UCSD CSE140L Spring 2014

# LAB#2 Report

Demonstration Date : 4 / 30 /14 Student CID____________268__________

Student Name: _______Kieth__________________________________Vo_________________

First M.I. Last

**TED Submission Date & Time :**

(FILLED BY Student BEFORE DEMO)

**Self-test Report**

| | Working | Not working |
|---|---|---|
| **Part1**: | _________ | ________ |
| **Part2**: | _________ | _________ |
| **Part3**: | _________ | _________ |
| **Part4**: | _________ | _________ |
| **Part5**: | _________ | ________ |

(*** FILLED BY TUTOR/INSTRUCTOR ***)

Demo Reviewer Name : ____________________________

| **Demo** score | **Report** score |
|---|---|
| ________/**2** | a)________/**1** |
| ________/**2** | b) ________/**1** |
| _________/**3** | c) ________/**1** |
| _________/**3** | d)________/**1** |
| ________ /**5** | e) ________/**1** |
| **Subtotal** | **Subtotal** |
| _______/**15** | ____________/**5** |

**TOTAL Score:** _______________/**20**

**A)**

**1.**

For part 1 I created a task that took a number as an input which was used in a case statement. Different cases set the HEX displays to different numbers or letters. Then I used this function to set the HEX[3:0] displays using another case statement which took the input of the switches 3 to 0. The switches create a binary number of four bits which I use in the case statement as a decimal number and display the corresponding output.

**2.**

For part 2 I used the same task functions to output the numbers and letters in the HEX displays. Then I used two if statements to determine if switch 0 was on or off to decide whether the output should be addition or multiplication. Inside the if conditions I used case statements to determine the output. This time the switches 4 to 1 provided the binary number which I converted to decimal and displayed the corresponding output.

**3.**

For part 3 I created a counter variable that activated whenever KEY[2] was pressed. The counter variable kept incrementing or decrementing one by one until it got to 16 or 0 and then the counter would reset to either 0 or 16 respectively. I used if statements for SW[1] which controlled whether the counter variable counts up or down. The HEX output was displayed using the same task functions in part 1 and 2.

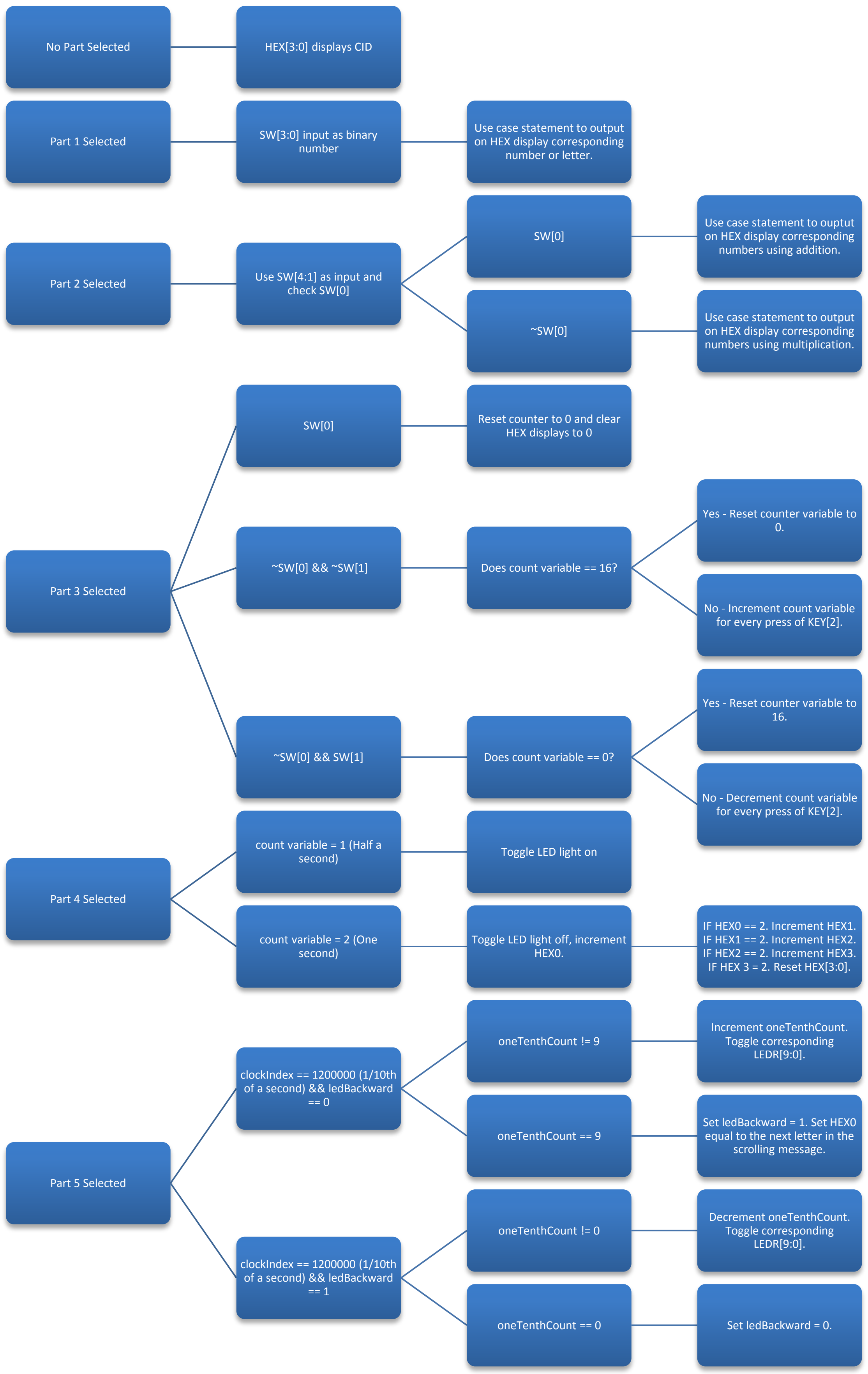
**4.**

For part 4 I created a clock index variable which incremented with the positive edge of the 24 MHz clock. When the value of the clock index got to 12 million I had a count variable which increased by 1. When the count variable got to 1 that meant that half a second passed so a variable ledToggleGreen was set to 1 which turned on the green LED light because ledg[0] was assigned to this variable. When count got to 2 that meant 1 second passed so the led variable is set to 0 and the LED light is turned off. Also at this time a variable for the HEX'S are incremented and checked. The HEX's numbers are incremented using modulus 3 operation. The HEX's are displayed using the same task functions in part 1 2 and 3.

**5.**

For part 5 I used another clock index variable which worked similarly to the clock index variable in part 4. The index this time though only got to 1.2 million because that is 1/10th of a second. Every 1/10th of a second a different LED would be switched on. This was controlled by a variable called oneTenthCount which incremented every 1/10th of a second. This variable was then used to tell the variable ledToggleRed which light to turn on or off. When the light got to the last LED which is LEDR[9], the variable oneTenthCount was then decremented using if statements to check whether a variable ledBackward, which controlled whether oneTenthCount was incrementing or decrementing, was on. Doing this would create the rolling LED lights which looked like a red ball bouncing side to side. Then for the scrolling message I created a case function which changed the output of HEX0. Every time the LED light hit the last one which was LEDR[9] a variable called message would be changed and a separate task function took the message variable and outputted HEX0's number or letter. At the same time there were trail variables which were created to hold the old values or HEX[3:0]. Then I would set HEX3 to the old value of HEX2 and HEX2 to the old value of HEX1 and HEX1 to the old value of HEX0. This gave us the scrolling message on the HEX displays. The HEX displays' outputs were created using the same task functions in part 1, 2, 3, and 4.

**B)**


No Part Selected
HEX[3:0] displays CID
Part 1 Selected
SW[3:0] input as binary number
Use case statement to output on HEX display corresponding number or letter.
Part 2 Selected
Use SW[4:1] as input and check SW[0]
SW[0]
Use case statement to ouptut on HEX display corresponding numbers using addition.
~SW[0]
Use case statement to output on HEX display corresponding numbers using multiplication.
Part 3 Selected
SW[0]
Reset counter to 0 and clear HEX displays to 0
~SW[0] && ~SW[1]
Does count variable == 16?
Yes - Reset counter variable to 0.
No - Increment count variable for every press of KEY[2].
~SW[0] && SW[1]
Does count variable == 0?
Yes - Reset counter variable to 16.
No - Decrement count variable for every press of KEY[2].
Part 4 Selected
count variable = 1 (Half a second)
Toggle LED light on
count variable = 2 (One second)
Toggle LED light off, increment HEX0.
IF HEX0 == 2. Increment HEX1. IF HEX1 == 2. Increment HEX2. IF HEX2 == 2. Increment HEX3. IF HEX 3 = 2. Reset HEX[3:0].
Part 5 Selected
clockIndex == 1200000 (1/10th of a second) && ledBackward == 0
oneTenthCount != 9
Increment oneTenthCount. Toggle corresponding LEDR[9:0].
oneTenthCount == 9
Set ledBackward = 1. Set HEX0 equal to the next letter in the scrolling message.
clockIndex == 1200000 (1/10th of a second) && ledBackward == 1
oneTenthCount != 0
Decrement oneTenthCount. Toggle corresponding LEDR[9:0].
oneTenthCount == 0
Set ledBackward = 0.

**C)**

```
module L2C268(
            input [9:0] sw,
            input [3:0] key,
            input clock,
            output [9:0] ledr,
            output [7:0] ledg,
            output reg [6:0] hex3,hex2,hex1,hex0);

assign init = ~sw[9] & ~sw[8] & ~sw[7] & ~sw[6] & ~sw[5];
assign part1 = sw[9] & ~sw[8] & ~sw[7] & ~sw[6] & ~sw[5];
assign part2 = ~sw[9] & sw[8] & ~sw[7] & ~sw[6] & ~sw[5];
assign part3 = ~sw[9] & ~sw[8] & sw[7] & ~sw[6] & ~sw[5];
assign part4 = ~sw[9] & ~sw[8] & ~sw[7] & sw[6] & ~sw[5];
assign part5 = ~sw[9] & ~sw[8] & ~sw[7] & ~sw[6] & sw[5];

always @(*)
begin
            if (init)
                        begin hx3(18); hx2(2); hx1(6); hx0(8); end
            else if (part1)
                        begin
                                    case(sw[3:0])
                                                0:  begin hx3(0); hx2(0); hx1(18); hx0(0);  end
                                                1:  begin hx3(0); hx2(1); hx1(18); hx0(1);  end
                                                2:  begin hx3(0); hx2(2); hx1(18); hx0(2);  end
                                                3:  begin hx3(0); hx2(3); hx1(18); hx0(3);  end
                                                4:  begin hx3(0); hx2(4); hx1(18); hx0(4);  end
                                                5:  begin hx3(0); hx2(5); hx1(18); hx0(5);  end
                                                6:  begin hx3(0); hx2(6); hx1(18); hx0(6);  end
                                                7:  begin hx3(0); hx2(7); hx1(18); hx0(7);  end
                                                8:  begin hx3(0); hx2(8); hx1(18); hx0(8);  end
                                                9:  begin hx3(0); hx2(9); hx1(18); hx0(9);  end
                                                10: begin hx3(1); hx2(0); hx1(18); hx0(10); end
                                                11: begin hx3(1); hx2(1); hx1(18); hx0(11); end
                                                12: begin hx3(1); hx2(2); hx1(18); hx0(12); end
                                                13: begin hx3(1); hx2(3); hx1(18); hx0(13); end
                                                14: begin hx3(1); hx2(4); hx1(18); hx0(14); end
                                                15: begin hx3(1); hx2(5); hx1(18); hx0(15); end
                                    endcase
                        end
            else if (part2)
                        begin
                                    if (~sw[0])
                                                case(sw[4:1])
                                                            0:   begin hx3(0); hx2(0); hx1(18); hx0(0);  end
                                                            1:   begin hx3(0); hx2(1); hx1(18); hx0(1);  end
                                                            2:   begin hx3(0); hx2(2); hx1(18); hx0(2);  end
                                                            3:   begin hx3(0); hx2(3); hx1(18); hx0(3);  end
                                                            4:   begin hx3(1); hx2(0); hx1(18); hx0(1);  end
                                                            5:   begin hx3(1); hx2(1); hx1(18); hx0(2);  end
                                                            6:   begin hx3(1); hx2(2); hx1(18); hx0(3);  end
                                                            7:   begin hx3(1); hx2(3); hx1(18); hx0(4);  end
                                                            8:   begin hx3(2); hx2(0); hx1(18); hx0(2);  end
                                                            9:   begin hx3(2); hx2(1); hx1(18); hx0(3);  end
                                                            10:  begin hx3(2); hx2(2); hx1(18); hx0(4);  end
                                                            11:  begin hx3(2); hx2(3); hx1(18); hx0(5);  end
                                                            12:  begin hx3(3); hx2(0); hx1(18); hx0(3);  end
                                                            13:  begin hx3(3); hx2(1); hx1(18); hx0(4);  end
                                                            14:  begin hx3(3); hx2(2); hx1(18); hx0(5);  end
                                                            15:  begin hx3(3); hx2(3); hx1(18); hx0(6);  end
                                                endcase
                                    else if (sw[0])
                                                case(sw[4:1])
                                                            0:   begin hx3(0); hx2(0); hx1(18); hx0(0);  end
```

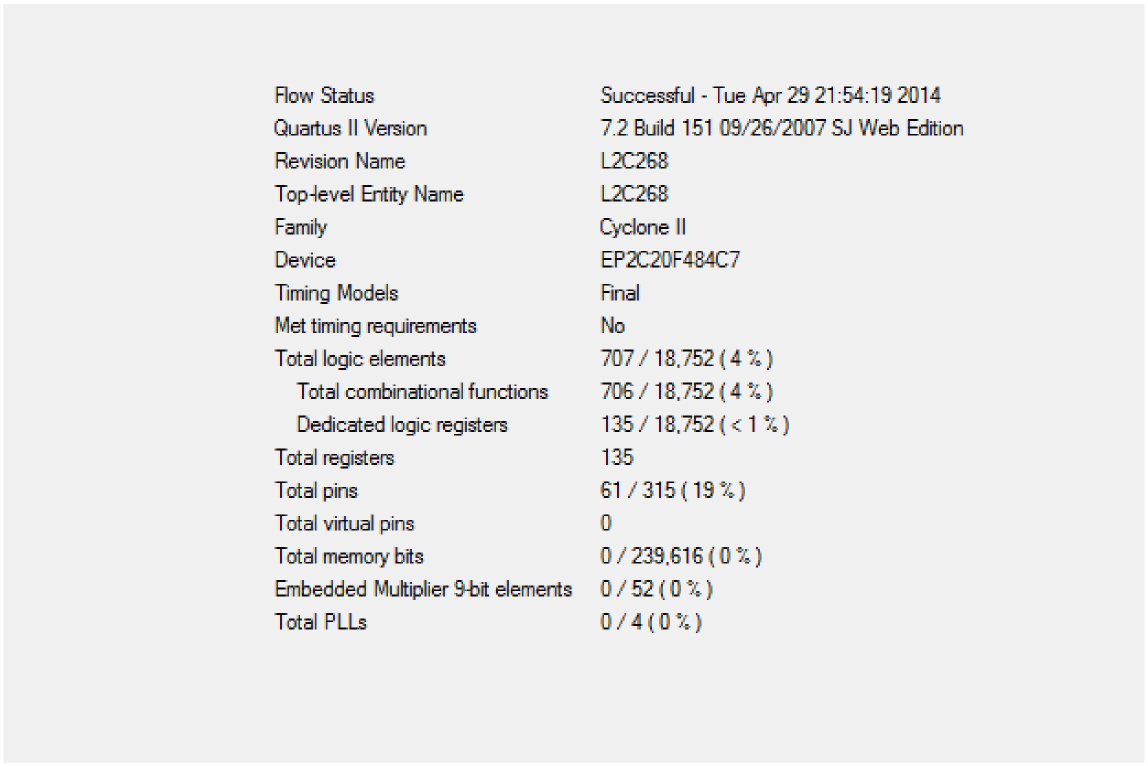
```
                                        1:   begin hx3(0); hx2(1); hx1(18); hx0(0);  end
                                        2:   begin hx3(0); hx2(2); hx1(18); hx0(0);  end
                                        3:   begin hx3(0); hx2(3); hx1(18); hx0(0);  end
                                        4:   begin hx3(1); hx2(0); hx1(18); hx0(0);  end
                                        5:   begin hx3(1); hx2(1); hx1(18); hx0(1);  end
                                        6:   begin hx3(1); hx2(2); hx1(18); hx0(2);  end
                                        7:   begin hx3(1); hx2(3); hx1(18); hx0(3);  end
                                        8:   begin hx3(2); hx2(0); hx1(18); hx0(0);  end
                                        9:   begin hx3(2); hx2(1); hx1(18); hx0(2);  end
                                        10:  begin hx3(2); hx2(2); hx1(18); hx0(4);  end
                                        11:  begin hx3(2); hx2(3); hx1(18); hx0(6);  end
                                        12:  begin hx3(3); hx2(0); hx1(18); hx0(0);  end
                                        13:  begin hx3(3); hx2(1); hx1(18); hx0(3);  end
                                        14:  begin hx3(3); hx2(2); hx1(18); hx0(6);  end
                                        15:  begin hx3(3); hx2(3); hx1(18); hx0(9);  end
                                endcase
                end
        else if (part3)
                begin
                        if (sw[0])
                                begin hx3(18); hx2(0); hx1(18); hx0(18); end
                        else
                                begin hx3(18); hx2(counter); hx1(18); hx0(18); end
                end
        else if (part4)
                begin
                        begin hx3(h3); hx2(h2); hx1(h1); hx0(h0); end
                end
        else if (part5)
                begin
                        begin hx3(m3); hx2(m2); hx1(m1); hx0(m0); end
                end
end


reg[5:0] counter;
always @(negedge key[2] or negedge sw[7] or posedge sw[0])
begin
        if (sw[0])
                begin counter = 0; end
        else if (~sw[7])
                begin counter = 0; end
        else if (~sw[1] && ~sw[0])
                begin
                        counter = counter + 1;
                        if (counter == 16)
                                begin counter = 0; end
                end
        else if (sw[1] && ~sw[0])
                begin
                        if (counter == 0)
                                begin counter = 16; end
                        counter = counter - 1;
                end
end

reg[6:0] h0;
reg[6:0] h1;
reg[6:0] h2;
reg[6:0] h3;
reg[6:0] count;
reg[6:0] oneTenthCount;
reg[6:0] message;
reg[6:0] m0;
reg[6:0] m1;
reg[6:0] m2;
```

```
reg[6:0] m3;
reg[6:0] m0trail;
reg[6:0] m1trail;
reg[6:0] m2trail;
reg [23:0] clockIndex;
reg [23:0] clockIndex2;
reg [1:0] ledToggleGreen;
reg [9:0] ledToggleRed;
reg [1:0] ledBackward;
assign ledg[0] = ledToggleGreen;
assign ledr[9:0] = ledToggleRed;
always @ (posedge clock)
begin
        if (sw[6] && ~sw[0])
                begin
                        clockIndex = clockIndex + 1;
                        if (clockIndex == 12000000)
                        begin
                                        clockIndex = 0;
                                        count = count + 1;
                                        if (count == 1)
                                                begin
                                                        ledToggleGreen = 1;
                                                end
                                        if (count == 2)
                                        begin
                                                        ledToggleGreen = 0;
                                                        count = 0;
                                                        h0 = h0 + 1;
                                                        if (h0 == 3)
                                                        begin
                                                                h0 = 0;
                                                                h1 = h1 + 1;
                                                                if (h1 == 3)
                                                                        begin
                                                                                h1 = 0;
                                                                                h2 = h2 + 1;
                                                                                if (h2 == 3)
                                                                                        begin
                                                                                          h2 = 0;
                                                                                          h3 = h3 + 1;
                                                                                          if (h3 == 3)
                                                                                                begin
                                                                                                h3 = 0;
                                                                                                h2 = 0;
                                                                                                h1 = 0;
                                                                                                h0 = 0;
                                                                                                end
                                                                                        end
                                                                        end
                                                        end
                                        end
                        end
                end
        else if (sw[6] && sw[0])
                begin ledToggleGreen = 0; clockIndex = 0; count = 0; h3 = 0; h2 = 0; h1 = 0; h0 = 0; end
        else if (~sw[6])
                begin ledToggleGreen = 0; clockIndex = 0; count = 0; h3 = 0; h2 = 0; h1 = 0; h0 = 0; end
end

always @ (posedge clock)
begin
        if (sw[5] && ~sw[0])
                begin
                        if (clockIndex2 == 0 && oneTenthCount == 0)
                                begin ledToggleRed[oneTenthCount] = 1; end
                        clockIndex2 = clockIndex2 + 1;
                        if (clockIndex2 == 1200000 && ledBackward == 0)
                                begin
                                        clockIndex2 = 0;
                                        ledToggleRed[oneTenthCount] = 0;
                                        oneTenthCount = oneTenthCount + 1;
                                        ledToggleRed[oneTenthCount] = 1;
                                        if (oneTenthCount == 9)
                                                begin
                                                        ledBackward = 1;
                                                        if (message == 0)
                                                        begin m3 = 18; m2 = 18; m1 = 18; m0 = 18; end
                                                        if (message == 19)
                                                        begin message = 0; end
                                                        message = message + 1;
                                                        m0trail = m0;
                                                        m1trail = m1;
                                                        m2trail = m2;
                                                        m1 = m0trail;
                                                        m2 = m1trail;
                                                        m3 = m2trail;
                                                        messageScroll(message);
                                                end
                                end
                        else if (clockIndex2 == 1200000 && ledBackward == 1)
                                begin
                                        clockIndex2 = 0;
                                        ledToggleRed[oneTenthCount] = 0;
                                        oneTenthCount = oneTenthCount - 1;
                                        ledToggleRed[oneTenthCount] = 1;
                                        if (oneTenthCount == 0)
                                                begin ledBackward = 0; end
                                end
                end
        else if (~sw[5])
                begin
                ledToggleRed[oneTenthCount] = 0;
                clockIndex2 = 0;
                oneTenthCount = 0;
                ledBackward = 0;
                message = 0;
                m3 = 18; m2 = 18; m1 = 18; m0 = 18;
                end
end

task messageScroll;
input [6:0] num;
begin
        case(num)
                0:      begin m0 = 16; end
                1:  begin m0 = 16; end
                2:  begin m0 = 14; end
                3:  begin m0 = 17; end
                4:  begin m0 = 17; end
                5:  begin m0 = 0;  end
                6:  begin m0 = 18; end
                7:  begin m0 = 18; end
                8:  begin m0 = 12; end
                9:  begin m0 = 1;  end
                10: begin m0 = 13; end
                11: begin m0 = 18; end
                12: begin m0 = 2;  end
                13: begin m0 = 6;  end
                14: begin m0 = 8;  end
                15: begin m0 = 18; end
                16: begin m0 = 18; end
```

```
			17: begin m0 = 18; end
			18: begin m0 = 18; end
			19: begin m0 = 18; end
		endcase
end
endtask

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
		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
		endcase
end
endtask

task hx2;
input  [6:0] num;
begin
		case(num)
			0:  hex2 =  7'b1000000;  //0
```

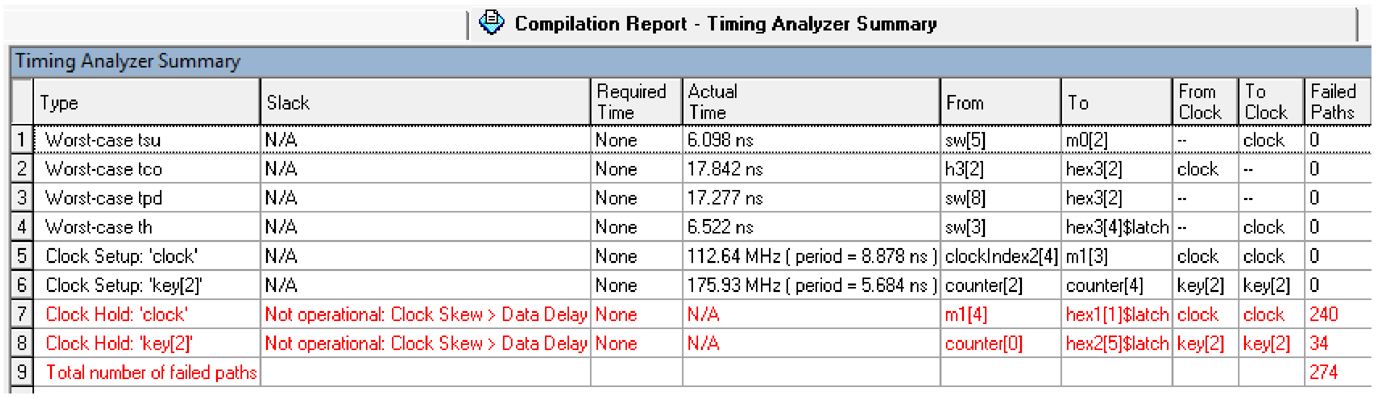
```
                1:  hex2 =  7'b1111001;  //1
                2:  hex2 =  7'b0100100;  //2
                3:  hex2 =  7'b0110000;  //3
                4:  hex2 =  7'b0011001;  //4
                5:  hex2 =  7'b0010010;  //5
                6:  hex2 =  7'b0000010;  //6
                7:  hex2 =  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
        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
        endcase
end
endtask

endmodule
```

**D)**



| | |
|---|---|
| Flow Status | Successful - Tue Apr 29 21:54:19 2014 |
| Quartus II Version | 7.2 Build 151 09/26/2007 SJ Web Edition |
| Revision Name | L2C268 |
| Top-level Entity Name | L2C268 |
| Family | Cyclone II |
| Device | EP2C20F484C7 |
| Timing Models | Final |
| Met timing requirements | No |
| Total logic elements | 707 / 18,752 ( 4 % ) |
| Total combinational functions | 706 / 18,752 ( 4 % ) |
| Dedicated logic registers | 135 / 18,752 ( < 1 % ) |
| Total registers | 135 |
| Total pins | 61 / 315 ( 19 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 239,616 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 52 ( 0 % ) |
| Total PLLs | 0 / 4 ( 0 % ) |

**E)**



Compilation Report - Timing Analyzer Summary

Timing Analyzer Summary

| | Type | Slack | Required Time | Actual Time | From | To | From Clock | To Clock | Failed Paths |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Worst-case tsu | N/A | None | 6.098 ns | sw[5] | m0[2] | -- | clock | 0 |
| 2 | Worst-case tco | N/A | None | 17.842 ns | h3[2] | hex3[2] | clock | -- | 0 |
| 3 | Worst-case tpd | N/A | None | 17.277 ns | sw[8] | hex3[2] | -- | -- | 0 |
| 4 | Worst-case th | N/A | None | 6.522 ns | sw[3] | hex3[4]$latch | -- | clock | 0 |
| 5 | Clock Setup: 'clock' | N/A | None | 112.64 MHz ( period = 8.878 ns ) | clockIndex2[4] | m1[3] | clock | clock | 0 |
| 6 | Clock Setup: 'key[2]' | N/A | None | 175.93 MHz ( period = 5.684 ns ) | counter[2] | counter[4] | key[2] | key[2] | 0 |
| 7 | Clock Hold: 'clock' | Not operational: Clock Skew > Data Delay | None | N/A | m1[4] | hex1[1]$latch | clock | clock | 240 |
| 8 | Clock Hold: 'key[2]' | Not operational: Clock Skew > Data Delay | None | N/A | counter[0] | hex2[5]$latch | key[2] | key[2] | 34 |
| 9 | Total number of failed paths | | | | | | | | 274 |