

**SRI VENKATESWARA COLLEGE
OF
ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)**

**Recognized by AICTE, NBA, NAAC and
Govt. of A.P. Affiliated by J.N.T.U.A., ANANTAPUR**

R.V.S. Nagar, Tirupati Road, CHITTOOR- 517127



**DEPARTMENT
OF
ELECTRONICS & COMMUNICATION ENGINEERING**

IC APPLICATIONS & ECAD LAB

Name: _____

Roll No: _____

Class: _____

Branch: _____

14AEC23 IC APPLICATIONS & ECAD LAB

LIST OF EXPERIMENTS

PART A (IC APPLICATIONS LAB)

1. OP AMP Applications – Adder, Subtractor and Comparator Circuits.
2. Active Filter Applications – LPF, HPF (first order).
3. Function Generator using OP AMPs.
4. IC 555 Timer – Monostable and Astable Operation of Circuit
5. Voltage Regulator using IC 723.
6. 4 bit DAC using OP AMP.

PART B (ECAD LAB)

1. 3-8 Decoder -74X138 & 8-3 Encoder- 74X148.
2. 8 x 1 Multiplexer -74X151 and 2x4 Demultiplexer-74X155.
3. 4 bit Comparator-74X85.
4. D Flip-Flop 74X74.
5. JK Flip-Flop 74X109.
6. Decade counter-74X90.

INDEX**IC APPLICATIONS LAB**

| S.NO | DATE | NAME OF THE EXPERIMENT | FACULTY SIGNATURE |
|-------------|-------------|--|--------------------------|
| 1 | | Op amp applications – adder, subtractor and comparator circuits. | |
| 2 | | Active filter applications – LPF, HPF (first order). | |
| 3 | | Function generator using op amps. | |
| 4 | | IC 555 timer – monostable and astable operation of circuit | |
| 5 | | Voltage regulator using IC 723 | |
| 6 | | 4 bit DAC using op amp | |

ECAD LAB

| S.NO | DATE | NAME OF THE EXPERIMENT | FACULTY SIGNATURE |
|-------------|-------------|--|--------------------------|
| 7 | | 3-8 decoder – IC 74X138 & 8-3 priority encoder– IC 74X148 | |
| 8 | | 8 X 1 multiplexer – IC 74X151 & 2 X 4 demultiplexer– IC 74X155 | |
| 9 | | 4-bit comparator – IC 74X85 | |
| 10 | | D-flip flop – IC 74X74 | |
| 11 | | JK-flip flop – IC 74X109 | |
| 12 | | decade counter – IC 74X90 | |

OP- AMP APPLICATIONS – ADDER, SUBTRACTOR AND COMPARATOR CIRCUITS.

EXP NO:

DATE:

OP-AMP APPLICATIONS-ADDER AND SUBTRACTOR

AIM:

To demonstrate the operation of an inverting summing amplifier.

APPARATUS REQUIRED:

1. IC 741
2. connecting wires
3. Fixed power supply(+15V and -15V)
4. breadboard
5. resistors
6. R.P.S

THEORY:

ADDER:

Op-amp may be used to design a circuit whose output is sum of several input signals such a circuit is know as adder, it can be used as inverting amplifier or non inverting amplifier.

In the above we are using inverting amplifier by using applying analysis we get.

$$V_1/R_1 + V_2/R_2 = -V_0/R_F \quad \text{since } V_A=0$$

$$V_0 = -R_f (V_1/R_1 + V_2/R_2)$$

But R_1 , R_2 and R_F have same resistance values.

$$V_0 = - (V_1 + V_2)$$

SUBTRACTOR:

A basic differential amplifier can be used as subtractor.

By applying model equation at A we get

$$(V_2 - V_A)/R_1 + (V_A - V_0)/R_F = 0$$

$$(V_B - V_1)/R_1 + V_B/R_2 = 0$$

As resistance value are same

$$V_0 = V_1 - V_2$$

PROCEDURE:

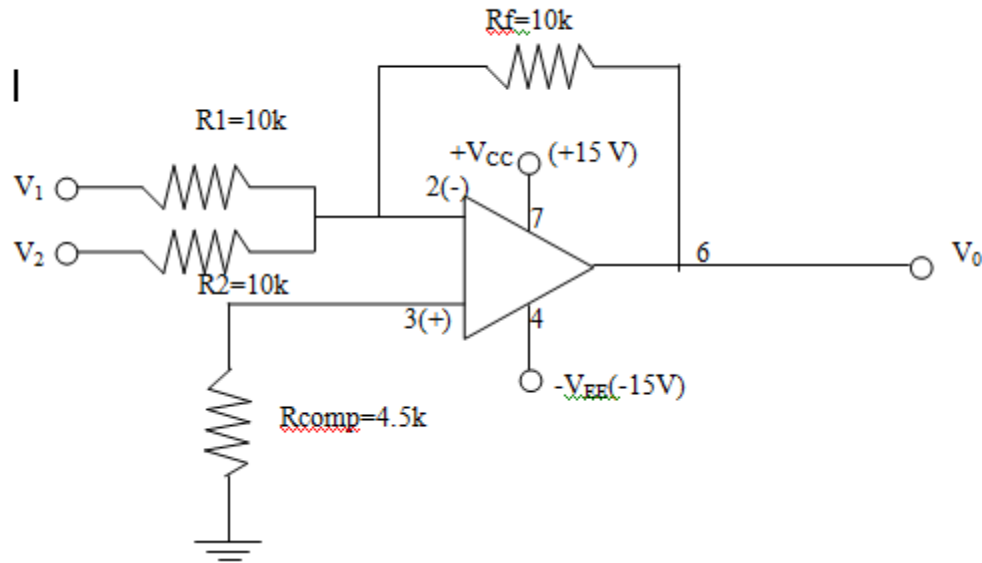
Adder:

1. Connections are made as per the diagram.
2. Switch on power supply.
3. Observe the output in DMM and calculate the voltage is nothing but the sum of the applied voltages.
4. Observe that the output voltage is nothing but the sum of the two applied voltages.
5. The output voltage using DMM was measured $V_0 = [(R_f/R_1) V_1 + (R_f/R_2) V_2]$
6. The phase of output voltage with respect to input was noted using CRO.
7. The output has observed for different configuration of input.

Subtractor:

1. The circuit was connected as shown.
2. Apply different input voltages for V1 and V2 using RPS.
3. The output voltage using DMM was measured $V_0 = V_1 - V_2$.
4. The output was observed by applying different input voltages.

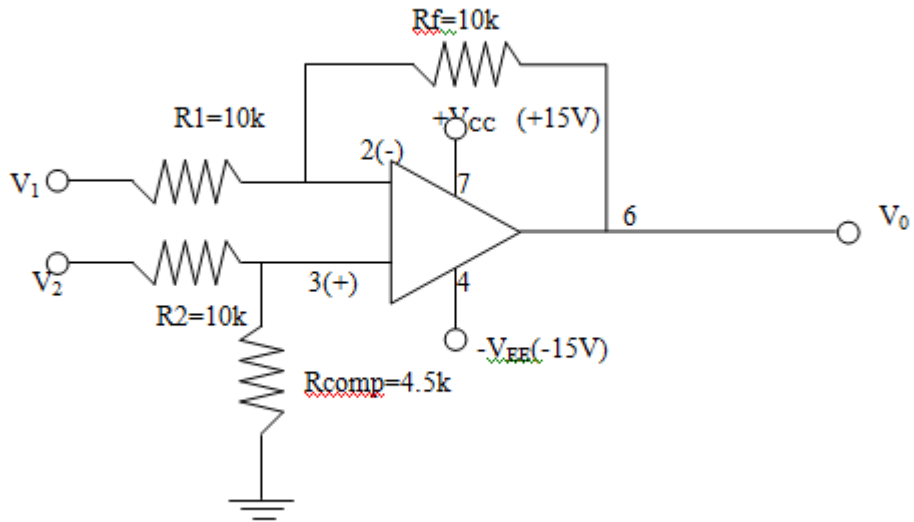
ADDER:



ADDER:

| S.No | V1(V) | V2(V) | $V_0 = -(V_1 + V_2)$ |
|------|-------|-------|----------------------|
| | | | |
| | | | |
| | | | |
| | | | |

SUBTRACTOR:



SUBTRACTOR:

| S.No | V1(V) | V2(V) | $V_0 = -(V_1 - V_2)$ |
|------|-------|-------|----------------------|
| | | | |
| | | | |
| | | | |
| | | | |

RESULT:

B) OP-AMP AS COMPARATOR

AIM:

To design a comparator circuit and to demonstrate the operation.

APPARATUS REQUIRED:

1. IC 741
2. Resistance
3. Fixed power supply(+15V and -15V)
4. Function generator
5. CRO
6. RPS

THEORY:

When Op-Amp is used as a comparator no feedback is used to the circuit operates with full gain.

Comparator is a device, which compares the signal voltage with the applied reference voltage at one of its input terminal. Depending on which of the two voltages are greater that output is either positive or negative.

It $V_{IN} > V_{REF}$ the output voltage V_0 is positive i.e $V_0 = +V_{SAT} < +V_{CC}$

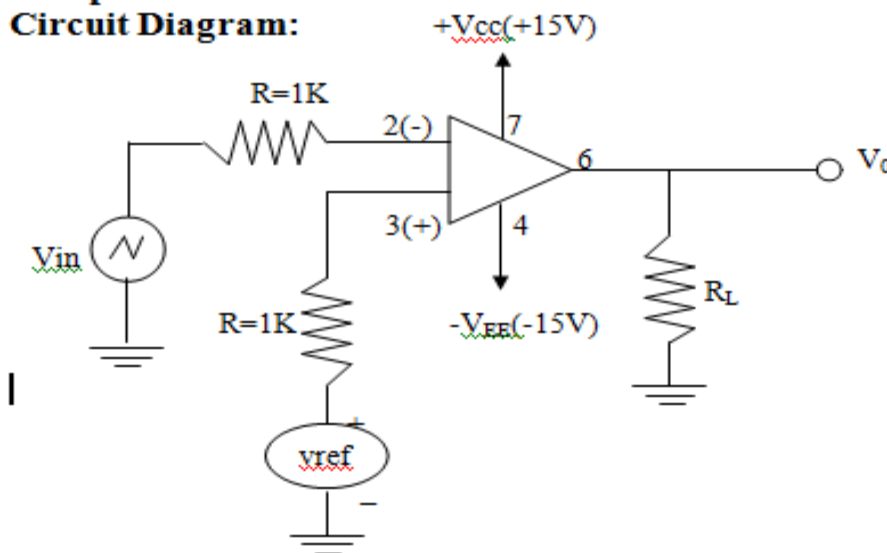
It $V_{IN} < V_{REF}$ the output voltage V_0 is negative i.e. $V_0 = -V_{SAT} > -V_{EE}$

PROCEDURE:

1. Connections are given as per circuit diagram.
2. Adjust function generator to producer wave of 1 KHZ.
3. V_{ref} (+ve and -ve) is applied from fixed DC power supply or RPS.
4. The input and output waveforms was observed using CRO.

Comparator:

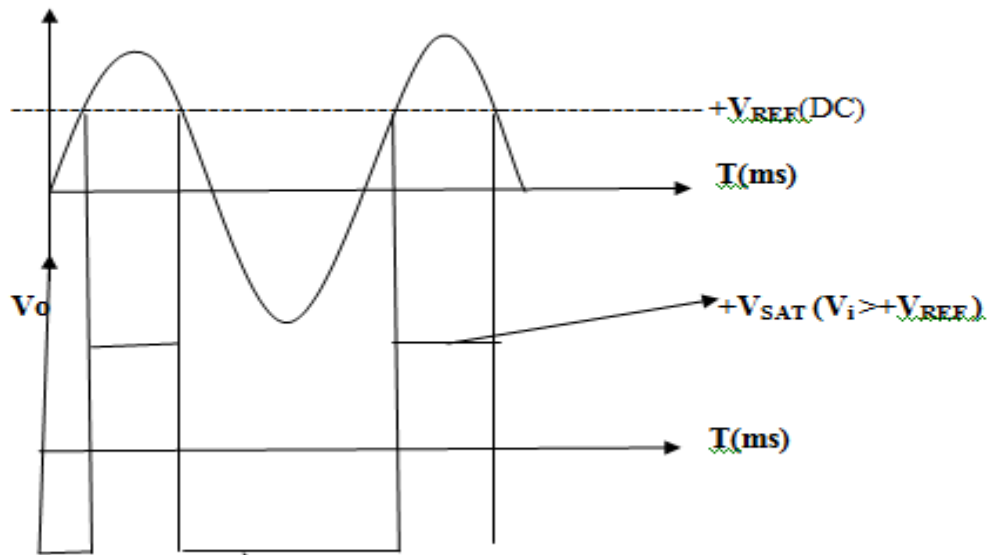
Circuit Diagram:



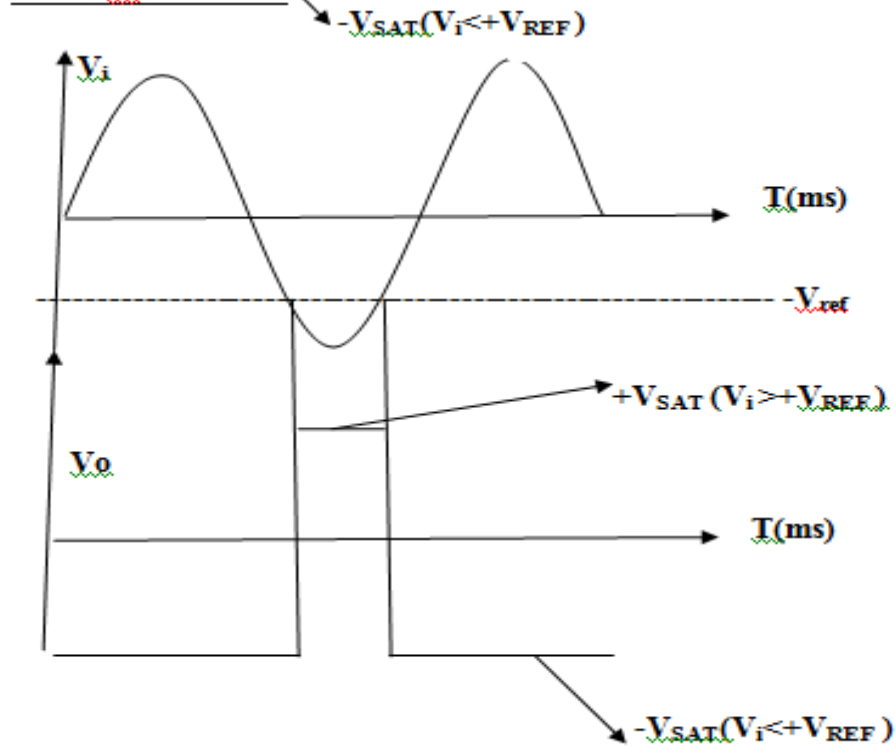
INPUT AND OUTPUT WAVE FORMS:

With +ve reference:

Input(V)



With -ve reference:



RESULT:

ACTIVE FILTER APPLICATIONS-LPF&HPF (FIRST ORDER)

EXP NO:

DATE:

ACTIVE FILTER APPLICATIONS-LPF&HPF (FIRST ORDER)

AIM:

To design and verify the functions of LPF & HPF of first order using Op-amp IC 741.

APPARATUS:-

1. Resistors
2. Capacitors
3. Fixed power supply(+15V and -15V)
4. DRB
5. CRO& probes
6. Function generator
7. Bread board
8. IC 741
9. Connecting wires

THEORY:

As their name implies, **Active Filters** contain active components such as operational amplifiers, transistors or FET's within their circuit design. An active filter generally uses an operational amplifier (op-amp) within its design and Op-amp has a high input impedance, a low output impedance and a voltage gain determined by the resistor network within its feedback loop.

LPF:

The most common and easily understood active filter is the **Active Low Pass Filter**. Its principle of operation and frequency response is exactly the same as those for the previously seen passive filter, the only difference this time is that it uses an op-amp for amplification and gain control. The simplest form of a low pass active filter is to connect an inverting or non-inverting amplifier.

A low-pass filter has a constant gain ($=V_{out}/V_{in}$) from 0 Hz to a high cut off frequency f_H . This cut off frequency is defined as the frequency where the voltage gain is reduced to 0.707, that is at f_H the gain is down by 3 dB; after that ($f > f_H$) it decreases as f increases. The frequencies between 0 Hz and f_H are called pass band frequencies, whereas the frequencies beyond f_H are the so-called stop band frequencies.

HPF:

The high-pass filter is formed by interchanging the resistor and capacitor in the low-pass filter that you made the rest of the circuit is the same. For these first-order low-pass and high-pass filters, the gain rolls off at the rate of about 20dB/decade in the stop band. A high-pass filter has a stop band for $0 < f < f_L$ and where f_L is the low cut off frequency. A common use for a high-pass filter is to remove the dc component of a signal for which you are only interested in the ac components (such as an audio signal). A bandpass filter has a pass band between two cut off frequencies f_H and f_L , ($f_H > f_L$), and two stop bands $0 < f < f_L$ and $f > f_H$. The bandwidth of a band pass filter is equal to $f_H - f_L$.

PROCEDURE:-

FOR LPF&HPF:

1. Connect the circuit as shown in circuit diagram.
2. Set power supply voltage to +/-15v
3. Set i/p amplitude to 1v/2v and vary i/p freq in steps.
4. Note the o/p amplitude and frequency and calculate gain and convert into decibels.
5. Plot the graph b/w frequency and gain.

DESIGN OF FIRST ORDER LPF AND HPF:

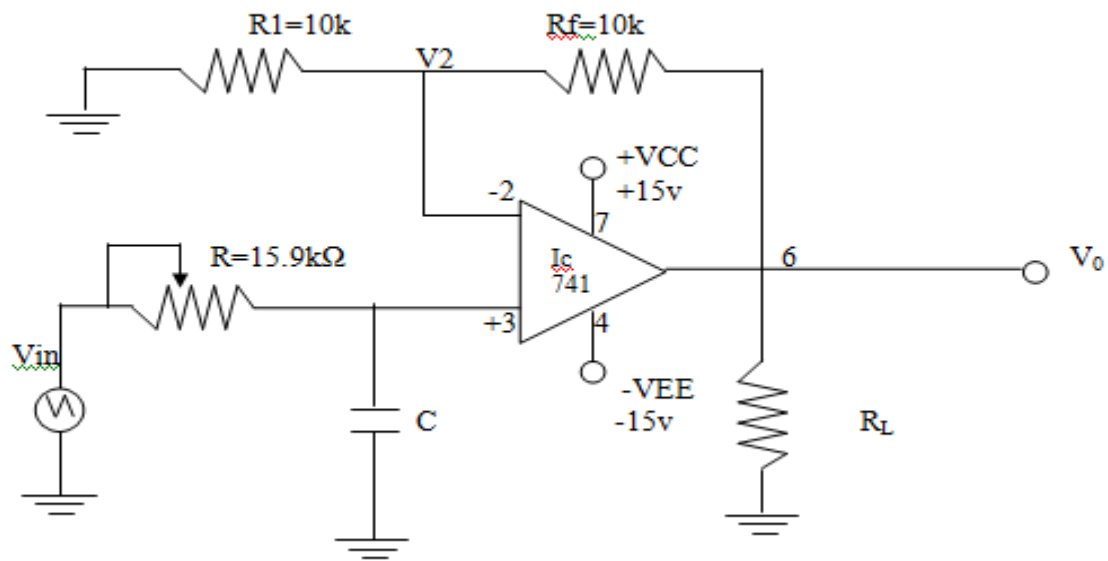
LPF DESIGN:

$F_H = 1 \text{ KHZ}$

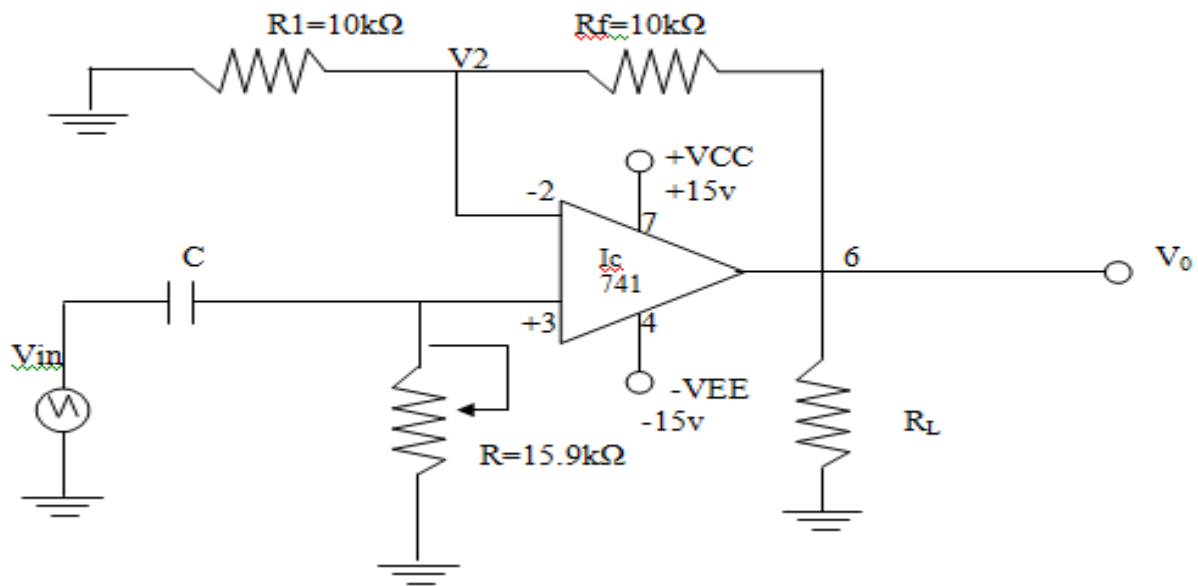
HPF DESIGN:

$F_L = 1 \text{ KHZ}$

**Low pass filter:
Circuit diagrams:**

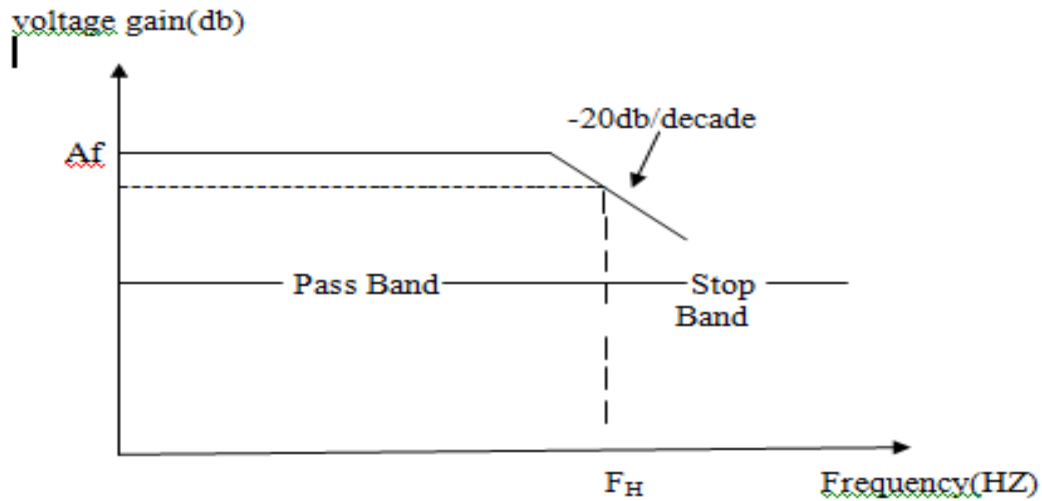


High pass filter:

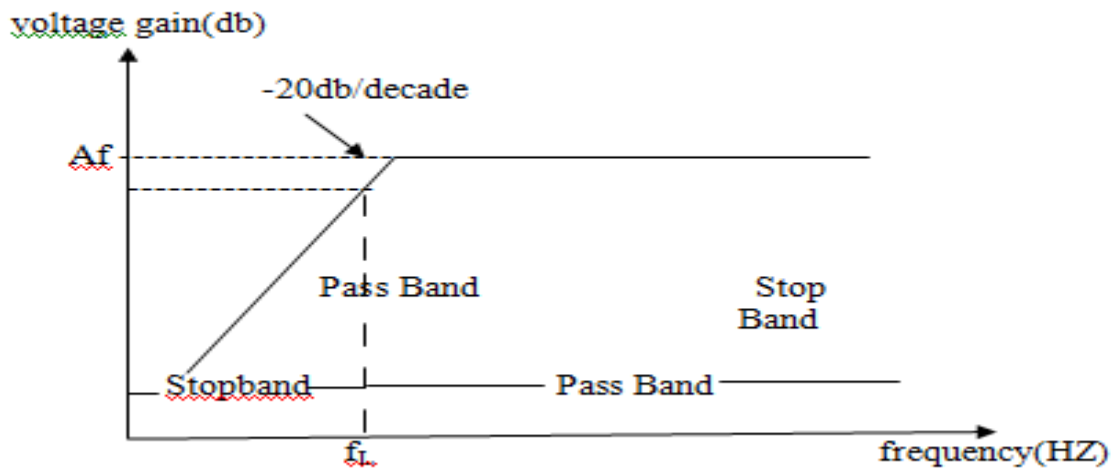


Frequency response:

Low pass filter:



High passFilter



LOW PASS FILTER:

| S.No | Frequency | o/p Voltage Vo in Volts | Vo/Vin | Gain =20 log Vo/Vin |
|------|-----------|----------------------------|--------|-------------------------|
| | | | | |

HIGH PASS FILTER:

| S.No | Frequency | o/p Voltage Vo in Volts | Vo/Vin | Gain =20 log Vo/Vin |
|------|-----------|----------------------------|--------|-------------------------|
| | | | | |

RESULT:

SQUARE WAVE GENERATOR

EXP NO:

DATE:

SQUARE WAVE GENERATOR

AIM: To design a square wave generator circuit for the frequency of Oscillations of 1KHZ

APPARATUS:

| | |
|---|-----------|
| 1 | OP-AMP |
| 2 | RESISTOR |
| 3 | CAPACITOR |
| 4 | CRO |
| 5 | RPS |

DESIGN:

$$F=1\text{KHZ} = T=1\text{ms}$$

$$R2 \text{ _____}, C = \text{ _____ } \mu\text{F}$$

$$R1 = 1.16R2$$

$$T = 2RC$$

$$R = T/2C =$$

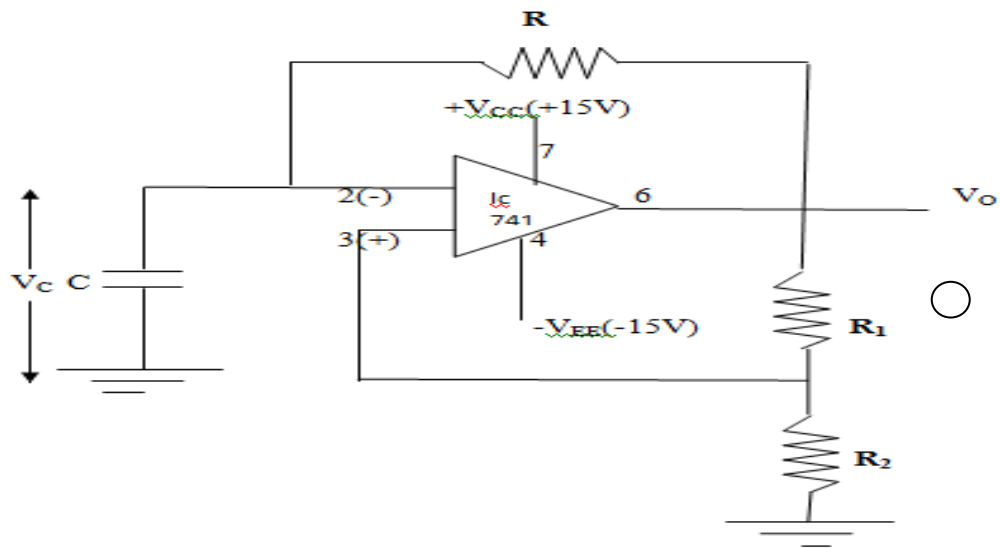
THEORY:

A simple op-Amp square wave generator is also called as free running oscillator, the principle of generation of square wave output is to force an op-amp to operate in the saturation region . A fraction $\beta = R2/(R1+R2)$ of the output is fed back to the (+) input terminal. The output is also fed to the (-) terminal after integrating by means of a low pass Rc combination in astablemultivibrator both the states are quasistables. The frequency is determined by the time taken by the capacitor to charge from- βV_{sat} to $+\beta V_{\text{sat}}$.

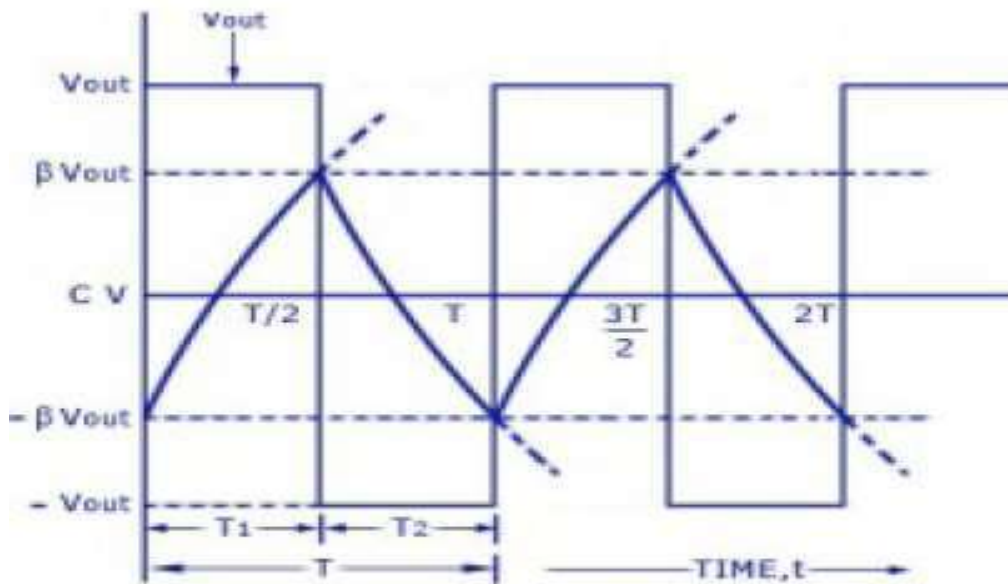
PROCEDURE:

1. The connection is given as per the circuit diagram.
2. Connect the CRO in the output and trace the square waveform.
3. Calculate the practical frequency and compare with the theoretical Frequency.
4. Plot the waveform obtained and mark the frequency and time period.

CIRCUIT DIAGRAM:



OUTPUT WAVE FORMS:



Output and Capacitor Voltage Waveforms

RESULT:

IC 555 TIMER – MONOSTABLE AND ASTABLE OPERATION OF CIRCUIT

EXP NO:

DATE:

MONOSTABLE MULTIVIBRATOR USING 555

AIM:

To design a Monostablemultivibrator using 555 IC Timer to produce a pulse width
Of -----.

APPARATUS:

1. 555 IC.
2. Resistors and capacitors according to design.
3. Fixed power supply(+5V)
4. Connecting wires.
5. C.R.O and function generator.

THEROY:

Monostablemultivibrator has one state and the other state is quasi stable state. The circuit is useful for generating signal output pulse of adjustable time duration in response to a triggering signal. The width of the output pulse depends only on external components connected to IC.

DESIGN:

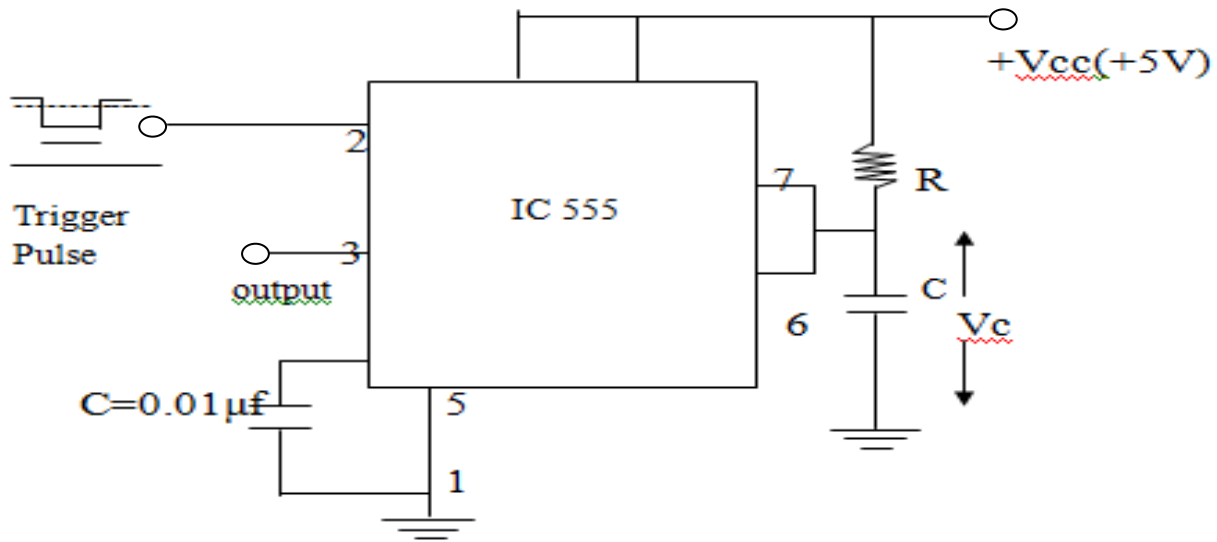
$$V_C = V_{CC} (1 - e^{-t/RC})$$

Width of the pulse $T = 1.1RC$ sec.

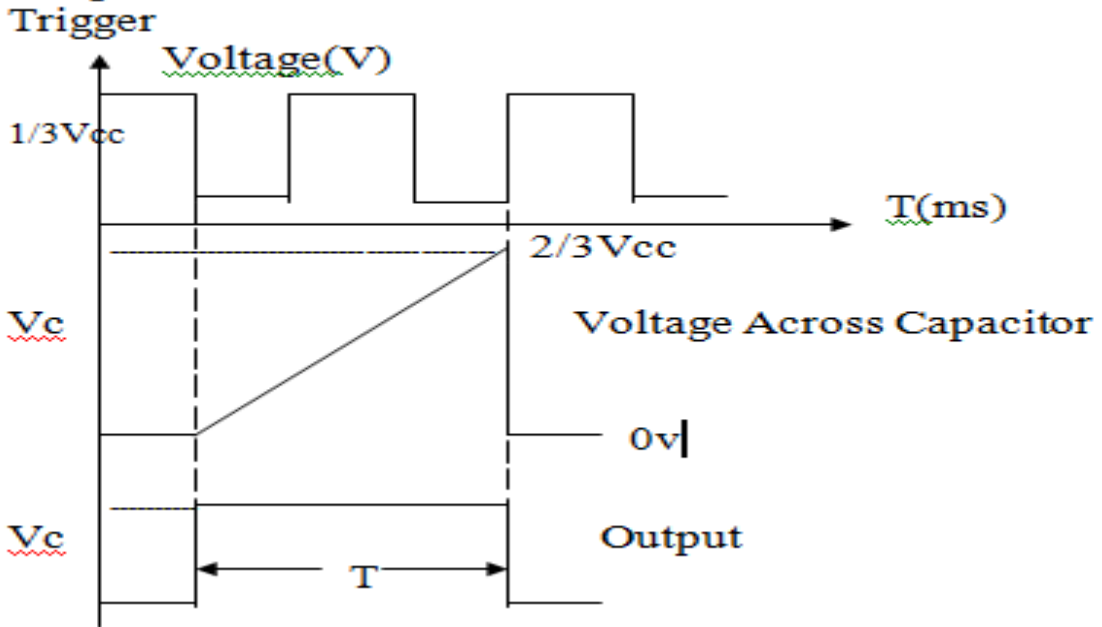
PROCEDURE:

1. Study the pin diagram of 555 IC Timer carefully.
2. Connect the circuit carefully with out causing damage to the IC.
3. Apply Triggering pulse from FG to PIN NO:2
4. Switch on the power supply and observe the wave forms across capacitor and at pin no.3 from CRO.
5. Tabulate all the readings and plot graph for V_c , Trigger pulse & output.

Monostable using 555 timer:
Circuit diagram:



Output waveforms:



RESULT:

$T = \dots\dots\dots$ Duty cycle $D = (t_{low}/T) = \dots\dots\dots$, O/P Frequency = $\dots\dots\dots$

ASTABLE MULTIVIBRATOR USING 555

AIM:

To design a Monostablemultivibrator using 555 IC Timer to produce a pulse width
Of ----- and duty cycle _____%

APPARATUS:

1. 555 IC.
2. Resistors and capacitors according to design.
3. Connecting wires.
4. C.R.O and function generator.
- 5.Fixed power supply(+5V)

THEROY:

Monostablemultivibrator has one state and the other state is quasi stable state. The circuit is useful for generating signal output pulse of adjustable time duration in response to a triggering signal. The width of the output pulse depends only on external components connected to IC.

$$\text{Total Time } T = 0.69(R_A + R_B) C$$

$$\text{Duty cycle } \%D = (R_A + R_B) / (R_A + 2R_B) * 100$$

DESIGN:

$$V_c = V_{cc} (1 - e^{-t/RC})$$

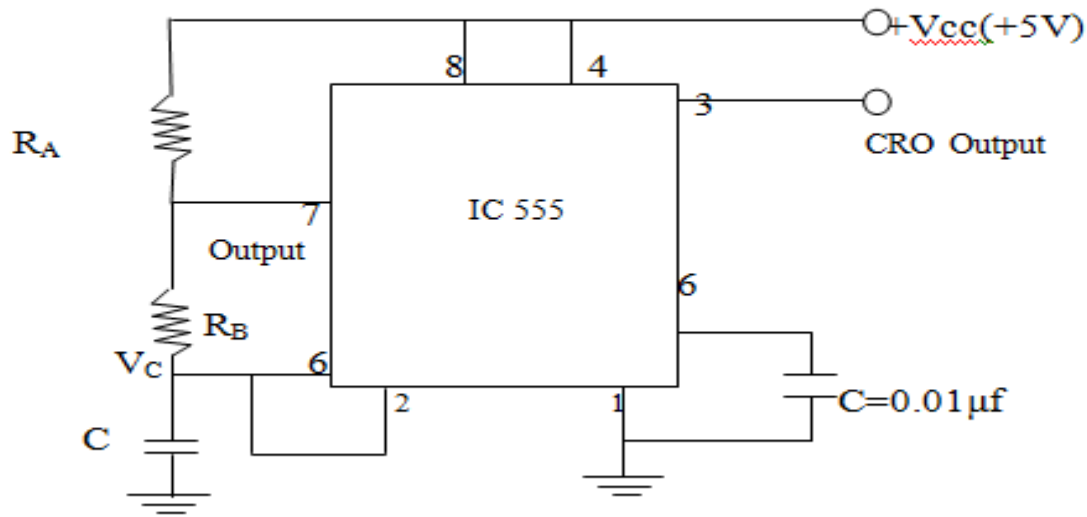
$$\text{Total Time } T = 0.69(R_A + R_B) C$$

$$\text{Duty cycle } \%D = (R_A + R_B) / (R_A + 2R_B) * 100$$

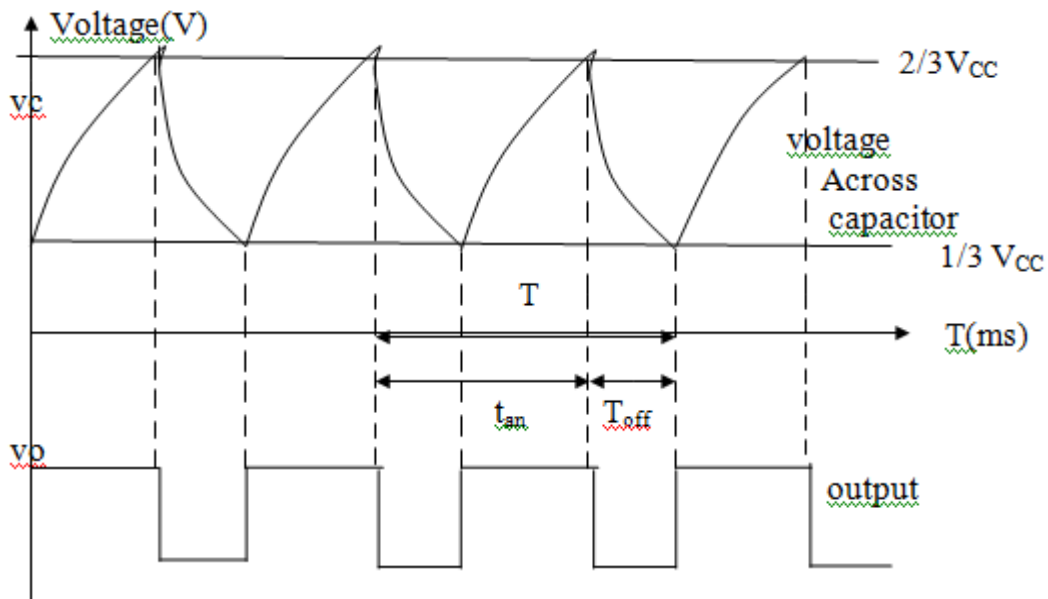
PROCEDURE:

1. Study the pin diagram of 555 IC Timer carefully.
2. Connect the circuit carefully with out causing damage to the IC.
3. Switch on the power supply and observe the capacitor and output wave forms from CRO.
4. Tabulate all the readings and plot graph for V_c , Trigger pulse & output.

Astable circuit using 555 timer:



Output waveforms:



RESULT:

T = ----- Duty cycle $D = (t_{on}/T) =$ -----, Free running Frequency = -----

VOLTAGE REGULATOR USING IC 723

EXP NO:

DATE:

VOLTAGE REGULATOR USING IC 723

AIM:

To determine and obtain the characteristics of voltage regulator using IC 723

APPARATUS:

1. IC 723
2. DMM
3. Connecting wires.
4. RPS
5. Bread board
6. Resistors-2.7K, 5.6K, 1K, 2.2K
7. Capacitors-0.001 μ f, 0.1 μ f
8. DRB

THEROY:

A voltage regulator is circuit that supplies constant voltage regardless of the changes in load current. Except for switching regulators; all other types of regulators are referred as linear regulators. The functional diagram of the voltage regulator is shown below. It consists of a voltage reference source (Pin 6), an error amplifier with its inverting input on pin 4 and non-inverting input on pin 5, a series pass transistor (pins 10 and 11), and a current limiting transistor on pins 2 and 3. The device can be set to work as both positive and negative voltage regulators with an output voltage ranging from 2 V to 37 V, and output current levels upto 150 m A. The maximum supply voltage is 40 V, and the line and load regulations are each specified as 0.01%.

PROCEDURE:

Line regulation:

1. Connect the circuit as per the diagram.
2. Initially disconnect the load resistance and note down the o/p voltage which is the no load voltage V_{NL}
3. Now, at constant load resistance, note down the o/p the voltage by varying the i/p voltage.
4. Tabulate the reading

Load regulation:

1. Connect the circuit as per the diagram.
2. Initially disconnect the load resistance and note down the o/p voltage which is the no load voltage V_{NL}
3. Now, at constant i/p voltage, note down the o/p the voltage by varying the load resistance.
4. Tabulate the readings.

CIRCUIT DIAGRAMS:

HIGH VOLTAGE REGULATOR:

MODEL GRAPHS:

LINE REGULATION:

LOAD REGULATION:

LOW VOLTAGE REGULATOR:

MODEL GRAPHS:
LINE REGULATION:

LOAD REGULATION:

**TABULAR COLUMNS:
HIGH VOLTAGE REGULATOR:
LINE REGULATION:**

LOAD REGULATION:

**HIGH VOLTAGE REGULATOR:
LINE REGULATION:**

LOAD REGULATION:

RESULT:

DIGITAL TO ANALOG CONVERTER (DAC)

EXP NO:

DATE:

DIGITAL TO ANALOG CONVERTER (DAC)

AIM:

To construct a 4-bit R-2R ladder type DAC converter. Plot the transfer characteristics. . Calculate resolution of converter from the graph.

APPARATUS:

1. Resistors
2. DMM
3. Fixed power supply(+15V and -15V and 5V)
4. Connecting wires
5. Power supply

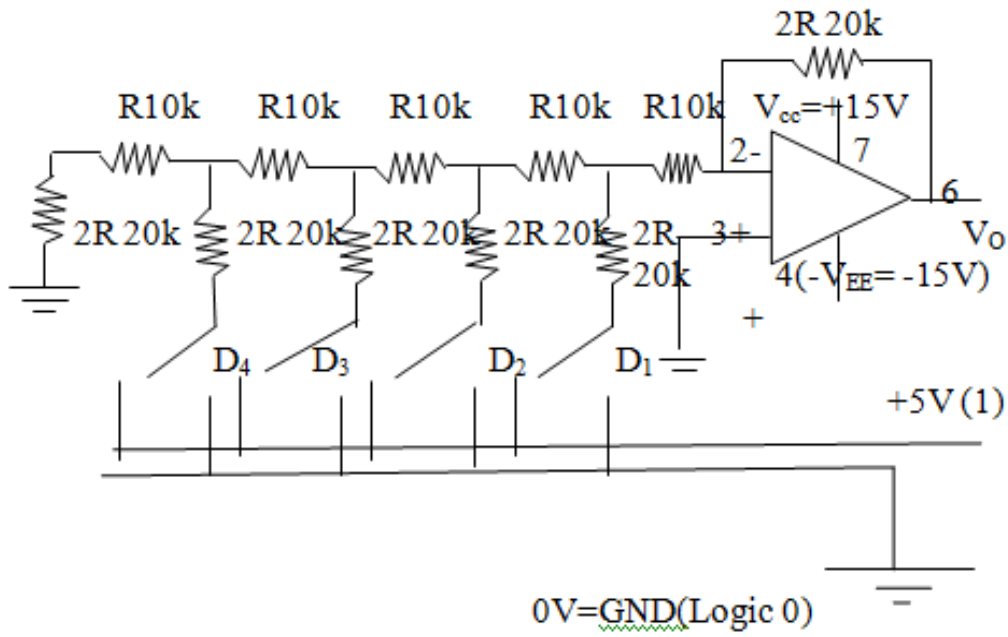
DESIGN:

1. Choose $R=10K\Omega, 2R=20K\Omega$
2. For logic '0' short to ground and logic '1' connected to a +5v

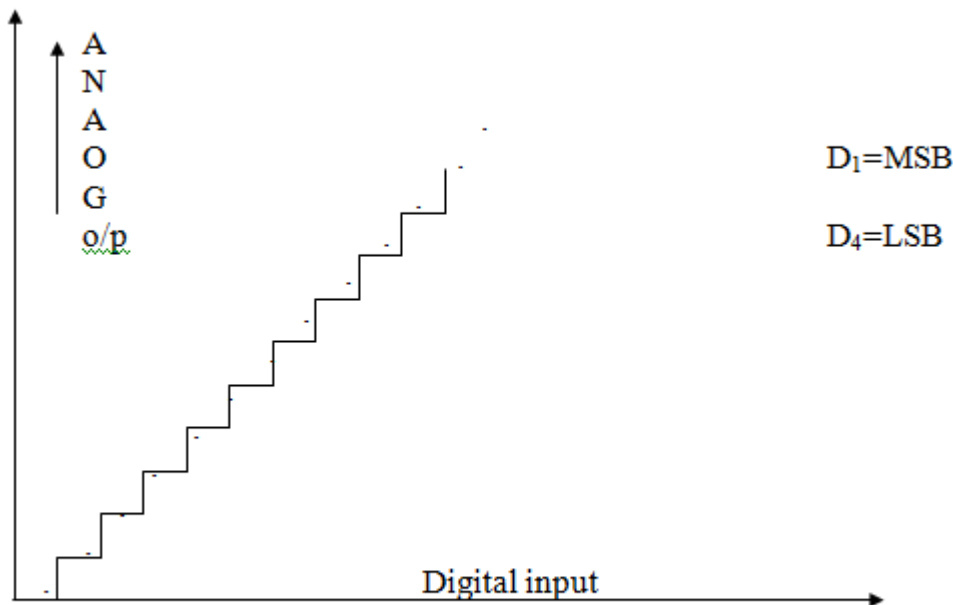
PROCEDURE:

1. Set up the circuit as shown in figure.
2. Measure the output voltage for all binary inputs (0000to 1111) states and plot graph of binary inputs Vs output voltage.
3. Measure the size of each step and hence calculate resolution.
$$\text{Resolution} = [V_{fs} / (2^n - 1)]$$
4. Tabulate all the readings.

Circuit diagram:



Transfer Characteristics:



RESULT:

**3-8 DECODER – IC 74X138
&
8-3 PRIORITY ENCODER – IC 74X148**

EXP NO:

DATE:

3-8 DECODER-IC 74X138 & 8-3 PRIORITY ENCODER-IC 74X148

3-8 DECODER-IC 74X138

AIM:To verify the truth table of 3-8 Decoder.

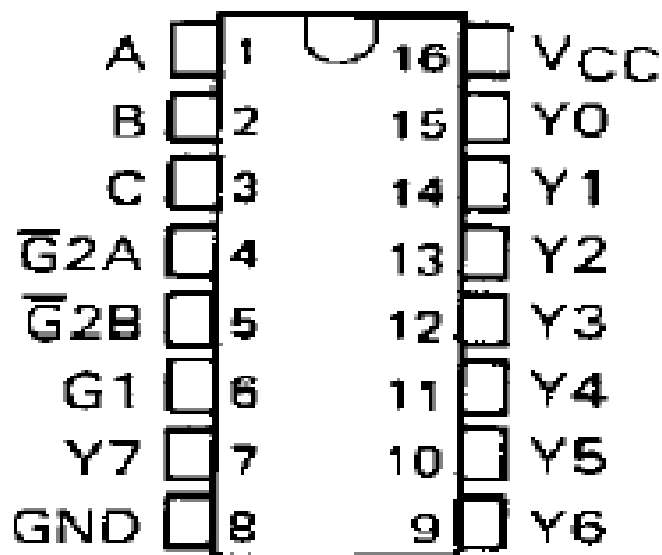
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X138.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

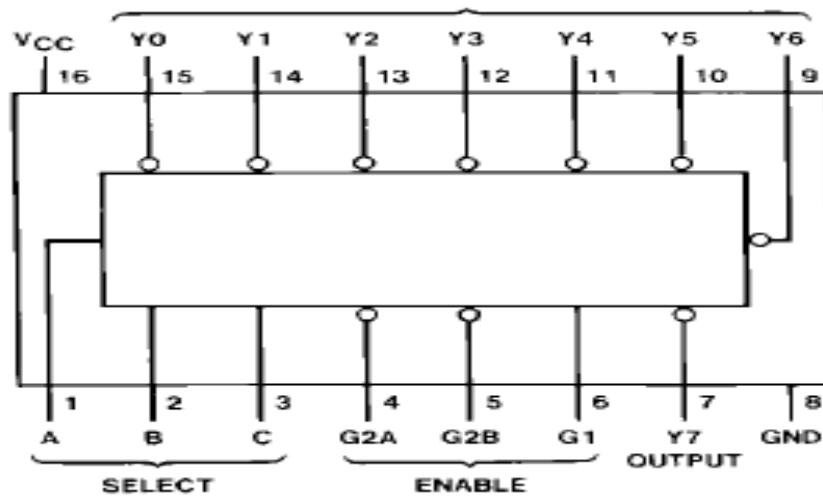
PIN CONFIGURATION:



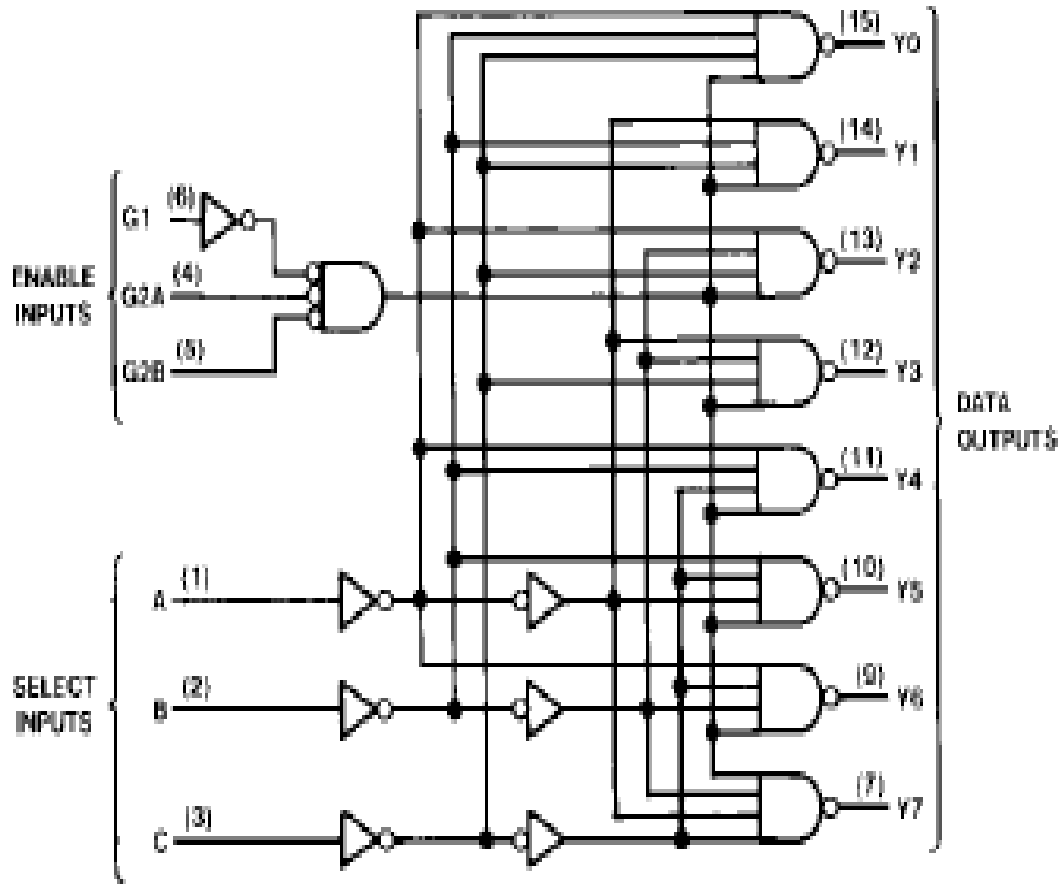
TRUTH TABLE: $G2 = G2A' + G2B'$

| Inputs | | | | | Outputs | | | | | | | |
|--------|-------------|--------|---|---|---------|----|----|----|----|----|----|----|
| Enable | | Select | | | | | | | | | | |
| G1 | G2 (Note 1) | C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| X | H | X | X | X | H | H | H | H | H | H | H | H |
| L | X | X | X | X | H | H | H | H | H | H | H | H |
| H | L | L | L | L | L | H | H | H | H | H | H | H |
| H | L | L | L | H | H | L | H | H | H | H | H | H |
| H | L | L | H | L | H | H | L | H | H | H | H | H |
| H | L | L | H | H | H | H | H | L | H | H | H | H |
| H | L | H | L | L | H | H | H | H | L | H | H | H |
| H | L | H | L | H | H | H | H | H | H | L | H | H |
| H | L | H | H | L | H | H | H | H | H | H | L | H |
| H | L | H | H | H | H | H | H | H | H | H | H | L |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

8-3PRIORITY ENCODER – IC 74X148

AIM:To verify the truth table of 8-3 Priority Encoder.

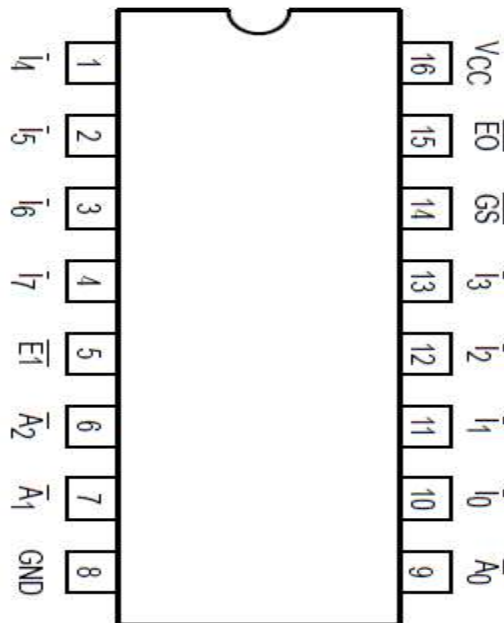
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X148.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

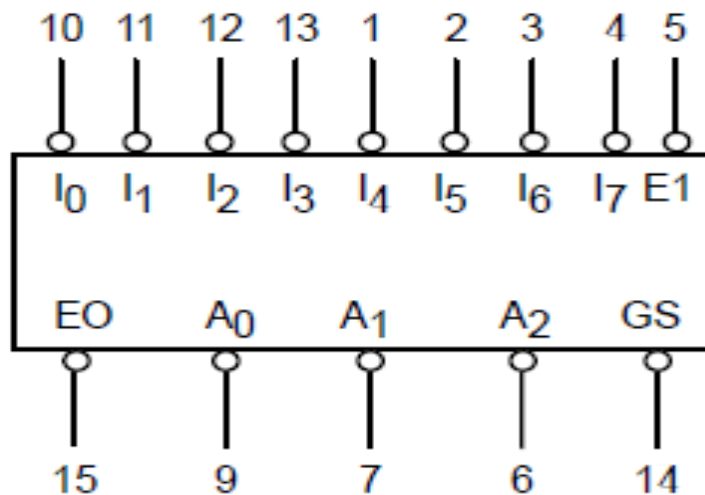
PIN CONFIGURATION:



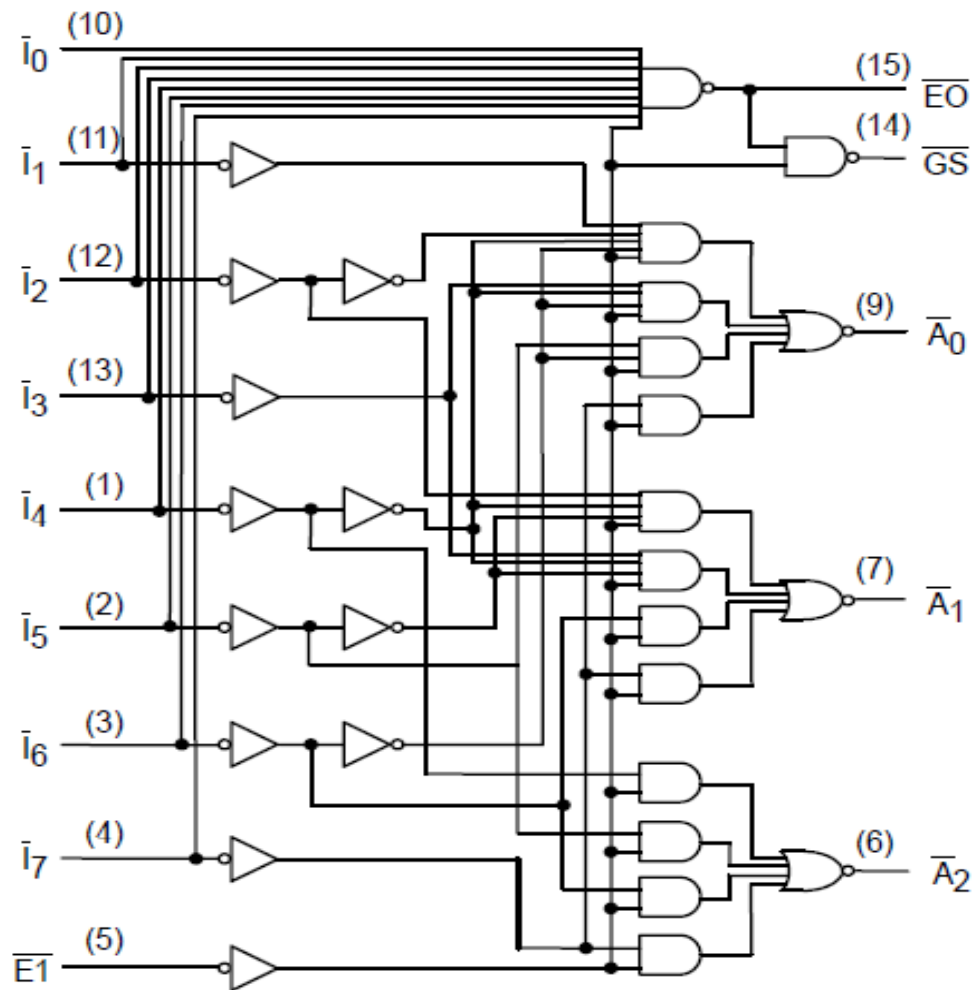
TRUTH TABLE:

| Inputs | | | | | | | | | Outputs | | | | |
|------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------------|-------------|-------------|-------------|------------|
| $\bar{E}1$ | \bar{l}_0 | \bar{l}_1 | \bar{l}_2 | \bar{l}_3 | \bar{l}_4 | \bar{l}_5 | \bar{l}_6 | \bar{l}_7 | \overline{GS} | \bar{A}_0 | \bar{A}_1 | \bar{A}_2 | $\bar{E}O$ |
| H | X | X | X | X | X | X | X | X | H | H | H | H | H |
| L | H | H | H | H | H | H | H | H | H | H | H | H | L |
| L | X | X | X | X | X | X | X | L | L | L | L | L | H |
| L | X | X | X | X | X | X | L | H | L | H | L | L | H |
| L | X | X | X | X | X | L | H | H | L | L | H | L | H |
| L | X | X | X | X | L | H | H | H | L | H | H | L | H |
| L | X | X | X | L | H | H | H | H | L | L | L | H | H |
| L | X | X | L | H | H | H | H | H | L | H | L | H | H |
| L | X | L | H | H | H | H | H | H | L | L | H | H | H |
| L | L | H | H | H | H | H | H | H | L | H | H | H | H |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for 3-8 decoder and 8-3 priority encoder by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

PROCEDURE:

1. Switch on the system and open the Modelsim Window.
2. Open the new editor window through File → New → Source → VHDL.
3. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
4. Compile the program. If there is any error found, rectify the error in a program and again Compile until getting the error free code.
5. Simulate the program. Add the waveform with input and output signals in the program.
6. Apply different values to the input signals and observe the output signals for corresponding inputs.

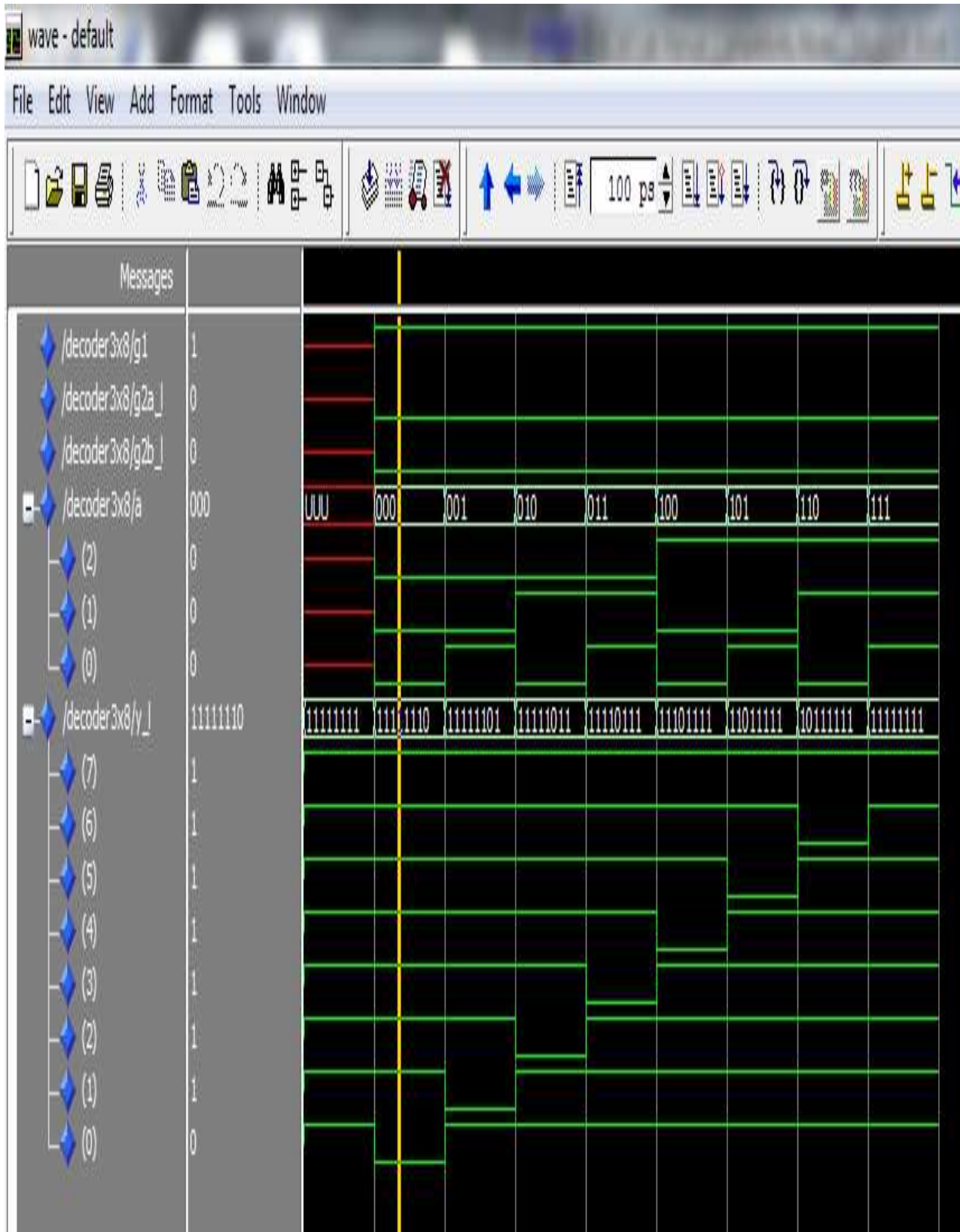
VHDL PROGRAM FOR 3-8DECODER:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DEC3TO8 IS
    PORT (G1, G2A_L, G2B_L: IN STD_LOGIC;
          A: IN STD_LOGIC_VECTOR (2 DOWNTO 0);
          Y_L: OUT STD_LOGIC_VECTOR (0 TO 7));
END DEC3TO8;

ARCHITECTURE DATAFLOW OF DEC3TO8 IS
    SIGNAL Y_L_I:STD_LOGIC_VECTOR (0 TO 7);
    BEGIN
        WITH A SELECT
            Y_L_I<="11111110" WHEN "000" ,
                "11111101" WHEN "001",
                "11111011" WHEN "010",
                "11110111" WHEN "011",
                "11101111" WHEN "100",
                "11011111" WHEN "101",
                "10111111" WHEN "110",
                "01111111" WHEN "111",
                "01111111" WHEN OTHERS;
            Y_L<=Y_L_I WHEN (G1 AND (NOT G2A_L) AND (NOT G2B_L)) ='0'
                ELSE "11111111";
    END DATAFLOW;
```

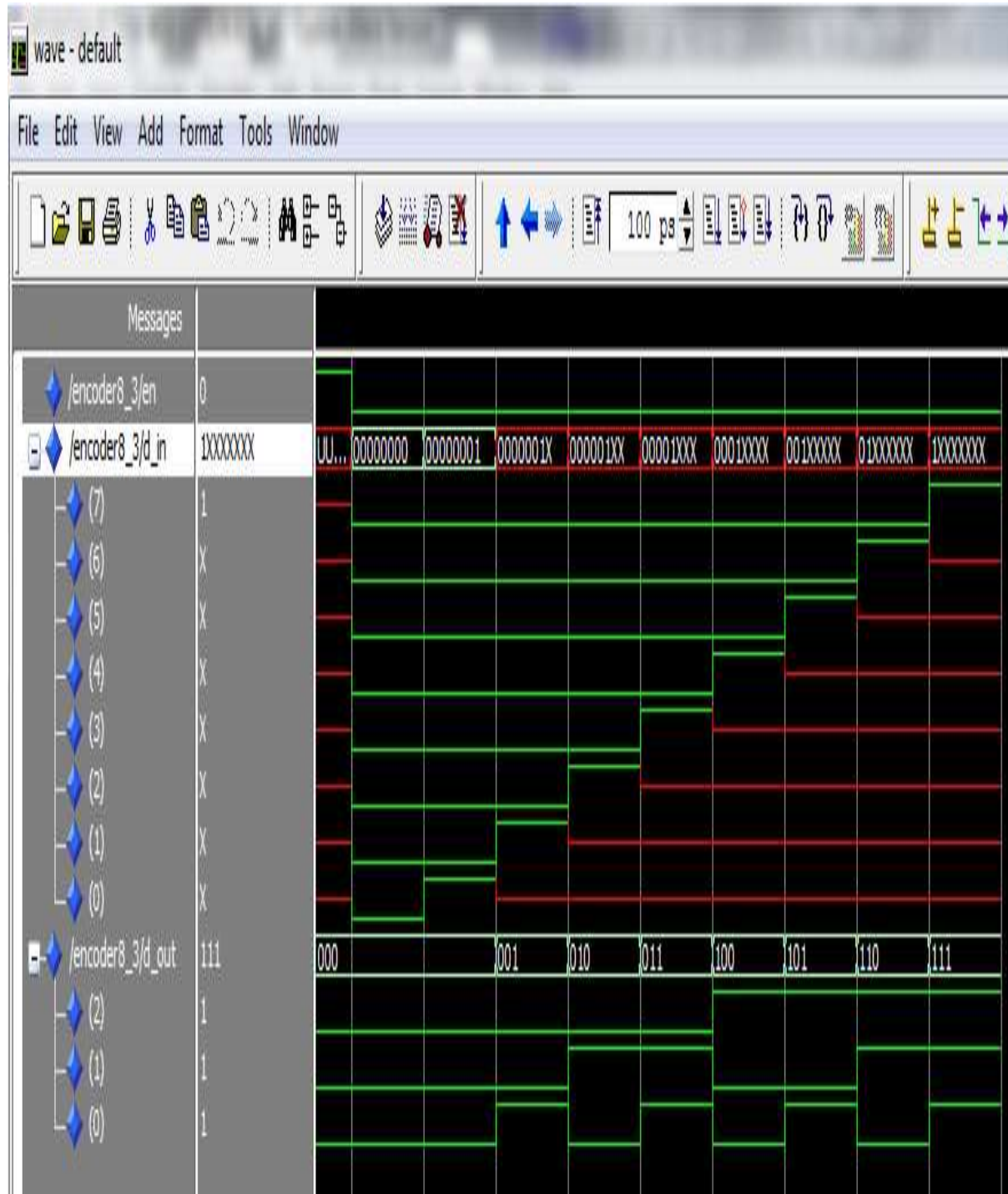

SIMULATION RESULTS:



VHDL PROGRAM FOR 8-3 PRIORITY ENCODER:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ENCODER8_3 IS
    PORT (EN: IN STD_LOGIC;
          D_IN: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
          D_OUT: OUT STD_LOGIC_VECTOR (2 DOWNTO 0) );
END ENCODER8_3;
ARCHITECTURE ENCODER_ARCH OF ENCODER8_3 IS
    BEGIN
        PROCESS (EN, D_IN)
            BEGIN
                IF (EN = '1') THEN
                    D_OUT <= "000";
                ELSE
                    CASE D_IN IS
                        WHEN "00000001" => D_OUT <= "000";
                        WHEN "0000001X" => D_OUT <= "001";
                        WHEN "000001XX" => D_OUT <= "010";
                        WHEN "00001XXX" => D_OUT <= "011";
                        WHEN "0001XXXX" => D_OUT <= "100";
                        WHEN "001XXXXX" => D_OUT <= "101";
                        WHEN "01XXXXXX" => D_OUT <= "110";
                        WHEN "1XXXXXXX" => D_OUT <= "111";
                        WHEN OTHERS => NULL;
                    END CASE;
                END IF;
            END PROCESS;
        END ENCODER_ARCH;
```

SIMULATION RESULTS:



RESULT:

**8X1 MULTIPLEXER - IC 74X151
&
2X4 DEMULTIPLEXER - IC 74X155**

EXP NO:

DATE:

8X1 MULTIPLEXER-IC 74X151&2X4 DEMULTIPLEXER-IC 74X155

8X1 MULTIPLEXER – IC 74X151

AIM: To verify the truth table of 8X1 Multiplexer.

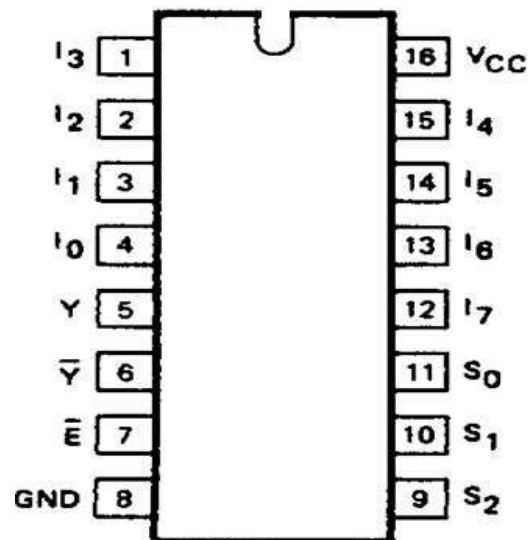
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X151.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

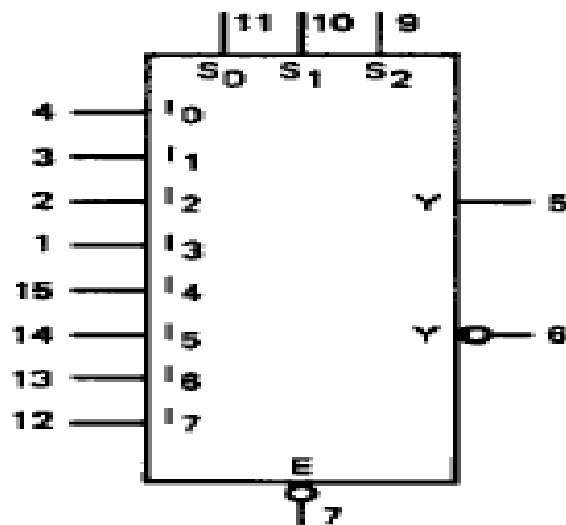
PIN CONFIGURATION:



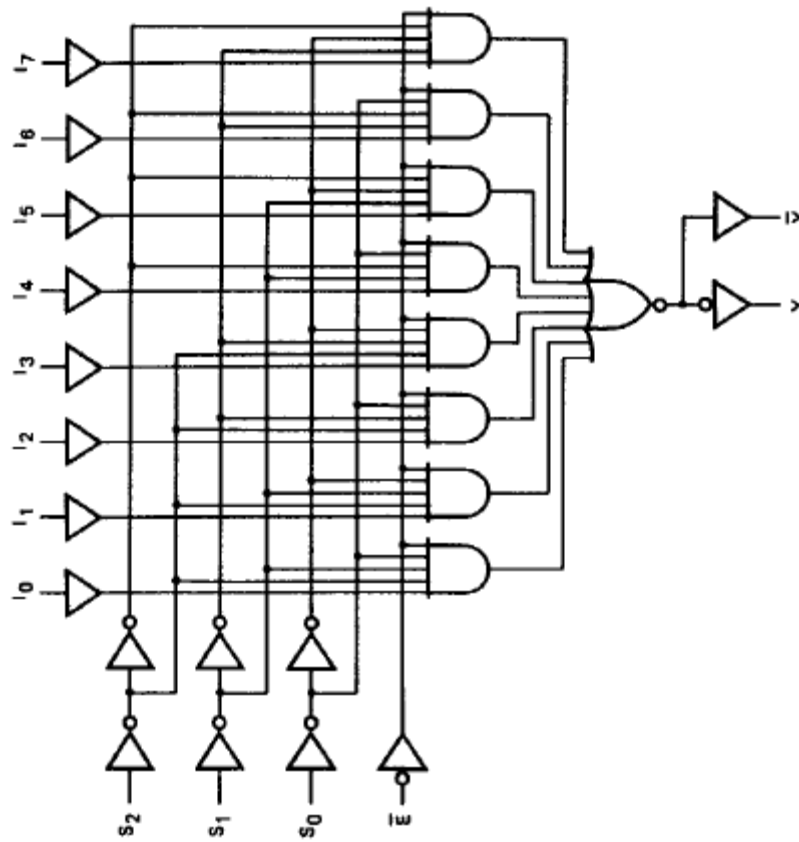
TRUTH TABLE:

| INPUTS | | | | | | | | | | | | OUTPUTS | |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----------|---|
| \bar{E} | S_2 | S_1 | S_0 | I_0 | I_1 | I_2 | I_3 | I_4 | I_5 | I_6 | I_7 | \bar{Y} | Y |
| H | X | X | X | X | X | X | X | X | X | X | X | H | L |
| L | L | L | L | L | X | X | X | X | X | X | X | H | L |
| L | L | L | L | H | X | X | X | X | X | X | X | L | H |
| L | L | L | H | X | L | X | X | X | X | X | X | H | L |
| L | L | L | H | X | H | X | X | X | X | X | X | L | H |
| L | L | H | L | X | X | L | X | X | X | X | X | H | L |
| L | L | H | L | X | X | H | X | X | X | X | X | L | H |
| L | L | H | H | X | X | X | L | X | X | X | X | H | L |
| L | L | H | H | X | X | X | H | X | X | X | X | L | H |
| L | H | L | L | X | X | X | X | L | X | X | X | H | L |
| L | H | L | L | X | X | X | X | H | X | X | X | L | H |
| L | H | L | H | X | X | X | X | X | L | X | X | H | L |
| L | H | L | H | X | X | X | X | X | H | X | X | L | H |
| L | H | H | L | X | X | X | X | X | X | L | X | H | L |
| L | H | H | L | X | X | X | X | X | X | H | X | L | H |
| L | H | H | H | X | X | X | X | X | X | X | L | H | L |
| L | H | H | H | X | X | X | X | X | X | X | H | L | H |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

2X4 DEMULTIPLEXER – IC 74X155

AIM:To verify the truth table of 1X4 Demultiplexer.

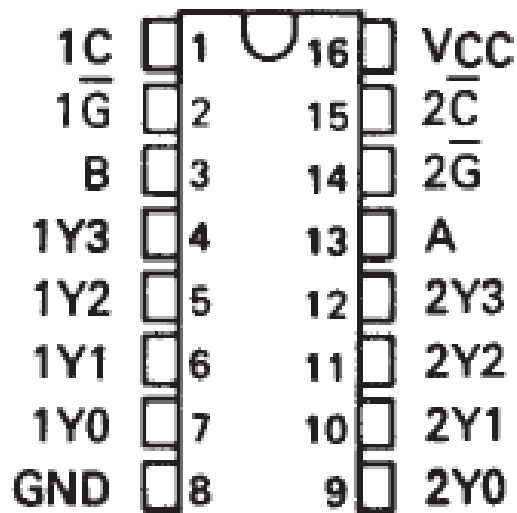
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X155.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

PIN CONFIGURATION:

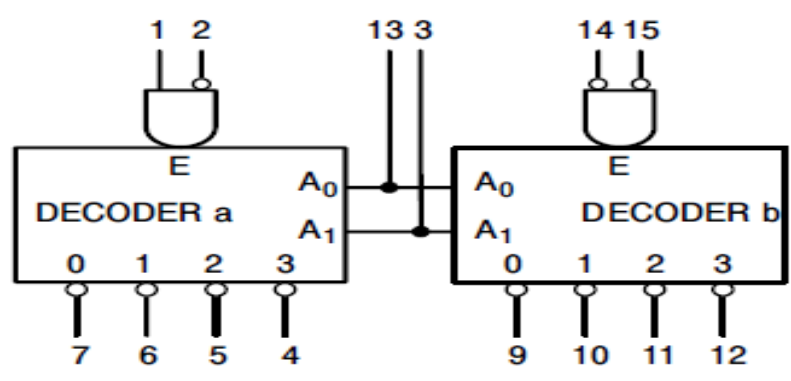


TRUTH TABLE:

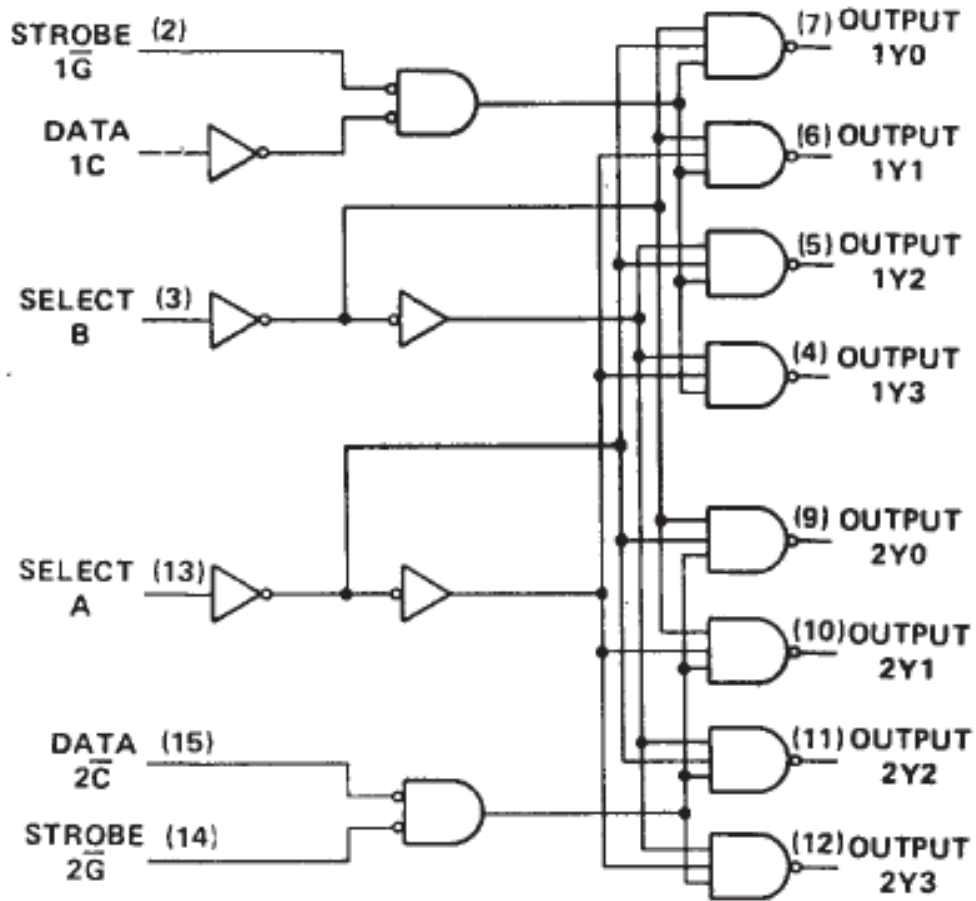
| INPUTS | | | | OUTPUTS | | | |
|--------|---|------------|------|---------|-----|-----|-----|
| SELECT | | STROBE | DATA | 1Y0 | 1Y1 | 1Y2 | 1Y3 |
| B | A | $1\bar{G}$ | 1C | | | | |
| X | X | H | X | H | H | H | H |
| L | L | L | H | L | H | H | H |
| L | H | L | H | H | L | H | H |
| H | L | L | H | H | H | L | H |
| H | H | L | H | H | H | H | L |
| X | X | X | L | H | H | H | H |

| INPUTS | | | | OUTPUTS | | | |
|--------|---|------------|------------|---------|-----|-----|-----|
| SELECT | | STROBE | DATA | 2Y0 | 2Y1 | 2Y2 | 2Y3 |
| B | A | $2\bar{G}$ | $2\bar{C}$ | | | | |
| X | X | H | X | H | H | H | H |
| L | L | L | L | L | H | H | H |
| L | H | L | L | H | L | H | H |
| H | L | L | L | H | H | L | H |
| H | H | L | L | H | H | H | L |
| X | X | X | H | H | H | H | H |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for 8X1 Multiplexer and 1X4 Demultiplexer by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

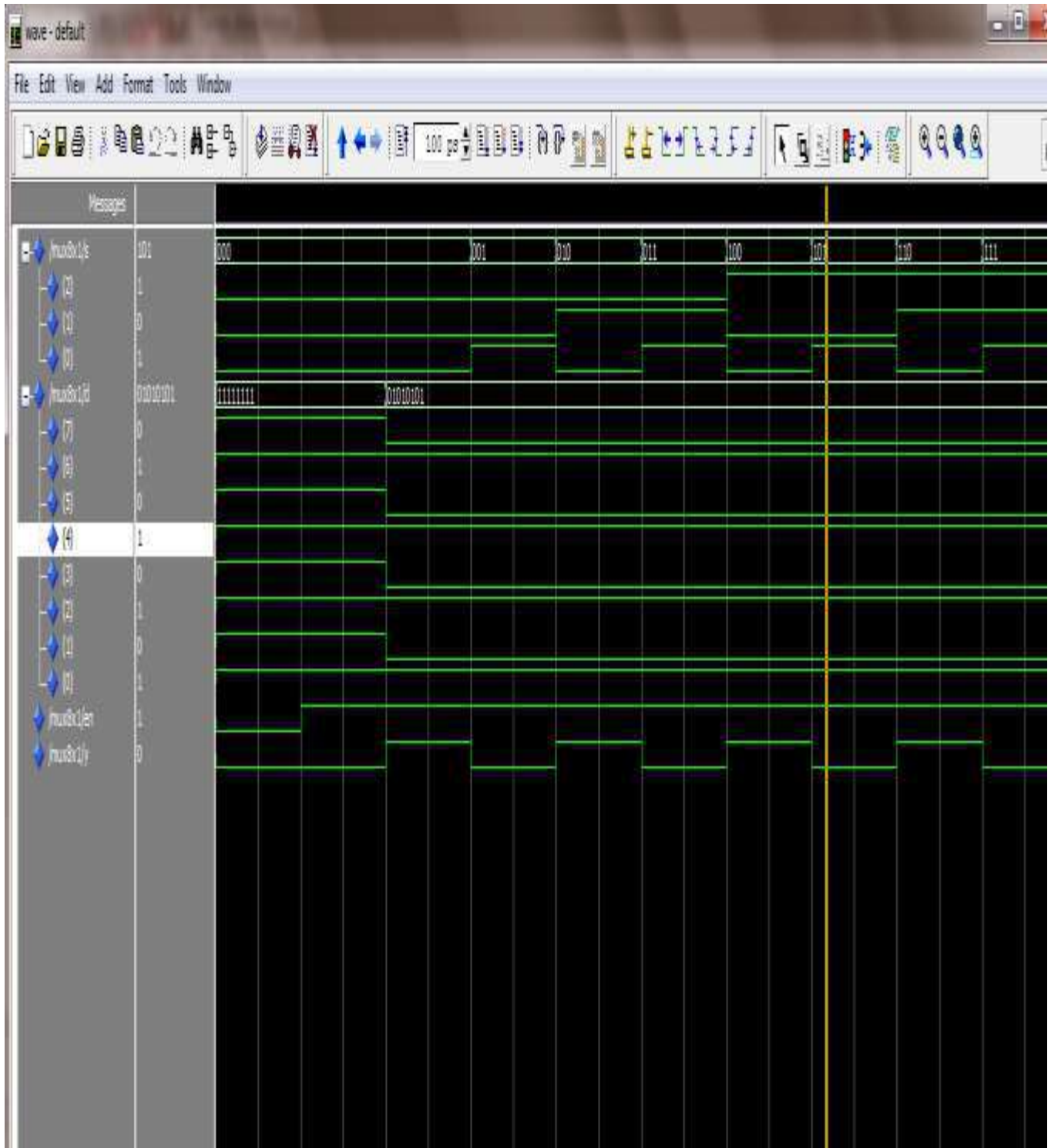
PROCEDURE:

1. Switch on the system and open the Modelsim Window.
2. Open the new editor window through File→New→Source→VHDL.
3. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
4. Compile the program. If there is any error found, rectify the error in a program and again Compile until getting the error free code.
5. Simulate the program. Add the waveform with input and output signals in the program.
6. Apply different values to the input signals and observe the output signals for corresponding inputs.

VHDL PROGRAM FOR 8X1 MULTIPLEXER:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX8X1 IS
    PORT (S: IN STD_LOGIC_VECTOR (2 DOWNTO 0);
D: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
EN: IN STD_LOGIC;
Y: OUT STD_LOGIC);
END MUX8X1;
ARCHITECTURE MUX_8X1 OF MUX8X1 IS
BEGIN
    PROCESS (S, D, EN)
    BEGIN
        IF (EN='0')THEN Y<='0';
        ELSE
            CASE S IS
                WHEN "000"=> Y<=D (0);
                WHEN "001"=> Y<=D (1);
                WHEN "010"=> Y<=D (2);
                WHEN "011"=> Y<=D (3);
                WHEN "100"=> Y<=D (4);
                WHEN "101"=> Y<=D (5);
                WHEN "110"=> Y<=D (6);
                WHEN "111"=> Y<=D (7);
                WHEN OTHERS=> Y<='0';
            END CASE;
        END IF;
    END PROCESS;
END MUX_8X1;
```

SIMULATION RESULTS:



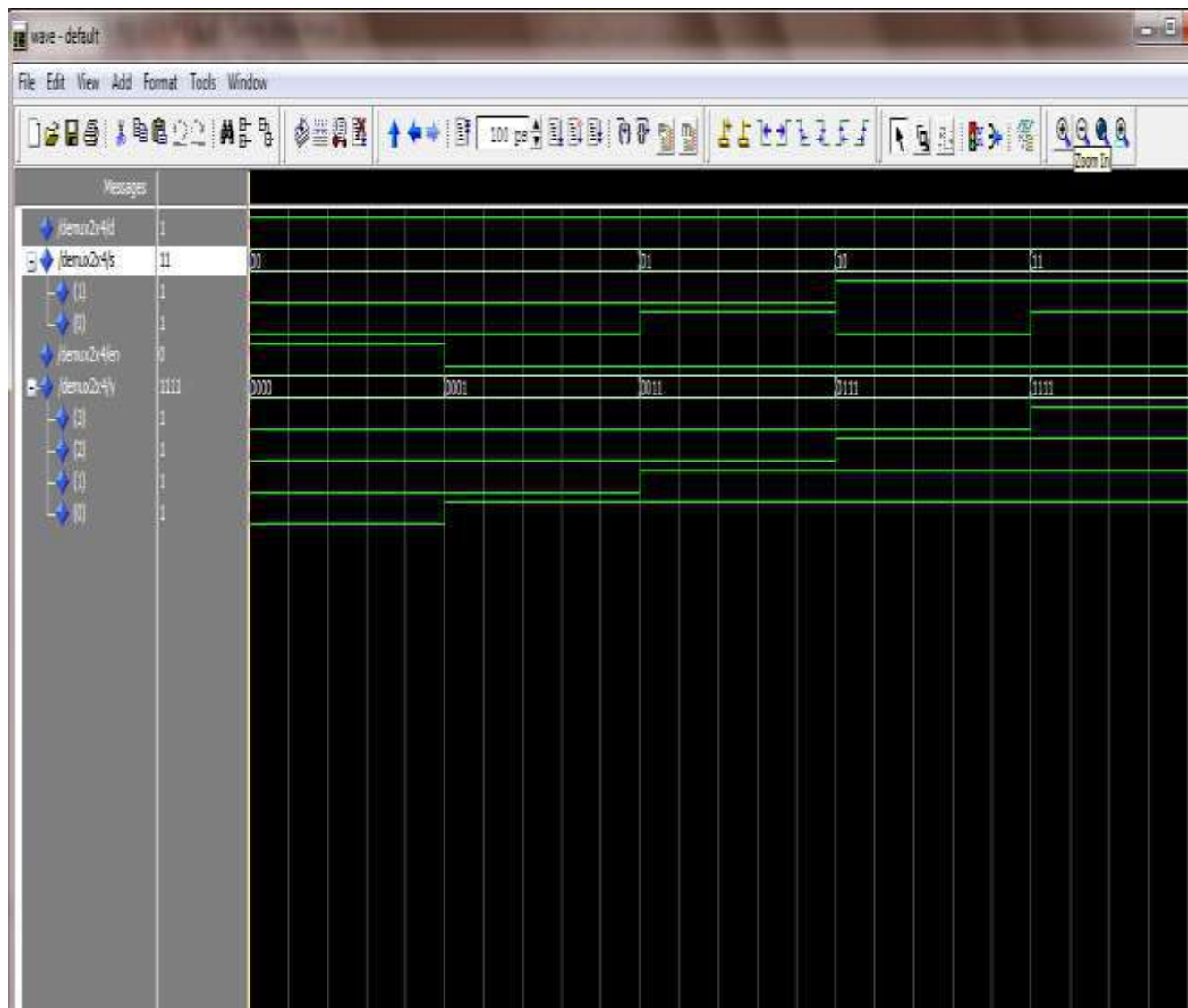
VHDL PROGRAM FOR 1X4 MULTIPLEXER

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DEMUX2X4 IS
    PORT (D, EN: IN STD_LOGIC;
S: IN STD_LOGIC_VECTOR (1 DOWNT0 0);
Y: OUT STD_LOGIC_VECTOR (3 DOWNT0 0));
END DEMUX2X4;

ARCHITECTURE DEMUX_2X4 OF DEMUX2X4 IS
BEGIN
    PROCESS (S, EN, D)
        BEGIN
            IF (EN='0') THEN
                Y<="0000";
            ELSE
                CASE S IS
                    WHEN "00"=> Y (0) <=D;
                    WHEN "01"=> Y (1) <=D;
                    WHEN "10"=> Y (2) <=D;
                    WHEN "11"=> Y (3) <=D;
                    WHEN OTHERS=> Y<="0000";
                END CASE;
            END IF;
        END PROCESS;
END DEMUX_2X4;
```

SIMULATION RESULTS:



RESULT:

4-BIT COMPARATOR – IC 74X85

EXP NO:

DATE:

4-BIT COMPARATOR – IC 74X85

AIM: To verify the truth table of 4-bit comparator.

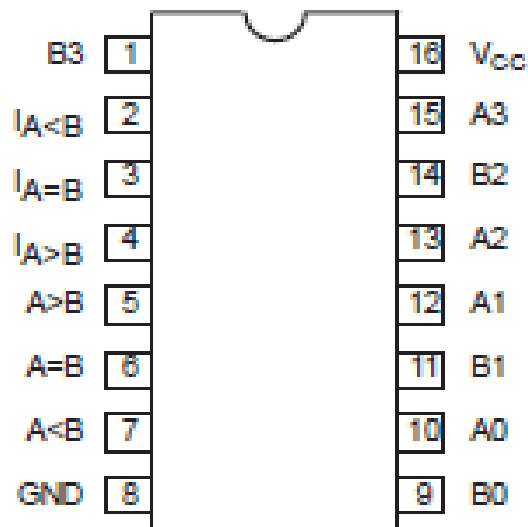
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X85.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

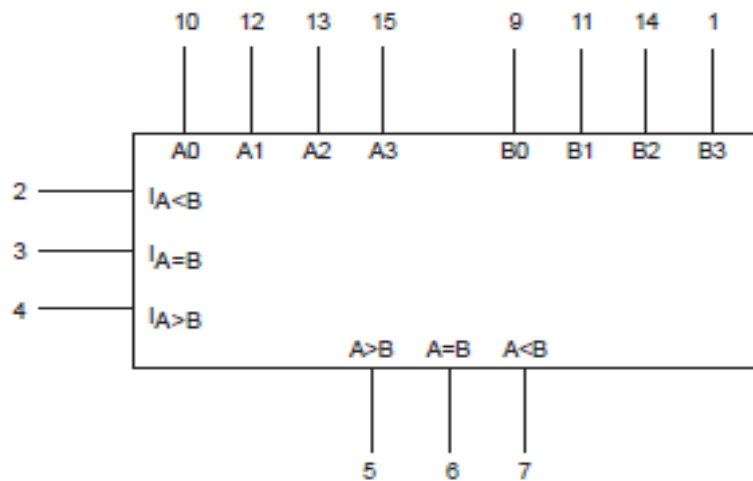
PIN CONFIGURATION:



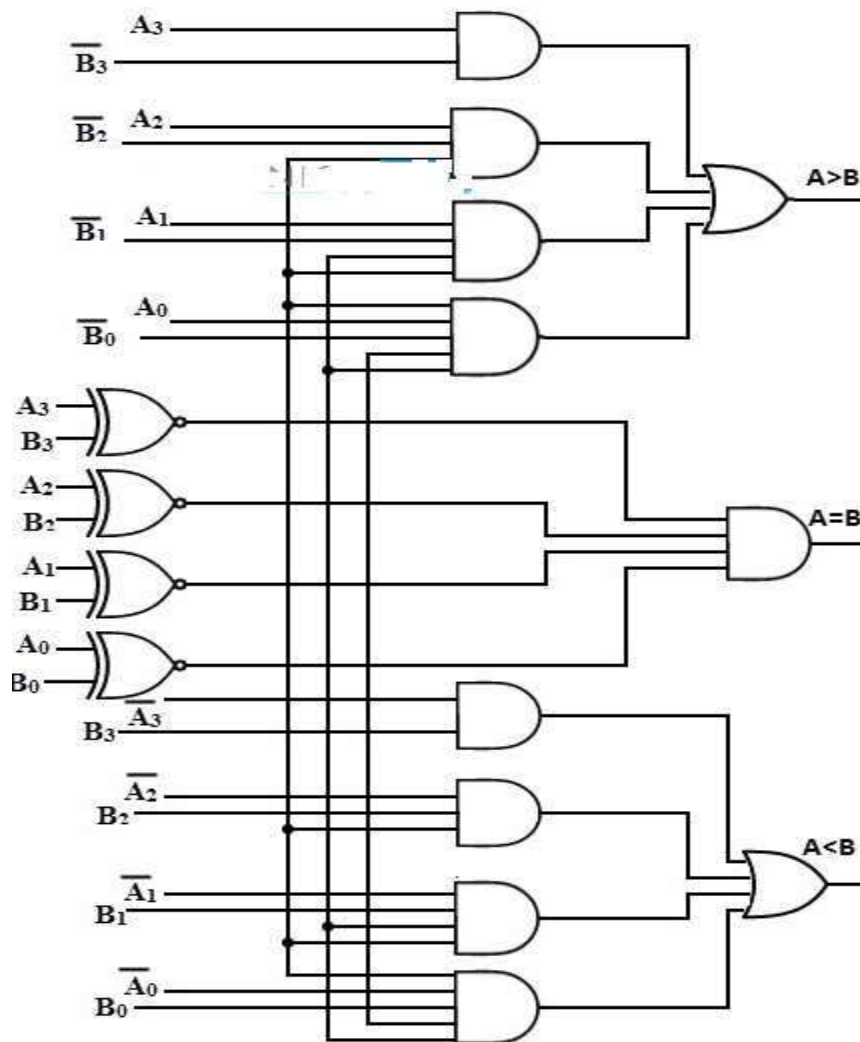
TRUTH TABLE:

| COMPARING INPUTS | | | | EXPANSION INPUTS | | | OUTPUTS | | |
|------------------|-------|-------|-------|---------------------|---------------------|------------------|---------|-----|-----|
| A3,B3 | A2,B2 | A1,B1 | A0,B0 | I _{A>B} | I _{A<B} | I _{A=B} | A>B | A<B | A=B |
| A3>B3 | X | X | X | X | X | X | H | L | L |
| A3<B3 | X | X | X | X | X | X | L | H | L |
| A3=B3 | A2>B2 | X | X | X | X | X | H | L | L |
| A3=B3 | A2<B2 | X | X | X | X | X | L | H | L |
| A3=B3 | A2=B2 | A1>B1 | X | X | X | X | H | L | L |
| A3=B3 | A2=B2 | A1<B1 | X | X | X | X | L | H | L |
| A3=B3 | A2=B2 | A1=B1 | A0>B0 | X | X | X | H | L | L |
| A3=B3 | A2=B2 | A1=B1 | A0<B0 | X | X | X | L | H | L |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | H | L | L | H | L | L |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | L | H | L | L | H | L |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | L | L | H | L | L | H |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | X | X | H | L | L | H |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | H | H | L | L | L | L |
| A3=B3 | A2=B2 | A1=B1 | A0=B0 | L | L | L | H | H | L |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for 4-bit Comparator by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

PROCEDURE:

1. Switch on the system and open the Modelsim Window.
2. Open the new editor window through File → New → Source → VHDL.
3. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
4. Compile the program. If there is any error found, rectify the error in a program and again compile until getting the error free code.
5. Simulate the program. Add the waveform with input and output signals in the program.
6. Apply different values to the input signals and observe the output signals for corresponding inputs.

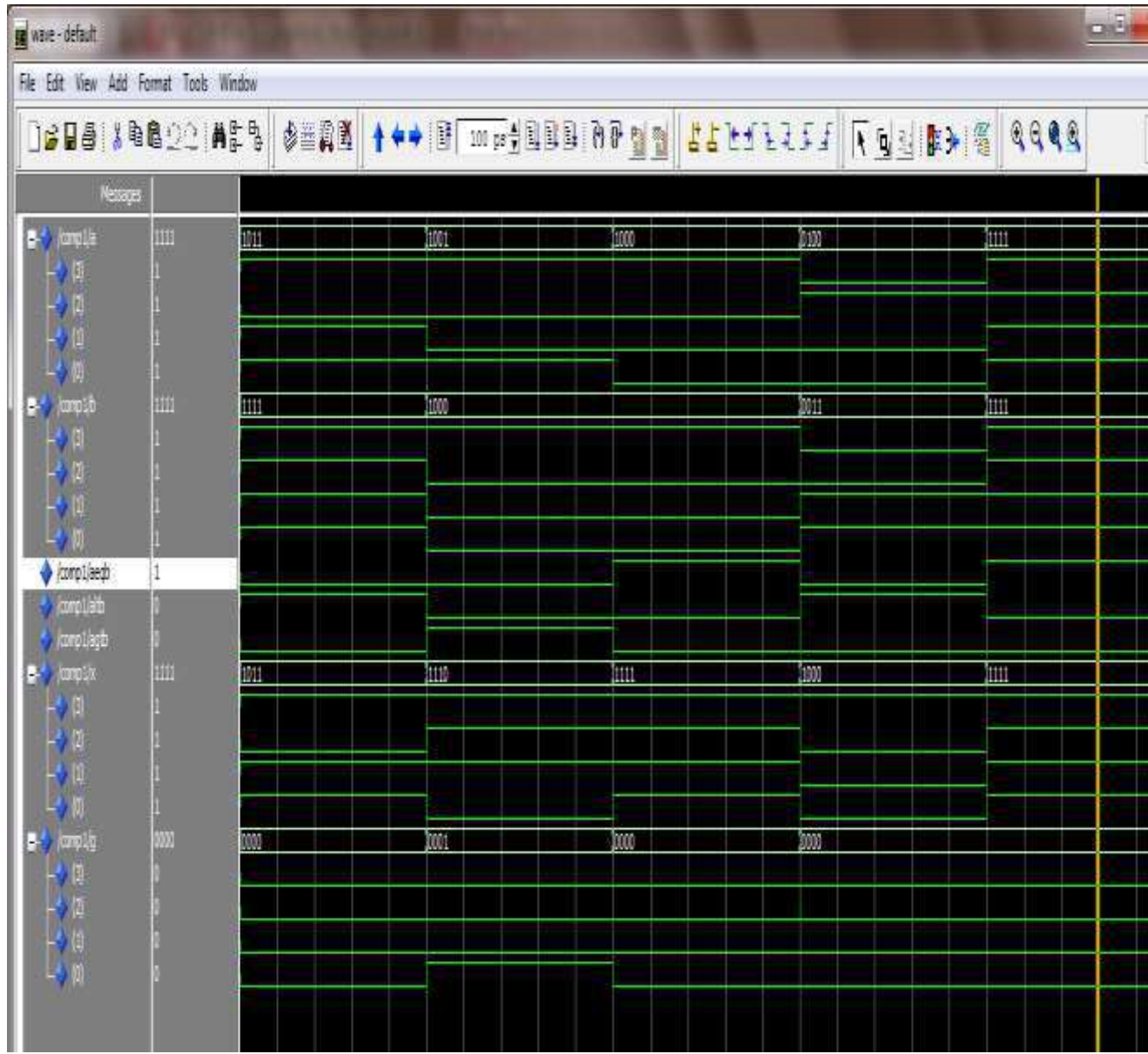
VHDL PROGRAM FOR 4-BIT COMPARATOR:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY COMP IS
    PORT (A, B: IN STD_LOGIC_VECTOR (3 DOWNT0) ;
          AGTB, ALTB, AEQB: OUT STD_LOGIC);
END COMP;

ARCHITECTURE COMP_BEH OF COMP IS
BEGIN
    PROCESS (A, B)
    BEGIN
        IF (A>B) THEN
            AGTB<= '1';
            AEQB<='0';
            ALTB<= '0';
        ELSIF (A<B) THEN
            AGTB<= '0';
            AEQB<='0';
            ALTB<= '1';
        ELSIF (A=B) THEN
            AGTB<= '0';
            AEQB<='1';
            ALTB<= '0';
        END IF;
    END PROCESS;
END COMP_BEH;
```

SIMULATION RESULTS:



RESULT:

D - FLIP FLOP – IC 74X74

EXP NO:

DATE:

D-FLIP FLOP – IC 74X74

AIM: To verify the truth table of D-flip flop.

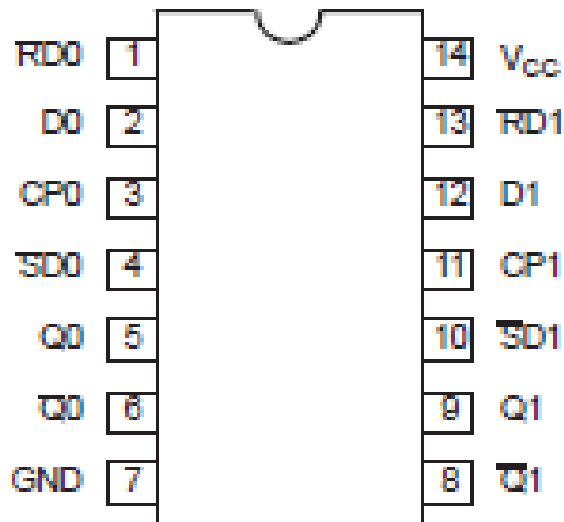
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X74.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

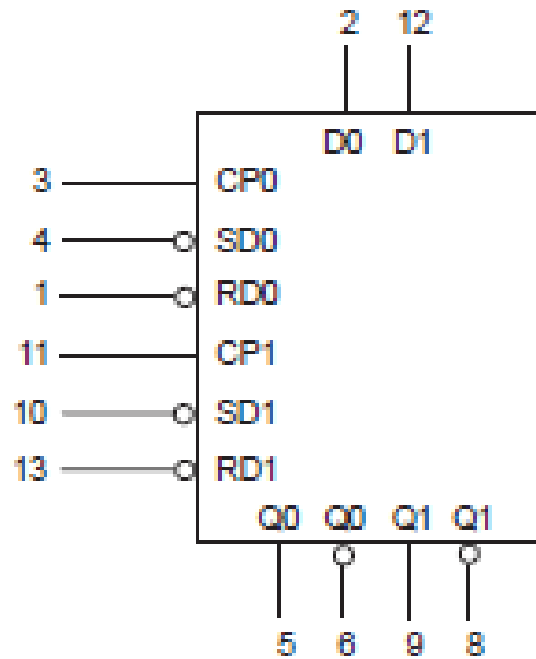
PIN CONFIGURATION:



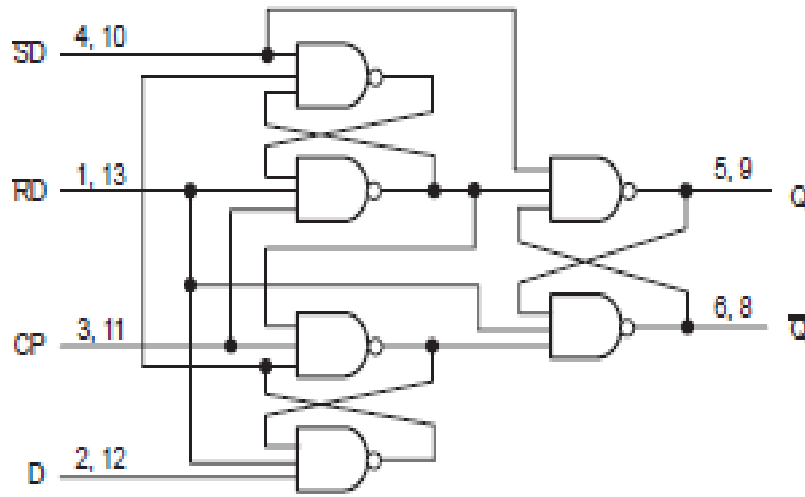
TRUTH TABLE:

| INPUTS | | | | OUTPUTS | |
|-----------------|-----------------|----------------|---|---------|----------------|
| \overline{SD} | \overline{RD} | CP | D | Q | \overline{Q} |
| L | H | X | X | H | L |
| H | L | X | X | L | H |
| L | L | X | X | H | H |
| H | H | \uparrow | h | H | L |
| H | H | \uparrow | l | L | H |
| H | H | \updownarrow | X | NC | NC |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for D-Flip-Flop by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

PROCEDURE:

1. Switch on the system and open the Modelsim Window.
2. Open the new editor window through File→New→Source→VHDL.
3. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
4. Compile the program. If there is any error found, rectify the error in a program and again Compile until getting the error free code.
5. Simulate the program. Add the waveform with input and output signals in the program.
6. Apply different values to the input signals and observe the output signals for corresponding inputs.

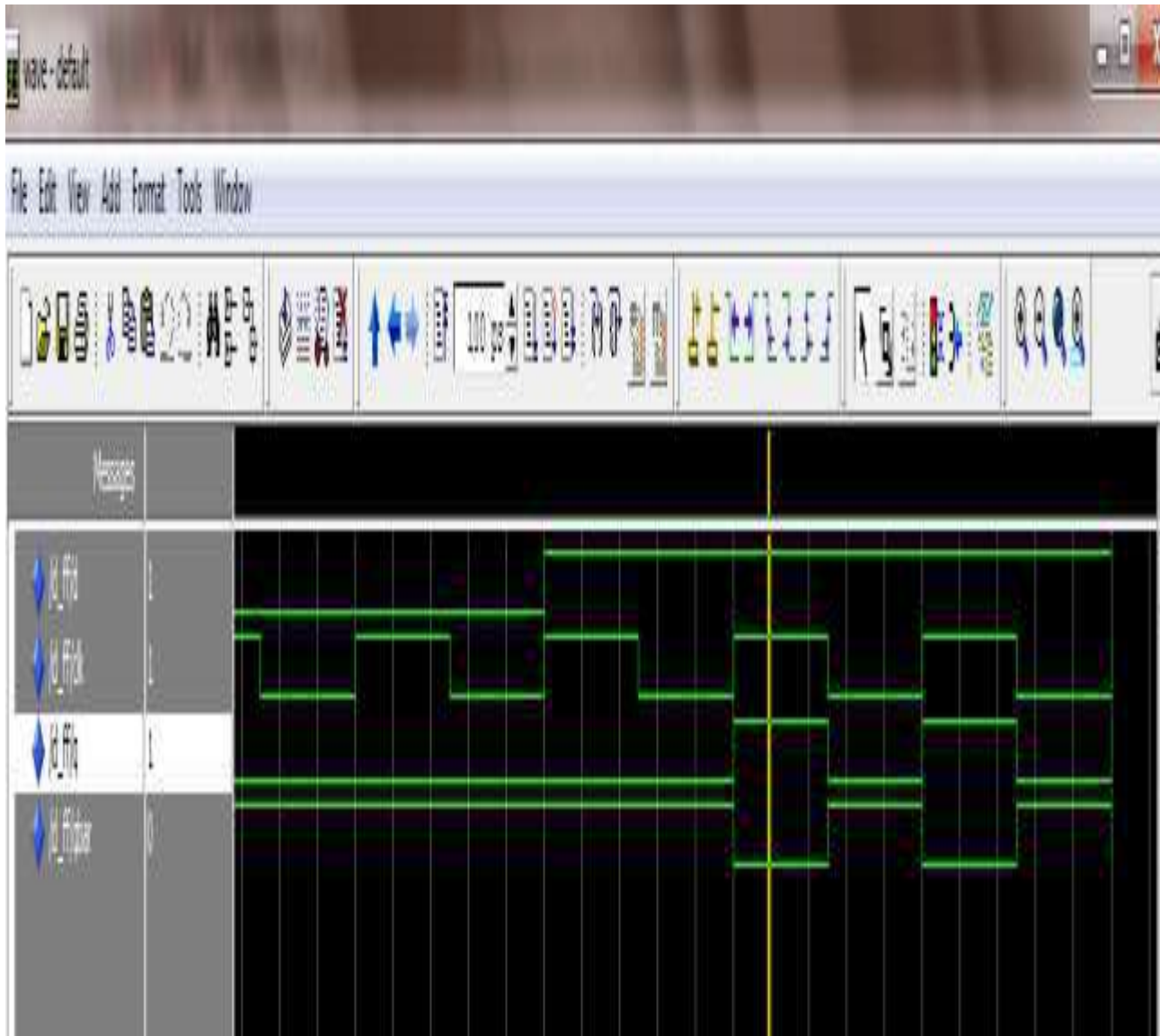
VHDL PROGRAM FOR D-FLIP FLOP:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY D_FF IS
    PORT (D, CLK: IN STD_LOGIC;
          Q, QBAR: INOUT STD_LOGIC);
END D_FF;

ARCHITECTURE D_FF1 OF D_FF IS
    BEGIN
        QBAR<=NOT Q;
        PROCESS (D, CLK)
            BEGIN
                IF (CLK='1' AND CLK'EVENT) THEN
                    Q<=D;
                ELSE
                    Q<='0';
                END IF;
            END PROCESS;
        END D_FF1;
```

SIMULATION RESULTS:



RESULT:

JK - FLIP FLOP – IC 74X109

EXP NO:

DATE:

JK-FLIP FLOP – IC 74X109

AIM: To verify the truth table of JK-flip flop.

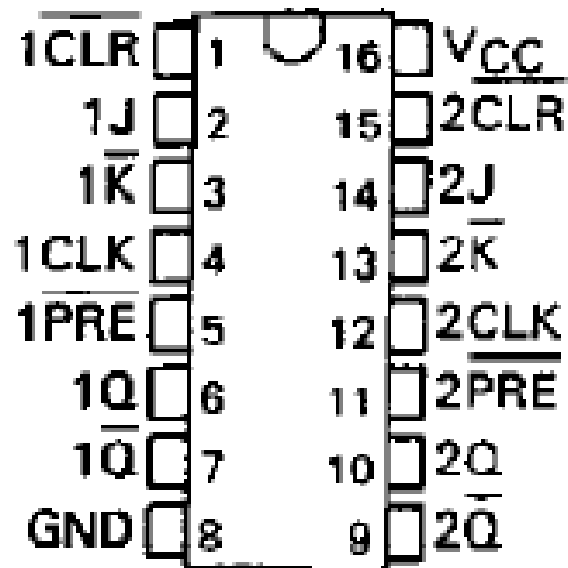
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X109.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

PIN CONFIGURATION:

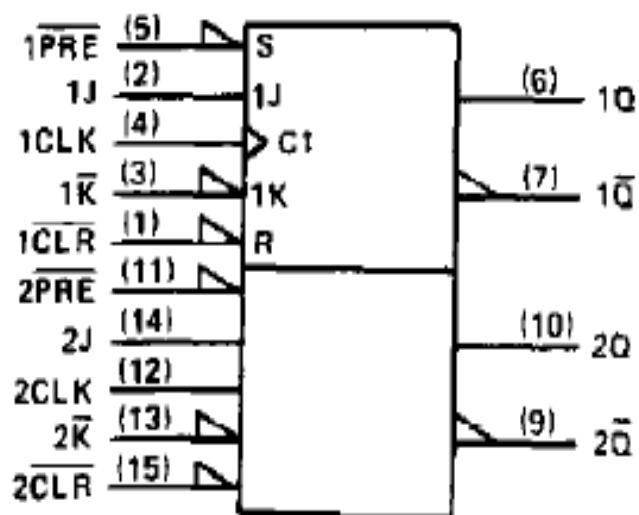


TRUTH TABLE:

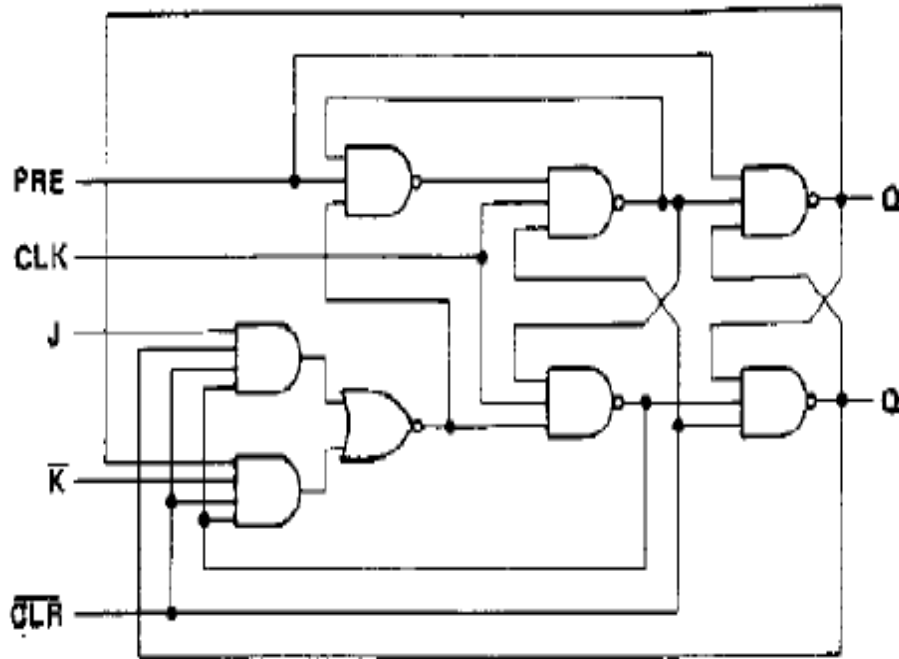
FUNCTION TABLE (each flip-flop)

| INPUTS | | | | | OUTPUTS | |
|--------|-----|-----|---|---|----------------|----------------|
| PRE | CLR | CLK | J | K | Q | \bar{Q} |
| L | H | X | X | X | H | L |
| H | L | X | X | X | L | H |
| L | L | X | X | X | H [†] | H [†] |
| H | H | † | L | L | L | H |
| H | H | † | H | L | TOGGLE | |
| H | H | † | L | H | Q ₀ | \bar{Q}_0 |
| H | H | † | H | H | H | L |
| H | H | L | X | X | Q ₀ | \bar{Q}_0 |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for JK-Flip-Flop by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

PROCEDURE:

7. Switch on the system and open the Modelsim Window.
8. Open the new editor window through File→New→Source→VHDL.
9. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
10. Compile the program. If there is any error found, rectify the error in a program and again Compile until getting the error free code.
11. Simulate the program. Add the waveform with input and output signals in the program.
12. Apply different values to the input signals and observe the output signals for corresponding inputs.

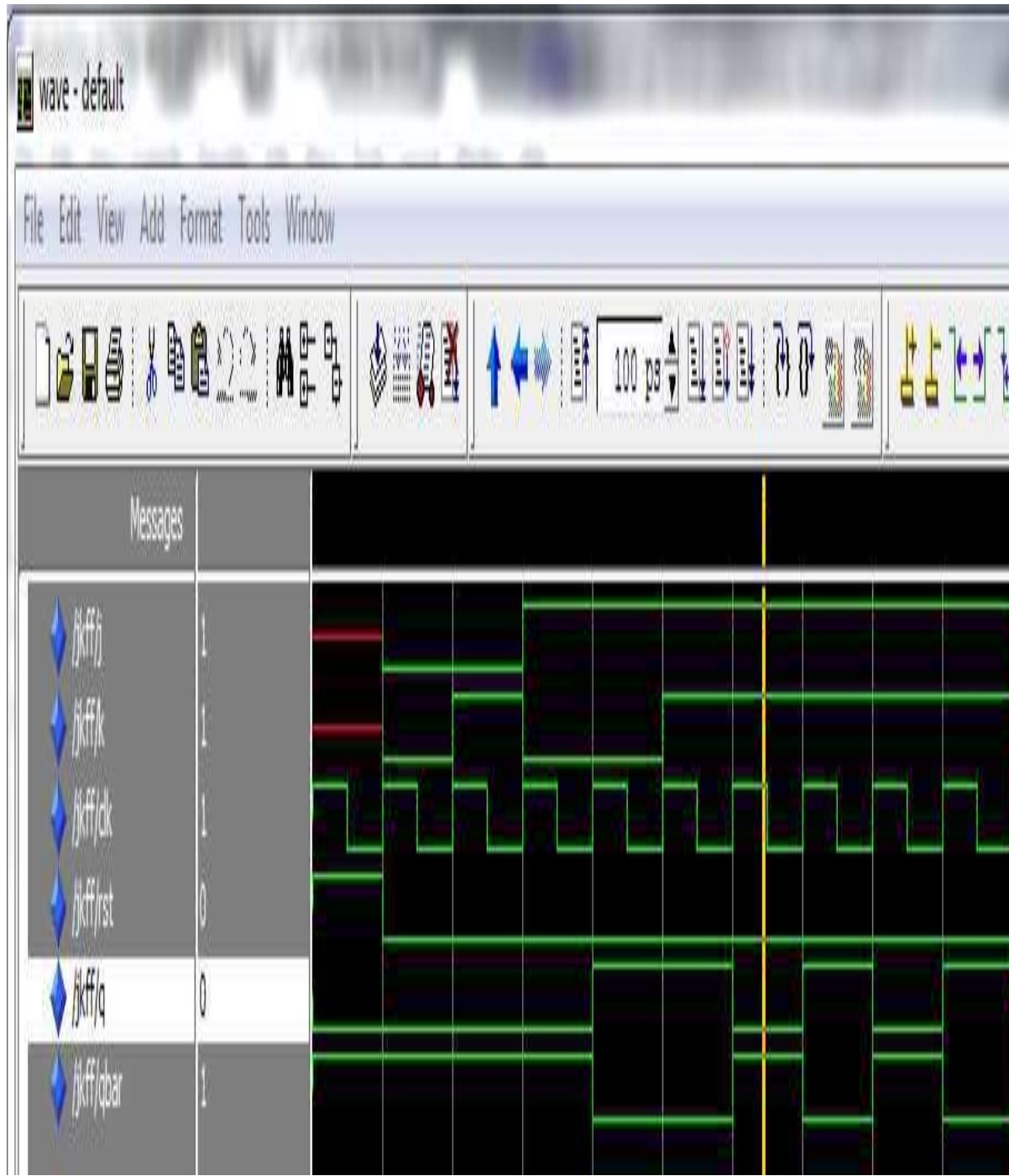
VHDL PROGRAM FOR JK-FLIP FLOP:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY JKFF IS
    PORT (J, K, CLK,RST: IN STD_LOGIC;
Q:INOUT STD_LOGIC);
END JKFF;

ARCHITECTURE JKFF1 OF JKFF IS
    SIGNAL QBAR: STD_LOGIC;
BEGIN
    PROCESS (CLK)
        BEGIN
            IF (CLK='1' AND CLK'EVENT) THEN
                IF (RST='1') THEN
                    Q<='0';
                ELSE
                    Q<= (J AND (NOT Q))OR ((NOT K) AND Q);
                END IF;
            END IF;
        END PROCESS;
        QBAR<= NOT Q;
    END JKFF1;
```

SIMULATION RESULTS:



RESULT:

DECADE COUNTER – IC 74X90

EXP NO:

DATE:

DECADE COUNTER – IC 74X90

AIM: To verify the truth table of Decade counter.

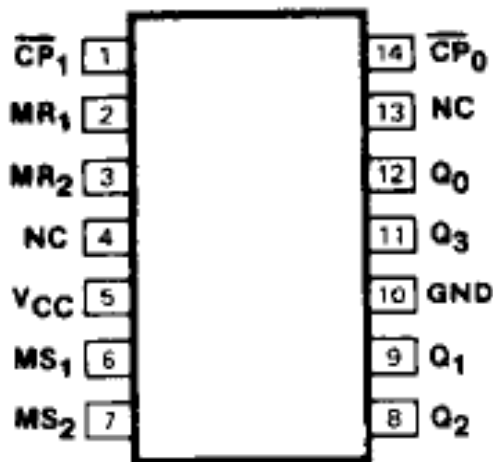
APPARATUS:

1. Digital IC Trainer Kit.
2. Patch Cards.
3. IC 74X90.

PROCEDURE:

1. Connect the parameters as per pin configuration.
2. Switch on the experimental kit.
3. Vary the input according to the truth table.
4. Repeat the same procedure for different values of input.
5. Compare the output values according to the input values.

PIN CONFIGURATION:



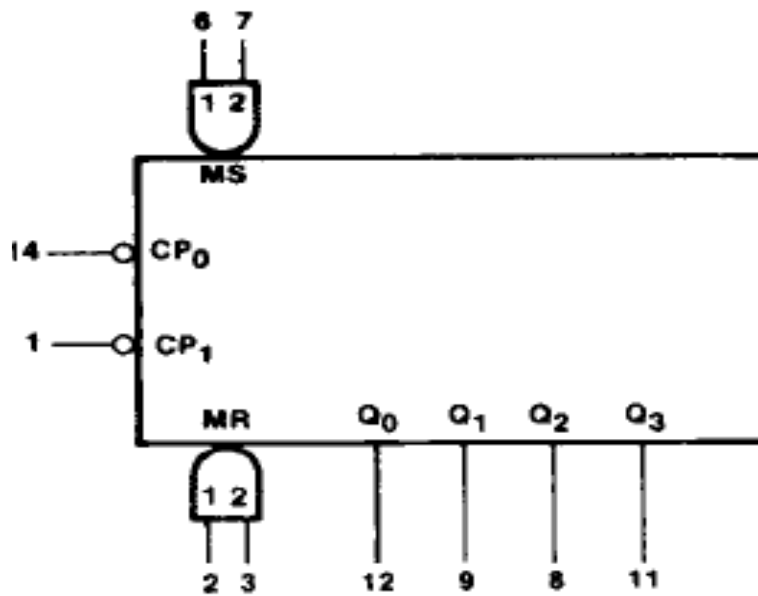
TRUTH TABLE:

| RESET/SET INPUTS | | | | OUTPUTS | | | |
|------------------|-----------------|-----------------|-----------------|----------------|----------------|----------------|----------------|
| MR ₁ | MR ₂ | MS ₁ | MS ₂ | Q ₀ | Q ₁ | Q ₂ | Q ₃ |
| H | H | L | X | L | L | L | L |
| H | H | X | L | L | L | L | L |
| X | X | H | H | H | L | L | H |
| L | X | L | X | | Count | | |
| X | L | X | L | | Count | | |
| L | X | X | L | | Count | | |
| H | L | L | X | | Count | | |

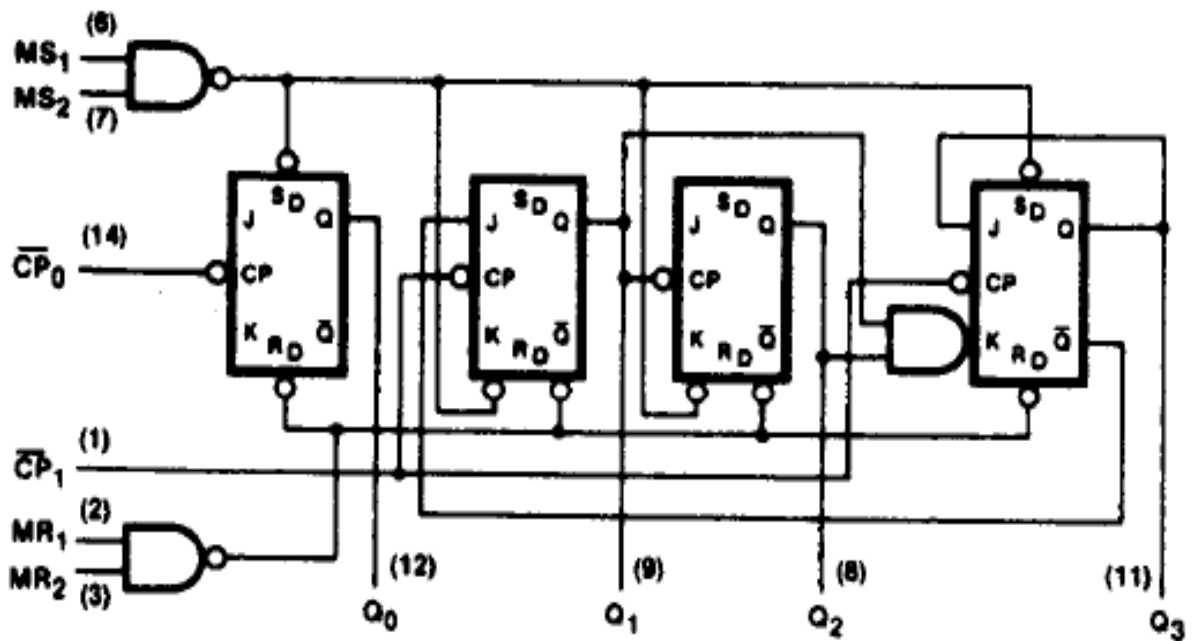
BCD COUNT SEQUENCE:

| COUNT | OUTPUTS | | | |
|-------|----------------|----------------|----------------|----------------|
| | Q ₀ | Q ₁ | Q ₂ | Q ₃ |
| 0 | L | L | L | L |
| 1 | H | L | L | L |
| 2 | L | H | L | L |
| 3 | H | H | L | L |
| 4 | L | L | H | L |
| 5 | H | L | H | L |
| 6 | L | H | H | L |
| 7 | H | H | H | L |
| 8 | L | L | L | H |
| 9 | H | L | L | H |

LOGIC SYMBOL:



LOGIC DIAGRAM:



RESULT:

SOFTWARE

AIM: To write and simulate a VHDL Program for Decade counter by using Modelsim.

SOFTWARES REQUIRED:

- System with Modelsim 6.3 Version.

PROCEDURE:

1. Switch on the system and open the Modelsim Window.
2. Open the new editor window through File → New → Source → VHDL.
3. Type the VHDL Program in editor window and save the program with **file_name.vhd**.
4. Compile the program. If there is any error found, rectify the error in a program and again Compile until getting the error free code.
5. Simulate the program. Add the waveform with input and output signals in the program.
6. Apply different values to the input signals and observe the output signals for corresponding inputs.

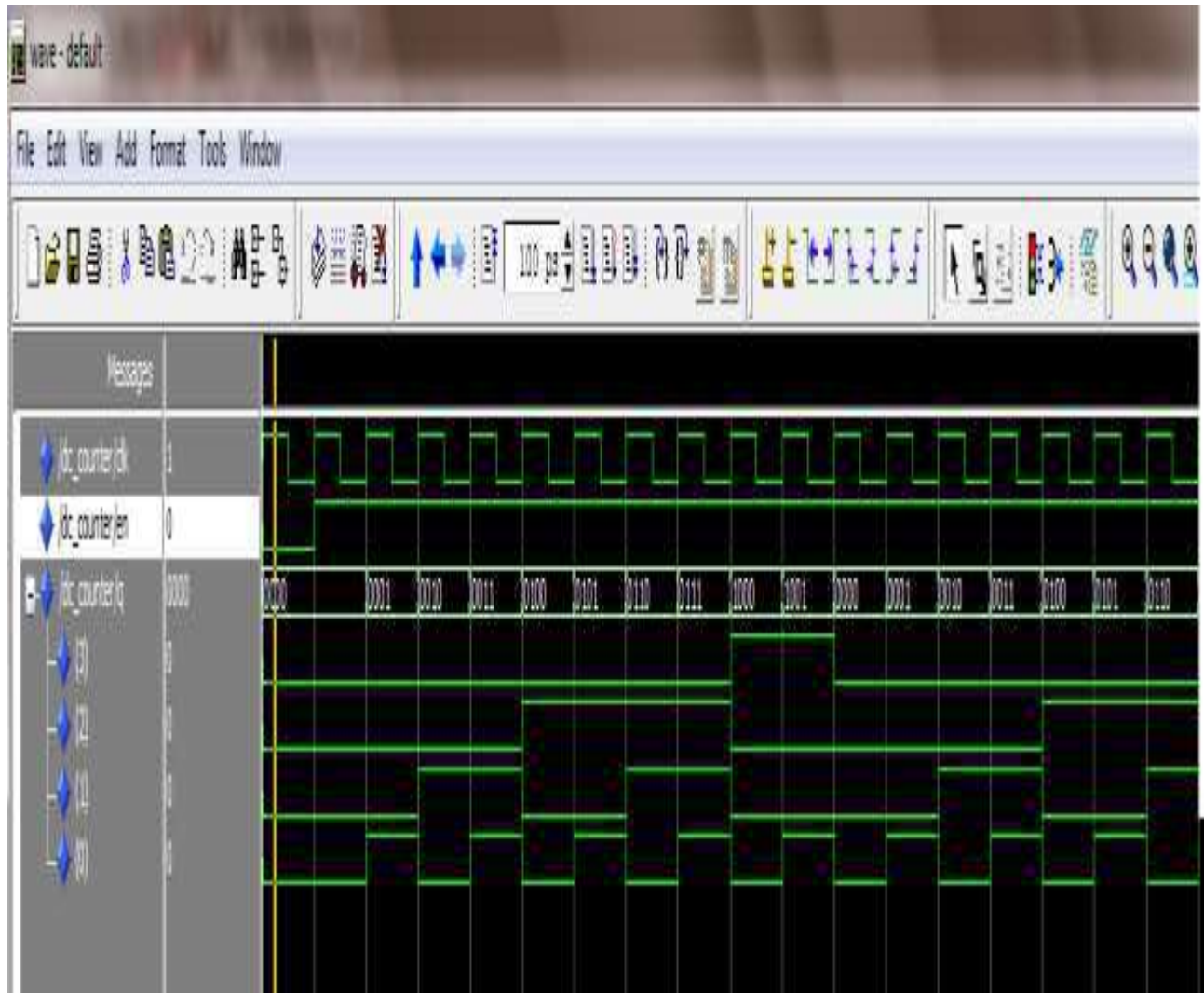
VHDL PROGRAM FOR DECADE COUNTER:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY DC_COUNTER IS
    PORT (CLK, CLR: IN STD_LOGIC;
          Q: INOUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END DC_COUNTER;

ARCHITECTURE BEHAVE OF DC_COUNTER IS
    BEGIN
        PROCESS (CLK,EN)
            BEGIN
                IF (CLR='0') THEN Q<="0000";
                ELSE
                    IF (CLK'EVENT AND CLK='1') THEN
                        IF(Q="1001")THEN
                            Q<="0000";
                        ELSE
                            Q<=Q+1;
                        END IF;
                    END IF;
                END IF;
            END PROCESS;
        END BEHAVE;
```

SIMULATION RESULTS:



RESULT: