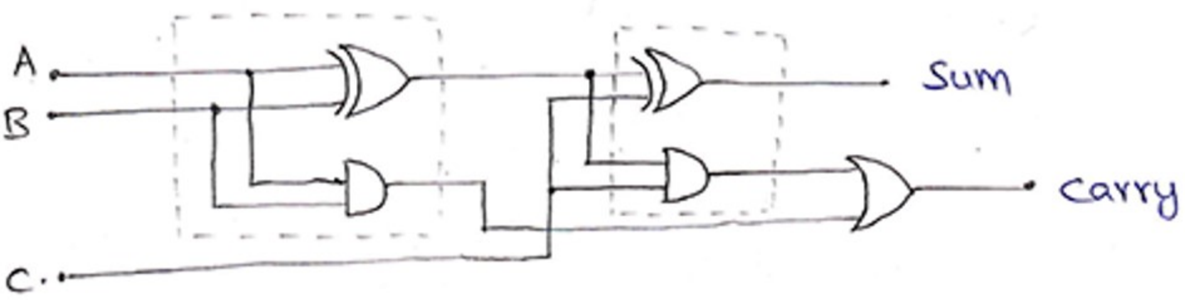
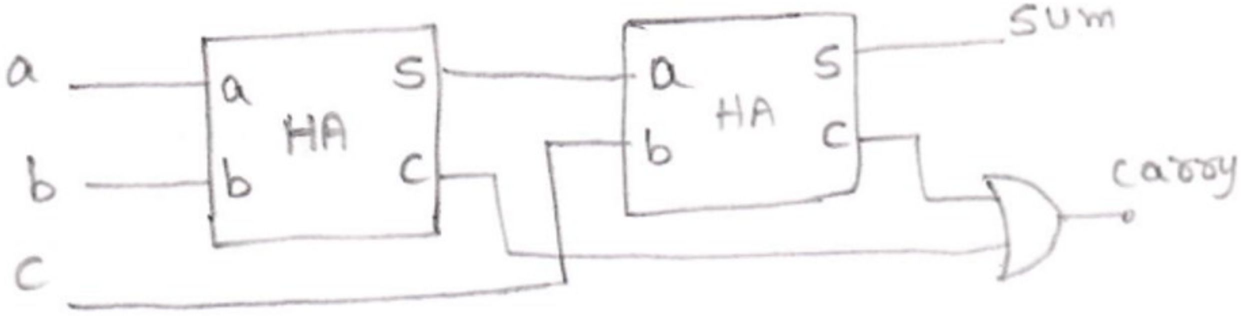
1) Design and implement the Full adder by using the Half adder

Full Adder using Half Adder
A Full Adder can also be implemented using two half adders and one OR gate.
The circuit diagram for this can be drawn as,



And, it could be represented in block diagram as,



The Boolean expression for Sum and Carry is as,

$$\begin{aligned}
\text{Sum} &= A \oplus B \oplus C \\
\text{Carry} &= AB + (A \oplus B). \qquad = AB + (\ A.\ B + A.\ B).\ C \\
&= AB + A.\ BC + A.\ B.\ C \\
&= B\ (A + A.\ C) + A.\ B.\ C \ \ = B\ [(A+ A)\ (A + C)] + A.\ B.\ C \\
&= AB + AC + A.\ B.\ C \\
&= AB + C\ (B + A.\ B) \\
&= AB + C\ [(B + A)\ (B + B)] \quad = AB + BC + AC
\end{aligned}$$

2) Implement Excess-3 code

Step 1

We have to convert the numbers (which are to be added) into excess 3 forms by adding 0011 with each of the four bit groups them or simply increasing them by 3.

Step 2

Now the two numbers are added using the basic laws of binary addition, there is no exception for this method.

Step 3

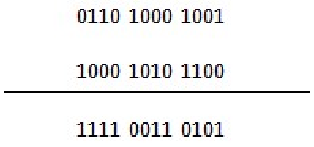
Now which of the four groups have produced a carry we have to add 0011 with them and subtract 0011 from the groups which have not produced a carry during the addition.

Step 4

The result which we have obtained after this operation is in excess 3 forms and this is our desired result

Example

To understand the **excess 3 code addition** method better we can observe the method with the help of an example, let us take two numbers which we will to add. 0011 0101 0110 and 0101 0111 1001 are the two binary numbers. Now following the first step we take the excess 3 form of these two numbers which are 0110 1000 1001 and 1000 1010 1100, now these numbers are added following the basic rules of addition

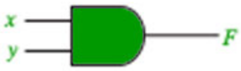
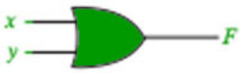
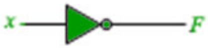
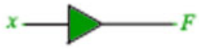
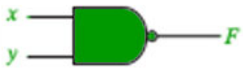
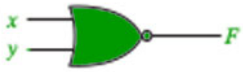
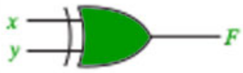
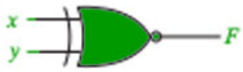


```
0110 1000 1001
1000 1010 1100
--------------
1111 0011 0101
```

now adding 0011 to the groups which produces a carry and subtracting zero from the groups which did not produced carry we get the result as 1100 0110 1000 is the result of the addition in excess 3 code and the BCD answer is 1001 0011 0101.

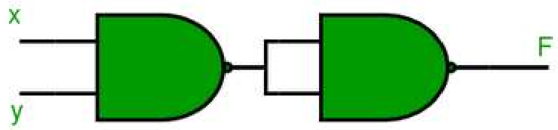
3) Design logic gates using Universal gates(NAND and NOR)

In Boolean Algebra, the **NAND** and **NOR** gates are called **universal gates** because any digital circuit can be implemented by using any one of these two i.e. any logic gate can be created using NAND or NOR gates only.
Every logic gate has a representation symbol. The below image shows a graphical representation of all logic gates.

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y F 0 0 0 0 1 0 1 0 0 1 1 1
OR		$F = x + y$	x y F 0 0 0 0 1 1 1 0 1 1 1 1
Inverter		$F = x'$	x F 0 1 1 0
Buffer		$F = x$	x F 0 0 1 1
NAND		$F = (xy)'$	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$F = (x + y)'$	x y F 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y F 0 0 0 0 1 1 1 0 1 1 1 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y F 0 0 1 0 1 0 1 0 0 1 1 1

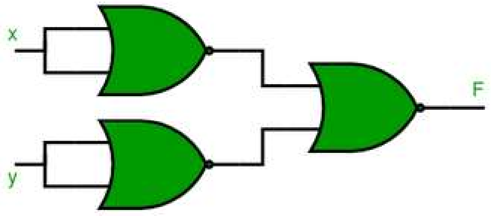
1. Implementation of AND Gate using Universal gates.

a) Using NAND Gates
The AND gate can be implemented by using two NAND gates in the below fashion:



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

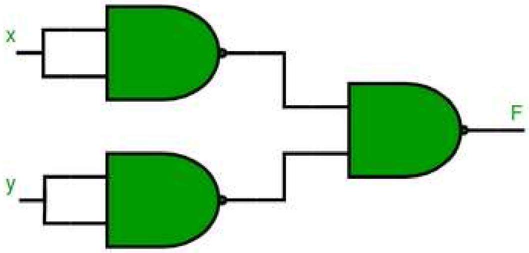
b) Using NOR Gates
Implementation of AND gate using only NOR gates as shown below:

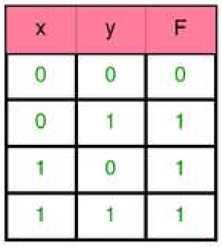


x	y	F
0	0	0
1	0	0
1	1	1
0	1	0

2. Implementation of OR Gate using Universal gates.

a) Using NAND Gates
The OR gate can be implemented using the NAND gate as below:

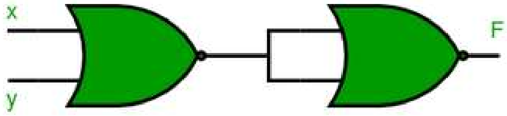


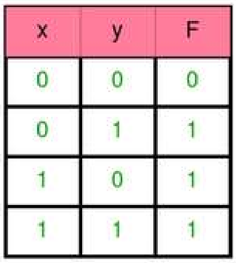


x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

b) Using NOR Gates

Implementation of OR gate using two NOR gates as shown in the picture below:

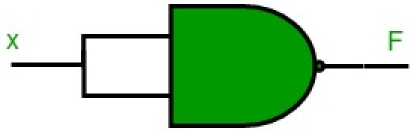


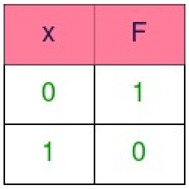


x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

3. Implementation of NOT Gate using Universal gates.

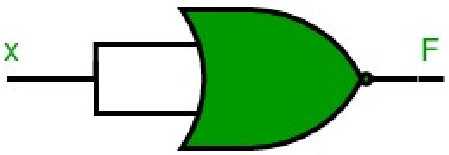
a) Using NAND Gates

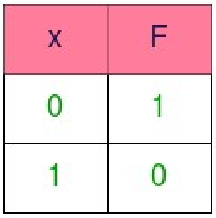




x	F
0	1
1	0

b) Using NOR Gates

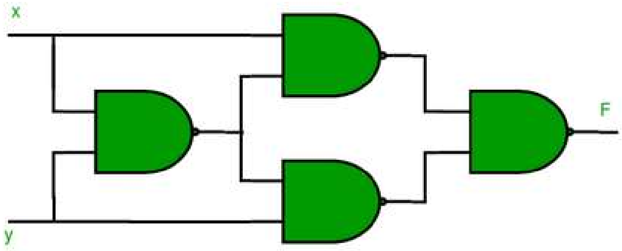


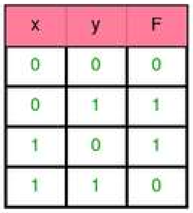


x	F
0	1
1	0

4. Implementation of XOR Gate using Universal gates.

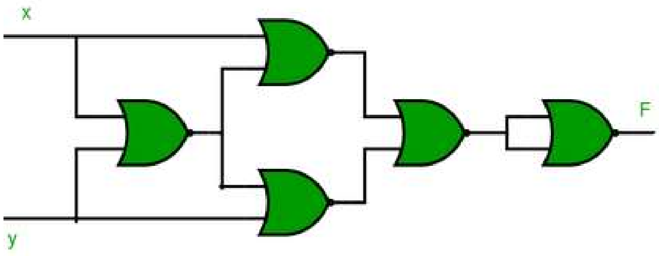
a) Using NAND Gates

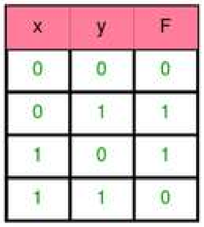




x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

b) Using NOR Gates

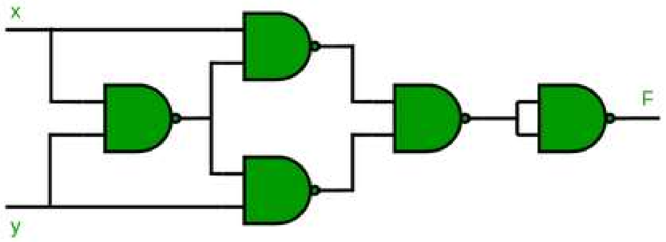


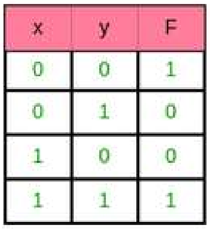


x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

5. Implementation of XNOR Gate using Universal gates.

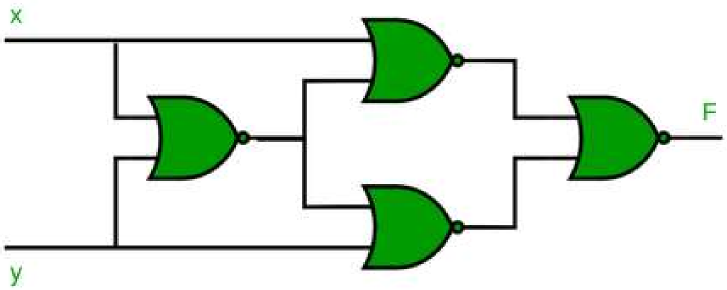
a) Using NAND Gate

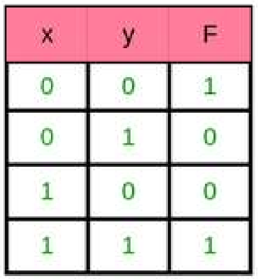




x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

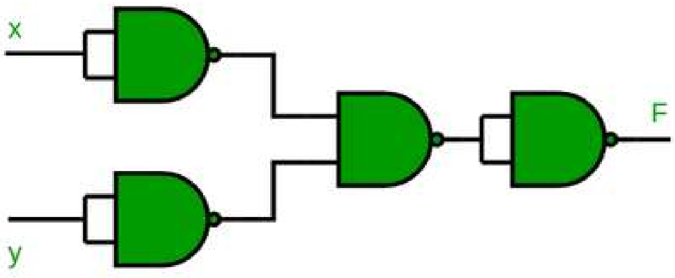
b) Using NOR Gate

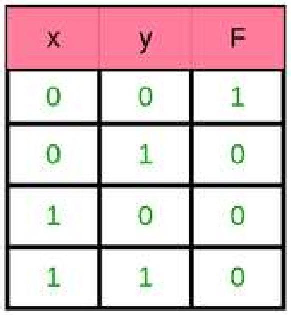




x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

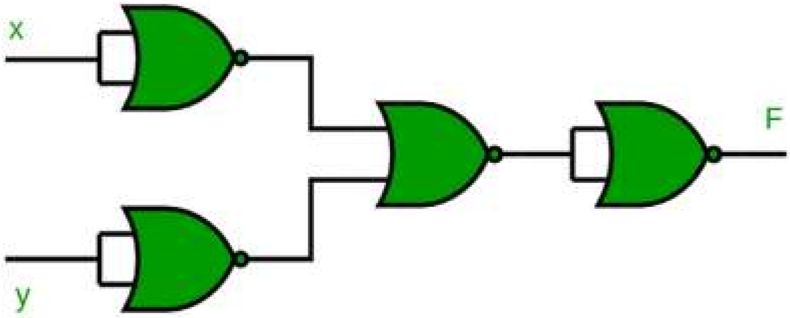
6. Implementation of NOR Gate using NAND Gates

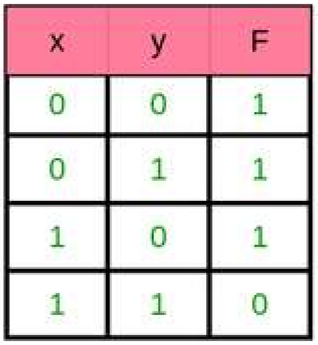




x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

7. Implementation of NAND Gate using NOR Gates





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

4) Design a 3 input circuit which tells whether input is odd or even.(Hint: take 3 inputs, 2 outputs)

A	B	C	Y1=even	Y2=odd
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

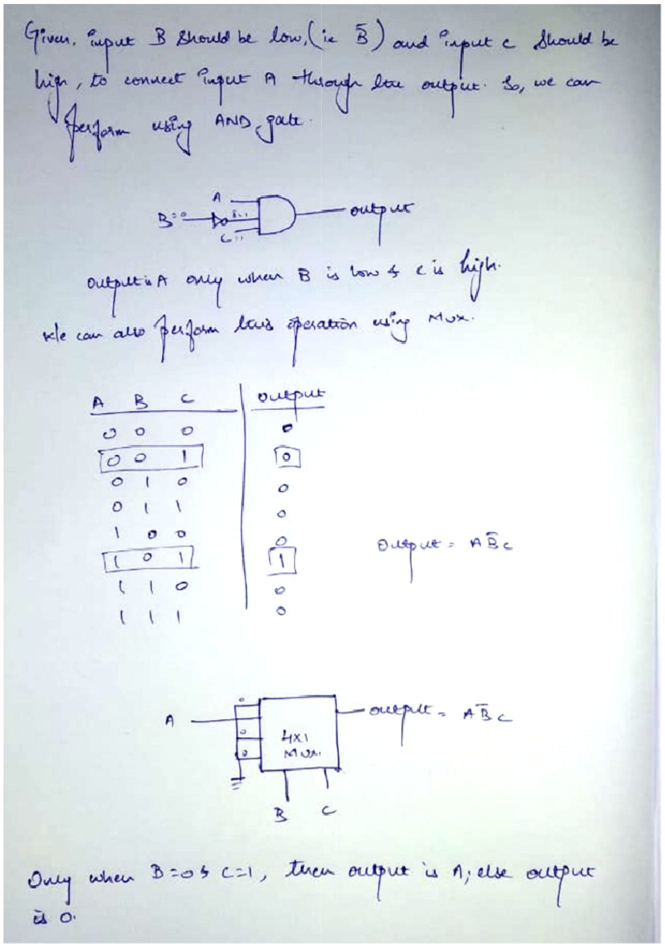
Y1 = C’

Y2 = C

5) Using XOR gate prepare OR gate.

XOR is not Universal gate. We cannot do it.

6) Design a logic circuit that allow input signal a that connect through to the output only when control input b is low only and control input c is high otherwise output should be low.



Given, input B should be low, (ie $\bar{B}$) and input c should be high, to connect input A through the output. So, we can perform using AND gate.

A
B=0
B=1
C=1
output

output is A only when B is low & c is high.

We can also perform this operation using Mux.

A	B	C	output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Output = $A\bar{B}C$

A
0
0
0
4x1
MUX
B
C

output = $A\bar{B}C$

Only when B=0 & C=1, then output is A; else output is 0.

7) Design a combinational circuit that takes 4bit input x, if x is even output is x+1 else output=x-1.

As per above question conditions the truth table is look like as follows:

A	B	C	D	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0

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

Here if we take K-map and take equations for outputs Y0, Y1, Y2, Y3 we get as mentioned below.

⇨ $Y0 = \bar{D}$;
⇨ $Y1 = C$;
⇨ $Y2 = B$;
⇨ $Y3 = A$;