Devang Patel Institute of Advance Technology and Research, CHARUSAT

CE252 Digital Electronics

20DCS103 - RUSHIK RATHOD

DEPSTAR CSE-2 : Batch-B

Practical List

1. Study of logic gates (AND, OR, NOT, NAND, NOR, Ex•OR).
2. Draw a circuit diagram and Verify the truth table of Ex•OR & Ex•NOR gates.
3. Verify the operation of NAND & NOR gate as universal gates.
4. Study of 4•variable K•map and verify using example.
5. Implement half adder and half subtractor circuit.
6. Implement full adder combinational circuit.
7. Show the conversion from BCD to Excess•3.
8. Study 4•bit magnitude comparator and implement it.
9. Implement 8:1 multiplexer.
10. Demonstrate the operation of RS, JK &D flip•flops. Verify truth table, Excitation table and functional table.
11. Study and Implement different types of shift register.
12. Implement the operation of binary and decade counter.

Experiment No. 1

Aim: Study of logic gates (AND, OR, NOT, NAND, NOR, Ex•OR).

Apparatus: connecting wires, power supply, bread board, ICs as follow

Sr. No.	Component	Specification	Quantity
1	AND Gate	IC 7408	1
2	OR Gate	IC 7432	1
3	NOT Gate	IC 7404	1
4	NAND Gate	IC 7400	1
5	NOR Gate	IC 7402	1
6	X-OR Gate	IC 7486	1

Theory:

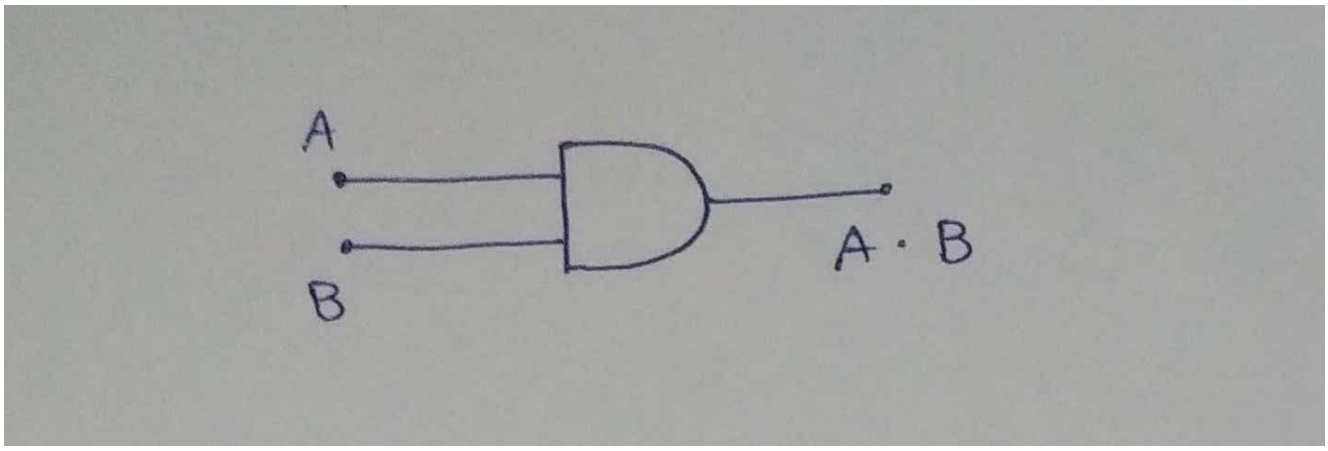
Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output.

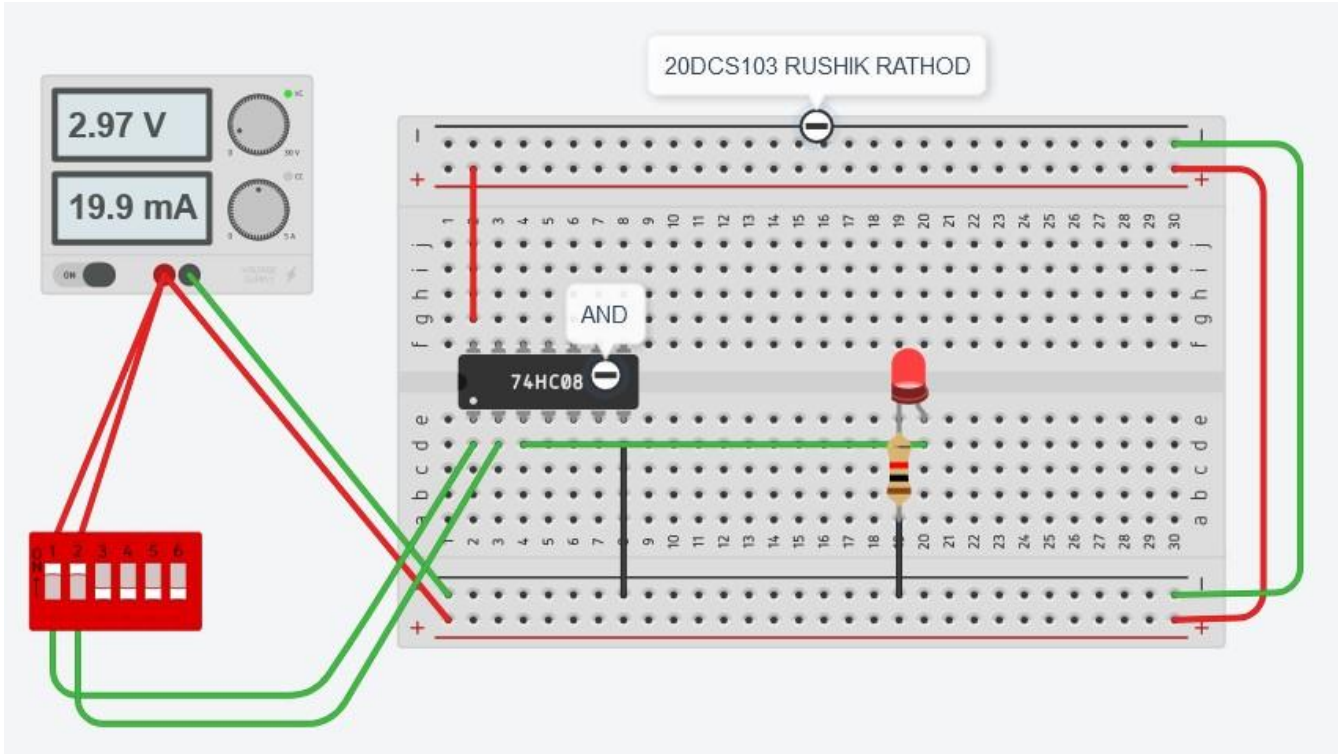
OR, AND & NOT are basic gates. NAND, NOR, XOR are known as universal gates. Basic gates can be obtained from all this gate.

AND Gate:

The AND gate performs a logical multiplication commonly known as AND function. The output is high only when both the input are either one high or one low. When both the input are high the output is low level.

SYMBOL:

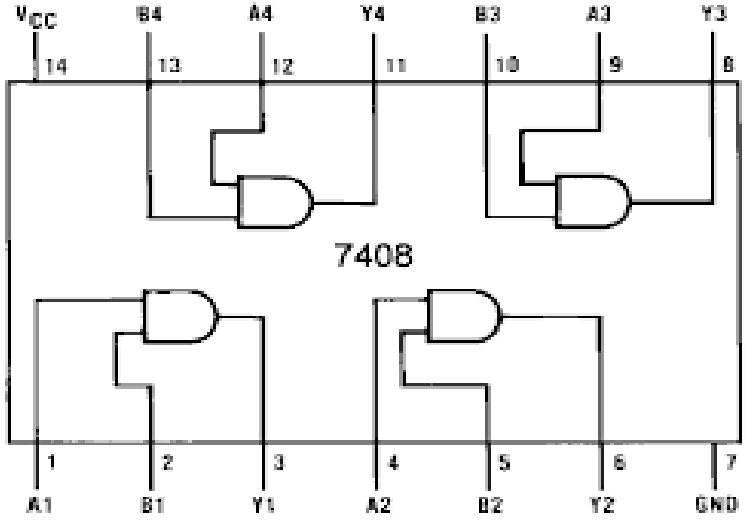




Observation Table:

Input	Input	Output
0	0	0
0	1	0
1	0	0
1	1	1

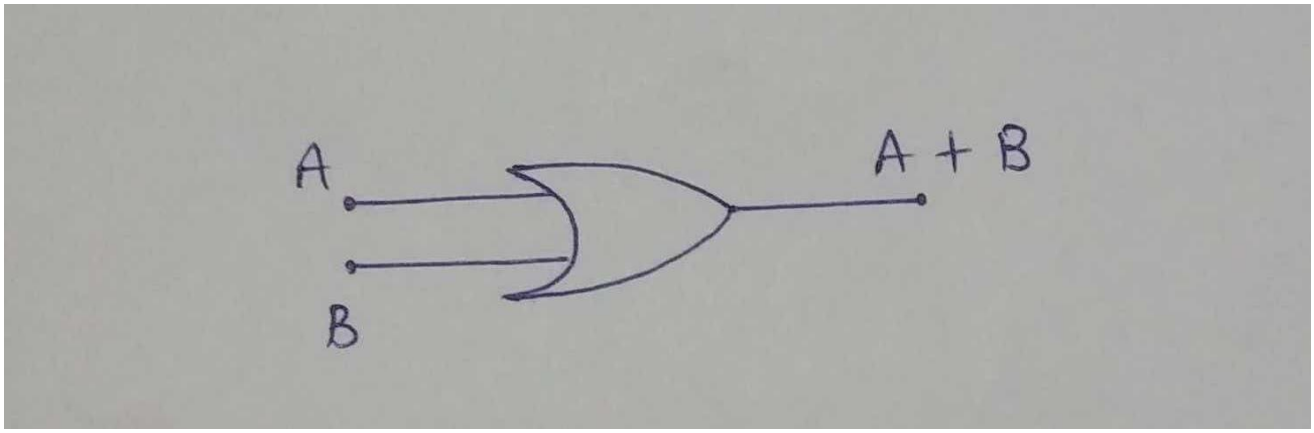
IC Diagram:

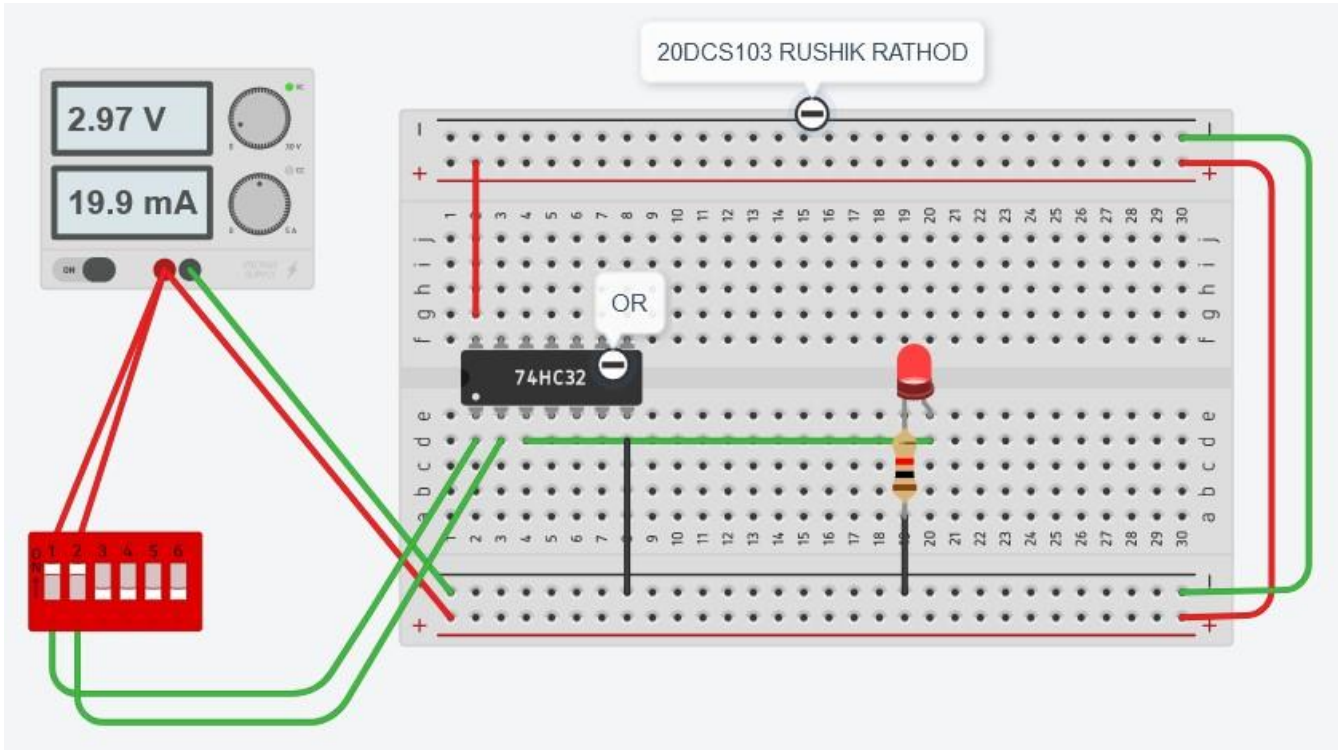


OR Gate:

The OR gate performs a logical addition commonly knows as OR function. The output is high when any one of the inputs is high and the output is low level when both the inputs are low.

Symbol:

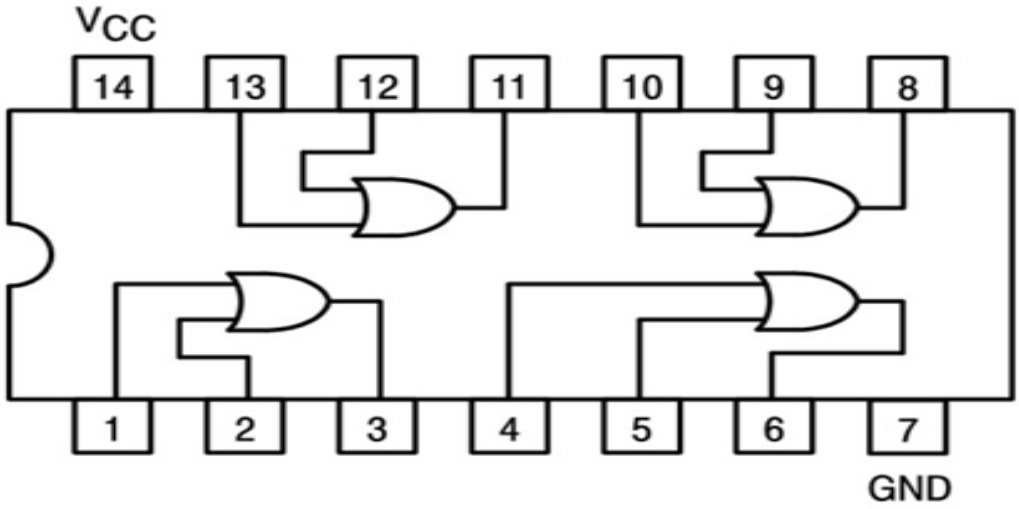




Observation Table:

Input	Input	Output
0	0	0
0	1	1
1	0	1
1	1	1

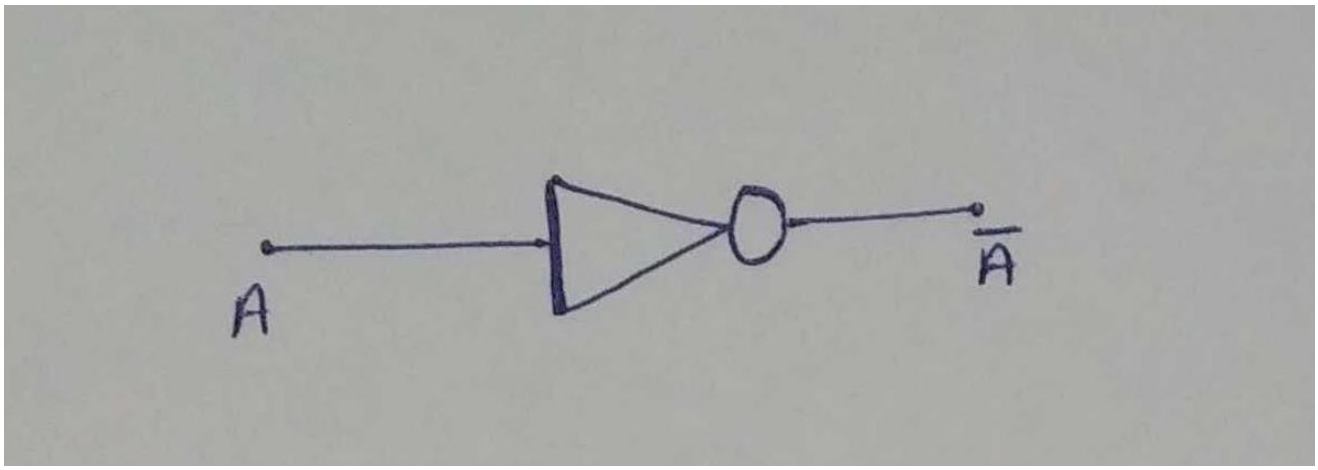
IC Diagram:

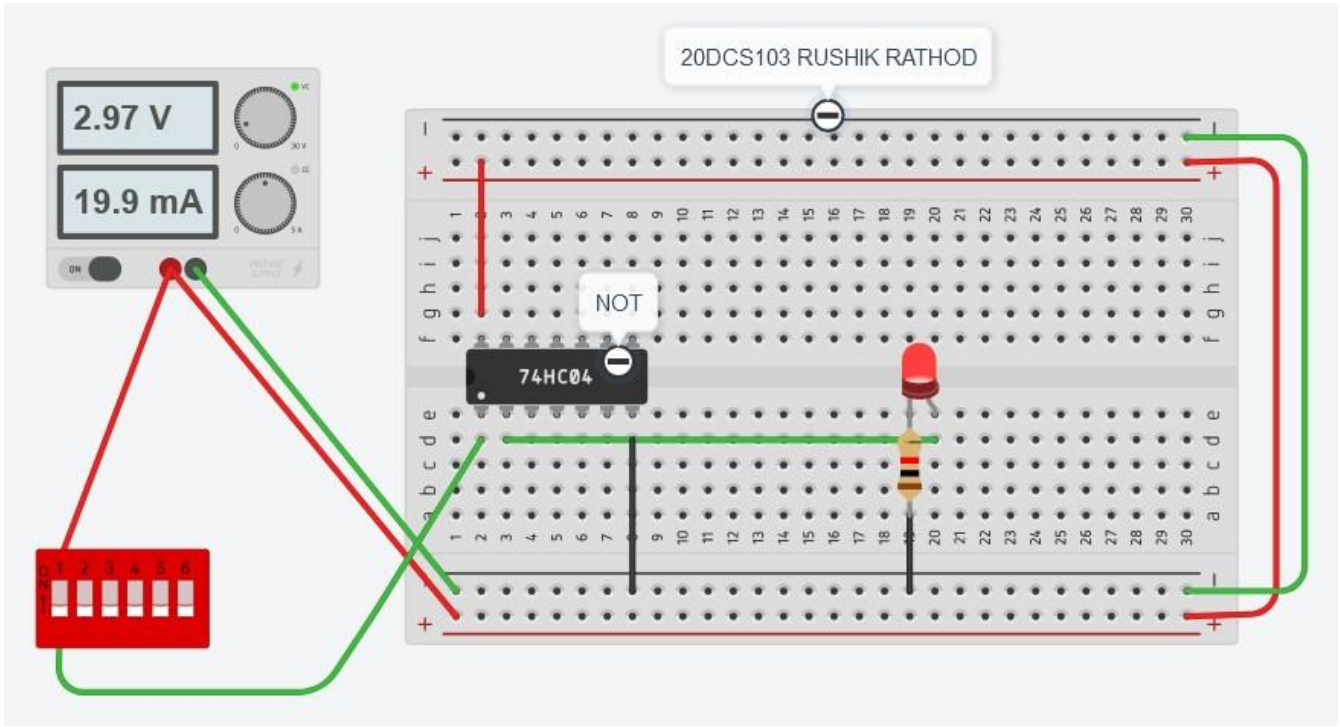


NOT Gate:

The NOT gate is called an inverter. The output is high when the input is low.The output is low when the input is high.

Symbol:

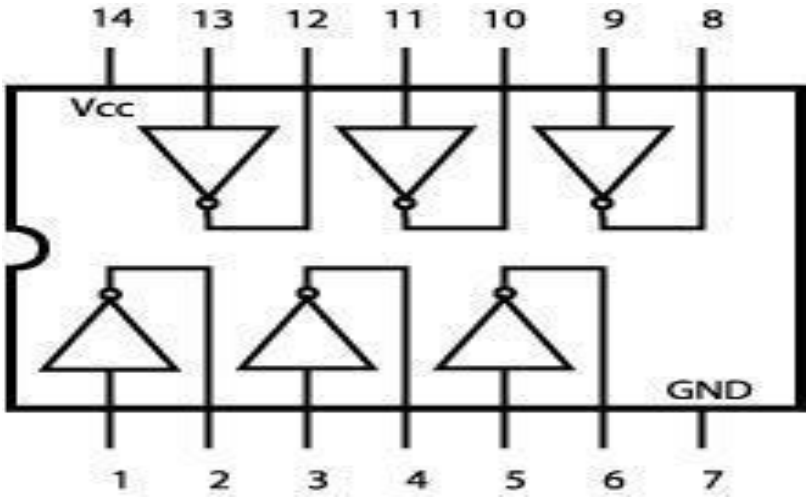




Observation Table:

Input	Output
0	1
1	0

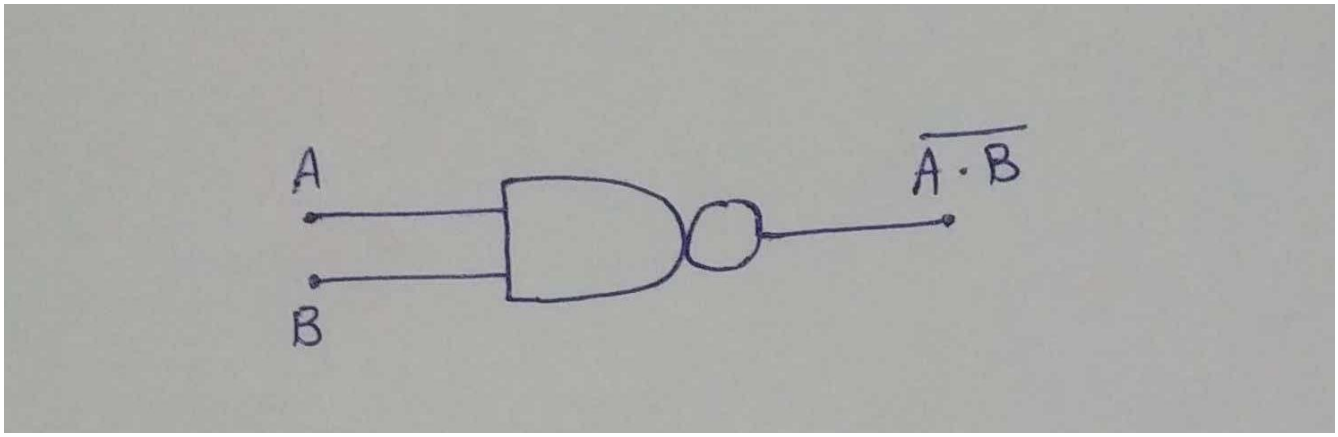
IC Diagram:

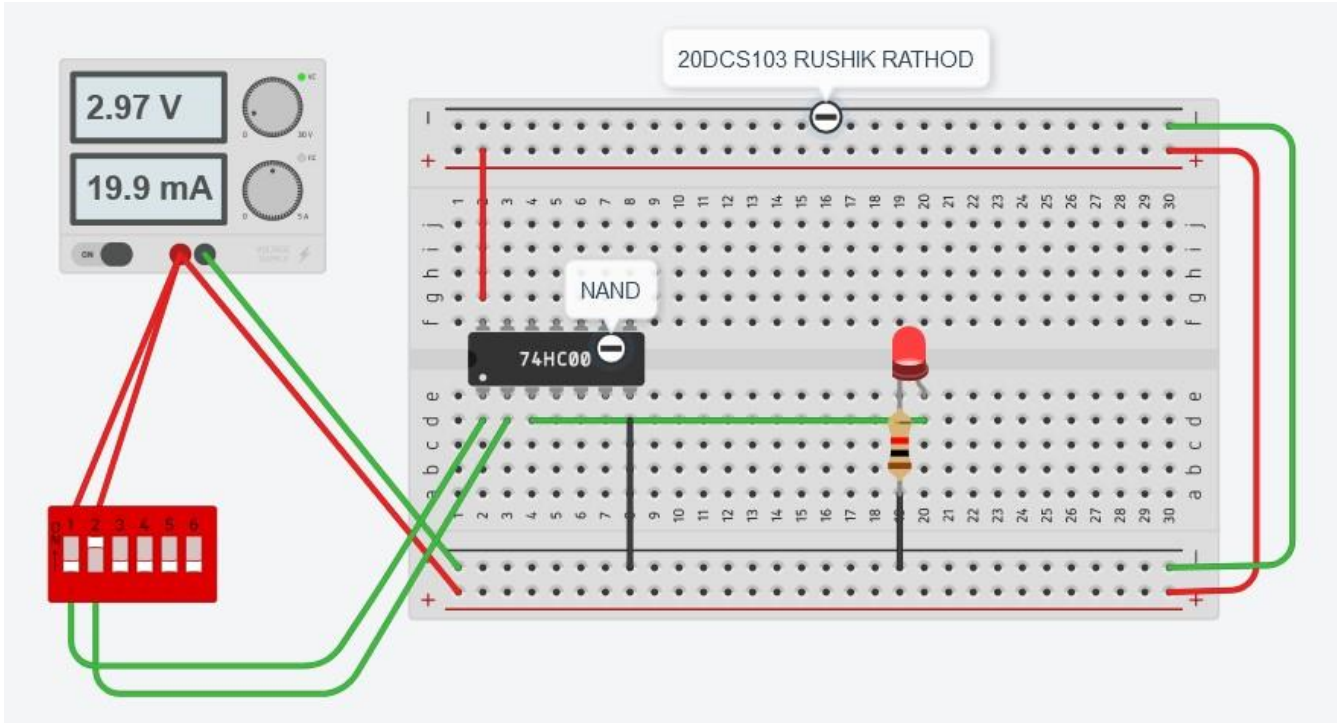


NAND Gate:

The NAND gate is a contraction of AND•NOT. The output is high when both inputs are low and any one of the input is low. The output is low level when the input are high.

Symbol:

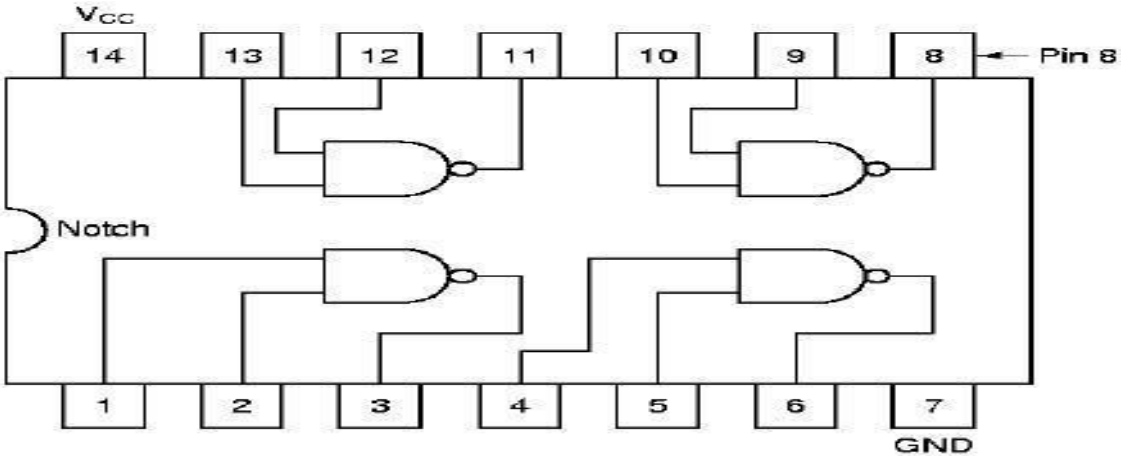




Observation Table:

Input	Input	Output
0	0	1
0	1	1
1	0	1
1	1	0

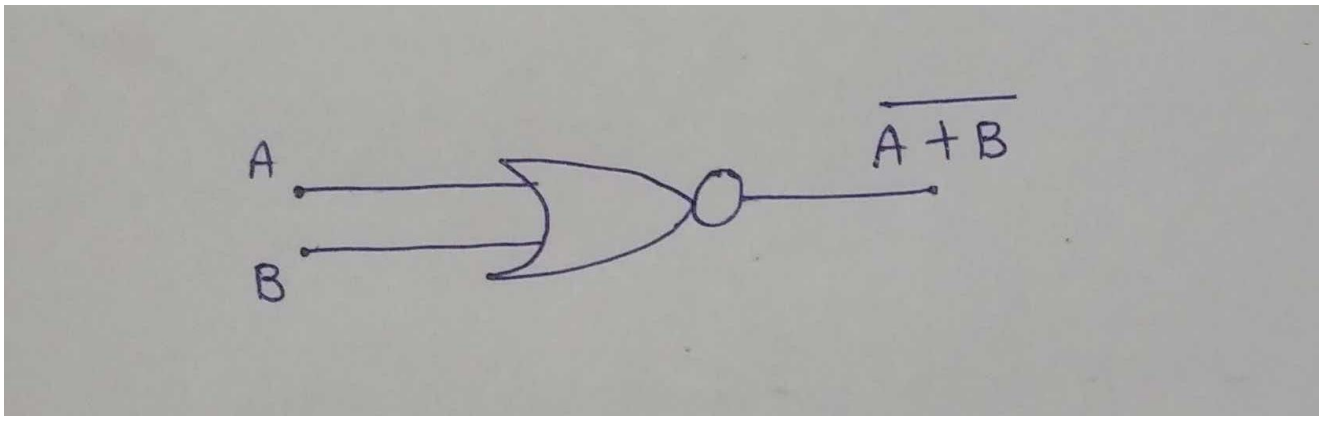
IC Diagram:

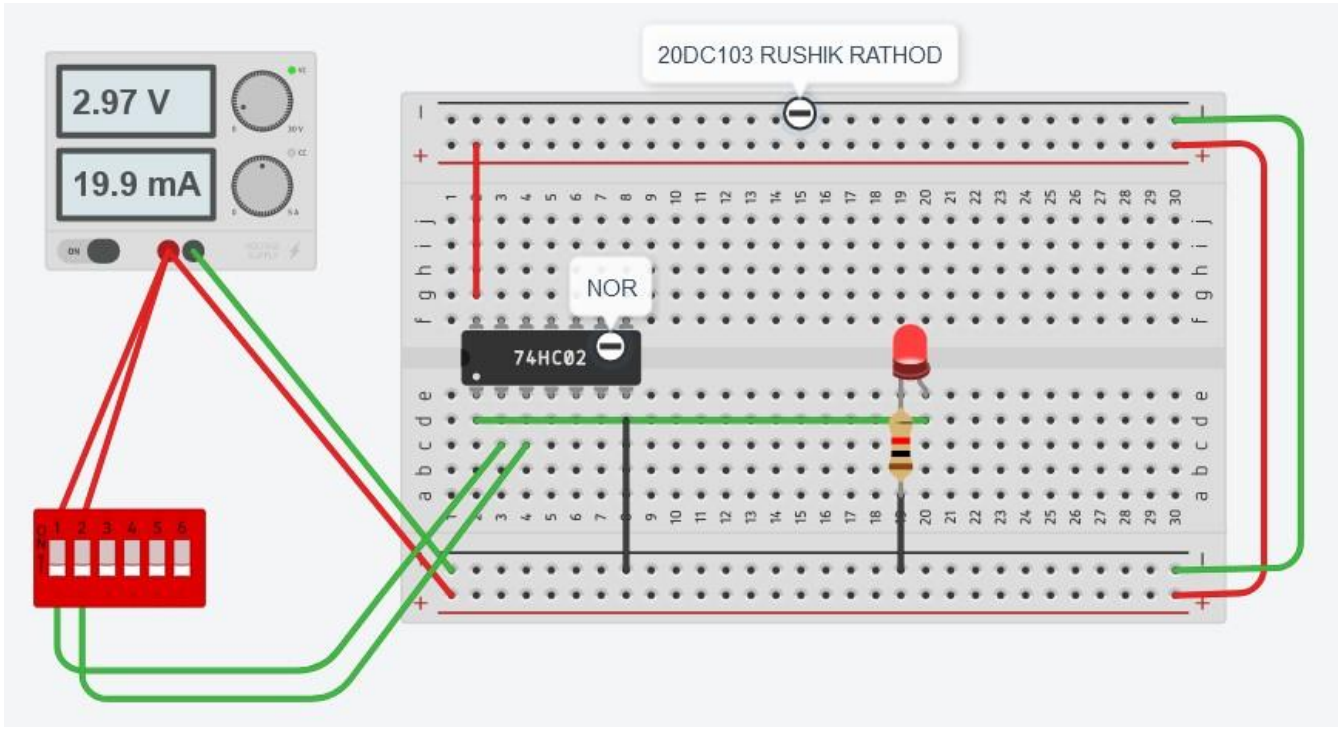


NOR Gate

The NOR gate is contraction of OR•NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

Symbol:

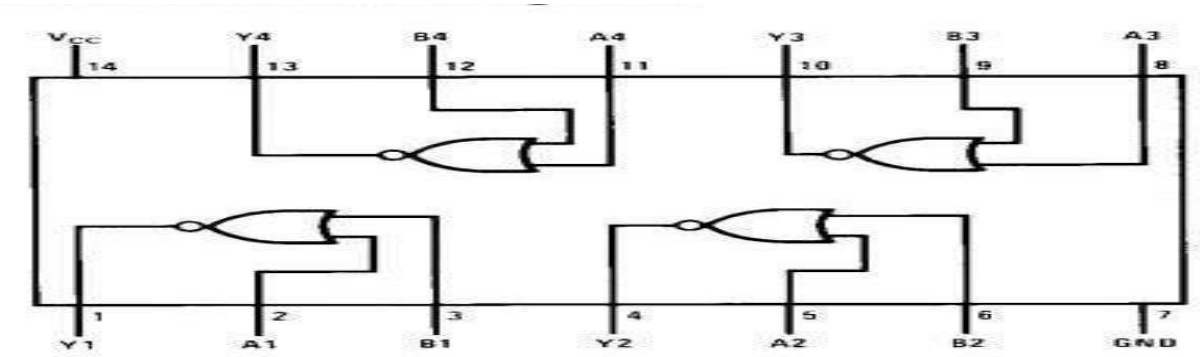




Observation Table:

Input	Input	Output
0	0	1
0	1	0
1	0	0
1	1	0

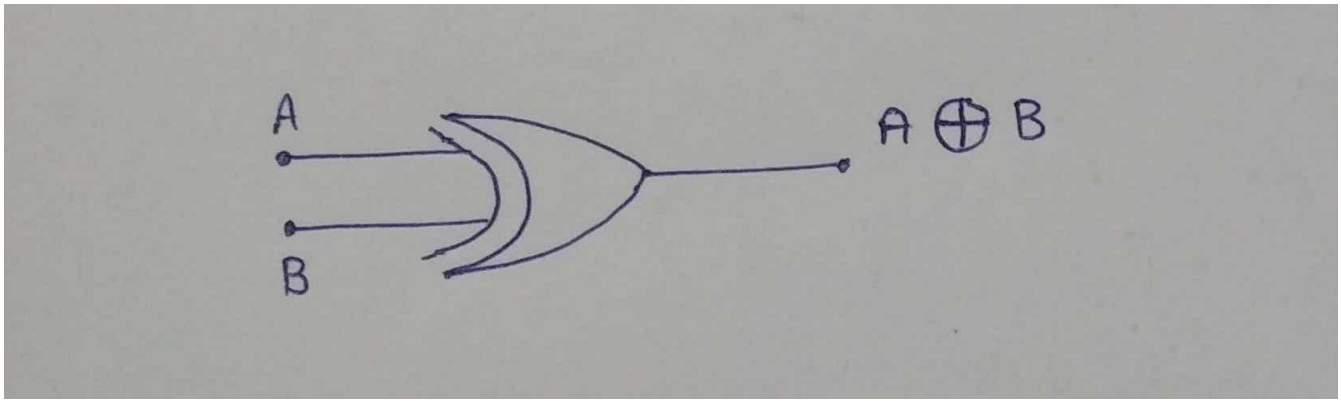
IC Diagram:

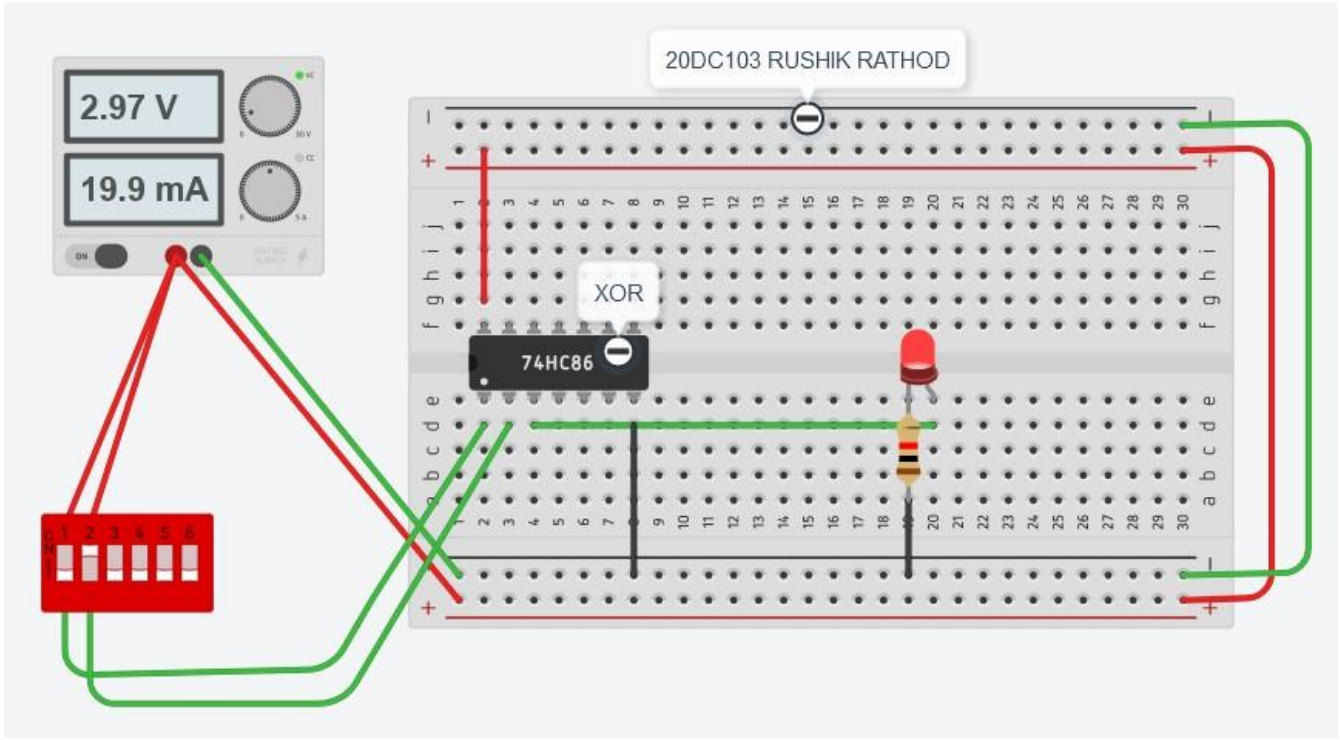


X•OR Gate:

The output is high when any one of the input is high. The output is also low when both the inputs are low and both inputs are high.

Symbol:

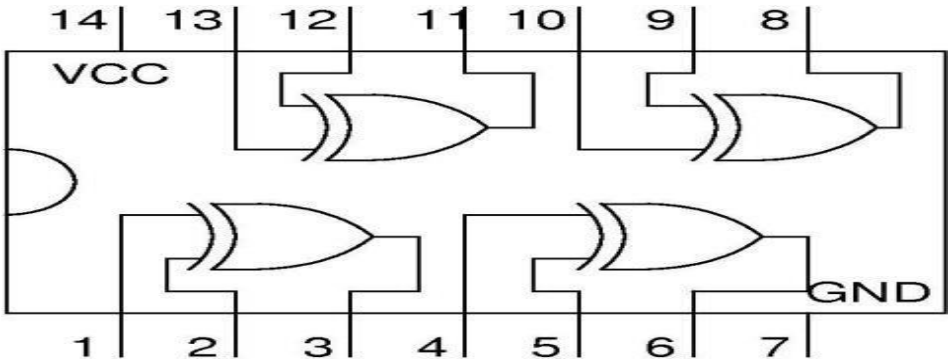




Observation Table:

Input	Input	Output
0	0	0
0	1	1
1	0	1
1	1	0

IC Diagram:



Procedure:

i) Connections are given as per circuit diagram.
ii) Logical inputs are given as per circuit diagram.
iii) Observe the output and verify the truth table.

Conclusion:

From this experiment, I learnt different types of basic logic gates and it's functionalities by implementing them on Tinkercad software and generating truth tables from them.

Experiment No. 2

Aim: Draw a circuit diagram and Verify the truth table of Ex•OR & Ex•NOR gates.

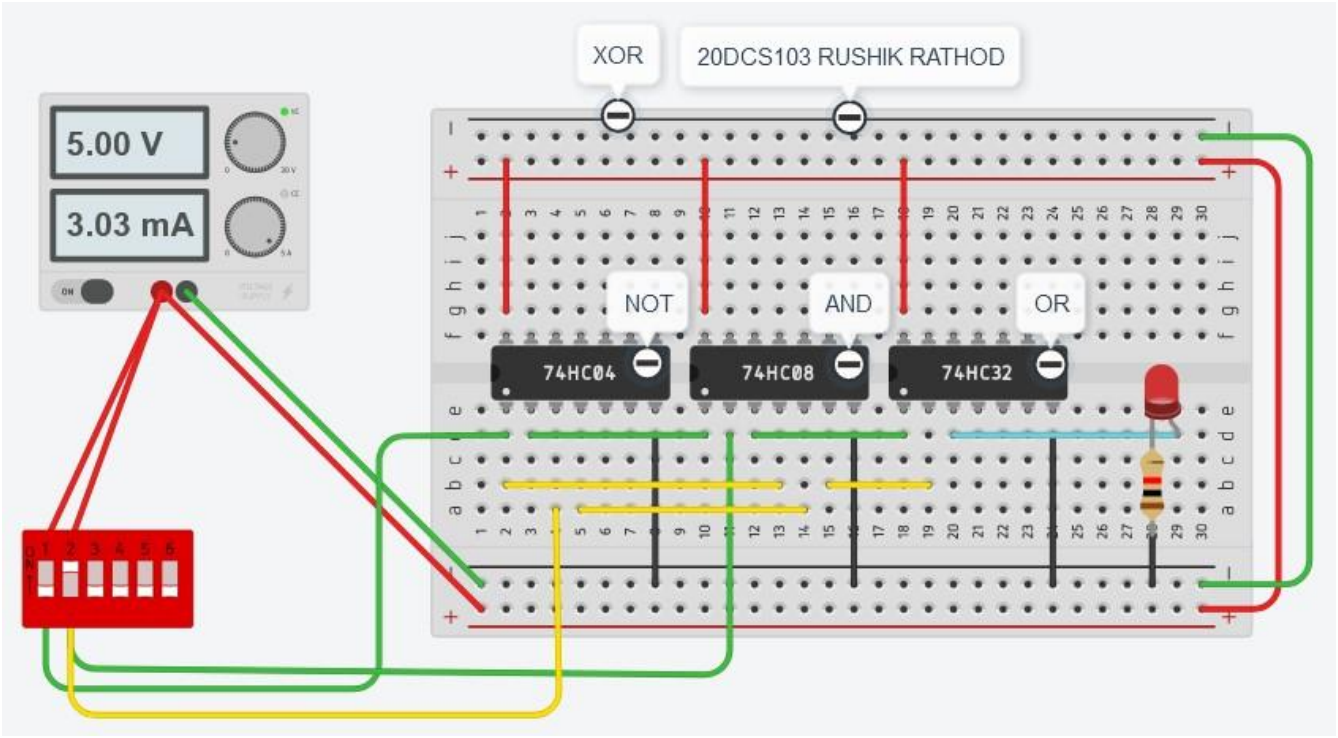
Apparatus: connection wires, power supply, power project board, LED, ICs

Theory:

The Ex•OR gate is a two input, one output logic circuit whose output assumes a logic 1 state when one and only one of its two inputs assumes a logic 1 state. Under the conditions when both the inputs assume the logic 0 state, or when both the inputs assume the logic 1 state, the output assumes a logic 0 state. Since an Ex•OR gate produces an output 1 only when the inputs are not equal, it is called an anti•coincidence gate or inequality detector.

$$A \oplus B = A'B + B'A$$

2 Input XOR gate using AOI logic:



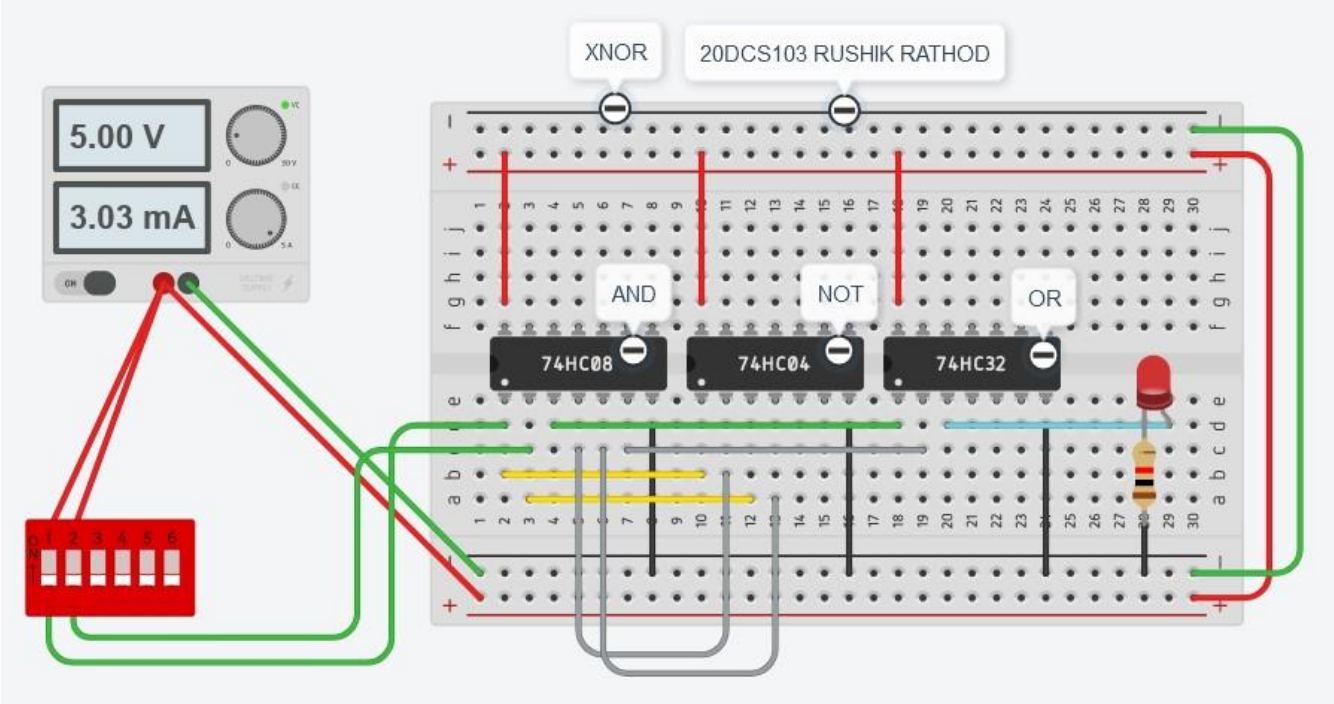
Observation Table:

Input	Input	Output
0	0	0
0	1	1
1	0	1
1	1	0

The Ex•NOR gate is a two input, one output logic circuit whose output assumes a logic 0 state or when both the inputs assume a logic 1 state. The output assumes a logic 0 state, when one of the inputs assumes a 0 state and the other a 1 state. It is also known as a coincidence gate or equality detector.

$$A \odot B = AB + A'B'$$

2 Input Ex•NOR gate using AOI logic:



Observation Table:

Input	Input	Output
0	0	1
0	1	0
1	0	0
1	1	1

Procedure:

1) Connect the circuit according to circuit diagram.
2) Apply different input combination at the input pin of ICs.
3) Verify the truth table of Ex•OR and Ex•NOR gate for different input combinations.

Conclusion:

From this experiment, I learnt to implement the Ex-OR and Ex-NOR gates on Tinkercad software and also verified the truth tables for both of them.

Experiment No. 3

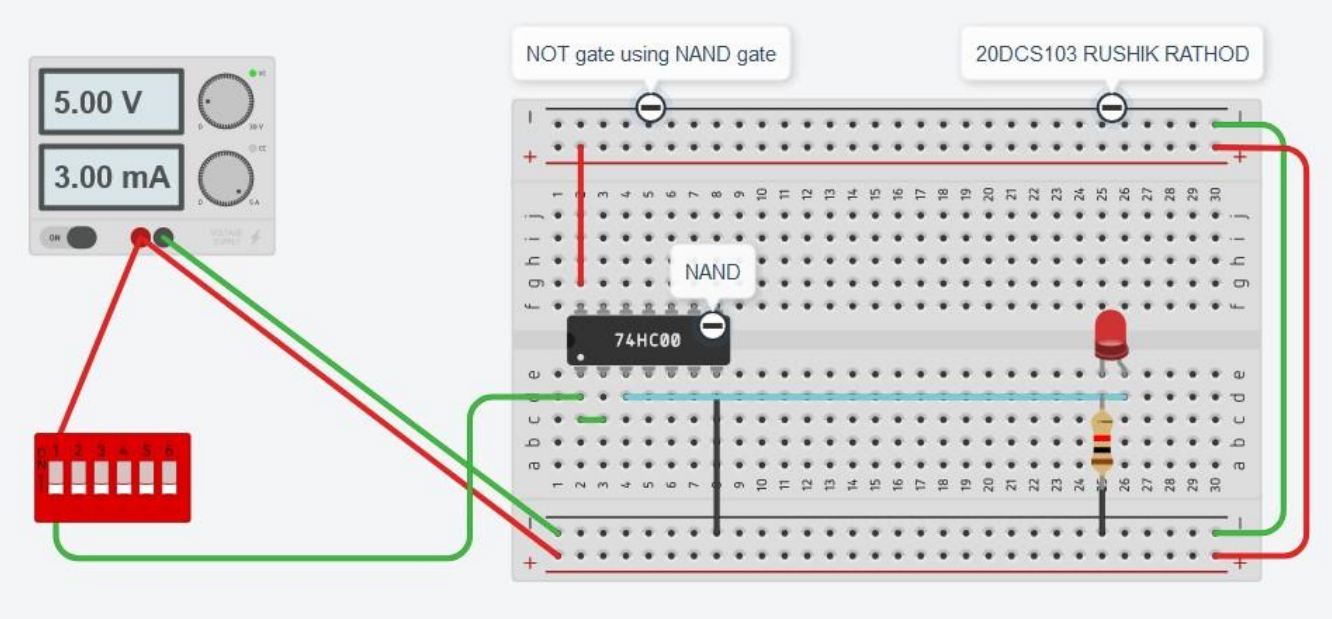
Aim: Verify the operation of NAND & NOR gate as universal gates.

Apparatus: connection wires, power supply, power project board, LED, ICs

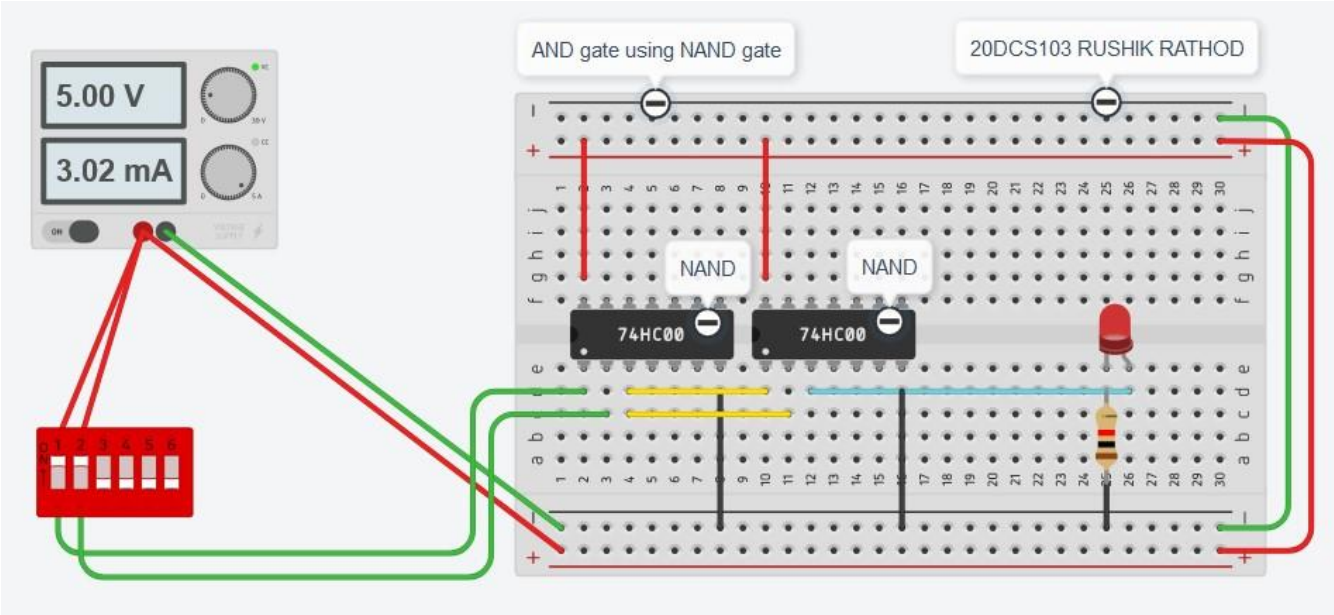
Theory:

The AND, OR and NOT gates are the basic building blocks of digital system. These gates are known as basic gates. Any digital circuit of any complexity can be built using only these three gates. A universal gate is a gate which alone can be used to build any logic circuit. As the basic gates can be realized using only NAND gates or using only NAND gates or using only NOR gates. So NAND gate and NOR gate are also known as Universal gates. The following design diagrams are shown the realization of AND, OR and NOT function using either only NAND gates or only NOR gates.

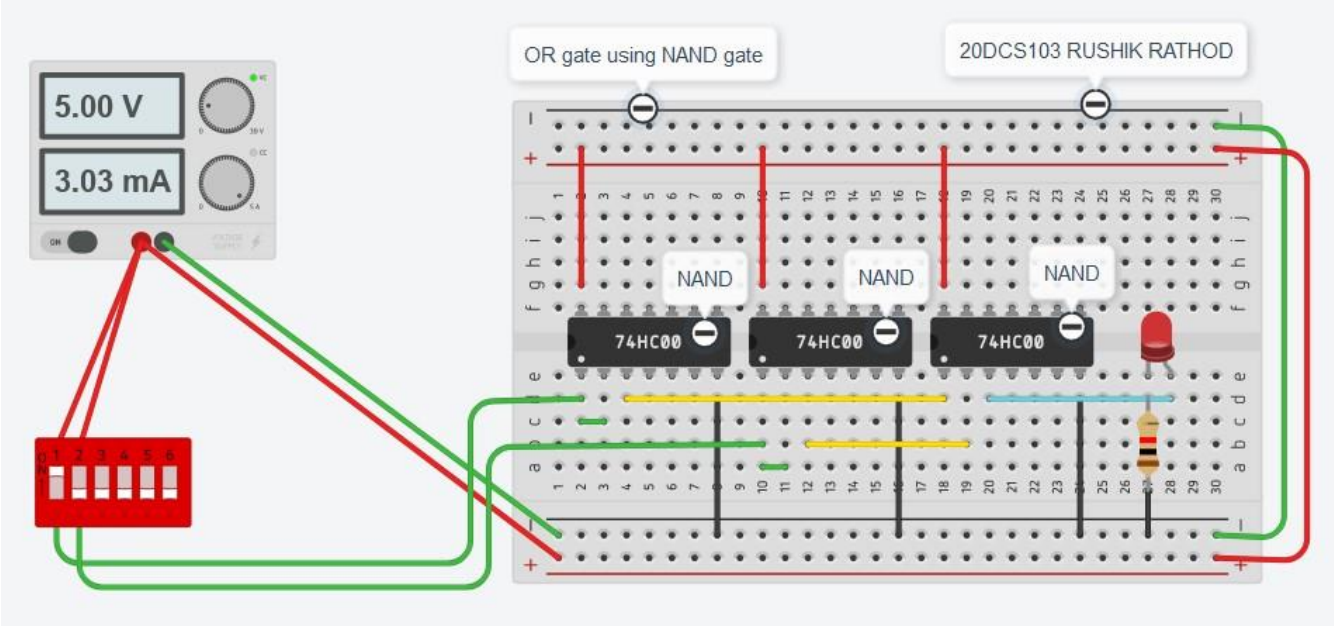
NAND gate as NOT gate:



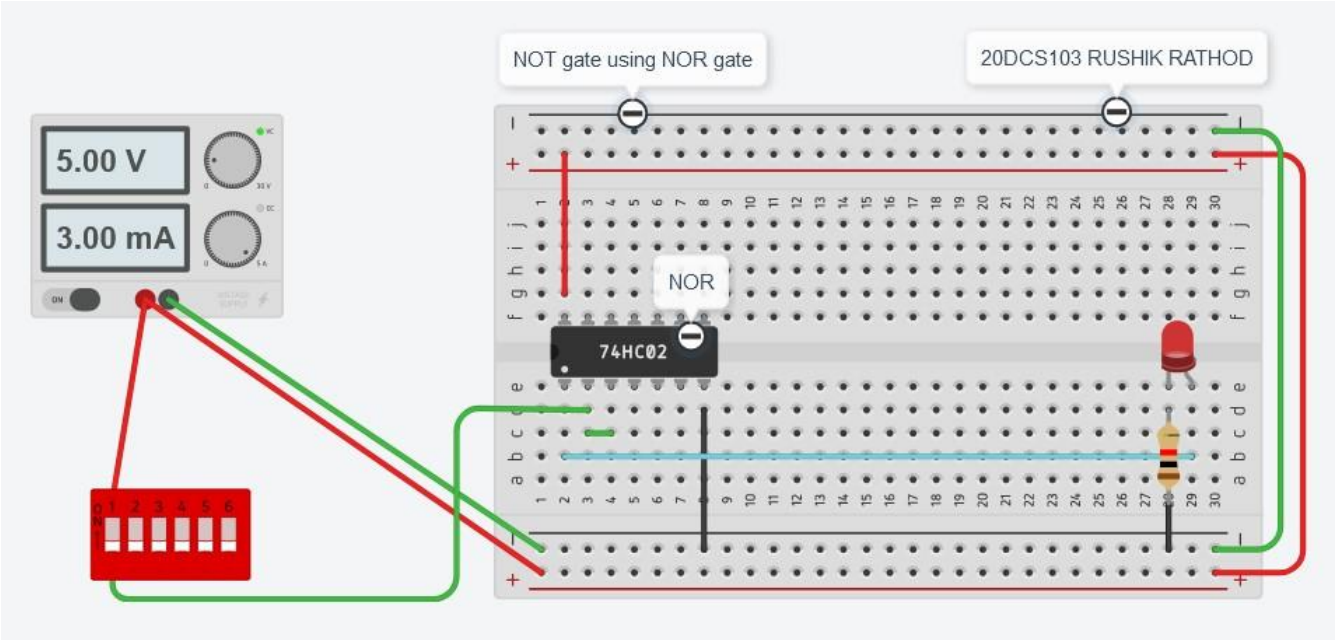
NAND gate as AND gate:



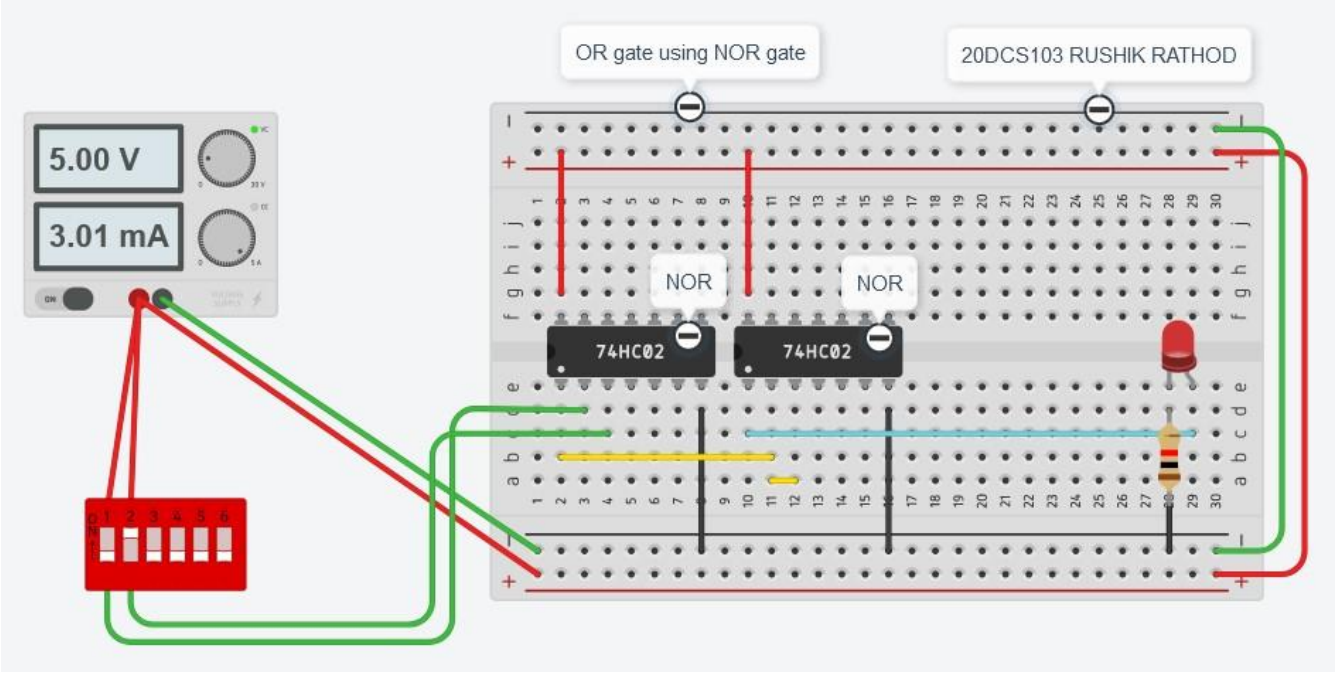
NAND gate as OR gate:



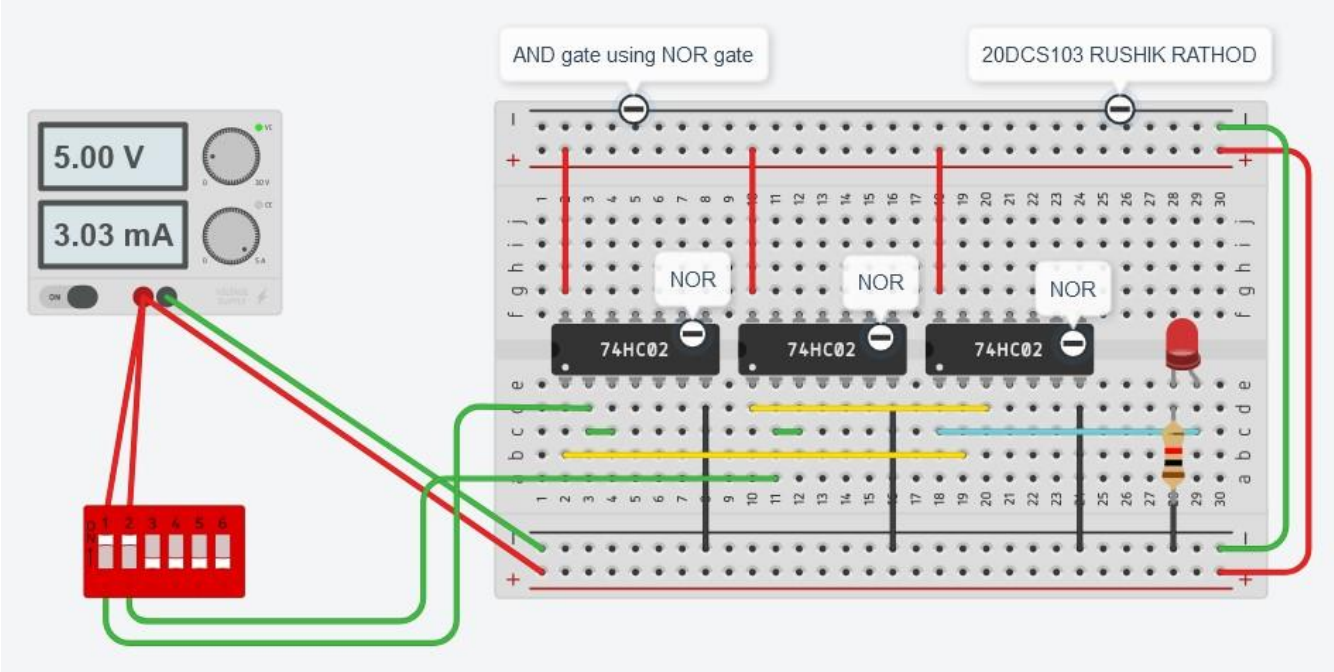
NOR gate as NOT gate:



NOR gate as OR gate:



NOR gate as AND gate:



Procedure:

1) Mount ICs 7400, 7402 on the power project board.
2) Connect pin number 7 and 14 of all ICs to ground and +5V supply respectively.
3) Make the connection as shown in the logic diagram.
4) Verify the truth table of all the data.

Conclusion:

From this experiment, I learnt to create NOT, AND, OR gate using the NAND and NOR gate on Tinkercad software.

Experiment No. 4

Aim: Study of 4•variable K•map and verify using example.

Apparatus: Connection wires, power supply, power project board.

Theory:

A map method provides a simple straight procedure for minimizing boolean functions. This method may be regulated either as a pictorial form of a truth or as an extension of the venn•diagram.

The map method, first propose by veitch and slightly modified by Karnaugh. So it also known as the “Veitch diagram” or the “Karnaugh map”.

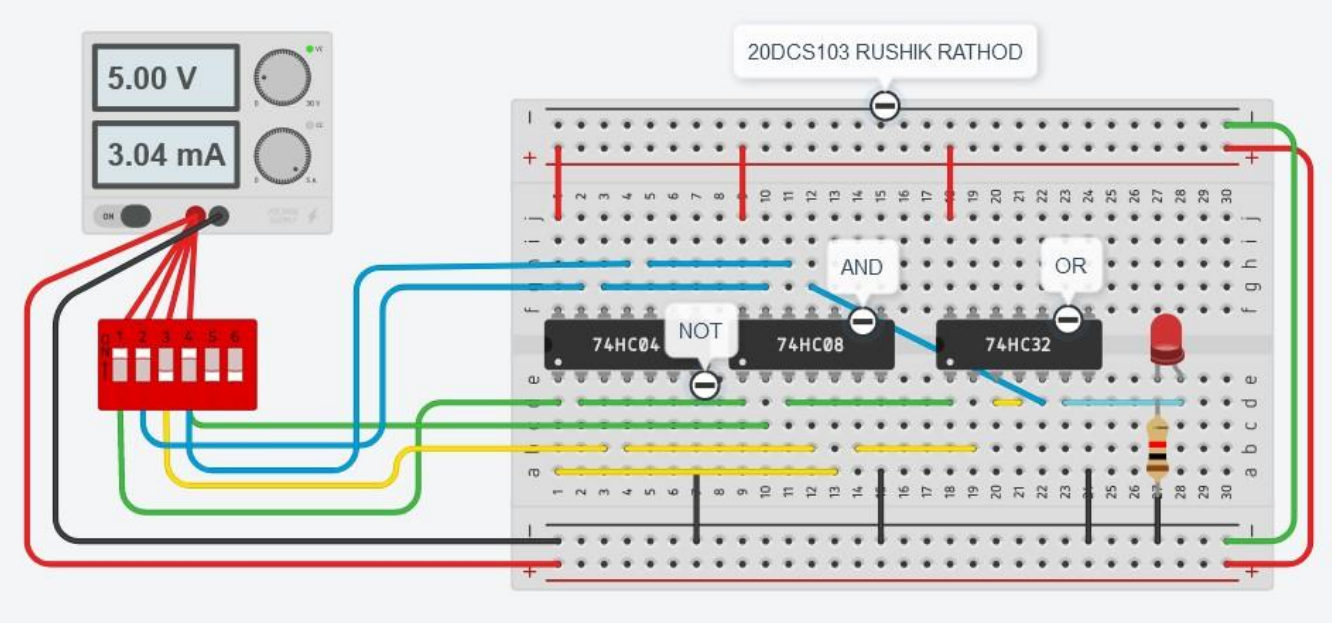
The map is a diagram made up of squares. Each square represents one min•term. Since any Boolean function can be expressed as a sum of min terms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose min•terms are included in the function.

Procedure:

i) Solve the problem given by lab instructor with help of basic logic Gates.
ii) Find out the proper input and output.
iii) Use pin diagram and make proper connection as per requirement.
iv) Measure the various output in LED on/off condition with different input condition. Note down the observation table.
iii) Compare truth table with observation table and write conclusion.

Example:

F = A’D + AC’ + B’D’



Conclusion:

In this experiment, I have verified the 4 variables K-map by implementing it on Tinkercad software.

Experiment No. 5

Aim: Implement half adder and half subtractor circuit.

Apparatus: Connecting wires, power project board, LED, power supply
IC [7408•AND, 7404•NOT, 7402•OR]

Theory:

Logic Circuits for digital system may be combinational or sequential. The output of a combinational circuit depends on its present inputs only. Combinational circuit perform a specific information processing operation fully specified logically by a set of boolean functions. The example of combinational circuit is Adder, Subtractor, Multiplexer, Demultiplexer, Encoder, Decoder, Magnitude Comparator etc.

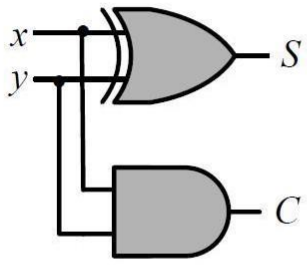
A half adder can add two bits. It has two inputs, generally labelled A and B, and two outputs, the sum S and carry C.

S is the two•bit XOR of A and B, and C is the AND of A and B. Essentially the output of a half adder is the sum of two one•bit numbers, with C being the most significant of these two outputs.

A half adder is a logical circuit that performs an addition operation on two one•bit binary numbers. The half adder outputs a sum of the two inputs and a carry value. The drawback of this circuit is that in case of a multibit addition, it cannot include a carry.

A half subtractor is a logical circuit that performs a subtractor operation on two one bit binary numbers. The half subtractor outputs a difference of the two inputs and a borrow value. In half subtractor circuit difference is the XOR operation of two inputs and borrow is the AND operation of A'B.

Half-Adder



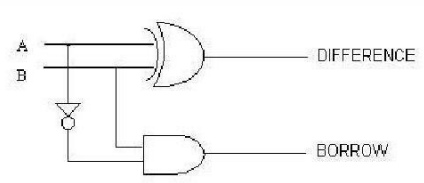
truth table

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = x'y + xy' = x \oplus y$

$C = xy$

Half Subtractor



Inputs		Outputs	
Minuend *A*	Subtrahend *B*	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0
A − B		*Di*	*Bo*

Difference = A'B+AB'=A⊕B

Borrow=A'B

Procedure:

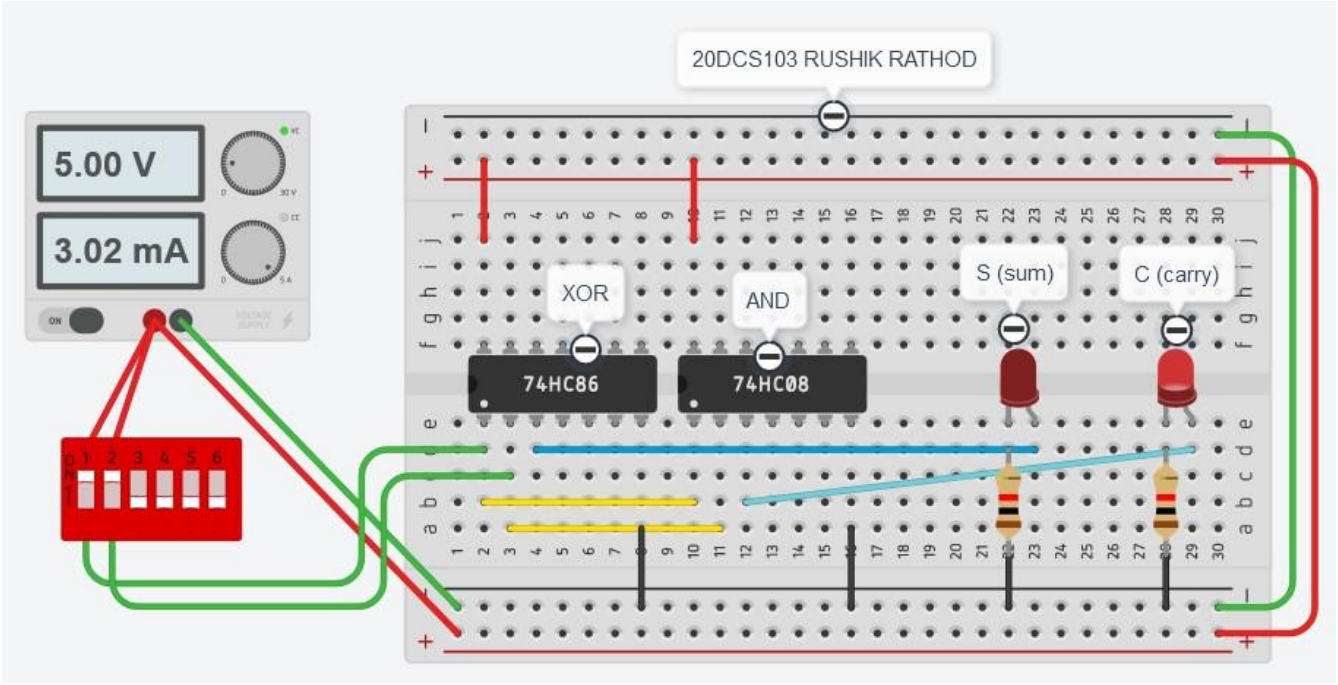
i) Do the connection as per Combinational logic diagram for various input data.
ii) Apply proper input condition and observe the output information of using DMM.
iii) Compare theoretical data with observation and write conclusion.

Observation Table:

Half Adder

Input A	Input B	Sum	Carry (C_{out})
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

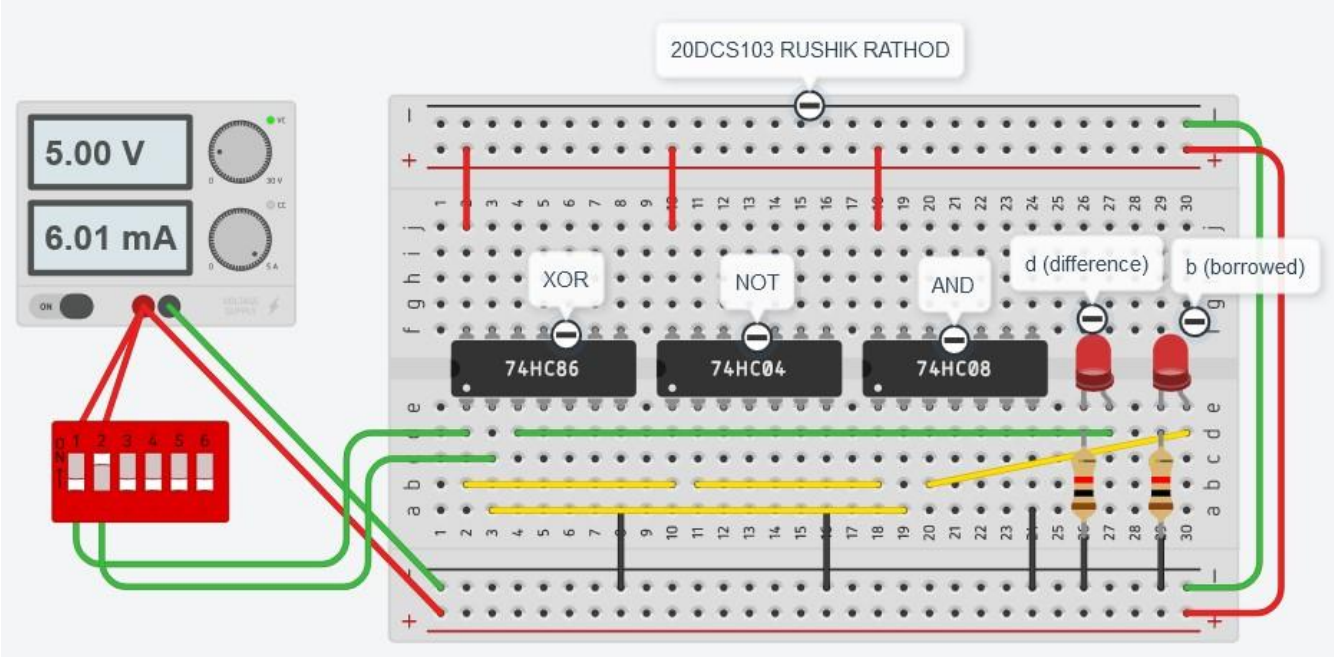
Tinker Cad Implementation:



Half Subtractor

Input A	Input B	Difference	Borrow (Cout)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tinker Cad Implementation:



Conclusion:

From this experiment, I learnt to implement Half adder and Half subtractor using Tinkercad software and verified the truth table for both of them.

Experiment No. 6

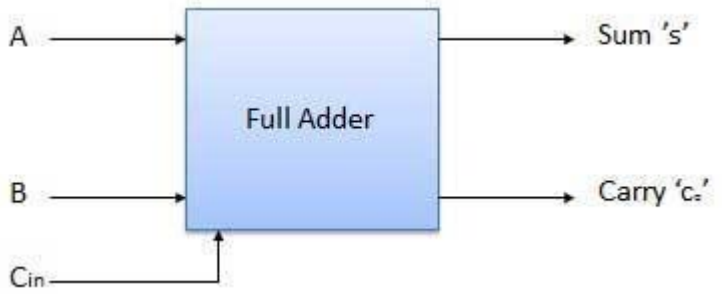
Aim: Implement full adder combinational circuit.

Apparatus: Logic Gate ICs, connecting wires, Bread Board, Power supply, LED, DMM.

Theory:

Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one•bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit.

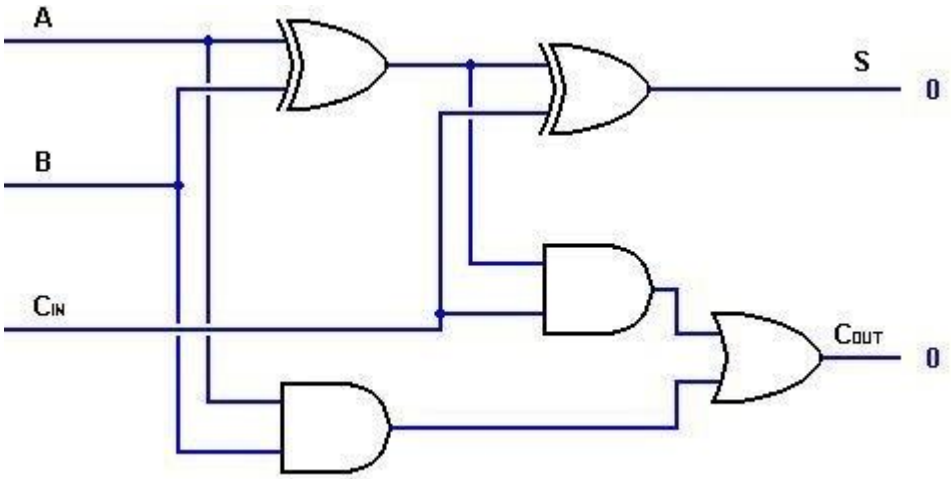
BLOCK DIAGRAM:



TRUTH TABLE:

Inputs			Output	
A	B	Cin	S	Co
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

Combinational Logic:



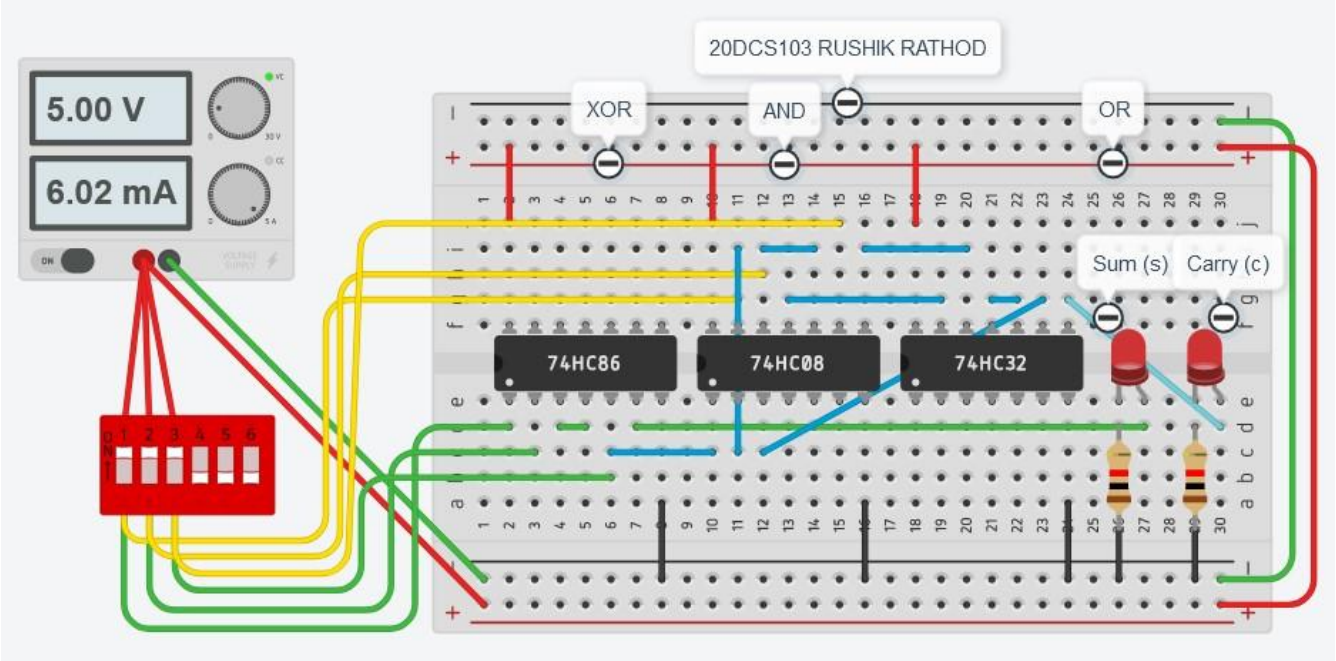
Procedure:

i) Do the connection as per Combinational logic diagram for various input data.
ii) Apply proper input condition and observe the output information of using DMM.
iii) Compare theoretical data with observation and write conclusion.

Observation Table:

Input A	Input B	C_{in}	Sum	Carry(C_{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

Tinker Cad Implementation:



Conclusion:

From this experiment, I learnt to implement the full adder on Tinkercad software and also verified the truth table for the same.

Experiment No. 7

Aim: Show the conversion from BCD to Excess•3.

Apparatus: Logic Gate ICs, connecting wires, Bread Board, Power supply, LED, DMM.

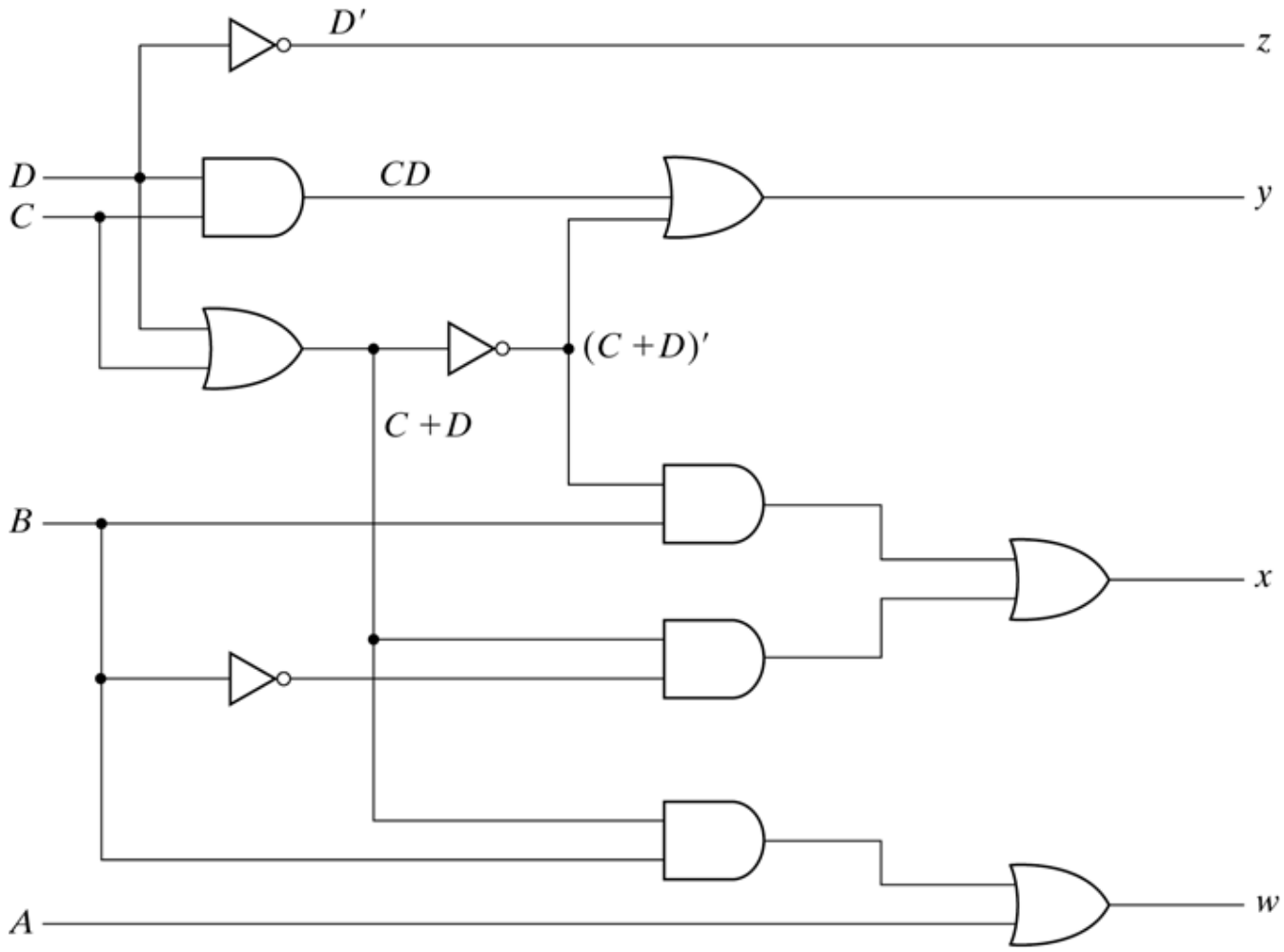
Theory:

We will complete this experiment to code converters by designing an Excess•3 Binary Coded Decimal (BCD) circuit. The term BCD refers to representing the ten decimal digits in binary forms; which simply means to count in binary; see Table below. The Excess•3 system simply adds 3 to each number to make the codes look different. We will not venture to discuss the importance of the Excess•3 BCD system because the discussion would serve too great a distraction from our present purpose and the cost would outweigh the benefit. Suffice it to say that the Excess•3 BCD system has some properties that made it useful in early computers.

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

The Excess•3 BCD system is formed by adding 0011 to each BCD value as in Table. For example, the decimal number 7, which is coded as 0111 in BCD, is coded as 0111+0011=1010 in Excess•3 BCD

BCD to Excess•3 Code conversion logic diagram



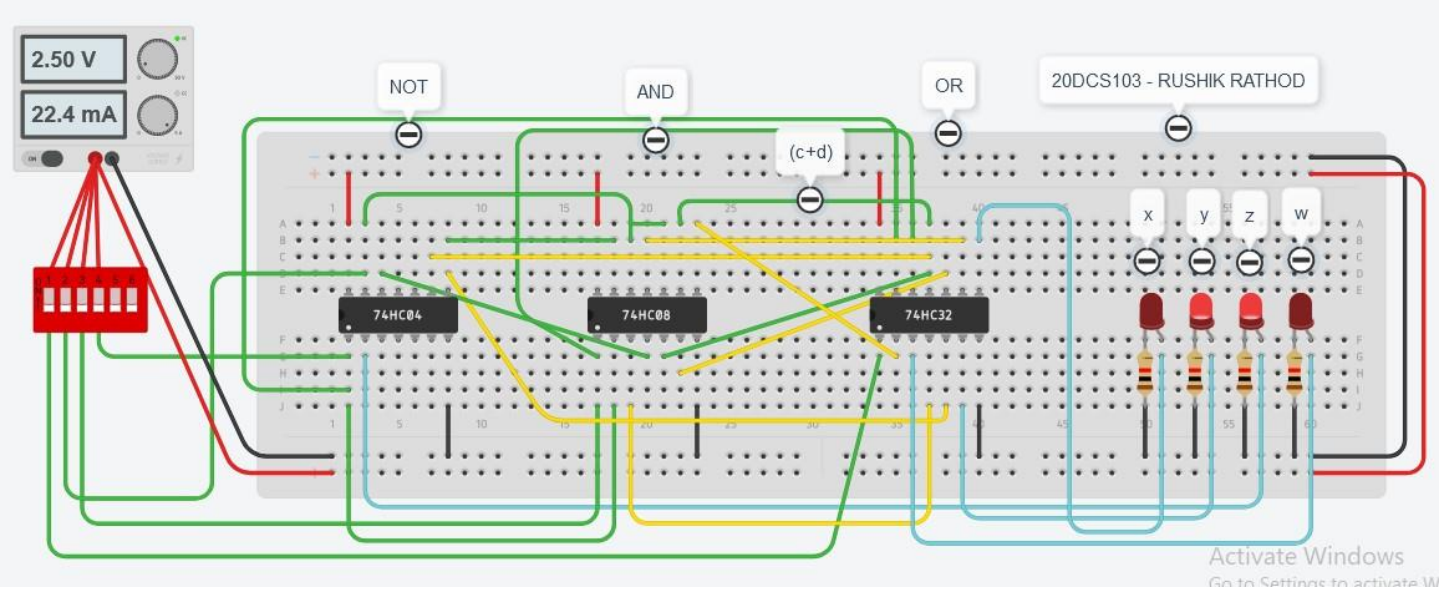
Procedure:

i) Do the connection as per Combinational logic diagram for various input data.
ii) Apply proper input condition and observe the output information of using DMM.
iii) Compare theoretical data with observation and write conclusion.

Observation Table:

	Input (BCD)				Output (Excess – 3)			
Decimal	**A**	**B**	**C**	**D**	**W**	**X**	**Y**	**Z**
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Tinker Cad Implementation:



Conclusion:

From this practical, I learnt the conversion from BCD to Excess 3 and also implemented it in Tinker Cad software.

Experiment No. 8

Aim: Study 4•bit magnitude comparator and implement it.

Apparatus: 4 bit magnitude comparator IC (74LS85), Connecting wires, Bread Board, Power supply, LED, DMM.

Theory:

Definition

A magnitude comparator is a combinational circuit that compares two numbers A & B to determine whether:

- A > B, or
- A = B, or
- A < B

Inputs

First *n*-bit number A

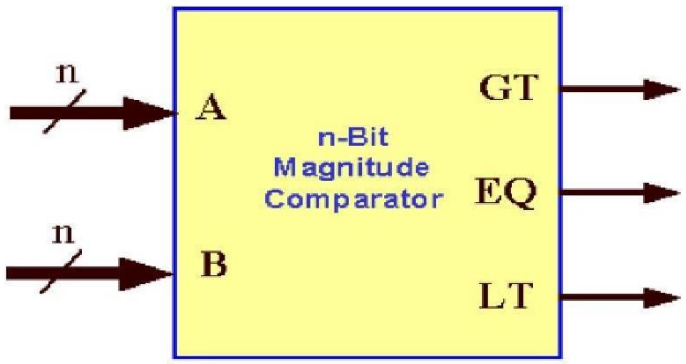
Second *n*-bit number B

Outputs

3 output signals (GT, EQ, LT), where:

1. GT = 1 IFF A > B
2. EQ = 1 IFF A = B
3. LT = 1 IFF A < B

Note: *Exactly One* of these 3 outputs equals 1, while the other 2 outputs are 0`s



4-bit magnitude comparator

Inputs: 8-bits (A ⇒ 4-bits , B ⇒ 4-bits)

A and B are two 4-bit numbers

- Let A = $A_3A_2A_1A_0$, and
- Let B = $B_3B_2B_1B_0$
- Inputs have 2^8 (*256*) possible combinations
- Not easy to design using conventional techniques

Design of the EQ output (A = B) in 4-bit magnitude comparator

Define $X_i = (A_i\, B_i) + (A_i'\, B_i')$

Thus $X_i = 1$ IFF $A_i = B_i$ $\quad \forall\, i = 0, 1, 2$ and 3
$X_i = 0$ IFF $A_i \neq B_i$

Condition for A= B
EQ=1 (i.e., A=B) IFF

1. A_3=B_3 → (X_3 = 1), and
2. A_2=B_2 → (X_2 = 1), and
3. A_1=B_1 → (X_1 = 1), and
4. A_0=B_0 → (X_0 = 1).

Thus, EQ=1 IFF $X_3\, X_2\, X_1\, X_0 = 1$. In other words, EQ = $X_3\, X_2\, X_1\, X_0$

Design of the GT output (A > B) 4-bit magnitude comparator

If $A_3 > B_3$, then A > B (GT=1) irrespective of the relative values of the other bits of A & B. Consider, for example, A = 1000 and B = 0111 where A > B.
This can be stated as GT=1 if $\mathbf{A_3\, B_3'}$ =1

If $A_3 = B_3$ ($X_3 = 1$), we compare the next significant pair of bits (A_2 & B_2).

If $A_2 > B_2$ then A > B (GT=1) irrespective of the relative values of the other bits of A & B. Consider, for example, A = 0100 and B = 0011 where A > B.
This can be stated as GT=1 if $\mathbf{X_3 A_2\, B_2'}$ =1

If $A_3 = B_3$ ($X_3 = 1$) and $A_2 = B_2$ ($X_2 = 1$), we compare the next significant pair of bits (A_1 & B_1).

If $A_1 > B_1$ then A > B (GT=1) irrespective of the relative values of the remaining bits A_0 & B_0. Consider, for example, A = 0010 and B = 0001 where A > B
This can be stated as GT=1 if $\mathbf{X_3\, X_2 A_1\, B_1'}$ =1

If $A_3 = B_3$ ($X_3 = 1$) and $A_2 = B_2$ ($X_2 = 1$) and $A_1 = B_1$ ($X_1 = 1$), we compare the next pair of bits (A_0 & B_0).
If $A_0 > B_0$ then A > B (GT=1). This can be stated as GT=1 if $\mathbf{X_3 X_2 X_1 A_0 B_0'}$=1

To summarize, GT =1 (A > B) IFF:

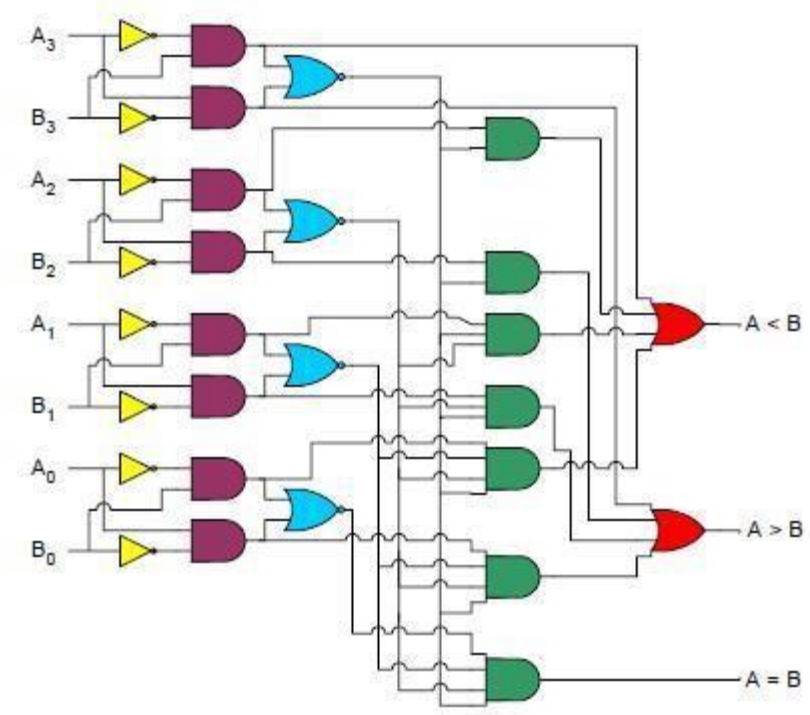
1. $\mathbf{A_3\, B_3'}$ =1, or
2. $\mathbf{X_3 A_2\, B_2'}$ =1, or
3. $\mathbf{X_3\, X_2 A_1\, B_1'}$ = 1, or
4. $\mathbf{X_3 X_2 X_1 A_0 B_0'}$ =1

In other words, $\mathbf{GT = A_3\, B_3' + X_3 A_2\, B_2' + X_3\, X_2 A_1\, B_1' + X_3 X_2 X_1 A_0 B_0'}$

Design of the LT output (A < B) 4-bit magnitude comparator

In the same manner as above, we can derive the expression of the LT (A < B) output
$\mathbf{LT = B_3\, A_3' + X_3 B_2\, A_2' + X_3\, X_2 B_1\, A_1' + X_3 X_2 X_1 B_0 A_0'}$
The gate implementation of the three output variables (EQ, GT & LT) is shown in the figure below.

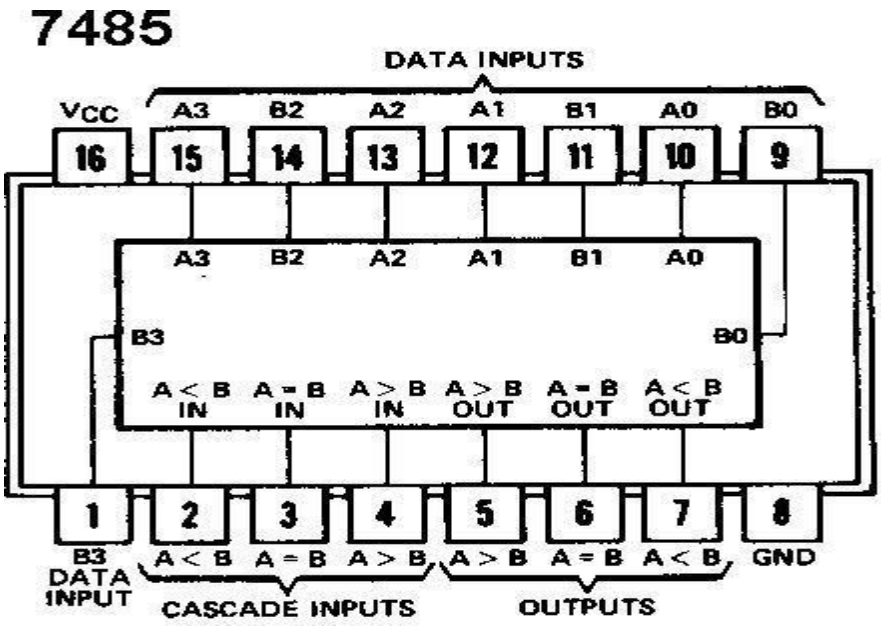
Logic Diagram:



Procedure:

i) Do the connection as per below pin diagram for various input data or 4 bit number.
ii) Apply proper input condition and observe the output information of led on/off.
iii) Compare theoretical data with observation and write conclusion.

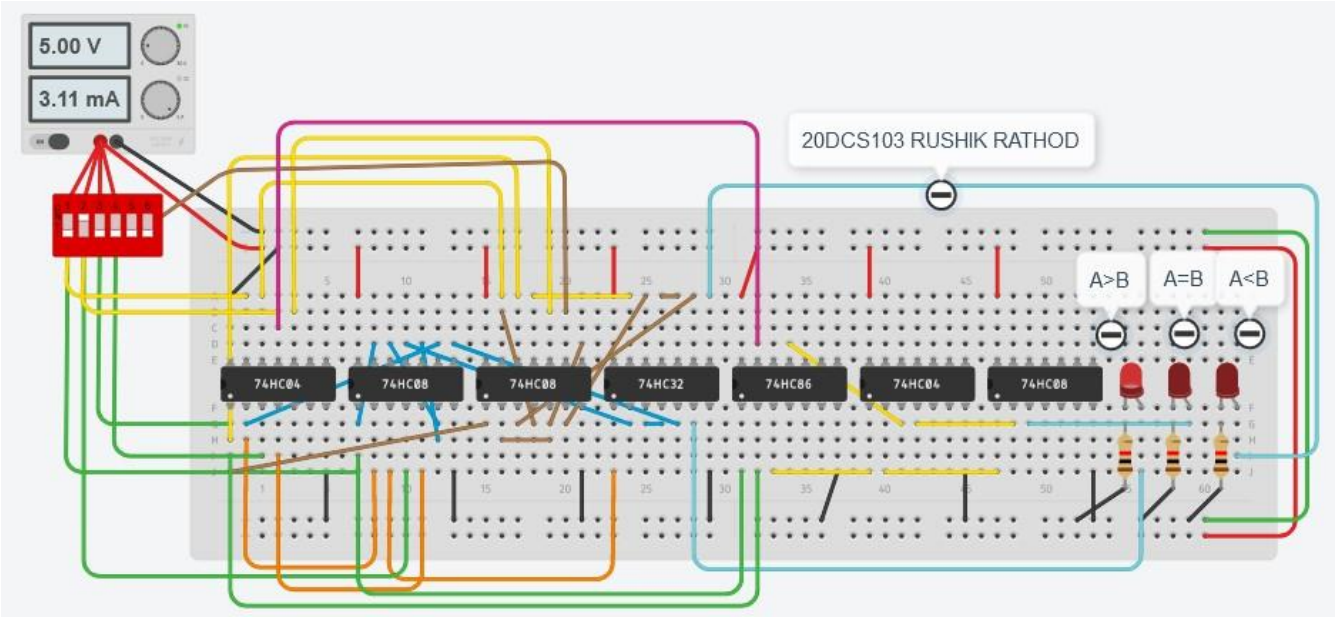
Pin Diagram:



Observation Table:

Input A		Input B		OUTPUT		
A1	**A0**	**B1**	**B0**	**A < B**	**A = B**	**A>B**
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Tinker Cad Implementation:



Conclusion:

From this practical, I learnt the working of 4 bit magnitude comparator and learnt to implement it in Tinkercad.

Experiment No. 9

Aim: Implement 8:1 multiplexer.

Apparatus: Multiplexer IC (74LS151), Connecting wires, Bread Board, Power supply, LED, DMM.

Theory:

In electronics, a multiplexer or mux is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output.

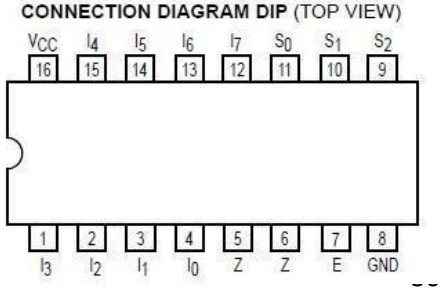
An electronic multiplexer can be considered as a multiple-input, single-output switch i.e. digitally controlled multi-position switch. The digital code applied at the select inputs determines which data inputs will be switched to output.

A common example of multiplexing or sharing occurs when several peripheral devices share a single transmission line or bus to communicate with computer. Each device in succession is allocated a brief time to send and receive data. At any given time, one and only one device is using the line. This is an example of time multiplexing since each device is given a specific time interval to use the line.

In frequency multiplexing, several devices share a common line by transmitting at different frequencies.

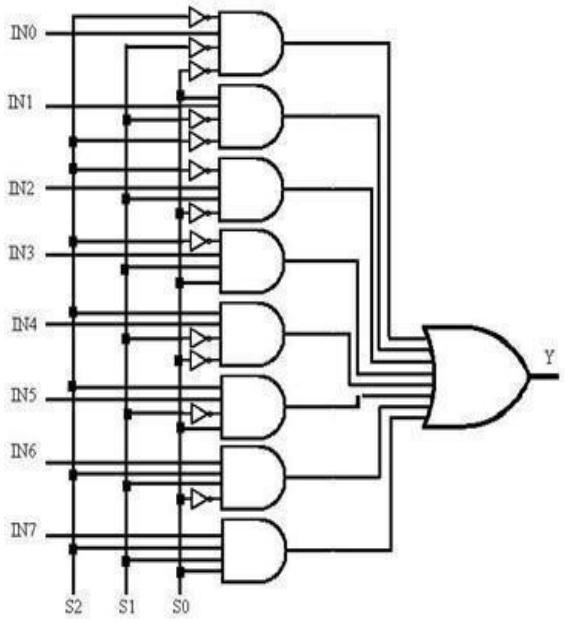
S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Truth Table of 8:1 MUX



PIN NAMES

$\overline{S_0}-\overline{S_2}$	Select Inputs
$\overline{E}$	Enable (Active LOW) Input
I_0-I_7	Multiplexer Inputs
$\overline{Z}$	Multiplexer Output (Note b)
Z	Complementary Multiplexer Output (Note b)



Logic Diagram of 8:1 MUX

Procedure:

i) Solve the problem given by lab instructor with help of 8x1 Multiplexer using implementation table method.
ii) Find out the proper input and output as well as define the select line variable.
iii) Use pin diagram and make proper connection as per requirement.
iv) Measure the various output in LED on/off condition with different input condition. Note down the observation table.
iii) Compare truth table with observation table and write conclusion.

Observation Table:

B	C	D	O/P
0	0	0	A
0	0	1	1
0	1	0	A
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Conclusion: In this Practical we learned 8:1 Multiplexer and how to solve it.

Experiment No.10

Aim: Demonstrate the operation of RS, JK &D flip•flops. Verify truth table, Excitation table and functional table.

Apparatus: Sigma Flip Flop trainer model•DFF11, Connecting wires

Theory:

Flip flops are actually an application of logic gates. With the help of Boolean logic you can create memory with them. Flip flops can also be considered as the most basic idea of a Random Access Memory [RAM]. When a certain input value is given to them, they will be remembered and executed, if the logic gates are designed correctly. A higher application of flip flops is helpful in designing better electronic circuits.

The most commonly used application of flip flops is in the implementation of a feedback circuit. As a memory relies on the feedback concept, flip flops can be used to design it.

There are mainly four types of flip flops that are used in electronic circuits. They are

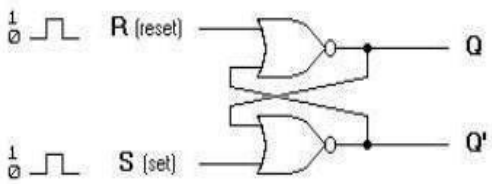
1. The basic Flip Flop or S•R Flip Flop
2. Delay Flip Flop [D Flip Flop]
3. J•K Flip Flop
4. T Flip Flop

1. S•R Flip Flop

The SET•RESET flip flop is designed with the help of two NOR gates and also two NAND gates. These flip flops are also called S•R Latch.

• S•R Flip Flop using NOR Gate

The design of such a flip flop includes two inputs, called the SET [S] and RESET [R]. There are also two outputs, Q and Q'. The diagram and truth table is shown next page.



(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after S=1, R=0)
0	1	0	1	
0	0	0	1	(after S=0, R=1)
1	1	0	0	

(b) Truth table

Basic flip-flop circuit with NOR gates

From the diagram it is evident that the flip flop has mainly four states. They are S=1, R=0—Q=1, Q'=0

This state is also called the SET state.

S=0, R=1—Q=0, Q'=1 This state is known as the RESET state.

In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the value of S.

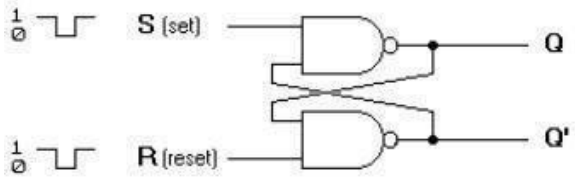
S=0, R=0—Q & Q' = Remember If both the values of S and R are switched to 0, then the circuit remembers the value of S and R in their previous state.

S=1, R=1—Q=0, Q'=0 [Invalid]

This is an invalid state because the values of both Q and Q' are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

- **S•R Flip Flop using NAND Gate**

The circuit of the S•R flip flop using NAND Gate and its truth table is shown below.



(a) Logic diagram

S R	Q Q'	
1 0	0 1	
1 1	0 1	(after S=1, R=0)
0 1	1 0	
1 1	1 0	(after S=0, R=1)
0 0	1 1	

(b) Truth table

Basic flip-flop circuit with NAND gates

S•R Flip Flop using NAND Gate

Like the NOR Gate S•R flip flop, this one also has four states. They are S=1, R=0—Q=0, Q'=1

This state is also called the SET state.

S=0, R=1—Q=1,

Q'=0 This state is known as the RESET state.

In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the compliment value of S.

S=0, R=0—Q=1, & Q' =1 [Invalid]

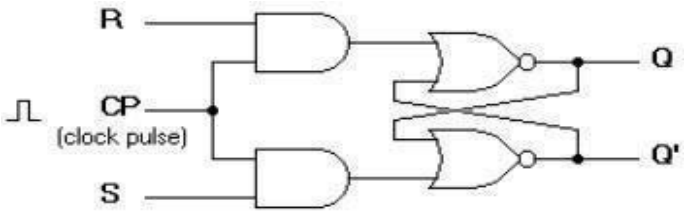
If both the values of S and R are switched to 0 it is an invalid state because the values of both Q and Q' are 1. They are supposed to be compliments of each other. Normally, this state must be avoided.

S=1, R=1—Q & Q'= Remember

If both the values of S and R are switched to 1, then the circuit remembers the value of S and R in their previous state.

- **Clocked S•R Flip Flop**

It is also called a Gated S•R flip flop. The problems with S•R flip flops using NOR and NAND gate is the invalid state. This problem can be overcome by using a bistable SR flip•flop that can change outputs when certain invalid states are met, regardless of the condition of either the Set or the Reset inputs. For this, a clocked S•R flip flop is designed by adding two AND gates to a basic NOR Gate flip flop. The circuit diagram and truth table is shown below.

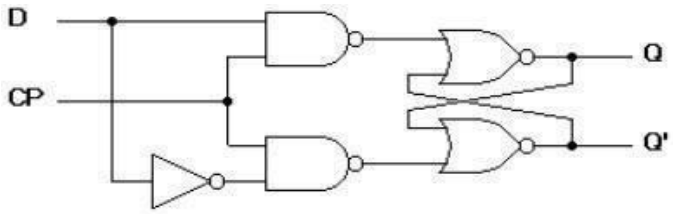


(a) Logic diagram

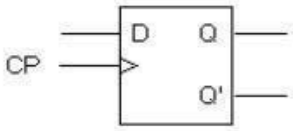
Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

(b) Truth table

Clocked SR flip-flop



(a) Logic diagram with NAND gates



(b) Graphical symbol

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(c) Transition table

Clocked D flip-flop

Clocked S•R Flip Flop

A clock pulse [CP] is given to the inputs of the AND Gate. When the value of the clock pulse is ’0′, the outputs of both the AND Gates remain ’0′. As soon as a pulse is given the value of CP turns ’1′. This makes the values at S and R to pass through the NOR Gate flip flop. But when the values of both S and R values turn ’1′, the HIGH value of CP causes both of them to turn to ’0′ for a short moment. As soon as the pulse is removed, the flip flop state becomes intermediate. Thus either of the two states may be caused, and it depends on whether the set or reset input of the flip•flop remains a ’1′ longer than the transition to ’0′ at the end of the pulse. Thus the invalid states can

be eliminated.

2. D Flip Flop

The circuit diagram and truth table is given below.

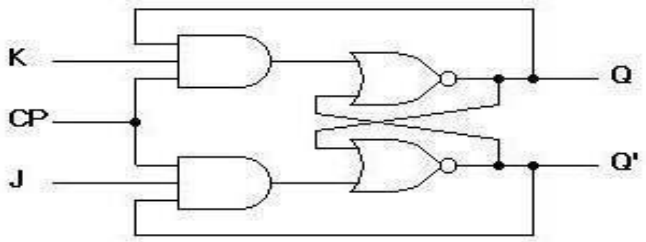
D Flip Flop

D flip flop is actually a slight modification of the above explained clocked SR flip•flop. From the figure you can see that the D input is connected to the S input and the complement of the D input is connected to the R input.

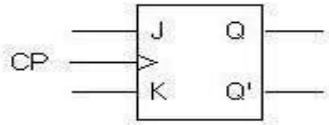
The D input is passed on to the flip flop when the value of CP is ’1′. When CP is HIGH, the flip flop moves to the SET state. If it is ’0′, the flip flop switches to the CLEAR state. To know more about the triggering of flip flop click on the link below.

3. J•K Flip Flop

The circuit diagram and truth•table of a J•K flip flop is shown below.



(a) Logic diagram



(b) Graphical symbol

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(c) Transition table

Clocked JK flip-flop

J•K Flip Flop

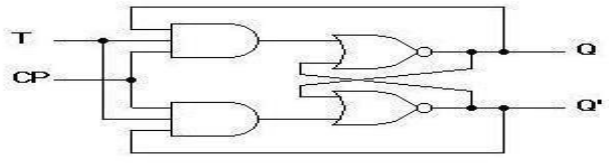
A J•K flip flop can also be defined as a modification of the S•R flip flop. The only difference is that the intermediate state is more refined and precise than that of a S•R flip flop. The behavior of inputs J and K is same as the S and R inputs of the S•R flip flop. The letter J stands for SET and the 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 back 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.

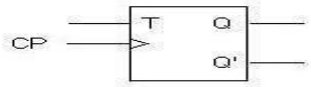
The output may be repeated in transitions once they have been complimented for J=K=1 because of the feedback connection in the JK flip•flop. This can be avoided by setting a time duration lesser than the propagation delay through the flip•flop. The restriction on the pulse width can be eliminated with a master•slave or edge•triggered construction.

4. T Flip Flop

This is a much simpler version of the J•K flip flop. Both the J and K inputs are connected together and thus are also called a single input J•K flip flop. When clock pulse is given to the flip flop, the output begins to toggle. Here also the restriction on the pulse width can be eliminated with a master•slave or edge•triggered construction. Take a look at the circuit and truth table below.



(a) Logic diagram



(b) Graphical symbol

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

(c) Transition table

Clocked T flip-flop

Procedure:

i) Do the connection as per below kit connection diagram for various flip flops.
ii) Apply proper input condition and observe the output information of led on/off.
iii) Compare truth table with observation table and write conclusion.

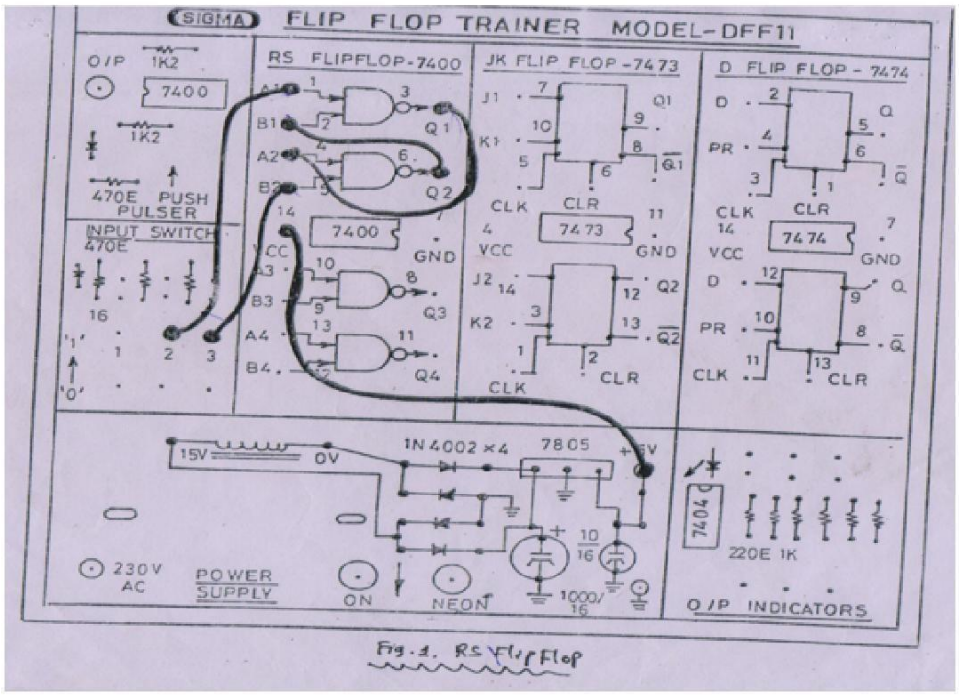


Fig.1. RS Flip Flop

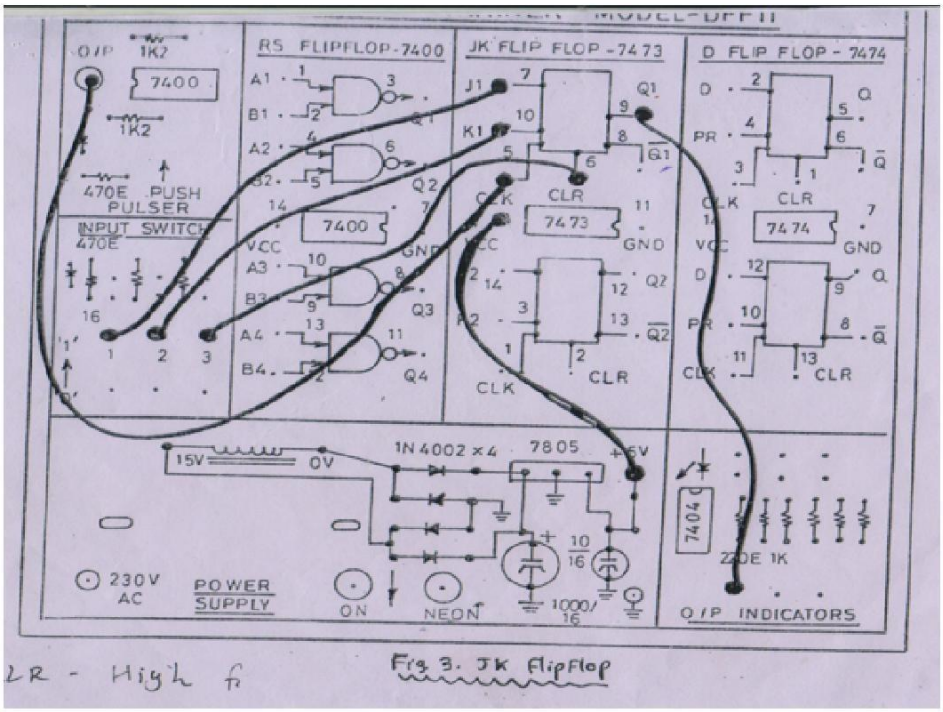


Fig 3. JK FlipFlop

LR - High f

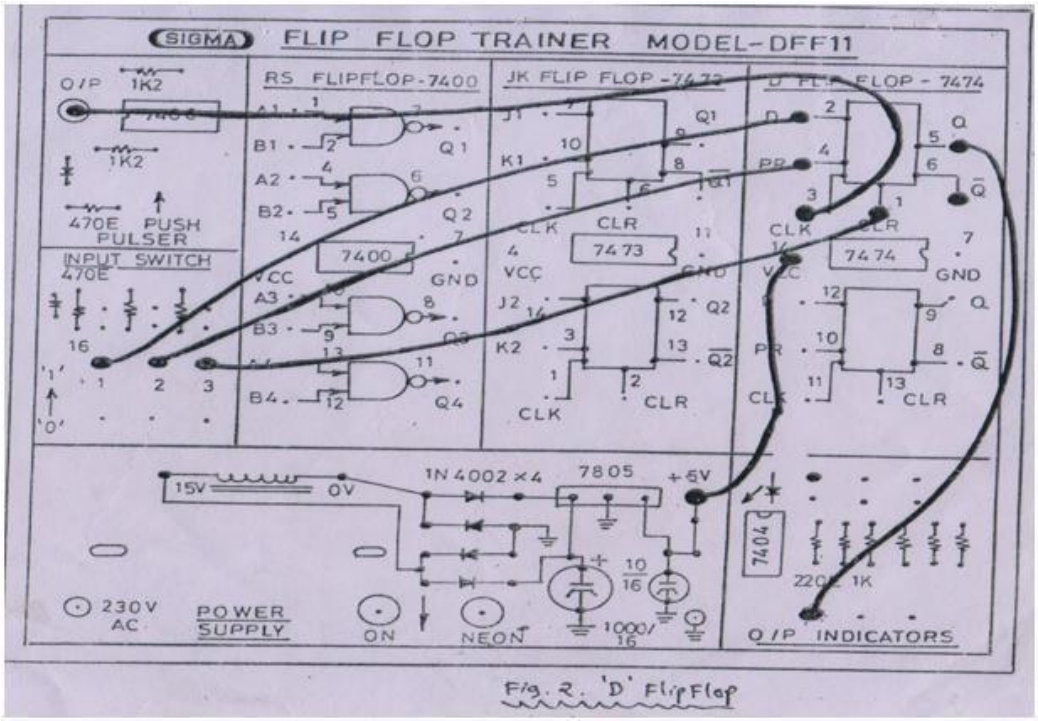


Fig. 2. 'D' Flip Flop

Observation Table:

S•R Flip Flop

S	R	Q	Q'
0	0	0	1
0	1	0	1
1	0	1	0
1	1	0	0

J•K Flip Flop

Q(t)	J	K	Q(t+1)	Q'(t+1)
0	0	0	0	1
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

D Flip Flop

Q(t)	D	Q(t+1)	Q'(t+1)
0	0	0	1
0	1	1	0
1	0	0	1
1	1	1	0

Conclusion: In this practical we learned about different types of Filp Flop and their Truth Table.

Experiment No. 11

Aim: Study and Implement different types of shift register.

Apparatus: Sigma shift registers trainer • DSR08, Connecting wires

Theory:

A universal shift register is an integrated logic circuit that can transfer data in three different modes. Like a parallel register it can load and transmit data in parallel. Like shift registers it can load and transmit data in serial fashions, through left shifts or right shifts. In addition, the universal shift register can combine the capabilities of both parallel and shift registers to accomplish tasks that neither basic type of register can perform on its own. For instance, on a particular job a universal register can load data in series (e.g. through a sequence of left shifts) and then transmit/output data in parallel.

Universal shift registers, as all other types of registers, are used in computers as memory elements. Although other types of memory devices are used for the efficient storage of very large volume of data, from a digital system perspective when we say computer memory we mean registers. In fact, all the operations in a digital system are performed on registers. Examples of such operations include multiplication, division, and data transfer.

In order for the universal shift register to operate in a specific mode, it must first select the mode. To accomplish mode selection the universal register uses a set of two selector switches, S1 and S0. As shown in Table 1, each permutation of the switches corresponds to a loading/input mode.

Operating Mode	S1	S0
Locked	0	0
Shift•Right	0	1
Shift•Left	1	0
Parallel Loading	1	1

Table 1

In the locked mode (S1S0 = 00) the register is not admitting any data; so that the content of the register is not affected by whatever is happening at the inputs. You may verify this detail by playing around with the main interactive circuit. For example, set L3L2L1L0 = 1010 and then cycle the clock to see that nothing changes at the outputs as long as S1S0 = 00. See Table 2.

Clock Cycle	L3	L2	L1	L0		Q3	Q2	Q1	Q0
Initial Value	1	0	1	0		0	0	0	0
Cycle 1	1	0	1	0		0	0	0	0

Table 2

In the shift•right mode (S1S0 = 01) serial inputs are admitted from Q3 to Q0. You can confirm this aspect by setting the value of the shift•right switch according to the sequence 1100100 as you cycle the clock; see Table 3. Watch as the signals move from Q3 to Q0. In the shift•left mode (S1S0 = 10) the register works in a similar fashion, except that the signals move from Q0 to Q3.

Clock Cycle	Shift•Right Switch	Q3	Q2	Q1	Q0
Initial Value	Initial Value	0	0	0	0
Cycle 1	1	1	0	0	0
Cycle 2	1	1	1	0	0
Cycle 3	0	0	1	1	0
Cycle 4	0	0	0	1	1
Cycle 5	1	1	0	0	1
Cycle 6	0	0	1	0	0
Cycle 7	0	0	0	1	0

Table 3

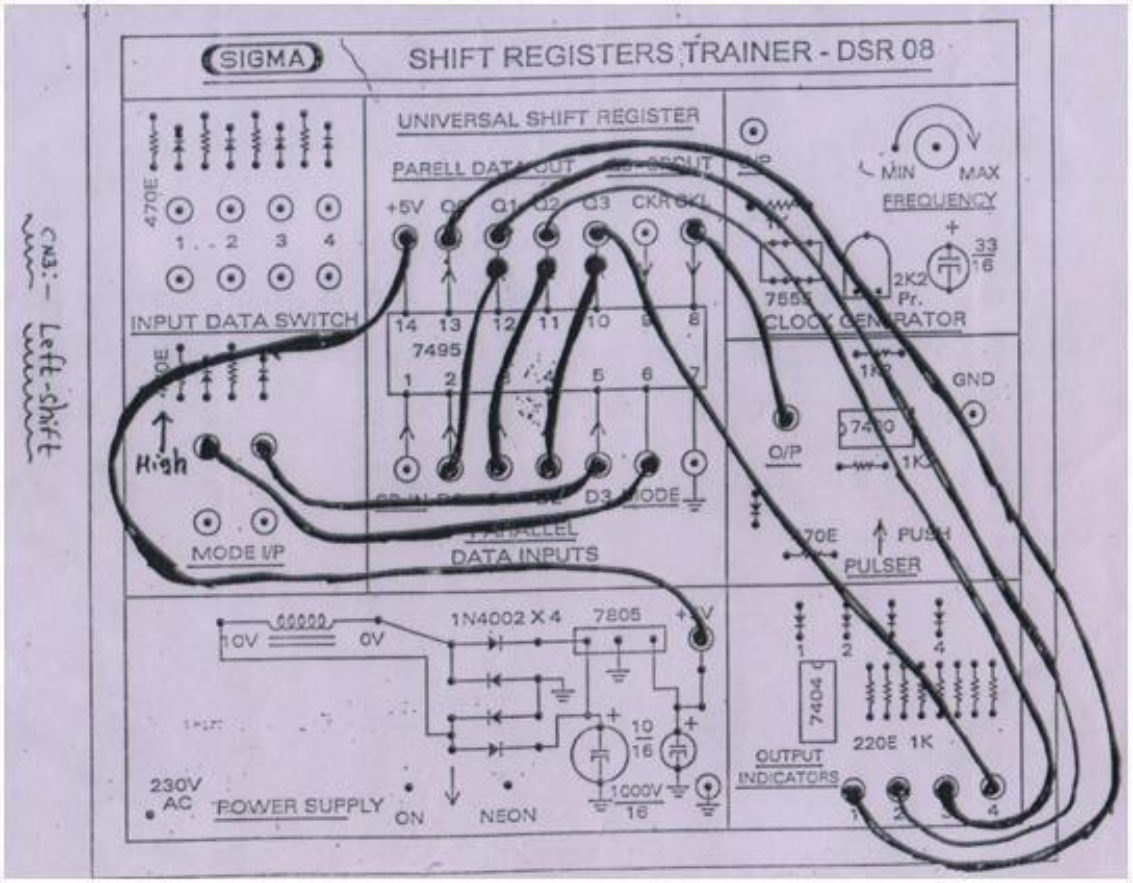
Finally, in the parallel loading mode (S1S0 = 11) data is read from the lines L0, L1, L2, and L3 simultaneously. Here, setting L3L2L1L0 = 1010 will cause Q3Q2Q1Q0 = 1010 after cycling the clock as depicted in Table 4.

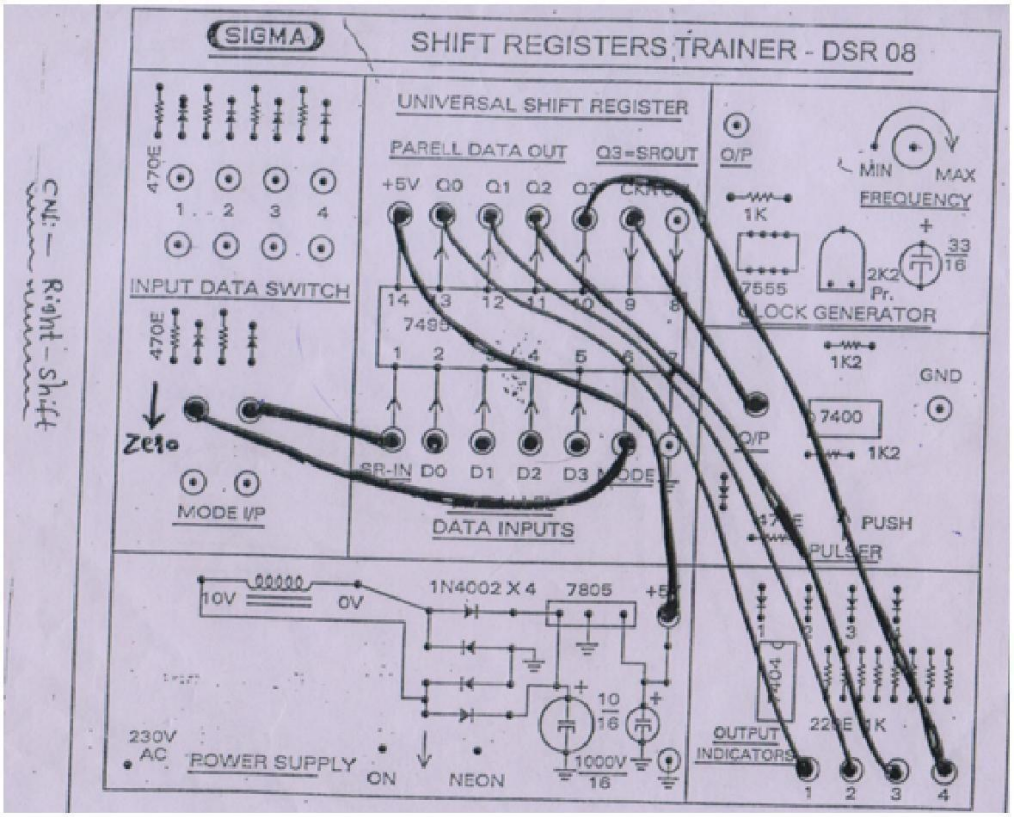
Clock Cycle	L3	L2	L1	L0		Q3	Q2	Q1	Q0
Initial Value	1	0	1	0		0	0	0	0
Cycle 1	1	0	1	0		1	0	1	0

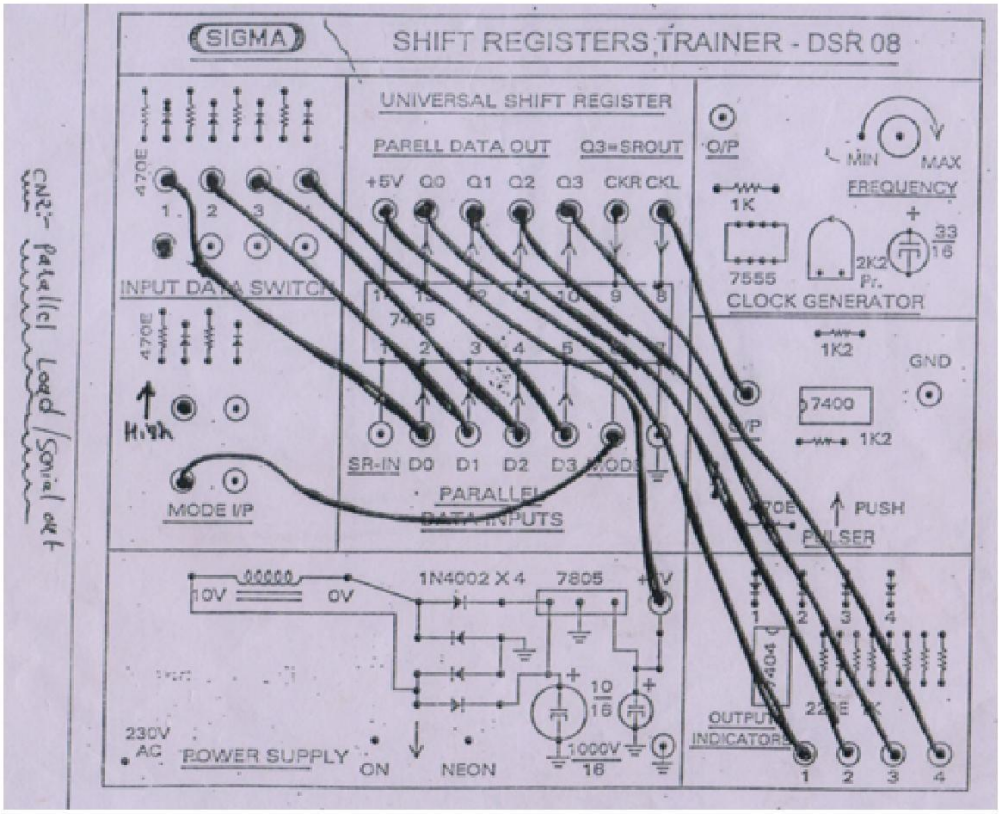
Table 4

Procedure:

i) Do the connection as per below kit connection diagram for various shift register.
ii) Apply proper input condition and observe the output information of led on/off.
iii) Compare theoretical data with observation and write conclusion.




SIGMA
SHIFT REGISTERS TRAINER - DSR 08
UNIVERSAL SHIFT REGISTER
PARELL DATA OUT
Q3=SROUT
+5V Q0 Q1 Q2 Q3 CKR CKL
INPUT DATA SWITCH
470E
1 2 3 4
7495
SR-IN D0 D1 D2 D3 MODE
DATA INPUTS
MODE I/P
Zero
O/P
MIN MAX
FREQUENCY
1K
7555
2K2 Pr.
33/16
CLOCK GENERATOR
1K2
GND
7400
PUSH
PULSER
10V 0V
1N4002 X 4
7805
+5V
10/16
1000V
230V AC
POWER SUPPLY
ON
NEON
OUTPUT INDICATORS
CNT:- Right-shift


SIGMA
SHIFT REGISTERS TRAINER - DSR 08
UNIVERSAL SHIFT REGISTER
PARELL DATA OUT
Q3=SROUT
+5V Q0 Q1 Q2 Q3 CKR CKL
INPUT DATA SWITCH
470E
1 2 3
7495
SR-IN D0 D1 D2 D3 MODE
PARALLEL DATA INPUTS
MODE I/P
High
O/P
MIN MAX
FREQUENCY
1K
7555
2K2 Pr.
33/16
CLOCK GENERATOR
1K2
GND
7400
PUSH
PULSER
10V 0V
1N4002 X 4
7805
+5V
10/16
1000V
230V AC
POWER SUPPLY
ON
NEON
OUTPUT INDICATORS
CNT:- Parallel Load/Serial out

Observation:

Right Shift

Clock Cycle	Shift•Right Switch	Q3	Q2	Q1	Q0
Initial Value	Initial Value	1	0	1	0
Cycle 1	1	1	1	0	1
Cycle 2	1	1	1	1	0
Cycle 3	1	1	1	1	1
Cycle 4	0	0	1	1	1
Cycle 5	0	0	0	1	1
Cycle 6	1	1	0	0	1
Cycle 7	0	0	1	0	0

Left Shift

Clock Cycle	Shift•Left Switch	Q3	Q2	Q1	Q0
Initial Value	Initial Value	1	1	0	1
Cycle 1	0	1	0	1	0
Cycle 2	1	0	1	0	1
Cycle 3	0	1	0	1	0
Cycle 4	0	0	1	0	0
Cycle 5	1	1	0	0	1
Cycle 6	1	0	0	1	1
Cycle 7	0	0	1	1	0

Parallel Loading

Clock Cycle	L3	L2	L1	L0		Q3	Q2	Q1	Q0
Initial Value	1	0	1	0		1	0	1	0
Cycle 1	0	0	0	0		0	0	0	0

Conclusion: In this practical we have learned to implement left shift register, right shift registers and Parallel loading register.

Experiment No. 12

Aim: Implement the operation of binary and decade counter.

Apparatus: Sigma counters trainer • DCO13, connecting wires

Theory:

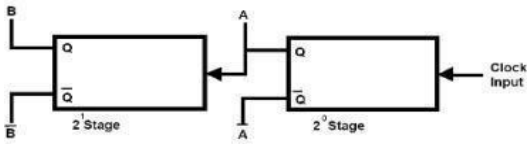
Counting is frequently required in digital computers and other digital systems to record the number of events occurring in a specified interval of time. Normally an electronic counter is used for counting the number of pulses coming at the input line in a specified time period. The counter must possess memory since it has to remember its past states. As with other sequential logic circuits counters can be synchronous or asynchronous. As the name suggests, it is a circuit which counts. The main purpose of the counter is to record the number of occurrence of some input. There are many types of counter both binary and decimal. Commonly used counters are

1. Binary Ripple Counter
2. Ring Counter
3. BCD Counter
4. Decade counter
5. Up down Counter
6. Frequency Counter

Binary Ripple Counter

A binary ripple counter is generally using bistable multivibrator circuits so that cache input applied to the counter causes the count to advance or decrease. A basic counter circuit is shown in Figure 1 using two triggered (T•type) flip flop stages. Each clock pulse applied to the T•input causes the stage to toggle. The Q and $\bar{Q}$ output terminals are always logically opposite. If the Q output is logical 1 (SET), the $\bar{Q}$ output is then logical 0. If the Q output is logical 0 (REST), then the $\bar{Q}$ output is logical 1.

The clock input causes the flip flop to toggle or change stage once clock pulse Figure 2 (a) shows the clock input signal and Q output signal. Notice that the circuit used in this case toggles on the trailing edge of the clock signal (when logic signal goes from 1 to 0). Referring back to Figure 1 the Q output of the first stage (called the 2^o stage or units position stage) is used here as the toggle input to the second stage (called the 2^1 or two's position stage). The Q output from the two successive stage are marked A and B, respectively, to differentiate them. Notice that the $\bar{Q}$ output of each stage is marked with a negative bar over the letter designation, so that whatever logical stage A is at, $\bar{A}$ is the opposite logical state. Since the Q output (A signal) from the first stage triggers the second stage, the second stage changes state only when the Q output of first stage goes from logical 1 to logical 0 as shown in Figure 2(b).



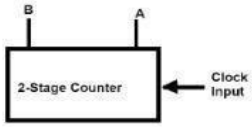


Fig 1: Basic Two-stage binary Counter

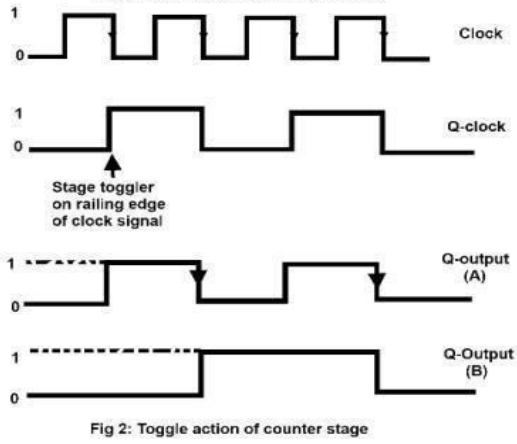


Fig 2: Toggle action of counter stage

Table 1

COUNT FOR 2•STAGE BINARY COUNTER

Input Pulses	2^n Output (B)	2^n Output (A)
0	0	0
1	0	1
2	1	0
3	1	1
4 or 0	0	0

An arrow is included on the waveform of stage A as a reminder that it triggers stage B only on a trailing edge (1 or 0 logical change). Notice that the output waveform of succeeding stage operates half as fast as its input. To see that this circuit operates as a binary counter a table can be prepared to show the Q output states after each clock pulse is applied. Table 1 shows this operation for the circuit of Figure 1.

To see how a counter is made using more stage considers the 4 stage counter of Figure 3. The counter is simply made with the Q output of each state connected as the toggle input to the succeeding state. With four stages the counter cycle will repeat every sixteen clock pulses. In general there are 2^n counts with an n•stage counter. For the four stages used here the count goes 2^4 or 16 steps as a rule, for a binary counter.

$$\text{Number of counts} = N = 2^n$$

Where, n = number of counter stage. A six stage counter n = 6 would be provide a count that repeats every $N = 2^6 = 64$ counts. A ten•stage counter (n = 10) would recycle every $N = 2^{10} = 1024$ counts.

Decade Counter

A decade counter is the one which goes through 10 unique combinations of outputs and then resets as the clock proceeds. We may use some sort of a feedback in a 4•bit binary counter to skip any six of the sixteen possible output states from 0000 to 1111 to get to a decade counter. A decade counter does not necessarily count from 0000 to 1001 it could count as 0000,0001, 0010, 1000, 1001, 1010, 1011, 1110, 1111, 0000, 0001 and so on. Figure 6 shows a decade counter having a binary count that is always equivalent to the input pulse count. The circuit is essentially, a ripple counter which count up to 16. We desire however, a circuit operation in which the count advance from 0 to 9 and then reset to 0 for a new cycle. This reset is a accomplished at the desired count as follows.

1. With counter REST count = 0000 the counter is ready to stage counter cycle.
2. Input pulses advance counter in binary sequence up to count of a (count = 1001)
3. The next count pulse advance the count to 10 count = 1010. A logic NAND gate decodes the count of 10 providing a level change at that time to trigger the one shot unit which then resets all counter stages. Thus, the pulse after the counter is at count = 9, effectively results in the counter going to count = 0.

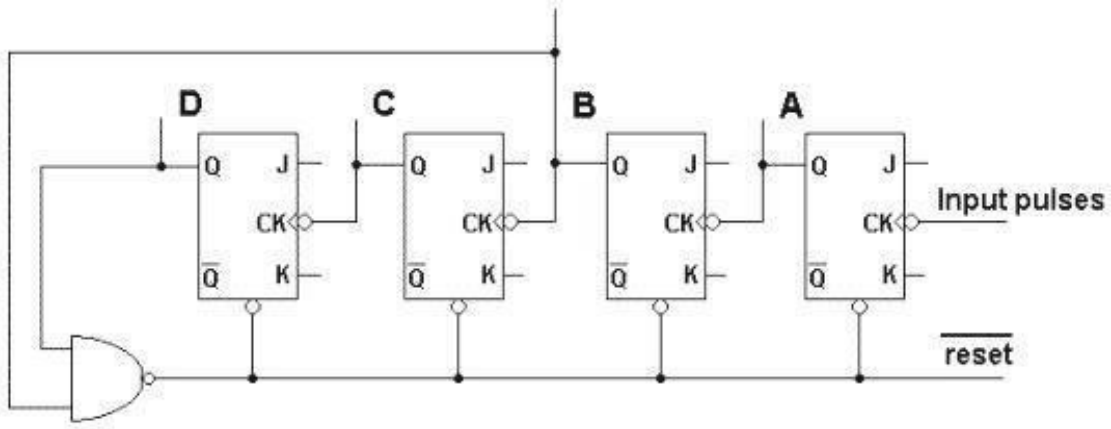


Figure 6: Decade Counter

Input Pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
0	0	0	0	0

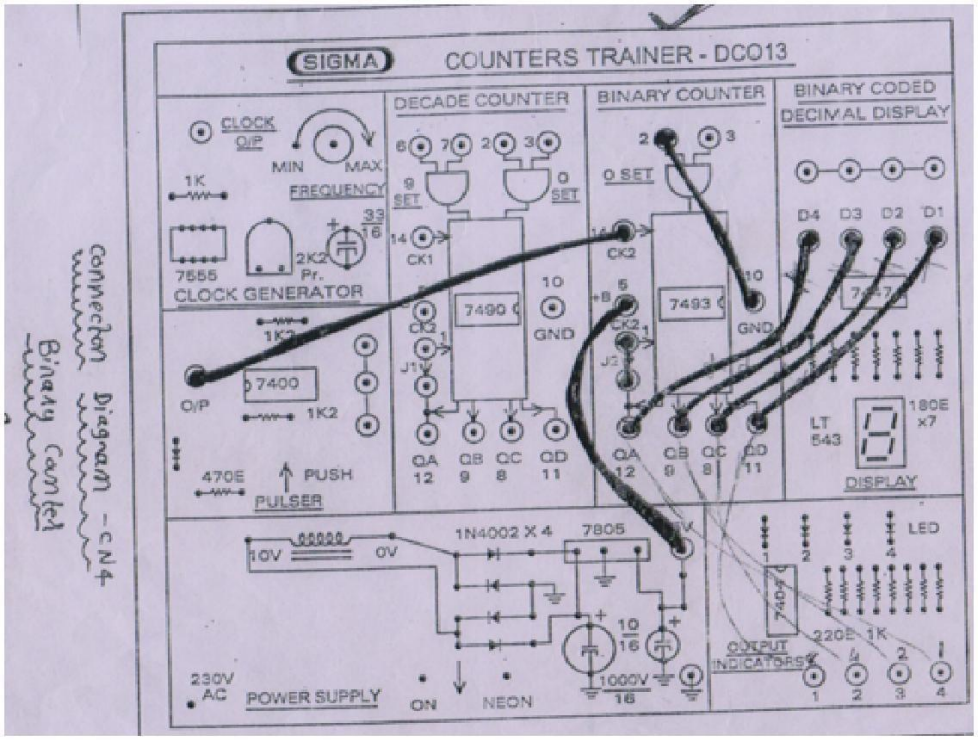
Table provides a count table showing the binary count equivalent to the decimal count of input pulses. The table also shows that the count goes momentarily count from nine (1001) to ten (1010) before resetting to zero(0000). The NAND gate provides an output of 1 until the count reach ten. The count of ten is decoded (or sensed in this case) by using logic inputs that are all 1 at the count of ten. When the count becomes ten the NAND gate output goes to logical 0, providing a 1 to 0 logic change to trigger the one shot unit, which then provides a short pulse to reset all counter stages.

The Q signal is used since it is normally high and goes low during the one shot timing period the flip flop in this circuit being reset by a low signal level (active low clearing). The one shot pulse need only be long enough so that slowest counter stage resets. Actually, at this time only the $2^1$and 2^3 stage need be reset, but all stages are reset to insure that a new cycle at the count 0000.

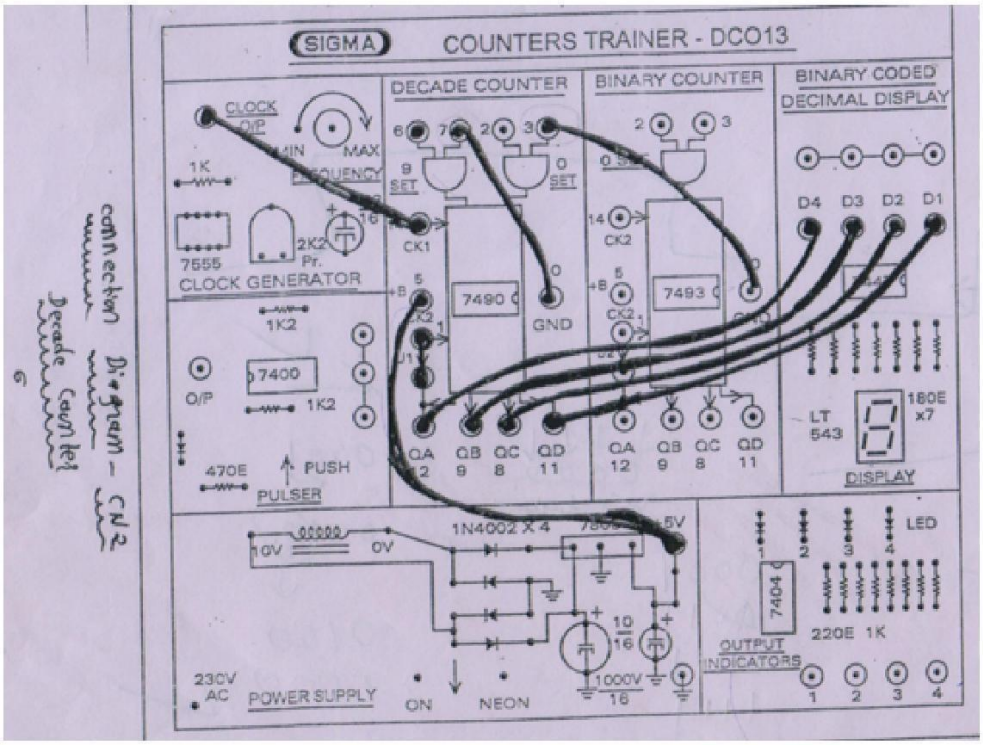
Procedure:

i) Do the connection as per below kit connection diagram for various shift register.
ii) Apply proper input condition and observe the output information of led on/off.
iii) Compare theoretical data with observation and write conclusion.

connection Diagram - CN4
Binary Counter



connection Diagram - CN2
Decade Counter



Observation:

4 -Bit Binary Counter

Input pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Decade Counter:

Input pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

Conclusion: In this practical, I have learnt to implement the operation of binary and decade