Module - 1

CO1 - Perform number system conversion, binary arithmetic operation and binary coding.

Number systems

Number system is an ordered set of symbols called digits. with rules defined for addition, substraction etc.

Base / Radix

Base / radix of a number system specifies the actual number of digits in its set.

Positional weight

It is the position of a digit with reference to decimal points.

In this module we are going discuss four types of number system.

1) Decimal number system.
2) Binary number system.

3) Octal number System.

4) Hexadecimal number system.

1) Decimal number System

Symbols - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Base / radix = 10

Positional weight = 10^x ($x = 0, 1, 2, 3, \ldots$)

2) Binary number System

Symbols - 0, 1

Base / radix = 2

Positional weight = 2^x

3) Octal number System

Symbols = 0, 1, 2, 3, 4, 5, 6, 7

Base / radix = 8

Positional weight = 8^x

4) Hexadecimal number System

Symbols = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

Base / radix = 16

number system conversions

1) Decimal to binary

$(46.25)_{10} \longrightarrow (\ )_2$?

2	46	0
2	23	1
2	11	1
2	5	1
2	2	0
	1	

$0.25 \times 2 = 0.5 \quad 0$

$0.5 \times 2 = 1.0 \quad 1$

$(101110.01)_2$

2) $(25.75)_{10} \longrightarrow (\ )_2$

2	25	
2	12	1
2	6	0
2	3	0
	1	1

2	25	
2	12	1
2	6	
	3	

$0.75 \times 2 = 1.5 \quad 1$

$0.5 \times 2 = 1.0 \quad 1$

$(11001.11)_2$

3) $(420.24)_{10} \longrightarrow (\ )_2$

2	420	
2	210	0
2	105	0
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
	1	1

$0.24 \times 2 = 0.48 \quad 0$

$0.48 \times 2 = 0.96 \quad 0$

$0.96 \times 2 = 1.92 \quad 1$

$0.92 \times 2 = 1.84$

$(110100100.001)_2$

2) Decimal to octal

1) $(86.45)_{10} \longrightarrow (\ )_8$

8	86	
8	10	6
	1	2

$45 \times 8 = 3.6 \quad 3$

$.6 \times 8 = 4.8 \quad 4$

$.8 \times 8 = 6.4 \quad 6$

$.4 \times 8 = 3.2 \quad 3$

$(126.3463)_8$

2) $(2142.53)_{10} \rightarrow (\quad)_8$

8	2142	
8	267	6
8	33	3
	4	1

$0.53 \times 8 = 4.24 \quad 4$

$.24 \times 8 = 1.92 \quad 1$

$.92 \times 8 = 7.36 \quad 7$

$.36 \times 8 = 2.88 \quad 2$

$(4136.4172)_8$

3) Decimal to Hexadecimal

1) $(214.35)_{10} \rightarrow (\quad)_{16}$

16	214	
	13	6

D

$0.35 \times 16 = 5.6.0.5$

0.6 x 16 = 9.6 9

0.6 x 16 = 9.6 9

$(D6.599)_{16}$

2) $(1731.56)_{10} \rightarrow (\quad)_{16}$

16 | 1731

16 | 108 3

6 12

0.56 x 16 = 8.96 8

0.96 x 16 = 15.36 15

0.36 x 16 = 5.76 5

0.76 x 16 = 12.16 12

$(6C3.8F5C)_{16}$

Any base number system to decimal (Binary to decimal)

1 $(1101.11)_2 \rightarrow (\quad)_{10}$

$= (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2})$

$= 8 + 4 + 0 + 1 + 0.5 + 0.25 = (13.75)_{10}$

2) $(10.1011.101)_2 \longrightarrow (\quad)_{10}$

$= (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2)$
$+ (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) +$
$(0 \times 2^{-2}) + (1 \times 2^{-3})$

$= (43.625)_{10}$

3) $(110110.1101)_2 \longrightarrow (\quad)_{10}$

$= (1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1)$
$+ (0 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3})$
$+ (1 \times 2^{-4})$

$= (54.81)_{10}$

octal to decimal

i) $(427.35) \longrightarrow (\quad)_{10}$

$(4 \times 8^2) + (2 \times 8^1) + (7 \times 8^0) + (3 \times 8^{-1}) + (5 \times 8^{-2})$

$= (279.45)_{10}$

$(4136.4172)_8 \rightarrow (\quad)_{10}$

$(4\times8^3)+(1\times8^2)+(3\times8^1)+(6\times8^0)+(4\times8^{-1})$
$+(1\times8^{-2})+(7\times8^{-3})+(2\times8^{-4})$

$=(2142.5397)_{10}$

Hexadecimal to Decimal

1) $(GA13C.2A)_{16} \rightarrow (\quad)_{10}$

$=(6\times16^3)+(10\times16^2)+(11\times16^1)+(12\times16^0)$
$+(2\times16^{-1})+(10\times16^{-2})$

$=(27324.1640)_{10}$

2) $(1A13C.7C)_{16} \rightarrow (\quad)_{10}$

$=(1\times16^3)+(10\times16^2)+(1\times16^1)+(12\times16^0)$
$+(7\times16^{-1})+(12\times16^{-2})$

$=(6a6b.6845.4843)_{10}$

Binary to Octal

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

1) $(000011011010.101101100)_2 \rightarrow (\quad)_8$

$(0332.554)_8$

2) $(001100101110.101111)_2 \rightarrow (\quad)_8$

$(1456.57)_8$

octal to binary

1) $(155.52)_8 \rightarrow (\quad)_2$

$(001101101.101010)_2$

2) $(3047.62)_8 \rightarrow (\quad)_2$

$(011000100111.110010)_2$

Binary to Hexadecimal

Hexadecimal	Binary
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
d	1 1 0 1
e	1 1 1 0
f	1 1 1 1

1) $(0001\,0011\,0111.\,1010\,1110\,1000)_2 \rightarrow (\quad)_{16}$

$(137.AE8)_{16}$

2) $(1111\,1011\,1010.\,1111\,1011)_2 \rightarrow (\quad)_{16}$

$(FBA.FB)_{16}$

Hexadecimal to binary

1) $(FBA.FB)_{16} \rightarrow (\quad)_2$

$(1111\,1011\,1010.\,1111\,1011)_2$

2) $(1CEF.213)_{16} \rightarrow (\quad)_2$

$(0001\,1100\,1110\,1111.\,0010\,0001\,0011)_2$

octal to hexadecimal

1) $(7324.456)_8$

$(111011010100.100101110000)_2$

$(ED4.970)_{16}$

2) $(5137.26)_8$

$(101001011111.010110000)_2$

$(A5F.58)_{16}$

Hexadecimal to octal

1) $(4ECE.43f)_{16}$

$(0100111011001110.0100001111111)_2$

$(047316.2077)_8$

2) $(ABCD.EF)_{16}$

$(0.1010\,1011\,1100\,1101.\,1110\,1111\,0)$

(125715.736)

Binary Addition

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

i)
```
  11
 1111 +
 1010
------
11000
```

ii)
```
  11
 1011 +
 1101
------
11000
```

```
 1 1 1 1
 1011.10   +
  100.1
-------------
10010.00
```

```
  1 1 1
 1 1 1 1.101   +
 1 0 1 1.1 1
---------------
1 0 0 1 1.0 1 1
```

Binary multiplication

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

i)

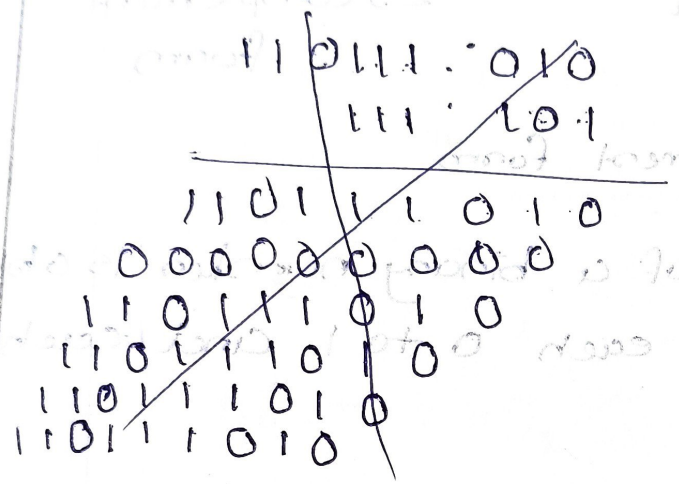
```
 1011  x
  101
-------
 1011
 0000
1011
-------
110111
```

ii)

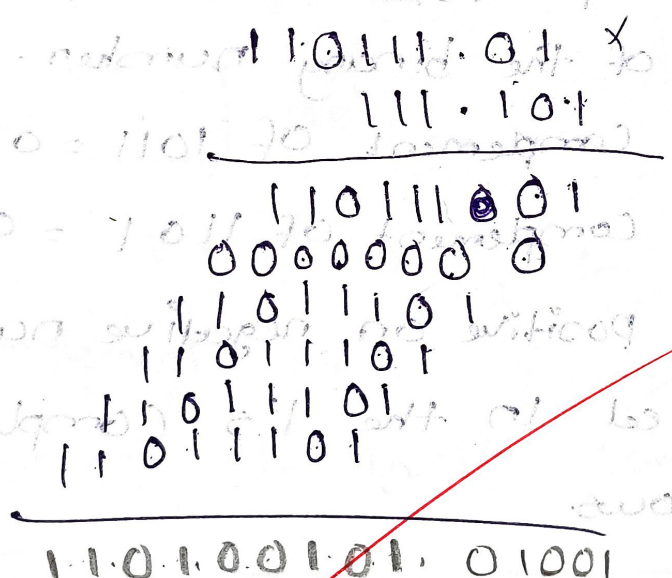
```
  1 1 0 1.1 0  x
    1 1  1.1 1
-----------------
  1 1 0 1 1 0
1 1 0 1 1 0
1 1 0 1 1 0
-----------------
101 0 1 1 0 1 0
```

iii) 110111.01 × 111.101

1101.1



110111.010 × 111.101



1101001101.01001

Signed number representations

1's Complement form

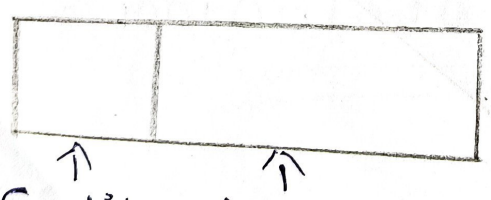
1's Complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

The Complemented value represents the negative of the binary number.

e.g. 1's Complement of 1011 = 0100

1's Complement of 1101 = 0010

Here a positive or negative number is represented in the 1's Complement form as follows.



In this form as shown above first bit will be the Signbit which is followed by 1's Complement of Original binary number

If Signbit = 0 the number is positive. And if Signbit = 1 binary number is negative.

1) Represent -7 1's complement form.

2 | 7
2 | 3 1
  | 1 1

$(7)_{10} \rightarrow (111)_2$

1's of 7 → 000

1	000

So $-7 = (1000)_2$ In 1's Complement form.

2) Represent -16 In 1's Complement form

2 | 16
2 | 8 0
2 | 4 0
2 | 2 0
  | 1 0

$(16)_{10} \rightarrow (10000)_2$

1's of 7 → ~~00111~~ 01111

1	01111

$-16 = (101111)_2$ In 1's

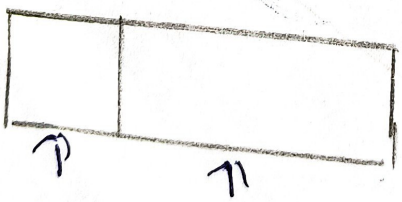
2's Complement form

2's complement of a binary number is obtained by adding 1 to the 1's complement of a number. That is 2's complement = 1's complement + 1

eg: 2's Complement = 0100 + 1

0101

Here positive or negative number is represented in 2's complement form as follows:



Sign bit 2's complement

* Represent -14 in 2's complement form.

2	19	
2	9	1
2	4	1
2	2	0
	1	0

$(-19)_2 \rightarrow (10011)_2$

2's of $-19 \rightarrow (01100)$

$$\begin{array}{r} 01100 \\ +\quad\ 1 \\ \hline 01101 \end{array}$$

1	01101

$(-19)_2 \rightarrow (101101)_2$

$(19)_2 \rightarrow (10011)_2$

2's of $= -19 \rightarrow (01100)$

$$\begin{array}{r} 01100 \\ +\quad\ 1 \\ \hline 01101 \end{array}$$

1	01101

$-19 = 101101$ in 2's complement form

represent -34 in 2's complement form

2	34
2	17 0
2	8 1
2	4 0
2	2 0
	1 0

$(34)_2 \rightarrow (100010)$

2's of 34 = (011101)

011101 +
1
011110

0 011110

34 = 0011110 2's Complement form

Binary Substraction using 1's Complement method

1111 −
1010

1111 +
0101

1 0100 +
1

0101

* Determined the 1's complement of negative (second number)
* Add to the positive number.
* If there is a carry the result is positive and the magnitude is given by removing in the carry and adding it to the LSB of the result
* If there is no carry the result is negative. And magnitude of the result is obtained by taking the 1's complement of the result.

Eg.

```
  1011  -
  1111
 ------
  1011  +
  0000
 ------
  1011
```

Actual result = -ve 0100

1)
```
 1000  -
 1010
------
 1000  +
 0101
------
 1101
```

result = -ve 0010

2)
```
 1010  -
 1110
------
 1010  +
 0001
------
 1011
```

= -0100

Binary Subtraction using 2's Complement method

```
 1010  -
 1111
------
 1010  +
 0001
------
 1011
```

1's to 2's

```
 0000 +
 0001
------
```

2's of 1011

```
 0100 +
 0101
------
```

∴ Carry number is -ve

Answer is -ve so 0101

ii)

```
1111 -          -1010
1010            -0101
----            -0101 +
 11                  1
1111 +          -------
0110            0110
----------
(1)0101
```

~~Answer is true~~

Answer is +ve = 0101

Binary Substraction using 2's complement

```
1111 -          1's -  1010
1010                   0101
----
1111            2's -  0101 +
0110                      1
----------             -------
(1)0101                0110
```

If there is carry result is +ve

So Final Answer is +0101

```
 1010 -
 1111
-------
 1010  +
 0011
-------
 1011
```

There is no carry

result is -ve

final result 1011

result is - 0101

```
 1111
 0000 +
    1
-------
 0001
```

```
 1011
 0100 +
    1
-------
 0101
```

```
 110101 -
 101110
--------
 110101  +
 010010
--------
(1)000111
```

+ 000111

```
 101110
 010001 +
      1
--------
 010010
```

Binary Codes

As digital System Operate in a binary System the numerous, alphabets and other special characters as to be Converted into binary this process of Converting into binary format

is called coding and codes the generated are called binary codes.

binary codes are classified into two

~~weig~~ weighted codes	unweighted codes
the digital codes with exibist shows the specific base assign to the bit positions is know as weighted codes. eg: BCD(8421), 2421, 5211 Applications Used to represents decimal digits in system like digital calculations, Voltmeters etc.	the digital code with does not exibist specific base assign to the bit positions is known as unweighted codes. eg: excess - 3 code Gray code Applications. To perform contained Arithmetic operation in digital components encodes etc.

Binary Coded Decimal (BCD) or 8421 Codes

Decimal	BCD (8421)
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

10 - 1010
11 - 1011
12 - 1100
13 - 1101
14 - 1110
15 - 1111

} Invalid

* Convert each of the following decimal numbers into BCD.

i) 9 - 1001

ii) 7.5 - 0111.0101

iii) 7287 - 0111 0010 1000 0111

iv) 26.8 - 0010 0110.1000

Q Find the decimal numbers represented by the following BCD or 8421 codes.

1) 1000 0001 - 81

2) 0011 1000 . 0001 - 38 . 1

3) 1001 0110 0111 1000 - 9678

BCD addition

i) 0001 0100

\+

1000 0001

1001 0101

9 5

ii) 1001 1001

\+

0101 0111

1 0000

Here BCD addition is invalid. Since there is a carry generated out of 4th bit. hence BCD addition cannot be performed.

iii)

0100	0110
+	
0101	0100
1001	1010

Here BCD addition is invalid since the sum is greater than 9.

Gray codes

~~Binary to gray~~

The code which exibist only a single bit change from 1 code number to the next is known as gray code.

Binary to gray

i) 1+0011101

10101011

Binary	gray
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

gray to binary

1 + 1 0 1 0 1 1

1 1 0 1 0 1 1

1 0 0 1 1 0 1

✗ 1 1 0 1 1 0 1 0 1

1 0 0 1 0 0 1 1 0

gray	binary
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

Alphanumeric Code

The code which contains Letters, numbers and other symbols are known as alphanumeric codes

eg: ASCII (American Standard for Information Interchange)

EBCDIC (code (extend binary code decimal interchange code)

ASCII Code.

It is a widely used alphanumeric code It is a 7 bit code in which Decimal digits are represented by BCD or 8421 which is preced by 011.

eg: 0 → 0110000

7 → 0110111

A → 1000001

B → 1000010

a → 1100001

b → 1100010

Parity check code for error detection

Parity check code is more widely used error detection code. Parity is nothing but the number of 1's in binary.

Decimal	BCD	old Parity bit	even parity bit
0	0000	1	0
1	0001	0	1
2	0010	0	1
3	0011	1	0
4	0100	0	1
5	0101	1	0
6	0110	1	0
7	0111	0	1
8	1000	10	0
9	1001	1	0

Error Connection Code - Hamming Codes.

Hamming code is a 7 bit Correcting Code in which four data bit can be transmitted this code formed is as follows.

D_7 D_6 D_5 P_4 D_3 P_2 P_1

where D_7 D_6 D_5 P_4 D_3 P_2 P_1 are data bits. and P_4, P_2, P_1 is Parity bits.

i) P_1 bit is even parity bit over $D_3\ D_5\ D_7$

ii) P_2 bit is even parity bit over $D_3\ D_6\ D_7$

iii) P_4 bit is even parity bit over $D_5\ D_6\ D_7$

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	1	0	0	0

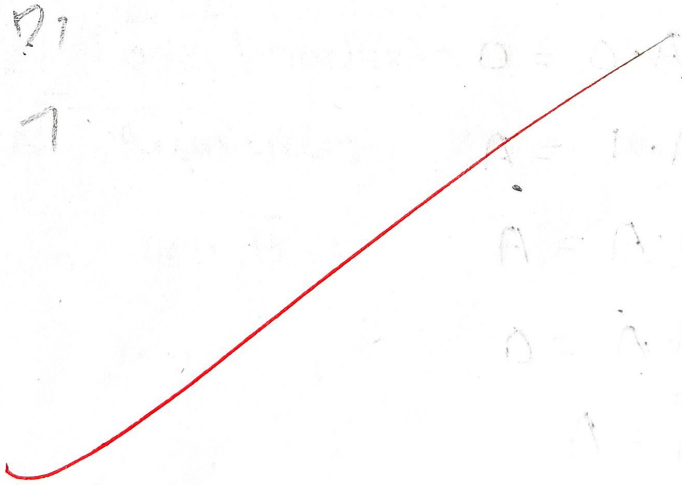
* n code checks bits 0110 into a 7 bit even parity hamming code.

Ans:

D_7	D_6	D_5	P_4	D_3	P_2	P_1
0	1	1	0	0	1	1

$D_3\ D_5\ D_7$

$D_3\ D_6\ D_7$



MODULE - 2

Axiomatic definition of Boolean Algebra.

It is a set of rules, laws and theorems by which all the logical operations can be expressed

Boolean Theorems / postulates

1) $A + 0 = A$
2) $A + 1 = 1$
3) $A + A = A$
4) $A + \bar{A} = 1$
5) $A \cdot 0 = 0$
6) $A \cdot 1 = A$
7) $A \cdot A = A$
8) $A \cdot \bar{A} = 0$
9) $\bar{\bar{A}} = A$
10) $A + B = B + A$
11) $A \cdot B = B \cdot A$
12) $(A + B) + C = A + (B + C)$
13) $(A \cdot B) \cdot C = A \cdot (B C)$

14) A (
15) (
16)
17)

14) $A(B+C) = AB + AC$

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

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

17) $\overline{AB} = \bar{A} + \bar{B}$

Demorgan's theorem

The first theorem states that compliment of a product of two variables is equal to the sum of the compliments of the variables individually.

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

Second theorems states that compliment of a sum of two variables is equal to products of the compliments of variables individually.

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

First theorem proof

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

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

Second theorem proof

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

A	B	A+B	$\overline{A+B}$	$\bar{B}\bar{A}$	$\bar{B}$	$\bar{A}\cdot\bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

By using boolean theorems following logical expressions can be obtained.

1) $A + AB = A$

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

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

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

Simplification of Boolean Functions Using boolean theorems

$A + AB = A$

$A(A+B) = A$

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

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

1) Prove that $AB + B\bar{C} + \bar{B}C = AB + C$

Ans: $AB + BC + \bar{B}C = AB + C(B+\bar{B})$

$= AB + C \cdot 1 \quad (\because B+\bar{B} = 1)$

$= AB + C$

2) Simplify $A\bar{B} + AB + \bar{A}\bar{B}$

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

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

$= B + \bar{B}\bar{A} \quad (\text{Since } B + \bar{B}A = \bar{B} + \bar{A})$

$= B + \bar{A}$

3) Simplify $A + A\bar{B} + \bar{A}B$

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

$= A + \bar{A}B \quad (\because 1 + \bar{B} = 1)$

$= \underline{A + B} \quad (\because A + \bar{A}B = A + B)$

4) Simplify $y = (\bar{A} + B)(A + B)$

Ans $y = (\bar{A} + B)(A + B)$

$= B + \bar{A}A$

$= B + 0 = \underline{B} \quad (\because \bar{A}A = 0)$

5) Simply $y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$

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

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

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

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

$= \underline{\bar{C}}$

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

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

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

Sum of Products and products of Sums

Product term

A learning boolean expression where one or more uncomplimented or complimented variables and multiplied (ANDed)

eg: $\bar{A}B$, $AB\bar{c}$, $A\bar{B}C$, $\bar{A}D$, $\bar{B}E$

Sum terms

It is a term in boolean expression where one or more complimented or uncomplimented variables ~~and~~ are added (ORed)

eg: $A+B$, $\bar{A}+B$, $\bar{A}+\bar{D}+\bar{C}$, $\bar{B}+\bar{G}+\bar{L}$

Sum of product (SOP) Boolean expression

It is a type of boolean expression where several product terms are some added (ORed) together.

eg: $\bar{A}B+\bar{B}C+BC\bar{D}$

$\bar{A}D+A\bar{B}\bar{C}+\bar{A}BC$

Product of sum (POS) Boolean expressions

It is a type of boolean expression where several some terms are multiplied (ANDed)

eg: $(A+\bar{B})(B+\bar{C})(\bar{A}+B+\bar{D})$

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

Canonical or Standard SOP

* It is a SOP in which all the product terms contains all the variables in the expression in either complimented or uncomplimented form.
* Products terms in canonical SOP are called minterms.

$Y = AB + BC + \bar{B}C$ → not canonical SOP

$Y = AB + \bar{A}B$ → canonical → minterms

$Y = AB\bar{C} + \bar{A}BC + ABC$ → canonical → "

$Y = AB + \bar{A}D + \bar{A}C$ → not canonical SOP

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

Canonical or standard POS

It is a POS in which all the sum terms contains all the variables in the expression in either complimented or uncomplimented form.

Sum terms in canonical POS are called max terms

eg. $(A+\bar{B})(\bar{A}+C)$ - not canonical POS

$(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+\bar{C})$ → canonical POS → max terms

$(A+\bar{B})(\bar{A}+\bar{B})$ → canonical POS

$(A+\bar{B}+C)(A+\bar{B}+D)$ non canonical POS

minterms and maxterms from truth table

	A B C D	minterms	maxterms
0	0 0 0 0	$\bar{A}\bar{B}\bar{C}\bar{D}$	$A+B+C+D$
1	0 0 0 1	$\bar{A}\bar{B}\bar{C}D$	$A+B+C+\bar{D}$
2	0 0 1 0	$\bar{A}\bar{B}C\bar{D}$	$A+B+\bar{C}+D$
3	0 0 1 1	$\bar{A}\bar{B}CD$	$A+B+\bar{C}+\bar{D}$
4	0 1 0 0	$\bar{A}B\bar{C}\bar{D}$	$A+\bar{B}+C+D$
5	0 1 0 1	$\bar{A}B\bar{C}D$	$A+\bar{B}+C+\bar{D}$
6	0 1 1 0	$\bar{A}BC\bar{D}$	$A+\bar{B}+\bar{C}+D$
7	0 1 1 1	$\bar{A}BCD$	$A+\bar{B}+\bar{C}+\bar{D}$
8	1 0 0 0	$A\bar{B}\bar{C}\bar{D}$	$\bar{A}+B+C+D$
9	1 0 0 1	$A\bar{B}\bar{C}D$	$\bar{A}+B+C+\bar{D}$
10	1 0 1 0	$A\bar{B}C\bar{D}$	$\bar{A}+B+\bar{C}+D$
11	1 0 1 1	$A\bar{B}CD$	$\bar{A}+B+\bar{C}+\bar{D}$
12	1 1 0 0	$AB\bar{C}\bar{D}$	$\bar{A}+\bar{B}+C+D$
13	1 1 0 1	$AB\bar{C}D$	$\bar{A}+\bar{B}+C+\bar{D}$
14	1 1 1 0	$ABC\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+D$
15	1 1 1 1	$ABCD$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$

Sop expression = $\bar{A}1BCD + \ldots\ldots + ABCD$

Pos. expression = $(A+1B+C+D)*(A+B+C+\bar{D})$

$\ldots\ldots (\bar{A}+1B+C+D)$

* Obtain the canodical sop of the function

$y = A+B$

Sol $y = A+B$

$= A.1 + 1B.1$

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

$= AB + A1\bar{B} + AB + \bar{A}\,1B$

$= AB + A1\bar{B} + \bar{A}\,1B$

11 10 01

3 2 1

8421

11
10
01

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

Obtained the canonical of the function

$y = A + BC$

$= A.1.1 + 1BC.1$

$= A.(B+1\bar{B})$ ~~$B.(C+1\bar{C})$~~ $+1BC(A+\bar{A})$

$= AB + A1\bar{B}.(C+\bar{C}) + ABC + \bar{A}BC$

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

Obtain the canonical sop of the function

$y = AB + \bar{A}\bar{B}CD$

$= AB.1.1 + \bar{A}\bar{B}CD.1$

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

$= ABCD + AB\bar{C}\bar{D}$

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

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

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

$= ABCD + AB\bar{C}D + AB\bar{C}B + AB\bar{C}\bar{D} + ABCD + ABCD$

$= ABCD + AB\bar{C}D + AB\bar{C}\bar{D} + ABC\bar{D} + A\bar{B}CD.$

1111 1110 1101 1100 1011

Obtain the canonical pos expression

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

$= (A+\bar{B}+0)(B+C+0)(A+\bar{C}+0)$

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

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

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

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

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

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

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

$\pi M(0,1,2,3,4)$

Obtain the canonical pos expression

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

$(A+B+0)(A+\bar{B}+0)(\bar{B}+C+0)$

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

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

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

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

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

$= (0,0,0)(0,$

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

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

$\Sigma m = (0, 1, 2, 3, 7)$

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

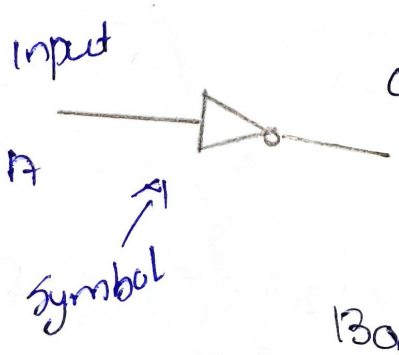
logic gates

- logic gates is a electronic circuit which makes logic decisions
- logic gates are classified into basic gates and universal gates.

Basic gates

OR, AND, NOT gates are called basic gates. Or basic building blocks of a digital system

NOT gate

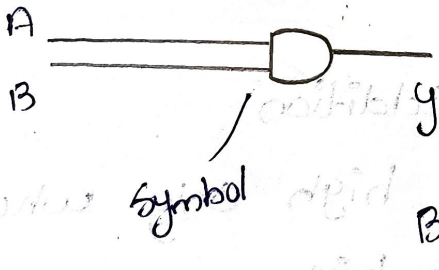


input	output
A	$y = \bar{A}$
0	1
1	0

$$y = \bar{A} \cdot 1 + \bar{A} \cdot 0$$
$$= \bar{A} + 0$$
$$= \bar{A}$$

* NOT gate has only one input and one output.
* It performes a basic logic function called inversion or complimentation.
* In this gate logic high appears at output only when input is logic low

AND gate



Symbol

$y = A \cdot B$

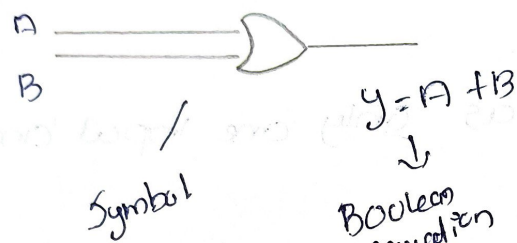
Boolean equation

Input		output
A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

$y = \bar{A}B \cdot 1 = \bar{A}B$

* AND gate has two or more inputs and one output
* It performes logical multiplication.
* Here output is logic high only when all the inputs are logic high

OR gate



Symbol

$y = A + B$

↓

Boolean equation

Input A	Input B	Output y
0	0	0
0	1	1
1	0	1
1	1	1

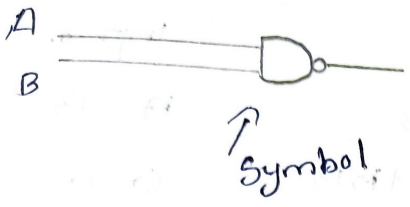
$$y = \bar{A}B + A\bar{B} + AB$$
$$= B(A + \bar{A}) + A\bar{B}$$
$$= B + A\bar{B}$$
$$= B + A = A + B$$

* OR gate has two or more inputs and one output
* It performs logical addition
* Here output is logic high only when any of the input is high.

Universal gates

NAND and NOR gate are called Universal gates because all other gates can be realised using NAND gates only and NOR gates only.

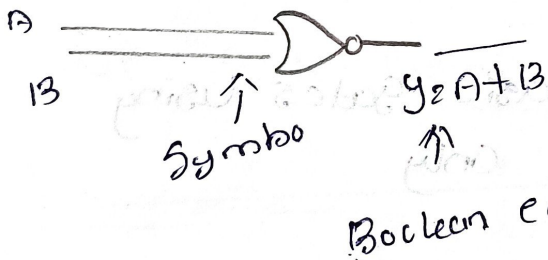
NAND gate



Input		Output
A	B	$y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

* NAND gate is nothing but Noted AND gate
* It has two or more inputs and one output
* Here output is logic high whenever either one of the input is logic low.

NOR gate



Input		Output
A	B	$y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

* NOR gate is noting but Noted OR gate
* It has two or inputs and one output
* Here Output is logic high ~~whenever~~ if only when all the inputs are logic low.

EXOR gate

Symbol

$y = A \oplus B$

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

boolean equation

Input		Output
A	B	$y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

EXNOR gate

Symbol

$y = A \odot B$

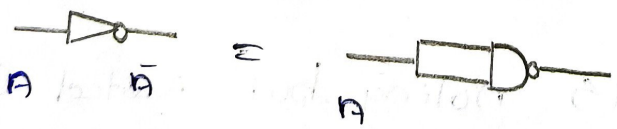
$= \bar{A}\bar{B} + AB$

boolean equation

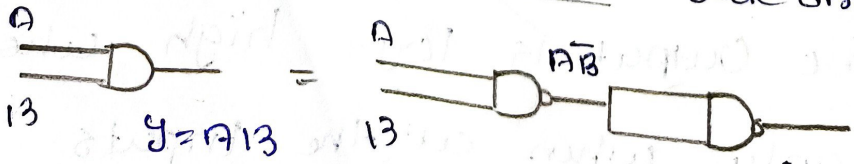
Input		Output
A	B	$y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

Realisation of basic gates using NAND gates only

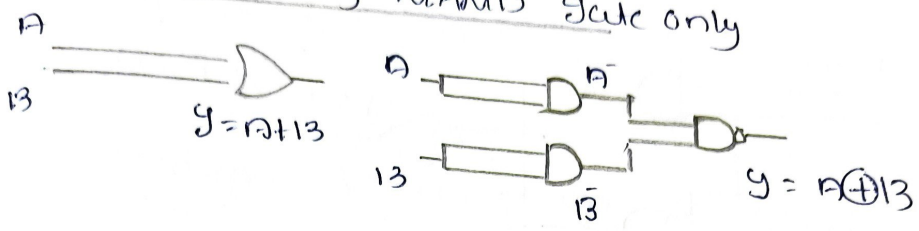
1) NOT gate using NAND Gate only



2) AND gate using NAND Gate only

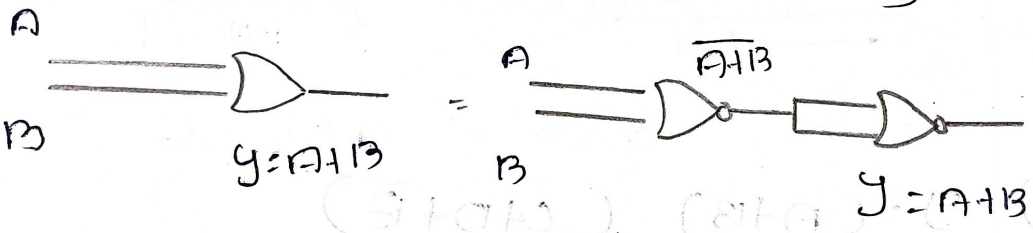


3) OR gate using NAND gate only

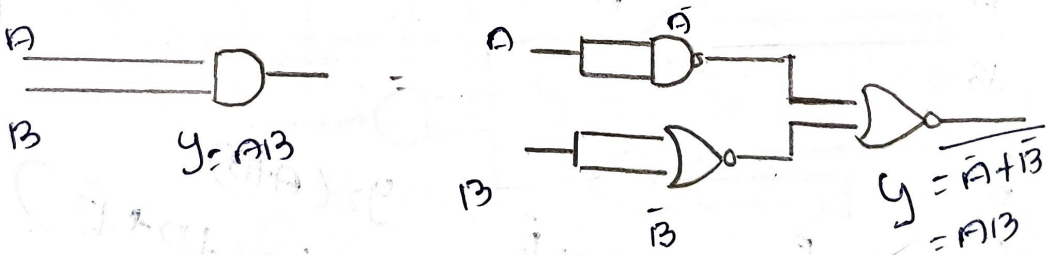


Realisation of basic gates using NOR gates only.

OR gate using NOR gates only

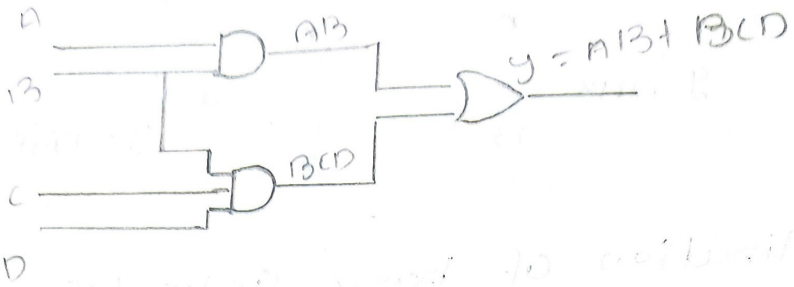


AND gate using NOR gate only.

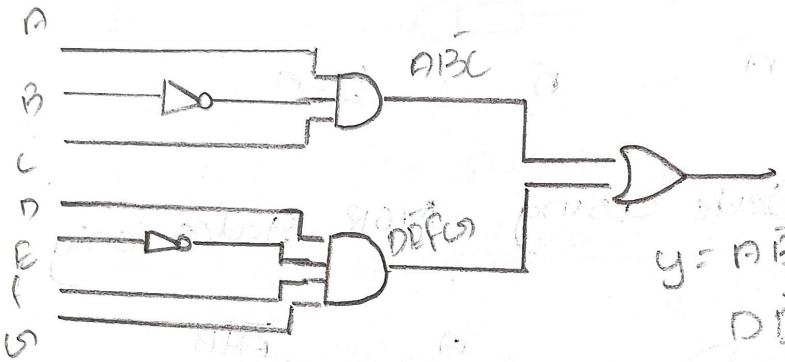


Implementation of logic circuits form boolean expressions.

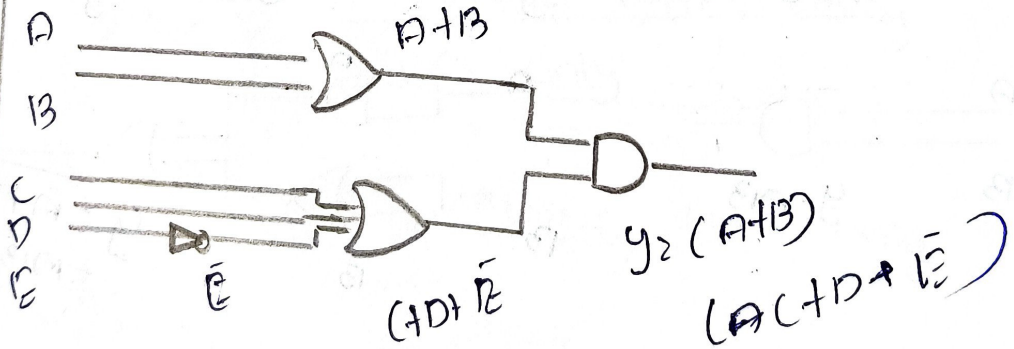
1) $y = AB + \bar{B}C\bar{D}$



2) $y = A\bar{B}C + D\bar{E}FG$



3) $y = (A+B)\ (C+D+\bar{E})$



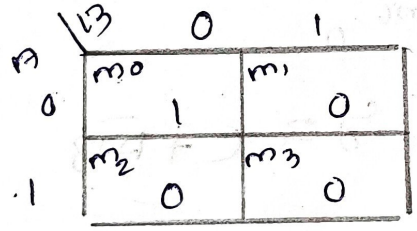
Karnaugh mapping (K map)

* Karnaugh is map is a graphical tool for finding the Sop or pos Simplification of a boolean function.

* A K map work by arranging the terms of expression in such a way that variables can be cancelled by grouping minterms or maxterms

2 variable K map.

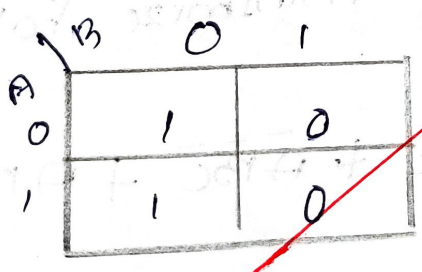
1)



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

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

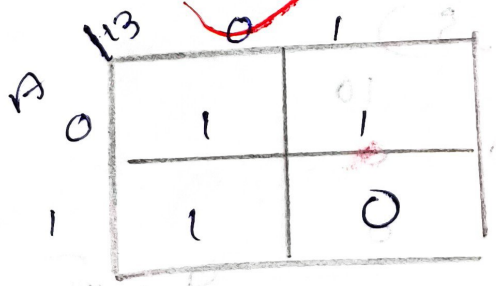
2)



$y = \bar{B}$

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

3)



$y = B + \bar{A}$

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

4)

A \ B	0	1
0	1	1
1	1	1

$y = 1$

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

3 variable k map (3 var)

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

* Minimize the following function using k map

$y = \Sigma m(1, 2, 3, 5, 7)$

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

$y = C + A\bar{B}$ $\quad \bar{A} = 0$, $A = 1$

* Use a k map to minimize following expression

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

$\Sigma m(0, 1, 4, 5)$ 421

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

$y = \bar{B}$

4 variable k map

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

Simplify the following expression using k map

$y = m_1 + m_3 + m_5 + m_7 + m_9 + m_{12} + m_{13}$

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

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

Simplify the following expression using k map

$y = \Sigma m(3, 2, 10, 11)$

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

$y = \bar{C} B$

Simplify the boolean equation using k map

$y = \Sigma m(8, 10, 11, 12, 13, 14, 15)$

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

$y = A +$

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

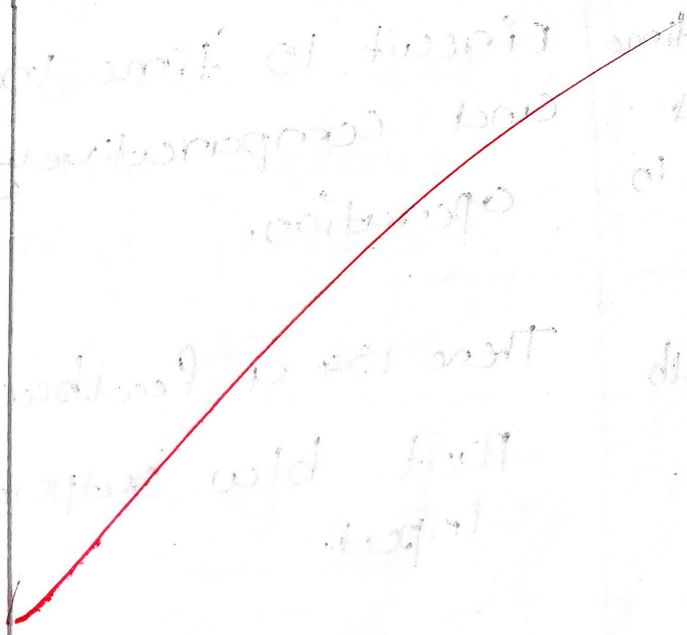
AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$y = A$

$y = \overset{Pos}{\Pi} m(9, 10, 11, 12, 13, 14, 15)$

SOP $= \Sigma m(0, 1, 2, 3, 4, 5, 6, 7, 9)$

$\bar{y} = \Pi m(6, 2) = \Sigma m(1, 3)$



Module - 3

CO3 - Design Combinational Circuits

Comparision of Combinational and Sequential Circuits

	Combinational	Sequential
*	Here the output depends on present input only.	Here the output depends on present input as well as passed output.
*	The circuit is time independent and it is very fast in operation.	Circuit is time independent and comparatively slowly operation.
*	There is no feedback path blw output and input	There is a feedback path blw output and input.
*	~~Combination~~ main building of combination circuit is logic gates.	main building combination circuit is flip flops.

Combination Circuit	Sequential Circuit
~~Circuit~~ mainly used for mainly arithmetic and boolean operation	are used for data storing
It doesn't have ~~block~~ clock	It does have block clock
Operation of combination is very simple	Operation sequential is very complex
eg: adder, subtractors, encoders, decoders, multiplexer, demultiplexer	eg: flip flops, shift registers, counters, etc.

Half Adder

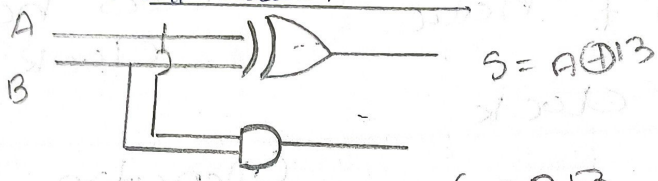
It is a logic circuit that adds two binary bits.

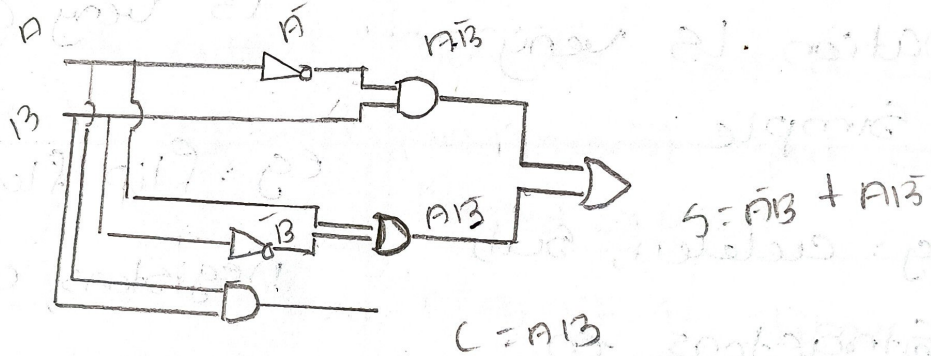
It has two input A, b and two output call Sum(s) and Carry(c)

Truth table

Input		output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half adder circuit





Full adder

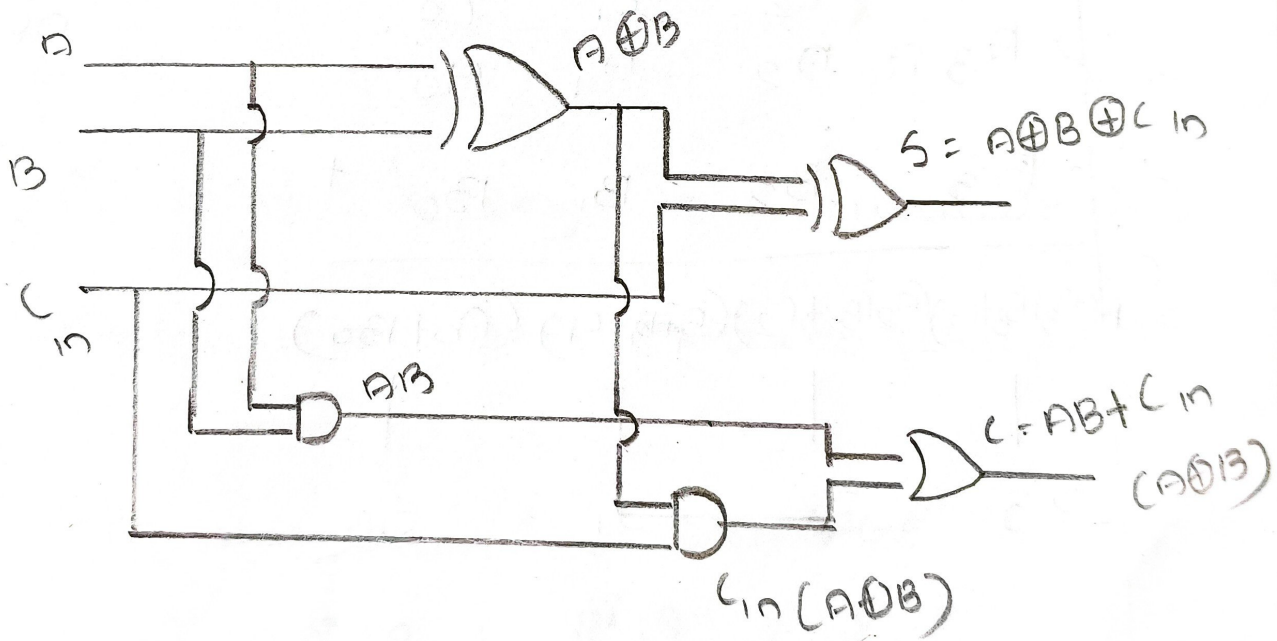
It is a logic circuit has three binary bits.

It has 3 input a, b, c and two outputs called sum and carry.

Input			output	
A	B	C_{in}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

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



4 bit Parallel adder

* To add two four bit numbers four bit parallel adder uses four full adder circuit.
* Output carry ~~of~~ BF one full adder connected to input carry of the next full adder
* Let $a_3a_2a_1a_0$ and $B_3B_2B_1B_0$ be the two four bit numbers

C_3 C_2 C_1 C_0

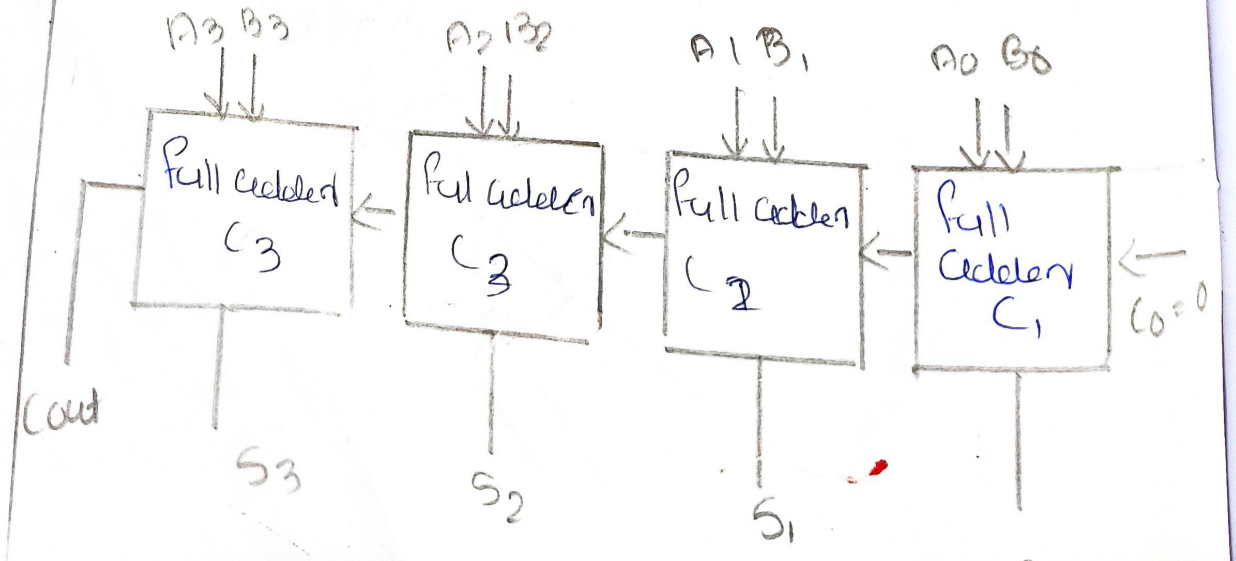
A_3 A_2 A_1 A_0

B_3 B_2 B_1 B_0 +

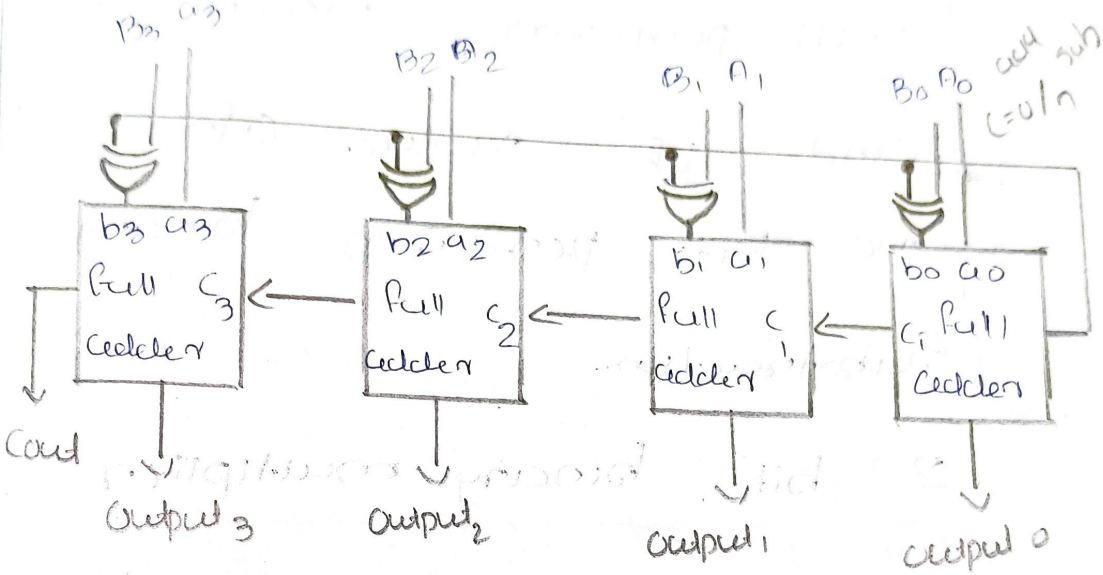
C_{out} $(A_3+B_3+C_3)(A_2+B_2+C_2)(A_1+B_1+C_1)(A_0+B_0)$

| | | | |

S_3 S_2 S_1 S_0



4 bit binary adder / subtractor



* A bit binary is calculate both addition and subtraction.

* The circuit ~~consists~~ consist of four full adder and a control line C which decides a binary value of either zero or one.

* If value of c zero circuit will perform binary addition and if value of c is one bit perform binary substraction.

2 bit binary multiplier

$A_1 A_0 \times$

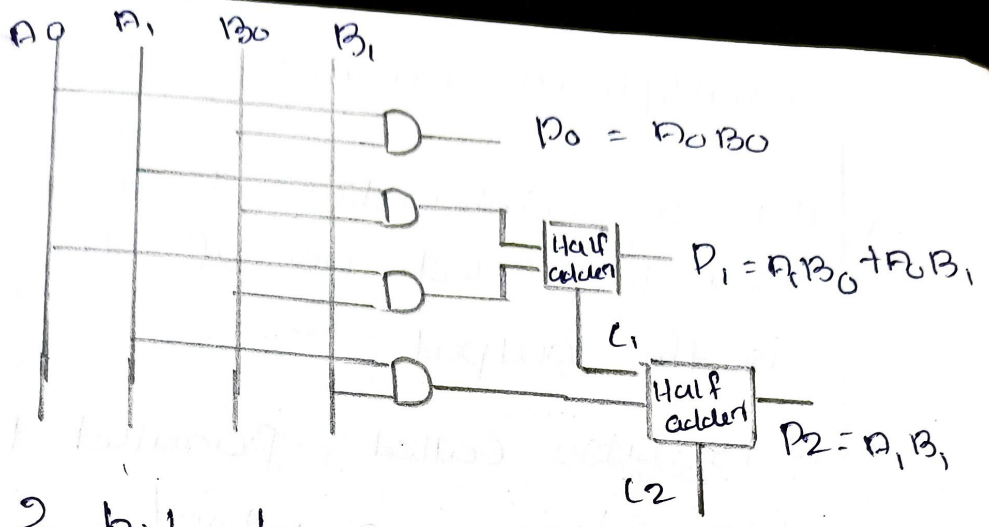
$B_0 \quad B_0 B_0$

$A_1 B_0 \quad A_0 B_0$

$A_1 B_1 \quad A_0 B_1$

$C_2 \quad A_1 B_1 \quad A_1 B_0 \; A_0 B_0 \quad A_0 B_1$

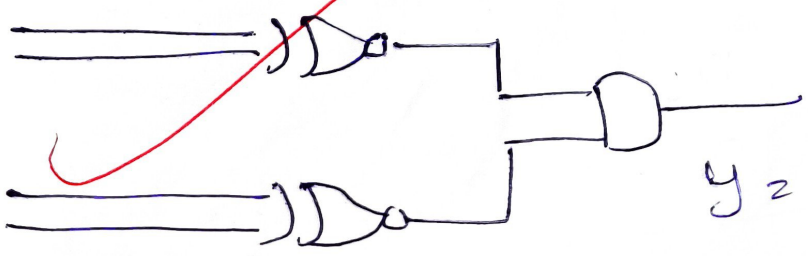
$P_3 \quad P_2 \quad P_1 \quad P_0$



* A 2 bit binary multimplier of a comple- ssion lesen for two binary numbers.
* And gate is basic multiplier which is used for one bit multiplication

magnitude comparator (Digital Comparator)

Digital comparator use to compare the ma- gnitude of two cont . The circuit determine where two number equal or not EXOR gate basic comparator as output is high when the two inputs are not equal And output is low when the input are equal



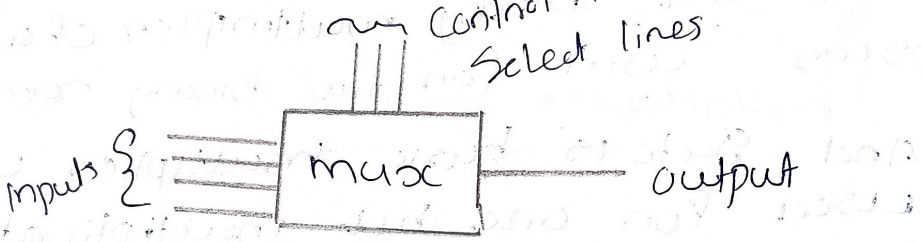
y = 1 For equal

y = 0 For equal

multiplexer (mux)

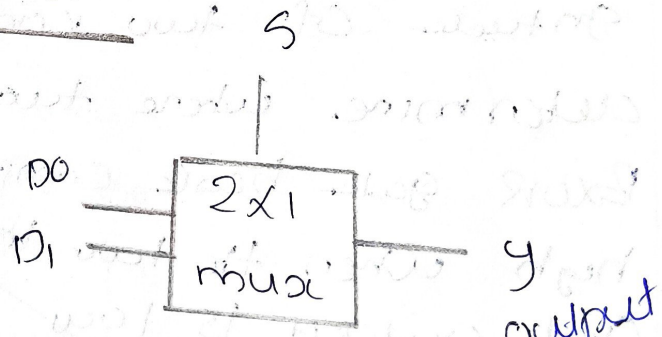
$\bar{S}$ - NOT
$\cdot$ - AND
$+$ - OR

* It is a combination logic circuit used to select one of its input as the output.
* It is also called parallel to serial converter.



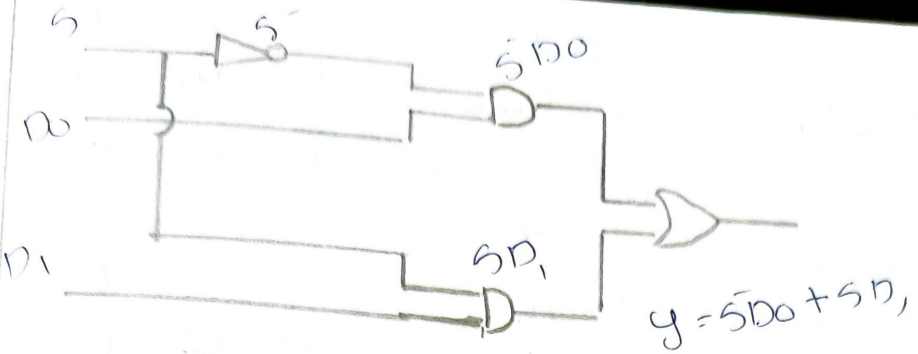
number of inputs = 2 number of control / select lines.

2 x 1 mux



Input S	output Y
0	D_0
1	D_1

$$y = \bar{S} D_0 + S D_1$$



two inputs and one output

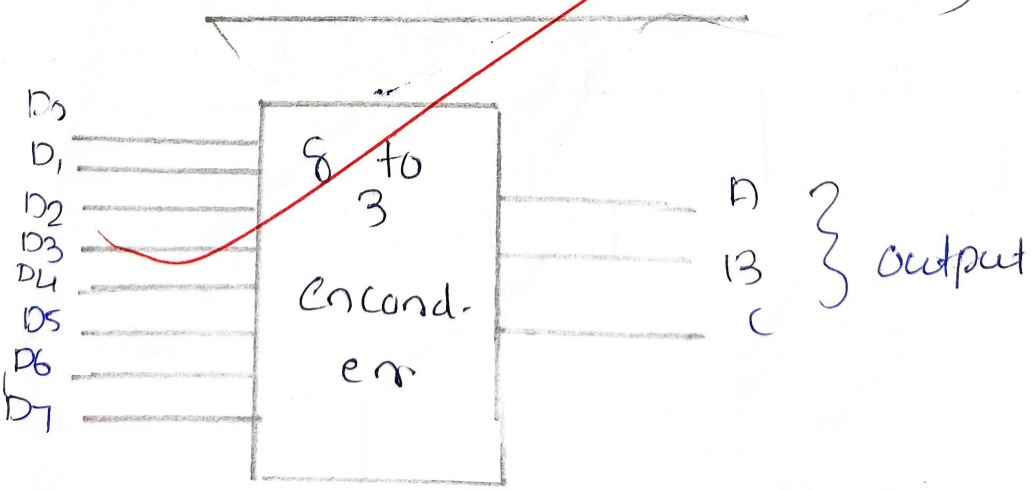
Encoder

It is a combination circuit which accepts one of its inputs which represent a digit and coverts it to a coded outputs such as binary.

Here ~~number of digits~~

no: of inputs = 2. no: of outputs.

8 to 3 encoder (octal to binary encoder)

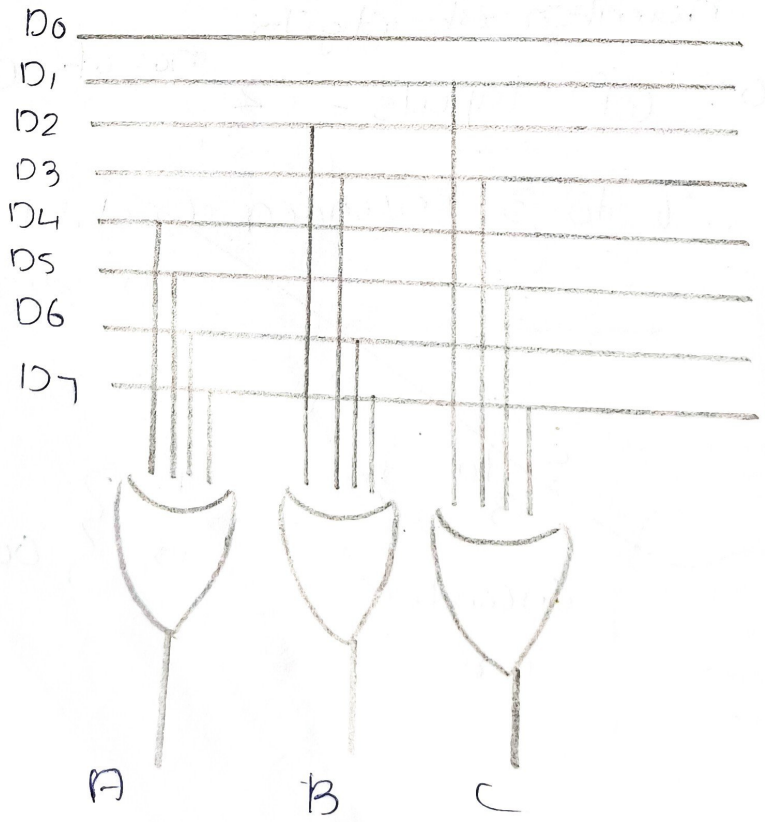


Inputs								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

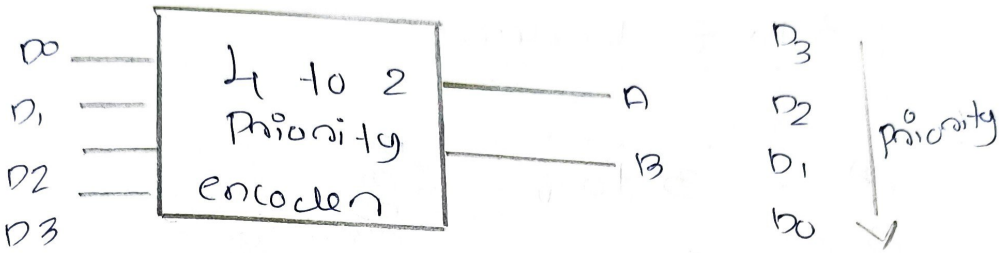
$A = D_4, D_5, D_6, D_7$

$B = D_2, D_3, D_6, D_7$

$C = D_1, D_3, D_5, D_7$



4 to 2 priority encoder



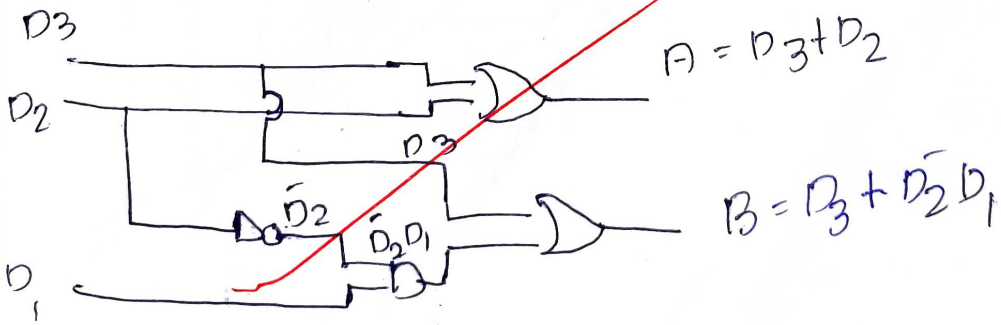
Input				Output	
D_3	D_2	D_1	D_0	A	B
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$$A = \bar{D_3} D_2 + D_3$$

$$= D_3 + D_2$$

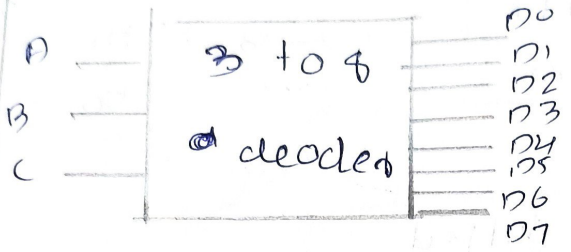
$$B = \bar{D_3} \bar{D_2} \bar{D_1} + D_3$$

$$= D_3 + D_2 \bar{D_1}$$



Decoder

3 to 8 decoder



Input			Output							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{A}\bar{B}\bar{C}$

$D_1 = \bar{A}\bar{B}C$

$D_2 = \bar{A}B\bar{C}$

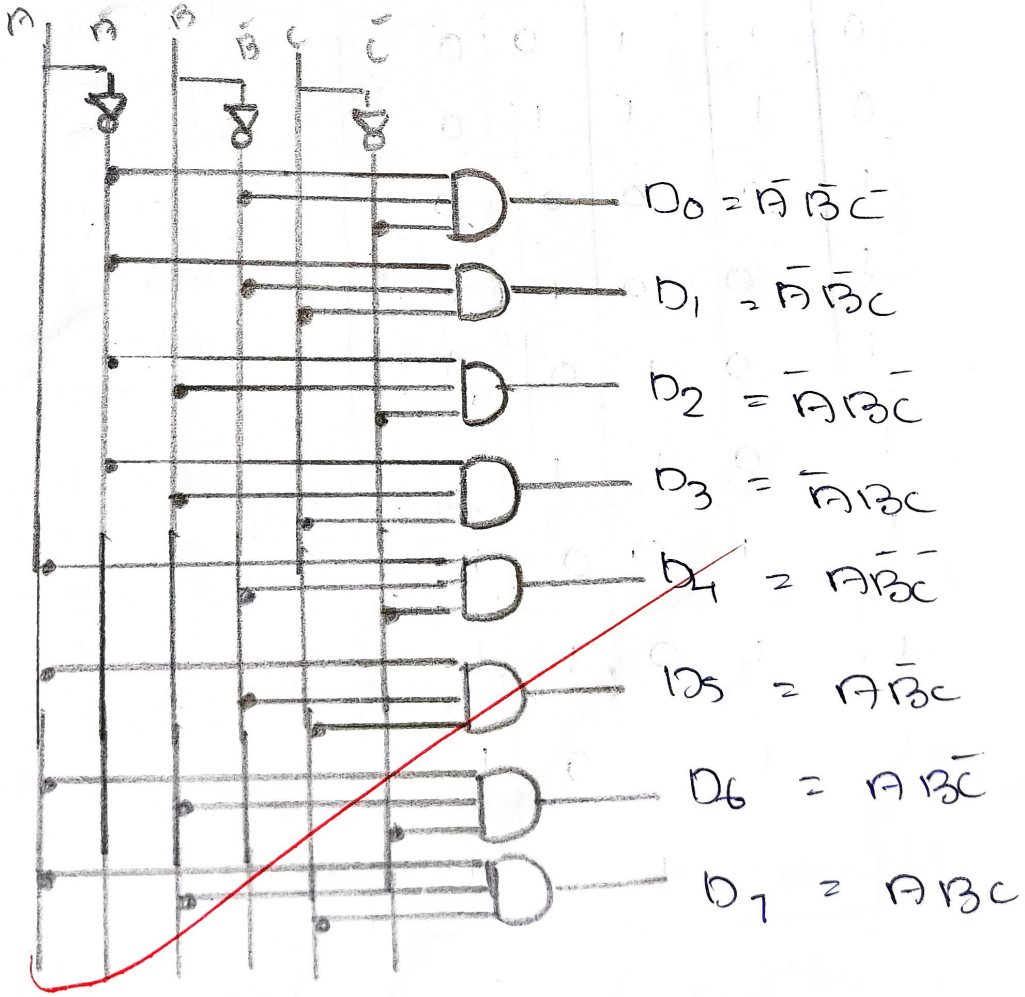
$D_3 = \bar{A}BC$

$D_4 = A\bar{B}\bar{C}$

$D_5 = A\bar{B}C$

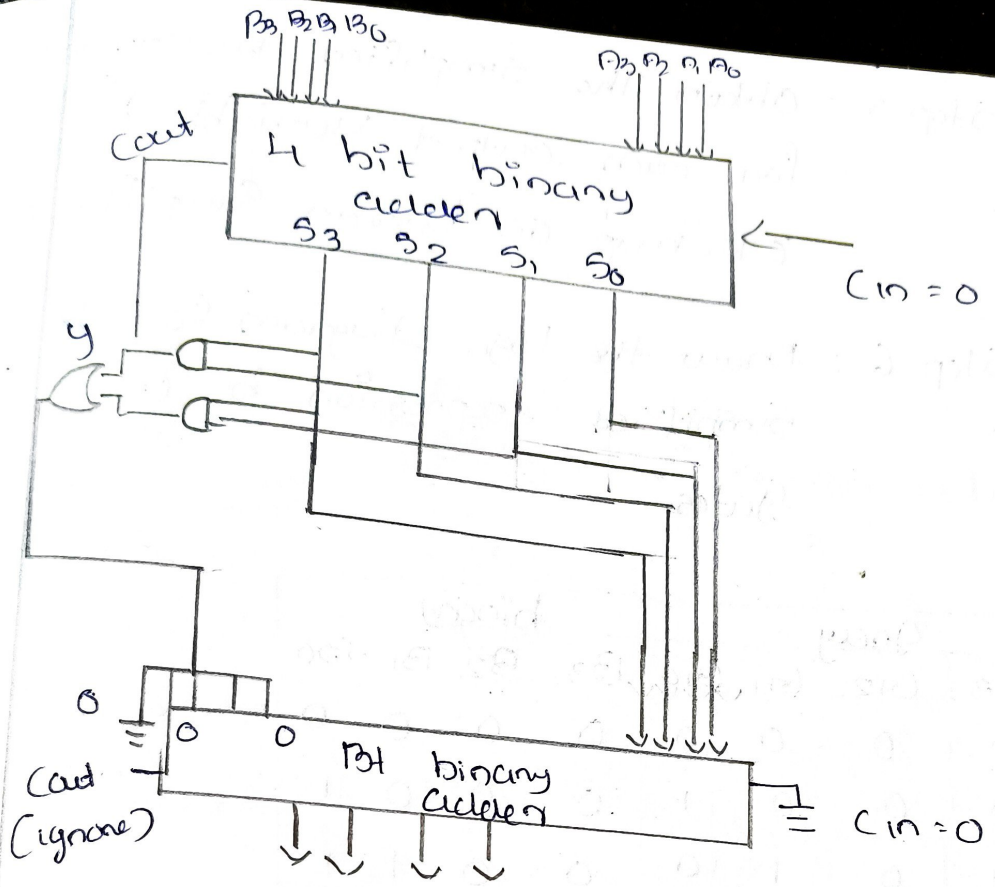
$D_6 = AB\bar{C}$

$D_7 = ABC$



BCD adder

Input				output
S_3	S_2	S_1	S_0	y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



Design and Usis procedure of combinational logic circuits

Step 1 : Analyze the given Problem

Step 2 : Determine the number of input and output variables

Step 3 : Assigning letter Symbols to input and output variables.

Step 4 : Derive the truth table indicating the relationship b/w input and output variables.

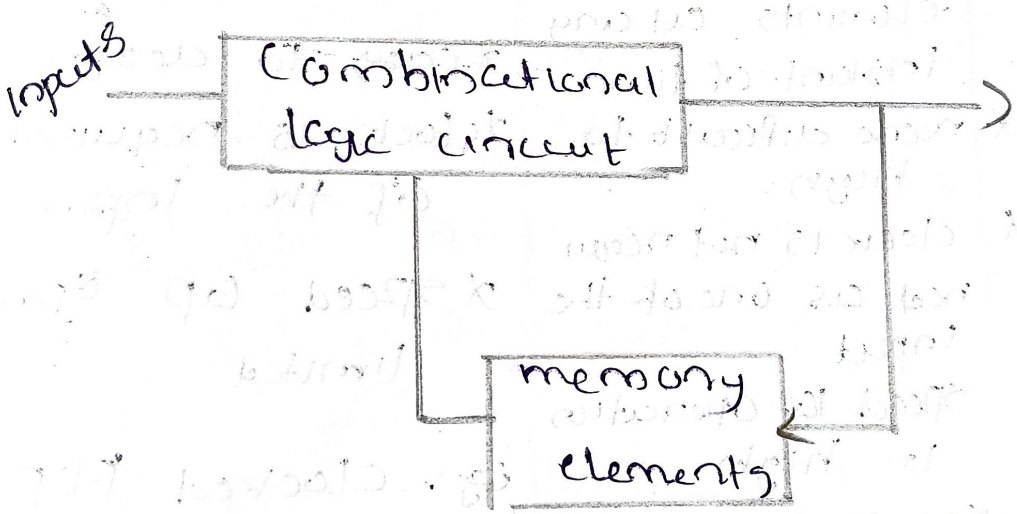
Step 5 : Obtain the simplified boolean functions for each output variable by using K-map or boolean simplifications.

Step 6 : Draw the logic diagram for the above simplified expression by using logic gates.

Gray				binary			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Module - 4

CO4 - Design Sequential Circuit



Bolock diagram of Sequential Circuit

* Sequential Circuit is digital Circuit whose output depends on present Input as well as Past output.

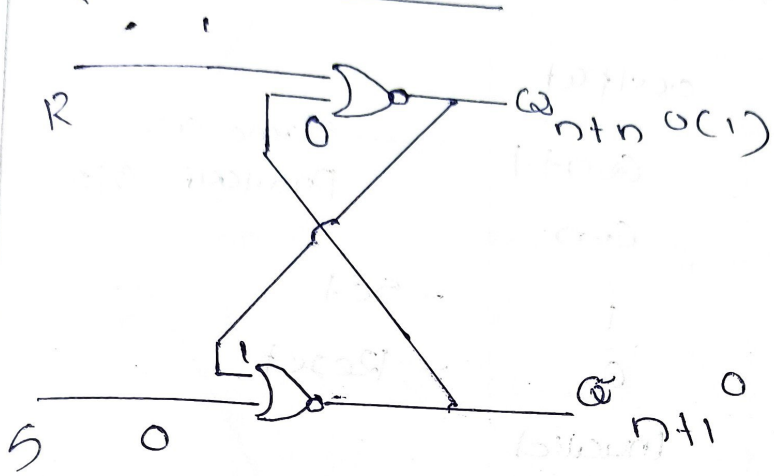
Comparison of Synchronous and Asynchronous Sequential Circuit

Asynchronous Sequential circuit	Synchronous Sequential Circuit
* Input signal can affect memory elements at any instant of time	Input signals can affect memory at discreet instant of time
* more difficult to high	* easy to design
* clock is not required as one of the input	* clock is required as one of the input
* speed of operation is high	* speed of operation limited
eg: unclocked flipflops	eg: clocked flip flops

Latches (unclocked flipflops)

* It is the simplest kind of sequential circuit
* It is capable of storing bit of information that is either logic 1 or logic 0.

SR NOR latch



R = 1 = S0 $Q_n = 0$

$Q_{n+1} = 0$

R = 1 S = 0 $Q_n = 1$

$Q_{n+1} = 0$

$Q_{n+1} = 0$

NOR

Input		output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Case - 4

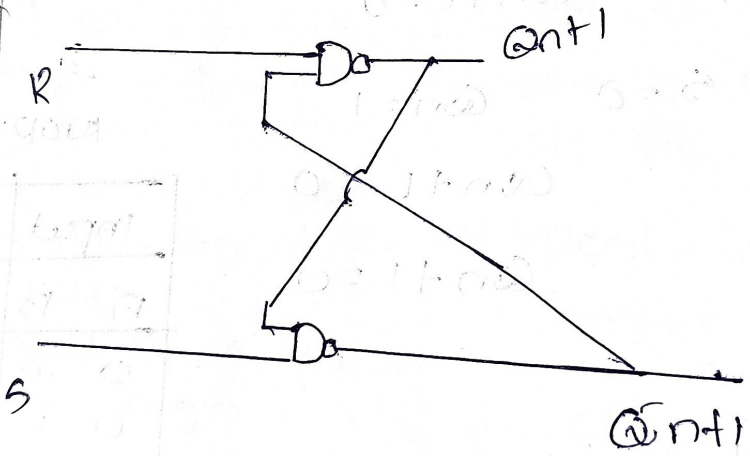
R = 1 S = 1

Q_{n+1} = Invalid

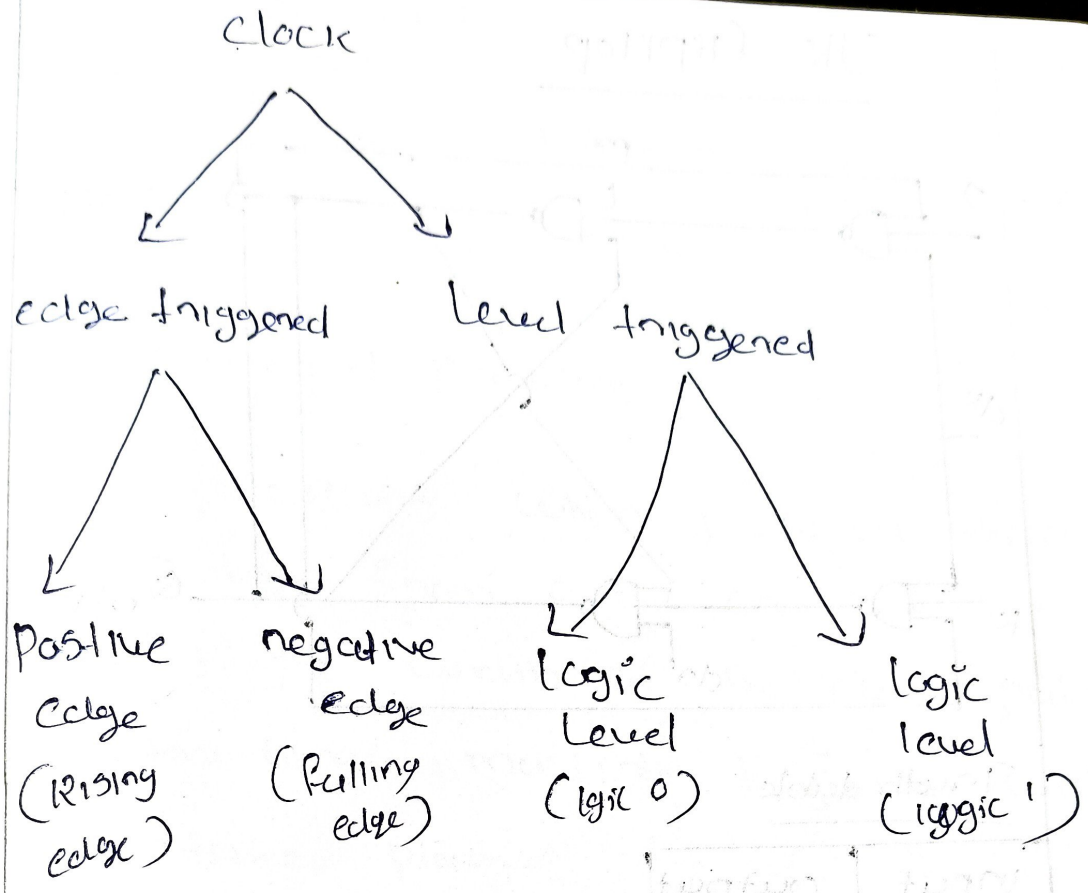
Truth table

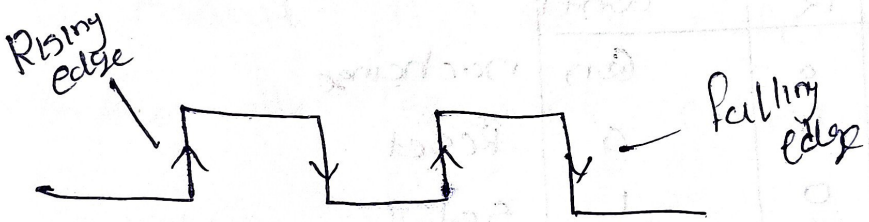
input		output
R	S	Q n+1
0	0	Q n — Same as Previous O/P
0	1	1 — set
1	0	0 — Reset
1	1	Invalid

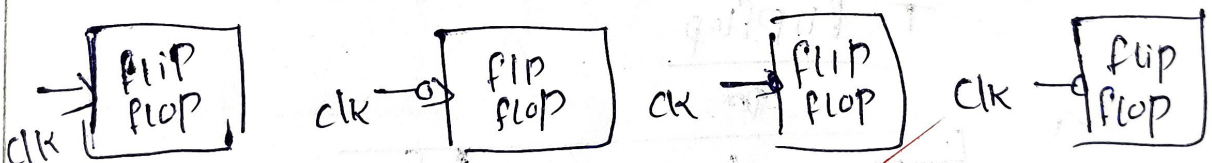
SR NAND latch



~~truth table~~







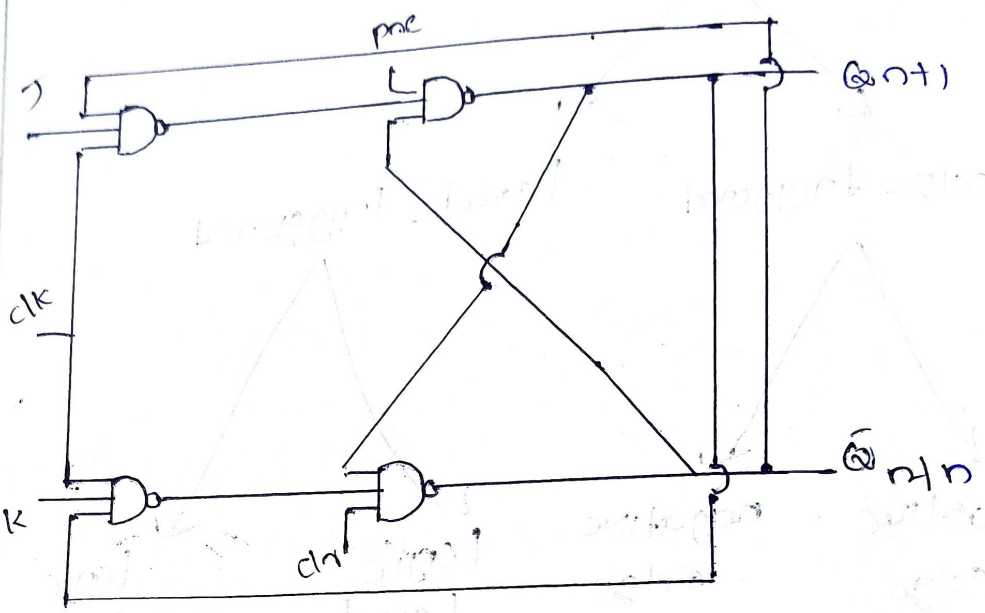
Symbol of positive edge triggered flip flop

Symbol of negative edge triggered flip flop

Symbol of logic level triggered / positive level

Symbol of logic level triggered / negative level

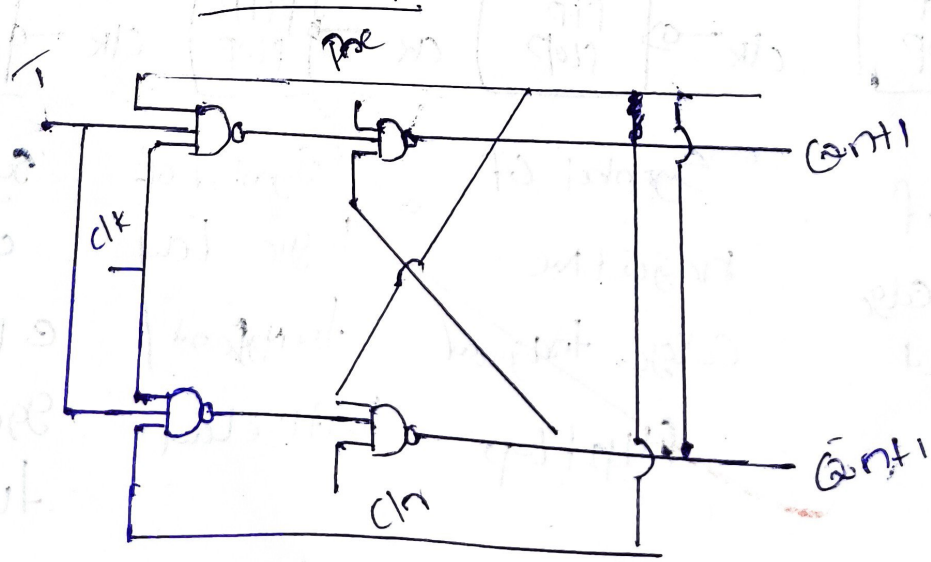
JK flipflop



Truth table

input		output	
J	K	Q n+1	
0	0	Qn	nochange
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}$n	toggle

T flipflop



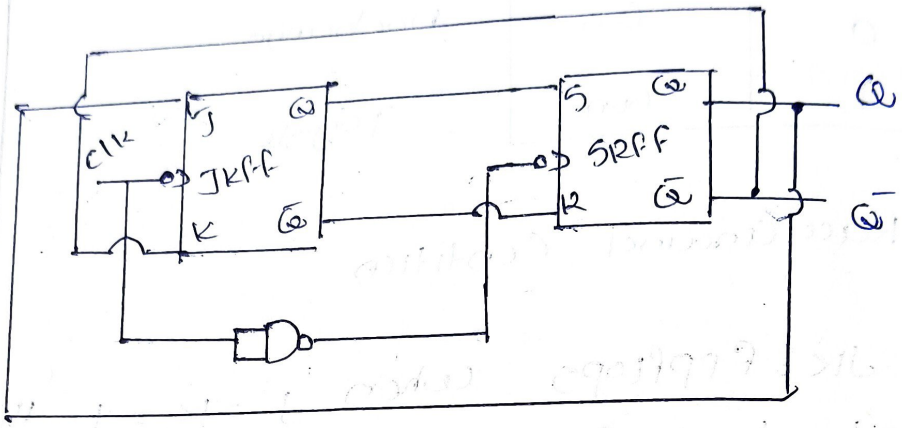
Input	Output.	
T	Qn+1	
0	Qn	nochange
1	$\bar{Q}n$	Toggle

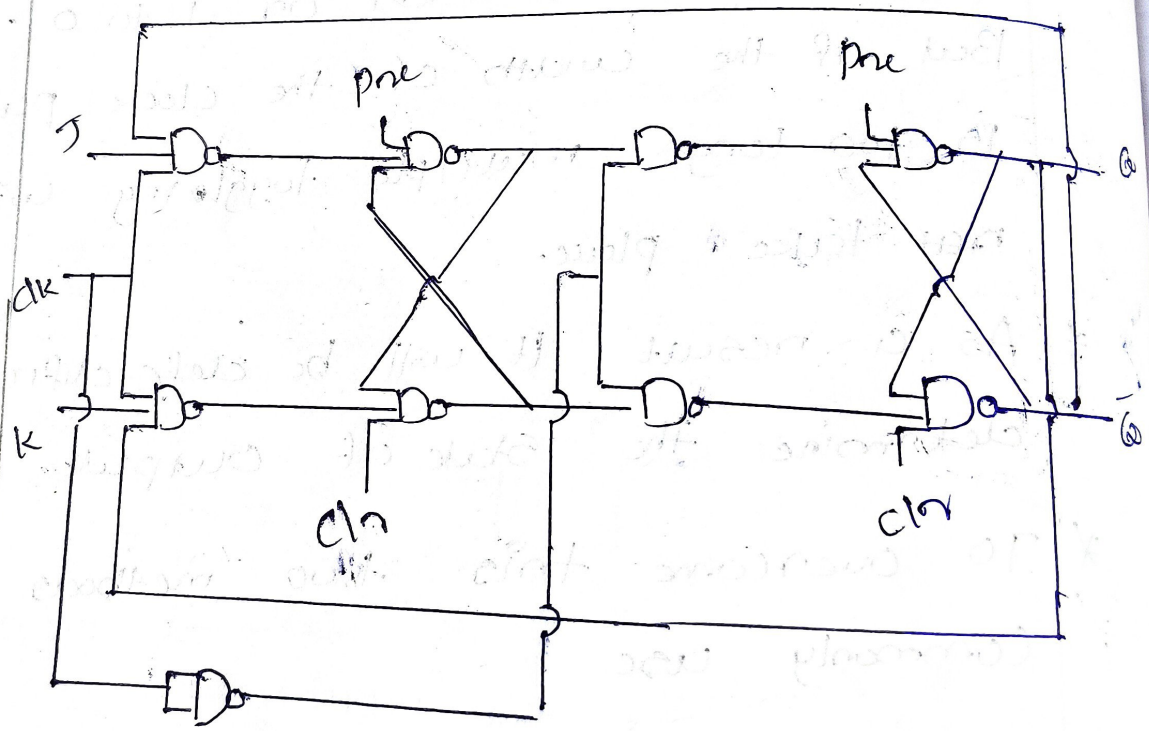
Race Around Condition

* In JK Flipflops when J = K = 1 the output will toggle from 0 to 1 or 1 to 0. But if the width of the clock pulse is too long multiple toggleing will ~~not~~ take place.
* As a result it will be ~~clefe~~ difficult to determine the state of output.
* To overcome this two methods are commonly use

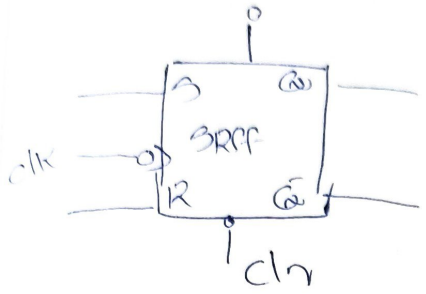
1) Use of ~~heelge~~ edge triggering
2) Use of master slave JK flipflops

master slave JK flipflop



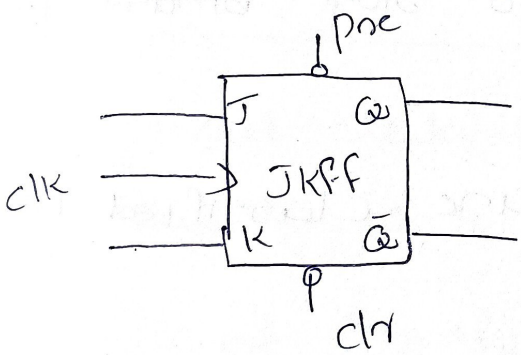


logic Symbols of flipflops



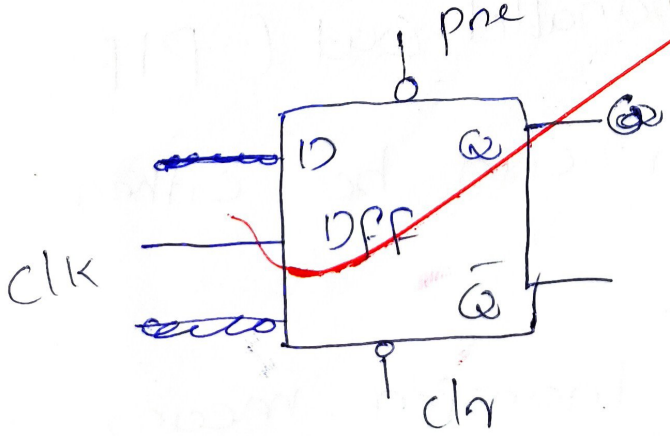
negative edge triggered SRFF

Positive edge triggered JKFF

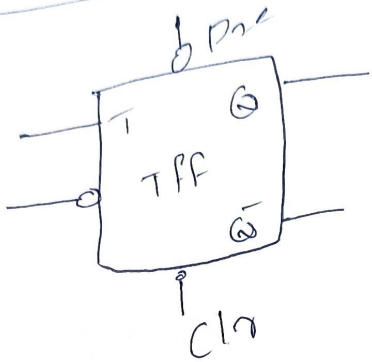


Positive edge triggered JKFF

Positive level triggered D flipflop



Negative level triggered T flip flop



Shift Registers

* Shift register is a set of flip flops logically connected to store n-bit binary data.
* On the mode of Shift registers are classified into four

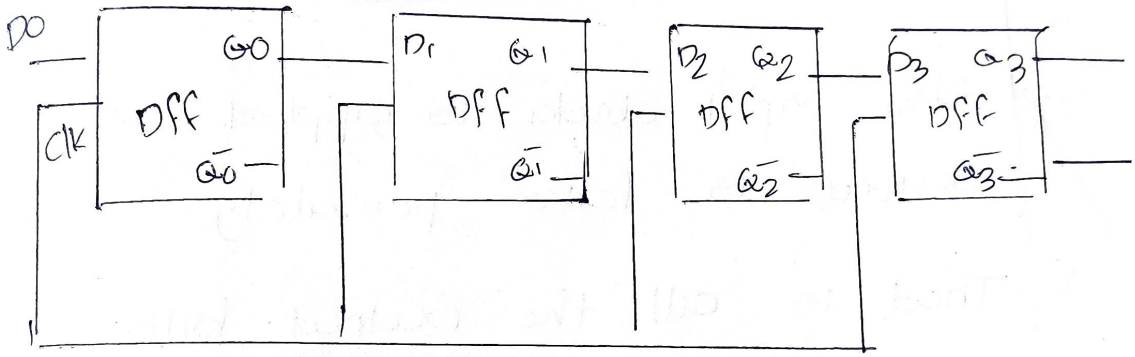
1) Serial In Serial out (SISO)
2) Serial In parallel out (SIPO)
3) Parallel In Serial Out (PISO)
4) Parallel In parallel out (PIPO)

* data transfer can be either serial or parallel
* Serial data transfer means we can

Input only a single data at a time.

Here parallel data transfer means we can input a set of data same time.

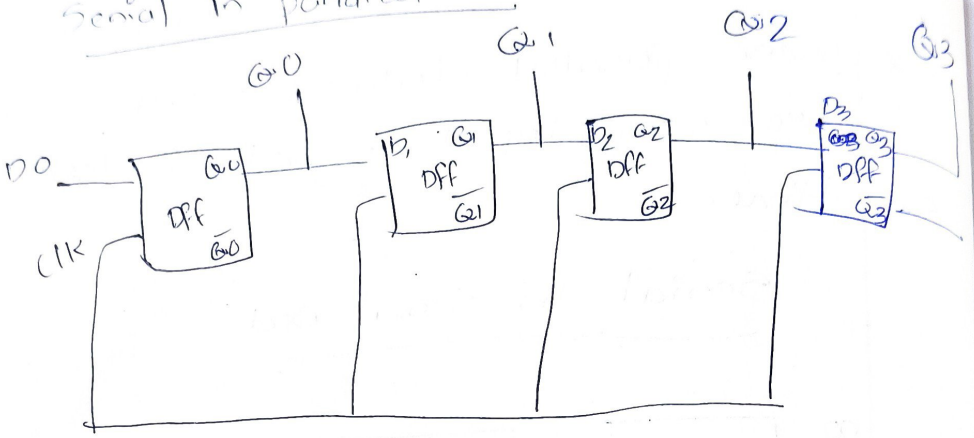
Serial In Serial Out



Here input data is applied serialy and output is also taken serialy.

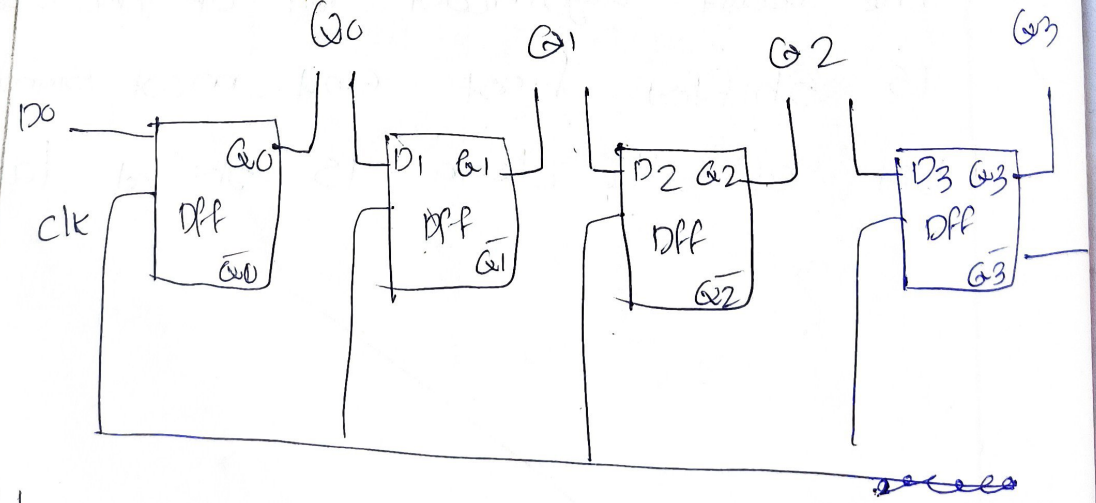
The least significant bit of the data is shifted first and most significant bit of the data is shifted last.

Serial In parallel out



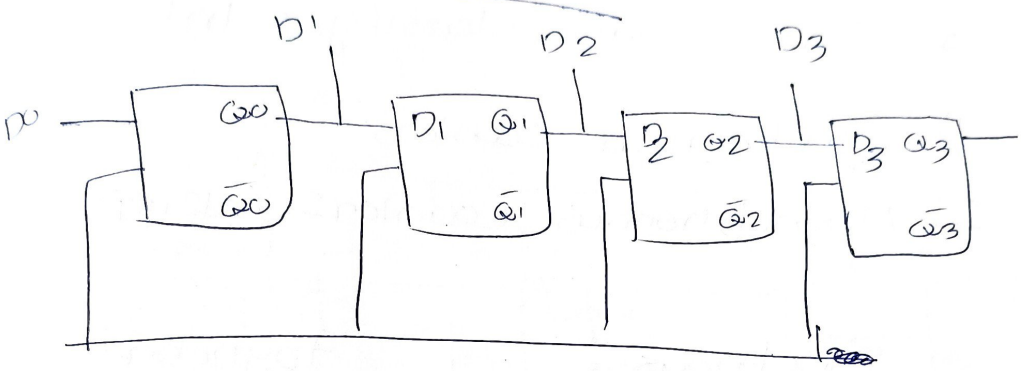
* Here Input data is applied serially and output is taken parallely.
* That is all the output bits are available simultaneously

Parallel In serial out



* Here input data applied parallely
* and output is also taken parallely.

Parallel In Parallel out



Here Input data is applied parallely output is taken serialy.

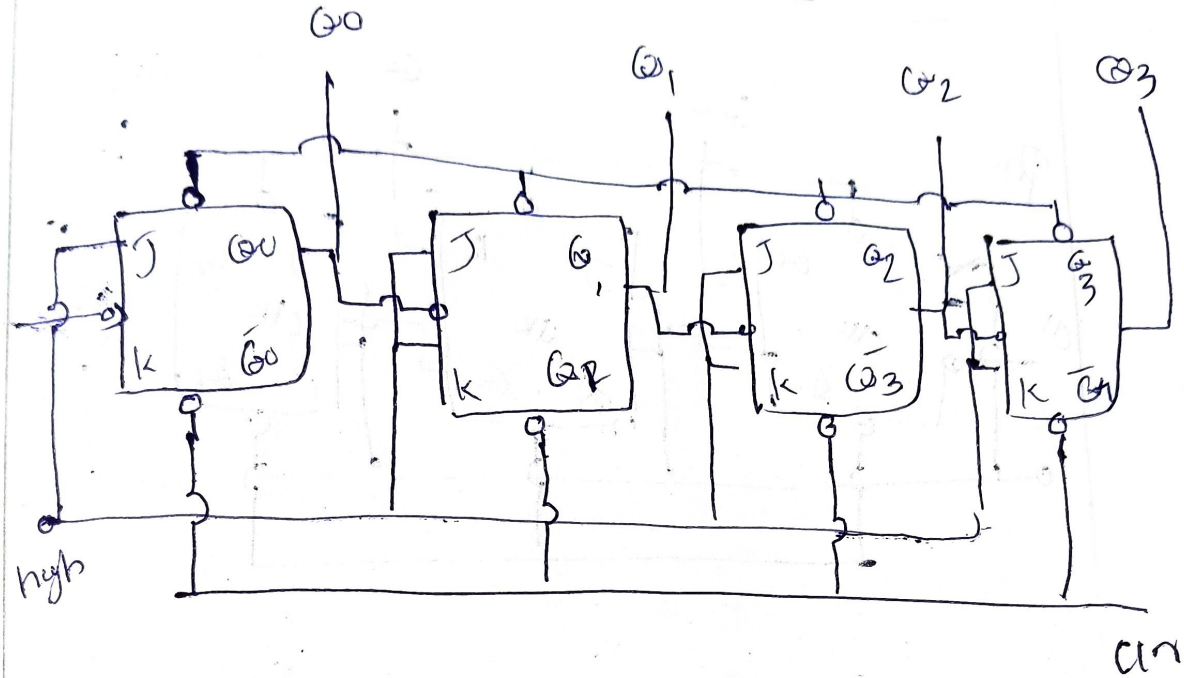
Counters

* Counters is a sequential circuit where output states progress in a repeated pattern advancing by one state for each ~~pre~~ clock pulse
* If a Counter has n flipflops then it ~~is~~ may count clock pulses up to $2^n - 1$
* modulus of a counter means number of states of the counter.

* Counters uses a chain of flipflops.
* Counters are classified into
1) Synchronous counters
2) Asynchronous counters (Ripple)

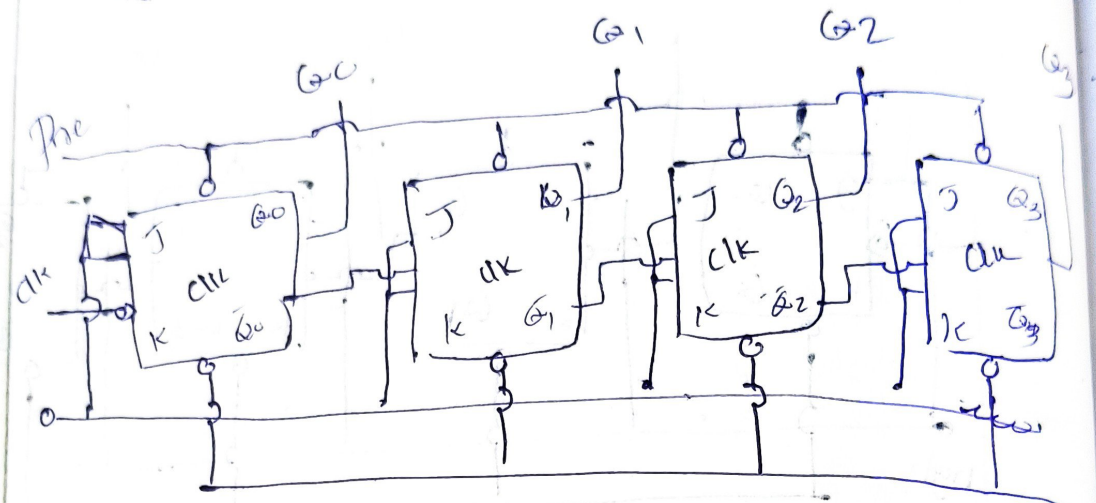
Synchronous Counters	Asynchronous Counters
* Common clock is connected to all the flipflops	* ~~In CS~~ first flipflop is ~~to~~ clocked by external clock pulse and ~~others~~ remaining flipflops are clocked from previous flip flops output.
* Here design is complex	* design is simple
* operation is fast	* Operation is slow compared to synchronous
* Costly high	* cost is less compared to synchronous

4 bit Asynchronous Up Counter

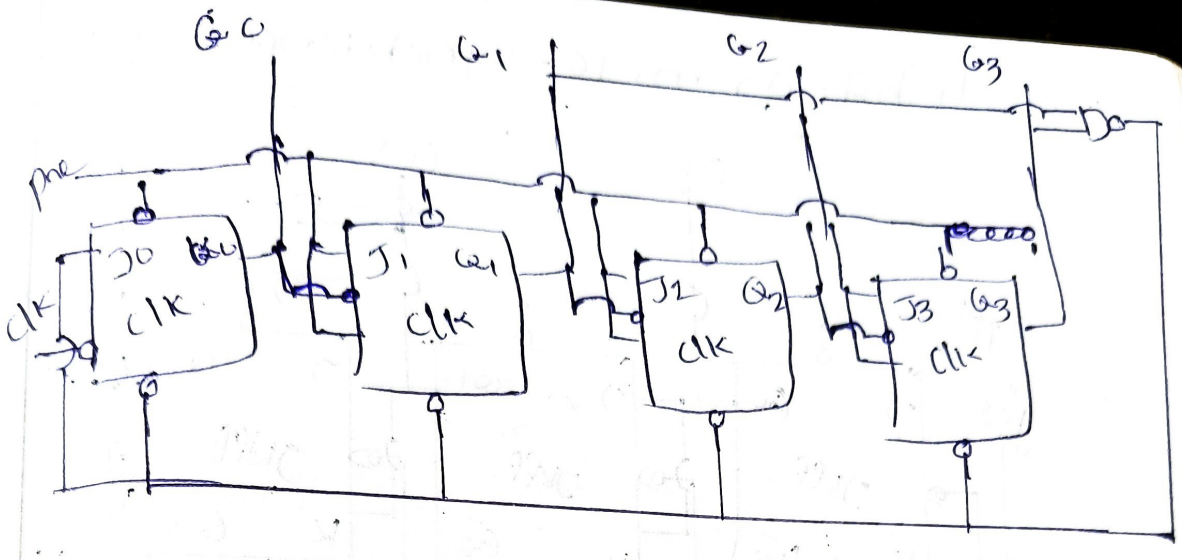


Clk	Q_0	Q_1	Q_2	Q_3
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

4 bit Asynchronous down Counter

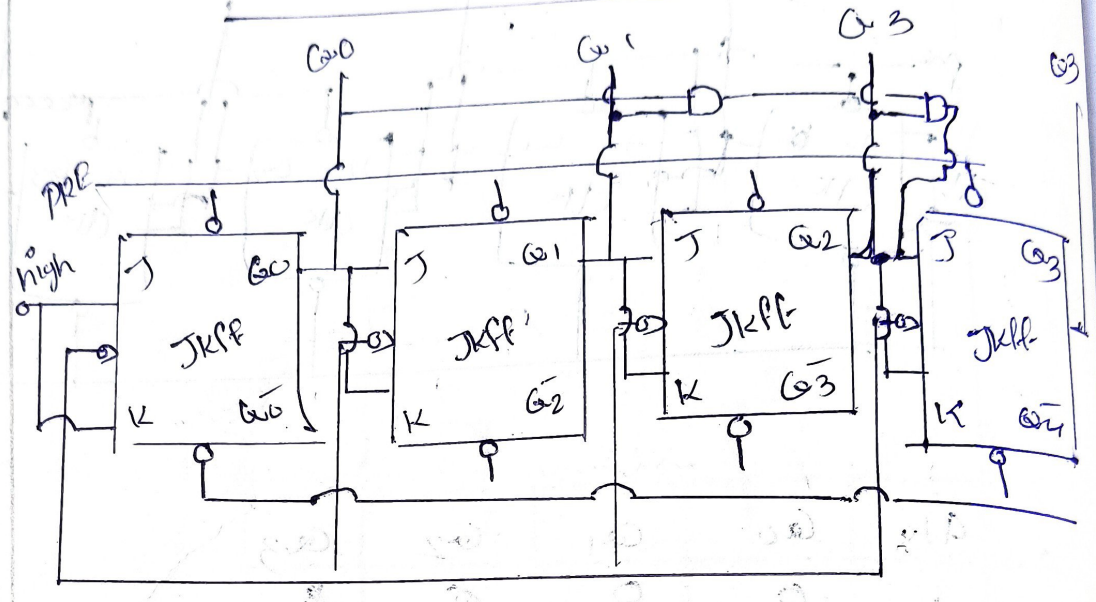


CLK	Q_0	Q_1	Q_2	Q_3
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0



clk	Q_0	Q_1	Q_2	Q_3
1	0	0	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	1	0
4	0	1	0	0
5	0	1	0	0 1
6	0	1	1	0
7	0 1	1	1	0 1
8	1	0	0	0
9	1	0	0	0 1
10	0	0	0	0

4 bit or mod-16 - Synchronous Upcounter



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

ounter

Q3

D

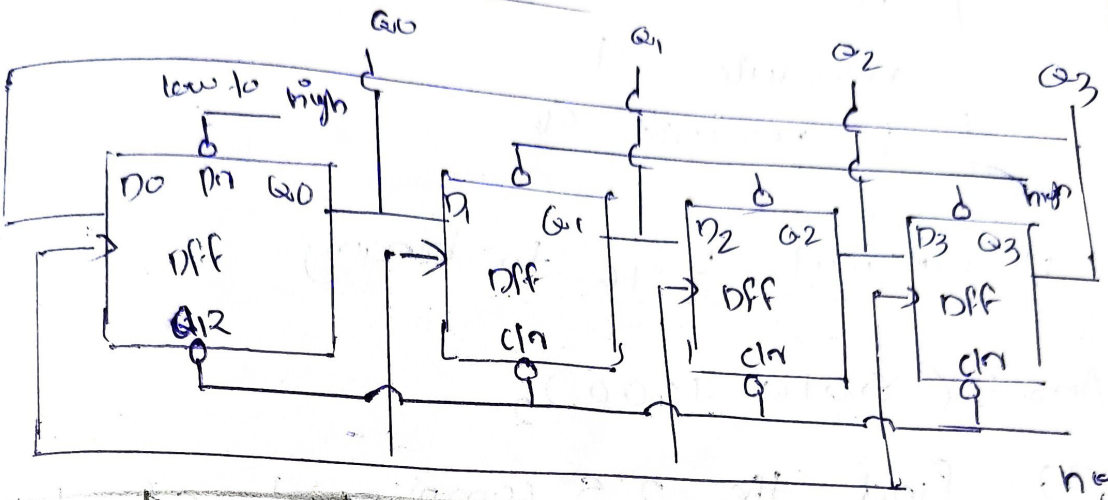
Q3

JK ff

Q3̄

Q

ring Counter



clr	Q3	Q2	Q1	Q0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0