Ajay Kumar Garg Engineering College, Ghaziabad

DEPARTMENT OF INFORMATION TECHNOLOGY

COMPUTER ORGANIZATION LAB (BCS 352)

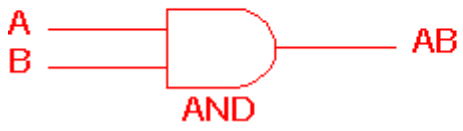
Submitted By

Dr. Shivani Agarwal
Mr. Pankaj Singh
Ms. Tanu Gupta
Ms. Rinki Tyagi

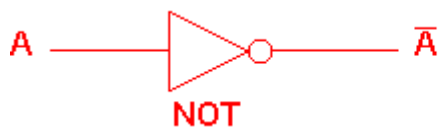
LIST OF EXPERIMENTS

Sr. No.	Title of experiment	Date of Conduct	Date of Submission	Faculty Signature
1.	Study and verify the outputs of the logic gates (AND, OR, NOT, NAND, NOR, Ex-OR, andEx-NOR).			
2.	Implement Half Adder, Full Adder, Half Subtractor and Full Subtractor.			
3.	Implement 3-bit parallel Binary Adder/Subtractor.			
4.	Implement 3-bit carry look-ahead adder.			
5	Implement 4-bit Binary -to -Gray, Gray -to -Binary code converter.			
6.	Implement a 2x2 Binary Multiplier.			
7.	Implement (4 to 2) line and (8 to 3) line Encoders.			
8.	Implement (2 to 4) line and (3 to 8) line Decoders.			
9.	Implement 4x1 and 8x1 Multiplexers.			
10.	Implement 1x4 and 1x8 De-multiplexers.			
11.	Verify the characteristic/state tables of SR and D FLIP-FLOPS using NAND gates.			
12.	Design a 8-bit Arithmetic Logic Unit.			

13.	Design the data path of a computer from its register transfer language description			
14.	Design the control unit of a computer using either hardwiring or microprogramming based on its register transfer language description.			
15.	Implement a simple instruction set computer with a control unit and a data path			

Experiment Number < 1 >		
1	**Aim / Objective / problem statement**	**Study and verify the outputs of the logic gates (AND, OR, NOT, NAND, NOR, Ex-OR, and Ex-NOR).**
	sample input	Input for all gates: A=0, B=1
	expected output	Suppose the output of all gates dented by symbol X, then Output of NOT Gate: X=1 {for input A} Output of AND Gate: X=0 Output of OR Gate: X=1 Output of NAND Gate: X=1 Output of NOR Gate: X=0 Output of XOR Gate: X=1 Output of XNOR Gate: X=0
2	**Theory**	**AND Gate**: A multi-input circuit in which the output is 1 only if all inputs are 1. A dot (.) is used to show the AND operation i.e. (A.B). The symbolic representation and truth table of the 2-input AND gate are given below:  2 Input AND gate A B A.B 0 0 0 0 1 0 1 0 0 1 1 1 **OR Gate**: A multi-input circuit in which the output is 1 when any input is 1. A plus (+) is used to show the OR operation i.e. (A+B). The symbolic representation and truth table of 2-input OR gate are given below:  2 Input OR gate A B A+B 0 0 0 0 1 1 1 0 1 1 1 1

NOT Gate:
The output is 0 when the input is 1, and the output is 1 when the input is 0. It is also known as an inverter. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The symbolic representation and truth table of an inverter are given below:



NOT gate	
A	$\overline{A}$
0	1
1	0

NAND Gate:
AND followed by an INVERTER. The output of a NAND gate is high if any of the inputs are low. It is also known as universal gate. The symbolic representation and truth table of 2-input NAND gate are given below:

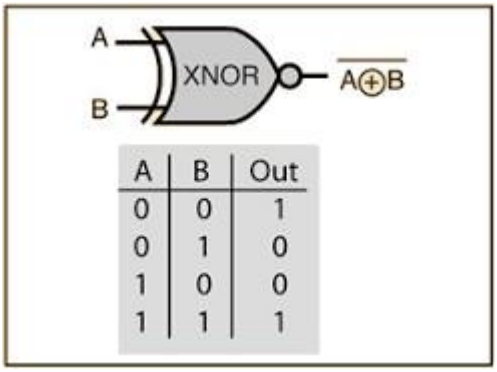
2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate:
OR followed by inverter. The outputs of a NOR gate is low if any of the inputs are high. It is also known as universal gate. The symbolic representation and truth table of 2-input NOR gate are given below:

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Ex-OR Gate:
The output of the Exclusive-OR gate is 0 when its two inputs are the same and its output is 1 when its two inputs are different. **X-OR gate is a digital logic gate that gives a true (1 or HIGH)output when the number of true inputs is odd.** An encircled plus sign ( ) is used to show the XOR operation. The symbolic representation and truth table of 2-input XOR gate are given below:

2 Input EXOR gate		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

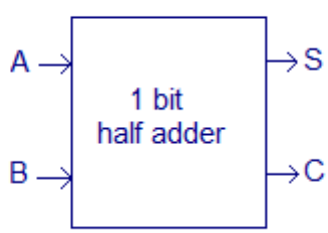
		Ex-NOR Gate: The output of the Exclusive-NOR gate, is 1 when its two inputs are the same and its output is 0 when its two inputs are different. **X-NOR gate is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is even.** An encircled dot sign is used to show the Ex-NOR operation. The symbolic representation and truth table of 2-input XNOR gate are given below:  A | B | Out 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
3	**Procedure**	**Steps:** 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	1. How many AND gates are required to realize Y = CD + EF + G?
		2. How many NAND gates are required to design a XOR gate? Draw XOR gate using NAND gate.
		3. Which gates are the universal gates? Why these gates are called universal gates?

Experiment Number < 2>

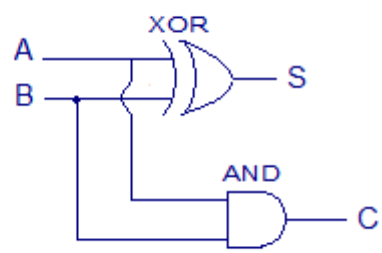
1	**Aim / Objective / problem statement**	**Implement Half Adder, Full Adder, Half Subtractor and Full Subtractor.**
	sample input	Inputs for Half Adder and Half Subtractor A=0, B=1 Inputs for Full Adder and Full Subtractor A=0, B=1, C=1
	expected output	Output of Half Adder Sum S= 1, Carry C=0 Output of Half Subtractor Diff D= 1, Borrow Bo =1 Output of Full Adder Sum S= 0, Carry C=1 Output of Full Subtractor Diff D= 0, Borrow Bo =1
2	**Theory**	**Half Adder:** The Half Adder is a combinational digital circuit that adds two binary digits and produces two outputs as Sum(S) and Carry (C). The circuit, Block diagram and truth table of Half Adder are shown below:

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



Schematic

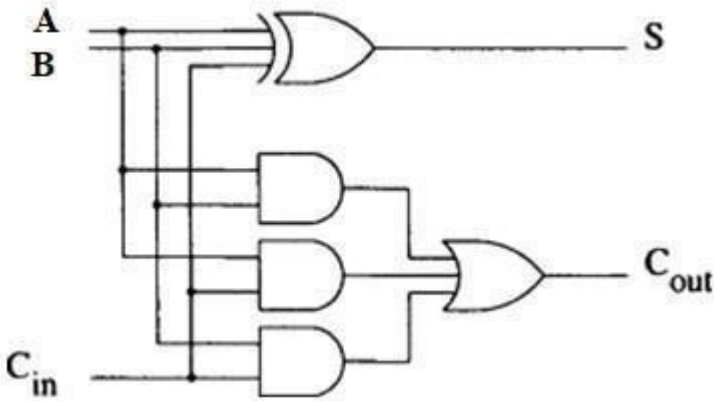


Realization

Full Adder:

The Full Adder is a combinational digital circuit that adds three binary digits and produces two outputs as Sum (S) and Carry (C_{out}). The first two inputs are A and B and the third input is an

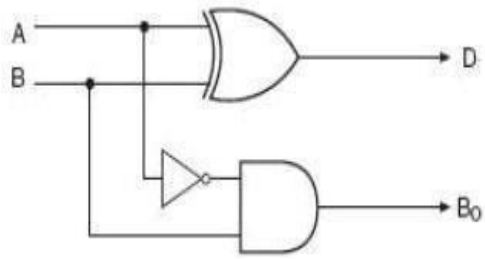
input carry designated as C_{in}. The circuit diagram and Truth Table of a full adder are given below:



Input(A)	Input(B)	Input(Cin)	Sum(S)	Carry(Cout)
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

Half Subtractor:

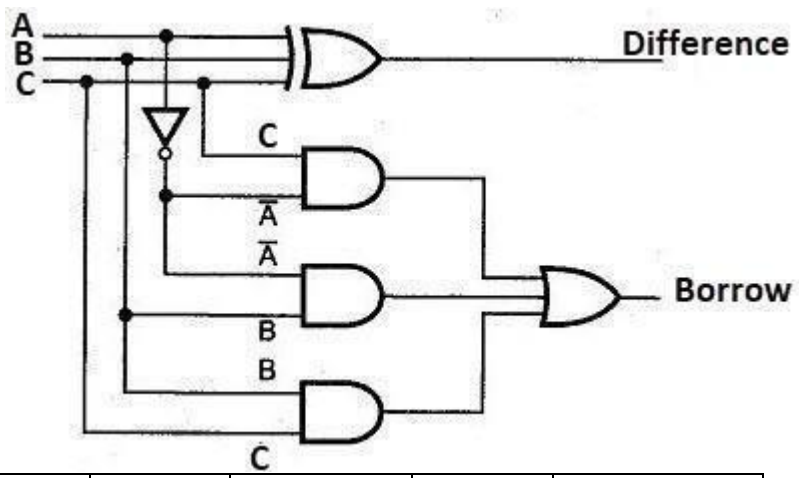
The Half-Subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, A (minuend) and B (subtrahend) and two outputs D (difference) and Bo (borrow). The circuit diagram and Truth Table of a half subtractor are given below:



Inputs		Outputs	
Input(A)	Input(B)	Diff (D)	Borrow (Bo)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Full Subtractor:

A combinational circuit which performs the subtraction of three input bits is called full subtractor. The three input bits include two significant bits and a previous borrow bit. A full subtractor circuit can be implemented with two half subtractors and one OR gate. The circuit diagram and Truth Table of a full subtractor are given below:



Input(A)	Input(B)	Input(C)	Diffence D	Borrow(Bout)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

3 Procedure

Steps:

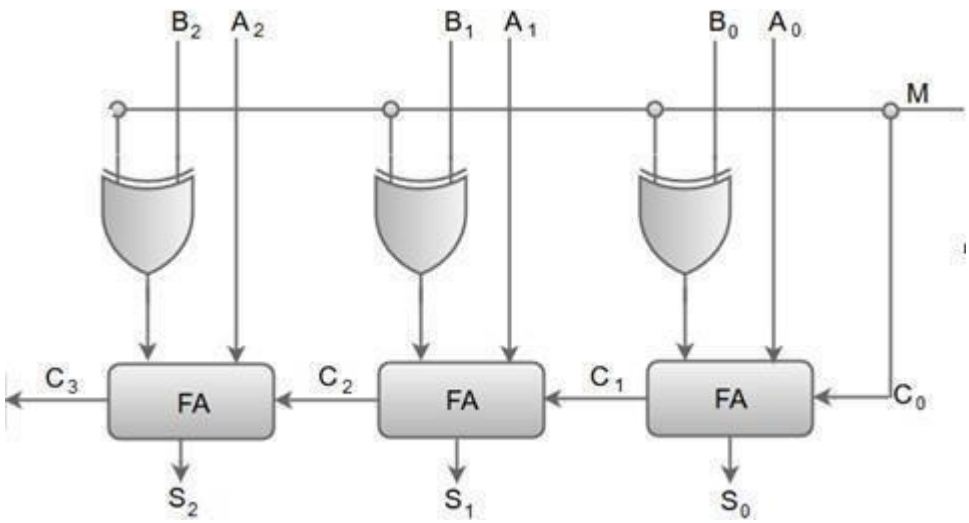
1. Start the simulator by using https://circuitverse.org/simulator link
2. Design the circuit as shown in the theory section.
3. Set the inputs as shown in the Input section
4. And verify the output as mentioned in the output section
5. Take the screenshot and create a PDF document
6. Upload this PDF document on the given assignment link onto the moodle.

4 Viva questions

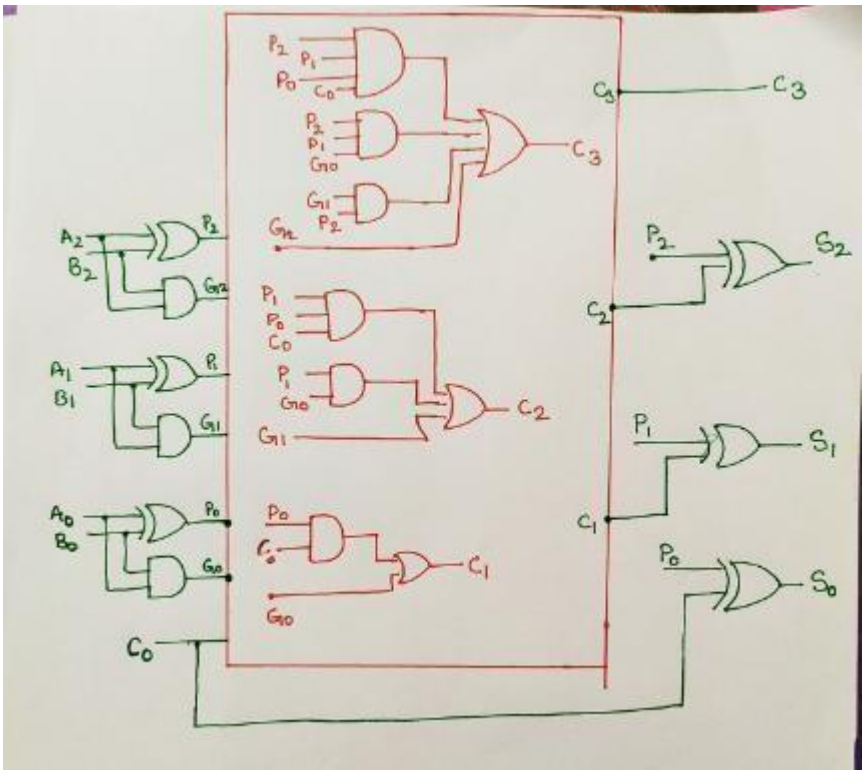
Draw the full adder using half adders.

Draw the full subtractor using half subtractor.

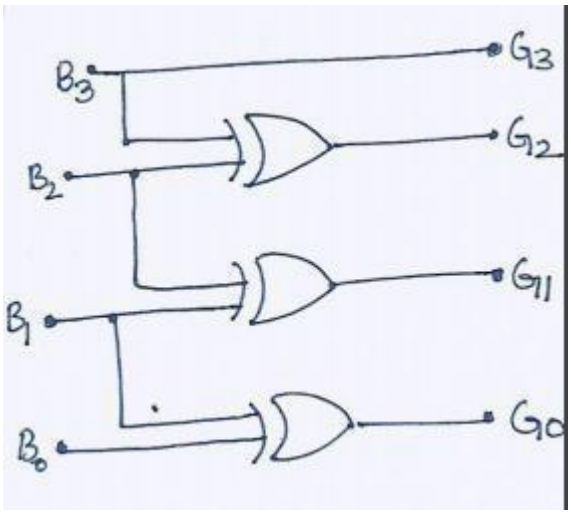
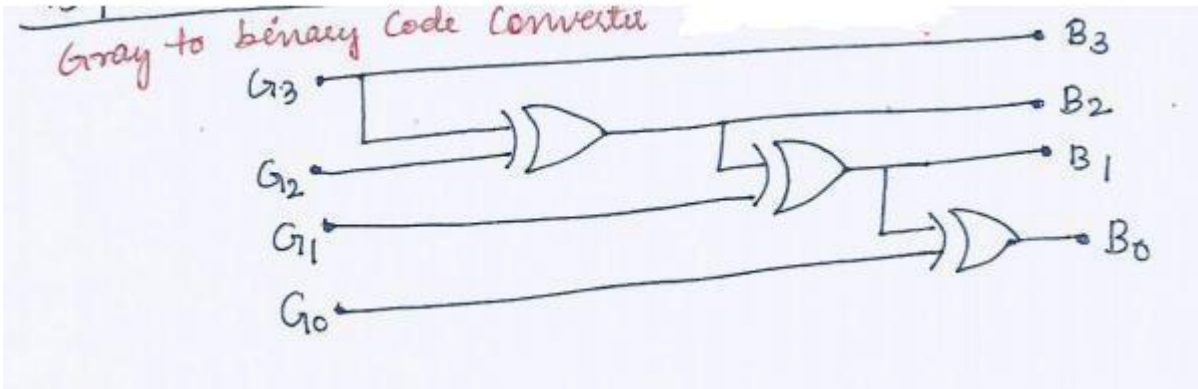
What is a major limitation of a half adder?

Experiment Number < 3 >		
1	**Aim / Objective / problem statement**	**Implement 3-bit parallel Binary Adder/Subtractor.**
	sample input	$A_2A_1A_0$= 110 $B_2B_1B_0$= 101
	expected output	When M=0, $S_2S_1S_0$ = 011, C_3= 1 When M=1, $S_2S_1S_0$ = 001 C_3 = 1
2	**Theory**	**Binary Adder/Subtractor:** A 3-bit Binary Adder-Subtractor is a digital circuit for both addition and subtraction of two 3-bit binary numbers in one circuit itself. The operation being performed depends upon the binary value the control signal M. The circuit consists of 3 full adders. There is a control line M that holds a binary value of either 0 or 1. M=0 determines that the operation is addition. M=1 determines that the operation is subtraction. XOR gate complements its input if the other input is 1. The XOR gate properties given below. $X \oplus 1 = X'$ $X \oplus 0 = X$ The 3-bit Binary Adder/Subtractor is shown below: 
3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.

4	**Viva questions**	How many full adders and XOR gates are required to make 4-bit Binary Adder/Subtractor?
		What is the disadvantage of a ripple adder?
		Draw the 4-bit parallel Binary Adder/Subtractor

Experiment Number < 4 >		
1	**Aim / Objective / problem statement**	**Implement 3-bit carry look-ahead adder.**
	sample input	$A_2A_1A_0$= 110 $B_2B_1B_0$= 101
	expected output	$S_2S_1S_0$ = 011, C_3= 1
2	**Theory**	**Carry look ahead Adder:** Carry look-ahead adder utilizes the logic gates to look at the lower order bits of augmend and addend to see if a higher order carry is to be generated or not. Carry look-ahead uses the two concepts of carry propagate and carry generate functions. This adder uses the following equations for i^{th} stage: Carry propagate $P_i=A_i \oplus B_i$ Carry generate $Gi=A_i.B_i$ Sum $S_i= P_i \oplus C_i$ Carry $C_{i+1}=Gi+Pi.Ci$ The circuit diagram of 3-bit Carry Look-Ahead Adder is shown in the following figure: 
3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator lnk. 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	1. What property distinguishes a look-ahead-carry adder from ripple adder?

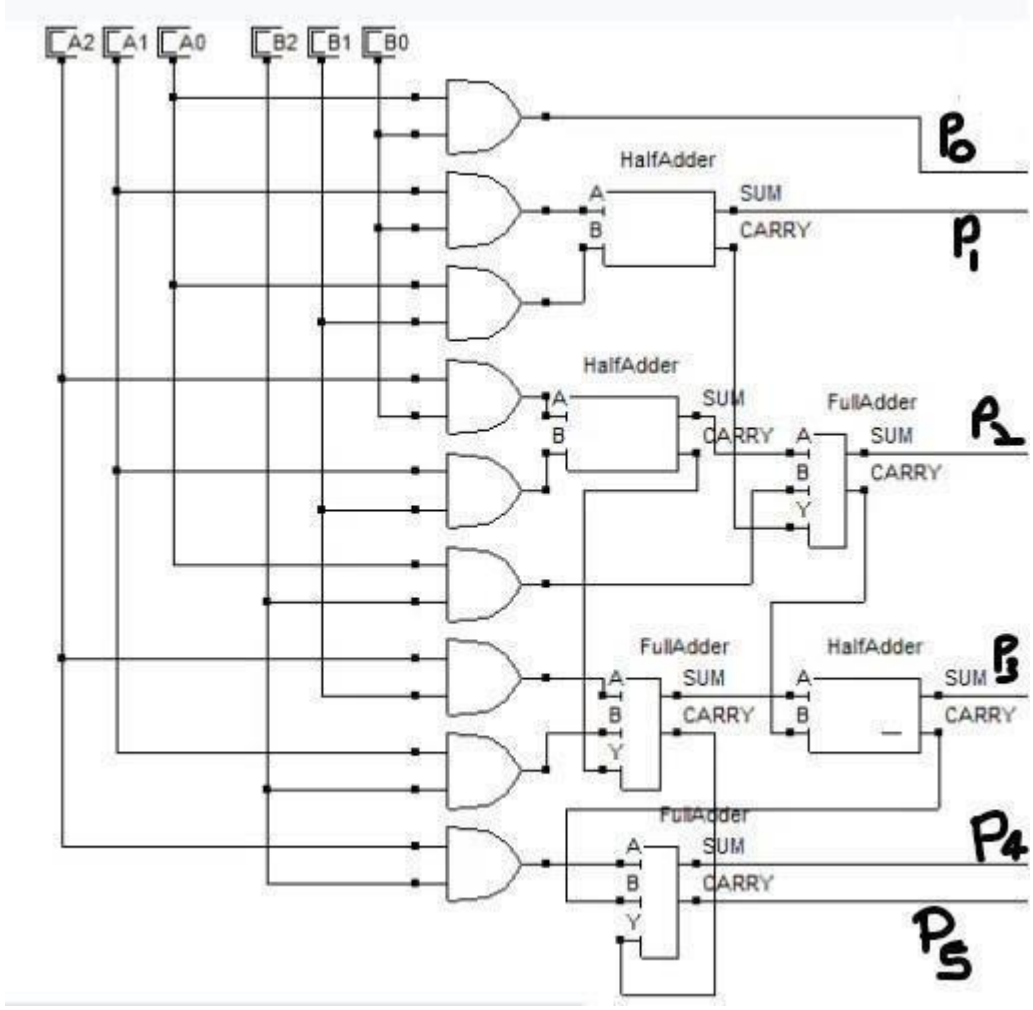
		2. What are the two functions Carry look-ahead logic uses to create carry look-ahead generator circuit?
		3. What is carry propagation delay?

Experiment Number < 5 >		
1	**Aim / Objective / problem statement**	**Implement 4-bit Binary -to -Gray, Gray -to -Binary code converter.**
	sample input	Input: For circuit 1: Binary code ($B_3B_2B_1B_0$) = 0101 Input: For circuit 2: Gray code ($G_3G_2G_1G_0$)= 0111
	expected output	Output: For circuit 1: Gray code ($G_3G_2G_1G_0$)= 0111 Output: For circuit 2: Binary code ($B_3B_2B_1B_0$) = 0101
2	**Theory**	**Gray Code:** Gray code is a binary numeral system where two successive values differ in only one bit. **Binary to Gray**: Binary to gray code converter is a combinational circuit that converts a binary number into a gray code. The circuit diagram of Binary to Gray Converter is shown in the following figure:  **Gray to Binary:** Gray to binary code converter is a combinational circuit that converts a gray code into binary code. The circuit diagram of Gray to binary code converter is shown in the following figure:  Truth table for conversion:

S. No	Binary code ($B_3B_2B_1B_0$)	Gray code ($G_3G_2G_1G_0$)
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	What is the property of gray code?
		What are the applications of gray code?
		What are the weighted and un-weighted codes? Is gray code weighted?

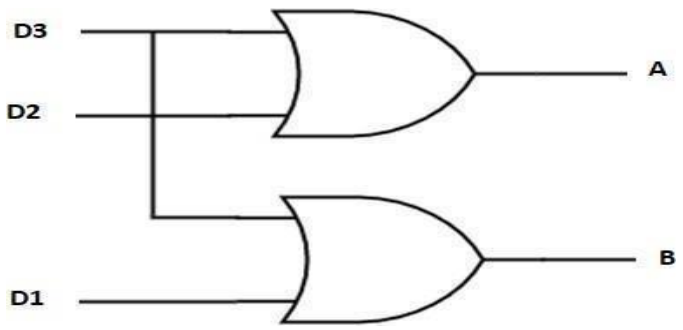
Experiment Number < 6>		
1	**Aim / Objective / problem statement**	**Implement a 2x2 Binary Multiplier.**
	sample input	Inputs : A_1A_0= 110 B_1B_0= 111
	expected output	$P_5P_4P_3P_2P_1P_0$ = 101010
2	**Theory**	**Binary multiplier:** A binary multiplier is a combinational digital circuit to multiply two binary numbers. The binary multiplication of two 2-bit numbers A (A_1 A_0) and B (B_1 B_0) can be performed as given in the following image A2 A1 A0 B2 B1 B0 A2B0 A1B0 A0B0 A2B1 A1B1 A0B1 X A2B2 A1B2 A0B2 X X P_5 P_4 P_3 P_2 P_1 P_0 The circuit diagram of 3x3 bit binary multiplier has been shown in the following image.

		
3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	Draw 4x4-Binary Multiplier.
		How many full adders and AND gates are will be required to draw 4x3 binary multiplier?
		How many bits will in the result of multiplication of M-bit and N-bit numbers?

Experiment Number < 7 >		
1	**Aim / Objective / problem statement**	**Implement (4 to 2) line and (8 to 3) line Encoders.**
	sample input	Input for (4 to 2) line Encoder: $D_0D_1D_2D_3$ =0001 Input for (8 to 3) line Encoder: $D_0D_1D_2D_3D_4D_5D_6D_7$ = 00010000
	expected output	Output for (4 to 2) line Encoder: AB=11 Output for (8 to 3) line Encoder: ABC=011
2	**Theory**	

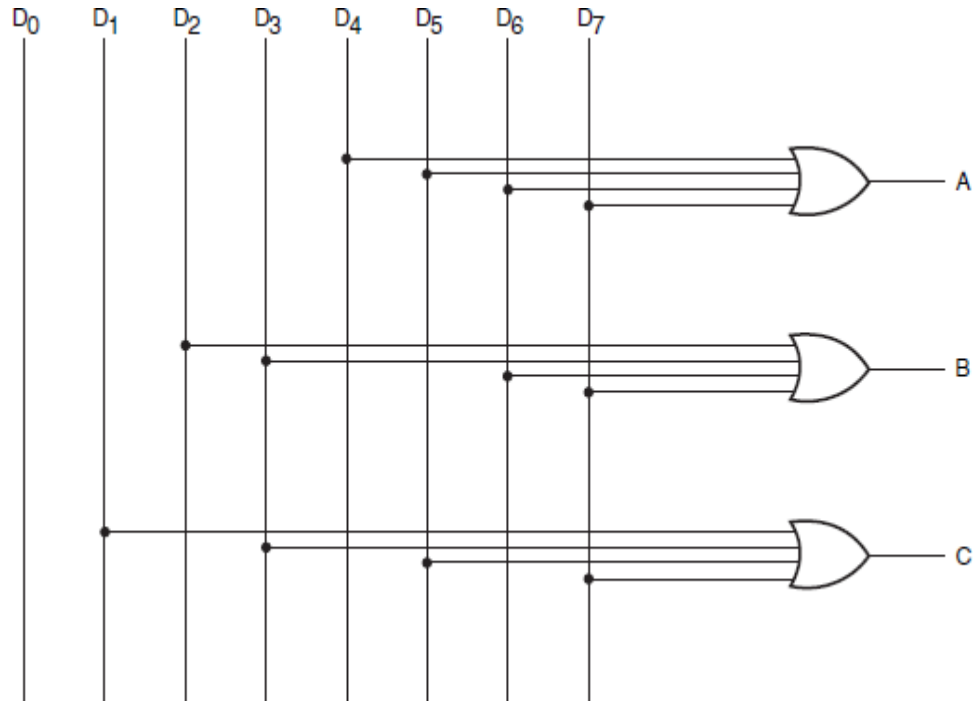
Encoder:
An Encoder is a combinational circuit that has maximum of 2^n input lines and 'n' output lines; hence it encodes the information from 2^n inputs into an n-bit code.
It will produce a binary code equivalent to the input, which is active High.

(4 to 2) line Encoder:
The 4 to 2 Encoder consists of four inputs D_0 D_1 D_2 D_3, and two outputs A and B. (4 to 2) Encoder encodes the information from 4 inputs into a 2-bit code. The circuit diagram and truth table of (4 to 2) line Encoder are shown below:



Inputs				Outputs	
D0	D1	D2	D3	A	B
1	0	0	0	0	0
0	1	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1

(8 to 3) line Encoder:
The 8 to 3 Encoder or octal to Binary encoder consists of **8 inputs**: D7 to D0 and 3 outputs A, B and C. Each input line corresponds to each octal digit and three outputs generate corresponding binary code. The circuit diagram and truth table of (8 to 3) line Encoder are shown below:



Truth table for (8 to 3) line Encoder:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	What is Priority Encoder?
		What is decimal to BCD encoder?
		What are the applications of an encoder circuit?

Experiment Number < 8 >

1	**Aim / Objective / problem statement**	**Implement (2 to 4) line and (3 to 8) line Decoders.**
	sample input	Input for (2 to 4) line decoder: B_1B_0=11 Input for (3 to 8) line decoder: ABC=011
	expected output	Output for (2 to 4) line Encoder: $D_0D_1D_2D_3$ =0001 Output for (3 to 8) line Encoder: $D_0D_1D_2D_3D_4D_5D_6D_7$ = 00010000

2 Theory

Decoder:

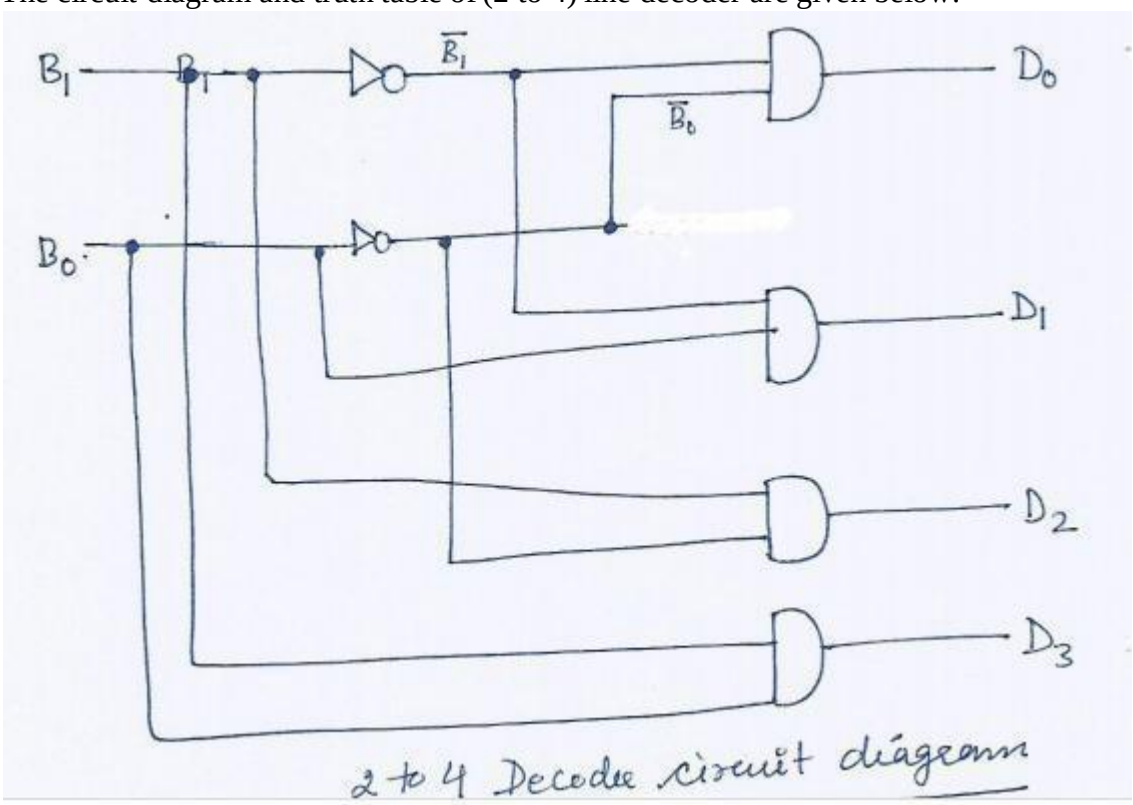
In Digital Electronics, discrete quantities of information are represented by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information.

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

(2 to 4) Decoder:

The (2 to 4) decoder consists of **two** inputs $B_1$and B_0, and four outputs D_0 D_1 D_2 D_3. (4 to 2) decoder decodes the information from 2 inputs into a 4-bit code.

The circuit diagram and truth table of (2 to 4) line decoder are given below:



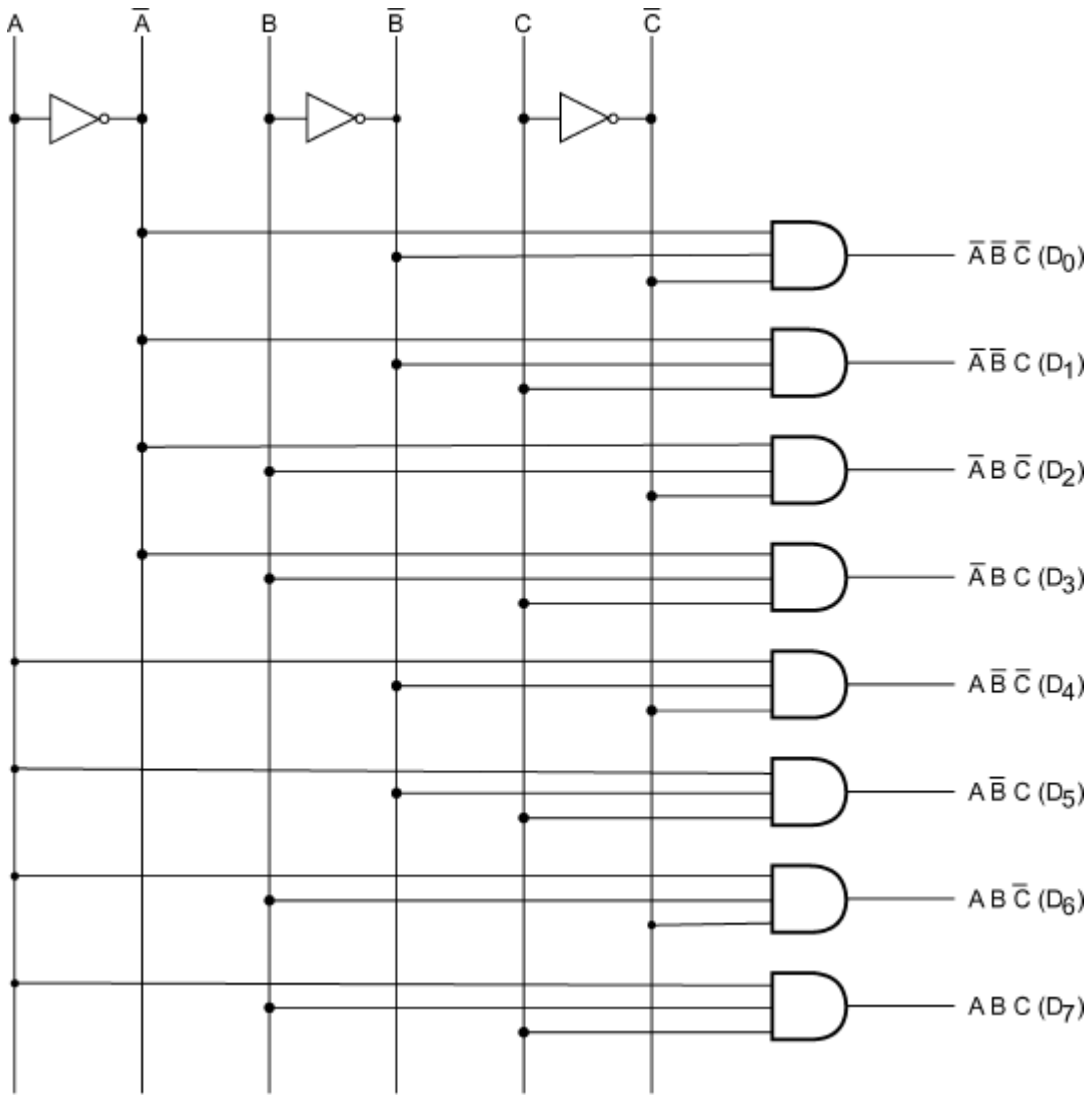
Truth Table for (2 to 4) line decoder:

Inputs		Outputs			
B1	B0	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	1
1	0	0	0	1	0
1	1	0	0	1	1

(3 to 8) line DECODER:

The (3 to 8) decoder consists of **three** inputs A, B, and C, and eight outputs D_0 D_1 D_2 D_3 $D_4D_5D_6D_7$. (3 to 8) decoder decodes the information from 2 inputs into a 4-bit code.

The circuit diagram and truth table of (2 to 4) line decoder are given below:



Truth Table of (3 to 8) line DECODER:

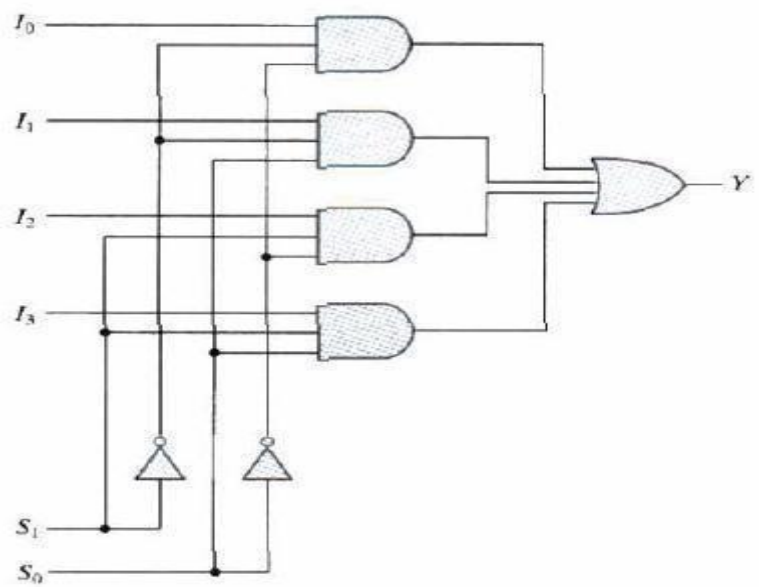
INPUTS			OUTPUTS							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	Draw the 3x8 decoder using 2x4 decoders.
		What is a binary decoder?
		What are the applications of a decoder?

Experiment Number < 9 >		
1	**Aim / Objective / problem statement**	**Implement 4x1 and 8x1 Multiplexers.**
	sample input	Inputs For 4x1Mux: S1S0 = 00, Inputs For 8x1Mux: S2S1S0 = 000
	expected output	When I_0=0, then Y=0 When I_0=1, then Y=1 {in both the circuits}
2	**Theory**	**Multiplexer:** Multiplexers are circuits that can select one of many inputs. **4x1 Multiplexer:** 4:1-Mux has 4 inputs with only 1 output. It has 2 data selector inputs namely S0 and S1, at which the control bits are applied. I_0, I_1, I_2, and I_3 represent the inputs bits. Only one of these will be transmitted to the output. But which one ofthe inputs will be transmitted will depend on the values of the controls. The selection table is given below:

S.No	SELECTION INPUT		OUTPUT
	S1	S0	Y
1.	0	0	I_0
2.	0	1	I_1
3.	1	0	I_2
4.	1	1	I_3

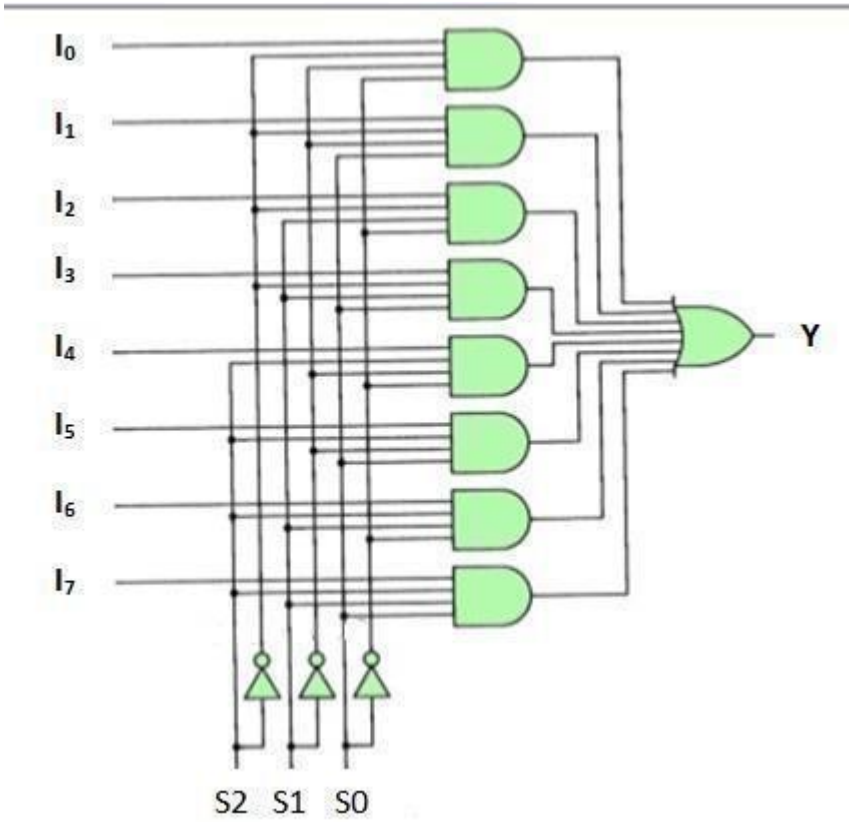
The following circuit diagram shows 4x1 multiplexer:



8x1 Multiplexer:

It has 8 inputs with only 1 output Y. It has 3 data selector inputs namely S0, S1, and S_2 at which the control bits are applied.

I_0, I_1, I_2, I_3, I_4, I_5, I_6, and I_7 represent the inputs bits. Only one of these will be transmitted to the output Y. But which one of the inputs will be transmitted will depend on the values of the controls. The circuit diagram and the selectiontable (truth table) of 8x1 Multiplexer are given below:



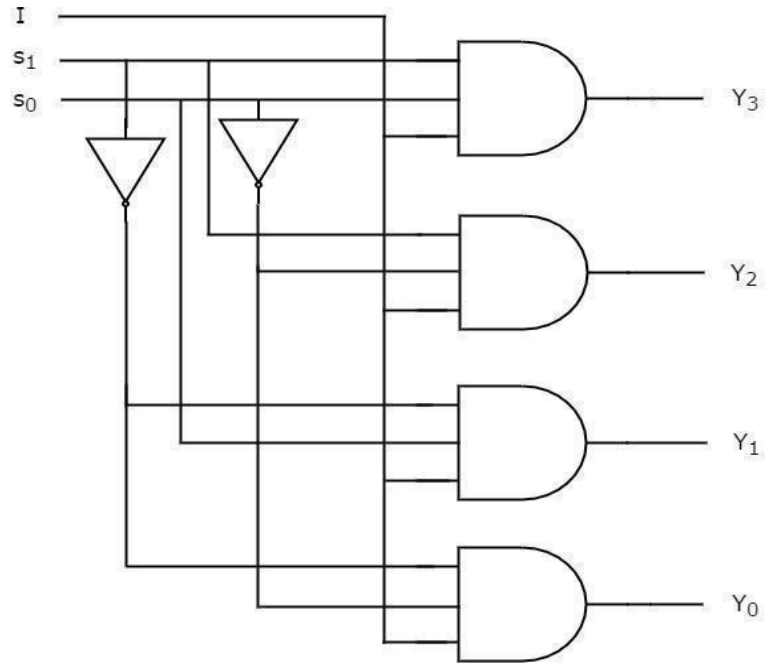
SELECTION INPUT			OUTPUT
S2	S1	S0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

3 **Procedure**

Steps:

1. Start the simulator by using https://circuitverse.org/simulator link
2. Design the circuit as shown in the theory section.
3. Set the inputs as shown in the Input section
4. And verify the output as mentioned in the output section

		5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	1. Draw the block diagram of 8 input multiplexer using 2-input multiplexer, 16 input multiplexer using 2-input multiplexer.
		2. What are the applications of a multiplexer?
		3. Implement the following function using 4x1 Mux. F(A, B, C) =∑ (2, 4,7)

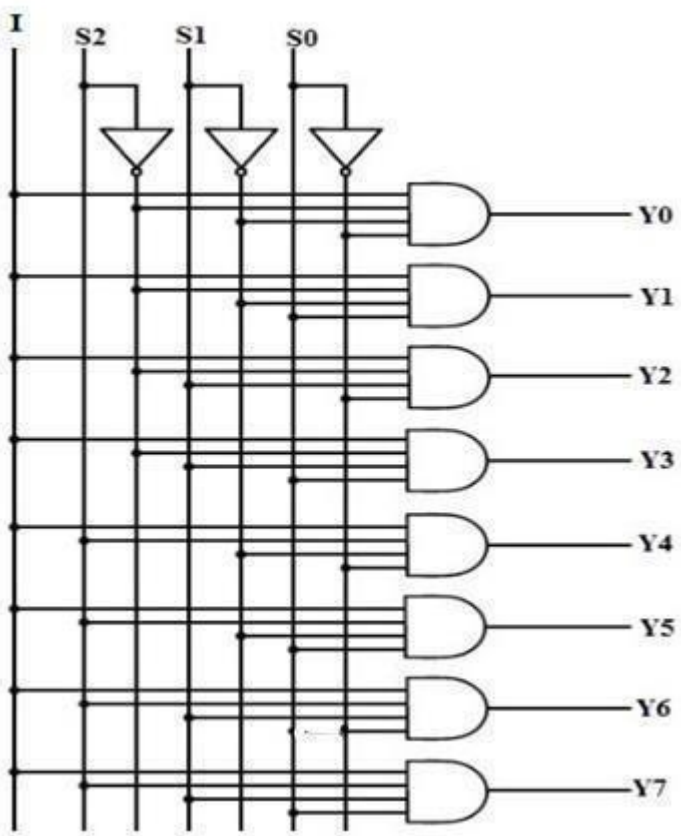
Experiment Number < 10>		
1	**Aim / Objective / problem statement**	**Implement 1x4 and 1x8 De-multiplexers.**
	sample input	Inputs For 1x4 De-Mux: S1S0 = 00, Inputs For 1x8 De-Mux: S2S1S0 = 000
	expected output	When I=0, then Y_0=0 When I=1, ten Y_0=1
2	**Theory**	**Demultiplexer**: A demultiplexer is a combinational logic circuit with an input line, 2^n output lines and n select lines. It routes the information present on the input line to any of the output lines. The output line that gets the information present on the input line is decided by the bit status of the selection lines. **1x4 Demultiplexer:** It has only data input I with 4 outputs namely Y_0, Y_1, Y_2, and Y3. It has two data selector inputs namely S0 and S1, at which control bits are applied. The data bit is transmitted to any of the data bit Y_0, Y_1, Y_2, and Y3 of the output lines. Which particular output line will be chosen will depend on the value of S1 and S0 the control input. The circuit diagram and the selection table (truth table) of 1x4 Demultiplexer are given below: 

Selection table:

INPUT			OUTPUT			
S1	S2	I	Y0	Y1	Y2	Y3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

1x8 Demultiplexer:

It has only data input I with 8 outputs namely Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7. It has three data selector inputs namely S0, S1 and S2, at which control bitsare applied. The data bit is transmitted to any of the Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 output lines. Which particular output line will be chosen will depend onthe value of S0, S1, and S2 the control input. The circuit diagram and the selection table (truth table) of 1x8 Demultiplexer are given below:



Selection Table

Selection Input			Output Selected
S2	S1	S0	Y
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

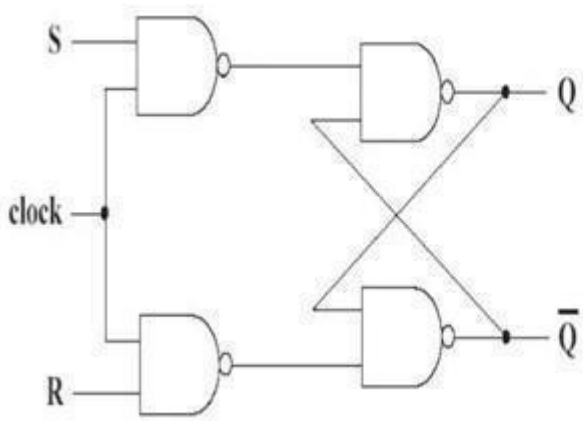
3 **Procedure**

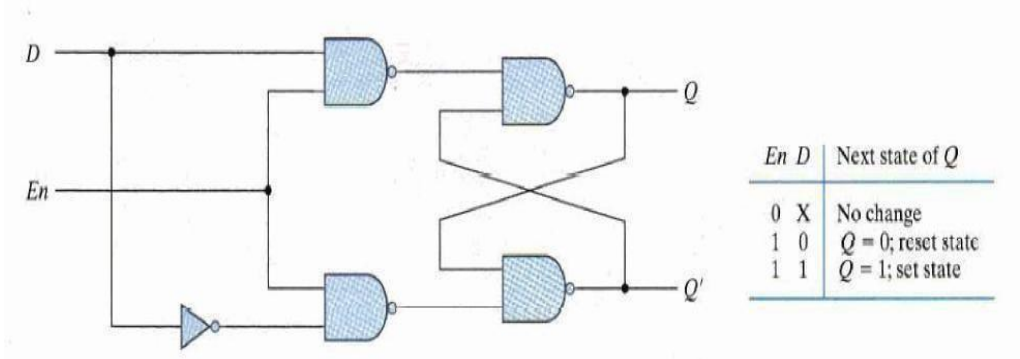
Steps:

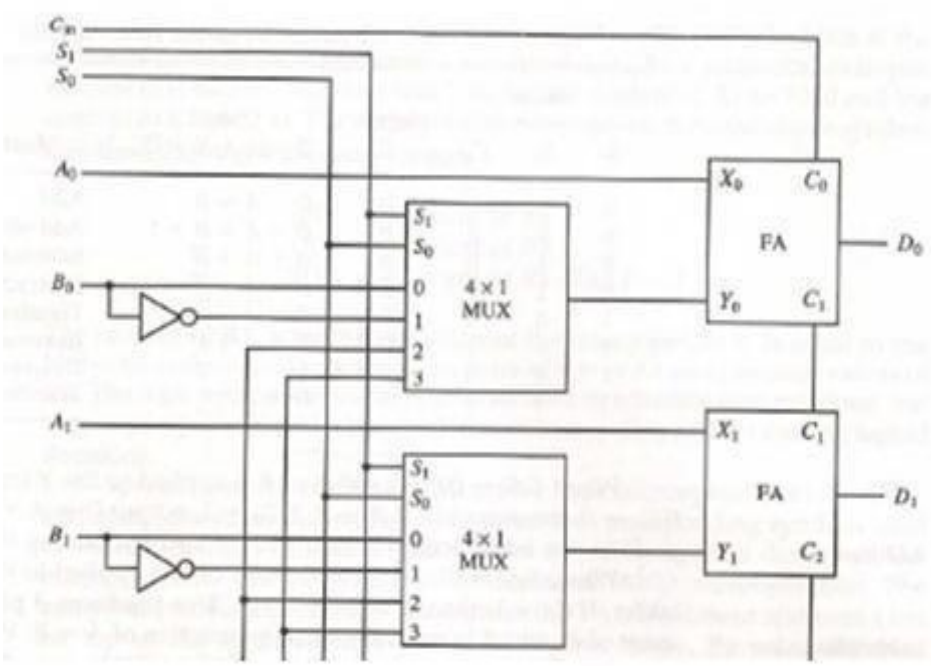
1. Start the simulator by using https://circuitverse.org/simulator link
2. Design the circuit as shown in the theory section.
3. Set the inputs as shown in the Input section
4. And verify the output as mentioned in the output section
5. Take the screenshot and create a PDF document
6. Upload this PDF document on the given assignment link onto the moodle.

4 **Viva questions**

1. Draw a 1x8 demultiplexer using 1x4-demultiplexer.
2. Why a demultiplexer is called a data distributor?
3. How many selection lines are required in 1x64 demultiplexer.

Experiment Number < 11 >		
1	**Aim / Objective / problem statement**	**Verify the characteristic/state tables of SR and D FLIP-FLOPS using NAND gates.**
	sample input	Input for SR Flip Flop: S=1, R=0 Input for D Flip Flop: D=1, En=1,
	expected output	Q=1 { output for both the circuits}
2	**Theory**	**Flip Flop:** Flip Flop is a sequential digital circuit that stores one bit. Types of the flip flops are : • SR Flip Flop • D Flip Flop • JK Flip Flop • T Flip Flop • Master-Slave JK Flip Flop **SR Flip Flop** SR flip flop is a circuit with two cross coupled NAND gates or NOR gates, and two inputs labeled S for set and R for Reset. The flip flop has two useful states. When output Q=1 and Q'=0 the flip flop is said to be in set state and when output Q=0 and Q'=1, it is in the reset state. Outputs Q and Q' are normally complement of each other. However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 occurs. If both the inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state. In practical applications, setting both inputs to 1 is forbidden. The logic diagram of SR flip flop using NAND gate and its characteristic table are given below:  Operation Mode | S | R | Q_{n+1} No change | 0 | 0 | Q_n SET | 1 | 0 | 1 RESET | 0 | 1 | 0 Forbidden | 1 | 1 | — **D Flip Flop:** The D input goes directly to the S input and its complement is applied to R input. As long as the enable is at 0, the cross-coupled SR latch has both inputs at the level 1 and the circuit cannot change state regardless of the value of D. The D input is sampled when En =1. If D =1, the Q output goes to 1, placing the circuit in the set state. If D=0, output Q goes to 0, placing the circuit in the reset state. The logic diagram of D flip flop using NAND gate and its characteristic table are given below:

		
3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and create a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	1. What is difference between the flip flop and latch?
		2. Draw the SR flip flop and D flip flop using NOR gates
		3. What is a sequential digital circuit?

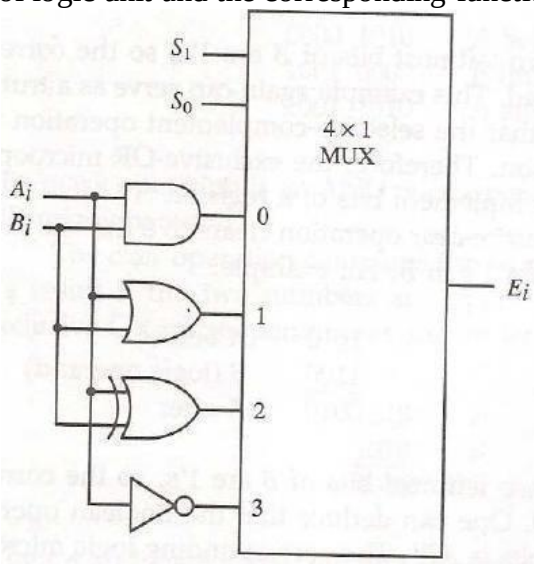
Experiment Number < 12 >		
1	**Aim / Objective / problem statement**	**Design a 8-bit Arithmetic Logic Unit.**
	sample input	S1S0=00 A1A0=11 B1B0= 01
	expected output	**Output of Arithmetic Unit:** When $C_0 = 0$ D1D0=00, C2=1 When $C_0 = 1$ D1D0= 01, C2=1 **Output of Logic Unit** : $E0 = A_0 . B_0 = 1$
2	**Theory**	**Arithmetic Unit:** The circuit diagram of arithmetic unit given below can perform the arithmetic microoperations listed in the given table. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations. The diagram of a 4-bit arithmetic circuit shown below. It has two full adders and two multiplexers for selecting different operations. There are two 2-bit inputs A and B and 2-bit output D. the input A directly goes to X inputs of the binary adders. The multiplexer takes input B, Complement of B, 0 and 1 as input. The selection of inputs is controlled by S1 and S0 selection lines. The output of the binary adder is calculated by D = A + Y+ Cin. The possible generated microperations using this equation for different combinations of inputs are given in the arithmetic function table. 

The arithmetic function table:

Select S_1	Select S_0	Select C_{in}	Input Y	Output $D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

One Stage of Logic Unit:

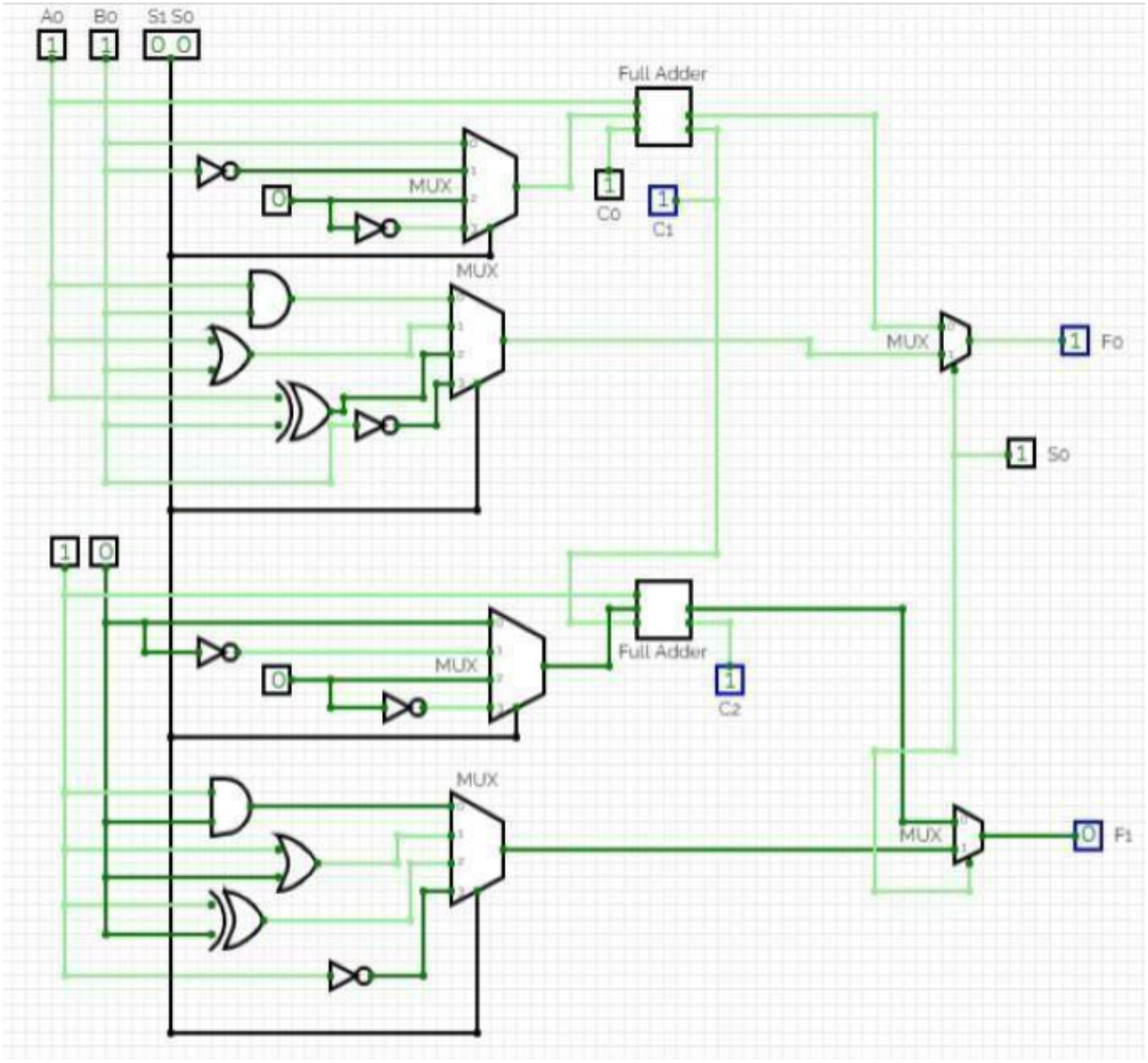
One stage of logic unit given in the following figure generates the four basic logic microoperations. It consists of four gates and a multiplexer. The outputs of the logic gates are applied to the input of multiplexer. The selection inputs S1 and S0 choose one of the data inputs. The diagram of one stage of logic unit and the corresponding function table are given below:



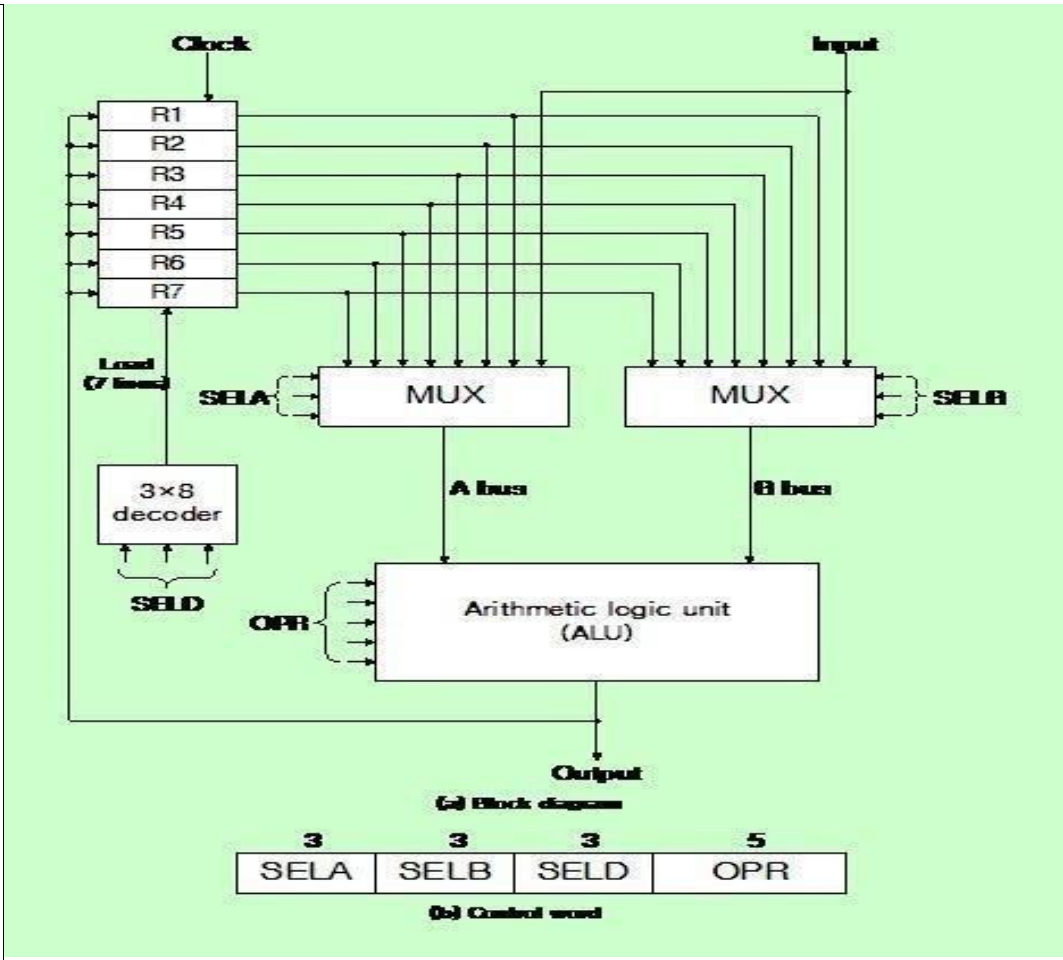
(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

		The Final ALU circuit design: 
3	**Procedure**	Steps: 1. Start the simulator by using https://circuitverse.org/simulator link 2. Design the circuit as shown in the theory section. 3. Set the inputs as shown in the Input section 4. And verify the output as mentioned in the output section 5. Take the screenshot and a PDF document 6. Upload this PDF document on the given assignment link onto the moodle.
4	**Viva questions**	1. Draw the 4-bit arithmetic unit:
		2. Register A holds the 8-bit binary 11011001. Determine the B operand and the logic microoperation to be performed in order to change the value in A to 01101101
		3. What is the purpose of ALU?

Experiment Number < 13>		
1	**Aim / Objective / problem statement**	**Design the data path of a computer from its register transfer language description**
	Components required	3:8 decoder , D flip flop , Clock , Multiplexers , ALU and Connecting wires
2	**Theory**	The number of registers in a processor unit may vary from just one processor register to as many as 64 registers or more. 1. One of the CPU registers is called as an accumulator AC or 'A' register. It is the main operand register of the ALU. 2. The data register (DR) acts as a buffer between the CPU and main memory. It is used as an input operand register with the accumulator. 3. The instruction register (IR) holds the opcode of the current instruction. 4. The address register (AR) holds the address of the memory in which the operand resides. The program counter (PC) holds the address of the next instruction to be fetched for execution. Additional addressable registers can be provided for storing operands and address. This can be viewed as replacing the single accumulator by a set of registers. If the registers are used for many purpose, the resulting computer is said to have general register organization. In the case of processor registers, a registers is selected by the multiplexers that form the buses. When a large number of registers are included in the CPU, it is most efficient to connect them through a common bus system. The registers communicate with each other not only for direct data transfers, but also while performing various micro-operations. Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift micro-operation in the processor. A Bus organization for seven CPU registers:



The output of each register is connected to true multiplexer (mux) to form the two buses A & B.

The selection lines in each multiplexer select one register or the input data for the particular bus. The A and B buses forms the input to a common ALU. The operation selected in the ALU determines the arithmetic or logic micro-operation that is to be performed. The result of the micro-operation is available for output and also goes into the inputs of the registers. The register that receives the information from the output bus is selected by a decoder. The decoder activates one of the register load inputs, thus providing a transfer both between the data in the output bus and the inputs of the selected destination register.

The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the systems.

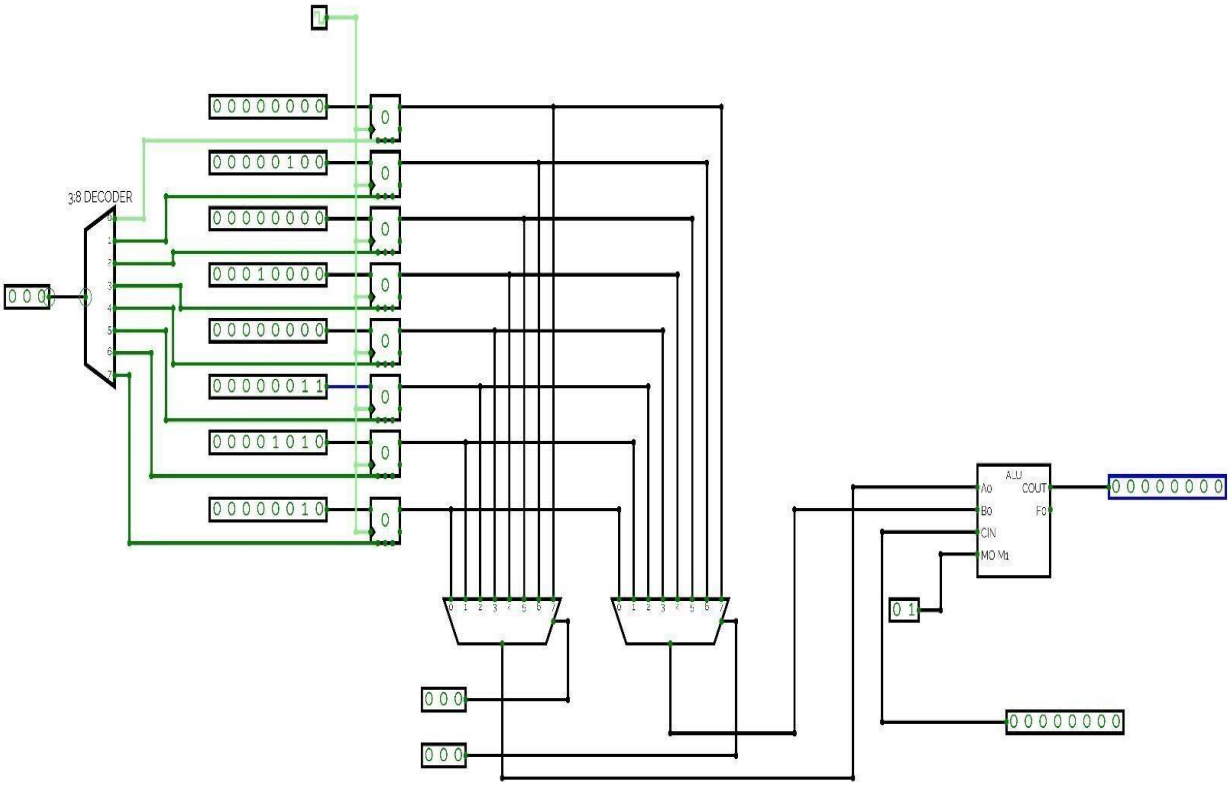
R1 ® R2 + R3

(1) MUX A selection (SEC A): to place the content of R2 into bus A

(2) MUX B selection (sec B): to place the content of R3 into bus B

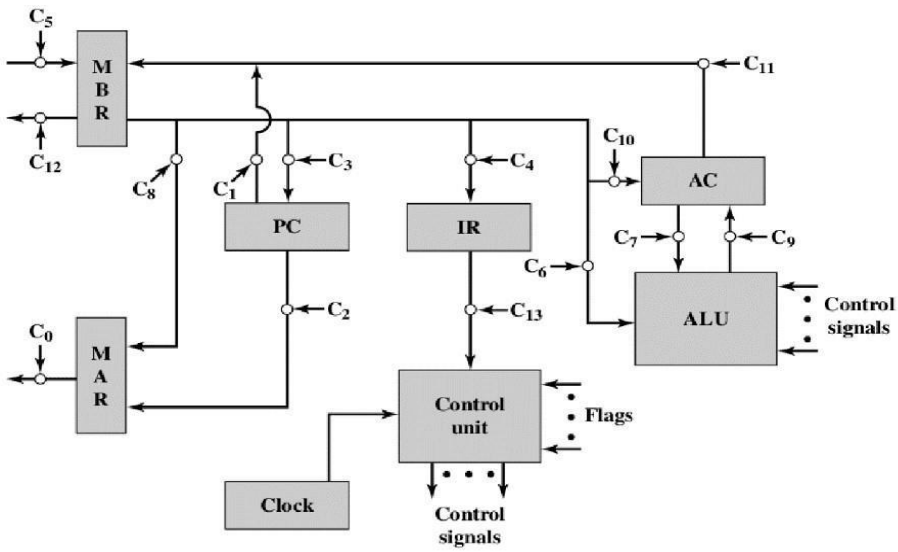
(3) ALU operation selection (OPR): to provide the arithmetic addition (A + B)

(4) Decoder destination selection (SEC D): to transfer the content of the output bus into R1

3	**Implementation and result :**	Circuit diagram 
4	**Precautions :**	1-All ICs must be checked before start. 2- All connections should be tight.
5	**Viva questions**	What is Register Move Language (RTL)?
		For what reason is RTL significant in the equipment plan?
		How does RTL vary from machine code?

Experiment Number < 14>		
1	**Aim / Objective / problem statement**	**Design the control unit of a computer using either hardwiring or microprogramming based on its register transfer language description.**
	Components required	Logisim Simulator
2	**Theory**	**Control unit performs two basic tasks:** 1 Sequencing: Go through a sequence of program specified microoperations; 2 Execution: causes each micro-operation to be performed 3 **The inputs are:** • Clock: • Causes a set of simultaneous micro-operations to be performed; •Instruction register: • Opcode and addressing mode of the current instruction are used to determine which microoperations to perform; • Flags: • To determine the status of the processor and the outcome of previous ALU operations; • Control signals: • signals to the control unit. **The outputs are:** • Control signals within the processor: • Those that cause data to be moved from one register to another; • and those that activate specific ALU functions. • Control signals to control bus: • control signals to memory; • control signals to the I/O modules. **Consider again the fetch cycle, the control unit needs to:** • Transfer the contents of the PC to the MAR. • This is done by activating a control signal: • opening the gates between the bits of the PC and the bits of the MAR. • Read a word from memory into the MBR and increment the PC. **Control unit sends the following control signals simultaneously (1/2):** • Control signal that opens gates: • Allowing the contents of the MAR onto the address bus; • Memory read control signal on the control bus; • Control signal that opens the gates: • allowing the contents of the data bus to be stored in the MBR. **Control unit sends the following control signals simultaneously (2/2):** • Control signals to logic thats • Adds 1 to the contents of the PC; •Stores the result back to the PC. Following this, the control unit: Sends a control signal that opens gates between the MBR and the IR

3	**Implementation and result :**	



Then the following Boolean expression defines C_5:

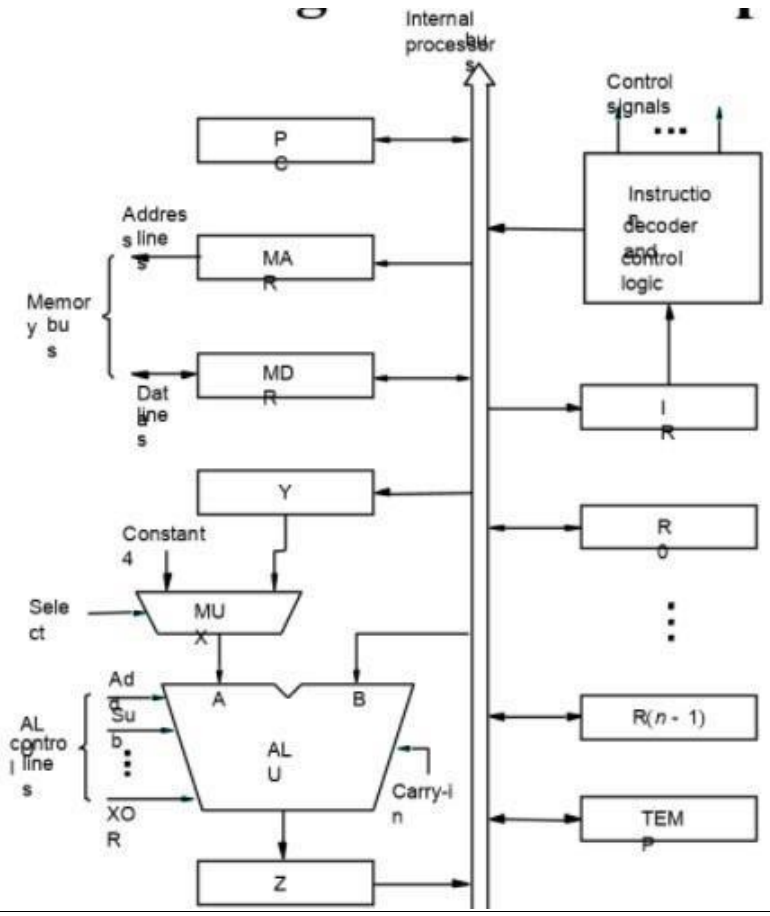
$$C_5 = \overline{P}QT_2 + \overline{P}QT_2$$

	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

4	**Precautions :**	1-All ICs must be checked before start. 2- All connections should be tight.
5	**Viva questions**	Differentiate between Hardwired and micro programmed control unit?
		Describe the usage of clocks and flags in the experiment?
		Describe MAR, IR, PC, MBR registers with their size?

Experiment Number < 15>		
1	**Aim / Objective / problem statement**	**Implement a simple instruction set computer with a control unit and a data path**.
	Components required	Registers , Multiplexers , ALU , Decoder and Connecting wires
2	**Theory**	**Single bus organization**: 1) ALU, control unit and all the registers are connected via a single common bus. 2) Bus is internal to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices. **Data lines of the external memory bus are connected to the internal processor bus via MDR.** 1) Register MDR has two inputs and two outputs. 2) Data may be loaded to (from) MDR from (to) internal processor bus or external memory bus. **Address lines of the external memory bus are connected to the internal processor bus via MAR.** 1) MAR receives input from the internal processor bus. 2) MAR provides output to external memory bus. **Instruction decoder and control logic block, or control unit issues signals to control the operation of all units inside the processor and for interacting with the memory bus.** 1)Control signals depend on the instruction loaded in the Instruction Register (IR) **Outputs from the control logic block are connected to:** 1) Control lines of the memory bus. 2) ALU, to determine which operation is to be performed. 3) Select input of the multiplexer MUX to select between Register Y and constant 4. 4) Control lines of the registers, to select the registers. **Registers Y, Z, and TEMP:** 1) Used by the processor for temporary storage during execution of some instructions. 2) Note that Registers R0 to R(n-1) are used to store data generated by one instruction for later use by another instruction. 3) Data is stored in R0 through R(n-1) after the execution of an instruction.

		Multiplexer MUX selects either the output of register Y or a constant 4, depending upon the control input Select. 1)Constant 4 is used to increment the value of the PC.
3	**Implementation and Results :**	
4	**Precautions :**	1-All ICs must be checked before start. 2- All connections should be tight.
5	**Viva questions**	Differentiate beyween MUX and DEMUX?
		Define Single Bus Organization?
		Define Common Bus Organization?