1. a) Design and construct a Schmitt trigger using Op-Amp for given UTP and LTP values and demonstrate its working.

Aim: To design and construct a Schmitt trigger using Op-Amp for given UTP and LTP values and demonstrate its working.

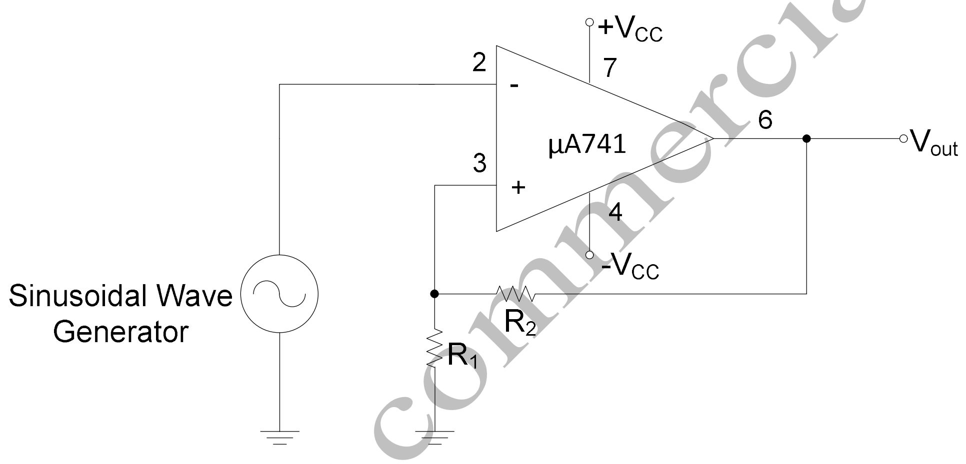
Equipments and Components required:

Sl. No.	Equipments and Components	Quantity
1	Functional generator	1
2	Regulated Power Supply	2
3	CRO	1
4	IC 741	1
5	Resistors: As per design of the circuit	4

Design:

Case1: When given UTP and LTP have equal magnitude but opposite in sign

Circuit diagram:



We have, from the above circuit diagram,

$$UTP = \frac{R_1}{R_1 + R_2}.V_{sat} \text{ ----(1)}$$

$$LTP = -\frac{R_1}{R_1 + R_2}.V_{sat} \text{ ----(2)}$$

Assume that given UTP=4V and LTP=-4V. Also assume $V_{sat} = 10V$

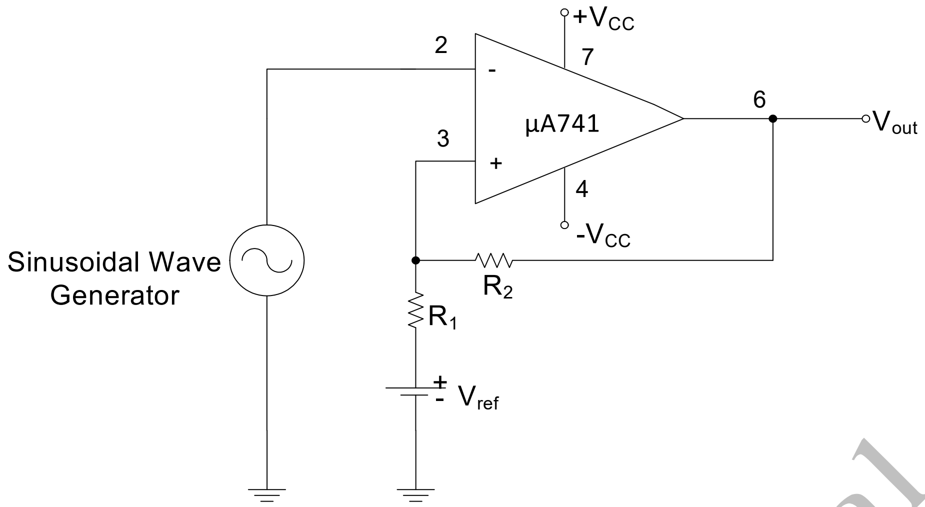
Then, From (1), $UTP = \frac{R_1}{R_1 + R_2}.V_{sat} \Rightarrow 4 = \frac{R_1}{R_1 + R_2}.10 \Rightarrow 3R_1 = 2R_2 --(3)$

We have to choose the values of R_1 and R_2 so as to satisfy the condition of (3).

We can choose $R_1 = 2K\Omega$ and $R_2 = 3K\Omega$

Case2: When given UTP and LTP are not of equal magnitude

Circuit Diagram:



We have, from the above circuit diagram,

$$\text{UTP}=\frac{R_2}{R_1+R_2}.V_{ref}+\frac{R_1}{R_1+R_2}.V_{sat} \text{ -------(1)}$$

$$\text{LTP}=\frac{R_2}{R_1+R_2}.V_{ref}-\frac{R_1}{R_1+R_2}.V_{sat} \text{ --------(2)}$$

From (1) and (2), UTP + LTP= $\frac{2R_2}{R_1+R_2}.V_{ref}$ -----(3) and UTP - LTP= $\frac{2R_1}{R_1+R_2}.V_{sat}$ ------(4)

Now, Suppose given UTP=4V and LTP=2V

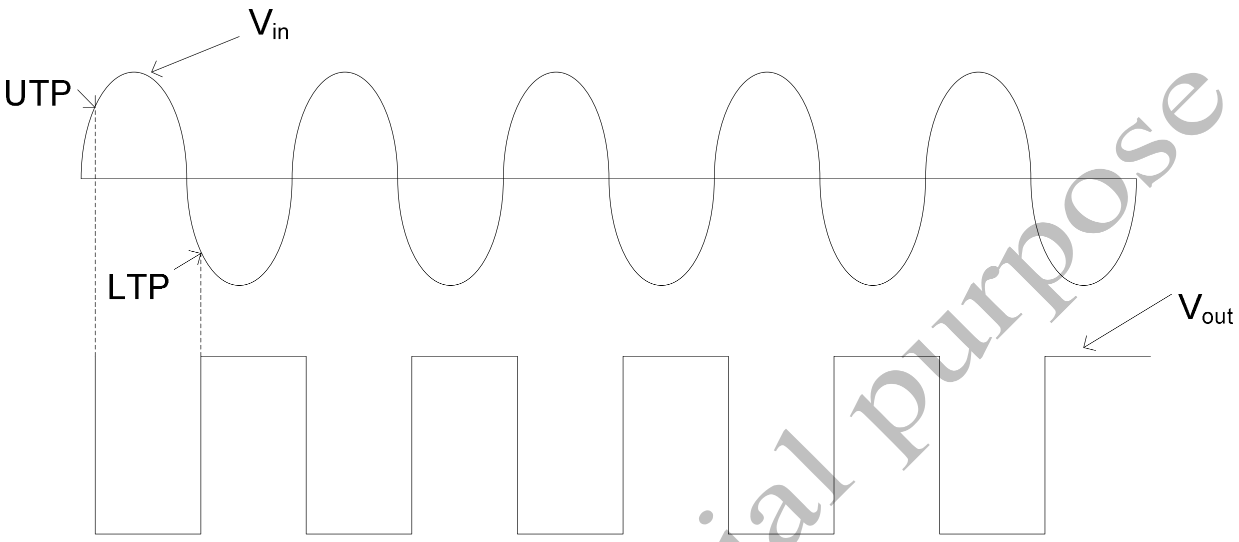
Then, From (4), 2= $\frac{2R_1}{R_1+R_2}.10 \Rightarrow R_2 = 9R_1$-----(5) [$V_{sat}$ is taken as 10V]

We can choose $R_1 = 1K\Omega$ and $R_2 = 9K\Omega$

Again, from (3), 6= $\frac{2R_2}{R_1+R_2}.V_{ref} \Rightarrow 6 = \frac{2\times 9000}{1000+9000}.V_{ref} \Rightarrow V_{ref} = 3.3V$----(6)

Procedure:
(i) Make the connection as per the circuit diagram and the designed value of resistors.
(ii) Apply peak to peak of 14V sinusoidal waveform at 50Hz
(iii) Observe the Input and Output waveforms at CRO.

Input and output waveforms:



b) Design and implement a Schmitt trigger using Op-Amp using a simulation package for two sets of UTP and LTP values and demonstrate its working.

Use PSPICE to draw the circuits as shown for 1(a) in Case (1) and Case (2). Use the same values of all the parameters used in 1(a). Simulate the circuit. The output waveforms should be same as in 1(a) for applied same sinusoidal input signal.

2. a) Design and construct a rectangular waveform generator (Op-Amp relaxation oscillator) for given frequency and demonstrate its working.

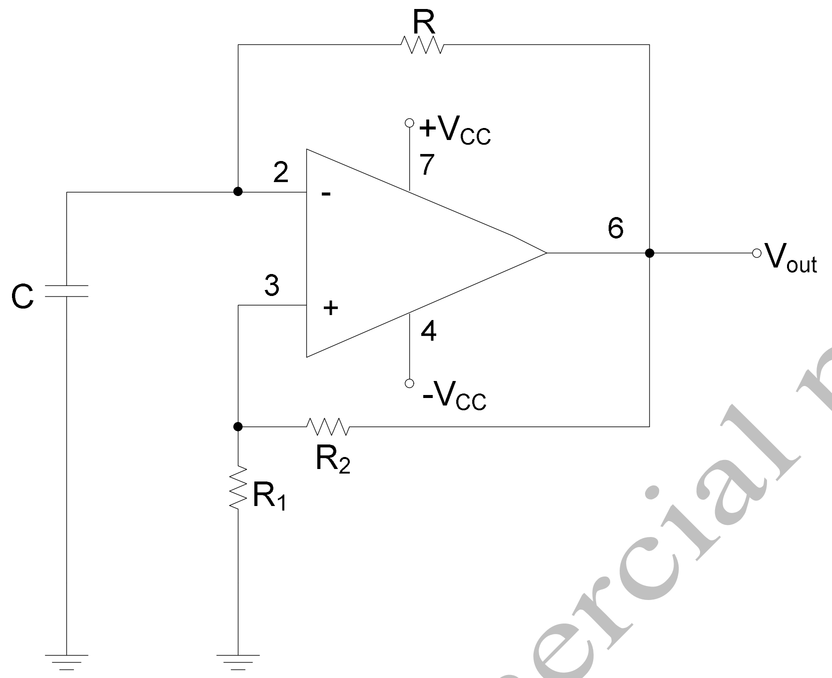
Aim: To design and construct a rectangular waveform generator (Op-Amp relaxation oscillator) for given frequency and demonstrate its working.

Equipment and Component required:

Sl. No.	Equipments and Components	Quantity
1	Regulated Power supply	1
2	CRO	1
3	IC 741	1
	Resistors: As per design	3
5	Capacitor: As per design	1

Design:

Circuit diagram:



The period of the output rectangular waveform is given by $T = 2RC\ln\left(\frac{1+B}{1-B}\right) --- (1)$

Where $B = \frac{R_1}{R_1 + R_2}$

If R_2=1.16R_1, then $B = \frac{1}{2.16}$=0.463. From (1), We have $T = 2RC\ln\left(\frac{1.463}{0.537}\right) \Rightarrow T = 2RC --- (2)$

If the given frequency is 1KHz, then $T = \frac{1}{f} = \frac{1}{1000} = 1ms$

If We take R_1=10KΩ , then We have to take R_2= 11.6KΩ so that equation (2) is satisfied.

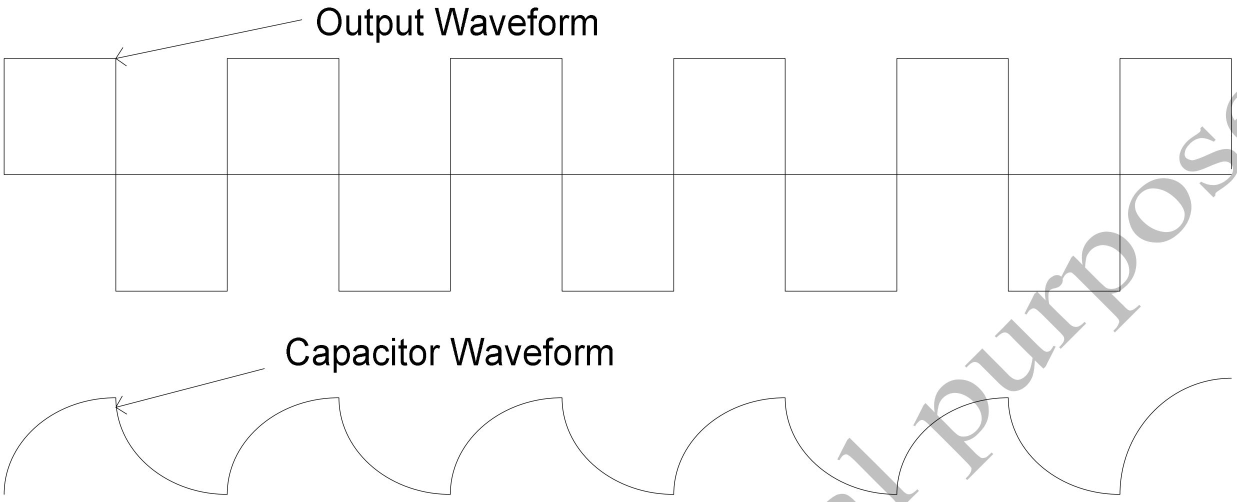
If we take C=0.1μf, then from (2), $R = \frac{T}{2C} = \frac{10^{-3}}{2 \times 0.1 \times 10^{-6}}$ =5 KΩ

Procedure:

(i) Make the connection as per the circuit diagram and the designed value of resistors and capacitor.

(ii) Observe the output waveforms and the capacitor waveform at CRO.

Waveform:



b) Design and implement a rectangular waveform generator (Op-Amp relaxation oscillator) using a simulation package and demonstrate the change in frequency when all resistor values are doubled.

Use PSPICE to draw the circuit above and find the time period, then find the frequency.

3. **Design and implement an Astable multivibrator circuit using 555 timer for a given frequency and duty cycle.**

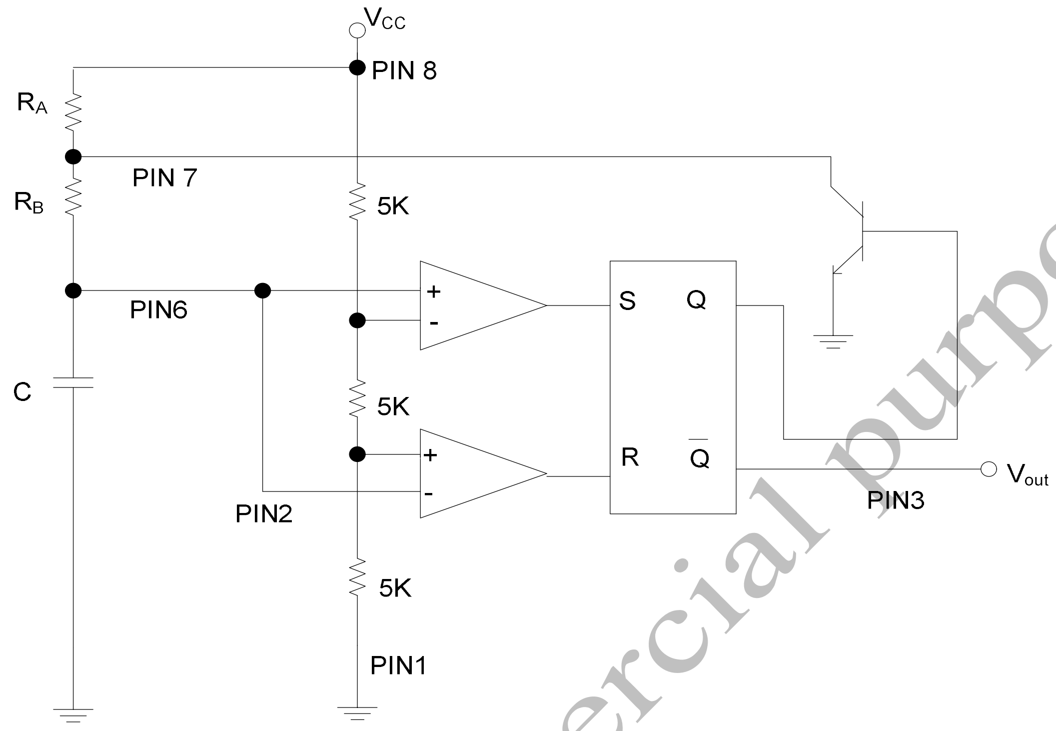
Aim: To Design and implement an Astable multivibrator circuit using 555 timer for a given frequency and duty cycle.

Equipments and Components required:

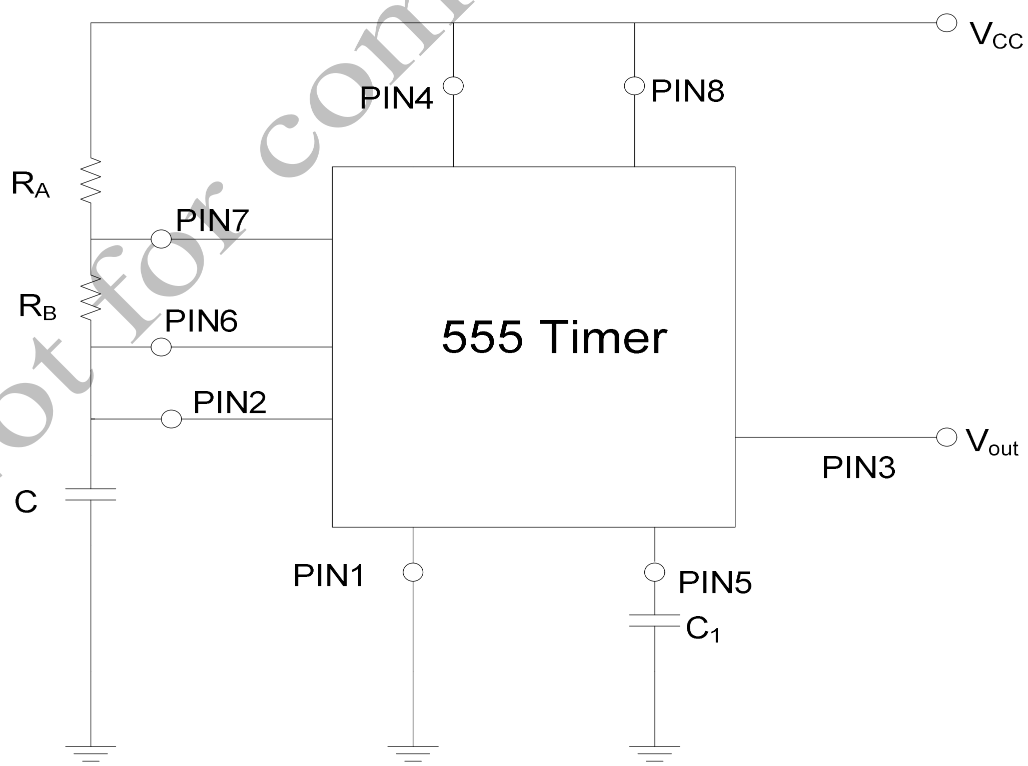
Sl. No.	Equipments and Components	Quantity
1	Regulated Power supply	1
2	CRO	1
3	IC 555	1
4	Resistors: As per design	2
5	Capacitors: As per design	2

Design:

555 Timer Circuit:



Circuit diagram for astable multivibrator using 555 timer:



Take the given frequency (f)=1KHz and Duty cycle=60%

Then, Time Period, $T=\frac{1}{f}=1ms$

$T=t_H+t_L$ where t_H is the duration when output is high and t_L is the duration when the output is low

Given Duty cycle=60% (=0.6), hence t_H=0.6ms and t_L=0.4ms

We have from the theory of astable multivibrator circuit using 555 timer

$t_H=0.693(R_A+R_B)C$ ------(1)

$t_L=0.693R_BC$ -------------(2)

We have t_L=0.4ms and take C=0.1µf. Substituting these values in (2), we get

$$R_B=\frac{t_L}{0.693\times C}=\frac{0.4\times10^{-3}}{0.693\times0.1\times10^{-6}}=5.8K\Omega$$

We have t_H=0.6ms and C=0.1µf. Substituting these values in (1), we get

$$R_A+R_B=\frac{t_H}{0.693\times C}=\frac{0.6\times10^{-3}}{0.693\times0.1\times10^{-6}}=8.7K\Omega$$

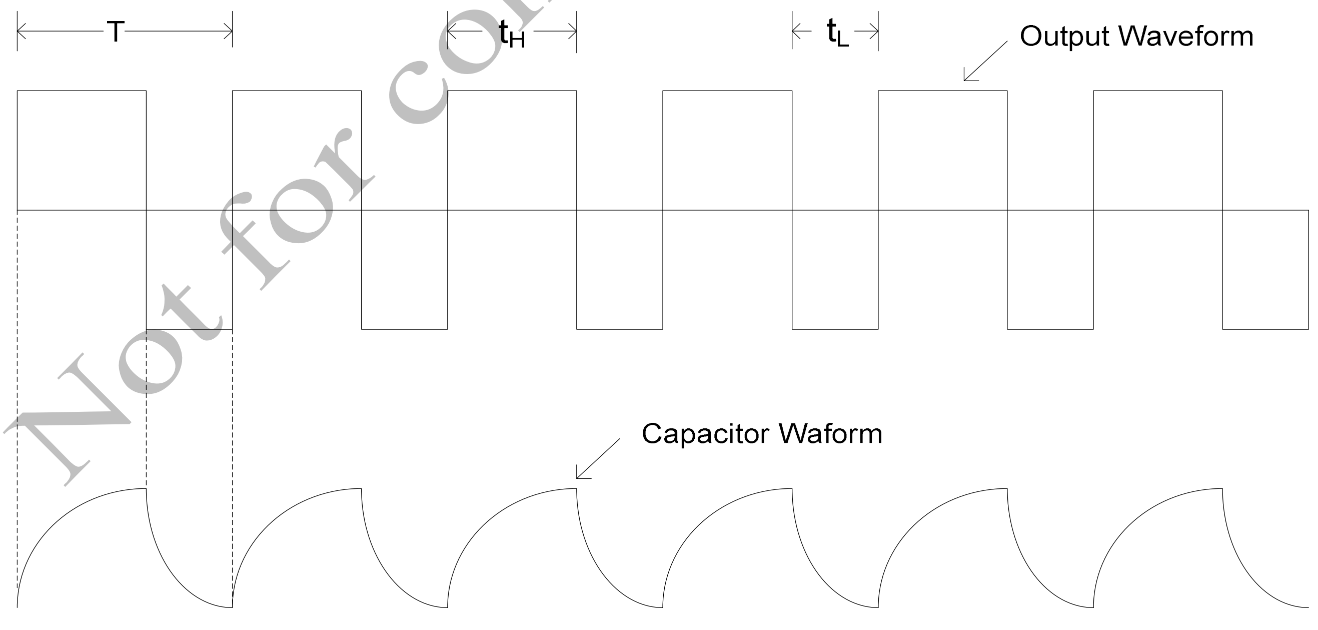
$$R_A=8.7-R_B=8.7-5.8=2.9K\Omega$$

Procedure:

(i) Make the connection as per the circuit diagram and the designed value of resistors and capacitor.

(ii) Observe the output waveforms and the capacitor waveform at CRO.

Waveform:



4. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates.

Aim: To Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates.

Equipment and Components required:

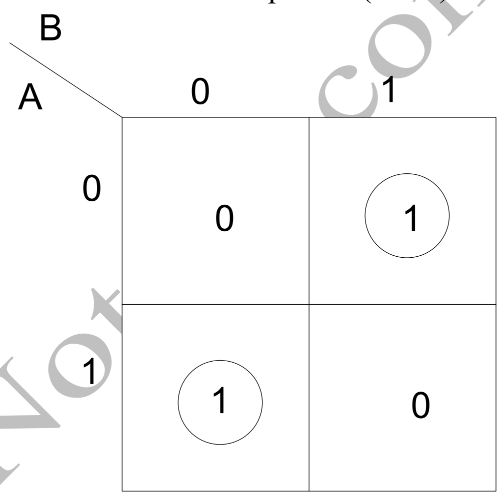
Sl. No.	Equipments and Components	Quantity
1	Trainer Kit	1
3	IC 7404 (NOT Gate)	1
4	IC 7408 (AND Gate)	1
5	IC 7432 (OR Gate)	1
6	IC 7486 (XOR Gate)	1
7	Patch Cord	1 Bunch

Design and Implementation of Half Adder:

Truth Table:

Input		Output	
A	B	S (SUM)	C (CARRY)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-Map for S (SUM):



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

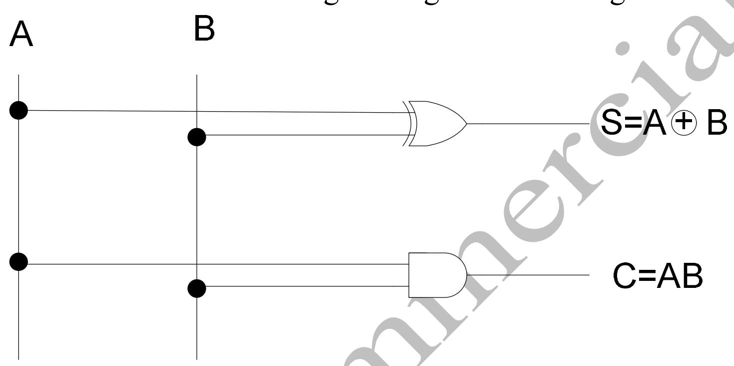
$S = A \oplus B$

K-Map for C (CARRY):

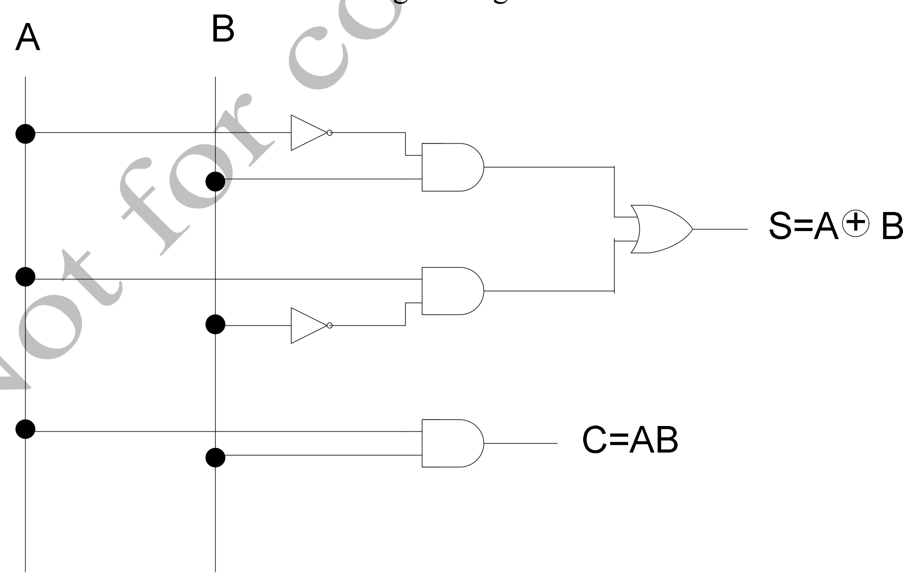
A \ B	0	1
0	0	0
1	0	1

C=AB

Realization of Half Adder using Basic gate and XOR gates:



Realization of Half Adder using Basic gates:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

Design and Implementation of Full Adder:

Truth Table:

Input			Output	
A	B	C	S(SUM)	C(CARRY)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K Map for S (Sum):

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

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

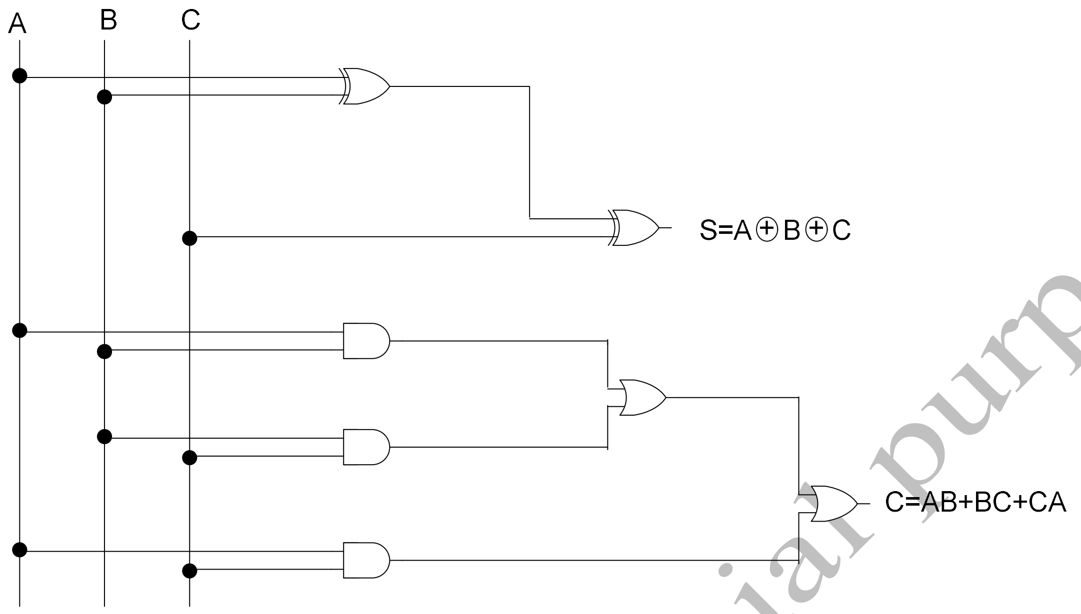
$S=A\oplus B\oplus C$

K Map for C (CARRY):

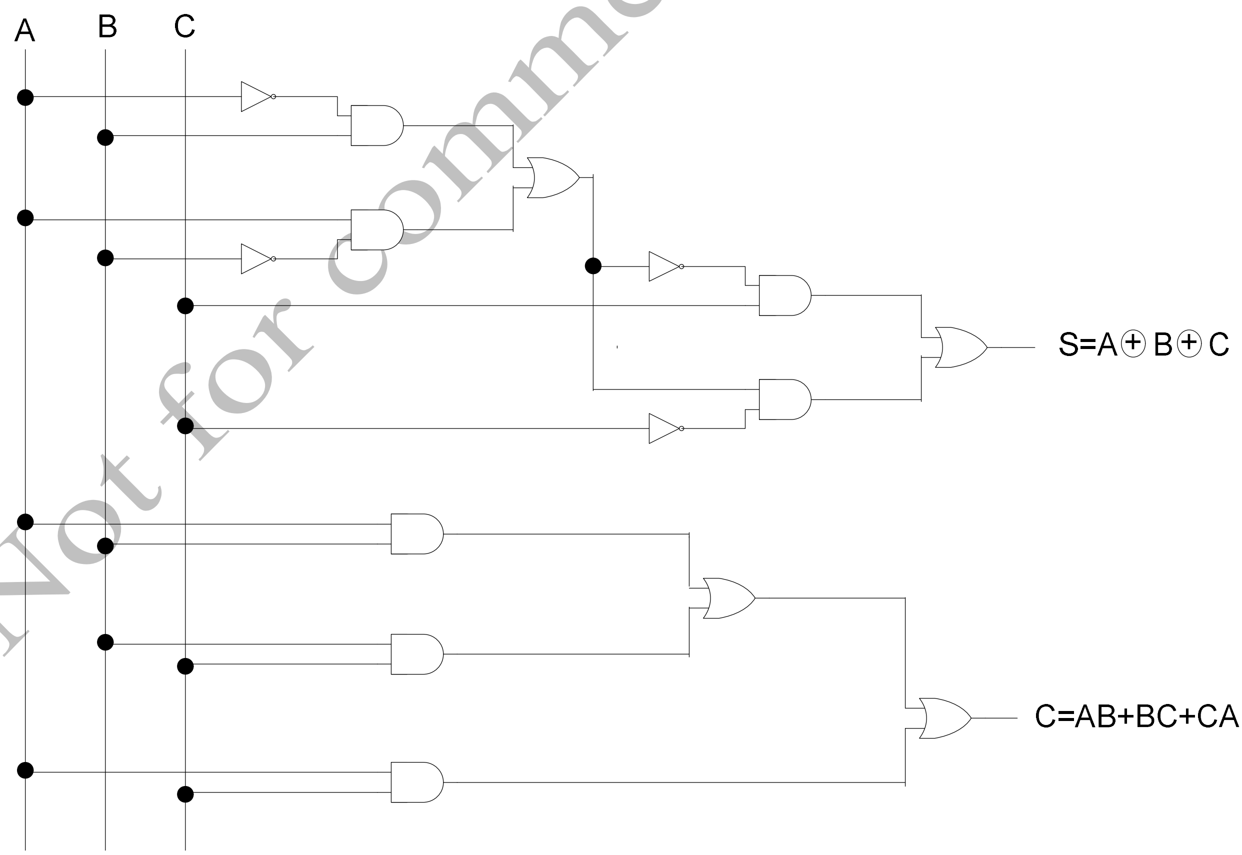
A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C=AB+BC+AC$

Realization of full adder using XOR and Basic Gates:



Realization of full adder using Basic Gates:



Procedure:

(i) Connect the circuit as per design.

(ii) Apply the input combination and verify the truth table.

Design and Implementation of Half Subtractor:

Truth Table:

Input		Output	
A	B	Di(Difference)	Bo(Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K -Map for Di (Difference):

A \ B	0	1
0	0	1
1	1	0

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

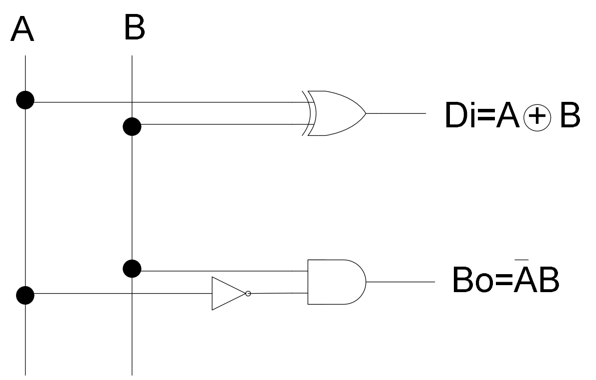
$Di = A \oplus B$

K-Map for Bo (Borrow):

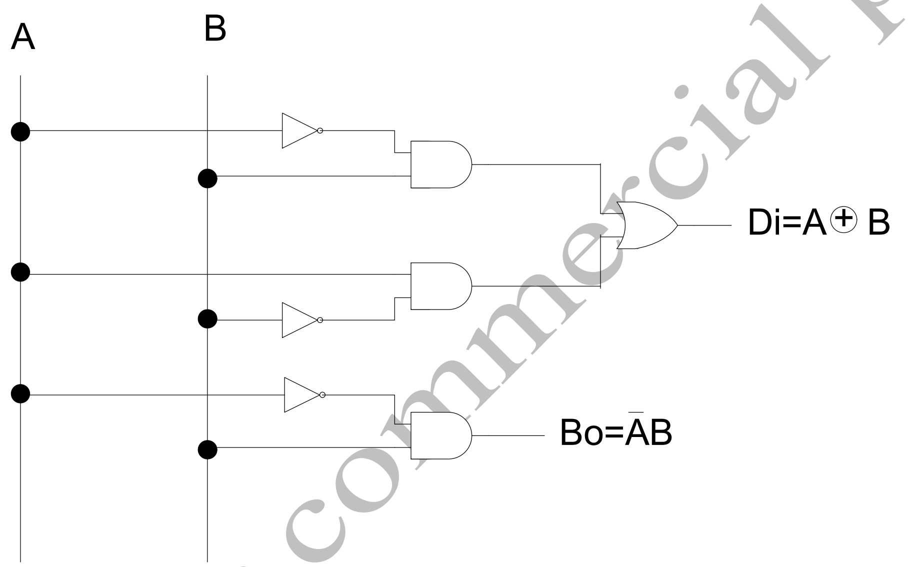
A \ B	0	1
0	0	1
1	0	0

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

Realization of Half Subtractor using Basic gates and XOR gate:



Realization of Half Subtractor using Basic gates:



Design and Implementation of Full Subtractor:

Truth table:

Inputs			Outputs	
A	B	C	Di	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Map for Di (Difference):

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

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

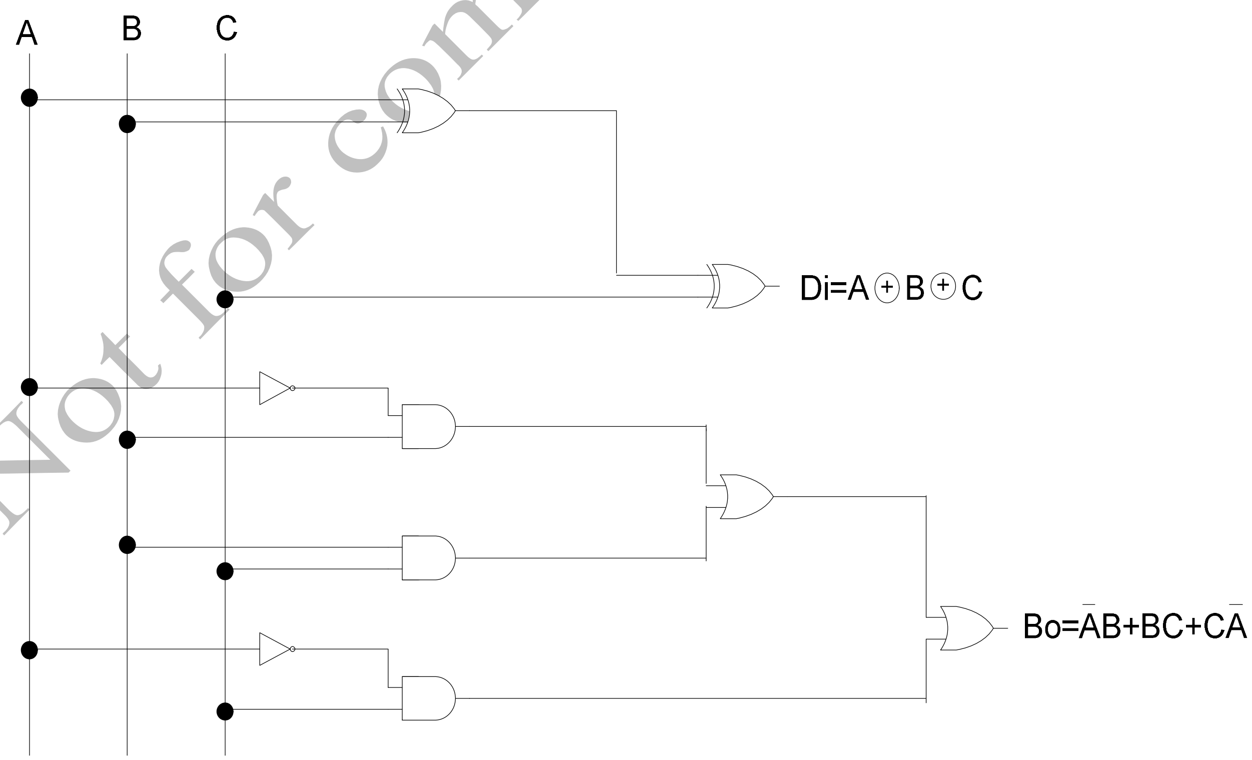
$Di=A\oplus B\oplus C$

K-map for Bo (Borrow):

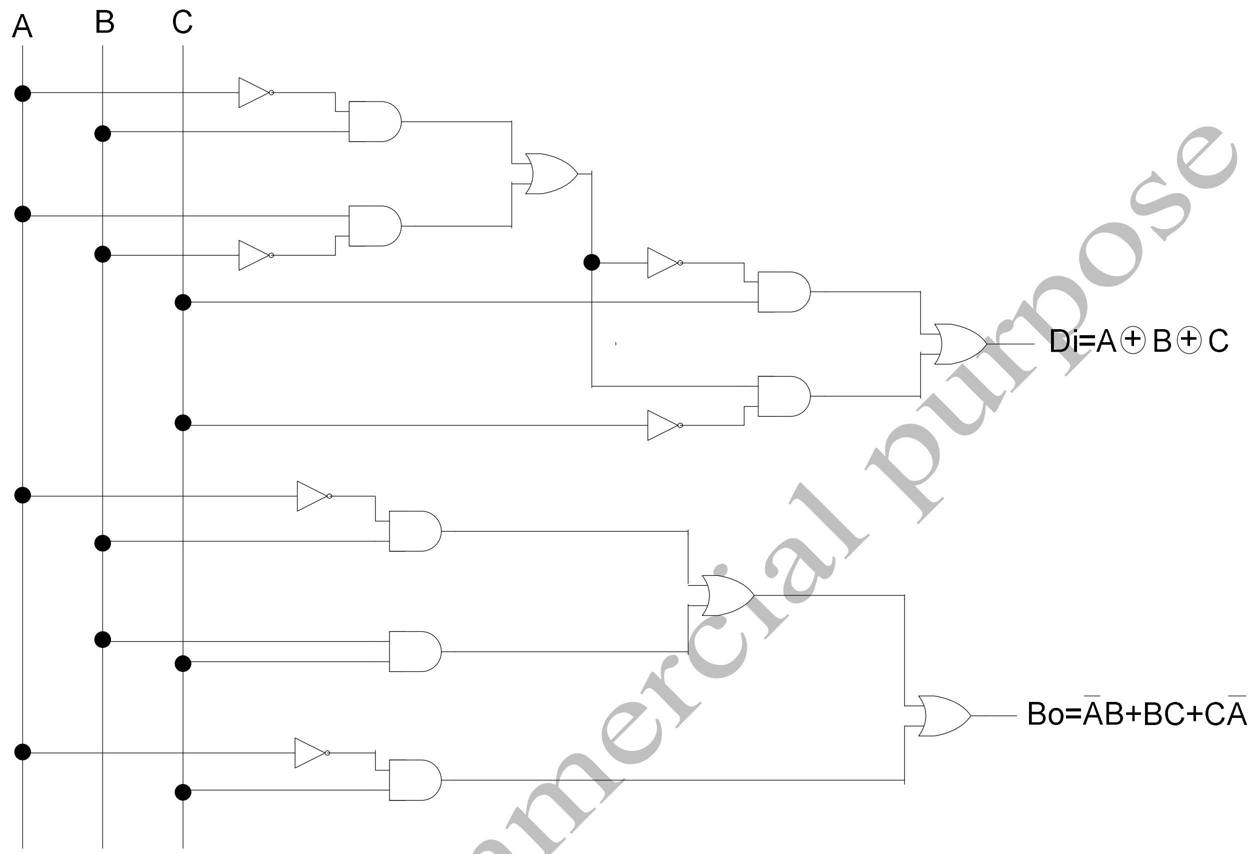
A \ BC	00	01	11	10
0	0	1	1	1
1	0	0	1	0

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

Realization of full subtractor using Basic gates and XOR gates:



Realization of full subtractor using Basic gates:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

5. a)Given a 4-variable logic expression, simplify it using Entered Variable Map and realize the simplified logic expression using 8:1 multiplexer IC.

Aim: To simplify a given 4-variable logic expression using Entered Variable Map and to realize the simplified logic expression using 8:1 multiplexer IC.

Equipments and Components required:

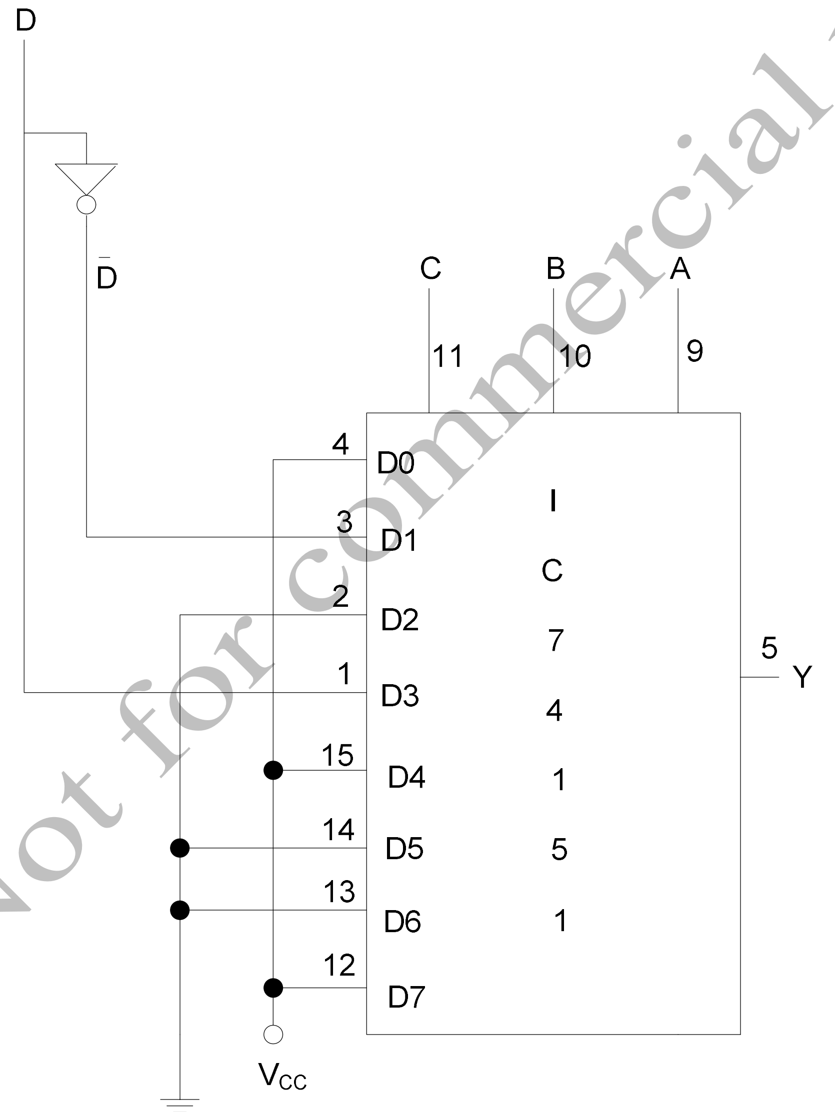
Sl. No.	Equipments and Components	Quantity
1	Trainer Kit	1
2	IC 74151 (8:1 MUX)	1
3	IC 7404 (NOT Gate)	1
4	Patch Cords	1 Bunch

Design:

Given expression: $F(A,B,C,D)=\sum m\,(0,1,2,7,8,9,14,15)$

ABC	000	001	010	011	100	101	110	111
D=0	1	1	0	0	1	0	0	1
D=1	1	0	0	1	1	0	0	1
Y	1	$\overline{D}$	0	D	1	0	0	1
8:1 MUX Data Input	$D_0=1$	$D_1=\overline{D}$	$D_2=0$	$D_3=D$	$D_4=1$	$D_5=0$	$D_6=0$	$D_7=1$

Circuit diagram:



Procedure:

(i) Connect the circuit as per design.

(ii) Apply all combination of input with the 4-input variables and verify the output as per given expression.

b) Design and develop the Verilog /VHDL code for an 8:1 multiplexer. Simulate and verify its working.

Truth table for 8:1 Multiplexer:

Inputs			Output
sel (2)	sel(1)	sel(0)	Y
0	0	0	D(0)
0	0	1	D(1)
0	1	0	D(2)
0	1	1	D(3)
1	0	0	D(4)
1	0	1	D(5)
1	1	0	D(6)
1	1	1	D(7)

VHDL code for 8:1 Multiplexer (Behavioral modeling):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux8to1 is
Port( D:in std_logic_vector(7 downto  0);
      sel: in std_logic_vector(2 downto  0);
      Y:  out std_logic);
end mux8to1;
architectural Behavioral of mux8to1 is
begin
Y<=D(0) when sel=”000” else
   D(1) when sel=”001” else
   D(2) when sel=”010” else
   D(3) when sel=”011” else
   D(4) when sel=”100” else
   D(5) when sel=”101” else
   D(6) when sel=”110” else
   D(7);
 end Behavioral;
```

6. a) Design and implement code converter I)Binary to Gray II) Gray to Binary Code using basic gates.

(I) Aim: To Convert Binary to Gray code

Equipments and components required:

Sl. No.	Equipments and Components	Quantity
1	Trainer Kit	1
2	IC 7486 (XOR Gate)	1
4	IC 7408 (AND Gate)	2
5	IC 7432 (OR Gate)	1
6	IC 7404 (NOT Gate)	2
7	Patch cords	1 bunch

Truth table:

Inputs				Outputs			
B_1	B_2	B_3	B_4	G_1	G_2	G_3	G_4
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

K-Map for G_1:

B_1B_2 \ B_3B_4	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_1=B_1$

K-Map for G_2:

B_1B_2 \ B_3B_4	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2=\bar{B}_1B_2+\bar{B}_1B_2$

$G_2=B_1 \oplus B_2$

K-Map for G_3:

$B_1B_2 \backslash B_3B_4$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$G_3=\bar{B}_2B_3+B_2\bar{B}_3$

$G_3=B_2 \oplus B_3$

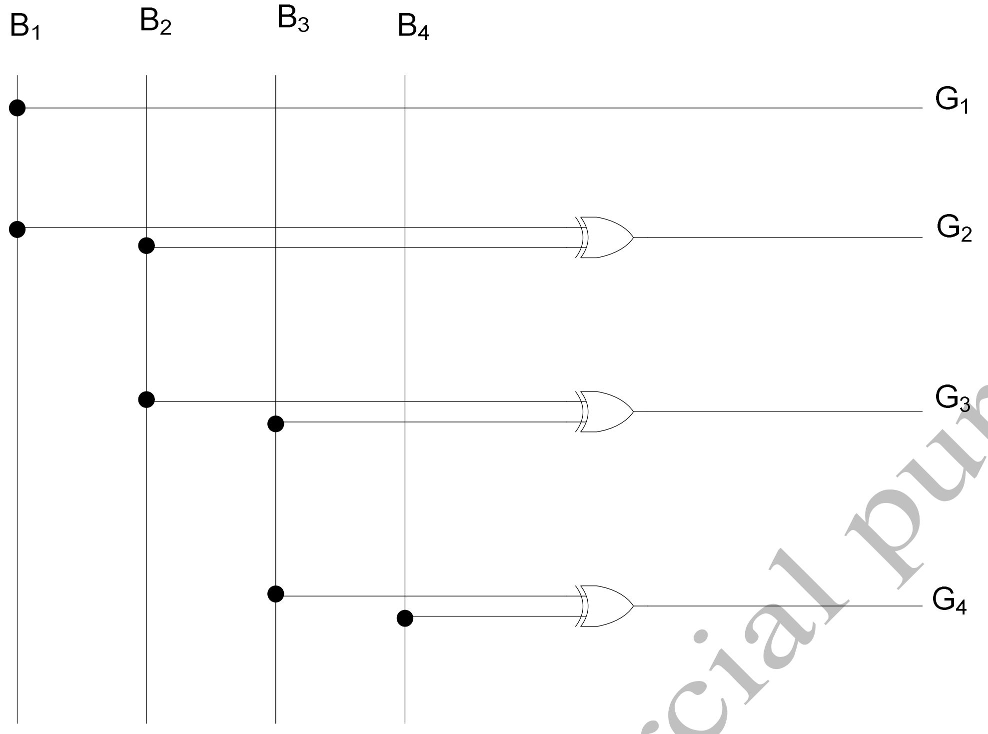
K-Map for G_4:

$B_1B_2 \backslash B_3B_4$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

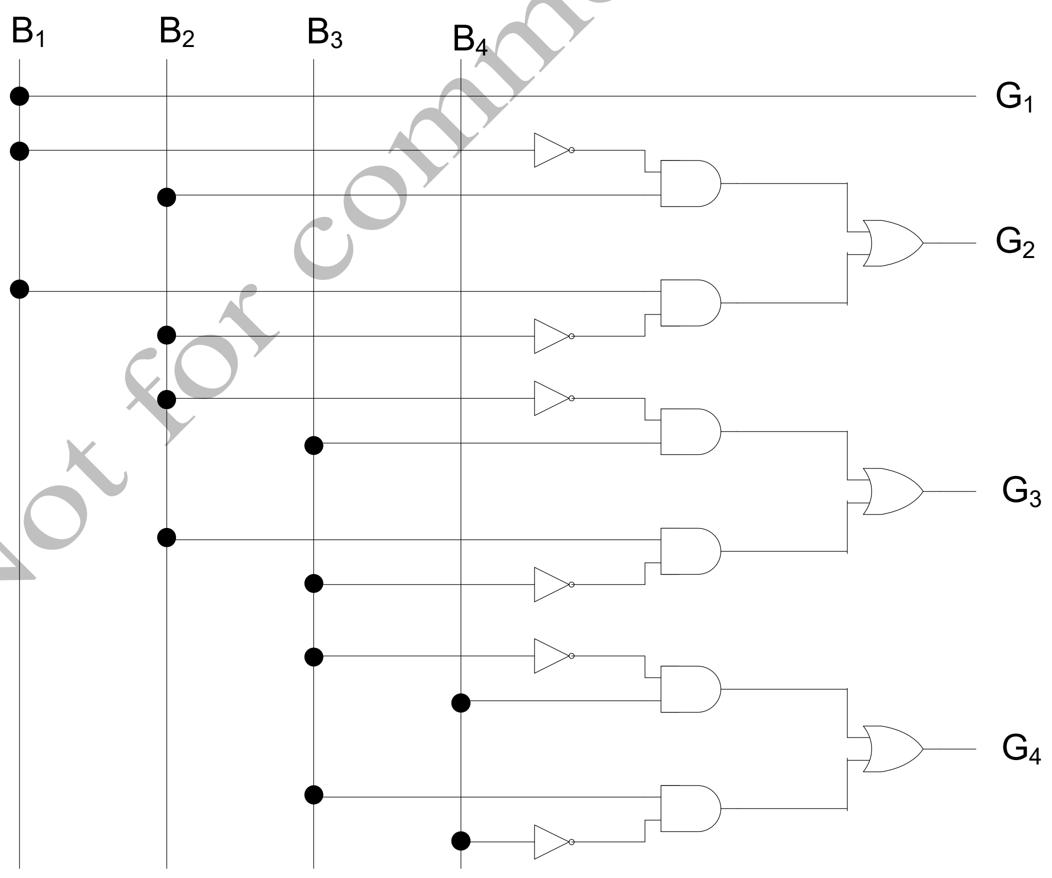
$G_4=\bar{B}_3B_4+B_3\bar{B}_4$

$G_4=B_3 \oplus B_4$

Realization of Binary to Gray code conversion using XOR Gates:



Realization of Binary to Gray code conversion using Basic Gates:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

(II) Aim: To convert Gray code to Binary:

Truth Table:

Inputs				Outputs			
G_1	G_2	G_3	G_4	B_1	B_2	B_3	B_4
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	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

K-Map for B_1:

G_1G_2 \ G_3G_4	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$B_1 = G_1$

K-Map for B_2:

G_1G_2 \ G_3G_4	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$B_2=\bar{G}_1G_2+G_1\bar{G}_2$

$B_2=G_1 \oplus G_2$

K-map for B_3:

G_1G_2 \ G_3G_4	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

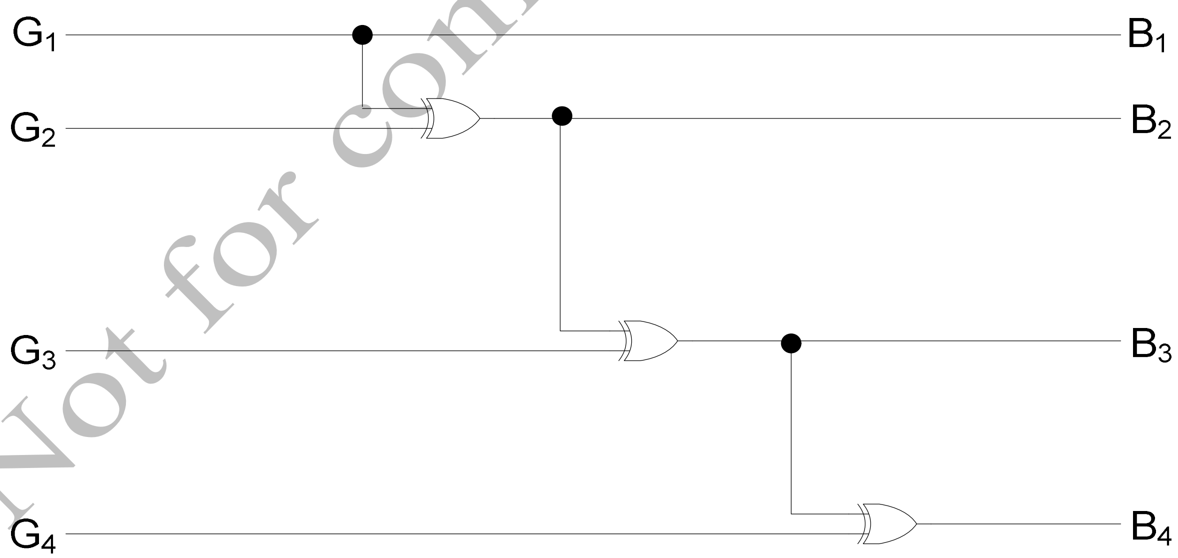
$B_3=G_1 \oplus G_2 \oplus G_3$

K-Map for B_4:

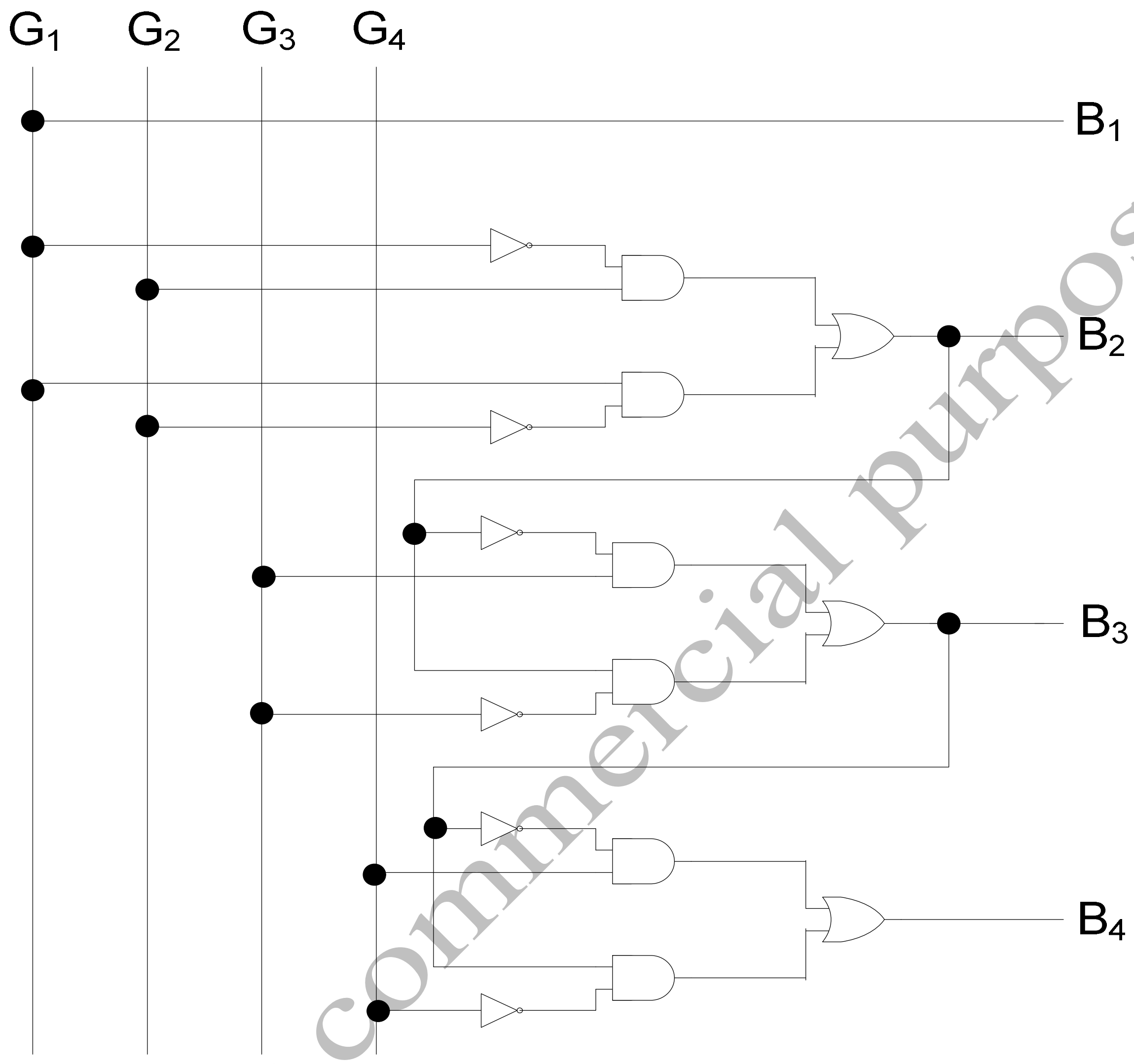
G_1G_2 \ G_3G_4	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$B_4 = G_1 \oplus G_2 \oplus G_3 \oplus G_4$$

Realization of Gray code to Binary using XOR gates:



Realization of Gray code to Binary code using Basic Gates:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

7. Design and verify the Truth Table of 3-bit Parity Generator and 4-bit Parity Checker using basic Logic Gates with an even parity bit.

Aim: To design and implement 3-bit Parity Generator and 4-bit Parity Checker:

3-bit Parity Generator:

Truth Table for 3-bit Parity Generator:

3-bit message			Even parity bit generator
A	B	C	P
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

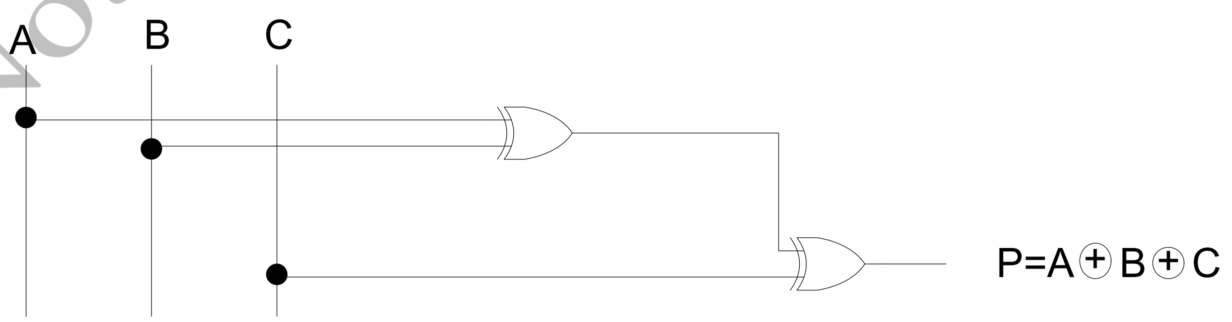
K-Map Simplification for P:

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

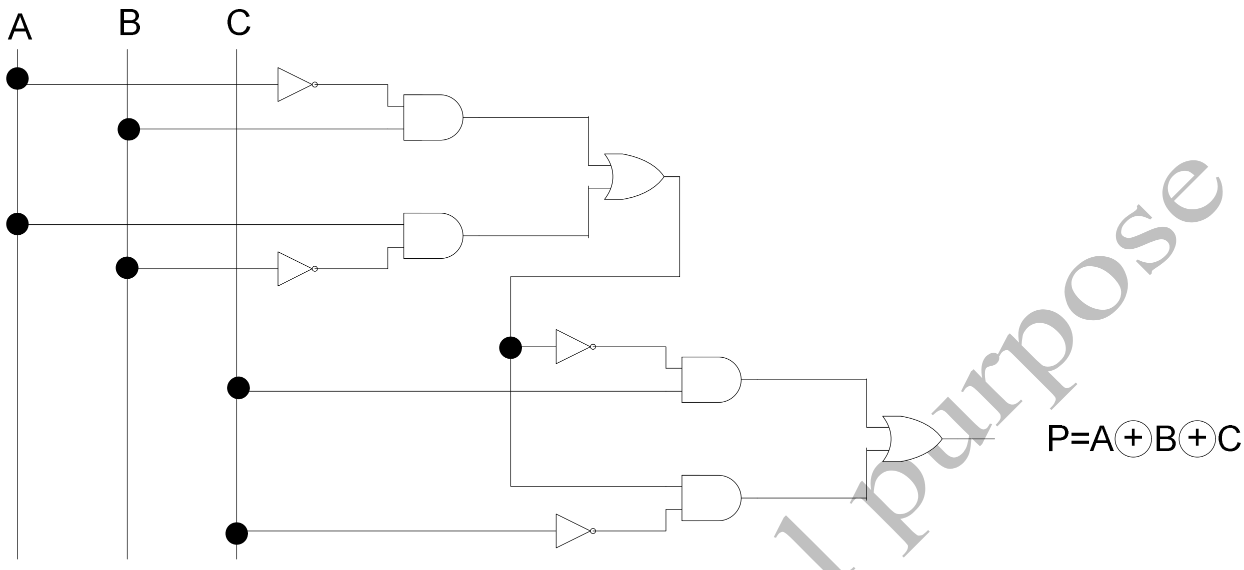
$P=\bar{A}\bar{B}C+\bar{A}B\bar{C}+A\bar{B}\bar{C}+ABC$

$P=A\oplus B\oplus C$

Realization 3-bit parity generator with XOR gates:



Realization 3-bit parity generator with Basic gates:



4-bit Parity Checker:

Truth table for 4-bit Parity Checker

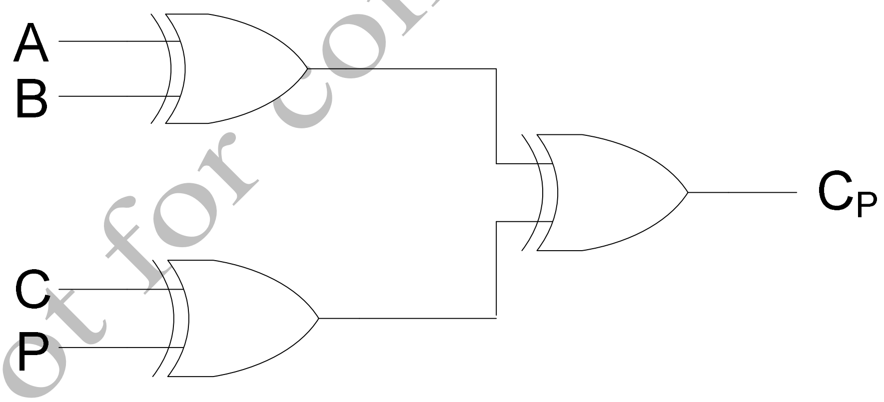
4-bit received message				Parity error check(C_P)
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K-Map Simplification for C_P:

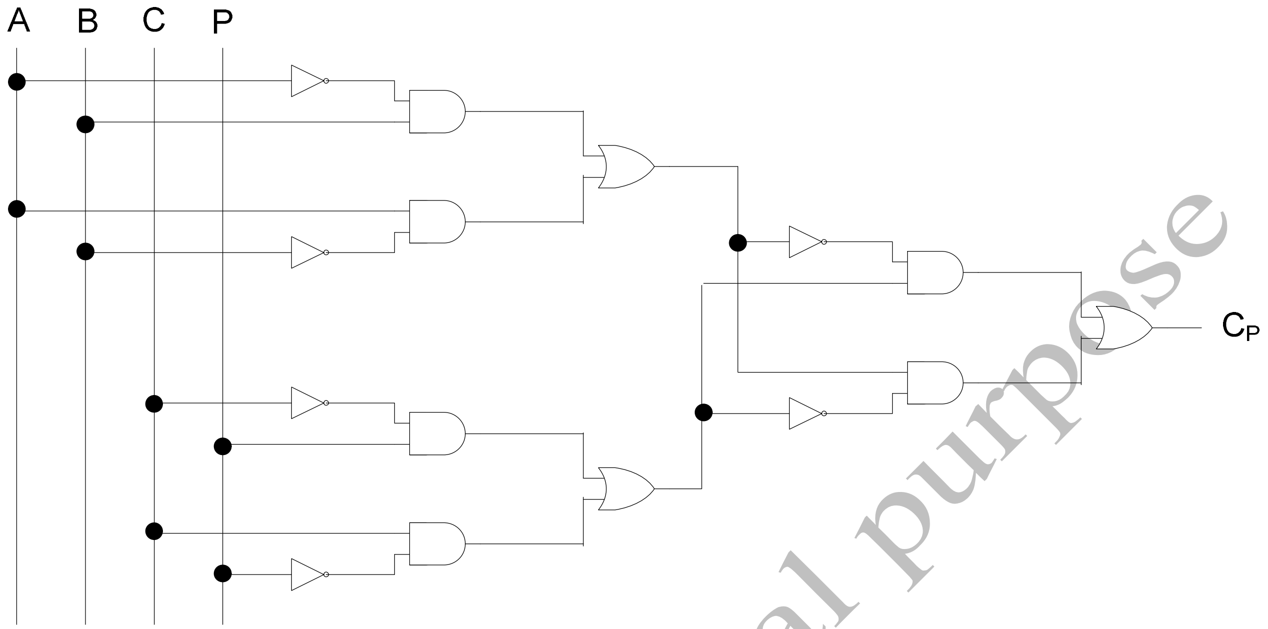
AB \ CP	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

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

Realization of 4-bit parity checker using XOR gates:



Realization of 4-bit parity checker using Basic gates:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

8. a) Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table.

Aim: To realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table.

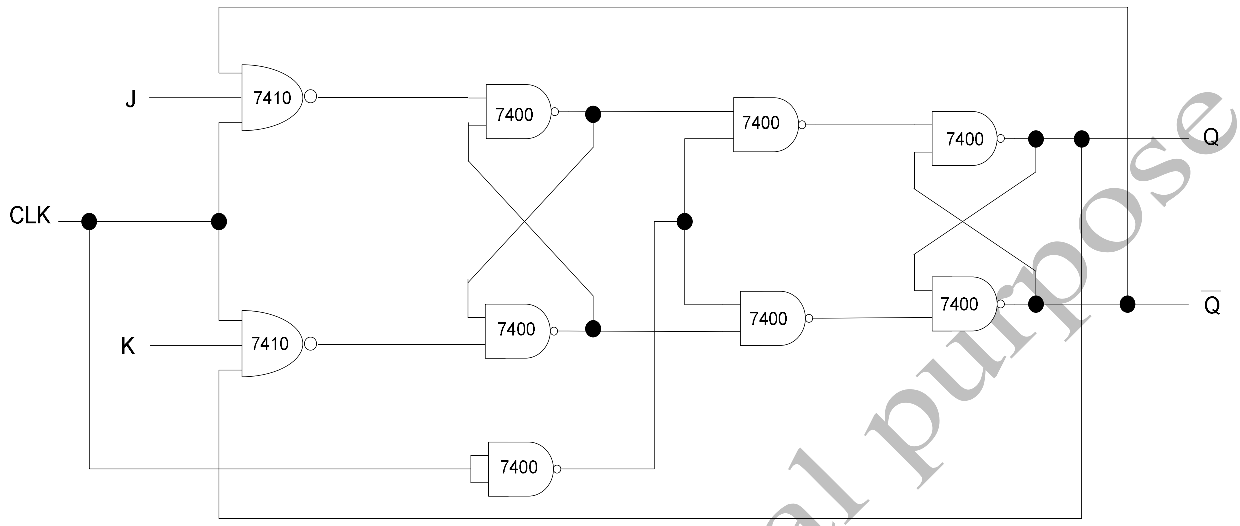
Equipment and Components required:

SL No.	Equipments and Components	Quantity
1	IC Trainer Kit	1
2	IC 7400 (2 input NAND Gate)	2
3	IC 7410 (3 input NAND Gate)	1
4	Patch Cord	1 Bunch

Truth table

J	K	Q
0	0	No Change
0	1	0
1	0	1
1	1	Toggle

Circuit Diagram:



Procedure:

(i) Connect the circuit as shown in the above diagram.

(ii) Apply the input combination and verify the truth table.

b) **Design and develop the Verilog / VHDL code for D Flip-Flop with positive-edge triggering. Simulate and verify its working.**

Truth Table

Clock(CLK)	D	Q
1	0	0
2	1	1
3	0	0
4	1	1

VHDL code for D Flip Flop (Behavioral modeling):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dflipflop is
Port( D,CLK : in std_logic;
      Q: inout  std_logic;
      QBAR: inout std_logic);
end dflipflop;
architectural Behavioral of  dflipflop  is
begin
process(CLK) is
begin
if rising_edge(CLK) then
Q<=D;
end if;
end process;
QBAR<= not Q;
end Behavioral;
```

9. **a) Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.**

Aim: To design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.

Component and Equipment required

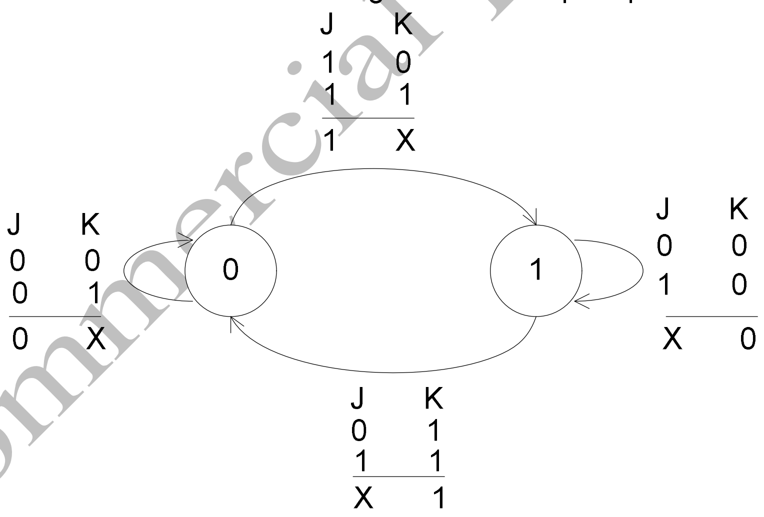
Sl. No.	Component	Quantity
1	IC Trainer Kit	1
2	IC 7476 (JK Flip Flop IC)	2
3	IC 7408 (AND Gate IC)	1
4	Patch Cord	1 Bunch

(I) Design and implementation of mod-5:

Truth Table:

Clock	Counter Output (IC 7476)		
	Q_3	Q_2	Q_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0

State Transition Diagram for JK Flip Flop



Truth Table for JK Flip Flop

J	K	Q_{n+1}	Action
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}_n$	Toggle

Table for the design of Modulo-5 synchronous counter

Present State			Next State			J_3	K_3	J_2	K_2	J_1	K_1
Q_3	Q_2	Q_1	Q_3+	Q_2+	Q_1+						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

K-map for J_3:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_3=Q_2Q_1$

K-map for K_3:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	X	X
1	1	X	X	X

$K_3=1$

K-map for J_2:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0	1	X	X
1	0	X	X	X

$J_2=Q_1$

K-map for K_2

Q_3 \ Q_2Q_1	00	01	11	10
0	X	X	1	0
1	X	X	X	X

$K_2 = Q_1$

K-map for J_1:

Q_3 \ Q_2Q_1	00	01	11	10
0	1	X	X	1
1	0	X	X	X

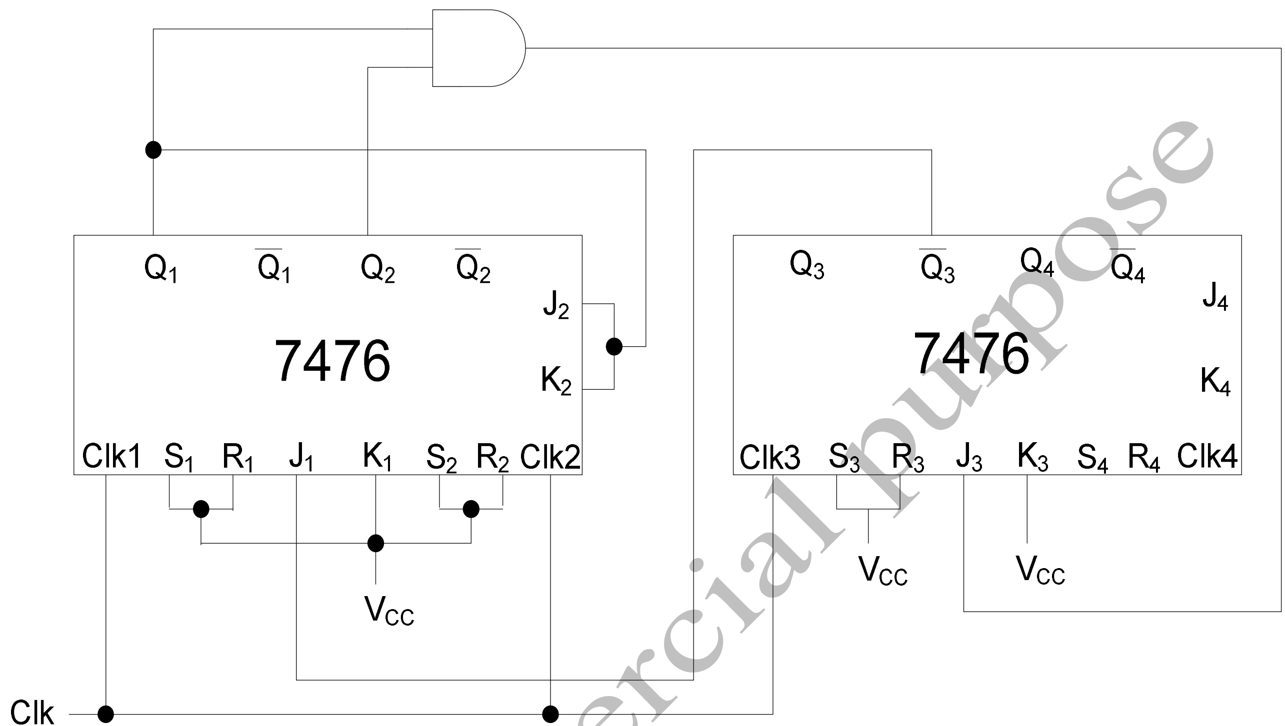
$J_1 = \overline{Q}_3$

K-map for K_1:

Q_3 \ Q_2Q_1	00	01	11	10
0	X	1	1	X
1	1	X	X	X

$K_1 = 1$

Circuit diagram:



(II) Design and implementation of mod-6:

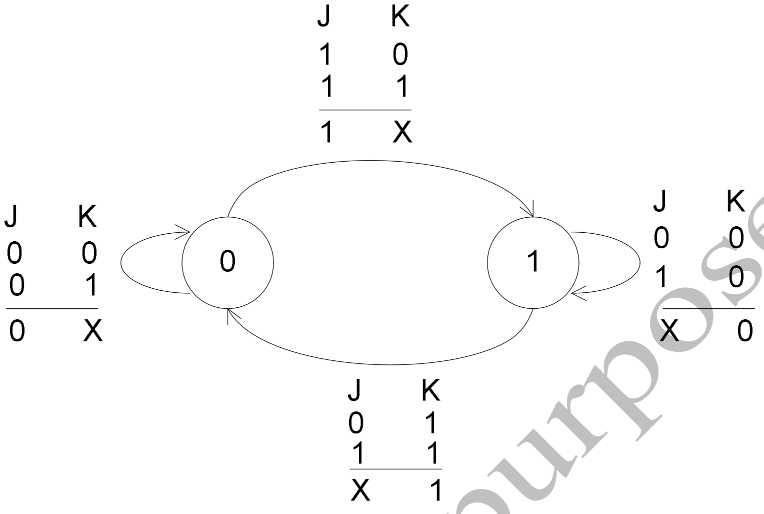
Truth Table:

Clock	Counter Output (IC 7476)		
	Q_3	Q_2	Q_1
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	0	0	0

Truth Table for JK Flip Flop

J	K	Q_{n+1}	Action
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}_n$	Toggle

State Transition Diagram for JK Flip Flop



State Table for the design of Modulo-5 synchronous counter

Present State			Next State			J_3	K_3	J_2	K_2	J_1	K_1
Q_3	Q_2	Q_1	Q_3+	Q_2+	Q_1+						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1

K-map for J_3:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_3=Q_2Q_1$

K-map for K_3:

Q_3 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	0	1	X	X

$K_3=Q_1$

K-map for J_2:

Q_3 \ Q_2Q_1	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$J_2=\overline{Q_3}Q_1$

K-map for K_2:

Q_3 \ Q_2Q_1	00	01	11	10
0	X	X	1	0
1	X	X	X	X

$K_2=Q_1$

K-map for J_1:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	1	X	X	1
1	1	X	X	X

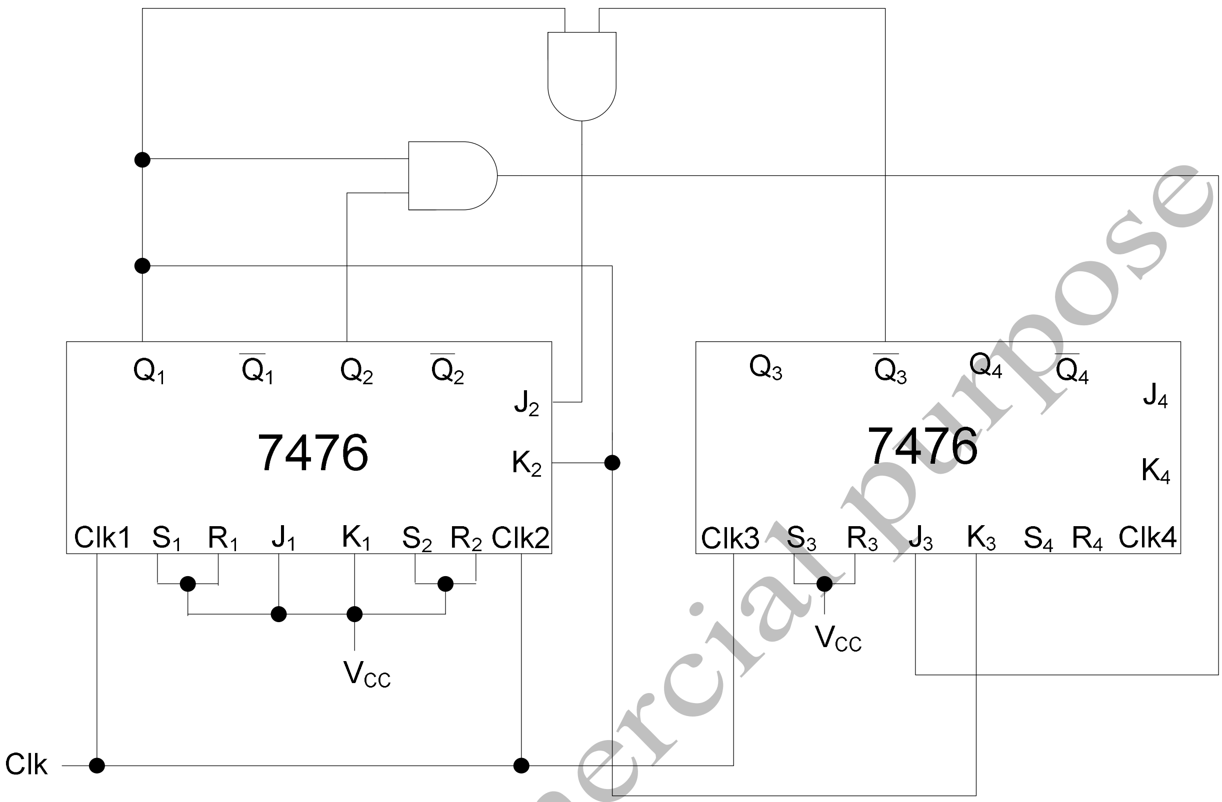
$J_1=1$

K-map for K_1:

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	1	1	X
1	1	X	X	X

$K_1=1$

Circuit diagram:



Procedure:
(i) Connect the circuit as per design.
(ii) Apply the input combination and verify the truth table.

b) Design and develop the Verilog / VHDL code for mod-8 up counter. Simulate and verify its working.

Truth Table

Clock (CLK)	R	E	Q
0	1	0	0000
1	0	1	0001
2	0	1	0010
3	0	1	0011
4	0	1	0100
5	0	1	0101
6	0	1	0110
7	0	1	0111
8	0	1	0000

VHDL code for Mod-8 counter (Behavioral modeling):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mod8upcounter  is
Port( R,CLK,E : in std_logic;
 Q: inout  std_logic_vector(3 downto 0));
 end mod8upcounter;
architectural Behavioral of  mod8upcounter  is
begin
process(CLK,R) is
begin
if R='1' then Q<="0000";
else if rising_edge(CLK) then
if E='1' then
Q<=Q+1;
end if;
if Q="0111" then
Q<="0000"
end if;
end if:
end if;
end process;
end Behavioral;
```

10. Design and implement an asynchronous counter using decade counter IC to count up from 0 to n(n<=9) and demonstrate on 7-segment display (using IC-7447).

Aim: To design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and demonstrate on 7-segment display (using IC-7447).

Components and Equipment required:

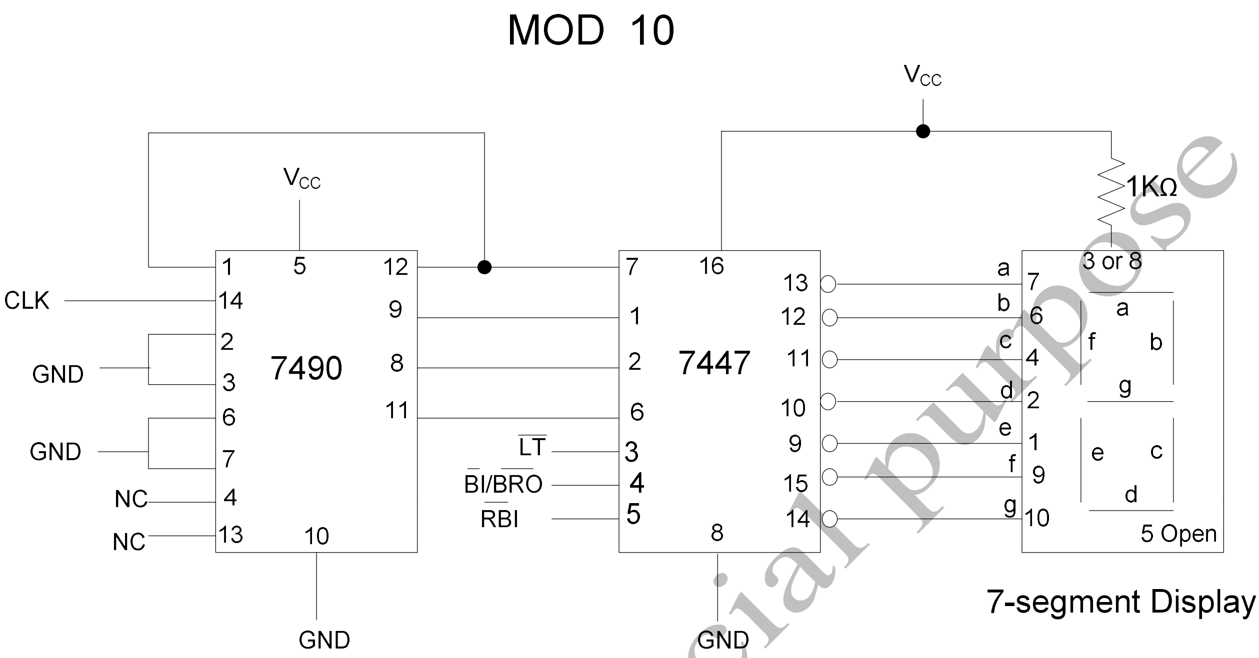
Sl. No.	Component	Quantity
1	IC Trainer Kit	1
2	IC 7490 (Decade counter IC)	1
3	IC 7447 (Decoder IC)	1
4	IC 7410 (3 input NAND Gate)	1
5	DRB	1
6	7-Segment Display	1
7	Patch Cord	1 Bunch

Design and implementation:

(I) Count up from 0 to 9:

Truth Table

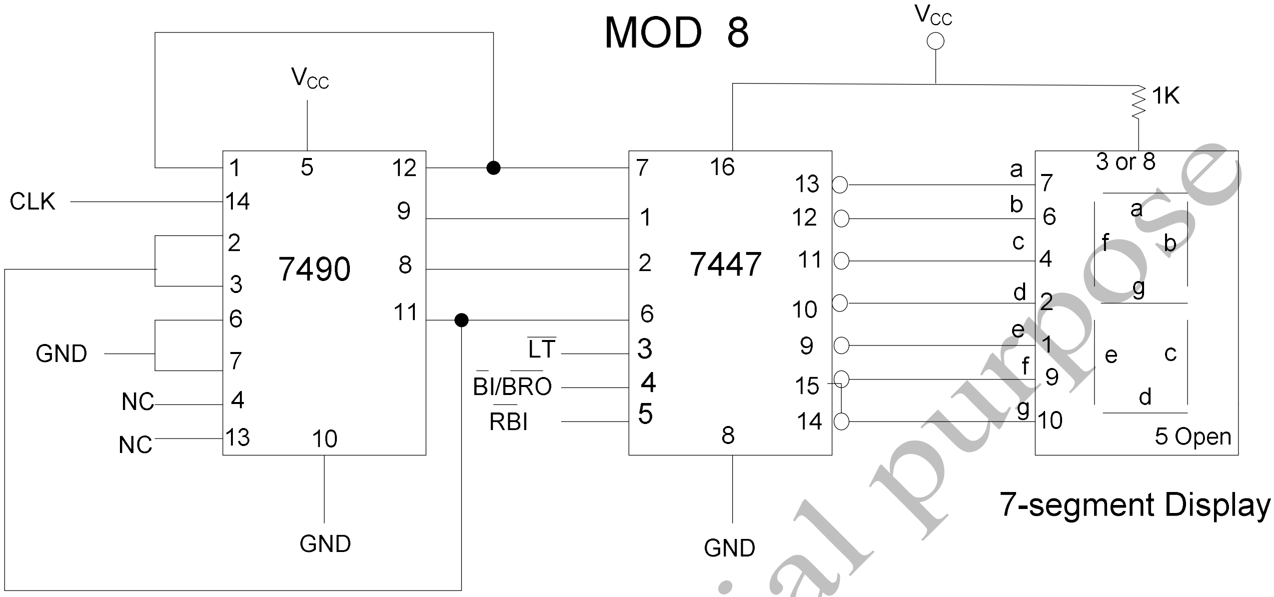
Clock	Counter Output (IC7490)				Decimal Output (7-Segment Display)
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	0	0	0	0	0



(II) Count from 0 to 7:

Truth Table

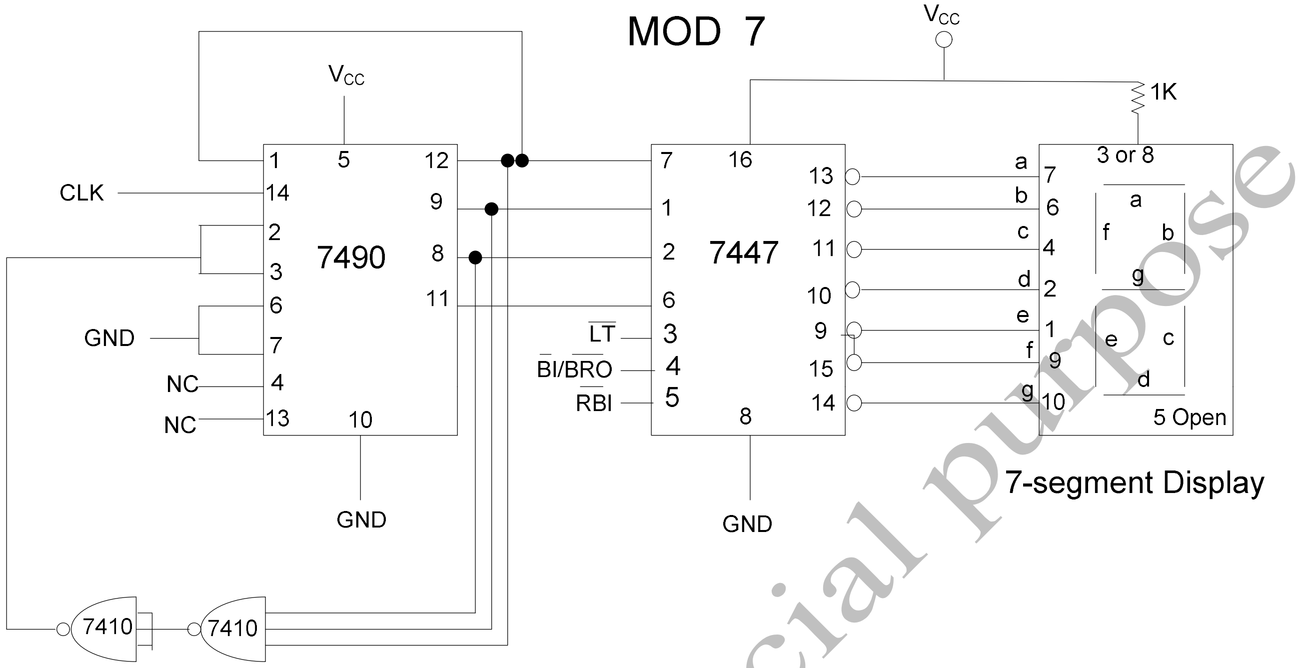
Clock	Counter Output (IC7490)				Decimal Output (7-Segment Display)
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	0	0	0	0	0



(III) Count from 0 to 6

Truth Table

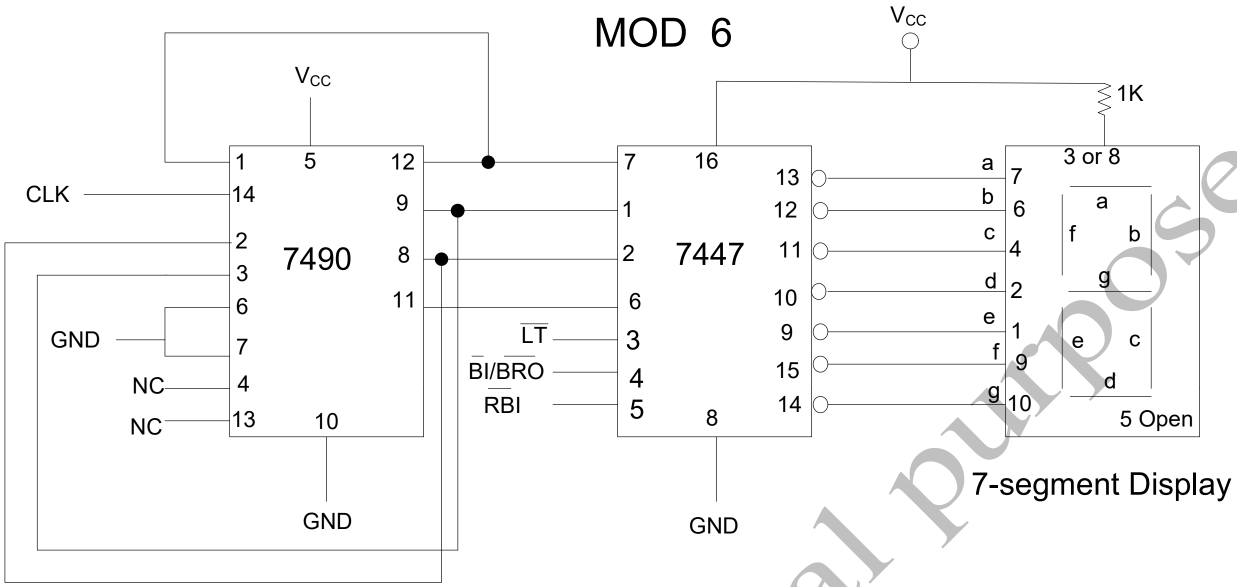
Clock	Counter Output (IC7490)				Decimal Output (7-Segment Display)
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	0	0	0	0



(IV) Count from 0 to 5:

Truth Table

Clock	Counter Output (IC7490)				Decimal Output (7-Segment Display)
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	0	0	0	0



Procedure:
(i) Connect the circuit as shown in the above diagram.
(ii) Apply the input combination and verify the truth table.

NOTE:
$\overline{LT}$ stands for Lamp Test. When $\overline{LT}$ is low, all the segments on the 7-segment display are lit regardless Of DCBA.

$\overline{BI}$ stands for Blanking Input. When $\overline{BI}$ is low, the display is blank so all the segments on the 7-segment display are off regardless of DCBA.

$\overline{RBI}$ stands Ripple Blanking Input. When $\overline{RBI}$ is low and DCBA=0000 the display is blank otherwise the number is displayed on the display. This is used to remove leading zeroes from a number.

11. Generate a Ramp output waveform using DAC0800 (Inputs are given to DAC through IC74393 dual 4-bit binary counter).

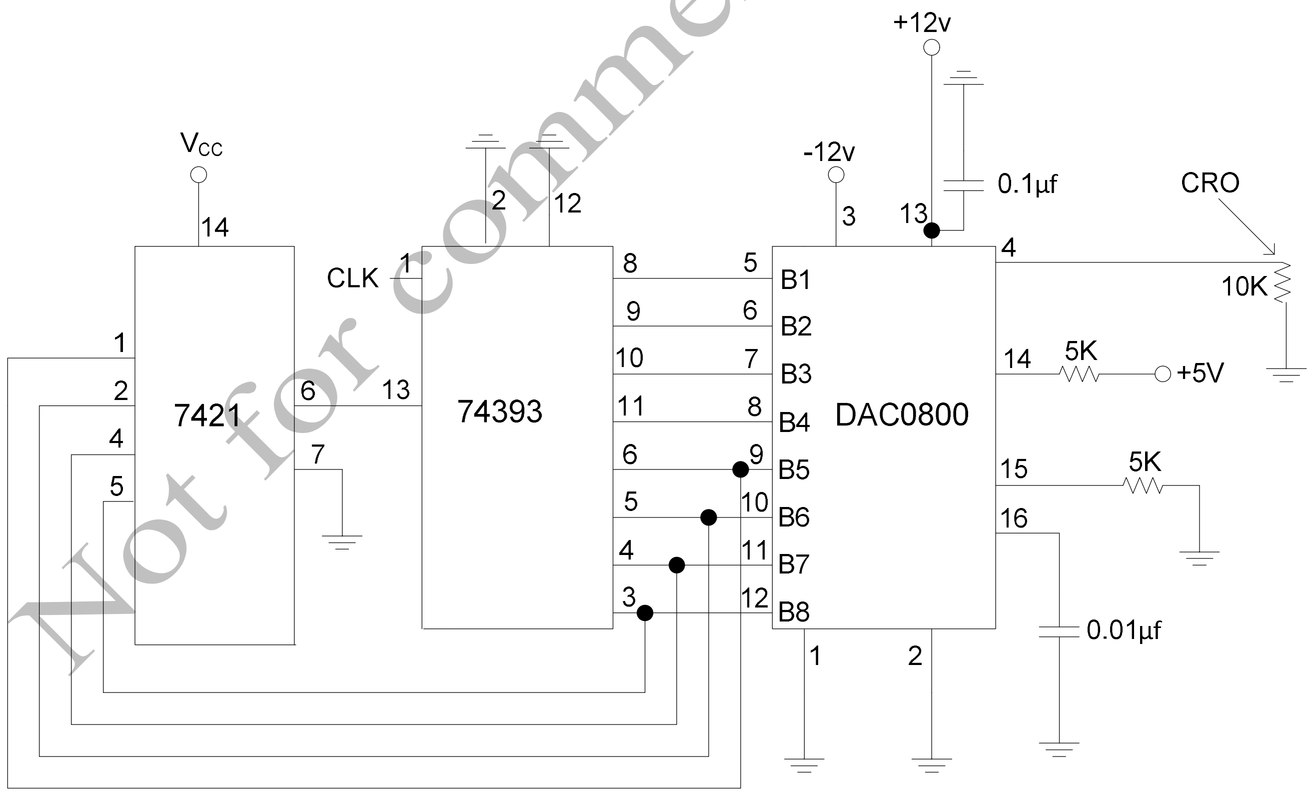
Aim: Generate a Ramp output waveform using DAC0800 (Inputs are given to DAC through IC74393 dual 4-bit binary counter).

Equipments and components required:

Sl. No.	Equipments and components	Quantity
1	IC Trainer Kit	1
2	CRO	1
3	DAC0800 (Digital to Analog Converter IC)	1
4	IC 74393 (Dual 4-bit binary counter)	1
5	IC 7421 (4 input AND Gate)	1
6	Resistors:5KΩ	2
7	Resistor: 10KΩ	1
8	Capacitor: 0.1μf and 0.01μf	2
9	Patch Cord	1 Bunch

Design and implementation:

Circuit Diagram:



Procedure:
(i) Connect the circuit as shown.
(ii) Apply the input clock pulse.
(iii) Observe the waveform at CRO

Waveform: