3. ASTABLE MULTIVIBRATOR USING 555 TIMER

AIM : To design and implement an astable multivibrator using 555 Timer for a given frequency and duty cycle.

COMPONENTS REQUIRED: 555 Timer IC, Resistors 3.3KΩ , 6.8KΩ, Capacitors of 0.1μF,

0.01 μF, Regulated power supply,CRO

THEORY:

Multivibrator is a form of oscillator, which has a non-sinusoidal output. The output waveform is rectangular. The multivibrators are classified as: **Astable or free running multivibrator**: It alternates automatically between two states (low and high for a rectangular output) and remains in each state for a time dependent upon the circuit constants. It is just an oscillator as it requires no external pulse for its operation. **Monostable or one shot multivibrator**: It has one stable state and one quasi stable. The application of an input pulse triggers the circuit time constants. After a period of time determined by the time constant, the circuit returns to its initial stable state. The process is repeated upon the application of each trigger pulse. **Bistable Multivibrators**: It has both stable states. It requires the application of an external triggering pulse to change the output from one state to other. After the output has changed its state, it remains in that state until the application of next trigger pulse. Flip flop is an example.

CIRCUIT DIAGRAM :

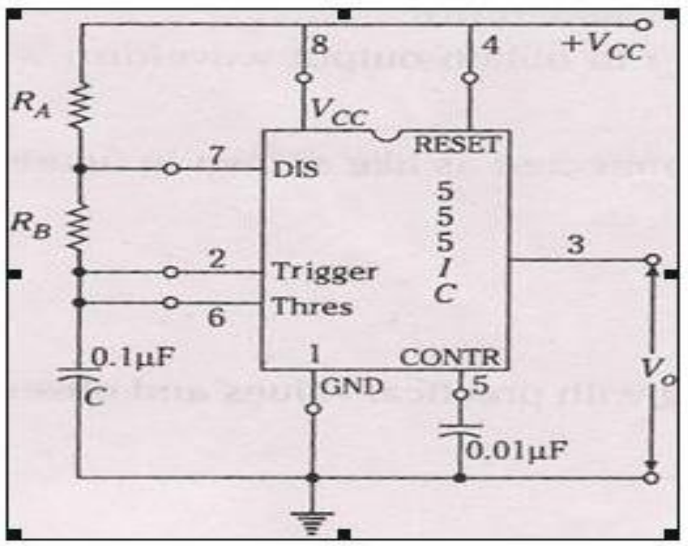


Fig 3a.1: Circuit Diagram and actual connections

DESIGN :

Given frequency (f) = 1KHz and duty cycle = 75% (=0.75)

The time period $T = 1/f = 1ms = t_H + t_L$

Where t_H is the time the output is high and t_L is the time the output is low.

From the theory of astable multivibrator using 555 Timer(refer Malvino), we have

$t_H = 0.693\ R_B\ C$ ------(1)

$t_L = 0.693\ (R_A + R_B)C$ ------(2)

$T = t_H + t_L = 0.693\ (R_A + 2\ R_B)\ C$

Duty cycle = $t_H / T = 0.75$.

Hence $t_H = 0.75T = 0.75ms$ and

$t_L = T - t_H = 1ms - 0.75ms = 0.25ms$.

Let C=0.1μF and substituting in the above equations,

$R_B = 6.8K\Omega$ (from equation 1) and $R_A = 3.3K\Omega$ (from equation 2 & R_B values).

The Vcc determines the upper and lower threshold voltages (observed from the capacitor voltage waveform) as $V_{UT} = \frac{2}{3}V_{CC}\ \&\ V_{LT} = \frac{1}{3}V_{CC}$.

PROCEDURE:

1. Before making the connections, check the components using multimeter.
2. Make the connections as shown in figure and switch on the power supply **Fig 3.1**.
3. Observe the capacitor voltage waveform at 6th pin of 555 timer on CRO.
4. Observe the output waveform at 3rd pin of 555 timer on CRO (shown below) **Fig 3.2**.
5. Note down the amplitude levels, time period and hence calculate duty cycle.

WAVEFORMS

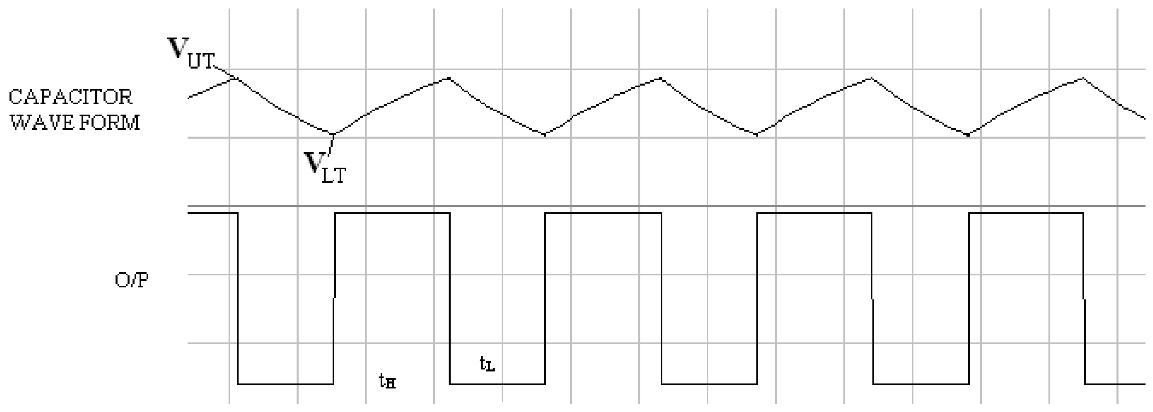


Fig 3.2: waveform of Astable multivibrator 555 timer

RESULT:

t_H =-------------

t_L = ------------

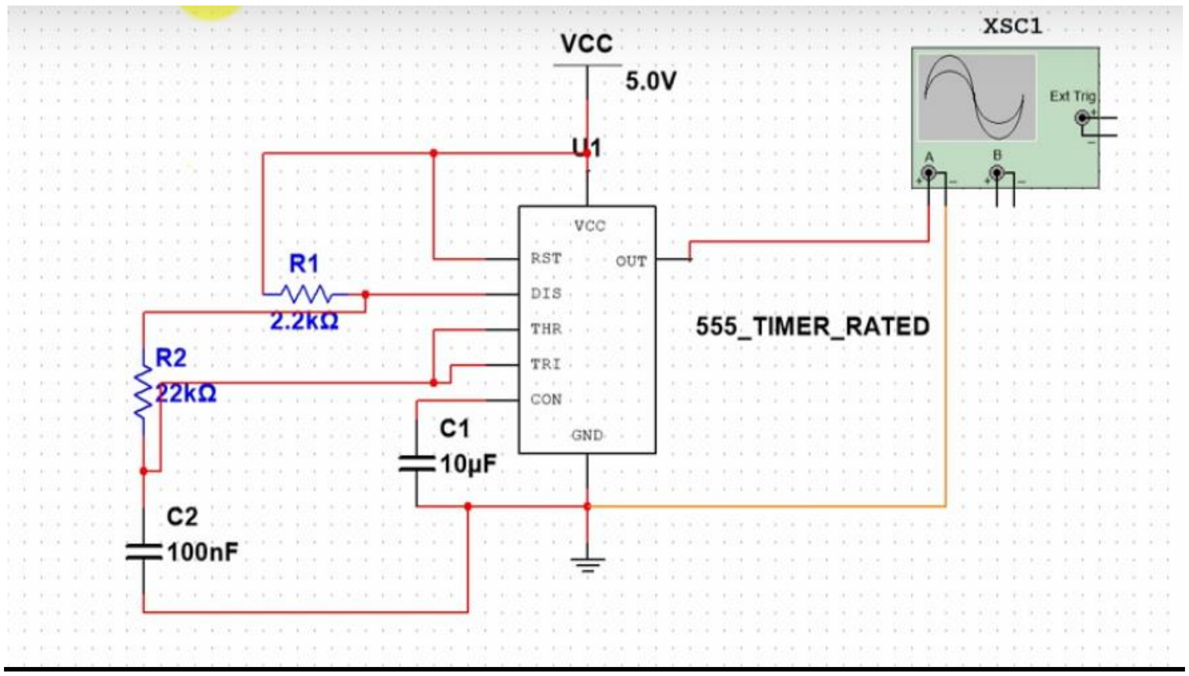
T = -------------

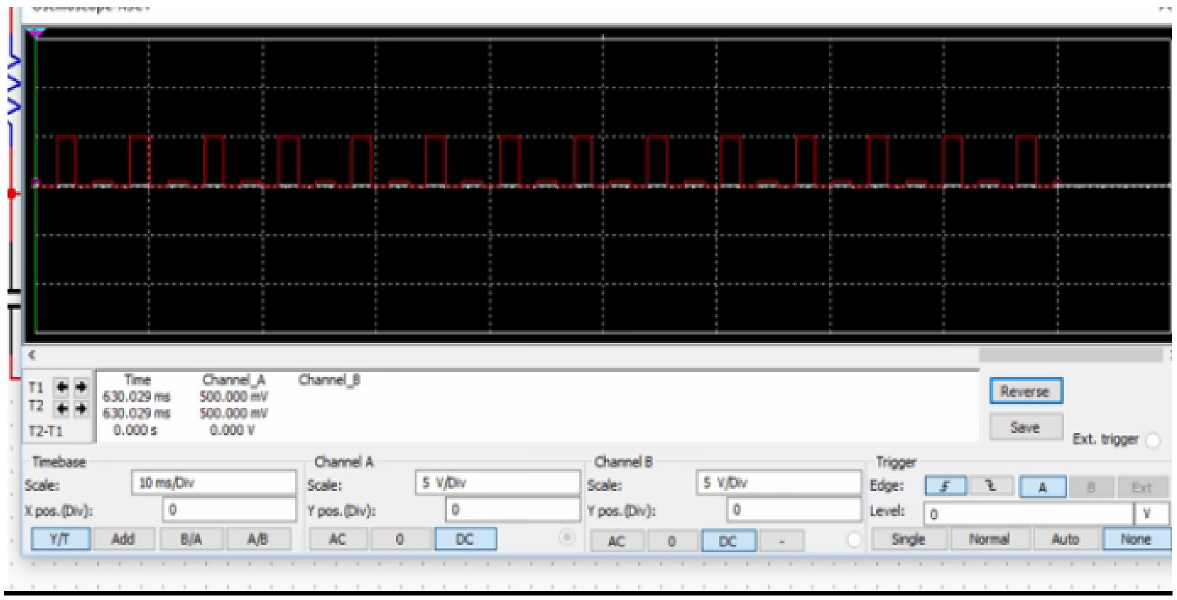
The frequency of the oscillations = 1/T= ___ Hz.

%Duty cycle (DC) = tH/T*100 = ----------

Sl.no	**f KHz**	**DUTY CYCLE(%)**	**R_A (KHz)**	**R_B (KHz)**
1	1KHz	60 %		
2				
3				

1B. 555 TIMER USING MULTISIM





	Time	Channel_A	Channel_B
T1	630.029 ms	500.000 mV	
T2	630.029 ms	500.000 mV	
T2-T1	0.000 s	0.000 V	

Timebase Scale: 10 ms/Div X pos.(Div): 0 [✓ Y/T] Add B/A A/B

Channel A Scale: 5 V/Div Y pos.(Div): 0 AC 0 [✓ DC]

Channel B Scale: 5 V/Div Y pos.(Div): 0 AC 0 [✓ DC] -

Trigger Edge: [✓ ↑] ↓ [✓ A] B Ext Level: 0 V Single Normal Auto [✓ None]

2a. OP-AMP AS A RELAXATION OSCILLATOR

AIM : To design and implement a rectangular waveform generator (op-amp relaxation oscillator) for a given frequency.

COMPONENTS REQUIRED: Op-amp μA 741, Resistor of 1KΩ, 10KΩ, 20 kΩ, Capacitor of 0.1 μF, Potentiometer, Regulated DC power supply, CRO

THEORY

Op-Amp Relaxation Oscillator is a simple Square wave generator which is also called as a Free running oscillator or Astable multivibrator or Relaxation oscillator. In this figure the op-amp operates in the saturation region. Here, a fraction (R2/(R1+R2)) of output is fed back to the noninverting input terminal. Thus reference voltage is (R2/(R1+R2)) Vo and may take values as +(R2/(R1+R2)) Vsat or - (R2/(R1+R2)) Vsat. The output is also fed back to the inverting input terminal after integrating by means of a low-pass RC combination. Thus whenever the voltage at inverting input terminal just exceeds reference voltage, switching takes place resulting in a square wave output.

CIRCUIT DIAGRAM

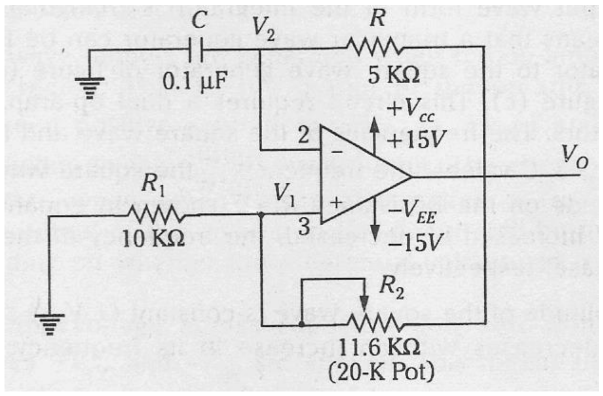


Fig 2a.1: Circuit Diagram & actual connections

DESIGN

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

Where, $\beta = \frac{R_1}{R_1 + R_2}$ is the feedback fraction

If $R_1 = R_2$, then from equation (1) we have T = 2RC ln(3)

Another example, if R_2=1.16 R_1, then T = 2RC ---------- (2)

Example: Design for a frequency of 1 kHz (implies $T = \frac{1}{f} = \frac{1}{10^3} = 10^{-3} = 1ms$)

Use R_2=1.16 R_1, for equation (2) to be applied.

Let R_1 = 10kΩ, then R_2 = 11.6kΩ (use 20kΩ potentiometer as shown in circuit figure)

Choose next a value of C and then calculate value of R from equation (2).

Let C=0.1μF (i.e., 10^{-7}), then $R = \frac{T}{2C} = \frac{10^{-3}}{2\times10^{-7}} = 5K\Omega$

The voltage across the capacitor has a peak voltage of $V_c = \frac{R_1}{R_1+R_2}V_{sat}$

V_{sat}= 90% * V_{cc}

=90/100* 15=13.5=12V

Vc= 10/10+11.6*12=5V

PROCEDURE

1. Before making the connections check all the components using multimeter.
2. Make the connections as shown in **Fig 2a.1** and switch on the power supply.
3. Observe the voltage waveform across the capacitor on CRO.
4. Also observe the output waveform on CRO. Measure its amplitude and frequency. Shown in **Fig 2a.2**

WAVEFORMS

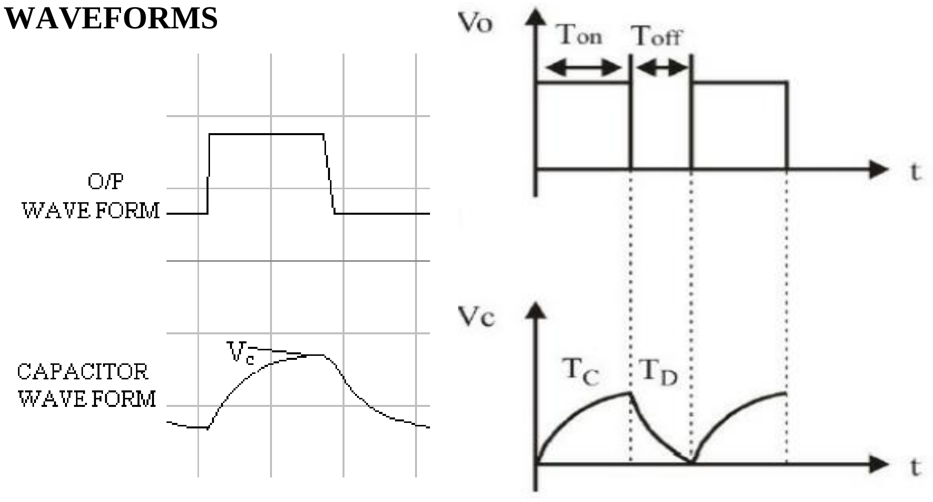


Fig 2a.2: Waveform

RESULT:

Sl.no	**f KHz**	**R**	**V_c**
1	**1 KHz**	**5kΩ**	**5V**
2	**1.5 KHz**		
3	**2 KHz**		

The frequency of the oscillations = _____ Hz.

Vc= Voltage across the capacitors = ---------V

2B. OP-AMP AS A RELAXATION OSCILLATOR

Aim : 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.

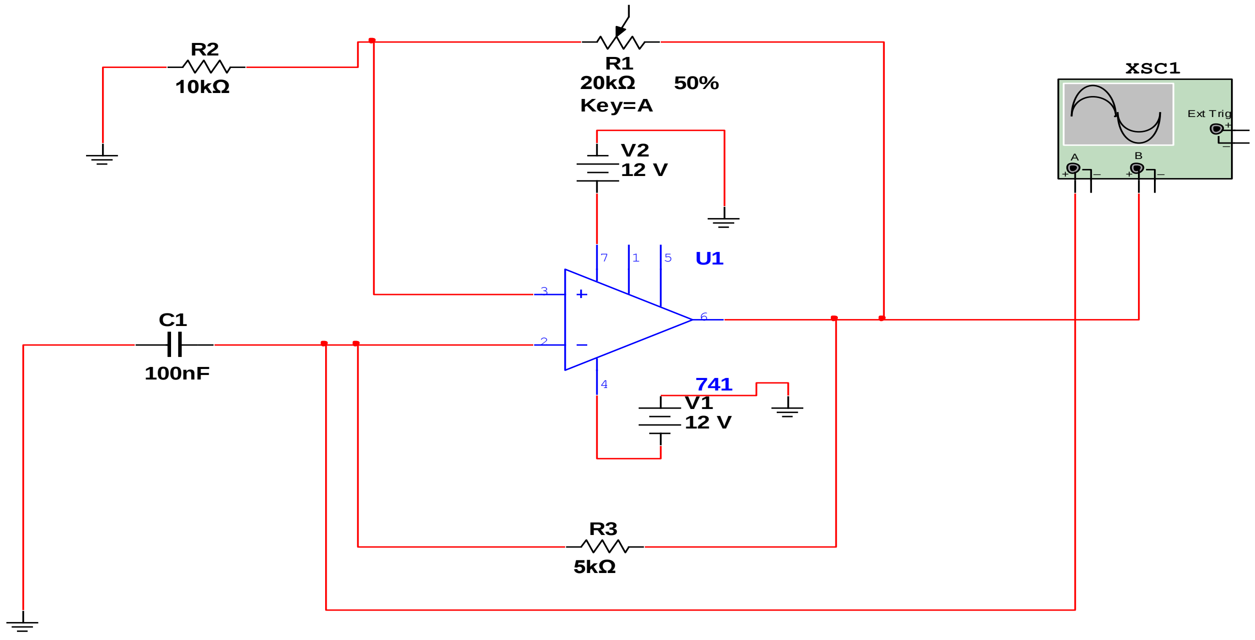
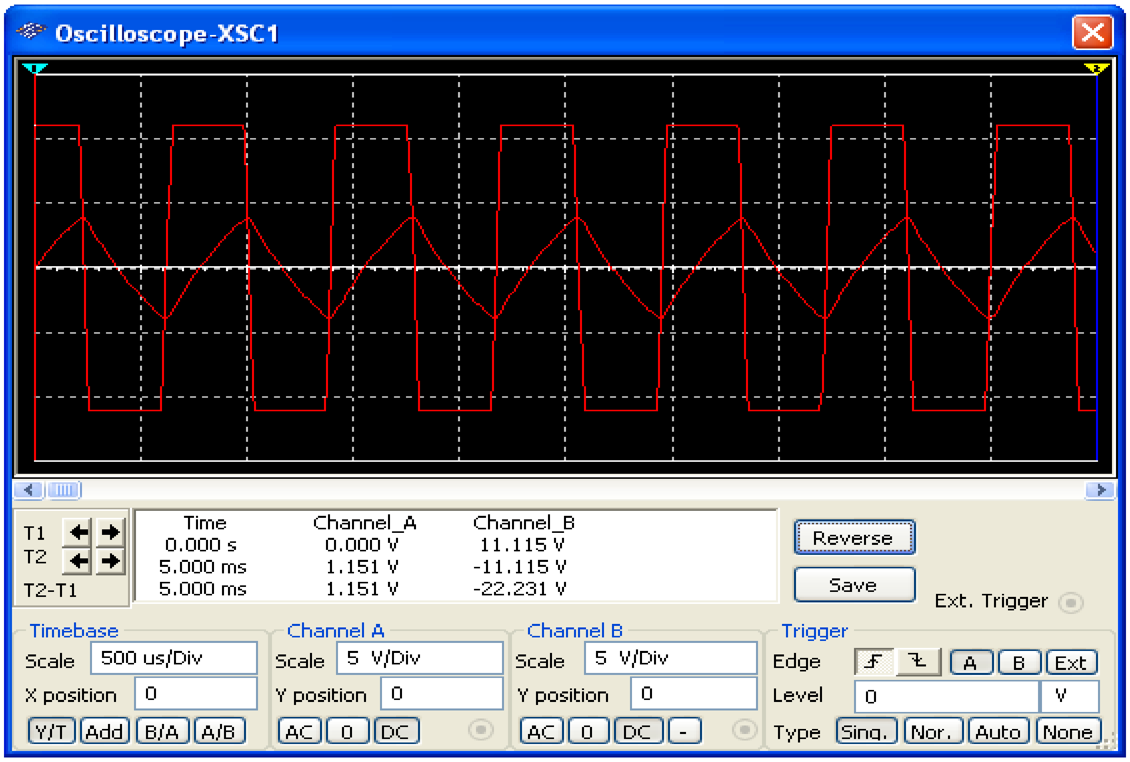


Fig 2b.1 Circuit Daigram relaxation Oscillator



```
Time        Channel_A   Channel_B
0.000 s     0.000 V     11.115 V
5.000 ms    1.151 V     -11.115 V
5.000 ms    1.151 V     -22.231 V
```

Timebase: Scale 500 us/Div; X position 0
Channel A: Scale 5 V/Div; Y position 0
Channel B: Scale 5 V/Div; Y position 0
Trigger: Edge; Level 0 V; Type

Fig 2b.2 Input and output waveform of relaxation oscillator

Experiment No.3: Using ua 741 Opamp, design window comparator for any given UTP and LTP. And simulate the same.

Description:

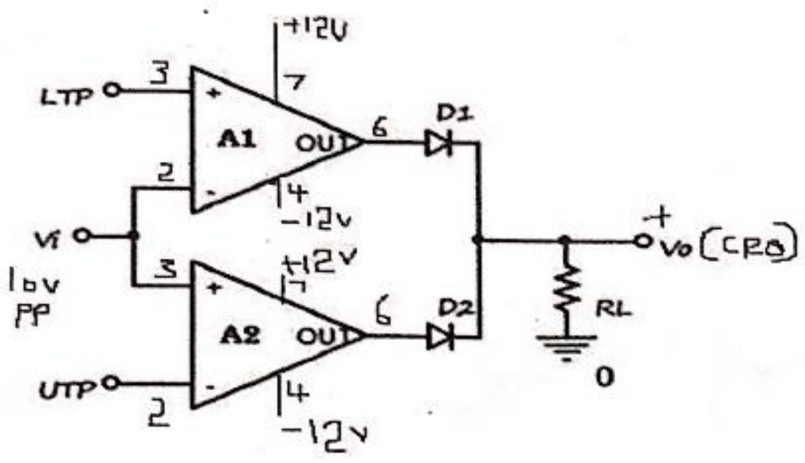
A **Window Comparator** is basically the inverting and the non-inverting comparators combined into a single comparator stage. The window comparator detects input voltage levels that are within a specific band or window of voltages, instead of indicating whether a voltage is greater or less than some preset or fixed voltage referencepoint.

This time, instead of having just one reference voltage value, a window comparator will have two reference voltages implemented by a pair of voltage comparators. One which triggers an op-amp comparator on detection of some upper voltage threshold, $V_{REF(UPPER)}$ and one which triggers an op-amp comparator on detection of a lower voltage threshold level, $V_{REF(LOWER)}$. Then the voltage levels between these two upper and lower reference voltages is called the "window", hence its name.

Components Required:

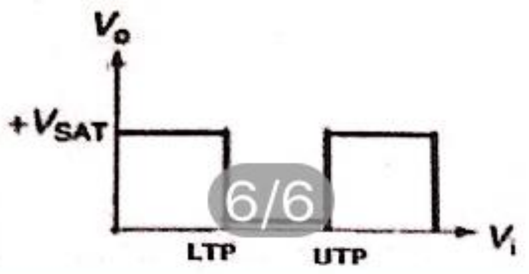
Two Op amp IC µA 741, Two diode 1N4007, Resistor of 1KΩ, DC regulated power Supply, trainer kit (+12v & -12v is given to Op amp from this), Signal generator, CRO.

Circuit:



Circuit Diagram for Window comparator

Output waveform



STUDY OF LOGIC GATES

AIM: To study about logic gates and verify their truth tables

COMPONENTS REQUIRED:

SL No.	COMPONENT	SPECIFICATION
1.	AND GATE	IC 7408
2.	OR GATE	IC 7432
3.	NOT GATE	IC 7404
4.	NAND GATE 2 I/P	IC 7400
5.	NOR GATE	IC 7402
6.	X-OR GATE	IC 7486
7.	NAND GATE 3 I/P	IC 7410
8.	IC TRAINER KIT	-
9.	PATCH CORD	-

THEORY:

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND ,NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

AND GATE:

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE:

The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

NOT GATE:

The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

NAND GATE:

The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.

NOR GATE:

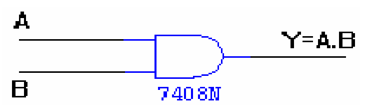
The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

X-OR GATE:

The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

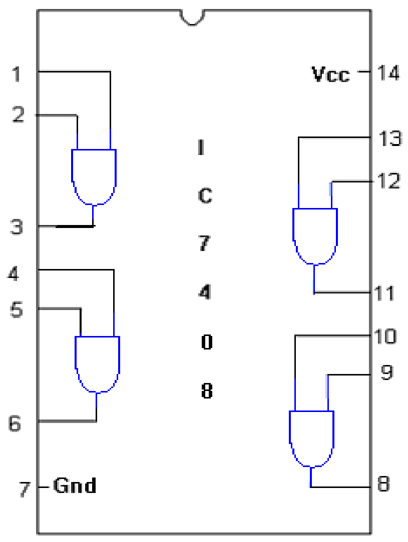
AND GATE:

SYMBOL: **PIN DIAGRAM:**



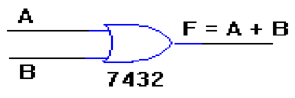
TRUTH TABLE

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

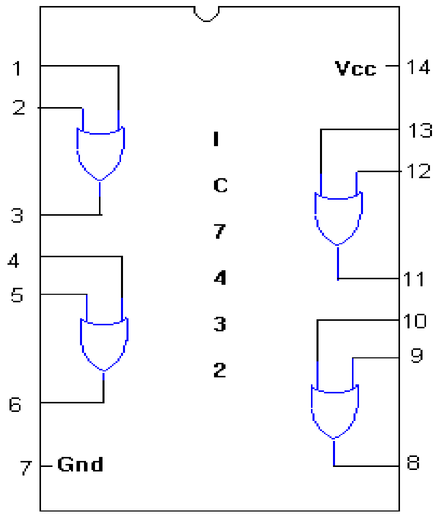


OR GATE:

SYMBOL :



PIN DIAGRAM :

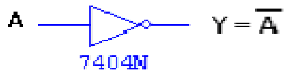


TRUTH TABLE

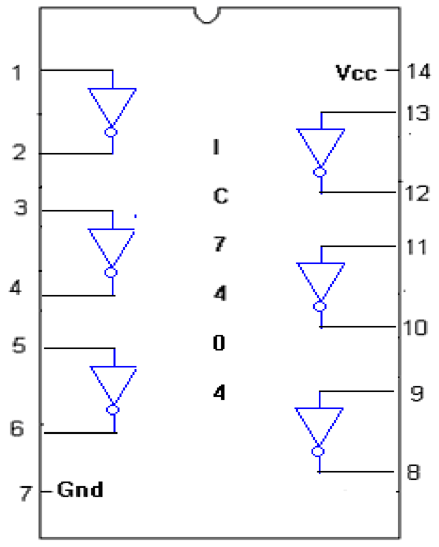
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

SYMBOL:



PIN DIAGRAM:

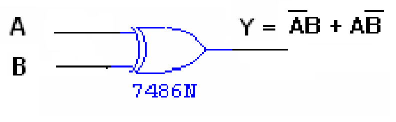


TRUTH TABLE :

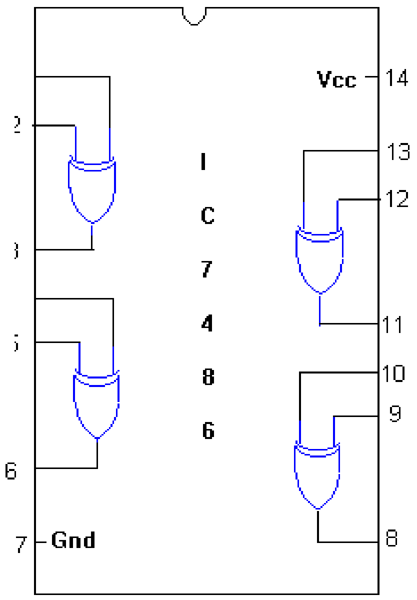
A	$\overline{A}$
0	1
1	0

XOR GATE:

SYMBOL :



PIN DIAGRAM :

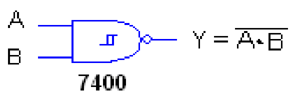


TRUTH TABLE :

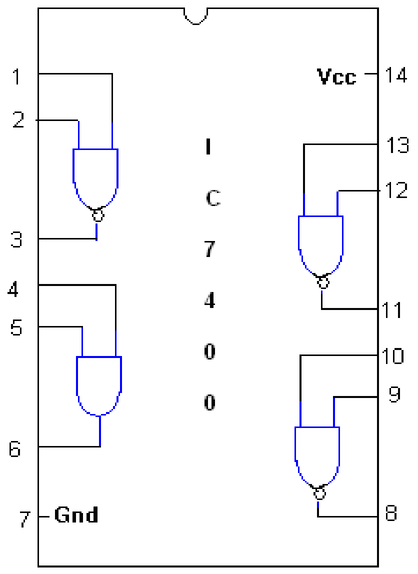
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

NAND GATE:

SYMBOL:



PIN DIAGRAM:

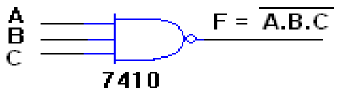


TRUTH TABLE

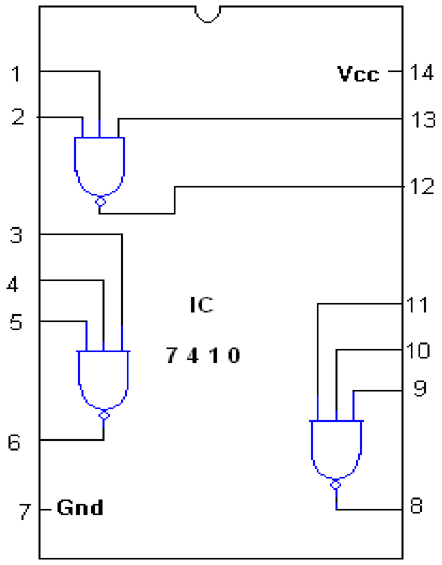
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

3 INPUT NAND GATE:

SYMBOL :



PIN DIAGRAM :

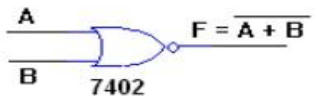


TRUTH TABLE

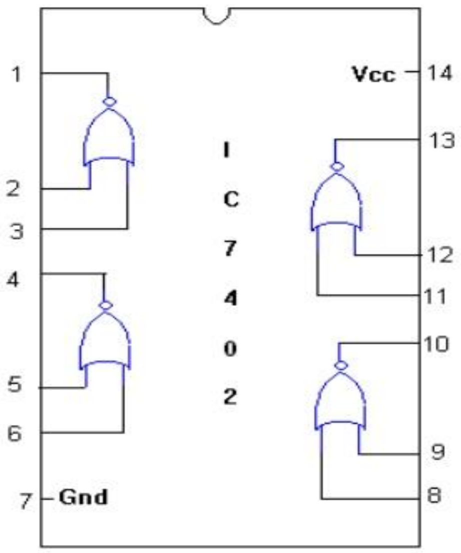
A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

NOR GATE

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

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

PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram

(iii) Observe the output and verify the truth table.

RESULT: All basic gates are verified

4. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates. And implement the same in HDL.

AIM: To realize

i) Half Adder and Full Adder

ii) Half Subtractor and Full Subtractor by using Basic gates.

COMPONENTS REQUIRED: IC 7400, IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C. The Boolean functions describing the half-adder are:

$S = A \oplus B$ $C = A B$

Full-Adder: The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin , is called a full-adder. The Boolean functions describing the full-adder are:

$S = (A \oplus B) \oplus Cin$ $C = AB + Cin\,(A \oplus B)$

Half Subtractor: Subtracting a single-bit binary value B from another A (i.e. A -B) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor. The Boolean functions describing the half-Subtractor are:

$S = A \oplus B$ $C = A' B$

Full Subtractor: Subtracting two single-bit binary values, B, Cin from a single-bit value A produces a difference bit D and a borrow out Br bit. This is called full subtraction. The Boolean functions describing the full-subtracter are:

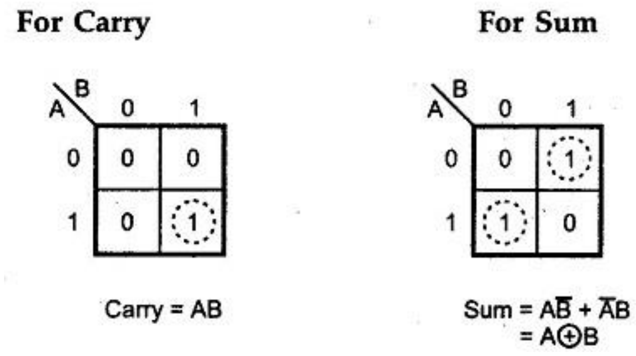
$D = (A \oplus B) \oplus Cin$ $Br = A'B + A'(Cin) + B(Cin)$

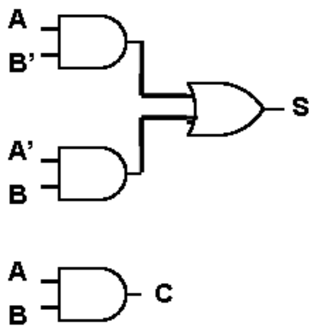
I. TO REALIZE HALF ADDER

TRUTH TABLE

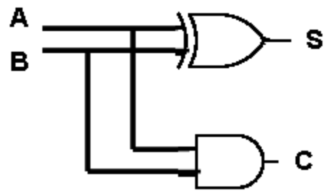
INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-maps





Half Adder implementation using AND-OR



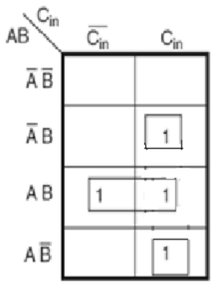
Half Adder implementation using AND-XOR

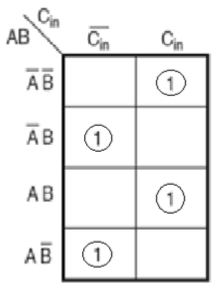
II. FULL ADDER

TRUTH TABLE

INPUTS			OUTPUTS	
A	B	Cin	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

k-maps

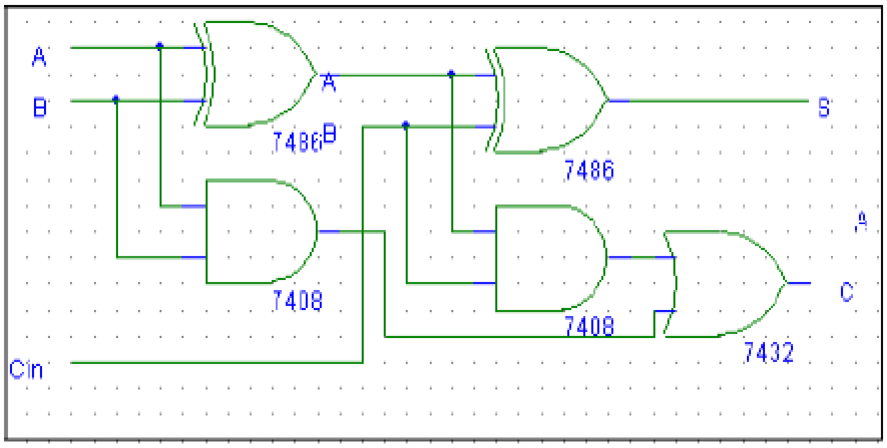




C= AB+ $\bar{A}BC_{in}$ + A $\bar{B}C_{in}$

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

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

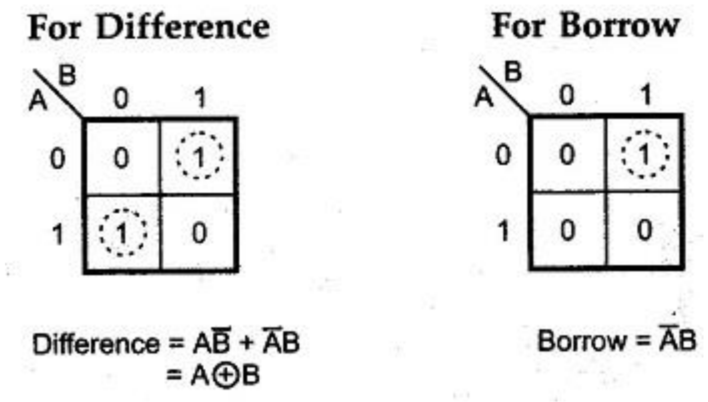


III. HALF SUBTRACTOR

TRUTH TABLE

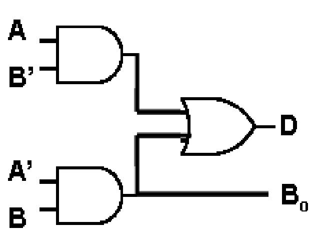
INPUTS		OUTPUTS	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

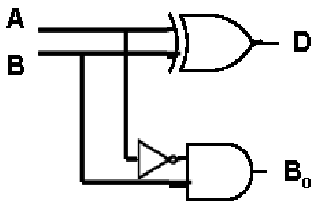
K-maps



Difference = $A\overline{B} + \overline{A}B$
$= A \oplus B$

Borrow = $\overline{A}B$





Half Subtractor implementation using AND-OR Half Subtractor implementation using AND-XOR

IV. FULL SUBTRACTOR

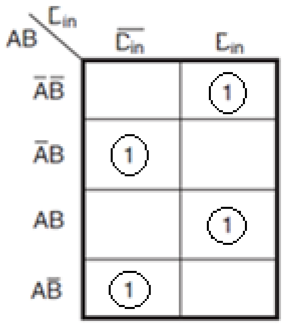
TRUTH TABLE

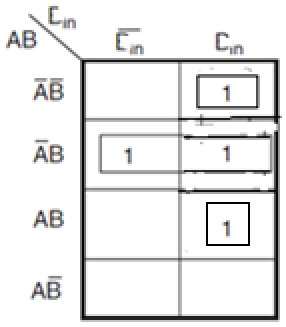
INPUTS			OUTPUTS	
A	B	Cin	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSIONS:

$D = A \oplus B \oplus C$

$Br = \bar{A}\, B + B\, Cin + \bar{A}\, Cin$

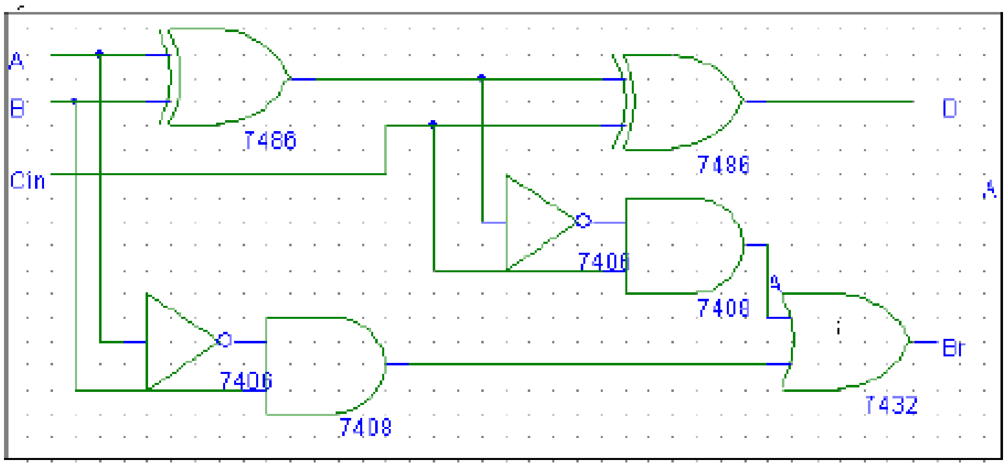




$$D = A \oplus B \oplus C_{in}$$

$$Br = \bar{A}B + \bar{A}\;\bar{B}C_{in} + ABC_{in}$$

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



VHDL

Procedure:

1. Click on Xilinx ise 7.1i.

2. Go to file and if any project is opened close that.

3. File→New project→project name(multix)

4. Device family(spartan2),Generated simulation language(VHDL),Modelsim(ISE simulator)

4. Click next till finish.

5. Select multix.ise file→right click→New source→VHDL module→File name→next

6. Give port names,input variables and output variables→next→finish.

7. Write VHDL code and save it.

8. Goto process view→synthesize-XST(double click).If any error clear it.

9. Select filename.behavioral→right click→new source→Testbench waveform→filename→ Next→next→finish.

10. Select combinatorial for multiplexer and single clock for other experiments,click OK.

11. Give the input waveform and save.

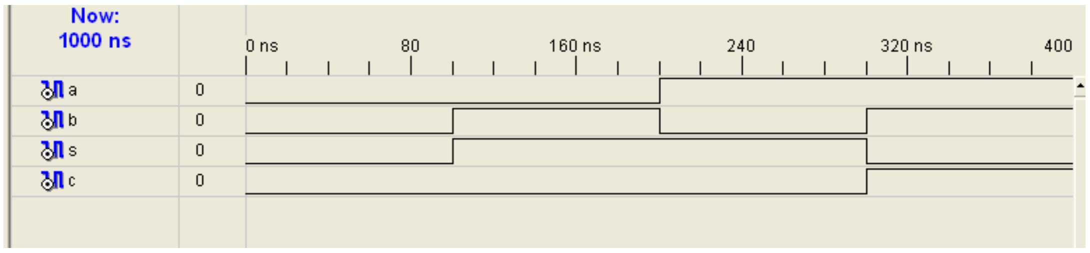
12. Select filename.tbw→click process view→double click simulate behavioral model.

13. The output waveform is shown.

AIM: Design and develop the VHDL code for **Half adder, Full Adder, Half Subtractor, Full Subtractor**. Simulate and verify its working.

CODE:Half adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adr is
        Port ( A,B: in std_logic;
        sum,carry : out std_logic;
end adr;

architecture Behavioral of adr is
        begin
                sum <= A xor B;
                carry <= A and B;

end Behavioral;
```



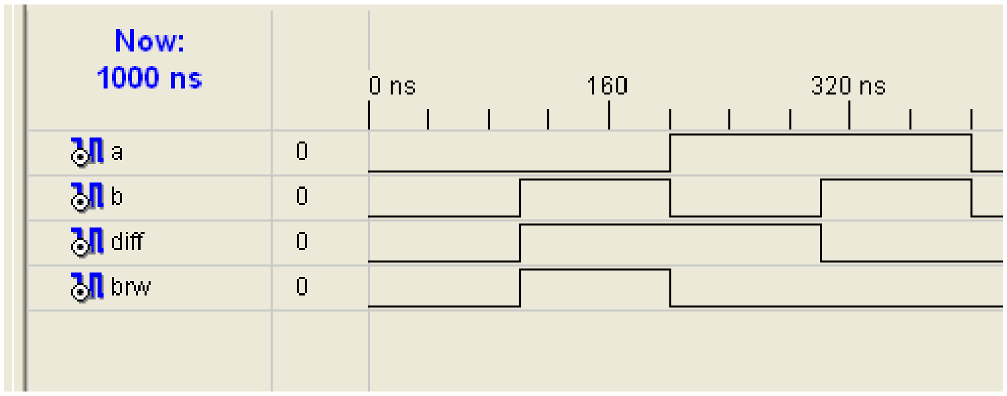
CODE: Full adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adr is
        port ( a,b,c: in std_logic;
        sum,carry : out std_logic;
end adr;

architecture behavioral of adr is
        begin
                sum <= a xor b xor c;
                carry <=  (a and b) or (c and (a xor b));
end behavioral;
```

CODE: Half subtractor

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adr is
        port ( a,b: in std_logic;
        diff,brw : out std_logic;
```

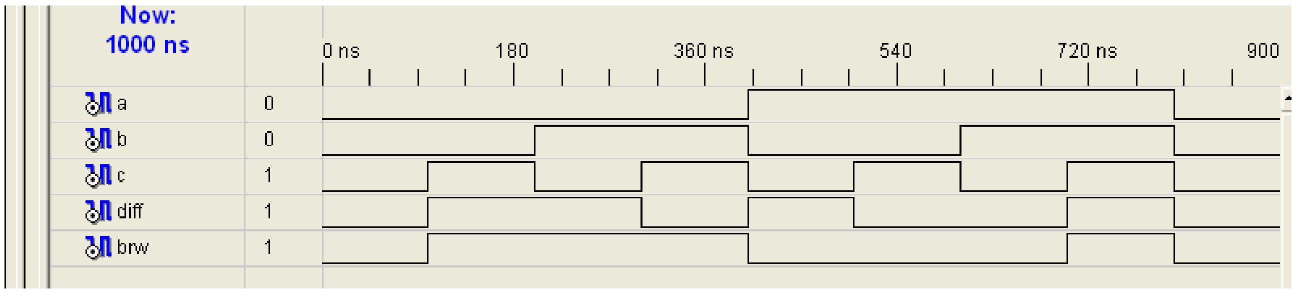
```
end adr;

architecture behavioral of adr is
        begin
                diff <= a xor b
                brw <=  (not a) and b;
        end behavioral;
```



CODE: Full subtractor

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity adr is
        port ( a,b: in std_logic;
        sum,carry : out std_logic;
end adr;

architecture behavioral of adr is
        begin
                diff <=a xor b xor c
                brw <= ((not A) and B) or (c and not(a xor b));
```

end behavioral;



Half Adder	sum <= a xor b;
	carry <= a and b;

Half Subtractor	diff <= a xor b brw <= (not a) and b;
Full Adder	sum <= a xor b xor c; carry <= (a and b) or (c and (a xor b));
Full Subtractor	diff <= a xor b xor c brw <= ((not A) and B) or (c and not(a xor b));

Result: Verified Half adder, Full adder, Half subtractor and Full subtarctor using Basic gates and also simulated using Xilinx.

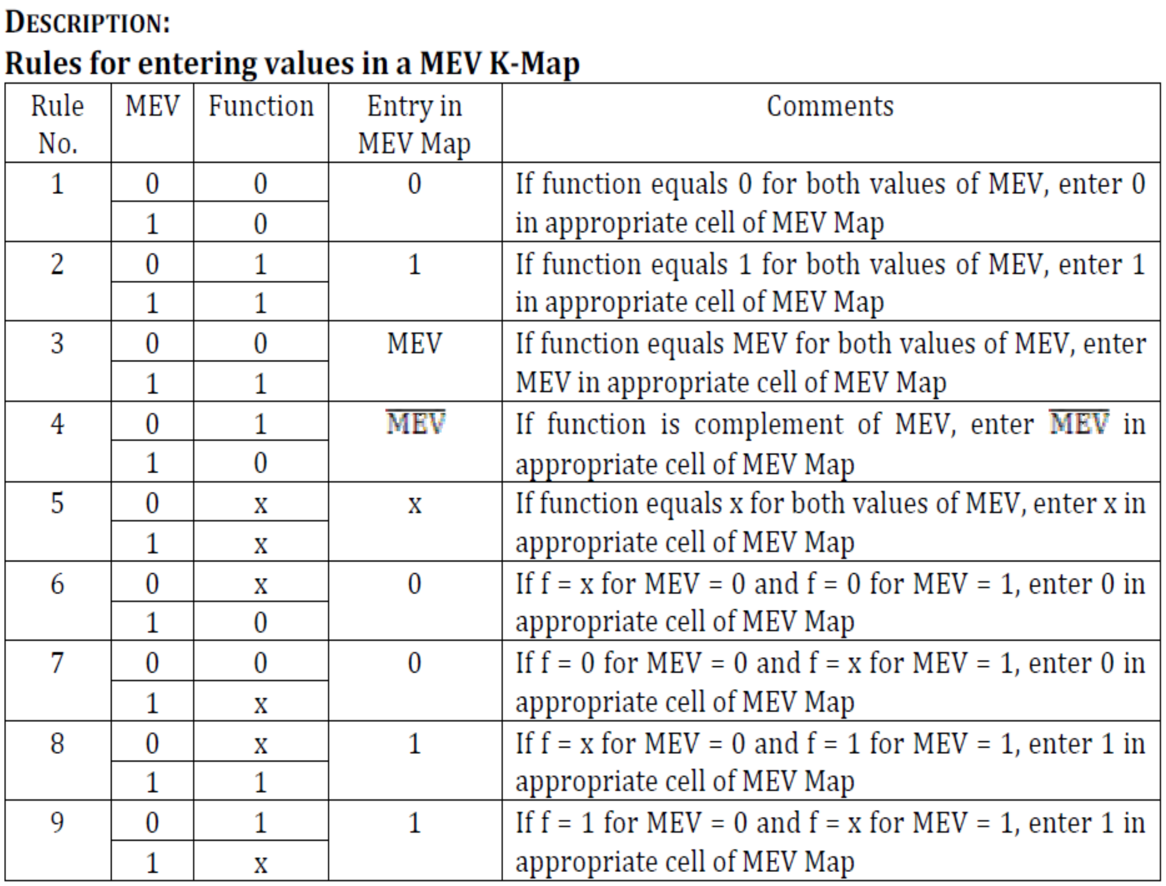
5 a. Given a 4-variable logic expression, simplify it using appropriate Technique and realize the simplified logic expression using 8:1 multiplexer IC. And implement the same in HDL.

AIM: To simplify 4 variables Boolean expression using Entered Variable Map method and realize the simplified logic using 8:1 MUX

COMPONENTS REQUIRED: IC74151,patch chords, trainer kit

THEORY:

Multiplexer sometimes is called universal logic circuit because a 2n to 1multiplexer can be used as a design solution for any n variable truth table. Let’s consider A Band C variables to be fed as select inputs, the fourth variable D as data input. We write all the combinations of 3 select inputs in first row along different columns. Now corresponding to each value of 4th variable D Truth table output Y is written. The 4th column Y as a function of D.



DESCRIPTION:

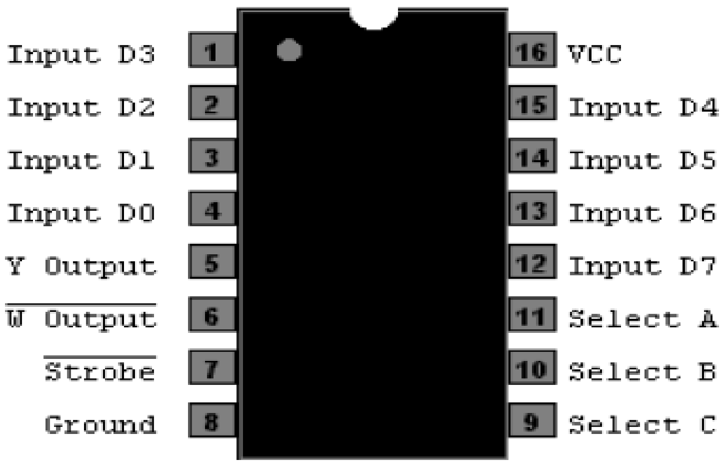
Rules for entering values in a MEV K-Map

Rule No.	MEV	Function	Entry in MEV Map	Comments
1	0	0	0	If function equals 0 for both values of MEV, enter 0 in appropriate cell of MEV Map
	1	0		
2	0	1	1	If function equals 1 for both values of MEV, enter 1 in appropriate cell of MEV Map
	1	1		
3	0	0	MEV	If function equals MEV for both values of MEV, enter MEV in appropriate cell of MEV Map
	1	1		
4	0	1	$\overline{MEV}$	If function is complement of MEV, enter $\overline{MEV}$ in appropriate cell of MEV Map
	1	0		
5	0	x	x	If function equals x for both values of MEV, enter x in appropriate cell of MEV Map
	1	x		
6	0	x	0	If f = x for MEV = 0 and f = 0 for MEV = 1, enter 0 in appropriate cell of MEV Map
	1	0		
7	0	0	0	If f = 0 for MEV = 0 and f = x for MEV = 1, enter 0 in appropriate cell of MEV Map
	1	x		
8	0	x	1	If f = x for MEV = 0 and f = 1 for MEV = 1, enter 1 in appropriate cell of MEV Map
	1	1		
9	0	1	1	If f = 1 for MEV = 0 and f = x for MEV = 1, enter 1 in appropriate cell of MEV Map
	1	x		

PROCEDURE:

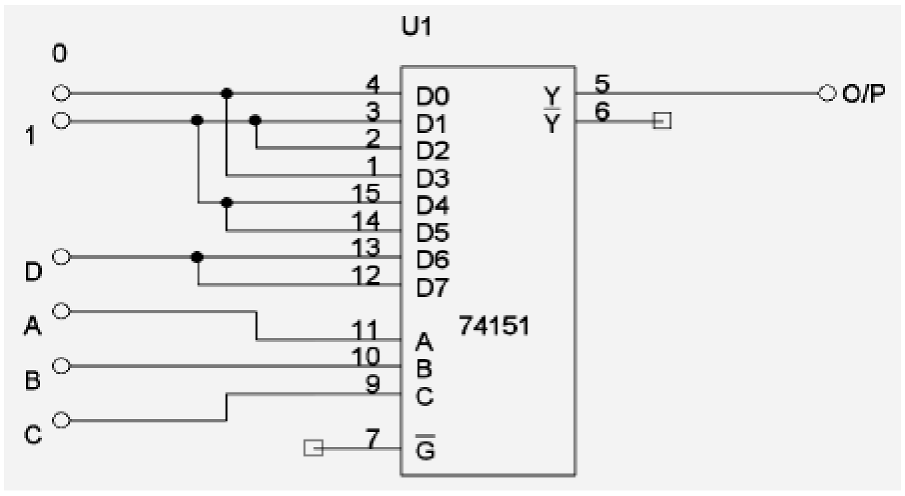
1. Verify all the components and patch chords whether they are in good condition or not.
2. Make connections as shown in the circuit diagram (fig 1.1).
3. Give power supply to the trainer kit.
4. Provide input data to the circuit via switches.
5. Record and verify the output sequence for each combination of the select lines.

PIN DETAILS – IC 74151



CIRCUIT DIAGRAM:

8:1 MULTIPLEXER



MEV TECHNIQUE

f(A,B,C,D) = ∑m(2,3,4,5,13,15) + dc(8,9,10,11)

Decimal	A	B	C	D	f	MEV map entry
0	0	0	0	0	0	0 →D0
1	0	0	0	1	0	
2	0	0	1	0	1	1 →D1
3	0	0	1	1	1	
4	0	1	0	0	1	1 →D2
5	0	1	0	1	1	
6	0	1	1	0	0	0 →D3
7	0	1	1	1	0	
8	1	0	0	0	X	X →D4
9	1	0	0	1	X	
10	1	0	1	0	X	X →D5
11	1	0	1	1	X	
12	1	1	0	0	0	D →D6
13	1	1	0	1	1	
14	1	1	1	0	0	D →D7
15	1	1	1	1	1	

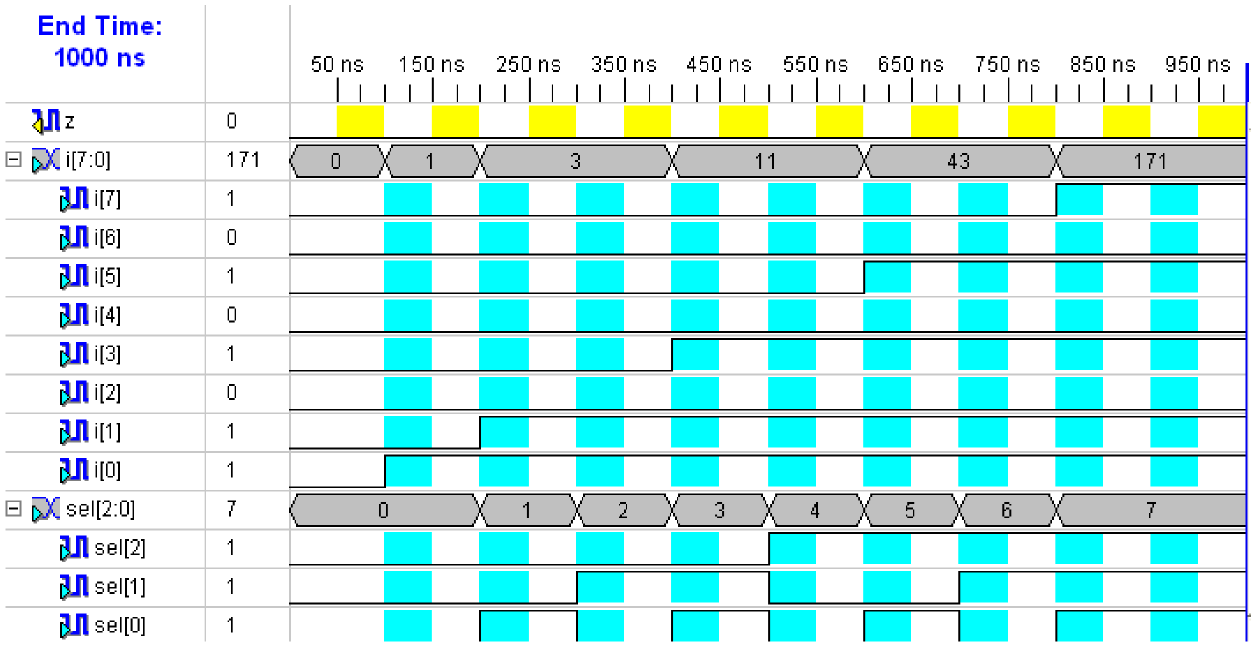
5B. 8:1 MULTIPLEXER

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

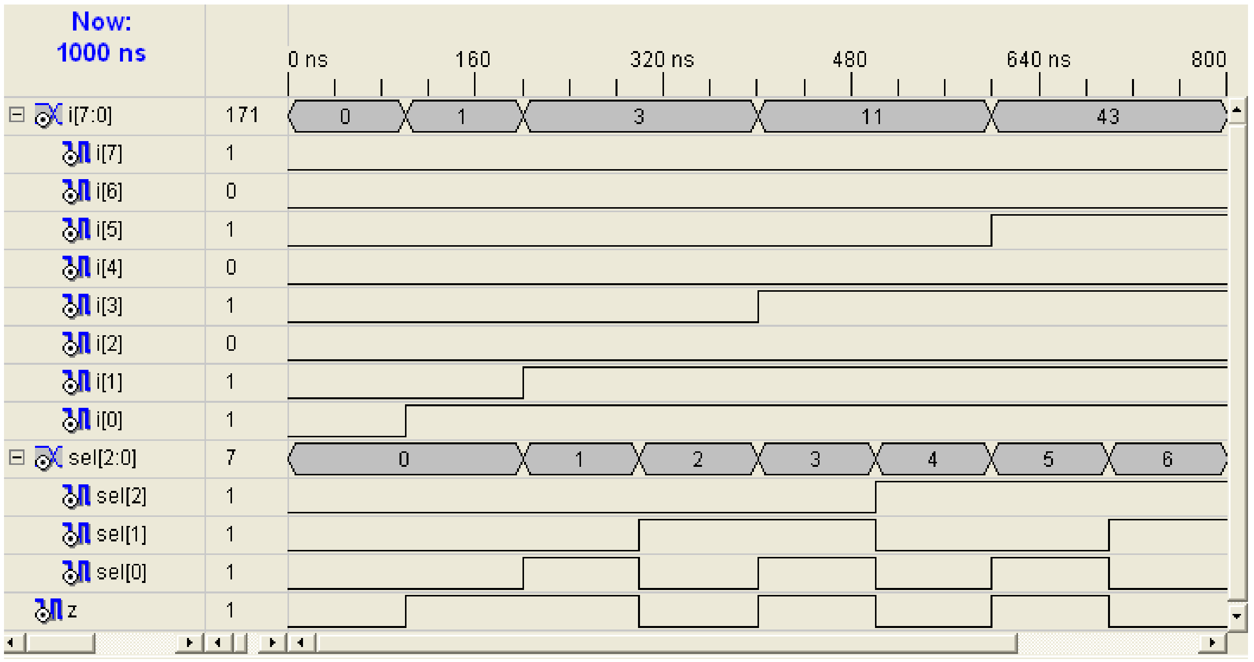
CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity rr is
        Port ( i : in std_logic_vector(7 downto 0);
        sel : in std_logic_vector(2 downto 0);
        z : out std_logic);
end rr;

architecture Behavioral of rr is
        begin
        z <= i(0) when sel="000"  else
        i(1) when sel="001"  else
        i(2) when sel="010"  else
        i(3) when sel="011"  else
        i(4) when sel="100"  else
        i(5) when sel="101"  else
        i(6) when sel="110"  else
        i(7);
end Behavioral;
```

Input Waveform



Output Waveform



6. Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table. and implement the same in HDL.

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

COMPONENTS REQUIRED: IC7410, IC7400, patch chords, trainer kit

THEORY

It does the same function as JK flip-flop with the difference that Master Slave units are pulse triggered ones rather than edge triggered ones. The Master Slave JK flipflop consists of flip-flop's one a master and the other a slave , the clock input is given to the master and the clock' is given to the slave such that when the clock is high master is enabled and the slave is disabled and the output of the master is at steady state, when the clock goes low the clock' is high and the master is disabled while slave is enabled now the output of the master reaches the slave output there by overcoming the race around problem

DESCRIPTION

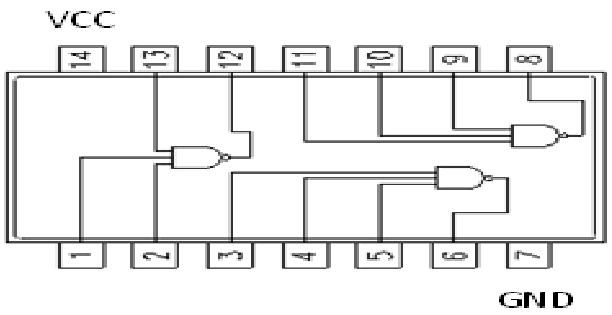
The control inputs to a clocked flip flop will be making a transition at approximately the same times as triggering edge of the clock input occurs. This can lead to unpredictable Triggering.

A JK master flip flop is positive edge triggered, where as slave is negative edge triggered. Therefore master first responds to J and K inputs and then slave. If J=0 and K=1, master resets on arrival of positive clock edge. High output of the master drives the K input of the slave. For the trailing edge of the clock pulse the slave is forced to reset. If both the inputs are high, it changes the state or toggles on the arrival of the positive clock edge and the slave toggles on the negative clock edge. The slave does exactly what the master does.

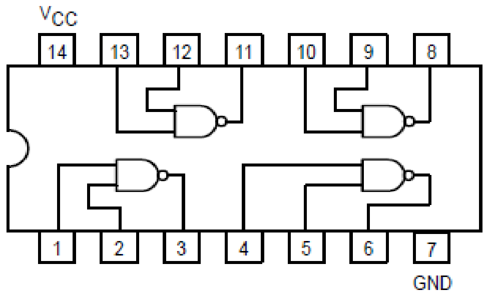
PROCEDURE

1. Verify all components and patch chords whether they are in good condition or not.
2. Make connections as shown in the circuit diagram.
3. Give supply to the trainer kit.
4. Provide input data to the circuit via switches.
5. Verify truth table sequence and observe outputs

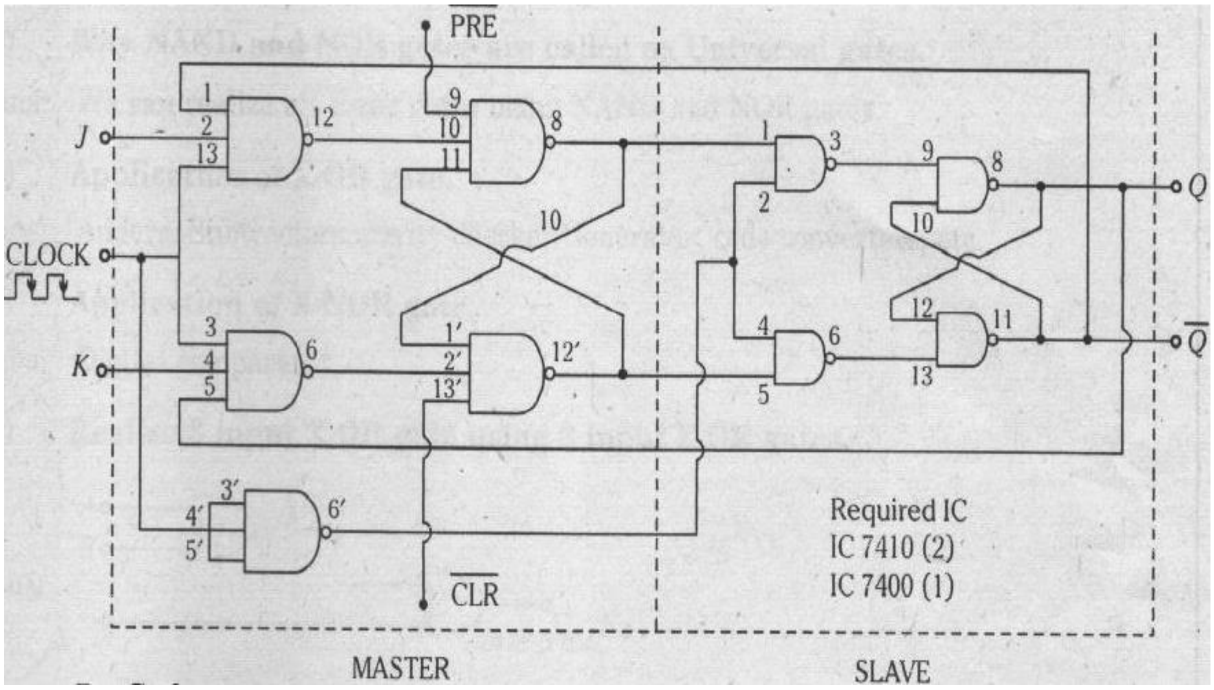
PIN DIAGRAMS



IC 7410 – 3 INPUT NAND GATE



2 INPUT NAND Gate 74LS00



Note: Keep Pre' = 1 and Clr' = 1 for verifying the Truth Table of MSJK FF

TRUTH TABLE

Clock	J	K	Q	Q`	Comment
⊓	0	0	Q_0	Q_0`	No Change
⊓	0	1	0	1	Reset
⊓	1	0	1	0	Set
⊓	1	1	Q_0	Q_0	Toggle

PROCEDURE:

(1) Rig up the circuit as shown in the circuit diagram.

(2) Give power supply to trainer kit

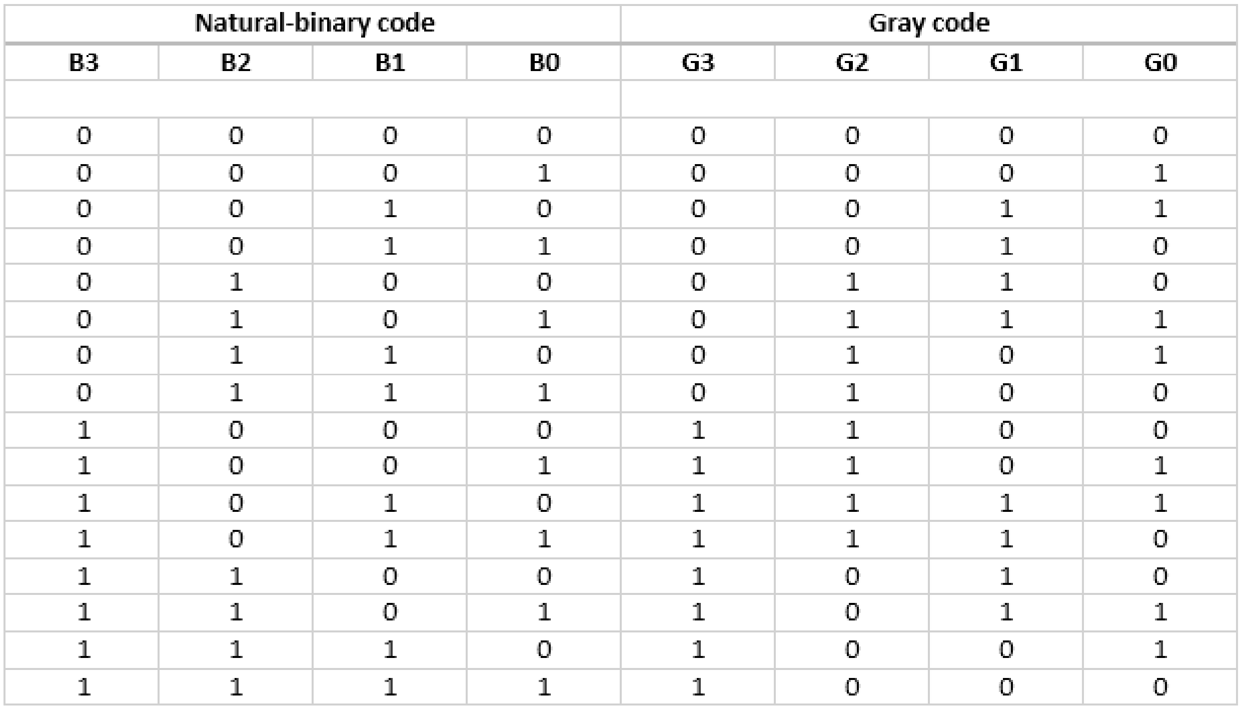
(3) Connect Preset (Pre) and Clear (Clr) inputs to Vcc (logic 1) and then apply J& K inputs to the circuit via logic switches.

(3) Note down the corresponding output and verify it with the given truth table.

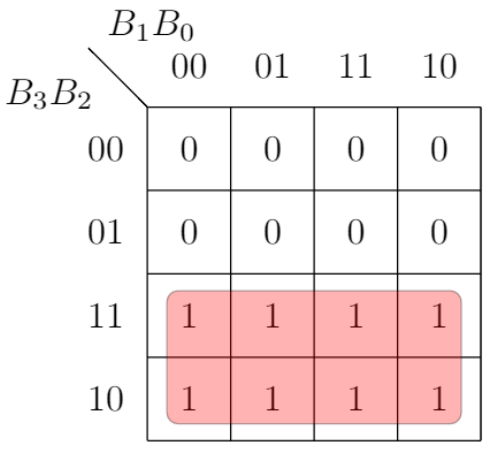
RESULT : Working of Master Slave J K flip-flop is verified

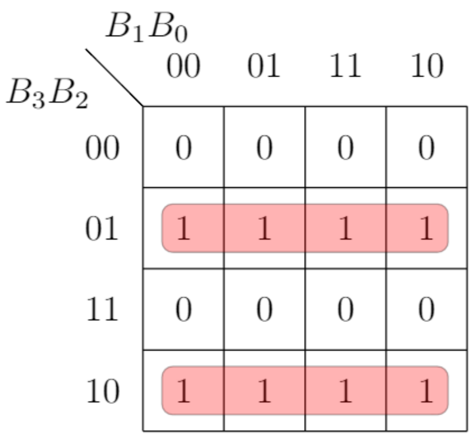
7. Design and implement code converter I) Binary to Gray II) Gray to Binary Code using basic gates.

AIM: To realize Binary to Gray code converter and vic



Natural-binary code				Gray code			
B3	B2	B1	B0	G3	G2	G1	G0
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

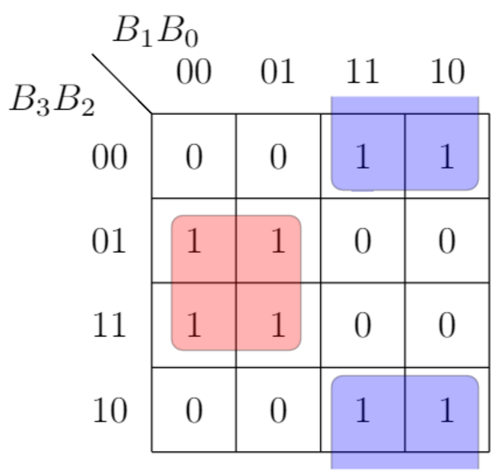


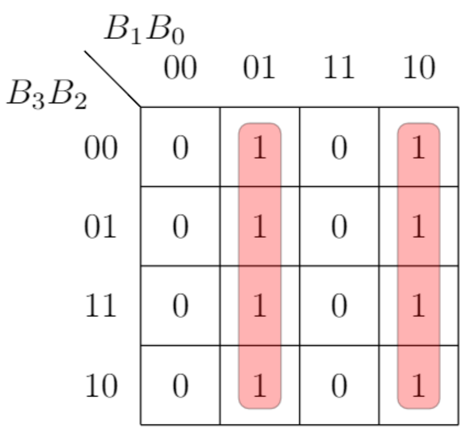


$G_3 = B_3$

$G_2 = \overline{B}_3 B_2 + B_3 \overline{B}_2$

$G_2 = B_3 \oplus B_2$





$$G_1 = \bar{B}_2 B_1 + B_2 \bar{B}_1 \qquad G_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0$$

$$G_1 = B_2 \oplus B_1 \qquad G_0 = B_1 \oplus B_0$$

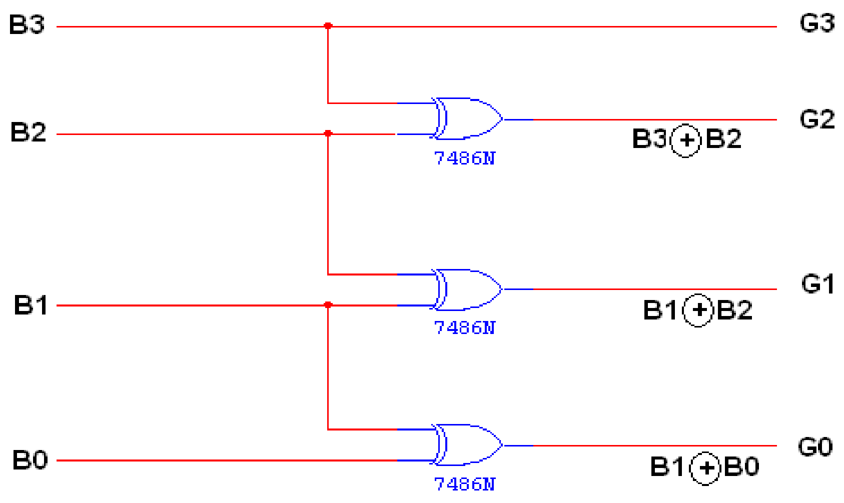
BOOLEAN EXPRESSIONS:

$G_3 = B_3$

$G_2 = B_3 \oplus B2$

$G_1 = B_1 \oplus B_2$

$G_0 = B_1 \oplus B_0$



BINARY TO GRAY CODE USING EX-OR GATES

II) GRAY TO BINARY CONVERSION

Gray code				Natural-binary code			
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	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

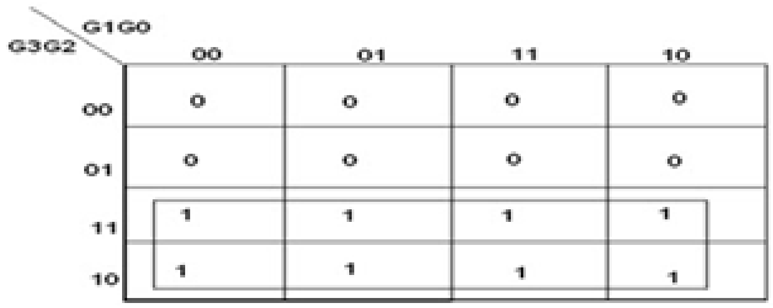


Fig6 (a): K – map for B_3

$B_3 = G_3$

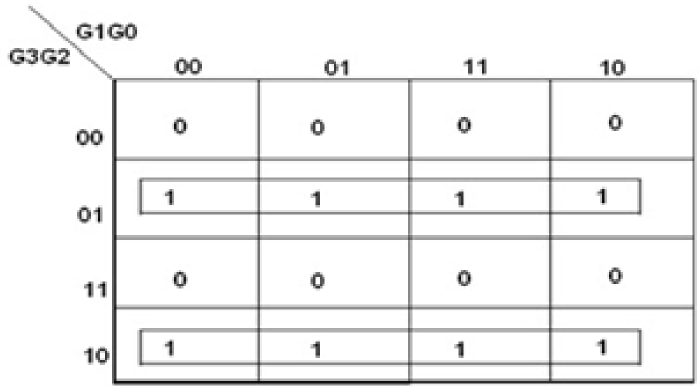
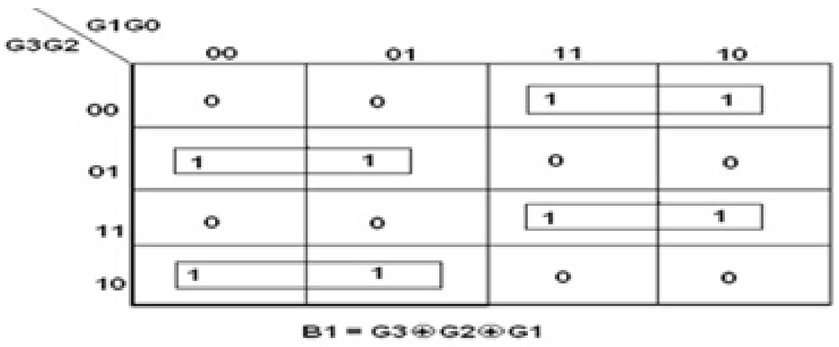


Fig6 (b): K – map for B_2

$B_2 = G_3\ \overline{G}_2 + G2\ \overline{G}_3$

$= G_3 \oplus G_2$



G3G2 \ G1G0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

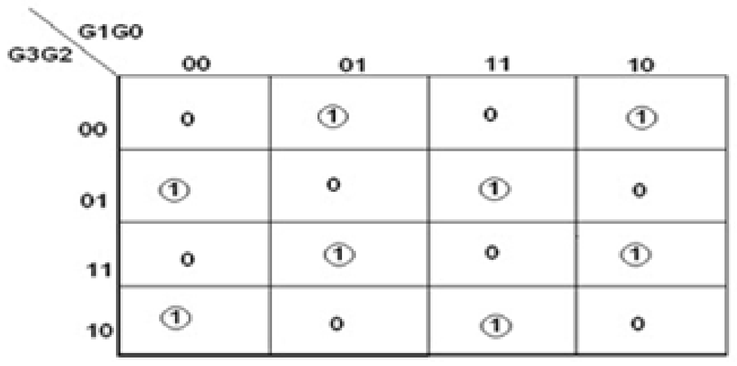
B1 = G3⊕G2⊕G1

Fig6 (c): K – map for B_1

$$B_1 = G_1\overline{G_2G_3} + G_2\overline{G_3G_1} + G_3G_2G_1 + G_3\overline{G_1G_2} = \overline{G_1}(G_3 \oplus G_2) + G_1(\overline{G_3 \oplus G_2})$$

$$= G_1X + G_1X \text{ where } X = G_3 \oplus G_2$$

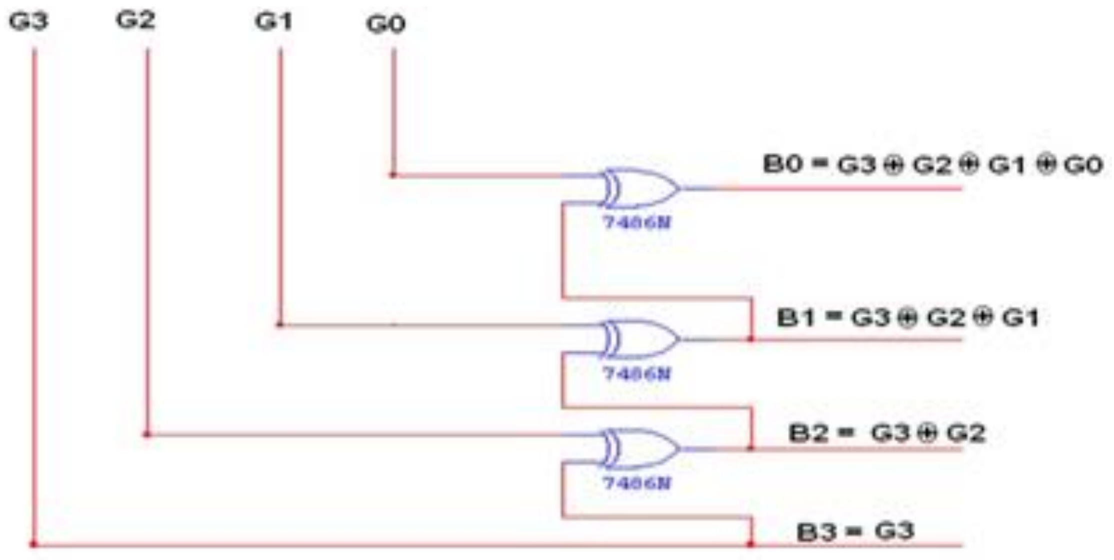
$$= G_1 \oplus X = G_1 \oplus G_2 \oplus G_3$$



G3G2 \ G1G0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$

Fig6 (d): K – map for B_0



GRAY TO BINARY CODE CONVERTER

PROCEDURE:

Check all the components for their working.

Insert the appropriate IC into the IC base.

Make connections as shown in the circuit diagram.

Verify the Truth Table and observe the outputs.

RESULT: Binary to gray code conversion and vice versa is realized using EX-OR gates

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

AIM: Design and implement a mod-n (n<8) synchronous up counter using J-K Flip -Flop ICs.

COMPONENTS REQUIRED: IC7476, IC7408, patch chords, trainer kit

Notes & observation (w.r.t IC7476):

- Each 7476 IC has two J –K flip flops (FFs) since we are dealing with 3 bit counters, 3 FFs are needed. So. two 7476 IC’s are used.
- J and K i/ps of FFs can be connected to logic ‘1’ to operate the FF’s in toggle mode.
- When Pre=’1’, Clr = ‘0’, counter is cleared, $Q_o = Q_1 = Q_2 = 0$
  When Pre=’0’, Clr = ‘1’, counter is preset, $Q_o = Q_1 = Q_2 = 1$

  Keep Pre=’1’, Clr = ‘1’, for normal count mode.

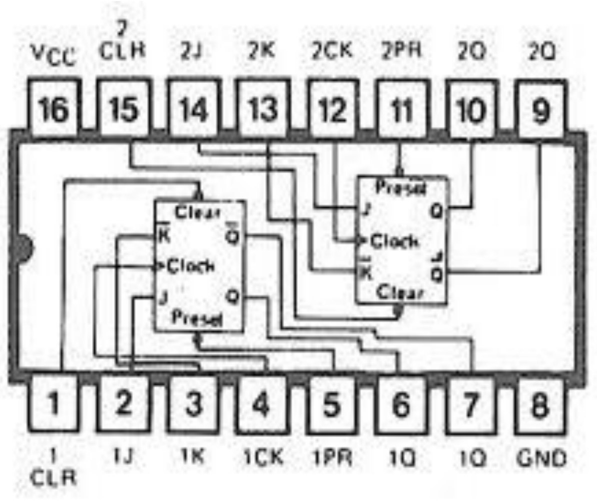
Notes & observation (w.r.t Synchronous Counter):

- In synchronous counter, all the flip-flops are clocked simultaneously.
- The table, which lifts the required input states for a given change of state, is called as **excitation table**.
- The table, which represents the relationship between present state and the next state, is called a **state table**.
- Excitation table is defined for a flip-flop where as State table is defined for a counter.

COMPONENTS REQUIRED:

Digital trainer kit, patch chords, IC7476, IC7432, IC7408, IC7404.

PIN DETAILS - IC 74109(IC7476)



J	K’	Clock	S_D	R_D	Q	Q’
2	3	4	5	1	6	7
14	13	12	11	15	10	9

Pre Clr

SYNCHRONOUS MOD-6 UP COUNTER:

(Say for N=5)

DESIGN STEPS:

COUNT TABLE:

CLK	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0

TRUTH TABLE OF JK FF:

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

Note: $\left\{ \begin{array}{l} Q_{n+1} \text{ is present state} \\ Q_n \text{ is previous state} \end{array} \right\}$ OR $\left\{ \begin{array}{l} Q_{n+1} \text{ is next state} \\ Q_n \text{ is present state} \end{array} \right\}$

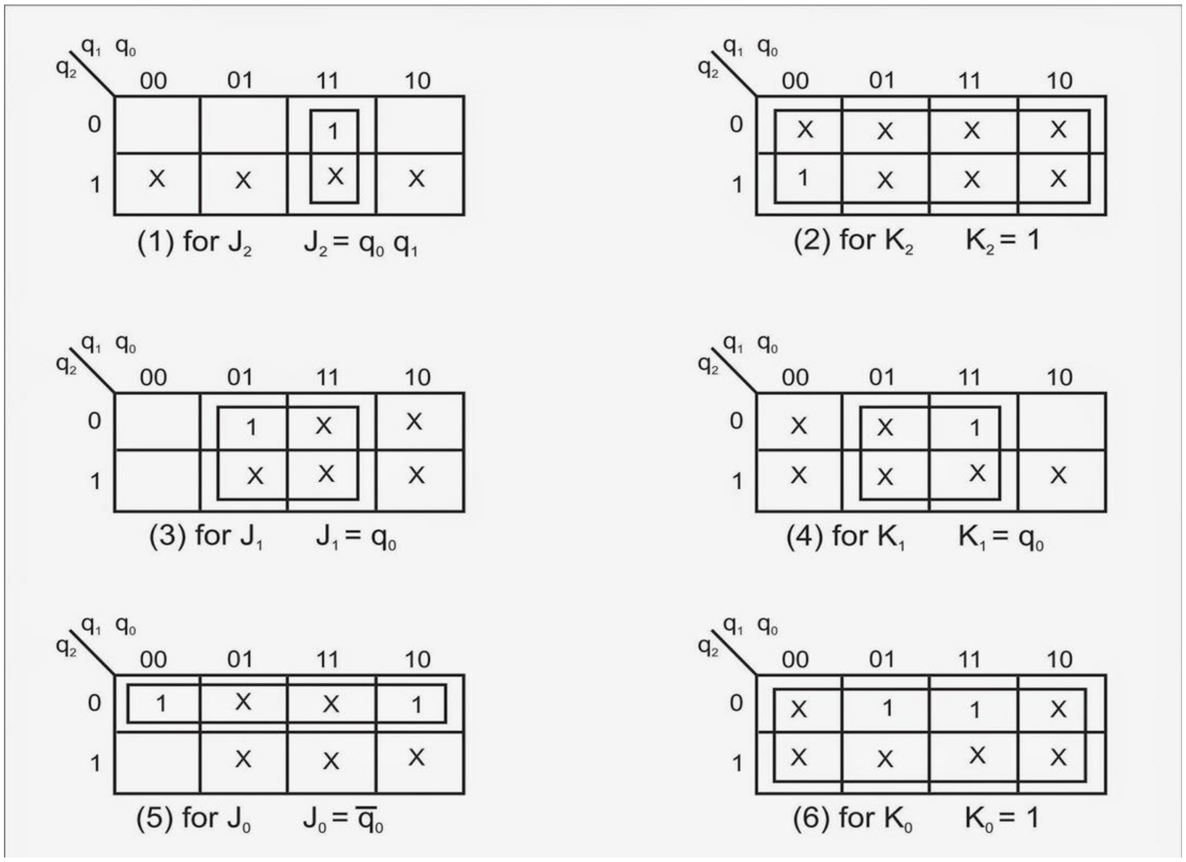
FF EXCITATION TABLE

P.S	N.S	I/Ps	
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

STATE TABLE FOR MOD-6 COUNTER

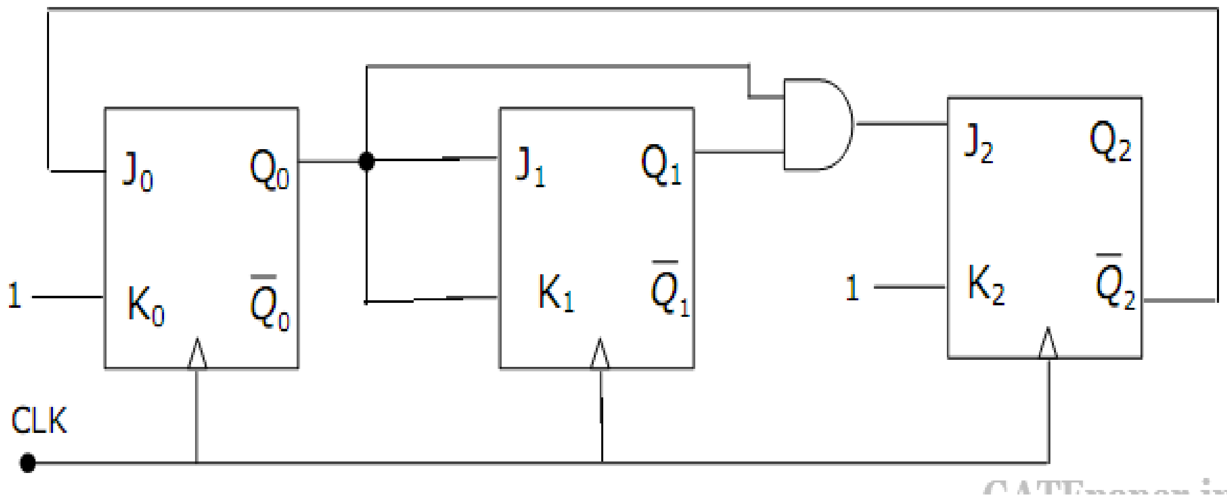
Present State			Next State			Excitation Table		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2 K_2	J_1K_1	J_0K_0
0	0	0	0	0	1	0 x	0 x	1 x
0	0	1	0	1	0	0 x	1 x	x 1
0	1	0	0	1	1	0 x	x 0	1 x
0	1	1	1	0	0	1 x	x 1	x 1
1	0	0	0	0	0	x 1	0 x	0 x
1	0	1	x	x	x	x x	x x	x x
1	1	0	x	x	x	x x	x x	x x
1	1	1	x	x	x	x x	x x	x x

K – MAPS: -


(1) for J_2 $J_2 = q_0 q_1$
(2) for K_2 $K_2 = 1$
(3) for J_1 $J_1 = q_0$
(4) for K_1 $K_1 = q_0$
(5) for J_0 $J_0 = \overline{q_0}$
(6) for K_0 $K_0 = 1$

<u>NOTE: COMPLIMENT ALL K EXPRESSIONS BEFORE REALIZING IN HARDARE CIRCUIT</u>

LOGIC CIRCUIT DIAGRAM FOR MOD 6 UP COUNTER



PROCEDURE:

1. Make connections as shown in logic diagram
2. Provide power supply to trainer kit
3. Keep Preset & Clear inputs at logic level ‘1’ as to operate in count mode.
4. Apply the clock pulses and observe the count sequence as per the count table.

RESULT: Mod-n Synchronous Counter is designed and verified

9. 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: Design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and and test a 7-segment static display system to display numbers 0 to 9.

COMPONENTS REQUIRED: IC7490, patch chords, trainer kit

THEORY

Asynchronous counter is a counter in which the clock signal is connected to the clock input of only first stage flip flop. The clock input of the second stage flip flop is triggered by the output of the first stage flip flop and so on. This introduces an inherent propagation delay time through a flip flop. A transition of input clock pulse and a transition of the output of a flip flop can never occur exactly at the same time. Therefore, the two flip flops are never simultaneously triggered, which results in asynchronous counter operation.

The 74LS90, is 4-bit ripple type Decade Counter. It consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), Each section has a separate clock input which initiates state changes of the counter on the HIGH-to- LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q0 output of each device is designed and specified to drive the rated fan-out plus the CP1 input of the device.

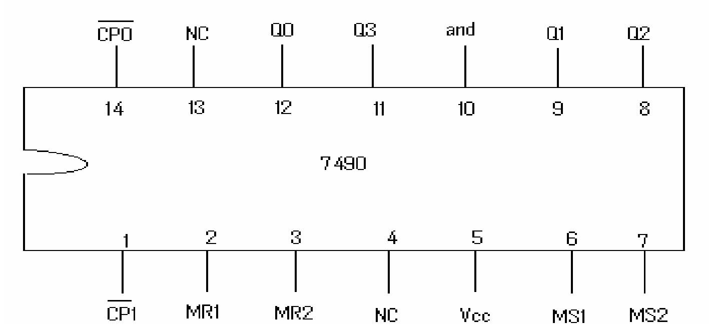
A gated AND asynchronous Master Reset (MR1 MR2) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set (MS1 MS2) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH)

PROCEDURE

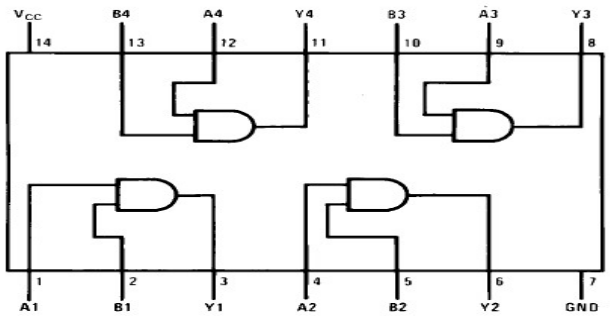
1. Connections are made as shown in the logic diagram.
2. Switch ON the trainer kit power.
3. Apply the clock input (Mono pulse/1Hz clock).

4. When clock pulses are applied, the desired count sequence is observed on output pins Q3, Q2, Q1, and Q0 as illustrated in the truth table
5. **For connecting Mod-7 disconnect the ground terminal of MR1 & MR2 and then connect into the o\p terminal of AND gate.**
6. For other counting sequence (Mod-6 and Mod-4) repeat steps 1 to 5.

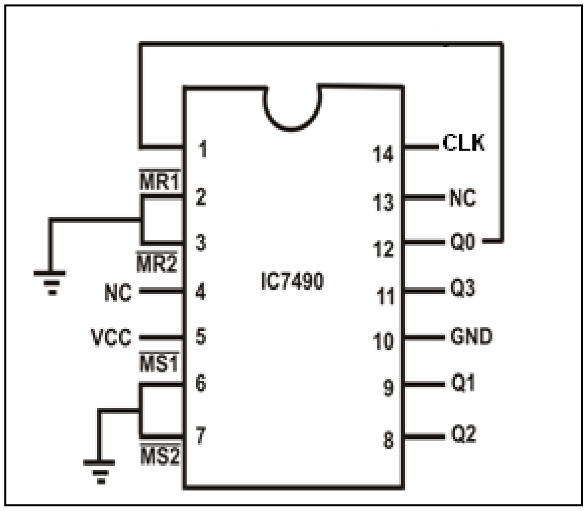
PIN DIAGRAMS



IC 7408 – AND GATE



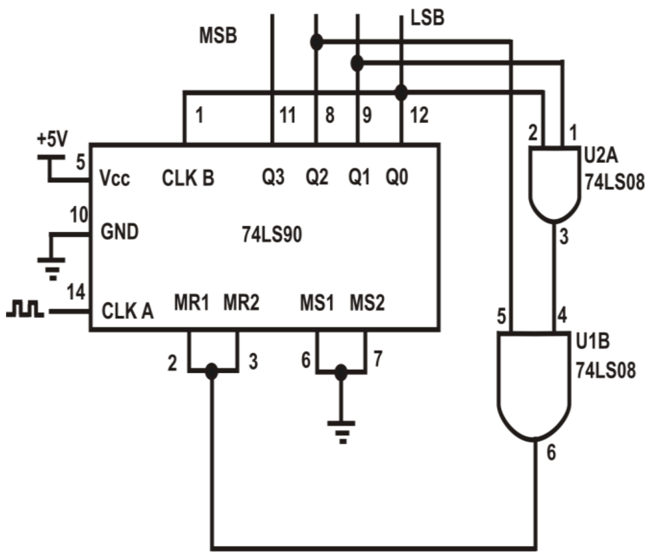
Logic Diagram of Decade counter



Truth Table

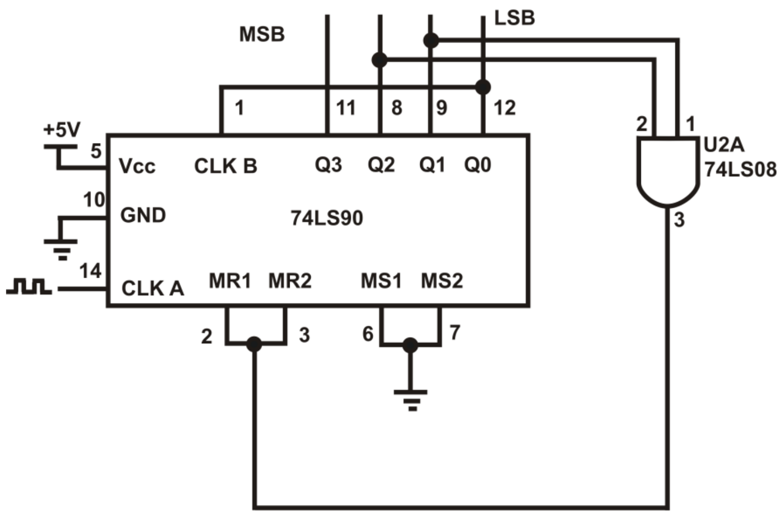
Clock pulse	Outputs			
	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0

Logic Diagram of MOD-7 Counter



Clock pulse	Outputs			
	Q3	Q2	Q1	Q0
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	Sequence repeats			

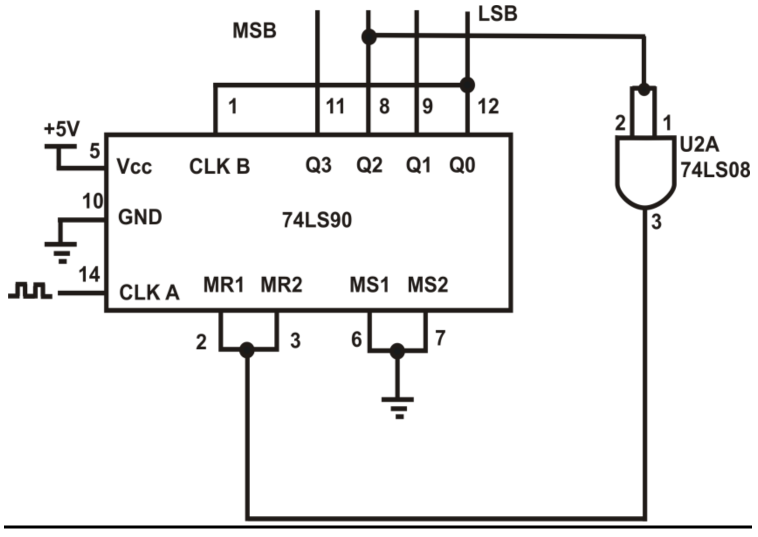
Logic Diagram of MOD-6 Counter Truth Table



Clock pulse	Outputs			
	Q3	Q2	Q1	Q0
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	Sequence repeats			

Logic Diagram of MOD-4 Counter **Truth Table**

Clock pulse	Outputs			
	Q3	Q2	Q1	Q0
0	0	0	0	0



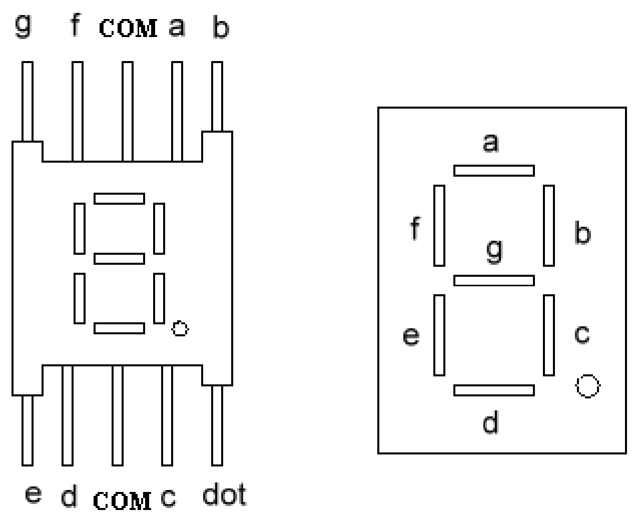
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	0	0	0
5	Sequence repeats			

BCD TO 7-SEGMENT DECODER

THEORY:

The Light Emitting Diode (LED) finds its place in many applications in these modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in “Eight” (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.

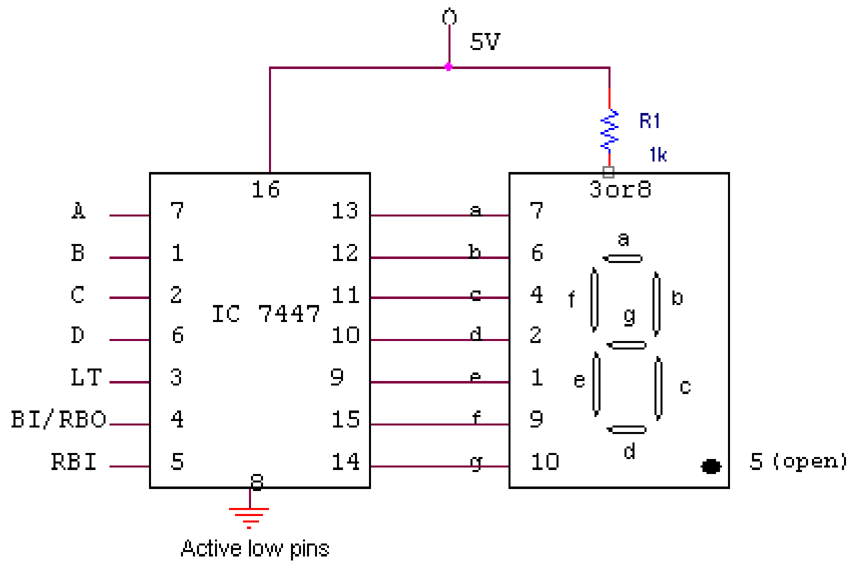
The Light Emitting Diode (LED), finds its place in many applications in this modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in “Eight” (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.



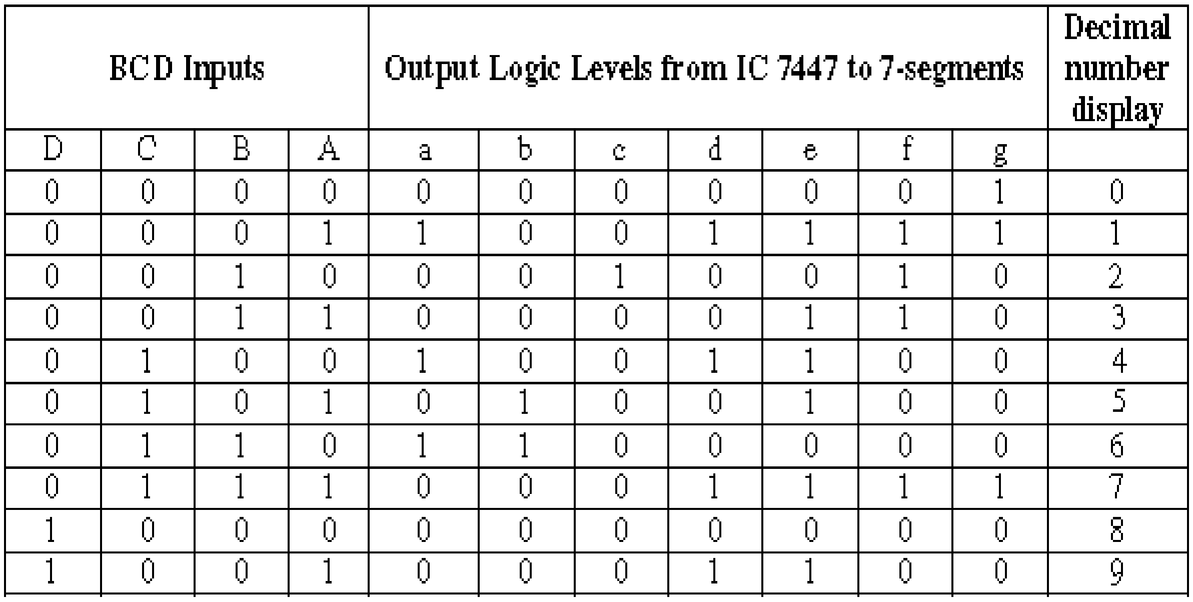
Seven-Segment Display

A decoder is a combinational circuit that connects the binary information from ‘n’ input lines to a maximum of 2n unique output lines. The IC7447 is a BCD to 7-segment pattern converter. The IC7447 takes the Binary Coded Decimal (BCD) as the input and outputs the relevant 7 segment code.

CIRCUIT DIAGRAM:



TRUTH TABLE:



BCD Inputs				Output Logic Levels from IC 7447 to 7-segments							Decimal number display
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	1	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	1	1	0	0	9

RESULT: Thus asynchronous counter was designed using decade counter for different counting sequence with necessary reset logic and their respective truth tables are verified