Weifan Lin
Csc343
05/14/2015

Keyboard

Objectives:

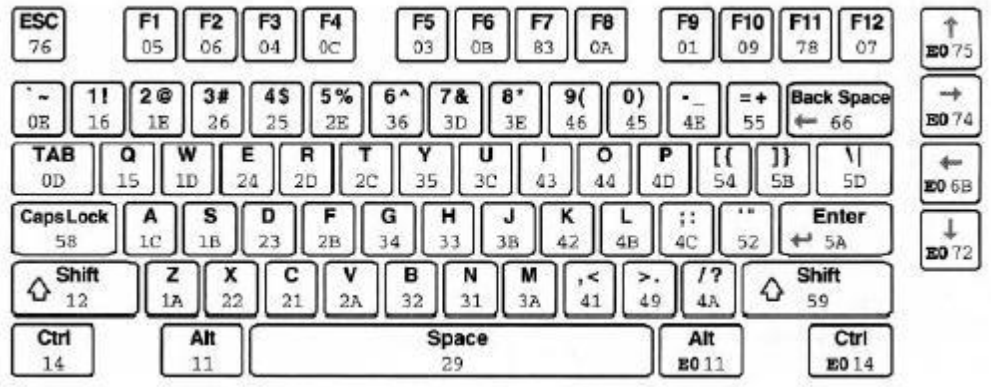
The goal of this lab was to gain an understanding of the ps2 communication protocol and to design and implement a ps2 keyboard. At the end of this lab, we wished to be able to print out our name based on input from a ps2 keyboard.

Specifications:

- PS2 Receiver – This system has five 1-bit input signals, Clk, Reset, ps2Data, ps2Clock, and rec_en. Also, this system has two output signals, a 1-bit rec_done signal, and a 8-bit Dout signal.
- Shift Dectector – This system has three input signal, two 1-bit signals, clk and done, and one 8-bit signal, scan. Also, this system has one 1-bit output signal, shift.
- Ascii Converter – This system has two input signals, a 1-bit shift signal and an 8-bit scan_code signal. Also, this system has an 8-bit output signal, ascii.
- Keyboard Interface – This system has three 1-bit input signals, Clock, Reset, and En, and three output signals, two 1-bit signals, ps2D and ps2C, and one 56-bit display signal. This system requires the ps2 receiver, the shift detector, and the ascii converter.

Functionality:

- PS2 Receiver – This system has two functions, one of which is to output the scan codes of the keys on the keyboard being pressed, and the other being to correctly identify the scan codes and neglect noise or jitter. To preform the first function, the ps2 receiver uses a finite state machine with three states, idle, receive, and done. This finite state machine is shown below in Figure 1, where the variable N represents a count down of the iterations of the system.



- Shift Detector – The function of the shift detector is to determine when the “Shift” button on the keyboard has been pressed. To accomplish this task, a finite state machine with three states, idle, pressed, and released, is used. This finite state machine is illustrated below in Figure 3, where scan represents the scan codes captured from the previous system, PS2 Receiver.
- Keyboard Interface – The function of this system is to map together all the previous components to a seven-segment display system that will display not only the ASCII conversion of the keys, but also the scan codes. To display the ASCII and scan codes on the seven-segment display, this system utilizes another system, the Hex to Seven converter. The Hex to Seven converter's function is to map

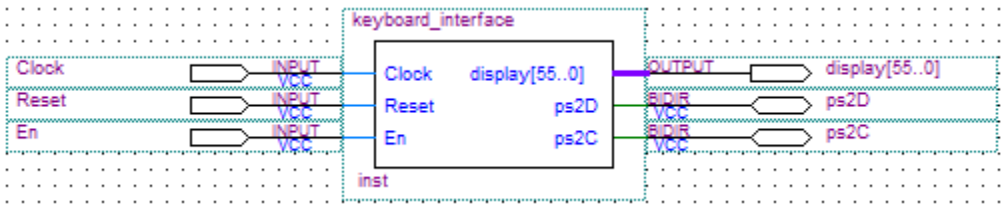
together eight seven- segment displays to display a thirty-two bit input signal.

Design:

Scan Code to Ascii:

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.all;
3   USE IEEE.STD_LOGIC_ARITH.all;
4   USE IEEE.STD_LOGIC_UNSIGNED.all;
5
6   entity scan_code_to_ascii is
7   port(
8   scan_code: in std_logic_vector(7 downto 0);
9   shift: in std_logic;
10  ascii: out std_logic_vector(7 downto 0)
11  );
12  end scan_code_to_ascii;
13
14  architecture arch of scan_code_to_ascii is
15
16  begin
17
18  ascii <= "01010111" when shift='1' and scan_code="00011101" --1D W
19      else "01100101" when shift='0' and scan_code="00100100" --24 e
20      else "01101001" when shift='0' and scan_code="01000011" --43 i
21      else "01000110" when shift='1' and scan_code="00101011" --2B F
22      else "01100001" when shift='0' and scan_code="01000011" --1C a
23      else "01101110" when shift='0' and scan_code="00011100" --31 n
24
25  end arch;
```

Keyboard:



Testing on DE2 Board

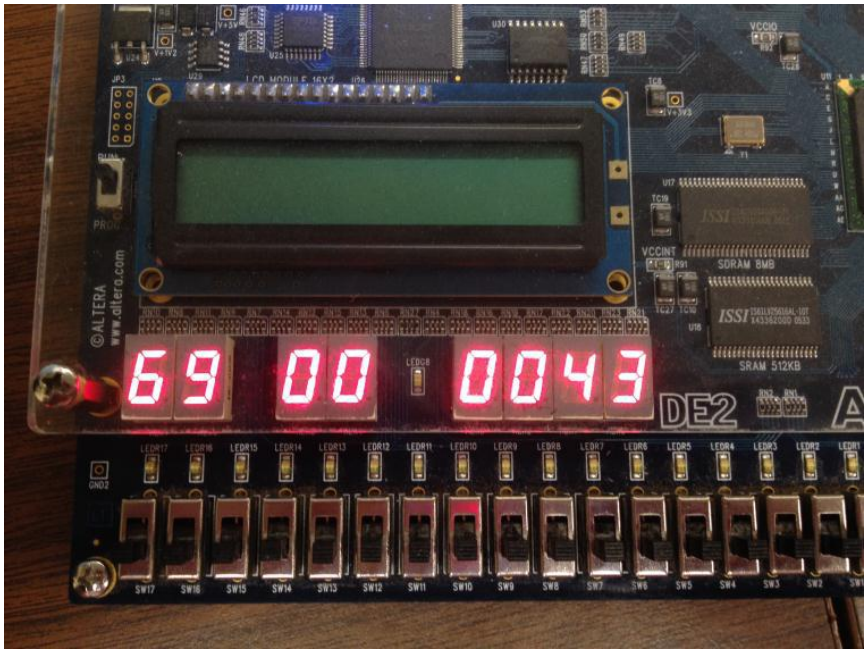


Figure above represents letter i

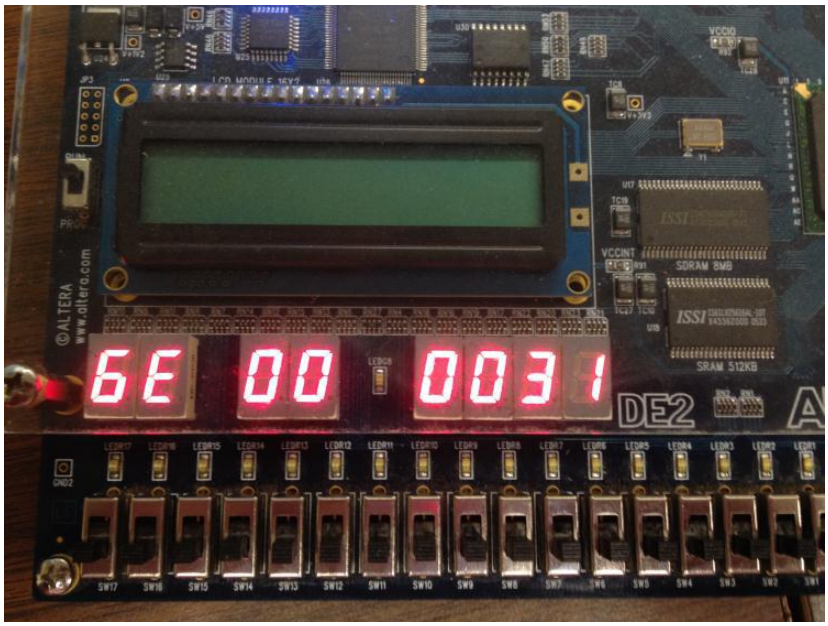


Figure above represents letter n

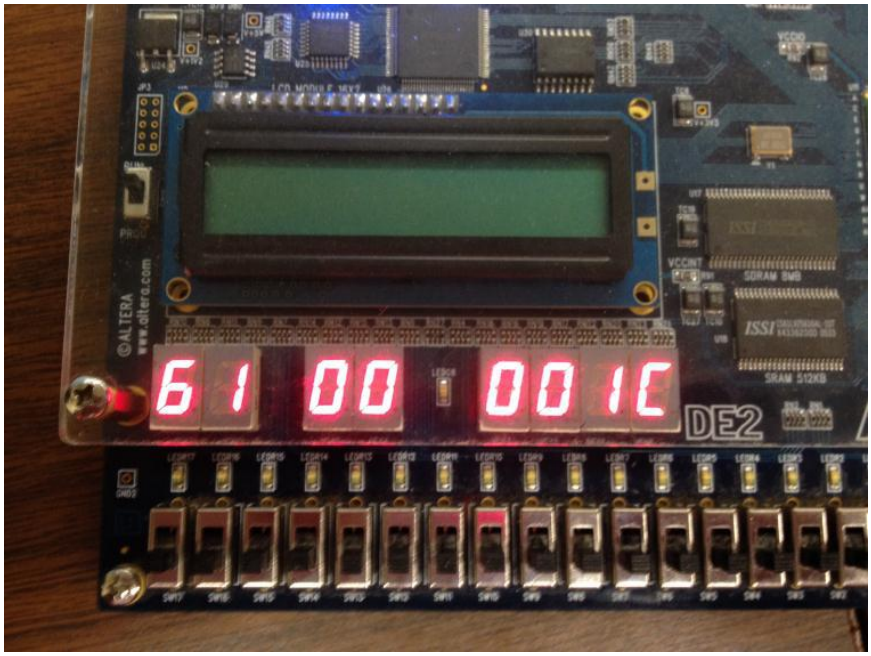


Figure above represents letter a

Conclusion:

The objective of this lab was achieved because after designing the individual part that make up the keyboard interface, we tested the system on the DE2 board and it preformed as expected. The first system we had to make was the PS2 Receiver, which interprets the keys the user pressed into scan codes. The next system was the shift detector, which takes the scan code from the previous system and determines if the shift key was pressed. The final system combines the two previous systems and converts the scan code and shift signal into the ascii representation of the key being pressed.

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

entity scan_code_to_ascii is
port(
scan_code: in std_logic_vector(7 downto 0);
shift: in std_logic;
ascii: out std_logic_vector(7 downto 0)
);
end scan_code_to_ascii;

architecture arch of scan_code_to_ascii is

begin

ascii <= "01010111" when shift='1' and scan_code="00011101" --1D W
    else "01100101" when shift='0' and scan_code="00100100" --24 e
    else "01101001" when shift='0' and scan_code="01000011" --43 i
    else "01000110" when shift='1' and scan_code="00101011" --2B F
    else "01100001" when shift='0' and scan_code="01000011" --1C a
    else "01101110" when shift='0' and scan_code="00011100" --31 n

end arch;
```

Appendix:

Pin assignments:

To, Location

Reset, PIN_G26

Clock, PIN_N2

En, PIN_N23

ps2D, PIN_C24

ps2C, PIN_D26

display[0], PIN_AF10

display[1], PIN_AB12

display[2], PIN_AC12

display[3], PIN_AD11

display[4], PIN_AE11

display[5], PIN_V14

display[6], PIN_V13

display[7], PIN_V20

display[8], PIN_V21

display[9], PIN_W21

display[10], PIN_Y22

display[11], PIN_AA24

display[12], PIN_AA23

display[13], PIN_AB24

display[14], PIN_AB23

display[15], PIN_V22

display[16], PIN_AC25

display[17], PIN_AC26

display[18], PIN_AB26

display[19], PIN_AB25

display[20], PIN_Y24

display[21], PIN_Y23

display[22], PIN_AA25

display[23], PIN_AA26

display[24], PIN_Y26

display[25], PIN_Y25

display[26], PIN_U22

display[27], PIN_W24

display[28], PIN_U9

display[29], PIN_U1

display[30], PIN_U2

display[31], PIN_T4

display[32], PIN_R7

display[33], PIN_R6

display[34], PIN_T3

display[35], PIN_T2

display[36], PIN_P6

display[37], PIN_P7

display[38], PIN_T9

display[39], PIN_R5

display[40], PIN_R4

display[41], PIN_R3

display[42], PIN_R2

display[43], PIN_P4

display[44], PIN_P3

display[45], PIN_M2

display[46], PIN_M3

display[47], PIN_M5

display[48], PIN_M4

display[49], PIN_L3

display[50], PIN_L2

display[51], PIN_L9

display[52], PIN_L6

display[53], PIN_L7

display[54], PIN_P9

display[55], PIN_N9

Keyboard interface VHDL:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY keyboard_interface IS
	PORT
	(
		Clock :  IN  STD_LOGIC;
		Reset :  IN  STD_LOGIC;
		En :  IN  STD_LOGIC;
		display: out std_logic_vector(55 downto 0);
		ps2D :  INOUT  STD_LOGIC;
		ps2C :  INOUT  STD_LOGIC
	);
END keyboard_interface;

ARCHITECTURE bdf_type OF keyboard_interface IS

COMPONENT ps2_rec
	PORT(Clk : IN STD_LOGIC;
		 Reset : IN STD_LOGIC;
		 ps2Data : IN STD_LOGIC;
```

```
            ps2Clock : IN STD_LOGIC;
            rec_en : IN STD_LOGIC;
            rec_done : OUT STD_LOGIC;
            Dout : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
END COMPONENT;

COMPONENT hex_to_sevenPORT(en : IN STD_LOGIC;
            input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            HEX0 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX1 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX2 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX3 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX4 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX5 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX6 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
            HEX7 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
        );
END COMPONENT;

COMPONENT shift_detector
        PORT(clk : IN STD_LOGIC;
            done : IN STD_LOGIC;
            hex : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            shift : OUT STD_LOGIC
```

```
        );
END COMPONENT;

COMPONENT scan_code_to_ascii
        PORT(shift : IN STD_LOGIC;
                scan_code : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
                ascii : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
END COMPONENT;


SIGNAL      NotReset :  STD_LOGIC;
SIGNAL      rec_done :  STD_LOGIC;
SIGNAL      dis_input :  STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL      ScanCode :  STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL      shift_output :  STD_LOGIC;
SIGNAL      asci_output :  STD_LOGIC_VECTOR(7 DOWNTO 0);


BEGIN



rec : ps2_rec
PORT MAP(Clk => Clock,
```

```
                Reset => NotReset,
                ps2Data => ps2D,
                ps2Clock => ps2C,
                rec_en => En,
                rec_done => rec_done,
                Dout => ScanCode );


sev : hex_to_seven
PORT MAP(en => rec_done,
                input => dis_input,
                HEX0 => display(6 downto 0),
                HEX1 => display(13 downto 7),
                HEX2 => display(20 downto 14),
                HEX3 => display(27 downto 21),
                HEX4 => display(34 downto 28),
                HEX5 => display(41 downto 35),
                HEX6 => display(48 downto 42),
                HEX7 => display(55 downto 49)
);

dis_input <= asci_output & "0000000000000000" & ScanCode;


NotReset <= NOT(Reset);
```

```
shft : shift_detector
PORT MAP(clk => Clock,
                done => rec_done,
                hex => ScanCode,
                shift => shift_output);
ascii : scan_code_to_ascii
PORT MAP(shift => shift_output,
                scan_code => ScanCode,
                ascii => asci_output);
END bdf_type;
```

PS2 Receiver VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity ps2_rec is
        port
        (
                Clk ,Reset: in std_logic;
                ps2Data : in std_logic;
                ps2Clock : in std_logic;
                rec_en : in std_logic;
                rec_done: out std_logic;
                Dout: out std_logic_vector(7 downto 0)
```

```
        );
end ps2_rec;

architecture arch of ps2_rec is

type statetype is (idle, receive, done);
signal current_state, next_state : statetype;
signal current_data, next_data: std_logic_vector(10 downto 0);
signal current_n, next_n        : unsigned(3 downto 0);
signal fall_edge: std_logic;
signal current_filter, next_filter : std_logic_vector(7 downto 0);
signal current_filter_clock, next_filter_clock : std_logic;

begin

process(clk, reset)
begin
        if reset = '1' then
                current_filter <= (others => '0');
                current_filter_clock <= '0';
        elsif (rising_edge(clk)) then
                current_filter <= next_filter;
                current_filter_clock <= next_filter_clock;
        end if;
end process;
```

```
next_filter <= ps2clock & current_filter(7 downto 1);
next_filter_clock <= '1' when current_filter = "11111111" else
                                        '0' when current_filter = "00000000" else
                                        current_filter_clock;
fall_edge <= current_filter_clock and (not next_filter_clock);



process (Clk, reset)
begin
        if(reset = '1') then
                current_state <= idle;
                current_n <= (others => '0');
                current_data <= (others => '0');
        elsif(rising_edge(Clk)) then
                current_state <= next_state;
                current_n <= next_n;
                current_data <= next_data;
        end if;
end process;

process(current_state, current_data, current_n, ps2Data, fall_edge)
begin
        rec_done <= '0';
        next_state <= current_state;
```

```
        next_n <= current_n;
        next_data <= current_data;
        case current_state is

            when idle =>
                if(fall_edge = '1' and rec_en = '1') then
                    next_data <= ps2Data & current_data(10 downto 1);
                    next_n <= "1001";
                    next_state <= receive;
                end if;
            when receive =>
                if(fall_edge = '1') then
                    next_data <= ps2Data & current_data(10 downto 1);
                    if current_n = 0 then
                        next_state <= done;
                    else
                        next_n <= current_n - 1;
                    end if;
                end if;
            when done =>
                next_state <= idle;
                rec_done <= '1';
        end case;
end process;
Dout <= current_data(8 downto 1);
```

```
end arch;
```

Shift Detector VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity shift_detector is
        port
        (clk, done: in std_logic;
        hex: std_logic_vector(7 downto 0);
        shift: out std_logic
        );
end shift_detector;

architecture arch of shift_detector is

type statetype is (idle, shft, f0, pause);
signal current_state, next_state : statetype;

begin

process(clk)
begin
        if(rising_edge(clk)) then
```

```
        current_state <= next_state;
end if;
end process;

process(hex, done)
begin

shift <= '0';
next_state <= current_state;

case current_state is
        when idle =>
        shift <= '0';

        if(done = '1' and hex = "00010010") then
        next_state <= shft;
        end if;

        when shft =>
        shift <= '1';

        if(done = '1' and hex = "11110000") then
        next_state <= f0;
        end if;
```

```
        when f0 =>
        shift <= '1';
        if(done = '1') then
                if(hex= "00010010") then
                        next_state <= idle;
                else
                next_state <= shft;
                end if;
        end if;

        when pause =>
        shift <= '1';
end case;

end process;

end arch;
```