EE2174 Lab 8

Multiplexers and Decoders

Introduction

In this lab, you will further control DE10-Lite pheripherials by incorporating multiplexers and decoders. You have previously seen a decoder implemented in the numeric display driver "**hex7seg**". In this lab, you will create a new decoder to display alphabet letters. Then, multiplexers will be implemented to display a full word on the seven segment displays.

Multiplexer Background

Multiplexers are a logic device which acts as a selector to implement logic functions. The device has a set of selector bits, a set of input bits, and one output. The naming convention used for multiplexers is the number of inputs to the number of outputs. To manually design a multiplexer, determine all inputs to the device. Then, map selector values to the inputs to correspond which input becomes the output. Throughout the lab, you will be led through this process to develop a large 6 to 1 multiplexer.

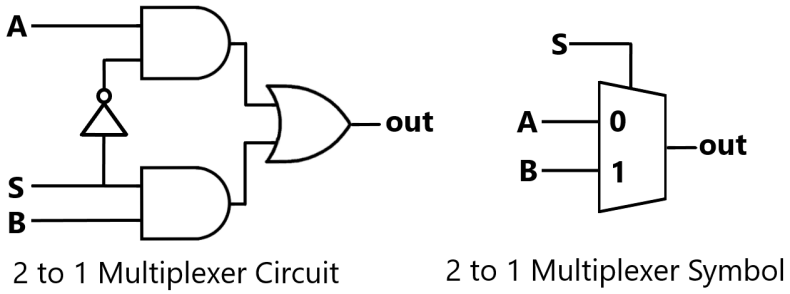
1 Building the Multiplexer

1.1 2 to 1 Multiplexer

In this part of the lab, you will create a 2 to 1 multiplexer to use as a base point for implementing larger order multiplexers. Create a new project and use the code provided in Listing 1 to create a 2 to 1 multiplexer.

```
11  // 2 to 1 Multiplexer
12  module MUX2to1(
13  input A, B, S,
14  output out);
15
16      // Outputs A or B based on the
17      // value of the select bit (S).
18      assign out = (~S&A)|(S&B);
19
20  endmodule
```

Listing 1: 2 to 1 Multiplexer



2 to 1 Multiplexer Circuit 2 to 1 Multiplexer Symbol

Figure 1: Circuit and symbol representation of a 2 to 1 multiplexer

Simulate the 2 to 1 multiplexer module using a functional simulation.

Signoff

Show your TA your functional simulation. **Save a screenshot of the simulation's output and submit it with this lab.**

1.2 3-bit Wide 2 to 1 Multiplexer

In this part of the lab, you will extend the bit width of your 1-bit 2 to 1 multiplexer to 3 bits. This will be used later in the lab to make a 6 to 1 multiplexer.

Create a new Verilog HDL file named "**MUX3bit2to1**". The behavior of this 2 to 1 multiplexer will be similar to the 1-bit wide 2 to 1 multiplexer built in Part 1.1, however the bit width of the input bits "**A**" and "**B**" and the output bits "**out**" will be 3.

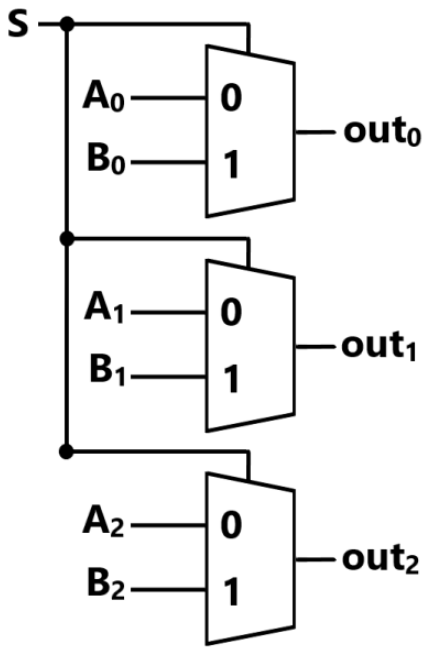


Figure 2: 3-bit wide 2 to 1 multiplexer implemented using three 1-bit wide 2 to 1 multiplexers.

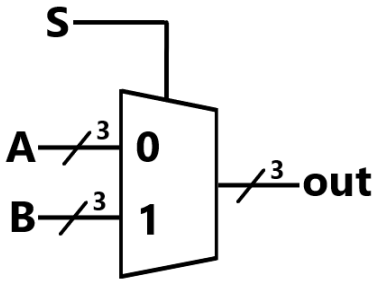


Figure 3: High-level 3-bit wide 2 to 1 multiplexer symbol.

Complete the module based on Figure 2, and test its functionality using a functional simulation. To test the module, set A equal to a random 3-bit value and B equal to the inverse of A (meaning all the bit values are flipped). For both values of S, ensure the multiplexer outputs A or B correspondingly.

Signoff

Show your TA your functional simulation. **Save a screenshot of the simulation's output and submit it with this lab.**

2 HELLO Display Decoder

In previous labs, the seven segment displays were used to display numbers. Now, a display decoder will be made to allow you to display letters on the displays. In this part of the lab, you will output individual letters “H, E, L and O” on a single display based on switch input.

Start by creating a hello decoder module. Use the code in Listing 2 to get started; utilize your prelab to assist in developing the binary value needed to display each alphabet character. Choose an appropriate default state such that the display is off for all values not explicitly defined in the case statement, and map each display to the input described in Table 1.

```
11  // Hello decoder
12  module HelloDecoder(
13  input [2:0] num,                    // Three bit input that can take a value between 0 and 7
14  output reg [6:0] display);          // Output intended to go to 7-segment display i.e. HEX0
15
16      always@(num)
17      begin
18          case(num)
19              // Complete this section based on hex7seg decoder from Lab 4
20                                                  // Display H
21                                                  // Display E
22                                                  // Display L
23                                                  // Display O
24              default: display = 7'b_______;      // Choose an approprate default case value
25          endcase
26      end
27
28  endmodule
```

Listing 2: Hello decoder starting code

Table 1: Select input to character mappings

SW	Character
000	H
001	E
010	L
011	O
100	
101	
110	
111	

Now, create a main module to control which character is output to a display. Use the first three switches as input to the decoder and HEX0 as the output.

Signoff

Demonstrate to your TA the physical implementation of the program. Cycle through the switch inputs to show the four possible letter outputs and verify additional inputs result in an off display.

3 HELLO Display

In the world HELLO, there are five letters. There are six seven segment displays on the DE10-Lite, so a blank character will be used to fill the sixth position. In this part of the lab, you will control all six seven segment displays and output the word HELLO. To control which character displays on each segment, you will control each segment with a 6 to 1 multiplexer.

3.1 6 to 1 Multiplexer

A 6 to 1 multiplexer will allow the display to select between the five letters in HELLO and a blank character. Controlling each seven segment display with its own multiplexer allows you to control each segment individually by inputting different select values to each segment's multiplexer.

You will create a 6 to 1 multiplexer by instantiating multiple 3-bit wide 2 to 1 multiplexers. Start with the code shown in Lising 3 to get started. Complete the 6 to 1 multiplexer using Figure 4 as a guide.

```
11  // 3 bit wide 6 to 1 Multiplexer created
12  // using 3 bit wide 2 to 1 multiplexers
13  module MUX6to1(
14  input [2:0] S,                // 3-bit select bit
15  input [2:0] A, B, C, D, E, F, // 3-bit input values
16  output [2:0] out);            // 3-bit output value
17
18      // 3-bit temporary variables
19      wire [2:0] tempA, tempB, tempC, tempD;
20
21      // This is one of the five 3-bit wide 2 to 1 multiplexers needed
22      MUX3bit2to1 MUX0(.A(A), .B(B), .S(S[0]), .out(tempA));
23
24      // Complete the rest of the module
25
26  endmodule
```

Listing 3: Start of 6 to 1 multiplexer

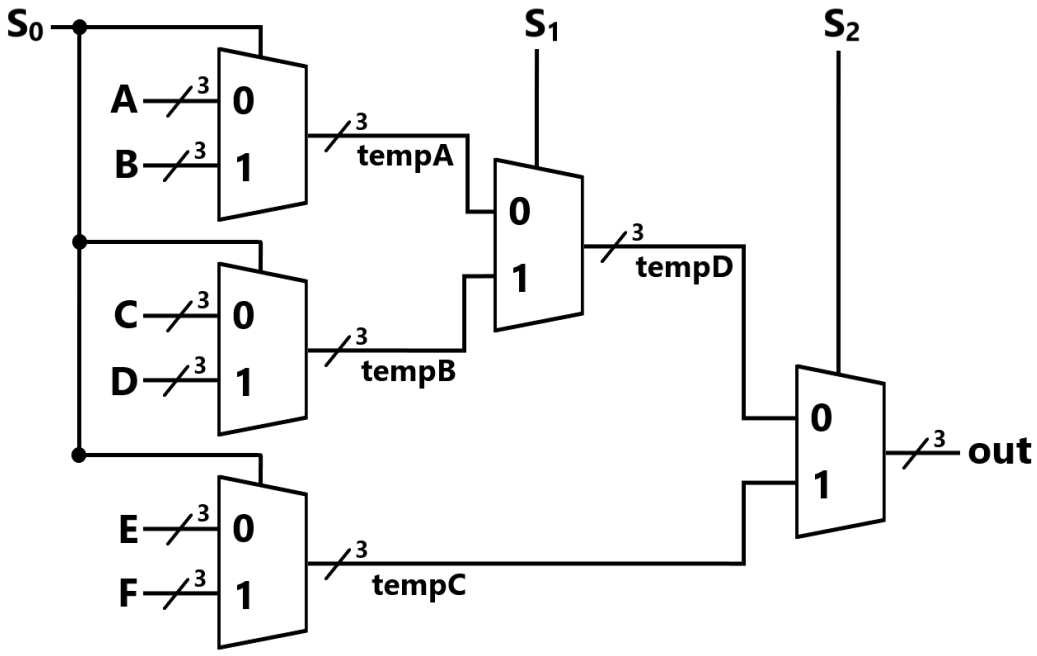


Figure 4: 3-bit wide 6 to 1 multiplexer created from 3-bit wide 2 to 1 multiplexers.

3.2 Main Display Control

You will now create a main module that will control all six seven segment displays. Each seven segment display will be controlled using a 3-bit wide 6 to 1 multiplexer. The same 3-bit select bits will be used to control each multiplexer, but the values for each multiplexer to choose from will be offset by one for each instantiation made. This technique means that each display will output a unique character for any given select input.

Use Listing 4 to get started on this main module. Use Table 2 to assist in offsetting the inputs to each multiplexer instance.

To accomplish efficiency in instantiating inputs, parameters have been used for each of the inputs. A parameter is a localized constant, similar to final variables in Java or #define constants in C.

```
11  // Main Display Control Module
12  module Lab8Part3(
13  input [2:0] SW,
14  output [6:0] HEX0,HEX1,HEX2,
15  output [6:0] HEX3,HEX4,HEX5);
16
17        parameter H = 3'b000;
18        parameter E = 3'b001;
19        parameter L = 3'b010;
20        parameter O = 3'b011;
21        parameter B = 3'b100;
22
23        wire [2:0] out1, out2, out3, out4, out5, out6;
24        wire [2:0] sel;
25
26        // Set select bits equal to the first three switches
27        assign sel = SW;
28
29        // First character pattern:      HELLOB
30        MUX6to1 case1(.S(sel), .A(H), .B(E), .C(L), .D(L), .E(O), .F(B), .out(out1));
31        HelloDecoder display1(.num(out1), .display(HEX5));
32
33        // Complete the other 5 display control character patterns
34
35  endmodule
```

Listing 4: Starting basis for main display control module. The “B” character refers to the blank character state on a display.

Table 2: Pattern of the display based on control switch input. The "_" character refers to a blank character on the display.

SW[2]SW[1]SW[0]	Character Pattern
000	H E L L O _
001	E L L O _ H
010	L L O _ H E
011	L O _ H E L
100	O _ H E L L
101	_ H E L L O

Set this module as the "Top-Level Entity" and compile. Program the DE10-Lite and ensure the display reads "**HELLO_**" with no switch input. The "_" character refers to a blank display. Cycle through the switch inputs to test the other five character patterns. Incrementing through the switch inputs in order should result in the word "HELLO" scrolling along the seven segment displays.

Signoff

There are two signoffs: For the first signoff, demonstrate to your TA the general output behavior of all 6 displays by displaying "**HELLO_**". For the second signoff, show the positional switch control functionality to your TA. These sign offs can be completed simultaneously, or as individual tasks.

4 Scrolling HELLO

In this part of the lab, a counter will be used to increment the 3-bit select input. First, the counter will be incremented positively to rotate the word through the display in one direction. Then, switch input will be added to change the direction of the scroll.

4.1 Clock Incorporation

Start by making a new module based on the main display control created in Part 3. Relying on past work, create a counter that will increment the 3-bit select value at a rate of 1 Hz.

A 3-bit variable can represent eight possible values (0-7). Given we only have six display cases to select from, you will need to add behavior to the counter module to force the select variable to roll-over after it reaches a maximum value of 5. Use your code draft from the prelab to help develop this algorithm.

Signoff

Demonstrate to your TA that the word “HELLO” scrolls on the DE10-Lite.

4.2 Direction Control

The last component to implement is to control the direction of the scrolling. Use switch zero as the direction control. Determine the best way to implement this feature.

One way is to choose to decrement the select counter if the opposite direction is desired. Note that special care needs to be taken when decrementing the select bit given the value must stay between 0 and 5; after the smallest value of 0 is reached, the counter must roll-over to the maximum value of 5.

Signoff

Demonstrate to your TA that the word “HELLO” scrolls on the DE10-Lite and the direction of the scroll can be flipped by toggling switch 0.

5 Submission

List of screenshots that must be submitted with the submission sheet in one combined pdf, and list of code files that must be submitted. All code required to run the project should be submitted.

Single PDF

- Submission Sheet
- Part 1.1 Functional Simulation Screenshot (One bit wide 2 to 1 Mux)
- Part 1.2 Functional Simulation Screenshot (Three bit wide 2 to 1 Mux)

Code

- MUX2to1.v (One bit wide 2 to 1 Mux)
- MUX3bit2to1.v (Three bit wide 2 to 1 Mux)
- HelloDecoder.v (Display Decoder)
- Lab8Part2.v (Single Display Main Module)
- MUX6to1.v (6 to 1 Mux)
- Lab8Part3.v (Main Display Control)
- Lab8Part4.v (Scrolling HELLO with direction)