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_{9-0} on the DE1-SoC board as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.

Part I

The DE1-SoC, DE0-CV, and DE2-115 boards provide the following switches and lights:

Board	Number of Switches	Name of Switches	Number of Lights	Name of Lights
DE0-CV	10	SW_{9-0}	10	$LEDR_{9-0}$
DE1-SoC	10	SW_{9-0}	10	$LEDR_{9-0}$
DE2-115	18	SW_{17-0}	18	$LEDR_{17-0}$

Table 1: DE-series board peripherals

The switches can be used for input, and the lights can be used as output devices. Figure 1 shows a simple Verilog module that uses these switches and shows their states on the LEDs. Since there are multiple switches and lights it is convenient to represent them as vectors in the VHDL code, as shown. We have used a single assignment statement for all *LEDR* outputs, which is equivalent to the individual assignments:

```
LEDR(9) <= SW(9);
LEDR(8) <= SW(8);
. . .
LEDR(0) <= SW(0);
```

The DE-series boards have hardwired connections between its FPGA chip and the switches and lights. To use the switches and lights it is necessary to include in your Quartus II project the correct pin assignments, which are given in your board's user manual. For example, the DE1-SoC manual specifies that 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 II software the pin assignment file for your board, which is provided on the University Program section of Altera's web site. The procedure for making pin assignments is described in the tutorial *Quartus II Introduction using Verilog Design*, which is also available from Altera.

It is important to realize that the pin assignments in the file are useful only if the pin names that appear in these files are exactly the same as the port names used in your Verilog module. The files uses the names *SW*[0] . . . *SW*[*n* - 1] and *LEDR*[0] . . . *LEDR*[*n* - 1], where *n* is the number of lights and switches your board has. This is the reason that we have used these names in Figure 1.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

- - Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
   PORT ( SW      : IN      STD_LOGIC_VECTOR(9 DOWNTO 0);
          LEDR  : OUT     STD_LOGIC_VECTOR(9 DOWNTO 0)); - - red LEDs
END part1;

ARCHITECTURE Behavior OF part1 IS
BEGIN
   LEDR <= SW;
END Behavior
```

Figure 1: VHDL code that uses the DE1-SoC and DE0-CV board's switches and lights. (Part *a*)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

- - Simple module that connects the SW switches to the LEDR lights
ENTITY part1 IS
   PORT ( SW      : IN      STD_LOGIC_VECTOR(17 DOWNTO 0);
          LEDR  : OUT     STD_LOGIC_VECTOR(17 DOWNTO 0)); - - red LEDs
END part1;

ARCHITECTURE Behavior OF part1 IS
BEGIN
   LEDR <= SW;
END Behavior
```

Figure 1. VHDL code that uses the DE2-115 board's switches and lights. (Part *b*)

Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the DE-series boards.

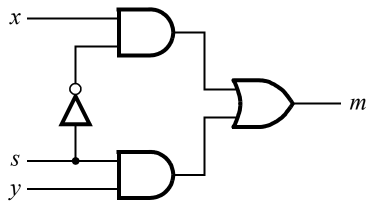
1. Create a new Quartus II project for your circuit. Select the target chip, that corresponds to your DE-series board. Refer to Table 2 for a list of devices.
2. Create a VHDL entity for the code in Figure 1 and include it in your project.
3. Include in your project the required pin assignments for your DE-series board, as discussed above. Compile the project.
4. Download the compiled circuit into the FPGA chip by using the Quartus II Programmer tool (the procedure for using the Programmer tool is described in the tutorial *Quartus II Introduction*). Test the functionality of the circuit by toggling the switches and observing the LEDs.

Board	Device Name
DE0-CV	Cyclone IVE 5CEBA4F23C7
DE1-SoC	Cyclone V SoC 5CSEMA5F31C6
DE2-115	Cyclone IVE EP4CE115F29C7

Table 2: DE-series FPGA device names

Part II

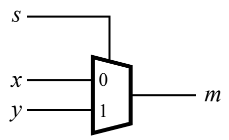
Figure 2a shows a sum-of-products circuit that implements a 2-to-1 *multiplexer* with a select input s. If $s = 0$ the multiplexer's output m is equal to the input x, and if $s = 1$ the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows its circuit symbol.



a) Circuit

s	m
0	x
1	y

b) Truth table



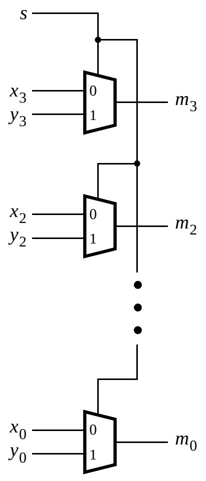
c) Symbol

Figure 2: A 2-to-1 multiplexer.

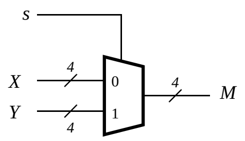
The multiplexer can be described by the following VHDL statement:

```
m <= (NOT (s) AND x) OR (s AND y);
```

You are to write a VHDL entity 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.



a) Circuit



b) Symbol

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

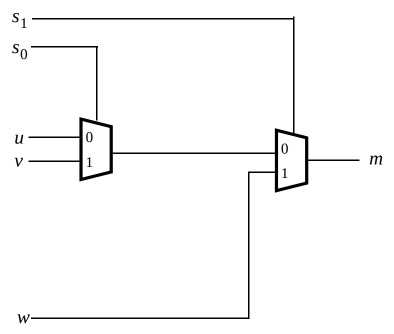
Perform the steps listed below.

1. Create a new Quartus II project for your circuit.
2. Include your VHDL file for the four-bit wide 2-to-1 multiplexer in your project. Use switch *SW*$_9$ as the s input, switches *SW*$_{3-0}$ as the X input and *SW*$_{7-4}$ as the Y input. Display the value of the input s on *LEDR*$_9$, connect the output M to *LEDR*$_{3-0}$, and connect the unused LEDR lights to the constant value 0.
3. Include in your project the required pin assignments for your DE-series board. As discussed in Part I, these assignments ensure that the ports of your Verilog code will use the pins on the FPGA chip that are connected to the *SW* switches and *LEDR* lights.
4. Compile the project, and then download the resulting 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.

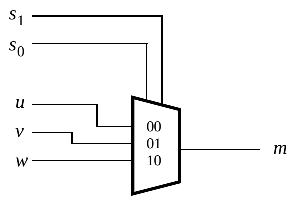
Recall from Figure 3 that a four-bit wide 2-to-1 multiplexer can be built by using four instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a two-bit wide 3-to-1 multiplexer. It contains two instances of the circuit in Figure 4a.



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.

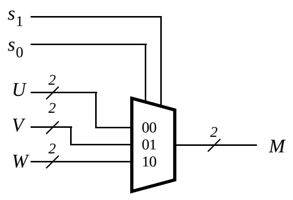


Figure 5: A two-bit wide 3-to-1 multiplexer.

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

1. Create a new Quartus II project for your circuit.
2. Create a VHDL entity for the two-bit wide 3-to-1 multiplexer. Connect its select inputs to switches SW_{9-8}, and use switches SW_{5-0} to provide the three 2-bit inputs U to W. Connect the output M to the red lights $LEDR_{1-0}$.
3. Include in your project the required pin assignments for your DE-series 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

The objective of this part is to display a character on a 7-segment display. The specific character displayed depends on a two-bit input. Figure 6 shows 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 3 lists the characters that should be displayed for each valuation of c_1c_0 for your DE-series board. 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 logic functions that represent circuits needed to activate each of the seven segments. Use only simple VHDL assignment statements in your code to specify each logic function using a Boolean expression.

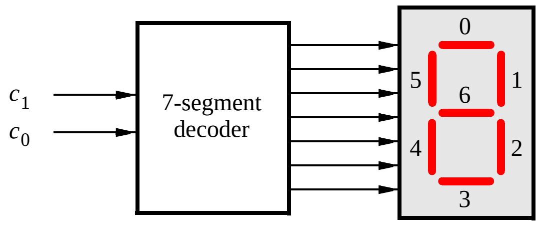


Figure 6: A 7-segment decoder.

c_1c_0	DE0-CV Characters	DE1-SoC Characters	DE2-115 Characters
00	d	d	d
01	E	E	E
10	0	1	2
11			

Table 3: Character codes for the DE-series boards.

Perform the following steps:

1. Create a new Quartus II project for your circuit.
2. Create a VHDL entity for the 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 your DE-series board. The segments in this display are called $HEX0_0$, $HEX0_1$, . . ., $HEX0_6$, corresponding to Figure 6. You should declare the 7-bit port

   HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6);

   in your VHDL code so that the names of these outputs match the corresponding names in your board's user manual and pin assignment file.
3. After making the required pin assignments, compile the project.
4. Download the compiled 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, 'blank' and 0, 1, or 2 depending on your DE-series board. The character codes are set according to Table 3 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. Note that 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 7-segment displays that is composed of the characters in Table 3, 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 4.

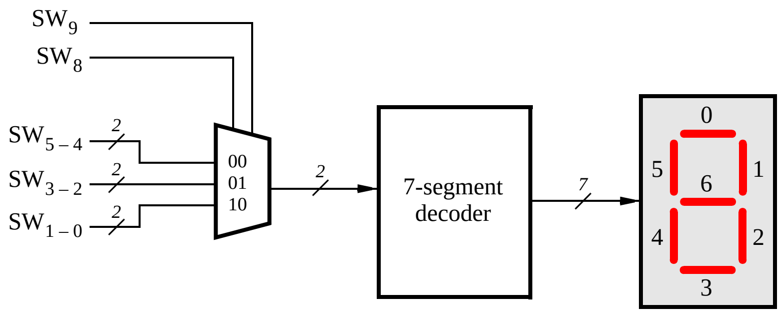


Figure 7: A circuit that can select and display one of three characters.

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

Table 4: Rotating the word dE1 on three displays.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part5 IS
   PORT ( SW      : IN      STD_LOGIC_VECTOR(9 DOWNTO 0);
          LEDR : OUT   STD_LOGIC_VECTOR(9 DOWNTO 0));
          HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6) );
END part5;

ARCHITECTURE Behavior OF part5 IS
   COMPONENT mux_2bit_3to1
      PORT ( S, U, V, W      : IN        STD_LOGIC_VECTOR(1 DOWNTO 0);
             M               : OUT     STD_LOGIC_VECTOR(1 DOWNTO 0));
   END COMPONENT;
   COMPONENT char_7seg
      PORT ( C               : IN        STD_LOGIC_VECTOR(1 DOWNTO 0);
             Display         : OUT     STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;
   SIGNAL M0 : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
   U0: mux_2bit_3to1 PORT MAP (SW(9 DOWNTO 8), SW(5 DOWNTO 4), SW(3 DOWNTO 2),
      SW(1 DOWNTO 0), M0);
   H0: char_7seg PORT MAP (M0, HEX0);
   . . .
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- implements a 2-bit wide 3-to-1 multiplexer
ENTITY mux_2bit_3to1 IS
   PORT ( S, U, V, W      : IN        STD_LOGIC_VECTOR(1 DOWNTO 0);
          M               : OUT     STD_LOGIC_VECTOR(1 DOWNTO 0));
END mux_2bit_3to1;

ARCHITECTURE Behavior OF mux_2bit_3to1 IS
   . . . code not shown

END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY char_7seg IS
   PORT ( C               : IN        STD_LOGIC_VECTOR(1 DOWNTO 0);
          Display         : OUT     STD_LOGIC_VECTOR(0 TO 6));
END char_7seg;

ARCHITECTURE Behavior OF char_7seg IS
   . . . code not shown

END Behavior;
```

Figure 8: VHDL code for the circuit in Figure 7

Perform the following steps.

1. Create a new Quartus II project for your circuit.
2. Include your VHDL entity in the Quartus II project. 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. Also connect SW_{5-0} to each instance of the multiplexers as required to produce the patterns of characters shown in Table 2. Connect the SW switches to the red lights LEDR, and connect the outputs of the three multiplexers to the 7-segment displays *HEX2*, *HEX1*, and *HEX0*.
3. Include the required pin assignments for your DE-series 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_{9-8} to observe the rotation of the characters.

Part VI

Extend your design from Part V so that is uses all 7-segment displays on your DE-series board. board. Your circuit needs to display a three letter word on three displays while the rest of the displays show a 'blank'. The letters in the word are selected by three two-bit inputs as shown in Table 3. Also implement rotation of this word from right-to-left as shown in Table 5 and Table 6. To do this, you will need to connect two-bit wide 6-to-1 multiplexers to each of six 7-segment display decoders for the DE0-CV and DE1-SoC. For the DE2-115, you will need to connect two-bit wide 8-to-1 multiplexers to each of the eight 7-segment display decoders. You will need to use three select lines for each of the 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 5: Rotating the word dE1 on six displays.

SW_{9-7}	Character pattern							
000						d	E	2
001					d	E	2	
010				d	E	2		
011			d	E	2			
100		d	E	2				
101	d	E	2					
110	E	2						d
111	2						d	E

Table 6: Rotating the word dE2 on eight displays.

Perform the following steps:

1. Create a new Quartus II project for your circuit.
2. Include your VHDL entity in the Quartus II 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 5 or Table 6 depending on your DE-series

board. (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* of the DE0-CV and DE1-SoC or *HEX7*, . . ., *HEX0* for the DE2-115.

3. Include the required pin assignments for your DE-series 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.