Laboratory Exercise 1

Switches, Lights, and Multiplexers

The purpose of this exercise is to learn how to connect simple input and output devices to an FPGA chip and implement a circuit that uses these devices. As input devices we will use the SW_{9-0} switches on a DE1-SoC board, and we will use light emitting diodes (LEDs) and 7-segment displays as output devices.

Preparation

In advance of the lab exercise you are required to write the Verilog code for Parts I to IV. Your lab teaching assistant (TA) will mark your preparation for Parts III and IV. We are not using lab books for this course, so you can simply open your Verilog code in an editor of your choice to show to your TA.

In-lab Work

You are required to implement and test Parts I to V of the exercise. You will only need to demonstrate Parts III, IV, and V to your teaching assistant. Part VI of the exercise is optional; do this part if you want to.

Part I

The DE1-SoC board provides ten slide switches, called SW_{9-0}, that can be used as inputs to a circuit, and ten red lights, called $LEDR_{9-0}$, that can be used to display output values. Figure 1 provides a simple Verilog module that uses these switches and shows their states on the LEDs. Since there are ten switches and lights it is convenient to represent them as vectors in the Verilog code, as shown. We have used a single assignment statement for all ten *LEDR* outputs, which is equivalent to the individual assignments

```
assign LEDR[9] = SW[9];
assign LEDR[8] = SW[8];
...
assign LEDR[0] = SW[0];
```

The DE1-SoC board has hardwired connections between its FPGA chip and the switches and lights. To use SW_{9-0} and $LEDR_{9-0}$ it is necessary to include in your Quartus project the correct pin assignments, which are given in the *DE1-SoC User Manual*. For example, the manual specifies that on the DE1-SoC board SW_0 is connected to the FPGA pin *AB12* and $LEDR_0$ is connected to pin *V16*. A good way to make the required pin assignments is to import into the Quartus software the file called *DE1-SoC.qsf*. This file can be downloaded from the `Teaching and Projects Boards` section of the website FPGAcademy.org.

```
// Simple module that connects the SW switches to the LEDR lights
module part1 (SW, LEDR);
    input  [9:0] SW;     // toggle switches
    output [9:0] LEDR;   // LEDs

    assign LEDR = SW;
endmodule
```

Figure 1: Verilog code that uses the DE1-SoC board switches and lights.

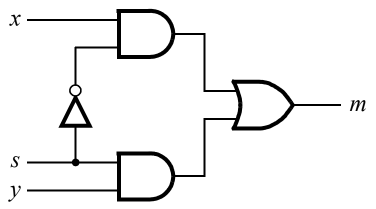
The pin assignments in the *.qsf* file are useful only if the pin names given in the file are exactly the same as the port names used in your Verilog module. The assignments file uses the names *SW*[0] . . . *SW*[9] and *LEDR*[0] . . . *LEDR*[9] for the switches and lights, which is the reason that we used these names in Figure 1.

Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the DE1-SoC board.

1. Create a new Quartus project for your circuit. For the DE1-SoC board, select Cyclone V 5CSEMA5F31C6 as the target chip.
2. Type the Verilog code from Figure 1 into a file, and include it in your project.
3. Include in your project the required pin assignments, as discussed above. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the switches and observing the LEDs.

Part II

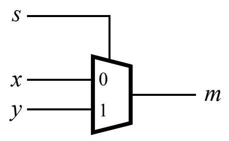
Figure 2a shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input s. If $s = 0$ the multiplexer's output m is equal to the input x, and if $s = 1$ the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows its circuit symbol.



a) Circuit

s	m
0	x
1	y

b) Truth table



c) Symbol

Figure 2: A 2-to-1 multiplexer.

The multiplexer can be described by the following Verilog statement:

```
assign m = (~s & x) | (s & y);
```

You are to write a Verilog module that includes four assignment statements like the one shown above to describe the circuit given in Figure 3a. This circuit has two four-bit inputs, X and Y, and produces the four-bit output M. If $s = 0$ then $M = X$, while if $s = 1$ then $M = Y$. We refer to this circuit as a four-bit wide 2-to-1 multiplexer. It has the circuit symbol shown in Figure 3b, in which X, Y, and M are depicted as four-bit wires.

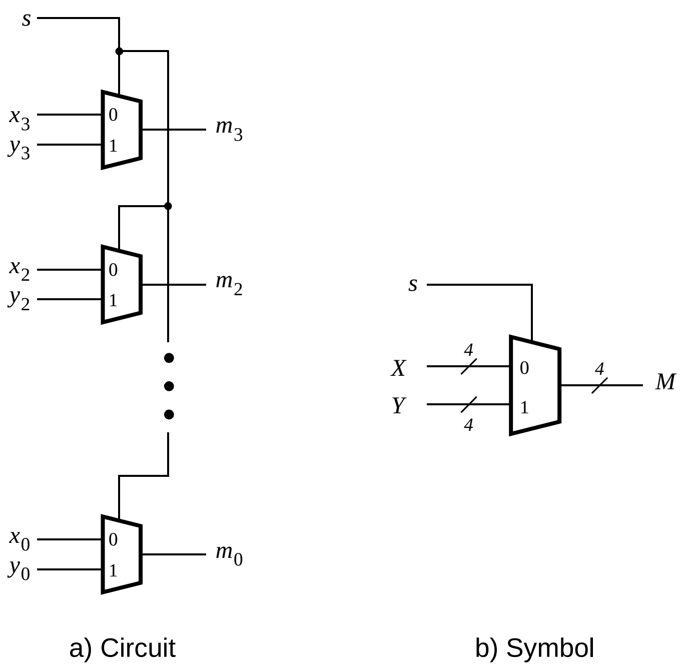


Figure 3: A four-bit wide 2-to-1 multiplexer.

Perform the steps below.

1. Create a new Quartus project for your circuit.
2. Include your Verilog file for the four-bit wide 2-to-1 multiplexer in your project. Use switch *SW*$_9$ on the DE1-SoC board as the s input, switches *SW*$_{3-0}$ as the X input and *SW*$_{7-4}$ as the Y input. Display the value of s on *LEDR*$_9$, connect the output M to *LEDR*$_{3-0}$, and connect the unused LEDR lights to the constant value 0.

   Your Verilog code will be more readable if you use assignment statements of the form

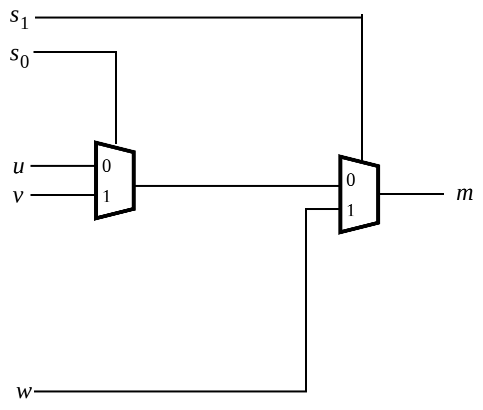
   ```
   ...
   assign M[0] = (~s & X[0]) | (s & Y[0]);
   assign M[1] = (~s & X[1]) | (s & Y[1]);
   ...
   ```

   and then use separate assignment statements to associate *s*, *X*, and *Y* to *SW* switches, and to assign *s* and *M* to *LEDR* lights.
3. Include in your project the required pin assignments for the DE1-SoC board. As discussed in Part I, these assignments ensure that the input ports of your Verilog code will use the pins on the FPGA that are connected to the *SW* switches, and the output ports will use the FPGA pins connected to the *LEDR* lights.
4. Compile the project.
5. Download the compiled circuit into the FPGA chip. Test the functionality of the four-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.

Part III

In Figure 2 we showed a 2-to-1 multiplexer that selects between the two inputs x and y. For this part consider a circuit in which the output m has to be selected from three inputs u, v, and w. Part a of Figure 4 shows how we can build the required 3-to-1 multiplexer by using two 2-to-1 multiplexers. The circuit uses a 2-bit select input $s_1 s_0$ and implements the truth table shown in Figure 4b. A circuit symbol for this multiplexer is given in part c of the figure.

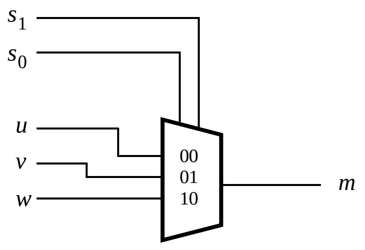
Recall from Figure 3 that a four-bit wide 2-to-1 multiplexer can be built by using four instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a two-bit wide 3-to-1 multiplexer. It contains two instances of the circuit in Figure 4a.



a) Circuit

s_1 s_0	m
0 0	u
0 1	v
1 0	w
1 1	w

b) Truth table



c) Symbol

Figure 4: A 3-to-1 multiplexer.

Perform the following steps to implement the two-bit wide 3-to-1 multiplexer.

1. Create a new Quartus project for your circuit.
2. Create a Verilog module for the two-bit wide 3-to-1 multiplexer. Connect its select inputs to switches SW_{9-8}, and use switches SW_{5-4}, SW_{3-2}, SW_{1-0} to provide the three 2-bit inputs U, V, W, respectively. Connect the output M to the red lights $LEDR_{1-0}$.

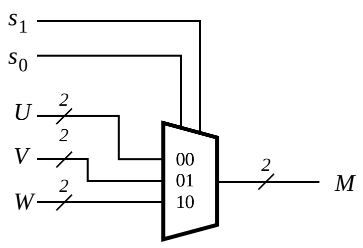


Figure 5: A two-bit wide 3-to-1 multiplexer.

3. Include in your project the required pin assignments for the DE1-SoC board. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the two-bit wide 3-to-1 multiplexer by toggling the switches and observing the LEDs. Ensure that each of the inputs U to W can be properly selected as the output M.

Part IV

Figure 6 depicts a *7-segment decoder* module that has the two-bit input c_1c_0. This decoder produces seven outputs that are used to display a character on a 7-segment display. Table 1 lists the characters that should be displayed for each valuation of c_1c_0. Three characters are included plus the 'blank' character, which is selected for code 11.

The seven segments in the display are identified by the indices 0 to 6 shown in the figure. Each segment is illuminated by driving it to the logic value 0. You are to write a Verilog module that implements a logic function for each of the seven segments. Use only simple Verilog **assign** statements in your code to specify each logic function using a Boolean expression.

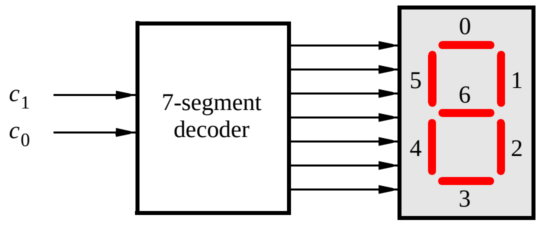


Figure 6: A 7-segment decoder.

c_1c_0	Character
00	d
01	E
10	1
11	

Table 1. Character codes.

Perform the following steps:

1. Write Verilog code for your 7-segment decoder. Connect the c_1c_0 inputs to switches SW_{1-0}, and connect the outputs of the decoder to the *HEX0* display on the DE1-SoC board. The segments in this display are called $HEX0_0$, $HEX0_1$, . . ., $HEX0_6$, corresponding to Figure 6.
2. A *top.v* file is provided as part of the *design files* for this exercise, for use with the *DESim* tool. Compiling and simulating your 7-segment decoder with *DESim* is a good way to see how the 7-segment displays will look when your circuit is implemented in a DE1-SoC board. To use the *top.v* file directly, declare your Verilog module as:

```
module part4 (SW, LEDR, HEX0);
    input [1:0] SW;             // toggle switches
    output [9:0] LEDR;          // red LEDs
    output [6:0] HEX0;          // 7-seg display
    ...
endmodule
```

3. After you have finished testing your Verilog code using simulation, create a Quartus project for your circuit. Make sure to include all required pin assignments in the project.
4. Compile your project and then download the resulting circuit into the FPGA chip. Test the functionality of the circuit by toggling the SW_{1-0} switches and observing the 7-segment display.

Part V

Consider the circuit shown in Figure 7. It uses a two-bit wide 3-to-1 multiplexer to enable the selection of three characters that are displayed on a 7-segment display. Using the 7-segment decoder from Part IV this circuit can display the characters d, E, 1, and ‘blank’. The character codes are set according to Table 1 by using the switches SW_{5-0}, and a specific character is selected for display by setting the switches SW_{9-8}.

An outline of the Verilog code that represents this circuit is provided in Figure 8. We have used the circuits from Parts III and IV as subcircuits in this code. You are to extend the code in Figure 8 so that it uses three 7-segment displays rather than just one. You will need to use three instances of each of the subcircuits. The purpose of your circuit is to display any word on the three displays that is composed of the characters in Table 1, and be able to rotate this word in a circular fashion across the displays when the switches SW_{9-8} are toggled. As an example, if the displayed word is dE1, then your circuit should produce the output patterns illustrated in Table 2.

Perform the following steps.

1. Write Verilog code for your circuit that can display three-letter words. Your code should have the same structure indicated in Figure 8, but supporting three displays *HEX2*, *HEX1*, and *HEX0*, rather than just *HEX0*. In your Verilog code connect the switches SW_{9-8} to the select inputs of each of the three instances of the two-bit wide 3-to-1 multiplexers that is needed for each display. Also, connect SW_{5-0} to the data inputs of these multiplexers in such a way as required to produce the patterns of characters shown in Table 2. Display the values of the *SW* switches on the red lights *LEDR*, in addition to displaying your multiplexer outputs on *HEX2*, *HEX1*, and *HEX0*.
2. A *top.v* file is provided as part of the *design files* for this exercise, for use with the *DESim* tool. To use the *top.v* file directly, declare your Verilog module as:

```
module part5 (SW, LEDR, HEX2, HEX1, HEX0);
    input [9:0] SW;
    output [9:0] LEDR;
    output [6:0] HEX2, HEX1, HEX0;
    ...
endmodule
```

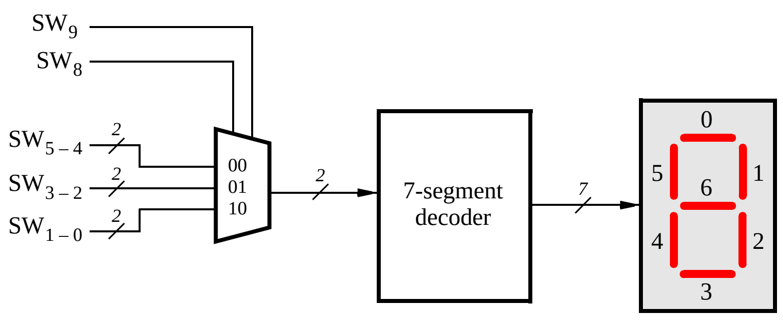


Figure 7: A circuit that can select and display one of three characters.

3. After completing simulations of your code, make a Quartus project for this part of the exercise.
4. Include the required pin assignments for the DE1-SoC board for all switches, LEDs, and 7-segment displays. Compile the project.
5. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper character codes on the switches SW_{5-0} and then toggling SW_{9-8} to observe the rotation of the characters.

```
module fig7 (SW, LEDR, HEX0);
    input [9:0] SW;          // toggle switches
    output [9:0] LEDR;       // red LEDs
    output [6:0] HEX0;       // 7-seg displays

    assign LEDR = SW;

    // declare any wires needed
    ... code not shown

    // instantiate module mux_2bit_3to1 (S, U, V, W, M);
    mux_2bit_3to1 M0 (...);

    // instantiate module char_7seg (C, Display);
    char_7seg H0 (...);
endmodule

// implements a 2-bit wide 3-to-1 multiplexer
module mux_2bit_3to1 (S, U, V, W, M);
    input  [1:0] S, U, V, W;
    output [1:0] M;
    ... code not shown

endmodule

// implements a 7-segment decoder for d, E, 1 and 'blank'
module char_7seg (C, Display);
    input  [1:0] C;              // input code
    output [6:0] Display;        // output 7-seg code
    ... code not shown

endmodule
```

Figure 8: Verilog code for the circuit in Figure 7.

SW_{9-8}	Characters		
00	d	E	1
01	E	1	d
10	1	d	E

Table 2. Rotating the word dE1 on three displays.

Part VI

Extend your design from Part V so that is uses all six 7-segment displays on the DE1-SoC board. Your circuit needs to display the word dE1 on the six displays and to be able to rotate this word from right-to-left as shown in Table 3. To do this, you will need to connect two-bit wide 6-to-1 multiplexers to each of six 7-segment display decoders. You will need to use three select lines for each of the 6-to-1 multiplexers: connect the select lines to switches SW_{9-7}.

SW_{9-7}	Character pattern					
000				d	E	1
001			d	E	1	
010		d	E	1		
011	d	E	1			
100	E	1				d
101	1				d	E

Table 3. Rotating the word dE1 on six displays.

Perform the following steps:

1. Create a new Quartus project for your circuit.
2. Include your Verilog module in the Quartus project. Connect the switches SW_{9-7} to the select inputs of each instance of the multiplexers in your circuit. Also connect SW_{5-0} to each instance of the multiplexers as required to produce the patterns of characters shown in Table 3. (Hint: for some inputs of the multiplexers you will want to select the 'blank' character.) Connect the outputs of your multiplexers to the 7-segment displays *HEX5*, . . ., *HEX0*.
3. Include the required pin assignments for the DE1-SoC board for all switches, LEDs, and 7-segment displays. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper character codes on the switches SW_{5-0} and then toggling SW_{9-7} to observe the rotation of the characters.