BHARATI VIDYAPEETH DEEMED UNIVERSITY
COLLEGE OF ENGINEERING, PUNE - 43

DEPARTMENT OF COMPUTER ENGINEERING

Lab Manual

Digital Techniques and Logic Design

B.Tech. Computer Sem III

Name of Faculty: Gauri Rao

VISION OF THE DEPARTMENT

To pursue and excel in the endeavor for creating globally recognized computer engineers through quality education.

MISSION OF THE DEPARTMENT

a) To impart engineering knowledge and skills confirming to a dynamic curriculum.
b) To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.
c) To produce qualified graduates exhibiting societal and ethical responsibilities in working environment.

Program Educational objectives

The B. Tech Computer Engineering Programme graduates upon completion of this Programme will be able to:

a) Demonstrate technical and professional competencies by applying engineering fundamentals, computing principles and technologies.
b) Learn, Practice, and grow as skilled professionals/ entrepreneur/researchers adapting to the evolving computing landscape.
c) Demonstrate professional attitude, ethics, understanding of social context and interpersonal skills leading to a successful career.

Program Outcomes

a) To apply knowledge of computing and mathematics appropriate to the domain.
b) To logically define, analyze and solve real world problems.
c) To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs.
d) To interpret and analyze data for providing solutions to complex engineering problems.
e) To use and practice engineering and IT tools for professional growth.
f) To understand and respond to legal and ethical issues involving the use of technology for societal benefits.
g) To develop societal relevant projects using available resources.
h) To exhibit professional and ethical responsibilities.

Course Name: - Digital signal processing
Hours Per Week: 2 Hrs.

Course Outcomes

1. Comprehend different number systems and Boolean algebraic principles.
2. Apply Boolean algebra to simplify and apply design logic.
3. Analyze a given digital system involving combinational circuit elements.
4. Design and synthesize a system with sequential circuit elements.
5. Understand structure and characteristics of Memory.
6. Validate and implement the PLD based designs using both schematic capture and VHDL.

CO-PO mapping

PO →	a	b	c	d	e	f	g	h	i	j	k	l
CO1	H	M	M	M	H			M	L	M		M
CO2	H	H	H	H	H			M	L	M		M
CO3	M	H	H	H	H		H	M	M	M		M
CO4	M	H	H	H	H		H	M	M	M		M
CO5	L						L	M	L	M		M
CO6	L	M	M	M	M			M	L	M		H

LIST OF PRACTICAL Assignments:

No	Detail of Assignment
1.	Verify truth tables of logic gates. (AND, OR, XOR, NOT, NAND, NOR). Simplify the given Boolean expression using K-map and implement using gates.
2.	State De-Morgan's theorem and write Boolean laws. Implement NAND and NOR as Universal gates.
3.	Design (truth table, K-map) and implement half and full adder /subtractor.
4.	Design (truth table, K-map) and implement 4-bitBCD to Excess-3 and Excess-3 to BCD Code converters.
5.	Implement of logic functions using multiplexer IC 74151 (Verification, cascading & logic function implementation)
6.	Implement logic functions using 3:8 decoder IC 74138.
7.	Design (State diagram, state table & K map) and implement 3 bit Up and Down Asynchronous and Synchronous Counter using JK flip-flop .
8.	Design and implement modulo 'n' counter with IC 7490.
9.	Design and implementation of Shift Register using IC 7495.
10.	VHDL Programming: - Simulation of Full adder using behavioural & structural modelling.
11.	Mini project

EXPERIMENT NO: 1

Aim: - Verify truth tables of logic gates. (AND, OR, XOR, NOT, NAND, NOR). Simplify the given Boolean expression using K-map and implement using gates.

APPARATUS REQUIRED: Digital Trainer Kit., Connecting Leads, IC's (7400, 7402, 7404, 7408, 7432, and 7486)

BRIEF THEORY:

AND Gate: The AND operation is defined as the output as (1) one if and only if all the inputs are (1) one. 7408 is the two Inputs AND gate IC.A&B are the Input terminals &Y is the Output terminal.

$$Y = A.B$$

OR Gate: The OR operation is defined as the output as (1) one if one or more than 0 inputs are (1) one. 7432 is the two Input OR gate IC. A&B are the input terminals & Y is the Output terminal.

$$Y = A + B$$

NOT GATE: The NOT gate is also known as Inverter. It has one input (A) & one output (Y). IC No. is 7404. Its logical equation is,

$$Y = A \text{ NOT } B,\ Y = A'$$

NAND GATE: The IC no. for NAND gate is 7400. The NOT-AND operation is known as NAND operation. If all inputs are 1 then output produced is 0. NAND gate is inverted AND gate.

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

NOR GATE: The NOR gate has two or more input signals but only one output signal. IC 7402 is two I/P IC. The NOT- OR operation is known as NOR operation. If all the inputs are 0 then the O/P is 1. NOR gate is inverted OR gate.

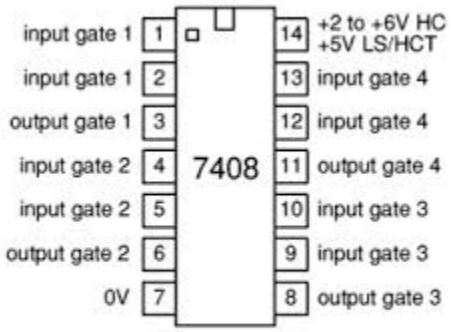
$$Y = (A+B)'$$

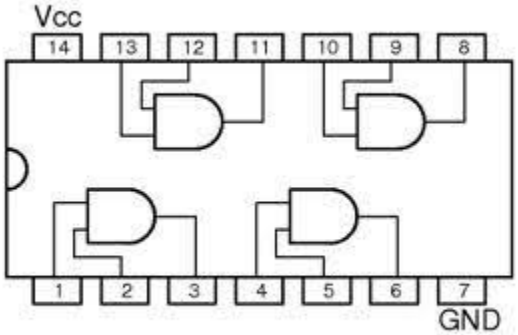
EX-OR GATE: The EX-OR gate can have two or more inputs but produce one output. 7486 is two inputs IC. EX-OR gate is not a basic operation & can be performed using basic

PROCEDURE:

1. Fix the IC's in the socket on trainer kit..
2. Connect the Vcc to pin 14 & Gnd to pin 7.
3. Connect inputs to switches and outputs to LED as per pin diagram.
4. Note the values of output for different combination of inputs in the truth table.

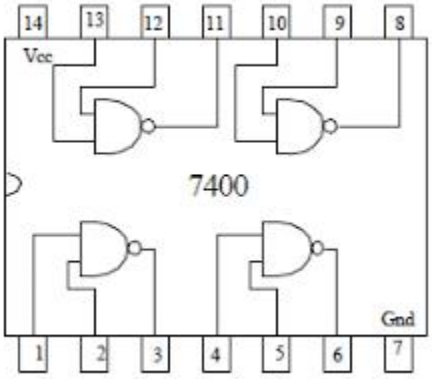
1. AND gate

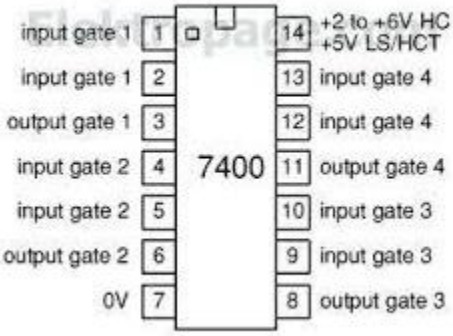




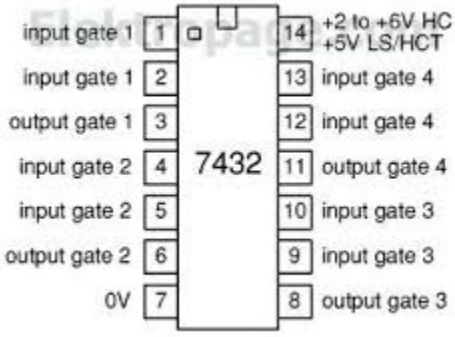
7408 Quad 2 Input AND

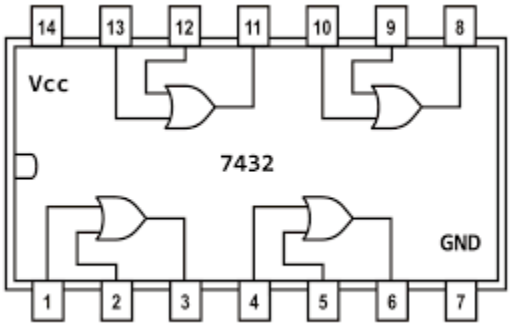
2. NAND gate



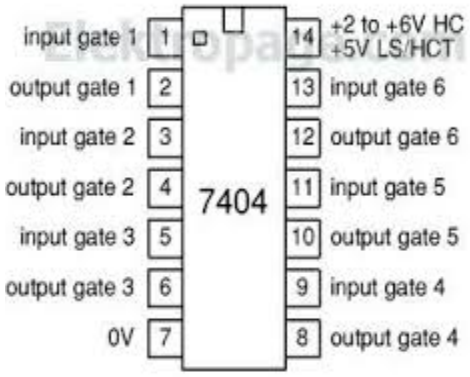


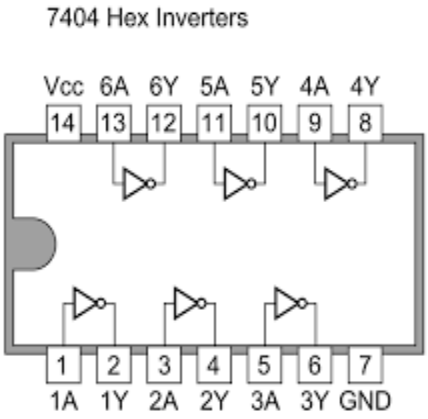
3. OR Gate



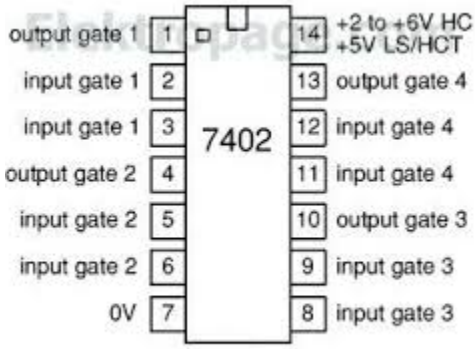


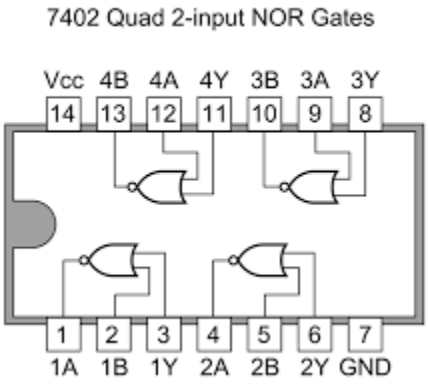
4. NOT Gate-



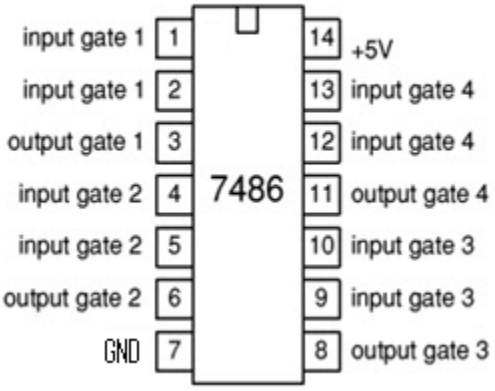


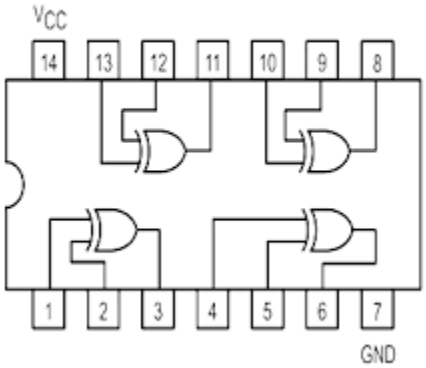
5. NOR gate-





6. EXOR Gate-



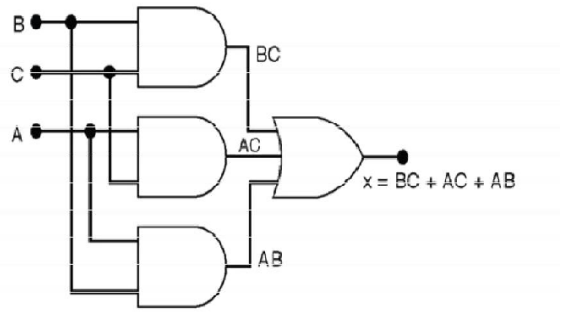


OBSERVATIO N TABLE:

INP UTS		OUTP UTS					
A	B	A' NOT	A+B OR	(A+B)' NOR	(A *B) A ND	(A*B)' NAND	(A B) Ex-OR
0	0	1	0	1	0	1	0
0	1	1	1	0	0	1	1
1	0	0	1	0	0	1	1
1	1	0	1	0	1	0	0

Example- f (A, B, C, D) =A'BC+AB'C+ABC'+ABC (SOP)

Reduced form is BC+AC+AB



Conclusion:- Verified the truth tables of all gates and implemented the given Boolean expression using logic gates.

Questions:-

1. Determine 2's complement of 1011.
2. Write truth table for 3 input EXOR gate.
3. Simplify using K-map :- f(A,B,C,D)= $\pi\, M(0,1,2,6,8,9,10)$ and implement using logic gates.
4. Realize the following expression using NAND gates only.
   Y=(AB +BC) C
5. Construct NAND, NOR and EXOR gate using basic gates.

EXPERIMENT NO: 2

Aim: - State De-Morgan's theorem and write Boolean laws. Implement NAND and NOR as Universal gates.

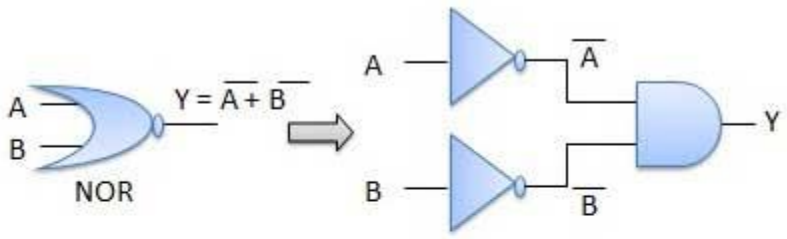
APPARATUS REQUIRED: Digital Trainer Kit., Connecting Leads, IC's (7400, 7402, 7404, 7408, and 7432)

BRIEF THEORY:

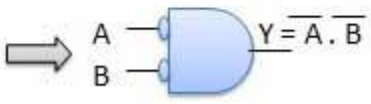
$\overline{A.B} = \overline{A} + \overline{B}$

NAND = Bubbled OR

A	B	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0



NOR ☰ Bubbled AND



Bubbled AND

$\overline{A+B} = \overline{A}.\overline{B}$

NOR = Bubbled AND

A	B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

PROCEDURE:

1. Fix the IC's in the socket on trainer kit..
2. Connect the Vcc to pin 14 & Gnd to pin 7.
3. Connect inputs to switches and outputs to LED as per pin diagram.
4. Note the values of output for different combination of inputs in the truth table.

Conclusion: - Verified the truth tables for De' Morgan's theorem.

Questions:-

1. Why NAND and NOR are called Universal gates.
2. State and prove De-Morgan's theorem and write Boolean laws.
3. Simplify Y=(AB +C) (AB+ D)
4. State associative distributive and commutative laws.
5. Prove that A+ AB = A

EXPERIMENT NO: 3

Aim: - Design and implement half and full adder /subtractor.

APPARATUS REQUIRED: Digital Trainer Kit., Connecting Leads, IC's (7400, 7402, 7404, 7408, and 7432)

THEORY:

Half Adder: It is a logic circuit that adds two bits. It produces the O/P, sum & carry. The Boolean equation for sum & carry are:

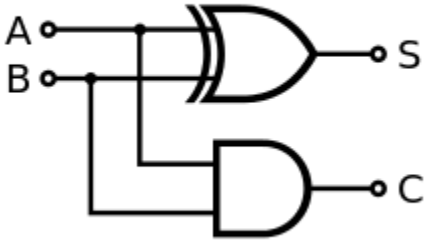
$$SUM = A + B$$
$$CARRY = A.\ B$$

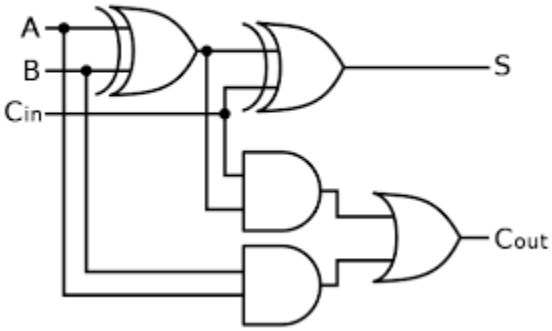
Therefore, sum produces 1 when A&B are different and carry is 1when A&B are 1. Application of Half adder is limited.

Full Adder: It is a logic circuit that can add three bits. It produces two O/P sum & carry. The Boolean Equation for sum & carry are:

$$SUM = A + B + C$$
$$CARRY = A.B + (A+B)\ C$$

Therefore, sum produces one when I/P is containing odd no's of one & carry is one when there are two or more one in I/P.





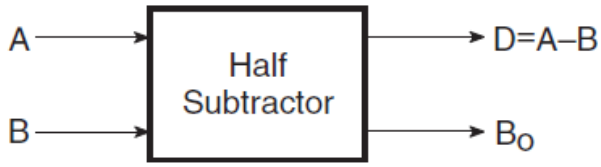
Half Adder

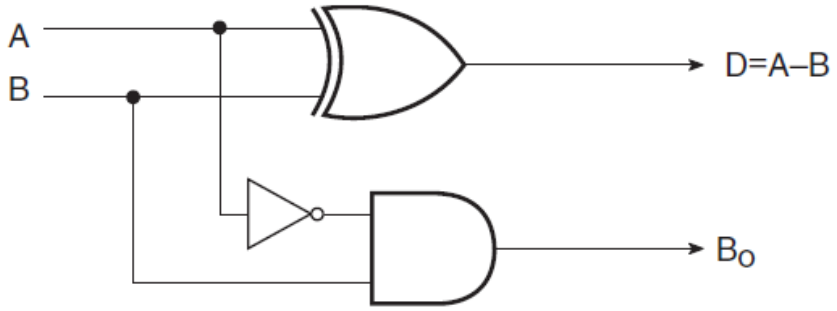
INPUTS		OUTPUT	
A	**B**	**S**	**C**
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

Full Adder

INPUTS			OUTPUTS	
A	**B**	**C**	**S**	**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**

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0





Conclusion- Truth tables for Half and Full Adder/Subtractor is verified.

Questions:-

1. Design full adder using two half adders.
2. Perform following addition 1101+ 1011
3. Subtract using 2's complement method 9 – 6 in binary.
4. Design BCD adder using 4 bit binary adders.
5. Describe parallel adders.

EXPERIMENT NO: 4

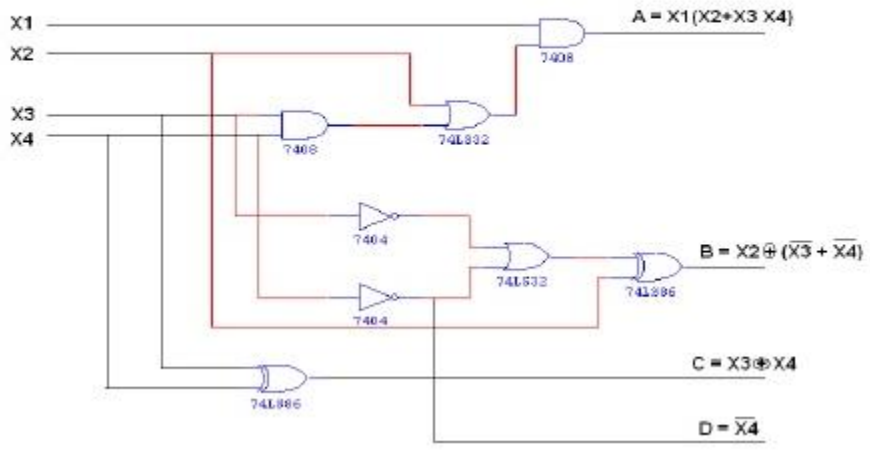
Aim: - Design and implement 4-bitBCD to Excess-3 and Excess-3 to BCD Code converters.

APPARATUS REQUIRED: Digital Trainer Kit., Connecting Leads, IC's (7400, 7402, 7404, 7408, 7432, and 7486)

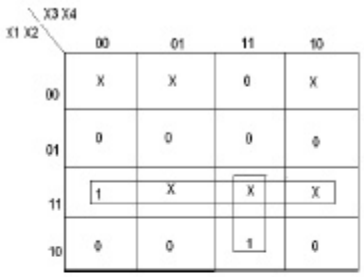
Theory:-

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

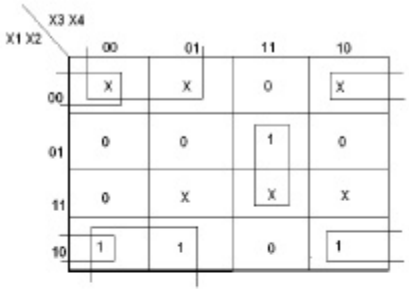
EXCESS-3 TO BCD CONVERTOR:



K-Map for A:



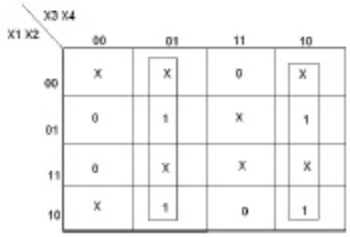
K-Map for B:



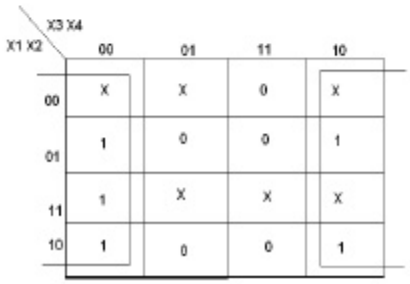
A = X1 X2 + X3 X4 X1

B = X2 ⊕ ($\overline{X3}$ + $\overline{X4}$)

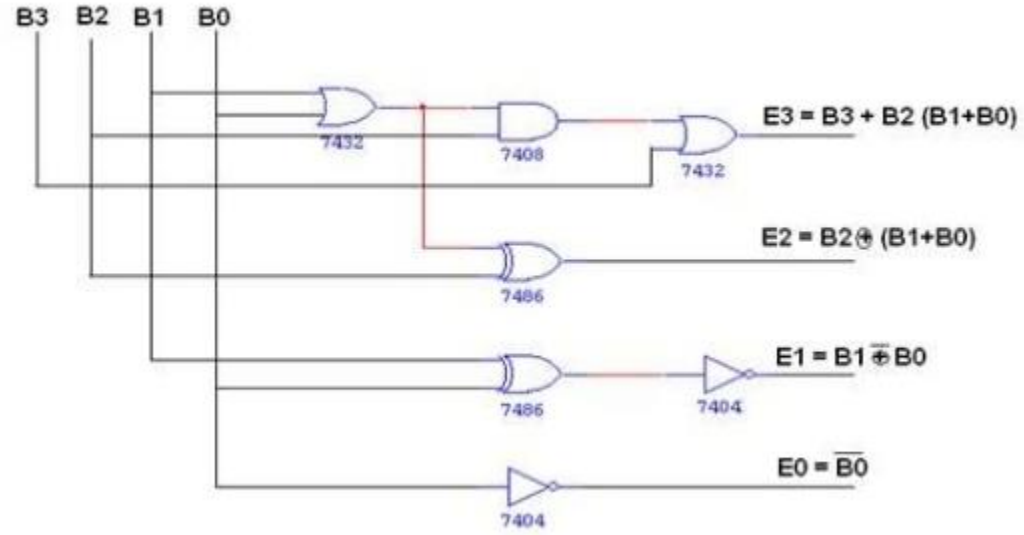
K-Map for C:



K-Map for D:



BCD to Excess-3 code converter



AB\CD	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

E3=B3+ B2(B1+B0)

AB\CD	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

E2 = B2 xor B1+B0

AB\CD	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
10	1		X	X

E1= B1 XNOR B0

AB\CD	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

E3= $\overline{B0}$

Conclusion- Code converters are implemented and Truth tables are verified.

Questions:-

1. State rules for BCD addition.
2. List the steps to convert from BCD to Excess-3.
3. Represent the decimal number 237 in BCD and Excess-3
4. Design and implement Binary to Gray code converter.
5. Which code is used in K-map design? Why?

EXPERIMENT NO: 5

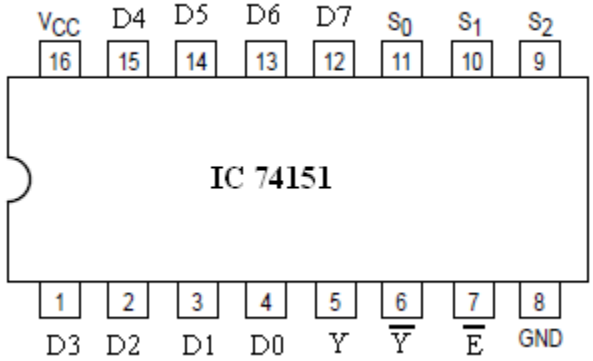
Aim: - Implement of logic functions using multiplexer IC 74151 (Verification, cascading & logic function implementation)

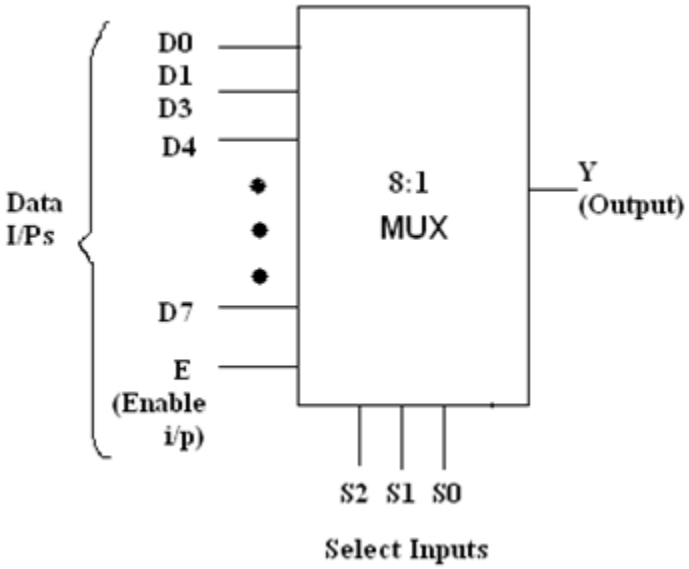
APPARATUS REQUIRED: Digital Trainer Kit., Connecting Leads, IC 74153

THEORY:

MULTIPLEXER: Multiplexer generally means many into one. A multiplexer is a circuit with many Inputs but only one output. By applying control signals we can steer any input to the output .The fig. (1) Shows the general idea. The circuit has n input signal, m select lines & one output signal. Where 2^m= n. For 8:1 multiplexer, which has 8 input lines, 3 select lines & 1 output.

PIN CONFIGURATION;–

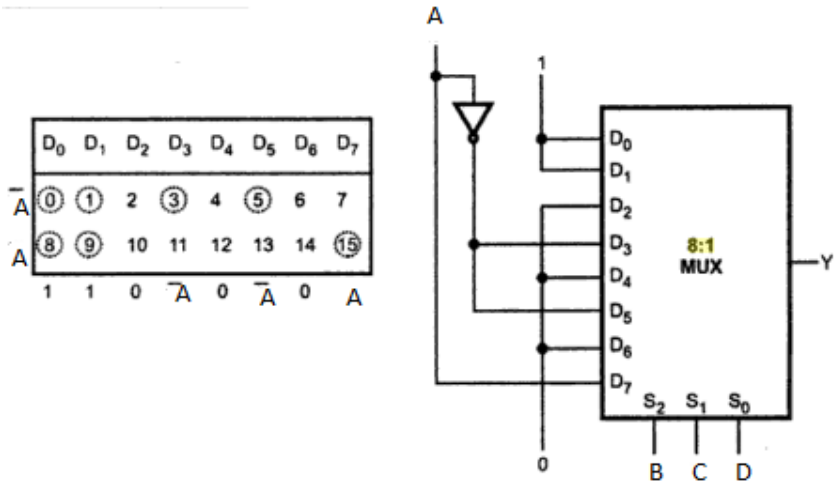




Enable	Select Inputs			Output
E	S2	S1	S0	Y
0	×	×	×	0
1	0	0	0	D0
1	0	0	1	D1
1	0	1	0	D2
1	0	1	1	D3
1	0	0	0	D4
1	0	0	1	D5
1	0	1	0	D6
1	0	1	1	D7

Implement the following function using 8:1 Multiplexer.

F(A,B,C,D)=∑m(0,1,3,5,8,9,15)



Conclusion- Verified the truth table for 8:1 Multiplexer and implemented the given function.

Questions:-

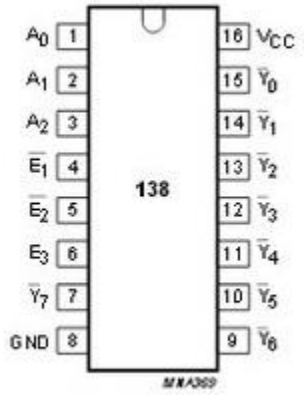
1. Design 16:1 MUX using 4:1 MUX with enable inputs.
2. State applications of multiplexer and demultiplexer.
3. Can you design full adder using multiplexer?
4. Compare multiplexer with demultiplexer.
5. Implement following function using 8:1 MUX and some logic gates:
   F(A,B,C,D) = ∑m (0,1,3,4,8,9,15)

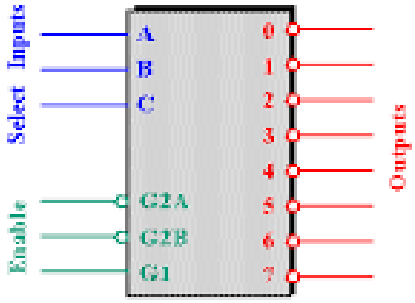
EXPERIMENT NO: 6

Aim: Implement logic functions using 3:8 decoder IC 74138.

APPARATUS REQUIRED: Digital Trainer Kit. Connecting Leads, IC 74138

THEORY: 74138 is a commonly used 3:8 decoder. 74138 is specifically designed for high speed memory decoders and data transmission systems. Pin diagram and truth table of 74138 is given below. It has three enable pins (G1, G2A, G2B), three select pins (A, B, C) and eight output pins (Y0 - Y7).





Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Conclusion- Verified the truth table for 3:8 decoder and implemented the given function.

Questions:-

1. Compare demultiplexer with decoder.
2. How Demultiplexer can be used as decoder?
3. Implement following function using 3:8 decoder and some logic gates:
   F(A,B,C,D) = ∑m (1,3,4,6,8,10,12,15)

EXPERIMENT NO: 7

Aim: Design (State diagram, state table & K map) and implement 3 bit Up and Down Asynchronous and Synchronous Counter using JK flip-flop.

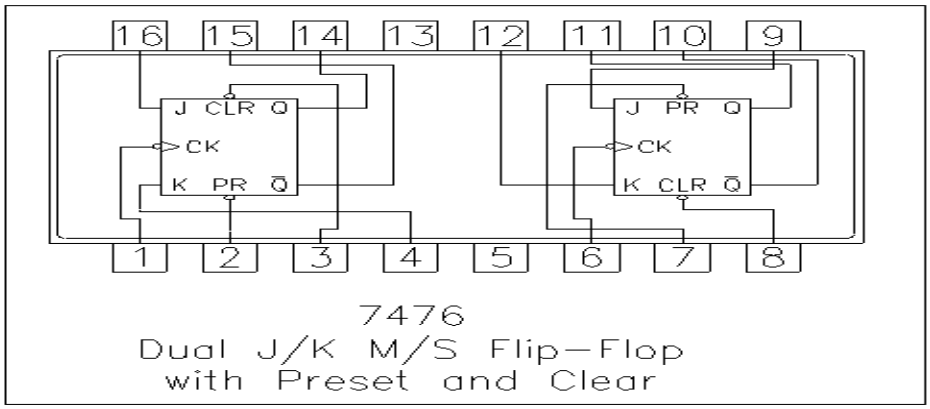
APPARATUS REQUIRED: Digital Trainer Kit. Connecting Leads, IC 7476

THEORY:

- **RS FLIP-FLOP:** There are two inputs to the flip-flop defined as R and S. When I/Ps R = 0 and S = 0 then O/P remains unchanged. When I/Ps R = 0 and S = 1 the flip-flop is switches to the stable state where O/P is 1 i.e. SET. The I/P condition is R=1 and S=0 the flip-flop is switched to the stable state where O/P is 0 i.e. RESET. The I/P condition is R = 1 and S = 1 the flip-flop is switched to the stable state where O/P is forbidden.
  JK FLIP-FLOP: For purpose of counting, the JK flip-flop is the ideal element to use. The variable J and K are called control I/Ps because they determine what the flip- flop does when a positive edge arrives. When J and K are both 0s, both AND gates are disabled and Q retains its last value.

- **D FLIP –FLOP**: This kind of flip flop prevents the value of D from reaching the Q output until clock pulses occur. When the clock is low, both AND gates are disabled D can change value without affecting the value of Q. On the other hand, when the clock is high, both AND gates are enabled. In this case, Q is forced to equal the value of D. When the clock again goes low, Q retains or stores the last value of D. a D flip flop is a bistable circuit whose D input is transferred to the output after a clock pulse is received.

- **T FLIP-FLOP**: The T or "toggle" flip-flop changes its output on each clock edge, giving an output which is half the frequency of the signal to the T input. It is useful for constructing binary counters, frequency dividers, and general binary addition devices. It can be made from a J-K flip-flop by tying both of its inputs high.



TRUTH TABL E:
SR F LIP FLOP:

CLOCK	S	R	n +1
1	0	0	NO CH ANGE
1	0	1	0
1	1	0	1
1	1	1	?

D FL IPFLOP:

INPUT	OU TPUT
0	0
1	1

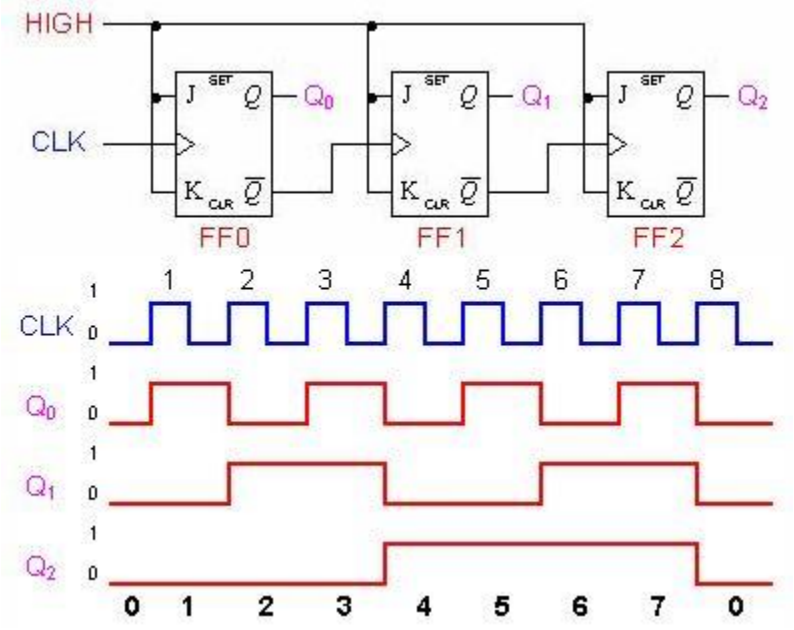
JK FLIPFLOP

CLOCK	S	R	n +1
1	0	0	NO CH ANGE
1	0	1	0
1	1	0	1
1	1	1	Qn '

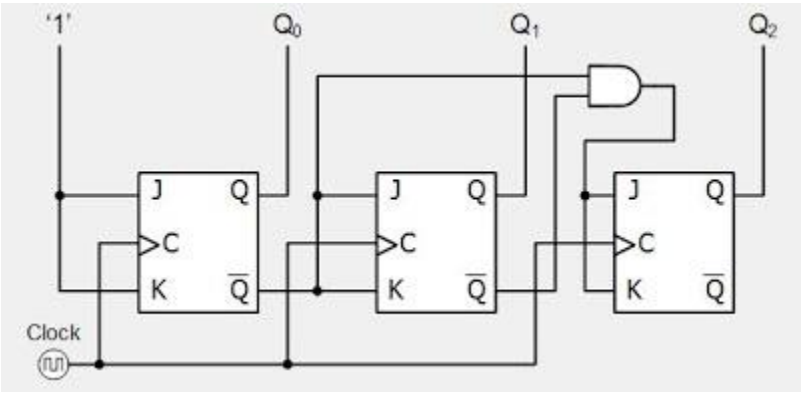
T FL IPFLOP

CLOCK	S	R	n +1
1	0	1	NO CH ANGE
1	1	0	Qn '

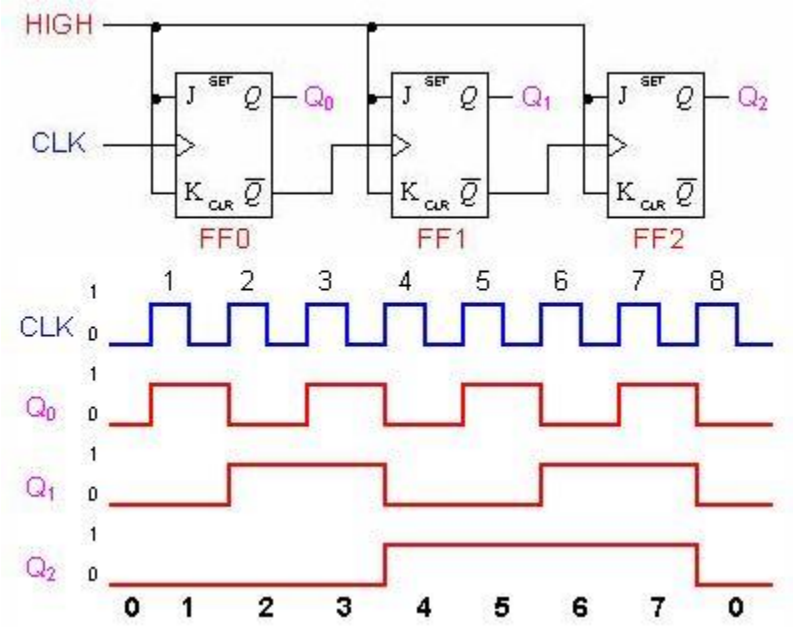
3 bit Asynchronous up counter



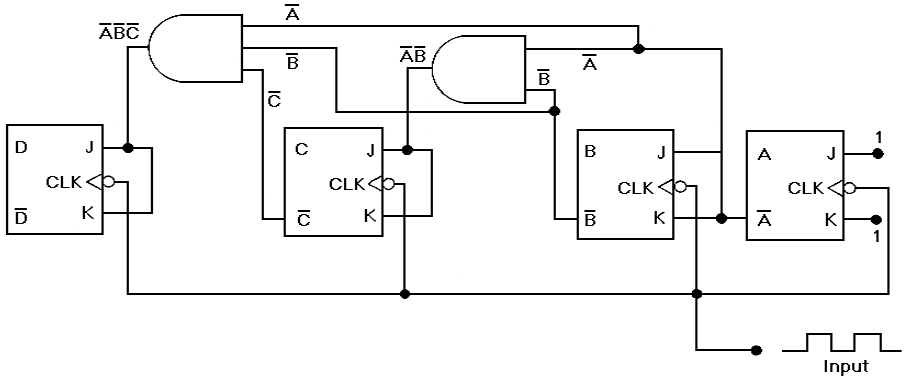
3 bit synchronous up counter



3 bit Asynchronous down counter



3 bit Synchronous down counter



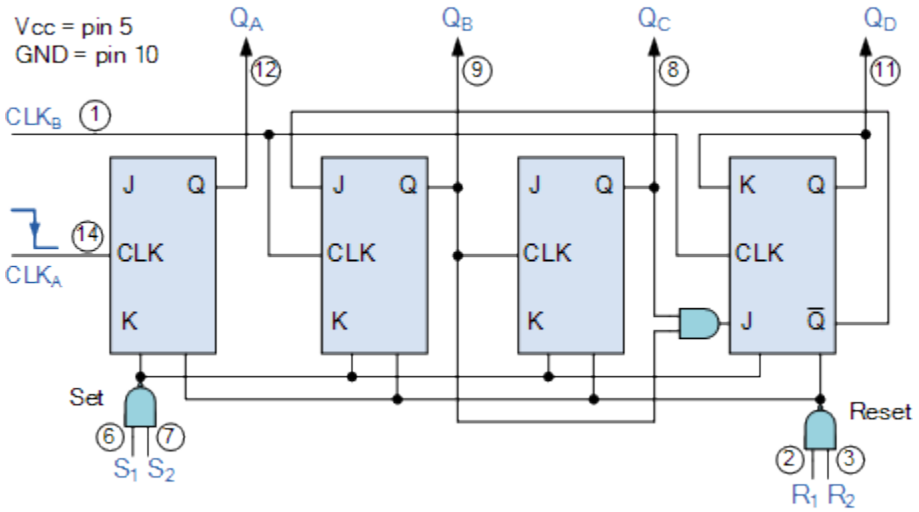
Conclusion- Verified the truth table for JK Flip Flop using IC 7476 and implemented the given counters using JK flip flop.

EXPERIMENT NO: 8

Aim: Design and implement modulo 'n' counter with IC 7490.

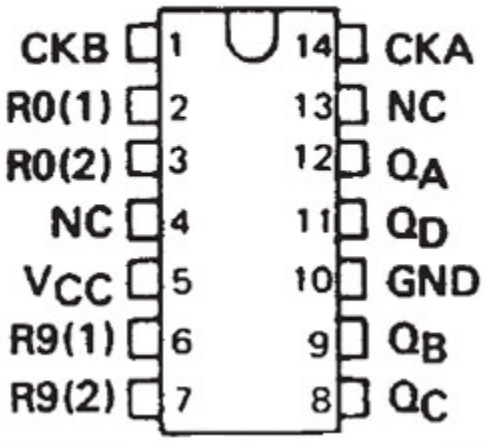
APPARATUS REQUIRED: Digital Trainer Kit. Connecting Leads, IC 7490.

THEORY: The 74LS90 integrated circuit is basically a MOD-10 decade counter that produces a BCD output code. The 74LS90 consists of four master-slave JK flip-flops internally connected to provide a MOD-2 (count-to-2) counter and a MOD-5 (count-to-5) counter. The 74LS90 has one independent toggle JK flip-flop driven by the CLK A input and three toggle JK flip-flops that form an asynchronous counter driven by the CLK B input as shown.



The counters four outputs are designated by the letter symbol Q with a numeric subscript equal to the binary weight of the corresponding bit in the BCD counter circuits code. So for example, Q_A, Q_B, Q_C and Q_D. The 74LS90 counting sequence is triggered on the negative going edge of the clock signal, that is when the clock signal CLK goes from logic 1 (HIGH) to logic 0 (LOW).

The additional input pins R_1 and R_2 are counter "reset" pins while inputs S_1 and S_2 are "set" pins. When connected to logic 1, the Reset inputs R_1 and R_2 reset the counter back to zero, 0 (0000), and when the Set inputs S_1 and S_2 are connected to logic 1, they Set the counter to maximum, or 9 (1001) regardless of the actual count number or position.The 74LS90 counter consists of a divide-by-2 counter and a divide-by-5 counter within the same package. Then we can use either counter to produce a divide-by-2 frequency counter only, a divide-by-5 frequency counter only or the two together to produce our desired divide-by-10 BCD counter.



Truth Table

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

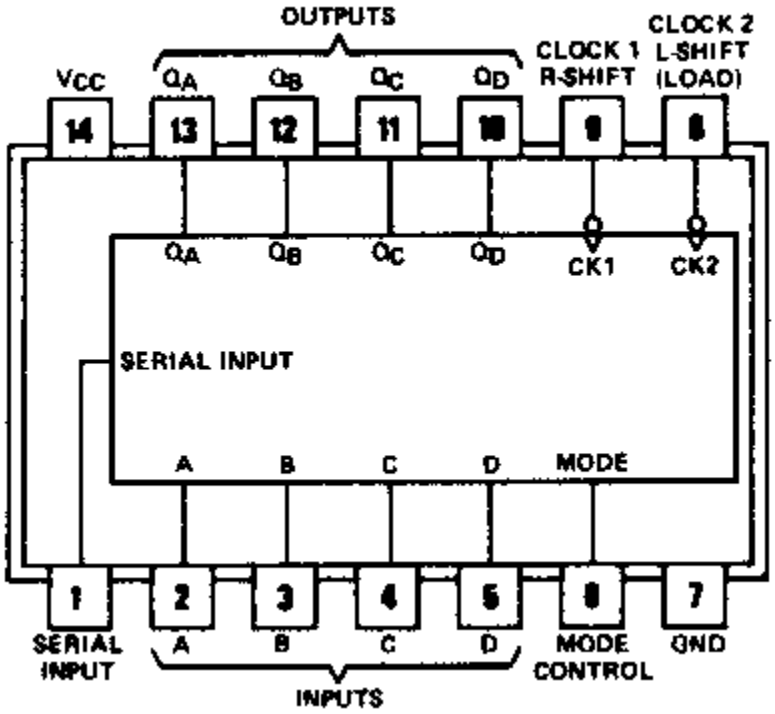
EXPERIMENT NO: 9

Aim: Design and implementation of Shift Register using IC 7495.

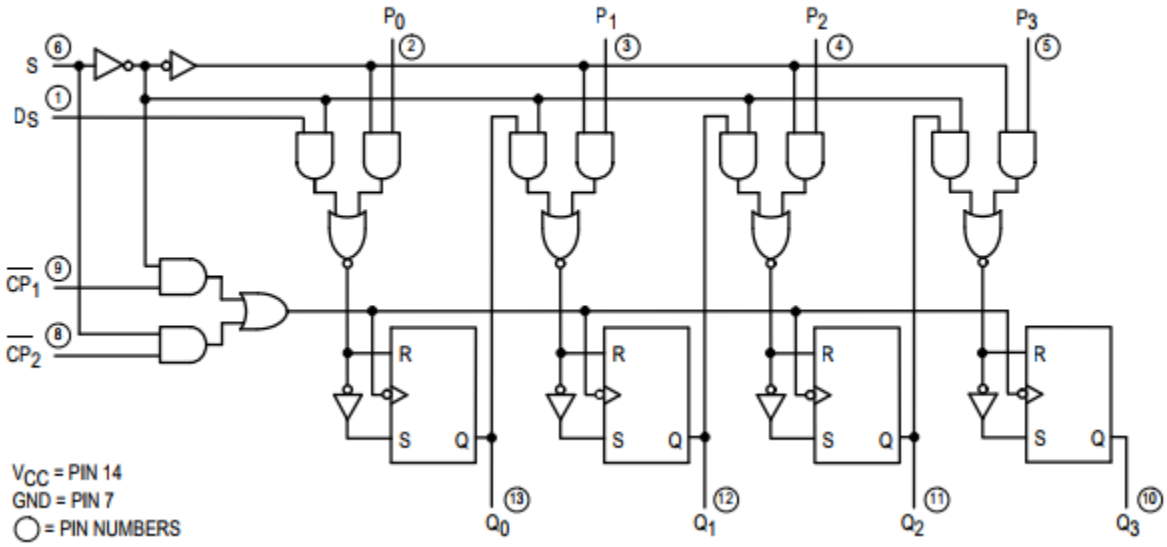
APPARATUS REQUIRED: Digital Trainer Kit. Connecting Leads, IC 7495.

THEORY: 4-BIT SHIFT REGISTER The SN54/74LS95B is a 4-Bit Shift Register with serial and parallel synchronous operating modes. The serial shift right and parallel load are activated by separate clock inputs which are selected by a mode control input. The data is transferred from the serial or parallel D inputs to the Q outputs synchronous with the HIGH to LOW transition of the appropriate clock input.

The LS95B is a 4-Bit Shift Register with serial and parallel synchronous operating modes. It has a Serial (DS) and four Parallel (P0–P3) Data inputs and four Parallel Data outputs (Q0–Q3). The serial or parallel mode of operation is controlled by a Mode Control input (S) and two Clock Inputs (CP1) and (CP2). The serial (right-shift) or parallel data transfers occur synchronous with the HIGH to LOW transition of the selected clock input. When the Mode Control input (S) is HIGH, CP2 is enabled. A HIGH to LOW transition on enabled CP2 transfers parallel data from the P0–P3 inputs to the Q0–Q3 outputs. When the Mode Control input (S) is LOW, CP1 is enabled. A HIGH to LOW transition on enabled CP1 transfers the data from Serial input (DS) to Q0 and shifts the data in Q0 to Q1, Q1 to Q2, and Q2 to Q3 respectively (right-shift). A left-shift is accomplished by externally connecting Q3 to P2, Q2 to P1, and Q1 to P0, and operating the LS95B in the parallel mode (S = HIGH). For normal operation, S should only change states when both Clock inputs are LOW. However, changing S from LOW to HIGH while CP2 is HIGH, or changing S from HIGH to LOW while CP1 is HIGH and CP2 is LOW will not cause any changes on the register outputs.



LOGIC DIAGRAM



MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS					OUTPUTS			
	S	CP_1	CP_2	D_S	P_n	Q_0	Q_1	Q_2	Q_3
Shift	L	↓	X	l	X	L	q_0	q_1	q_2
	L	↓	X	h	X	H	q_0	q_1	q_2
Parallel Load	H	X	↓	X	P_n	P_0	P_1	P_2	P_3
Mode Change	↓	L	L	X	X	No Change			
	↑	L	L	X	X	No Change			
	↓	H	L	X	X	No Change			
	↑	H	L	X	X	Undetermined			
	↓	L	H	X	X	Undetermined			
	↑	L	H	X	X	No Change			
	↓	H	H	X	X	Undetermined			
	↑	H	H	X	X	No Change			

EXPERIMENT NO: 10

Aim: VHDL Programming: - Simulation of Full adder using behavioral & structural modelling.

Software requirement:- Xilinx or ghdl simulator.

Theory: Hardware Descriptive Languages (HDL) are the basic language that are used for the designing of most of the digital circuits using software tools. VHDL is one of most important type of HDL. VHDL is the abbreviation for "Very High Speed Integrated Circuit Hardware Description Language (VHSIC HDL)".This language is combination of Concurrent, sequential, wave simulation, timing and net list simulation language. . VHDL supports both structural and behavioural descriptions of a system at multiple levels of abstraction
Components of VHDL are:-

1. Library-
2. Entity- Each design has only one entity block which describes the interface signals into and out of the design unit.
3. Architecture- The architecture block defines how the entity operates.

VHDL is supposed to be the heart for the production of electronic design of any digital circuit. With the advancement of time and the shrinking of the semiconductor device dimensions into compact sizes, the VHDL has gained a lot of importance. There are three architectural styles of modeling of a VHDL statement. The internal functionality of digital circuit can be specified using any of the modeling styles as discussed below

- Structural: - A structural architectural design refers to the architectural design where all the used components are interconnected to each other.
- Dataflow: - A dataflow architectural design refers to the architectural design where a set of concurrent assignment statements are used to design the program.
- Behavioral:-A behavioral architectural design refers to the architectural design where a set of sequential assignment statements are used to design the program.

Implementation

Entity statement:

```
entity Full_Adder is
 Port ( Input1 : in STD_LOGIC;
 Input2 : in STD_LOGIC;
 Input3 : in STD_LOGIC;
 Sum : out STD_LOGIC;
 Carry : out STD_LOGIC);
end Full_Adder;
```

Architectural design using:

1. **Structural**

```
architecture Structural of Full_Adder is
signal xor_1, and_1, and_2:STD_LOGIC;
 component XOR1 port(A, B : in STD_LOGIC;
 X: out STD_LOGIC);
 end component;
 component OR1 port(M, N : in STD_LOGIC;
 Y: out STD_LOGIC);
 end component;
 component AND1 port(P, Q : in STD_LOGIC;
 Z: out STD_LOGIC);
 end component;
 begin
 X1:XOR1 port map(Input1, Input2, xor_1);
 X2:XOR1 port map(xor_1, Input3, Sum);
 A1:AND1 port map(Input1, Input2, and_1);
 A2:AND1 port map(xor_1, Input3, and_2);
 O1:OR1 port map(and_1, and_2, Carry);
end Structural;
```

2. Dataflow

```
architecture DataFlow of Full_Adder is
signal xor_1, and_1, and_2:STD_LOGIC;
 begin
xor_1 <= Input1 xor Input2;
and_1 <= Input1 and Input2;
and_2 <= xor_1 and Input3;
Sum <= xor_1 xor Input3;
Carry <= and_1 or and_2;
end DataFlow;
```

3. Behavioral

```
architecture Behavioral of Full_Adder is
begin
process(Input1, Input2, Input3)
begin
if (Input1='0') and (Input2='0') and (Input3='0')
then Sum<='0'; Carry<='0';
elsif (Input1='0') and (Input2='0') and (Input3='1')
then Sum<='1'; Carry<='0';
elsif (Input1='0') and (Input2='1') and (Input3='0')
then Sum<='1'; Carry<='0';
elsif (Input1='0') and (Input2='1') and (Input3='1')
then Sum<='0'; Carry<='1';
elsif (Input1='1') and (Input2='0') and (Input3='0')
then Sum<='1'; Carry<='0';
elsif (Input1='1') and (Input2='0') and (Input3='1')
then Sum<='0'; Carry<='1';
```

```
elsif (Input1='1') and (Input2='1') and (Input3='0')
then Sum<='0'; Carry<='1';
elsif (Input1='1') and (Input2='1') and (Input3='1')
then Sum<='1'; Carry<='1';
end if;
end process;
end Behavioral;
```

Conclusion: - Simulation of Full adder using behavioral & structural modelling is performed.