

Digital Logic Design

(EL-227)

Spring-2021



LAB 4

Week 4

Software based

Implementation of Logic gates

,expressions and truth table

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Date:03/01/2021

Lab Session 4: Implementation of logic gates

OBJECTIVES:


The objectives of this lab is:

- Start up Logic Works 5
- Build and simulate circuits in the design/schematic window

Introduction:

Logic-Works is an interactive circuit design tool intended for teaching and learning digital logic. It is a program that we can use for designing and simulating circuits. This lab material will introduce you the basic features of Logic-Works and familiarize you with the program by stepping you through the construction of a simple circuit.

Start up Logic-Works:

You start Logic Works by clicking on the icon  or selecting it from the Microsoft Windows Start menu. Once the tool is open, you will see the "Welcome to the Logic-Works" screen.

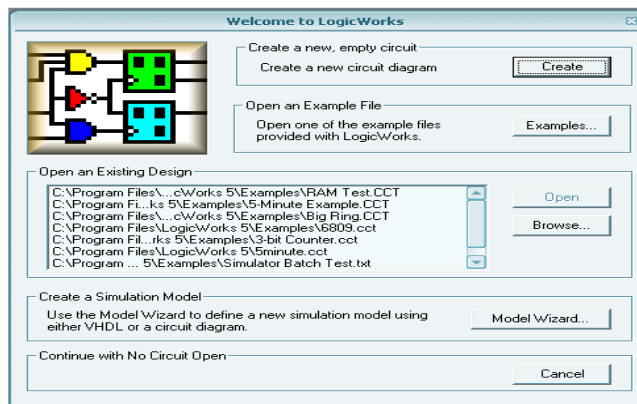


Figure 1: Start-up screen

Select "Create a new circuit diagram" by clicking on the "Create" button. Then the following screen is displayed.

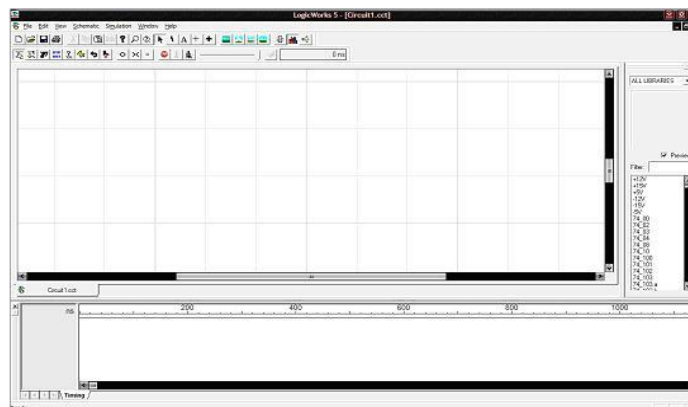


Figure 2: Main Panel

There are three primary window components in Logic-Works.

1. Circuit Window
2. Timing Window
3. Parts Palette.

1. **Circuit Window:**

The circuit window is where you build your circuits. For example, you can select different parts from the Parts Palette and place the device in the circuit Window to build your circuits. The following is a screen capture of the circuit window with a sample circuit already drawn. You can select any part of the circuit by left clicking on it, once selected, a right click can bring up a pop menu that displays the options for manipulating the selected component

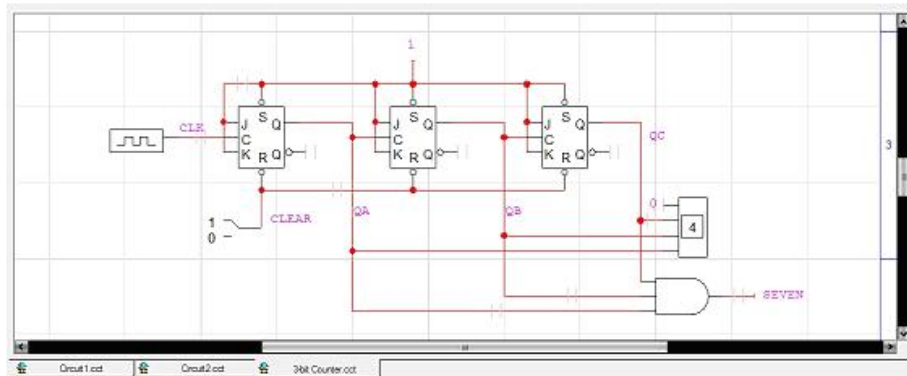


Figure 3: Simple Circuit Diagram

2. **Timing Window:**

When a circuit is simulated, the Timing diagram window can be displayed to show signal values versus time. Only one Timing window can be displayed at any given time, and it gives waveforms generated by the current design. Closing the Timing window does not close the circuit design file. Like the Circuit window, the intervals of the timing diagram can be selected by using click and drag the mouse over the desired interval. Once a section of the diagram is selected, right click brings up a pop menu with options that can be used to manipulate the Timing diagram.

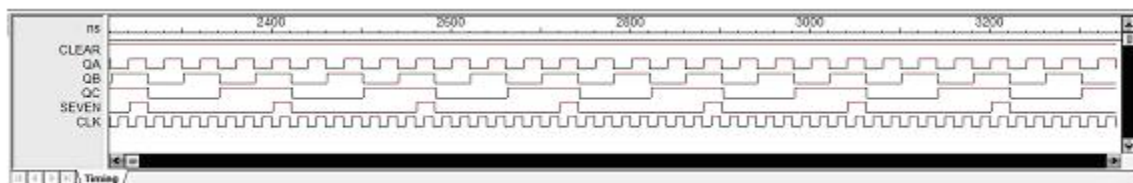


Figure 4: Simple Timing Diagram Example

3. Parts Palette:

Parts Palette is another very important part of Logic-Works interface. By default, it is located to the right of the Circuit Window. The location can be changed by holding down the left mouse button on the double striped title bar near the top and dragging the window to the desired location. Usually, I would recommend that you keep it in the default location.

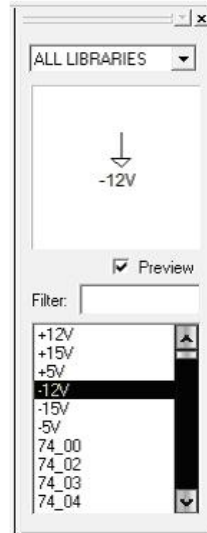


Figure 5: Parts Palette

Building a Simple Circuit:

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

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

This requires the following components

- Three AND gates with two inputs (AND-2)
- Two OR gates with two inputs (OR-2)
- A single NOT gate commonly called an inverter
- Three switches to provide a way to modify the input values for testing
- A binary probe to show the results of the circuit

The first three components can be found in the "Simulation Gates.clf" library:

1. Click on the pull-down menu on the Parts Window
2. Click on the "Simulation Gates.clf" library.

"Simulation Gates.clf" should now be displayed in the pull-down menu window of the *Parts Window* as shown in Fig. 1. The next portion 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.

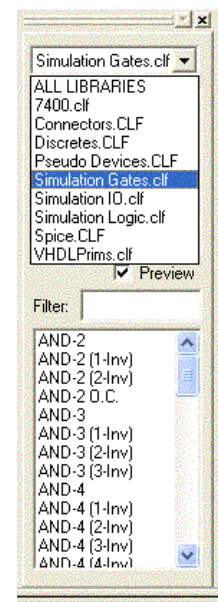


Figure 6: Selection of the libraries

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

AND Gates:

- Find the AND-2 gate in the Parts Window and double click on it.
- Move the mouse pointer to the Circuit Design Window and place three of these gates where you want them by clicking once for each gate
- Hit Esc/Space so that no more AND gates are selected.

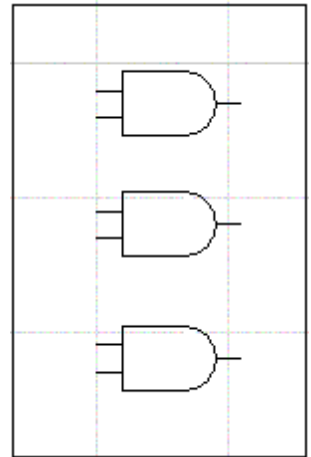


Figure 7: AND Gate

OR Gates:

- Find the OR-2 gate in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Circuit Design Window* and place two of these gates to the right of the AND gates.

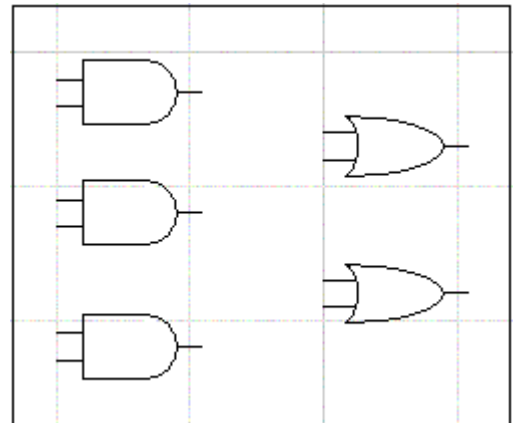
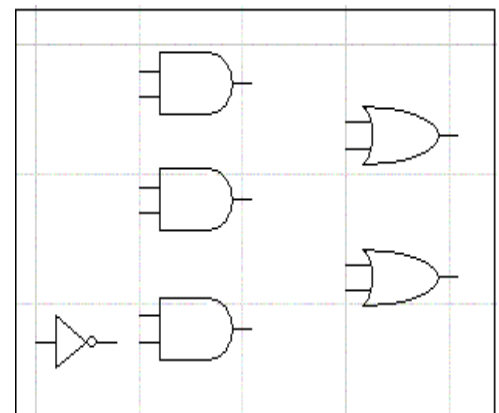


Figure 8: Or Gate Placement

NOT Gates (Inverters):

- Find the NOT gate in the *Parts Window* and double click on it.
- Move the mouse pointer to the *Circuit Design Window* and place one NOT gate to the left of the bottom AND gate.



Moving and deleting existing gates:

- By single clicking on an item on the schematic (The Circuit Design Window) the device will be selected and highlighted. While the device is selected, the Delete key will remove the item from the circuit window.
- 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 "Simulation IO.clf" library.
- Double click on the binary switch entry and place three of them on the circuit window on the left of your design.
- Double click on the Binary Probe entry and place it on the circuit window on the right of your design.

Your design should now resemble the circuit window in Fig. 5 below.

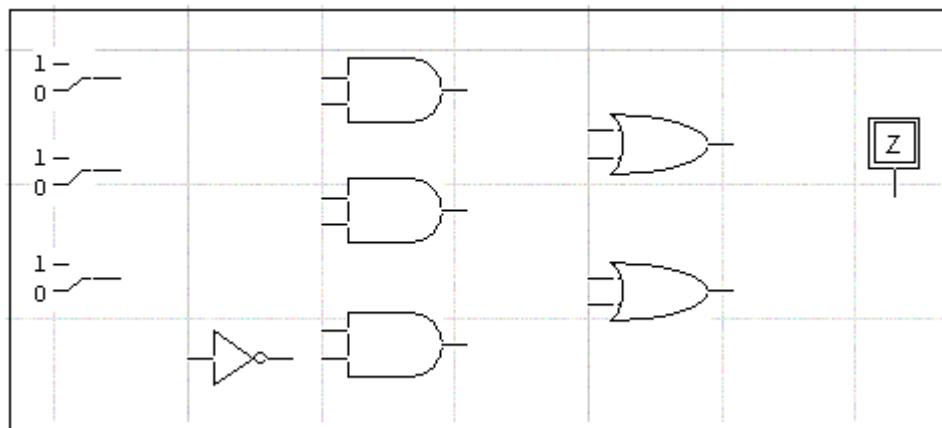


Fig. 10 Binary Switch and Probe Placement

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.
- Repeat this for each of the three switches. Right-click on the wire you want to name.
- Select Name from the box that appears.
- Type the name of the wire in the text box. Be sure to check Visible as in Fig

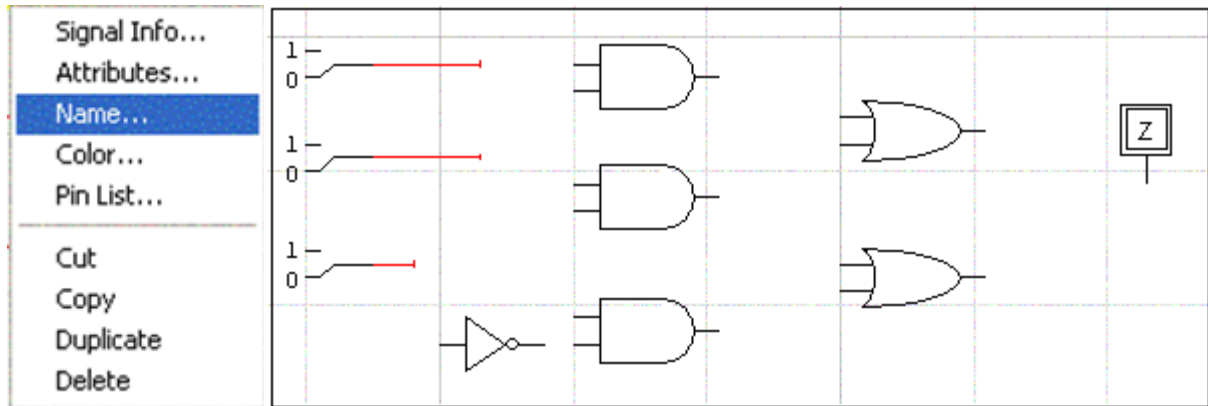



Fig. 10 Wire Connection and labeling

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, C, and Z.

This process can also be done by clicking on the button **A** on the top tool bar.  The mouse pointer then looks like a little pencil. Point this pencil at one of the wires (remember only the red parts are wires, if you click on a black part you are labeling a pin) and click on it. A text box appears and you can type a name for the wire. Hit *Esc/Space* to exit the name process.

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 other input of the first AND gate in a similar manner.
- Click on the first input of the second AND gate.

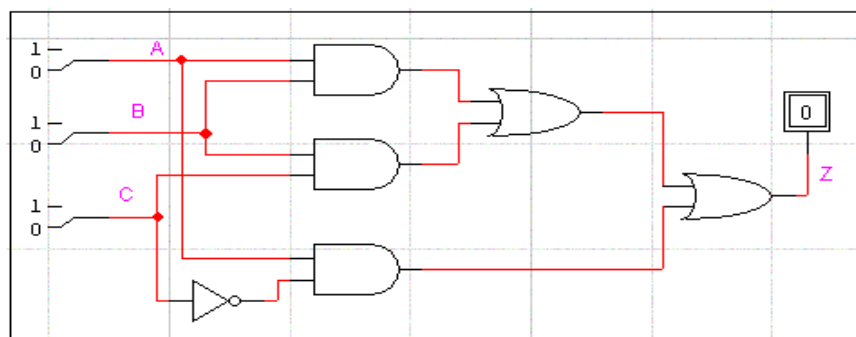


Fig. 11 Connection

- 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 circuit window in Fig.11

Simulation Controls

Here is the Simulator Toolbar given below:



Position your mouse pointer on each button, you will find the function of each of them. You may try it on your own.

For example,

1. Clicking on the \diamond or \times button can adjust the range of the time values to suit the display data.
2. Clicking on the Reset button can make the simulation restart at time 0.
3. Clicking on the Run button can start the simulation.

You may have noticed 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.

Testing your Circuit

- Click on the handle of switch A to point to the value 1.
- Change the values of A, B, and C and observe the results in the Timing Window.
- Click on the button \diamond in the Simulator toolbar. This will stretch the time out, making the waveforms easier to see.

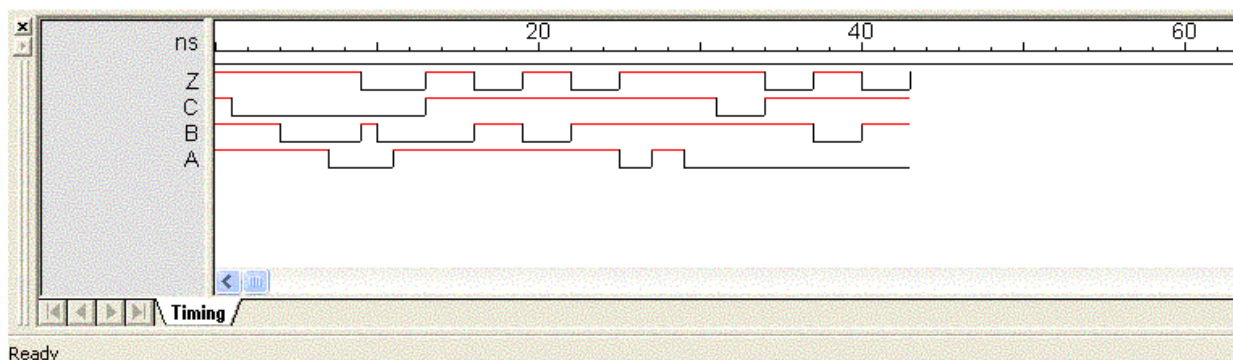


Fig. 12 Timing window

Inserting Text in the Circuit Window:

To insert the text in the circuits window, click on the button A in the design toolbar , then follow the following steps.

- 1.Position your pencil (cursor) at the desired location.
2. Type in the text
3. Change cursor to pointer by clicking on the pointer in the toolbar or press space bar or escape key.
- 4.Click on the text entered, then right click.
- 5.Select "Text Style" and then select the preferred options and then, press ok.



Saving and printing your work:

To save or print your work, click the File menu and then select the proper option. The following picture illustrates the idea.

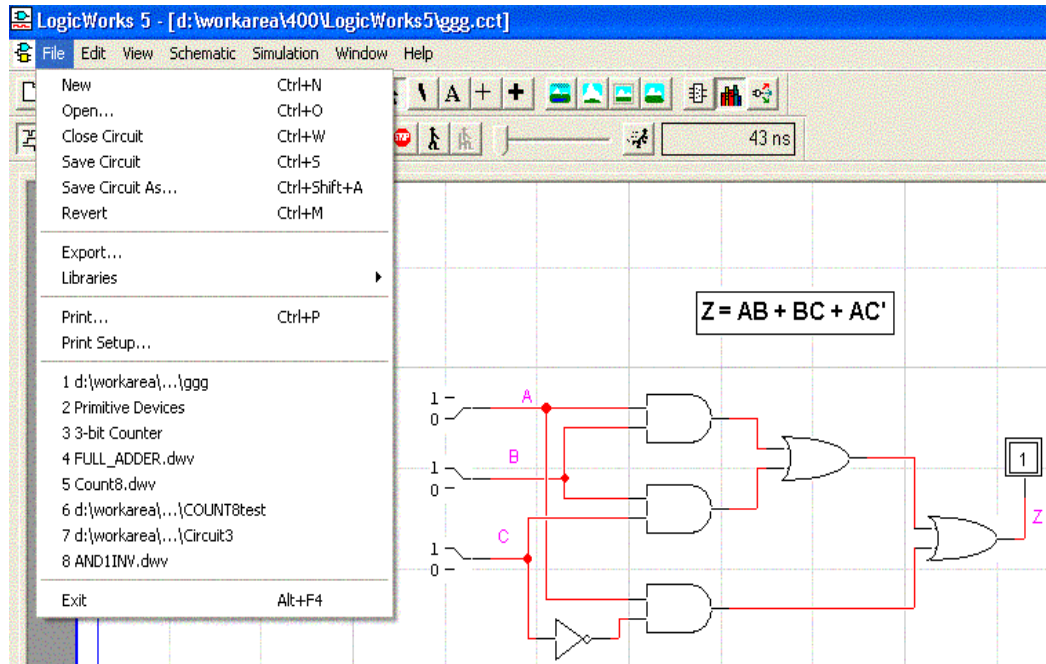


Fig. 13 Print and save option

XOR

XOR

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

XOR Symbol



Simplified expression

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

3 input XOR gate truth table

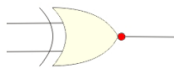
A	B	C	Y (ODD 1'S)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

XNOR

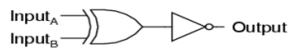
The XNOR function is the inverse of the XOR function. Since the output of a 2-input XNOR is asserted when both inputs are the same, it is sometimes referred to as the Equivalence function (EQV), but this name is misleading because it does not hold for three or more variables (i.e., the output of a 3-input XNOR is not asserted whenever all three inputs are the same). Truth tables for 2 and 3 input XNOR functions are shown in Fig. 2, and it can be seen that for each combination of inputs, the output is the inverse of the XOR truth tables above.

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

The symbol of the XNOR gate:



Equivalent gate circuit



Expression

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

3 input XNOR gate truth table

A	B	C	Y (EVEN 1'S) =0/P HIGH
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fig 2

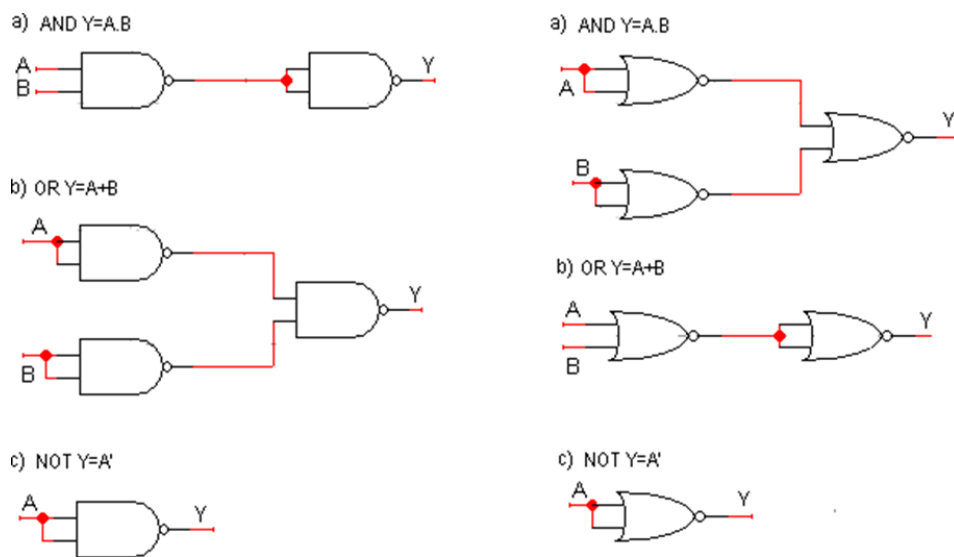
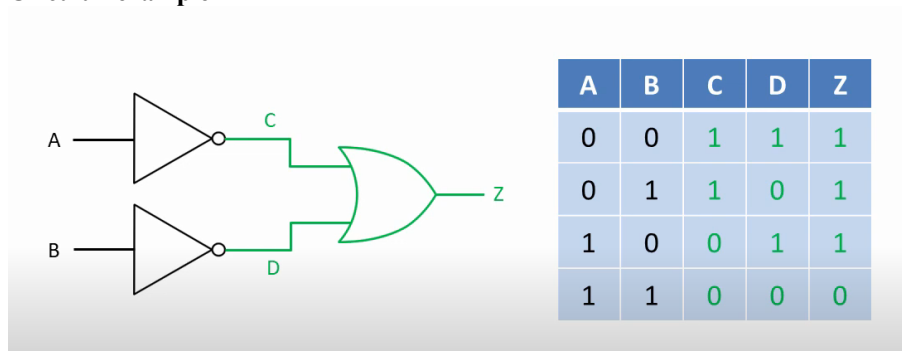
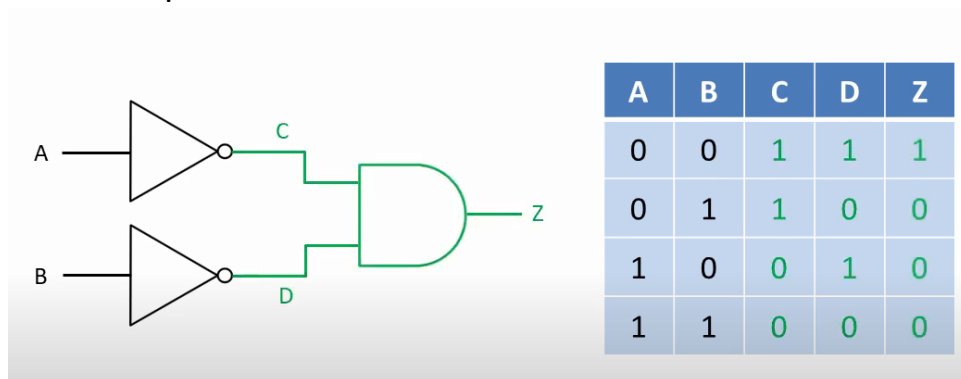


Figure 5 NAND-NAND and NOR-NOR representation of basic logic gate

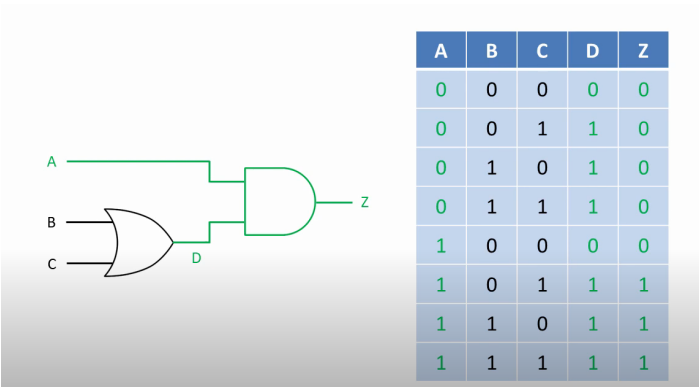
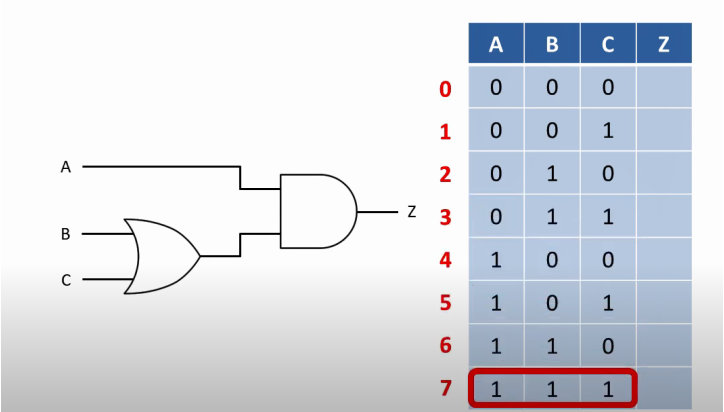
Circuit 1 example



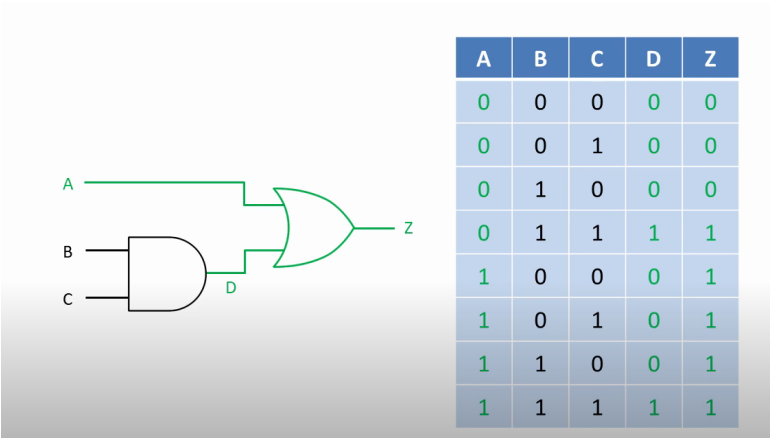
Circuit 2 example



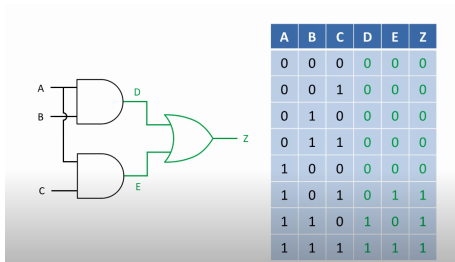
Circuit 3 example



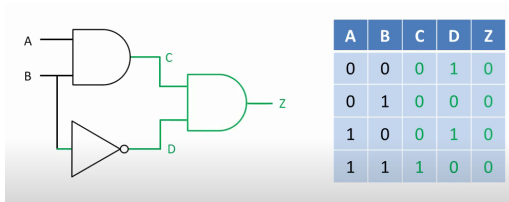
Circuit 4



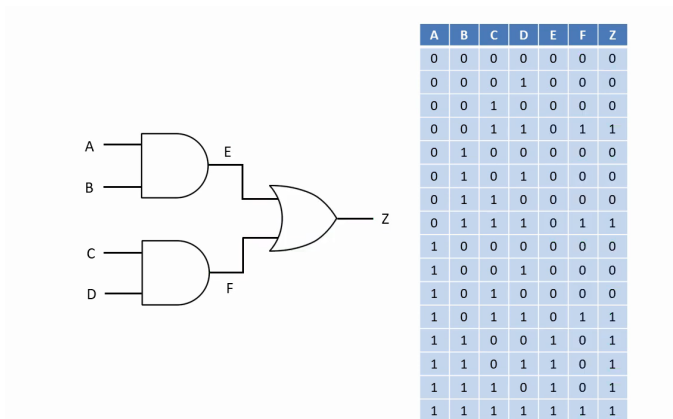
Circuit 5



Circuit 6

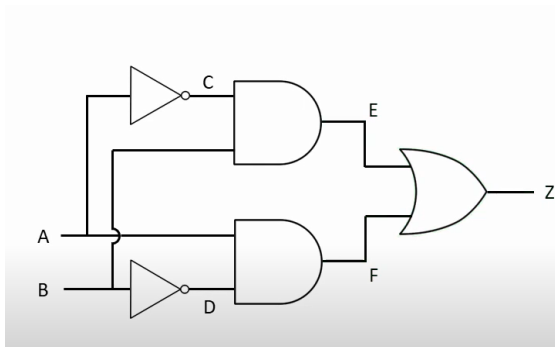


Circuit 7

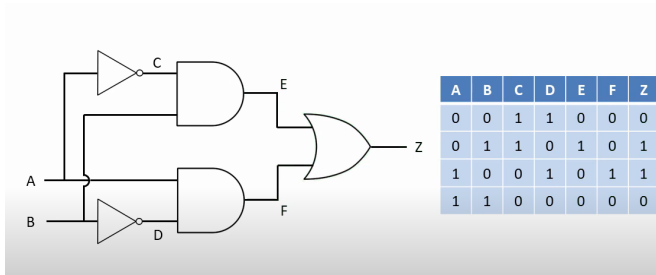


Circuit 8

Give the truth table of following expressions



Solution



Boolean Algebra:

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. **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 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.

Rules and Law of Boolean algebra:

i. Commutative law

Commutative law states that the inter-changing of the order of operands in a Boolean equation does not change its result.

- a. Using OR operator $\rightarrow A + B = B + A$
- b. Using AND operator $\rightarrow A * B = B * A$

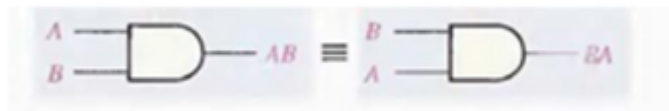


Figure 8 Commutative law in AND Gate

ii. Associative Law

a. Associate Law of Addition:

Associative law of addition states that OR more than two variables i.e. mathematical addition operation performed on variables will return the same value irrespective of the grouping of variables in an equation. It involves in swapping of variables in groups. The Associative law using OR operator can be written as

$$A+(B+C) = (A+B)+C$$

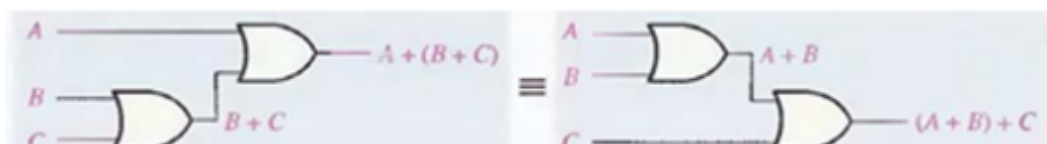


Figure 9 Application of Associative law of addition

b. Associate Law of Multiplication

Associative law of multiplication states that AND more than two variables i.e. mathematical multiplication operation performed on variables will return the same value irrespective of the grouping of variables in an equation. The Associative law using AND operator can be written as

$$A * (B * C) = (A * B) * C$$

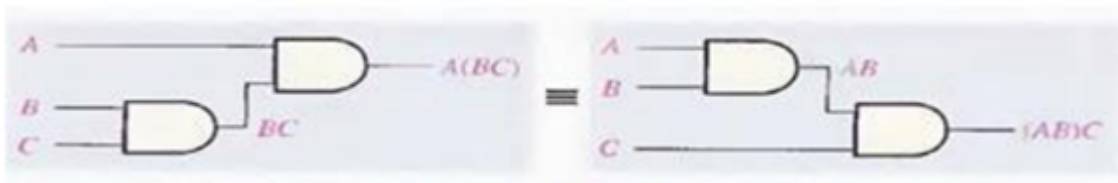


Figure 10 Application of Associative law of Multiplication

iii. **Distributive law**

This is the most used and most important law in Boolean algebra, which involves in 2 operators: AND, OR. The multiplication of two variables and adding the result with a variable will result in same value as multiplication of addition of the variable with individual variables. Distributive law can be written as

$$A + BC = (A + B)(A + C)$$

This is called OR distributes over AND.

The addition of two variables and multiplying the result with a variable will result in same value as addition of multiplication of the variable with individual variables. Distributive law can be written as

$$A (B+C) = (A B) + (A C)$$

This is called AND distributes over OR.

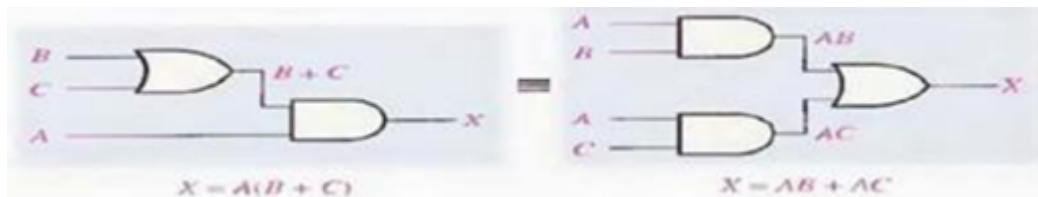


Figure 11 Application of Distributive law of Multiplication over addition and vice-versa

FINDING EXPRESSION FOR GIVEN LOGIC DIAGRAM

LOGIC DIAGRAM

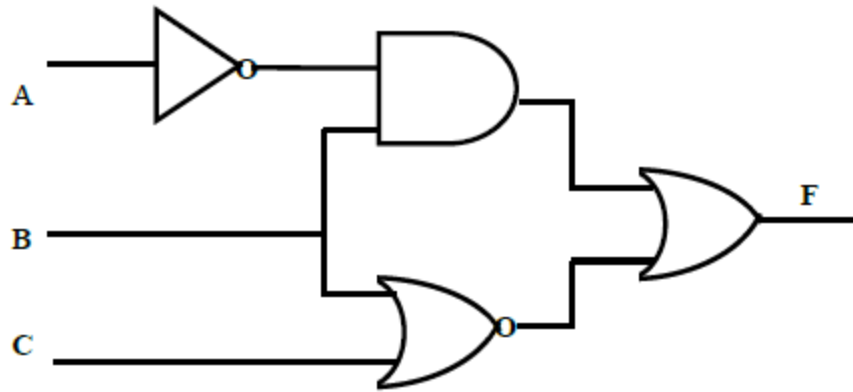


Figure 2: Logic Diagram

PROCEDURE

1. Set the power supply to 5V. With the help of a multimeter check the voltage at the output knobs of the power supply.
2. Connect wires, long enough to reach the bread board, with the two knobs of the power supply. Again using millimeter, check the voltage at the non-connected end of the wires.
3. Insert ICs on the bread board and make their supply and ground connections.
4. As given in the logic diagram, make connections using wires and gates in the ICs.
5. Apply different combinations at the three inputs and observe the output

OBSERVATION

A	B	C	EXPECTED OUTPUT	OBSERVED OUTPUT
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Lab Tasks

Task 1:

Design 3 input xor and xnor gate and write its truth table and draw the circuit diagram on the software.

Lab Task#2:

Implement the following logic circuit on logic trainer, and write Boolean Expression

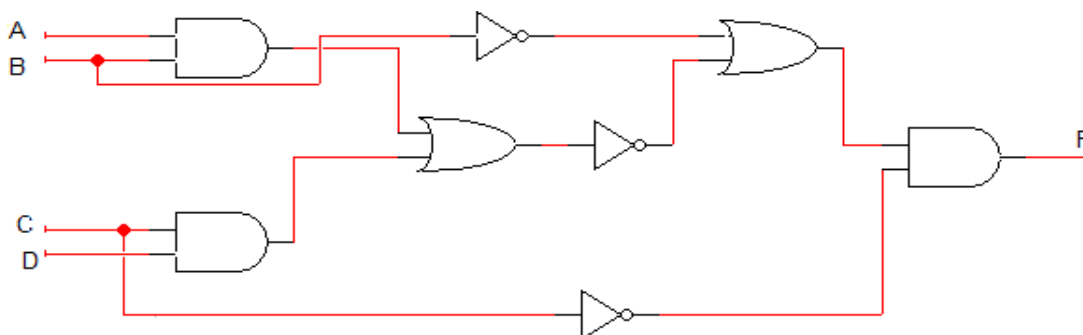
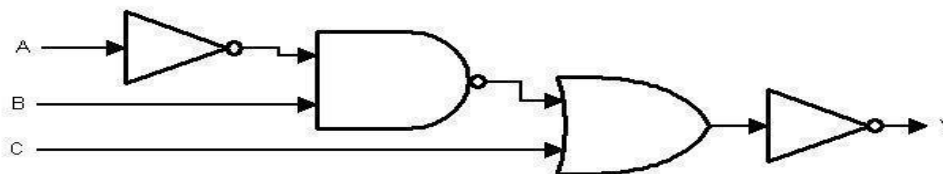


Figure 9: Combinational Circuit

Lab Task#3

Write the Boolean expression for each of the logic circuits in Figure 2.2. Also implement the given circuits on breadboard and draw Truth tables:



Combinational Circuit

Lab Task#4

Draw a circuit diagram corresponding to the following Boolean expression and implement it on breadboard:

1. $(A + B)(B + C)$
2. $(AB + C)D$
3. $((A + B'C)(A + BC))'$
4. $A'BC + AB'C + ABC' + (ABC)'$
5. $(A' + BC)'$

**Lab Task#5**

Transform the given diagram figure 9 of logic circuit to new logic diagram using NAND /NOR gates only, in the space given below. Show complete working. Implement the transformed logic circuit on logic trainer.



Task# 6

Implement associative , cumulative and distributive properties, draw its truth table.