**DESIGN AND IMPLEMENTATION OF A 4:1 MULTIPLEXER**

**1. Objective**

The objective of this lab is to design, simulate, and analyze the working of a **4:1 Multiplexer (MUX)** using logic gates in **Proteus**. Students will understand the fundamental concepts of multiplexers, their applications, and their implementation in digital circuits.

**2. Introduction to Multiplexers**

**2.1 What is a Multiplexer?**

A **Multiplexer (MUX)** is a combinational logic circuit that selects one input from multiple data inputs and forwards it to a single output based on the values of control (selection) inputs. It acts as a data selector, allowing multiple signals to share a single transmission line, reducing hardware complexity.

**2.2 Working Principle**

A multiplexer uses a set of **select lines** to choose one of the many input signals and pass it to the output. The number of select lines determines how many input signals the multiplexer can handle. For an **N-to-1 MUX**, the number of select lines required is **log₂(N)**.

**2.3 Applications of Multiplexers**

* Data Routing in Communication Systems
* Arithmetic and Logic Unit (ALU) Operations
* Memory and Register Access in Computers
* Digital Signal Processing (DSP)

**2.4 Multiplexer Overview**

A **4:1 MUX** has:

* **Four data inputs:** I1, I2, I3, I4
* **Two select lines:** S1, S0
* **One output:** Y

The **truth table** and Boolean expression determine which input is passed to the output based on the select lines.

**3. Truth Table for 4:1 MUX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S1** | **S0** | **Selected Input** | **Output (Y)** |
| 0 | 0 | I1 | I1 |
| 0 | 1 | I2 | I2 |
| 1 | 0 | I3 | I3 |
| 1 | 1 | I4 | I4 |

**3.1 Boolean Expression**

The output (Y) can be represented using the Boolean equation:

Y = (I1 \* S1' \* S0') + (I2 \* S1' \* S0) + (I3 \* S1 \* S0') + (I4 \* S1 \* S0)

Where **'** represents **NOT (inversion)** operation.

**4. Circuit Design of 4:1 Multiplexer in Proteus**

**4.1 Components Required**

* AND Gate (4)
* OR Gate (1)
* NOT Gate (2)
* Logic States (6) (I1, I2, I3, I4, S1, S0)
* LED (1) (Output Indicator)
* Ground

**4.2 Steps to Design in Proteus**

**Step 1: Open Proteus Software**

* Launch Proteus and create a **New Schematic Project**.

**Step 2: Add Components**

* Open the **Pick Devices** window and search for:
  + **AND Gate (2-input)** → Place **four** AND gates.
  + **OR Gate (4-input)** → Place **one** OR gate.
  + **NOT Gate** → Place **two** NOT gates.
  + **Logic States** → Place **six** for I1, I2, I3, I4, S1, and S0.
  + **LED** → Place **one** LED to observe the output.

**Step 3: Wiring the Circuit**

* Connect **S1 and S0** to the inputs of **NOT gates**.
* Use NOT gate outputs to generate **S1' and S0'**.
* Connect AND gate inputs according to the Boolean expression:
  + First AND Gate: Inputs = **I1, S1', S0'**
  + Second AND Gate: Inputs = **I2, S1', S0**
  + Third AND Gate: Inputs = **I3, S1, S0'**
  + Fourth AND Gate: Inputs = **I4, S1, S0**
* Connect the **outputs of all AND gates** to the **inputs of the OR gate**.
* Connect the **OR gate output** to the **LED**.

**Step 4: Simulating the Circuit**

* Run the simulation and toggle **I1, I2, I3, I4, S1, and S0** to observe the behavior.
* Verify the output with the **truth table**.

**5. Observations & Results**

* The output LED lights up based on the selected input.
* The circuit follows the expected **truth table behavior**.
* The design successfully implements a **4:1 Multiplexer using logic gates**.

**6. Conclusion**

In this lab, we designed and simulated a **4:1 multiplexer** using **Proteus**. We explored its **working principle, Boolean equation, circuit design, and applications**. The experiment demonstrated how select lines control the **data selection process**, proving the **importance of MUX in digital systems**.

**7. LAB TASKS**

1. Design a 2:1 MUX using Proteus

2. Design a 8:1 MUX using Proteus