

EL1005-DIGITAL LOGIC DESIGN

LAB MANUAL



**DEPARTMENT OF ELECTRICAL ENGINEERING,
FAST-NU, LAHORE**

Lab Manual of Digital Logic Design

Created by: Maimoona Akram, Bushra Rashid

Date: January 2010

Last Updated by: Aroosa Umair

Date: July 2023

Approved by: Head of Electrical Engineering Department

Date: July, 2023

Table of Contents

Sr. No.	Description	Page No.
1	List of Equipment	4
2	Experiment No. 1, Introduction to logic trainer and logic probe	5
3	Experiment No. 2, Introduction to basic gates and troubleshooting	10
4	Experiment No. 3, Boolean algebra to obtain cost effective circuit for implementation	15
5	Experiment No. 4, Karnaugh map to obtain cost effective circuit for implementation	20
6	Experiment No. 5, The use of universal gates & mapping	30
7	Experiment No. 6, Design of binary comparator	37
8	Experiment No. 7, Design of decoder, priority encoder, & multiplexer	46
9	Experiment No. 8, Implementation of logic circuit using decoder & MUX ICs	59
10	Experiment No. 9, Design of Adder & Subtractor circuits	66
11	Experiment No. 10, Introduction to Logic Works	74
12	Experiment No. 11, Design and analysis of sequential circuits	90
13	Experiment No. 12, Design of registers	97
14	Appendix A: Lab Evaluation Criteria	102
15	Appendix B: Safety around Electricity	103

List of Equipment

Sr. No.	Description
1	Logic trainer
2	Logic probe
3	74LS08 (2-input AND gate IC)
4	74LS32 (2-input OR gate IC)
5	74LS04 (NOT gate IC)
6	74LS00 (2-input NAND gate IC)
7	74LS02 (2-input NOR gate IC)
8	74LS86 (2-input XOR gate IC)
9	74LS139 (Dual 2x4 Decoder IC)
10	74LS153 (Dual 4x1 MUX IC)
11	74LS74 (Dual positive edge triggered D flip flop IC)

EXPERIMENT # 1

INTRODUCTION TO LOGIC TRAINER & LOGIC PROBE

OBJECTIVE:

- To get familiar with the equipment that would be used in rest of the semester for the implementation and testing of logic circuits

EQUIPMENT: Logic trainer, Logic probe

THEORY:

In this manual, all experiments would be performed on ETS-5000 logic trainer using low power TTL ICs. Moreover, for testing of logic circuits GLP-1A logic probe would be used. In this experiment, both logic trainer and logic probe are introduced.

Logic trainer:

ETS-5000 logic trainer is a low cost, high performance digital logic teaching system. It is designed to provide all the basic tools necessary to conduct logic experiments. It is also ideally suited for developing, debugging, integrating and testing digital systems. Figure 1-1 shows the front panel of ETS-5000 logic trainer. The front panel is explained in detail below:

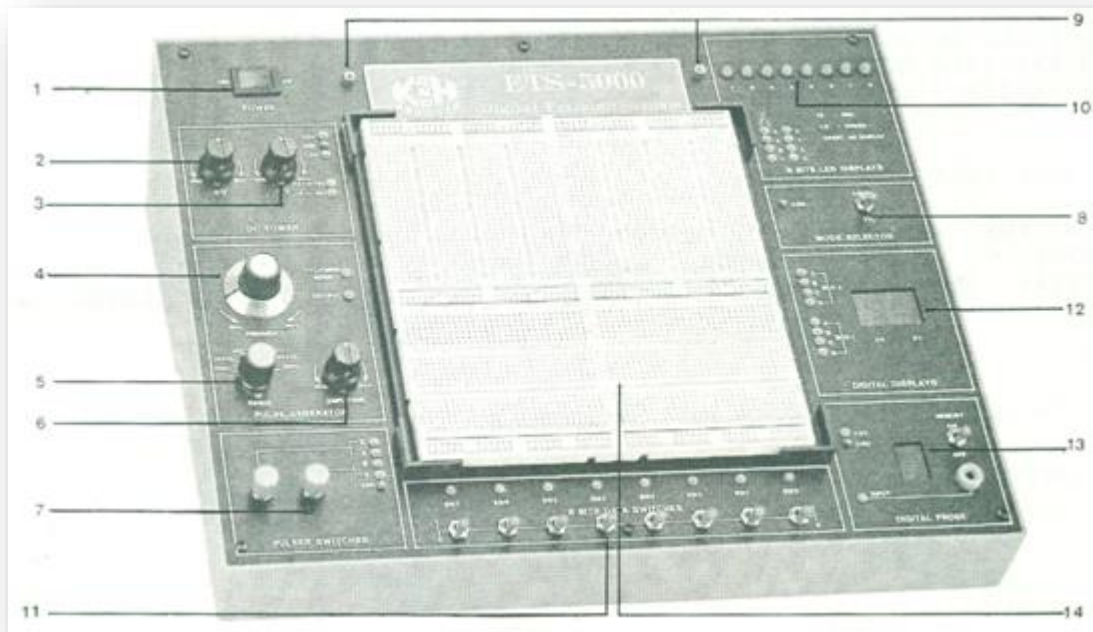


Figure 1-1 Front Panel of Digital Training System ETS-5000

Components of ETS-5000 Trainer:

1. **Power Switch:**A switch to turn ON/OFF the logic trainer.
2. **Variable Positive Power:**A variable DC power supply with DC output from 3V to 15V, 500mA.
3. **Variable Negative Power:**A variable DC power supply with DC output from -3V to -15V, 500mA.
4. **Frequency Variable:**A knob to vary the frequency of the output square waveform.
5. **Frequency Range:**A knob to select one of the frequency ranges which are as follows:
 - a) 1Hz – 10Hz
 - b) 10Hz – 100Hz
 - c) 100Hz – 1kHz
 - d) 1kHz – 10kHz
 - e) 10kHz – 100kHz
 - f) 100kHz – 1MHz
6. **Amplitude Variable:**A knob to vary the amplitude of the output square waveform. The amplitude can vary from **0 – 10V_{p-p}**.
7. **Pulse Switches:**Two pulse switches (push buttons) A and B along with their complements \bar{A} and \bar{B} are available to generate pulse as input.
8. **Mode Selector:** Using mode selector switch the mode can be selected between TTL logic and CMOS logic. TTL logic will be used throughout in this manual.
9. **Universal Connector Fixed Holders:**These are reserved fixed holders on the panel in order to be connected with various connectors.
10. **8 Bits LED Display:**Eight LEDs from 0 -- 7 are available to test the state of the logic output. The logic level in correspondence to the input applied at the LED will be as follow in table 1-1:

Logic Level	Input Level	Display Light Up
LO	$< 0.8V \pm 0.2V$	Green
HI	$> 2.3V \pm 0.2V$	Red
OPEN	0.8V - 2.3V	No display (LED is OFF)

Table 1-1: Input and Logic level

11. **8 Bits Data Switches:** Eight binary switches from SW0 -- SW7 are available to provide binary input to the logic circuit. The output level of a switch in correspondence to the logic level will be as follows in table 1-2:

Logic Level	Output Level
LO	0V
HI	5V

Table 1-2: Logic and Voltage Level

12. Digital Displays:Two digits of BCD to 7-segment LED display are available on the logic trainer.The output on 7-segment display in correspondence to the BCD input applied at the input connections will be as follows in table 1-3:

DECIMAL	BCD INPUT			
	D	C	B	A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

Table 1-3: Decimal to BCD(H= Logic High, L= Logic Low)

13. Digital Probe:Digital probe is available on the trainer to troubleshoot the logic circuit. The logic level in correspondence to the input level applied at digital probe will be as follows in table 1-4:

Logic Level	Input Level	Display
LO	$< 0.8V \pm 0.2V$	L
HI	$> 2.3V \pm 0.2V$	H
OPEN	$0.8V - 2.3V$	0
TRANSIT	LO \rightarrow HI	P

Table 1-4: Logic probe input and output level logic levels

14. Breadboard:Breadboard is available on the trainer to implement the digital circuit on it.

Logic Probe:

A logic probe can be very handy for troubleshooting and analysis of a digital logic circuit. As compared to a DC voltmeter or an oscilloscope, it needs to distinguish between the states of LOW and HIGH, and so it can be very simple and inexpensive. Its features include two LED indicators: HI (red LED) and LO (green LED) as shown in Figure 1-2.



Figure 1-2 Logic Probe GLP-1A

Specifications:

Model: GLP-1A Logic Probe

Operation:

1. Connect the black alligator clip to ground or common of the circuit under test.
2. Connect the red alligator clip to Vcc (4V DC minimum to 18V DC maximum) of the circuit under test.
3. Touch the probe tip to the circuit point under test. The LEDs on probe indicate the logic level or signals present when the circuit node is probed.
4. The LED response is noted below in table 1-5:

Input signal	LEDs
Logic "1"	HI (Red) ON
Logic "0"	LO (Green) ON
Bad Logic Level or Open circuit	none
Square Wave < 200kHz	HI and LO blinking at frequency rate
Square Wave > 200kHz	HI and LO blinking at frequency rate

Table 1-5: Logic Probe function table

LAB TASK#1:

Using a switch and LED, on logic trainer, and jumper wires, determine the internal connections of the breadboard on logic trainer.

LAB TASK#2:

Display the number '39' on the two BCD-to-seven segment displays available on logic trainer. What BCD input should be applied at the two BCD-to-seven segment displays?

EXPERIMENT # 2**INTRODUCTION TO BASIC GATES AND TROUBLESHOOTING****OBJECTIVE:**

- To learn and understand the working of basic gates (AND, OR, NOT)
- To learn and understand troubleshooting of logic circuits

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04

THEORY:

Digital circuits are the electronic circuits that manipulate binary information. Logic gates are the basic building blocks in constructing digital circuits. Logic gate has one output and one or more inputs. Each logic gate performs a specific logical operation. The interconnection between inputs and outputs of gates form a digital circuit. Any digital circuit can be implemented using three basic logical operations called AND, OR, and NOT. That is why AND, OR, and NOT gates are referred to as basic logic gates. AND and OR logic functions exhibit the phenomenon of dominance. In both cases, there is an input value that will force the output of the gate to a known value regardless of the state of other inputs. This value is known as the dominant value of the gate. The dominant value of an AND gate is zero, while the dominant value of an OR gate is one.

In this experiment, we will use 74LS08, 74LS32, and 74LS04 ICs for the implementation of AND, OR, and NOT logical operations. 74LS08 IC contains four 2-input AND gates. The function table and connection diagram for this IC are shown below in table 2-1 and figure 2-1 respectively:

Function Table:

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

Table 2-1: And gate truth

Connection Diagram:

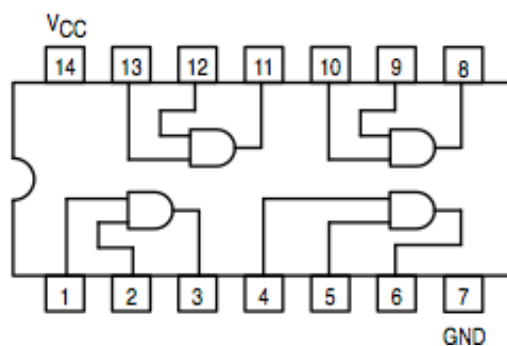


Figure 2-1: 74LS32 connection diagram

74LS32 IC contains four 2-input OR gates. The function table and connection diagram for this IC are shown below in table 2-2 and figure 2-2 respectively:

Function Table:

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

Table2-2: Or gate Truth table

Connection Diagram:

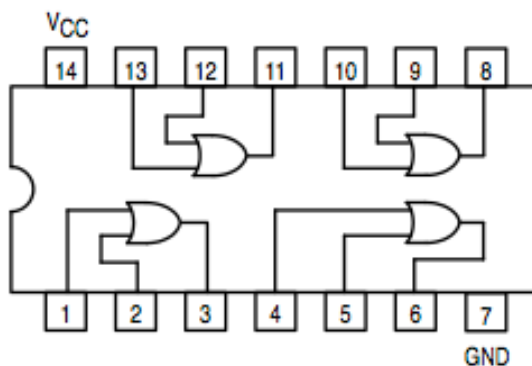


Figure 2-2:74LS32 connection diagram

74LS04 IC contains six NOT gates. The function table and connection diagram for this IC are shown below in table 2-3 and figure 2-3 respectively:

Function Table:

Input	Output
A	Y
L	H
H	L

Table 2-3 : Not gate truth table

Connection Diagram:

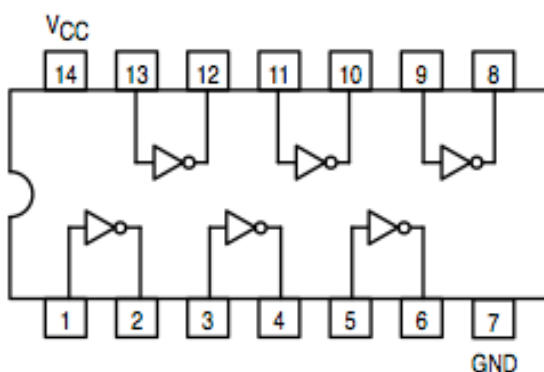


Figure 2-3: 74LS04 Connection diagram

Testing of ICs:

Before starting implementation of a specific logic circuit, all basic gate ICs should be tested in order to make sure that the ICs are working properly. Using the function table (truth table) for each gate, in a particular IC, apply all input combinations one by one and check its output logic level on LED.

Troubleshooting:

After testing all required number and type of ICs we need to implement a digital circuit, we start implementing the circuit on logic trainer. Once we complete the implementation, we need to test the output of the circuit to make sure that whether the circuit is working accurately or not. Using the truth table that represents the functionality of the logic circuit, we apply all input combinations one by one and check its output logic level on LED.

If the output of circuit is incorrect for some input combinations then there must be some fault in hardware implementation of the circuit. The fault can be an open circuit, short circuit, incorrect interconnection between the gates, incorrect connection with switches etc. Therefore, we need to troubleshoot the circuit. To troubleshoot the logic circuit apply the input combination at which the output of the circuit is incorrect, and then trace back the fault by checking each intermediate connection of the circuit using logic probe. The most efficient way to quickly reach the fault location is to exploit the low logic level dominance in AND gate and high logic level dominance in OR gate.

LAB TASK#1:

Implement the following logic circuit in figure 2-4 on logic trainer.

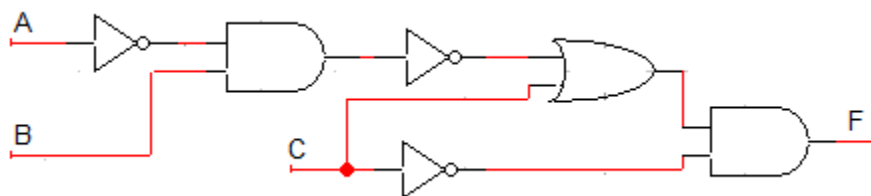


Figure 2-4: Logic Circuit 1

Write truth table in the space provided below:

[illegible]

LAB TASK#2:

For the logic circuit given in figure 2-5 do the following:

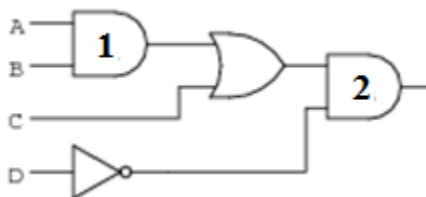


Figure 2-5: Logic Circuit 2

- i. If there is a fault in the circuit i.e. output of OR gate is stuck at zero, then predict how the operation of the logic circuit will be affected due to this fault.

--

- ii. If there is a fault in the circuit i.e. output of AND gate **1** is stuck at one, then at what input combinations the output would be wrong.

Post lab question:

Let us assume that the logic circuit given in lab task#2 is implemented on logic trainer and its output is wrong at (0000, 0010, 0100, 0110, 1000, and 1010) input combinations. What can be the fault in the implementation? Give answer with proper reasoning.

EXPERIMENT#3

BOOLEAN ALGEBRA TO OBTAIN COST EFFECTIVE CIRCUIT FOR IMPLEMENTATION

OBJECTIVE:

- To learn Boolean algebra and its usage to obtain cost effective circuit for implementation

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04

THEORY:

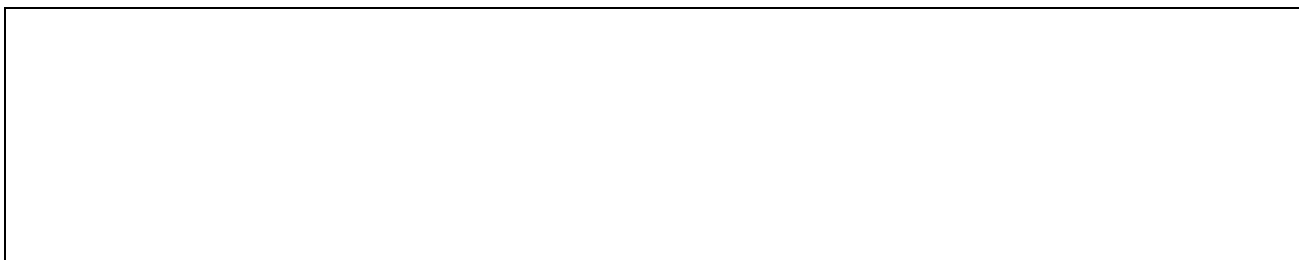
When a Boolean expression is implemented with logic gates, each term requires a gate, and each variable within the term designates an input to the gate. We define a **literal** as a single variable within the term that may or may not be complemented. By reducing the number of terms, the number of literals, or both in a Boolean expression, it is often possible to reduce the cost of circuit and obtain a simpler circuit. **Boolean algebra is applied to reduce an expression for obtaining a simpler circuit.** A Boolean function can be written in a variety of ways when expressed algebraically. There are, however, a few ways of writing algebraic expressions that are considered to be standard forms. The standard forms facilitate the simplification procedures for Boolean expressions and frequently result in more desirable logic circuits.

The standard forms contain product terms and sum terms. An example of a product term is XYZ. This is a logical product consisting of an AND operation among three literals. An example of a sum term is $X+Y+Z$. This is a logical sum consisting of OR operation among the literals.

LAB TASK#1:

For the Boolean function $F1 = [(A + B\bar{C})(D + \bar{A}\bar{C})] + CD$ do the following:

- a) Draw logic circuit diagram in the space provided below and implement the circuit on logic trainer.



--

b) Draw timing diagrams for construction and verification of the circuit

Inputs	A	
	B	
	C	
	D	
Output	F1(expected result)	
	F1(implementation result)	

c) Fill the following table 3-1 in order to determine the gate cost for the implementation of Boolean function $F1$

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 3-1: Gate cost for task 1

LAB TASK#2: For the Boolean function $F2 = \bar{A}\bar{B} + A\bar{D} + C$ do the following:

a) Draw logic circuit diagram in the space provided below and implement the circuit on logic trainer.

--

b) Draw timing diagrams for construction and verification of the circuit

Inputs	A	
	B	
	C	
	D	
Output	F1(expected result)	
	F1(implementation result)	

c) Fill the following table 3-2 in order to determine the gate cost for the implementation of Boolean function $F2$

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of Ics			

Table 3-2: Gate cost for task 2

LAB TASK#3: For the Boolean function $F3 = (A + \bar{B} + C)(\bar{A} + C + \bar{D})$, do the following:

a) Draw logic circuit diagram in the space provided below and implement the circuit on logic trainer.

b) Draw timing diagrams for construction and verification of the circuit

Lab Manual of Digital Logic Design

Inputs	A	
	B	
	C	
	D	
Output	F1(expected result)	
	F1(implementation result)	

- c) Fill the following table 3-3 in order to determine the gate cost for the implementation of Boolean function $F3$

IC type	Required No. of Gates	Gates per IC	Required No. of Ics
Total no. of Ics			

Table 3-3: Gate Cost for the task 3

Post lab questions:

- a) If you are asked to implement the following Boolean function using 2-input AND and NOT gates only. How you would implement it? Explain. Moreover, what would be the gate cost?

$$F = ABC + \overline{B}C + \overline{A}C$$

- b) Simplify the Boolean function using algebraic manipulation. What would be the gate cost if the simplified expression is implemented using 2-input AND and NOT gates only?

EXPERIMENT#4

KARNAUGH MAP TO OBTAIN COST EFFECTIVE CIRCUIT FOR IMPLEMENTATION

OBJECTIVE:

- To learn K-map and its usage in order to obtain cost effective circuit for implementation

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04

THEORY:

The Karnaugh map (K-map) is a method used to simplify Boolean expressions. K-Map is a grid-like representation of a truth table that gives more insight. The required Boolean results are transferred from a truth table onto a two-dimensional grid where the cells are ordered in gray code and each cell position represents one combination of input conditions, while each cell value represents the corresponding output value. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table. These terms can be used to write a minimal Boolean expression representing the required logic.

Karnaugh maps are used to obtain optimized logic representation so that it can be implemented using a minimum number of logic gates. The sum-of-product form can always be implemented using AND gates feeding into an OR gate, and a product-of-sum form leads to OR gates feeding an AND gate.

The minimization will result in reduction of the number of gates (resulting from less number of terms) and the number of inputs per gate (resulting from less number of variables per term). The minimization will reduce cost and power consumption of the logic circuit.

LAB TASK#1:

The light bulb is ON, if switch A is OFF and switch B is ON and either switch B is OFF or switch C is ON, or if switch A is ON and switch D is ON and either switch D is OFF or switch C is OFF

a) Find truth table for the light bulb

b) Write Boolean function for the light bulb in canonical form using minterms

c) Write Boolean function for the light bulb in canonical form using maxterms

- d)** Find minimal SOP expression for the light bulb using K-map. Draw K-map in the space given below:

- e)** Fill the following table 4-1 in order to determine the gate cost for the implementation of Boolean function for the light bulb found in part (d).

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-1: Gate cost for the task 1 part(d)

- f)** Find minimal POS expression for the light bulb using K-map. Draw K-map in the space given below:

- g)** Fill the following table 4-2 in order to determine the gate cost for the implementation of Boolean function for the light bulb found in part (f).

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-2: Gate cost for the task 1 part(f)

- h)** Find Boolean expression (mixed form) for the light bulb which is neither SOP nor POS expression.

- i)** Fill the following table 4-3 in order to determine the gate cost for the implementation of Boolean function for the light bulb found in part (h)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-3: Gate cost for the task 1 part(h)

- j) Implement the cost effective expression. Write the Boolean expression you have chosen to be cost effective in the space given below along with its logic diagram. Give proper reasoning for the chosen expression.

LAB TASK#2:

For the Boolean function $F1(A, B, C, D) = \sum m(2, 4, 12, 14)$ do the following:

- a) Find truth table

- b)** Find minimal SOP expression for Boolean function $F1$ using K-map. Draw K-map in the space given below:

- c)** Fill the following table 4-4 in order to determine the gate cost for the implementation of Boolean function $F1$ found in part (b).

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-4: Gate cost for the task 2 part(b)

- d)** Find minimal POS expression for Boolean function $F1$ using K-map. Draw K-map in the space given below:

- e) Fill the following table 4-5 in order to determine the gate cost for the implementation of Boolean function for $F1$ found in part (d).

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-5: Gate cost for the task 2 part(d)

- f) Find Boolean expression (mixed form) for Boolean function $F1$ which is neither SOP nor POS expression.

- g) Fill the following table 4-6 in order to determine the gate cost for the implementation of Boolean function $F1$ found in part (f).

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-6: Gate cost for the task 2 part(f)

- h)** Implement the cost effective expression. Write the Boolean expression you have chosen to be cost effective in the space given below along with its logic diagram. Give proper reasoning for the chosen expression.

LAB TASK#3:

For the Boolean function $F2(A, B, C, D) = \prod M(9, 13, 15)$ do the following:

- a)** Find truth table.

- b)** Find minimal SOP expression for Boolean function $F2$ using K-map. Draw K-map in the space given below

- c)** Fill the following table 4-7 in order to determine the gate cost for the implementation of Boolean function $F2$ found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-7: Gate cost for the task 3 part(b)

- d)** Find minimal POS expression for Boolean function $F2$ using K-map. Draw K-map in the space given below

- e) Fill the following table 4-8 in order to determine the gate cost for the implementation of Boolean function for F_2 found in part (d)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-8: Gate cost for the task 3 part(d)

- f) Find minimal expression (mixed form) for Boolean function F_2 which is neither SOP nor POS expression

- g) Fill the following table 4-9 in order to determine the gate cost for the implementation of Boolean function F_2 found in part (f)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 4-9: Gate cost for the task 3 part(f)

EXPERIMENT#5

THE USE OF UNIVERSAL GATES & MAPPING

OBJECTIVE:

- To study the realization of basic gates using universal gates (NAND gate & NOR gate)
- To learn technology mapping (NAND-NAND & NOR-NOR implementation) and its significance in order to obtain cost effective circuit for implementation

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS02, 74LS00

THEORY:

The design of a combinational circuit starts from the specification of the problem and culminates in a logic diagram or netlist that describes a logic diagram. The procedure involves the specification, formulation, optimization, & technology mapping.

Technology mapping is actually transformation of logic diagram or netlist to a new diagram using the available implementation technology. Typically NAND and NOR gates are more desirable to use in technology mapping due to the following reasons:

- 1) NAND and NOR gates are said to be universal gates where universal gate is a gate which can implement any Boolean function without needing any other type of gate.
- 2) Using universal gate in technology mapping may further reduce cost of optimized logic diagram.
- 3) Universal gates are easier to fabricate with electronic components.

A convenient way to implement a Boolean function with NAND gates only (NAND-NAND implementation) is to begin with the optimized logic diagram of the circuit consisting of AND, OR and NOT gates. The function is converted to pure NAND logic by replacing each gate in logic diagram with its representation using NAND gates only as shown in figure 5-1. The function table of the NAND gate is given in table 5-1. After that, all inverter pairs are cancelled. The same conversion procedure is applied to implement a Boolean function with NOR gates only (NOR-NOR implementation).

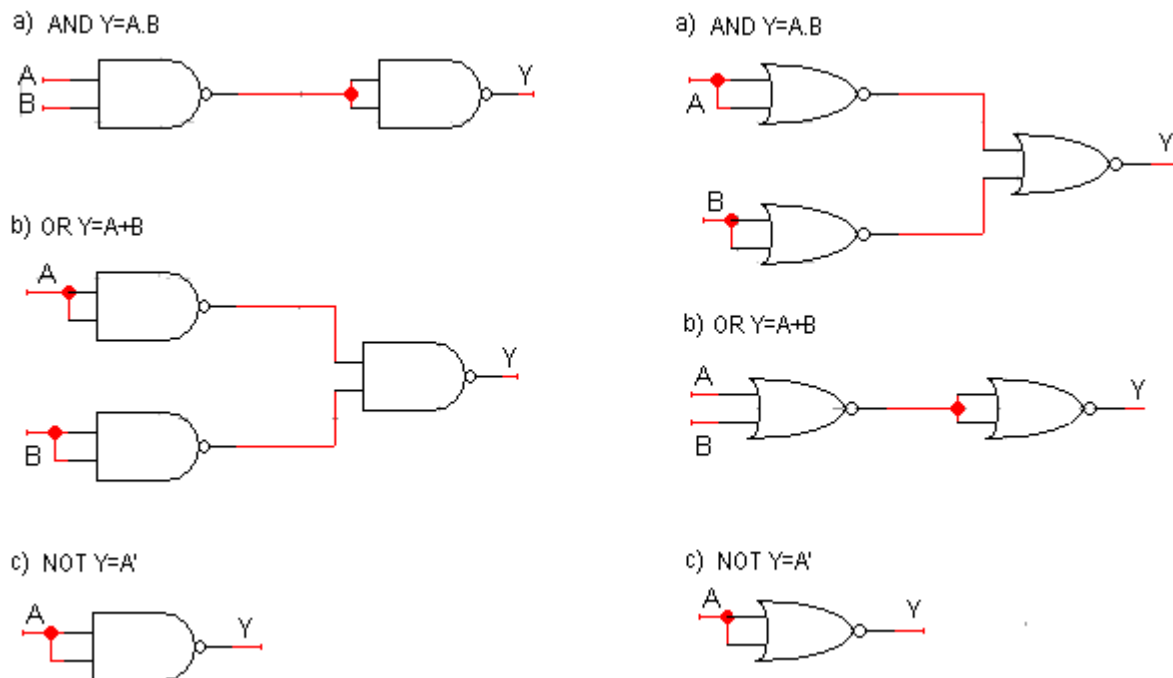


Figure 5-1: NAND-NAND and NOR-NOR representation of basic logic gates

In this experiment, we will use 74LS00 and 74LS02 ICs for NAND-NAND & NOR-NOR implementation. 74LS00 IC contains four 2-input NAND gates. The function table and connection diagram for this IC are shown below in table 5-2 and figure 5-2 respectively:

Function Table:

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

Table 5-1: Nand gate truth table

Connection Diagram:

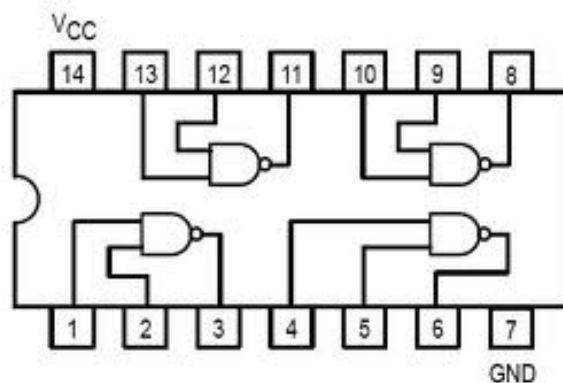


Figure 5-2: 74LS00 Connection diagram

74LS02 IC contains four 2-input NOR gates. The function table and connection diagram for this IC are shown below in table 5-2 and figure 5-3 respectively:

Function Table:

Inputs		Output
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

Table 5-2: Nor gate truth table

Connection Diagram:

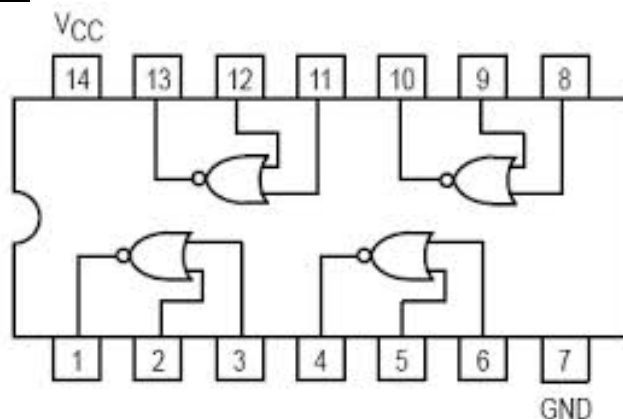


Figure 5-3: 74LS02 Connection diagram

LAB TASK#1:For the logic circuit given below in figure 5-4 do the following:

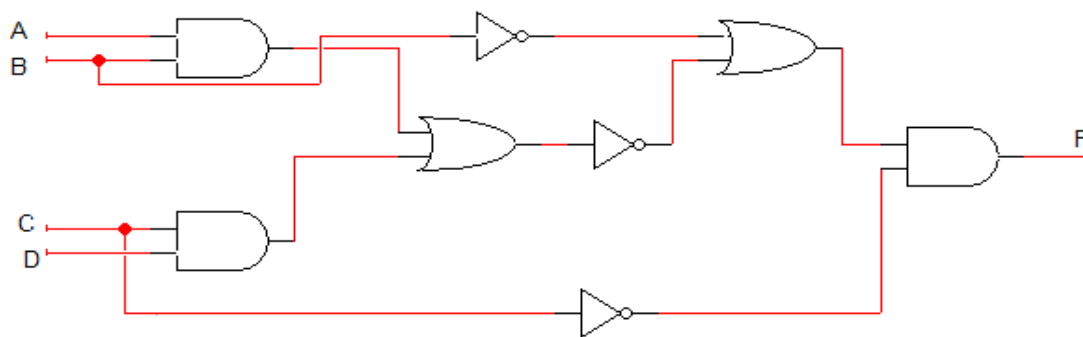


Figure 5-4 logic Circuit 1

a) Write Boolean expression for **F**

b) Write the truth table for Boolean expression **F**

c) Write Boolean function **F** in canonical form using minterms

d) Write Boolean function **F** in canonical form using maxterms

- e) Fill the following table 5-3 in order to determine the gate cost for the implementation of Boolean function **F** found in part (a)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 5-3: Gate Cost task 1 part (a)

- f) Transform the given diagram of logic circuit to new logic diagram using NAND gates only, in the space given below. Show complete working. Implement the transformed logic circuit on logic trainer.

- g) Fill the following table 5-4 in order to determine the gate cost for the implementation of Boolean function **F** using NAND gates only.

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
74LS00 (NAND gate IC)			
Total no. of ICs			

Table 5-4: Gate Cost task 1 part (f)

- h) Transform the given diagram of logic circuit to new logic diagram using NOR gates only, in the space given below. Show complete working. Implement the transformed logic circuit on logic trainer.

- i) Fill the following table 5-5 in order to determine the gate cost for the implementation of Boolean function **F** using NOR gates only.

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
74LS02 (NOR gate IC)			
Total no. of ICs			

Table 5-5: Gate Cost task 1 part (h)

Post lab question:

Design a logic circuit that implements the equation of a straight line i.e. $y = mx + c$ where $m = 3$, $c = 1$, & x is a 3-bit input vector. Would technology mapping (NAND-NAND & NOR-NOR implementation) help you obtain cost effective circuit for implementation? If yes, then which transformation (NAND-NAND or NOR-NOR) is preferable and how? Describe all steps while designing the required digital system, from truth table to the circuit diagram (using your optimized solution). Also, explain how you obtained the optimized solution.

EXPERIMENT#6**DESIGN OF A BINARY COMPARATOR****OBJECTIVE:**

- To learn and understand how to design a multiple output combinational circuit
- To learn and understand the working of 2-bit binary comparator
- To learn and understand the working and usage of Exclusive-OR and Exclusive-NOR gates

EQUIPMENT: Logic trainer, Logic probe

6+COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86, 74LS02

THEORY:

Binary comparator is a combinational circuit that compares magnitude of two binary data signals A & B and generates the results of comparison in the form of three output signals $A > B$, $A = B$, $A < B$. Binary comparator is a multiple input and multiple output combinational circuit. When a combinational circuit has two or more than two outputs then each output is expressed separately as a function of all inputs. Separate K-map is made for each output. The truth table of the binary comparator is given in table 6-1.

One-bit comparator:

One-bit comparator compares magnitude of two numbers A and B, 1 bit each, and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

$$L = 1 \text{ if } A < B$$

$$E = 1 \text{ if } A = B$$

$$G = 1 \text{ if } A > B$$

Truth Table:

Inputs		Outputs		
A	B	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Table 6-1: 1 Bit comparator truth table

K-Maps for Outputs:

		B	
		0	1
A	0		1
	1		

K-Map for Output **L**

		B	
		0	1
A	0	1	
	1		1

K-Map for Output **E**

		B	
		0	1
A	0		
	1	1	

K-Map for Output **G**

Boolean Expressions of Outputs:

L: $\bar{A}B$

E: $AB + \bar{A}\bar{B}$

G: $A\bar{B}$

Circuit Diagram for one-bit comparator:

The circuit diagram of the one bit comparator circuit is given in figure 6-1 below.

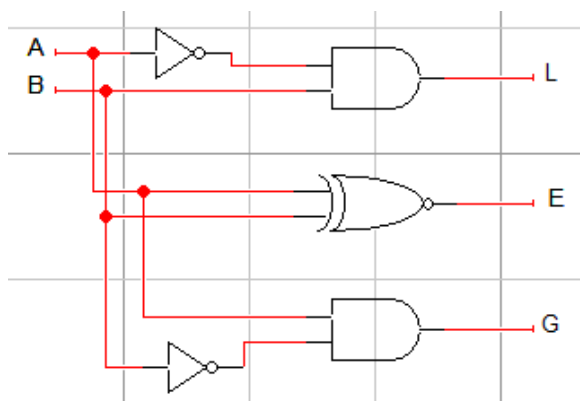


Figure 6-1 one-bit comparator circuit

Exclusive-OR & Exclusive-NOR gates:

The figure 6-2 and figure 6-3 given below shows the symbol of Exclusive-OR (XOR) and Exclusive-NOR (XNOR) gates.

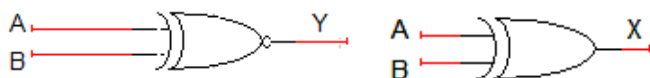


Figure 6-2 XNOR gate

Figure 6-3 XOR gate

Boolean expression of XNOR gate is $AB + \bar{A}\bar{B}$ and Boolean expression of XOR is $\bar{A}B + A\bar{B}$. Boolean expression of XNOR gate can be implemented using XOR gate as shown in figure 6-4 below:

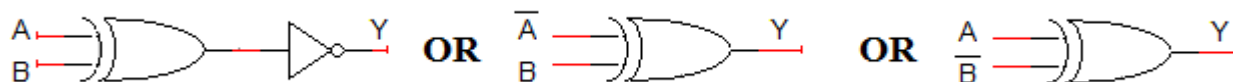


Figure 6-4 XNOR gate using XOR gate

In this experiment 74LS86 IC will be used for implementation of XOR gate function. 74LS86 IC contains four 2-input XOR gates. The function table and connection diagram for this IC are shown below in table 6-2 and figure 6-5.

Function Table:

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

Table 6-2: Xor gate truth table

Connection Diagram:

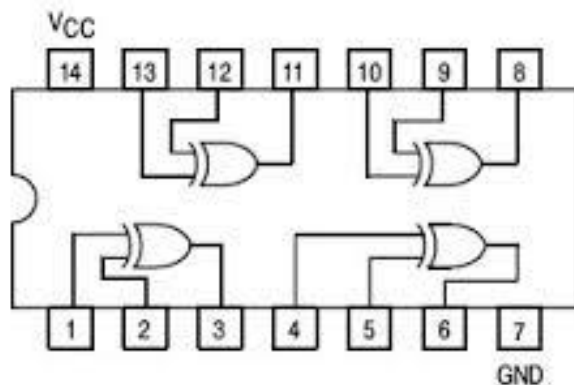


Figure 6-5: 74LS86 Connection Diagram

LAB TASK:

Design a combinational circuit that compares two 2-bit numbers and generates the comparison result. The result consists of three outputs let us say L, E, G, so that

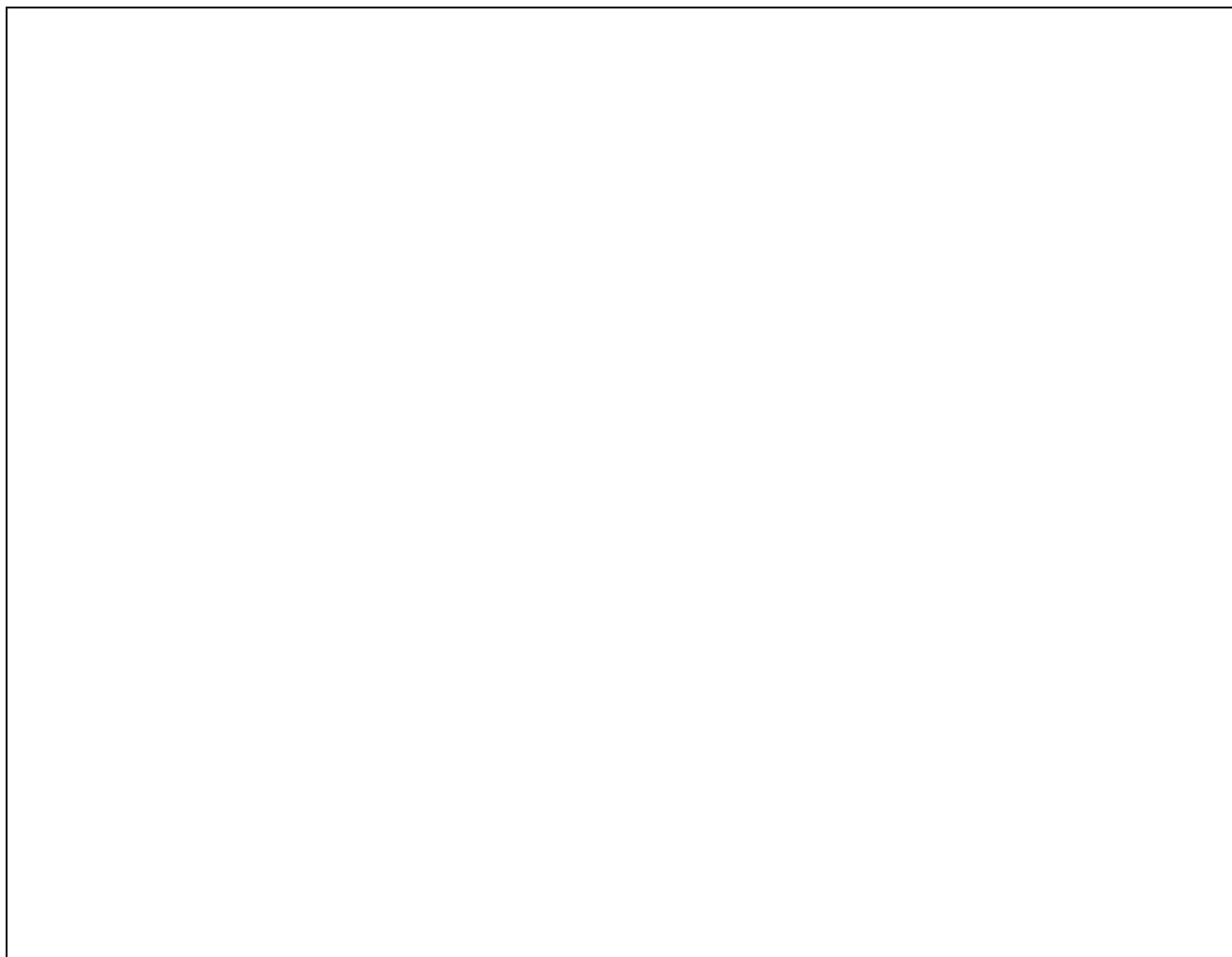
$$L = 1 \text{ if } A < B$$

$$E = 1 \text{ if } A = B$$

$$G = 1 \text{ if } A > B$$

- a) Write truth table

- b) Find minimal SOP and POS expressions for the outputs **L**, **E**, and **G** using K-map. Draw separate K-map for each output in the space given below



- c) Fill the following table 6-3 in order to determine the gate cost for the implementation of binary comparator using SOP expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 6-3: Gate Cost for task 1 SOP part(b)

- d) Fill the following table 6-4 in order to determine the gate cost for the implementation of binary comparator using POS expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 6-4: Gate Cost for task 1 POS part(b)

- e) Find Boolean expressions (mixed form) for the outputs of binary comparator. (**Hint:** Apply factoring on SOP expressions of the outputs)

- f) Fill the following table 6-5 in order to determine the gate cost for the implementation of binary comparator using expressions found in part (e)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 6-5: Gate Cost for task part(e)

- g) Can we implement any one of the three outputs of the binary comparator from the rest of the two outputs? If yes, then explain how we can do that. (**Hint:** Write Boolean expression for one output in terms of the other two outputs)

- h) What are the advantages and disadvantages of implementing any one of the three outputs of the binary comparator from the other two outputs?

- i) Implement the optimal solution. In the space given below, write the Boolean expressions you have chosen to be cost effective for the implementation of 2-bit comparator along with the logic diagram. Give proper reasoning for the chosen solution.

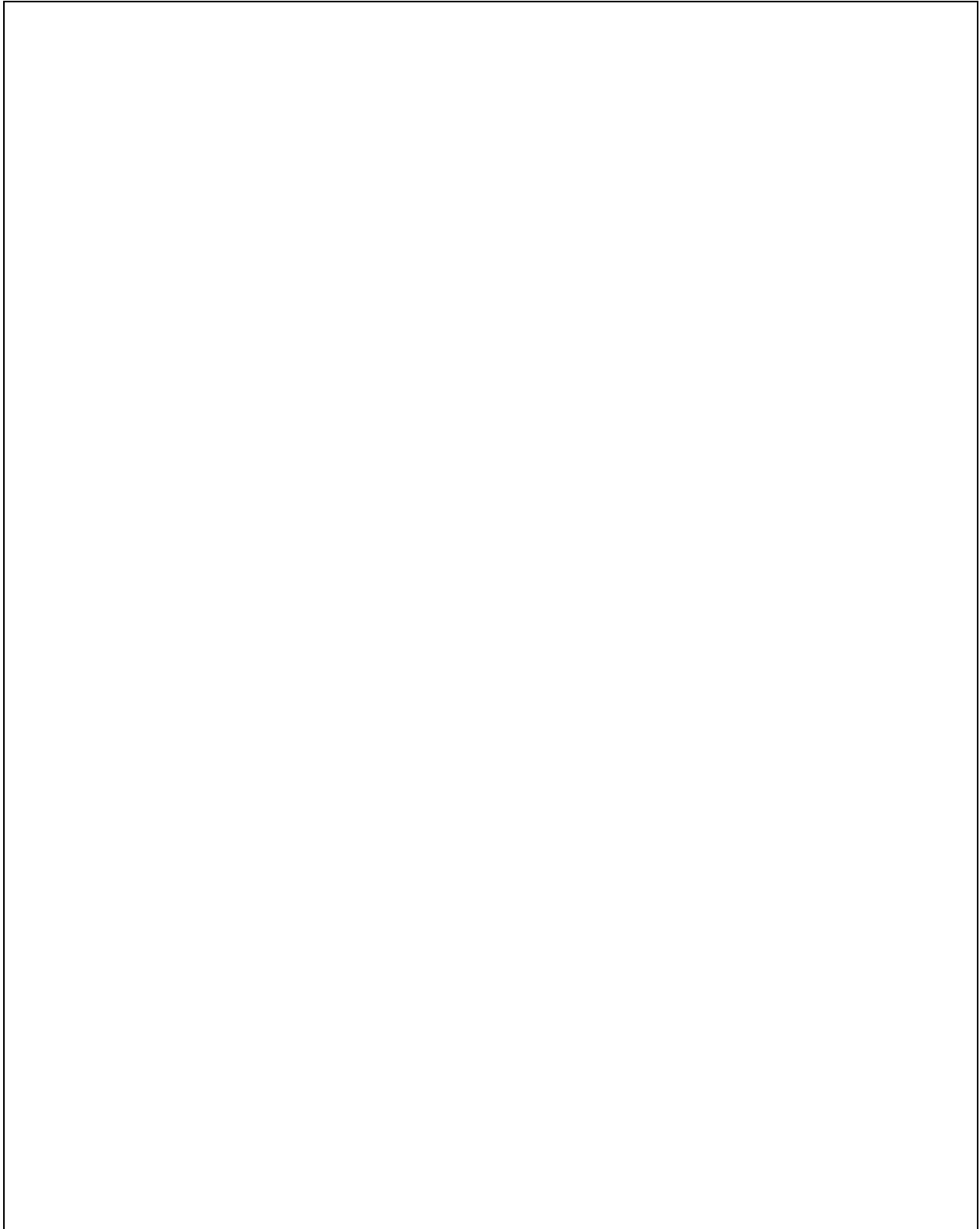
Post lab question:

Design a logic circuit that take two 2-bit unsigned numbers **A** and **B** as input and has two outputs **X** and **Y** so that

$$X = 1 \text{ if } (A + B) = 2$$

$$Y = 1 \text{ if } (A + B) > 2$$

Describe all steps while designing the required digital system, from truth table to the circuit diagram (using your optimized solution). Also, explain how you obtained the optimized solution.



EXPERIMENT#7**DESIGN OF DECODER, PRIORITY ENCODER, & MULTIPLEXER****OBJECTIVE:**

- To study the basic operation and design of the decoder, priority encoder and multiplexer circuits
- To learn the concept of enabling a signal (active-low and active-high enable)

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS00, 74LS02

THEORY:**Enabling:**

The concept of enabling is to permit an input signal to pass through or is blocked completely. The enable bit is also used to enable or disable the normal functioning of a logic circuit. A logic circuit can be enabled or disabled using a single enable bit. This enable bit can be either active low or active high. Let us have an example of an AND gate with enable bit. If the enable bit is active high then it means that the AND gate performs normal operation if the enable bit is one else the output of AND gate is forced to be zero. The truth table and logic circuit are blow in table 7-1 and figure 7-1 respectively.

Truth Table:

Inputs			Output
E	A	B	$F = \begin{cases} AB & \text{if } E = 1 \\ 0 & \text{otherwise} \end{cases}$
0	X	X	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table:7-1 Truth table for AND gate with active High enable

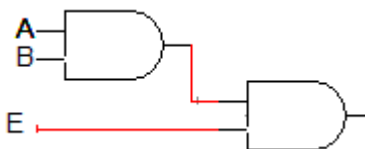
Circuit Diagram:

Figure 7-1:AND gate with active high enable logic circuit

If the enable bit is active low then it means that the AND gate performs normal operation if the enable bit is zero else the output of AND gate is forced to be zero. The truth table and logic circuit are blow in table 7-2 and figure 7-2 respectively.

Truth Table:

Inputs			Output
E	A	B	$F = \begin{cases} AB & \text{if } E = 0 \\ 0 & \text{otherwise} \end{cases}$
1	X	X	0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1

Table:7-2 Truth table for AND gate with active High enable

Circuit Diagram:

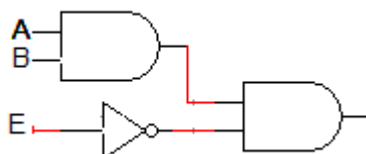


Figure 7-2:AND gate with active high enable logic circuit

Decoder:

A decoder is a combinational circuit that decodes the encoded inputs. A binary decoder has n inputs and a maximum of 2^n outputs. An n -bit binary number provides 2^n minterms or maxterms. The decoder indicates one of the 2^n minterms or maxterms at the outputs based on the input combinations. The decoder that produces 2^n minterms as its outputs is said to be a decoder with active high outputs, whereas, the decoder that produces 2^n maxterms as its outputs is said to be a decoder with active low outputs. Let us take $n=2$ as an example, so that we obtain the 2-to-4 line decoder with active high outputs. Figure 7-3 shows the block diagram of 2x4 decoder. Table 7-3 show the truth table

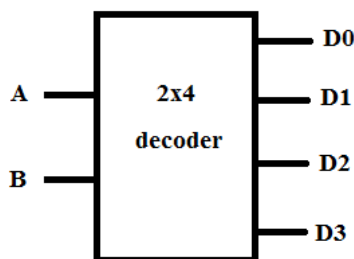


Figure 7-3: Decoder block diagram

Truth Table:

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

Table 7-3: Decoder truth table

Boolean Expressions of Outputs:

D0: $\bar{A}\bar{B}$

D1: $\bar{A}B$

D2: $A\bar{B}$

D3: AB

The Boolean expressions show that four outputs of 2x4 decoder show four minterms of two binary variables **A** and **B**. Figure 7-4 show the logic circuit of the decoder and figure 7-5 is logic circuit for the decoder with active high enable/

Circuit diagram for 2x4 decoder with active high outputs:

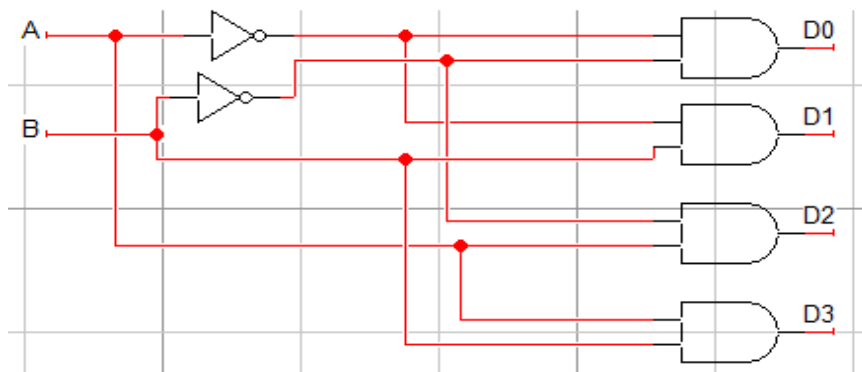


Figure 7-4: Decoder circuit Diagram

Circuit diagram for 2x4 decoder with active high outputs and active high enable:

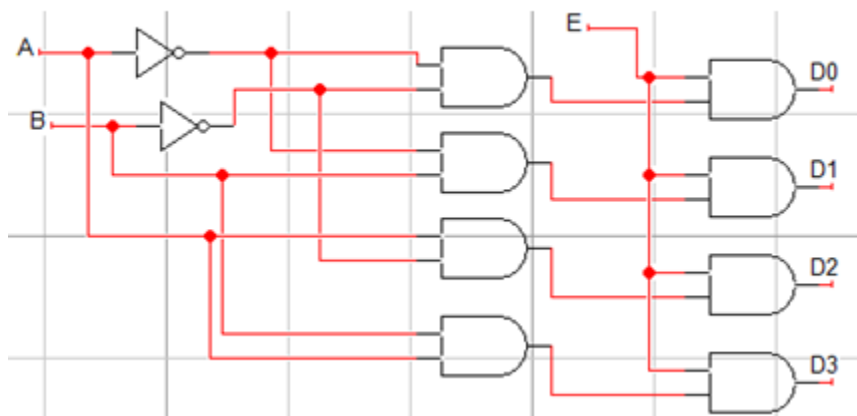


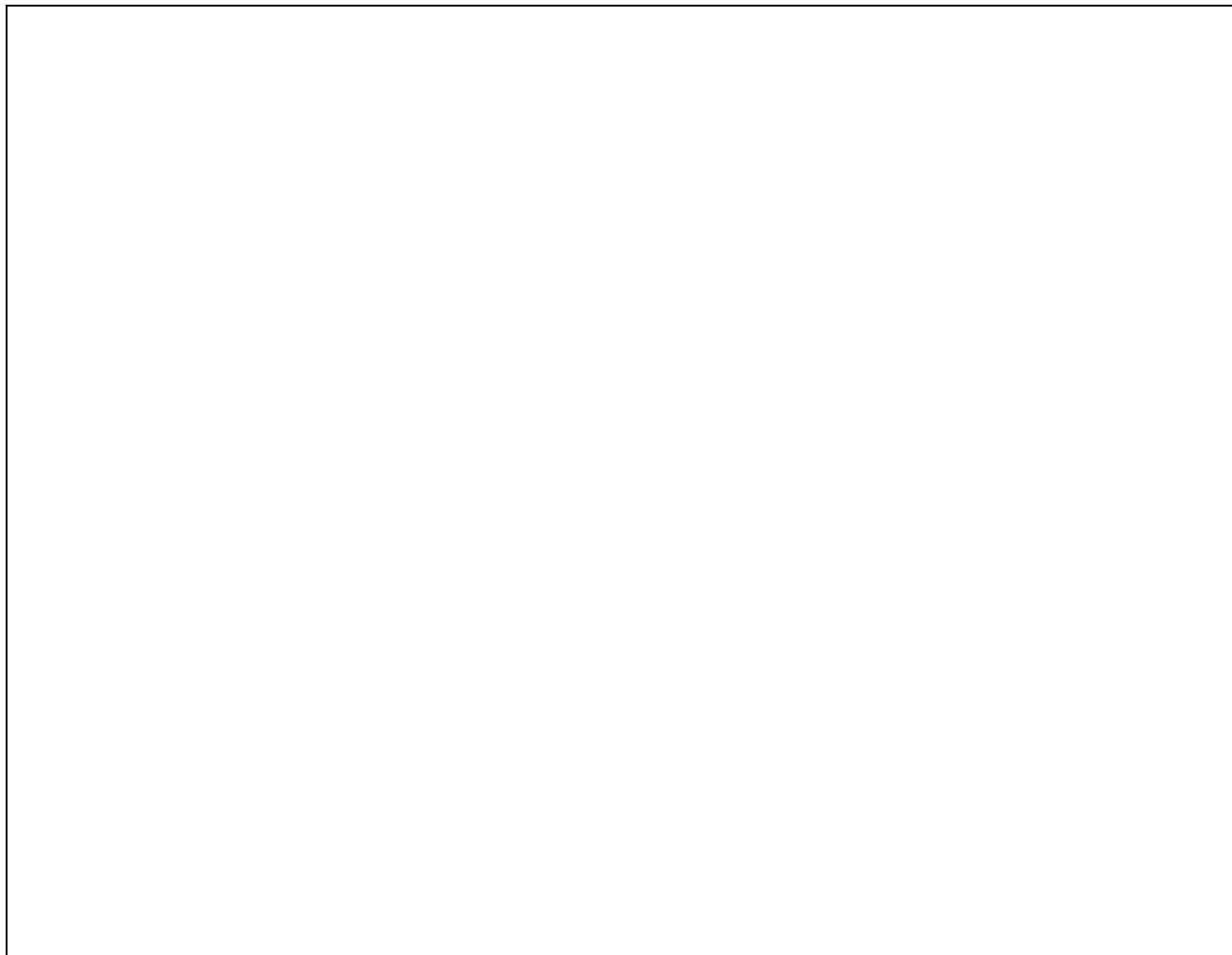
Figure 7-5: Decoder with active high enable circuit Diagram

LAB TASK#1:

Design and implement a 2-to-4 line decoder with active low outputs along with active low enable input E. When E is low, the decoder will operate normally, when E is high, all outputs should be high regardless of the inputs.

a) Write truth table

- b)** Find minimal SOP and POS expressions for the 2x4 decoder outputs using K-map. Draw separate K-map for each output in the space given below



- c)** Fill the following table 7-4 in order to determine the gate cost for the implementation of 2x4 decoder using SOP expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 7-4: Gate cost for task 1 part(b) SOP

- d)** Fill the following table 7-5 in order to determine the gate cost for the implementation of 2x4 decoder using POS expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

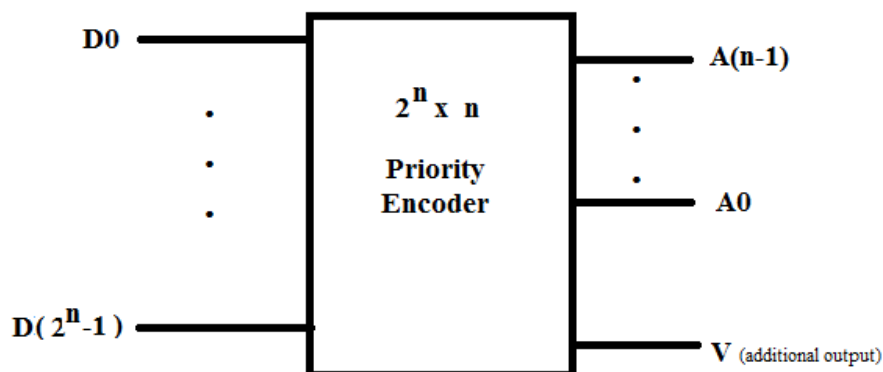
Table 7-5: Gate cost for task 1 part(b) POS

- e)** In the space given below, write the Boolean expressions for the implementation of 2x4 decoder along with the logic diagram.

Priority Encoder:

An encoder is a combinational circuit that performs the inverse operation of a decoder. An encoder has a maximum of 2^n input lines and n output lines. The encoder generates binary code at its output lines that represents which input line is active at a given time. In encoder, it is assumed that only one input is active high at a time, if more than one inputs are high simultaneously then ambiguous output is generated. In order to resolve this ambiguity, there must be some input priority function to ensure that only one input is encoded at a time.

A priority encoder is a combinational circuit that encodes the input using priority function i.e. if more than one inputs are high simultaneously then the input having the highest priority will take precedence. Each input line is assigned priority. The most significant input line may be given highest priority and least significant input line the lowest or vice versa. The priority encoder has an additional output to ensure that at least one input line is active high and the binary code at the output lines is valid. Figure 7-6 shows the block diagram of $2^n \times n$ priority encoder.

Figure 7-6: Priority encoder ($2^n \times n$)**LAB TASK#2:**

Design and implement a 4-input priority encoder in which least significant input line has highest priority and most significant input line has the lowest priority. The priority encoder must have an additional output to ensure that at least one input line is active high and the binary code at the output lines is valid.

a) Write truth table

b) Find minimal SOP and POS expressions for the 4-input priority encoder outputs using K-map.
Draw separate K-map for each output in the space given below

- c) Fill the following table 7-6 in order to determine the gate cost for the implementation of 4-input priority encoder using SOP expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 7-6: Gate cost for task 2 part(b) SOP

- d) Fill the following table 7-7 in order to determine the gate cost for the implementation of 4-input priority encoder using POS expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 7-7: Gate cost for task 2 part(b) POS

- e) Implement the optimal solution. In the space given below, write the Boolean expressions you have chosen to be cost effective for the implementation 4-input priority encoder along with the logic diagram. Give proper reasoning for the chosen solution.

Multiplexer:

A multiplexer is a multiple input and a single output combinational circuit that selects and directs the binary information from multiple input lines to a single output line. This selection of a particular input is done by selection inputs. A multiplexer has 2^n input lines and n selection lines whose bit combinations determine which input line is selected. Figure 7-7 shows the block diagram of a $2^n \times 1$ multiplexer.

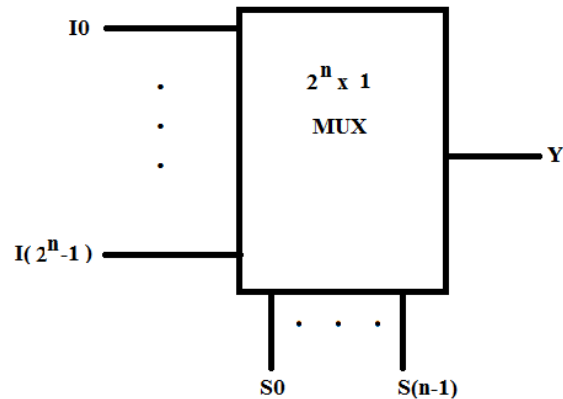



Figure 7-7: Multiplexer ($2^n \times 1$)

Lab Task#3:

Design and implement 4x1 multiplexer with active low enable E. When E is low, the multiplexer will work normally, when E is high, the output should be high regardless of the inputs.

- a) Write truth table**



b) Write minimal SOP expression for the output of 4x1 multiplexer in the space given below

c) Fill the following table 7-8 in order to determine the gate cost for the implementation of 4x1 multiplexer using expression found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 7-8: Gate cost for task 2 part(b) SOP

d) In the space given below, write the Boolean expression for the implementation of 4x1 multiplexer along with the logic diagram.

e) Test your implementation of 4x1 multiplexer for the following inputs:

$$I_0 = A, I_1 = B, I_2 = 0, \& I_3 = 1$$

Write truth table for the given inputs in the space given below.

Post lab questions:

- 1) If we have a 6x1 array of LEDs, and at a time one LED in the array should glow. Which functional block can be used to model the desired situation? What specification the functional block must possess?

- 2)** In a live quiz competition to answer the questions you have one help line. You can contact one of your selected four friends to get help. Which functional block comes in your mind that best suit the mentioned situation? What should be the specification of functional block that you are thinking?

EXPERIMENT#8**IMPLEMENTATION OF LOGIC CIRCUIT USING DECODER & MULTIPLEXER IC****OBJECTIVE:**

- To study the concept of dual multiplexers
- To learn the implementation of Boolean function using decoder and multiplexer
- To learn how to implement large multiplexer using small size multiplexers
- To learn how to implement large decoder using small size decoders

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS139, 74LS153, 74LS08, 74LS32, 74LS00

THEORY:

In this experiment, we will use 74LS139 and 74LS153 ICs as decoder and multiplexer. 74LS139 IC contains two fully independent 2-to-4 line decoders with active low enables. The function table and connection diagram for this IC are shown below in table 8-1 and figure 8-1 respectively:

Function Table:

Enable	Selection Inputs		Outputs			
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

Table 8-1: 74LS139 Decoder function table

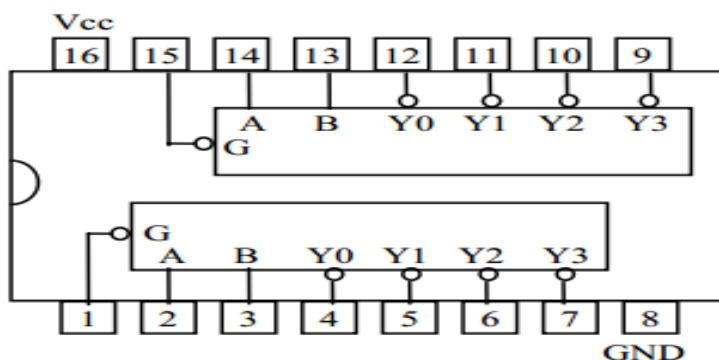
Connection Diagram:

Figure 8-1: 74LS139 Decoder Connection Diagram

74LS153 IC is a dual 4x1 MUX with active low enables. Two 4x1 MUXs with common selection pins but independent inputs and independent outputs is known as dual 4x1 MUX. The function table and connection diagram for this IC are shown below in table 8-2 and figure 8-2 respectively:

Function Table:

Strobe (Enable)	Selection Inputs		Data Inputs				Output
G	B	A	C0	C1	C2	C3	Y
H	X	X	X	X	X	X	L
L	L	L	L	X	X	X	L
L	L	L	H	X	X	X	H
L	L	H	X	L	X	X	L
L	L	H	X	H	X	X	H
L	H	L	X	X	L	X	L
L	H	L	X	X	H	X	H
L	H	H	X	X	X	L	L
L	H	H	X	X	X	H	H

Table 8-2: 74LS153 Multiplexer function table

Connection Diagram:

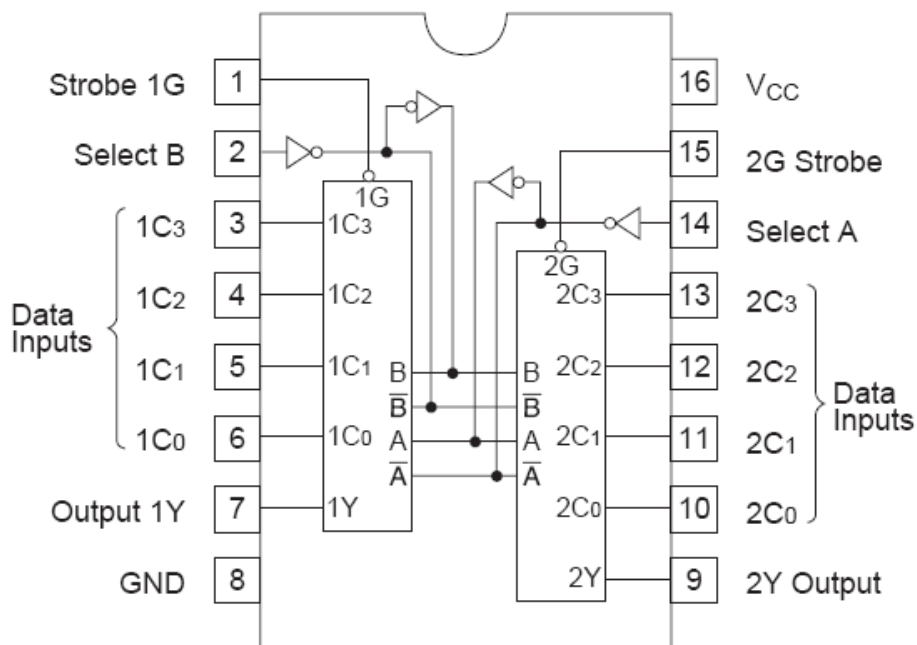


Figure 8-2: 74LS153 Multiplexer Connection Diagram

LAB TASK#1:

Three PCs are sharing a single printer. Design a logic circuit that displays how many PCs are sending print request to the printer at a time. The count is displayed on LEDs.

a) Write truth table

b) Design a 3-to-8 line decoder using two 2-to-4 line decoders with active low outputs and active low enables. Show circuit diagram in space given below.

- c) Implement the printer request counter using 74LS139 (two 2x4 decoders). Show the circuit diagram in the space given below

LAB TASK#2:

Implement the printer request counter using 74LS153 (dual 4x1 MUX). Show complete working from truth table to circuit diagram.

LAB TASK#3:

- a) Construct 8x1 MUX using dual 4x1 MUX with active low enables. Show circuit diagram in the space given below

- b) Implement the Boolean function $F(A, B, C, D) = \sum m(0, 1, 5, 6, 8, 11, 14, 15)$ using 74LS153 (dual 4x1 MUX). Show complete working in the space given below from truth table to circuit diagram.

Post lab question:

Design a logic circuit that take 3-bit input number and display its square as an output on seven-segment display (common anode configuration). How many seven-segment display digits you need in order to show the output? First design the circuit using decoders and then design the circuit using multiplexers. Which design among the two is cost effective and why? Give comprehensive explanation.

EXPERIMENT#9**DESIGN OF ADDER & SUBTRACTOR CIRCUITS****OBJECTIVE:**

- To study the basic operation and design of half adder, half subtractor, full adder, and full subtractor circuits

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86

THEORY:**Half Adder:**

Half adder is a logic circuit that performs binary addition of two 1-bit numbers. It generates two outputs namely 'Sum' and 'Carry' as shown in table 9-1.

Truth Table:

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

Table 9-1: Half Adder Truth table

Boolean Expressions of Outputs:

$$Sum = A \oplus B$$

$$Carry = AB$$

Half Subtractor:

Half subtractor is a logic circuit that performs binary subtraction of two 1-bit numbers. It generates two outputs namely 'Difference' and 'Borrow' as shown in table 9-2.

Truth Table:

A	B	Borrow	Difference
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Table 9-2: Half Subtractor Truth table

Boolean Expressions of Outputs:

$$\text{Difference} = A \oplus B$$

$$\text{Borrow} = \bar{A}B$$

Full Adder:

Full adder is a logic circuit that performs binary addition of two 2-bit numbers. It generates two outputs namely 'Sum' and 'Carry' as shown in table 9-3.

Truth Table:

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

Table 9-3: Full Adder Truth table

Boolean Expressions of Outputs

$$\text{Sum} = A \oplus B \oplus C$$

$$\text{Carry} = AB + BC + AC \text{ or } \text{Carry} = AB + C(A \oplus B)$$

Full Subtractor:

Full subtractor is a logic circuit that performs binary subtraction of two 2-bit numbers. It generates two outputs namely 'Difference' and 'Borrow' as shown in table 9-4.

Truth Table:

A	B	C	Borrow	Difference
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 9-4: Full Subtractor Truth table

Boolean Expressions of Outputs:

$$\text{Difference} = A \oplus B \oplus C$$

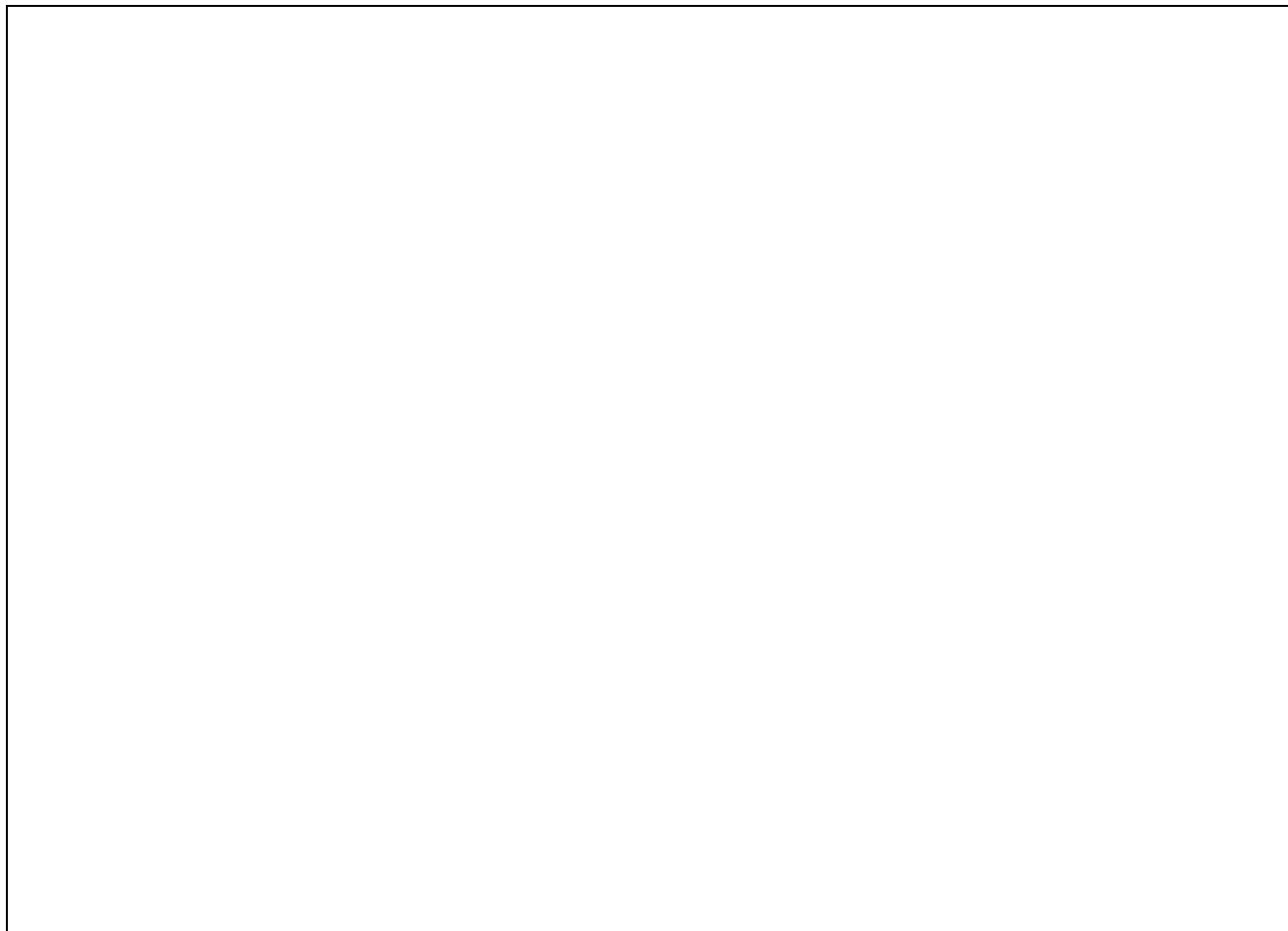
$$\text{Borrow} = \bar{A}B + BC + \bar{A}C \text{ or } \text{Carry} = \bar{A}B + C(\bar{A} \oplus B)$$

LAB TASK:

Design a logic circuit that acts as a full adder if selection bit **S** is low and when the selection bit **S** is high, the logic circuit acts as a full subtractor.

- a) Write truth table

- b)** Find the minimal SOP and POS expressions for the required logic circuit outputs using K-Map. Draw separate K-map for each output in the space given below:



- c)** Fill the following table9-5 in order to determine the gate cost for the implementation of required logic circuit using SOP expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 9-5: Gate Cost task1 part(b) SOP

- d) Fill the following table9-6 in order to determine the gate cost for the implementation of required logic circuit using POS expressions found in part (b)

IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 9-6: Gate Cost task1 part(b) POS

- e) Find Boolean expressions (mixed form) for the outputs. (**Hint:** Apply factoring on SOP expressions of the outputs)

- f) Fill the following table9-7 in order to determine the gate cost for the implementation of required logic circuit using expressions found in part (e)

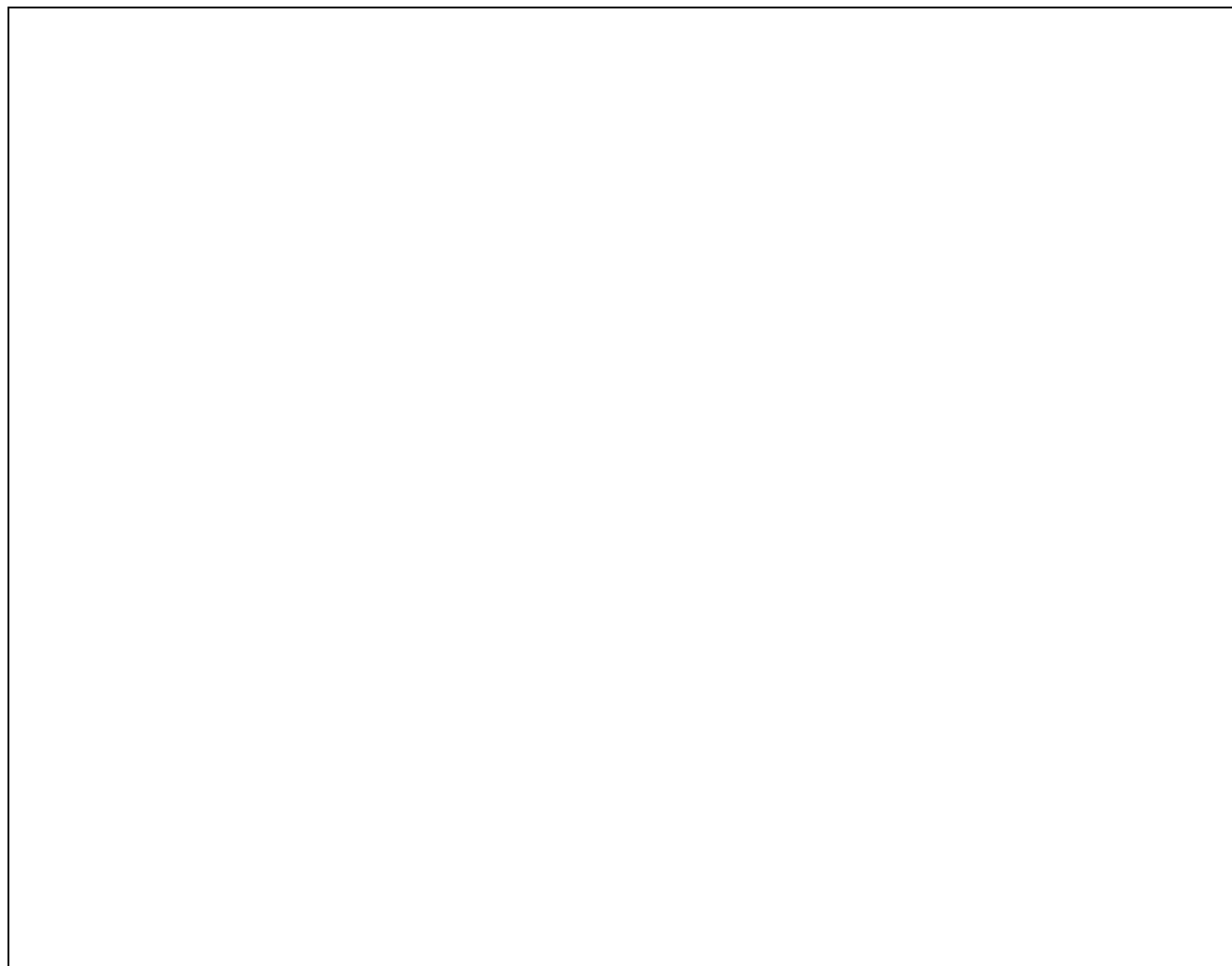
IC type	Required No. of Gates	Gates per IC	Required No. of ICs
Total no. of ICs			

Table 9-7: Gate Cost task1 part(e)

- g)** Implement the optimal solution. In the space given below, write the Boolean expressions you have chosen to be cost effective for the implementation of the logic circuit along with the logic diagram. Give proper reasoning for the chosen solution.

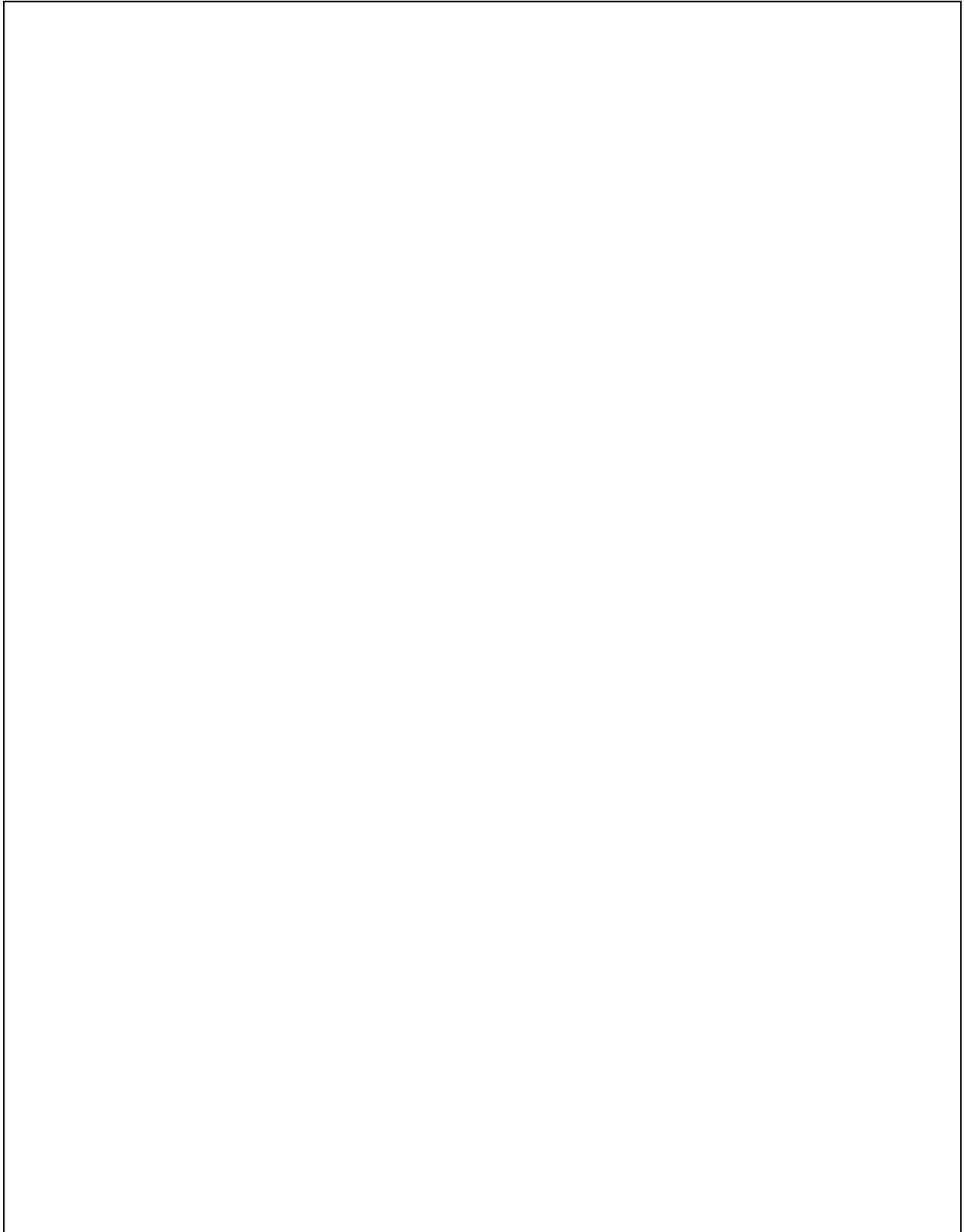
- h)** Suppose that the logic circuits for full adder and full subtractor are already constructed on logic trainer. If you are asked to add a functionality in this hardware that the user should be given a selection bit **S**. If **S** is low, only the outputs of full adder are displayed and if **S** is high then the outputs of full subtractor are shown. The user sees only two outputs at any instant. In the space given below, show through circuit diagram that how you would add the needed

functionality in this hardware instead of redesigning the whole circuit including selection bit **S**.



Post lab question:

Design a logic circuit that performs addition of two numbers **A** (3 bit) and **B** (4 bit) when the selection bit “**S**” is high. When “**S**” is low, the circuit performs subtraction of these two numbers. What design approach you think is appropriate here and why? Describe all the steps you would follow in order to design the required circuit.



EXPERIMENT#10

INTRODUCTION TO LOGIC WORKS

OBJECTIVE:

- To learn and study how to create and test combinational as well as sequential logic circuit using Logic Works

THEORY:

Logic Works 4 is a program used for designing and simulating circuits.

Start Logic Works by selecting it from the Microsoft Windows Start menu as shown in Fig. 10-1. Once the application is started several windows will appear on your screen:

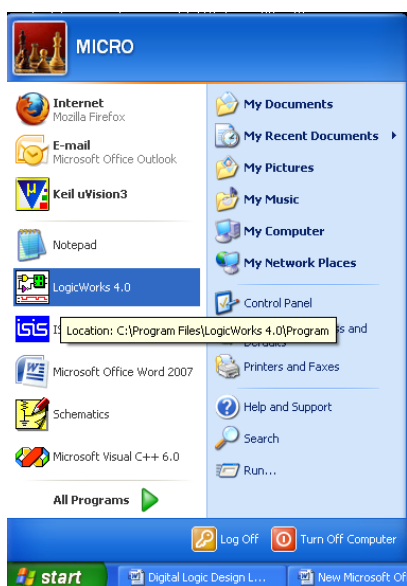


Figure 10-1

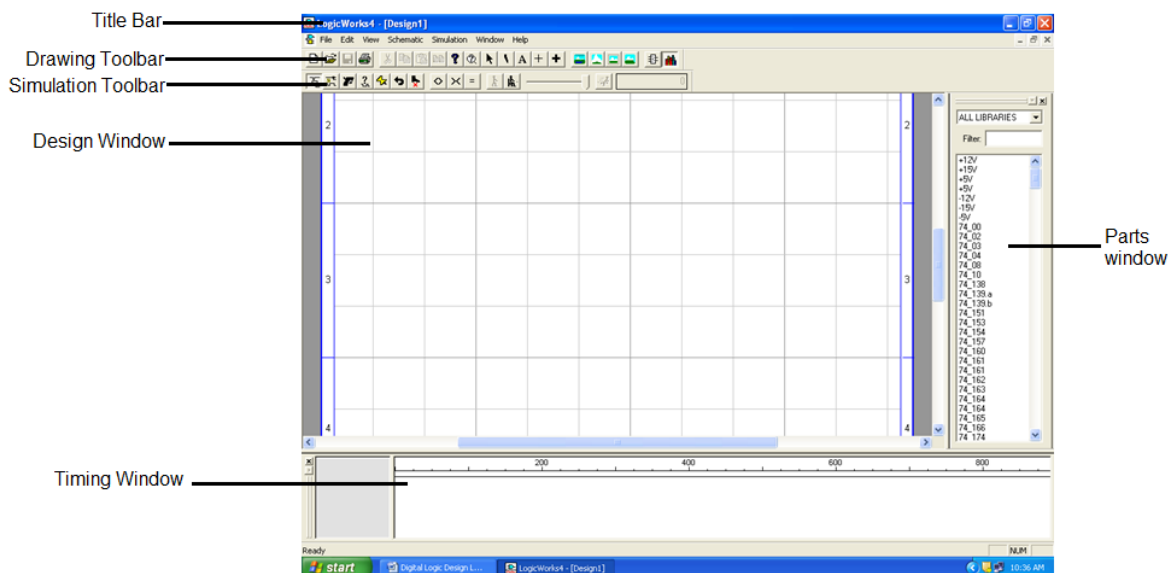


Figure 10-2

We will refer to these windows by the labels associated with them in Figure 10-2. *The Design Window* is the main window where circuits are drawn. Once the circuit is drawn, it is simulated and the output appears in the *Timing Window*. The *Logic Works Bar* is used to open existing designs and start useful tools. The *Parts Window* is used to select devices to place on the circuit design. Circuit designs are often referred to as schematics.

Let's Build a Circuit

In this tutorial, you will build and test a circuit that implements the following Boolean equation:

$$Z = AB + BC + AC'$$

This requires the following components:

1. Three AND gates with two inputs (AND-2)
2. Two OR gates with two inputs (OR-2)
3. A single NOT gate commonly called an inverter
4. Some wires to connect the gates
5. Three switches to provide a way to modify the input values for testing.

The first three components can be found in the SIMULATION GATES.CLF library:

- Click on the pull-down menu on the *Parts Window*
- Click on the SIMULATION GATES.CLF library.

SIMULATION GATES.CLF should now be displayed in the pull-down menu window of the *Parts Window*. The remainder of the *Parts Window* displays a list of the logic gates contained in this library. The scroll bar can be used to view them all.

Put the desired devices on the schematic (The *Design Window*):

AND Gates:

- Find the AND-2 gate in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Design Window* and place three of these gates where you want them by clicking once for each gate. See Figure 10-3.
Note: Spreading the gates apart a little makes it easier to connect parts to them later.
- Hit *Esc/Space* so that no more AND gates are selected.

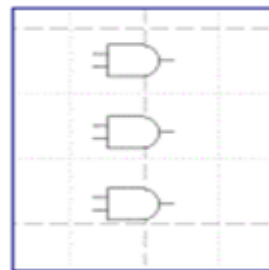


Figure 10-3

OR Gates:

- Find the OR-2 gate in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Design Window* and place two of these gates to the right of the AND gates. See Figure 10-4
- Hit *Esc/Space* again so no more OR gates are selected.

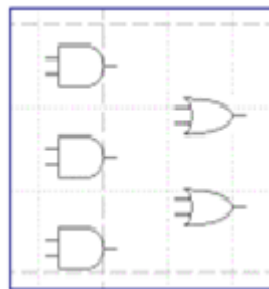


Figure 10-4

NOT Gates (Inverters):

- Find the NOT gate in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Design Window* and place one NOT gate to the left of the bottom AND gate. See Figure 10-5
- Remember to hit *Esc/Space* to deselect the NOT gate.

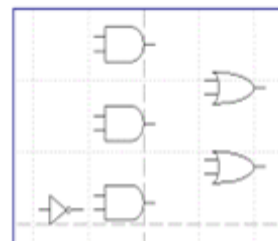


Figure 10-5

Moving and deleting existing gates:

By single clicking on an item on the schematic (The *Design Window*) the device will be selected and highlighted. While the device is selected, the *Delete* key will remove the item from the schematic. To move a device, point at it and hold down the left mouse button, then move the mouse to the desired location. Release the mouse button.

Adding switches:

- Click on the pull-down menu in the *Parts Window*
- Click on the ALL LIBRARIES library
- Double click on the binary switch entry and place three of them on the schematic on the left of your design as shown in Figure 10-6.

Your design should now resemble the schematic in Figure 10-7

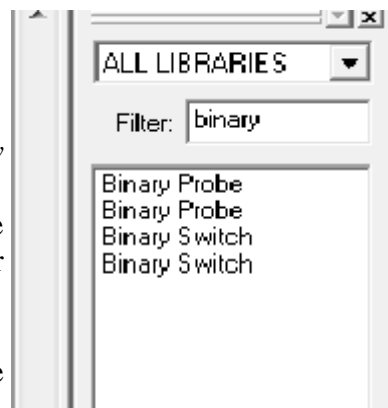


Figure 10-6

Connecting the Devices:

Adding wires:

- Place the cursor on the right edge of the switch and hold down the left mouse button.
- Drag the mouse a half-inch or so to the right and release the button. A red wire should now be attached to the switch, ending in the middle of nowhere. See Figure 10-7.
- Repeat this for each of the three switches.

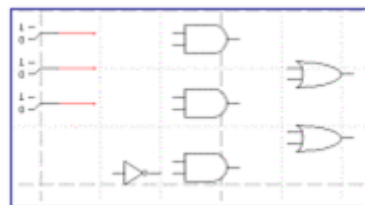


Figure 10-7

Recall that we are trying to implement $Z = AB + BC + AC'$. To clarify your design and show its relation to the equation above, we will label the wires A, B, and C.

Labeling wires:

- Right-click on the wire you want to name as shown in figure 10-8..
- Select *Name* from the box that appears, as in figure 10-8.
- Type the name of the wire in the text box. Be sure to check Visible as in Figure 10-9.

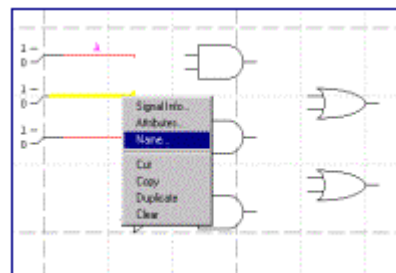


Figure 10-8

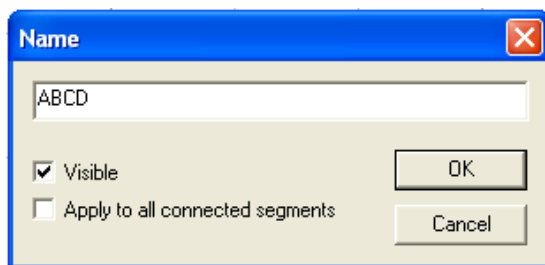


Figure 10-9

When you named these wires you should have noticed that in The Timing Window the names appeared. This will later show you the values of these three variables.

Connect the devices together with wires:

- Place the pointer on the end of the wire extending from switch A and hold down the left button, then drag the mouse to an input of the first AND gate. You have now connected the switch for input A to the first input of the AND gate.
- Connect the wire from switch B to the other input of the first AND gate in a similar manner.
- Click on the first input of the second AND gate.

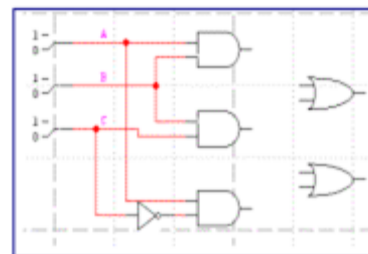


Figure 10-10

- Drag a wire to any point of the wire extending from switch B. A dot will appear on the intersection if the wires were successfully connected.
- Connect the C switch wire to the input of the inverter.
- Connect the output of this inverter to one of the inputs of the last AND gate and connect the other AND gate input to switch A. Your design should now resemble the schematic in Figure 10-10.

Note: The wire will go around one corner for you, but you may have to stop in the middle of the schematic with one wire and then restart to go in another direction. In addition, an intersection without a dot is simply two wires crossing without making a connection.

- Connect the outputs of two of the AND gates to the inputs of one of the OR gates.
- Finally we connect the output of the remaining AND gate and the output of the first OR gate to the input of the last OR gate.
- A short wire should be connected to the output of the final OR gate and this wire should be labeled Z. See Figure 10-11. Notice that the output Z showed up in the *Timing Window*.

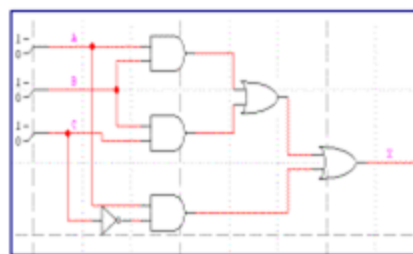


Figure 10-11

Simulation

Notice that in the *Timing Window* that there are small lines drawn like underscores for inputs A, B, and C. These are low because the switches are all set to the zero value.

Starting the Simulator:

- Select *Simulation* from the Logic WorksBar.
- Click *Reset Simulation*. This clears the *Timing Window*.
- Click *Run* to start the simulation. The *Timing Window* is now active and records the values of the inputs and outputs of your circuit.

Testing your Circuit:

- Click on the handle of switch A to point to the value 1.

Notice that the A line in the *Timing Window* went up to a high level, indicating the value 1. Also, notice that the Z output went high. This is because A is high and C' is high and according to the equation Z should be high with these input values.

- Change the values of A, B, and C and observe the results in the Timing Window.
- Click on the >< and <> buttons and observe that they affect the time scale of the Timing window.
- Click on the *Signal Probe* tool in the Tool Palette.



- Click the tip of the probe tool along any signal line. It will show the current value of the signal as the simulation progresses.
- The *Binary Probe* is a device for displaying the level present on any signal line. Any change in the signal state is shown on the probe. Possible displayed values are 0(low), 1 (high), Z (High Impedance) and X (unknown)

Saving your work:

Of course using the file menu in The Design Window can save all of your work. Using the file menu in the Logic Works Bar when the program is first started can open previous designs.

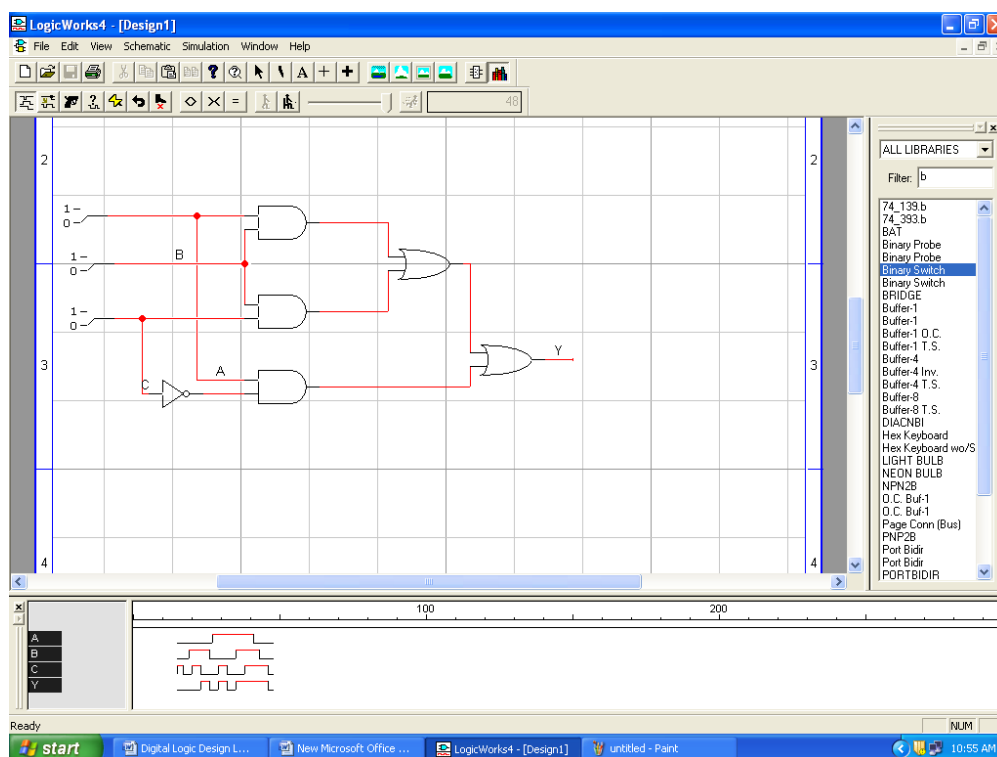


Figure 10-12

Your timing diagrams should resemble like as shown in figure 10-12.

Building a Subcircuit:

There are a variety of ways of creating a subcircuit and attaching it to a symbol for simulation purposes. Following is a simple procedure:

- Make a Logic Works[®] circuit and make sure that it works before proceeding any further. In this example we will implement the function:

$$F = A'B + AB'$$

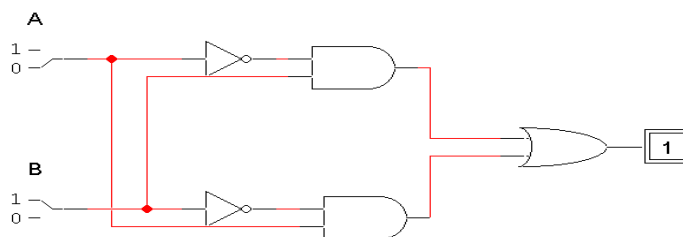


Figure 10-13

A subcircuit is basically the old circuit in figure 10-13 and **port connectors** connected to its inputs and outputs. Port connectors can be found in the **connect.clf** library. There are two types of port connectors: "Port In" and "Port Out" which are connected to the inputs and outputs, respectively, of the subcircuit.

- The port connectors should also be given names. Make sure that these names are distinct from each other as well as from other names commonly found on other devices. Also, make sure that the names are attached to the ports by clicking the names and seeing if both the names and ports are highlighted together as shown in figure 10-14.
- Keep the window of the subcircuit open.

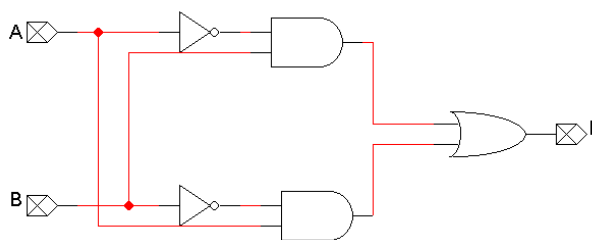


Figure 10-14

- Create and load your own device library by right clicking on the part palette and selecting new lib. Give a distinct name to your library. See figure 10-15.

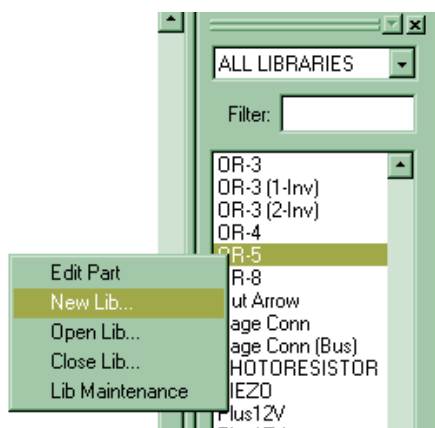


Figure 10-15

- Now to create the subcircuit go to File > New> Device Symbol, this will open the Device Editor window. In the Options menu in the *DevEditor* window select the “Subcircuit/Part Type” command. Select the second option "Create a sub-circuit symbol and select an open circuit to attach to it". When you click on this option, a box will appear containing a list of all open circuits. Select the one you just created. See figure 10-16.

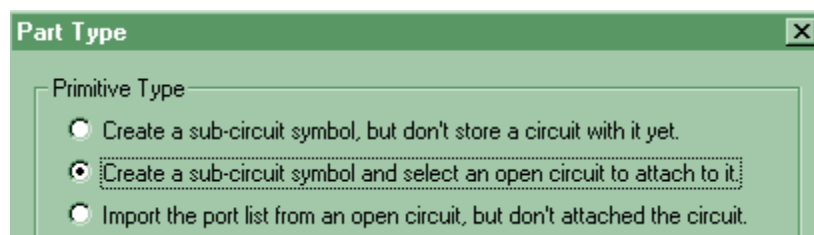


Figure 10-16

- Click the Done button on the Subcircuit/Part Type box. You should now see the three pin names in the pin list on the left. See figure 10-17.



Figure 10-17

- Select the “Autocreate Symbol” command in the DevEditor's Options menu.
- Click the Extract Pin List button. This will place the three pin names in the appropriate boxes.
- Click the Generate button. This will create a symbol. You can modify the symbol graphics if you want. See figure 10-18.

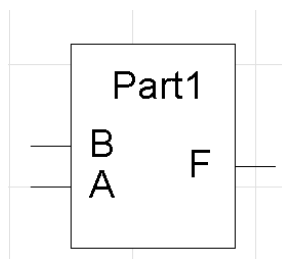


Figure 10-18

- Select the Save As command in the DevEditor's File menu and save the new part to your own library.
- You can now use this subcircuit as a device in your circuits as shown in figure 10-19.

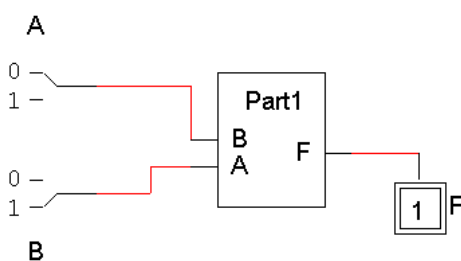


Figure 10-19

D Flip Flop:

- Find the D flip flop in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Design Window* and place it where you want it by clicking. See figure 10-20 given below:

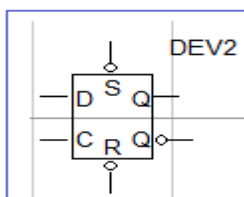


Figure 10-20

This figure above shows a D flip flop with active low asynchronous set and reset inputs. The asynchronous set and reset inputs are used for setting and resetting the flip flop independent of the clock input. These asynchronous inputs are useful for bringing the flip flops in a digital system to an initial state prior to the normal clocked operation of the system. The D input is the same in Logic Works as it is on your chip. The C input represents the clock pin. The Q and Q' outputs represent the standard and complemented outputs. The S and R lines represent the set and reset asynchronous inputs of the flip flop.

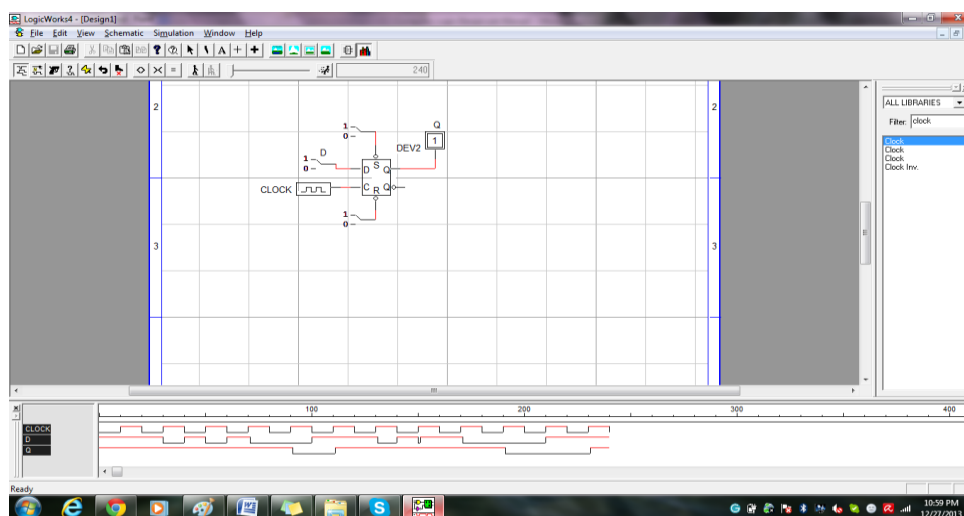


Figure 10-21

In order to get standard output Q of D flip flop according to its characteristic table we need to connect binary switch with D input and clock at C pin. However, the set and reset pin should be high.

The timing diagram in the figure 10-21 above shows that the D flip flop in Logic Works 4 is a positive edge triggered D flip flop.

LAB TASK#1:

Implement a logic circuit in Logic Works 4 that performs multiplication of two 2-bit numbers using 74LS153(dual 4x1 MUX) ICs and basic logic gates.

- a) In the space given below, paste the circuit diagram that you have implemented in Logic Works 4.

b) In the space given below, paste the timing diagram (created in Logic Works 4) to show the working of required 2-bit multiplier.

LAB TASK#2:

Implement the logic sequential circuit from figure 10-22 in Logic Works 4.

Note: Create your own 2x1 MUX and then use it for the implementation of the following circuit.

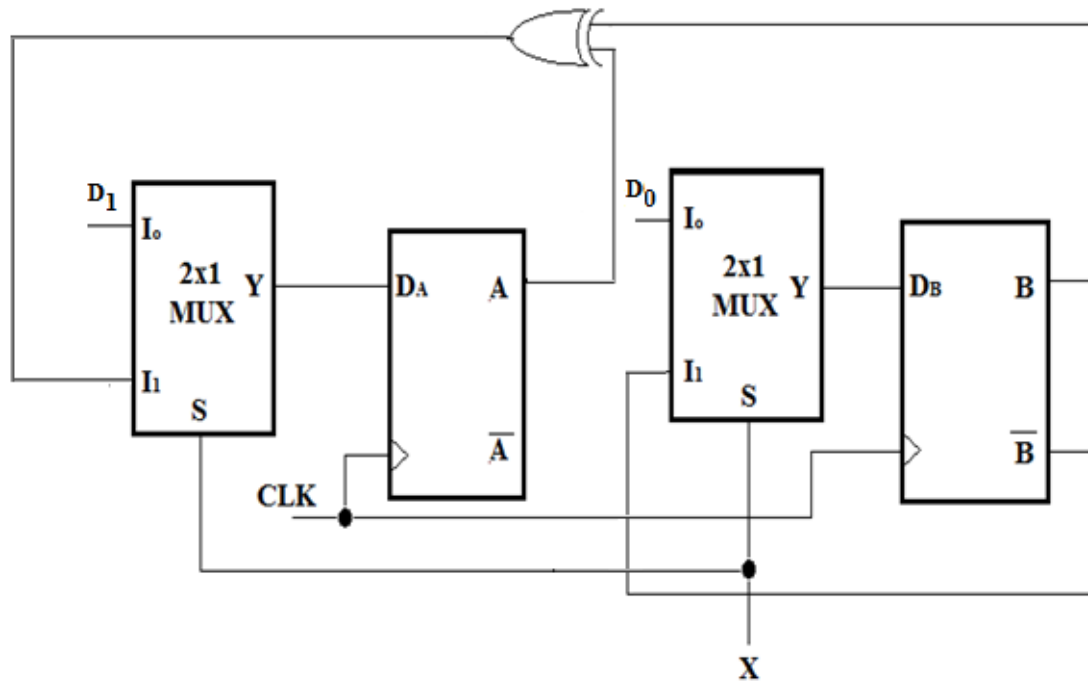
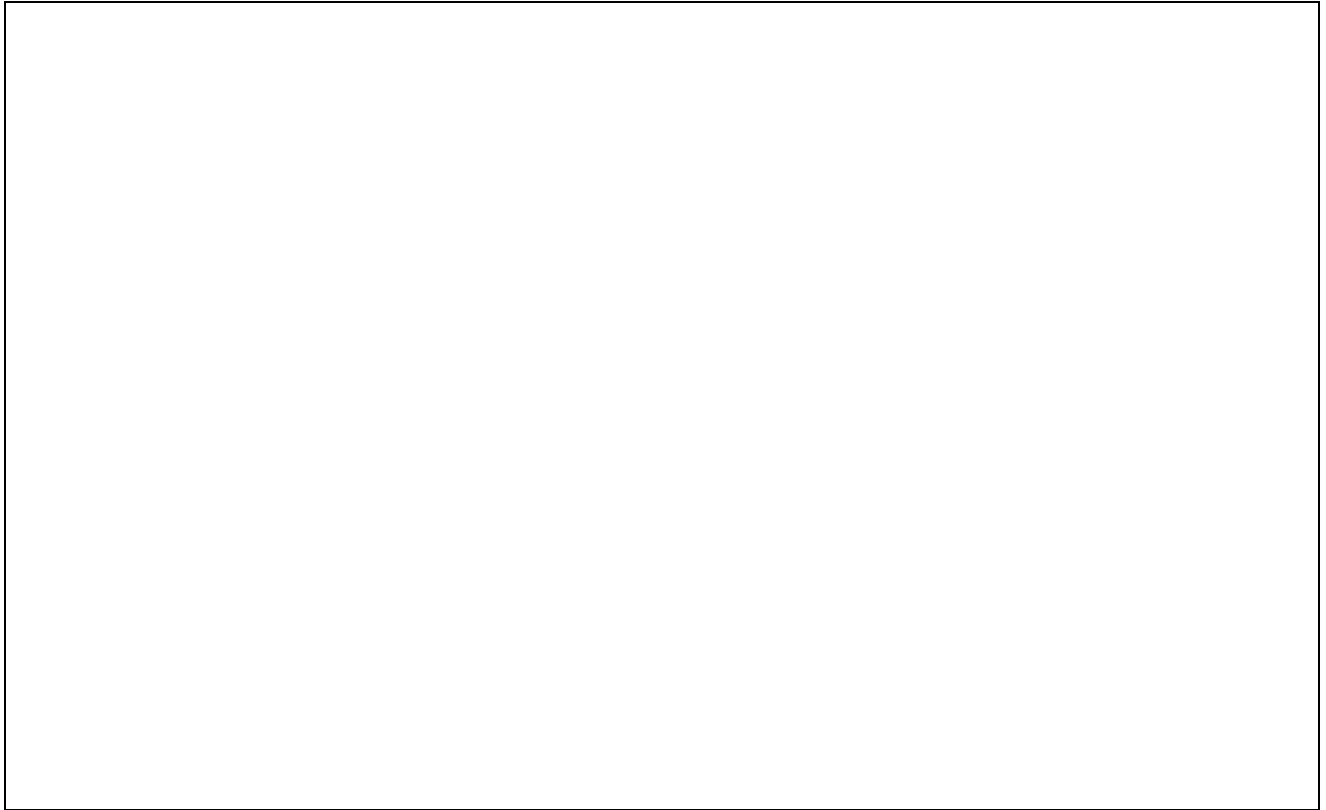


Figure 10-22: Sequential logic Circuit

a) In the space given below, paste the timing diagram (created in Logic Works 4).

b) Write state table.

c) Draw state diagram.



EXPERIMENT#11

DESIGN & ANALYSIS OF SEQUENTIAL CIRCUITS

OBJECTIVE:

- To learn how to design and implement sequential circuits using flip flops

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86, 74LS02, 74LS74, 74LS153

THEORY:

The design of clocked sequential circuits starts from a set of specifications and culminates in a logic diagram or a list of Boolean functions from which the logic diagram can be obtained. **A synchronous sequential circuit is made up of flip-flops and combinational gates.** The design of circuits consist of choosing the flip flops and finding combinational circuit structure which, together with the flip flops, produces a circuit that fulfills the stated specifications. The design procedure for sequential circuits is as follows:

- 1) First, obtain a state diagram, and then obtain the state table from it.
- 2) Assign binary codes to the states.
- 3) Derive the flip flop input equations from the state table.
- 4) Derive the output equations from the state table.
- 5) Simplify both input and output equations.
- 6) Draw the logic diagram with the flip-flops and combinational gates, as specified by the input equations and output equations.

A flip flop is usually constructed by combining two same or different types of latches. Flip-flop circuits are constructed in such a way as to make them operate properly when they are a part of a sequential circuit that employs a single clock. **The key to the proper operation of flip flops is to prevent them from being transparent.** In a flip-flop, before an output can change, the path from its inputs to its outputs is broken. Therefore, a flip-flop cannot “see” the change of its output or of the outputs of other, like flip flops at its input during the same clock pulse. Thus, the new state of a flip flop depends only on the immediately preceding state, and the flip-flops do not go through multiple changes of state. There are different types of flip flops e.g. D flip flop, T flip flop, JK flip flop etc. The most commonly used flip flop is D flip flop.

D-Flip Flop:

D flip flops are considered as one-bit memory. The D stands for "data"; this flip-flop stores the value that is on the data line.

Characteristic Table:

The characteristic table is the truth table in which inputs being the control bit(s) of the flip flop and Q, and the output being Q+. The characteristic table is show in table 11-1:

Characteristic Table of D flip flop:

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

Table 11-1: D flip flop Characteristic table

Boolean Expression of Next State:

$$Q(t + 1) = D$$

Excitation Table:

The excitation table rearranges the order of characteristic table. Instead of the truth table inputs being the control bit(s) of the flip flop and Q, and the output being Q+, the truth table inputs are now Q and Q+, and the "outputs" is the control bit(s) of the flip flop. The excitation table is shown in table 11-2

Excitation table of D Flip Flop:

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Table 11-2: D flip flop Characteristic table

LAB TASK#1:

Construct T flip flop using D flip flop.

- a) Write state table

- b) Draw logic diagram

LAB TASK#2:

Design and implement hardware for a 2-bit array with selection bit **S**. If **S** is zero, then load a 2-bit data in 2-bit array. If **S** is one, then swap the bit values with each other, e.g. if at an instant, **S** is one and the 2-bit array contains data:

0	1
---	---

then new state of array should be:

1	0
---	---

a) Draw state diagram

b) Write state table

- c) In the space given below, write the Boolean expressions for above implementation along with the logic diagram. (Use K-maps if required)

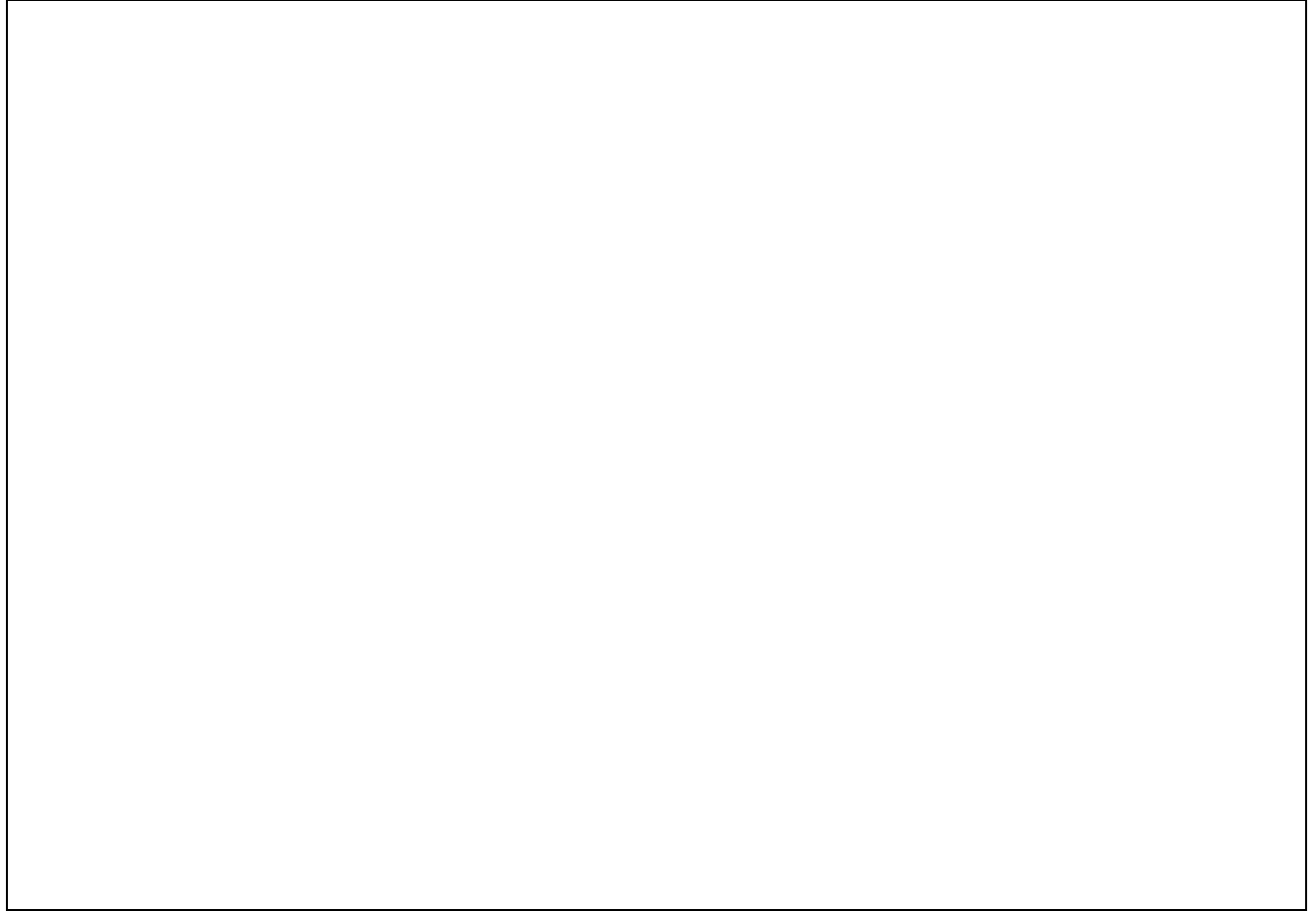
LAB TASK#3:

Design and implement a sequential logic circuit that detects the sequence '101'.

a) Draw state diagram

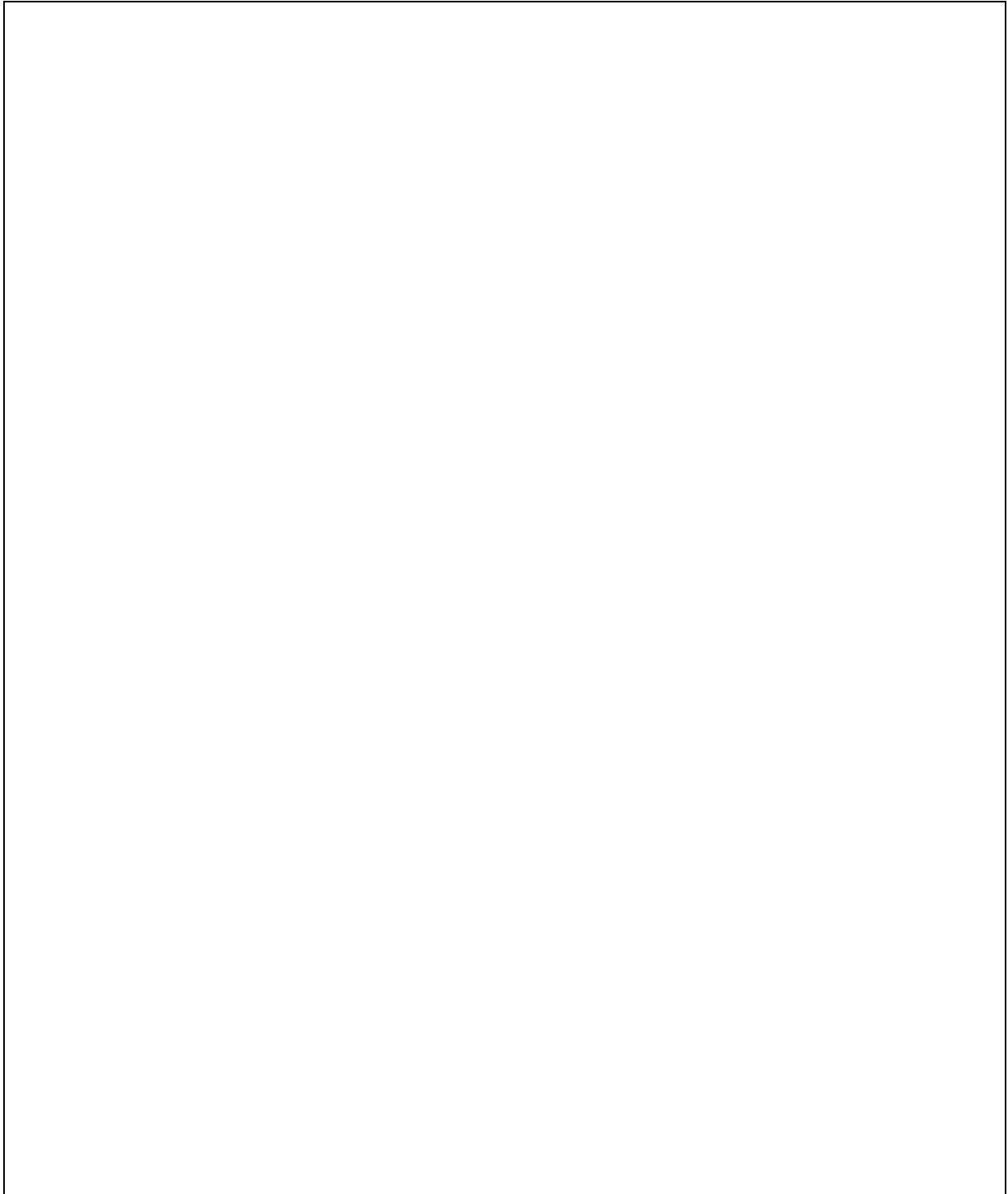
b) Write state table

c) In the space given below, write the Boolean expressions for above implementation along with the logic diagram. (Use K-maps if required)



Post lab question:

Implement the sequence detector that detects the sequence '101' in Logic Works 4. Give schematic diagram and timing diagram created in Logic Works 4.



Experiment#12**DESIGN OF REGISTERS****OBJECTIVE:**

- To learn how to design and implement registers using flip flops

EQUIPMENT: Logic trainer, Logic probe

COMPONENTS: ICs 74LS08, 74LS32, 74LS04, 74LS86, 74LS02, 74LS74, 74LS153

THEORY:

In this lab, we will design a combinational circuit of register. A register is used to store n-bits of information, where n is the number of flipflops. A register consists of a set of flip flops, together with gates that perform data processing tasks. The flip flops hold data, and the gates determine the new or transformed data to be transferred into the flip flops. The registers have two types, one simple register and other register with parallel load. The simple register is discussed above. The register with parallel load is the register in which we can easily store the value of our own choice.

This ability of register is controlled by a control input, if control input is 1 then the data that we want to enter is stored on the register, and when the value is 0 then the data that is already stored in the register remains unchanged. Another type of register is known as shift register.

The shift register is capable of shifting its stored bits laterally in one or both direction. The logical configuration of a shift register consists of a chain of flip flops in cascade, with the output of one flip flop connected to the input of the next flip flop. All flip flops receive a common clock pulse, which activates the shift from each stage to the next.

LAB TASK#1:

Design and implement a 2-bit bidirectional shift register with selection bit S according to the following function table 12-1:

S	Operation
0	Shift left
1	Shift right

Table 12-1: Register Function Table

a) Write state table

b) In the space given below, write Boolean expressions for the above implementation along with the logic diagram. (Use K-maps if required)

LAB TASK#2:

Design and implement a 2-bit register, with selection bits S_0 and S_1 , according to the following function table 12-2:

S_1	S_0	Operation
0	0	Preserve value or No change
0	1	Parallel Load
1	0	Shift left
1	1	Shift Right

Table 12-2: Register function Table

a) Write state table

- b)** In the space given below, write Boolean expressions for the above implementation along with the logic diagram. (Use K-maps if required)

- c) Can we implement the required 2-bit register using the hardware already constructed for bidirectional shift register? If yes, then what additional hardware is required? Show logic diagram.

Appendix A: Guidelines for Lab Evaluation

Labs with projects

1. Class Participation	10%
2. Lab Work	40%
3. Quiz (2)	20%
4. Project	30%
a. Project Demonstration	20%
b. Project Report	5%
c. Project Quiz	5%

Labs without projects

1. Class Participation	10%
2. Lab Work	40%
3. Quiz (2)	20%
4. Lab Final	30%
a. Lab Final (Practical)	20%
b. Lab Final (Written)	10%

Notice:

Copying and plagiarism of lab reports is a serious academic misconduct. First instance of copying may entail ZERO in that experiment. Second instance of copying may be reported to DC. This may result in awarding FAIL in the lab course.

Appendix B: Safety around Electricity

In all the Electrical Engineering (EE) labs, with an aim to prevent any unforeseen accidents during conduct of lab experiments, following preventive measures and safe practices shall be adopted:

- Remember that the voltage of the electricity and the available electrical current in EE labs has enough power to cause death/injury by electrocution. It is around 50V/10 mA that the “cannot let go” level is reached. “The key to survival is to decrease our exposure to energized circuits.”
- If a person touches an energized bare wire or faulty equipment while grounded, electricity will instantly pass through the body to the ground, causing a harmful, potentially fatal, shock.
- Each circuit must be protected by a fuse or circuit breaker that will blow or “trip” when its safe carrying capacity is surpassed. If a fuse blows or circuit breaker trips repeatedly while in normal use (not overloaded), check for shorts and other faults in the line or devices. Do not resume use until the trouble is fixed.
- It is hazardous to overload electrical circuits by using extension cords and multi-plug outlets. Use extension cords only when necessary and make sure they are heavy enough for the job. Avoid creating an “octopus” by inserting several plugs into a multi-plug outlet connected to a single wall outlet. Extension cords should ONLY be used on a temporary basis in situations where fixed wiring is not feasible.
- Dimmed lights, reduced output from heaters and poor monitor pictures are all symptoms of an overloaded circuit. Keep the total load at any one time safely below maximum capacity.
- If wires are exposed, they may cause a shock to a person who comes into contact with them. Cords should not be hung on nails, run over or wrapped around objects, knotted or twisted. This may break the wire or insulation. Short circuits are usually caused by bare wires touching due to breakdown of insulation. Electrical tape or any other kind of tape is not adequate for insulation!
- Electrical cords should be examined visually before use for external defects such as: Fraying (worn out) and exposed wiring, loose parts, deformed or missing parts, damage to outer jacket or insulation, evidence of internal damage such as pinched or crushed outer jacket. If any defects are found the electric cords should be removed from service immediately.
- Pull the plug not the cord. Pulling the cord could break a wire, causing a short circuit.
- Plug your heavy current consuming or any other large appliances into an outlet that is not shared with other appliances. Do not tamper with fuses as this is a potential fire hazard. Do not overload circuits as this may cause the wires to heat and ignite insulation or other combustibles.
- Keep lab equipment properly cleaned and maintained.
- Ensure lamps are free from contact with flammable material. Always use lights bulbs with the recommended wattage for your lamp and equipment.
- Be aware of the odor of burning plastic or wire.

- ALWAYS follow the manufacturer recommendations when using or installing new lab equipment. Wiring installations should always be made by a licensed electrician or other qualified person. All electrical lab equipment should have the label of a testing laboratory.
- Be aware of missing ground prong and outlet cover, pinched wires, damaged casings on electrical outlets.
- Inform Lab engineer / Lab assistant of any failure of safety preventive measures and safe practices as soon you notice it. Be alert and proceed with caution at all times in the laboratory.
- Conduct yourself in a responsible manner at all times in the EE Labs.
- Follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ASK YOUR LAB ENGINEER / LAB ASSISTANT BEFORE PROCEEDING WITH THE ACTIVITY.
- Never work alone in the laboratory. No student may work in EE Labs without the presence of the Lab engineer / Lab assistant.
- Perform only those experiments authorized by your teacher. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed.
- Be prepared for your work in the EE Labs. Read all procedures thoroughly before entering the laboratory. Never fool around in the laboratory. Horseplay, practical jokes, and pranks are dangerous and prohibited.
- Always work in a well-ventilated area.
- Observe good housekeeping practices. Work areas should be kept clean and tidy at all times.
- Experiments must be personally monitored at all times. Do not wander around the room, distract other students, startle other students or interfere with the laboratory experiments of others.
- Dress properly during a laboratory activity. Long hair, dangling jewelry, and loose or baggy clothing are a hazard in the laboratory. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured. Shoes must completely cover the foot.
- Know the locations and operating procedures of all safety equipment including fire extinguisher. Know what to do if there is a fire during a lab period; "Turn off equipment, if possible and exit EE lab immediately."