SRM IST
DEPARTMENT OF ECE

ANALOG & DIGITAL ELECTRONICS LAB – (18CSS201J)

NAME :

SECTION :

REG NO :

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY RAMAPURAM CAMPUS

BONAFIDE CERTIFICATE

Registration No: ………………………………..

This is to certify that this is a bonafide record of practical work done by

…………………………………………………..….of. .... Semester

of ………………………………………………………… Department in the

……………………………………………………………..…...Laboratory,

during the year 20 - 20

SIGNATURE
Head Of the Department

SIGNATURE
Staff Incharge

Submitted for university examination held on …………………………

Signature of Internal Examiner-I

Signature of Internal Examiner-II

CONTENT

10.		**Design and Implement an A/D converter**		
11.		**HDL program for combinational circuits**		
12.		**HDL program for Binary counters**		
13.		**HDL program for Mod-10 counters**		

Experiment No:1a Date:

Design and Implementation of Half Wave and Full Wave Rectifiers using simulation package and demonstrate its working.

AIM

To design and analysis of half wave rectifier for the varying R and C components.

APPARATUS REQUIRED

S.no	Apparatus	Type	Range	Quantity
1)	Diode	1N4001		1
2)	Resistor		470Ω	1
3)	Capacitor		470µF	1
4)	Ac voltage source		4Vpk,60Hz	1

THEORY

The process of converting an alternating current into direct current is known as rectification. The unidirectional conduction property of semiconductor diodes (junction diodes) is used for rectification. Rectifiers are of two types: (a) Half wave rectifier and (b) Full wave rectifier. In a half-wave rectifier circuit, during the positive half-cycle of the input, the diode is forward biased and conducts. Current flows through the load and a voltage is developed across it. During the negative half cycle, it is reverse bias and does not conduct. Therefore, in the negative half cycle of the supply, no current flows in the load resistor as no voltage appears across it. Thus the dc voltage across the load is sinusoidal for the first half cycle only and a pure a.c. input signal is converted into a unidirectional pulsating output signal.

FORMULA:

$V_{rms}=V_m/2$

$V_{dc}=V_m/\pi$

$\gamma = \sqrt{(V_{rms}/V_{dc})^2-1}$

$\eta = P_{dc}/P_{ac} * 100\%$

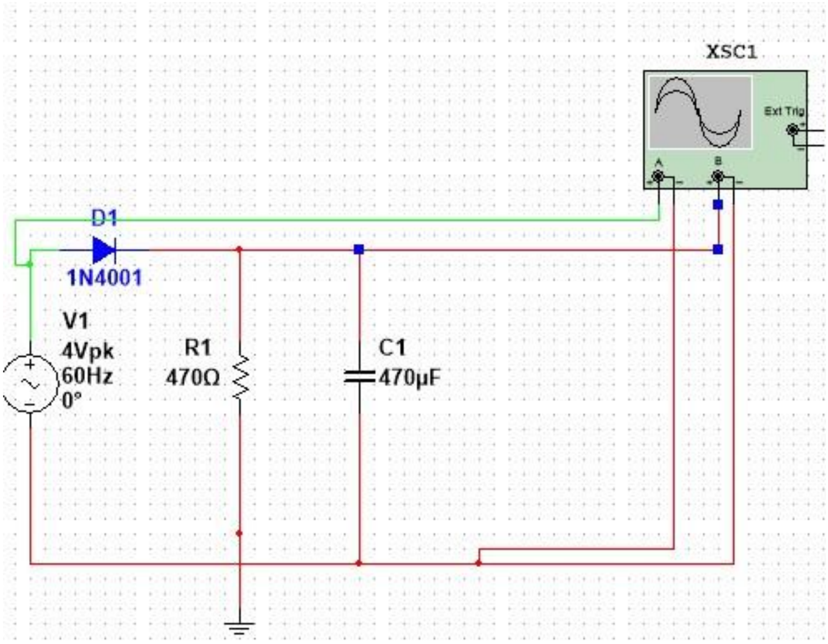
$P_{dc}= (V_{dc})^2 / R_L$

$P_{ac} = (V_{rms})^2 / R_L$

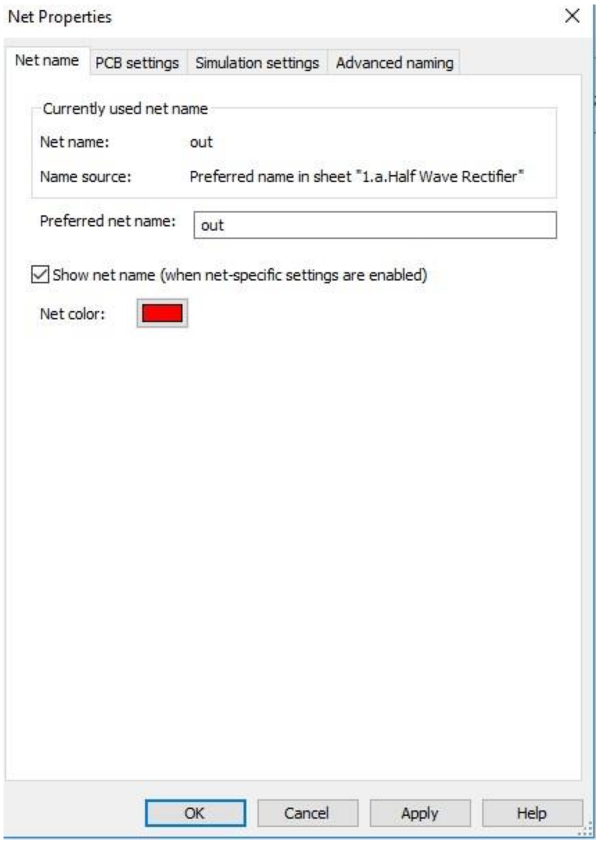
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as diode and select 1N4001 from component.
5. Select the Resistor and capacitor from master database and select group as basic and family as resistor and capacitor respectively, select the resistor and capacitor from components.
6. Place all the components, connect the components via wire to get the circuit diagram as below.

CIRCUIT DIAGRAM

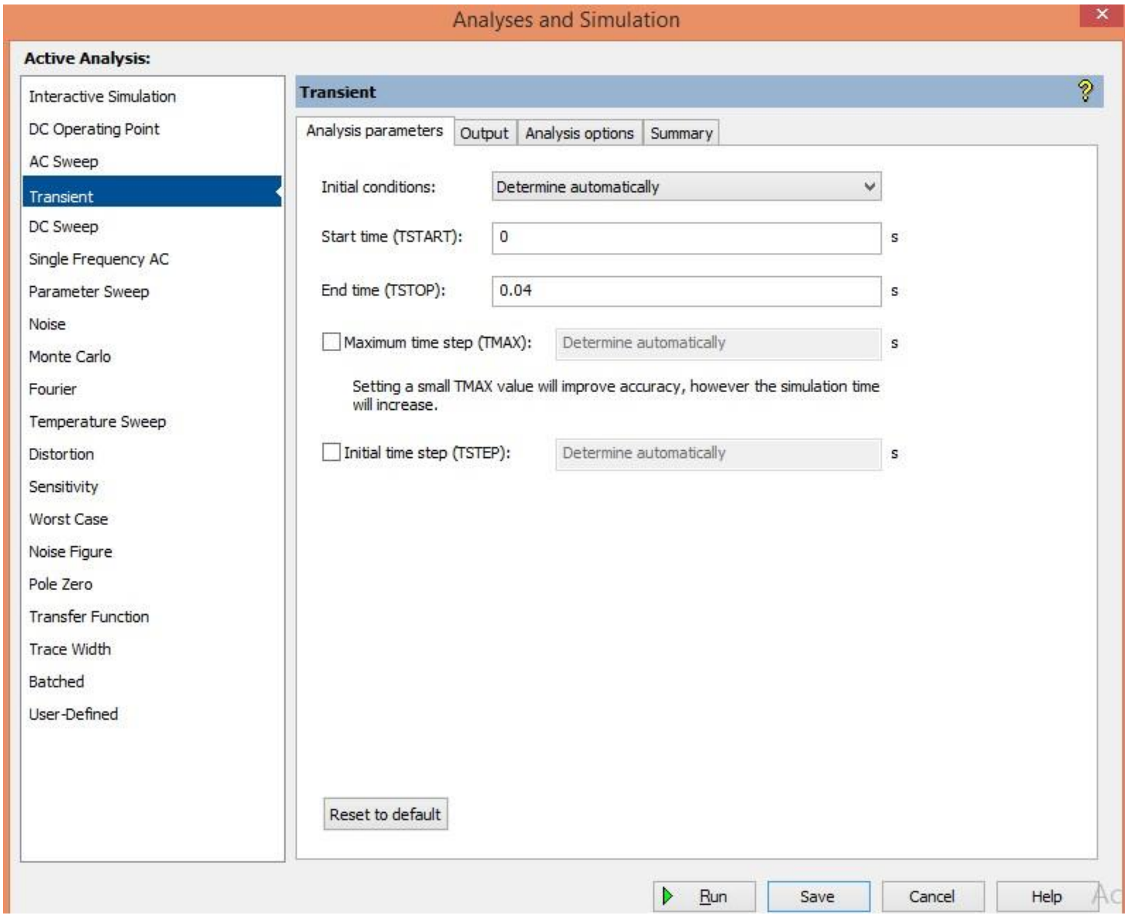


7.Double click the output wire and change the net properties as below.

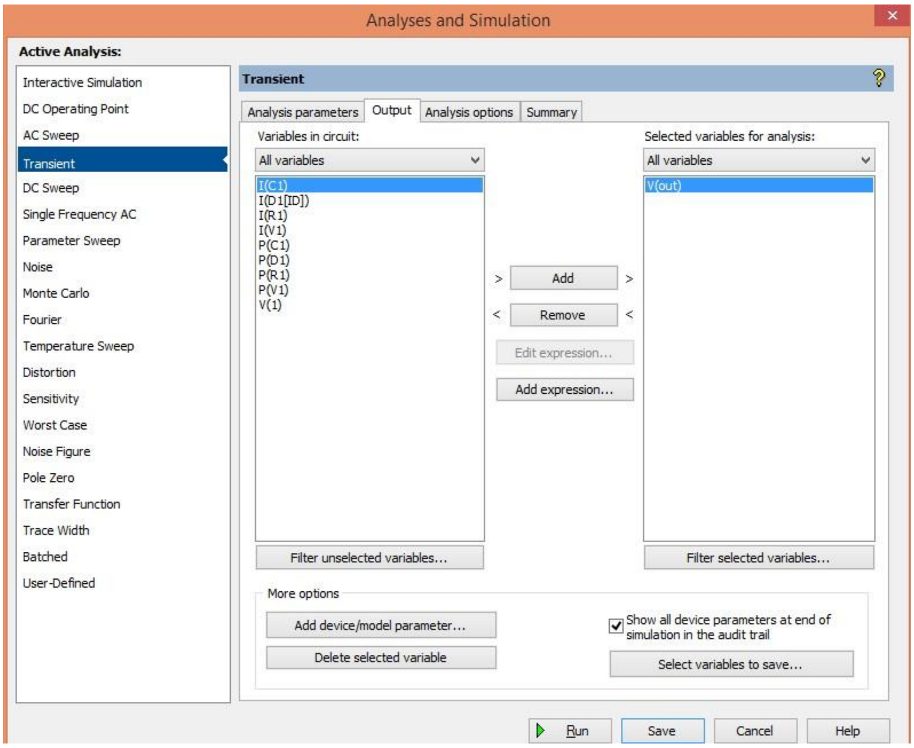


8.Select the option Simulate → Analysis and simulation → Transient.

9.In analysis parameter tab, set the following values found in the below figure.

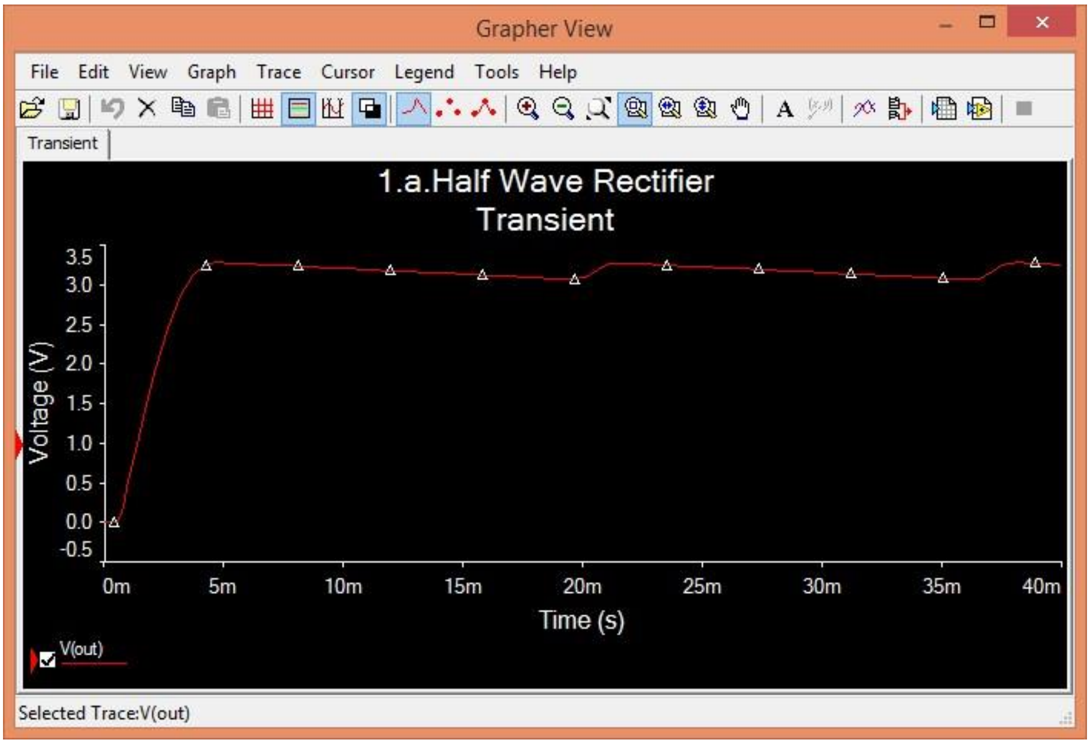


10.Go to the Output tab and select V(Out) variable and press add button to push the selected variable for analysis. Refer figure below.



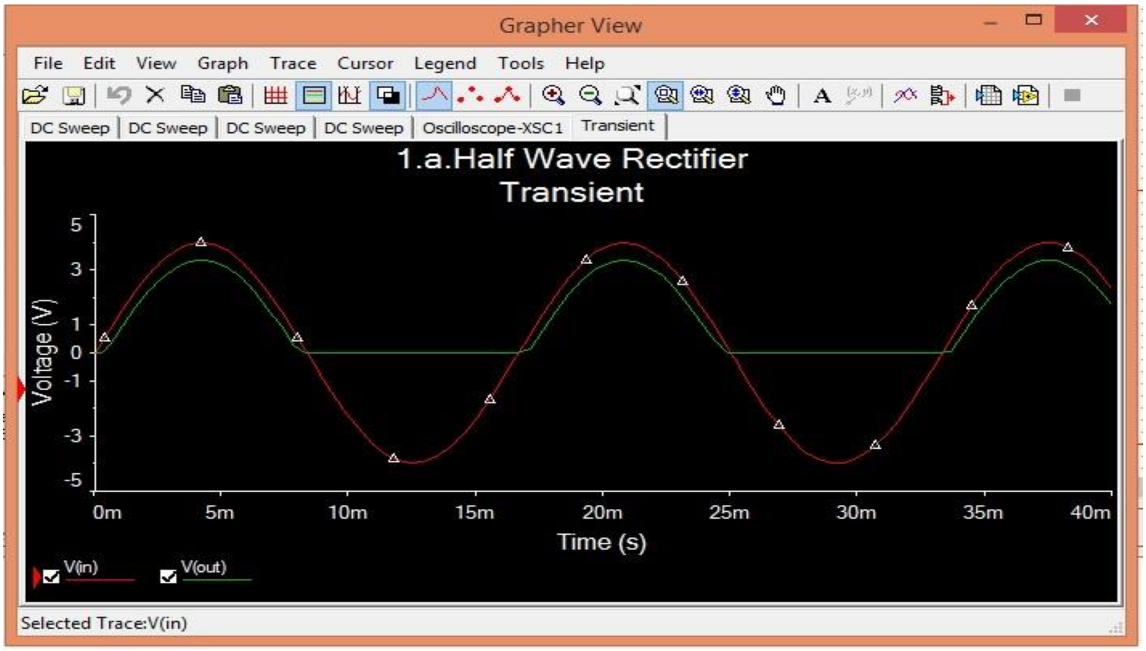
11.Press Run to see the simulation results in the Grapher View.

12.Result of transient analysis of half wave rectifier with filter.

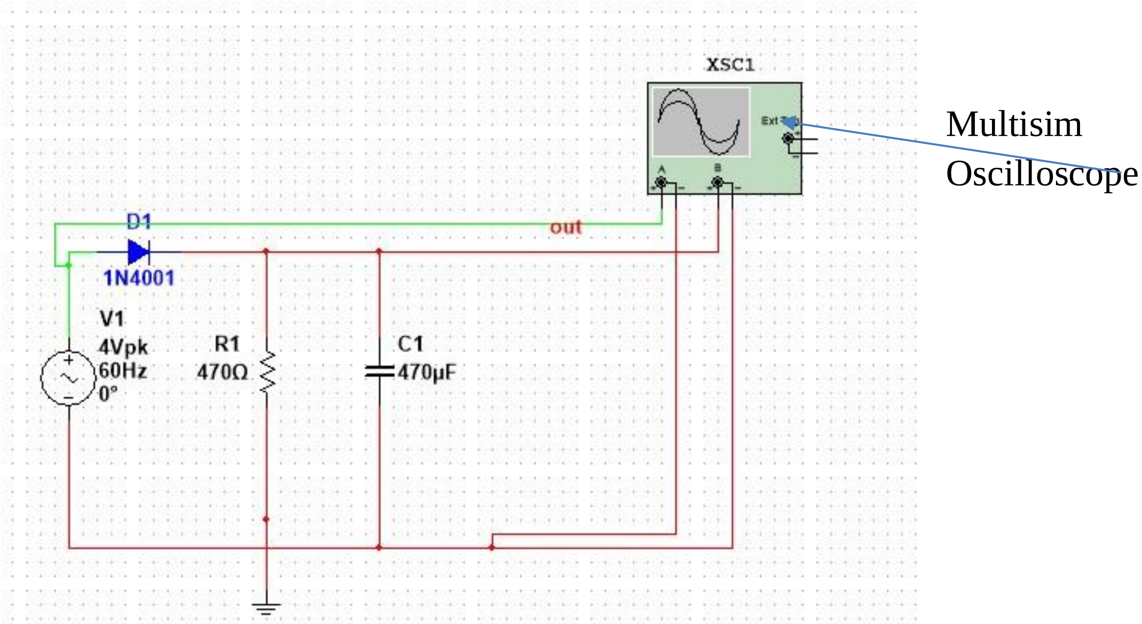


13.Remove the Capacitor from the circuit for rectifier without filter.

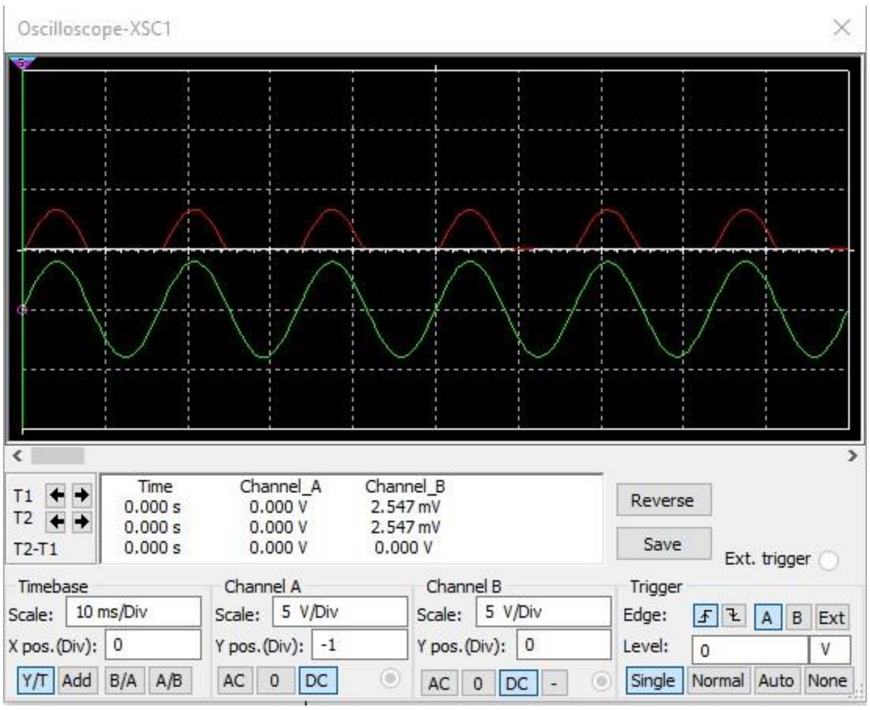
14.Run the simulation again to get the result of transient analysis of half wave rectifier without filter.

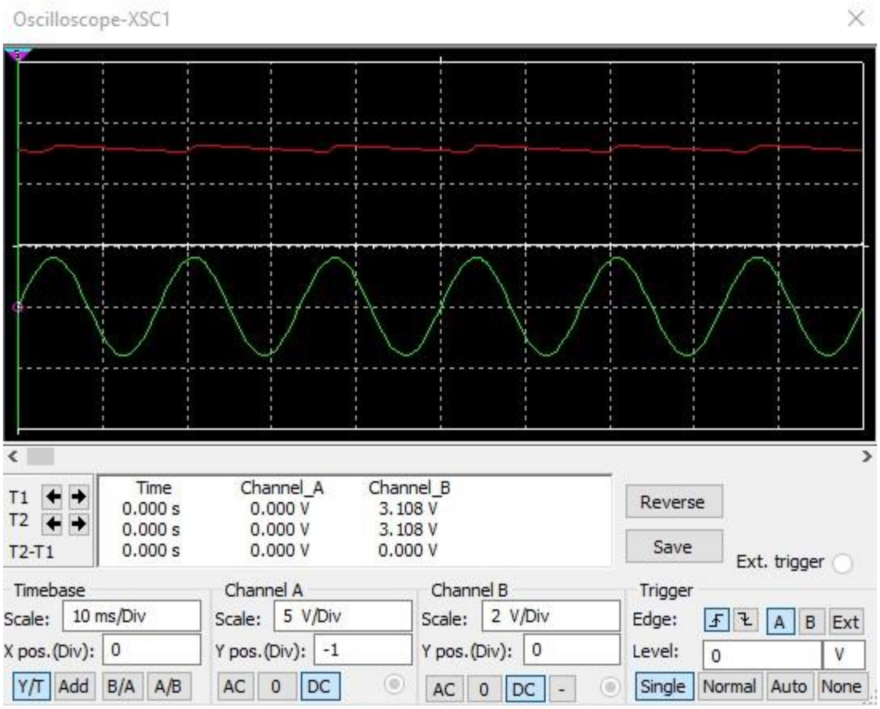


15.You can also view the resultant graph for Halfwave rectifier with and without filter through the Multisim oscilloscope.


XSC1
Multisim
Oscilloscope
Ext
A
B
D1
1N4001
out
V1
4Vpk
60Hz
0°
R1
470Ω
C1
470µF

WAVEFORMS:


Oscilloscope-XSC1
T1
T2
T2-T1
Time
0.000 s
0.000 s
0.000 s
Channel_A
0.000 V
0.000 V
0.000 V
Channel_B
2.547 mV
2.547 mV
0.000 V
Reverse
Save
Ext. trigger
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): -1
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



16. Note down the V_m value from the above oscilloscope Image for with and without filter.

17. Replace the R_L Values, note down $V_{m,}$ Do the necessary calculation and fill the tabulation as below.

TABULATION

R_L (KΩ)	With C Filter				Without Filter			
	Vac (Volts)	Vdc (Volts)	γ	η	Vac (Volts)	Vdc (Volts)	γ	η
0.1								
1								
10								
50								
100								
500								
1000								

Experiment No:1b Date:

Design and Implementation of Half Wave and Full Wave Rectifiers using simulation package and demonstrate its working.

AIM

To design and analysis of full wave rectifier for the varying R and C component.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	Diode	1N4001		4
2)	Resistor		1kΩ	1
3)	Capacitor		100µF	1
4)	AC voltage source		$4V_{pk}$,60 Hz	1

THEORY

Another type of circuit that produces the same output as a full-wave rectifier is that of the Bridge Rectifier. This type of single-phase rectifier uses 4 individual rectifying diodes connected in a "bridged" configuration to produce the desired output but does not require a special centre tapped transformer, thereby reducing its size and cost. The single secondary winding is connected to one side of the diode bridge network and the load to the other side. The 4 diodes labeled D arranged in "series pairs" with only two diodes conducting current during each half cycle. During the positive half cycle of the supply, diodes D1 and D2 conduct in se D3 and D4 are reverse biased and the current flows through the load as shown below . During the negative half cycle of the supply, diodes D3 and D4 conduct in series, but diodes D1 and D2 switch of as they are now reverse biased. The current flowing through the load is the same direction as before.

FORMULA:

$V_{rms} = V_{r(P-P)}/2\sqrt{3}$

$V_{dc} = V_m - V_{r(P-P)}$ γ

$= V_{rms}/V_{dc}$

$\eta = P_{dc}/P_{ac} * 100\%$

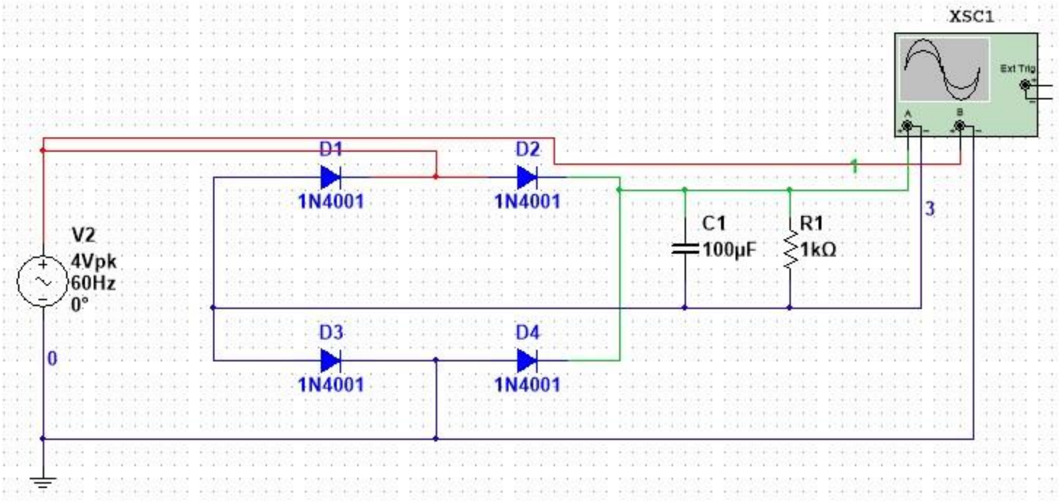
$P_{dc} = (V_{dc})^2 / R_L$

$P_{ac} = (V_{rms})^2 / R_L$

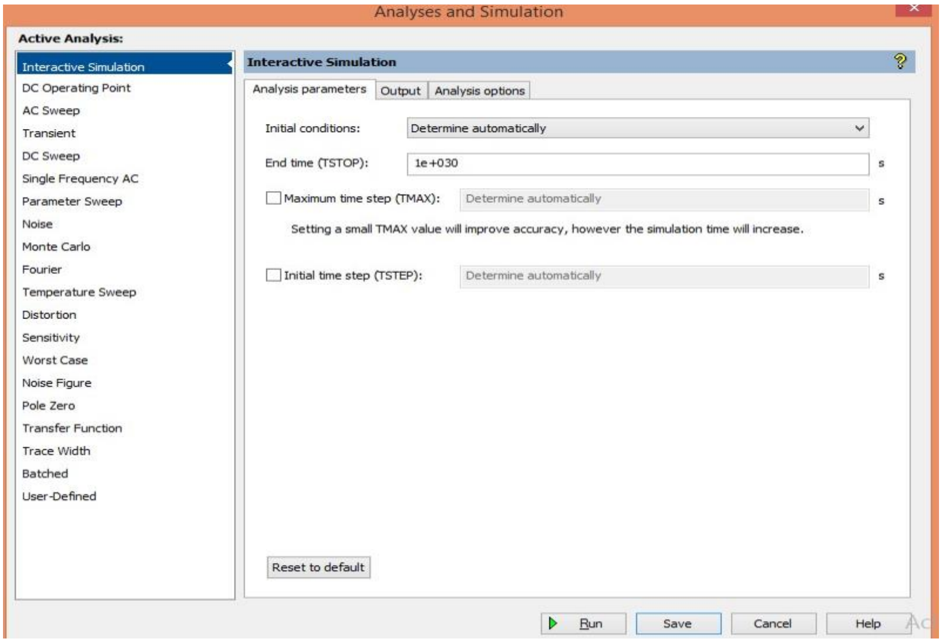
PROCEDURE

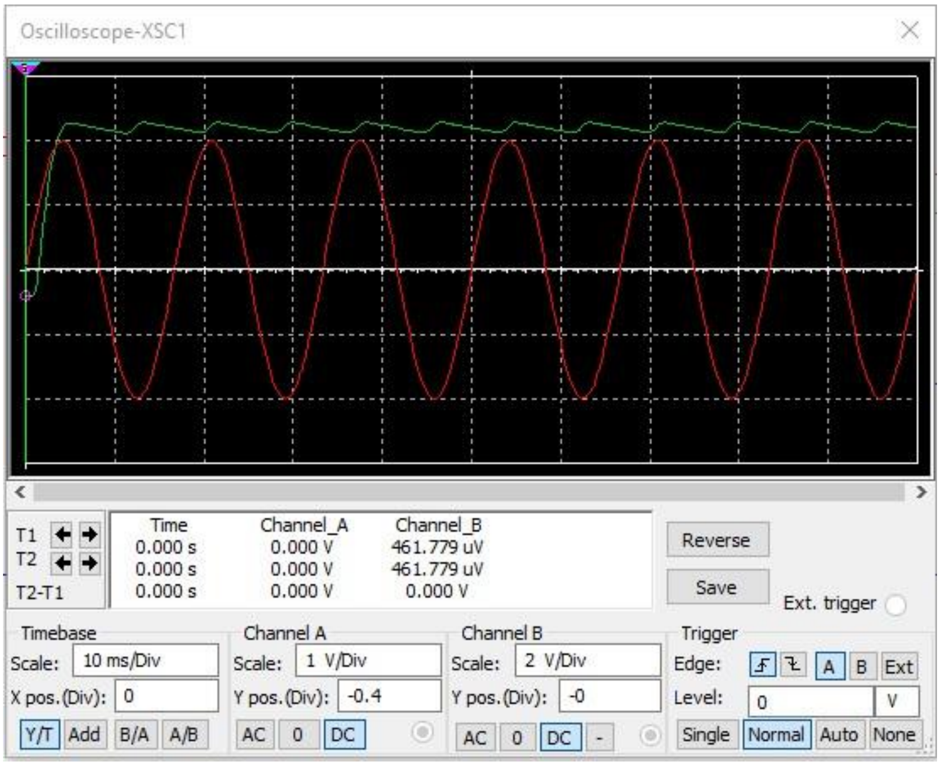
1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing Ctrl+W.
4. Select Master database, select group as diode and select 1N4001 from component.
5. Select the Resistor and capacitor from master database and select group as basic and family as resistor and capacitor respectively, select the resistor and capacitor from components.
6. Place all the components, connect the components via wire to get the circuit diagram as below.

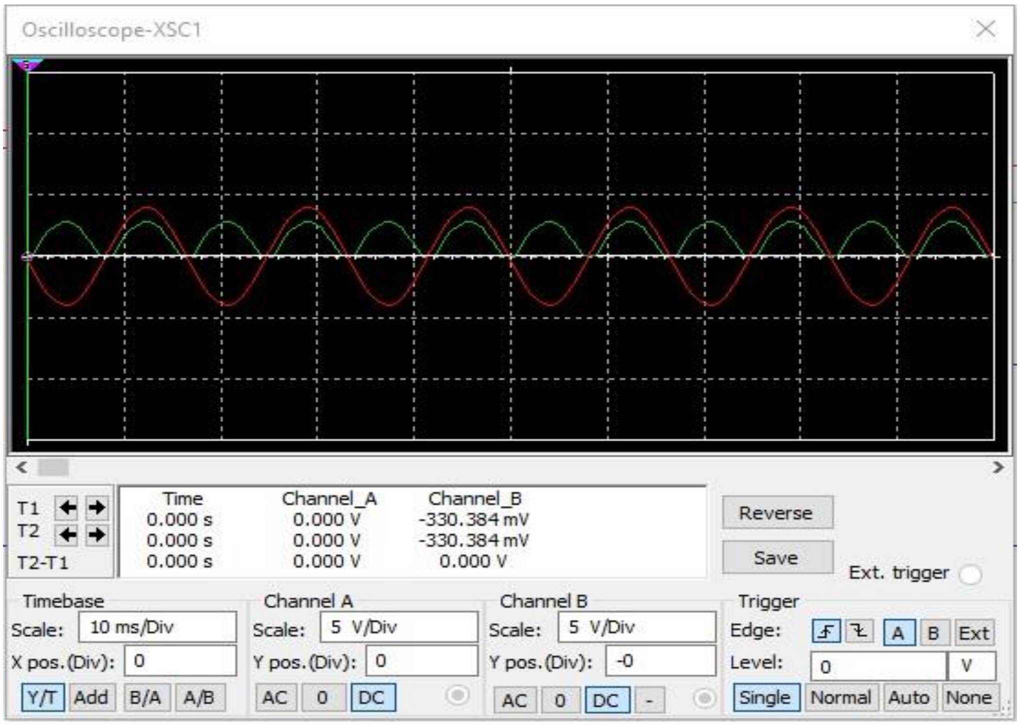
CIRCUIT DIAGRAM



7. Select the option Simulate → Analysis and simulation → interactive simulation.
8. In analysis parameter tab, set the default values found in the below figure and hit Run then view the output using oscilloscope.







9. Note down the V_m , $V_{r(P-P)}$/ value from the above oscilloscope.

10. Replace the R_L Values, note down V_m , $V_{r(P-P)}$/, Do the necessary calculation and fill the tabulation as below.

11. Remove the Capacitor from the circuit for rectifier without filter.

12.Repeat Step 9 and 10 to complete the tabulation.

TABULATION

R_L (KΩ)	With C Filter				Without Filter			
	Vac (Volts)	Vdc (Volts)	γ	η	Vac (Volts)	Vdc (Volts)	γ	η
0.1								
1								
10								
50								
100								
500								
1000								

RESULT

Thus, the characteristics of Half Wave, Full Wave were studied.

Experiment No:2 Date:

Design and implement a Schmitt trigger using Op-Amp using a simulation package and demonstrate its working.

AIM

Design and implement a Schmitt trigger using Op-Amp using a simulation package and demonstrate its working.

APPARATUS REQUIRED

S.NO	APPARATUS	TYPE	RANGE	QUANTITY
1)	OP-AMP	741		1
2)	Resistor		3kΩ,10kΩ,100kΩ	3
3)	Function generator		1Hz	1
4)	DC power		12V,12V,2.2V	3
5)	oscilloscope			1

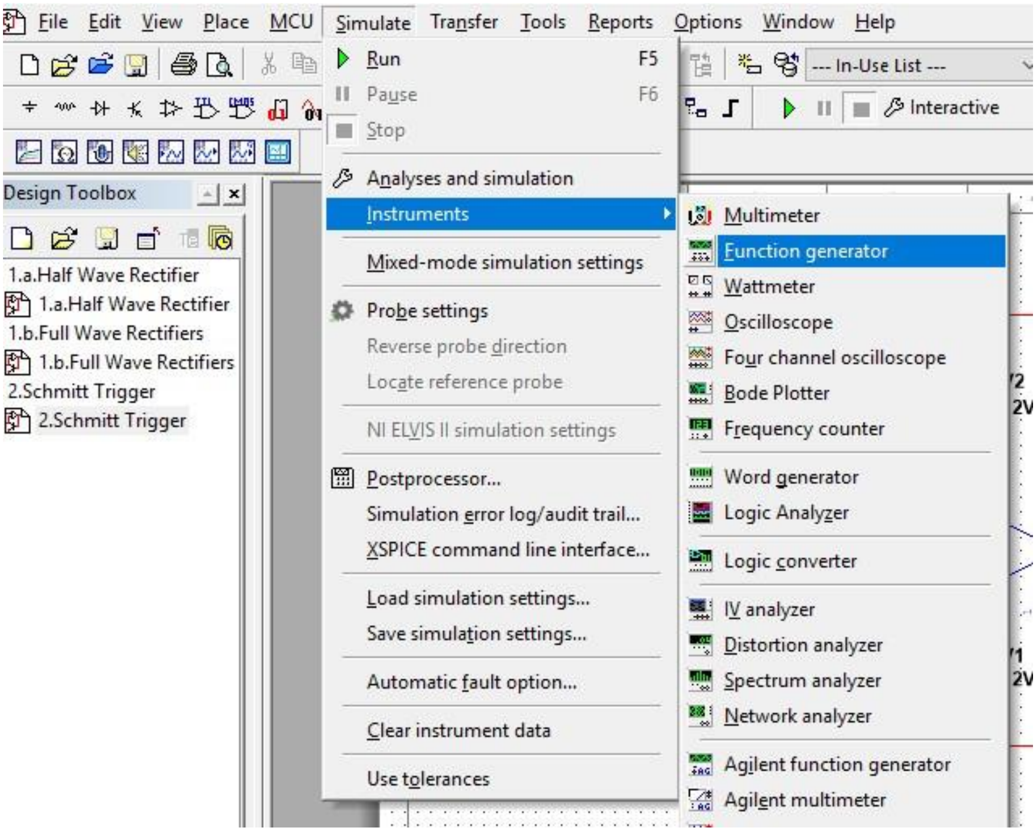
THEORY

A Schmitt trigger circuit is also called a regenerative comparator circuit. The circuit is designed with a positive feedback and hence will have a regenerative action which will make the output switch levels. Also, the use of positive voltage feedback instead of a negative feedback, aids the feedback voltage to the input voltage, instead of opposing it. The use of a regenerative circuit is to remove the difficulties in a zero crossing detector circuit due to low frequency signals and input noise voltages.

Shown below is the circuit diagram of a Schmitt trigger. It is basically an inverting comparator circuit with a positive feedback. The purpose of the Schmitt trigger is to convert any regular or irregular shaped input waveform into a square wave output voltage or pulse. Thus, it can also be called a squaring circuit.

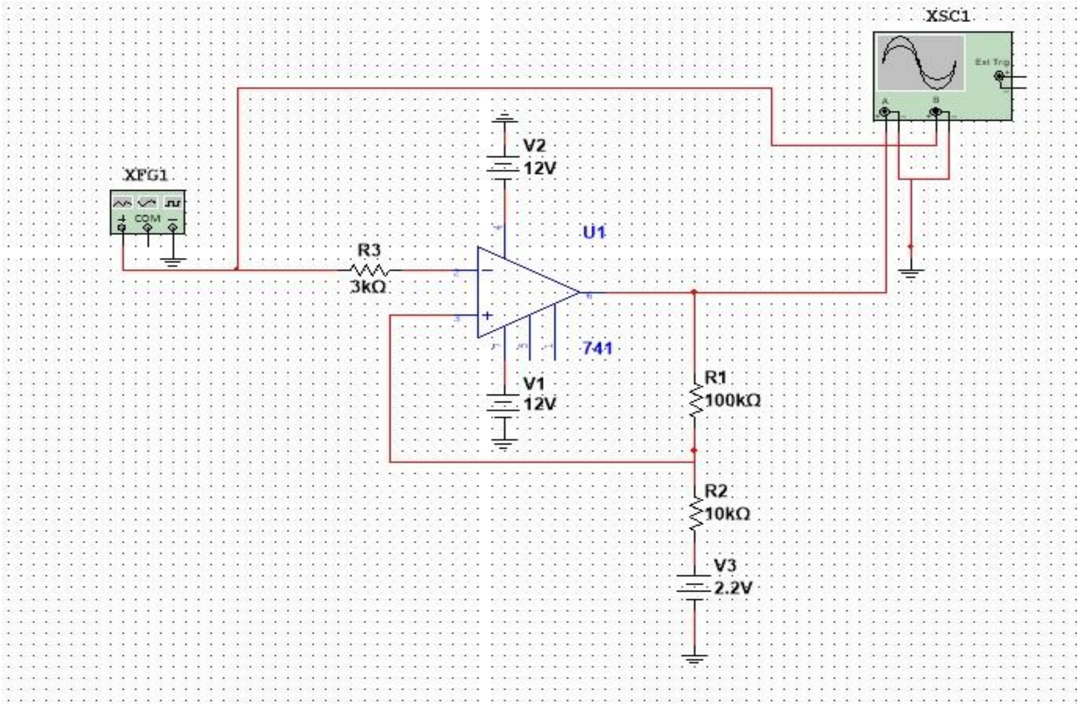
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as Analog and select OP-AMP 741 from component.
5. Select the Resistor and voltage from master database and select group as basic and source as resistor and voltage respectively from components.
6. Place the function generator (refer figure below) and select Sinusoid Signal with the frequency of 1Hz by double clicking the function generator.

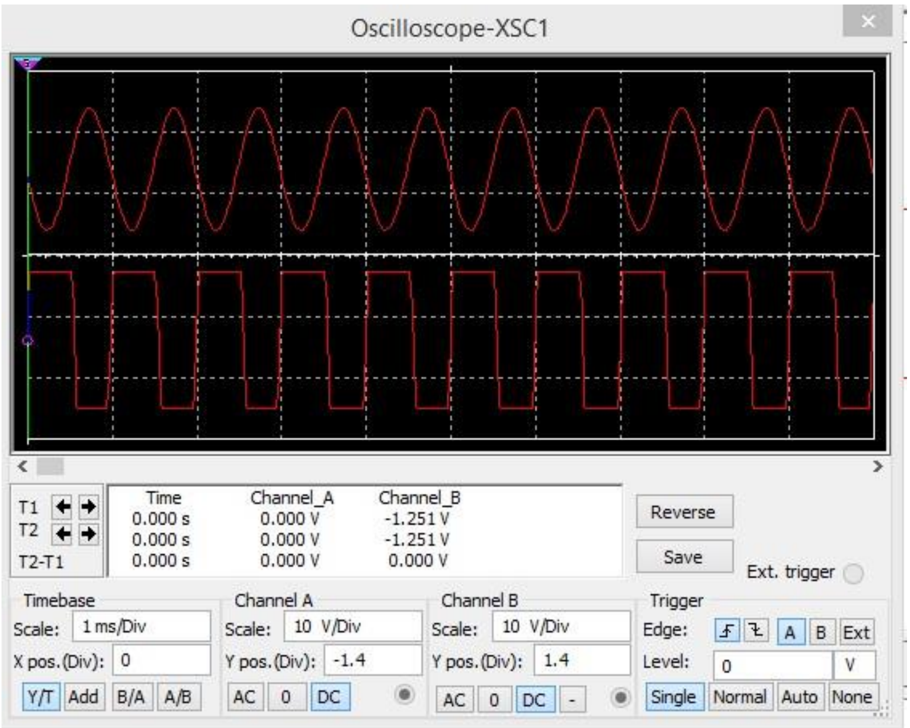


7. Place Oscilloscope in the circuit (Refer the Image above).
8. Place all the components, connect the components via wire to get the circuit diagram as below.

CIRCUIT DIAGRAM



9. Run the simulation to view the Oscilloscope results.



10.Note down the result and note down the Peak to Peak Voltage, Frequency of the square wave generated and tabulate the result below.

TABULATION

I/P VOLTAGE	I/P SIGNAL FREQUENCY	OUTPUT VOLTAGE	OUTPUT FREQUENCY
10	1Hz		

RESULT

Thus the Schmitt trigger was designed and the output voltage was tabulated.

Experiment No: 3 Date:

Design and implement a rectangular waveform generator (Op-Amp relaxation oscillator) using a simulation package and demonstrate the working of it.

AIM

To design and implement a rectangular waveform generator (Op-Amp relaxation oscillator) using a simulation package and demonstrate the working of it.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	OP-AMP	741		1
2)	Resistor		50kΩ,35kΩ,30kΩ	3
3)	Capacitor		.01µF	1
4)	Voltage source		12V	2
5)	oscilloscope			1

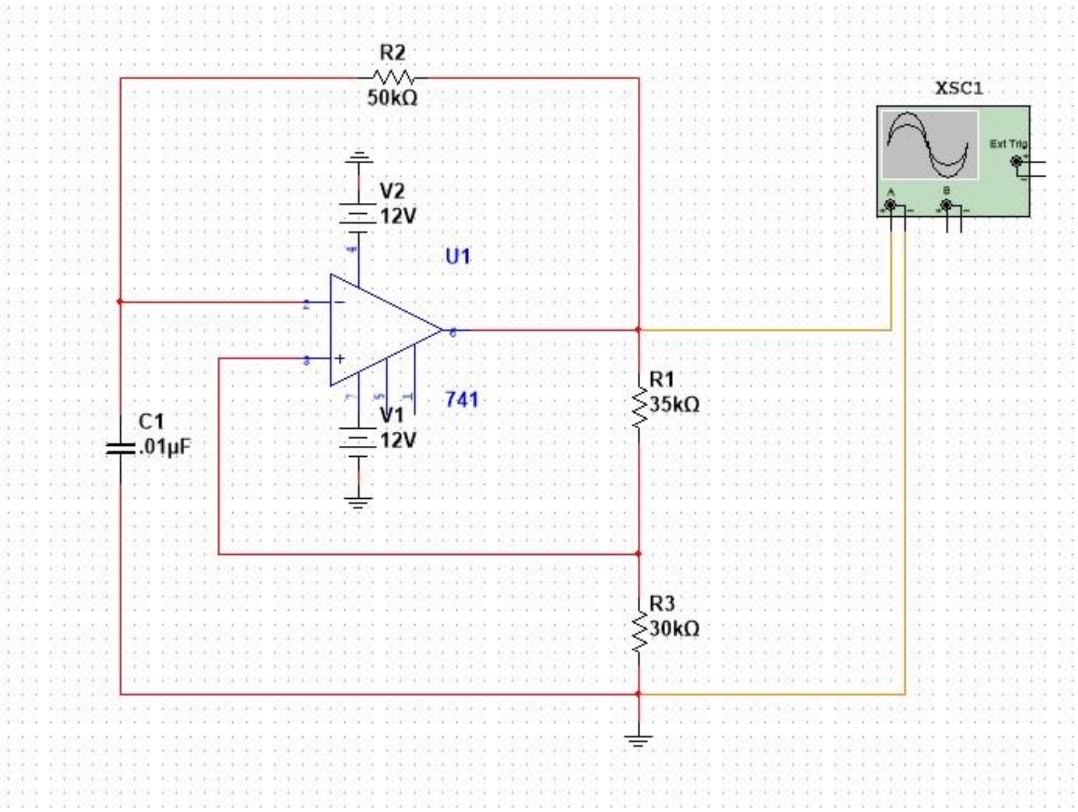
THEORY

Rectangular Waves are generated when the Op-Amp is forced to operate in the saturation region. That is, the output of the op-amp is forced to swing respectively between +Vsat And -Vsat resulting in the generation of square wave. The square wave generator is also called a free-running or astable Multivibrator Assuming the voltage across capacitor C is zero at the instant the d.c Supply voltage at +Vcc and VEE are applied. Initially the capacitance C acts, as a short circuit. The gain of the Op-Amp is very large hence V1 drives the output of the Op-Amp to its saturation.

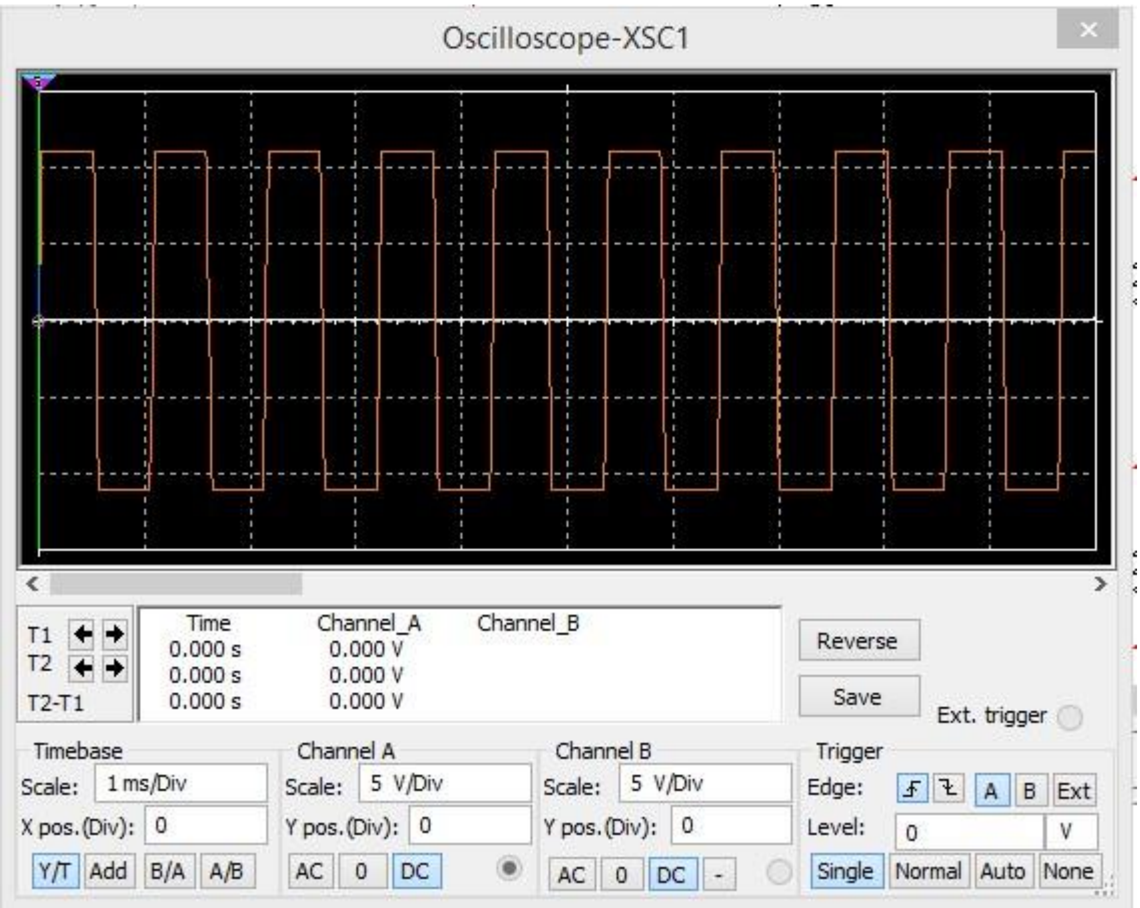
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as Analog and select OP-AMP 741 from component.
5. Select the Resistor and capacitor from master database and select group as basic and family as resistor and capacitor respectively, select the resistor and capacitor from components
6. Select the voltage source from the source group and set as 12V.
7. Place Oscilloscope in the circuit.
8. Place all the components, connect the components via wire to get the circuit diagram as below.

CIRCUIT DIAGRAM



9. Run the simulation.



10.Note down the result and note down the Peak to Peak Voltage, Frequency of the wave generated and tabulate the result below.

TABULATION

FREQUENCY	AMPLITUDE(V)	TIME(ms)

RESULT

Thus, the rectangular wave generator was designed, and the corresponding values are tabulated.

Experiment No:4 Date:

Design and implementation of transistor as a switch

AIM:

1. To observe the action of a Transistor as an electronic switch.
2. To measure the voltage across the transistor when it is ON and when it is OFF.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	Transistor	BC107BP		1
2)	Resistor		1kΩ,10kΩ	1(each)
3)	Multimeter			1
4)	DC power		5V	2(each)
5)	Switch			1
6)	Probe			1

THEORY

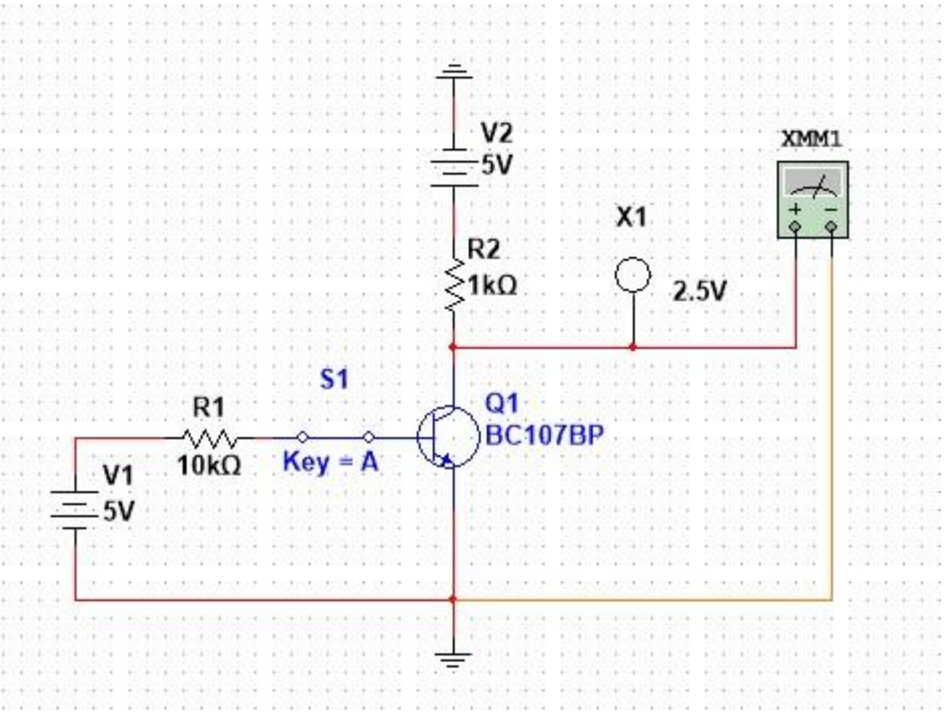
The computers of today do not process numbers in the base 10 (i.e., 0, 1, 2, 3, ...,9). Computers instead use binary logic of base 2 (0 and 1) to perform their functions. One fundamental circuit is the transistor switch, also known as an inverter. Here, a transistor connected in a common-emitter fashion inverts a signal. That is, if a high-input signal is applied, a low-output signal is created. If a low-input signal is applied, then a highoutput signal is created.

In a transistor switch circuit, a voltage level applied to the base terminal will control the potential at the collector. In this fashion, the transistor can be used to turn on or off circuitry connected to the collector. This common-emitter circuit is being switched from cutoff to saturation. In this experiment, a transistor will be connected to demonstrate this switching ability.

PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as Transistor and select BC107BP from component.
5. Select the Resistor and voltage from master database and select group as basic and source as resistor and voltage respectively from components.
6. Select the Switch from master database and select group as basic to get components.
7. Place all the components, connect the components via wire to get the circuit diagram as below.
8. Select the probe from master database and select group as Indicators, select the family as probe to get component. Place it as per circuit diagram.
9. Place multimeter from instruments to know the voltage when the switch is ON and OFF.

CIRCUIT DIAGRAM:



10. Run the simulation to view the multimeter voltage results voltage when the switch is ON and OFF.
11. Tabulate the results below.

TABULATION

Switch	Status of the Probe	Multimeter voltage

ON		
OFF		

RESULT

Thus, the transistor as a switch was designed and the output voltage and status of the Probe was tabulated.

Experiment No:5 Date:

Design CMOS Inverter and measure its propagation delay for both the rising edge and the falling edge

AIM:

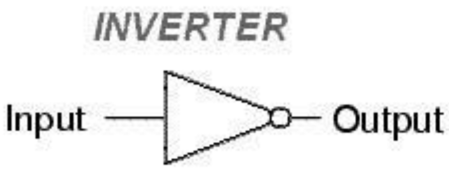
To Design CMOS Inverter and measure its propagation delay for both the rising edge and the falling edge.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	Transistor	Pmos		1
2)	Transistor	Nmos		1
3)	Pulse Voltage			1
4)	Oscilloscope			1

THEORY

The inverter is universally accepted as the most basic logic gate doing a Boolean operation on a single input variable. Fig.1 depicts the symbol, truth table and a general structure of a CMOS inverter. As shown, the simple structure consists of a combination of an pMOS transistor at the top and a nMOS transistor at the bottom.



Input	Output
1	0
0	1

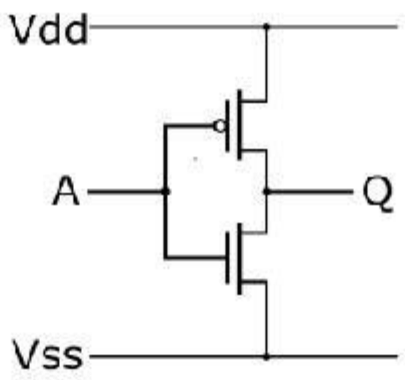


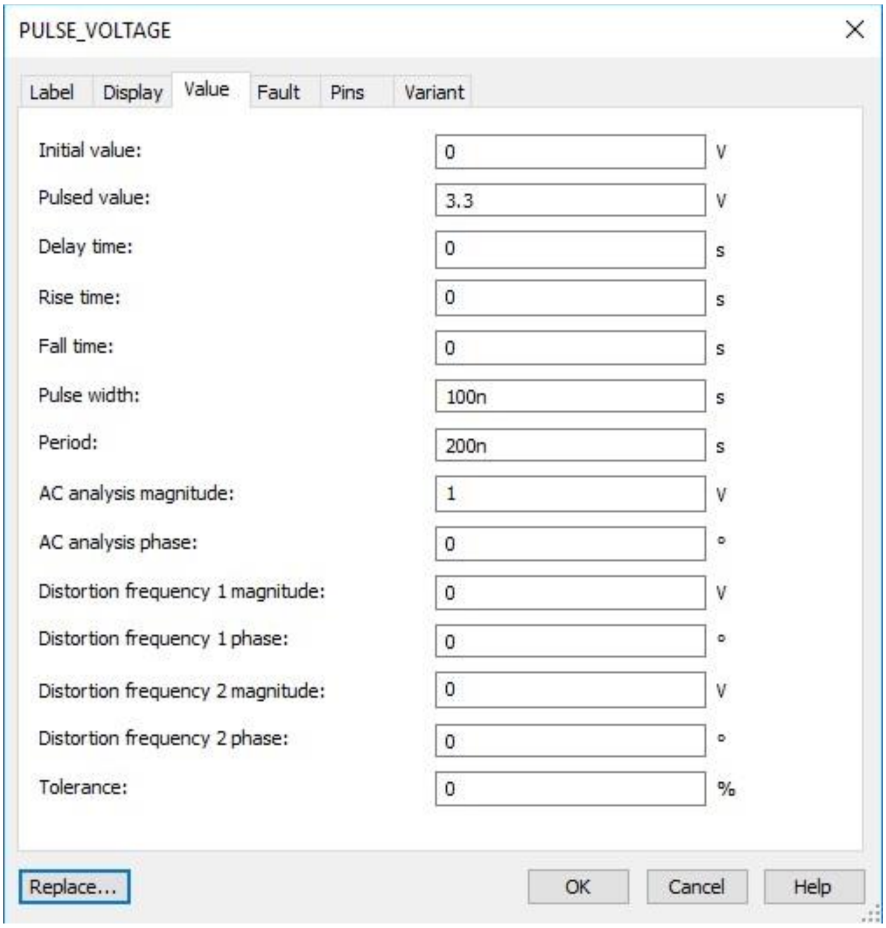
Fig.1: Symbol, circuit structure and truth table of a CMOS inverter

CMOS is also sometimes referred to as complementary-symmetry metal–oxide–

semiconductor. The words "complementary-symmetry" refer to the fact that the typical digital design style with CMOS uses complementary and symmetrical pairs of p-type and n-type metal oxide semiconductor field effect transistors (MOSFETs) for logic functions. Two important characteristics of CMOS devices are high noise immunity and low static power consumption. Significant power is only drawn while the transistors in the CMOS device are switching between on and off states. Consequently, CMOS devices do not produce as much waste heat as other forms of logic, for example transistor-transistor logic (TTL) or NMOS logic, which uses all n-channel devices without p-channel devices.

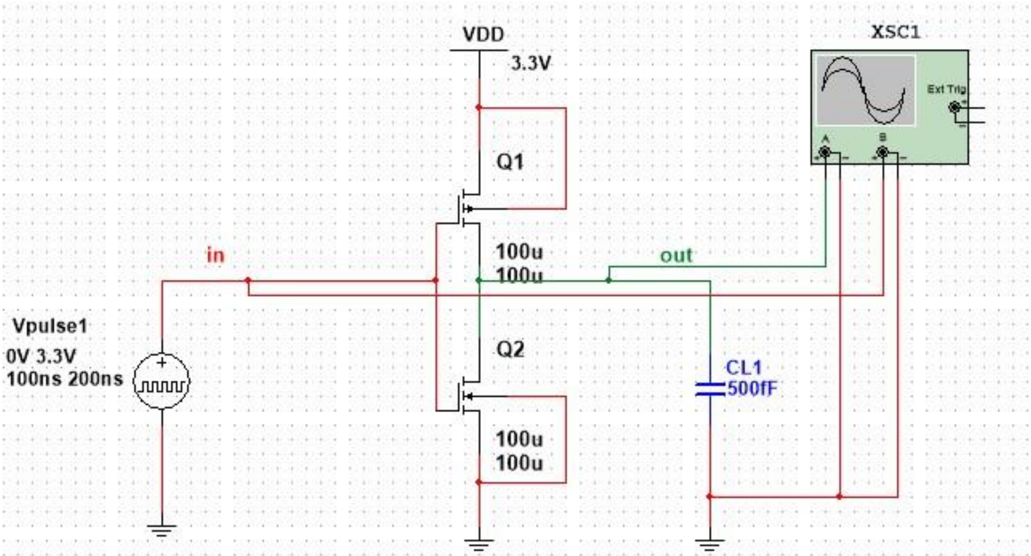
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as Transistor, family as Transistors Virtual select MOS_N_4T from component.
5. Select Master database, select group as Transistor, family as Transistors Virtual select MOS_P_4T from component.
6. Select Master database, select group as Source, family as Signal Voltage Sources, select Pulse Voltage from component.
7. Double click the pulse voltage and make the settings as below.



8. Place all the components, connect the components via wire to get the circuit diagram as below.
9. Place Oscilloscope from instruments to see the results.

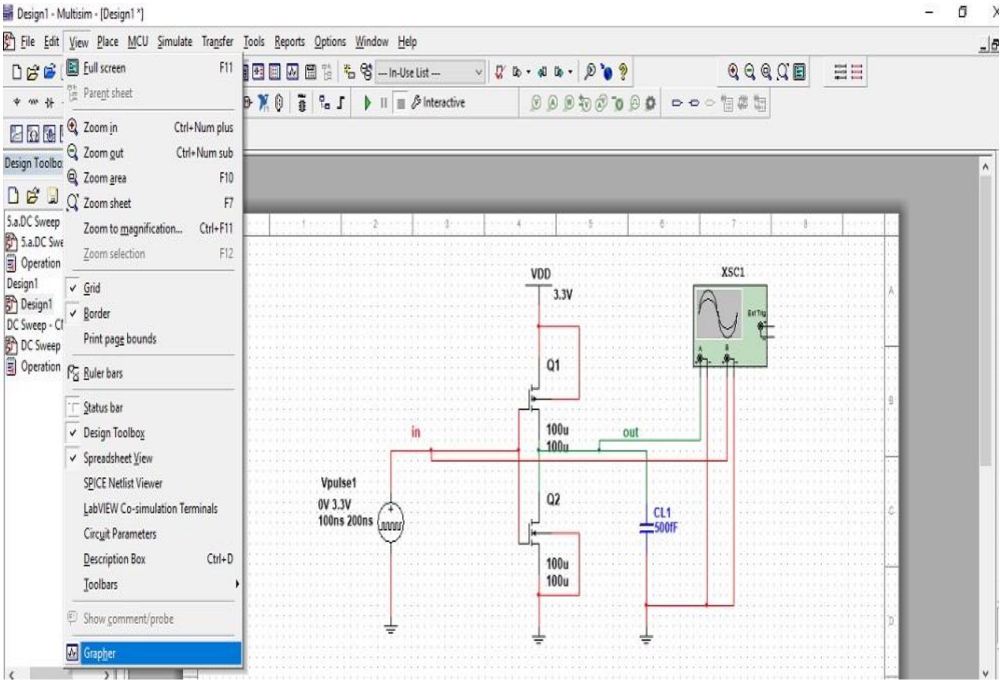
CIRCUIT DIAGRAM:

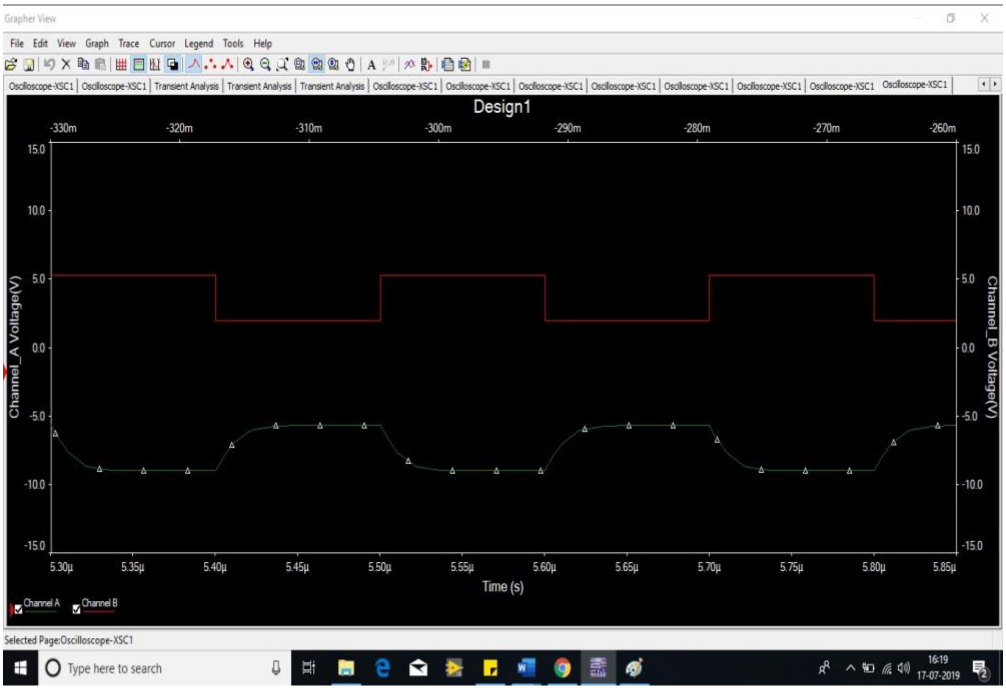


10. Run the simulation to view the Oscilloscope.

11. If you look at the input and output curve, output is inverted w.r.t input.

12. Go to View and select grapher view to measure the Propagation delay.





13. Tabulate the results below.

TABULATION

Input Voltage(V)	Output Voltage(V)	Propagation Delay
0		
3.3		

RESULT

Thus, the Design of CMOS Inverter and measure its propagation delay for both the rising edge and the falling edge was tabulated.

Experiment No:6 Date:

6.a. Design and implementation of Binary to gray code converters using logic gates

AIM:

1. To design and implementation of Binary to gray code converters using Multisim. 2. Hardware Implementation of the same with NI Analog Discovery 2.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	IC	IC 7486		1
2)	LED			4
3)	Switch			4
4)	DC Power Source			1
5)	NI Analog Discovery 2			
6)	Wires			As Required

THEORY

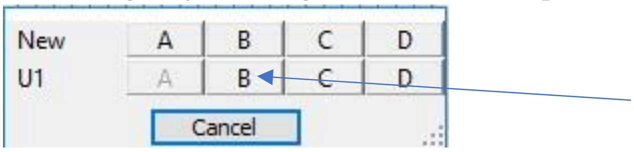
The logical circuit which converts binary code to equivalent gray code is known as binary to gray code converter. The gray code is a non-weighted code. The successive gray code differs in one bit position only that means it is a unit distance code. It is also referred as cyclic code. It is not suitable for arithmetic operations. It is the most popular of the unit distance codes. It is also a reflective code. An n-bit Gray code can be obtained by reflecting an n-1 bit code about an axis after 2n-1 rows, and putting the MSB of 0 above the axis and the MSB of 1 below the axis.

PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as TTL, select Family as 74STD,Select 7486N from component.

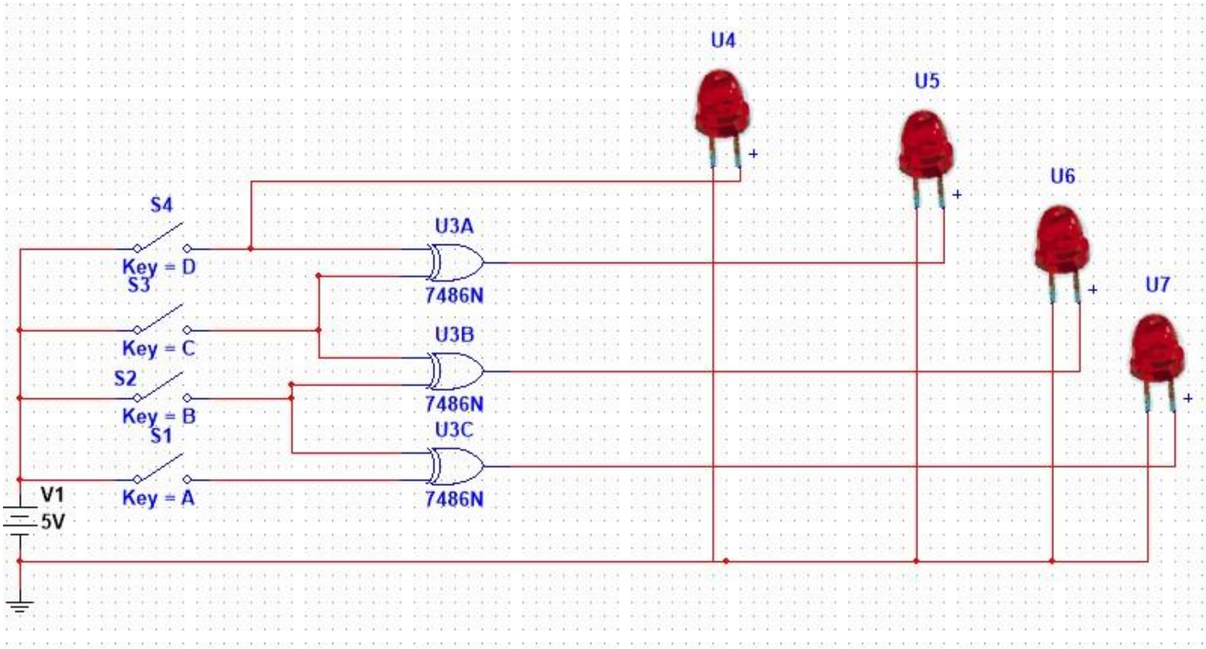
5. Multi-section component being placed, a dialog box appears as shown in the figure below:

6. Click A, you will get one Xor gate. It will be labelled as U1A.
7. Again you will get Multi-section option in that you select B from label U1.



8. Like wise try to get another Xor gate.
9. Select the Switch from master database and select group as basic to get components.
10. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -DC Power.
11. Double the DC Power to change Voltage as 5V.
12. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -Ground.
13. Select the LED from master database, Group-Basic, Family-3D_Virtual, Component Led1_Red.
14. Place all the components, connect the components via wire to get the circuit diagram as below.

CIRCUIT DIAGRAM:

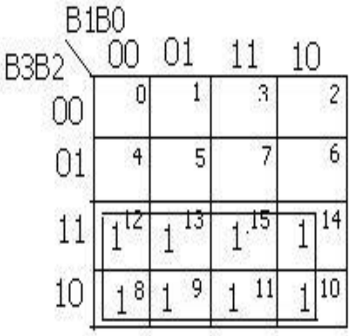


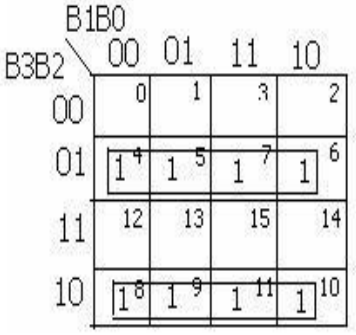
15. Run the simulation change the value of the switches to verify the truth table.

Truth Table

BINARY				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

$G3 = \sum(8,9,10,11,12,13,14,15)$ $G2 = \sum(4,5,6,7,8,9,10,11)$





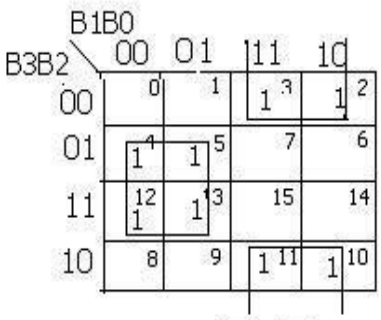
$G3 = B3$

$G2 = \overline{B3}B2 + B3\overline{B2}$

$G2 = B3 \oplus B2$

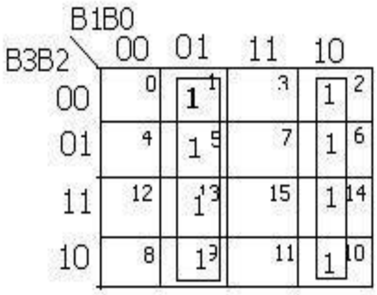
G1= ∑(2,3,4,5,10.,11,12,13)

G0 = ∑(1,2,3,5,6,9,10,13,14)



$G1= B2\overline{B1}+\overline{B2}B1$

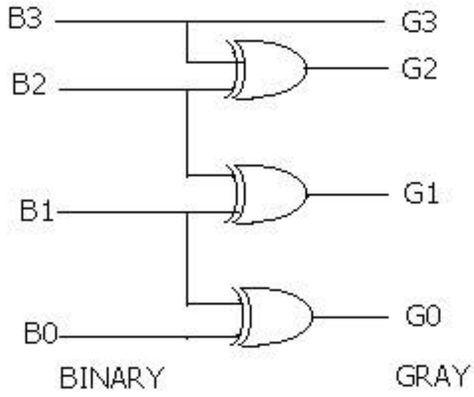
$G1= B1 \oplus B2$



$G0= \overline{B1}B0 + B1\overline{B0}$

$G0= B1 \oplus B0$

Binary to Gray code converter Using XOR Gates Only



$G3=B3$

$G2= \overline{B3}B2 + B3\overline{B2}$

$G2= B3 \oplus B2$

$G1= B2\overline{B1}+\overline{B2}B1$

$G1= B1 \oplus B2$

$G0= \overline{B1}B0 + B1\overline{B0}$

$G0= B1 \oplus B0$

6.b.Design and implementation of Gray to Binary code converters using logic gates

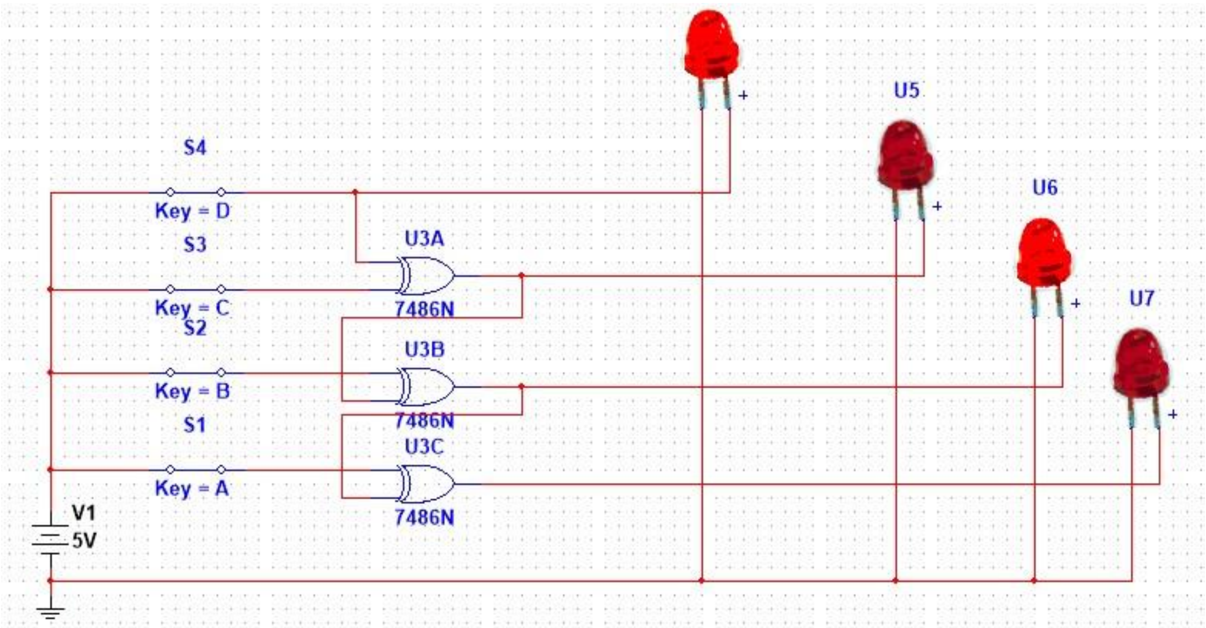
AIM:

1. To design and implementation of Gray to Binary code converters using Multisim. 2. Hardware Implementation of the same with NI Analog Discovery 2.

PROCEDURE:

Follow the same procedure as for Binary to gray to complete the circuit diagram as below.

CIRCUIT DIAGRAM:

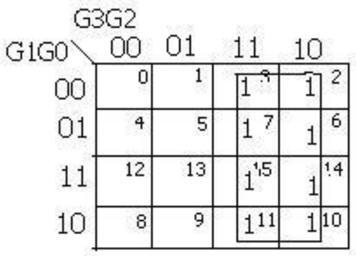


Truth Table

GRAY CODE	ƷINARY CODE

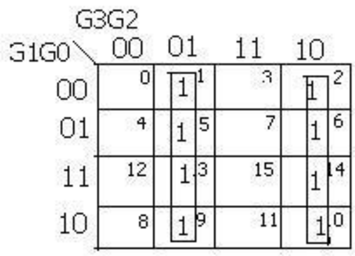
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

K MAP FOR B3



$B3=G3$

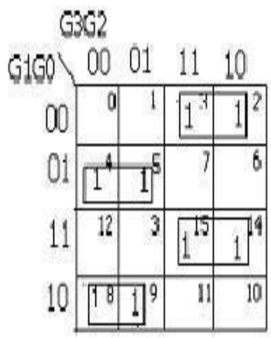
K MAP FOR B2



$B2=\overline{G3}G2 + G3\overline{G2}$

$B2=G3\oplus G2$

K MAP FOR B1



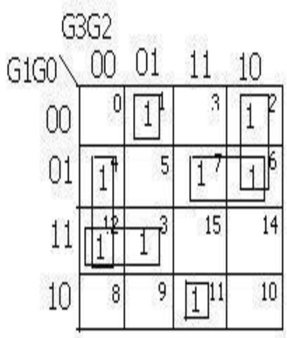
$B1=\overline{G1}G0\overline{G3}+\overline{G1}\,\overline{G0}G3+G1G0G3+G1\overline{G0}\,\overline{G3}$

$=\overline{G1}(G0\overline{G3}+\overline{G0}G3)+G1(G0G3+\overline{G0}\,\overline{G3})$

$=\overline{G1}(G0\oplus G3)+G1(\overline{G0\oplus G3})$

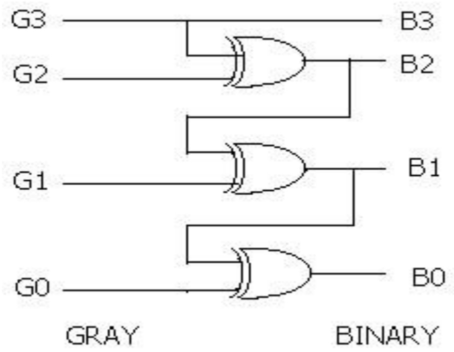
$B1=G3\oplus G2\oplus G1$

K MAP FOR B0



$B0=\overline{G1}\,\overline{G0}\,\overline{G3}G2+G1\overline{G0}G2G3+G0G3G2+G1G0\overline{G3}+\overline{G1}G0G3+\overline{G1}G3\overline{G2}$

$B0=G0\oplus G1\oplus G2\oplus G3$



RESULT

Thus, design and implementation of Binary to gray code converters and Vice Versa using logic gates using Multisim is done.

Experiment No:7 Date:

7. Design and implementation of Magnitude Comparator combinational circuits using simulation package

AIM:

1. To design and implementation of Magnitude Comparator using Multisim. 2. Hardware Implementation of the same with NI Analog Discovery 2.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	IC	IC 7485		1
2)	LED			4
3)	Switch			4
4)	DC Power Source			1
5)	NI Analog Discovery 2			1
6)	Wires			As Required
7)	Breadboard			1

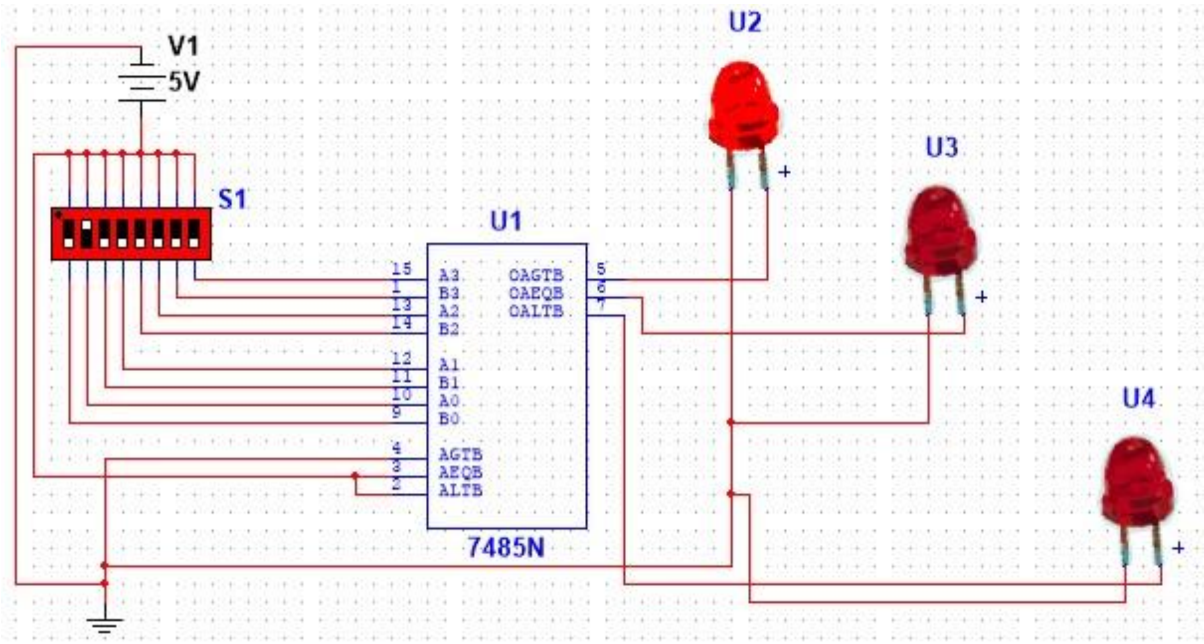
THEORY

Magnitude Comparator is a logical circuit, which compares two signals A and B and generates three logical outputs, whether A > B, A = B, or A < B. IC 7485 is a high speed 4-bit Magnitude comparator, which compares two 4-bit words. The A = B Input must be held high for proper compare operation.

PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as TTL, select Family as 74STD, Select 7485N from component.
5. Select the Master database, select group as Basic, select Family as Switch, Select DSWPK_8 from component.
6. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -DC Power.
7. Double the DC Power to change Voltage as 5V.
8. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -Ground.
9. Select the LED from master database, Group-Basic, Family-3D_Virtual, Component Led1_Red.
10. Place all the components, connect the components via wire to get the circuit diagram as below.
11. Run the simulation change the value of the switches to verify the truth table.

CIRCUIT DIAGRAM:



Truth Table:

A				B				Result
A3	A2	A1	A0	B3	B2	B1	B0	
0	0	0	1	0	0	0	0	A > B
0	0	0	1	0	0	0	1	A = B
0	0	0	0	0	0	0	1	A < B

RESULT

Thus, design and implementation of Magnitude Comparator using Multisim is done.

Experiment No:8 Date:

8. Design and implementation of Synchronous sequential circuits using Simulation Package

AIM:

1. To design and implementation of D Flip Flop using Multisim. 2. Hardware Implementation of the same with NI Analog Discovery 2.

APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	IC	IC 7474		1
2)	LED			4
3)	Switch			4
4)	DC Power Source			1
5)	Digital Clock			1

THEORY

A D-type flip-flop is a clocked flip-flop which has two stable states. A D-type flipflop operates with a delay in input by one clock cycle. Thus, by cascading many D-type flipflops delay circuits can be created, which are used in many applications such as in digital television systems.

A D-type flip-flop is also known as a D flip-flop or delay flip-flop. A D-type flip-flop consists of four inputs:

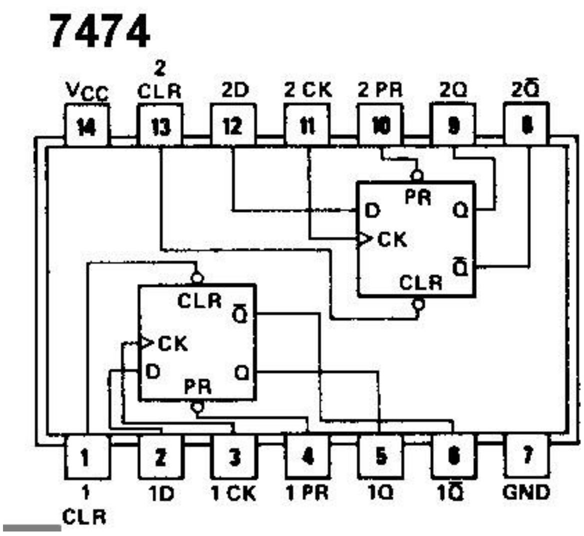
- Data input
- Clock input
- Set input
- Reset input

It also has two outputs, with one being logically inverse of other. The data input is either logic 0 or 1, meaning low or high voltage. The clock input helps in synchronizing the circuit to an external signal. The set input and reset input are mostly held low. A D-type flip-flop can have two possible values. When input D = 0, the flip-flop undergoes a reset, which means the output would be set to 0. When input D = 1, the flip-flop does a set, which makes the output 1. There are several applications in which a D-type flip-flop is used, such as in frequency dividers and data latches.

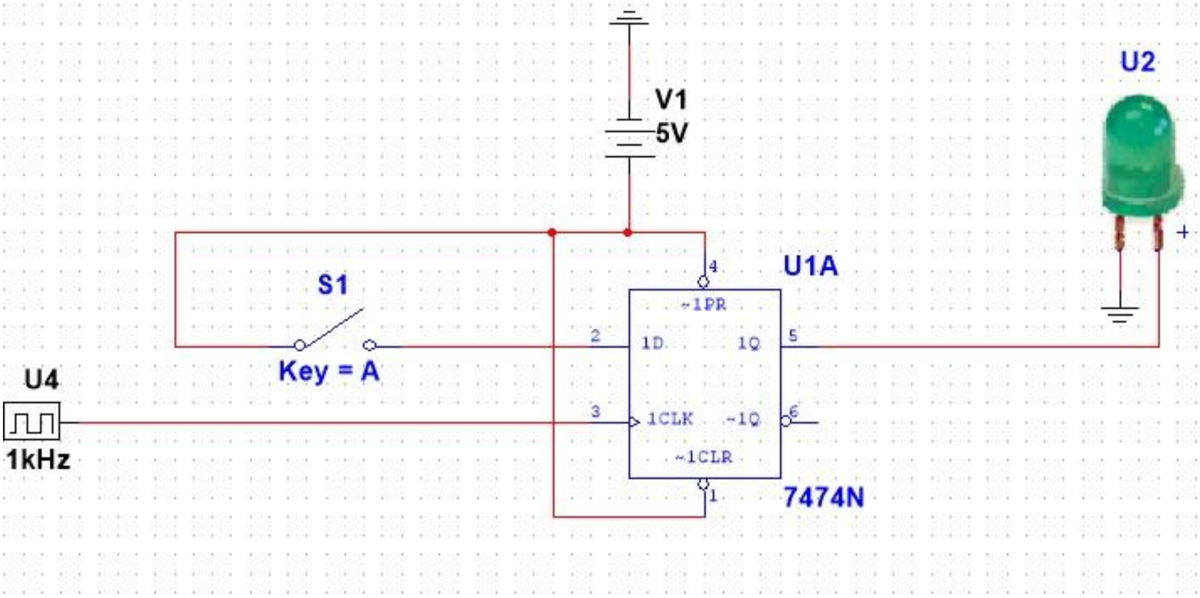
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as TTL, select Family as 74STD, Select 7474N from component.
5. Select the Master database, select group as Basic, select Family as Switch, Select DIPSW1 from component.
6. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -DC Power.
7. Double the DC Power to change Voltage as 5V.
8. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -Ground.
9. Select the LED from master database, Group-Basic, Family-3D_Virtual, Component Led1_Red.
10. Select the Digital Clock from master database, Group-Source, Family-Digital _Sources, Component -Digital _Clock.
11. Place all the components, connect the components via wire to get the circuit diagram as below.
12. Preset and clear pin of the IC 7474 is given +5V.
13. Run the simulation change the value of the switches to verify the truth table.

PIN DIAGRAM:



CIRCUIT DIAGRAM:



Truth Table:

TRUTH TABLE

INPUTS				OUTPUTS	
$\overline{PR}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	X	X
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q_0	$\overline{Q}_0$

RESULT

Thus, the implementation of D flip flop using Multisim is done.

Experiment No: 9 Date:

9. Implementation of SISO, SIPO, PISO and PIPO shift registers using Flip- flops

SISO

AIM:

1. To design and implementation of Shift Register using Multisim. 2. Hardware Implementation of the same with NI Analog Discovery 2.

APPARATUS REQUIRED

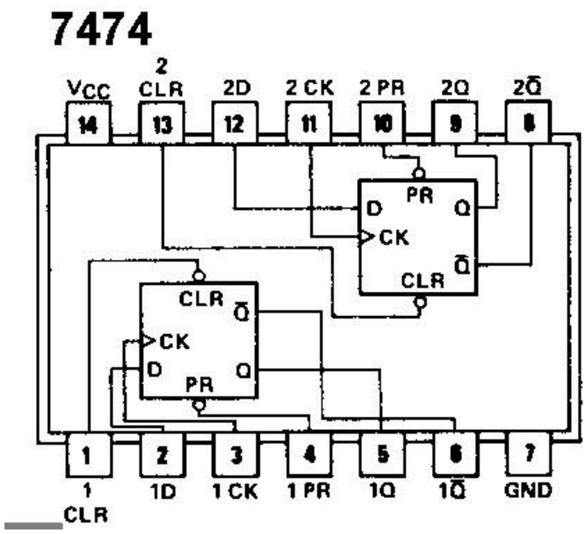
S.No	Apparatus	Type	Range	Quantity
1)	IC	IC 7474		4
2)	LED			4
3)	Switch			4
4)	DC Power Source			1
5)	Digital Clock			1

THEORY

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. All the flip-flops are driven by a common clock, and all are set or reset simultaneously.

The serial in/serial out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output also in serial form. The serial in/parallel out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output in parallel form. The parallel in/serial out shift register accepts data in parallel. It produces the stored information on its output also in serial form. The parallel in/parallel out shift register accepts data in parallel. It produces the stored information on its output in parallel form.

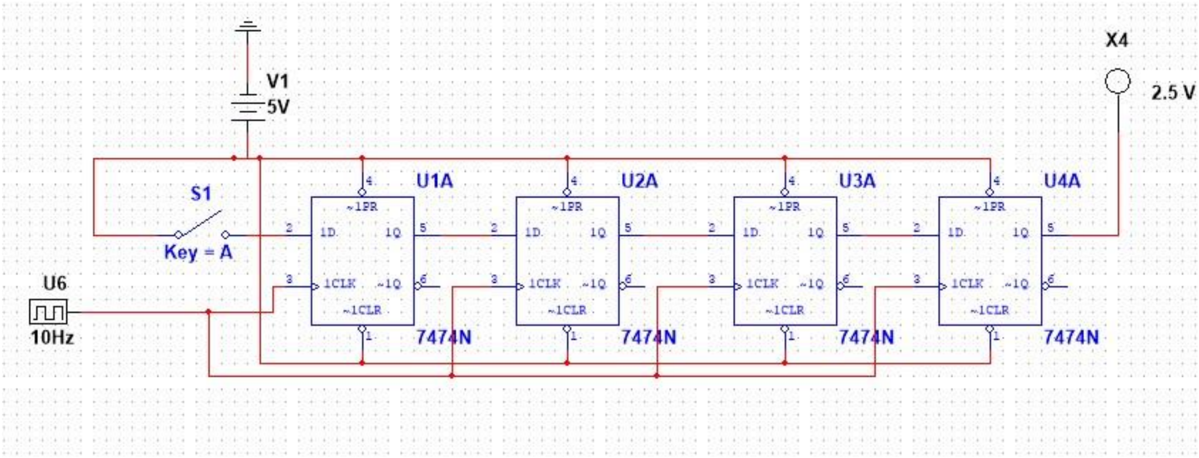
PIN DIAGRAM:



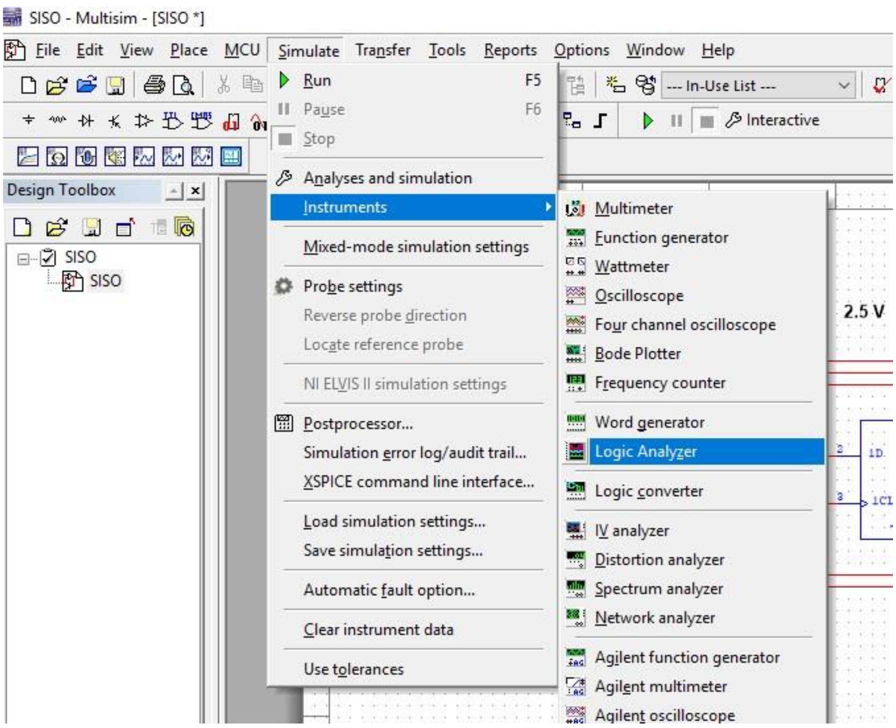
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl+W.
4. Select Master database, select group as TTL, select Family as 74STD, Select 7474N from component.
5. Place 4 such ICs in the workspace.
6. Select the Master database, select group as Basic, select Family as Switch, Select DIPSW1 from component.
7. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -DC Power.
8. Double the DC Power to change Voltage as 5V.
9. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component -Ground.
10. Select the Probe from master database, Group-Indicator, Family-Probe, Component Probe_ Dig_ green.
11. Select the Digital Clock from master database, Group-Source, Family-Digital _Sources, Component -Digital _Clock.
12. Double click clock source and change to 50Hz.
13. Preset and clear pin of the IC 7474 is given +5V.
14. Place all the components, connect the components via wire to get the circuit diagram as below.

CIRCUIT DIAGRAM:

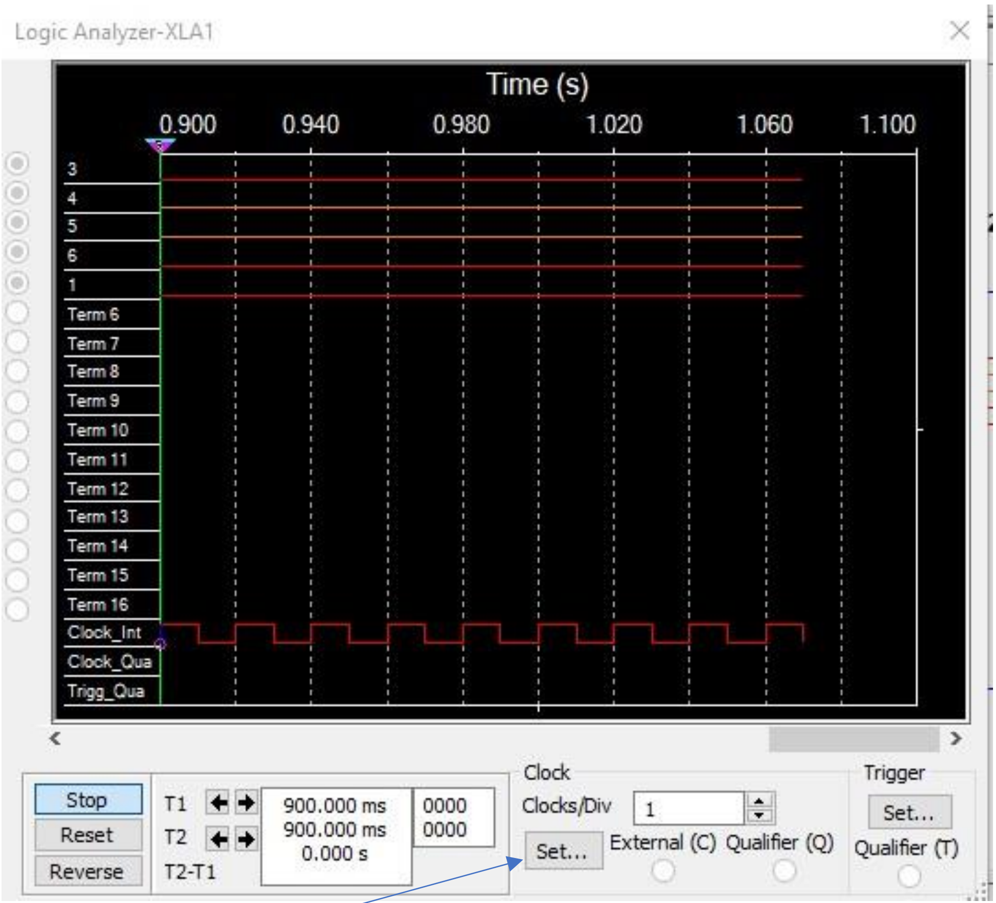


15. Run the simulation change the value of the switch you can see the probe lights glows one by one.
16. Use Logic analyzer instrument in Multisim to view all the signal.

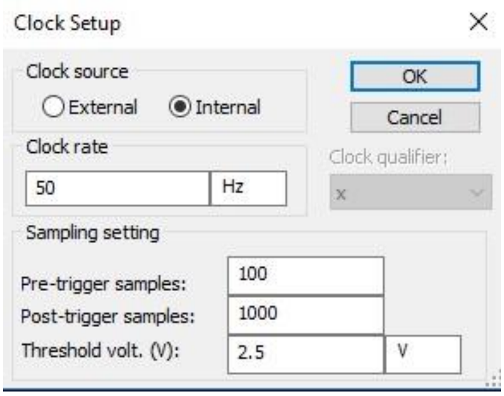


16.Connect output from each of the flip flop to Logic analyzer Pin 1 to 4.

17. Double click the instrument to open.



18.Click the set button

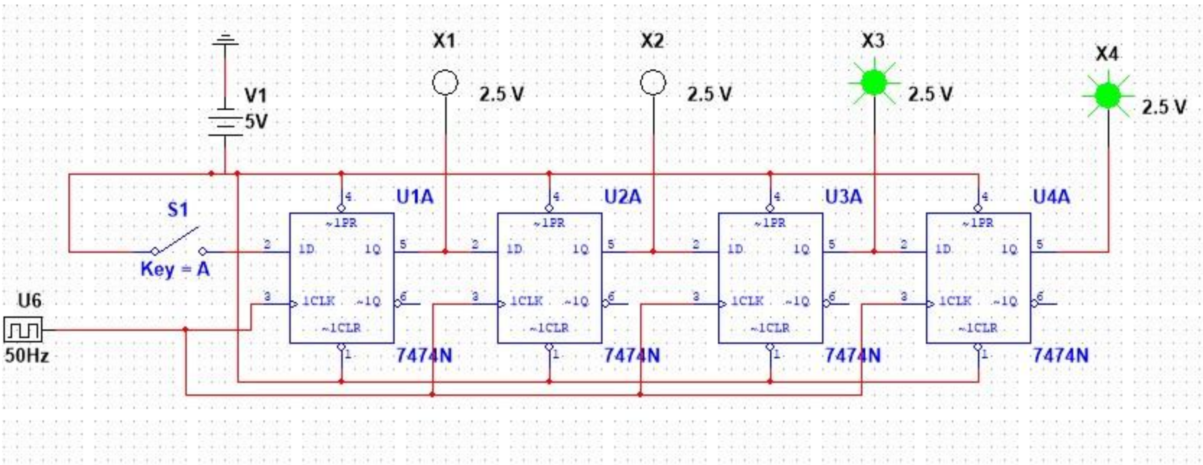


19. Set the clock rate to 50hz and clock source as Internal. Unchange all the other parameters.
20.Change the switch position in the Multisim while running the simulation.
21.stop the simulation and open the Grapher view to analyze your signals.

OUTPUT:

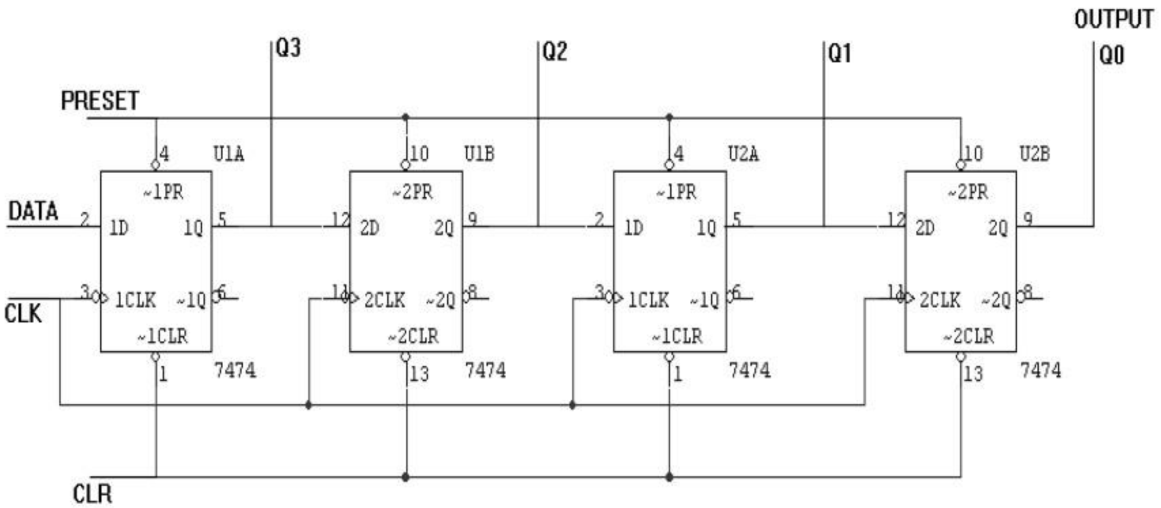
SIPO

MULTISIM



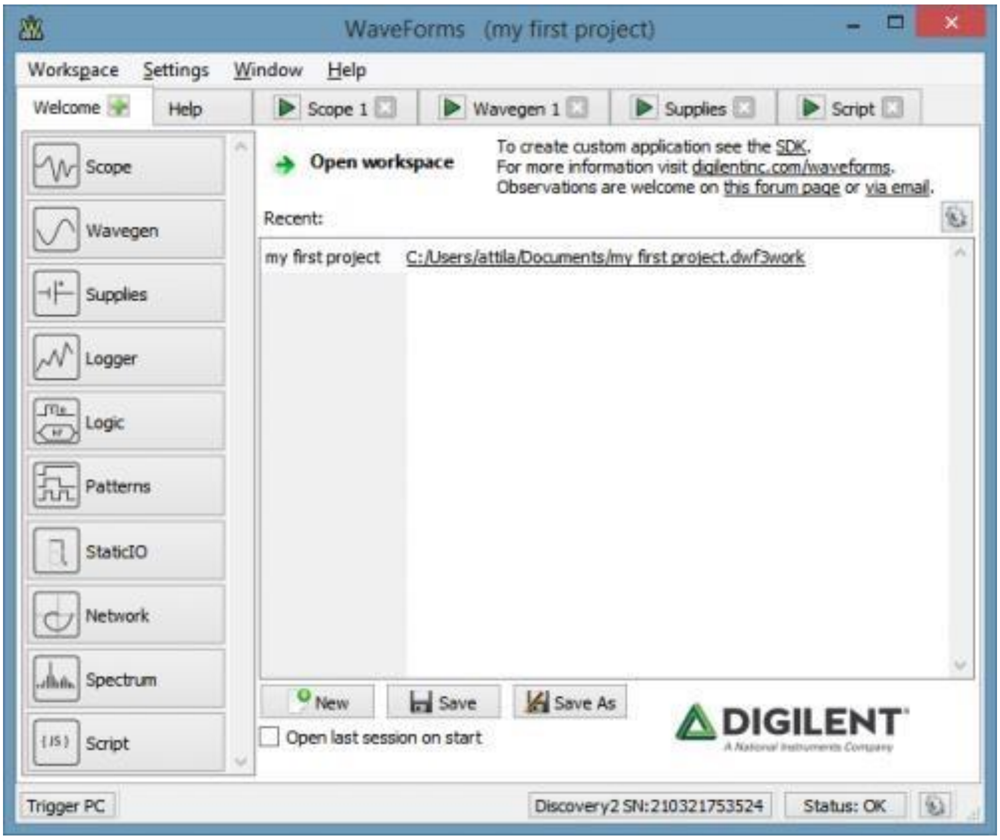
ANALOG DISCOVERY 2

CIRCUIT DIAGRAM

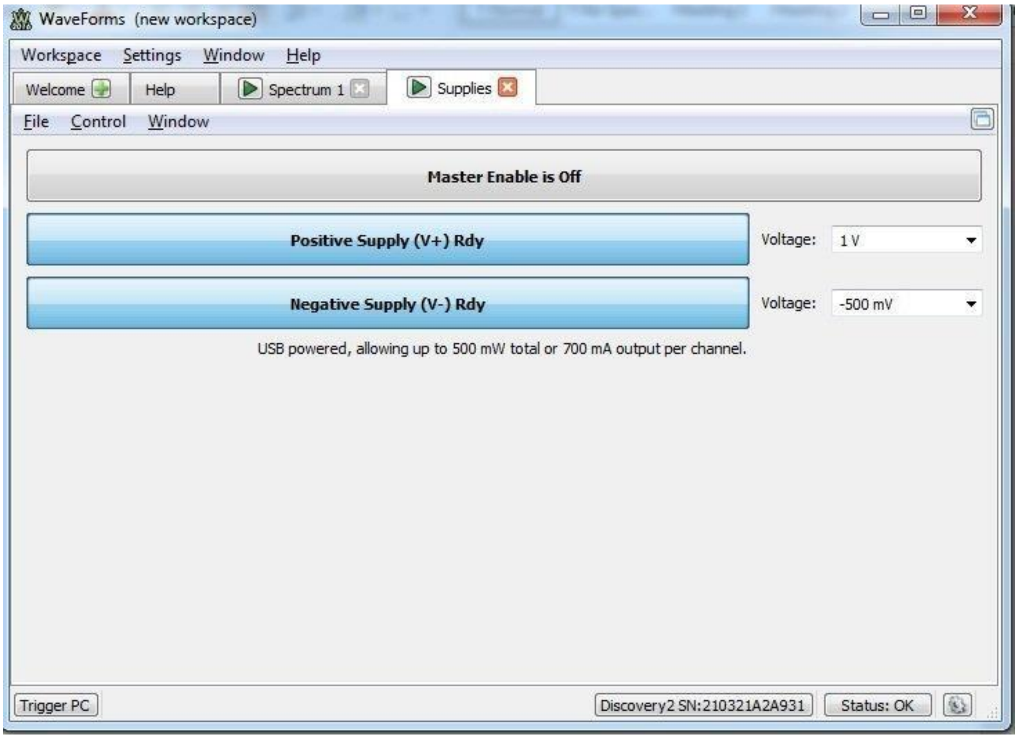


PROCEDURE:

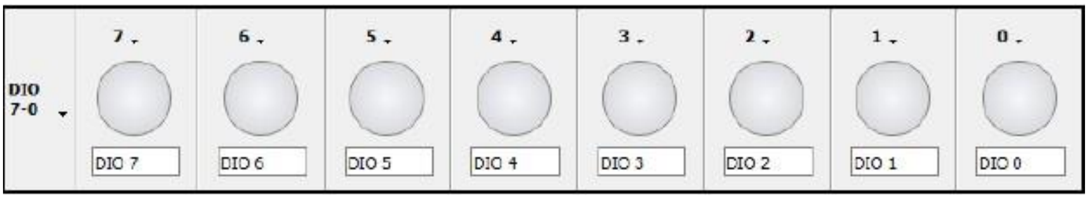
1. Fix the IC 7474 in the breadboard.
2. Red wire of AD2 belongs to power. Take a wire connect to red wire, take it to all the IC pin 14.
3. Black wire of AD2 belongs to ground. Take a wire connect to Black wire take it to all the IC pin 7.
4. Give the connections in the bread board as per the circuit diagram above.
5. Use Pin 0 of AD2 as Input. Connect to IC7474 Pin 2.
6. Use Pin 1 of AD2 as Clock. Connect to IC7474 Pin 3.
7. Take the output of each flip flop connect it to AD2 DIO Pin 2-Pin 5.
8. Search the application in PC for Waveform 2015.



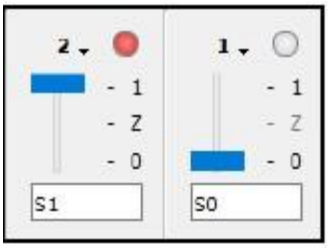
9. In the above window click the Supplies Instrument.



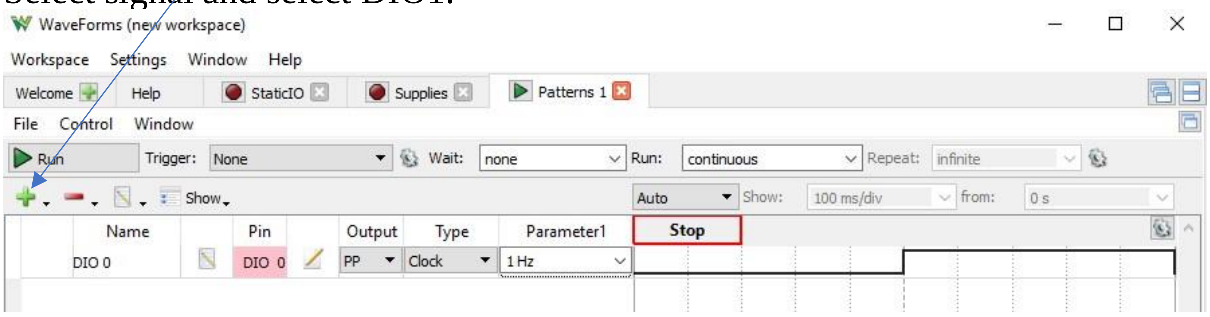
10. Use only positive supply. Change the voltage as 5.
11. Click Master Enable button to enable the Instrument.
12. In the Welcome tab, select Static IO Instrument to open.



13. Configure Digital I/O signal into a switch as Switch to Push/Pull (1/0) as seen in Figure below for DIO 0



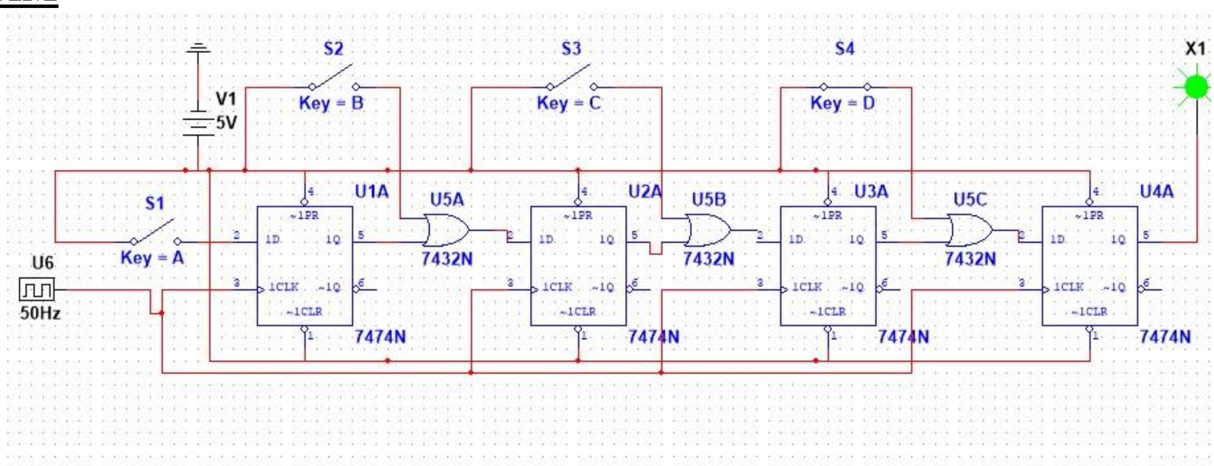
14. Click open the pattern instrument.
15. Click this green plus to add the signal.
16. Select signal and select DIO1.



17. Click the black arrow under type column and select clock.
18. Double click the parameter column and change the value to 1hz, otherwise output will respond very fast to view.
19. Run Pattern, Static IO and Power Supplies Instrument to see the output.

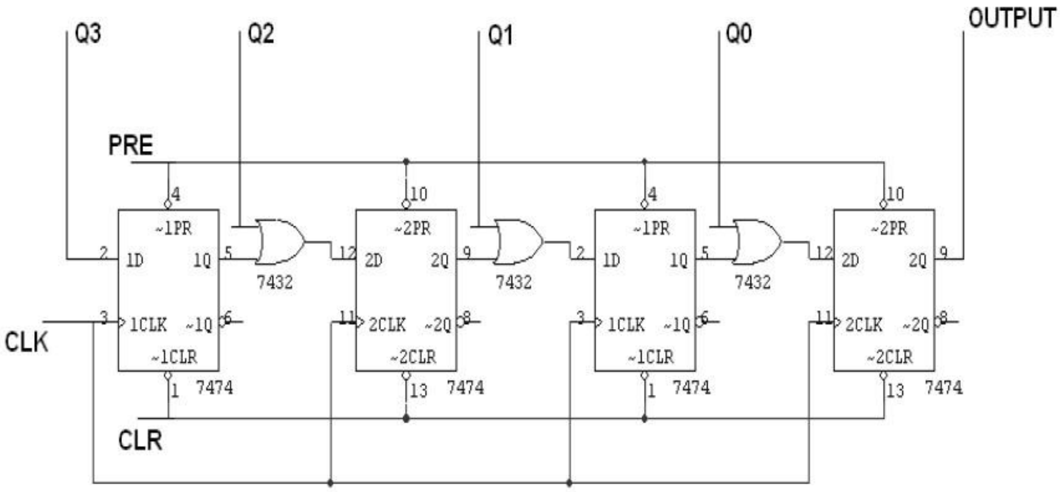
PISO

MULTISIM



ANALOG DISCOVERY 2

CIRCUIT DIAGRAM

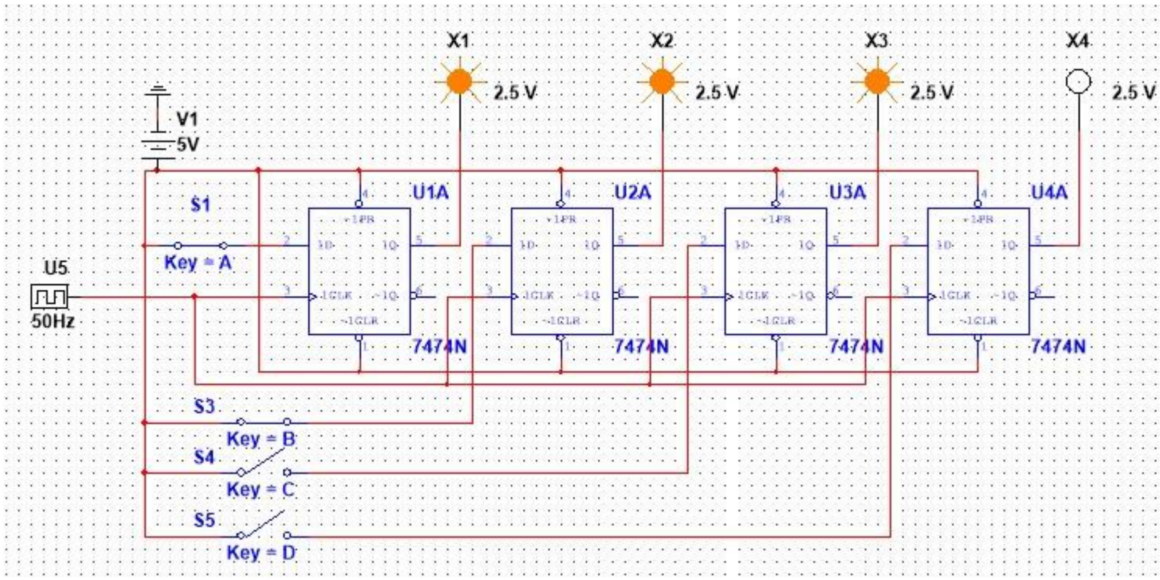


TRUTH TABLE

CLK	D_A	D_B	D_C	D_D	Q_{out}
1	1	0	1	1	1
2	0	1	0	1	1
3	0	0	1	0	0
4	0	0	0	1	1

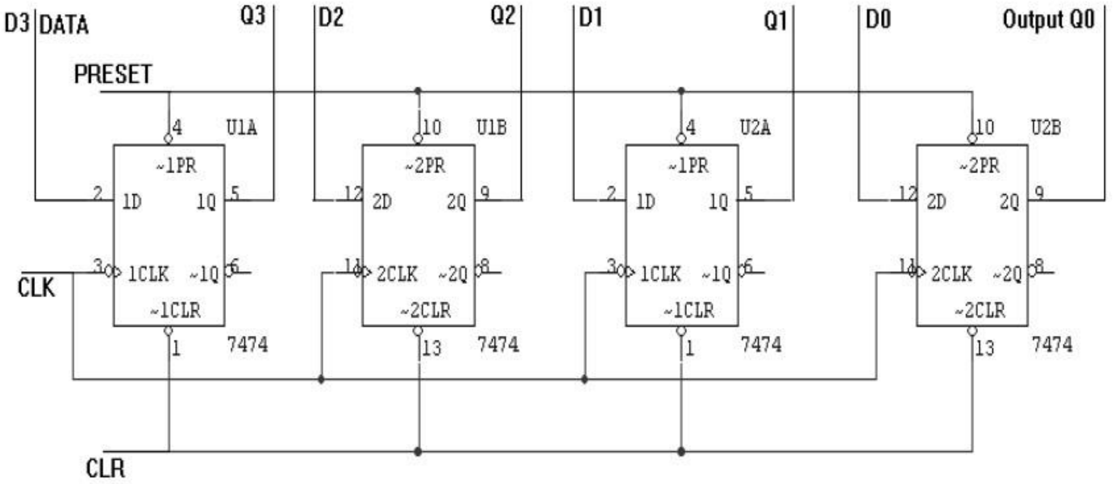
PIPO

MULTISIM



ANALOG DISCOVERY 2

CIRCUIT DIAGRAM



TRUTH TABLE

CLK	DATA INPUT				OUTPUT			
	D_A	D_B	D_C	D_D	Q_A	Q_B	Q_C	Q_D
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

RESULT

Thus, the design and implementation of shift registers using Multisim is done.

Experiment No:10 Date:

Design and Implement an A/D converter

AIM:

To Design and Implement an A/D converter using Multisim

.

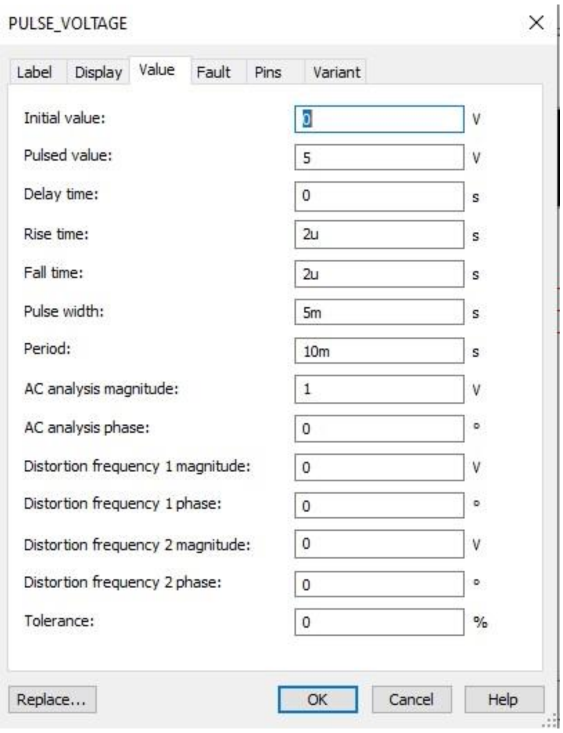
APPARATUS REQUIRED

S.No	Apparatus	Type	Range	Quantity
1)	ADC			1
2)	DC Source		10v	1
3)	Potentiometer		1k	1
4)	Pulse Voltage Source			1
5)	Probe			8
6)	Hex display			2
7)	Voltmeter			1

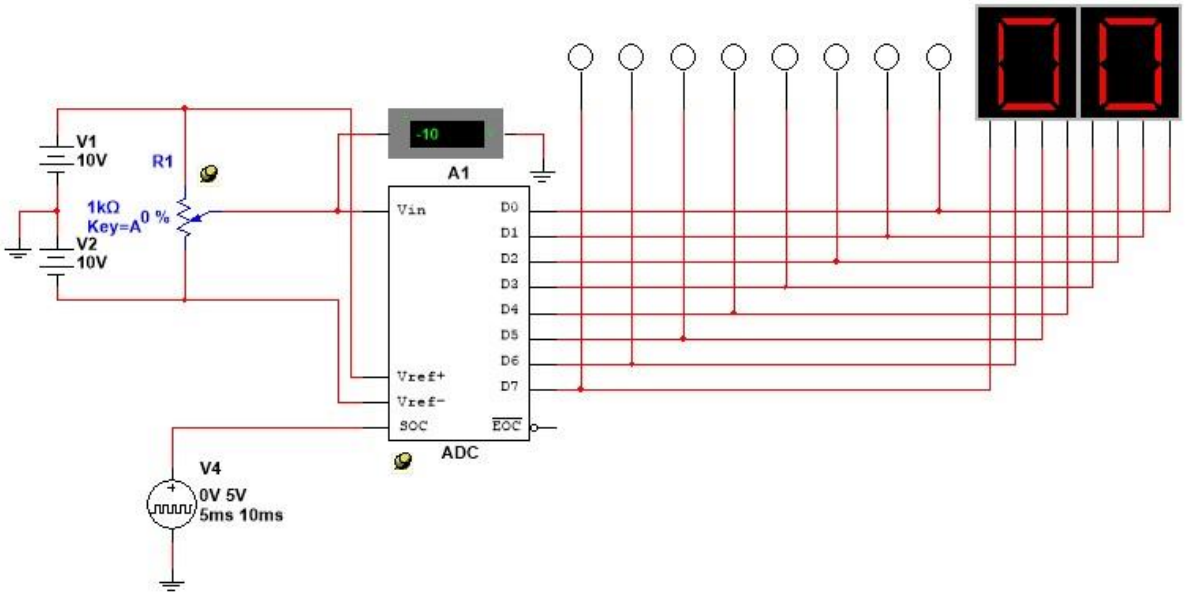
PROCEDURE

1. Open Multisim.
2. Select file → New → Blank and recent → select Blank and click create button on the bottom right corner of the window opened.
3. Select the components by pressing ctrl + W.
4. Select the Probe from master database, Group-Indicators, Family-Probe, Component Probe_ BLUE.
5. Select the DC Power Source from master database, Group-Source, Family-Power Source, Component - Vcc.
6. Select the ADC from master database, Group-Mixed, Family-ADC_DAC, Component - ADC.
7. Select the Potentiometer from master database, Group-Basic, Family-Potentiometer, Component -1k.
8. Select the Pulse Voltage Source from master database, Group-Source, Family-signal_ voltage_ sources, Component - pulse_ voltage.

9.	Select the Hex display from master database, Group-Indicators, Family-Hex_ Displays, Component -DCD_ Hex.
10.	Select the Voltmeter from master database, Group-Indicators, Family-Voltmeter, Component - VOLTMETER_H.
11.	Set Pulse voltage source as in the figure below.



12.	Place all the components. Give the connection as per the figure below.



13.	Note down the value from the voltmeter for analog value, digital value from probes and hex display for the corresponding change in potentiometer value.
14.	Complete the tabulation below. One example is given in the tabulation below.

TABULATION:

Potentiometer (%)	**Analog(v)** (Output of Voltmeter)	**Digital** (Value of probe) (D0 D1 D2 D3 D4 D5 D6 D7)	**Hex display**
20%=200Ω	1	00110001	8C

Result:

Thus the design and implementation of ADC is done with Multisim.

Experiment no: 11 **Date:**

HDL PROGRAM FOR COMBINATIONAL CIRCUITS

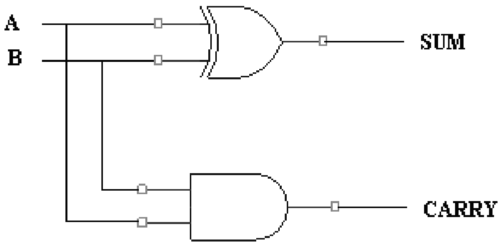
AIM:
To develop the source code for adders and subtractors by using VERILOG and obtain the simulation & synthesis.

ALGORITM:
Step1: Define the specifications and initialize the design.
Step2: Declare the name of the entity and architecture by using VHDL source code.
Step3: Write the source code in VERILOG.
Step4: Check the syntax and debug the errors if found, obtain the synthesis report.
Step5: Verify the output by simulating the source code.
Step6: Write all possible combinations of input using the test bench.
Step7: Obtain the place and route report.

<u>BASIC ADDERS & SUBTRACTORS:</u>

HALF ADDER:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VHDL SOURCE CODE:

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module ha_dataflow(a, b, s, ca);
   input a;
   input b;
   output s;
   output ca;
            assign#2 s=a^b;
            assign#2 ca=a&b;
endmodule
```

Behavioral Modeling:

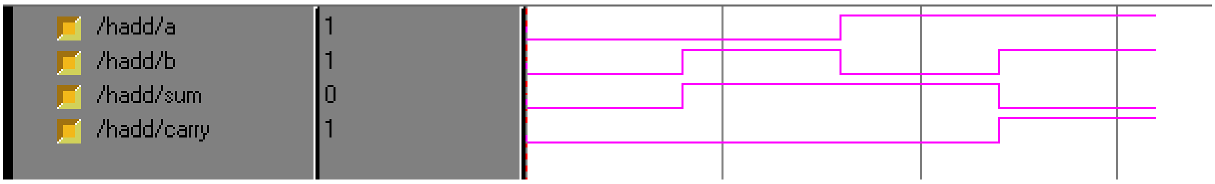
```
module ha_behv(a, b, s, ca);
   input a;
   input b;
```

```
    output s;
    output ca;
                reg s,ca;
                always @ (a or b) begin
                s=a^b;
                ca=a&b;
                end
endmodule
```

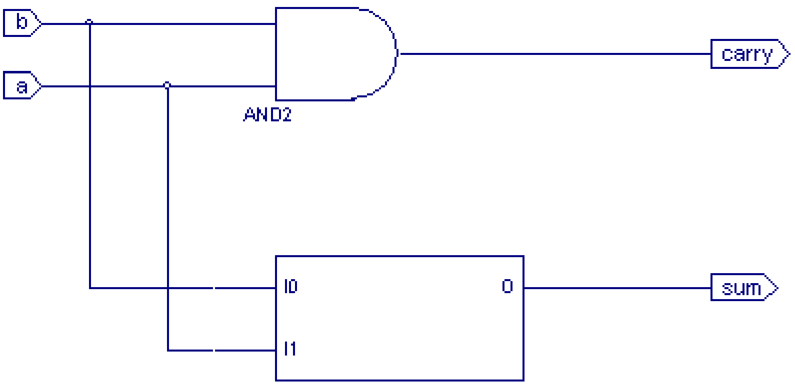
Structural Modeling:

```
module ha_struct(a, b, s, ca);
    input a;
    input b;
    output s;
    output ca;
                xor
                x1(s,a,b);
                and
                a1(ca,a,b);
endmodule
```

Simulation output:

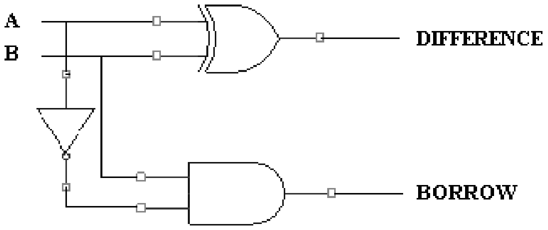


Synthesis RTL Schematic:



HALF SUBTRACTOR:

LOGIC DIAGRAM:



TRUTH TABLE

A	B	DIFFERENCE	BORROW
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module hs_dataflow(a, b, dif, bor);
   input a;
   input b;
   output dif;
   output bor;
            wire abar;
            assign#3 abar=~a;
            assign#3 dif=a^b;
            assign#3 bor=b&abar;
endmodule
```

Behavioral Modeling:

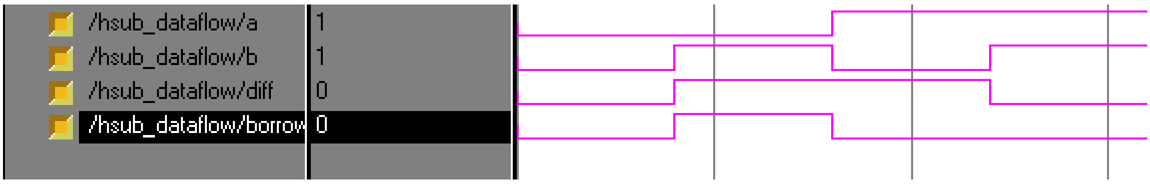
```
module hs_behv(a, b, dif, bor);
   input a;
   input b;
```

```
    output dif;
    output bor;
            reg dif,bor;
            reg abar;
            always@(a or b) begin
            abar=~a;
            dif=a^b;
            bor=b&abar;
            end
endmodule
```

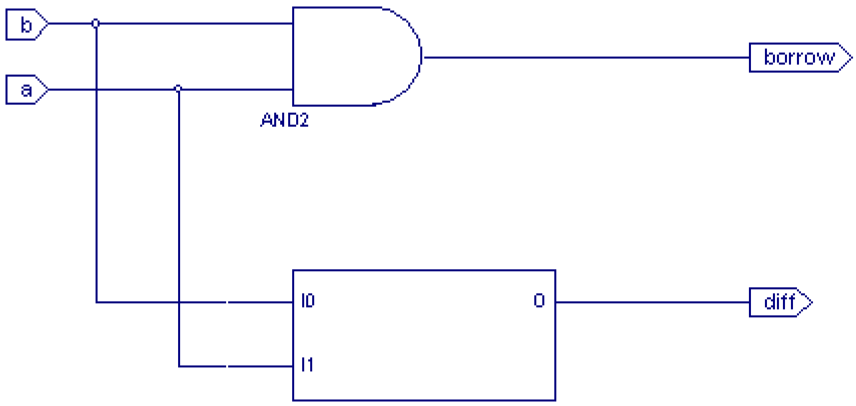
Structural Modeling:

```
module hs_struct(a, b, dif, bor);
    input a;
    input b;
    output dif;
    output bor;
            wire abar;
            xor
            x1(dif,a,b);
            not
            n1(abar,a);
            and
            a1(bor,abar,b);
endmodule
```

Simulation output:

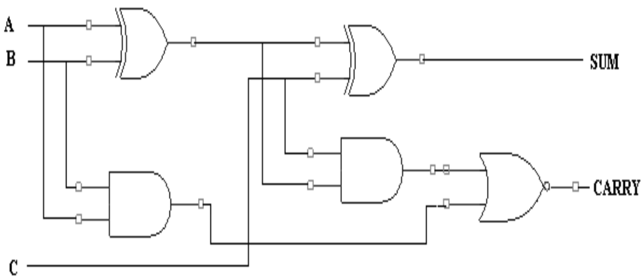


Synthesis RTL Schematic:



FULL ADDER:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module fulladddataflow(a, b, cin, sum, carry);
    input a;
    input b;
    input cin;
    output sum;
    output carry;
 assign sum=a^b^cin;
 assign carry=(a & b) | (b & cin) | (cin & a);
 endmodule
```

Behavioral Modeling:

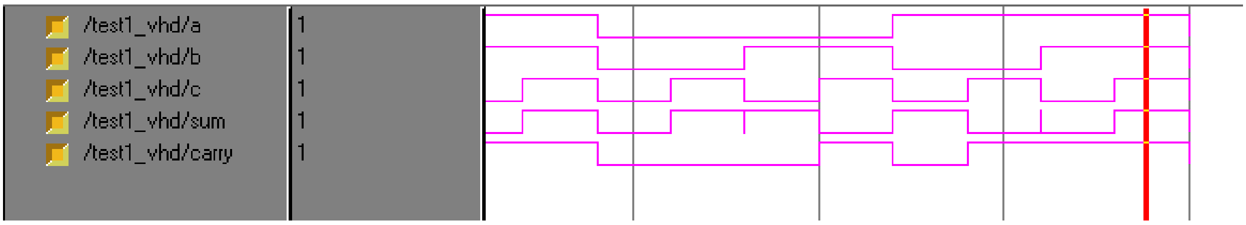
```
module fuladbehavioral(a, b, c, sum, carry);
    input a;
    input b;
    input c;
```

```
    output sum;
    output carry;
            reg sum,carry;
            reg t1,t2,t3;
            always @ (a or b or c) begin
            sum = (a^b)^c;
            t1=a & b;
            t2=b & c;
            t3=a & c;
            carry=(t1 | t2) | t3;
            end
endmodule
```

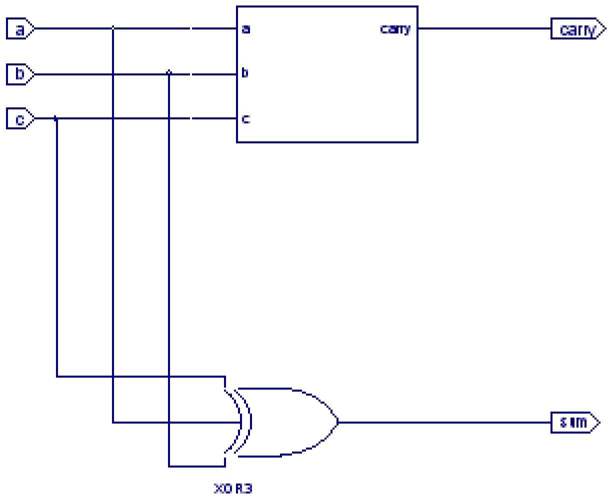
Structural Modeling:

```
module fa_struct(a, b, c, sum, carry);
    input a;
    input b;
    input c;
    output sum;
    output carry;
            wire p,q,r,s;
            xor
            x1(p,a,b),
            x2(sum,p,c);
            and
            a1(q,a,b),
            a2(r,b,c),
            a3(s,a,c);
            or
            o1(carry,q,r,s);
endmodule
```

Simulation output:

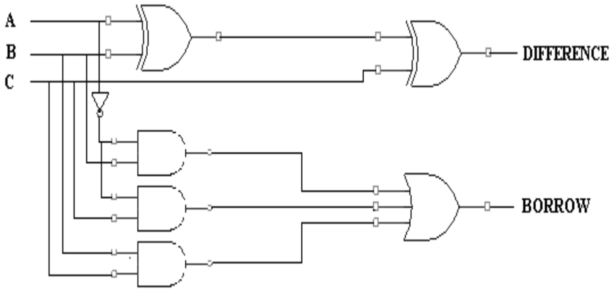


Synthesis RTL Schematic:



FULL SUBTRACTOR:

LOGIC DIAGRAM:



TRUTH TABLE:

A	B	C	DIFFERENCE	BORROW
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

VERILOG SOURCE CODE:

Dataflow Modeling:

```
module fulsubdataflow(a, b, cin, diff, borrow);
    input a;
```

```
    input b;
    input cin;
    output diff;
    output borrow;
            wire abar;
            assign abar= ~ a;
            assign diff=a^b^cin;
            assign borrow=(abar & b) | (b & cin) |(cin & abar);

endmodule
```

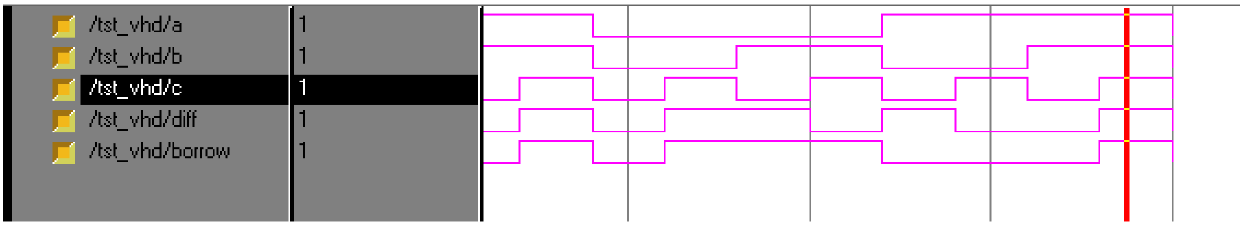
Behavioral Modeling:

```
module fulsubbehavioral(a, b, cin, diff, borrow);
    input a;
    input b;
    input cin;
    output diff;
    output borrow;
            reg t1,t2,t3;
            reg diff,borrow;
            reg abar;
            always @ (a or b or cin) begin
            abar= ~ a;
            diff = (a^b)^cin;
            t1=abar & b;
            t2=b & cin;
            t3=cin & abar;
            borrow=(t1 | t2) | t3;
            end
           endmodule
```

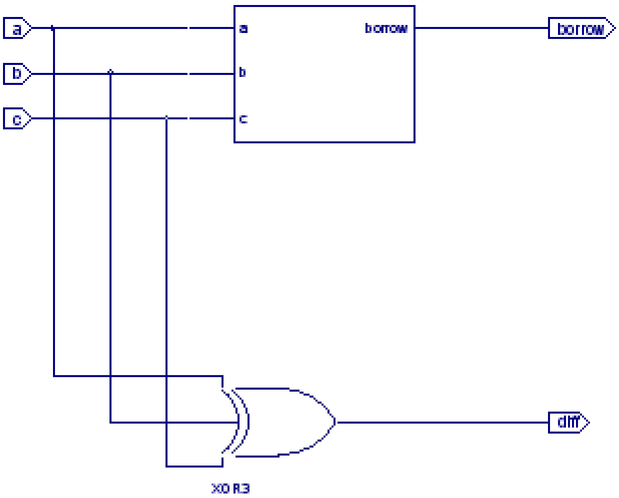
Structural Modeling:

```
module fs_struct(a, b, c, diff, borrow);
    input a;
    input b;
    input c;
    output diff;
    output borrow;
            wire abar,p,q,r,s;
            not
            n1(abar,a);
            xor
            x1(p,a,b),
            x2(diff,p,c);
            and
            a1(q,abar,b),
            a2(r,abar,c),
            a3(s,a,c);
            or
            o1(borrow,q,r,s);
endmodule
```

Simulation output:



Synthesis RTL Schematic:



<u>RESULT:</u>

Thus the OUTPUT of HDL program for Combinational circuits is done and verified.

Experiment no: 12 **Date:**

BINARY COUNTER

AIM:

To write a verilog HDL program for binary counter and verify its output.

SOFTWARE REQUIRED:

Xilinx ISE 10.1

ALGORITHM:

Step1: Define the specifications and initialize the design.

Step2: Write the source code in VERILOG.

Step3: Check the syntax and perform synthesis .

Step4: Write different combinations of input using the test bench.

Step5:Verify the output by simulating the source code.

VERILOG SOURCE CODE:

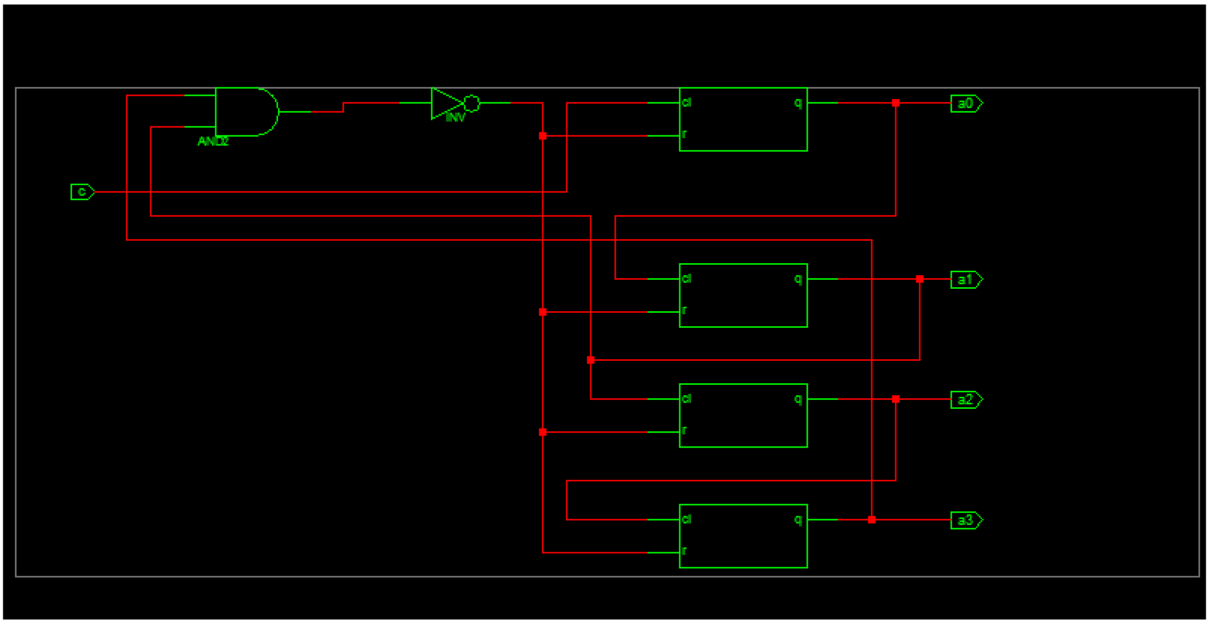
```
module count1(count ,clk,rst);

output [3:0] count;

input clk;

input rst;

reg [3:0]count;

always @ (posedge clk)

if(rst)

count=4'b0000;

else

begin

if(count==4'b1111)
```

```
count=4'b0000;
else
count=count+1;
end
endmodule
```

TESTBENCH:

```
module test;
	// Inputs
	reg clk;
	reg rst;
	// Outputs
	wire [3:0] count;
	// Instantiate the Unit Under Test (UUT)
	count1 uut (
		.clk(clk),
		.rst(rst),
		.count(count)
	);
	initial begin
		// Initialize Inputs
		clk = 0;
		rst = 0;
		// Wait 100 ns for global reset to finish
		#100;
		// Add stimulus here
	end
endmodule
```

RTL SCHEMATIC:



SYNTHESIS REPORT:

*   Final Report   *

==

Final Results

RTL Top Level Output File Name   : count1.ngr

Top Level Output File Name   : count1

Output Format   : NGC

Optimization Goal   : Speed

Keep Hierarchy   : NO

Design Statistics

IOs   : 6

Cell Usage :

BELS   : 6

INV   : 1

\#   LUT2   : 1

\#   LUT2_L   : 1

\#   LUT3   : 1

\#   LUT4   : 2

\# FlipFlops/Latches   : 4

\#   FDR   : 4

\# Clock Buffers   : 1

\#   BUFGP   : 1

\# IO Buffers   : 5

\#   IBUF   : 1

\#   OBUF   : 4

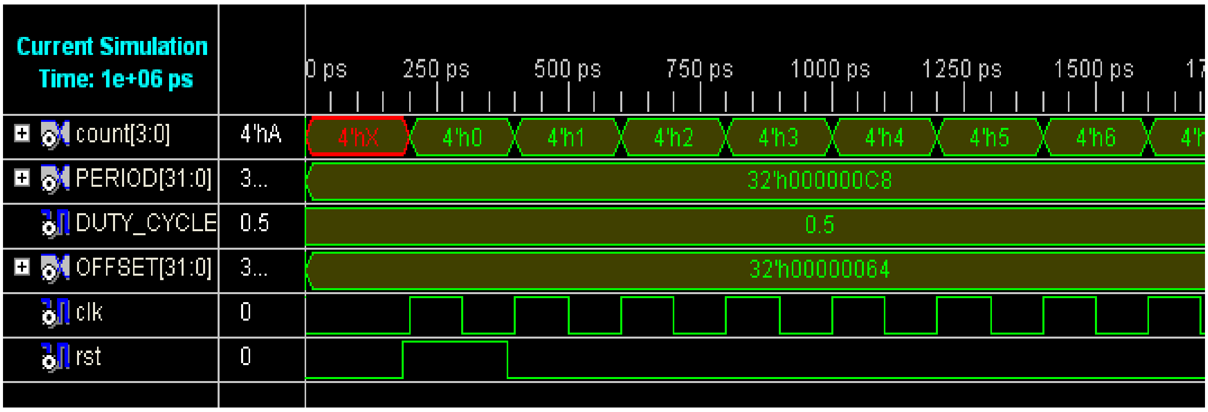
===

Device utilization summary:

\---------------------------

Selected Device :	3s100evq100-4
Number of Slices:	3 out of 960 0%
Number of Slice Flip Flops:	4 out of 1920 0%
Number of 4 input LUTs:	6 out of 1920 0%
Number of IOs:	6
Number of bonded IOBs:	6 out of 66 9%
Number of GCLKs:	1 out of 24 4%

SIMULATION OUTPUT:



RESULT:

Thus a verilog HDL program was written for binary counter and its output was verified.

EXP NO: 13 **DATE:**

MOD-10 COUNTER

AIM:

To write a verilog HDL program for mod-10 counter and verify its output.

SOFTWARE REQUIRED:

Xilinx ISE 10.1

ALGORITHM:

Step1: Define the specifications and initialize the design.

Step2: Write the source code in VERILOG.

Step3: Check the syntax and perform synthesis .

Step4: Write different combinations of input using the test bench.

Step5:Verify the output by simulating the source code

VERILOG SOURCE CODE:

```
module modten(count ,clk,rst);

output [3:0] count;

input clk;

input rst;

reg [3:0]count;

always @ (posedge clk)

if(rst)

count=4'b0000;

else

begin

if(count==4'b1010)

count=4'b0000;
```

```
else
count=count+1;
end
endmodule
```

TEST BENCH(VERILOG):

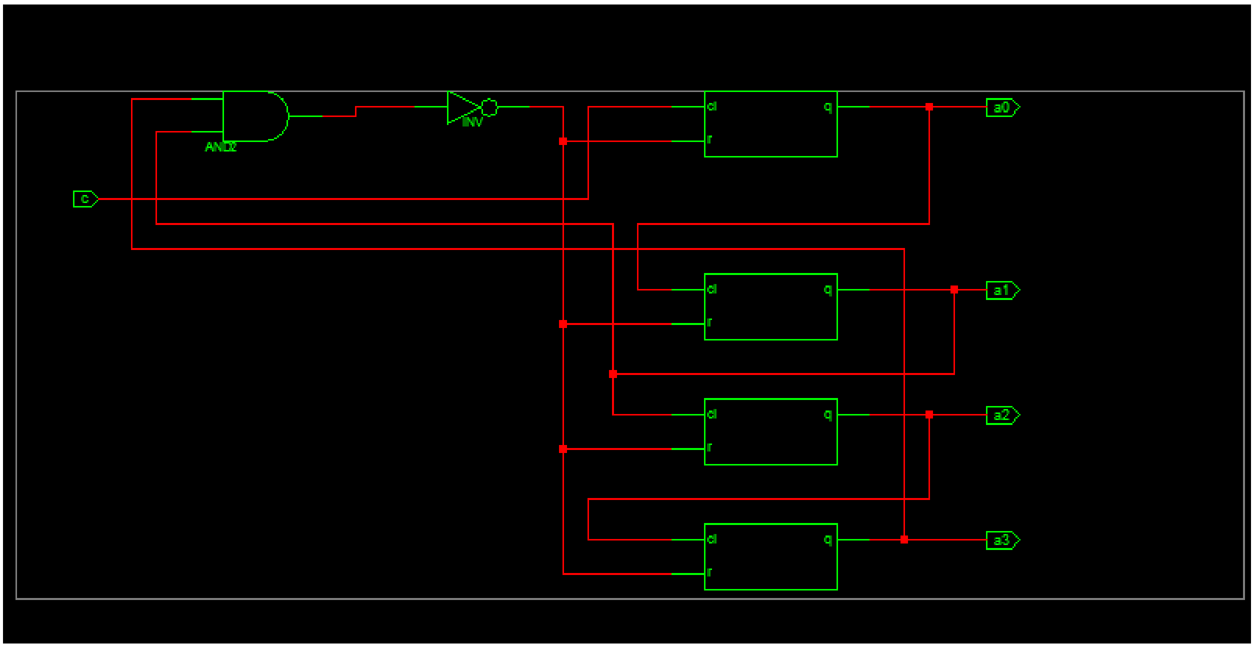
```
module MOD10_TB_v;
        // Inputs
        reg clk;
        reg rst;
        // Outputs
        wire [3:0] count;
        // Instantiate the Unit Under Test (UUT)
        MOD10  uut (
                .clk(clk),
                .rst(rst),
                .count(count)
        );
        initial begin
                // Initialize Inputs
                clk = 0;
                rst = 0;
                // Wait 100 ns for global reset to finish
                #100;
```

// Add stimulus here

end

endmodule

RTL SCHEMATIC:



SYNTHESIS REPORT:

===

*   Final Report   *

===

Final Results

RTL Top Level Output File Name   : modten.ngr

Top Level Output File Name   : modten

Output Format   : NGC

Optimization Goal   : Speed

Keep Hierarchy   : NO

Design Statistics

IOs   : 6

Cell Usage :

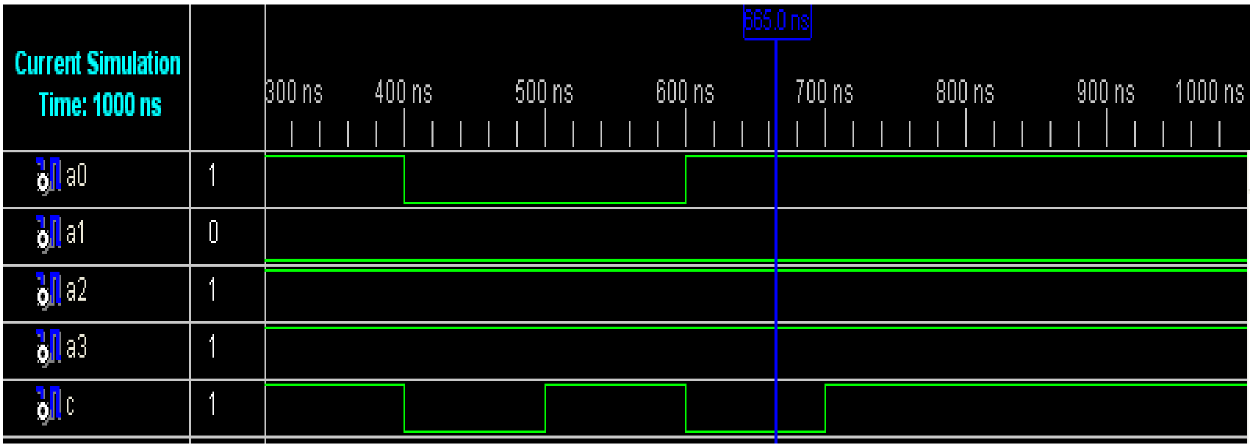
# BELS		: 6
#	INV	: 1
#	LUT2	: 1
#	LUT2_L	: 1
#	LUT3	: 1
#	LUT4	: 2
# FlipFlops/Latches		: 4
#	FDR	: 4
# Clock Buffers		: 1
#	BUFGP	: 1
# IO Buffers		: 5
#	IBUF	: 1
#	OBUF	: 4

===

Device utilization summary:

Selected Device :	3s100evq100-4			
Number of Slices:	3	out of	960	0%
Number of Slice Flip Flops:	4	out of	1920	0%
Number of 4 input LUTs:	6	out of	1920	0%
Number of IOs:	6			
Number of bonded IOBs:	6	out of	66	9%
Number of GCLKs:	1	out of	24	4%

SIMULATION OUTPUT:



RESULT:

Thus a verilog HDL program was written for mod-10 counter and its output was verified.