1 0 1 0 1 1 1 0

Most significant Bit (MSB) — least significant bit (LSB)

Yubraj Devkota (YD)

PAGE NO.:
DATE:

Nature made → velocity, temp, ... → analog — value vs t

Human made → computer → digital — discrete.

NUMBER FORMAT:

1. Binary — base 2 — 0, 1 — eg: 1011101
2. Octal — base 8 — 0, 1, . . ., 7 — eg: 5210367
3. Decimal — base 10 — 0, 1, - - - , 9 — eg: 19625
4. Hexadecimal — base 16 — 0, 1, . . 9, A, B, C, D, E, F — eg 21AF7

A. Conversion from decimal to other formats:

$(12345)_{10}$

→ Binary → $(?)_2$ = $(11000000111001)_2$

2	12345	→ 1
2	6172	→ 0
2	3086	→ 0
2	1543	→ 1
2	771	→ 1
2	385	→ 1
2	192	→ 0
2	96	→ 0
2	48	→ 0
2	24	→ 0
2	12	→ 0
2	6	→ 0
2	3	→ 1
2	1	→ 1
	0	

(read remainders from bottom to top ↑)

$(12345)_{10} \rightarrow (?)_8 = 30071$

8	12345	→	1
8	1543	→	7
8	192	→	0
8	24	→	0
	3		

$(12345)_{10} \rightarrow (?)_{16} = (3039)_{16}$

16	12345	9
16	771	3
	3	

Conversion from binary to other formats.

$(1100000011100 1)_2 = (?)_{10} = (?)_8 = (?)_{16}$

$\rightarrow (?)_{10} = 1\times2^{13} + 1\times2^{12} + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^0$
$= 8192 + 4096 + 32 + 16 + 8 + 1$
$= (12345)_{10}$

$\rightarrow (?)_8 \quad (011\ 000\ 000\ 111\ 001)_2 = (30071)_8$

$\rightarrow (?)_{16} \quad (0011\ 0000\ 0011\ 1001)_2 = (3039)_{16}$
3 0 3 9

decimal.

PAGE NO. :
DATE :

binary → octal

x	y	z	
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

w	x	y	z	
0	0	0	0	– 0
0	0	0	1	– 1
0	0	1	0	– 2
0	0	1	1	– 3
0	1	0	0	– 4
0	1	0	1	– 5
0	1	1	0	– 6
0	1	1	1	– 7
1	0	0	0	– 8
1	0	0	1	– 9
1	0	1	0	– A
1	0	1	1	– B
1	1	0	0	– C
1	1	0	1	– D
1	1	1	0	– E
1	1	1	1	– F

$(101101.1011)_2 = (?)_{10} = (45.6875)_{10}$

$\Rightarrow 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 . 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 32 + 0 + 8 + 4 + 0 + 1 . \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}$

$= (45.6875)_{10}$

2	45	→ 1
2	22	→ 0
2	11	→ 1
2	5	→ 1
2	2	→ 0
	1	

↑

10 1101

$$0.6875 \times 2 = 1.3750$$
$$0.3750 \times 2 = 0.7500$$
$$0.7500 \times 2 = 1.5000$$
$$0.5000 \times 2 = 1.0000$$

$\Rightarrow (1011)_2$

$(101101.1011)_2$

COMPLEMENTS.

- used for simplification of substraction and logical manipulations.
- 2 types: $(r-1)$'s complement; r's complement.

$$10 - 2 = 8$$

For decimal; $\Rightarrow r = 10$.
↳ 9's complement & 10's complement.

For binary $\Rightarrow r = 2$
↳ 1's complement & 2's complement.

PAGE NO.:
DATE:

Find 9's complement of : 32457 ⇒

```
 99999
-32457
 67542
```

Ans: 67542

+1

Find 10's complement of : 32457

Ans: 67543

Find 1's complement of 10110110

```
 11111111
-10110110
 01001001   ← 1's complement.
    +1
 01001010   ← 2's complement
```

Ans.

Shortcut.

```
1 0 1 1 0 1 1 0
0 1 0 0 1 0 0 1   ← 1's complement.
```

Q. Perform this operation: 72532 - 13250 using 10's complement.

```
 99999
-13250
 86749
+72532
 159281
```

```
 99999
-13250
 86749
    +1
 86750
+72532
 159282
```

discard it.

Ans: 59282

A-B → Process.

i. Take compliment of B
ii. Add with A

Arithematic operation using 2's compliment.

1. Take 2's ~~complement~~ of ADD two numbers.
2. If any no. is negative, take 2's complement.
3. If ans is negative, ans will be in 2's complement.

Unsigned no. – any positive number including zero.
signed no.- negative number

Unsigned no. : +14 = 0 0 0 0 1 1 1 0
signed no. :- can be represented in;
a. sign magnitude rep.
b. 1's complement
c. 2's complement.

a. Sign-magnitude rep. → {0-positive, 1-negative} MSB.
~~+14~~ → 1000 1110 → -14

b. 1's complement → 1 1 1 1 0 0 0 1 → -14 → mostly used for logical manipulation, old computer.

c. 2's complement → 1 1 1 1 0 0 1 0 → -14 → widely used.

1. Convert the following hexadecimal no. to decimal and octal no.
a. OFFF b. 3FFF.

a. OFFF.

→ doln.

= ~~$0\times16^3+F\times16^2+F\times2^1+F\times02^0$~~ $0\times16^3+F\times16^2+F\times16^1+F\times16^0$

= ~~$0+16\times4+16\times2+16\times1$~~ $0+3840+240+15$.

= $(4095)_{10}$

8	4095	→	7
8	511	→	7
8	63	→	7
	7		

$(7777)_8$

Ans.

b. 3FFF

= $3\times16^3+F\times16^2+F\times16^1+F\times16^0$

= $12288+3840+240+15$

= $(16383)_{10}$

8	16383	→	7
8	2047	→	7
8	255	→	7
8	31	→	7
	3		

$(37777)_8$

2068

1. Convert following decimal no. to hexadecimal and octal number.

a. 504

⇒

16 | 504	→ 8	↑
16 | 31	→ 15	↑
1		

15 = F

∴ $(1F8)_{16}$

8 | 504	→ 0	
8 | 63	→ 7	↑
7		

∴ $(770)_8$

b. 250

⇒

16 | 250	→ 10	
16 | 15	→ 15	↑
0		

∴ $(FA)_{16}$

8 | 250	→ 2	
8 | 31	→ 7	↑
3		

∴ $(372)_8$

66

1. Convert following octal number to hexadecimal.

a. 1760.46

$\Rightarrow 1\times8^3 + 7\times8^2 + 6\times8^1 + 0\times8^0 \cdot 4\times8^{-1} + 6\times8^{-2}$

$= 512 + 448 + 48 + 0 \cdot 0.5 + 0.09375$

$= 1008 \cdot + 0.59375$

1008

→ Binary;

001 111 110 0000 ⇒ 1760

3 F 0

100 0 1 1000 ⇒ 46

9 8

⇒ 3F0.98

b. 6055.263

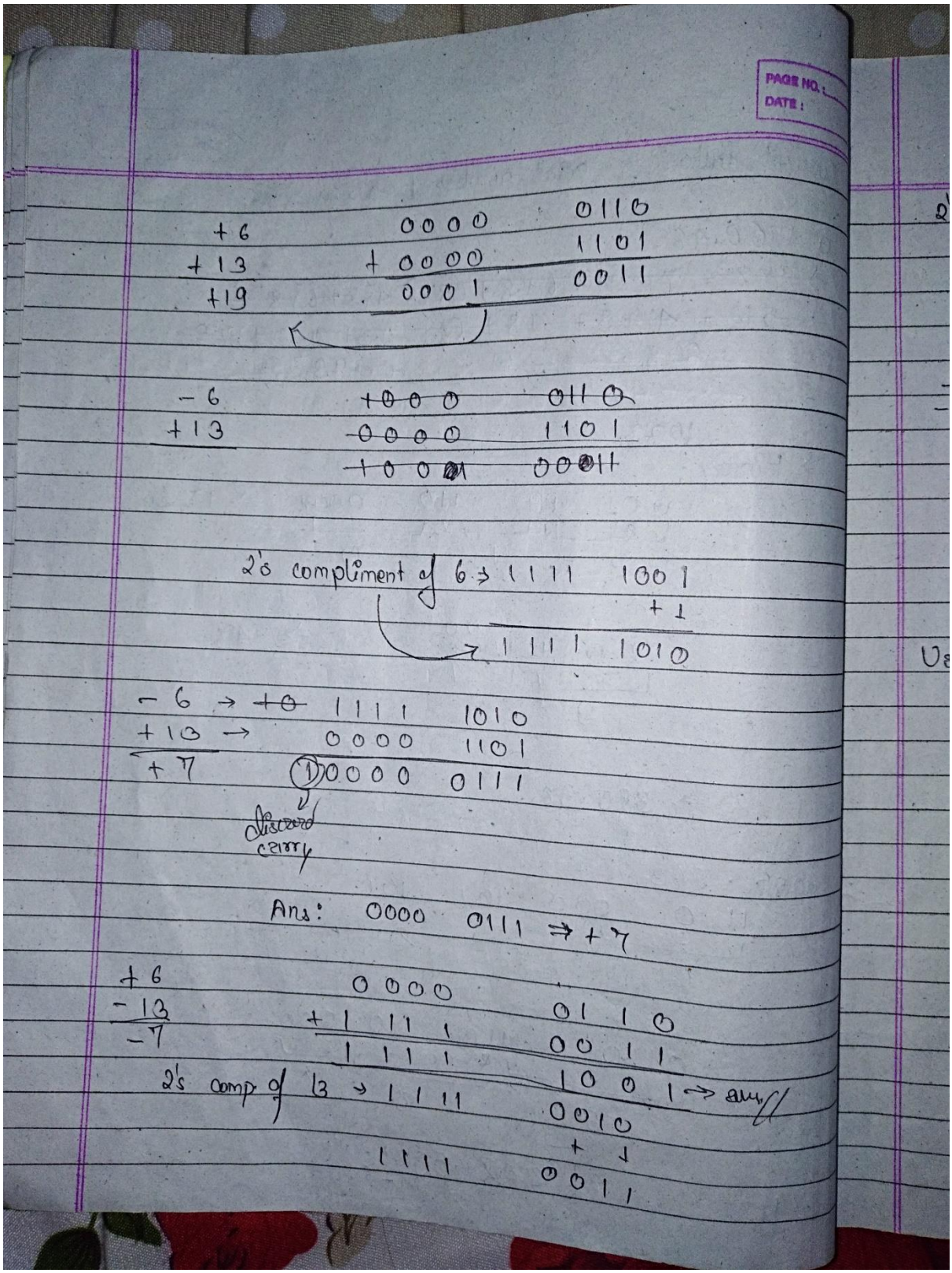
110 000 101 101

C 2 D

010 110 011000

5 9 8

$(C2D.598)_{16}$.



PAGE NO.:
DATE:

0110

2's comp. of ans; => 0000 ~~1001~~
+1
0000 ~~1010~~ Ans.
0111 check

−6	1111 1000
−13	+ 1111 0011
	~~11110 11001~~ (1)1110 1101

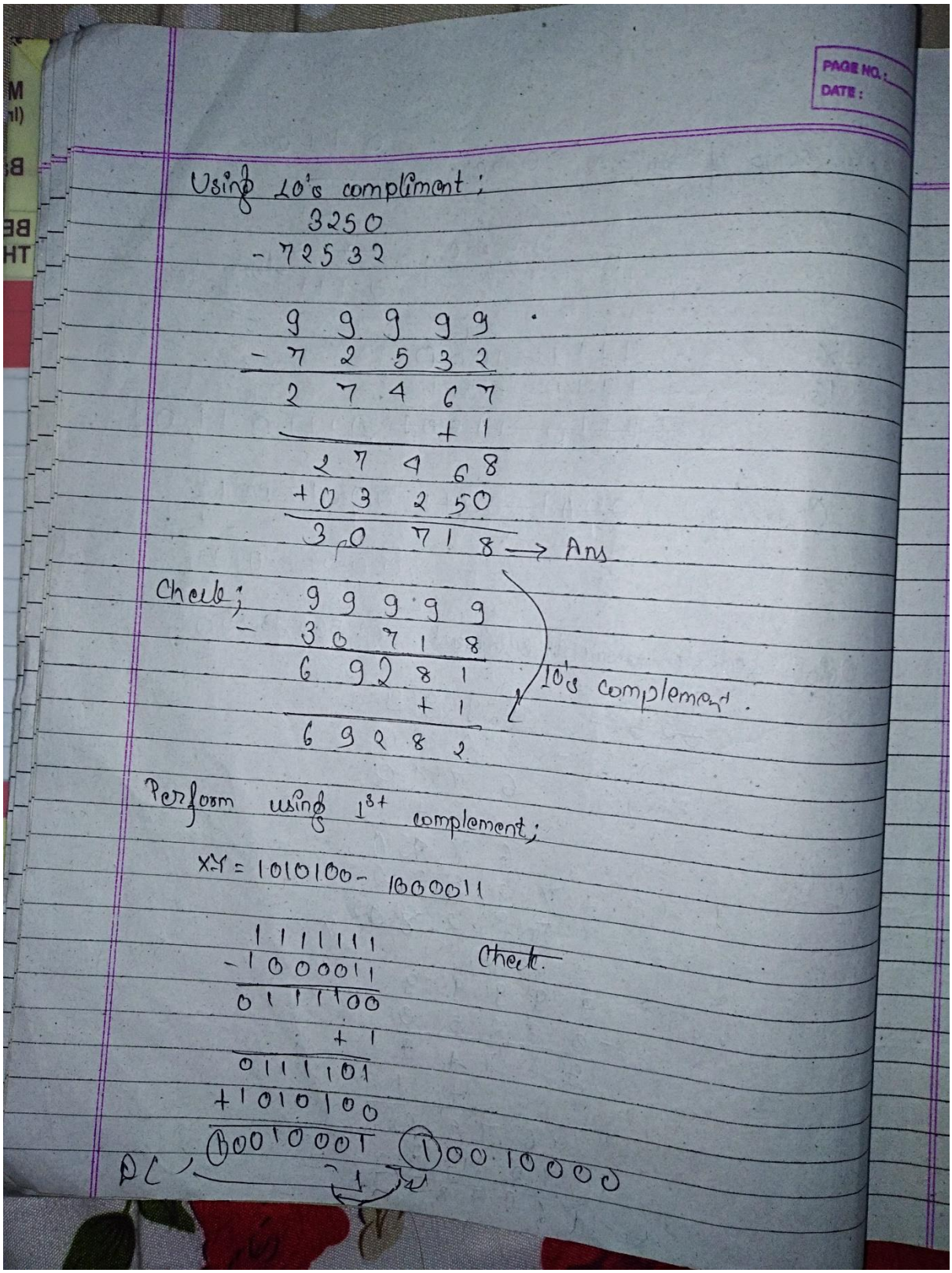
Check: ~~0001 0011~~ 0001 0010
+1
0001 0011

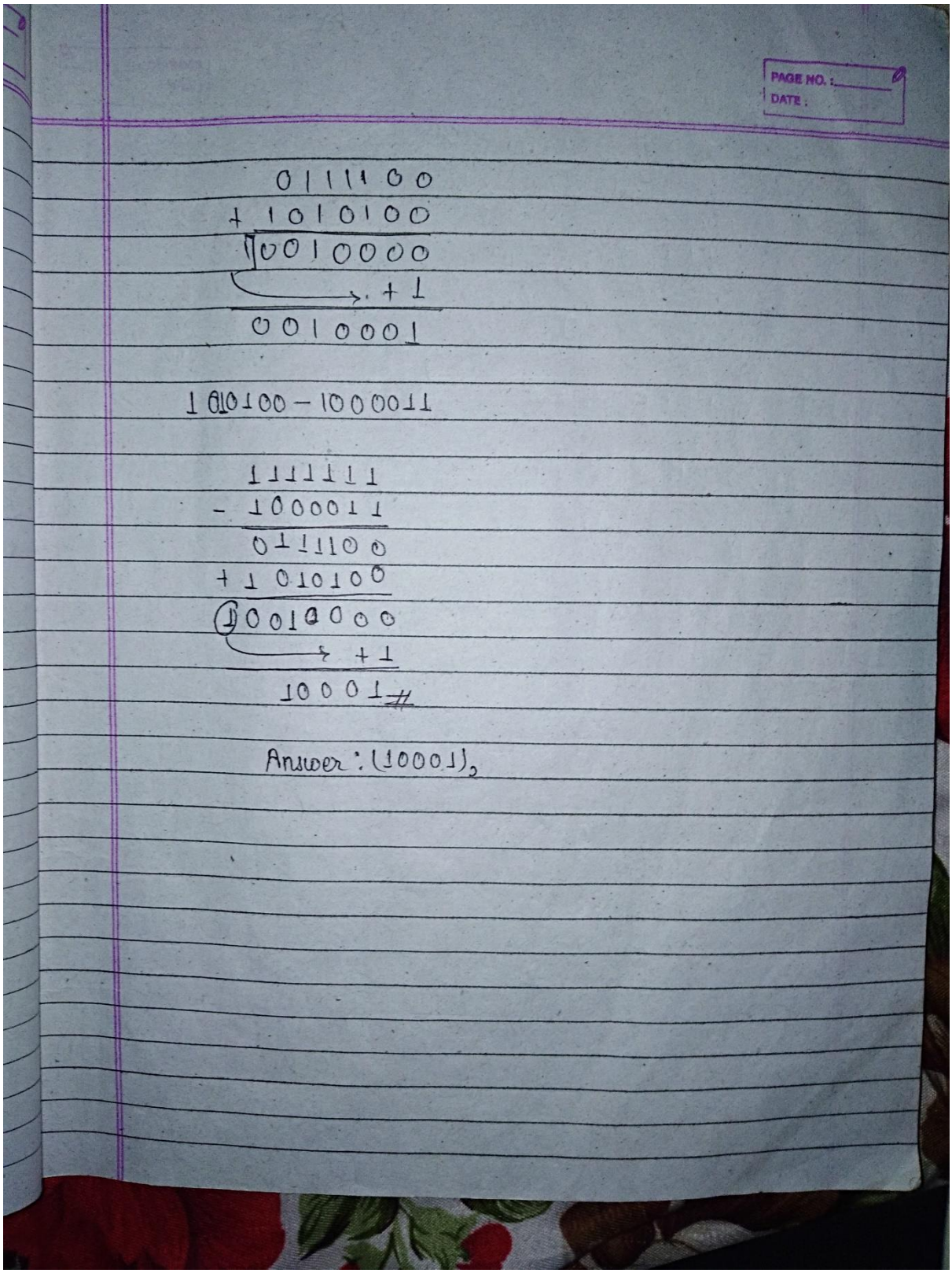
Using 10's complement, subtract 72532 − 3250

~~3250~~

9999
− 3250
6749
+1
6750
+ 72532
79282

99999
− 03250
96749
+1
96750
+ 72532
(1) ← (1) 69282 Ans.





PAGE NO. :
DATE :

$$\begin{array}{r} 0111100 \\ +\ 1010100 \\ \hline 10010000 \\ \hline 0010001 \end{array} \quad \begin{array}{l} \\ \\ \\ \rightarrow +1 \\ \\ \end{array}$$

$1010100 - 1000011$

$$\begin{array}{r} 1111111 \\ -\ 1000011 \\ \hline 0111100 \\ +\ 1010100 \\ \hline 10010000 \\ \hline 10001 \end{array} \quad \begin{array}{l} \\ \\ \\ \\ \rightarrow +1 \\ \\ \end{array}$$

Answer : $(10001)_2$

Binary Codes

1. BCD Codes
2. Gray Codes
3. Error Detection Codes
4. Alphanumeric Codes
5. Other codes (2421, Excess-3)

1. BCD codes

- Binary Coded Decimal
- Digital Calculator, Digital thermometer, Digital codes

Binary

w	x	y	z
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

BCD

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

↳ only upto 0-9 not from 10-15. (unused)

Eg: 22

Binary no. ⇒ 00010110

BCD ⇒ 0010 0010

1 2 4 8 16 32 64 128 256

PAGE NO. :
DATE :

(1420)$_{BCD}$ ⇒ 0001 0010 0010 0000

Q. Find binary and BCD representation of 99.

Binary : $(01100011)_2$

BCD : $(10011001)_{BCD}$

Q. Why BCD has 4-bits ?

BCD addition

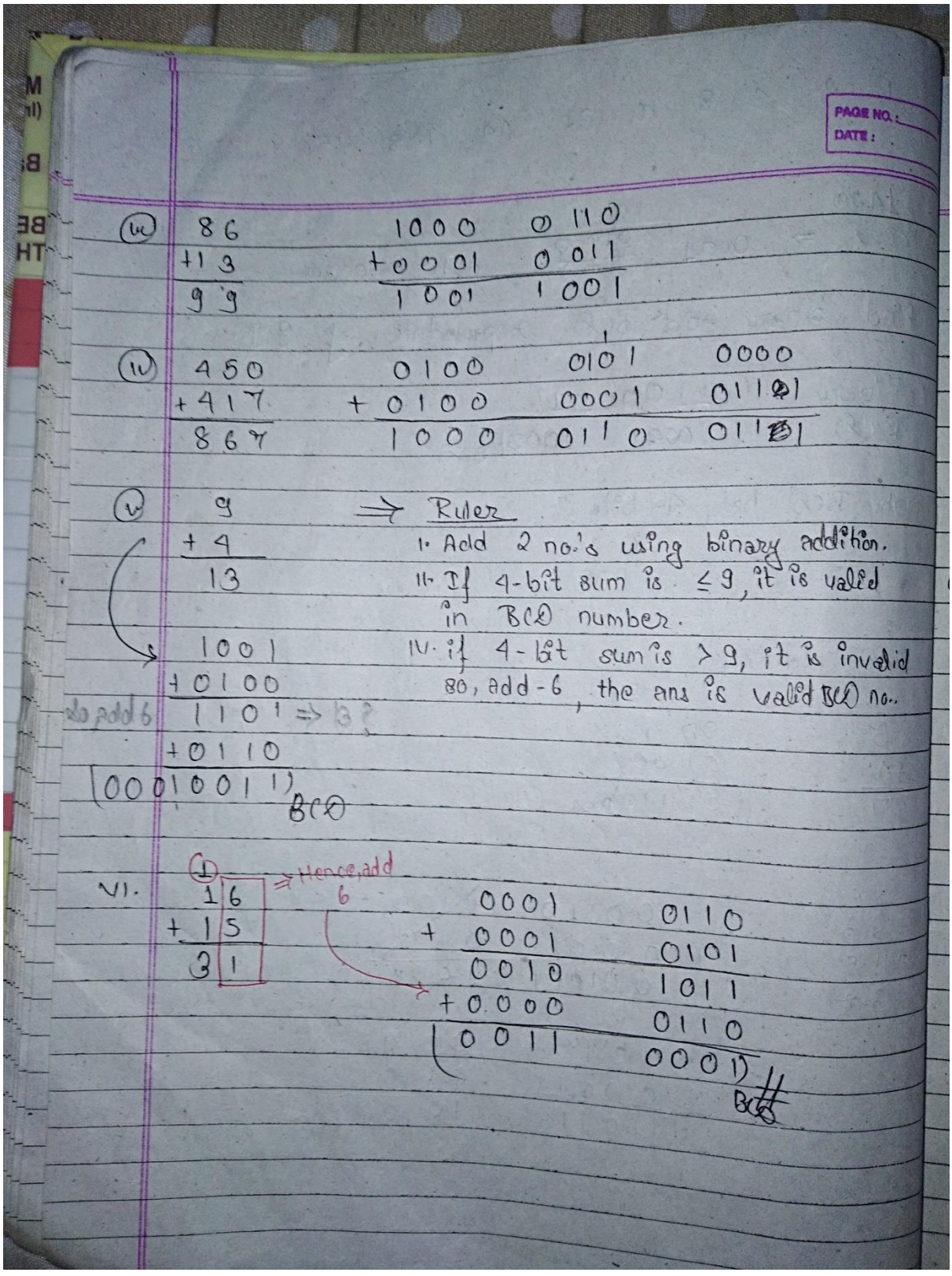
① 3 + 4 = 7

0011 + 0100 = $(0111)_{BCD}$

② 23 + 15 = 38

0010 0011 + 0001 0101 = 0011 1000

0010 0011 + 0001 0101 = $(0011\ 1000)_{BCD}$



11>9 → add 6

10>9 → add 6

7. 6 7
+5 3
1 2 0

0 1 1 0 + 0 1 0 1 = 1 0 1 1

1 0 1 1 + 0 0 0 0 = 1 0 1 1

1 1 0 0 + 0 1 1 0 = 0 0 1 0 0

0 1 1 1 + 0 0 1 1 = 1 0 1 0

1 0 1 0 + 0 1 1 0 = 0 0 0 0

0 0 0 0 + 0 0 0 0 = 0 0 0 0

(0 0 0 1 0 0 1 0) BCD

Disadvantages of BCD

↳ difficult in calculating 9's complement.

To overcome this problem → 2421 & Excess-3 codes is used.

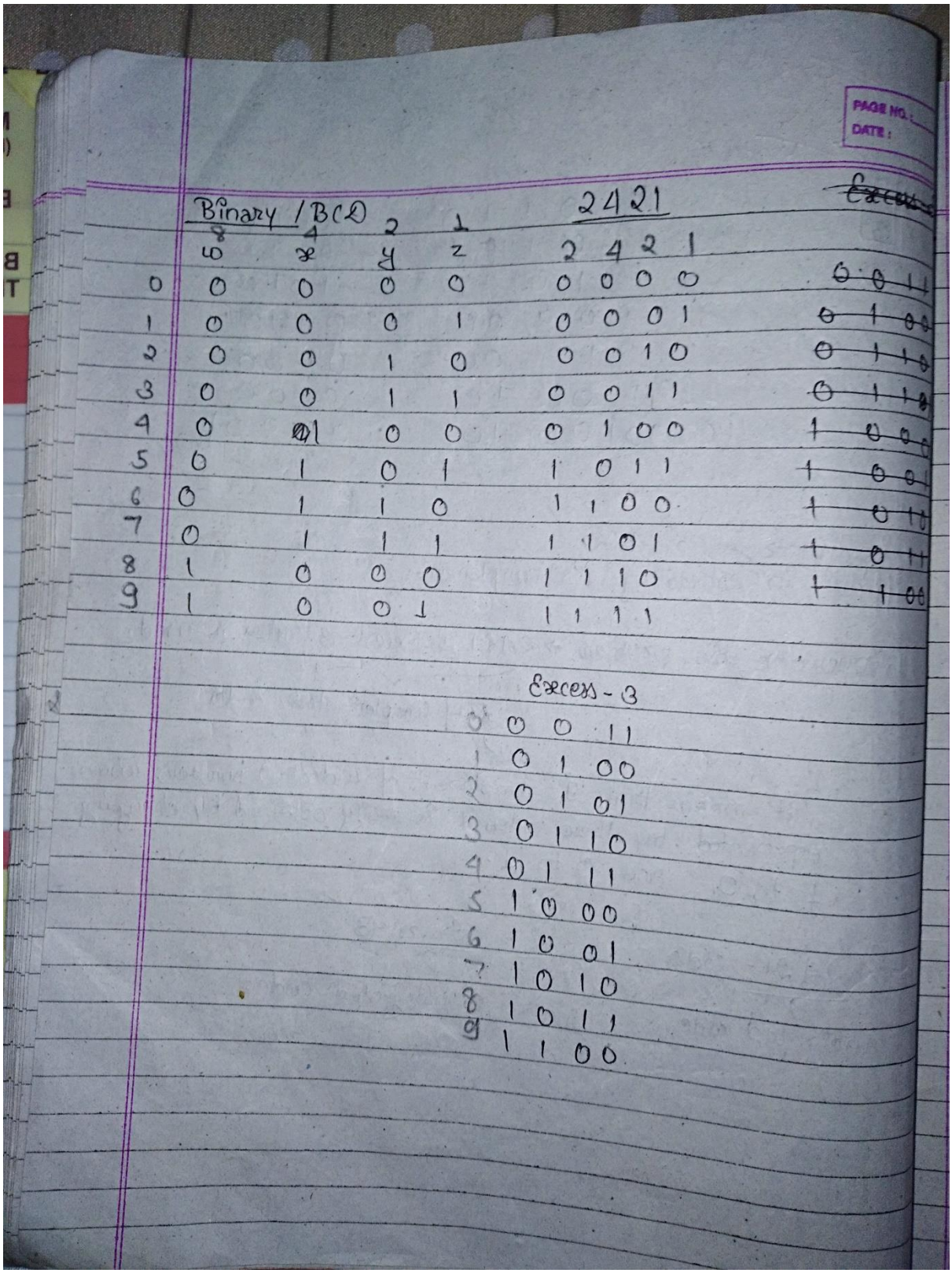
self complementary codes

↙

it means that, 9's complement of a decimal number, when represented by these codes, it is easily obtained by changing 1 to 0 and 0 to 1.

2421 - code
↑
weighted code

Excess-3
↑
Unweighted code.



Binary / BCD

	8 w	4 x	2 y	1 z	2421 2 4 2 1
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 0
3	0	0	1	1	0 0 1 1
4	0	0	0	0	0 1 0 0
5	0	1	0	1	1 0 1 1
6	0	1	1	0	1 1 0 0
7	0	1	1	1	1 1 0 1
8	1	0	0	0	1 1 1 0
9	1	0	0	1	1 1 1 1

Excess - 3

0 0 0 1 1
1 0 1 0 0
2 0 1 0 1
3 0 1 1 0
4 0 1 1 1
5 1 0 0 0
6 1 0 0 1
7 1 0 1 0
8 1 0 1 1
9 1 1 0 0

2. Gray code

- Unweighted code and not-arithematic code.
- Single bit changes from one code coord to next in sequence.
- Important in various applications such as shaft position Encoder, where error occurs on no. of bit changes.

Decimal	Binary	Gray code
0	0000	0000 - 0
1		0001 - 1
2		0011 - 2
3		0010 - 3
4		0110 - 4
5		0111 - 5
6		0101 - 6
7		0100 - 7
8		1100 - 8
9		1101 - 9
10		1111 - 10
11		1110 - 11
12		1010 - 12
13		1011 - 13
14		1001 - 14
15	1111	1000 - 15

single bit change

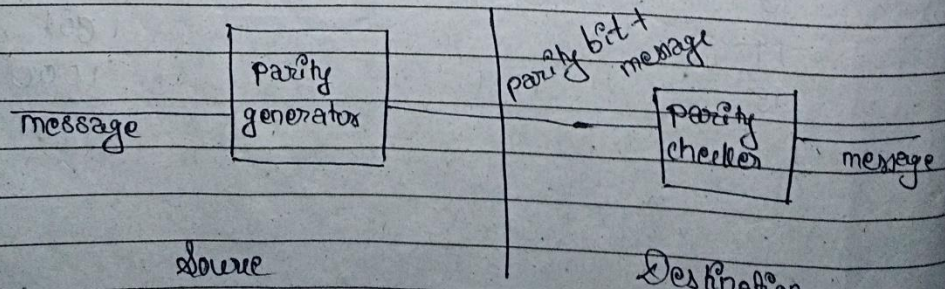
PAGE NO. :
DATE :

Errors Detection Code.

- Errors might occur during transmission of data.
- Error detection codes are binary codes.
- it can detect errors but cannot correct them.
- most common method for error detection is parity check.
- Parity bit is an extra bit added in the message.

Message			odd parity	Even Parity.
x	y	z		
0	0	0	1	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	0	1

Parity - Generator.



eg. 0101 → 10101 → eg. 10001

(5) odd parity. parity bit 10101

- Parity bit is chosen in such a way that no. of ~~0~~ 1's is even or odd.

3 | 4 | 5
odd par. 10011 | 01100 | 10101
even parity 00011 | 00100 | 00101

Disadvantages of Error-Detection Code
An even no. of errors are not detected.

Q. List 10 BCD Digits with Even parity in left most position. Repeat with odd parity.

BCD digits	Even parity	Odd parity
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1

Q. Find odd and even parity of the following :-

a.

	Odd parity	Even parity
a. 1010	11010	01010
b. 10111011	110111011	010111011
c. 101011110001	0101011110001	1101011110001

IV. Alphanumeric codes.

alphabet, numeric & others (keyboard characters)

a. ABCII. (American Standard Code Information Interchange
b. ~~BBC~~ EBCDIC (Extended BCD Interchange Codes.

ABCII ⇒ 128 characters. → used widely.
↳ 7 bit message + 1 bit of parity.
95 characters = Graphic symbols (a, A, ⊥)
23 " = format effectors (backspace, tab, etc)
10 " = used for direct data ~~ect~~ communication flow and report status.

A → 1000001
!
Z → 1011010
6 → 0110110
space → 0100000
* → 0101010

examples

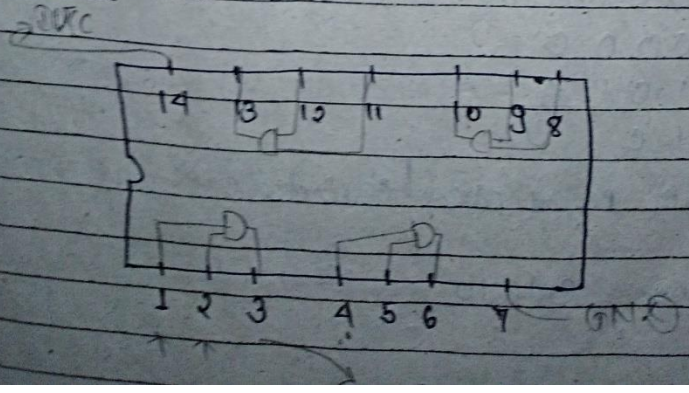
PAGE NO. :
DATE :

EBCDIC
- Use same symbols as ASCII.
- Used in IBM equipments computers
- 8 bit message + 1 bit for parity.

ICs (Integrated Circuits).
- chips / Micro chips.
a. Linear ICs / Analog - continuous in nature.
b. Digital ICs - discrete in nature.
→ logic gates.
→ amplifier

e.g. : (a) IC 7408 - AND Gate
(b) IC 7432 - OR Gate
(c) IC 7404 - NOT Gate
(d) IC 7400 - NAND Gate
(e) IC 7402 - NOR Gate
(f) IC 7480 - XOR Gate
(g) IC 4077 - XNOR Gate

IC 7408 :-



Revision Numericals

PAGE NO.:
DATE :

a. (+42) + (-13) → perform the following using 2's complement.

b. 11010 - 10000

c. 1010100 - 1010100

d. Find 2's complement of DEAF.

f. (-6) + (-13) - Using 2's complement.

g. $(AF17.C6)_{16} = (?)_{10}$

h. (-638) + (+785) → Using 10's complement.

i. $(111.111)_2 = (?)_{10} = (?)_8 = (?)_{16}$

j. $(F3A1)_{16} = (?)_8$

Solutions:

b. Here, 11010 - 10000
(a - b)

Taking 2's complement of 'b':

```
  10000
  01111
+     1
-------
  10000
```

Now, adding the ans to a;

```
  11010
+ 10000
-------
 (1)01010
```

Discard the carry over / encircled one.

∴ Answer = 01010

c. Soln,

Here, The given operation is;

$1010100 - 1010100$

Comparing the operation with a-b;

Then, $a = 1010100$ & $b = 1010100$.

Now,

Taking 2's complement of b;

$$\begin{array}{r} 1111111 \\ -\ 1010100 \\ \hline 0101011 \\ +\ 1 \\ \hline 0101100 \end{array}$$

Adding the ~~ans~~ output to a;

$$\begin{array}{r} 1011000 \\ +\ 1010100 \\ \hline (1)0000000 \end{array}$$

Discard the carry ~~over~~ over / the encircled ones;

$\therefore$ Answer = 0000000

d. Soln,

We have, to find 2's complement of $(\text{DEAF})_{16}$.

Now,

Converting $(\text{DEAF})_{16}$ into binary digits;

D E A F

$= D \times 16^3 + E \times 16^2 + A \times 16^1 + F \times 16^0$

$= 13 \times 16^3 + 14 \times 16^2 + 10 \times 16 + 15 = (57007)_{10}$

a

2	57007	→ 1
2	28503	→ 1
2	14251	→ 1
2	7125	→ 1
2	3562	→ 0
2	1781	→ 1
2	890	→ 0
2	445	→ 1
2	222	→ 0
2	111	→ 1
2	55	→ 1
2	27	→ 1
2	13	→ 1
2	6	→ 0
2	3	→ 1
	1	

Binary :- $(1101111010101111)_2$

1's complement :- 0010000101010000

2's complement :- $(1000010101010001)_2$.

Q. $(-6) + (-13)$

⇒ Soln,

The given operation is,

$(-6) + (-13)$

Taking 2's complement of 6;

PAGE NO. :
DATE :

6 :→ 0000 0110

1111 1001 :→ 1's complement.

+1

1111 1010 → 2's complement.

Taking 2's complement of 13;

13 :→ 0000 1101

1111 0010 → 1's complement

+1

1111 0011 → 2's complement

-6	1111	1010
-13	1111	0011
-19	(1) 1110	1101

Discard

Answer: 1110 ~~1011~~ 1101 1101

Check; Taking 2's compliment of answer

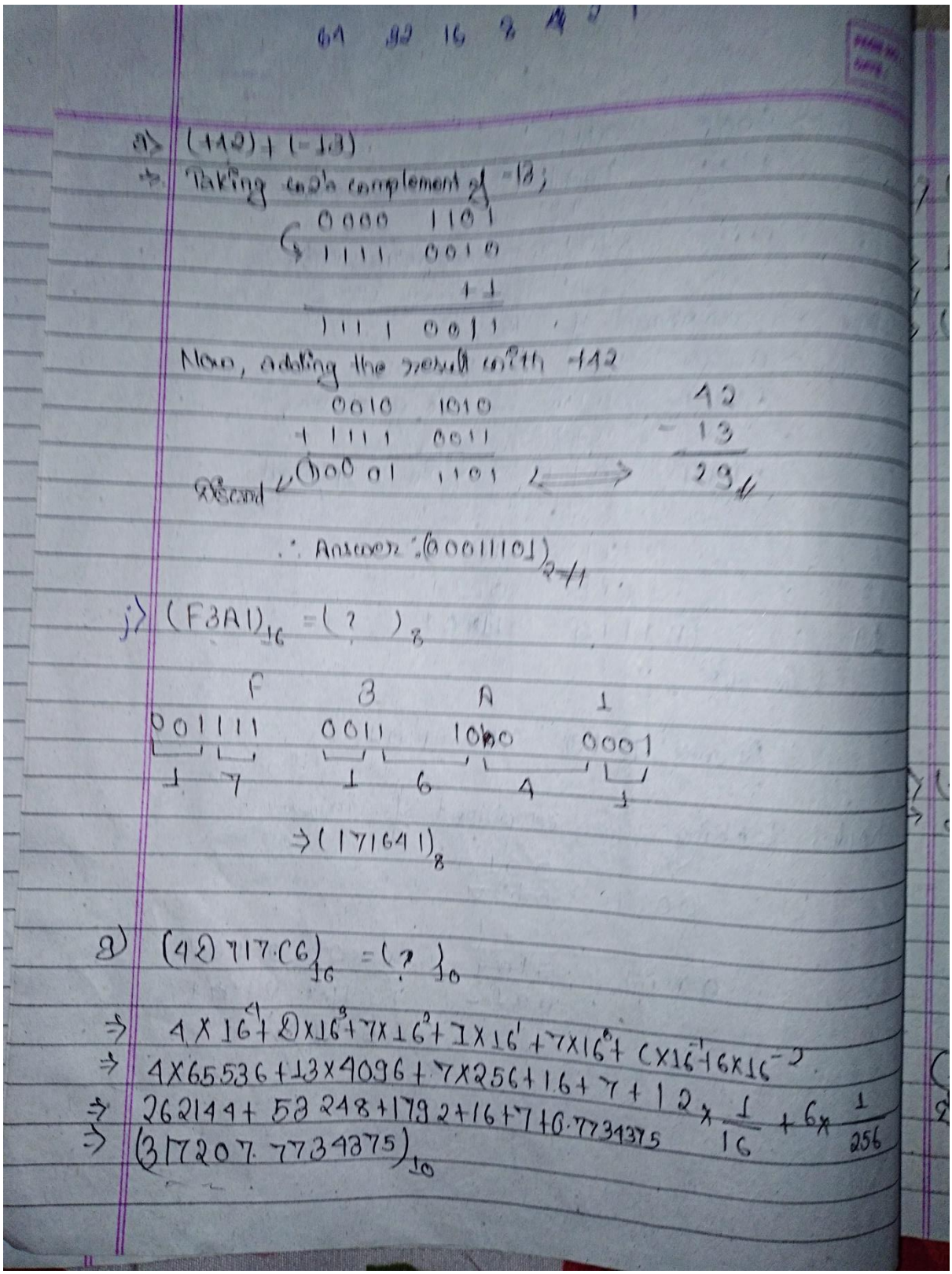
0001 ~~0100~~ 0010

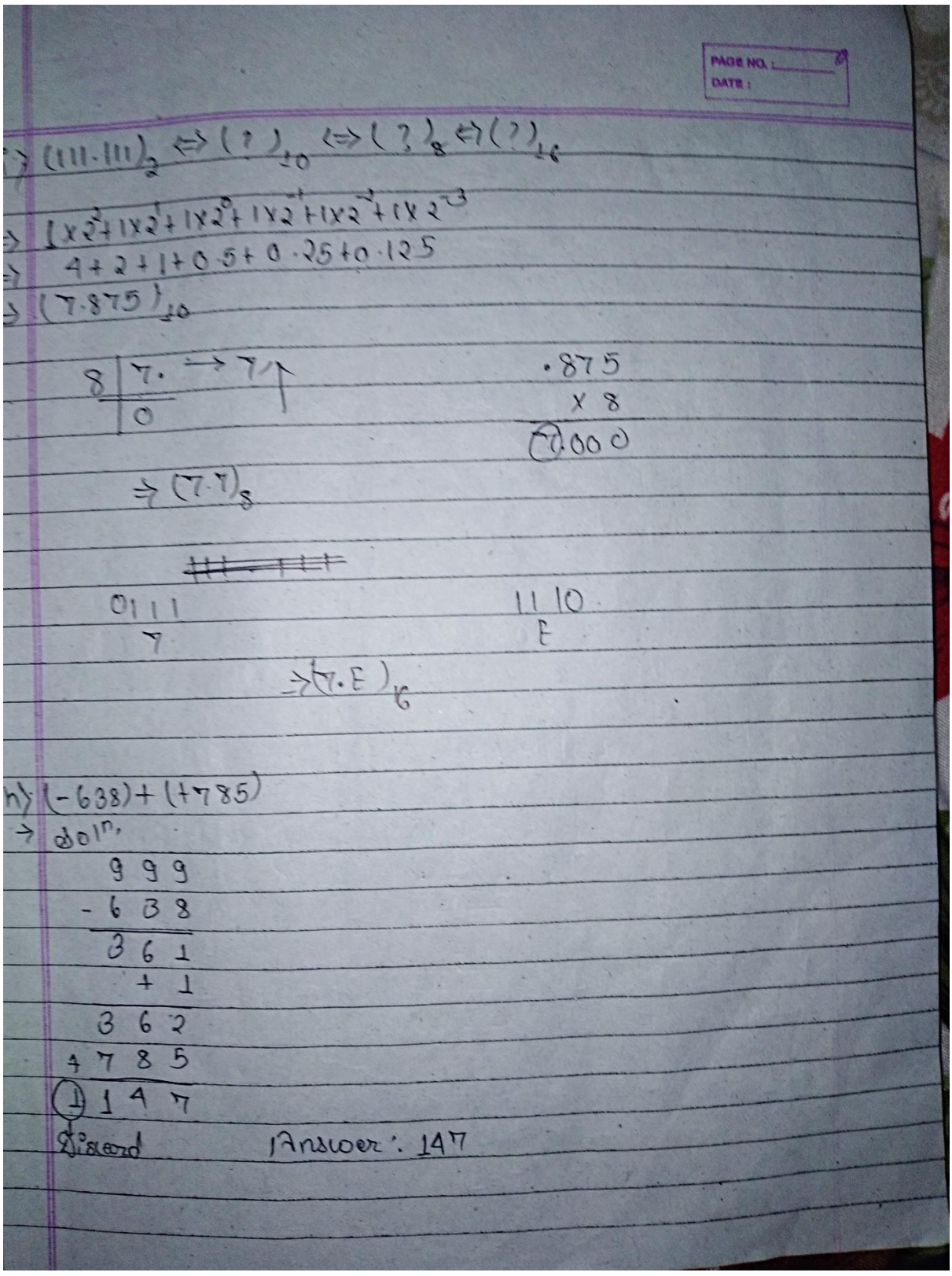
+1

0001 ~~0101~~ 0011 → $2^4 + 2^1 + 2^0$

$= 16 + 2 + 1$

$= 19$ //

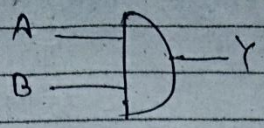




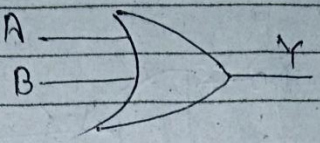
PAGE NO. :
DATE :

Basic Gates

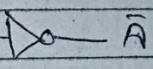
I. AND → $A \cdot B$ →



II. OR → $A+B$ →



III. NOT → $\bar{A}$ or A' →



Timing Signals

A = 1001
B = 0101

A	1	0	0	1
B	0	1	0	1
A·B	0	0	0	1
A+B	1	1	0	1
$\bar{A}$	0	1	1	0
$\bar{B}$	1	0	1	0

A B A·B A+B $\bar{A}$ $\bar{B}$

Q. $F = x + y'z$.
Draw Truth Table and Logic Diagram.

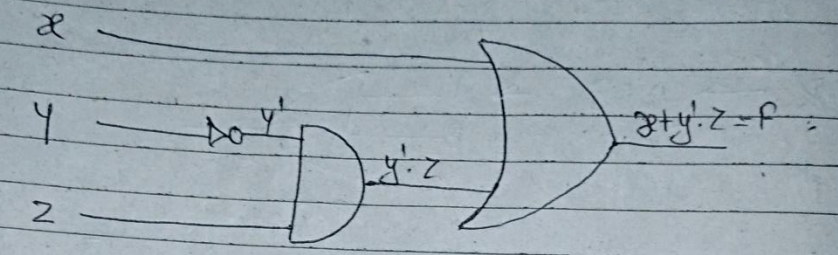


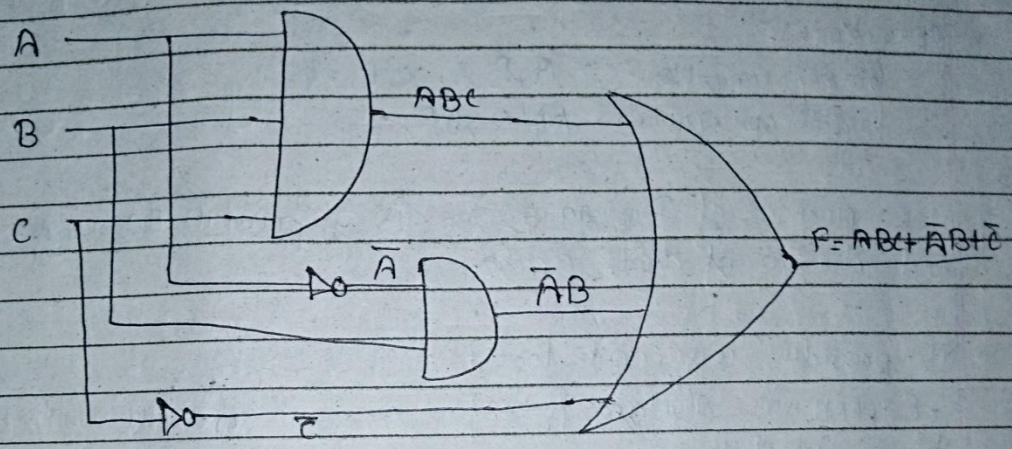
fig: Logic Diagram

Truth Table:

x	y	z	y'	y'.z	f=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

Next Method:
F=1 ; when $y'z = 1$ or $x = 1$
y' & z both $= 1$
$y = 0$, $z = 1$.

Q. $F = ABC + \bar{A}B + \bar{C}$; draw truth table & logic diagram.



Truth Table:

f = 1; when ABC = 1 or $\bar{A}B = 1$ or $\bar{C} = 1$

↳ A=1 & B=1 & C=1. ↳ $\bar{A}$ & B = 1 ↳ C=0

↳ A=0 & B=1

→

A	B	C	ABC	$\bar{A}$	$\bar{A}B$	$\bar{C}$	$ABC + \bar{A}B + \bar{C}$
0	0	0	0	1	0	1	1
0	0	1	0	1	0	0	0
0	1	0	0	1	1	1	1
0	1	1	0	1	1	0	1
1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	1	1
1	1	1	1	0	0	0	1

PAGE NO. :
DATE :

Boolean Algebra

→ is an algebra which deals with binary variables and logic operations.

Binary variables :→ A, B, C, x, y, etc.
logical operations → AND, OR, NOT.

→ The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits.

→ It provides convinent tool to:
- Express in algebric ~~from~~ form a truth table relationship both binary variables.
- Express in algebric form the input output relationship of logic diagrams.
- Find simpler circuit for the same function.

Postulates

P1: Closure: Boolean Algebra is closed under AND, OR, and NOT operations.

P2: Commutativity: The . and + operator are commutative.
i.e. $x.y = y.x$ and $x+y = y+x$ for all $x, y \in B$.

P3: Distribution: . and + are distributive w.r.t. one another
$x.(y+z) = x.y + x.z$
$x+(y.z) = (x+y).(x+z)$
for all $x, y, z \in B$.

P4 : Identity :

The identity element w.r.t. · is 1 and + is 0.

i.e.

$x + 0 = 0 + x = x$

$x \cdot 1 = 1 \cdot x = x$

P5 : Inverse

For every value of x, there exists a value x' such that $x \cdot x' = 0$ and $x + x' = 1$

P6 : There exists at least two elements, $x, y \in B$ such that $x \neq y$.

~~Principles~~ Principles of Boolean Algebra

Duality ~~Basic~~ Principles

'Every algebric expression deducible from postulates of Boolean algebra remains valid if the operators ~~are~~ and identity elements are interchanged'

i.e. We interchange ~~or~~ OR and AND operators and interchange 1 & 0.

Basic Theorems :

Theorem 1 : Idempotence :

(a) $x + x = x$ (b) $x \cdot x = x$

Theorem 2 : Existance

(a) $x + 1 = 1$ (b) $x \cdot 0 = 0$

1 variable theorem

Theorem 3 : Involution

$(x')' = x$

Theorem 4 : Association

(a) $x + (y+z) = (x+y)+z$ (b) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

2 | 3 variable theorem

Theorem 5 : De-Morgan

(a) $(x+y)' = x' \cdot y'$ (b) $(x \cdot y)' = x' + y'$

Theorem 6 : Absorption

(a) $x + xy = x$ (b) $x \cdot x + y = x$

1.a. $x + x = x$

Proof:

$x + x$

$= (x+x) \cdot 1$ // P_4 $(1 \cdot x = x \cdot 1 = x)$

$= (x+x) \cdot (x+x')$ // P_5 $(x + x' = 1,\ x \cdot x' = 0)$

$= x + xx'$ // P_3

$= x + 0$ // P_5

$= x$ // P_4

1. b. $x \cdot x = x$

Proof:

$x \cdot x$

$= x \cdot x + 0$ // P_4

$= (x \cdot x) + (x \cdot x')$ // P_5

$= x \cdot (x + x')$ // P_3

$= x \cdot 1$ // P_5

$= x$ // P_4 #

PAGE NO.:
DATE:

Proof De-Morgan's theorem with truth table.

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

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

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

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

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

$\therefore (x \cdot y)' = x' + y'$ #

PAGE NO.:
DATE:

Simplify the Boolean functions.

a) $x + x'y$

→ Soln

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

$= 1 \cdot (x + y)$

$= (x + y)$ #

b) $x(x' + y)$

$= x \cdot x' + xy$

$= 0 + xy$

$= xy$ #

c) $x'y'z + x'yz + xy'$

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

$= x'z \cdot 1 + xy'$

$= x'z + xy'$

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

$= 0 + xy'$

$= y'$ #

d) $xy + x'z + yz$

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

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

$= zy(x + x')$

$= zy \cdot$

$= zy$ #

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

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

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

$= xy + x'z$

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

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

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

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

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

$A' \quad A$

$A + A' = 1$

$A \cdot A' = 0$

$xy + (x'+y') = 1$

$(x+y) \cdot (x'y') = 0$

Simplify:

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

$= [A\bar{B}C + A\bar{B}\bar{B}D + \bar{A}\bar{B}]C$

$= [A\bar{B}C + D + \bar{A}\bar{B}]C$

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

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

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

$= \bar{B}C$ #

State and Proove De-Morgan's Theorem:

→ First Theorem:

The compliment of product of variables is equal to sum of compliments of variables

i.e. $\overline{XY} = \bar{X} + \bar{Y}$

Second Theorem:

The compliment of sum of variables is equal to product of compliment of variables.

i.e. $\overline{X+Y} = \bar{X}\bar{Y}$

Proof:

One way is: by truth table

other way is:

PAGE NO. :
DATE :

Simplify the expression using De-Morgan's;

a ⓐ $\overline{XYZ}$

$= \overline{XY} + \overline{Z}$

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

ⓑ $\overline{(X+Y+Z)}$

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

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

ⓒ $\overline{(A+B+C)D}$

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

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

$= [\overline{A} \cdot \overline{B} \cdot \overline{C}] + \overline{D}$

ⓓ $\overline{AB(+D)EF}$

$= \overline{ABC} \cdot \overline{D}\,\overline{EF}$

$= (\overline{AB+C}) \cdot (\overline{D}\,\overline{E} + \overline{F})$

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

Find the compliment of following Boolean expressions.

a. $F_1 = x'yz' + x'y'z$; $F_1' = ?$

b. $F_2 = x(y'z' + yz)$; $F_2' = ?$

Two ways of getting complement.

1. Apply the De-Morgans theorem.

or,

2. First find dual of expression & then complement each variable.

(a) $F_1' = \overline{(x'yz' + x'y'z)}$

$= \overline{(x'yz')} \cdot \overline{(x'y'z)}$

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

$F_1' = (x + y' + z) \cdot (x + y + z')$ //

(b) $F_2' = \overline{x(y'z' + yz)}$

$= \overline{x(y+z) + yz}$ $= \bar{x} + \overline{(y'z' + yz)}$

$= \bar{x} + \overline{(y'z')} \cdot \overline{(yz)}$

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

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

Next method for calculating compliment:

(a) $F_1 = x'yz' + x'y'z$

By dualing principle,

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

Then compliment each variable,

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

$= F_1''$ //

PAGE NO. :
DATE :

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

By duality principle,

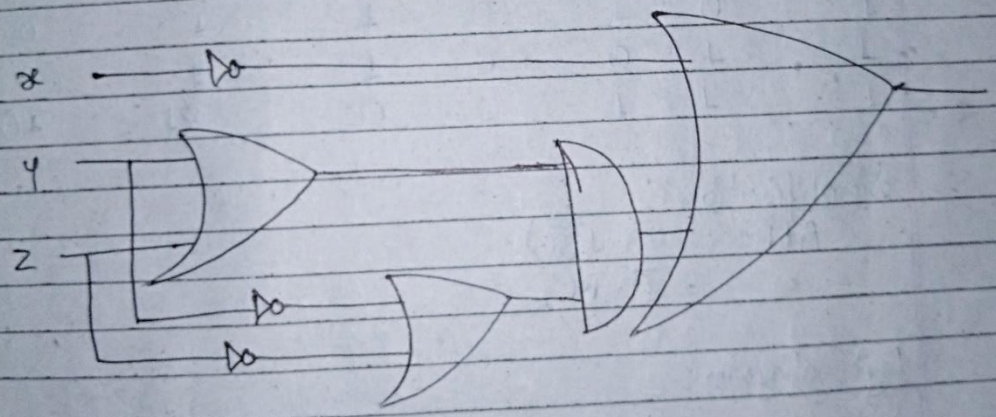
$= x + [(y' + z') \cdot (y + z)]$

Then compliment each variable,

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

$= F_2'$ #

Draw $[x' + (xy + z)(y' + z)]$



Simplify, find truth table and logic diagram.

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

→ Soln,

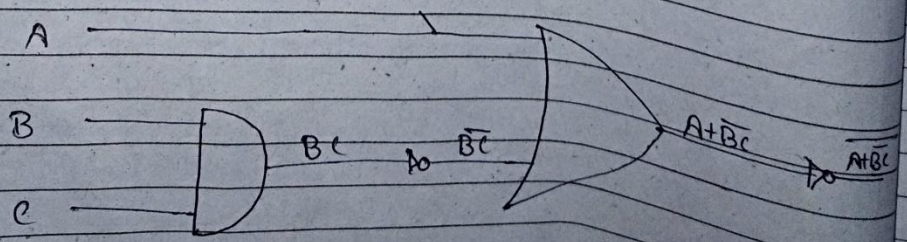
A	B	C	BC	$\bar{B}C$	$A + \bar{B}C$	$\overline{A+\bar{B}C}$
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	0	1	0

Simplification:

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

$= \bar{A} \cdot BC$ #

Logic diagram:



PAGE NO. :
DATE :

$F = \overline{(\bar{x}+z)(\bar{x}Y)}$

Simplification:

$F = \overline{(\bar{x}+z)(\bar{x}Y)}$

$= \overline{(\bar{x}+z)} + \overline{(\bar{x}Y)}$

$= (\bar{\bar{x}} \cdot \bar{z}) + (\bar{\bar{x}} + \bar{Y})$

$= (x\bar{z}) + (x + \bar{Y})$

$= x(\bar{z}+1) + \bar{Y}$

$= x + \bar{Y}$

Truth Table:

X	Y	Z	$x+\bar{Y}$	$(\bar{x}+z)$	$\bar{x}Y$	$\overline{(\bar{x}+z)(\bar{x}Y)}$
0	0	0	1	1	0	1
0	0	1	1	1	0	1
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	0	1.

Benefits of Boolean Algebra.
a) Expression - shorter
b) Simple Truth-Table
c) Simple Logic diagram for logic gate req'd.

Basic Gates - AND, OR, NOT — Derived gates

Universal gates — NAND, NOR

Exclusive — XOR, XNOR

NAND Gate.

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

Output is low, when all input is high.

NOR Gate

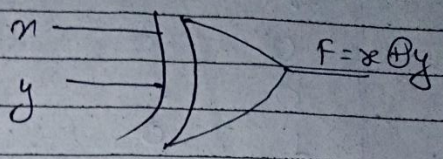
x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

Output is high, when all input is high.

PAGE NO. :
DATE :

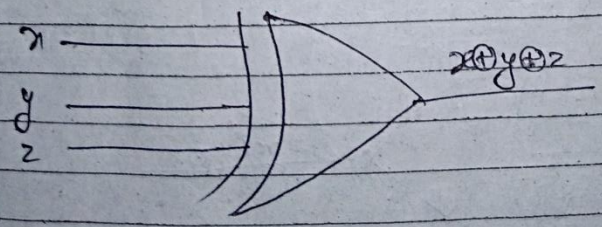
XOR Gate.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



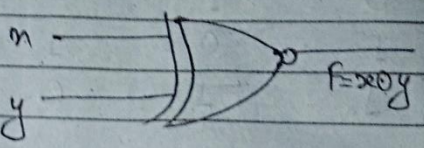
x	y	z	$x \oplus y \oplus z$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

even no. of 1's → 0
odd no. of 1's → 1



XNOR Gate

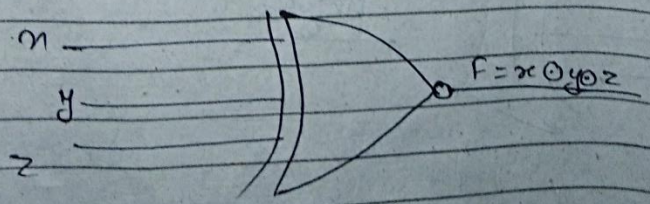
x	y	$x \odot y = \overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1



Three inputs;

x	y	z	$x \odot y \odot z$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

even no. of 1's = 1
odd no. of 1's = 0



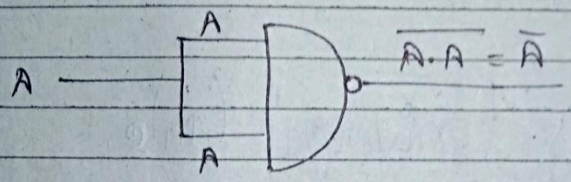
Universal Gates

→ other gates can be constructed from NAND/NOR gates.

NAND Gate As Universal Gate.

It can function as NOT gate, AND gate & OR gate.

NAND Gate as ~~Not~~ & NOT gate.

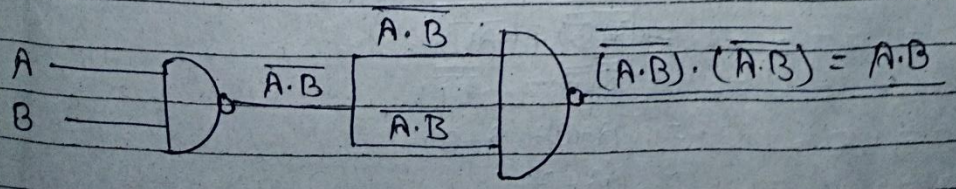


$\overline{A \cdot A} = \bar{A}$ $\{x \cdot x = x\}$

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

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

NAND Gate as AND Gate.

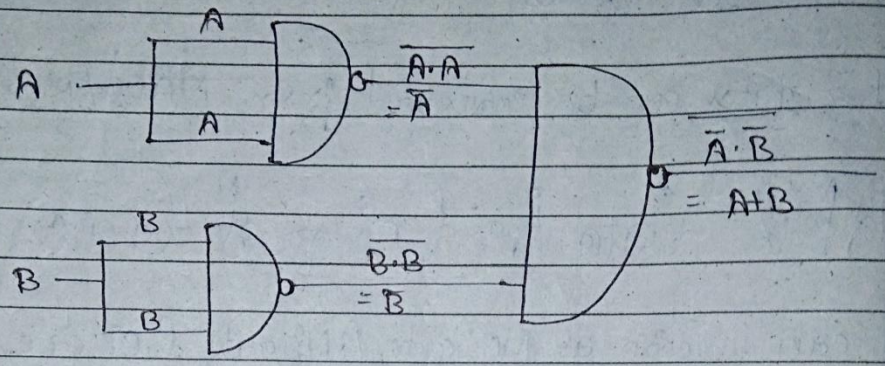


$\overline{(A \cdot B)} \cdot \overline{(A \cdot B)} = \overline{(A \cdot B)} + \overline{(A \cdot B)}$ (DeMorgan's)

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

$(x + x = x)$.

iii. NAND Gate as OR gate

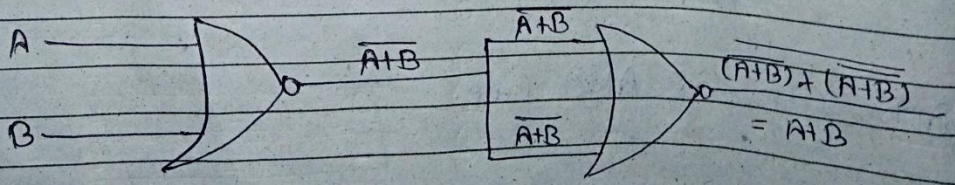


$\overline{A \cdot B} = \overline{A} + \overline{B}$

$= A + B$ #

NOR Gate as Universal Gate

1. NOR Gate as OR gate.



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

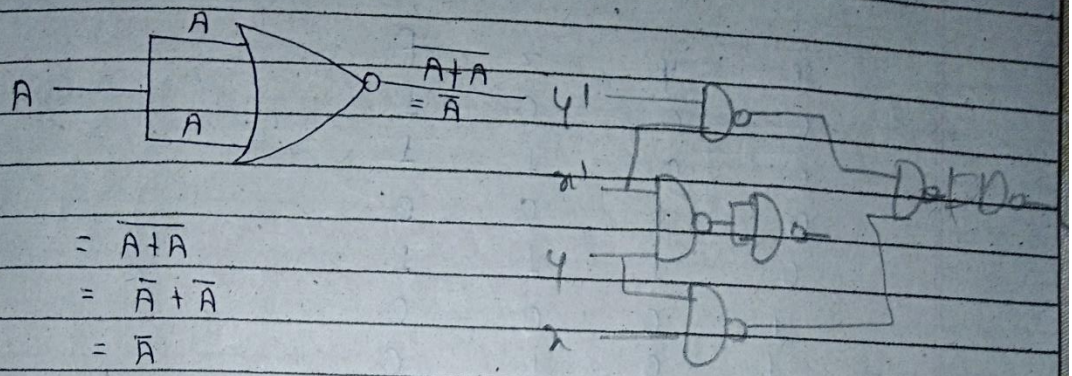
$= (\overline{\overline{A+B}}) + (\overline{\overline{A+B}})$

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

$= (A+B)$

PAGE NO. :
DATE :

11. NOR Gate as NOT Gate.

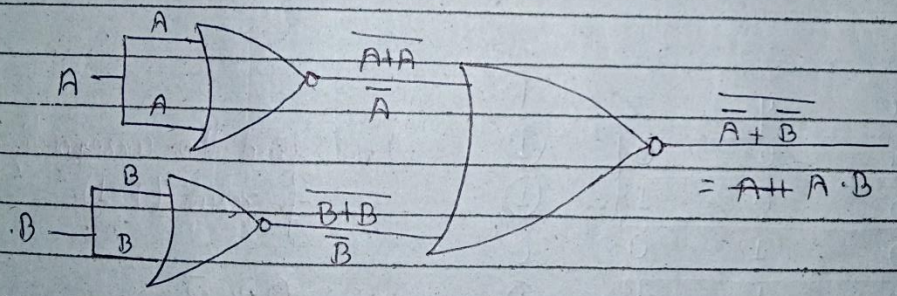


$= \overline{A+A}$

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

$= \bar{A}$

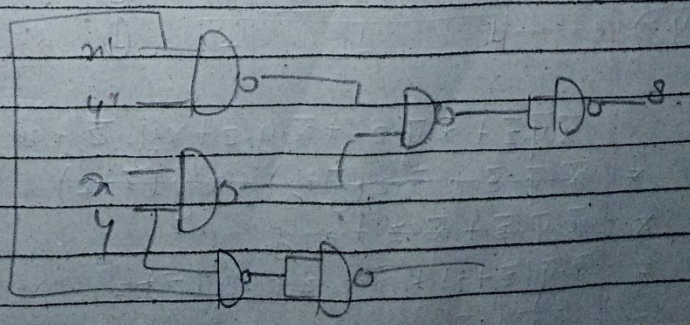
11. NOR Gate as AND Gate



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

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

$= A \cdot B$ #



Q. Find the boolean expression for the truth given truth table.

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

→ Soln,

	x	y	z	F	
$\bar{x}\cdot\bar{y}\cdot\bar{z}$	0	0	0	1	Step 1: Find the corresponding elements of result (F=1).
$\bar{x}\cdot\bar{y}\cdot z$	0	0	1	1	
	0	1	0	0	
$\bar{x}\cdot y\cdot z$	0	1	1	1	Step 2: 0 → $\bar{x}$, 1 → x
	1	0	0	0	
	1	0	1	0	
$x\cdot y\cdot\bar{z}$	1	1	0	1	Step 3: Add the resulting products
$x\cdot y\cdot z$	1	1	1	1	Step 4: Solve the (Step) expression.

$\bar{x}\cdot\bar{y}\cdot\bar{z} + \bar{x}\cdot\bar{y}\cdot z + \bar{x}\cdot y\cdot z + x\cdot y\cdot\bar{z} + x\cdot y\cdot z$

$= \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,z(\bar{y}+y) + x\cdot y(\bar{z}+z)$

$= \bar{x}\cdot\bar{y}\,\bar{z} + \bar{x}\,z + x\cdot y$

$= \bar{x}\cdot\bar{y}(\bar{z}+z) + \bar{x}\cdot y\cdot z + xy(\bar{z}+z)$

$= \bar{x}\,\bar{y} + \bar{x}\cdot y\cdot z + xy$

PAGE NO. :
DATE :

Characteristics of Digital Logic Families

1. Fan-in → ~~Fan~~ Input [Normally input is restricted not more than 4/5].
2. Propagation delay
3. Fan-out → Output
4. Power-dissipation
5. Noise-Margin → AC Noise (caused by switching signals)
   DC Noise (caused by voltage level of signal)

Digital Logic I.C. Family

DL (Diode Logic) RTL (Resistor-Transistor Logic)

a. DTL (Diode to Transistor Logic) } Bipolar
b. TTL (Transistor to Transistor Logic) }
c. ECL (Emitter Coupled Logic) }
d. MOS (Metal Oxide Semiconductor) } Unipolar
   PMOS
   NMOS
e. CMOS (Complementary Metal Oxide Semiconductor) → PMOS + NMOS
f. IIL (Integrated Injection Logic) (I²L)
g. GaAs & SiGe — Germanium
   Gallium Arsenide — Silicon (CCD → Charge coupled logic)

CMOS is mostly used today
1. high packaging ~~desting~~ density.
2. simple processing technique.
3. during fabrication
4. less power consumption.

* ECL has highest speed, so used in super computers.

Expression In Boolean Algebra

~~It is of too~~ Two ways :-

(a) Canonical Forms -
1. Minterm (product of all terms)
11. Maxterm (sum of all terms)

(b) Standard Forms -
1. Sum of Products (SoP)
11. Product of sums (PoS)

x	y	z	Minterm ($x=1; \bar{x}=0$)	Maxterm ($x=0; \bar{x}=1$)
0	0	0	$\bar{x}\cdot\bar{y}\cdot\bar{z}$ (m_0)	$x+y+z$ (M_0)
0	0	1	$\bar{x}\cdot\bar{y}\cdot z$ (m_1)	$x+y+\bar{z}$ (M_1)
0	1	0	$\bar{x}\cdot y\cdot\bar{z}$ (m_2)	$x+\bar{y}+z$ (M_2)
0	1	1	$\bar{x}\cdot y\cdot z$ (m_3)	$x+\bar{y}+\bar{z}$ (M_3)
1	0	0	$x\cdot\bar{y}\cdot\bar{z}$ (m_4)	$\bar{x}+y+z$ (M_4)
1	0	1	$x\cdot\bar{y}\cdot z$ (m_5)	$\bar{x}+y+\bar{z}$ (M_5)
1	1	0	$x\cdot y\cdot\bar{z}$ (m_6)	$\bar{x}+\bar{y}+z$ (M_6)
1	1	1	$x\cdot y\cdot z$ (m_7)	$\bar{x}+\bar{y}+\bar{z}$ (M_7)

$\# m_0 = \overline{M_0}$ $\quad$ $\# M_0 = \overline{m_0}$

$\overline{x\cdot y\cdot z} = \bar{x}+\bar{y}+\bar{z}$
$= x+y+z$

Q. $F = A + \overline{BC}$. Express it in terms of minterm.

→ Soln,

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

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

$= AB + A\overline{B}$

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

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

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

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

$F = ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\overline{C} + A\overline{B}C + \overline{A}\overline{B}C$

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

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

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

Maxterm = ?

Maxterm; $F(A,B,C) = \Pi(0, 2, 3)$

→ If '+' in Question, first calculate minterm.

→ If '.' in Question, first calculate maxterm.

Q. $F = \overline{\overline{A+B} + C}$. Express in terms of minterm & maxterm.

→ Soln,

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

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

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

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

~~F(A,B,C)~~ $F = A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$

$= m_6 + m_5 + m_2$

minterm ; $F(A,B,C) = \Sigma(2,5,6)$

maxterm ; $F(A,B,C) = \pi(0,1,3,4,7)$

$F = A'B + AB'C + ABC'$

→ SOP ⇒ but not min term (absense of all terms)

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

→ POS ⇒ but not maxterm (absence of all terms)

K-Map
- SOP → minterm
- POS → maxterm.

Simplify:

a. $F = a'bc + ab'c + abc' + abc$

a \ bc	00	01	11	10
0	000	001	(011)	010
1	100	(101)	(111)	(110)

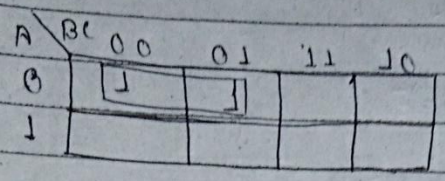
a \ bc	00	01	11	10
0			1	
(1)	1	1	1	1

→ Pairing in ..... 16, 8, 4, 2.
→ More preference to large no.
→ Naming the common ones.

ac, ab

Ans: = ac + bc + ab

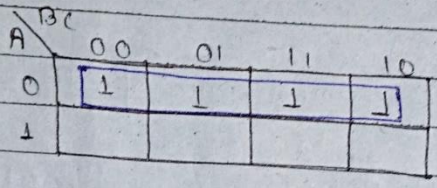
b) $F = A'B'C' + A'B'C$



Common elements are;
$A = 0 = A'$
$B = 0 = B'$

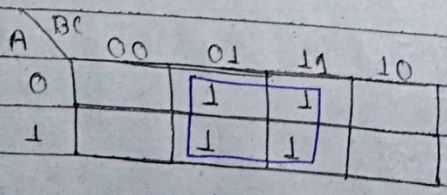
Ans: ~~a'b'~~ $A'B'$

c) $F = A'B'C' + A'B'C + A'BC + A'BC'$



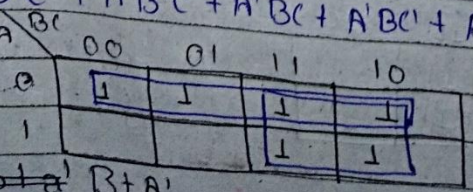
Ans: A'

d) $F = A'B'C + A'BC + AB'C + ABC$



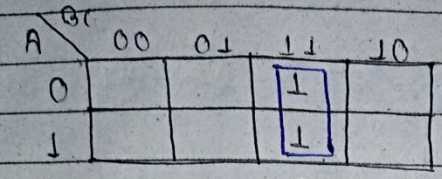
Ans: C

e) $F = A'B'C' + A'B'C + A'BC + A'BC' + ABC + ABC'$



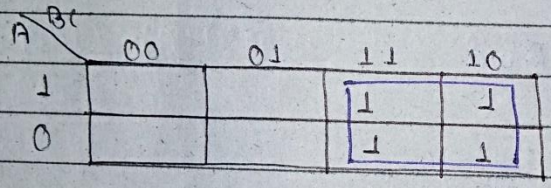
Ans: ~~b+a'~~ $B + A'$

f. $F = A'BC + ABC$



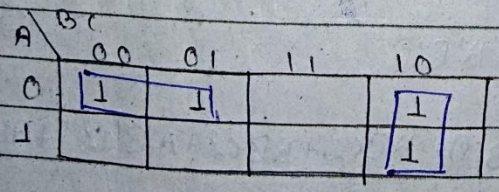
ans: BC

g. $F = A'BC + A'BC' + ABC + ABC'$



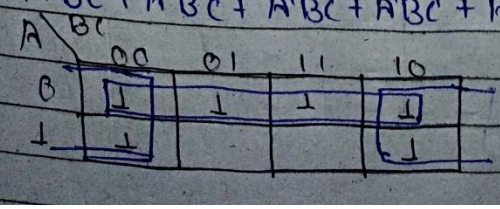
Ans: B

h. $F = A'B'C' + A'B'C + A'BC' + ABC'$



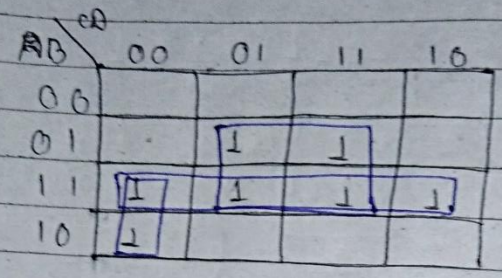
Ans : $A'B' + BC'$

i. $F = A'B'C' + A'B'C + A'BC + A'BC' + AB'C' + ABC'$



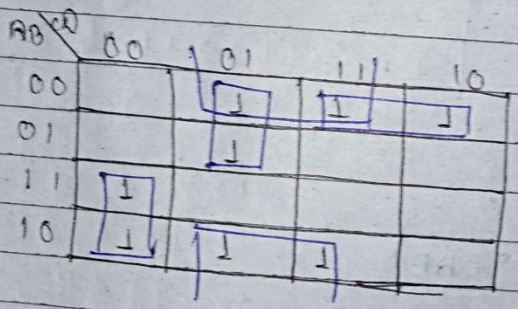
Ans: $C' + A'$

j. $F = AB + A'BC'D + A'BCD + AB'C'D'$



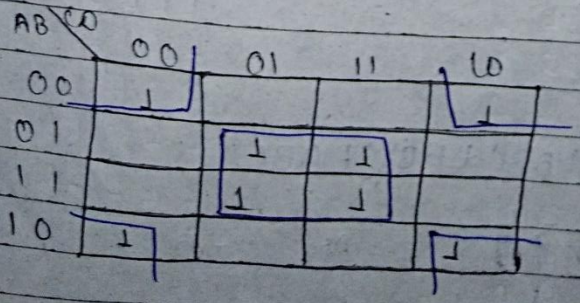
Ans : $BD + AC'D' + AB$

k. $F = AC'D' + A'B'C' + A'C'D + AB'D$



Ans : $B'D + AC'D' + A'C'D + A'B'C$

l. $A'B'C'D' + AB'C'D' + A'BC'D + A'BCD + ABC'D + ABCD + A'B'CD' + AB'CD'$



Ans: $BD + B'D'$

PAGE NO. :
DATE :

Assignment-1

1. Convert the following decimal numbers into hexadecimal and octal number.

a. 526.

16 | 526	526	→ 14 = E
16 | 32	32	→ 0
2		

$(20E)_{16}$

8 | 526	526	→ 6
8 | 65	65	→ 1
8 | 8	8	→ 0
1		

$(1016)_8$

24-01-2019

b. 681

Conversion of decimal to binary first,

1 0 1 0 1 0 1 0 0 1

Dividing in group of 4 bit	Dividing in group of 3 bit
0 0 1 0 1 0 1 0 1 0 0 1	0 0 1 0 1 0 1 0 0 1
2 A 9	1 2 5 1
$\therefore (2A9)_{16}$	$\therefore (1251)_8$

C. 412

Conversion into binary digits,

1 1 0 0 1 1 1 0 0

Dividing in groups of 4 ~~else~~ bits,

0 0 0 1 1 0 0 1 1 1 0 0

1 9 C

$\therefore (19C)_{16}$

Dividing in group of 3 bits,

1 1 0 0 1 1 1 0 0

6 3 4

$\therefore (634)_8$

2. Perform the following subtraction of binary numbers using i) 2's complement & ii) 1's complement

a) 1101 - 1011

i)

1 1 1 1
- 1 0 1 1
0 1 0 0
+1 1 0 1 1
1 1 1 1 1

ii)

1 1 1 1
- 1 0 1 1
0 1 0 0
+ 1
1 0 1
+1 1 0 1 1
(1) 0 0 0 0 0

Discard.

b. 10011 - 11001.

1 1 0 0 1

0 0 1 1 0

+ 1 0 0 1 1

1 1 0 0 1 #

Using 1's complement.

0 0 1 1 0

0 0 1 1 0

+ 1

0 0 1 1 1

+ 1 0 0 1 1

1 1 0 1 0

Using 2's complement.

c. 100 - 10101.

Using 1's complement,

0 1 0 1 0

+ 1 1 0 0

1 1 1 0 #

Using 2's complement,

0 1 0 1 0

+ 1

1 0 1 1

+ 1 1 0 0

1 1 1 1

3. Convert the decimal 220.5 to base 3, base 9, & base 16.

→ Conversion of decimal to base 3.

3	220	→ 1
3	73	→ 1
3	24	→ 0
3	8	→ 2
	2	

$(22021)_3$

Also,

.5

× 3

(1).5

× 3

(1).5

× 3

(1).5

$(.111)_3$

$\Rightarrow (22011.111)_3$

Conversion of decimal to base 4.

4	220	→ 0
4	55	→ 3
4	13	→ 1
	3	

$(3130)_4$

$\cdot 5$
$\times 4$
$2 \cdot 0$

$(0 \cdot 2)_4$

$(3130 \cdot 2)_4$

Conversion to base 5

5	220	→ 0
5	44	→ 4
5	8	→ 3
	1	

$(1340)_5$

$\cdot 5$
$\times 5$
$2 \cdot 5$
$\times 5$
$2 \cdot 5$

$(.22)_5$

$(1340.22)_5$

4. What is digital system? List out important feature of digital system.

→ A digital system is a system that stores data, process, and communicate information in a ~~de~~ discrete way (digital form). Digital systems process digital signals which can take only a limited number of values (discrete steps), usually just two values are used: the positive supply voltage (+Vs) and zero volts (0V).

Digital systems contains devices such as logic gates, flip-flop, shift registers & counters.

The important feature of digital system are as follows;

1. It has made possible many scientific, industrial and commercial advantages that would have been unattainable otherwise.
2. It's less expensive, more reliable and easy to manipulate.
3. It is flexible and compatible.
4. Information storage can be easier in digital computer systems than is analog one. New features can often be added to a digital system more easily too.

5 Differentiate analog system and digital system.

Analog system	Digital system
1. A combination of devices designed to manipulate physical quantities that are represented in analogue form.	1. A combination of devices designed to manipulate physical quantities that are represented in digital form.
2. Analog signal is continuous signal which represents physical measurement	2. Digital signals are discrete time signals generated by digital modulation.
3. Denoted by sine waves	3. Denoted by square waves
4. Uses continuous range of values to represent information.	4. Uses discrete or discontinuous values to represent information.

6. Write short notes:

a. BCD

→ BCD (Binary Coded Decimal)

The binary number system is the most natural system for a computer, but people are accustomed to the decimal system. So, to resolve this difference, computer uses decimals in coded form which the hardcoare understands. A binary code that distinguishes among 10 elements of decimal digits must contain at least four bits. Numerous different binary bits can be obtained by arranging four bit into 10 distinct combinations. The code most ~~continuously~~ commonly used for ~~binary to~~ decimal digits is the straightforward binary assignment listed in the table below. This is called binary-coded decimal and is commonly referred to as BCD.

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

A number with n decimal digits will require 4n bits in BCD. Eg decimal 396 is represented in BCD with 12 bits as 0011 1001 0110.

b.

PAGE NO. :
DATE :

b. Error detection code.

Binary information can be transmitted from one location to another by electric wires and other communication medium. Any external noise introduced into the physical communication medium may change some of the bits from 0 to 1 or vice versa.

The purpose of an error detection code is to detect such bit-reversal errors. One of the most common ways to achieve error detection is by means of a parity bit. A parity bit is the extra bit included to make the total number of 1's in the resulting code word either even or odd. A message of 4-bits and a parity bit P is shown in the table below:

Odd parity		Even parity	
Message	P	Message	P
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

PAGE NO. :
DATE :

c. Reflected code.

Reflected code (Gray code) is a binary coding scheme used to represent digits generated from a mechanical sensor that may be prone to error. Used in telegraphy in the late 1800s, and also known as "reflected binary code". Gray code was patented by Bell Labs researcher Frank Gray in 1947. In Gray code, there is only one bit location different between two successive values, which makes mechanical transitions from one digit to next less error prone. The following chart shows normal binary representations from 0 to 15 and the corresponding Gray code.

Decimal digit	Binary code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

PAGE NO.:
DATE:

The Gray code is used in applications where the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next. If binary numbers are used, a change from 0111 to 1000 may produce an intermediate erroneous number 1001 if the rightmost bit takes more time to change than the three other bits. The Gray code eliminates this problem since only one bit changes in value during any transition between two numbers.

d ASCII

The standard binary code for the alphanumeric characters is called ASCII (American Standard Code for Information Interchange). It uses seven bits to code 128 characters as shown in the table below. The seven bits of the code are designated by B_1 through B_7 with B_7 being the most significant bit.

Decimal digits in ASCII can be converted to BCD by removing the three higher order bits, 011.

Example:

ASCII for each symbol is (B_7 B_6 B_5 B_4 B_3 B_2 B_1)

G ← 100 0111, (← 0101000, h ← 110 1000, > ← 011 1110 & so on.

7. List out 5 advantages of ICs.

→ Any five advantages of ICs are as follows:-

a. In consumer electronics, ICs have made possible the development of many new products, including personal calculators and computers, digital watches, and video games.

b. They have also been used to improve or lower the cost of many existing products, such as appliances, televisions, radios and high-fidelity equipment.

c. The logic and arithmetic functions of a small computer can now be performed on a single VLSI chip called a microprocessor.
d. Complete logic, arithmetic, and memory functions of a small computer can be packaged on a single printed circuit board, or even on a single chip.
e. The reduction in power consumption is achieved due to extremely small size IC.

8. What is LSI? How does it differ from VLSI?
→ LSI (Large Scale Integration) is the process of integrating or embedding thousands of transistors on a single silicon semiconductor microchip.

VLSI (Very large scale integration) is the current level of computer microchip miniaturization & refers to microchips containing in the hundreds of thousands of transistors. LSI (Large Scale Integration) meant microchips containing thousands of transistors.

9. Define r's complement and (r-1)'s complement.
→ r's complement (radix complement)
r's complement of a number N is defined as $(r^n - N)$
where N is the given number.
r is the base number system.
n is the number of digits in the given number.
To get r's complement fast, add 1 to the lower order digit of it's (r-1)'s complement.
Ex: 10's complement of 835_{10} is $164_{10} + 1 = 165_{10}$
2's complement of 1010_2 is $0101_2 + 1 = 0110_2$

PAGE NO.:
DATE:

(r-1)'s Complement (diminised radix complement).

(r-1)'s complement of a number N is defined as (r^n-1) - N

where, N is the given number

r is the base of number system.

n is the number of digits in the given number.

To get the (r-1)'s complement fast, subract each digit of a number from (r-1).

Example:

- 9's complement of 835_{10} is 164_{10} (Rule: $(10^n-1)-N$)
- 1's complement of 1010_2 is 0101_2 (bit by bit complement operation).

Write the process to obtain r's complement from (r-1)'s complement with examples.

→ To find r's complement, just add 1 to (r-1)'s complement.

Here is an example:

Q. Find 7's & 8's complement of the number $(563)_8$

Step 1: Identify the base (or radix). Here, r=8.

Step 2: Since 7 is the largest digit in the number system, subract each digit of give number from 7 i.e. if it's a three digit number, subtract the number from 777.

$$\begin{array}{cccc} & 7 & 7 & 7 \\ & (-) & (-) & (-) \\ & 5 & 6 & 3 \\ \hline \rightarrow & 2 & 1 & 4 \end{array}$$

$\therefore (214)_8$ is the 7's complement of given number.

Step 3: To find r's complement i.e. 8's complement then '1' to the result of 7's complement number.

$$\begin{array}{r} 2\ 1\ 4 \\ (+1) \\ \hline \rightarrow\ 2\ 1\ 5 \end{array}$$

$\therefore (215)_8$ is the 8's complement of the given number.

11. Write BCD code, Excess 3-code, 8-4-2-1 code, 2421 code and Biquinary code for 1024.

→ Here, the given number is 1024.

BCD code: (0001 0000 0010 0100)

Excess-3 code: $(1000100001\ 11)_2$

8-4-2-1 code: ~~It's the BCD code~~ 0111 0000 0110 0100

~~(0001000000100100)~~

~~Biquinary code:~~

#84-2-1 code

8	4	-2	-1
0	0	0	0
0	1	0	1
0	1	1	0
0	1	0	1
0	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0

What is parity bit? Explain how even and odd parity is obtained.

→ A parity bit is the extra bit included to make the total number of 1's in the resulting code word either even or odd. Parity bits are used as simplest forms of error detecting code.

In case of even parity, for a given set of bits, the occurrences of bits whose value is 1 is counted. If that count is odd, the parity bit value is set to 1, making the total count of occurances of 1s in the whole set (including the parity bit) an even number. If the count of 1s in a given set of bits is already even, the parity bit's value is 0.

In case of odd parity, the coding is reversed. For a given set of bits, if the count of bits with a value of 1 is even, the parity bit value is set to 1 making the total count of 1s in the whole set, including the parity bit, an odd number. If the count of bits with a value of 1 is odd, the count is already odd so this parity bit's value is 0.

Assignment #2

1. Demonstrate by the means of truth table the validity of following theorems;

a) Associative laws.

→ First Statement of associative laws;

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

Proof using truth table;

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

$\therefore x+(y+z) = (x+y)+z$ ✓

Second statement of associative laws;

ii. $x.(y.z) = (x.y).z$

Proof using truth table;

x	y	z	x·y	y·z	x·(y·z)	(x·y)·z
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	1	0	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

$\therefore x(y \cdot z) = (x \cdot y) \cdot z$ Ans.

b. De-Morgan's law.

Statements of De-Morgan's law:

(i) $(x+y)' = x' \cdot y'$ (ii) $(x \cdot y)' = x' + y'$

Proof of 1st statement using truth table:

x	y	x+y	(x+y)'	x'	y'	x'·y'
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

$\therefore (x+y)' = x' \cdot y'$ Ans.

P.T.O.

Proof of IInd statement using truth table

x	y	x.y	(x.y)'	x'	y'	(x'+y')
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

$\therefore (x.y)' = (x'+y')$ #

2. Simplify the following Boolean functions to a minimum number of literals.

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

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

$= x(y+y')$

$= x$ #

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

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

$= (1+1)'$

$= 1' = 0$

c. $xz+xz'y$.

3. Obtain the truth table of function.

a. $F = xy + xy' + y'z$

x	y	z	xy	xy'	y'z	xy+xy'	xy+xy'+yz
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	0	1	0	1	1
1	0	1	0	1	1	1	1
1	1	0	1	0	0	1	1
1	1	1	1	0	0	1	1

b. $F = xy + x'y' + xz + yz$

→ The truth table for given function is;

x	y	z	x'	y'	xy	x'y'	xz	yz	xy+x'y'+xz+yz
0	0	0	1	1	0	1	0	0	1
0	0	1	1	1	0	1	0	0	1
0	1	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	1
1	0	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	1	0	1
1	1	0	0	0	1	0	0	0	0
1	1	1	0	0	1	0	1	1	1

4. Express the following functions is a sum of minterms and product of maxterms.

a. $F(A,B,C) = ((A'+B)+B'C$

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

$= A'C + BC + B'C$

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

$= m_3 + m_1 + m_7 + m_5$ $A'BC + A'B'C + ABC + AB'C$

$= m_1 + m_3 + m_5 + m_7$

$F(A,B,C) = \Sigma(1,3,5,7)$

Maxterm; $F(A,B,C) = \Pi(0,2,4,6)$

(a) 11011 - 1011

1 0 1 1 free 2' complement

2's complement of 1011

The number of bits in the subtrahend is 4 while that of minuend is 5. We make the number of bits in the subtrahend equal to the minuend by taking a '0' in the sixth place of the subtrahend.

Now 2's complement of 01011 is (10100+1) i.e. 10101.

Adding this with minuend.

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

Carry over (1)

Result of addition.

After dropping the carry over we get the result of subtraction to be 10000.

(b)

1st complement:

1's complement of 01011 is 10100. Hence,

Minued: → 1 1 0 1 1

1's complement of subtrahend → 1 0 1 0 0

Carry over → (1) 0 1 1 1 1

$$\begin{array}{r} 01111 \\ +\quad 1 \\ \hline 10000 \end{array}$$

Hence, the required difference is 10000.

(b) 10011 - 11001.

2's complement:

2's complement of 11001 is (00110+1) i.e. 00111.

$$\begin{array}{r} 00111 \\ +\ 10011 \\ \hline 11010 \end{array} \rightarrow \text{Result of addition.}$$

As there is no carry over, the result of subtraction is negative and is obtained by writing 2's complement of 11010 i.e. (= 00101+1) or 00110.

Hence, the difference is −110.

1's complement :

1's complement of 11001 is 00110. Hence;

Minued − 1 0 0 1 1

1's complement. + 0 0 1 1 0

1 1 0 0 1

Hence, the difference is −110.

(c) 100 − 10101

2's complement:

2's complement of 10101 is (01010+1) ie. 01011

0 1 0 1 1

+ 0 0 1 0 0

0 1 1 1 1

The difference is (10000+1) => −10001

1's complement:

1's complement of 10101 is (01010)

0 1 0 1 0

+ 0 0 1 0 0

0 1 1 1 0

The difference is −10001 #.

(a)

(b)

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

Simplify using K-map.

(a) SOP

$F = xyz + x\bar{y}z + x\bar{z}$

x \ yz	00	01	11	10
0				
1	1	1	1	1

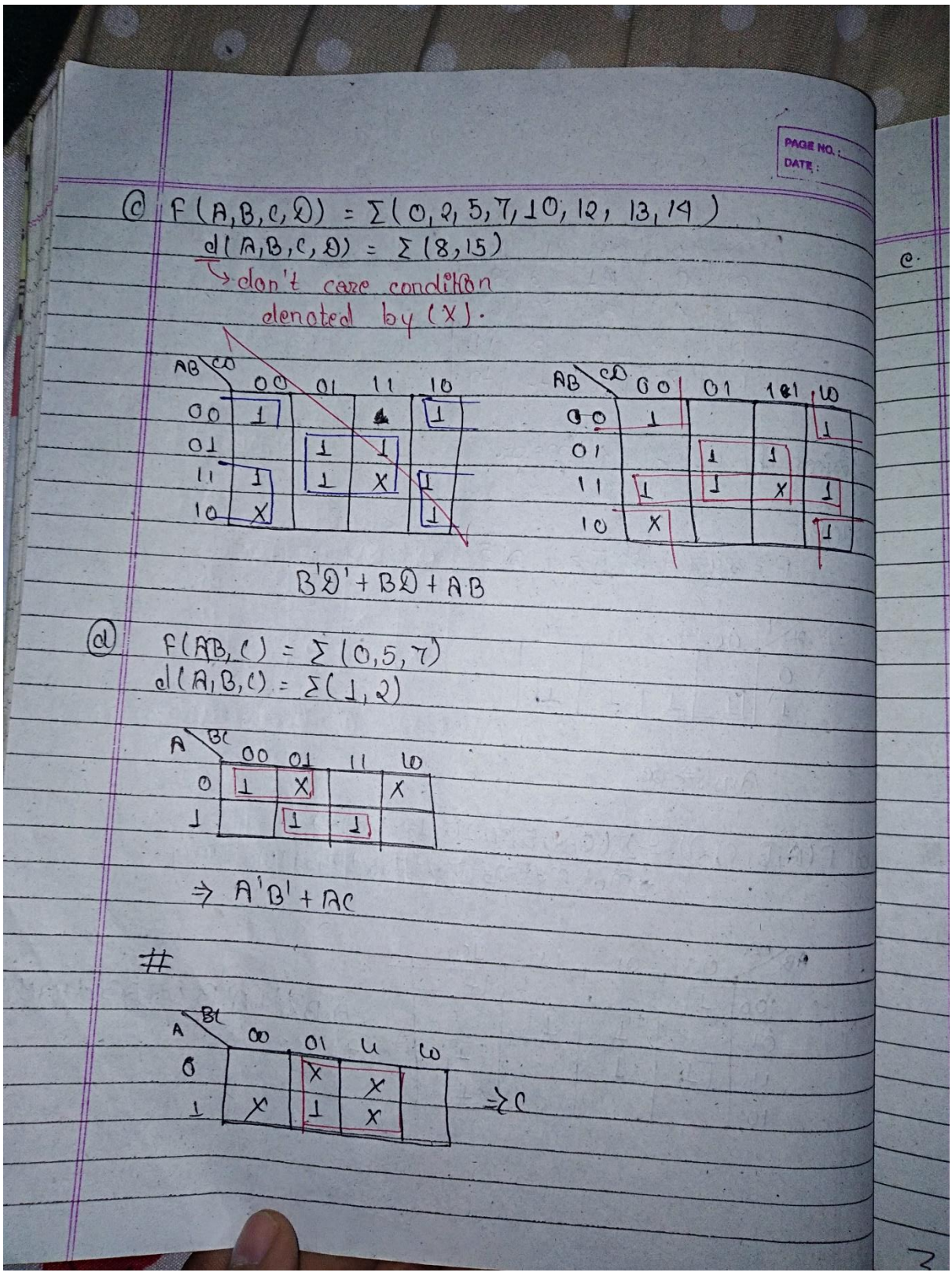
Ans :⇒ x

(b) $F(A,B,C,D) = \sum(0,2,5,7,10,12,13,14)$

$= m_0 + m_2 + m_5 + m_7 + m_{10} + m_{12} + m_{13} + m_{14}$.

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

$A'B'D' + A'BD + ABC' + ACD'$



PAGE NO. :
DATE :

c.

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

$\Rightarrow A'D$

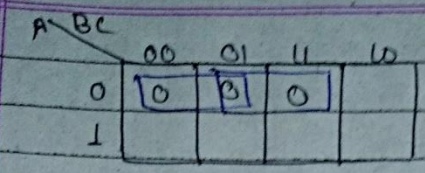
Don't care condition
- use it only for the combination of maximum 1.
- if single X, don't ~~care~~ use it.

POS (Product of Sums)

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

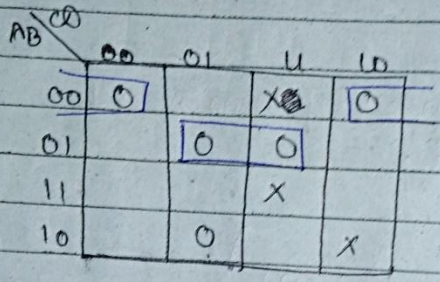
A \ BC	00	01	11	10
A ← 0	0		0	
A' ← 1		0		

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

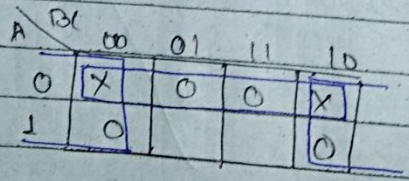


$\Rightarrow (A+B)\,(A+C')$

Q. $F = \Pi(0,2,5,7,9)$
$d(3,10,15)$



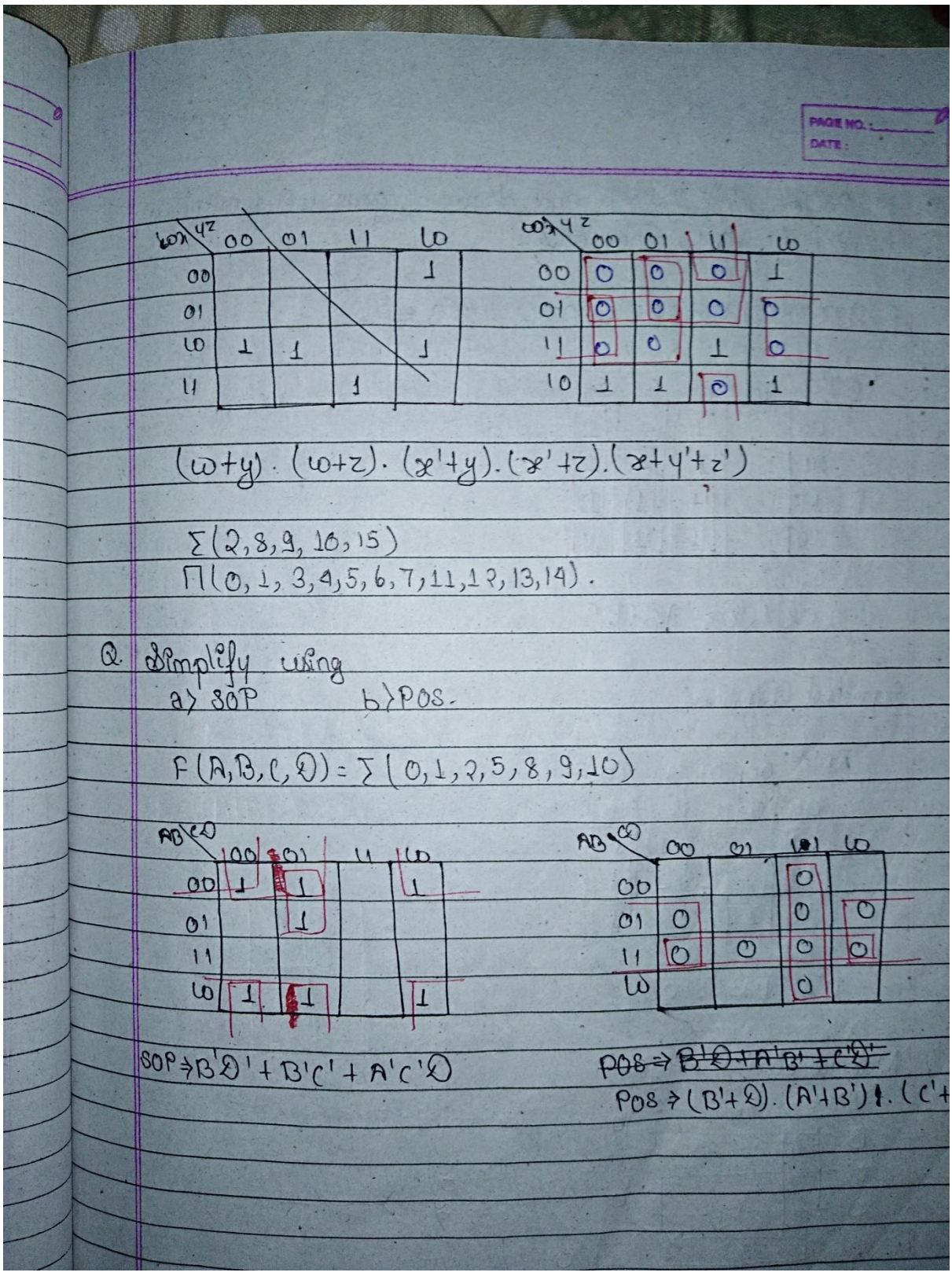
$\Rightarrow (A+B'+D')\,(A+B+D)\,(A'+B+C+D')$



$\Rightarrow C \cdot \bar{A}$

Q. Reduce into POS form:

$F = \omega xyz + \bar{x}y\bar{z} + \omega\bar{x}y\bar{z} + \omega\bar{x}\bar{y}z$



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

$\Sigma(2,8,9,10,15)$

$\Pi(0,1,3,4,5,6,7,11,12,13,14)$.

Q. Simplify using
a) SOP b) POS.

$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$

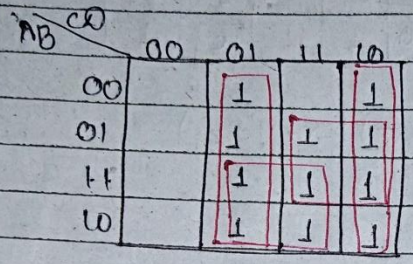
SOP $\Rightarrow B'D' + B'C' + A'C'D$

~~POS $\Rightarrow B'D + A'B' + C'D'$~~

POS $\Rightarrow (B'+D).(A'+B').(C'+$

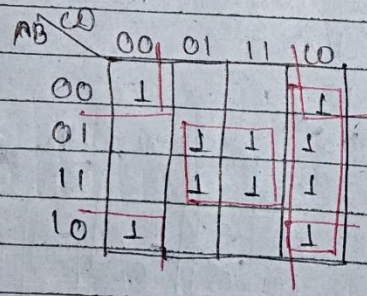
Q. Minimize using k-map and draw diagram using minimum number of NAND gates only.

$F = ABC + BC'D + CD' + AB'D + A'BCD + A'B'C'D$



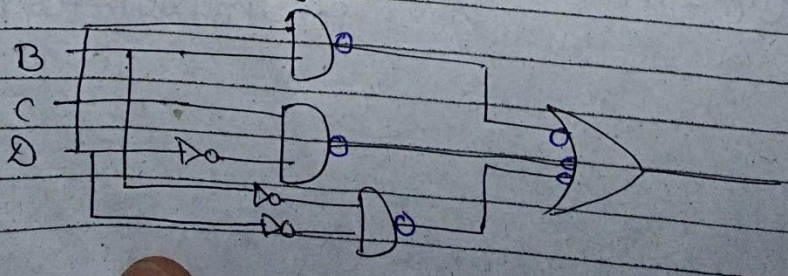
$\Rightarrow C'D + CD' + AD + BC$

Expected Question:



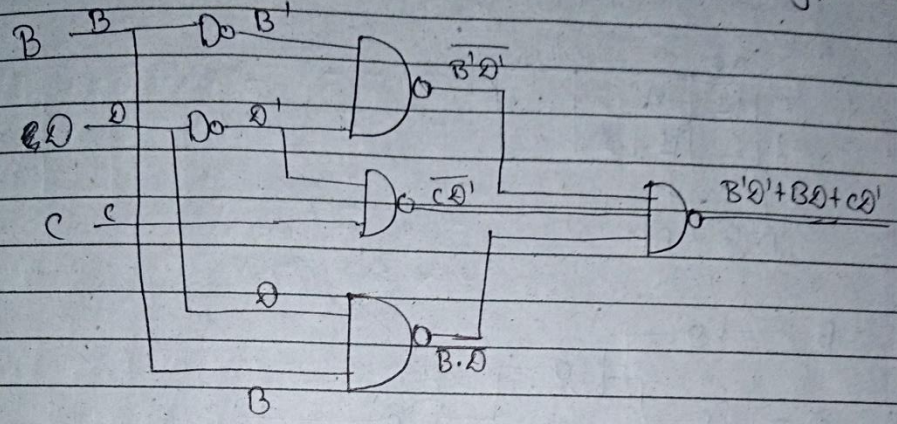
$B'D' + BD + CD'$

Normal circuit diagram:



min. no. of NAND → 8 OP
" " " NOR → 5 Ops

Circuit diagram using minimum no. of NAND Gates only;

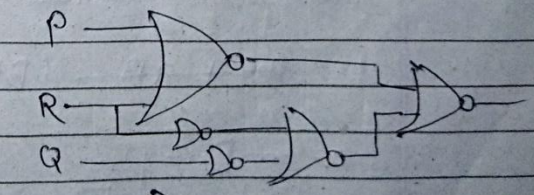


Q. Draw circuit diagram to represent K-map using minimun no. of NOR gates.

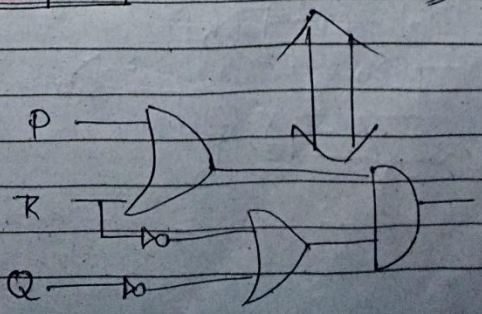
R \ PQ	00	01	11	10
0		X	1	1
1	1		X	1

→ The req. K-map is

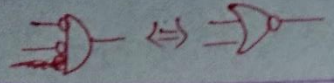
R \ PQ	00	01	11	10
0	0	X		
1		0	X	



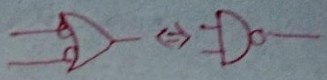
(R+P).(R'+Q')



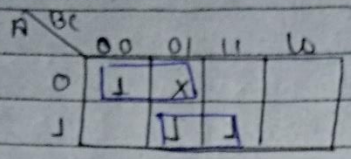
Invert AND ⇔ NOR gate



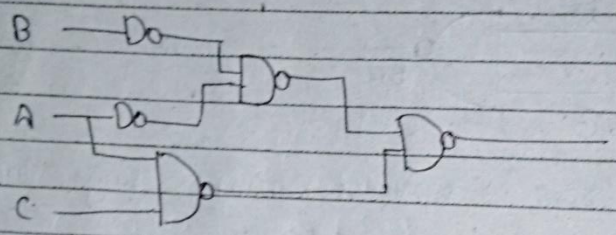
Invert OR ⇔ NAND gate



Q Draw circuit using minimum no. of NAND Gates.

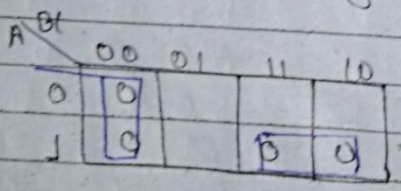


$A'B' + AC$

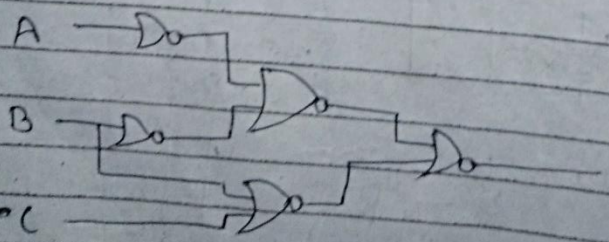


Q. Draw a circuit using minimum number of NOR Gates.

Ans.



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



b.

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

→ soln,

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

$(B+C).(A'+B') \Rightarrow$ same as (a).

Q. Case study:
Draw a circuit for Error Detection code using odd parity

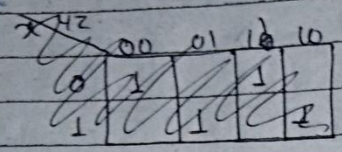
→ soln,

x	y	z	Odd parity	
0	0	0	1	← $\bar{x}.\bar{y}.\bar{z}$
0	0	1	0	
0	1	0	0	
0	1	1	1	← $\bar{x}.yz$
1	0	0	0	
1	0	1	1	← $x.\bar{y}.z$
1	1	0	1	← $x.y.\bar{z}$
1	1	1	0	

x \ yz	00	01	11	10
0	1		1	
1		1		1

$\Rightarrow \bar{x}.\bar{y}\bar{z} + \bar{x}yz + x\bar{y}z + xy\bar{z}$

$\bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}z + xy\bar{z}$



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

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

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

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

Parity Checker

w	x	y	z	P_{check} (odd)	
0	0	0	0	1	$\rightarrow \bar{w}\bar{x}\bar{y}\bar{z}$
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	$\rightarrow \bar{w}\bar{x}yz$
0	1	0	0	0	
0	1	0	1	1	$\rightarrow \bar{w}x\bar{y}z$
0	1	1	0	1	$\rightarrow \bar{w}xy\bar{z}$
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	1	$\rightarrow w\bar{x}\bar{y}z$
1	0	1	0	1	$\rightarrow w\bar{x}y\bar{z}$
1	0	1	1	0	
1	1	0	0	1	$\rightarrow wx\bar{y}\bar{z}$
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	$\rightarrow wxyz$

$\bar{w}\bar{x}(\bar{y}\bar{z}+yz)+\bar{w}x(\bar{y}z+y\bar{z})+w\bar{x}z(\bar{y}+y)+wx(\bar{y}\bar{z}+yz)$

$= \bar{w}\bar{x}(y\odot z)+\bar{w}x(y\oplus z)+w\bar{x}z+wx(y\odot z)$

$= (y\odot z)(\bar{w}\bar{x}+wx)+\bar{w}x(y\oplus z)+w\bar{x}yz$

$= (y\odot z)(w\odot x)+\bar{w}yx(y\oplus z)+w\bar{x}yz$

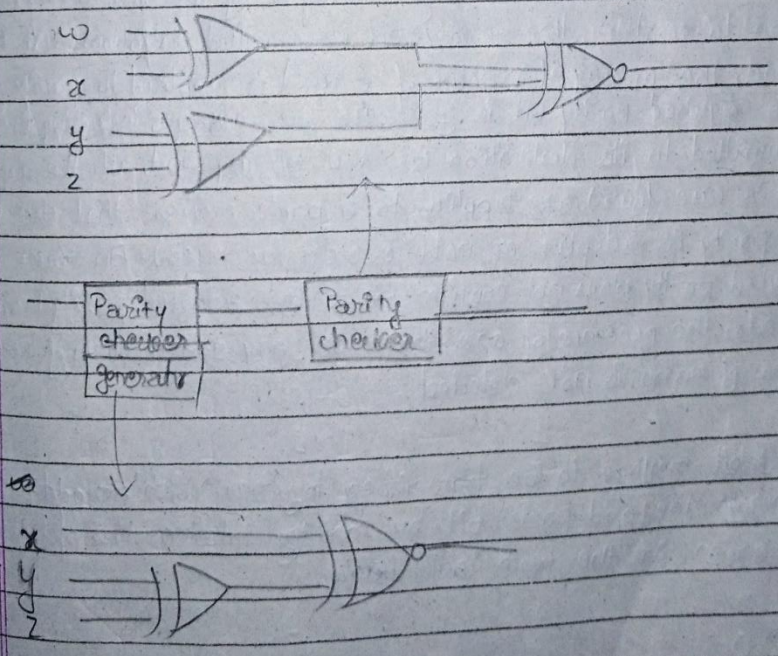
$= w'x'(y'z'+yz)+w'x(y'z+yz')+wx'(y'z+yz')+wx(y'z'+yz)$

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

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

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

$= (y\oplus z)\odot(w\oplus x)$



PAGE NO. :
DATE :

Draw a circuit for error detection code using add ~~even~~ parity bit.

→

(1) Introduction.

Binary information transmitted through some form of communication medium is subjected to external noise that could change bits from 1 to 0 and vice versa. An error detection code is a binary code that detects digital errors during transmission. The detected errors cannot be corrected but their presence is indicated. The most common error detection code used is the parity bit. A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even. During transfer of information from one location to another, the parity bit is handled as follows. At the sending end, the message (in case of 3 bits) is applied to parity generator where the required parity bit is generated. The message, including the parity bit, is transmitted to its destination. At the receiving end, all the incoming bits (in this case four) are applied to a parity checker that checks whether the proper parity is adopted or not (i.e. odd or even). An error is detected if the checked parity does not confirm ~~the~~ to the adopted parity. The parity method detects the presence of one, three or any odd numbers of errors only. An even no. of errors is not detected..

(2) Consider a 3-bit message to be transmitted together with an odd parity bit. We generate the odd parity bit by making total number of bits odd which is shown in the truth table below;

Three-Bit Message			Parity bit (Odd)
x	y	z	P_{odd}
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

fig: Odd parity generator truth table.

From above truth table, the simplified expression of odd parity bit generator can be written as:

$$\begin{aligned} P_{odd} &= \bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}z + xy\bar{z} \\ &= \bar{x}(\bar{y}\bar{z} + yz) + x(\bar{y}z + y\bar{z}) \\ &= \bar{x}\,\overline{(y \oplus z)} + x(y \oplus z) \\ &= \bar{x}\,\overline{(y \oplus z)} + x(y \oplus z) \\ &= x \odot (y \oplus z) \end{aligned}$$

The above expression can be implemented by using one X-OR gate and one XNOR gate in order to design a 3-bit odd parity generator.

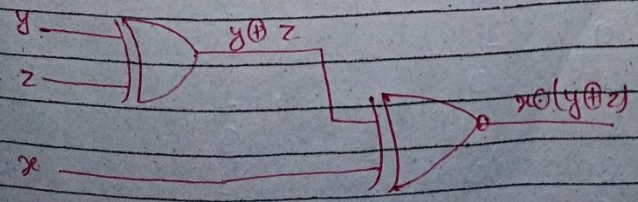


fig: logic diagram of odd parity generator.

PAGE NO. :
DATE :

Combinational circuit.

i. adder / substracter	defn
ii. Multiplexer / deMux	Block diagram
iii. Encoder / decoder	Equation
iv. code converter	circuit diagram
v. Magnitude comperators.	Truth table.

Now, the three bits in the message together with the parity bit generated are transmitted to their destination, where they are applied to a parity checker circuit to check for possible errors in transmission (only for checking not correction of errors).

Four bits Recieved				Parity Error Check.
w	x	y	z	Odd
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

From above truth table the simplified expression of odd parity bit checker can be written as

$$\begin{aligned}
C_{odd} &= \bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}xyz + w\bar{x}\bar{y}z + w\bar{x}y\bar{z} + wx\bar{y}\bar{z} + wxyz \\
&= \bar{w}\bar{x}(\bar{y}\bar{z} + yz) + \bar{w}x(\bar{y}z + y\bar{z}) + w\bar{x}(\bar{y}z + y\bar{z}) + wx(\bar{y}\bar{z} + yz) \\
&= \bar{w}\bar{x}(y \odot z) + \bar{w}x(y \oplus z) + w\bar{x}(y \oplus z) + wx(y \odot z) \\
&= (\bar{w}\bar{x} + wx)(y \odot z) + (\bar{w}x + w\bar{x})(y \oplus z) \\
&= \overline{(w \oplus x)}(y \odot z) + (w \oplus x)(y \oplus z) \\
&= \overline{(w \oplus x)}\,\overline{(y \oplus z)} + (w \oplus x)(y \oplus z) \\
&= (w \oplus x) \odot (y \oplus z)
\end{aligned}$$

The above expression can be implemented by using two X-OR gate and one ex-NOR gate in order to design a 3-bit parity checker

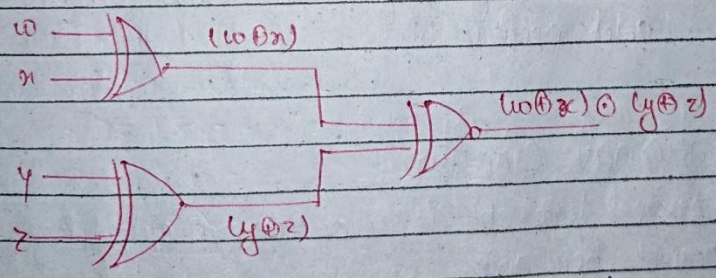


fig: logic diagram of odd parity checker.

~~Odd~~

Odd parity generator and odd parity checker as a whole in a single figure is shown below;

P.T.O.

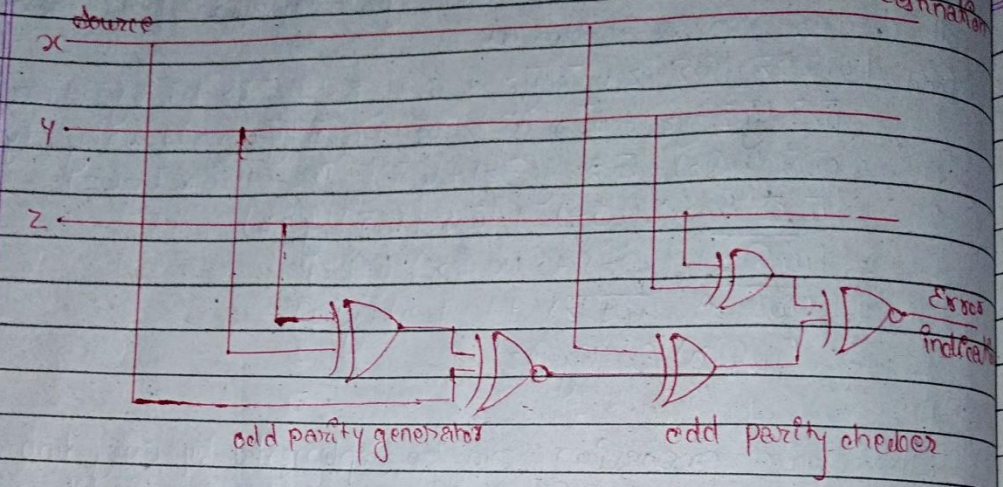
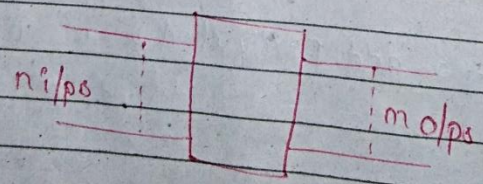


fig: Combined logic diagram for error detection with odd parity bit.

Combinational Circuits.

Combinational logic cbt is a ckt where output is a pure function of present input only.



Characteristics of combinational ckts.

1. Output is dependent on present input only.
2. For 'n' number of inputs there are 'm' numbers of outputs
3. There is no memory elements / flip flops.

PAGE NO. :
DATE :

Adders

→ digital circuits that perform additions.
→ adders reside in ALU.

Types of adders

a. Half-adder
   ↳ sum & carry
   ↳ add^n of 2 bits

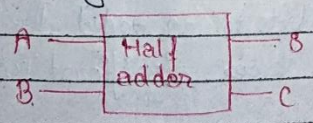
b. Full-adder
   ↳ sum & carry
   ↳ 2-half adders
   ↳ add^n of 3 bits.

Half adder

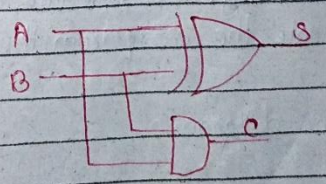
A combinational circuit that needs two inputs(augend & addend) and gives two outputs(sum and carry) to perform the addition of two bits is called half adder.

let, A and B be augend & addend respectively.
S and C be sum & carry respectively.

Block Diagram



Logic circuit.



Truth Table;

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

P.T.O.

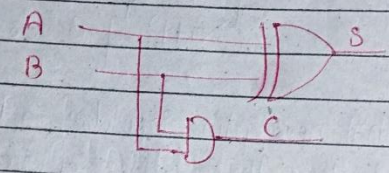
Boolean functions for the two outputs can be obtained directly from the truth table.

The simplified sum of products expressions are:

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

$C = AB$

Logic circuit:

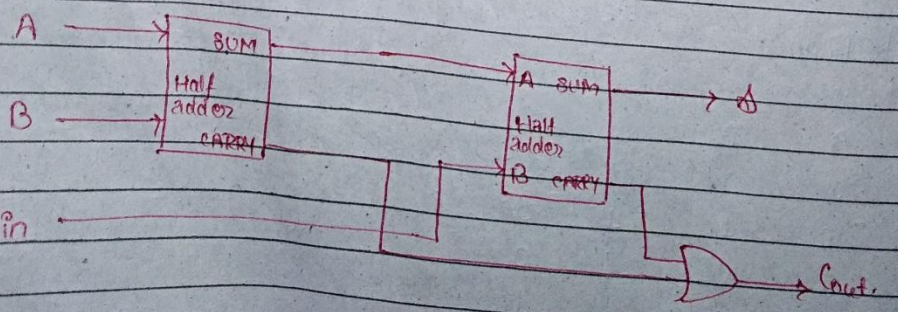


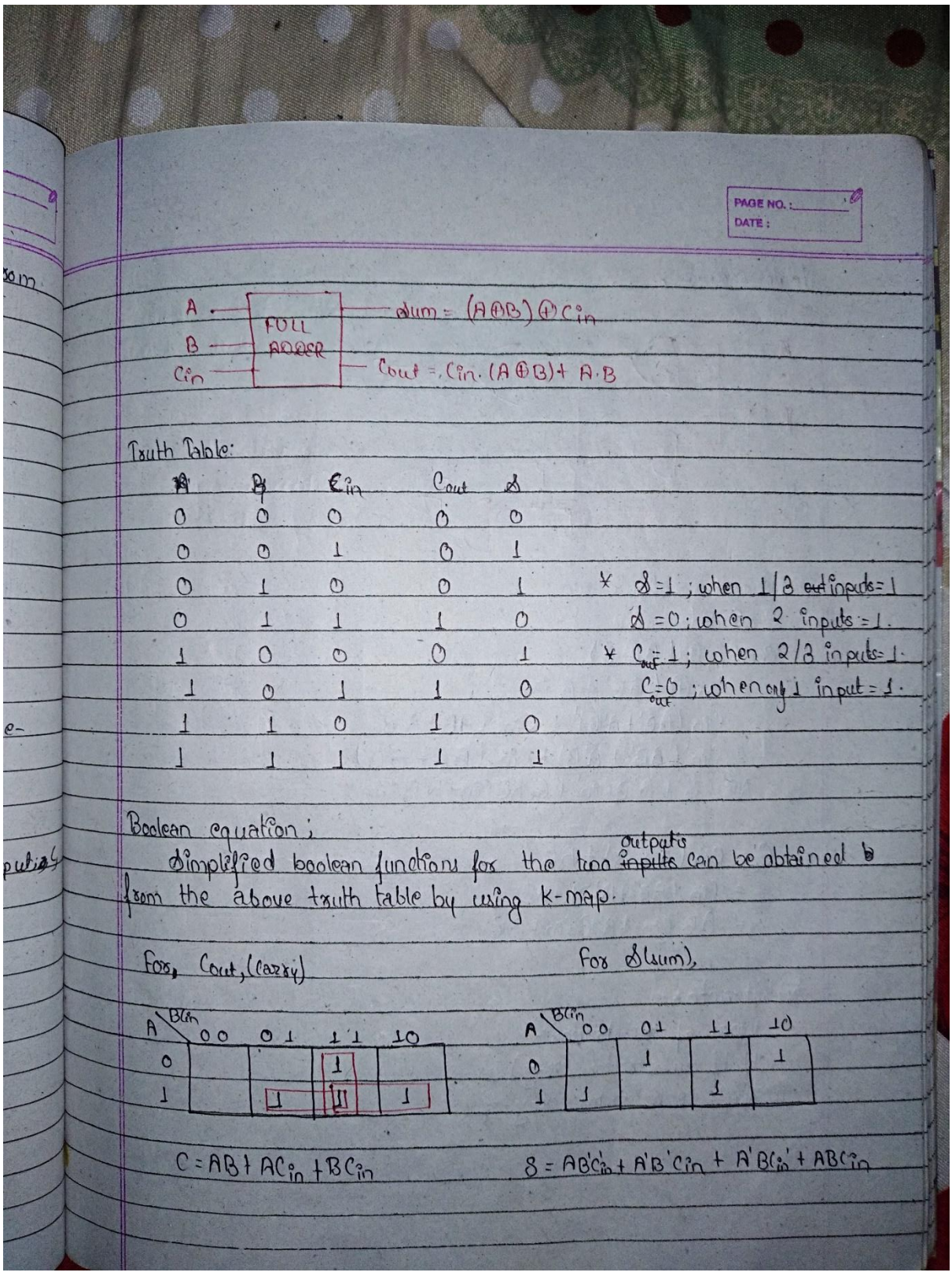
Full adder.

A full adder is a combinational circuit that forms the arithematic sum of three input bits (3 inputs & 2 outputs).

Let A, B, C are three inputs and S, C are two outputs. Here, A, B represent two significant bits to be added. The third output represents the carry from the previous lower significant position.

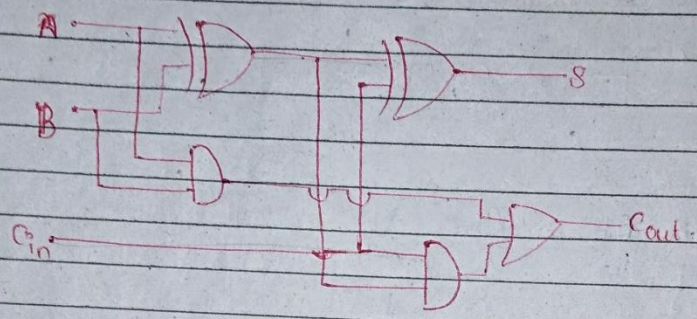
Block diagram:





PAGE NO. :
DATE :

Logic circuit



Here, the outputs are;

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

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

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

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

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

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

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

$= AB'C_{in} + A'BC_{in} + AB$

Substractor

Types of substractor:-

i. Half substractor (x-y)

ii. Full substractor (x-y-z)

PAGE NO. :
DATE :

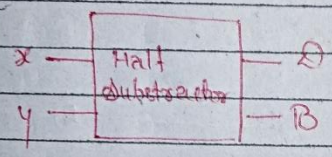
Half substractor

A half-substractor is a combinational circuit that substracts two bits and produces their difference bit.

Taking two inputs x and y where x is minuend & y is substrahend.

let, D (difference) and B (borrow) are the two outputs.

Block Diagram:



Truth Table

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

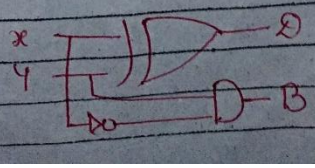
Boolean functions:

The Boolean functions for the two outputs of the half substractor are derived directly from truth table:

$$B = x'y$$

& $$D = x'y + xy' = x \oplus y$$

Logic Circuit:



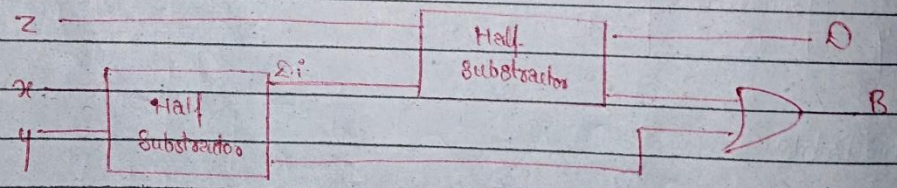
PAGE NO. :
DATE :

Full-Substractor. (x-y-z)

A full substractor is a combinational circuit that performs a substraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage.

This circuit has three inputs and two outputs. The three inputs x, y, z denoted the minuend, subtrahend, previous borrow, respectively. The two outputs, D and B, represent difference and output-borrow respectively.

Block diagram:



Truth Table:

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

PAGE NO.:
DATE :

Boolean functions:

The simplified Boolean functions for the two outputs of the full-substractor are derived in the following K-maps:

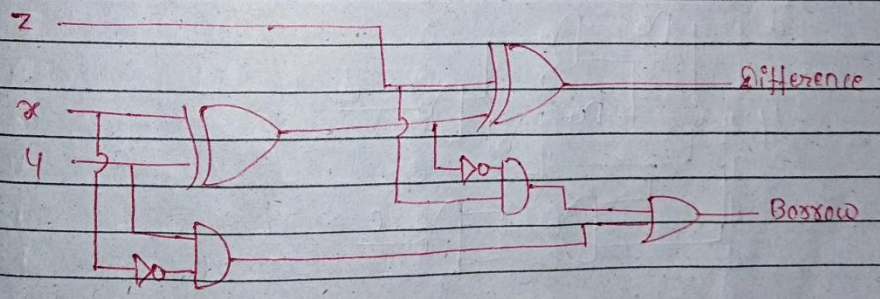
x \ yz	00	01	11	10
0		1		1
1	1		1	

x \ yz	00	01	11	10
0		1	1	1
1			1	

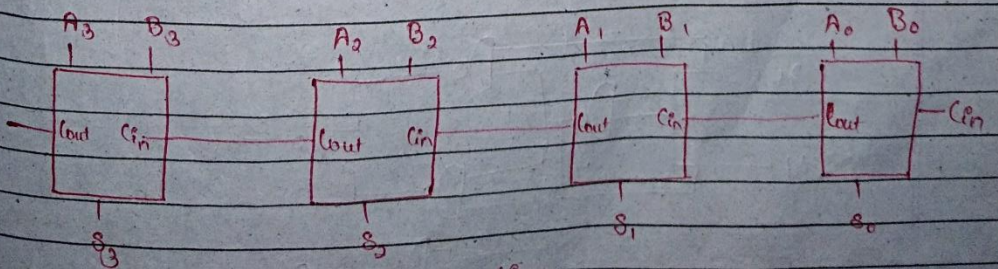
$D = x'y'z + x'yz' + xy'z' + xyz$

$B = x'y + x'z + yz$

Circuit diagram:



Binary Parallel adder / Ripple adder.



4-bit parallel binary adder

Past Question: Design Half adder using Universal gate.

→ Half adder:

It is a combinational circuit that performs the addition of two bits.

The simplified sum of product expressions are;

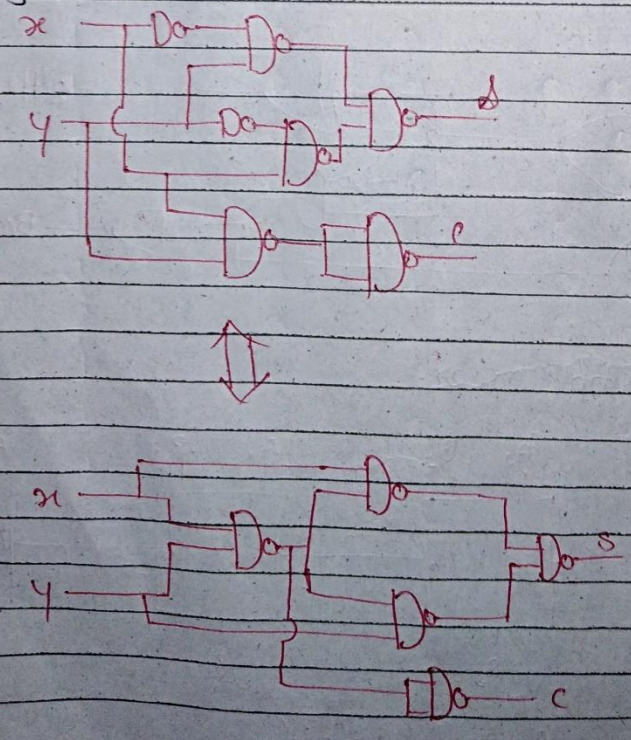
$$S = x \oplus y = x'y + xy'$$

and $C = xy$

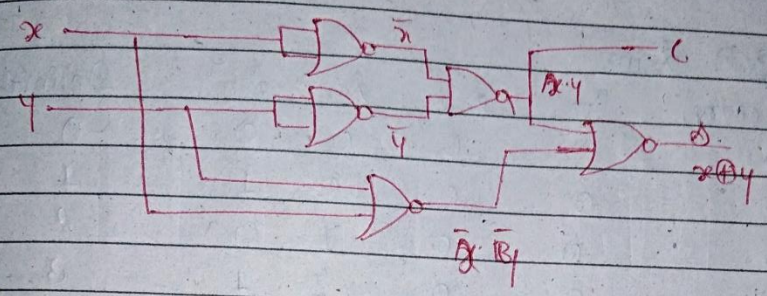
where x & y are two inputs of the circuit and S and C are the outputs.

The circuit diagram of half adder is shown below;

Using NAND Gate,



Using NOR Gate:



Decimal Adder

Decimal adder is a combinational circuit that sums up two decimal numbers adopting particular encoding technique. It has minimum of 9 inputs and 5 outputs; 4 bits ~~req~~ is required to code a decimal digit and the circuit must have an input carry and output carry.

BCD Adder

This combinational ckt adds up two decimal numbers encoded in BCD form. Adding two decimal digits in BCD, together with a possible carry, the output sum cannot be greater than $9+9+1=19$.

PAGE NO.:
DATE :

Q. Construct Decimal Adder (~~Binary~~ also BCD Adder)

→ Soln.

BCD Sum

C (Output carry)	S_3	S_2	S_1	S_0	Decimal
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
1	0	0	0	0	8
1	0	0	0	1	9
1	0	0	1	0	10
1	0	0	1	1	11
1	0	1	0	0	12
1	0	1	0	1	13
1	0	1	1	0	14
1	0	1	1	1	15.
1	1	0	0	0	16
1	1	0	0	1	17

PAGE NO.:
DATE :

Code Conversion

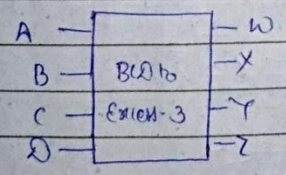
A code converter is a circuit that makes the two systems compatible even though each uses a different binary code.

BCD to Excess-3 Converter

BCD code: In this code, each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code.

Excess-3: It is an unweighted self-complementary BCD code. The self complementary property means that the 1's complement of the excess-3 is the excess-3 code of the 9's complement of the corresponding decimal number.

Block diagram

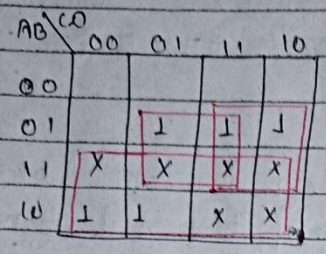


Truth Table:

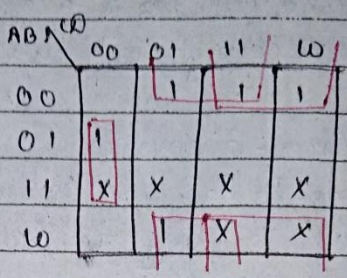
Decimal Digit	Input BCD				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

4 BCD input variables may have 16 bit combinations, 10 are listed here. Others denote "don't care conds."

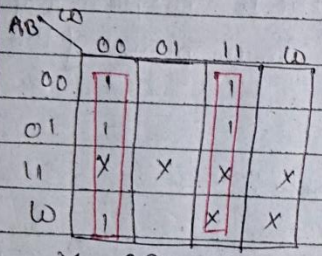
Boolean Equation:



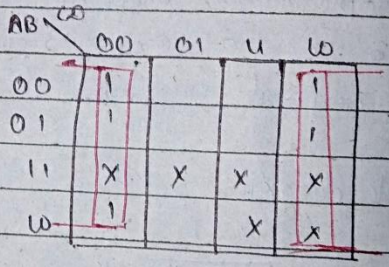
$W = A + B\bar{C} + B\bar{D}$



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

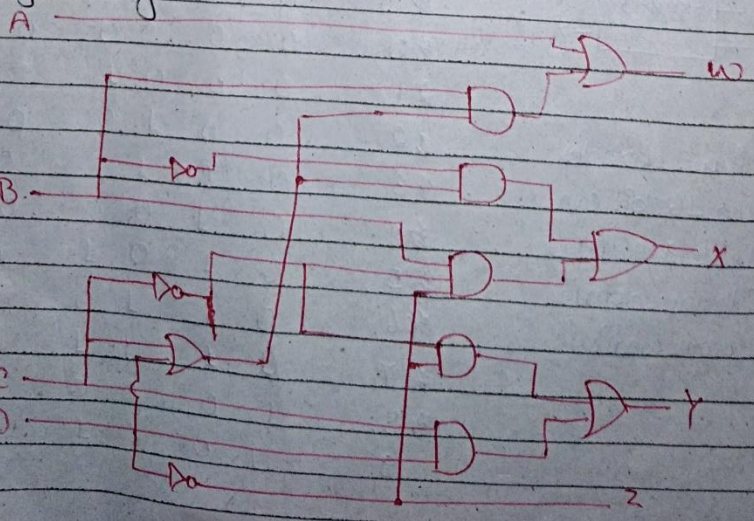


$Y = CD + \bar{C}\bar{D}$



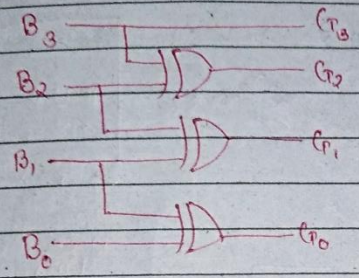
$Z = \bar{D}$.

Logic Diagram



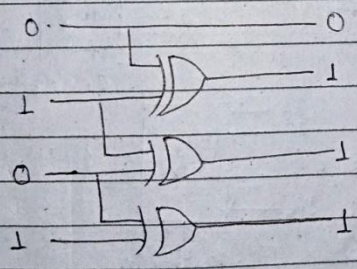
PAGE NO. :
DATE :

BCD to Gray:



Q. Convert binary number. 0101 to Gray code with X-OR Gate.

→ Soln,

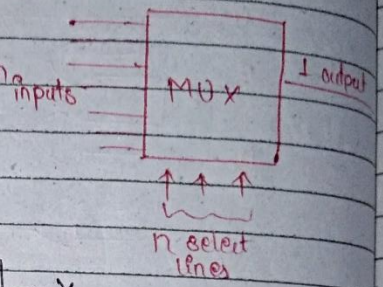


Answer: 0111 #

PAGE NO.:
DATE:

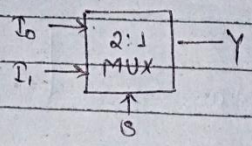
MUX (Multiplexer)

A combinational circuit having 2^n input and 1 output is selected on n select lines.

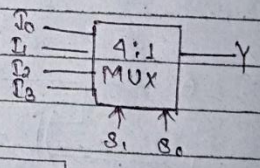


Examples:

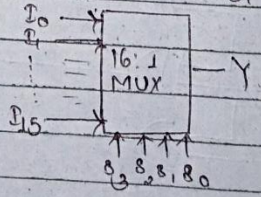
2:1 MUX → 2 i/p lines, 1 output
2 i/p lines
1 select line



4:1 MUX → 4 i/p lines, 1 output
2 select line



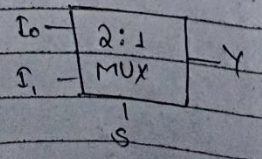
16:1 MUX → 16 i/p lines, 1 o/p
4 i/p ~~lines~~ select line



2:1 MUX

A combinational circuit having 2 inputs and 1 output which is selected on 1 select lines.

Block diagram

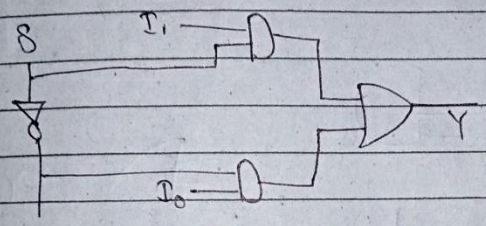


Truth Table

S	Y
0	I_0
1	I_1

Equation : $Y = \bar{S} I_0 + S I_1$

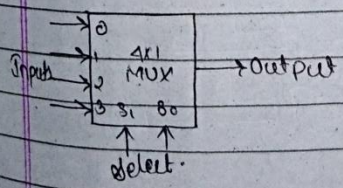
Circuit Diagram



4:1 MUX

A 4:1 MUX is a combinational ckt that selects binary info from one of ~~two~~ 4 input lines and directs it to single output line w.r.t. 2 select

Block diagram :



Truth Table :

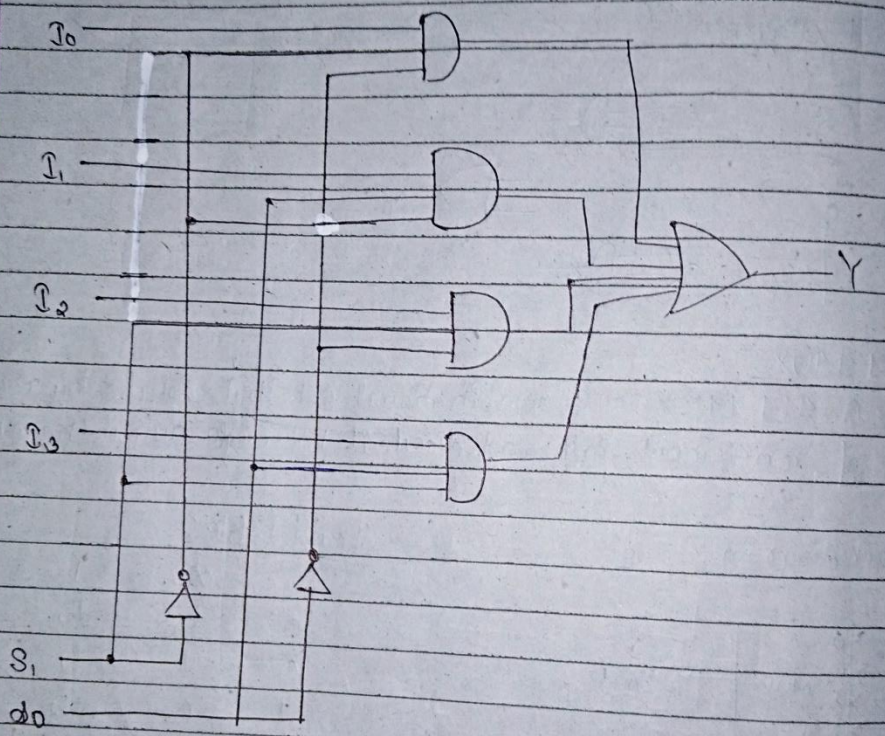
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Table: function table.

Equation :

$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$

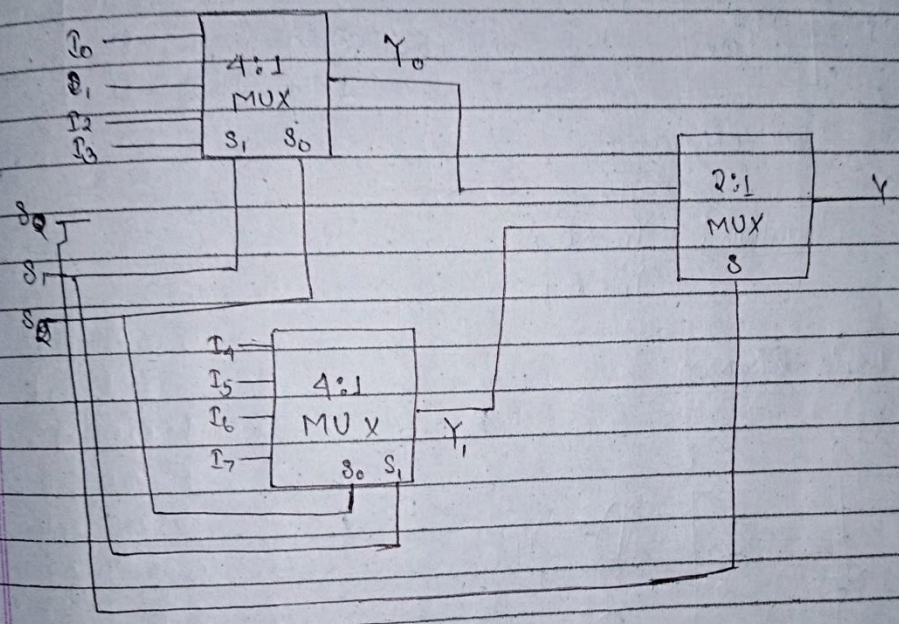
Circuit Diagram.



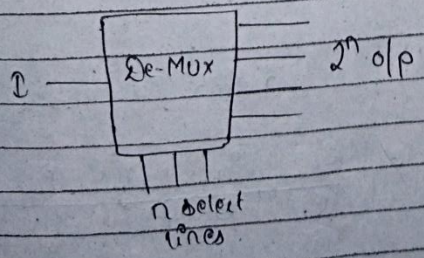
Logic diagram: 4-to-1 line Multiplexer.

PAGE NO. :
DATE :

Draw block diagram of 8:1 MUX using 4:1 MUX.



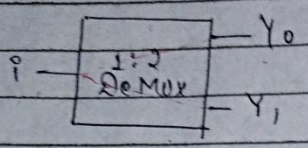
De-Multiplexer



A combinational circuit having 1 input and 2^n output lines ~~with~~ are selected on n select lines.

PAGE NO.:
DATE :

1:2 DeMUX



Block diagram

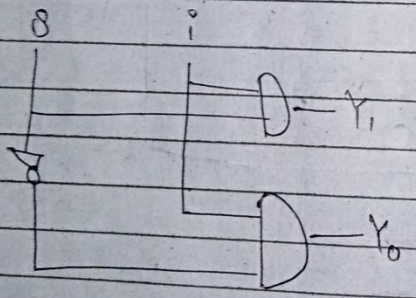
Truth Table.

S	Y_0	Y_1
0	i	0
1	0	i

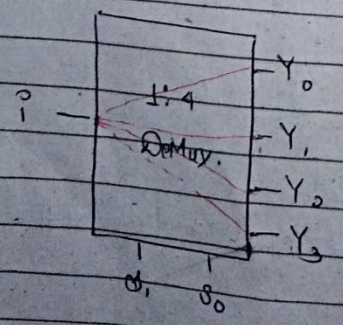
Equation: $Y_0 = \bar{S} i$

$Y_1 = S i$

Logic Diagram



1:4 DeMux.



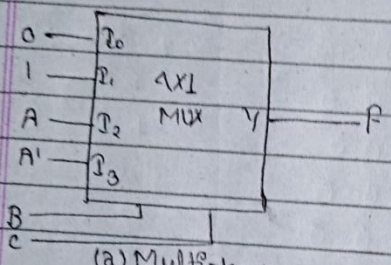
Function Table.

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

Implement using Boolean function $F(A,B,C) = \Sigma(1,3,5,6)$ using multiplexer.

→ Soln,

The function can be implemented with a 4:1 multiplexer, as shown in fig below. Two of the variables, B and C, are applied to the select lines in that order, ie B is connected to S_1 and C to S_0. The inputs to the multiplexer are 0, 1, A and A'.



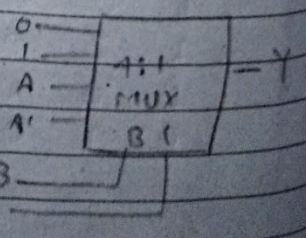
(a) Multiplexer implementation.

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

(b) Truth table

Implementation Table:

	I_0	I_1	I_2	I_3
A'	0	(1)	2	(3)
A	4	(5)	(6)	7



Rules: (i) If both not circled, input = 0
(ii) If both circled, input = 1
(iii) If $\bar{A}$ is circled, input = $\bar{A}$
(iv) If A is circled, input = A

PAGE NO. :
DATE :

Magnitude Comparator

→ compares bits

For 1-bit magnitude comparator,

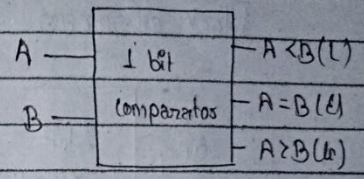
i/p is A & B

So, total combination of output is,

A<B (L)

A=B (E)

A>B (G)



Example: $L = A'B$

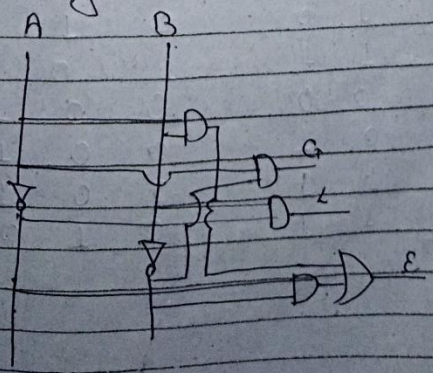
$E = A'B' + AB = A \odot B$

$G = AB'$

→ Soln,

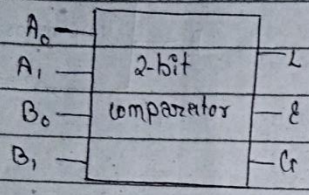
A	B	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Circuit diagram:



2-bit Comparator:

Block diagram

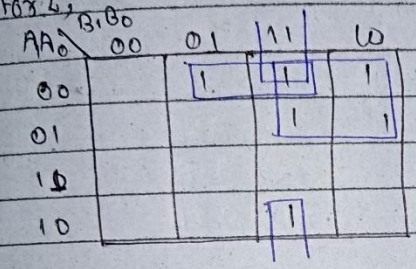


Minterm	A_1	A_0	B_1	B_0	L	E	G
0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	1	0	0
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	0	1	0

PAGE NO. :
DATE :

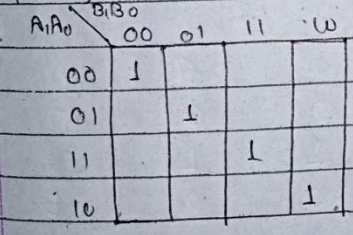
Boolean eqns;

For L,

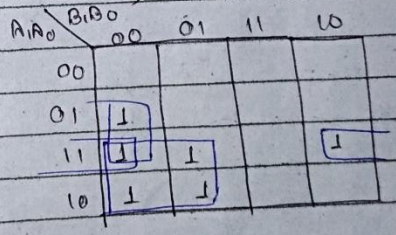


$L \Rightarrow \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0 + \bar{A}_1B_1$

For E,



For G,



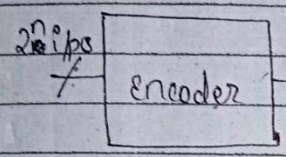
$E \Rightarrow \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$G = A_0\bar{B}_1\bar{B}_0 + A_1\bar{B}_1 + A_1A_0\bar{B}_0$

PAGE NO. :
DATE :

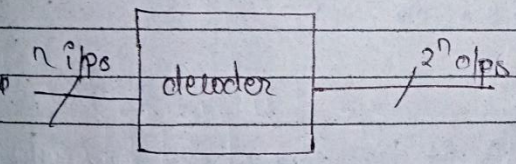
Encoder / Decoder

Encoder



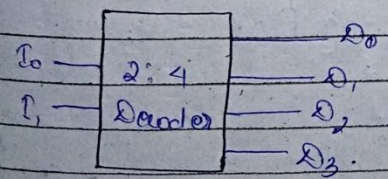
eg: 4:2 encoder
8:3 encoder
16:4 encoder

Decoder



eg: 2:4 decoder
3:8 decoder
4:16 decoder.

2:4 Decoder

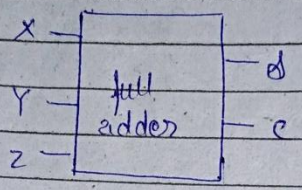


I_1	I_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$D_0 = \overline{I_1}\,\overline{I_0}$; $D_1 = \overline{I_1} I_0$; $D_2 = I_1 \overline{I_0}$ $D_3 = I_1 I_0$

Implement a full adder circuit with a decoder.

⇒ Block diagram of full adder



Truth table for full adder

Minterm	x	y	z	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

From the truth table of full adder, we obtain the function of for this combinational circuit in sum of minterms as:

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

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

Since, there are three inputs and a total of 8 minterms, we need a 3-to-8 line decoder

Fig: Implementation of full adder with a decoder circuit.

Combinational circuit

The output depends on present i/p only.

* No memory elements
* eg: Adders, Substractors etc.

Sequential circuit.

The present output depends on present i/p & past o/p.

* Memory element is present to store.
* eg: flip flops, etc.

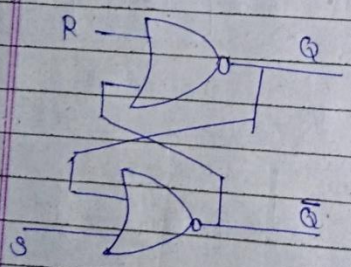
Types of sequential circuit:

1. Synchronous circuit.
2. Asynchronous circuit.

Imp. (long)

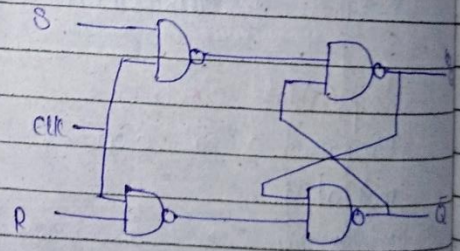
Synchronous	Asynchronous
1. These circuits are easy to design.	1. Difficult to design.
2. A clocked flipflop acts as memory element.	2. An unclocked flipflop or time delay element is used as memory element.
3. They are slower.	3. They are faster as clock is not present.
4. The status of memory element is affected only at the active edge of clock if i/p is changed.	4. The status of memory element change any time as soon as i/p changed.
5. Flipflops are examples of ~~seq~~ synchronous seq. ckt.	5. Latches are examples of asynchronous sequential ckt.

SR latch.

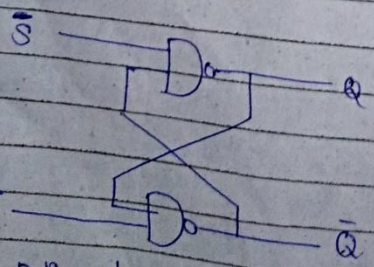


Active high SR latch.

SR flipflop.

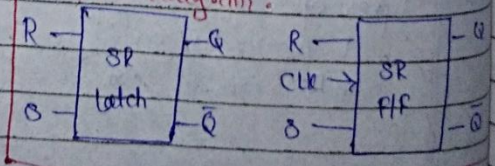


SR flipflop.



Active low SR latch.

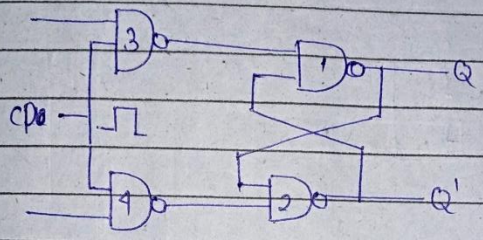
Block diagram:



PAGE NO.:
DATE:

SR flipflop / RS flipflop.

It is a bistable flipflop.



(a) logic diagram.

Characteristic table (Truth Table):

Q(t) (past o/p)	S (present i/p)	R (present i/p)	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

⇕

S	R	Q(t+1)
0	0	Q(t) (No change)
0	1	0 (Reset)
1	0	1 (Set)
1	1	? (Unpredictable)

In seq. ckt, present o/p = combination of past o/p & present i/p.

Disadvantage of SR F/F.
↳ we should never design a ckt with both i/p equal to 1, otherwise creates ambiguity problem.

#SR flip flop is the an important circuit because all other flip flops are constructed from it.

Characteristic equation.

From the truth table,

Q(t) \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

$Q(t+1) = S + R'Q$

$SR = 0$

2. flip flop

→ One way to remove ambiguity problem of SR flip flop is never make a condition, wherein both S & R is equal to 1.

→ D flip flop also solves it by making only one input.

Block diagram.

Circuit logic diagram.

PAGE NO.:
DATE:

Characteristic Equation Table:

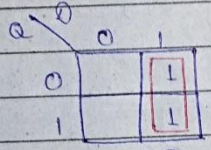
Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

⟺

D-Flipflop

D	Q(t+1)
0	0 (Reset)
1	1 (Set)

Characteristic equation.



Q(t+1) = D

Summary
Q(t+1) = D
Next o/p is same as present D i/p.

never

As long as Clk is 0, the outputs of gates 3 & 4 are the level 1 and the circuit cannot change state regardless the value of D.
The D input is sampled when Clk = 1
→ If D is 1, the Q output goes to 1, placing the ckt in the set state.
→ If D is 0, output Q goes to 0, the circuit switches to clear state.

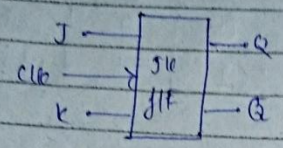
PLA (Programmable Logic Array)
It has programmable AND gate followed by programmable OR Gate.

PAL (Programmable Array Logic)
It has programmable AND gate followed by fixed OR gate.

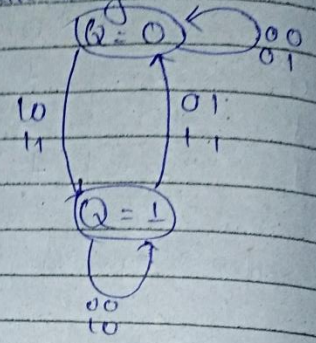
PAGE NO.:
DATE :

JK Flipflop.

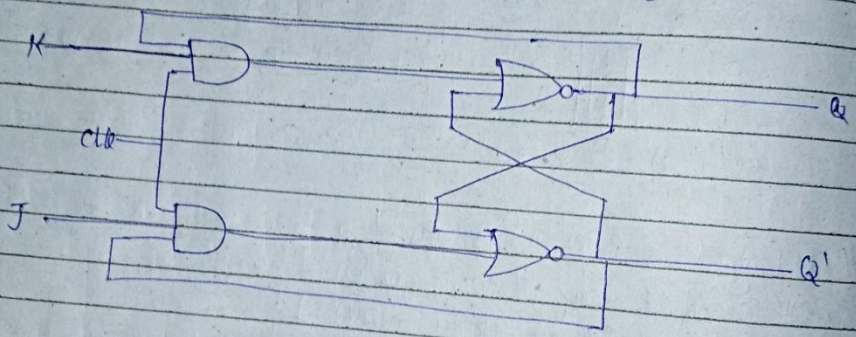
Block diagram



State diagram



~~Circuit~~ Logic Diagram



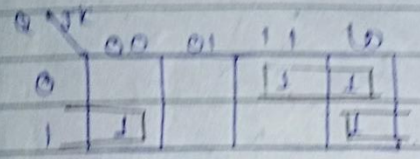
Characteristic Table:

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

JK Flipflop.

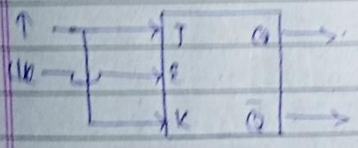
J	K	Q(t+1)
0	0	Q(t) No change
0	1	0 Reset
1	1	1 Set
1	0	Q'(t) Complement

Characteristic Flip flop Equation



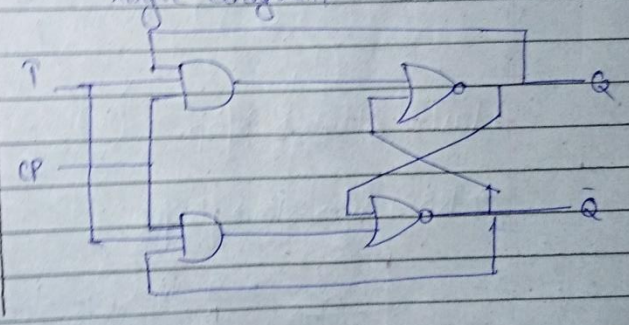
$Q(t+1) = JQ' + K'Q$

T Flip flop



Block diagram

Logic diagram



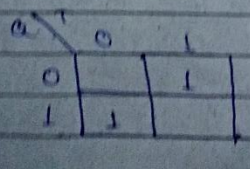
Characteristic table

Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

T-flip flop

T	Q(t+1)
0	Q(t) (No change)
1	Q'(t) (Complement)

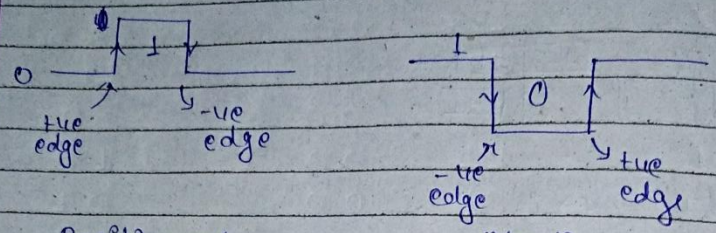
Characteristic Equation



$Q(t+1) = TQ' + \bar{T}Q$

PAGE NO. :
DATE :

Triggering of Flip-flops.

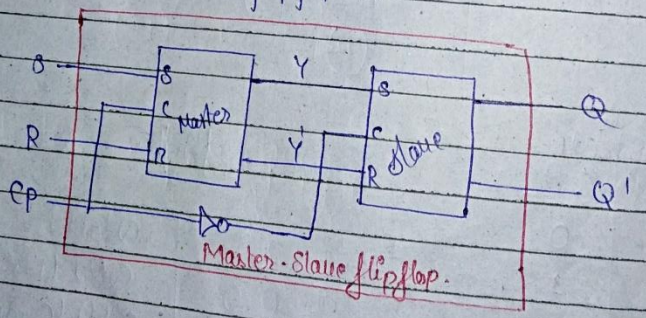


Positive pulse Negative pulse.

fig: Def^n of clock pulse transition.

Master Slave flipflop.

→ RS master-slave flipflop.



PAGE NO. :
DATE :

Draw master slave JK flipflop.

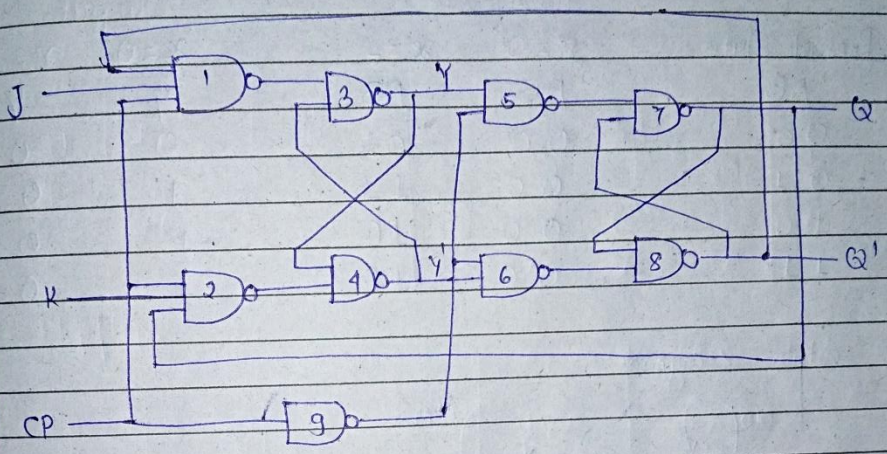


fig: Clocked master slave J-K flipflop

State Diagram.

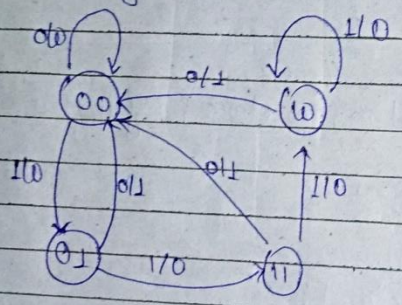
State Table for example circuit:

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

This table can alternatively represented as,

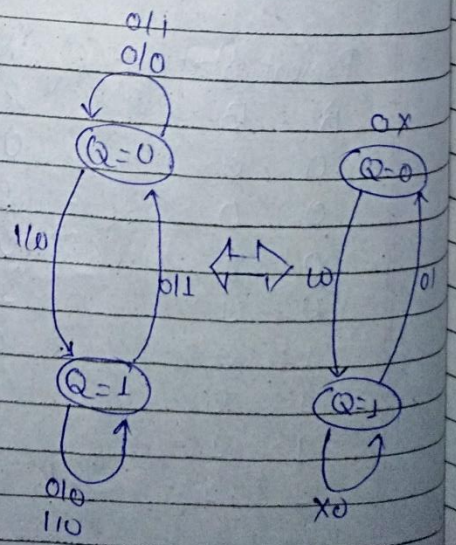
Present state	Next state $x=0$	Next state $x=1$	Output $x=0$	Output $x=1$
AB	AB	AB	Y	Y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

State Diagram



State diagram for S-R flip flop:

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

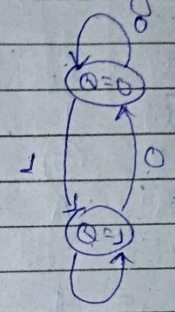


PAGE NO. :
DATE :

State diagram for D Flipflop

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

State diagram

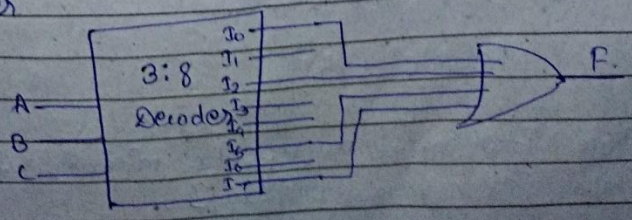


Q. Implement $F(A,B,C) = \Sigma(0,2,5,7)$ using (a) Decoder, (b) Multiplexer (c) PLA.

→ Soln,

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

(a) Decoder



(b) Multiplexer

Implementation Table:

	I_0	I_1	I_2	I_3
A'	(0)	1	(2)	3
A	4	(5)	6	(7)
	A'	A	A'	A

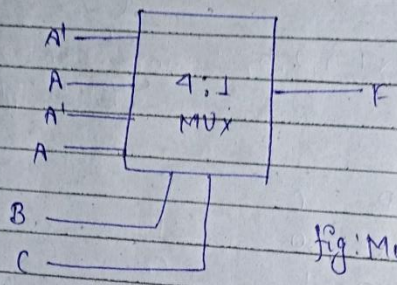
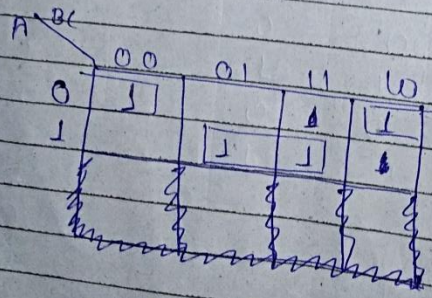


fig: Multiplexer Implementation.

(c) PLA



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

$F = A'C' + AC$

AND Gate : Total no. of non repeated minterms = 2

OR Gate : Total no. of output = 1.

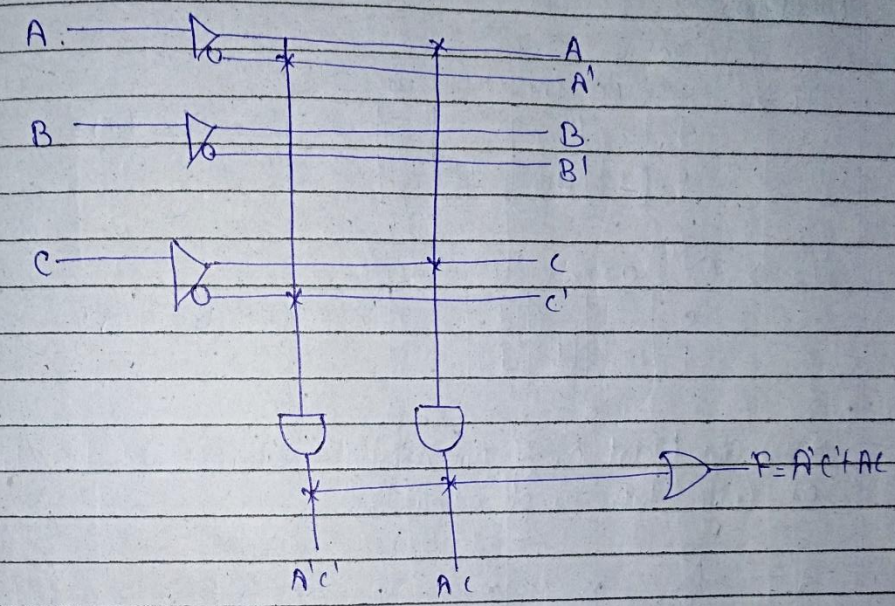


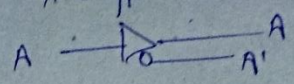
fig: PLA Implementation.

Example for implementation of PLA

Sample Table;

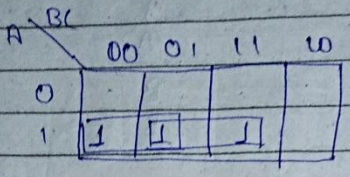
A	B	C	Y_1	Y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Buffer



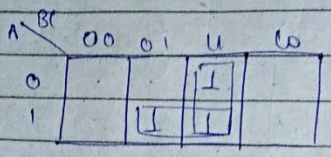
Equations,

$Y_1 \Rightarrow$



$Y_1 = AB' + AC$

$Y_2 \Rightarrow$



$Y_2 = BC + AC$

AND Gate (Total no. of non repeated minterms) = 3.
OR Gate (Total no. of output) = 2

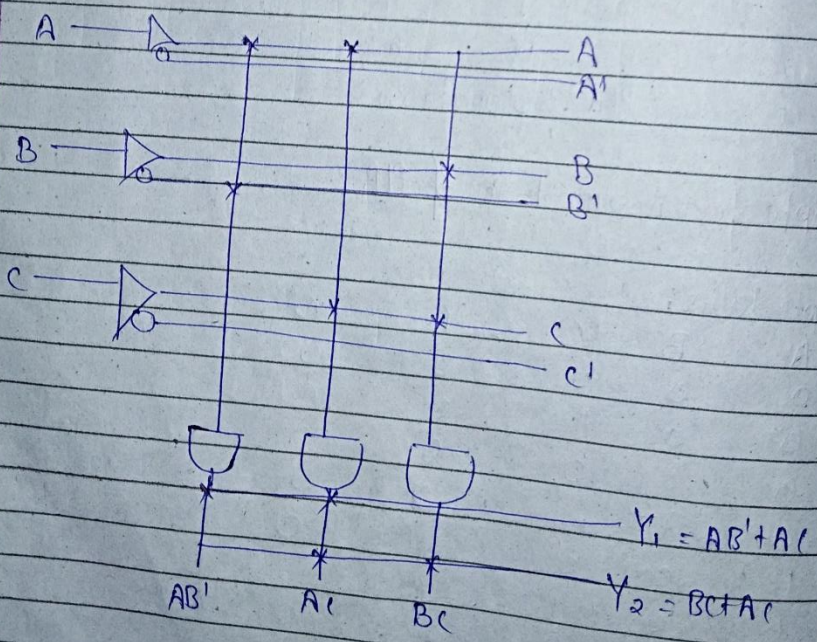


fig: PLA implementation

#

Unit - 7

Counters, Registers, Memory Unit.

Counter

A sequential ckt formed by cascading flip flops is called counters.

Uses: → counting pulses
→ frequency distribution methods.
→ distance measure in RADARs.

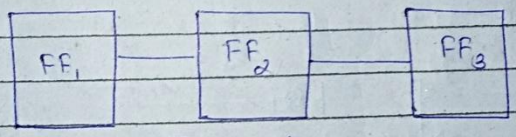


fig: Counter

If 3 flipflops are used,
$2^3 = 8$ combinations ; max count = $2^3 - 1$.
⇒ 0 to 7 → 7 is max count.

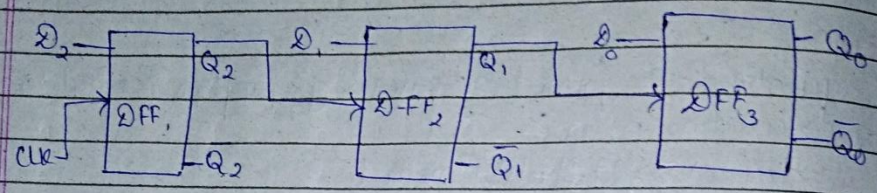
Types of counter.

1. On the basis of clock pulses provided;
   (a) Asynchronous counter
   (b) Synchronous counter

2. On the basis of counting sequence:
   (a) Up counter
   (b) Down counter.

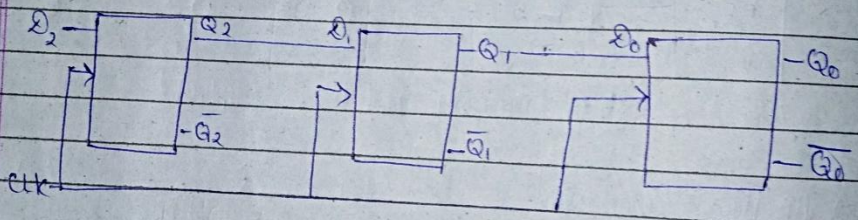
Asynchronous Counter

→ Different clock pulse provided to different flipflops.



Synchronous Counter

→ same clock pulse



Up Counter → counting sequence upwards
Eg: 0, 1, 2, ... 7

Down Counter → counter sequence downwards
Eg: 7, 6, 5, ... 0

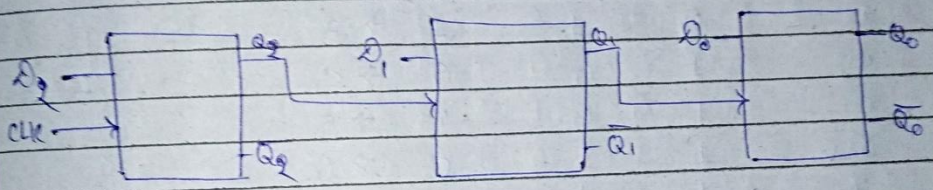
Mod:

→ Mod - 10 counter : 4 flipflops
→ Mod - 5 counter : 3 flipflops
→ Mod - 11 counter : 4 flipflops
→ Mod - 3 counter : 2 flipflops

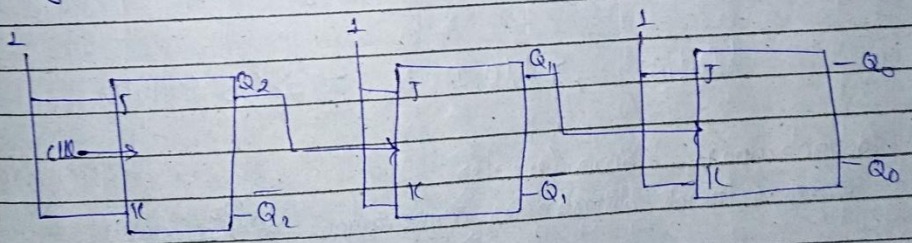
PAGE NO.:
DATE :

Asynchronous up counter

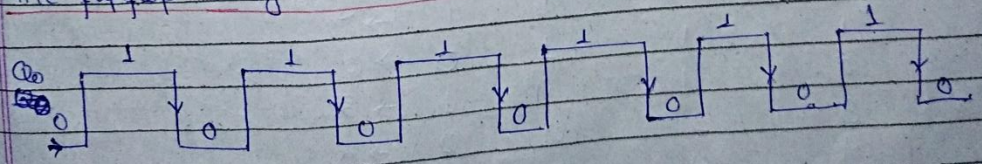
Using D flip-flop



Using J-K flipflop



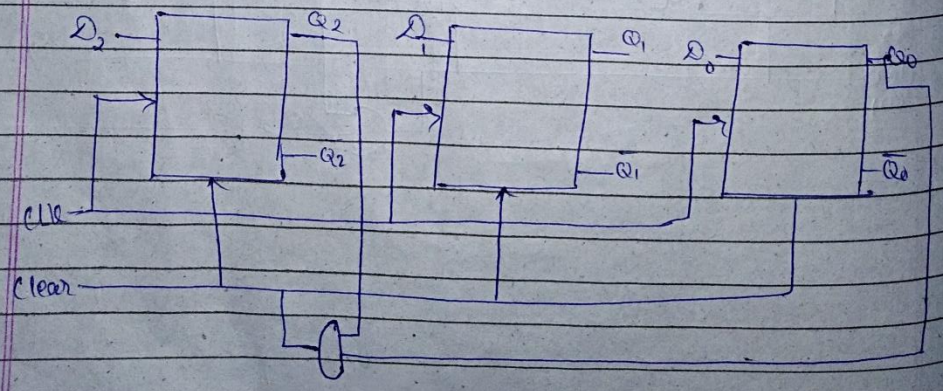
→ The flipflop changes it's state from 1 to 0.



P.T.O.

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	→ 0
0	0	0	1	→ 1
0	0	1	0	→ 2
0	0	1	1	→ 3
0	1	0	0	→ 4
0	1	0	1	→ 5
0	1	1	0	→ 6
0	1	1	1	→ 7
1	0	0	0	→ 8
1	0	0	1	→ 9
1	0	1	0	→ 10
1	0	1	1	→ 11
1	1	0	0	→ 12
1	1	0	1	→ 13
1	1	1	0	→ 14
1	1	1	1	→ 15

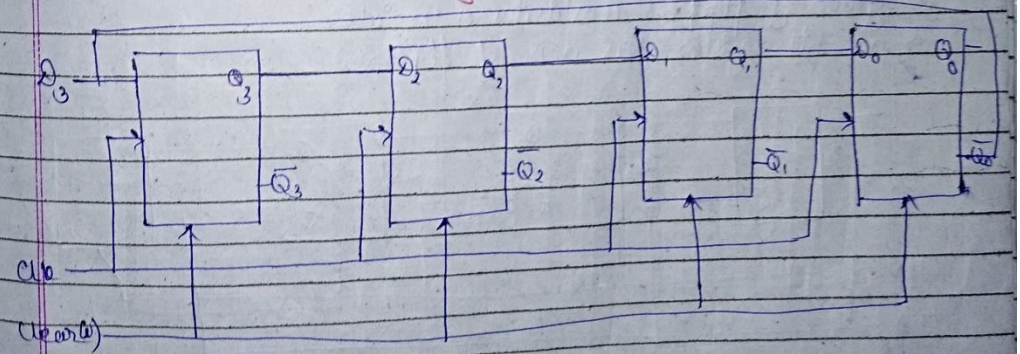
MOD-5 Counter Synchronous



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

Synchronous Counter

* Johnson Counter (Twisted Ring Counter)



For 4 flipflops Johnson counter;
⇒ 8 states

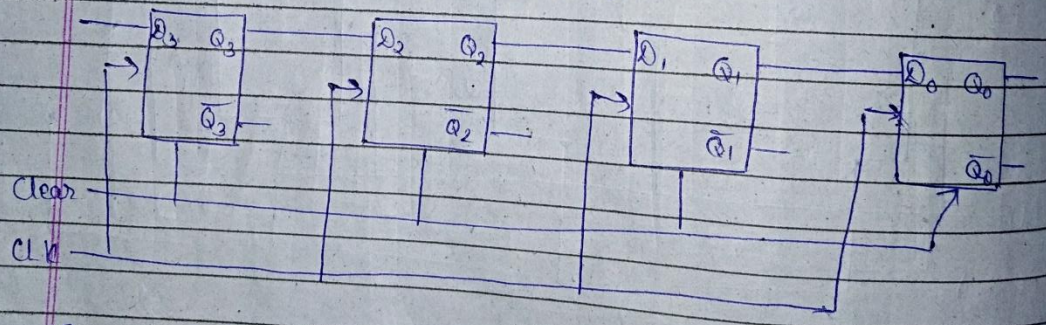
N.B. For n ff's → 2^n ~~count~~ states

P.T.O.

	Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0.	0	0	0	0	1	0	0	0
1	1	0	0	0	1	1	0	0
2	1	1	0	0	1	1	1	0
3	1	1	1	0	1	1	1	1
4	1	1	1	1	0	1	1	1
5.	0	1	1	1	0	0	1	1
6	0	0	1	1	0	0	0	1
7	0	0	0	1	0	0	0	0
8	0	0	0	0	1	0	0	0

Registers

→ group of flipflops constitutes a register
→ n bit register has n no. of flipflops.



Example: Registers

1 0 1 0 → initial

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

PAGE NO.:
DATE:

→ For n-bit register, n no. of clock cycles are needed to load.

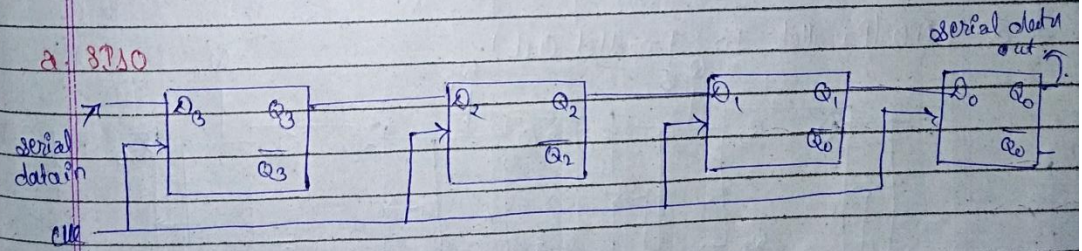
Shift register

A register capable of shifting binary information either to left or right is called shift register.

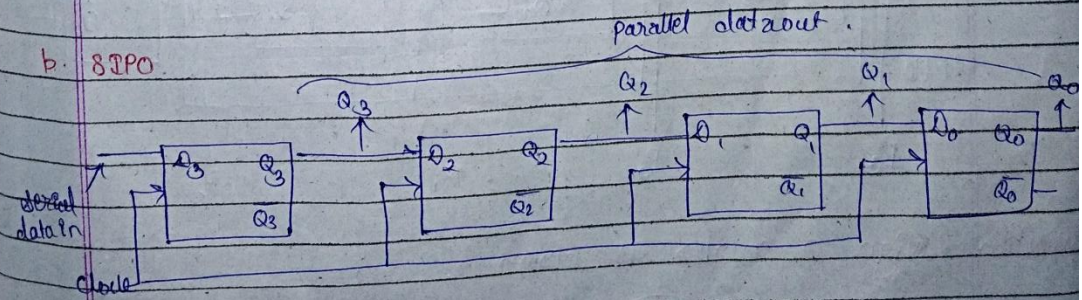
Types of shift register

(A) SISO (Serial in Serial Out)
(B) SIPO (Serial in Parallel out)
(C) PISO
(D) PIPO

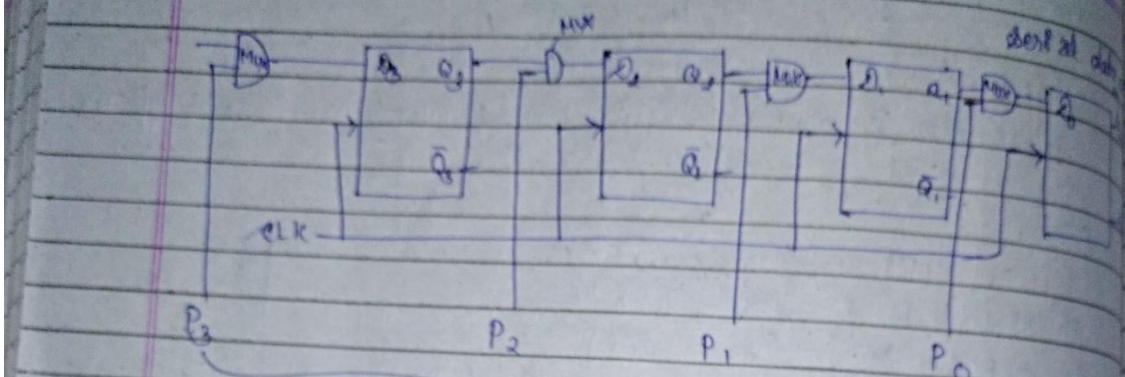
a. SISO



b. SIPO

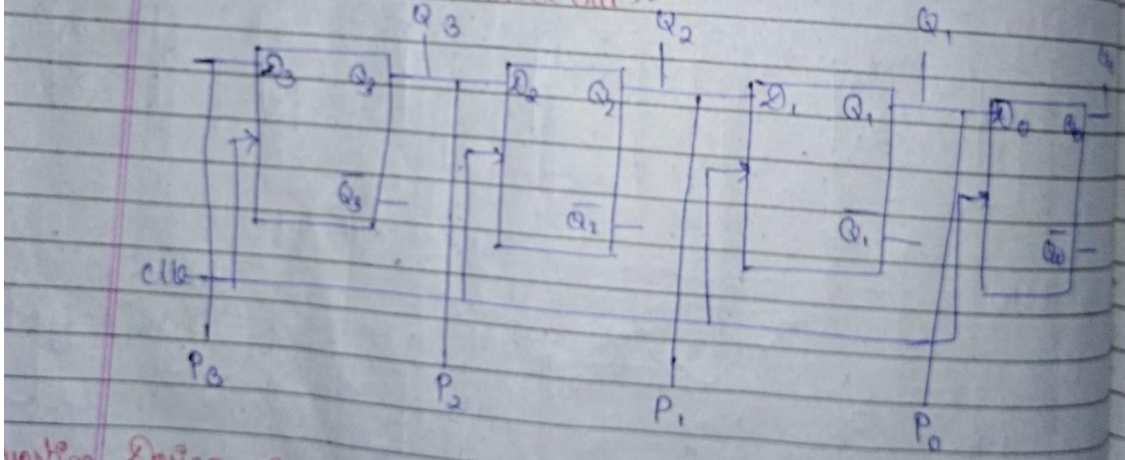


c. PISO (Parallel In Serial Out)

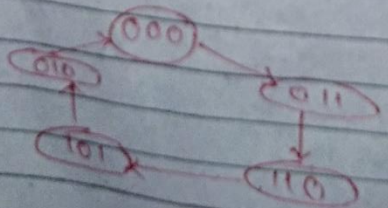


parallel data in

d. PIPO (Parallel In Parallel Out).



Question Design a counter as shown



→ 1, 4, 7 missing.

PAGE NO. :
DATE :

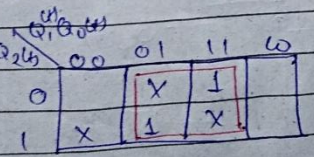
A. T flip flop

past o/p $Q(t)$	present o/p $Q(t+1)$	T → input
0	0	0
0	1	1
1	0	1
1	1	0

Minterm	Past state $Q_2(t)$	$Q_1(t)$	$Q_0(t)$	Present state $Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	Input T_A	T_B	T_C
0	0	0	0	0	1	1	0	1	1
1	0	0	1				x	x	x
2	0	1	0	0	0	0	0	1	0
3	0	1	1	1	1	0	1	0	1
4	1	0	0				x	x	x
5	1	0	1	0	1	0	1	1	1
6	1	1	0	1	0	1	0	1	1
7	1	1	1				x	x	x

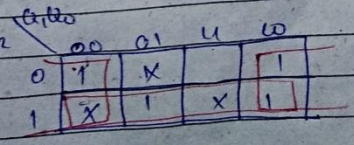
Characteristic Equations

For T_A,

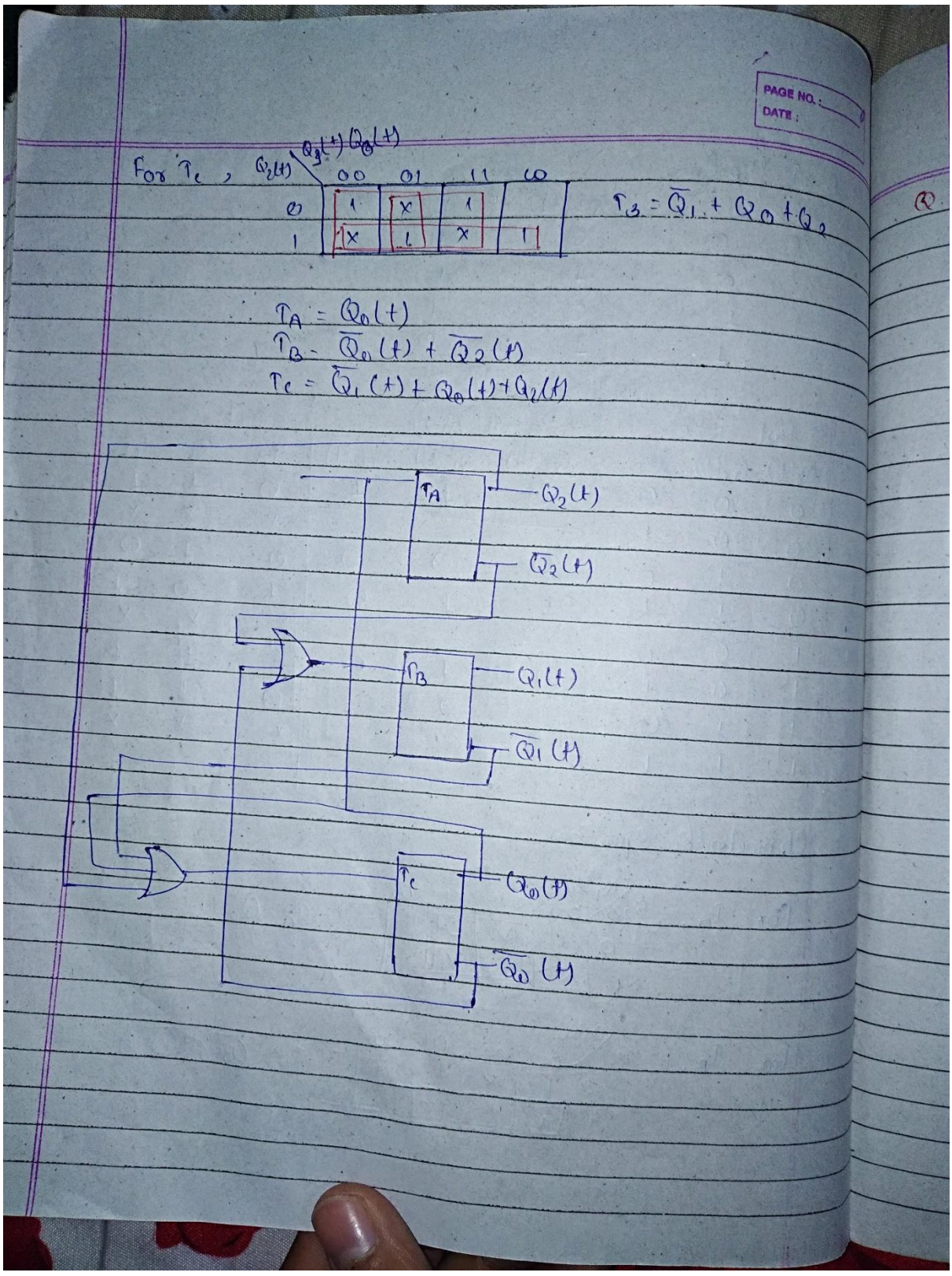


$T_A = Q_0(t)$

For T_B,



$T_B = \overline{Q_0(t)} + \overline{Q_2(t)}$

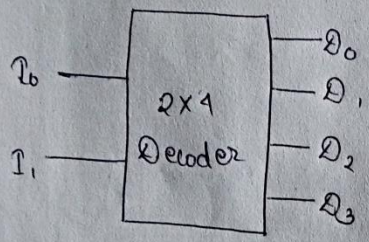


Assignment #3

1. Design the decoder using universal gates.

→ A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit decoded information has unused and don't-care combinations, the decoder output will have fewer than 2^n outputs.

Consider a 2:4 line decoder circuit in fig(1). The two inputs are decoded into four outputs, each output representing one of the minterms of the two input variables.



Block diagram

I_1	I_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Truth table

From the truth table, the simplified Boolean expressions are;

$D_0 = I_1'I_0'$; $D_2 = I_1I_0'$

$D_1 = I_1'I_0$; $D_3 = I_1I_0$

Designing of 2 to 4 line decoder,

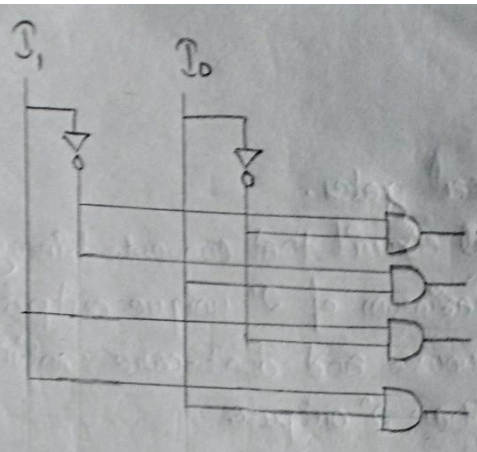


fig: logic diagram of 2:4 Decoder

Now, using universal gates only,

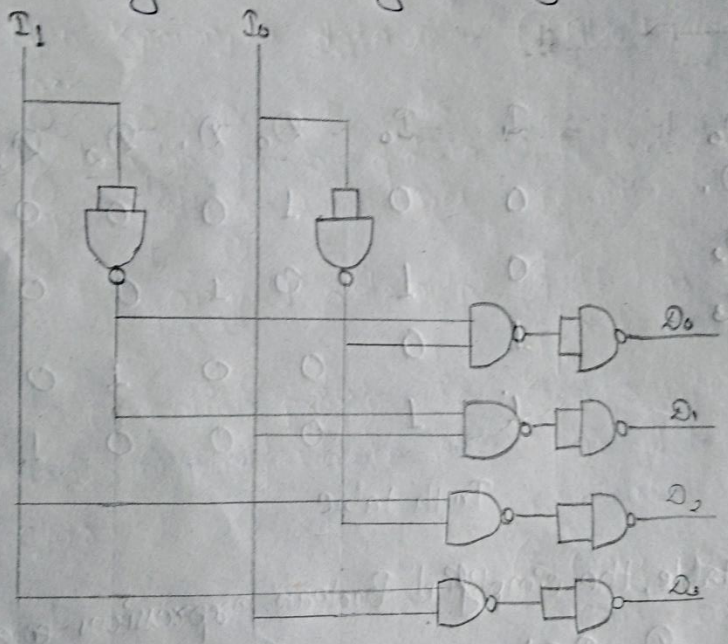


fig: Logic diagram of 2:4 decoder using NAND gate only.

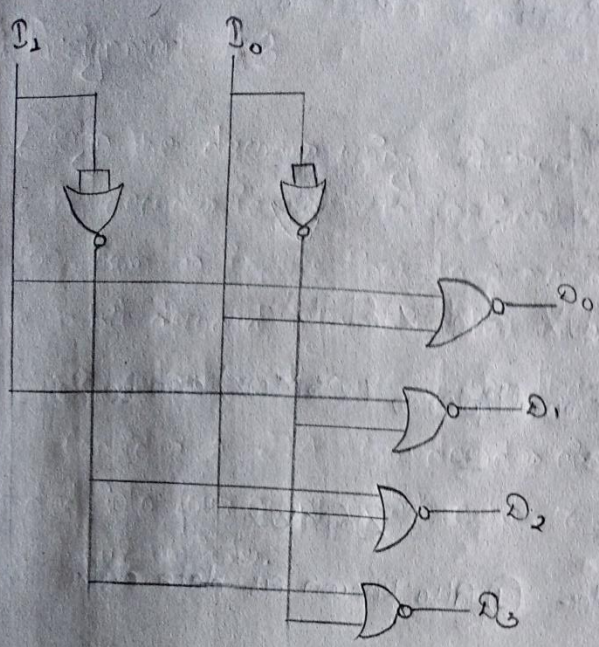


fig: logic circuit for 2:4 line decoder using NOR gates only

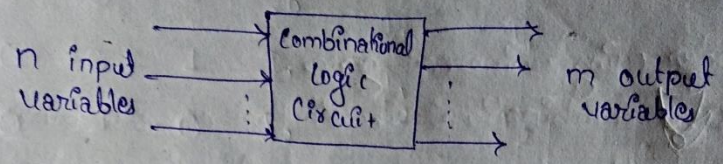
2) What is combinational logic? What are important features?

→ A combinational logic is a type of digital logic which is implemented by Boolean circuits, where the output is a pure function of the present input only. This digital logic doesn't consist of memory.
In this digital logic, we combine the different gates in the circuit, for example, encoder, decoder, multiplexer and demultiplexer.

The important features of combinational logic are as follows;

1. The output of combinational circuit at any instant of time, depends only on the levels present at input terminals
2. The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
3. A combinational circuit can have an n number of inputs and m numbers of outputs.
4. An external register does not influence the behaviour of the combinational logic.

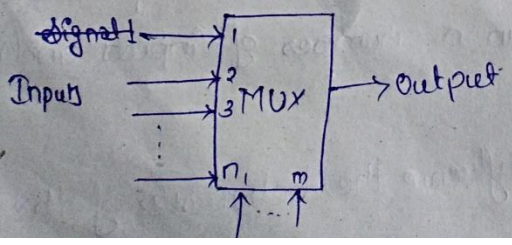
A block diagram for combinational circuit is;

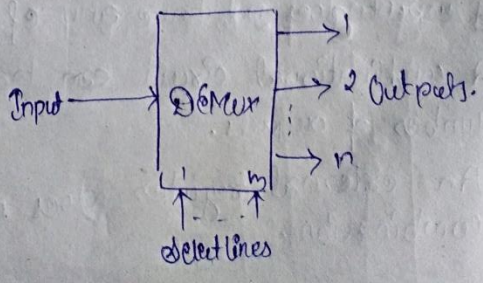


3. Differentiate between MUX and DeMUX.

→ Multiplexer (MUX) and Demultiplexer (DeMUX) can be differentiated as follows:

MUX	DeMUX
① Also known as data selector.	① Also known as data distributor
② It has 'n' number of data inputs with 'm' select lines	② It has only 'one' data input with 'm' select lines
③ It has only '1' data output.	③ It has 'n' number of data output
④ It's operation principle is many to one as data selector.	④ It's operation principle is 1 to many as data distributor.
⑤ It can be used as universal logic ckt as one can implement any combinational ckt.	⑤ Only some combinational ckt can be implemented.





4. Explain the operation of Decoder.

→ Decoder is a combinational circuit that converts binary information from n inputs lines to a maximum of 2^n unique output lines.

→ If the n-bit decoded information has unused or don't care combinations, the decoder output will have fewer than 2^n outputs.

→ n-to-m-line decoders have $m <= 2^n$.

Operation of Decoder.

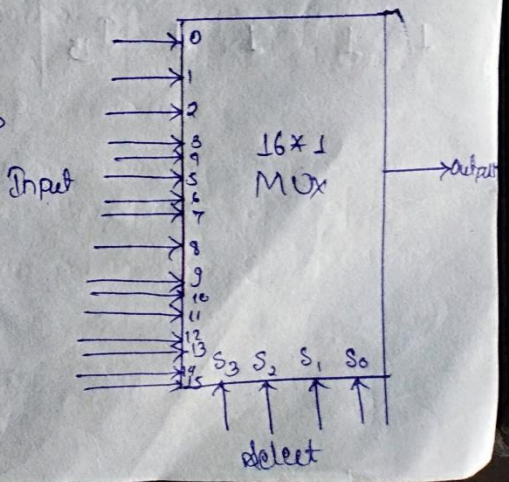
A decoder provides 2^n minterms of n input ~~lines~~ variables. Decoders are digital ICs which are used for decoding. In other words the decoders decrypt or obtain the actual data from the received code, i.e. convert the binary input as it's input to a form, which is reflected as its output. The 2^n minterms obtained from the decoder are unique and are ~~for~~ later expressed in sum of minterms form using OR gate as per the requirement of the information. This SOP form can be easily obtained from the truth table or by expanding the functions to their sum of minterms.

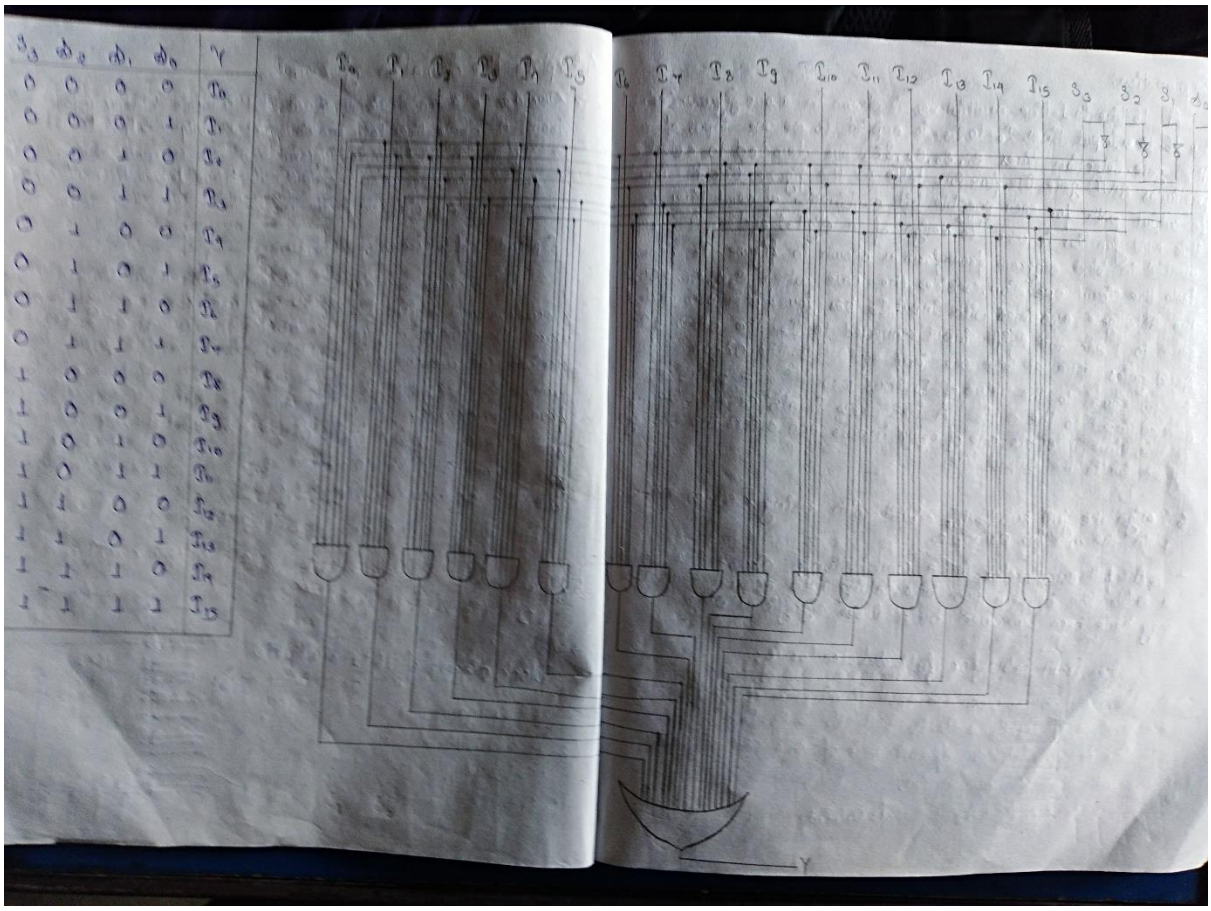
[n inputs and m outputs ⇒ n-to 2^n line decoder & m OR gates.

5) Draw a block diagram, truth table and logic circuit of a 16*1 multiplexer and explain it.

→ Soln,

Block diagram of 16*1 multiplexer





Explanation:

A 16-line to 1-line multiplexer is shown in above figure. Each of the 16-~~lines~~, input lines, I_0 to I_{15}, is applied to ~~one~~ one input of an AND gate. Selection lines S_1, S_0, S_2, & S_3 are decoded to select a particular AND gate. The function table in the figure lists the input-to-output path for each possible bit combination of select lines. When this MSI function is used in the design of a digital system, it is represented in block diagram ~~from~~ form as shown in figure. To demonstrate the circuit operation, consider the case when $S_3S_2S_1S_0 = 0010$. The AND gate associated with input I_2 has four of inputs ~~connected~~ equal to 1 and the ~~*~~fifth input connected to I_2. The other 14 AND gates have at least one input equal to 0. The OR-gate output is now equal to the value of I_2, thus providing a path from the selected input to the output. A multiplexer is also called a data-selectors, since it selects one of many inputs and steers the binary information to the output line.

6. Design the full substractor circuit with using Decoder and explain the working principle.

→ A full substractor is a combinational circuit that performs a substraction between two bits, taking into account that a 1 may have been burrowed by a lower significant stage.

The truth table for full substractor is;

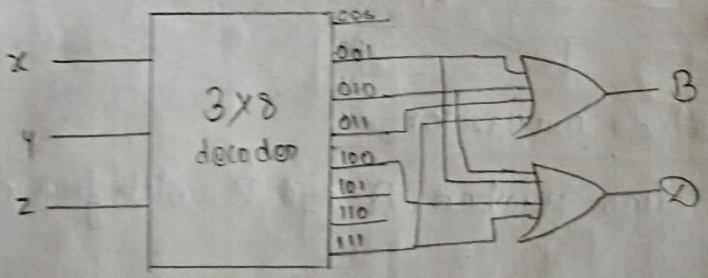
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

⇒ From truth table,

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

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

Combinational circuit Implementation using decoder



Working Principle of full substractor

Full substractor requires the 3 inputs, x, y & z, denote the minuend, subtrahend, and previous borrow respectively. The two outputs, D and B represent the difference and output borrow, respectively. The 1's and 0's for the output variables are determined from the substraction x-y-z.

7. What is magnitude comparator? Design a logic circuit of 4-bit mag nitude comparator and explain it.

→ A magnitude comparator is a combinational circuit that compares two numbers, A & B, and determines their relative mag nitudes. The outcome of the comparison is specified by three binary ~~numbers~~ variables that indicate whether A>B (G), A=B (E) or A<B (L).

4-bit magnitude comparator

Consider two numbers, A and B, with four digits each such that,

$$A = A_3 A_2 A_1 A_0$$

$$B = B_3 B_2 B_1 B_0$$

Here, each subscripted letter represents one of the digits in the number. The two numbers are equal if all pairs of significant digits are equal i.e, if $A_3 = B_3$ & $A_2 = B_2$ & $A_1 = B_1$ & $A_0 = B_0$

The equality relation of each pair of bits can be expressed logically with a equivalence function:

$$x_i = A_i B_i + A_i' B_i' \qquad i = 0, 1, 2, 3.$$

where, $x_i = 1$ only if the pair of bits in position i are equal.

For the equality condition to exist, all x_i variables must be equal to 1. This indicates an AND operation of all variables:

(i) $(A=B) = x_3 x_2 x_1 x_0$

To determine if A is greater than or less than B, we inspect the relative magnitudes of pairs of significant digits starting from the most significant position. If the two digits are equal, we compare the next lower significant pair of digits. This comparison continues until a pair of unequal digits is reached. If the corresponding digit of A is 1 and that of B is 0, we conclude that $A > B$. If the corresponding digit of A is 0 and that of B is 1, we have that $A < B$. The sequential comparison can be expressed logically by using following Boolean functions:

(ii) $(A>B) = A_3 B_3' + x_3 A_2 B_2' + x_2 x_3 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$

(iii) $(A<B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$

the symbols $(A>B)$ & $(A<B)$ are binary output variables which are equal to 1 when $A>B$ or $A<B$, respectively.

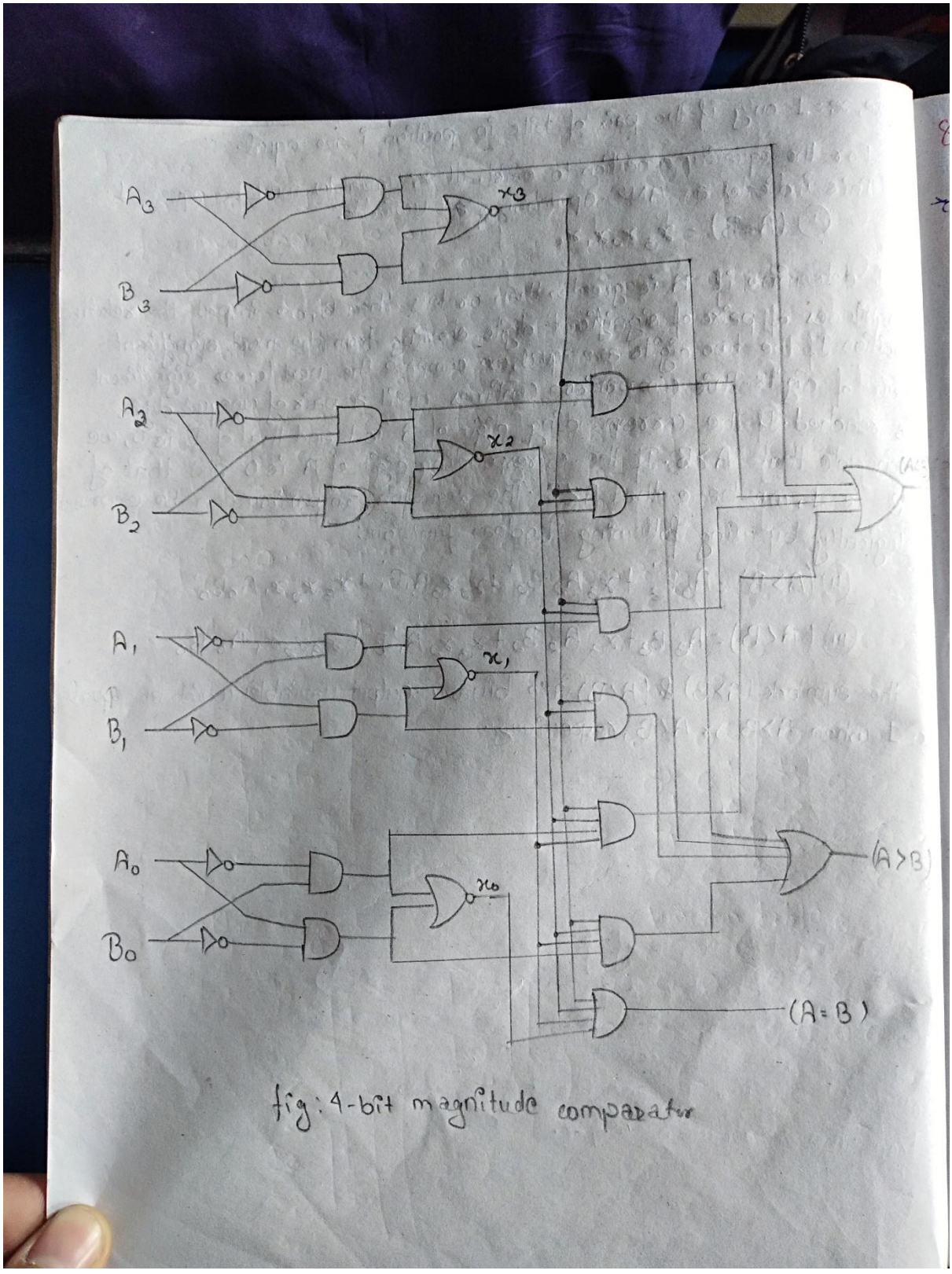


fig: 4-bit magnitude comparator

8. Design a 3to8 line decoder using two 2:4 line decoder and explain it.

→ Soln,

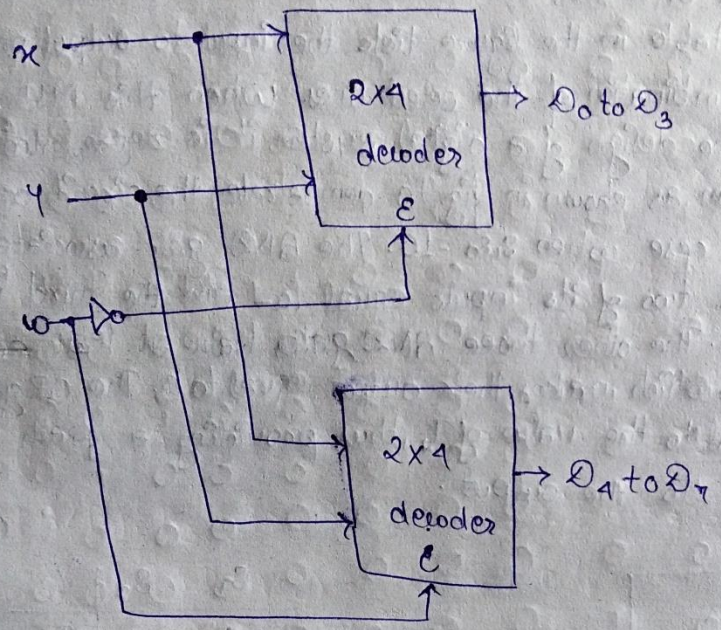


fig: A 3x8 line decoder using two 2x4 line decoder.

Decoder circuits can be connected together to form a larger decoder circuit. Above fig. shows two 2x4 decoders with enable inputs connected to form a 3x4 decoder. When w=0, the top decoder is enabled and the other is disabled. The bottom decoder outputs are all 0's, and the top four outputs generate minterm 000 to 011. When w=1, the enable conditions are reversed; the bottom decoder outputs generate minterms 100 to 111, while the outputs of the top decoder are all 0's. This is how the circuit works.

9. Draw a logic circuit of 4x1 multiplexer.

→ A 4-line to 1-line multiplexer is shown in fig below. Each of the four input lines, I_0 to I_3, is applied to one input of an AND gate. Selection lines S_1 and S_0 are decoded to select a particular AND gate. The function table in the figure lists the input-to-output path for each possible bit combination of the select lines. When this MUX function is used in the design of a digital system, it is represented in block diagram form as shown in fig. To demonstrate the circuit operation, consider the case when $S_1S_0 = 10$. The AND gate associated with input I_2 has two of its inputs equal to 1 and the third input connected to I_2. The other three AND gates have at least one input equal to 0, which makes their output equal to 0. The OR gate output is now equal to the value of I_2, thus providing a path from the selected input to the output.

Block diagram

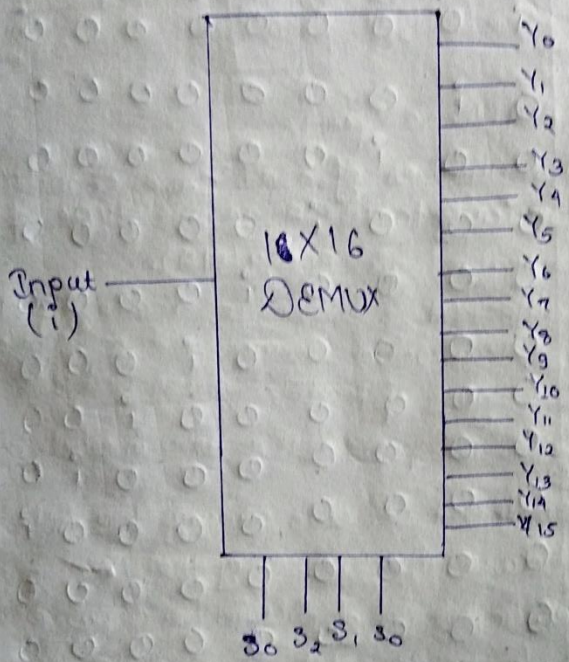
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

function table.

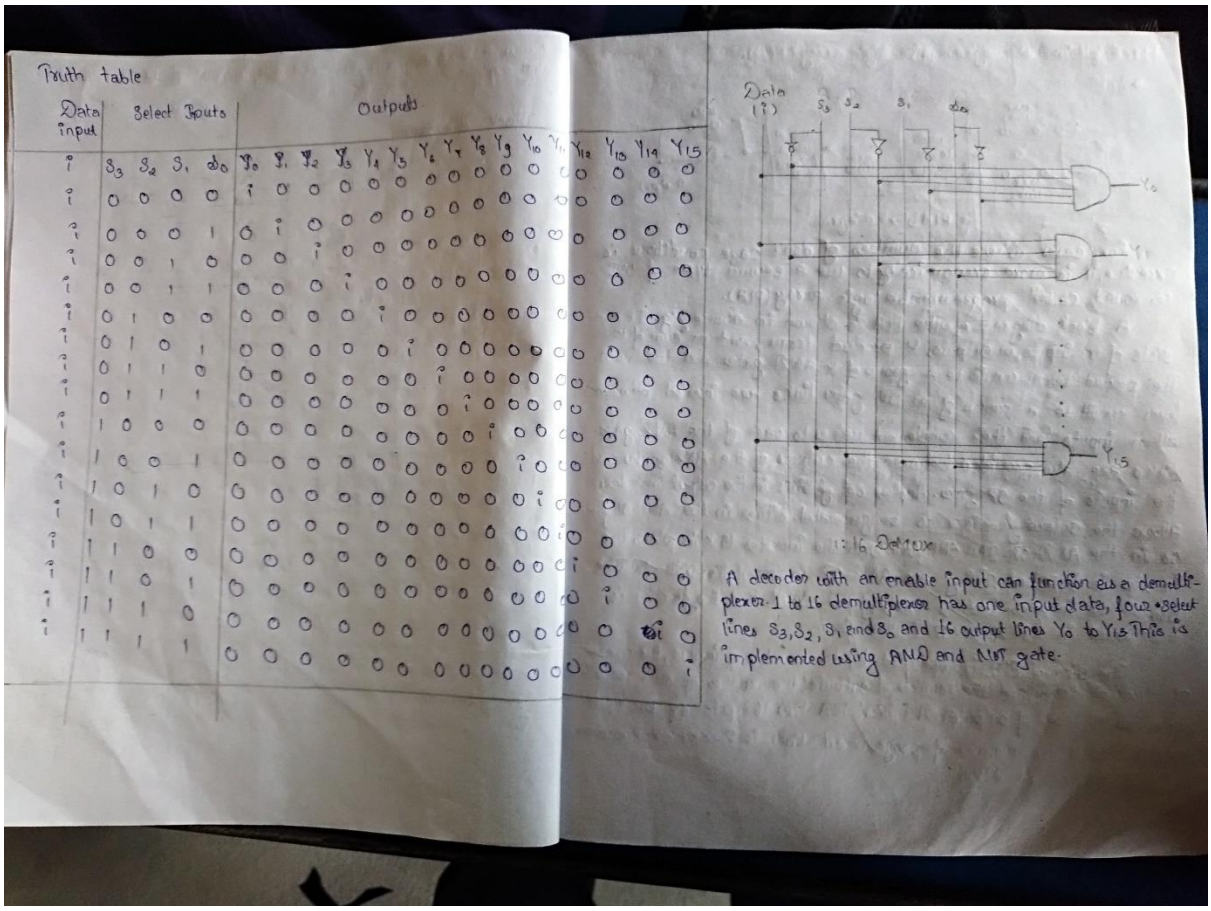
10. Draw a block diagram, truth table, and logic circuit of 1x16 demultiplexer and explain its working principle.

→

Block diagram.



P.T.O.



Truth table

Data input	Select Inputs				Outputs															
	S_3	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1:16 Demux

A decoder with an enable input can function as a demultiplexer. 1 to 16 demultiplexer has one input data, four select lines S_3, S_2, S_1 and S_0 and 16 output lines Y_0 to Y_{15}. This is implemented using AND and NOT gate.

11. Explain the programmable logic array (PLA).

→

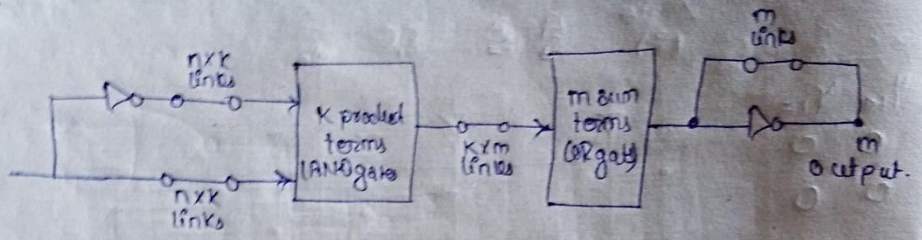


fig: PLA block diagram.

For cases where the number of don't care conditions is excessive, it is more economical to use a second type of LSI component called programmable logic array (PLA).

A block diagram of the PLA is shown in above fig. It consists of n inputs, m outputs, k product terms, and m sum of terms. The product terms constitute a group of k AND gates and the sum terms constitute a group of m OR gates. links are inserted between all n inputs and their complement values to each of the AND gates. Links are also provided between the outputs of the AND gates and the inputs of the OR gates. Another set of links in the output inverters allows the output functions to be generated either in the AND-OR form or in the AND-OR-INVERT form. With the inverter link in place, the inverter is bypassed, giving an AND-OR implementation. With the link broken, the inverter becomes part of the circuit and the function is implemented in the AND-OR-INVERT form.

The size of PLA is specified by the number of inputs, the number of product terms, and the number of outputs (the number of sum terms is equal to number of outputs). A typical PLA has 16 inputs, 48 product terms and 8 outputs. The number of programmed links is $2n \times k + k \times m + m$, whereas that of a ROM is $2^n \times m$.

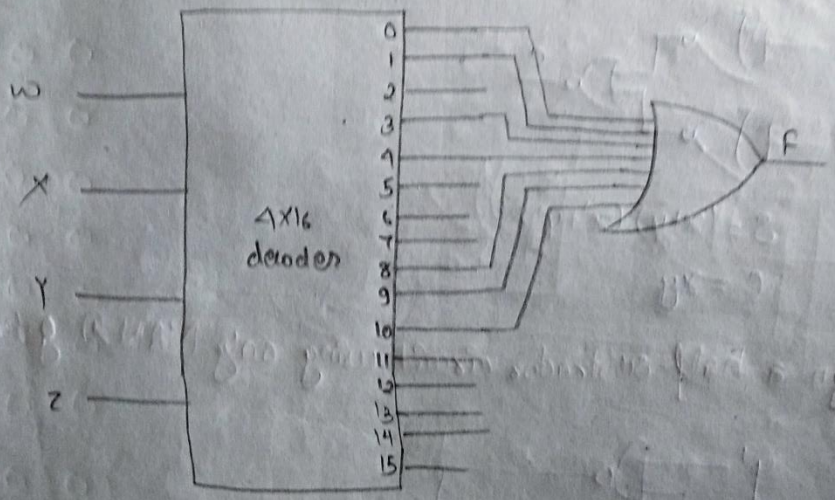
2. What is a decoder? Implement the following using decoder

a. $F(W, X, Y, Z) = \Sigma(0, 1, 3, 4, 8, 9, 10)$

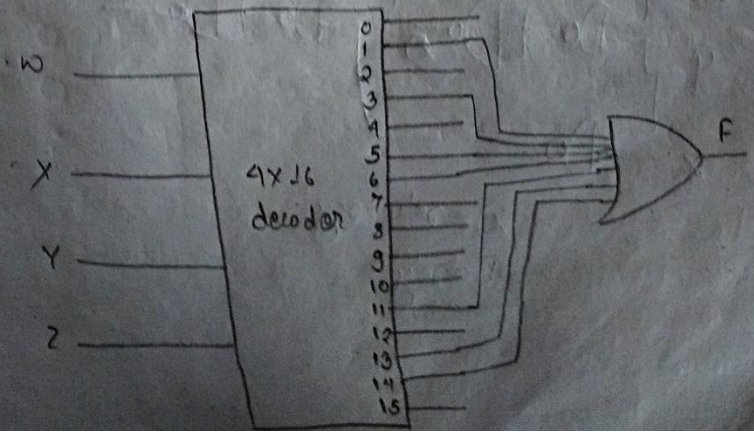
b. $F(W, X, Y, Z) = \Sigma(1, 3, 5, 6, 11, 13, 14)$

→ A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.

(a) $F(W, X, Y, Z) = \Sigma(0, 1, 3, 4, 8, 9, 10)$

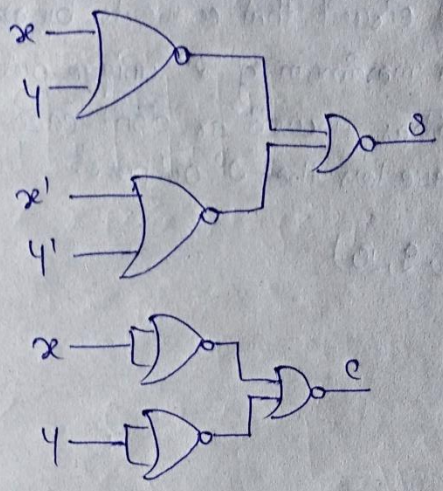


(b) $F(W, X, Y, Z) = \Sigma(1, 3, 5, 6, 11, 13, 14)$



13. Design a half adder logic using only NOR gate.

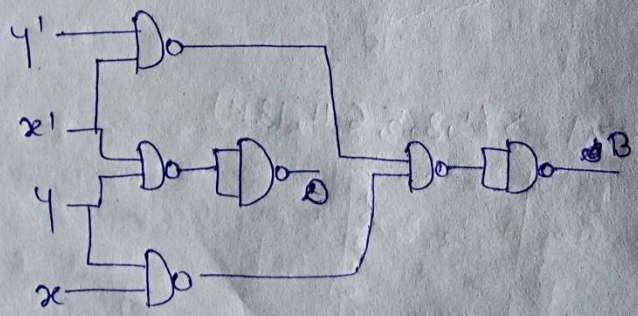
→ soln



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

$C = xy$

14. Design a half-subtractor circuit using only NAND gate.



$D = x'y + xy'$

$B = x'y$

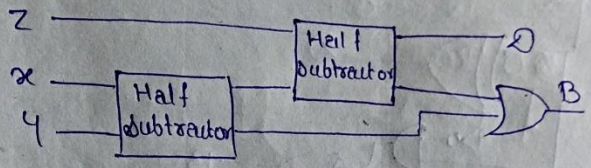
15. Design a full subtractor with truth table and logic gates.

→ Full subtractor

A full subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage.

This circuit has three inputs and two outputs. The three inputs, x, y and z, denote the minuend, subtrahend, and previous borrow respectively. The two outputs, D and B, represent the difference and output borrow respectively.

Block diagram

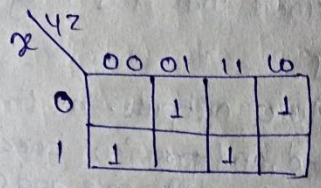


Truth Table

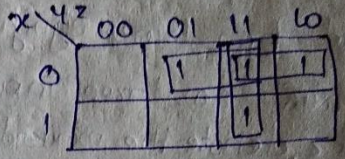
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Boolean functions:

The simplified Boolean functions for the two outputs of the full subtractor are derived in the following k-maps:

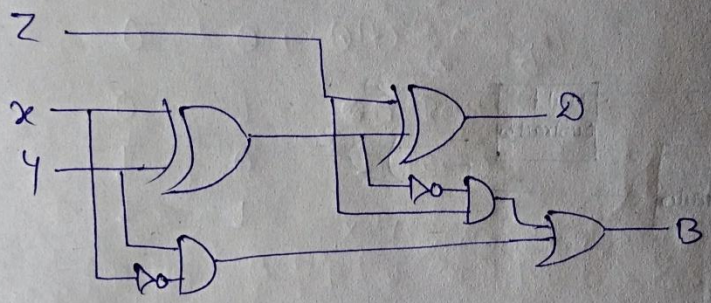


$D = x'y'z + x'yz' + xy'z' + xyz$



$B = x'y + x'z + yz$

Circuit diagram:



17. What do you mean by full adder and full subtractor?

→ Full adder:

A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs.

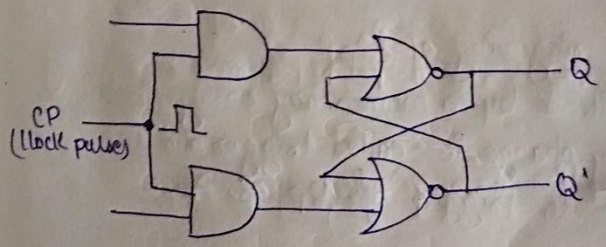
Full Subtractor:

A full subtractor is a combinational ckt that performs a subtraction between two bits, taking into account, that a 1 may have been borrowed by a lower significant stage. It This circuit has three inputs & two outputs.

Assignment #1

1. Describe the clocked RS flipflop.

→ The clocked RS flipflop shown in fig (a) consists of a basic NOR flip-flop and two AND gates. The output of the two AND gates remain at 0 as long as the clock pulse (CP) is 0, regardless of S and R input values. When the clock pulse goes to 1, information from the S and R inputs is allowed to reach the basic flipflop. The set state is reached with S=1, R=0 and CP=1. To change to the clear state, the inputs must be S=0, R=1 and CP=1. With both S=1 and R=1, the occurance of a clock pulse causes both outputs to momentarily go to 0. When the pulse is removed, the state of the flipflop is indeterminate. i.e. either state may result, depending on whether the set or the reset input of the basic flipflop remains a 1 longer before the transition of to 0 at the end of the pulse.



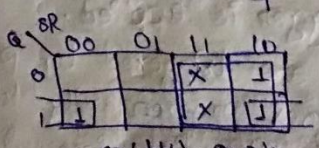
(a) logic diagram

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

(c) Characteristic table

(b) Graphical

d) Characteristic eqn



$Q(t+1) = S + R'Q$

$SR = 0$

In fig(c), Q is the binary state of the flipflop at a given time (referred as present state), the S and R columns give the possible values of the inputs, and Q(t+1) is the state of the flip flop after the occurrance of a clock pulse (referred to next state).

In fig(d), the two indeterminate states are marked by x's in the map, since they may result in either a 1 or a 0. However, the relation SR=0 must be included as part of the characteristic eqn to specify that both S and R cannot equal to 1 simultaneously.

The characteristic table can also be written as;

S	R	Q(t+1)
0	0	Q(t) (No change)
0	1	0 (Reset)
1	0	1 (set)
1	1	x (Indeterminate)

2. What do you mean by triggering of flipflop?

→ The state of flipflop is switched by a momentary change in the input signal. This momentary change is called trigger and the transition it causes is said to trigger the flipflop. ~~Asyn~~

Asynchronous flip flops require an input trigger defined by a change of signal level. This level must be returned to it's initial value (0 in the NOR and 1 in the NAND flipflop) before a second trigger is applied.

Clocked flipflops are ~~triggered~~ by pulses. A pulse starts from an initial value of 0, goes momentarily to 1, and after a short time returns to initial 0 value. The time interval from the application of the pulse until the output transition occurs is a critical factor that needs further investigation.

3. What is JK master slave flipflop? Design it's logic circuit, truth table and explain the working principle.

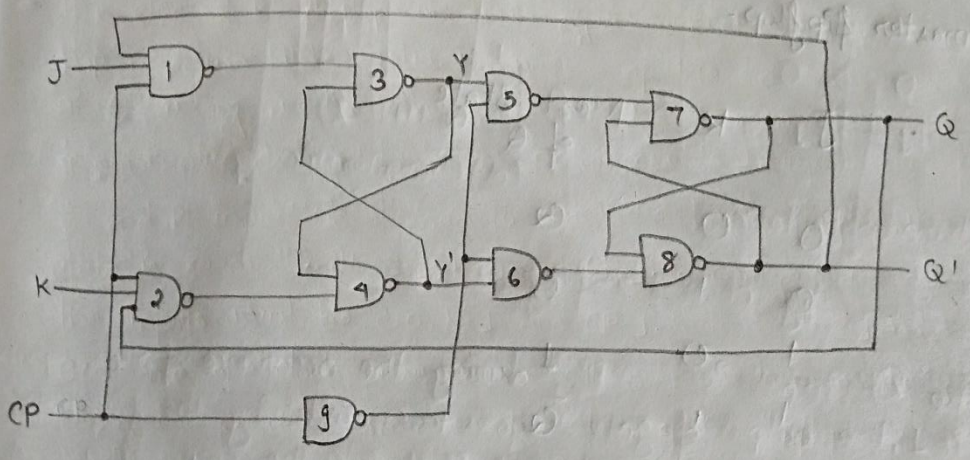


fig: Clocked master-slave JK flipflop.

A master-slave JK flip-flop constructed with NAND gates is shown in the above figure. It consists of two flipflops; gates 1 through 4 from the master flipflop, gates 5 through 8 form the slave flipflop. The information present at the J and K inputs is transmitted to the master flipflop on the positive edge of a clock pulse & is held there until the negative edge of the clock pulse occurs, after which it is allowed to pass through the slave flip-flop. The clock input is normally 0, which keeps the outputs of gates 1 and 2 at the 1 level. This prevents the J and K inputs from affecting the master flip-flop. The slave flipflop is clocked RS type, with the master flipflop supplying the inputs and the clock input being inverted by gate 9. When the clock is 0, the output of the gate 9 is 1, so the output Q is equal to Y. and Q' is equal to Y'. When the true edge of a clock pulse occurs, the master flipflop is affected and may switch states. The slave flipflop is isolated as long as the clock is at that 1 level, because the output of gate 9 provides a 1 to

both inputs of the NAND basic flip flops of gate x and y. When the clock input returns to 0, the master flip-flop is isolated from the J and K inputs and the slave flipflops goes to the same states the master flipflop.

J	K	Next state of Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

4) What is flip-flop? Mention the application of flip flop.

→ The memory elements used in clocked sequential circuits are called flip-flops. A flip flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it. These circuits are binary cells capable of storing one bit of information.

Some of the main applications of flipflops are;

a. Data Storage

A flipflop storage one bit at a time in a digital slot. In order to store more than one bit flop can be ~~co~~ connected in series and parallel called registers.

b. Data Transfer

Flipflops can also be used extensively to transfer the data. For this purpose shift register is used. A shift register is a register which is able to shift or transfer ~~data~~ its content within itself without changing the order of its bits

c. Counter.

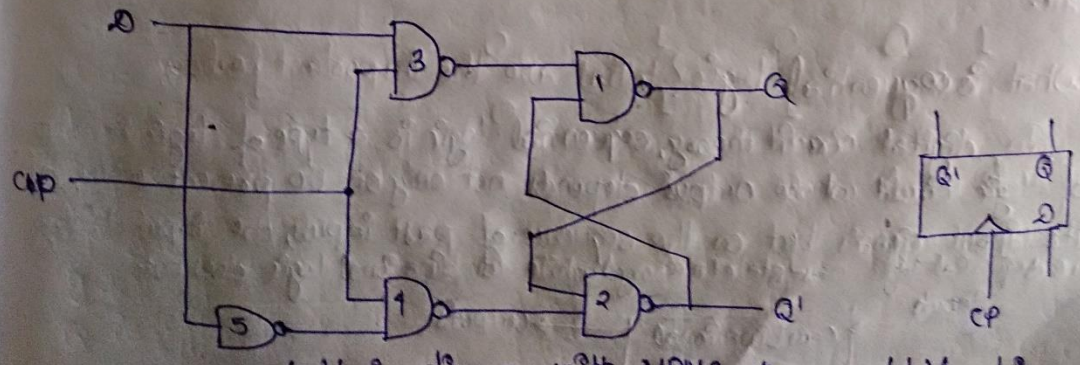
Another major application of flipflops is a digital counter. It is used to count pulses or events and it can be made by connecting a series of flipflops. Counter can count 2^n pulses, where n is number of flipflops.

d. Frequency Division.

Flipflops can divide the frequency of periodic waveform. When pulse wave is used to toggle an flipflop, the output frequency becomes one half the input frequency.

5) What do you mean by D-flipflop?

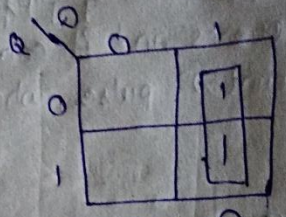
→ D flipflop shown in fig below is a modification of the clocked RS flipflop. NAND gates 1 and 2 form a basic flipflop and gates 3 and 4 modify it into a clocked RS flipflop. The D input goes directly to the S input, and it's complement, through gate 5, is applied to the R input. As long as the clock pulse is at 0, gates 3 and 4 have a 1 in their outputs, regardless the value of the other inputs. This conforms the requirement that the two inputs of a basic NAND flipflop remain initially at 1 level.



(a) Logic diagram with NAND gate (b) Graphic symbol

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(c) Characteristic table



Q(t+1) = D

(d) Characteristic eqn.

fig: Clocked D-flipflop.

The D input is sampled during the occurance of a clock pulse, if it is 1, the output of gate 3 goes to 0, switching the flipflop to set state (unless it was already set). If it is 0, the output of gate 4 goes to 0, switching the flip-flop to the clear state.

The D-flipflop receives the designation from it's ability to transfer "data" into a flipflop. It is basically an RS-flipflop with an inverter in the R-input. The added inverter reduces the number of inputs from two to one. The characteristic eqn shows that the next state of flip flop is the same as the D input and is independent of the value of the present state.

6) What is sequential logic? What are the important features?

→ In digital circuit theory, sequential logic is a type of logic is a type of logic circuit whose output depends not only on the present value of it's input signals but on the sequence of past inputs, the input history i.e. sequential logic has state(memory) while combinational logic does not.

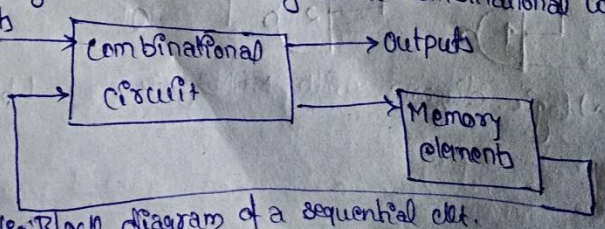


fig: Block diagram of a sequential ckt.

The important features of sequential logic are as follows.

1. Output is a function of clock, present inputs and the previous state of the system.
2. Have memory to store the present states that is sent as control input (enable) for the next operation.
3. It ~~enables~~ involves feedback from output to input that is stored in the memory for the next operation.
4. Uses clock pulses (triggered for operation with ~~clock~~ electronic pulses).
5. Used for storing data and hence used in RAMs.
6. Flip flops (binary storage device) are the elementary building unit.

7) Explain the Master slave SR flf with logic diagram, truth table and timing diagram.

→

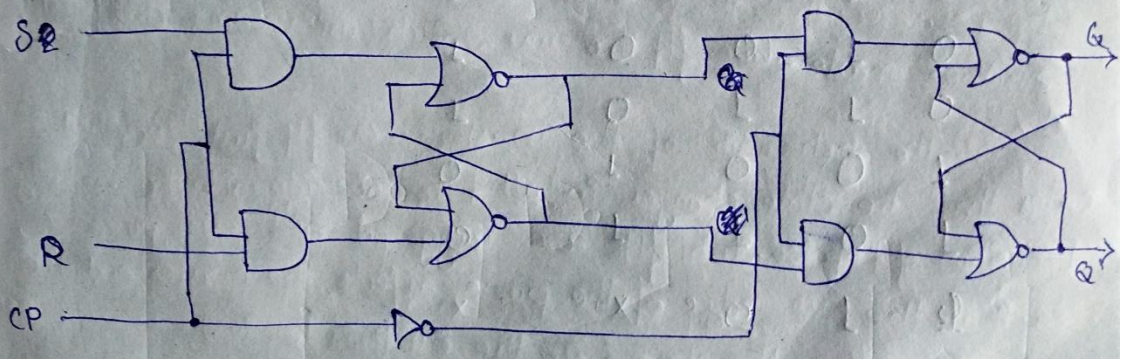


fig (a): Master Slave SR flf.

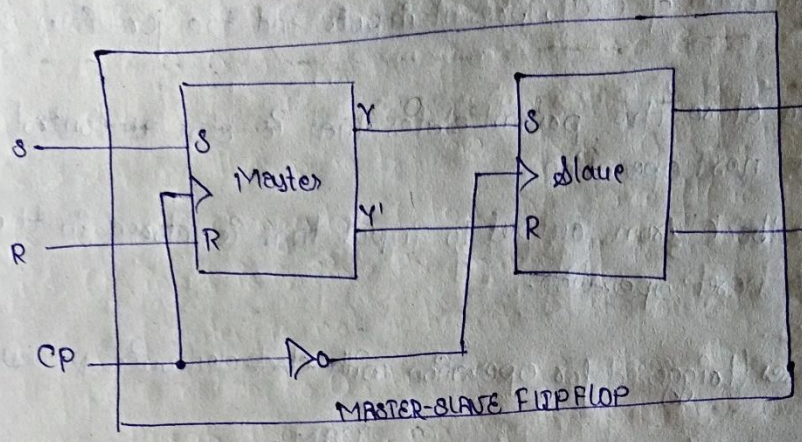


fig : Master-slave flipflop.

Truth table

S	R	Y	Q_{out}	Q_{out}'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

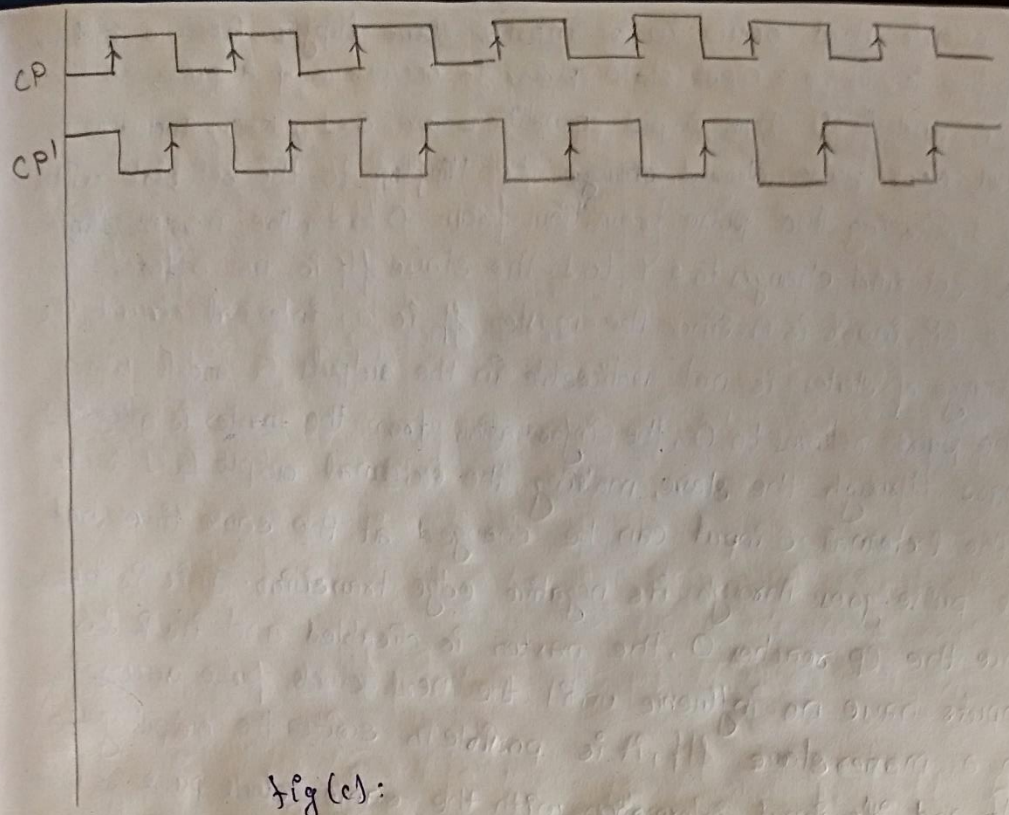


fig (c):

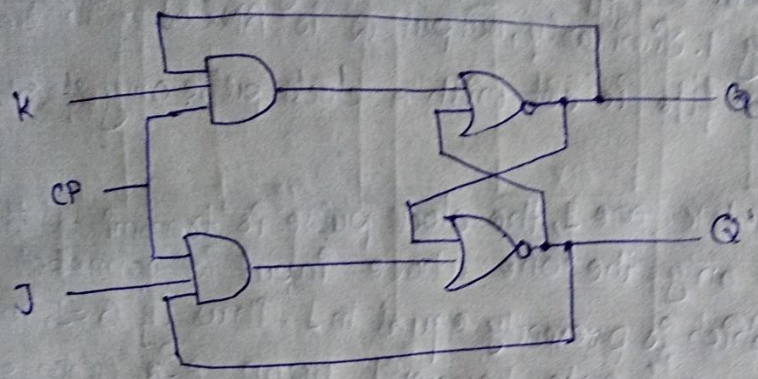
The logic diagram of an RS master-slave flipflop is shown in fig(a). It consists of a master flip-flop, a slave flipflop, and an inverter. When clock pulse CP is 0, the output of the inverter is 1. Since the clock input of the slave is 1, the flipflop is enabled and output Q is equal to Y, while Q' is equal to Y'. The master flipflop is disable because CP=0. When the pulse becomes 1, the information then at the external R and S inputs is transmitted to master flipflop. The slave flipflop, however, is isolated as long as the pulse is at it's 1 level, because the output of the inverter is 0. When the pulse returns to 0, the master flip-flop is isolated, which prevents the external inputs from affecting it. The slave flipflop then goes to the same state as the master flipflop.

The timing relationships shown in fig (c) illustrate the sequence

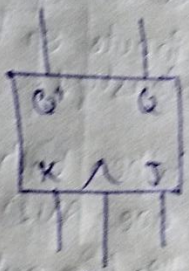
of events that occur in a master-slave flipflop. Assume that the flip flop is in the clear state prior to occurance of a pulse, so that Y=0 and Q=0. The input conditions are S=1, R=0, and the next clock pulse should change the flipflop to the set state with Q=1. During the pulse transition from 0 to 1, the master flipflop is set and changes to Y to 1. The slave flf is not affected because its CP input is 0. Since the master ff is an internal circuit, it's change of states is not noticeable in the outputs Q and Q'. When the pulse returns to 0, the information from the master is allowed to pass through the slave, making the external output Q=1. Note that the external S input can be changed at the same time that the pulse goes through its negative edge transition. This is because once the CP reaches 0, the master is disabled and it's R & S inputs have no influence until the next clock pulse occurs. Thus, in a master slave flf, it is possible to switch the output of the flf and it's input information with the same clock pulse. It must be realized that the S input could come from the output of another master-slave flf that was switch with same clock pulse.

The behaviour of the master-state flf just described indicates that the state changes in all flfs coincide with the -ve edge transition of the pulse. Such flfs are triggered with -ve pulses &, so that the -ve edge of the pulse affects the master and the +ve edge affects the slave & its the output terminals.

8) What is J-K ff? Explain.



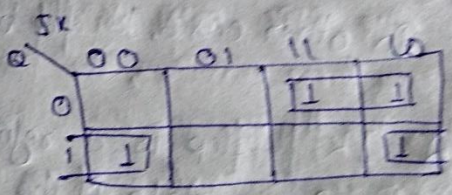
(a) Logic diagram



(b) Graphical Symbol

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(c) Characteristic table



$Q(j+1) = JQ' + K'Q$

(d) Characteristic eqn.

A JK flipflop is shown in fig(a). Output Q is ANDed with K and CP inputs so that the flf is cleared during the clock pulse only if Q was previously 1. Similarly, output Q' is ANDed with J and CP inputs so that the flfp is set with a clock pulse only if Q' was previously 1.

When both J and K are 1, the clock pulse is transmitted through one AND gate only - the one whose input is connected to the flipflop output which is presently equal to 1. Thus if Q=1, the output of the upper AND gate becomes 1 upon application of a clock pulse, and the flipflop is cleared. If Q'=1, the output of the lower AND gate becomes a 1 and the flipflop is set. In either case, the output is complemented.

In other cases, JK flf behaves as RS flf.

⇒ A JK flipflop is a refinement of the RS flf in ~~that~~ the ~~inter~~ indeterminate state of the RS type is defined in the JK type. (J→Set & ~~R→~~ K→Reset).

9. Differentiate between combinational logic and sequential logic. List some applications of sequential logic.

→

Combinational logic	Sequential logic
a. The circuit whose output at any instant depends only on the input present at ~~their~~ that instant only. is known as combinational logic.	b. The circuit whose output at any instant depends not only on the input present but also on the past output is known as sequential logic.
b. This type of logic has no memory unit	b. It has memory unit to store past output.

b. It does not require any feedback. It simply outputs the ~~log~~ input according to the logic desired.	c. It involves feedback from output to input i.e. that is stored in the memory for next operation.
d. Used mainly for Arithematic & Boolean operations.	d. Used for storing data. (and hence used in RAM).
e. Logic gates are the elementary building blocks	e. Flipflops are the elementary building blocks.
f. Independent of clock and hence does not require triggering to operate	f. Clocked (Triggered for operation with electronic pulses).
g. Examples of combinational logic are half adder, full adder, magnitude comparator, multiplexer, etc.	g. Examples of sequential logics are flipflops, registers, counters, etc.

The applications of sequential logic are as follows;

1. As a counter, flipflops, shift register.
2. As a memory unit.
3. As a programmable devices (PLDs, FPGA, CPLDs).

10. Draw a parallel-in-parallel out shift register and explain it.

→ The PIPO (parallel-in-parallel out) register also acts as a temporary storage devices or as a time delay device similar to SISO configuration. The data is presented in a parallel format to the parallel input pins P_A to P_D and then transferred together directly to their respective outputs pins Q_A to Q_D by the same clock pulse. The one clock pulse loads and uploads the register. This arrangement for parallel loading & uploading as shown below.

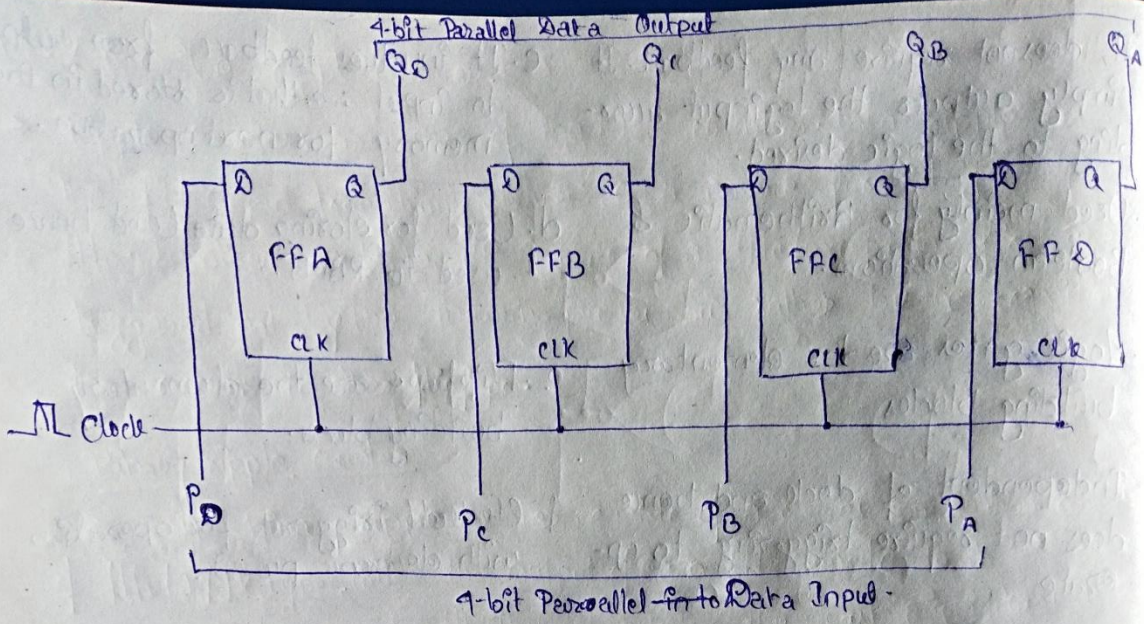


fig: 4-bit Parallel-in to Parallel-out shift register.

The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flf, the parallel output (PO) and the sequencing clock signal (clk). Similar to the SISO shift register, this type of register also acts as a temporary storage device ~~as~~ or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual ffs since no serial shifting of the data is required.

11. What are shift registers operations?

12. Describe ripple counter.

→ Ripple counters are one of the MSI counters. In a ripple counter, the flf output transitions serves as a source for a triggering othe flfs. In other words, the CP inputs of all flfs (except the first are triggered not by the incoming pulses but rather by the transition that occurs in other flipflops.

Binary ripple counter

A binary ripple counter consists of a series ~~combination~~ connection of flfs (T/JK types, withe the output of each flf connected to th CP input of the next higher order flf. The flf holding the least significant bit recieves the incoming clock pulses. The diagram below shows a 4-bit binary ripple counter. All J & K inputs are equal to 1. A small circle in the CP input indicates that the flf complements during a negative-going transition or when the output to which it is connected goes from 1 to 0. It is obvious that the lowest-order bit A_1 must be complemented with each count pulse. Every time A_1 goes from 1 to 0, the it complements A_2. Every time A_2 goes from 1 to 0, it complements A_3 and so on. The flfs change one at a time in rapid succession, and the signal propagates through the counter in a ripple fashion. Ripple counters are also called asynchronous counters.

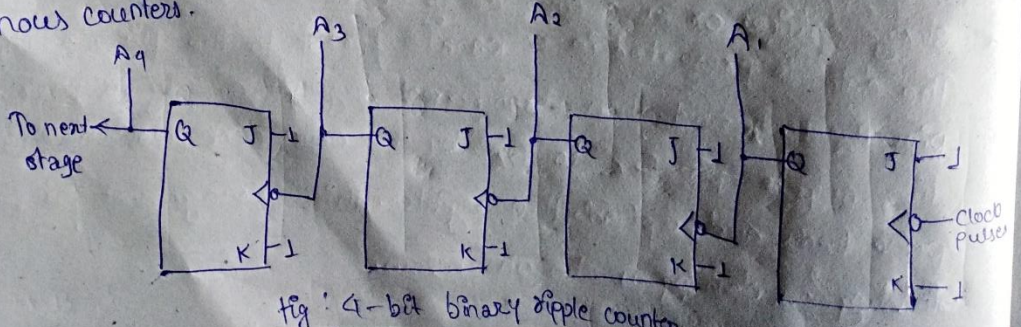


fig: 4-bit binary ripple counter

Table: Count sequence for a binary ripple counter.

Count sequence				Conditions for complementing flip-flops.
A_4	A_3	A_2	A_1	
0	0	0	0	Complement A_1,
0	0	0	1	Complement A_1. A_1 goes from 1 to 0 & complement A_2
0	0	1	0	Complement A_1,
0	0	1	1	Complement A_1. A_1 will go from 1 to 0 & complement A_2; A_2 will go from 1 to 0 & complement A_3
0	1	0	0	Complement A_1,
0	1	0	1	Complement A_1. A_1 will go from 1 to 0 & complement A_2
0	1	1	0	Complement A_1,
0	1	1	1	Complement A_1. A_1 will go from 1 to 0 & complement A_2; A_2 will go from 1 to 0 & complement A_3, A_3 will go from 1 to 0 & complement A_4.
1	0	0	0	and so on....

13) What are the various types of shift registers?

→ A register capable of shifting its binary information either to the right or to the left is called a shift register.

The various types of shift registers are as follows;

(a) Serial-in to Parallel-out (SIPO)

The register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.

(b) Serial in-to serial out (SISO)

The data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

(c) Parallel-in to serial-out (PISO)

The parallel data is loaded ~~to~~ into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

(d) Parallel-in to Parallel-out (PIPO)

The parallel data is loaded simultaneously into the register and transferred together to their respective outputs by the same clock pulse.

14) What do you mean by synchronous counter?

→ A synchronous counter is an MSI counter in which the input pulses are applied to all CP inputs of all flip-flops. The change of state of a particular flip-flop is dependent on the present state of other flip-flops.

15) Explain the 4-bit ripple counter and also draw a timing diagram.

→

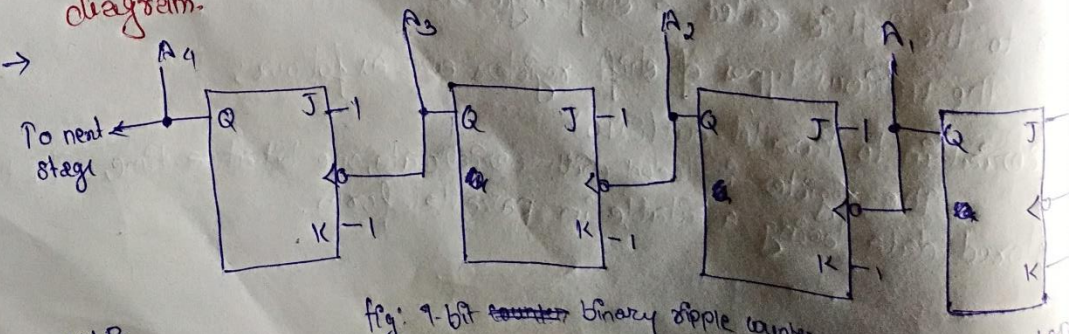


fig: 4-bit ~~counter~~ binary ripple counter