Digital Logic Design Project

Mahdi Shabany
Electrical Engineering Department
Sharif University of Technology

Fall 2011

1. Introduction

The purpose of this Project is to introduce you to the software tools and hardware that are the Altera Quartus II Computer Aided Design (CAD) system and the DE2 Board. You will need to install that software on your computer before you can start this project.

You will be learning to design hardware that will go into a kind of programmable logic chip call a Field-Programmable Gate Array, or FPGA. The FPGA chip is mounted on a board called the Altera DE2 Development and Education board, pictured below (Fig. 1).

The board contains many useful features for learning about logic circuits, including simple input and output mechanisms like switches and lights, and more complicated features like audio and video devices. This assignment will use only the switches and lights that are provided on the bottom edge of the board, as illustrated below, but other assignments will utilize more advanced features. A detailed description of the board can be found on the course website at: http://ee.sharif.edu/~asic

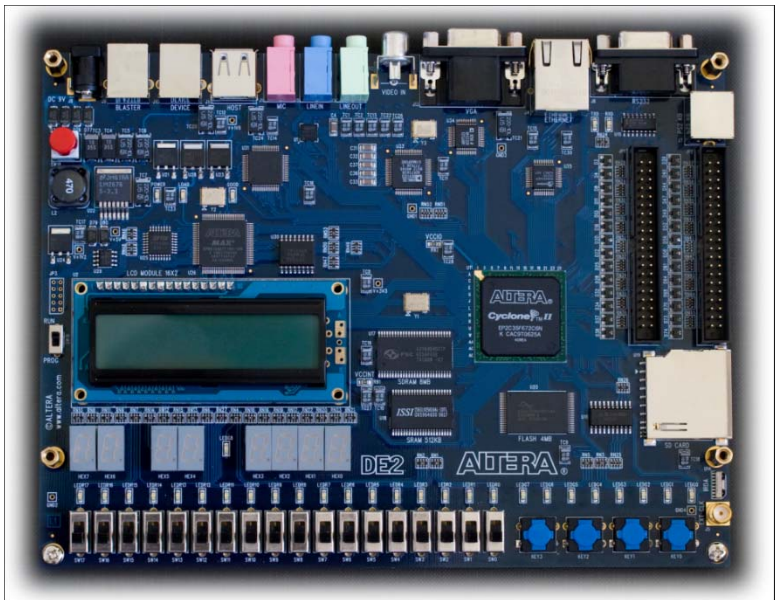


Fig. 1. The Altera DE2 Development and Education board.

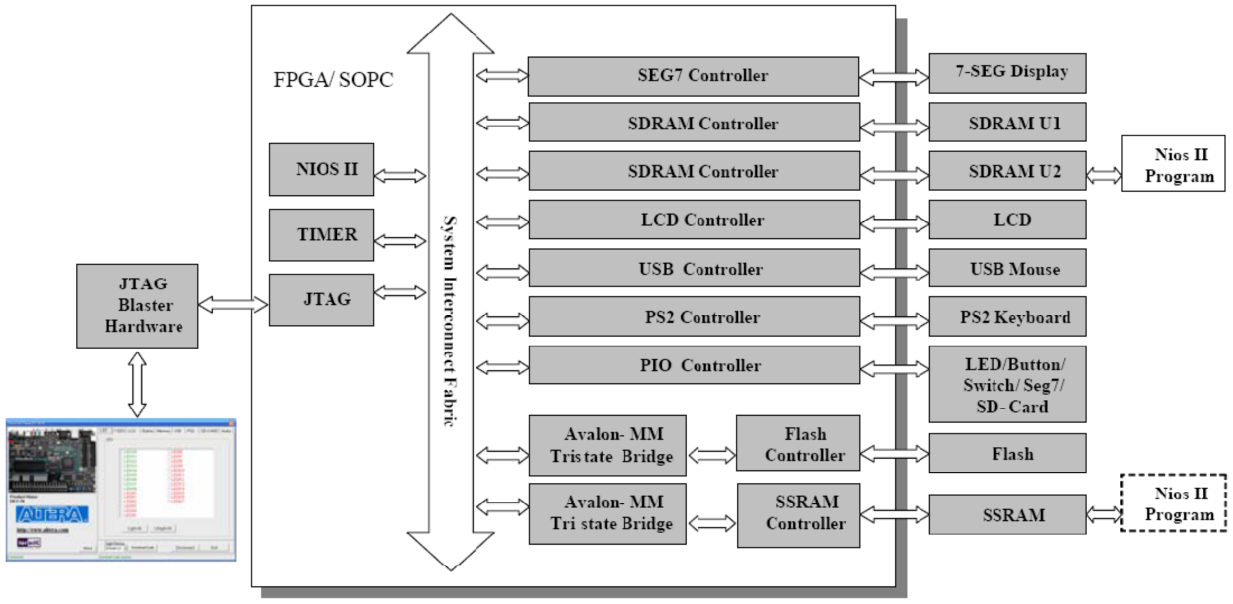


Fig. 2. The block diagram of the DE2-70 Control Panel.

Note:

The blue boxes identified by the word "DE2" are the parts that should be done on-line on the board. The rest can be done at home.

Part I:

For this part you need to do three important tutorials indicated in the following. Make sure you perform every single step specified in them as they are the foundation of all you will do in this project.

Tutorial 1. Do the tutorial called *Using Quartus II CAD Software*, which is in fact the Appendix B of the book Fundamentals of Digital Logic with Verilog Design, 2nd Edition. This tutorial is available on-line on the course website. This tutorial describes the basics of how Quartus II helps a designer describe circuits and check them for correctness. It shows two ways of creating circuits: either using schematic capture, in which you "draw" a picture of the circuit, or describing a circuit in language form. Please do the same circuit in both ways so that you understand the schematic form and the textual form describing exactly the same thing. We will use the textual form often because it is far more powerful and quicker.

Tutorial 2. Do the first part of the tutorial called *Implementing Circuits in Altera Devices* (Appendix C.1), available on the course website.

Tutorial 3. Do the tutorial *Physical Implementation in an FPGA* (Appendix D), available on the course website.

Part II:

Create a simple circuit to connect four switches to four lights on the Altera board, by extending the following Verilog code:

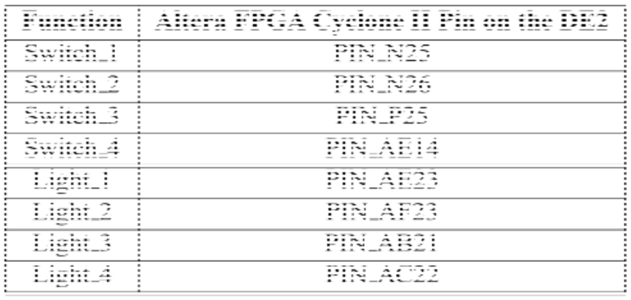
```
// Simple module that connects the SW switches to the LEDR lights
module part3 (Switch 1, Switch 2, Switch 3, Switch 4, Light 1, Light 2, Light 3, Light 4);
input Switch 1, Switch 2, Switch 3, Switch 4; // toggle switches
output Light 1, Light 2, Light 3, Light 4; // lights

// Your code goes here

endmodule
```

On the board the FPGA chip that your designs will be programmed into has hardwired connections between the pins of the FPGA chip and the switches and lights on the board. Note that these lights and switches are connected to circuits that allow them to generate 1s and 0s as inputs to your FPGA circuit (for the switches) and to turn on and off in response to digital 1s and 0s that are outputs from your circuit. To use switches and lights you have to tell Quartus II which of the input/output signals in your Verilog code should be connected to which pins on the FPGA chip and which are connected to the switches or lights. The procedure for doing this is called *pin assignment* and was covered in the tutorial 3 of Part I above. Table 1 below indicates which pins (which are referred to by names such as PIN N25) of the FPGA are connected to which switches and lights on the board. There are 18 total switches and lights on the board, but the table only lists 4 of each needed for this part.

NOTE:

Please only connect the SW0…SW3 to LEDR0…LEDR3 on the Altera board .Do NOT try to connect the other switches and lights.

Table 1. Pin assignment table for lights and switches in Part II.



Function	Altera FPGA Cyclone II Pin on the DE2
Switch_1	PIN_N25
Switch_2	PIN_N26
Switch_3	PIN_P25
Switch_4	PIN_AE14
Light_1	PIN_AE23
Light_2	PIN_AF23
Light_3	PIN_AB21
Light_4	PIN_AC22

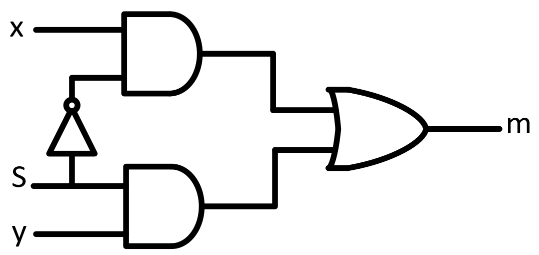
Do the following steps to download and test the circuit in Part II:

1. Create a Verilog module for the code above and include it in your project. Make sure to complete the code by adding the assignment statements for the lights.
2. Use Quartus II to make the pin assignments shown in Table 1 as described in tutorial 3. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by flipping the switches and observing the lights.

Part III:

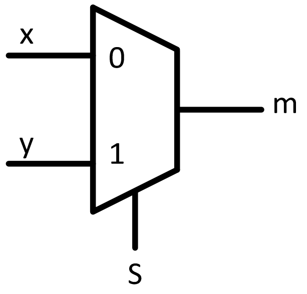
Fig. 3a shows a sum-of-products circuit that implements a 2-to-1 multiplexer (MUX) of inputs x and y select input s and output m. If s = 0 the multiplexer’s output m is equal to the input x, and if s = 1 the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows the schematic circuit symbol.



a) Circuit

S	m
0	x
1	y

b) Truth Table



c) Symbol

Fig. 3. A 2-to-1 Multiplexer.

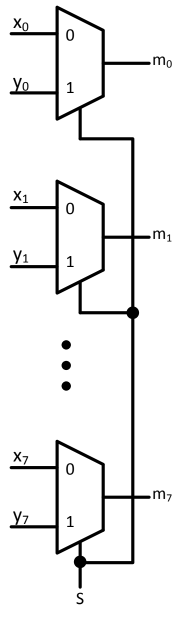
The multiplexer can be described by the following Verilog statement:

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

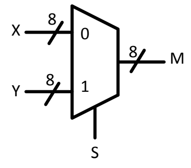
You are to design a circuit, using Verilog, which is a more complex version of a multiplexer. Rather than select between two signals, your circuit is to select between two sets of *eight* signals, as illustrated in Fig. 4a. This circuit has two eight-bit inputs, X and Y , and produces the eight-bit output 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 Fig. 4b, in which X, Y , and M are depicted as eight-bit wires.

Do the following steps to download and test the circuit in Part III:

1. Write your Verilog file for the eight-bit wide 2-to-1 multiplexer in your project. Use switch SW17 on the board as the s input, switches SW7−0 as the X input and SW15−8 as the Y input. Connect the output M to the green lights on the board, called LEDG7−0.



a) Circuit



b) Symbol

Fig. 4. An eight-bit wide 2-to-1 multiplexer.

2. Include in your project the required pin assignments for the board using the pin assignment file as described above. As discussed in previous parts, these assignments ensure that the inputs declared in your Verilog code will use the pins on the Cyclone II FPGA that are connected to the *SW* switches and LEDRs, and the outputs of your Verilog code will use the FPGA pins connected to the LEDG lights. Compile the project.

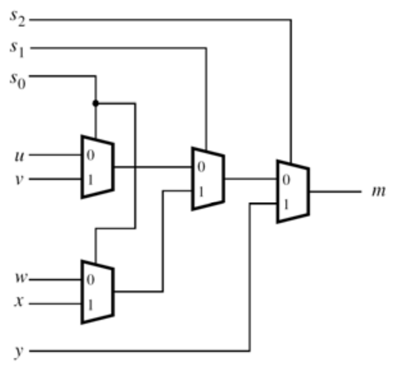
NOTE:

Always Connect the inputs to the LEDRs in Part III and Part IV.

DE2: 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 IV:

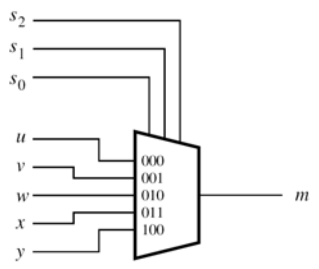
In Fig. 3 we showed a 2-to-1 multiplexer that selects between the two inputs *x* and *y*. For this part consider a circuit in which the output *m* has to be selected from *five* inputs u, v, w, x, and y. Part a of Fig. 5 shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers. The circuit uses a 3-bit select input {s2,s1,s0} and implements the truth table shown in Fig. 5b. A circuit symbol for this multiplexer is given in Fig. 5c. Recall from Fig. 4 that an eight-bit wide 2-to-1 multiplexer can be built by using eight 2-to-1 multiplexers. Fig. 6 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the 5-to-1multiplexer circuit in Fig. 5.



a) Circuit

s_2 s_1 s_0	m
0 0 0	u
0 0 1	v
0 1 0	w
0 1 1	x
1 0 0	y
1 0 1	y
1 1 0	y
1 1 1	y

b) Truth table



c) Symbol

Fig. 5. A 5-to-1 MUX.

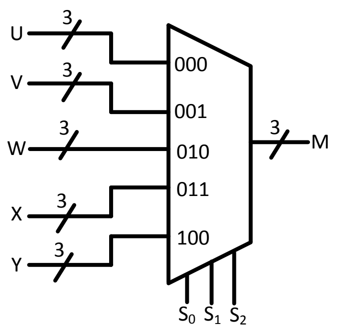


Fig. 6. A three-bit wide 5-to-1 multiplexer.

Do the following steps to download and test the circuit in Part IV:

1. Create a Verilog module for the three-bit wide 5-to-1 multiplexer (called *mux_3bit_5to1.v)*. Connect its select inputs to switches SW17−15, and use the remaining 15 switches on the Altera board (SW14−0) to provide the five 3-bit inputs U through Y. Connect the output M to the green lights *LEDG*2−0.
2. Include in your project the required pin assignments for the board. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the three-bit wide 5-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 the output M.

Part V - Design of a 7-Segment Decoder Circuit

Fig. 7 shows a 7-segment character display controlled by a logic circuit. These displays are common on digital watches and various electrical devices. This circuit is called a 7-segment *decoder* and it has a three-bit input {c_2, c_1, c_0} (which you will connect to switches on the board), and 7 outputs that turn on or off the 7 different segments in the character display. The three inputs present a code that are translated by the circuit into 7 outputs to create a particular character by lighting up some of the lights on the display.

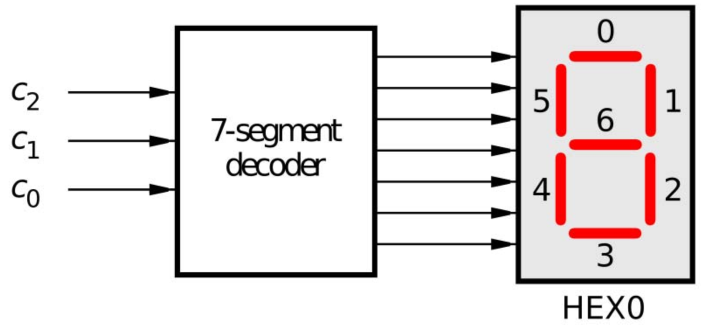


Fig. 7. A 7-segment decoder.

You are to design a circuit that is able to decode and display *the last 5 digits of your student number*. An example mapping is given in Table 2, which lists the characters that should be displayed for each value of {c_2, c_1, c_0}. To keep the design simple, you only need to decode five numbers, represented using the codes {c_2, c_1, c_0} = 000, 001, 010, 011, and 100. The fifth code should produce a blank character with all of the lights off. Since we don't care what is displayed for the remaining code values (110 and 111) you can be treat these as don't care values. The seven segments in the display are identified by the numbers 0 to 6 shown in Fig. 7. Each segment is lit up by driving it to the logic value 0 (which is a little opposite of what you might expect). You must implement a separate logic function that controls each segment in the display. Use a Karnaugh map to determine the minimal (optimal) sum-of-products expressions for each of these 7 outputs.

Create a Verilog module for your 7-segment decoder circuit (call it *char_7seg.v)*. In your Verilog code, use only simple assign statements to specify each of the sum-of-products expressions generated from your Karnaugh maps. In your Verilog code assign the {c_2, c_1, c_0} inputs to switches SW2–0 on the board, and assign the outputs of the decoder to the HEX0 display on the board. The segments in this display are called HEX00, HEX01, . . ., HEX06, corresponding to Fig. 7 using the pin assignment file provided in the previous assignment.

Note that in the pin assignments file the HEX0 segments are declared as an array. To use the same names in your Verilog code, your should declare the seven-bit output port as

```
output [0:6] HEX0;
```

By using this 7-bit array, the names of the seven segments in your code will match the names that are used for these segments in the pin assignment file.

Table 2. Character codes (for the case where your student number ends in '12345')

$c_2c_1c_0$	Character
000	1 (replace with last digit of your student number)
001	2 (replace with 2nd last digit of your student number)
010	3 (replace with 3rd last digit of your student number)
011	4 (replace with 4th last digit of your student number)
100	5 (replace with 5th last digit of your student number)
101	'blank'
110	'Don't Care'
111	'Don't Care'

DE2: Compile and download your circuit onto the board. Test the functionality of the circuit by toggling the SW2−0 switches and observing the 7-segment display HEX0.

Part VI - Selecting the Character to be Displayed

Consider the circuit shown in Fig. 8. It uses a three-bit wide 5-to-1 multiplexer (just like the one you built above) to enable the selection of five characters that are displayed on a 7-segment display. Using the 7-segment decoder from Part V, this circuit can display any of the five digits and 'blank'. The character codes are set according to Table 1 by using the switches SW14−0, and a specific character is selected for display by setting the switches SW17−15.

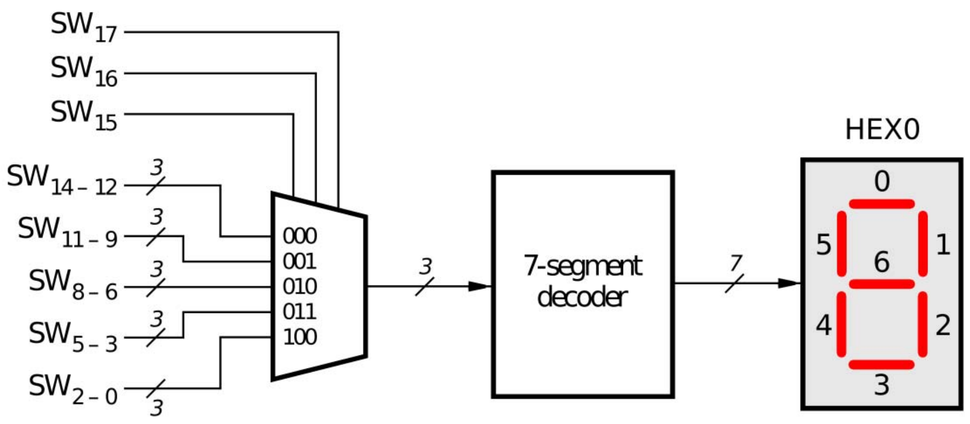


Fig. 8. A circuit that can select and display one of five characters

To build this circuit you can reuse two smaller designs already created: the three-bit 5-to-1 multiplexer and the 7-segment decoder that you design in previous parts. Your task is to create a new *top-level* design that implements the circuit in Fig. 8 using the multiplexer and decoder as subcircuits. You are to design this top-level circuit twice, using the following approach:

Create the top-level design by writing complete Verilog code for the circuit in Fig. 8. An outline of this code, which shows how to include the multiplexer and decoder subcircuits in the top-level Verilog module, is given in Fig. 9. Complete this section by performing the following steps:

a) Use the modified codes in the previous sections, i.e., *mux_3bit_5to1.v* and *char_7seg.v*.
b) Use the File > New command to create a new Verilog file named *hierarchy_verilog.v*. Type your top-level Verilog code, following the style of code shown in Fig. 9, into this file.
c) Include the required pin assignments for the board switches and 7-segment display. Compile the project.
d) Simulate the compiled circuit using a timing simulation in the Quartus II Simulator.

```
module hierarchy_verilog (SW, HEX0);
input [17:0] SW; // toggle switches
output [0:6] HEX0; // 7-seg displays
wire [2:0] M;
    mux_3bit_5to1 M0 (SW[17:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M);
    char_7seg H0 (M, HEX0);
endmodule

// implements a 3-bit wide 5-to-1 multiplexer
module mux_3bit_5to1 (S, U, V, W, X, Y, M);
input [2:0] S, U, V, W, X, Y;
output [2:0] M;

. . . code not shown

endmodule

// implements a 7-segment decoder for each character
module char_7seg (C, Display);
input [2:0] C; // input code
output [0:6] Display; // output 7-seg code

. . . code not shown

endmodule
```

Fig. 9. Verilog code for the circuit in Fig. 8.

DE2: Download your circuit onto the board. Test the functionality of the circuit by setting the proper character codes on the switches SW14−0 and then toggling SW17−15 to observe the display of characters.

Part VII - Displaying and Rotating a Sequence

In this part you will reuse your previous designs, and extend the code from part VIso that it uses **five** 7-segment displays, rather than just one. You will need to use five instances (copies) of the subcircuits from Part VI. The purpose of your circuit is to display a sequence on the five displays, and be able to manually rotate this sequence in a circular fashion across the displays when the switches SW17−15 are toggled. As an example, if the displayed sequence is 12345, then your circuit should produce the output patterns illustrated in Table 3. The sequence must be the last five digits of your student number.

Table 3. Manually rotating the sequence 12345 on five displays.

$SW_{17}\ SW_{16}\ SW_{15}$	HEX4	HEX3	HEX2	HEX1	HEX0
000	1	2	3	4	5
001	2	3	4	5	1
010	3	4	5	1	2
011	4	5	1	2	3
100	5	1	2	3	4

Perform the following steps.

1. Copy the Verilog files *mux_3bit_5to1.v* and *char_7seg.v* from previous parts into the project directory for Part VII.

2. Create a new Verilog file and write the code to instantiate the required subcircuits. Connect the switches SW17−15, in the same order, to the select inputs of each of the five instances of the three-bit wide 5-to-1 multiplexers. Also connect SW14−0 to each instance of the multiplexers as required to produce the patterns of characters shown in Table 3. Each multiplexer will share the same set of inputs (SW14−0), but the inputs will connect to each multiplexer in a different manner. It is up to you to arrange them properly such that you can manually "rotate" your numbers. Following this, you must connect the outputs of the five multipexers to the 7-segment displays HEX4, HEX3, HEX2, HEX1, and HEX0 on the board.

3. Include the required pin assignments for the board switches and 7-segment displays. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper digit codes on the switches SW14−0 and toggle SW17−15 to observe the rotation of the digits.

Part VIII (Bonus Mark)

Extend your design from Part VII so that is uses all eight 7-segment displays on the board. Your circuit should be able to display the last five digits of your student number on the eight displays, and manually rotate the displayed sequence when the switches SW17−15 are toggled. If the displayed sequence is 12345, then your circuit should produce the patterns shown in Table 4.

Table 4. Rotating the sequence 12345 on eight displays.

SW_{17} SW_{16} SW_{15}	HEX7	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0
000				1	2	3	4	5
001			1	2	3	4	5	
010		1	2	3	4	5		
011	1	2	3	4	5			
100	2	3	4	5				1
101	3	4	5				1	2
110	4	5				1	2	3
111	5				1	2	3	4

Perform the following steps:

1. Write the required Verilog code for this part of the exercise and include these Verilog modules in the Quartus II project. Connect the switches SW17−15 to the select inputs of each instance of the multiplexers in your circuit. Also connect SW14−0 to each instance of the multiplexers as required to produce the patterns of characters shown in Table 4. Connect the outputs of your multiplexers to the 7-segment displays HEX7, . . ., HEX0.

2. Include the required pin assignments for the board switches and 7-segment displays. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper character codes on the switches SW14−0 and then toggling *SW*17−15 to observe the rotation of the characters.