Lab 3A – Connecting to the DE10-Lite Hardware

Introduction

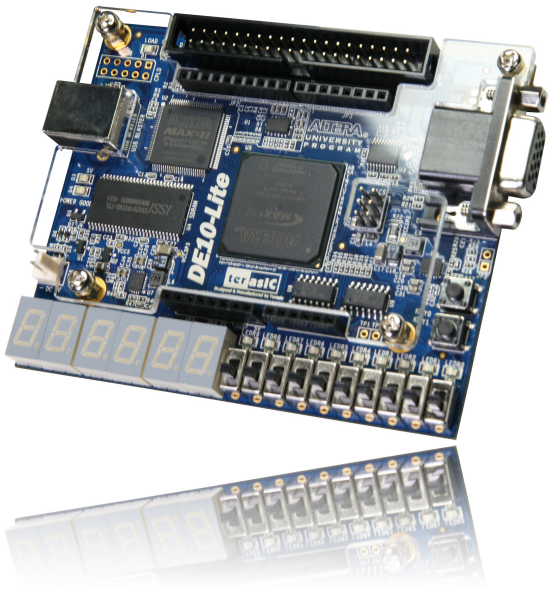
In a previous lab, you designed some simple logical gates and tested the design in hardware. The hardware connections were done automatically for you in the templates provided. In this lab, you will learn how these connections are done through the naming of signals in a top-level wrapper.

The mapping of the SystemVerilog top-level signals to the actual hardware device can be a pretty tedious operation. The mapping is defined in a file called a *Quartus Settings File* (with a *.qsf* extension). There exists extensive documentation on setting up such a file for a particular design.

Fortunately, the board manufacturer (Terasic), along with the tool provider (Altera), provide a support ***qsf*** file from which we can edit this connection information for our particular design. In this lab, you will be editing this file so that you can correctly connect your design to the hardware in Lab 3, where you will be designing a seven-segment decoder.

The main design module for Lab 3 is named: ***lab3_decoder***. It will have seven input signals (eventually connected to seven input switches) and eleven output signals (eventually connected to the seven-segment display).

In addition to modifying the Quartus Settings File, you will also be enabling the proper i/o signals within the **DE10_LITE_Temple_Top.sv** file.



Within the lab directory, there is a file called **lab3_top.qsf**. This file contains all of the usable input and output connections on the FPGA defined with respect to their connection to the peripheral hardware. The signals are grouped by functionality. Consider the following sections of the file.

```
71   # ------------------------------LED section---------------------------------- #
72   # --------------------------------------------------------------------------- #
73   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[0]
74   #set_location_assignment PIN_A8 -to ledr[0]
75   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[1]
76   #set_location_assignment PIN_A9 -to ledr[1]
77   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[2]
78   #set_location_assignment PIN_A10 -to ledr[2]
79   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[3]
80   #set_location_assignment PIN_B10 -to ledr[3]
81   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[4]
82   #set_location_assignment PIN_D13 -to ledr[4]
83   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[5]
84   #set_location_assignment PIN_C13 -to ledr[5]
85   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[6]
86   #set_location_assignment PIN_E14 -to ledr[6]
87   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[7]
88   #set_location_assignment PIN_D14 -to ledr[7]
89   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[8]
90   #set_location_assignment PIN_A11 -to ledr[8]
91   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ledr[9]
92   #set_location_assignment PIN_B11 -to ledr[9]
```

```
93   # ----------------------------switch section-------------------------------- #
94   # --------------------------------------------------------------------------- #
95   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[0]
96   #set_location_assignment PIN_C10 -to sw[0]
97   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[1]
98   #set_location_assignment PIN_C11 -to sw[1]
99   #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[2]
100  #set_location_assignment PIN_D12 -to sw[2]
101  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[3]
102  #set_location_assignment PIN_C12 -to sw[3]
103  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[4]
104  #set_location_assignment PIN_A12 -to sw[4]
105  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[5]
106  #set_location_assignment PIN_B12 -to sw[5]
107  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[6]
108  #set_location_assignment PIN_A13 -to sw[6]
109  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[7]
110  #set_location_assignment PIN_A14 -to sw[7]
111  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[8]
112  #set_location_assignment PIN_B14 -to sw[8]
113  #set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to sw[9]
114  #set_location_assignment PIN_F15 -to sw[9]
```

You can see above that hardware ***pin A8*** is connected to ***led 0***, ***pin C10*** to ***switch 0***, etc. These pin-signal mappings are defined in the **lab3_top.qsf** file above.

On the DE10-Lite board, there are 10 switches, numbered 0-9, using signal name ***sw***, which is a multi-bit or bussed signal. So ***sw[9]*** refers to switch number 9, etc.

Note the relationship between the FPGA pin numbers and the SystemVerilog signal names. Terasic and Altera have taken care of the particulars of the format of this file for us.

It is beyond the scope of this lab and course to delve into the details of this format. However, it is important to see the connections and the top-level signal names in case you need to change them in future designs. Most often when you purchase a piece of hardware like this, the vendor will supply the appropriate support files for the board. So typically you would only have to do some minor edits of their master file.

A final important note – the format for this file allows for comments. If the first non-space character in the line is a '**#**' symbol, the line is treated as a comment – similar to the format of shell scripts and Python programming. Looking at their master file, all of these useful lines of code are commented out! That is their default – no pin is connected. So we need to uncomment the ones being used (by removing the '**#**' symbol for the signals of interest in your design).

After the pins are connected to the proper signals, these signals must be enabled in the wrapper file (**DE10_LITE_Temple_Top.sv**). This is accomplished by defining macros at the top of the file.

Design Specification and Steps

The goal of this lab is to step through a process to modify a *Quartus Settings File* that will be used for Labs 3A and 3. This exercise will teach you how the top-level signals in your designs are connected to the hardware switches, LEDs, push buttons, etc. In future labs, the ***qsf*** files will be provided to you.

Part A – modifying the qsf file

Open the **lab3_top.qsf** file in VS Code.

Proceed to uncomment (i.e., remove the '#' symbol) the lines for the following signals:

- All LEDs: ***ledr[0]*** through ***ledr[9]*** (lines 73-92)
- Switches 0 through 6: ***sw[0]*** through ***sw[6]*** (lines 95-108)
- Seven-segment displays 0 through 3: ***hex0*** through ***hex3*** (lines 117-180)

Save the file. You should now have a *Quartus Settings File* for use in Labs 3A and 3.

Part B – editing the top module

Open the top module: **DE10_LITE_Temple_Top.sv**.

Note how all of the i/o data sets are grouped. Each grouping is bracketed by an ***`ifdef/`endif*** directive. You can read more about these SystemVerilog constructs here. Essentially, this directive tells the compiler to include this portion of the code ***if*** a specific FLAG macro is ***defined***. For instance, consider the following piece of code.

```
73      ////////////// LED: 3.3-V LVTTL //////////
74      `ifdef ENABLE_LED
75          output          [9:0]      ledr,
76      `endif
```

The LED bus ***ledr[9:0]*** will only be defined if the ENABLE_LED macro is included. Otherwise, the compiler will ignore this segment of the code, which may cause issues if the design makes use of the LEDs on the DE10-Lite hardware.

In this lab, we need the following signals to be defined:

- All LEDs: define ENABLE_LED
- Switches 0 through 6: define ENABLE_SW
- Seven-segment displays 0 through 3: define ENABLE_HEX0, ENABLE_HEX1, ENABLE_HEX2, ENABLE_HEX3

These macros can be defined at the top of the wrapper file. Simply use the `define directive as such:

```
`define ENABLE_LED
```

Make sure to only place one macro per line. You can begin defining your macros on line 29 as instructed by the comments in the file.

Now, create a simple design inside the top-level module. Connect the seven input switches to the first seven LEDs. This can be done with a single assign statement.

Simulation/Verification

Because the design is so simple – connecting six inputs to six outputs, there is no simulation necessary.

Synthesis

Make sure all of your files are saved. Run the synthesis (i.e., right-click and run: **lab3_top.qsf**).

Download your binary file (**lab3_top.sof**) and test your design.

Preparation for Lab 3

Open **DE10_LITE_Temeple_Top.sv** and comment out your Lab 3A connections. Inside this top module, instantiate the module: ***lab3_decoder***. Previously, when this lab used the Basys3 board, each display had its own anode that had to be toggled individually. Additionally, segments between displays shared common cathodes. Thus, the same pattern would show on all four displays at once. According to the DE10-Lite User Manual, each segment of each display can be toggled separately, thus eliminating the need for an "anode" decoder. This is why we had to enable all eight signals (seven segments plus a decimal point) for each display (e.g., ***hex0[0]*** - ***hex0[7]***, ***hex1[0]*** - ***hex1[7]***, ***hex2[0]*** - ***hex2[7]***, ***hex3[0]*** - ***hex3[7]***). However, to keep the code consistent, we will still create an anode decoder module so that we can select which display we want turned on. To facilitate compatibility with the DE10-Lite board, the ***muxed_display*** module is already instantiated for you.

In order to complete the instantiation, include the ***lab3_decoder*** module. Within the instantiation, connect the switches as the inputs. Two internal signals have already been declared: ***an[3:0]*** and ***cathode[6:0]***. These will be the outputs for the lab3_decoder module. Simply insert your instantiation where the comment reads: "instantiate top level design here".

```
lab3_decoder u_top (.sw(sw), .an(an), .cathode(cathode));
```

You will not see any errors you make in this instantiation until you synthesize the design in Lab 3. However, you can check for the correct syntax of your edits by running **lab3_top.qsf**.