ECEN3763 – Homework Week 3

Fall, 2021

Due Monday January 31st, 11:59pm - 15 Points

The purpose of the homework assignment is to serve as an introduction to using the Tcl (Tool Command Language) language with the Quartus environment. Tcl allows you to do perform many powerful tasks when designing and debugging FPGA designs.

For this homework assignment, you will follow through the tutorial material and answer questions. Please take enough time to understand the reasons behind the steps of the tutorial, as the information you learn here will be used throughout the semester

You can use Chapter 8 of the Quartus User Guide – Debug for this assignment. It is not necessary to read the entire chapter, you will be instructed to read certain sections.

Submission:

Create a text file and enter the answers to the 8 questions below. Each question is worth 1 point. Your Tcl script is worth 7 points. Submit a .zip file to Canvas containing both files.

Part 1: Program Board

1. Extract the Homework_Week3.zip file provided. This file contains ISSP_Demo.sof, the file used to program your DE10-Standard board, and also includes the Debug User Guide mentioned previously.

2. Using the Quartus programmer, program your board with the provided file.

   Question 1: Briefly describe the board behavior after programming.

3. Press the 4 pushbuttons on the board in any order.

   Question 2: What do the pushbuttons do?

Part 2: Testing the Design

The code for the top level module is shown below. A phase lock loop or PLL (U0) is used to generate a 1 Mhz clock, that is fed to an up counter (U3) and a down counter (U4). The PLL outputs a locked signal that is connected to LEDR[9]. The only control signal to the counters is an active low reset signal.

An In-System Signals and Probes (ISSP) block named reset (U1) is used to drive the one bit wide signal reset_n to the counters. The ISSP block named direction (U2) is used to select either the up counter or down counter for display on the LEDs. The ISSP block named count_value (U5) is used to read the LED values. Chapter 8 of the Debug manual explains the details of the ISSP cores, review this material as needed.

Note that this design could have been done with one ISSP block.

```
module ISSP_Demo(
      input        CLOCK_50,
      output [9:0] LEDR);

logic clock, reset_n, up_down_n;
logic [21:0] countup, countdown;

my_pll U0 (.refclk(CLOCK_50), .rst(1'b0), .outclk_0(clock), .locked(locked));
reset U1 (.source(reset_n));
direction U2 (.source(up_down_n));
count_up U3 (.clock(clock), .reset_n(reset_n), .counter(countup));
count_down U4 (.*, .counter(countdown));
count_value U5 (.probe(LEDR[7:0]));

assign LEDR[7:0] = up_down_n ? countup[21:14] : countdown[21:14];
assign LEDR[9] = locked;
assign LEDR[8] = 0;

endmodule
```

Question 3: What signal is used to control the direction of counting on the LEDRs?

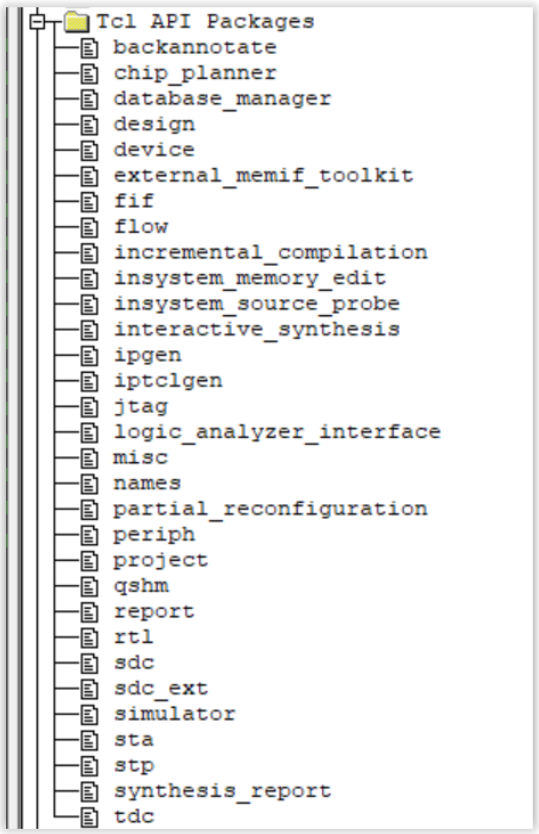
Quartus contains an In-System Signals and Probes Editor GUI that allows you to interact with the ISSP cores using mouse and keyboard. However, we are going to interact with the ISSP

cores using Tcl.

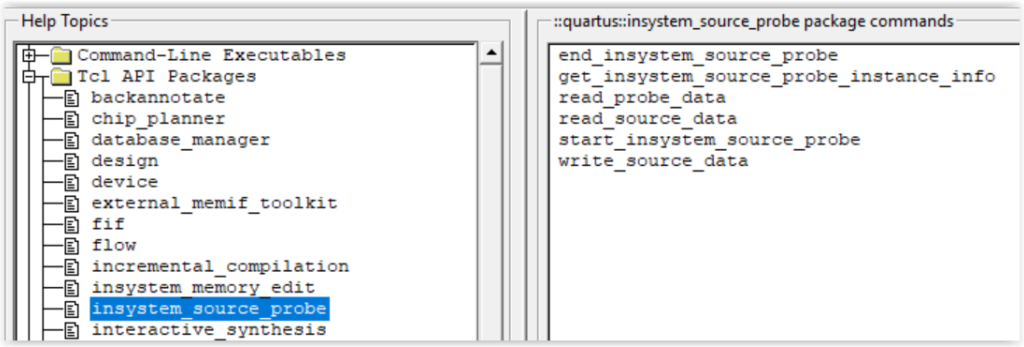
Part 3: Tcl Consoles

Tcl is an interpreted language, and the base language can be extended through the use of custom Tcl packages. Quartus provides a set of Tcl packages and command specific to certain tasks. The complete list of Tcl packages provided by Quartus is shown below.

> Note: Quartus contains a very useful help tool that many Quartus users are not aware of, called Qhelp. Qhelp is very useful, and is contains information spread across many different Quartus manuals. To launch Qhelp, open a command window and type the following: quartus_sh -Qhelp



Each Tcl package has a unique set of commands. Highlighting insystem_source_probe in Qhelp displays the 6 available commands. Highlighting each command on the right will provide help information in the Qhelp GUI.



Ideally, Quartus would provide a single Tcl console that loads all available Tcl packages, so that any Tcl command could be executed. Unfortunately, this is not the situation. Quartus provide many different consoles, that have limited use. One of the very frustrating things about using Quartus can be knowing which Tcl console to use.

For most of this class, we will use the **quartus_stp** Tcl console. If you look in your Quartus installation area, in the bin64 directory, you will find a quartus_stp.exe file. If you just double click on this file, or type in the name in a command window, the window will open and quickly close. In order to keep the window open, you must include a -s option.

Once you have a Tcl console open, type help, and you will see a listing of the Tcl packages that are already loaded, and a list of Tcl packages that you can load manually using the `load_package` command. For the **quartus_stp** console:

```
tcl> help
------------------------------------------------------------------------------


---------------------------------
Available Quartus Prime Tcl Packages:
---------------------------------

Loaded                                Not Loaded
-----------------------------------   ---------------
::quartus::insystem_memory_edit       ::quartus::flow
::quartus::insystem_source_probe
::quartus::jtag
::quartus::logic_analyzer_interface
::quartus::misc
::quartus::project
::quartus::report
::quartus::stp
```

You see that the insystem_source_probe package is already loaded.

Part 4: Using Tcl to Control the Design

1. Open the **quartus_stp** Tcl console. At the Tcl prompt, type help and confirm you see the list of available Tcl packages.

2. The first step is to interrogate the system for the name of the USB-Blaster cable. the command `get_hardware_names`. This command is part of the jtag package, and returns a list of all USB-Blaster cables connected to your computer. In order to parse the list, you use the lindex (list index) Tcl command. Since you should only have one cable connected to your computer, the list will contain 1 item at index 0 (remember, Tcl is 0 based).

   Type in this command:

   ```
   set cable [lindex [get_hardware_names] 0]
   ```

   Question 4: What value is returned for cable?

3. Now interrogate the system for the specific FPGA containing the ISSP cores using the command `get_device_names`, which is also part of the jtag Tcl package. This command returns a list of devices. Since the FPGA fabric is the second device, you will want the value of index 1.

   Type in this command:

   ```
   set device_name [lindex [get_device_names -hardware_name $cable] 1]
   ```

   Question 5: What value is returned for device_names?

4. If you had created the design for this lab, you would know the details of the ISSP cores. The instance names of the cores will be in the order that the cores were placed into the design. The reset core was the first core instantiated, so will be instance 0, followed by the direction core as instance 1, and the count_value core as instance 2. You can verify this by using the get_insystem_source_probe_instance_info command.

   ```
   get_insystem_source_probe_instance_info -device_name $device_name -hardware_name $cable
   ```

The returned information is in the form {instance number # of sources # of probes}

`{0 1 0 {}} {1 1 0 {}} {2 0 8 {}}` is what this command should return.

5. Issue the start_insystem_source_probe command.

```
start_insystem_source_probe -device_name $device_name -hardware_name $cable
```

Any time you want to interact with ISSP cores in your design, you will need to include these three Tcl commands (the instance_info command is optional). And remember, the board must be programmed with you design containing the ISSP cores, and your board must be powered and connected to the USB-Blaster USB cable.

6. Reset the board by issuing a `write_source_data` command to the reset instance, by setting the value of the source to 0.

```
write_source_data -instance_index 0 -value 0
```

Question 6: What command will cause the counter to run?

7. Reverse the count direction.

Question 7: What command will cause the counter to reverse direction?

8. To read the LED values, you must use a read_probe_data command. By using the Qhelp utility, issue a command that will display the value of the LEDs in hex.

Question 8: What command will display the LED values in hex?

9. When finished, end the insystem_source_probe session.

```
end_insystem_source_probe
```

Part 5: Creating a Tcl script

While it is interesting to type in Tcl commands one at a time, writing a script that can control your hardware is more interesting. By creating a simple text file with a .tcl extension, you can execute a series of commands in order.

> Useful things: To add a delay to a Tcl script, using the Tcl command `after`. Adding the command `after 5000` will create a 5 second delay (value is milliseconds). You can add a comment using a puts command with the text in double quotes.
>
> `puts "Counter in reset for 5 seconds."`

1, Take the commands you used in Part 4 and create a Tcl script that will do the following:

a) Hold the counter in reset for 5 seconds
b) Let the counter increment for 5 seconds.
c) Change counter direction and let run for 5 seconds.
d) Reset the counter.

2. Execute and verify your script.

```
source my_script.tcl
```

The script should execute without error when run multiple times.