EE 3610 Digital Systems
Lab 3

Title: Keyboard Interface.

Objective: The student will gain a basic understanding of synchronous communication, signal debouncing and data translation using a lookup table.

Equipment: Spartan 3E Starter Board
9-pin D-Sub Cable.
PC keyboard with a PS/2 interface
Computer Configured as a Terminal Emulator

Design: In this lab, you are to design a system that receives scan codes from the keyboard, generates the corresponding ASCII characters, and outputs them on the RS-232 port

Your top level module should be named lab3 (in file lab3.vhd) and should contain at least three modules. The first is a PS/2 interface module that takes synchronous serial data from the PS/2 clock and data inputs and produces 8-bit parallel data out. The keyboard generates scan codes instead of ASCII, so you will need to design a second module to perform that conversion. The third module is the one you designed in lab2. It takes ASCII data and sends it out the asynchronous serial port. You may either remove the LCD driver or connect it to the ASCII output of the conversion module.

PS/2 Interface Module
Design a VHDL module to interface the PS/2 port. Your module should take four inputs: asynchronous reset, clock, PS/2 data and PS/2 clock. Your module should have an 8-bit parallel output and a “ready” output that is asserted for one cycle each time a scan code is received.

The PS/2 interface should shift the PS/2 data into a shift register on the falling edge of the PS/2 clock input. Each data frame consists of one start bit, eight data bits, a parity bit and a stop bit. After these 11 bits have been received, the PS/2 module should output the 8 data bits (i.e. scan code) and start over.

You needn’t scan for a start bit or measure time in any way. Simply watch for PS/2 clock to go from high to low, then shift the PS/2 data into a shift register. After eleven PS/2 clocks, output the 2nd through 9th bits and start over.

Unfortunately, the PS/2 clock is slow, and its transitions can be noisy. It can seem to oscillate for as long as 500ns as it works its way from high to low or vice versa. If you do not debounce it, you can easily miscount the bits. One way to debounce a signal is to use an up-down counter. If the

PS/2 clock is high, increment the counter (unless it is already at its maximum value). If it is low, decrement the counter. When you reach the maximum count, set the internal (clean) PS/2 clock high. When you reach the minimum, set the internal PS/2 clock low. Then use the falling edge of the internal PS/2 clock to drive the shift register.

Scan Code Conversion Module
Design a second VHDL module to convert scan codes to ASCII. Your module should take four inputs: asynchronous reset, clock, parallel data and data valid (which is asserted for 1 clock cycle to indicate that a valid scan code is on the parallel input). Your module should generate a 7- or 8-bit output for ASCII data and a "ready" output that is asserted for one cycle each time an ASCII character is available.

For most keys, the PS/2 keyboard generates a 1-byte scan code each time the key is pressed and a 2-byte scan code each time the key is released. The key-release code is always the byte F0 followed by the key-press scan code. You will have to keep track of the state of the shift and control keys to generate the correct ASCII code. (e.g. when the shift key is down, scan code 1E should produce an @ character, but if the shift key is up, the same scan code should produce the number 2.) You can do this by including a SHIFT flip-flop that is set when a shift scan code (12 or 59) is received and cleared when a release+shift scan code (F0+12 or F0+59) is received. Similarly, you can include a CTRL flip-flop that is set and cleared when the CTRL key is pressed (code 14) or released (code F0+14), respectively. (You will need a simple state machine to keep track of whether or not the release code, F0, was just received).

Your module must support all the printable characters (except those on the keypad) plus the Backspace, Tab, Enter and Esc keys. You must also handle control characters. For example, if the J key is pressed while the CTRL key is down, your module should output a 0A (line feed). An easy way to do this is to concatenate the CTRL state, the SHIFT state and the scan code into 10-bit index, then use that index to lookup ASCII character codes in a lookup table. (A VHDL declaration for such a lookup table is provided; see the scan2ascii link on the course website.) Many scan codes do not translate to ASCII characters. For these scan codes, the lookup table contains 7F. This code should be used to indicate that the scan code is invalid and should be ignored.

Preparation: Write the title and a short description of this lab in your lab book. Make sure the page is numbered and make an entry in the table of contents for this lab.

Draw a schematic that includes only the components you will be using for this lab. Specifically, include the FPGA (but show only the pins you are using), the MAX3232, the DB-9 connector and the PS/2 connector (See the Spartan 3E Starter Board Schematic.) Include pin numbers or coordinates. Omit the power supply.

Write a separate, independent test bench for each of the two modules described above. Simulate the designs, verify they work and affix the simulation waveforms to your lab book.

Design the lab3 top level module that connects your PS/2 module to your ASCII converter and your ASCII converter to your RS-232 transmitter (and possibly the LCD). Create a user constraints file to configure the clock, reset, PS/2 clock, PS/2 data and TXD pins and synthesize your design to verify that there are no errors.

Bring your lab notebook and the Spartan board, above, to your lab period.

Set up: Connect the USB cable, serial cable and power supply to the Spartan board. Connect the other end of the serial cable to the computer and run a terminal emulation program. Configure the serial protocol to be 9600 baud and have 8 data bits, no parity and 1 stop bit. Connect a keyboard to the PS/2 port of the Spartan 3E board.

Procedure: Download your code to the Spartan board. Verify that each time a key on the PS/2 keyboard is pressed, that character is displayed on the terminal emulator. Check that Enter, Tab, Backspace, and CTRL+J (Line Feed) move the cursor as expected. Demonstrate your system to the lab instructor.

Conclusions: In the conclusion section, write a short summary of what you did, what you learned, and what could be done better.