UCSD CSE140L Spring 2014

**LAB#2 Report**

Demonstration Date : 05 / 01 /14 Student CID\_\_\_\_\_\_\_\_\_\_580\_\_\_\_\_\_\_\_\_\_

Student Name: \_\_\_\_\_Adrian\_\_Jimenez\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

first M.I. Last

**TED Submission Date & Time : May 1, 2014 10:35am**

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

**Self-test Report** Demo Reviewer

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

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

**Part1**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/**2** a)\_\_\_\_\_\_\_\_\_/**1**

**Part2**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/**2** b) \_\_\_\_\_\_\_\_\_/**1**

**Part3**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_/**3** c) \_\_\_\_\_\_\_\_\_/**1**

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

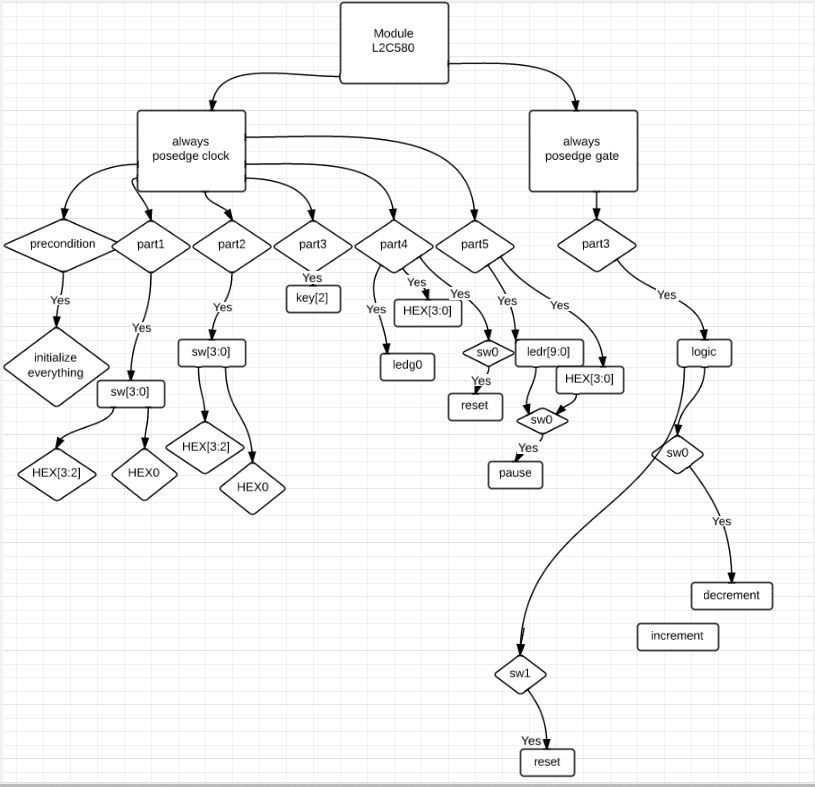
**Part5**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /**5** e) \_\_\_\_\_\_\_\_\_/**1**

**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 on HEX2-HEX0 and HEX3 being blank.
   2. In part one, I used sw[3:0] and HEX[3:0]. HEX[3:2] displays the decimal number of sw[3:0] represented as binary. HEX1 is blank as it is used as a space and HEX 0 displays the Hex number of sw[3:0].
   3. In part two, I used sw[4:0] and HEX[3:0]. Sw[4:3] represented the first operand in binary and sw[2:1] as the second operand in binary and sw0 is the operator selector, 0 for addition and 1 for multiplication. HEX3 displays the decimal value of the first operand and HEX2 displays the decimal value of the second operand. HEX1 is blank to represent a space and HEX0 is the decimal value of the result.
   4. In part three, I used key[2], sw[1:0] and HEX2. Key[2] is used as the counter, sw0 is used for resetting the counter back to zero and sw1 causes the counter to increment if it’s down and decrements if the switch is up. HEX2 displays the counter in hex and all other HEX’s are blank.
   5. In part four, I used sw0, HEX[3:0] and ledg0. Sw0 is used for resetting the clock back to zero, HEX[3:0] displays the clock in modulo-3 operation and ledg0 starts blinking every second with 50% duty cycle.
   6. In part five, I used sw0, ledr[9:0] and HEX[3:0]. Ledr[9:0] starting from ledr0 lights one led at a time moving back and forth with a duration of a second for the round trip. HEX[3:0] displays a moving message that repeats and is synchronized with the movement of the lights. The message moves one letter to the left every time ledr9 is on. And sw0 is used to pause if the switch is up and resumes if the switch is down.
2. Flowchart



1. Verilog Code

`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

module L2C580 // 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 [7:0]ledg, // eight Green LEDs, LEDG8 - LEDG0

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

);

integer counter;

integer cycle = 0;

integer cycle1 = 0;

integer real\_time = 0;

integer real\_time1 = 0;

integer real\_time2 = 0;

integer real\_time3 = 0;

integer dir = 0;

integer mes = 0;

assign gate = ~key[2] | sw[0] | ~sw[7];

reg light;

assign ledg[0] = light;

always @(posedge gate) begin

   if (sw[7] == 0) begin

 counter = 0;

end

if (sw[0] == 0) begin

if (sw[1] == 0 && sw[7] == 1) begin

 counter = counter + 1;

 if (counter > 15) begin

counter = 0;

 end

end

else if (sw[1] == 1 && sw[7] == 1) begin

 counter = counter - 1;

 if (counter < 0) begin

counter = 15;

 end

end

end

else begin

counter = 0;

end

end

always @(posedge clock) begin

if (sw[9:5]==5'b00000) begin // all sw are in DOWN position

 //Initial state (No Part is selected)

 hex3 = `BLANK;

 hex2 = `FIVE;

 hex1 = `EIGHT;

 hex0 = `ZERO;

 cycle = 0;

 light = 0;

 ledr[9:0] = 0;

 real\_time = 0;

 real\_time1 = 0;

 real\_time2 = 0;

 real\_time3 = 0;

 mes = 0;

end

else if (sw[9:5]==5'b10000) begin // only sw[9] is in UP position

 //Only Part1 is selected

 hex1 = `BLANK;

 if (sw[3:0] == 4'b0000) begin

hex3 = `ZERO;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[3:0] == 4'b0001) begin

hex3 = `ZERO;

hex2 = `ONE;

hex0 = `ONE;

 end

 else if (sw[3:0] == 4'b0010) begin

hex3 = `ZERO;

hex2 = `TWO;

hex0 = `TWO;

 end

 else if (sw[3:0] == 4'b0011) begin

hex3 = `ZERO;

hex2 = `THREE;

hex0 = `THREE;

 end

 else if (sw[3:0] == 4'b0100) begin

hex3 = `ZERO;

hex2 = `FOUR;

hex0 = `FOUR;

 end

 else if (sw[3:0] == 4'b0101) begin

hex3 = `ZERO;

hex2 = `FIVE;

hex0 = `FIVE;

 end

 else if (sw[3:0] == 4'b0110) begin

hex3 = `ZERO;

hex2 = `SIX;

hex0 = `SIX;

 end

 else if (sw[3:0] == 4'b0111) begin

hex3 = `ZERO;

hex2 = `SEVEN;

hex0 = `SEVEN;

 end

 else if (sw[3:0] == 4'b1000) begin

hex3 = `ZERO;

hex2 = `EIGHT;

hex0 = `EIGHT;

 end

 else if (sw[3:0] == 4'b1001) begin

hex3 = `ZERO;

hex2 = `NINE;

hex0 = `NINE;

 end

 else if (sw[3:0] == 4'b1010) begin

hex3 = `ONE;

hex2 = `ZERO;

hex0 = `A;

 end

 else if (sw[3:0] == 4'b1011) begin

hex3 = `ONE;

hex2 = `ONE;

hex0 = `b;

 end

 else if (sw[3:0] == 4'b1100) begin

hex3 = `ONE;

hex2 = `TWO;

hex0 = `C;

 end

 else if (sw[3:0] == 4'b1101) begin

hex3 = `ONE;

hex2 = `THREE;

hex0 = `d;

 end

 else if (sw[3:0] == 4'b1110) begin

hex3 = `ONE;

hex2 = `FOUR;

hex0 = `E;

 end

 else if (sw[3:0] == 4'b1111) begin

hex3 = `ONE;

hex2 = `FIVE;

hex0 = `F;

 end

end

else if (sw[9:5]==5'b01000) begin // only sw[8] is in UP position

 //Only Part2 is selected

 hex1 = `BLANK;

 if (sw[4:0] == 5'b00000) begin

hex3 = `ZERO;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b00010) begin

hex3 = `ZERO;

hex2 = `ONE;

hex0 = `ONE;

 end

 else if (sw[4:0] == 5'b00100) begin

hex3 = `ZERO;

hex2 = `TWO;

hex0 = `TWO;

 end

 else if (sw[4:0] == 5'b00110) begin

hex3 = `ZERO;

hex2 = `THREE;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b01000) begin

hex3 = `ONE;

hex2 = `ZERO;

hex0 = `ONE;

 end

 else if (sw[4:0] == 5'b01010) begin

hex3 = `ONE;

hex2 = `ONE;

hex0 = `TWO;

 end

 else if (sw[4:0] == 5'b01100) begin

hex3 = `ONE;

hex2 = `TWO;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b01110) begin

hex3 = `ONE;

hex2 = `THREE;

hex0 = `FOUR;

 end

 else if (sw[4:0] == 5'b10000) begin

hex3 = `TWO;

hex2 = `ZERO;

hex0 = `TWO;

 end

 else if (sw[4:0] == 5'b10010) begin

hex3 = `TWO;

hex2 = `ONE;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b10100) begin

hex3 = `TWO;

hex2 = `TWO;

hex0 = `FOUR;

 end

 else if (sw[4:0] == 5'b10110) begin

hex3 = `TWO;

hex2 = `THREE;

hex0 = `FIVE;

 end

 else if (sw[4:0] == 5'b11000) begin

hex3 = `THREE;

hex2 = `ZERO;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b11010) begin

hex3 = `THREE;

hex2 = `ONE;

hex0 = `FOUR;

 end

 else if (sw[4:0] == 5'b11100) begin

hex3 = `THREE;

hex2 = `TWO;

hex0 = `FIVE;

 end

 else if (sw[4:0] == 5'b11110) begin

hex3 = `THREE;

hex2 = `THREE;

hex0 = `SIX;

 end

 else if (sw[4:0] == 5'b00001) begin

hex3 = `ZERO;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b00011) begin

hex3 = `ZERO;

hex2 = `ONE;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b00101) begin

hex3 = `ZERO;

hex2 = `TWO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b00111) begin

hex3 = `ZERO;

hex2 = `THREE;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b01001) begin

hex3 = `ONE;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b01011) begin

hex3 = `ONE;

hex2 = `ONE;

hex0 = `ONE;

 end

 else if (sw[4:0] == 5'b01101) begin

hex3 = `ONE;

hex2 = `TWO;

hex0 = `TWO;

 end

 else if (sw[4:0] == 5'b01111) begin

hex3 = `ONE;

hex2 = `THREE;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b10001) begin

hex3 = `TWO;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b10011) begin

hex3 = `TWO;

hex2 = `ONE;

hex0 = `TWO;

 end

 else if (sw[4:0] == 5'b10101) begin

hex3 = `TWO;

hex2 = `TWO;

hex0 = `FOUR;

 end

 else if (sw[4:0] == 5'b10111) begin

hex3 = `TWO;

hex2 = `THREE;

hex0 = `SIX;

 end

 else if (sw[4:0] == 5'b11001) begin

hex3 = `THREE;

hex2 = `ZERO;

hex0 = `ZERO;

 end

 else if (sw[4:0] == 5'b11011) begin

hex3 = `THREE;

hex2 = `ONE;

hex0 = `THREE;

 end

 else if (sw[4:0] == 5'b11101) begin

hex3 = `THREE;

hex2 = `TWO;

hex0 = `SIX;

 end

 else if (sw[4:0] == 5'b11111) begin

hex3 = `THREE;

hex2 = `THREE;

hex0 = `NINE;

 end

end

else if (sw[9:5]==5'b00100) begin // only sw[7] is in UP position

 //Only Part3 is selected

 hex3 = `BLANK;

 hex1 = `BLANK;

 hex0 = `BLANK;

 if (counter == 0) begin

   hex2 = `ZERO;

 end

 else if (counter == 1) begin

   hex2 = `ONE;

 end

 else if (counter == 2) begin

   hex2 = `TWO;

 end

 else if (counter == 3) begin

   hex2 = `THREE;

 end

 else if (counter == 4) begin

   hex2 = `FOUR;

 end

 else if (counter == 5) begin

   hex2 = `FIVE;

 end

 else if (counter == 6) begin

   hex2 = `SIX;

 end

 else if (counter == 7) begin

   hex2 = `SEVEN;

 end

 else if (counter == 8) begin

   hex2 = `EIGHT;

 end

 else if (counter == 9) begin

   hex2 = `NINE;

 end

 else if (counter == 10) begin

   hex2 = `A;

 end

 else if (counter == 11) begin

   hex2 = `b;

 end

 else if (counter == 12) begin

   hex2 = `C;

 end

 else if (counter == 13) begin

   hex2 = `d;

 end

 else if (counter == 14) begin

   hex2 = `E;

 end

 else if (counter == 15) begin

   hex2 = `F;

 end

end

else if (sw[9:5]==5'b00010) begin // only sw[6] is in UP position

 //Only Part4 is selected

 cycle = cycle + 1;

 if (cycle == 24000000) begin

real\_time = real\_time + 1;

cycle = 0;

 end

 if (cycle == 0) begin

   light = 0;

 end

 else if (cycle == 12000000) begin

   light = 1;

 end

 if (real\_time == 3) begin

   real\_time = 0;

   real\_time1 = real\_time1 + 1;

 end

 if (real\_time1 == 3) begin

   real\_time1 = 0;

   real\_time2 = real\_time2 + 1;

 end

 if (real\_time2 == 3) begin

   real\_time2 = 0;

   real\_time3 = real\_time3 + 1;

 end

 if (real\_time3 == 3) begin

   real\_time3 = 0;

 end

 if (real\_time % 3 == 0) begin

   hex0 = `ZERO;

 end

 else if (real\_time % 3 == 1) begin

   hex0 = `ONE;

 end

 else if (real\_time % 3 == 2) begin

   hex0 = `TWO;

 end

 if (real\_time1 % 3 == 0) begin

   hex1 = `ZERO;

 end

 else if (real\_time1 % 3 == 1) begin

   hex1 = `ONE;

 end

 else if (real\_time1 % 3 == 2) begin

   hex1 = `TWO;

 end

 if (real\_time2 % 3 == 0) begin

   hex2 = `ZERO;

 end

 else if (real\_time2 % 3 == 1) begin

   hex2 = `ONE;

 end

 else if (real\_time2 % 3 == 2) begin

   hex2 = `TWO;

 end

 if (real\_time3 % 3 == 0) begin

   hex3 = `ZERO;

 end

 else if (real\_time3 % 3 == 1) begin

   hex3 = `ONE;

 end

 else if (real\_time3 % 3 == 2) begin

   hex3 = `TWO;

 end

 if (sw[0] == 1) begin

   real\_time = 0;

   real\_time1 = 0;

   real\_time2 = 0;

   real\_time3 = 0;

   cycle = 0;

   light = 0;

 end

end

else if (sw[9:5]==5'b00001) begin // only sw[5] is in UP position

 //Only Part5 is selected

 if (sw[0] == 0) begin

     if (mes % 19 == 0) begin

hex0 = `BLANK;

hex1 = `BLANK;

hex2 = `BLANK;

hex3 = `BLANK;

 end

     else if (mes % 19 == 1) begin

       hex0 = `H;

     end

     else if (mes % 19 == 2) begin

       hex0 = `E;

       hex1 = `H;

     end

     else if (mes % 19 == 3) begin

       hex0 = `L;

       hex1 = `E;

       hex2 = `H;

     end

     else if (mes % 19 == 4) begin

       hex0 = `L;

       hex1 = `L;

       hex2 = `E;

       hex3 = `H;

     end

     else if (mes % 19 == 5) begin

       hex0 = `ZERO;

       hex1 = `L;

       hex2 = `L;

       hex3 = `E;

     end

     else if (mes % 19 == 6) begin

       hex0 = `BLANK;

       hex1 = `ZERO;

       hex2 = `L;

       hex3 = `L;

     end

     else if (mes % 19 == 7) begin

       hex0 = `BLANK;

       hex1 = `BLANK;

       hex2 = `ZERO;

       hex3 = `L;

     end

     else if (mes % 19 == 8) begin

       hex0 = `C;

       hex1 = `BLANK;

       hex2 = `BLANK;

       hex3 = `ZERO;

     end

     else if (mes % 19 == 9) begin

       hex0 = `ONE;

       hex1 = `C;

       hex2 = `BLANK;

       hex3 = `BLANK;

     end

     else if (mes % 19 == 10) begin

       hex0 = `d;

       hex1 = `ONE;

       hex2 = `C;

       hex3 = `BLANK;

     end

     else if (mes % 19 == 11) begin

       hex0 = `BLANK;

       hex1 = `d;

       hex2 = `ONE;

       hex3 = `C;

     end

     else if (mes % 19 == 12) begin

       hex0 = `FIVE;

       hex1 = `BLANK;

       hex2 = `d;

       hex3 = `ONE;

     end

     else if (mes % 19 == 13) begin

       hex0 = `EIGHT;

       hex1 = `FIVE;

       hex2 = `BLANK;

       hex3 = `d;

     end

     else if (mes % 19 == 14) begin

       hex0 = `ZERO;

       hex1 = `EIGHT;

       hex2 = `FIVE;

       hex3 = `BLANK;

     end

     else if (mes % 19 == 15) begin

       hex0 = `BLANK;

       hex1 = `ZERO;

       hex2 = `EIGHT;

       hex3 = `FIVE;

     end

     else if (mes % 19 == 16) begin

       hex0 = `BLANK;

       hex1 = `BLANK;

       hex2 = `ZERO;

       hex3 = `EIGHT;

     end

     else if (mes % 19 == 17) begin

       hex0 = `BLANK;

       hex1 = `BLANK;

       hex2 = `BLANK;

       hex3 = `ZERO;

     end

     else if (mes % 19 == 18) begin

       hex0 = `BLANK;

       hex1 = `BLANK;

       hex2 = `BLANK;

       hex3 = `BLANK;

     end

 cycle1 = cycle1 + 1;

 if(ledr[9:0] == 0) begin

       ledr[0] = 1;

     end

 if (cycle1 % (24000000/18) == 0 && (ledr[0] == 1)) begin

ledr[1] = 1;

ledr[0] = 0;

dir = 0;

 end

 else if (cycle1 % (24000000/18) == 0 && ledr[1] == 1) begin

       if (dir == 0) begin

ledr[2] = 1;

ledr[1] = 0;

end

else begin

           ledr[0] = 1;

           ledr[1] = 0;

       end

     end

     else if (cycle1 % (24000000/18) == 0 && ledr[2] == 1) begin

   if (dir == 0) begin

ledr[3] = 1;

ledr[2] = 0;

end

else begin

       ledr[1] = 1;

       ledr[2] = 0;

   end

 end

 else if (cycle1 % (24000000/18) == 0 && ledr[3] == 1) begin

       if (dir == 0) begin

ledr[4] = 1;

ledr[3] = 0;

end

else begin

           ledr[2] = 1;

           ledr[3] = 0;

       end

     end

     else if (cycle1 % (24000000/18) == 0 && ledr[4] == 1) begin

   if (dir == 0) begin

ledr[5] = 1;

ledr[4] = 0;

end

else begin

       ledr[3] = 1;

       ledr[4] = 0;

   end

 end

 else if (cycle1 % (24000000/18) == 0 && ledr[5] == 1) begin

       if (dir == 0) begin

ledr[6] = 1;

ledr[5] = 0;

end

else begin

ledr[4] = 1;

ledr[5] = 0;

end

     end

     else if (cycle1 % (24000000/18) == 0 && ledr[6] == 1) begin

   if (dir == 0) begin

ledr[7] = 1;

ledr[6] = 0;

end

else begin

ledr[5] = 1;

ledr[6] = 0;

end

 end

 else if (cycle1 % (24000000/18) == 0 && ledr[7] == 1) begin

       if (dir == 0) begin

ledr[8] = 1;

ledr[7] = 0;

end

else begin

ledr[6] = 1;

ledr[7] = 0;

end

     end

     else if (cycle1 % (24000000/18) == 0 && ledr[8] == 1) begin

   if (dir == 0) begin

ledr[9] = 1;

ledr[8] = 0;

mes = mes + 1;

end

else begin

ledr[7] = 1;

ledr[8] = 0;

end

 end

 else if (cycle1 % (24000000/18) == 0 && ledr[9] == 1) begin

ledr[8] = 1;

ledr[9] = 0;

dir = 1;

     end

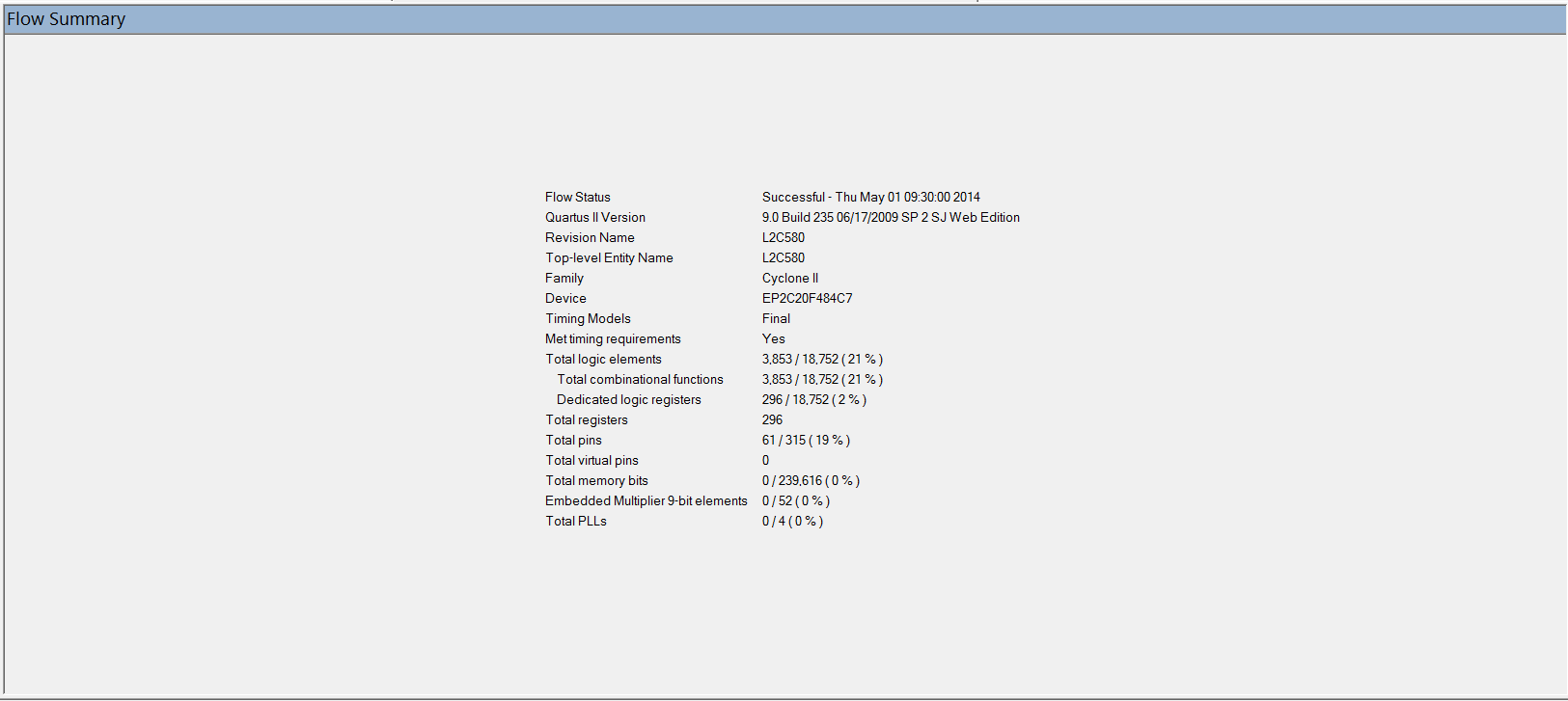
  end

end

 end

endmodule

1. Flow Summary



1. Timing Analyzer Summary

