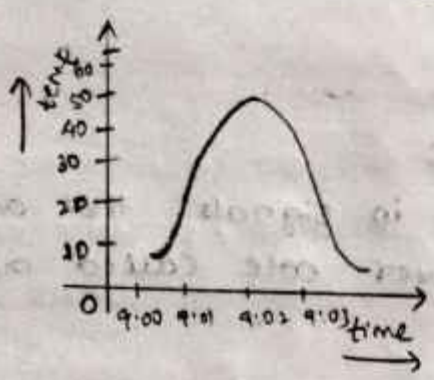
6/9/22 DIGITAL SYSTEMS AND BINARY NUMBERS.

* SIGNAL :- A signal is a function represents the variation of physical quantity with respect to any Parameters.

* ANALOG SIGNAL :- All real life signals are analog in nature.

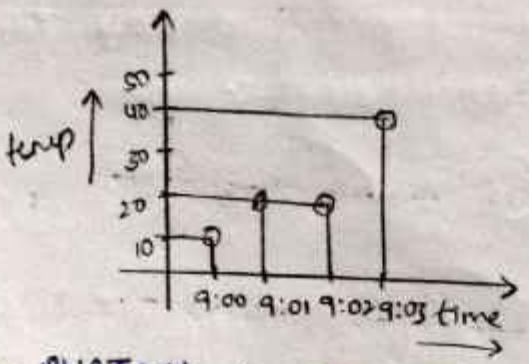
→ In Analog we have every value in the given limit.

EX :- A person is monitoring his body temperature for every minute from 9 am to 9 pm



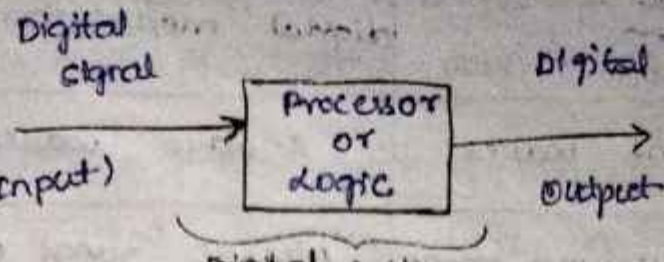
* DIGITAL SIGNAL :- In digital signals will take the values equal to the fixed levels only.

→ The values in digital signals are discrete in nature.



* DIGITAL SYSTEM :- These are designed to store Process & Communicate information in digital form.

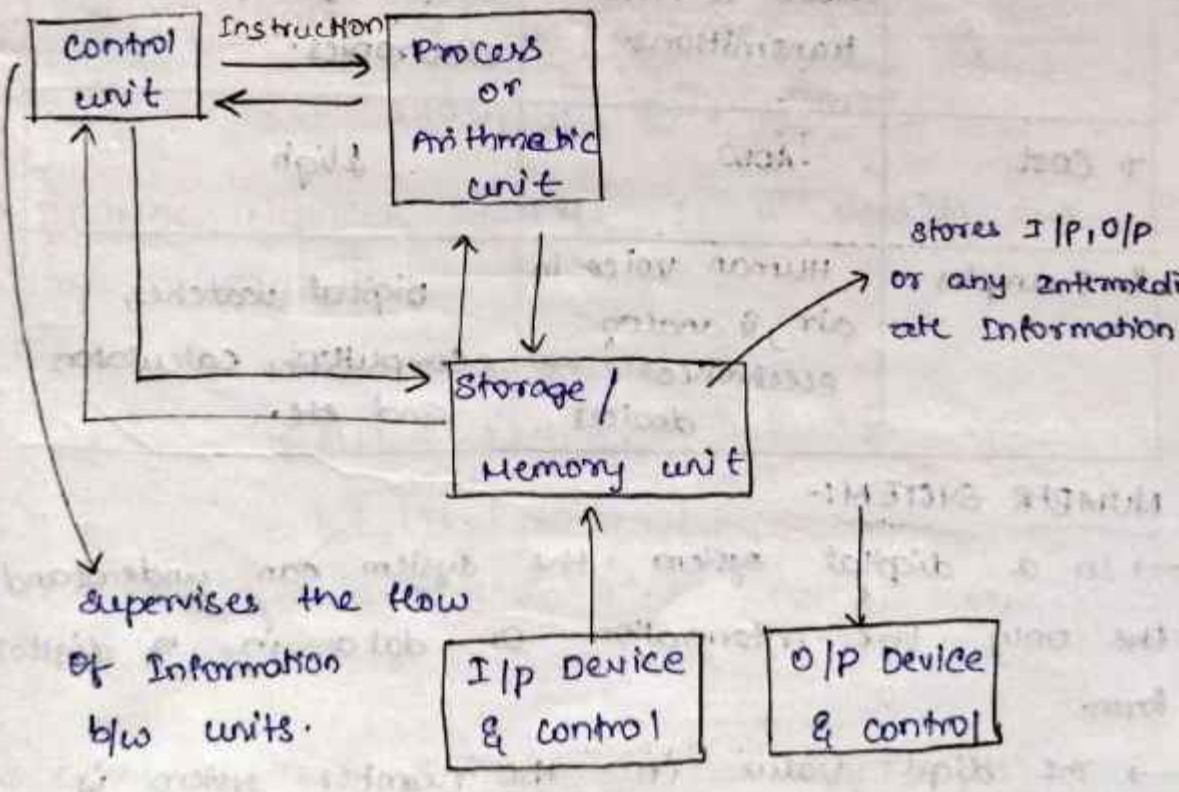
→ Applications :- control, communication systems, digital instruments & consumer products.



* CHARACTERISTICS AND BENEFITS of DIGITAL SYSTEM:-

1. Data is represented in a digital system as a vector of Binary Variables.
2. Less prone to errors than a analog systems
3. Data representation in digital system is suitable for error detection and correction.

BLOCK DIAGRAM OF DIGITAL COMPUTER.



* DIFFERENCE BETWEEN ANALOG SYSTEMS AND DIGITAL SYSTEMS:

FEATURE	ANALOG	DIGITAL
1. Signal	Analog signal represents physical measurements	Digital signals are discrete and generated by

		digital modulation.
2. Waves	sine waves	Square waves.
3. Representation	continuous range of values to represent information	Discrete range of values for information.
4. Response to noice	More likely to get effected.	less likely to get effected.
5. Bandwidth	less bandwidth	More bandwith to carry same information.
6. Uses	Best suited for audio & video transmitions.	Best suited for computing digital electronics.
7. Cost	Low	High
8. Examples	Human voice in air, & analog electronic devices	Digital watches, computers, calculators and etc.

NUMBER SYSTEM:-

→ In a digital system the system can understand the only the information or data in a digital form.

→ The digit value in the number system is Calculated by using

1. Digit
2. Base / Radix – Total No. of digits present in a number.
3. Weight – weight of a particular digit is the position of digit from decimal point

To the left of decimal point - positive weights and to the right of decimal point - negative weights.

TYPES OF NUMBER SYSTEMS :-

→ In digital computers there are various types of number systems. The main four types are:

1. Binary
2. Decimal
3. Octal
4. Hexadecimal

7/9/22

1. BINARY NUMBER SYSTEM :- In this N.S it carries only 2 values {0,1}.

→ 1 represents on / true / presence

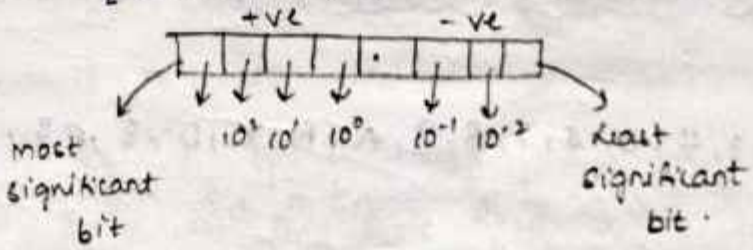
→ 0 represents off / false / absence.

Ex:- $(01101)_2$, $(1100)_2$, $(1110)_2$

2. DECIMAL NUMBER SYSTEM :- It is used in our day to day life.

It carries 10 values.

{0,1,2,3,4,5,6,7,8,9}.



3 OCTAL NUMBER SYSTEM :- It has only 8 digits.

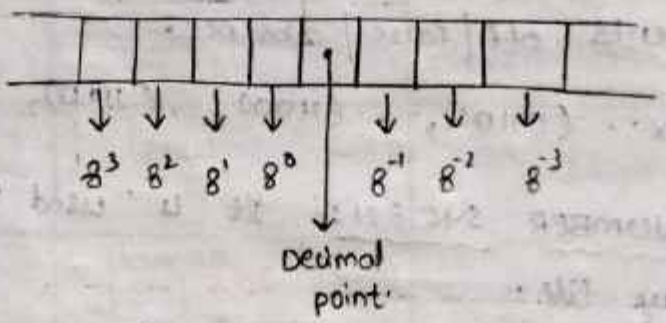
{0,1,2,3,4,5,6,7}

An octal number in binary format can be represented in 3 bits.

$2^n = 8$.

$n = 3$. (Hence 3 bits are used)

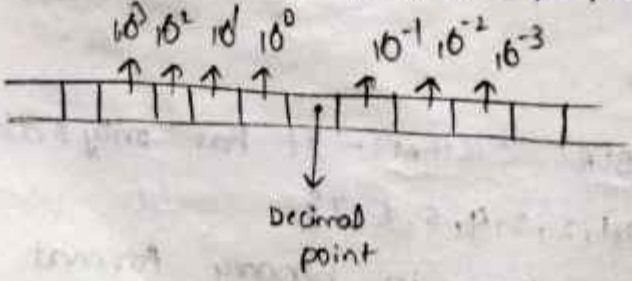
Octal	Binary
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1



Ex:- $(236)_8$, $(345)_8$:

4. HEXADECIMAL NUMBER SYSTEM :- It has only 16 digits.

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.



A Hexadecimal number in binary format can be represented in 4 bits.

$2^n = 16$ $n=4$ (Hence we use 4 bits).

Hexadecimal	Binary	Hexadecimal	Binary
0	0 0 0 0	9	1 0 0 1
1	0 0 0 1	A	1 0 1 0
2	0 0 1 0	B	1 0 1 1
3	0 0 1 1	C	1 1 0 0
4	0 1 0 0	D	1 1 0 1
5	0 1 0 1	E	1 1 1 0
6	0 1 1 0	F	1 1 1 1
7	0 1 1 1		
8	1 0 0 0		

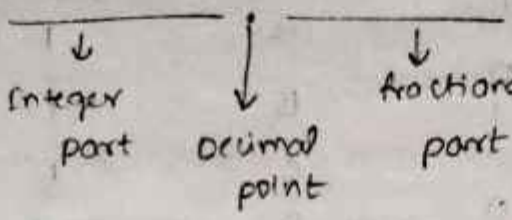
Ex:- $(DADA)_{16}$, $(1BA8)_{16}$, $(EAT)_{16}$.

Number System	Base	Start digit	End digit	Set of Digits.
Binary	2	0	1	{0,1}
Decimal	10	0	9	{0-------9}
Octal	8	0	7	{0-------7}
Hexa decimal	16	0	F	{0-------F}

*NUMBER BASE CONVERTIONS:- (4 types).

1. Decimal to Non-decimal.
2. Non-decimal to Decimal.
3. Any Base to Any Base conversion.
4. Shortcut Methods: 1. Binary to octal
   2. Octal to Binary
   3. Hexadecimal to Binary
   4. Binary to Hexadecimal.
   5. Octal to Hexadecimal.
   6. Hexadecimal to octal.

* DECIMAL TO NON-DECIMAL :-



2 Cases
- Integer part : Successive Division of Numbers with a particular Base
- Fractional part : Successive multiplication of number with a particular Base.

Decimal to Binary :-

1)

Ex: $(37)_{10} = (100101)_2$

INTEGER PART :-

2	37	
2	18	1
2	9	0
2	4	1
2	2	0
2	1	0
2	0	1

least significant Bit
top to bottom (Kaadu)

More significant Bit

2)

$(48)_{10} = (110000)_2$

2	48	
2	24	0
2	12	0
2	6	0
2	3	0
2	1	1
	0	1

* Divide the number with Base
* Divide the quotient obtained in step1 repeatedly until the coefficient is zero.
* Collect all the remainders from bottom to top. (MSB) (LSB)

Decimal to octal :-

Ex:- i) $(214)_{10} = (326)_8$

8	214
8	26 - 6
8	3 - 2
8	0 - 3

ii) $(3509)_{10} = (DB5)_{16}$

16	3509
16	219 - 5
16	13 - 11 - B
	0 - 13 - D

$(DB5)_{16}$.

→ FRACTIONAL PART :-

* successively multiply the fractional part with base to get the result.
* Take the fractional part of the result obtained from the step1 and perform multiplication until the fractional part becomes zero.
* Collect all the carrys from each product from top to bottom.

FRACTIONAL PART :-

Decimal to Binary :-

Ex:- i) $(0.8126)_{10} = (1101)_2$

Carrys.

$0.8125 \times 2 = 1.625$ — 1

$0.625 \times 2 = 1.25$ — 1

$0.25 \times 2 = 0.5$ — 0

$0.5 \times 2 = 1.0$ — 1

(1101)

ii) $(0.640625)_{10} = (101001)_2$

	Carrys
$0.640625 \times 2 = 1.28125$	1
$0.28125 \times 2 = 0.5625$	0
$0.5625 \times 2 = 1.125$	1
$0.125 \times 2 = 0.25$	0
$0.25 \times 2 = 0.5$	0
$0.5 \times 2 = 1.0$	1

$(101001)_2$

iii) $(0.640625)_{10} = (51)_8$

	Carrys
$0.640625 \times 8 = 5.125$	5
$0.125 \times 8 = 1$	1

$(51)_8$

⇒ Tutorial.

1) $(24.6)_{10} = (\ )_2$

2) $(34.5)_{10} = (\ )_2$.

1) Integer Part : $(24)_{10} = (\ )_2$

2 | 24	
2 | 12 – 0	
2 | 6 – 0	⇒ 11000
2 | 3 – 0	
2 | 1 – 1	
2 | 0 – 1	

Fractional part : $(0.6)_{10} = (\ )_2$

$0.6 \times 2 = 1.2 \quad -1$

$0.2 \times 2 = 0.4 \quad -0$

$0.4 \times 2 = 0.8 \quad -0$

$0.8 \times 2 = 1.6 \quad -1$

$\cdot 1001100/\ldots$

$0.6 \times 2 = 1.2 \quad -1$

$0.2 \times 2 = 0.4 \quad -0$

$0.4 \times 2 = 0.8 \quad -0$

$0.8 \times 2 = 1.6 \quad -1.$

$(24.6)_{10} = (11000.1001\overline{1001})_2$

B) $(34.5)_{10} = (\ )_2$

Integer part:

2 | 34	
2 | 17 - 0	↑
2 | 8 - 1	
2 | 4 - 0	
2 | 2 - 0	
2 | 1 - 0	
0 - 1	

100010.

Fractional part:

$0.5 = 1.0$

$(34.5)_{10} = (100010.1)_2$

9/9/22

* NON - DECIMAL TO DECIMAL :-

Decimal value :- Sum of each digit multiplied by their weights.

→ BINARY TO DECIMAL :-

1. Ex :- $(1101)_2 = (\ )_{10} = (13)_{10}$

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 8 + 4 + 0 + 1$

$= 12 + 1$

$= 13$

2. Ex :- $(1101.1)_2 = (\ )_{10} = (13.5)_{10}$

Integral. Positive $= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$.

$= 13.$

Q→ Negative weights (Fractional) $= 1\times2^{-1} = 0.5$

OCTAL TO DECIMAL:

$(475.26)_8 = (\ )_{10} = (317.328)_{10}$.

Integral part.

$= 8^0 \times 5 + 8 \times 7 + 8^2 \times 4$

$= 5 + 56 + 64 \times 4$

$= 61 + 64 = 61 + 256$

$= 80. = 317$

Fractional part $= 2 \times 8^{-1} + 5 \times 8^{-2}$

$= 0.25 + 0.0781$

$= 0.328$.

→ HEXADECIMAL TO DECIMAL.

$(21)_{16} = (\ )_{10}. = (33)_{10}$.

Integral part.

$= 1 \times 16^0 + 2 \times 16.$

$= 1 + 32$

$= 33$

* ANY BASE TO ANY BASE.

1. Any base to Decimal
2. Decimal to Any base.

Here we convert anybase to decimal then that decimal is converted into anybase.

This is how anybase to anybase convertion takes place.

1- Ex:- $(21)_8 = (\ )_2 = (10101)_2$

To decimal.

$= 8^0 \times 5 + 8^1 \times 2$

$= 5 + 16$

$= 21$

$(21)_{10} = (\ )_2$.

2	21
2	10 - 1
2	5 - 0
2	2 - 1
2	1 - 0
	0 - 1

10101

2. Ex:- $(10101)_2 = (\ )_8 = (25)_8$.

3. Ex:- $(356)_8 = (\ )_2 = (11101110)_2$.

To decimal

$= 3 \times 8^2 + 5 \times 8 + 6 \times 8^0$

$= 192 + 40 + 6$.

$= 238$.

$(238)_{10} = (\ )_2$.

11101110.

2	238
2	119 - 0
2	59 - 1
2	29 - 1
2	14 - 1
2	7 - 0
2	3 - 1
2	1 - 1

* SHORT-CUT METHODS:

1. BINARY TO OCTAL:

= 1. Group the binary number into 3 bits starting from least significant Bit.

2. Replace each 3 digit binary code with octal digit

1. Ex:- $(10101)_2 = (\ )_8 = (25)_8$

$\leftarrow$ MSB 10 | 101 $\rightarrow$ LSB

2 5

2. Ex :- $(010110110)_2 = (\ )_8 = (266)_8$.

$\leftarrow$ MSB 010 | 110 | 110 $\rightarrow$ LSB

2 6 6

3 Ex:- $(1100100111)_2 = (\ )_8 = (1447)_8$.

001 | 100 | 100 | 111

1 4 4 7

2. OCTAL TO BINARY:-

= 1. Replace each digit with its binary code.

1. Ex :- $(357)_8 = (\ )_2$

3 → 011, 5 → 101, 7 → 111

$= (011101111)_2$

2. Ex:- $(4576)_8 = (\ )_2$

4 → 100, 5 → 101, 7 → 111, 6 → 110

$= (100101111110)_2$.

3. Ex :- $(2547)_8 = (\quad)_2$.

7 - 111
4 - 100 $= (010101100111)_2$.
5 - 101
2 - 010

17/9/22

3. BINARY TO HEXADECIMAL :-

1. Group the given binary number into 4 digit
2. Replace the 4 digit binary code by the hexa-decimal digit.

1. Ex :- $(0101|0101)_2 = (55)_{16}$
   5 5

2. Ex :- $(1|0101|0111)_2 = (15B)_{16}$.
   1 5 B

3. Ex :- $(1101|1100 \cdot 1010|1100)_2 = (DC \cdot AC)_{16}$.
   D C A C

4. HEXADECIMAL TO BINARY :-

Replace each digit with its binary code.

1. Ex :- $(ABC)_{16} = (\quad)_2$
   A → 1010, B → 1011, C → 1100
   $= (101010111100)_2$

2. Ex :- $(BF9)_{16} = (\quad)_2$
   B → 1011, F → 1111, 9 → 1001
   $= (101111111001)_2$.

3. $(CE12)_{16} = (\quad)_2$

1100 1110 0001 0010

$= (1100111000010010)_2$.

4. $(DF4)_{16} = (\quad)_2$.

1101 1111 0100

$= (110111110100)_2$.

5. OCTAL TO HEXADECIMAL :-

1. Replace with octal digit by 3 bit binary code.
2. Group the result into 4 digits starting from LSB.
3. Replace each 4 digit binary code with hexadecimal digit.

1. Ex :- $(675)_8 = (1BD)_{16}$

110 111 101

(0001 1011 1101)

1 B D

6. HEXADECIMAL TO OCTAL :-

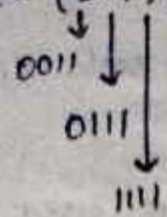
Ex :- $(25B)_{16} = (\quad)_8 = (1133)_8$.

0010 0101 1011

001 001 011 011

1 1 3 3

2. Ex $(37F)_{16} = (\quad)_8 = (1577)_8$.



0011 0111 1111

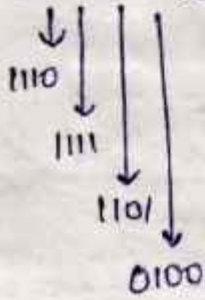
1 5 7 7

3. Ex $(48E1)_{16} = (\quad)_8 = (044341)_8$.

0100 1000 1110 0001

0 4 4 3 4 1

4. $(EFD4)_{16} = (\quad)_8 = (167724)_8$.



0001 1101 1111 0101 0100

1 6 7 7 2 4

1. Replace with Hexadecimal digit by 4 bit binary code.
2. Group the result into 3 digits starting from LSB.
3. Replace each 3 digit binary code with octal digit.

* COMPLEMENTS:-

1. For finding the subraction of a number base system the complements are used.
2. We can also say that compliments are used in digital Computers to simplify the subraction operation as well as the logical manipulation.
3. Simplifying operation leads to simpler less expensive Circuits to implement the operations.
4. There are 2 types of compliment.
   1. r's complement / Radix complement
   2. (r-1)'s complement / Diminished Radix complement.

13/9/22

→ DIMINISHED RADIX COMPLEMENT:-

Formula = $(r^n - 1) - N$

N → Given number

Let N be the number

r be the radix.

n be No. of digits in N.

Decimal system:-

1-Ex:- N = 546700 $(r^n - 1) - N$ (r-1) = 9's complement

r = 10

n = 6

$= (10^6 - 1) - (546700)$

$= 999999 - 546700 = 453299.$

2-Ex:- N = 0123

r = 10

n = 4

$= (10^4 - 1) - (0123)$

$= 9999 - 0123 = 9876.$

NOTE:- We will get the 9's complement by subtracting each digit by 9. of 'N' be from '9'.

Binary system:- (r=2)

(r-1) = 1's complement.

$(r^n-1) - N$

1. Ex:- 1011000.

n=7

r=2

$2^7 = (128)_{10} = (10000000)_2$

$2^7-1 = (127)_{10} = (1111111)_2$.

$= (2^7-1) - (1011000)$

$= (10000000 - 1)_2 - (1011000)$.

$= 9999999 - (1011000)$.

$= 8988999$.

$= (1111111) - (1011000)$

$= 0100111$.

for octal & hexa decimal system the (r-1)'s complement is obtained by subtracting each digit from 7 & 15 respectively.

2. Ex :- N = 11001

1's 1's complement = 00110.

9's complement:-

3. Ex:- N = 78321

9's complement = 21678.

→ RADIX COMPLEMENT:-

1. Decimal system:- (r=10)

10's complement = $(r)^9$

1. Ex:- 4328

9999 (5671) 9's complement +1

-4328 = 5672.

2. Ex:- 2839

9999 — 7160 + 1

2839 — = 7161 (10's complement).

7160

Binary System:- (r = 2)

	Sum	Carry
0+0 = 0		0
1+0 = 1		0
0+1 = 1		0
1+1 = 0		1
1+1+1 = 1		1

2's complement:

Ex:-
N = 1101100

1's complement → 0010011

+ 0000001

0010100

2's complement = 0010100.

2. Ex - 2:-

N = 0110111

1's complement → 1001000

2's complement → 0110111 + 0000001 = 0111000 / 1001000 − 0000001 = 1001001

2's complement = 1001001

* SUBTRACTION WITH COMPLIMENTS:-

1) r's complement subtraction.

2) (r-1)'s complement subtraction.

DECIMAL SYSTEM:-

1) r's Complement :- a) Let M, N be unsigned numbers.

10's complement subtraction | 2's complement subtraction.

Steps:-

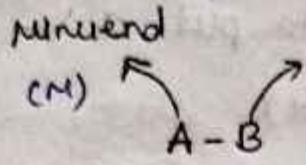
1) Add r's complement of subtrahend 'N' to minuend (M) M + (-N)

r's complement.

2) If we get a carry, discard it.

3) If you don't get a carry, take the r's complement of the result & put '-' sign before it.

4) Check whether the no of digits of both numbers are equal or not. If not equal add zero in front of the smaller numbers.



1. Ex:- 72532 - 3250

Equate both M & N (No. of digits).

72532 - 03250

10's complement of 03250 is (96749 + 1 = 96750)

99999
03250
96749

72532
(+) 96750
(1) (1)
69282 → result.

Carry

72532 - 3250 = 69282

2. Ex:- 3250 - 72532
M N

Equate No. of digits in both M & N.

03250 - 72532.

10's complement of 72532 = (27467 + 1) = (27468)

99999
72532
27467

Now add 10's complement of 72532 to 03250.

$$\begin{array}{r} 03250 \\ +27468 \\ \hline +30718 \end{array}$$

(We didnot get any extra carry here. Hence we have to put a '-' symbol)

Ans: (10's Complement of result)

(10's complement of 30718) = 69282.

$$\begin{array}{r} 99999 \\ 30718 \\ \hline 69281 \end{array}$$

3250 - 72532 = -69282

BINARY SYSTEM:-

1. Ex :- 1010100 - 1000011
M N

2's complement of ~~1010100~~ 1000011 = (1's + 1)

1's complement of ~~1010100~~ 1000011

0111100.

2's complement = $\begin{array}{r} 0111100 \\ +\quad 1 \\ \hline 0111101 \end{array}$

NOW

$$\begin{array}{r} 1010100 \\ +\ 0111101 \\ \hline 0010001 \end{array}$$

1010100 - 1000011 = 0010001.

M N

2. Ex :- 1000011 - 1010100.

2's complement of 1010100 = (1's +1)

1's complement of 1010100 = 0101011.

2's complement = $\begin{array}{r} 0101011 \\ +\quad 1 \\ \hline \end{array}$

$$\begin{array}{r} 0101011 \\ 000+111 \\ \hline 0101100 \end{array}$$

1's −

2's complement · is 0101100.

$$1000011 + 0101100 = -0010001.$$

2) (r-1)'s complement :-

9's complement / 1's complement.

1. Let M, N be two unsigned numbers.
2. Add (r-1)'s complement of subtrahend 'N' to minuend 'M'. i.e. M+(-N)

(-N) complement.

3. If we get a carry, add it to the result.
4. If we don't get a carry, take (r-1)'s complement of the result & put '-' sign before it.

DECIMAL NUMBER SYSTEM:-

Ex:-

1. 72532 − 03250
   M N

9's complement ⇒ 99999
− 03250
96749

72532
+ 96749
169281
+1
69282

Ans:- 72532 − 03250 = 69282.

2. 03250 − 72532.
M N

9's complement => 99999
72532
27467

27467
03250
30717

99999
30717
− 69282

since, no carry.

Ans:- 03250 − 72532 = − 69282.

BINARY NUMBER SYSTEM:-

1. M = 1010100
N = 1000011

2. M = 1000011
N = 1010100.

Ex:-
M = 1010100.
N = 1000011

1's complement of N is 0111100.

1010100
+ 0111100
0010000

As:- 1010100 − 1000011 = 0010000. 0010001.

M − N = 0010001.

ii) $M = 1000011$ $N = 1010100$.

1's complement of $N = 0101011$.

$1000011 + 0101011 \Rightarrow 1101110$.

$1000011 - 1010100 = -0010001$

$\therefore M-N = -0010001$.

→ Find 'x'.

i) $(193)_x = (623)_8$

$x^2\ x^1\ x^0$ $\quad 8^2\ 8^1\ 8^0$

$1 \times x^2 + 9 \times x + 3 \times x^0 = 8^0 \times 3 + 8^1 \times 2 + 8^2 \times 6$.

$x^2 + 9x + 3 = 3 + 16 + 384$

$x^2 + 9x = 400$

$x^2 + 9x - 400 = 0$.

$x = 16, (-25)$.

$\therefore$ Radix is 16.

19/9/22

SIGNED BINARY NUMBERS:-

1. positive integers including 0 can be represented as unsigned numbers.
2. However to represent a negative integer we need a notation for negative value.
3. In ordinary arithematic a positive number is indicated with '+' sign and negative number is indicated with '-' sign.
4. Because of hardware limitations computers must represent everything with binary number
5. For this we have mainly 2 methods.
   i) signed magnitude representation.
   ii) signed complement representation.

1. SIGNED MAGNITUDE REPRESENTATION:-

1. It is generally in 8 bit format or representation.
2. The most significant bit is called signed bit.
3. If the signed bit is zero we call it as positive number, if it is one we call it as negative number.

Ex:-

+9: 0 0001001
↓ (MSB / sign bit) — 9

-9: 1 0001001
↓ (sign)

Drawback:-

Eg:- +0 0 000
-0 1 000

2. SIGNED COMPLEMENT REPRESENTATION:-

1's complement, 2's complement

+9: => 00001001

1's complement => 11110110

Drawback:- +0 => 0000 } But we have only one '0'.
-0 => 1111 }

2's complement +9 => 00001001
(-9) => 11110111

2's complement +0 => 0000 ✓
-0 => 0000

Addition:- ex:-

1) +6 => 0 0 0 0 0 1 1 0
+13 => 0 0 0 0 1 1 0 1
+19 +19 => 0 0 0 1 0 0 1 1

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$= 16+2+1$

$= 19.$

2) +6 => 0 0 0 0 0 1 1 0
-13 => 1 1 1 1 0 0 1 1
-7 (-7) 1 1 1 1 1 0 0 1

(-)

2's comp
00000110
+1
00000111
(+7)

13 => 00001101
2's comp => 11110010
+1
11110011

3) -6 => 00001101 (13)
+13

-6 => 1 1 1 1 1 0 0 1
+13 => 0 0 0 0 1 1 0 1
+(7) 00000111

-6 => 1 1 1 1 1 0 1 0
+13 => 0 0 0 0 1 1 0 1
00000111

bin (6) = 00000110 = 11111010
1's = 11111001
2's = 00000111
11111010

4) -6 => 11111010 (2's comp cuz '-' symbol).
-13 => 11110011 (2's comp cuz '-' symbol).
-19. 11101101

(-ve)

2's comp:
00010011
1's = 00010010
2's = 00000111
00010011

$2^3\ 2^2$
$2^4\ 2^1\ 2^0$

$= 16+0+0+2+1$
$= 19.$

* BINARY CODES:- (20/9/22)

1. The group of symbols is called as code.
2. The digital data is represented / stored and transmitted as a group of binary bits. This group is called Binary code.

n bits → 2^n distinct values

1 → 2^1 → (0,1)

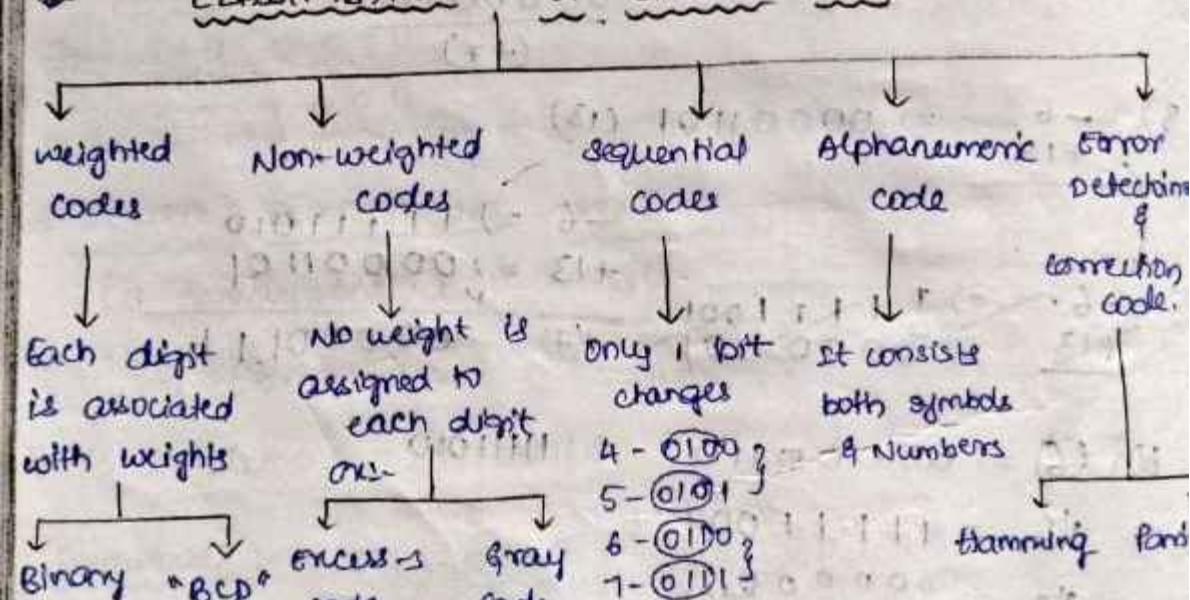
2 → 2^2 → (00, 01, 10, 11).

3 → 2^3 → (000, 001, 010, 011, ... 111).

ADVANTAGES OF BINARY CODE:-

1. Binary codes are suitable for computer applications and digital communication.
2. Since only 0's and 1's are being used implementation becomes easy.

* CLASSIFICATION OF BINARY CODE.



→ BINARY CODED DECIMAL : - (BCD)

* Each decimal digit in 4 bits.

n=4 → 2^4 = 16 (distinct values)

→ But we take only first 10 digits / numbers i.e 0 to 9 as we are considering decimal system. remaining 10 to 16 are invalid.

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

2 types of Conversions:-

1. Binary to BCD.
2. BCD to binary.

1. Binary to BCD:-

* convert binary to decimal and then convert decimal to BCD.

1. Ex:- $(11101)_2 = (\quad)_{BCD}$

$(11101)_2 = (29)_{10}$

$= 16+8+4+0+1$

$= 29.$

$(29)_{10} = (\quad)_{BCD}$

$= (00101001)_{BCD}$

2. BCD to Binary:-

* Convert BCD to decimal and then convert BCD to (decimal)x Binary.

1. Ex:- $(00101001)_{BCD} = (\quad)_2.$

$(0010\,1001)_{BCD} = (29)_{10}$

$(29)_{10} = (\quad)_2.$

$2 \mid 29$
$2 \mid 14-1$
$2 \mid 7-0$
$2 \mid 3-1$
$1-1$

$= (11101)_2.$

$\therefore (00101001)_{BCD} = (11101)_2.$

BCD Code:- (21|9|22)

8421 7421 5421 84-2-1

Decimal	8421	7421	84-2-1	5421
0	0000	0000	0000	
1	0001	0001	0111	
2	0010	0010	0110	
3	0011	0011	0101	
4	0100	0100	0100	
5	0101	0101	1011	
6	0110	0110	1010	
7	0111	0111	1001	
8	1000	1001	1000	
9	1001	1010	1111	

BCD Addition:-

1. Add the BCD numbers same like Binary numbers
2. If the 4-bit result is less than or equal to 9, then it is valid BCD code.
3. If it is
   - greater than 9
   - If a carry is generated

   } Invalid code.
4. We have to add $(6)_{10} = (0110)_2$ to invalid code to make valid code.
5. If a carry is generated, add it to the higher BCD code.

1. Ex:- $6 + 3 = 9$

$6 \Rightarrow 0110$

$3 \Rightarrow 0011$

$1001 \rightarrow$ valid code.

2. 8+6 = 14

8 => 1000

6 => 0110

1110 → Invalid code

0110 → Add (6)

0100 (4)

3. 24+18 = 42

24 => 0010 0100

18 => 0001 1000

0011 1100 → Invalid code

0110 → Add (6).

0011 0010

+1111 2 => 42

0100

4

4. 175+326 = 501

175 => 0001 0111 0101

326 => 0011 0010 0110

0100 1001 1011

1 0011 +0110 => 501

→ 1010 (1)

Invalid 0110 0001

(1) 0101 0000 1

5 0

* BCD Subtraction:-

1. Take the 9's complement of subtrahend and add it to minuend i.e, A+(-B)
2. If the result is less than or equal to 9 then it is valid code.
3. If it is Invalid, then add $(6)_{10} = (0110)_2$.
4. If you get a carry, add it to Higher BCD code.

5. If a end around carry is generated, add it to the result of first BCD code.

1) (206 - 147) using 9's complement subtract 147 from 206.

$$206 - 147 = 059$$

M N

9's comp (147)

999
147
=) 852

206 =) 0010 0000 0110

852 =) 1000 0101 0010

1 1

→ 1010 0101 1000

Invalid

0110

5

0000

0 Add 1000

1001

9.

Excess 3 code :-

1. It is an unweighted code.
2. We get this code by adding $(3)_{10}$ / $(0011)_2$ to BCD code.

Decimal → BCD —Add(3)→ Excess-3 code.

Decimal	BCD	Excess -3
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

3. It is called as "self-complementing code".

Excess-3 code: conversions (2 types)

1. BCD to Excess-3
2. Excess-3 to BCD.

BCD to Excess-3

$(1001)_{BCD} = (\quad)_{XS+3} = (1100)_{XS-3}$

9
3
12.

$= 1001$
0011
1100
$2^3\ 2^2\ 2^1\ 2^0$
$8+4+0+0$
$= 12.$

Excess-3 to BCD

$(1001)_{XS-3} = (0110)_{BCD}$

9
-3
6 → 0110

$= 1001$
$(+)\ 1101$
$\overset{\text{carry}}{1}\ 0110$
$2^2\ 2^1\ 0$
$= 4+2 = 6.$

Ex 1:-

BCD to excess-3

$(0010\ 1000)_{BCD}$ to excess-3 $= (0101\ 1011)_{XS-3}$

0010 1000
0011 0011 (add 3).
0101 1011
↓ ↓
6 11

2. Excess-3 to BCD

$(1001)_{XS-3} = (\quad)_{BCD} = 0110.$

$= 9-3 = 6 = 0110.$

* GRAY CODE :-

It is an unweighted code. It is also called as Reflected Binary Code (RBC) (or) unit distance code (or) ~~minimum~~ error code minimum error code.

2. It is called a Gray code because it was introduced by Frank gray.

3. Gray codes are widely used for error corrections in digital communication systems like cable TV systems.

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

Gray code conversion:-

1. Bamy to Gray code.
2. Gray code to Binary.

1. Binary to Gray code:-

Binary - 0 1 0 0 0 1

Gray code - 0 1 1 0 0 1

→ Exclusive or operation.

Binary - 1 1 0 1 - 13

Gray - 1 0 1 1 - 13.

Binary - 1 1 1 0 - 14

Gray - 1 0 0 1 - 14.

1. The MSB of given binary code is the MSB of required gray code.
2. The second bit of gray code will be the result of exclusive-or operation of 1st and 2nd bits of given binary code.
3. The third bit is the result of X-OR operation of 2nd & 3rd Bits of binary code.
4. The process continues till the end.

2. Gray code to Binary code :-

Gray - 01001

Binary - 01110

1. The MSB of given gray code is the MSB of required Binary code.
2. The second bit of binary code is the result of exclusive-or operation of 1st bit of binary and second bit of given gray code.
3. The process continues till the end.

26/9/22.

* BINARY LOGIC :- It is the basic element of electronic systems such as computers and cells phones

Input → Logic → Output

↓

It performs some logical operations.

2. It takes binary inputs perform logic function and produces the output.
3. By these connecting these gates in different forms we design a circuit.
4. Logic operation of any gate is given by the the truth table.
5. The truth table consists of all possible combination of given inputs to the corresponding outputs.

→ Types of Logic Gates:-

Mainly there are 3 types of logic gates.

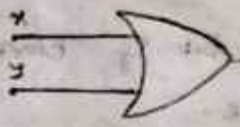
1. Basic gates - AND, OR, NOT

2. Universal gates - NAND (AND+NOT), NOR (OR+NOT)

3. Special gates - EX-OR, EX-NOR.

S.NO	Name	Symbol	Function	Truth Table
1.	AND		$F=x \cdot y$	

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

2. OR  $F=x+y$

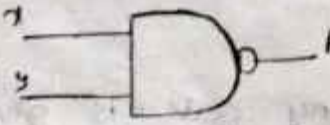
x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT/ Inverter  $F=x'$

x	$F=x'$
0	1
1	0

4. Buffer  $F=x$

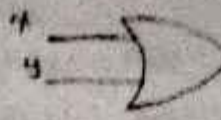
x	F=x
0	0
1	1

5. NAND (AND+NOT)  $F=(x \cdot y)'$

x	y	$(x \cdot y)'$
0	0	1
0	1	1
1	0	1
1	1	0

S.NO	Name	Symbol	Function	Truth Table

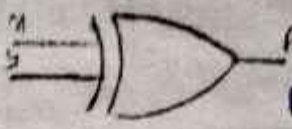
6. NOR (OR + NOT)



$F = (x+y)'$

x	y	$(x+y)'$
0	0	1
0	1	0
1	0	0
1	1	0

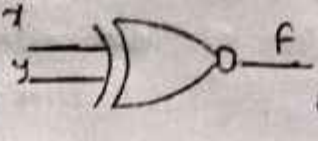
7. Exclusive-OR



$F = x'y + xy'$

x	y	$x'y + xy'$
0	0	0
0	1	1
1	0	1
1	1	0

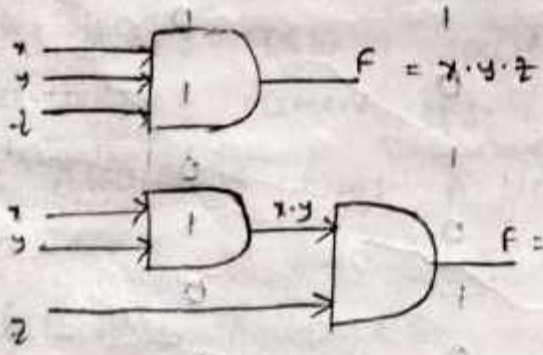
8. Exclusive - NOR



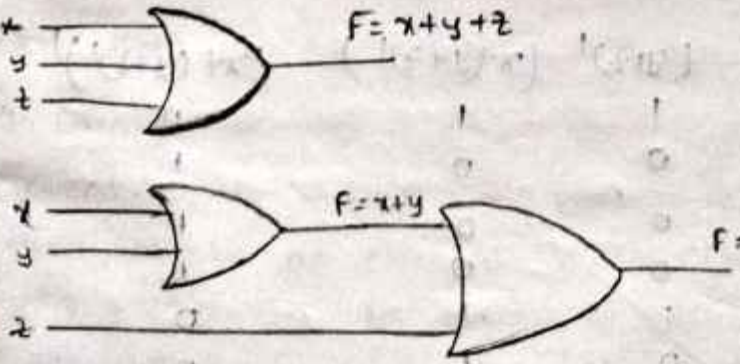
$F = xy + x'y'$

x	y	$xy + x'y'$
0	0	1
0	1	0
1	0	0
1	1	1

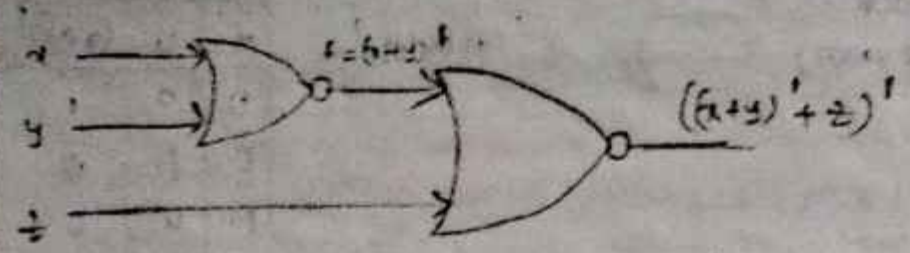
* $F = x \cdot y \cdot z$



* $F = x + y + z$



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



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

Truth Table:-

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

x	y	z	(x+y)	(x+y)'	(x+y)'+z	((x+y)'+z)'
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	1	0	1	0	0	1
0	1	1	1	0	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

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

x	y	z	y+z	(y+z)'	(x+(y+z)')	(x+(y+z)')'
0	0	0	0	1	1	0
0	0	1	1	0	0	1
0	1	0	1	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	1	1	0	1	0
1	1	0	1	0	1	0
1	1	1	1	0	1	0

27/9/22 2. BOOLEAN ALGEBRA AND LOGIC GATES.

→ BASIC DEFINITIONS:-

* mathematical methods that simplify binary logics or circuits rely primarily on Boolean algebra.

* BOOLEAN ALGEBRA :- A set of unproved axioms or operators, and a number of unproved axioms or postulates.

* A set of elements in any collection of objects, usually having a common property; A = {1,2,3,4} indicates that set A has the elements of 1, 2, 3, and 4.

* A binary operator defined on a set of elements is a rule that assigns, to each pair of elements from S, a unique element from S.

* The most common postulates used to formulate various algebraic structures are as follows:

1. Closure: A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule of obtaining a unique element of S.

2. Associative law:- A binary operator (+, -, *) on a set S is said to be associative $(x*y)*z = x*(y*z)$ for all $x, y, z \in S$.

3. Commutative law:- A binary operator (*, +, -) on a set S is said to be commutative $x*y = y*x$ for all $x, y \in S$.

4. Identity element:- A set S is said to have identity element with respect to binary operator * on S if there exists an element $e \in S$ with the property that $e*x = x*e = x$ for every $x \in S$.

Ex:- The element 0 is an identity element with respect to the binary operator + on the set of integers

$I = \{ e, -3, 2, -1, 0, 1, 2, 3, c \}$ since $x + 0 = 0 + x = x$ for any $x \in I$.

The set of natural numbers, N, has no identity element, since 0 is excluded from the set.

5. Inverse :- In A set S having the identity element e with respect to binary operator * is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that $x * y = e$.

Ex:- In the set of Integers, I and the operator +, with $e = 0$, the integer of an element a is $(-a)$, since $a + (-a) = 0$.

6. Distributive law :- If * and $\pm$ are two binary operators on a set S*, is said to be distributive over $\pm$ whenever $x * (y \pm z) = (x * y) \pm (x * z)$

* AXIOMATIC DEFINITION OF BOOLEAN ALGEBRA :

→ 1854 - George Boole developed an algebraic system now called Boolean algebra.

→ 1904 - E.V. Huntington formulated a set of postulates that formally define the Boolean algebra.

→ 1938 - C.E. Shannon introduced a two-valued Boolean algebra, called switching algebra that represents the properties of bistable electrical switching circuits

→ two binary operators, + and *, (Huntington) postulates

1. a) The structure is closed w.r.t the operator +
   b) The structure is closed w.r.t the operator *
2. a) The element 0 is an identity element w.r.t +, that is $x + 0 = 0 + x = x$.
   b) The element 1 is an identity element w.r.t *, that is $x * 1 = 1 * x = x$.
3. a) The structure is commutative w.r.t to +, that is $x + y = y + x$.

b) The structure is commutative w.r.t * & +, that is

$$x * y = y * x.$$

4a) The operator * is distributive over +, that is,

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

b) The operator + is distributive over *, that is

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

5. For every element $x \in B$, there exists an element $x' \in B$ (called the complement of x) such that

(a) $x + x' = 1$ and (b) $x * x' = 0$.

6. There exist at least two elements $x, y \in B$ such that $x \neq y$.

COMPARING BOOLEAN ALGEBRA WITH ARITHMETIC AND ORDINARY ALGEBRA.

1. Huntington postulates do not include the associative law. However, this law holds for Boolean algebra and can be derived (for both operators) from the other postulates.

2. The distributive law of + over * (i.e. $x + (y * z) = (x+y) * (x+z)$) is valid for boolean algebra, but not for ordinary algebra.

3. Boolean algebra does not have additive or multiplicative inverses; i.e. there are no subtraction or division operations.

4. Postulate 5 defines an operator called the complement that is not available in ordinary algebra.

Two-valued Boolean Algebra.

- $B = \{0, 1\}$
- The rules of operation.

AND

x	y	x*y
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	x'
0	1
1	0

* Closure, the result of each operation is either 1 or 0 and 1,0 ∈ B.

* Identity elements 0 for + and 1 for .*.

* The commutative laws are obvious from the symmetry of the binary operation tables.

RULES OF BOOLEAN ALGEBRA.

1. The element in Boolean Algebra called as "variable"
2. Logical AND is represented by dot (multiplication)
3. Logical OR is represented by plus (addition).
4. The compliment is represented by (x').
5. It takes only 2 values either '0' or '1'.

Logical AND → 4 rules, Logical OR → 4 rules, NOT → 4 rules

RULES OF BOOLEAN ALGEBRA:-

* LOGICAL OR:- (+)

Rule 1:- $A+0 = A$

$0+A = A$.

Proof:- $A+0 = A$

$A=0$, $0+0=0$

$A = 1$, $1+0=1$.

Rule 2:- $A+1 = 1$

$1+A = 1$

Proof:- $A+1 = A1$

$A = 0$, $0+1 = 1$

$A = 1$, $1+1 = 1$

Rule 3:- $A+A = A$

Proof:- $A+A = A$

$A=0$, $0+0=0$

$A=1$, $1+1=1$

Rule 4:- $A+A' = 1$ $A'+A = 1$

Proof :- $A = 0, 1+0 = 1$

$A = 1, 0+1 = 1$

LOGICAL AND:- (·)

Rule 5 :- $0 \cdot A = 0$

$A \cdot 0 = 0$.

Proof :- $0A = 0$.

$A = 0, 0 \cdot 0 = 0$.

$A = 1, 0 \cdot 1 = 0$.

Rule 6 :- $1 \cdot A = A$

$A \cdot 1 = A$

Proof :- $A \cdot 1 = A$

$A = 0, 0 \cdot 1 = 0$.

$A = 1, 1 \cdot 1 = 1$.

Rule 7:- $A \cdot A = A$

Proof :- $A \cdot A = A$

$A = 0, 0 \cdot 0 = 0$

$A = 1, 1 \cdot 1 = 1$.

Rule 8 :- $A \cdot A' = 0$

$A' \cdot A = 0$.

Proof :- $A' \cdot A = 0$.

$A = 1$ (x) $0 \cdot 1 = 0$.

$A = 0$, (0) $1 \cdot 0 = 0$.

NOT :- (')

Rule 9:- $(A')' = A$

Proof :- $(A')' = A$.

$A = \underline{0} \Rightarrow (0')' \Rightarrow (1)' = \underline{0}$.

$A = \underline{1} \Rightarrow (1')' \Rightarrow (0)' = \underline{1}$

Rule 10 :- $A + AB = A$.

Proof :- $A + AB = A$

$\Rightarrow A(1+B) = A$ [Rule -2 $1+A = 1$]

$A(1) = A$.

Rule 11:- $A+A'B = A+B$.

Proof:- $A+A'B = A+B$

$A+AB+A'B = A+B$ (from rule 10 $A = A+AB$)

$A+B(A+A') = A+B$. (from rule 4) $(A+A' = 1)$

$\therefore A+B(1) = A+B$.

$A+B = A+B$

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

Proof:- $A \cdot A + AC + AB + BC$.

$= A + AC + AB + BC$. (from rule 7).

$= A(1+C) + AB + BC$ $A \cdot A = A$ (from rule 2)

$= A + AB + BC$ $1+A = 1$

$= A(1+B) + BC$ (from rule 2 $1+B = 1$).

$= A + BC$

* LAWS OF BOOLEAN ALGEBRA.

1. COMMUTATIVE LAW:

Law 1:- $A+B = B+A$ (+)

Law 2:- $A \cdot B = B \cdot A$. (·)

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

$\therefore A+B = B+A$ and $A \cdot B = B \cdot A$.

2. ASSOCIATIVE LAW:-

Law 1:- (+) $\Rightarrow (A+B)+C = A+(B+C)$

Law 2:- (·) $\Rightarrow (A \cdot B) \cdot C = A \cdot (B \cdot C)$

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

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

A	B	C	A·B	(A·B)·C	(B·C)	A·(B·C)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

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

3. DISTRIBUTIVE LAW :-

Law (1) :- (·) = $A \cdot (B+C) = A \cdot B + A \cdot C$

Law (2) :- (+) = $A+(B \cdot C) = (A+B) \cdot (A+C)$

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

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

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

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

4. DEMORGANS THEOREM:-

Law 1:- $(A+B)' = A'\cdot B'$ (+)

$(A\cdot B)' = A'+B'$ (·)

A	B	A+B	(A+B)'	A'	B'	A'·B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

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

A	B	A·B	(A·B)'	A'	B'	A'+B'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

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

OPERATOR PRECIDENCE:-

1. Paranthesis
2. Complument / NOT
3. Logical AND (·)
4. Logical OR (+).

10/10/22

* BOOLEAN FUNCTIONS:-

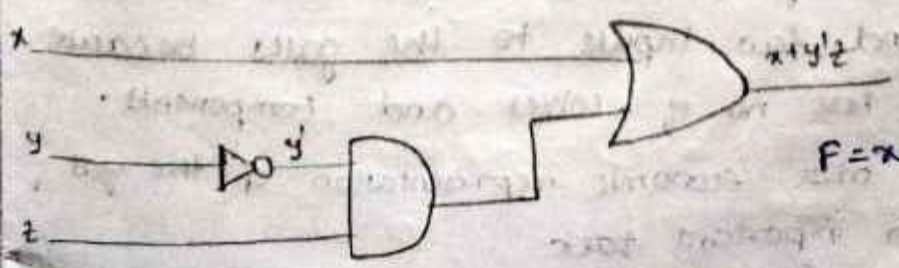
1. Boolean algebra is an algebra that deals with binary variables and logic operations (AND, OR, NOR, XOR, XNOR)
2. A boolean function can be described by an algebraic expression consists of binary variables, the constants (0, 1) and the logic operation symbols.
3. A boolean function can be represented using truth table and as well as circuit diagram.

Ex:- Boolean function $F = x + y'z$

Truth Table:-

x	y	z	y'	y'z	x + y'z
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

Gate Implementation:-



* Consensus Theorem:-

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

$AB + A'C + BC(1) \rightarrow \{(A + A') = 1\}$.

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

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

$= AB(1 + C) + A'C(1 + B)$

$= AB + A'C$.

2 Ex:- Boolean function $F = x'y'z' + x'yz + xy'$

$\Rightarrow F = x'z(y'+y) + xy'$

$\Rightarrow F = x'z(1) + xy'$ (∵ Rule 4)

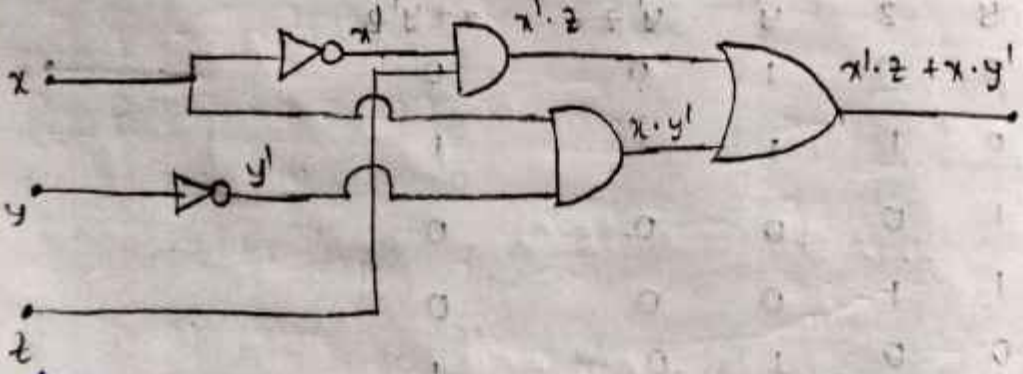
$\Rightarrow F = x'z + xy'$

Truth Table:-

x	y	z	x'	y'	x'z	x·y'	x'z + x·y'
0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	1
0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

Circuit Diagram:-

$F = x'z + xy'$



* since both expressions produces the same truth table means that they are equivalent.

* Each circuit implements the same identical function but it is always preffered to consider the one with few gates and few inputs to the gates because it requires less no of wires and components.

* Finding the most economic representation of the gate logic is an important task

Dual of a Boolean Function:-

The dual of a Boolean function can be obtained by representing AND with OR and OR with AND, of the given boolean function.

Example: $F = x + y'z$

Dual of function, $x + y'z \Rightarrow x \cdot (y' + z)$

Complement Of Boolean Function:-

The complement of Boolean function can be obtained by using Demorgan's theorem.

The following steps are the to obtain complement:-

1. Dual the given boolean function.
2. Complement each literal.

Example:- $F = x + y'z$

Dual of $F \Rightarrow x \cdot (y' + z)$

Complement each literal $\Rightarrow x' \cdot (y + z')$

11/10/22. Simplification:-

1. $xy + xy'$

$x(y + y') \Rightarrow$ (Rule - 4:- $A + A' = 1$)

$x(1) = x$

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

$x \cdot x + x \cdot y' + y \cdot x + y \cdot y' \Rightarrow$ ($A \cdot A' = 0$ Rule :- 8)

$\Rightarrow x + x \cdot y' + xy + 0 \Rightarrow$ ($A \cdot A = A$ Rule :- 7)

$\Rightarrow x + x(y + y') \Rightarrow$ ($A + A' = 1$ Rule :- 4)

$\Rightarrow x + x$

$= x$

3. $xyz + x'y + xyz'$

$xy(z + z') + x'y \Rightarrow$ ($A + A' = 1$ Rule :- 4)

$xy(1) + x'y$

$xy + x'y$

$y(x + x')$

$y(1)$

$= y$

4. $(A+B)' \quad (A'+B')'$

$(A' \cdot B') \quad (A')' \cdot (B')'$

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

$A \cdot A' \cdot B \cdot B'$

$= 0$

5. $ABC + A'B + ABC'$

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

$AB + A'B$

$B(A + A')$

$= B$

6. $xy + x(wz + wz')$

$xy + xwz + xwz'$

$xy + xw(z + z')$

$xy + xw$

$x(y + w)$

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

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

$x'y'x' + x'y'y'$

$x'y' + x'y'$

$= x'y'$ $\{A + A = A\}$

Rule :-3.

8. $(BC' + A'D)(AB' + CD')$

$ABB'C' + BCC'D' + AB'A'D + A'CDD'$

$= 0 + 0 + 0 + 0$

$= 0$ $\{A \cdot A' = 0$ Rule :-8$\}$

9. $F = AB + A'C + AB'$

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

$= A + A'C \quad \{ A + A'B = A + B \text{ Rule 1-11} \}$

$= A + C$

10. $(x+y)(x+y') + (xy' + x')'$

$\Rightarrow x \cdot x + xy' + xy + yy' + (xy' + x')'$

$\Rightarrow x + (x(y+y')) + 0 + (xy' + x')'$

$\Rightarrow x + x(1) + (xy' + x)'$

$\Rightarrow x + (xy + x')' \Rightarrow \{ (AB)' = A' + B' \}$

$\Rightarrow x + (x' + y)$

$\Rightarrow x + x \cdot x' + xy$

$\Rightarrow x + xy$

$\Rightarrow x[1 + y]$

$\Rightarrow x(1)$

$\Rightarrow x.$

* Canonical & Standard forms:.

MINTERMS:-

1. Binary variable contains two forms:

   i) Normal form (x)

   ii) Complement form (x')

2. Two binary variable x and y combined with an "and" operation or will have 4 combinations

   $x \cdot y, \; x' \cdot y, \; x \cdot y', \; x' \cdot y'$

3. Each of these four "and" terms are called minterms or standard products.

4. Each minterm is obtained from an "and" term of n variables with each variable being primed if the corresponding bit is (0) and unprimed if the

if the binary bit is (1).

5. Minterms can be represented as m_j where j denotes the decimal equivalent.

MAXTERMS:-

1. In a similar fashion n variables forming an 'OR' term with each variable being primed or unprimed.
2. If we take 2 variables x, y

$x+y$, $x'+y$, $x+y'$, $x'+y'$.

3. Each maxterm is obtained from an 'OR' term of n variables with each variable being primed if the bit is (1) and unprimed if the bit is (0).
4. Maxterm can be represented as M_j where j denotes the decimal equivalent.

Minterms & Maxterms for 3 variables:- (x, y, z)

x	y	z	Minterm		Maxterm	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	$M_0 \leftarrow 0$
0	0	1	$x'y'z$	m_1	$x+y+z'$	$M_1 \leftarrow 1$
0	1	0	$x'yz'$	m_2	$x+y'+z$	$M_2 \leftarrow 2$
0	1	1	$x'yz$	m_3	$x+y'+z'$	$M_3 \leftarrow 3$
1	0	0	$xy'z'$	m_4	$x'+y+z$	$M_4 \leftarrow 4$
1	0	1	$xy'z$	m_5	$x'+y+z'$	$M_5 \leftarrow 5$
1	1	0	xyz'	m_6	$x'+y'+z$	$M_6 \leftarrow 6$
1	1	1	xyz	m_7	$x'+y'+z'$	$M_7 \leftarrow 1$

Sum of minterms / Sum of products

product of maxterms. / product of sums

=) $F = x + y'z$ Example:-

	x	y	z	y'	$y'z$	$x+y'z$	
m_0	0	0	0	1	0	0	$x+y+z$
m_1	0	0	1	1	1	1 ·	$x'y'z$
m_2	0	1	0	0	0	0	$x+y'+z$
m_3	0	1	1	0	0	0	$x+y'+z'$
m_4	1	0	0	1	0	1 ·	$xy'z'$
m_5	1	0	1	1	1	1 ·	$xy'z$
m_6	1	1	0	0	0	1 ·	xyz'
m_7	1	1	1	0	0	1 ·	xyz

Sum of minterms:- we can obtain sum of minterms by taking each combination that "produces1" from the truth table & then ORing those terms.

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

$$F = m_1 + m_4 + m_5 + m_6 + m_7.$$

$$F = \Sigma(1,4,5,6,7)$$

Product Sum of maxterms :- we can obtain product of maxterms by taking each combination that "produces 0" from the truth table & then ANDing those terms.

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

$$F = M_0 \cdot M_2 \cdot M_3$$

$$F = \Pi(0,2,3).$$

14|10|22

ANOTHER METHOD:-

1. Sum of products :-

1. Check whether the expression is in "sum of AND" terms
2. Check whether all the variables are present in each term

3. If any variable miss, then the term is ANDed with $(x+x')$, where 'x' is a missing variable.

Ex:- $F = x + y'z$

$x \cdot 1$ $\quad y'z$

y, z are missing — x is missing

$x + y'z$

$\Rightarrow x(y+y')(z+z') + (x+x')y'z$

$\Rightarrow x(yz + yz' + y'z + y'z') + xy'z + x'y'z$

$\Rightarrow xyz + xyz' + xy'z + xy'z' + xy'z + x'y'z$

$\Rightarrow xyz + xyz' + xy'z + xy'z' + x'y'z$

$\Rightarrow m_7 + m_6 + m_5 + m_4 + m_1$

$= \Sigma(7,6,5,4,1) \quad [\Pi(0,2,3)]$

2. Product of sum's:-

1. check whether the expression contain product of OR terms.
2. check whether all variables are present in each term.
3. If an variable is missing, then the term is ORed with $(x \cdot x')$, where x' is a missing variable.

Ex:- $F = xy + x'z$

B C A

$A + BC = AB + BC \quad (A+B)\cdot(A+C)$

$xy + x'z$

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

C B A

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

$\Rightarrow (x + x') \neq 1$

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

$\Rightarrow \big((x+z)+(y\cdot y')\big)\cdot\big((y+x')+z\cdot z'\big)\cdot\big((y+z)+x\cdot x'\big)$

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

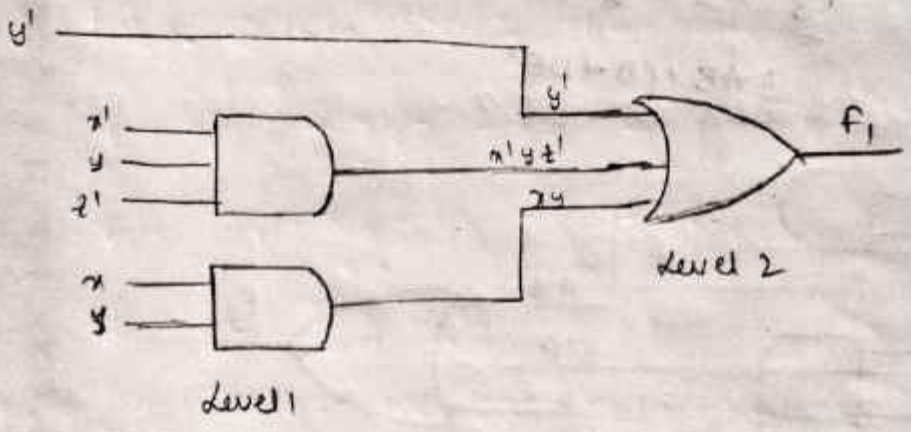
$\Rightarrow (x+y+z)\cdot(x+z+y')\cdot(y+x'+z)\cdot(y+x'+z')$

$\Rightarrow M_0\cdot M_2\cdot M_4\cdot M_5$

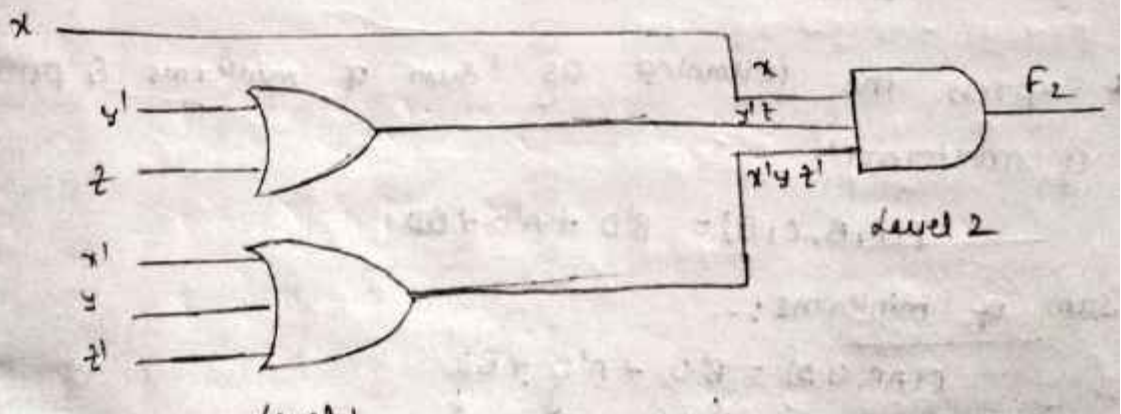
$= \Pi(0,2,4,5) \quad [\Sigma(1,3,6,7)]$

Standard Form :-

$F_1 = xy + y' + x'yz'$

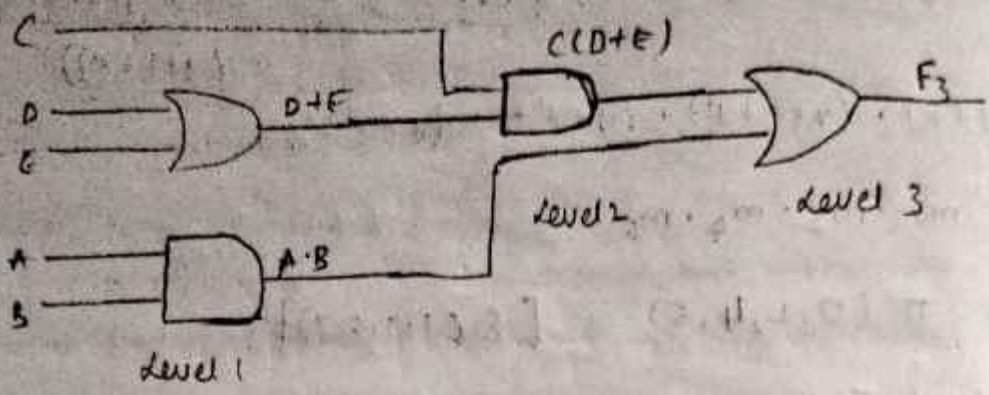


$F_2 = x(y'+z)(x'+y+z')$



Draw logic diagram in 3 level format

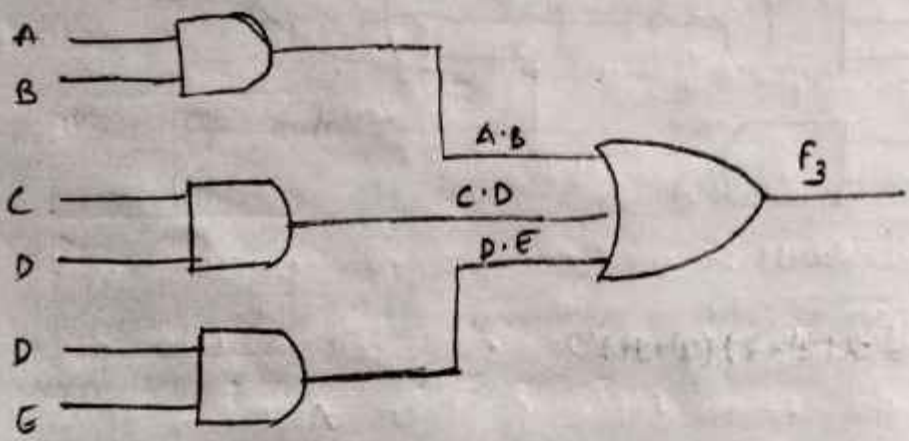
$$F_3 = AB + C(D+E)'$$



Draw logic diagram in 2 level format

$$F_3 = AB + C(D+E)'$$

$$= AB + CD + DE'$$



* Express the following as 'sum of minterms & product of maxterms'.

$$F(A,B,C,D) = B'D + A'D + BD$$

Sum of minterms :-

$$F(A,B,C,D) = B'D + A'D + BD$$

(A,C) (B,C) (AC)

$$[= B'D'(A+A')(C+C') + A'D]$$

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

$$= B'D(A\cdot C + AC' + A'C + A'C') + A'D(BC + BC' + B'C + B'C') + BD(AC + AC' + A'C + AC)$$

$= B'DAC + B'DAC' + \underline{B'DA'C} + \underline{B'DA'C'} + A'DBC + A'BC'D' + \underline{A'DB'C} + \underline{A'DBC'} + BDAC + BDA'C + BDAC' + BDA'C'$

$= B'DAC + B'DAC' + B'DA'C + B'DA'C' + A'DBC + A'BC'D + A'DB'C + BDAC + BDA'C + BDAC' + BDA'C'$.

Product of maxterms:

$F(A,B,C,D) = B'D + A'D + BD$

$B'D \rightarrow A,C \quad A'D \rightarrow B,C \quad BD \rightarrow A,C$

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

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

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

→ 'karnaugh' Map:- (Veitch diagram)

1. Minimising the boolean expressions without using laws, theorems.
2. In the year 1953
3. k-map can be drawn
   1. 2 variables
   2. 3 variables
   3. 4 variables.
4. K-map is represented by rectangular boxes where each box represents either minterms (or) maxterms.

1. 2 Variable k-map:-

'n' is the no. of variable

$n = 2$

$2^n = 2^2 = 4$.

4 → boxes will be there

15|10|22

m_0	m_1
m_2	m_3

	A	B	
0 → 0	0	0	0
1 → 0	0	1	0
2 → 1	1	0	0
3 → 1	1	1	1

$F(x,y) = x'y + xy' + xy$

01 10 11

x \ y	0	1
0	m_0	m_1 (1)
1	m_2 (1)	m_3 (1)

$= \Sigma(1,2,3)$

$= x + y$

2. 3 Variable k-map:-

'n' is no. of variables

$n = 3$

$2^n = 2^3 = 8$

8 → boxes will be there.

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

	B	C
→	0	0
→	0	1
→	1	0
→	1	1

A	B	C	
0	0	0	→ 0
0	0	1	→ 1
0	1	0	→ 2
0	1	1	→ 3
1	0	0	→ 4
1	0	1	→ 5
1	1	0	→ 6
1	1	1	→ 7

Example 1:- $\Sigma(2,3,4,5) = F$

x \ yz	00	01	11	10
0	m_0	m_1	m_3 1	m_2 1
1	m_4 1	m_5 1	m_7	m_6

$= xy' + x'y$

3. 4 variable

Example 2:- $F = \Sigma(3,4,6,7)$

x \ yz	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4 1	m_5	m_7 1	m_6 1

$= xz' + yz$

Example 3:- $f(x,y,z) = \Sigma(0,2,4,5,6)$

x \ yz	00	01	11	10
0	m_0 1	m_1	m_3	m_2 1
1	m_4 1	m_5 1	m_7	m_6 1

$z' + xy'$

Example 4:- $F(x,y,z) = \Sigma(0,2,4,5)$

x \ yz	00	01	11	10
0	m_0 1	m_1	m_3	m_2 1
1	m_4 1	m_5 1	m_7	m_6

$xy' + x'z'$

Example 5:- $F(x,y,z) = \Sigma(0,1,5,7)$

x \ yz	00	01	11	10
0	m_0 1	m_1 1	m_3	m_2
1	m_4	m_5 1	m_7 1	m_6

$xz + x'y'$

Example 6:- $F(x,y,z) = \Sigma(0,1,2,3,5)$

x \ yz	00	01	11	10
0	m_0 1	m_1 1	m_3 1	m_2 1
1	m_4	m_5 1	m_7	m_6

$z + x'$

17/10/22

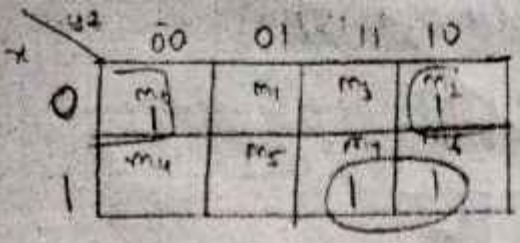
Example 1:- $F(x,y,z) = xyz + xyz' + xy'z + x'yz$

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

$= xy + xz + yz$

$= m_7 + m_6 + m_5 + m_3$

$= \Sigma(3,5,6,7)$



$x'z' + xy$.

3. 4 variable k-map:-

n' is no of variables

$n = 4$

$2^n = 2^4 = 16$

16 → boxes will be there.

xy \ zw	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Example 1:-

$F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$

wx \ yz	00	01	11	10
00	m_0 1	m_1 1	m_3	m_2 1
01	m_4 1	m_5 1	m_7	m_6 1
11	m_{12} 1	m_{13} 1	m_{15}	m_{14} 1
10	m_8 1	m_9 1	m_{11}	m_{10}

$y' + wz' + xz'$

Example 2:- $F(A,B,C,D) = \Sigma(4,6,7,15)$

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	
00	8	9	11	10

$BCD + A\bar{B}D$

Example 3:- $F(A,B,C,D) = \Sigma(3,7,11,13,14,15)$

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

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

Don't care conditions :-

1. It is a state which is not really mattered
2. We may consider these don't care, when the don't cares are used to group maximum number of cells.
3. Otherwise we can ignore these don't ~~tears~~ cares.
4. Don't cares are represented by 'X' cross mark.

Example : $F(A,B,C) = \Sigma(2,3,4,5) + \Sigma_d(6,7)$

A \ BC	00	01	11	10
0	m_0	m_1	m_3 1	m_2 1
1	m_4 1	m_5 1	m_7 X	m_6 X

$A + B$

Example 2:- $F(A,B,C,D) = \Sigma(5,6,7,12,14,15) + \Sigma_d(3,9,11)$

AB\CD	00	01	11	10
00	0	1	3 X	2
01	4	5 1	7 1	6 1
11	12 1	13	15 1	14 1
10	8	9 X	11 X	10

$CD + BC + ABD' + A'BD$

18/10/22

Product of sums:-

1. $F = \Pi(3,4,6,7,11,12,13,14,15)$ $[F(A,B,C,D)]$
2. $F = \Pi(2,4,6,12,14,15)$ $[F(A,B,C,D)]$

1.

AB\CD	00	01	11	10
00	M_0	M_1	M_3 0	M_2
01	M_4 0	M_5	M_7 0	M_6 0
11	M_{12} 0	M_{13} 0	M_{15} 0	M_{14} 0
10	M_8	M_9	M_{11} 0	M_{10}

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

2.

AB\CD	00	01	11	10
00	M_0	M_1	M_3	M_2 0
01	M_4 0	M_5	M_7	M_6 0
11	M_{12} 0	M_{13}	M_{15}	M_{14} 0
10	M_8	M_9	M_{11}	M_{10}

0 – A
1 – A'

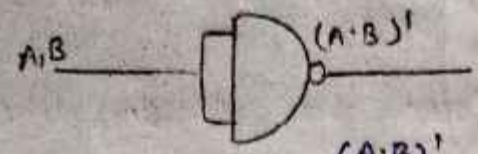
$(B'+D) \cdot (A'+B'+C') \cdot (A+C'+D)$

* NAND/NOR GATE IMPLEMENTATION:-

NAND and NOR gates are known as universal gates.

NAND:-

1. NOT gate using NAND gate.



$(A = B)$

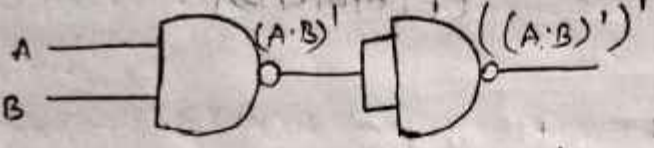
$(A \cdot B)'$

$= A' + B'$

$= A' + A'$ $(\because A + A = A)$

$= A'$

2. AND gate using NAND gate.

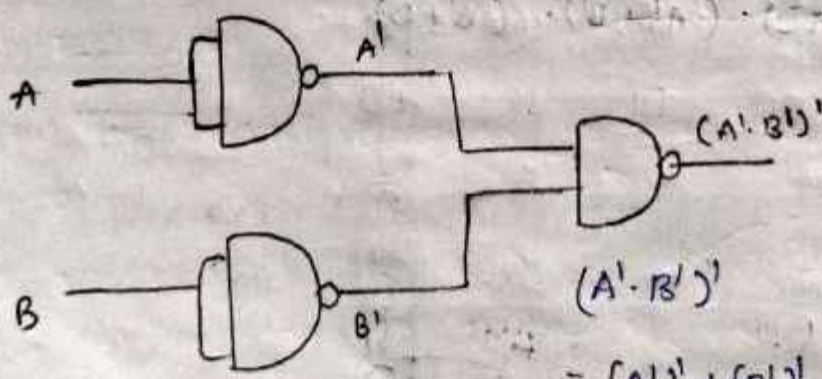


$= ((A \cdot B)')'$

$= A \cdot B$

∴ Two ~~cas~~ a NAND gates are required to represent AND gate.

3. OR gate using NAND gate.



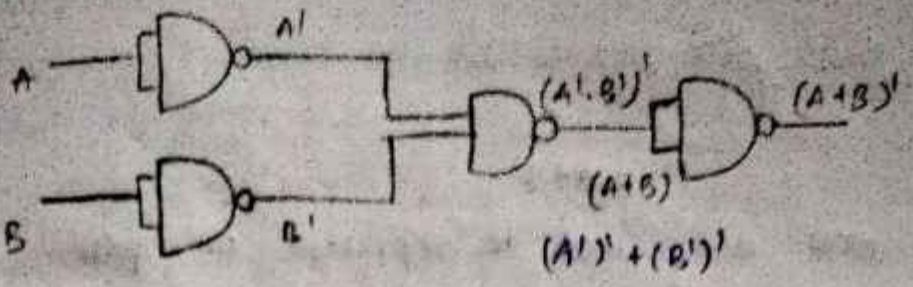
$(A' \cdot B')'$

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

$= A + B$

∴ Three NAND gates are required to represent a OR gate.

4. NOR Gate can be drawn by NAND gate



$(A')' + (B')'$

$= A + B$

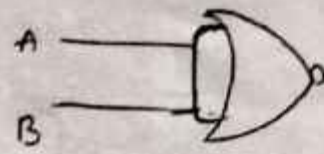
∴ four NAND gates are used to represent NOR gate.

Number of NAND gates used to represent other gates

	NAND
NOT	1
AND	2
OR	3
NOR	4

NOR :-

1. NOT using NOR Gate.



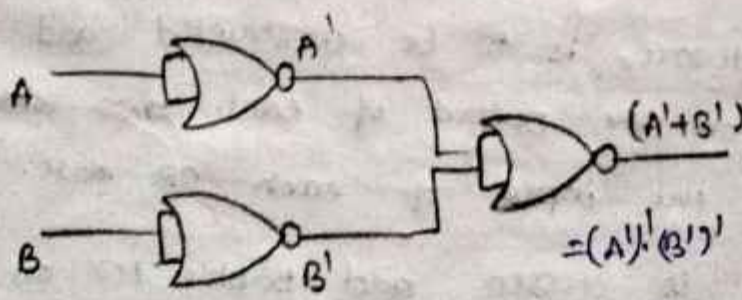
$(A = B)$

$(A+A)'$

$= A'$

∴ One NOR gate is used to represent NOT gate.

2. AND gate using NOR gate.

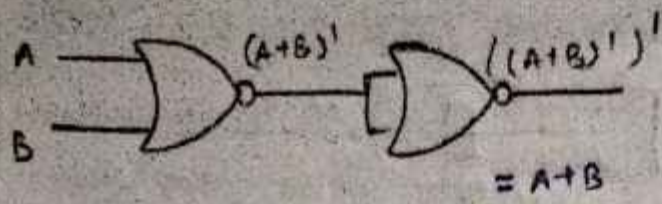


$= (A')'(B')'$

$= A \cdot B$

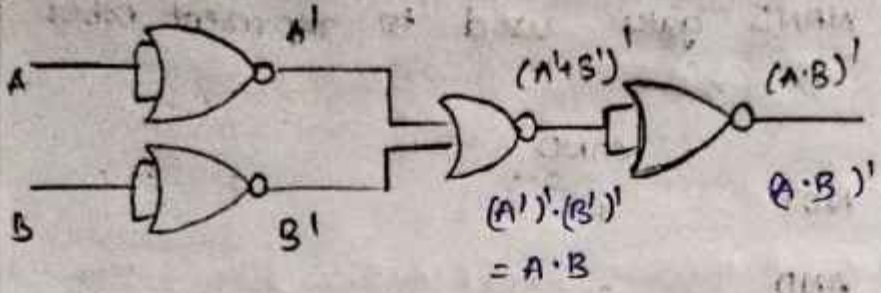
∴ Three NOR gates are used to represent NAND AND gate

3. OR gate using NOR gate.



∴ TWO NOR are used to represent OR gate.

4. NAND gate using NOR gate



∴ Four NOR gates are used to represent NAND gate.

Number of NOR gates required to represent other gates

	NOR
NOT	1
OR	2
AND	3
NAND	4

* CONVERSION OF

1. Draw the circuit diagram for the given boolean expression.

2. If NAND hardware is to be constructed add bubble (O) on the output of each AND gate and bubble (O) on the input of each OR gate.

3. If NOR gate is choosen add bubble (O) on the output of each OR gate and add bubble (O) on the input of each AND gate.

4. Add the inverter on each line that receives the bubble (o). (—o—) (')

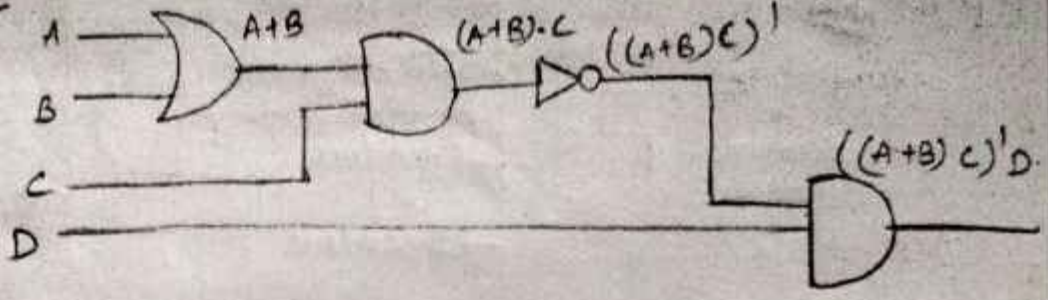
5. Replace bubbled (o) OR by NAND gate and bubbled (o) AND by NOR gate.

6. Eliminate any double inversions.

19/10/22

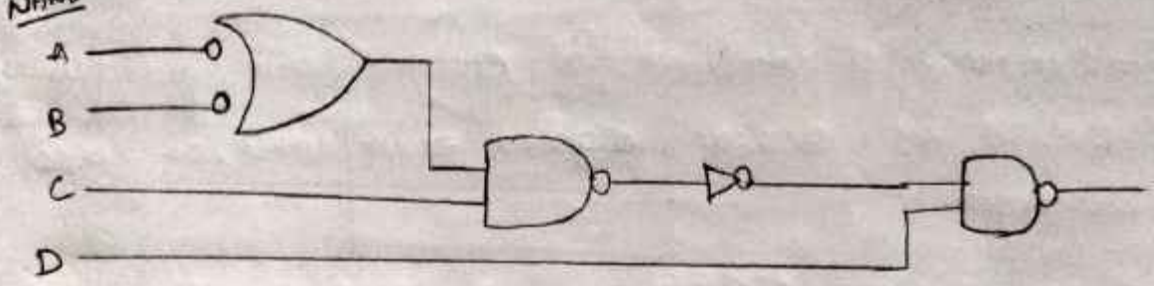
Example 1 :- $F = ((A+B)C)'D$

Step 1:-

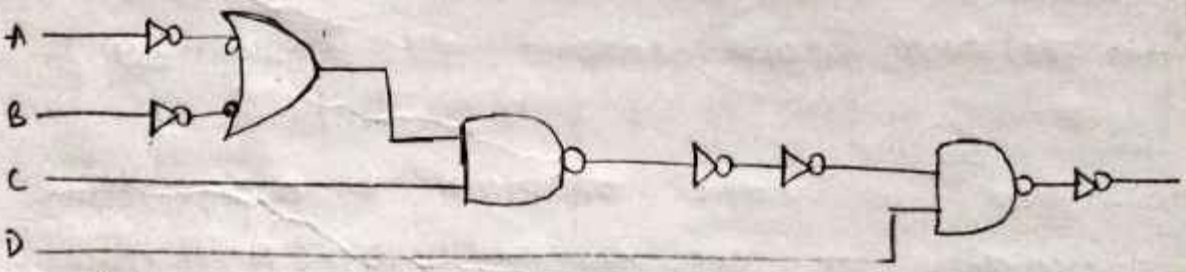


Step 2:-

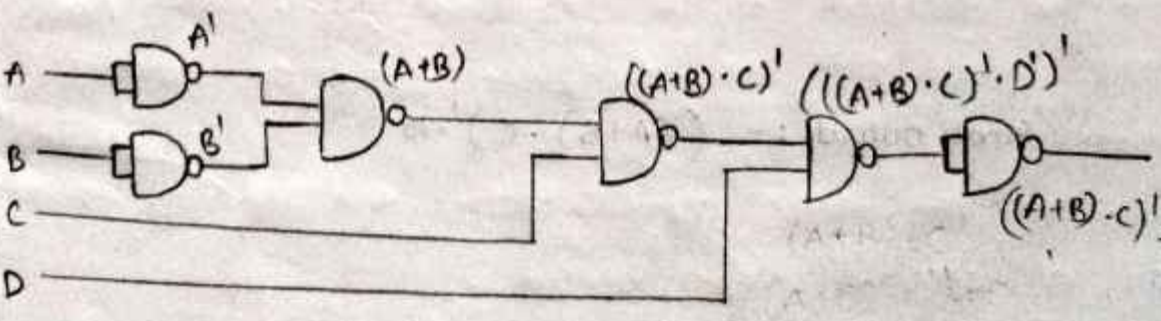
NAND



Step 3:-



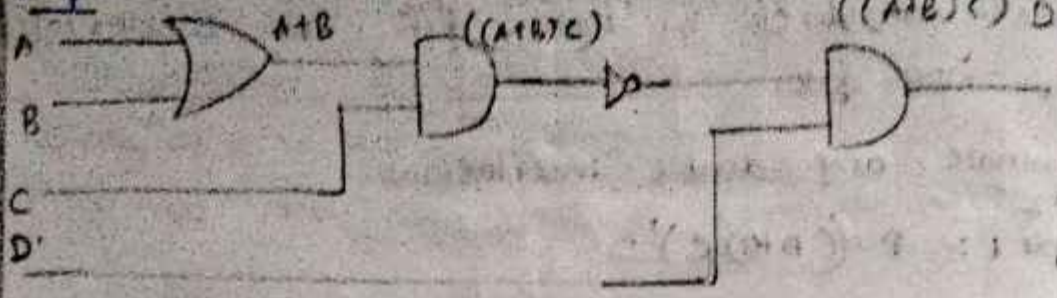
Step 4:-



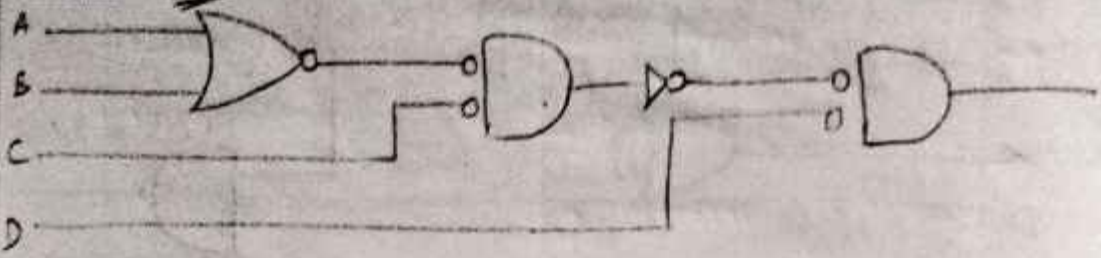
Final output $= ((A+B)\cdot C)'D$

Example :- $((A+B)C)'.D$

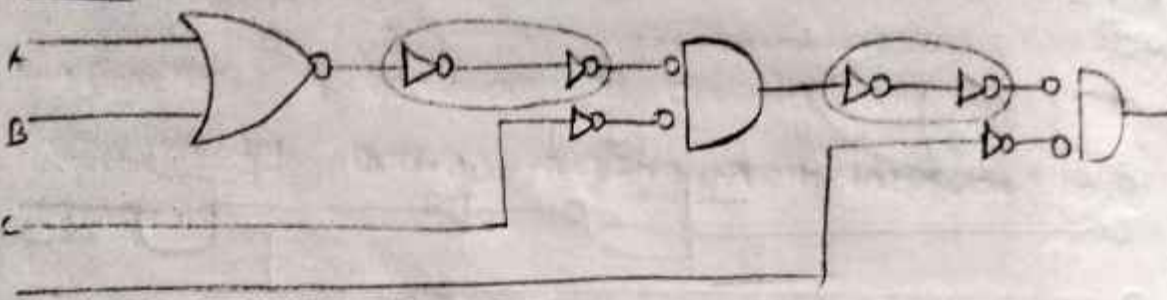
Step 1:-



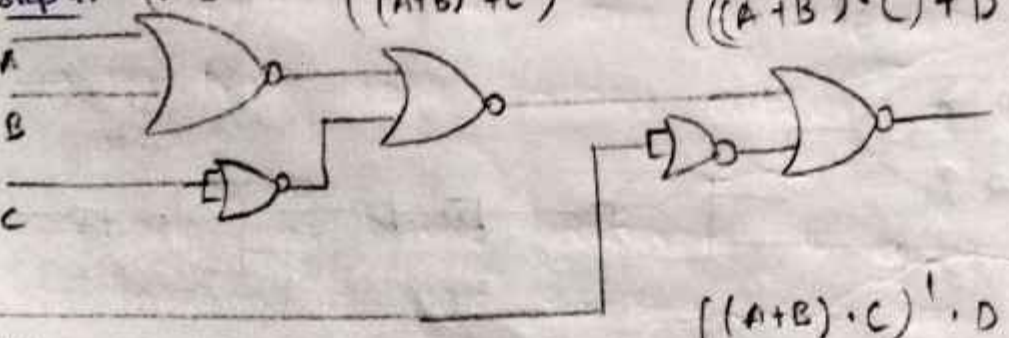
Step 2:- NOR



Step 3:-



Step 4:- $(A+B)'$ $((A+B)'+C')'$ $(((A+B)\cdot C)+D')'$



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

$C'+C' = C'$

Final output :- $((A+B)\cdot C)'\cdot D$

19/10/22

UNIT 3 · COMBINATIONAL CIRCUITS.

In Digital Electronics, we have 2 different types of circuits.

1. combinational circuits.
2. sequential circuits.

*Combinational circuits:- Simply, a circuit in which different types of logic gates are combined is known as a combinational logic circuit.

Combinational	sequential
→ Logic Gates	→ Logic Gates, storage elements
→ O/p will be based only on present I/p	→ The O/p is based on both present I/p & as well as the past I/p.
→ Adders, subtractors.	→ Latches.

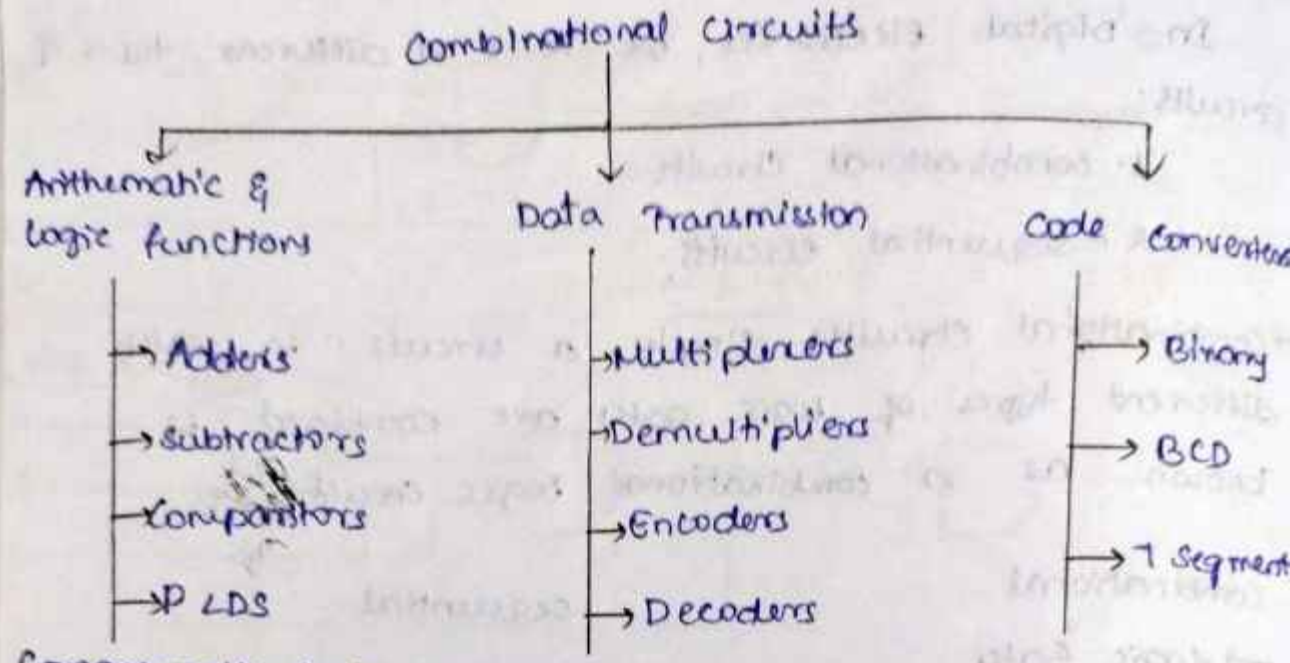
i.e combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.

A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.

The input variables, logic gates, and output variables are the basic components of the combinational logic circuit.

There are different types of combinational logic circuits.

*Types of Combinational circuits:-



→ Functions of combinational circuits are generally expressed by Boolean algebra, Truth table, or Logic diagram.

→ The Boolean Algebra is a logic functions that is expressed in the form of expression.

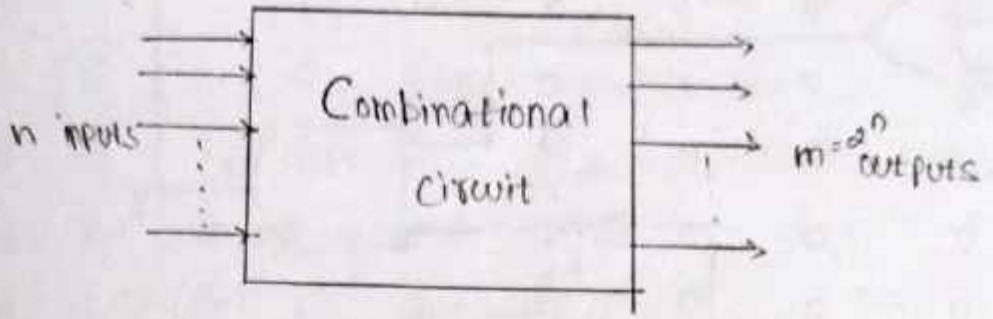
→ The truth table contains outputs of all possible inputs.

→ Where as logic diagram is a graphical representation of logic functions using logic gates.

→ The diagram of a combinational circuit has logic gates with no feedback paths or memory elements.

→ All the three representations are always equal for a specific logic circuit.

* BLOCK DIAGRAM OF THE COMBINATIONAL CIRCUIT :-



9/10/22

For n input variables, there are 2^n possible combinations of the binary outputs.

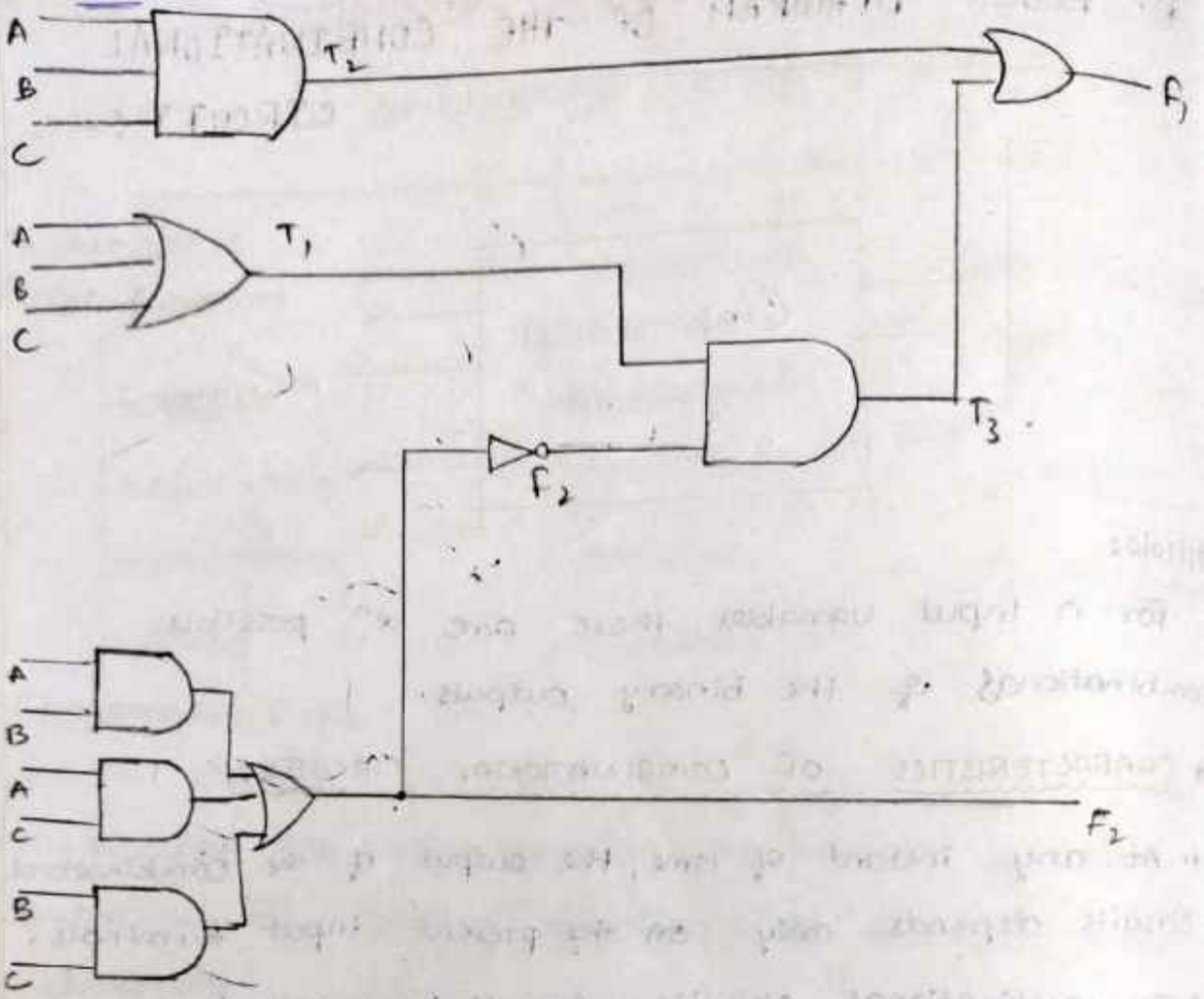
* CHARACTERISTICS OF COMBINATIONAL CIRCUITS :-

1. At any instant of time, the output of the combinational circuits depends only on the present input terminals.
2. The combinational circuits doesn't have any backup or previous memory. The present state of the circuit is not affected by the previous state of the input.
3. The n number of inputs and m number of outputs are possible in combinational logic circuits.

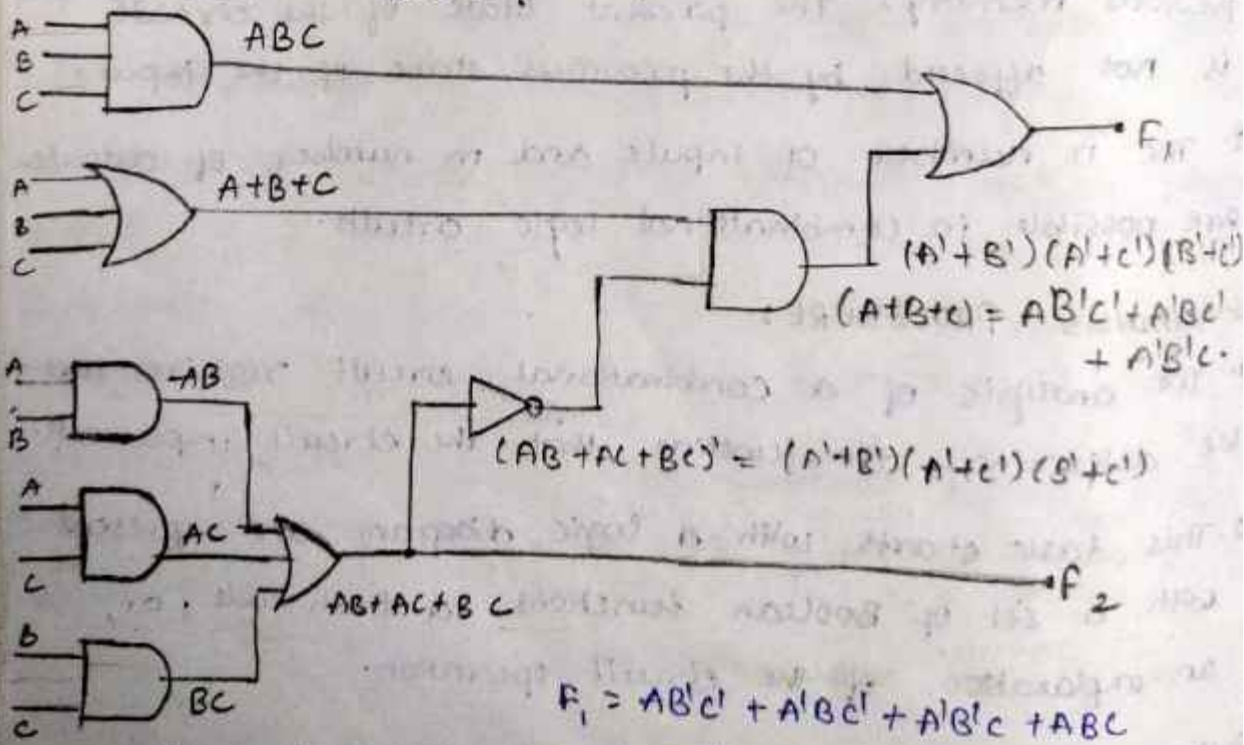
* ANALYSIS PROCEDURE :-

1. The analysis of a combinational circuit requires that we determine the function that the circuit implements.
2. This task starts with a logic diagram and represent with a set of Boolean functions, a truth table, or, an explanation of the circuit operation.
3. Here, First we given the circuit and findout the function.

Question:-



Boolean Expression Approach:-



$F_1 = AB'C' + A'BC' + A'B'C + ABC$

$F_2 = AB + AC + BC.$

Truth table approach:-

A	B	C	A'	B'	C'	AB'C'	A'BC'	A'B'C	ABC	F_1
0	0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1
0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	1	0	0	0	1
1	0	1	0	1	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1

F_2 AB	AC	BC	$F_2 = AB + AC + BC$
0	0	0	0 → 0
0	0	0	0 → 1
0	0	0	0 → 2
0	0	1	1 → 3
0	0	0	0 → 4
0	1	0	1 → 5
1	0	0	1 → 6
1	1	1	1 → 7

F_1:-

A \ BC	00	01	11	00
0	m_0	m_1 (1)	m_3	m_2 (1)
1	m_4 (1)	m_5	m_7 (1)	m_6

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

F_2:-

A \ BC	00	01	11	10
0	m_0	m_1	m_3 (1)	m_2
1	m_4	m_5 (1)	m_7 (1)	m_6 (1)

$AC + AB + BC$

*Uses of Combinational Circuits.

1. One of the most common uses of Combinational Logic is in Multiplexer and De-multiplexer type circuits. Here, multiple inputs or outputs are connected to a common signal line.

2. Applications :- Eg Calculators, digital measuring techniques, computers, digital processing, automatic control of machines, industrial processing, digital communications.

Most Important Standard Combinational Circuits are:-

Adders - Half Adder and Full Adder.

Subtractor - Half Subtractor, Full Subtractor.

Comparators

Decoders

Encoders

Multiplexers

DeMultiplexers etc.

Adders

→ Adder is based on the addition of Binary System.

For eg:-

* There are 2 kinds of Additions $1+0=1$, $1+1=10$, $1+1+1=11$, which are identified to be Half Addition and Full Addition.

* Half Addition is the Addition of 2 bits data, that Produces 2 bits outputs.

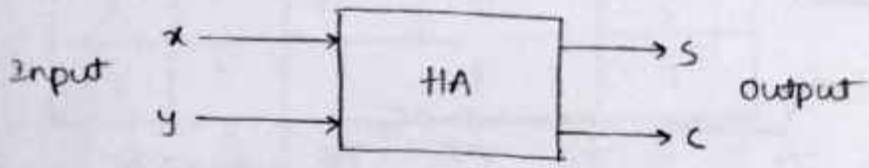
* Logic Circuit for Half Addition is known as "Half Adder".

* Full Addition is the Addition of 3 Bits data, that Produces 2 Bits outputs.

* Logic Circuit for Full Addition is known as "Full Adder".

Half Adder

* Half Adder is a combinational logic circuit with two inputs and two outputs.
* We assign symbols x and y to the two inputs and S (for sum) and C (for carry) to the outputs.
* Block Diagram:-



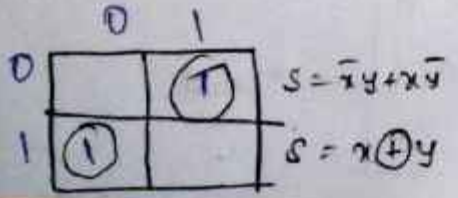
Truth Table:-

INPUT		OUTPUT	
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

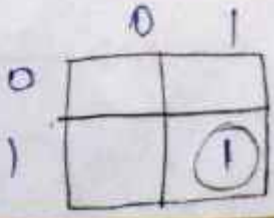
* In the above table, 'x' and 'y' are the input states, and 'sum' and 'carry' are the output states.
* The carry output is 0 in case of both inputs are not 1.
* The least significant bit of the sum is defined by the sum bit.
* The simplified sum of products expressions are:

The expressions for S and C using karnaugh map

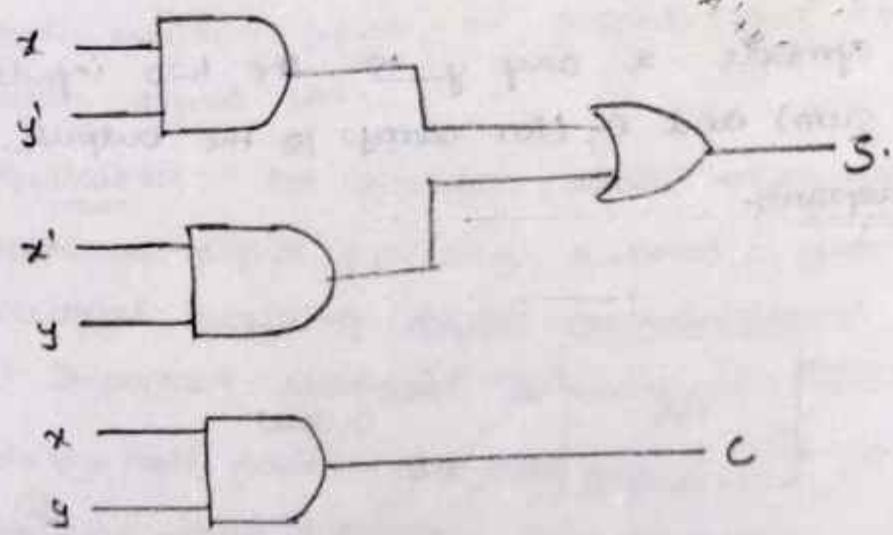
For S



For C



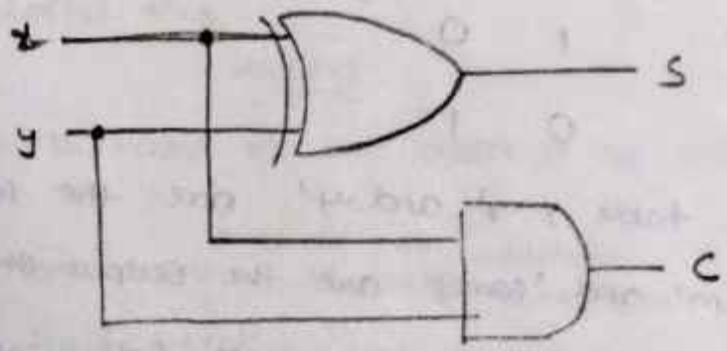
*The logic diagram of the half adder implemented in sum of products is shown in given figure:



(a) $S = xy' + x'y$

$C = xy$

*It can also be implemented with an exclusive-OR and AND gate as shown in given figure.



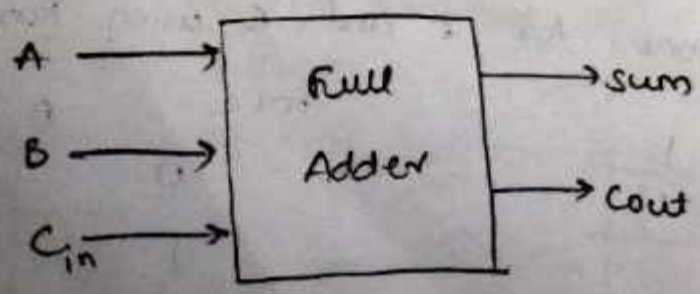
$S = x \oplus y$

$C = xy$

01/11/22

FULL ADDER :-

It is the adder which adds three inputs and produces two outputs.



A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum = $\Sigma(1,2,4,7)$

	00	01	11	10
0	m_0	m_1 1	m_3	m_2 1
1	m_4 1	m_5	m_7 1	m_6

$AB'C_{in}' + A'B'C_{in} + ABC_{in} + AB C_{in}'$

$C_{in}'(AB' + A'B) + C_{in}(A'B' + AB)$

$C_{in}'(A \oplus B) + C_{in}(\overline{A \oplus B})$

$A' \quad B \quad + \quad A \quad B'$

$\Rightarrow C_{in} \oplus A \oplus B$

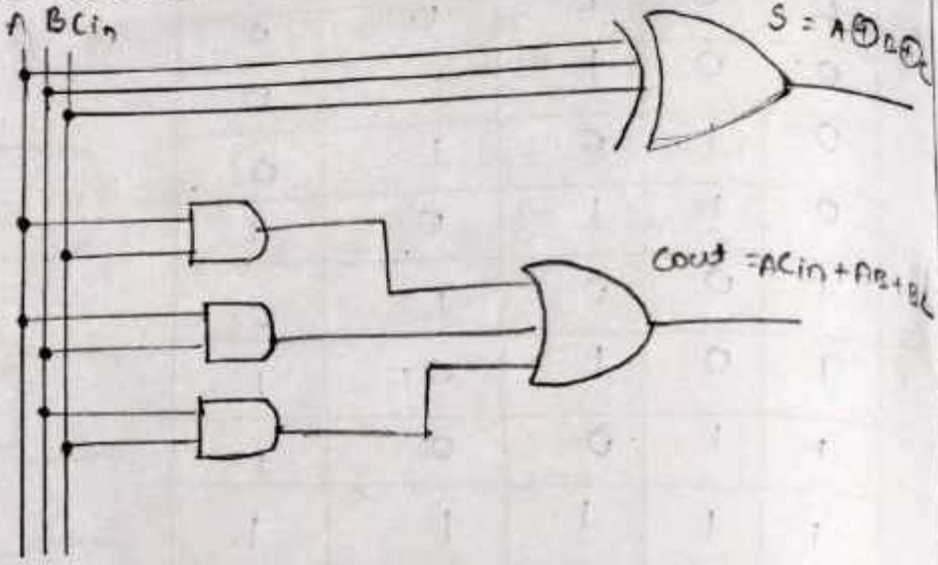
$\Rightarrow A \oplus B \oplus C_{in}$

Cout = $\Sigma(3,5,6,7)$

	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4	m_5 1	m_7 1	m_6 1

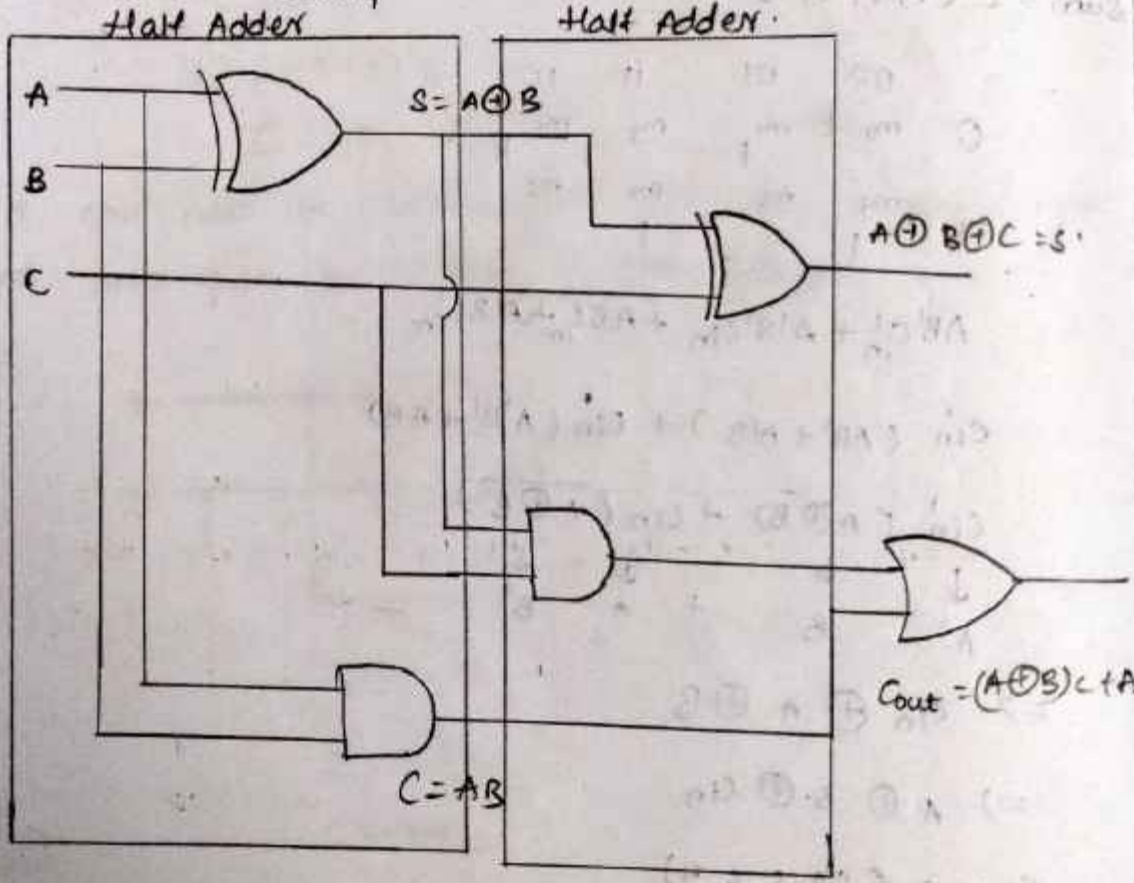
Cout = $AC_{in} + AB + BC_{in}'$

FULL ADDER DIAGRAM:-



8/11/22

FULL ADDER USING 2 HALF ADDERS :-



* The half adder is used to add only two numbers.
* TO overcome this problem, the full adder was developed. The full adder is used to add three 1-bit binary numbers A, B and carry C.
* The full adder has three inputs and two output states i.e. Sum and carry.

* BINARY ADDER:-

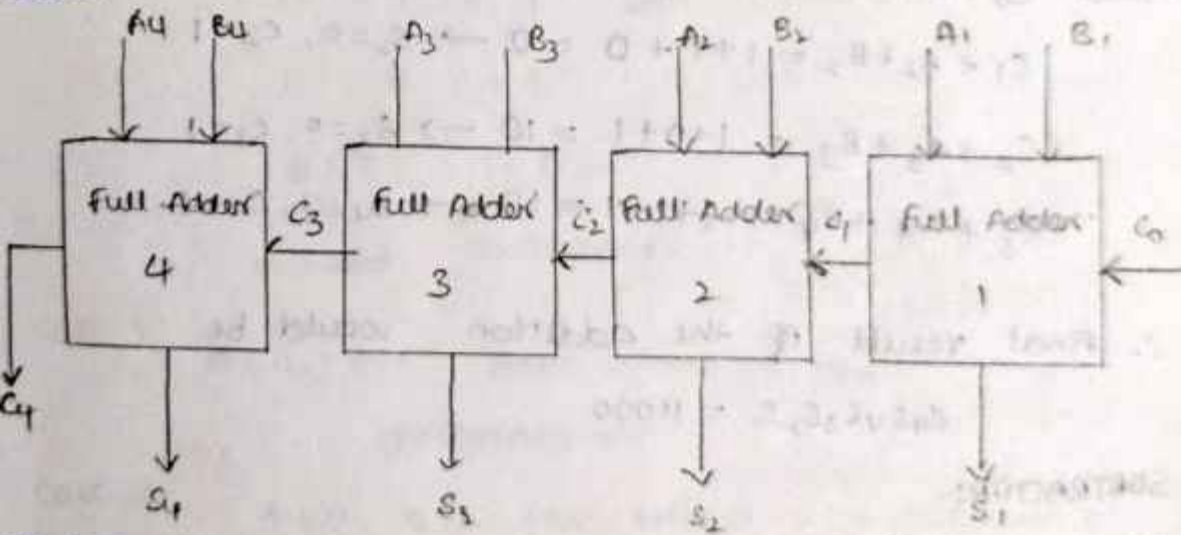
→ A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.

→ It can be constructed with full adders connected in series, with the output carry from each full adder connected to the input carry of the next full adder in the chain.

→ This parallel combination of full adders which perform addition of specific bits binary numbers is called binary adder: parallel adder.

→ By connecting 'n' number of full adders in parallel, an n-bit parallel adder can be constructed.

For adding two 4 bit binary numbers we have to connect 4 full adders to make 4 bit parallel adder.



→ The augend bits(A) and the addend bits(B) are designated by subscript numbers from right to left, with subscript '0' denoting the low-order bit.

→ The carry outputs for starts from C0 to C3 connected in a chain through the full-adders. C4 is the resultant carry generated by the last full-adder circuit.

→ The output carry from each full-adder is connected to the input carry of the next high-order full-adder.

→ The sum outputs (S0 to S3) generates the required arithmetic sum of augend and addend bits.

* Addition of two 4 bit binary numbers.

Add 1011 with 1101

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

Here $A_1=1, A_2=1, A_3=0, A_4=1$

$B_1=1, B_2=0, B_3=1, B_4=1$

As there is no previous carry $C_0=0$.

Now $C_0+A_1+B_1 = 0+1+1 = 10 \longrightarrow S_1=0, C_1=1$

$C_1+A_2+B_2 = 1+1+0 = 10 \longrightarrow S_2=0, C_2=1$

$C_2+A_3+B_3 = 1+0+1 = 10 \longrightarrow S_3=0, C_3=1$

$C_3+A_4+B_4 = 1+1+1 = 10 \longrightarrow S_4=1, C_4=1.$

∴ Final result of the addition would be

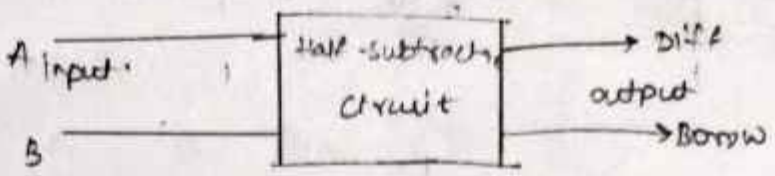
$C_4S_4S_3S_2S_1 = 11000$

SUBTRACTOR:-

1. Half Subtractor:-

* Half subtractor is the most essential combinational logic circuit which is used in digital electronics.
* The half subtractor is also a building block, for subtracting two binary numbers.
* It has 2 i/p's & 2 o/p's.
* This circuit is used to subtract two single bit

binary numbers A and B. The Diff' and borrow are 2 o/p states of the half subtractor.



* The combinational circuit which produces the differ between the two binary bits and borrow to indicat '1' has borrowed.

* Two inputs (A and B), two outputs difference (D) and borrow (Br or C).

* In the subtraction (A-B), A is called minuend and B is called subtrahend.

* TRUTH TABLE :-

$0 - 0 = 0$

$0 - 1 = 1$ with 1 borrow.

$1 - 0 = 1$

$1 - 1 = 0$

(1 - minuend, 1 - subtrahend.)

Case 1: A=0, B=0 then borrow =0 and difference =0

Case 2: A=0, B=1 then borrow =1 & difference =1

Case-3: A=1, B=0 then borrow =0 & difference =1

Case-4: A=1, B=1 then borrow =0 & difference =0.

A	B	$D = A \oplus B$	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

* K-MAP FOR HALF SUBTRACTOR:-

for D:-

A \ B	0	1
0		1
1	1	

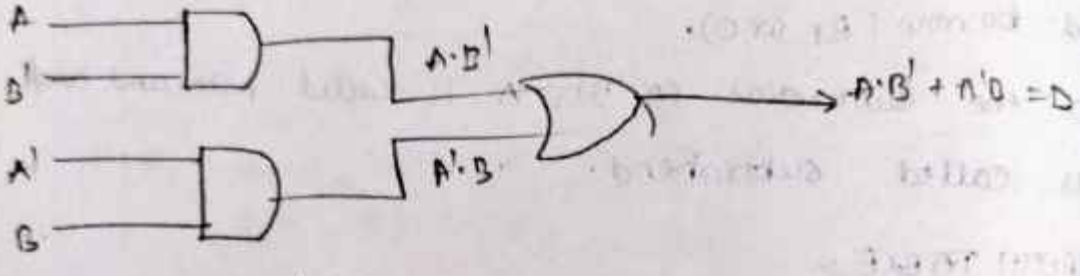
$D = A \oplus B$

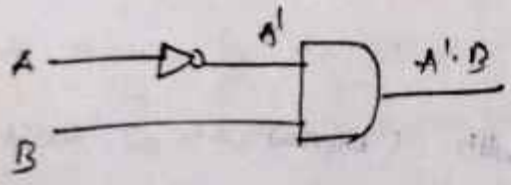
for b_r:-

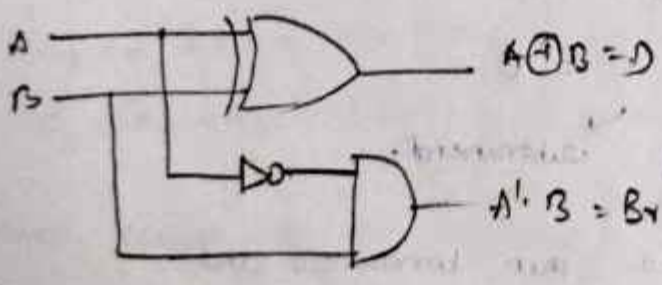
A \ B	0	1
0		1
1		

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

Circuit diagram:-







~~Half Adder~~.-

~~* Full Half adder~~.

* Full Subtractor:-

* The full subtractor is used to subtract only 2 numbers.

* To overcome this problem, a full subtractor is used.

* The full subtractor is used to subtract the three inputs are A, B and Bin denote the minuend, subtrahend

and previous borrow resp.

* The two outputs, D and Bout represent the diff and output borrow, respectively.

TRUTH TABLE:-

A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

* K-MAP REPRESENTATION:-

$\Sigma(1,2,4,7)$

A \ BBin	00	01	11	10
0		1		1
1	1		1	

$D = A'B'Bin + A'B Bin' + AB'Bin' + ABBin$

$\Sigma(1,2,3,7)$

A \ BBin	00	01	11	10
0		1	1	1
1			1	

$Bout = A'Bin + A'B + BBin'$

Simplification for D & Bout:-

$D = A'B'Bin + A'B B'in + AB' B'in + AB Bin$

$= Bin\,(A'B' + AB) + B'in\,(A'B + AB')$

$= Bin\,(\overline{A \oplus B}) + Bin'\,(A \oplus B)$

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

$B_{out} = A'B_{in} + A'B + BB_{in}'$

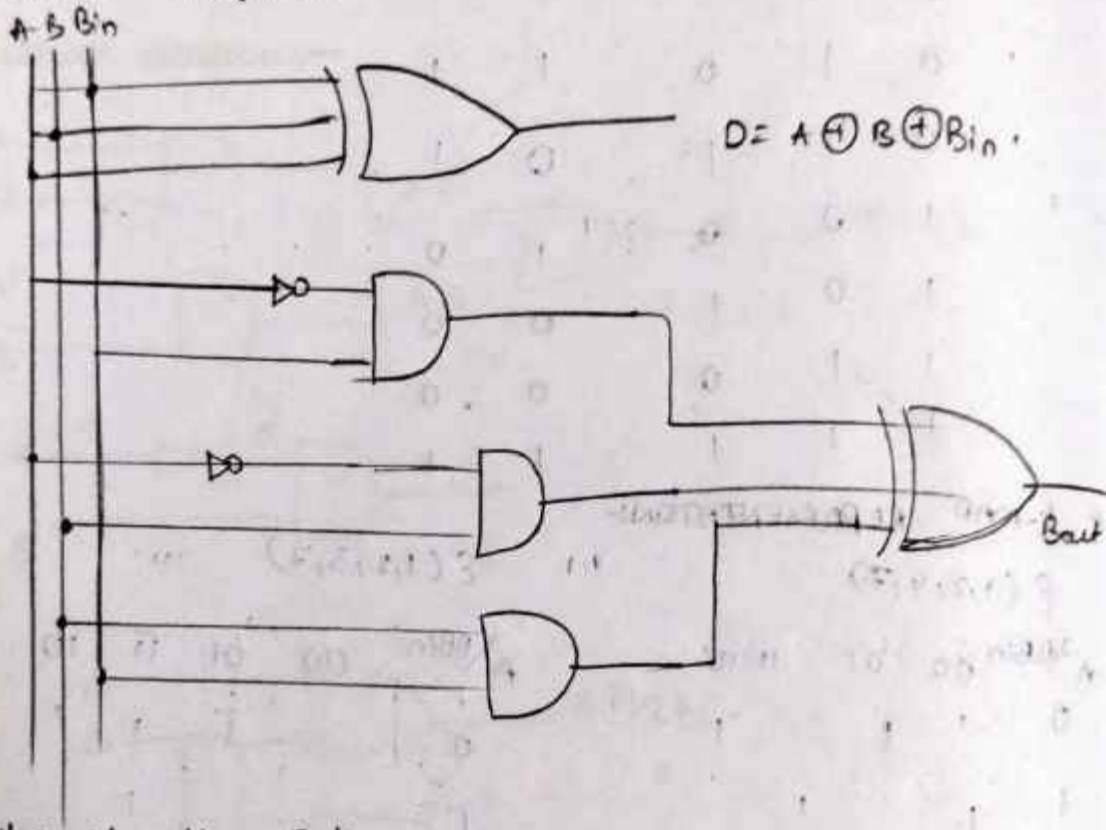
$B_{out} = A'B'B_{in} + A'B\,B_{in}' + A'BB_{in} + ABB_{in}.$

$= B_{in}(A'B' + AB) + A'B(B_{in}' + B_{in})$

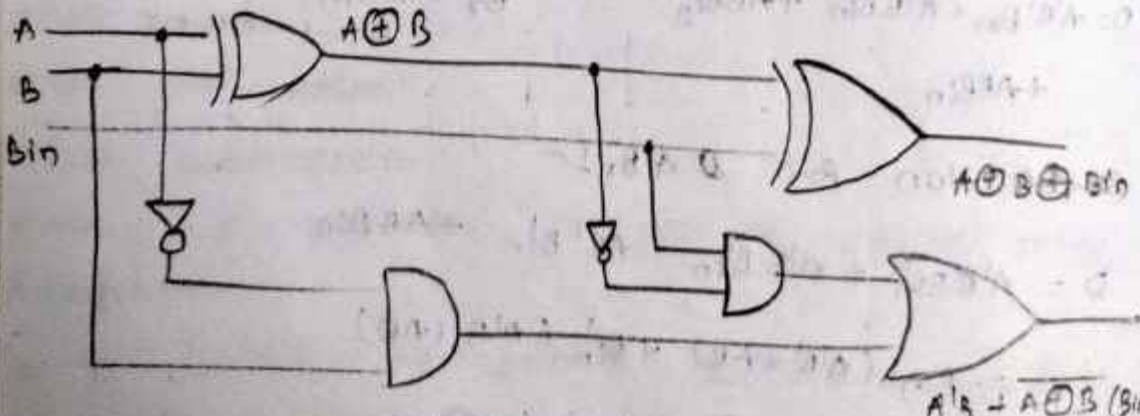
$= B_{in}(\overline{A \oplus B}) + A'B.$

$B_{out} = B_{in}(\overline{A \oplus B}) + A'B'$

Circuit Diagram:-



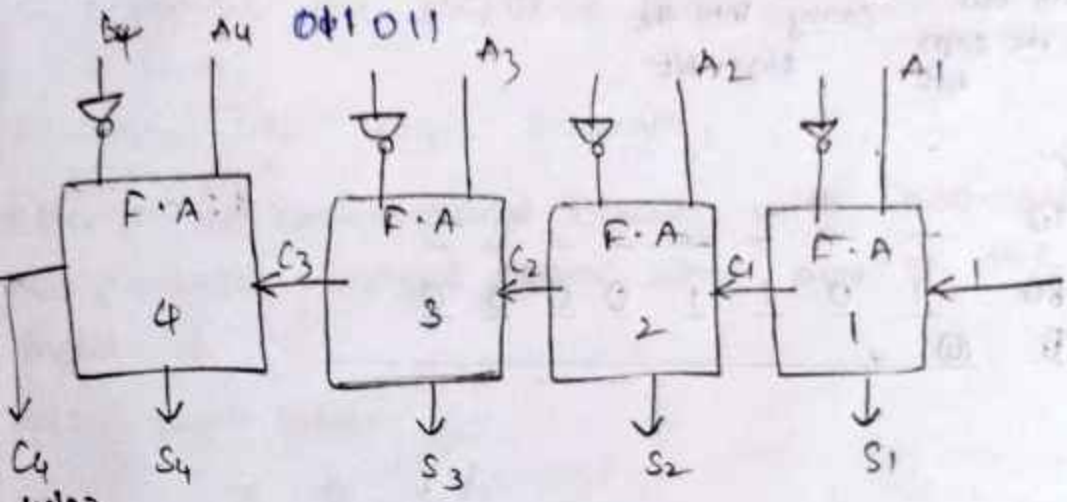
* Constructing Full Subtractor using Half - Subtractors:



* Binary subtractor:

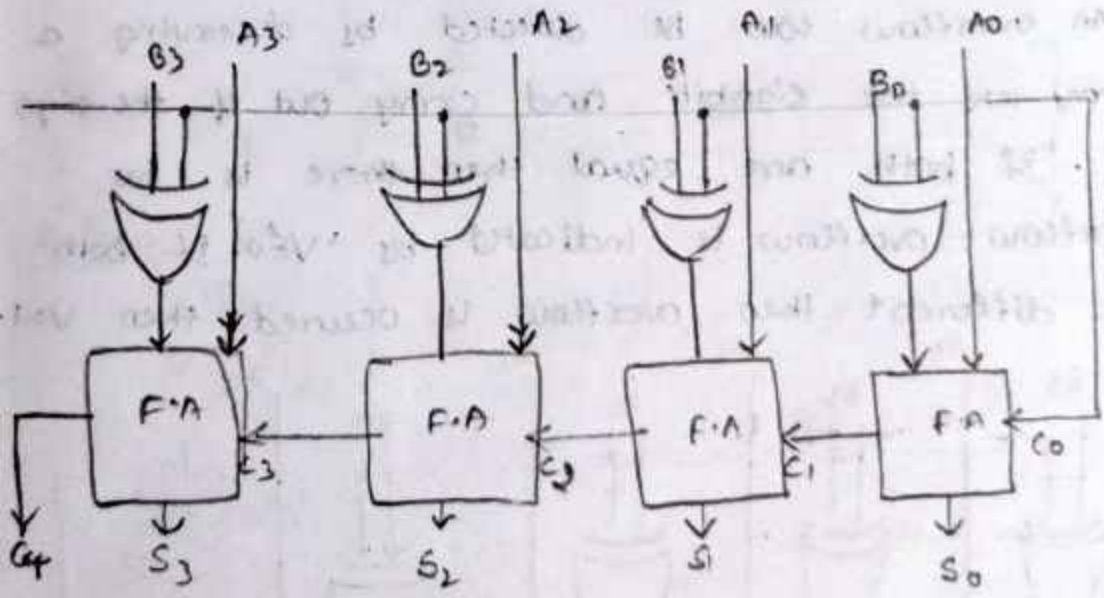
* Eg :- Let A = 11001 & B = 100101, 1's complement of 100101 => 011010. Now by adding 1 with LSB of this 1's complement number we get,.

```
  0 11010
+       1
---------
  0 11011
```



7/11/22

BINARY ADDER/ SUBTRACTOR :- (4-BIT):-



OVERFLOW:-

1. If we add n bits and we get n+1 digits as result. So this is overflow can be detected using flip-flop (It is device which can store one bit of info).
2. If we add one positive & one negative number then no overflow is occured. Overflow occurs only when we add two positive (or) negative numbers.

Example:-

+70 0 1 0 0 0 1 1 0

+80 0 1 0 1 0 0 0 0

(0) (1)

1 0 0 1 0 1 1 0

Carry out of the sign bit.

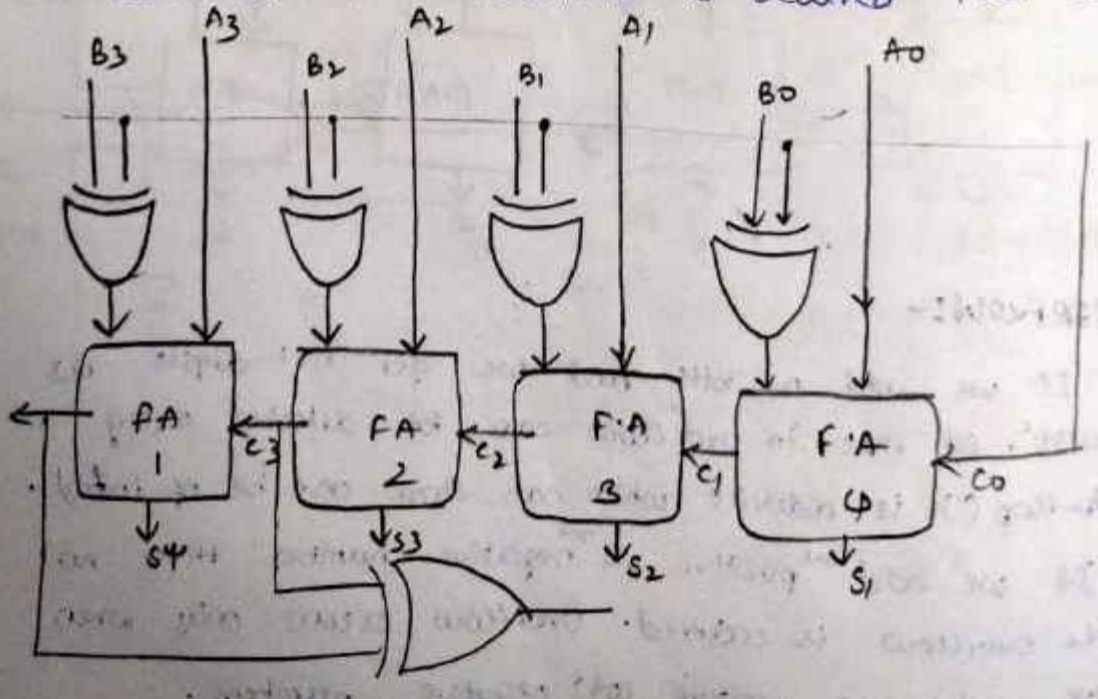
Carry into the sign bit

-70 — — — — — — — —

-80 1 0 1 1 0 0 0 0

(1) (0)

3. An overflow can be detected by observing a carry into the signbit and carry out of the sign bit. If both are equal then there is no overflow. overflow is indicated by 'V=0'. If both are different then overflow is occured then V=1.



DESIGN PROCEDURE:-

1. understand the problem definition definition.
2. Identify the no. of I/p's & no. of O/p's.
3. Assign letters/symbols to the I/p & o/p variable
4. Design Truth table accordingly to the requirement
5. Determine the simplified boolean expression using k-map.
6. Draw the Logic Diagram.

* Design a combinational circuit with two inputs that produces output 'zero' when any of the input is '1'.

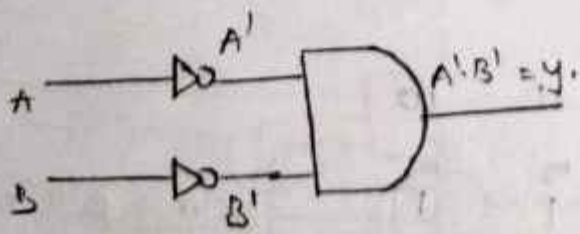
Ans:- Truth table:-

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

K-map

A\B	0	1
0	(1)	
1		

$A'B'$



* Design a combinational circuit that converts BCD code into excess -3 code.

	BCD				Excess-3			
	A	B	C	D	W	x	y	z
0→	0	0	0	0	0	0	1	1
1→	0	0	0	1	0	1	0	0
2→	0	0	1	0	0	1	0	1
3→	0	0	1	1	0	1	1	0
4→	0	1	0	0	0	1	1	1
5→	0	1	0	1	1	0	0	0
6→	0	1	1	0	1	0	0	1
7→	0	1	1	1	1	0	1	0
8→	1	0	0	0	1	0	1	1
9→	1	0	0	1	1	1	0	0

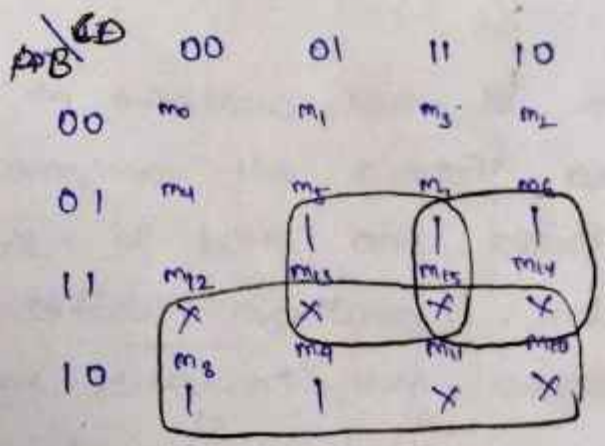
$w = \Sigma(5,6,7,8,9) + d(10,11,12,13,14,15)$

$x = \Sigma(1,2,3,4) + d(10,11,12,13,14,15)$

$y = \Sigma(0,3,4,7,8) + d(10,11,12,13,14,15)$

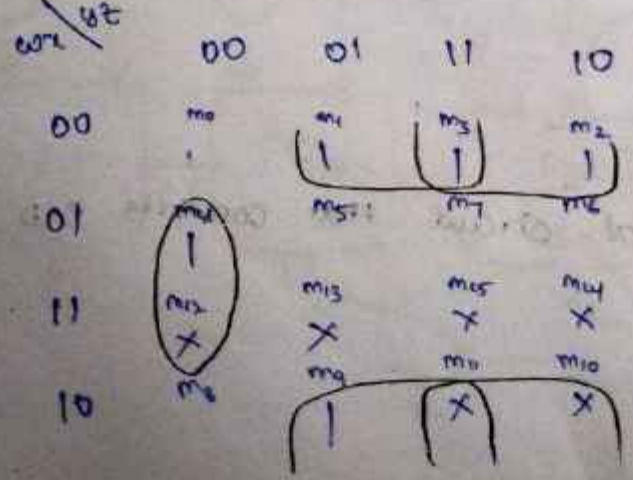
$z = \Sigma(0,2,4,6,8) + d(10,11,12,13,14,15)$

Expression for w.



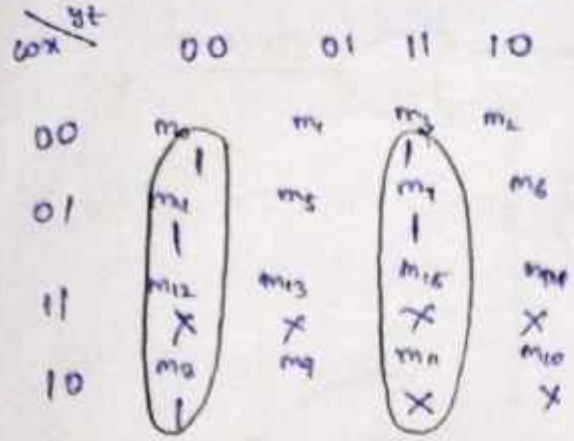
$A + BD + BC$

Expression for x.

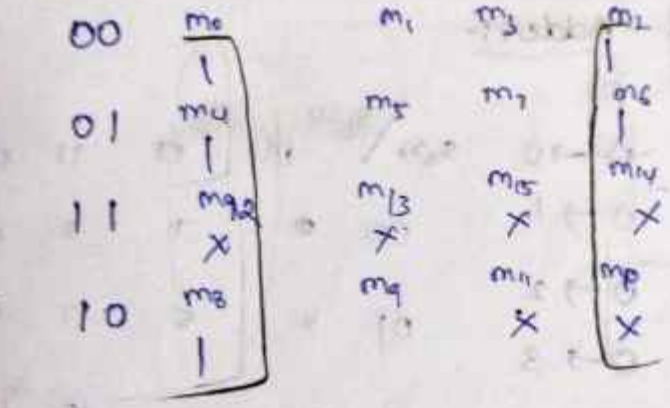


$BC'D' + B'D + B'C$

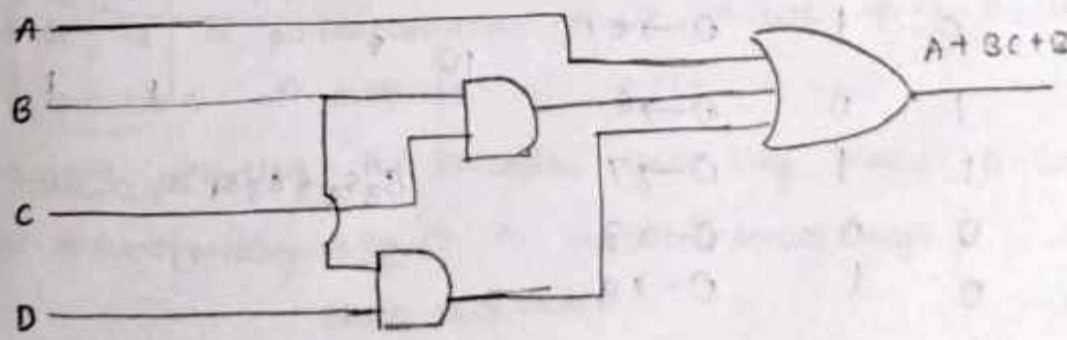
Expression for y:



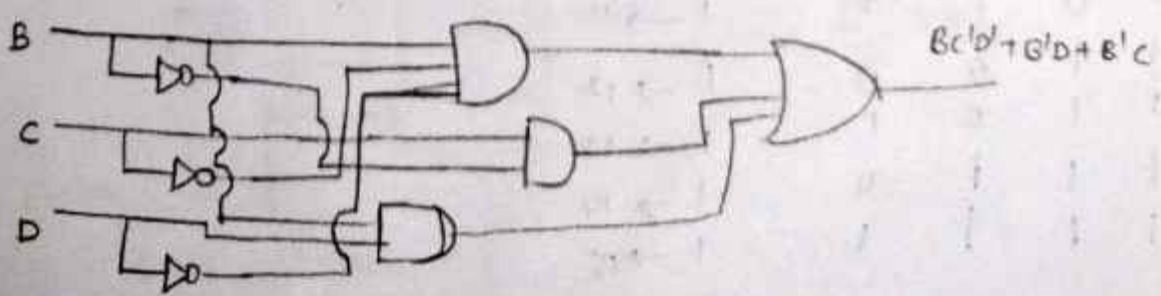
Expression for z:



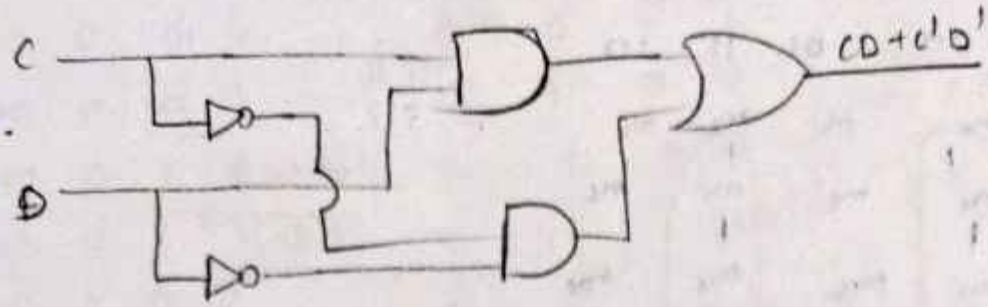
w:-



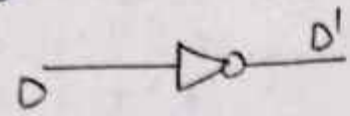
x:-



y:-



z.

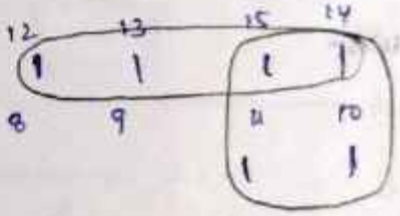


9/11/22

* BCD Adder / Decimal Adder:-

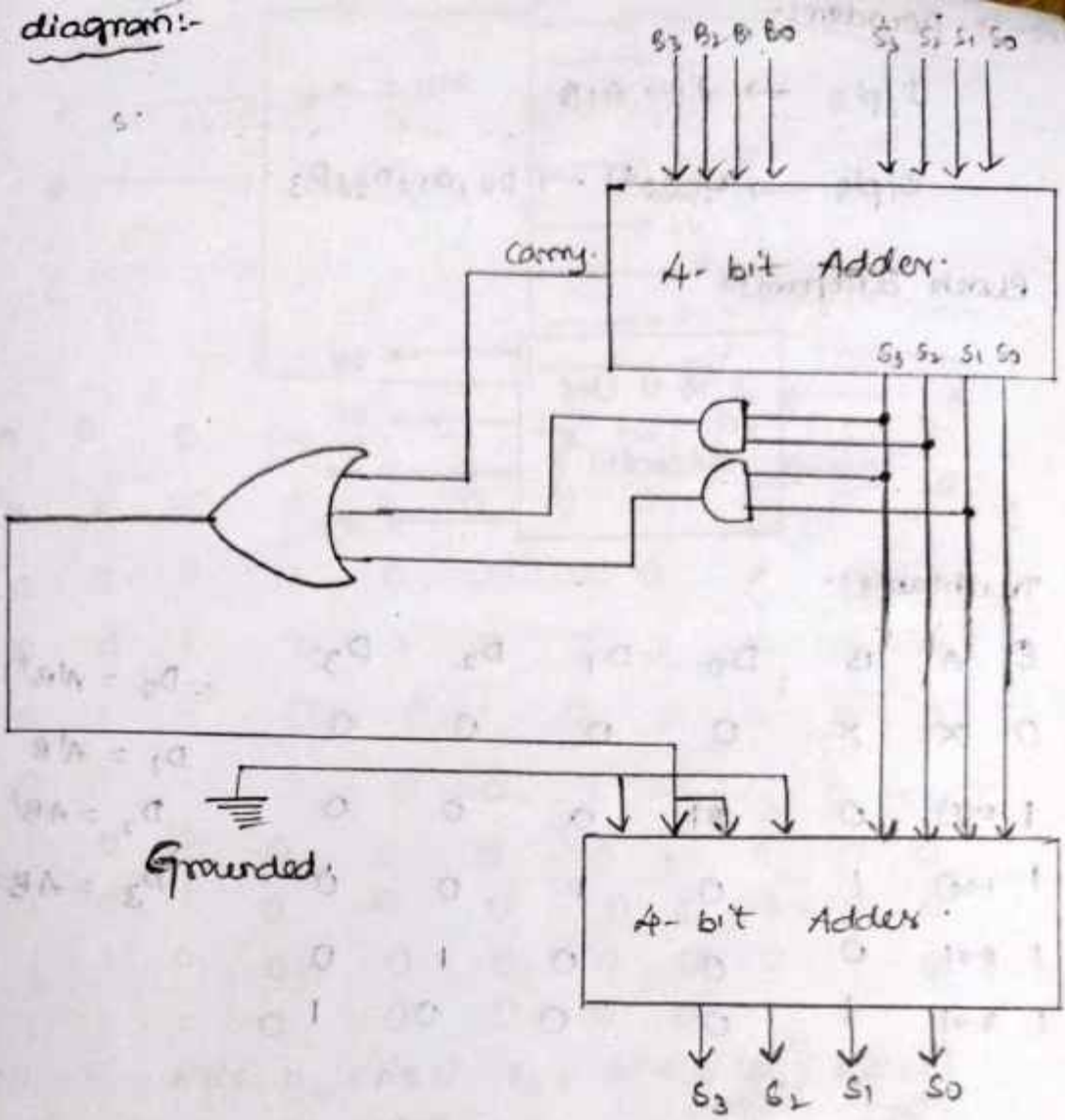
S_3	S_2	S_1	S_0	F
0	0	0	0	0 → 0
0	0	0	1	0 → 1
0	0	1	0	0 → 2
0	0	1	1	0 → 3
0	1	0	0	0 → 4
0	1	0	1	0 → 5
0	1	1	0	0 → 6
0	1	1	1	0 → 7
1	0	0	0	0 → 8
1	0	0	1	0 → 9
1	0	1	0	1 → 10
1	0	1	1	1 → 11
1	1	0	0	1 → 12
1	1	0	1	1 → 13
1	1	1	0	1 → 14
1	1	1	1	1 → 15

S_3S_2 \ S_1S_0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10



$S_3S_2 + S_3S_1 \Rightarrow$ Invalid
Sum greater than 9.

Logic diagram:-

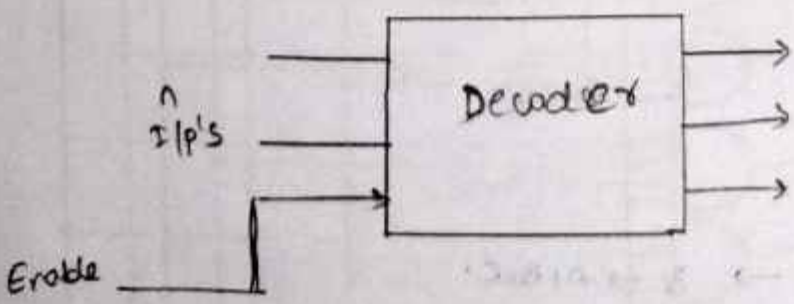


Decoder:-

→ It is a combinational circuit, if we give n inputs, it produces 2^n outputs.

→ while decoding the decoder generally places a logic '1' at one of its outputs to create exact code.

Block diagram:-



$E = 0$ → Disable

$E = 1$ → enable

2-4 Decoder:-

I/p's → 2 → A, B

O/p's → 4 (2^2) → D_0, D_1, D_2, D_3

Block diagram:-

Truth table:-

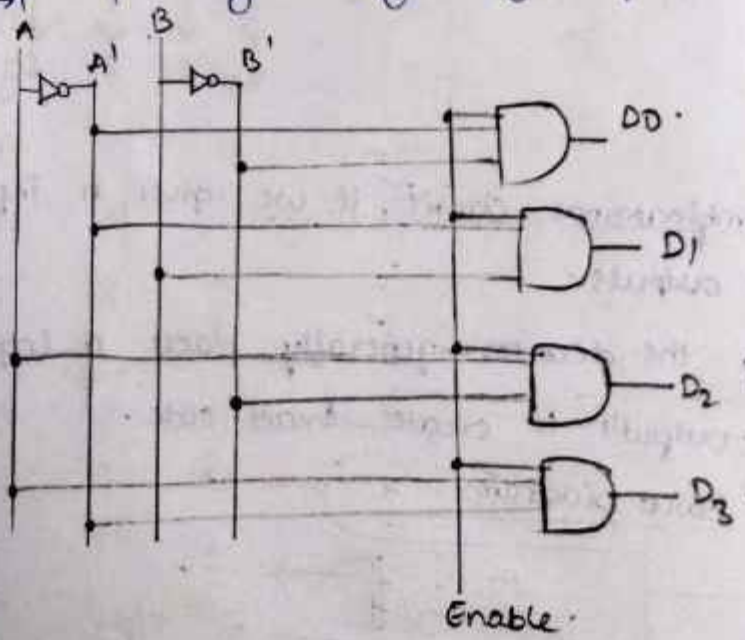
E	A	B	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0→0	0	1	0	0	0
1	1→0	1	0	1	0	0
1	2→1	0	0	0	1	0
1	3→1	1	0	0	0	1

$D_0 = A'B'$

$D_1 = A'B$

$D_2 = AB'$

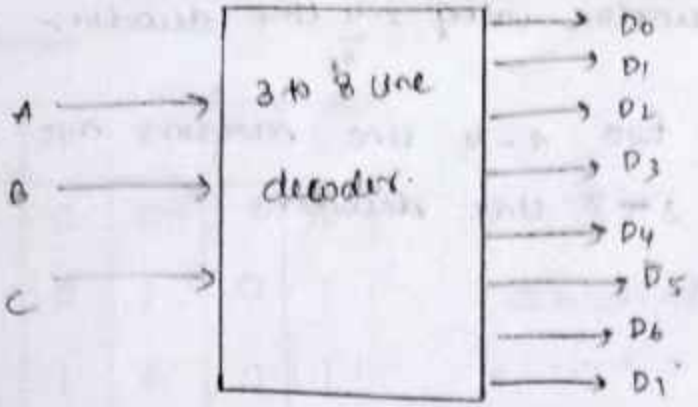
$D_3 = AB$



3-8 Decoder:-

I/p's → 3 → A, B, C.

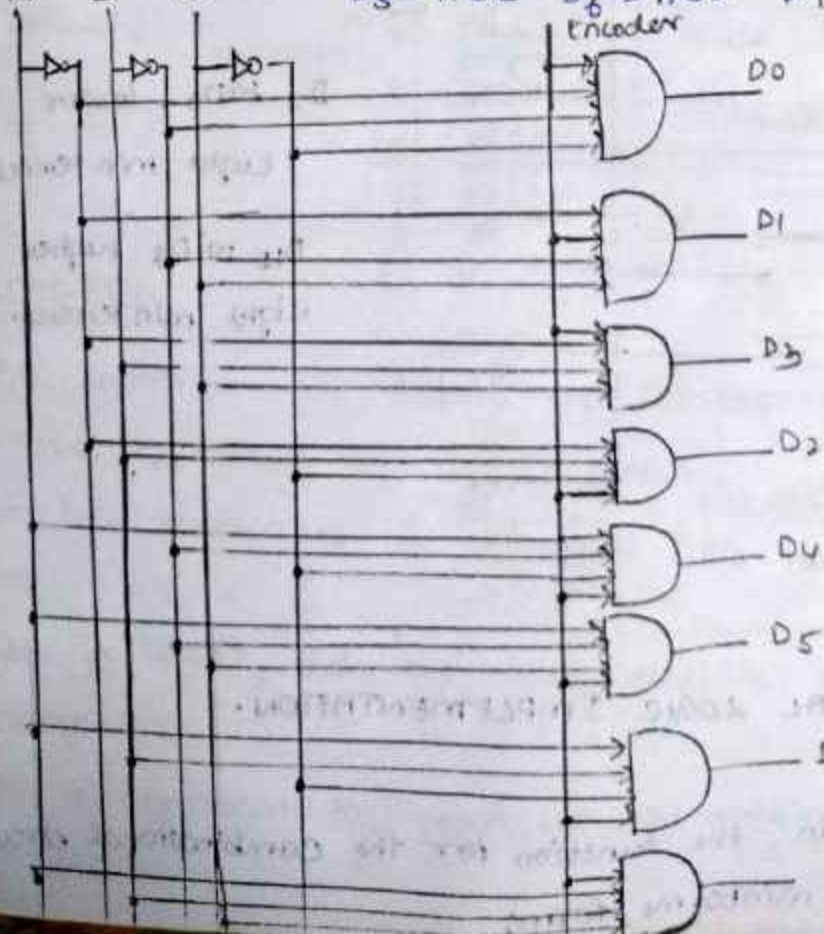
O/p's → 8 (2^3) → $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$.



E	A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

$D_0 = A'B'C'$ $D_1 = A'B'C$ $D_2 = A'BC'$ $D_3 = A'BC$ $D_4 = AB'C'$

$D_5 = AB'C$ $D_6 = ABC'$ $D_7 = ABC$.

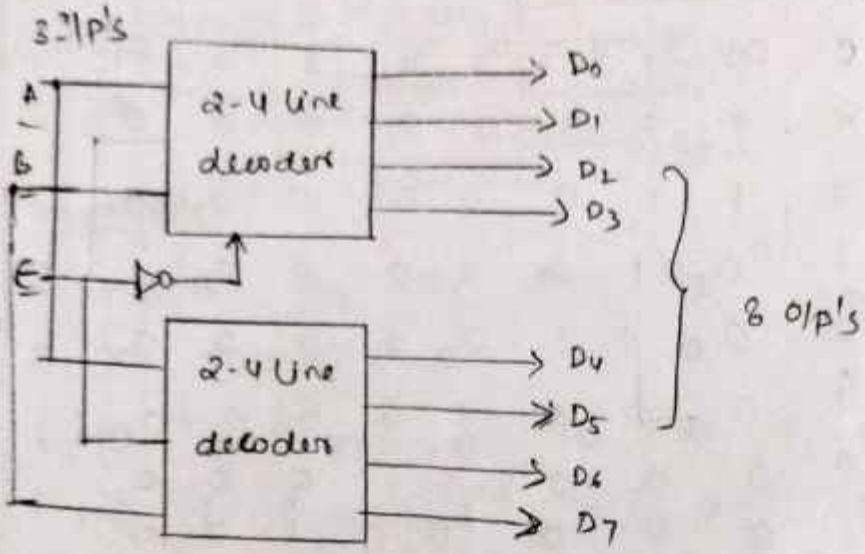


* 3-8 line decoder using 2-4 line decoder:- $\frac{n_2}{n_1} = \frac{8}{4} = 2$

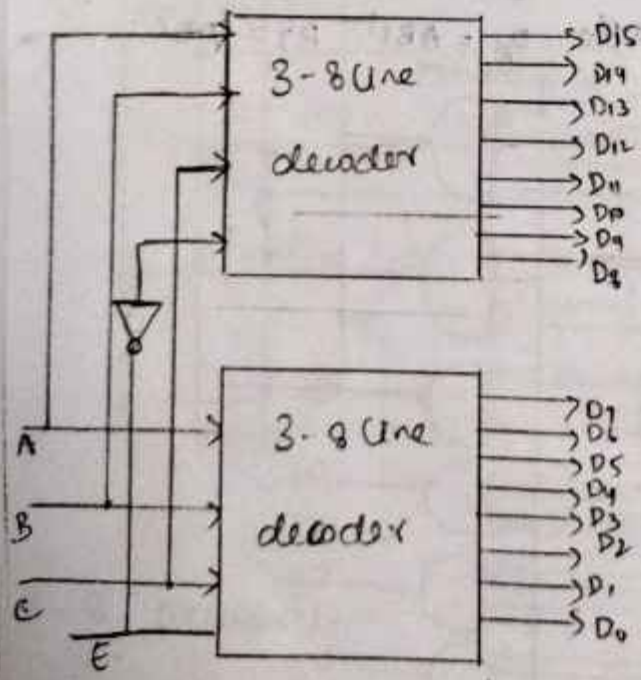
n_2 n_1

As $n_2/n_1 = 2$ two 2-4 line decoders are required to implement 3-8 line decoder.

Circuit diagram:-



* 4-16 line decoder using 3-8 line decoder:-



D_7 to D_0 lower eight min terms

D_{15} to D_8 higher eight min terms.

COMBINATIONAL LOGIC IMPLEMENTATION.

Ex: 1:-

We obtain the function for the combinational circuit in sum-of-minterms form:

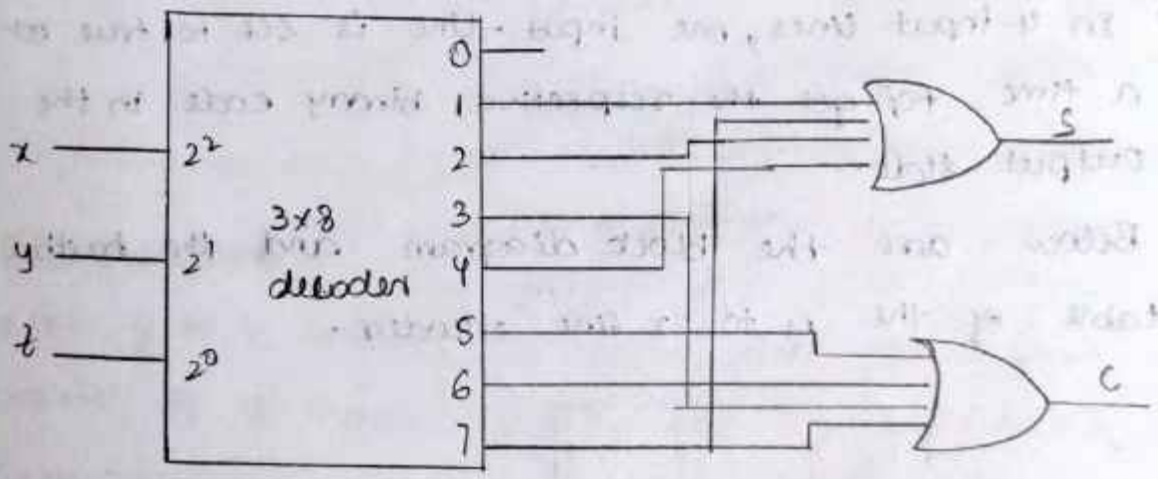
Full Adder

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

$S(x,y,z) = (1,2,4,7)$

$C(x,y,z) = (3,5,6,7)$

IMPLEMENTATION OF A FULL ADDER WITH A DECODER.



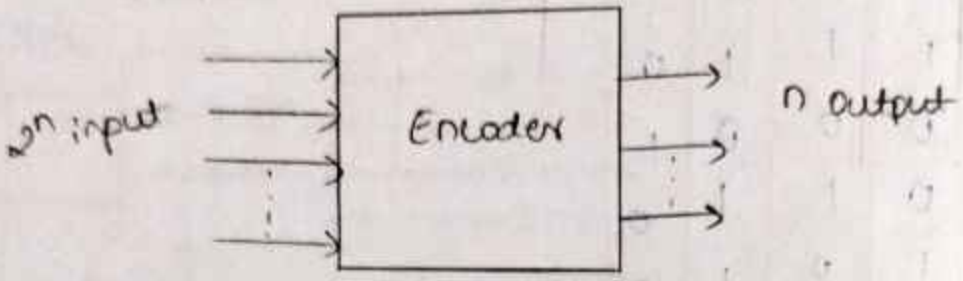
ENCODER:-

* An encoder is a Digital circuit that performs the reverse operation of the Decoder.
* It has maximum of 2^n input lines and n output lines.
* At a time, only one input line is activated for simplicity.
* It is optional to represent the enable signal in encoders.

There are various types of Encoders

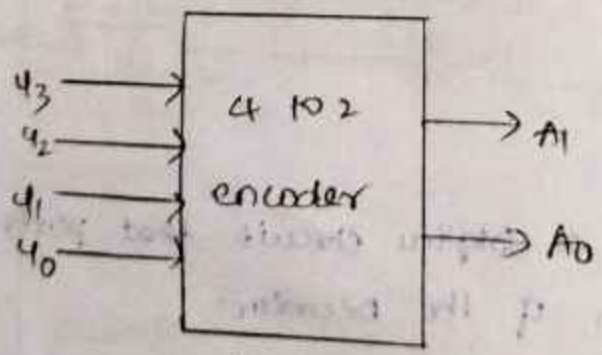
* 4-2 encoder
* 8-3 encoder
* 16-4 encoder and many more.

BLOCK DIAGRAM:-



4-2 ENCODER:-

* In 4 to 2 line encoder, there are total of four inputs, i.e., Y_0, Y_1, Y_2 and Y_3.
* In 4-input lines, one input-line is set to true at a time to get the respective binary code in the output side.
* Below are the block diagram and the truth table of the 4 to 2 line encoder.



From truth table, we can write the Boolean functions for each output as:

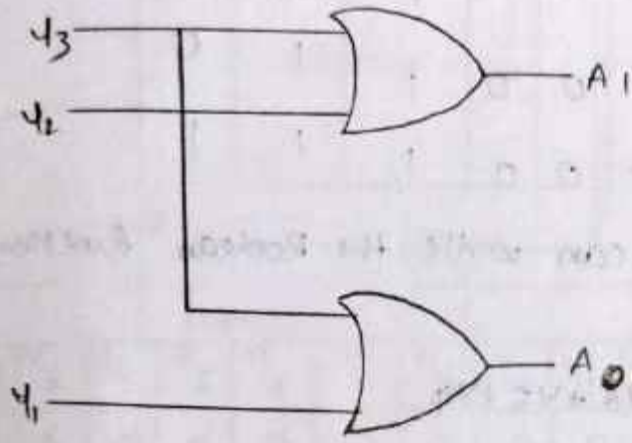
$$A_1 = Y_3 + Y_2$$

$$A_0 = Y_3 + Y_1$$

TRUTH TABLE:-

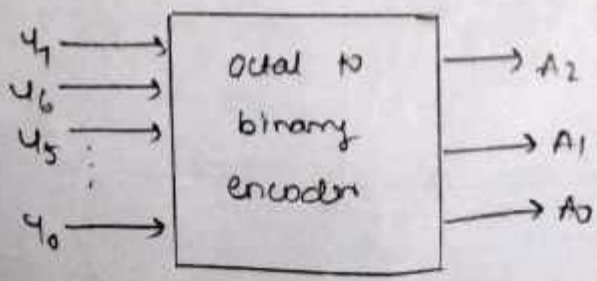
INPUTS				OUTPUTS	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

CIRCUIT DIAGRAM:-



8-3 ENCODER.

* The 8 to 3 encoder or octal to binary encoder consists of 8 inputs: Y_0 to Y_7 and 3 outputs: A_0 to A_2. Each input line corresponds to each octal digit and three outputs generate corresponding binary code.
* The 8 line 8 input line, one input-line is set to true at a time to get the respective binary code in the output side.



TRUTH TABLE :-

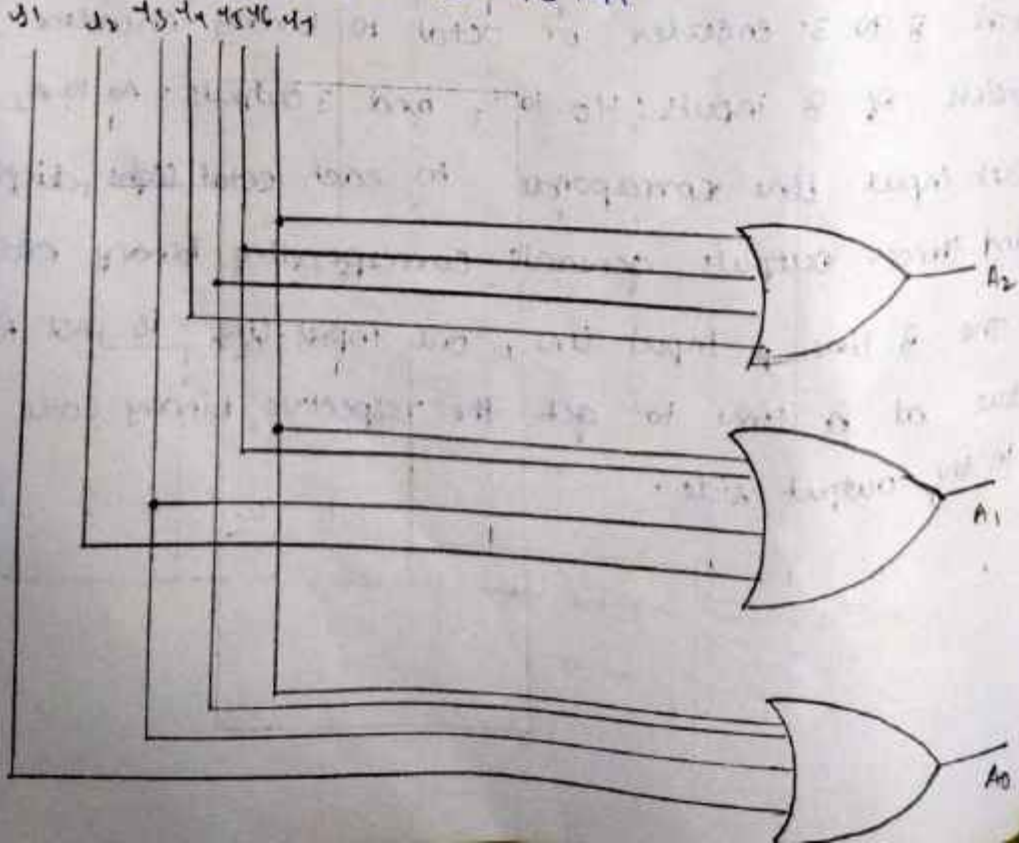
INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

From truth table we can write the Boolean function for each output as

$$A2 = Y7 + Y6 + Y5 + Y4$$

$$A1 = Y7 + Y6 + Y3 + Y2$$

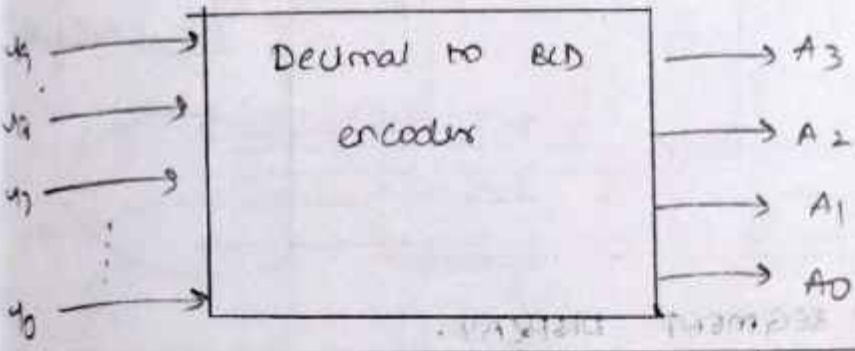
$$A0 = Y7 + Y5 + Y3 + Y1$$



DECIMAL TO BCD ENCODER.

* The octal to binary encoder is also known as 10 to 4 line encoder, there are total of ten inputs i.e., $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8$, and Y_9 and four outputs i.e., A0, A1, A2 and A3.

* In 10- input lines, one input line is set to true at a time to get the respective BCD code in the output side.



INPUTS										OUTPUTS.			
Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

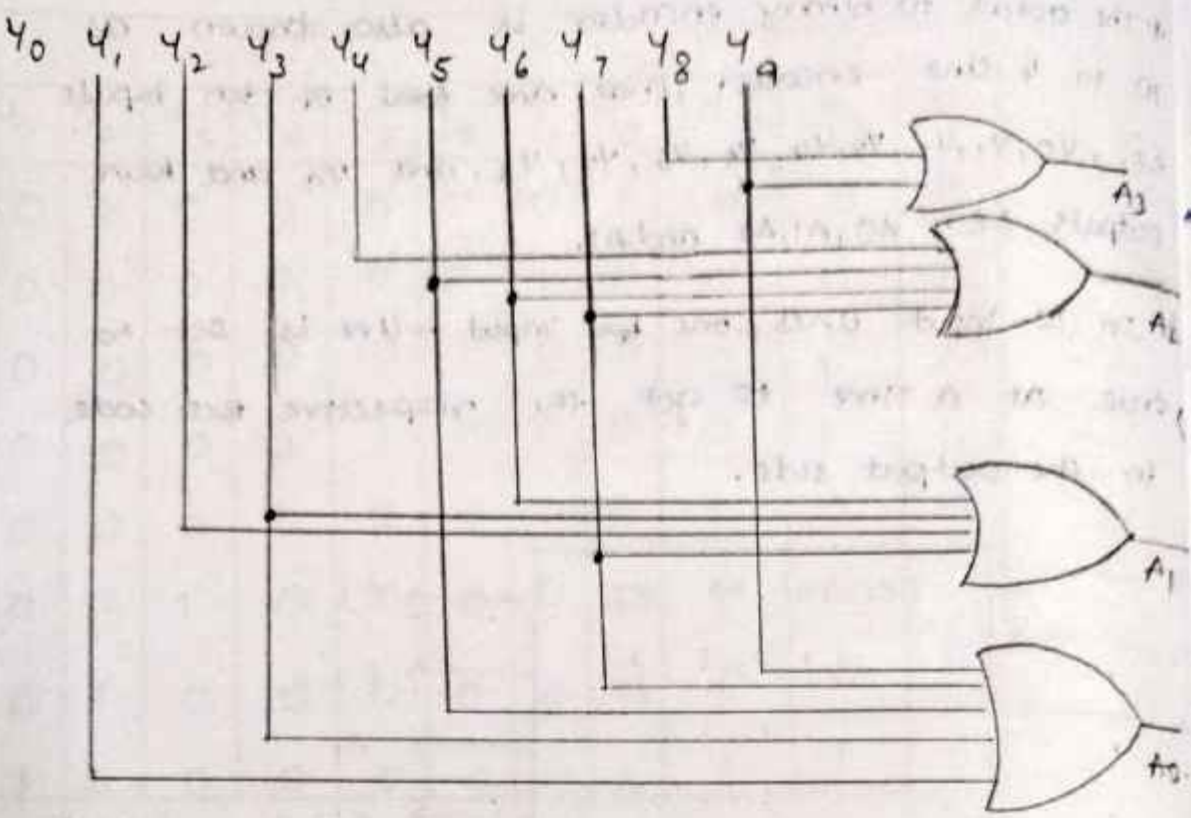
$$A_3 = Y_9 + Y_8$$

$$A_2 = Y_7 + Y_6 + Y_5 + Y_4$$

$$A_1 = Y_7 + Y_6 + Y_3 + Y_2$$

$$A_0 = Y_9 + Y_7 + Y_5 + Y_3 + Y_1$$

CIRCUIT DIAGRAM:-

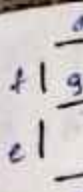


BCD TO SEVEN SEGMENT DISPLAY.

* A digital Decoder IC, is a device which converts one digital format into another and one of the most Commonly used devices for doing this is called the Binary Coded Decimal (BCD) to 7- Segment Display Decoder.

* BCD (Binary Coded Decimal) is an encoding scheme which represents each of the decimal numbers by its equivalent 4-bit binary pattern.

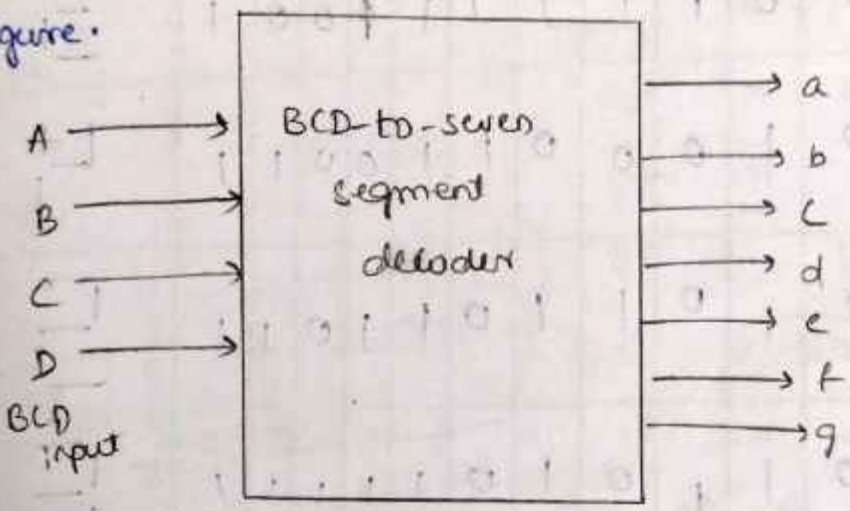
* Seven segment displays comprise of seven individual segments formed by either Light Emitting Diodes (LED's) or Liquid crystal Displays (LCD's) arranged in a definite pattern.

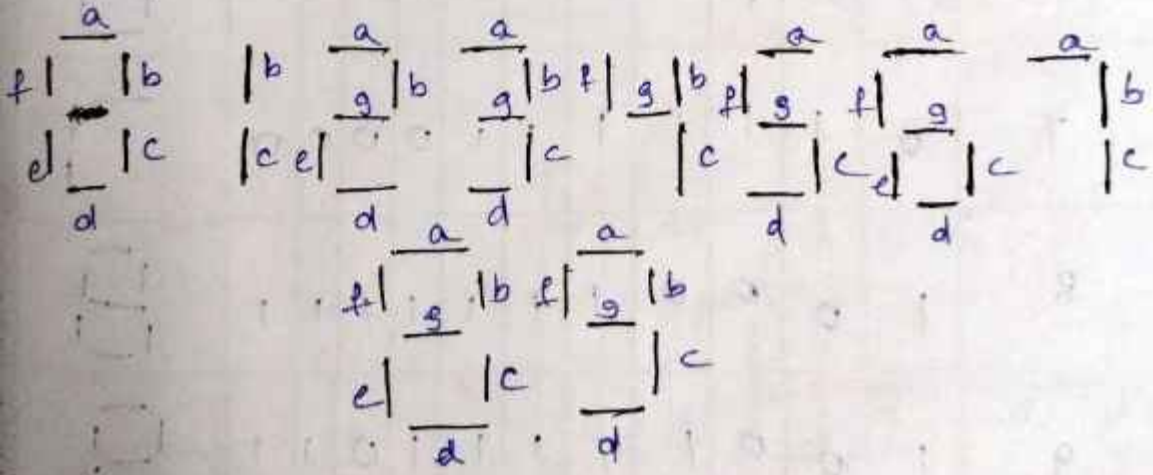


seven segment display.

* BCD to seven segment decoder is a circuit used to convert the input BCD into a form suitable for the display.

* It has four input lines (A, B, C and D) and 7 outputs lines (a, b, c, d, e, f and g) as shown in given figure.





Considering common cathode type of arrangement the truth table for the decoder can be given as in table:

Decimal Digit	Input Lines				output Lines							Display Pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

$a = F_1(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,9) + \Sigma(10,\ldots,15)$

$b = F_2(A,B,C,D) = \Sigma m(0,1,2,3,4,7,8,9) + \Sigma(10,\ldots,15)$

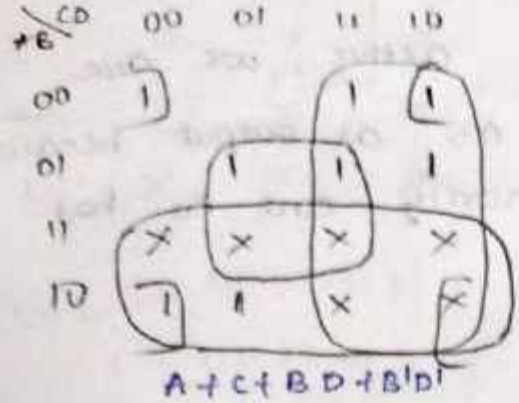
$c = F_3(A,B,C,D) = \Sigma m(0,1,3,4,5,6,7,8,9) + \Sigma(10,\ldots,15)$

$d = F_4(A,B,C,D) = \Sigma m(0,2,3,5,6,8,9) + \Sigma(10,\ldots,15)$

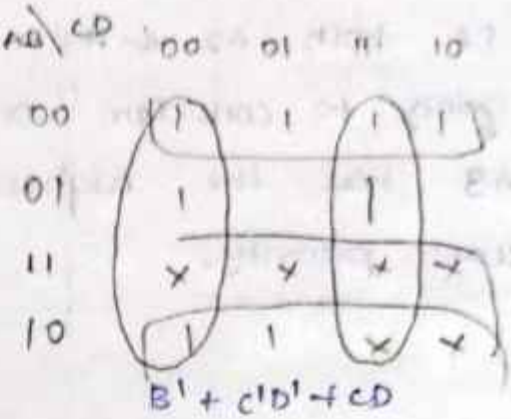
$e = F_5(A,B,C,D) = \Sigma m(0,2,6,8) + \Sigma(10,\ldots,15)$

$f = F_6(A,B,C,D) = \Sigma m(0,2,4,6,8,9) + \Sigma(10...15)$

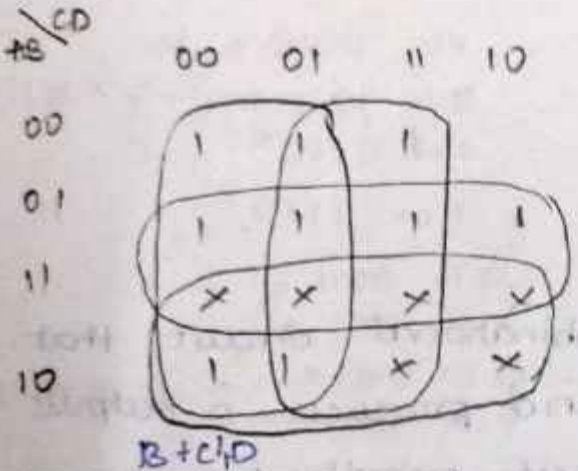
$g = F_7(A,B,C,D) = \Sigma m(2,3,4,5,6,8,9) + \Sigma(10...15)$



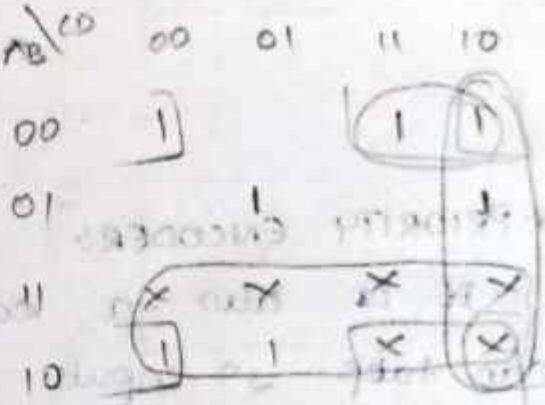
$A + C + BD + B'D'$



$B' + C'D' + CD$

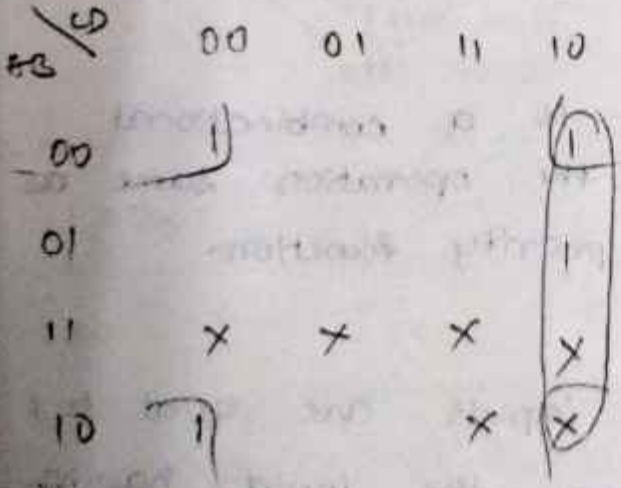


$B + C'D$

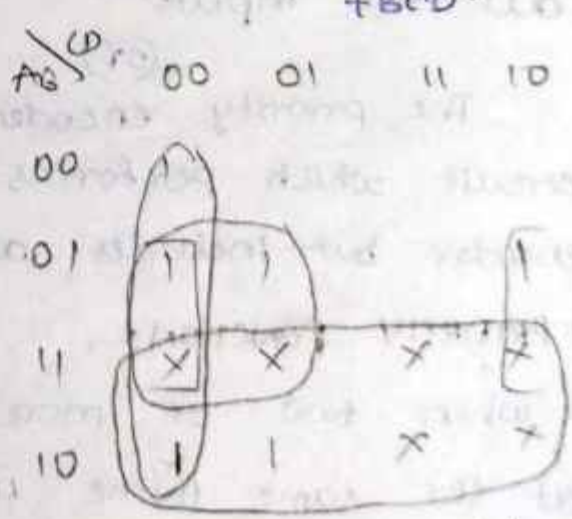


$A + B'C + B'D' + CD'$

$+ BC'D.$



$BD' + CD'$



$A + BC' + D'B + C'D'$

$A + BC' + B'C + CD'$

15/11/22

LIMITATIONS OF 4×2 ENCODER :-

1. At a time only one input should be active.
2. If both A2 & A3 are active, we are going to consider only A3 as output because A3 has the highest priority and A0 has least priority.

* PRIORITY ENCODER :-

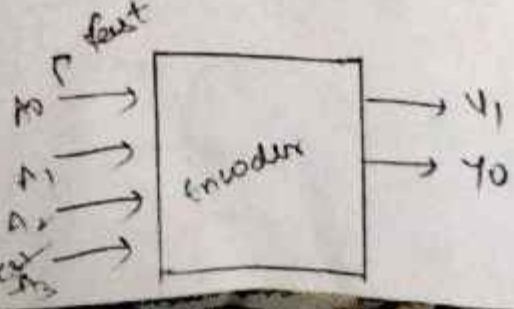
It is also a combinational circuit that can take 2^n inputs and produces n outputs and represent the highest priority input among all the inputs.

(or)

The priority encoder is a combinational circuit which performs the operation same as encoder but includes a priority function.

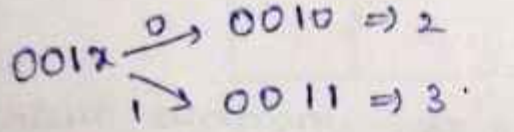
* PRIORITY FUNCTION :-

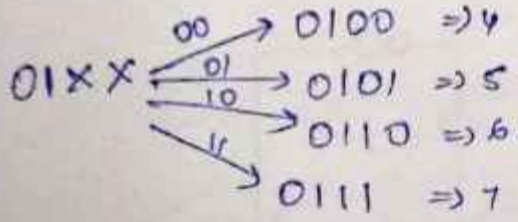
When two or more inputs are equal to 1 at the same time then the input having the highest subscript will be given the precedence or priority.

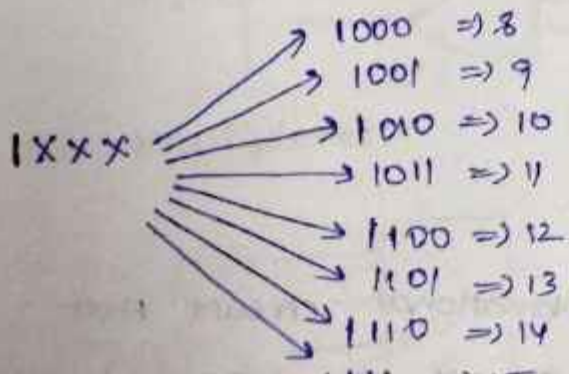


Truth table :-

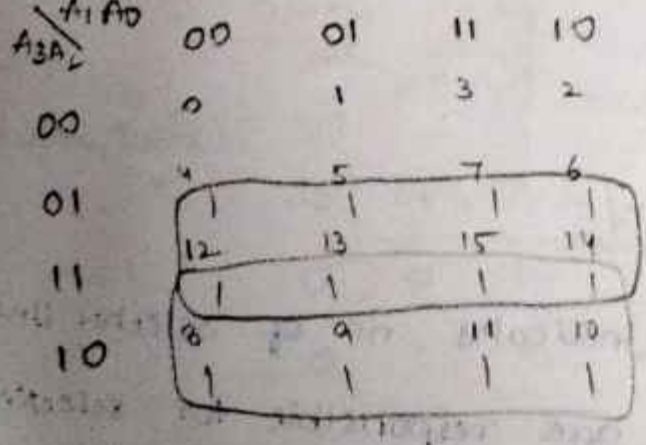
A_3	A_2	A_1	A_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1



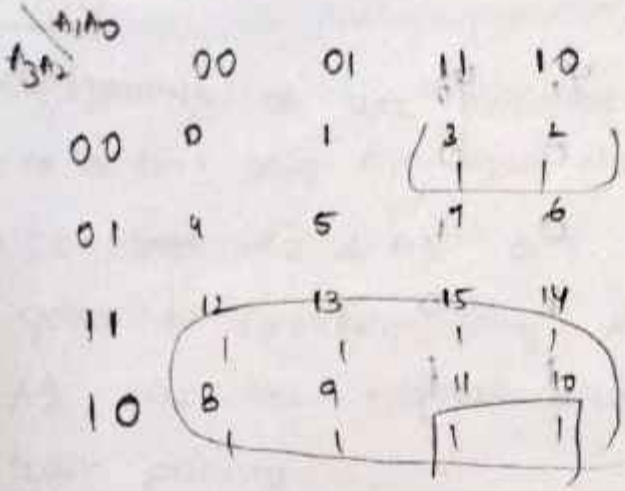




k-map :-

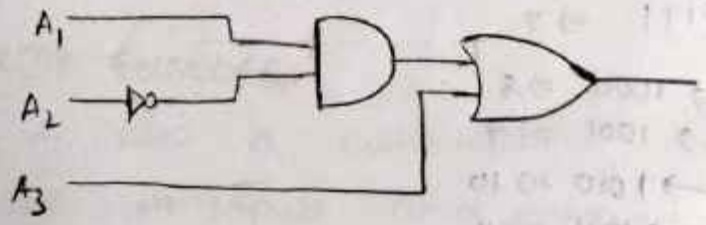


$Y_1 = A_2 + A_3$.



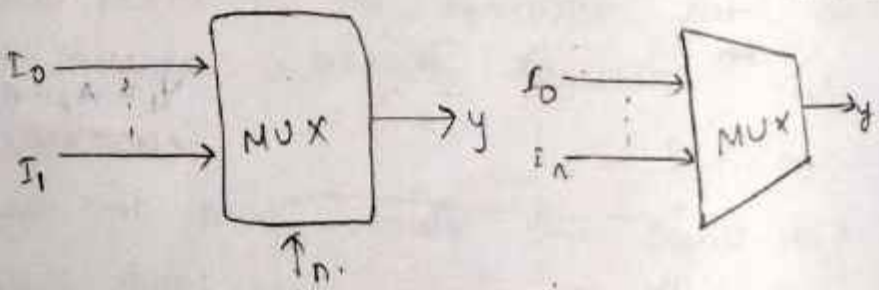
$A_3 + A_2' A_1$

LOGIC GATE :-



16/11/22 :-

Multiplexer :- It is a combinational circuit that selects single output from a set of many inputs. It is also called as MUX (or) data selector.



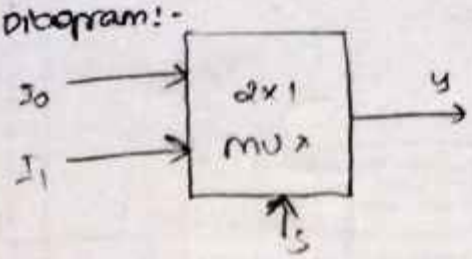
2^n inputs where n indicates no of selector lines

* Selector lines :- These are responsible for selecting the input to be given as the output.

2 x 1 multiplexer :-

2 inputs, 1 output.

Block Diagram:-

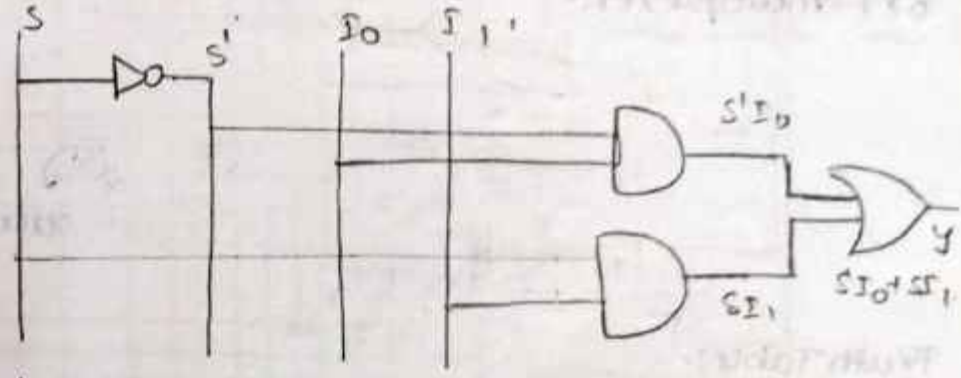


Truth table:-

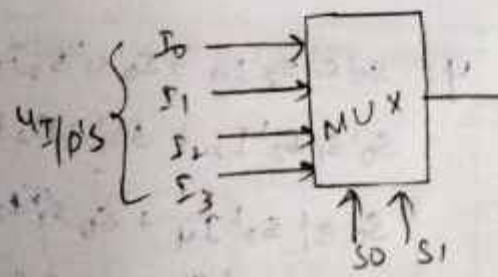
S	Y
0	I_0
1	I_1

$Y = S'I_0 + SI_1$

Circuit diagram:-



4 x 1 Multiplexer:-



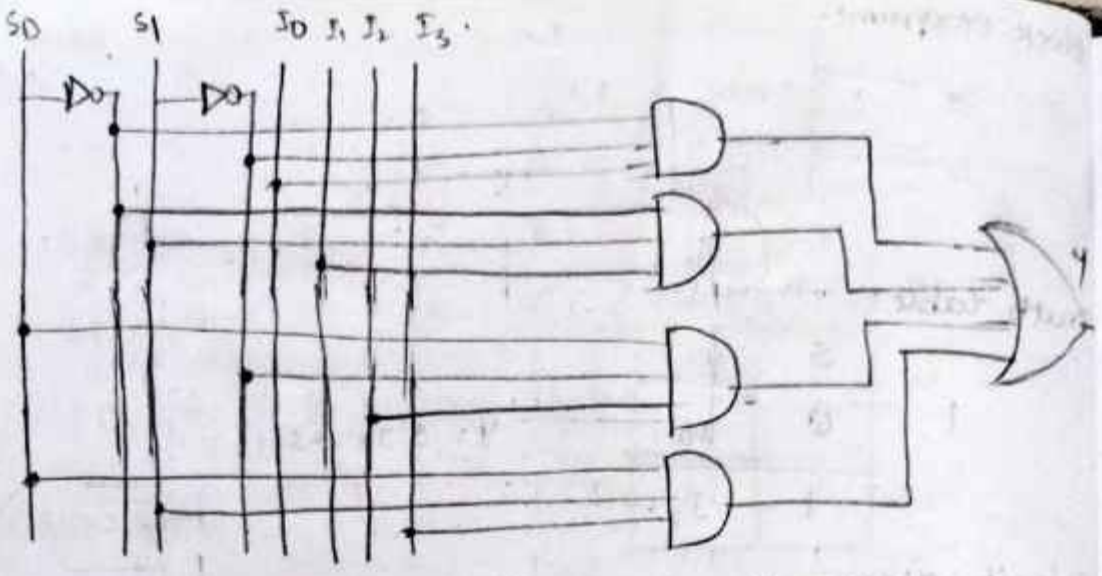
2 (2) selection lines.

Selection lines: (0) (0), (0) (1)

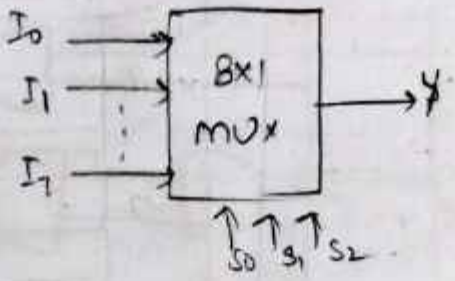
Truth Table:-

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

$Y = S_0'S_1'I_0 + S_0'S_1I_1 + S_0S_1'I_2 + S_0S_1I_3$



8x1 multiplexer:-

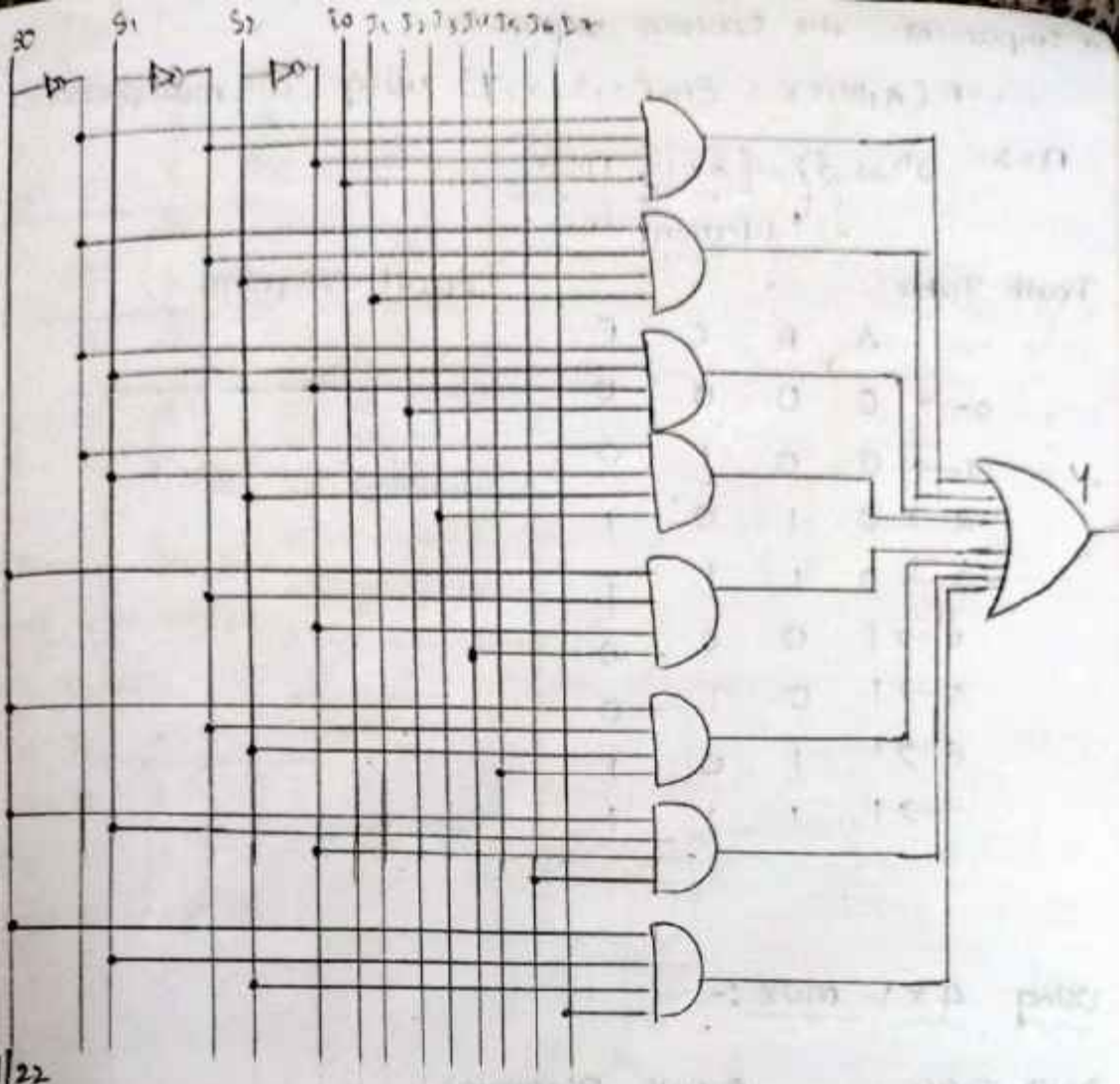


3 selection lines

Truth Table:-

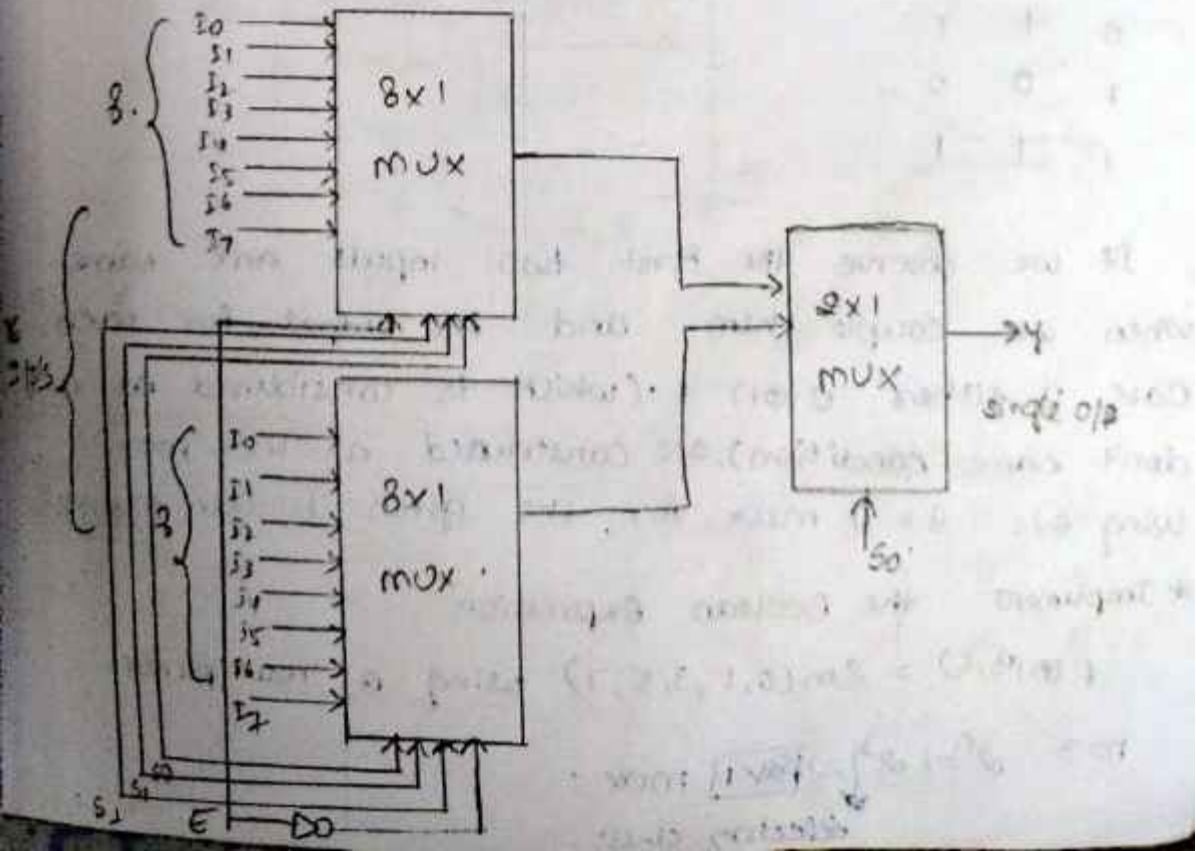
S_0	S_1	S_2	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

$$Y = S_0'S_1'S_2'I_0 + S_0'S_1'S_2I_1 + S_0'S_1S_2'I_2 + S_0'S_1S_2I_3 + S_0S_1'S_2'I_4 + S_0S_1'S_2I_5 + S_0S_1S_2'I_6 + S_0S_1S_2I_7$$



17/11/22

1. Construct 16x1 mux with => 2 8x1 mux
=> 1 2x1 mux.



* Implement the Boolean expression.

$F(A,B,C) = \Sigma m(2,3,6,7)$ using a multiplexer.

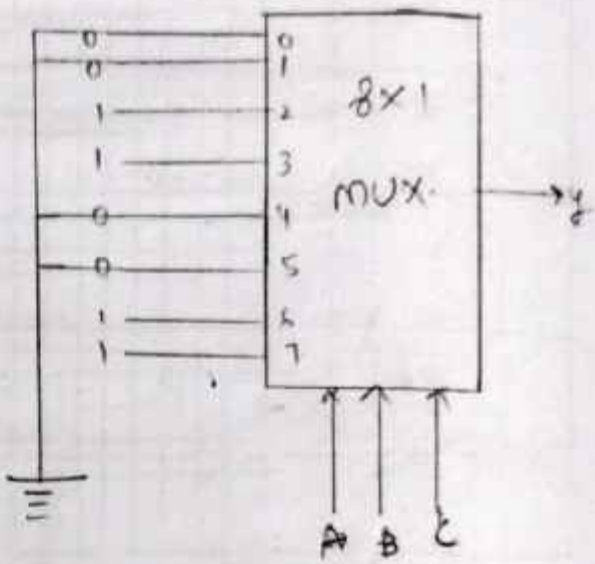
$n=3$. $2^n \Rightarrow 2^3 \Rightarrow$ 8x1 mux

↳ selection lines.

Truth Table:-

	A	B	C	F
0→	0	0	0	0
1→	0	0	1	0
2→	0	1	0	1
3→	0	1	1	1
4→	1	0	0	0
5→	1	0	1	0
6→	1	1	0	1
7→	1	1	1	1

Circuit Diagram:-

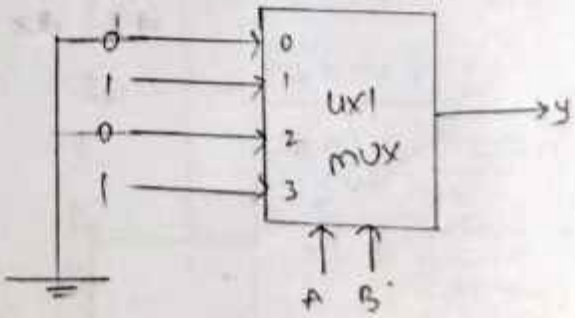


Using 4x1 mux:-

Truth Table:-

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

Circuit Diagram:-



If we observe the first two inputs are same when we couple them and the output for each case is either 0 or 1. (which is considered as a don't care condition). We constructed a 4x1 mux using this 8x1 mux for the given boolean expression.

* Implement the Boolean Expression.

$F(A,B,C) = \Sigma m(0,1,3,5,7)$ using a multiplexer.

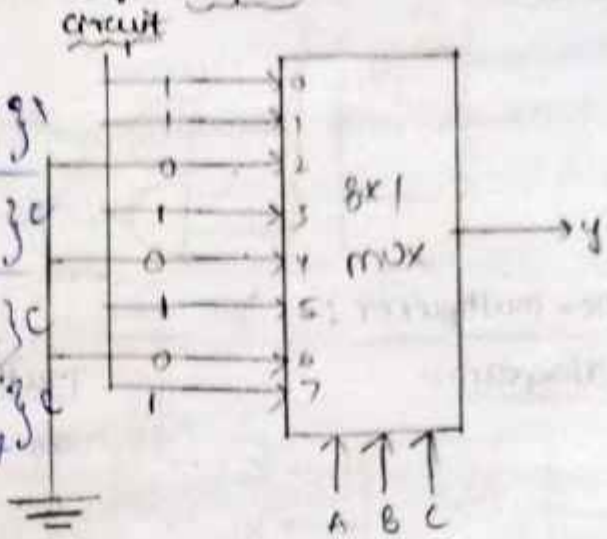
$n=3$ $2^n \Rightarrow 2^3 \Rightarrow$ 8x1 mux.

↳ selection lines.

Truth Table :-

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

Simple Diagram:-

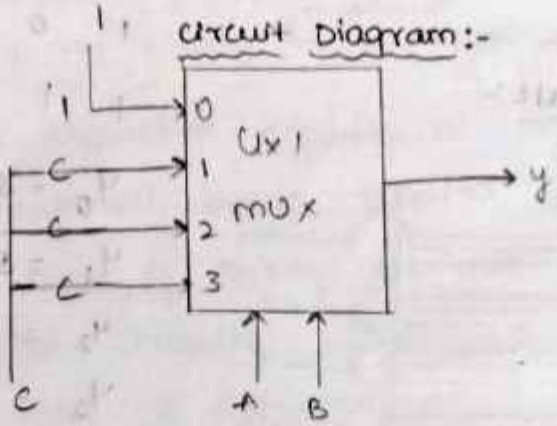


Using 4x1 mux :-

Truth Table :-

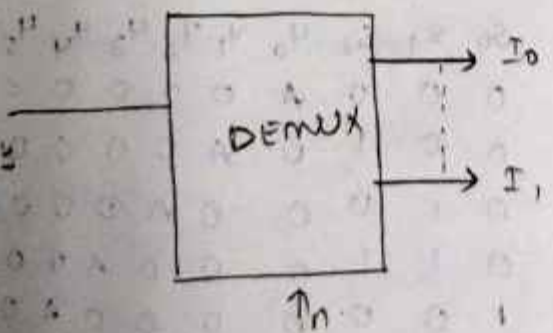
A	B	F
0	0	1
0	1	C
1	0	C
1	1	C

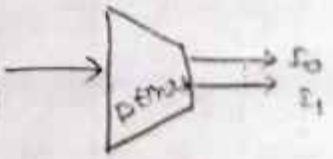
Circuit Diagram :-



* DEMULTIPLEXER :-

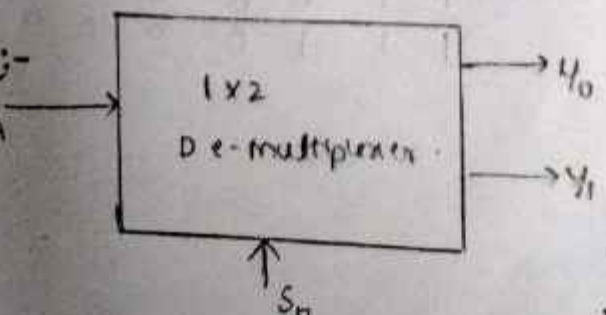
A de-multiplexer is a combinational circuit that has 1 input and 2^n outputs.





1x2 Demultiplexer :-

Block Diagram:-



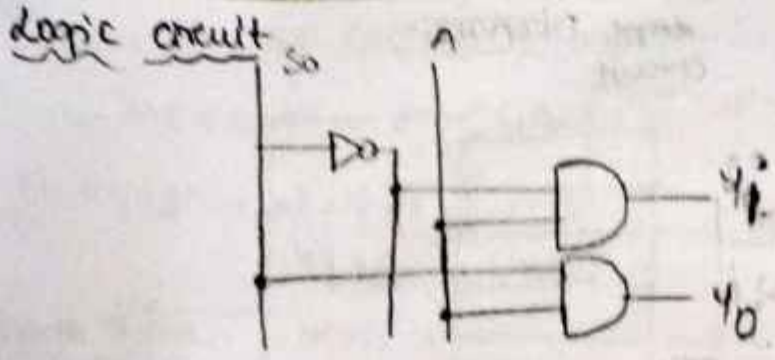
Truth Table :-

S_0	Y_1	Y_0
0	0	A
1	A	0

$Y_0 = S_0' A$

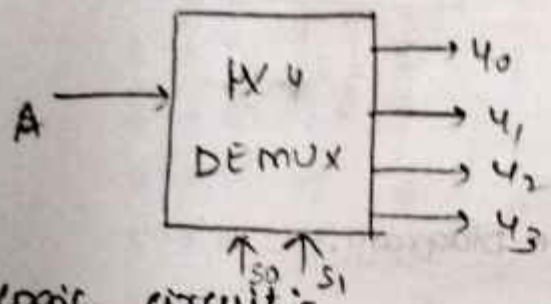
$Y_1 = S_0 A$

Logic circuit



1X4 De-multiplexer :-

Block diagram:-



Truth Table:-

S_0	S_1	Y_0	Y_1	Y_2	Y_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

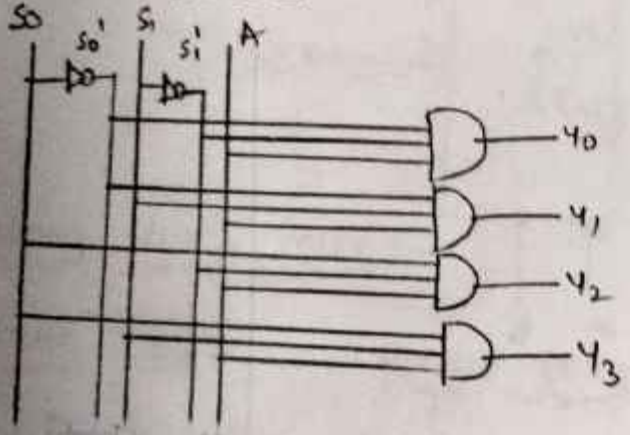
$Y_0 = S_0' S_1' A$

$Y_1 = S_0' S_1 A$

$Y_2 = S_0 S_1' A$

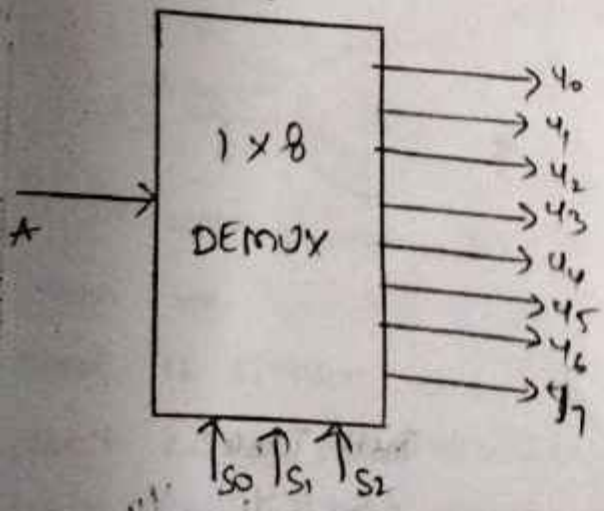
$Y_3 = S_0 S_1 A$

Logic circuit:-



1x8 De-multiplexer :-

Block diagram:-



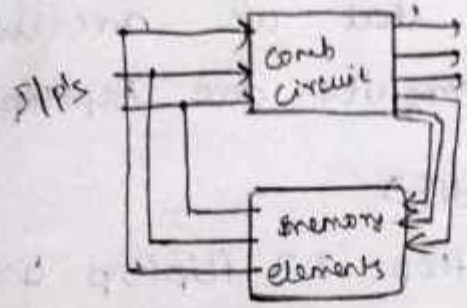
Truth Table:-

S_0	S_1	S_2	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	A	0	0	0	0	0	0	0
0	0	1	0	A	0	0	0	0	0	0
0	1	0	0	0	A	0	0	0	0	0
0	1	1	0	0	0	A	0	0	0	0
1	0	0	0	0	0	0	A	0	0	0
1	0	1	0	0	0	0	0	A	0	0
1	1	0	0	0	0	0	0	0	A	0
1	1	1	0	0	0	0	0	0	0	A

21/11/22

4. SEQUENTIAL CIRCUITS.

* A sequential circuit consists of a combinational circuit to which storage elements are connected to form a feedback path.



* These storage elements or) memory elements are capable of storing the binary information.
* The state of a sequential circuit at any point of time depends on i) External inputs applied on that time. ii) Present state of storage elements.
* Types of sequential circuits
1. Asychronous and 2. Synchronous.

ASYNCHRONOUS:-

* In asynchronous sequential circuits, the signals are given at any instance of time, the storage elements that are used in Asynchronous circuits are latches.
* Latches are also called as level sensitive devices.
* Latches are made of logic gates.
* Latches are 2 types. 1. SR Latch (Set, Reset) 2. D latch (Data / Delay)
* Latches were the basic block of flipflops
*

SYNCHRONOUS :-

* In this sequential circuit we use clock pulse to control the storage elements are called clocked (or) edge sequential circuits also called as edge sensitive devices.

* The storage elements that we are using in synchronous sequential circuits are flipflops.

* Flipflops are of 4 types

1. SR flipflop 2. JK flipflop 3. T flipflop 4. D flipflop.
Set Reset Jack Kilby Toggle Delay (or) Data.

* DIFFERENCE BETWEEN LATCHES AND FLIPFLOPS.

LATCH	FLIP-FLOP.
1. Latches do not require clock signal.	1. Flip flops have clock signals.
2. A latch is an asynchronous device.	2. A flip-flop is a synchronous device.
3. Latches are transparent devices i.e when they are enabled, the output changes immediately if the input changes.	3. A transition from Low to high or high to low of the clock signal will cause the flip-flop to either change its output or retain it depending on the input signal.
4. A Latch is a level sensitive device (Levels triggering involved).	4. A flip-flop is an edge sensitive device (edge triggering involved).
5. The operation of a latch is faster as they do not have to wait for any clock signal.	5. Flip-flops are comparatively slower than latches due to clock signal.

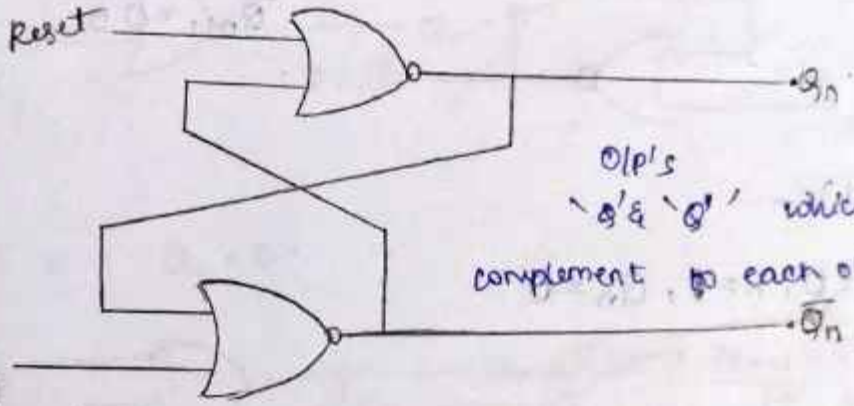
* SR LATCH (Set-Reset Latch):-

It is a circuit, with "2 cross coupled NOR Gates" (or) "2 cross coupled NAND Gates".

NOR Gates:-

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

Logic circuit for SR Gate:-



I/p's Set & Reset

O/p's 'Q' & 'Q̄' which are complement to each other.

S	R	Q_n	Q_{n+1}	
0	0	0	0	Q_n
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	✗	Indetermined state.
1	1	1	✗	

Case (1):- $S=0, R=0$

$Q_{n+1} = (0+Q_n')' = (Q_n')' = Q_n$

Case (2):- $S=0, R=1$ (Reset state).

$Q_{n+1} = (1+Q_n')' = 0$

Case (3):- (Set state) $S=1, R=0$

$Q_{n+1} = (0+Q_n)' = Q_n' = 0$

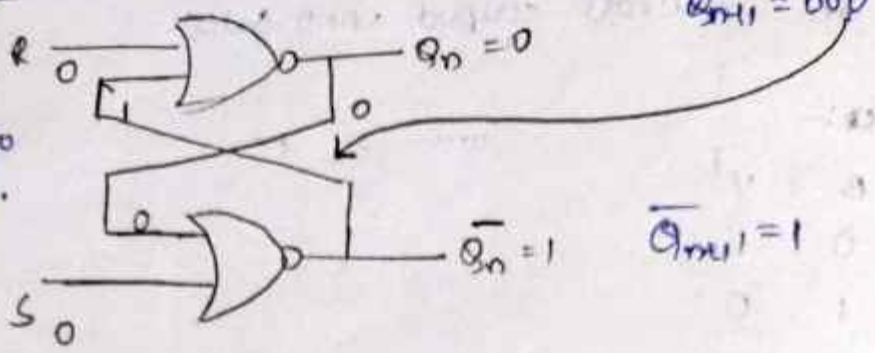
$Q_{n+1}' = 1+Q_n = 1.$

Case (4):- $S=1, R=1$

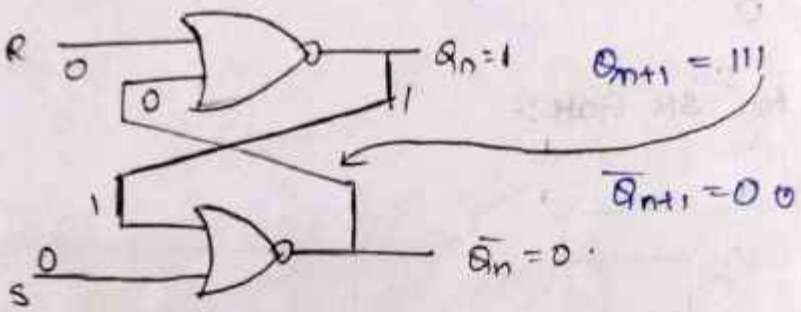
$Q_{n+1} = (1+Q_n)' = (1)' = 0$ ✗

22/11/22

Case (i) :-

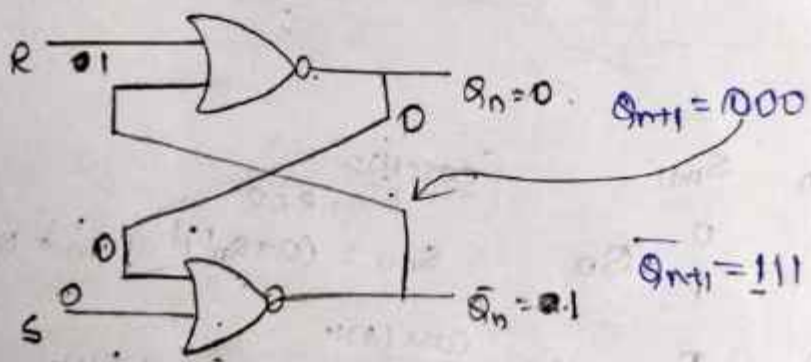
a) 

b) S=0, R=0, Q_n=1

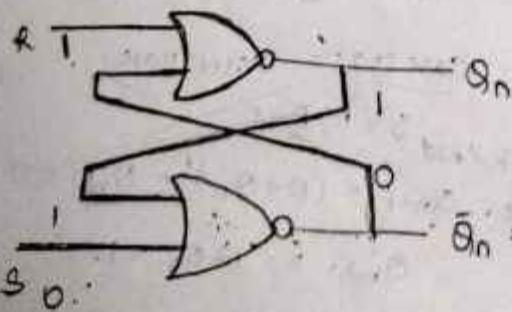


Case (ii) :-

a) S=0, R=1, Q_n=0

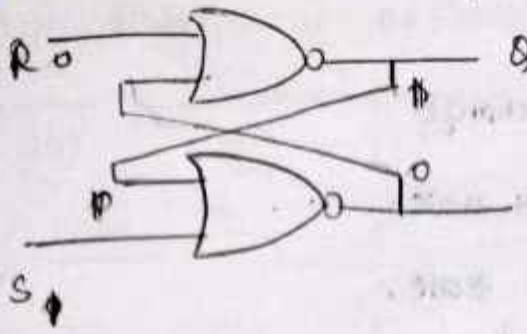


b) S=0, R=1, Q_n=1



case-(iii):- $S=1$ $R=0$ $Q_n=0$.

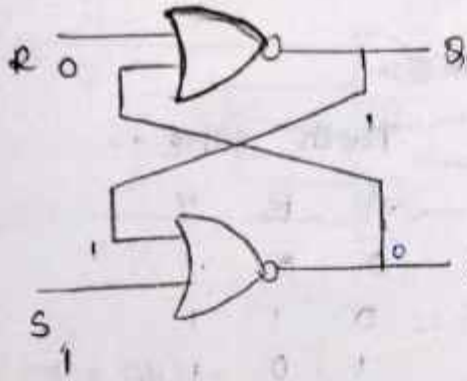
a)



$Q_{n+1} = 111$

$\overline{Q_{n+1}} = 00$

b) $S=1$, $R=0$ $Q_n=1$

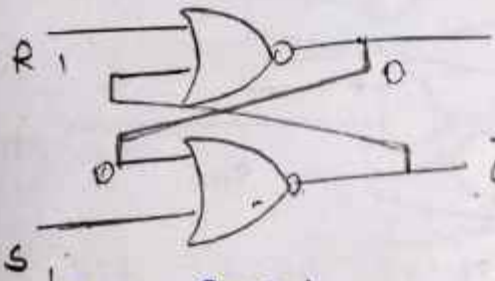


$Q_{n+1} = 111$

$\overline{Q_{n+1}} = 000$.

case-(iv)

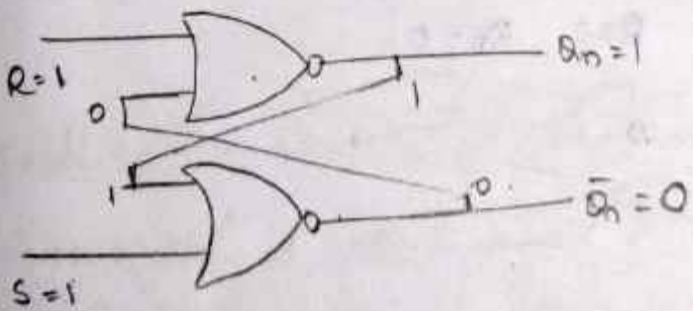
a) $S=1$ $R=1$ $Q_n=0$.



$Q_{n+1} = 0$

$\overline{Q_{n+1}} = 0$

b) $S=1$, $R=1$ $Q_n = 1$



$Q_{n+1} = 000$

$\overline{Q_{n+1}} = 00$

Truth Table :-

S	R	Q_n	Q_{n+1}
0	0	x	Q_n → No change
0	1	x	0 → Reset state.
1	0	x	1 → Set state.
1	1	x	Invalid state.

23/11/22

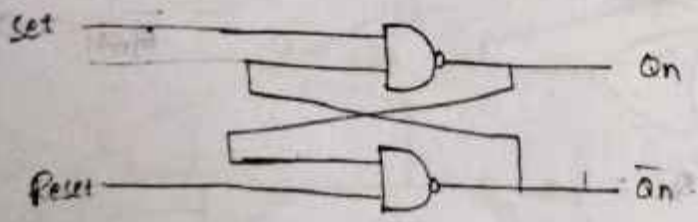
SR LATCH USING NAND GATE :-

S	R	Q_n	Q_{n+1}
0	0	0	I.D
0	0	1	I.D
0	1	0	1
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Truth table :-

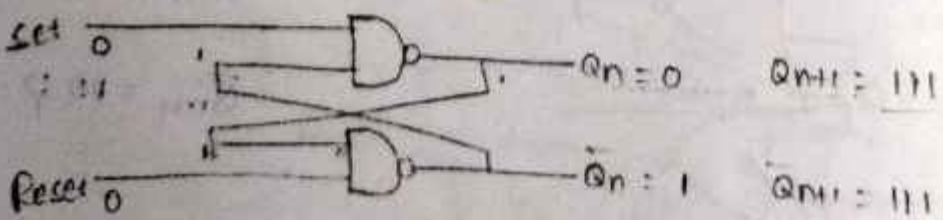
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

* Logic circuit :-

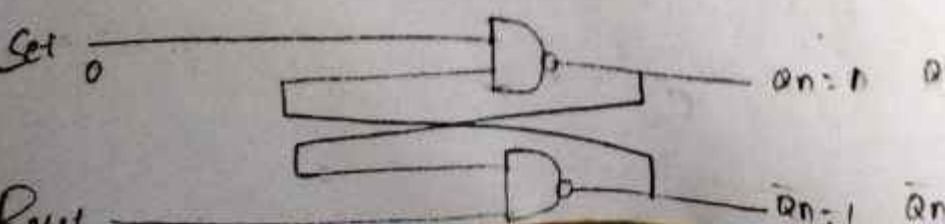


* Case (i).

(a) :- S:0, R:0, Q_n:0.

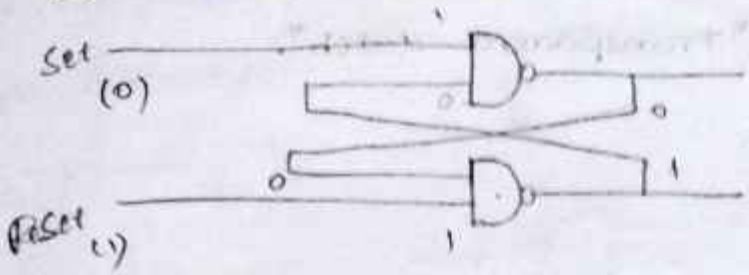


(b). S:0, R:0, Q_n:1.

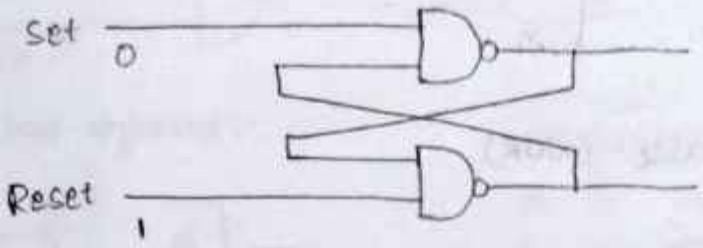


* case (ii):-

(a):- R=~~0~~1, S=0, Qn=0.

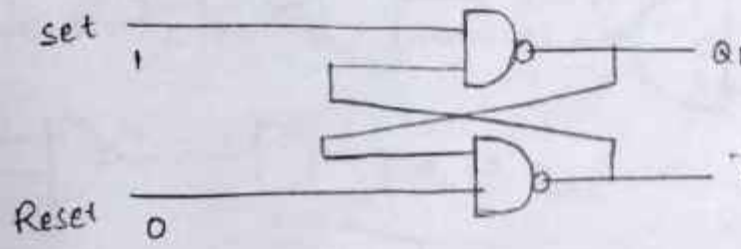


(b):- S=0, R=1, Qn=1.

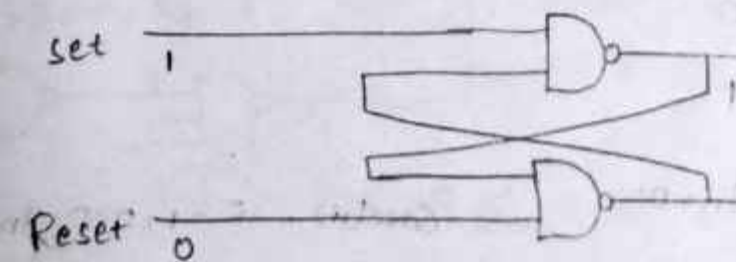


* case (iii):-

(a):- S=1, R=0, Qn=0.



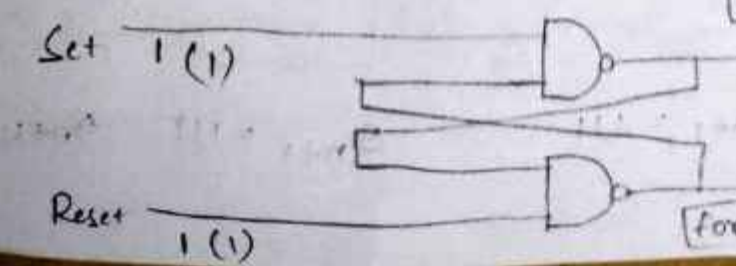
(b):- S=1, R=0, Qn=1.



* case (iv):-

(a):- S=1, R=1, Qn=0. [(b):- S=1, R=1, Qn=1]

[for b:- Qn=1 Qn+1=11]



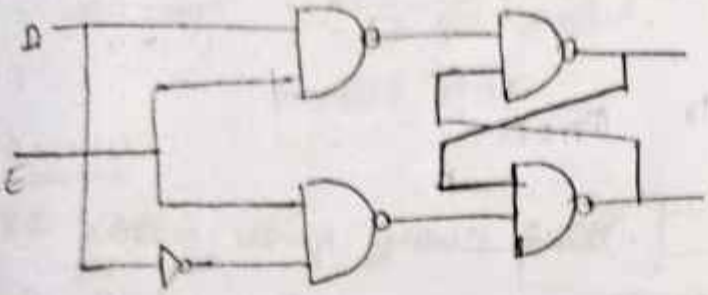
[for b:- Qn=0 Qn+1=00]

D Latch:-

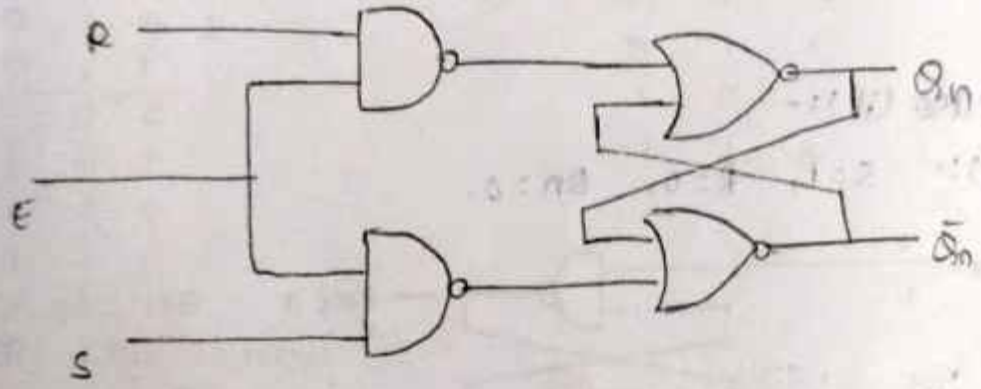
→ Data Latch

→ Also called as "Transparent Latch".

Logic circuit:-



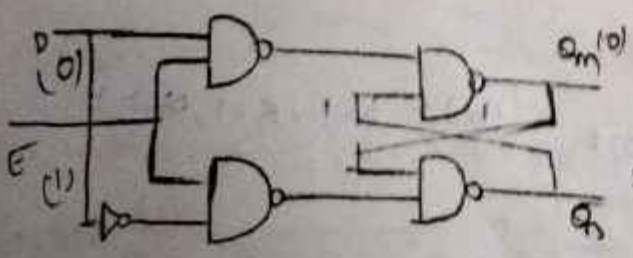
SR Latch using enable. (NOR)



For D-Latch:-

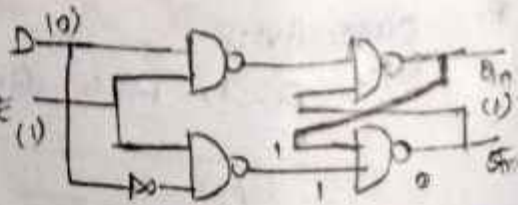
E	D	Q_n	Q_{n+1}
0	x	—	—
1	0	0	0
1	1	1	1

Case(i) - E=1, D=0, Q_n=0.



Q_{n+1} = 0000

Case(ii) - E=1, D=0, Q_n=1.



Q_{n+1} = 111

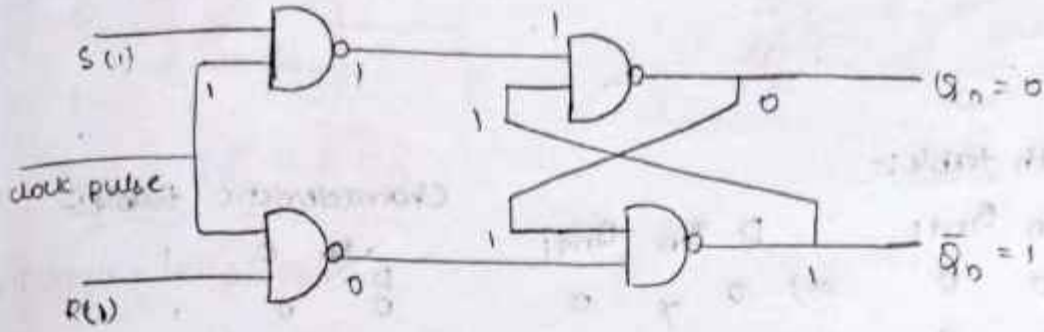
Q_{n+1} = 111

Q_{n+1} = 00.

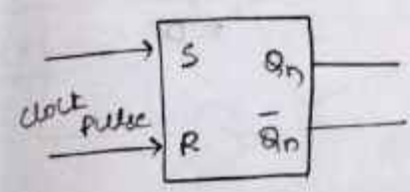
* FLIP-FLOP (combination of latch & clock pulse):-

1. SR flip-flop:-

Logic circuit:-



Graphical symbol:-

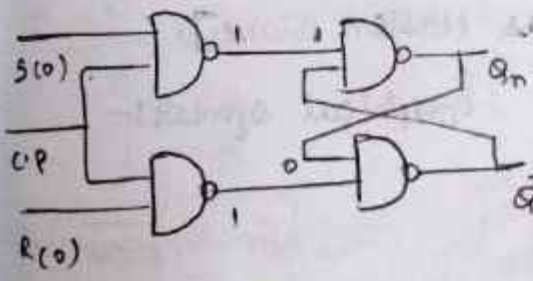


Truth table:-

S	R	Q_n	Q_{n+1}	
0	0	0	0	
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	forbidden state.
1	1	1	X	

* case (i):-

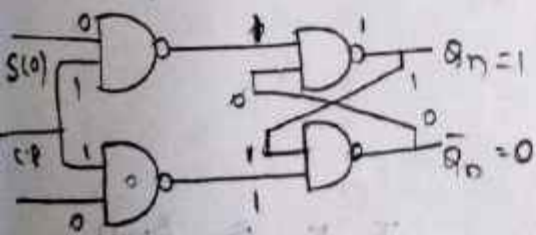
a) $S=0, R=0, Q_n=0$.



$Q_{n+1} = 00$

$\bar{Q}_{n+1} = 11$

b) $S=0, R=0, Q_n=1$



$\bar{Q}_{n+1} = 00$

* case (ii)

$Q_{n+1} = 11$ $S=0$, $R=1$, $Q_n=0$.

$Q_{n+1} = 0$.

Excitation table:-

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

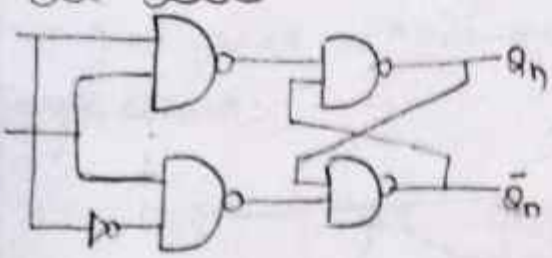
RQ) Characteristic equation

R \ $S Q_n$	00	01	11	10
0	0	1	3	2
1	4	5	7	6
	1	1	X	X

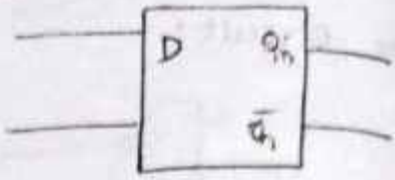
$= S + \bar{R} Q_n$

* D - flipflop:-

Logic circuit:-



Graphical symbol:-



* Truth table:-

D	Q_n	Q_{n+1}
0	0	0
1	0	0
0	1	1
1	1	1

$\Rightarrow$

D	Q_n	Q_{n+1}
0	x	0
1	x	1

Characteristic table:-

D \ Q_n	0	1
0	0	0
1	1	1

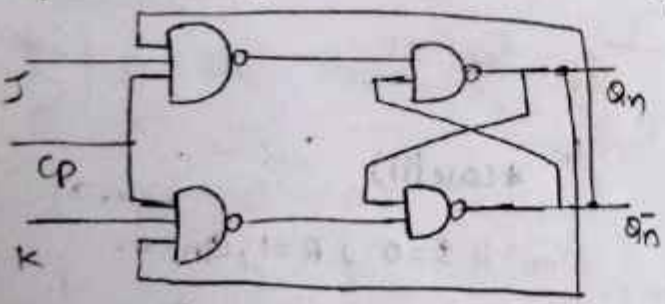
$= D$

Excitation table:-

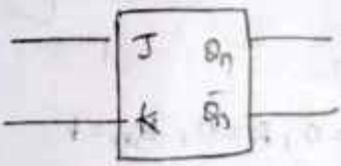
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

* J-K - flipflop:- [Also called as Master Slave].

Logic circuit:-



Graphical symbol:-



Truth table:-

J	K	Q_n	Q_{n+1}	
0	0	0	0	(No change)
0	0	1	1	
0	1	0	0	(Reset)
0	1	1	0	
1	0	0	1	(Set)
1	0	1	1	
1	1	0	↑	$\bar{Q}_n$ toggle state.
1	1	1	0	

$\Rightarrow$

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

Characteristic equation:-

$Q_{n+1} = \Sigma(1,4,5,6)$

J \ kQn	00	01	11	10
0	0	1	3	2
1	4	5	7	6

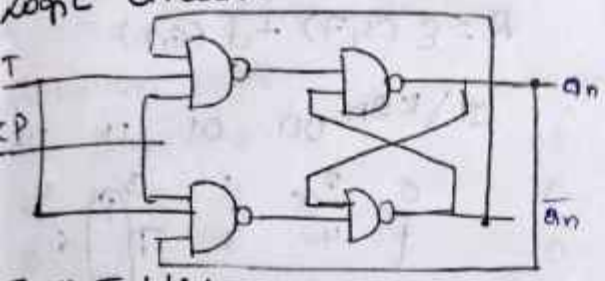
$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

Excitation Table:-

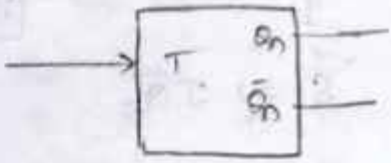
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

* Toggle - flipflop:-

Logic circuit:-



Graphical symbol:-



Truth Table:-

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

⇒

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

Characteristic table:-

T \ Q_n	0	1
0	0	1
1	2	3

⇒ $T\bar{Q}_n + \bar{T}Q_n$

Excitation Table:-

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Flip Flop	Characteristic Equation
SR	$S + \bar{R}Q_n$
JK	$J\bar{Q}_n + \bar{K}Q_n$
T	$T\bar{Q}_n + \bar{T}Q_n$
D	D

Conversions:-

1. SR to JK:-

* SR to other flip flops.

Excitation Table:-

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

Truth Table for J-K:-

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

Characteristic equation:-

$S = \Sigma(4,6) + d(1,5)$

J \ KQn	00	01	11	10
0	0	1	3	2
		X		
1	4	5	7	6
	1	X		1

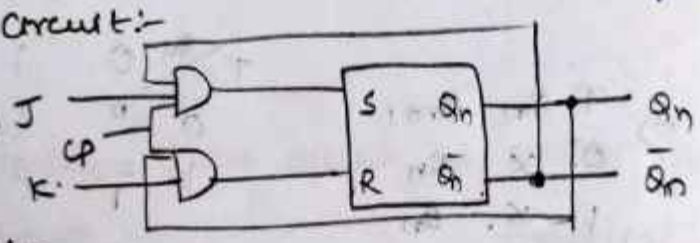
$S \Rightarrow J\overline{Q_n}$

$R = \Sigma(3,7) + d(0,2)$

J \ KQn	00	01	11	10
0	0	X 1	3	2 X
1	4	5	7	6
			1	

$R = KQ_n$

Logic Circuit:-



2-SR to D:-

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

Truth Table for D:-

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

Characteristic equation:-

$S = \Sigma(2) + d(3)$

D \ Qn	0	1
0	0	1
1	2	3 X

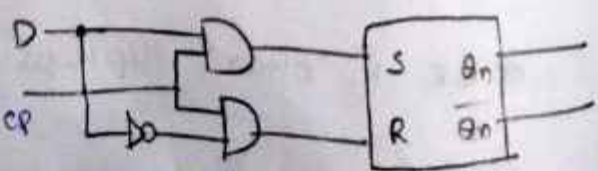
$S = D$

$R = \Sigma(1) + d(0)$

D \ Qn	0	1
0	0 X	1
1	2	3

$R = \overline{D}$

Circuit:-



3. SR to T:-

Excitation Table:-

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

T-T for T:-

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

Characteristic equations:-

$S = \Sigma(2) + d(1)$

T \ Q_n	0	1
0	0	1 (2)
1	1 (1)	X (3)

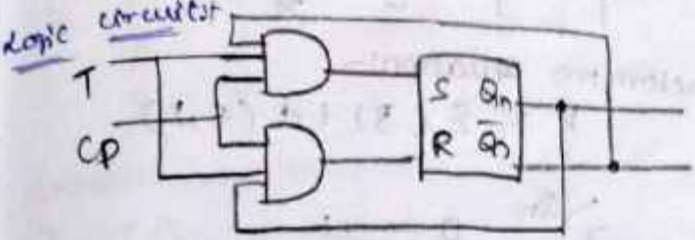
$S = T\bar{Q}_n$

$R = \Sigma(3) + d(0)$

T \ Q_n	0	1
0	X (0)	1 (2)
1	1 (1)	X (3)

$R = TQ_n$

Logic circuit:-



28/11/22:-

JK to other flip flops:- 1. JK to SR.

Excitation table:-

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

Truth table for SR:-

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

Characteristic equation:-

$J = \Sigma(4) + d(1,3,5,6,7)$

S \ Q_n	00	01	11	10
0	0	1	3 X	2
1	4 1	5 X	7 X	6 X

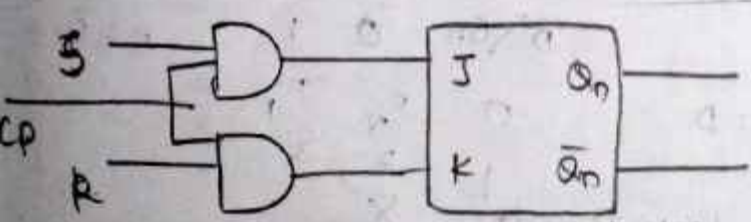
$J = S$

$K = \Sigma(3) + d(0,2,4,6,7)$

S \ Q_n	00	01	11	10
0	0 X	1	3 1	2 X
1	4 X	5	7 X	6 X

$K = R$

Logic Diagram:-



2. JK to T :-

Excitation Table :-

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

Truth Table for T :-

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

Characteristic equation :-

$J = \Sigma(2) + d(1,3)$

T \ Q_n	0	1
0	0	1
1	2 1	3 X

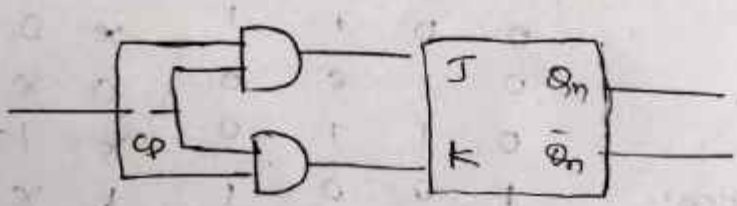
$J = T$

$K = \Sigma(3) + d(0,2)$

T \ Q_n	0	1
0	0	X
1	2 X	3 1

$K = T$

Logic circuit :-



3. JK to D :-

Excitation Table :-

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

Truth Table for D :-

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

Characteristic Equation :-

$J = \Sigma(2) + d(1,3)$

D \ Q_n	0	1
0	0	1
1	2 1	3 X

$J = D$

$K = \Sigma(1) + d(0,2)$

D \ Q_n	0	1
0	0 X	1 1
1	2 X	3 X

$K = \bar{D}$

Logic circuit :-

T to other flip-flop conversion:-

1. T to SR:-

Excitation table:-

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table for SR:-

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

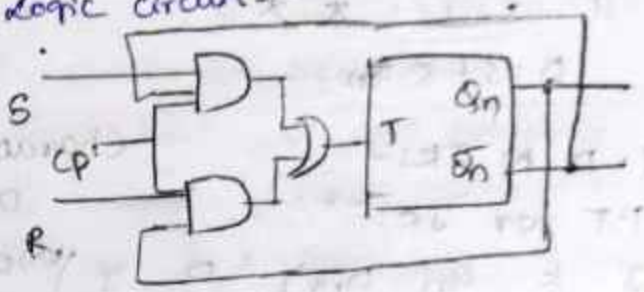
Characteristic equation:-

$T = \Sigma(3,4) + d(6,7)$

S \ R Q_n	00	01	11	10
0	0	1	3 (1)	2
1	4	5	7 (1, X)	6 (X)

$T = S\bar{Q}_n + RQ_n$

Logic circuit:-



2. T to JK:-

Truth Table for JK:-

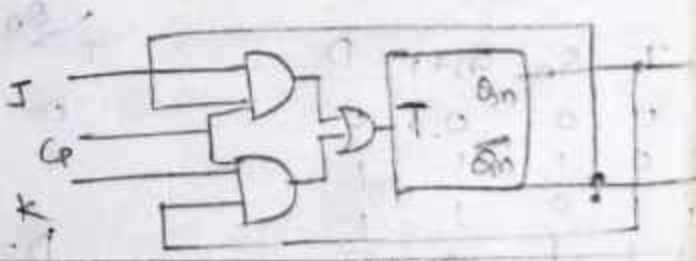
J	K	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Characteristic equation:-

$T = \Sigma(1,4,6,7)$.

J \ K Q_n	00	01	11	10
0	0	1	3 (1)	2
1	4	5	7 (1)	6 (1)

$T = J\bar{Q}_n + KQ_n$



3. T to D:-

T.T for D:-

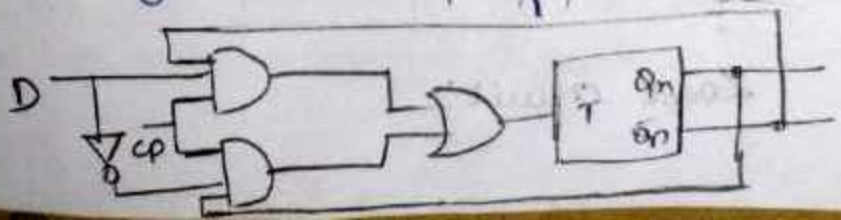
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Characteristic equation.

D \ Q_n	0	1
0	0	1 (1)
1	2	3 (1)

$T = \Sigma(1,2)$

$T = D\bar{Q}_n + \bar{D}Q_n$



D to other flipflops:-

1. D to SR.

Excitation Table:-

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

T.T for SR:-

S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	x	x
1	1	1	x	x

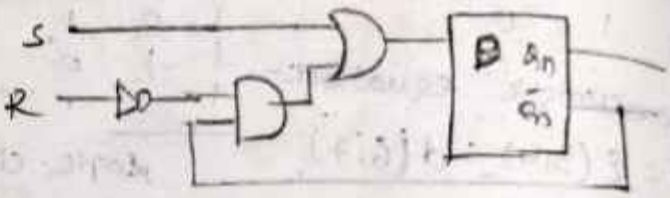
Characteristic eqn:-

$\Sigma(1,4,5) + d(6,7)$

S \ RQn	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$D = S + \bar{R} Q_n$

Logic diagram:-



---

2. D to JK:-

T.T for JK:-

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

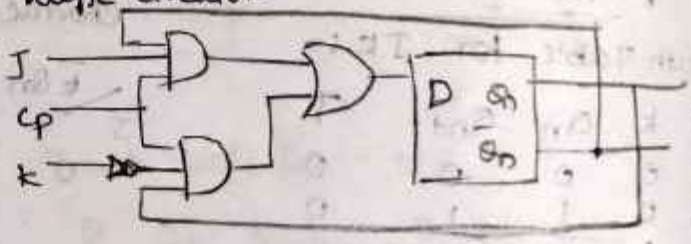
Characteristic equation:-

$D = \Sigma(1,4,5,6)$

J \ KQn	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$D = \bar{K} Q_n + J \bar{Q}_n$

Logic circuit:-



---

3. D to T:-

Truth table for T:-

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

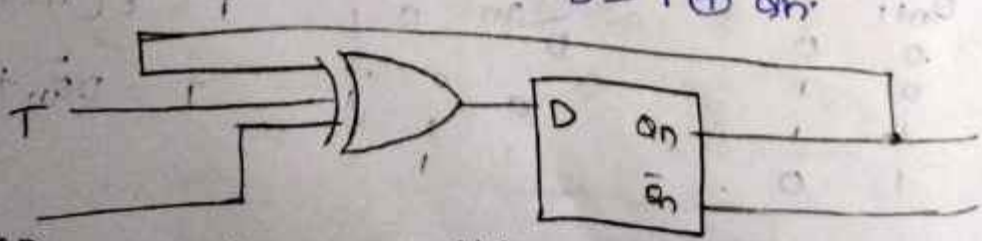
Characteristic equation:-

T \ Qn	0	1
0	0	1
1	2	3

$D = \Sigma(1,2)$

$D = T\bar{Q}_n + \bar{T} Q_n$

$D = T \oplus Q_n$



Logic circuit:-