Experiment 5
Multiplexers Design and Implementation

Introduction:

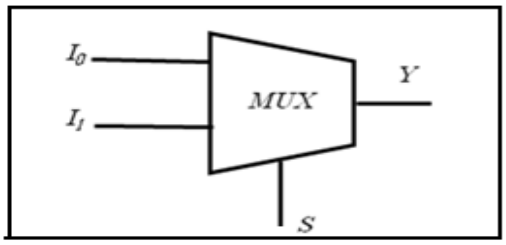
A multiplexer (or data selector) is a device that is capable of taking two or more data lines and converting them into a single data line for transmission to another point.
A multiplexer (MUX) performs the function of selecting the input on any one of 'n' input lines and feeding this input to one output line. Multiplexers are used as one method of reducing the number of integrated circuit packages required by a particular circuit design. This in turn reduces the cost of the system.

Objectives:

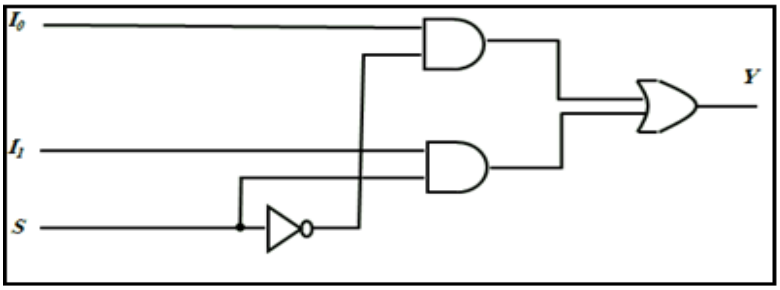
- Design, build, and test Multiplexers.
- Demonstrate the operations and applications of Multiplexers.
- Implement logic functions using Multiplexers.
- Use Tri-State Buffers to implement a Multiplexer.

Multiplexer

- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output lines.
- The selection of particular input line is controlled by a set of selection lines.
- Normally, there are $(2)^n$ input line and n selection lines whose combinations determine which input is selected.
- A 2-to-1 line multiplexer connects one of two 1-bit sources to a common destination as shown in figure below.



(a) Block Diagram



(b) Logic Diagram

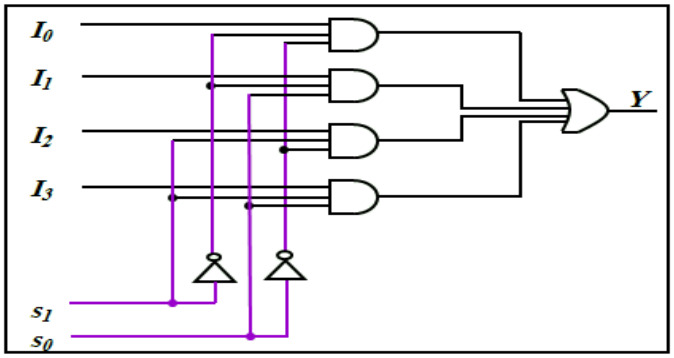
Figure(1)

Truth Table of 2-1 mux:

*i*nput			Output
S	I0	I1	
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

- The circuit has two data input lines, and one selection line S.
- When S=0, the upper AND gate is enabled and the I0 has a path to the output
- When S=1, the lower AND gate is enabled and I1 has path to the output.
- The multiplexer acts like an electric switch that selects one of two sources.

A *4-to-1* line multiplexer is shown below



Figure(2)

- Each of the four inputs, I_0 through I_3, is applied to one input of an AND gate.
- Selection lines S_1 and S_0 are decoded to select a particular AND gate
- The output of AND gates are applied to a single OR gate that provides the 1-line output.

Function implementation using multiplexer:

There are three main methods for implementing a circuit using multiplexers. These are:

1. Algebraic Method of Multiplexer Implementation :

This is an approach where you can transform one boolean expression into a form so that a multiplexer can be implemented, This can be achieved by applying Boolean Theorems. Before attempting the design of a multiplexer using the algebraic method, the function to be considered should be minimized, Minimizing the terms and expressions can be important because this allows designers to use the least amount of components and use the most efficient type of multiplexer.

Example:
Consider the function:

$$f(A,B,C) = \overline{A}B + BC + \overline{A}C$$

Expanding to standard sum of products form:

$$f(A,B,C) = \bar{A}B(C + \bar{C}) + BC(A + \bar{A}) + \bar{A}C(B + \bar{B})$$

$$= \bar{A}BC + \bar{A}B\bar{C} + ABC + \cancel{A\bar{B}C} + \cancel{A\bar{B}C} + \bar{A}\bar{B}C$$

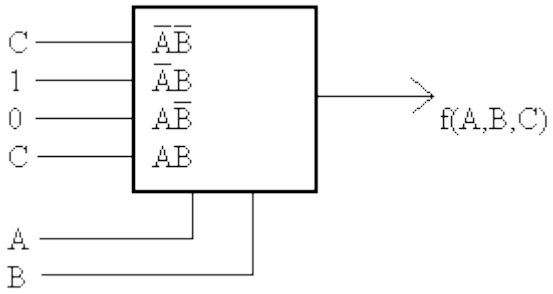
$$= \bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC$$

$$= \bar{A}\bar{B}C + \bar{A}B(C + \bar{C}) + ABC$$

$$= \bar{A}\bar{B}(C) + \bar{A}B(1) + A\bar{B}(0) + AB(C)$$

The resulting multiplexer implementation is:

Figure(3)



2. Karnaugh map method

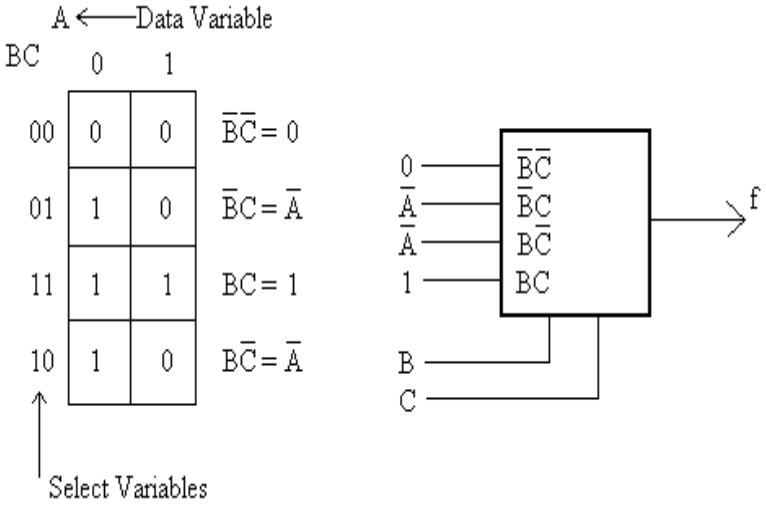
It can be seen that applying Boolean algebra can be awkward in order to implement multiplexers. This is because it takes a lot of practice and can be very difficult to determine the set of laws and propositions to use. Karnaugh maps provide a simple and straight-forward method of implementing multiplexers. With the Karnaugh map Boolean expressions having up to four and even six variables can be implemented.

Example:

Consider the function:

$$f(A,B,C) = \bar{A}B + BC + \bar{A}C$$

In this example we could have picked any variable to be the data variable and the other two as select variables. Suppose we take A as the data variable. The corresponding Karnaugh map is then:

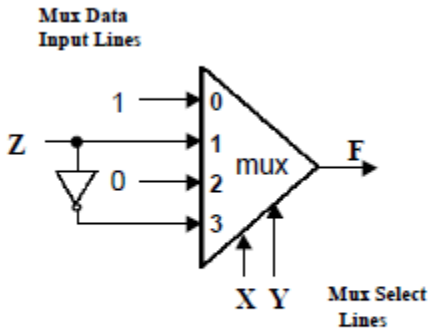


3-Truth table Method of Multiplexer Implementation.

Example:

Implement the function F(X,Y,Z) = $\sum$(0,1,3,6) using a single 4-to-1mux and an inverter.
– We choose the two most significant inputs X, Y as mux select lines.
– Construct truth table:

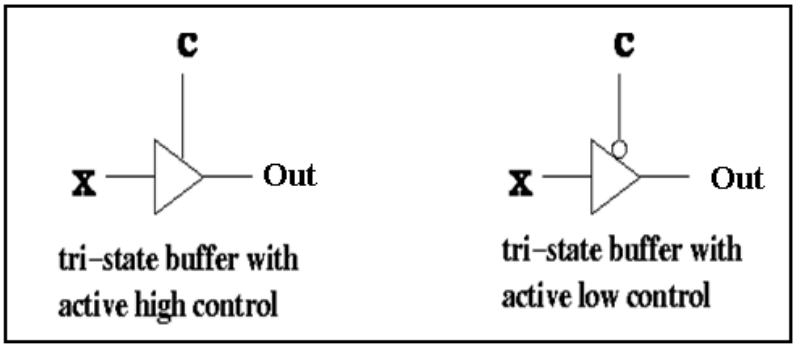
Select Lines Value i	Select Lines X	Y	Z	F	Mux Input i
0	0	0	0	1	1
	0	0	1	1	
1	0	1	0	0	Z
	0	1	1	1	
2	1	0	0	0	0
	1	0	1	0	
3	1	1	0	1	Z'
	1	1	1	0	



• We Determine multiplexer input line i values by comparing the remaining input variable Z and the function F for the corresponding selection lines value i:
– when XY=00 the function F=1 (for both Z=0, Z=1) thus mux input0 = 1
– when XY=01 the function F=Z thus mux input1 = Z
– when XY=10 the function F=0 (for both Z=0, Z=1) thus mux input2 = 0
– when XY=11 the function F=Z' thus mux input3 = Z'

Tri-state buffer

A tri-state buffer is a digital device that is capable of three different outputs, high, low and disconnected (high impedenace).
Here's two diagrams of the tri-state buffer.



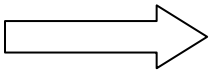
Figure(4)

A tri-state buffer has two inputs: a data input **x** and a control input **c**. The control input acts like a valve. When the control input is active, the output is the input. That is, it behaves just like a normal buffer.

When the control input is not active no electrical current flows through,so the tri state buffer is in **high impedance state (Z)** ,Thus, even if **x** is 0 or 1, that value does not flow through.

Here's a truth table describing the behavior of a active-high tri-state buffer.

c	x	Out
0	0	Z
0	1	Z
1	0	0
1	1	1



c	Out
0	Z
1	x

Active-low tri-state buffers
Some tri-state buffers are active low. In an active-low tri-state buffer, **c = 0** turns open the valve, while **c = 1** turns it off.

Here's the condensed truth table for an active-low tri-state buffer.

c	Out
0	x
1	Z

Procedure:

I. Use the following pin assignment for inputs and outputs when needed:

- In1: SW [1] (Pin_AB26)
- In2: SW [2] (Pin_AB25)
- In3: SW [3] (Pin_AC27)
- In4: SW [4] (Pin_AC26)
- In5: SW [5] (Pin_AC24)
- In6: SW [6] (Pin_AC23)
- In7: SW [7] (Pin_AD25)
- In8: SW [8] (Pin_AD24)
- In9: SW [9] (Pin_AE27)

- Out0: LEDR [0] (Pin_AJ6)
- Out1: LEDR [1] (Pin_AK5)

Part 1: Basics of Multiplexer

1. Implement the logic diagram for 8-1 MUX on Quartus II (Verilog).

Part 2: Functions Implementation Using Multiplexer

1. Implement the logic function F (A, B, C) = Σ (1, 2, 4, 5) using (IC 74153) Multiplexer on Quartus II (schematic).

2. For circuit shown in Figure below, what logic function does it produce?

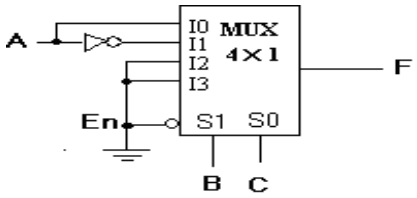


Figure 1

Part 3: Multiplexer Expansion:

- Implement an 8-to-1 MUX using only one 74153 IC and a single inverter and 2-input OR gate on Quartus II (schematic).

Part4:Tri-State Buffer:

- Implement 2-to-1 Multiplexer; see Figure 2, using only two tri-state buffers (others →maxplus2 →Tribuf) and a single inverter on Quartus II (schematic).
- Note: the tri state buffer above is active low control line.

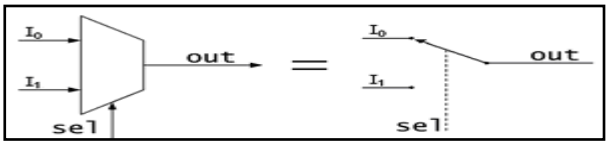


Figure 2