UCSD CSE140L Spring 2014

**LAB#3 Report**

Demonstration Date : / /14 Student CID\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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**

1. Description:
   1. In the precondition, I have everything I use in all the other parts being initialized and set all the lights to be off. I have my CID displaying.
   2. In part one, I used key[1], sw[2:0] and key[1] was for entering money. The switches are for nickel, dime, and quarter input.
   3. In part two, I used sw[9] which is a counter for the number of times coin inputs reach 35 cents.
   4. In part three, I used sw[8] & sw[4] for credit card and reset inputs.
   5. In part four, I used sw[3] for one-dollar inputs.
   6. In part five, I used all HEX’s to display “Err” when an error has occurred.
2. Verilog code

`d`define BLANK 7'b1111111

`define ZERO 7'b1000000

`define ONE 7'b1111001

`define TWO 7'b0100100

`define THREE 7'b0110000

`define FOUR 7'b0011001

`define FIVE 7'b0010010

`define SIX 7'b0000010

`define SEVEN 7'b1111000

`define EIGHT 7'b0000000

`define NINE 7'b0011000

`define A 7'b0001000

`define b 7'b0000011

`define C 7'b1000110

`define d 7'b0100001

`define E 7'b0000110

`define F 7'b0001110

`define H 7'b0001001

`define L 7'b1000111

`define r 7'b0101111

module L3C580 // where 580 = CID

(

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

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

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

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

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

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

);

parameter zero = 3'b000, one = 3'b001, two = 3'b010, three = 3'b011, four = 3'b100, five = 3'b101, six = 3'b110, seven = 3'b111;

parameter none = 9'b000000000, nickel = 9'b000000001, dime = 9'b000000010, quarter = 9'b000000100, dollar = 9'b000001000, credit\_card = 9'b000010000, reset = 9'b100000000;

integer in;

reg[3:0] state, next\_state;

reg [6:0] h3, h2, h1, h0;

integer VM = 0;

reg[3:0] counter = 0;

integer err = 0;

integer cycle = 0;

integer rep = 0;

integer flag = 0;

integer clear = 0;

always @(posedge clock) begin

state = next\_state;

if (flag == 1) begin

h3 = `THREE;

h2 = `FIVE;

clear = 1;

end

if (sw[9] == 1 && err == 0 && rep == 0) begin

rep <= 1;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

hex3 <= `BLANK;

hex2 <= `BLANK;

hex1 <= `BLANK;

case (counter)

0: begin

hex0 <= `ZERO;

end

1: begin

hex0 <= `ONE;

end

2: begin

hex0 <= `TWO;

end

3: begin

hex0 <= `THREE;

end

4: begin

hex0 <= `FOUR;

end

5: begin

hex0 <= `FIVE;

end

6: begin

hex0 <= `SIX;

end

7: begin

hex0 <= `SEVEN;

end

8: begin

hex0 <= `EIGHT;

end

9: begin

hex0 <= `NINE;

end

10: begin

hex0 <= `A;

end

11: begin

hex0 <= `b;

end

12: begin

hex0 <= `C;

end

13: begin

hex0 <= `d;

end

14: begin

hex0 <= `E;

end

15: begin

hex0 <= `F;

end

endcase

end

else if (VM == 1 && sw[9] == 0) begin

rep = 0;

if (err == 0) begin

case(state)

zero: begin

hex3 <= `ZERO;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

cycle = 0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

one: begin

hex3 <= `ZERO;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

two: begin

hex3 <= `ONE;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

three: begin

hex3 <= `ONE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

four: begin

hex3 <= `TWO;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

five: begin

hex3 <= `TWO;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

six: begin

hex3 <= `THREE;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

seven: begin

cycle <= cycle + 1;

case (in)

none: begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

nickel: begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

end

dime: begin

if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `FIVE;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

quarter: begin

if (h3 == `ONE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

else if (h3 == `ONE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `FIVE;

end

else if (h3 == `TWO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ONE;

hex0 <= `ZERO;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ONE;

hex0 <= `FIVE;

end

else if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `TWO;

hex0 <= `ZERO;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

dollar: begin

if (h3 == `ZERO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SIX;

hex0 <= `FIVE;

end

else if (h3 == `ZERO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SEVEN;

hex0 <= `ZERO;

end

else if (h3 == `ONE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SEVEN;

hex0 <= `FIVE;

end

else if (h3 == `ONE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `ZERO;

end

else if (h3 == `TWO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `FIVE;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `NINE;

hex0 <= `ZERO;

end

else if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `NINE;

hex0 <= `FIVE;

end

else if (h3 == `THREE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SIX;

hex0 <= `FIVE;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

credit\_card: begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

default: begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

endcase

if (cycle == 6000000) begin

ledg[0] <= 1;

ledg[1] <= 1;

ledg[2] <= 1;

ledg[3] <= 1;

ledg[4] <= 1;

ledg[5] <= 1;

ledg[6] <= 1;

ledg[7] <= 1;

end

else if (cycle == 12000000) begin

ledg <= 0;

cycle <= 0;

end

ledr[9:0] <= 0;

end

default: begin

end

endcase

end

else if (err == 1) begin

cycle <= 0;

ledg <= 0;

hex3 <= `E;

hex2 <= `r;

hex1 <= `r;

hex0 <= `BLANK;

state <= zero;

end

end

else if (VM == 0 && sw[9] == 0) begin

hex3 <= `ZERO;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `ZERO;

end

end

always @(negedge key[1]) begin

if (clear == 1) begin

flag = 0;

end

if (VM == 0) begin

VM = 1;

next\_state = zero;

end

else if (VM == 1) begin

if (state != seven && err == 0) begin

case(sw[8:0])

none: begin

in = none;

end

nickel: begin

in = nickel;

if (state < seven) begin

if (state == six) begin

counter = counter + 1;

next\_state = seven;

end

next\_state = state + one;

end

end

dime: begin

in = dime;

if (state < six) begin

next\_state = state + two;

end

else begin

next\_state = seven;

counter = counter + 1;

end

end

quarter: begin

in = quarter;

if (state < three) begin

next\_state = state + five;

end

else begin

next\_state = seven;

counter = counter + 1;

end

end

dollar: begin

in = dollar;

next\_state = seven;

counter = counter + 1;

end

credit\_card: begin

in = credit\_card;

next\_state = seven;

end

reset: begin

in = reset;

next\_state = zero;

end

default: begin

err = 1;

next\_state = zero;

end

endcase

end

else if (state == seven && err == 0) begin

case(sw[8:0])

none: begin

in = none;

end

nickel: begin

in = nickel;

next\_state = one;

end

dime: begin

in = dime;

next\_state = two;

end

quarter: begin

in = quarter;

next\_state = five;

end

dollar: begin

in = dollar;

if (hex1 != `ZERO && hex1 != `ONE && hex1 != `TWO) begin

err = 1;

next\_state = zero;

end

else if (hex1 == `ZERO || hex1 == `ONE || hex1 == `TWO) begin

next\_state = seven;

flag = 1;

err = 0;

end

end

credit\_card: begin

in = credit\_card;

if (hex1 == `ZERO && hex0 == `ZERO) begin

next\_state = zero;

err = 1;

end

else if (hex1 != `ZERO && hex0 != `ZERO) begin

next\_state = seven;

flag = 1;

err = 0;

end

end

reset: begin

next\_state = zero;

end

default: begin

err = 1;

next\_state = zero;

end

endcase

end

else if (err == 1) begin

next\_state = zero;

err = 0;

end

end

end

endmodule

efine BLANK 7'b1111111

`define ZERO 7'b1000000

`define ONE 7'b1111001

`define TWO 7'b0100100

`define THREE 7'b0110000

`define FOUR 7'b0011001

`define FIVE 7'b0010010

`define SIX 7'b0000010

`define SEVEN 7'b1111000

`define EIGHT 7'b0000000

`define NINE 7'b0011000

`define A 7'b0001000

`define b 7'b0000011

`define C 7'b1000110

`define d 7'b0100001

`define E 7'b0000110

`define F 7'b0001110

`define H 7'b0001001

`define L 7'b1000111

`define r 7'b0101111

module L3C580 // where 580 = CID

(

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

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

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

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

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

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

);

parameter zero = 3'b000, one = 3'b001, two = 3'b010, three = 3'b011, four = 3'b100, five = 3'b101, six = 3'b110, seven = 3'b111;

parameter none = 9'b000000000, nickel = 9'b000000001, dime = 9'b000000010, quarter = 9'b000000100, dollar = 9'b000001000, credit\_card = 9'b000010000, reset = 9'b100000000;

integer in;

reg[3:0] state, next\_state;

reg [6:0] h3, h2, h1, h0;

integer VM = 0;

reg[3:0] counter = 0;

integer err = 0;

integer cycle = 0;

integer rep = 0;

integer flag = 0;

integer clear = 0;

always @(posedge clock) begin

state = next\_state;

if (flag == 1) begin

h3 = `THREE;

h2 = `FIVE;

clear = 1;

end

if (sw[9] == 1 && err == 0 && rep == 0) begin

rep <= 1;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

hex3 <= `BLANK;

hex2 <= `BLANK;

hex1 <= `BLANK;

case (counter)

0: begin

hex0 <= `ZERO;

end

1: begin

hex0 <= `ONE;

end

2: begin

hex0 <= `TWO;

end

3: begin

hex0 <= `THREE;

end

4: begin

hex0 <= `FOUR;

end

5: begin

hex0 <= `FIVE;

end

6: begin

hex0 <= `SIX;

end

7: begin

hex0 <= `SEVEN;

end

8: begin

hex0 <= `EIGHT;

end

9: begin

hex0 <= `NINE;

end

10: begin

hex0 <= `A;

end

11: begin

hex0 <= `b;

end

12: begin

hex0 <= `C;

end

13: begin

hex0 <= `d;

end

14: begin

hex0 <= `E;

end

15: begin

hex0 <= `F;

end

endcase

end

else if (VM == 1 && sw[9] == 0) begin

rep = 0;

if (err == 0) begin

case(state)

zero: begin

hex3 <= `ZERO;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

cycle = 0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

one: begin

hex3 <= `ZERO;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

two: begin

hex3 <= `ONE;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

three: begin

hex3 <= `ONE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

four: begin

hex3 <= `TWO;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

five: begin

hex3 <= `TWO;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

six: begin

hex3 <= `THREE;

hex2 <= `ZERO;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

ledg[7:0] <= 0;

ledr[9:0] <= 0;

end

seven: begin

cycle <= cycle + 1;

case (in)

none: begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

nickel: begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

end

dime: begin

if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `FIVE;

h3 <= hex3;

h2 <= hex2;

h1 <= hex1;

h0 <= hex0;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

quarter: begin

if (h3 == `ONE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

else if (h3 == `ONE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `FIVE;

end

else if (h3 == `TWO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ONE;

hex0 <= `ZERO;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ONE;

hex0 <= `FIVE;

end

else if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `TWO;

hex0 <= `ZERO;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

dollar: begin

if (h3 == `ZERO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SIX;

hex0 <= `FIVE;

end

else if (h3 == `ZERO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SEVEN;

hex0 <= `ZERO;

end

else if (h3 == `ONE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SEVEN;

hex0 <= `FIVE;

end

else if (h3 == `ONE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `ZERO;

end

else if (h3 == `TWO && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `FIVE;

end

else if (h3 == `TWO && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `NINE;

hex0 <= `ZERO;

end

else if (h3 == `THREE && h2 == `ZERO) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `NINE;

hex0 <= `FIVE;

end

else if (h3 == `THREE && h2 == `FIVE) begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `SIX;

hex0 <= `FIVE;

end

else begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

end

credit\_card: begin

hex3 <= `THREE;

hex2 <= `FIVE;

hex1 <= `ZERO;

hex0 <= `ZERO;

end

default: begin

hex3 <= h3;

hex2 <= h2;

hex1 <= h1;

hex0 <= h0;

end

endcase

if (cycle == 6000000) begin

ledg[0] <= 1;

ledg[1] <= 1;

ledg[2] <= 1;

ledg[3] <= 1;

ledg[4] <= 1;

ledg[5] <= 1;

ledg[6] <= 1;

ledg[7] <= 1;

end

else if (cycle == 12000000) begin

ledg <= 0;

cycle <= 0;

end

ledr[9:0] <= 0;

end

default: begin

end

endcase

end

else if (err == 1) begin

cycle <= 0;

ledg <= 0;

hex3 <= `E;

hex2 <= `r;

hex1 <= `r;

hex0 <= `BLANK;

state <= zero;

end

end

else if (VM == 0 && sw[9] == 0) begin

hex3 <= `ZERO;

hex2 <= `FIVE;

hex1 <= `EIGHT;

hex0 <= `ZERO;

end

end

always @(negedge key[1]) begin

if (clear == 1) begin

flag = 0;

end

if (VM == 0) begin

VM = 1;

next\_state = zero;

end

else if (VM == 1) begin

if (state != seven && err == 0) begin

case(sw[8:0])

none: begin

in = none;

end

nickel: begin

in = nickel;

if (state < seven) begin

if (state == six) begin

counter = counter + 1;

next\_state = seven;

end

next\_state = state + one;

end

end

dime: begin

in = dime;

if (state < six) begin

next\_state = state + two;

end

else begin

next\_state = seven;

counter = counter + 1;

end

end

quarter: begin

in = quarter;

if (state < three) begin

next\_state = state + five;

end

else begin

next\_state = seven;

counter = counter + 1;

end

end

dollar: begin

in = dollar;

next\_state = seven;

counter = counter + 1;

end

credit\_card: begin

in = credit\_card;

next\_state = seven;

end

reset: begin

in = reset;

next\_state = zero;

end

default: begin

err = 1;

next\_state = zero;

end

endcase

end

else if (state == seven && err == 0) begin

case(sw[8:0])

none: begin

in = none;

end

nickel: begin

in = nickel;

next\_state = one;

end

dime: begin

in = dime;

next\_state = two;

end

quarter: begin

in = quarter;

next\_state = five;

end

dollar: begin

in = dollar;

if (hex1 != `ZERO && hex1 != `ONE && hex1 != `TWO) begin

err = 1;

next\_state = zero;

end

else if (hex1 == `ZERO || hex1 == `ONE || hex1 == `TWO) begin

next\_state = seven;

flag = 1;

err = 0;

end

end

credit\_card: begin

in = credit\_card;

if (hex1 == `ZERO && hex0 == `ZERO) begin

next\_state = zero;

err = 1;

end

else if (hex1 != `ZERO && hex0 != `ZERO) begin

next\_state = seven;

flag = 1;

err = 0;

end

end

reset: begin

next\_state = zero;

end

default: begin

err = 1;

next\_state = zero;

end

endcase

end

else if (err == 1) begin

next\_state = zero;

err = 0;

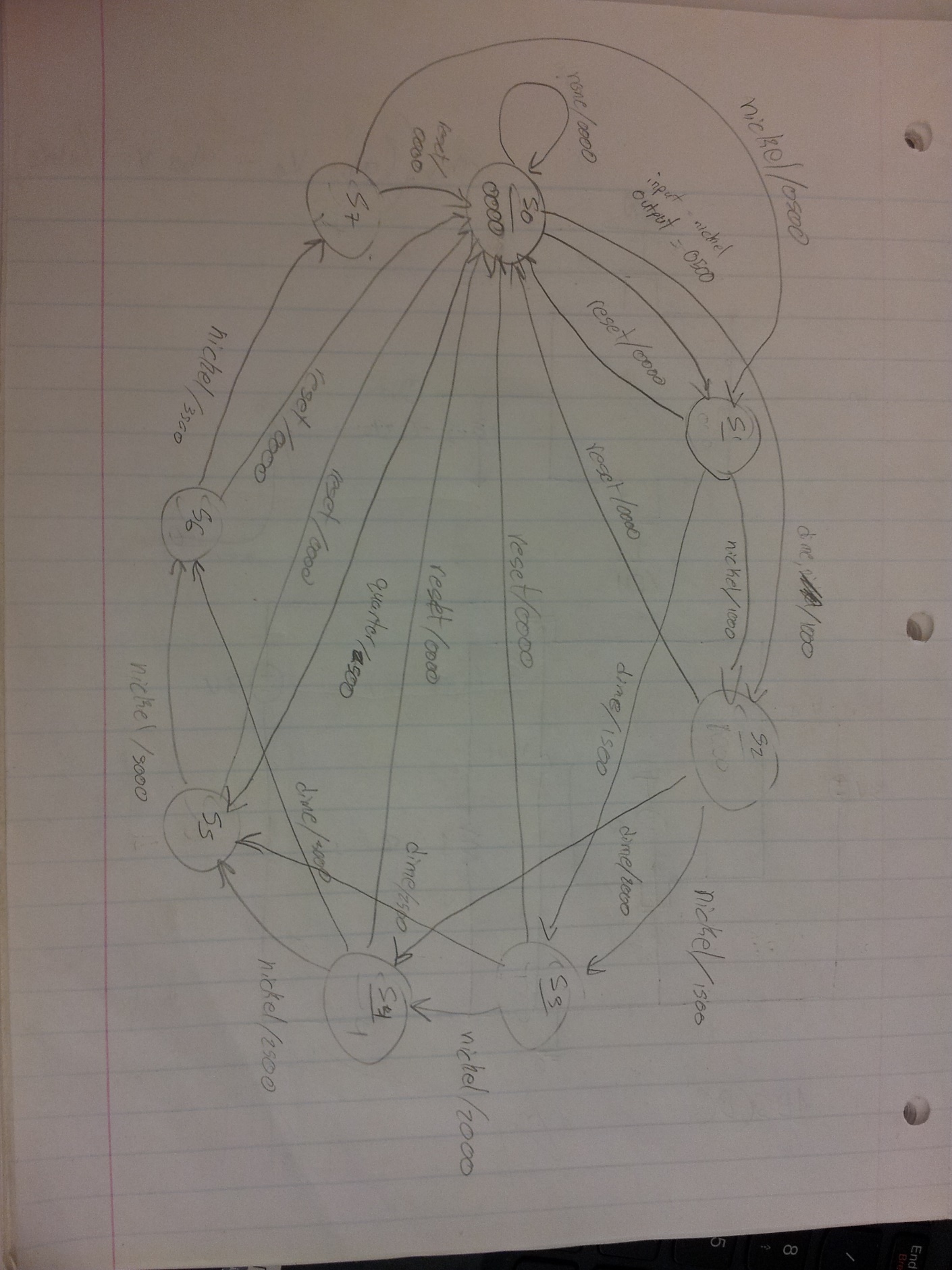
end

end

end

endmodule

1. State Diagram



1. Compilation Report

