

# 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 DE boards as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.

### Part I

Both DE boards provide 10 toggle switches, called  $SW_{9-0}$ , that can be used as inputs to a circuit. The DE1 board has 10 red lights, called  $LEDR_{9-0}$ , and the DE0 board has 10 green lights ( $LEDG_{9-0}$ ) that can be used to display output values. Figure 1 shows a simple VHDL entity that uses these switches and shows their states on the LEDs. Since there are 10 switches and lights it is convenient to represent them as arrays in the VHDL code, as shown. We have used a single assignment statement for all 10  $LEDR$  outputs, which is equivalent to the individual assignments

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

Similar expressions can be written for the  $LEDG$  outputs if you are targeting the DE0 board. Both the DE0 and DE1 boards have hardwired connections between its FPGA chip and the switches and lights. To use  $SW_{9-0}$  and  $LEDR_{9-0}$  or  $LEDG_{9-0}$  it is necessary to include in your Quartus II project the correct pin assignments, which are given in the *DE0* and *DE1 User Manuals*. For example, the DE1 manual specifies that  $SW_0$  is connected to the FPGA pin  $L22$  and  $LEDG_0$  is connected to pin  $U22$ , whereas the DE0 manual specifies these resources are connected to FPGA pins  $J6$  and  $J1$  respectively. A good way to make the required pin assignments is to import into the Quartus II software the file called *COMP3222\_[DE0|DE1]\_pin\_assignments.qsf*, which is provided in the *Labs* section of the course web site. The procedure for making pin assignments is described in the tutorial *Quartus II Introduction using VHDL Design*, which is also available from Altera.

It is important to realize that the pin assignments in the *COMP3222\_[DE0|DE1]\_pin\_assignments.qsf* file are useful only if the pin names given in the file are exactly the same as the port names used in your VHDL entity. The file uses the names  $SW[0] \dots SW[9]$  and  $LEDR[0] \dots LEDR[9]$  and  $LEDG[0] \dots LEDG[9]$  for the switches and lights, which is the reason we used these names in Figure 1 (note that the Quartus II software uses  $[ ]$  square brackets for array elements, while the VHDL syntax uses  $( )$  round brackets).

```
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 DOWNT0 0);
          LEDR     : OUT     STD_LOGIC_VECTOR(9 DOWNT0 0)); -- NOTE: use LEDG on the DE0
END part1;

ARCHITECTURE Behavior OF part1 IS
BEGIN
    LEDR <= SW; -- NOTE: use LEDG <= SW when using the DE0 board
END Behavior
```

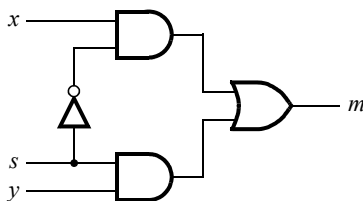
Figure 1. VHDL code that uses the **DE1 board** switches and lights.

Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the DE0/DE1 board.

1. Create a new Quartus II project for your circuit. If you are using the DE0 board, select the Cyclone III EP3C16F484C6 as the target FPGA, otherwise select Cyclone II EP2C20F484C7 as the target FPGA, if you are using the DE1 board.
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 the DE board you are using, as discussed above and demonstrated in the introductory video for this lab. 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

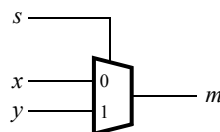
Figure 2a shows a sum-of-products circuit that implements a 2-to-1 multiplexer with a select input  $s$ . If  $s = 0$  the multiplexer's output  $m$  is equal to the input  $x$ , and if  $s = 1$  the output is equal to  $y$ . Part b of the figure gives a truth table for this multiplexer, and part c shows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

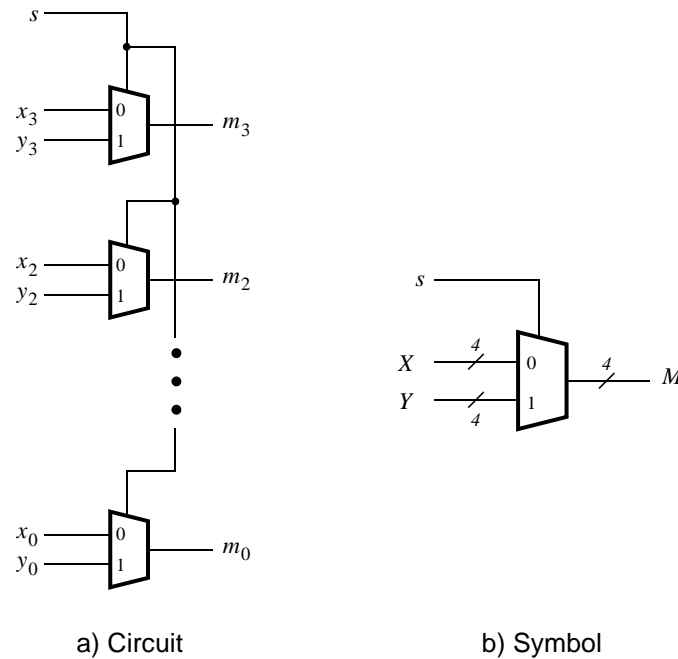


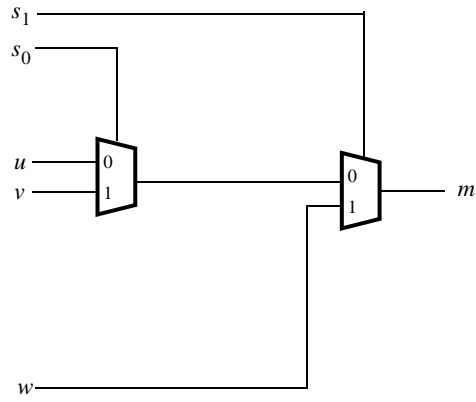
Figure 3. A four-bit wide 2-to-1 multiplexer.

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  $SW_9$  on the DE board as the  $s$  input, switches  $SW_{3-0}$  as the  $X$  input and  $SW_{7-4}$  as the  $Y$  input. Connect the output  $M$  to the green lights  $LEDG_{3-0}$ , which both boards have.
3. Include in your project the required pin assignments for the DE board you are using. As discussed in Part I, these assignments ensure that the input ports of your VHDL code will use the pins on the Cyclone II/III FPGA that are connected to the  $SW$  switches, and the output ports of your VHDL code will use the FPGA pins connected to the  $LEDG$  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  $s_1s_0$  and implements the truth table shown in Figure 4*b*. A circuit symbol for this multiplexer is given in part *c* of the figure.

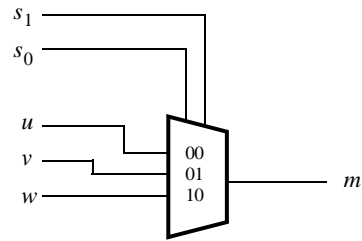
Recall from Figure 3 that a four-bit wide 2-to-1 multiplexer can be built by using four instances of a 2-to-1 multiplexer. Figure 5 applies this concept to define a two-bit wide 3-to-1 multiplexer. It contains two instances of the circuit in Figure 4*a*.



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.

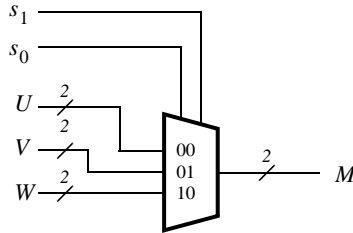



Figure 5. A two-bit wide 3-to-1 multiplexer.

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 green lights  $LEDG_{1-0}$ .
3. Include in your project the required pin assignments for your DE 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 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 1 lists the characters that should be displayed for each valuation of  $c_1c_0$ . To keep the design simple, only three characters are included in the table (plus the 'blank' character, which is selected for codes 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 VHDL entity 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.**

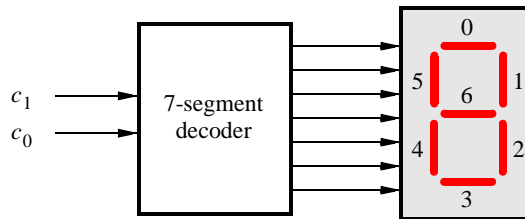


Figure 6. A 7-segment decoder.

$c_1c_0$	Character
00	d
01	E
10	1
11	

Table 1. Character codes.

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 the DE board. The segments in this display are called  $HEX0_0, HEX0_1, \dots, HEX0_6$ , corresponding to Figure 6. You should declare the 7-bit port

$HEX0 : OUT \text{ STD\_LOGIC\_VECTOR}(0 \text{ TO } 6);$

in your VHDL code so that the names of these outputs match the corresponding names in the *COMP3222\_[DE0|DE1]\_pin\_assignments.qsf* file.

3. After making the required DE1 board 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 any of 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 VHDL 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 four 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.

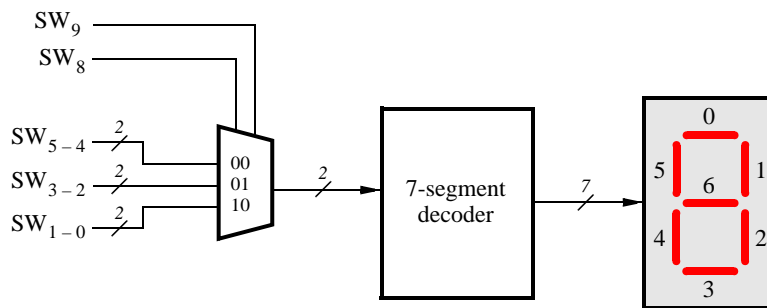


Figure 7. A circuit that can select and display one of three characters.

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part5 IS
    PORT ( SW      : IN      STD_LOGIC_VECTOR(9 DOWNT0 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 DOWNT0 0);
              M                : OUT     STD_LOGIC_VECTOR(1 DOWNT0 0));
    END COMPONENT;
    COMPONENT char_7seg
        PORT ( C              : IN      STD_LOGIC_VECTOR(1 DOWNT0 0);
              Display         : OUT     STD_LOGIC_VECTOR(0 TO 6));
    END COMPONENT;
    SIGNAL M : STD_LOGIC_VECTOR(1 DOWNT0 0);
BEGIN
    M0: mux_2bit_3to1 PORT MAP (SW(9 DOWNT0 8), SW(5 DOWNT0 4), SW(3 DOWNT0 2),
                               SW(1 DOWNT0 0), M);
    H0: char_7seg PORT MAP (M, 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 DOWNT0 0);
          M                : OUT     STD_LOGIC_VECTOR(1 DOWNT0 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 DOWNT0 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.

$SW_9 SW_8$	Character pattern		
00	d	E	1
01	E	1	d
10	1	d	E

Table 2. Rotating the word dE1 on three 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  $SW_{9-8}$  to the select inputs of each of the three instances of the three-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 outputs of the three multiplexers to the 7-segment displays *HEX2*, *HEX1*, and *HEX0*.
3. Include the required pin assignments for your DE 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 it uses all four 7-segment displays on the DE1 board. Your circuit should be able to display words with three (or fewer) characters on the four displays, and rotate the displayed word when the switches  $SW_{9-8}$  are toggled. If the displayed word is dE1, then your circuit should produce the patterns shown in Table 3.

$SW_9 SW_8$	Character pattern		
00	d	E	1
01	d	E	1
10	E	1	d
11	1	d	E

Table 3. Rotating the word dE1 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  $SW_{9-8}$  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 *HEX3*, . . . , *HEX0*.
3. Include the required pin assignments for your DE 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.

Copyright ©2011 Altera Corporation.

Modified in 2020 for COMP3222@UNSW to enable use with DE0 and DE1 boards.