COE3DQ5 – Lab # 2 Report

Group # 33
Fahad Mahmood – 001414984 – mahmof4@mcmaster.ca
Wei Che Kao – 001328256 – kaow@mcmaster.ca
September 26, 2018

Exercise 1:

For exercise 1, we first assign keys for PS2_keyboard code F,P,G and A. in order to avoid the make code to continuously re-sending by the keyboard until the key is released, we check for conditions that if we release the keyboard, PS2_make_code will equal to zero because we are checking for releasing, and PS2_code doesn't have F0 since we already release the keyboard, so it will only load the first make code to the register.

We also create finite state machine for FPGA, so we set the light flag at state g because as soon as key A is pressed and released, light flag will change (turn off turn on). We also have 32 bits shift register that passing make code values and displaying from right to left. Lastly, we assign conditions in seven segment display. If light flag value is 1, all lights turn off, otherwise the seven-segment display will turn on.

And when seven segment display are all turn off, you type FPGA again and at state g, flag will flip again, and the seven-segment display will turn on again since flag flip from 1 to 0.

Exercise 2:

First, we extended our mif file provided with the lab to extend it for uppercase characters. We created a flag called shift_key to keep track of when the right shift key or left shift was pressed and accordingly assigned value of 0 and 1 using if statements that read the make codes for them. To display the characters as they were typed we used our understanding for experiment 4 and displayed our PS2 codes instead of displaying the last register in ROM for translate PS2 code to LCD code. To compare the first eight and last eight characters we stored all the PS2 codes for characters in the data registers and used if statement in our S_IDLE state but since the last register does not get updated at the comparison we checked if it was same as PS2 code. We did the same thing for reverse match using if statements. To check for top line or bottom line we used the pre-defined flag called LCD_line, when it was 1 we check top line when it was 0 we check bottom line. Finally, for red light blinking we created two new flags, red_LED and red_LED_blink. Flag red_LED would only be 1 when we find the match during our character comparison, this flag is then used in S_LCD_FINISH_CHANGE_LINE where we check the end of line and then use if statement to blink the red light or not blink the red light. The lower and upper case did not matter during comparison because the PS2 code for them was same, we just concatenated a 1 every time left shift was pressed and concatenated 0 every time right shift was pressed.