Recitation 7

I/O Interfacing: PS2

Introduction

Over the coming two recitations, students will be introduced to interfacing with different input/output (I/O) devices, starting with PS2 interfaces and controllers during this recitation. Understanding the interaction between the interfaces and controllers is key to understanding how to utilize different functional modules. Students will study the provided PS2 controller, then alter the code so that pressing a key on a PS2 keyboard causes the FPGA board to display the corresponding character on the built-in LCD screen.

Collaboration Policy

You will be working in groups of 2-3. Groups are allowed to collaborate.

Equipment

- Computer with Quartus Prime software
- DE2 FPGA board
- PS2 Keyboard
- DE2 board manual found under Sakai → Resources
- FPGA Blast Tutorial found under Sakai → Resources

Tasks

To receive credit for this recitation, you must complete:

- ❏ Task 1: Integrate the PS2 input and display results on the FPGA LCD

You must complete all parts of this recitation to receive credit. A TA must sign-off on the completion of each task. Ensure that the TA marks the completion of the tasks in Sakai.

Grading

- Completing Recitation Tasks: 1 point (pass/fail)

Practicing with connecting Verilog with FPGA LCD

In this section of the recitation, you will be using the PS2_skeleton file provided to learn about how to integrate a PS2 input, and how to use the LCD built into the FPGA Additionally, you will alter the skeleton so that it can take in an input (use a character from the range a-z) and display it on the LCD on the FPGA.

To begin this recitation, download the PS2_skeleton archive file from the class Sakai site -> Resource. This will be similar to the skeleton you used to implement your processor, but has been altered slightly for the purposes of this recitation.

Open the “PS2_Interface.v” and “PS2_Controller.v” modules. In this first step, you will look at how these modules interact with one another. Focus on the PS2_Controller.v file first; in this file, there are multiple sections, and it’s important to understand what each section does:

Port Declarations	This section contains the port declarations and definitions used for the module.
Constant Declarations	This section contains a list of local parameters whose scope is that of the module they are contained in. These ‘localparam’ values are protected from being altered by any module that calls an instance of the PS2_Controller.v. Leave this section as is determined.
Internal Wires and Registers Declarations	This section has the internal wire and register declarations used throughout the module. Leave this section alone.
Finite State Machine(s)	This section contains a simple FSM which controls the current state of the PS2_Controller. In the first always block, the reset condition is defined so that if a reset signal is given, the controller moves to the default ‘PS2_STATE_0_IDLE’ state. Else, it allows the current state of the ps2_transceiver to be assigned the correct next ps2_transceiver state. In the next always block, the conditions for which state the controller will move into are defined. Using a case statement, the logic is defined for what the next state should be based on the inputs being seen. The states are: PS2_STATE_0_IDLE, PS2_STATE_1_DATA_IN, PS2_STATE_2_COMMAND_OUT, PS2_STATE_3_END_TRANSFER, PS2_STATE_4_END_DELAYED. S_ps2_transceiver is in parenthesis at the beginning of the case

	statement because the case statement is sensitive to that signal; this means whenever the value of this changes, the case statement is entered. The correct current state that the PS2_transceiver should be in is updated on every clock cycle.
Sequential Logic	This section contains the sequential logic of the controller. The always statement uses the clock signal to control when the values of certain registers are updated. Don't worry about this section too much.
Combinational Logic	This section contains the logic used to determine what the positive and negative edges of the clock signal are. In addition, some basic assignments are made for asserting the correct values on the nodule ports.
Internal Modules	These are internal instances of Altera's IP blocks that handle communication with the internal PS2 hardware of the FPGA.

For this activity, it is enough to understand how the PS2 works down to the level of the controller; understanding how each of the internal modules of the PS2_Controller.v file work is unnecessary, but if you are curious you should check it out.

The PS2_Interface.v file simply calls an instance of the controller; your skeleton will pass data to the controller through the interface, and the controller will control when its internal modules are communicating with the PS2 core hardware in the FPGA. Then, the interface will output data to be fed into the LCD module.

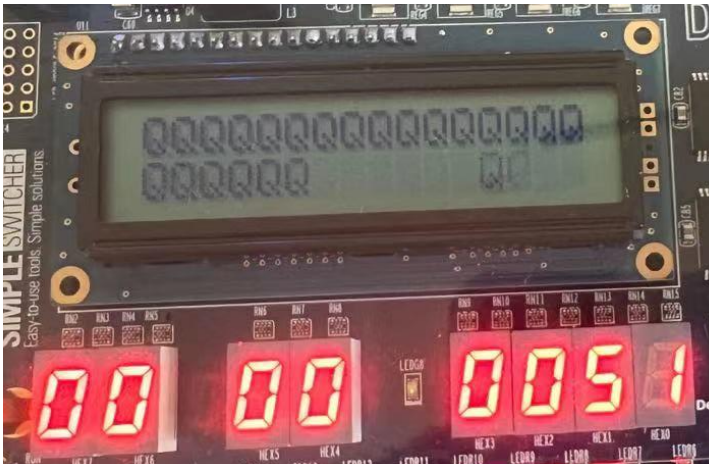
Following three modules in skeleton.v will help you to see the output when you press a key. **Connect a PS2 keyboard to the FPGA and observe the output on the LCD screen when a key is pressed.**

```
// keyboard controller
PS2_Interface myps2(clock, resetn, ps2_clock, ps2_data, ps2_key_data, ps2_key_pressed, ps2_out);

// lcd controller
lcd mylcd(clock, ~resetn, 1'b1, ps2_out, lcd_data, lcd_rw, lcd_en, lcd_rs, lcd_on, lcd_blon);

// example for sending ps2 data to the first two seven segment displays
Hexadecimal_To_Seven_Segment hex1(ps2_out[3:0], seg1);
Hexadecimal_To_Seven_Segment hex2(ps2_out[7:4], seg2);
```

Go back to PS2_Interface.v. Find the correct relationship between "ps2_key_data" and "last_data_received". For this part, add some code so that a given key press will have the correct character displayed on the LCD screen. **Choose 4 characters to implement.** Following picture is the expected output when you press "Q".



A hint for using the LCD screen: if you want to show a letter on the screen, the value you pass to the LCD should be the letter's ASCII code. For example, if you want to show "Q" on the LCD screen, its ASCII value is 81(decimal), so the input of the LCD module should be 51.

Refer to the tutorial on Sakai for how to blast your project on the FPGA.

Show your work to a TA when you are done.

Note: We understand that the LCD module and PS2 could be a little more challenging than usual and expect this recitation to take a bit more time. Start by focusing on getting one character to be correct, then expand to 4. A good starting point is to monitor the data being passed to the LCD module from the PS2 interface; the data passed is the scancode of the key pressed on the keyboard.