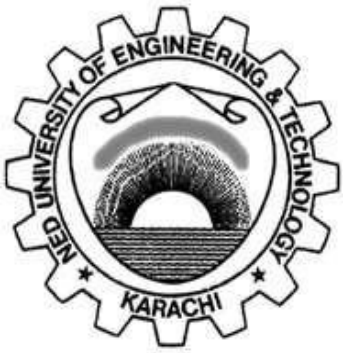


Department of Electronic Engineering
NED University of Engineering & Technology

LABORATORY WORKBOOK

DIGITAL LOGIC DESIGN(TC-201)

Instructor Name: Muneeb Shaikh

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Department: Electronic engineering

LABORATORYWORKBOOK

DIGITAL LOGIC DESIGN

(TC-201)

Prepared By:
Dr. Rizwan Aslam (Assistant Professor)

1st Revision By:
Engr.Shafaq Mustafa(Lecturer)
2nd Revision By:
Dr. Muhmmad Fahim Ul Haque (Assistant Professor)
Engr. Saba Ahmed (Assistant Professor)

Reviewed By:
Dr. Tahir Malik (Assistant Professor)

.....

Approved By:
The Board of Studies of Department of Electronic Engineering

INTRODUCTION

Digital Logic Design Practical Workbook covers those practical that are very knowledgeable and quite beneficial in grasping the core objective of the subject. These practical solidify the theoretical and practical concepts that are very essential for the engineering students.

This work book comprise of practical covering the topics of Digital Logic Design that are arranged on modern concepts. Above all this workbook contains a relevant theory about the Lab session.

CONTENTS

LABSESSION01

To study basic logic gates and their functions

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION01

OBJECTIVE:

To study basic logic gates and their functions.

THEORY:

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions *low* (0) or *high* (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

AND GATE:

The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true."Otherwise, the output is "false."

AND gate

Input1	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:

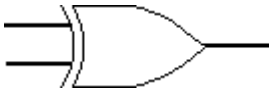
The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false."

OR gate

Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR GATE:

The *XOR* (*exclusive- OR*) *gate* acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

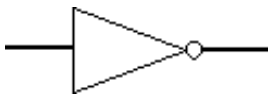


XOR gate

Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	0

NOT GATE:

A logical *inverter*, sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.



Inverter or NOT gate

Input	Output
1	0
0	1

NAND GATE:

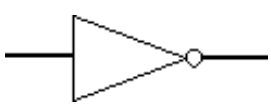
The *NAND gate* operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

NAND gate

Input1	Input2	Output
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

The *NOR gate* is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

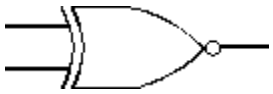


NOR gate

Input1	Input2	Output
0	0	1
0	1	0
1	0	0
1	1	0

XNOR GATE:

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same and "false" if the inputs are different.



XNOR gate

Input1	Input2	Output
0	0	1
0	1	0
1	0	0
1	1	1

Using combinations of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital integrated circuits (ICs). As IC technology advances, the required physical volume for each individual logic

gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.

Common Gate ICs:

Part number	Description
7400	quad 2-input NAND gate
7402	quad 2-input NOR gate
7408	quad 2-input AND gate
7410	triple 3-input NAND gate
7432	quad 2-input OR gate
7486	quad 2-input XOR gate

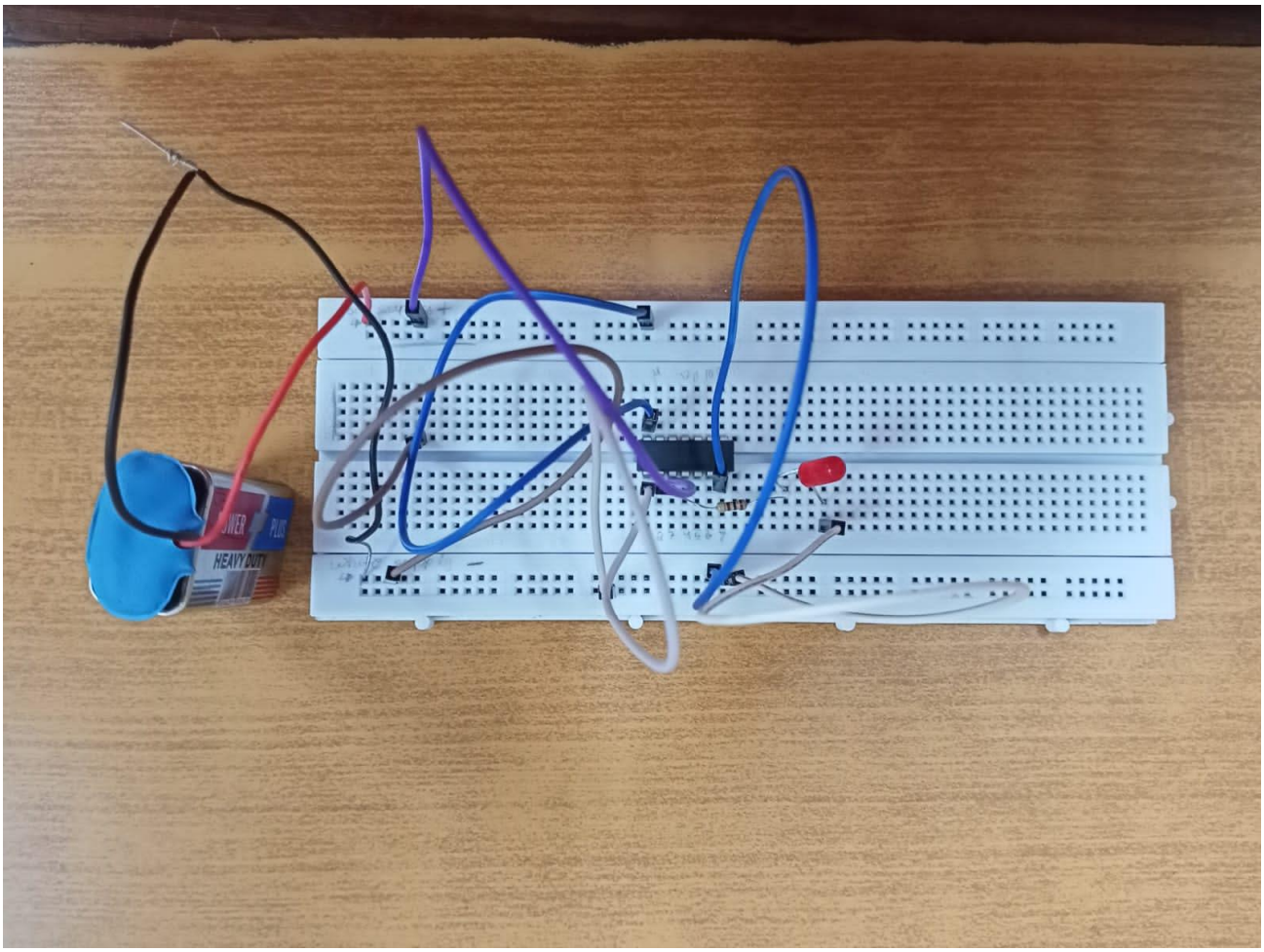
LABORATORY TASK:

1) Power up the 2-input AND , OR and NOT TTL ICs on a breadboard.
2) Apply inputs using push-to-on/off switches and observe the output via LEDs.
3) Fill the Table provided in the result area.

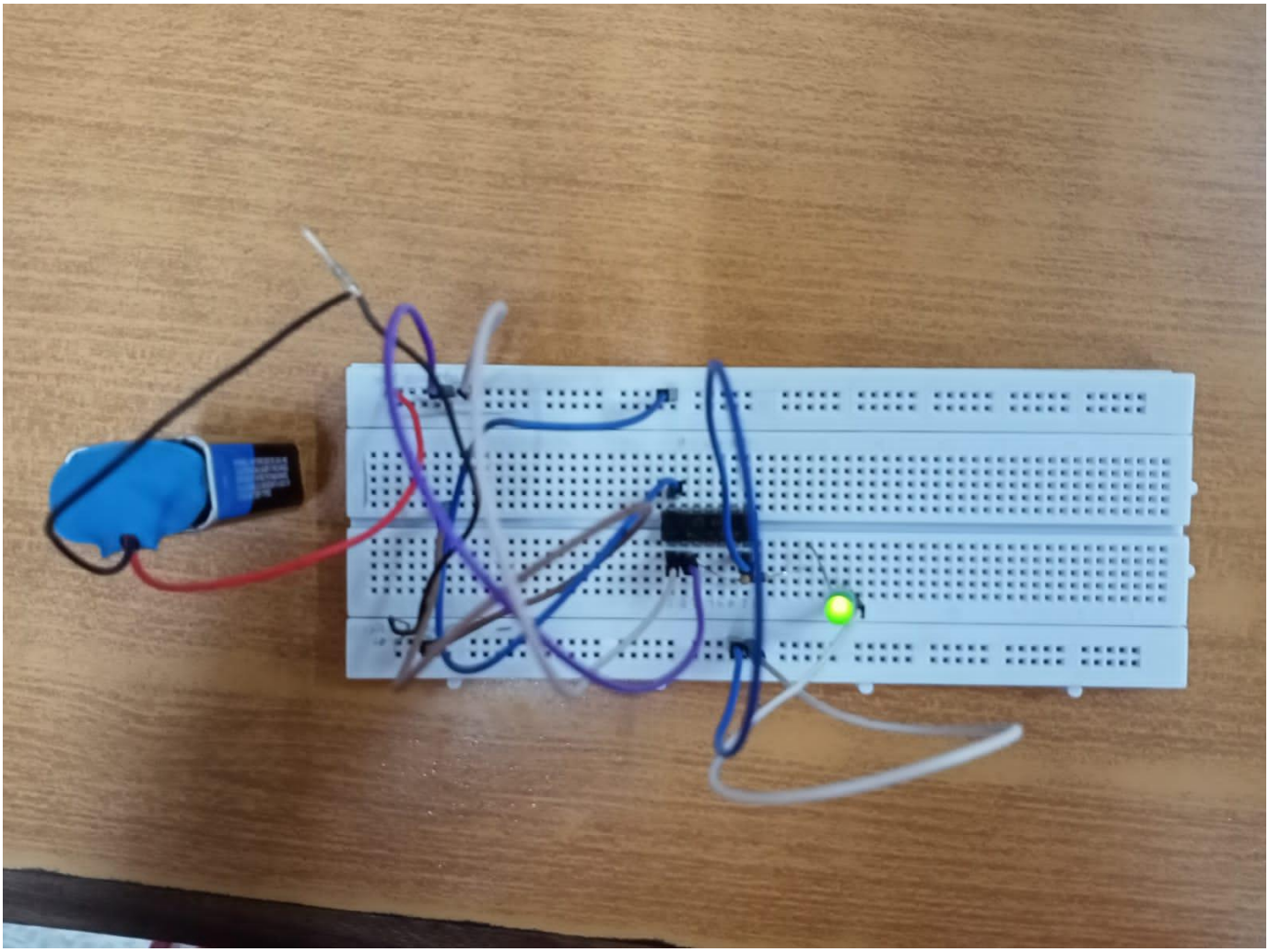
RESULT:

Not Gate :

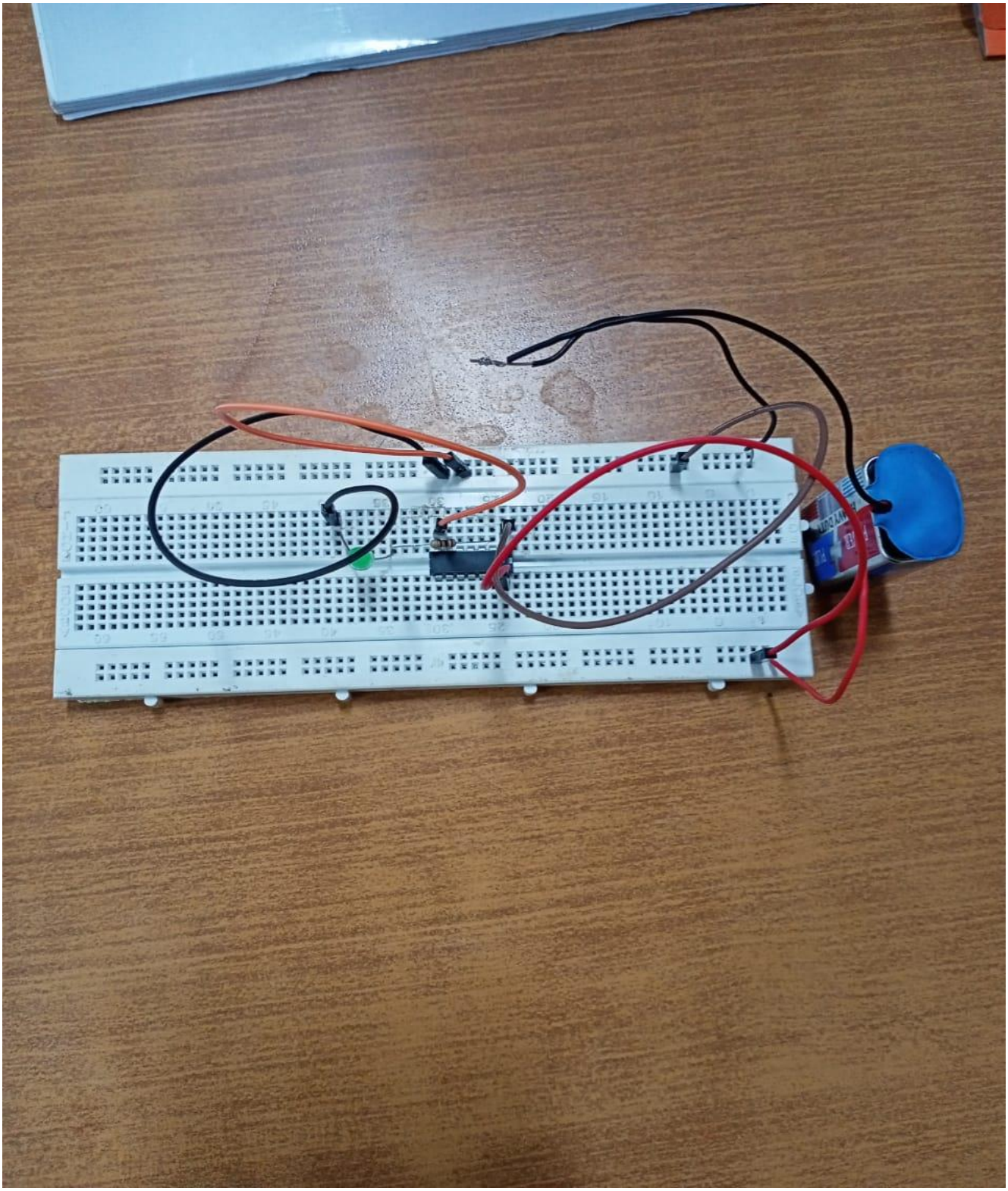
A	B	A.B	A+B	A	A'
0	0	0	0	1	0
0	1	0	1	0	1
1	0	0	1		
1	1	1	1		



AND CIRCUIT



OR CIRCUIT



NOT CIRCUIT

LABSESSION02

Design a half adder circuit

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION02

OBJECTIVE:

Design a half adder circuit.

THEORY:

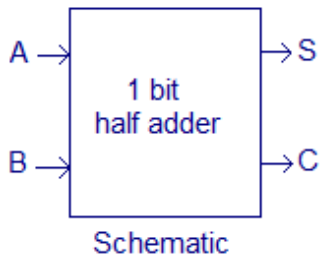
To understand what is a half adder you need to know what is an adder first. Adder circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Typically adders are realized for adding binary numbers but they can be also realized for adding other formats like BCD (binary coded decimal, XS-3 etc. Besides addition, adder circuits can be used for a lot of other applications in digital electronics like address decoding, table index calculation etc. Adder circuits are of two types: Half adder ad Full adder.

Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (S) and carry bit (C) as the output. If A and B are the input bits, then sum bit (S) is the X-OR of A and B and the carry bit (C) will be the AND of A and B. From this it is clear that a half adder circuit can be easily constructed using one X-OR gate and one AND gate. Half adder is the simplest of all adder circuit, but it has a major disadvantage. The half adder can add only two input bits (A and B) and has nothing to do with the carry if there is any in the input. So if the input to a half adder have a carry, then it will be neglected it and adds only the A and B bits. That means the binary addition process is not complete and that's why it is called a half adder. The truth table, schematic representation and XOR//AND realization of a half adder are shown in the figure below.

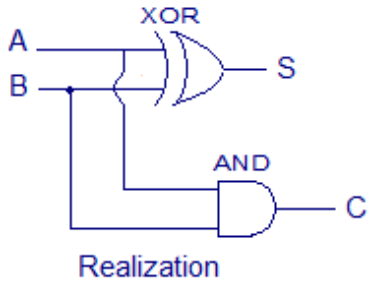
TRUTH TABLE:

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



Schematic



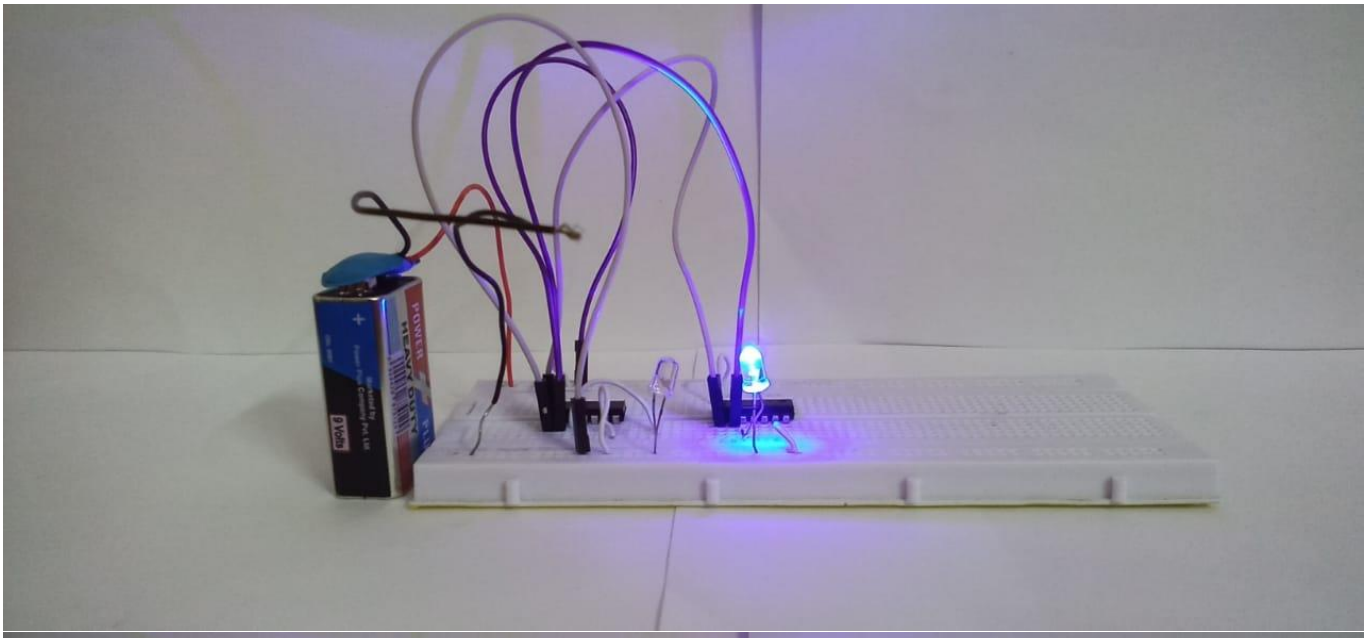
Realization

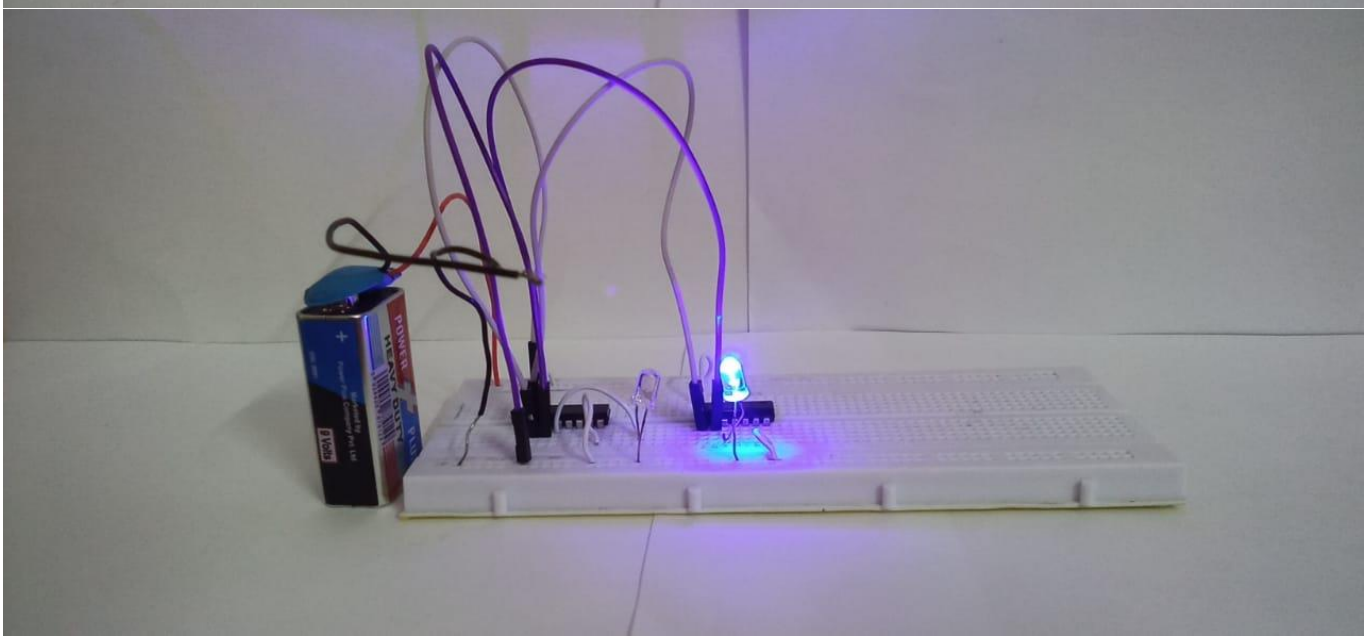
OBSERVATIONS:

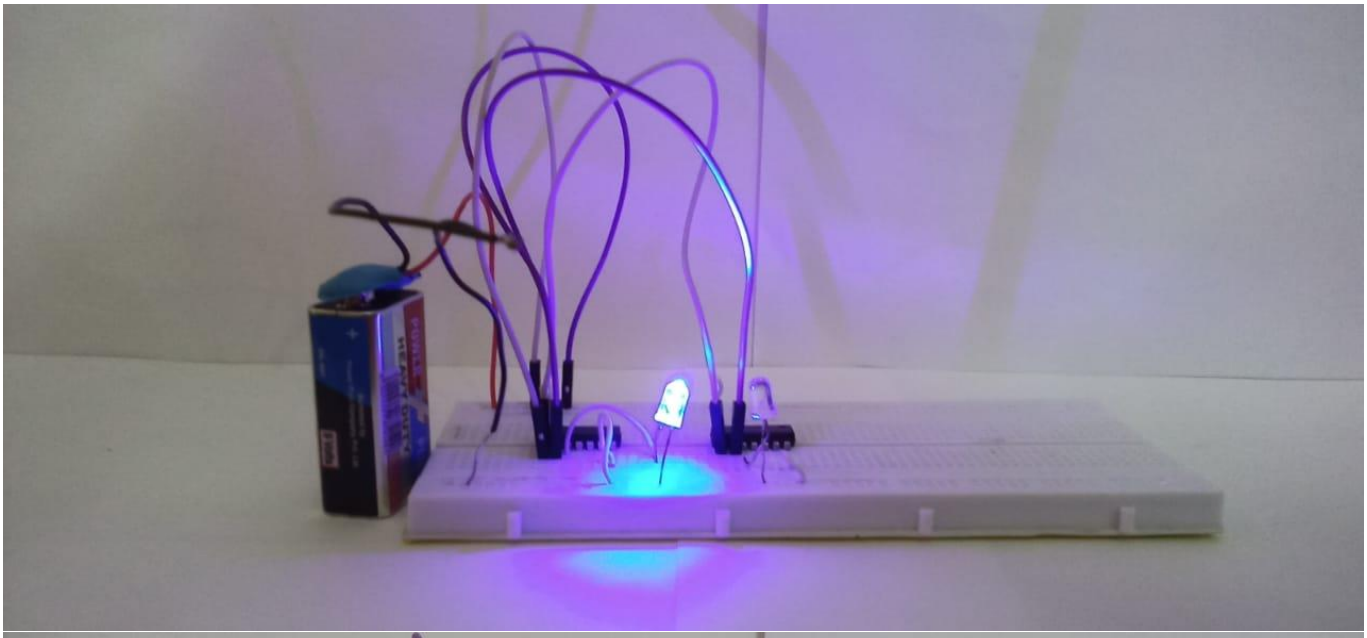
A	B	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

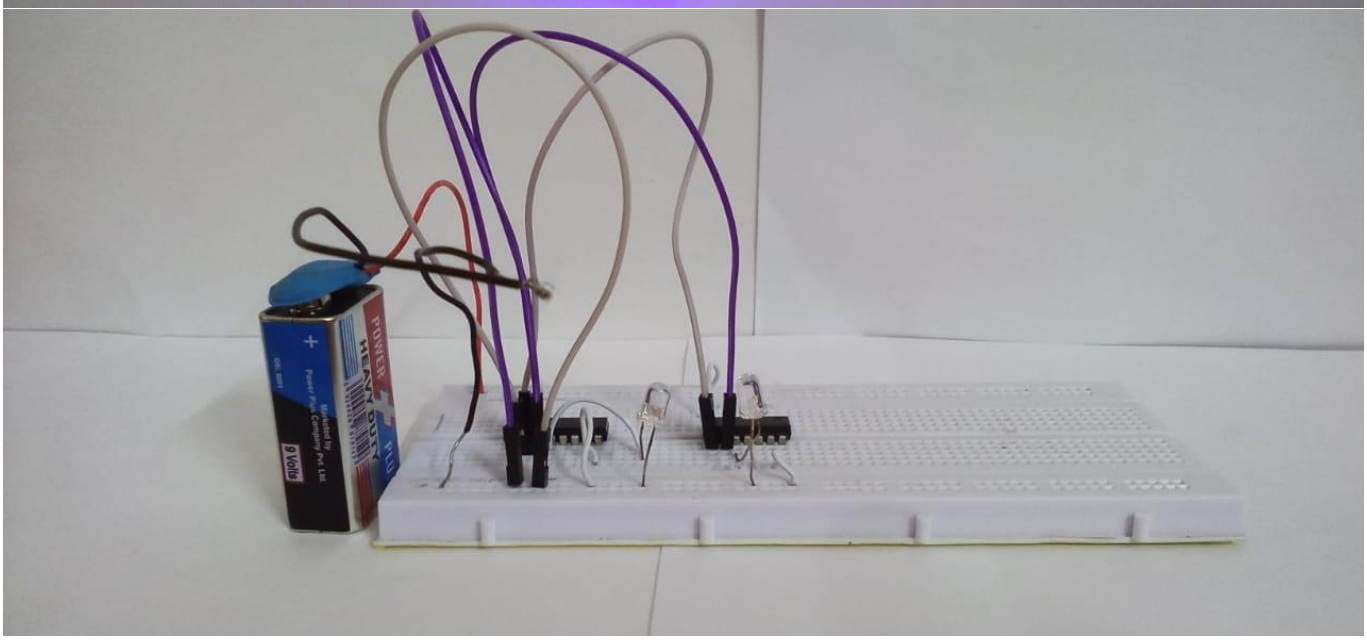
RESULT:

The half adder circuit was implemented on a bread board using ICs.








POWER
HEAVY DUTY
9 Volts

LABSESSION03

Design a full adder circuit.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

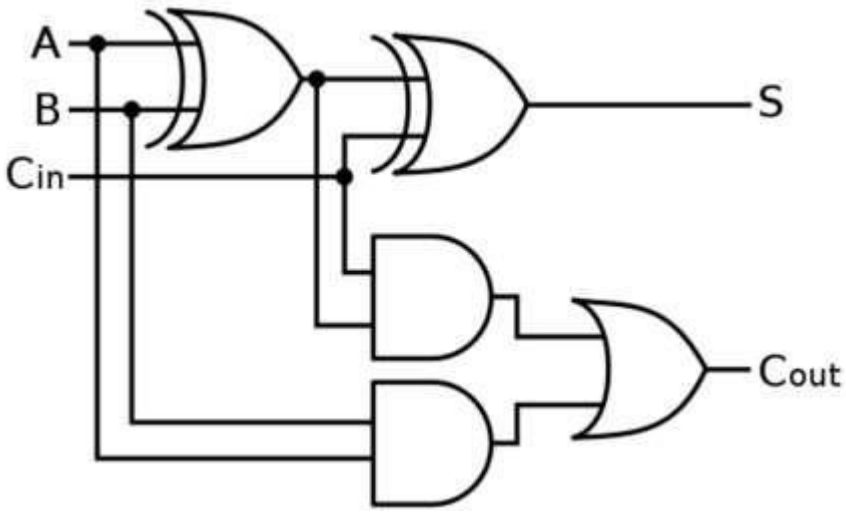
LABSESSION03

OBJECTIVE:

Design a full adder circuit.

THEORY:

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A, B, and C_{in} ; A and B are the operands, and C_{in} is a bit carried in from the next less significant stage. The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. binary numbers. The circuit produces a two-bit output, output carry and sum typically represented by the signals C_{out} and S.



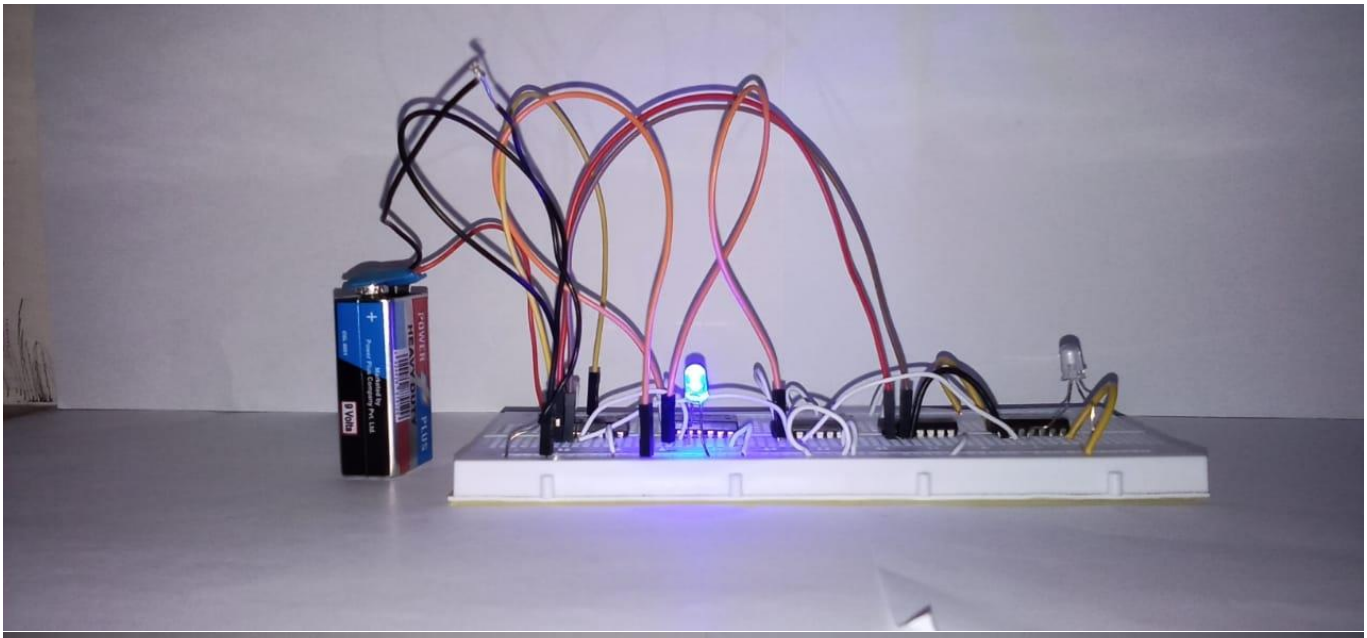
OBSERVATIONS:

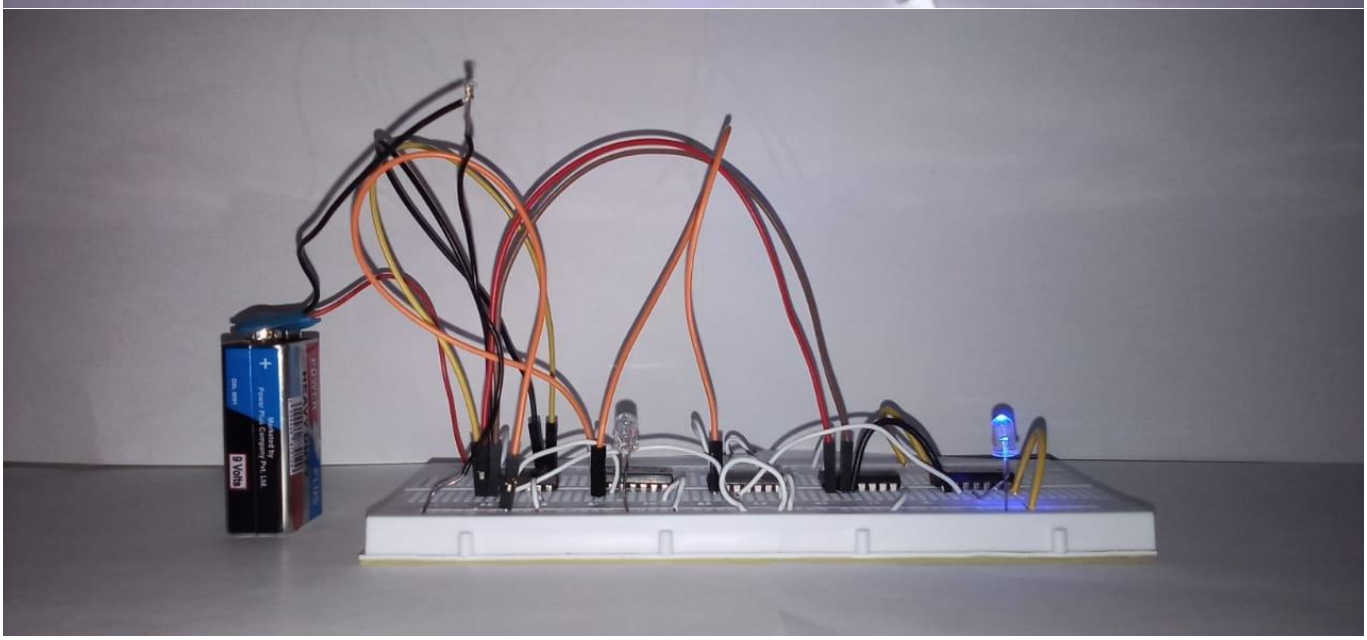
The required outputs observed as described in the truth table for sum and carry out are as follows.

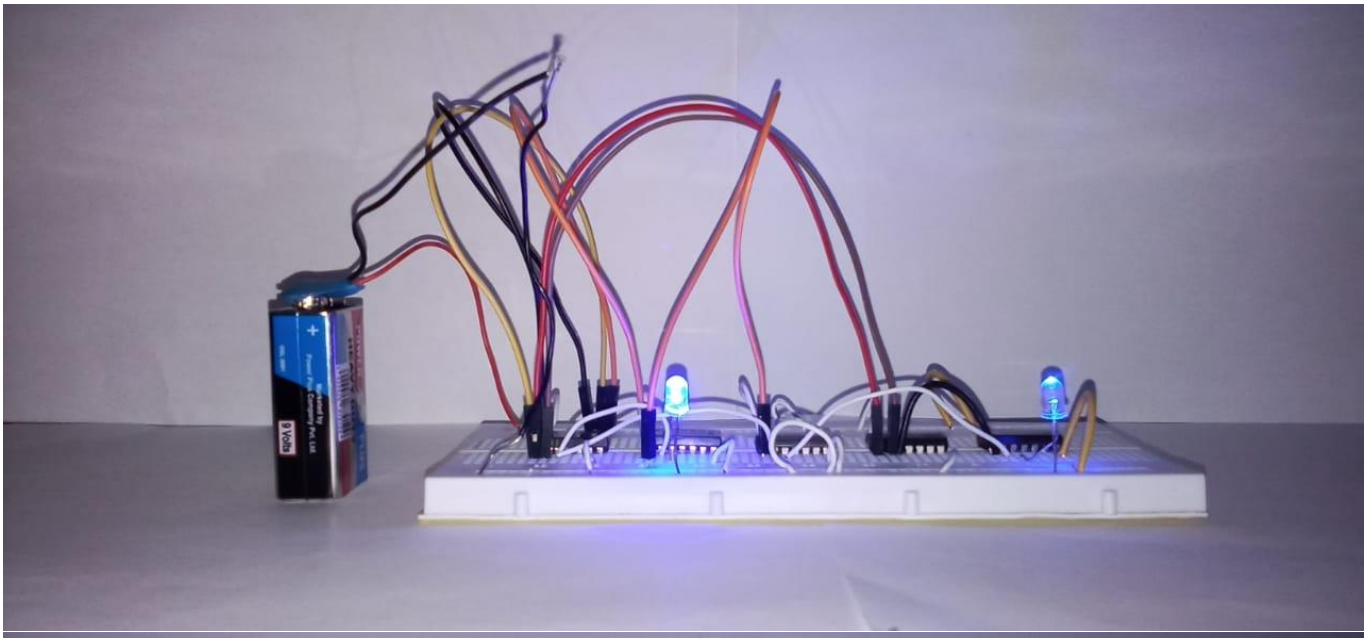
A	B	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

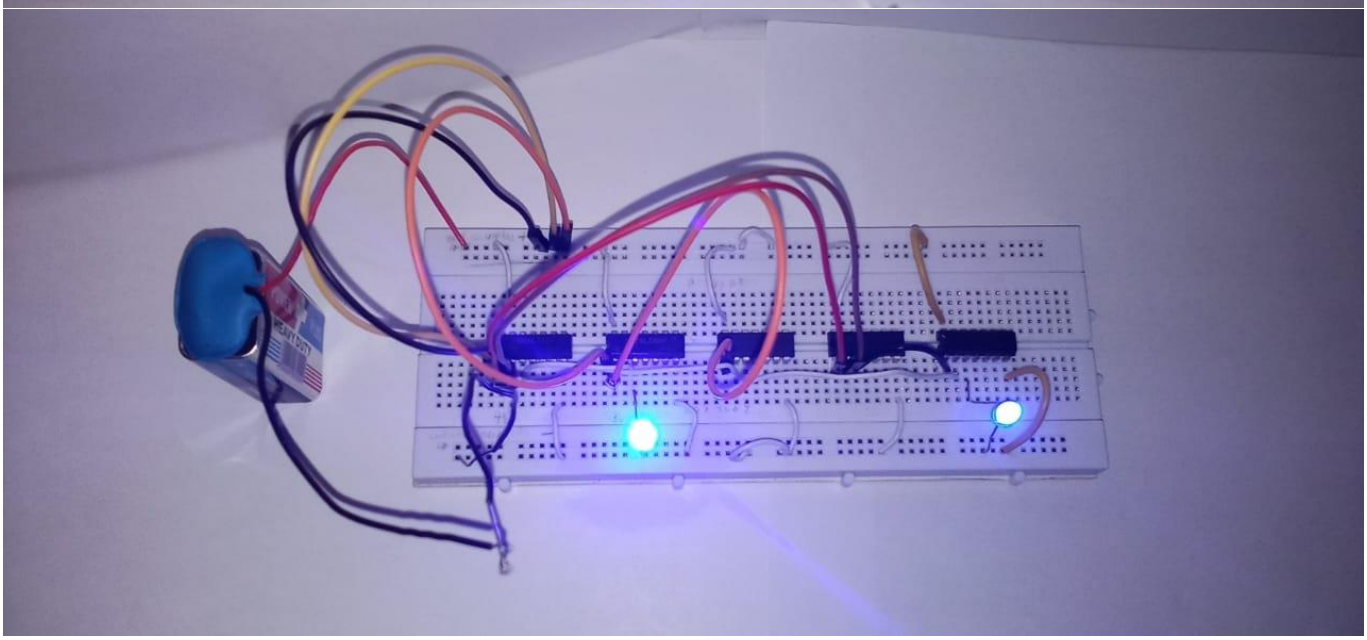
RESULT:

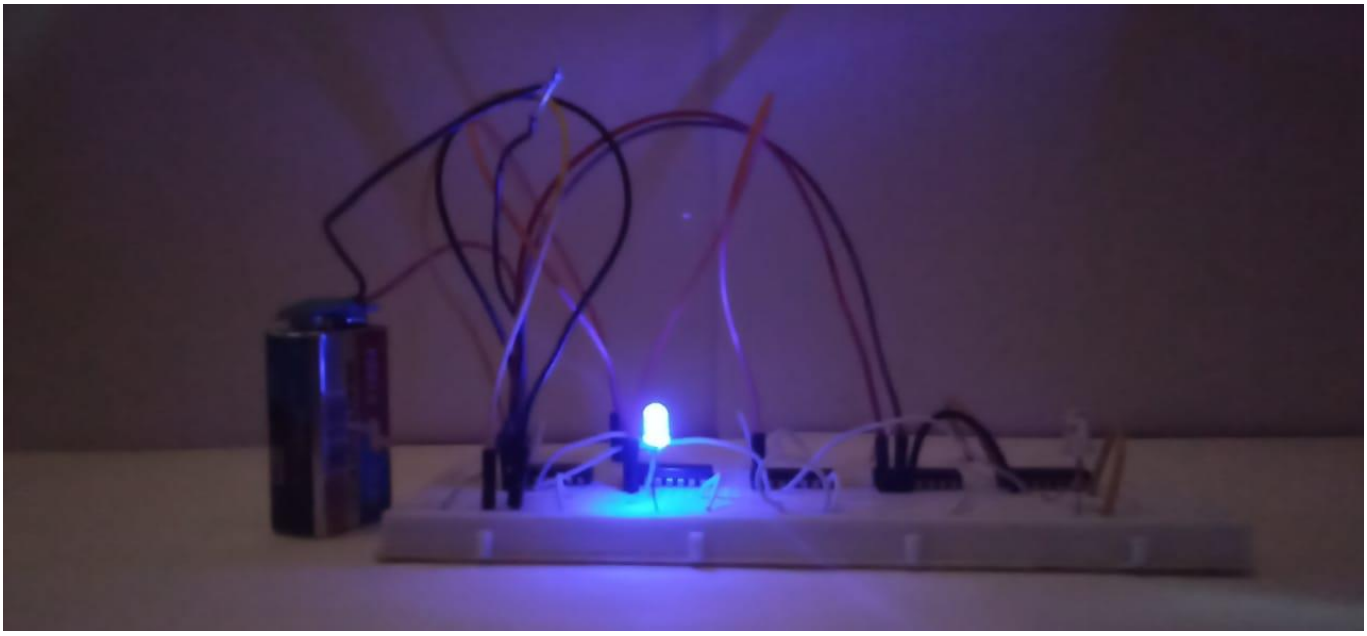
The Full Adder circuit was implemented using discrete IC and the outputs of sum and carry out were observed on LEDs.











LABSESSION 04

Design and Implement a four bit adder using logic gate ICs

Student Name: ALI MUSAWIR

Roll Number: El-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

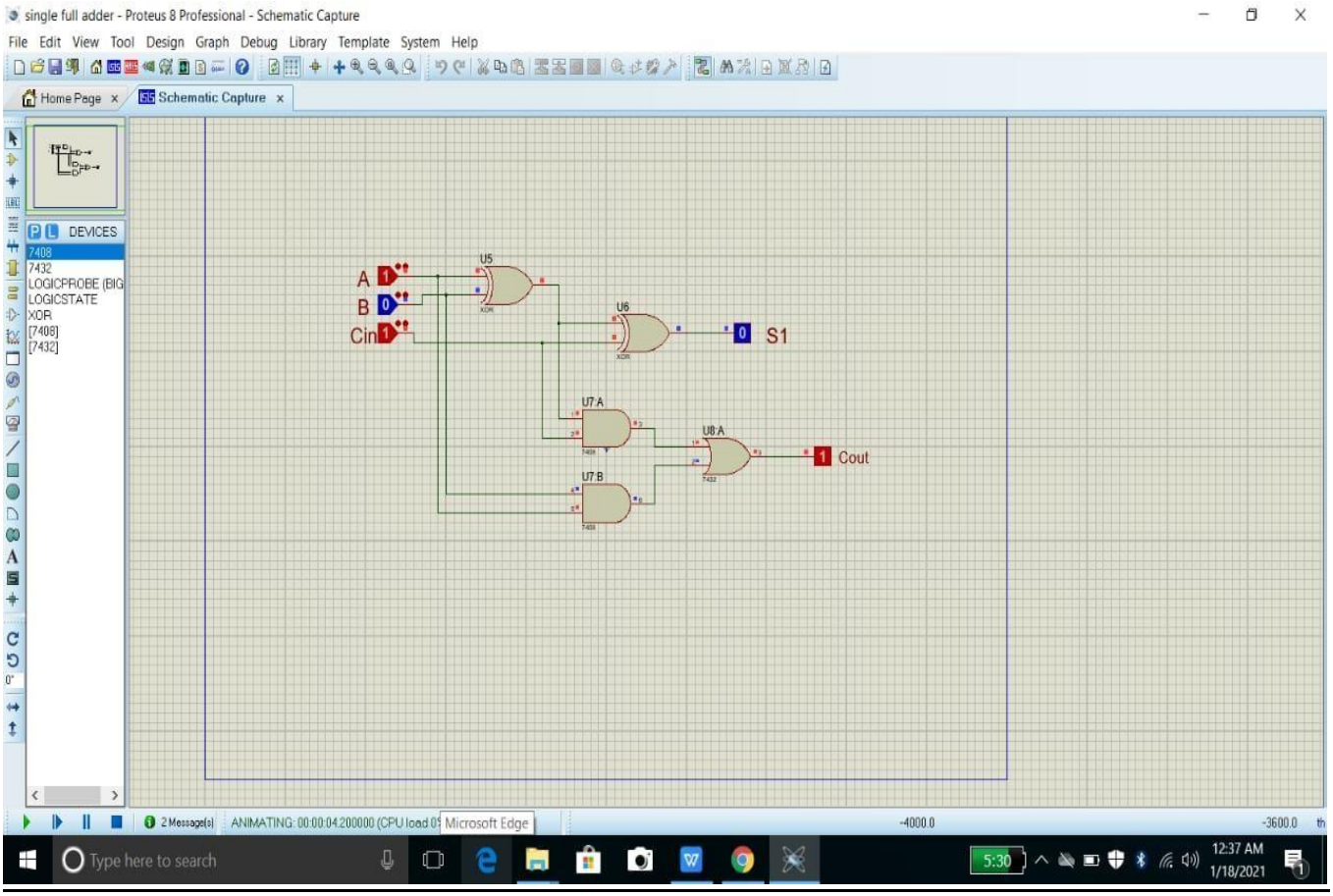
Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

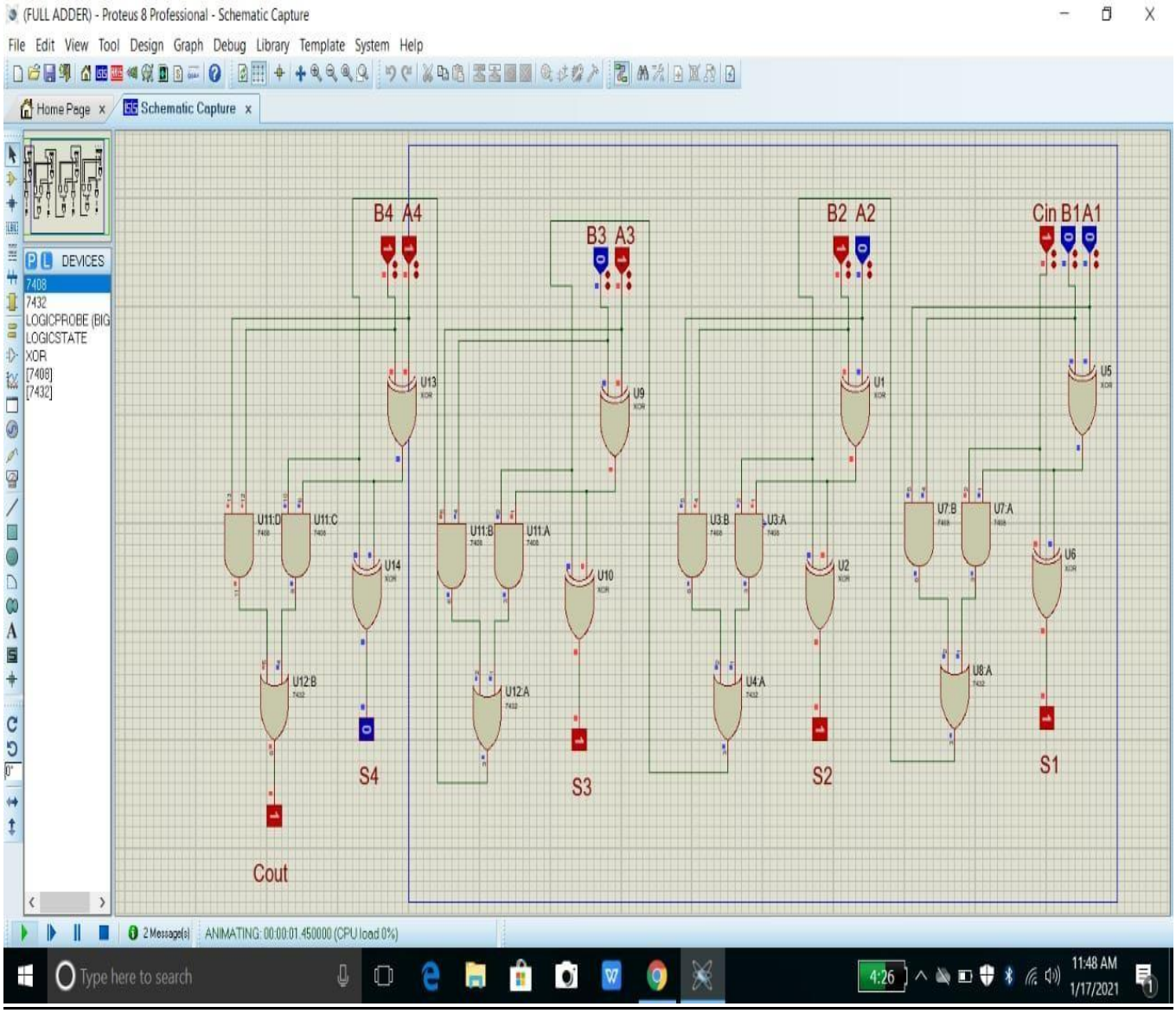
OBJECTIVE:

- Design and Implement a four bit adder using logic gate ICs.



Above figure represents the schematic of single bit full adder and our 4 bit full adder made by the combination of these 4 single bit full adders

FOUR BIT FULL ADDER


(FULL ADDER) - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Home Page
Schematic Capture
DEVICES
7408
7432
LOGICPROBE (BIG
LOGICSTATE
XOR
[7408]
[7432]
B4 A4
B3 A3
B2 A2
Cin B1A1
U13
U9
U1
U5
U11:D
U11:C
U11:B
U11:A
U3:B
U3:A
U7:B
U7:A
U14
U10
U2
U6
U12:B
U12:A
U4:A
U8:A
S4
S3
S2
S1
Cout
2 Message(s)
ANIMATING: 00:00:01.450000 (CPU load 0%)
Type here to search
4:26
11:48 AM
1/17/2021

	A				B				Sum				Carry
Cin	A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0	Cout
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

CONCLUSION:

So, in this open ended lab we try to simulate the 4 bit full adder by the help of single bit adder and these single bit adder can be further formed by 2 half adders. The adder circuit are the most important circuit in digital signal processing, used in multiplication bit additional many more applications also it is the basic cell of Arithematic circuits

LABSESSION 05

Design and Implement a four bit subtractor using logic gate ICs

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

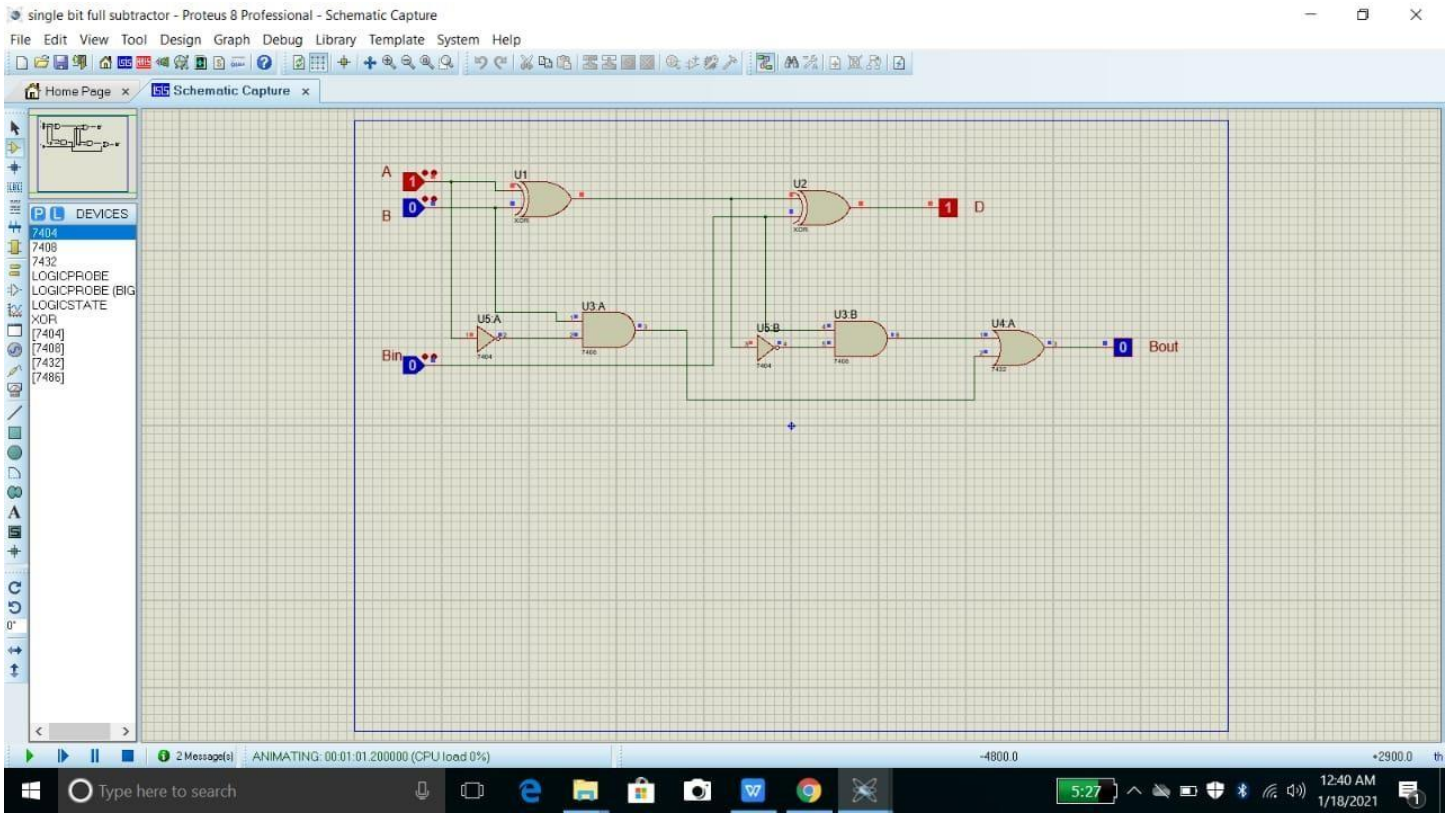
Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

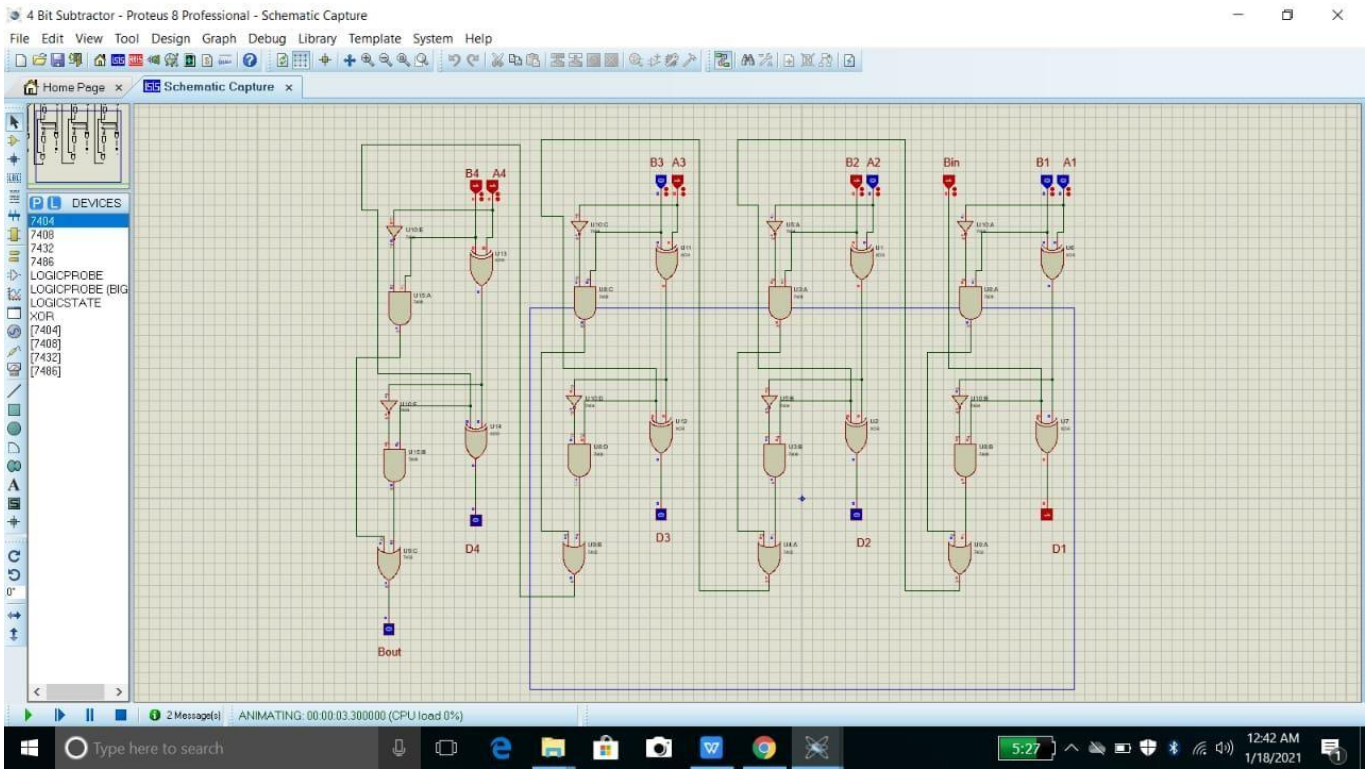
OBJECTIVE:

Design and Implement a four bit subtractor using logic gate ICs.



Above figure represents the schematic of single bit full subtractor and our 4 bit full adder made by the combination of these 4 single bit full subtractors

FOUR BIT FULL SUBTRACTOR



CONCLUSION:

So in this lab we try to simulate a four bit subtractor by the combination of single full bit subtractor. In Digital Circuits, A **Binary Subtractor** is one which is capable of subtraction of binary numbers. The operation being performed depends upon the binary value the control signal holds. It is one of the components of the ALU (Arithmetic Logic Unit).Thus we conclude that the operation of subtraction is the opposite to that of addition.

LABSESSION 06

To analyze the operation of BCD to 7-segment decoder.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

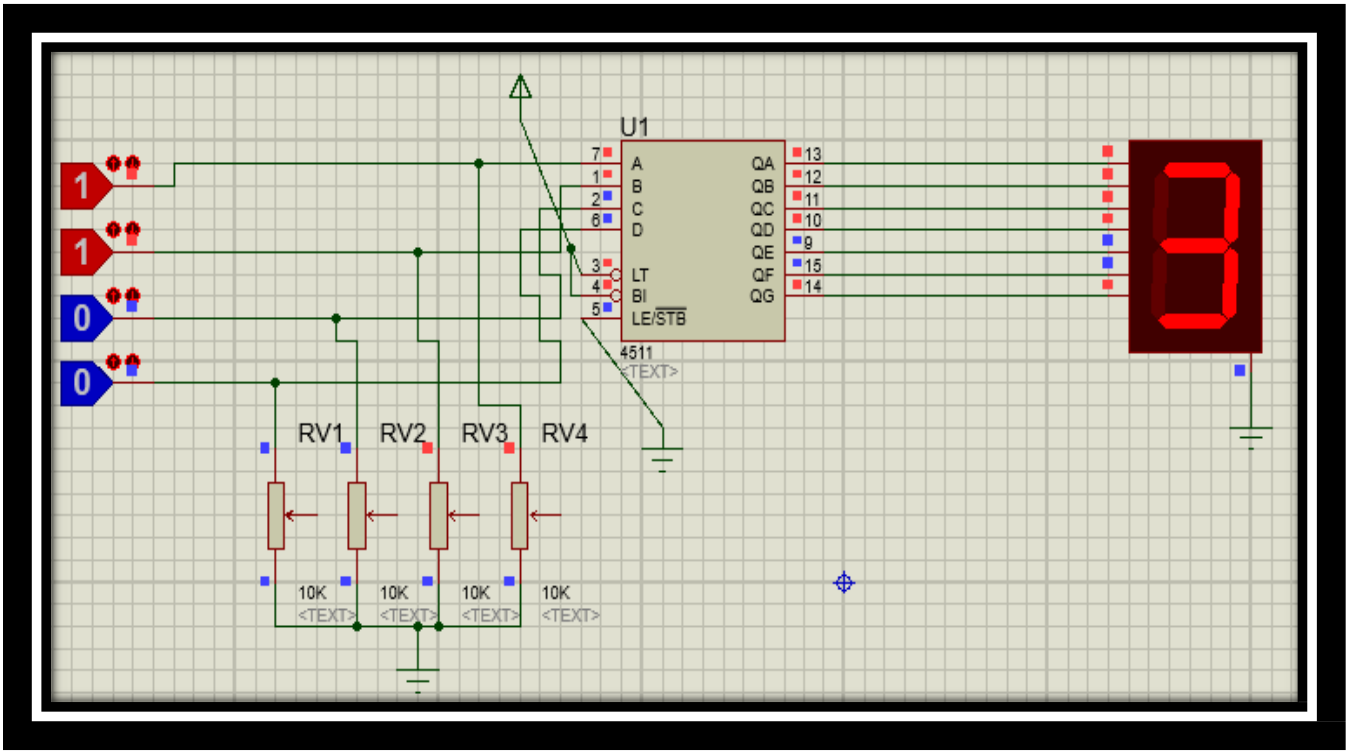
Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION 06

OBJECTIVE:

To analyze the operation of BCD to 7-segment decoder.



CIRCUIT DIAGRAM:

THEORY:

Binary Coded Decimal (BCD or 8421 code) is a way to express each decimal digit (0-9) with a binary code of four bits (0000-1001). With 4 bits, sixteen numbers (0000-1111) can be represented but in BCD only ten of these are used. The six codes combinations that are not used are called “invalid codes”.

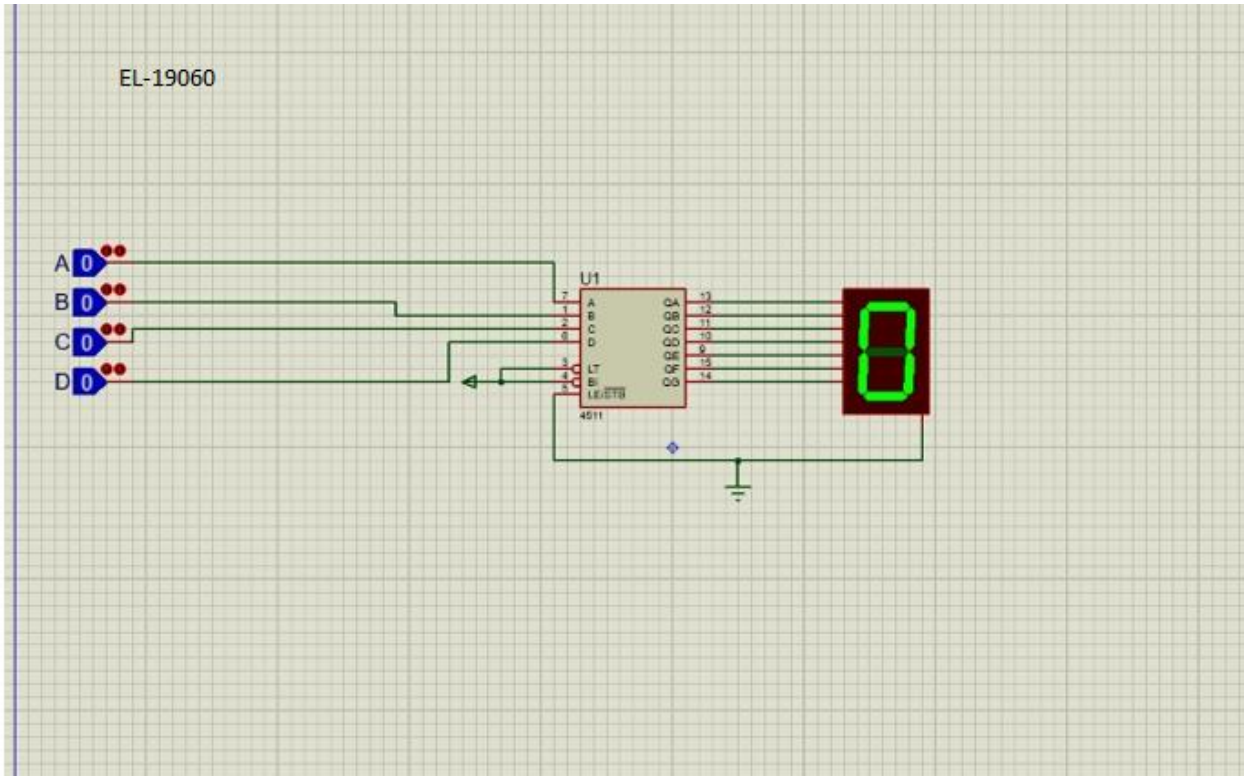
A BCD to 7-segment display decoder such as 4511, has 4 BCD inputs and 7 output lines, one for each LED segment. The 4511 is designed to drive a common cathode display and won't work with a common anode display. In a common cathode display, the cathodes of all the LEDs are joined together and the individual segments are illuminated by HIGH voltages. If invalid codes, binary values greater than 1001, are connected to the inputs of the 4511, the outputs are all 0's and the display is blank.

OBSERVATIONS:

BCD Inputs				Segment Outputs							Display
D	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	1	1	1	-	-	-	-	-	-	-	-

RESULT:

The above circuit was implemented using 4511 BCD to 7-segment decoder and a common cathode display.


EL-19060
A
B
C
D
0
0
0
0
U1
4511

LABSESSION07

To design an Astable multi vibrator using 555 timer and to understand Flip Flop operation.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION07

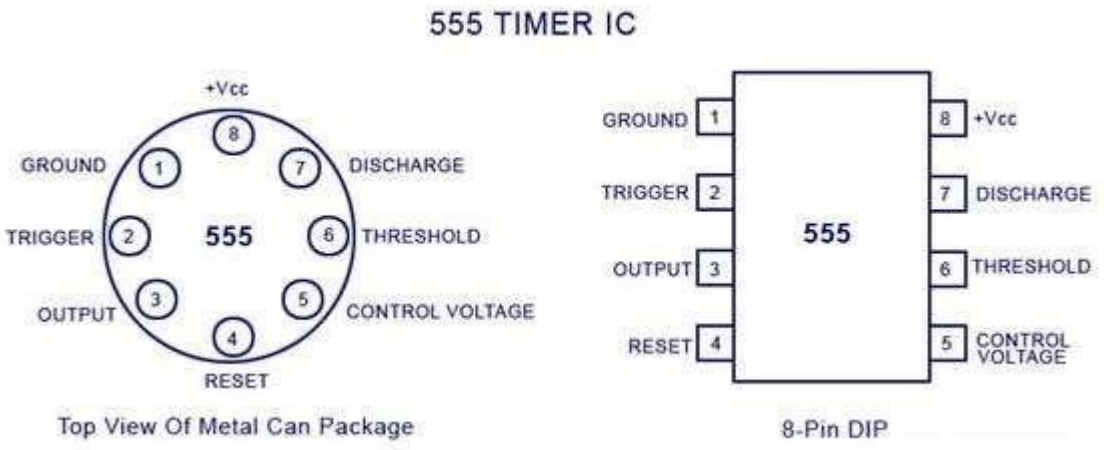
OBJECTIVE:

To design an Astable multi vibrator using 555 timer and to understand Flip Flop operation.

THEORY:

555 IC is a monolithic timing circuit that can produce accurate and highly stable time delays or oscillation. Like other commonly used op-amps, this IC is also very much reliable, easy to use and cheaper in cost. It has a variety of applications including **monostable** and **astable multi vibrators**, **dc-dc converters**, digital logic probes, **waveform generators**, analog frequency meters and tachometers, **temperature measurement** and control devices, **voltage regulators** etc. The timer basically operates in one of the two modes either as a monostable (one-shot) multivibratoror as an astable (free-running) multivibrator. The **SE 555** is designed for the operating temperature range from – 55°C to 125° while the **NE 555** operates over a temperature range of 0° to70°C.

IC PIN CONFIGURATION:

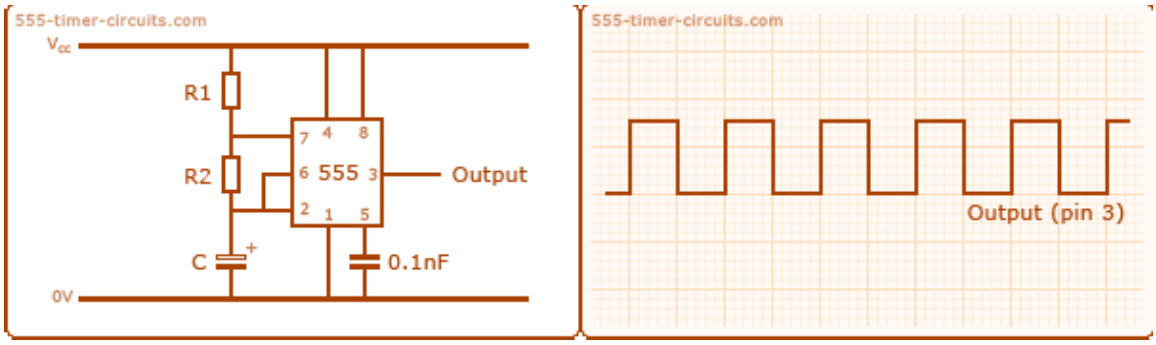


WORKING MODES:

The 555 has three main operating modes, Monostable, Astable, and Bistable. Each mode represents a different type of circuit that has a particular output.

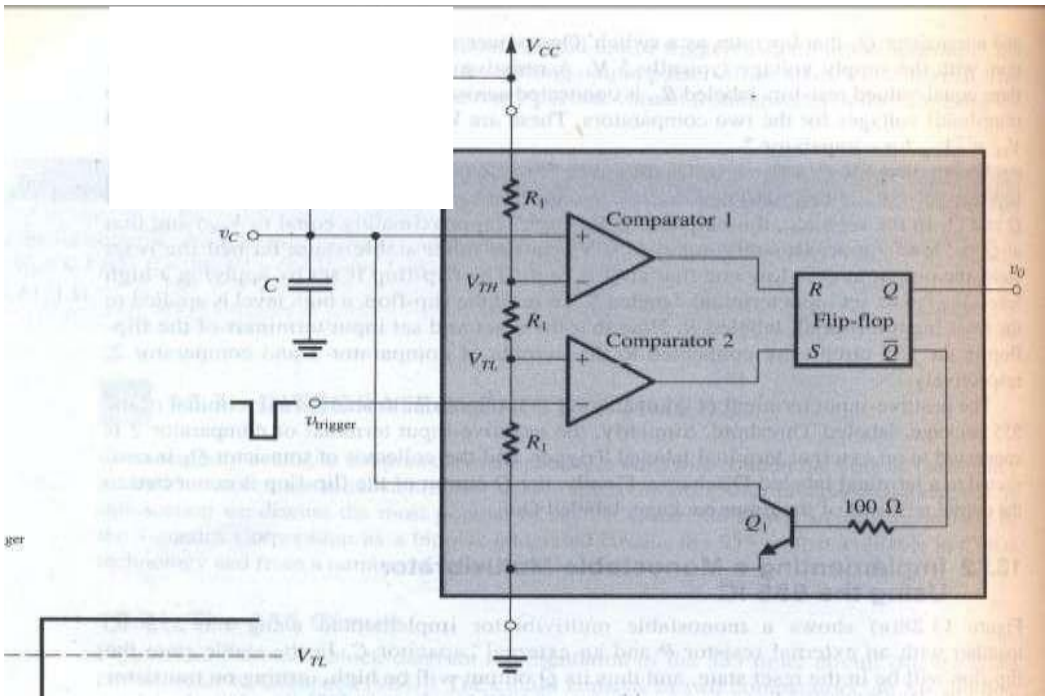
Astable mode :

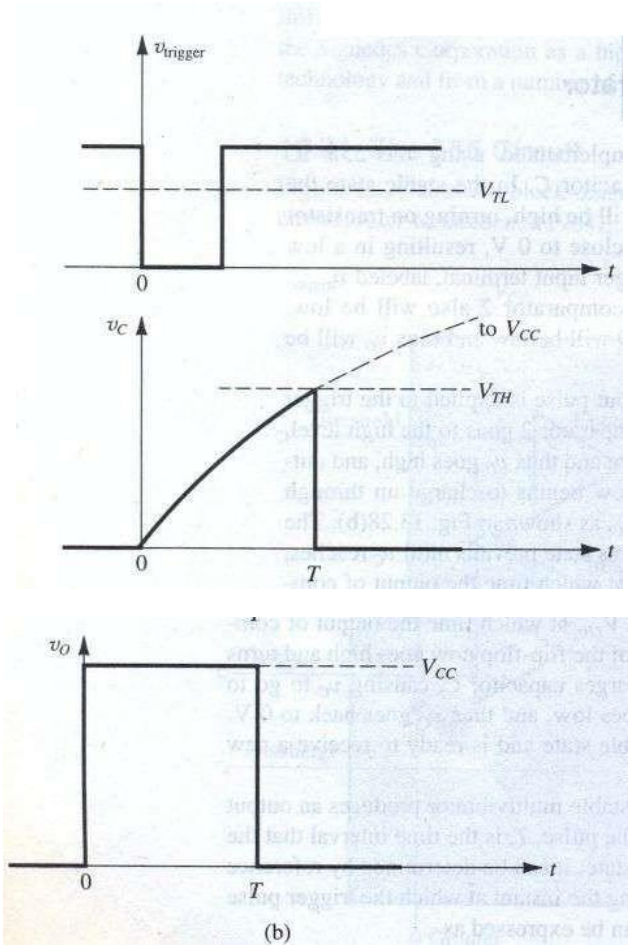
An Astable Circuit has no stable state - hence the name "astable". The output continually switches state between high and low without any intervention from the user, called a 'square' wave. This type of circuit could be used to give a mechanism intermittent motion by switching a motor on and off at regular intervals. It can also be used to flash lamps and LEDs, and is useful as a 'clock' pulse for other digital ICs and circuits.



Monostable mode :

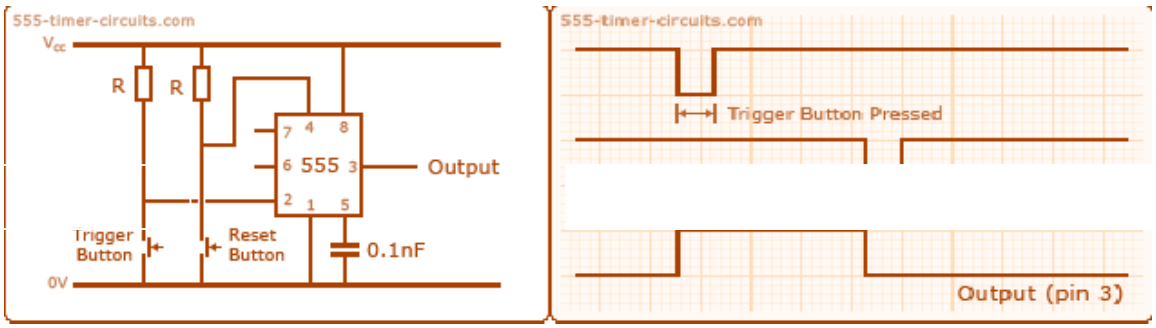
A Monostable Circuit produces one pulse of a set length in response to a trigger input such as a push button. The output of the circuit stays in the low state until there is a trigger input, hence the name "monostable" meaning "one stable state". his type of circuit is ideal for use in a "push to operate" system for a model displayed at exhibitions. A visitor can push a button to start a model's mechanism moving, and the mechanism will automatically switch off after a set time.



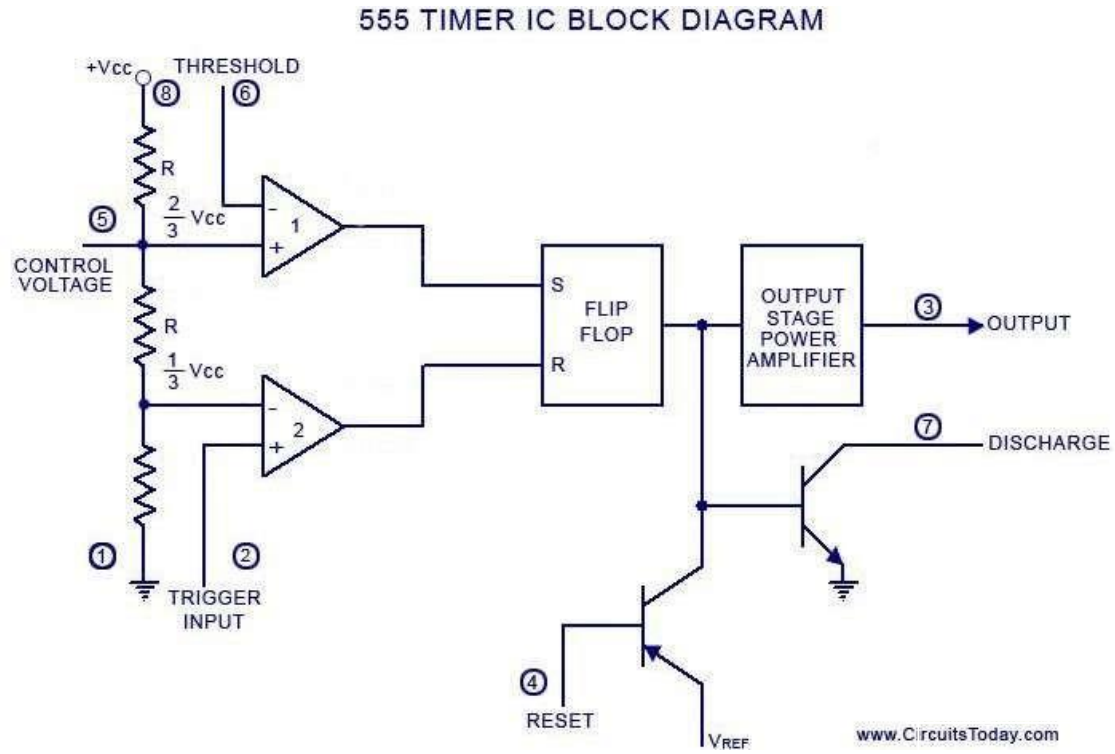


Bistable Mode (or Schmitt Trigger):

A Bistable Mode or what is sometimes called a Schmitt Trigger, has two stable states, high and low. Taking the Trigger input low makes the output of the circuit go into the high state. Taking the Reset input low makes the output of the circuit go into the low state. This type of circuit is ideal for use in an automated model railway system where the train is required to run back and forth over the same piece of track. A push button (or reed switch with a magnet on the underside of the train) would be placed at each end of the track so that when one is hit by the train, it will either trigger or reset the bistable. The output of the 555 would control a DPDT relay which would be wired as a reversing switch to reverse the direction of current to the track, thereby reversing the direction of the train.



FLIP FLOP OPERATION:



The block diagram of a **555 timer** is shown in the above figure. A 555 timer has two comparators, which are basically 2 op-amps), an R-S flip-flop, two transistors and a resistive network.

- Resistive network consists of three equal resistors and acts as a voltage divider.
- Comparator 1 compares threshold voltage with a reference voltage + 2/3 V_{CC} volts.
- Comparator 2 compares the trigger voltage with a reference voltage + 1/3 V_{CC} volts.

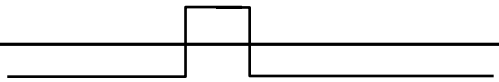
Output of both the comparators is supplied to the flip-flop. Flip-flop assumes its state according to the output of the two comparators. One of the two transistors is a discharge transistor of which collector is connected to**pin**7. This transistor saturates or cuts-off according to the output state of the flip-flop. The saturated transistor provides a discharge path to a capacitor connected externally. Base of another transistor is connected to a reset terminal. A pulse applied to this terminal resets the whole timer irrespective of any input.

OBSERVATIONS:

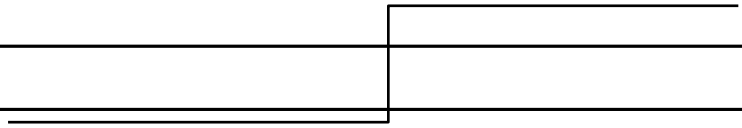
Draw here the output wave form obtained from your designed circuit.

Astable:

Monostable wave form :

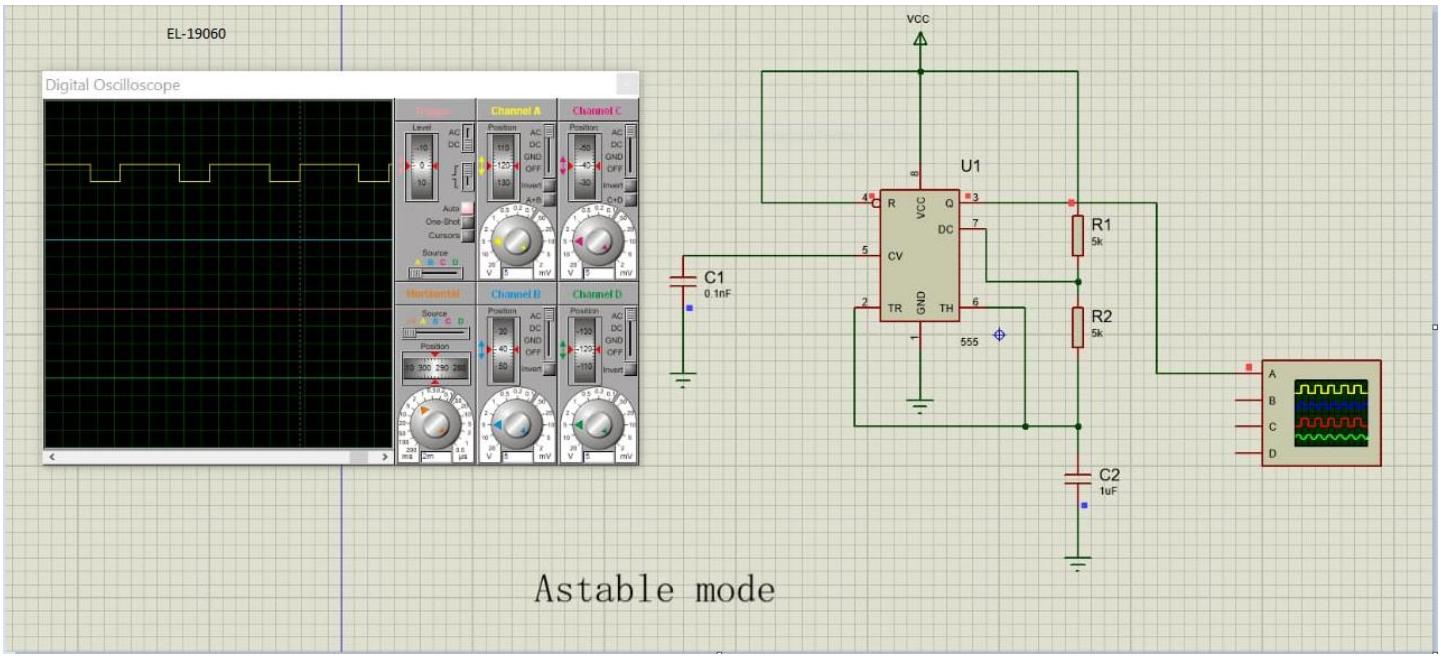


Bistable wave form :



RESULT:

The circuits were implemented and the required waveforms were observed on an oscilloscope.


EL-19060
Digital Oscilloscope
VCC
U1
R1
5k
R2
5k
C1
0.1nF
C2
1uF
555
Astable mode

LABSESSION08

To design a two bit gray counter and a two bit binary counter using J K flip flops.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION08

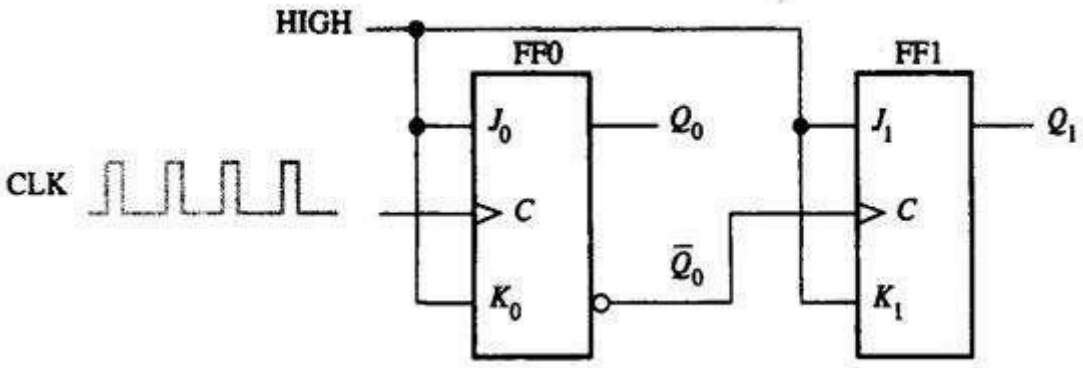
OBJECTIVE:

To design a two bit asynchronous and synchronous binary counters using J K flip flops.

THEORY:

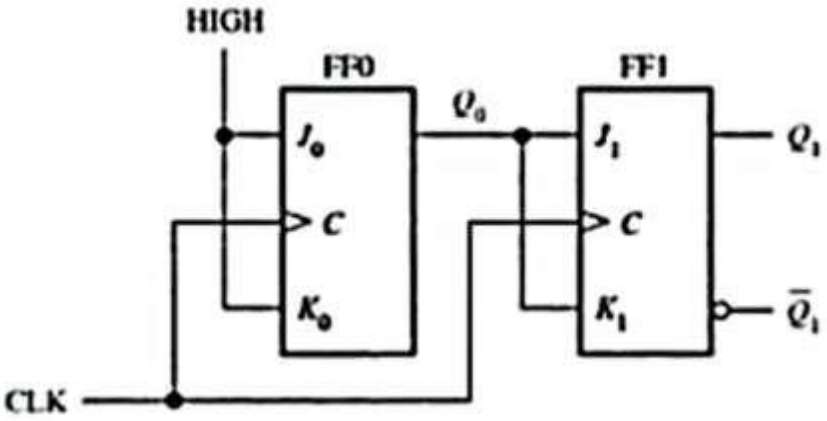
2 BIT ASYNCHRONOUS COUNTER:

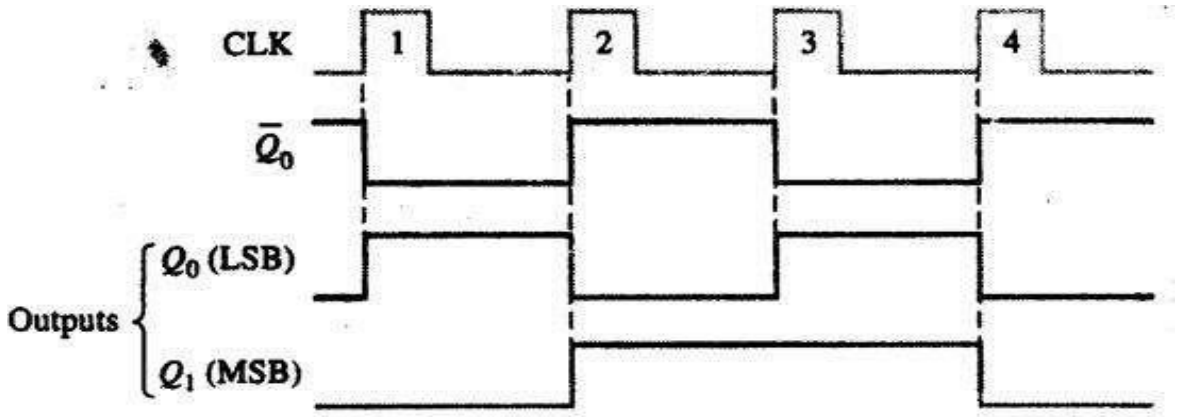
Asynchronous counter is one in which flip flops within the counter do not change states at exactly the same time because they do not have a common clock pulse.



2 BIT SYNCHRONOUS COUNTER:

Synchronous counter is one in which all the flip flops are clocked at the same time by a common clock pulse.





OBSERVATIONS:

Clk	Q1	Qo
1↑		
2↑		
3↑		
4↑		

RESULT:

LABSESSION09

To design combinational circuits using multiplexer and demultiplexer

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION09

OBJECTIVE:

To design combinational circuits using multiplexer and demultiplexer

EQUIPMENTS:

- IC type 7404 HEX inverter
- IC type 7408 quad 2-input AND gate
- IC type 74151 8x1 multiplexer(1)
- IC type 74153 dual 4x1 multiplexer(2)
- IC type 7446 BCD-to-Seven-Segment decoder(1)
- Resistance network(1)
- Seven-Segment Display (1)

THEORY:

74151 is a 8 line-to-1 line multiplexer. It has the schematic representation shown in Fig 1. Selection lines S2, S1 and S0 select the particular input to be multiplexed and applied to the output.
Strobe S acts as an enable signal. If strobe =1, the chip 74151 is disabled and output y = 0. If strobe = 0 then the chip 74151 is enabled and functions as a Multiplexer. Table 1 shows the multiplex function of 74151 in terms of select lines.

Table 1.

Strobe	Select Lines			Output
S	S_2	S_1	S_0	Y
1	X	X	X	0
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7

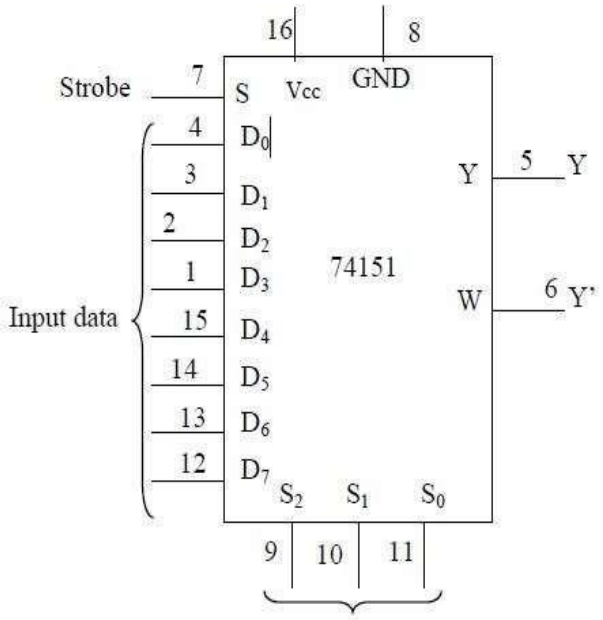


Fig.1 IC type 74151 Multiplexer 8×1

74153 is a dual 4 line-to-1 line multiplexer. It has the schematic representation shown in Fig 2. Selection lines S1 and S0 select the particular input to be multiplexed and

applied to the output IY{1 = 1, 2}.
Each of the strobe signals *IG* {I = 1, 2} acts as an enable signal for the corresponding multiplexer.

Table 2.shows the multiplex function of 74153 in terms of select lines. Note that each of the on-chip multiplexers act independently from the other, while sharing the same select lines S1 and S0.

Table 2

	Multiplexer 1		
Strobe	Select lines		Output
1G	S_1	S_0	1Y
1	X	X	0
0	0	0	$1D_0$
0	0	1	$1D_1$
0	1	0	$1D_2$
0	1	1	$1D_3$

	Multiplexer 2		
Strobe	Select lines		Output
2G	S_1	S_0	2Y
1	X	X	0
0	0	0	$2D_0$
0	0	1	$2D_1$
0	1	0	$2D_2$
0	1	1	$2D_3$

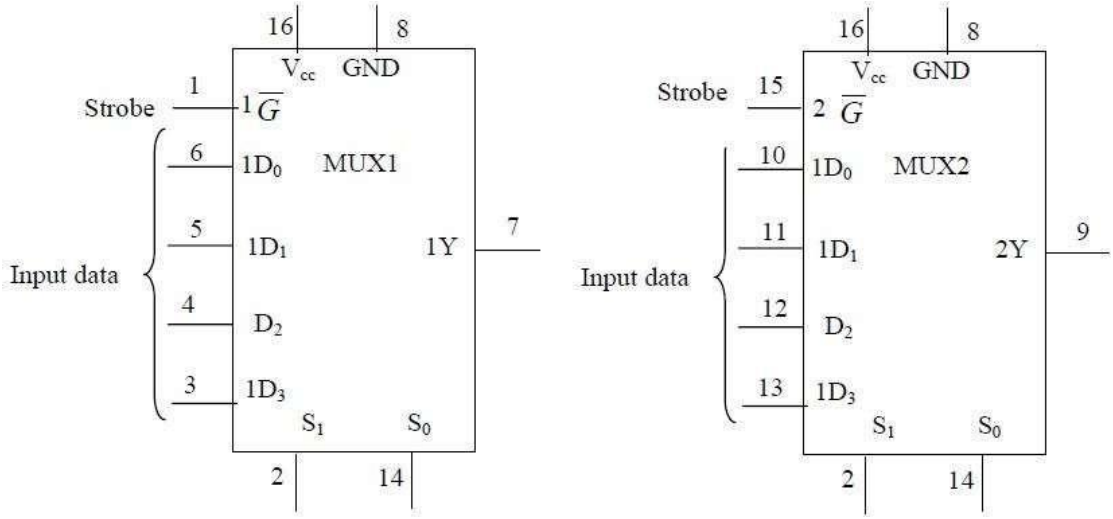


Fig.2 Pin out of 74153

IC 7446 is a BCD to seven segment decoder driver. It is used to convert the Combinational circuit outputs in BCD forms into 7 segment digits for the 7 segment LED display units.

PROCEDURE:

Part I: Parity Generator:

a) Design a parity generator by using a 74151 multiplexer. Parity is an extra bit attached to a code to check that the code has been received correctly .Odd parity bit means that the number of 1's in the code including the parity bit is an odd number. Fill the output column of the truth table in Table 2 for a 5-bit code in which four of the bits (A,B,C,D) represents the information to be sent and fifth bit (x), represents the parity bit. The required parity is an odd parity.

The inputs B,C and D correspond to the select inputs of 74151. Complete the truth table in Table 3 by filling in the last column with 0,1, A or A'.

b) Simulate the circuit using proteus , use 74-151 multiplexer and Binary switches for inputs and Binary Probes for outputs. The 74151 has one output for Y and another inverted output W. Use A and A' for providing values for inputs 0-7. The internal values "A, B, C" are used for selection inputs B, C, and D. Simulate the circuit and test each input combination filling in the table shown below. In the Lab connect the circuit and verify the operations. Connect an LED to the multiplexer output so that it represents the parity bit which lights any time when the four bits input have even parity.

c) **TRUTH TABLE:**

A	B	C	D	X	Connect data to
0	0	0	0	1	A'
0	0	0	1	0	A
0	0	1	0	**0**	A
0	0	1	1	1	A'
0	1	0	0	0	A
0	1	0	1	1	A'
0	1	1	0	1	A'
0	1	1	1	0	A
1	0	0	0	0	A
1	0	0	1	1	A'
1	0	1	0	1	A'
1	0	1	1	0	A
1	1	0	0	1	A'
1	1	0	1	0	A
1	1	1	0	0	A

d)

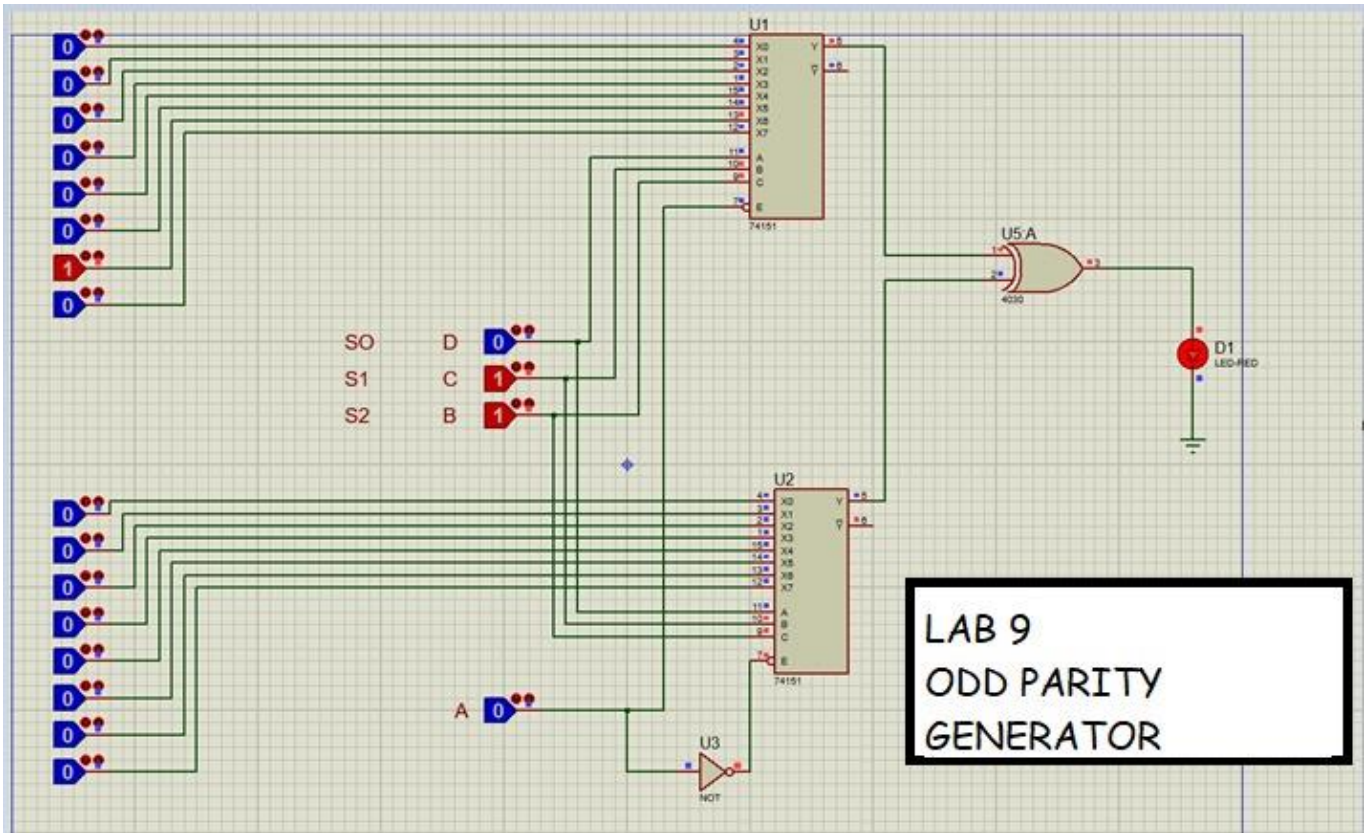
Part 2: Vote Counter:

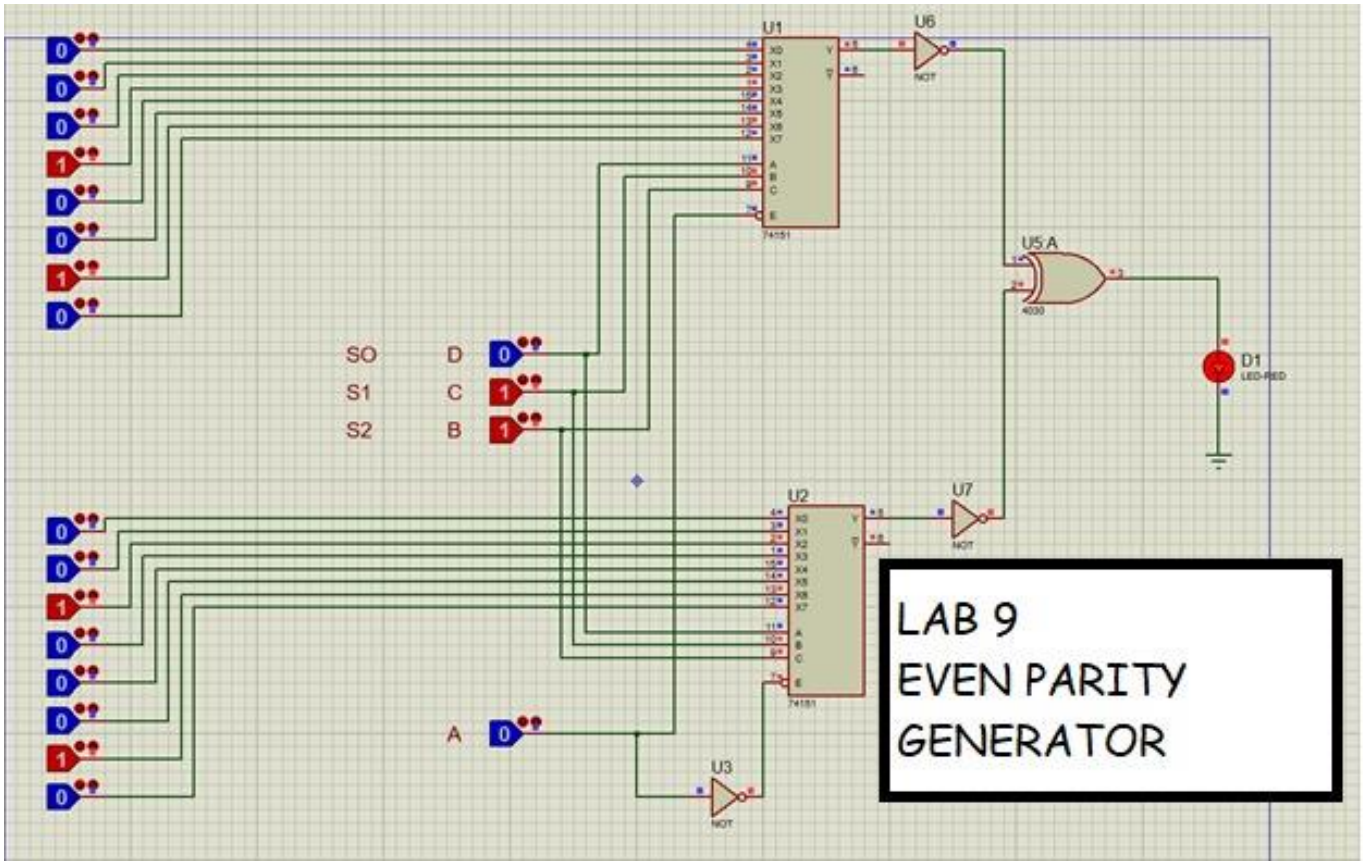
A committee is composed of a chairman (C), a senior member (S), and a member (M).

The rules of the committee state that:
• The vote of the member (M) will be counted as 2votes
• The vote of the senior member will be counted as 3 votes.
• The vote of the chairman will be counted as 5votes.
Each of these persons has a switch to close (“l”) when voting yes and to open (“0”) when voting no.

It is necessary to design a circuit that displays the total number of votes for each issue.
Use a seven segment display and a decoder to display the required number.
If all members vote no for an issue the display should be blank. (Recall from Experiment #5, that a binary input 15 into the 7446 blanks all seven segments).
If all members vote yes for an issue, the display should be 0. Otherwise the display shows a decimal number equal to the number of 'yes' votes. Use two 74153 units, which include four multiplexers to design the combinational circuit that converts the inputs from the members' switch to the BCD digit for the 7446.
In Proteus use +5V for Logic 1 and ground for Logic 0 and use switches for C, S,and M. Use two chips 74153 and one decoder 7446 verify your design and get a copy of your circuit with the pin numbers to Lab so that you could connect the hardware in exactly the sameway.

RESULT:


U1
74151
U5:A
4030
D1
LED-RED
SO
S1
S2
D
C
B
A
U2
74151
U3
NOT
LAB 9
ODD PARITY
GENERATOR


U1
U6
NOT
U5:A
4030
D1
LED-RED
SO
S1
S2
D
C
B
A
U2
U7
NOT
U3
NOT
LAB 9
EVEN PARITY
GENERATOR

LABSESSION10

To analyze and study the operations of the following circuits:

- **RS and Clocked RS Flip-Flop**
- **D Flip-Flop**

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION10

OBJECTIVE:

To analyze and study the operations of the following circuits:

- RS and Clocked RS Flip-Flop
- D Flip-Flop

THEORY:

So far you have encountered with *combinatorial logic*, i.e. circuits for which the output depends only on the inputs. In many instances it is desirable to have the next output depending on the current output. A simple example is a *counter*, where the next number to be output is determined by the current number stored. Circuits that remember their current output or state are often called *sequential logic* circuits. Clearly, sequential logic requires the ability to store the current state. In other words, *memory* is required by sequential logic circuits, which can be created with Boolean gates. If you arrange the gates correctly, they will remember an input value. This simple concept is the basis of RAM (random access memory) in computers, and also makes it possible to create a wide variety of other useful circuits.

Memory relies on a concept called **feedback**. That is, the output of a gate is fed back into the input. The simplest possible feedback circuit using two inverters is shown below (Fig.1):

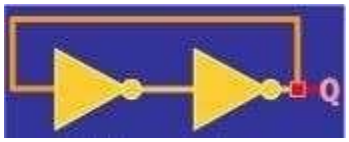


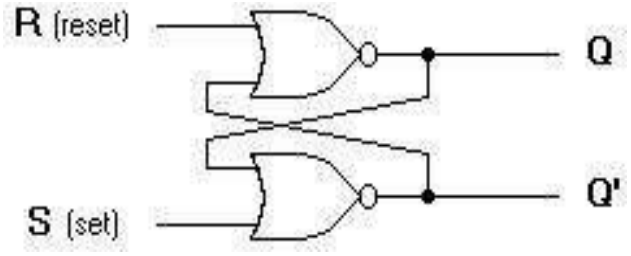
Fig.1: Simplest realization of feedback circuit

If you follow the feedback path, you can see that if Q happens to be 1 (or 0), it will always be 1 (or 0). Since it's nice to be able to control the circuits we create, this one doesn't have much use -- but it does let you see how feedback works. It turns out that in "real" sequential circuits, you can actually use this sort of simple inverter feedback approach. The memory elements in these circuits are called *flip-flops*. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Binary information can enter a flip-flop in a variety of ways and gives rise to different types offlip-flops.

RS Flip-Flop

RS flip-flop is the simplest possible memory element. It can be constructed from two NAND gates or two NOR gates. Let us understand the operation of the RS flip-flop using NOR gates as shown below using the truth table for 'A NOR B' gate. The inputs R and S are referred to as the Reset and Set inputs, respectively. The outputs Q and Q' are complements of each other and are referred to as the normal and complement outputs, respectively. The binary state of the flip- flop is taken to be the value of the normal output. When Q=1 and Q'=0, it is in the *set state* (or 1-state). When Q=0 and Q'=1, it is in the *reset/clear state* (or 0-state).

Circuit Diagram:



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

S=1 and R=0: The output of the bottom NOR gate is equal to zero, Q'=0. Hence both inputs to the top NOR gate are equal to 0, thus, Q=1. Hence, the input combination S=1 and R=0 leads to the flip-flop being **set** to Q=1.
S=0 and R=1: Similar to the arguments above, the outputs become Q=0 and Q'=1. We say that the flip-flop is **reset**.
S=0 and R=0: Assume the flip-flop was previously in set (S=1 and R=0) condition. Now changing S to 0 results Q' still at 0 and Q=1. Similarly, when the flip-flop was previously in a reset state (S=0 and R=1), the outputs do not change. Therefore, with inputs S=0 and R=0, the flip-flop holds its state.
S=1 and R=1: This condition violates the fact that both outputs are complements of each other since each of them tries to go to 0, which is not a stable configuration. It is impossible to predict which output will go to 1 and which will stay at 0. In normal operation this condition must be avoided by making sure that 1's arenot applied to both inputs simultaneously, thus making it one of the main disadvantages of RS flip-flop.

All the above conditions are summarized in the characteristic table below:

Characteristic Table:

R	S	Q	Q’	Comment
0	0	Q	Q’	Hold state
0	1	1	0	Set
1	0	0	1	Reset
1	1	?	?	Indeterminate

Debounce circuit

An elementary example using this flip-flop is the debounce circuit. Suppose a piece of electronics is to change state under the action of a mechanical switch. When this switch is moved from position S to R (S=0, R=1), the contacts make and break several times at R before settling to good contact. It is desirable that the electronics should respond to the first contact and then remain stable, rather than switching back and forth as the circuit makes and breaks. This is achieved by RS flip-flop which is reset to Q=0 by the first signal R=1 and remains in a fixed state until the switch is moved back to position S, when the signal S=1 sets the flip-flop toQ=1.

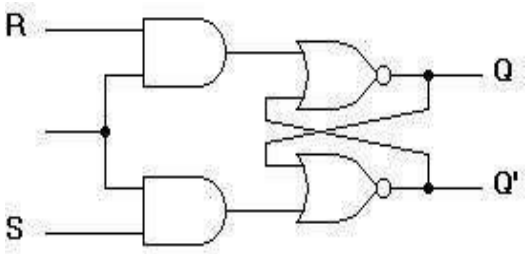
Gated or Clocked RS Flip-Flop

It is sometimes desirable in sequential logic circuits to have a bistable RS flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs. By connecting a 2-input AND gate in series with each input terminal of the RS NOR Flip-flop a Gated RS Flip-flop can be created. This extra conditional input is called an "Enable" input and is given the prefix of "EN" as shown below. When the Enable input "EN" = 0, the outputs of the two AND gates are also at logic level 0, (AND Gate principles) regardless of the condition of the two inputs S and R, latching the two outputs Q and Q' into their last known state. When the enable input "EN" = 1, the circuit responds as a normal RS bistable flip-flop with the two AND gates becoming transparent to the Set and Resetsignals. This Enable input can also be connected to a clock timing signal adding clock synchronisation tothe flip-flop creating what is sometimes called a "Clocked SR Flip-flop".

So a **Gated/Clocked RS Flip- flop** operates as a standard bistable latch but the outputs are only activated when a logic "1" is applied to its EN input and deactivated by a logic "0". The property of this flip-flop is summarized in its characteristic table where Qn
is the logic state of the previous output and Qn+1 is that of the next output and the clock input being at logic 1 for all the R and S input combinations.

Circuit Diagram:

EN/Clock pulse



Characteristic Table:

Qn	R	S	Qn+1
0	0	0	0 (Hold)
0	1	0	0
0	0	1	1
0	1	1	Indeterminate
1	0	0	1 (Hold)

1	1	0	0
1	0	1	1
1	1	1	Indeterminate

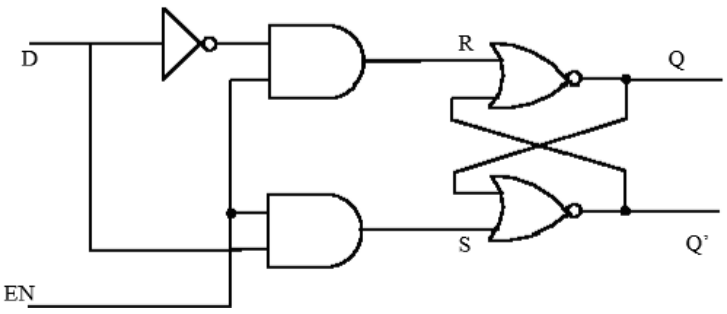
D FLIP-FLOP

An RS flip-flop is rarely used in actual sequential logic because of its undefined outputs for inputs R= S= 1. It can be modified to form a more useful circuit called D flip-flop, where D stands for data. The D flip-flop has only a single data input D as shown in the circuit diagram. That data input is connected to the S input of an RS flip-flop, while the inverse of D is connected to the R input. To allow the flip-flop to be in a holding state, a D-flip flop has a second input called Enable, EN. The Enable-input is AND-ed with the D-input.

When **EN=0**, irrespective of D-input, the R = S = 0 and the **state is held**.
When **EN= 1**, the S input of the RS flip-flop equals the D input and R is the inverse of D. Hence, output **Q follows D**, when EN= 1. When **EN returns to 0**, the most recent input **D is 'remembered'**.

The circuit operation is summarized in the characteristic table for **EN=1**.

Circuit Diagram:



Characteristic Table:

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

PROCEDURE:

1. Assemble the circuits one after another on your breadboard as per the circuit diagrams. Circuit diagrams given here do not show connections to power supply and LEDs assuming that you are alreadyfamiliar with it from your previous lab experience.
2. Connect the ICs properly to power supply (pin 14) and ground (pin 7) following the schematics for ICs given above.
3. Using dip switch and resistors, facilitate all possible combinations of inputs from the power supply. Use the switch also to facilitate pulse input to the circuit.
4. Turn on power to your experimental circuit.
5. For each input combination, note the logic state of the normal and complementary outputs as indicated by the LEDs (ON = 1; OFF = 0), and record the results in a table.
6. Compare your results with the characteristic tables.
7. When you are done, turn off the power to your experimental circuit.

OBSERVATIONS:

Table for RS Flip Flop:

R	S	Q	Q'	Comment
0	0	Q	Q'	Hold state
0	1	1	0	Set
1	0	0	1	Reset
1	1	-	-	Indeterminate

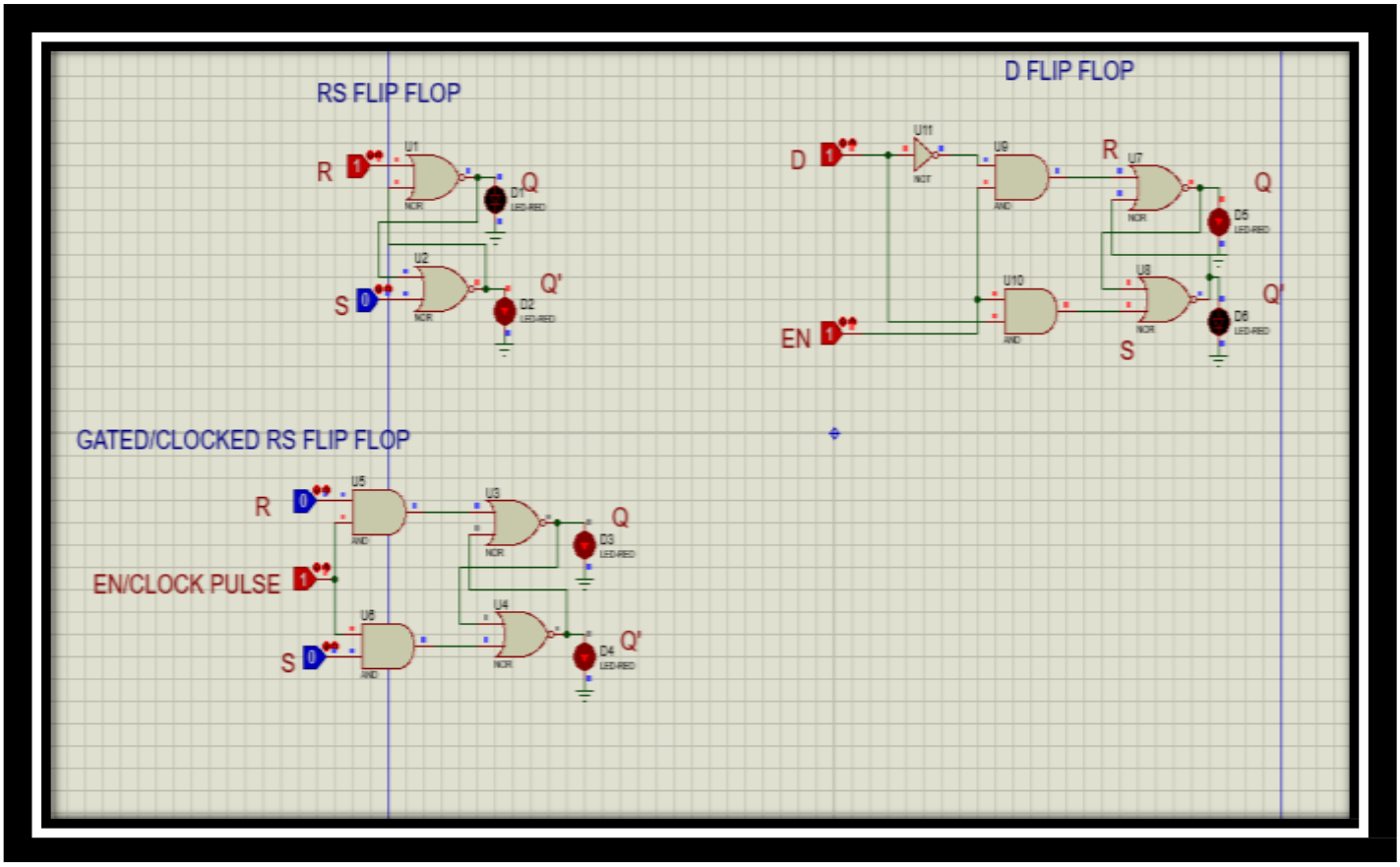
Table for Gated RS Flip Flop:

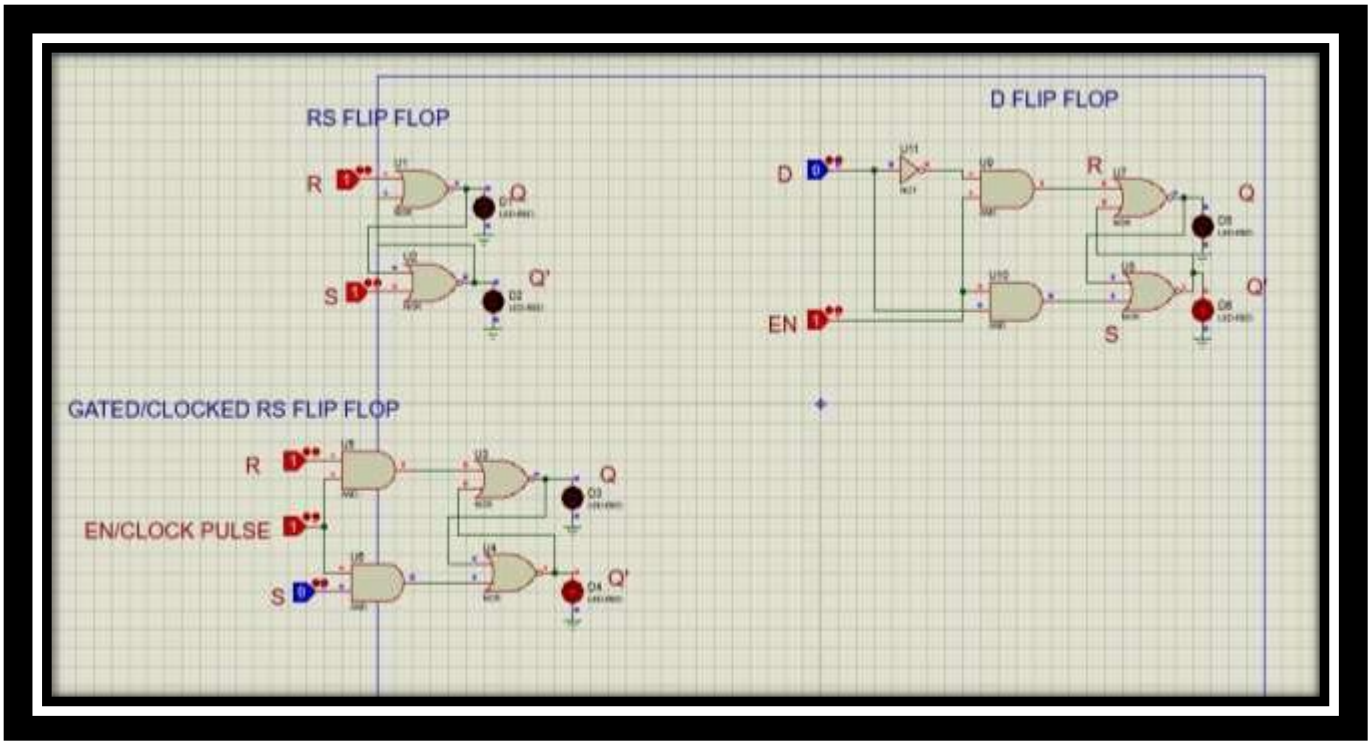
Qn	R	S	Qn+1
0	0	0	hold (0)
0	1	0	0
0	0	1	1
0	1	1	In determinant
1	0	0	hold(1)
1	1	0	0
1	0	1	1
1	1	1	In determinant

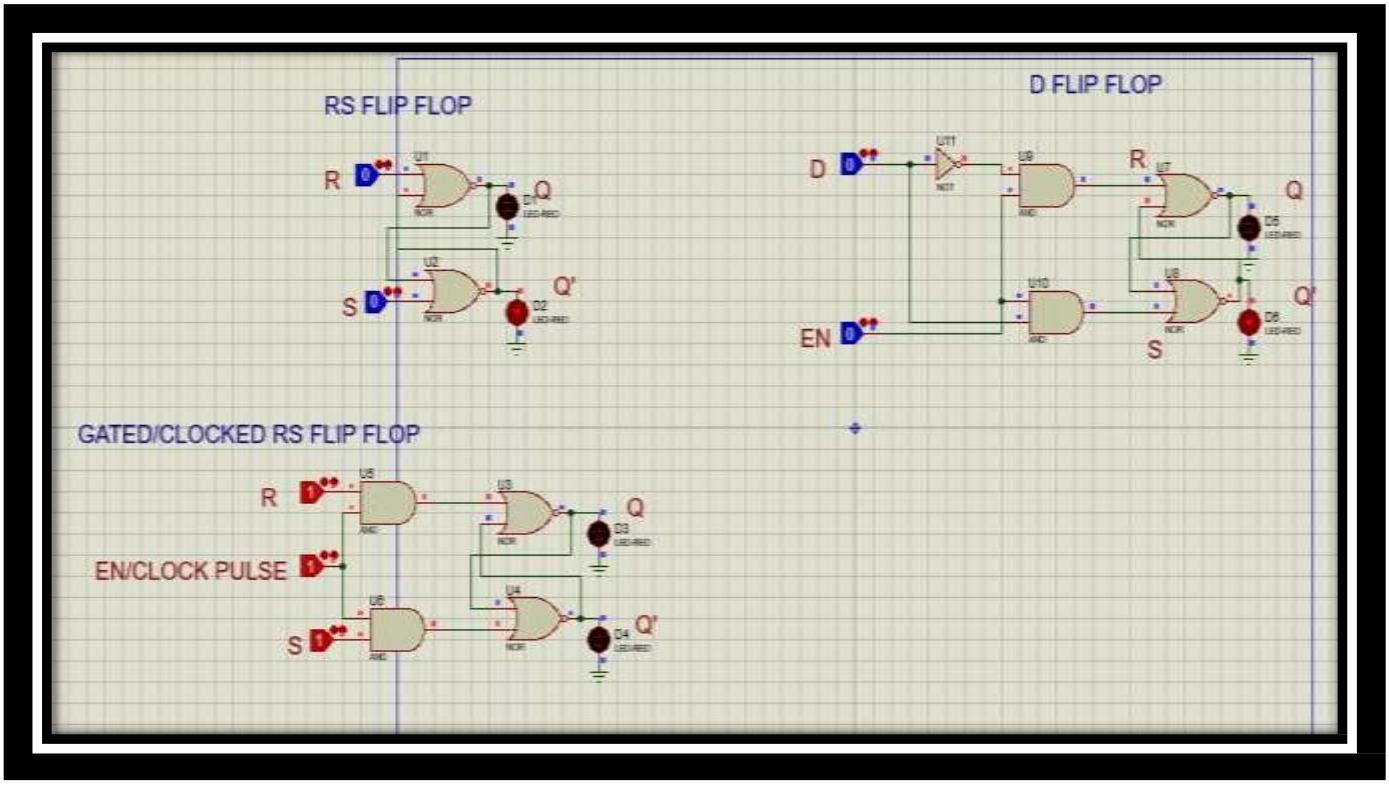
Table for D Flip Flop:

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

RESULT:




RS FLIP FLOP
R
S
Q
Q'
D FLIP FLOP
D
EN
R
S
Q
Q'
GATED/CLOCKED RS FLIP FLOP
R
EN/CLOCK PULSE
S
Q
Q'


RS FLIP FLOP
R
S
Q
Q'
D FLIP FLOP
D
EN
R
S
Q
Q'
GATED/CLOCKED RS FLIP FLOP
R
EN/CLOCK PULSE
S
Q
Q'

LABSESSION11

To analyze and study the operations of the following circuits:

- **JK and Master-Slave JK Flip-Flop**
- **T Flip-Flop**

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION11

OBJECTIVE :

To analyze and study the operations of the following circuits:

- JK and Master-Slave JK Flip-Flop
- T Flip-Flop

THEORY:

So far you have encountered with *combinatorial logic*, i.e. circuits for which the output depends only on the inputs. In many instances it is desirable to have the next output depending on the current output. A simple example is a *counter*, where the next number to be output is determined by the current number stored. Circuits that remember their current output or state are often called *sequential logic* circuits. Clearly, sequential logic requires the ability to store the current state. In other words, *memory* is required by sequential logic circuits, which can be created with boolean gates. If you arrange the gates correctly, they will remember an input value. This simple concept is the basis of RAM (random access memory) in computers, and also makes it possible to create a wide variety of other useful circuits.

Memory relies on a concept called **feedback**. That is, the output of a gate is fed back into the input. The simplest possible feedback circuit using two inverters is shown below (Fig.1):

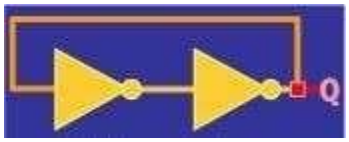
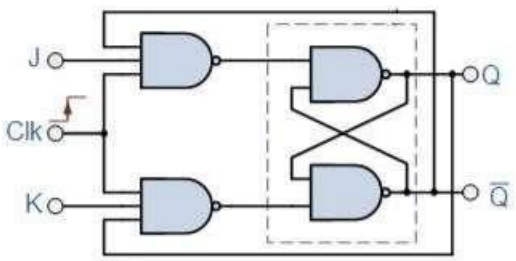


Fig.1: Simplest realization of feedback circuit

If you follow the feedback path, you can see that if Q happens to be 1 (or 0), it will always be 1 (or 0) . Since it's nice to be able to control the circuits we create, this one doesn't have much use -- but it does let you see how feedback works. It turns out that in "real" sequential circuits, you can actually use this sort of simple inverter feedback approach. The memory elements in these circuits are called *flip-flops*. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Binary information can enter a flip-flop in a variety of ways and gives rise to different types of flip-flops.

JK FLIP-FLOP:

The JK flip flop (JK means Jack Kilby, a Texas instrument engineer, who invented it) is the most versatile flip-flop, and the most commonly used flip flop. Like the RS flip-flop, it has two data inputs, J and K, and an EN/clock pulse input (CP). Note that in the following circuit diagram NAND gates are used instead of NOR gates. It has no undefined states, however. The fundamental difference of this device is the feedback paths to the AND gates of the input, i.e. Q is AND-ed with K and CP and $Q^{`}$ with J and CP.



Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1(Toggle,$\overline{Q}_n$)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0(Toggle,$\overline{Q}_n$)

The JK flip-flop has the following characteristics:

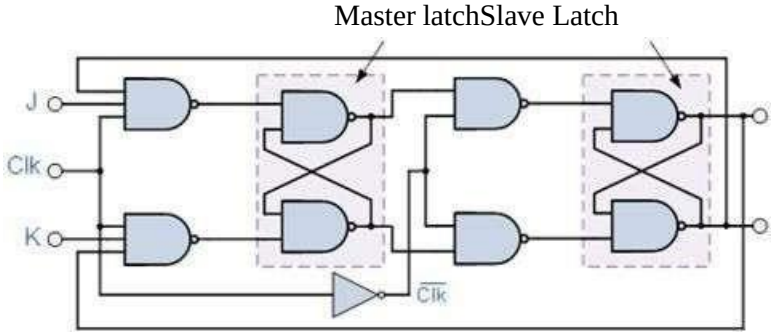
- If one input (J or K) is at logic 0, and the other is at logic 1, then the output is set or reset (by J and K respectively), just like the RS flip-flop.
- If both inputs are 0, then it remains in the same state as it was before the clock pulse occurred; again like the RS flip flop. CP has no effect on the output.
- If both inputs are high, however the flip-flop changes state whenever a clock pulse occurs; i.e., the clock pulse toggles the flip-flop again and again until the CP goes back to 0 as shown in the shaded rows of the characteristic table above. Since this condition is undesirable, it should be eliminated by an improvised form of this flip-flop as discussed in the next section.

MASTER-SLAVE JK FLIP-FLOP:

Although JK flip-flop is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF", so the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved Master-Slave J-K Flip-Flop was developed. This eliminates all the timing problems by using two SR flip-flops connected together in series, one for the "Master" circuit, which triggers on the leading edge of the clock pulse and the other, the "Slave" circuit, which triggers on the falling edge of the clock pulse.

The master-slave JK flip flop consists of two flip flops arranged so that when the clock pulse enables the first, or master, it disables the second, or slave. When the clock changes state again (i.e., on its falling edge) the output of the master latch is transferred to the slave latch. Again, toggling is accomplished by the connection of the output with the input AND gates.

CIRCUIT DIAGRAM:



CHARACTERISTIC TABLE:

CP	J	K	Q_m	$\bar{Q}_m$	Q_n	$\bar{Q}_n$
0→1	0	0	Hold		Hold	
1→0	0	0	Hold		Hold	
0→1	0	1	0	1	Hold	
1→0	0	1	Hold		0	1
0→1	1	0	1	0	Hold	
1→0	1	0	Hold		1	0
0→1	1	1	Toggle		Hold	
1→0	1	1	Hold		Toggle	

T FLIP-FLOP:

The T flip-flop is a single input version of the JK flip-flop. The T flip-flop is obtained from the JK type if both inputs are tied together.

CIRCUIT DIAGRAM:

Same as Master-Slave JK flip-flop with J=K=1The toggle, or T, flip-flop is a bistable device, where the output of the T flip-flop "toggles" with each clock pulse .Till CP=0, the output is in hold state (three input AND gate principle).When CP=1, for T=0, previous output is memorized by the circuit. When T=1 along with the clock pulse, the output toggles from the previous value as given in the characteristic table below.

CHARACTERISTIC TABLE:

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

PROCEDURE:

1. Assemble the circuits one after another on your breadboard as per the circuit diagrams. Circuit diagrams given here do not show connections to power supply and LEDs assuming that you are already familiar with it from your previous lab experience.
2. Connect the ICs properly to power supply (pin 14) and ground (pin 7) following the schematics for ICs given above.

3. Using dip switch and resistors, facilitate all possible combinations of inputs from the power supply. Use the switch also to facilitate pulse input to the circuit.
4. Turn on power to your experimental circuit.
5. For each input combination, note the logic state of the normal and complementary outputs as indicated by the LEDs (ON = 1; OFF = 0), and record the results in atable.
6. Compare your results with the characteristic tables.
7. When you are done, turn off the power to your experimental circuit.

OBSERVATIONS:

Table for JK FF:

Qn	j	k	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1(toggle)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	toggle (0)

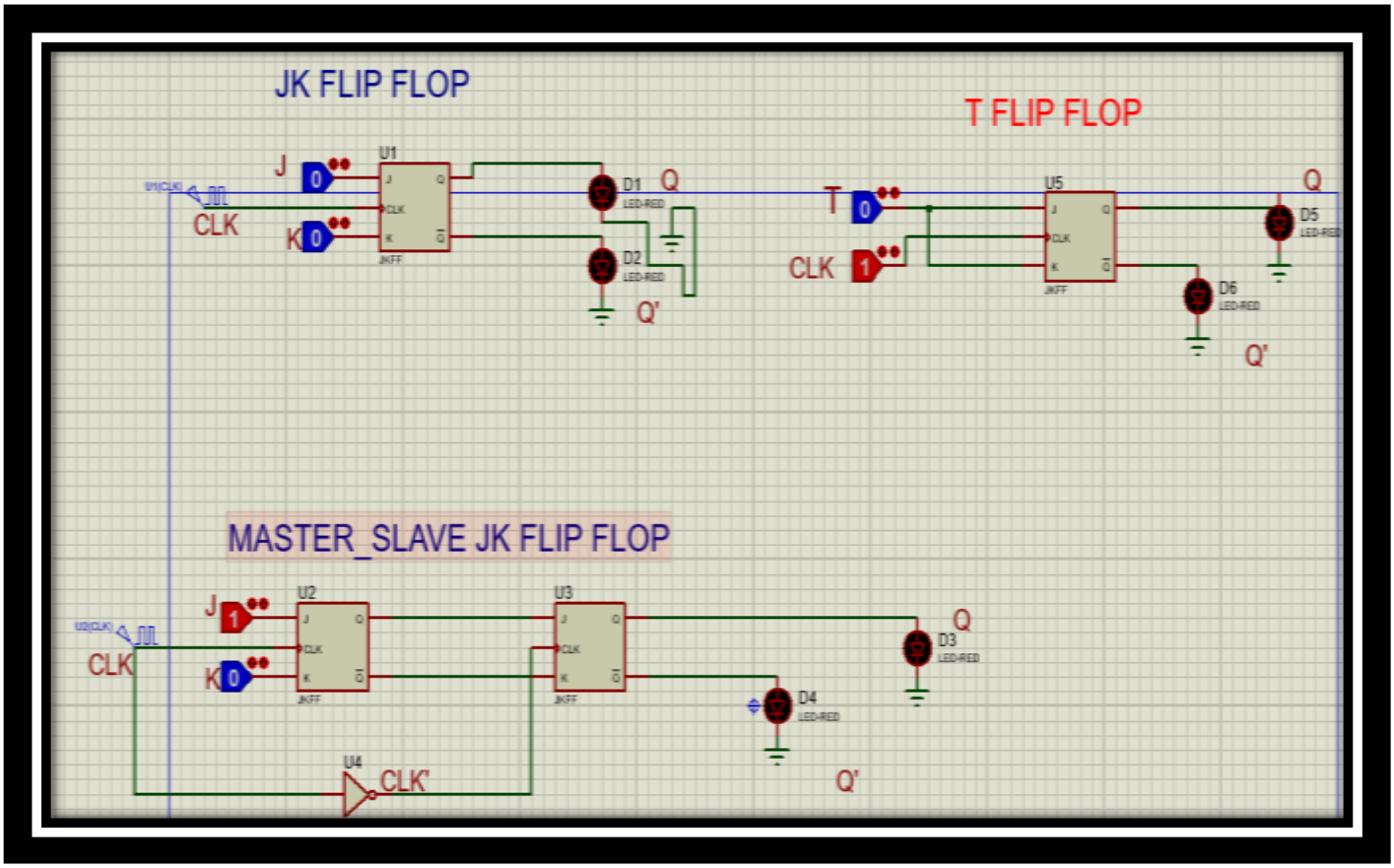
Table for Master-Slave JK FF:

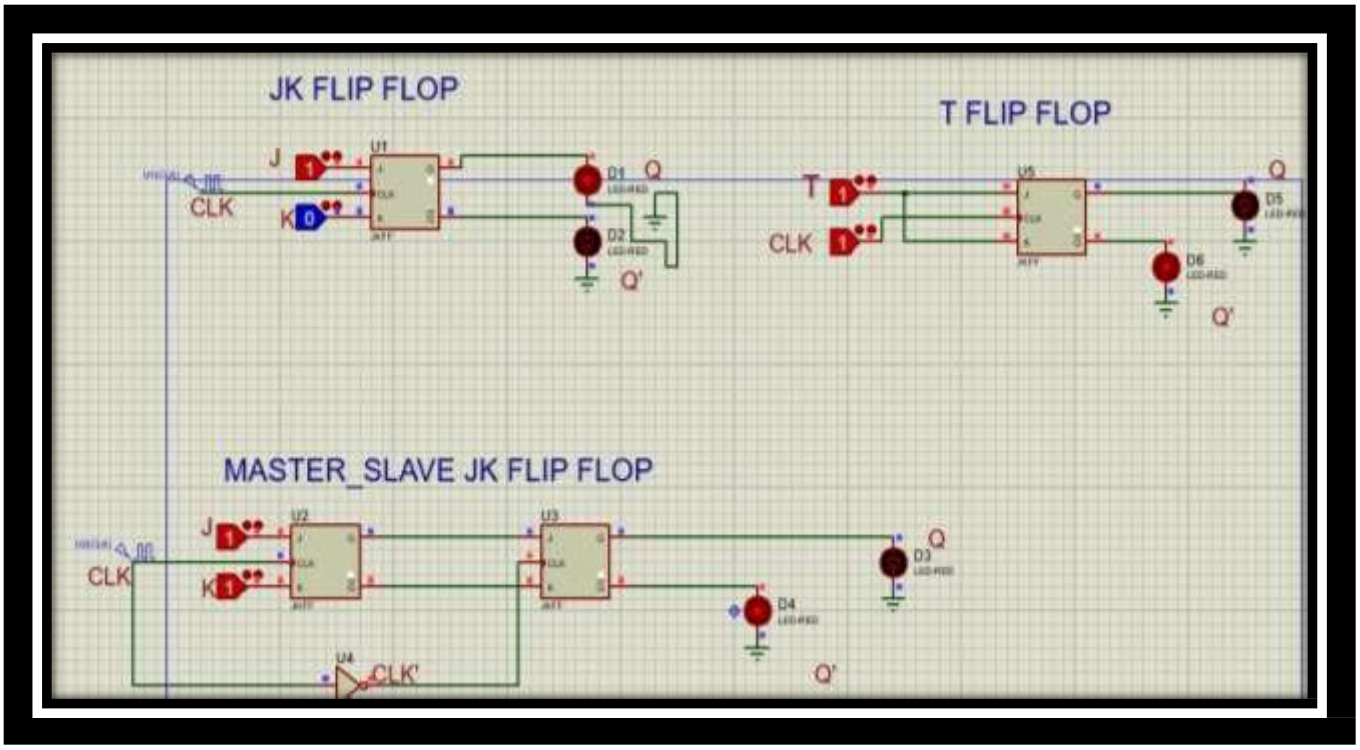
Trigger	Inputs		Output						Inference
			Present State		Intermediate		Next State		
CLK	J	K	Q	$\bar{Q}$	M_1	M_2	Q	$\bar{Q}$	
↑	0	0	0	1	0	1	Latched		No Change
↓			0	1	Latched		0	1	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	0	1	0	1	0	1	Latched		Reset
↓			0	1	Latched		0	1	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	
↑	1	0	0	1	1	0	Latched		Set
↓			0	1	Latched		1	0	
↑			1	0	1	0	Latched		
↓			1	0	Latched		1	0	
↑	1	1	0	1	1	0	Latched		Toggles
↓			0	1	Latched		1	0	
↑			1	0	0	1	Latched		
↓			1	0	Latched		0	1	

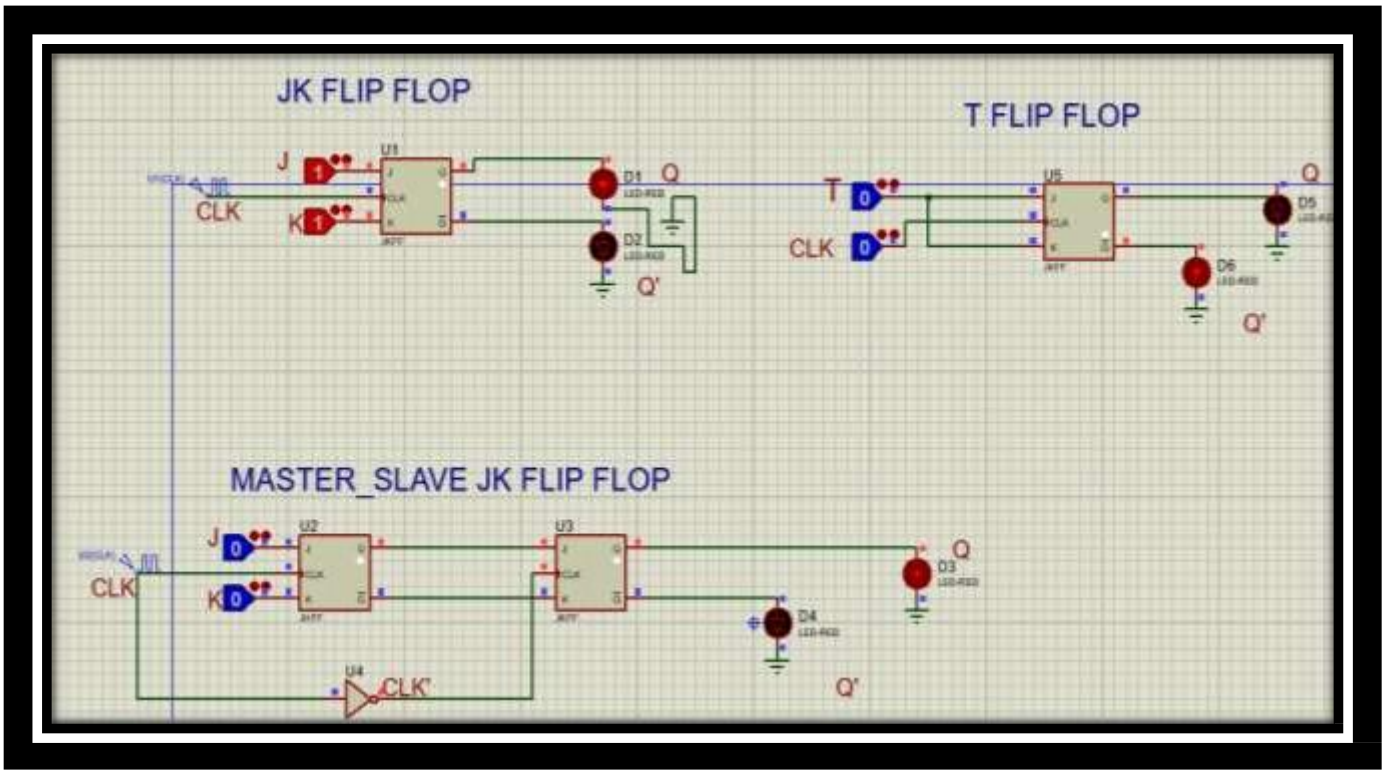
Table For T FF:

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

RESULT:




JK FLIP FLOP
T FLIP FLOP
MASTER_SLAVE JK FLIP FLOP
J
K
CLK
T
Q
Q'
CLK'
U1
U2
U3
U4
U5
D1
D2
D3
D4
D5
D6


JK FLIP FLOP
T FLIP FLOP
MASTER_SLAVE JK FLIP FLOP
J
K
CLK
T
Q
Q'
CLK'
U1
U2
U3
U4
U5
D1
D2
D3
D4
D5
D6

LABSESSION12

Design and implement eight bit adder on FPGA.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

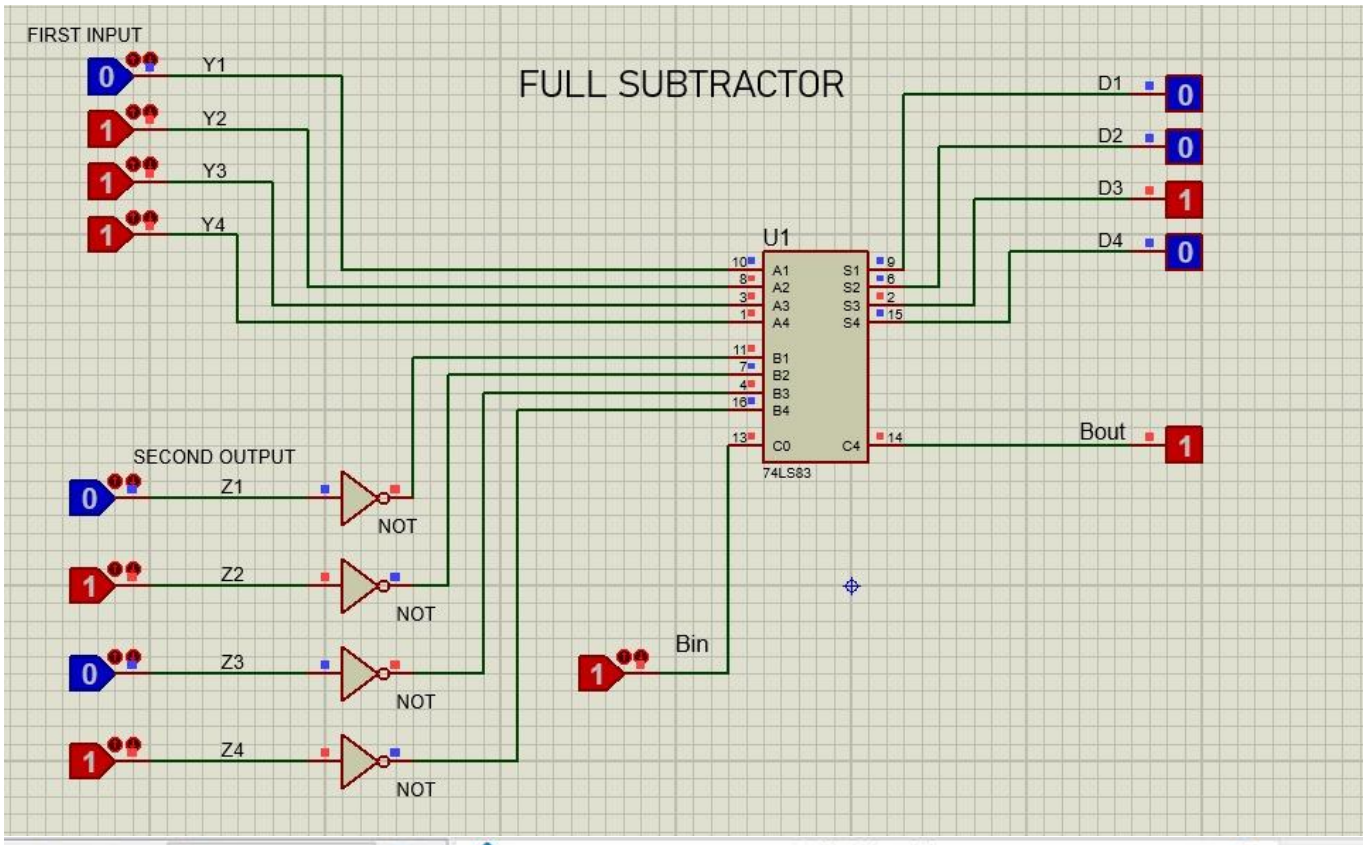
Remarks (If Any):

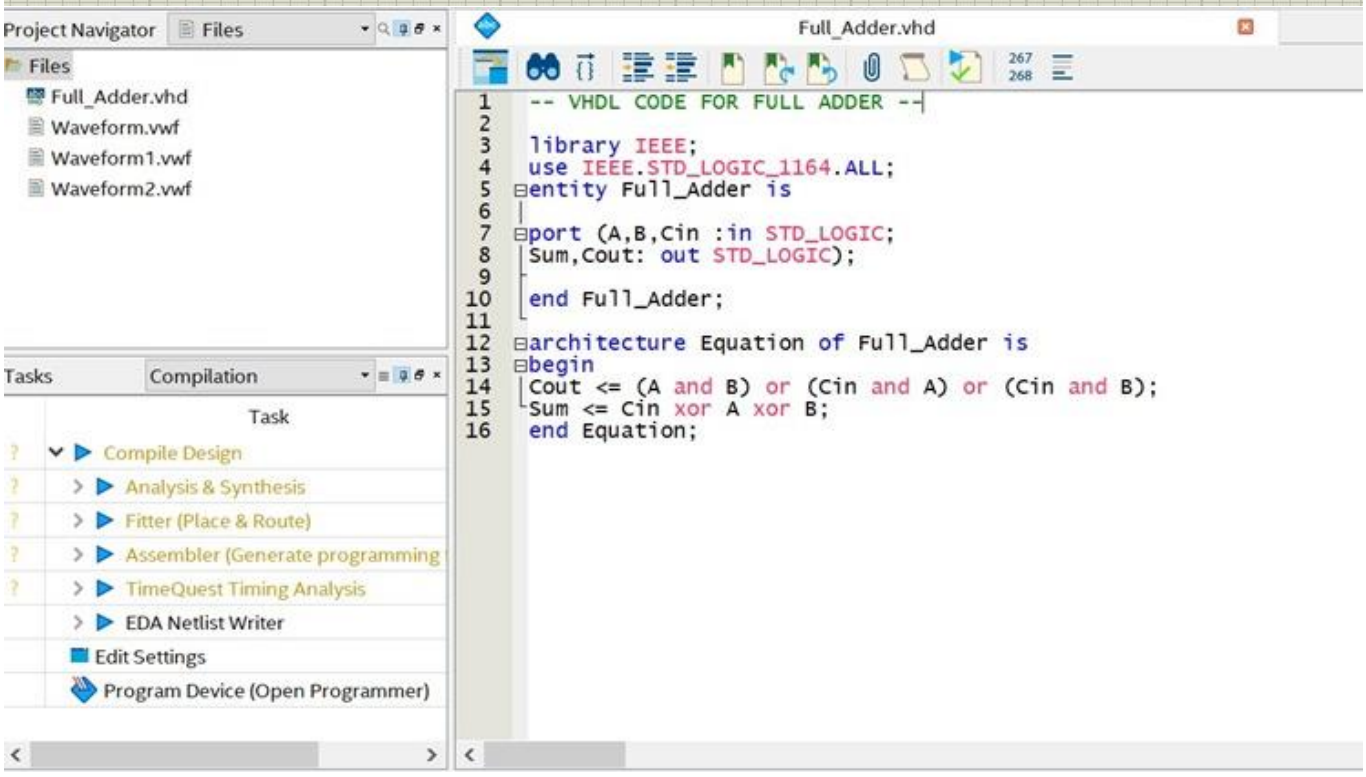
Instructor Name: Muneeb Shaikh

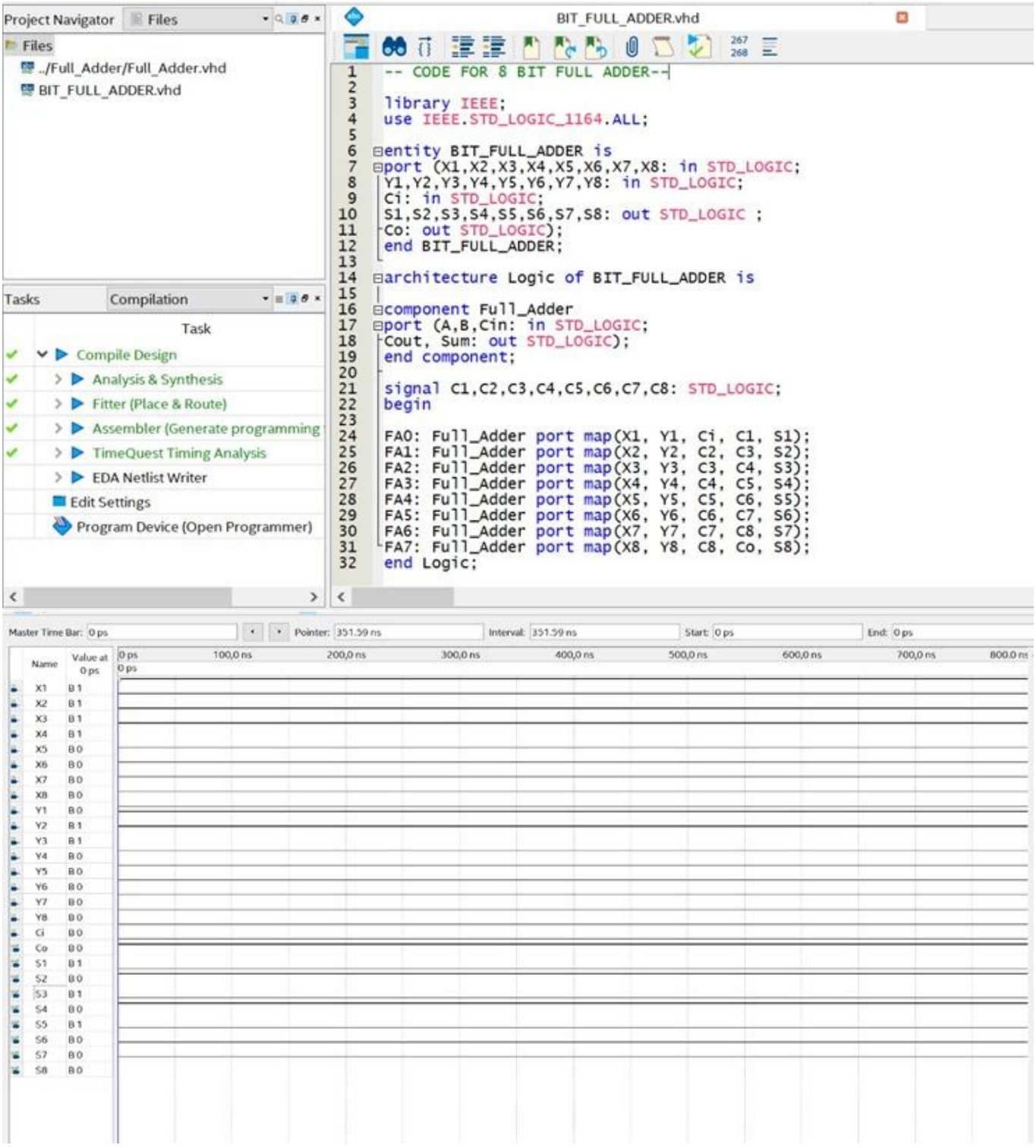
Instructor Signature: Date:

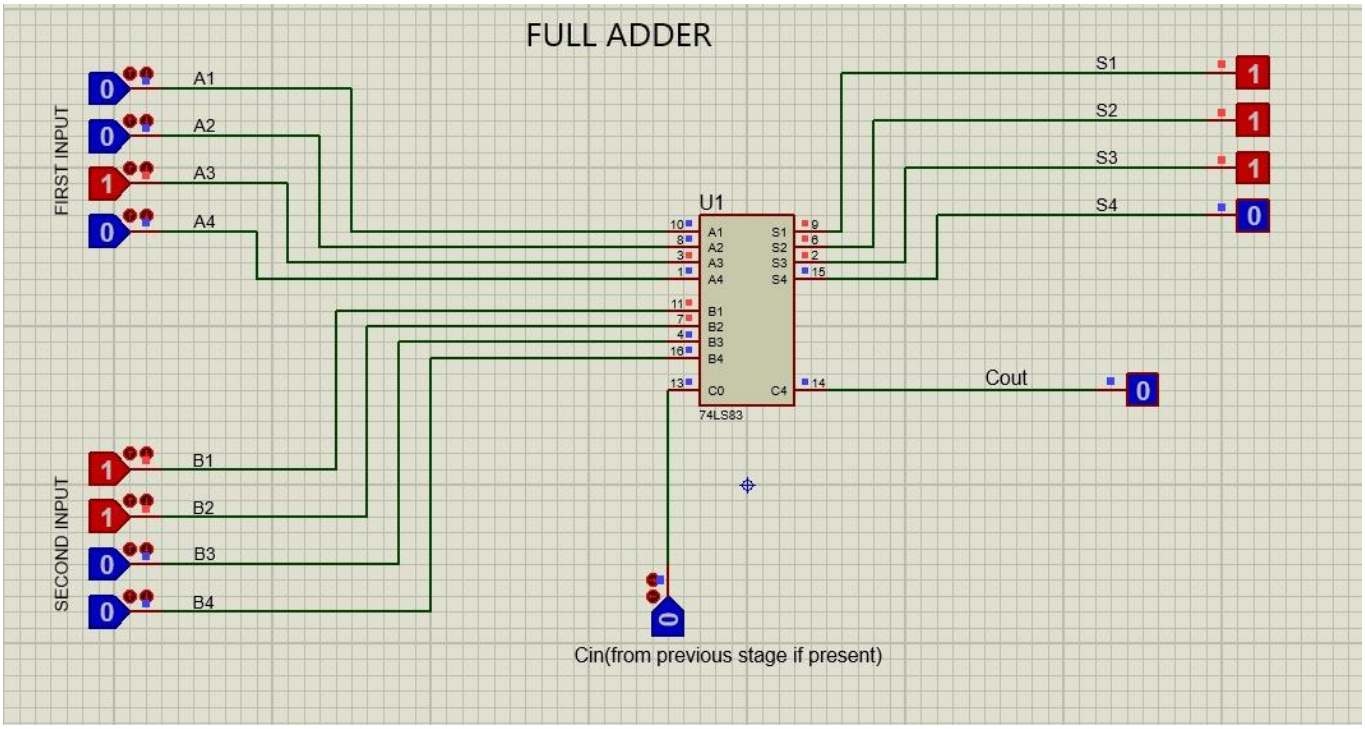
OBJECTIVE :

Design and implement 8 bit adder on FPGA.






Project Navigator
Files
Files
../Full_Adder/Full_Adder.vhd
BIT_FULL_ADDER.vhd
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
BIT_FULL_ADDER.vhd
-- CODE FOR 8 BIT FULL ADDER--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity BIT_FULL_ADDER is
port (X1,X2,X3,X4,X5,X6,X7,X8: in STD_LOGIC;
Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8: in STD_LOGIC;
Ci: in STD_LOGIC;
S1,S2,S3,S4,S5,S6,S7,S8: out STD_LOGIC ;
Co: out STD_LOGIC);
end BIT_FULL_ADDER;
architecture Logic of BIT_FULL_ADDER is
component Full_Adder
port (A,B,Cin: in STD_LOGIC;
Cout, Sum: out STD_LOGIC);
end component;
signal C1,C2,C3,C4,C5,C6,C7,C8: STD_LOGIC;
begin
FA0: Full_Adder port map(X1, Y1, Ci, C1, S1);
FA1: Full_Adder port map(X2, Y2, C2, C3, S2);
FA2: Full_Adder port map(X3, Y3, C3, C4, S3);
FA3: Full_Adder port map(X4, Y4, C4, C5, S4);
FA4: Full_Adder port map(X5, Y5, C5, C6, S5);
FA5: Full_Adder port map(X6, Y6, C6, C7, S6);
FA6: Full_Adder port map(X7, Y7, C7, C8, S7);
FA7: Full_Adder port map(X8, Y8, C8, Co, S8);
end Logic;
Master Time Bar:
Pointer:
Interval:
Start:
End:
Name
Value at 0 ps
100,0 ns
200,0 ns
300,0 ns
400,0 ns
500,0 ns
600,0 ns
700,0 ns
800,0 ns
X1
X2
X3
X4
X5
X6
X7
X8
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y8
Ci
Co
S1
S2
S3
S4
S5
S6
S7
S8


FULL ADDER
FIRST INPUT
A1
A2
A3
A4
SECOND INPUT
B1
B2
B3
B4
U1
74LS83
S1
S2
S3
S4
Cout
Cin(from previous stage if present)

LABSESSION13

Design and implement BCD to Seven segment decoder on FPGA.

Student Name: ALI MUSAWIR

Roll Number: EL-19034 Batch: 2019

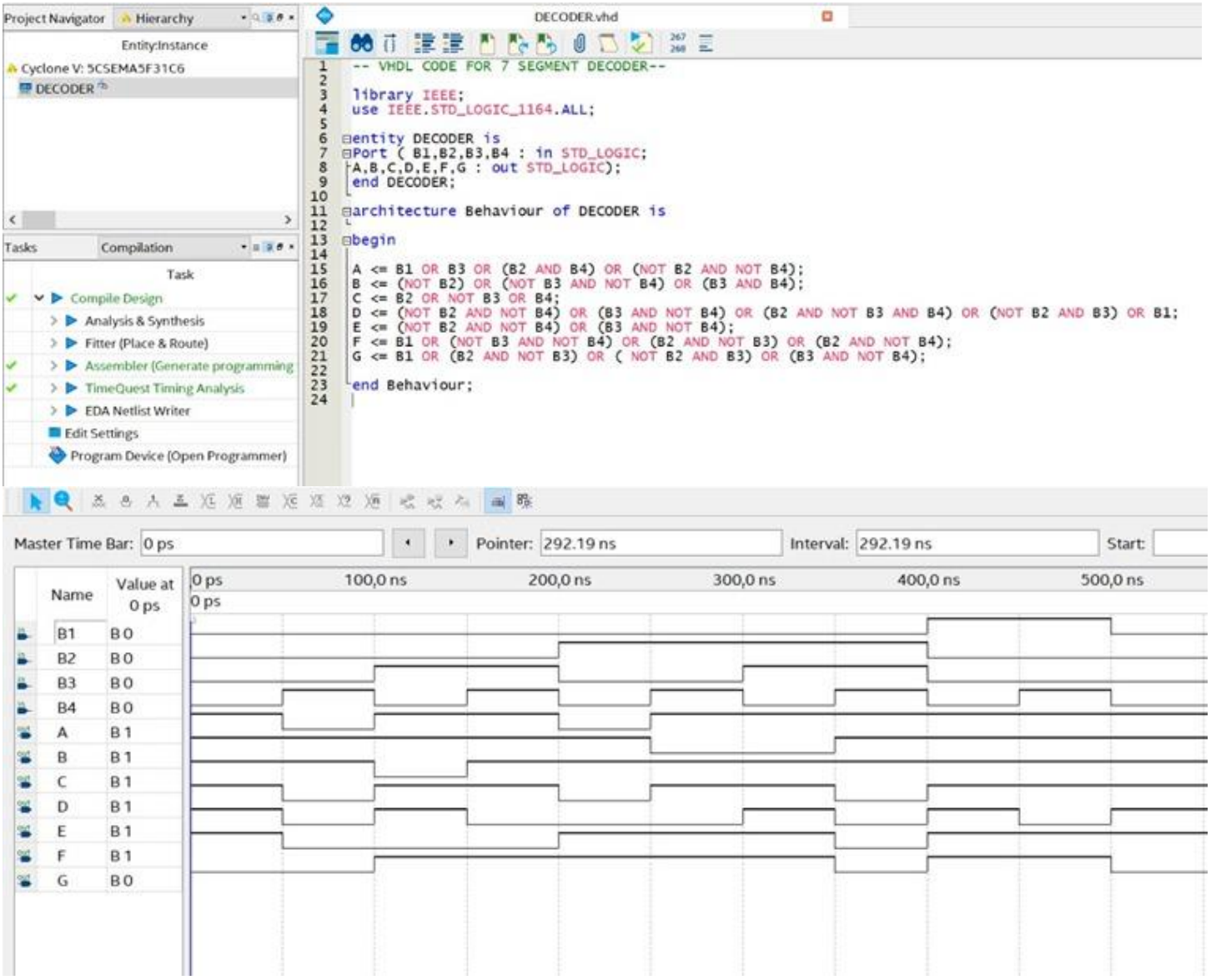
Semester: 4th(Spring) Year: 2020

Total Marks	
Marks Obtained	

Remarks (If Any):

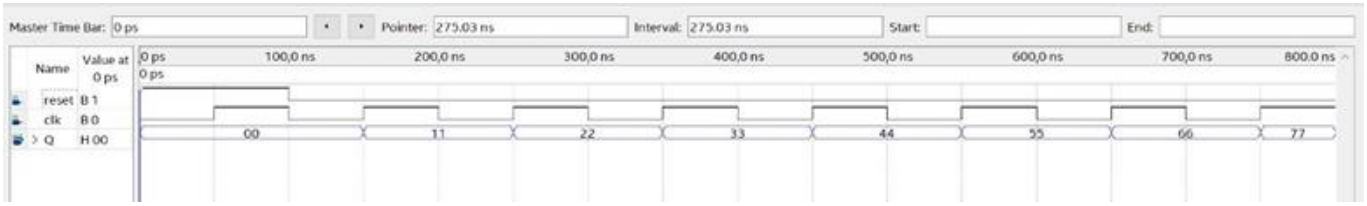
Instructor Name: Muneeb Shaikh

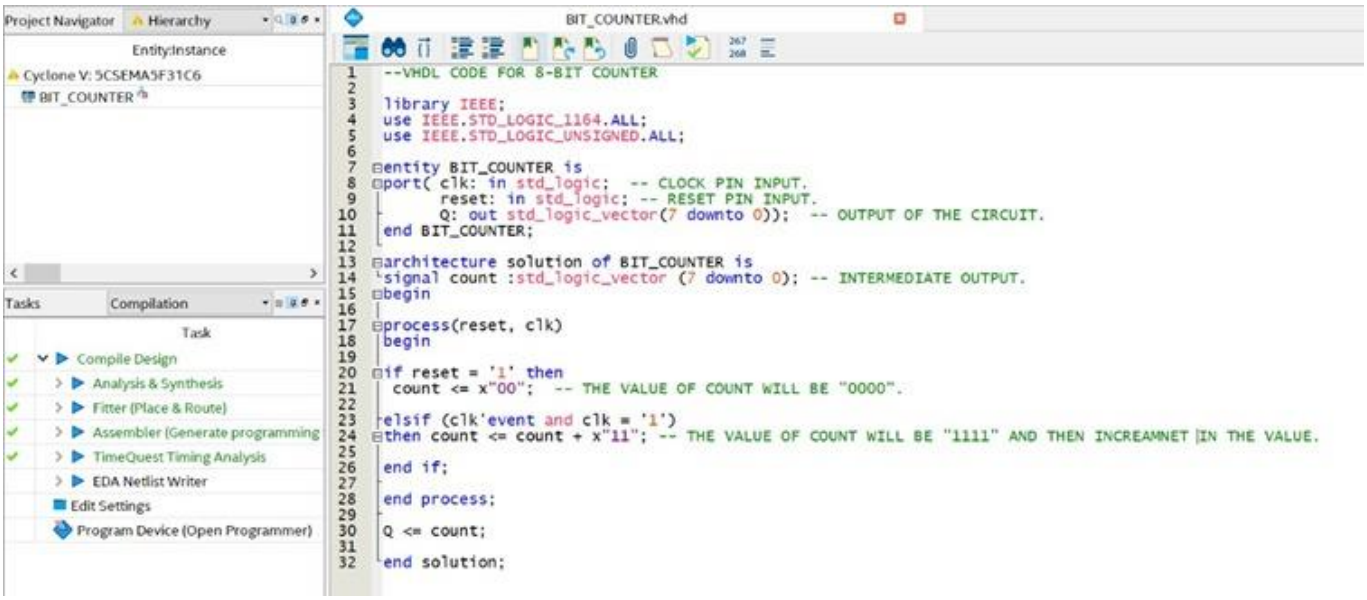
Instructor Signature: Date:


Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CSEMA5F31C6
DECODER
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
DECODER.vhd
-- VHDL CODE FOR 7 SEGMENT DECODER--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity DECODER is
Port (B1,B2,B3,B4 : in STD_LOGIC;
A,B,C,D,E,F,G : out STD_LOGIC);
end DECODER;
architecture Behaviour of DECODER is
begin
A <= B1 OR B3 OR (B2 AND B4) OR (NOT B2 AND NOT B4);
B <= (NOT B2) OR (NOT B3 AND NOT B4) OR (B3 AND B4);
C <= B2 OR NOT B3 OR B4;
D <= (NOT B2 AND NOT B4) OR (B3 AND NOT B4) OR (B2 AND NOT B3 AND B4) OR (NOT B2 AND B3) OR B1;
E <= (NOT B2 AND NOT B4) OR (B3 AND NOT B4);
F <= B1 OR (NOT B3 AND NOT B4) OR (B2 AND NOT B3) OR (B2 AND NOT B4);
G <= B1 OR (B2 AND NOT B3) OR (NOT B2 AND B3) OR (B3 AND NOT B4);
end Behaviour;
Master Time Bar: 0 ps
Pointer: 292.19 ns
Interval: 292.19 ns
Start:
Name
Value at 0 ps
0 ps
100,0 ns
200,0 ns
300,0 ns
400,0 ns
500,0 ns
B1 B 0
B2 B 0
B3 B 0
B4 B 0
A B 1
B B 1
C B 1
D B 1
E B 1
F B 1
G B 0

OBJECTIVE :

Design and implement BCD to seven segment encoder on FPGA.





LABSESSION14

Design and implement 8 bit counter with synchronous reset and load functionality on FPGA.

Student Name: ALI MUSAWIR

Rol l Number: EL-19034 Batch: 2019

Semester: 4th(Spring) Year: 2021

Total Marks	
Marks Obtained	

Remarks (If Any):

Instructor Name: Muneeb Shaikh

Instructor Signature: Date:

LABSESSION14

OBJECTIVE:

Design and implement 8 bit counter with synchronous load and reset functionality on FPGA.