ABES Engineering College, Ghaziabad

Affiliated to Dr. APJ AKTU Lucknow

Department of Computer science and engineering

Lab Manual

Session 2020-21 (Odd Semester)

Name : HARSH MOHAN

Roll No. : 1900320100065

Subject Name : Computer Organization Lab

Subject Code : KCS352

Semester : III semester

LIST OF EXPERIMENTS

S. No.	Experiment Name	CO Mapping
1.	Implementation of basic logic gates.	CO1
2.	Implementing HALF ADDER, FULL ADDER using basic logic gates.	CO4
3.	Implementing Binary -to -Gray, Gray -to -Binary code conversions.	CO1
4.	Implementing 4*1 and 8*1 multiplexer.	CO1
5.	Design of an 8-bit Arithmetic Logic Unit	CO1
6.	Design the data path of a computer from its register transfer language.	CO1
7.	Designing large Memory Unit using smaller size memory units	CO1
8.	. Design a bus system using Multiplexer for data transfer from register.	CO1
9.	Implement an array multiplier of 2*2 and 4*3 bit.	CO1
10.	Implement 4 bit binary incrementer and decrementer	CO1

EXPERIMENT 1

Problem Statement: Implementation of logic gates

Apparatus Required:

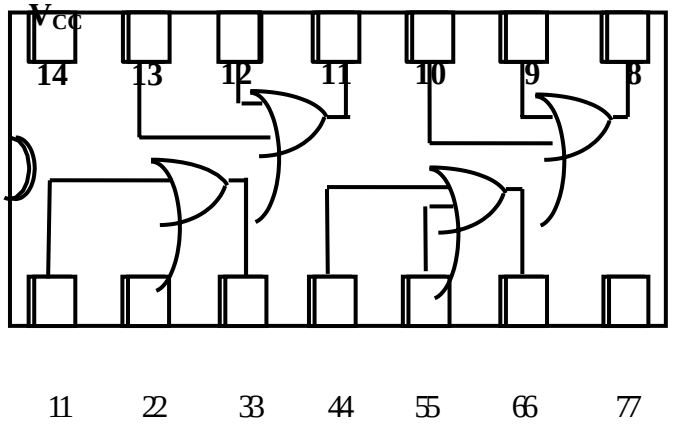
S.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

S.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

Theory:

- Details of IC used and pin configurations.
- Working of logic gates.

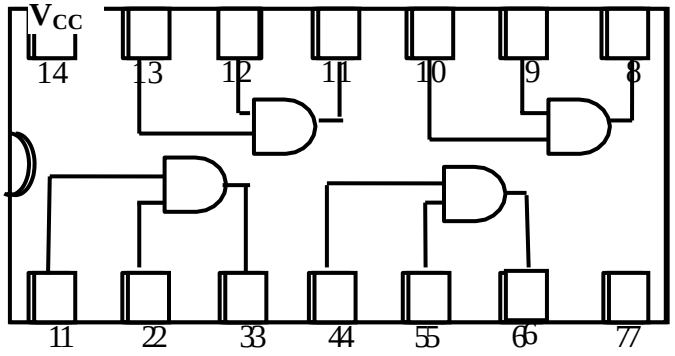
1. **OR GATE**:



INPUT A	INPUT B	OUTPUT Y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

TRUTH TABLE

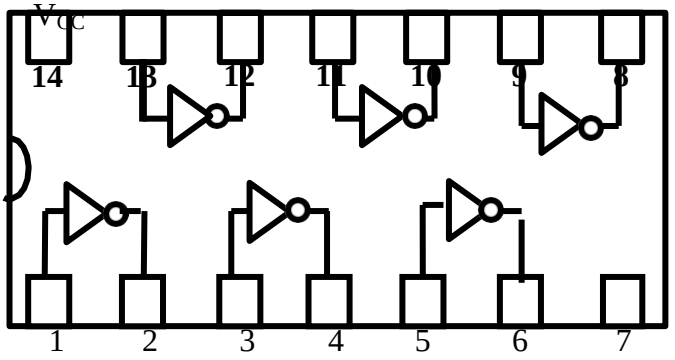
2. AND GATE:



TRUTH TABLE

INPUT A	INPUT B	OUTPUT Y
0	0	0
0	1	0
1	0	0
1	1	1

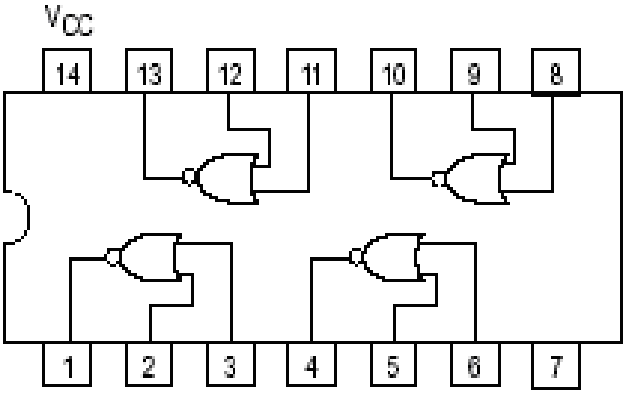
3. NOT GATE:



TRUTH TABLE

INPUT A	OUTPUT Y
0	1
1	0

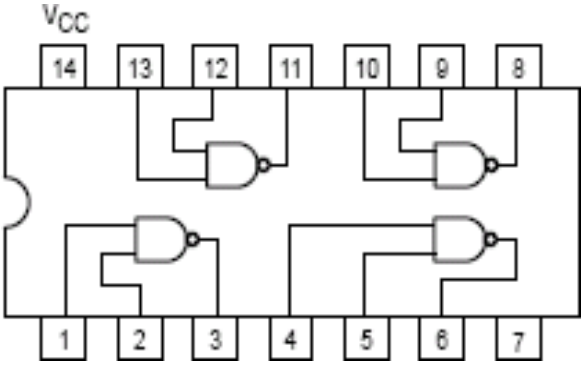
4. NOR GATE



TRUTH TABLE

INPUT A	INPUT B	OUTPUT Y
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

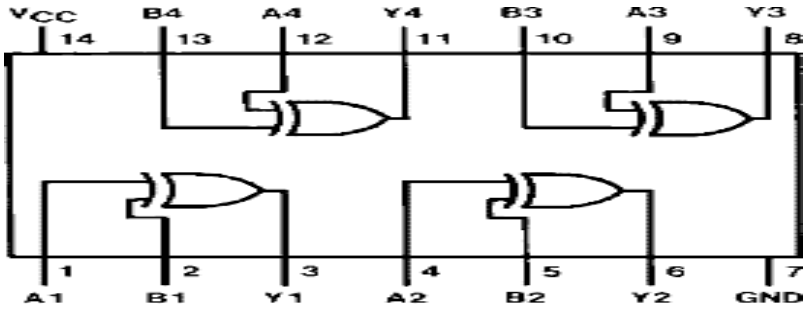
5. NAND GATE:



TRUTH TABLE

INPUT A	INPUT B	OUTPUT Y
0	0	1
0	1	1
1	0	1
1	1	0

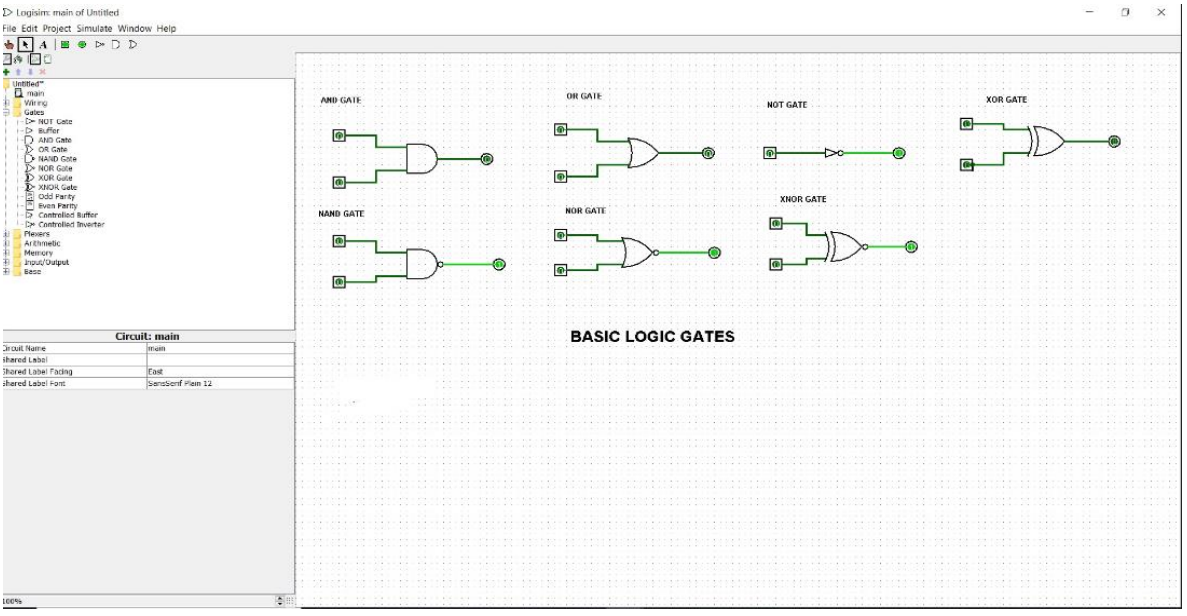
6. XOR GATE:



TRUTH TABLE

INPUT A	INPUT B	OUTPUT Y
0	0	0
0	1	1
1	0	1
1	1	0

Implementation:



Result: All Logic Gates are verified.

EXPERIMENT 2

Problem Statement: Design and implementation of HALF ADDER, FULL ADDER using basic logic gates.

Apparatus Required:

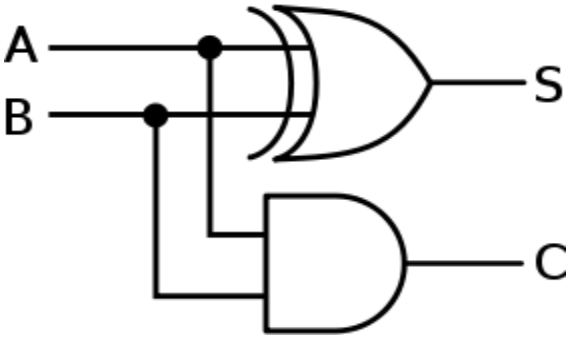
S.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

S.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

Theory:

a) To design and implement half adder using logic gates

HALF ADDER

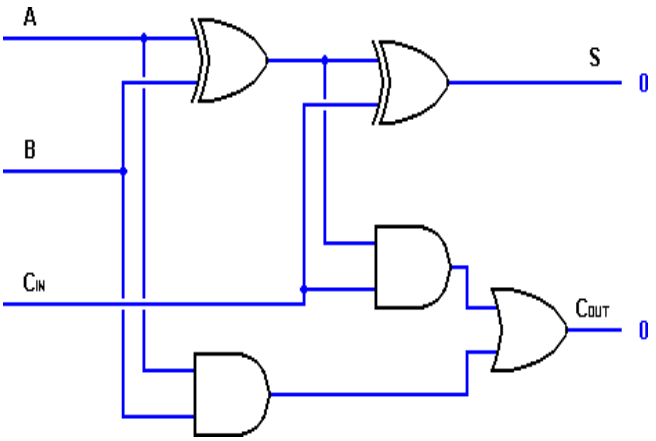


INPUT A	INPUT B	OUTPUTS	
		S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

CIRCUIT DIAGRAM **TRUTH TABLE**

b) To design and implement full adder using logic gates

FULL ADDER

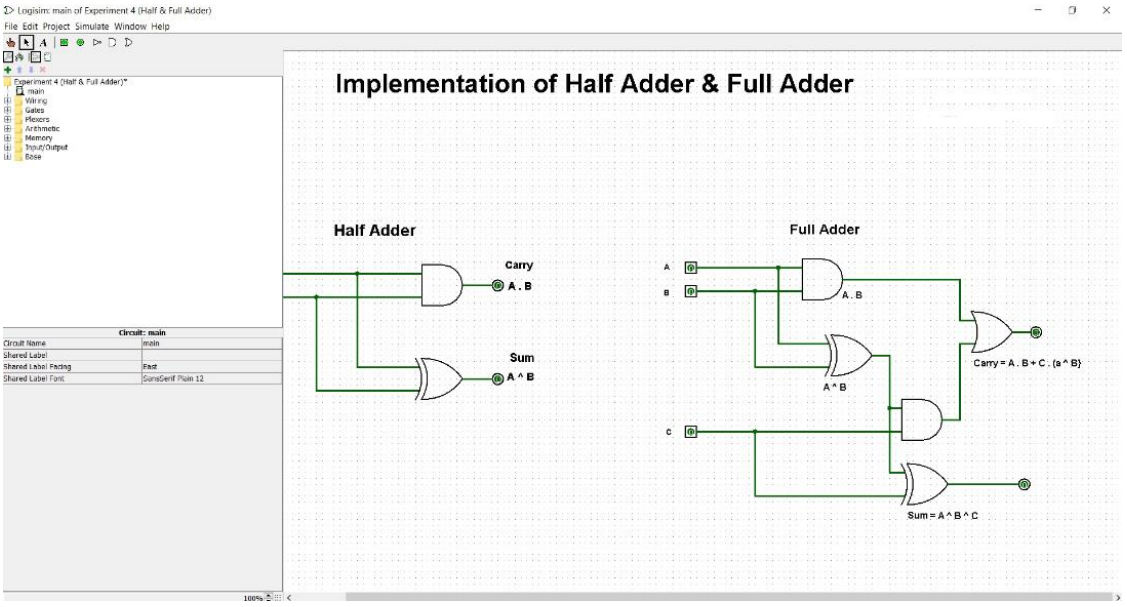


CIRCUIT DIAGRAM

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

TRUTH TABLE

Implementation:



Result: All logical circuits have been implemented & verified through truth table.

EXPERIMENT 3

Problem Statement: Design and implementation of Binary to Gray, Gray to Binary Code conversions

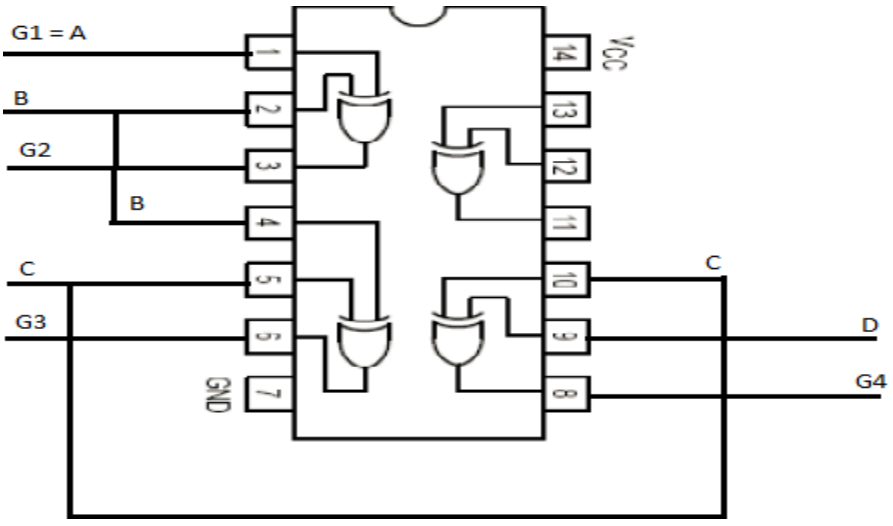
Apparatus Required:

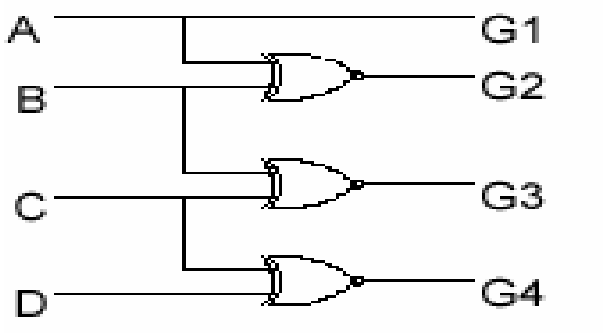
1S.No.	Equipments	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

S.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

Theory:

a) To design and implement Binary to Gray Code conversions



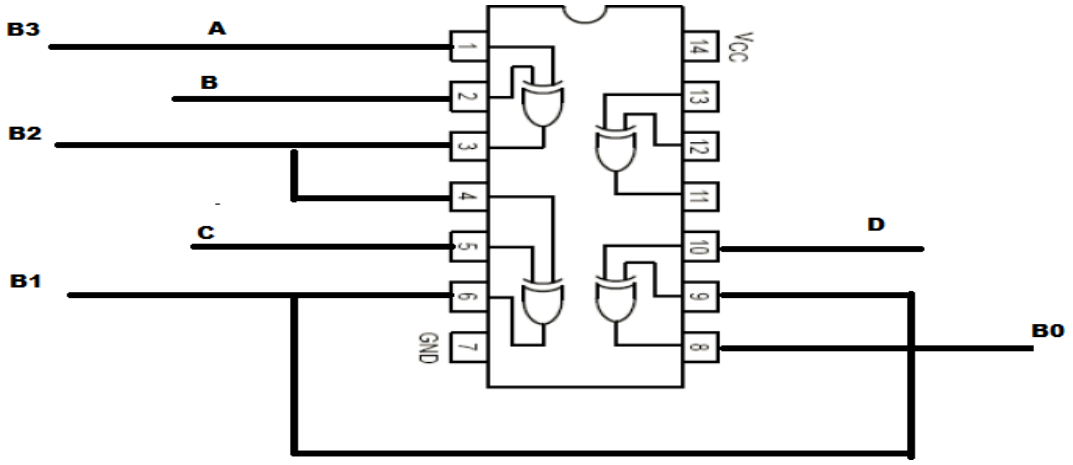


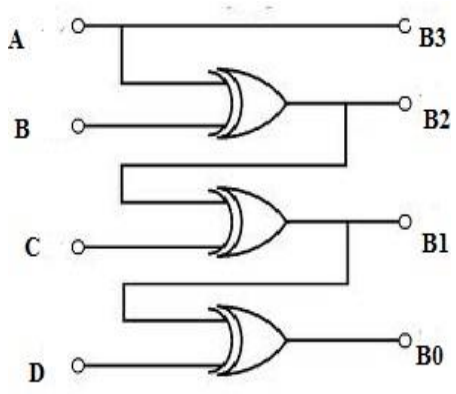
Circuit Diagram of Binary to Gray Code Converter

A	B	C	D	G_4	G_3	G_2	G_1
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

Truth Table

b) To design and implement Binary to Gray Code conversions

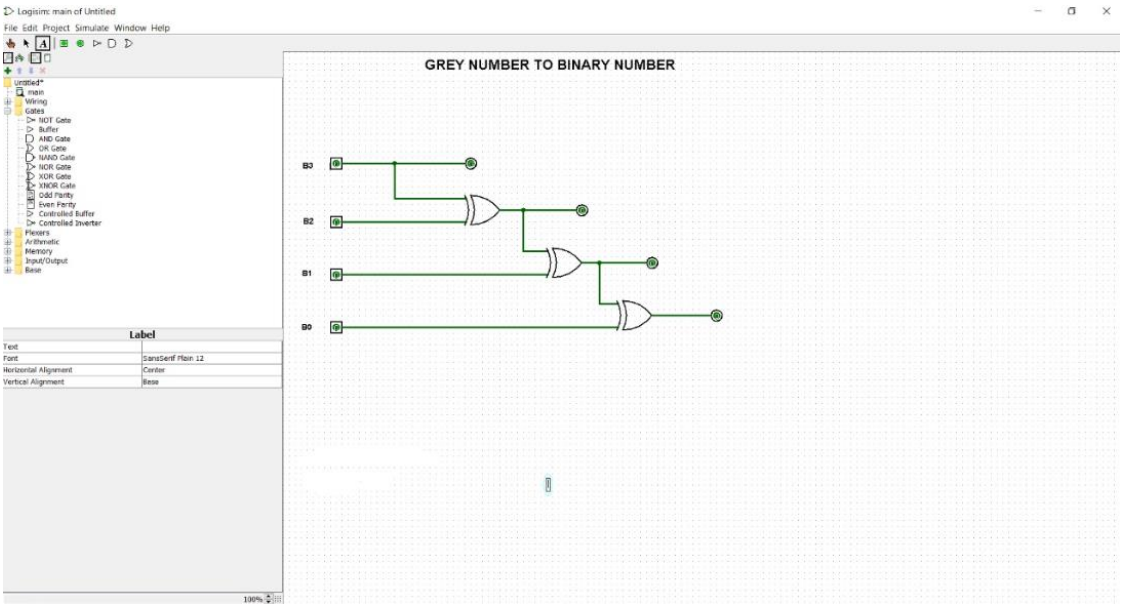


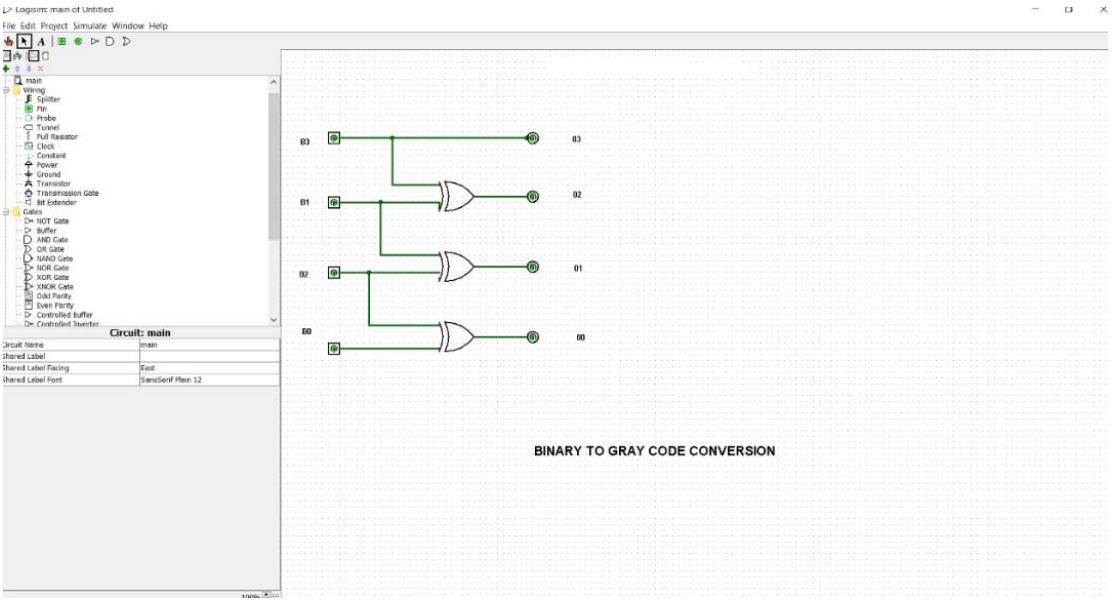


Circuit Diagram for Gray to Binary Code Converter

INPUTS				OUTPUTS			
A	B	C	D	B_3	B_2	B_1	B_0
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

Implementation:





Result & Conclusion: Binary to gray and gray to binary code converter has been designed using EXOR gate and its truth table verified.

EXPERIMENT 4

Problem Statement: Design and implementation of 3-8 line DECODER, 4x1 MUX and 8x1 MULTIPLEXERS

Apparatus Required :

S.No.	Equipment's	Specification	Quantity
1	Digital IC Trainer kit	-	1
2	Digital Multimeter		1

S.No.	Components	Specification	Quantity
1	Digital ICs	7400, 7402, 7404, 7408, 7432, 7486.	1 each
2	Patch cords	-	6

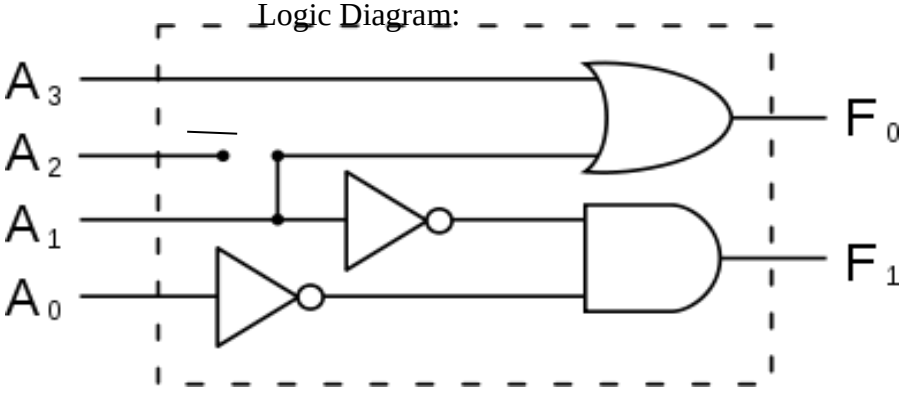
Theory:

a) 4 to 2 encoder using logic gates:

Truth Table

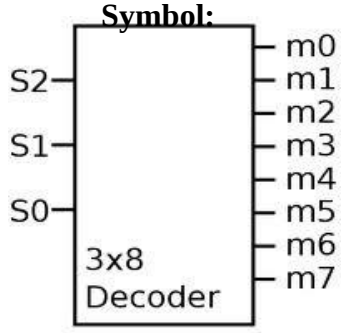
I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Logic Diagram:



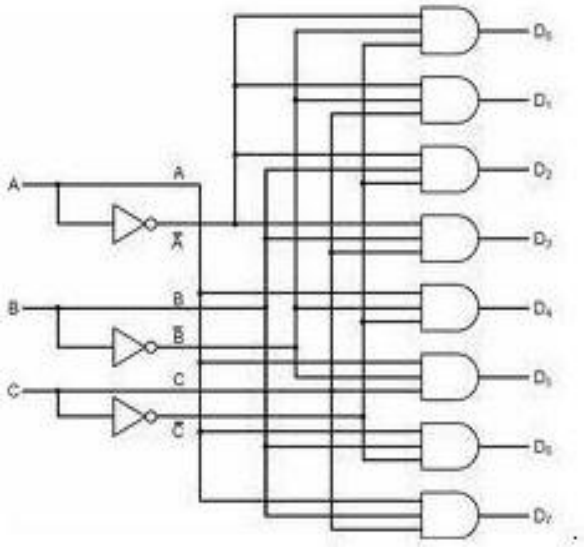
b) 3 to 8 decoder using logic gates:

Symbol:



Truth table:

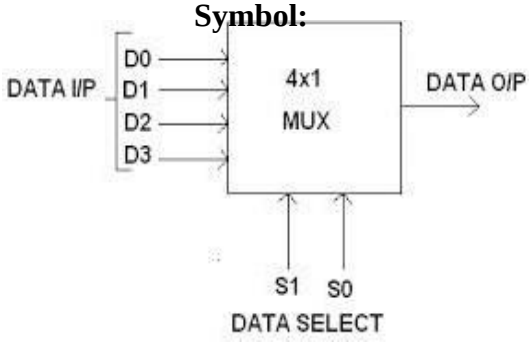
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



Logic Diagram of 3 to 8 decoder

c) 4 to 1 Multiplexer:

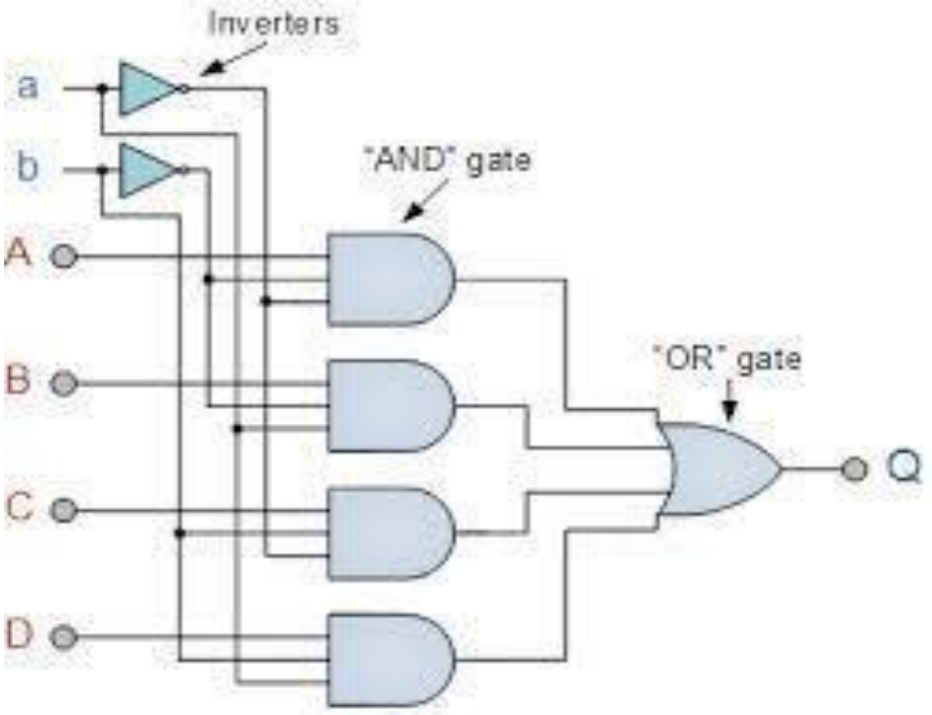
Symbol:



Truth table:

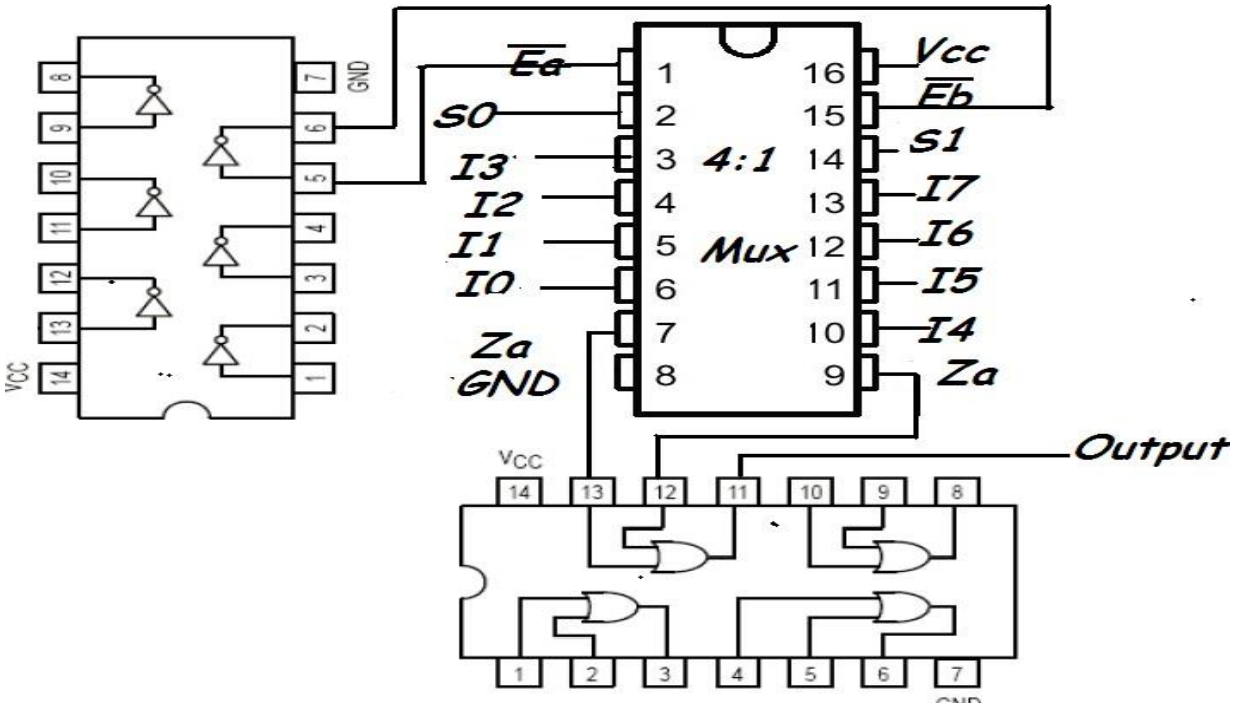
Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

Logic Diagram:

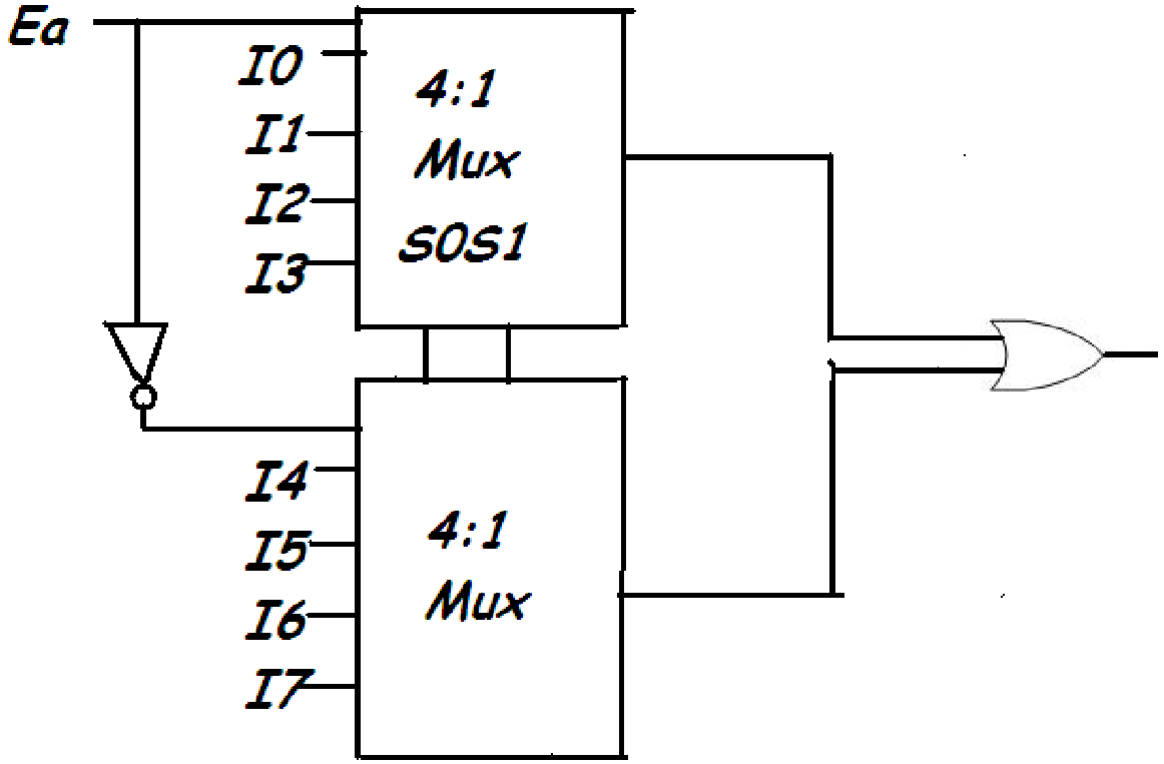


$$Q=\bar{a}\bar{b}A+\bar{a}\bar{b}B+a\bar{b}C+abD$$

d) 8x1 Multiplexer



Pin diagram of 8:1 Mux using two 4:1 Mux



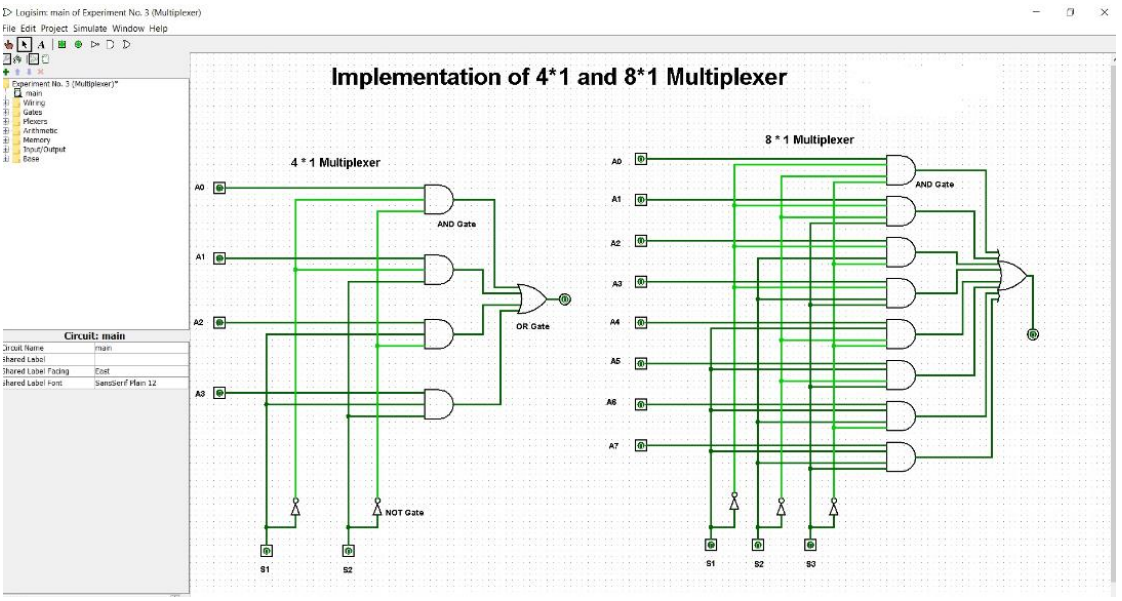
Circuit of 8:1 Mux using dual 4:1 Mux

Truth Table of 8:1 Mux Using Dual 4:1 Mux

Select Lines			Inputs								Output		
E_a	S_0	S_1	I_o	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Z_a	Z_b	Y
0	0	0	0	×	×	×	×	×	×	×	0	×	0
0	0	0	1	×	×	×	×	×	×	×	1	×	1
0	0	1	×	0	×	×	×	×	×	×	0	×	0
0	0	1	×	1	×	×	×	×	×	×	1	×	1
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

Implementation:



Result: 3-8 line decoder, 4x1 and 8x1 Mux have been implemented & verified through truth table.

EXPERIMENT 5

Problem Statement: Design of an 8- bit ARITHMETIC LOGIC UNIT.

Apparatus Required :

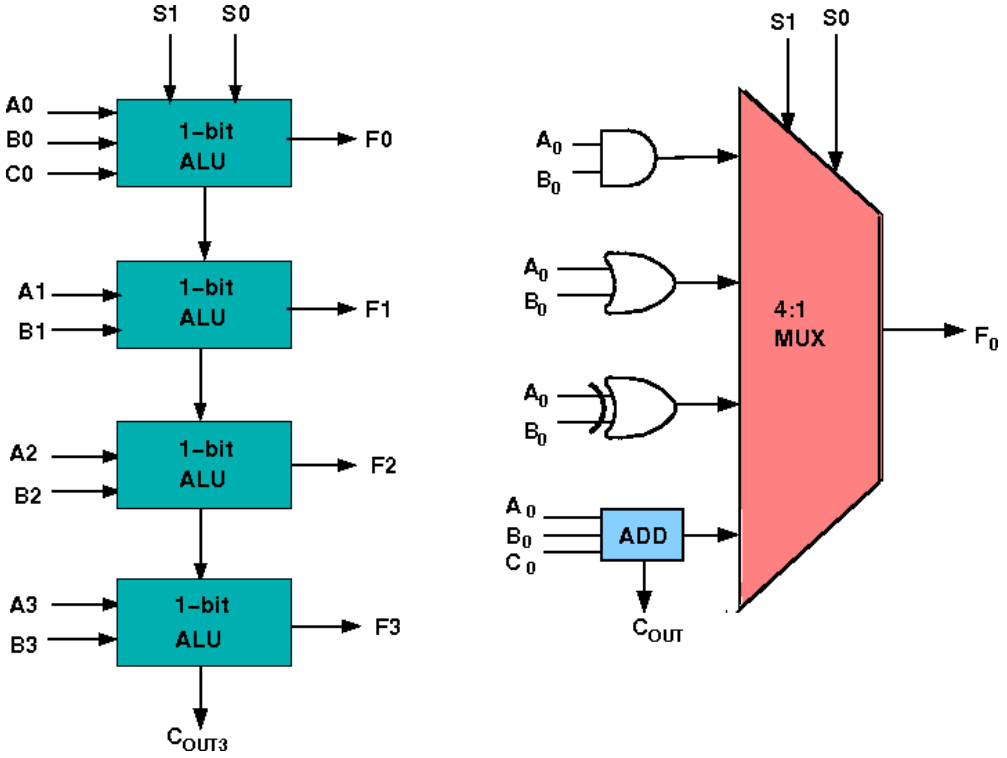
S.No.	Equipments	Specification	Quantity
1	Logic Simulator	-	1

Theory:

ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction, division, multiplication and logical operations like and, or, xor, nand, nor etc. A simple block diagram of a 4 bit ALU for operations and, or, xor and Add is shown in the Logic diagram.

LOGIC DIAGRAM:

Block diagram of a 4 bit ALU



Design Issues :

The circuit functionality of a 1 bit ALU is shown here, depending upon the control signal S1 and S0 the circuit operates as follows:

for Control signal S1 = 0 , S0 = 0, the output is A And B,

for Control signal S1 = 0 , S0 = 1, the output is A Or B,

for Control signal S1 = 1 , S0 = 0, the output is A Xor B,

for Control signal S1 = 1 , S0 = 1, the output is A Add B.

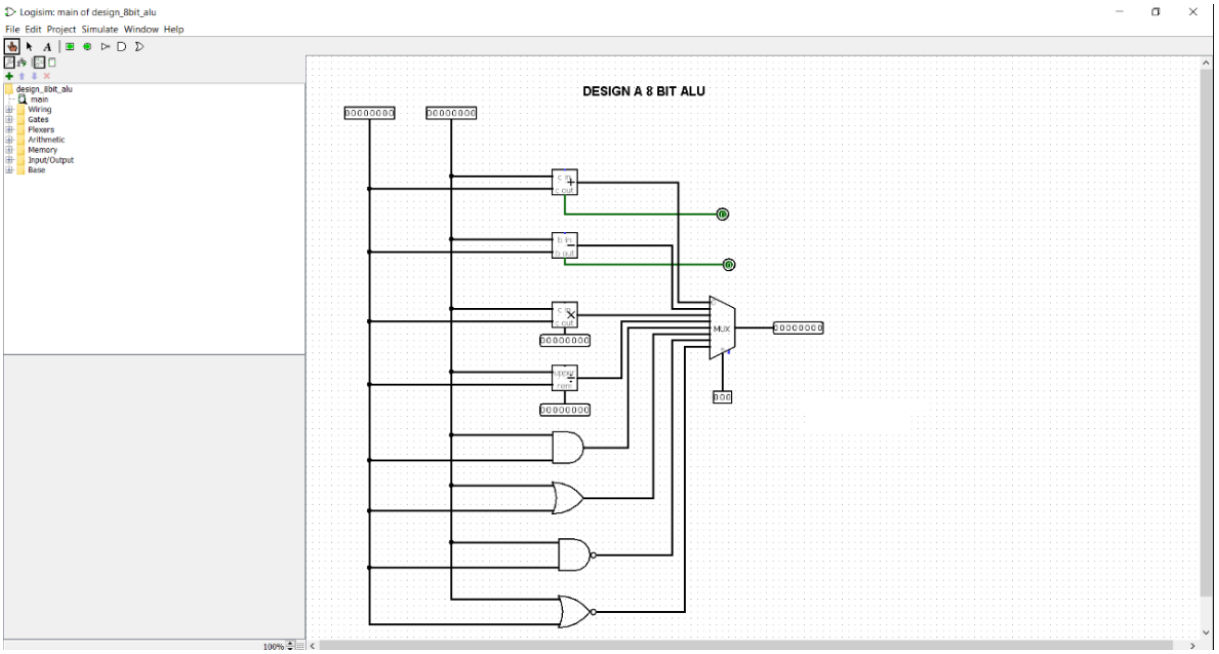
The truth table for 16-bit ALU with capabilities similar to 74181 is shown here:

Required functionality of ALU (inputs and outputs are active high)

Mode Select Inputs				F_n for active HIGH operands	
				Logic	Arithmetic (note 2)
S3	S2	S1	S0	(M = H)	(M = L) **(C_n=L)**
L	L	L	L	A'	A
L	L	L	H	A'+B'	A+B
L	L	H	L	A'B	A+B'
L	L	H	H	Logic 0	minus 1
L	H	L	L	(AB)'	A plus AB'
L	H	L	H	B'	(A + B) plus AB'
L	H	H	L	A $\oplus$ B	A minus B minus 1
L	H	H	H	AB'	AB minus 1
H	L	L	L	A'+B	A plus AB
H	L	L	H	(A $\oplus$ B)'	A plus B
H	L	H	L	B	(A + B') plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	Logic 1	A plus A (Note 1)
H	H	L	H	A+B'	(A + B) plus A
H	H	H	L	A+B	(A + B') plus A
H	H	H	H	A	A minus 1

The L denotes the logic low and H denotes logic high.

Implementation:



Result: Verified the design of an 8 bit ALU.

EXPERIMENT 6

Problem Statement: Design the data path of a computer from its register transfer language.

Apparatus Required:
1. Registers
2. AND Gates
3. OR Gate
4. Connecting Wires

Theory: The symbolic notation used to describe the micro-operation transfers among registers is called Register Transfer Language.

The term "register transfer" implies the availability of hardware logic circuits that can perform a stated micro-operation and transfer the result of the operation to the same or another register.

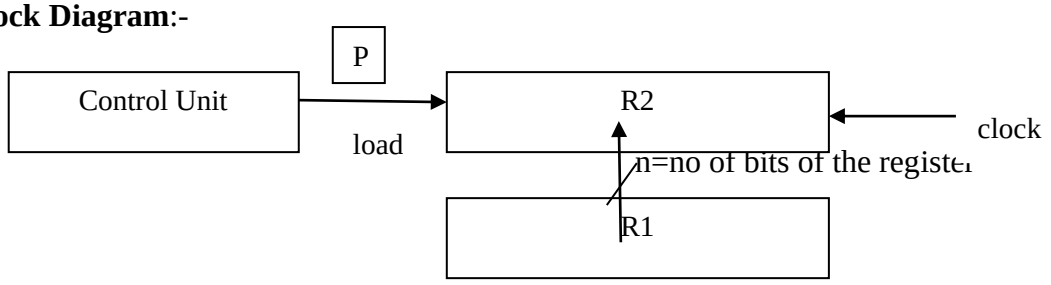
A statement that specifies a register transfer implies that circuits are available from the outputs of the source register to the inputs of the destination register and that the destination register has a parallel load capacity.
If the transfer is to occur under a predetermined condition i.e

If(P=1) then R2←R1

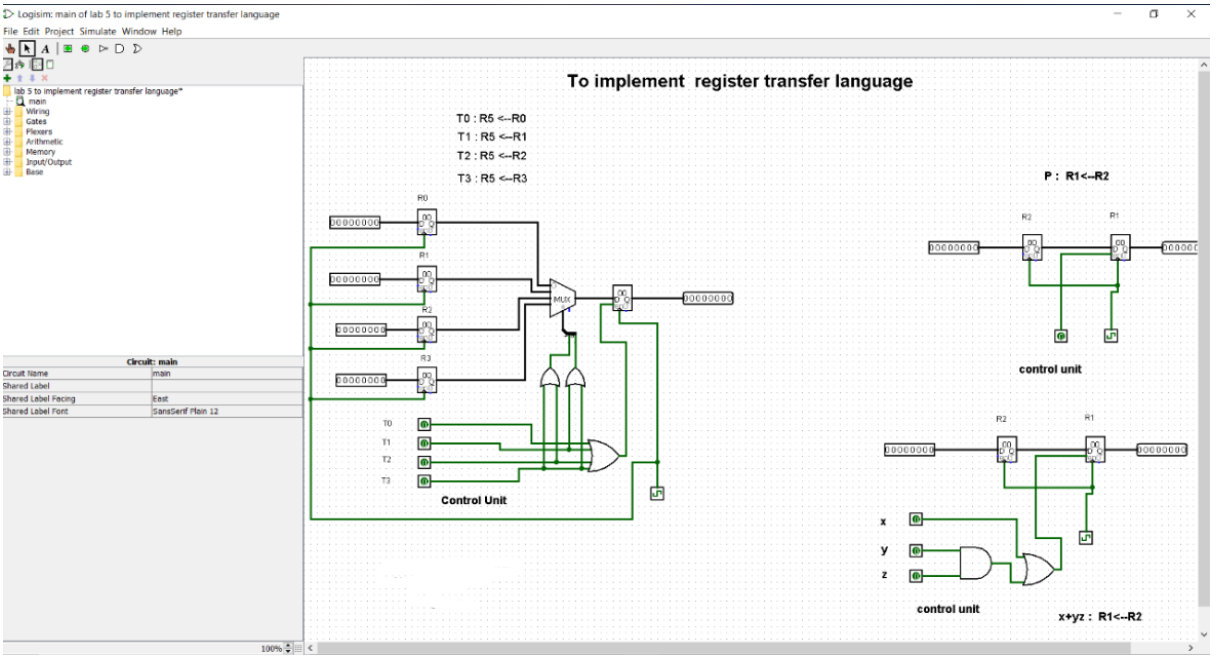
Where P is the control signal generated in the control section. A Control Function is a Boolean variable that is equal to 0 or 1

P: R2←R1

Block Diagram:-



Implementation:



Result:- The circuit has been designed according to the given Register Transfer statement and verified.

Precautions:-

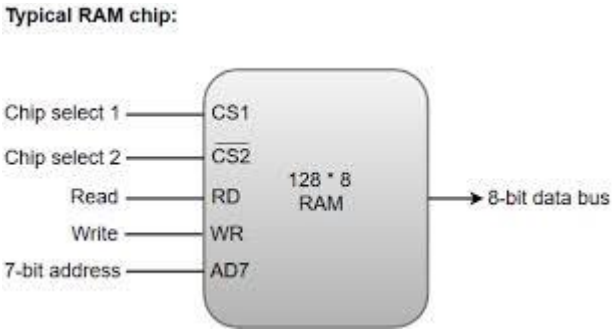
1. All Connections should be according to the block diagram
2. All Connections should be done in a proper way, i.e no open wire or dangling wire
3. Reading should be taken carefully.

EXPERIMENT 7

Problem Statement: Designing large Memory Unit using smaller size memory units.

Apparatus Required:
1. Decoder
2. 4 RAMs

Theory: A memory unit is a collection of storage cells together with associated circuits needed to transform information in and out of the device. Memory cells which can be accessed for information transfer to or from any desired random location is called random access memory (RAM).
The block diagram of a memory unit is:



A basic RAM cell has been provided here as a component which can be used to design larger memory units.
Formula for numbers of IC required is:

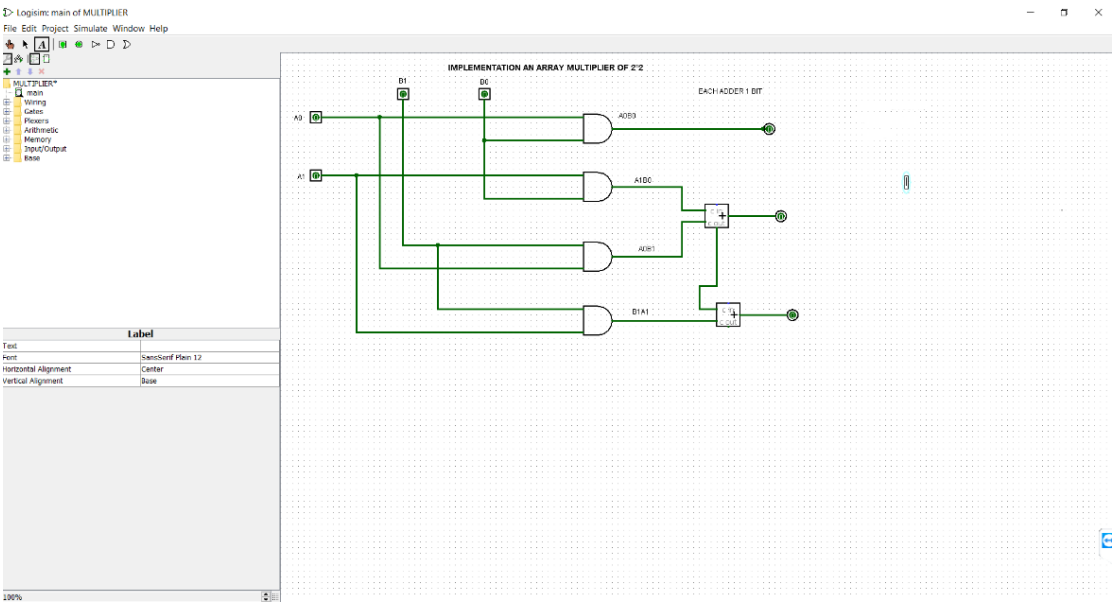
$$\text{No. of IC required} = \frac{\text{Size of memory unit (required)}}{\text{Size of memory unit (given)}}$$

Formula for Size of Decoder is:

$$\text{Size of Decoder} = n * 2^{n}$$

Where n = no. of address line required – no. of address line given

Implementation: -



Result:

For Designing a larger size memory, we need to fin no. of IC required. In this case,
No. of IC required is = 1024*8 / 256*8 = 4
So, we need 4 RAM in this circuit.

No. of Address Line for 1024 =10
No. of Address Line for 256 = 8
n = 10-8 = 2
size of decoder = 2 x 2^2 = 2 x 4.

EXPERIMENT 8

Problem Statement:- Desgin Bus System using Multiplexer to transfer data from registers

Objective:
1.Design a bus system using mltiplexer and register
Apparatus Required:
1.Bread Board Trainer Kit
2.4x1 Multiplexer
3.4 Bit Register
4.Connecting Wires

Theory:-
A digital system composed of many registers, and paths must be provided to transfer information from one register to another. The number of wires connecting all of the registers will be excessive if separate lines are used between each register and all other registers in the system.

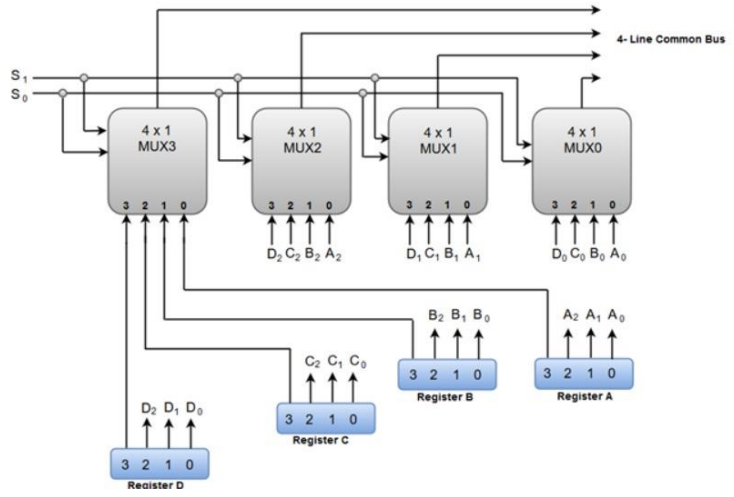
A bus structure, on the other hand, is more efficient for transferring information between registers in a multi-register configuration system.

A bus consists of a set of common lines, one for each bit of register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during a particular register transfer.

The following block diagram shows a Bus system for four registers. It is constructed with the help of four 4 * 1 Multiplexers each having four data inputs (0 through 3) and two selection inputs (S1 and S2).

We have used labels to make it more convenient for you to understand the input-output configuration of a Bus system for four registers. For instance, output 1 of register A is connected to input 0 of MUX1.

Block Diagram



The two selection lines S1 and S2 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus.

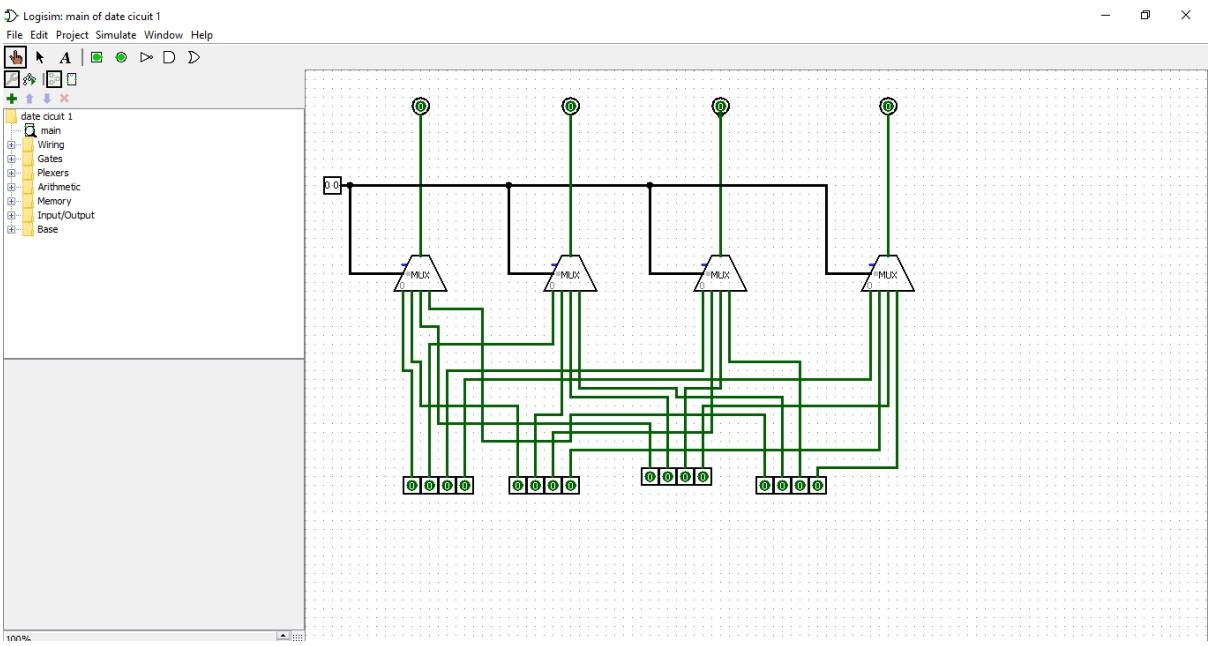
When both of the select lines are at low logic, i.e. S1S0 = 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that forms the bus. This, in turn, causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers.

Similarly, when S1S0 = 01, register B is selected, and the bus lines will receive the content provided by register B

S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

TRUTH TABLE

Implementation: -



Result: - The observation table has been verified

EXPERIMENT 9

Problem Statement:- Implement an array multiplier of 2*2 and 4*3 bit.

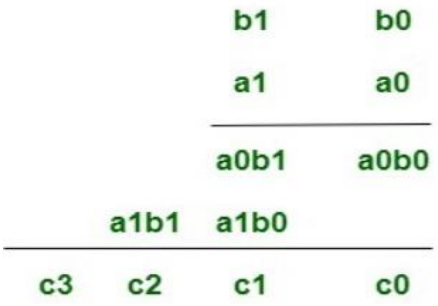
Objective:
1.Design a 2*2 Array Multiplier.
2.Design a 4*3 Array Multiplier.

Apparatus Required:
1.Bread Board Trainer Kit
2.Half Adders
3.And Gates
4.Connecting Wires

Theory:-
An array multiplier is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders. This array is used for the nearly simultaneous addition of the various product terms involved. To form the various product terms, an array of AND gates is used before the Adder array.

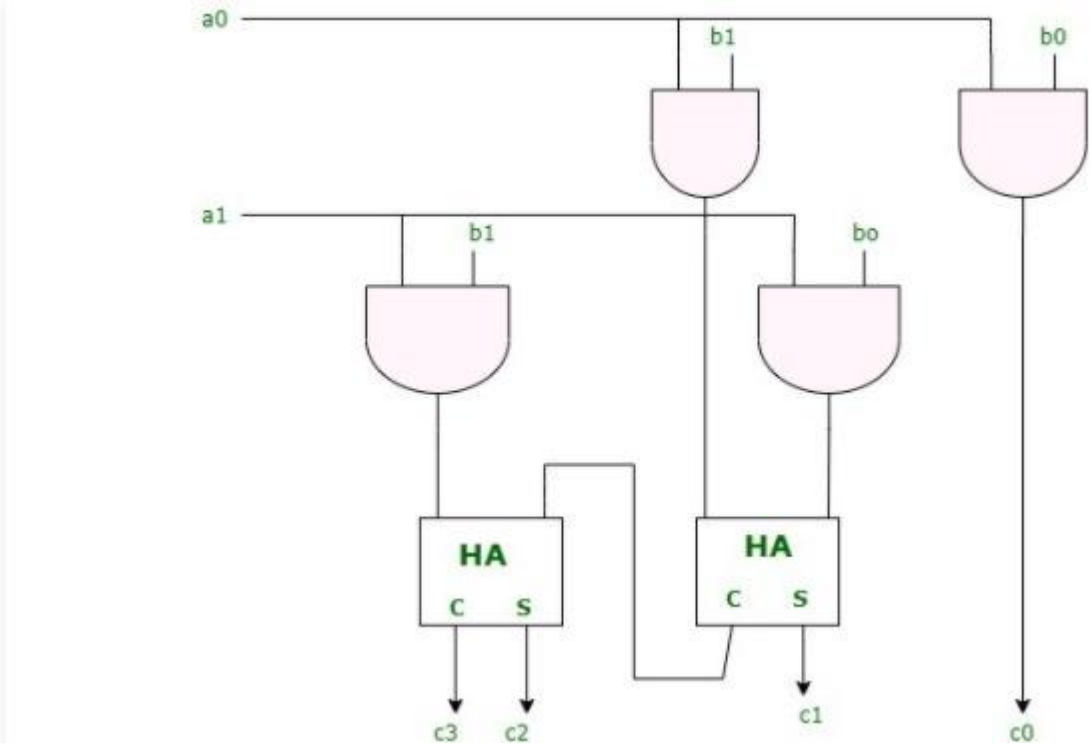
Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro-operations. The multiplication of two binary numbers can be done with one micro-operation by means of a combinational circuit that forms the product bits all at once. This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that form the multiplication array. However, an array multiplier requires a large number of gates, and for this reason it was not economical until the development of integrated circuits.

For implementation of array multiplier with a combinational circuit, consider the multiplication of two 2-bit numbers as shown in figure. The multiplicand bits are b1 and b0, the multiplier bits are a1 and a0, and the product is
c3c2c1c0

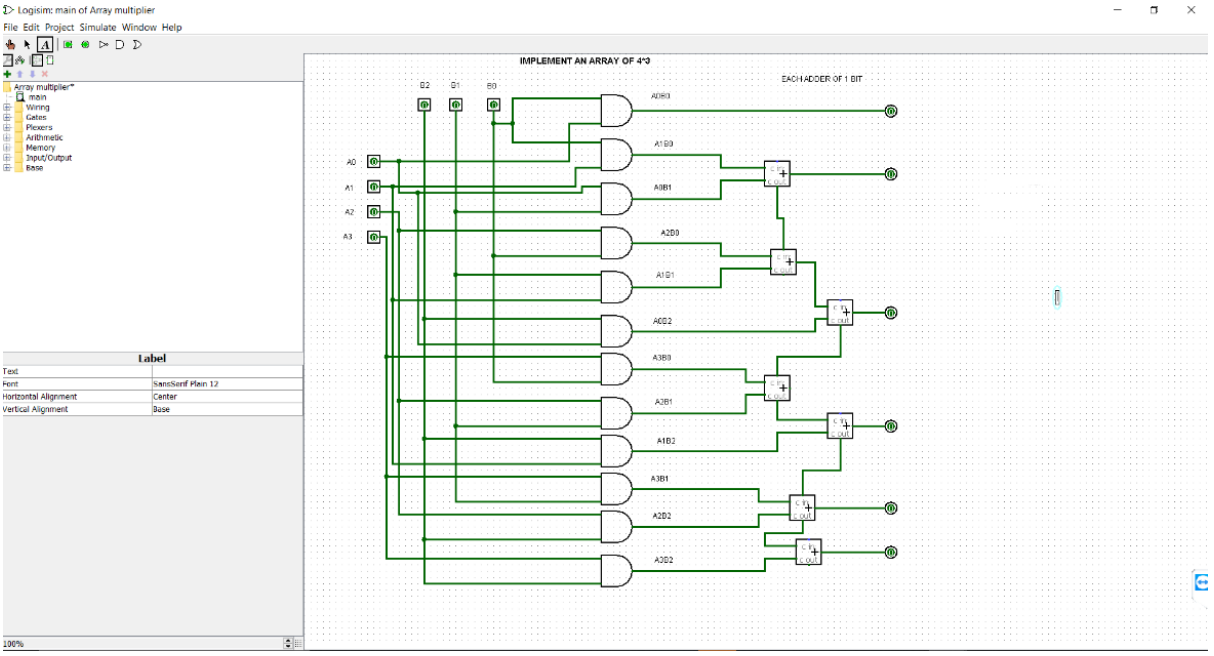


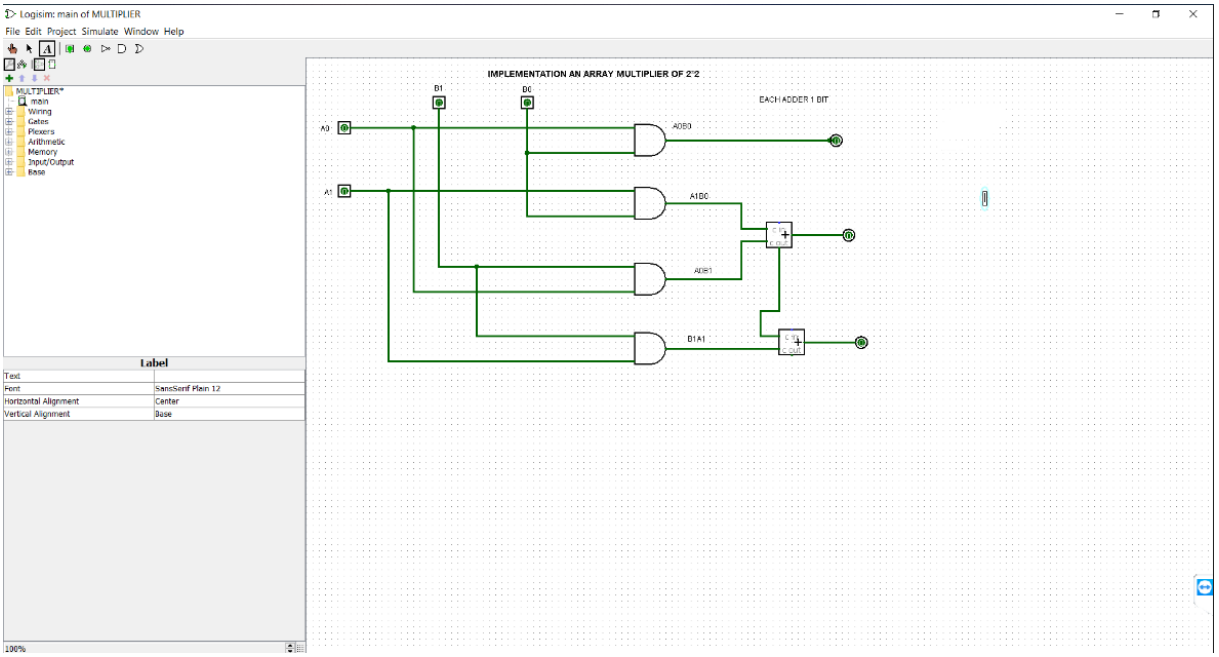
Assuming A = a1a0 and B= b1b0, the various bits of the final product term P can be written as:-
1. P(0)= a0b0
2. P(1)=a1b0 + b1a0
3. P(2) = a1b1 + c1 where c1 is the carry generated during the addition for the P(1) term.
4. P(3) = c2 where c2 is the carry generated during the addition for the P(2) term.

Block Diagram



Implementation: -





Result:- All outputs are verified.

EXPERIMENT 10

Problem Statement: Implement 4 bit binary incrementer and decrementer.

Apparatus Required:

1. Bread Board Trainer Kit
2. Half Adders
3. And Gates
4. Connecting Wires

Theory: The binary incrementer increases the value stored in a register by '1'. For this, it simply adds '1' to the existing value stored in a register. It is made by cascading 'n' half adders for 'n' number of bits i.e. the storage capacity of the register to be incremented. Hence, a 4-bit binary incrementer requires 4 cascaded half adder circuits.

Block Diagram of Incrementer :

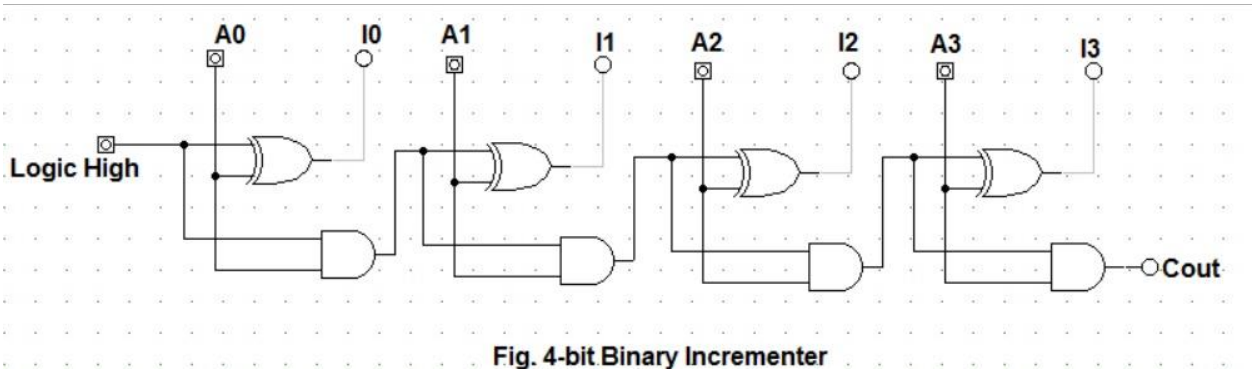


Fig. 4-bit Binary Incrementer

Observed Values:

Following set of values were obtained in observation.

1. 0011 => 0100
2. 1010 => 1011
3. 1101 => 1110
4. 0010 => 0011
5. 1111 => 0000; Cout = 1

The binary decrementer decreases the value stored in a register by '1'. For this, we can simply add '1' to the each bit of the existing value stored in a register. This is basically the concept of two's complement used for subtraction of '1' from given data. It is made by cascading 'n' full adders for 'n' number of bits i.e. the storage capacity of the register to be decremented. Hence, a 4-bit binary decrementer requires 4 cascaded full adder circuits. As stated above we add '1111' to 4 bit data in order to subtract '1' from it.

Block Diagram of decrementer :

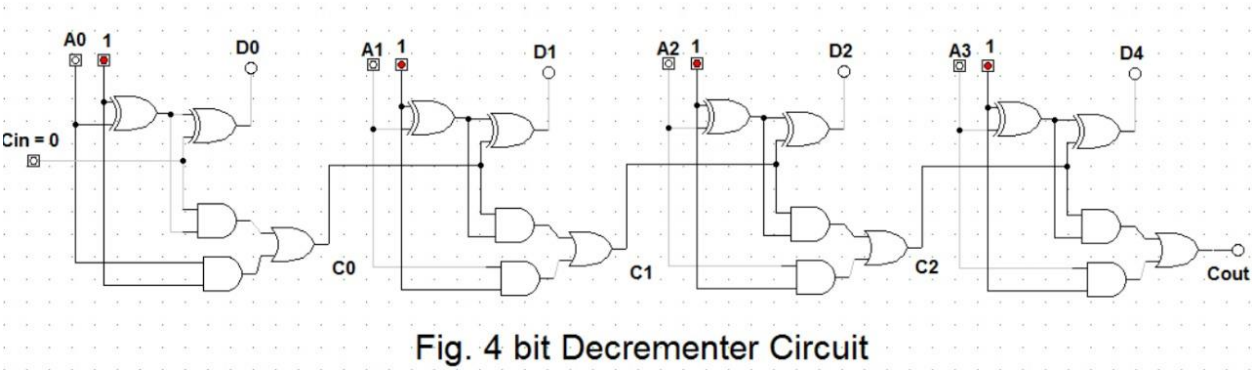


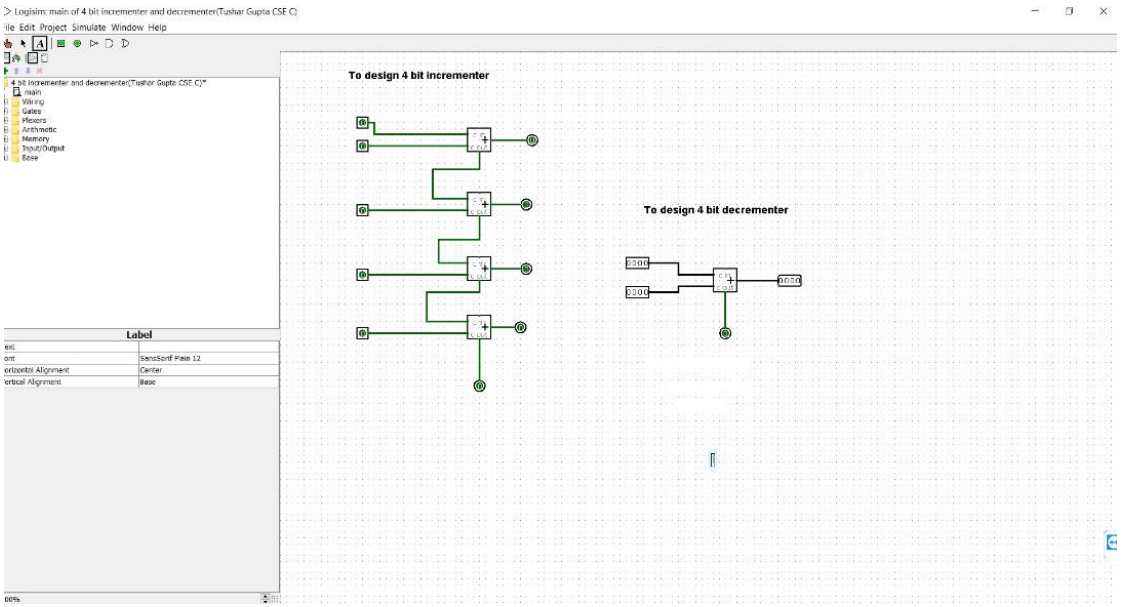
Fig. 4 bit Decrementer Circuit

Observed Values:

Following set of values were obtained in observation.

1. 0011 => 0010
2. 1010 => 1001
3. 1101 => 1100
4. 0010 => 0001
5. 0000 =>1111

Implementation:-



Result: All outputs are verified.