INDEX

SR. NO.	EXPERIMENT NAME
1.	To study and verify the truth table of various logic gates using ICs and realize Boolean expressions using gates.
2.	To realize basic gates using universal gates.
3.	To realize binary to gray code and gray code to binary converter.
4.	To realize parity generator and detector.
5.	To realize arithmetic circuits: i) Half adder ii) Full adder iii) Half subtractor iv) Full subtractor
6.	To realize 2 bit magnitude comparator.
7.	To Study multiplexer IC and realization of full adder using multiplexer IC.
8.	To Study decoder IC and realization of combinational logic using decoder IC.
9.	Study of flip-flops using IC's of J-K Flip Flop.
10.	To realize basic gates using VHDL.
11.	To realize 4:1 multiplexer using VHDL.

EXPERIMENT NO.1

AIM: To study and verify the truth table of various logic gates using ICs and realize Boolean expressions using gates.

Objectives:

1. To implement basic gates (AND,OR,NOT)and verify their truth table.
2. To implement universal gates(NAND,NOR) and verify their truth table.
3. To implement derived gates(XOR,XNOR) and verify their truth table

CO's to be achieved: 1 (Binary and Hexadecimal calculations and conversions)

PO's to be achieved:

1. An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2. An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3. An ability to design, implement, and evaluate a computer-based system. component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4. An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

APPARATUS REQUIRED:

SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE 2 I/P	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
7.	NAND GATE 3 I/P	IC 7410	1
8.	IC TRAINER KIT	-	1

THEORY:

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

OR, AND, and NOT are basic gates. NAND, NOR and X-OR are known as Universal gates. Basic gates form these gates.

AND GATE:

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

OR GATE:

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

NOT GATE:

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

NAND GATE:

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

NOR GATE:

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

X-OR GATE:

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

PROCEDURE:

(i) Connections are given as per circuit diagram.

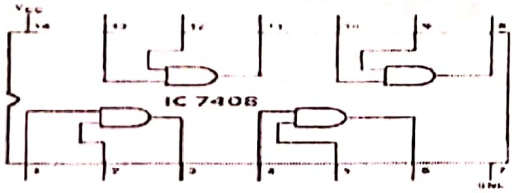
(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table.

AND GATE:

SYMBOL:

PIN DIAGRAM



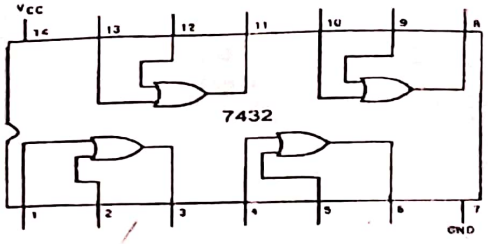
TRUTH TABLE:

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:

SYMBOL:

PIN DIAGRAM

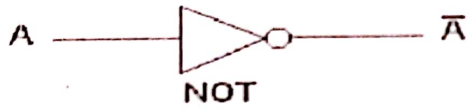


TRUTH TABLE:

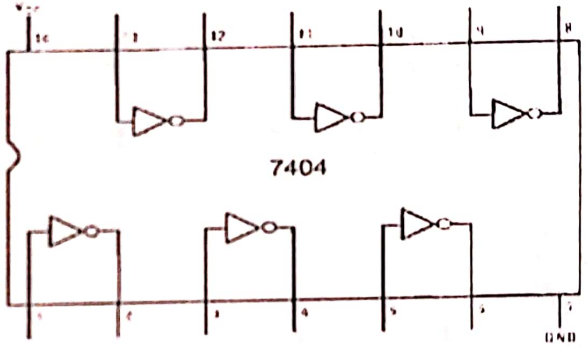
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE:

SYMBOL:



PIN DIAGRAM



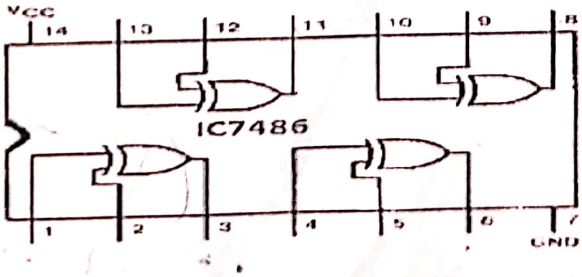
TRUTH TABLE

A	$\overline{A}$
0	1
1	0

X-OR GATE:

SYMBOL:

PIN DIAGRAM



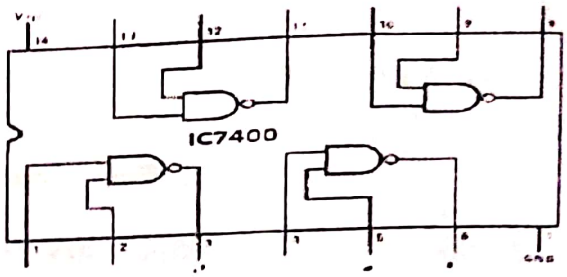
TRUTH TABLE

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

NAND GATE:

SYMBOL:

PIN DIAGRAM

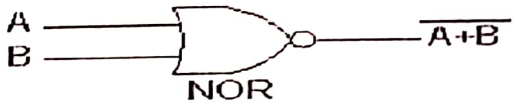


TRUTH TABLE

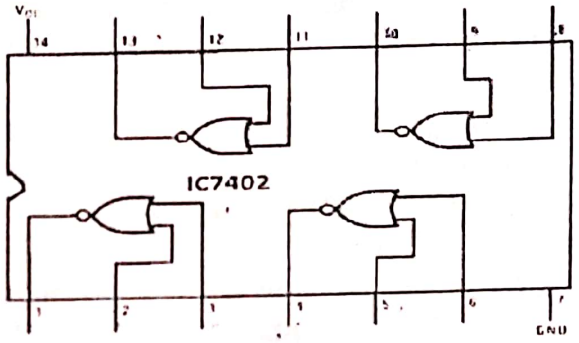
A	B	$\overline{\quad}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

SYMBOL:



PIN DIAGRAM



TRUTH TABLE

A	B	$\overline{\quad}$
0	0	1
0	1	0
1	0	0
1	1	0

CONCLUSION: Thus, we studied basic gates, universal and derived gates.

SAMPLE QUESTIONS:

1) What is universal gate?
2) Give truth table of AND. OR. NOT. NOR. NAND Gate.
3) Draw pin diagrams of AND. OR. NOT. NOR. NAND Gate.
4) Explain why NOR and NAND gate are called as universal gates.

EXPERIMENT NO. 2

AIM: To realize basic gates using universal gates.

Objectives:

(i) To implement basic gates (AND, OR, NOT) using NAND,NOR gate and verify their truth table.
(ii) To implement universal gates (NAND, NOR) using NAND, NOR gate and verify their truth table.
(iii) To implement derived gates(XOR,XNOR) using NAND,NOR gate and verify their truth table
(iv) To understand NAND –NOR realization.

CO's to be achieved : 1 (Binary and Hexadecimal calculations and conversions)

PO's to be achieved :

1. An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2. An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3. An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4. An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

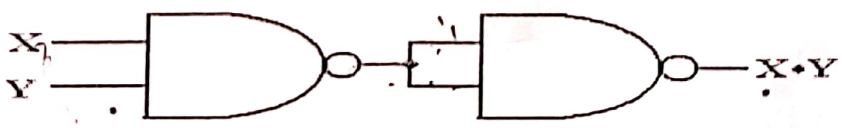
APPARATUS: Power supply, Breadboard, Trainer Kit.

COMPONENTS: ICs 7400, 7402.

CIRCUIT DIAGRAM:

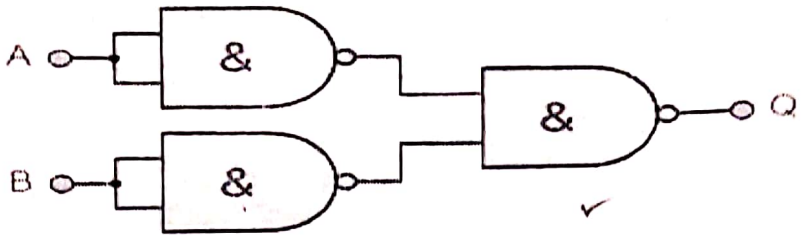
(a) **Using NAND:**

1. AND Y=A.B



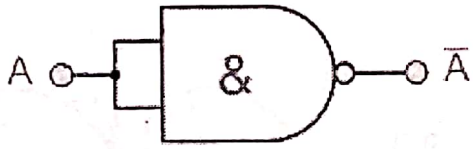
2. OR Y=A+B

2. OR Y=A+B



2-Input OR Gate

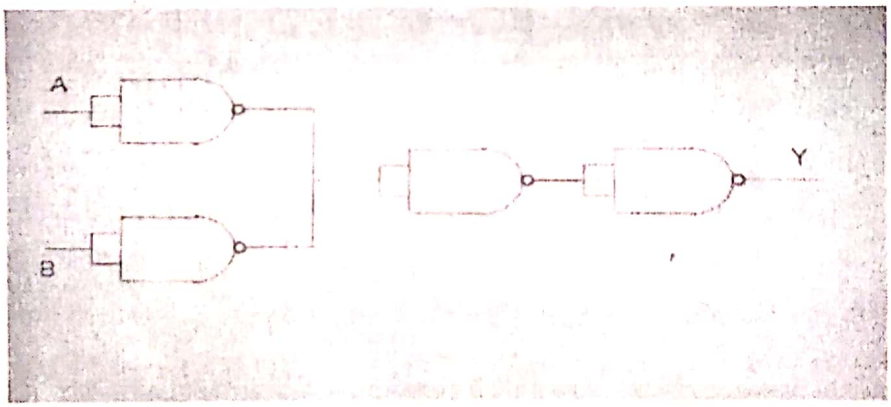
3. NOT Y=$\bar{A}$



NOT Gate

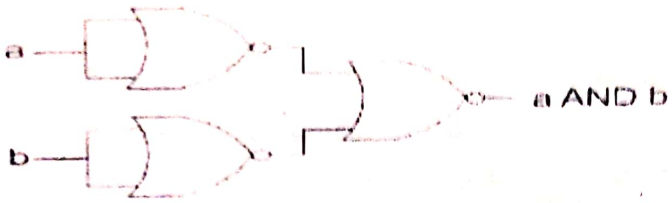
4. O Y=$\bar{A}$

5. NO Y=

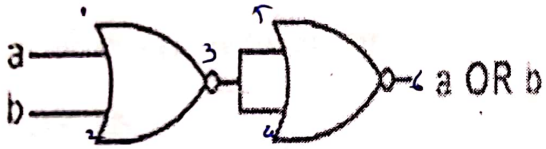


(b) Using NOR:

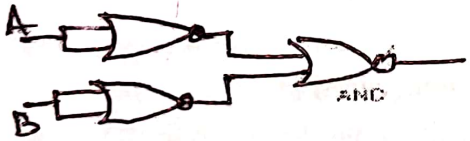
1. AND Y=A.B

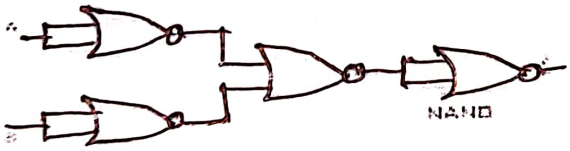


2. OR Y=A+B



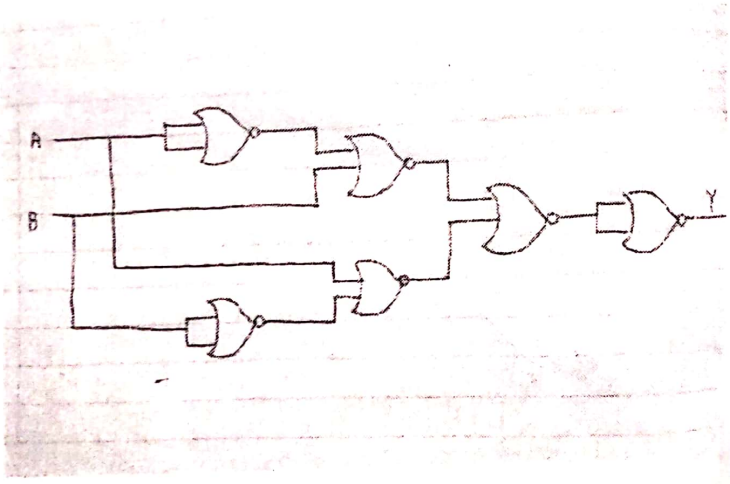
3 NOT Y=Ā





N N Y=

O Y=Ā



DIGITAL LOGIC DESIGN AND APPLICATIONS

THEORY: AND, OR, NOT are called basic gates as their logical operations cannot be simplified further.

NAND and NOR are called universal gates as using only NAND or only NOR can logic function be implemented. Using NAND and NOR gates and **De Morgan's Theorems** different basic gates & EX-OR gates are realized.

PROCEDURE:

1) Give biasing to the IC and do necessary connections as shown in the circuit diagrams.

2) Give various combinations of inputs and note down output using **LED.**

3) Repeat the procedure for all gates.

CONCLUSION:

Hence, We have verified the output generated by LED for various logical gates implemented by the different combinations of Universal gates NAND & NOR.

SAMPLE QUESTIONS:

1) State and prove Morgan's theorem
2) Using NAND gate implement basic gates.
3) Using NOR gate implement basic gates.
4) Implement XOR and XNOR gates using universal gates.

EXPERIMENT NO.3

AIM: To realize binary to gray code and gray code to binary converter.

OBJECTIVE: To learn the importance of non-weighted code. To learn to generate gray code.

CO's to be achieved : 2 (Designing for conversion)

PO's to be achieved :

1. An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.

2. An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.

3. An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.

4. An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

APPARATUS: Power supply , Breadboard.

COMPONENTS: IC 7400, IC 7486, and IC 7408, Patch Cords & IC Trainer Kit

THEORY:

The logical circuit which converts binary code to equivalent gray code is known as **binary to gray code converter**. The gray code is a non weighted code. The successive gray code differs in one bit position only that means it is a unit distance code. It is also referred as cyclic code. It is not suitable for arithmetic operations. It is the most popular of the unit distance codes. It is also a reflective code. An n-bit Gray code can be obtained by reflecting an n-1 bit code about an axis after 2n-1 rows, and putting the MSB of 0 above the axis and the MSB of 1 below the axis.

The 4 bits binary to gray code conversion table is given below,

1) BINARY TO GRAY CONVERSION

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

G3 = B3

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

G2= B3 ⊕ B2

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

G1=B1 ⊕ B2

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

G0=B1 ⊕ B0

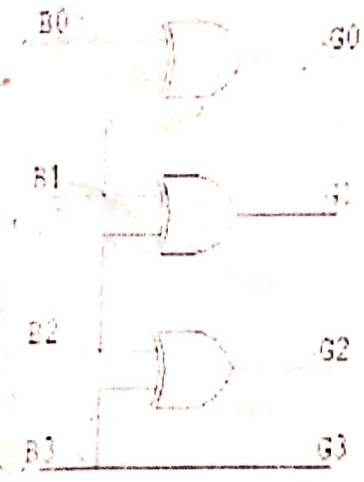
Binary				Gray			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

BOOLEAN EXPRESSIONS:

G3=B3

G2=B3 ⊕ B2

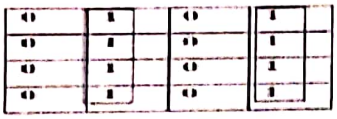
G1=B1 ⊕ B2: G0=B1 ⊕ B0



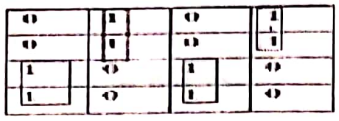
BINARY TO GRAY CODE USING EX-OR GATES

ID GRAY TO BINARY CONVERSION

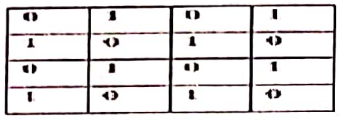
B3 = G3



B2=G3⊕G2



B1=G3⊕G2⊕G1



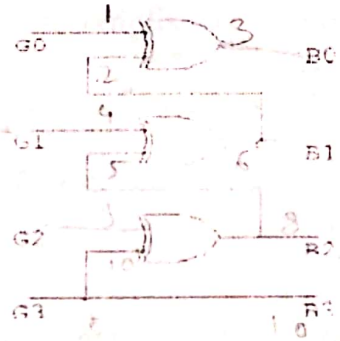
B0=G3 ⊕ G2⊕ G1⊕ G0

BOOLEAN EXPRESSIONS:

B3=G3
B2=G3 ⊕ G2
B1=G3 ⊕ G2⊕ G1
B0=G3 ⊕ G2⊕ G1⊕ G0

Gray				Binary			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

GRAY TO BINARY CODE CONVERSION USING EX-OR GATES



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. Verify the Truth Table and observe the outputs.

RESULT:

Binary to gray code conversion and vice versa is realized using EX-OR gates and NAND gates.

VIVA QUESTIONS:

1. What are code converters?
2. What is the necessity of code conversions?

EXPERIMENT NO.4

AIM: To design & implement 4 – bit parity generator/checker.

Objectives:

To design of 4- bit parity generator/checker using (a) Minimum number of gates (b) IC 74180.

CO's to be achieved : 2 (Designing of combinational circuits)

PO's to be achieved :

1. An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2. An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3. An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4. An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

Resources Required: IC 7486, IC 74180, IC 7404, Connecting Wires, Breadboard.

Equipments / Machines: Trainer Kit

Theory:

A parity generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called parity checker. A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word.

Procedure:

1 Place the required IC on the bread board / Trainer kit.

2 Connect ckt as shown in figure.

3 Turn on the power supply.

4 Verify the truth table.

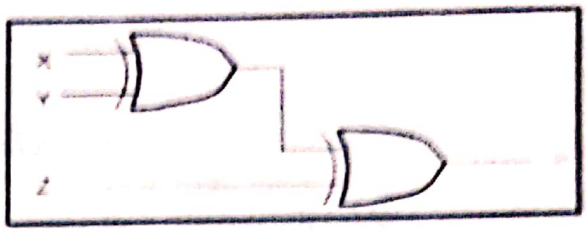
Diagram:

(a) 3 -bit Parity Generator

Truth Table:

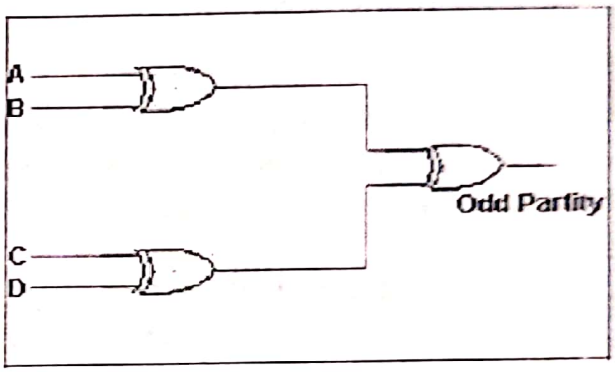
3-bit Message			Odd Parity Bit	Even Parity Bit
X	Y	Z		
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Diagram:



(b) 4 –bit Parity Checker

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



Analysis / Conclusion: Hence, we have studied the design 4 – bit parity generator/checker.

Viva Questions:

1 According to the circuit you have done, find the truth table for the odd-parity checker.

2 Considering 4-bit message to be transmitted, derive the parity circuit using odd parity bit.

EXPERIMENT NO.5

AIM: To realize arithmetic circuits.

1) Half adder 2) Full adder 3) Half subtractor 4) Full subtractor

Objectives:

4. To implement half adder ckt. And verify their truth table.
5. To implement full adder ckt. And verify their truth table.
6. To implement half subtractor ckt. And verify their truth table
7. To implement full subtractor ckt. And verify their truth table

CO's to be achieved : 2 (Designing of combinational circuits)

PO's to be achieved :

1. An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2. An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3. An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4. An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

APPARATUS: Power supply , Breadboard.

COMPONENTS: ICs 7486,7408, 7432

THEORY:

1. **Half adder** :A basic module used in binary arithmetic elements is the half-adder. The function of the half-adder is to add two binary digits, producing a sum according to the binary addition rules.

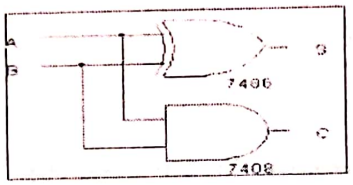
2. **Full adder**: The adder circuit is capable of adding the content of two registers. It must include provision for handling carries as well as an addend and augends bits. So there must be three inputs to each stage of a multi digit adder, except the stage for the least significant bits. One for each input from the numbers being added, one for any carry that might have been generated or propagated by the previous stage.

3. **Half subtractor**: A half subtractor subtracts a bit from another.. The half subtractor has two input bits A and B two output bits, a difference DIFF = (A-B) and a Borrow.
4. **Full subtractor** : A full subtractor subtracts with three bits (A-B-C). The third bit C is the borrow from previous stage.

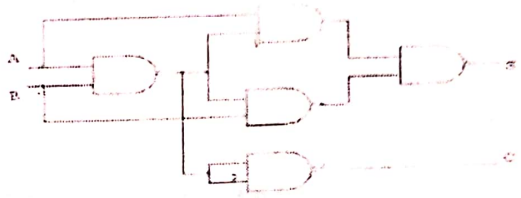
CIRCUIT DIAGRAM:

Half Adder:

i) Basic Gates



ii) NAND Gates



Truth table:

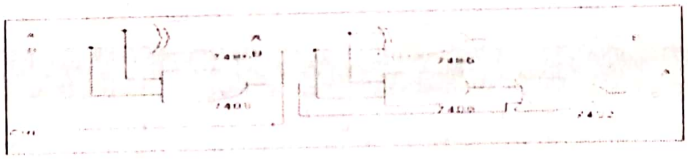
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Boolean Expressions:

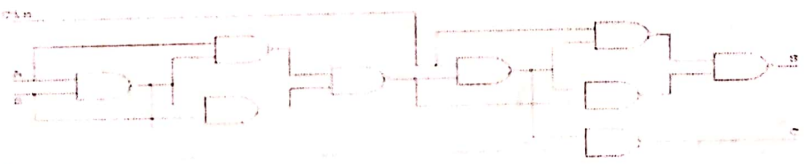
Carry=A B

Full Adder:

Basic gates:



ii) NAND GATES



Truth Table:

II. FULL ADDER

TRUTH TABLE

INPUTS			OUTPUTS	
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

BOOLEAN EXPRESSIONS:

$S = A \oplus B \oplus C$

C=A B + B Cin + A Cin

Half Subtractor:

TRUTH TABLE:

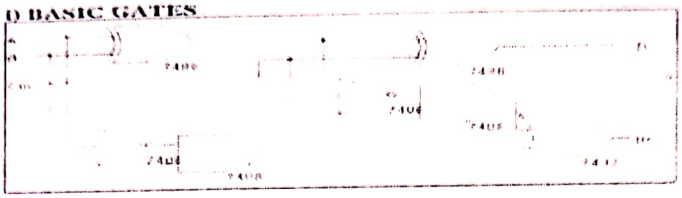
INPUTS			OUTPUTS	
A	B	Cin	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSIONS:

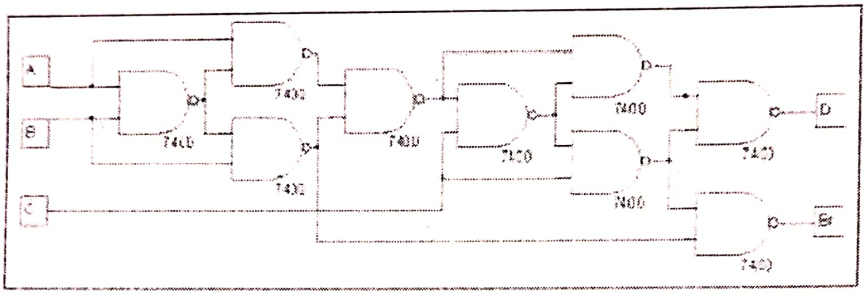
$D = A \oplus B \oplus C$

$Br = \bar{A} B + B Cin + \bar{A} Cin$

i) BASIC GATES



ii) To Realize the Full subtractor using NAND Gates only



Full Subtractor:

D = X- Y- Z.

B = 1 if (X < y+ z)

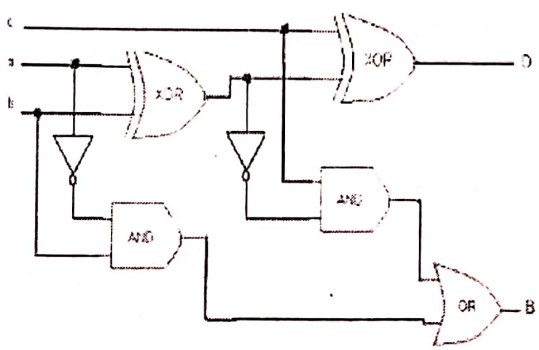
Truth table:

Diagram:

input			output	
X	Y	Z	D	B
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

D = X xor Y xor Z.

B = Z.(X xnor Y) + X.Y



PROCEDURE:

(i) Connections are given as per circuit diagram.

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table

CONCLUSION: Thus we have implement half adder, full adder and half subtractor, full subtractor circuits and verify their truth table.

VIVA QUESTIONS:

1) What is a half adder?
2) What is a full adder?
3) What are the applications of adders?
4) What is a half subtractor?
5) What is a full subtractor?
6) What are the applications of subtractors?
7) Obtain the minimal expression for above circuits.
8) Realize a full adder using two half adders
9) Realize a full subtractors using two half subtractors

EXPERIMENT NO. 6

AIM: To realize One & Two Bit Comparator and study of 7485 magnitude comparator.

OBJECTIVE: To learn about various applications of comparator To learn and understand the working of IC 7485 magnitude comparator To learn to realize 8-bit comparator using 4-bit comparator.

CO's to be achieved

Design synchronous and asynchronous sequential circuits.

PO's to be achieved

1 An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2 An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3 An ability to design, implement, and evaluate a computer-based system, component. or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4 An ability to understand real life challenges and use current techniques, skills, and modern tools necessary for computing practice.
5 An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

THEORY: Magnitude Comparator is a logical circuit, which compares two signals A and B and generates three logical outputs, whether A > B, A = B, or A < B. IC 7485 is a high speed 4-bit Magnitude comparator, which compares two 4-bit words . The A = B Input must be held high for proper compare operation.

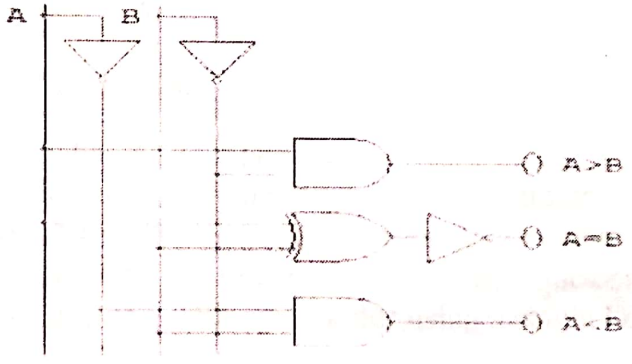
COMPONENTS REQUIRED: IC 7400, IC 7410, IC 7420, IC 7432, IC 7486, IC 7402. IC 7408, IC 7404, IC 7485, Patch Cords & IC Trainer Kit.

1) 1- BIT COMPARATOR

$A>B = A\bar{B}$

$A<B = \bar{A}B$

$A=B = \bar{A}\bar{B}+AB$



TRUTH TABLE

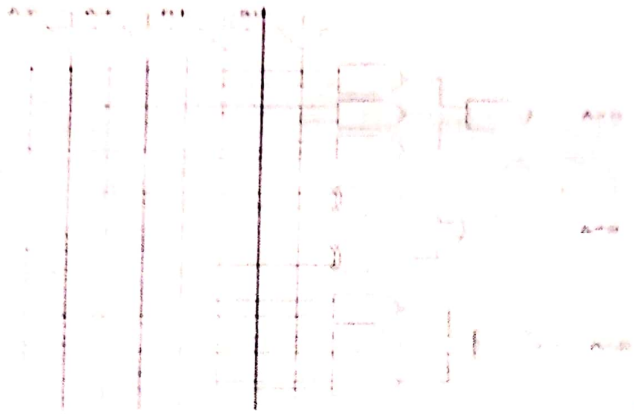
INPUTS		OUTPUTS		
A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

2) 2- BIT COMPARATOR

$(A>B) = A1\,\bar{B}1 + A0\,\bar{B}1\,\bar{B}0 + \bar{B}0A1A0$

$(A=B) = (A0 \odot B0)(A1 \odot B1)$

$(A<B) = \bar{B}1\,\bar{A}1 + \bar{B}0\,\bar{A}1\,\bar{A}0 + \bar{A}0\bar{B}1\bar{B}0$



2-bit comparator circuit diagram

TRUTH TABLE

INPUTS				OUTPUTS		
A_1	A_0	B_1	B_0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

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 Verify the Truth Table and observe the outputs.

RESULT: One bit, two bit and four bit comparators are verified using basic gates and magnitude comparator IC7485

VIVA QUESTIONS:

1) What is a comparator?
2) What are the applications of comparator?
3) Derive the Boolean expressions of one bit comparator and two bit comparators.
4) How do you realize a higher magnitude comparator using lower bit comparator
5) Design a 2 bit comparator using a single Logic gates?

EXPERIMENT NO. 7

AIM: To design and set up the following circuit

1) To design and set up a 8:1 Multiplexer
2) To set up a Half/Full Adder and Half/Full Subtractor using IC 74153.

OBJECTIVE: To learn about various applications of multiplexer. To learn and understand the working of IC 74151. To learn to realize any function using Multiplexer

THEORY: Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has 2n input signals, n control/select signals and 1 output signal.

CO's to be achieved

1. Design the combinational circuits.

PO's to be achieved

1) An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2) An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3) An ability to design, implement, and evaluate a computer-based system. component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4) An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

Apparatus:

- IC type 74151 8:1 multiplexer
- IC trainer kit

IC Description:

74151 is a 8 line-to-1 line multiplexer. It has the schematic representation shown in Fig 1. Selection lines S2 , S1 and S select the particular input to be multiplexed and applied to the output. 0

Strobe S acts as an enable signal. If strobe =1, the chip 74151 is disabled and output y = 0. If strobe = 0 then the chip 74151 is enabled and functions as a multiplexer. Table 1 shows the multiplex function of 74151 in terms of select lines.

Table 1.

Strobe	Select Lines			Output
S	S_2	S_1	S_0	Y
1	X	X	X	0
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7

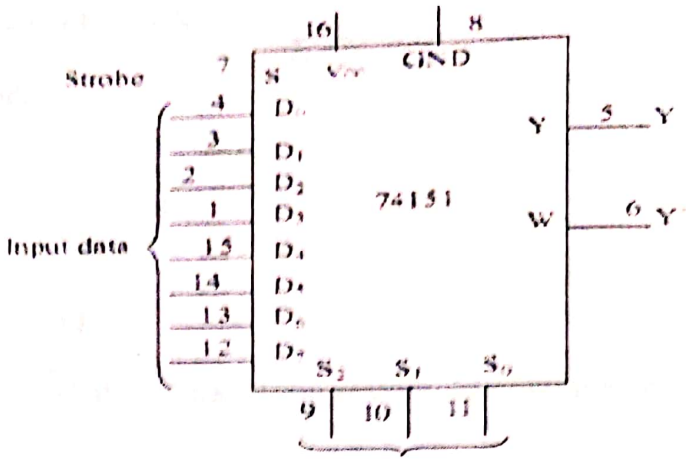
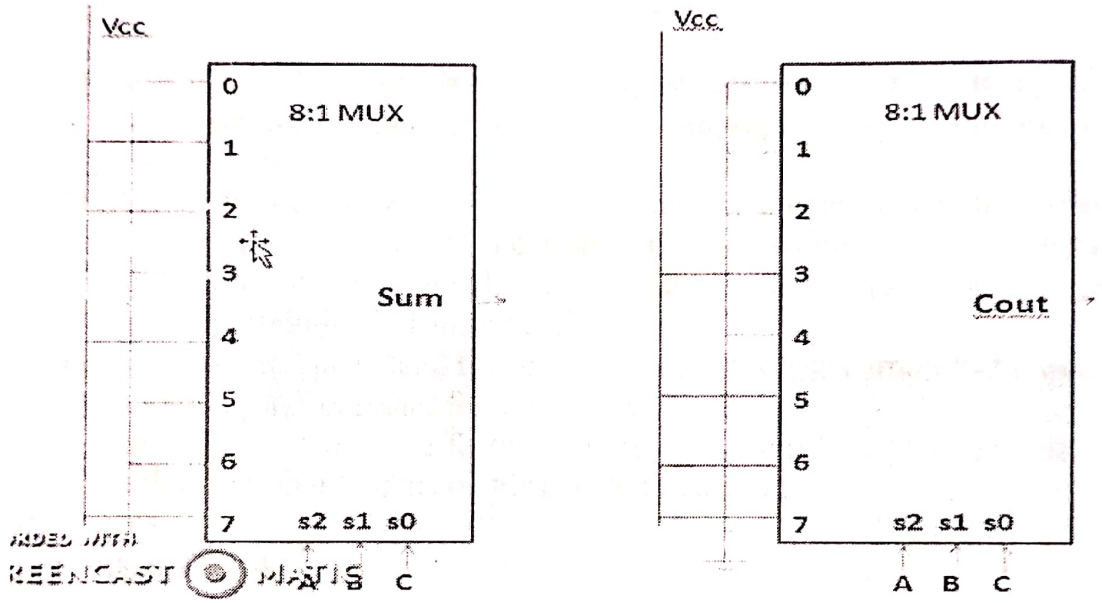


Fig.1 IC type 74151 Multiplexer 8×1

Design Full Adder using 8:1 MUX:

Implementation of Full Adder



Procedure:

1) Connections are as per given circuit diagram
2) Logical inputs are as per given circuit diagram
3) Observe the output and verify the truth table

Conclusion:

Studied use of multiplexers for implementation of combinational circuits.

Viva Questions:

1) What is Multiplexer?
2) Design Full adder using 8:1 Multiplexers.
3) What is multiplexer tree? Design 16:1 multiplexer using 8:1 Multiplexers.

EXPERIMENT NO. 8

AIM : To design Seven Segment Decoder

Objectives:

1) To learn about various applications of decoder
2) To learn about implementation of seven-segment display

Outcomes:

At the end of the expt. the students will be able

1. To design a combinational circuit and able to transform real world programs into digital logic

CO's to be achieved

Design synchronous and asynchronous sequential circuits.

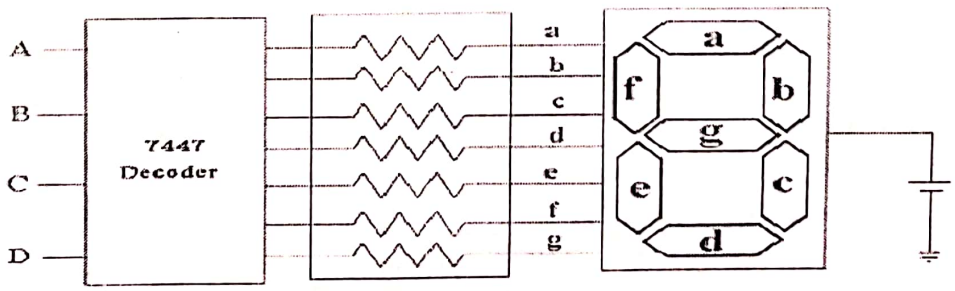
PO's to be achieved

1) An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2) An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3) An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4) An ability to understand real life challenges and use current techniques, skills, and modern tools necessary for computing practice.
5) An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

Apparatus:

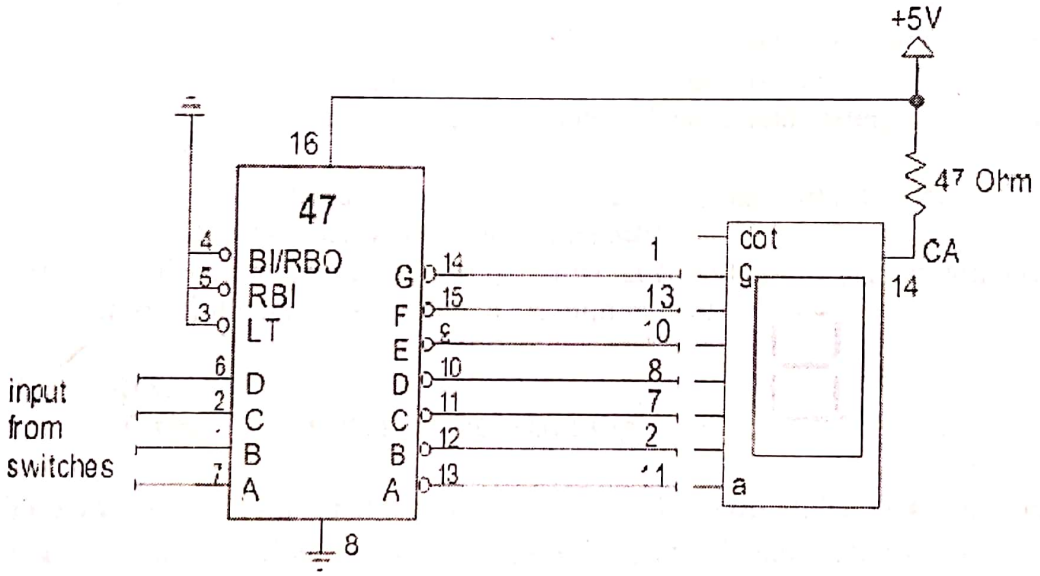
- IC type 7447
- IC trainer kit

Description: The display has 7 inputs each connected to an LED segment. All anodes of LEDs are tied together and joined to 5 volts (this type is called common anode type). A limiting resistance network must be used at the inputs to protect the 7segment from overloading. BCD inputs are converted into 7 segment inputs (a, b, c, d, e, f, g) by using a decoder, as shown in Fig.4.



A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n output lines. The input to the decoder is a BCD code and the outputs of the systems are the seven segments a, b, c, d, e, f, and g. For further information and pin connections, consult the specification sheet for decoder and 7-segment units.

Binary Inputs	Decoder Outputs							7 Segment Display Outputs
D C B A	a	b	c	d	e	f	g	
0 0 0 0	1	1	1	1	1	1	0	0
0 0 0 1	0	1	1	0	0	0	0	1
0 0 1 0	1	1	0	1	1	0	1	2
0 0 1 1	1	1	1	1	0	0	1	3
0 1 0 0	0	1	1	0	0	1	1	4
0 1 0 1	1	0	1	1	0	1	1	5
0 1 1 0	1	0	1	1	1	1	1	6
0 1 1 1	1	1	1	0	0	0	0	7
1 0 0 0	1	1	1	1	1	1	1	8
1 0 0 1	1	1	1	1	0	1	1	9



Procedure:

1) Connections are as per given circuit diagram
2) Logical inputs are as per given circuit diagram
3) Observe the output and verify the truth table

Conclusion:

Studies 7447 BCD Decoder. Implemented seven segment display interface with 7447.

Viva Questions:

1) What is BCD decoder?
2) List types of seven segment displays.
3) Design seven segment display using common anode and common cathode seven segment display.

EXPERIMENT NO. 9

AIM : Study of flip-flop using IC's

Objectives:

1. To become familiar with flip-flops.
2. To implement and observe the operation of different flip-flops

Outcomes:

At the end of the expt. the students will be able

1. To design sequential circuits.
2. To implement and analyze the behavior of JK FF.

CO's to be achieved

Design synchronous and asynchronous sequential circuits.

PO's to be achieved

6 An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.

7 An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.

8 An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.

9 An ability to understand real life challenges and use current techniques, skills, and modern tools necessary for computing practice.

10 An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

Apparatus:

IC type 7476 dual JK master-slave flip-flops.

Theory: A Flip-Flop or latch is a circuit that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control inputs and will have one or more outputs. Flip-flops and latches are a fundamental building block of digital electronics systems used in computers, communications and many other types of systems.

Flip-flops and latches are used as data storage elements. Such data storage can be used for storage of state, and such a circuit is described as sequential logic. When used in a finite-state machine, the output and next state depend not only on its current state (and hence,

previous inputs). It can also be used for counting pulses, and or synchronizing variably timed input signals to some reference timing signal.

Early flip-flops were known variously as trigger circuits or multivibrators. A multivibrator is a two-state circuit; they come in several varieties, based on whether each state is stable or not: an astable multivibrator is not stable in either state, so it acts as relaxation oscillator; a monostable multivibrator makes a pulse while in the unstable state, then returns to stable state, and is known as one-shot; a bistable multivibrator has two stable states, and this is the one usually known as flip-flop.

Procedure:

The 7476 is a dual JK master-slave flip-flops with preset and clear inputs. The function table given in table 1 defines the operation of the flip-flop. The +ve transition of the CLOCK (CP) pulse changes the master flip-flop, and the (-ve) transition changes the slave flip-flop as well as the output of the circuit. In Logic Works the chip 7476 is not available, however, the generic JK flip-flop behave in exactly the same way as the 7476. The "S" represents the Preset, the " " represents the Clear, and C represents the clock pulse (CP). Verify the table by connecting Binary switches to R, S, J, K, and C. Notice that only the negative edge of the clock affects the outputs (Q, and Q')

Table 1

Input					Output	
Preset	Clear	Clock	J	K	Q	Q'
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	↓	0	0	No change	
1	1	↓	0	1	0	1
1	1	↓	1	0	1	0
1	1	↓	1	1	Toggle	

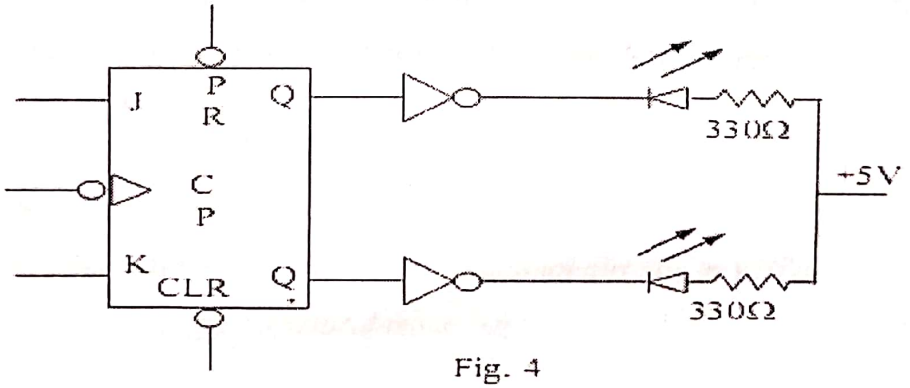


Fig. 4

Figure: A VHDL entity consisting of an interface (entity declaration) and a body (architectural description).

Entity Declaration

The entity declaration defines the NAME of the entity and lists the input and output ports. The general form is as follows.

entity NAME_OF_ENTITY **is** [**generic** *generic_declarations*);]

port (*signal_names*: **mode** *type*;
signal_names: **mode** *type*;
:
signal_names: **mode** *type*);

end [NAME_OF_ENTITY] ;

Architecture body

The architecture body specifies how the circuit operates and how it is implemented.

architecture architecture_name **of** NAME_OF_ENTITY **is**

-- Declarations

-- components declarations
-- signal declarations
-- constant declarations
-- function declarations
-- procedure declarations
-- type declarations

begin

-- Statements

end architecture_name;

Library and Packages:

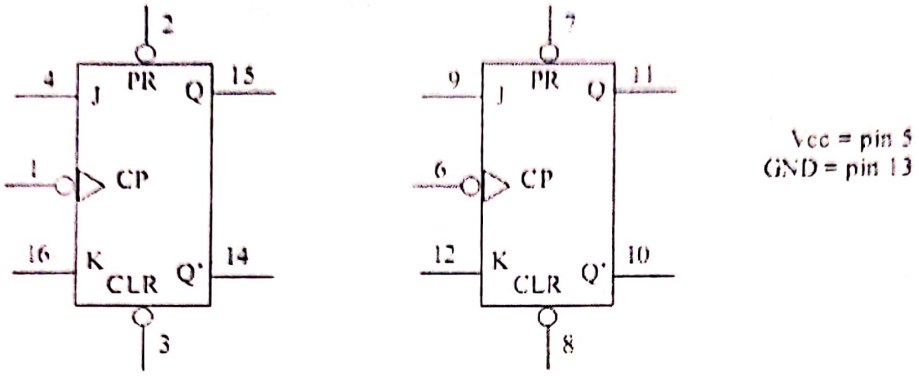
A library can be considered as a place where the compiler stores information about a design project. A VHDL package is a file or module that contains declarations of commonly used objects, data type, component declarations, signal, procedures and functions that can be shared among different VHDL models. library ieee; use ieee.std_logic_1164.all;.

Modeling Style

There are different ways of describing combinational circuits in VHDL

1. *Gate level modeling / Structural modeling*
2. *Dataflow modeling*
3. *Behavioral modeling*

In the Lab, Construct the circuit of Fig 4. Look at the data sheet for the 7476 and determine the inactive logic required at the PRE and CLR inputs. Connect the 7476 for the SET mode by connecting J = 1, K = 0. With CLOCK (CP) = 0; test the effect of PRE, CLR by putting a 0 on each, one at a time. Put CLR = 0, then pulse the clock (CP) by putting a HIGH then a LOW, on the clock. Does the CLR input override J input? Verify the operation of the JK flip flop by experimentally obtaining the characteristic table.



Procedure:

1) Connections are as per given circuit diagram
2) Logical inputs are as per given circuit diagram
3) Observe the output and verify the truth table

Conclusion: Hence, We have verified the JK Flip Flop implementation.

Sample Questions:

Explain sequential circuits.
Explain SR, D, JK and T Flip Flops with characteristics and excitation tables.
What is race condition?
Explain master – slave JK flip flop.
What is toggle condition?

EXPERIMENT NO. 10

AIM: To study VHDL and Basic Gate VHDL coding.

Objectives: To learn basic framework of VHDL Programming

CO's to be achived:

Construct test and debug digital networks using VHDL

PO's to achieved

1) An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.

2) An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.

3) An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.

4) An ability to understand real life challenges and use current techniques, skills, and modern tools necessary for computing practice.
5) An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

An ability to write VHDL code by using modern tools like Modelsim for design and simulation of simple digital circuits.

THEORY:

VHDL is an acronym for VHSIC hardware description language; VHSIC means a very high speed integrated circuit. . It is used for modelling of digital systems made of interconnection of various components

The main components are

a) Package

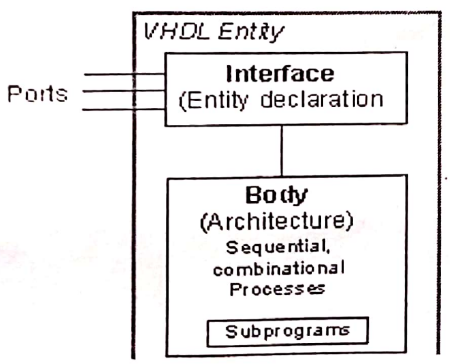
b) Entity

c)Architecture

d) Configuration

Basic Structure of a VHDL file

A digital system in VHDL consists of a design **entity** that can contain other entities that are then considered components of the top-level entity. Each entity is modeled by an entity declaration and an architecture body.

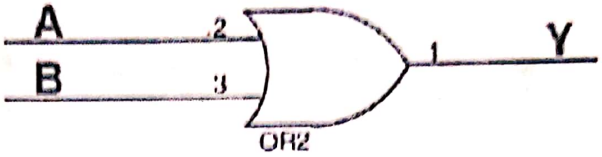


b)OR Gate: A logic gate whose output is logic '0' if and only if all of its inputs are logic '0'

Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Logic diagram

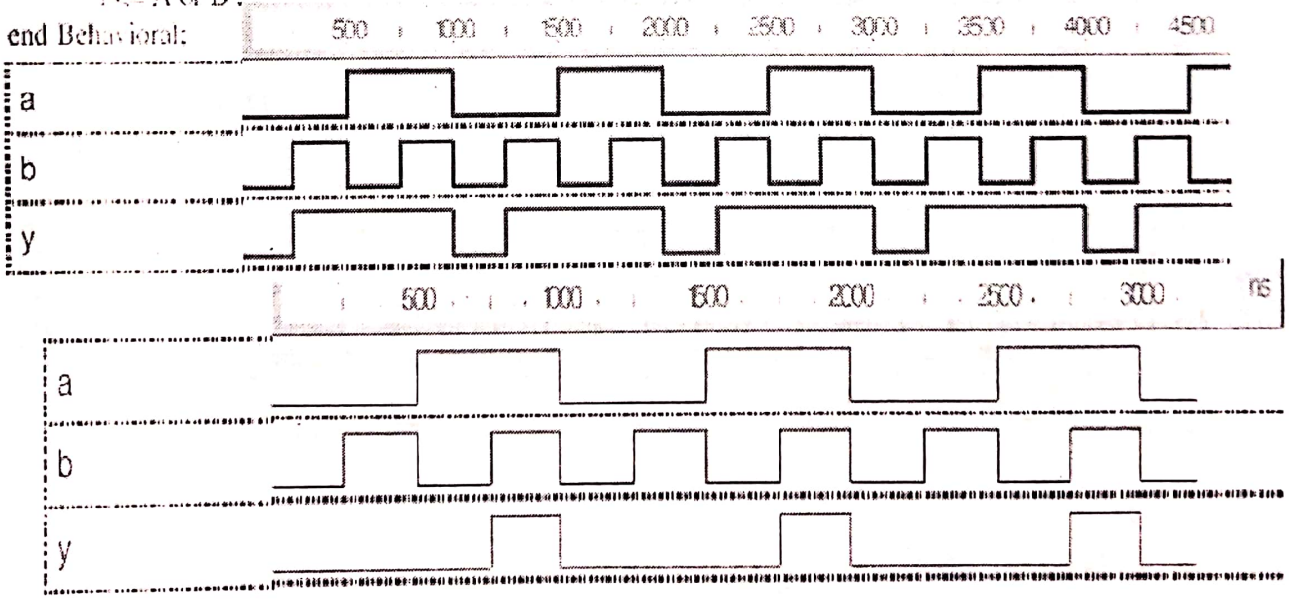


of its inputs

Y = A OR B
= A + B

VHDL Code for OR Gate:

```
--------------------------------------------------------------------------------
-- File                    :       orgate.vhd
-- Entity                  :       orgate
--------------------------------------------------------------------------------
-- File                    :       andgate.vhd
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
---------------------------- Entity Declarations ----------------------
entity orgate is
        Port(  A : in  std_logic;
               B : in  std_logic;
               Y : out std_logic
               );
end orgate;

architecture Behavioral of orgate is
begin
        Y<= A or B ;
end Behavioral;
```

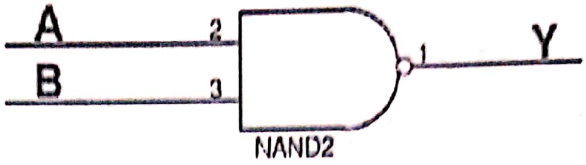


d) **NAND Gate:** A logic gate which gives logic '0' output if and only if all of its inputs are logic '1'

Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

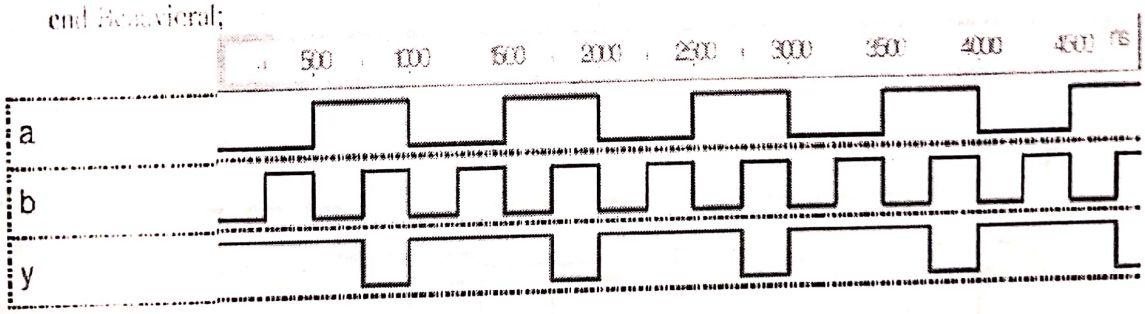
Logic diagram



Y = A NAND B
= (A. B)\

VHDL Code for NAND Gate:

```
-- File              :     nandgate.vhd
-- Entity            :     nandgate
--------------------------------------------------------------------------------
library IEEE;
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
---------------------------- Entity Declarations ----------------------
entity nandgate is
        Port( A : in  std_logic;
              B : in  std_logic;
              Y : out std_logic
              );
end nandgate;
architecture Behavioral of nandgate is
begin
        Y <= A nand B ;
    end Behavioral;
```

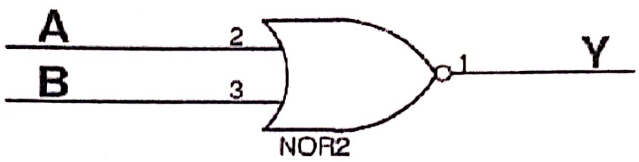


c) **NOR Gate:** A logic gate whose output logic '1' if and only if all of its inputs are logic '0'

Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

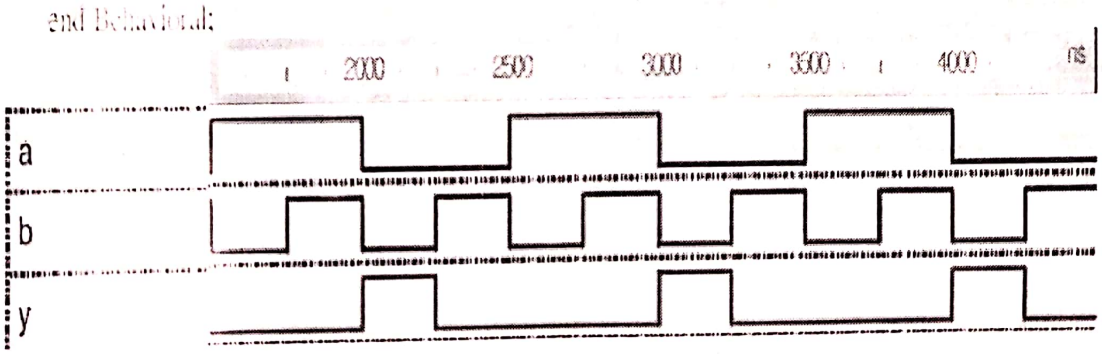
Logic diagram



Y= A NOR B
=(A+B)'

VHDL Code for NOR Gate:

```
--------------------------------------------------------------------------------
-- File                    :       norgate.vhd
-- Entity                  :       norgate
--------------------------------------------------------------------------------

--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
---------- ---------- Entity Declarations ----------
entity norgate is
        Port ( A : in  std_logic;
               B : in  std_logic;
               Y : out std_logic
               );
end norgate;

architecture Behavioral of norgate is
begin
        Y<= A nor B ;
    end Behavioral;
```

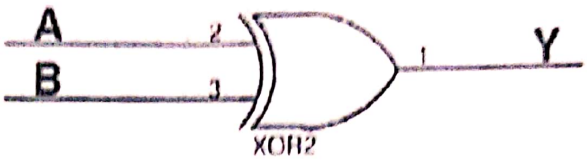


1) **EX-OR (Exclusive OR):** A logic gate whose output is logic '0' when all the inputs are equal and logic '1' when they are un equal.

Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

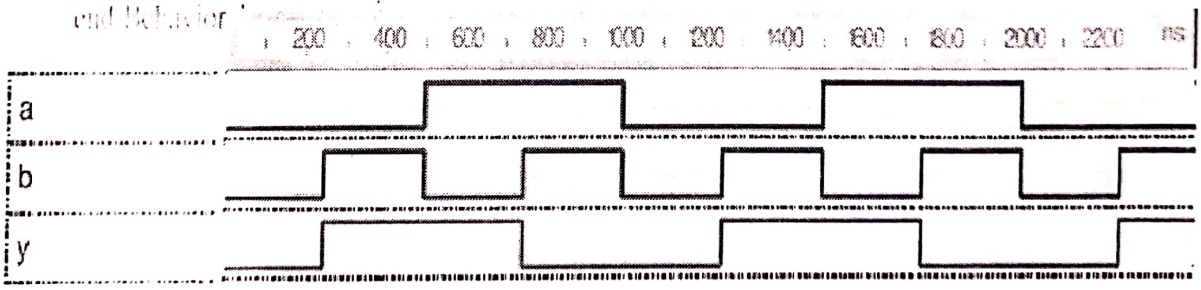
Logic diagram



Y= A EX-OR B
= A ⊕ B
= A $\bar{B}$ + $\bar{A}$B

VHDL Code for EX-OR Gate:

```
----------------------------------------------------------------------------------
-- File                    :       xorgate.vhd
-- Entity                  :       xorgate
----------------------------------------------------------------------------------

----------------------------------------------------------------------------------
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
---------------------------- Entity Declarations ----------------------------
entity xorgate is
        Port ( A : in  std_logic;
               B : in  std_logic;
               Y : out std_logic
               );
end xorgate;

architecture Behavioral of xorgate is
begin
        Y<= A xor B ;
end Behavioral;
```

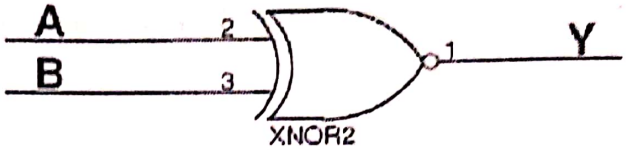


e) **EX-NOR (Exclusive -NOR) gate:** A logic gate that prodices a logic '1' only when the two inputs are equal

Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

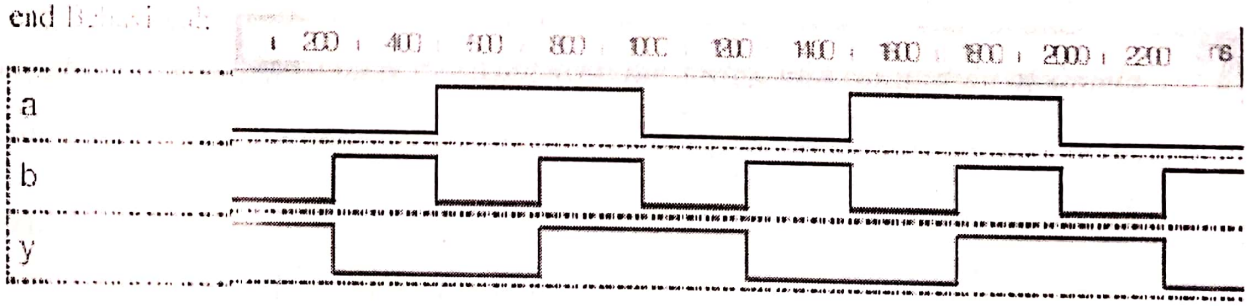
Logic diagram



Y= A XNOR B
= (A (+)B)\
= (A.B)\ + A.B

VHDL Code for EX-NOR Gate:

```
--------------------------------------------------------------------------------
-- File                    :       xnorgate.vhd
-- Entity                  :       xnorgate
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- Description             :       VHDL code to realize EX-NOR gate functionality
--------------------------------------------------------------------------------
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
---------------------------- Entity Declarations ----------------------
entity xnorgate is
        Port ( A : in  std_logic;
               B : in  std_logic;
               Y : out std_logic
               );
end xnorgate;
architecture Behavioral of xnorgate is
begin
        Y<= A xnor B ;
end Behavioral;
```



CONCLUSION:

Basic programming structure of VHDL is learned. VHDL is a hardware description language can be used to simulate simple to complex digital circuits.

Sample Questions:

1) Explain basic structure of VHDL program.
2) Define entity, architecture, package.
3) Explain different data types supported by VHDL
4) Explain advantages of simulation of digital circuits.

EXPERIMENT NO.11

AIM: Write a VHDL code for 4:1 MUX

Objectives: To simulate and synthesize digital circuit using Modelsim

CO's to be achieved:

Construct test and debug digital networks using VHDL

PO's to achieved

1) An ability to apply knowledge of Computing, Mathematics, Science and Engineering fundamentals appropriate to the discipline.
2) An ability to analyze a problem, interpret data to evaluate system performance and formulate the computing requirements appropriate to its solution and conducts experiments.
3) An ability to design, implement, and evaluate a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability and sustainability.
4) An ability to understand real life challenges and use current techniques, skills, and modern tools necessary for computing practice.
5) An ability to function effectively individually and on teams, including diverse and multidisciplinary, to accomplish a common goal.

An ability to use modern tools like Modelsim for design and simulation of simple digital circuits.

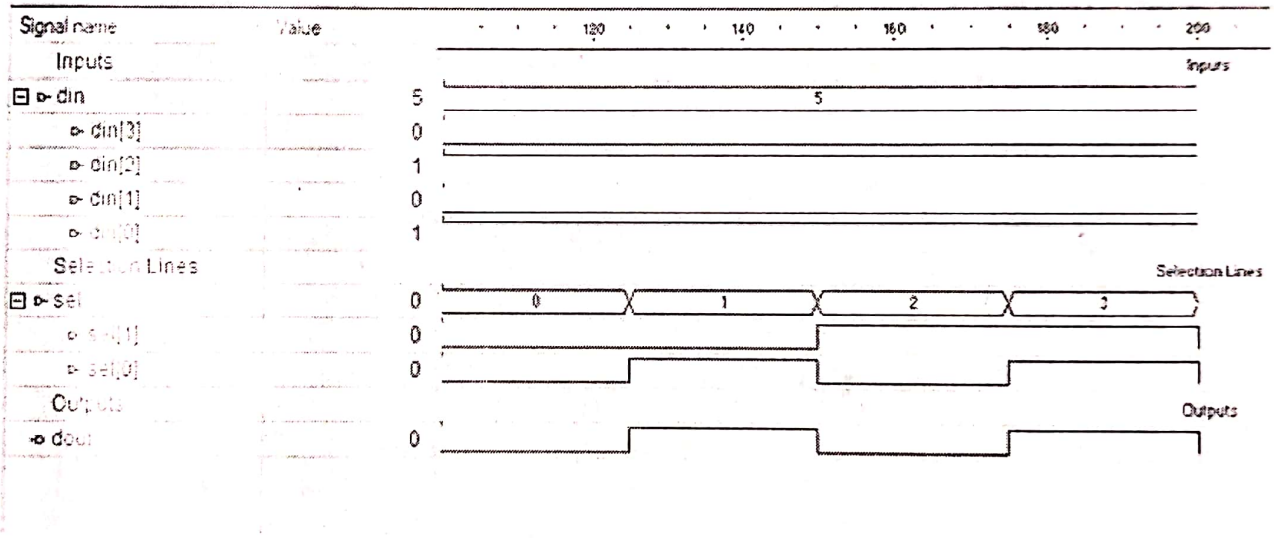
PROGRAM:

4:1 MUX VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity multiplexer_4_1 is
    port(
      din : in STD_LOGIC_VECTOR(3 downto 0);
```

```
        sel : in STD_LOGIC_VECTOR(1 downto 0);
        dout : out STD_LOGIC
        );
end multiplexer_4_1;

architecture multiplexer4_1_arch of multiplexer_4_1 is
begin
   mux : process (din,sel) is
   begin
      if (sel="00") then
         dout <= din(3);
      elsif (sel="01") then
         dout <= din(2);
      elsif (sel="10") then
         dout <= din(1);
      else
         dout <= din(0);
      end if;
   end process mux;

end multiplexer4_1_arch;
```

OUTPUT:



CONCLUSION:

Basic programming structure of VHDL is learned. VHDL, a hardware description language is used to simulate, synthisize and test 4:1.

SAMPLE QUESTIONS:

1) Implement and simulate full adder using 8:1 Multiplexer. Write VHDL program using structural, behavioral and dataflow model.
2) Explain method of execution, simulation and debugging of VHDL program