EELE 367 - Introduction to Logic Circuits

Lab 8(b) – Design Reuse and Binary Characters on the 7-Segment Display

Objective

The objective of this lab is to gain experience creating and using lower-level subsystems. This will be accomplished by creating a 7-segment decoder component and instantiating it numerous times within the top level entity. This lab will also give experience using signal concatenation, creating a new Quartus project from a prior project, and importing signal assignments from an external file.

Outcomes

After completing this lab you should be able to:

- Create a new Quartus project by copying an existing project.
- Create a 7-segment decoder component and instantiate multiple times.
- Use the DE0-CV User's Guide to find pin assignments for the Cyclone V FPGA.
- Import signal assignments into Quartus from an external CSV file.
- Use signal concatenations to effectively drive lower level sub-systems.

Deliverables

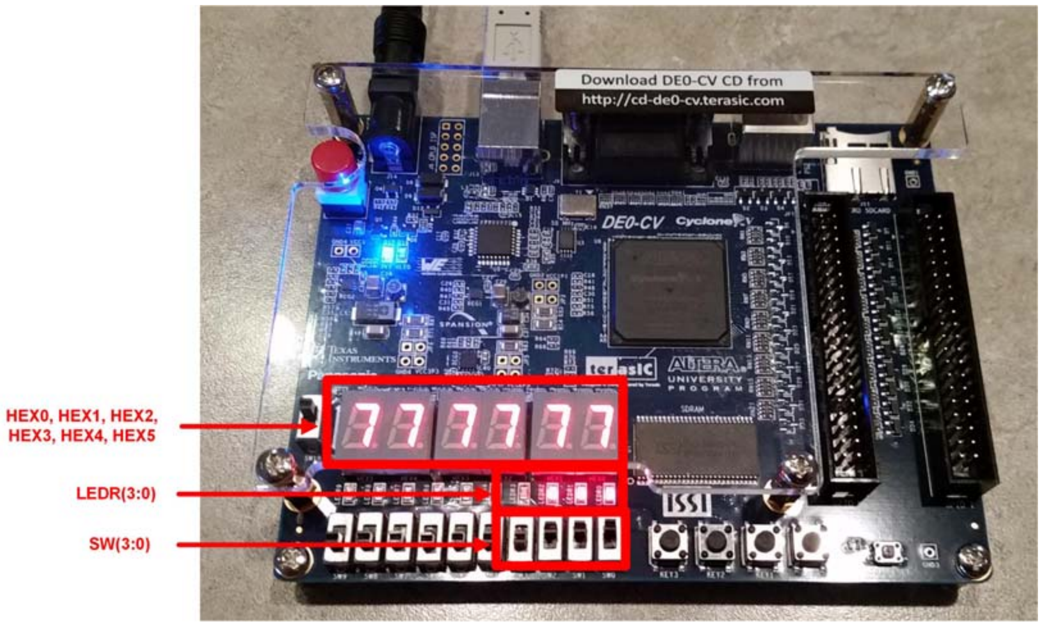
The deliverable(s) for this lab are as follows:

- Part 1 - Demonstrate a design that uses 7-segment decoder components to drive HEX characters to each of the six displays on the DE0-CV FPGA board (50%).
- Part 2 - Demonstrate a design that displays the binary values of the four slider switches on the character displays (40%).
- Upload your top.vhd file to the lab DropBox (10%).

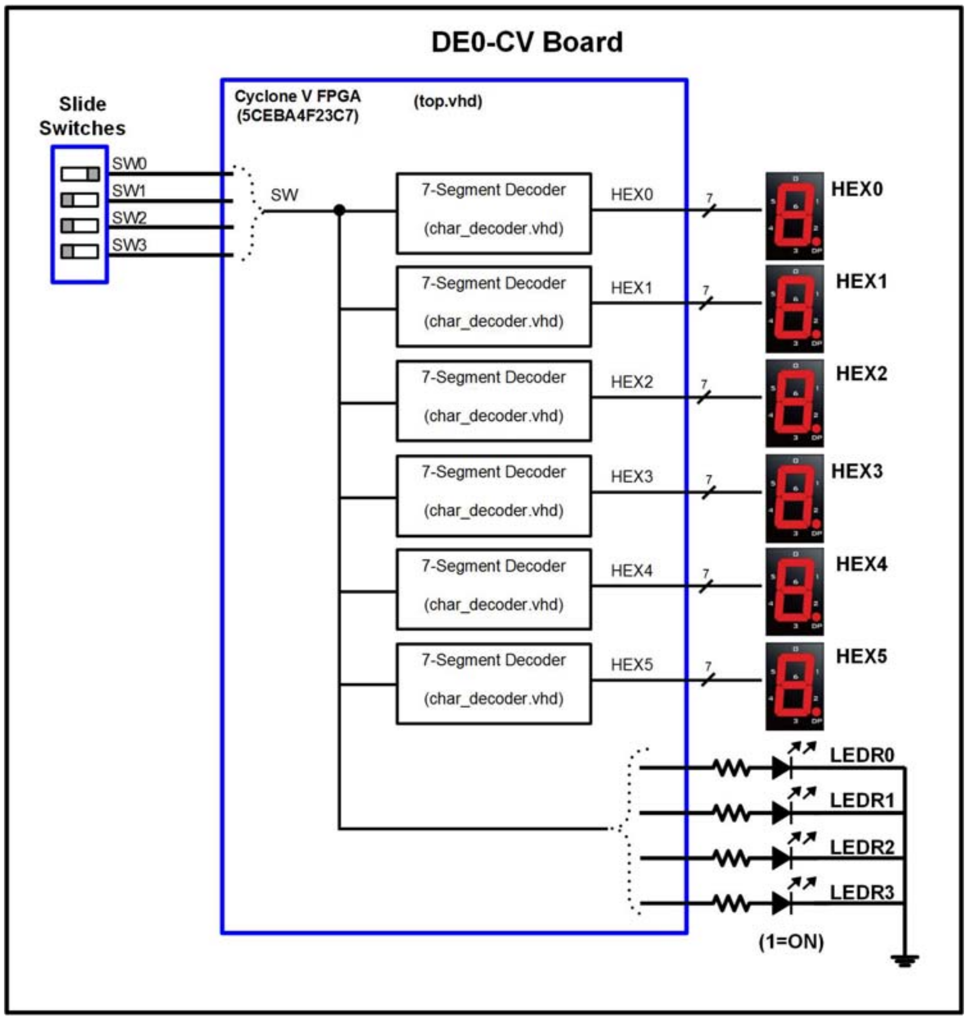
Lab Work & Demonstration

Part 1 – Creating a 7-Segment Decoder Subsystem

In the last lab you created a 7-segment decoder using a process. In this lab, you are going to put your decoder logic into its own subsystem so that it can be instantiated numerous time by the top level design. You will call your subsystem "char_decoder.vhd". You are going to drive all six of the character displays on the DE0-CV board (HEX5, HEX4, HEX3, HEX2, HEX1, and HEX0) with their own decoder. The input to all decoders will be a 4-bit code coming from the slider switches (SW3, SW2, SW1, and SW0). You will also drive the LEDs on the DE0-CV board (LEDR3, LEDR2, LEDR1, and LEDR0) with the values coming from the slider switches. As you change the binary values of the slider switches, the corresponding HEX character will show on all six character displays. The following figure shows the I/O on the DE0-CV that you will be using in this part.


HEX0, HEX1, HEX2, HEX3, HEX4, HEX5
LEDR(3:0)
SW(3:0)
Download DE0-CV CD from http://cd-de0-cv.terasic.com

The following figure shows the block diagram for this part of the lab.



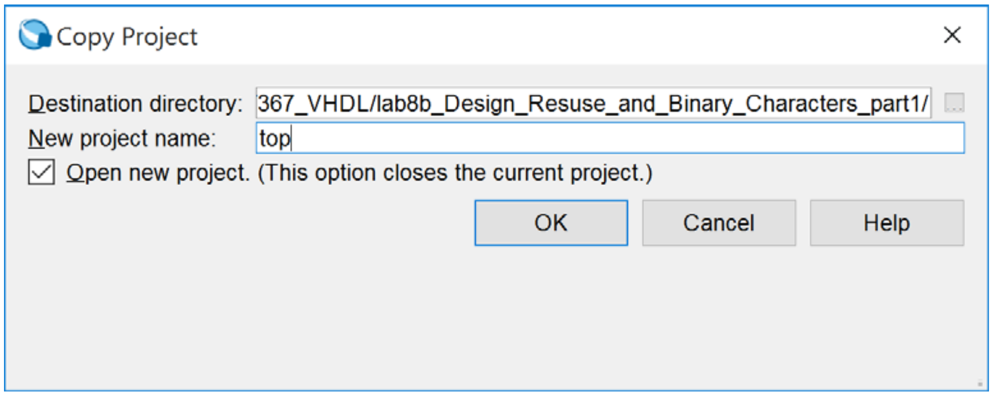
A) Create a Quartus project by copying last week's lab.

Last week you created a project in which you spent a significant amount of time entering project settings, creating the top.vhd file, and assigning pins. Quartus allows you to create a new project by copying a prior project so that you don't lose past effort.

- Launch Quartus

- Open last week's project by performing *File – Open Project* within Quartus. You should browse to last week's project folder and then underneath it select "top.qpf".

- Once the project is open, copy it to a new project by performing *Project – Copy Project*. In the dialog that comes up, you can specify the location and name of the new project. Note that Quartus will make the new project folder so you **don't have to** do this manually. Browse to the folder where you are working on these labs (i.e., ../EELE367) and give the name: "Lab8b_Design_Resuse_and_Binary_Characters_part1". Name the project "top". Make sure the "Open new project" box is checked. Click "OK".


Copy Project

Destination directory: 367_VHDL/lab8b_Design_Resuse_and_Binary_Characters_part1/
New project name: top
☑ Open new project. (This option closes the current project.)

B) Create the 7-segment decoder subsystem (char_decoder.vhd).

You are going to add a new file to your project called "char_decoder.vhd" that will contain the logic to drive the HEX character displays on the DE0-CV board. This logic was created in last week's lab so you just need to copy/paste it into the new VHDL architecture and change the names appropriately.

- In Quartus, create a new VHDL file by performing *File – New*. Select "VHDL File". A blank file will come up in Quartus. Perform a *File – Save As* and name it "char_decoder.vhd".

- Design the decoder. Consider using the following entity definition:

```
entity char_decoder is
   port (BIN_IN   : in  std_logic_vector (3 downto 0);
         HEX_OUT  : out std_logic_vector (6 downto 0));
end entity;
```

Enter the appropriate logic for the 7-segment decoder in the architecture. Make sure to include the STD_LOGIC_1164 package in the new VHDL file.

C) Modify your top.vhd file to drive all six HEX displays.

- In your top.vhd, you now need to change the entity to support driving five additional HEX displays. Add these displays to your top entity. The new output ports should be called HEX5, HEX4, HEX3, HEX2, and HEX1 and have a type of std_logic_vector(6 downto 0).

- You now need to instantiate the char_decoder.vhd subsystem in your architecture. Remember to remove the decoder process from last lab. The first step in using a subsystem is to declare it **before** the *begin* statement. The syntax for declaring this subsystem is:

```
component char_decoder
   port (BIN_IN   : in  std_logic_vector (3 downto 0);
         HEX_OUT  : out std_logic_vector (6 downto 0));
end component;
```

Once declared, you can instantiate the subsystem **after** the *begin* statement. You will need to instantiate it **six** times, once for each of the displays on the DE0-CV board. Remember that each of the subsystems will have an input of *SW*. The syntax for instantiating the subsystem is:

```
C0 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX0);
C1 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX1);
C2 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX2);
C3 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX3);
C4 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX4);
C5 : char_decoder port map (BIN_IN => SW, HEX_OUT => HEX5);
```

D) Compile your design.

- Save your VHDL files and compile. Fix any errors you have and re-compile until successful.

E) Assign the new pins using an external file.

After a successful compile, Quartus will read in the new ports being used in the design. You will need to assign the pins for the five new HEX displays (35 new signals!). As digital systems get larger and larger, it becomes time consuming to use the graphical Pin Planner tool within Quartus. Instead, designers typically use an external CSV file to import the assignments into Quartus. You are going to create one of these files manually and then import it into Quartus.

- Create the pin assignment CSV file. A CSV file is a text file with comma delimited data. For this lab, it is easiest to use a simple text editor to create the file. MS Excel is often used to create CSV files; however, sometimes the CSV file created in Excel contains special characters that prevent it from being imported successfully. As a result, it is suggested that you use a simple text editor. Start a text editor (e.g., notepad, notepad++,…). You are going to type in the signal name, port direction, and pin assignment comma delimited. Start by typing the following into your text file:

```
# This is the pinout assignments for the DE0-CV Board

To,Direction,Location
HEX0[6],Output,PIN_AA22
HEX0[5],Output,PIN_Y21
HEX0[4],Output,PIN_Y22
HEX0[3],Output,PIN_W21
HEX0[2],Output,PIN_W22
HEX0[1],Output,PIN_V21
HEX0[0],Output,PIN_U21
```

Notice that the above text defines the pin assignments for the HEX0 port. Save the file in your project directory and name it “pin_assignments.csv”.

- Look up the pin assignments for the rest of the I/O. You now need to enter the pin assignments for the slider switches (SW), the LEDs (LEDR) and the remaining five character displays (HEX5, HEX4, HEX3, HEX2, HEX1). You can find the pin locations for these I/O in the DE0-CV User’s Manual Guide. This can be found in the Content / Lab Datasheets area of D2L. Once complete, save the file.

- Import the CSV file into Quartus. In Quartus, import the pin assignments by performing *Assignments – Import Assignments*. Then browse to your CSV file and click “OK”. This should have imported in all of the assignments for your design. To verify that everything was OK, run the Pin Planner tool and scroll through the assignments. If you made any syntax errors in your CSV file, the assignments in Pin Planner will be blank. If you entered the wrong pin, you won’t be able to find the mistake until you download your design.

- Re-Compile your design.

- Download your program to the DE0-CV board.

F) Test your design.

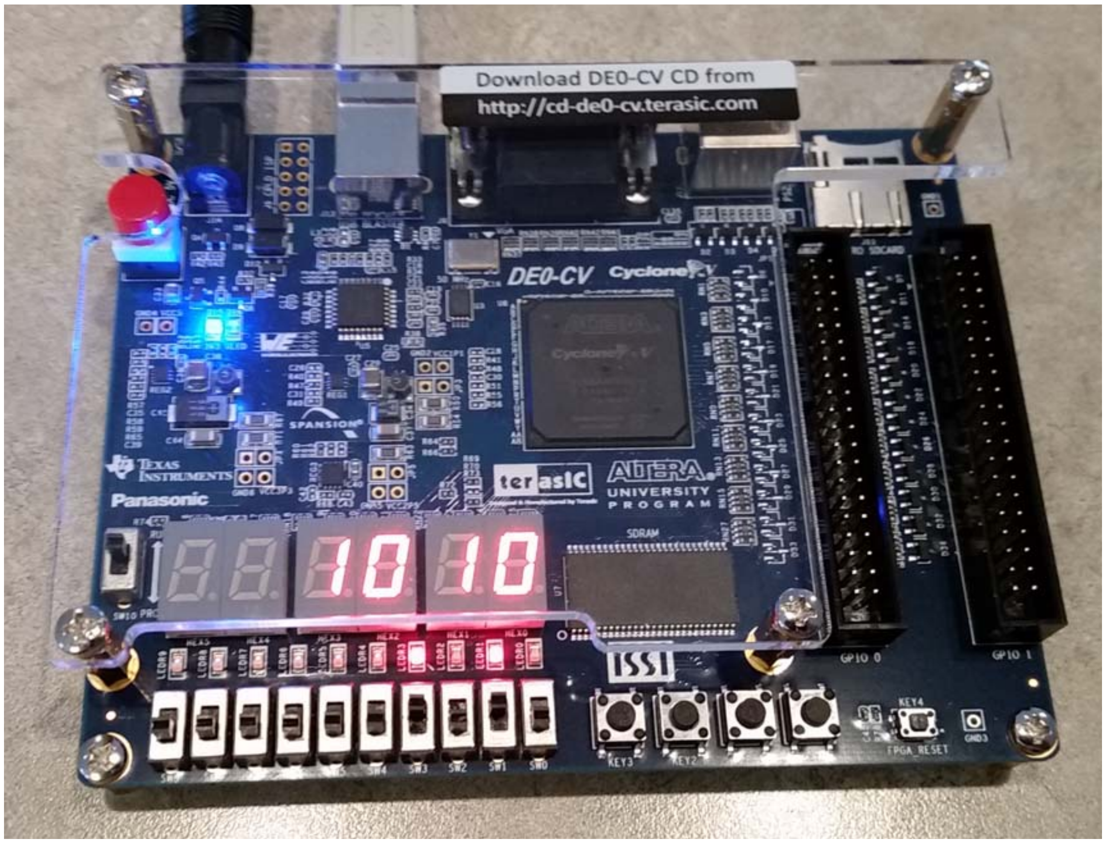
You should now see the values of the slider switches being displayed on the red LEDS and all of the HEX displays showing the same corresponding character. Verify that your design works for each of the 16 possible input codes. Fix any errors you discover.

G) Demonstrate your design.

DEMO – Show the lab instructor the proper operation of your design.

Part 2 – Displaying Binary Characters

Now you are going to design a system that will display the characters on the HEX displays corresponding to the binary values on the slider switches. You will only use four of the character displays for this part (HEX3, HEX2, HEX1, and HEX0). The character displays will only show the symbol 0 or 1 based on the value on the corresponding slider switch. For example, if you set the slider switches to “1010”, the displays should show the symbols 1 0 1 0. The following figure shows how the system will work:



A) Create a new project for this design by copying the project from part 1. Use the Quartus "Copy Project" operation and name the new project "lab8b_Design_Resuse_and_Binary_Characters_part2".

NOTE: The Quartus *copy project* command will only copy project files. The "pin_assignments.csv" file you put in the project folder for part 1 will not be automatically copied over. It is a good practice to manually copy this into your new project folder so that you have the most up to date assignment file.

B) Design the system. First, delete the HEX5 and HEX4 ports from the top entity. Next, delete the component instantiations that that drove the HEX5 and HEX4 ports. Now you need to figure out a way to drive only 0 and 1 characters to the displays based on the corresponding slider switch.

Hint: Consider the case of SW0 driving HEX0. Notice that the input into your char_decoder.vhd is a 4-bit vector. What if you could drive in 0's for bits (3 downto 1) of this 4-bit vector and then connect SW0 to bit 0? This would have the effect that when SW0 was a 0, the 4-bit input vector to the char_decoder would be "000**0**" and HEX0 would display the symbol 0. If SW0 was a 1, the 4-bit input vector to the char_decoder would be "000**1**" and HEX0 would display the symbol 1. All that you need to do to accomplish this design is concatenated three leading zero's each of the slider switches and then use the new vectors to drive the inputs of each char_decoder subsystem.

There are multiple ways to concatenate signals in VHDL. You *could* create a new 4-bit signal and then use a signal assignment and the concatenation operator (&). A faster way to accomplish this when driving component inputs is to put the concatenation directly in the port mapping. VHDL supports this type of concatenation using the following syntax:

```
C0 : char_decoder port map (BIN_IN => "000" & SW(0), HEX_OUT => HEX0);
```

Modify your architecture to implement this behavior for each of the four char_decoder.vhd subsystems.

C) Compile & verify your design.

D) Demonstrate your design.

DEMO – Once checked off, upload your top.vhd file from **part 2 only** to the lab DropBox.

Lab Grading

Demo 1 – Proper Operation of your Decoder Circuits	____________	**/ 50**
Demo 2 – Proper Operation of your Binary Display Circuit	____________	**/ 40**
Review of your top.vhd file	____________	**/ 10** (uploaded to DropBox)
Total	____________	**/ 100**

Post Lab Survey (Just for your own knowledge. This is not graded)

1) Can you create a new Quartus project by copying a prior project?
2) Can you create a subsystem that is instantiated in a higher level design as a component in Quartus?
3) Can you find the pin assignments for the Cyclone V FPGA on the DE0-CV board in its User's Manual?
4) Can you create a pin assignment CSV file and import it into Quartus instead of using the Pin Planner tool?
5) Can you use signal concatenations within a component port map to form new input vectors?