Engineering 378 Digital Systems Design

Exp. 5: Keyboard Interface

Submitted by:

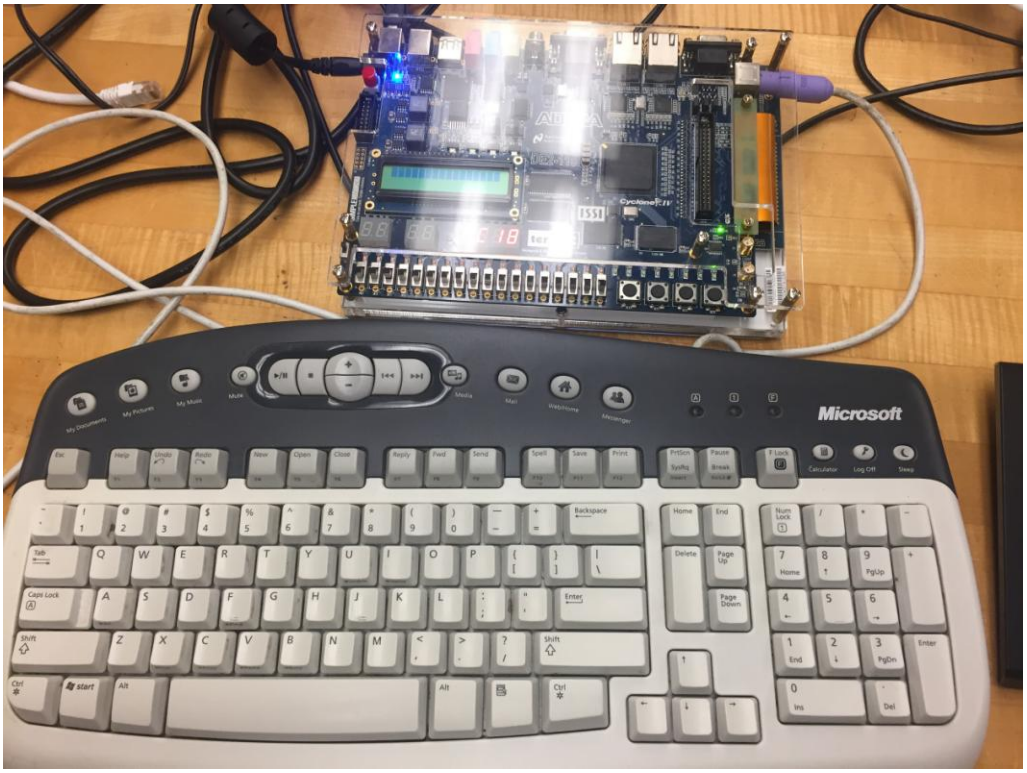
Norald Alejo
Farnam Adelkhani

November 10, 2016

Demonstration	_/5
Report: Presentation	_/2
Problem: analysis	_/2
Design: work	_/5
Results	_/5
Conclusion/Discussion	_/1
Total	_/20

Problem Analysis

This lab's objective is to learn how to interface a computer keyboard. This is where the PS/2 port is connected to the FPGA. The user sends 8 or 16 data bits by pressing a key on the keyboard and the FPGA displays the corresponding hexadecimal codes. Keys such as the arrow keys or the Ctrl/Alt keys have two transmissions while the rest transmits only one transmission.



DE2-115 Board

Hardware Design

The pre-lab was done by drawing a hardware schematic of how the data will capture the data transmitted by the keyboard. The last two of the data packets will display the data. So one module will acquire data from the keyboard and the other module will display and shift the signals.

Verilog Modeling

The module will begin @ the positive edge of the output then check for stop sequences where if found positive then it will shift in the code. A Reset signal is implemented. The receiver has an always statement triggered @ the negative edge of the ps/2 clock signal. The state is found from the current position in the sequence. If the start bit is 1 then the proceeding 8-bits are put into a buffer and then it checks @ the parity before checking for stop bit.

```
module keyBoard(
        output reg [6:0] Digi0, Digi1, Digi2, Digi3, // Out to 7-segment display
        input PS2_CLK, PS2_DAT, // in -- clock/data from PS/2
        input [0:0] KEY // Used to reset from board key
);

wire new;
wire [7:0] char;
reg [7:0] last, last2;

receiver M1(.new(new), .char(char), .PS2_CLK(PS2_CLK), .PS2_DAT(PS2_DAT), .reset(!KEY[0]));

always @ (posedge new or negedge KEY[0]) begin
        if (!KEY[0]) begin
                Digi0 = 7'b1111111; // OFF
                Digi1 = 7'b1111111; // OFF
                Digi2 = 7'b1111111; // OFF
                Digi3 = 7'b1111111; // OFF
end

else if (char == 8'hF0) begin // only display the last value after the button is released

if (last == 8'hE0) begin //for displaying code of buttons sending 16-bit signal
        Digi2 = display_driver(last[3:0]);
        Digi3 = display_driver(last[7:4]);
        Digi0 = display_driver(last2[3:0]);
        Digi1 = display_driver(last2[7:4]);
end

else begin // current values displayed are shifted
        Digi2 = Digi0;
        Digi3 = Digi1;
        Digi0 = display_driver(last[3:0]);
        Digi1 = display_driver(last[7:4]);
  end
end
else begin
        last2 = last;
        last = char;
end
end

function [6:0] display_driver;
input [3:0] in;
case (in)
4'b0000 : display_driver = 7'b1000000; // 0
4'b0001 : display_driver = 7'b1111001; // 1
4'b0010 : display_driver = 7'b0100100; // 2
4'b0011 : display_driver = 7'b0110000; // 3
4'b0100 : display_driver = 7'b0011001; // 4
4'b0101 : display_driver = 7'b0010010; // 5
4'b0110 : display_driver = 7'b0000011; // 6
4'b0111 : display_driver = 7'b1111000; // 7
4'b1000 : display_driver = 7'b0000000; // 8
4'b1001 : display_driver = 7'b0011000; // 9
4'b1010 : display_driver = 7'b0001000; // A
4'b1011 : display_driver = 7'b0000000; // B
4'b1100 : display_driver = 7'b1000110; // C
4'b1101 : display_driver = 7'b1000000; // D
4'b1110 : display_driver = 7'b0000110; // E
4'b1111 : display_driver = 7'b0001110; // F
default : display_driver = 7'b1111111; // OFF
end case
end function
end module
```

```
module receiver(
                output reg new,
                output reg[7:0] char,
                input PS2_CLK, PS2_DAT, reset
);
reg[3:0] bitPos; // Current bit position in transmission
wire parityCheck;

xor M1(parityCheck, char[0], char[1], char[2], char[3], char[4], char[5], char[6], char[7]);

always @ (negedge PS2_CLK) //triggered @ negedge of PS/2
        begin
                new = 0;
                if (reset) bitPos = 0; // Reset bit position
                else if (bitPos == 0 & PS2_DAT == 0) bitPos = 1; // Check start bit
        else if (bitPos >= 1 & bitPos <= 8) begin
                char[bitPos - 1] = PS2_DAT;
                bitPos = bitPos + 1'd1;
        end
        else if (bitPos == 9) begin // Check parity bit
                if (PS2_DAT == ~parityCheck)
                        bitPos = 10; // Go to stop bit state
                else
                        bitPos = 0; // Error in transmission
end
else if (bitPos == 10 & PS2_DAT == 1) begin // Check stop bit
        new = 1;
        bitPos = 0;
end
else bitPos = 0;
end

end module
```

Results/Verification:

Verification of the functionality of the system is achieved by downloading the verilog code into the FPGA board. The keyboard was connected to the FPGA using the PS/2 port. Each key was tested to see if it matched the corresponding matching data. The 7-segment display of the FPGA board showed the values in hexadecimal format.

The data is inputted through the keyboard. When a key is pressed, the hexadecimal is displayed. Then, when the next key is pressed, the previous data is shifted to the left two signals. For example, when “H” is hit, hexadecimal “33” is displayed on the FPGA board.

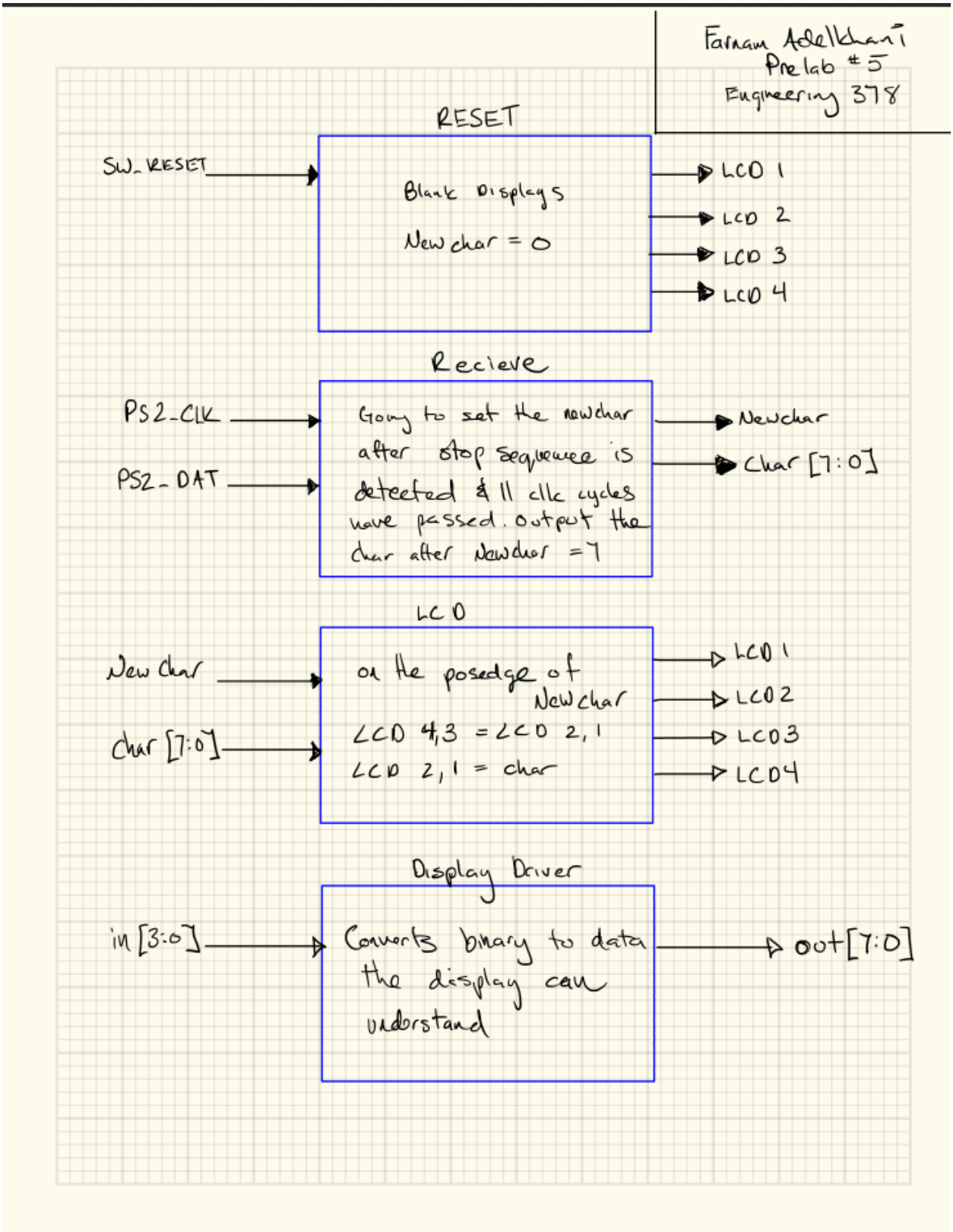
Conclusion and Discussion:

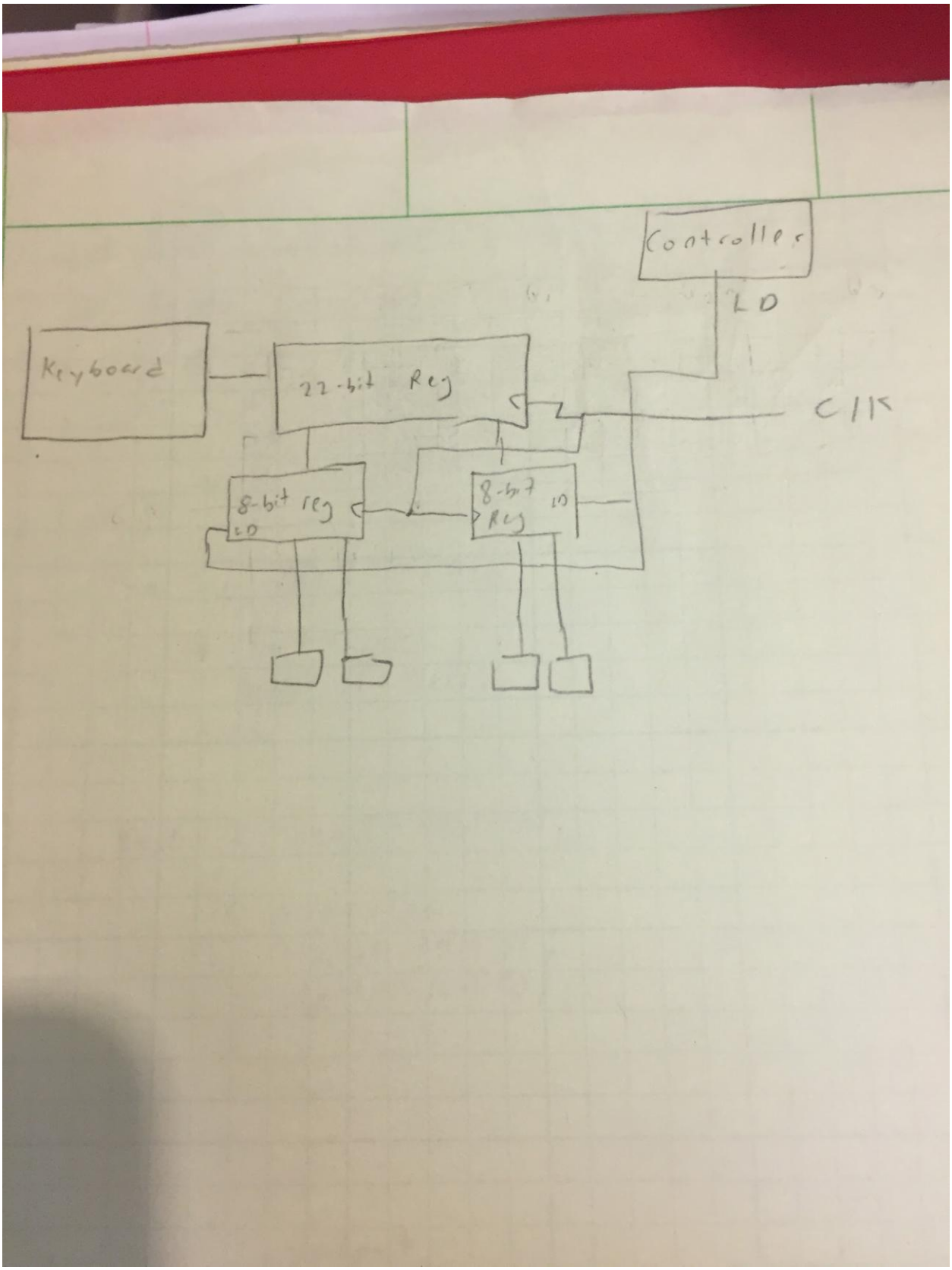
As engineer majors, the ability to use an input using a keyboard is significant in our field of study. This labs gives an powerful illustration on the user and computer interaction. This allows the user to interact with the board and the board then displays the input. The challenge in this assignment was creating a design that captures the data transmitted by the keyboard.

Work Breakdown

- Pre-lab assignments were completed individually.
- The laboratory assignment was completed as a team effort.

Prelabs


Farnam Adelkhani
Prelab #5
Engineering 378
RESET
SW_RESET
Blank Displays
New char = 0
LCD 1
LCD 2
LCD 3
LCD 4
Recieve
PS2_CLK
PS2_DAT
Going to set the newchar after stop sequence is detected & 11 clk cycles have passed. output the char after Newchar = 1
Newchar
Char [7:0]
LCD
New Char
Char [7:0]
on the posedge of Newchar
LCD 4,3 = LCD 2,1
LCD 2,1 = char
LCD1
LCD2
LCD3
LCD4
Display Driver
in [3:0]
Converts binary to data the display can understand
out [7:0]


Controller
LD
Keyboard
22-bit Reg
CLK
8-bit reg
LD
8-bit
Reg
LD