DIGITAL ELECTRONICS LAB MANUAL

NAME USAMA ATTA ABBASI
ROLL NO 20ES070
BATCH 20 ES
DEPARTMENT OF ELECTRONIC ENGG.

PREPARED BY:

DR.KHALIL DAYO

List Of Experiments

1. Arithmetic Circuit- construction and testing of
   a. Half adder and Full adder.
   b. Half subtractor and Full subtractor.

2. Combinational logic circuit design using 74xxICs.

3. Encoders and Decoders.

4. Multiplexer and Demultiplexer.

5. Study of Arithmetic Logic Unit(ALU) using IC 74181.

6. Construction of 1- bit comparator using 74xxICs and study of 4-bit comparator IC 7485.

7. code converters – Binary to gray and Gray to binary.

8. Verification of basic flip flops using 74xxICs and master-slave JK flip-flop using IC 7476

9. Asynchronous counter design and Mod-n counter.

10. 3-Bit synchronous counter design

11. Shift register- SIPO/SISO & PISO/PIPO.

12. Study of RAM.

The Laboratory Notebook:

Each student must have their own laboratory notebook. All pre-lab exercises and laboratory reports are to be entered into your notebook.

Your notebook must be clearly labelled on the cover with the following information:

Module: Digital Electronics -
Name: USAMA ATTA ABBASI
Roll no: 20ES070
Class: DIGITAL ELECTRONICS LAB
Lab Group: 1

Introduction

There are 3 hours allocated to a laboratory session in Digital Electronics. It is a necessary part of the course at which attendance is compulsory.

Here are some guidelines to help you perform the experiments and to submit the reports:

1. Read all instructions carefully and carry them all out.
2. Ask a lab person if you are unsure of anything.
3. Record actual results (comment on them if they are unexpected!)
4. Write up full and suitable conclusions for each experiment.
5. If you have any doubt about the safety of any procedure, contact the lab supervisor beforehand.
6. **THINK** about what you are doing!

The Breadboard

The breadboard consists of two terminal strips and two bus strips (often broken in the centre). Each bus strip has two rows of contacts. Each of the two rows of contacts are a node. That is, each contact along a row on a bus strip is connected together (inside the breadboard). Bus strips are used primarily for power supply connections, but are also used for any node requiring a large number of connections. Each terminal strip has 60 rows and 5 columns of contacts on each side of the centre gap. Each row of 5 contacts is a node.

You will build your circuits on the terminal strips by inserting the leads of circuit components into the contact receptacles and making connections with 22-26 gauge wire. There are wire cutter/strippers and a spool of wire in the lab. It is a good practice to wire +5V and 0V power supply connections to separate bus strips.

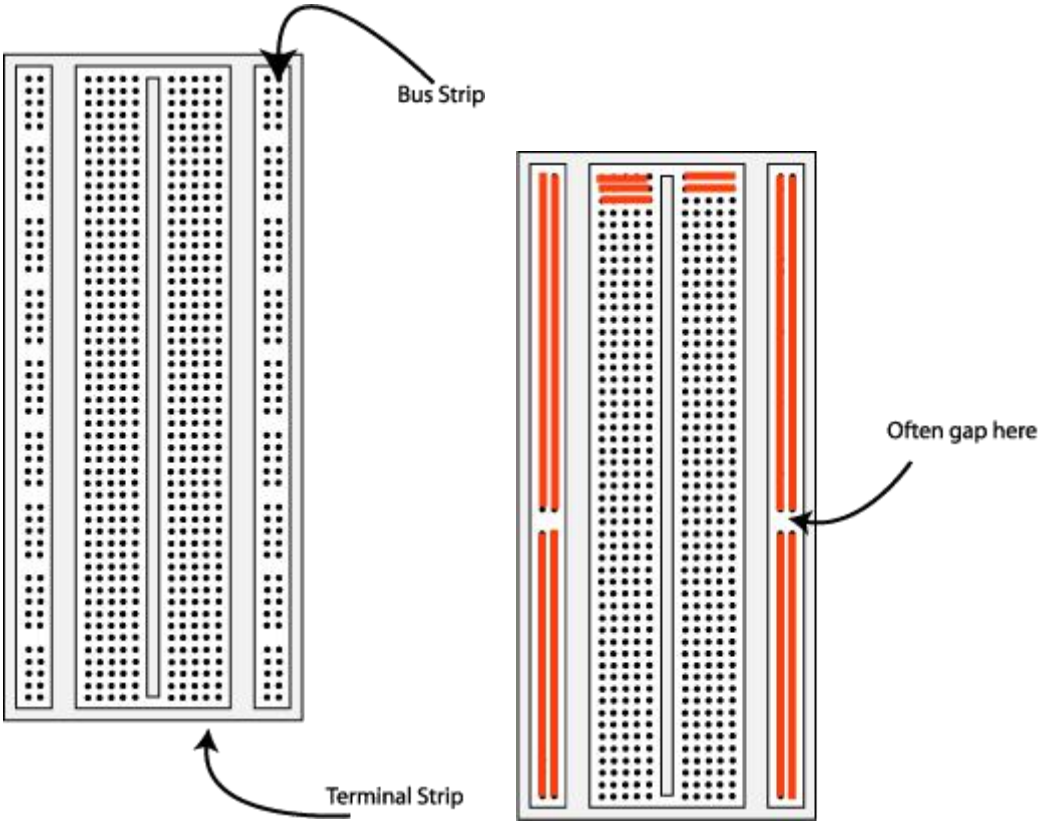


Fig 1. The breadboard. The lines indicate connected holes.

The 5V supply **MUST NOT BE EXCEEDED** since this will damage the ICs (Integrated circuits) used during the experiments. Incorrect connection

of power to the ICs could result in them exploding or becoming very hot - with the **possible serious injury occurring to the people working on the experiment! Ensure that the power supply polarity and all components and connections are correct before switching on power .**

Building the Circuit

Throughout these experiments we will use TTL chips to build circuits. The steps for wiring a circuit should be completed in the order described below:

1. Turn the power (Trainer Kit) off before you build anything!
2. Make sure the power is off before you build anything!
3. Connect the +5V and ground (GND) leads of the power supply to the power and ground bus strips on your breadboard.
4. Plug the chips you will be using into the breadboard. Point all the chips in the same direction with pin 1 at the upper-left corner. (Pin 1 is often identified by a dot or a notch next to it on the chip package)
5. Connect +5V and GND pins of each chip to the power and ground bus strips on the breadboard.
6. Select a connection on your schematic and place a piece of hook-up wire between corresponding pins of the chips on your breadboard. It is better to make the short connections before the longer ones. Mark each connection on your schematic as you go, so as not to try to make the same connection again at a later stage.
7. Get one of your group members to check the connections, **before you turn the power on**.
8. If an error is made and is not spotted before you turn the power on. Turn the power off immediately before you begin to rewire the circuit.
9. At the end of the laboratory session, collect you hook-up wires, chips and all equipment and return them to the demonstrator.
10. Tidy the area that you were working in and leave it in the same condition as it was before you started.

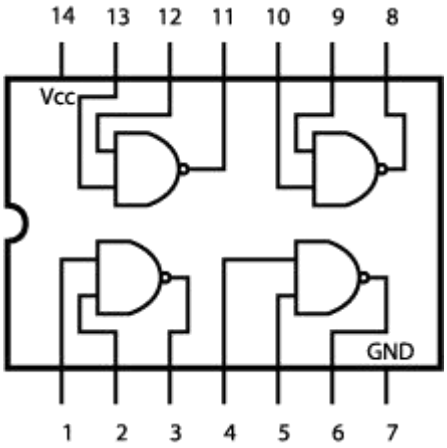
Common Causes of Problems

1. Not connecting the ground and/or power pins for all chips.
2. Not turning on the power supply before checking the operation of the circuit.
3. Leaving out wires.
4. Plugging wires into the wrong holes.
5. Driving a single gate input with the outputs of two or more gates
6. Modifying the circuit with the power on.

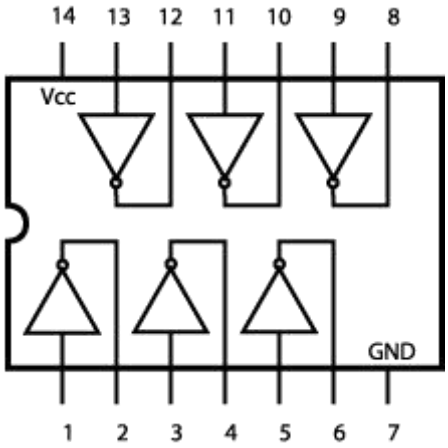
In all experiments, you will be expected to obtain all instruments, leads, components at the start of the experiment and return them to their proper place after you have finished the experiment. Please inform the demonstrator or technician if you locate faulty equipment. If you damage a chip, inform a demonstrator, don't put it back in the box of chips for somebody else to use.

Example Implementation of a Logic Circuit

Build a circuit to implement the Boolean function F = /(/A./B), please note that the notation **/A** refers to $\overline{A}$. You should use that notation during the write-up of your laboratory experiments.



Quad 2 Input 7400



Hex 7404 Inverter

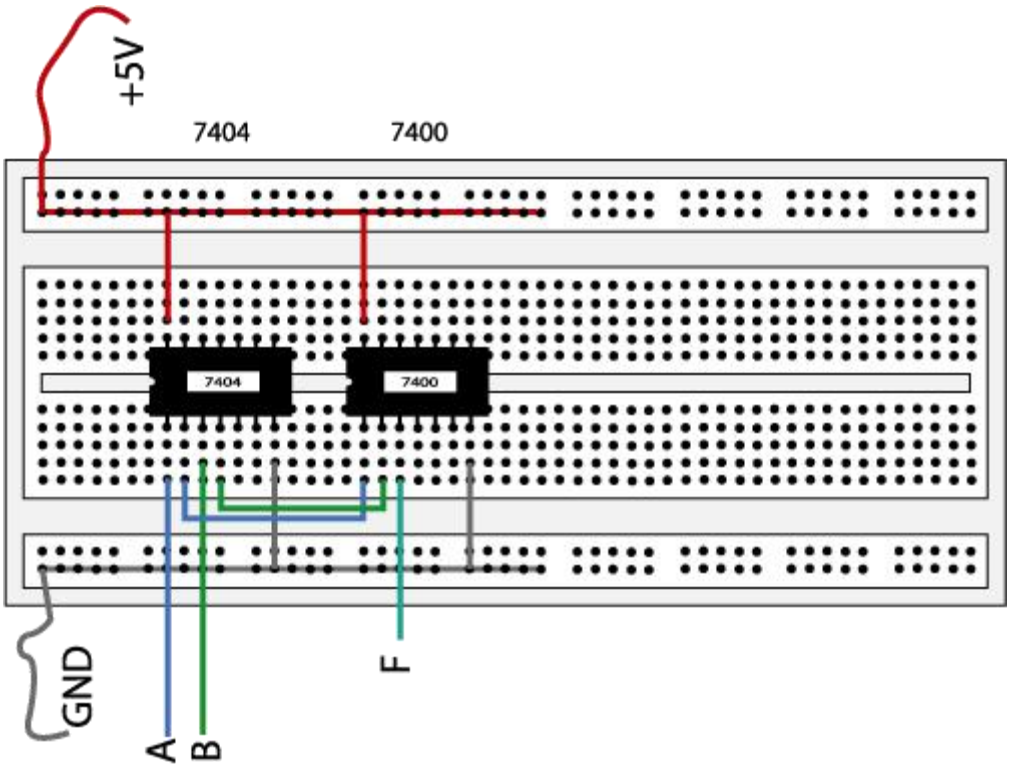
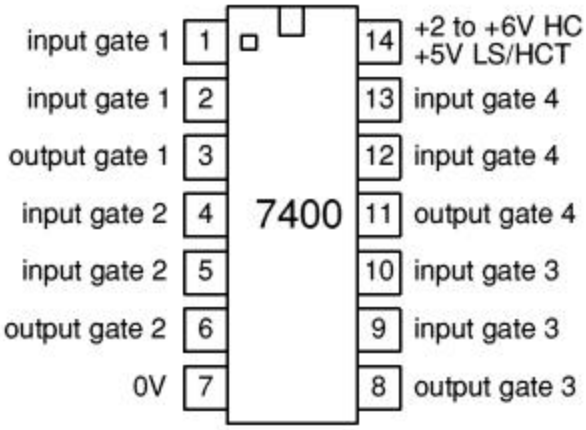


Fig 2. The complete designed and connected circuit

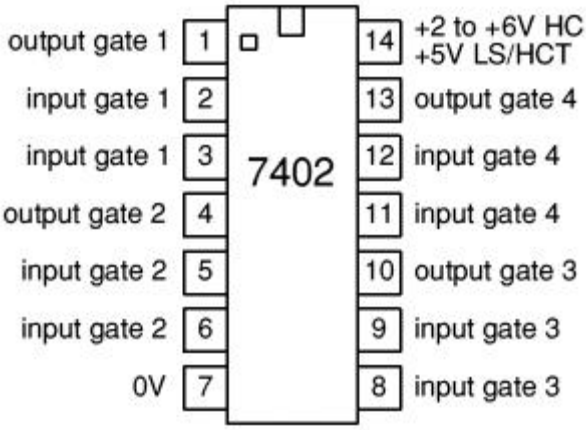
Sometimes the chip manufacturer may denote the first pin by a small indented circle above the first pin of the chip. Place your chips in the same direction, to save confusion at a later stage. Remember that you must connect power to the chips to get them to work.

Useful IC Pin details

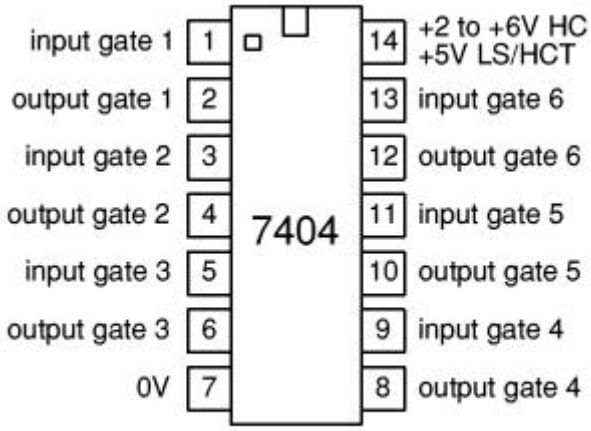
7400(NAND)



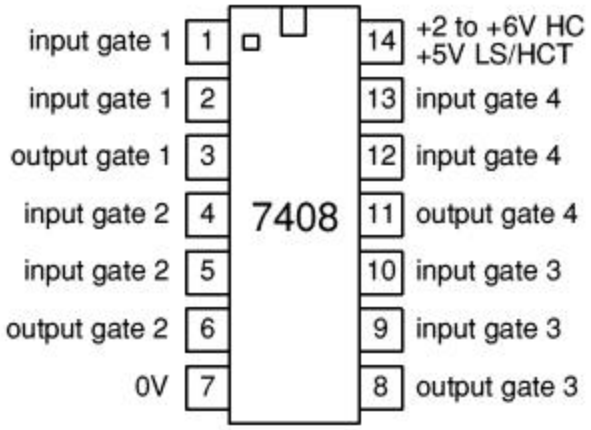
7402(NOR)



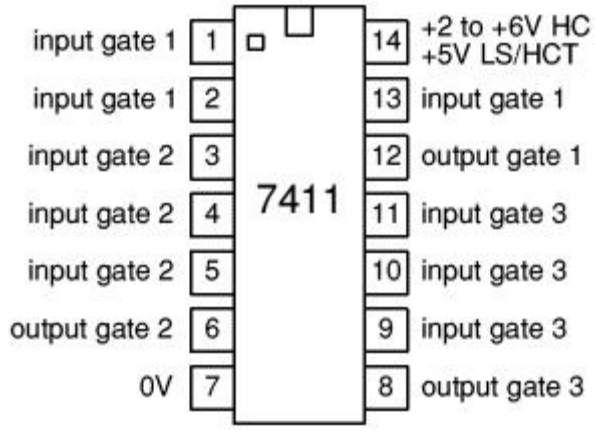
7404(NOT)



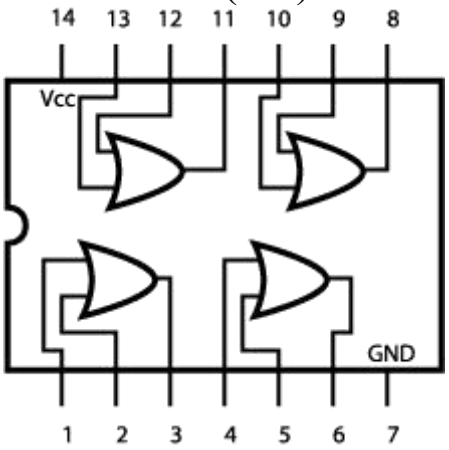
7408(AND)



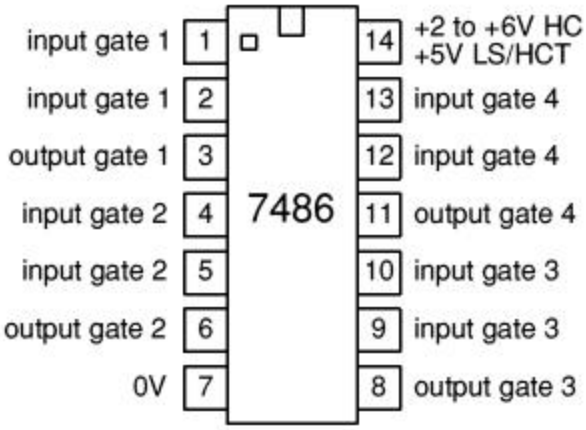
7411(3-i/p AND)

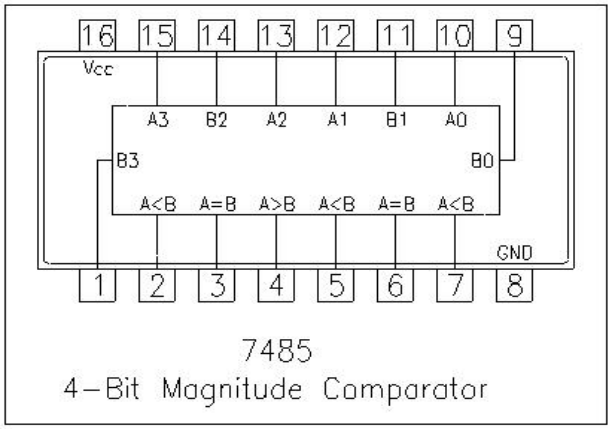


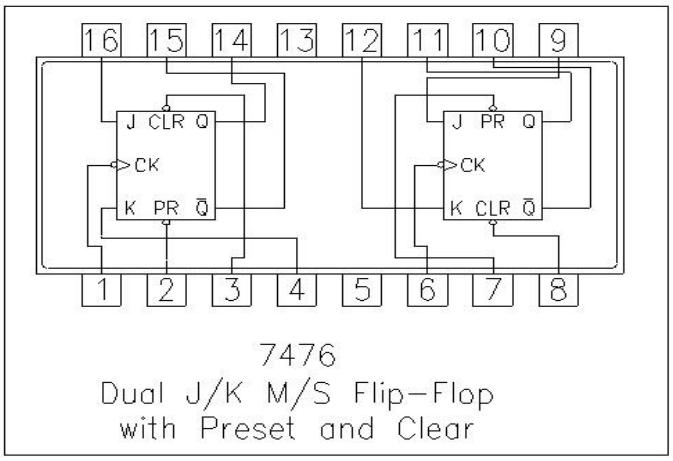
7432(OR)



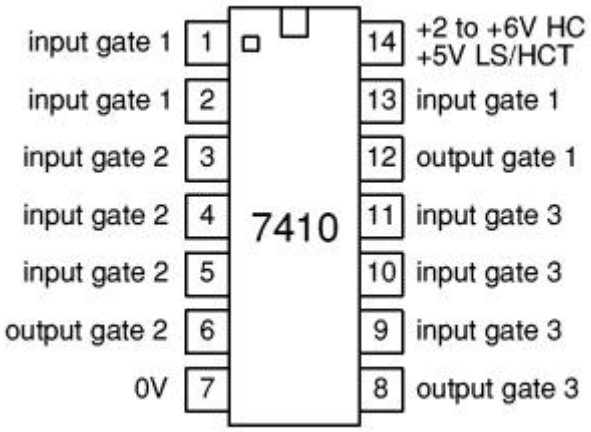
7486(EX-OR)



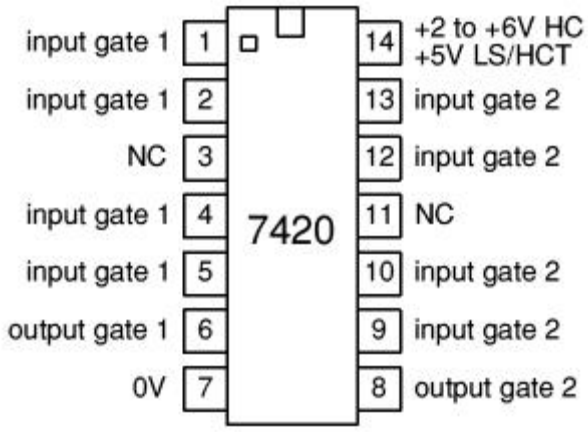




7410(3-i/p NAND)



7420(4-i/p NAND)



Mehran University of Engineering & Technology, Jamshoro

Department of Electronic Engineering

DIGITAL ELECTRONICS

Lab NO.1

Switching Analogy Of Logic Gates

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: Implantation of Logic gates with switches using Multisim platform.

EQUIPMENT:

- Power supply unit (EV)
- LED

COMPONENTS:

- Switches
- Resistors

Digital logic gates are electronic components:

- serve as the basic building blocks of any digital system irrespective of its complexity.
- operate with voltages of two logic levels namely Logic Low '0' and Logic High '1'.
- results in a particular output after implementing its logic on the input signals.

These logic gates are divided into three categories:

1. Basic Gates:

All digital systems can be constructed by only two basic logic gates:

- NOT Gate
- OR Gate
- AND Gate

2. Combinational Logic Gate

Gates constructed by combination of two or more basic gates are called combinational logic gates. These are:

i. Universal Logic Gate:

These gates listed below are called universal logic gates because using any of these gates alone it is possible to implement any logic operation by connecting them together in various combinations:

- NOR Gate
- NAND Gate

ii. Athematic Logic Gates

It is used in simple digital addition circuits which calculate the sum and carry of two (half-adder) or three (full adder) bit numbers.

- XOR Gate
- XNOR Gate

$\Rightarrow$ Digital electronics depends on the actions of these seven types of logic gates.

1. NOT Gate/Inverter

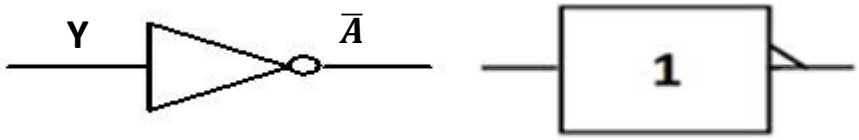
CHARACTERISTICS:

- Perform on a single input variable.

- The Boolean operation for the NOT operation is:

  Y = $\overline{A}$ ('-' overbar represents the NOT operation)

SYMBOL:

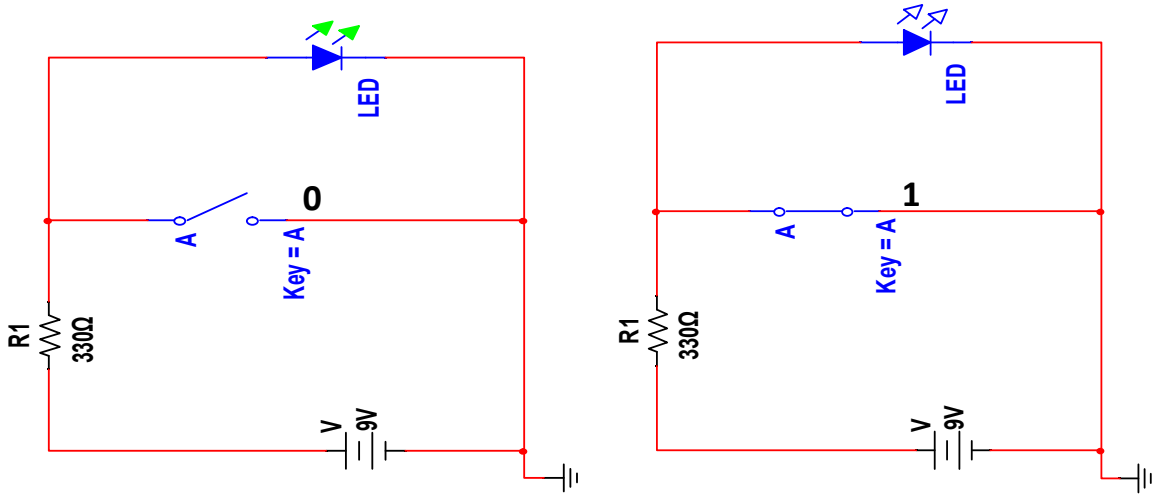


TRUTH TABLE:

INPUT	OUTPUT
Y	**$\overline{A}$**
1	0
0	1

SAMULATED CIRCUIT:

To implement the switch circuit for NOT gate, the switch circuit is connected in parallel to source. When switch is open the current flows through LED showing logic High n when switch is close shows the logic Low.



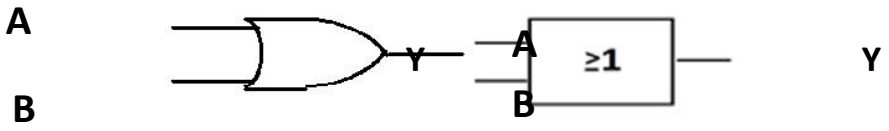
2. OR Gate

CHARACTERISTICS:

- The OR operation produce result (output High) of 1 wherever any input is '1', otherwise the result (output Low) is '0'.
- The Boolean expression for the OR operation is:

  Y = A + B ('+' represents the OR operation)

SYMBOL:

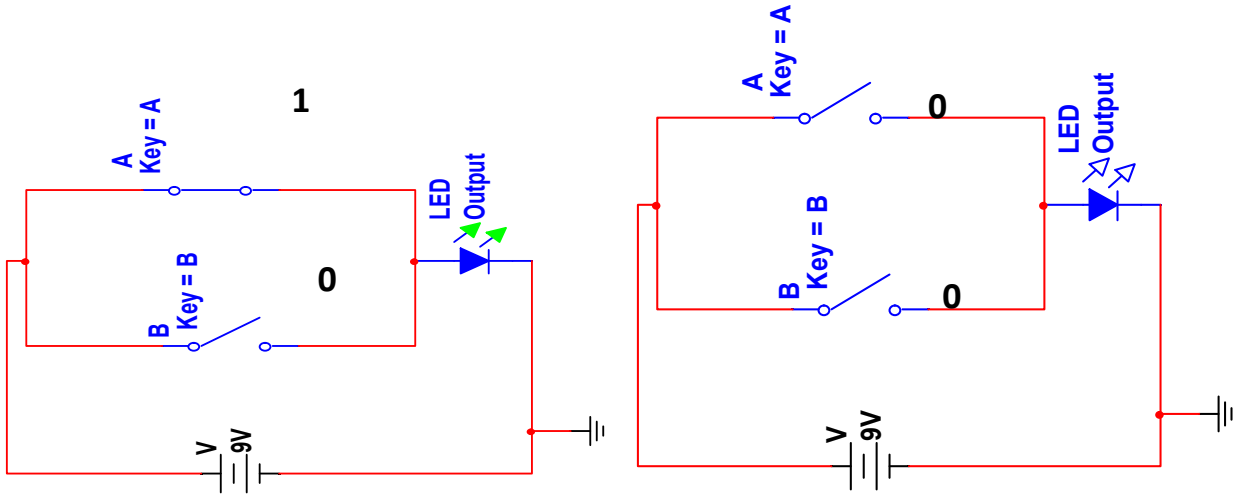


TRUTH TABLE:

INPUT		OUTPUT
A	**B**	**Y**
0	0	0
0	1	1
1	1	1
1	0	1

SIMULATED CIRCUIT:

To implement the switch circuit for two input OR gate, the switch circuit is connected in parallel to source n LED. When both switches are open the current does not flow through LED showing logic Low n when any of one switch is close or both switches closed current flows through LED showing the logic High.



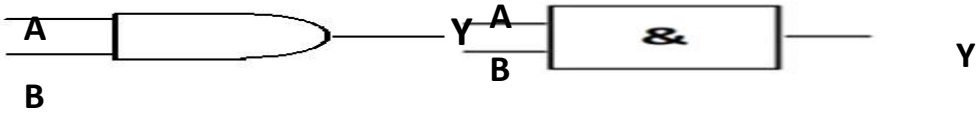
3. AND Gate

CHARACTERISTICS:

- The AND operation produce result (output High) of '1' wherever all input are '1 or high', otherwise the result (output Low) is '0'.
- The Boolean expression for the AND operation is:

  Y = A.B ('.' represents the AND operation)

SYMBOL:

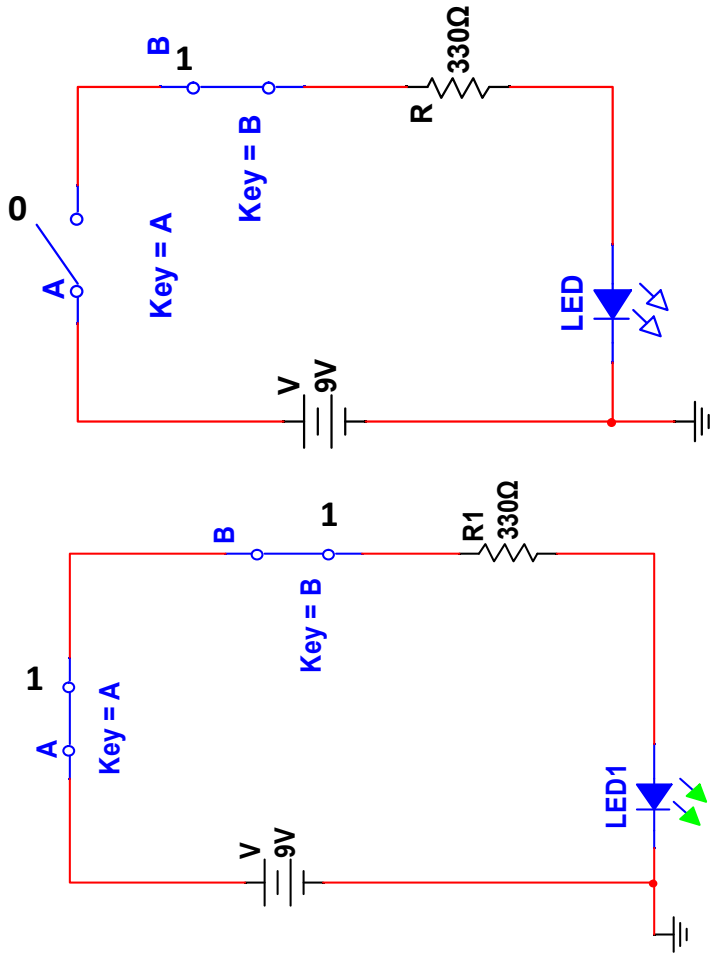


TRUTH TABLE:

INPUT		OUTPUT
A	**B**	**Y**
0	0	0
0	1	0
1	1	1
1	0	0

SIMULATED CIRCUIT:

To implement the switch circuit for two input AND gate, the switch circuit is connected in series to source n LED. When both switches are close the current flows through LED showing logic High n when any of one switch is open or both switches are open it is showing the logic Low.



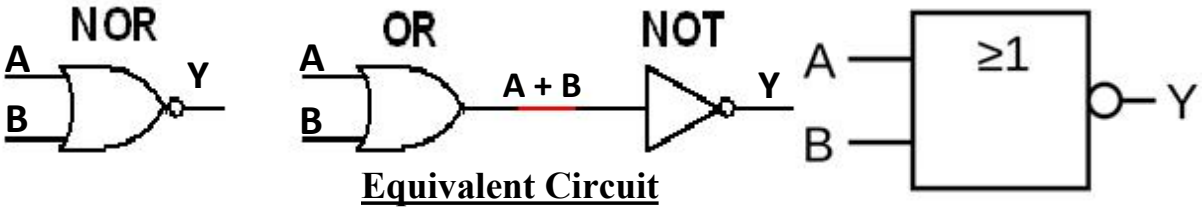
4. NOR Gate

CHARACTERISTICS:

- NOR gate is equal to an OR gate followed by a NOT gate.
- A NOR operation produces the result (output High) '1' if all the operands are '0', otherwise the result is (output Low) '0'.
- The Boolean operation for the NOR operation is:

  Y = $\overline{A + B}$ ('+' represents the OR operation & '-' overbar represents the NOT operation)

SYMBOL and EQUIVALENT CIRCUIT:

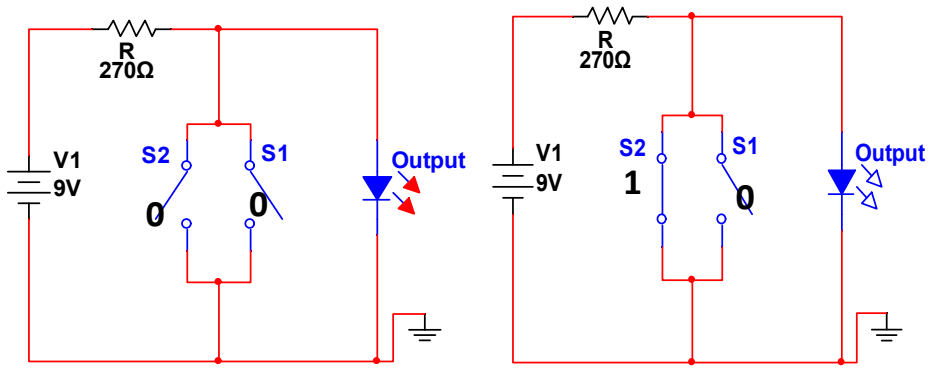


Equivalent Circuit

TRUTH TABLE:

INPUT		OUTPUT
A	B	Y
0	0	1
0	1	0
1	1	0
1	0	0

SIMULATED CIRCUIT:

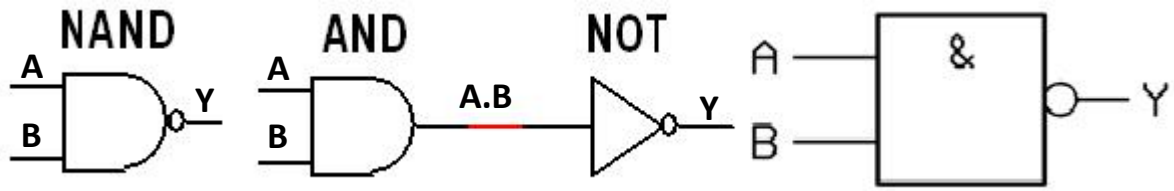


5. NAND Gate CHARACTERISTICS:

- NAND gate is equal to an AND gate followed by a NOT gate.
- A NAND operation produces the result (output Low) '0' when all operands are '1', otherwise the result is (output High) '1'.
- The Boolean operation for the NAND operation is:

  Y = $\overline{A \,.\, B}$ ('.' represents the AND operation & '-' overbar represents the NOT operation)

SYMBOL and EQUIVALENT CIRCUIT:

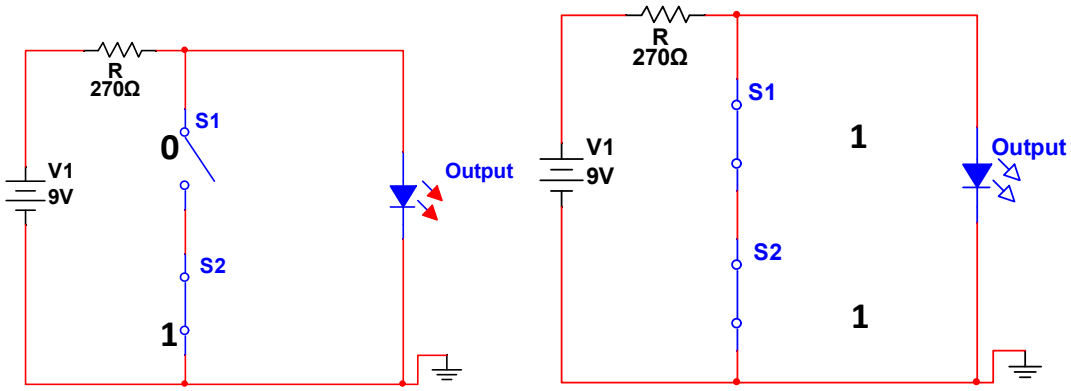


Equivalent Circuit

TRUTH TABLE:

INPUT		OUTPUT
A	**B**	**Y**
0	0	1
0	1	1
1	1	0
1	0	1

SIMULATED CIRCUIT:



Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.2

GATES LOGIC VARIFICATION and APPLICATION IN DIGITAL SYSTEM

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: Verification of AND, OR, NOT, NAND, NOR, XOR and XNOR gate and theire application in digital system using E-18 KIT/ MultiSim platform.

EQUIPMENT:

- Power supply unit (EV)

$\Rightarrow$ Note: in simulation Digital Switches are used.

- Green Probe

COMPONENTS:

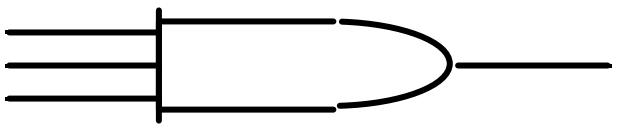
AND Gate	NOR Gate
OR Gate	XOR Gate
Not Gate	XNOR Gate
NAND Gate	

1. AND Gate

CHARACTERISTICS:

- The AND operation produce result (output High) of '1' wherever all input are '1 or high', otherwise the result (output Low) is '0'.
- The Boolean expression for the AND operation is:

Y = A.B ('.' represents the AND operation)**:**

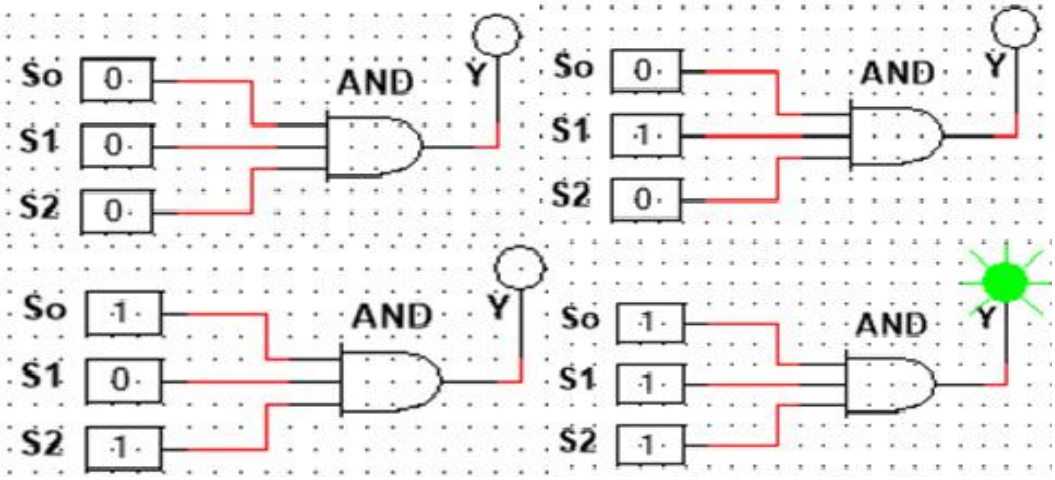


TRUTH TABLE for 3 Input:

Input			**Output**
S_2	S_1	S_0	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0

1	1	1	1

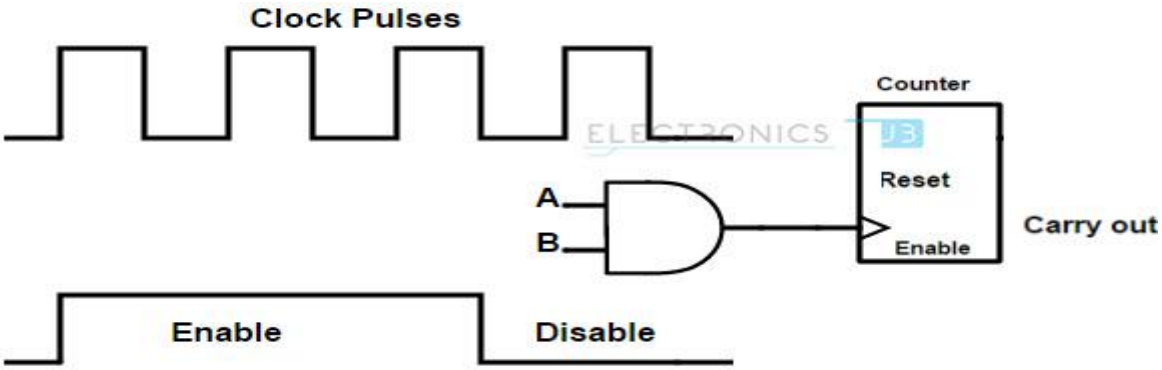
Varification:



AND gate Applications

Logic AND gate is used in many of the application s in our day to day life. Some of them are explained below.

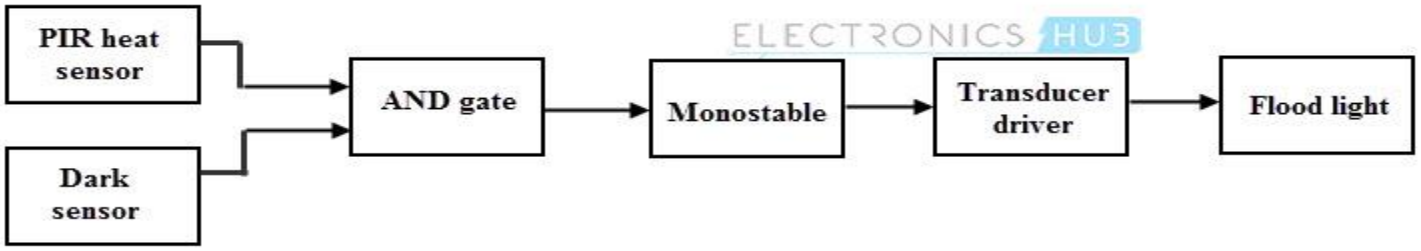
1. AND gate is used as Enable and Disable purpose on counter devices. If we observe the below circuit, when the counter starts counting from 0 to 100. When a counter receives the clock signal, then it increments its count by 1.



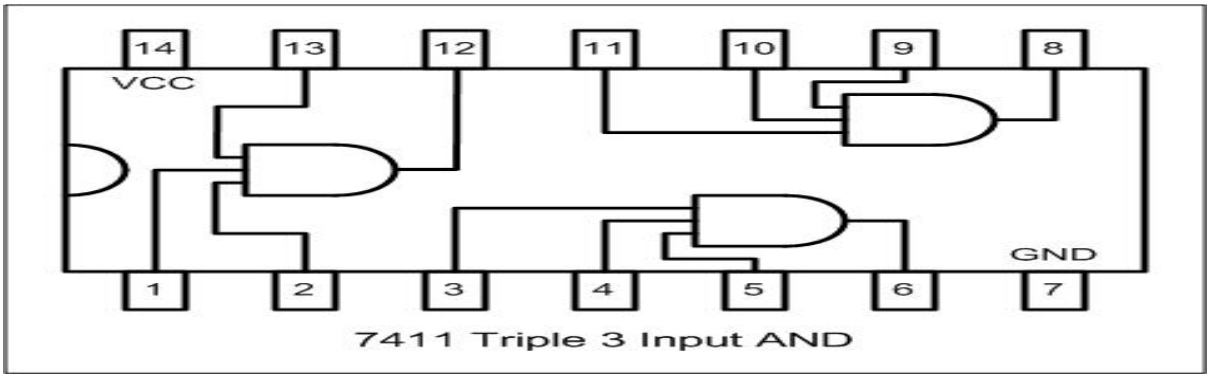
2. The logic AND gate is used in some sort of security devices like garden flood-lights and security lights etc. They have a heat- radiation sensitive device called “Passive Infra- Red device (PIR)”. So when a hot object such as intruder (unauthorized entries like neighbor’s pets) is detected by the device, the heat sensor produces a high voltage making it to set in logic 1.

As the light of flood- light did not clearly visible in daytime these devises comes in use, when the surroundings /atmosphere is dark. It comes to ON state when the heat sensor is triggered. The block diagrams this security system with AND gate is shown

below.



3-input AND gate IC



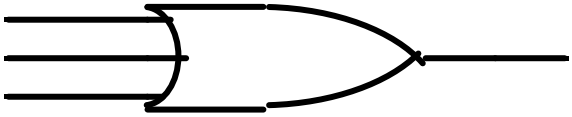
2. OR Gate

CHARACTERISTICS:

- The OR operation produce result (output High) of 1 wherever any input is '1', otherwise the result (output Low) is '0'.
- The Boolean expression for the OR operation is:

  Y = A + B ('+' represents the OR operation)

SYMBOL:

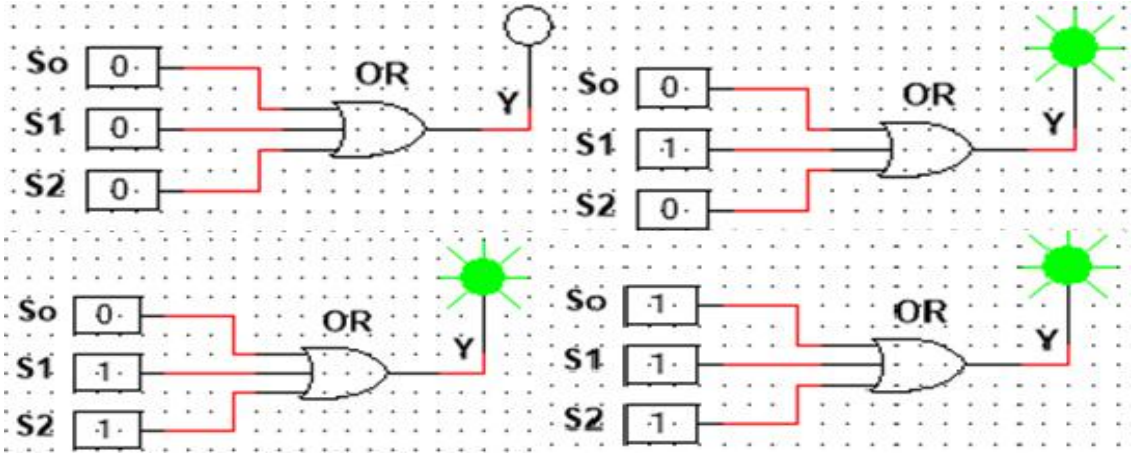


TRUTH TABLE for 3 Input:

Input			Output
S_2	S_1	S_0	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1

1	0	1	1
1	1	0	1
1	1	1	1

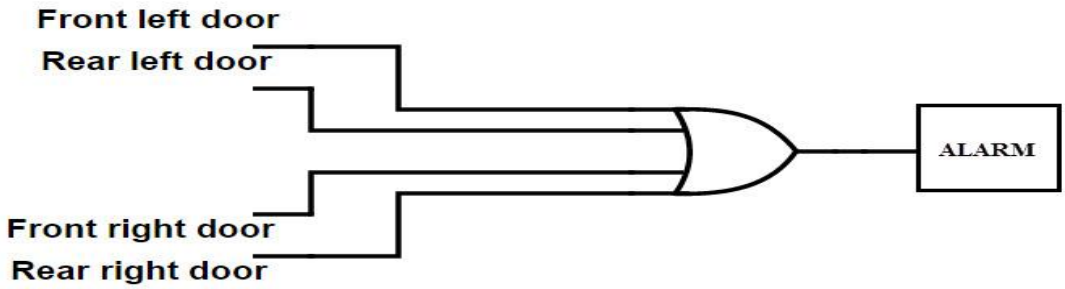
Varification:



OR gate Applications

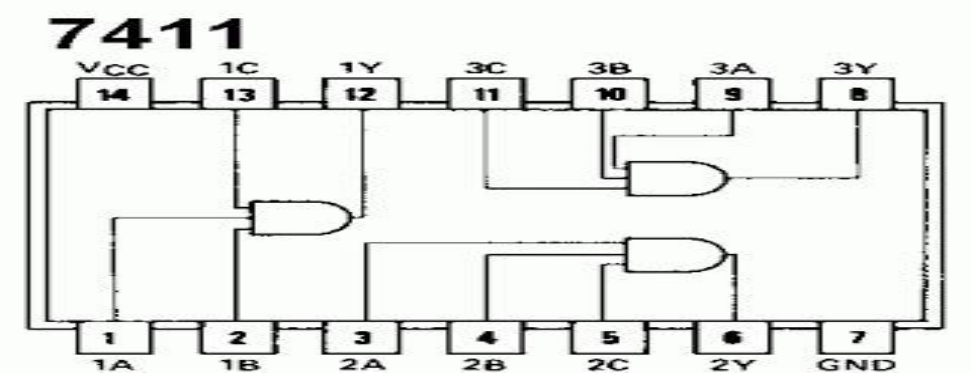
OR gate is used in many of our real life applications. One of its applications, "Alarm circuit for car door system" is explained below.

We design this alarm by connecting four circuits to the 4 doors of the car (or any vehicle) when the door is open, the circuit generates high output (Logic 1) and similarly when the door is closed, the circuit generates 0 (Logic 0). The 4 outputs of four doors are connected to a 4 –input OR gate, which output is connected to an alarm.



When any one or more doors are opened, then the output of the 4- input OR gate is 1, then the alarm rings. So that we can know the condition of doors, whether they are properly closed or not.

3-input OR gate IC



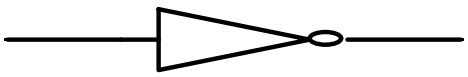
3. NOT Gate/Inverter

CHARACTERISTICS:

- Perform on a single input variable.
- The Boolean operation for the NOT operation is:

  Y = $\overline{A}$ ('-' overbar represents the NOT operation)

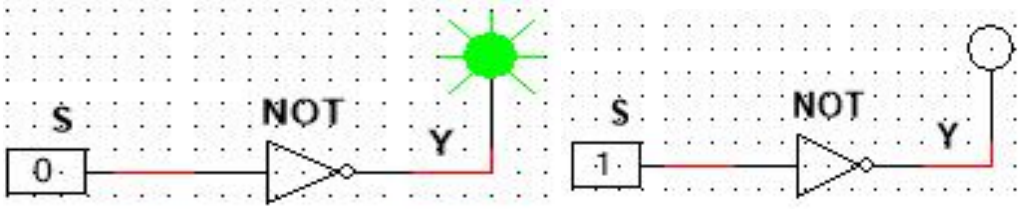
SYMBOL:



TRUTH TABLE:

Input	Output
S_0	$\overline{S_0}$
0	1
1	0

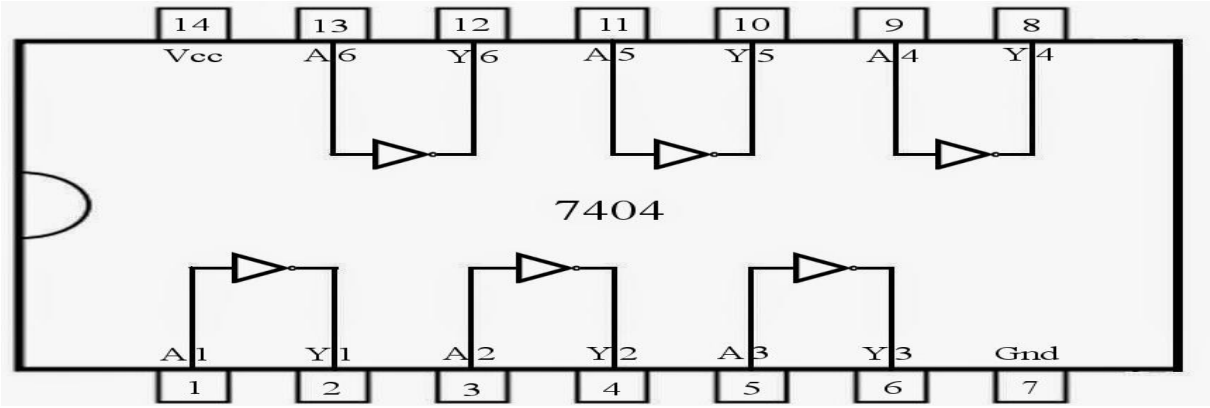
Varification:



Not Gate Applications

NOT gates produce reverse output of the input, so they are also called "Inverters". The CMOS inverters are generally used in designing of oscillators. They are used mostly, because of their low power consumption. Another advantage of CMOS inverter is that they can be easily interfaced than other logic devices.

NOT gate IC

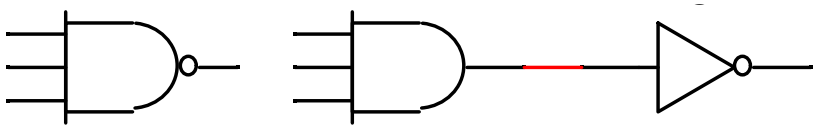


4. NAND Gate CHARACTERISTICS:

- NAND gate is equal to an AND gate followed by a NOT gate.
- A NAND operation produces the result (output Low) '0' when all operands are '1', otherwise the result is (output High) '1'.
- The Boolean operation for the NAND operation is:

  Y = $\overline{A \,.\, B}$ ('.' represents the AND operation & '-' overbar represents the NOT operation)

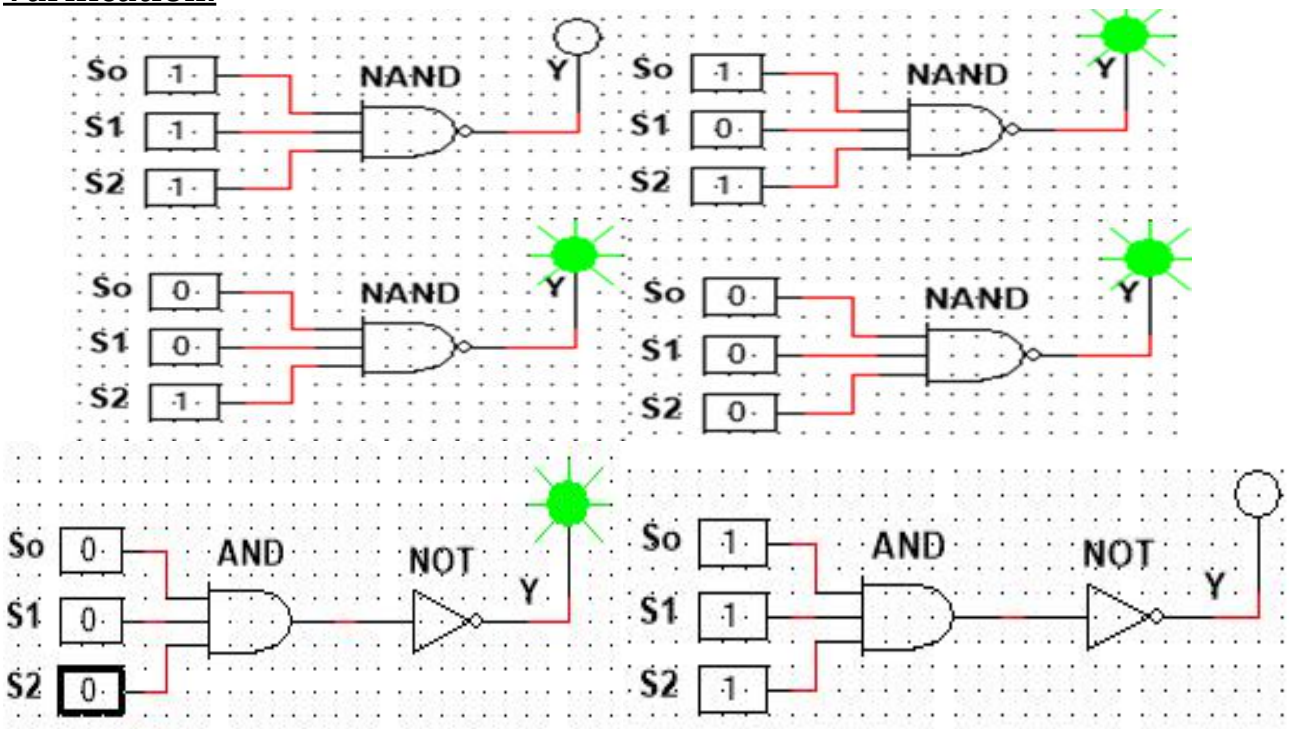
SYMBOL:



TRUTH TABLE for 3 Input:

Input			Output
S_2	S_1	S_0	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Varification:

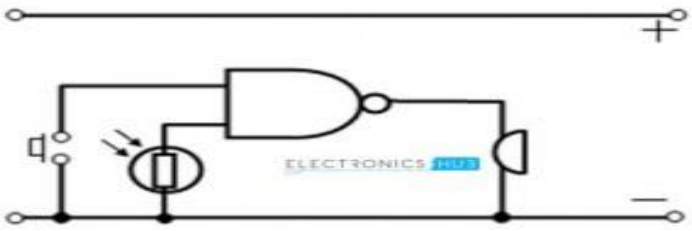


NAND Gate Applications

Logic NAND gate has many applications like Burglar alarm, freezer buzzer etc.

1. Burglar alarm or theft alarm

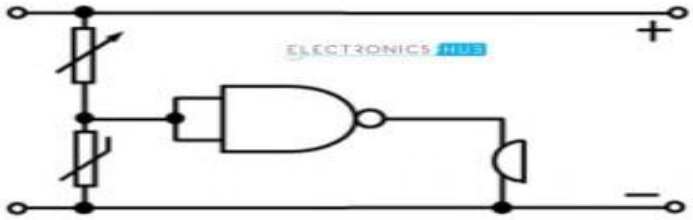
The burglar alarm circuit is shown below. It has a NAND gate with an LDR input. LDR means Light Dependent Resistor. When the alarm switch is closed, then one of the NAND gate inputs will be low. And if the LDR is kept in light, then the second input is also low. So the 2 inputs of the NAND gates are low. So if any of the two situations occurs, the output of NAND gate becomes HIGH and then the theft alarm rings, as a sign of alert.



The Burglar alarm is an electronic device which is used to detect the unauthorized entry, and as a security alarms and theft alarms. These are used in commercial as well as residential and military security purposes against intruders. These alarms can be connected to television sets and closed circuit television surveillance systems.

2. Freezer warning buzzer

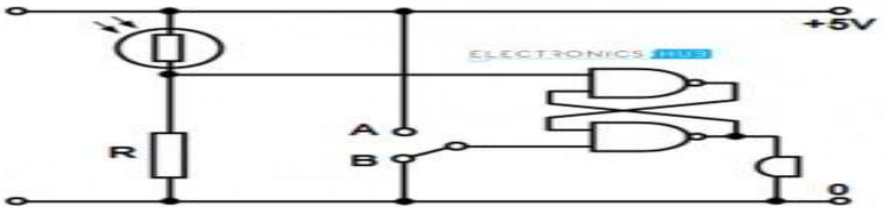
This freezer warning buzzer circuit uses NAND gate as a NOT gate. The single input NAND gate (functioning as NOT gate) has two thermistors at its input side. Whenever the thermistor is cold, its resistance will be high so the input of NAND gate will be HIGH. As the NAND gate is functioning as a NOT gate, the output of the NOT gate will be LOW.
Similarly, when the Thermistor gets warmer, its resistance will decrease. So, the voltage drop across the Thermistor will be low, making the input of the NAND gate to become LOW.



When the voltage drop across the two Thermistors will be low, then the output of the NAND gate will become high and then the Buzzer rings.

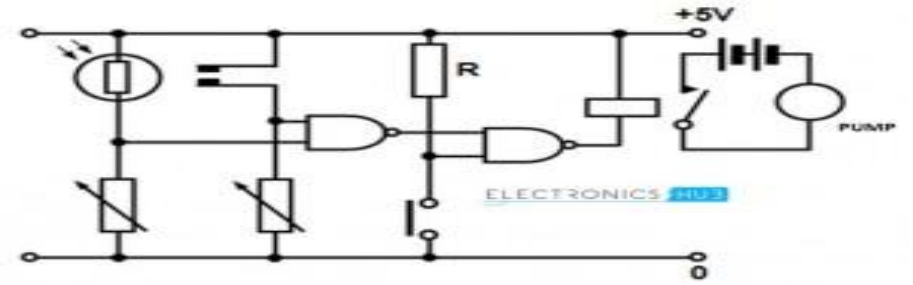
3. Light activated theft alarm

This circuit consists of NAND gates connected in the form of a simple latch circuit. When the switch is connected to 'A', the buzzer inputs will be off. At this condition there is no effect of LDR (which is connected as one of the inputs of Latch) on the circuit. But when the switch is at 'B' then the buzzer will be ON due to the effect of LDR. The buzzer will sound simultaneously the light or flash light also flashes up to alert. Buzzer can be turned off only by returning the switch to position 'A'.

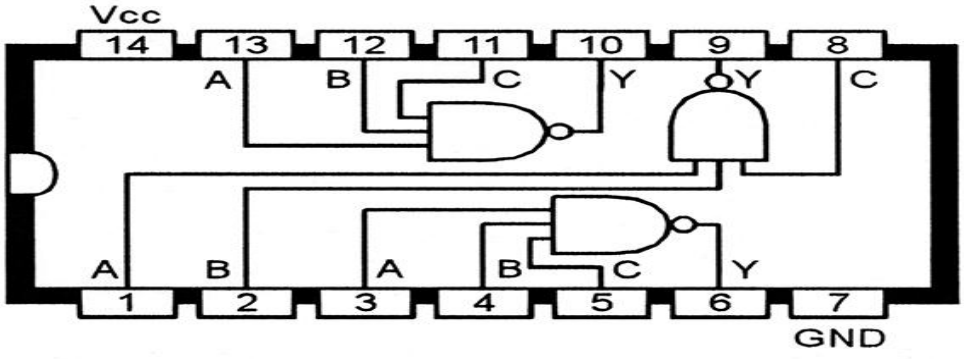


4. Automatic watering system

This technological invention is useful for water plants at nighttime. This circuit functions only when the LDR will OFF (generally happens at nighttime) and when the surrounded atmosphere of the Thermistor is wet. This circuit has a relay to act as switch which allows pumping the water only when the two input conditions of the NAND gate are fulfilled.



3-input NAND gate IC

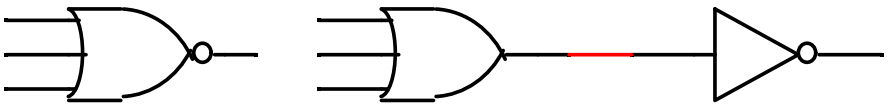


5. NOR Gate

CHARACTERISTICS:

- NOR gate is equal to an OR gate followed by a NOT gate.
- A NOR operation produces the result (output High) '1' if all the operands are '0', otherwise the result is (output Low) '0'.
- The Boolean operation for the NOR operation is:
  Y = $\overline{A + B}$ ('+' represents the OR operation & '-' overbar represents the NOT operation)

SYMBOL:

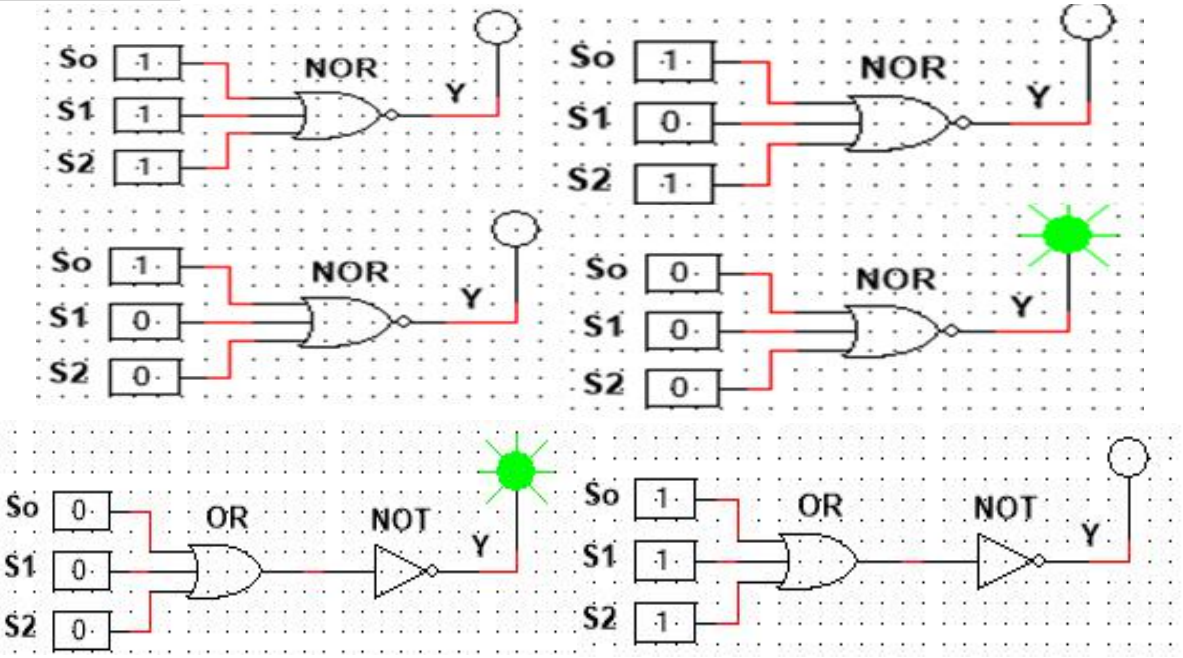


TRUTH TABLE for 3 Input:

Input			Output
S_2	S_1	S_0	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0

1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

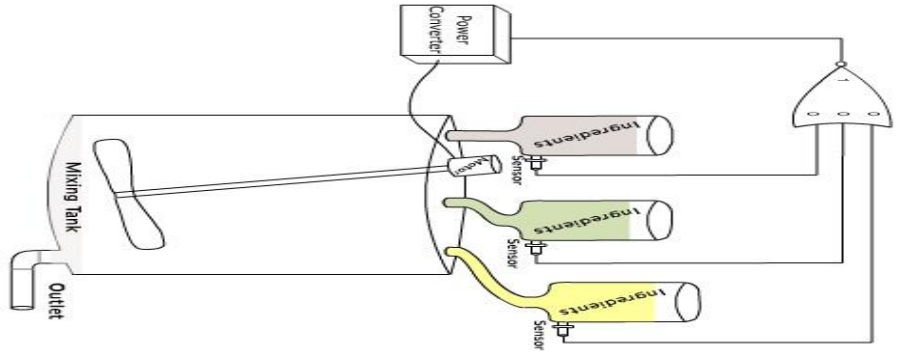
Varification:



NOR Gate Applications

Logic NOR gate can be used to construct EX-OR gates and some other real time applications. One of its real time applications is 'Mixer tank'. It is explained below.

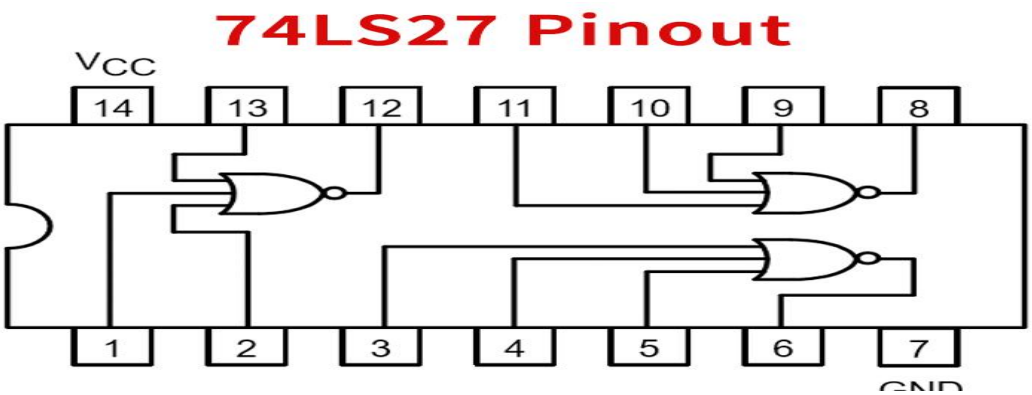
If you observe the below diagram, we use a 3-input NOR gate to control the flow of ingredients of a mixer tank. Different kinds of ingredients are stored in separate cylindrical Hooper. A capacitive proximity switch is arranged outside of each cylinder, at its bottom. This is to detect the amount or level of ingredient.



When the ingredients come down to the sensor level, the cylinder produces HIGH level logic output. The ingredient level higher than the sensor will consider as low logic level (0).When all the ingredients comes down to the sensor level, then all inputs of the NOR gate will become HIGH.

When all the ingredients comes down to the sensor level, the output of NOR gate will become HIGH. This output is connected to the Power converter and this power converter is connected to the Motot (to mix the ingredients). The HIGH output of the NOR logic gate activates the power converter and activates the motor, making the motor to run and mix all the ingredients.

3-input NOR gate IC



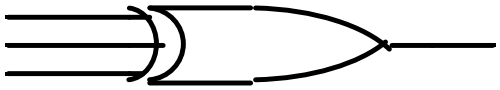
6. EX-OR Gate

CHARACTERISTICS:

- The 'Exclusive-OR' gate produces result (output high) '1' if either of its two inputs are high and produces result (output low) '0' if both of its two inputs are either high or low.
- The Boolean operation for the NOR operation is:

  Y = A $\oplus$ B = ($\overline{A}.B + A.\overline{B}$) ('$\oplus$' represents the EX-OR operation)

SYMBOL:

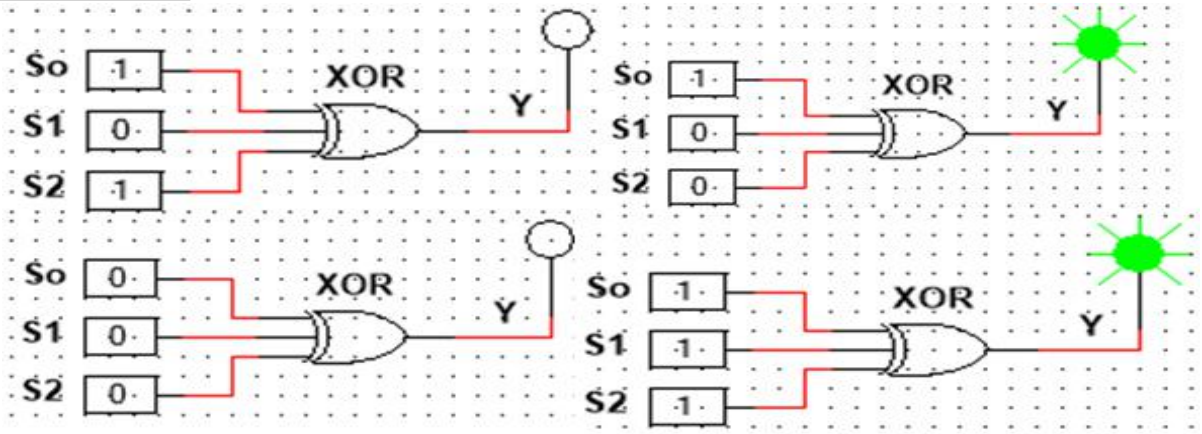


TRUTH TABLE for 3 Input:

Input			Output
S_2	S_1	S_0	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1

1	0	1	0
1	1	0	0
1	1	1	1

Varification:



Applications of Ex-OR Gate

XOR Logic Gate is used in many applications. Some of them are explained below.

1. Used in Adder (Addition)

We can design single bit adder (also known as Half Adders), which will add two bits and produce a single bit output. The single bit Adder designed by using XOR gate is shown below.

For example, if we add two bits '1' and '1' in binary addition, we get the answer '10' and in decimal addition method we get 2. The main principle of half adders is that the trailing sum is achieved by the output of XOR gate, and the carry bit is calculated by AND gate.
We can cascade many single bit adder circuits to form n-bit adder circuit, to calculate the sum of longer binary numbers.

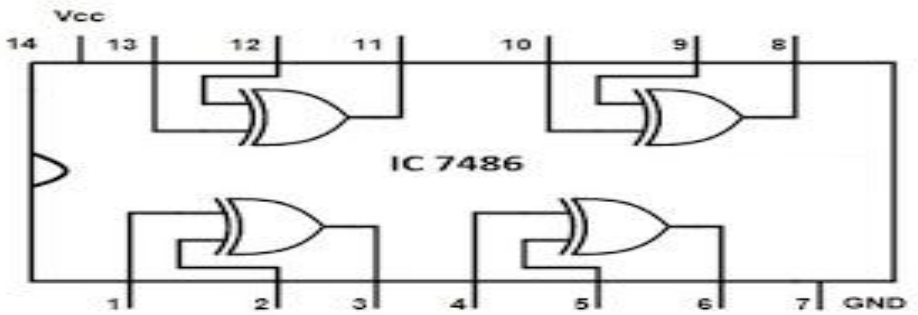
2. Pseudo-Random Number Generation

Linear shift registers are also called as Pseudo Random Number Generators (PNRs). In order to generate the random numbers, we arrange the XOR logic gate in a specific order by forming a linear feedback shift register.

3. Correlation and Sequence Detection

XOR gate is capable of producing a low-level input i.e., 0, when all of its inputs are HIGH or LOW. When we are searching for a particular bit sequence in a long data sequence, we use XOR gates to find the required data bit sequence.
The accuracy of finding the required string of data bits in the target sequence is determined by calculating the number of 0's obtained. In many communication devices such as decoders and CDMA receivers, we use co-relaters, which are used to extract the parity of a specific Pseudo Random Number sequence in a group of PRN sequence.

3-input XOR gate IC



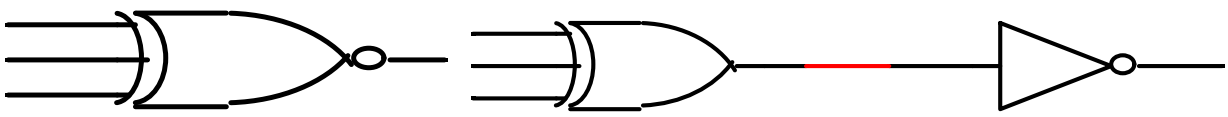
7. EX-NOR Gate

CHARACTERISTICS:

- EX-NOR gate is equal to an EX-OR gate followed by a NOT gate.
- The 'Exclusive-NOR' gate circuit does the opposite to the EX-OR gate. It produces result (output low) '0' if either of its two inputs are high and produces result (output High) '0' if both of its two inputs are either high or low.
- The Boolean operation for the NOR operation is:

  Y = $\overline{A \oplus B}$ = ($\overline{A.B} + A.B$) ('$\oplus$' represents the EX-OR operation & '-' overbar represents the NOT operation)

SYMBOL:

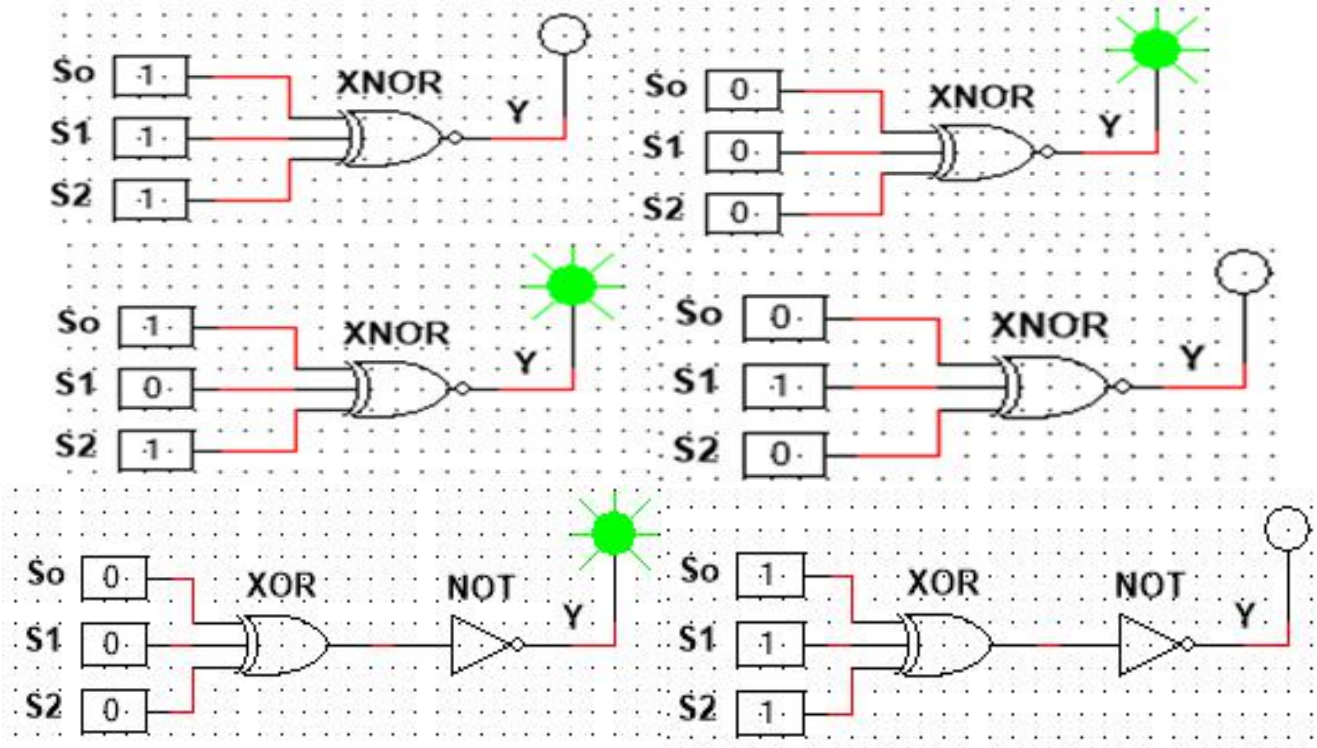


TRUTH TABLE for 3 Input:

Input			Output
S_2	S_1	S_0	Y
0	0	0	1
0	0	1	0

0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Varification:

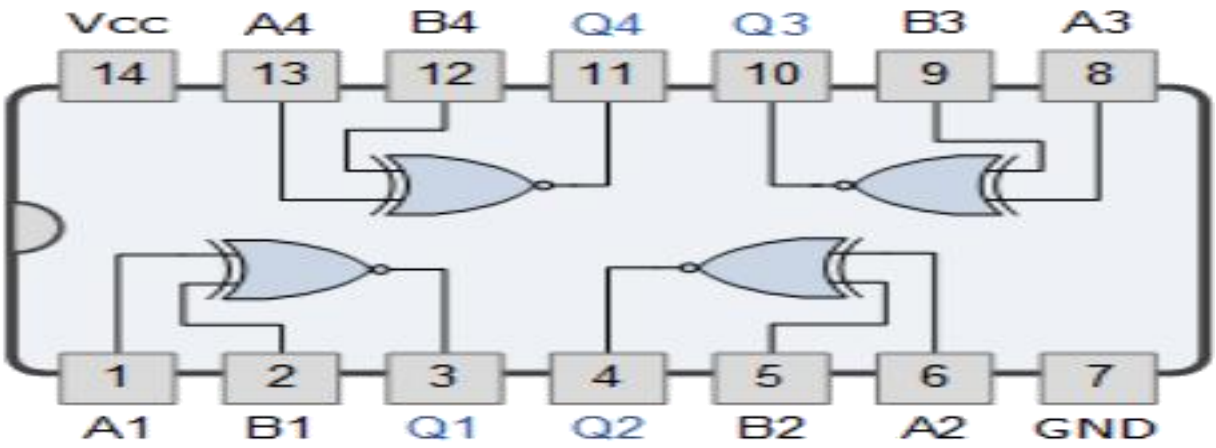


Ex-NOR gate applications

1. The XNOR logic gates are used in error detecting circuits which are to detect Odd parity or even parity bits in digital data transmission circuits.
2. XNOR gate is mainly used in arithmetic and encryption circuits. This process is the combinational operation of the XOR and XNOR gates, by using 6 transistors for low power application.

This is also used as **Heat exchanger tank**, which will ring the alarm when the water temperature level goes up or down, to the pre-set level.

3-input XNOR gate IC



Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.3

Logic Gates at Circuit Level

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: Implantation of AND, OR, NOT, NAND and NOR *at circuit level* using MultiSim platform.

EQUIPMENT:

- Power supply unit (EV)
- LED
- Probe

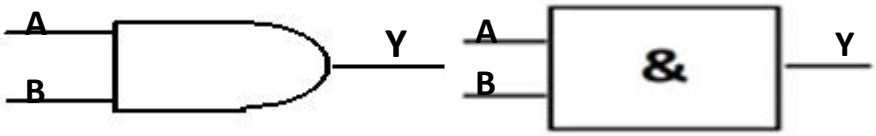
COMPONENT:

- Transistor
- Diode
- Resistor

And Gate

Implantation of AND at circuit level.

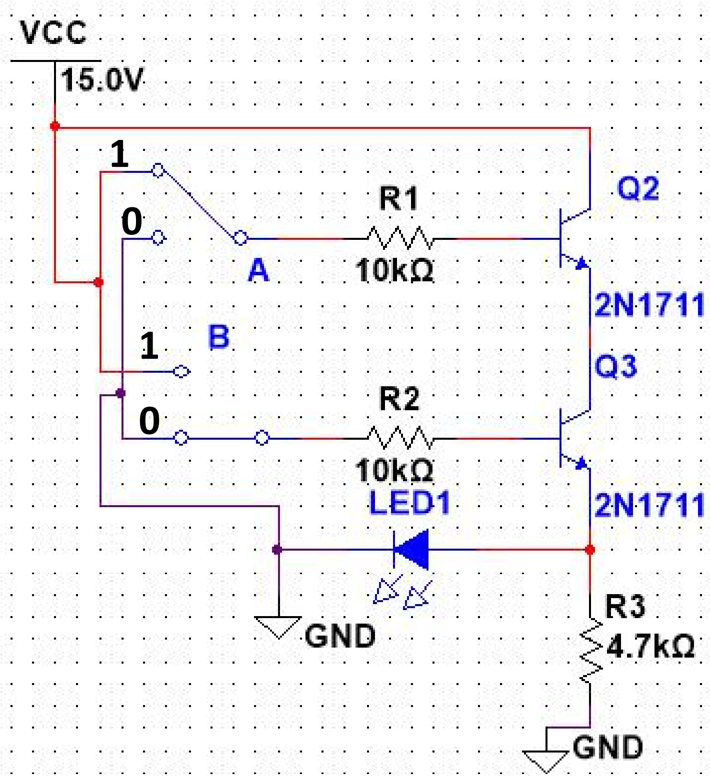
SYMBOL:

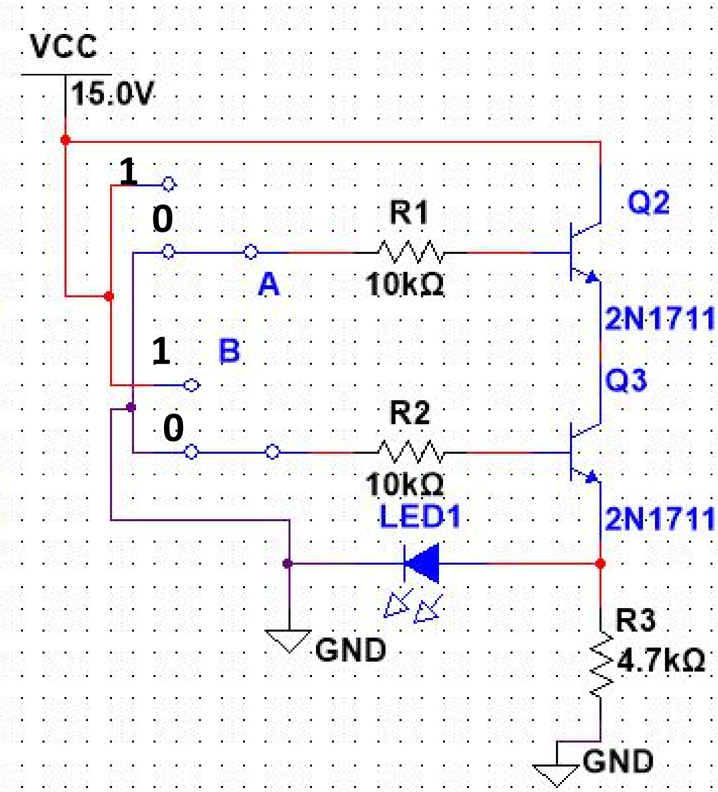


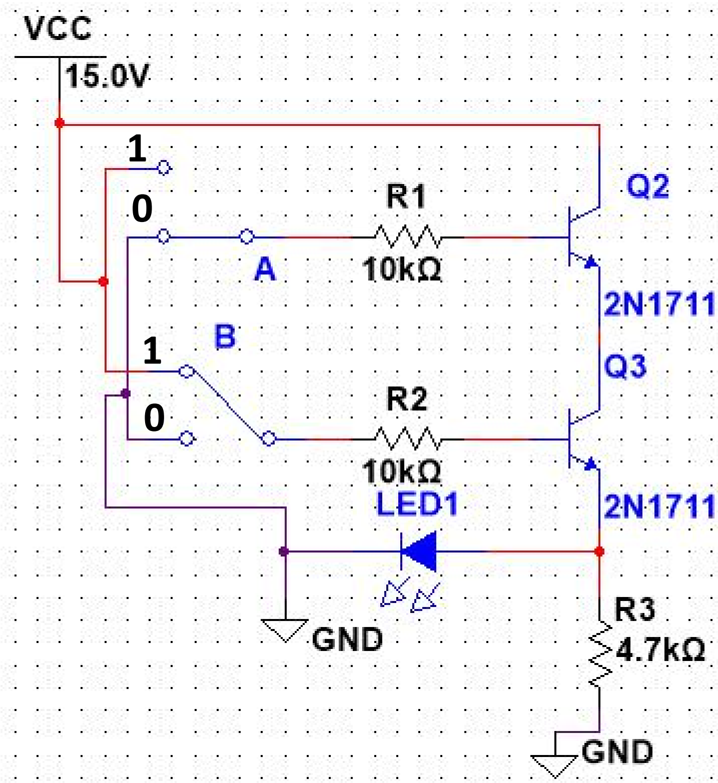
TRUTH TABLE:

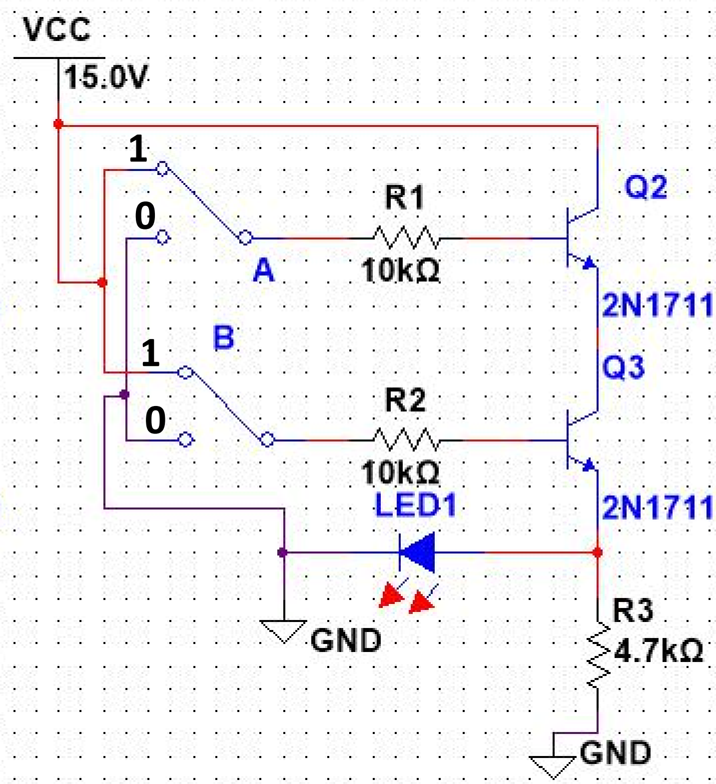
INPUT		OUTPUT
A	**B**	**Y**
0	0	0
0	1	0
1	1	1
1	0	0

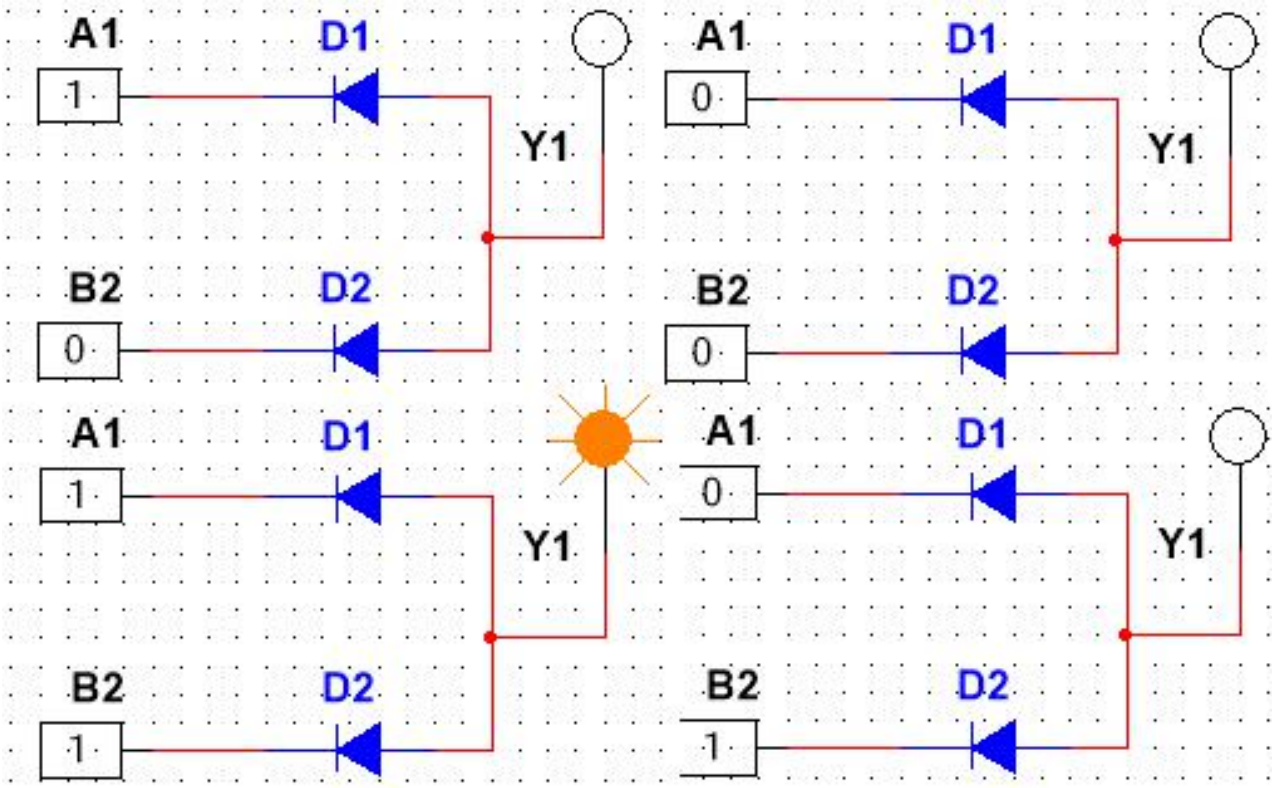
A simple 2-input logic AND gate can be constructed using RTL Resistor-transistor switches. Connected together as shown below with the inputs connected directly to the transistor bases. Both transistors must be saturated “ON” for an output at LED.


VCC
15.0V
1
0
A
B
1
0
R1
10kΩ
R2
10kΩ
Q2
2N1711
Q3
2N1711
LED1
GND
R3
4.7kΩ
GND


VCC
15.0V
1
0
A
B
1
0
R1
10kΩ
R2
10kΩ
Q2
2N1711
Q3
2N1711
LED1
GND
R3
4.7kΩ
GND


VCC
15.0V
1
0
A
B
1
0
R1
10kΩ
R2
10kΩ
Q2
2N1711
Q3
2N1711
LED1
GND
R3
4.7kΩ
GND


VCC
15.0V
1
0
A
B
1
0
R1
10kΩ
R2
10kΩ
Q2
2N1711
Q3
2N1711
LED1
GND
R3
4.7kΩ
GND

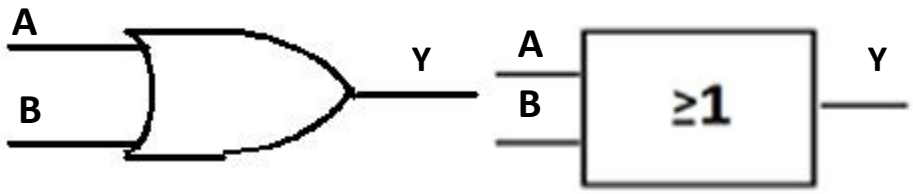


Logic AND Gates are available using digital circuits to produce the desired logical function and is given a symbol whose shape represents the logical operation of the AND gate. Commercially available AND gate IC's are only available in standard 2, 3, or 4-input packages**.** If additional inputs are required, then standard AND gates will need to be cascaded together to obtain the required input value.

OR Gate

Implantation of OR at circuit level.

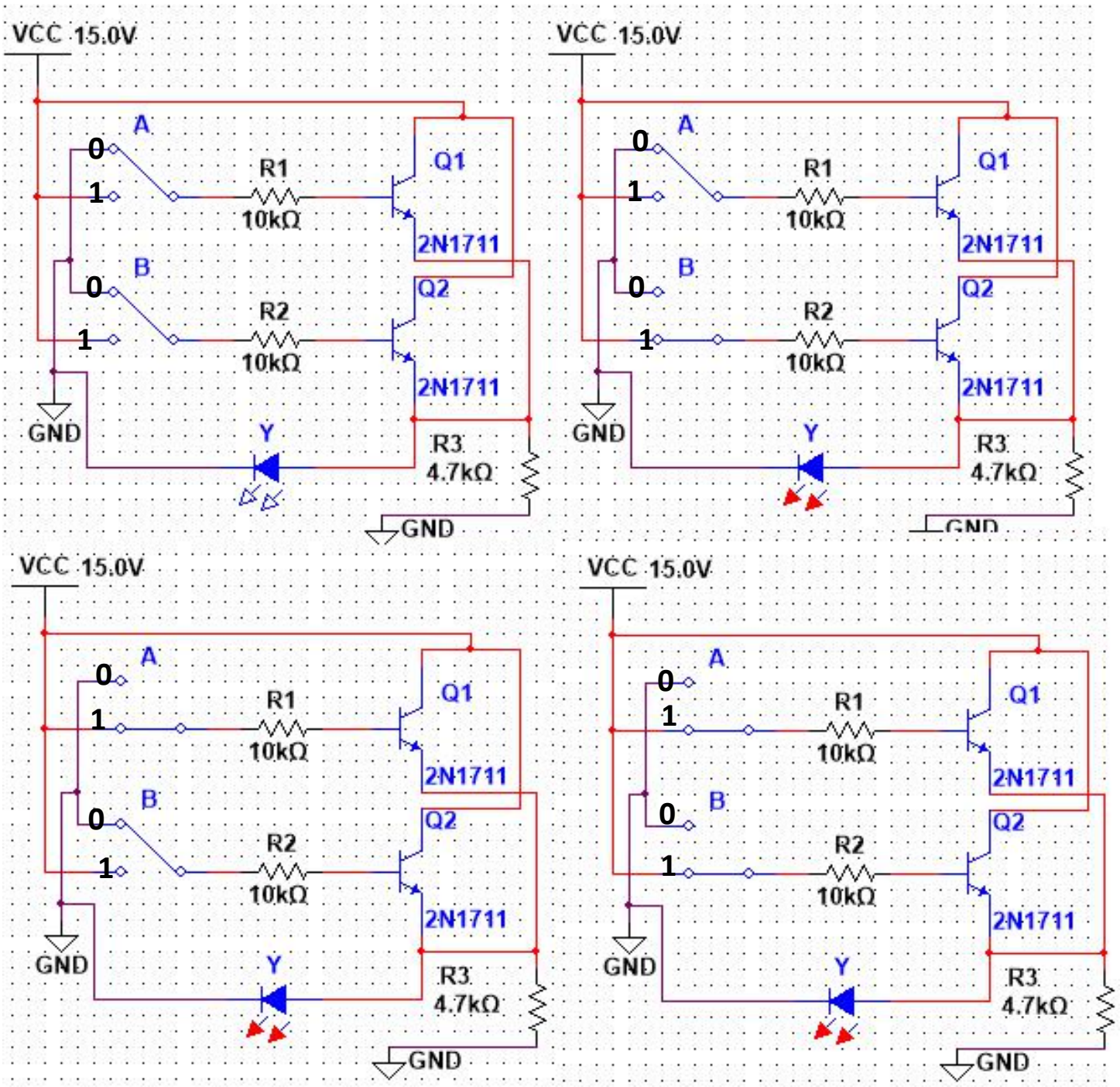
SYMBOL:

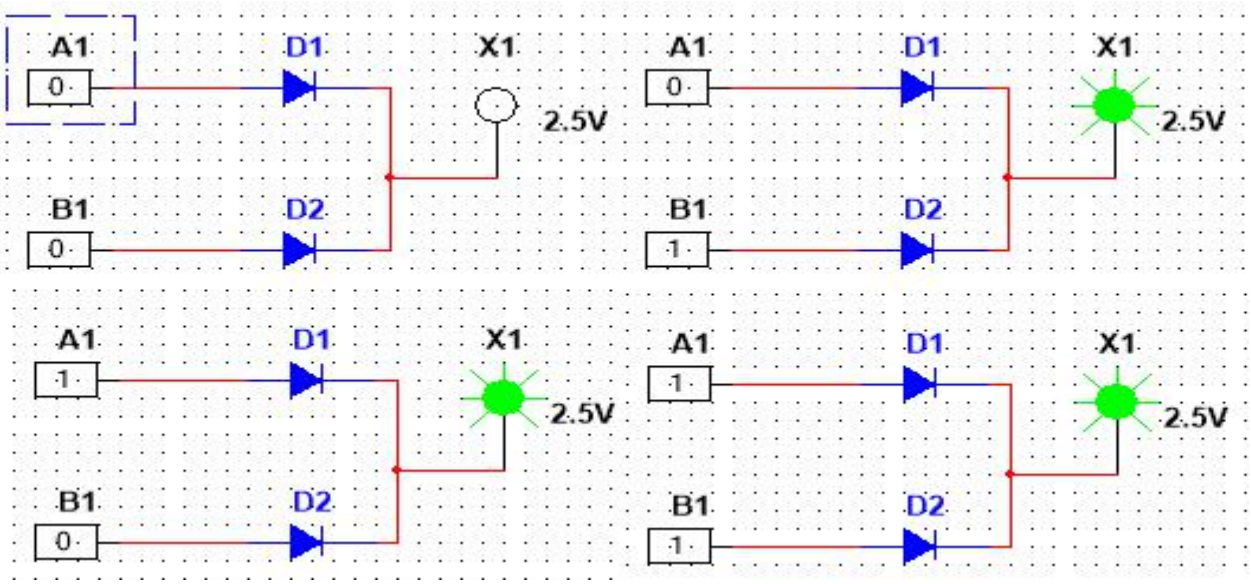


TRUTH TABLE:

INPUT		OUTPUT
A	B	Y
0	0	0
0	1	1
1	1	1
1	0	1

A simple 2-input inclusive OR gate can be constructed using RTL Resistor-transistor switches. Connected together as shown below with the inputs connected directly to the transistor bases.
Either transistor must be saturated "ON" for an output at Y.


VCC 15.0V
A
0
1
B
0
1
R1
10kΩ
R2
10kΩ
Q1
2N1711
Q2
2N1711
GND
Y
R3
4.7kΩ
GND

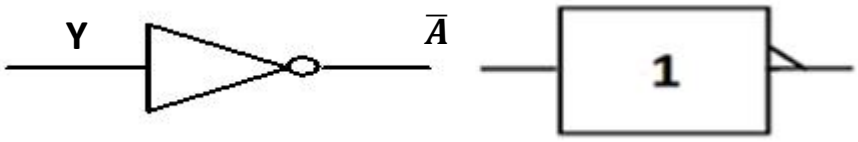


Logic OR Gates are available using digital circuits to produce the desired logical function and is given a symbol whose shape represents the logical operation of the OR gate. Commercially available OR gates are available in 2, 3, or 4 input types. Additional inputs will require gates to be cascaded together.

NOT Gate

Implantation of NOT at circuit level.

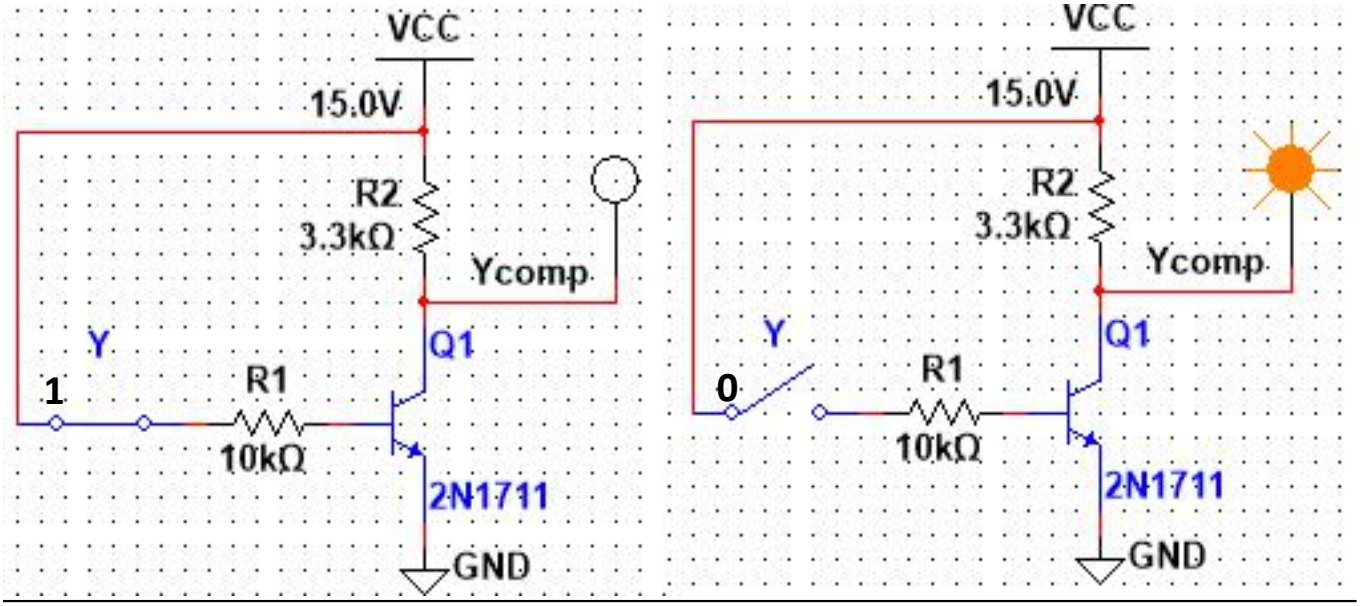
SYMBOL:



TRUTH TABLE:

INPUT	OUTPUT
Y	$\overline{A}$
1	0
0	1

A simple 2-input logic NOT gate can be constructed using a RTL Resistor-transistor switches as shown below with the input connected directly to the transistor base. The transistor must be saturated “ON” for an inverted output “OFF” at Y.

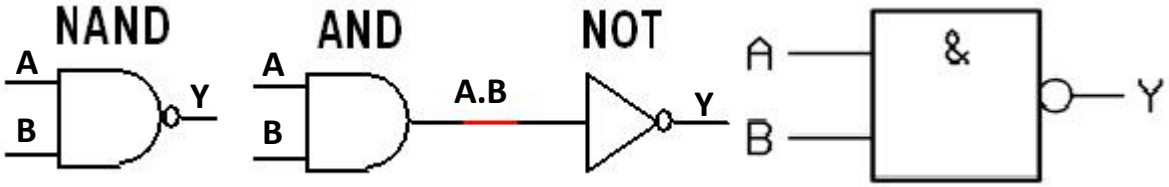


Logic NOT Gates are available using digital circuits to produce the desired logical function. The standard NOT gate is given a symbol whose shape is of a triangle pointing to the right with a circle at its end. This circle is known as an "inversion bubble". This bubble denotes a signal inversion (complementation) of the signal and can be present on either or both the output and/or the input terminals.

NAND Gate

Implantation of NAND at circuit level.

SYMBOL and EQUIVALENT CIRCUIT:

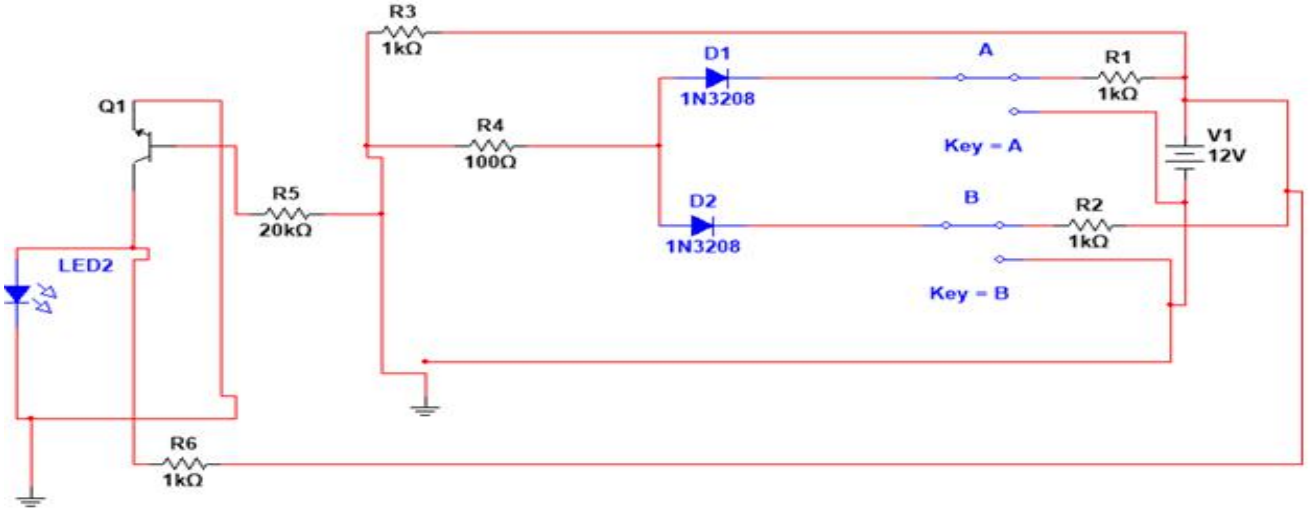


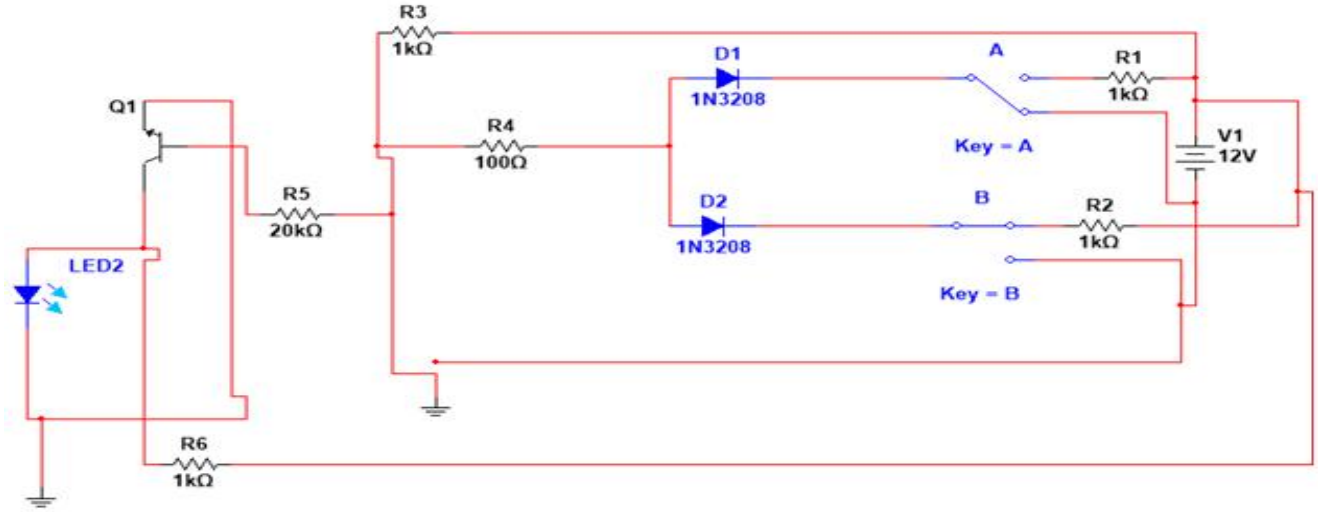
Equivalent Circuit

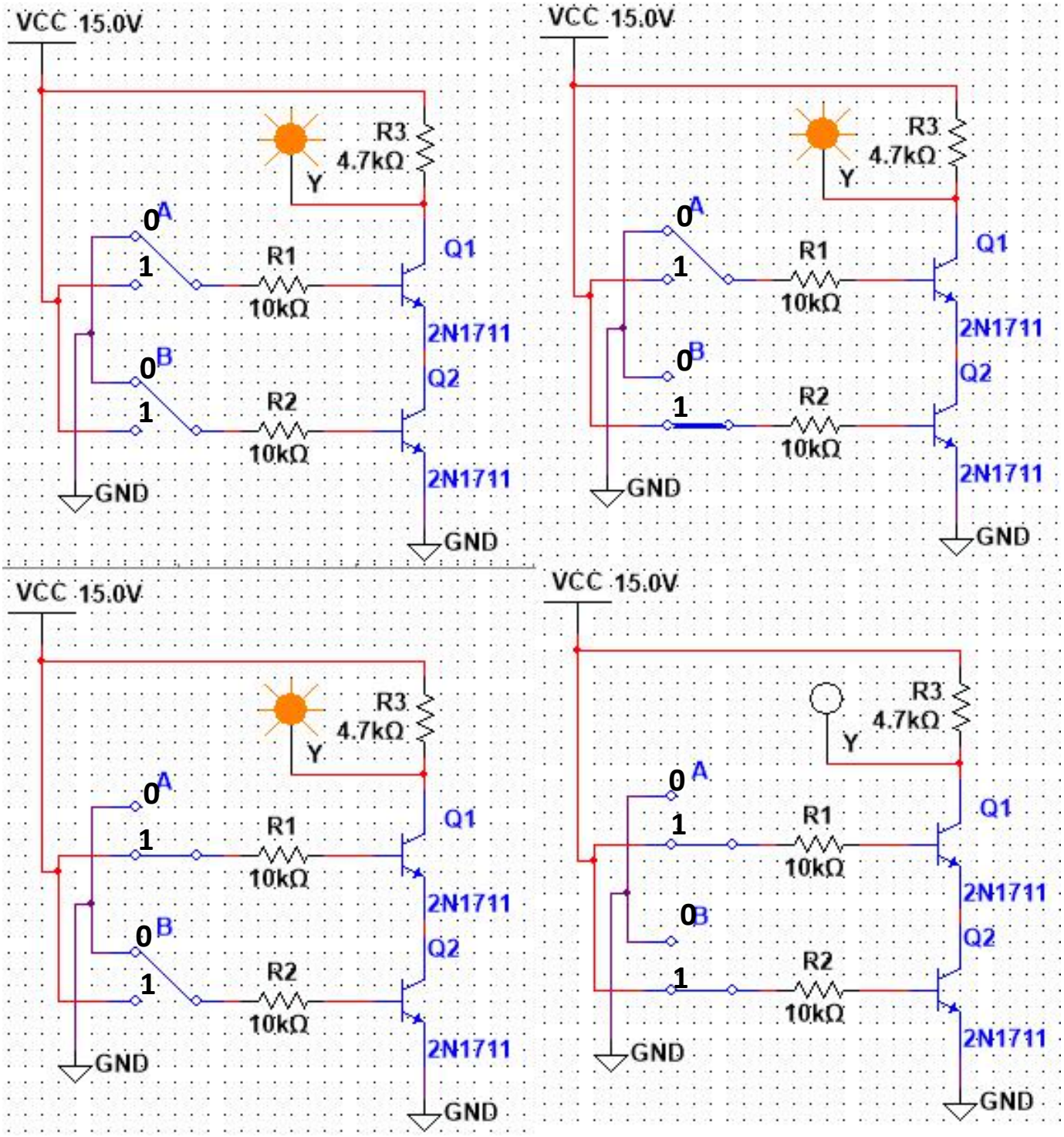
TRUTH TABLE:

INPUT		OUTPUT
A	**B**	**Y**
0	0	1
0	1	1
1	1	0
1	0	1

A simple 2-input logic NAND gate can be constructed using RTL Resistor-transistor switches connected together as shown below with the inputs connected directly to the transistor bases. Either transistor must be cut-off "OFF" for an output at Y.


R3
1kΩ
D1
1N3208
A
R1
1kΩ
Q1
R4
100Ω
Key = A
V1
12V
R5
20kΩ
D2
1N3208
B
R2
1kΩ
LED2
Key = B
R6
1kΩ

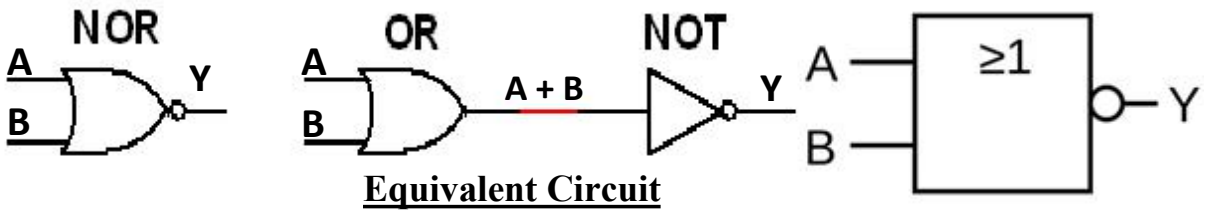

R3
1kΩ
D1
1N3208
A
R1
1kΩ
Q1
R4
100Ω
Key = A
V1
12V
R5
20kΩ
D2
1N3208
B
R2
1kΩ
LED2
Key = B
R6
1kΩ



Logic NAND Gates are available using digital circuits to produce the desired logical function and is given a symbol whose shape is that of a standard AND gate with a circle, sometimes called an "inversion bubble". Commercially available NAND Gate IC's are available in standard 2, 3, or 4 input types. If additional inputs are required, then the standard NAND gates can be cascaded together to provide more inputs.

NOR Gate

Implantation of NOR at circuit level.
SYMBOL and EQUIVALENT CIRCUIT:

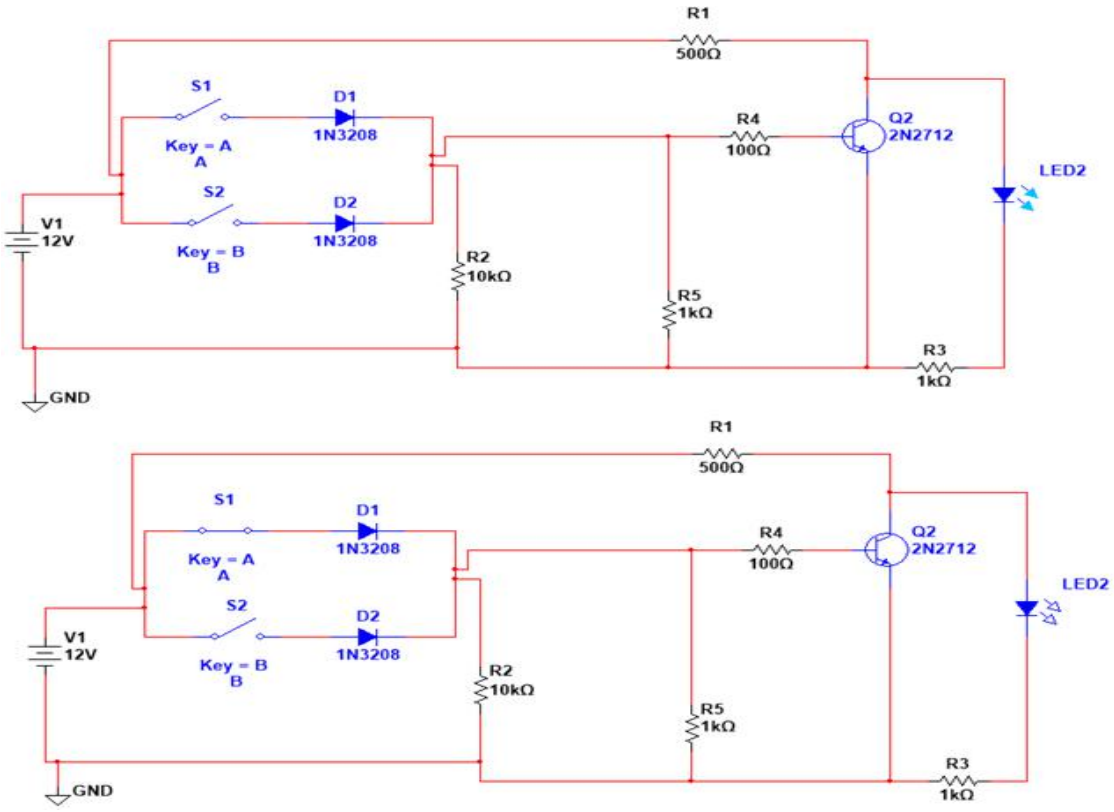


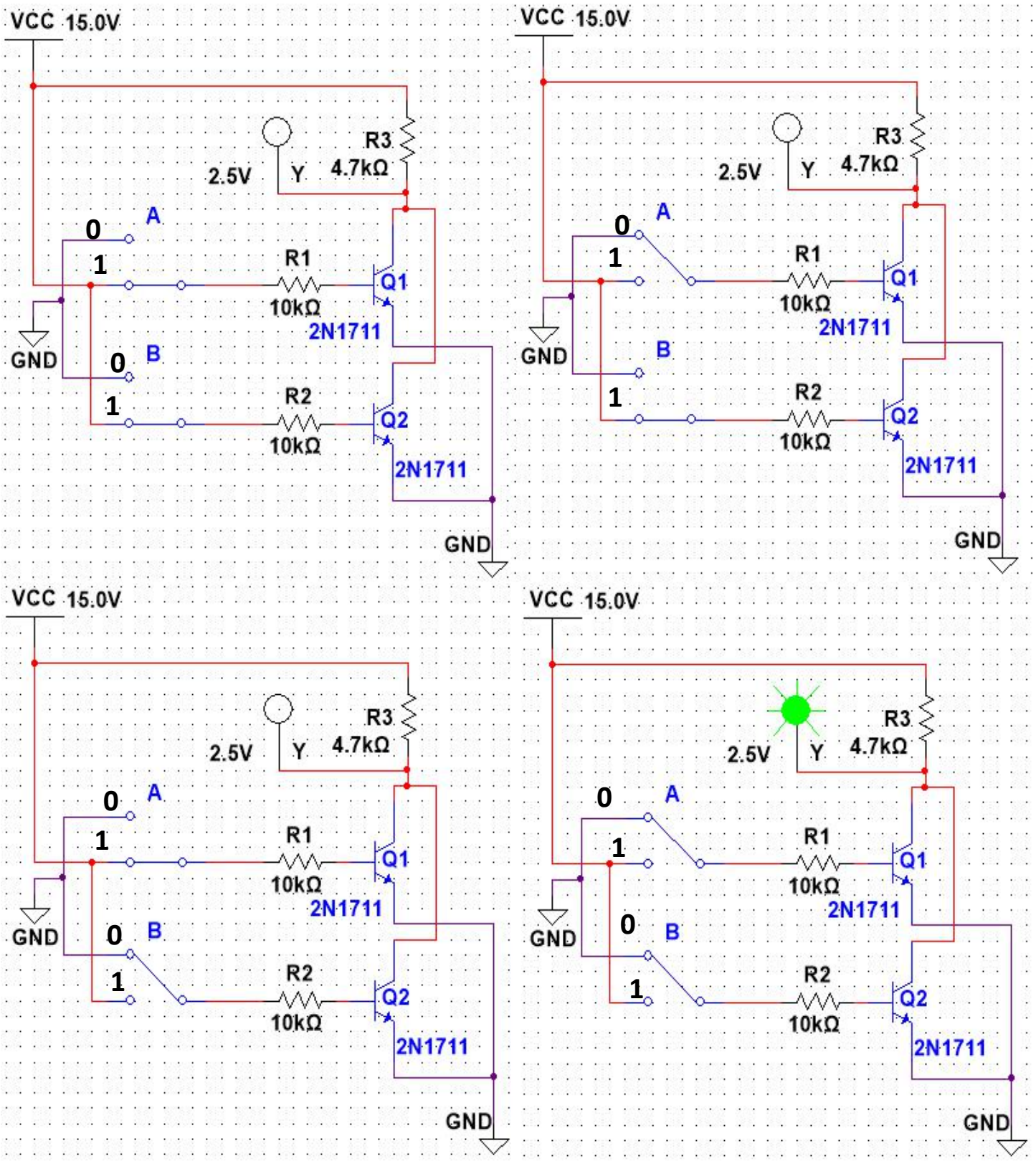
Equivalent Circuit

TRUTH TABLE:

INPUT		OUTPUT
A	**B**	**Y**
0	0	1
0	1	0
1	1	0
1	0	0

A simple 2-input logic NOR gate can be constructed using RTL Resistor-transistor switches connected together as shown below with the inputs connected directly to the transistor bases. Both transistors must be cut-off "OFF" for an output at Y.





Logic NOR Gates are available using digital circuits to produce the desired logical function and is given a symbol whose shape is that of a standard OR gate with a circle, sometimes called an “inversion bubble”. Commercially available NOR Gate IC’s are available in standard 2, 3, or 4 input types. If additional inputs are required, then the standard NOR gates can be cascaded together to provide more inputs.

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.5
HALF ADDER AND FULL ADDER

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: Design the Half Adder and Full Adder at Logic Level using MultiSim platform.

EQUIPMENT:

- Power supply unit (EV)
- Green and Red Probes

COMPONENTS FOR HALF ADDER:

- AND Gate
- Ex-OR Gate

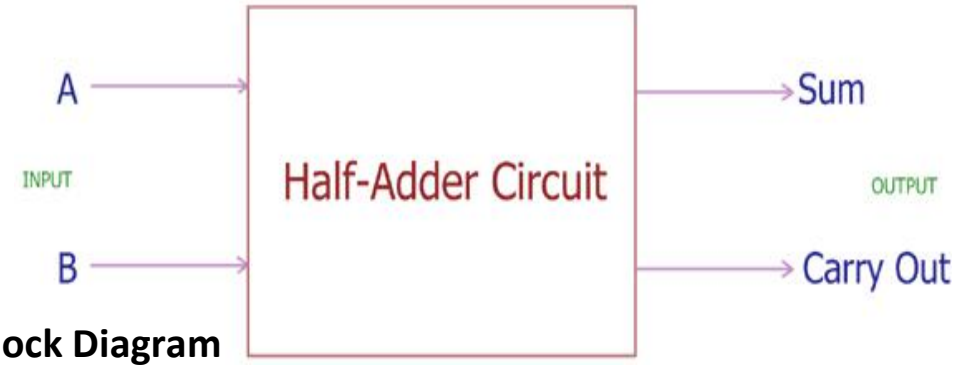
COMPONENTS FOR FULL ADDER:

- AND Gate
- OR Gate
- Ex-OR Gate

HALF ADDER:

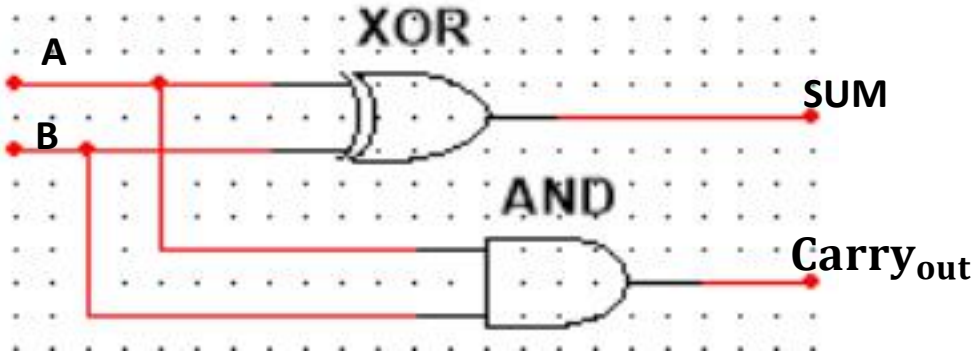
A half adder is a logical circuit that performs an addition operation on two binary digits. The half adder produces a sum and a carry value which are both binary digits.

SYMBOL:



Block Diagram

Circuit Diagram



TRUTH TABLE:

Input		Output	
B	A	Sum	$Carry_{out}$
0	0	0	0
1	0	1	0
1	1	0	1
0	1	1	0

From the truth table of the half adder, we can see that the SUM output is the result of the Exclusive-OR gate and the Carry-out is the result of the AND gate. Then the Boolean expression for a half adder is as follows.

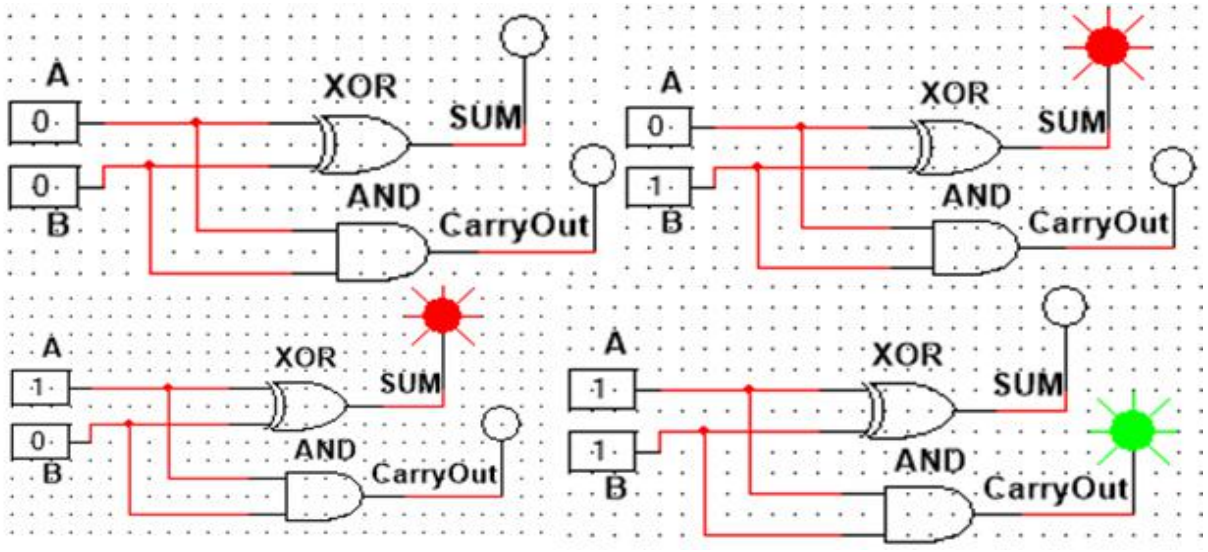
For the **SUM** bit:

$$SUM = A \text{ XOR } B = A \oplus B$$

For the **CARRY** bit:

$$CARRY = A \text{ AND } B = A.B$$

Verifications By Simulation:

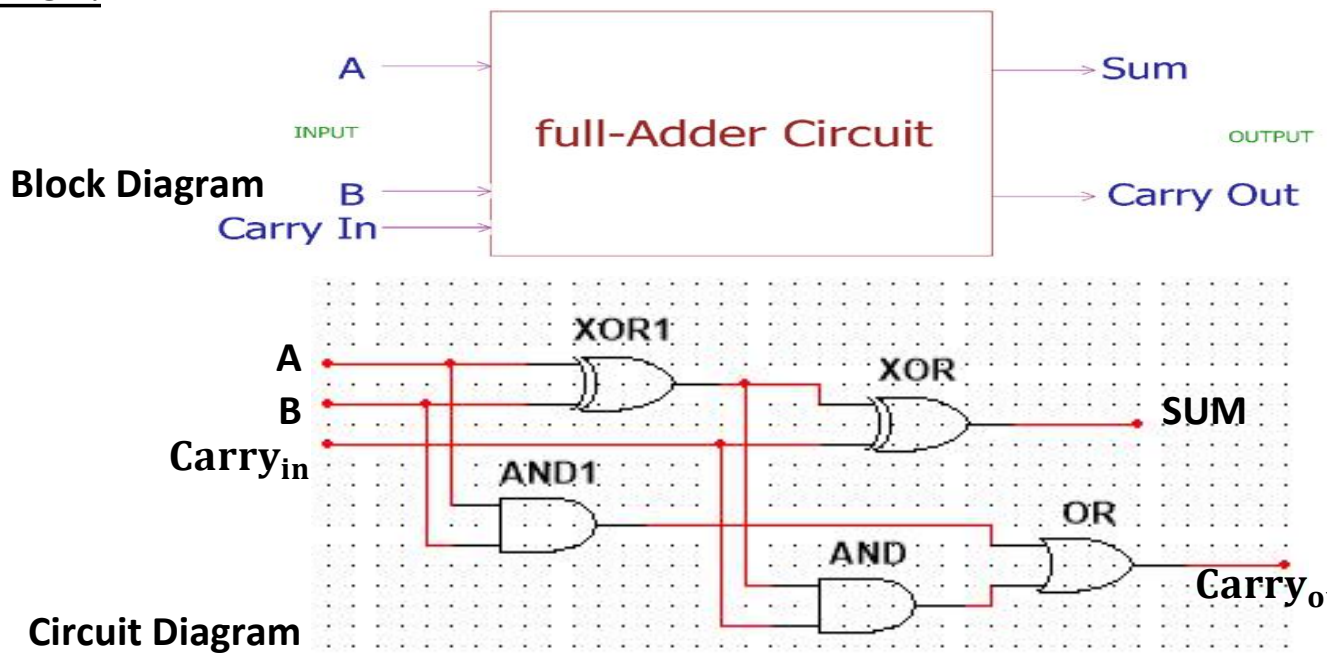


One major disadvantage of the Half Adder circuit when used as a binary adder, is that there is no provision for a "Carry-in" from the previous circuit when adding together multiple data bits. One simple way to overcome this problem is to use a **Full Adder** type binary adder circuit.

FULL ADDER:

The full adder accepts three inputs including an input carry and generates a sum output and an output carry. The basic difference between a full adder and a half adder is that the full adder accepts an input carry. The full adder must add the two input bits and the input carry. A ***Carry-in*** is a possible carry from a less significant digit, while a ***Carry-out*** represents a carry to a more significant digit.

SYMBOL:



TRUTH TABLE:

Input			Output	
$Carry_{in}$	B	A	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

Then the Boolean expression for a full adder is as follows.

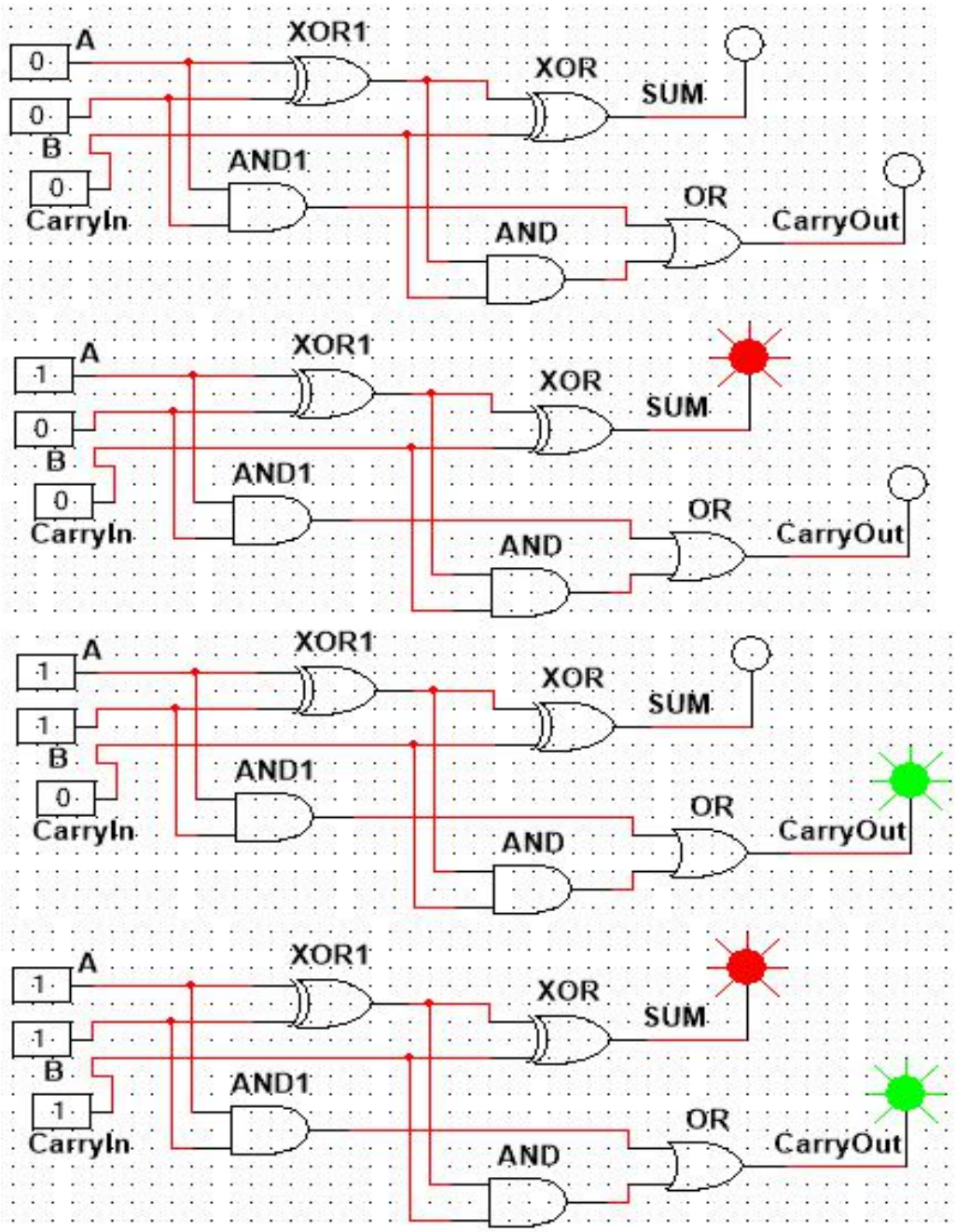
For the **SUM** (S) bit:

SUM = (A XOR B) XOR Cin = (A $\oplus$ B) $\oplus$ Cin

For the **CARRY-OUT** (Cout) bit:

CARRY-OUT = A AND B OR Cin (A XOR B) = A.B + Cin (A $\oplus$ B)

Verifications By Simulation:



We have seen above that single 1-bit binary adders can be constructed from basic logic gates. But what if we wanted to add together two n-bit numbers, then n number of 1-bit full adders need to be connected or "cascaded" together and which will add two 4-bit binary number and provide an additional input carry bit, as well as an output carry bit, so you can cascade them together to produce 8-bit, 12-bit, 16-bit adder.

74LS83 4-bit adder IC:

The 4-bit adder IC present on the module is an integrate of TTL family. Fig:9.2 shows a pin diagram of 4-bit TTL 74LS83 adder IC. This IC contains four full adder inside; carry out of each full adder is connected to carry in of next full adder. The Co is a basically carry input for 4-bit full adder and C4 is a carry out from 4th full adder for 4-bit full adder. A4A3A2A1 and B4B3B2B1 are the inputs and S4S3S2S1 are outputs of 4-bit adder. This IC can be used as half adder, 2-bit, 3-bit, and 4-bit full adder. It can be connected in cascade to other similar circuits, to form adders with more than 4-its.

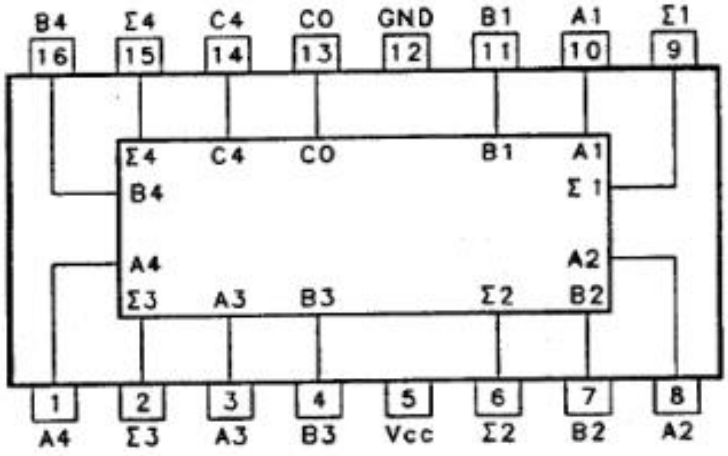


Fig: 9.2 Pin configuration for 74LS83

Applications

The applications of this basic adder are as follows:

- *To perform additions on binary bits the Arithmetic and Logic Unit present in the computer prefers this adder circuit.*
- *The combination of half adder circuits leads to the formation of the Full Adder circuit.*
- *These logic circuits are preferred in the design of calculators.*
- *To calculate the addresses and tables these circuits are preferred.*
- *Instead of only addition, these circuits are capable of handling various applications in digital circuits. Further, this becomes the heart of digital electronics.*

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.6

SR Flip-Flop Using NAND, NOR Gates

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: To implement and verify the operation of SR flip flop using MultiSim platform.

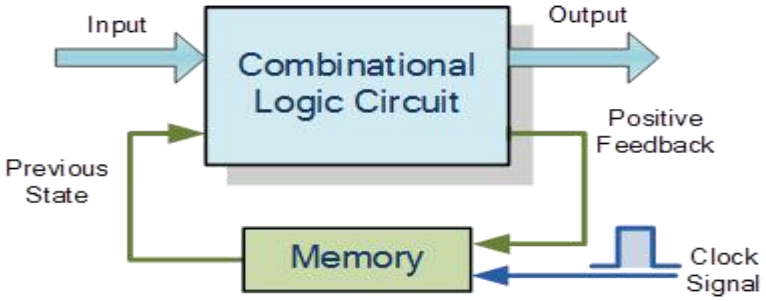
EQUIPMENT:

- Power supply unit (EV)
- Green and Red Probes

COMPONENTS FOR SR Flip-Flop:

- NAND Gate
- NOR Gate

What is sequential logic circuits?



Combinational logic circuits are the logic circuits whose output state depends on inputs at that time and any prior input state has no effect on present output because they have no memory element, but **sequential logic circuit** is combination of combinational logic circuit and memory element.

- Sequential logic circuits are the circuits whose output state is the function of the three states, the "present input", the "past input" and/or the "past output". **Sequential Logic circuits** have some form of inherent "Memory" built in to remember these conditions and stay fixed in their current state until the next clock signal changes one of the states.
- This memory is obtained by feedback connections, which are made so that history of the previous inputs is maintained.
- Simple sequential logic circuits can be constructed from standard bistable circuits such as: *Flip-flops*, *Latches* and *Counters* and which themselves can be made by simply connecting universal

NAND Gates and/or NOR Gates in a particular combinational way to produce the required sequential circuit.

Types of Sequential Circuits: There are two types of sequential circuit:

1. **Asynchronous sequential circuit:** These circuit do not use a clock signal but uses the pulses of the inputs. These circuits are faster than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal. But these circuits are more difficult to design, and their output is uncertain.
   Example: Latches.
2. **Synchronous sequential circuit:** These circuit uses clock signal and level inputs (or pulsed) (with restrictions on pulse width and circuit propagation). The output pulse is the same duration as the clock pulse for the clocked sequential circuits. Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit slower compared to asynchronous. Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.
   We use synchronous sequential circuit in synchronous counters, flip flops, and in the design of MOORE-MEALY state management machines.

SR Flip-Flop

Introduction

The **flip-flop** (abbreviated FF) can be considered as one of the most basic sequential logic circuit that is made up of assembly of logic gates that store one-bit memory. Even through a logic gate, by itself has no storage capacity; several different gates arrangements connected in a way that permit information to be stored.

FF is a bistable-multivibrator device that has the two stable states: HIGH or LOW.

Flip-flop can either be asynchronous circuits (un-clocked) i.e., **SR latches** or synchronous circuits (clocked or gated) i.e., **SR flip-flop**.

Flip-flop/Latches have two outputs Q and $\overline{Q}$ and under normal condition output will always be inverse of each other.

This asynchronous device consists of two inputs:

- SET input is the input that *sets* Q to the 1 state.
- RESET input is the input that re*sets* $\overline{Q}$ to the 0 state.

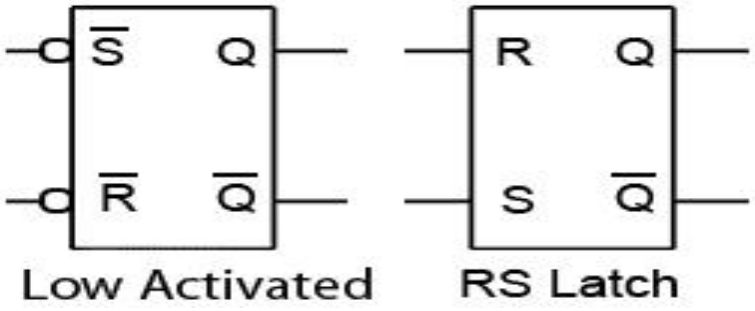
The SR stands for **SET/RESET.** The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level "1" or logic "0" depending upon this set/reset condition.

A basic SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs. Flip flop word means that it can be **"FLIPPED"** into one logic state or **"FLOPPED"** back into another.

The most used type of FF are:

1. NAND gate SR flip-flop (Active Low)
2. NOR gate SR flip-flop (Active High)

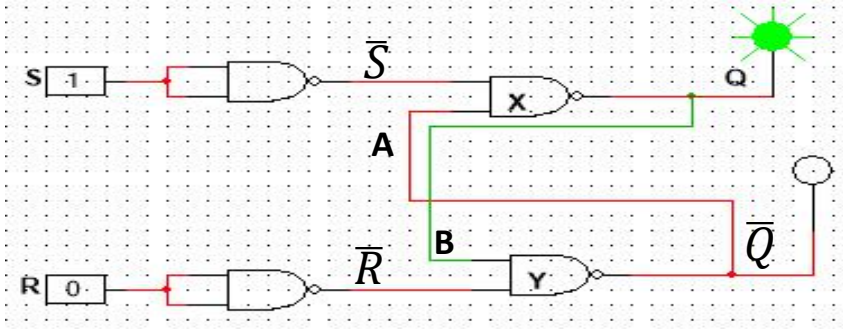
Block Diagram



SR Flip-Flop Designed Using NAND Gates:

A pair of cross-coupled 2-unit NAND gates is the simplest way to make any basic one-bit SR Flip Flop. It bi-stable or a **LOW RS NAND gate FF** and inputs are inverted by using **common input NAND gates** instead of using **inverter**. The feedback is fed from each output to one of the other NAND gate inputs.

Circuit Diagram



The Reset State

Considering the above circuit. If the input $\overline{R}$ is at logic level "0" ($\overline{R}$ = 0) and input $\overline{S}$ is at the logic "1" ($\overline{S}$ = 1), the NAND gate **Y** has, at least, one of its inputs at a logic "0". Therefore, its output $\overline{Q}$ must be at a logic level "1" (NAND gate principles). The Output ($\overline{Q}$) is fed back to the input "A". Both the inputs of the NAND gates X are at logic "1", and therefore, its output Q must be at the logic level "0".

The reset input $\overline{R}$ changes its state and goes HIGH to logic "1" with $\overline{S}$ constant at logic "1". The NAND gate **Y** input are now ($\overline{R}$ = 1) and (B= Q = 0). The output at $\overline{Q}$ remains at logic level "1" as one of its inputs is still at logic level "0" (Again, NAND gate principals). As a result, there is no change in state. Therefore, the flip-flop circuit is said to be "LATCHED" or "SET" with Q = 0 and $\overline{Q}$ = 1.

The Set State

In this second stable state, inputs are given by ($\overline{R}$ = 1) and ($\overline{S}$ = 0). As gate **X** has one of its inputs at a logic "0" its output Q must equal logic level "1". (According to the NAND

gate principle). The output Q is fed to input B, so both the inputs to NAND gate Y are at logic "1", therefore, $\overline{Q}$ = 0.

If the set input '$\overline{S}$' now changes the state to logic "1" with the input $\overline{R}$ remaining constant at logic "1", The NAND gate **X** input are now ($\overline{S}$ = 1) and (A = $\overline{Q}$ = 0). The output Q still remain High at logic level "1" as one of its inputs is still at logic level "0" (Again, NAND gate principals).

There is no change in the state. Therefore, the flip-flop circuit is said to be "LATCHED" or "RESET" with Q = 1 and $\overline{Q}$ = 0.

Memory Element

When both inputs $\overline{S}$ = "1" and $\overline{R}$ = "1" the outputs Q and $\overline{Q}$ can be at either logic level "1" or "0", depending upon the state of the inputs S or R **BEFORE** this input condition existed. Therefore, the condition of S = R = "1" does not change the state of the outputs Q and $\overline{Q}$. This condition is memory element.

Meta-stable state

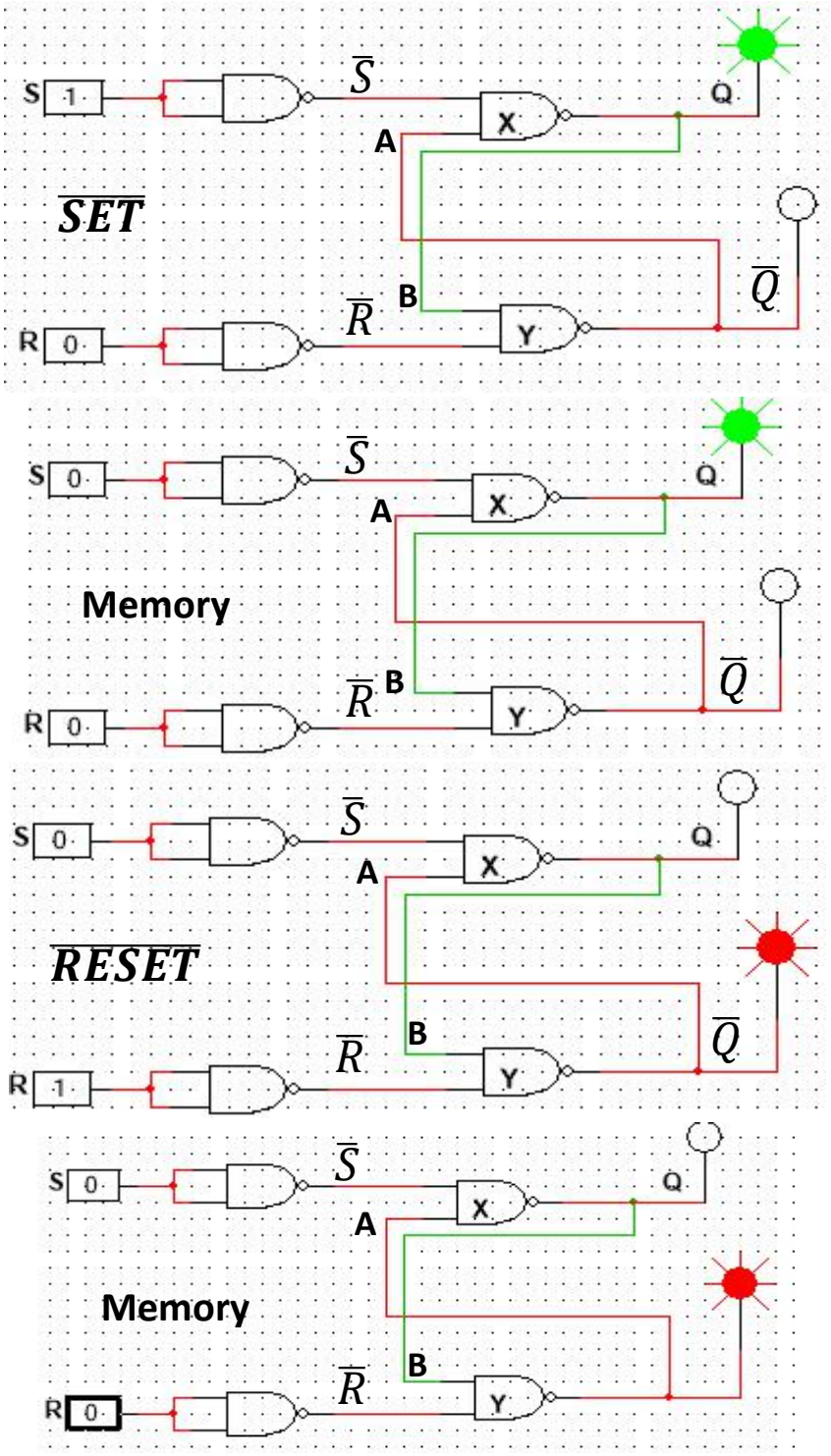
However, the input state of $\overline{S}$ = "1" and $\overline{R}$ = "1" is an undesirable or invalid condition and must be avoided. The condition of S = R = "0" causes both outputs Q and $\overline{Q}$ to be HIGH together at logic level "1" when we would normally want Q to be the inverse of $\overline{Q}$. The result is that the flip-flop loses control of Q and $\overline{Q}$, and if the two inputs are now switched "HIGH" or to logic "1" after this condition the flip-flop becomes unstable and data corruption will exist. This unstable condition is generally known as its **Meta-stable state.**

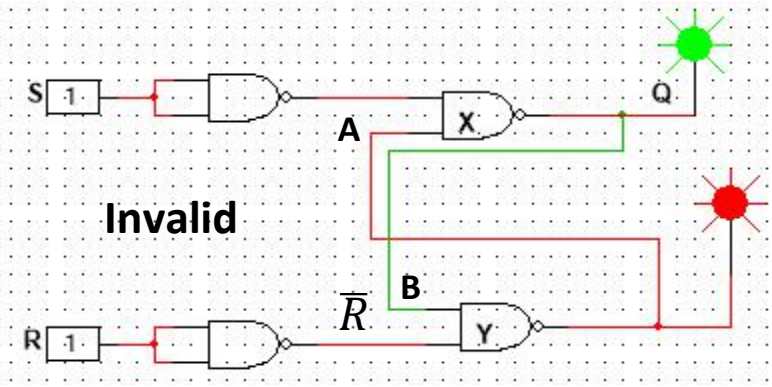
Truth Table:

Input				Output		State
S	R	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	-
0	1	1	0	0	1	Reset
0	0	1	1	0	1	Memory
1	0	0	1	1	0	Set
0	0	1	1	1	0	Memory
1	1	0	0	1	1	Invalid

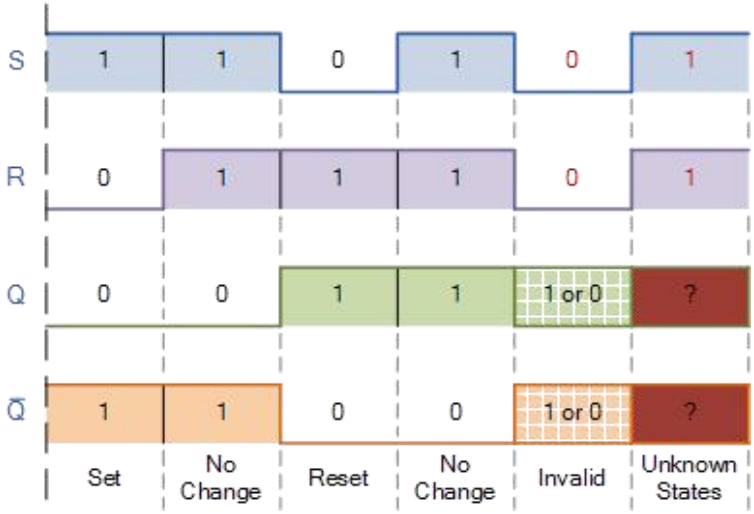
Simulated Circuit:

Verification through simulation.





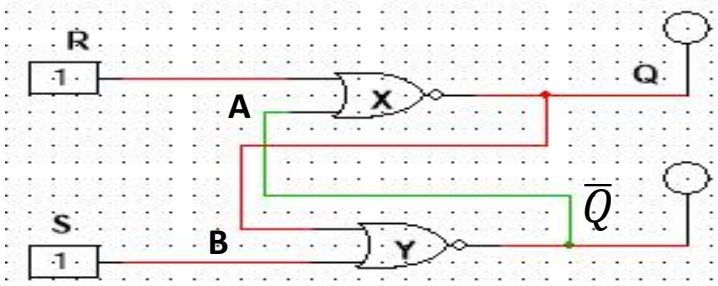
Timing Waveform



RS Flip-Flop Designed Using NOR Gates:

A pair of cross-coupled 2-unit NOR gates is the simplest way to make any basic one-bit RS Flip Flop. It bi-stable or an **ACTIVE HIGH RS NAND gate FF**. The feedback is fed from each output to one of the other NOR gate inputs.

Circuit Diagram



The Set State

Considering the above circuit. If the input S is at logic level "1" (S = 1) and input R is at the logic "0" (R = 0), the NOR gate **Y** has, at least, one of its inputs at a logic "1". Therefore, its output $\overline{Q}$ at a logic level "0" (NOR gate principles). The Output ($\overline{Q}$) is fed

back to the input "A". Both the inputs of the NOR gates **X** are at logic "0", therefore, its output Q must be at the logic level "1".
The set input S changes its state and goes LOW to logic "0" with R constant at logic "0". The NOR gate **Y** input are now (S = 0) and (B = Q = 1). The output at $\overline{Q}$ remains at logic level "0" as one of its inputs is still at logic level "1" (Again, NOR gate principals).
As a result, there is no change in state. Therefore, the flip-flop circuit is said to be "LATCHED" or "SET" with Q = 1 and $\overline{Q}$ = 0.

The Reset State

In this second stable state, inputs are given by (R = 1) and (S = 0). As gate **X** has one of its inputs at a logic "1" its output Q equal logic level "0". (According to the NOR gate principle). The output Q is fed to input B, so both the inputs to NOR gate **Y** are at logic "0", therefore, $\overline{Q}$ = 1.
If the reset input R now changes the state to logic "0" with the input S remaining constant at logic "0", The NOR gate **X** input are now (R = 0) and (A = $\overline{Q}$ = 1). The output Q remain LOW at logic level "0" as one of its inputs is still at logic level "1" (Again, NOR gate principals).
There is no change in the state. Therefore, the flip-flop circuit is said to be "LATCHED" or "RESET" with Q = 0 and $\overline{Q}$ = 1.

Memory Element

When both inputs S = "0" and R = "0" the outputs Q and $\overline{Q}$ can be at either logic level "1" or "0", depending upon the state of the inputs S or R **BEFORE** this input condition existed. Therefore, the condition of $S = R =$ "0" does not change the state of the outputs Q and $\overline{Q}$. This condition is memory element.

Meta-stable state

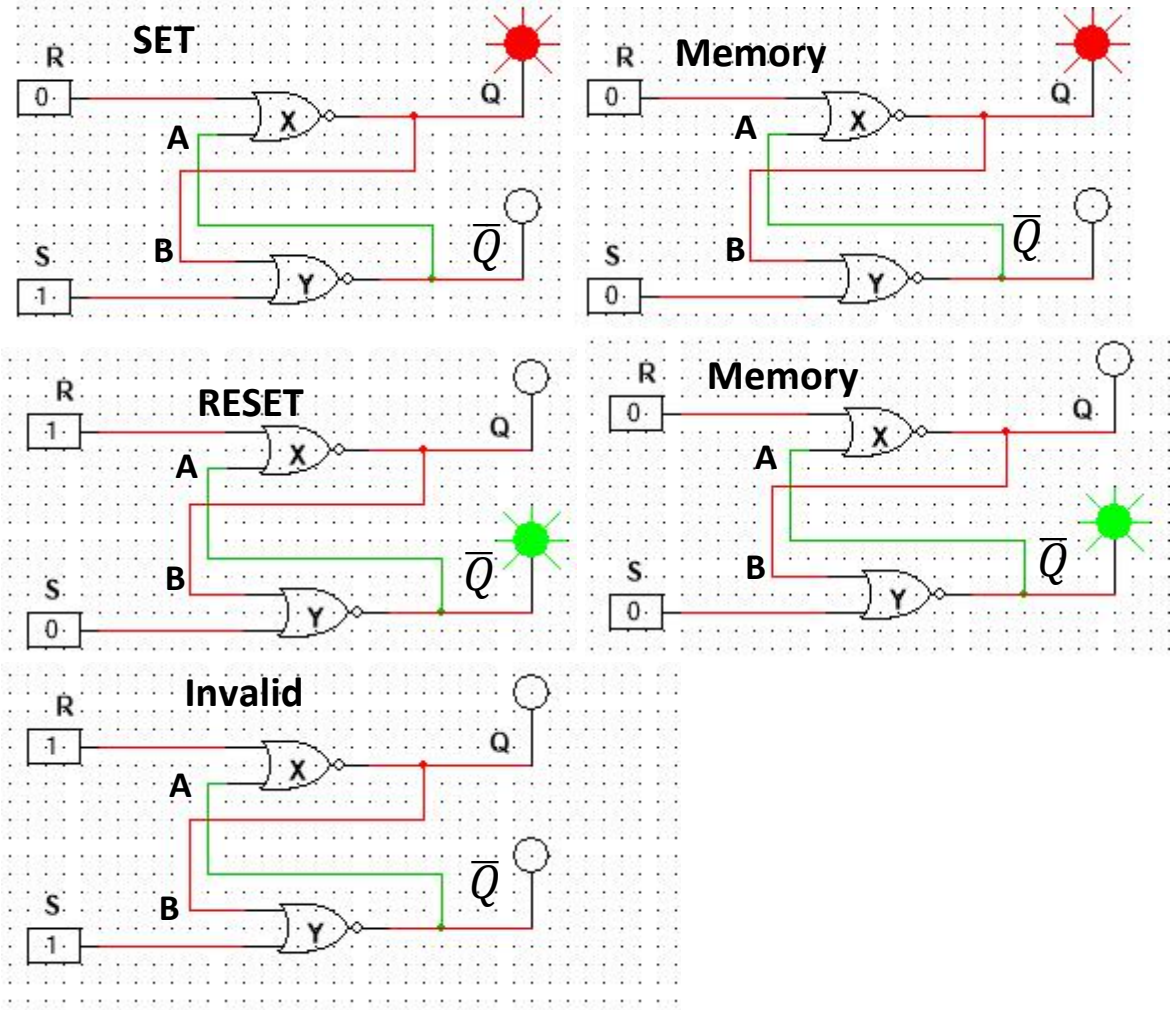
However, the input state of S = "1" and R = "1" is an undesirable or invalid condition and must be avoided. The condition of $S = R =$ "1" causes both outputs Q and $\overline{Q}$ to be LOW together at logic level "0" when we would normally want Q to be the inverse of $\overline{Q}$. The result is that the flip-flop loses control of Q and $\overline{Q}$, and if the two inputs are now switched "LOW" or to logic "0" after this condition the flip-flop becomes unstable and data corruption will exist. This unstable condition is generally known as its **Meta-stable state.**

Truth Table:

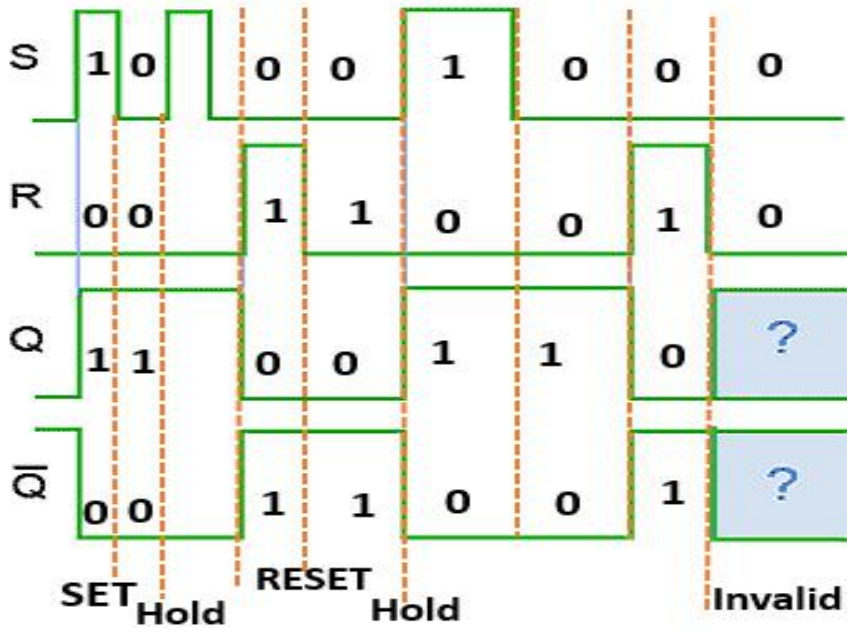
Input		Output		State
R	S	Q	$\overline{Q}$	-
0	1	1	0	Set
0	0	1	0	Memory
1	0	0	1	Reset
0	0	0	1	Memory
1	1	0	0	Invalid

Simulated Circuit:

Verification through simulation.



Timing Waveform



Application Of SR Flip-Flop

- *Generally, latches are used to keep the conditions of the bits to encode binary numbers.*
- *Latches are single bit storage elements which are widely used in computing as well as data storage.*
- *Latches are used in the circuits like power gating & clock as a storage device.*
- *D latches are applicable for asynchronous systems like input or output ports.*
- *Data latches are used in synchronous two-phase systems for reducing the transit count.*

Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.7
Gated SR Flip-Flop Using NAND, NOR Gates

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: To implement and verify the operation of Gated SR flip flop using MultiSim platform.

EQUIPMENT:

- Power supply unit (EV)
- Green and Red Probes

COMPONENTS FOR SR Flip-Flop:

- NAND Gate
- AND Gate
- NOR Gate

Gated SR Flip-Flop

Introduction:

A gated SR flip-flop can change its state of outputs only when certain conditions are met, regardless of its S and R inputs state. The conditional input is called **enable** signal. For this reason, it is also known as a **synchronous SR Flip-flop**.

The flip-flop is active when enable signal is **HIGH** and it is inactive when **enable** signal is **LOW**. This **HIGH LOW** enable signal is applied to the gated latch in the form of **clocked pulses**. The clock signal is generally a **rectangular pulse train or a square wave** and depends on **transition of clock signal Flip-Flops** are designed as:

- When the clock changes from a LOW state to a HIGH state are known as **Positive-Going Transition (PGT) SR flip-flop**.
- When the clock changes from a HIGH state to a LOW state are known as **Negative-Going Transition (NGT) SR flip-flop**.

Clocked Flip-flops have two outputs Q and $\overline{Q}$ and under normal condition output will always be inverse of each other.

This synchronous device consists of three inputs:

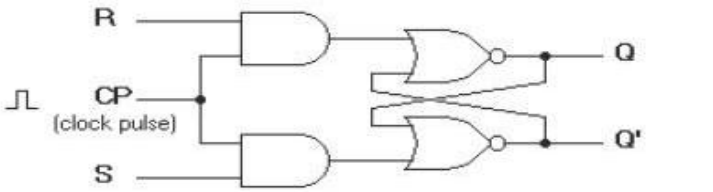
- Clock input that is responsible for activation of flip-flop.
- SET input is the input that *sets* Q to the 1 state.
- RESET input is the input that re*sets* $\overline{Q}$ to the 0 state.

The SR stands for **SET/RESET.** The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level "1" or logic "0" depending upon this set/reset condition.

Gated SR Flip-Flop using AND-NOR Gate

Introduction:

A Gated S-R flip flop is designed by 2-input AND gate in series with each input terminal of the SR Flip-flop. As shown below:

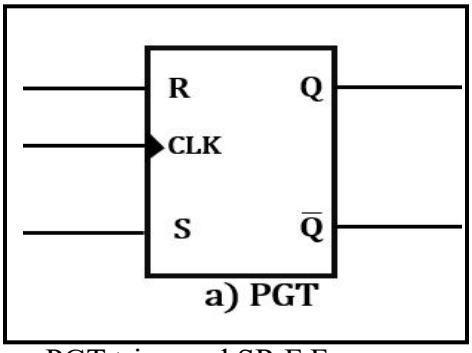


(a) Logic diagram

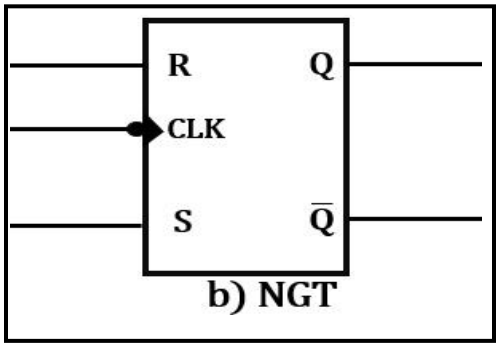
Clocked S-R Flip Flop

Block Symbol:

These SR Flip-Flop either is PGT or NGT as shown:



a) PGT



b) NGT

Theory PGT triggered SR F.F:

When the clock input "CLK" is at logic level "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, gated SR flip-flop remain into their last known state. When the **clock input "CLK" changes to logic level "1"** the **circuit responds** as a **PGT SR bistable flip-flop** with **the two AND gates becoming transparen**t to the Set and Reset signals. Hence this kind of clocked SR flip flop that is triggered by the **Positive Going Transition**. Its means that the flip flop can change the output states only when clock signal makes a transition from **LOW to HIGH**.

But when the values of both S and R values turn '1', the **HIGH** value of CLK causes both to turn to '0' for a short moment. As soon as the pulse is removed, the flip flop state becomes intermediate.

Truth Table:

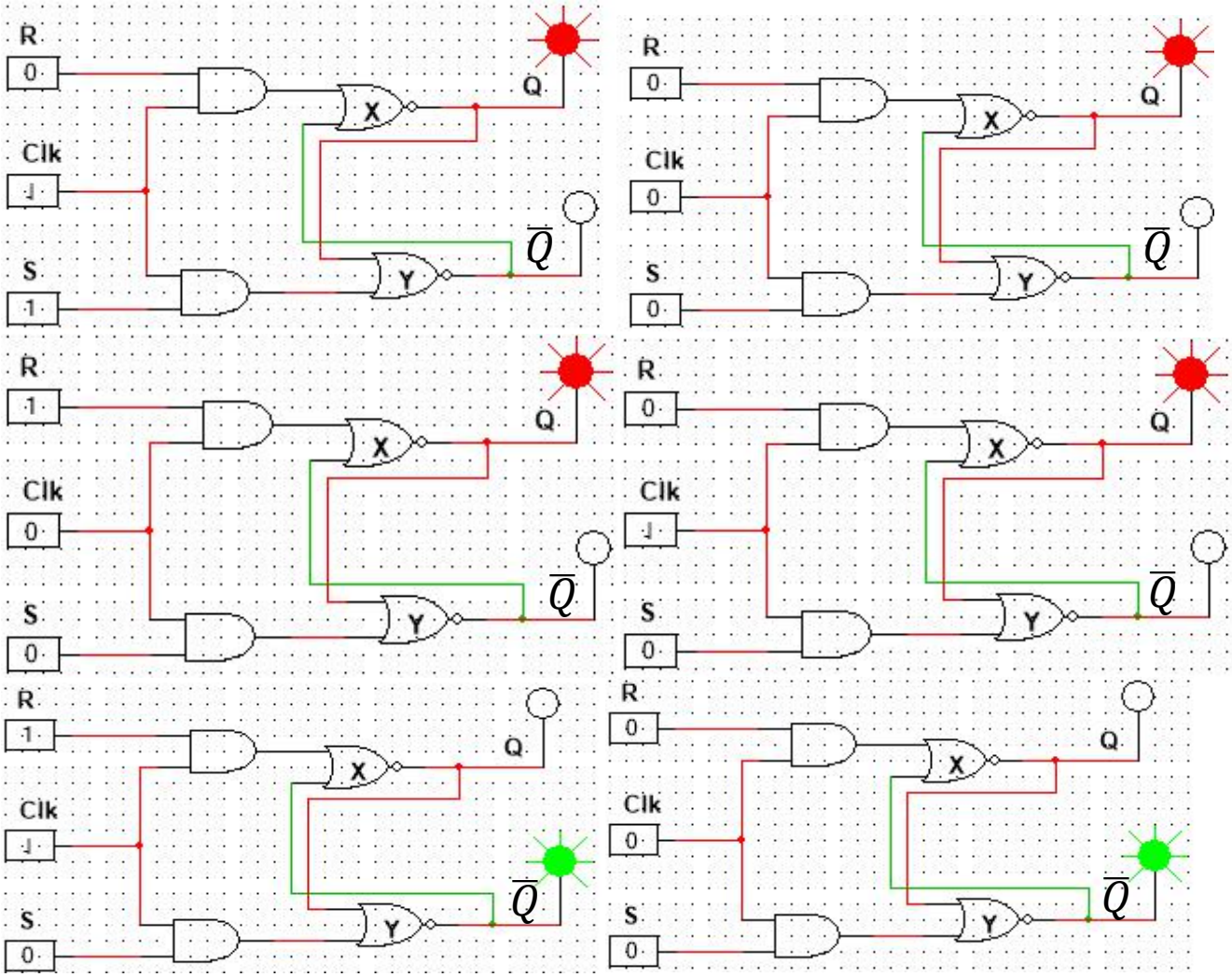
Input			Output		State
R	S	CLK	Q	$\overline{Q}$	-
0	1	↑	1	0	Set
0	0	↓	1	0	Hold

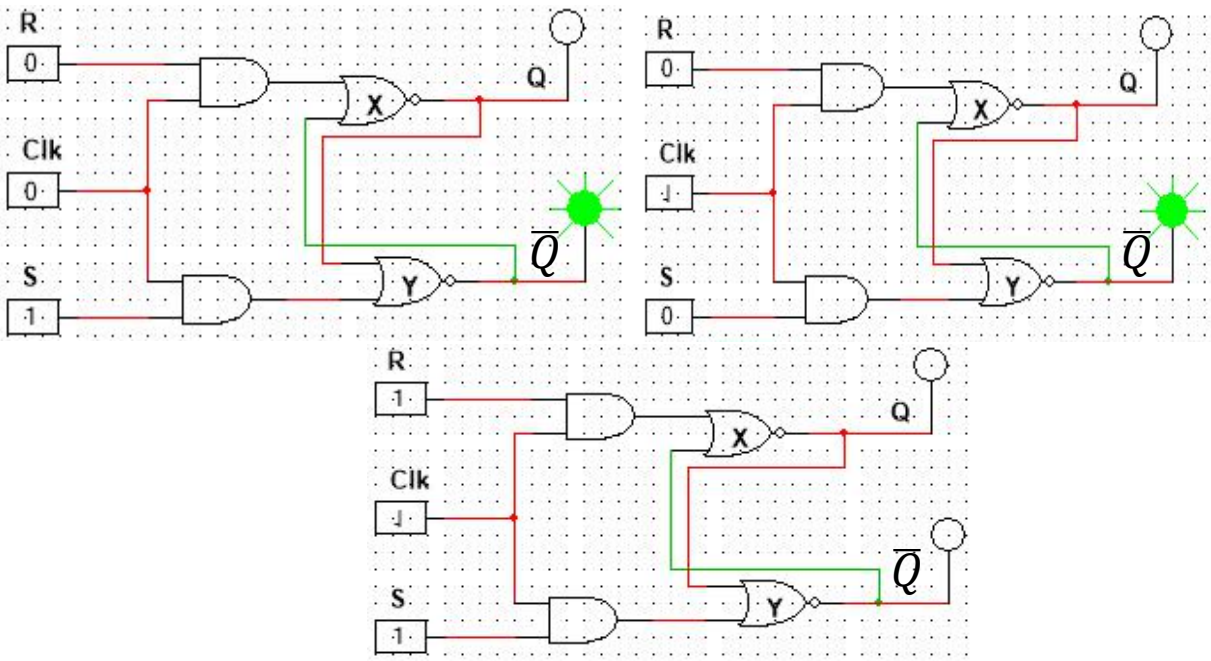
I) Truth Table for PGT triggered SR F.F

1	0	↓	1	0	Hold
0	0	↑	1	0	Memory
1	0	↑	0	1	Reset
0	0	↓	0	1	Hold
0	1	↓	0	1	Hold
0	0	↑	0	1	Memory
1	1	↑	x	x	Invalid

Simulated Circuit:

Verification of truth table for PGT triggered SR F.F.





Conclusion:

It should be noted from truth table that FF operation is only affected when there is PGT of clock pulse as well as operation of RS is not affected by R and S inputs. PGT of clock pulse trigger to change the state of FF according to what R and S input are. When S=R=1, condition is invalid.

NGT triggered SR F.F:

When the clock input "CLK" is at logic level "1", it is inverted to logic "0" due to presence of inverter in circuit of NGT triggered SR flip-flop, the outputs of the two AND gates are at logic level "0", (AND Gate principles) regardless of the condition of the two inputs S and R, gated SR flip-flop remain into their last known state. When the ***clock input "CLK" to logic level "0"*** *the* ***circuit responds*** *as a* ***NGT SR bistable flip-flop*** *with* ***the two AND gates becoming transparent*** *to the Set and Reset signals. Hence this kind of SR flip flop that is triggered by the* ***Negative Going Transition****. Its means that the flip flop can change the output states only when clock signal makes a transition from* ***HIGH to LOW****. But when the values of both S and R values turn '1', the* ***LOW*** *value of CLK causes both to turn to '0' for a short moment. As soon as the pulse is removed, the flip flop state becomes intermediate.*

The timing waveform shows how the flip flop output will respond to the PGT at the clocked input for the various combinations of SR inputs and output. Also, S and R inputs have no effect on Flip-flop accept occurrence of PGT of clock pulse.

Truth Table:

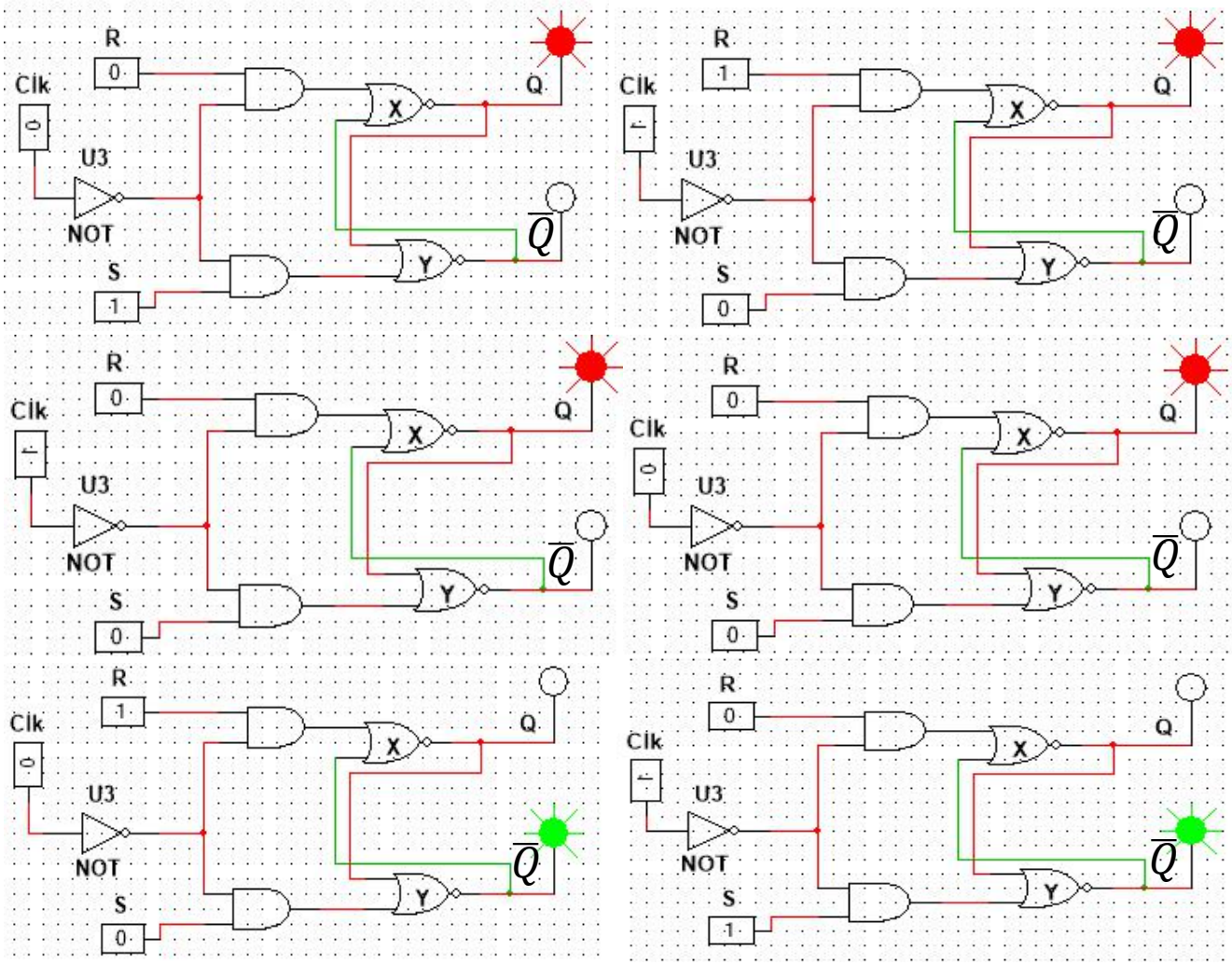
Input				Output		State
R	S	CLK	$\overline{CLK}$	Q	$\overline{Q}$	-

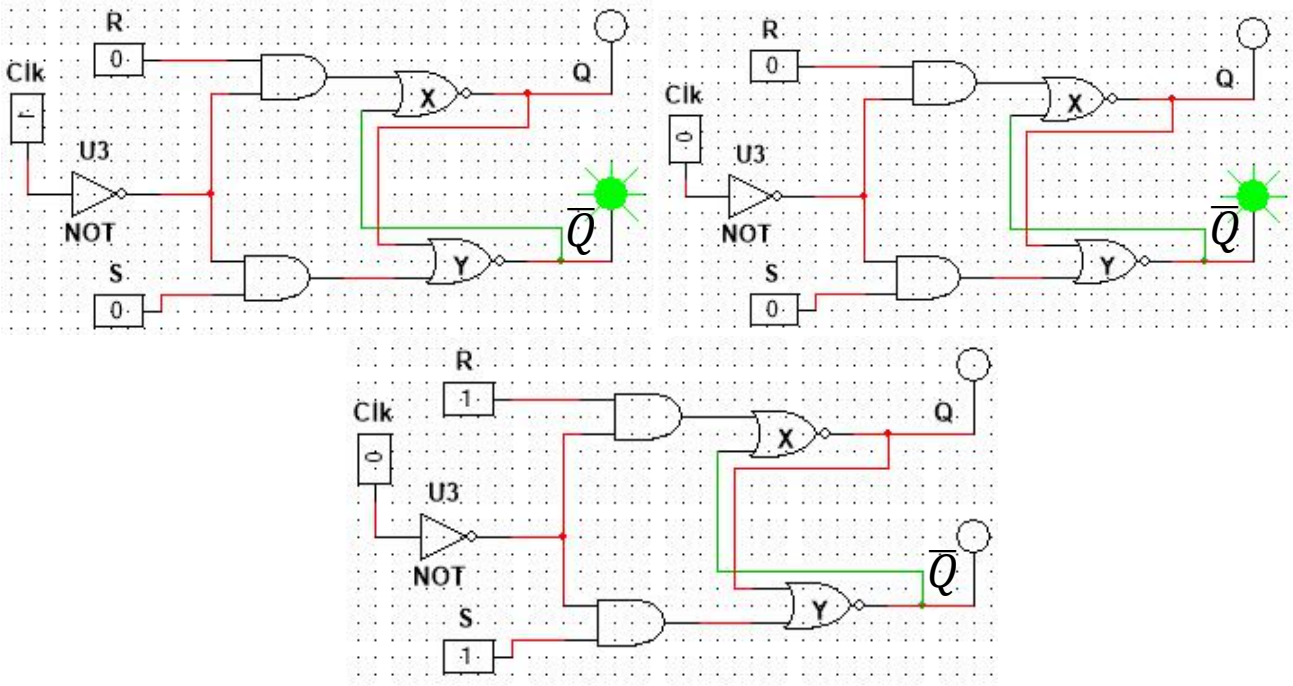
II) Truth Table for NGT triggered SR F.F

0	1	↓	↑	1	0	Set
0	0	↑	↓	1	0	Hold
1	0	↑	↓	1	0	Hold
0	0	↓	↑	1	0	Memory
1	0	↓	↑	0	1	Reset
0	1	↑	↓	0	1	Hold
0	0	↑	↓	0	1	Hold
0	0	↓	↑	0	1	Memory
1	1	↓	↑	x	X	Invalid

Simulated Circuit:

Verification of truth table for NGT triggered SR F.F.





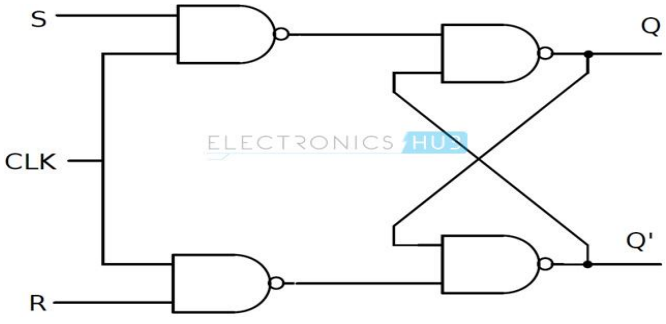
Conclusion:

It should be noted from truth table that FF operation is only affected when there is NGT of clock pulse as well as operation of RS is not affected by R and S inputs. NGT of clock pulse trigger to change the state of FF according to what R and S input are. When S=R=1, condition is invalid.

Gated SR Flip-Flop using NAND Gate

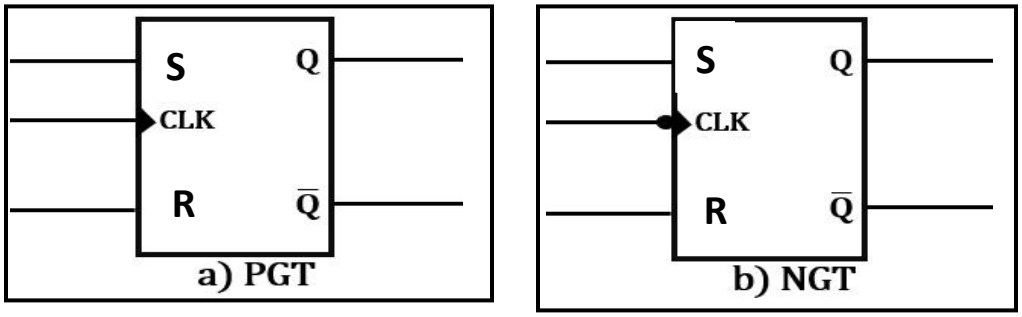
Introduction:

This circuit is formed by adding two NAND gates to NAND based SR flip – flop. The inputs are active high as the extra NAND gate invert the inputs. A clock pulse is given as input to both the extra NAND gates. The circuit of clocked SR flip – flop using NAND gates is shown below:



Block Symbol:

These SR Flip-Flop either is PGT or NGT as shown:



Theory PGT triggered SR F.F:

When the clock input "CLK" is at logic level "0", the outputs of the two NAND gates are also at logic level "1", (NAND Gate principles) regardless of the condition of the two inputs S and R, gated SR flip-flop remain into their last known state. When the **clock input "CLK" changes to logic level "1"** the **circuit responds** as a **PGT SR bistable flip-flop**. Hence, the kind of clocked SR flip flop that is triggered by the **Positive Going Transition**. Its means that the flip flop can change the output states only when clock signal makes a transition from **LOW to HIGH**.

But when the values of both $\overline{S}$ and $\overline{R}$ values turn '0', the **HIGH** value of CLK causes both to turn to '1' for a short moment. As soon as the pulse is removed, the flip flop state becomes intermediate.

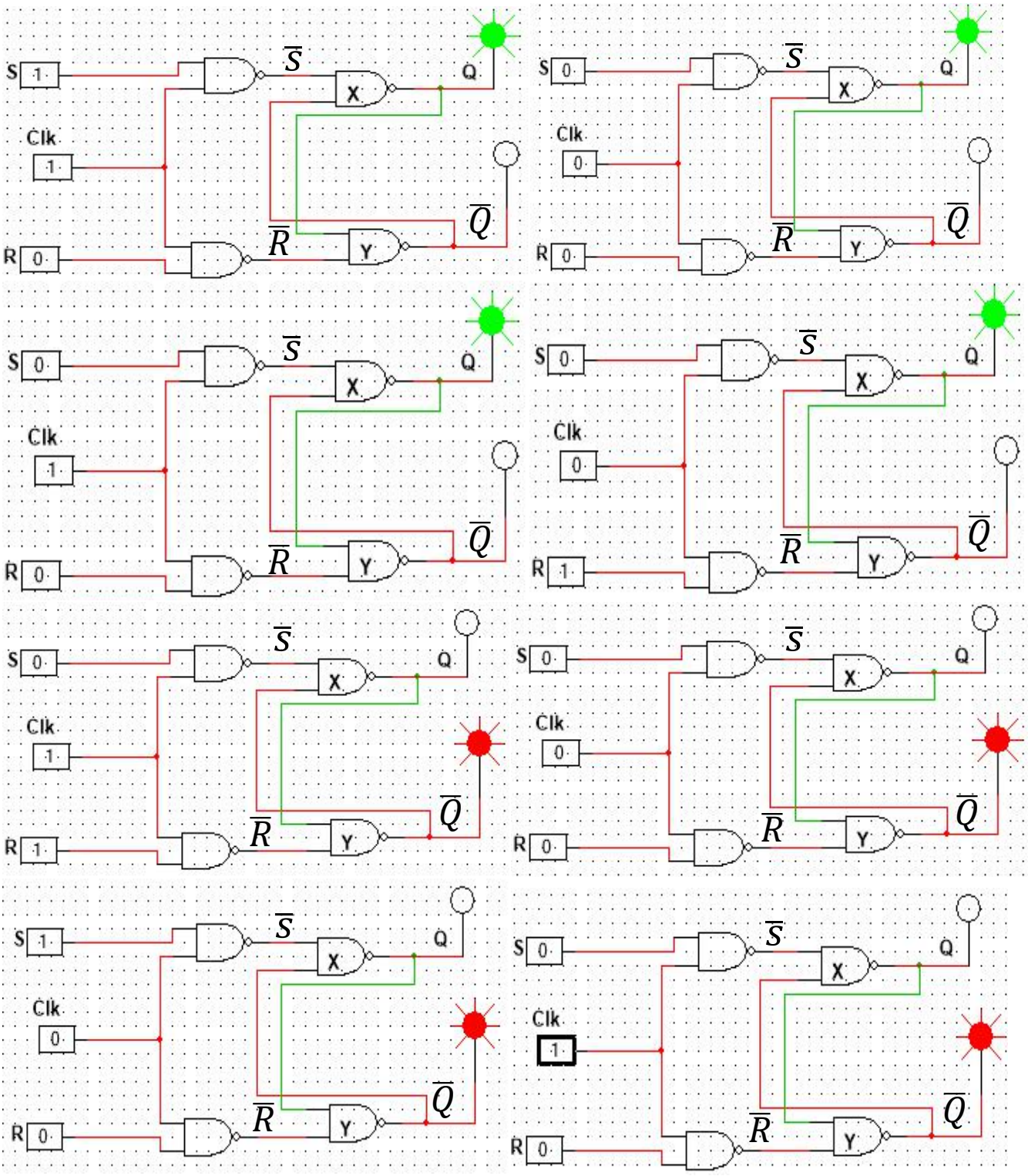
Truth Table:

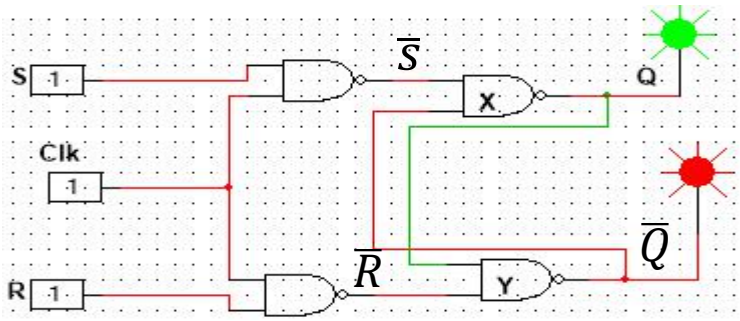
I) Truth Table for PGT triggered SR F.F

Input					Output		State
S	R	CLK	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	-
1	0	↑	0	1	1	0	Set
0	0	↓	1	1	1	0	Hold
0	1	↓	1	0	1	0	Hold
0	0	↑	1	1	1	0	Memory
0	1	↑	1	0	0	1	Reset
0	0	↓	1	1	0	1	Hold
1	0	↓	0	1	0	1	Hold
0	0	↑	1	1	0	1	Memory
1	1	↑	0	0	1	1	Invalid

Simulated Circuit:

Verification of truth table for PGT triggered SR F.F.





Conclusion:

It should be noted from truth table that FF operation is only affected when there is PGT of clock pulse as well as operation of SR Flip-Flop is not affected by S and R inputs. PGT of clock pulse trigger to change the state of FF according to what S and R input are. When $\overline{S} = \overline{R} = 0$*, condition is invalid.*

Theory NGT triggered SR F.F:

When the clock input "CLK" is at logic level "0" it is inverted to logic "1" due to presence of common input NAND gate in circuit of NGT triggered SR flip-flop, the outputs of the two NAND gates are also at logic level "1", (NAND Gate principles) regardless of the condition of the two inputs S and R, gated SR flip-flop remain into their last known state. When the **clock input "CLK" changes to logic level "0"** the **circuit responds** as a **NGT SR bistable flip-flop**. Hence, the kind of clocked SR flip flop that is triggered by the **Negative Going Transition**. Its means that the flip flop can change the output states only when clock signal makes a transition from **HIGH to LOW.**

But when the values of both $\overline{S}$ and $\overline{R}$ values turn '0', the **LOW** value of CLK causes both to turn to '1' for a short moment. As soon as the pulse is removed, the flip flop state becomes intermediate.

Truth Table:

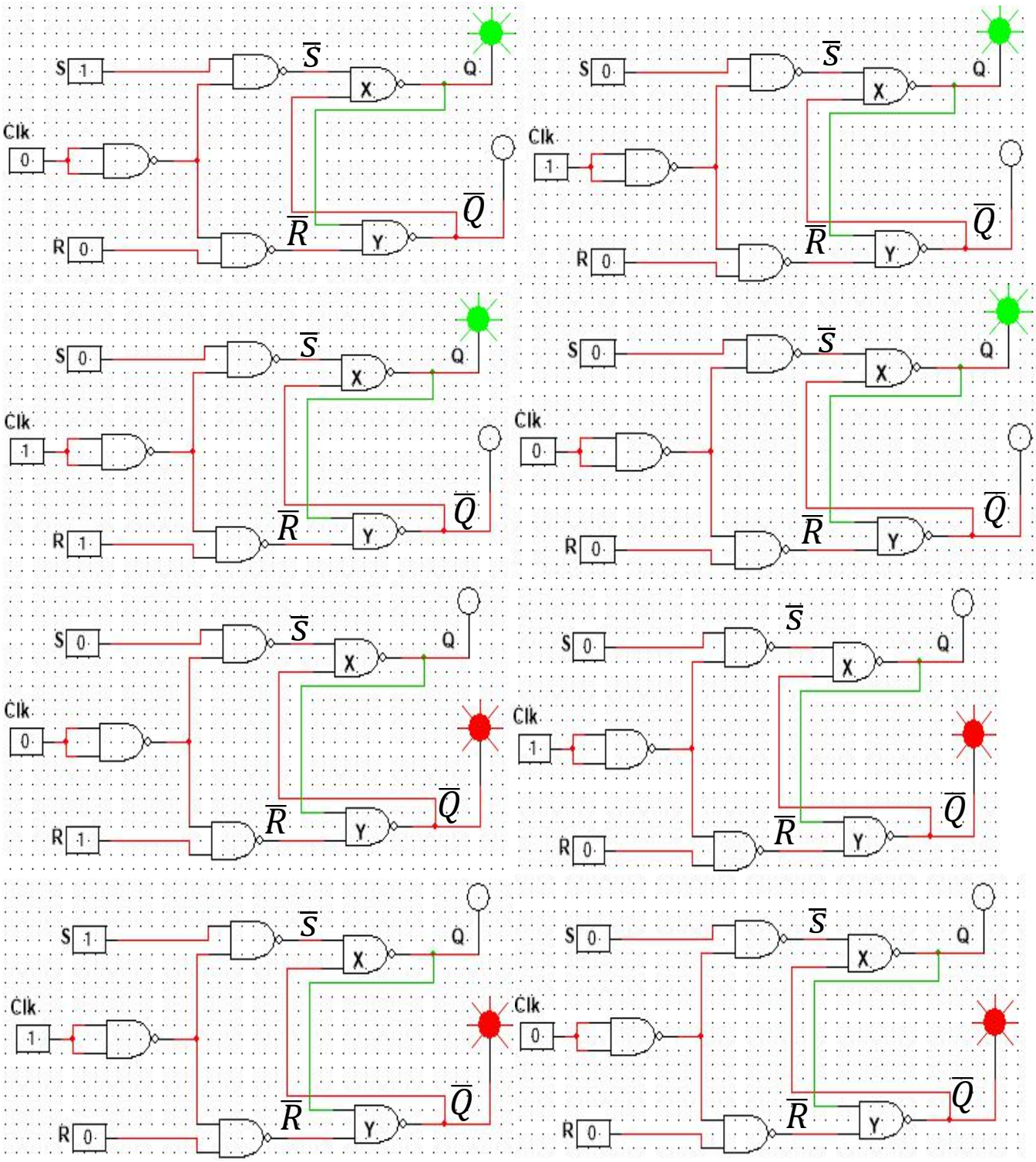
II) Truth Table for NGT triggered SR F.F

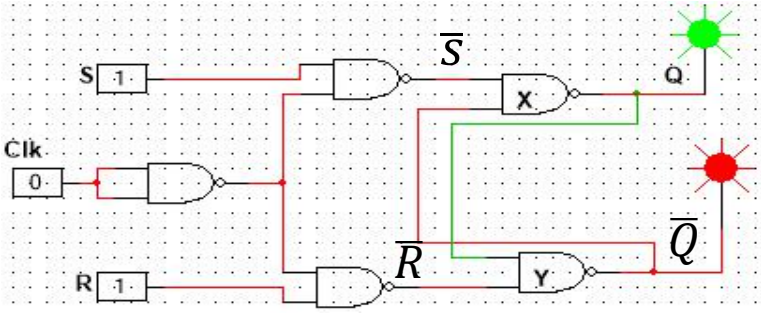
Input						Output		State
S	R	CLK	$\overline{CLK}$	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	-
1	0	↓	↑	0	1	1	0	Set
0	0	↑	↓	1	1	1	0	Hold
0	1	↑	↓	1	0	1	0	Hold
0	0	↓	↑	1	1	1	0	Memory
0	1	↓	↑	1	0	0	1	Reset
0	0	↑	↓	1	1	0	1	Hold
1	0	↑	↓	0	1	0	1	Hold
0	0	↓	↑	1	1	0	1	Memory

1	1	↓	↑	0	0	1	1	Invalid

Simulated Circuit:

Verification of truth table for NGT triggered SR F.F.

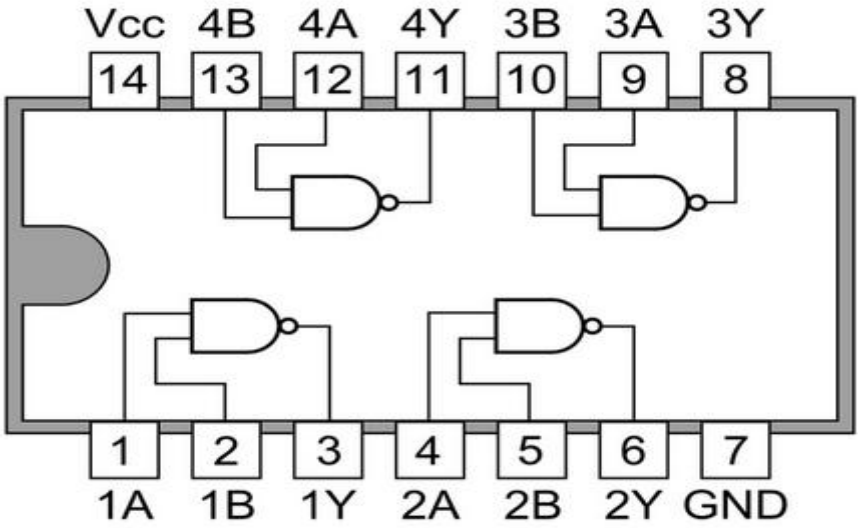




Conclusion:

It should be noted from truth table that FF operation is only affected when there is NGT of clock pulse as well as operation of SR Flip-Flop is not affected by S and R inputs. NGT of clock pulse trigger to change the state of FF according to what S and R input are. When $\overline{S} = \overline{R} = 0$*, condition is invalid.*

7400 Quad 2-input NAND Gates



Applications of Flip Flops

Application of the flip flop circuit mainly involves in bounce elimination switch, data storage, data transfer, latch, JK Flip-Flop, D Flip-Flop, T Flip-Flop, registers, counters, frequency division, memory, etc.

Department of Electronic Engineering

DIGITAL ELECTRONICS

Lab NO. 8

Gated D Flip-Flop Using NAND, NOR Gates

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: To implement and verify the operation of gated/Clocked D flip-flop using MultiSim platform.

Clocked D Flip Flop

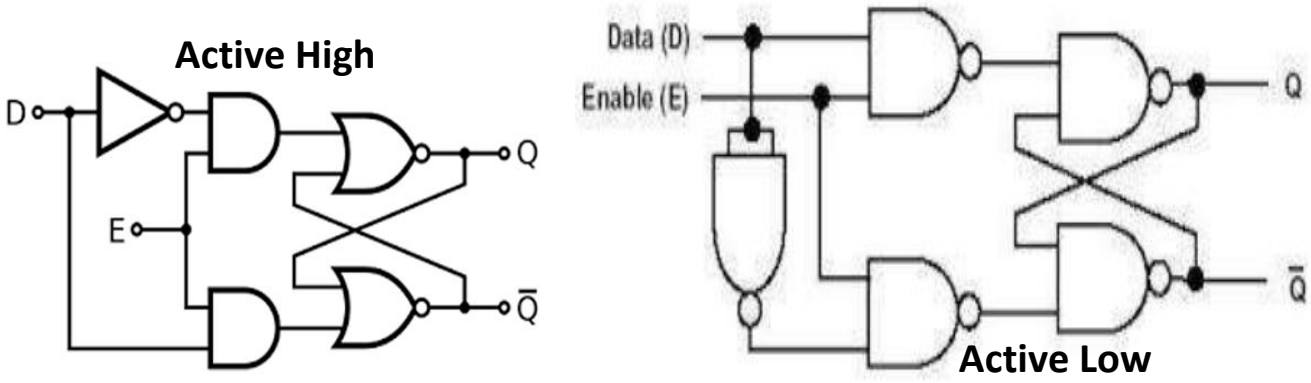
Introduction:

A D Flip Flop is a type of flip flop that tracks the input, making transitions with match those of the input D. The D stands for 'data or delay'; this flip-flop stores the value that is on the data line.

The D flip flop ensures that at the same time, both the inputs, i.e., S and R, are never equal to 1 or 0 so there will be never ***No Change*** **or** ***Invalid*** **output condition** there will be either **SET or RESET** conditions.

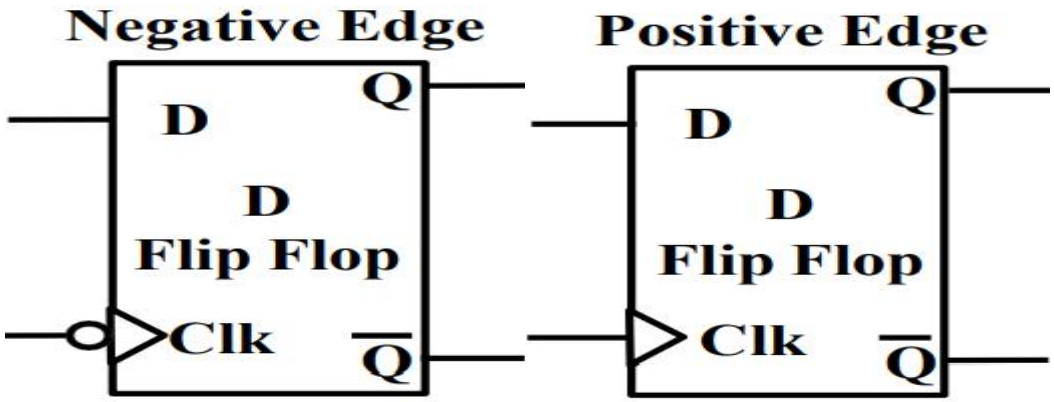
Construction:

The **D flip-flop** is designed using **a gated SR flip-flop with an inverter** connected between the inputs allowing for a single input D (Data) as shown below:



This single data input, which is labeled as "D" used in place of the "Set" input and "Reset" input is achieved by inverting "Set" input i.e., **S = D & R = $\bar{D}$**.

Block Symbol:



Truth Table:

Truth table are common for both D flip flop using NAND gate & NOR gate.

Input		Output		State
D	CLK	Q	$\overline{Q}$	-
X	↓	Q	$\overline{Q}$	No Change
1	↑	1	0	Set
0	↑	0	1	Reset

I) Truth Table for PGT triggered D F.F

Input		Output		State
D	CLK	Q	$\overline{Q}$	-
X	↑	Q	$\overline{Q}$	No Change
1	↓	1	0	Set
0	↓	0	1	Reset

II) Truth Table for NGT triggered D F.F

-
- Symbols ↓ and ↑ indicates the direction of the clock pulse. X indicate whatever the input is data will not change until the designed clock pulse is not reached.

Simulated Circuits:

Verification of truth table when Clock is **HIGH** i.e., PGT D Flip-Flop.

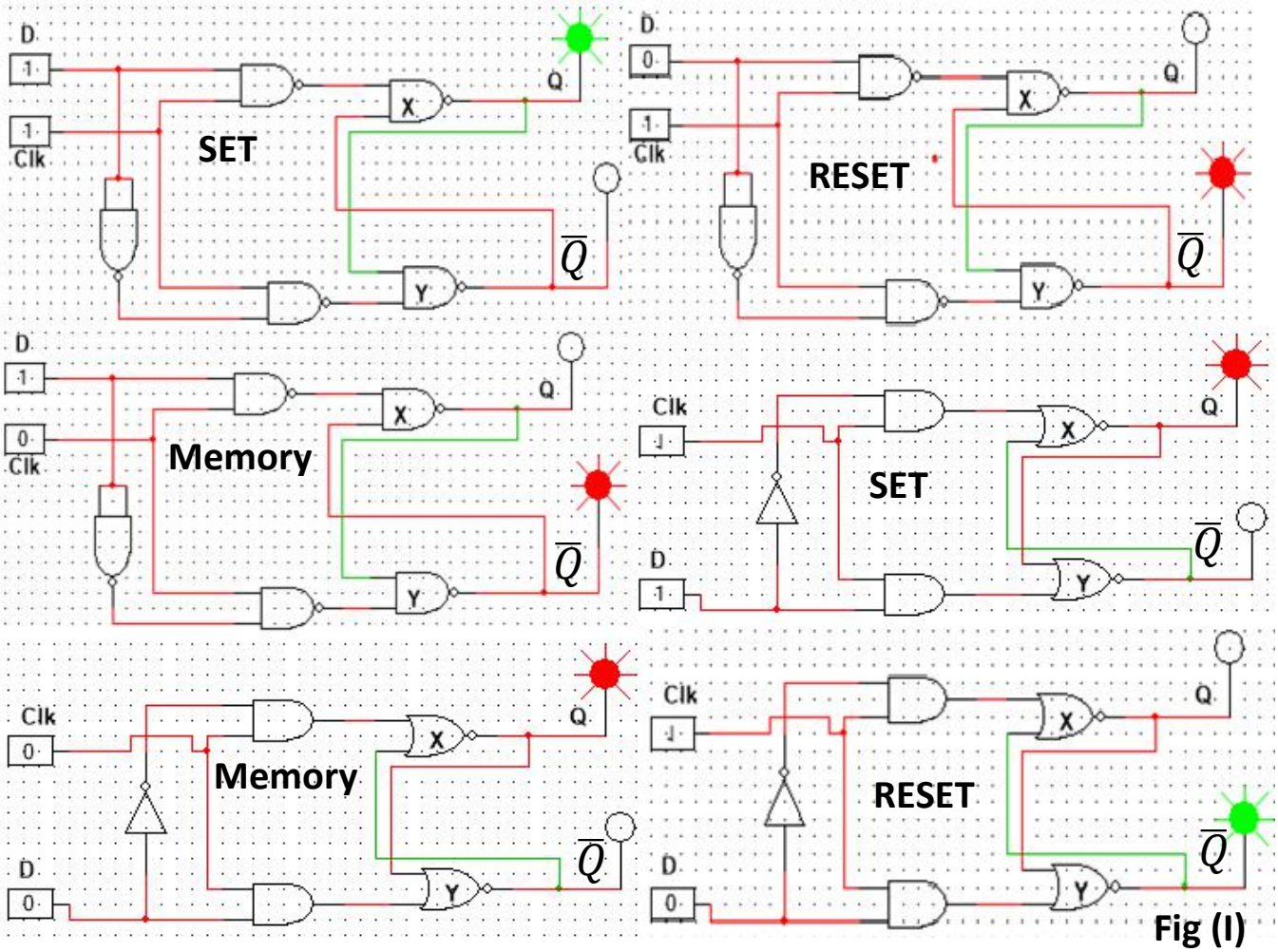
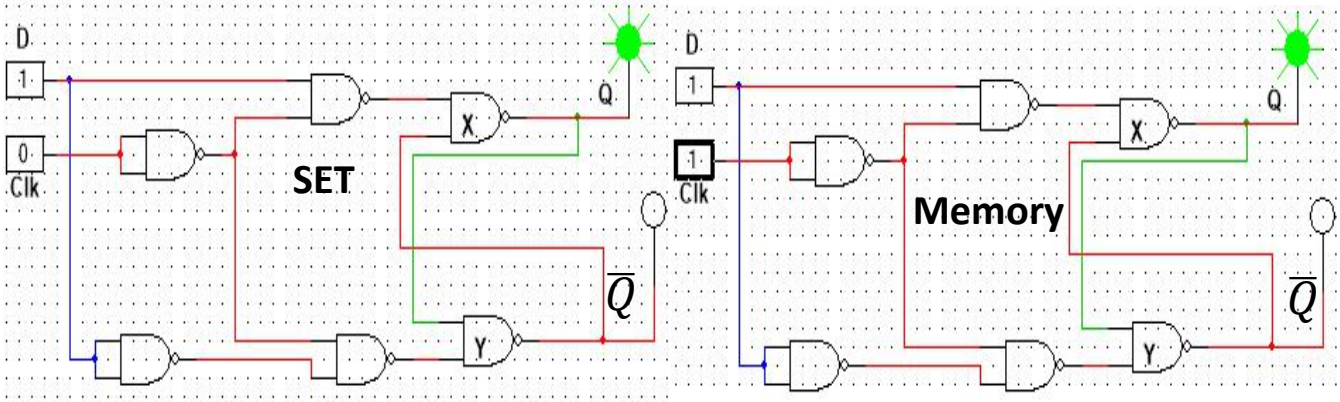


Fig (I)

Verification of truth table when Clock is **LOW** i.e., NGT D Flip-Flop.



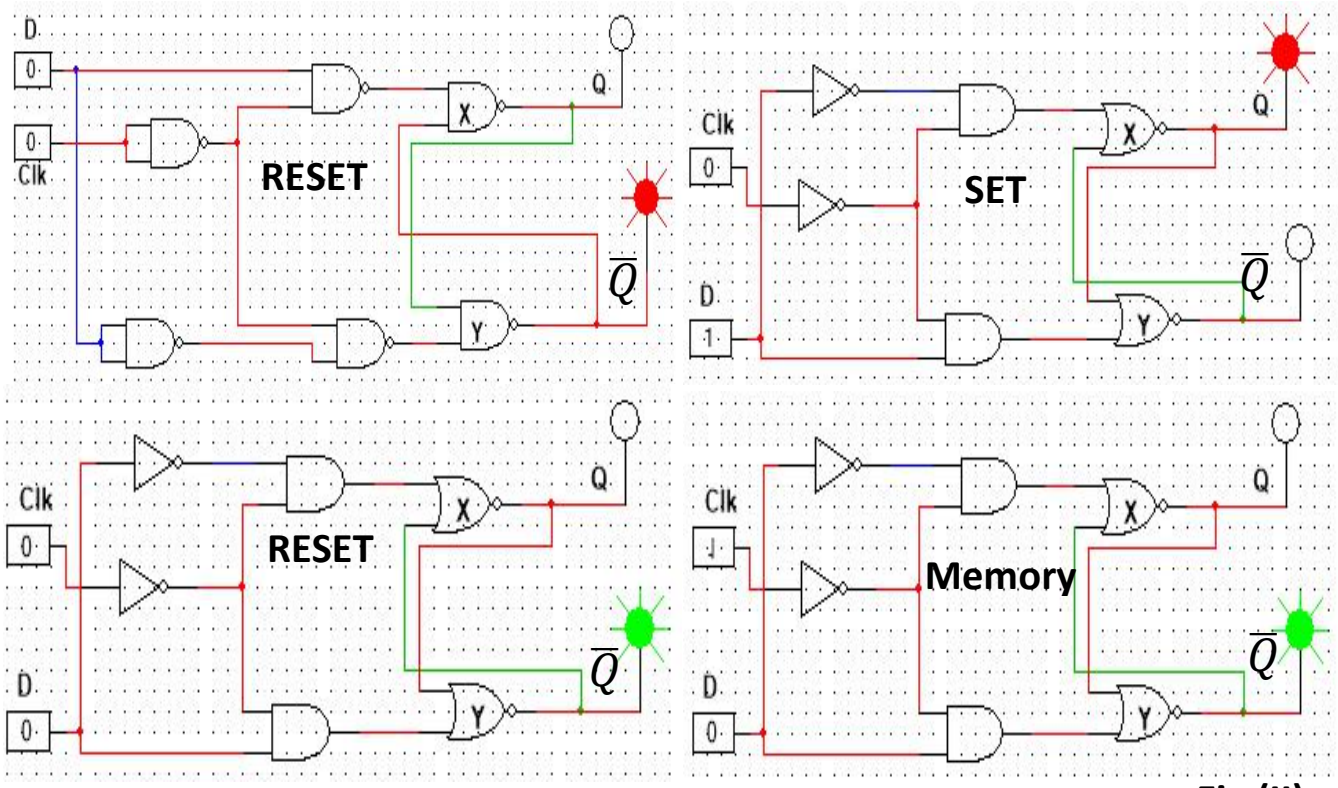


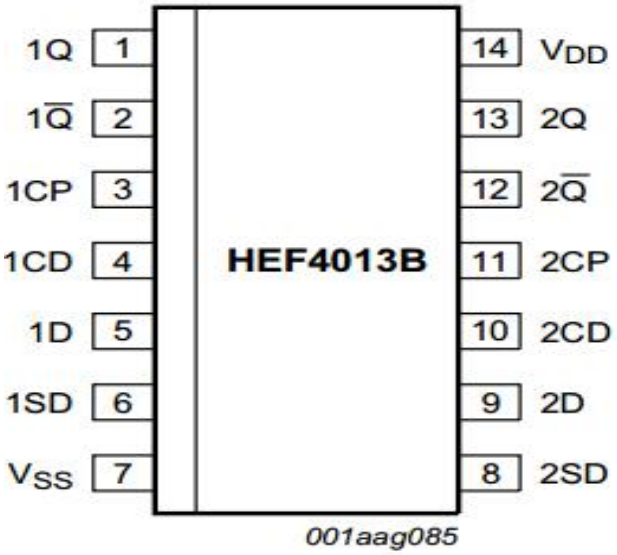
Fig (II)

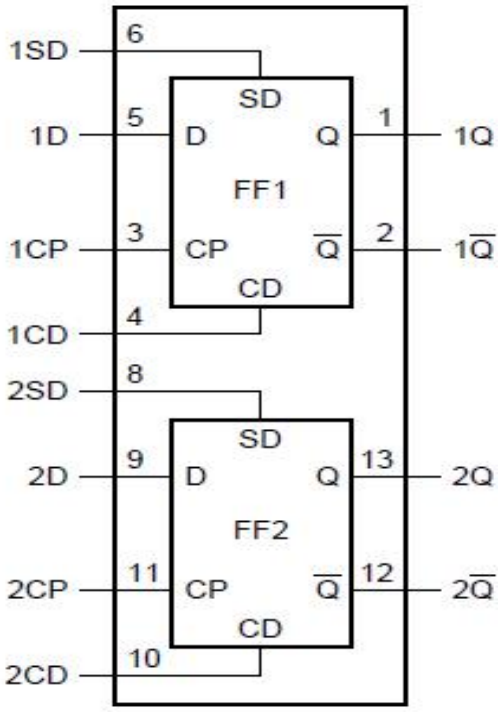
Conclusion:

- *It should be noted that Q output changes only when PGT occurs in positive edge data Flip-Flop shown in fig (I) and D input has no effect between PGTs i.e., NGTs.*
- *It should be noted that Q output changes only when NGT occurs in negative edge data Flip-Flop shown in fig (II) and D input has no effect between NGTs i.e., PGTs.*
- *From the simulated circuits it is clear that; In an active high SR Flip Flop when S (Set) and R (Reset) both are 0, there will be no change in the output of the Flip-Flop, and when both S and R, are 1 the output of the Flip-Flop is totally unpredictable. In an active low SR Flip Flop when S and R both are 1, there will be no change in the output of the Flip-Flop, and when both S and R, are 0 the output of the Flip-Flop is totally unpredictable. So, if both inputs of the flip-flop are the same there will either be a No Change or Invalid output condition. This ambiguity state is removed by the complement in D-flip flop.*

IC Package

The **IC used here is HEF4013BP (Dual D-type flip-flop).** It is a 14-pin package which contains 2 individual D flip-flop in it. Below are the pin diagram and the corresponding description of the pins.





Pin Description:

Q	**True Output**
$\overline{Q}$	**Compliment Output**
CP	**Clock Input**
CD	**CLEAR-Direct input**
D	**Data input**
SD	**PRESET-Direct input**
V_{SS}	**Ground**
V_{DD}	**Supply voltage**

Application:

1. *Event Detectors*
2. *Frequency Divider*
3. *Counters*
4. *Memory Devices*
5. *Shift Registers*

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.9
Gated JK Flip-Flop

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: To implement and verify the operation of gated JK flip flop and toggle mode application using MultiSim platform.

EQUIPMENT:

- Power supply unit (EV)
- Green and Red Probes

COMPONENTS FOR JK Flip-Flop:

- NAND Gate
- AND Gate
- NOR Gate

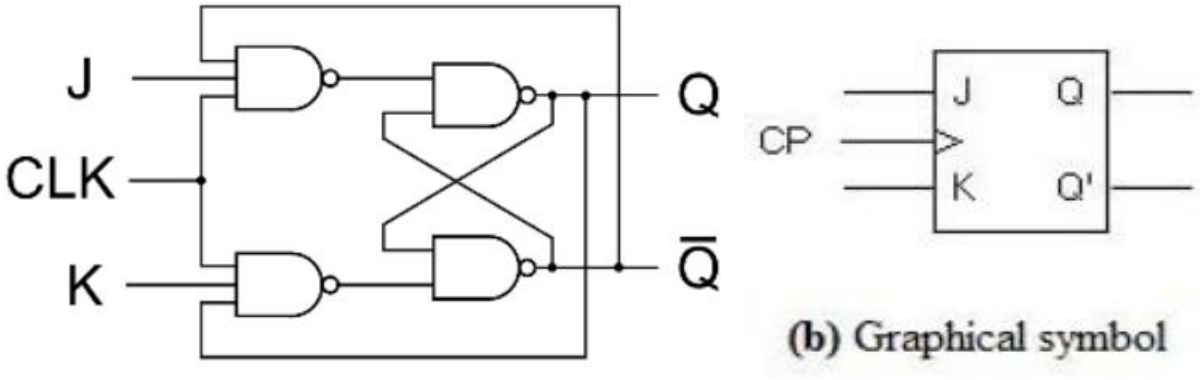
Gated JK Flip-Flop

Introduction:

JK flip flop using NAND Gates

A J-K flip-flop has very similar characteristics to an S-R flip-flop. The only difference is that the undefined condition for an S-R flip-flop, i.e., Sn = Rn = 1 condition, is also included in this case. Inputs J and K behave like inputs S and R to set and reset the flip-flop respectively. When J = K = 1, the flip-flop is said to be in a toggle state, which means the output switches to its complementary state every time a clock pass. The data inputs are J and K, which are NANDed with $\overline{Q}$ and Q respectively to obtain the inputs for S and

R respectively. A J-K flip-flop thus obtained is shown in Figure below.



Case 1: When the clock is applied and J = 0, whatever the value of $\overline{Q}$ (0 or 1), the output of NAND gate 1 is 1. Similarly, when K = 0, whatever the value of Q (0 or 1), the output of gate 2 is also 1. Therefore, when J = 0 and K = 0, the inputs to the basic flip-flop are S = 1 and R = 1. This condition forces the flip-flop to remain in the same state.

Case 2: When the clock is applied and J = 0 and K = 1 & the previous state of the flip-flop is reset (i.e., Q = 0 and $\overline{Q}$= 1), then S = 1 and R = 1. Since S = 1 and R = 1, the basic flip-flop does not alter the state and remains in the reset state. But if the flip-flop is in set condition (i.e., Q = 1 & $\overline{Q}$ = 0), then S = 1 and R = 0. Since S = 1 and R = 0, the basic flip-flop changes its state and resets.

Case 3: When the clock is applied and J = 1 and K = 0 and the previous state of the flip-flop is reset (i.e., Q = 0 and $\overline{Q}$ = 1), then S = 0 and R = 1. Since S = 0 and R = 1, the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (i.e., Q = 1 and $\overline{Q}$= 0), then S = 1 and R = 1. Since S = 1 and R = 1, the basic flip-flop does not alter its state and remains in the set state.

Case 4: When the clock is applied and J = 1 and K = 1 and the previous state of the flip-flop is reset (i.e., Q = 0 and $\overline{Q}$= 1), then S = 0 and R = 1. Since S = 0 and R = 1, the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (i.e., Q = 1 and $\overline{Q}$= 0), then S = 1 and R = 0. Since S = 1 and R = 0, the basic flip-flop changes its state and goes to the reset state. So we find that for J = 1 and K = 1, the flip-flop toggles its state from set to reset and vice versa. Toggle means to switch to the opposite state.

Truth Table:

Truth Table for PGT triggered JK F.F

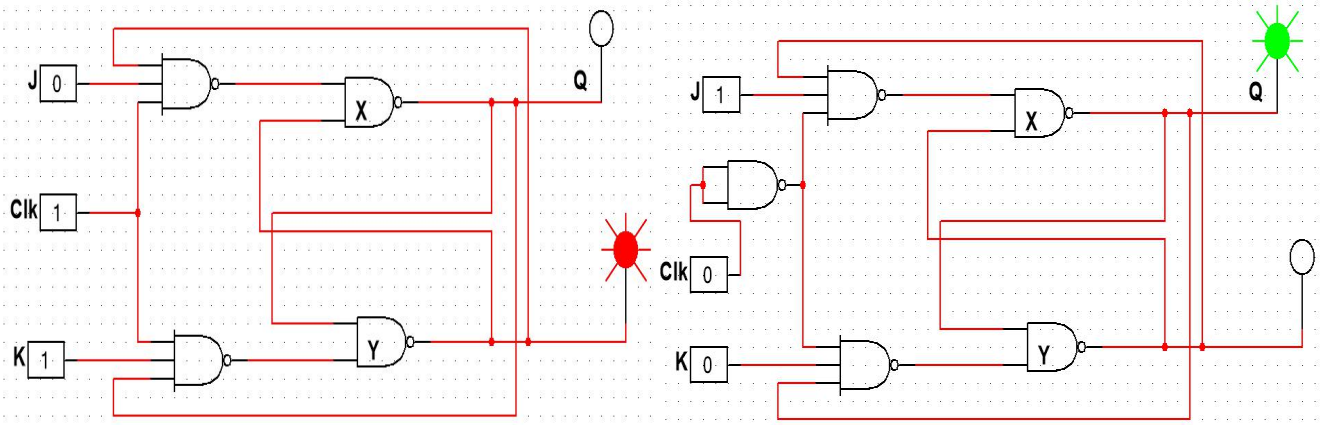
Input					Output		State
J	K	CLK	$\overline{J}$	$\overline{K}$	Q	$\overline{Q}$	-
1	0	↑	0	1	1	0	Set
0	0	↓	1	1	1	0	Hold
0	1	↓	1	0	1	0	Hold
0	0	↑	1	1	1	0	Memory
0	1	↑	1	0	0	1	Reset
0	0	↓	1	1	0	1	Hold
1	0	↓	0	1	0	1	Hold
0	0	↑	1	1	0	1	Memory
1	1	↑	0	0	1($\overline{Q}$)	0(Q)	Toggle

⇒ It is positive trigged JK flip flop. Negative triggered JK will operate same as above but will operate on negative going transition.

⇒ Below shown table proof it.

Input						Output		State
J	K	CLK	$\overline{CLK}$	$\overline{J}$	$\overline{K}$	Q	$\overline{Q}$	-
1	0	↓	↑	0	1	1	0	Set
0	0	↑	↓	1	1	1	0	Hold
0	1	↑	↓	1	0	1	0	Hold
0	0	↓	↑	1	1	1	0	Memory
0	1	↓	↑	1	0	0	1	Reset
0	0	↑	↓	1	1	0	1	Hold
1	0	↑	↓	0	1	0	1	Hold
0	0	↓	↑	1	1	0	1	Memory
1	1	↓	↑	0	0	$1(\overline{Q})$	0(Q)	Toggle

Simulated Circuit:



JK flip flop using NOR Gates

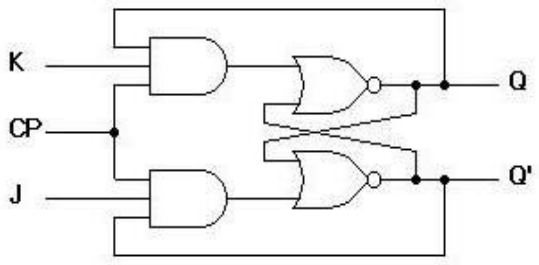
J-K flip flop can also be defined as a modification of the S-R flip flop. The only difference is that the intermediate state is more refined and precise than that of a S-R flip flop.

The behavior of inputs J and K is same as the S and R inputs of the S-R flip flop. The letter J stands for SET and the letter K stands for CLEAR.

When both the inputs J and K have a HIGH state, the flip-flop switch to the complement state. So, for a value of Q = 1, it switches to Q=0 and for a value of Q = 0, it switches to Q=1.

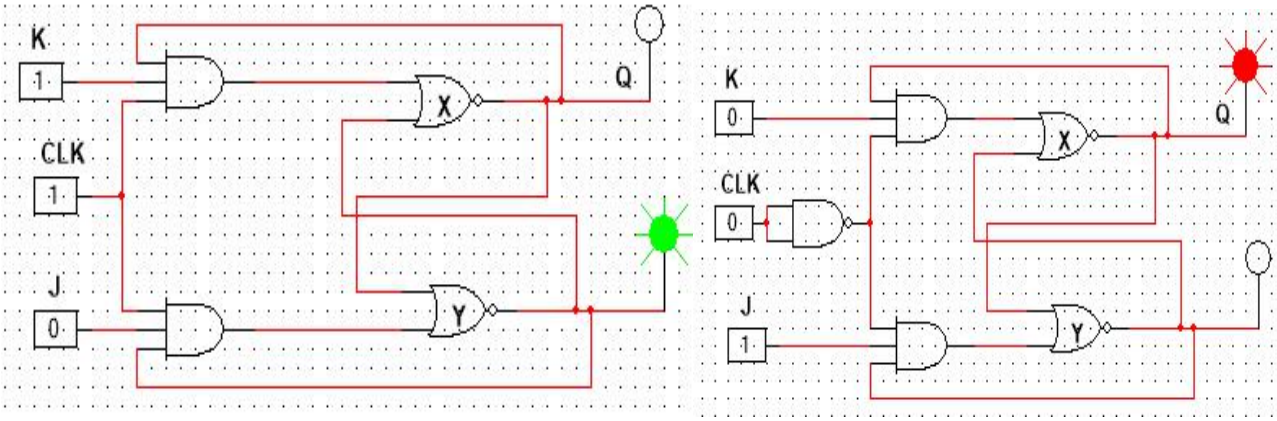
The circuit includes two 3-input AND gates. The output Q of the flip flop is returned as a feedback to the input of the AND along with other inputs like K and clock pulse [CP]. So, if the value of CP is '1', the flip flop gets a CLEAR signal and with the condition that the value of Q was earlier 1. Similarly output Q' of the flip flop is given as a feedback to the input of the AND along with other inputs like J and clock pulse [CP]. So, the output becomes SET when the value of CP is 1 only if the value of Q' was earlier 1.

The output may be repeated in transitions once they have been complimented for J=K=1 because of the feedback connection in the JK flip-flop. This can be avoided by setting a time duration lesser than the propagation delay through the flip-flop.



(a) Logic diagram

Simulation Results:



TASK: IMPLEMENT JK TO CONSTRUCT "T" FF

Introduction:

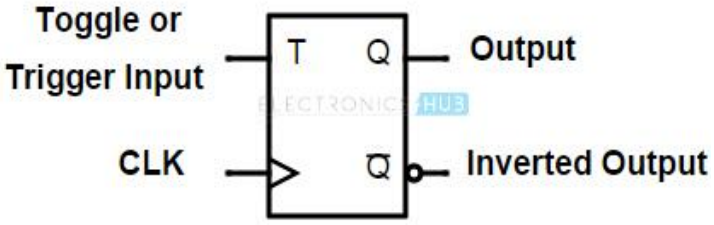
T flip – flop is also known as "Toggle Flip – flop". To avoid the occurrence of intermediate state (also known as the forbidden state) in SR flip – flop, we should provide only one input to the flip – flop called the Trigger input or Toggle input (T).

Then the flip – flop acts as a Toggle switch. Toggling means 'Changing the next state output to complement of the present state output'.

We can design the T flip – flop by making simple modifications to the JK flip – flop. The T flip – flop is a single input device and hence by connecting J and K inputs together and giving them with single input called T, we can convert a JK flip – flop into T flip – flop.

So, a T flip – flop is sometimes called as single input JK flip – flop.

The logic symbol of T flip – flop is shown below. It has one Toggle input (T) & one clock signal input (CLK).



T Flip – Flop Circuit

We can construct a T flip – flop by any of the following methods:

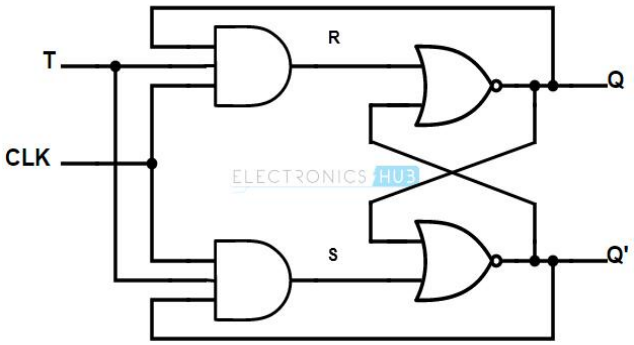
- Using NOR gates.
- Connecting an XOR with T input and Q PREVIOUS output to the Data input in D flip – flop.
- Hard – wiring the J and K inputs together and connecting it to T input in JK flip – flop.

Using NOR Flip-Flop

We can construct a T flip – flop by connecting AND gates as input to the NOR gate SR latch. And one of the inputs to these AND gates are the feedback from the present state output Q and its complement Q to the respective AND gates i.e., Q to the AND Gate associated with R input and Q to the AND Gate associated with S input.

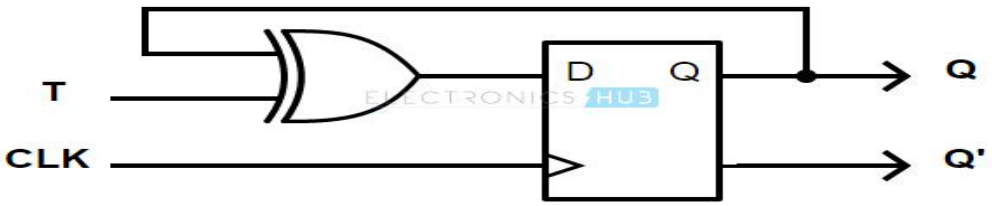
A toggle input (T) is connected in common to both the AND gates as an input. The AND gates are also connected with common Clock (CLK) signal.

In the T flip – flop, a pulse train of narrow triggers are provided as input (T) which will cause the change in output state of flip – flop. So, these flip – flops are also called Toggle flip – flops. The circuit diagram of a T flip – flop constructed from SR latch is shown below.



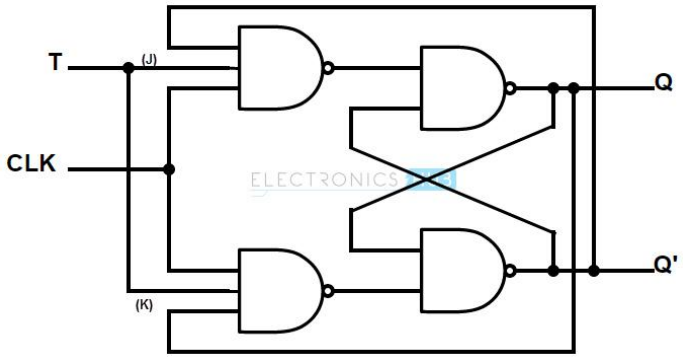
Using D Flip-Flop

Similarly, a T flip – flop can be constructed by modifying D flip – flop. In D flip – flop, the output Q is XORed with the T input and given at the D input. The circuit of a T flip – flop constructed from a D flip – flop is shown below.



Using JK Flip-Flop

The simplest of the constructions of a T flip – flop is with JK flip – flop. The J input and K input of the JK flip – flop is connected together and provided with the T input. The logic circuit of a T flip – flop constructed from a JK flip – flop is shown below.



Working

T flip – flop is an edge triggered device i.e., the low to high or high to low transitions on the input clock signal will cause a change in the output state of the flip – flop.

As mentioned earlier, T flip – flop is an edge triggered device. For example, consider a T flip – flop made of NAND JK latch as shown.

If the output Q = 0, then the upper NAND is in disabled state (doesn't have any dominating inputs) and lower NAND gate is in enable condition (feedback from Q is the dominating input). This means, the the Toggle Input will set the input condition for the JK Flip-Flop as K = 1 and J = 0. If you recall the truth table of RS Flip-Flop, this condition will SET the output. Hence, Q becomes 1.

If the output Q = 1, then the upper NAND is in enable state and lower NAND gate is in disable condition. This allows the Toggle Input to set the inputs for the JK Flip-Flop as K = 0 and J = 1. This will make the flip – flop to RESET i.e., Q = 0.

In simple terms, the operation of the T flip – flop is

When the T input is LOW, then the next sate of the T flip flop is same as the present state.

- T = 0 and present state = 0, then the next state = 0
- T = 0 and present state = 1, then the next state = 1

When the T input is HIGH and during the positive transition of the clock signal, the next state of the T flip – flop is the complement of the present state.

- T = 1 and present state = 0, then the next state = 1
- T = 1 and present state = 1, then the next state = 0

As each incoming trigger alternately changes the SET and RESET inputs, the flip – flop toggles. So, to complete one full cycle of output waveform, it needs two triggers. This means that the T flip flop produces the output at exactly half of the frequency of input frequency. So, a T flip – flops will act as "Frequency Divider Circuit".

- ➢ The main disadvantage of T flip – flop is that the state of the flip – flop at an applied trigger pulse is known only when the previous state is known.

⇒ Generally, T flip flops are not available as ICs. So, they can be constructed by using JK flip – flop, SR flip – flop and D flip – flop.

Truth Table of T flip – flop

Clock	Input		Output	
	Reset	T	Q	$\overline{Q}$
X	Low	X	0	1
High	High	0	No Change	
High	High	1	Toggle	
Low	High	X	No Change	

Applications

Let us now see a couple of important applications of T flip-flop.

- Frequency Division Circuit
- 2 – Bit Parallel Load Registers

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.10
ASYNCHRONOUS COUNTER DESIGN AND MOD-N COUNTER

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: - Realization of 3-bit Asynchronous counter and Mod-N counter design .

Apparatus Required: -

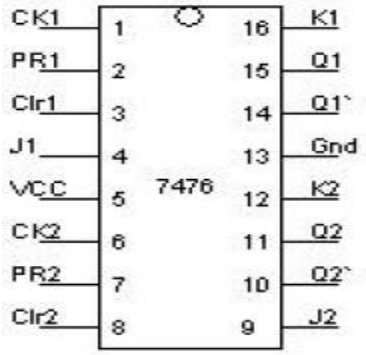
IC 7408, IC 7476, IC 7400, IC 7432 etc.

Procedure: -

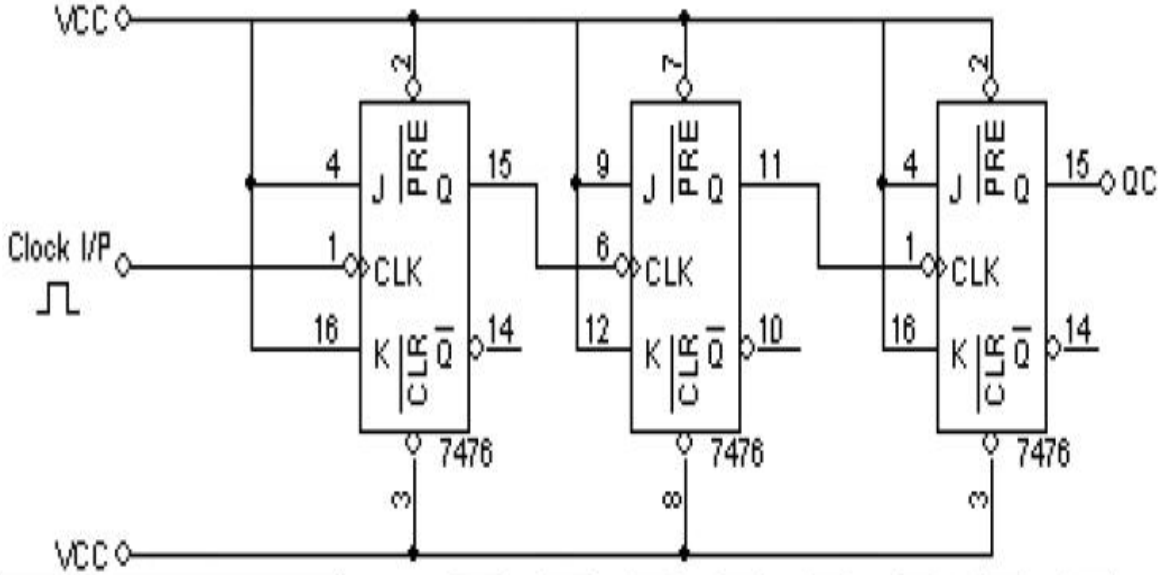
1. Connections are made as per circuit diagram.
2. Clock pulses are applied one by one at the clock I/P and the O/P is observed at QA, QB & QC for IC 7476.
3. Verify the Truth table .

Circuit Diagram:

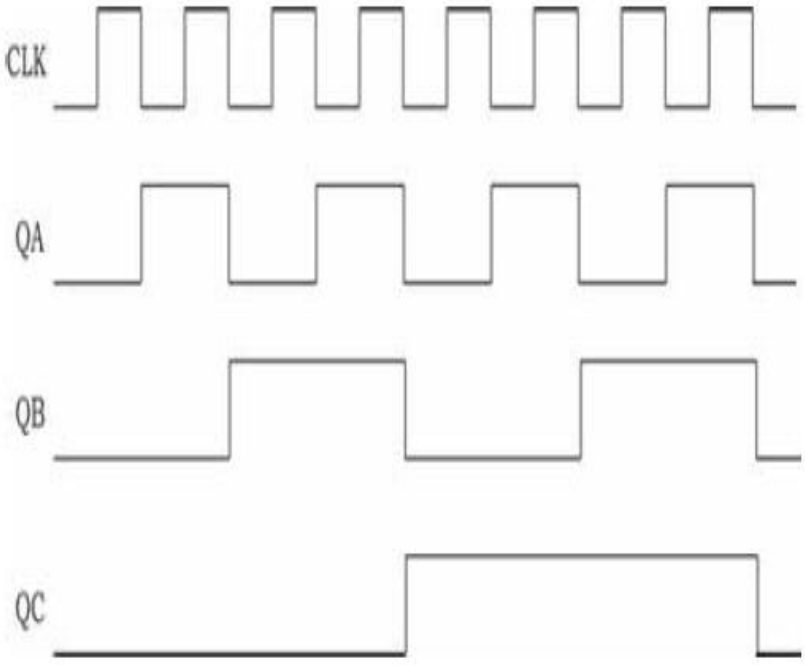
Pin Details: -



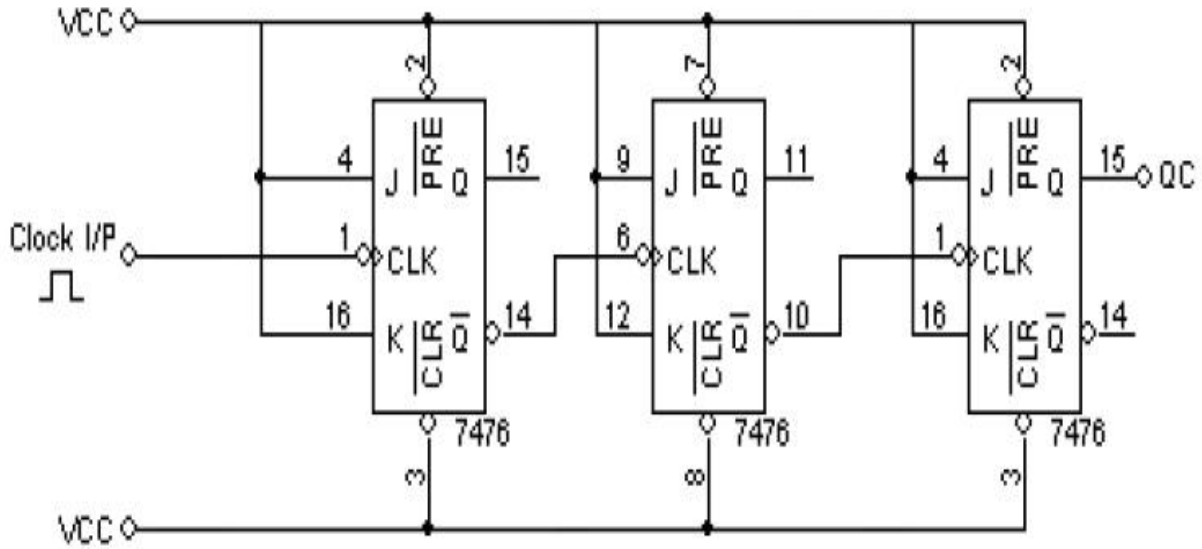
Circuit Diagram: - 3-Bit Asynchronous Up Counter



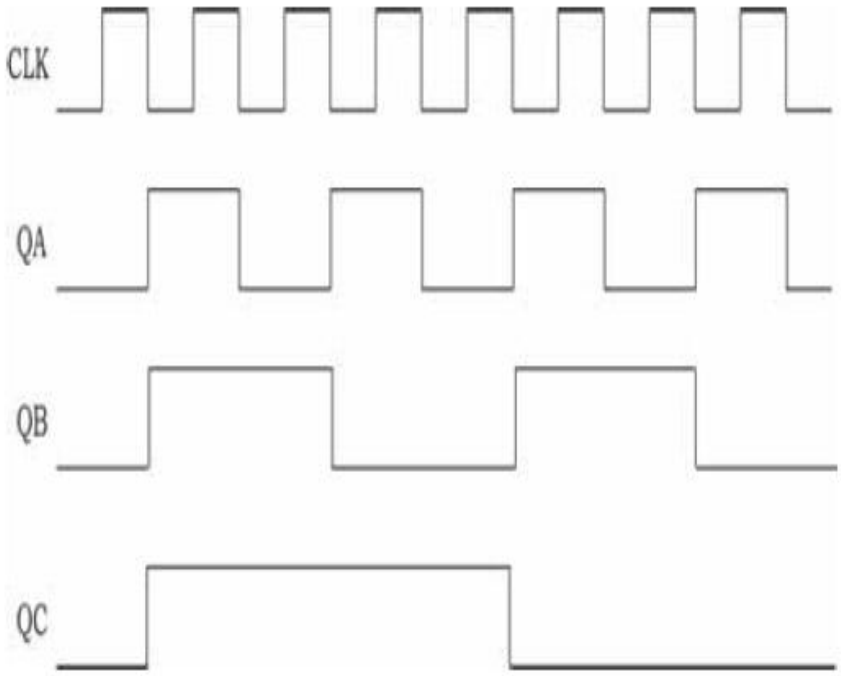
3-bit Asynchronous up counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



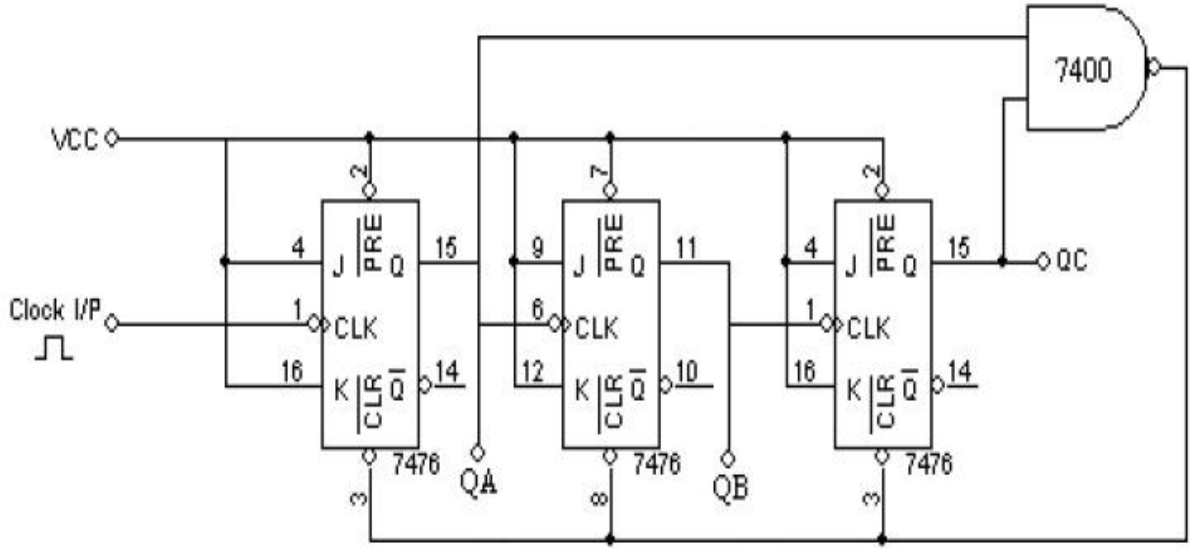
Circuit Diagram: - 3-Bit Asynchronous Down Counter



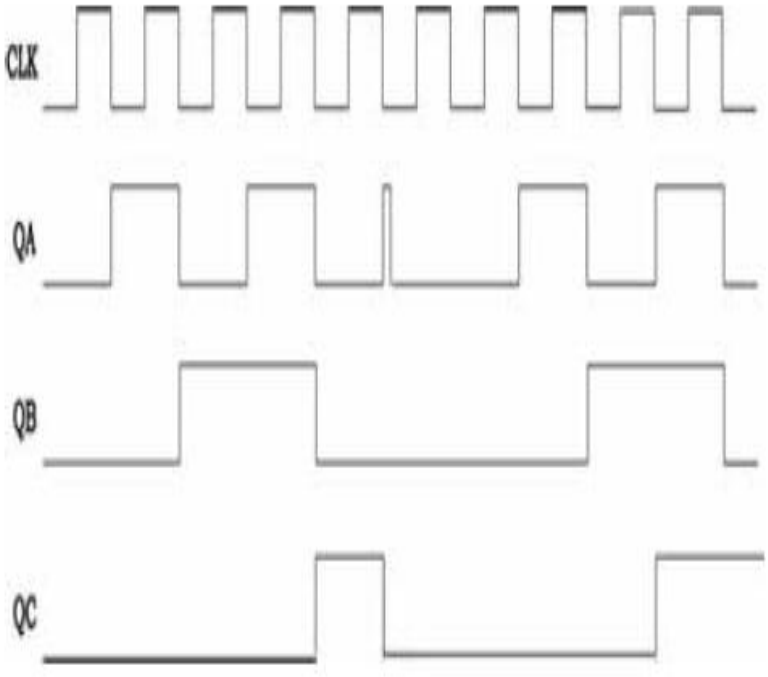
3-bit Asynchronous down counter			
Clock	QC	QB	QA
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0
8	1	1	1
9	1	1	0



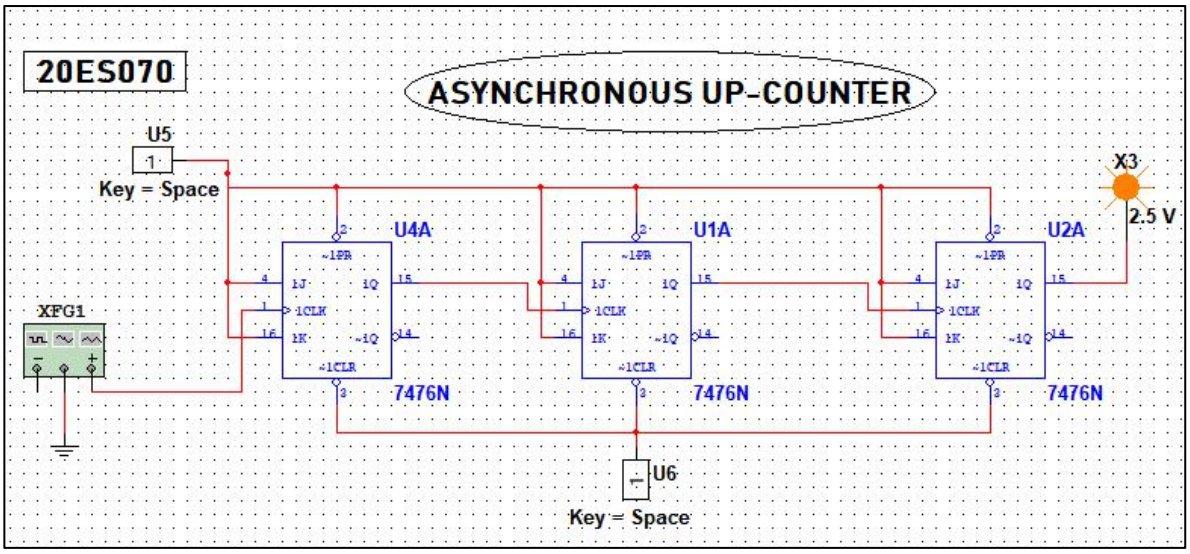
Mod 5 Asynchronous Counter:-

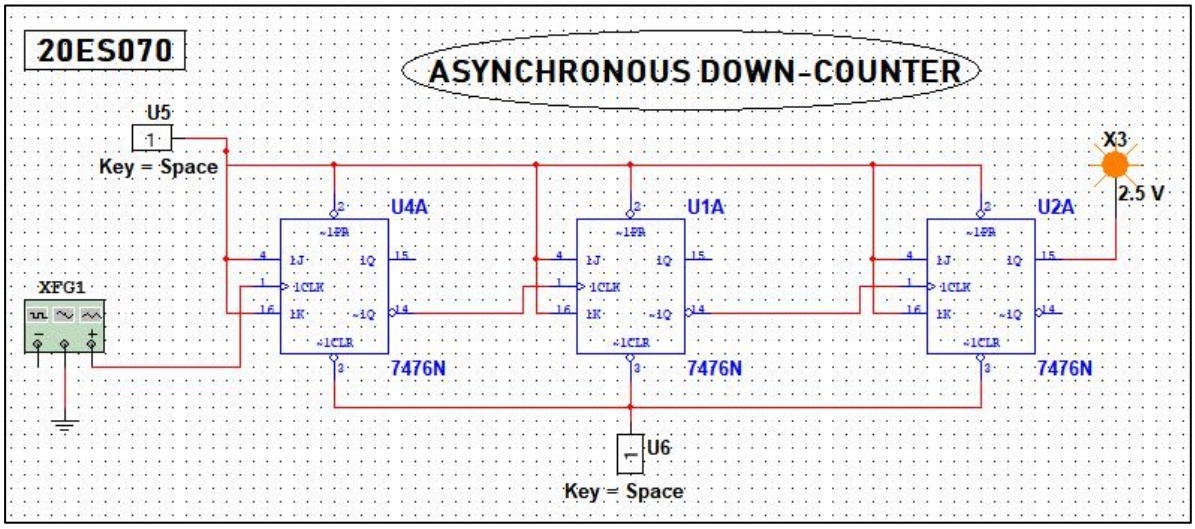


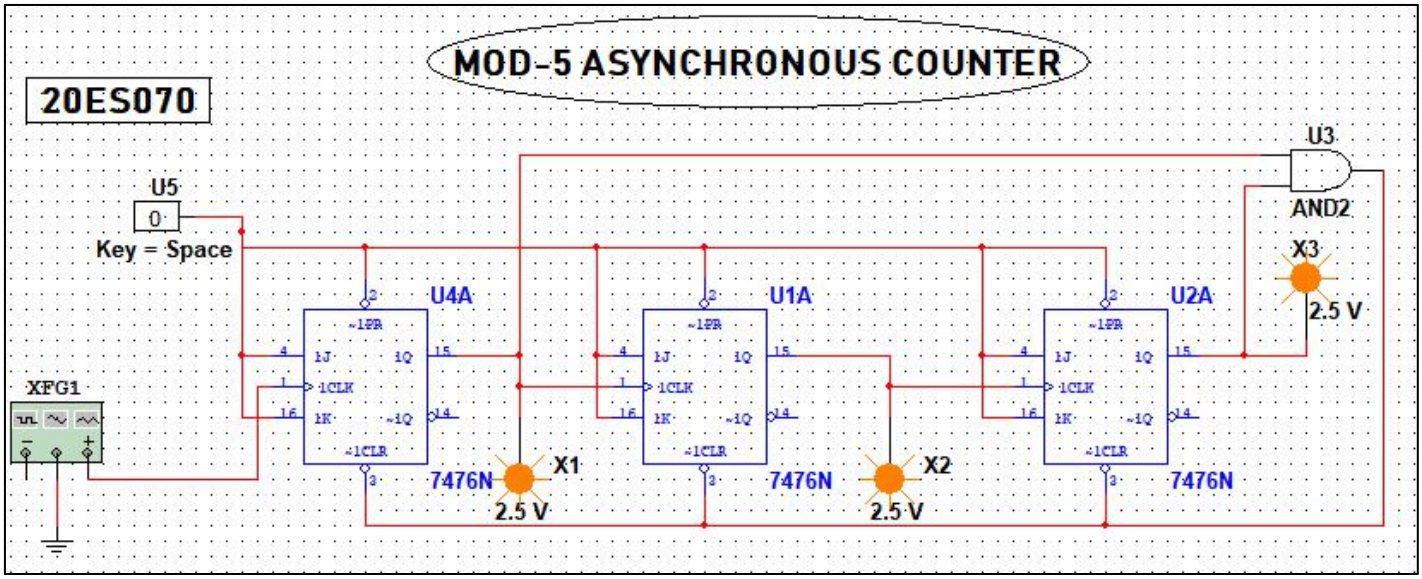
Mod 5 Asynchronous counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0



Task: perform simulation in multisim and show the results for this objective







Conclusion:-

Asynchronous counters are those whose output is free from the clock signal. Because the flip flops in asynchronous counters are supplied with different clock signals, there may be delay in producing output.

The required number of logic gates to design asynchronous counters is very less. So they are simple in design. Another name for Asynchronous counters is “Ripple counters”.

The number of flip flops used in a ripple counter is depends up on the number of states of counter (ex: Mod 4, Mod 2 etc). The number of output states of counter is called “Modulus” or “MOD” of the counter. The maximum number of states that a counter can have is 2n where n represents the number of flip flops used in counter.

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.11
SYNCHRONOUS COUNTER DESIGN

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: - Realization of 3-bit synchronous counter design.

Apparatus Required: -

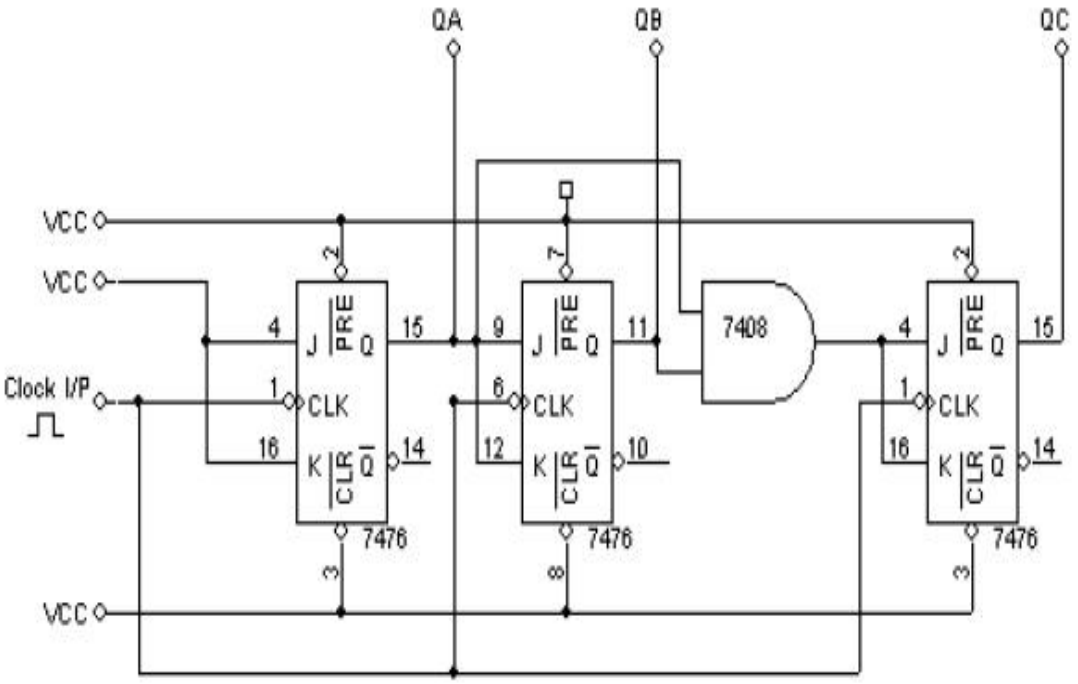
IC 7408, IC 7476, IC 7400, IC 7432 etc.

Procedure: -

1. Connections are made as per circuit diagram.
2. Clock pulses are applied one by one at the clock I/P and the O/P is observed at QA, QB & QC for IC 7476.
3. Verify the Truth table .

Circuit Diagram& Truth table:

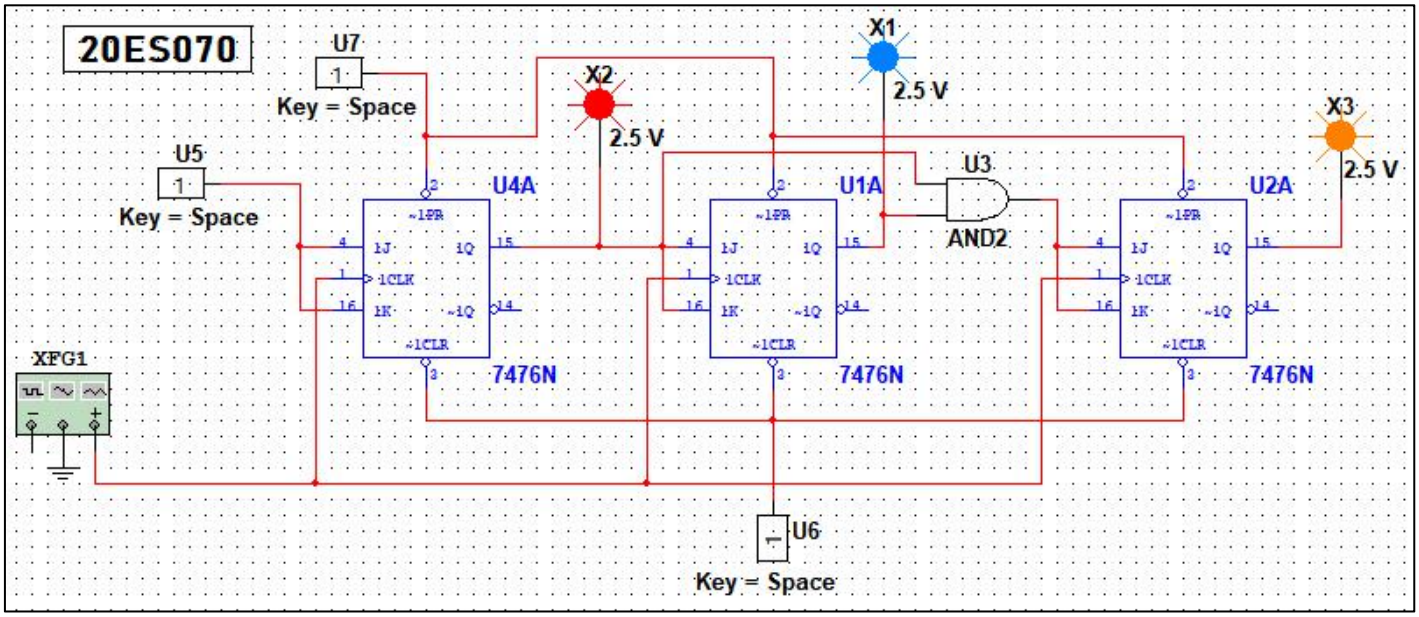
3-bit Synchronous Counter:-



Truth table :-

Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Task : Simulate the counter circuit in multisim and display the results.



Conclusion:-

The counters which use clock signal to change their transition are called “Synchronous counters”. This means the synchronous counters depends on their clock input to change state values. In synchronous counters, all flip flops are connected to the same clock signal and all flip flops will trigger at the same time.

Synchronous counters are also known as ‘Simultaneous counters. There is no propagation delay and no ripple effect in synchronous counters.

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.12
COMPARATORS

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: To verify the truth table of one bit and four bit comparators using logic Gates and IC 7485

Apparatus Required: -

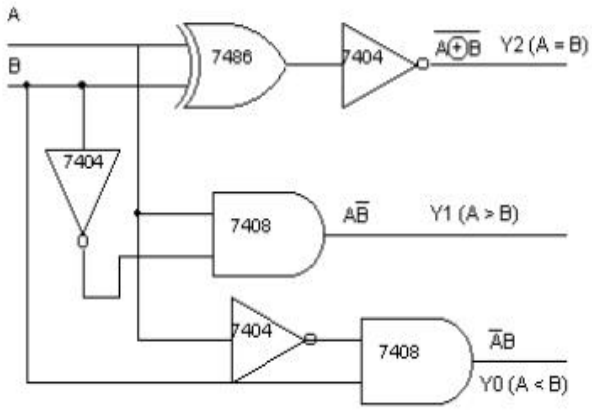
IC 7486, IC 7404, IC 7408, IC 7485etc.

Procedure: -

1. Verify the gates.
2. Make the connections as per the circuit diagram.
3. Switch on Vcc.
4. Applying i/p and Check for the outputs.
5. The readings of outputs should be tabulated .

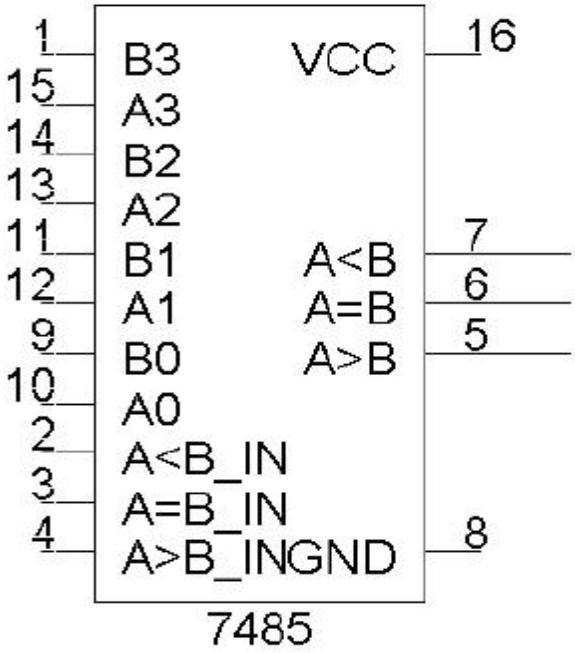
Circuit diagram& truth tables:-

One Bit Comparator: -



A	B	Y1 (A>B)	Y2 (A = B)	Y3 (A < B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

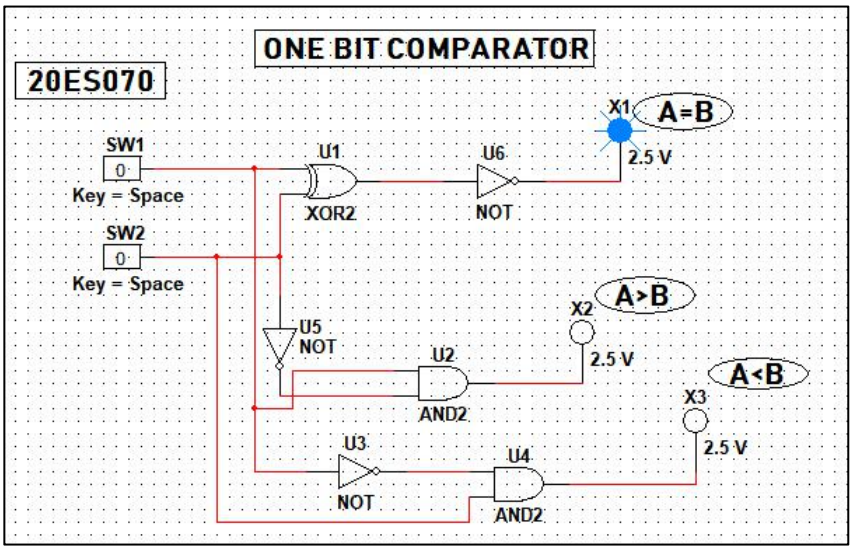
4-bit Comparator

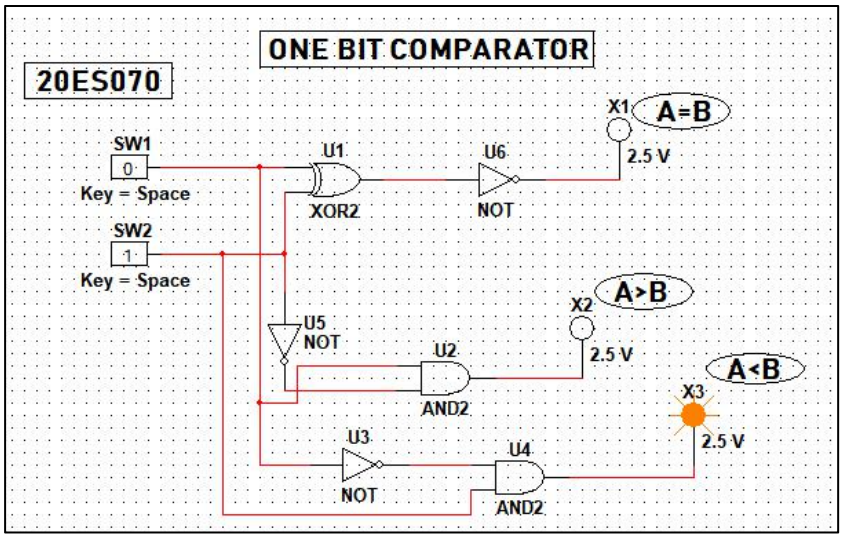


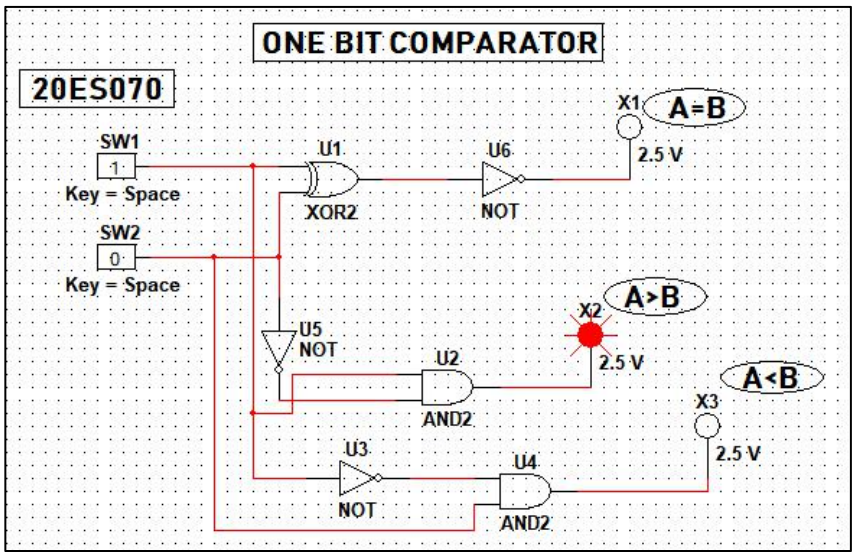
Tabular column :-

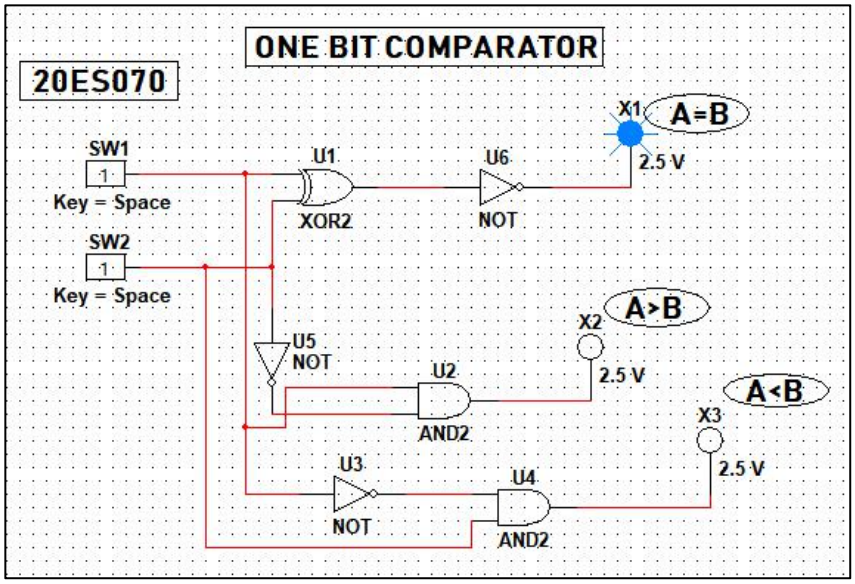
A_3 B_3	A_2 B_2	A_1 B_1	A_0 B_0	A>B	A=B	A<B	A>B	A=B	A<B
$A_3>B_3$	X	X	X	X	X	X			
$A_3<B_3$	X	X	X	X	X	X			
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X			
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X			
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X			
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X			
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X			
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X			

Task: simulate the comparator circuit, and show results for each action.









Conclusion: -

Comprator just compares the input binary bits and match the LSB then Display. It displays either (A=B) or (A>B) or (A<B). Furthermore,The comparator determines whether one number is greater than, less than or, equal to the other number. Comparators are used in Central processing Units CPUs and Microcontrollers.

Mehran University of Engineering & Technology, Jamshoro

Department of Electronic Engineering

DIGITAL ELECTRONICS

Lab NO.13

ENCODER AND DECODER

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: Realization of 2:4 / 3:8 decoder and 4:2/8:3 encoder

Pre-Laboratory :-

There are two tasks that you must perform **prior** to sitting this laboratory:

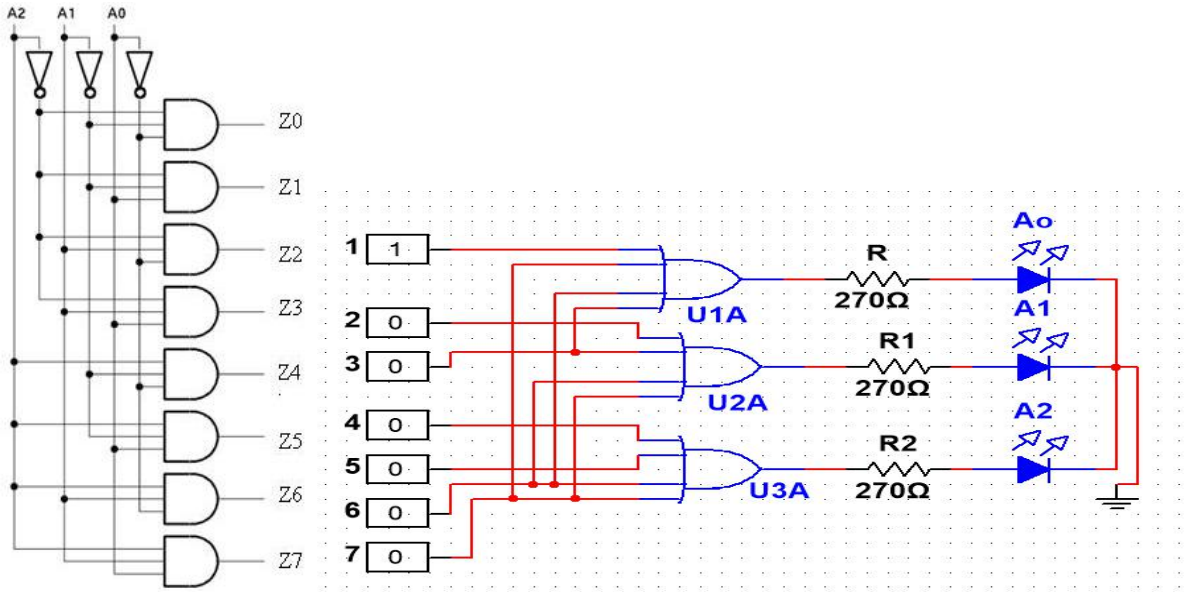
1. Understand the design problem given to you.
2. Draw up the truth tables and logic diagram for the design.

Apparatus Required: - E-18 kit / Multisim

Procedure: -

1. Verify the gates.
2. Make the connections as per the circuit diagram.
3. Switch on VCC and apply various combinations of input according to truth table.
4. Note down the output readings combinations of inputs.

Circuit Diagram& Truth table :-



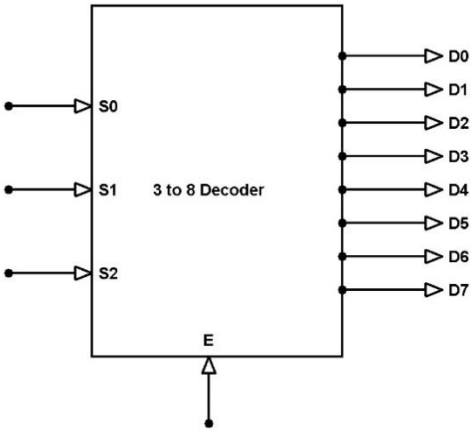
4:2 lines Decoder 8:3 lines encoder

Decoder:

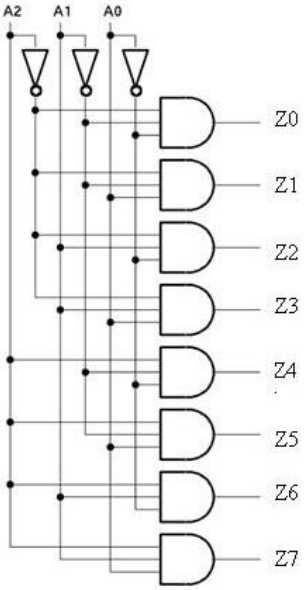
Decoder is a combinational circuit that has 'n' input lines and maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. It is the reverse process of an encoder.

3 to 8 Decoder

This decoder circuit gives 8 logic outputs for 3 inputs and has a enable pin. The circuit is designed with AND and NAND logic gates. It takes 3 binary inputs and activates one of the eight outputs. 3 to 8 line decoder circuit is also called a binary to an octal decoder.



The decoder circuit works only when the Enable pin (E) is high. S0, S1 and S2 are three different inputs and D0, D1, D2, D3. D4. D5. D6. D7 are the eight outputs. The logic diagram of the 3 to 8 line decoder is shown below.



3 to 8 Line Decoder and Truth Table

The below table gives the truth table of 3 to 8 line decoder.

S0	S1	S2	E	D0	D1	D2	D3	D4	D5	D6	D7
x	x	x	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

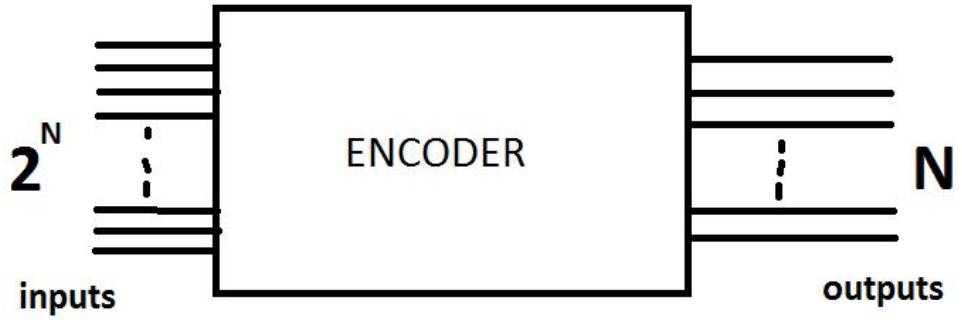
When the Enable pin (E) is low all the output pins are low.

Application of Decoder

- The Decoders were used in analog to digital conversion in analog decoders.
- Used in electronic circuits to convert instructions into CPU control signals.
- They mainly used in logical circuits, data transfer.

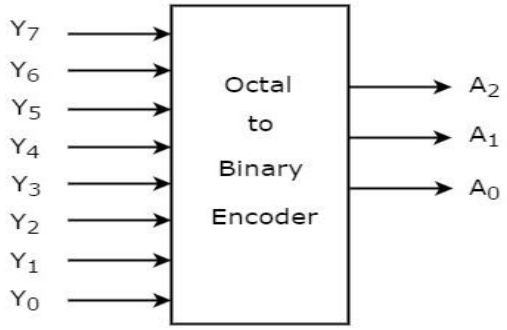
Encoder

An Encoder is a combinational circuit that performs the reverse operation of decoder. It has maximum of 2^N input lines and 'n' output lines, hence it encodes the information from 2^N inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^N input lines with 'n' bits.



Octal to Binary Encoder

The 8 to 3 Encoder or octal to Binary encoder consists of 8 inputs *Y*0 to *Y*7 and 3 outputs : *A*0, *A*1 & *A*2. Each input line corresponds to each octal digit and three outputs generate corresponding binary code: The figure below shows the logic symbol of octal to binary encoder:



The truth table for 8 to 3 encoder active high is as follows:

Deci.No	INPUTS								OUTPUTS		
	*Y*0	*Y*1	*Y*2	*Y*3	*Y*4	*Y*5	*Y*6	*Y*7	*A*2	*A*1	*1A*0
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1

From Truth table, we can write the Boolean functions for each output as:

- Bit *A*2 is always a 1 for decimal digit 4, 5, 6 or 7 and can be expressed as an OR function as follows:

$A2 = Y7+Y6+Y5+Y4$

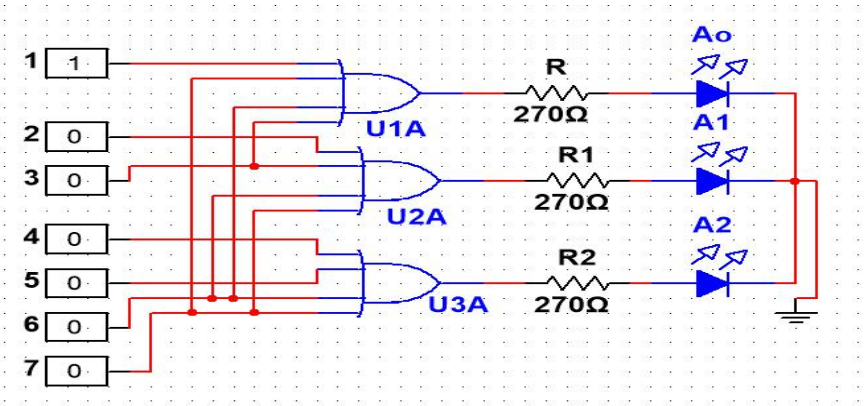
- Bit *A*1 is always a 1 for decimal digit 2, 3, 6, or 7 and can be expressed as

$A1 = Y7+Y6+Y3+Y2$

- Finally, *A*0 is always 1 for decimal digit 1, 3, 5, 7. The expression for *A*0 is

$A0 = Y7+Y5+Y3+Y1$

We can implement the above Boolean functions by using four input OR gates. The circuit diagram of octal to binary encoder is shown in the following figure.



The truth table for 8 to 3 encoder active Low is as follows:

Dec..No	INPUTS								OUTPUTS(low active)		
	*Y*0	*Y*1	*Y*2	*Y*3	*Y*4	*Y*5	*Y*6	*Y*7	*A*2	*A*1	*A*0
0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	0
2	1	1	0	1	1	1	1	1	1	0	1
3	1	1	1	0	1	1	1	1	1	0	0
4	1	1	1	1	0	1	1	1	0	1	1
5	1	1	1	1	1	0	1	1	0	1	0
6	1	1	1	1	1	1	0	1	0	0	1
7	1	1	1	1	1	1	1	0	0	0	0

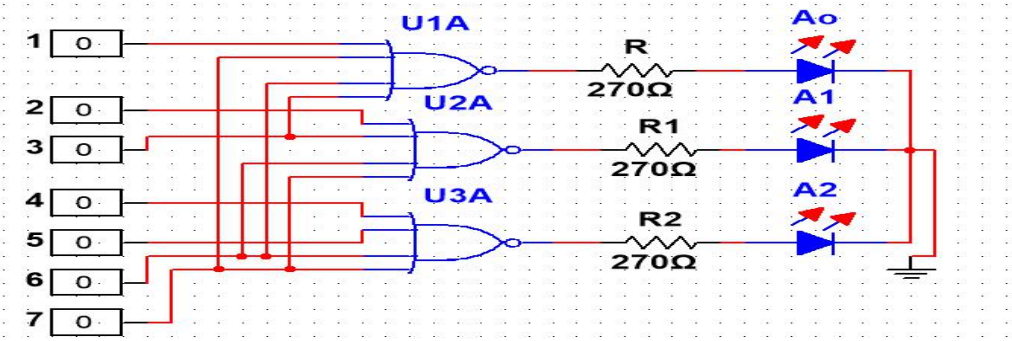
From Truth table, we can write the Boolean functions for each output as

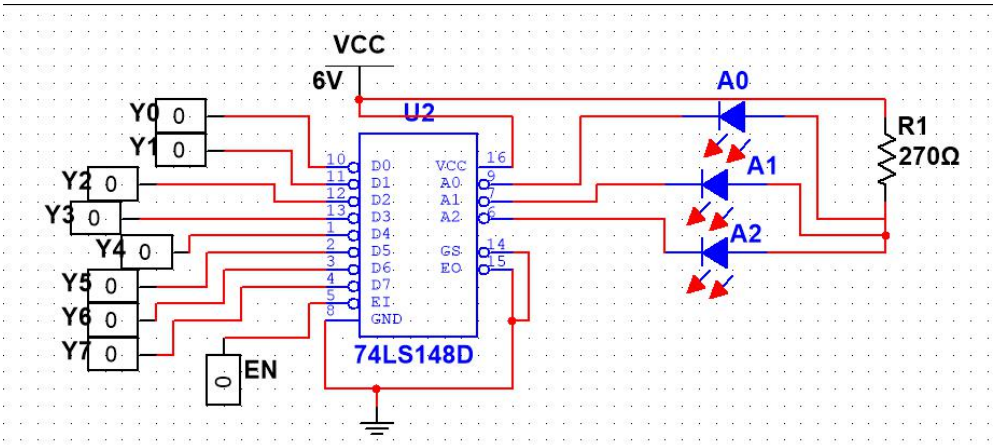
$\overline{A0} = \overline{Y7} + \overline{Y5} + \overline{Y3} + \overline{Y1}$

$\overline{A1} = \overline{Y7} + \overline{Y6} + \overline{Y3} + \overline{Y2}$

$\overline{A2} = \overline{Y7} + \overline{Y6} + \overline{Y5} + \overline{Y4}$

We can implement the above Boolean functions by using four input NOR gates. The circuit diagram of octal to binary encoder is shown in the following figure.





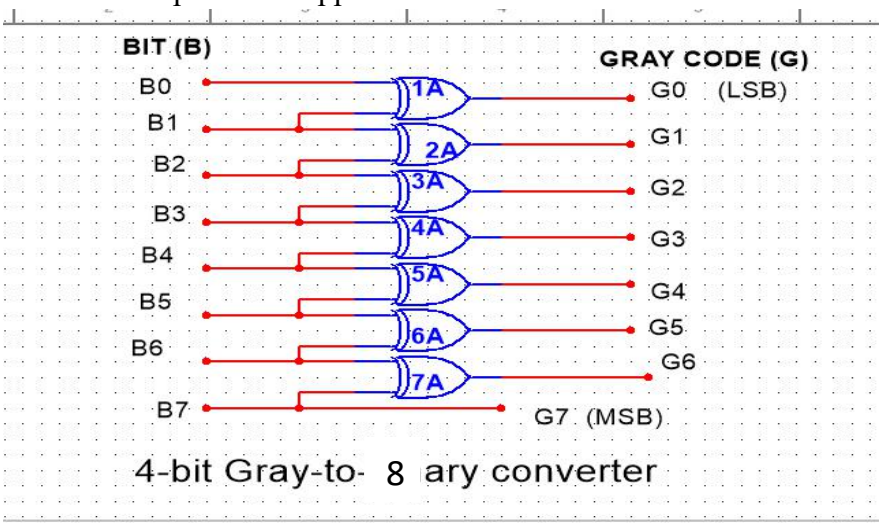
Drawbacks of Normal Encoders

1. There is an ambiguity, when all outputs of encoder are equal to zero.
2. If more than one input is active High, then the encoder produces an output, which may not be the correct code.

So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the (code corresponding to the active High inputs, which has higher priority.

Uses of Encoder

1. Encoders are very common electronic circuits used in all digital systems.
2. Encoders are used to translate the decimal values to the binary in order to perform the binary functions such as addition, subtraction, multiplication, etc.
3. Other applications especially for Priority Encoders may include detecting interrupts in microprocessor applications.



Conclusion: -

An Encoder is a combinational circuit that performs the reverse operation of decoder. It has maximum of 2^N input lines and 'n' output lines, hence it encodes the information from 2^N inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^N input lines with 'n' bits.

Mehran University of Engineering & Technology, Jamshoro

Department of Electronic Engineering

DIGITAL ELECTRONICS

Lab NO.14

MUX/DEMUX

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: To Design &verify the truth table of MUX and DEMUX .

Apparatus Required: -

IC 74153, IC 74139 etc.

Procedure: - (IC 74153)

1. The Pin [16] is connected to + Vcc.
2. Pin [8] is connected to ground.
3. The inputs are applied either to 'A' input or 'B' input.
4. If MUX 'A' has to be initialized, Ea is made low and if MUX 'B' has to be initialized, Eb is made low.
5. Based on the selection lines one of the inputs will be selected at the output and thus verify the truth table

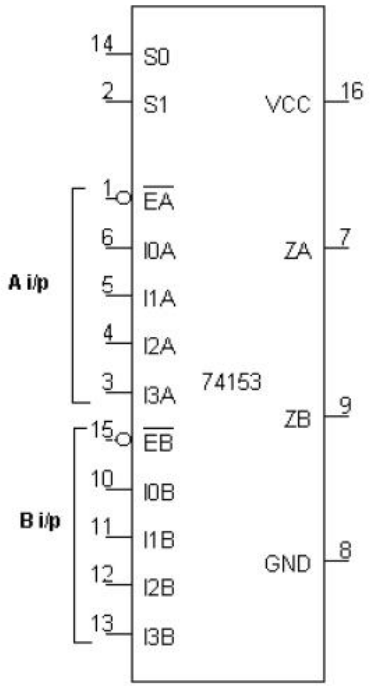
Procedure: - (IC 74139)

1. The inputs are applied to either ‘a’ input or ‘b’ input

2. The demux is activated by making Ea low and Eb low.

3. Verify the truth table .

Pin Details: -



Truth Table: -

CHANNEL – A							
INPUTS					SELECT LINES		O/P
Ēa	Ioa	I1a	I2a	I3a	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

CHANNEL – B							
INPUTS					SELECT LINES		O/P
Ēa	Iob	I1b	I2b	I3b	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

Pin Details: -

Truth Table For Demux: -

CHANNEL – A						
Inputs			Outputs			
Ēa	S1a	S0a	Y0a	Y1a	Y2a	Y3a
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

CHANNEL – B						
Inputs			Outputs			
Ēb	S1b	S0b	Y0b	Y1b	Y2b	Y3b
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

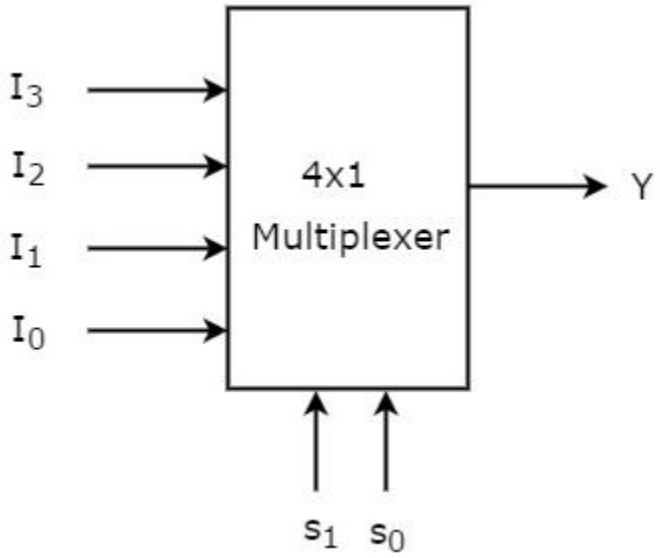
MULTIPLEXERS (DATA SELECTORS)

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as Mux.

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y. The block diagram of 4x1 Multiplexer is shown in the following figure.



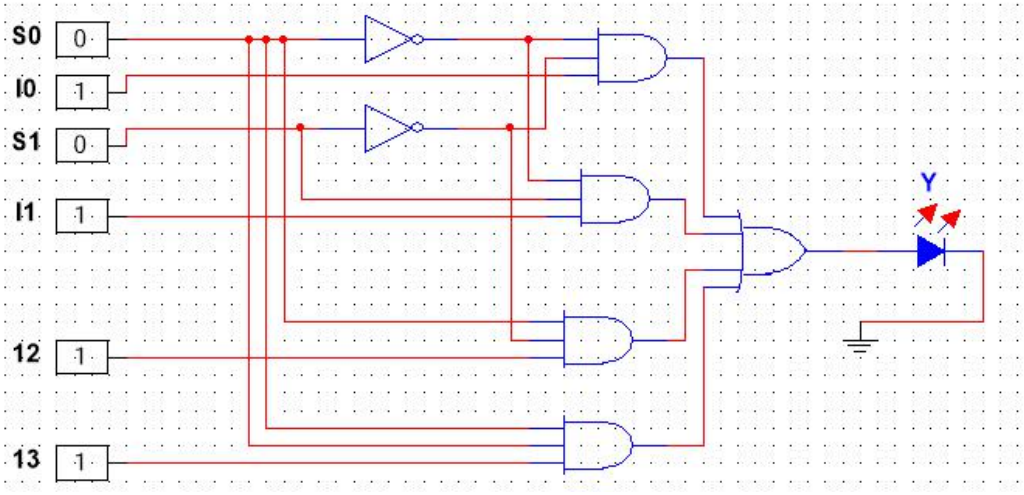
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. Truth table of 4x1 Multiplexer is shown below.

SELECTION LINES		OUTPUT
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the Boolean function for output, Y as

$$Y = \overline{S_0}\overline{S_1}I_0 + \overline{S_0}S_1 I_1 + S_0\overline{S_1}I_2 + S_0S_1I_3$$

We can implement this Boolean function using Inverters, AND gates & OR gate. The circuit diagram of 4x1 multiplexer is shown in the following figure.

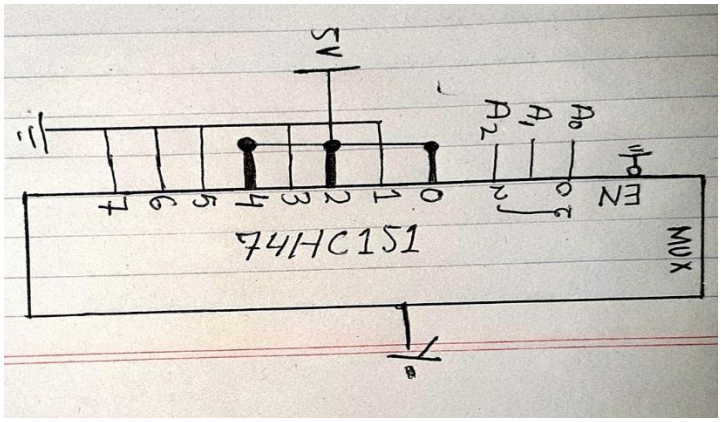


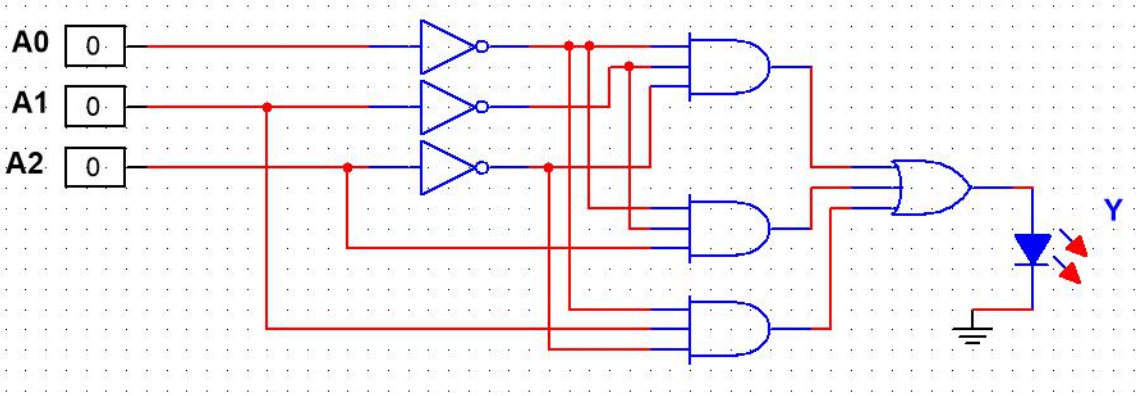
Ques. Implement the logic function specified in Table 6–9 by using a 74HC151 8-input data selector/multiplexer. Compare this method with a discreet logic gate implementation.

INPUT			OUTPUT
A_2	A_1	A_0	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

SOLUTION:

Notice from the truth table that Y is a 1 for the following input variable combinations: 000, 010 and 100. For all other combinations, Y is 0. For this function to be implemented with the data selector, the data input selected by each of the above-mentioned combinations must be connected to a HIGH (5 V). All the other data inputs must be connected to a LOW (ground), as shown in Figure 6–50.





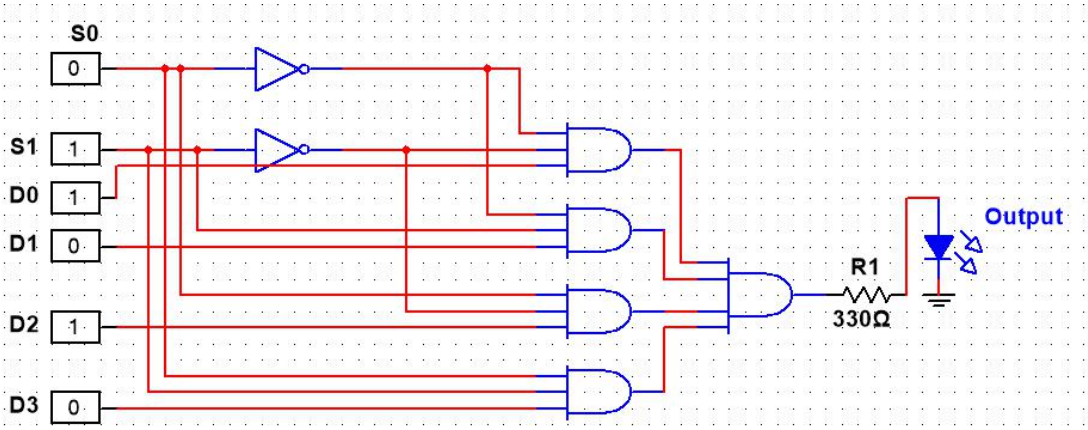
In Figure, D0 = 1, D1 = 0, D2 = 1, D3 = 0, S0 = 1, and S1 = 0. What is the output?

According to the truth table

SELECTION LINES		OUTPUT	Boolean Expression
S_0	S_1	Y	
0	0	D_0	$\overline{S_0}\overline{S_1}D_0$
0	1	D_1	$\overline{S_0}S_1\ D_1$
1	0	D_2	$S_0\overline{S_1}D_2$

1	1	D_3	$S_0S_1D_3$

The output is 0 because on given selection line, D_2 is low.



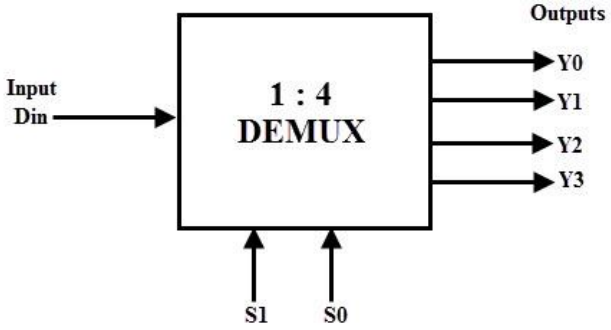
Demultiplexer

A demultiplexer is a device that takes a single input and gives one of the several output lines. A demultiplexer takes one single input data and then selects any one of the single output lines one at a time. It is the reverse process of a multiplexer. It is also called as a DEMUX or a data distributor. A DEMUX converts the input serial data line into output parallel data. A DEMUX gives '2n' outputs for 'n' selection lines with a single input.

1-to-4 demultiplexer

A 1-to-4 demultiplexer has a single input (D), two selection lines (S1 and S0) and four outputs (Y0 to Y3). The input data goes to any one of the four outputs at a given time for a particular combination of select lines.

This demultiplexer is also called as a 2-to-4 demultiplexer which means that two select lines and 4 output lines. The block diagram of 1:4 DEMUX is shown below.



Truth Table

DATA INPUT	SELECTON LINE		OUTPUT			
	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

From the table, the output logic can be expressed as min terms and are given below.

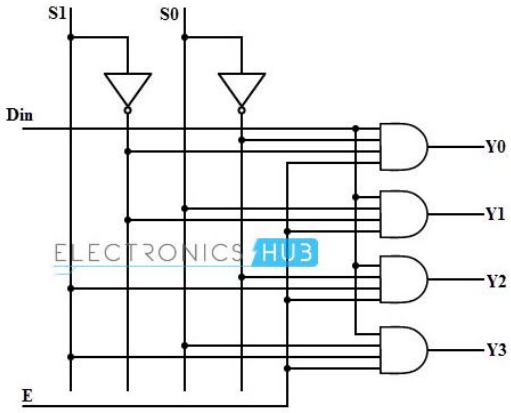
$$Y0 = \overline{S1}\ \overline{S0}\ D$$

$$Y1 = \overline{S1}\ S0\ D$$

$$Y2 = S1\ \overline{S0}\ D$$

$$Y3 = S1\ S0\ D$$

Where D is the input data, Y0 to Y3 are output lines and S0 & S1 are select lines.From the above Boolean expressions, a 1-to-4 demultiplexer can be implemented by using four 3-input AND gates and two NOT gates as shown in figure below. The two selection lines enable the particular gate at a time.



Applications of Demultiplexer

- Used to connect a single source to multiple destinations.
- The Demux is used in communication systems to carry multiple data signals into a single transmission line.
- Used in Arithmetic Logic Units
- Used in serial to parallel converters in data communications.

Conclusion: -

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines. Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as Mux.
A demultiplexer is a device that takes a single input and gives one of the several output lines. A demultiplexer takes one single input data and then selects any one of the single output lines one at a time. It is the reverse process of a multiplexer. It is also called as a DEMUX or a data distributor. A DEMUX converts the input serial data line into output parallel data. A DEMUX gives '2n' outputs for 'n' selection lines with a single input.

Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.15

Shift Register with Parallel Load, And Shift Operation

Roll No: **20ES070**	**Date:**
Checked by:	**Score:**

OBJECTIVE: To design and verify the operation of shift register, using Multisim platform.

EQUIPMENT:

- VDD
- Clock Signal
- Green and Red Probes

COMPONENTS FOR SHIFT REGISTERS FLIP-FLOP:

- D Flip Flop
- AND Gate
- OR Gate
- NOT Gate

Introduction

Flip flops can store a single bit of binary data i.e. 1 or 0. But if we need to store multiple bits of data, we need multiple flip flops. As a single flip flop is used for one bit storage, n flip flops are connected in an order to store n bits of data. In digital electronics, a Register is a device which is used to store the information.

Flip flops are use in constructing registers. Register is a group of flip flops used to store multiple bits of data. For example, if a computer is to store 16-bit data, then it needs a set of 16 flip flops. The input and outputs of a register may be serial or parallel based on the requirement.

The series of data bits are stored by registers is called “Byte” or “Word” where a Byte is collection of 8 bits and a Word is collection of 16 bits (or 2 Bytes).

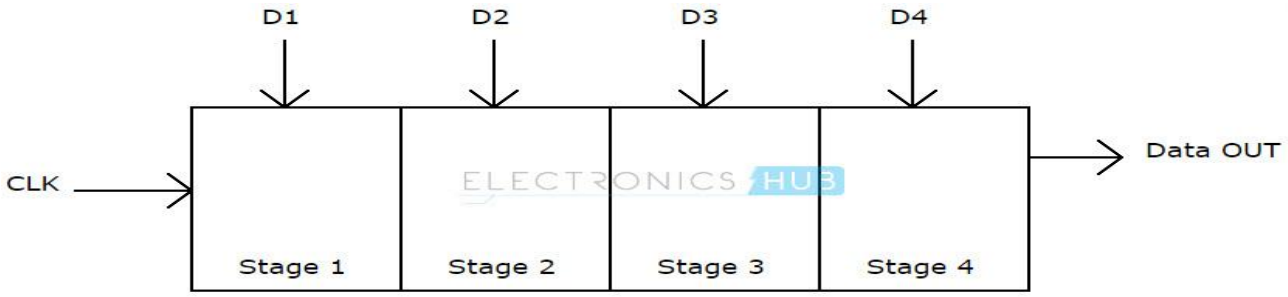
When several flip flops are connected in series, this arrangement is called a Register. The stored information can be transferred within the registers; these are called as ‘Shift Registers’. A shift register is a sequential circuit which stores the data and shifts it towards the output on every clock cycle.

Basically, shift registers are of 4 types. They are

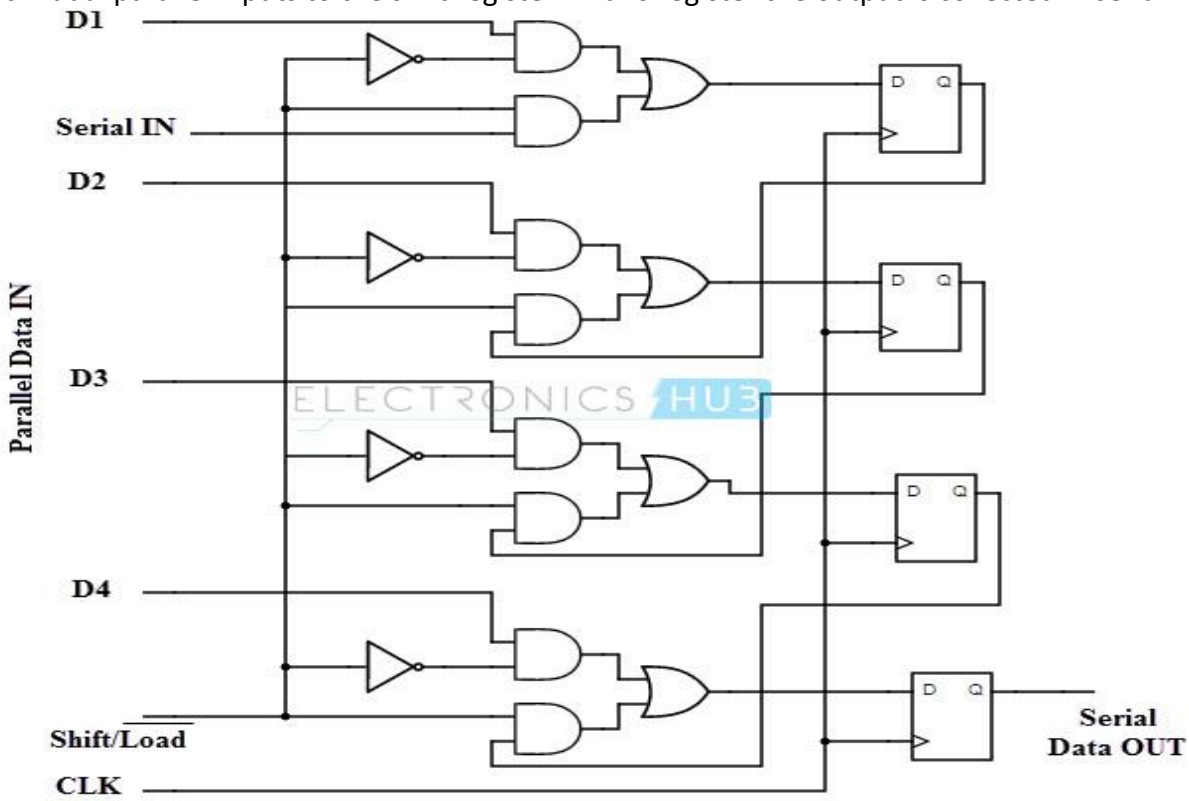
- ⇒ Serial In Serial Out shift register
- ⇒ Serial In parallel Out shift register
- ⇒ Parallel In Serial Out shift register
- ⇒ Parallel In parallel Out shift register

Parallel in Serial out shift register

The input to this register is given in parallel i.e., data is given separately to each flip flop and the output is collected in serial at the output of the end flip flop.



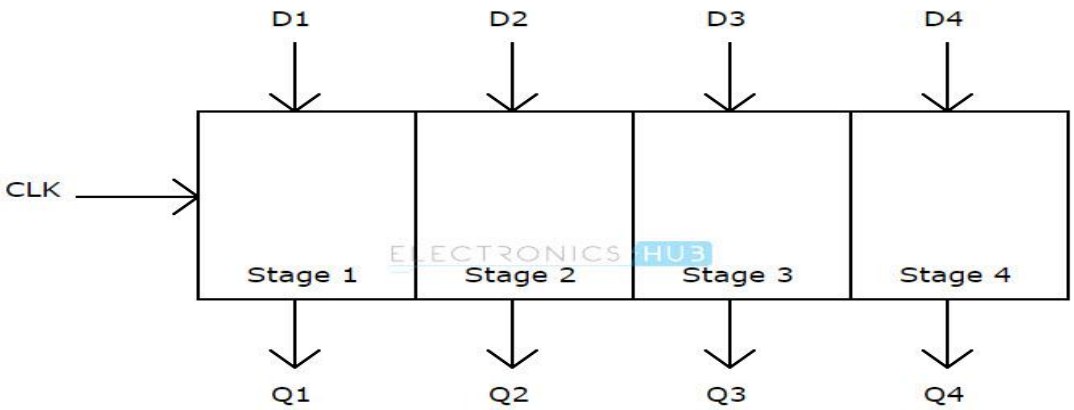
The clock input is directly connected to all the flip flops, but the input data is connected individually to each flip flop through a mux (multiplexer) at input of every flip flop. Here D1, D2, D3 and D4 are the individual parallel inputs to the shift register. In this register the output is collected in serial.



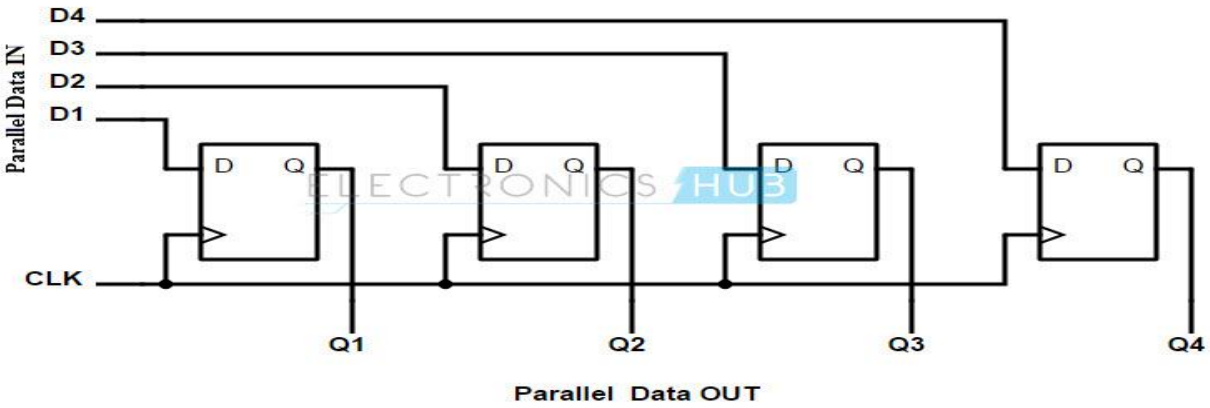
The output of the previous flip flop and parallel data input are connected to the input of the MUX and the output of MUX is connected to the next flip flop. A Parallel in Serial out (PISO) shift register converts parallel data to serial data. Hence, they are used in communication lines where a number of data lines are multiplexed into single serial data line.

Parallel in Parallel out shift register

In this register, the input is given in parallel, and the output also collected in parallel. The clear (CLR) signal and clock signals are connected to all the 4 flip flops. Data is given as input separately for each flip flop and in the same way, output also collected individually from each flip flop.



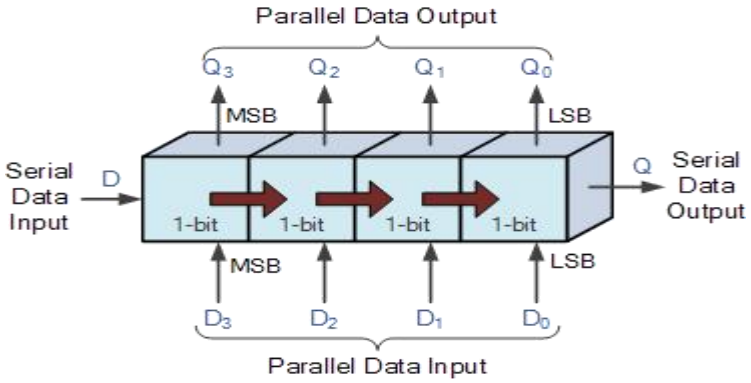
The above diagram shows the 4-stage parallel in parallel out register. Qa, Qb, Qc and Qd are the parallel outputs and Pa, Pb, Pc and Pd are the individual parallel inputs. There are no interconnections between any of the four flip flops.



A Parallel in Parallel out (PIPO) shift register is used as a temporary storage device and as a delay element similar to a SISO shift register.

The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data.

4 bit Shift Register



This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, hence the name Shift Register.

A shift register basically consists of several single bit “D-Type Data Latches”, one for each data bit, either a logic “0” or a “1”, connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

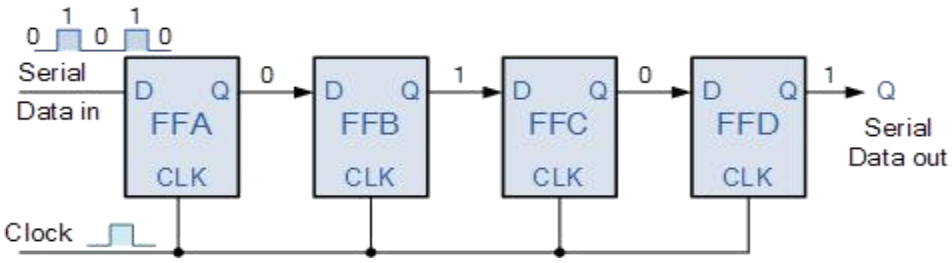
Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration.

The number of individual data latches required to make up a single Shift Register device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.

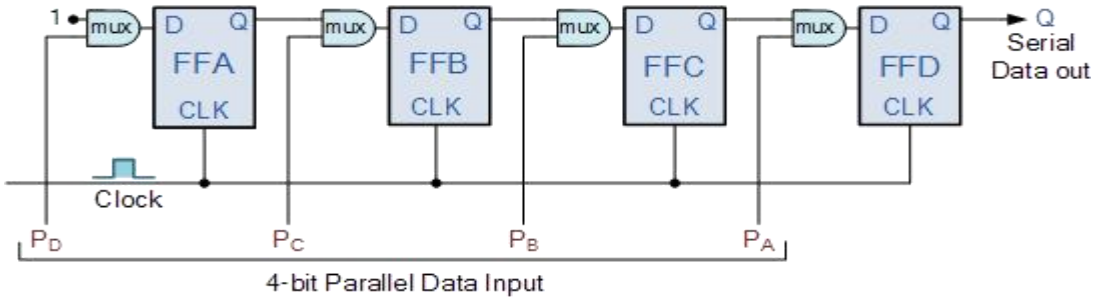
Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk) signal making them synchronous devices.

Shift register IC’s are generally provided with a clear or reset connection so that they can be “SET” or “RESET” as required. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:
Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.

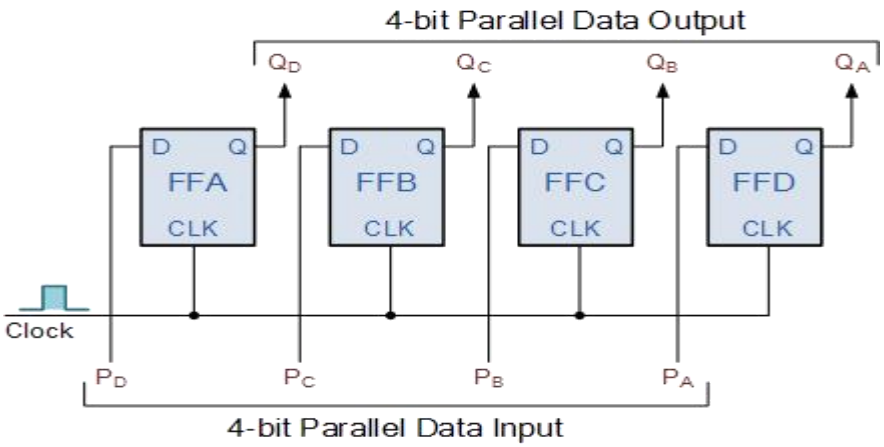
Serial-in to Serial-out (SISO) - the data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control.



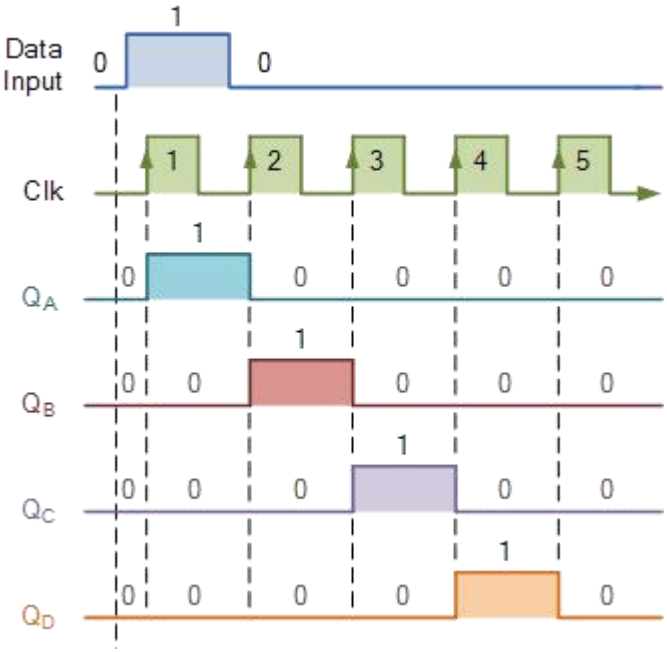
Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.



Parallel-in to Parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.



The effect of data movement from left to right through a shift register can be presented graphically as:

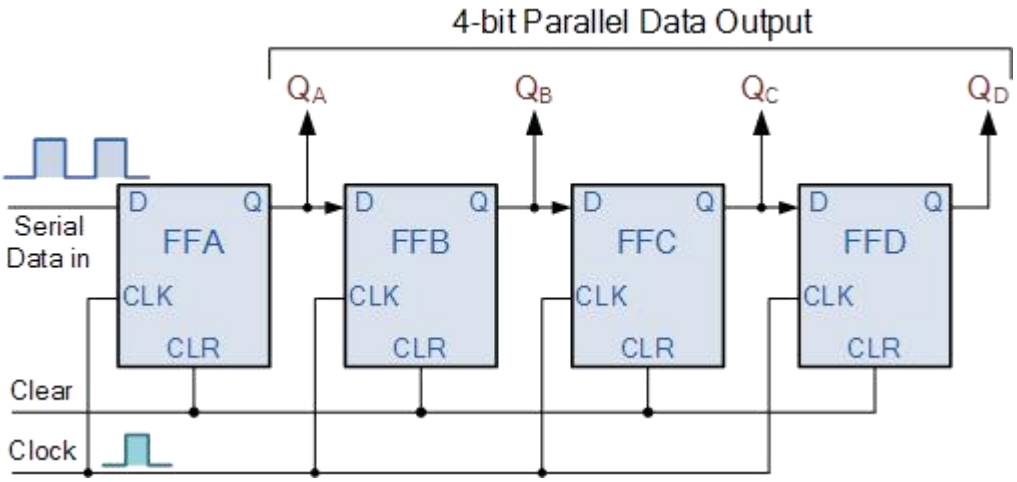


shift register data movement

Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it bidirectional. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

Serial-in to Parallel-out (SIPO) Shift Register

4-bit Serial-in to Parallel-out Shift Register



The operation is as follows. Lets assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs QA to QD are at logic level “0” ie, no parallel data output.

If a logic “1” is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting QA will be set HIGH to logic “1” with all the other outputs still remaining LOW at logic “0”. Assume now that the DATA input pin of FFA has returned LOW again to logic “0” giving us one data pulse or 0-1-0.

The second clock pulse will change the output of FFA to logic “0” and the output of FFB and QB HIGH to logic “1” as its input D has the logic “1” level on it from QA. The logic “1” has now moved or been “shifted” one place along the register to the right as it is now at QA.

When the third clock pulse arrives this logic “1” value moves to the output of FFC (QC) and so on until the arrival of the fifth clock pulse which sets all the outputs QA to QD back again to logic level “0” because the input to FFA has remained constant at logic level “0”.

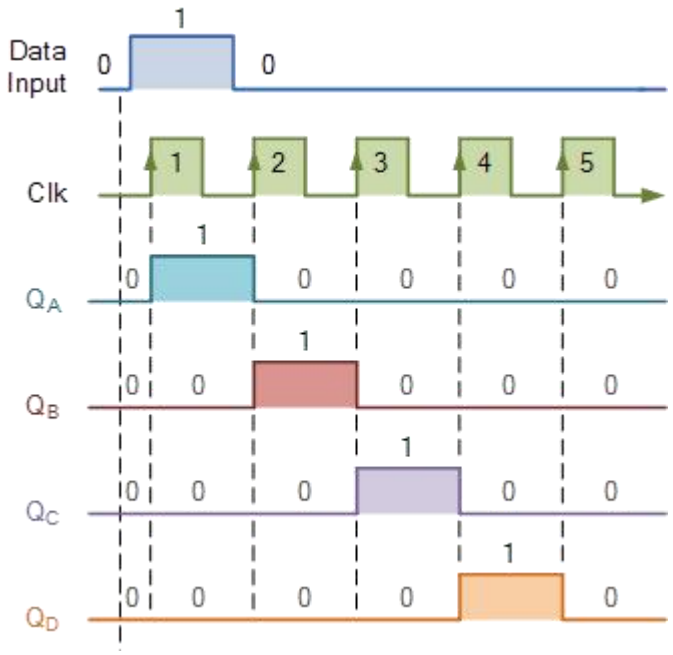
The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of QA to QD.

Then the data has been converted from a serial data input signal to a parallel data output. The truth table and following waveforms show the propagation of the logic “1” through the register from left to right as follows.

Basic Data Movement Through A Shift Register

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0

1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0



Note that after the fourth clock pulse has ended the 4-bits of data (0-0-0-1) are stored in the register and will remain there provided clocking of the register has stopped. In practice the input data to the register may consist of various combinations of logic “1” and “0”. Commonly available SIPO IC’s include the standard 8-bit 74LS164 or the 74LS594.

Serial-in to Serial-out (SISO) Shift Register

This shift register is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs QA to QD, this time the data is allowed to flow straight through the register and out of the other end. Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name Serial-in to Serial-Out Shift Register or SISO.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk). The logic circuit diagram below shows a generalized serial-in serial-out shift register.

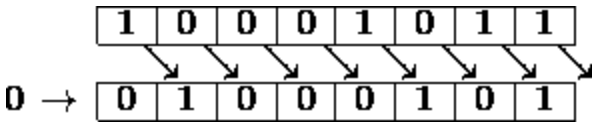
4-bit Serial-in to Serial-out Shift Register

serial in serial out shift register

Shift Operations

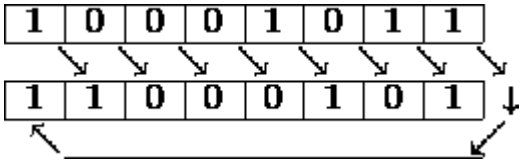
If is often necessary to align bits in a certain position in a word. The *shift* operations allow bits to be moved to the left or right in a word. There are **three types of shift operations: logical, rotate and arithmetic.**

A *logical* shift moves bits to the left or right. The bits which 'fall off' the end of the word are discarded, and the word is filled with 0's from the opposite end. A logical right shift of the 8-bit binary number 1000 1011 gives 0100 0101, as shown below:



Shift instructions include a repeat value, which is the number of times the single bit shift operation is repeated.

A *rotate* operation is a circular shift in which no bits are discarded. A rotate right of the 8 bit binary number 1000 1011 gives 1100 0101, as shown below.

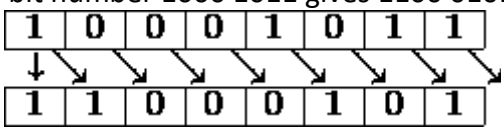


A right rotation by n bits of an n bit word returns the original word unchanged. A right rotation by n-1 bits is equivalent to a left rotation of 1 bit. The left rotation instruction is redundant because a left rotation of j bits is equivalent to a right rotation of n-j bits.

On positive integers, a logical left shift is equivalent to multiplication by 2 and a logical right shift is equivalent to division by 2. The *arithmetic shift* extends this operation to negative 2's complement integers.

An arithmetic right shift is similar to a logical right shift, except that the leftmost bits are filled with the sign bit of the original number instead of 0's.

An arithmetic right shift of the 8-bit number 1000 1011 gives 1100 0101, as shown below:



Parallel-in to Serial-out (PISO) Shift Register

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format in which all the data bits enter their inputs simultaneously, to the parallel input pins PA to PD of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at PA to PD.

This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this type of data register a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

4-bit Parallel-in to Serial-out Shift Register

parallel in serial out shift register

As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line. Commonly available IC's include the 74HC166 8-bit Parallel-in/Serial-out Shift Registers.

Parallel-in to Parallel-out (PIPO) Shift Register

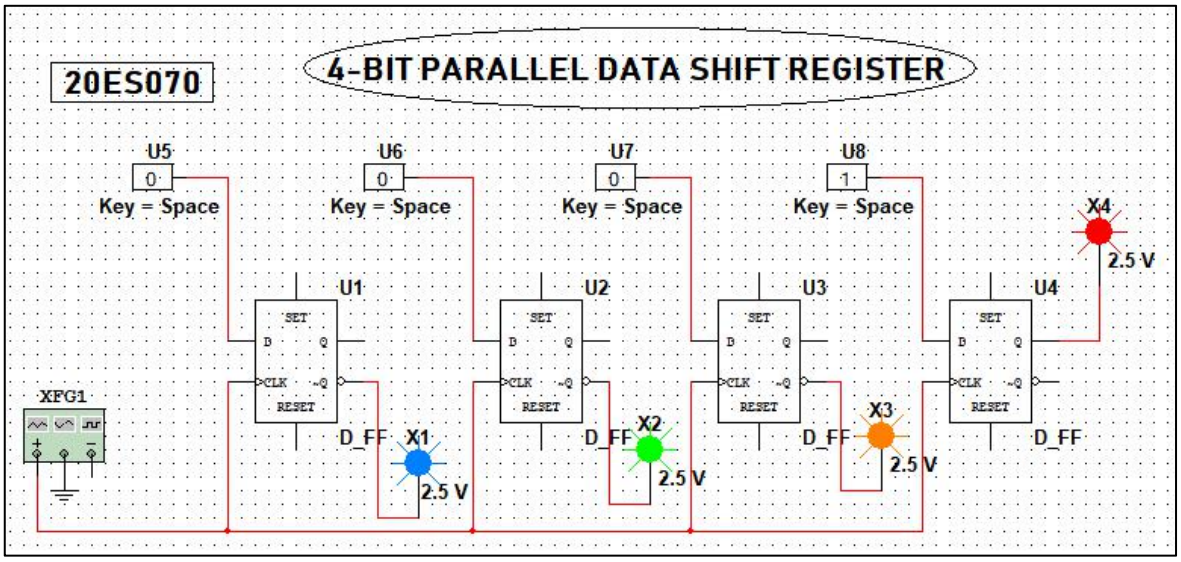
The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of shift register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins PA to PD and then transferred together directly to their respective output pins QA to QD by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

4-bit Parallel-in to Parallel-out Shift Register

parallel in parallel out shift register

The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

Task: perform the simulation, and show the results for various operations



Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.16

Study of CD 4017-decade Counter IC

OBJECTIVE: To design and verify the operation of Synchronous Decade counter and CD-4017 IC using MultiSim platform/ hardware design using bread board.

EQUIPMENT:

- VDD
- Clock Signal
- Green and Red Probes

COMPONENTS FOR SYNCHRONOUS FLIP-FLOP:

- JK Flip Flop
- AND Gate

Introduction to Synchronous Counter

The counters which use clock signal to change their transition are called "Synchronous counters". This means the synchronous counters depends on their clock input to change state values. In synchronous counters, all flip flops are connected to the same clock signal and all flip flops will trigger at the same time.

Synchronous counters are also known as 'Simultaneous counters. There is no propagation delay and no ripple effect in synchronous counters.

Different types of Synchronous Counters

There are many types of synchronous counters available in digital electronics. They are listed below.

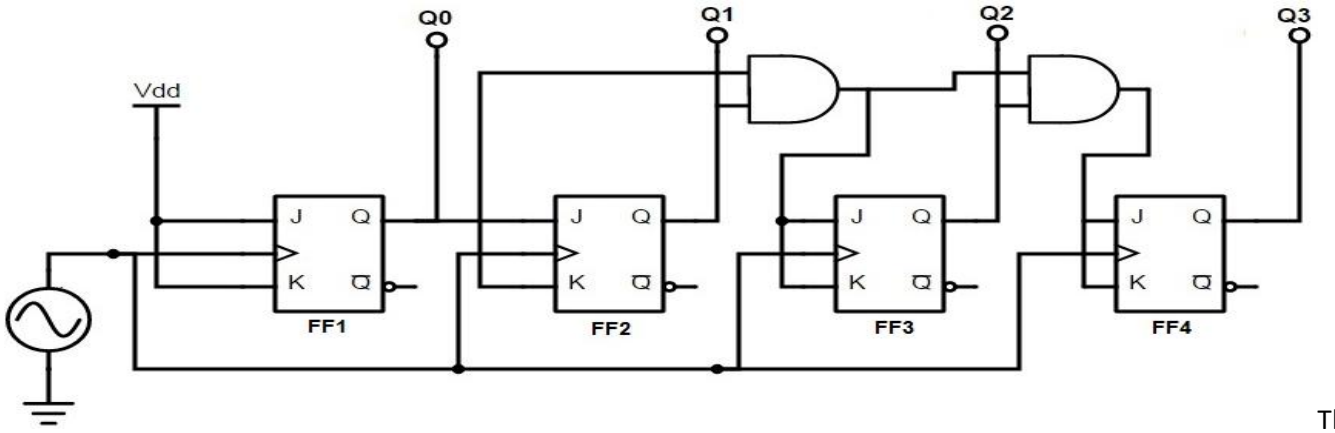
- Binary counters
- 4-bit synchronous UP counter
- 4-bit synchronous DOWN counter
- 4 bit synchronous UP / DOWN counter
- Loadable counters
- BCD counters
- Ring counters
- Johnson counters etc.

4-bit Synchronous UP Counter

The 4-bit up counter shown in below diagram is designed by using JK flip flop. External clock pulse is connected to all the flip flops in parallel.

For designing the counters JK flip flop is preferred. The significance of using JK flip flop is that it can toggle its state if both the inputs are high, depending on the clock pulse.

The inputs of first flip flop are connected to HIGH (logic 1), which makes the flip flop to toggle, for every clock pulse entered it. So, the synchronous counter will work with single clock signal and changes its state with each pulse.



The output of first JK flip flop (Q) is connected to the input of second flip flop. The AND gates (which are connected externally) drives the inputs of other two flip flops. The inputs of these AND gates, are supplied from previous stage lip flop outputs.

If inputs of FF2 are connected directly to the Q1 output of FF1, the counter would not function properly. This is because, the Q1 value is high at count of 210, this means that the FF2 flip flop will toggle for the 3rd clock pulse. This results in wrong counting operation, gives the count as 710 instead of 410.

To prevent this problem AND gates are used at the input side of FF2 and FF3. The output of the AND gate will be high only when the Q0, Q1 outputs are high. So, for the next clock pulse, the count will be 00012.

Similarly, the flip flop FF3 will toggle for the fourth clock pulse when Q0, Q1 and Q2 are high. The Q3 output will not toggle till the 8th clock pulse and will again remain high until 16th clock pulse. After the 16th clock pulse, the q outputs of all flip flops will return to 0.

Operation

In the up counter the 4-bit binary sequence starts from 0000 and increments up to 1111.Before understanding the working of the above up counter circuit know about JK Flip flop.

In the above circuit as the two inputs of the flip flop are wired together. So, there are only two possible conditions that can occur, that is, either the two inputs are high or low.

If the two inputs are high, then JK flip-flop toggles and if both are low JK flip flop remembers i.e., it stays in the previous state.

Let us see the operation. Here clock pulse indicates edge triggered clock pulse.

1. In the first clock pulse, the outputs of all the flip flops will be at 0000.
2. In the second clock pulse, as inputs of J and k are connected to the logic high, output of JK flip flop (FF0) change its state. Thus the output of the first flip-flop (FF0) changes its state for every clock pulse .This can be observed in the above shown sequence .The LSB changes its state alternatively. Thus producing -0001
3. In the third clock pulse next flip flop (FF1) will receive its J K inputs i.e. (logic high) and it changes its state. At this state FF0 will change its state to 0. And thus, input on the FF1 is 0. Hence output is -0010
4. Similarly, in the fourth clock pulse FF1 will not change its state as its inputs are in low state, it remains in its previous state. Though it produces the output to FF2, it will not change its state due to the presence of AND gate. FF0 will again toggle its output to logic high state. Thus, Output is 0011.
5. In the fifth clock pulse, FF2 receives the inputs and changes its state. While, FF0 will have low logic on its output and FF1 will also be low state producing 0100.

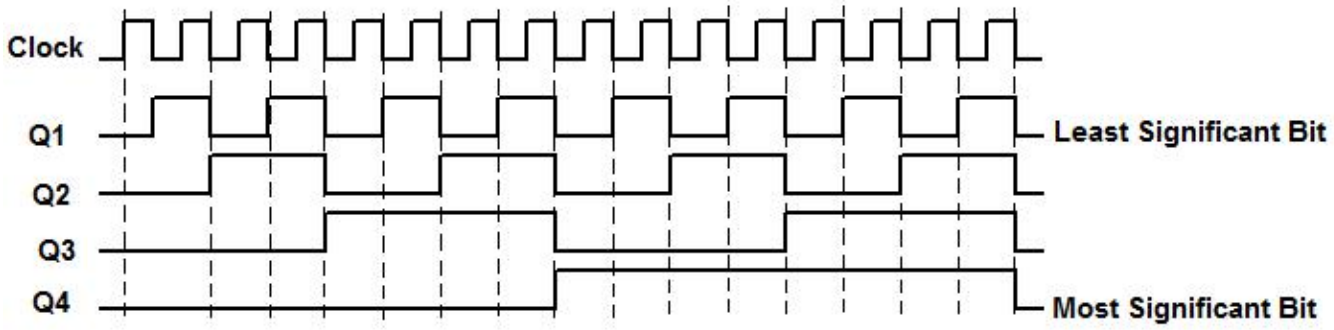
This process continuous up to 1111.Working can be explained in the below table. The above mentioned working of synchronous counter can be clearly given in the below table.

Clock pulse	Q4	Q3	Q2	Q1	Output	Output
1st Clock Pulse	Low	Low	Low	Low	0	0
2nd Clock Pulse	Low	Low	Low	High (starts toggling)	1	1
3rd Clock Pulse	Low	Low	High	Low	2	10
4th Clock Pulse	Low	Low (AND condition is not satisfied)	High (Remembers)	High	3	11
5th Clock Pulse	Low	High	Low (toggles)	Low	4	100
6th Clock Pulse	Low (AND condition not satisfied)	High (Remembers)	Low (remembers)	High	5	101
7th Clock Pulse	Low (AND condition satified)	High	High	Low	6	110
8th Clock Pulse	Low	High	High	High	7	111
9th Clock Pulse	High	Low	Low	Low	8	1000
10th Clock Pulse	High(Remembers)	Low	Low	High	9	1001
11th Clock Pulse	High	Low	High	Low	10	1010
12th Clock Pulse	High	Low	High	High	11	1011
13th Clock Pulse	High	High	Low	Low	12	1100
14th Clock Pulse	High	High	Low	High	13	1101
15th Clock Pulse	High	High	High	Low	14	1110
16th Clock Pulse	High	High	High	High	15	1111

The below table shows the outputs of 4 flip flops Q1, Q2, Q3, Q4. The first flip-flop toggles on every edge triggered pulse. While the second one triggers only if its inputs are high at a given clock pulse. The third flip-flop toggles if the two outputs Q1 and Q2 are high. Similarly, Q4 will toggle if all the three Q1, Q2, Q3 are high.

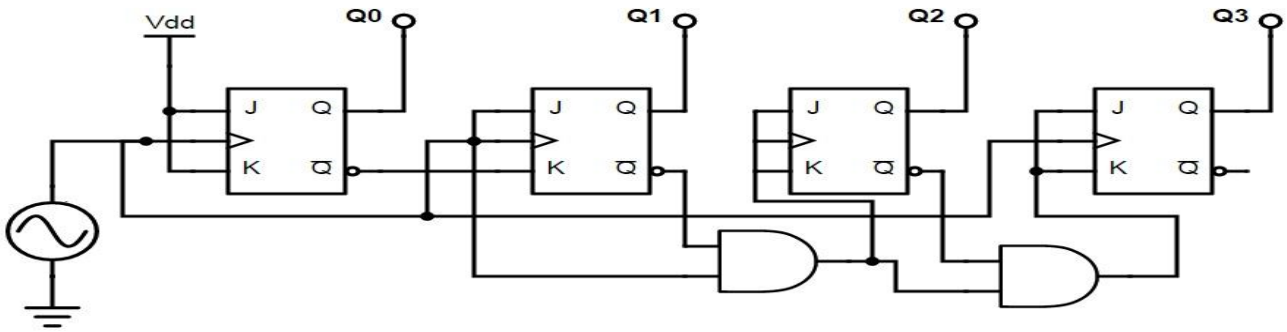
After reaching zero again the three flip flops toggles to logic low i.e 0000 and again count starts.

Timing diagram for up counter is shown below.



4-bit Synchronous DOWN Counter

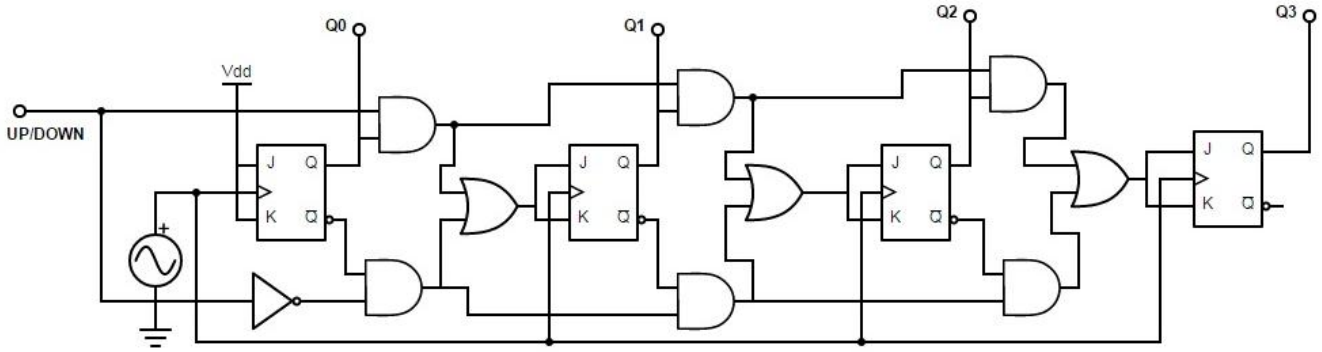
Down counter counts the numbers in decreasing order. This is similar to an up counter but should decrease its count. So inputs of JK flip- flop are connected to the inverted Q (Q'). The 4 bit down counter shown in below diagram is designed by using JK flip flop. The same external clock pulse is connected to all the flip flops.



As the counter must count down the sequence, initially all the inputs will be in high state as they have to count down the sequence. It will start with 1111 and ends with 0000, like the up counter.
In the down counter it should be remembered that, preceding flip flop will toggles only if front flip flop produces low logic at its output.

4 bit Synchronous Up/Down Counter

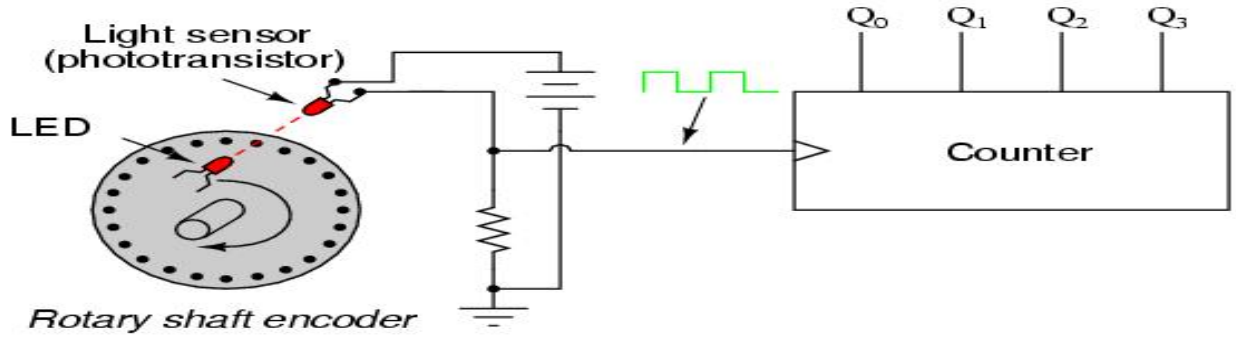
The above two counters can be implemented in a single counter called up down counter. This can be selected from its input. The design of up/ down counter with JK flip flops is shown below.



The up/ down counter has “Up” and “Down” count modes by having 2 input AND gates, which are used to detect the appropriate bit conditions for counting operation. OR gates are used to combine the outputs of AND gate, from each JK flip flop.
We provide a up/ down control line which enables upper or lower series of AND gates to pass the outputs of JK flip flops, Q, Q’ to the next stage of flip flop, in the cascaded arrangement.
If the up /down control line is set to HIGH, then the top AND gates are in enable state and the circuit acts as UP counter. If the up /down control line is set to low, then the bottom AND gates are in enable state and the circuit acts as DOWN counter.

Applications of Synchronous Counters

The most common and well-known application of synchronous counters is machine motion control, the process in which the rotary shaft encoders convert the mechanical pulses into electric pulses. These pulses will act as clock input of the up/ down counter and will initiate the circuit motion.



This circuit consists of photo transistor or light sensor and a LED connected to the rotor shaft. This arrangement is connected to the UP/ DOWN counter. When the machine started to move, it turns the encoder shaft by connecting and disturbing (making and breaking) the light beam between the light sensor and LED.
By this motion, the rotor creates clock pulses to increase the count of the up/ down counter circuit. So, the counter note downs the motion of the shaft and gives the value that how much distance the rotor has moved.
To count the motion of the rotor shaft we increment the count by moving shaft in one direction and decrement the count by moving in another direction. We also use an encoder /decoder circuit to differentiate the direction of motion.

4017-decade IC

The CD4017 is a CMOS Decade counter IC. CD4017 is used for low range counting applications. It can count from 0 to 10 (the decade count). The circuit designed by using this IC will save board space and also time required to design the circuit. CD4017 is as 'Johnson 10 stage decade counter'.

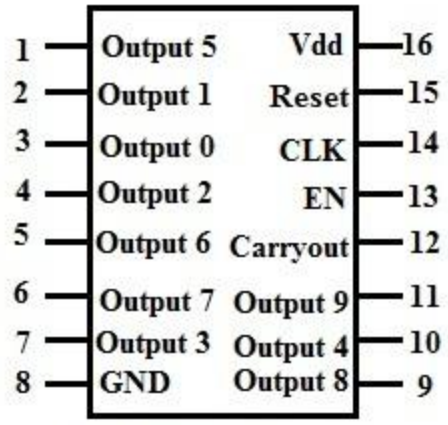
Features

- The supply voltage of this IC is 3V to 15V.
- It is compatible with TTL (Transistor -Transistor Logic).
- The clock speed or operational speed of CD4017 IC is 5 MHz

This IC is also used in electronic industries, automotive industries, manufacturing medical electronic devices, alarms and in electronic instrumentation devices.

CD4017 Pin description

It has 16 I/O pins.



4017 -Decade Counter

Output pins of CD4017 (Pin 1 to 7 & 9 to 11)

- Pins 1 to 7 and 9 to 11 are outputs pins.
- These pins changes to 'high' level one by one (one after another) in a sequence. For each clock signal each pin goes high in a sequence.

Enable pin/Clock Inhibit (Pin 13)

- Enable pin enables the CD4017 IC.IC is enabled when the pin is active low.
- To disable or switch off the IC, this pin should be connected to active high input. When this pin is active high, it ignores the clock signals.

Clock pin (pin 14)

- Clock signal provided to 14 the is responsible for sequential output.
- When the first clock pulse is detected pin 3 goes, for next clock pulse pin2 goes high, like this sequence is formed.
- The important thing to remember is, if we don't connect any clock signal to this input pin, it must be connected to either positive or negative voltage supply.
- It is not left unconnected as per the CMOS input standard rules.
- The clock input pin (pin number 14) responds only to the positive voltage signal or positive clock

Reset pin (Pin 15)

- Reset pin resets the output of the sequence. That is the current state of the output sequence is set to initial state.
- Reset pin should be connected to ground to reset the circuit.

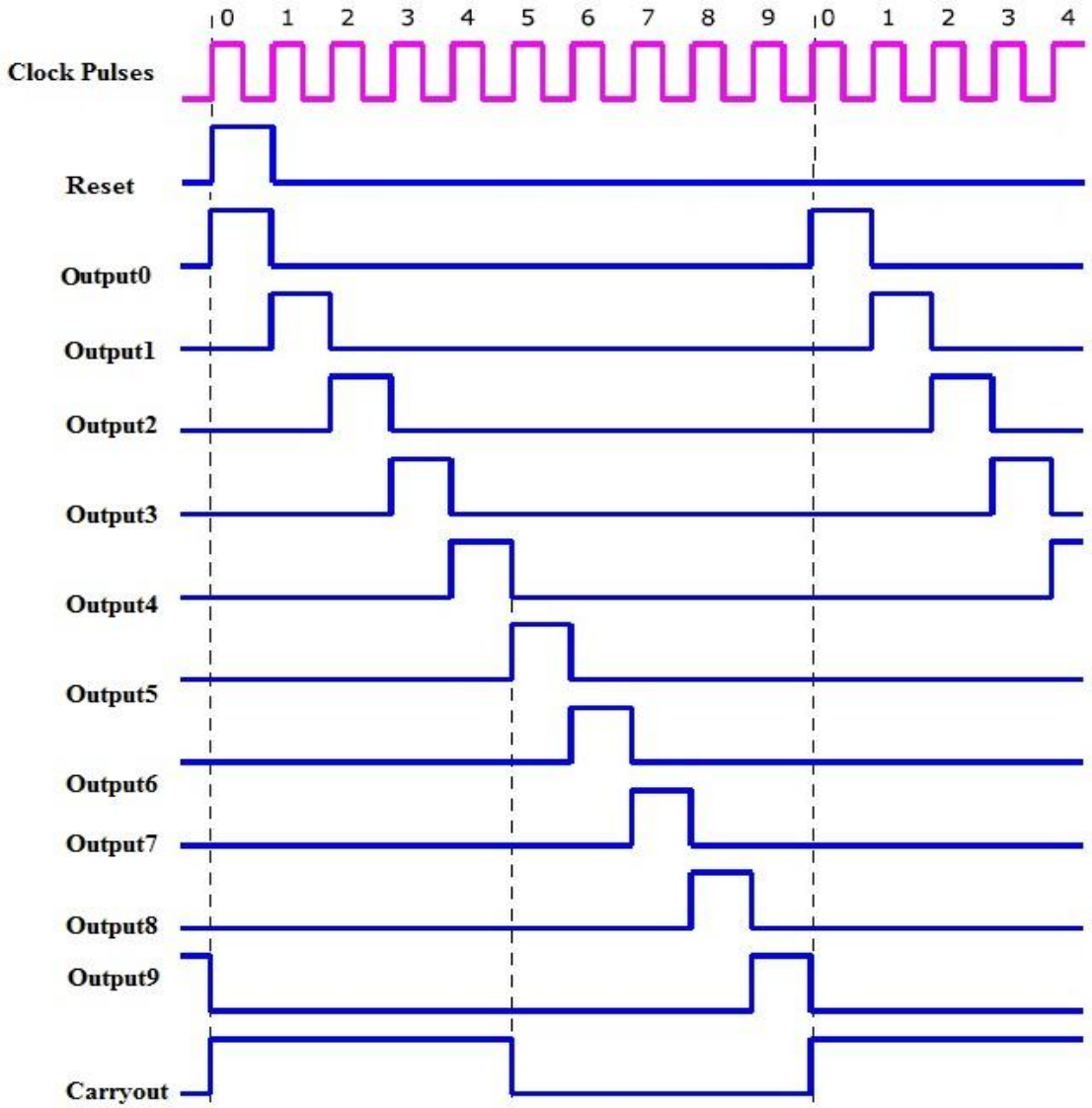
Ground pin & supply pin (Pin 8 & Pin 16)

Pin number 8 acts as ground and it must be connected to negative supply voltage & pin number 16 is the supply pin for CD4017 and it is connected to positive voltage supply.

Carry out pin (pin12)

The pin 12 is supplied with the CARRY OUT signal. It completes one full cycle for every 10 clock cycles. This is used to 'ripple' the IC, which means to delay in counting operations.

Counting operation of CD4017 using waveforms

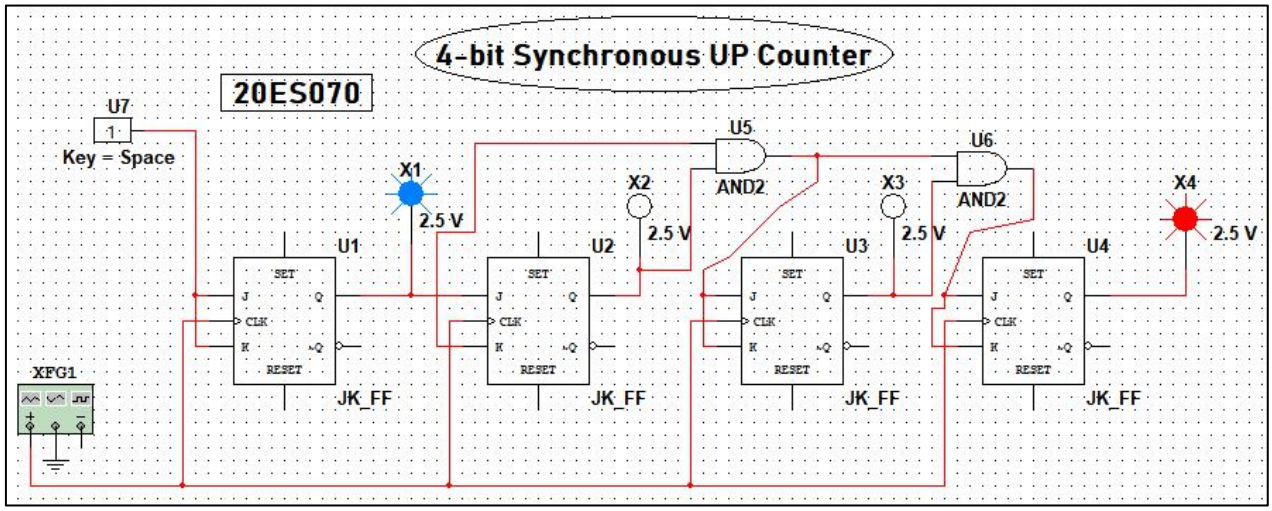


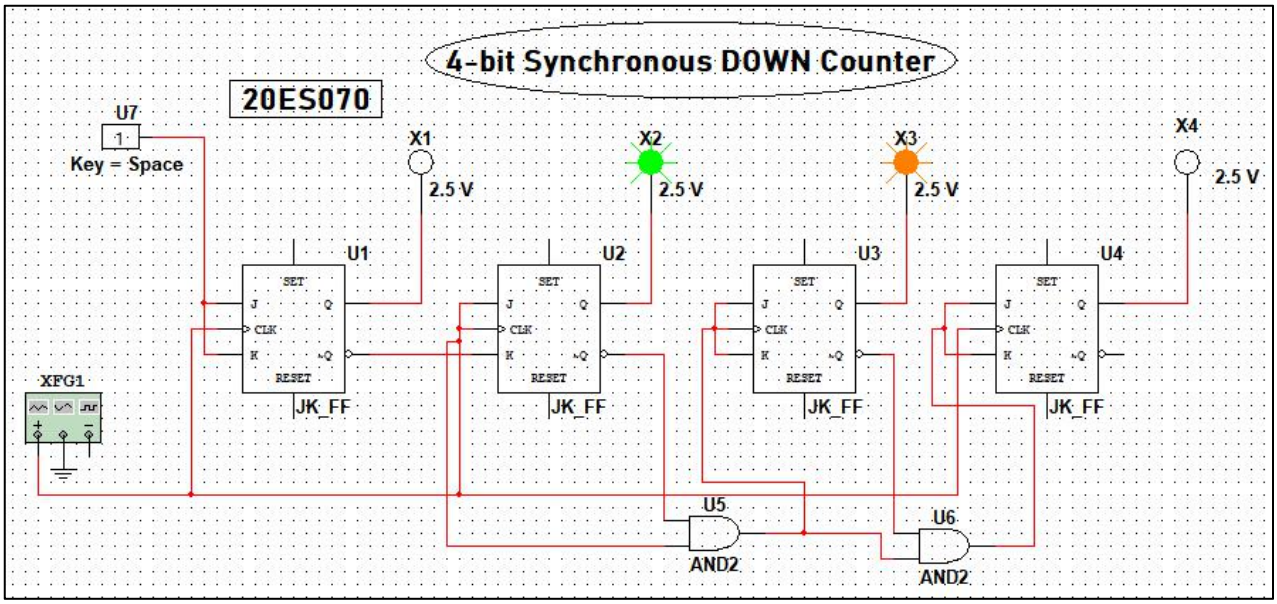
This is the timing diagram of the CD4017 with, shows us the comparison and explains the counting sequence of the outputs, shifting from one pin to its next.

If we observe that, before applying the clock signal, the **RESET** is set to **High**, so the reset pin input sets all the output to their initial state.

Then the output of the first output pin 3 will be high. Next this output is shifted to its next output pin and this sequence continues till the next clock cycle.

Task: show the design/ simulated results:





Mehran University of Engineering & Technology, Jamshoro
Department of Electronic Engineering
DIGITAL ELECTRONICS
Lab NO.17

Ring Counter and Johnson Counter

Roll No: 20ES070	**Date:**
Checked by:	**Score:**

OBJECTIVE: To design and Implement counters using shift registers – Ring Counter and Johnson Counter, at Multisim platform.

EQUIPMENT:

- VDD
- Clock Signal
- Green and Red Probes

COMPONENTS FOR SHIFT REGISTERS FLIP-FLOP:

- D Flip Flops
- LEDs
- Wires
- Switch

Introduction

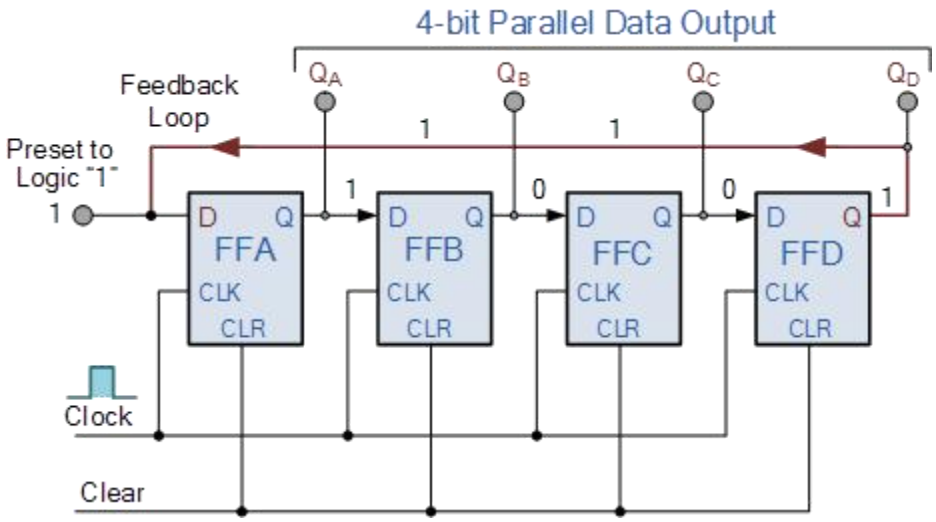
Ring Counter

A ring counter is basically a circulating shift register in which the output of the MSB is fedback to the input of the LSB. Here the 4 bit ring counter is constructed using D flip-flops. The output of each stage is shifted into the next stage on the positive edge of a clock pulse. Here the clear signal is high, all the flip-flops except first one FF are reset to 0.FF0 is preset to 1 instead.

Johnson Counter

They are a variation of standard ring counters, with the inverted output of the last stage fedback to the first stage. They are also known as twisted ring counters. An n-stage Johnson counter yields a count sequence of length 2^n,so it may be considered to be mod 2^n counter.

Ring counter Deign using D FFs:



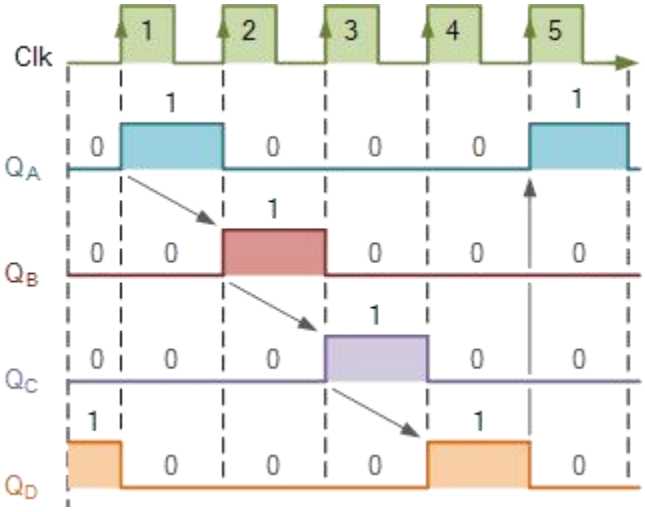
Ring counter

Truth table for Ring Counter:

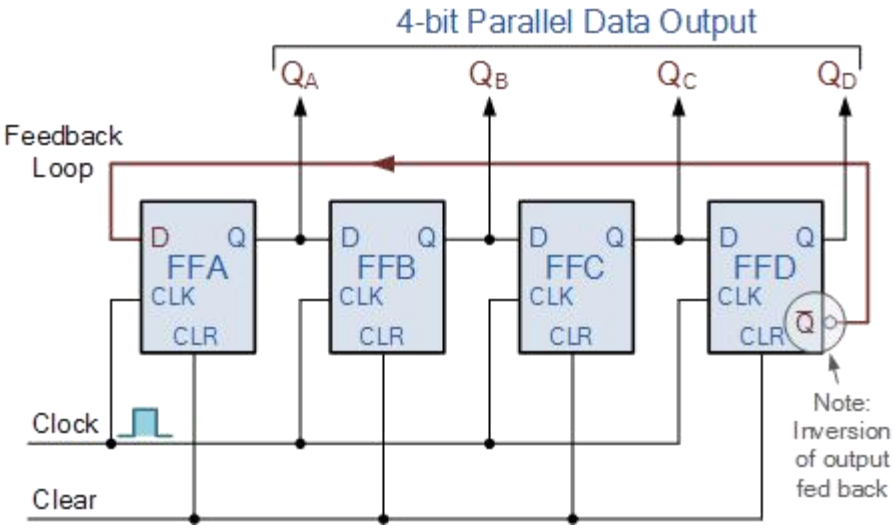
PRESETED 1

ORI	CLK	Q0	Q1	Q2	Q3
low	X	1	0	0	0
high	low	0	1	0	0
high	low	0	0	1	0
high	low	0	0	0	1
high	low	1	0	0	0

Timing Diagram for Ring Counter:



2. **Johnson Counter:**



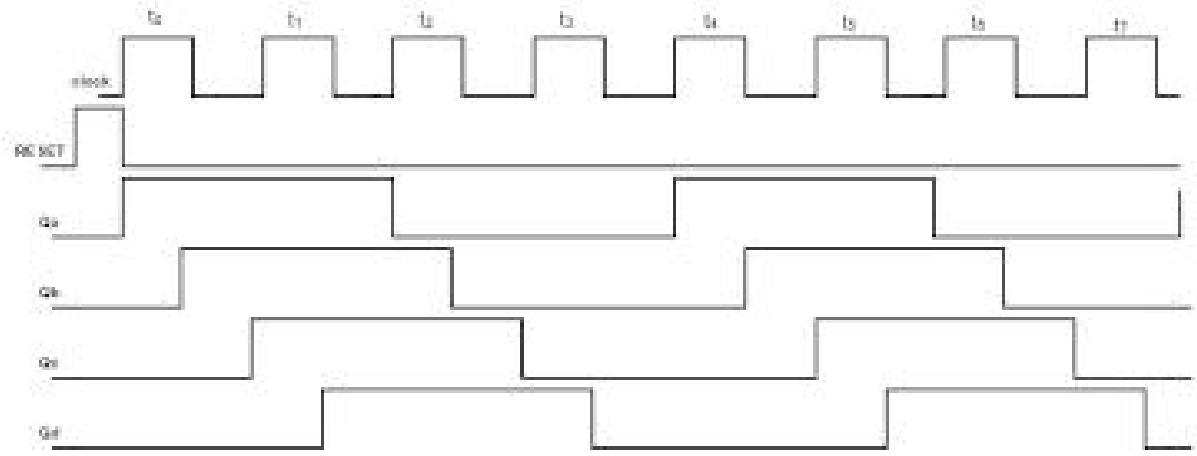
This inversion of Q before it is fed back to input D causes the counter to "count" in a different way. Instead of counting through a fixed set of patterns like the normal ring counter such as for a 4-bit counter, "0001"(1), "0010"(2), "0100"(4), "1000"(8) and repeat, the Johnson counter counts up and then down as the initial logic "1" passes through it to the right replacing the preceding logic "0".

A 4-bit Johnson ring counter passes blocks of four logic "0" and then four logic "1" thereby producing an 8-bit pattern. As the inverted output Q is connected to the input D this 8-bit pattern continually repeats. For example, "**1000", "1100", "1110", "1111", "0111", "0011", "0001", "0000"** and this is demonstrated in the following table below.

Truth Table for a 4-bit Johnson Ring Counter

Clock Pulse No	FFA	FFB	FFC	FFD
0	0	0	0	0
1	**1**	0	0	0
2	**1**	**1**	0	0
3	**1**	**1**	**1**	0
4	**1**	**1**	**1**	**1**
5	0	**1**	**1**	**1**
6	0	0	**1**	**1**
7	0	0	0	**1**

As well as counting or rotating data around a continuous loop, ring counters can also be used to detect or recognize various patterns or number values within a set of data. By connecting simple logic gates such as the *AND* or the *OR* gates to the outputs of the flip-flops the circuit can be made to detect a set number or value.



Timing diagram of Johnson Counter

Task: Perform the simulation, and show the results for Ring and Johnson Counter.