

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018, KARNATAKA



LAB MANUAL REPORT
ON
“LOGIC DESIGN LAB USING PSPICE/MULTISIM
(21EC381)
(IIIrd SEMESTER ECE).”

Submitted by

PRAJWAL N G
1CR21EC151

Department Of Electronics and Communication Engineering
October – February 2023



Department Of Electronics and Communication Engineering
CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,
BENGALURU-560037

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



CERTIFICATE

This is to certify the Ability Enhancement Course Report entitled “**Logic Design Lab Using Multi Sim**”, prepared by **PRAJWAL N G**, bearing **USN 1CR21EC151**, a bona fide student of **CMR Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of **Bachelor of Engineering in Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belagavi-590018** during the academic year 2022-23.

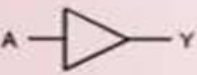


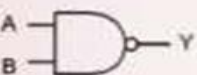


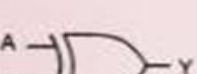
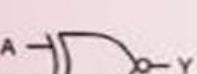
This is certified that all the corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Ability Enhancement Course has been approved as it satisfies the academic requirements prescribed for the said degree.

Signature of HOD
Dr. R Elumalai
Professor & HOD
Dept. of ECE, CMRIT

CONTENTS

Sl. No.	Experiments using Pspice/MultiSIM	Page No.
	Note: Standard design procedure to be adopted.	
1	Implementation of De Morgan's theorem and SOP/POS expressions using Pspice/Multisim.	4-10
2	Implementation of Half Adder, Full Adder, Half Subtractor and Full Subtractor using Pspice/Multisim.	11-16
3	Design and implementation of 4-bit Parallel Adder/ Subtractor using IC 7483 and BCD to Excess-3 code conversion and vice-versa using Pspice/Multisim.	17-24
4	Design and implement of IC 7485 5-bit magnitude comparator using Pspice/Multisim.	25-27
5	To Realize Adder & Subtractor using IC 74153 (4:1 MUX) and 4-variable function using IC74151 (8:1MUX) using Pspice/Multisim.	28-37
6	To realize Adder and Subtractor using IC 74139/74155N (Demux/Decoder) and Binary to Gray code conversion & vice versa using 74139/74155N using Pspice/Multisim.	38-49
7	SR, Master-Slave JK, D & T flip-flops using NAND Gates using Pspice/Multisim.	50-56
8	Design and realize the Synchronous counters (up/down decade/binary) using Pspice/Multisim.	57-64
9	Realize the shift registers and their modes (SISO, PISO, PIPO, SIPO) using 7474/7495 using Pspice/Multisim.	65-69
10	Design Pseudo Random Sequence generator using 7495 using Pspice/Multisim.	70-74
11	Design Serial Adder with Accumulator and simulate using Pspice/Multisim.	75-76
12	Design using Pspice/Multisim Mod-N Counters.	77-82

BASIC GATES

Buffer		<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Y	0	0	1	1	$Y = A$									
A	Y																	
0	0																	
1	1																	
Inverter (NOT gate)		<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	$Y = \bar{A}$									
A	Y																	
0	1																	
1	0																	
2-input AND gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$
A	B	Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
2-input NAND gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = \overline{A \cdot B}$
A	B	Y																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
2-input OR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
2-input NOR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = \overline{A + B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
2-input EX-OR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
2-input EX-NOR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = \overline{A \oplus B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

EXPERIMENT 1

Aim: To Verify (a) De Morgan's Theorem for 2 variables.

(b) The sum-of product and product-of-sum expressions.

Components Required:

Sl.No	Group	Family	Components
1.	TTL	74LS	IC7408
2.	TTL	74LS	IC7432
3.	TTL	74LS	IC7404
4.	TTL	74LS	IC7400
5.	TTL	74LS	IC7402
6.	Sources	Digital Sources	INTERACTIVE_DIGITAL_CONSTANT
7.	Indicators	PROBE	PROBE_GREEN,PROBE_RED

Theory: Augustus De Morgan, a 19th-century British mathematician developed a pair of important rules regarding group complementation in Boolean algebra referred to as De Morgan's theorems. Applications of these rules include simplification of logical expressions in computer programs and digital circuit designs. They are as follows.

De Morgan's First Theorem: The complement of a product is equal to the sum of complements.

$$(AB)' = A' + B'$$

De Morgan's Second Theorem: The complement of a sum term is equal to the product of complements.

$$(A+B)' = A' \cdot B'$$

In a Boolean function, the variables appear either in complemented or an uncomplemented form. Each occurrence of a variable in either form is called a "literal". These literals can be grouped either as a product term or sum term. A product term is defined as either a literal or product of literals (also called conjunction). A sum term is defined as either a literal or sum of literals (also called disjunction).

These literals and terms are arranged in two forms

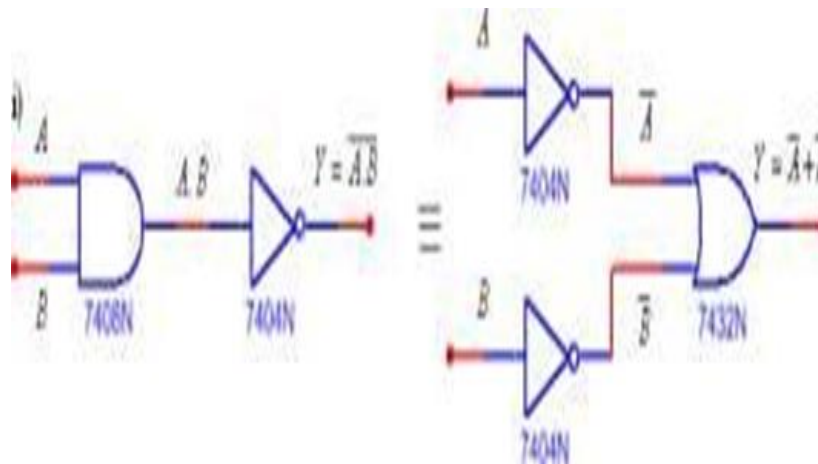
(i) Sum of Products (SOP): It is the group of product terms ORed together. If each product term (also called minterm) in SOP contains all literals, then it is called canonical SOP.

(ii) Product of Sum (POS): It is the group of sum terms ANDed together. If each sum term (also called maxterm) in POS contains all literals, then it is called canonical POS.

(a) Verification of De Morgan's theorem for two variables

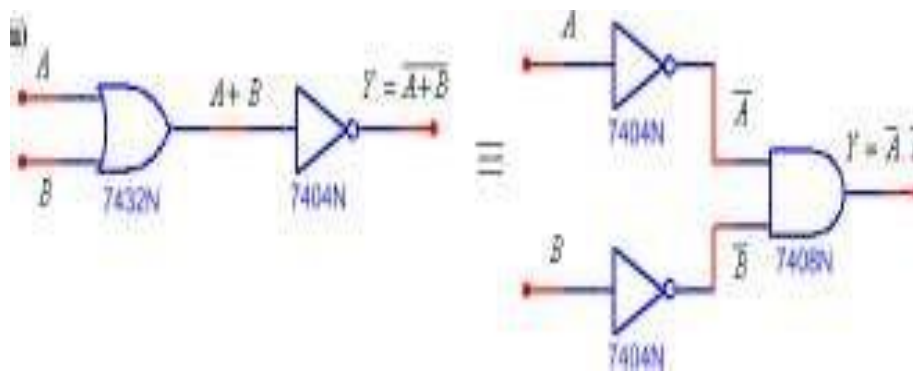
Truth Table:

Inputs					Outputs	
A	B	A'	B'	A.B	$(A.B)'$	A+B
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

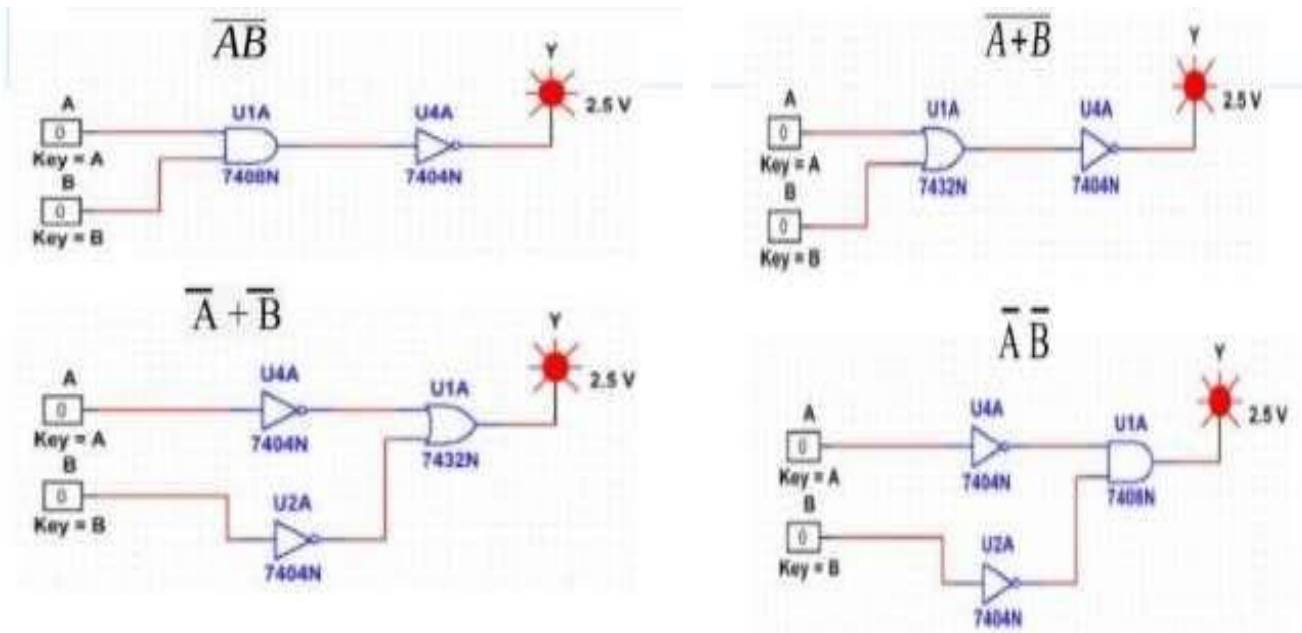


Truth Table:

Inputs					Outputs	
A	B	A'	B'	A+B	$\overline{(A+B)}$	A.B
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0



Multisim Simulation

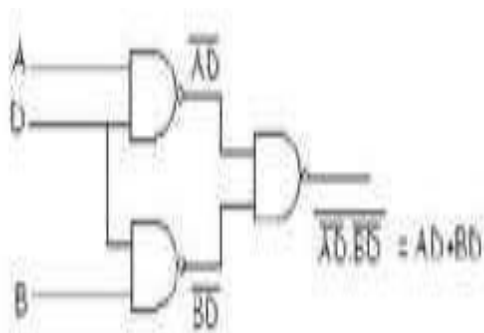


b) To verify ii) SOP and POS expressions using universal gates Sum of Product (SOP): $Y = BD + AD$ $Y = F(A, B, C, D) = \sum(5, 7, 9, 11, 13, 15)$

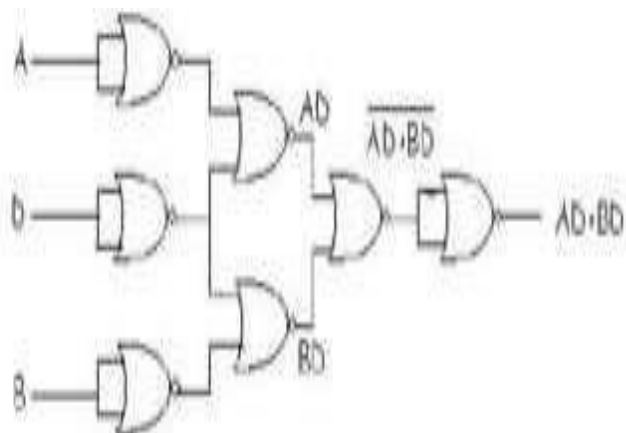
AB \ CD	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$Y = AD + BD$

Implementation Using NAND gates

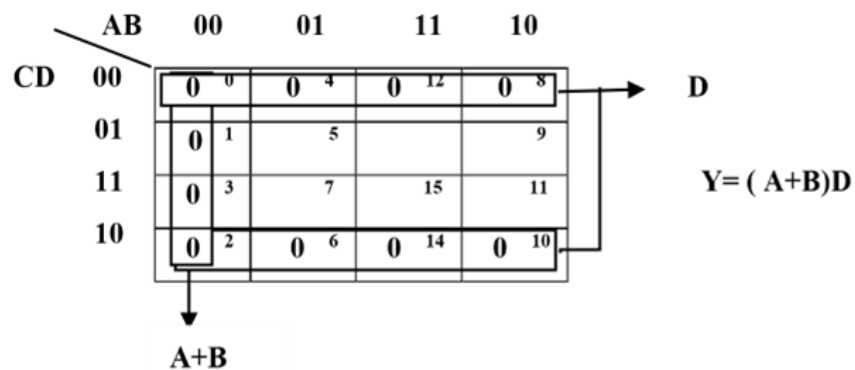


Implementation using NOR gates

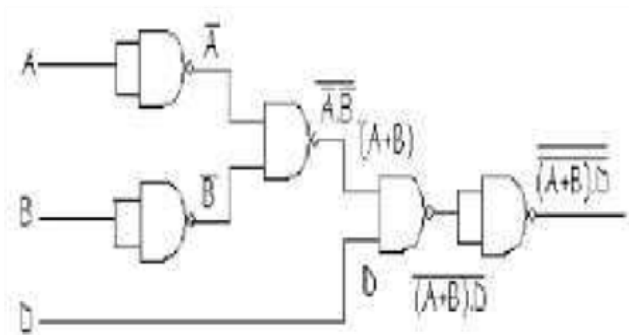


Product of Sum (POS): $Y=(A+B)D$ $Y=F(A,B,C,D) = \Pi(0,1,2,3,4,6,8,10,12,14)$

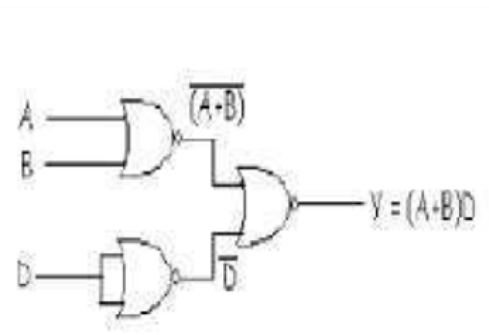
Simplification using K-map:



Implementation Using NAND gates



Implementation using NOR gates



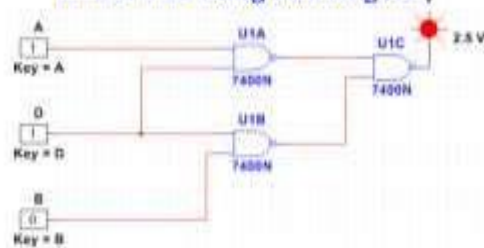
Truth Table:

Inputs						Output
A	B	C	D	AD	BD	$Y=AD+BD$
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	1	1	1	1

Inputs					Output
A	B	C	D	A+B	$Y=(A+B)D$
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	1	1

Multisim Simulation

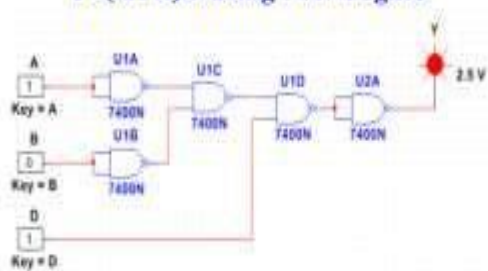
$Y=BD+AD$ using NAND gate



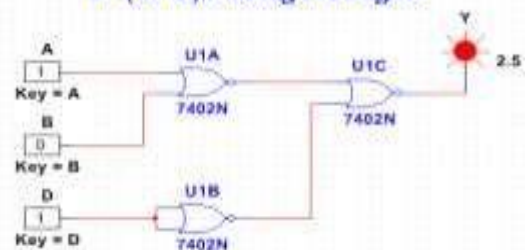
$Y=BD+AD$ using NOR gate



$Y=(A+B)D$ using NAND gate



$Y=(A+B)D$ using NOR gate



Result:

a) De morgan's theorem for 2 variables is simulated using multisim and verified.

b) SOP and POS expressions are simulated and verified

EXPERIMENT 2

Implementation of Half Adder, Full Adder, Half Subtractor and Full Subtractor

Aim: To realize half /full adder using logic gates and NAND Gates

Components required:

Sl.No	Group	Family	Components
1.	TTL	74LS	IC7408
2.	TTL	74LS	IC7486
3.	TTL	74LS	IC7404
4.	TTL	74LS	IC7400
5.	TTL	74LS	IC7432
6.	Sources	Digital Sources	INTERACTIVE_DIGITAL_CONSTANT
8.	Indicators	PROBE	PROBE_GREEN, PROBE_RED

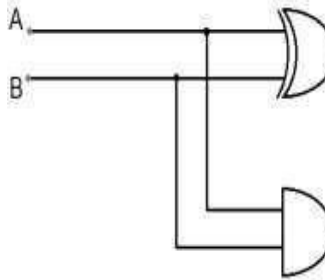
Theory: A combinational circuit that performs the addition of two bits is called a half adder. A half adder needs two binary inputs and two binary outputs. A half adder has no provision to add a carry from the lower order bits when binary numbers are added. When two input bits and a carry are to be added the number of input bits becomes three and the input combination increases to eight. For this full adder is used.

A combinational circuit that performs the addition of three bits is a full adder. It consists of three inputs and two outputs. Like half adder it also has a sum bit and a carry bit. The new carry generated is represented by 'Cout' and the carry generated from the previous addition is represented by 'Cin'.

When two input bits and a borrow have to be subtracted the number of input bits equal to three and the input combinations increases to eight, for this a full subtractor is used.

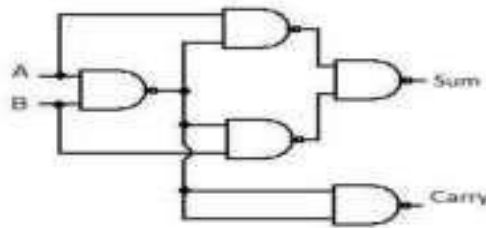
Half Adder using Basic gates

Truth Table



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

Half adder using NAND gates

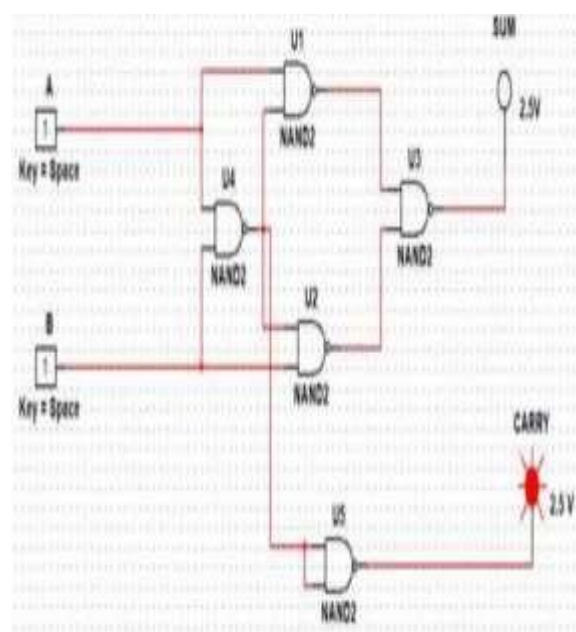
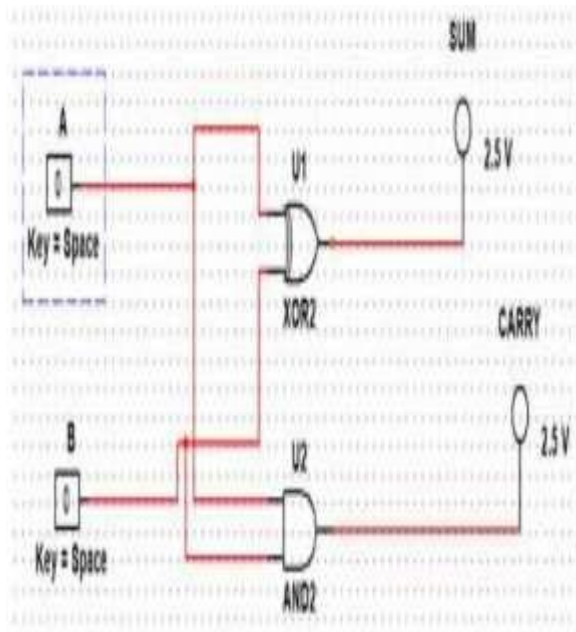


Boolean Expression:

$$\text{Sum} = A \oplus B$$

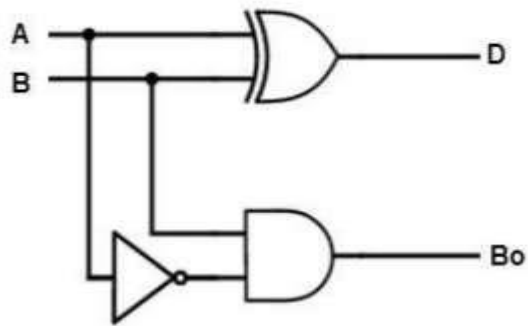
$$\text{Carry} = A \cdot B$$

Multisim Simulation



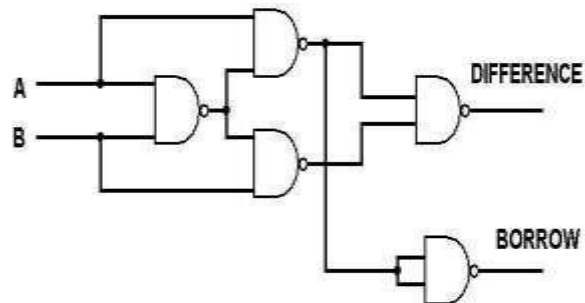
Half Subtractor using Basic gates

TruthTable

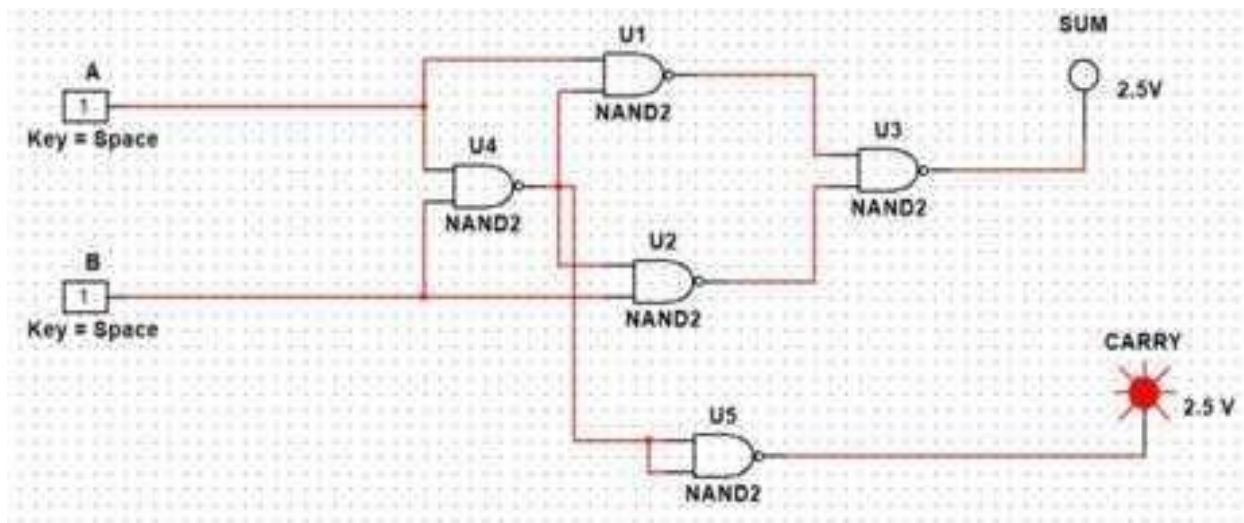


A	B	Difference (D)	Borrow(B ₀)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Half Subtractor using NAND gates



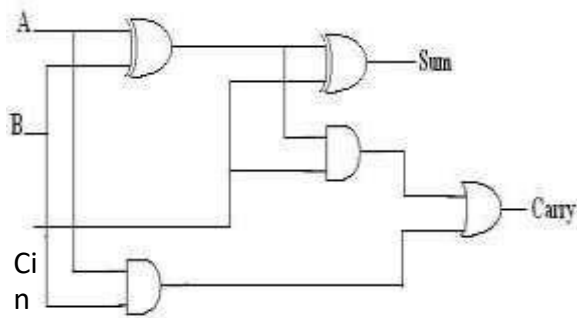
Multisim Simulation



Half Subtraction using NAND gates

Full Adder using Basic gates

NAND Gates

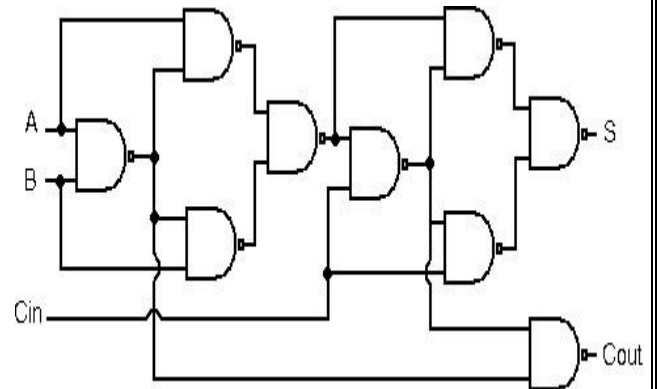


Truth Table

A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

AC

Full adder using

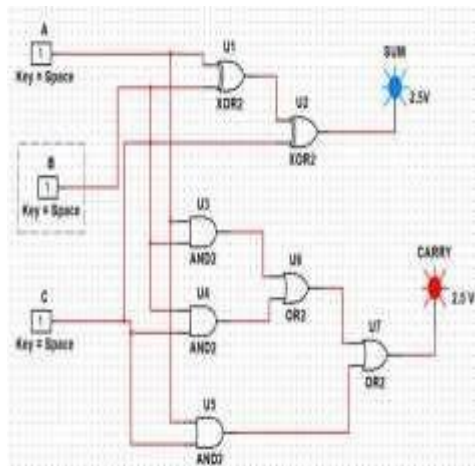


Boolean Expression:

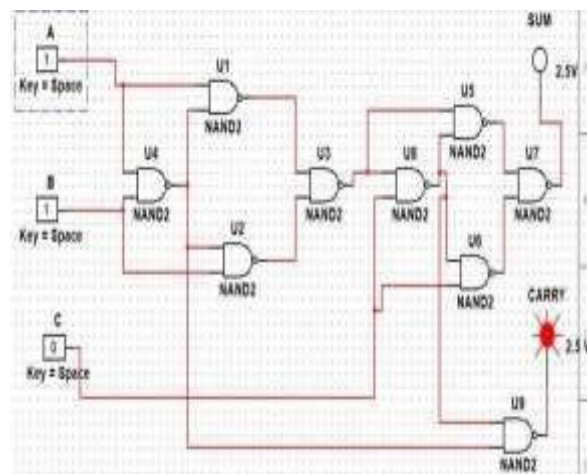
$$\text{Sum} = A \oplus B \oplus C$$

$$\text{Carry} = AB + BC + AC$$

Multisim Simulation

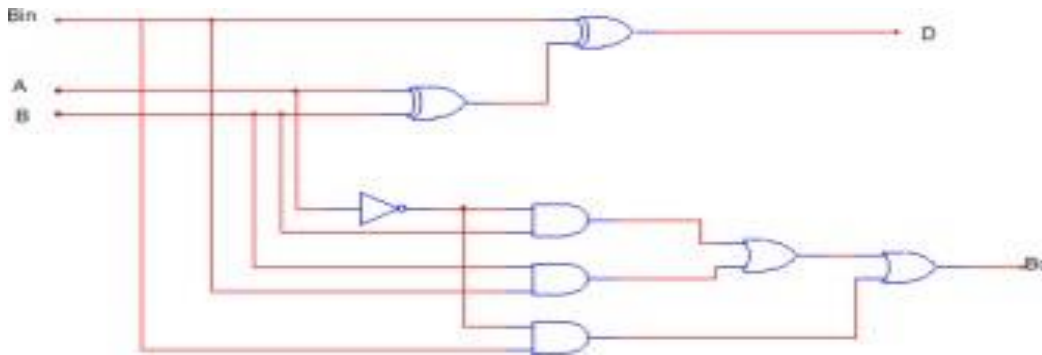


Using Basic Gates



Using NAND Gates

Full Subtractor using Basic gates



Truth Table

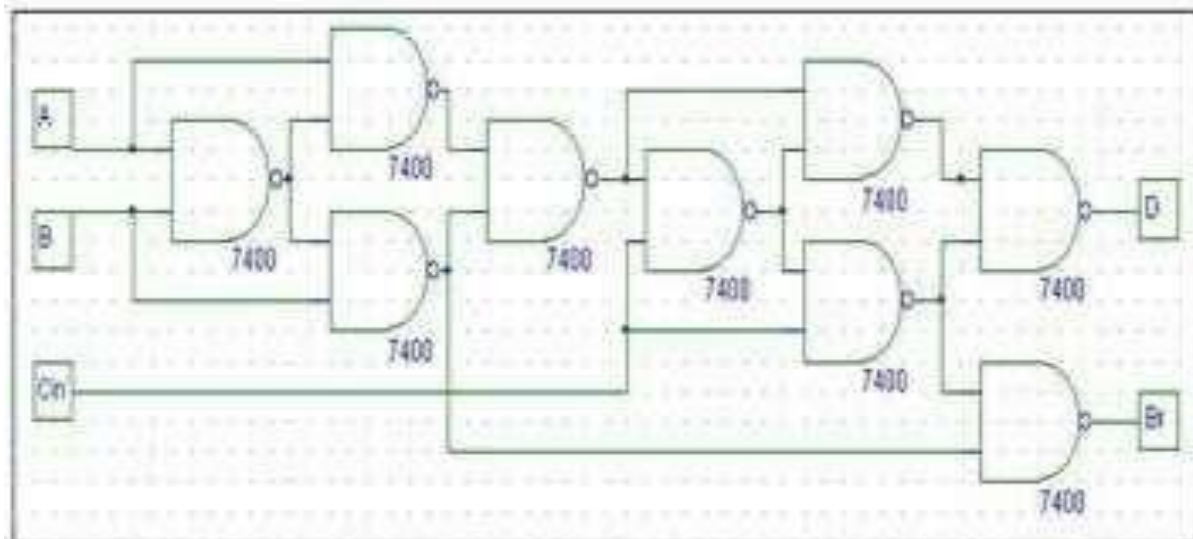
INPUTS			OUTPUTS	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Boolean Expression:

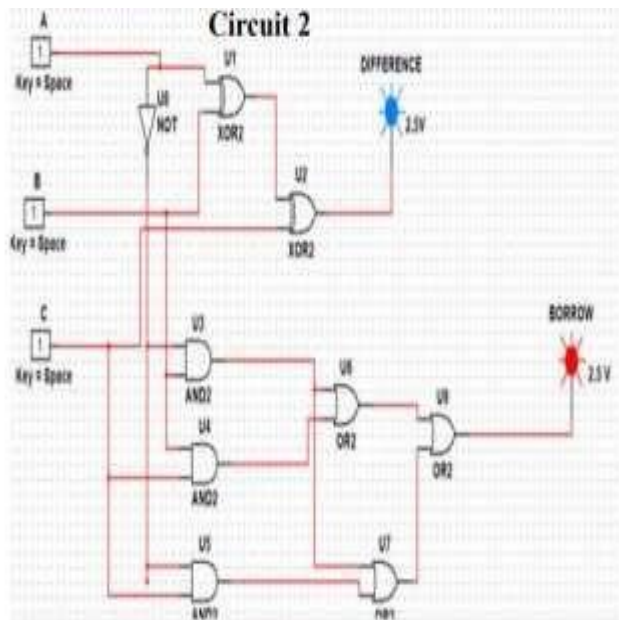
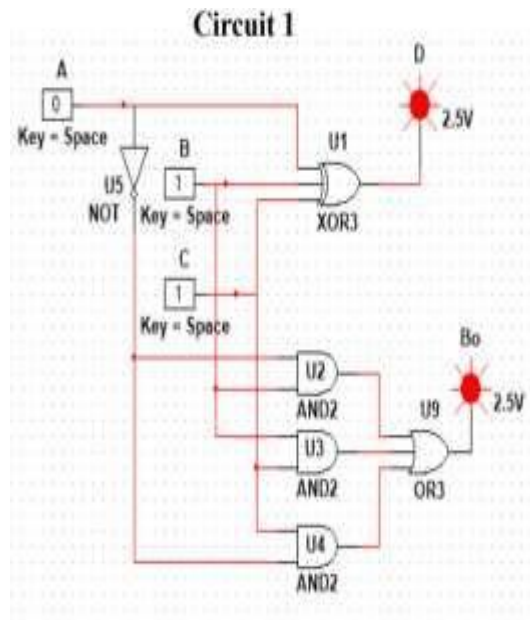
$$\text{Difference} = A \oplus B \oplus \text{Bin}$$

$$\text{Bout} = A(B \oplus \text{Bin}) + B \text{Bin}$$

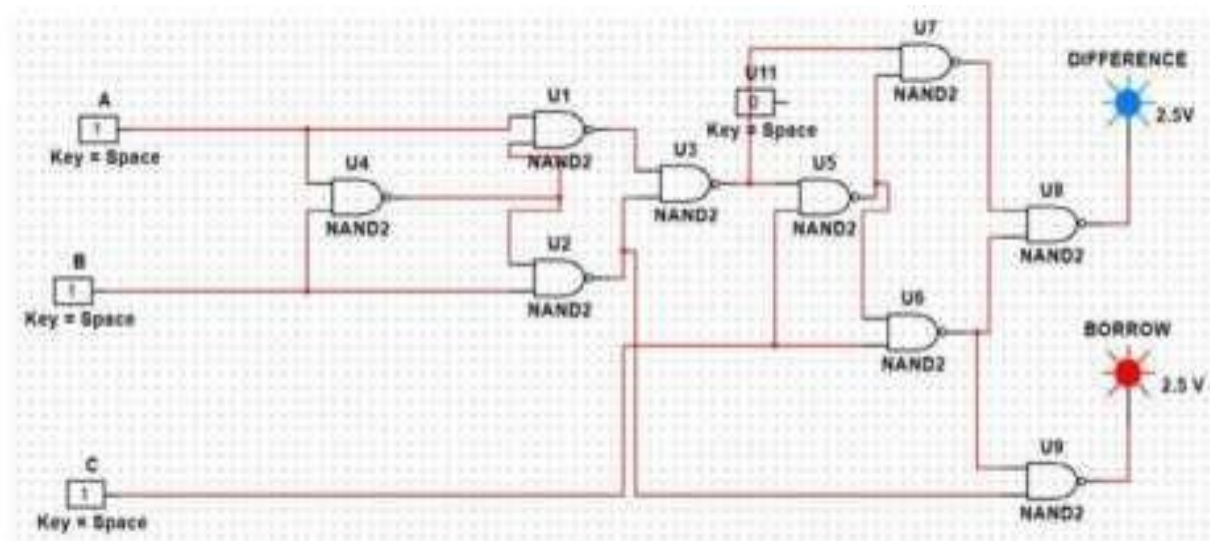
Full Subtractor using NAND Gates



Multisim Simulation:



Full Subtractor using Basic gates



Full Subtractor using NAND Gates

Result:

Adders and Subtractors are simulated and verified

EXPERIMENT 3

Design and implementation of 4-bit Parallel Adder/ Subtractor using IC 7483 and BCD to Excess-3 code conversion and vice-versa

Aim: To design and set up the following circuit using IC 7483.

- i) A 4-bit binary parallel adder.
- ii) A 4-bit binary parallel subtractor.
- iii) BCD to Excess -3 conversion and vice- Versa

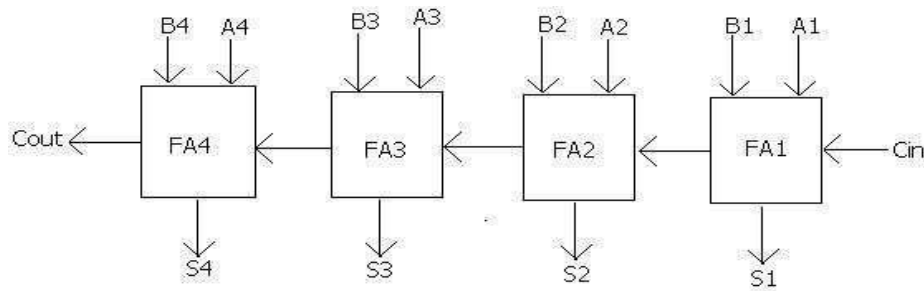
Components:

Sl.No	Group	Family	Components
1.	TTL	74LS	IC7483
2.	BASIC	SWITCH	DSWPK-4
3.	Sources	Power Sources	VCC
4.	Sources	Power Sources	GND
5.	Indicators	PROBE	PROBE_GREEN, PROBE_RED

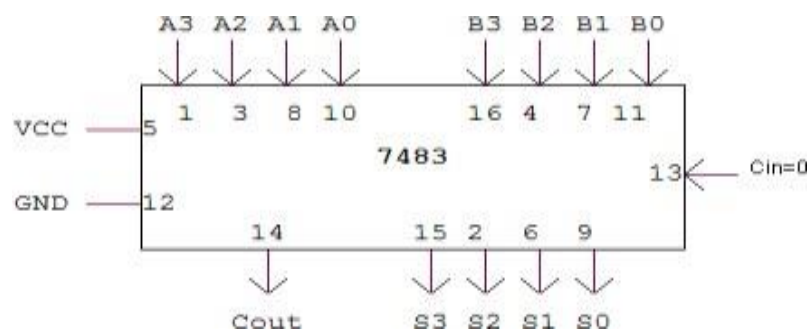
Theory: Many high speed adders available in integrated circuit form utilize the look ahead carry or a similar technique for reducing overall propagation delays. A parallel adder consists of n number of full adders and look ahead carry circuitry needed for high speed operation. A parallel subtractor is one where subtraction done by full adder and ahead carry circuitry. For subtraction C_{in} is made equal to 1 and A-B format is used.

Code converter is a combinational circuit that translates the input code word into a new corresponding word. The excess-3 code digit is obtained by adding three to the corresponding BCD digit. To Construct a BCD-to-excess-3code converter with a 4-bit adder feed BCD code to the 4- bit adder as the first operand and then feed constant 3 as the second operand. The output is the corresponding excess-3 code. To make it work as a excess-3 to BCD converter, we feed excess-3 code as the first operand and then feed 2's complement of 3 as the second operand. The output is the BCD code.

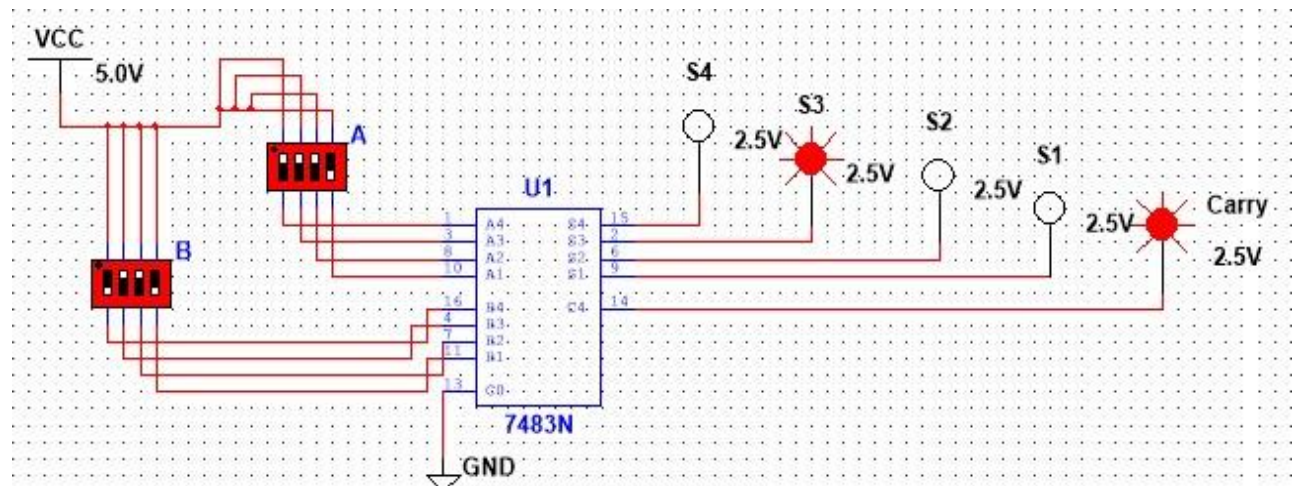
Block Diagram



Adder Circuit



Multisim Simulation



An Example: $14+6=20$ (0101)

7 is realized at **A3 A2 A1 A0 = 1 1 1 0**

2 is realized at **B3 B2 B1 B0 = 0 1 1 0**

Sum = 0 1 0 0

Cout = 1

A 4-bit binary Parallel Subtractor.

i) 4-BIT BINARY SUBTRACTOR (A>B).

Subtraction is carried out by adding 2's complement of the subtrahend.

Example: $8 - 3 = 5$ (0101)

8 is realized at **A3 A2 A1 A0** = 1000

3 is realized at **B3 B2 B1 B0** through X-OR gates = 0011

Output of **X-OR** gate is 1's complement of **3** = 1100

2's Complement can be obtained by adding **Cin** = 1

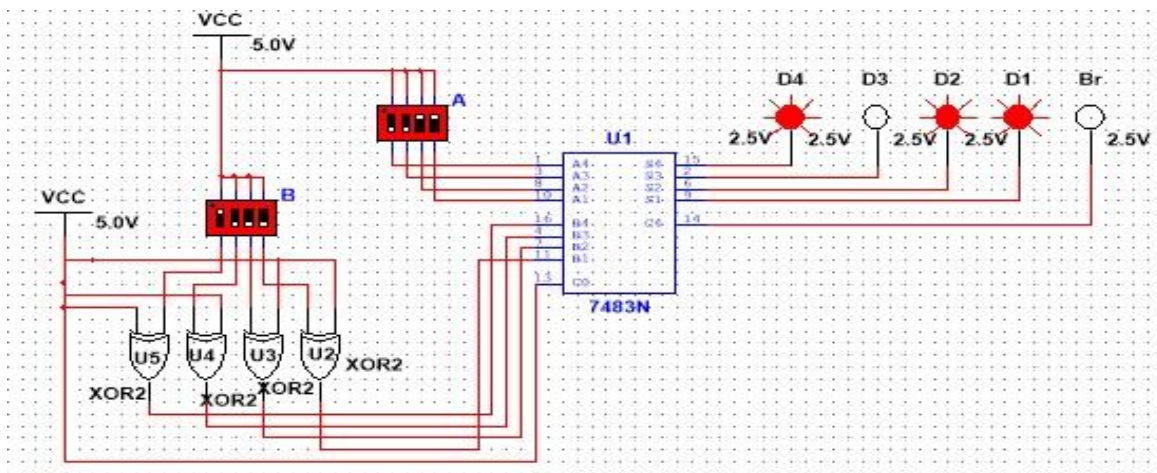
Therefore Cin = 1

A3 A2 A1 A0 = 1 0 0 0

B3 B2 B1 B0 = 1 1 0 1

S3 S2 S1 S0 = 0 1 0 1

Cout = 1 (Ignored) (Result is positive)



ii) 4-BIT BINARY SUBTRACTOR (A<B).

Subtraction is carried out by adding 2's complement of the Subtrahend

Example : $3 - 8 = -5$ (1011)

3 is realized at **A3 A2 A1 A0** = 0011

8 is realized at **B3 B2 B1 B0** = 1000

Output of **X-OR** gate is 1's complement of 8 = 0111

2's Complement can be obtained by adding **Cin** = 1

Cin = 1

A3 A2 A1 A0 = 0 0 1 1

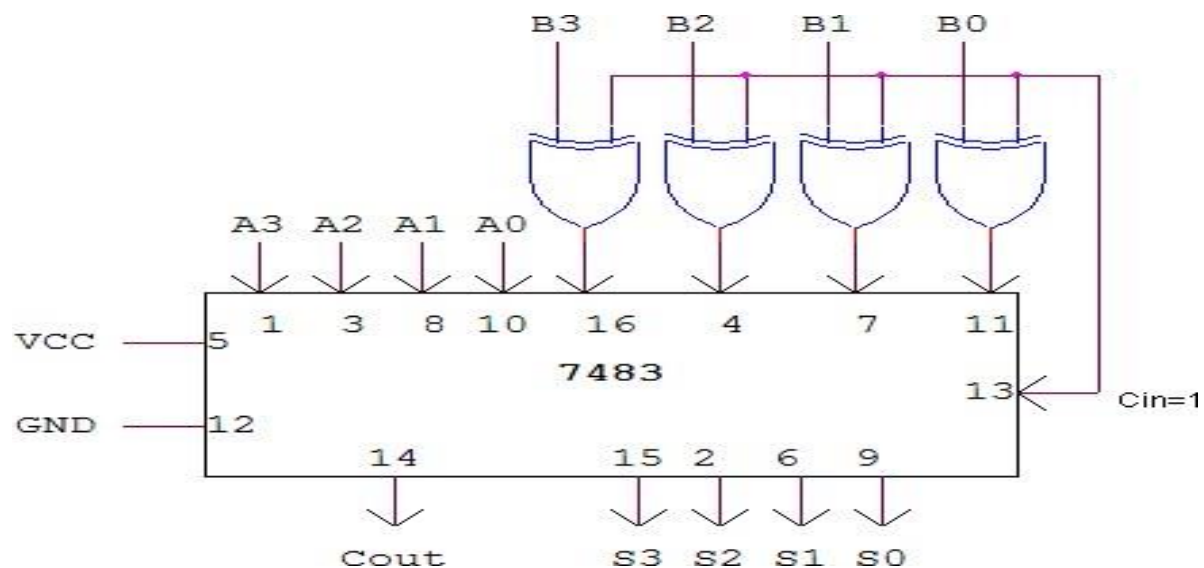
B3 B2 B1 B0 = 0 1 1 1

S3 S2 S1 S0 = 1 0 1 1

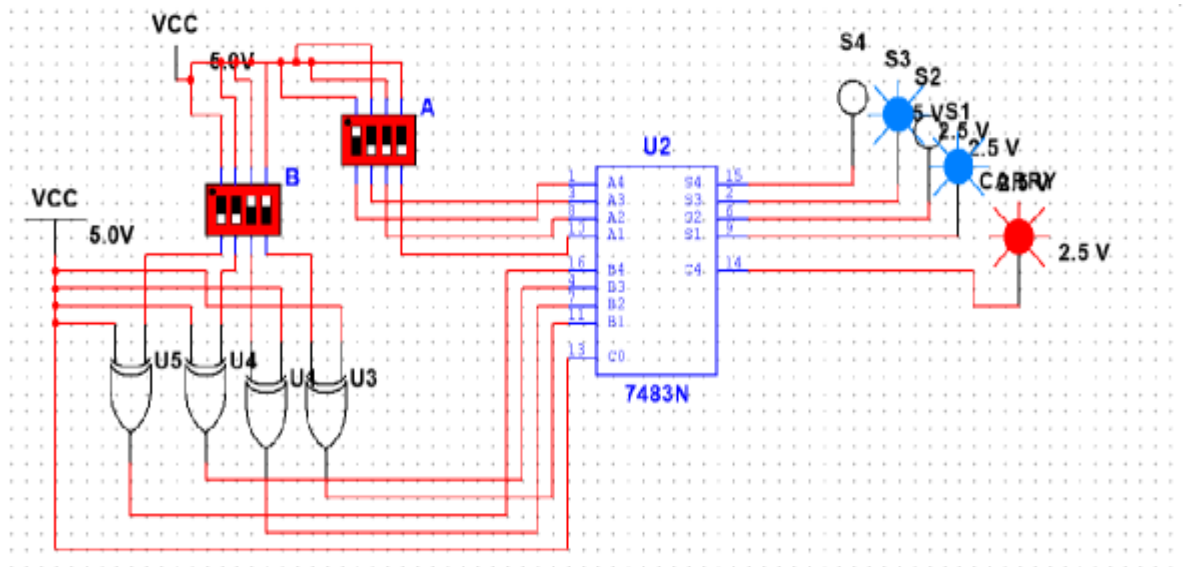
Cout = 0 (Result is negative)

Take 2's complement of **S3 S2 S1 S0** and the result will be **-5**

Block Diagram



Multisim Simulation



Full Subtractor using 7483

Cin	A	B	Cout	S3	S2	S1	S0	Remarks
0	7	2	0	1	0	0	1	Cout = 0
0	15	15	1	1	1	1	0	Cout = 1
1	8	3	1	0	1	0	1	A>B, result is positive, Cout is ignored.
1	3	8	0	1	0	1	1	A<B, result is negative (2's complement form), Cout=0 indicates negative result.

Multisim procedure:

1. To Place Components, click on **Place/Components**. On the Select Component window, click on **Group-> All groups**. Type the components needed for the circuit. Click **OK** to place the component on the schematic.
2. Place component 7483 from the components-all groups list.
3. Place digital ground DGND from the component list.
4. Place DSWPK-4 , and xor gates from the component list and name it as A and B.

5. Select Probes from component list and place at outputs to get 4 bit sum and carry out.
6. Once the components are placed, wire the circuit by clicking on **Place/Wire** drag and place the wire. Components can also be connected by clicking the mouse over the terminal edge of one component and dragging to the edge of another component.
7. After saving the circuit, Click simulation.
8. For different cases of input check the outputs

BCD to Excess 3 Conversion

In parallel adder circuit using IC 7483, B=0011, Give Values to A from 0000 to 1001(Valid BCD number)

Truth Table:

Boolean Expression:

BCD Inputs				Excess-3 Outputs			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

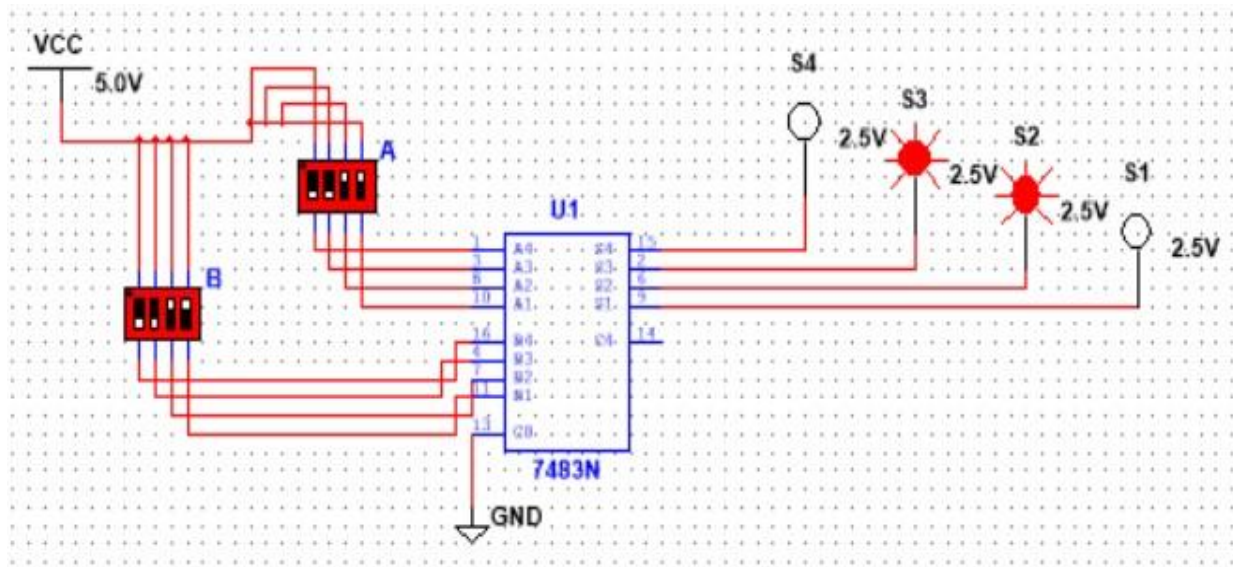
$$W = A + BC + BD$$

$$X = BC + BD + \overline{B}C\overline{D}$$

$$Y = CD + C\overline{D}$$

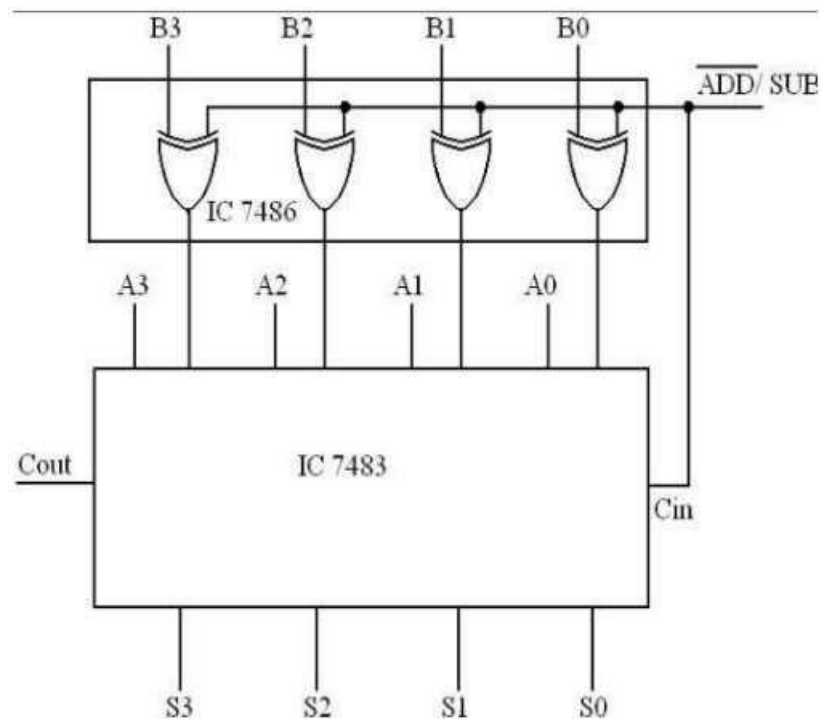
$$Z = D$$

Multisim Simulation:



Excess 3 to BCD

In parallel subtractor circuit using IC 7483, B=0011, Give values to A form 0011 to 1100 (valid BCD number)



Truth Table:

Boolean Expression:

Excess-3 Inputs				BCD Outputs			
W	X	Y	Z	A	B	C	D
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

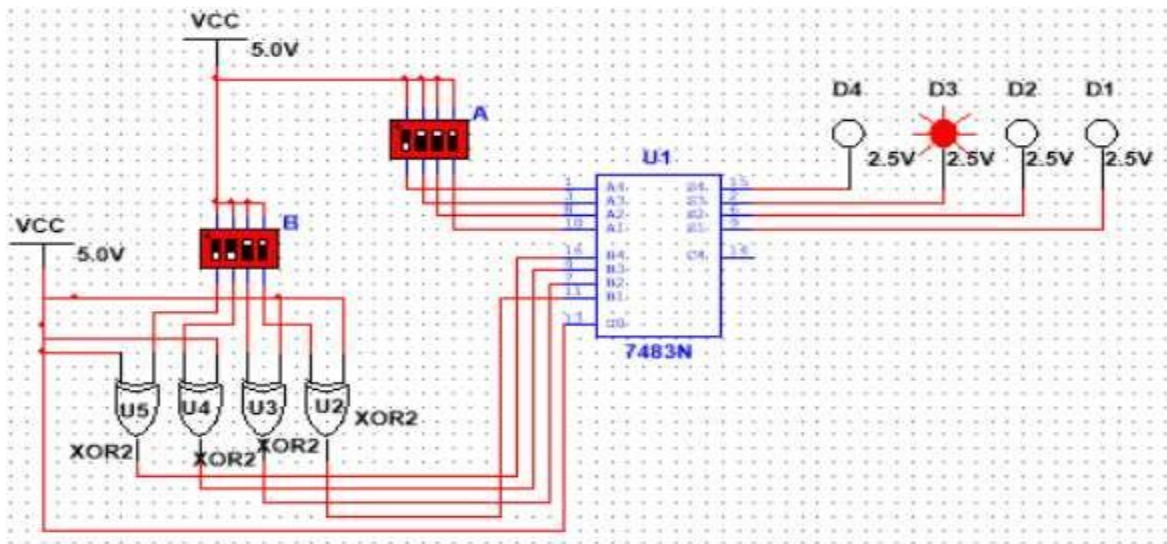
$$A = WX + WYZ$$

$$B = XY + XZ + XYZ$$

$$C = YZ + YZ$$

$$D = Z$$

Multisim Simulation



Result:

Parallel Adders and Subtractors, BCD to Excess-3 and vice versa are simulated and verified.

EXPERIMENT 4

Design and implementation of IC 7485 5-bit magnitude comparator

Aim: To design and implement 5-bit magnitude comparator using IC 7485.

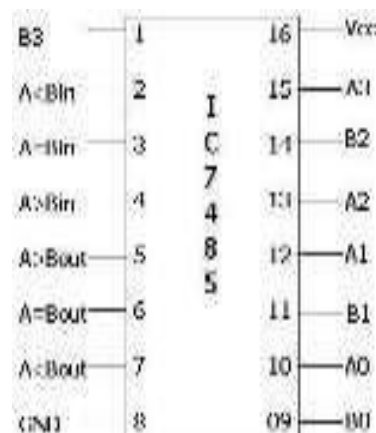
Components required:

Sl.No	Group	Family	Components
1.	TTL	74LS	IC7485
2.	BASIC	SWITCH	DSWPK-4
3.	Sources	Power Sources	VCC
4.	Sources	Power Sources	GND
5.	Indicators	PROBE	PROBE_GREEN,PROBE_RED

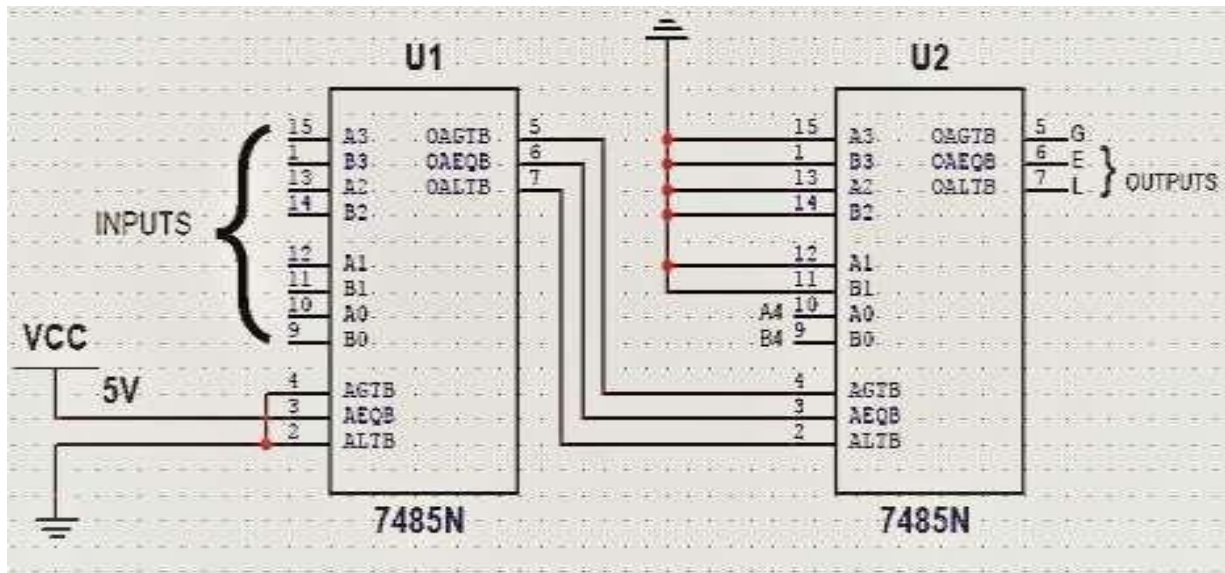
Theory: A **digital comparator** or **magnitude comparator** is a hardware electronic device that takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number. Comparators are used in Central Processing Unitss (CPUs) and Microcontrollers(MCUs).

IC 7485: The IC 7485 is a 4-bit magnitude comparator that can be expanded to almost any length. It compares two 4 bit binary and produces three magnitude results.

Pin Diagram of 7485



Logic diagram of 5-Bit Magnitude Comparator



Truth Table

	Inputs										Outputs		
EX.	A4	A3	A2	A1	A0	B4	B3	B2	B1	B0	A>B	A=B	A<B
1	1	0	0	1	0	0	1	1	0	1	1	0	0
2	1	1	1	1	1	1	1	1	1	0	1	0	0
3	0	1	0	0	1	0	1	0	0	1	0	1	0
4	0	0	1	0	1	1	0	0	0	0	0	0	1

Multisim Simulation

Aim: To realize

(a) Adder and Subtractor using IC 74153.

(b) 3-variable function using IC 74151(8:1 MUX).

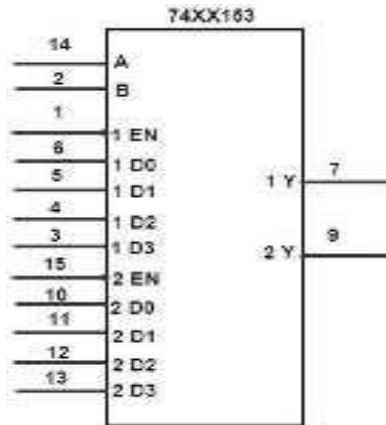
Components:

Sl. No	Group	Family	Components
1.	TTL	74STD	IC74153
2.	TTL	74STD	IC74151(8:1 MUX)
3.	Sources	Power Sources	VCC
4.	Sources	Power Sources	GND
5.	Indicators	PROBE	PROBE_GREEN, PROBE_RED
6.	Sources	DIGITAL_SOURCES	INTERACTIVE_DIGITAL_CONSTANT

Theory: A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input lines and n selection lines whose bit combinations determine which input is selected. An electronic multiplexer can be considered as a multiple-input, single-output switch. A multiplexer is also called a **data selector**.

Conversely, a **demultiplexer** (or **demux**) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A demultiplexer has one data input, 2^n select lines, and n output lines. A demultiplexer is also called a **data distributor**.

Pin Diagram:



Truth Table

Multiplexer (4:1)

S1	S0	I0	I1	I2	I3	Y
0	0	I0	X	X	X	I0
0	1	X	I1	X	X	I1
1	0	X	X	I2	X	I2
1	1	X	X	X	I3	I3

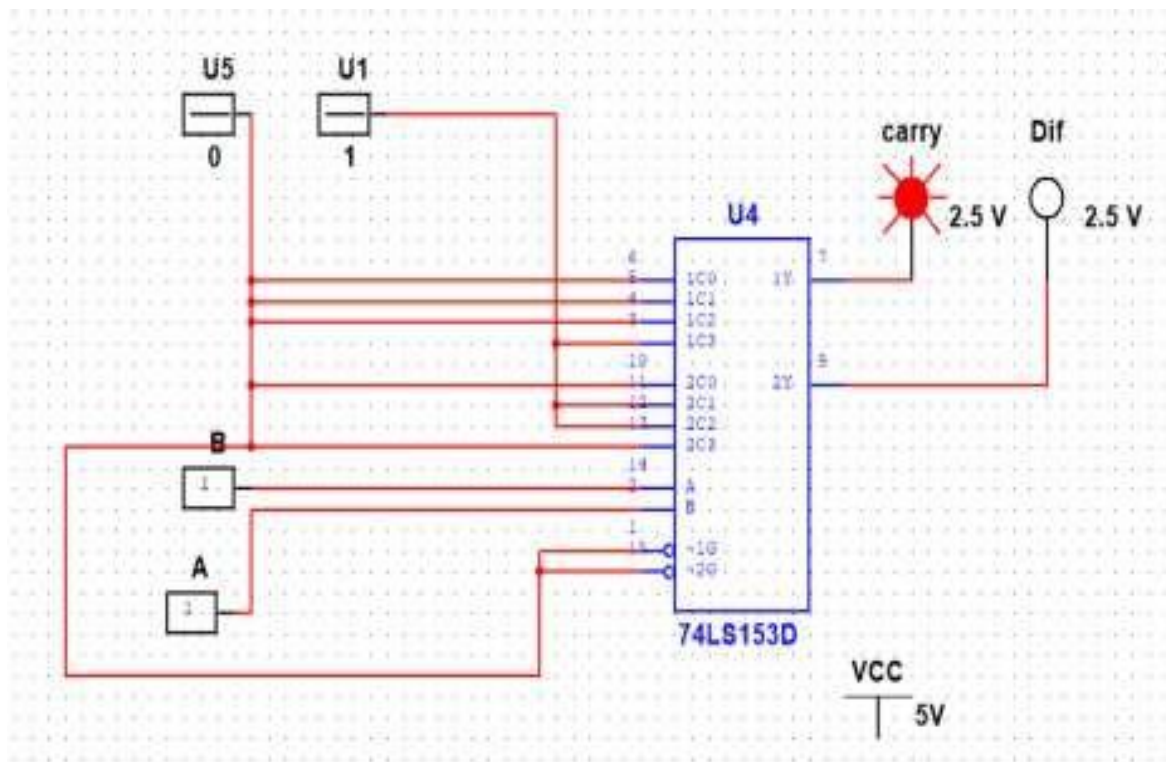
Half Adder

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

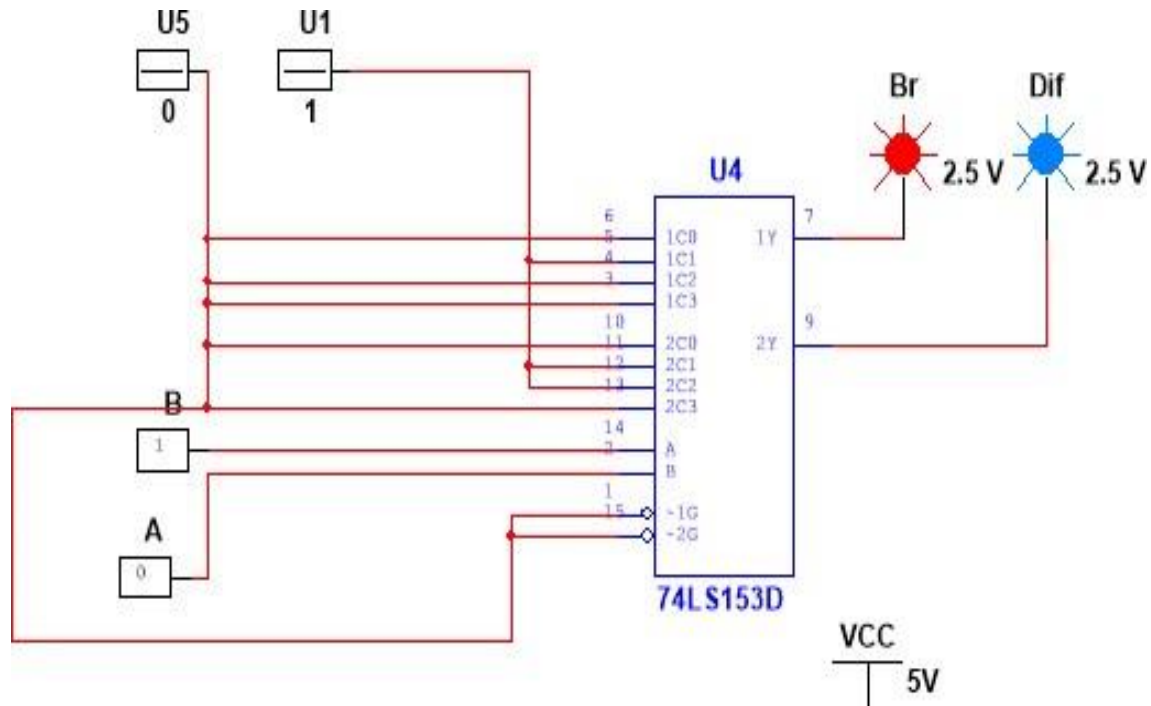
Half Subtractor

A	B	Difference (D)	Borrow(B ₀)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Half Adder realization using using Multisim



Half Subtractor realization using Multisim



Full Adder realization using 74153

Truth Table

Inputs			Outputs	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

DESIGN:

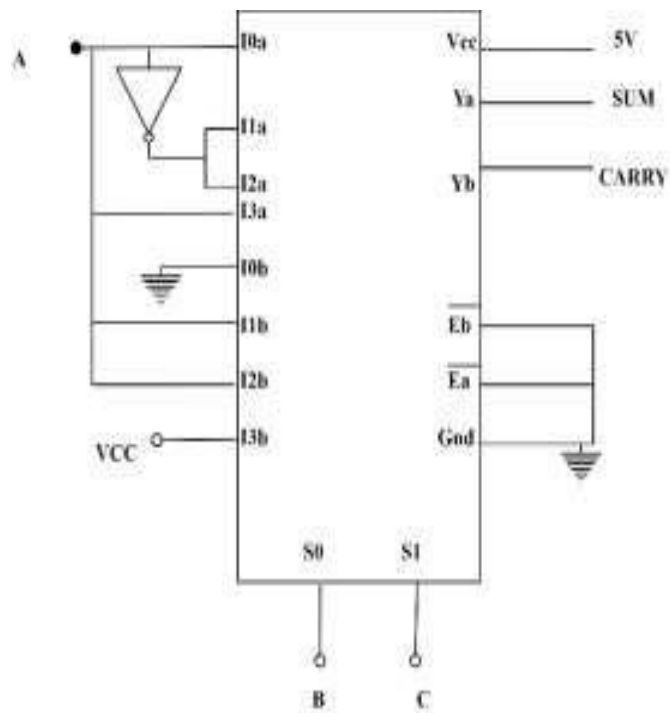
SUM

CARRY

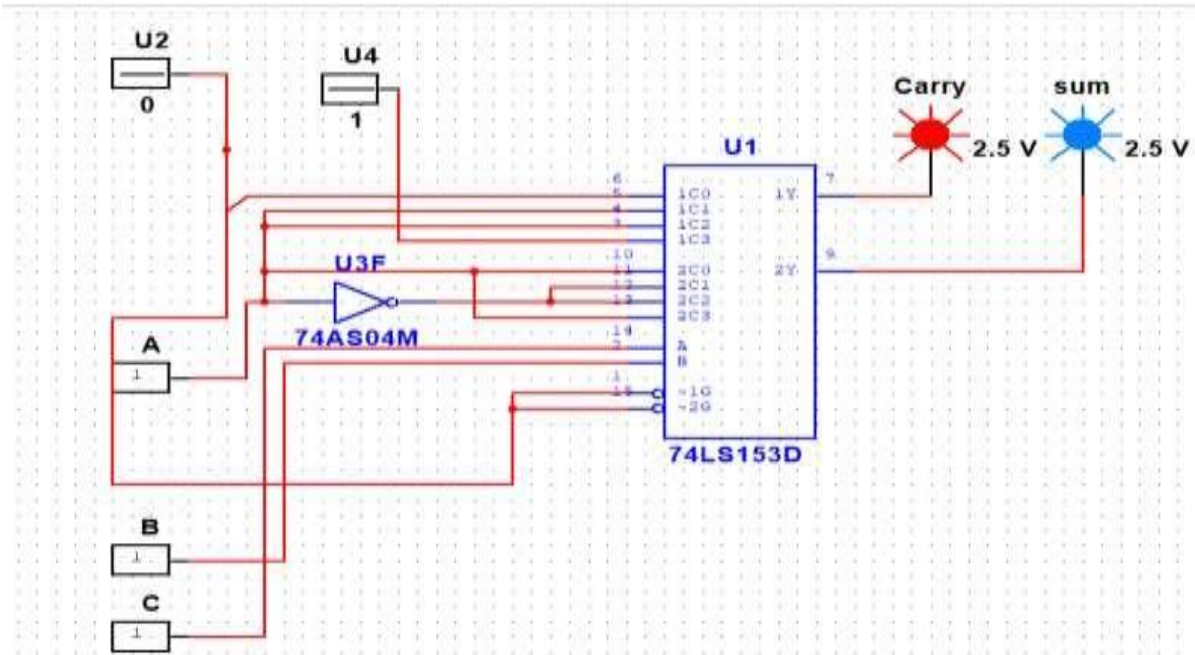
I0	I1	I2	I3
0	1	2	3
4	5	6	7
A	A'	A'	A

I0	I1	I2	I3
0	1	2	3
4	5	6	7
0	A	A	1

FULL ADDER CIRCUIT



Multisim Simulation



Full Subtractor using 74153

Inputs			Outputs	
A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

DESIGN:

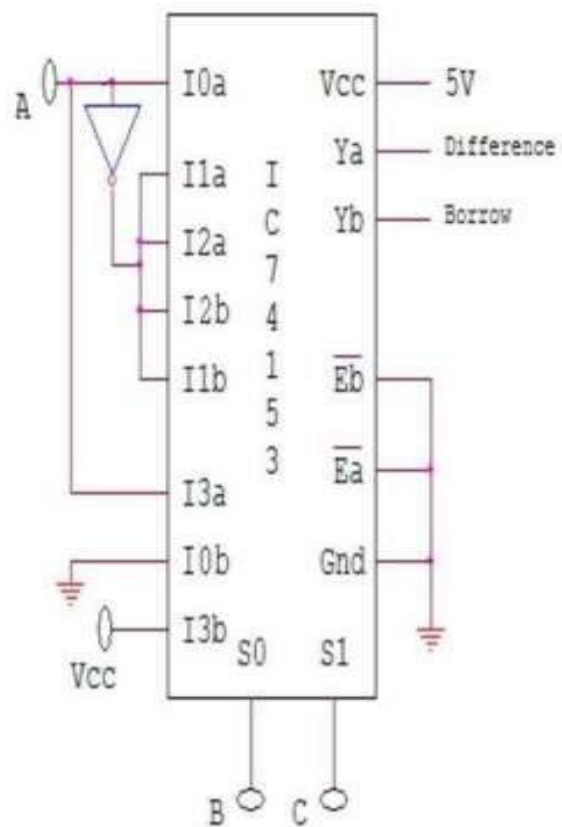
DIFFERENCE

I0	I1	I2	I3
0	1	2	3
4	5	6	7
A	A'	A'	A

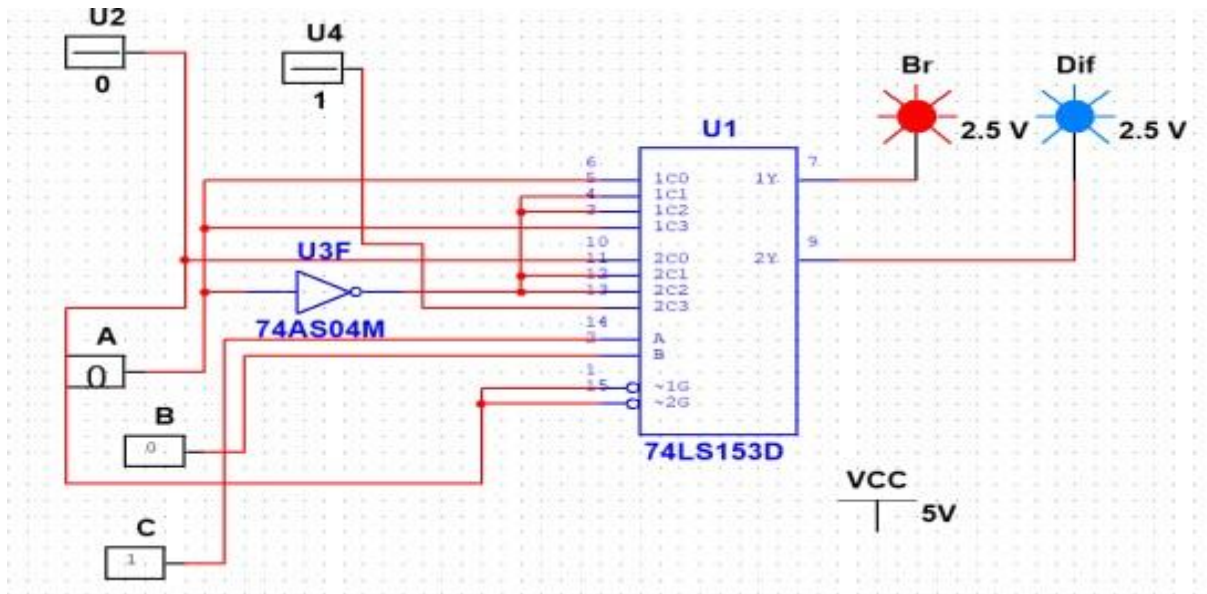
BORROW

I0	I1	I2	I3
0	1	2	3
4	5	6	7
0	A'	A'	1

FULL SUBTRACTOR CIRCUIT



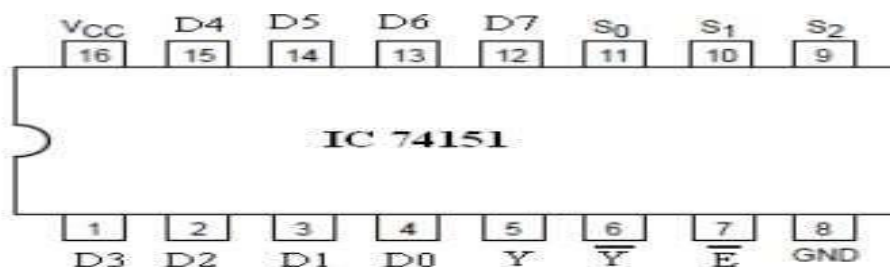
Full Subtractor using Multisim Simulation



Component Required:

Sl.No	Group	Family	Component
1.	TTL	74STD	74LS153, 7400
2.	Sources	DIGITAL_SOURCES	INTERACTIVE_DIGITAL_CONSTANT DIGITAL_CONSTANT
3.	Sources	POWER_SOURCES	VCC, GROUND
4.	Indicators	PROBE	PROBE_GREENN, PROBE_RED

4- Variable Boolean function using IC 74151(8:1 MUX)



7 4151 Pin diagram

Truth Table 8:1 Mux

Inputs	Output
--------	--------


E	S2	S1	S0	I0	I1	I2	I3	I4	I5	I6	I7	Y
1	X	X	X	X	X	X	X	X	X	X	X	0
0	0	0	0	1/0	X	X	X	X	X	X	X	I0
0	0	0	1	X	1/0	X	X	X	X	X	X	I1
0	0	1	0	X	X	1/0	X	X	X	X	X	I2
0	0	1	1	X	X	X	1/0	X	X	X	X	I3
0	1	0	0	X	X	X	X	1/0	X	X	X	I4
0	1	0	1	X	X	X	X	X	1/0	X	X	I5
0	0	1	1	X	X	X	X	X	X	1/0	X	I6
0	1	1	1	X	X	X	X	X	X	X	1/0	I7

Implement the following Boolean function using 8:1 multiplexer.

$$F(A, B, C, D) = \sum m(0, 2, 4, 5, 7, 10, 12, 14, 15)$$

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

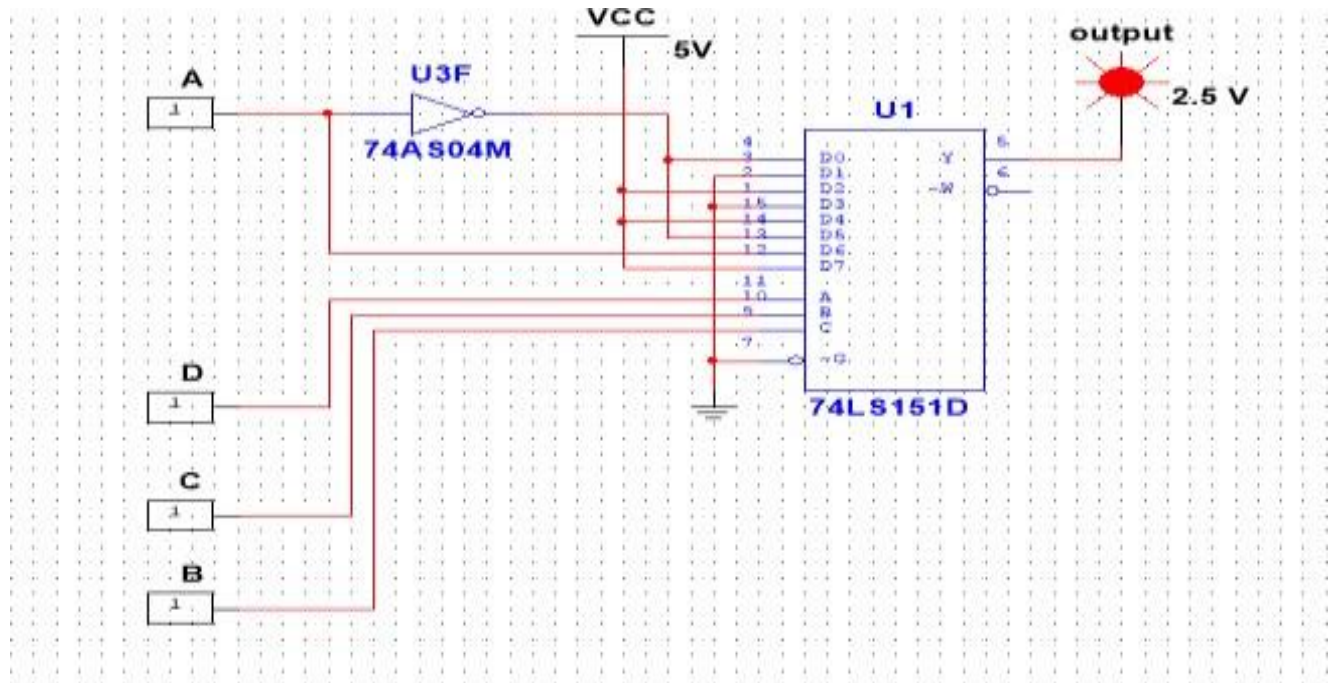
Design Equation of MUX Table



1	0	1	0	1	1	0	1
0	0	1	0	1	0	1	1
A	0	1	0	1	A	A	1

BCD 000 001 010 011
100 101 110 111 A 0 1

**Multisim Circuit
Simulation**



Result:

Adders and Subtractor circuit is simulated and verified using MUX

EXPERIMENT 6

Realization of Adder and Subtractor using IC 74139/ 74155N (Demux/Decoder) and Binary to Gray code conversion & vice versa using 74139/ 74155

Aim: Realize

- (i) Adders & Subtractors using IC74139.
- (ii) Binary to Gray code conversion & vice-versa (74139)

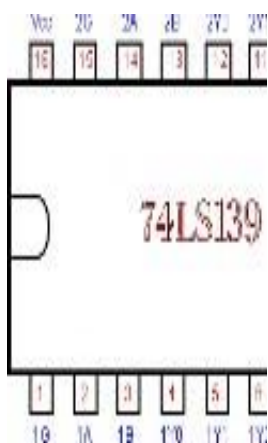
Components Required:

Sl.No	Group	Family	Component
1.	TTL	74STD	74LS139,74LS00

2.	Sources	DIGITAL_SOURCES	INTERACTIVE_DIGITAL_CONSTANT
3.	Sources	POWER_SOURCES	VCC, GROUND
4.	Indicators	PROBE	PROBE_GREEN, PROBE_RED

IC 74139 DEMUX/ DECODER

Pin Diagram:



Truth table:

Inputs			Outputs			
G	B	A	Y0	Y1	Y2	Y3
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

Theory: A Demultiplexer is a circuit that receives information from a single line and directs it to one of 2^n possible output lines. The selection of a specific output is controlled by the bit combination of n selection lines.

In 1:4 demultiplexer, Din is taken as a data input line and sel(0) and sel(1) are taken as the selection lines. The single input variable Din has a path to all four outputs, but the input information is directed to only one of the output lines, as specified by the binary combination of the 2 selection lines

Realization of Half ADDER/ Half Subtractor Using IC

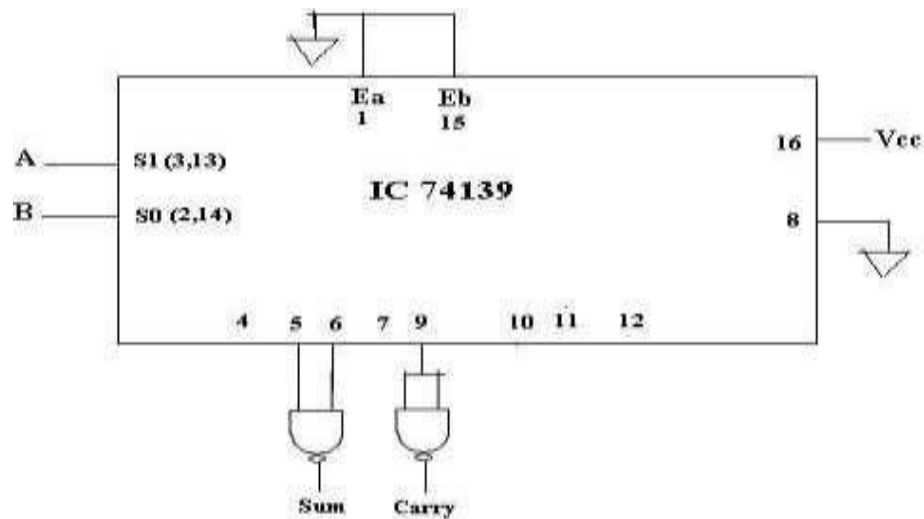
A	B	Sum	Carry	Diff	Bout
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0

1	1	0	1	0	0
---	---	---	---	---	---

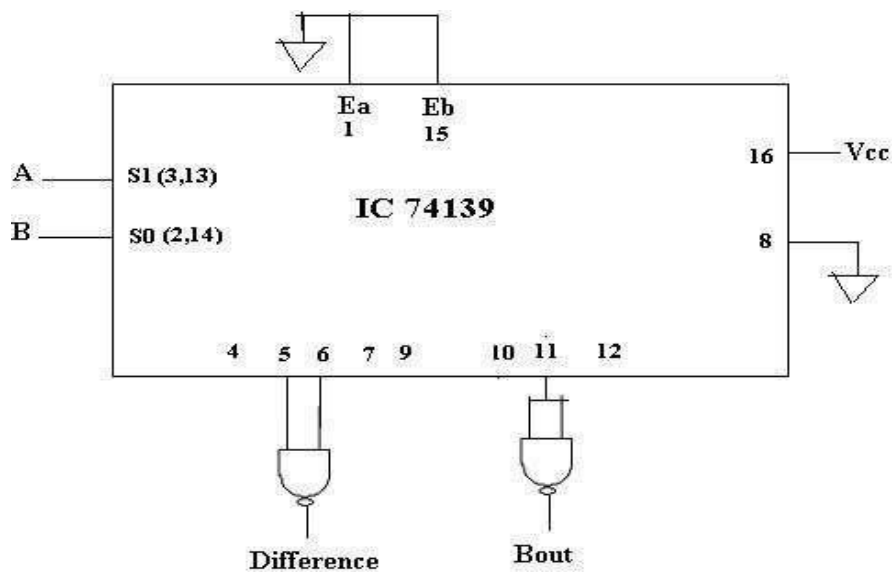
Sum: $\Sigma 1,2$ Carry: $\Sigma 3$

Diff: $\Sigma 1,2$ Bout: $\Sigma 1$

Logic Diagram for Half Adder



Logic Diagram for Half Subtractor

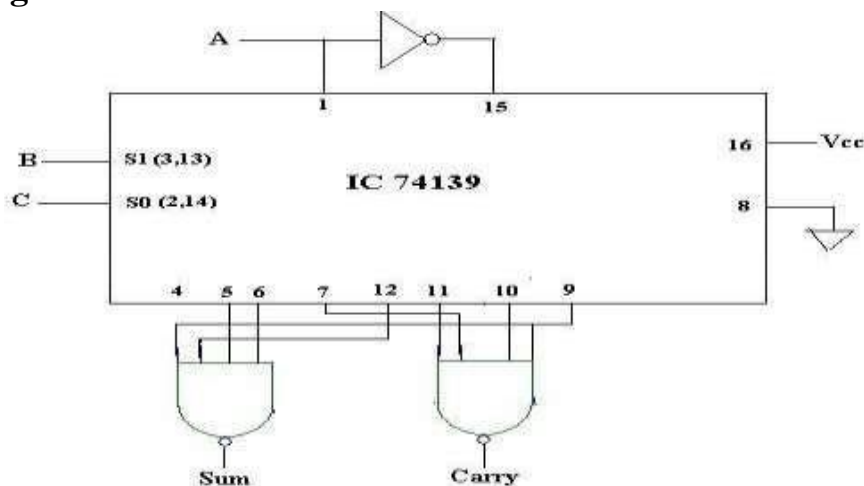


Components Required:

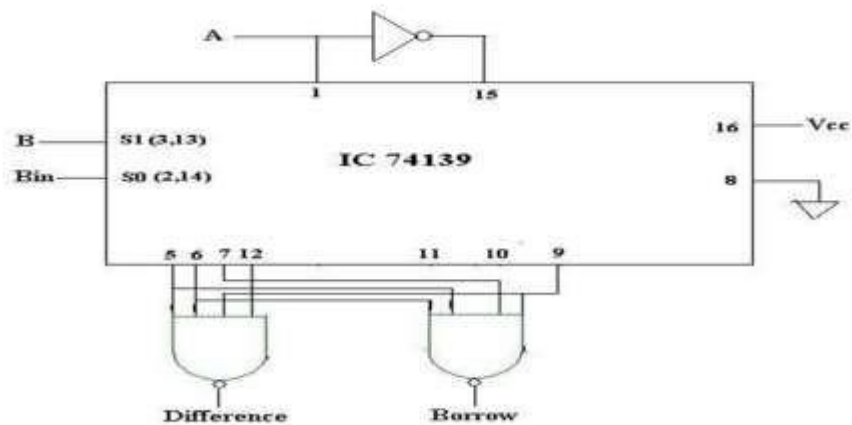
Sl.No	Group	Family	Component
1.	TTL	74STD	74LS139,74LS20
2.	Sources	DIGITAL_SOURCES	INTERACTIVE_DIGITAL_CONSTANT
3.	Sources	POWER_SOURCES	VCC, GROUND
4.	Indicators	PROBE	PROBE_GREEN, PROBE_RED

Realization of Full Adder/ Full Subtractor Using IC

Logic Diagram for Full Adder



Logic Diagram for full Subtractor using IC 74139

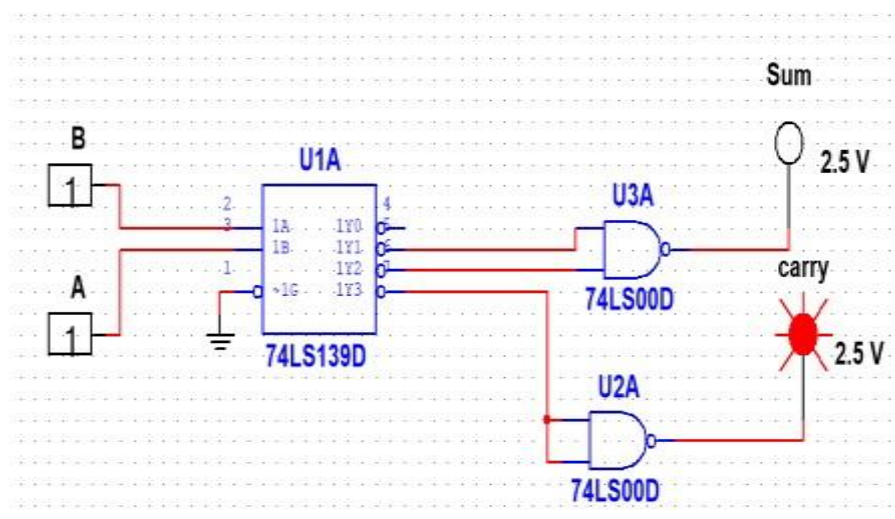


Truth Table:

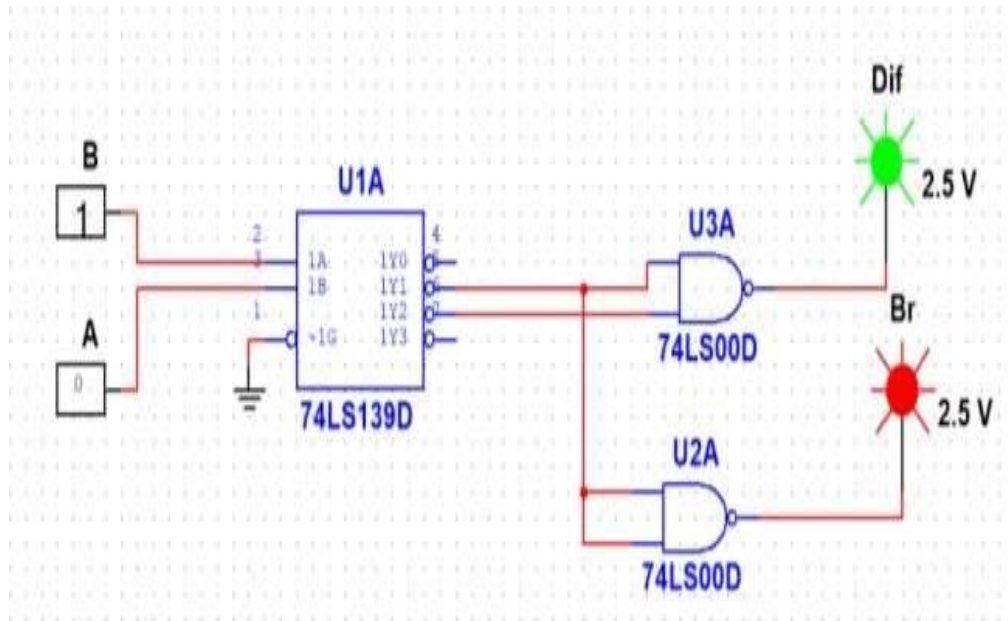
A	B	Cin/Bin	Sum	Cout	Difference	Borrow out
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Multisim Simulation

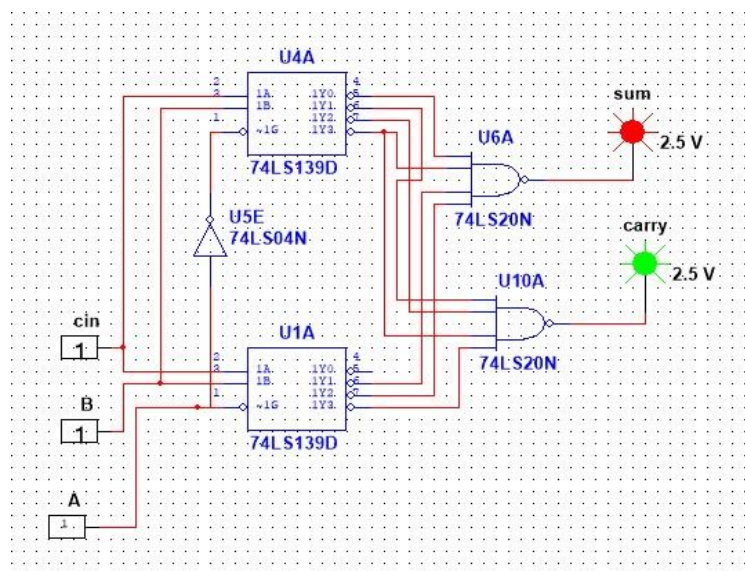
1. Half Adder



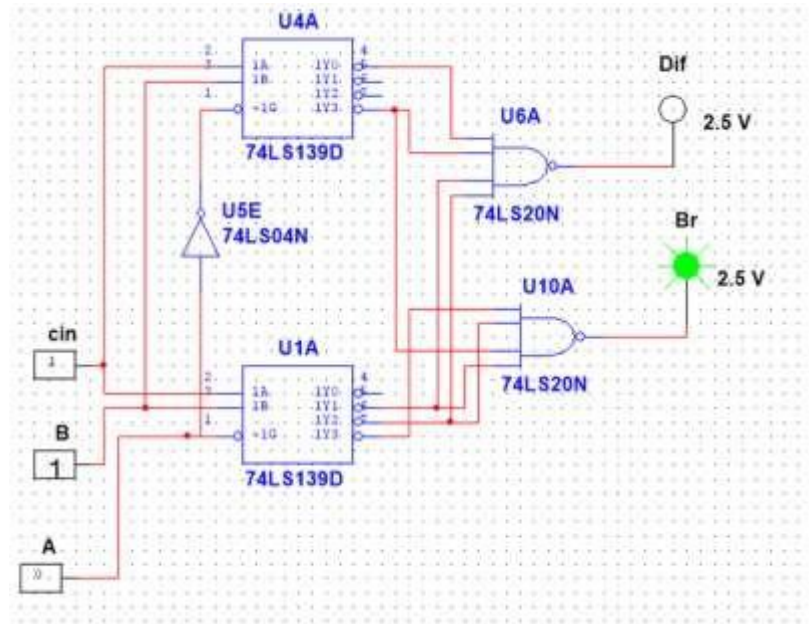
2. Half Subtractor



3. Full Adder



4. Full Subtractor



Aim: Realize Binary to Gray code conversion and vice versa using IC74154(2-4 decoder).

Component required:

Sl.No	Group	Family	component
1.	TTL	74STD	74LS154,74LS30
2.	Sources	DIGITAL_SOURCES	INTERACTIVE_DIGITAL_CONSTANT
3.	Sources	POWER_SOURCES	VCC, GROUND
4.	Indicators	PROBE	PROBE_GREENN, PROBE_RED

Theory:

Binary to gray code conversion is a very simple process. There are several steps to do this types of conversions. Steps given below elaborate on the idea on this type of conversion.

The M.S.B. of the gray code will be exactly equal to the first bit of the given binary number.

Now the second bit of the code will be exclusive-or of the first and second bit of the given binary number, i.e if both the bits are same the result will be 0 and if they are different the result will be 1.

The third bit of gray code will be equal to the exclusive -or of the second and third bit of the given binary number. Thus the **Binary to gray code conversion** goes on. One example given below can make your idea clear on this type of conversion.

Truth Table: Binary to Gray code conversion

Decimal	Binary Inputs				Gray Outputs			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0

12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

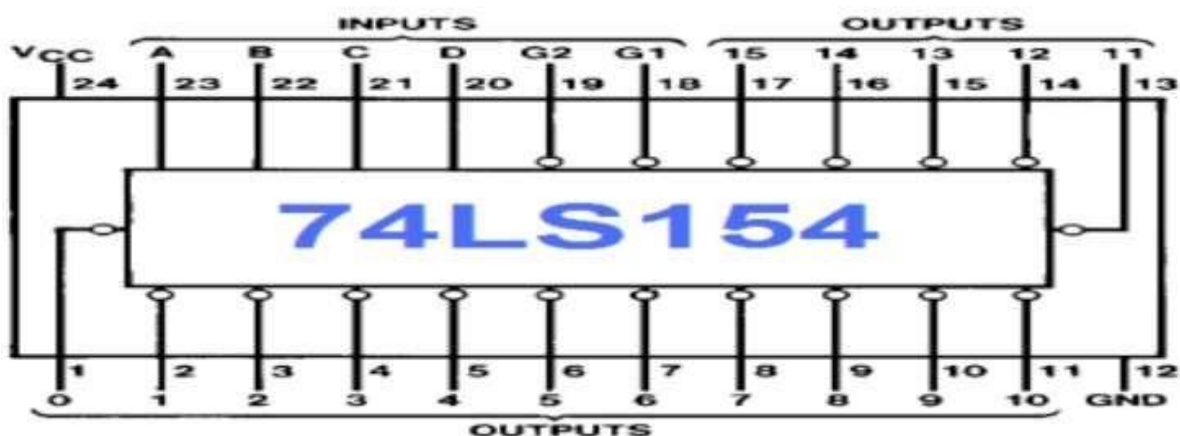
Design Equations:

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

$$G1 = \sum m(2,3,4,5,10,11,12,13) \quad G0 = \sum m(1,2,5,6,9,10,13,15)$$

74154 Pin diagram, Truth table and Logic symbol:

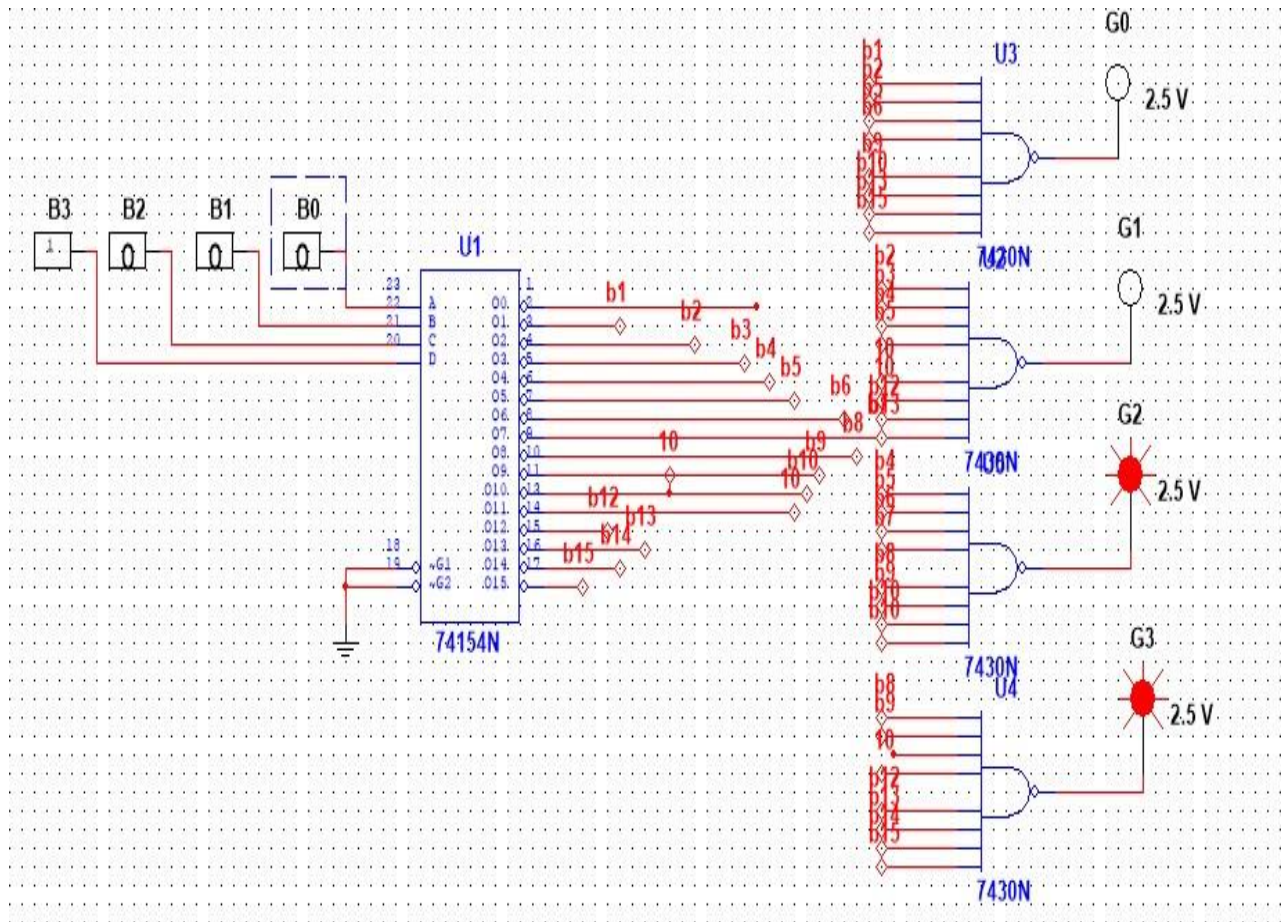
4-line-to-16-line decoders utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, G1 and G2, are low. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. These DE multiplexers are ideally suited for implementing high-performance memory decoders.



Inputs						Outputs															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Inputs						Outputs															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Multisim Binary to Gray Circuit Simulation



Gray code to binary conversion

Following steps can make your idea clear on this type of conversions.

- The M.S.B of the binary number will be equal to the M.S.B of the given gray code.
- Now if the second gray bit is 0 the second binary bit will be same as the previous or the first bit. If the gray bit is 1 the second binary bit will alter. If it was 1 it will be 0 and if it was 0 it will be 1.
- This step is continued for all the bits to do **Gray code to binary conversion**

Truth Table: Gray to Binary code conversion

Decimal	Gray Inputs				Binary Outputs			
	G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	0
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	0	1	0

Design Equations:

$$B3 = \Sigma m (8,9,10,11,12,13,14,15)$$

$$B2 = \Sigma m (4,5,6,7,8,9,10,11)$$

$$B1 = \Sigma m (2,3,4,5,8,9,14,15)$$

$$B0 = \Sigma m (1,2,4,7,8,11,13,14)$$

Multisim Gray to Binary Circuit Diagram

Note: The multi sim circuit can designed similar to Binary to Gray with relevant expressions.

Procedure:

1. Place the components as tabulated in component list.
2. Rig up the connections as shown in the circuit diagram.
3. Simulate the design. 4. Verify with respect to the truth table given.

Result:

Realised Adder and subtractor using IC74139

Realised Binary to Gray and Vice versa using IC 74154

EXPERIMENT 7

SR, Master-Slave JK, D & T flip-flops using NAND Gates using Pspice/Multisim.

Aim: To realize SR, Master-Slave JK, D and T Flip-Flops using NAND Gates

Components:

SR Flip Flop:

Sl No.	Group	Family	Components	Quantity
1.	TTL	74LS	IC7400	4
2.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	2
3.			Multisim Software	

Master-Slave JK Flip Flop:

Sl No.	Group	Family	Components	Quantity
1.	TTL	74LS	IC7400	4
2.	TTL	74LS	IC7410	5
3.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	2
4.			Multisim Software	

D Flip Flop:

Sl No.	Group	Family	Components	Quantity
1.	TTL	74LS	IC7400	4
2.	TTL	74LS	IC7410	6
3.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	2
4.			Multisim Software	

T Flip Flop:

Sl No.	Group	Family	Components	Quantity
1.	TTL	74LS	IC7400	4
2.	TTL	74LS	IC7410	5
3.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	2
4.			Multisim Software	

Theory:

Basically Flip-Flops are the bistable multivibrators that stores logic 1 and logic 0. Shift registers, memory, and counters are built by using Flip – Flops. Any complex sequential machines are build using Flip – Flops. Sequential circuit (machine) output depends on the present state and input applied at that instant.

The SR flip-flop, also known as a *SR Latch*, can be considered as one of the most basic sequential logic circuit possible. This simple flip-flop is basically a one-bit memory bi-stable device that has two inputs, one which will

“SET” the device (meaning the output = “1”), and is labeled S and another which will “RESET” the device (meaning the output = “0”), labeled R.

Then the SR description 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.

JK flip Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same “Set” and “Reset” inputs. The difference this time is that the “JK flip flop” has no invalid or forbidden input states of the SR Latch even when S and R are both at logic “1”.

The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”. Due to this additional clocked input, a JK flip-flop has four possible input combinations, “logic 1”, “logic 0”, “no change” and “toggle”.

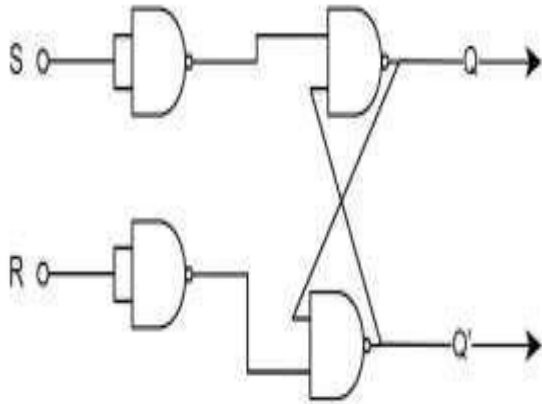
A **D flip flop** has a single data input. This type of FF is obtained from the SR FF by connecting the R input through an inverter, and the S input is connected directly to data input. The modified clocked SR flip-flop is known as Dflip-flop and is shown below. From the truth table of SR flip-flop we see that the output of the SR flip-flop is in unpredictable state when the inputs are same and high.

T flip-flop is known as toggle flip-flop. The T flip-flop is modification of the J-K flip-flop. Both the JK inputs of the JK flip – flop are held at logic 1 and the clock signal continuous to change.

Logic Diagram:

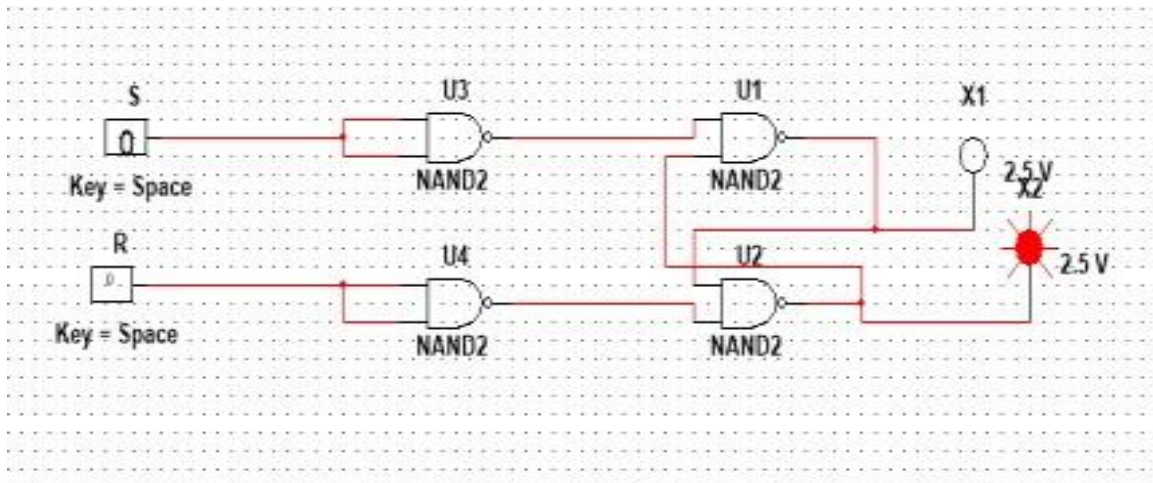
SR Flip Flop

Truth Table

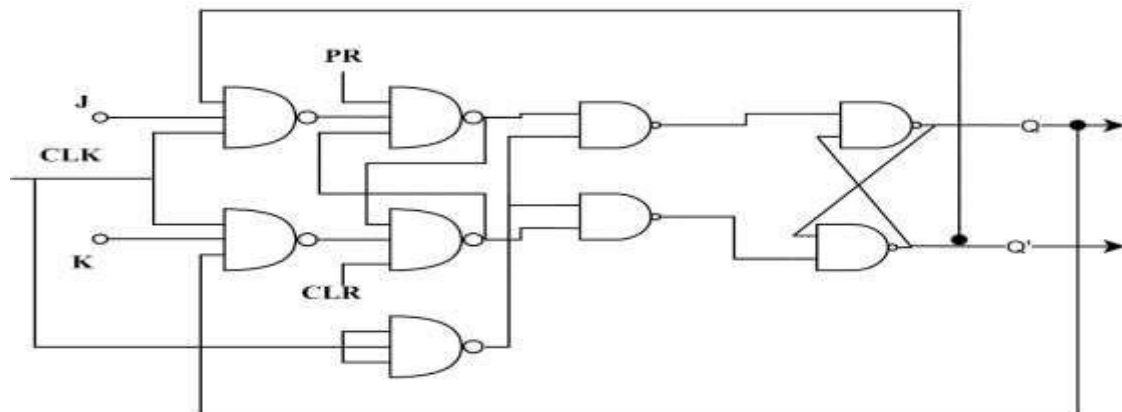


Inputs		Outputs		Comments
S	R	Q	Q_bar	
0	0	0	0	Previous State
0	0	1	1	Previous State
0	1	0	0	Previous State
0	1	1	0	Set
1	0	0	1	Reset
1	0	1	1	Previous State
1	1	0	-	Indeterminate
1	1	1	-	Indeterminate

Multisim Circuit Diagram:



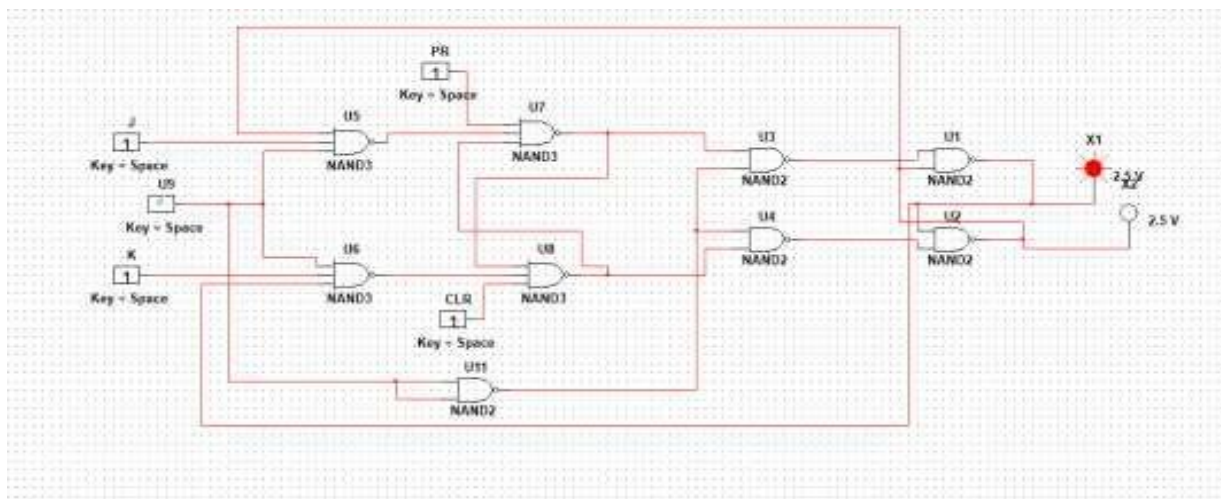
Master Slave JK Flip Flop



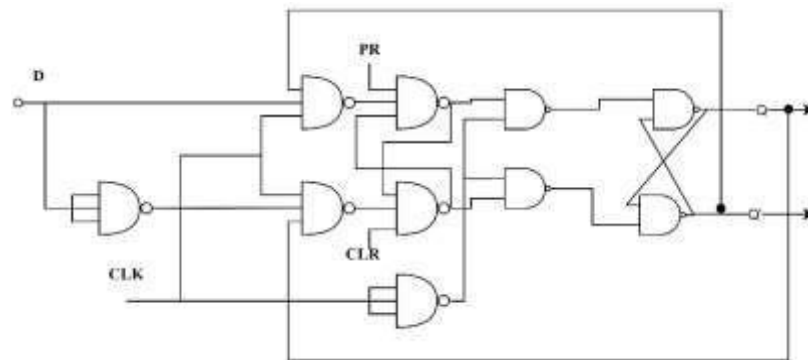
Truth Table

PR	CLR	Inputs			Outputs		Comments
		J	K	Clock	Q _{n+1}	Q _{bar_{n+1}}	
0	1	X	X	X	1	0	Set
1	0	X	X	X	0	1	Reset
1	1	0	0	⌊	Q _n	Q _{bar_n}	No Change
1	1	0	1	⌊	0	1	Reset
1	1	1	0	⌊	1	0	Set
1	1	1	1	⌊	Q _{bar_n}	Q _n	Toggle

Multisim Circuit Diagram:



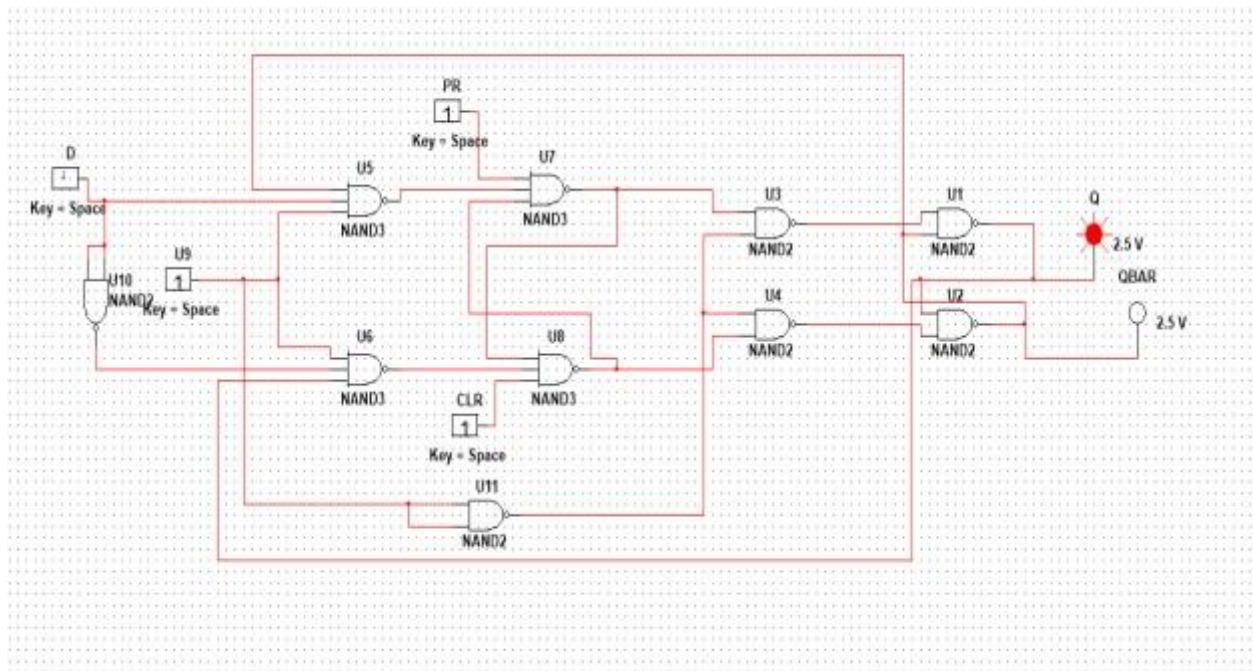
D Flip-Flop:



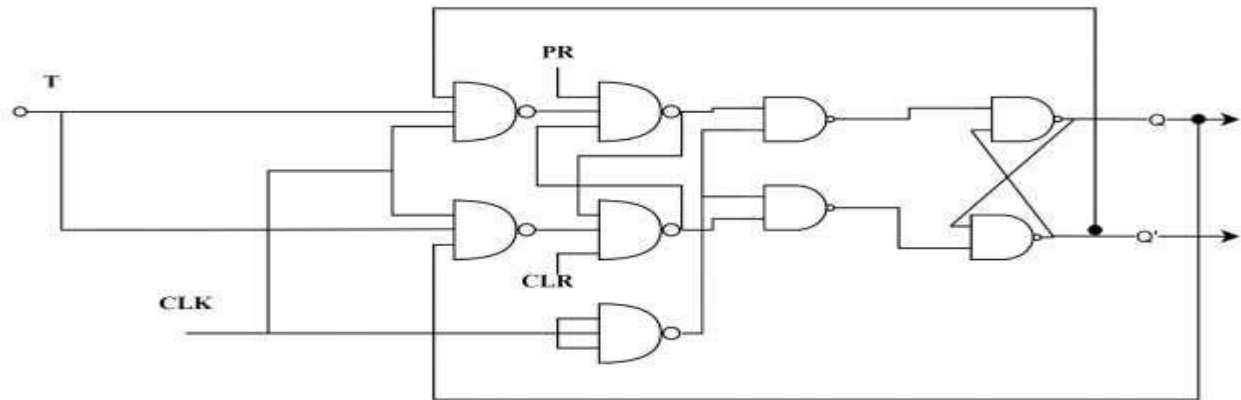
Truth Table

PR	CLR	Inputs		Outputs	
		D	Clock	Q _{n+1}	Q _{bar} _{n+1}
1	1	0		0	1
1	1	1		1	0

Multisim Circuit Diagram:



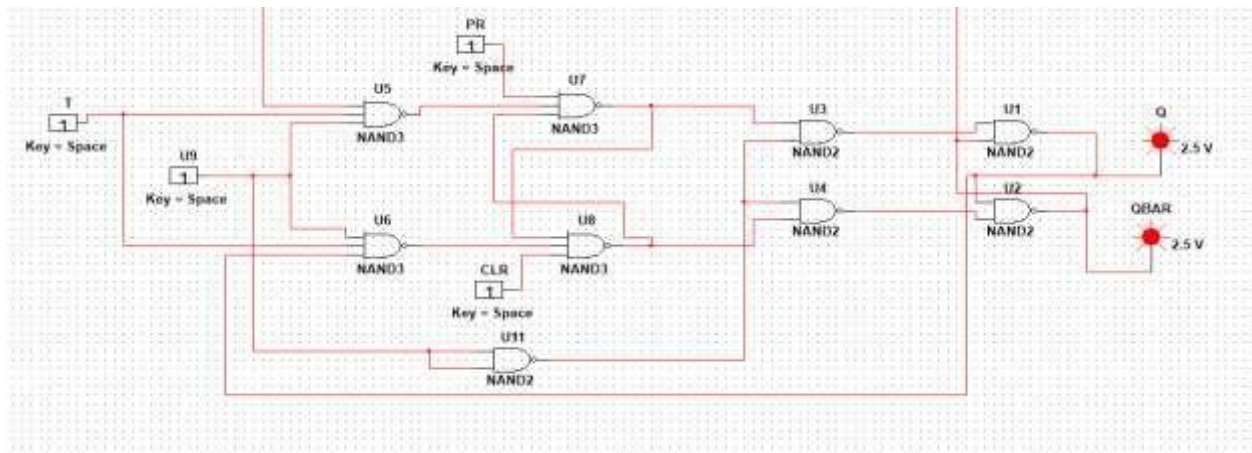
T-Flip Flop:



Truth Table:

PR	CLR	Inputs		Outputs	
		T	Clock	Q _{n+1}	Q _{bar_{n+1}}
1	1	0	⌋	Q _n	Q _{bar_n}
1	1	1	⌋	Q _{bar_n}	Q _n

Multisim Circuit Diagram:



Result: Realized SR Flip flop, Master-Slave JK Flip-Flop, D-Flip flop and T-Flip flop verified using NAND gates

EXPERIMENT 8

Design and realize the Synchronous counters (up/down decade/binary) using PSpice/Multisim

Aim: To Design Synchronous Counters using IC7490 (Decade) and IC74192 (Up/down binary)

Components:

Sl No.	Group	Family	Components	Quantity
1.	TTL	74LS	IC7490	4
2.	TTL	74LS	IC74192	
3.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	2
4.			Multisim Software	

Decade Counter using IC7490

Theory: Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. If the "clock" pulses are applied to all the flip-flops in a counter simultaneously, then such a counter is called as synchronous counter.

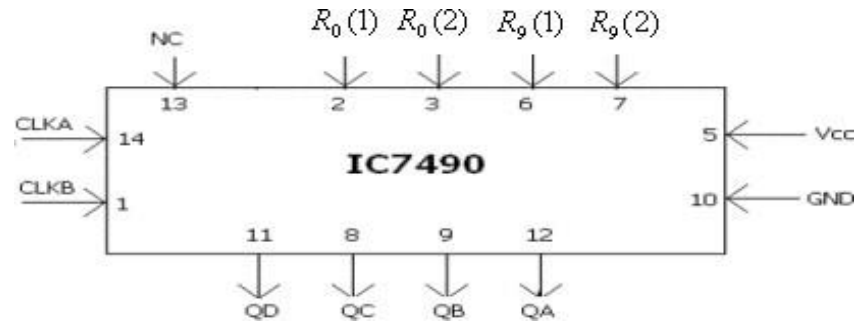
Modulus Counter (MOD-N Counter)- A 2-bit ripple counter is called as MOD-4 counter and 3-bit ripple counter is called as MOD-8 counter. So in general, an n-bit ripple counter is called as modulo-N counter. Where, MOD number = 2^n .

To realize Mod-N Counter using IC7490 / 7476

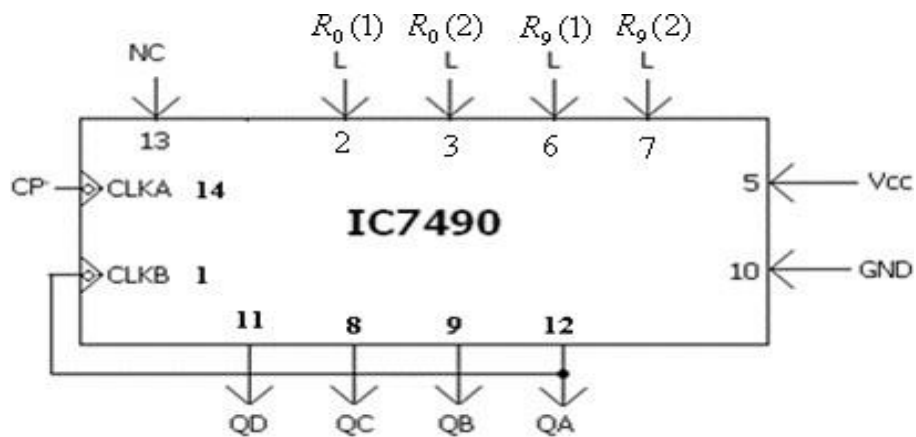
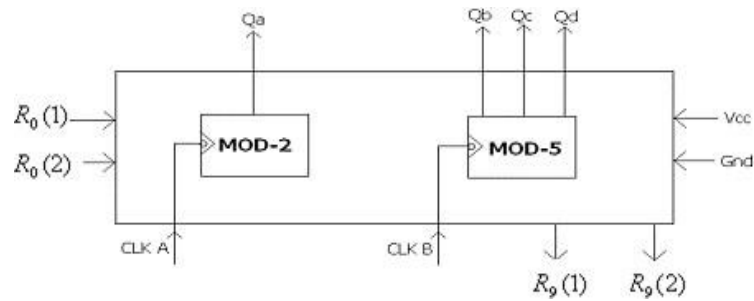
The 2-bit ripple counter is called as MOD-4 counter and 3-bit ripple counter is called as MOD-8 counter. So in general, an n-bit ripple counter is called as modulo-N counter. Where, MOD number = 2^n . Application of counters are Frequency counters, Digital clock, Time measurement, A to D converter, Frequency divider circuits etc.

The 7490 is a BCD/decade counter that consists of four Master-Slave JK flip-flops internally connected to provide Mod- 2 (Divide by 2) and Mod-5 Counter(Divide by 5). It can count from 0 to 9 cyclically in its natural mode. It counts the input pulses and the output is received as a 4-bit binary number through pins QA, QB, QC and QD. The binary output is reset to 0000 at every tenth pulse and count starts from 0 again. The Mod of the IC 7490 is set by changing the RESET pins R0 (1), R0 (2), R9 (1), R9 (2). It is possible to increase the counting capability of a Decade number by connecting more ICs in series.

Pin Diagram:



Internal Diagram: IC 7490 BCD/Decade Counter



Components:

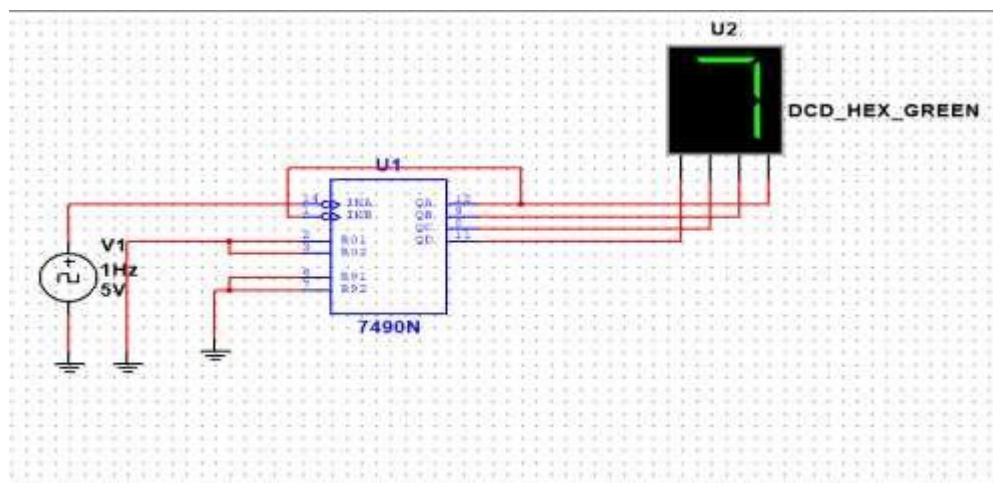
Sl.No	Group	Family	Component
1.	TTL	74STD	7490N

2.	Sources	SIGNAL VOLTAGE SOURCES	CLOCK VOLTAGE
3.	Sources	POWER SOURCES	GROUND
4.	Indicators	HEX_DISPLAY	DCD_HEX

Conditional Table:

R ₀ (1)	R ₀ (2)	R ₉ (1)	R ₉ (2)	Qa	Qb	Qc	Qd
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	L	H	H	1	0	0	1
L	X	L	X	MOD-2 COUNTER			
X	L	X	L	MOD-5 COUNTER			

Multisim Circuit Diagram:



Up/Down Binary Counter using IC74192:

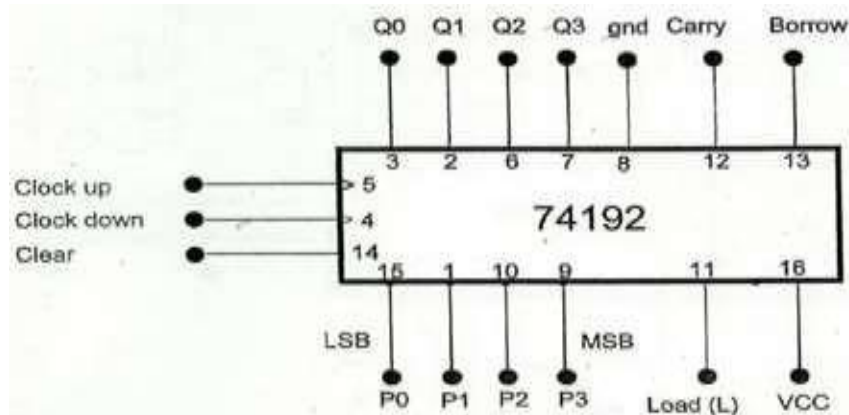
Theory: The 74192 is a Presettable Synchronous 4-Bit Up/Down Decade Counter. Presetting the counter to the number on the preset data inputs (Input A -

Input D) is accomplished by a LOW asynchronous parallel load input (Load). The counter is incremented on the low-to-high transition of the UP input (and a high level on the Clock- DOWN) and decremented on the low to high transition of the DOWN input (and a high level on the UP input).

A high level on the CLR input overrides any other input to clear the counter to its zero state. The Terminal Count up (CO) goes low half a clock period before the zero count is reached and returns to a high level at the zero count.

The Terminal Count Down (BO) in the count down mode likewise goes low half a clock period before the maximum count (9 in the 74192) and returns to high at the maximum count. Cascading is effected by connecting the CO [carry] and BO [borrow] outputs of a less significant counter to the Clock-Up [UP] and Clock-Down [DOWN] inputs, respectively, of the next most significant counter.

Pin Diagram:

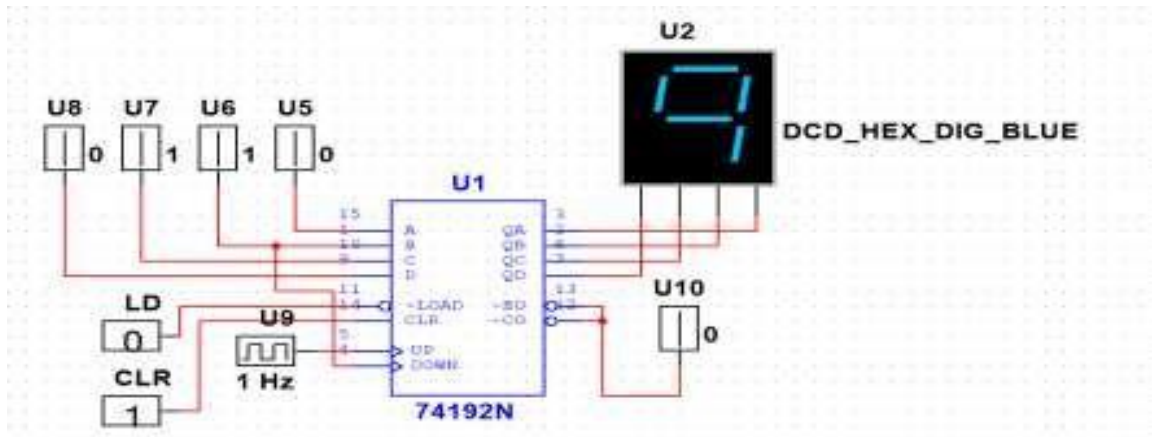


Function Table:

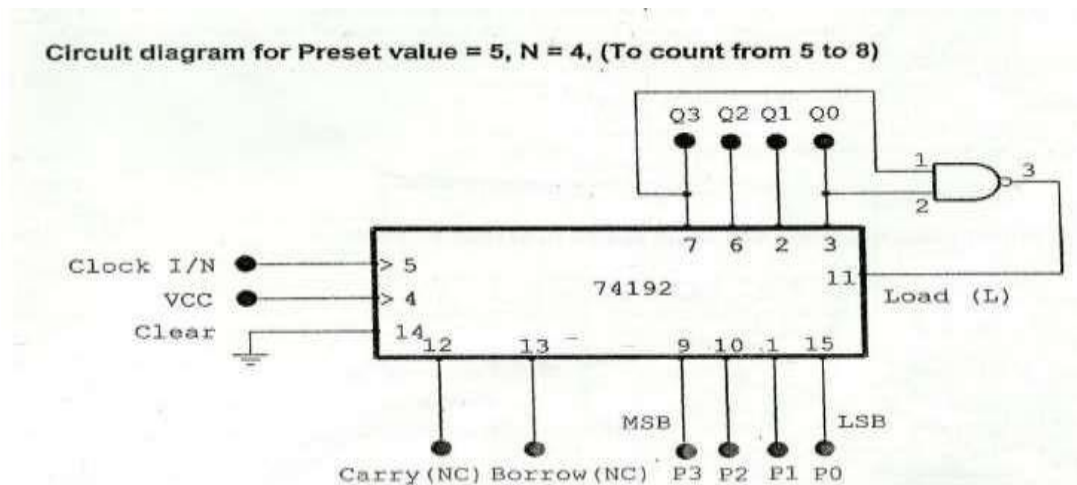
LOAD	CLEAR	CLK_UP	CLK_DOWN	MODE
X	1	X	X	Reset to Zero
1	0	↑	1	Up Count
1	0	1	↑	Down Count

0	0	X	X	Preset
1	0	1	1	Stop Count

Multisim Circuit Diagram:



NOTE: For down count connect **clock** to **pin 4** and **pin 5** to logic '1', Same circuit can be configured to work as **mod-4 up counter** according to the below circuit diagram.

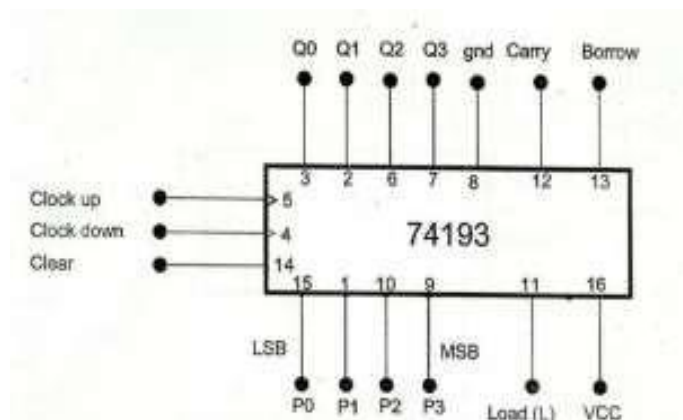


Up/down binary counter using IC74193

Theory: The 74193 is a Presettable Synchronous 4-Bit Up/Down Counter(same as 74192 the only difference is it is mod-15 where as 74192 is of mod-10).

Presetting the counter to the number on the preset data inputs (Input A - Input D) is accomplished by a LOW asynchronous parallel load input (Load). The counter is incremented on the low-to-high transition of the UP input (and a high level on the Clock- DOWN) and decremented on the low to high transition of the DOWN input (and a high level on the UP input). A high level on the CLR input overrides any other input to clear the counter to its zero state. The Terminal Count up (CO) goes low half a clock period before the zero count is reached and returns to a high level at the zero count.

Pin Diagram:



Function Table:

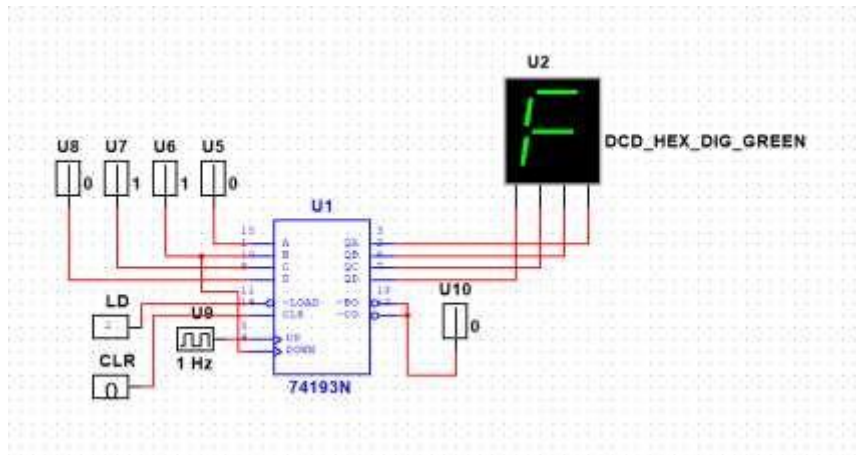
LOAD	CLEAR	CLK_UP	CLK_DOWN	MODE
X	1	X	X	Reset to Zero
1	0	↑	1	Up Count
1	0	1	↑	Down Count
0	0	X	X	Preset
1	0	1	1	Stop Count

Components:

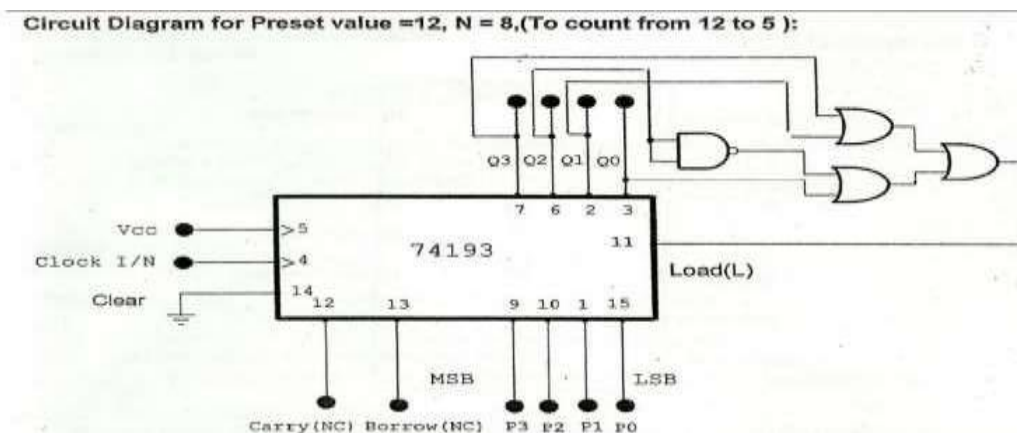
Sl.No	Group	Family	Component
1.	TTL	74STD	74192,74193

2.	Sources	DIGITAL SOURCES	INTERACTIVE_DIGITAL_CONSTANT
3.	Sources	DIGITAL SOURCES	DIGITAL_CLOCK
4.	Indicators	PROBE	DCD_HEX_DIG_GREEN

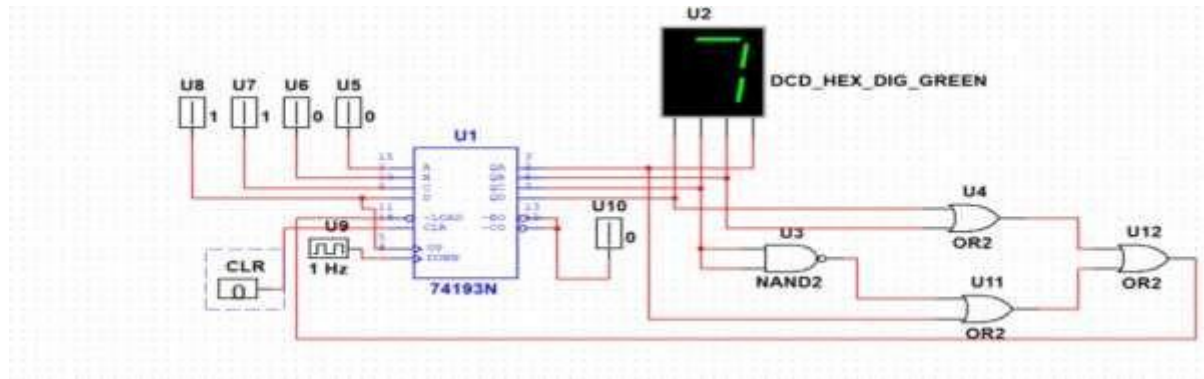
Multisim Circuit Diagram:



Note: For down count connect **clock** to **pin 4** and **pin 5** to logic '1'



Multisim Circuit Diagram:



Procedure:

1. Place the components as tabulated in component list.
2. Rig up the connections as shown in the circuit diagram.
3. Simulate the design.
4. Verify with respect to the truth table given.

EXPERIMENT 9

Realize the shift registers using IC 7474/7495

Aim: To realize the shift registers and their modes (i) SISO (ii) PIPO (iii) PISO (iv) SIPO using IC 7474/7495 using PSpice/Multisim.

Learning Objectives:

1. To illustrate the operation of shift registers
2. To study different shift register configurations

Components:

Sl.No	Group	Family	Component	Quantity
1.	TTL	74STD	74194	1
2.	Sources	DIGITAL SOURCES	INTERACTIVE_DIGITAL_CONSTANT	9
3.	Sources	DIGITAL SOURCES	DIGITAL_CLOCK	1
4.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	4

Theory:

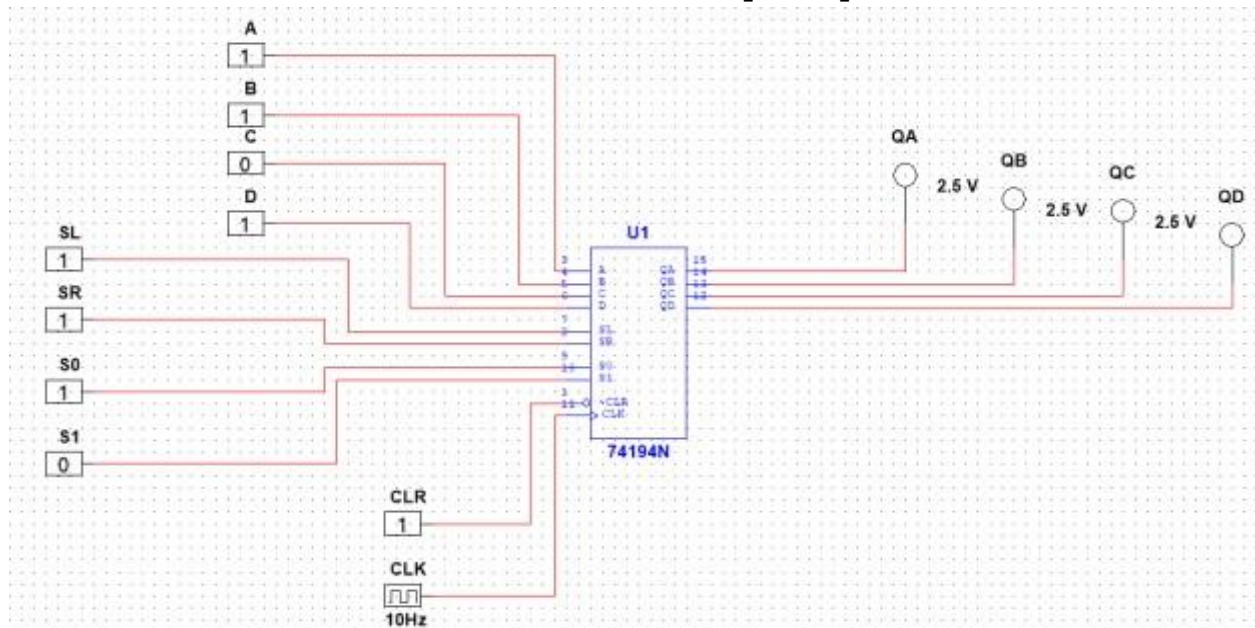
1. Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flipflop becomes the input of the next flipflop. All the flip-flops are driven by a common clock, and all are set or reset simultaneously.
2. Different types of shift register 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, Bidirectional shift register shift register.
3. The serial in/serial out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output also in serial form.
4. The serial in/parallel out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output in parallel form.
5. The parallel in/serial out shift register accepts data in parallel. It produces the stored information on its output also in serial form.
6. The parallel in/parallel out shift register accepts data in parallel. It produces the stored information on its output in parallel form.

7. The primary application of the shift registers is to binary information in a register can be moved from stage to stage within the register or into or out of the register upon application of clock pulses. This type of bit movement or shifting is essential for certain arithmetic and logic operations used in microprocessors.

Multisim Procedure:

1. To Place Components click on Place/Components. On the Select Component window, click on Group-> All groups. Type the components needed for the circuit. Click OK to place the component on the schematic.
2. Place component 74194 IC from the component list.
3. Place the component interactive_digital_constant & Digital_clock from the component list for inputs.
4. Select Probe from component list and place at outputs.
5. Once the components are placed, wire the circuit by clicking on Place/Wire drag and place the wire. Components can also be connected by clicking the mouse over the terminal edge of one component and dragging to the edge of another component.
6. After saving the circuit, click simulation.
7. For different cases of input check the output according to the truth table.

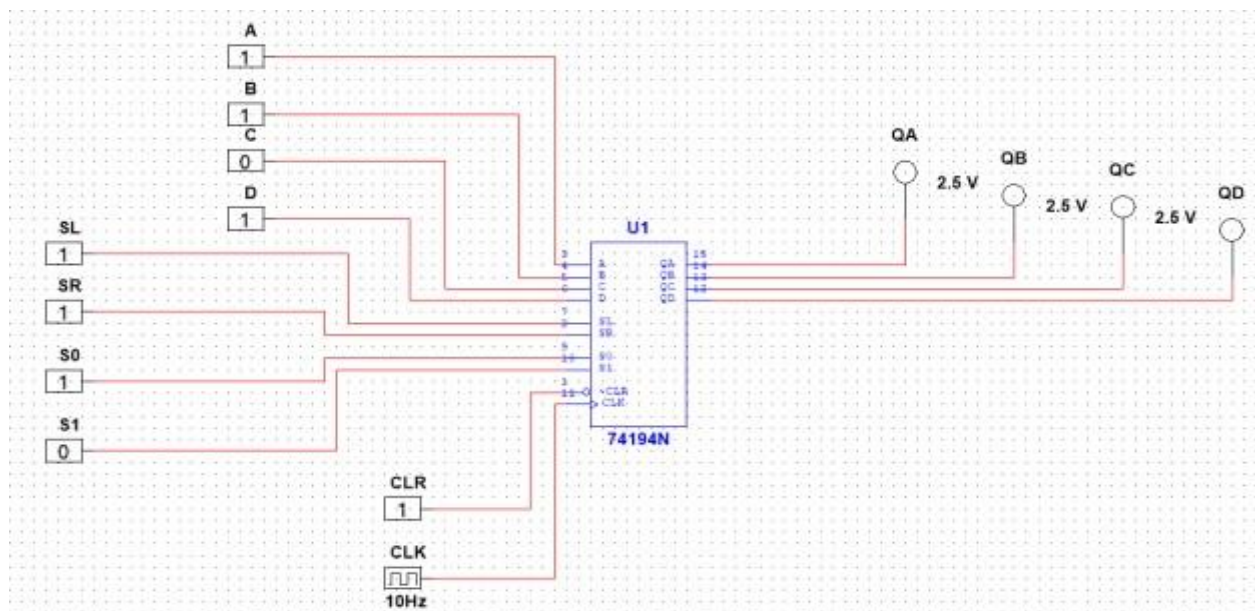
Multisim simulation: Serial In Serial Out [SISO]:



Truth Table:

CLR	SL	SR(Serial Input)	S0	S1	QA Serial Right Shift (Serial Out)
1	1	1	1	0	1
1	1	0	1	0	0
1	1	1	1	0	1
1	1	0	1	0	0

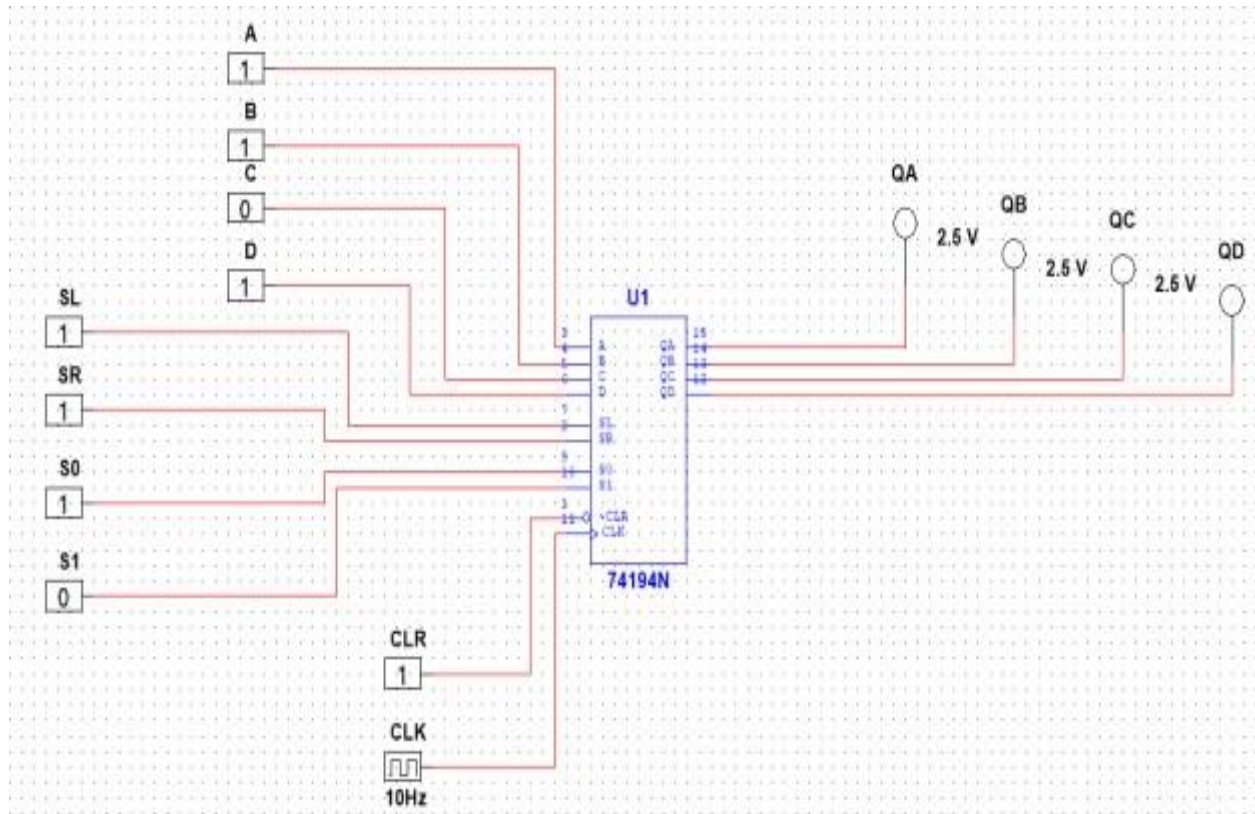
Parallel In Parallel Out [PIPO]:



Truth Table:

CLR	SL	SR(Serial Input)	S0	S1	A	B	C	D	QA	QB	QC	QD
1	X	X	1	1	1	0	1	0	1	0	1	0
1	X	X	1	1	1	1	1	0	1	1	1	0

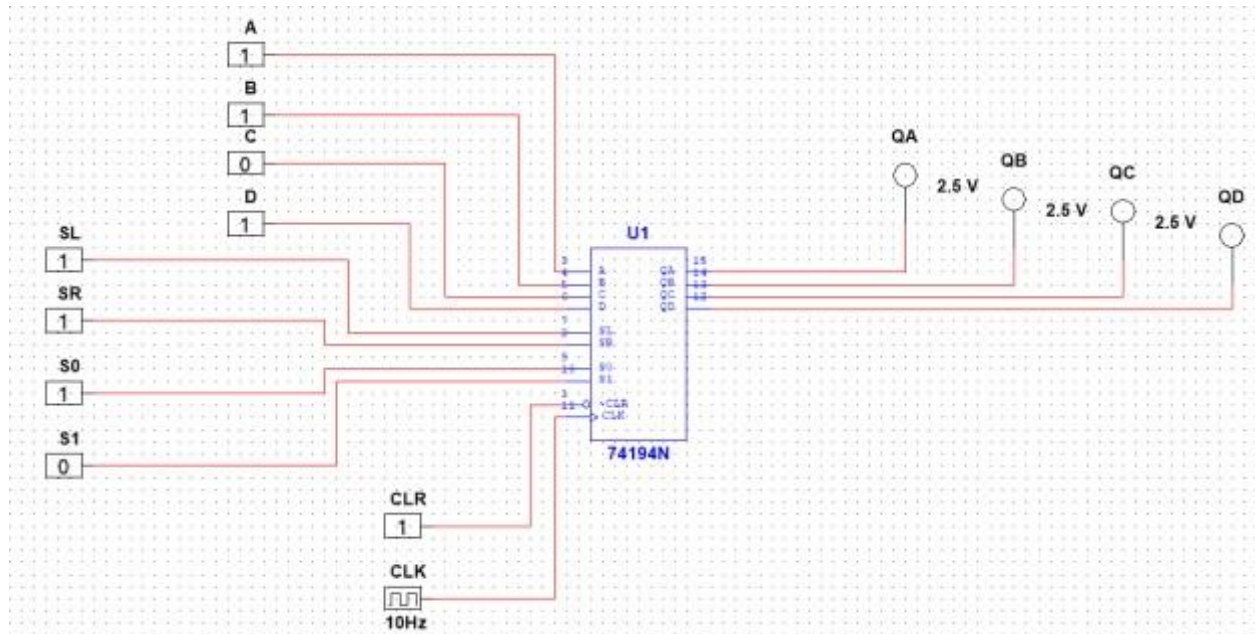
Parallel In Serial Out [PIPO]:



Truth Table:

CLR	SL	SR(Serial Input)	S0	S1	A	B	C	D	QA	QB	QC	QD
1	X	X	1	1	1	0	1	0	1	0	1	0
1	1	0	1	0								0
1	1	0	1	0								1
1	1	0	1	0								0
1	1	0	1	0								1

Serial In Parallel Out [SIPO]:



Truth Table:

CLR	SL	SR(Serial Input)	S0	S1	QA	QB	QC	QD
1	1	1	1	0	1			
1	1	0	1	0	0			
1	1	1	1	0	1			
1	1	0	1	0	0			
1	X	X	1	1	0	1	0	1

Result:

All the shift register operations are verified.

EXPERIMENT 10

Design Pseudo Random Sequence generator using 7495

Aim: To design Pseudo Random Sequence generator using IC 7495 using Pspice/Multisim.

Learning Objectives:

To learn about generation of given sequence using Flip flops.

Components Required:

Sl.No	Group	Family	Component	Quantity
1.	TTL	74STD	IC74194	1
2.	TTL	74LS	IC7486	1
3.	Sources	DIGITAL SOURCES	INTERACTIVE_DIGITAL_CONSTANT	6
4.	Sources	DIGITAL SOURCES	DIGITAL_CLOCK	1
5.	Indicators	PROBE	PROBE_BLUE,PROBE_RED	4

Theory:

The sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways in which these circuits can be designed including those which are based on multiplexers and flip-flops. Here in this article we deal with the designing of sequence generator using D flip-flops (please note that even JK flip-flops can be made use of). As an example, let us consider that we intend to design a circuit which moves through the states 0-1-3-2 before repeating the same pattern.

Design:

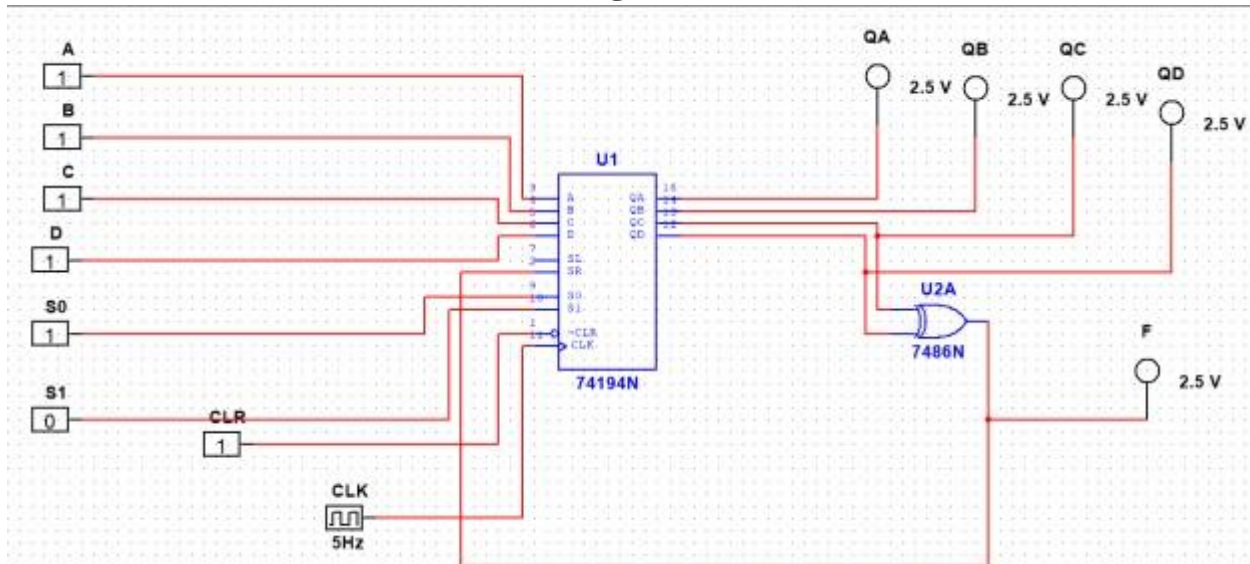
Example-1; The given sequence is **100010011010111**.

The length of the sequence $S=2N -1$, where N represents number of flip flops. The number of flip flops can be decided once after constructing the truth table. If any sequence does not repeat, retain the same number of flip flops (N). Otherwise, increase the number of flip flops by 1. For the given sequence, $S=15$. So, at least 4 flip-flops are required. For the truth table construct the K-map and write the expression for the output sequence f. For the given sequence, the output expression is,

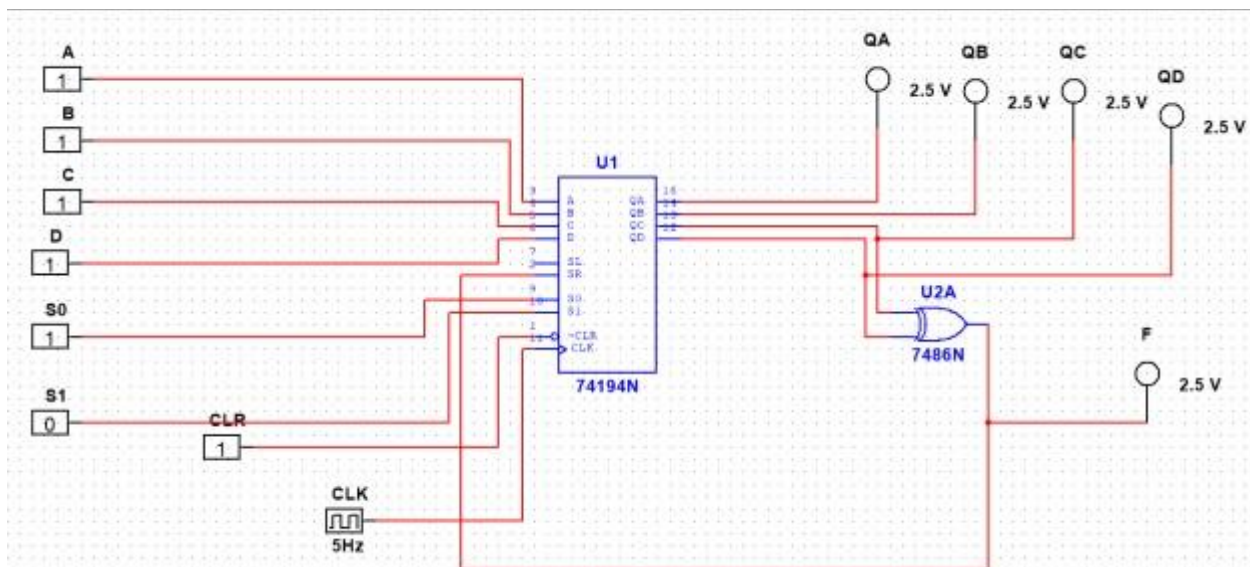
$$F = Q_C \oplus Q_D$$

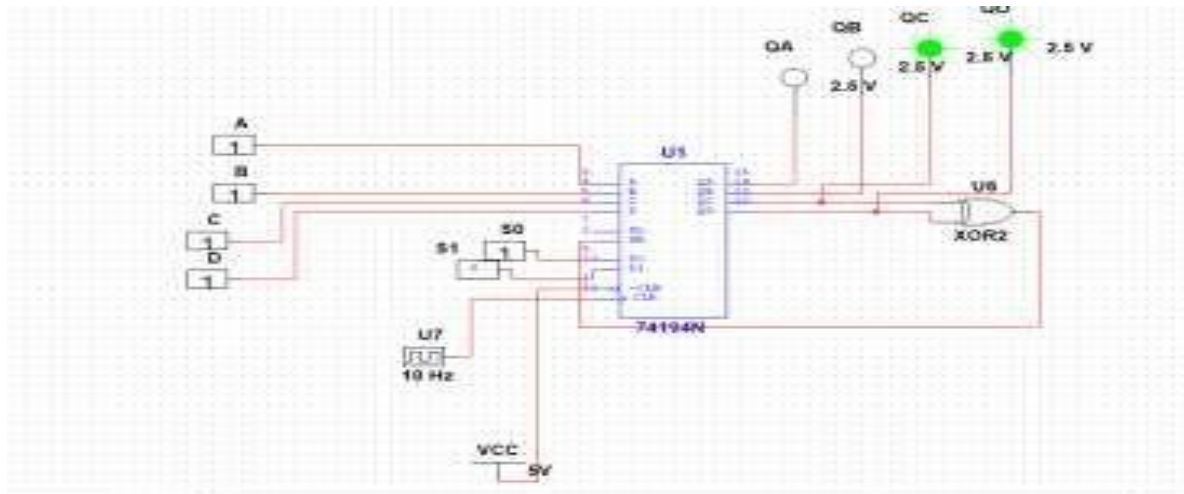
Therefore, use EX-OR gate is used to generate the sequence as shown in the diagram

Multisim simulation: Parallel Loading S1 =1, S0 = 1



Right Shifting S1 = 0, S0 = 1





Truth Table:

QA	QB	QC	QD	F
1	1	1	1	0
0	1	1	1	0
0	0	1	1	0
0	0	0	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	1
1	0	0	1	1
1	1	0	0	0
0	1	1	0	1
1	0	1	1	0
0	1	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

Multisim Procedure:

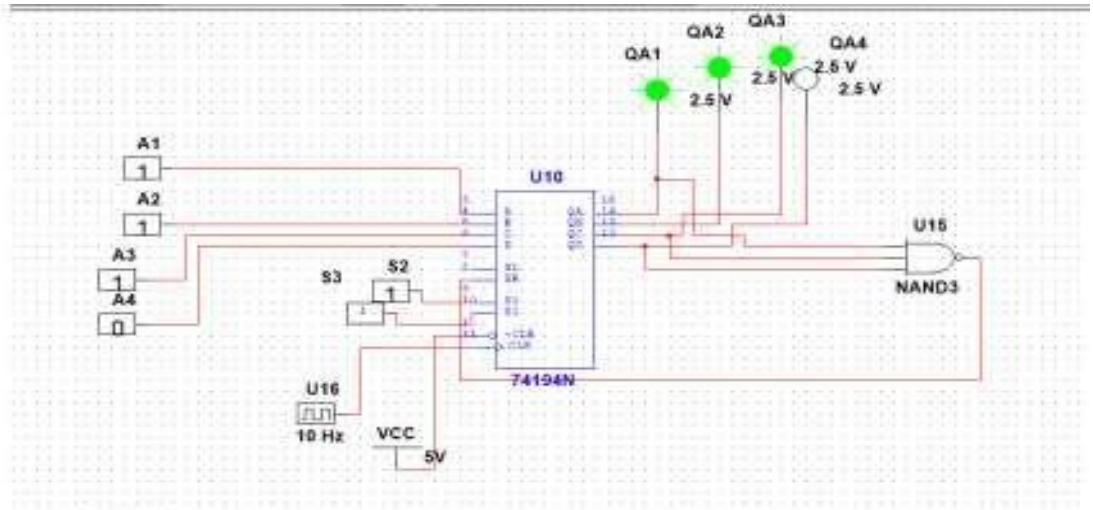
Example 1:100010011010111

1. Place the component 74194 IC
2. Give the inputs A, B, C, and D as digital interactive constants.

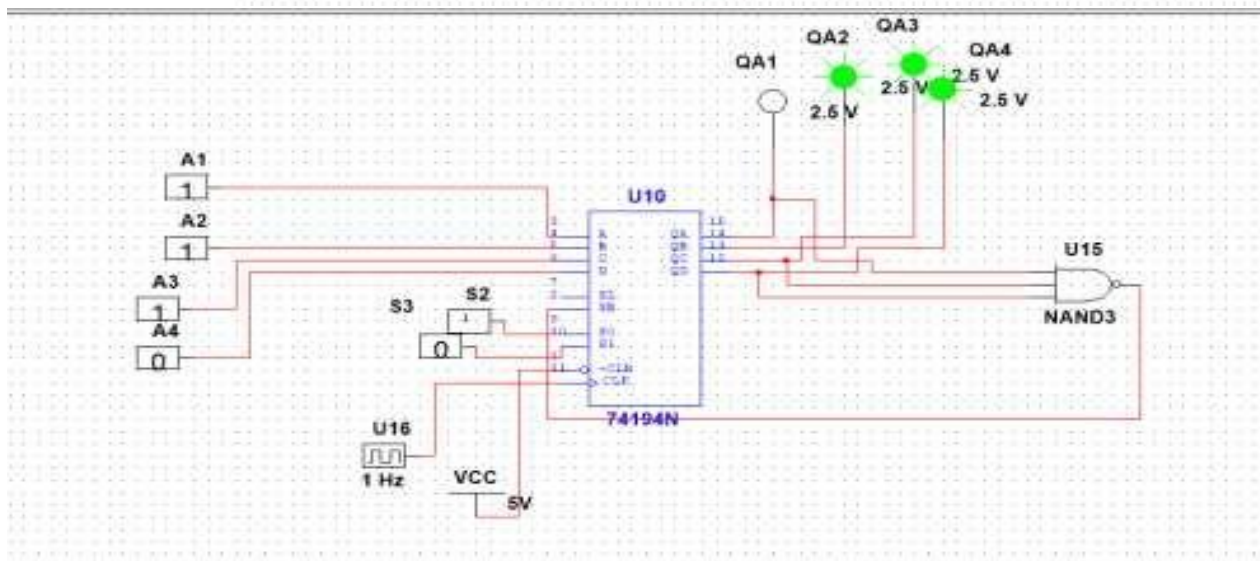
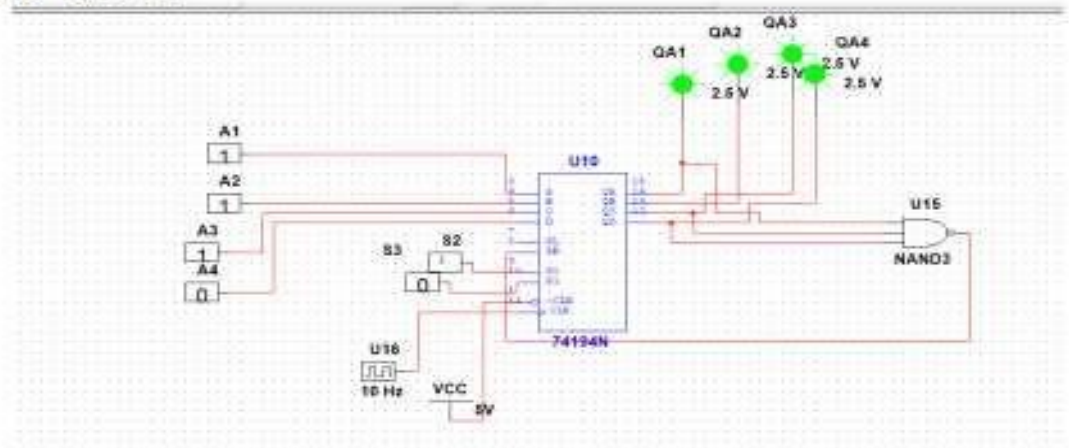
3. Place four probes at the output pins QA, QB, QC, QD.
4. Connect CLR' to VCC
5. Place and Connect digital clock pulse of 10 Hz frequency to the pin clock.
6. To load the parallel data at pins A,B,C,D make S1 and S0 as logic high.
7. Then to do right shift, Make S0 =1 and S1 as 0.
8. Here according to the sequence given, load A,B,C,D as 1111 with S1,S0=11.
9. The outputs QC and QD are connected to XOR gate and given to SR pin.
10. Then making S1=0, S0=1, the output gets shifted by one bit.

Example 2:1101011

1. Place the component 74194 IC.
2. Place the digital interactive constants at A,B,C,D pins.
3. Connect CLR' to VCC.
4. Place and connect component digital interactive constants at S1 and S0 pins.
5. Place and connect digital clock of 10 Hz frequency to the pin clock.
6. Place and connect probes at QA,QB,QC and QD pins.
7. According to the sequence given, Connect QA,QC and QD to the nand gate and output of nand gate to shift right pin of IC.
8. Make S1=S0=1 for parallel loading. Load data 1110 at A,B,C,D.
9. Then make s0=1, s1=0 for right shifting.



Right shifting: $s1=0, s0=0$:



Result: Pseudo random sequence generator are designed and verified.

EXPERIMENT 11

DESIGN SERIAL ADDER WITH ACCUMULATOR AND SIMULATE USING PSPICE/MULTISIM

AIM:

To design the circuit of binary serial adder with accumulator.

COMPONENTS REQUIRED:

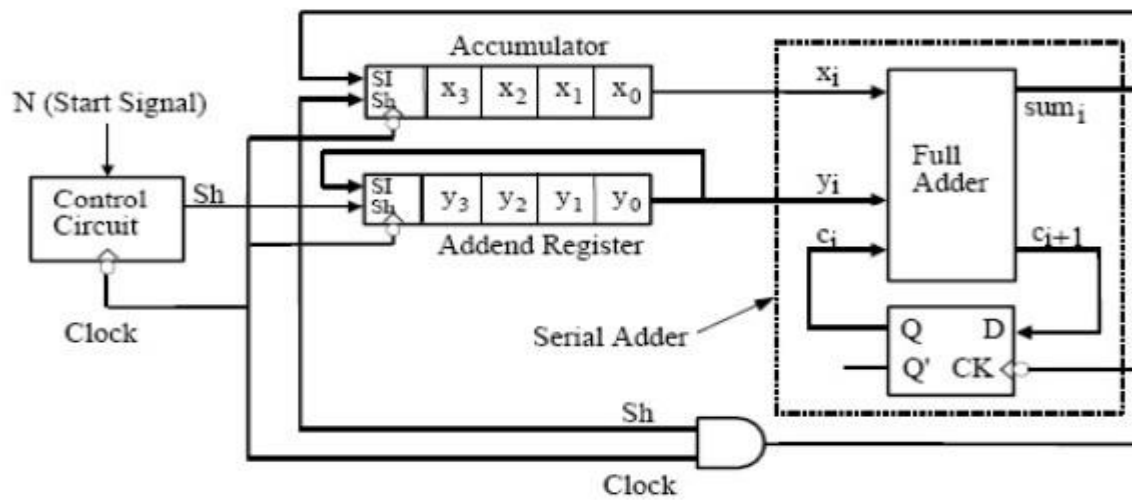
74194, 7474, 7408.

THEORY:

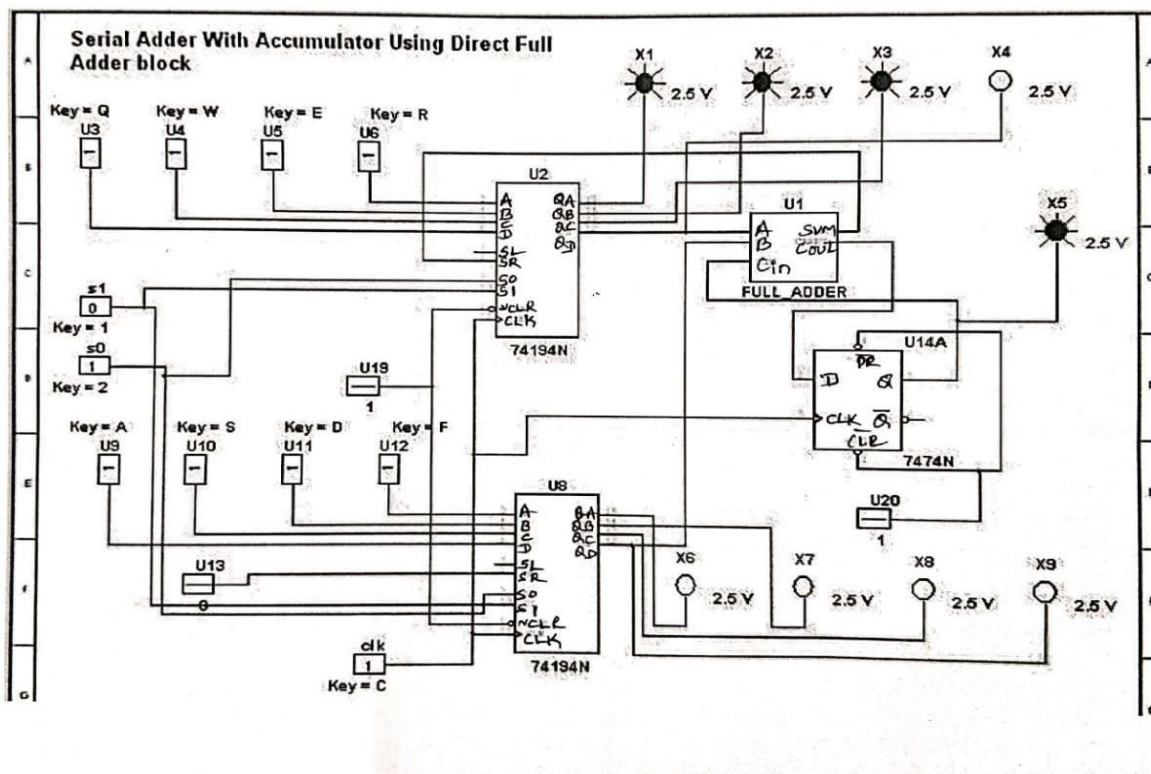
The full adder is used to perform bit by bit addition and D-Flip flop is used to store the carry output generated after addition. This carry is used to carry input for the next addition. Initially the D Flip flop is cleared and addition starts with the least significant bits of both register. After each clock pulse data within the right shift registers are shifted right 1-bit and we get from next digit and carry of previous addition as new inputs for the full adder.

TRUTH TABLE:

	X	Y	c_i	sum_i	c_{i+1}
t_0	0101	0111	0	0	1
t_1	0010	1011	1	0	1
t_2	0001	1101	1	1	1
t_3	1000	1110	1	1	0
t_4	1100	0111	0	(1)	(0)



SCHEMATIC DIAGRAM:



RESULT:

The output waveform of binary serial adder with accumulator is verified using multisim.

EXPERIMENT 12

DESIGN USING PSPICE/MULTISIM MOD – N COUNTERS

Aim:

To realize a Modulo N-counter using 7490 and verify the expected truth table and display the output waveform for a square wave input of given frequency. (N—to be specified, $N \leq 9$)

Equipment/ Apparatus Required:

Trainer Kit	01
IC 7490	01
IC 7400	01
IC 7410	01
Patch chord	25

Theory:

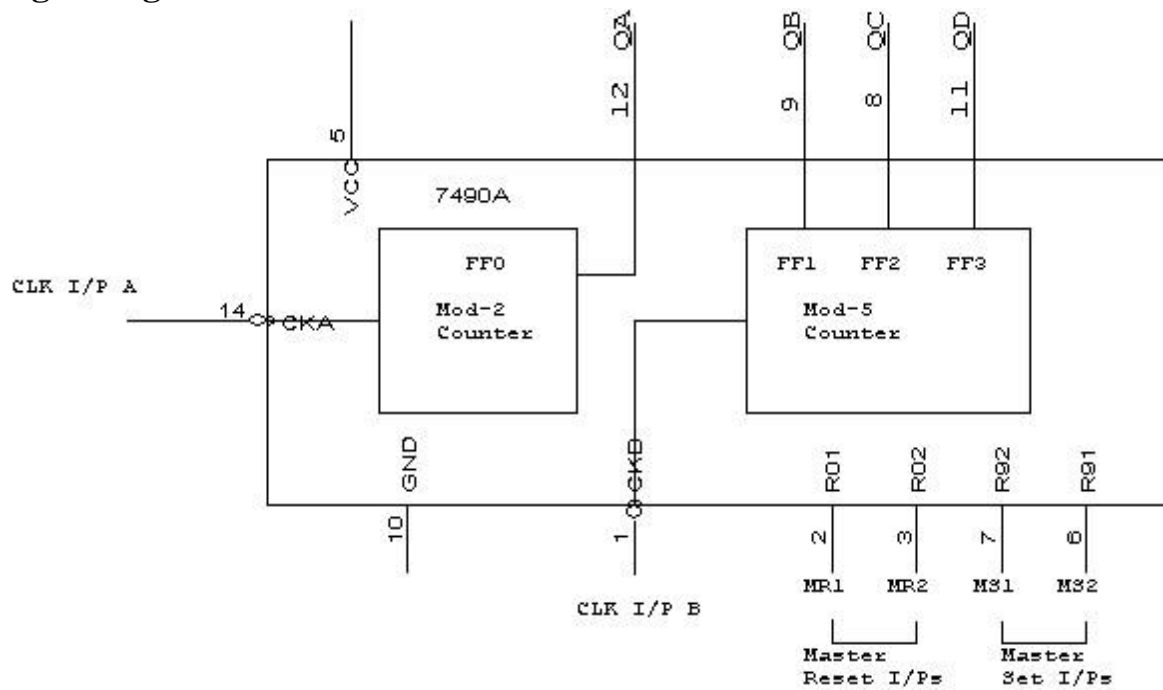
7490 is a $\square 10$ counter using 4 master slave JK flip-flops.

It contains $\square 2$ and $\square 5$ counters, which can be cascaded to give a $\square 10$ counter.

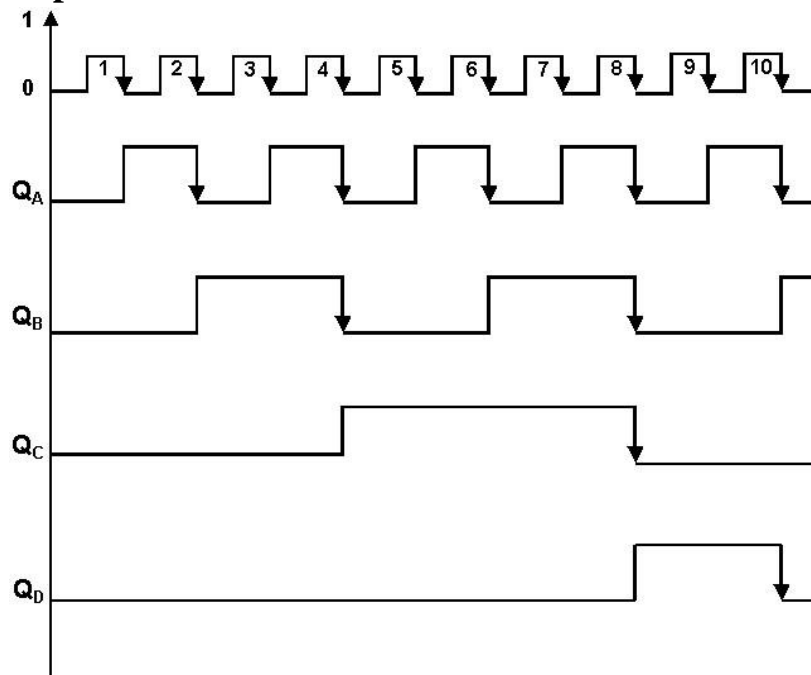
Procedure:

1. Connections are made as shown in the circuit diagram using 7490 IC.
2. Apply the clock pulse and verify the truth table

Logic Diagram:



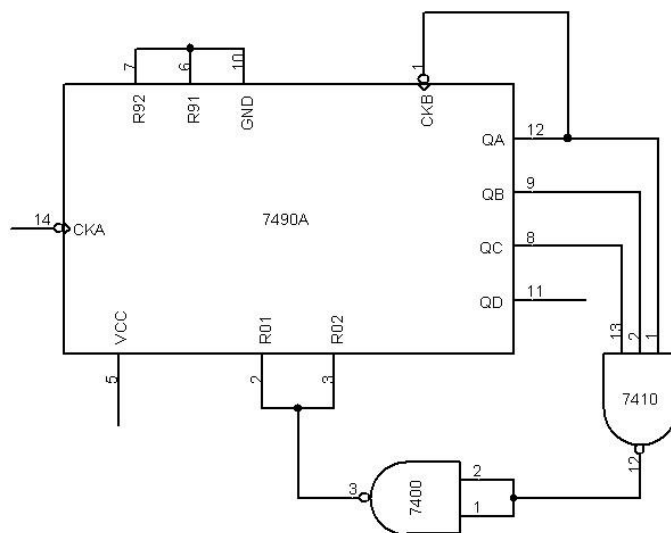
Expected Waveform:



Truth Table:

Clock pulse	Outputs			
	Q _D	Q _C	Q _B	Q _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

Realization Of Mod-7 Counter:



Note:

Logic high of the nth count is given to the reset pin via NAND gate to restart the counter. For mod-7 ($QAQBQCQD = 0111$).

For mod-6 ($QAQBQCQD = 0110$).

Truth Table for mod-7 Counter:

Clock pulses	Flip-flop outputs		
	Q _C	Q _B	Q _A
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	0	0	0

Result:

Thus the mod-N counters using IC 7490 is simulated and output waveform is verified using multisim

.