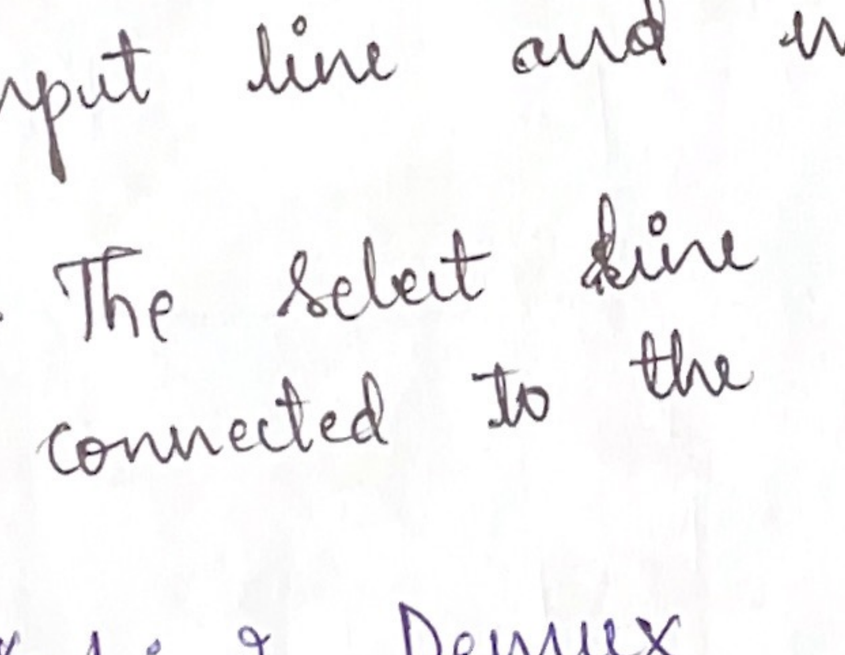


* De-multiplexer *

* It is a combinational circuit which has one input line and multiple output lines.
* The select line decides which output line is connected to the input line.

* 1:2 Demux

Block diagram



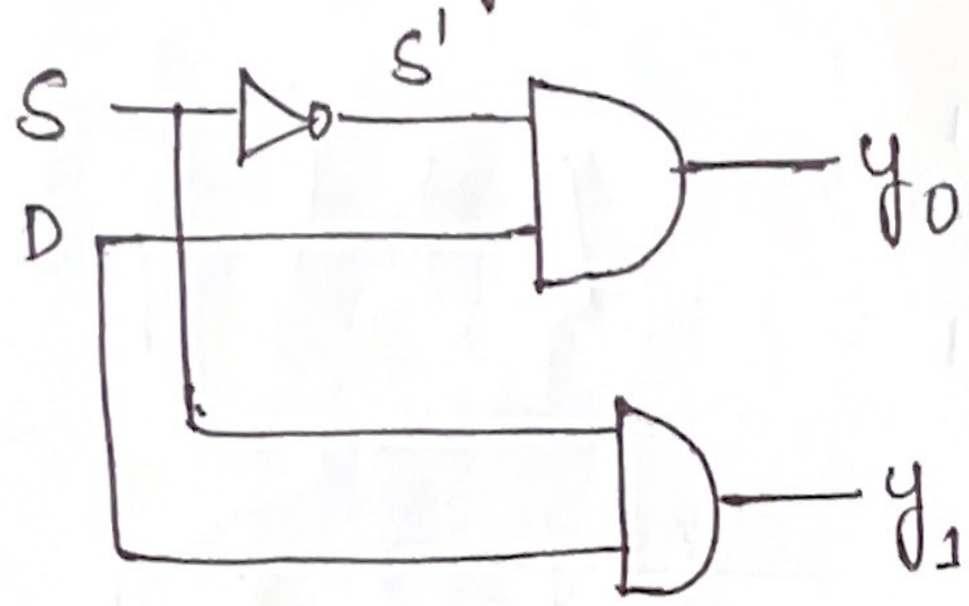
S	y_0	y_1
0	D	-
1	-	D

* Boolean expression is.

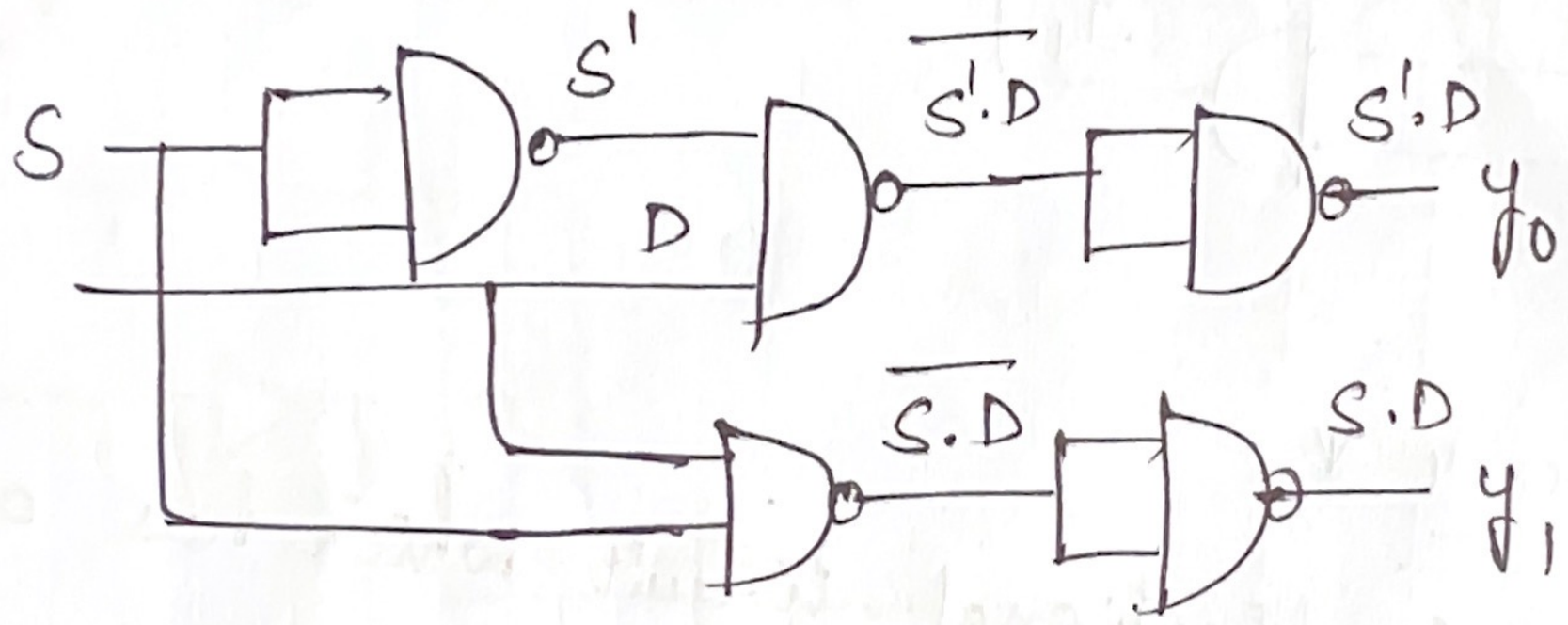
$$y_0 = S'D$$

$$y_1 = SD$$

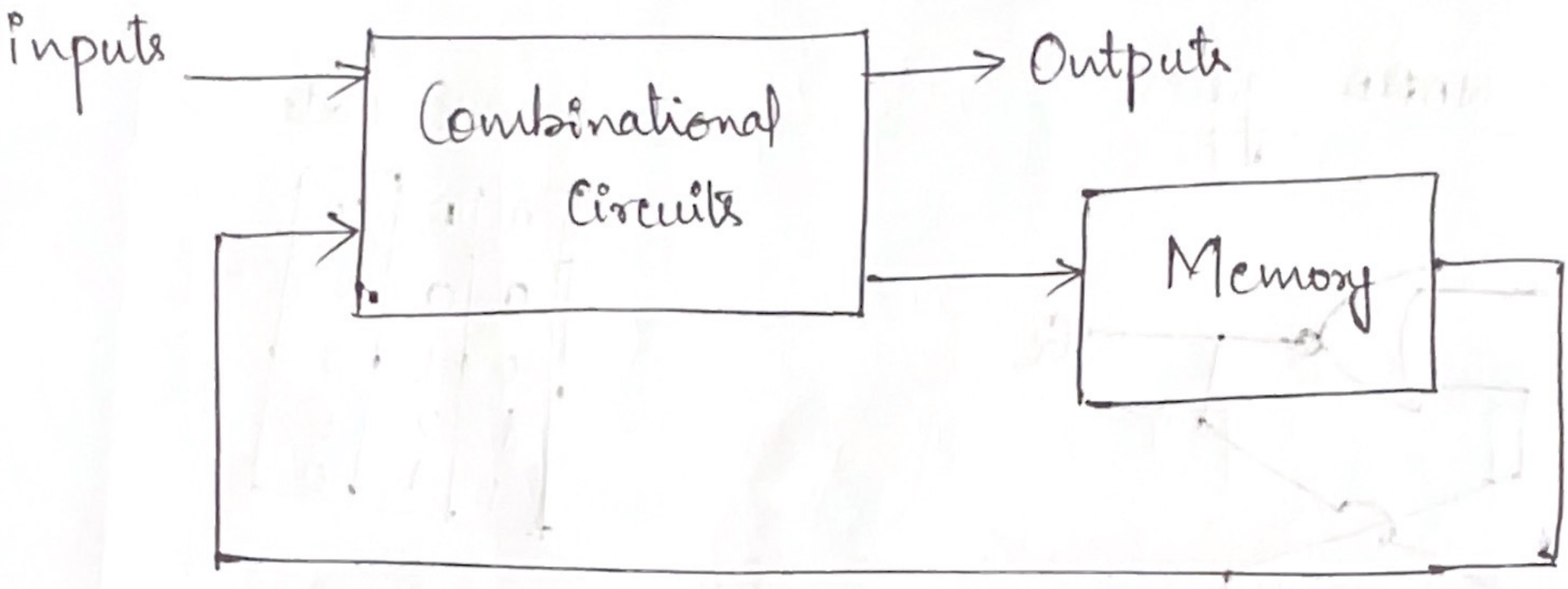
* Logic diagram



* Demux logic diagram using NAND gate.



* Sequential Circuits *



* The circuit in which the output depends on the current inputs and previous outputs is called Sequential Circuit.

Ex:- Flip-flops, Counters, Shift registers

* The storage elements are the devices capable of storing binary information. Ex: Latches, Flip-flops.

* FLIP-FLOPS :-

* Flip flops are digital circuit which are capable of storing data.

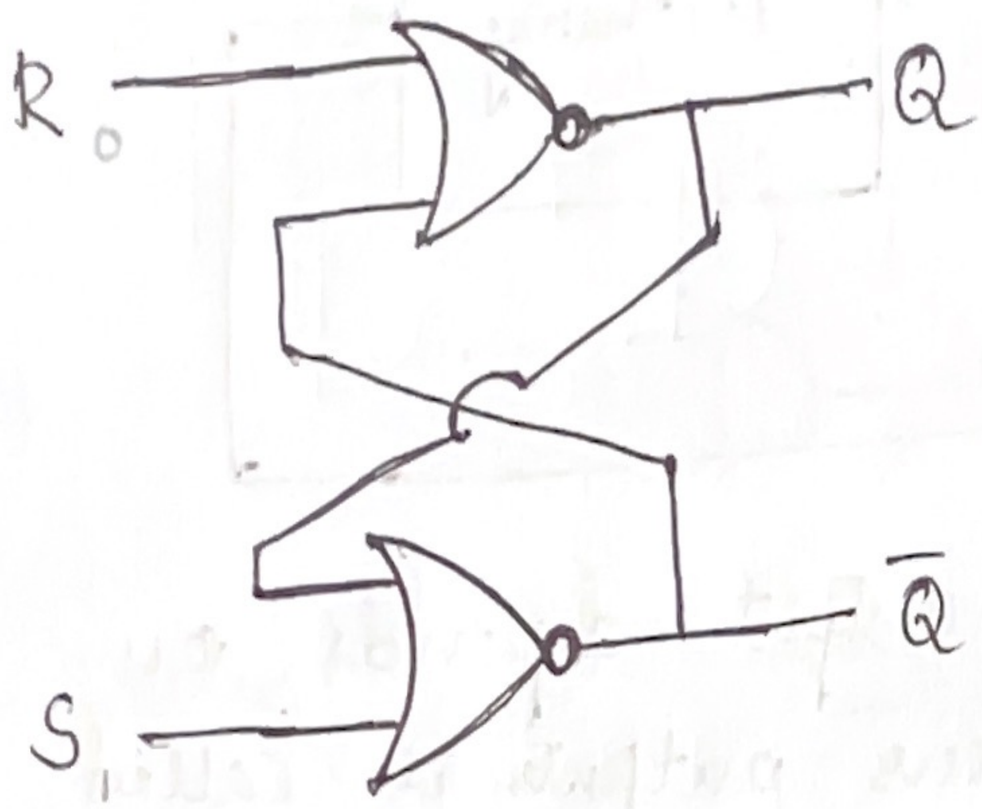
* Each Flip-flop will store one bit data (either 0 (or) 1).

* The stored data remains for indefinite time unless and until it is changed by inputs (or) clock signal.

:- SR Latch (Set - Reset) using NAND / NOR gates :-

* Circuit diagram

Using NOR gate:



NOR Gate

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

Case 1 :- R = 0 and S = 1

Q = 0 and $\bar{Q}$ = 1

R = 0 and S = 0 .

Q = 1 and $\bar{Q}$ = 0 → Memory state .

Case 2 :- R = 1 and S = 0

Q = 0 and $\bar{Q}$ = 1 .

R = 0 and S = 0 .

Q = 0 and $\bar{Q}$ = 1 → Memory state .

Case 3 :- R = 1 and S = 1

Q = 0 and $\bar{Q}$ = 0 (Forbidden state / Not used state)

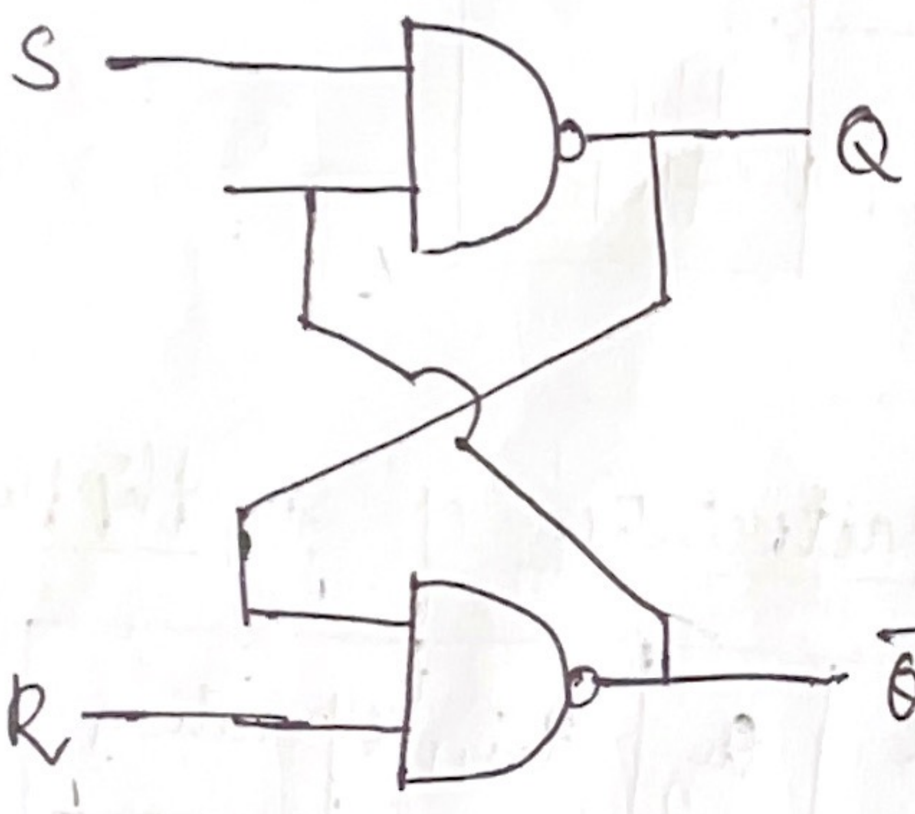
R = 0 & S = 0

Q = 0 & $\bar{Q}$ = 1 (No memory)

∴ Truth table is as follows

S	R	Q	$\bar{Q}$
0	0	Memory	
0	1	0	1
1	0	1	0
1	1	Not used	

* SR Flip-flop can also be constructed using NAND gate. as follows.



Truth table

S	R	Q	$\bar{Q}$
0	0	Not	used
0	1	1	0
1	0	0	1
1	1	Memory	

NAND Gate

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

i) S=0 and R=1

Q=1 and $\bar{Q}$=0

S=1 & R=1

Q=1 and $\bar{Q}$=0 (Memory)

ii) S=1 and R=0

Q=0 and $\bar{Q}$=1

S=1 & R=1

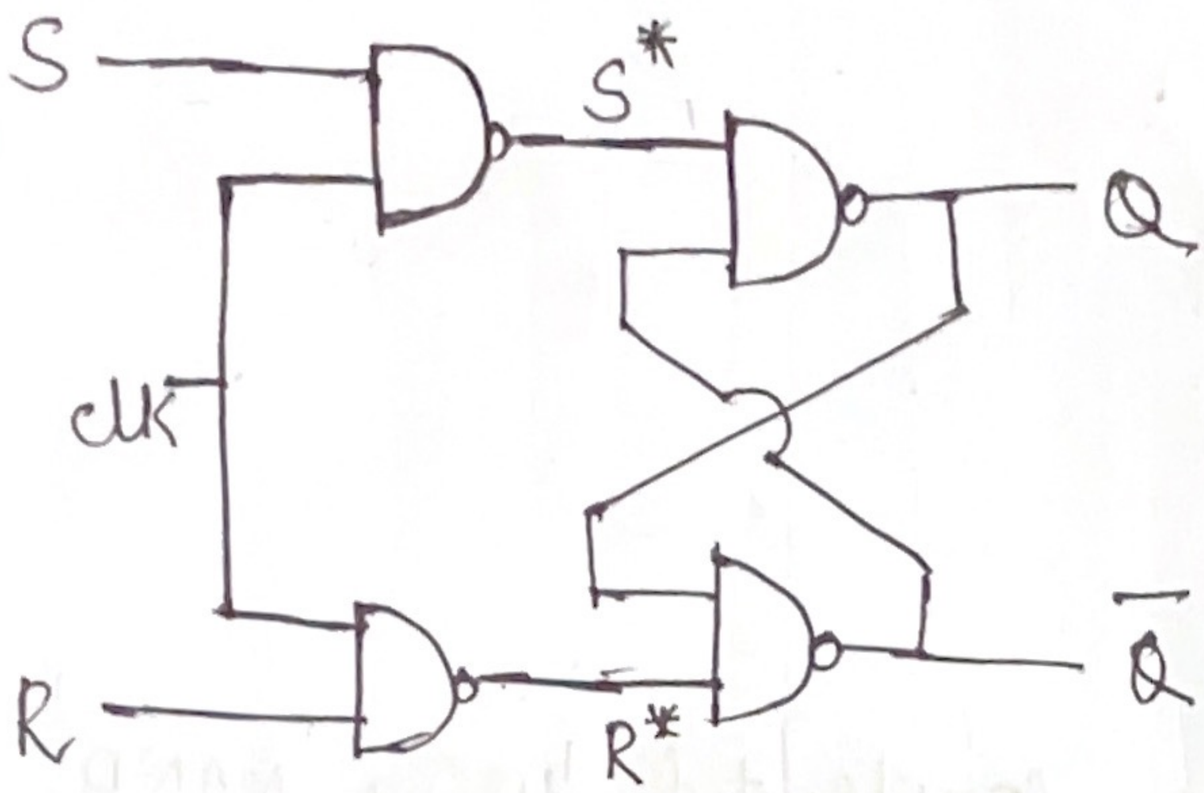
Q=0 and $\bar{Q}$=1 (Memory)

iii) S=0 and R=0

Q=1 and $\bar{Q}$=1 (Forbidden state / NOT USED state)

*SR Flip flop using NAND gate.

Circuit diagram

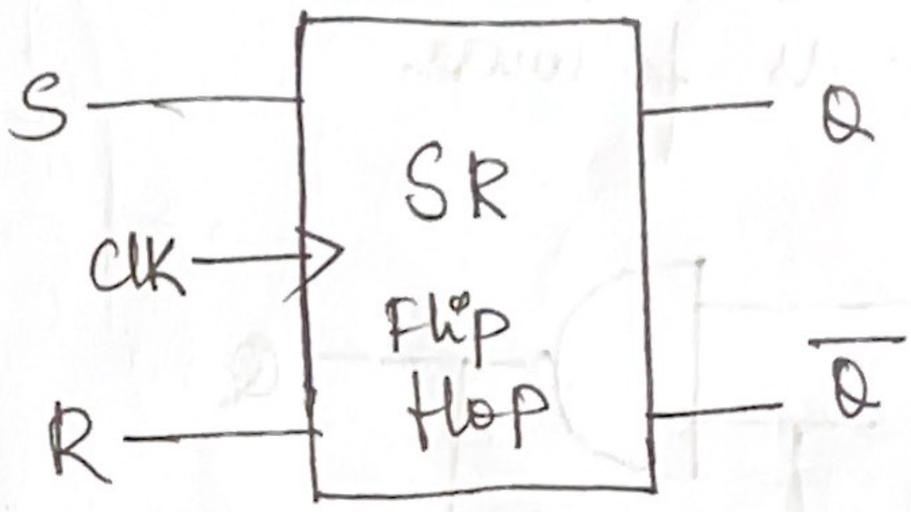


SR Latch T.T

S	R	Q	$\bar{Q}$
0	0	Not	used
0	1	1	0
1	0	0	1
1	1	Memory	

$$S^* = \overline{(S \cdot clk)} = \bar{S} + \overline{clk}$$

$$R^* = \overline{(R \cdot clk)} = \bar{R} + \overline{clk}$$



-: Truth table of SR flip flop:-

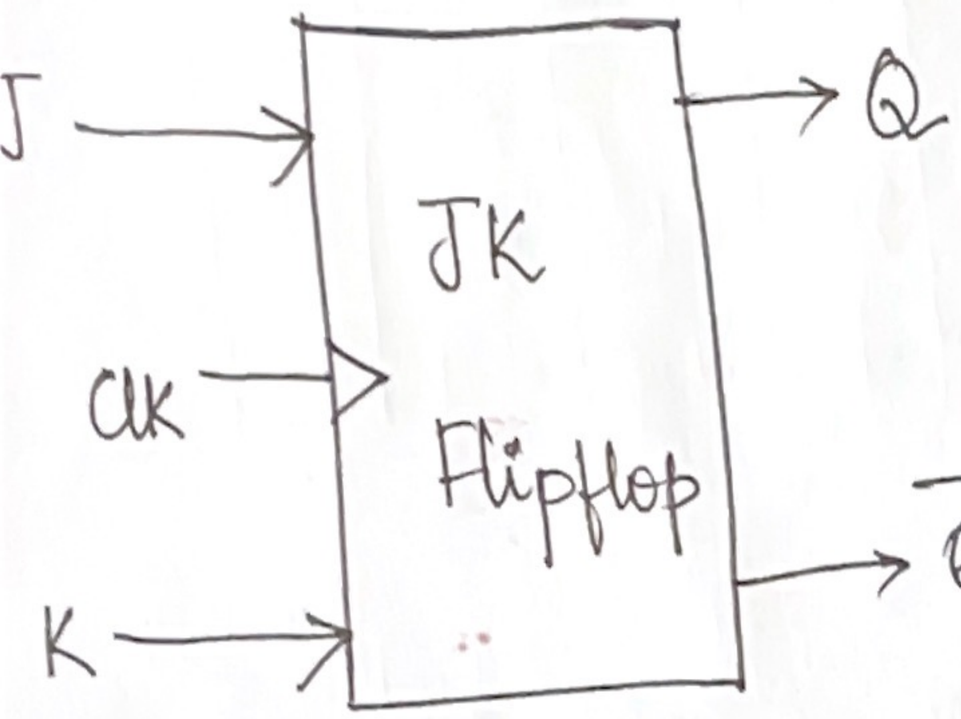
clk	S	R	Q	$\bar{Q}$
0	X	X	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Not used	

Characteristic of SR flip flop

clk	S	R	Q_n	Q_{n+1}	Operation
1	0	0	0	0	Memory(0)
1	0	0	1	1	Memory(1)
1	0	1	0	0	Reset
1	0	1	1	0	Reset
1	1	0	0	1	Set
1	1	0	1	1	Set
1	1	1	0	Not	used
1	1	1	1	Not	used
0	X	X	X	Q_n	Memory

JK Flip flop :- (Jack - Kilby) flip flop

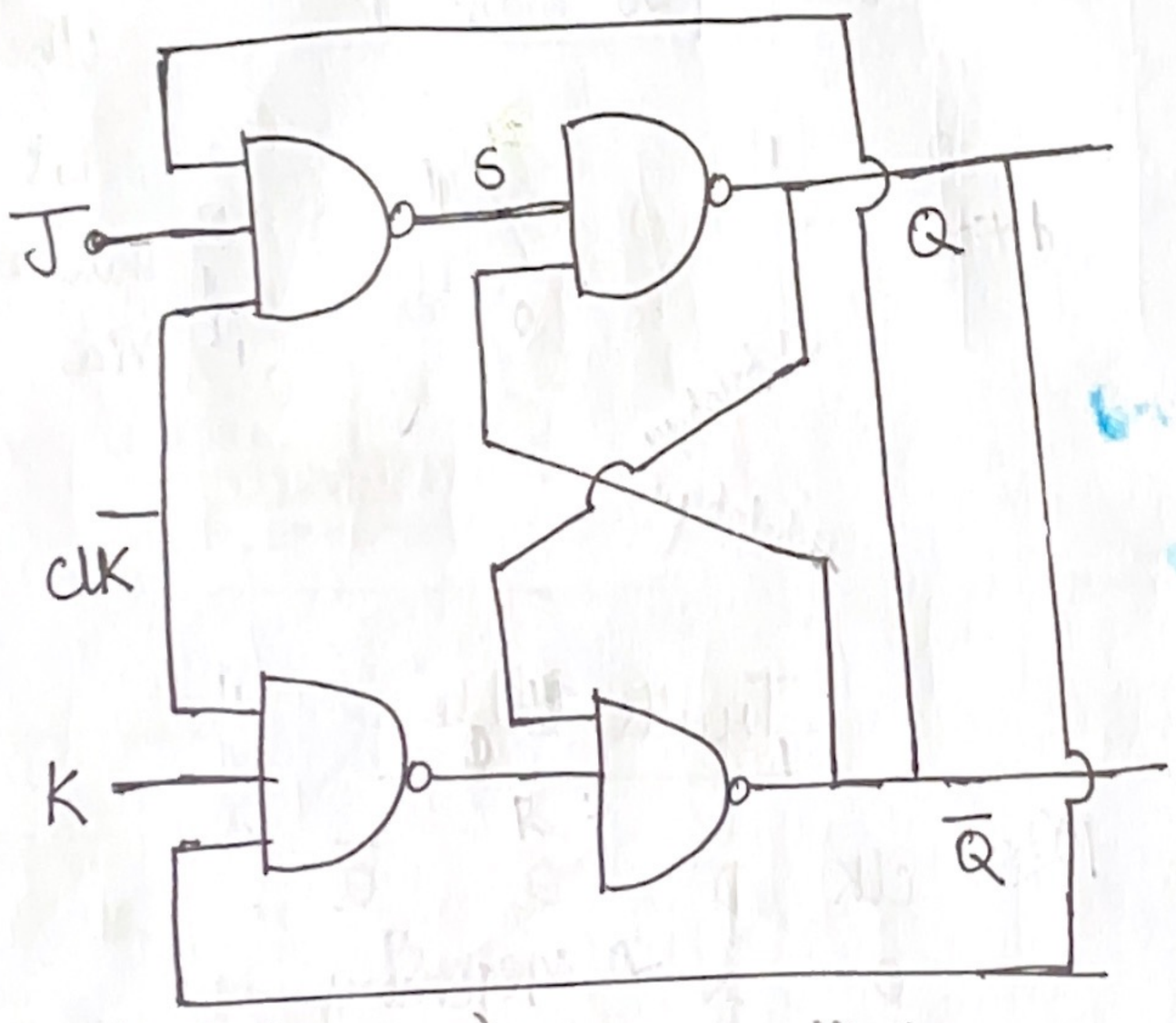
Symbol



Truth table

clk	J	k	Q	$\overline{Q}$
0	x	x	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Toggle	

Circuit diagram



clk=0 → Memory

clk=1, J=0, K=1

Q=0 & $\overline{Q}$=1

clk=1, J=1, K=0

Q=1 & $\overline{Q}$=0.

clk=1, J=1, K=1

assume Q=0 & $\overline{Q}$=1

Q = 0, 1, 0, 1 - - - -

$\overline{Q}$ = 1, 0, 1, 0 . . . .

2I NAND

A	B	C	y
0	0	0	1
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	0

2I NANA

A	B	y
0	0	1
0	1	1
1	0	1
1	1	0

i) J=0 K=1 clk=1

$Q = \overline{1 \cdot \overline{Q}} = \overline{1} + \overline{\overline{Q}} = Q$

$\overline{Q} = \overline{Q \cdot Q} = \overline{0} = 1 \Rightarrow Q=0$

∴ J=0, K=1 ⇒ Q=0 & $\overline{Q}$=1

ii) J=1 K=0 clk=1

$\overline{Q} = \overline{1 \cdot Q} = \overline{Q}$

$Q = \overline{Q \cdot \overline{Q}} = \overline{0} = 1 \Rightarrow \boxed{Q=1 \therefore Q=0}$

iii) J=0 & K=0

S=1 & R=1

∴ Memory

iv) J=1, K=1

Let Q=0 $\overline{Q}$=1

Q = 1, 0 1

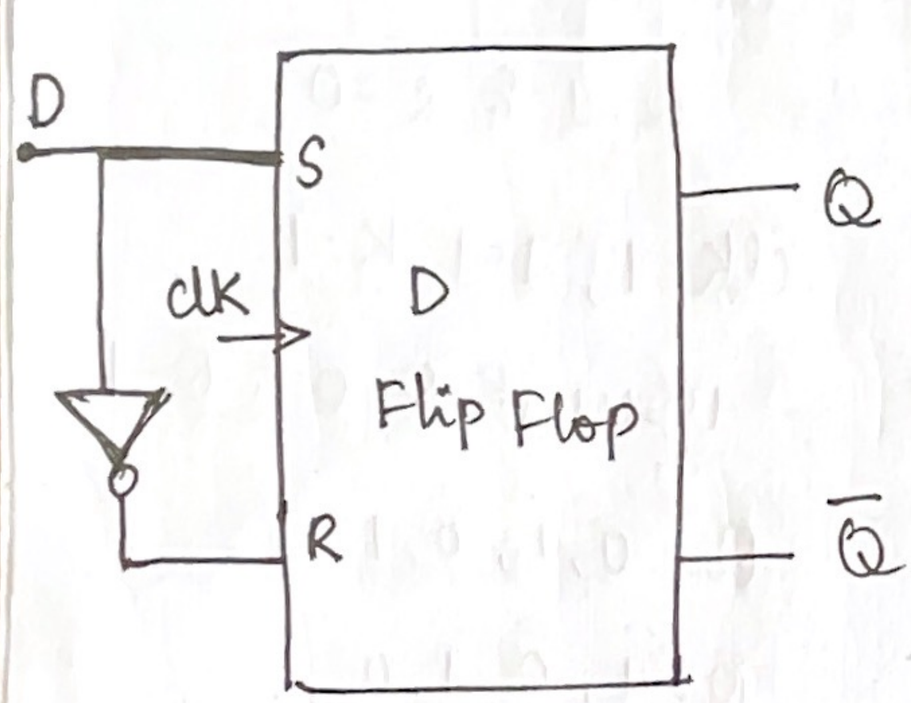
$\overline{Q}$ = 0,

* Characteristics table

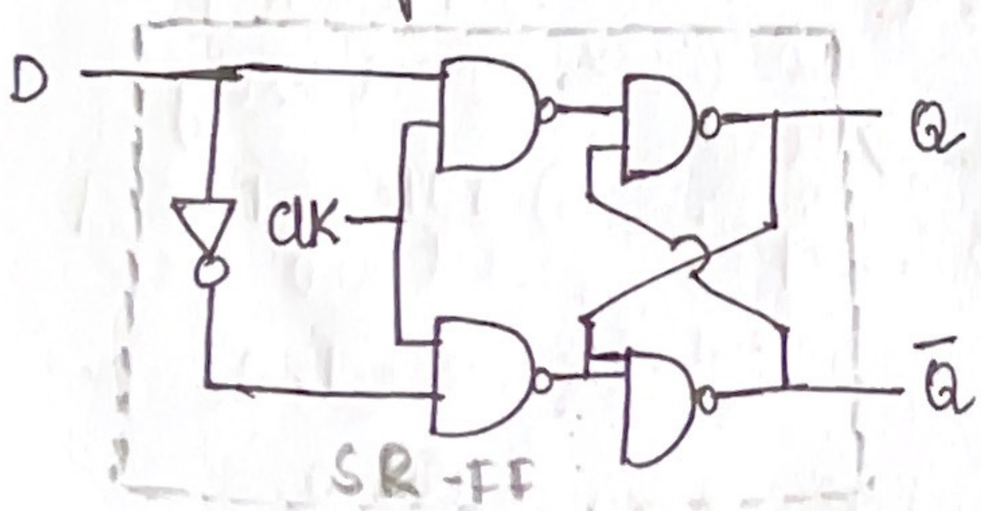
Clk	Q_n	J	K	Q_{n+1}
↑	0	0	0	0
↑	0	0	1	0
↑	0	1	0	1
↑	0	1	1	1
↑	1	0	0	1
↑	1	0	1	0
↑	1	1	0	1
↑	1	1	1	0
0	×	×	×	Memory

* D - Flip Flop (Data Flip Flop) :-

Symbol



Circuit diagram:-



S	R	Q	$\bar{Q}$
0	0	MEMORY	
0	1	0	1
1	0	1	0
1	1	NOT	USED

S&R are complement to each other.

∴ Let us short them 2 terminals via NOT gate

Truth table

clk	D	Q	$\bar{Q}$
0	×	Memory	
1	0	0	1
1	1	1	0

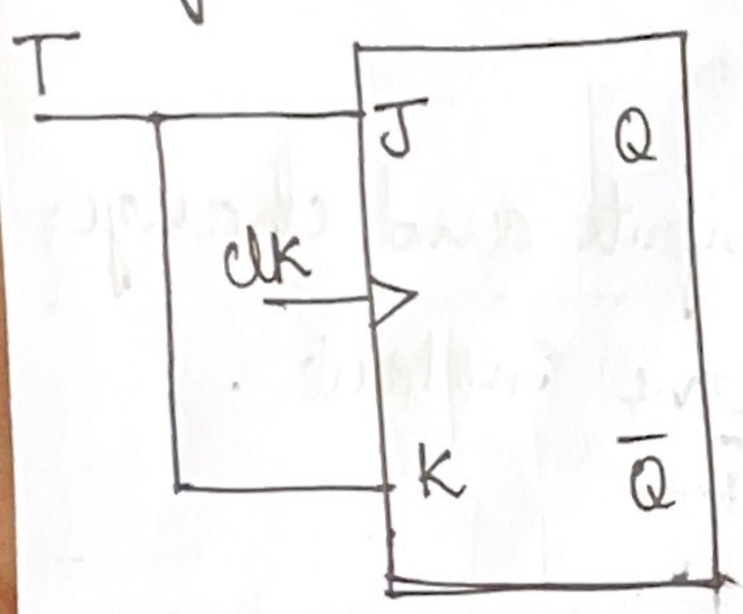
* characteristic table

clk	Q_n	D	Q_{n+1}	$\bar{Q}_{n+1}$
0	x	x	Memory	
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

∴ Present output is equals to present input and independent of Previous outputs.

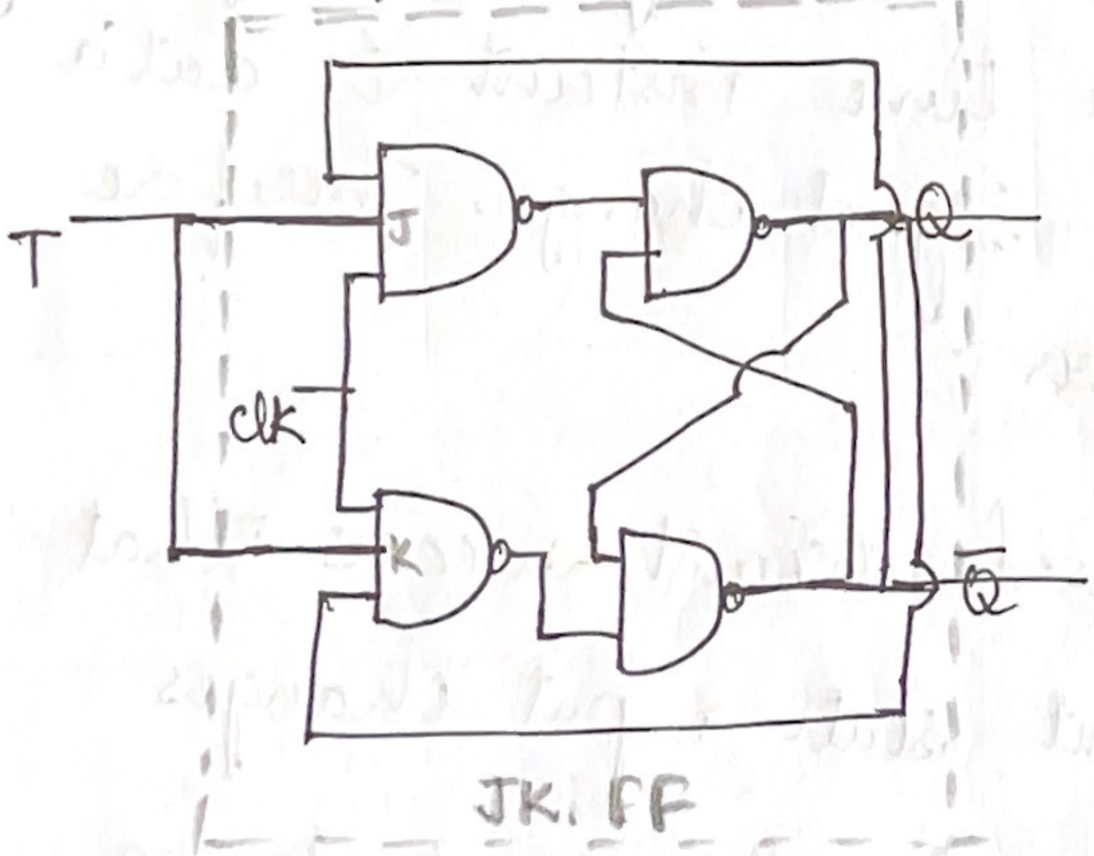
* T flip flop (Toggle flip flop)

Symbol



J	K	Q	$\bar{Q}$
0	0	Memory	
0	1	0	1
1	0	1	0
1	1	Toggle state	

Circuit diagram:



JK FF

Truth Table

clk	T	Q	$\bar{Q}$
0	x	Memory	
1	0	Memory	
1	1	Toggle	

∴ To get Toggling output T should be 1 otherwise the flip flop is in memory state.

*Characteristic Table

Clk	Qn	T	Q_{n+1}	$\bar{Q}_{n+1}$
0	x	x	Memory	
1	0	0	Memory	
1	0	1	1	0
1	1	0	Memory	
1	1	1	0	1

:Conclusion on Flip flops:-

* Flip flops are the buildeng blocks of sequential circuits. these are built from latches.

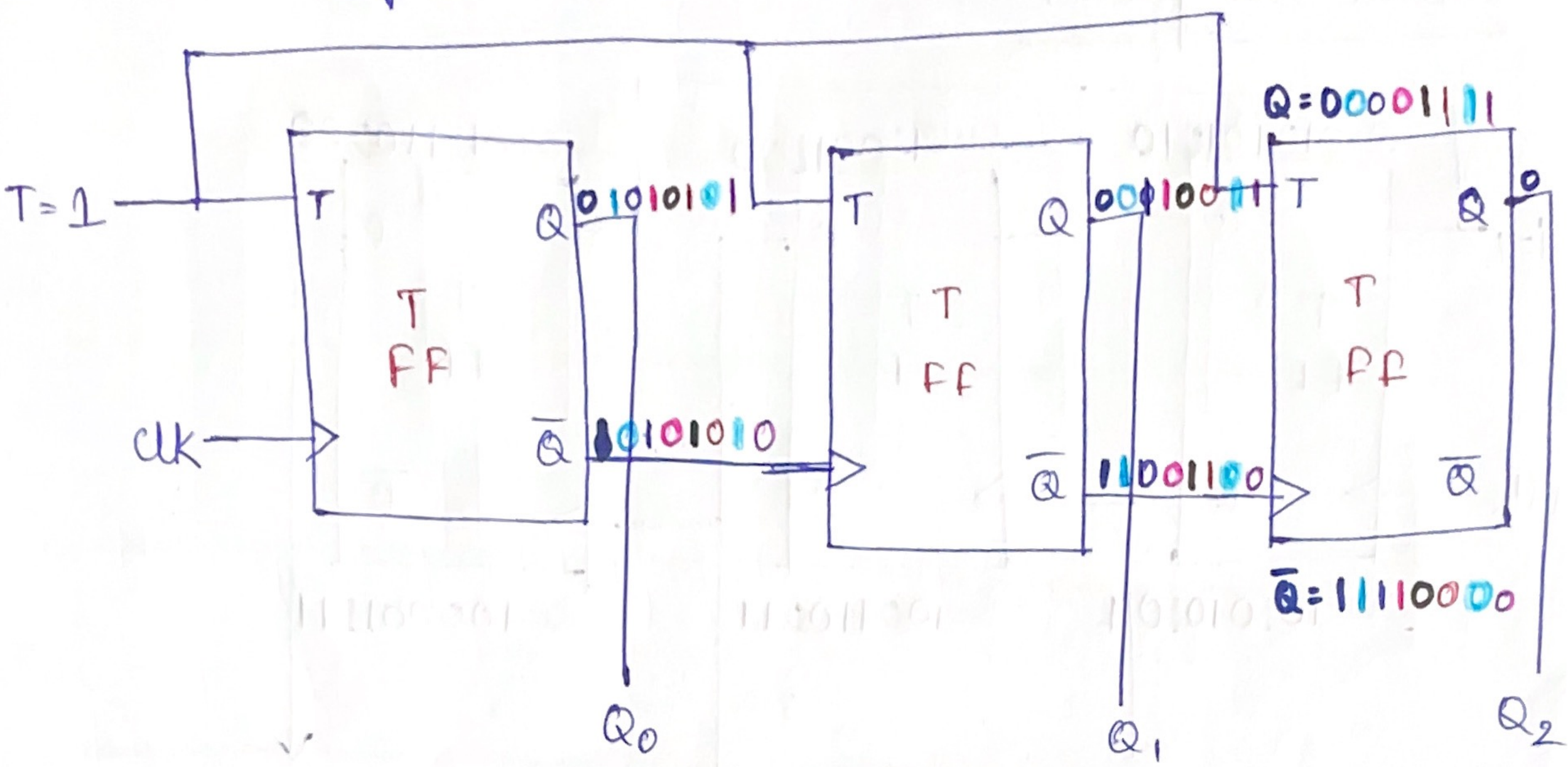
* Flip-flops continuously checks its inputs and changes its outputs correspondingly only at time instant. determined by the clock signal.

* Flip-flops are sensitive to signal change. they transfer data only at singe time instant & data can't be changed till next signal change. therefore they are used as registers.

* It is an edge triggered circuit means that the output and the next state input changes when there is a change in the clock pulse whether it can be +ve / Negative.

* 3 bit Asynchronous up-counter *

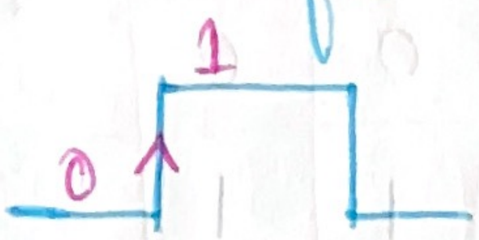
Circuit diagram:-



Q_2	Q_1	Q_0	Counter state
0	0	0	initially 0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Note:-

clock=1 ⇒ Positive edge of clock which is nothing but transition from 0 to 1



↓

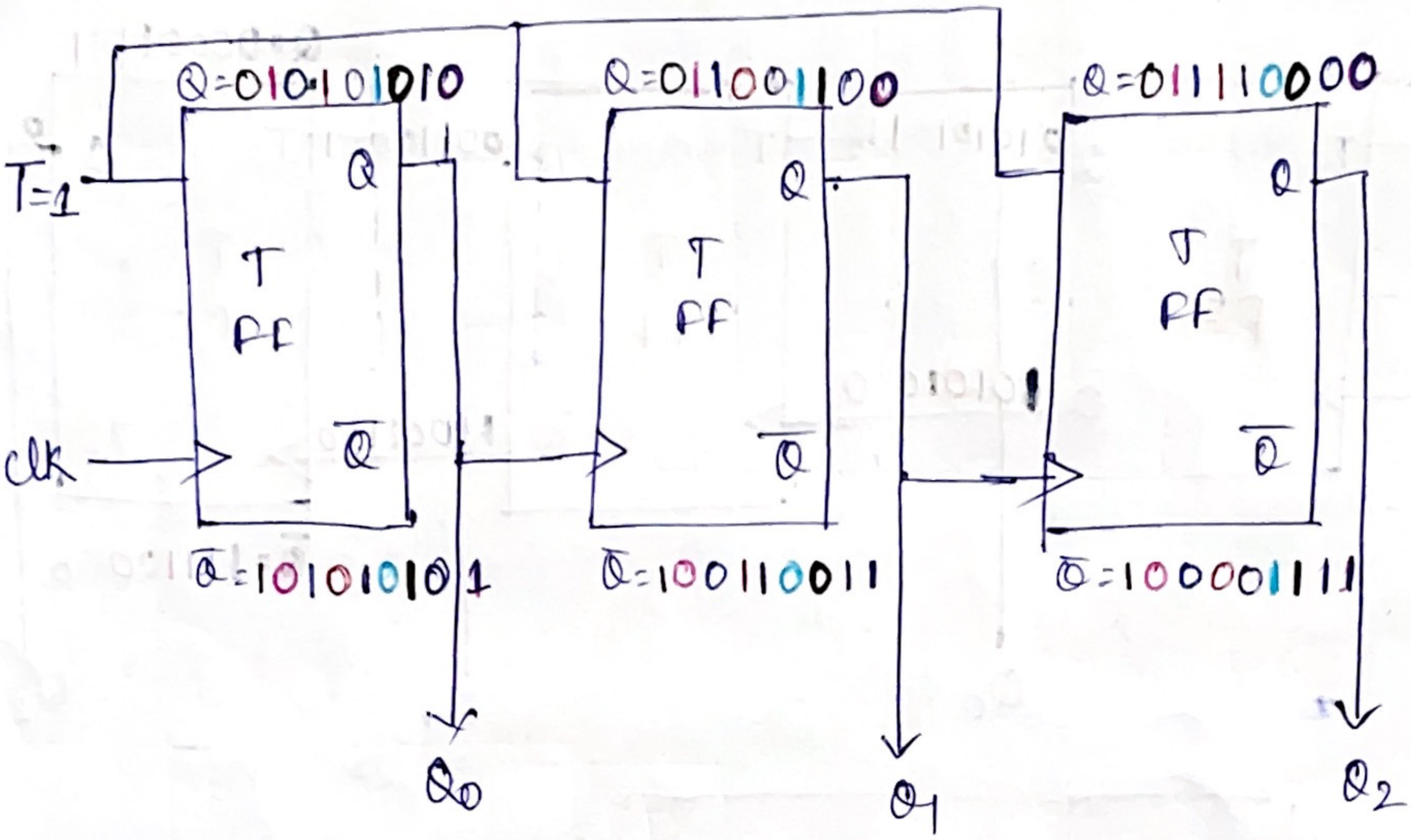
0 to 1 transition is positive edge.

J-Flip flop TT

clk	T	Q	$\bar{Q}$
0	X	Memory	
1	0	Memory	
1	1	Toggle.	

* 3 bit Asynchronous down Counter

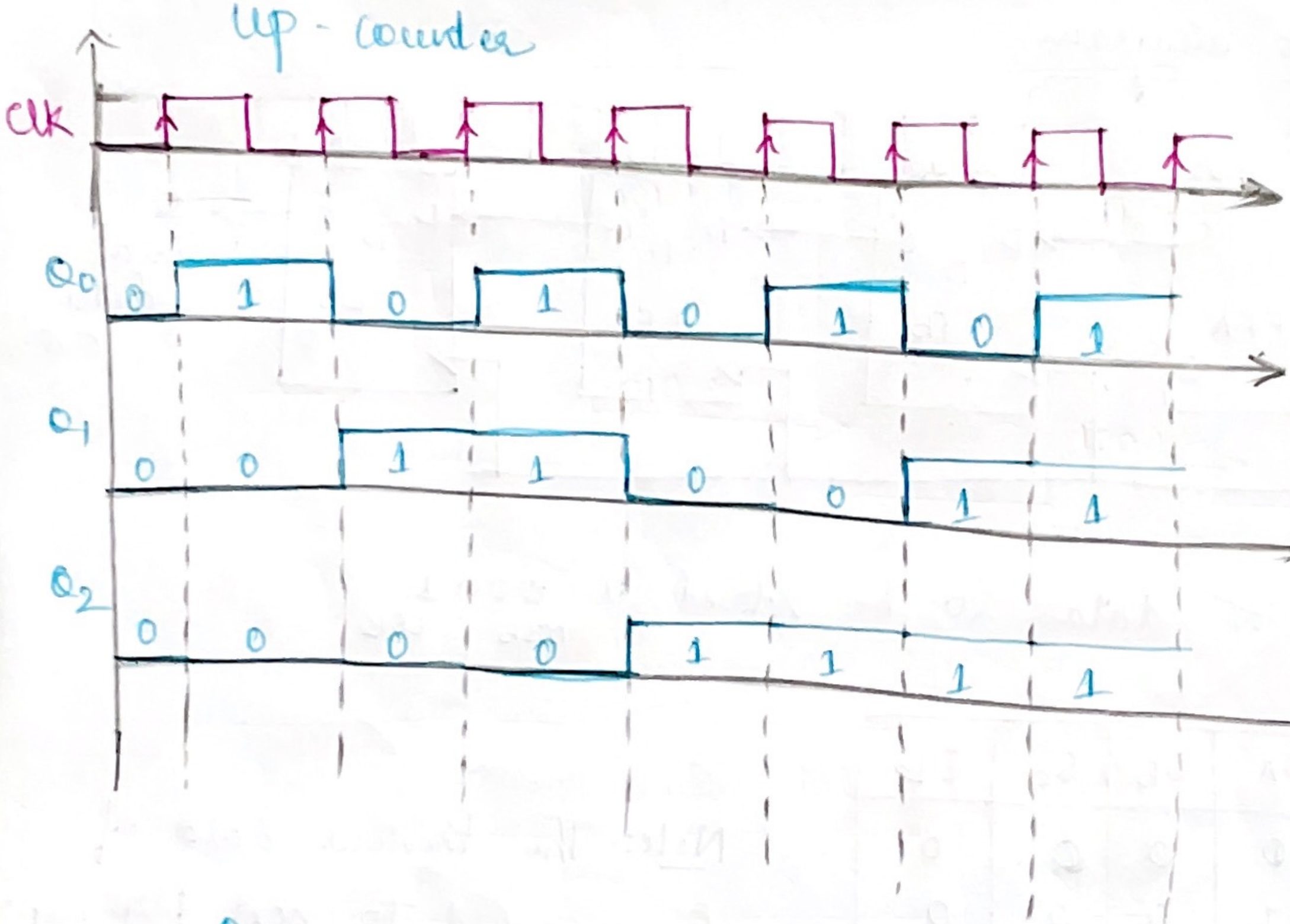
Circuit diagram :-



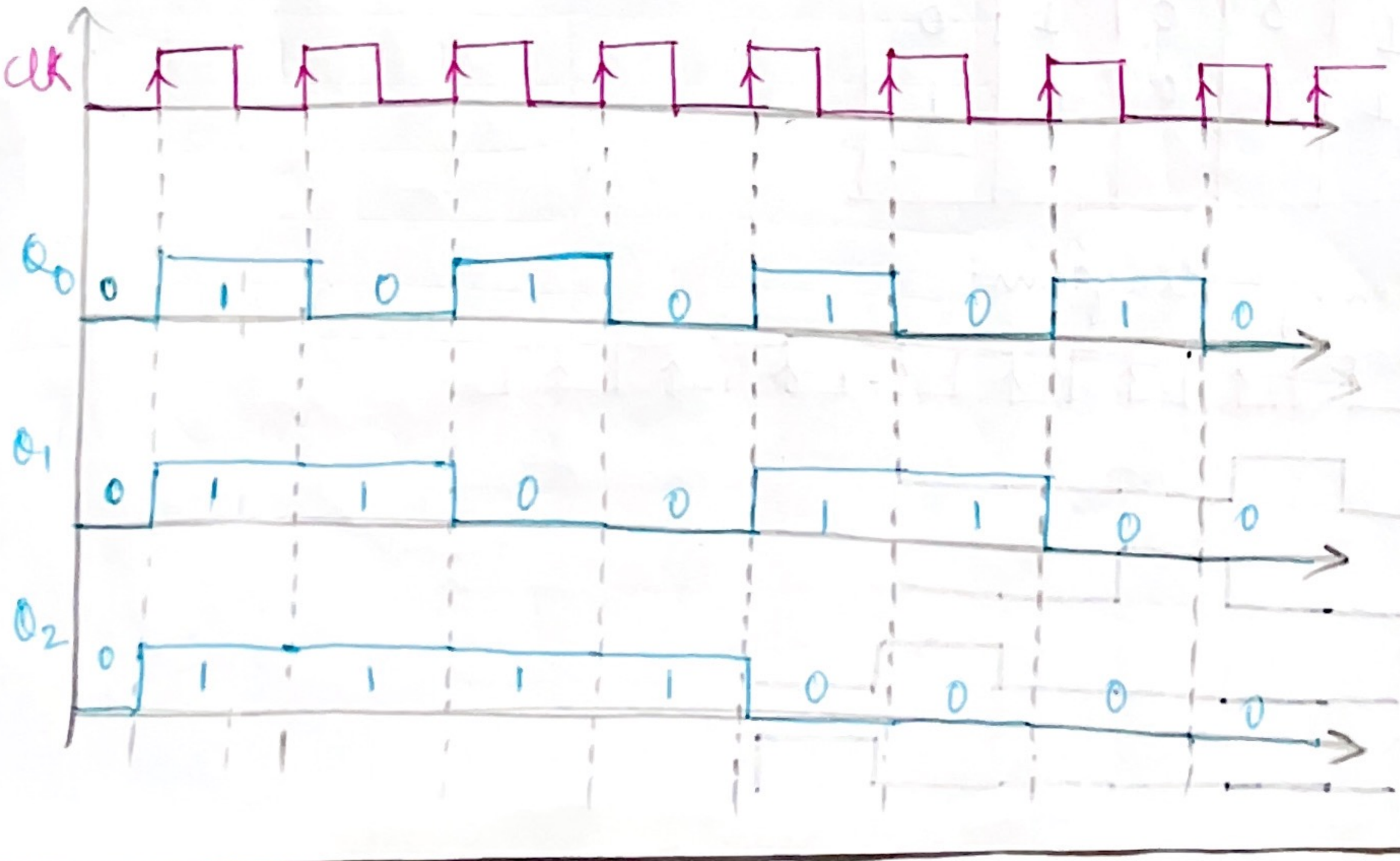
clk	Q_2	Q_1	Q_0	Counter state
✱	0	0	0	initially (0)
↓	1	1	1	7
↓	1	1	0	6
↓	1	0	1	5
↓	1	0	0	4
↓	0	1	1	3
↓	0	1	0	2
↓	0	0	1	1
↓	0	0	0	0

Timing diagram:-

Up - counter

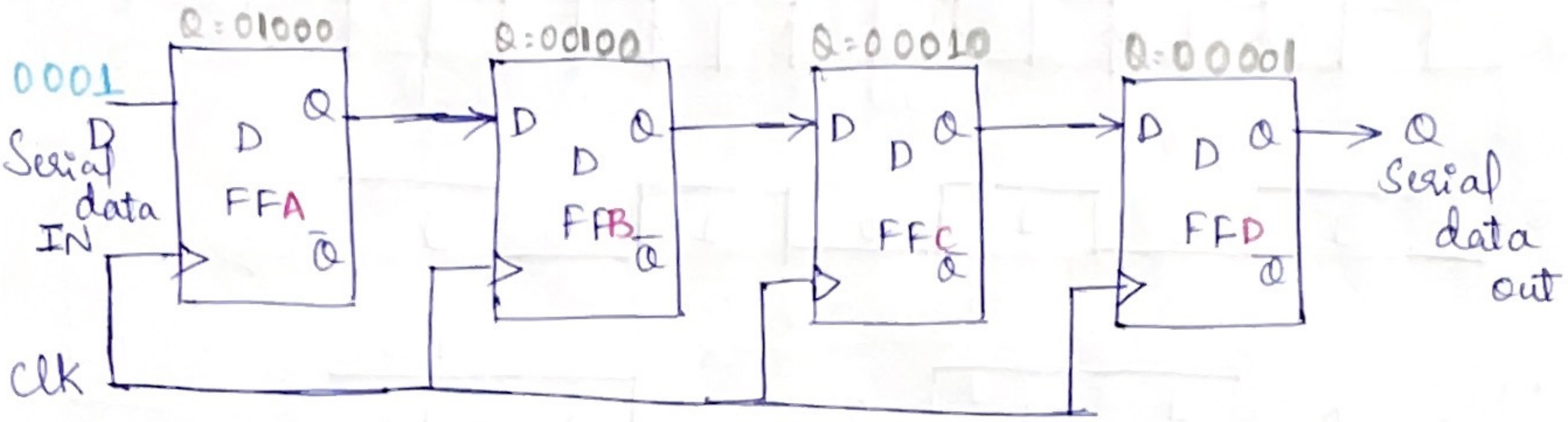


Down counter



4 Bit Serial i/p - Serial o/p Shift register

Circuit diagram:-



* Let the data to be stored is 0001
MSB LSB

clk	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1

Note:- The Previous data of Q is passed to next flip flop's input.

Timing diagram:-