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. We will use the switches SW_{15-0} on the BASYS3 board as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.

Part I

The BASYS3 board provides 16 toggle switches, called SW_{15-0}, that can be used as inputs to a circuit, and 18 red lights, called $LEDR_{17-0}$, that can be used to display output values. Figure 1 shows a simple Verilog module that uses these switches and shows their states on the LEDs. Since there are 16 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 16 *LEDR* outputs, which is equivalent to the individual assignments

```
assign LEDR[15] = SW[15];
assign LEDR[14]= SW[14];
assign LEDR[0] = SW[0];
```

The BASYS3 board has hardwired connections between its FPGA chip and the switches and lights. To use SW_{17-0} and $LEDR_{17-0}$ it is necessary to include in your Vivado project the correct pin assignments, which are given in the *BASYS3 User Manual*. For example, the manual specifies that SW_0 is connected to the FPGA pin *N25* and $LEDR_0$ is connected to pin *AE23*. A good way to make the required pin assignments is to import into the software the file called *BASYS3 pin assignments.csv*, which is provided on the *BASYS3 System CD* and in the University Program section of Altera's web site. The procedure for making pin assignments is described in the tutorial *Vivado Introduction using Verilog Design*, which is also available from Xilinx.

The file uses the names *SW*[0] . . . *SW*[17] and *LEDR*[0] . . . *LEDR*[17] for the switches and lights, which is the reason we used these names in Figure 1.

```
// Simple module that connects the SW switches to the LEDR lights
module part1 (SW, LEDR);

    input [17:0] SW;        // toggle switches
    output [17:0] LEDR;     // red LEDs

    assign LEDR = SW;
endmodule
```

Figure 1. Verilog code that uses the BASYS3 board switches and lights.

Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the BASYS3 board.

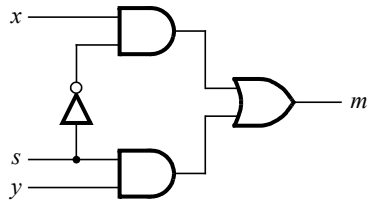
1. Create a new Vivado project for your circuit. Select as the target chip, which is the FPGA chip on the Xilinx board.
2. Create a Verilog module for the code in Figure 1 and include it in your project.

3. Include in your project the required pin assignments for the BASYS3 board, 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

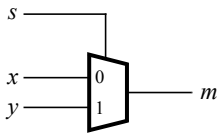
Figure 2a shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input s. If $s = 0$ the multiplexer's output m is equal to the input x, and if $s = 1$ the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows its circuit symbol.



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

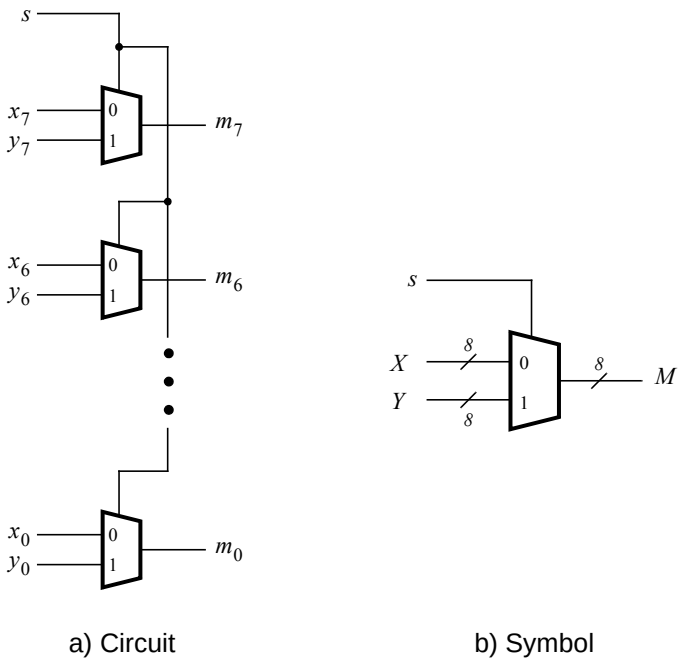


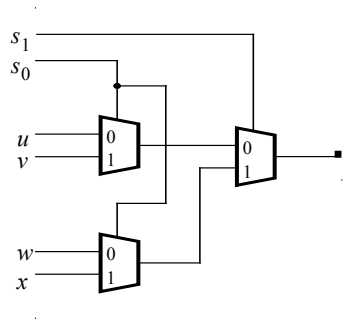
Figure 3. An eight-bit wide 2-to-1 multiplexer.

1. Create a new Vivado project for your circuit.
2. Include your Verilog file for the eight-bit wide 2-to-1 multiplexer in your project. Use switch SW_{17} on the BASYS3 board as the s input, switches SW_{7-0} as the X input and SW_{15-8} as the Y input. Connect the *SW* switches to the red lights *LEDR* and connect the output M to the green lights $LEDG_{7-0}$.
3. Include in your project the required pin assignments for the BASYS3 board. As discussed in Part I, these assignments ensure that the input ports of your Verilog code will use the pins on the Cyclone II FPGA that are connected to the *SW* switches, and the output ports of your Verilog code will use the FPGA pins connected to the *LEDR* and *LEDG* lights.
4. Compile the project.
5. Download the compiled circuit into the FPGA chip. Test the functionality of the eight-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 our inputs u, v, w, x Part a o f Figure 4 shows how we can build the required 4-to-1 multiplexer by using three 2-to-1 multiplexers. The circuit uses a 3-bit select input $s_2s_1s_0$ a nd implements the truth table shown in Figure 4b. A circuit symbol for this multiplexer is given in part c of the figure.

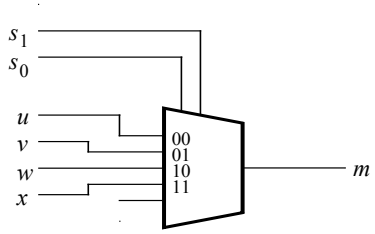
Recall from Figure 3 that an eight-bit wide 2-to-1 multiplexer can be built by using eight instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a two-bit wide 4-to-1 multiplexer. It contains three instances of the circuit in Figure 4a.



a) Circuit

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

b) Truth table



c) Symbol

1.Create a new Quartus II project for your circuit to implement the 2 bit wide 4-to 1 Multiplexer 2. Create a Verilog entity for the three-bit wide 4-to-1 multiplexer. Connect its select inputs to switches SW16ī 15, and use the remaining 13 switches SW8ī 0 to provide the four 2-bit inputs U to Y. Connect the SW switches to the red LEDR and the output to the green LEDG

3.Include in your project the required pin assignments(constraints) for the BASYS3 board and Compile the project. 4.Download the compiled circuit into the FPGA chip.

4.Test the functionality of the three-bit wide 4-to-1 multiplexer by toggling the switches and observing the LEDs. Ensure that each of the inputs U to Y can be properly selected as output M .

Part IV

Figure 6 shows a *7-segment* decoder module that has the three-bit input $c_2c_1c_0$. 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 $c_2c_1c_0$. To keep the design simple, only four characters are included in the table .

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 logic functions that represent circuits needed to activate each of the seven segments. Use only simple Verilog **assign** statements in your code to specify each logic function using a Boolean expression.

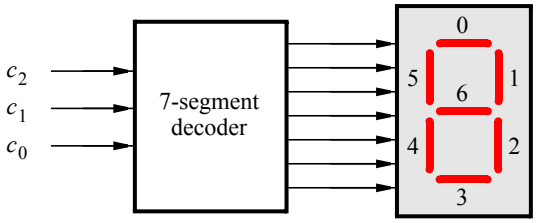


Figure 6. A 7-segment decoder.

$c_2c_1c_0$	Character
000	F
001	P
010	G
011	A
100	
101	
110	
111	

Table 1. Character codes.

Perform the following steps:

Create a Vivado project for your circuit and implement it in the Xilinx FPGA board to dispaly one fo the four characters using a Verilog code

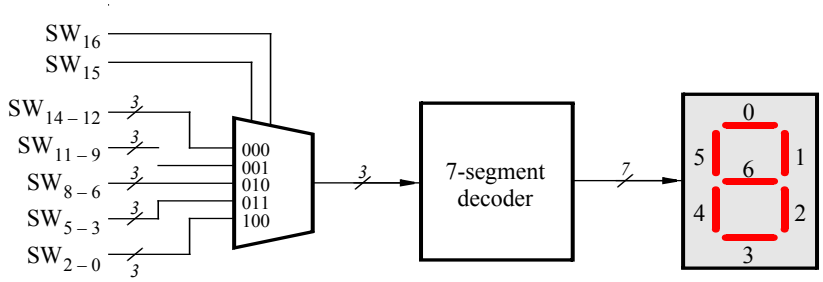


Figure 7. A circuit that can select and display one of four characters.

```
module  part5 (SW, HEX0);
    input [14:0] SW;          // toggle switches
    output [0:6] HEX0;        // 7-seg displays

    wire [2:0] M;

    mux_2bit_4to1 M0 (SW[16:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M);
    char_7seg H0 (M, HEX0);
endmodule

// implements a 2-bit wide 4-to-1 multiplexer
module mux_2bit_4to1 (S, U, V, W, X);
    input [1:0] S, U, V, W, X;
    output [2:0] M;

    . . .  code not shown

endmodule

// implements a 7-segment decoder for F, P, G, A
 module char_ 7seg (C, Display);
    input [1:0] C;            // input code
    output [0:6] Display;     // output 7-seg code

    . . . code not shown

endmodule
```

Figure 8. outline of the Verilog code for the circuit in Figure 7.

Perform the following steps.

1. Create a new Vivado project for your circuit.
2. Include your Verilog module in the Vivado project. Connect the switches $SW_{16\text{-}15}$ to the select inputs of each of the five instances of the three-bit wide 5-to-1 multiplexers. Also connect $SW_{14\ 0}$ to each instance of the multiplexers as required to produce the patterns of characters .Connect the outputs of the multiplexers to the 7-segment displays *HEX3*, *HEX2*, *HEX1*, and *HEX0*.
3. Include the required pin assignments as per the constraints .XDC file for the BASYS3 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_{14-0} and then toggling SW_{16-15} to observe the rotation of the characters.