UCSD CSE140L Spring 2014

**LAB#3 Report**

Demonstration Date : 5 / 20 /14 Student CID\_\_\_\_\_\_\_\_\_\_\_268\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Name: \_\_\_\_\_\_\_Kieth\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Vo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

first M.I. Last

**TED Submission Date & Time :**

(FILLED BY Student BEFORE DEMO) (\*\*\* FILLED BY TUTOR/INSTRUCTOR \*\*\*)

**Self-test Report** Demo Reviewer

Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Working Not working **Demo** score **Report** score

**Part1**: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/**3** a)\_\_\_\_\_\_\_\_\_/**1**

**Part2**: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/**3** b) \_\_\_\_\_\_\_\_\_/**1**

**Part3**: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_/**3** c) \_\_\_\_\_\_\_\_\_/**2**

**Part4**: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_/**3** d)\_\_\_\_\_\_\_\_\_/**1**

**Part5**: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /**3**

**Subtotal**  **Subtotal**

\_\_\_\_\_\_\_\_/**15** \_\_\_\_\_\_\_\_\_\_\_\_\_/**5**

**TOTAL Score:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/20**

For part 1 I created variables to hold deposit and change. Then I just incremented those and displayed them on the hex displays using the mod and divide operations and my task function. Switches 0-4 adds their respective values and modifies the deposit and change variables accordingly. Switch 8 resets the deposit and change values. All of this is in an always block that activates whenever I press the key[1] button.

For part 2 I used if statements in an always block. If I get an error the state goes back to 1. Else normal operation continues. If Switch 9 is up then I display the number dispensed and when it is down it returns to displaying deposit and change again.

For part 3 I used the clock variable in an always block so whenever the deposit was greater than or equal to 35 I made the green LEDs blink. When there is an error case and the state goes back to 0000 then I turn it off. They run at half second cycles with 50% duty.

For part 4 I used if statements to check the conditions of multiple switches being in the up position when I push key[1]. For the consecutive dollars and credit cards I used a count variable in each section where I take care of incrementing the deposit and change for them respectively. The count variable will set an error when it gets to 2 because that would be consecutive inputs and count returns to 0 otherwise. Using the same method I set an if statement where if the deposit is 35 and the change is 0 and credit card input is used then it will display an error.

**B)**

module L3C268( // where yyy=your CID. For example, L3C079 if your CID=079

input [9:0] sw, // ten up-down switches, SW9 - SW0

input [3:0] key, // four pushbutton switches, KEY3 - KEY0

input clock, // 24MHz clock source on Altera DE1 board

output [9:0] ledr, // ten Red LEDs, LEDR9 - LEDR0

output [7:0] ledg, // eight Green LEDs, LEDG8 - LEDG0

output reg [6:0] hex0, hex1, hex2, hex3 // four 7-segment, HEX3 - HEX0

);

// State controller

reg[6:0] deposit = 0;

reg[6:0] change = 0;

reg[1:0] state = 0;

reg[1:0] errorCase = 0;

always @ (\*)

begin

if (state == 0)

begin hx3(0); hx2(2); hx1(6); hx0(8); end // CID/Initial State

if (state == 1)

begin hx3(0); hx2(0); hx1(0); hx0(0); end // Zero State

else if (~sw[9] && state == 2)

begin

if (errorCase) // Error State

begin hx3(14); hx2(19); hx1(19); hx0(18); end

else // Normal Operation

begin hx3(deposit/10); hx2(deposit%10); hx1(change/10); hx0(change%10); end

end

else if (sw[9]) // Report State

begin hx3(18); hx2(18); hx1(18); hx0(totalDispensed); end

end

// Key Operations

reg[1:0] consecutiveCredit = 0;

reg[1:0] consecutiveDollar = 0;

reg[1:0] creditInput = 0;

reg[1:0] reset = 0;

reg[6:0] totalDispensed = 0;

assign switch = sw[0] | sw[1] | sw[2] | sw[3] | sw[4] | sw[8];

always @ (negedge key[1])

begin

if (state == 1 && errorCase)

begin state = 2; errorCase = 0; deposit = 0; change = 0; consecutiveDollar = 0; consecutiveCredit = 0; end

if (state < 2)

begin state = state + 1; end

else if (errorCase)

begin state = 1; end

if (state == 2 & switch)

begin

if (((sw[0] & (sw[1] | sw[2] | sw[3] | sw[4] | sw[8])) |

(sw[1] & (sw[0] | sw[2] | sw[3] | sw[4] | sw[8])) |

(sw[2] & (sw[0] | sw[1] | sw[3] | sw[4] | sw[8])) |

(sw[3] & (sw[0] | sw[1] | sw[2] | sw[4] | sw[8])) |

(sw[4] & (sw[0] | sw[1] | sw[2] | sw[3] | sw[8])) |

(sw[8] & (sw[0] | sw[1] | sw[2] | sw[3] | sw[4])))) // More than one input error

begin errorCase = 1; end

else

begin

if (reset == 1)

begin

if (sw[4] & deposit == 35 & change == 0) // Credit card error when HEX[3:0] == 3500

begin errorCase = 1; end

else

begin

change = 0;

deposit = 0;

reset = 0;

end

end

if (sw[0]) // NICKEL

begin

deposit = deposit + 5;

consecutiveDollar = 0;

consecutiveCredit = 0;

end

else if (sw[1]) // DIME

begin

deposit = deposit + 10;

consecutiveDollar = 0;

consecutiveCredit = 0;

end

else if (sw[2]) // Quarter

begin

deposit = deposit + 25;

consecutiveDollar = 0;

consecutiveCredit = 0;

end

else if (sw[3]) // Dollar

begin

deposit = deposit + 100;

consecutiveDollar = consecutiveDollar + 1;

if (consecutiveDollar == 2) // Consecutive dollars error

begin errorCase = 1; end

consecutiveCredit = 0;

end

else if (sw[4]) // Credit Card

begin

deposit = deposit + 35;

consecutiveCredit = consecutiveCredit + 1;

if (consecutiveCredit == 2) // Consecutive credit card error

begin errorCase = 1; end

consecutiveDollar = 0;

end

else if (sw[8]) // Reset. Does not clear # dispensed.

begin

deposit = 0;

change = 0;

consecutiveDollar = 0;

consecutiveCredit = 0;

end

if (deposit >= 35)

begin

if (reset == 0)

begin

if (errorCase != 1)

begin

change = change + (deposit - 35);

deposit = 35;

end

if (totalDispensed == 15)

begin totalDispensed = 0; end

else if (errorCase != 1)

begin totalDispensed = totalDispensed + 1; end

reset = 1;

end

end

end

end

end

// Light Controller

reg[23:0] clockIndex;

reg [7:0] ledGreen;

reg [6:0] count;

assign ledg[7:0] = ledGreen;

always @(posedge clock)

begin

if (deposit >= 35) // Flash at 50% duty for .5 second cycles

begin

clockIndex = clockIndex + 1;

if (clockIndex == 6000000)

begin

clockIndex = 0;

count = count + 1;

if (count == 1)

ledGreen = 8'b11111111;

if (count == 2)

begin ledGreen = 8'b00000000; count = 0; end

if (state == 1)

begin ledGreen = 8'b00000000; count = 0; end

end

end

else

begin

ledGreen = 8'b00000000;

count = 0;

clockIndex = 0;

end

end

// Controls HEX0 Display

task hx0;

input [6:0] num;

begin

case(num)

0: hex0 = 7'b1000000; //0

1: hex0 = 7'b1111001; //1

2: hex0 = 7'b0100100; //2

3: hex0 = 7'b0110000; //3

4: hex0 = 7'b0011001; //4

5: hex0 = 7'b0010010; //5

6: hex0 = 7'b0000010; //6

7: hex0 = 7'b1111000; //7

8: hex0 = 7'b0000000; //8

9: hex0 = 7'b0011000; //9

10: hex0 = 7'b0001000; //A

11: hex0 = 7'b0000011; //b

12: hex0 = 7'b1000110; //C

13: hex0 = 7'b0100001; //d

14: hex0 = 7'b0000110; //E

15: hex0 = 7'b0001110; //F

16: hex0 = 7'b0001001; //H

17: hex0 = 7'b1000111; //L

18: hex0 = 7'b1111111; //OFF

19: hex0 = 7'b0101111; //r

endcase

end

endtask

task hx1;

input [6:0] num;

begin

case(num)

0: hex1 = 7'b1000000; //0

1: hex1 = 7'b1111001; //1

2: hex1 = 7'b0100100; //2

3: hex1 = 7'b0110000; //3

4: hex1 = 7'b0011001; //4

5: hex1 = 7'b0010010; //5

6: hex1 = 7'b0000010; //6

7: hex1 = 7'b1111000; //7

8: hex1 = 7'b0000000; //8

9: hex1 = 7'b0011000; //9

10: hex1 = 7'b0001000; //A

11: hex1 = 7'b0000011; //b

12: hex1 = 7'b1000110; //C

13: hex1 = 7'b0100001; //d

14: hex1 = 7'b0000110; //E

15: hex1 = 7'b0001110; //F

16: hex1 = 7'b0001001; //H

17: hex1 = 7'b1000111; //L

18: hex1 = 7'b1111111; //OFF

19: hex1 = 7'b0101111; //r

endcase

end

endtask

task hx2;

input [6:0] num;

begin

case(num)

0: hex2 = 7'b1000000; //0

1: hex2 = 7'b1111001; //1

2: hex2 = 7'b0100100; //2

3: hex2 = 7'b0110000; //3

4: hex2 = 7'b0011001; //4

5: hex2 = 7'b0010010; //5

6: hex2 = 7'b0000010; //6

7: hex2 = 7'b1111000; //7

8: hex2 = 7'b0000000; //8

9: hex2 = 7'b0011000; //9

10: hex2 = 7'b0001000; //A

11: hex2 = 7'b0000011; //b

12: hex2 = 7'b1000110; //C

13: hex2 = 7'b0100001; //d

14: hex2 = 7'b0000110; //E

15: hex2 = 7'b0001110; //F

16: hex2 = 7'b0001001; //H

17: hex2 = 7'b1000111; //L

18: hex2 = 7'b1111111; //OFF

19: hex2 = 7'b0101111; //r

endcase

end

endtask

task hx3;

input [6:0] num;

begin

case(num)

0: hex3 = 7'b1000000; //0

1: hex3 = 7'b1111001; //1

2: hex3 = 7'b0100100; //2

3: hex3 = 7'b0110000; //3

4: hex3 = 7'b0011001; //4

5: hex3 = 7'b0010010; //5

6: hex3 = 7'b0000010; //6

7: hex3 = 7'b1111000; //7

8: hex3 = 7'b0000000; //8

9: hex3 = 7'b0011000; //9

10: hex3 = 7'b0001000; //A

11: hex3 = 7'b0000011; //b

12: hex3 = 7'b1000110; //C

13: hex3 = 7'b0100001; //d

14: hex3 = 7'b0000110; //E

15: hex3 = 7'b0001110; //F

16: hex3 = 7'b0001001; //H

17: hex3 = 7'b1000111; //L

18: hex3 = 7'b1111111; //OFF

19: hex3 = 7'b0101111; //r

endcase

end

endtask

endmodule

**S3**

**C)**

**Sw[9] ~Sw[9]**

**Error Case**

**S0 S1 S2**

**Key[1] Key[1]**

**Sw[0] & Key[1],**

**Sw[1] & Key[1],**

**Sw[2] & Key[1],**

**Sw[3] & Key[1],**

**Sw[4] & Key[1],**

**Sw[8] & Key[1]**

**D)**

