ECE 368 Digital Design

Spring 2014

Lab 2 – PS2 Keyboard Interface

Dates Performed: Wednesday, February 12th , 2014

Submission Date: Wednesday, February 19th , 2014

Team #2 :

Massarrah Tannous _________________________

Daniel Noyes _______________________________

Table of Contents

Problem Statement

For this lab, it is mainly based of the previous lab, the ALU, in an enhanced manner by using a key board to input the ALU values. We were to create PS/2 key board interface to the ALU. The key interface consists of two major parts: the PS2 controller and key-code to ASCII. The PS/2 controller forms the key code from the commands sent by the PS/2 device. Whereas key code to ASCII translates the key code in to an ASCII character. As side note, the key code to ascii considers whether a shift was pressed.

In order to implement this assignment, the Xilinx ISE software tools and the Digilent Spartan-3E Starter Board are to be used along with a PS/2 keyboard. The Xilinx software allows designing the VHDL code, simulating through the test bench and programming the Spartan 3E board.

As part of the extra credit for this lab, designing an additional element that interpret a keyboard input and translate into opcodes and operands for the UMD_ALU_FPU from the previous lab. This allowed using the concept of FSM machine and lookup tables apart from the PS/2 controller and the key code to ASCII. The further extra credit for this lab is to use the LCD to output the keystrokes as inputted into the ALU.

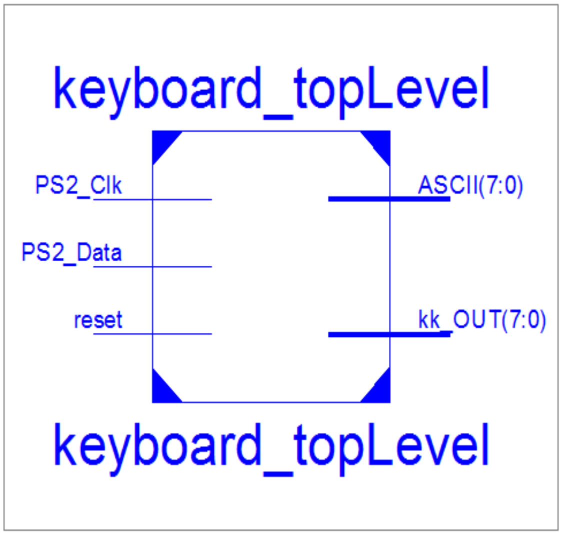
RTL Block Diagram (Hand Drawn Design)

The original designs for both the counter and the ALU are the attached engineer papers which are labeled as “Original Designs”.

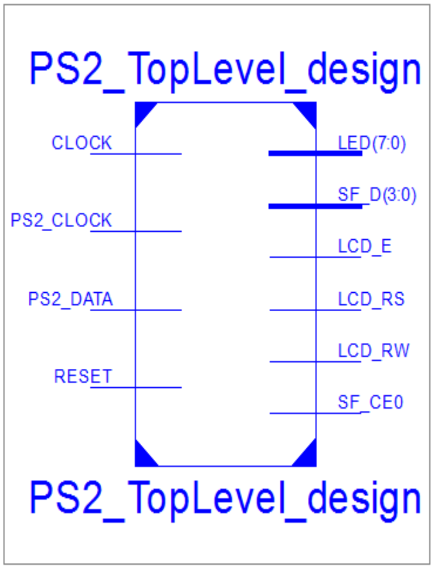
VHDL Entities Specification

The specifications of each entity used for the counter and the ALU are shown in Appendix A.

System design top level entity



Drawing 1: The top level entity of the key board interface.

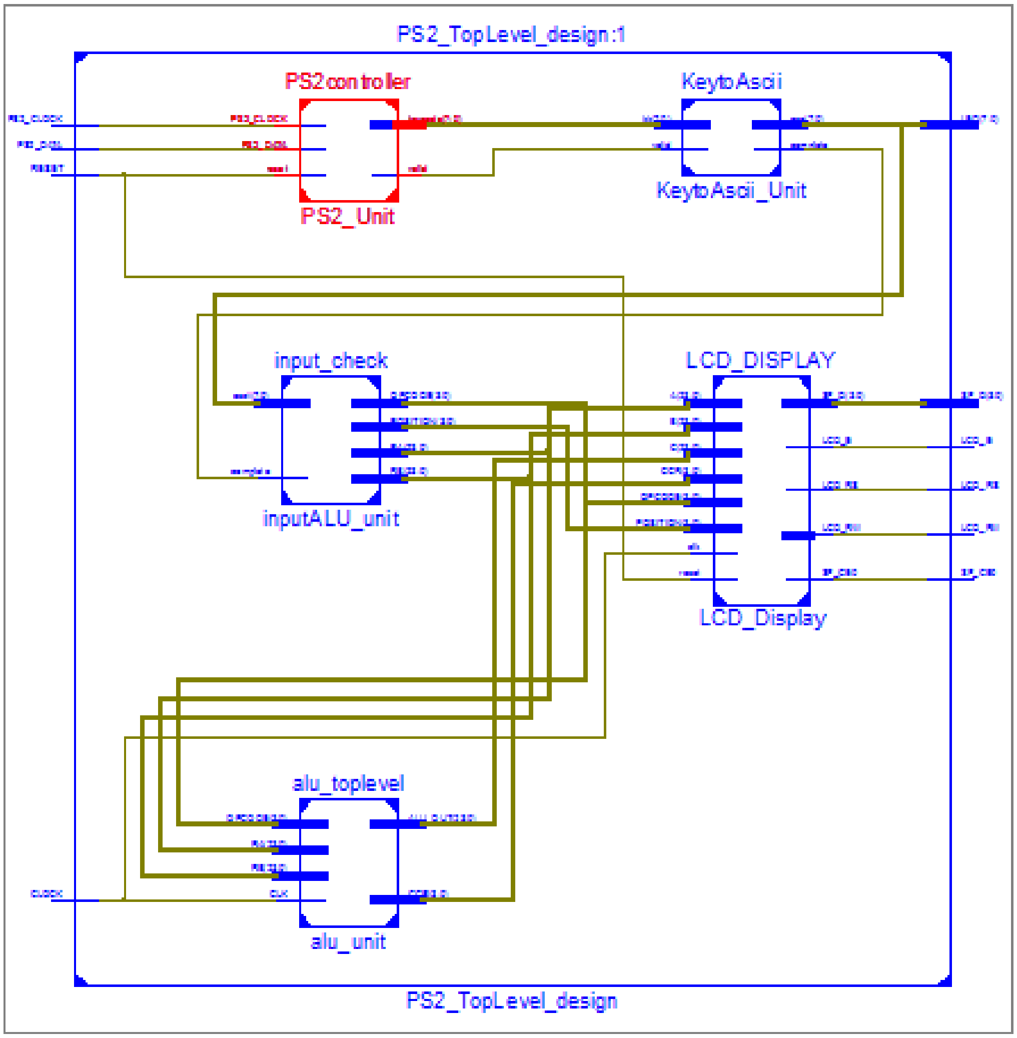


Drawing 2: Top level entity for the PS/2 Driver interfaced with the ALU and the LCD

VHDL Code

The VHDL code for both the counter and ALU has been emailed as zip files to Dr. Fortier.

Generated RTL Designs with comparisons.



Drawing 3: RTL Diagram for the PS/2 controller interfaced with the ALU and the LCD

Between the various RTL diagrams we incorporated, there were various types of entities in the lab. A lot of the RTL diagrams were based from lab 1 and the only difference was to change the input from button and switches to actual ASCII code. Overall with each RTL diagram, we reused some from the ALU lab and then incorporated new segments. The first new section of the code was the PS2 controller which was supplied to us from DR. Fortier. We then incorporated an overlay converter from key code to ASCII and replaced the original lab 1 input de-bouncer to an ASCII version which was an input checker.

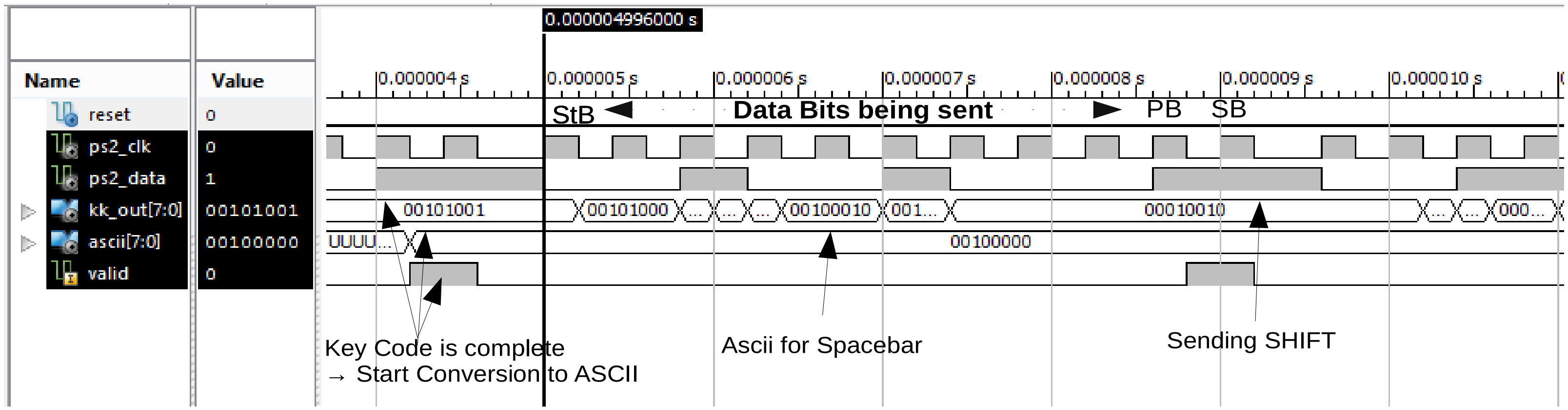
Test Plan

Overall plan (simulation and HW): For the overall plan in the simulation, the LCD was used to act as a test for the inputs from the keyboard to the display. This was then used to tie with the ALU.

Simulation: The current simulation for the PS2 controller is the following figure 4. When we incorporate the PS2 driver, it will wait till it grabs a 8 bit code from the keyboard and then was sent through a convert to see the ascii. The testing and observation of the Lab done more in the hardware level instead of simulation. This was due to the testing each keycode to ascii and then display on the LCD.

Testbench and simulation results: Overall the test bench and simulation results we on the mark for each function they had to achieve.

Hardware test plan and procedure (what will be tested, how you will do it): For the hardware test plan and procedure's, the first part was to observer the LCD changes with the keyboard inputs. Then observed the changes that the keyboard did upon the ALU input.



Name	Value
reset	0
ps2_clk	0
ps2_data	1
kk_out[7:0]	00101001
ascii[7:0]	00100000
valid	0

Key Code is complete
→ Start Conversion to ASCII

Ascii for Spacebar

Sending SHIFT

StB: *Start bit*
PB: *Parity Bit*
SB: *Stop Bit*

Drawing 4: Simulation of the key strokes: space bar, shift+1

Test Bench Output

```
Simulator is doing circuit initialization process.
at 0 ps: Note: starting PS2 Keyboard testing (/ps2_keyboard_tb/).
Finished circuit initialization process.
at 400 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Start Bit
at 800 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Data
at 4 us, Instance /ps2_keyboard_tb/ : Warning: Sending Parity Bit
at 5 us: Note: starting start for !, shift 1  (/ps2_keyboard_tb/).
at 5 us, Instance /ps2_keyboard_tb/ : Warning: Sending Start Bit
at 5400 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Data
at 8600 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Parity Bit
at 9600 ns: Note: starting shift 1  (/ps2_keyboard_tb/).
at 9600 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Start Bit
at 10 us, Instance /ps2_keyboard_tb/ : Warning: Sending Data
at 13200 ns, Instance /ps2_keyboard_tb/ : Warning: Sending Parity Bit
at 14 us: Note: Test sequence completed. (/ps2_keyboard_tb/).
```

Result Matrix

	Key Board Inputs			LCD Output					Test Results
	A	B	Opcode	A	B	Opcode	F	CCR	
Arithmetic Tests	h, 5	j, 3	SHIFT+ =	“000005”	“000003”	"++"	“000008”	---	Passed
	h, 5	j, 3	-	“000005”	“000003”	"- -"	“000002”	---	Passed
	h, 5	j, 3	SHIFT + 7	“000005”	“000003”	"&&"	“000001”	---	Passed
	h, 5	j, 3	SHIFT + \	“000005”	“000003”	"||"	“000007”	---	Passed
	h, 64	j, 32	SHIFT+ =	“000064”	“000032”	"++"	“000096”	---	Passed
	h, 64	j, 32	-	“000064”	“000032”	"- -"	“000032”	---	Passed
	h, 64	j, 32	SHIFT + 7	“000064”	“000032”	"&&"	“000020”	---	Passed
	h, 64	j, 32	SHIFT + \	“000064”	“000032”	"||"	“000076”	---	Passed
CCR Tests	h, 0	j, 0	SHIFT+ =	“000000”	“000000”	"++"	“000000”	-z--	Passed
	h, 1	j, FFFFFF	SHIFT+ =	“000001”	“FFFFFF”	"++"	“000000”	-zoc	Passed
	h, 7FFFFF	j, 1	SHIFT+ =	“7FFFFF”	“000001”	“++”	“800000”	n-o-	Passed
	h, FFFFFF	j, 1	SHIFT+ =	“FFFFFF”	“000001”	“++”	“000000”	-zoc	Passed
Mem Tests	h, FFFFFF	j, 1	,	“FFFFFF”	“000001”	“,,”	“000000”	----	Passed
	h, 16	j, 1	.	“000016”	“000001”	“..”	“000000”	----	Passed
	h, 16	j, 1	,	“000016”	“000001”	“,,”	“000016”	----	Passed

Table 1: Shows the test results of all test based on various inputs.

UCF File

The UCF for this lab has been emailed to Dr. Fortier.

Conclusion

Through this lab we were able to enhance our understanding towards top level entities and connections between components. Moreover, we learned how the PS/2 key board works and we implemented it as interface for the ALU. Through this implementation we passed through various stages. Starting from the design layout, to code implementation, to debugging to test bench simulation, and finally to hardware testing. The hardware testing involved outputting the key pressed at first, then testing with the ALU.

We implemented both of the extra credit where we were able to interpret a keyboard input. Moreover, translate the input into opcodes and operands for the UMD_ALU_FPU and display all the inputs (RA, RB, opcode) and the outputs (CCR, Result and Position).

The results of our unit were as expected, where we used special keys to inputs into the registers and the opcodes. This was performed by entering the key 'h' the data will be saved into the register A, while by entering the key 'j' the data will be saved into register B. Moreover, it automatically detects the opcodes by entering there known symbols. Finally, the display on the LCD was successful.

Reflection

Overall with this lab was one of a kind. Even with the shrinking time to work on the lab compared to the actual due date being. This lab was an interesting project since we learned how the PS2 controller was managed and create a driver to grab key codes and convert to ASCII. This lab had us dive deeper inside VHDL. At first, we got the PS/2 driver to work and function, but the daunting task was to tie it to the ALU. We had a key code to ASCII converter and tested the output on the LCD. In order for the ALU to properly function, we implemented an input check circuit that will select which register were are controlling.

One of the many problems we incountered and resolved was the ASCII converter. Originally, we used the supplied converter from Dr. Fortier but as the more we debug the more we noticed how many problems there were. One of the bugs was it did not differentiate lower and upper case ascii. So we incorporated a checker for both make and break codes. Once debugging the ascii converted. We then focused on attaching the ALU & LCD from lab 1. Upon assembling each part together, an input check was created to act as overlay for the keyboard to the ALU.

With all the progress of the Lab, one of the highlights was to create a 32 byte memory that would be acting like a buffer for the LCD. The memory was a dual access buffer with a data in & address in while also having a data out & address out. Later we created a test bench and proceeded to debug it. We consulted Dr. Fortier on improving the read/ write and simulation test, and he helped improve it by have a cycle. This memory will eventually be used as a memory buffer for the LCD but it still need more test.

With the daunting time limit of the lab we noticed how we must be on top of each lab in order to properly finished it and move to the next lab. This semester feels so short and with the short timeline for each of the labs. It feels like the semester is going to be a long one due to the amount of work in the labs.