

DIGITAL LOGIC

Ankit Sir

Uncode GATE Computer Science
GATE AIR 1 2014, 2018

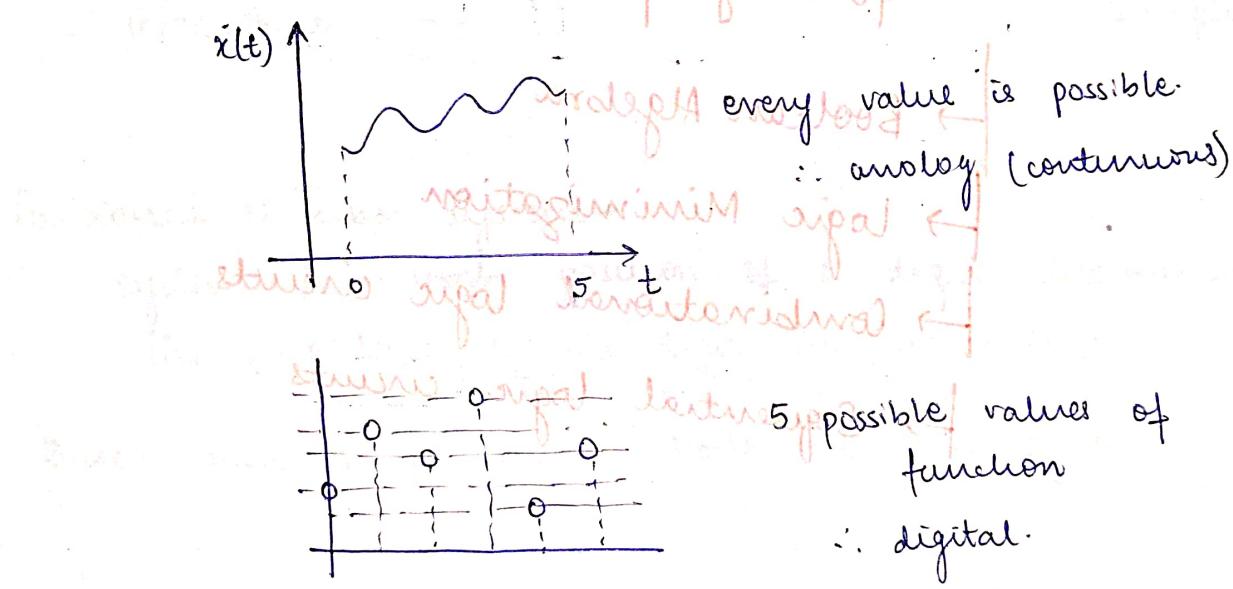
58 lectures

Digital logic -

- ① Digital has been derived from the word digit
- ② It represents a system with finite number of possibilities

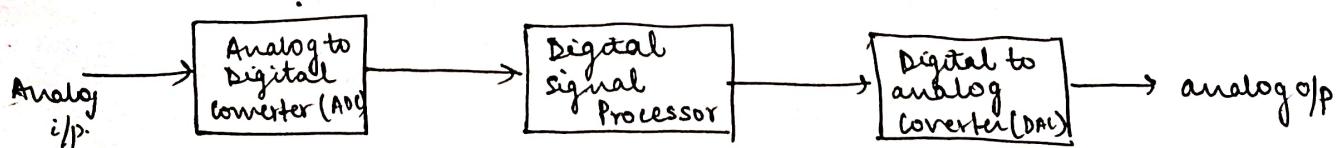
Analog v/s Digital -

→ Analog signals are continuous while digital signals are discrete.



→ Analog = infinite possibilities

Digital = finite possibilities



और भी गम हैं जमाने में।

5.2 TIME

- signal latencies

Tip: how fast work happens need new latencies
utilization for reducing latency after multiple or successive ST

Syllabus

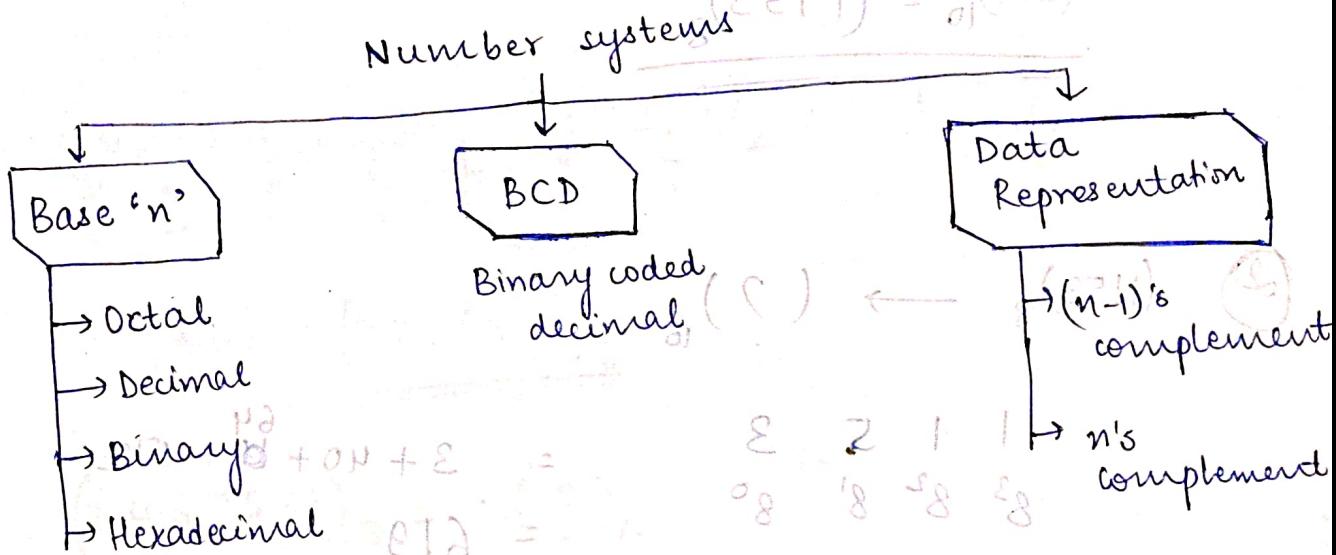
- patterns are simple blocks for us → latencies of patterns
- latencies always increase over time →
 - Number system
 - Computer Arithmetic (fixed point and floating point)
 - Boolean Algebra
 - logic Minimization
 - Combinational logic circuits
- for vector
 - Sequential logic circuits



→ patterns are simple blocks for us → latencies of patterns
utilization of latency = patterns
utilization of latency = latencies



NUMBER SYSTEMS



Positional Number system -

system in which position of a digit determines the weightage of the digit in the number.

Base:- number of unique digits in a no. system.

$$B = 8 + 1 \cdot 8^0$$

Base

2

3

8

10

16

Terminology

Binary

Ternary

Octal

Decimal

Hexadecimal

Range

0, 1

0, 1, 2

0-7

0-9

0-9, A-F

$$\epsilon^{(001)} = \alpha^{(P)}$$

$$\epsilon^{(001)} = \beta^{(Q)}$$

$$\epsilon^{(001)} = \gamma^{(R)}$$

$$\epsilon^{(001)} = \delta^{(S)}$$

$$\epsilon^{(001)} = \epsilon^{(T)}$$

$$\epsilon^{(001)} = \zeta^{(U)}$$

①

$$(619)_{10} \rightarrow (?)_8$$

$$\begin{array}{r} 619 \\ 8 \overline{)77} \\ 8 \overline{)9} \\ 8 \end{array} \quad \begin{array}{l} \text{NUMBER} \\ \text{BASE} \end{array}$$

1 = 1 ↑

$$\therefore (619)_{10} = (1153)_8$$

②

$$(1153)_8$$

$$\rightarrow (?)_{10}$$

$$\begin{array}{r} 1 & 1 & 5 & 3 \\ 8^3 & 8^2 & 8^1 & 8^0 \end{array}$$

$$\begin{aligned} &= 3 + 40 + 40 + 512 \\ &= 619 \end{aligned}$$

$$\therefore (1153)_8 = (619)_{10}$$

③

$$(1001)_2 = (?)_3$$

$$\begin{array}{r} 1 & 0 & 0 & 1 \\ 2^3 & 2^2 & 2^1 & 2^0 \end{array} = 0 \cdot 1 + 8 = 9$$

$$\therefore (1001)_2 = (9)_{10}$$

$$(9)_{10} = (?)_3$$

$$\begin{array}{r} 9 \\ 3 \overline{)3} \\ 3 \overline{)0} \\ 1 \end{array} \quad \begin{array}{l} \text{divisible by 3} \\ \text{remainder 1} \end{array}$$

$$\therefore (9)_{10} = (100)_3$$

$$\therefore (100)_2 = (100)_3$$

$$\textcircled{4} \quad (1100101)_2 = (?)_8$$

~~$\frac{001}{\perp} \frac{100}{4} \frac{101}{5}$~~ for answer ~~$(1100101)_2 = (145)_8$~~

$$\textcircled{5} \quad (10010001110011010001)_2 = (?)_8$$

~~$\frac{1}{2} \frac{1}{2} \frac{1}{1} \frac{6}{2} \frac{3}{3} \frac{2}{2} + 1$~~ for answer ~~$(11)_8$~~

$\therefore \text{Ans} = \underline{\underline{(2216321)_8}}$

$$\textcircled{6} \quad (2467531)_8 = (?)_{10}$$

$$(1010011011110101011001)_2$$

$$\textcircled{7} \quad (7542106)_8 = (?)_2$$

$$(11110110001010001000110)_2$$

$$\textcircled{8} \quad (0100) \frac{(ESE)}{4} (011) \frac{(NNJ)}{B} (101) \frac{(000)}{\perp} (1001) \frac{(100)}{2} (100) \frac{(100)}{C} = (?)_{16}$$

$\therefore \text{Ans} \rightarrow \underline{\underline{(4BB12C)}_{16}} = x(ESE)$

$$\textcircled{9} \quad (4BC\ DA975)_{16} = (?)_2$$

$$(010010111100110110101001101110101)_2$$

Ques

$$\text{If } (2.3)_{\text{base } 4} + (1.2)_{\text{base } 4} = (y)_{\text{base } 4}.$$

What is value of y ?

$$(2.3)_4 = \frac{2 + 0.25 \times 3}{4} = 2.75_{10}$$

$$(1.2)_4 = \frac{1 + 0.25 \times 2}{4} = 1.5_{10}$$

$$2.75 + 1.5 = (4.25)_{10}$$

$$4 \mid 4 \quad 0.25 \times 4 = 1 \uparrow$$

$$\therefore \underline{\text{Ans}} - (10.1)_4$$

$$(100.1101011110110010101)$$

$$\begin{array}{r} 2.3 \\ 1.2 \\ \hline 5 \end{array} \quad 4 \mid 5 \quad \underline{\text{Ans}}$$

$$\begin{array}{r} 10.1 \\ 01000100010 \end{array} \quad 4 \mid 10 \quad (2015 \text{ Ans})$$

Ques

$$\text{Given: } (135)_x + (144)_x = (323)_x$$

What is value of x ?

$$(135)_x = (x^2 + 3x + 5)_{10}$$

$$(323)_x = (3x^2 + 2x + 3)_{10}$$

$$(144)_x = (x^2 + 4x + 4)_{10}$$

$$x=2$$

$$x=6$$

$$(1010 \text{ Ans}) x^2 + 3x + 5 + x^2 + 4x + 4 = 3x^2 + 2x + 3$$

$$2x^2 + 7x + 9 = 3x^2 + 2x + 3 \quad (x+1)(x+6)=0$$

$$x^2 + x - 6x - 6 = 0 \quad x^2 - 5x - 6 = 0 \quad x^2 - 2x - 3x - 6 = 0$$

$$x(x+1) - 6(x+1) = 0 \quad x(x-2) - 3(x-2) = 0 \quad (x-2)(x+3) = 0$$

(10) $(43)_x = (y^3)_8$ (Ans) \Rightarrow $4x+3 = 8y+3 \Rightarrow 4x = 8y \Rightarrow x = 2y$

$0 \leq y \leq 7 \quad 0 \leq x \leq 14$

No. of possible solutions = 8

$x \geq 5 \quad y \geq 3$

possible soln = $y = 3, 4, 5, 6, 7$

$a_1(x) \leftarrow \underbrace{(0101 \cdot 0110110)}_{5 \text{ soln}}$

(11) $(37.625)_{10} = (?)_2$

$0.625 \times 2 = 1.25 = (1 \text{ L} + 0.25)$

$0.25 \times 2 = 0.5 = 0 \text{ D}$

$250.5 \times 2 = 500$

$\therefore \text{Ans} = (100101.101)_2$

$a_1(S) \leftarrow \underbrace{(21 \cdot 0815)}_{588 \cdot 8888}$

(12) $(37.625)_{10} = (?)_8$

$0.625 \times 8 = 5.000 - 5 \downarrow$

$\frac{5}{8} + 0.75 + 0.375 + 0.0625 = 1.25$

Aus = 251845.5 + PPE

$a_1(251845.5 \cdot NPE) \leftarrow \text{Ans}$

(13)

$$(78.025)_{10} = (?)_{16}$$

$$s(\text{EP}) \Rightarrow x(\text{EP})$$

$$\begin{array}{r} 4 \\ 78 \\ \times 16 \\ \hline 468 \\ 78 \\ \hline 1248 \end{array}$$

$$78 \times 16 = 1248$$

$$\begin{array}{r} 0.025 \times 16 = 0.4 \\ 0.4 \times 16 = 6.4 \\ 0.4 \times 16 = 6.4 \end{array}$$

$\text{Ans} \rightarrow \underline{(4E.066\dots)}_{16}$

0.4

$$\frac{512+128+64+8+4}{128} = \underline{0.625}_{10}$$

(14)

$$(0110110 \cdot 1010)_2 \rightarrow (x)_{10}$$

$$\begin{array}{r} 0110110 \\ \times 2^7 \quad 2^6 \quad 2^5 \\ \hline 32 + 16 + 4 + 2 \end{array}$$

$$32 + 16 + 4 + 2 = 54$$

$$\begin{array}{r} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{array}$$

$$54 = 5 \times 2^4 + 4$$

$$\text{Ans} \rightarrow \underline{54.0625}$$

$$\begin{array}{r} 01010 \\ \times 2^7 \quad 2^6 \\ \hline 128 + 64 + 16 + 8 + 4 \end{array}$$

$$128 + 64 + 16 + 8 + 4 = 204$$

$$204 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3$$

$$0.5 + 0.125 = 0.625$$

$$0.625 = 0.625$$

$$\begin{array}{r} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{array}$$

(15)

$$\begin{array}{r} (101 \cdot 101001) \\ \times 8^2 \quad 8^1 \quad 8^0 \\ \hline 7632 \cdot 15 \end{array} = (?)_{10}$$

$$= 2 + 24 + 64 \times 6 + 512 \times 7 + \frac{1}{8} + \frac{5}{64} = (252 \cdot 58)$$

$$\downarrow 2 - 0002 = 882580$$

$$0.20312$$

$$648 \overline{)130}$$

$$= 2 + 24 + 384 + 3584 + \frac{13}{64}$$

$$2 - 1 = 1$$

$$\frac{128}{200}$$

$$= 3994 + 0.203125$$

$$\frac{192}{80}$$

$$\text{Ans} \rightarrow \underline{(3994.203125)}_{10}$$

$$\frac{64}{160}$$

$$\frac{128}{320}$$

$$\frac{128}{320}$$

(16) $(DC4F \cdot 16)_{16} = (?)_{10}$

~~16³ 16² 16¹ 16⁰~~ = $15 + 16 \times 4 + 256 \times 12 + 16^3 \times 13$

~~16³ 16² 16¹ 16⁰~~ = $15 + 64 + 3072 + 53248$

~~16³ 16² 16¹ 16⁰~~ = 56399

$0.\overline{16}_{16} = \frac{1}{16} + \frac{6}{256} = \frac{22}{256} = 0.086$

$\therefore \text{Ans} \rightarrow \underline{(56399.086)}_{10}$

~~(00101110)~~ = $01(A5)$

(17) $0\cancel{1}\underline{110011}\cdot\underline{10110}_2 = (?)_8$

Ans - $(163.54)_8$

• 163.54 base 8
• 163 base 8
• 54 base 8

$01\ 11\ 01\ 01$ base 2 = $(163)_8$

• 163 base 8
• 54 base 8

Binary Coded Decimal

Express each digit as a combination of 4 bits

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

$$(74)_{10} = (0111\ 0100)_{BCD}$$

Weighted Codes / Non Weighted Codes

① Weighted Codes :-

binary codes which obey positional weight principle
Each position has specific weight.

Ex → (8421) code

$$\begin{array}{cccc} b_3 & b_2 & b_1 & b_0 \\ 2^3 & 2^2 & 2^1 & 2^0 \\ 8 & 4 & 2 & 1 \end{array}$$

② Non weighted codes :-

In these types of binary codes, positional weights are not assigned.

Ex → Excess 3 code, gray code.

Excess 3 Code -

also called XS-3

Decimal number	BCD	Add 010s	XS-3
0	0000	0000	0000
1	0001	0010	0011
2	0010	0100	0100
3	0011	0110	0111
4	0100	1000	1000
5	0101	1010	1010
6	0110	1100	1100
7	0111	1110	1110
8	1000	0001	1101
9	1001	0011	1111
10	1010	0101	0000
11	1011	0111	0001
12	1100	1001	0010
13	1101	1011	0011
14	1110	1101	0100
15	1111	1111	0101

6 sequences are wanted

Ques
decimal equivalent of XS-3 no. -

BCD	XS-3	Decimal
1100	1010	12
1010	0011	10
0011	0111	3
0111	0101	5
0101	1001	1
1001	1011	9
1011	1101	7
1101	1111	15
1111	0001	2

Ans \rightarrow 970.42

Gray code

- non weighted code (not arithmetic codes)
- no specific weight assigned to a bit position

- only one bit will change each time the decimal number is incremented.



- Also known as
 - mid distance code
 - cyclic code

- arithmetic operations cannot be performed!

Decimal	BCD	Gray	Gray code
0	0000	0	0000
1	0001	1	0001
2	0010	3	0011
3	0011	2	0010
4	0100	6	0110
5	0101	7	0111
6	0110	5	0101
7	0111	4	0100
8	1000	12	1100
9	1001	13	1101

- use E-2X for finding TRICK (K-map)

$\begin{matrix} 0 & 0 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 0 & 1 & 1 \end{matrix}$

$\begin{matrix} 1010 & 1110 & 1100 & 0101 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \therefore 2 \text{ in binary} & & & \\ S = 0011 \text{ in gray code.} & & & \end{matrix}$

7	10	1	3	2
4	5	7	6	
11	18	15	14	
8	9	11	10	
12	13	16	17	

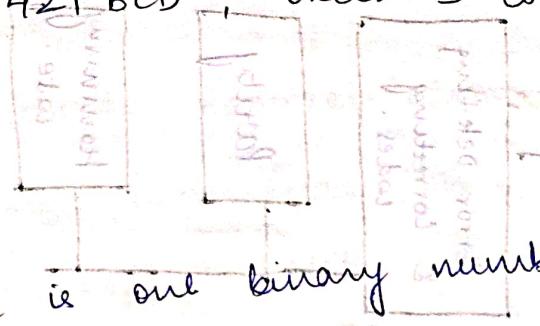
- SN GFP - 11A

Reflective codes

- ① Code is reflective when the code is self complementing
- ② In other words, two's complement of 0 is same 0
- reflecting code for 9 = complement of 0 is same 0
- 8 for 1 7 for 2 6 for 3 5 for 4
- etc and also reflects the Excess 3 (BCD + 0011)

Decimal	BCD	Excess 3 (BCD + 0011)
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Example - 2421 BCD, 5421 BCD, Excess 3 code

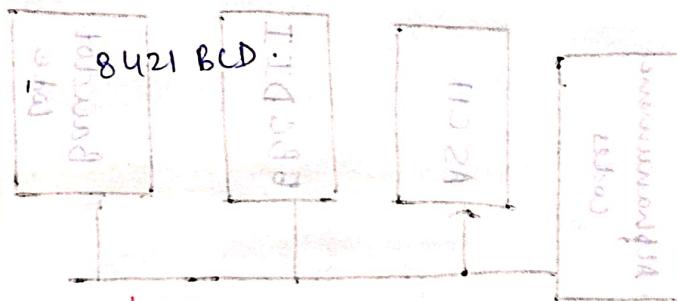


Sequential codes -

each succeeding code is one binary number greater than the preceding code.

Ex →

Excess 3

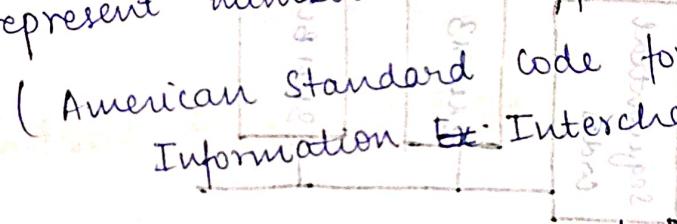


Alphanumeric codes

codes that represent numbers and alphabets

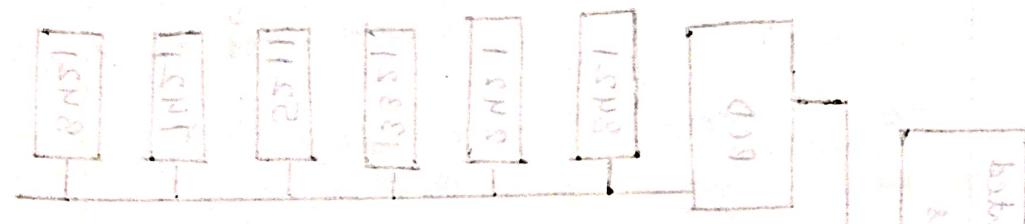
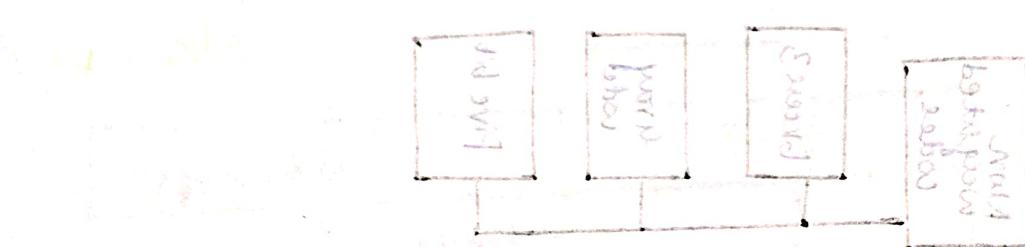
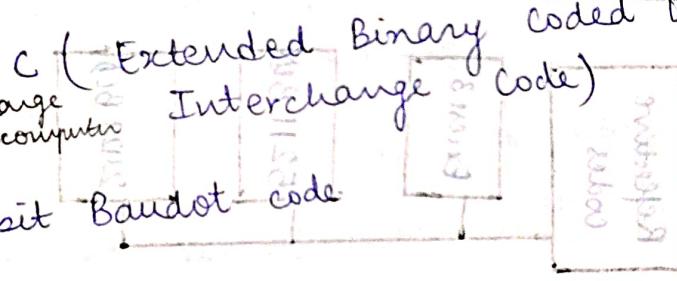
7 bit code → ASCII (American Standard Code for Information Interchange)

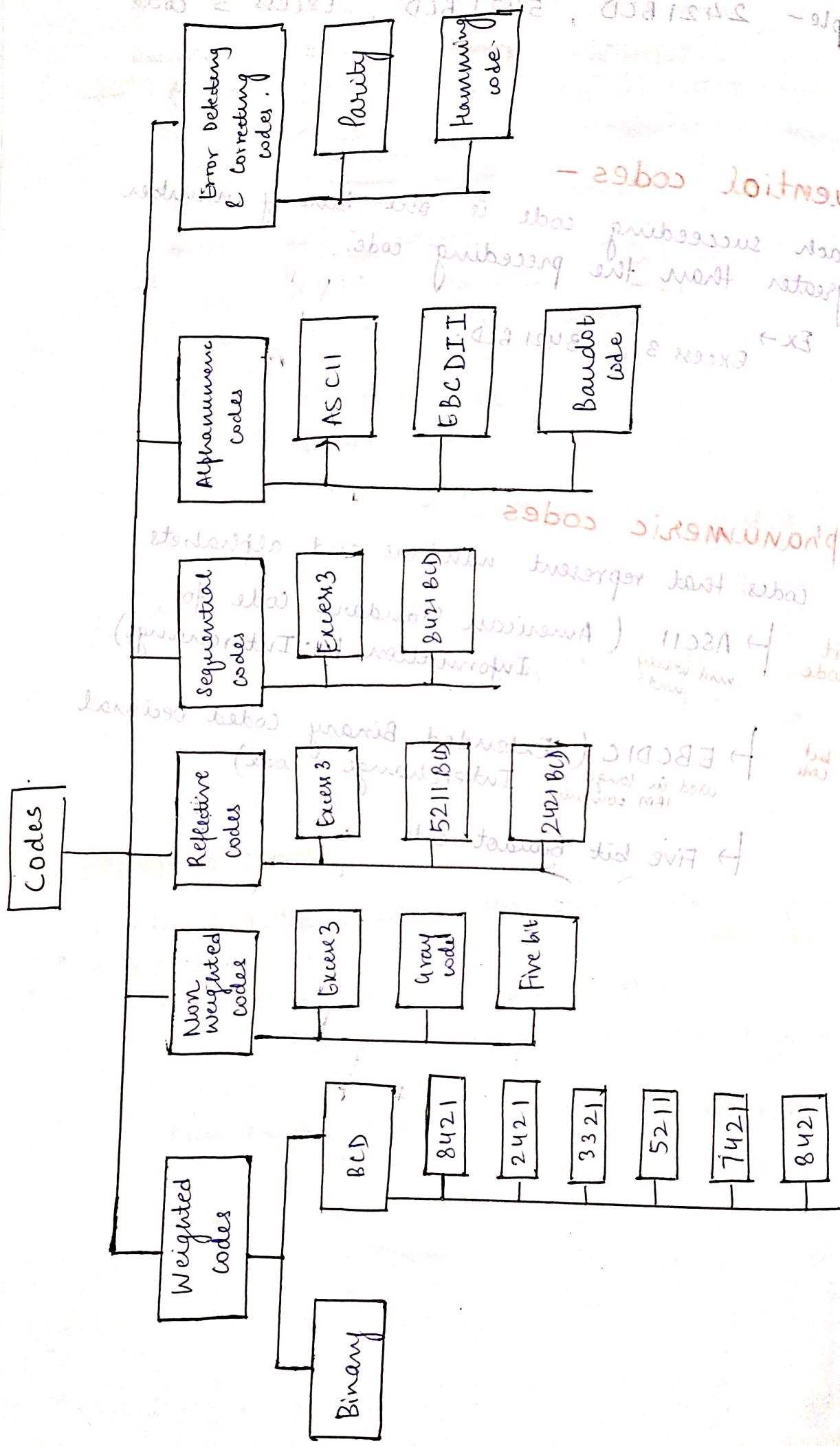
most widely used



- 8 bit code → EBCDIC (Extended Binary Coded Decimal Interchange code)
used in large IBM computers

→ Five bit Baudot code





Data Representation

- Unsigned Magnitude Representation
- Signed Magnitude Representation
- 1's complement Representation
- 2's complement Representation

1. Unsigned Magnitude Representation:

only +ve numbers can be represented.

exclusively 0 to $(2^N - 1)$ numbers.

N bits \Rightarrow maximum possible numbers.

2. Signed Magnitude Representation:

MSB is reserved for sign.

$0 \rightarrow +ve$ $1 \rightarrow -ve$.

Both +ve and -ve numbers can be represented.

N bits we can represent from $(2^{N-1} - 1)$ to $(2^{N-1} - 1)$

Representation	Range
Unsigned Magnitude Representation.	0 to $2^N - 1$
Signed Magnitude Representation.	$-(2^{N-1} - 1)$ to $2^{N-1} - 1$
1's complement	$-(2^{N-1} - 1)$ to $2^{N-1} - 1$
2's complement.	-2^{N-1} to $2^{N-1} - 1$

- ## Overflow
- ① occurs when more bits are required to store the result than is available.
 - ② Overflow cannot occur in addition (or subtraction) if operands have opposite signs.

Storing Real Numbers

2 major approaches of storing real numbers -

→ Fixed point representation.

→ Floating point representation.

1. Fixed point representation -

fixed no. of bits for integer part and fractional part

Ex → IIII.FFFF

Min value = .0000.0001

Max value = 1111.111111111111

3 parts

→ Sign field

→ Magnitude field

→ Fractional field

1 101101 100000

trivial part

trivial part

Unsigned	Fixed Point Representation	Integer	Fraction	
Signed	Fixed Point Representation			
Signed	Fixed Point Representation	Sign	Integer	Fraction

Ques

Assume number is using 32 bit format which reserves

1 bit for sign

16 bits for integer part.

15 bits for fractional part.

Find the value in decimal of the min & max that can be stored.

$$FSI = FSI \cdot F2h = 100000000000000000000000_2$$

$$\text{Min value} = 1000...0 \frac{000...01}{2^{15}} = -2^{-15}$$

$$\text{Max value} = 0111...1 \frac{111...11}{2^{15}}$$

$$= (2^{16}-1) + \frac{1 - (\frac{1}{2})^{15}}{1 - \frac{1}{2}}$$

$$= 2^{16} - 1 - (\frac{1}{2})^{15} = 2^{16} - \frac{1}{2^{15}}$$

2. Floating point representation

does not have specific number of bits for the integer part and fractional part.

Floating point representation has 3 parts

- Sign bit
- Mantissa
- Exponent

IEEE Floating Point Representation

Sign	Exponent	Mantissa
\leftarrow	8	\leftarrow 23 \rightarrow

Smallest number =

1.000...0 00...1

Exponent is in excess 127 notation

$$\text{Smallest no} = -1 * 2^{(1-127)} = -1 * 2^{-126}$$

$$\text{Largest number} = 2^{\text{exponent}} = 2^{127} = 11111111111111111111111$$

$$\text{exponent} = 254 - 127 = 127$$

$$\text{Largest number} = (2^{127} - 1) * 2^{127}$$

$$\frac{1}{2} - \frac{1}{2^2} + \frac{1}{2^3} - \frac{1}{2^4} + \dots$$

Binary floating point representation

Binary floating point representation is different from decimal floating point representation.

Binary floating point representation is based on scientific notation.

Exponent is in excess 127 notation.

Sign bit is 1 for negative numbers.

Exponent is 127 for zero and 254 for infinity.

Mantissa is 1.000...0 for zero and 1.111...1 for infinity.

Exponent is 127 for zero and 254 for infinity.

Mantissa is 1.000...0 for zero and 1.111...1 for infinity.

~~BOOLEAN ALGEBRA~~

LOGIC GATES

Logic gates -
Basic building block of a digital circuit which is used to make all logic decisions.

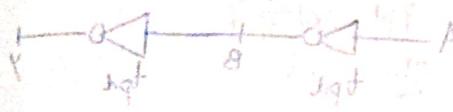
- (a) NOT gate
- (b) BUFFER
- (c) AND
- (d) NAND

- (e) OR

- (f) NOR

- (g) XOR

- (h) XNOR

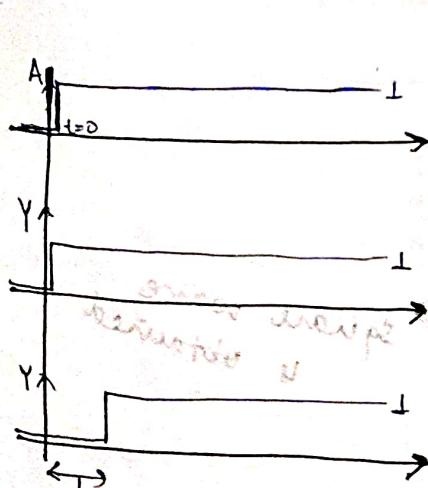


① Buffer

A buffer has the same output as the input but it increases signal strength so that it can travel for longer distance.



$$\text{Relation: } Y = A$$



t_{pd} = propagation delay

time taken by signal to go from the input to the output

A	Y
0	0
1	1

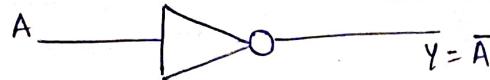
Truth table

tabular representation which tells us when op is 1 it gives value of op for all possible combination of inputs.

② Inverter (NOT)

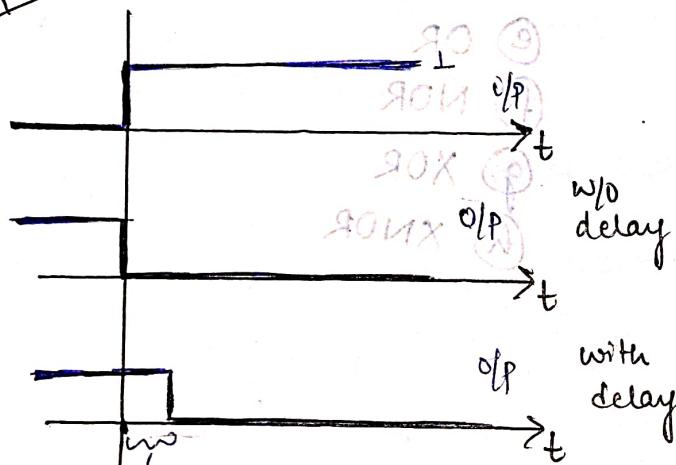
- ① It negates all logic.
- ② Complements the logic.

LOGIC GATES



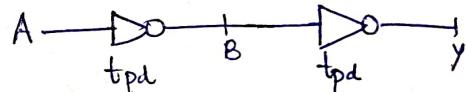
at low & devo time, bubble implies inversion.
In any logic circuit, bubble implies inversion.

waveform



A	Y
0	1
1	0

Cascading of inverters

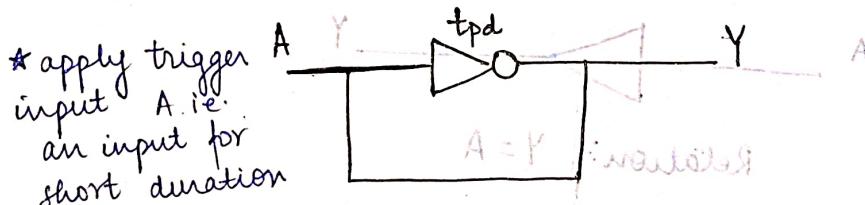


$$B = \bar{A}$$

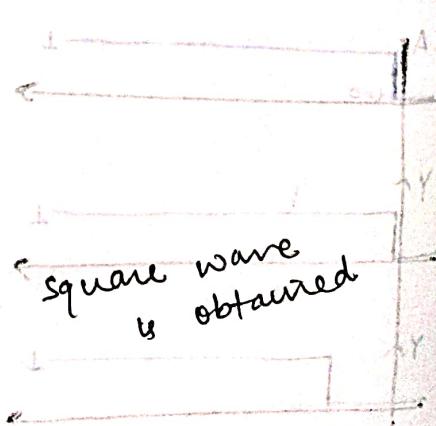
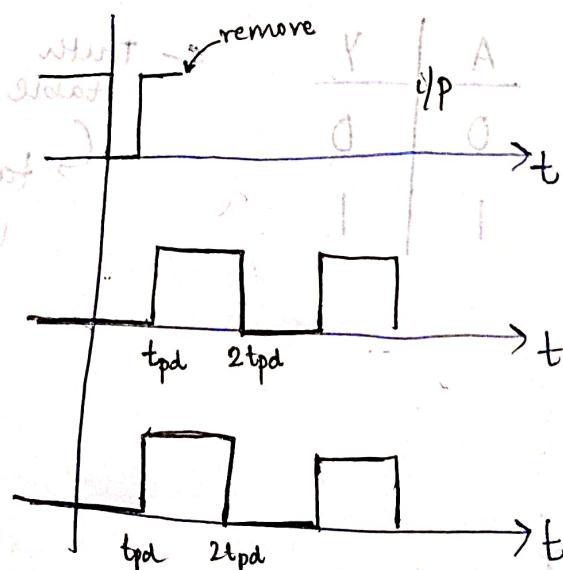
$$Y = \bar{B} = \bar{\bar{A}} = A \quad ①$$

2 inverters in cascade behave like a buffer with delay = $2 \cdot tpd$
exhibited repeat

NOT gate with feedback.



* apply trigger input A i.e. an input for short duration



start with a single pulse and repeat

* if even number of NOT gates are connected in a loop, the output \rightarrow ~~stays at 0 or 1~~ \rightarrow ~~remains stable~~ at 0 or 1.

* if odd number of NOT gates are connected in a loop, o/p becomes square wave.

$$\text{period of square wave} = 2nt_{pd}$$

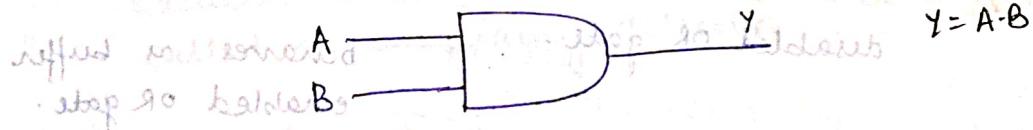
t_{pd} = period of square wave $\rightarrow n = \text{no. of NOT gates}$

$$t_{pd} = \Delta t_A$$

$$(t + \Delta t) + A = J + (A + \Delta t)$$

(3) AND gate

Both i/p or conditions must be true in order for o/p to be true or high.



* follows commutative law

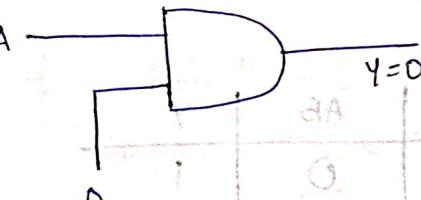
$$A \cdot B = B \cdot A$$

* follows associative law

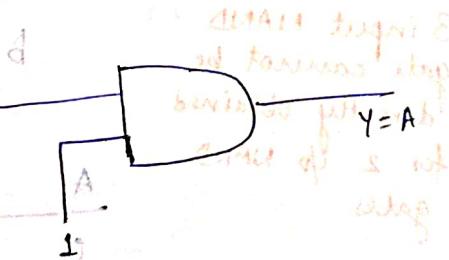
$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

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

Enable and disable AND gate.



The i/p A to o/p so, AND gate is disabled.

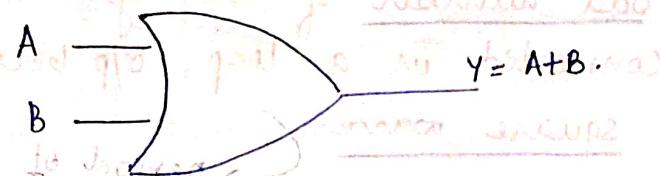


The i/p directly passes to o/p
Enabled AND gate
acts as a buffer.

(4)

OR gate

- atleast one of the inputs must be high in order for o/p to be high.



digit f = 1 for high
digit f = 0 for low

OR gate follows commutative law

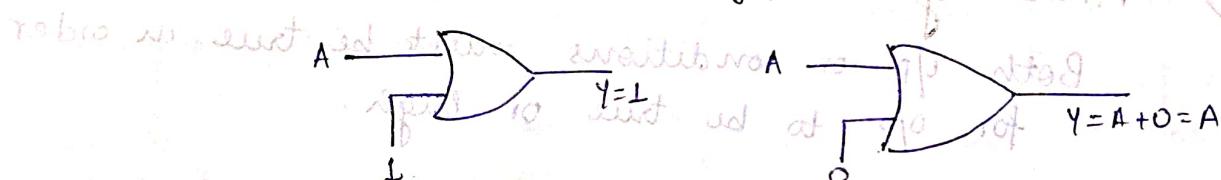
$$A+B = B+A$$

follows associative law

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

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

Enable and Disable OR gate.



disabled OR gate.

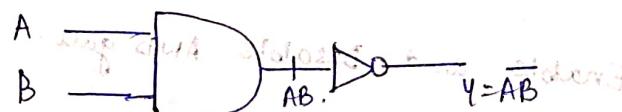
behaves as buffer
enabled OR gate.

(5)

NAND gate

Combination of inverter and AND gate.

$$\text{NAND} = \text{AND} \rightarrow \text{NOT } A \cdot B = \bar{A} \cdot \bar{B}$$



3 input NAND
gate cannot be
directly obtained
for 2 up NAND
gates.

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

* follows commutative law

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

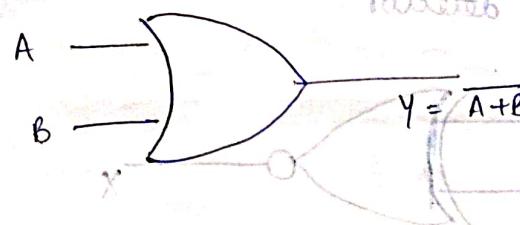
+ does not follow associative law

$$\overline{\overline{A}BC} \neq \overline{A}\overline{BC}$$

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

⑥ NOR gate

Combination of OR gate and NOT gate.



* 3 i/p NOR gate cannot be constructed directly from 2 i/p NOR gates.

2 i/p NOR gate is used.

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

* follows commutative law

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

* does not follow associative law

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

⑦ XOR gate

Exclusive-OR

Called as inequality detector.

gives 1 if two inputs are different & gives 0 if both inputs are same.



odd no.

of 1s in \rightarrow o/p is 1.
if

even no.

of 1s in \rightarrow o/p is 0.
if

A	B	X = A \oplus B
0	0	0
0	1	1
1	0	1
1	1	0

$$A \oplus B = B \oplus A$$

* follows associative law

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

$$\text{if } A \oplus B = C$$

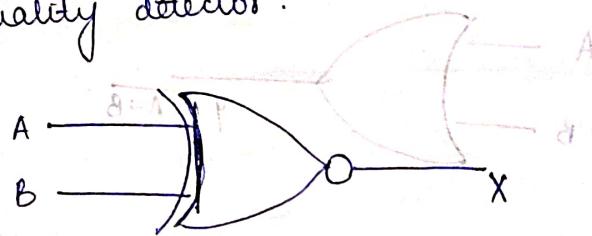
$$\text{then, } A \oplus C = B$$

$$B \oplus C = A$$



⑧ X NOR gate

Exclusive NOR gate
represented by ⑧
equality detector.



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

* follows associative law

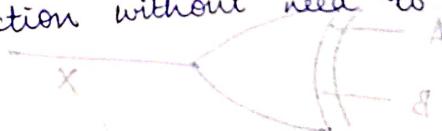
$$AO(BOC) = (AOB)OC$$

* follows commutation law.

$$AOB = BOA$$

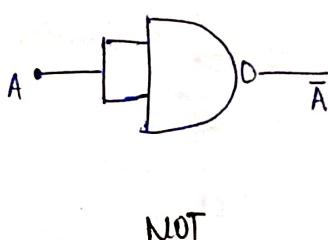
Universal gates

* universal gate is a gate which can implement any boolean function without need to use any other gate type.

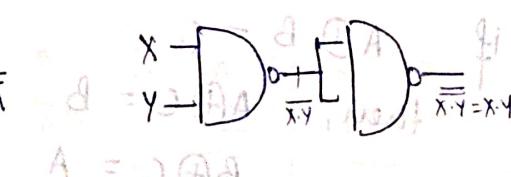


* AND & OR gates are designed from NOR gates (not other way round!!)

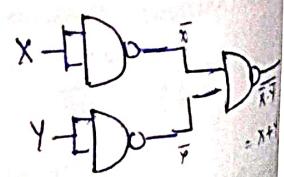
NAND gate as universal gate -



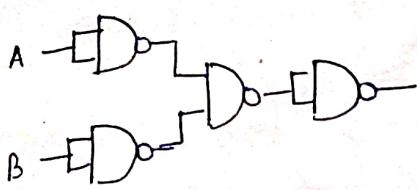
NOT



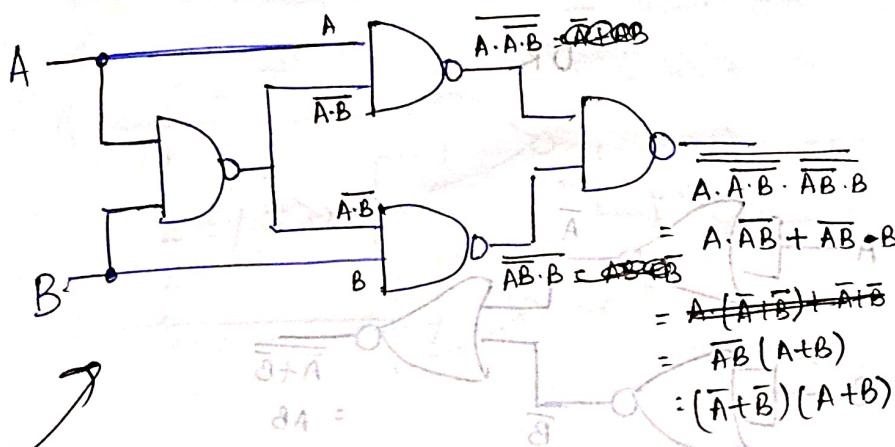
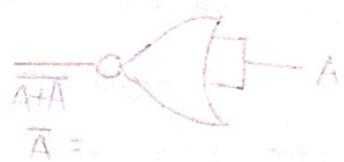
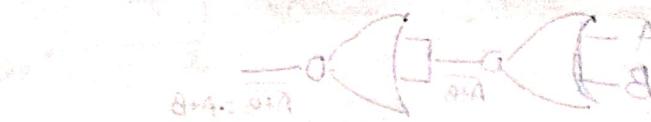
AND



OR

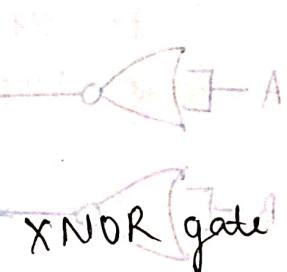
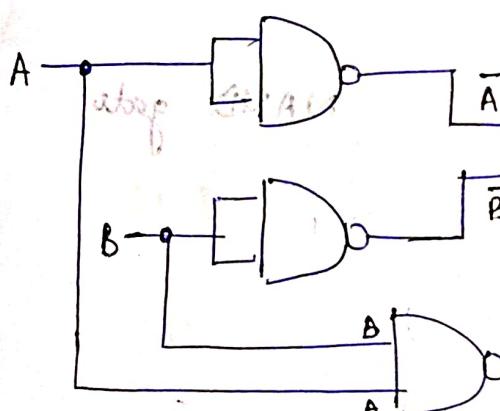


NOR gate.



XOR gate
(UNNAND gates)

QWP

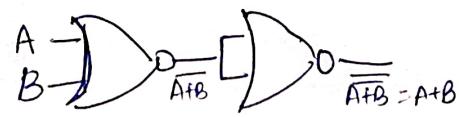
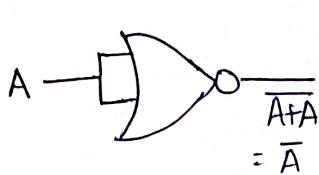


stop 80X

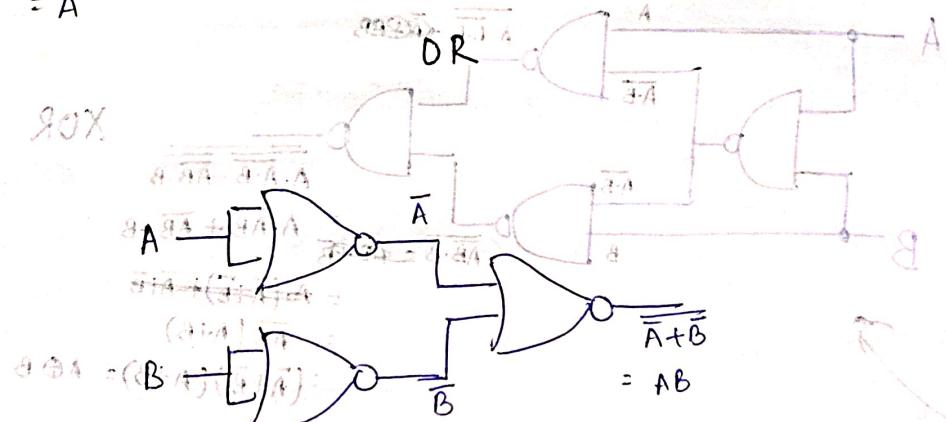
$$(A \oplus B) \cdot (\overline{A} \oplus \overline{B}) =$$

$$A \oplus B =$$

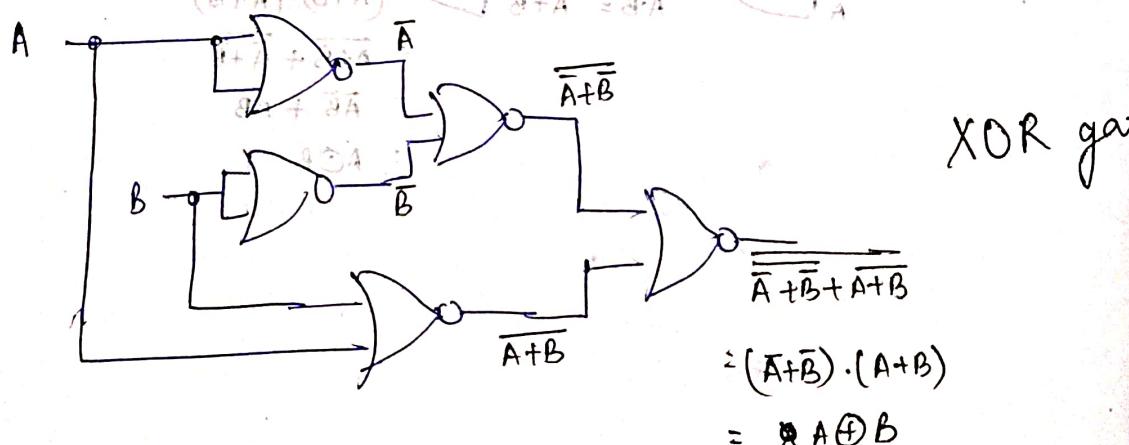
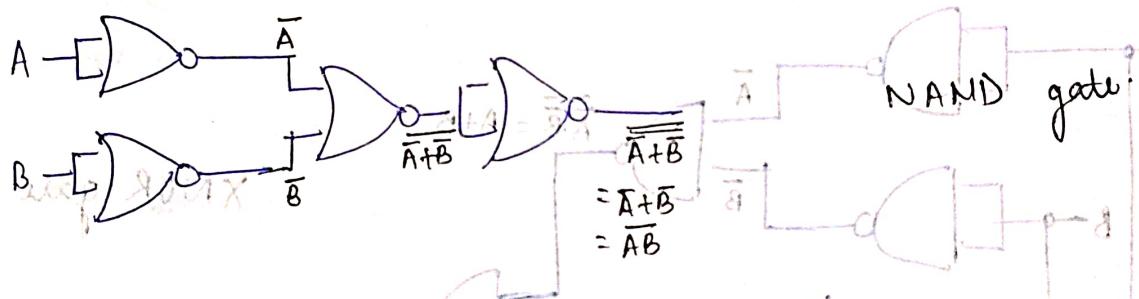
NOR gate as universal gate.



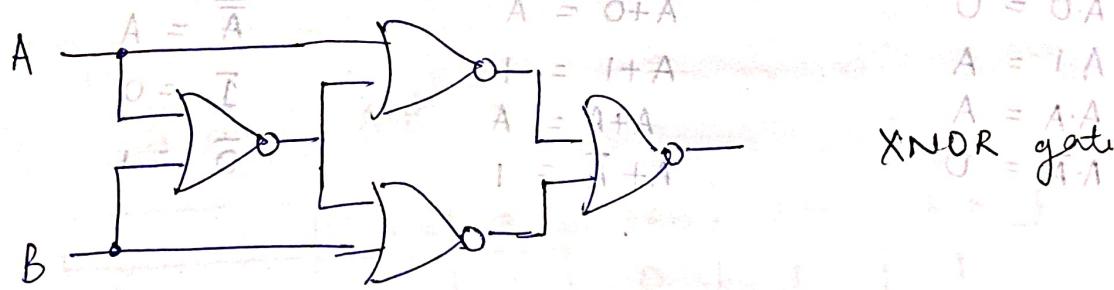
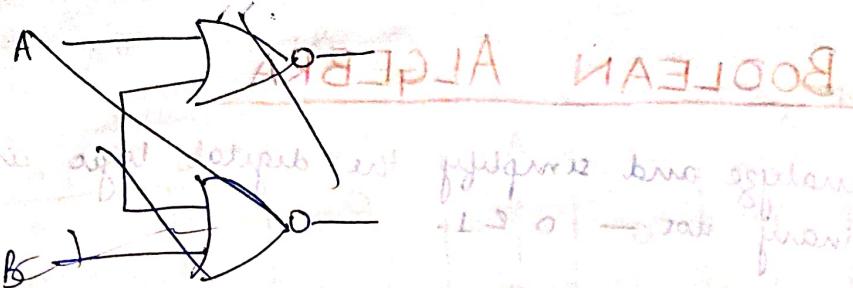
NOT
step 80X
(step of circuit)



AND

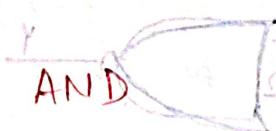


XOR gate.



$$\begin{aligned}
 & J \cdot A + S \cdot A = (J+S) \cdot A \\
 & \text{For odd no. of inputs, XOR (and) XNOR are same} \\
 & (J+A)(S+A) = JS + JA \\
 & A \oplus B \oplus C = A \odot B \odot C
 \end{aligned}$$

* No. of NAND gates vs No. of NOR gates



$$J \cdot A + S \cdot \bar{A} = J \bar{A} + \bar{S} \cdot A = Y$$

$$A = Y, 0 = \bar{Y}$$

NOT

$$\text{Bab } A = \bar{A}$$

if 0 then 1

XOR

step wait time

XNOR

NAND

NOR

$$J \cdot \bar{A} + S \cdot A = J \bar{A} + S \bar{A} = Y$$

$$A = Y, 1 = \bar{Y}$$

wait, give 0 if 1

so never 1

if 0

4

5

4

1

4

1

4

1

No. of

NOR gates

1

2

3

4

5

4

1

4

1

4

1

4

1

4

1

BOOLEAN ALGEBRA

Used to analyze and simplify the digital logic circuits.
uses 2 binary nos → 0 & 1.

AND Laws -

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

~~$$A \cdot \bar{A} = 0$$~~

OR Laws -

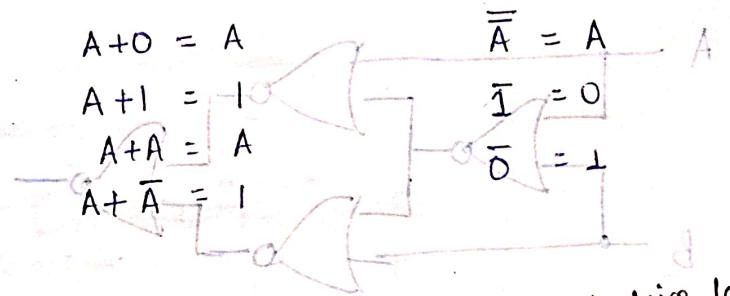
$$A + 0 = A$$

$$A + 1 = 1$$

$$A + A = A$$

$$A + \bar{A} = 1$$

Inversion law



Commutative law

$$A + B = B + A$$

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

Associative law

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

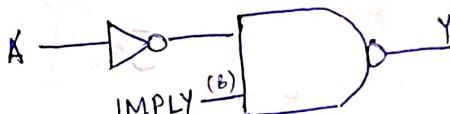
$$(A \cdot B)C = A \cdot (B \cdot C)$$

Distributive law

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

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

Implication gate and inhibition gate.

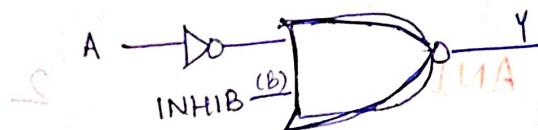


$$Y = \overline{A \cdot B} = \overline{A} + \overline{B} = A + \overline{B}$$

$$\begin{aligned} \text{if } B=0, Y=1 \\ \text{if } B=1, Y=A \end{aligned}$$

if B is true, then
A comes as
op.

Implication gate



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

$$\begin{aligned} \text{if } B=0, Y=A \\ \text{if } B=1, Y=0 \end{aligned}$$

if B=1, A does
not reach op.

Inhibition gate

A

B

A

B

A

B

De Morgan's first Theorem -

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

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

De Morgan's second Theorem -

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

A	B	$\overline{A} \cdot \overline{B}$	$\overline{B} \cdot \overline{A}$	$\overline{A+B}$	$\overline{A+B}$	$\overline{A-B}$
0	0	(0+1) · (0+1)	1 · 1	0	1	1
0	1	0 · 1	0	1	0	0
1	0	0 · 0	1	1	0	0
1	1	0 · 0	0	0	0	0

Principle of duality -

dual of an expression $A = \overline{A} + A$

- changing AND to OR (vice versa)
- changing zeros to ones and vice versa
- variables & complements remain unchanged.

Concensus Theorem / Redundancy Theorem

→ 3 variables must be present in the expression.

→ Each variable is repeated twice.

→ One variable is present in complemented and uncomplemented form.

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

Proof:

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

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

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

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

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

$$\cancel{\text{AB}} \quad \cancel{\text{AC}}$$

Redundant Literal Rule

Redundant Literal Rule says that

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

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

because two terms of AND operation are same.

before this operation, there was 3 literals in expression.

after this operation, there is only 2 literals in expression.

Shannon's Expansion Theorem

MIN TERMS — product of the variables for which each operand appears exactly once in true or complemented form.

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

MAXTERMS — sum of variables in which each variable appears exactly once in true or complemented form.

A	B	C	Maxterm
0	0	0	$A+B+C$ M_0
0	0	1	$A+B+\bar{C}$ M_1
0	1	0	$A+\bar{B}+C$ M_2
0	1	1	$A+\bar{B}+\bar{C}$ M_3
1	0	0	$\bar{A}+B+C$ M_4
1	0	1	$\bar{A}+B+\bar{C}$ M_5
1	1	0	$\bar{A}+\bar{B}+C$ M_6
1	1	1	$\bar{A}+\bar{B}+\bar{C}$ M_7

$$M_j = \overline{m_j}$$

Canonical SOP \rightarrow sum of products \rightarrow sum of minterms.

Canonical POS \rightarrow product of sums \rightarrow product of maxterms.

Standard POS \rightarrow simplified \rightarrow each term need not contain all the literals.

Standard POS \rightarrow simplified \rightarrow each term need not contain all the literals.

M	$\bar{A}\bar{B}\bar{C}$	0	0	0
M	$\bar{A}\bar{B}C$	1	0	0
M	$\bar{A}B\bar{C}$	0	1	0
M	$A\bar{B}\bar{C}$	0	0	1
M	$\bar{A}BC$	0	1	1
M	$AB\bar{C}$	1	1	0
M	ABC	1	1	1

K-map -

① Karnaugh map:

② graphical method which consists of 2^n cells for n variables.

The adjacent cells differ only in single bit position \Rightarrow gray code.

M	$\bar{A}\bar{B}\bar{C}$	0	0	0
M	$\bar{A}\bar{B}C$	1	0	0
M	$\bar{A}B\bar{C}$	0	1	0
M	$A\bar{B}\bar{C}$	0	0	1
M	$\bar{A}BC$	0	1	1
M	$AB\bar{C}$	1	1	0
M	ABC	1	1	1

Important terminologies -

1. Implicant: minterm or maxterm.

2. Prime implicant: it is formed by combining maximum possible adjacent cells.

3. Essential prime implicant:-

prime implicant which contains atleast one minterm or maxterm which

cannot be covered by any other prime implicant.

Design binary to gray code converter using logic gates

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	1	1	0	0

$$W = \sum m(8, 9) + d(10, 11, 12, 13, 14, 15)$$

$$X = \sum m(4, 5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$$

$$Y = \sum m(2, 3, 4, 5) + d(10, 11, 12, 13, 14, 15)$$

$$Z = \sum m(1, 2, 5, 6, 9) + d(10, 11, 12, 13, 14, 15)$$

For W -

AB	CD	$\bar{C}D$	$\bar{C}D$	CD	$\bar{C}D$	CD	$\bar{C}D$
AB	0	1		3		2	
$\bar{A}\bar{B}$	4	5		7		6	
AB	x	x	x	x	x	x	x
$\bar{A}\bar{B}$	1	1	x	x	x	x	x
	8	9	11	10			

For X -

AB	CD	$\bar{C}D$	$\bar{C}D$	CD	$\bar{C}D$	CD	$\bar{C}D$
AB	0	1		3		2	
$\bar{A}\bar{B}$	1	1	x	x	x	x	x
AB	x	x	x	x	x	x	x
$\bar{A}\bar{B}$	12	13	15	14	11	10	
	8	9	11	10			

$$\therefore W = A \oplus B \quad X = \bar{A}B + A\bar{B}$$

For Y -

AB	CD	$\bar{C}D$	$\bar{C}D$	CD	$\bar{C}D$	CD	$\bar{C}D$
AB	0	1		1	1	2	
$\bar{A}\bar{B}$	1	1	x	x	x	x	x
AB	x	x	x	x	x	x	x
$\bar{A}\bar{B}$	12	13	15	14	11	10	
	8	9	11	10			

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

For Z -

AB	CD	$\bar{C}D$	$\bar{C}D$	CD	$\bar{C}D$	CD	$\bar{C}D$
AB	0	1		3		2	
$\bar{A}\bar{B}$	1	1	x	x	x	x	x
AB	x	x	x	x	x	x	x
$\bar{A}\bar{B}$	12	13	15	14	11	10	
	8	9	11	10			

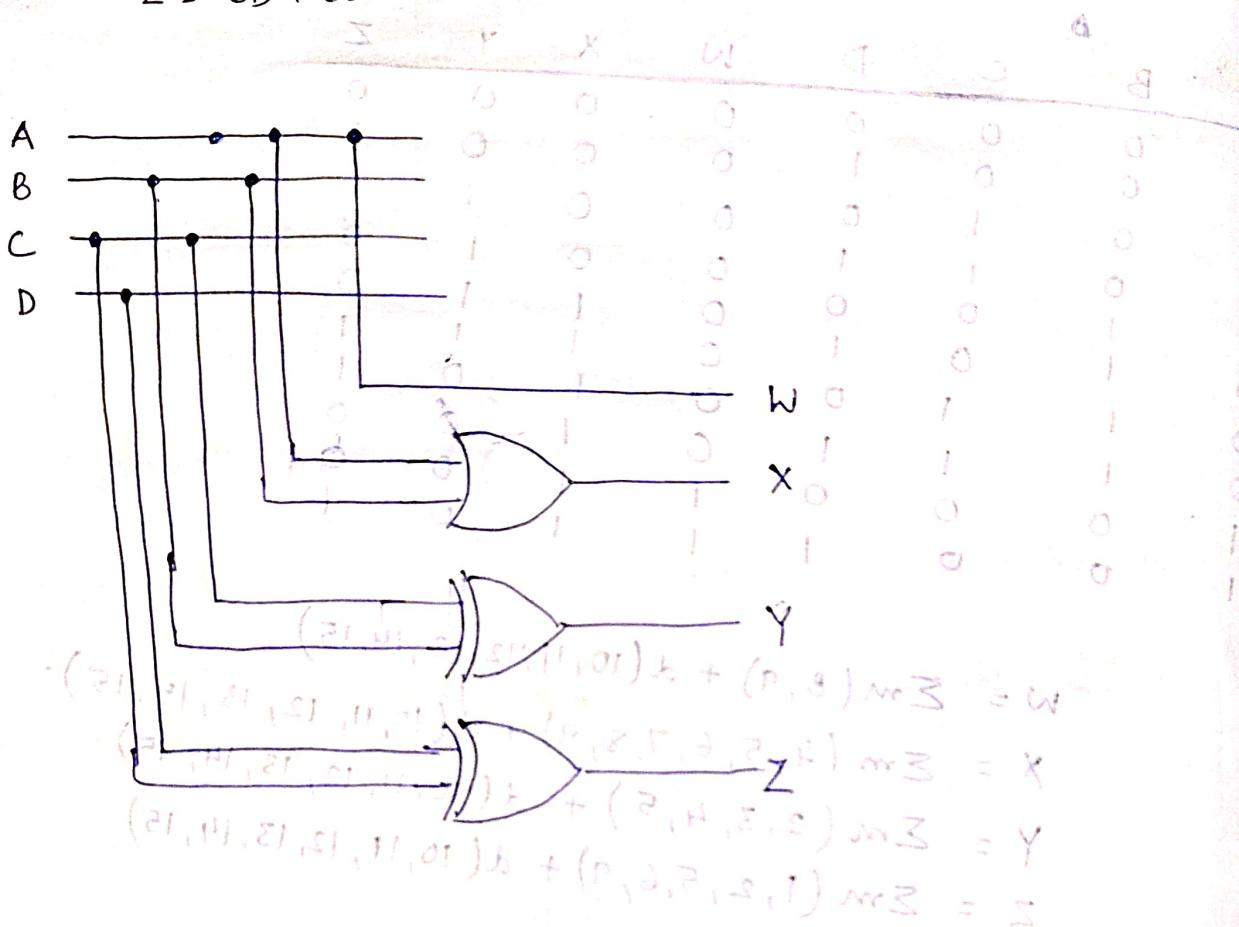
$$Z = \bar{C}D + C\bar{D}$$

Input : $w = A$ (from row 0) other rows of previous slide

$$x = A + B$$

$$y = BC + \bar{B}C$$

$$z = \bar{C}D + C\bar{D}$$



Implicant — Σx_{101}

→ product/minterm term in SOP or sum/maxterm term in POS of a Boolean function

$$\text{Ex: } F = AB + ABC + BC$$

Implicants are AB , ABC & BC

every x term in logic expression is called implicant.



$$\bar{x}_0 + \bar{x}_3 = y$$



PRIME IMPLICANTS -

group of square or rectangle made up of bunch of adjacent minterms / maxterms.

- Form as large group as possible
- Form all possible combinations
- do not form any smaller group within larger group.

Ex

1	1	2	3
	1	1	

3 prime implicants.

ESSENTIAL PRIME IMPLICANTS

Those prime implicants which cover atleast one minterm that can't be covered by any other prime implicant.

1	1		
	1	1	

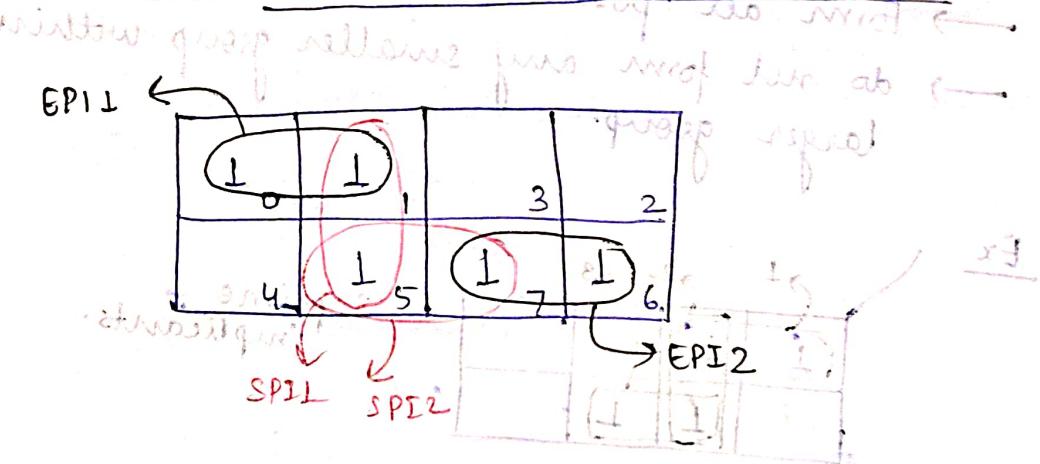
No. of essential prime implicants = 2

Redundant prime implicant do not appear in minimized logic expression

1	1		
	1	1	

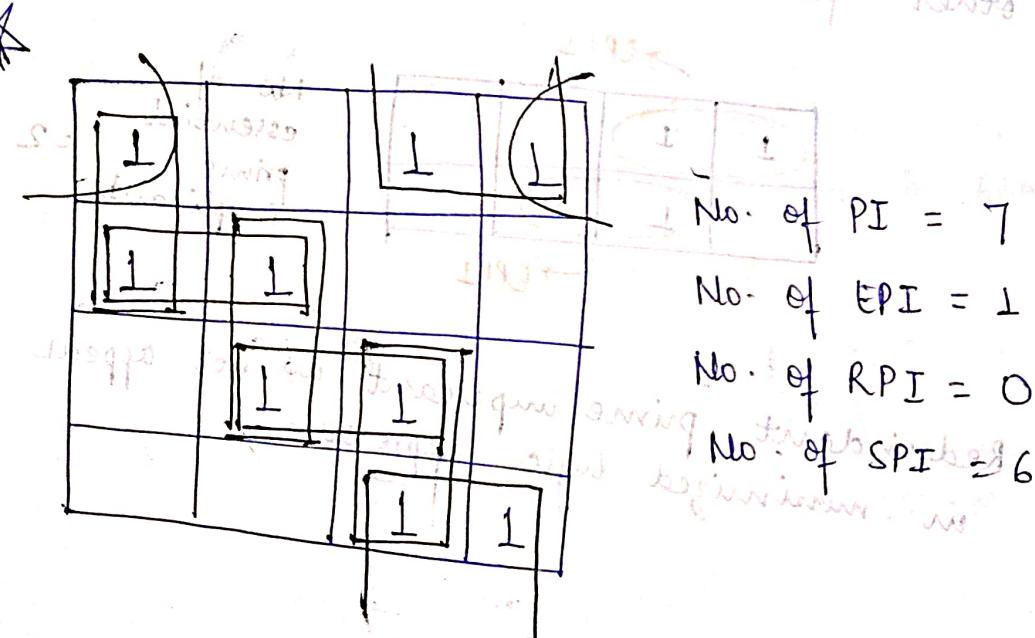
SELECTIVE PRIME IMPLICANTS

prime implicants which are neither essential nor redundant prime implicants are called Selective Prime Implicant (SPI).



* For maxterm, we replace prime implicant by false prime implicants.

* Smaller group within larger groups is implicant.



Ques

Given $F = \sum m(1, 5, 6, 7, 11, 12, 13, 15)$,
find number of PI, EPI, RPI and SPI.

0	1	3	2	
4	5	7	6	
12	13	15	14	
8	9	11	10	

$$\text{No. of implicants} = 8 + 5 + 4 = 17$$

$$\text{No. of PI} = 5$$

$$\text{No. of EPI} = 4$$

$$\text{No. of RPI} = 1$$

$$\text{No. of SPI} = 0$$

Ques

$$\sum m(0, 1, 5, 8, 12, 13)$$

Ans: If EPI is not there, RPI cannot exist.

1	1		3	2
4	1		7	6
12	13		15	14
8	9		11	10

$$\text{No. of implicants} = 6 + 5 = 12$$

$$\text{No. of PI} = 6$$

$$\text{No. of EPI} = 0$$

$$\text{No. of RPI} = 0$$

$$\text{No. of SPI} = 6$$

Ques

$$\sum (0, 1, 5, 7, 15, 14, 10)$$

1	1		3	2
4	1	1	7	6
12	13	1	15	14
8	9		11	10

$$\text{No. of implicants} = 6 + 7 = 13$$

$$\text{No. of PI} = 6$$

$$\text{No. of EPI} = 2$$

$$\text{No. of RPI} = 0$$

$$\text{No. of SPI} = 4$$

\star Number of self dual functions

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

$f(x) = P + Q + R + S = \text{maximum terms}$
 $S = 19$ for all
 \rightarrow Number of minterms = no. of max terms
 \rightarrow function does not contain mutually exclusive terms.

$$P = I92 \text{ for all}$$

$$Q = I93 \text{ for all}$$

$$R = I94 \text{ for all}$$

$$S = I95 \text{ for all}$$

Minterm Map

(01, 11, 21, 15, 2, 10, 0)

Maxterm Map

(01, 14, 21, 15, 2, 10, 0)

Functionally complete boolean function

for 3 variables

practise

$$S = P + Q = \text{maximum terms}$$

$$P = I9 \text{ for all}$$

$$Q = I93 \text{ for all}$$

$$\neg P = I91 \text{ for all}$$

$$\neg Q = I92 \text{ for all}$$

Minterm Map

(01, 11, 21, 15, 2, 10, 0)

$$(01, 14, 21, 15, 2, 10, 0)$$

$$S = P + Q = \text{maximum terms}$$

$$P = I9 \text{ for all}$$

$$Q = I93 \text{ for all}$$

$$\neg P = I91 \text{ for all}$$

$$\neg Q = I92 \text{ for all}$$

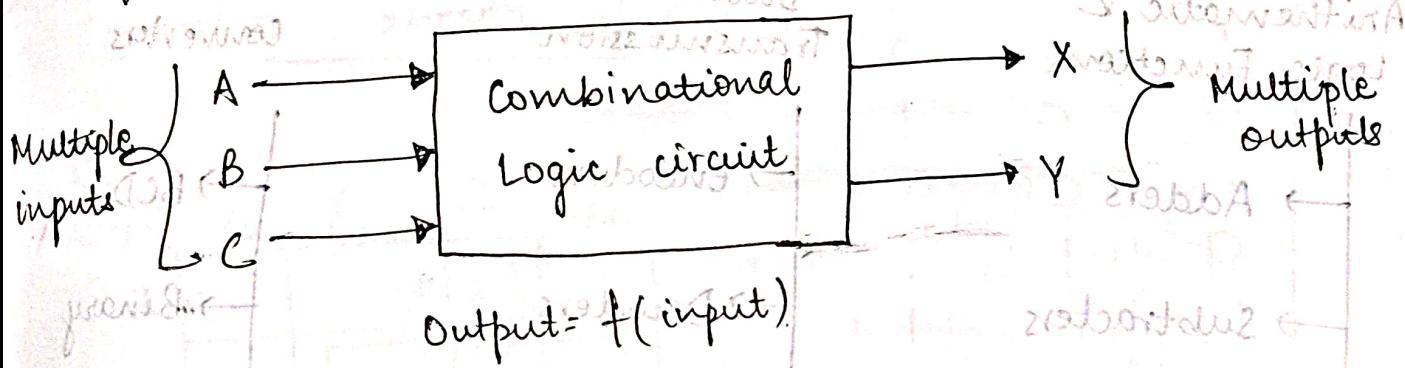
Minterm Map

(01, 14, 21, 15, 2, 10, 0)

COMBINATIONAL CIRCUITS

Combinational circuits consist of logic gates. These circuits operate with binary variables values.

The o/p of combinational circuit depends upon the combination of present inputs.



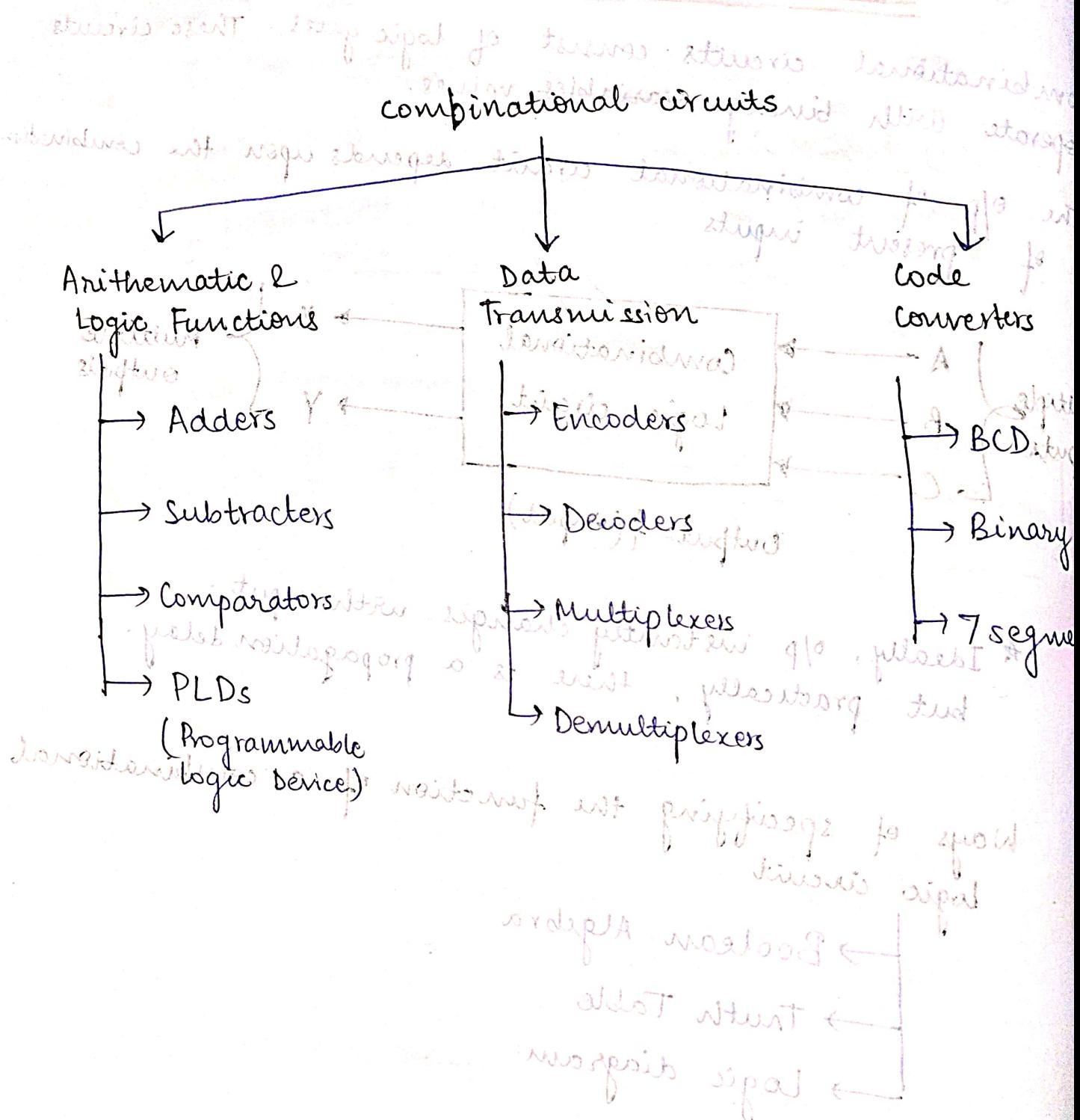
* Ideally, o/p instantly changes with input signals.
but practically, there is a propagation delay.

Ways of specifying the function of a combinational logic circuit

- Boolean Algebra
- Truth Table
- Logic diagram.

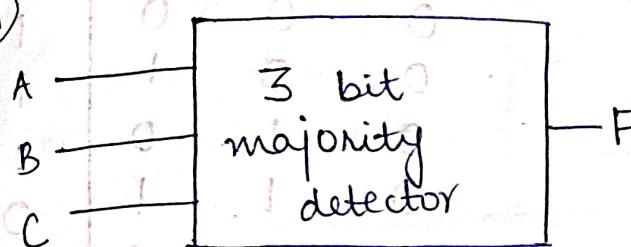
COMBINATIONAL CIRCUITS

Classification of combinational circuits



Majority Detectors -

This circuit gives the output 1 when inputs have more number of 1's than 0's.



③

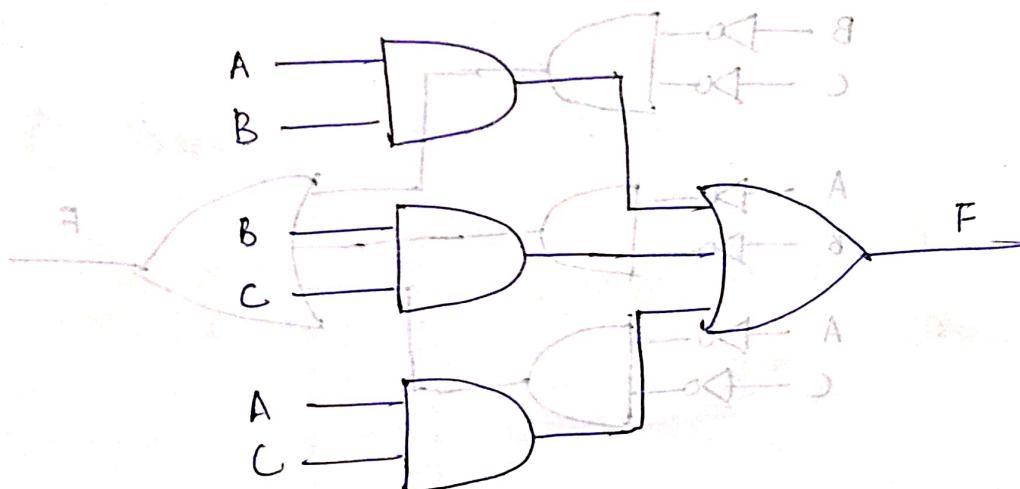
A	$\bar{B}C$	$B\bar{C}$	$\bar{B}C$	$B\bar{C}$
\bar{A}	1	1	1	2
A	4	5	7	6

④

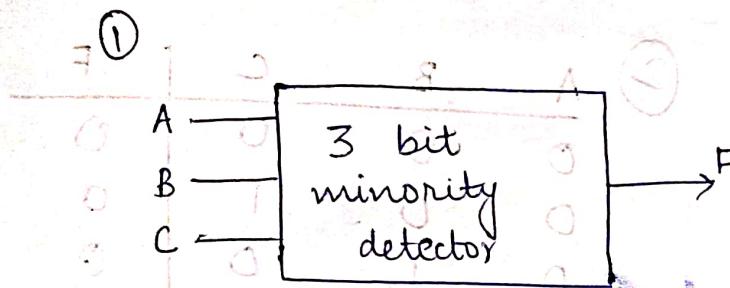
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

$$\therefore F = AC + AB + BC$$

④ Circuit diagram -

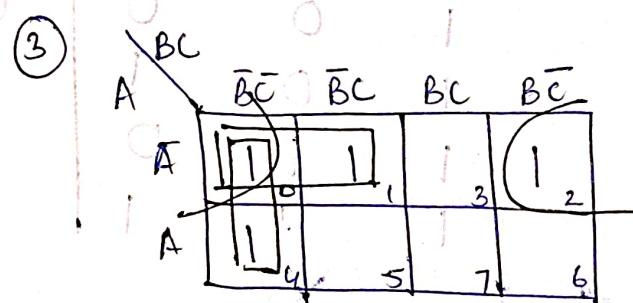


Minority detector - gives o/p 1 whenever inputs have lesser no. of 1s than 0s.



②

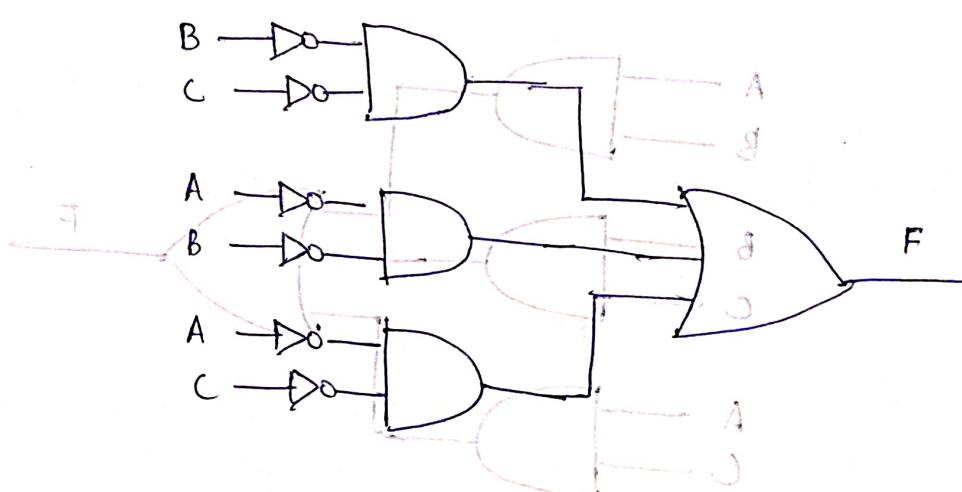
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0



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

$$\Sigma d + \Sigma A + \Sigma A = 7$$

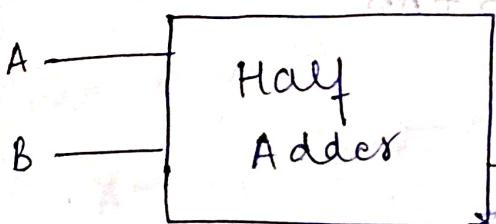
④ Logic circuit -



Half Adder -

device which adds 2 bits and produces 2 outputs sum and carry.

①

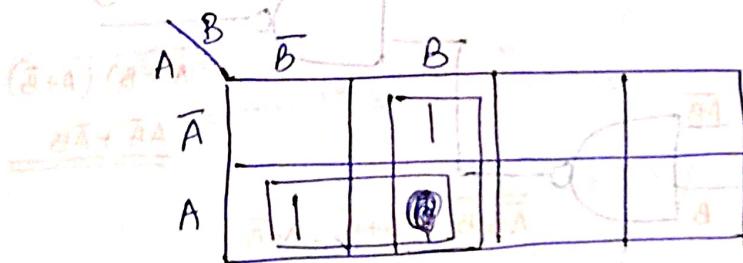


②

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

③

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

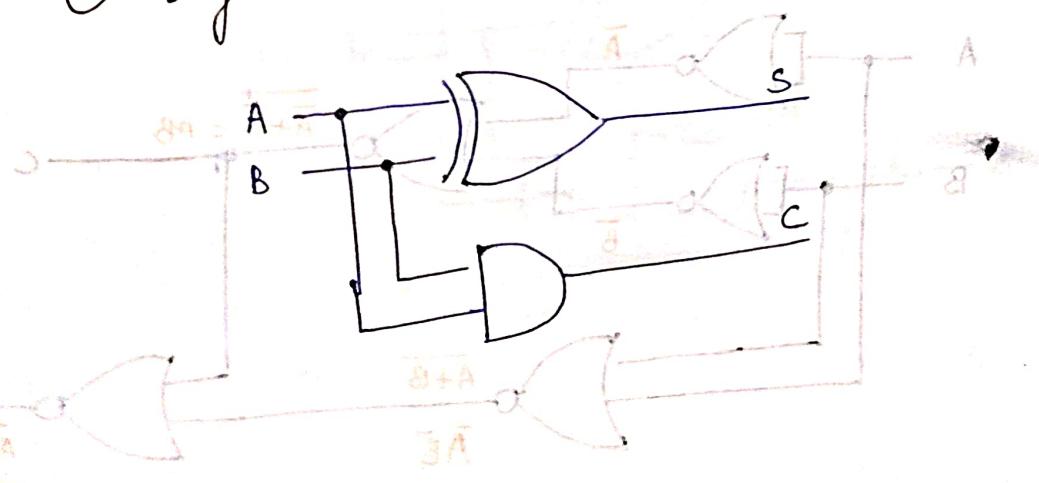


simplified -

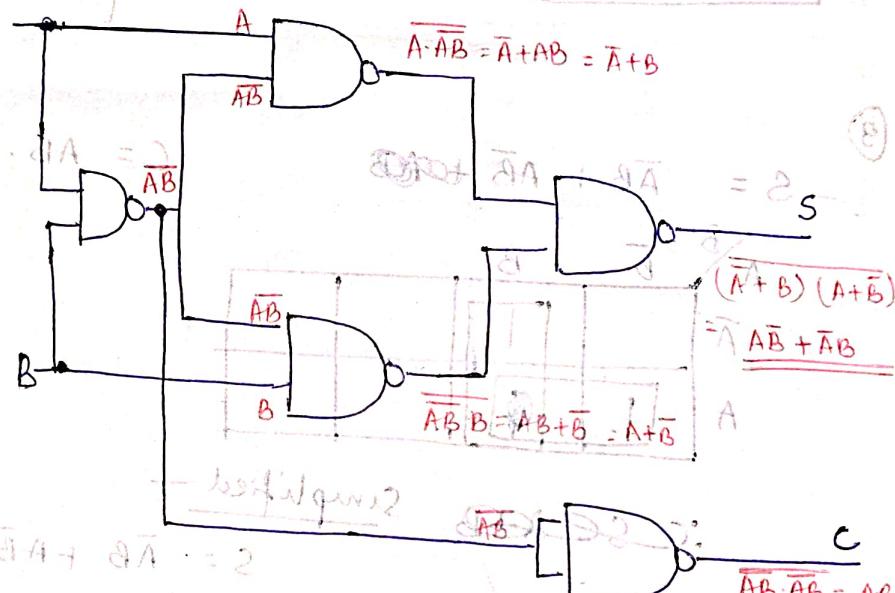
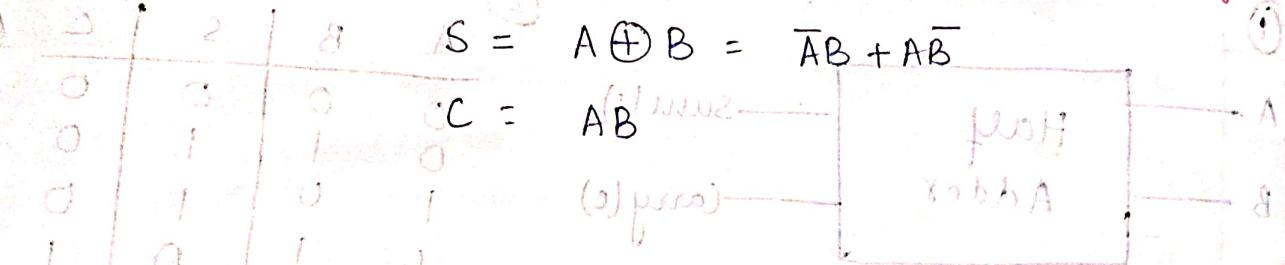
$$S = \overline{A}B + A\overline{B} = A \oplus B$$

$$C = AB$$

④ Logic circuit -

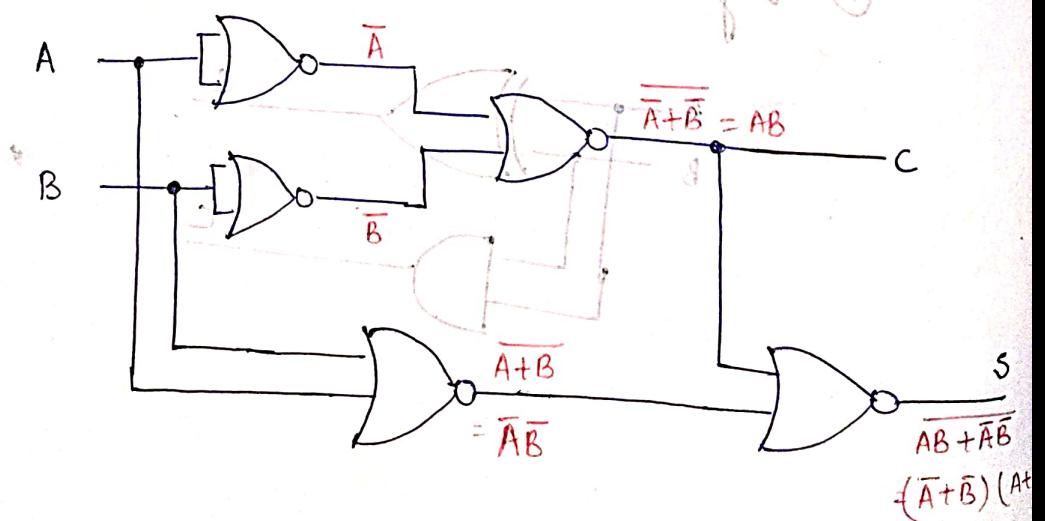


Half adder using only NAND and NOR gates



$$A \oplus B = \overline{A}A + A\overline{B} = 2$$

$\therefore 5$ NAND gates are required.

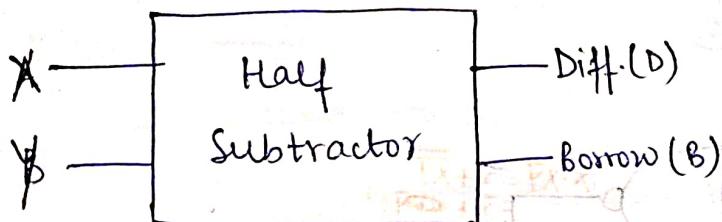


$\therefore 5$ NOR gates are required.

Half subtractor

circuit to perform subtraction of 2 binary bits.
 O/p → Difference and borrow.

①



②

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

③

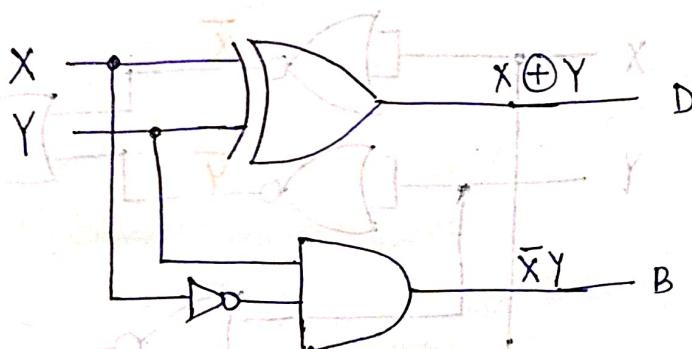
Simplified expression -

$$D = A \bar{X} \bar{Y} + \bar{X} Y = X \oplus Y$$

$$B = \bar{X} Y$$

④

Logic circuit diagram

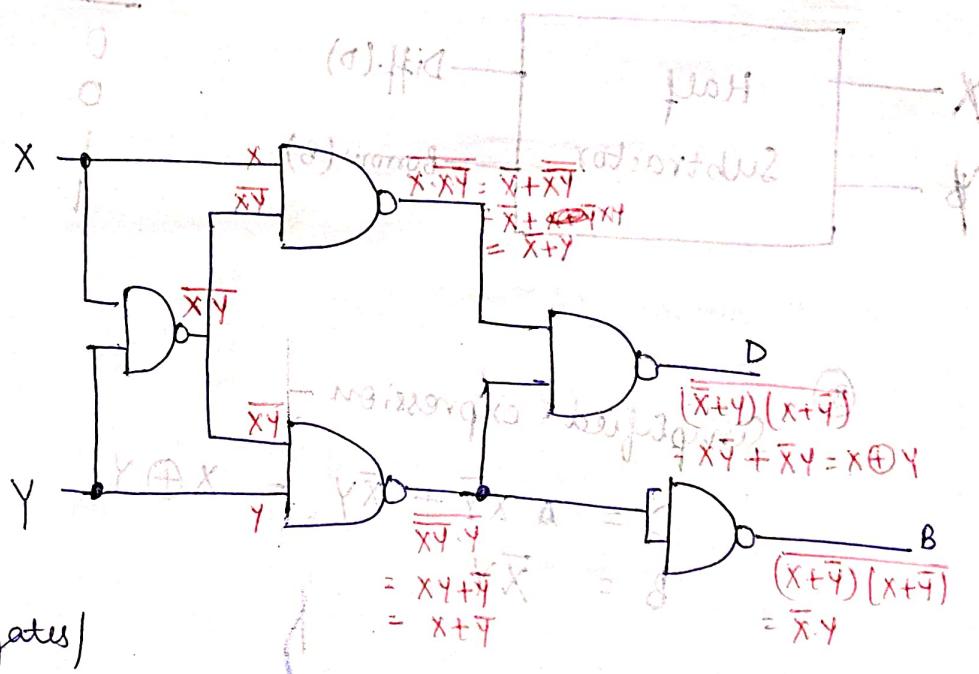


Half Subtractor using NAND and NOR gates.

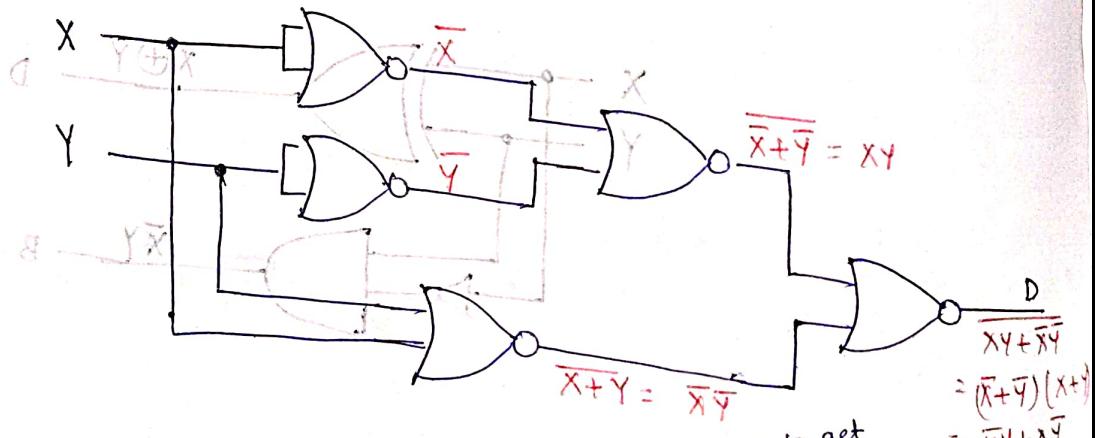
$$D = X \oplus Y$$

$$B = \overline{X}Y$$

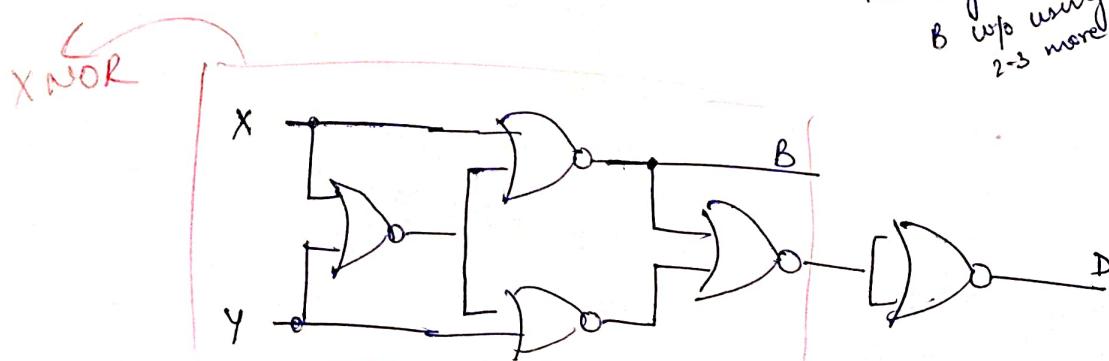
	X	Y	D	B
0	0	0	0	0
1	1	0	1	0
0	1	0	1	0
1	0	1	0	1



5 NAND gates
5 NOR gates
required.

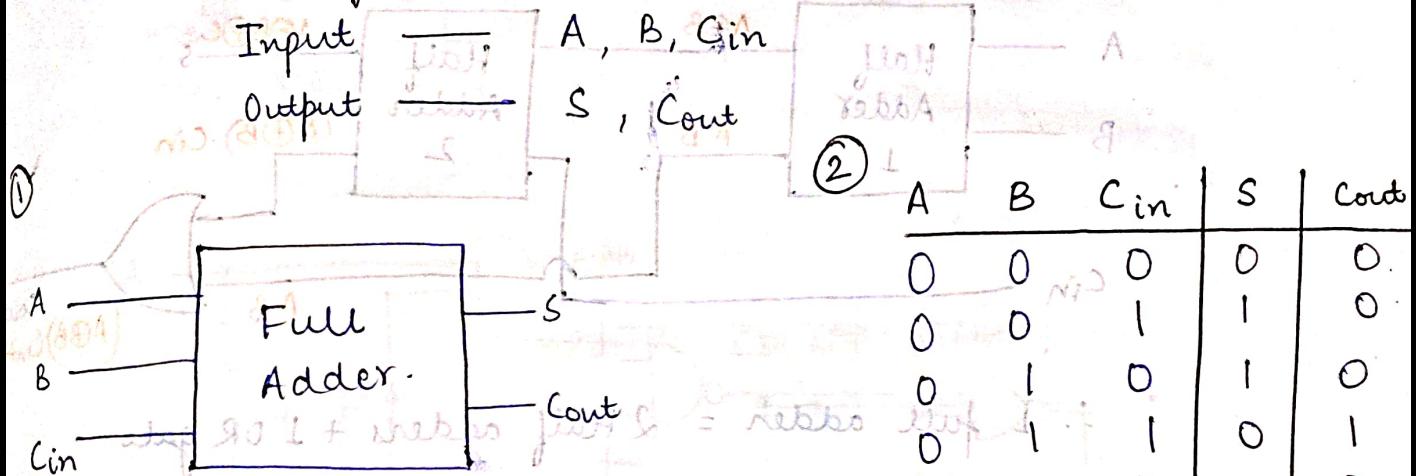


No way to get B w/o using 2-3 more gates.



Full Adder

takes 3 inputs and produces 2 outputs — sum and carry.



③

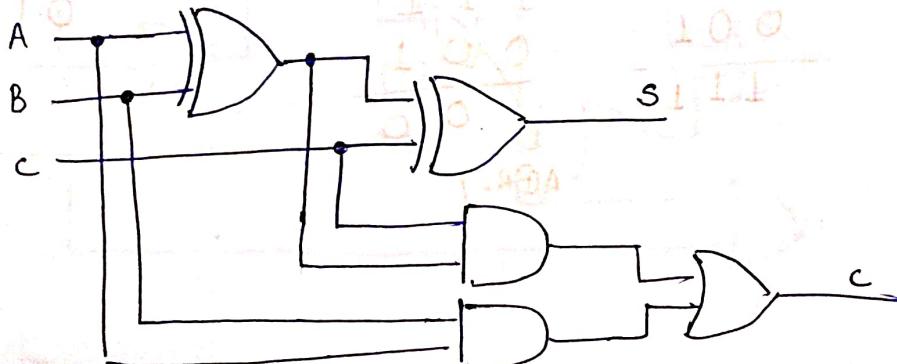
Sum = 1 when 1's at input are odd

$$S = A \oplus B \oplus C$$

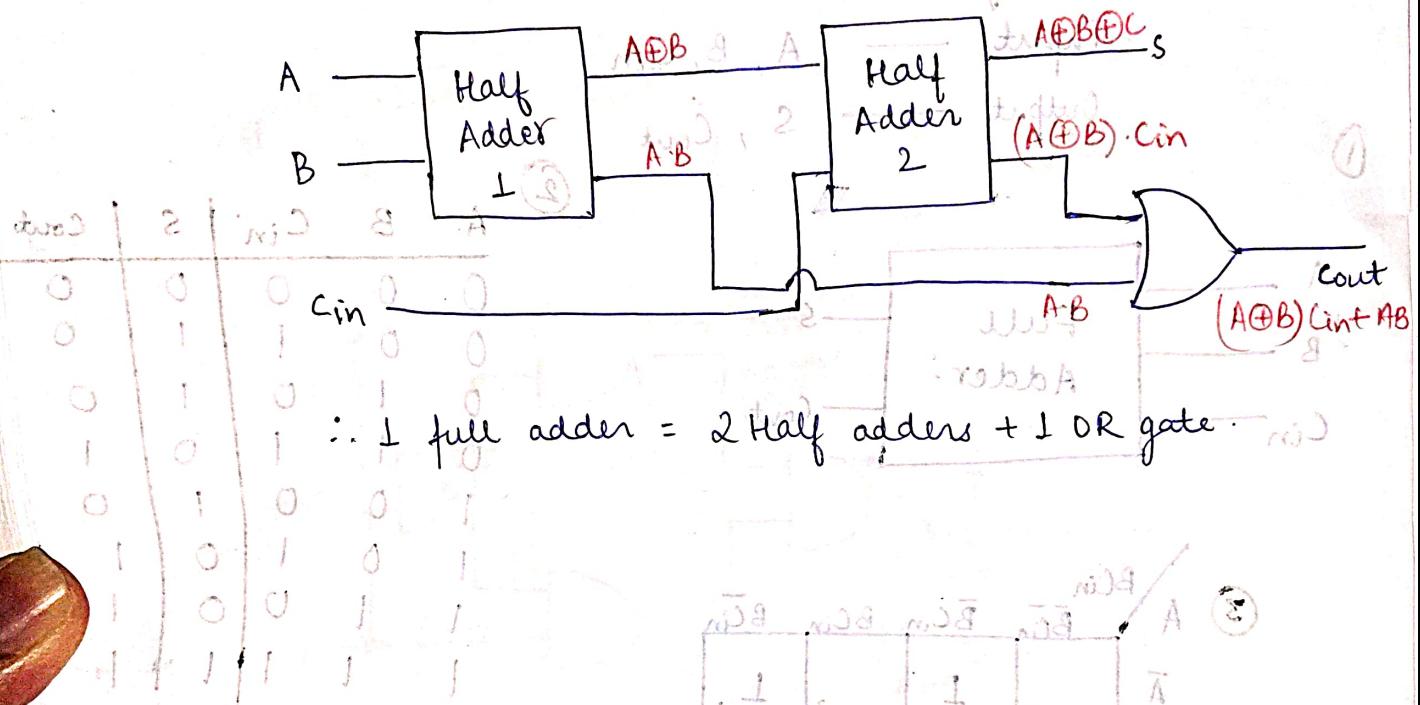
④

$$Cout = B \bar{C}in + A \bar{C}in + AB = (A \oplus B) \bar{C}in + AB$$

Logic circuit —



Implementation of Full Adder using Half Adders.



$\therefore 1 \text{ full adder} = 2 \text{ Half adders} + 1 \text{ OR gate}$

$$Cout = (A \oplus B) \cdot Cin + AB$$

carry propagated & $\oplus A = 2$ generated

If $A \oplus B = L$, i.e.

when $A=1, B=0$

$A=0, B=1$

$Cout = Cin$ i.e. carry propagates

$$\begin{array}{r} 1 \\ 1 \\ + 0 \\ \hline 11 \end{array}$$

$$\begin{array}{r} 1 \\ 0 \\ + 1 \\ \hline 11 \end{array}$$

$$\begin{array}{r} 0 \\ 1 \\ + 0 \\ \hline 01 \end{array}$$

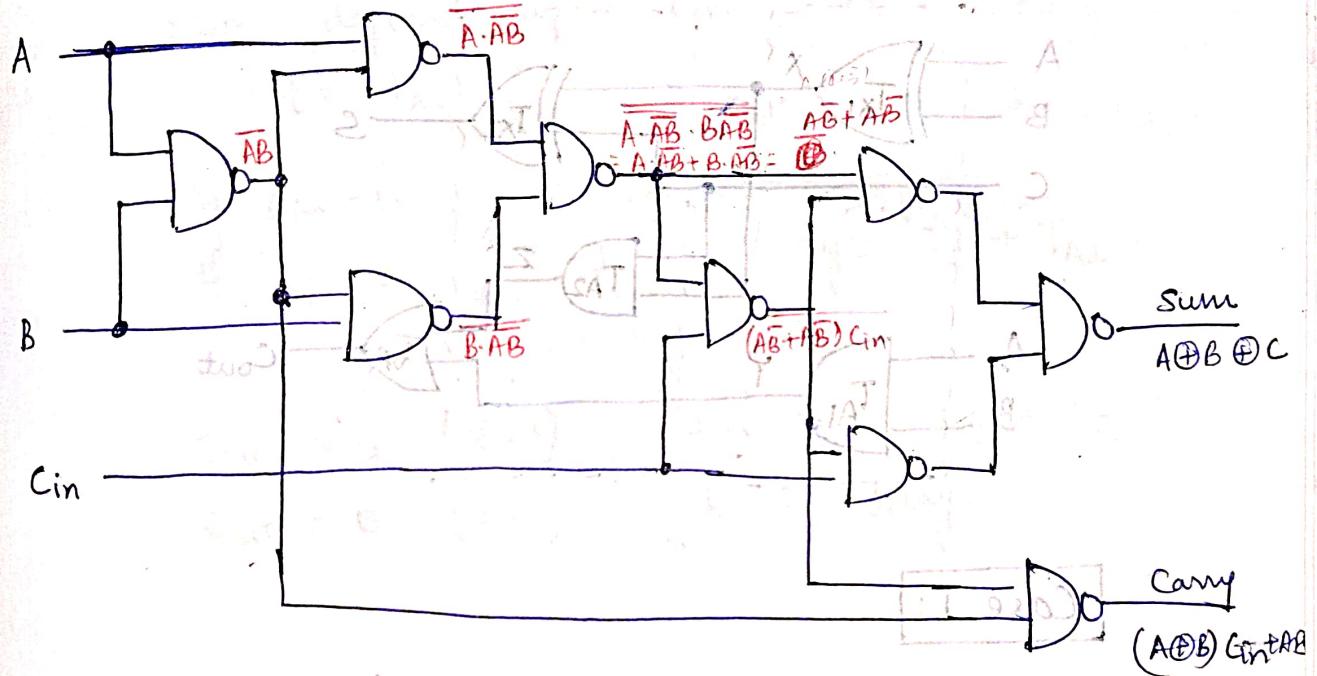
$$A \oplus B = 1$$

carry is generated.

Full adder using NAND gates

9 NAND gates are required.

to 4 for XOR1, 4 for XOR2 + 1 for carry.

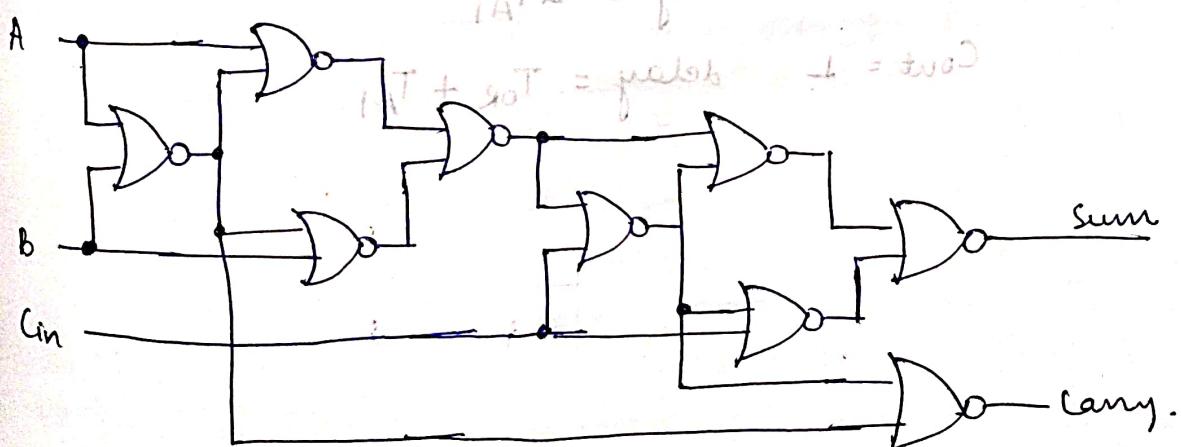


(better way) $I = S \quad I = A$

(just like : opt. way) $O = X$

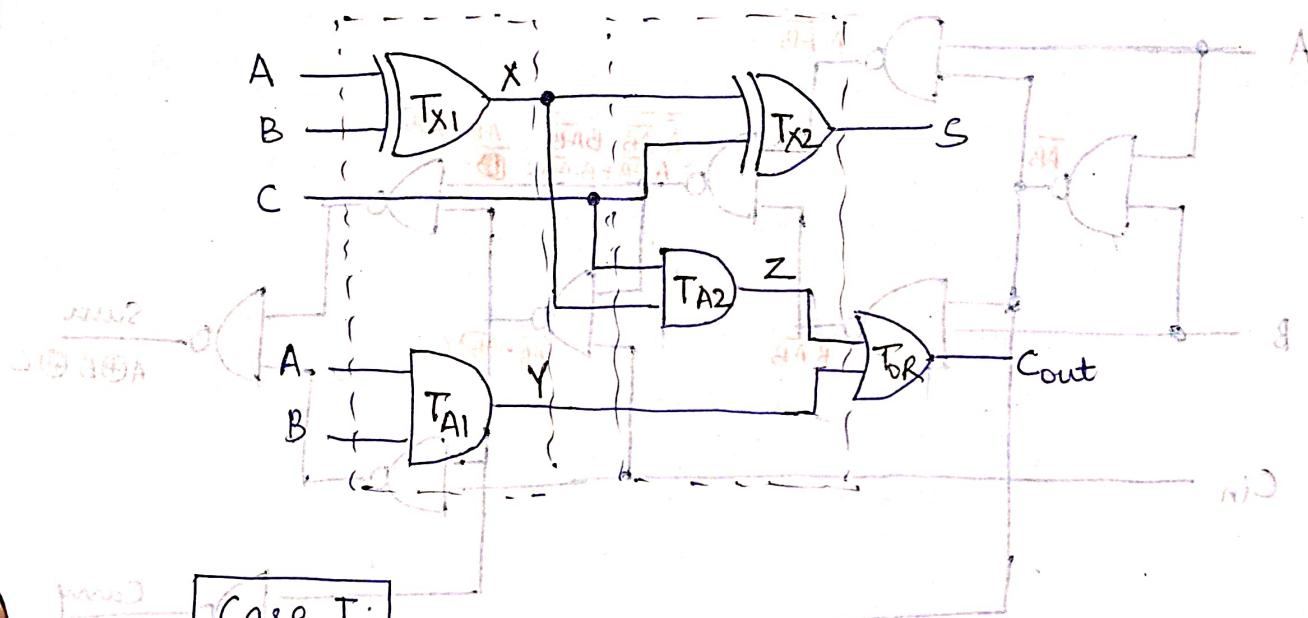
Full adder using NOR gates

9 NOR gates are required.



Propagation delay

It is the difference b/w the instant at which input changes and the time at which output changes.



$$A=1 \quad B=1 \quad (\text{carry generated})$$

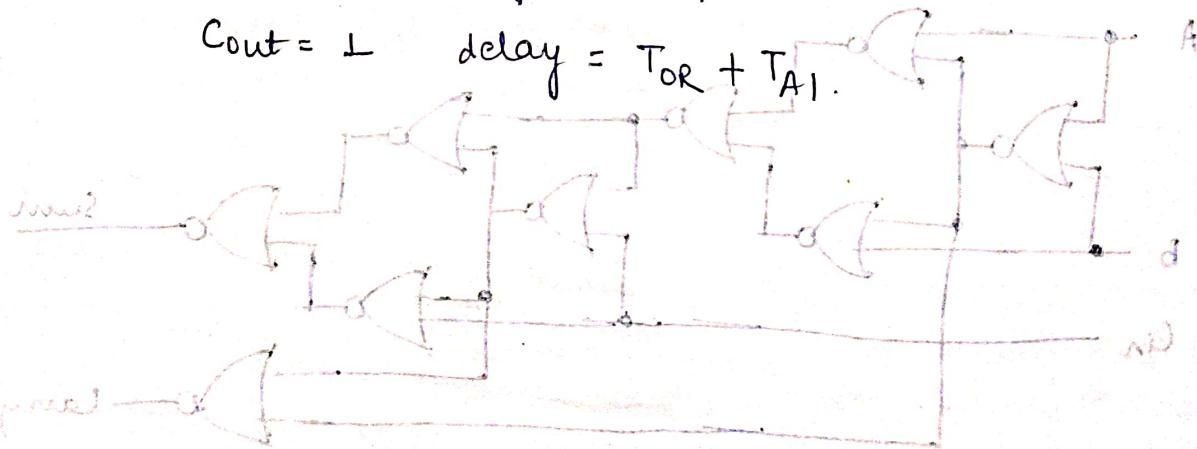
$$X=0 \quad (\text{no edge} \therefore \text{no delay})$$

if $C_{in} = 0$, $T_{sum} = T_{A1}$ no delay

if $C_{in}=1$, $T_{sum} = T_{A2}$ delay = T_{A2}

$$Y=1 \quad \text{delay} = T_{A1}$$

$$C_{out}=1 \quad \text{delay} = T_{OR} + T_{A1}$$



Case II

$A=1, B=0 \rightarrow$ [carry propagated] (borrowed)

$A=0, B=1 \rightarrow$ [no borrow, no carry]

$$X=1 \quad \text{delay} = T_{x1}$$

$$Y=0 \quad \text{no delay}$$

SUM: depends upon Cin

$$t = T_{x1} + T_{x2}$$

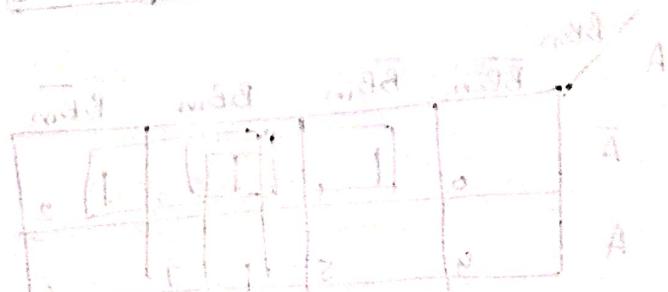
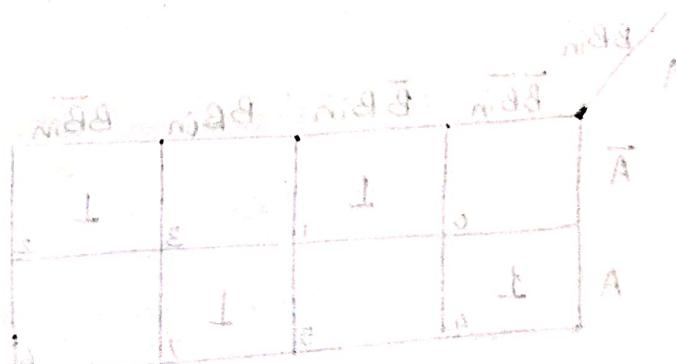
if $Cin = 0, Z = 0$

if $Cin = 1, Z = 1$

$$\text{delay} = T_{x1} + T_{A2}$$

Forward	Hold	Reverse	B	A
$1 \oplus 0 = 1$	$t = T_{x1} + T_{A2} + T_{OR}$	0	0	0
$0 \oplus 1 = 1$	$t = \text{no delay}$	0	0	1
$0 \oplus 0 = 0$		1	0	1
$0 \oplus 0 = 0$		0	1	1
$1 \oplus 1 = 0$		1	1	1

$$S \oplus C \oplus A = Q$$



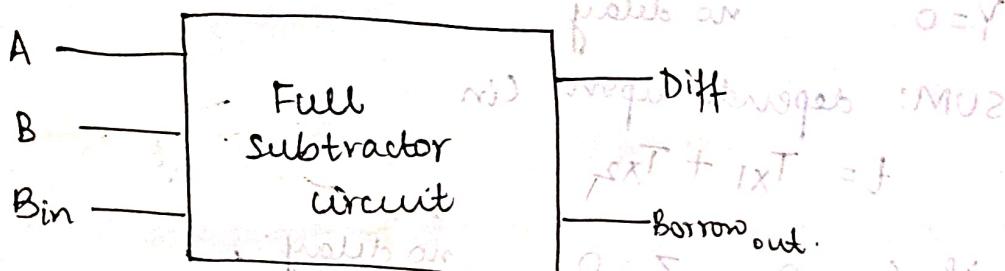
$$S + C + A = Q$$

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

Full Subtractor

IT 2023

combinational circuit that performs subtraction of 3 bits — minuend, subtrahend & borrow of previous adjacent lower minuend bits.



A	B	Borrow _{in}	Diff	Borrow _{out}
0	0	$SAT + SAT + FixT_0 = f$	$D = 0$	0
0	0	1	1	0
0	1	$f_{sub} \text{ out} = 1$	$D = 1$	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

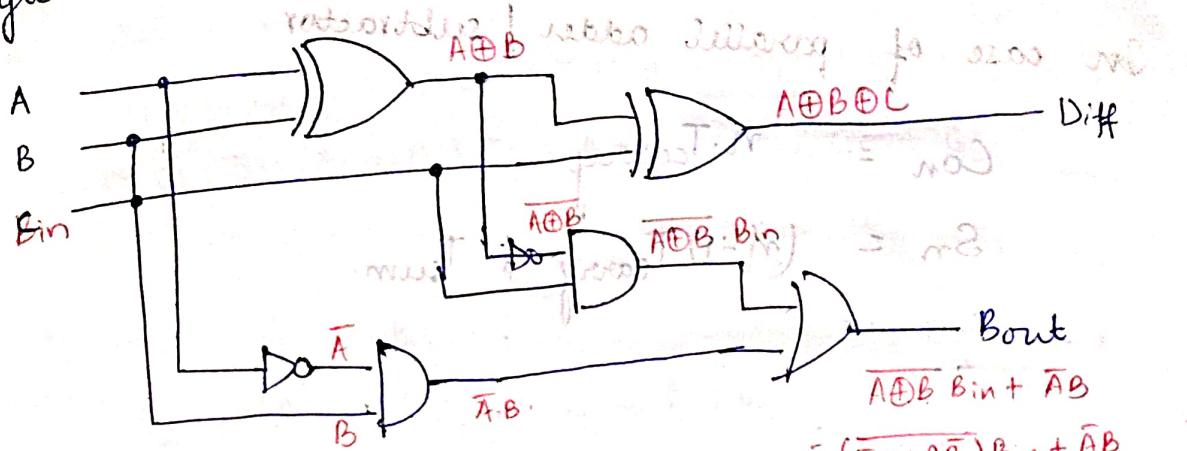
A	\bar{B}_{in}	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	$B\bar{B}_{in}$	$B\bar{B}B_{in}$
\bar{A}	0	1		3	1
A	1		0	2	4

$$\therefore D = A \oplus B \oplus C$$

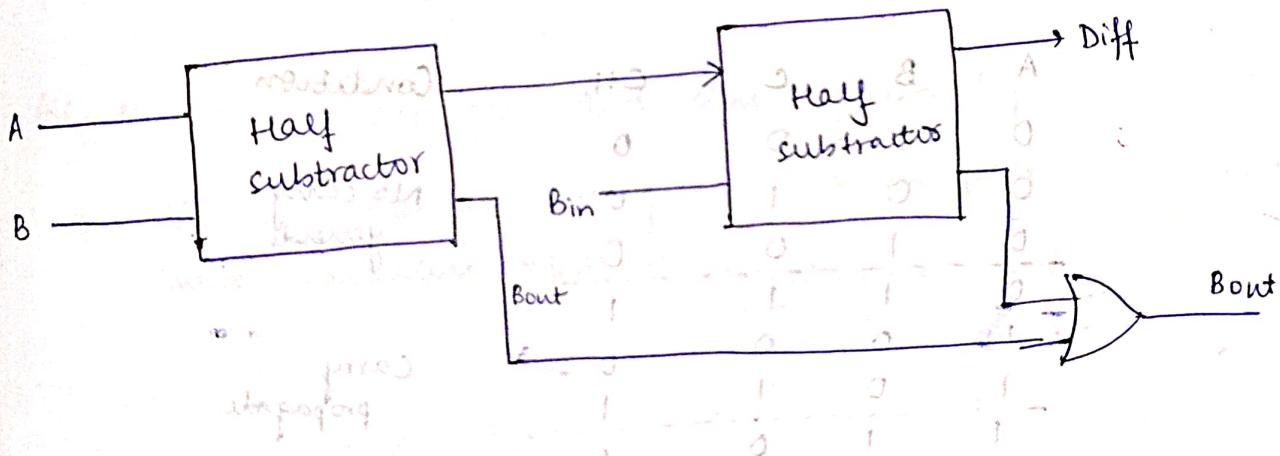
A	\bar{B}_{in}	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	$B\bar{B}_{in}$	$B\bar{B}B_{in}$
\bar{A}	0	1	1	3	1
A	1		1	2	4

$$\begin{aligned} \therefore B_{out} &= \bar{A}B_{in} + \bar{A}B + BB_{in} \\ &= \bar{A}B + (\bar{A} + B)B_{in} \end{aligned}$$

Logic circuit -



Full Subtractor using Half subtractors -



Full Subtractor using universal gates

- 9 NAND gates are required.
- 9 NOR gates are required.

In case of parallel adder / subtractor.

$$Con = n \cdot T_{carry}$$

$$S_n = (n-1) T_{carry} + T_{sum}$$

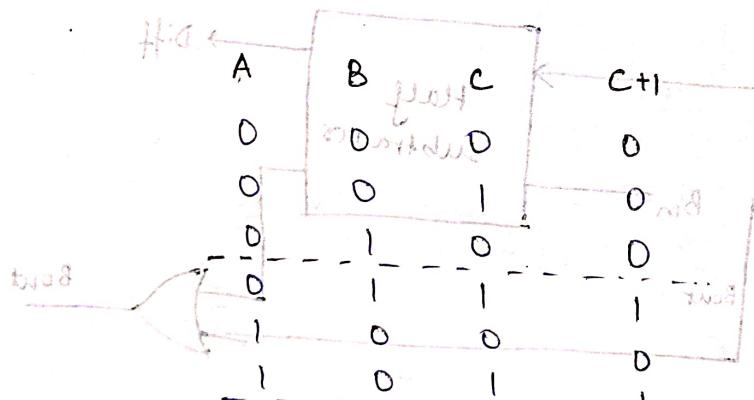
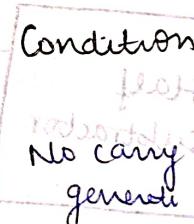
Carry Look Ahead Adder

- reduces propagation delay by introducing more complex circuit/hardware.

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

\uparrow
create carry propagate

\uparrow
carry propagate



Carry propagate

Carry generate

$$C_{i+1} = (A_i \oplus B_i) C_i + A_i B_i P$$

carry output of
ith stage

$$\text{Let } P_i = A_i \oplus B_i \quad G_i = A_i B_i$$

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

$$C_{i+1} = (A_i \oplus B_i) C_i + A_i B_i = P_i C_i + G_i$$

	$A = A$	$A \cdot C_1 =$	$G_0 + P_0 C_{in}$	
0	0	$C_2 =$	$G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_{in})$	A
0	1	$=$	$G_1 + P_1 G_0 + P_0 P_1 C_{in}$	$A > A$
1	0	$C_3 =$	$G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_{in})$	$A > A$
1	1	$=$	$G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_{in}$	$A > A$
		$C_4 =$	$G_3 + P_3 C_3 = G_3 + P_3 (G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_{in})$	$A > A$
		$=$	$G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$	$A > A$

All the carries are generated simultaneously.

can be implemented using AND & OR gates.

more hardware required.

$\oplus(A, A) = \text{don't care}$

$\oplus(A, \bar{A}) = \text{all forced 0's}$

$$jA + jA = jP \quad jB + jA = jS \quad jC + jA = jD$$

Comparators

① compare 2 numbers

② finds out whether one binary number is equal, less than or greater than the other binary number

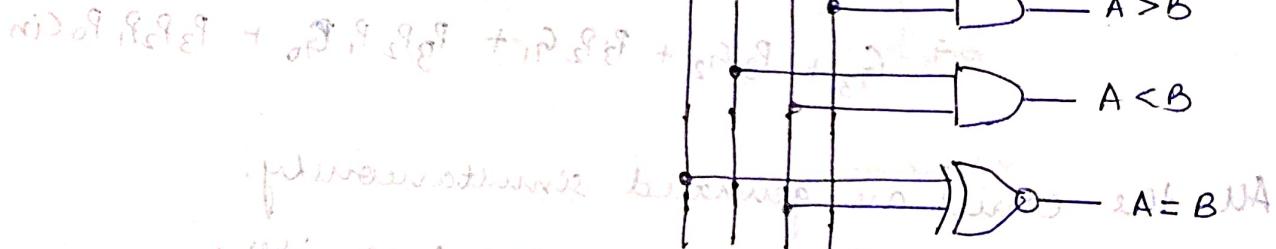
		<u>1 bit</u>	<u>A > B</u>	<u>A = B</u>	<u>A < B</u>
A					
B					
	N bit comparator				

O/P -

$$A > B : A\bar{B}$$

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

$$A < B : \bar{A}B$$



2 bit magnitude comparator

1st binary no. = A_1, A_0

2nd binary no. = A_1, B_0

A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	0	1
1	1	1	1	0	1	0

O/P

Partial Circuits

$$f(A > B) = \sum m(4, 8, 9, 12, 13, 14)$$

$$f(A < B) = \sum m(1, 2, 3, 6, 7, 11)$$

$$f(A = B) = \sum m(0, 5, 10, 15)$$

A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
1	0	0	0	1	0	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	1	0	1
1	1	1	0	0	1	0
1	1	1	1	0	1	0

$\oplus \otimes A$ by A

f is 15 don't pass

if we flip all bits
will stay same

Simplified logic expression -

$$A > B : A \oplus A_1 \bar{B}_1 + (A_1 \odot B_1) A_0 \bar{B}_0$$

$$A < B : \bar{A}_1 B_1 + (A_1 \odot B_1) \bar{A}_0 B_0$$

$$A = B : (A_1 \odot B_1) (A_0 \odot B_0)$$

Similarly, for 3 bit comparator -

$$A > B : A_2 \bar{B}_2 + (A_2 \odot B_2) A_1 \bar{B}_1 + (A_0 \odot B_0) (A_1 \odot B_1) A_0 \bar{B}_0$$

$$A < B : \bar{A}_2 B_2 + (A_2 \odot B_2) \bar{A}_1 B_1 + (A_0 \odot B_0) (A_1 \odot B_1) \bar{A}_0 B_0$$

$$A = B : (A_2 \odot B_2) (A_1 \odot B_1) (A_0 \odot B_0)$$

Parity Circuit

used for error detection $\bar{S} = (\bar{s} < s) \oplus$

counting no. of 1's

3 bit message

Parity bit:

A	B	C	Y
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

A	$\bar{B}C$	$\bar{B}C$	BC	$B\bar{C}$
\bar{A}	0	1	3	1
A	1	4	5	7
				6

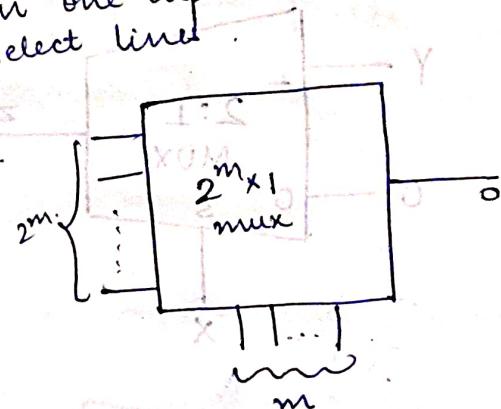
$$\therefore Y = A \oplus B \oplus C$$

Parity bit = 1 if odd
no. of 1's are present
in data stream.

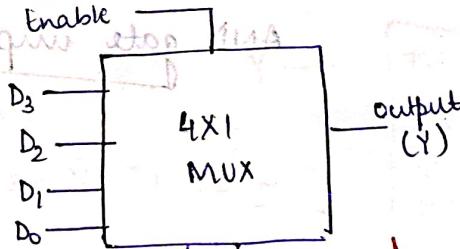
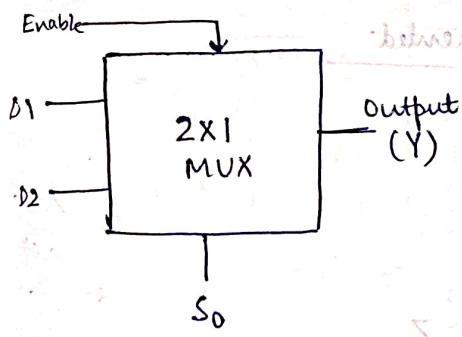
Multiplexer

- also called data selector.
- combinational circuit with more than one input lines, one output line and more than one select line.

Multiplexer $\rightarrow 2^m \times 1$ or $2^m : 1$ mux.
 $m = \text{no. of select lines.}$



2x1 MUX $Y = X \cdot \bar{S} + \bar{X} \cdot S$ 4x1 MUX



Select	Inputs	Output
0	0 0	0
0	0 1	0
1	1 0	1
1	1 1	1

Select	Y
S ₁ S ₀	
0 0	D ₀
0 1	D ₁
1 0	D ₂
1 1	D ₃

$$Y = X \cdot D_0 \bar{S} + D_1 S$$

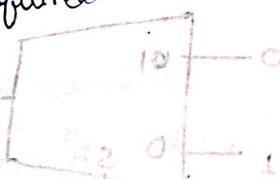
$$\begin{aligned} & (X \cdot D_0 \bar{S}) + (\bar{X} \cdot D_1 S) \\ \therefore Y &= \bar{S}_1 \bar{S}_0 D_0 + \\ & \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 \\ & + S_1 S_0 D_3 \end{aligned}$$

$$\begin{aligned} \text{No. of NAND gates} &= m + 2^m + 1. \\ \text{No. of NOT gates req.} &= \text{No. of select lines} \cdot (m) \\ \text{No. of AND gates req.} &= \text{No. of inputs} (2^m) \\ \text{No. of OR gates} &= 1. \end{aligned}$$

4 NAND gates

7 NAND gate required

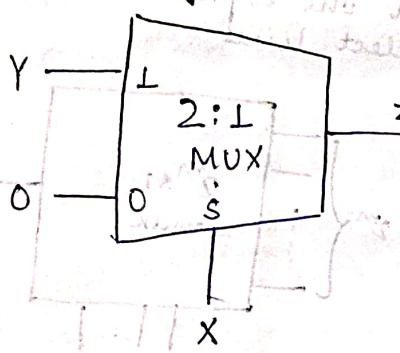
$$\begin{aligned} \text{No. of NAND gates req.} &= 2^{m+m+1} \\ &\downarrow \quad \downarrow \quad \downarrow \\ &\text{OR} \quad \text{NOT} \end{aligned}$$



Implementation of logic gates using MUX

DigitUM

1) AND gate



S = Y

Y = 0

Z = XY

0 = 0

1 = 1

0 = 0

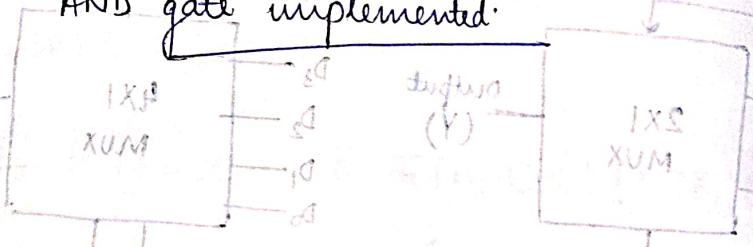
1 = 1

0 = 0

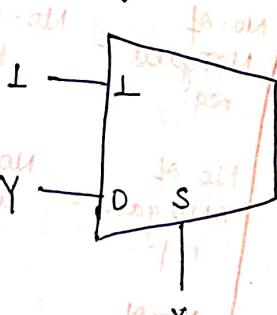
1 = 1

$$\text{Also, } Z = \overline{X} \cdot \overline{Y} + X \cdot Y = XY$$

AND gate implemented:



2) OR gate

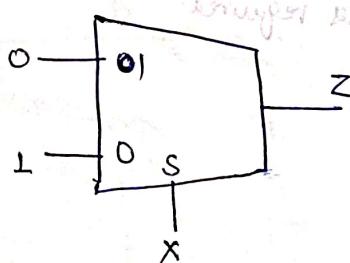


S	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

$$Z = \overline{X} \cdot \overline{Y} + X \cdot \overline{Y}$$

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

3) NOT gate

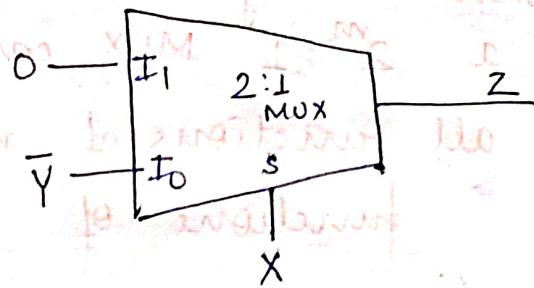


$$Z = \overline{X} \cdot 1 + X \cdot 0$$

$$= \overline{X}$$

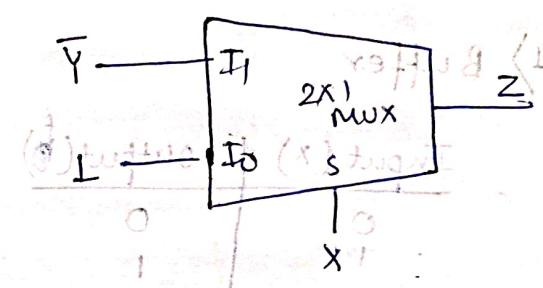
4) NOR gate

	X	Y	Z
S=0	0	0	1
S=1	1	0	0
			$Z = \bar{Y} = I_0$



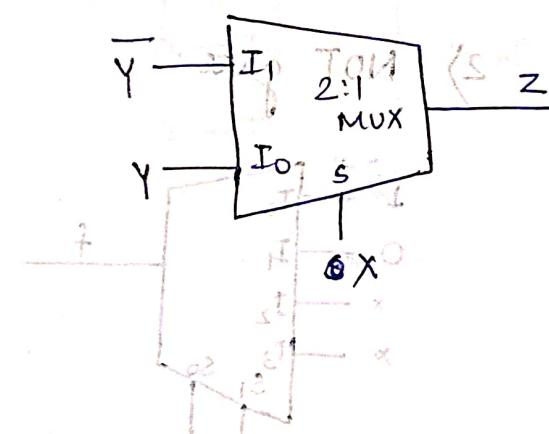
5) NAND gate

	X	Y	Z
S=0	0	0	1
S=1	1	0	0
			$Z = \bar{Y} = I_1$



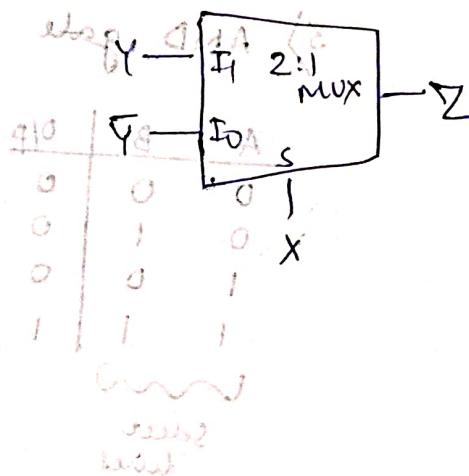
6) XOR gate

	X	Y	Z
S=0	0	0	0
S=1	1	0	1
			$Z = Y = I_1$
			$I = 0, 0, 0$
			$0, 1, 0$
			$1, 0, 1$



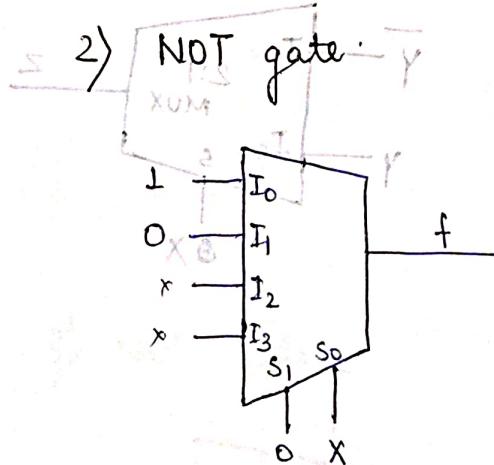
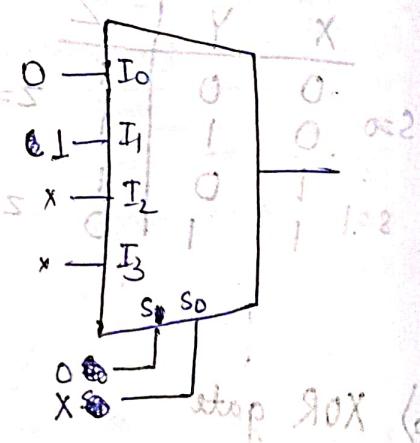
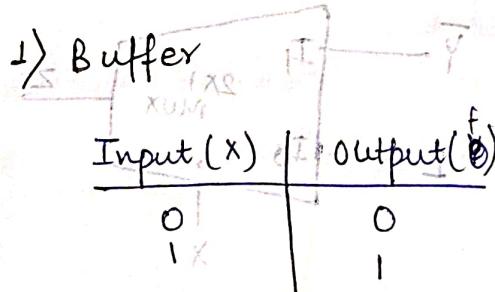
7) XNOR gate

	X	Y	Z
S=0	0	0	1
S=1	1	0	0
			$Z = \bar{Y} = I_0$
			$I = 0, 0, 0$
			$0, 1, 0$
			$1, 0, 1$



* Without any additional circuitry, step 9on
 a $2^m : 1$ MUX can be used to obtain
 all functions of m variables, only some
 functions of $(m+1)$ variables.

4:1 multiplexer as universal gate.



step 9on

S	Y	X
0	0	0
1	1	0

$f = \bar{X}$

$I_0 = 1$
 $I_1 = 0$

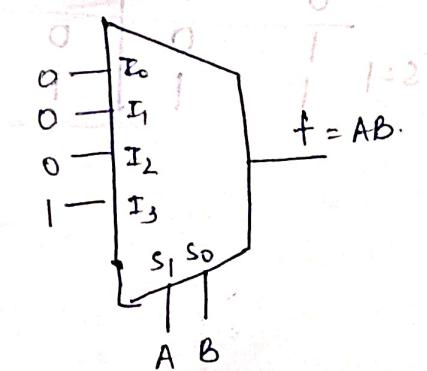
* choose selected line
 inputs of logic gate & then
 connect o/p of truth table
 as inputs of MUX.

3) AND gate

step 9on

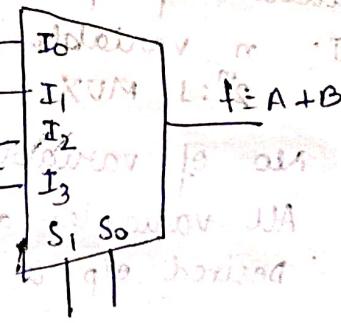
A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1

Select lines

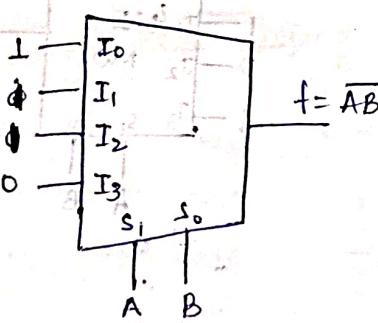


4) OR gate

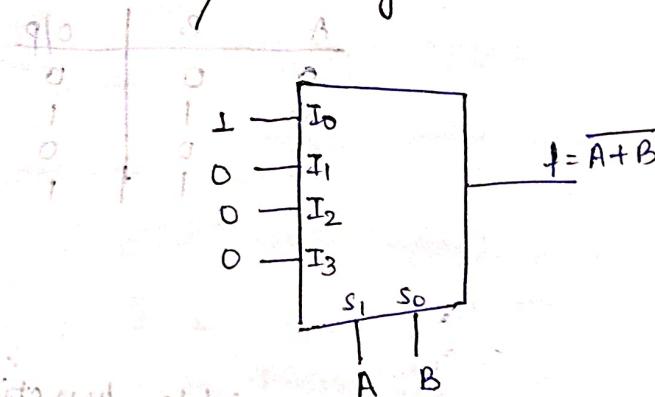
A	B	O/P
0	0	0
0	1	1
1	0	1



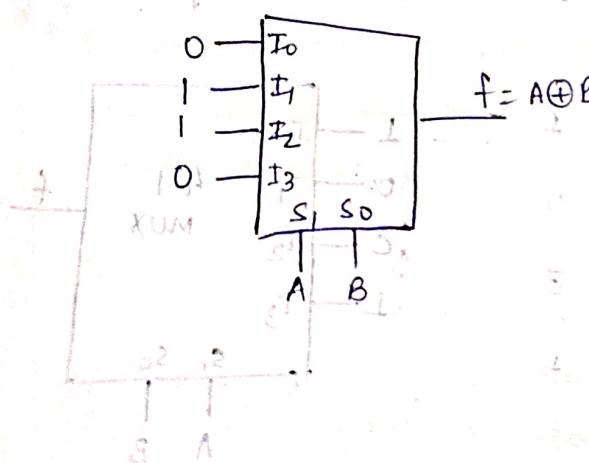
5) NAND gate



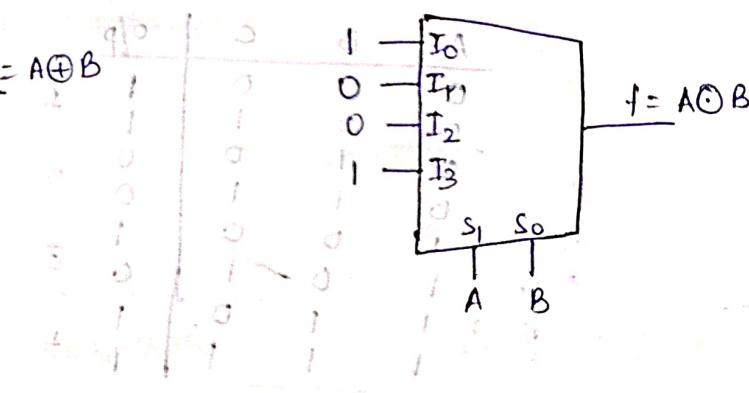
6) NOR gate



7) XOR gate



8) XNOR gate



11	10	11	00	01
①	②	③	④	⑤
⑥	⑦	⑧	⑨	⑩
⑪	⑫	⑬	⑭	⑮
⑯	⑰	⑱	⑲	⑳

Implementation of functions using multiplexer.

case I: n variable function

~~2ⁿ:1 MUX~~

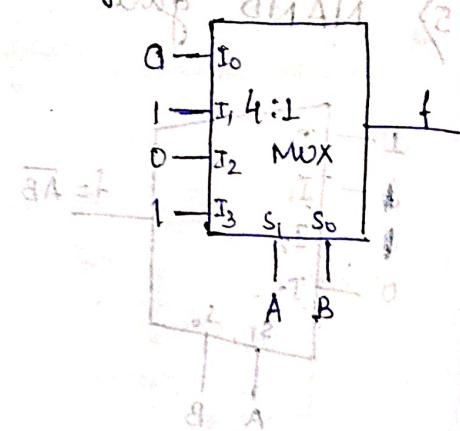
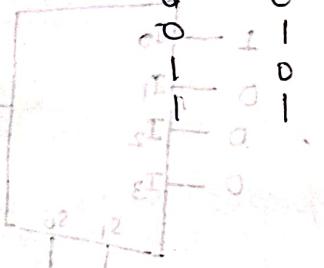
No. of variables = No. of select lines

All variables are connected to the select lines

Desired o/p is connected to the input lines.

Ex - $f(A, B) = \sum m(1, 3)$ using 4:1 MUX

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

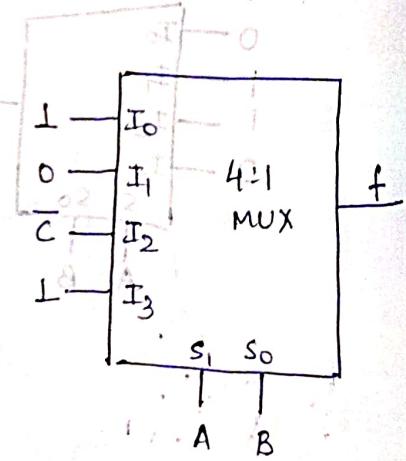


Case II: n variable function

~~2ⁿ⁻¹:1 MUX~~

Ex - $f(A, B, C) = \sum m(0, 1, 4, 6, 7)$ using 4:1 MUX

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



variables in select lines

Implementation Table.

variable to be used as o/p

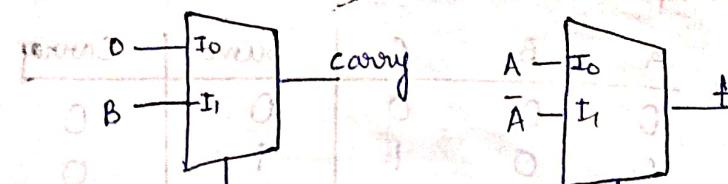
$\bar{C} 0$	00	01	10	11
0	0	2	4	6
$C 1$	1	0	\bar{C}	1
1	3	5	7	1

Half adder using 2:1 MUX

Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\text{Sum} = A \oplus B$$

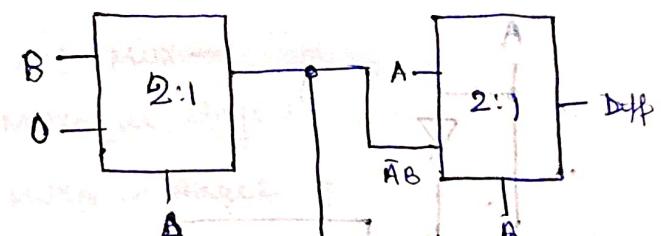
$$\text{Carry} = AB$$



Half subtractor using 2:1 MUX

Input		Diff.	Borrow
A	B	A	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\text{Diff} = A \oplus B, \text{ Borrow} = \bar{A}B$$



If complement is not available,

half adder requires 3 2:1 MUX

half subtractor requires 2 2:1 MUX.

Case III: n variable function

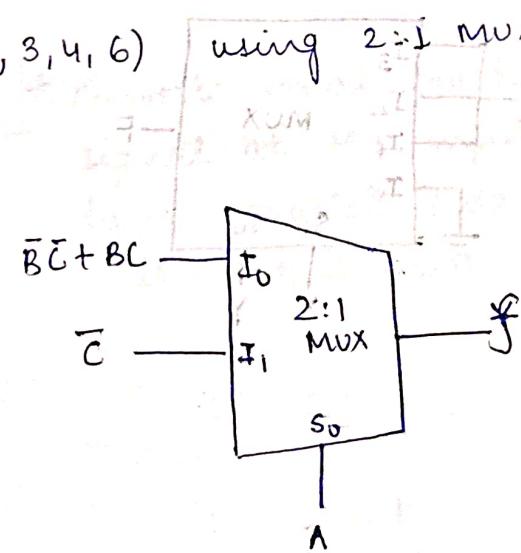
$2^{n-2} : 1$ MUX

$$\text{Ex- } F(A, B, C) = \sum m(0, 3, 4, 6)$$

using 2:1 MUX

Implementation table

$\bar{A}YX + A$	
$\bar{B}\bar{C}$	0 (4)
$\bar{B}C$	1 (5)
$B\bar{C}$	2 (6)
BC	3 (7)



Implementation of full adder using multiplexers - 4:1 MUX

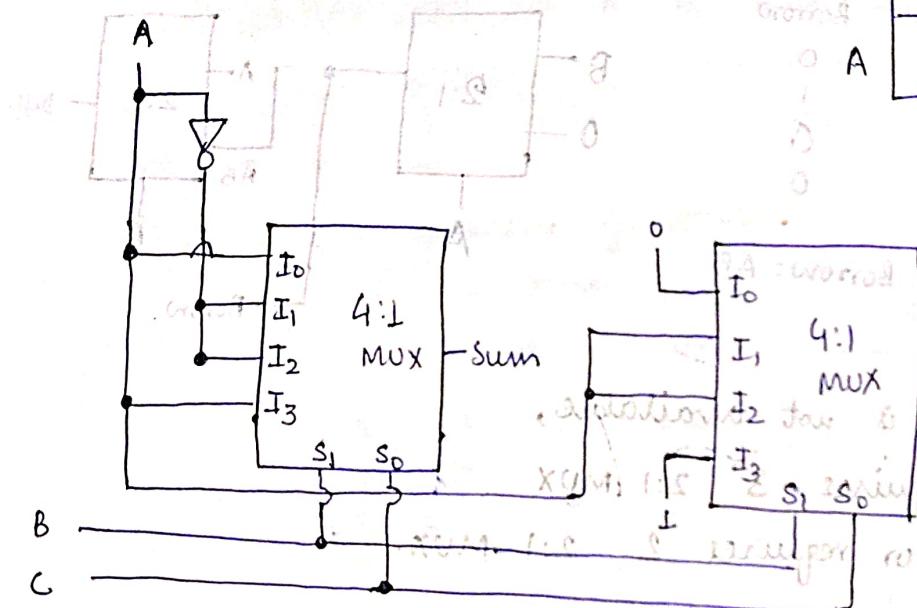
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Implementation table for sum

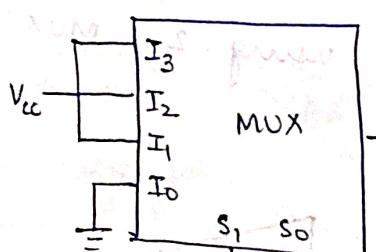
	$\bar{B}\bar{C}$	$\bar{B}C$	$B\bar{C}$	BC
\bar{A}	0	(1)	(2)	3
A	(4)	5	6	(7)

Implementation table for carry

	$\bar{B}\bar{C}$	$\bar{B}C$	$B\bar{C}$	BC
\bar{A}	0	1	2	(3)
A	4	(5)	(6)	(7)



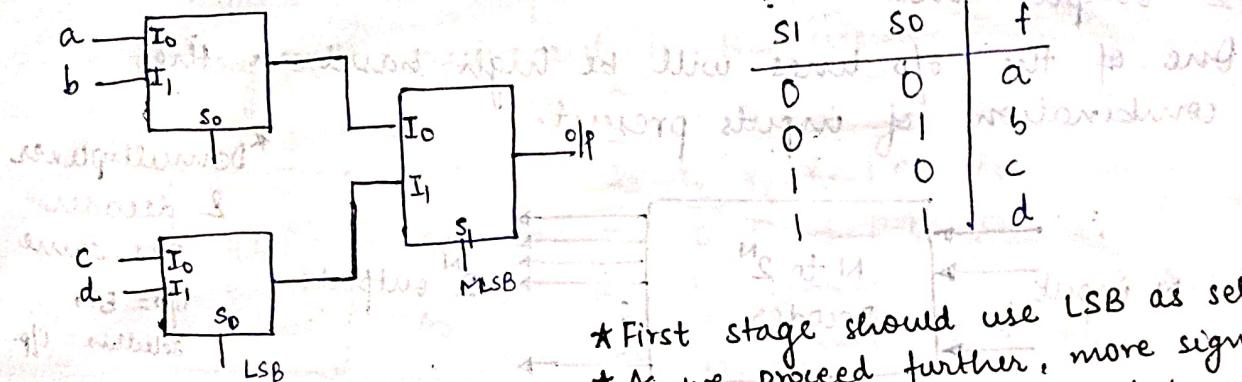
Output of multiplexer -



$$\begin{aligned}
 & \text{MUX output: } F = (I_0, I_1, I_2, I_3)_{S_1, S_0} \\
 & = \bar{X}\bar{Y} \cdot 0 + \bar{X} \cdot Y \cdot 1 + X\bar{Y} \cdot 1 + XY \cdot 1 \\
 & = \bar{X}Y + X\bar{Y} + XY \\
 & = \bar{X}Y + X(\bar{Y} + Y) \\
 & = \bar{X}Y + X \\
 & = (X + \bar{X})(X + Y) \\
 & = \underline{\underline{X + Y}}
 \end{aligned}$$

Implementation of higher order multiplexer using lower order MUXes

4.1 b) MUX using 2:1 MUX



- * First stage should use LSB as select lines
- * As we proceed further, more significant bits will be used as select lines.

General formula-

To implement B/A B:1 MUX using A:1 MUX

$$B/A = K_1 \rightarrow \text{no. of MUXes in stage 1}$$

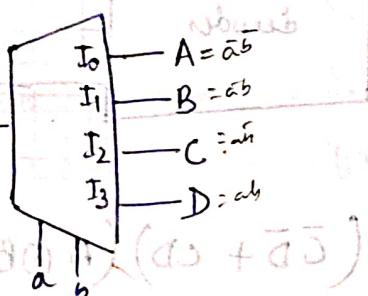
$$\frac{B}{K_1}/A = K_2 \rightarrow \text{no. of MUXes in stage 2}$$

$$K_{N-1}/A = K_N = 1 \quad (\text{till we obtain } 1)$$

$$\text{No. of MUXes required} = K_1 + K_2 + K_3 + \dots + L \text{ MUXes}$$

DEMULTIPLEXER

→ takes one single input data line & then switches it to any of the no. of individual o/p lines based on select lines.

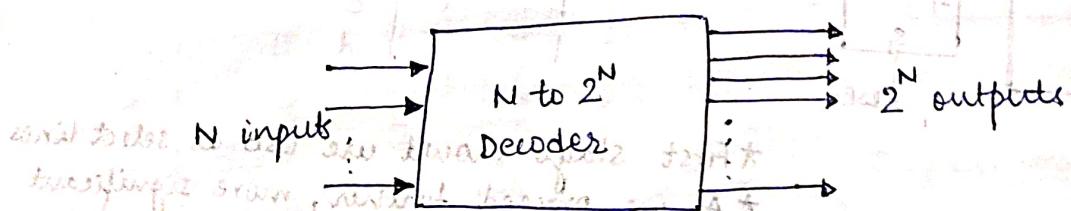


- * Converts serial data signal at the input to a parallel data at its output line.

$$(a + ā)(b + b̄)$$

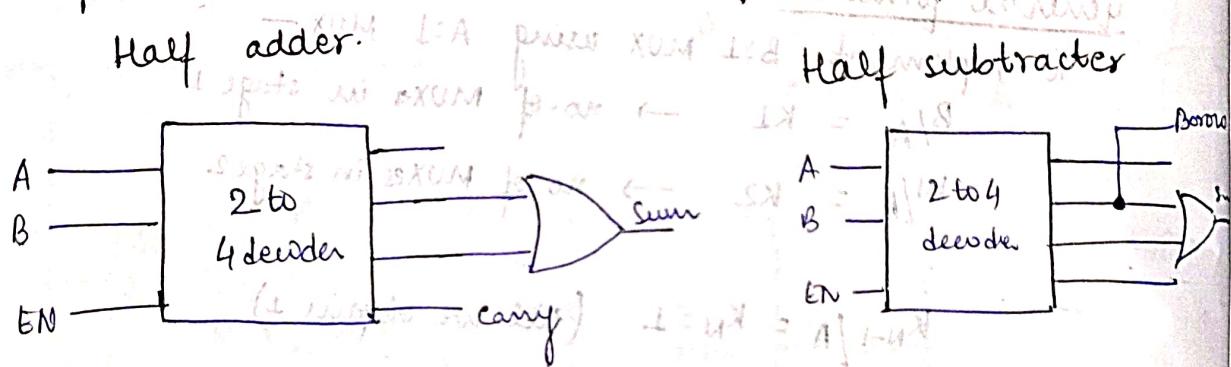
DECODER

Combinational circuit that has n input lines and maximum 2^n output lines.
One of the op lines will be high based on the combination of inputs present.

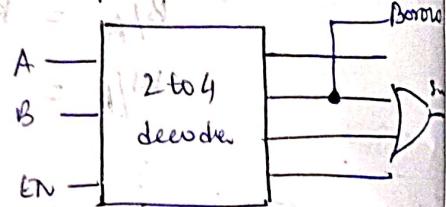


* Demultiplexer & decoder are same
i/p = EN
selective = i/p.

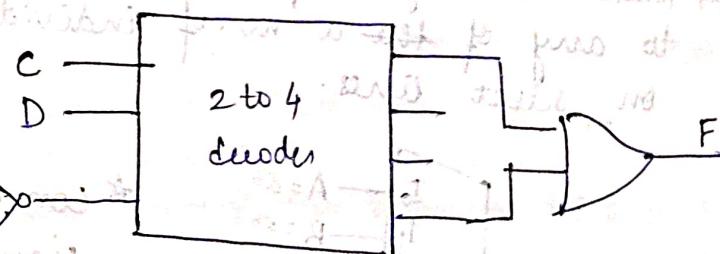
Implementation of functions using decoder.



Half subtractor



* If No. of variables > No. of inputs,
Then, few variables are used as enable input



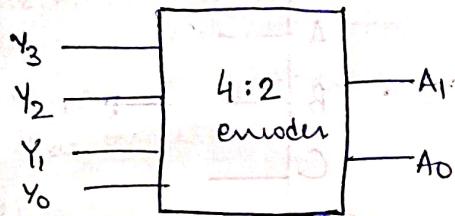
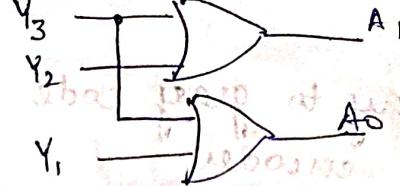
$$F = (\bar{C}\bar{D} + CD)(A \oplus B)$$

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

ENCODER

2^N inputs N outputs

binary code equivalent to the input.



Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	0	0	0	1	1

$$A_1 = Y_3 + Y_2$$

$$A_0 = Y_1 + Y_3$$

Priority encoder

If more than one i/p are 1, then, o/p depends upon the highest priority.

Y_3	Y_2	Y_1	Y_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

$\bar{Y}_3 \bar{Y}_2$	$\bar{Y}_1 \bar{Y}_0$	$\bar{Y}_1 Y_0$	$Y_1 Y_0$	$Y_1 \bar{Y}_0$
0	1	3	2	
1	4	5	7	6
1	12	10	15	14
1	8	9	11	10

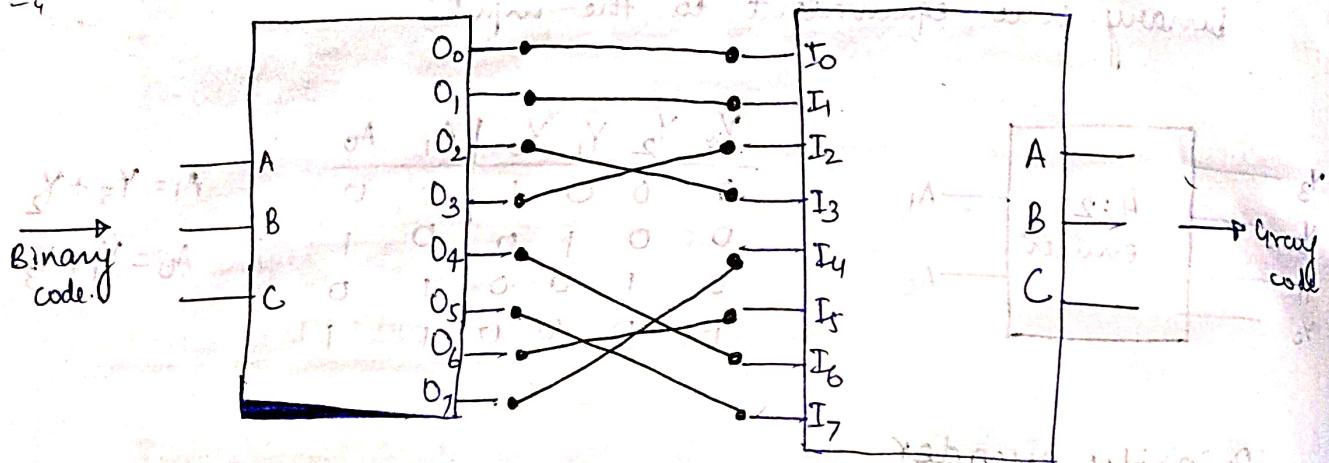
$$A_1 = Y_2 + Y_3$$

$\bar{Y}_3 \bar{Y}_2$	$\bar{Y}_1 \bar{Y}_0$	$\bar{Y}_1 Y_0$	$Y_1 Y_0$	$Y_1 \bar{Y}_0$
0	1	3	2	
4	5	7	6	
12	10	15	14	
8	9	11	10	

$$A_0 = Y_3 + Y_1 \bar{Y}_2$$

0-0
1-1
2-3
3-2
4-6
5-7
6-5
7-4

Binary to gray code converter using 3:8 decoder and 8:3 encoder



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

	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	0
001	0	1	0	1	0	1	0	1
010	1	0	1	0	1	0	1	0
011	1	0	1	1	0	1	1	0
100	1	1	0	0	1	0	0	1
101	1	1	0	0	1	0	1	1
110	1	1	0	0	1	1	0	1
111	1	1	0	0	1	1	1	1

	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	0
001	0	1	0	1	0	1	0	1
010	1	0	1	0	1	0	1	0
011	1	0	1	1	0	1	1	0
100	1	1	0	0	1	0	0	1
101	1	1	0	0	1	0	1	1
110	1	1	0	0	1	1	0	1
111	1	1	0	0	1	1	1	1

Sequential Circuit

Combinational logic -
output depends upon given combination of inputs.

Sequential logic -
output depends upon combination of inputs as well as present state of system

To store current state of system, in order to use it in next cycle, memory elements are needed.

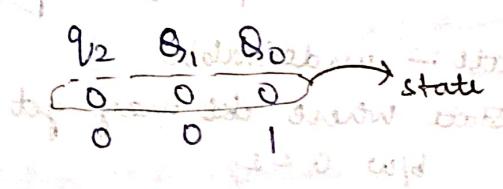
latches flip flops

State and State Variables

State :- present stored value in the system

State variable :- variable used to represent stored value in the system.

Ex → 3 bit counter



Q_2, Q_1, Q_0 :- state variables

Combinational circuits

- ① O/p only depends upon present state of inputs.
- ② No memory element required.
- ③ Faster
- ④ Easier to design

Sequential circuits

- ① Output depends upon past output and present input.
- ② Memory elements are required.
- ③ Slower
- ④ Difficult to design.

Synchronous Circuits

- ① all variables change at the same time in synchronism with clock.

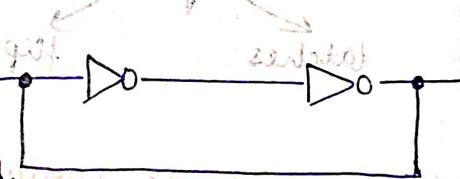
- ② o/p of system can change only when clock signal arrives & in the meantime, if i/p changes, the system does not detect it.

Asynchronous Circuits

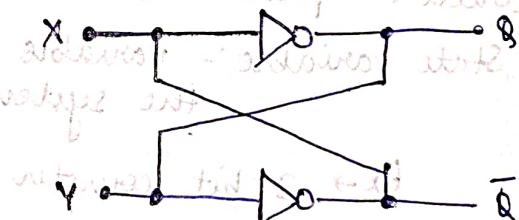
- ① circuit does not need a clock signal to operate.
- ② As soon as i/p/p changes, the change is reflected to the o/p as well.

Bistable Multivibrator

→ Both states 0 & 1 are stable & ckt can stay in any of these states for infinite time until disturbed.



In such a circuit, a particular value of I/O gets locked/stored/latched.



* metastable state - undesirable state where ckt may get b/w 0 & 1.

[Ball full Analogy]

Stable state - The system remains in a state even after disturbance is applied.

Unstable state - The system moves to other states even without disturbance.

Metastable state - The system remains in that state if not disturbed. If disturbed, it changes state.

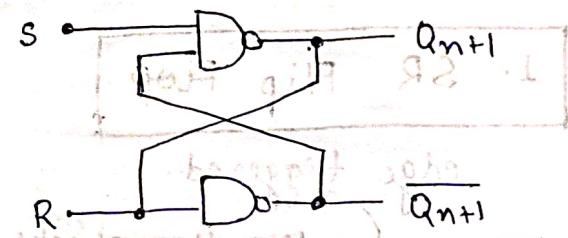
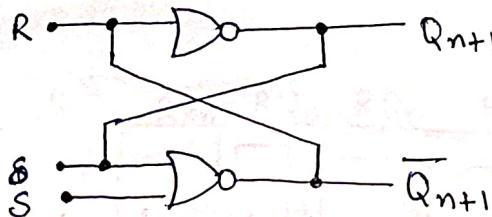
LATCHES

basic memory element that is used to store 1 bit of information.

Latches

Active High - i/p becomes active when HIGH
(using NOR gate)

Active Low - i/p becomes active when LOW.
(using NAND gate)

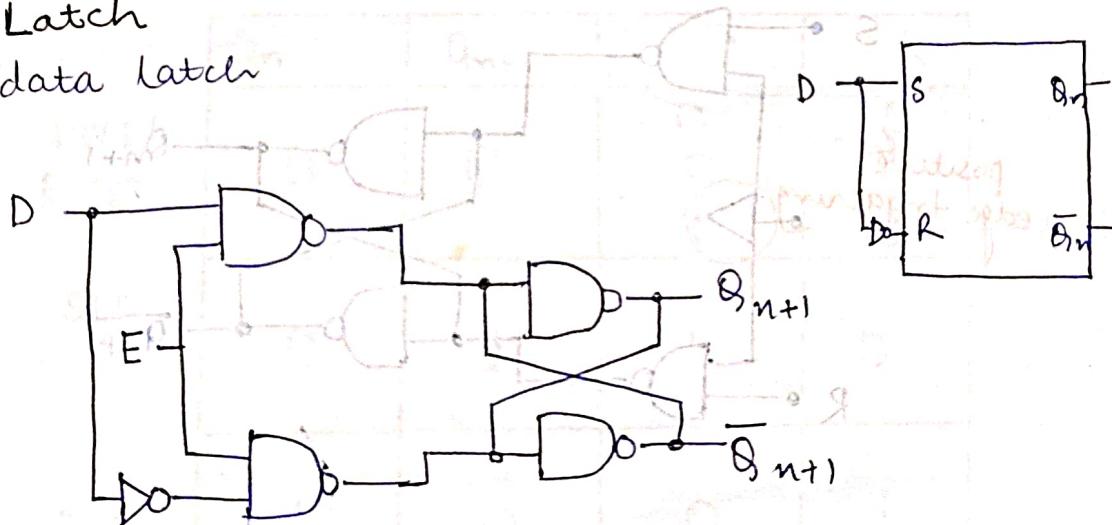


S	R	Q_{n+1}	\bar{Q}_{n+1}	State
0	0	Q_n	\bar{Q}_n	HOLD
0	1	0	1	RESET
1	0	1	0	SET
1	1	0	0	INVALID

S	R	Q_{n+1}	\bar{Q}_{n+1}	State
0	0	1	1	INVALID
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q_n	\bar{Q}_{n+1}	HOLD

D-Latch

data latch



also known as LSR latch with enable i/p.

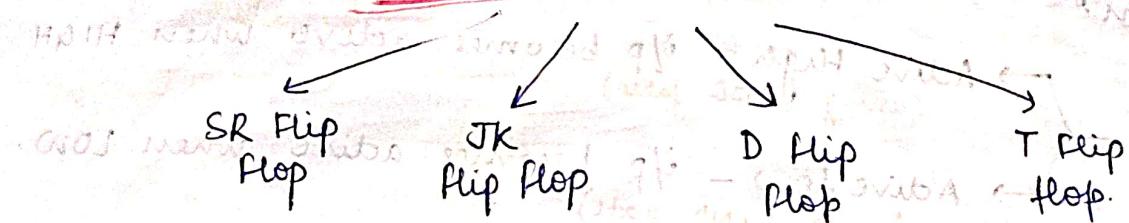
D	E	Q_{n+1}	\bar{Q}_{n+1}	Hold
X	0	Hold	Hold	Hold
0	1	0	1	0
1	1	1	0	1

$\rightarrow S=D$ $R=D$

also called
transparent latch

$$Q_{n+1} = D$$

FLIP FLOPS



1. SR Flip-flop

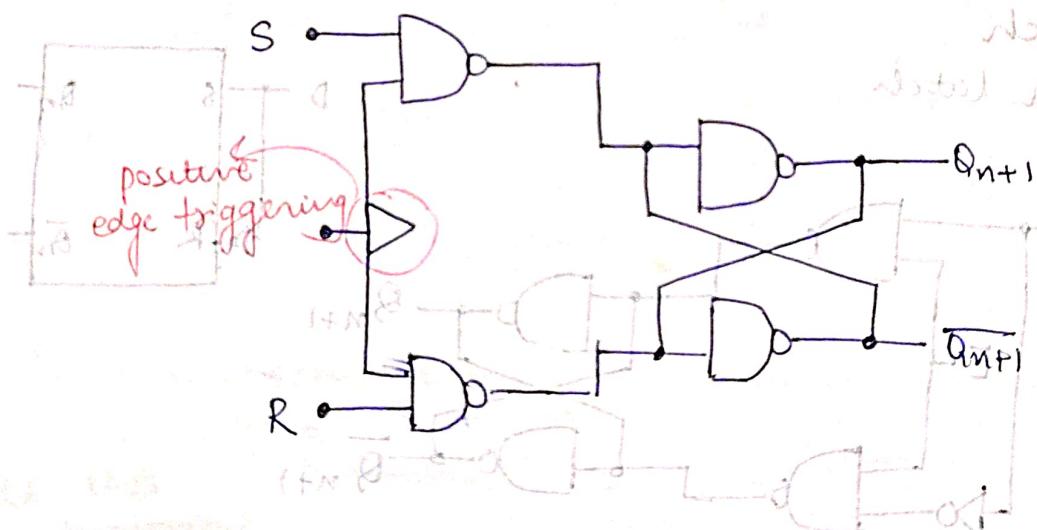
edge triggered.

→ flip flop checks the o/p values at desired edge of clock and produces corresponding o/p.

Inputs	T _{J2}	T _{J2J3}	T _{J3}	T _{J3J2}	T _{J2}	T _{J3}	T _{J3J2}
Q _{n+1}	1	0	0	0	0	1	0
T _{J2}	0	1	0	0	1	0	0
T _{J2J3}	1	0	1	0	0	1	0
Q _n	0	1	0	1	0	1	0

positive edge triggering

negative edge triggering



After \rightarrow state Table

Clock	S	R	Q _{n+1}	Q _{n+1}	Q _n	Q _n	Hold
$\uparrow t = 0$	0	X	X	X	Q _n	\bar{Q}_n	HOLD
$\uparrow t = 0.04$	0	1	1	0	Q _n	\bar{Q}_n	HOLD
$\uparrow t = 0.08$	0	0	0	0	L	L	Reset
$\uparrow t = 0.12$	0	0	L	L	0	0	Set
$\uparrow t = 0.16$	1	1	1	1	1	1	Invalid

↳ characteristic Table.

S	R	Q_n	Q_{n+1}
0	0	0	0 } HOLD
0	0	1	1 } RESET
0	1	0	0 } SET
0	1	0	1 } INVALID
1	0	1	1 }
1	0	1	1 }
1	1	0	1 }
1	1	1	1 }

S	$\bar{R}Q_n$	$\bar{R}Q_n$	RQ_n	$R\bar{Q}_n$
0	0	1	1	2
1	1	1	1	1

$$Q_{n+1} = S + \bar{R}Q_n$$

characteristic equation

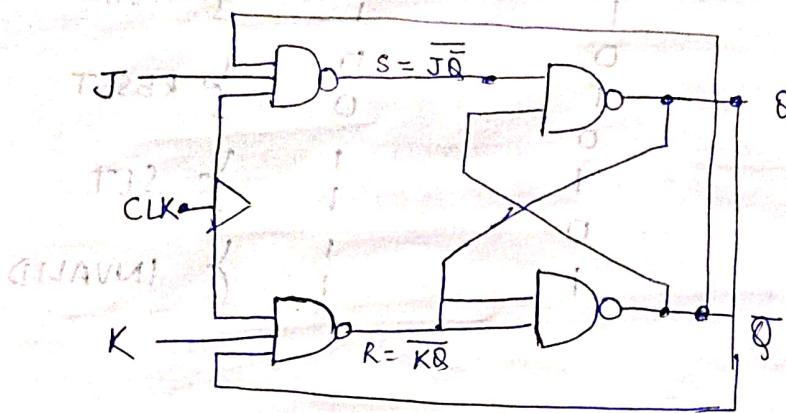
↳ Excitation Table

determine inputs of flip flop using present and next state.

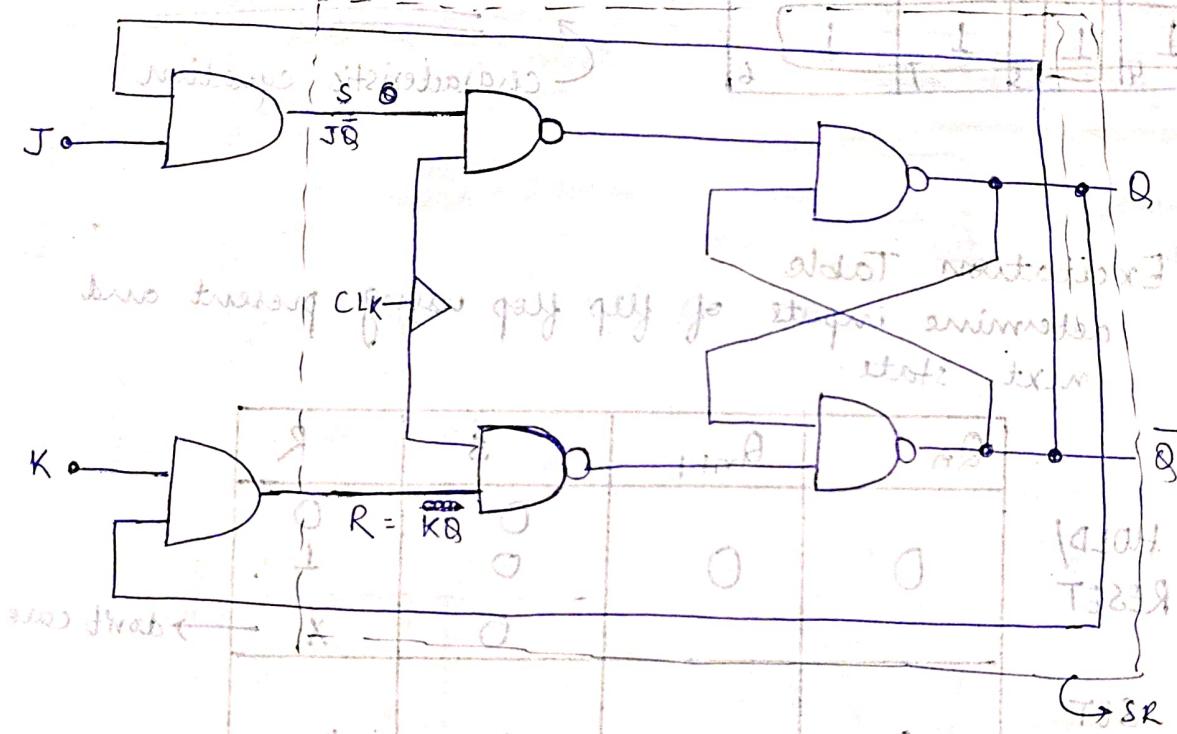
Q_n	Q_{n+1}	S	R
0	0	0	0 } HOLD/RESET
0	1	0	1 } SET
1	0	1 } RESET	1 }
1	1	0 } HOLD/SET	0 }

→ don't care.

2. JK Flip Flop



JK flip flop using SR flip flop.



↳ State table

$$S = \bar{J}\bar{Q} \quad R = KQ$$

J	K	S	R	Q_{n+1}	\bar{Q}_{n+1}	
0	0	0	0	Q_n	\bar{Q}_n	HOLD
0	1	0	1	0	1	RESET
1	0	1	0	1	0	SET
1	1	Toggle		Q_n	\bar{Q}_n	TOGGLE

Characteristic Table -

<u>J</u>	<u>K</u>	<u>Q_n</u>	<u>Q_{n+1}</u>
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	1	1
1	1	0	0

HOLD

RESET

SET

TOGGLE

<u>J</u>	<u>KQ_n</u>	<u>K_{Qn}</u>	<u>K_{Qn}</u>	<u>K_{Qn}</u>
0	0	1	1	0
1	1	0	0	1

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

Excitation Table -

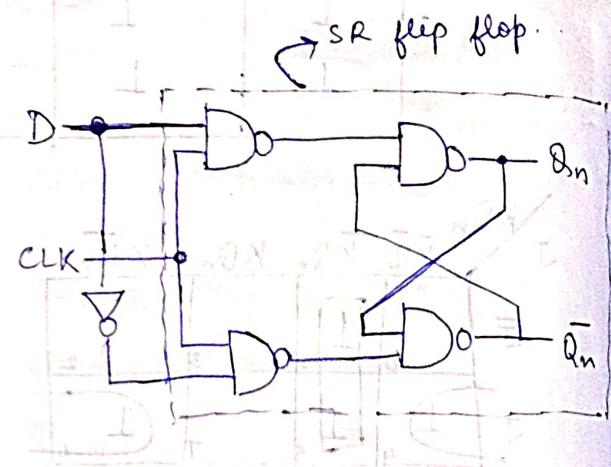
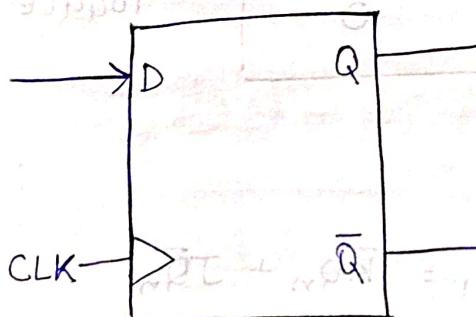
<u>Q_n</u>	<u>Q_{n+1}</u>	<u>J</u>	<u>K</u>
0	0	0	0
0	1	1	1
1	0	0	X
1	1	1	0
1	0	0	1
X	X	X	X

* No race around condition in edge triggering

Condition for race around $\rightarrow T_p > \frac{T_{CLK}}{2}$

3. D - Flip Flop

↳ Transparent flip flop.



↳ State Table.

D	CLK	S	R	Q _{n+1}
X	-	X	X	Q _n
0	↑	0	1	0
1	↑	1	0	1

* The I/p D propagates as it is to the o/p IT
so, it is called transparent flip flop.

↳ characteristic Table.

CLK	D	Q_n	Q_{n+1}
-	x	0	0
-	x	1	1
↑	0	0	0
↑	0	1	0
↑	1	0	1
↑	1	1	1

$$Q_{n+1} = D$$

↳ Excitation Table.

CLK	Q_n	Q_{n+1}	D
Q0H →	0	0	0
	0	1	1
	1	0	0
INPUT →	1	1	1

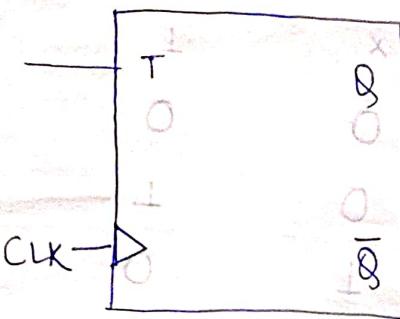
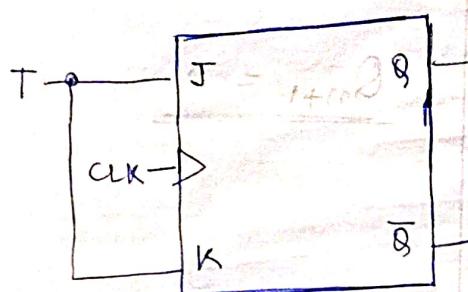
* if value of D changes at the same instant the clock edge arrives then, output is based on value of input just before the edge.

14H	8H	AT	X1D
0	0	0	0
1	1	1	1
0	0	0	1
1	1	1	1
0	1	1	1
1	0	1	1
0	1	1	1
1	0	1	1

$$X1D = 14H$$

4. T - Flip Flop

↳ Toggle Flip Flop



↳ State Table.

CLK	T	J	K	Q_{n+1}
-	x	x	x	Q_n
↑	0	0	0	0 \leftarrow HOLD
↑	0	1	0	1 \leftarrow TOGGLE
↑	1	1	1	1

↳ characteristic Table.

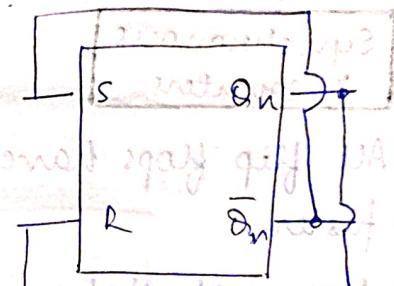
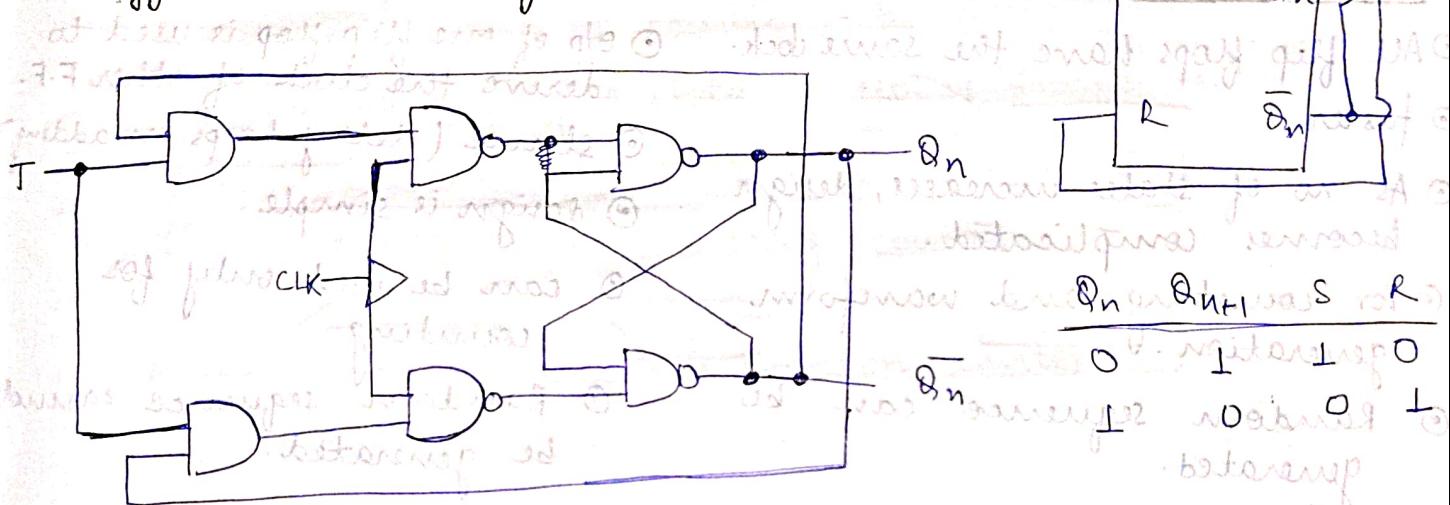
CLK	T	Q_n	Q_{n+1}
-	x	0	0
-	x	1	1
↑	0	0	0
↑	0	1	1
↑	1	0	1
↑	1	1	0

$$\underline{\underline{Q_{n+1} = T \oplus Q_n}}$$

↳ Excitation Table
(प्रोग्रेस फॉर माइक्रोसिप्ट)

	Q_n	Q_{n+1}	T
at time t	0	0	0
0	0	1	1
1	1	0	1
1	1	1	0

Toggle circuit using SR flip flop -



* All flip flops are free from race around (because they are edge triggered)

* D-flip flops are used in buffer registers because they can store data temporarily (no hold state)

Counters (Application of flip flop)

Type of sequential circuit whose state represents the number of clock pulses given and can be used to count the number of clock pulses.

Types of counters

Synchronous counters

- ① All flip flops have the same clock.
- ② faster
- ③ As no. of states increases, design becomes complicated.
- ④ For counting and waveform generation.
- ⑤ Random sequence can be generated.

Asynchronous counters

- ⑥ O/p of one flip flop is used to derive the clock of other F.
- ⑦ slower (delay keeps on adding)
- ⑧ design is simple.
- ⑨ can be used only for counting
- ⑩ Random sequence cannot be generated.

Important Terminology

1. Up Counter :-

at every clock pulse, the count increases until it reaches maximum.

2. Down Counter :-

at every clock pulse, count reduces by 1 until it reaches minimum.

3. Modulus of a counter :-

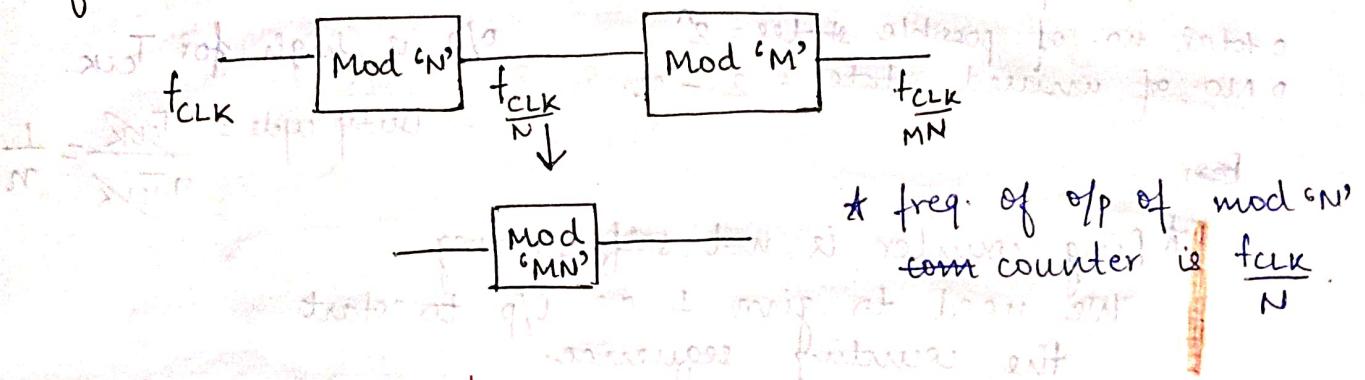
number of states in the counting sequence of counter.

4. Lock out :-

- ⑪ It cannot come back to desired states once.

condition when the counter is stuck in a sequence of states which are not part of the counting sequence.

- When 2 counters are cascaded, the overall modulus of combination is product of modulus of 2 counters

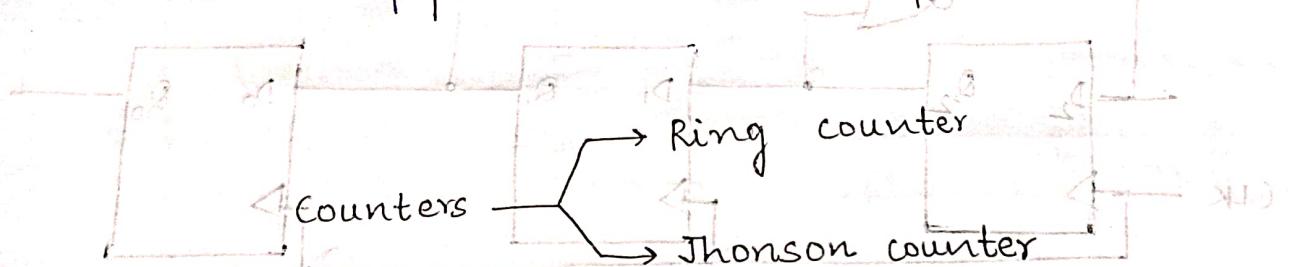


* Asynchronous counter

With 'n' flip flops,

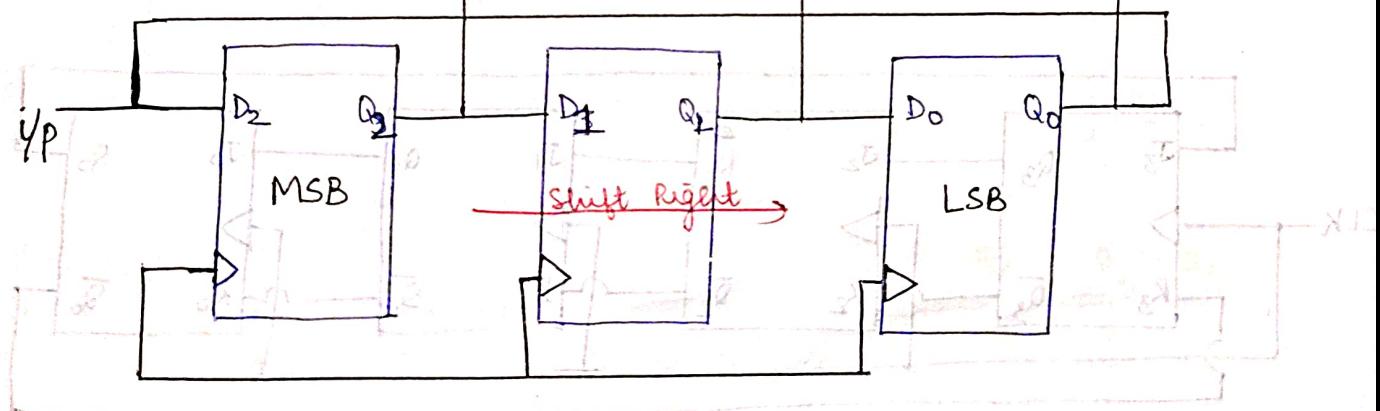
$$\text{Max. no. of possible states} = 2^n$$

The max. no. of possible states \geq No. of desired states.



1. Ring counter

o/p of last bit is fed as input to the first stage.



also called shift register counter.

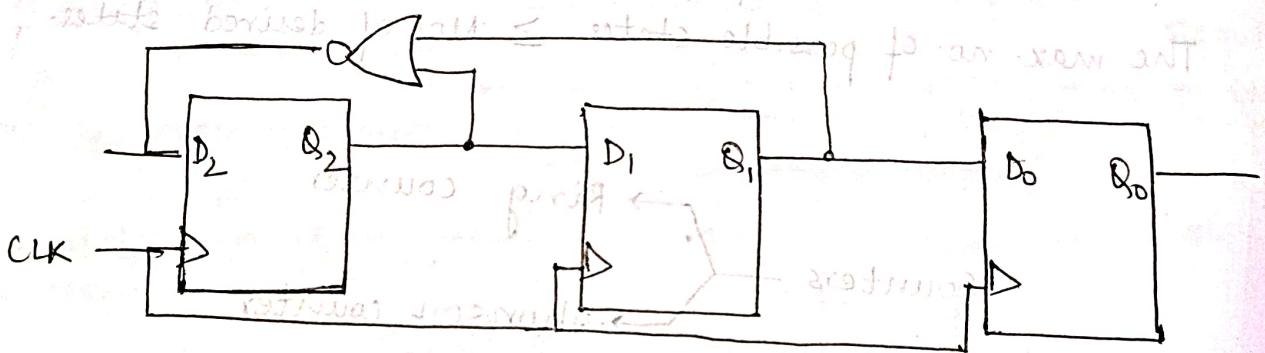
o only one o/p can be high at a time.

o Number of states = No. of flip flops.

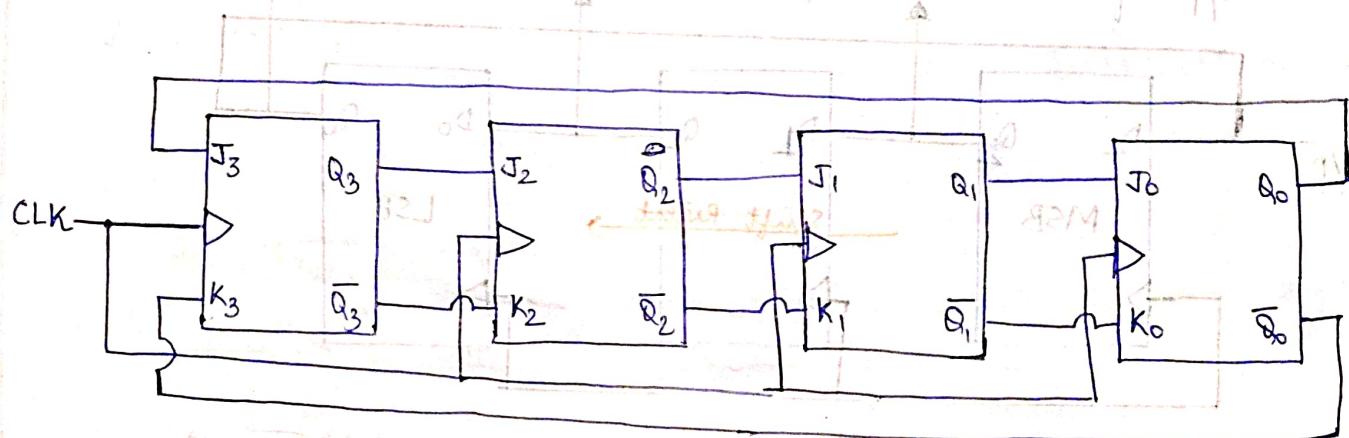
n bit ring counter.
 \downarrow n flip flops
 \downarrow no. of states
 o total no. of possible states = 2^n
 o No. of unused states = $2^n - n$.
 o/p is high for T_{CK}
 Duty cycle = $\frac{T_{CK}}{nT_{CK}} = \frac{1}{n}$

Non
 * Ring counter is not self starting.
 We need to give 1 as i/p to start the counting sequence.

Self starting ring counter.



Ring counter with JK flip flop.



3 bit RING counter states Hello Golu

100 010 001 011

010 001 100 110

001 110 010 101

2. Johnson counter

also known as Twisted ring counter

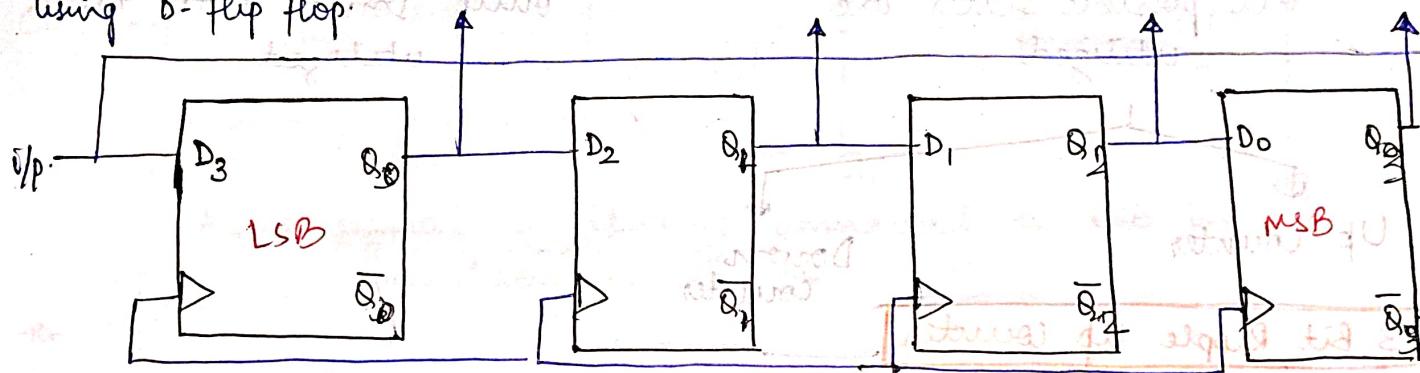
Switched tail counter

Möbius counter.

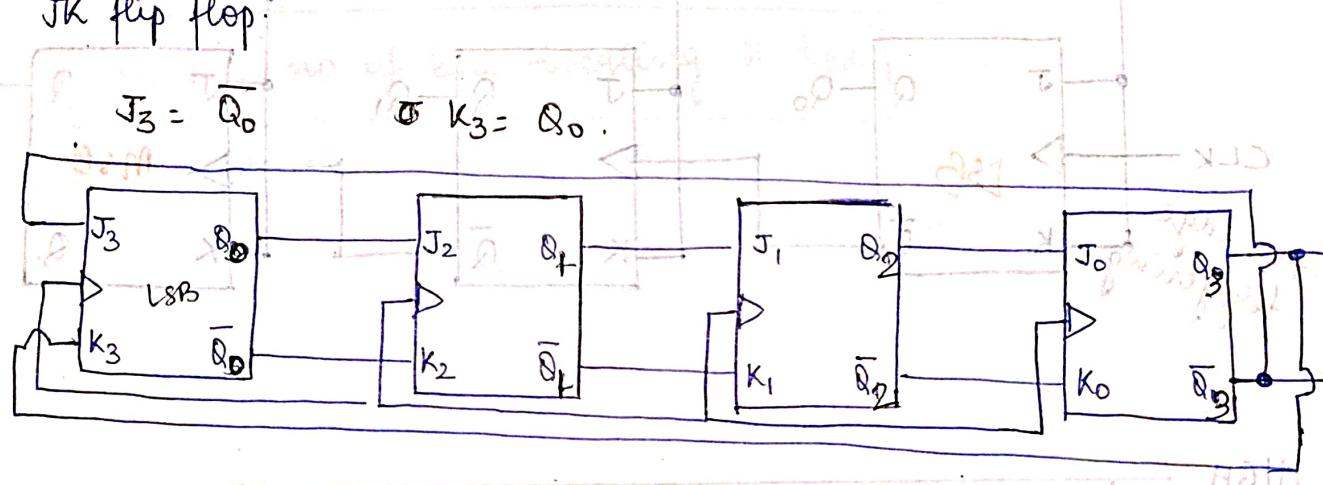
Logic diagram —

↳ only one change wrt. ring counter is that complement of o/p of last stage is fed as i/p to the first stage.

using D-flip flop:



using JK flip flop:



For n flip flops,

$$\text{no. of used states} = 2^n$$

$$\text{unused states} = 2^n - 2^n$$

	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1

B at antennae \Rightarrow AND \leftarrow principle of O/p 1 0 1 1 1

Q at antennae \Rightarrow AND \leftarrow principle of O/p 0 0 1 1 1

above effect in stage 7 q3 0 q2 0 1 q1 0 1

Asynchronous counters

In asynchronous counters, the output of previous state acts as the input to the next state.

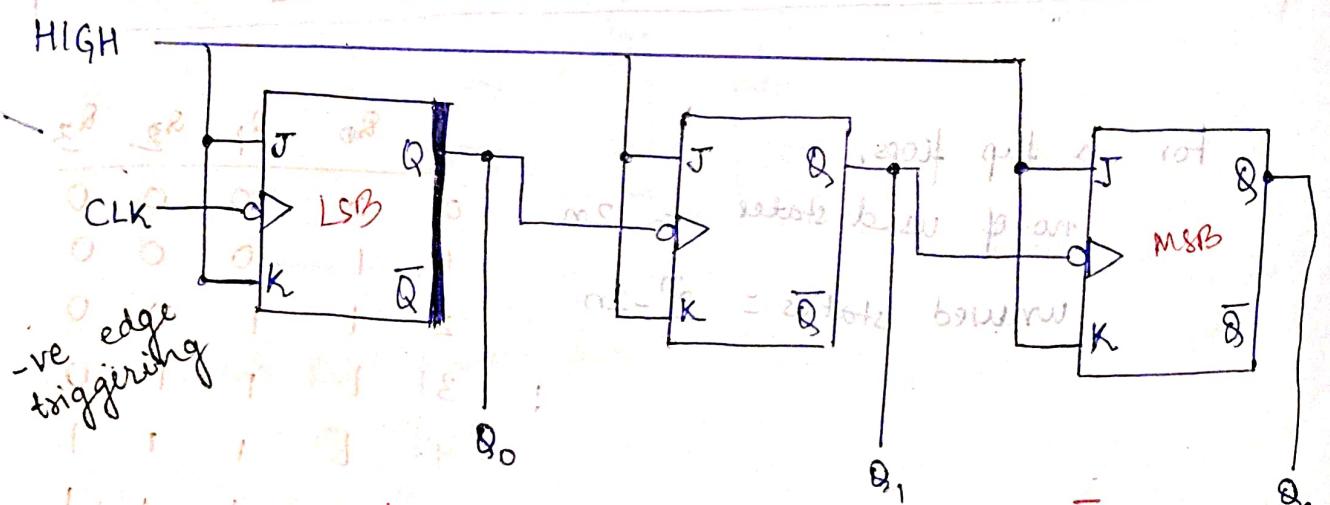
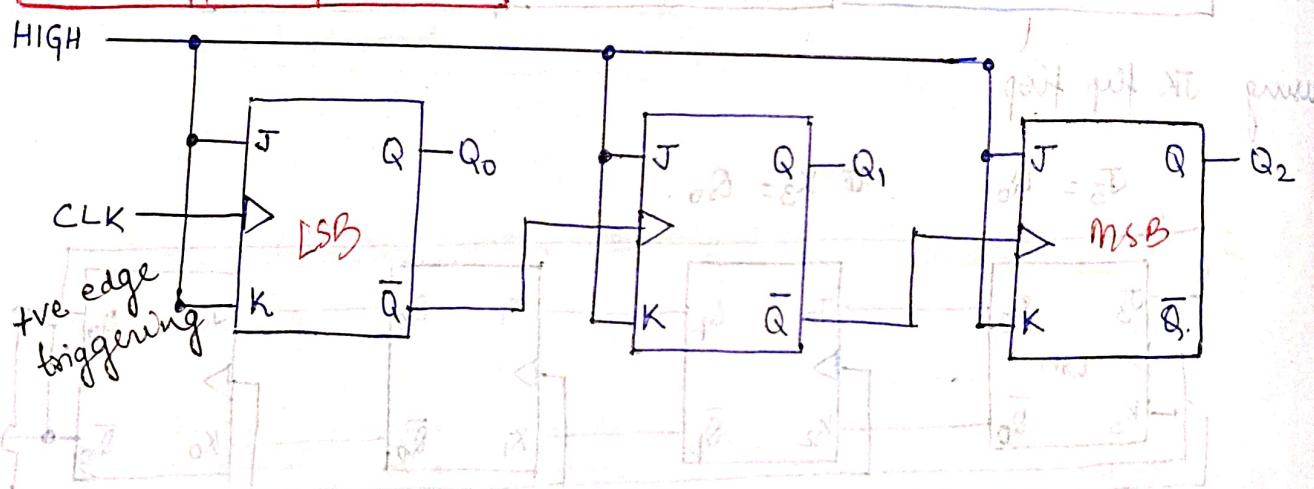
Binary
 $MOD = 2^n$

all possible states are utilized

Non-binary
 $MOD < 2^n$

only some states are utilized.

3 Bit Ripple Up Counter



1. tve edge triggering \rightarrow CLK is connected to \bar{Q}

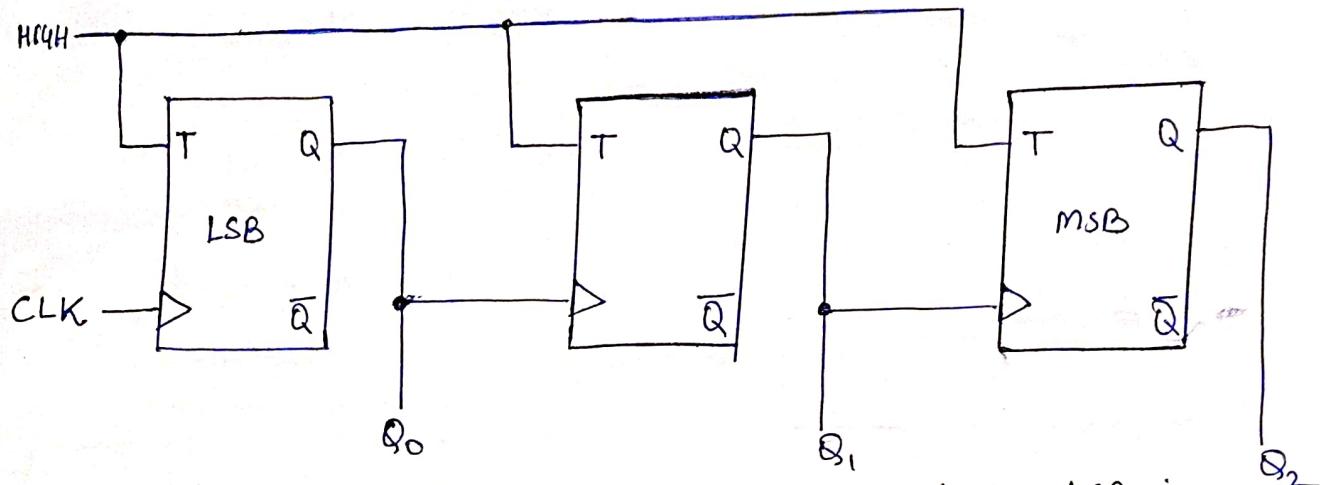
2. -ve edge triggering \rightarrow CLK is connected to Q

3. all flip flops operate in toggle mode.

3 Bit Ripple Down Counter

Triggering logic is opposite to an up counter.

positive edge triggered \rightarrow CLK is connected to Q
negative edge triggered \rightarrow CLK is connected to \bar{Q}



* CLK signal is always connected to LSB in asynchronous circuit.

$$\text{delay} = \text{no. of bits toggling} * t_{pd}$$

DIGITAL

LOGIC



Principle of duality -

If expression contains only AND OR and NOT,

Dual of expression can be obtained by

① changing AND to OR and OR to AND

② changing 1 to 0 and 0 to 1

③ variables and their complement remain unchanged

Practicing drawing and solving will help.

Consensus Theorem

→ 3 variables in expression

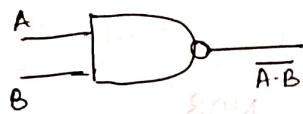
→ each variable is repeated twice

→ one variable is present in complemented & uncomplemented form.

$$\boxed{\overline{A}B + AC + BC = \overline{A}B + AC}$$

Terms containing complemented variable.

Logic gates



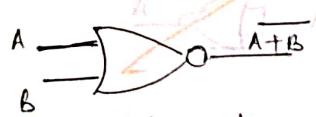
NAND gate.

Commutative

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

Not associative.

$$\overline{A \cdot B \cdot C} \neq \overline{A \cdot B} \cdot C$$



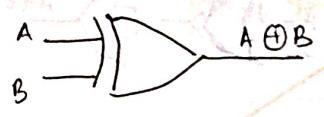
NOR gate

Commutative

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

Not associative.

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



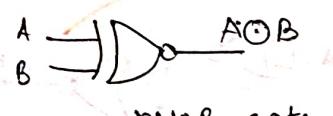
XOR gate

Commutative \wedge

$$A \oplus B = B \oplus A$$

Associative \wedge

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



XNOR gate

Commutative \wedge

$$A \odot B = B \odot A$$

Associative \wedge

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

* NAND and NOR gates are called universal gates.

* NOR gates have advantages over NAND gates

↳ becoz of simpler transistor structure, NOR gates are faster than NAND gates.

↳ NOR gates consume less energy than NAND gates.
∴ beneficial for low power applications

* Advantages of NAND over NOR -

because of these reasons, NAND is preferred in industry.

↳ NAND has higher memory capacity than NOR.

↳ NOR is more expensive than NAND.

↳ Size of NOR is greater than size of NAND.

↳ all transistors of NAND are of equal sizes where as NOR gates don't.

NAND vs NOR

stop sign

~~Logic gate~~

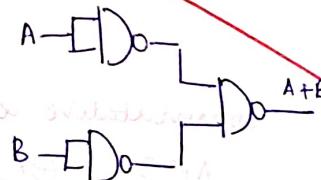
NAND

NOR

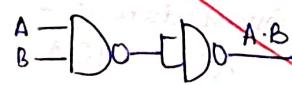
NOT A



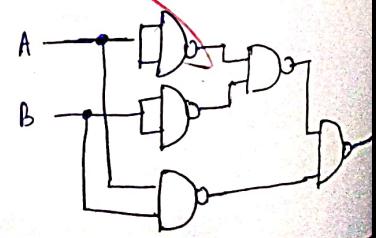
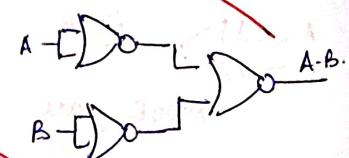
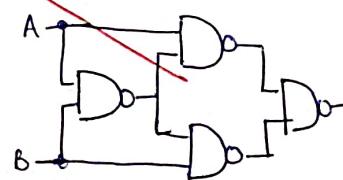
A+B



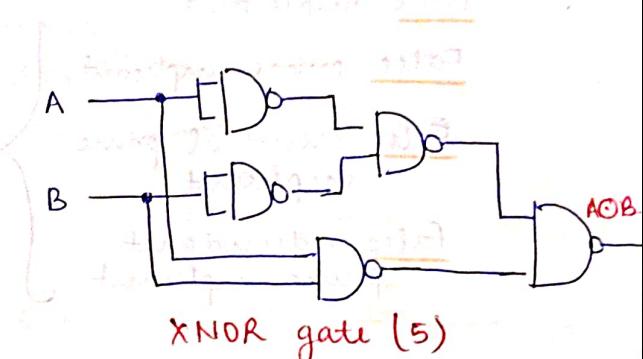
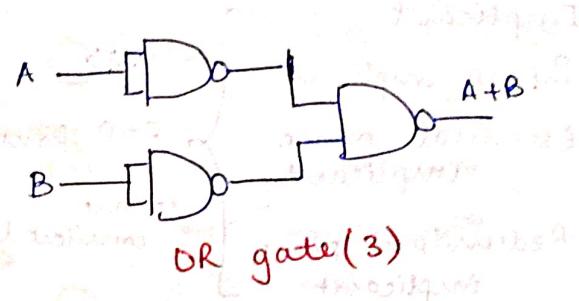
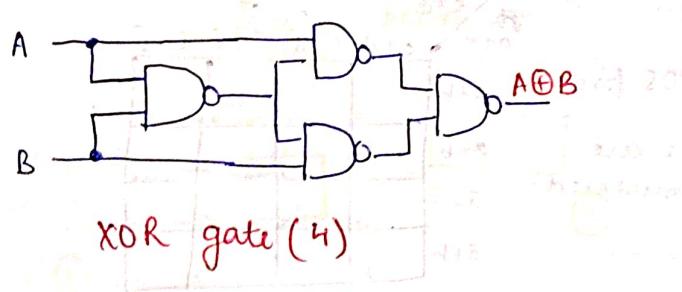
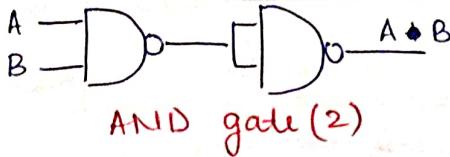
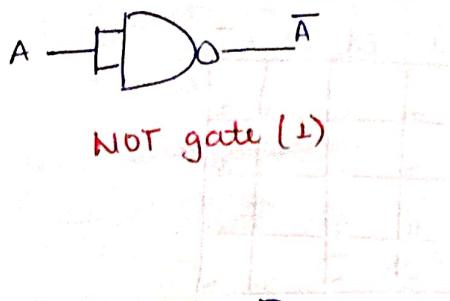
A·B



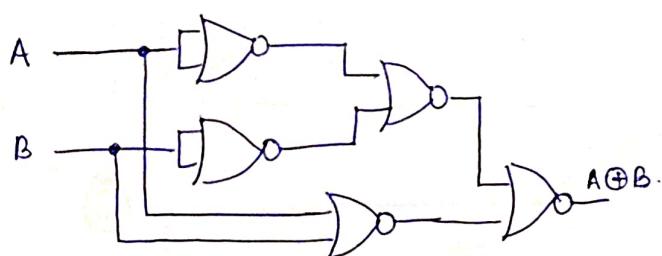
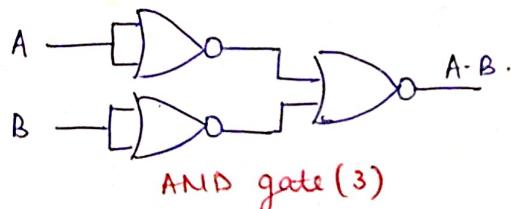
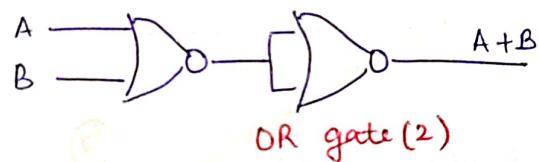
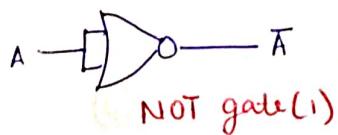
A⊕B



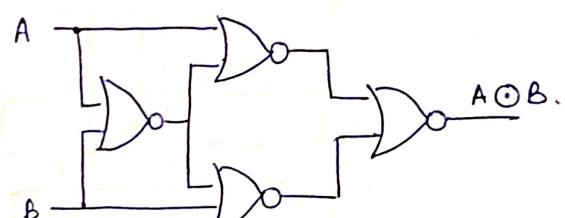
NAND gate circuits



NOR gate circuits



XOR gate (5).



XNOR gate (4)

K-Map important

Implicant

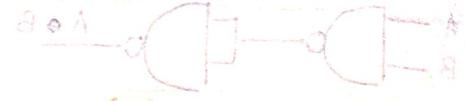
Prime implicant

Essential prime implicant

Redundant prime implicant

SOP form
is considered

ab	cd	$\bar{c}\bar{d}$	$\bar{c}d$	$c\bar{d}$	cd	$c\bar{d}$
$\bar{a}b$	0	1	3	2		
$\bar{a}b$	4	5	7	6		
ab	11	12	13	14		
$a\bar{b}$	8	9	11	10		



(s) step 001

False implicant

False prime implicant

False essential prime implicant

False redundant prime implicant

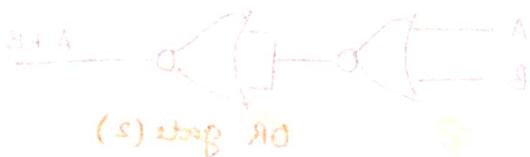
(s) step 001

POS form

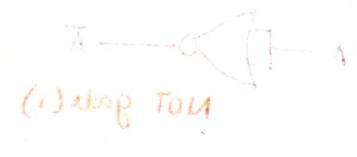
0s are considered.

$\bar{a}b$	$c\bar{d}$	$c+d$	$\bar{c}+d$	$\bar{c}+\bar{d}$	$c+\bar{d}$
$a\bar{b}$	0	1	3	2	
$a\bar{b}$	4	5	7	6	
$\bar{a}b$	12	13	15	14	
$\bar{a}b$	8	9	11	10	

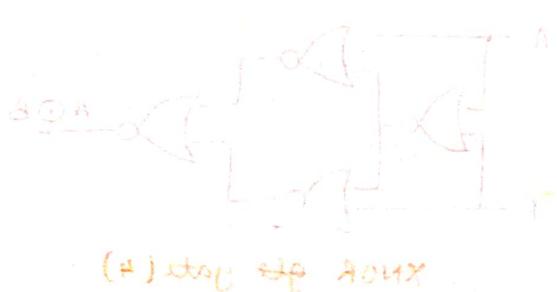
(s) step 001



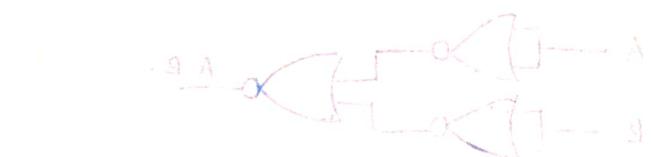
(s) step 001



(s) step 001



(s) step 001



(s) step 001

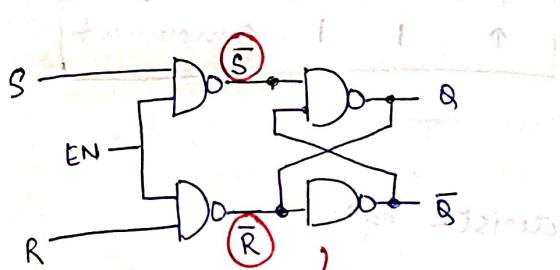


(s) step 001

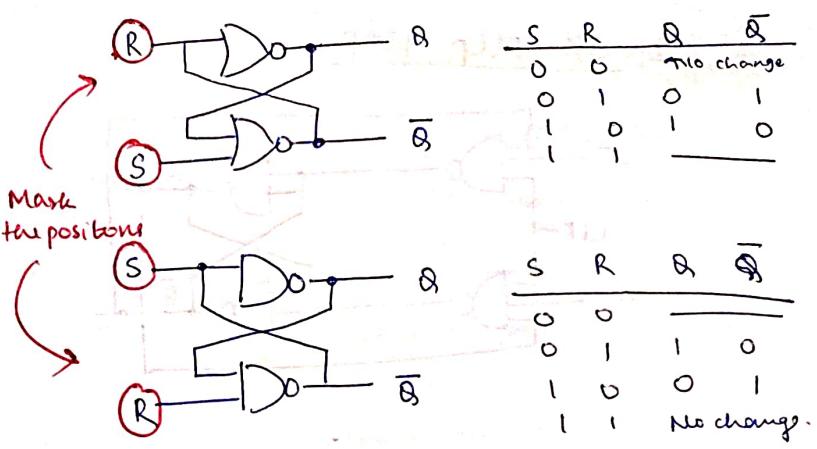
Sequential Circuits

NOR gate - Active High
NAND gate - Active Low

using ENABLE input & NAND gates

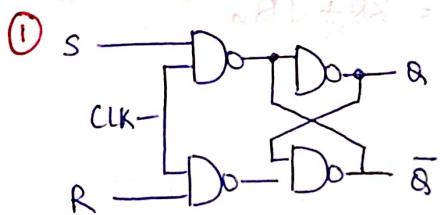


behaves as ACTIVE HIGH.

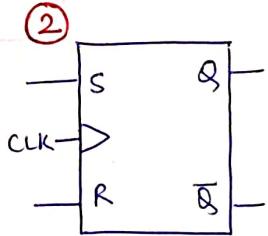


- ① Truth Table / Functions Table
- ② characteristic table
- ③ characteristic equation
- ④ Excitation Table.

1. SR Flip-flop



CLK	S	R	Q	\bar{Q}
0	x	x	No change	
\uparrow	0	0	No change	
\uparrow	0	1	0	1
\uparrow	1	0	1	0
\uparrow	1	1	-	-



④

S	R	Q_n	Q_{n+1}
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	-

⑤ characteristic equation

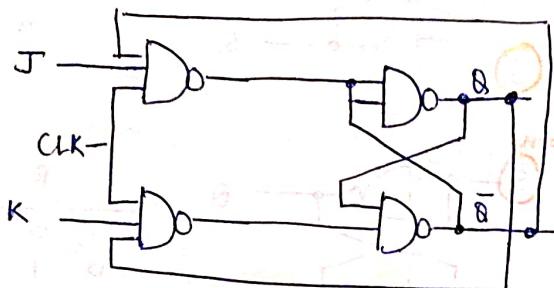
$\bar{R}Q_n$	S	$\bar{R}Q_n$	$\bar{R}Q_n$	RQ_n	RQ_n
0	1	1	1	1	1
1	1	1	1	1	1

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

⑥

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

2. JK Flip flop.



CLK	J	K	Q _n	Q _{n+1}
0	x	x	-	No change
↑	0	0	-	No change
↑	0	1	-	-
↑	1	0	-	-
↑	1	1	-	complement

J	K	Q _n	Q _{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

characteristic eqn.

J	K	Q _n	Q _{n+1}
0	0	1	1
1	1	1	0

$$Q_{n+1} = \bar{K}Q_n + \bar{J}Q_n$$



* Race around condition occurs in level triggered flip flop

when $J=K=1$ and $t_{pd} < T_{CLK}$

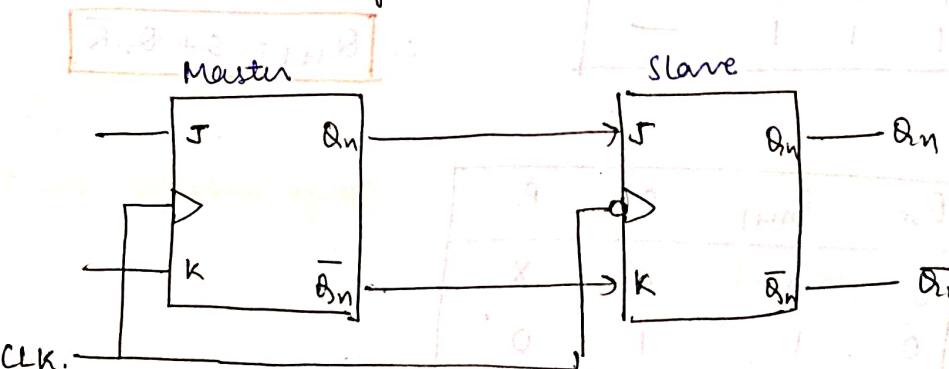
the state of flip flop oscillates b/w 0 and 1.

solution

→ Master slave level triggered flip flop

→ $t_{pd} > T_{CLK}$.

Master slave configuration —



When $CLK=1$, master is in operating mode

When $CLK=0$, slave is in operating mode
(-ve level triggered)

In master slave FFs —

↳ O/p may change only once

↳ change in o/p is triggered by -ve edge of clock.

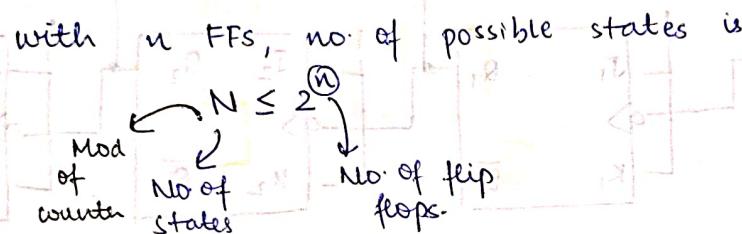
↳ change may occur only during clock's -ve level.

Counters

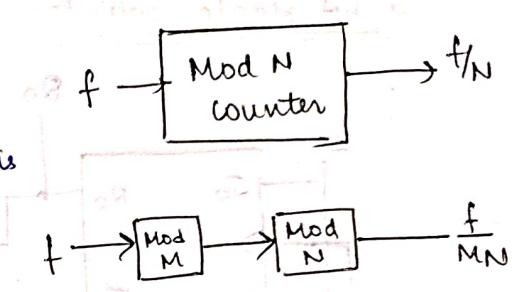
① sequential logic circuit capable of counting no. of clock pulses arriving at its CLK input.

② can be used as frequency divider.

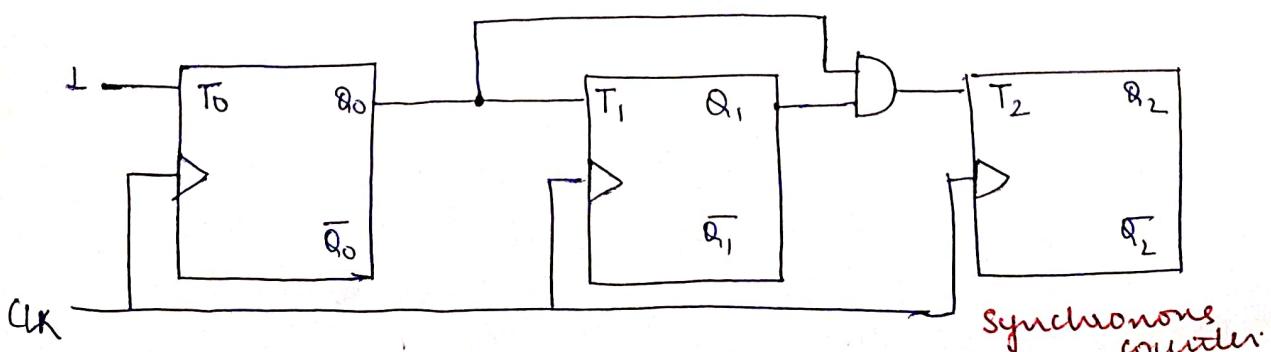
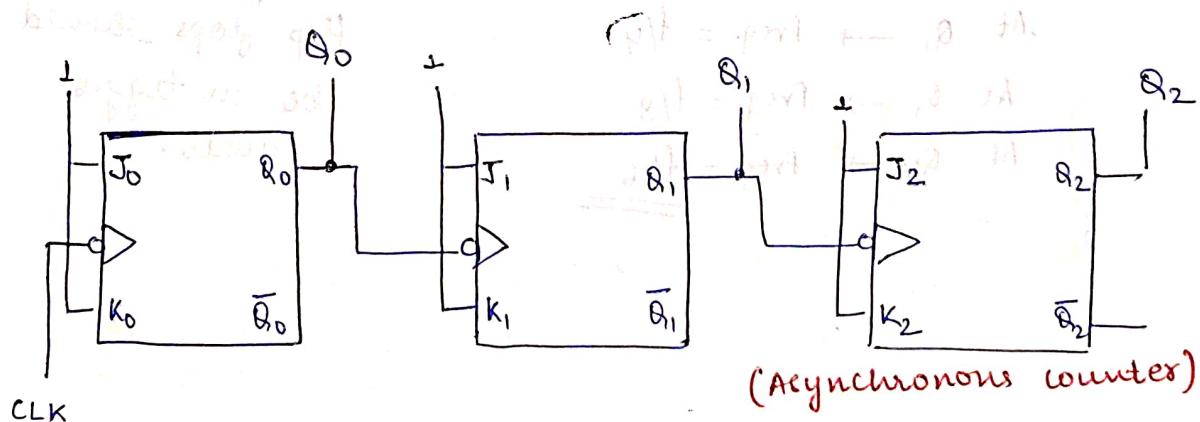
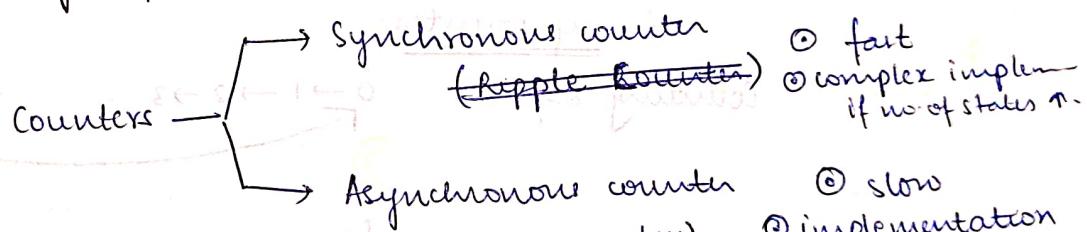
with n FFs, no. of possible states is



$N \leq 2^n$
Mod of counter
No. of states
No. of flip-flops.



Depending upon clock pulse applied,



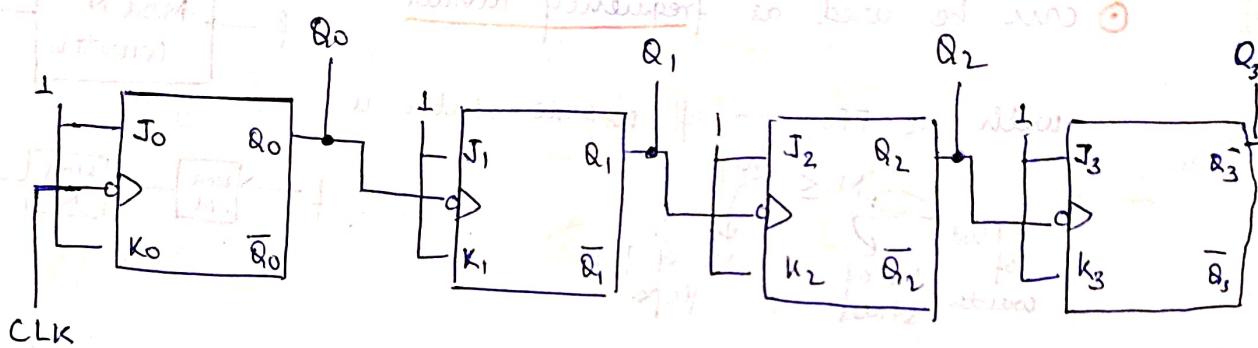
Imp.

- ① -ve edge triggered FF + Q as clock up counter
- ② -ve edge triggered FF + \bar{Q} as clock down counter
- ③ +ve edge triggered FF + Q as clock down counter
- ④ +ve edge triggered FF + \bar{Q} as clock up counter

counter >

stands for what was the stages there signal describes? Q
Signal goes to previous stage

4 bit ripple counter



-ve edge triggered flip flops.

Q_0 is connected to clock

up counter

Counting seq. \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow ... \rightarrow 14 \rightarrow 15

At $(Q_0 \rightarrow)$ Freq. = $f/2$

At $Q_1 \rightarrow$ freq. = $f/4$

At $Q_2 \rightarrow$ freq. = $f/8$

At $Q_3 \rightarrow$ freq. = $f/16$

For ripple counter,
flip flops should
be in toggle
mode.

(when we want to)