Measurements and Testing (1)

التجارب المعملية

لمادة القياسات والاختبارات (١)

للفرقة الثانية قسم كهرباء

الفصل الدراسي الثاني

-٢٠١٨-

اعداد /

قسم هندسة الحاسبات والمنظومات

كلية الهندسة

جامعة الزقازيق

Table of Contents

LAB (1): BCD COUNTER

Objectives:

1. Introduce the student to Proteus which is a design software developed by Labcenter Electronics for electronic circuit simulation, schematic capture and PCB design.
2. Teach the student how to read datasheets and connect different hardware component.

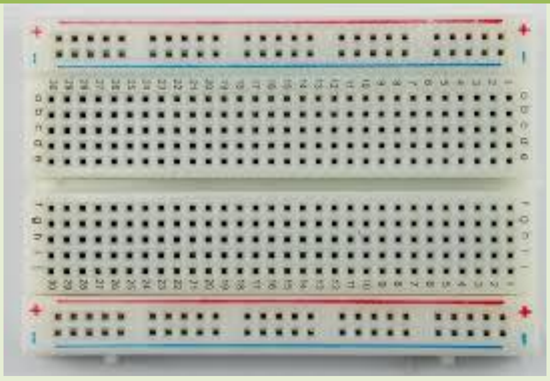
Introduction:

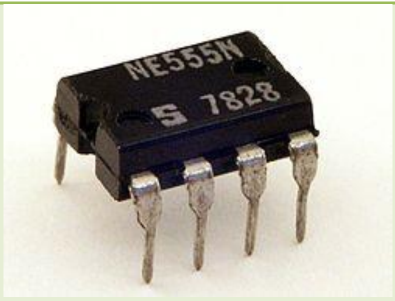
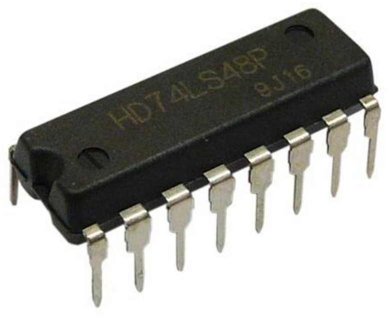
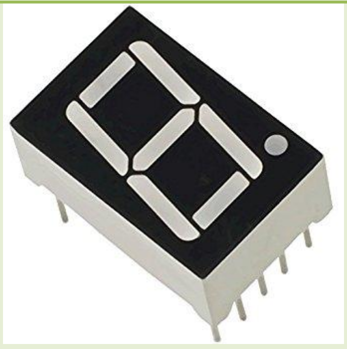
In this lab, the student should be able to wire a counter circuit and display the count on 7-segments and in binary representation using LEDs.

Software requirements:

Proteus

Hardware requirements:

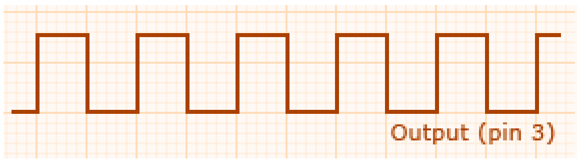
#	Component	Purpose	Shape
1	Bread board	Connect the components and test them	
2	BCD Counter (74LS90)	Generate the binary coded counted output (0-9)	

3	Square Wave Generator (555)	Generate a clock signal	
4	BCD to Seven Segment Decoder Common Cathode (7448)	Decode the counter output to be displayed on the 7-segment	
5	Common Cathode 7-segment	Display the result	
6	LEDs	Display the result	
7	Wires	Connect the components	
8	Resistors	reduce current flow	

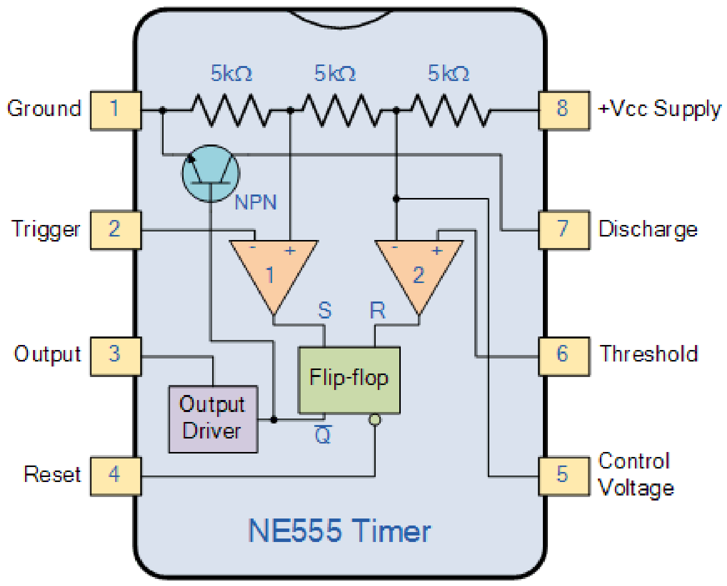
9	Capacitors	Charge and discharge comprising an oscillation used as the clock signal.	

Procedure

1- Generating the clock signal

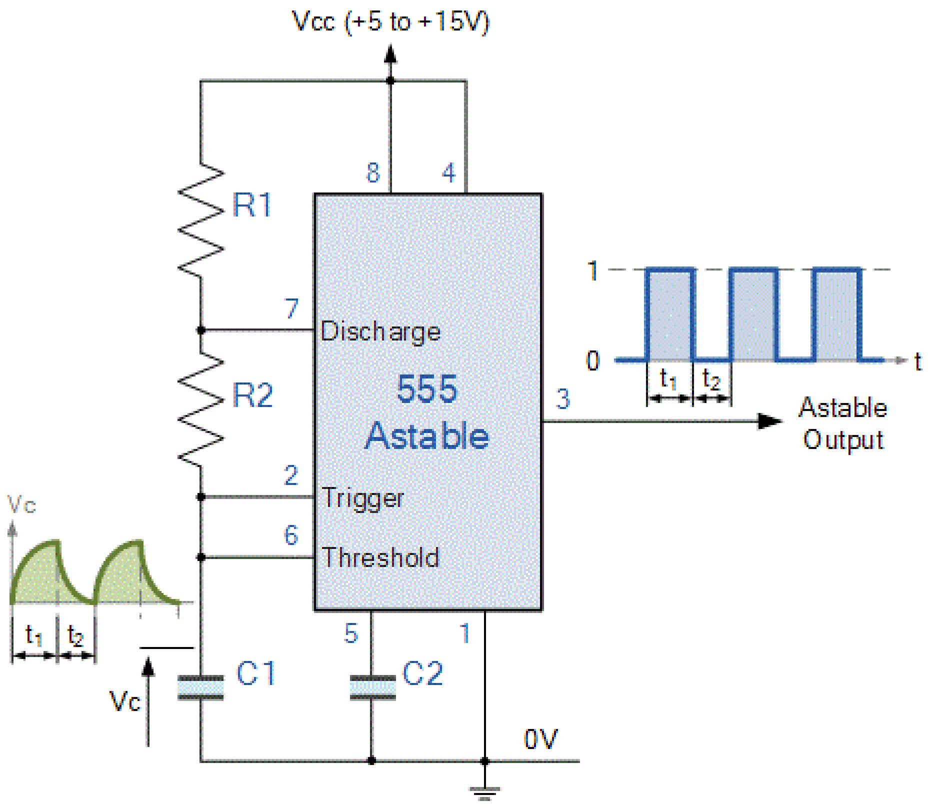


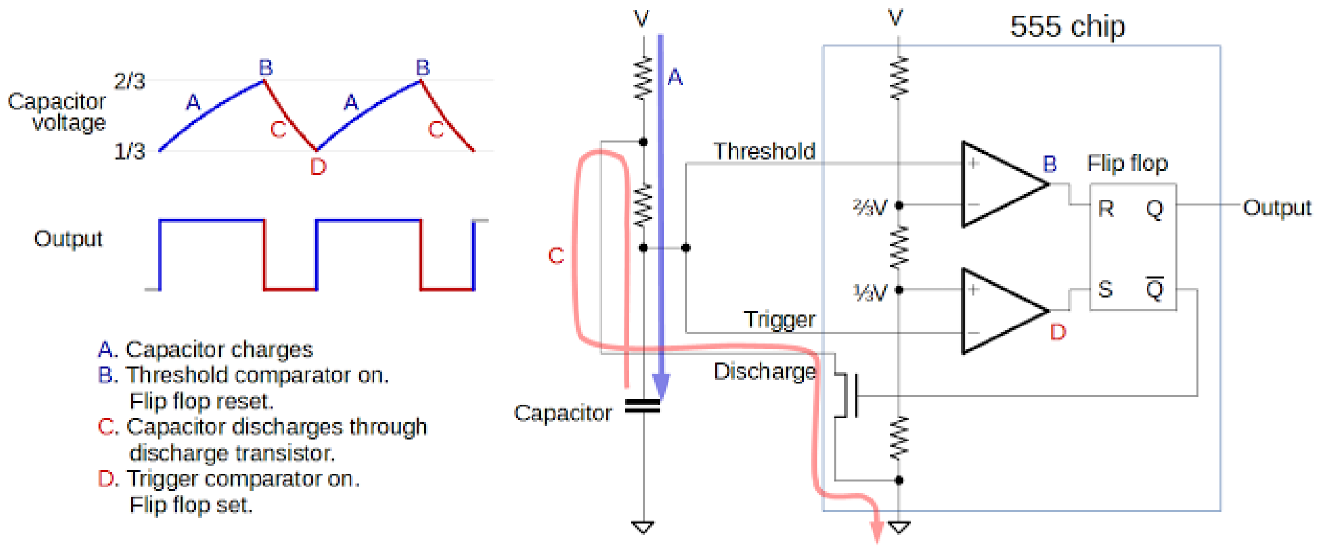
The clock signal is used to drive the counter and with each edge the counter adds up. In order to generate that clock, the 555 IC is used.



Pin 1. – Ground	The ground pin connects the 555 timer to the negative (0v) supply rail.
Pin 2. – Trigger	The negative input to comparator No 1. A negative pulse on this pin “sets” the internal Flip-flop when the voltage drops below 1/3Vcc causing the output to switch from a “LOW” to a “HIGH” state.
Pin 3. – Output	The output pin can drive any TTL circuit and is capable of sourcing or sinking up to 200mA of current at an output voltage equal to approximately Vcc – 1.5V so small speakers, LEDs or motors can be connected directly to the output.
Pin 4. – Reset	This pin is used to “reset” the internal Flip-flop controlling the state of the output, pin 3. This is an active-low input and is generally connected to a logic “1” level when not used to prevent any unwanted resetting of the output.
Pin 5. – Control Voltage	This pin controls the timing of the 555 by overriding the 2/3Vcc level of the voltage divider network. By applying a voltage to this pin the width of the output signal can be varied independently of the RC timing network. When not used it is connected to ground via a 10nF capacitor to eliminate any noise.
Pin 6. – Threshold	The positive input to comparator No 2. This pin is used to reset the Flip-flop when the voltage applied to it exceeds 2/3Vcc causing the output to switch from “HIGH” to “LOW”

The 555 IC has three main operating modes, Monostable, Astable, and Bistable. Clock generation is the Astable mode as it is not stable in any state: the output is continually changing between 'low' and 'high' and it is connected as follows.





Capacitor voltage
2/3
1/3
A
B
C
D
Output
A. Capacitor charges
B. Threshold comparator on. Flip flop reset.
C. Capacitor discharges through discharge transistor.
D. Trigger comparator on. Flip flop set.
V
555 chip
Threshold
Trigger
Discharge
Capacitor
⅔V
⅓V
Flip flop
R
Q
S
$\overline{Q}$
Output

The 555 IC equations:

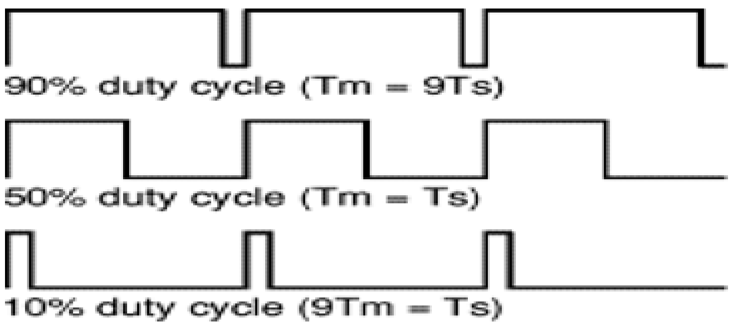
$$t1 = 0.693(R1 + R2)C1$$

$$t2 = 0.693\ R2\ C1$$

$$T = t1 + t2 = 0.693\ (R1 + 2R2\ )C1$$

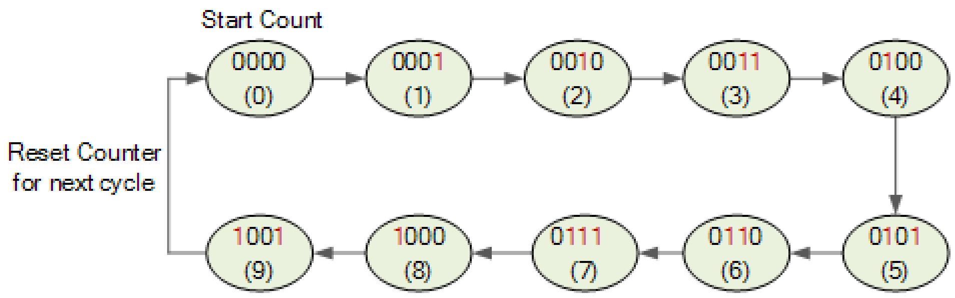
$$f = \frac{1}{T} = \frac{1.44}{(R1 + 2R2)C1}$$

$$Duty\ Cycle = \frac{Ton}{T\ off + T\ on} = \frac{R1 + R2}{R1 + 2R2}\ \%$$

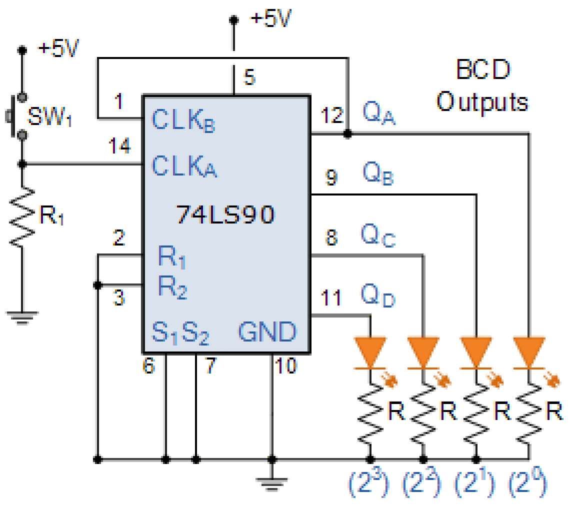


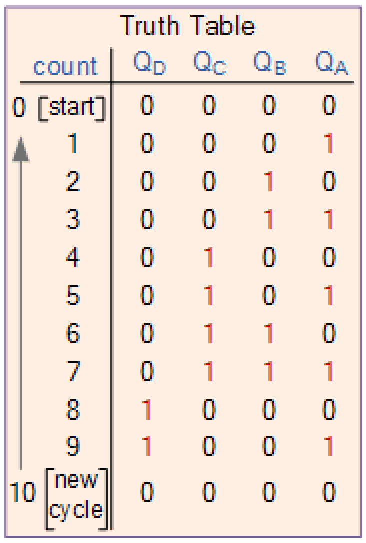
Counting using the clock

Digital counters count upwards from zero to some pre-determined count value on the application of a clock signal. Once the count value is reached, resetting them returns the counter back to zero to start again.



7490 Connection



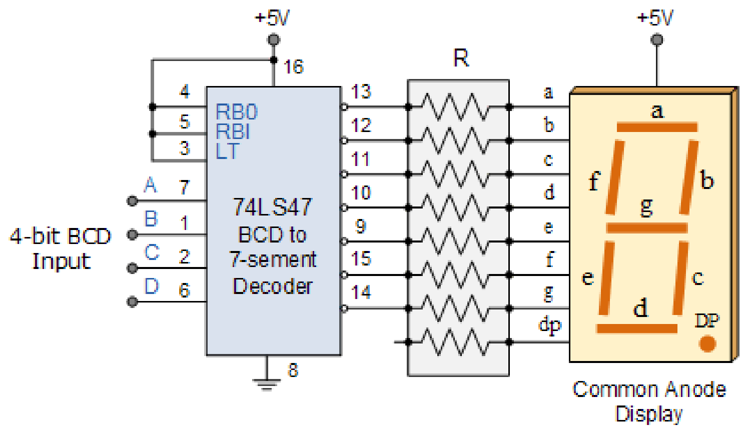


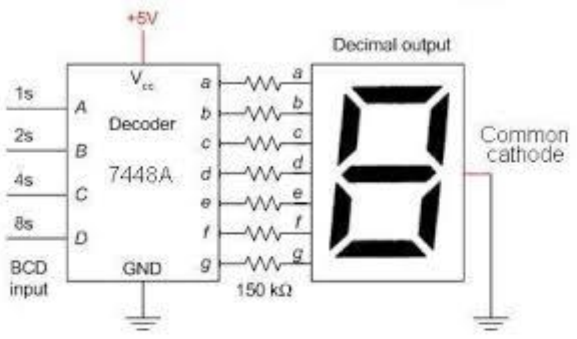
Truth Table

count	Q_D	Q_C	Q_B	Q_A
0 [start]	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 [new cycle]	0	0	0	0

2- Displaying the count on 7–segment

In order to display the count on 7-segmment, a BCD to 7-segment must be used. In particular, the 7448 decoder is used and its connection is as follows.





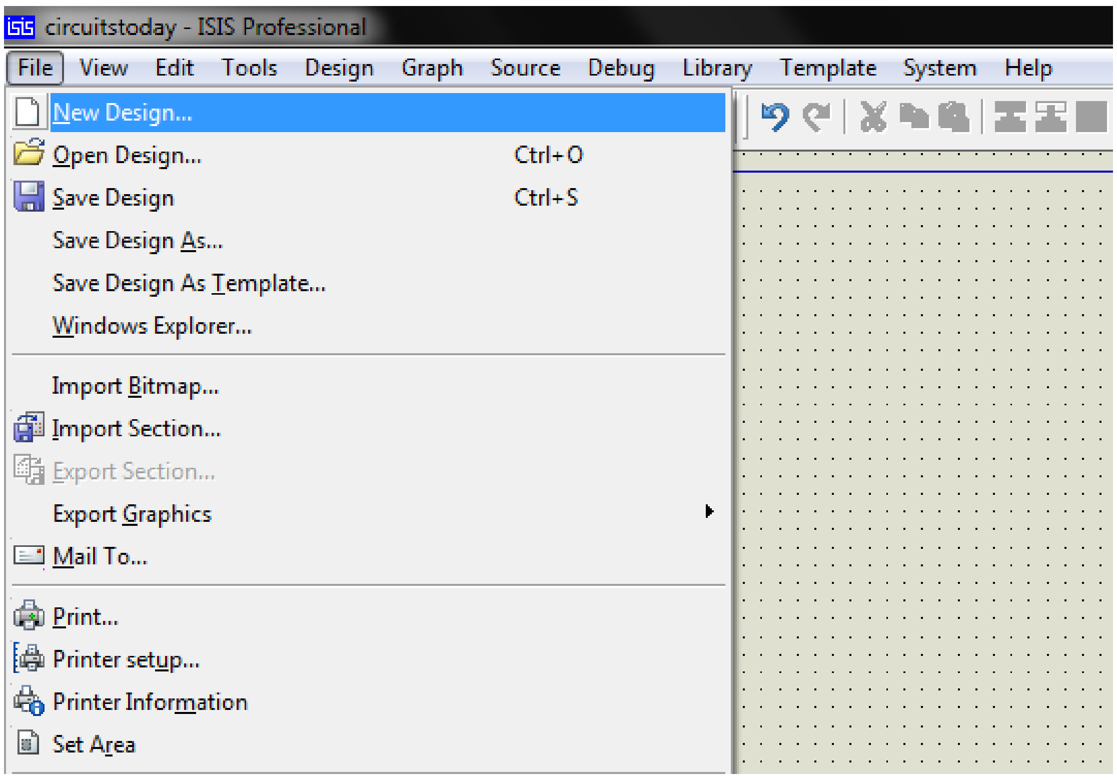
Software simulation

Proteus is a simulation and design software tool developed by Labcenter Electronics for Electrical and Electronic circuit design.

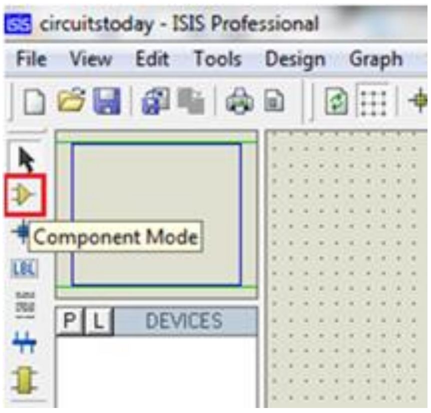
- ISIS is the software used to draw schematics and simulate the circuits in real time. The simulation allows human access during run time, thus providing real time simulation.
- ARES is used for PCB designing. It has the feature of viewing output in 3D view of the designed PCB along with components.

Starting a new project in Proteus

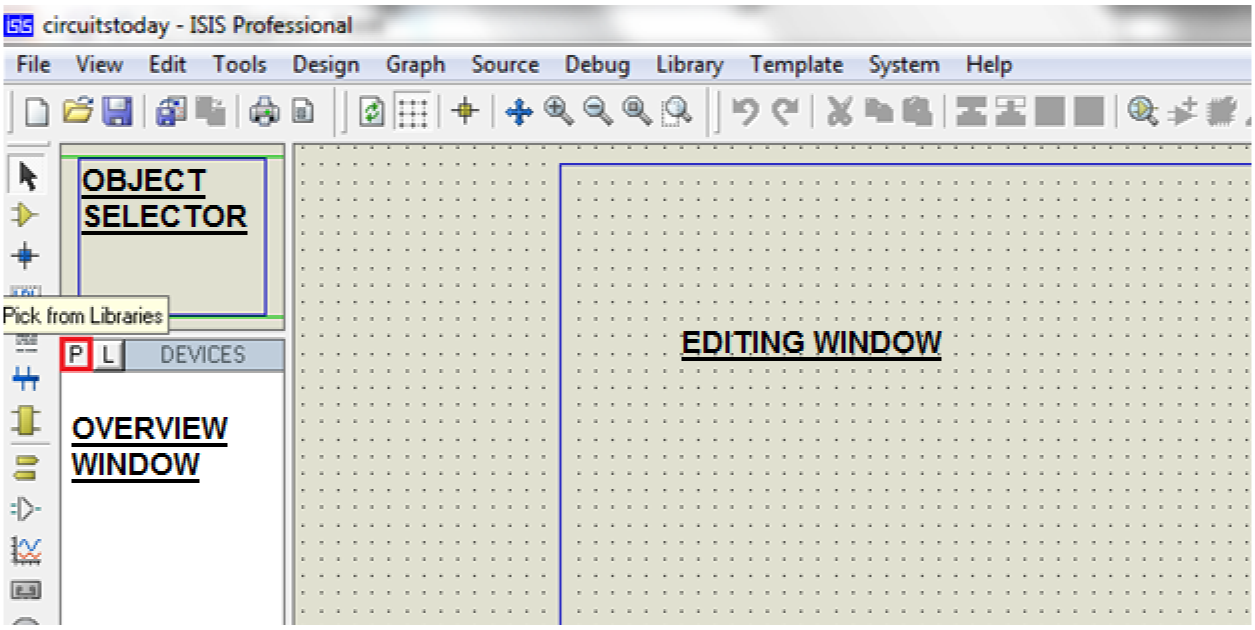
1- Open ISIS software and select New design in File menu



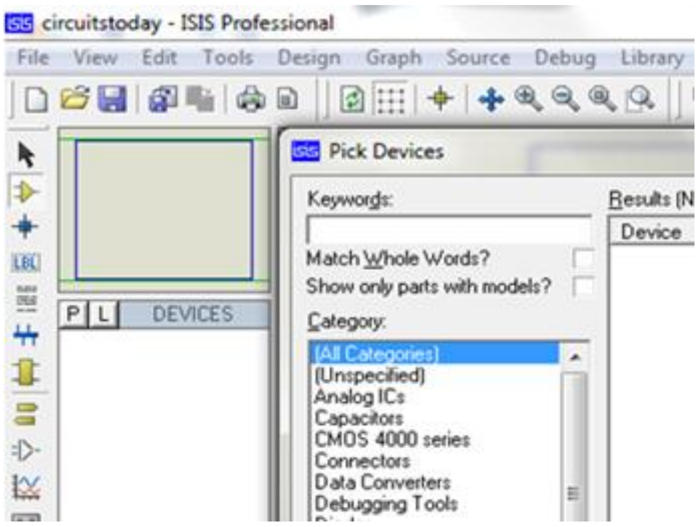
2- A dialogue box appears to save the current design. However, we are creating a new design file so you can click yes or cancel depending on the content of the present file. Then a Pop-Up appears asking to select the template. It is similar to selecting the paper size while printing. For now select default or according to the layout size of the circuit.
3- An untitled design sheet will be opened, save it according to your wish, it is better to create a new folder for every layout as it generates other files supporting your design. However, it is not mandatory.
4- Select components, Click on the component mode button.


circuitstoday - ISIS Professional
File View Edit Tools Design Graph
Component Mode
P L DEVICES

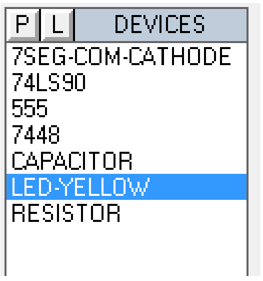
5- Click On Pick from Libraries. It shows the categories of components available and a search option to enter the part name.


circuitstoday - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
OBJECT SELECTOR
Pick from Libraries
P L DEVICES
OVERVIEW WINDOW
EDITING WINDOW

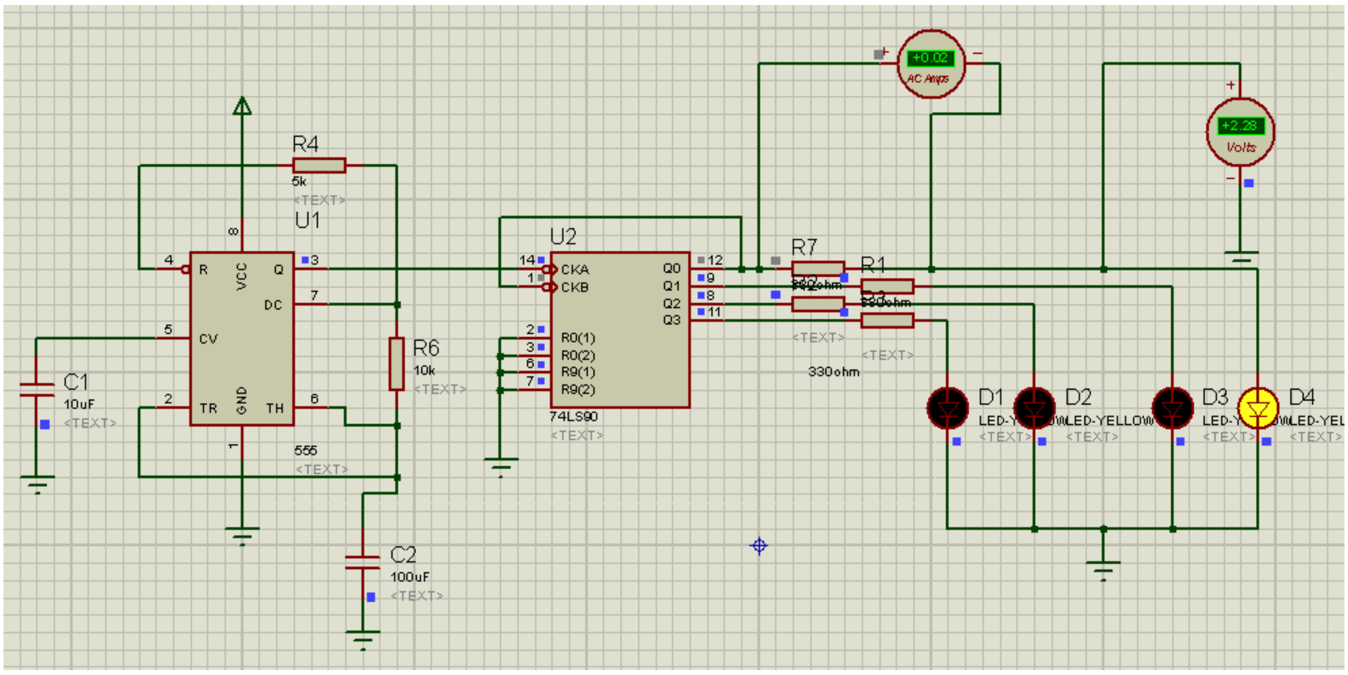
6- Select the components from categories or type the part name in Keywords text box.



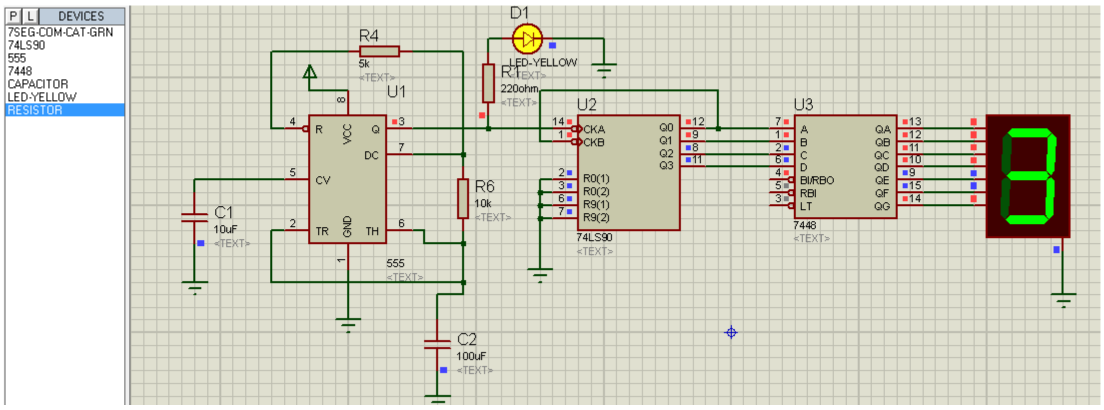
7- The selected components will appear in the devices list. Select the component and place it in the design sheet by left-click.



8- Place all the required components and route the wires i.e, make connections as follows.



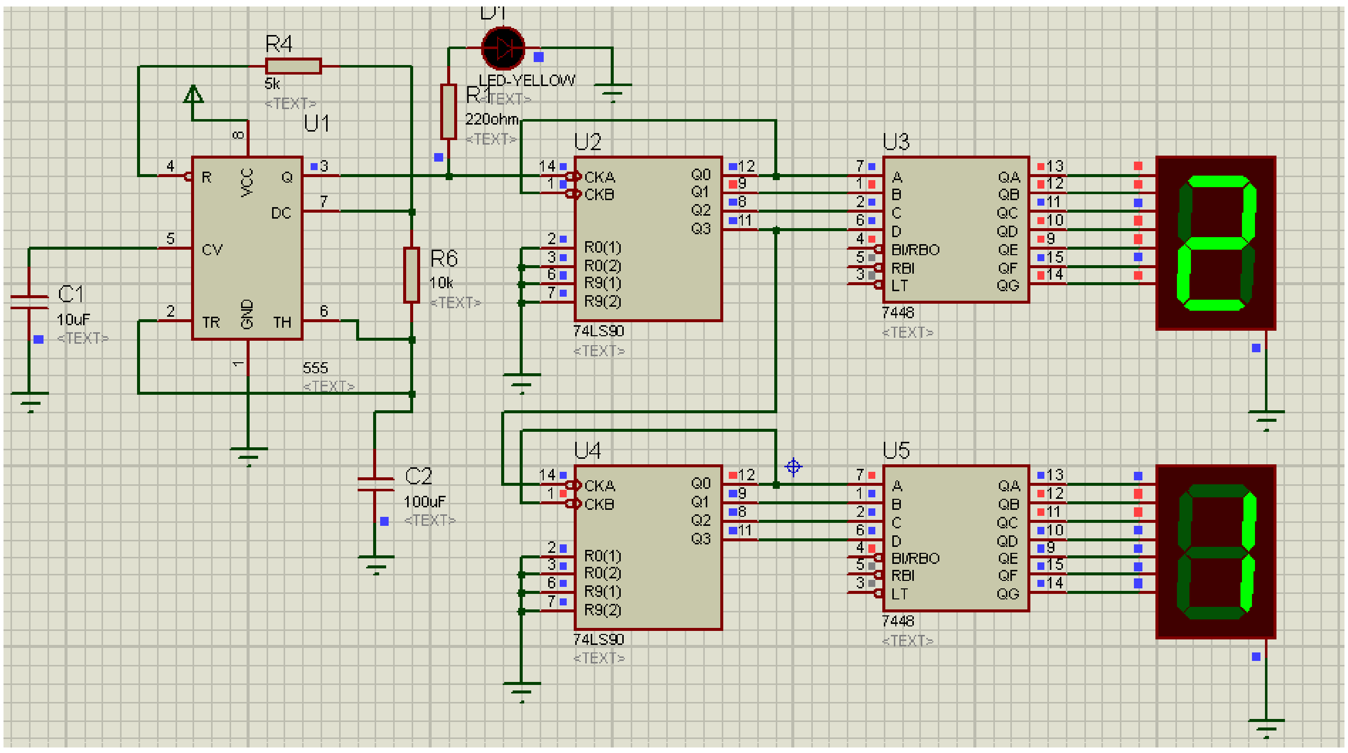
9- Adding the BCD to 7-segment decoder(7448)



DEVICES

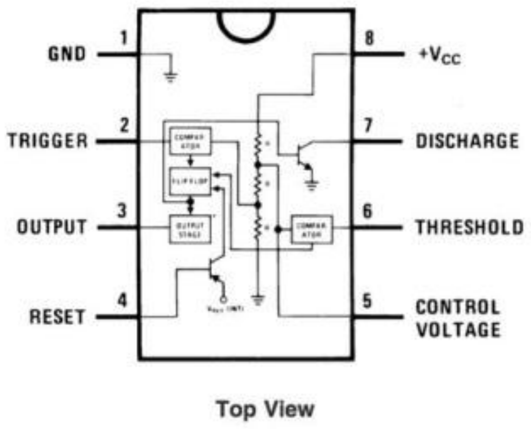
7SEG-COM-CAT-GRN
74LS90
555
7448
CAPACITOR
LED-YELLOW
RESISTOR

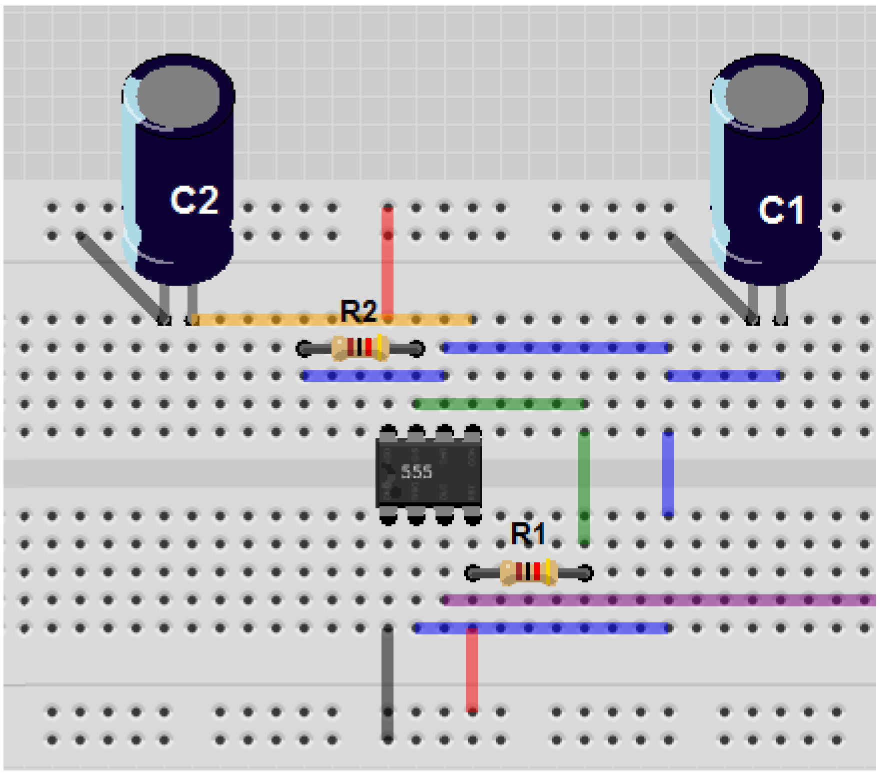
10- Counting in two digits up to 99



Hardware connection

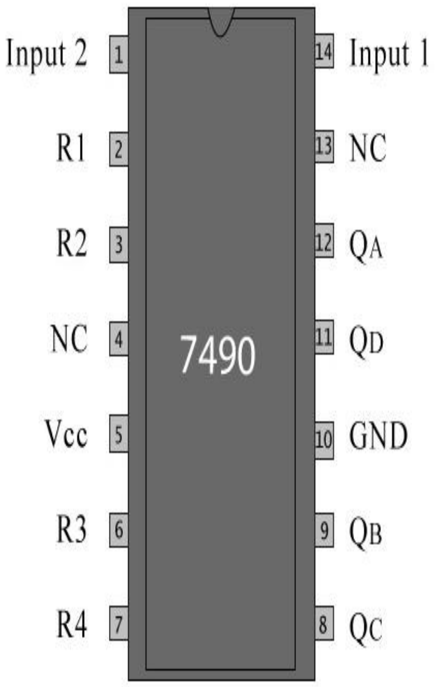
The 555 timer connection:

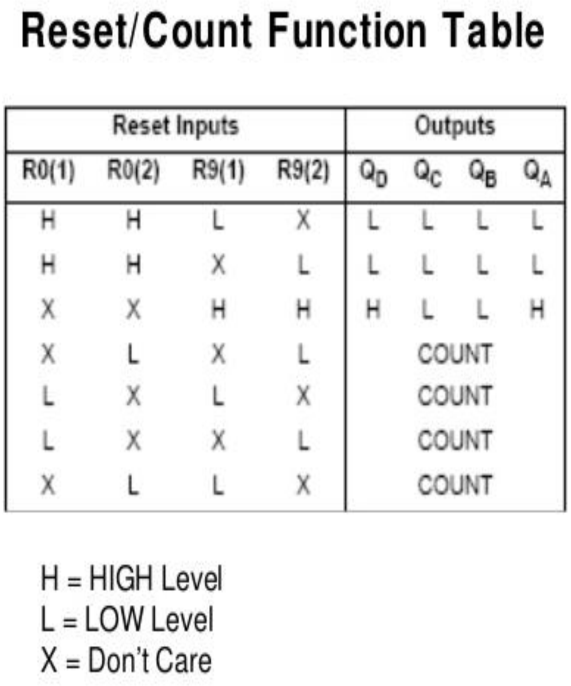




Counter connection

The 7490 connection



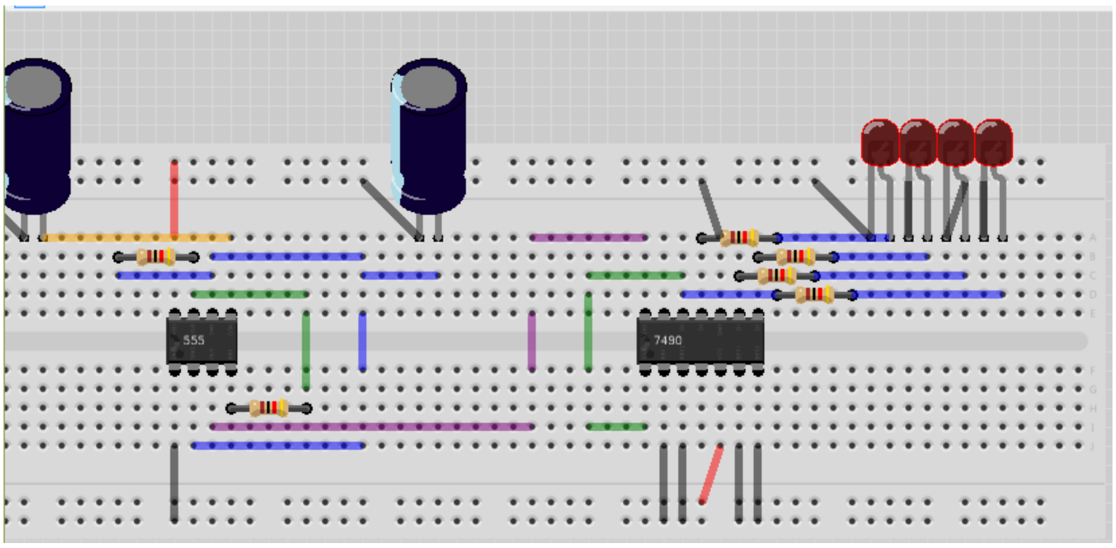


Reset/Count Function Table

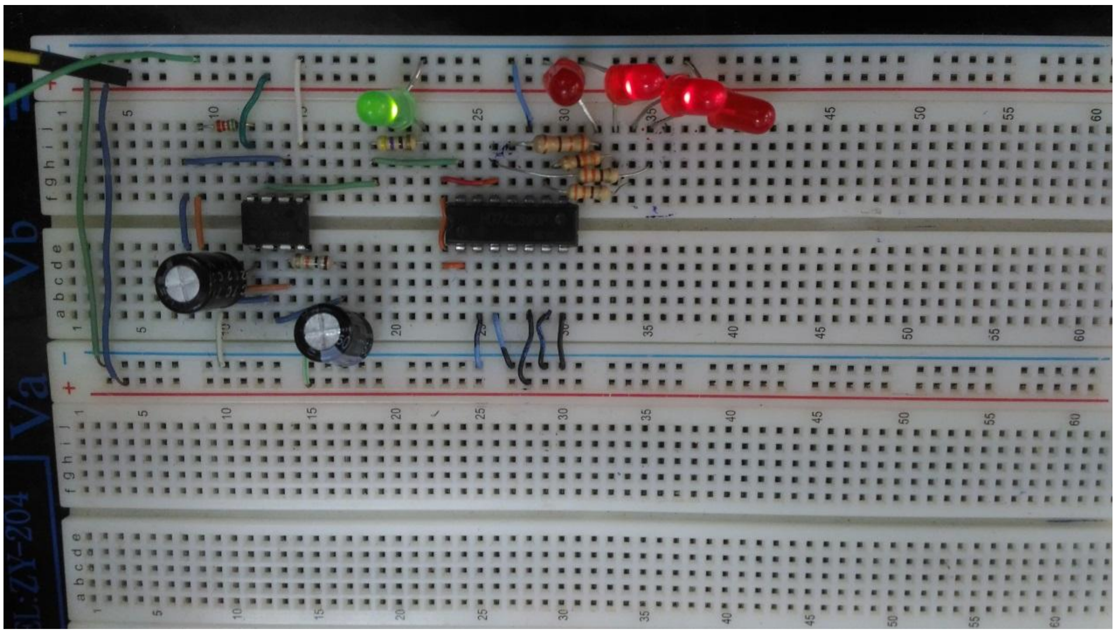
Reset Inputs				Outputs			
R0(1)	R0(2)	R9(1)	R9(2)	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

H = HIGH Level
L = LOW Level
X = Don't Care

#	Function	Name
1	Clock input 2	Input2
2	Reset1	R1
3	Reset2	R2
4	Not connected	NC
5	Supply voltage; 5V (4.75V – 5.25V)	Vcc
6	Reset3	R3
7	Reset4	R4
8	Output 3, BCD Output bit 2	QC
9	Output 2, BCD Output bit 1	QB
10	Ground (0V)	Ground
11	Output 4, BCD Output bit 3	QD
12	Output 1, BCD Output bit 0	QA
13	Not connected	NC
14	Clock input 1	Input1



The entire circuit:



Lab (2) Finite State Machines (FSM): Traffic Lights Example Design and Implementation

OBJECTIVE

1- Learn finite state machine concepts.
2- Learn finite state machine design steps.
3- Implement a traffic light controller FSM.

INTRODUCTION:

Finite state machines, or FSMs, are a class of sequential circuits. A FSM is a system which transitions from one discrete state to another, and has a finite number of states.
The state is stored in a **state register,** typically implemented as a set of D-flip flops. Flip Flops will maintain or memorize the current system state and transited to the next state with each clock positive edge. The next state of the FSM is determined from its **inputs** and its **current state**. The output of the FSM may be determined from the input and present state (known as a **Mealy** machine) or the present state only (a **Moore** machine), so some combinational logic is required.

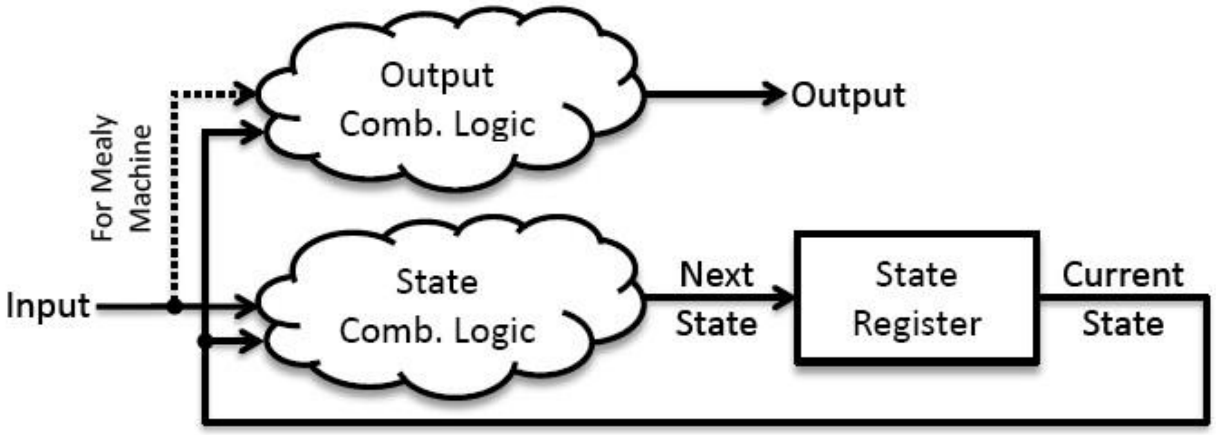


Figure 1: A Schematic of a finite state machine.

FSM: Traffic Lights Example Lab

State diagrams are used to define how a state machine transitions from one state to the next. A very simple state diagram is shown in Figure 2. This "bubble" state diagram has two states, State 0 and State 1. The name of the state and any outputs that are set in that state (LED) are shown inside the bubble. The state transitions, which depend on the inputs (button (0) and button (1)) are shown using arrows. This state diagram shows that if the machine is in State 0, and we press button (1), we'll transition to State 1 and the LED will turn on. Pressing button (1) again will do nothing. Pressing button .2(0), however, will transition back to State 0 and turn the LED off.

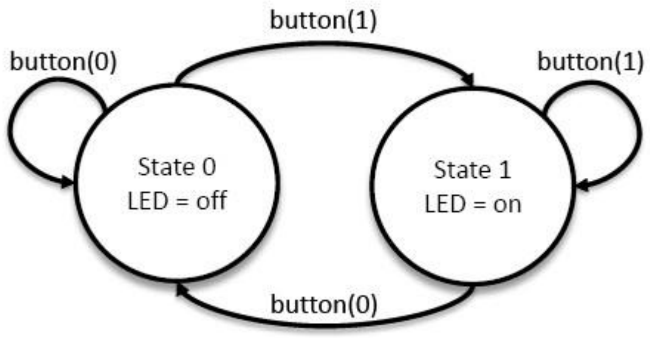


Figure 2: An example state diagram, for a simple state machine.

FSM System's Design Steps:

- Identify system states from system verbal description.
- Construct state transition diagram.
- Construct state transition table (PS-NS/O table).
- Deduce state transition equations.
- Specify hardware requirements (ICs, resistors…etc.).
- Simulate your system on appropriate software program.
- Implement your system and test your hardware.

1- *System Verbal Description:*

Traffic lights exist in an intersection of two one-way roads. The two roads are denoted by A and B. There is a car sensor in each road which senses the car existence, as shown in Figure 3-a.

When road B is on (Green light on B is on) and road A is off (Red light on A is on), and there aren't any cars on road B but there is a car on road A, the lights must switch to make road B off and road A on.

The same case when road A is on (Green light on A is on) and road B is off (Red light on B is on), and there aren't any cars on road A but there is a car on road B, the lights must switch to make road A off and road B on.

When any of the two roads is on and will be switched off it must pass with the ready case (Yellow light is on). So when the Yellow light is on, it means that the drivers should be ready to stop as the Red lights will be on soon and the road will be off.

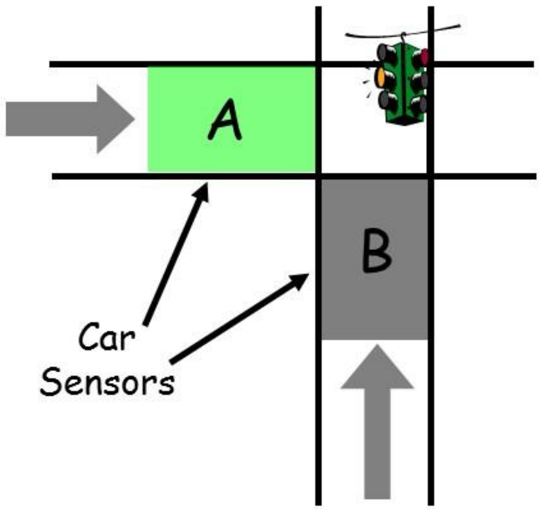


Figure 3-a: System Description.

2- *Identify System States:*

According to Figure 3-b, the system has only four possible states. Table 1 illustrates all possible system states with their description and encoding.

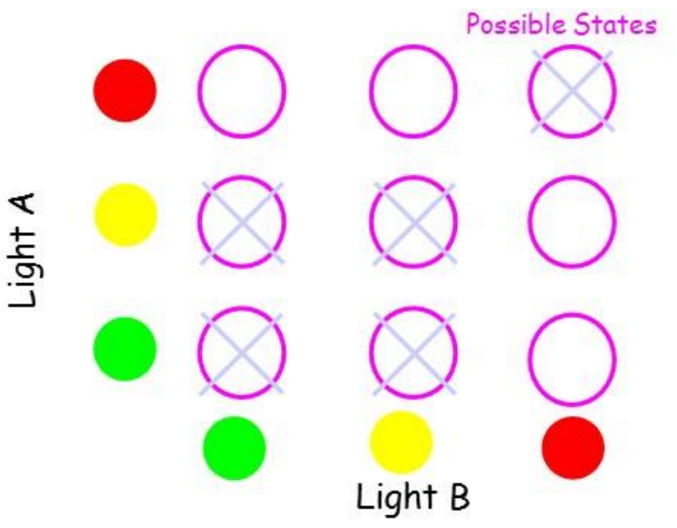
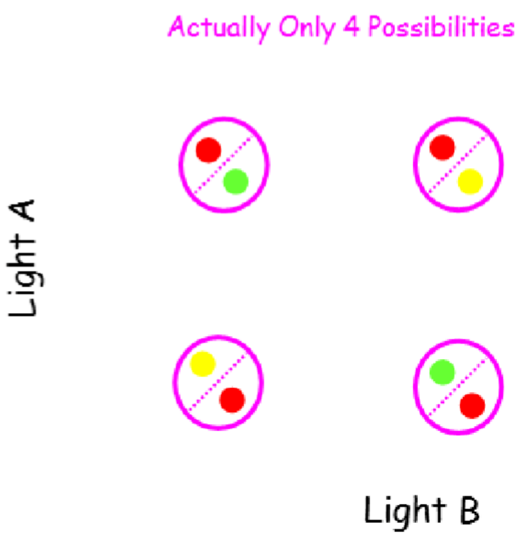


Figure 3- b: Identify System States.

Table 1: Identify System States.

	State description	Output
S_0	Road A is off(R_A is on) Road B is on (G_B is on)	R_A G_B
S_1	Road A is off(R_A is on) Road B is being ready to stop (Y_B is on)	R_A Y_B
S_2	Road A is on(G_A is on) Road B is off (R_B is on)	G_A R_B
S_3	Road A is being ready to stop (Y_A is on) Road B is off (R_B is on)	Y_A R_B



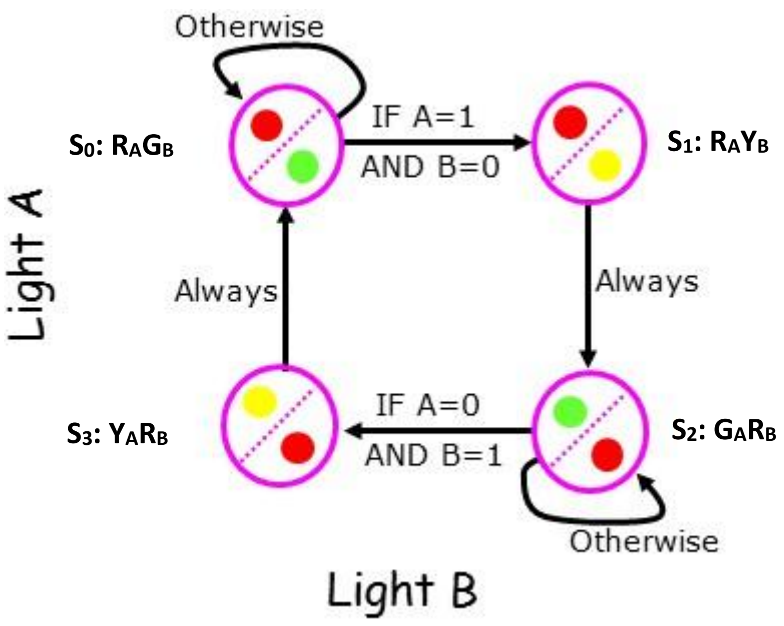
3- System State Diagram:

The system will be transited form S0 to S1, if and only if there is a car on road A and there aren't any cars on road B. Otherwise the system will stay on S0.

The system will be transited form S2 to S3, if and only if there is a car on road B and there aren't any cars on road A. Otherwise the system will stay on S2.

Transition from S1 to S2 and from S3 again to S0, will be done without any conditions to apply but only after one clock cycle. So, Yellow lights will be on for the clock cycle time.

Figure 4:
System State Diagram



4- *State Transition Table:*

For any FSM system, number of required Flip Flops is dependent on number of system states. In our traffic lights example, the system has four states encoded with two binary bits, so two D-Type Flip Flops are required. Note that:

Next state values, D1 and D2, depend on the current state and the inputs A, B. Outputs are only dependent on the current state. That's according to Moore State Machine Design.

Table 2: State Transition Table.

Current state		**Inputs**		**Next state**		**Outputs**					
Q_1	Q_2	A	B	D_1	D_2	G_A	Y_A	R_A	G_B	Y_B	R_B
0	0	0	0	0	0	0	0	1	1	0	0
		0	1	0	0						
		1	0	0	1						
		1	1	0	0						
0	1	0	0	1	0	0	0	1	0	1	0
		0	1	1	0						
		1	0	1	0						
		1	1	1	0						
1	0	0	0	1	0	1	0	0	0	0	1
		0	1	1	1						
		1	0	1	0						
		1	1	1	0						
1	1	0	0	0	0	0	1	0	0	0	1
		0	1	0	0						
		1	0	0	0						
		1	1	0	0						

5- *State Transition Equations:*

K-Maps technique could be used to drive transition equations.

$$D_1 = \overline{Q_1}Q_2 + Q_1\overline{Q_2}$$

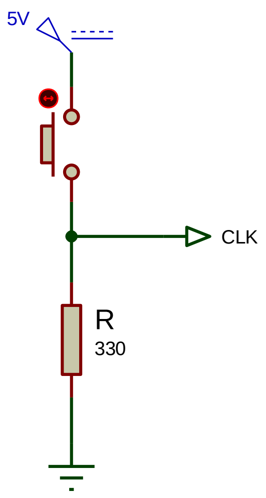
$$D_2 = \overline{Q_1}\,\overline{Q_2}\,A\,\overline{B} + Q_1\overline{Q_2}\,\overline{A}\,B$$

$$G_A = Q_1\overline{Q_2} \qquad Y_A = Q_1Q_2 \qquad R_A = \overline{Q_1}$$

$$G_B = \overline{Q_1}\,\overline{Q_2} \qquad Y_B = \overline{Q_1}Q_2 \qquad R_B = Q_1$$

6- *System Implementation:*

a. Generating the clock signal:



A push-button and a pull-down resistor could be used as ***manual clock*** signal. Pull-down resistor pulls the pin to a logical low value when the button isn't pushed. It is connected between ground and the appropriate pin on a device. Whenever the button is being pushed, it generates +ve edge.

More appropriate idea to generate the clock signal is to use ***timer 555 IC*** described in LAB1.

Assume that 0.67Hz is the needed CLK frequency, to produce 1.5 seconds as the CLK period. So use R1=1KΩ, R2=10KΩ, C1=100μF

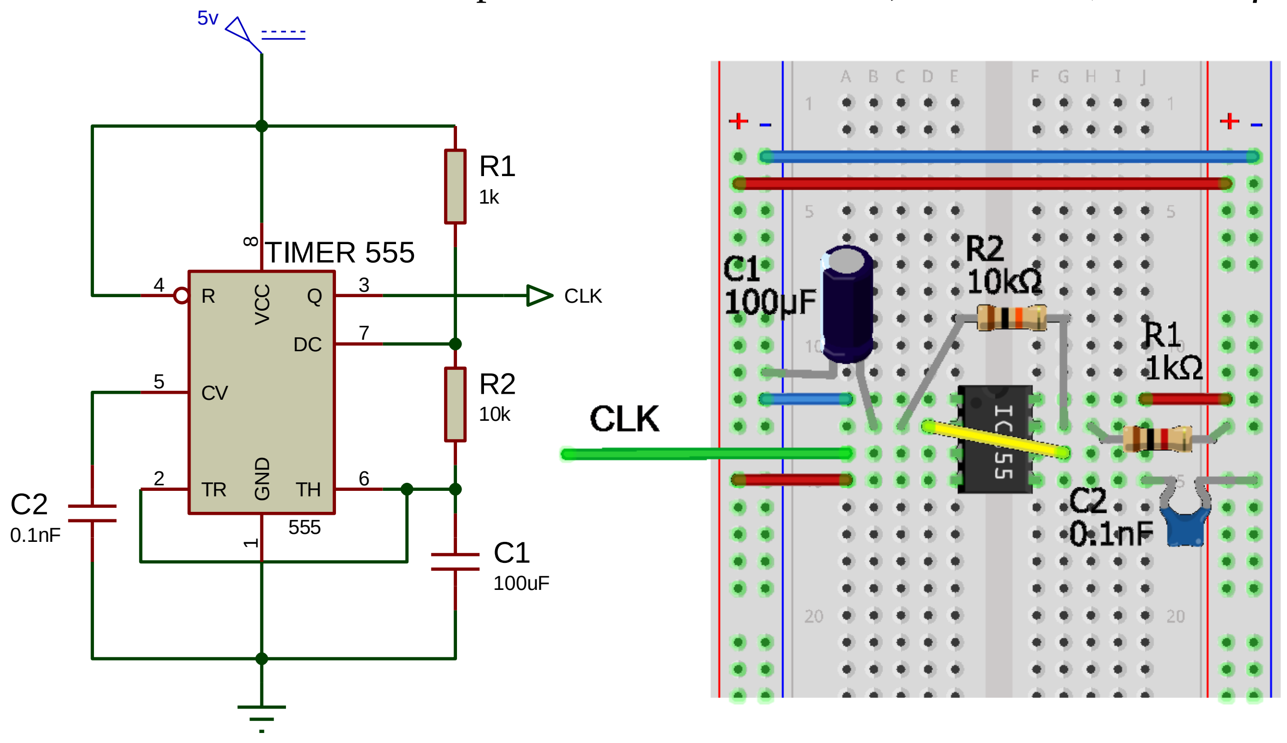
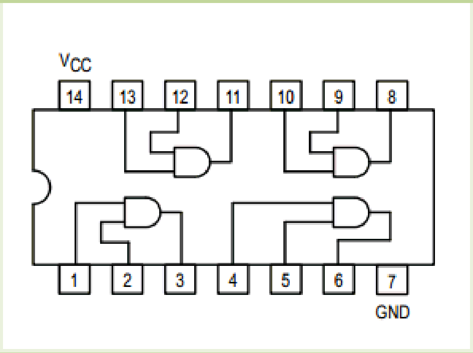
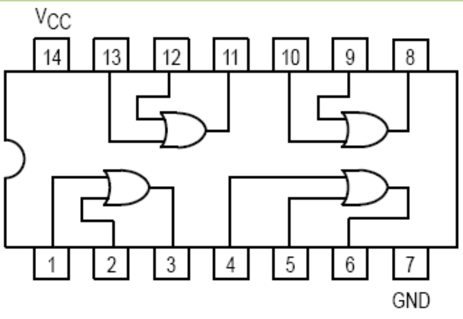
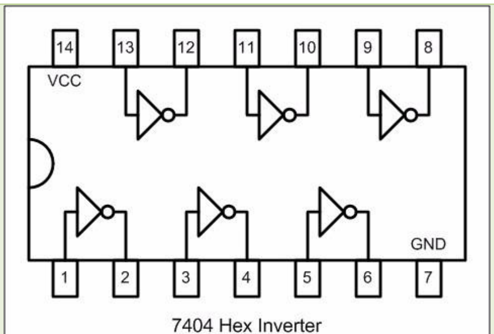
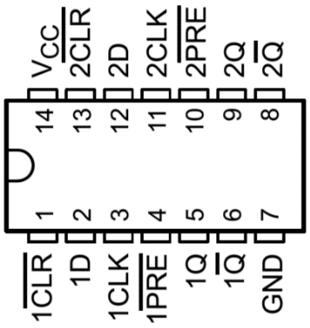


Figure 5: Clock implementation using Timer IC555

b. Hardware requirements:

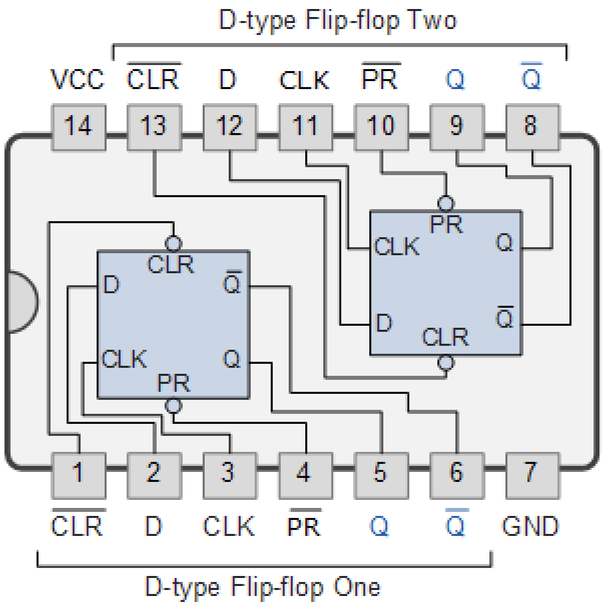
- Bread board, wires, IC555, resistors (1KΩ, 10KΩ), and capacitors (100µF, 0.1nF).
- 2 on-off switches instead of the 2 car sensors A and B.
- 2 Green, 2 Yellow and 2 Red Color LEDs.
- 8 resistors (330Ω)

And the following ICs:

IC	Quantity	Description	Pin configuration
7408	3	Quad 2 input AND gate	 VCC 14 13 12 11 10 9 8 1 2 3 4 5 6 7 GND
7432	1	Quad 2 input OR gate	 VCC 14 13 12 11 10 9 8 1 2 3 4 5 6 7 GND
7404	1	Hex inverter	 14 13 12 11 10 9 8 VCC GND 1 2 3 4 5 6 77404 Hex Inverter
7474	1	Dual D-type Flip flop	 VCC 2CLR 2D 2CLK 2PRE 2Q 2Q̄ 14 13 12 11 10 9 8 1 2 3 4 5 6 7 1CLR 1D 1CLK 1PRE 1Q 1Q̄ GND

c. D-Flip Flop IC description:

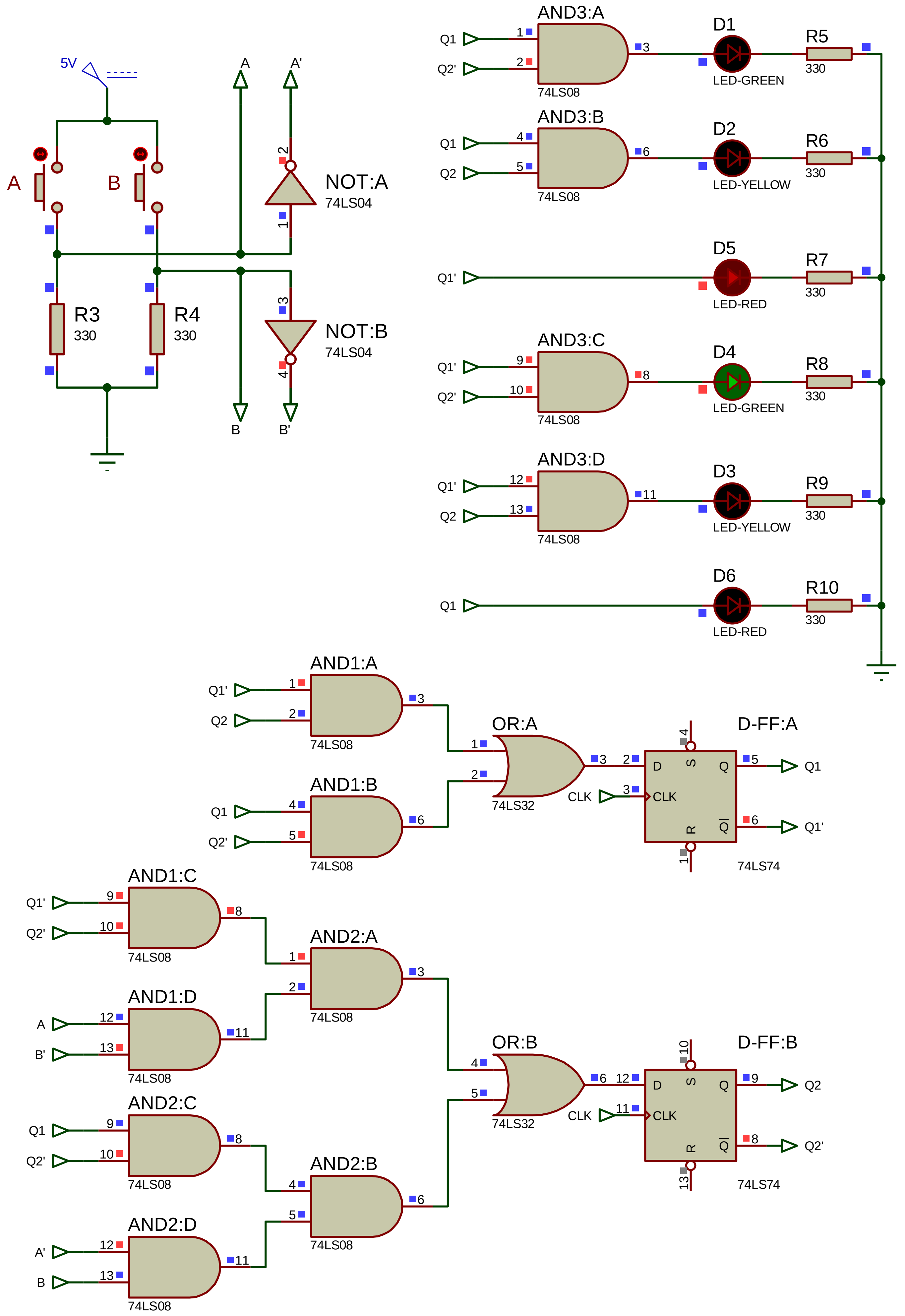
74LS74: dual D positive edge triggered flip-flop, asynchronous preset and clear.



Pin	Symbol	Description
1	$\overline{CLR}$	asynchronous clear input (active LOW)
2	D	data input
3	CLK	clock input (LOW-to-HIGH, edge-triggered)
4	$\overline{PR}$	asynchronous preset input (active LOW)
5	Q	Output
6	$\overline{Q}$	complement output
7	GND	ground(0V)
14	VCC	Supply voltage(5V)

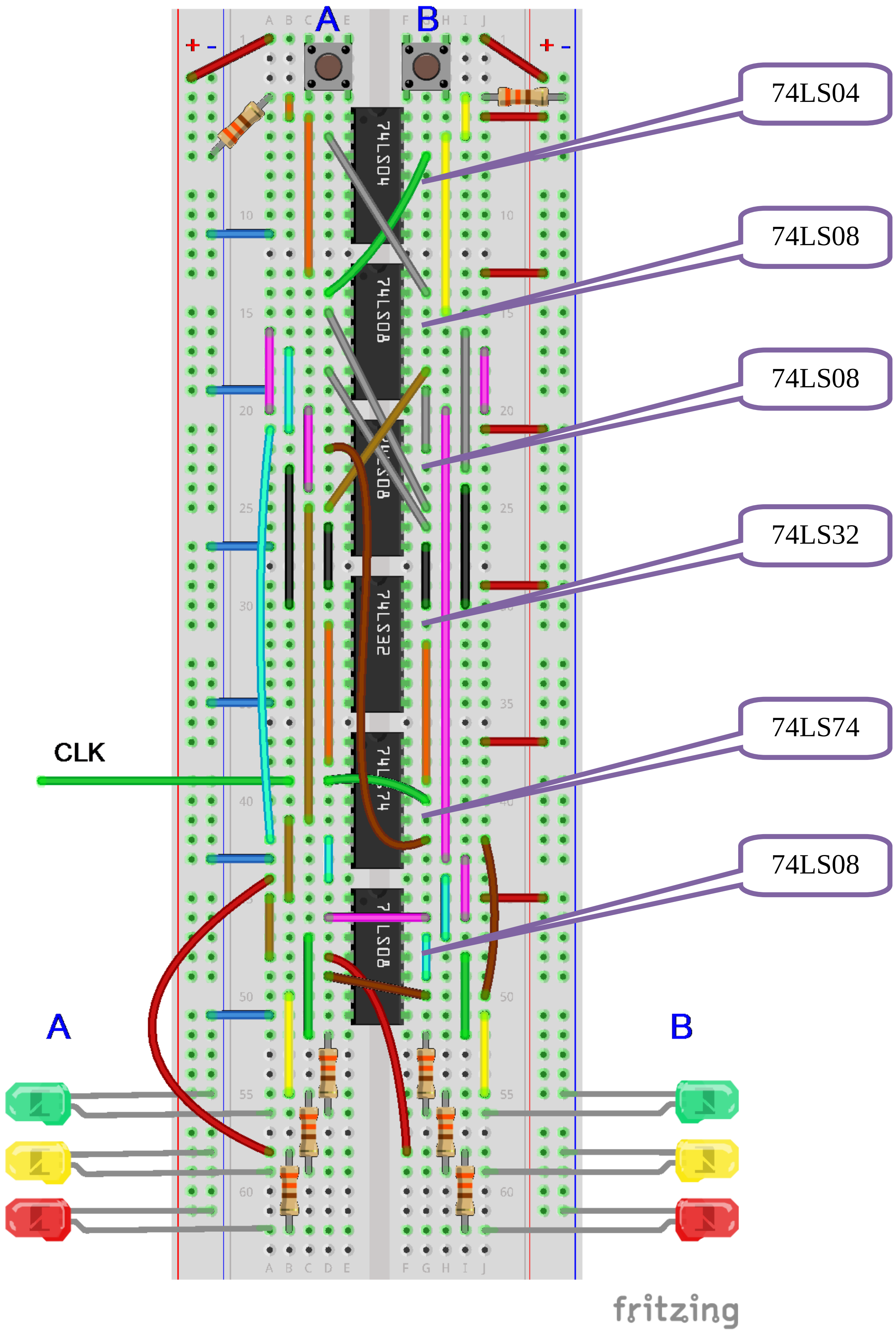
d. Software simulation:

Following the same steps from LAB1 to Place all the required components and route the wires as the following figure.

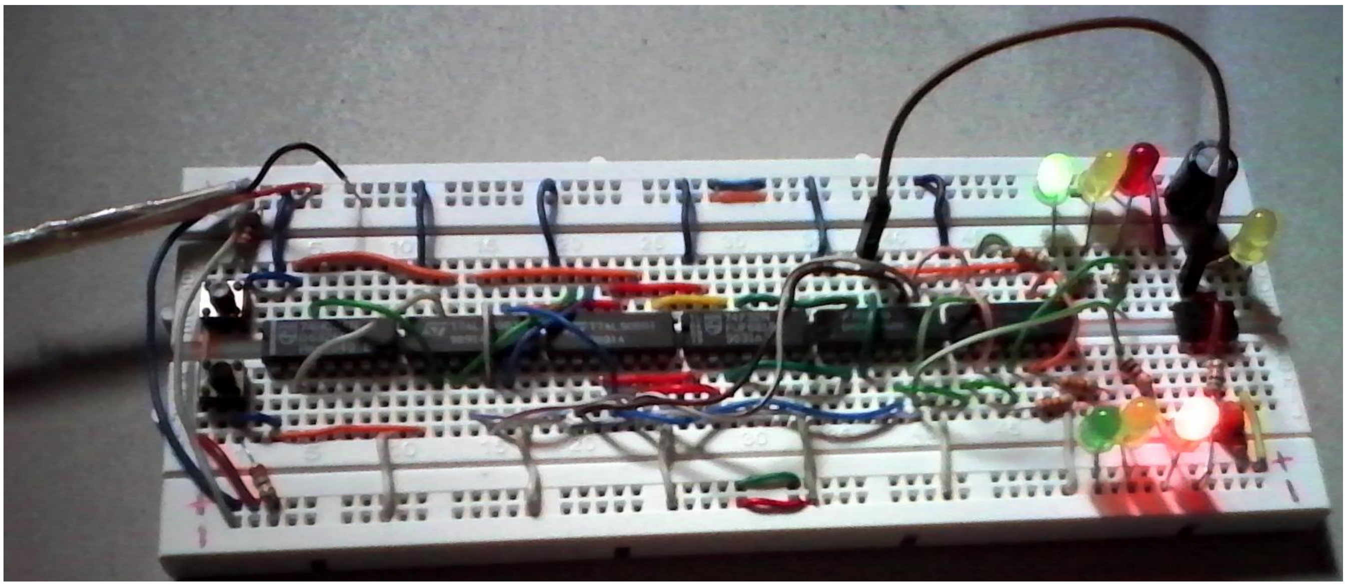


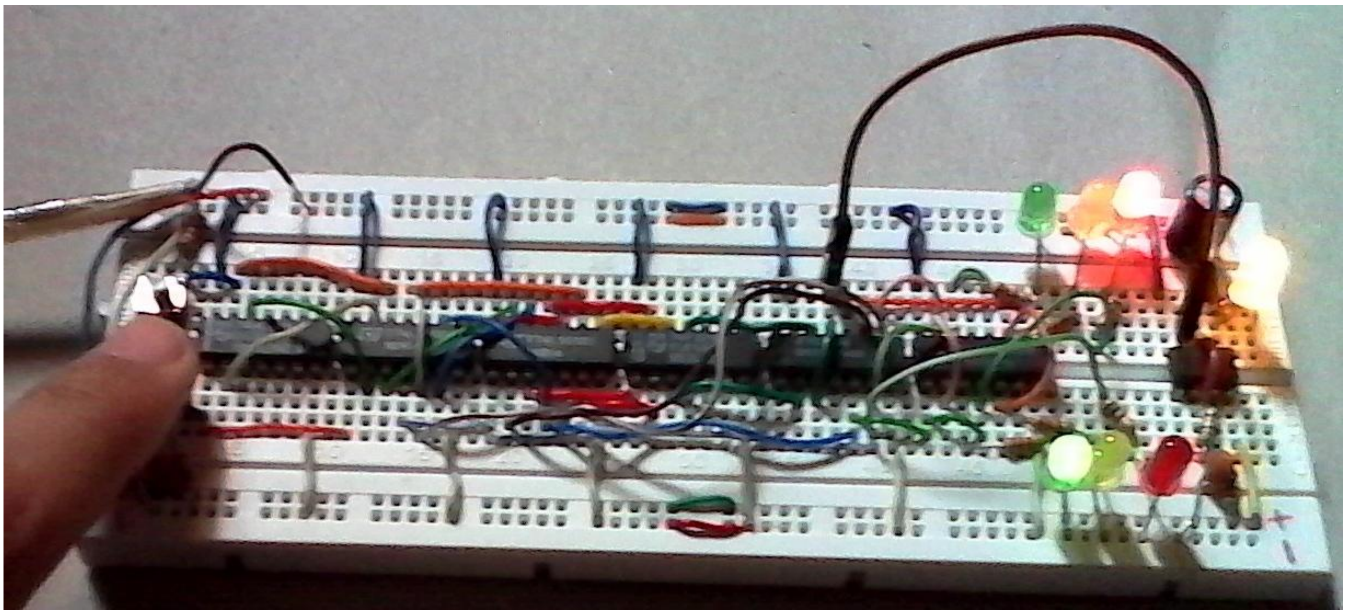
5V
A
B
R3
330
R4
330
A
A'
B
B'
NOT:A
74LS04
NOT:B
74LS04
AND3:A
AND3:B
AND3:C
AND3:D
74LS08
D1
LED-GREEN
D2
LED-YELLOW
D5
LED-RED
D4
LED-GREEN
D3
LED-YELLOW
D6
LED-RED
R5
R6
R7
R8
R9
R10
330
Q1
Q2'
Q2
Q1'
AND1:A
AND1:B
AND1:C
AND1:D
AND2:A
AND2:B
AND2:C
AND2:D
OR:A
OR:B
74LS32
CLK
D-FF:A
D-FF:B
74LS74
D
Q
S
R
CLK

e. Hardware connections:



f. Hardware on breadboard:





Lab (3) Full Adder Circuit Implementation with the Standard Logic Gates

Objectives:

3. Construct and investigate the behavior of simple logic gates and simple logic Circuits using TTL components.
4. Understand the arithmetic and logical operations using Half and Full Adder, Half and Full subtractor and Binary Adder

Introduction:

Digital computers perform variety of information tasks. Among the functions encountered are the various arithmetic operations. The most basic arithmetic operation is the addition and subtraction of two binary digits. This simple operation consists of four possible elementary operations. 0 (+ or -) 0, 0 (+ or -) 1 , 1 (+ or -) 0 , 1 (+ or -) 1 .In summation for example the first three operations produce sum of one digit, but when the both augends and addend bits are equal 1, the binary sum consists of two digits. The higher significant bit of the result is called carry .When the augends and addend number contains more significant digits, the carry obtained from the addition of the two bits is called **half adder**. One that performs the addition of three bits (two significant bits and a previous carry) is called **Full adder**. The name of circuits from the fact that two half adders can be employed to implement a full adder

Requirements:

1. Basic gates (AND – OR – NOT – XOR)
2. four led , resistors ,wires and switches
3. Five Volts Power supply.
4. Bread Board.

Procedure :

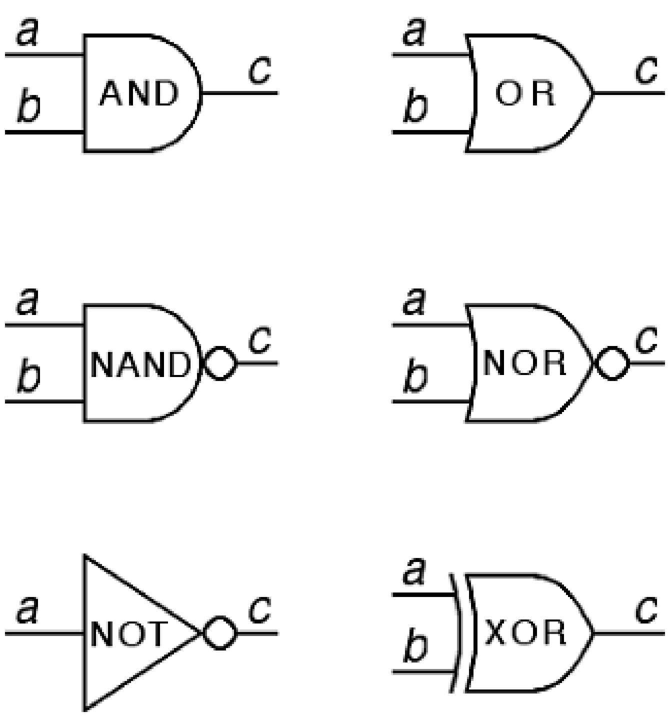
Logic Level:

- We know that any processor work with digital signal ,which are 0 or 1 signal
- Physically zero mean (zero volts to about 1 volt) , one mean (5 volts + or – 1 volt)

The standard logic gates are :

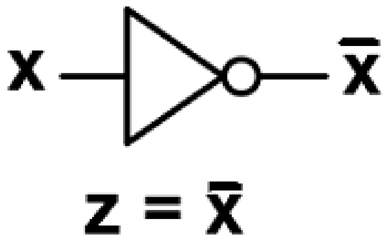
- Not – AND – OR – NAND – NOR – XOR – XNOR

The standard logic gates Symbols:



Not Gate :

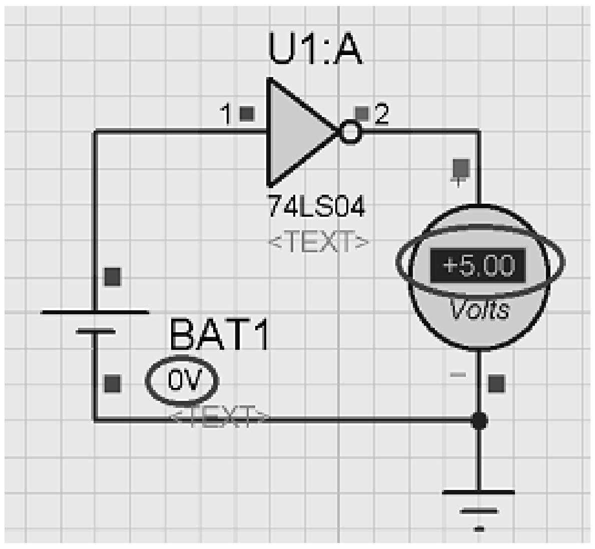
- **It is called (the inverter gate) because it inverts the input**
- **If the input is 0 it convert it to 5 volts**
- **If the input is 5 it converts it to 0 volts**

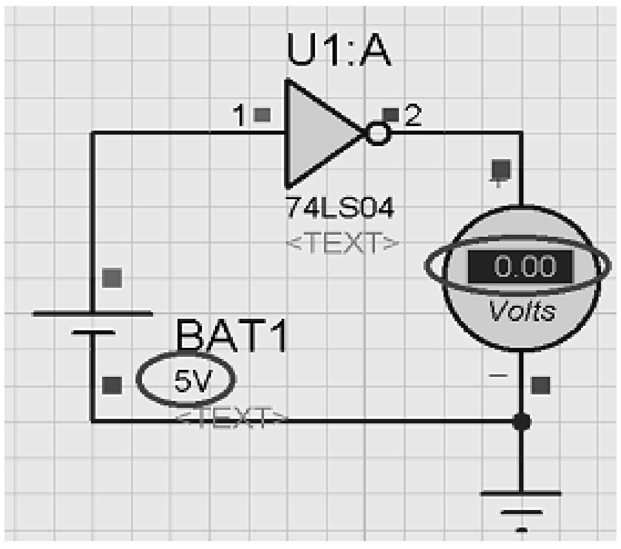


Truth table:

NOT gate truth table

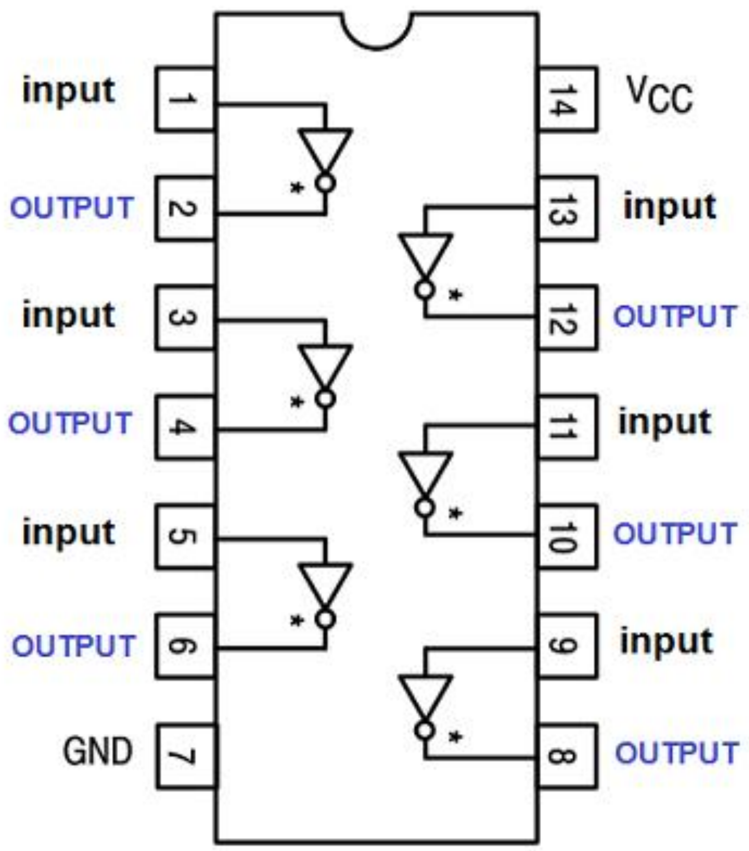
Input	Output
0	1
1	0





One of the IC's that is used as an inverter the following :

- **74ls04 You can look at the datasheet for 74ls04 :**

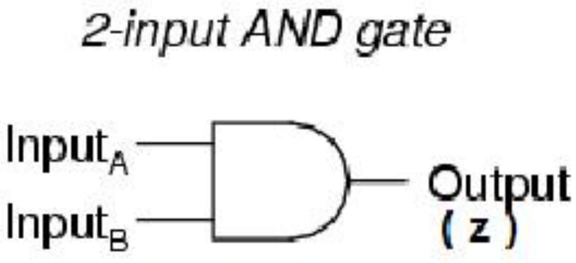

input
OUTPUT
input
OUTPUT
input
OUTPUT
GND
VCC
input
OUTPUT
input
OUTPUT
input
OUTPUT
1
2
3
4
5
6
7
14
13
12
11
10
9
8

HD74LS04P
3M36

74LS04J
RAYT7520

AND Gate

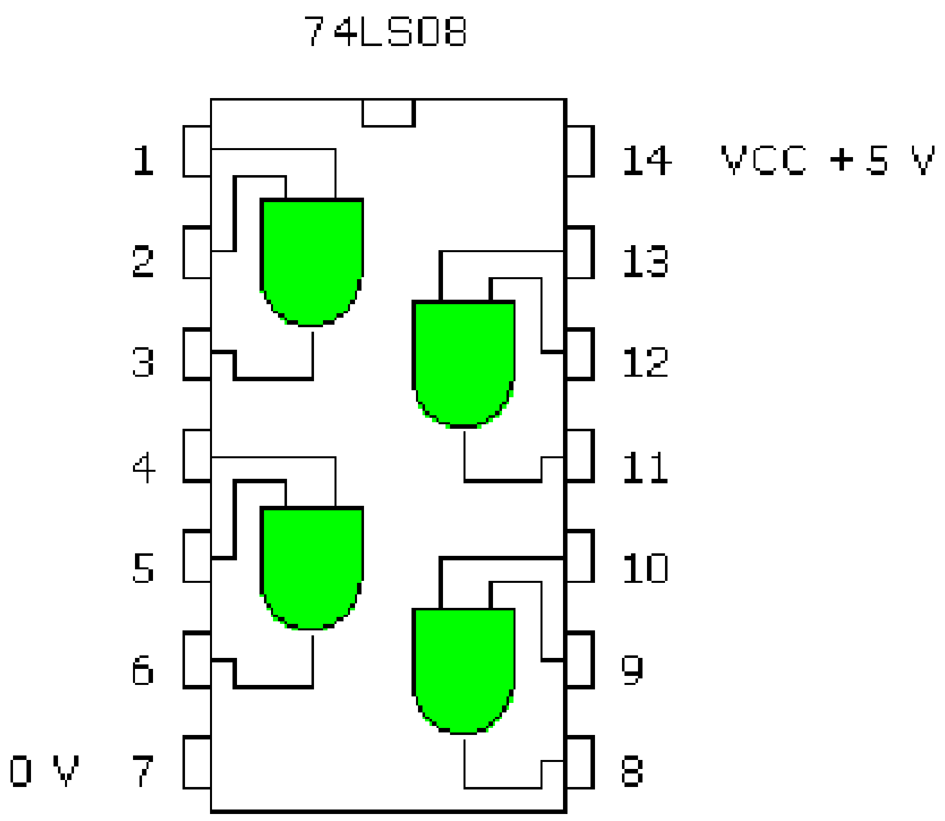
- It may have more than 2 inputs and only one outputs
- The output is 1 when all the input are 1



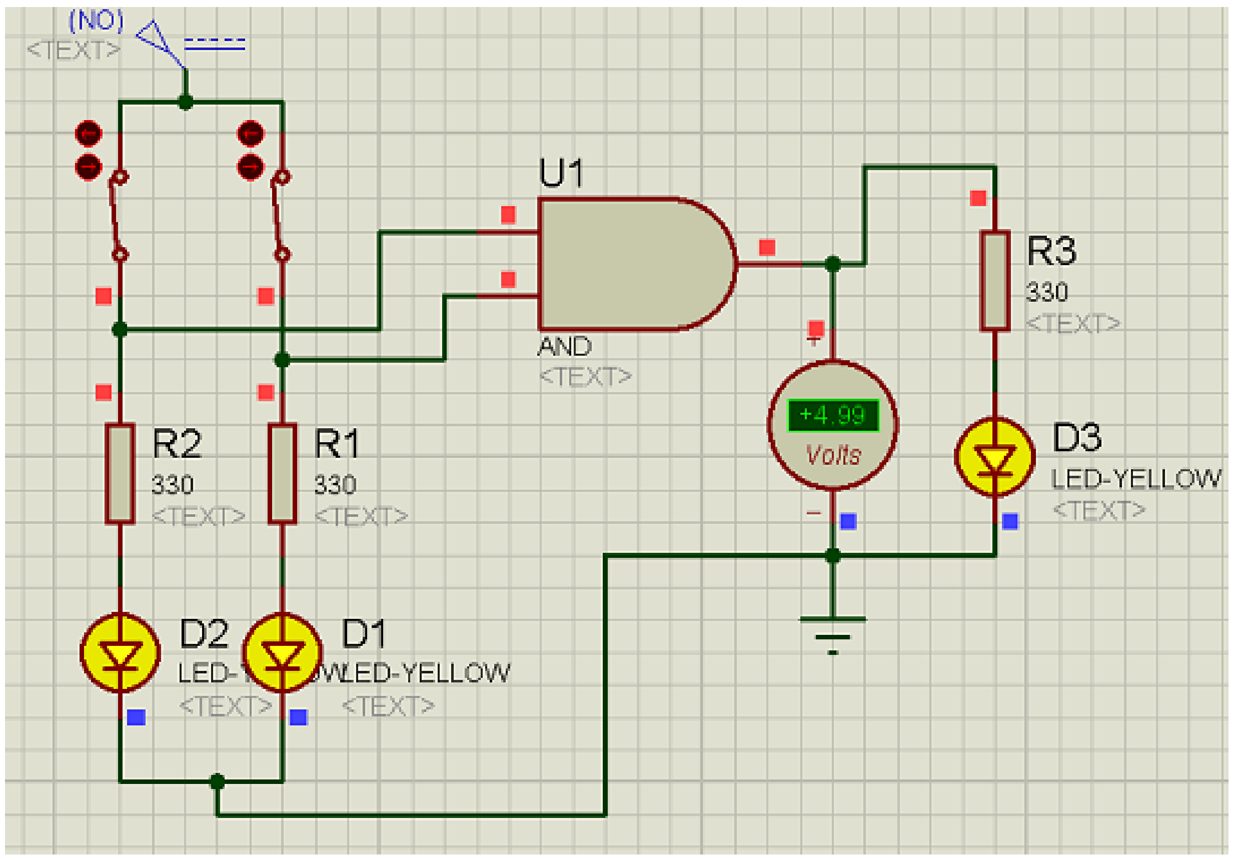
Z = A . B

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

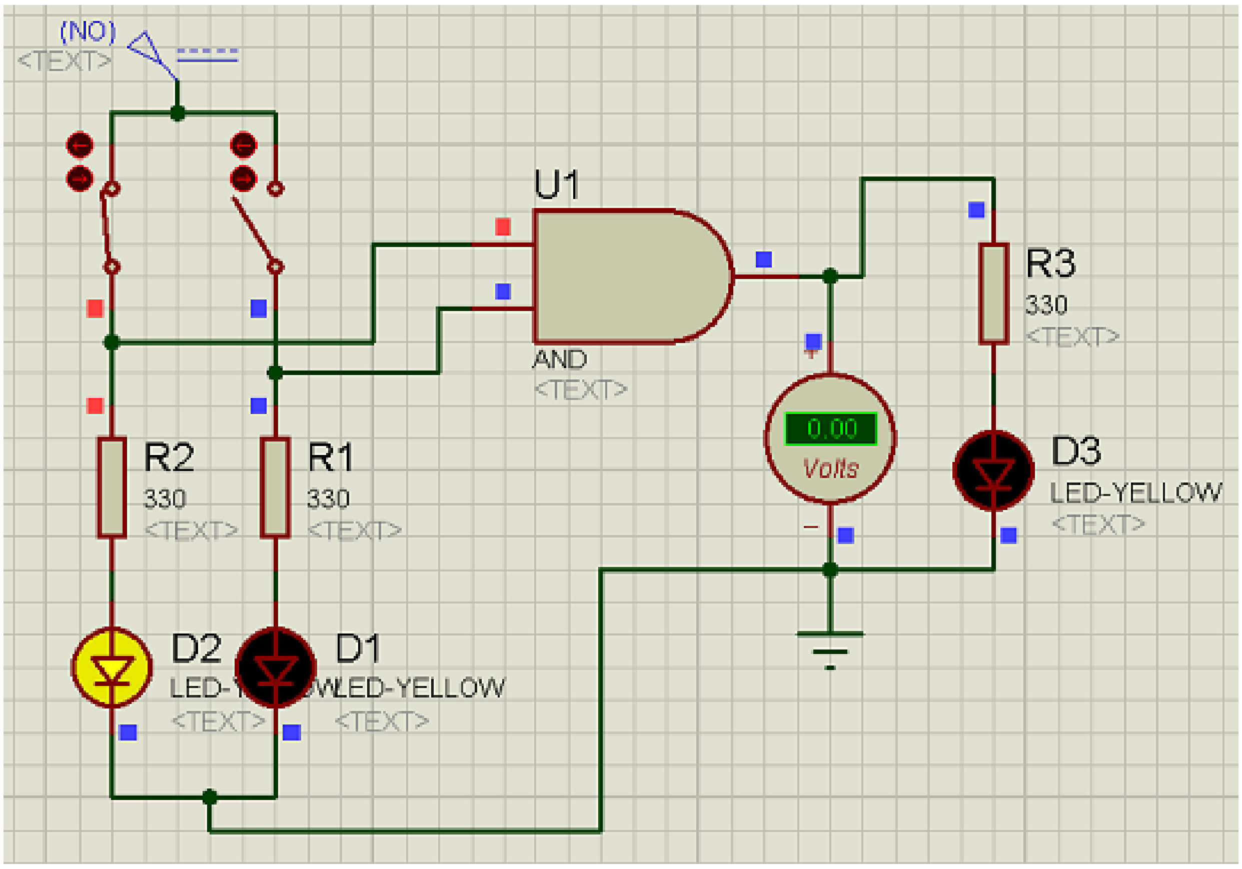
- You can use 74Ls08N Ic as an quad and gate

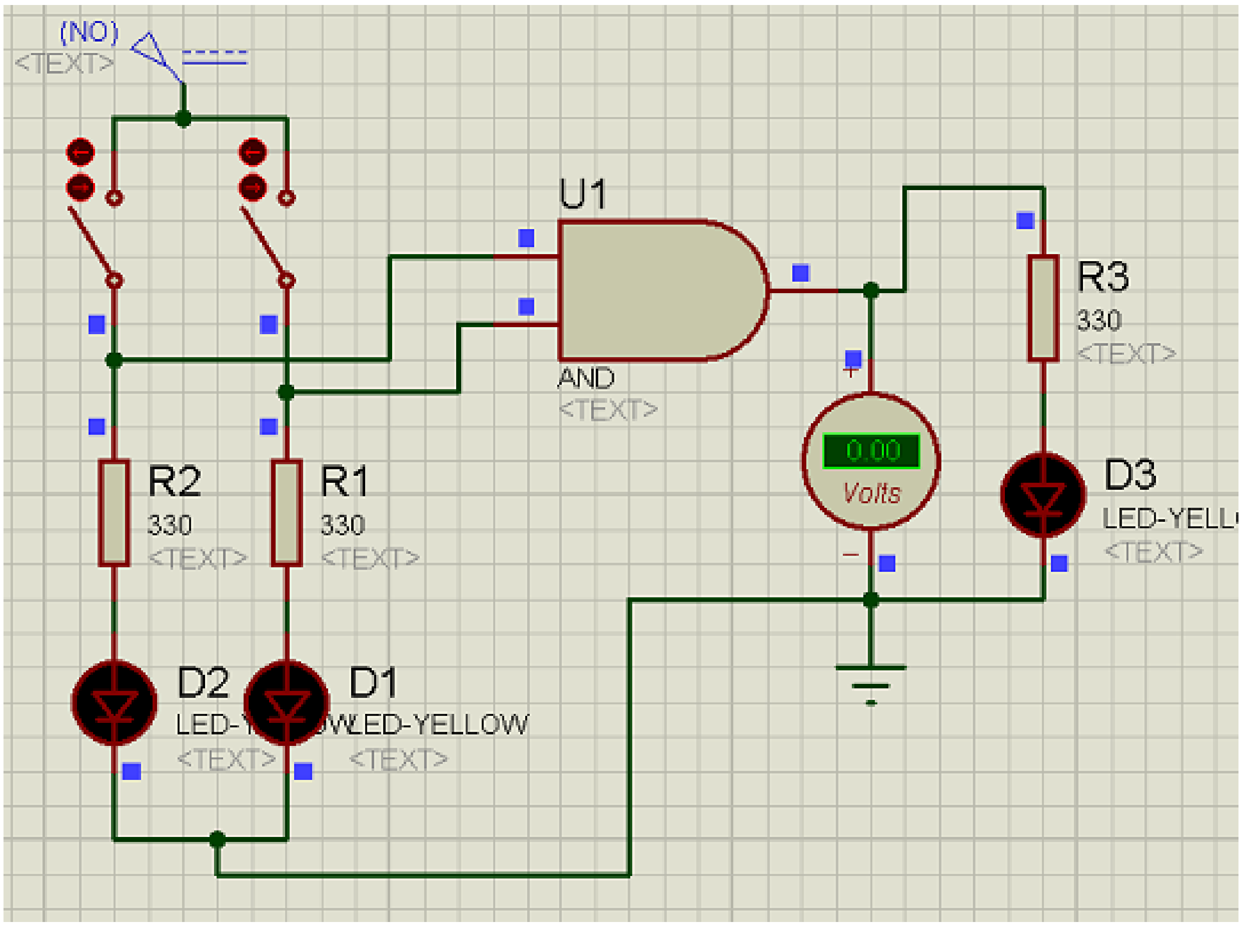


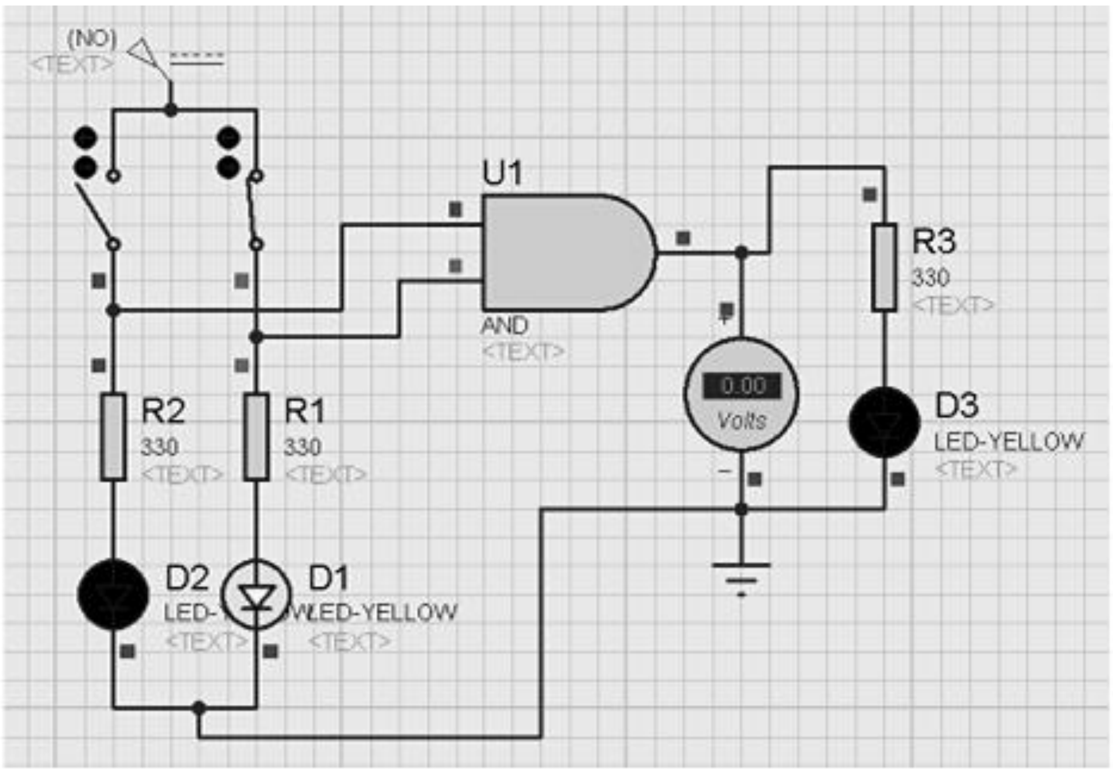
74LS08
1
2
3
4
5
6
0 V
7
14
VCC +5 V
13
12
11
10
9
8

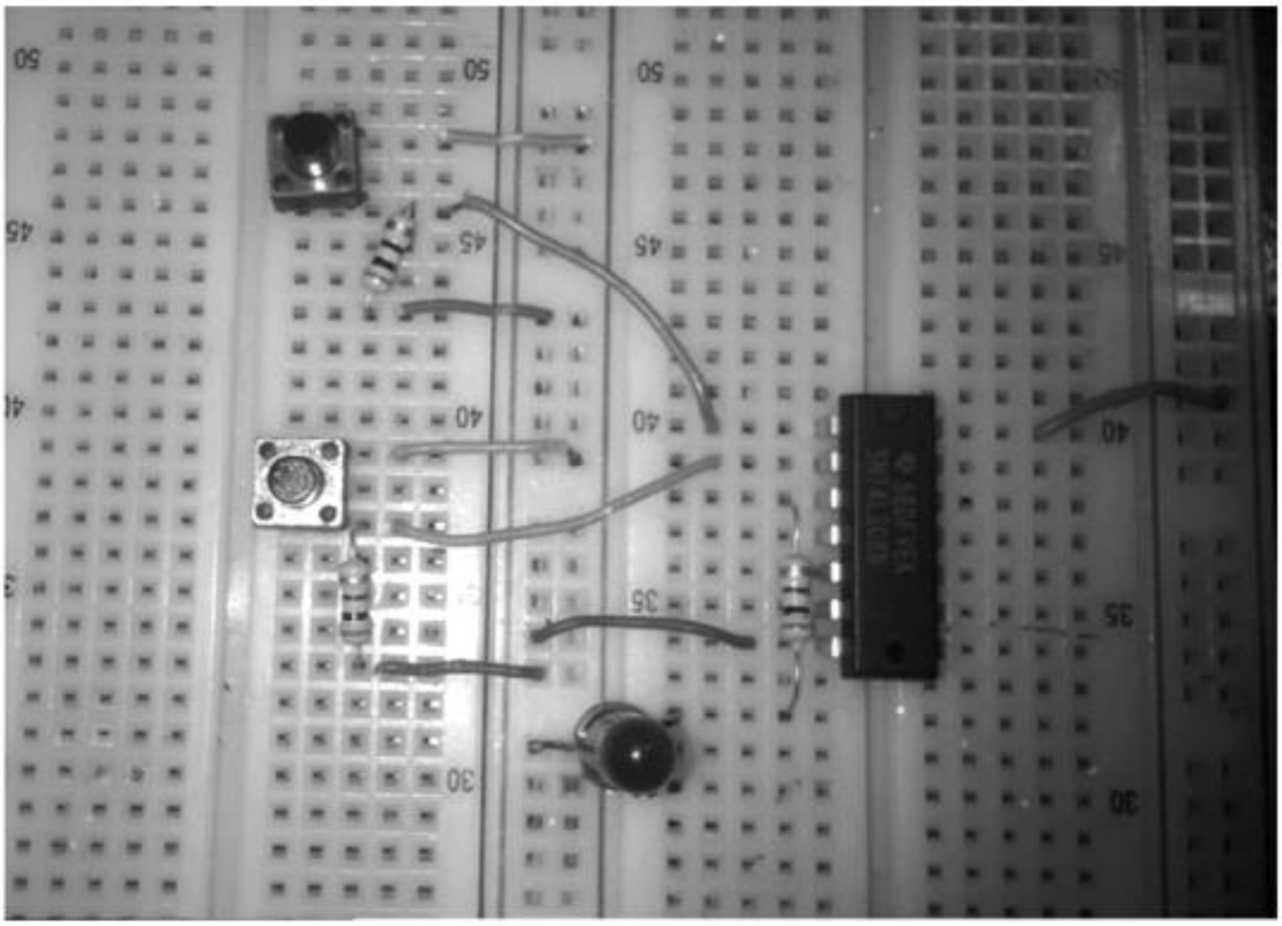


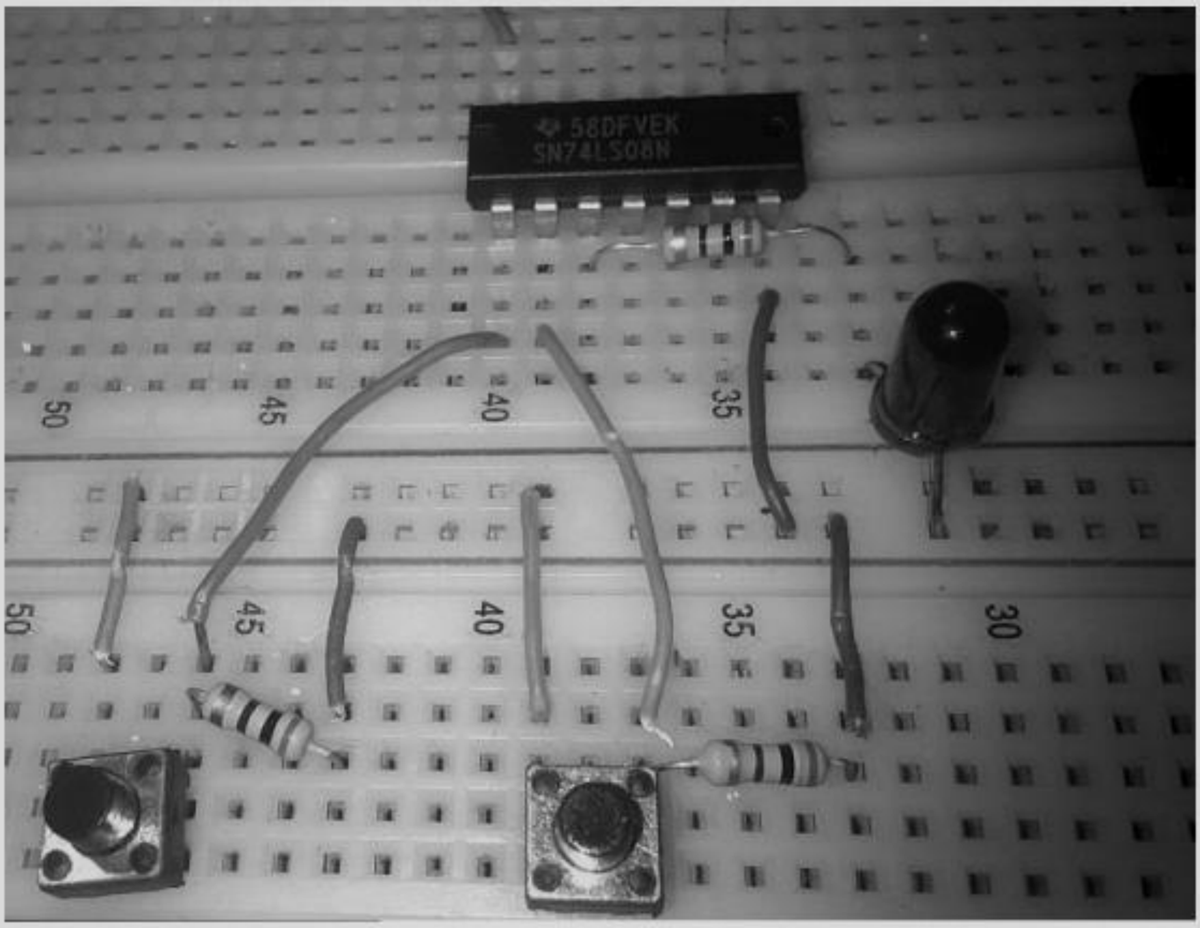
(NO)
<TEXT>
U1
AND
<TEXT>
R3
330
<TEXT>
+4.99
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
D2
D1
LED-YELLOW
<TEXT>
<TEXT>

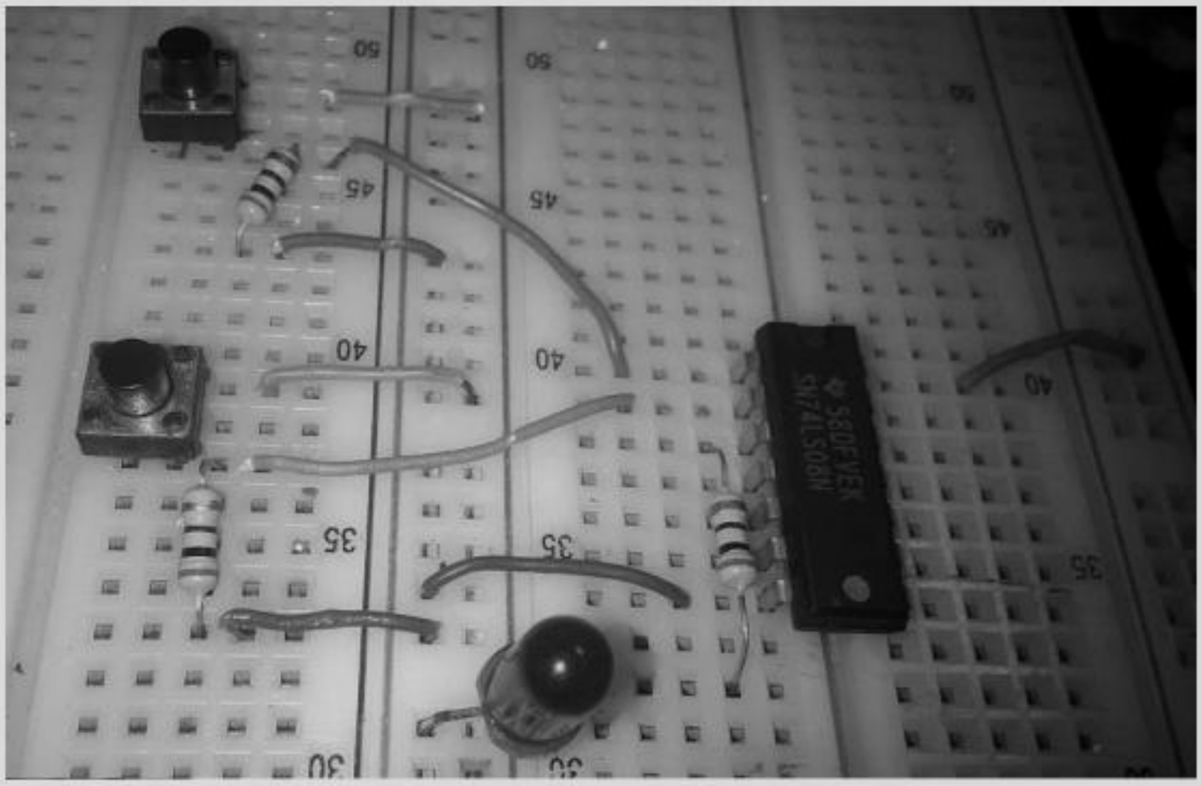

(NO)
<TEXT>
U1
AND
<TEXT>
R3
330
<TEXT>
0.00
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
D2
D1
LED-YELLOW
<TEXT>


(NO)
<TEXT>
U1
AND
<TEXT>
R3
330
<TEXT>
0.00
Volts
D3
LED-YELL
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
D2
D1
LED-YELLOW
<TEXT>


(NO)
<TEXT>
U1
AND
<TEXT>
R3
330
<TEXT>
0.00
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
D2
LED-
<TEXT>
D1
LED-YELLOW
<TEXT>

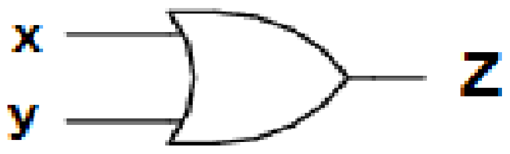



58DFVEK
SN74LS08N


58DFVEK
SN74LS08N

- This gate used as following :
- The output is 1 if any of the input or all of them is 1 .

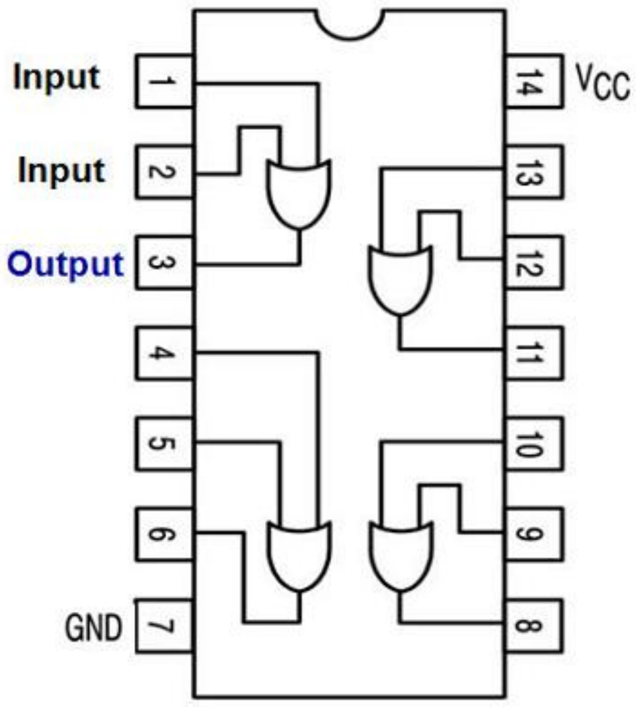
Or Gate

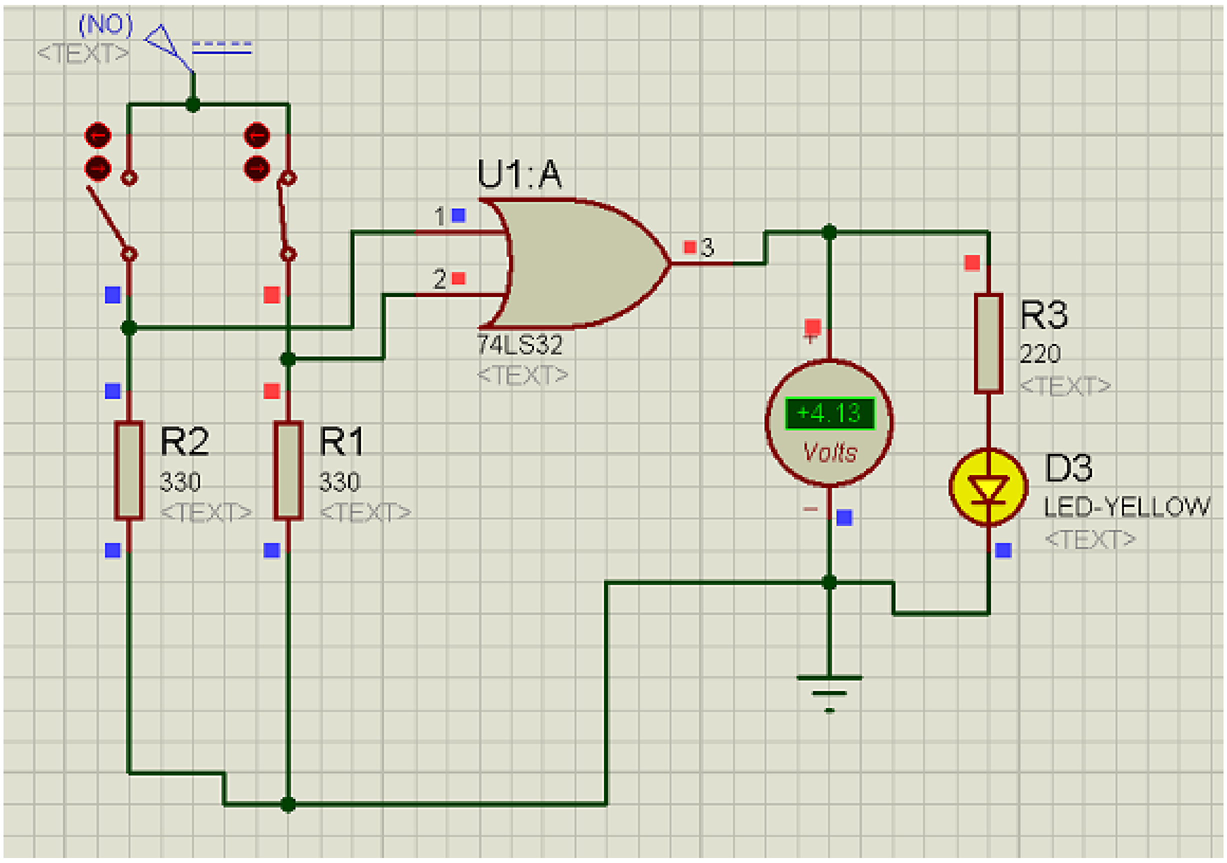


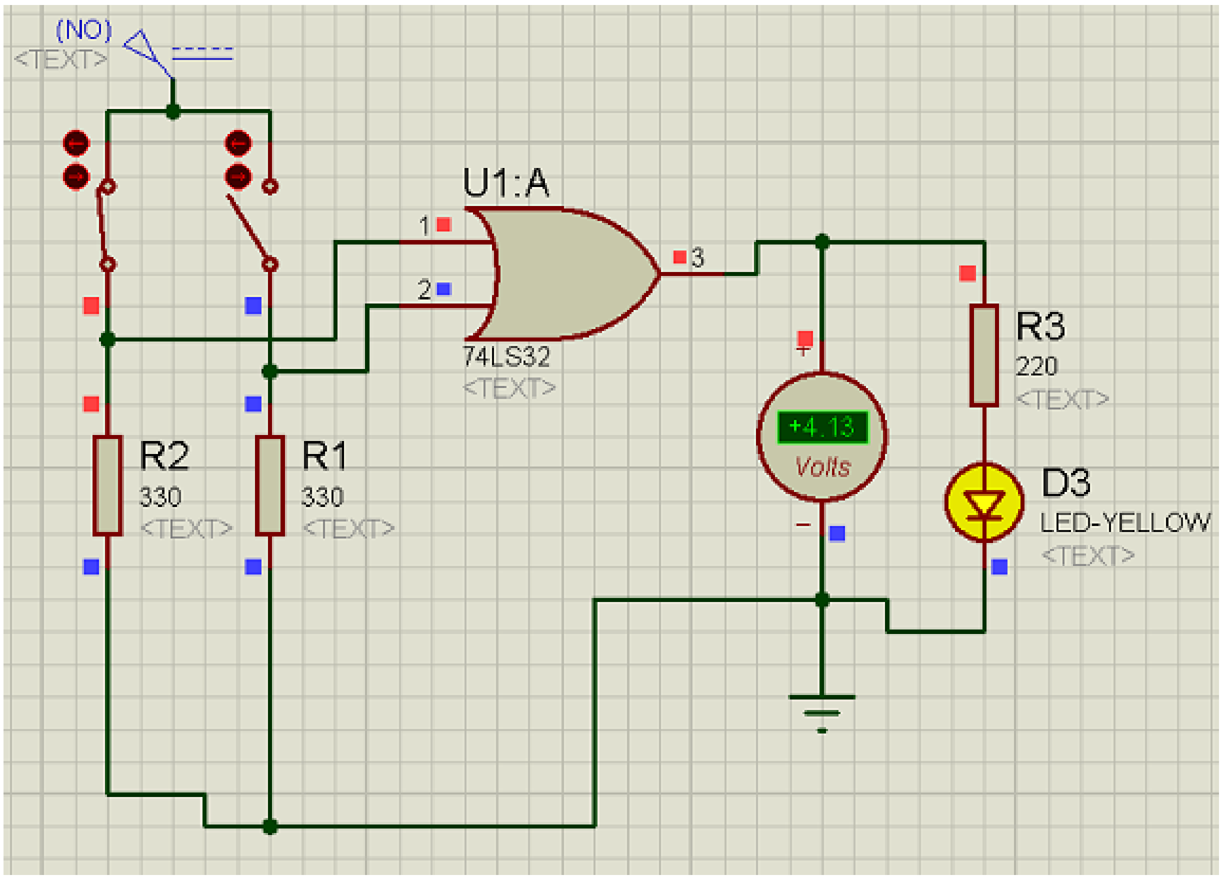
$Z = x + y$

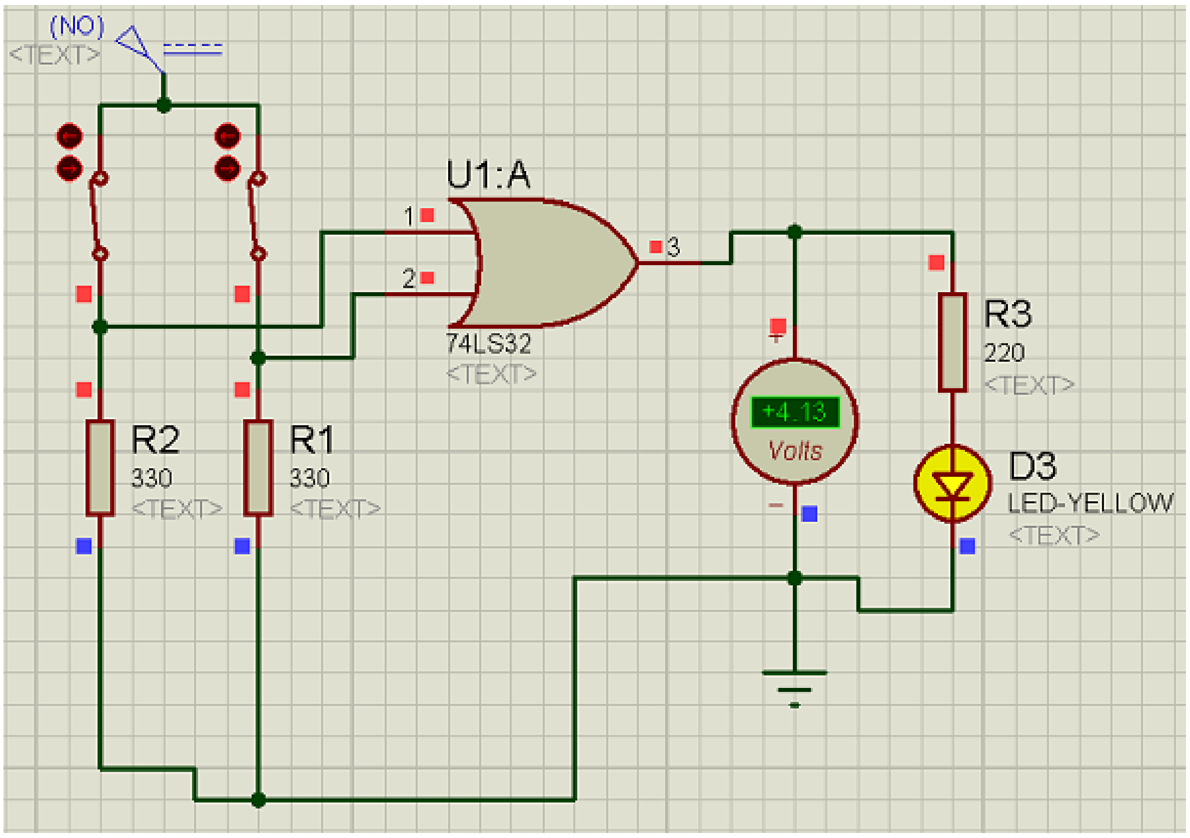
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

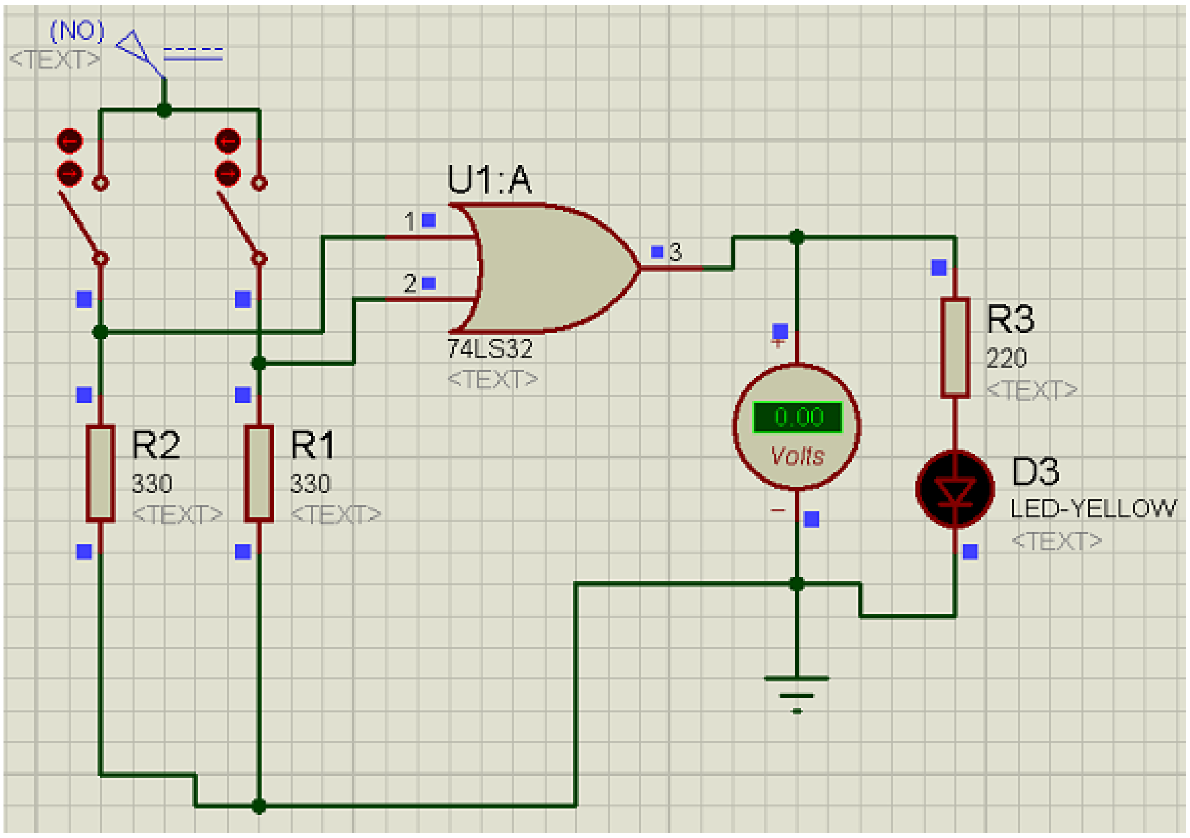
- You can use 74Ls32 Ic as an quad and gate




(NO)
<TEXT>
U1:A
1
2
3
74LS32
<TEXT>
R3
220
<TEXT>
+4.13
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>


(NO)
<TEXT>
U1:A
1
2
3
74LS32
<TEXT>
R3
220
<TEXT>
+4.13
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>

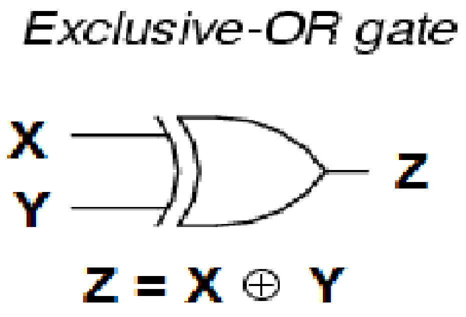

(NO)
<TEXT>
U1:A
1
2
3
74LS32
<TEXT>
R3
220
<TEXT>
+4.13
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>


(NO)
<TEXT>
U1:A
1
2
3
74LS32
<TEXT>
R3
220
<TEXT>
0.00
Volts
D3
LED-YELLOW
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>

XOR Gate

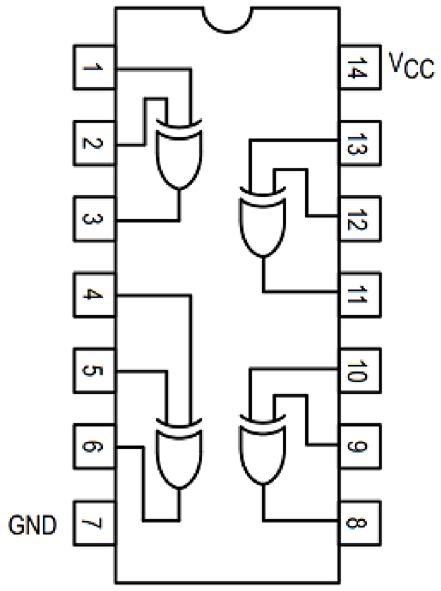
It's work as following :

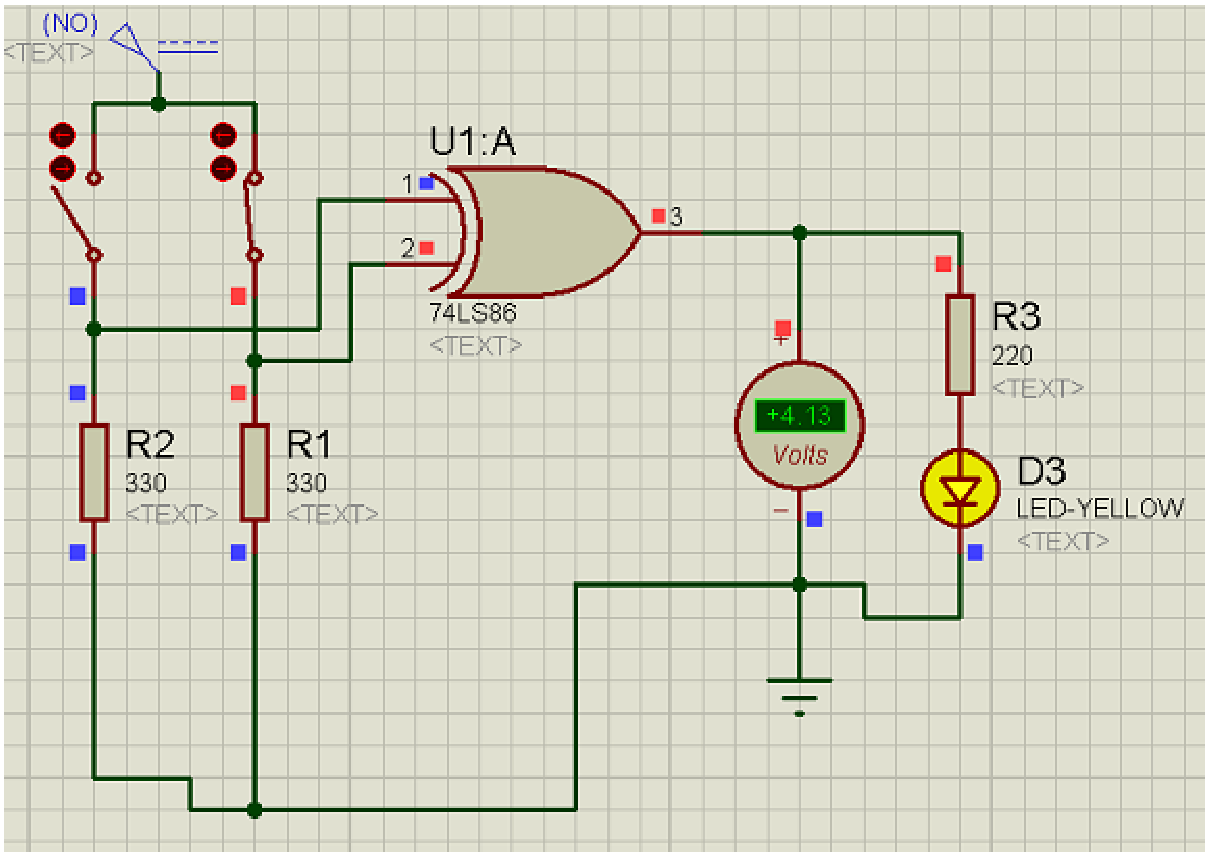
- **If the input is similar , then the output is 0 , else the output is 1**

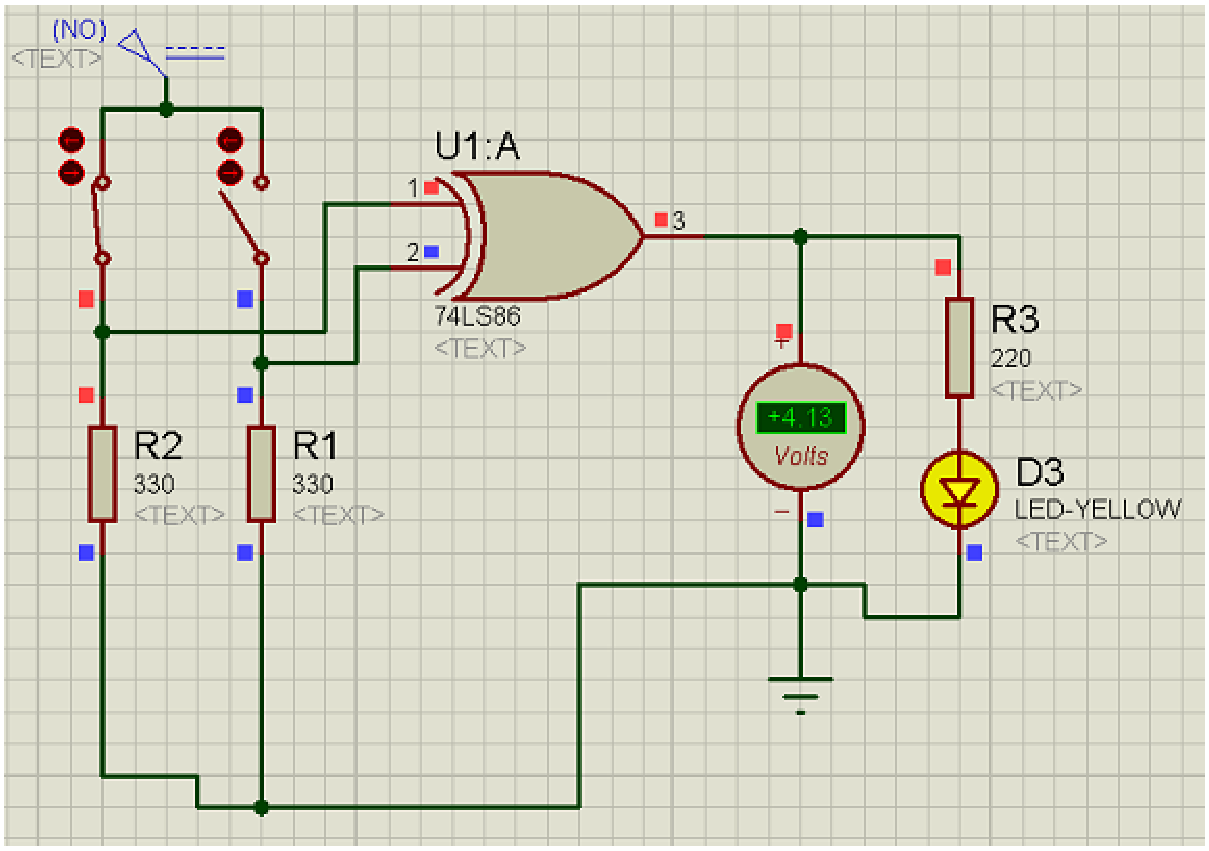


A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

- **You can use 74Ls86 Ic as an quad and gate**



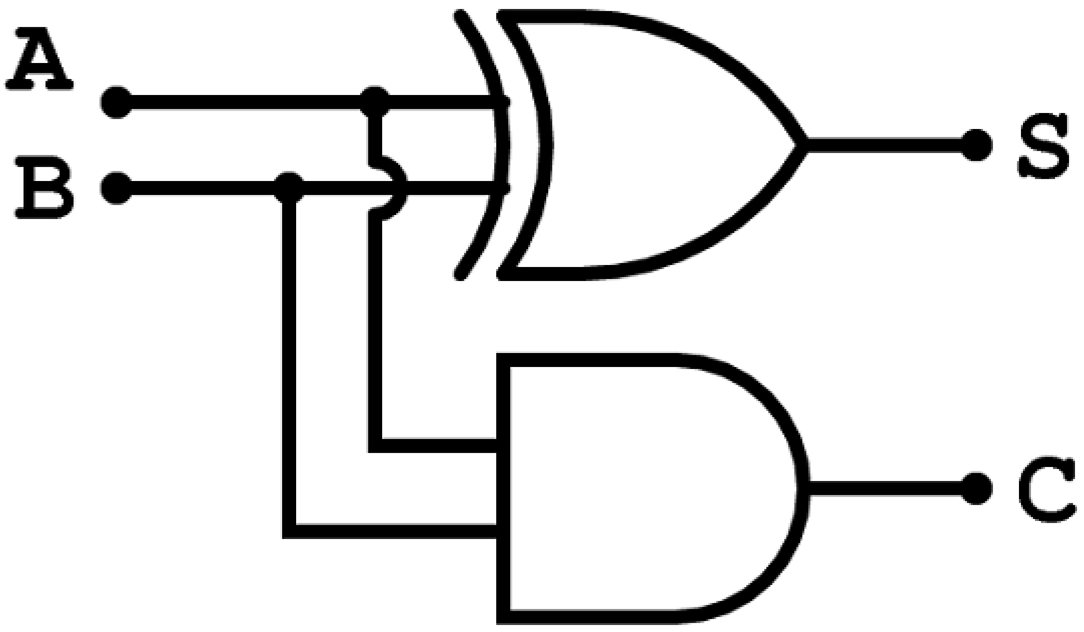

(NO)
<TEXT>
U1:A
1
2
3
74LS86
<TEXT>
R3
220
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
+4.13
Volts
D3
LED-YELLOW
<TEXT>

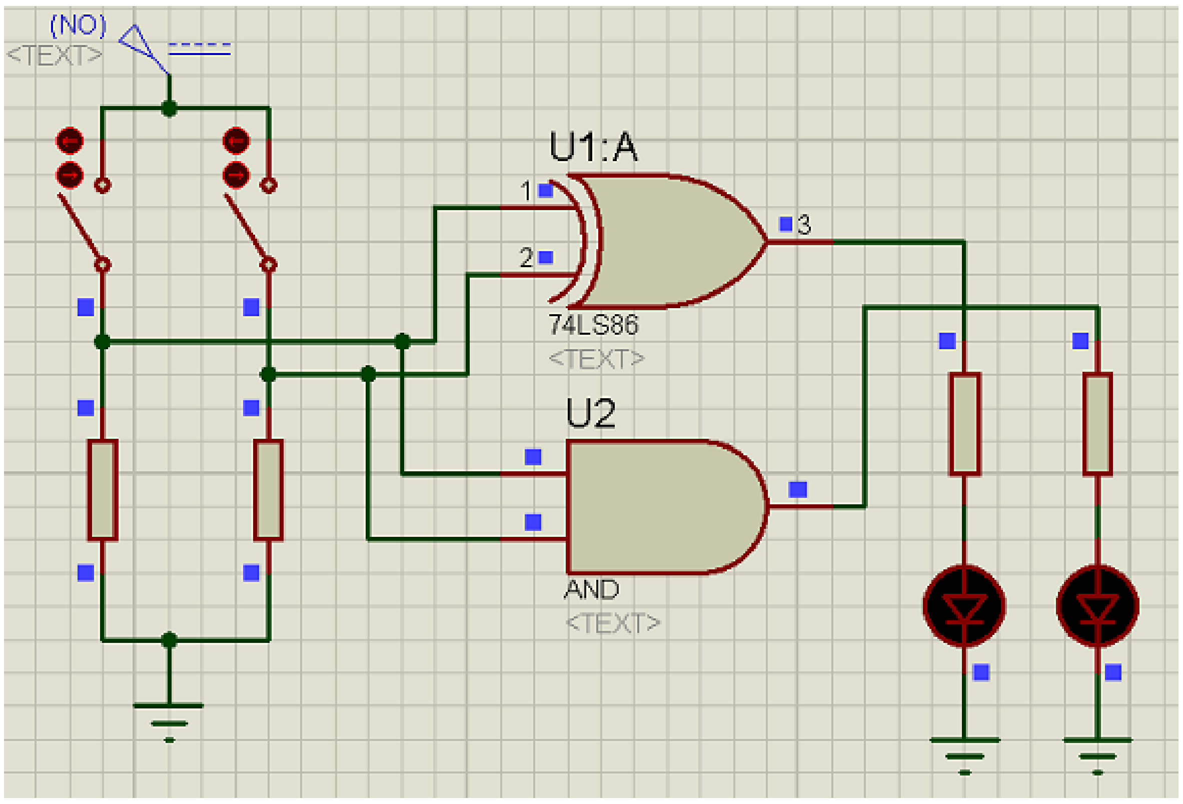

(NO)
<TEXT>
U1:A
1
2
3
74LS86
<TEXT>
R3
220
<TEXT>
R2
330
<TEXT>
R1
330
<TEXT>
+4.13
Volts
D3
LED-YELLOW
<TEXT>

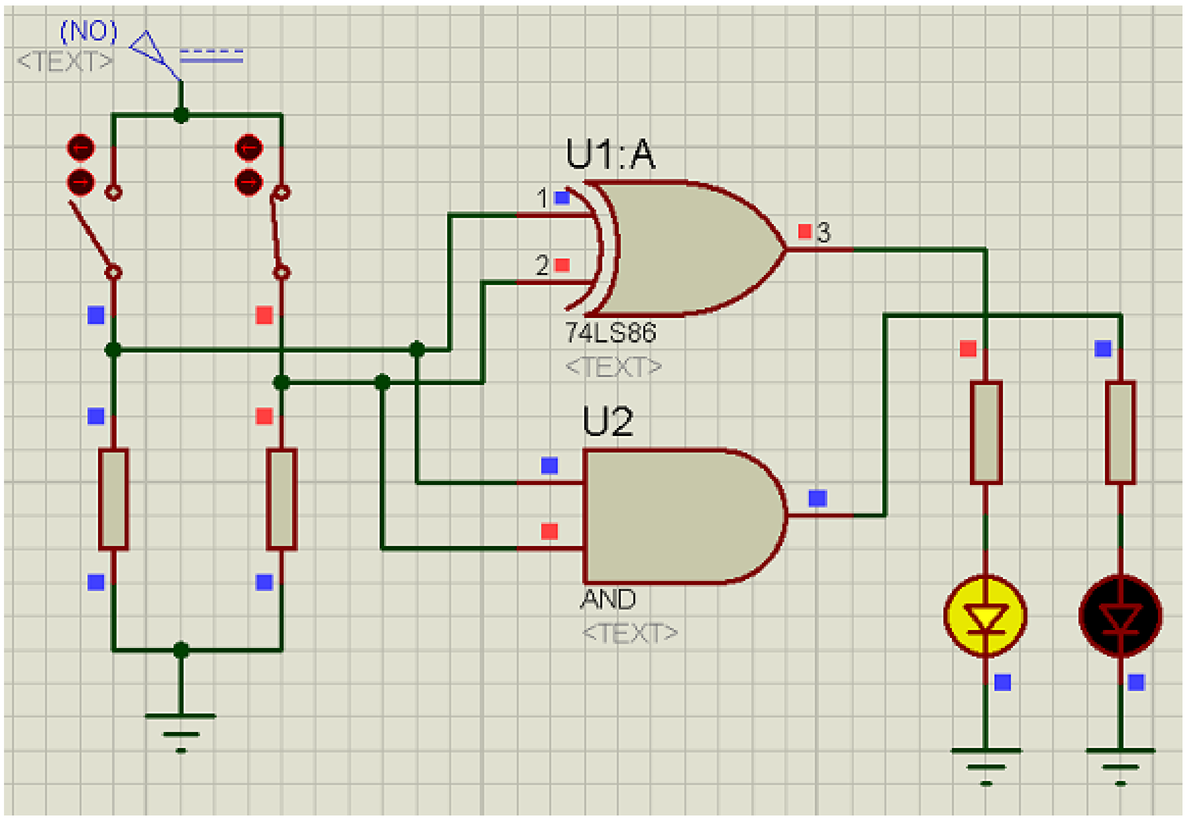
Half Adder Circuit

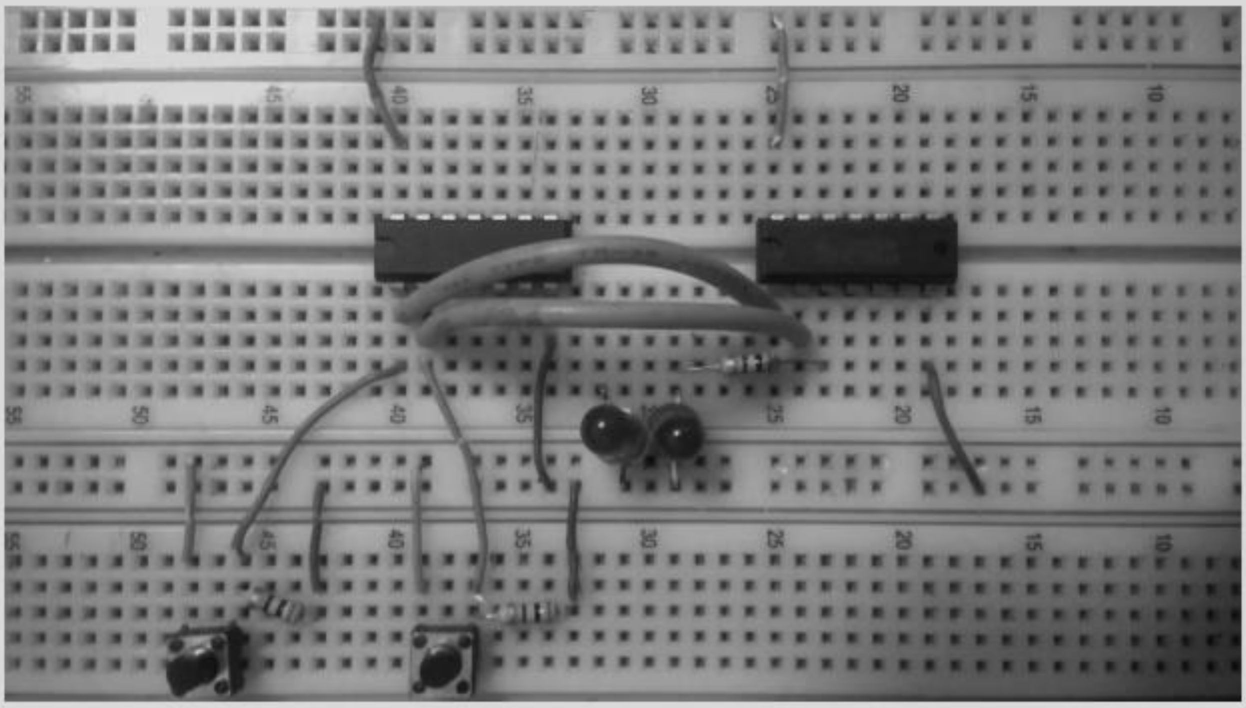
Truth table for Half adder

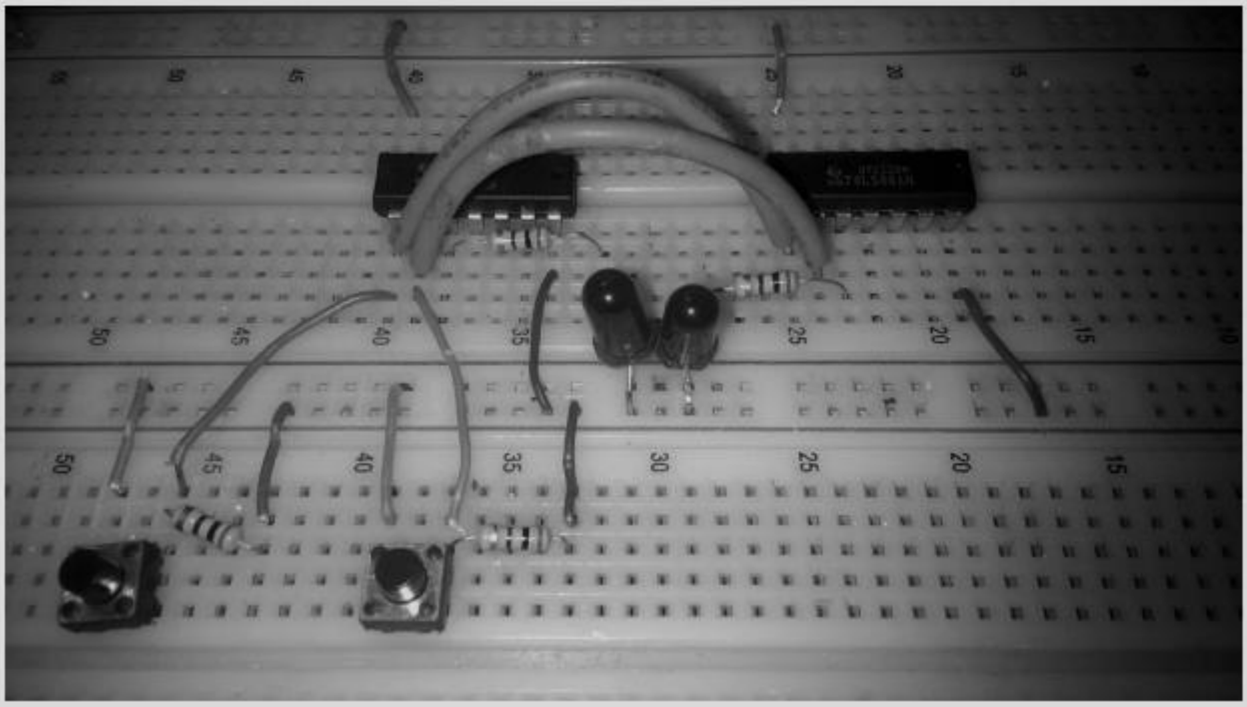
INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

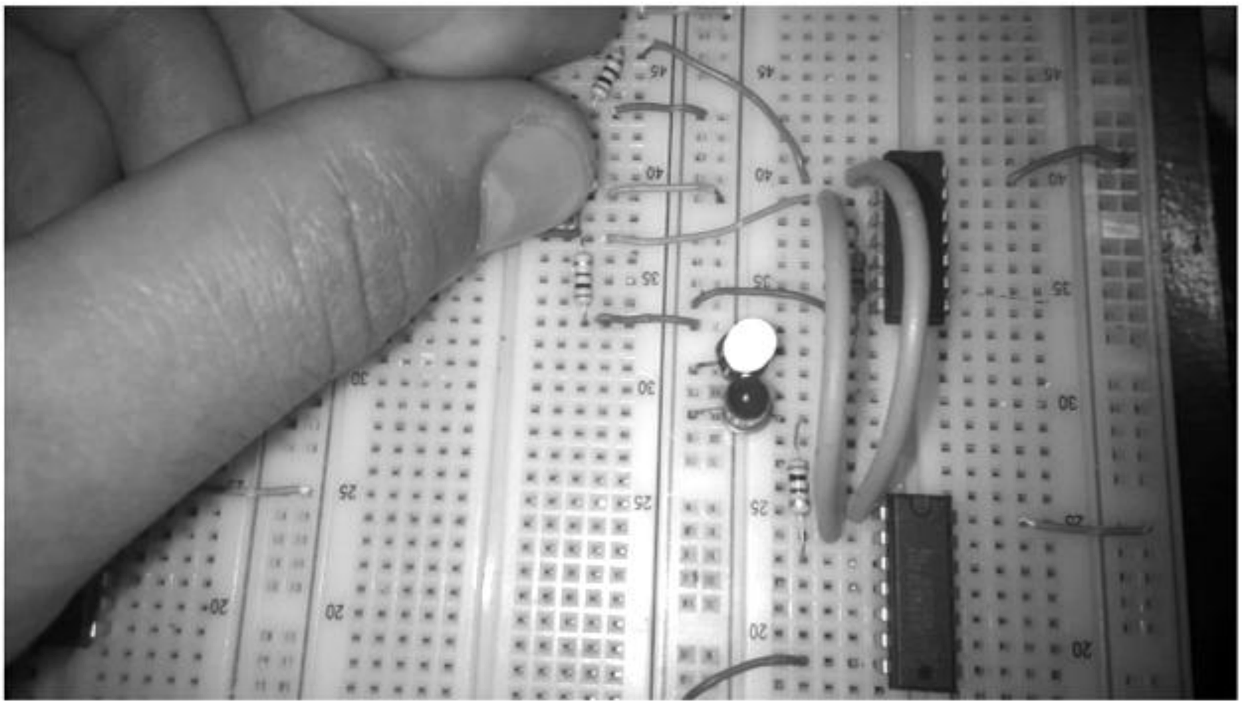



(NO)
<TEXT>
U1:A
1
2
3
74LS86
<TEXT>
U2
AND
<TEXT>


(NO)
<TEXT>
U1:A
1
2
3
74LS86
<TEXT>
U2
AND
<TEXT>

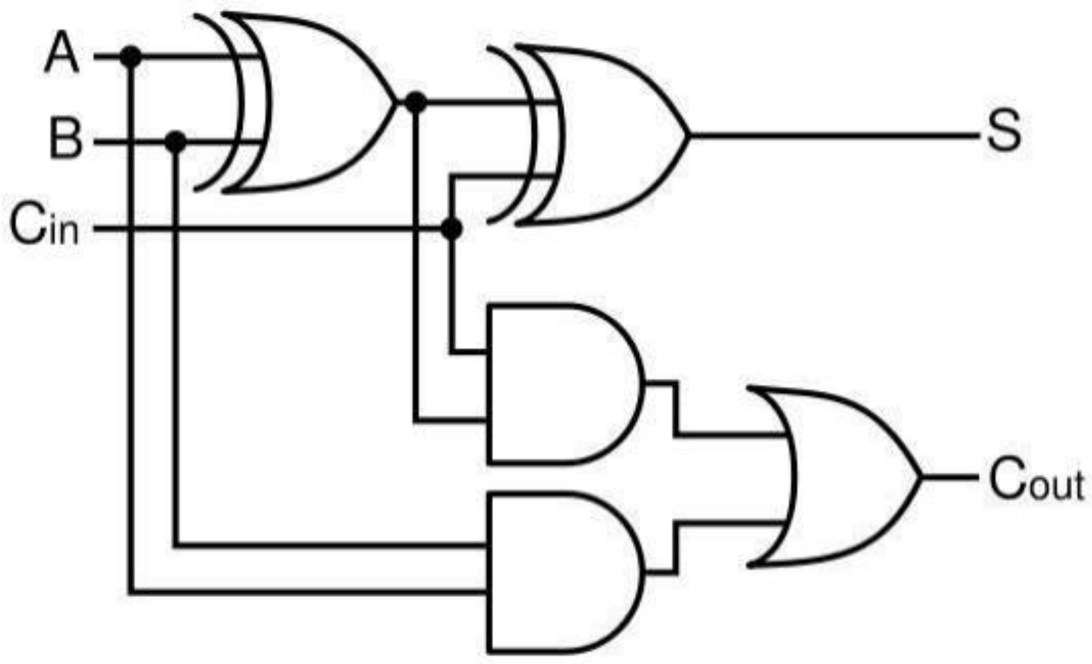


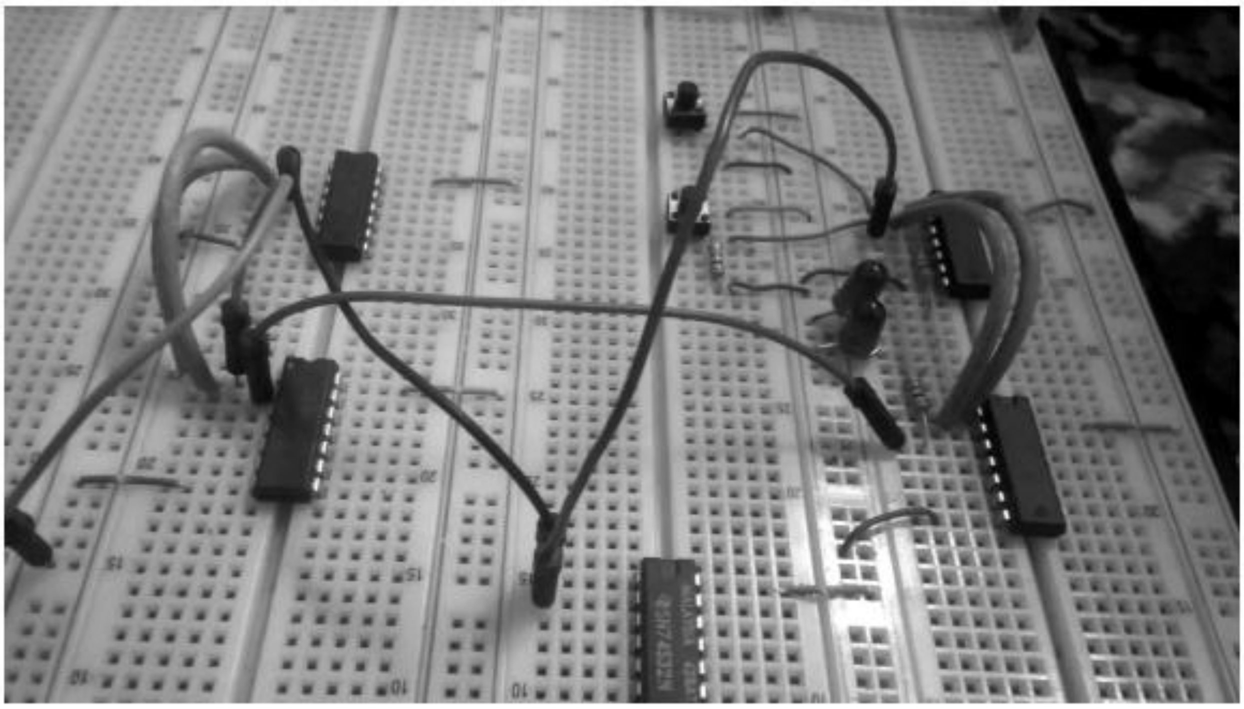


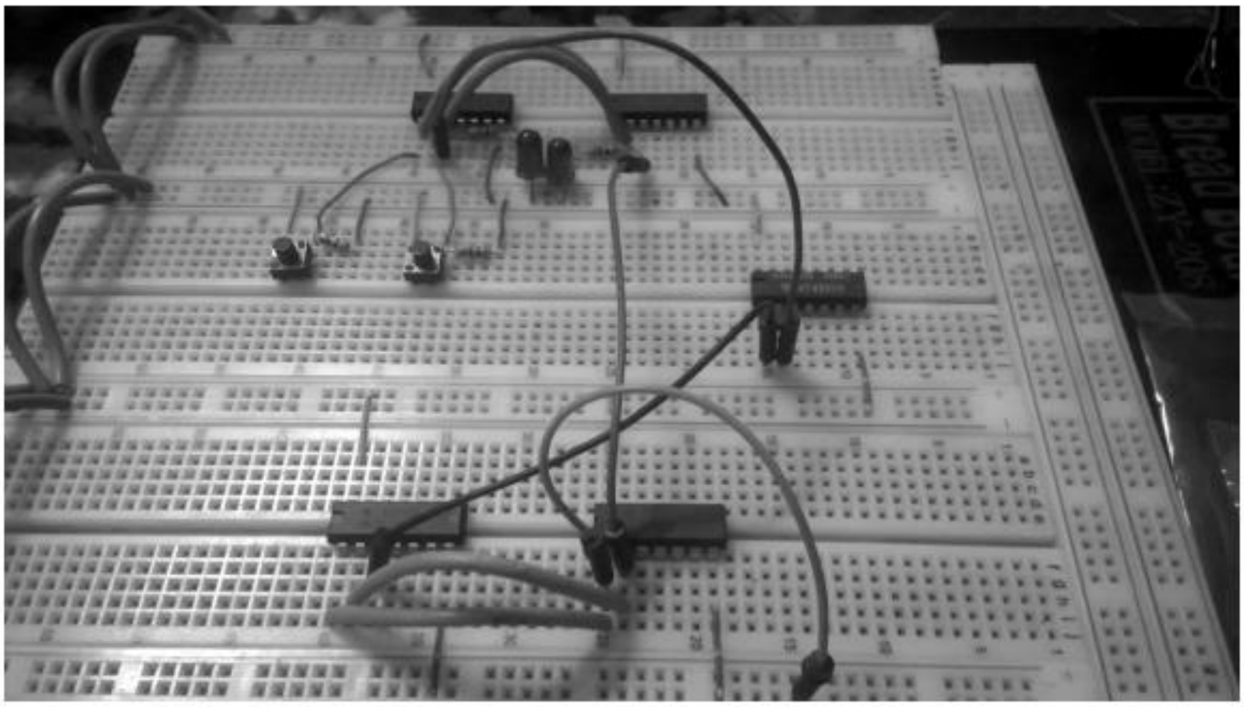


Full Adder

Input			Output	
A	B	Cin	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







LAB SHEET

Student Name

Section

B.N

Experiment Name	**Degree**
BCD Counter Using Proteus + H/W	
Adder And Subtractor Circuits	
Finite State Machines (FSM):	
TOTAL Degree	

LAB (1) ASSIGNMENT

Task (1): Group Project (solve during the week)

- Students are divided in groups of 6 students each.
- Each group should hand in a complete circuit with a report describing the steps followed to implement, design, and wire the circuit.

Group Number	
Student Name	
Section	
B.N	
Grade	

Task (2): Questions (solve during the lab)

1) What is the benefit of capacitors in the counter circuit?

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

What is the function of IC 7490 ?

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

2) Describe Duty cycle?

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

3) What happen if we change the value of capacitor c2

a) C2=1uf

…………………………………………………………………………………
…………………………………………………………………………………
………………………………………………………………………………..

b) C2=1000uf

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

4) What is the purpose of pins 4 and 13 in IC 7490?

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

5) Why we use resistors before LEDs?

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

LAB (2): ASSIGNMENT:

Task (1): Project Task (Software simulation + Hardware)

Design a vending machine that delivers a package of gum after it has received 1.5 pounds in coins. The machine has a single coin slot that accepts half pounds and pounds, one coin at a time. A mechanical sensor indicates to the control whether a pound or a half pound has been inserted into the coin slot. The controller's output causes a single package of gum to be released down to the customer. One further specification: The designed machine does not give change. A customer who pays with two pounds is out a half pound.

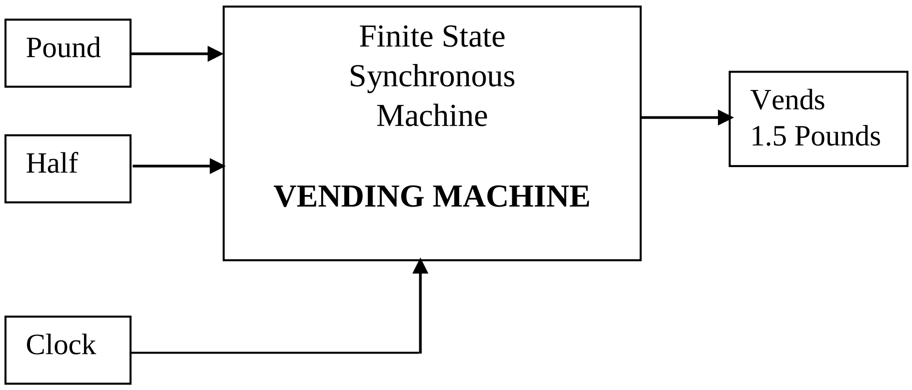


Figure 6: Vending Machine Block Diagram

Hint: Estimating all the possible input sequences that lead to an output of releasing a package of gum (1.5 Pounds) will help you to define the states of the vending finite state machine.

The solution should pass through the following steps:

- Identify system states.
- Draw state diagram.
- Form state transition table.
- Drive state transition equations.
- Draw circuit diagram.
- Simulate your circuit with PROTEUS.

Group Number	
Student Name	
Section	
B.N	
Grade	

<u>Task (2): Questions</u>

1- What is the main difference between mealy model and more model?

………………………………………………………………………………………
………………………………………………………………………………………
……………………………………………

2- What is the main difference between synchronous state machine and asynchronous state machine?

………………………………………………………………………………………
………………………………………………………………………………………
……………………………………………

3- Can ***IC 7474*** be used for asynchronous state machine? And why?

………………………………………………………………………………………
………………………………………………………………………………………
……………………………………………

4- What is the main difference between ***IC 74175*** and ***IC 7474***? Can the first be used for synchronous / asynchronous state machine? And why?

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
……………………………………………

5- State main steps for design state machine.

………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
………………………………………………………………………………………
……………………………………………

LAB (3) ASSIGNMENT

Task (1): project task

Design a full adder circuit that can add two digital numbers, each number has only one bit?

Group Number	
Student Name	
Section	
B.N	
Grade	

Task(2): Question

1. What is the meaning of zero and one in logic circuit?

……………………………………………………………………………
…………………………………………………………………………..

2. Explain briefly how to get the equation form a truth table? Give a simple example?

……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………

……………………………………………………………………………
……………………………………………………………………………

3. Explain briefly the theory of subtracting digital numbers?

……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………

4. What is the difference between TTL and CMOS? (search)

……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………

5. Draw the internal structure for AND, OR, Not Gates?

……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………
……………………………………………………………………………