ECEN3763– Homework Week 8
Spring, 2022
Due Monday March 7th, end of day – 15 Points

This assignment provides an introduction to Platform Designer and System Console.

After completing this assignment, you will submit a .qar file for your completed project.

Part 1: Create a Platform Designer design

1) Create a new Quartus Project for your DE10-Standard board. You will need to include clocks, switches, and LEDs.

2) File > New > Qsys System File to create your Platform Designer (PD) block.

3) In the Platform Designer (PD) IP Catalog, search for JTAG, and add a JTAG to Avalon Master block. Accept the default configuration.

4) In the IP Catalog, search for PIO and add it to your PD block. In the PIO configuration window, set the PIO as 8 bits wide output. Leave the Reset Value as it was.

5) Add a second PIO block, a 1 one bit output.

6) In the System Contents tab, do the following:
Note: To make connections in the PD System Contents tab, you click on the white circles, and the connections will appear as darker black lines. Also notice that as you make the connections below, the errors that appear in the bottom Messages window are removed.

a) Connect the master output of the JTAG bridge to the slave inputs (s1) of the PIO blocks.

b) Connect the clk output from the clk_0 component to the clock inputs of all other blocks.

c) Connect the master_reset output of the JTAG bridge to the reset inputs of PIO blocks.

d) Export the external connections of both PIO blocks. Use whatever names you want. To do this, double click in the Export column, and enter a name. These external connections, which are termed Conduits in PD, add these ports as module ports for the PD module being created.

5) At this point, notice the errors listed in the Messages tab.
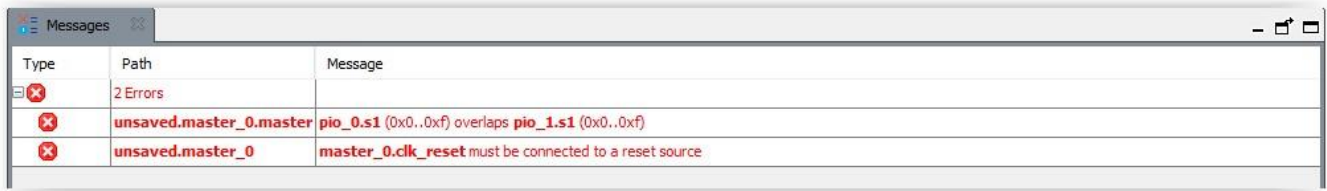
The first message indicates that the two PIO blocks have overlapping address ranges. Since both PIOs are slave or target components, they must have a unique memory address so that the JTAG to Avalon Master Bridge can communicate with each block. You can correct this by going to System > Assign Base Addresses.

Write down the address ranges of the PIOs here:

PIO_0:   `0x0000_0010 >>> 0x0000_001f`
_____

PIO_1:   `0x0000_0000 >>> 0x0000_000f`
_____

The remaining error message says you need to connect the master_0 master_reset output. Connect the clk_0 clk_reset output to the master_0 clk_reset input.

6) No errors should remain. Click Generate HDL (bottom right corner of PD GUI). When the Generation window opens, you can rename the directory when the PD files will be placed with a meaningful name (the name is shown as unnamed in the Output Directory path). When the Save window opens, rename the PD component something meaningful. Once the PD project builds you can close PD.

7) Per the prompt, you will need to manually add the .qip file to your Quartus project. The PD files are located in the directory you renamed in part 6 above. From that directory, go into the synthesis directory, and you will find the .qip file. Do not also add the .qsys file, this file is not needed. You do not need to add the .sip file (if created) as that is specific to simulation, and we are not going to simulate the design at this time.

What did you just create?

 Platform Designer construct the system you created using the graphical patch panel interface. You can see how complex the system is by looking in the synthesis / submodules directory, and see how many files were created. Fortunately, all you will need to do is instantiate the top level module and allow Quartus to connect and synthesis the entire system.

Part 2: Create your overall design

You will connect the 8 output PIO to 8 of the LEDs on the board, and the 1 output PIO to a LED on the board.

1) In your top level source file, instantiate the PD component. To see how to instantiate the component, click on the arrow to the left of your component name in the Quartus Project Navigator pane, and open the .v file. Your instantiation should only have 3 ports, the clock and the two PIOs.

2) Connect the 8 bit PIO to LEDR[7:0], connect the 1 bit PIO to LEDR[9]. Connect LEDR[8] to logic 0.

3) Compile your project and download to your board.

Part 3: Test your design

To communicate with the JTAG master in order to write values to the LEDs, you will use the System Console tool in Quartus.

1) Open System Console with Tools > System Debugging Tools > System Console.

In the Messages pane, notice that the JTAG connection is established with the FPGA and that the name of the programming file is reported. System Console recognizes that the design contains a JTAG Master bridge component.

You will use the TCL console to issue commands.

2) Issue the following commands in order to communicate with the JTAG master component. Be aware the cutting and pasting commands from a PDF document often results in bizarre behavior due to hidden characters. You should type these commands in at the System Console prompt.

```
set master_path [lindex [get_service_paths master] 0]
```

This command identifies the communication path that is required for communication.

```
set mpath [claim_service master $master_path ""]
```

Create a path variable by claiming the communication channel.

```
set jpath [lindex [get_service_paths jtag_debug] 0]
```

Create a second path variable to the JTAG control circuitry in the master device.

```
jtag_debug_reset_system $jpath
```

The master reset output of the JTAG master is connected to the PIOs. Executing this command sends a system reset. Later in this lab, when you have written values to the LEDs, issuing this command should reset the outputs to 0, turning off all the LEDs.

```
set clk [jtag_debug_sense_clock $jpath]
```

This command is used to sense that a clock is running in the FPGA, the clock connected to the PD design. A value of 1 indicates the clock is toggling.

```
set rst [jtag_debug_sample_reset $jpath]
```

This command is used to test the value of the reset signal driven by the JTAG master.

3) To write values to the LEDs, you use a master write command. The format of the command is `master_write_xx $mpath <address> <value>`. xx can be 8, 16, or 32 depending on the width of the components. For this assignment we use 8.

Review the address of the PIO components. For the examples below, PIO_1 is located at address 0, and PIO_0 is located at address 0x10. If your addresses are different, modify the commands below.

4) Verify your design is working properly:

```
master_write_8 $mpath 0x0 0x1
```

LEDR[9] should light. Reissue the command with a value of 0 to turn off the LED.

Now test the other PIO.

```
master_write_8 $mpath 0x10 0x55
```

An alternating 0 / 1 pattern should appear on LEDRs[7:0]

Experiment with different values.

Issue a JTAG reset command to confirm all LEDs turn off.

End this test with `close_service master $mpath`, and close System Console.

Part 4: Add read capability (modify your PD component)

Reopen your PD design, and add another PIO component. Make the component input only, 10 bits wide, and connect the component to your PD system using the patch panel. In your Verilog, connect the new PIO signals to the SW on your board.

The command to read from the 10 bit PIO is

```
master_read_16 $mpath <addr> 1
```

1) Add the PIO component, rebuilt your design, and verify correct operation of all 3 PIOs.

You might be asking yourself what is different about using System Console compared to a simple Tcl console. The biggest difference is that, instead of sending commands directly to a specific component in your design, you are sending commands to a master or initiator component. These commands are interpreted by the component, which then initiates one or more Avalon interface transactions to communicate with slave or target devices.

We use the JTAG to Avalon Master Bridge component, since we don't have another master in our system, such as a NIOS or ARM processor.

If you are working on a complex system with other designers, you may have the hardware portion of your design completed before the firmware engineers have any code running on a microprocessor. Instead of waiting to test your design, you can add a JTAG to Avalon Master Bridge component, create various tests to confirm hardware operation, and remove the bridge when the firmware people are ready. And, you can be confident that problems are due to the firmware and not your hardware…………..