BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Avalahalli, Doddaballapur Main Road, Bengaluru – 560064

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DIGITAL SYSTEM DESIGN LABORATORY MANUAL (18ECL38)
III ECE

Course Co-ordinators: **Mrs. Hamsavahini R, Mr.Anil Kumar D, Mrs.Asha G H**
Assisnt Professors,Dept.of ECE

Lab Instructor: **Sharmas P, Instructor**

Vision of the Department

Be a Pioneer in Providing Quality Education in Electronics, Communication and Allied Engineering fields to serve as Valuable Resource for Industry and Society

Mission of the Department

- Impart Sound Theoretical Concepts & Practical Skills
- Promote Interdisciplinary Research
- Inculcate Professional Ethics

Programme Educational Objectives

Graduates of the programme will:

PEO1: Work as professionals in the area of Electronics and allied engineering fields.

PEO2: Pursue higher studies and involve in the interdisciplinary research work.

PEO3: Exhibit ethics, professional skills and leadership qualities in their profession.

Programme Specific Outcomes

Graduates will be able to:

PSO1: Exhibit competency in embedded system and VLSI Design.

PSO2: Capability to comprehend the technological advancements in RF Communication and Digital Signal Processing.

Course Objectives

This laboratory course enables students to get practical experience in design, realisation and verification:

1. DE Morgan's Theorem, SOP, POS forms
2. Full/Parallel Adders, Sub tractors and Magnitude Comparator
3. Multiplexer using logic gates
4. De-multiplexers and Decoders
5. Flip-Flops, Shift registers and Counters

Course Outcomes

On the completion of this laboratory course, the students will be able to

CO1: Apply the knowledge of Boolean algebra to demonstrate the truth table of various expressions and combinational circuits using logic gates.
CO2: Analyse and Design various combinational and Sequential circuits
CO3: Simulate Serial adder and Binary Multiplier.
CO4: Conduct and record the experimental data, analyse the results and prepare a formal laboratory report.

CO-PO MAPPING

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	2												2	
CO2		2	2										2	
CO3					1								2	
CO4				2					2	2			2	
Cii	2	2	2	2	1				2	2			2	

Instructions to students:

1. Students must bring observation book, lab record and manual along with necessary stationaries, no borrowing from others.

2. Students must handle the trainer kit and other components carefully, as they are expensive.

3. Before entering to lab, the students must prepare for Viva for which they are going to conduct experiment.

4. Before switching ON the trainer kit, the student must show the connections to one of the faculties or instructors.

5. After the completion of the experiment, student should return the components to the respective lab instructors.

6. Before leaving the lab, students are required to switch off the power supply and arrange the chairs properly.

DIGITAL SYSTEM DESIGN LABORATORY

Laboratory Code: 18ECL38 **IA Marks: 40**

Number of Lecture Hours/Week: 02Hr Tutorial (Instructions) + 02 Hours Laboratory
Exam Marks: 60 Exam Hours: 03

VTU SYLLABUS

Laboratory Experiments:	
1. Verify (i) Demorgan's Theorem for 2 variables. (ii) The sum-of product and product-of-sum expressions using universal gates.	**L1, L2, L3**
2. Design and implement (i) Half Adder & Full Adder using a) basic gates. b) NAND gates (ii) Half subtractor & Full subtractor using a) basic gates b) NAND gates	**L3, L4**
3. Design and implement (i) 4-bitParallelAdder/Subtractor using IC 7483. (ii) BCD to Excess-3 code conversion and vice-versa.	**L3, L4**
4. Design and Implementation of (i) 1-bit Comparator (ii) 5-bit Magnitude Comparator using IC 7485.	**L3, L4**
5. Realize (i) Adder & Subtactors using IC 74153. (ii) 4-variable function using IC74151(8:1MUX).	**L2, L3, L4**
6. Realize (i) Adder & Subtractors using IC74139. (ii) Binary to Gray code conversion & vice-versa (74139)	**L2, L3, L4**
7. Realize the following flip-flops using NAND Gates. i) Master-Slave ,JK, D & T Flip-Flop.	**L2, L3**
8. Realize the following shift registers using IC7474/7495 (i) SISO (ii) SIPO (iii)) PISO(iv))PIPO (v) Ring (vi) Johnson counter	**L2, L3**
9. Realize (i) Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop (ii) Mod-N Counter using IC7490 / 7476 (iii) Synchronous counter using IC74192	**L2, L3**
10. Design Pseudo Random Sequence generator using 7495.	**L2, L3**
11. Design Serial Adder with Accumulator and Simulate using Simulation tool.	**L2, L3, L4**
12. Design Binary Multiplier and Simulate using Simulation tool.	**L2, L3, L4**

NOTE:
1. Use discrete components to test and verify the logic gates. The IC numbers given are suggestive; any equivalent ICs can be used.
2. For experiment No. 11 and 12 any open source or licensed simulation tool may be used.

DIGITAL SYSTEM DESIGN LABORATORY

CYCLE1

1. Verify

(a) DE Morgan's Theorem for 2 variables.

(b) The sum-of product and product-of-sum expressions using universal gates.

2. Design and implement

(a) Full Adder using basic logic gates and universal gates.

(b) Full subtractor using basic logic gates and universal gates .

3. Design and implement

a)4-bit Parallel Adder/ subtractor using IC 7483.

b)BCD to Excess-3 code conversion and vice-versa.

4. Design and Implementation of 1-bit and 5-bit Magnitude Comparator using IC 7485.

Cycle 2

5. Realize (a) Adders and Subtractors using IC74153

(b) 4-variable function using IC 74151(8:1MUX).

6. (a) Realize addres and subtractors using IC74139.

(b) Binary to Gray code conversion & vice-versa

Cycle 3

7. Realize the following flip-flops using NAND Gates.

(a) Master Slave JK ,D and T Flip-Flop

8. Realize the following shift registers using IC7474/7495

(a) SISO (b) SIPO (c) PISO (d) PIPO e)Ring Counter f)Johnson Counter

Cycle 4

10. Design Pseudo Random Sequence generator using 7495.
11. Design Serial Adder with Accumulator and Simulate using Simulation tool.
12. Design Binary Multiplier and Simulate using Simulation tool.

EXPERIMENT NO-1

Verify
(i) De Morgan's Theorem for 2 variables.
(ii) The sum-of product and product-of-sum expressions using universal gates.

Aim: To verify De Morgan's Theorem

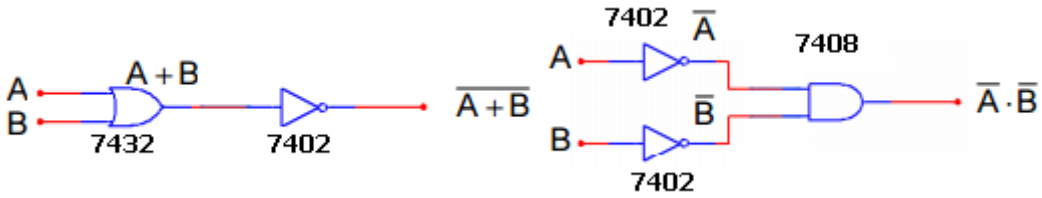
Theory: DeMorgan's Theorem is mainly used to solve the various Boolean algebra expressions.The Demorgan's theorem defines the uniformity between the gate with same inverted input and output. It is used for implementing the basic gate operation likes NAND gate and NOR gate. The Demorgan's theorem mostly used in digital programming and for making digital circuit diagrams. There are two DeMorgan's Theorems. They are described below in detail.

De Morgan's First Theorem

According to De Morgan's first theorem, a NOR gate is equivalent to a bubbled AND gate. The Boolean expressions for the bubbled AND gate can be expressed by the equation shown below. For NOR gate, the equation is

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

Logic Diagram:



Truth Table:

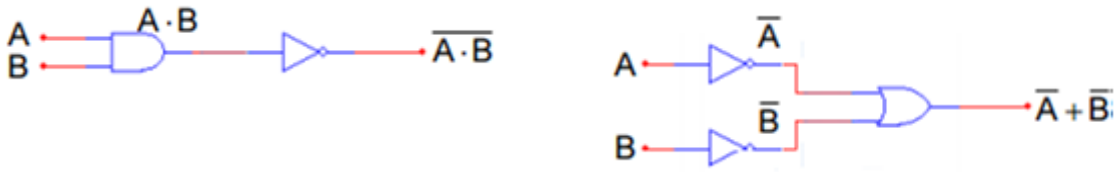
A	B	A´	B´	(A+B)´	A´.B´
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

DeMorgan's Second Theorem

DeMorgan's Second Theorem states that the NAND gate is equivalent to a bubbled OR gate.

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

Logic Diagram:



Truth Table:

A	B	A.B	A´.B´	(A.B)´	A´+B´
0	0	0	1	1	1
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

Result: Verified De Morgan's Theorem using basic gates.

Aim: To Realize the Following Expressions in SOP Form (Sum of Product) and POS Form (Product of Sum)

Theory: To minimize a Boolean expression we can employ any one of the following techniques:

(i) Boolean Algebra
(ii) Karnaugh maps.

Before we proceed to simplification techniques, two forms of the Boolean expression

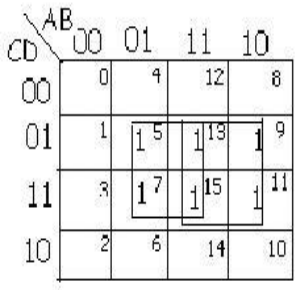
must be noted.

1. **Sum of product (SOP): Ex: ABC+AB+AC**
2. **Product of Sum (POS): Ex: (A+B+C) (A+B) +(A+C)**

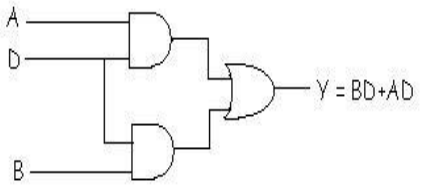
Procedure:

1. Place the IC in the socket of the trainer kit. Complex Boolean Expressions are simplified by using K maps.
2. Make the connections as shown in the circuit diagram.
3. Apply different combinations of inputs according to the truth table. Verify the output.
4. Repeat the above procedure for all the circuit diagrams.

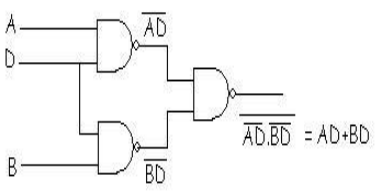
1). Simplification- SOP form using basic gates
F(A,B,C,D) = ∑(5,7,9,11,13,15)



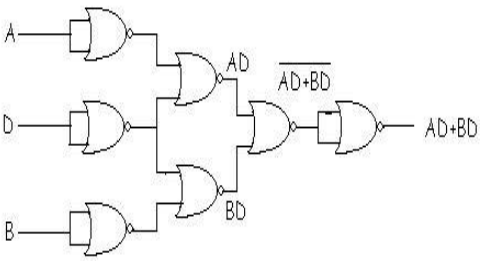
Y = BD+AD



Using NAND gates

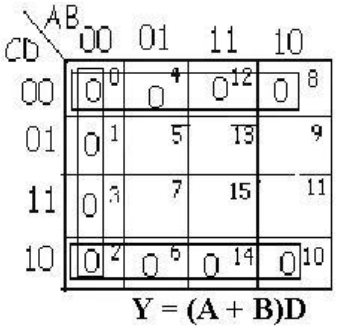


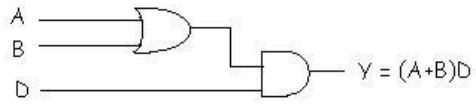
Using NOR gates



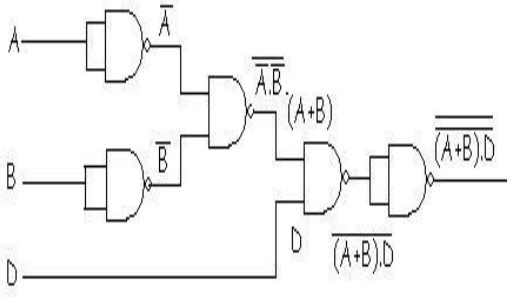
2. Simplification- POS form using basic gates

F(A,B,C,D) =∏(0,1,2,3,4,6,8,10,12,14)

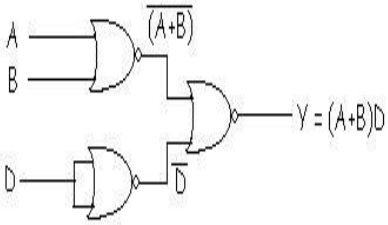




Using NAND gates



Using NOR gates



Truth table:

A	B	C	D	Y=BD+AD	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	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

Result: Verified De Morgan's Theorem and realized both SOP and POS forms of Boolean expressions.

EXPERIMENT NO- 2

Design and implement
(i) Half Adder & Full Adder using i) basic gates. ii) NAND gates
(ii) Half subtractor& Full subtractor using i) basic gates ii) NAND gates

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

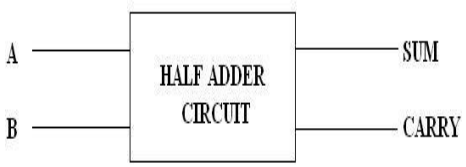
Theory:

(a) ADDER:

An Adder is a circuit which performs addition of binary numbers. Producing sum and carry. An half adder is a digital circuit which performs addition of two binary numbers which are one bit each and produces a sum and a carry (one bit each). A full adder is a digital circuit which performs addition of three binary numbers (one bit each), to produce a sum and a carry (one bit each). Full adders are basic block of any adder circuit as they add two numbers along with the carry from the previous addition.

1. Half Adder

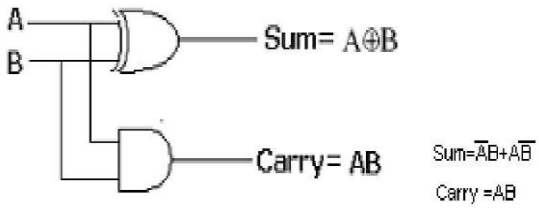
Block Diagram:



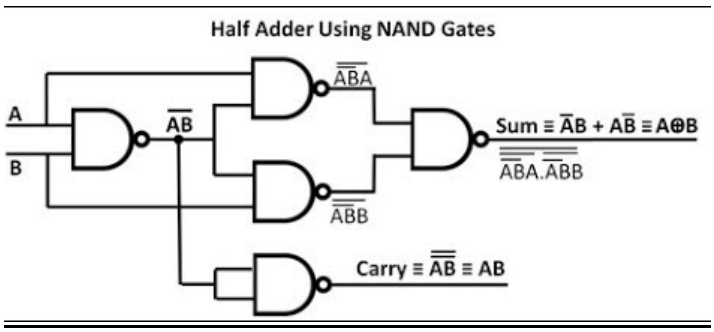
Truth Table

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

Logic Diagram: i) Using Logic Gates

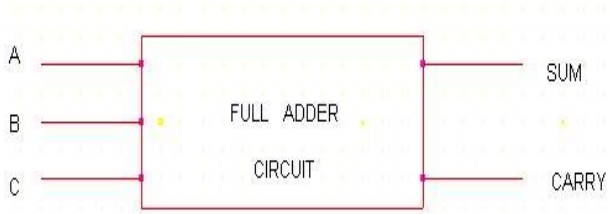


Logic Diagram: i) Using NAND Gates



2. Full Adder

Block Diagram:



(a) Full Adder Using Logic Gates

Truth Table (Full Adder)

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$SUM = \overline{A}\,\overline{B}\,Cin + \overline{A}\,B\,\overline{Cin} + A\,\overline{B}\,\overline{Cin} + A\,B\,Cin$$

$$= (\overline{A}\,\overline{B} + AB)Cin + (\overline{A}\,B + A\,\overline{B})\overline{Cin}$$

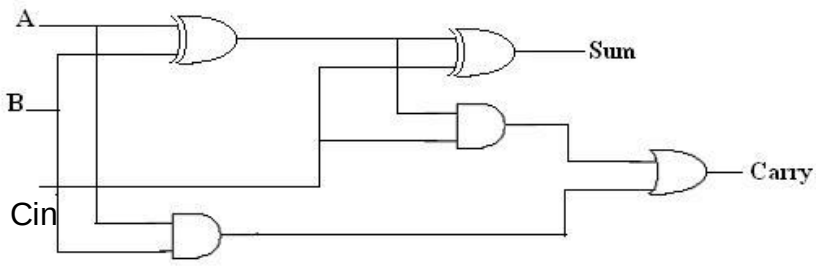
$$SUM = \overline{(A \oplus B)}Cin + (A \oplus B)Cin$$

$$= A \oplus B \oplus C$$

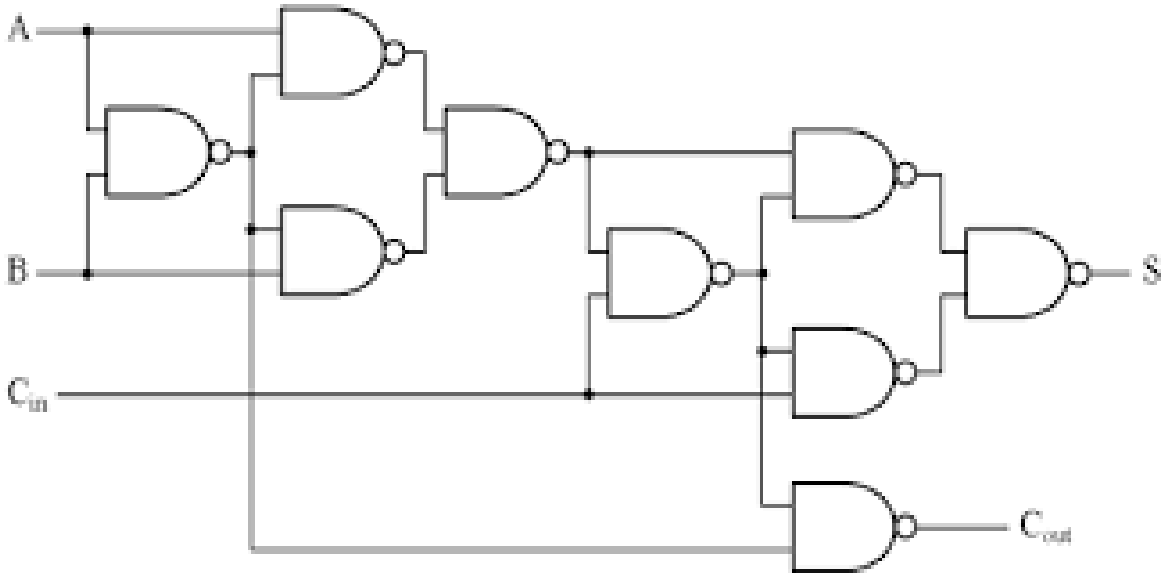
$$carry = (A \oplus B)\,Cin + A\,B$$

$$= AB + BC + CA$$

Logic Diagram: : i) Using Logic Gates



Logic Diagram: : i) Using NAND Gates



Procedure:

1. Place the IC in the socket of the trainer kit.
2. Make the connections as shown in the circuit diagram.
3. Verify the truth table for half adder and full adder circuits using basic and Universal gates.

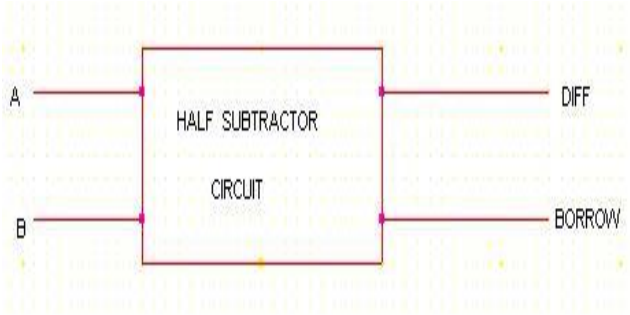
SUBTRACTORS

Theory:

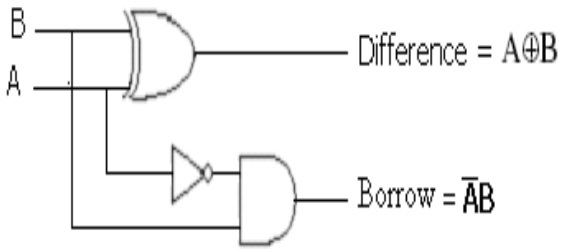
Subtractors are digital circuits which perform subtraction of binary numbers to produce a difference and a borrow if any. A half subtractor subtracts two one bit numbers to give their difference and a borrow if any. A full subtractor subtracts two one bit numbers along with a borrow (from previous stage) to generate a difference and a borrow.

1. Half Subtractor

Block Diagram:



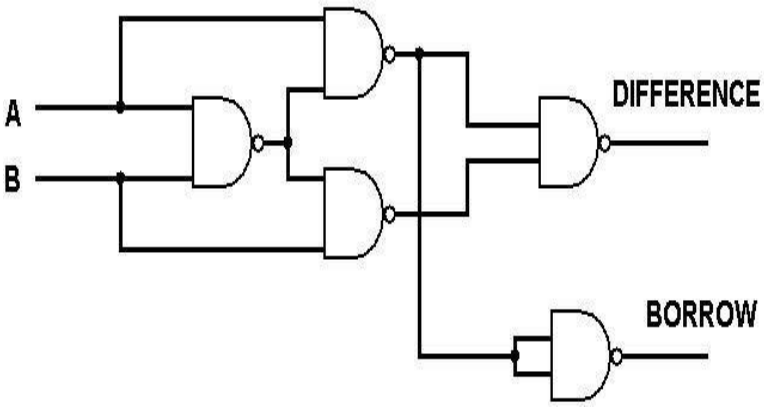
Logic Diagram: : i) Using Logic Gates



Truth Table

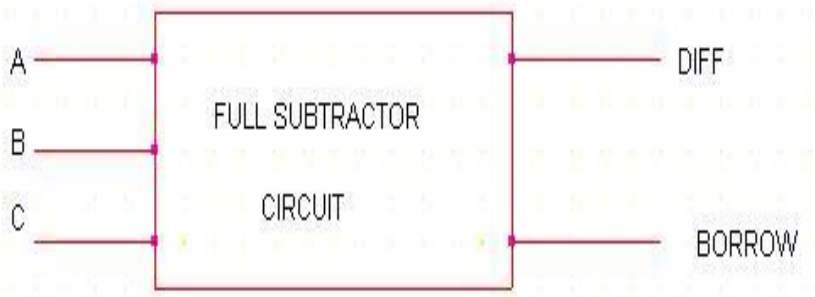
A	B	Sum	Carry
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Logic Diagram: : i) Using NAND Gates



2. Full Subtractor

Block Diagram



(a) Full Subtractor Using Logic Gates

$$\text{Diff} = \bar{A}\,\bar{B}\,Bin + \bar{A}\,B\,\overline{Bin} + A\,\bar{B}\,\overline{Bin} + A\,B\,Bin$$

$$= (\bar{A}\,\bar{B} + A\,B)Bin + (\bar{A}\,B + A\,\bar{B})\,\overline{Bin}$$

$$= (A \odot B)\,Bin + (A \oplus B)\overline{Bin}$$

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

$$Bout = \bar{A}\,\bar{B}\,Bin + \bar{A}\,B\,\overline{Bin} + \bar{A}\,B\,Bin + A\,B\,Bin$$

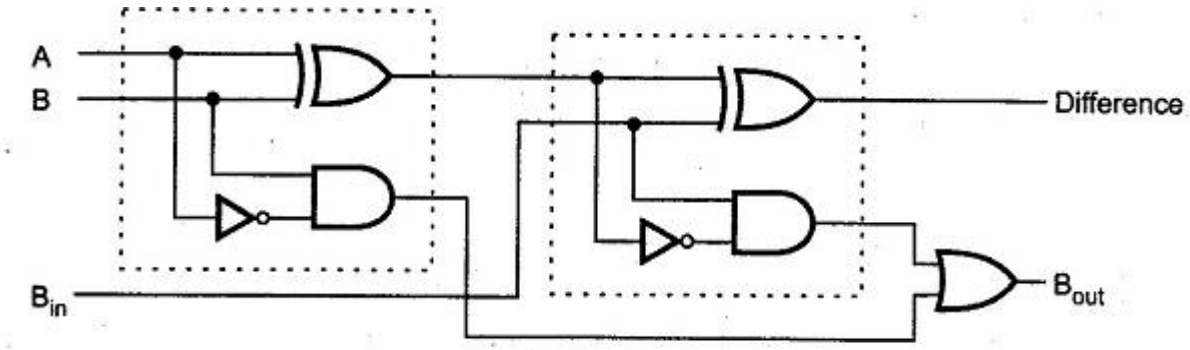
$$= \bar{A}\,[\bar{B}\,Bin + B\overline{Bin}\,] + [A + \bar{A}]\,B\,Bin$$

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

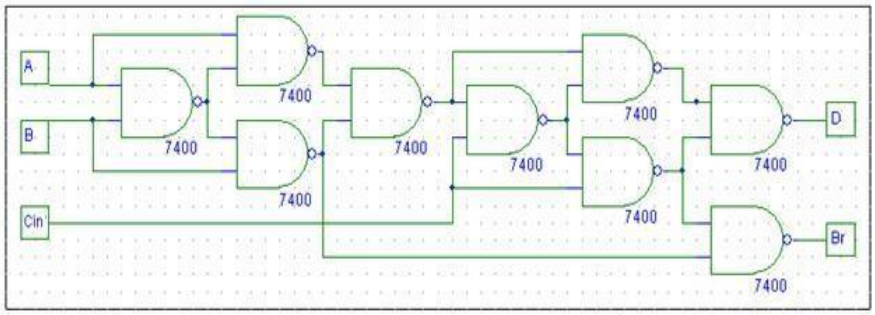
Truth Table

A	B	Bin	Diff	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

Logic Diagram: i) Using Logic Gates



Logic Diagram: **: i) Using NAND Gates**



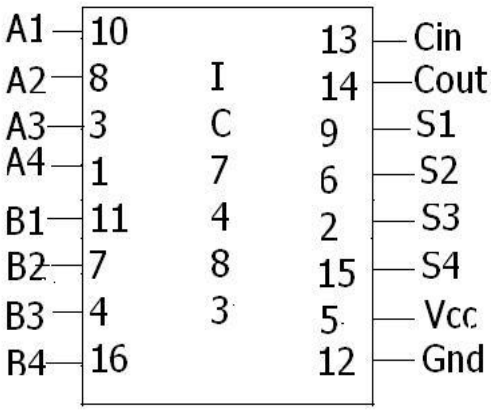
Result: Adders and subtractors are verified using logic gates and Universal gates.

EXPERIMENT NO- 3

Design and implement
(i) 4-bitParallelAdder/Subtractor using IC 7483.
(ii) BCD to Excess-3 code conversion and vice-versa.

Aim: To realize 4-bit Parallel adder/subtractor using 7483

Pin diagram:

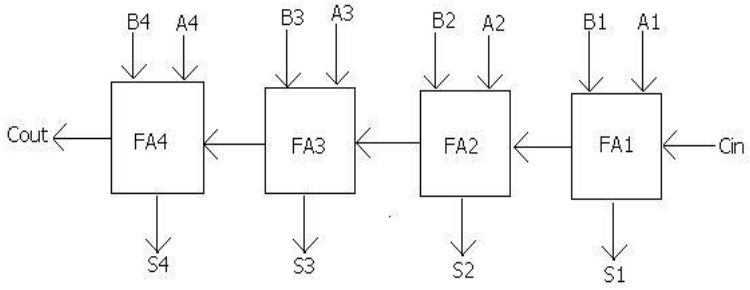


Theory:

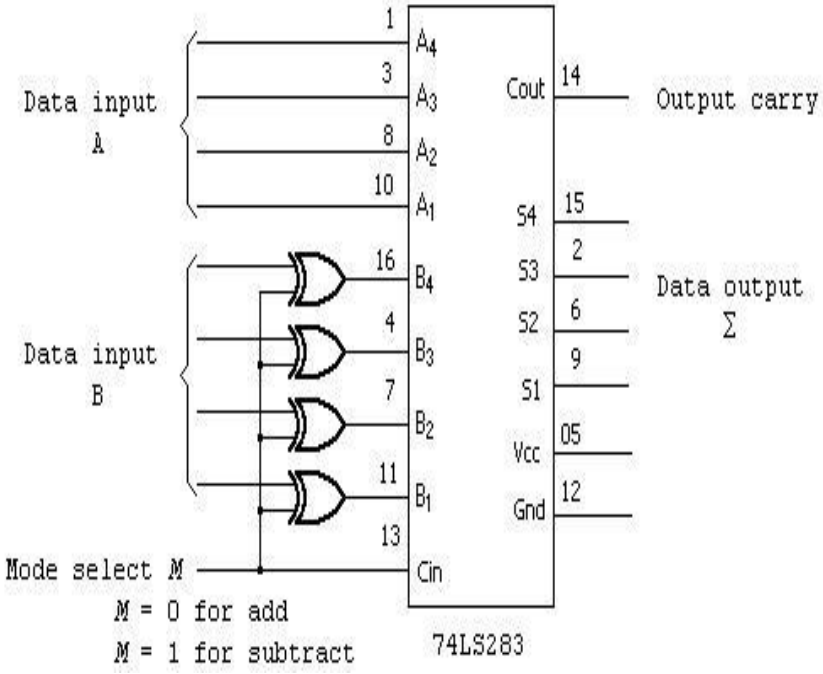
Many high speed adders available in integrated circuit form utilize the look ahead carry or a similar technique for reducing overall propagation delays. A parallel adder consists of n number of full adders and look ahead carry circuitry needed for high speed operation. A parallel subtractor is one where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

There are wide variety of binary codes used in digital systems. Some of these codes are binary-coded- decimal (BCD), Excess-3, gray, and so on. Many times it is required to convert one code to another.

Block Diagram:



Logic Diagram:



Procedure:

1. Make the connections as shown.
2. For addition ,make Cin=0 and apply the 4 bits as i/p for A and apply another set of A bits to B. Observe the o/p at S3, S2 S1 S0 and carry generated at Cout. Repeat the above steps for different inputs and tabulate the result.
3. For subtration Cin is made equal to 1 and A-B format is used. A-First no ,B- second no .By Xoring the i/p bits of 'B' by 1 , is complement of 'B' is obtained.
4. Further Cin , which is 1 is added to the LSB of the Xor –ed bits. This generates 2's complement of B.
5. Verify the difference and polarity of differences at S0, S1, S2, S3. and Cout. If Cout is

0 , diff is –ve and diff is 2's complement form.If Cout is 1, diff is +ve .

6. Repeat the above steps for different inputs. And tabulate the result.

Examples:

i) 4 bit subtraction operation using 7483 for A>B and Cin=1 Example: 8 – 3 =

5 which is equal to $(0101)_2$

• 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 = 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 0

S3 S2 S1 S0 = 0 1 0 1 Cout = 1 (Ignored)

(ii) 4 bit subtraction operation using 7483 for A<B and Cin=1

Example: 14 – 15 = -1 $(1111)_2$

• 14 is realized at A3 A2 A1 A0 = 1110

• 15 is realized at B3 B2 B1 B0 through X-OR gates = 1111

• Output of X-OR gate is 1's complement of 15 = 0000

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

Therefore Cin = 1

A3 A2 A1 A0 = 1 1 1 0

B3 B2 B1 B0 = 0 0 0 0

S3 S2 S1 S0 = 1 1 1 1

Since the most significant bit of the result is 1, this is a negative number, so form the two's complement of (1111)=-$(0001)_2$

Result: Realized Parallel adder and subtractor.

Aim: To realize BCD TO EXCESS-3 CODE CONVERSION AND VISE VERSA USING IC 7483

Theory:

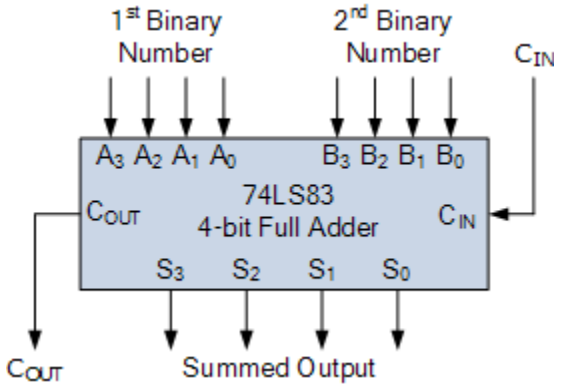
Code converter is a combinational circuit that translates the input code word into a new corresponding word. The excess-3 code digit is obtained by adding three to the corresponding BCD digit. To Construct a BCD-to-excess-3-code converter with a 4-bit adder feed BCD code to the 4- bit adder as the first 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.

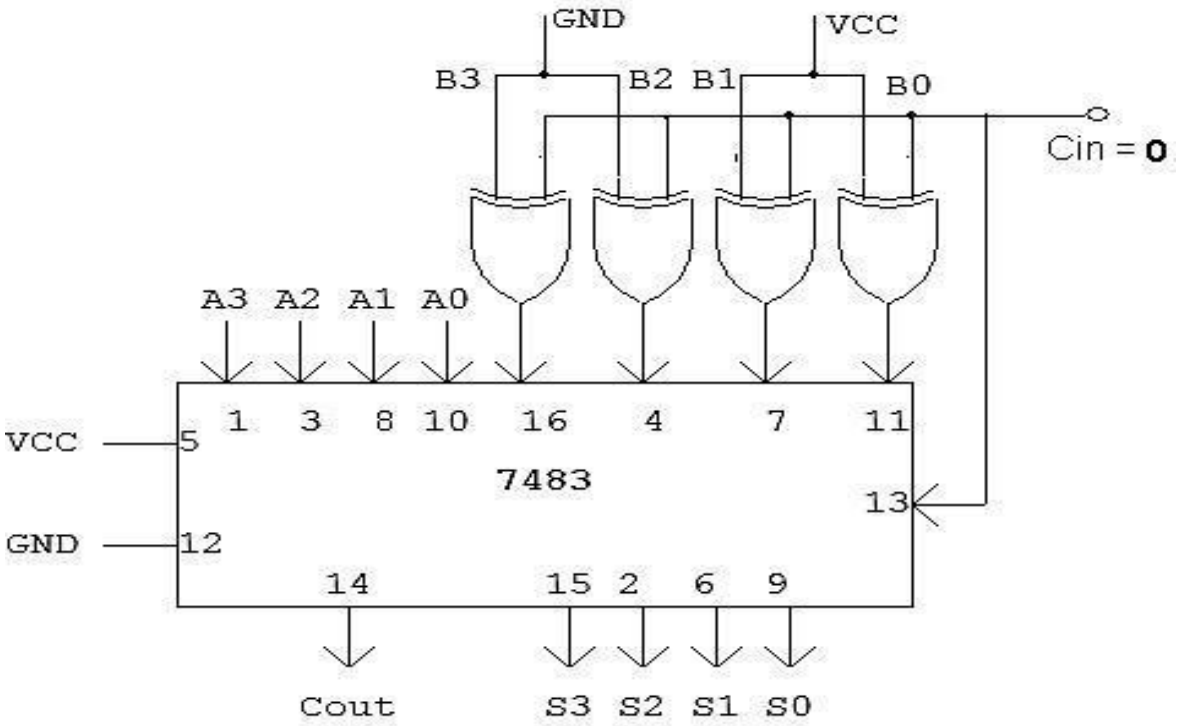
Truth Table:

BCD(8421)				Excess-3			
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
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Pin Diagram:



Logic Diagram:

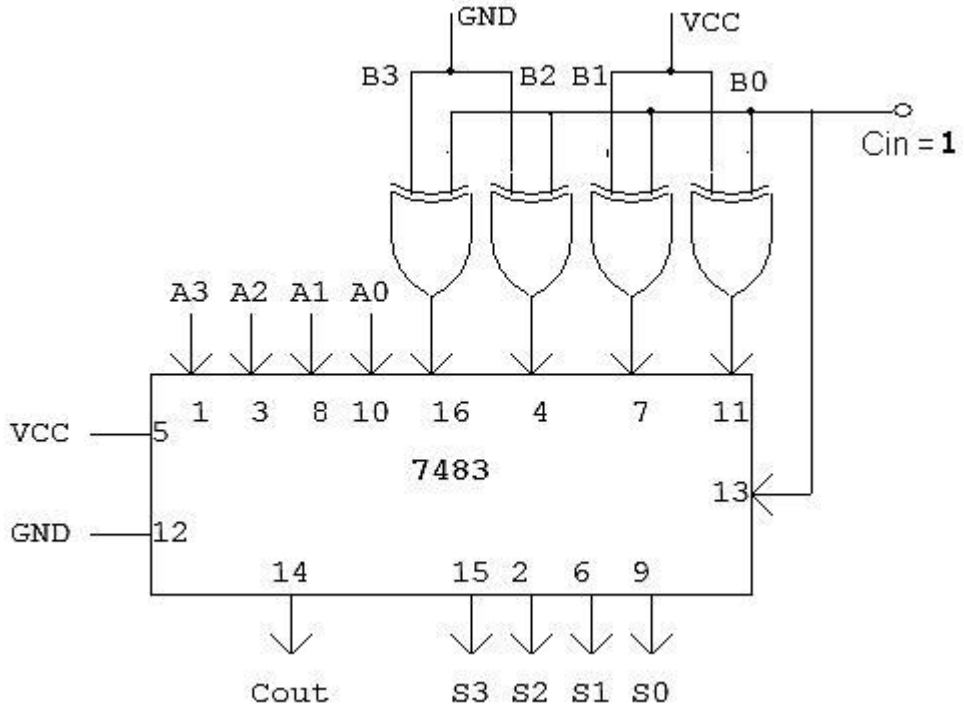


Aim: To realize EXCESS-3 CODE to BCD conversion

Truth Table:

Excess-3				BCD			
w	x	y	z	A	B	C	D
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
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
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Logic Diagram:



Procedure:

1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Apply Excess-3-code code as first operand (A) and binary 3 as second operand (B) and Cin=1 for realizing Excess-3-code to BCD.

Result: Realised BCD to Excess3 code and Vice versa

EXPERIMENT NO-4

Design and Implementation of

(i) 1-bit Comparator
(ii) 5-bit Magnitude Comparator using IC 7485.

Aim: To realise 1-Bit Comparator and 5-bit Comparator using IC 7485

Theory:

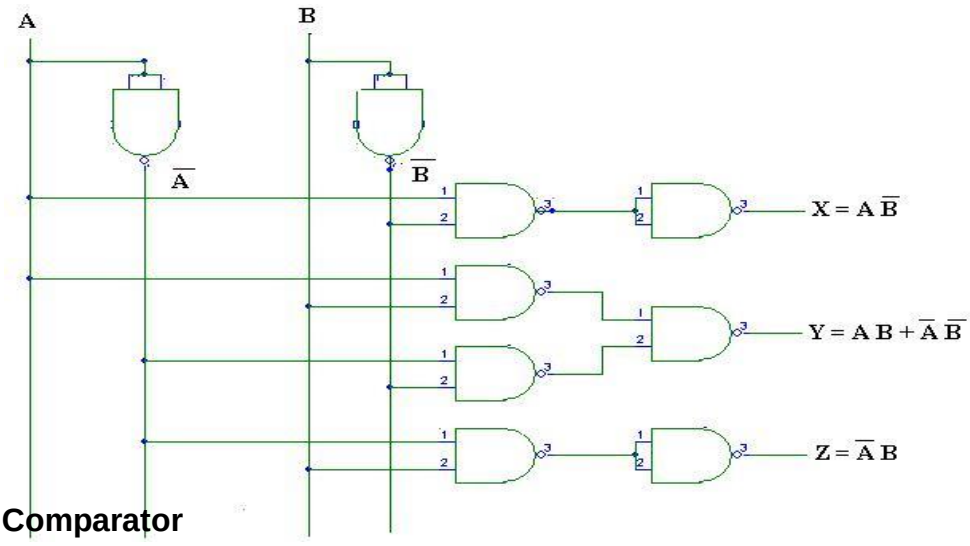
Comparison of two numbers is an operation that determines if one number is greater than, less than, or equal to the other number. A magnitude comparator is a combinational circuit that compares two numbers, A and B and determines the relative magnitudes. The outcome of the comparison is specified by three binary variables that indicate whether A>B, A=B or A<B.

(a) 1-Bit Comparator

Truth Table

		X	Y	Z
A0	**B0**	**A>B**	**A=B**	**A<B**
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Logic Diagram Using NAND Gates:

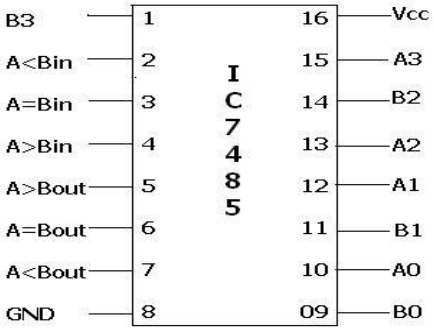


b)5 -Bit Comparator

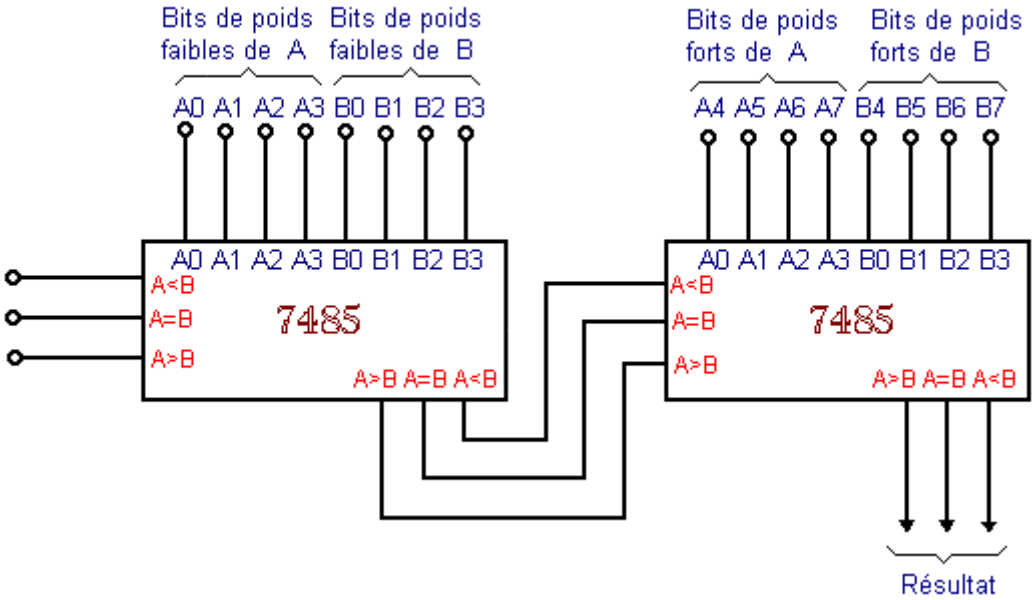
Truth Table:

A4	A3	A2	A1	A0	B4	B3	B2	B1	B0	A>B	A=B	A<B
0	0	0	1	0	1	1	1	0	1	0	0	1
1	1	1	1	1	0	1	1	1	0	1	0	0
1	1	0	0	1	1	1	0	0	1		1	0

Pin Diagram:



Logic Diagram:



Procedure:

1) Rig up the circuit for one bit &5- bit comparator as shown in the figure using IC 7485 magnitude comparator and basic gates.
2) Verify the Table of values .The output obtained should match the required result.
3) For IC 7485 connect the numbers to be compared to input A and input B pins.
4) The inputs A<B, A> B should be connected to logic '0' or grounded. The input A=B should be connected to logic '1' or Vcc. (It is used for cascading).
5) We can cascade two 7485 to design an 5-bit comparator. While cascading, the outputs A>B,A<B and A=B of the first chip should be connected to the inputs A>B,A<B and A=B of the second chip as shown in the figure.

Result: Realized 1-bit and 5- bit magnitude comparators.

EXPERIMENT NO-5

Realize
(i) Adder & Subtactors using IC 74153.
(ii) 4-variable function using IC74151(8:1MUX).

Aim: To Realize **Adders and Subtractors using 74153**

Theory:

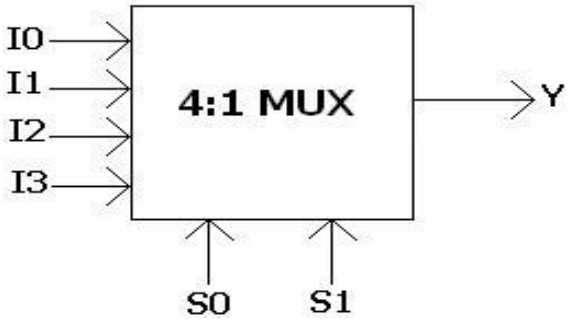
A Multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

a)Multiplexer

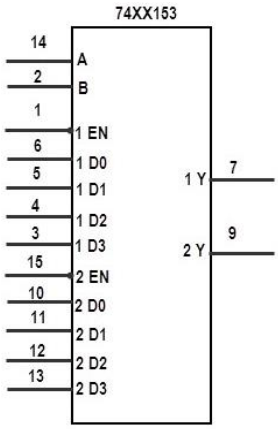
Truth table (4:1 MUX)

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

Symbol



Pin Diagram:

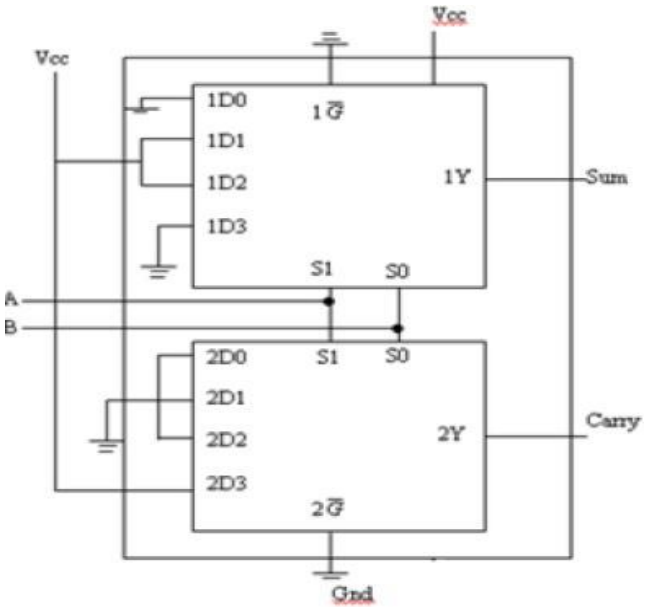


(i) Half Adder:

Truth Table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuit diagram:

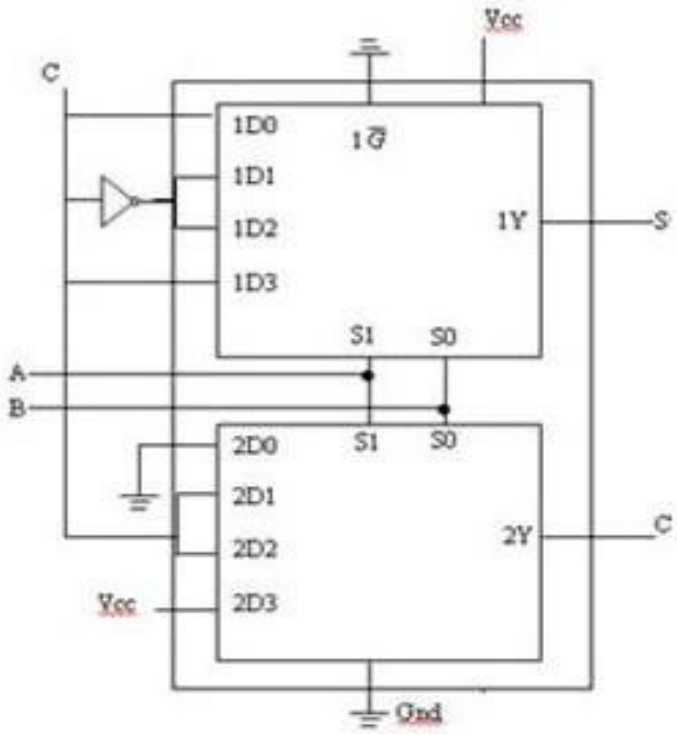


(ii) Full Adder:

Truth Table:

A	B	Cin	S	C
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

Circuit diagram:



Result: Realized Adders and subtractors using 4:1 Mux.

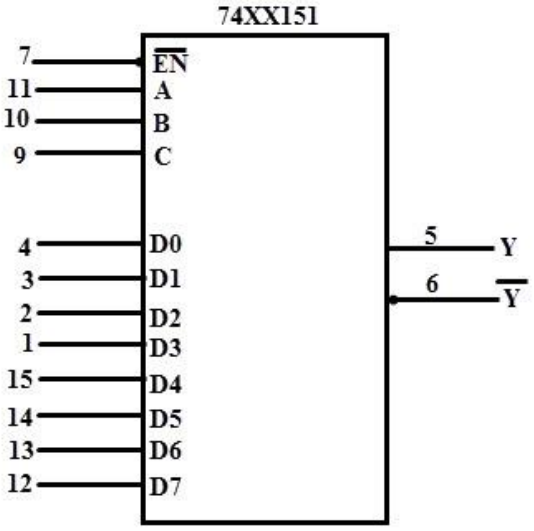
Aim: To realise 4-variable function F(A,B,C,D) = ∑m (2,4,5,7,10,14)

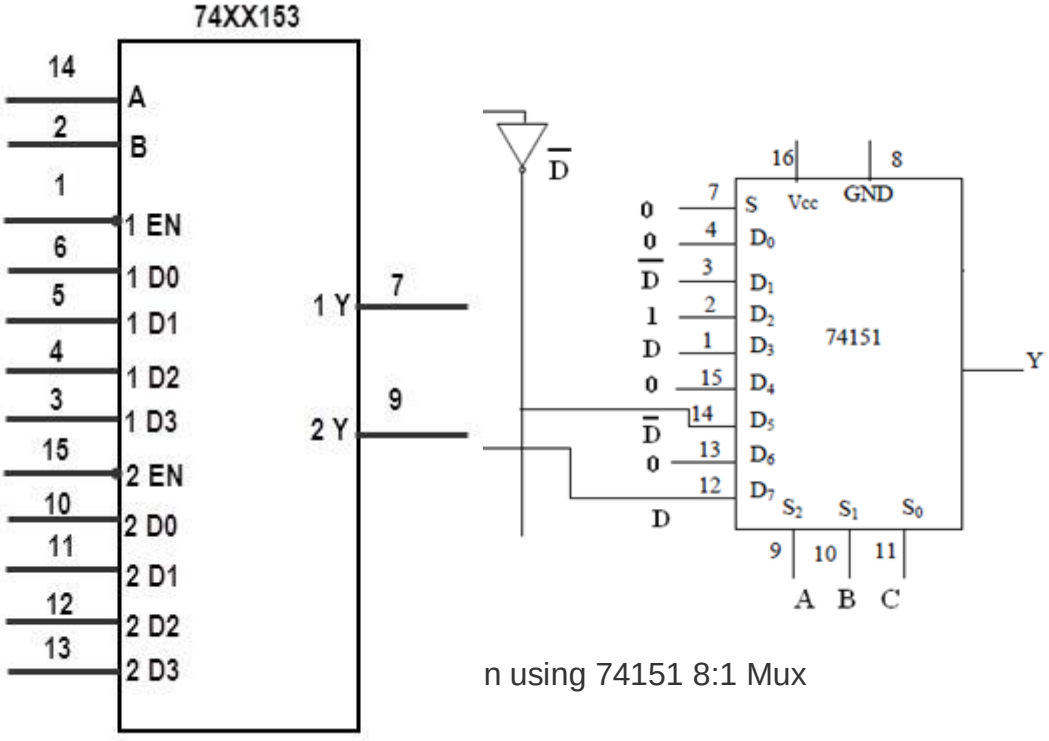
using IC74151(8:1MUX).

Theory:

The given function is in terms of minterms and is to be implemented using a 8:1 MUX. An 8:1 MUX has three select lines, whereas the given function is a 4 variable function. Hence a logic is needed to give combination of D as inputs while only A,B,and C as select line inputs. The method for the same is described below.

Pin Diagram:





n using 74151 8:1 Mux

EXPERIMENT NO-6

Realize (i) Adders & Subtractors using IC74139.

(ii) Binary to Gray code conversion & vice-versa (74139)

Aim: Realize **Adder and subtractor** using IC74138/139.

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.

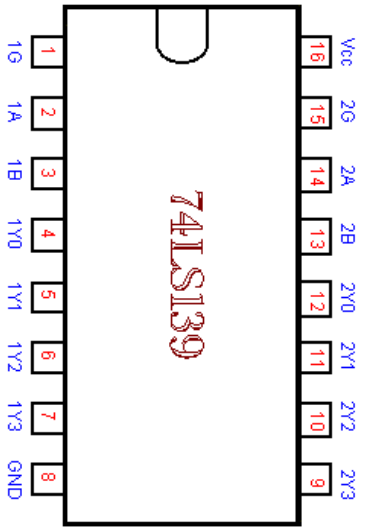
In 1:4 demultiplexer, Din is taken as a data input line and sel(0) and sel(1) are taken as the selection lines. The single input variable Din has a path to all four outputs, but the input information is directed to only one of the output lines, as specified by the binary combination of the 2 selection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

1. IC 74139 DEMUX/ DECODER

Truth Table:

Entrées			Sorties			
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

Pin Diagram:



(a) Realization of Half ADDER/ Half Subtractor Using IC

Truth Table:

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

From truth table

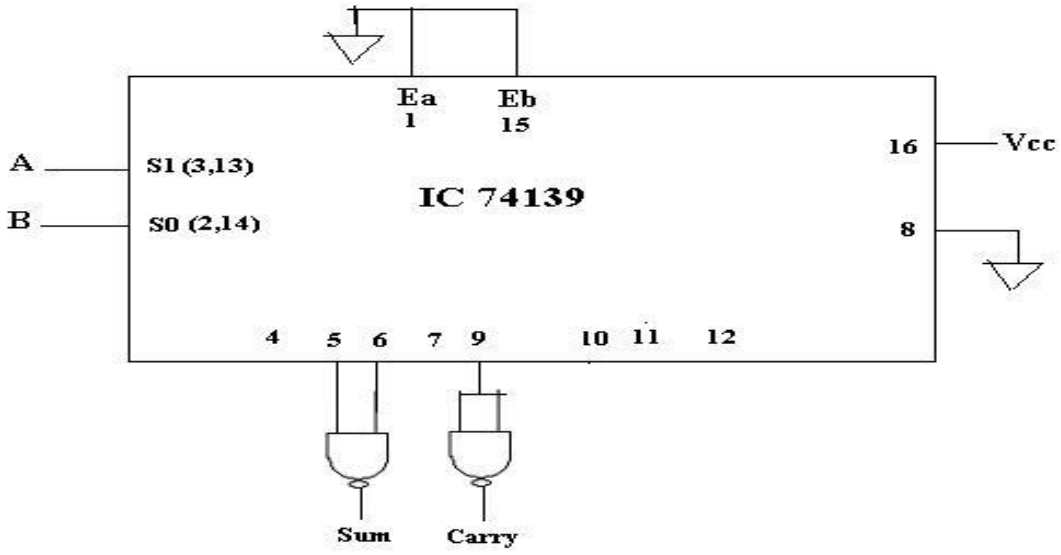
Sum = ∑1,2

Carry =∑3

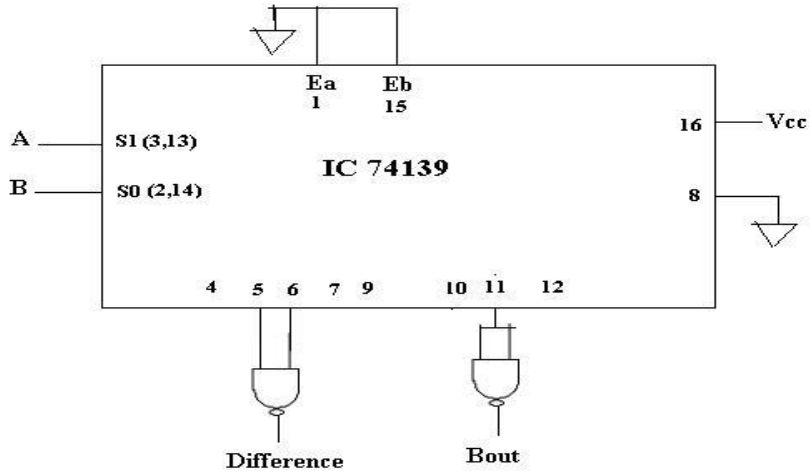
Diff =∑1,2

Bout =∑1

Logic Diagram for Half Adder:

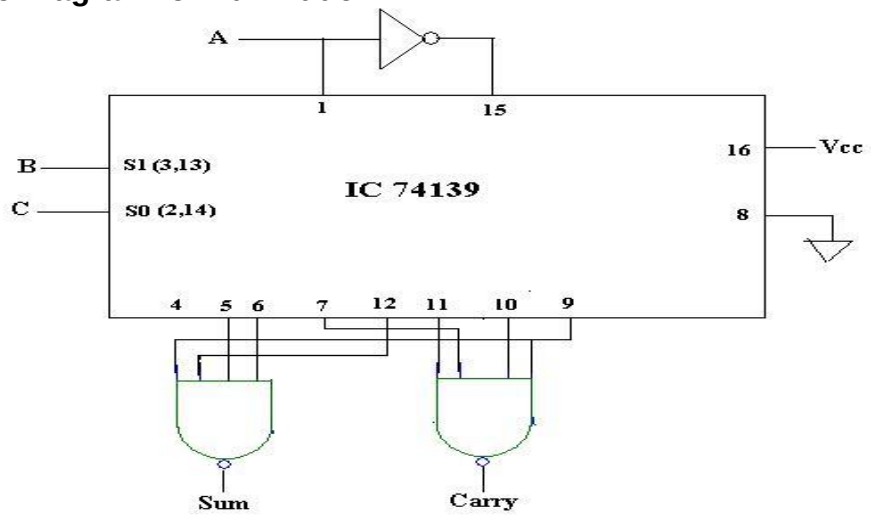


Logic Diagram for Half Subtractor



(b) Realization of Full ADDER/ Full Subtractor Using IC

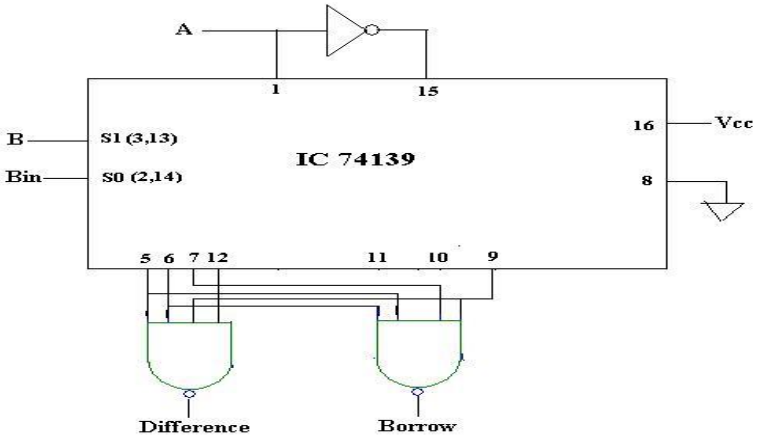
Logic Diagram for Full Adder



Truth Table

A	B	Cin/Bin	SUM	Cout	Diff	Bout
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

Logic Diagram for Full Subtractor



Result: Realised Adder and subtractor using IC74139

Aim: Realise Binary to gray code conversion and Vice versa USING IC74139 (2-4 Decoder).

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.

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

(2) 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.

(3)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.

Gray code to binary conversion is again very simple and easy process.

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

(1) The M.S.B of the binary number will be equal to the M.S.B of the given gray code.

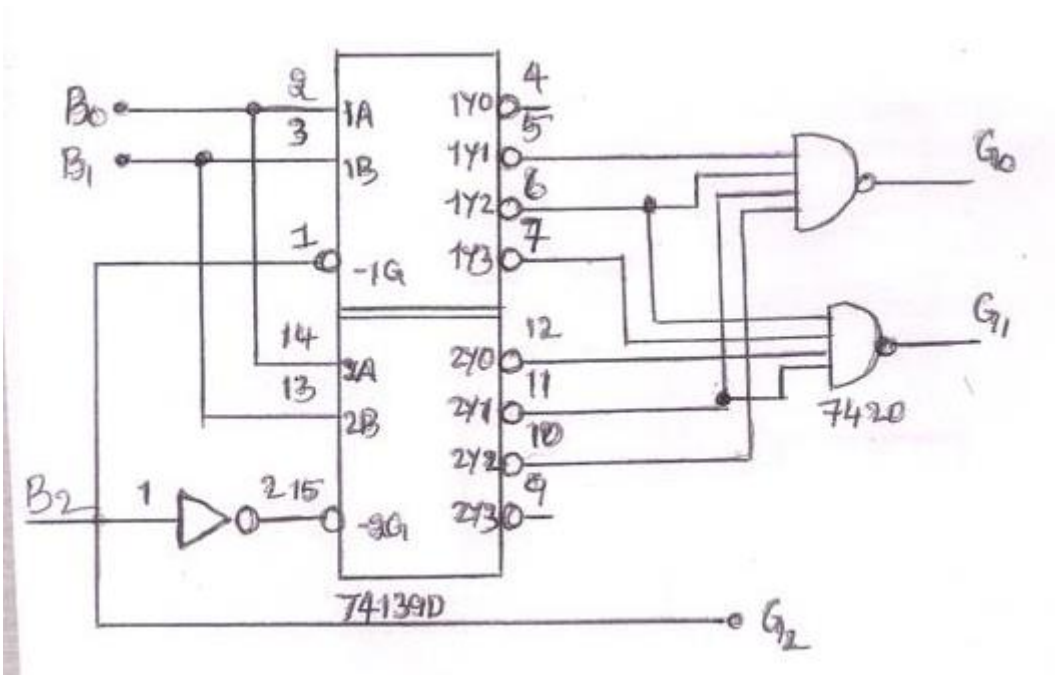
(2) Now if the second gray bit is 0 the second binary bit will be same as the previous or the first bit. If the gray bit is 1 the second binary bit will alter. If it was 1 it will be 0 and if it was 0 it will be 1.

(3) This step is continued for all the bits to do **Gray code to binary conversion**.

Truth Table: Binary to Gray Code Converter

Decimal	binary	Gray
	b_2, b_1, b_0	g_2, g_1, g_0
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Circuit Diagram:

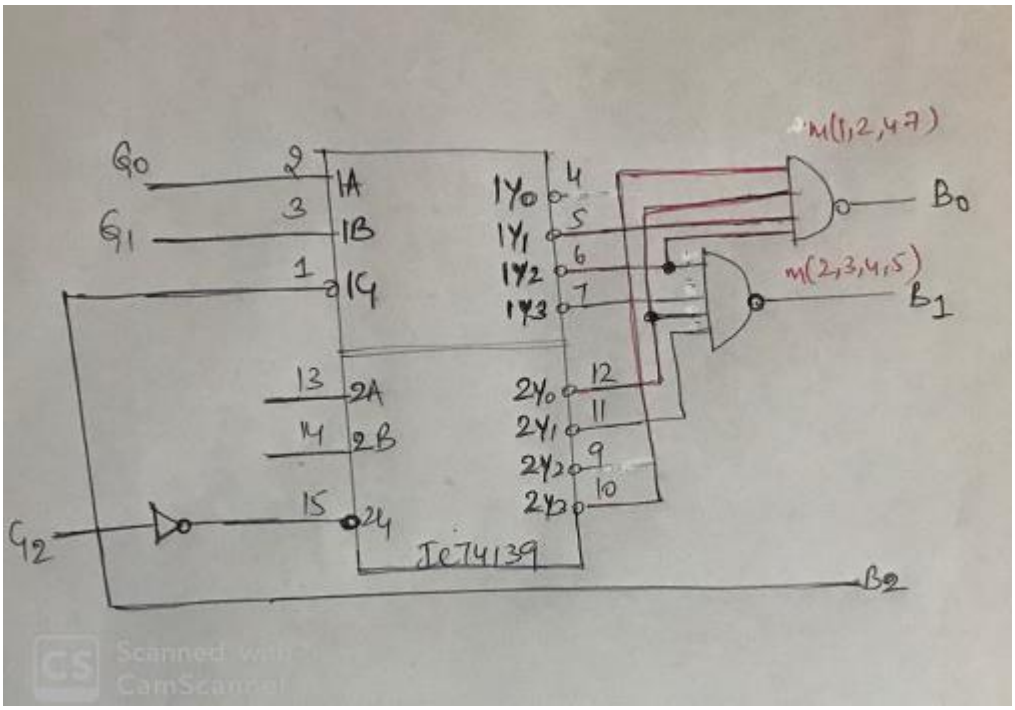


G0=Σm(1,2,5,6)
G1=Σm(2,3,4,5)
G2=B2

Truth Table: Gray to Binary Code Converter

Gray G2G1G0	Binary B2B1B0
0 0 0	0 0 0
1 0 0	0 0 1
1 1 0	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 1 1	1 0 1
1 0 1	1 1 0
0 0 1	1 1 1

Circuit Diagram:



B2=G2

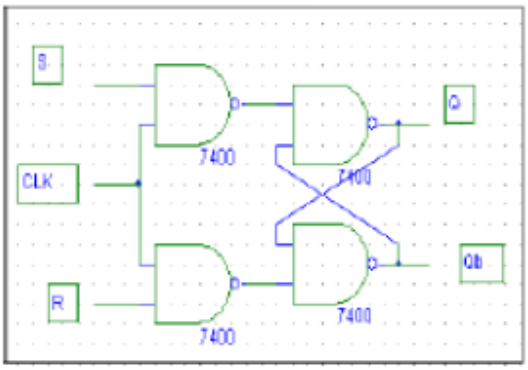
B1=Σm(2,3,4,5)

B0=Σm(1,2,4,7)

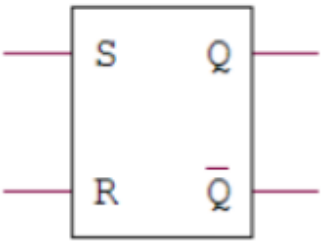
Result: Realised Binary to gary and Vice versa using IC 74139

2) SR FLIP FLOP:

CIRCUIT DIAGRAM:



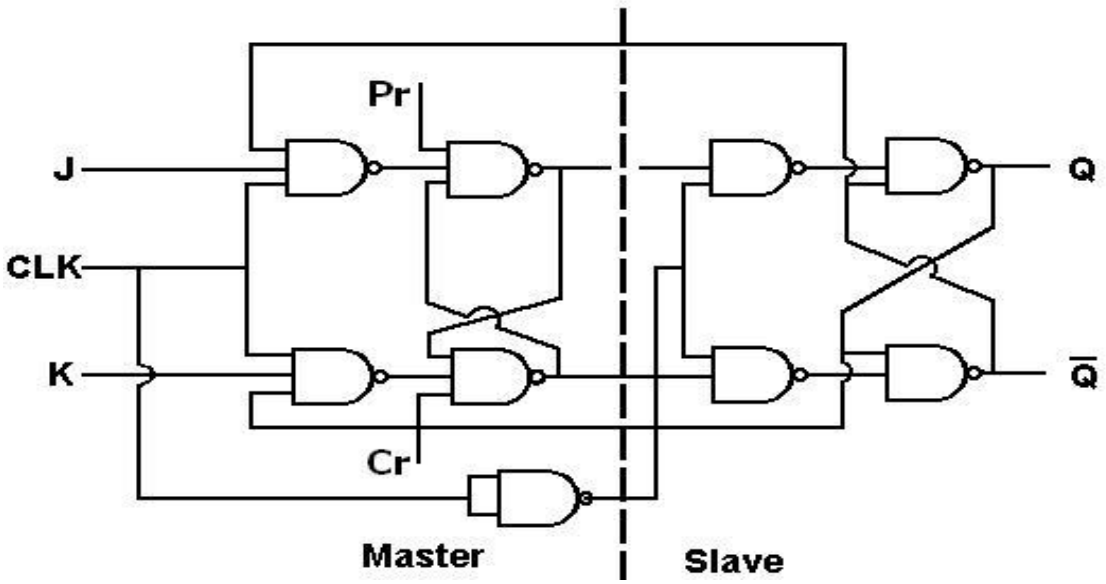
(A) LOGIC DIAGRAM



(B) SYMBOL

TRUTH TABLE

S	R	Q+	$\overline{Q}$b+
0	0	Q	$\overline{Q}$b
0	1	0	1
1	0	1	0
1	1	0*	0*



Truth Table:

$\overline{\text{Preset}}$	$\overline{\text{Clear}}$	J	K	Clock	Q_{n+1}	$\overline{Q_{n+1}}$	Status
0	1	X	X	X	1	0	Set
1	0	X	X	X	0	1	Reset
1	1	0	0	⎍	Q_n	$\overline{Q_n}$	No Change
1	1	0	1	⎍	0	1	Reset
1	1	1	0	⎍	1	0	Set
1	1	1	1	⎍	$\overline{Q_n}$	Q_n	Toggle

(b) D Flip-Flop

Symbol

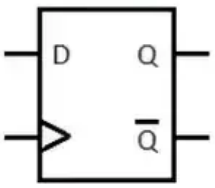
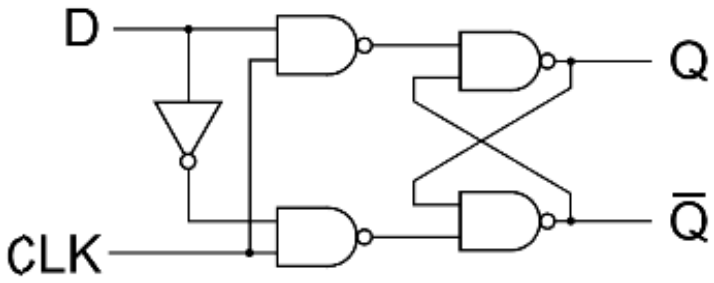


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Logic diagram:



(c)T Flip Flop

Symbol

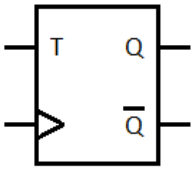
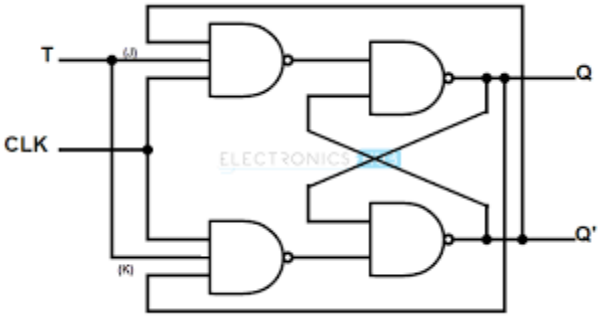


Table of truth:

T	Q	$\overline{Q}$
0	Q	$\overline{Q}$
1	$\overline{Q}$	Q
0	$\overline{Q}$	Q
1	Q	$\overline{Q}$

Logic Diagram:



Result: Realized Flip Flops using NAND Gates

EXPERIMENT NO-8

Realize the following shift registers using IC7474/7495
(i) SISO (ii) SIPO (iii)) PISO(iv))PIPO (v) Ring (vi) Johnson counter

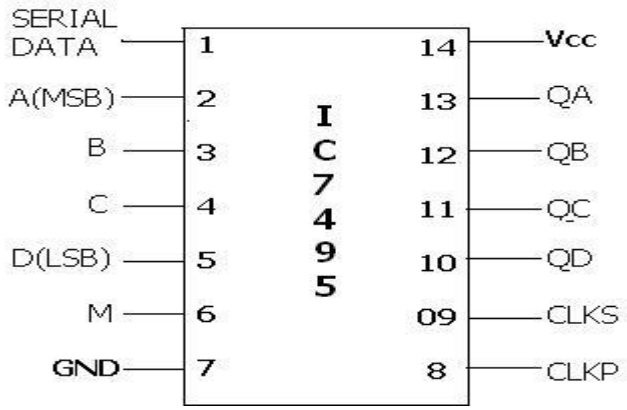
Aim: To realise SIPO, SISO, PISO, PIPO operations Ring and Johnson counters using IC7495/7474

Theory:

Group of flip flops are called as registers. The basic operation of a register is to store information in the form of bits. A shift register is a group of flip-flops arranged in such a manner that binary numbers stored in the flip flops can be shifted from one flip flop to another for every clock pulse.

IC 7495 is an universal 4-bit shift register (consists of 4-flip flops) that can accept data either serially or parallel and can perform left shift or right shift of the information.

Pin Diagram of IC 7495:



M=1 for parallel operation

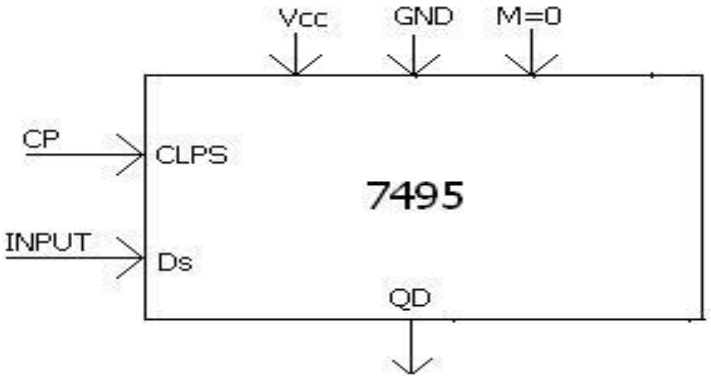
M=0 for serial operation

(a) SERIAL IN SERIAL OUT (SISO)-shift right

Truth Table

Clk	TIME	Qa	Qb	Qc	Qd
clks	T0 1	1			
	T1 0	0	1		
	T2 1	1	0	1	
	T3 1	1	1	0	1
	T4	x	1	1	0
	T5	x	x	1	1
	T6	x	x	X	1

Logic Diagram:



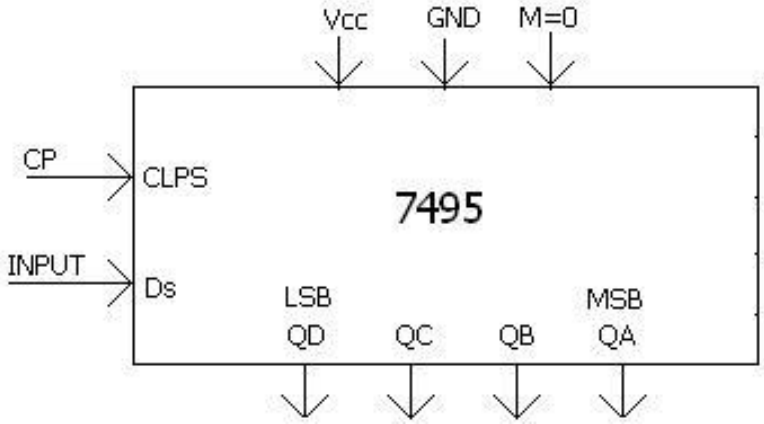
SISO:-

1. M=0, clks-> CP,
2. Input is given at DS[give 1 or 0 at DS press the mono pulsar]
3. The o/p shifts right QA, to QD.
4. After the 4th clock pulse o/p is seen at QA, QB, QC,QD.
5. Continue pressing the mono pulse , o/p is seen at QD.

(a) SERIAL IN PARALLEL OUT (SIPO) Truth Table

Time	Serial data	Qa	Qb	Qc	Qd
T0	1	1			
T1	0	0	1		
T2	1	1	0	1	
T3	1	1	1	0	1

Logic Diagram:



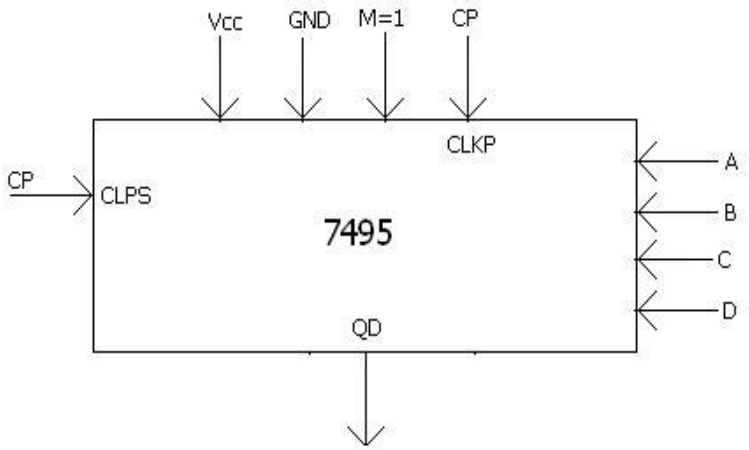
SIPO:-

After the 4th clock pulse o/p is seen at. QA, QB, QC,QD.

Example if i/p is 1011 o/p is 1101.

(c) PARALLEL IN SERIAL OUT (PISO)

Logic Diagram:



To check the serial out, M is made to 0 and clk is given clock pulse

Truth Table:

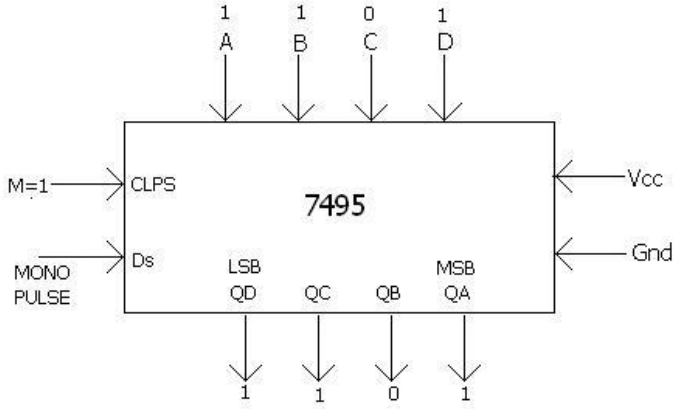
Clk	TIME	Qa	Qb	Qc	Qd
clks	T0	1	1	0	1
	T1	X	1	1	0
	T2	X	X	1	1
	T3	X	x	x	1

PISO:-

1. M=1, clk-> CP, clks- >1.
2. Load the parallel data ABCD , which gets stored in QA, QB, QC,QD.
3. Clks- >CP, M=0.
4. Output is observed at QD.

(d) PARALLEL IN PARALLEL OUT (PIPO)

Logic Diagram



PIPO:-

1. M=1, clk-> CP
2. Give the data through ABCD , give CP
3. The o/p is stored in . QA, QB, QC,QD.

Procedure:

1) Rig up the circuit as shown in the diagram.
2) Apply the input to Shift register as per the Truth table and observe the o/p Verify with the truth table.

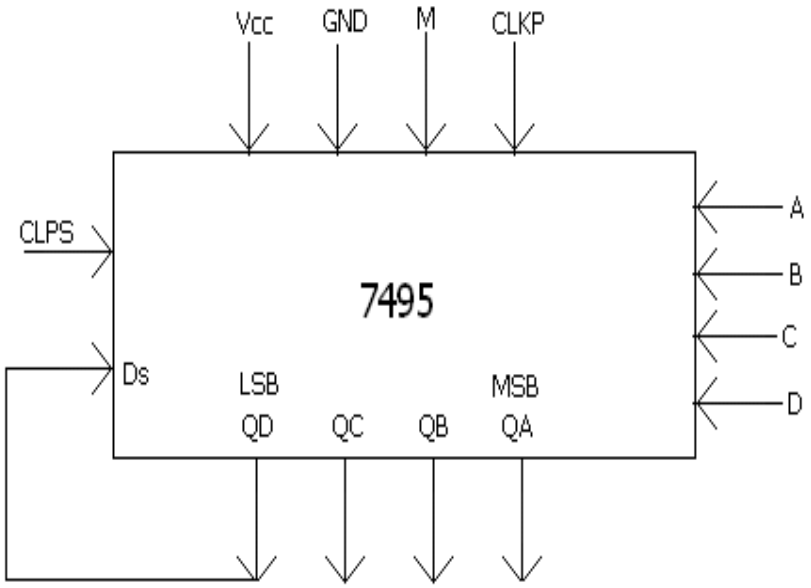
Result: Realized truth table of all shift register.

(e)RING COUNTER

Truth Table

CP	QA	QB	QC	QD
t0	1	0	0	0
t1	0	1	0	0
t2	0	0	1	0
t3	0	0	0	1
t4	1	0	0	0

Logic Diagram



Procedure:

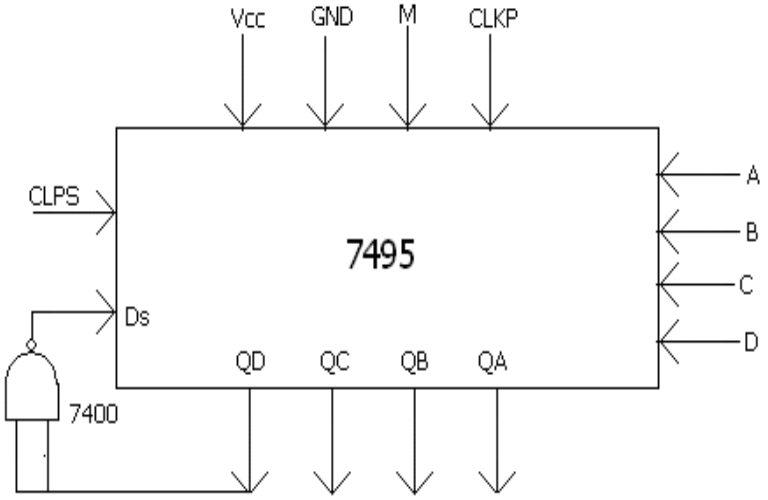
(1). Rig up the circuit as shown in the diagram, D_S is not given as input.
(2). Load data parallely with clock pulse and M=1
(3). Then make M=0,Clks-cp
(4). Verify the working of a ring counter.

(f)Johnson Counter Using IC-7495

Truth Table

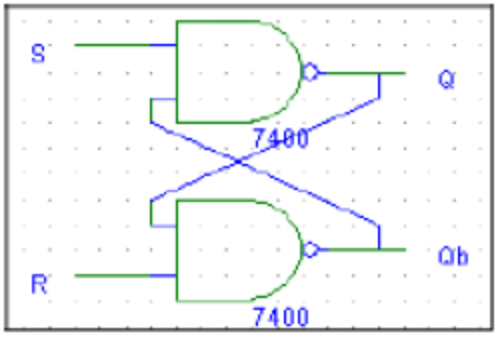
CP	QA	QB	QC	QD
t0	1	0	0	0
t1	1	1	0	0
t2	1	1	1	0
t3	1	1	1	1
t4	0	1	1	1
t5	0	0	1	1
t6	0	0	1	1
t7	0	0	0	1
t8	1	0	0	0

Logic Diagram



S	R	Q+	$\overline{Q}$b+
0	0	Q	$\overline{Q}$b
0	1	0	1
1	0	1	0
1	1	0*	0*

$\overline{SR}$ LATCH:



TRUTH TABLE

S	R	Q+	$\overline{Q}$b+
0	0	1*	1*
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$b

EXPERIMENT NO-9

Realize (i) Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop
(ii) Mod-N Counter using IC7490 / 7476
(iii) Synchronous counter using IC74192

Aim: Design Mod – N Synchronous Up Counter & Down Counter using 7476 JK Flip-flop

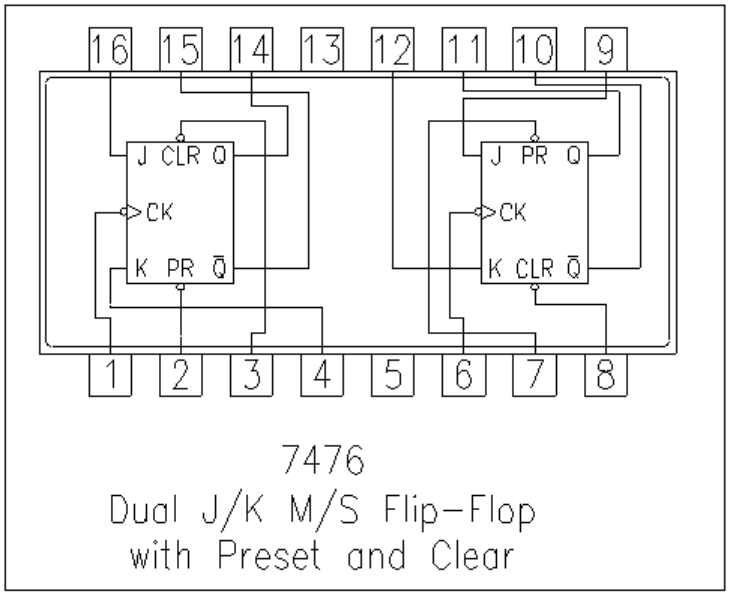
Theory: Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.

- Asynchronous or ripple counters.
- Synchronous counters.

Synchronous counters

If the "clock" pulses are applied to all the flip-flops in a counter simultaneously, then such a counter is called as synchronous counter.

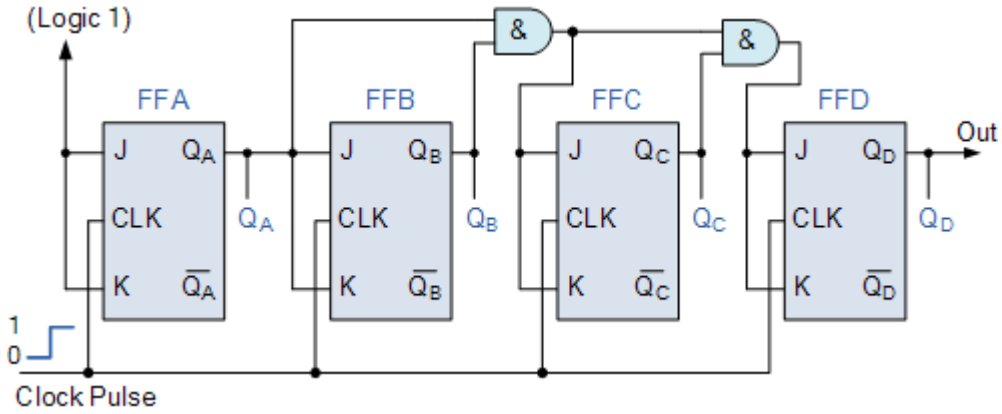
Pin Diagram:



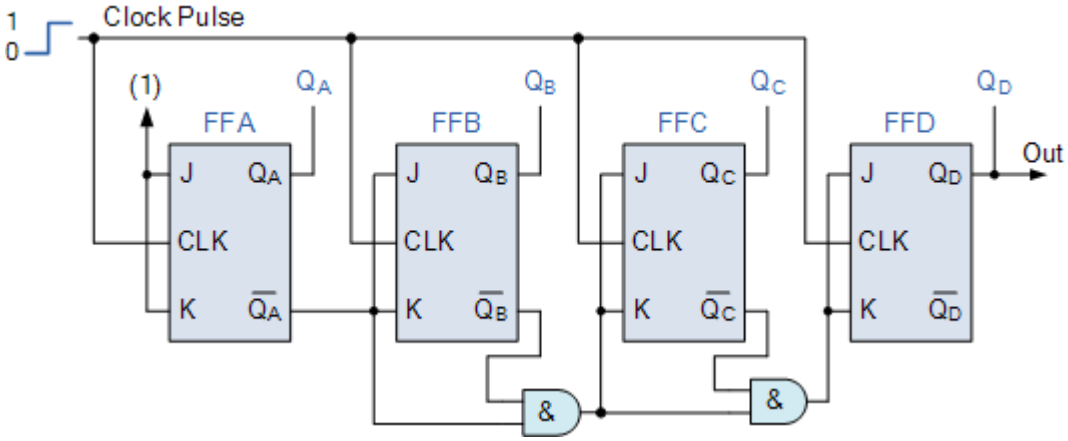
Truth Table : Up Counter

State	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	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
0	0	0	0	0

Logic Diagram: Binary 4-bit Synchronous Up Counter



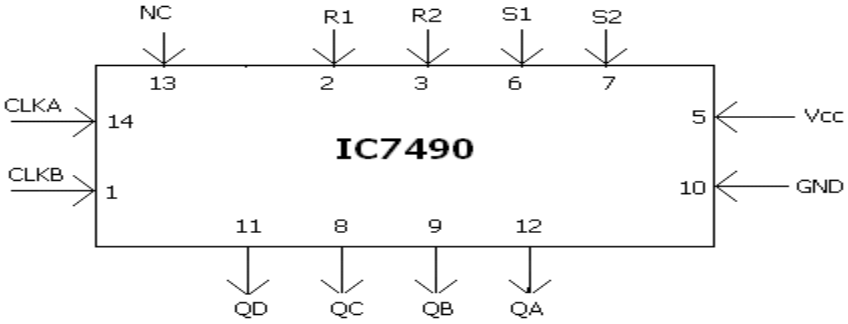
Logic Diagram: Binary 4-bit Synchronous Down Counter



Truth Table:

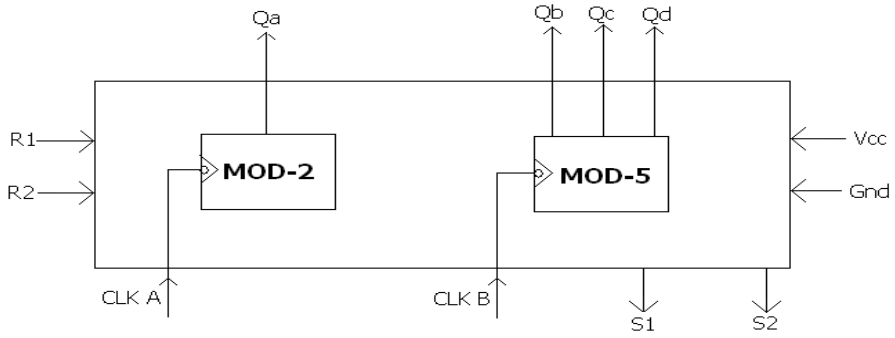
CLK	QD	QC	QB	QA
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
7	1	0	0	0
8	0	1	1	1
9	0	1	1	0
10	0	1	0	1
11	0	1	0	0
12	0	0	1	1
13	0	0	1	0
14	0	0	0	1
15	0	0	0	0

i) Realization of MOD – N Counters Using IC7490



Pin Diagram of IC7490:

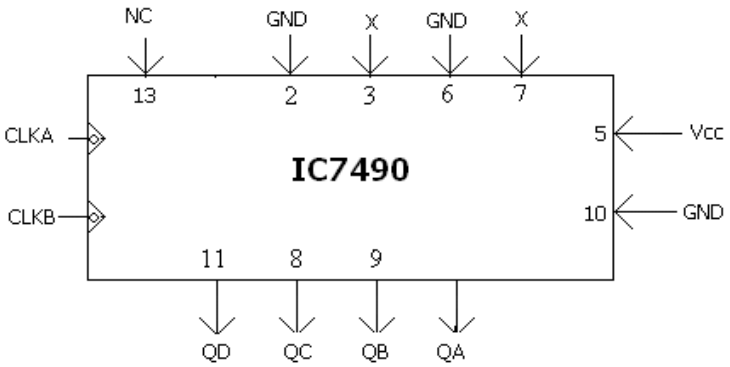
Internal Diagram:



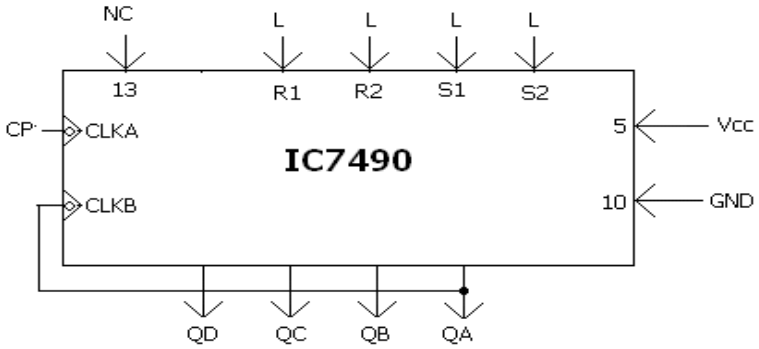
Truth Table:

R1	R2	S1	S2	Qd	Qc	Qb	Qa
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			

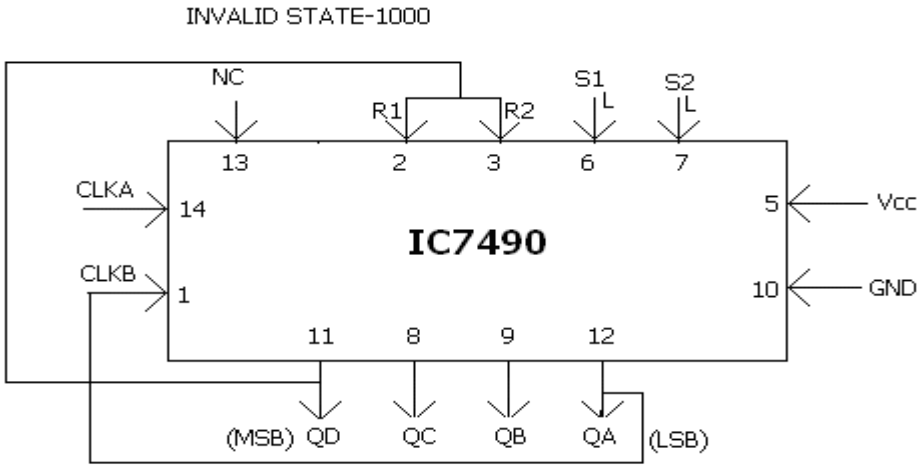
(a) 7490 AS MOD-2 / MOD-5 COUNTER



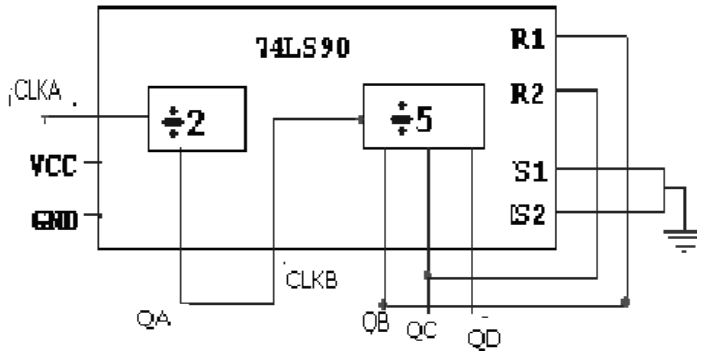
(b) 7490 AS MOD-10 COUNTER



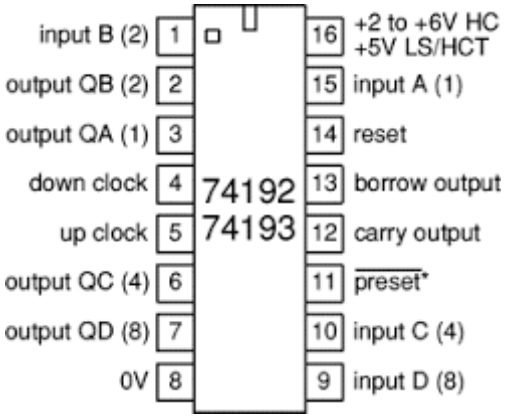
(c) 7490 AS MOD-8 COUNTER



(d) 7490 AS MOD-6 COUNTER



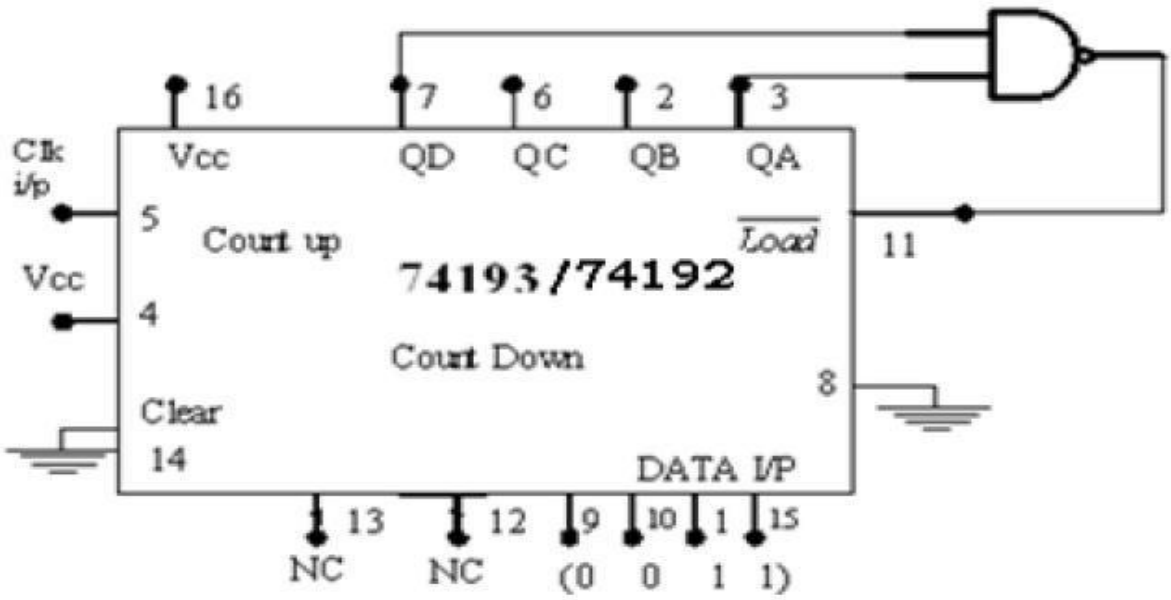
ii) Synchronous counter using IC74192



Procedure:

1) Rig up the circuit as shown in the diagram.
2) Apply the inputs to these counters as per the Truth table and observe the o/p verify with the truth table.

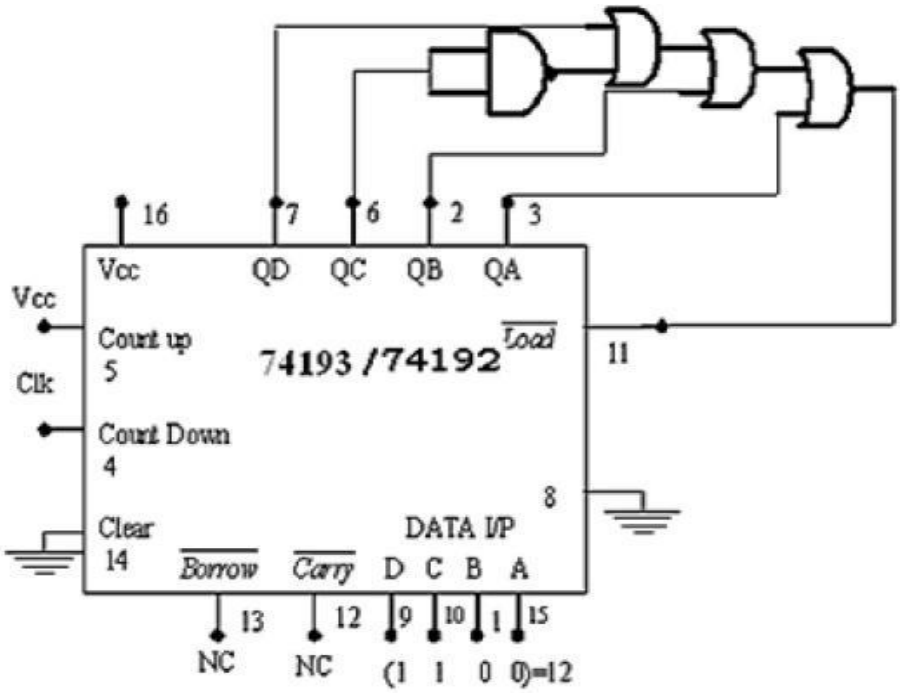
Logic diagram: **Count up from 3 to 8**



Truth table:

Clk	QD	QC	QB	QA
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	0	0	1	1

Logic Diagram: Design down counter 12 to 5



Truth Table:

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

Result: Realized truth table of all Counters.

EXPERIMENT NO-10

Aim: Design Pseudo Random sequence generator using 7495

Theory: **Pseudo Random Number Generator** (PRNG) refers to an algorithm that uses mathematical formulas to produce **sequences** of **random** numbers. ... Many numbers are generated in a short time and can also be reproduced later, if the starting point in the **sequence** is known. Hence, the numbers are deterministic and efficient.

Procedure:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- By Keeping mode=1. Load the input A,B,C,D as in Truth Table 1st Row and give a clock pulse
- For count mode make mode = 0.
- Verify the Truth Table and observe the outputs.

DESIGN 1:

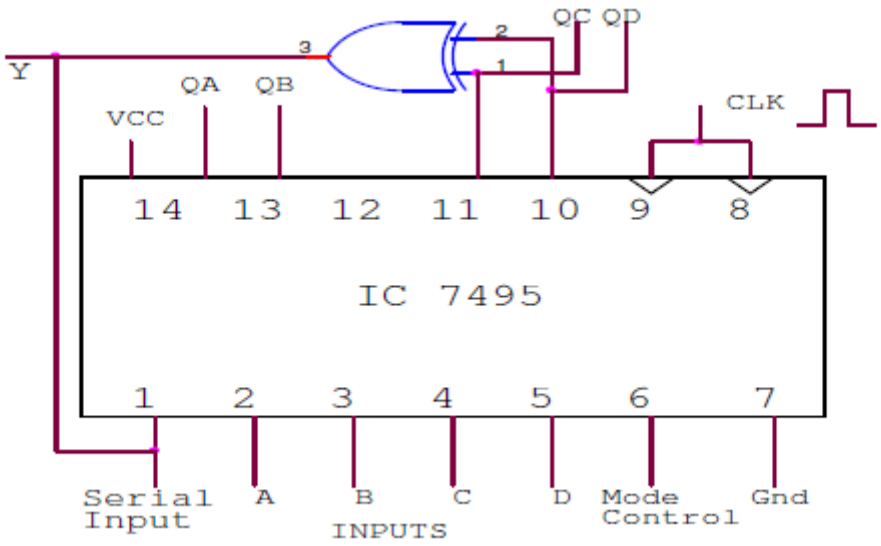
Sequence = 100010011010111

Sequence length S = 15

$Y = Q_C (+) Q_D$

Q_A	Q_B	Q_C	Q_D	Y
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
	1	1	1	
		1	1	
			1	

X	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

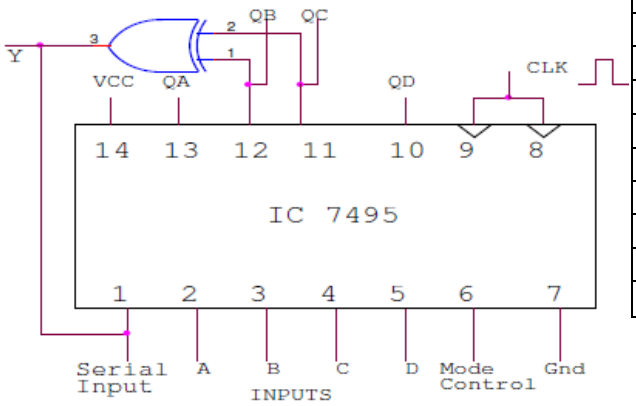


DESIGN 2:

Sequence = 1001011

Sequence length S = 7

$Y = Q_B (+) Q_C$



Q_A	Q_B	Q_C	Q_D	Y
1	1	1	1	0
0	1	1	1	0
0	0	1	1	1
1	0	0	1	0
0	1	0	0	1
1	0	1	0	1
1	1	0	1	1
	1	1	0	
		1	1	
			1	

X	X	1	X
0	X	0	X
X	1	X	0
X	1	X	1

Result: Verified the Pseudo Random sequence generator using 7495

EXPERIMENT NO-11

Design Serial Adder with Accumulator and Simulate using Simulation tool.

Aim: To Simulate **Serial adder with Accumulator** using simulation tool.

Theory: http://cc.ee.ntu.edu.tw/~jhjiang/instruction/courses/fall12-ld/unit18.pdf

Procedure: https://www.youtube.com/watch?v=FVlPpMrMQRA

Result: Simulated Serial adder with Accumulator using Multisim

EXPERIMENT NO-12

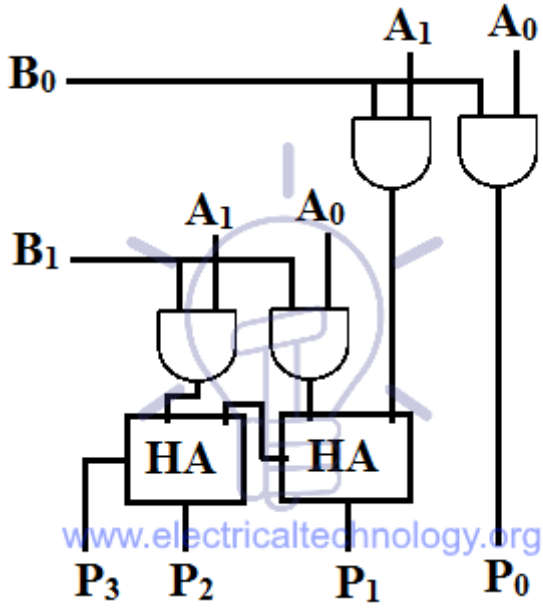
Design Binary Multiplier and Simulate using Simulation tool.

Aim: To Simulate **Binary Multiplier** using simulation tool.

Theory:

A **binary multiplier** is a combinational logic circuit or digital device used for multiplying two **binary** numbers. The two numbers are more specifically known as multiplicand and **multiplier** and the result is known as a product. The multiplicand & **multiplier** can be of various bit size.

Logic Diagram:



This multiplier can multiply two numbers having bit size = 2 i.e. the multiplier and multiplicand can be of 2 bits. The product bit size will be the sum of the bit size of the input i.e. **2+2=4**. The maximum range of its output is **3 x 3 = 9**. So we can accommodate decimal 9 in 4 bits. It is another way of finding the bit size of the product.

Suppose multiplicand A_1 A_0 & multiplier B_1 B_0 & P_3 P_2 P_1 P_0 as a product of the 2×2 multiplier.

First, multiplicand A_1A_0 is multiplied with LSB B_0 of the multiplier to obtain the partial product. This is obtained using AND gates. Then the same multiplicand is multiplied (AND) with the 2nd LSB to get the 2nd partial product. The multiplicand is multiplied with each bit of the multiplier (from LSB to MSB) to obtain partial products.
The number of partial products is equal to the number of bit size of the multiplier. In 2×2 multiplier, multiplier size is 2 bits so we get 2 partial products.

Result: Simulated Binary Multiplier using simulation tool.

VIVA QUESTIONS

1. What do you mean by Logic Gates?
2. What are the applications of Logic Gates?
3. What is Truth Table?
4. Why we use basic logic gates?
5. Write down the truth table of all logic gates?
6. What do you mean by universal gate?
7. Write truth table for 2 I/P OR, NOR, AND and NAND gate?
8. Implement all logic gate by using Universal gate?
9. Why is they called Universal Gates?
10. Give the name of universal gate?
11. Draw circuit diagram of Half Adder circuit?
12. Draw circuit diagram of Full Adder circuit?
13. Draw Full Adder circuit by using Half Adder circuit and minimum no. of logic gate?
14. Write Boolean function for half adder? Q.5 Write Boolean function for Full adder?
15. Design the half Adder & Full Adder using NAND-NAND Logic.
16. Draw circuit diagram of Half Subtractor circuit?
17. Draw circuit diagram of Full Subtractor circuit?
18. Draw Full Subtractor circuit by using Half Subtractor circuit and minimum no. of logic gate?
19. Write Boolean function for half Subtractor?
20. Write Boolean function for Full Subtractor?
21. What is Excess-3 code? Why it is called Excess-3 code?
22. What is the application of Excess-3 Code?
23. What is ASCII code?
24. Excess-3 code is Weighted or Unweighted?
25. Out of the possible 16 code combination? How many numbers used in Excess-3 code?
26. What is Demorgan's Law?

27. Show the truth table for Demorgan's Theorem?
28. What is Minterm & Maxterm?
29. How Min term can be converted in Max term?
30. What is Hybrid function?
31. What is Flip-Flop?
32. What is Latch circuit?
33. Draw a truth –tables of S-R, J-K, D and T?
34. What is the disadvantages of S-R Flip-Flop?
35. How can you remove the problem of S-R Flip –Flop?
36. Make circuit diagram of S-R, J-K, D and T Flip-Flop?
37. What do you understand by Race Aground condition? How it is over come in J-K Flip Flop?
38. Explain the principle of Multiplexer?
39. Draw a circuit diagram of 4: 1 Multiplexer?
40. What are the advantages of Multiplexer?
41. What are the disadvantages of Multiplexer?
42. Make the Truth-table of Multiplexer?
43. Explain about Demultiplexer?
44. Draw a circuit diagram of 1: 4 Demultiplexer?
45. Make a logic diagram of 1: 4 Demultiplexer?
46. What is the application of Demultiplexer?
47. Define Registers with example.
48. Define Counters with example.
49. Explain ripple, asynchronous and synchronous counters.
50. Explain Race Around Condition.