UCSD CSE140L Spring 2014

**LAB#4 Report**

Demonstration Date : 06 / 3 /14 Student CID\_\_\_\_\_\_\_\_\_\_580\_\_\_\_\_\_\_\_\_\_\_\_\_

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

First M.I. Last

**TED Submission Date & Time : June 3, 2014 9:30am**

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

**Self-test Report** Demo Reviewer

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

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

**Part1**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/3 Procedural Description( )/1

**Part2**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/3 Verilog HDL codes ( )/1

**Part3**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_/3 State Diagram ( )/2

**Part4**: \_\_\_\_X\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_/3 Compilation Report ( )/1

(Screen copy)

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

**Subtotal**  **Subtotal**

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

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

1. Description:
   1. In the precondition, Since the program was already functional with these requirements to begin with. The only task I had to do for this step was change the CID to my CID.
   2. In part one, I used key[3], sw[2:0] and HEX[3:0]. Key[3] was used as the reset button, HEX[3:0] to display values, and sw[2:0] was used as inputs to determine what to display.
   3. In part two, I used key[3], sw[2:0] and HEX[3:0]. Key[3] was used as the reset button, HEX[3:0] to display values, and sw[2:0] was used as inputs to determine what to display.
   4. In part three, I used key[3], sw[2:0] and HEX[3:0]. Key[3] was used as the reset button, HEX[3:0] to display values, and sw[2:0] was used as inputs to determine what to display.
   5. In part four, I found the sw[?] that freezes the cpu by testing all the switches until I found the right switch and finding it in lab4\_de1.v.
   6. In part five, I used sw[8] in order to indicate if I want the clock speed to be 5 times faster. I put in a condition if the switch is up it will run 5 times faster.
2. Verilog Code:

Lab4\_de1.v----------------------------------------------------------------------------------------------------------------------------------

always @ (\*)

begin

case (SW[2:0])

3'b000: hexdata <= 16'h0580;

3'b001: hexdata <= register\_A ;

3'b010: hexdata <= program\_counter ;

3'b011: hexdata <= instruction\_register ;

3'b100: hexdata <= memory\_data\_register\_out ;

3'b111: hexdata <= out;

default: hexdata <= 16'h0580 ;

endcase

end

clock\_divider clk1Hzfrom50MHz (

CLOCK\_50,

KEY[3],

clk\_1Hz,

SW[8]

);

Tc140l.v------------------------------------------------------------------------------------------------------------------------------------------------

reg [3:0] random = 4'b0101;

// State Encodings

parameter reset\_pc = 5'h0,

fetch = 5'h1,

decode = 5'h2,

execute\_add = 5'h3,

execute\_store = 5'h4,

execute\_store2 = 5'h5,

execute\_store3 = 5'h6,

execute\_load = 5'h7,

execute\_jump = 5'h8,

execute\_jump\_n = 5'h9,

execute\_out = 5'ha,

execute\_xor = 5'hb,

execute\_or = 5'hc,

execute\_and = 5'hd,

execute\_jpos = 5'he,

execute\_jzero = 5'hf,

execute\_addi = 5'h10,

execute\_shl = 5'h11,

execute\_shr = 5'h12,

execute\_sub = 5'h13,

execute\_random = 5'h14,

execute\_addind = 5'h15,

execute\_addpcr = 5'h16;

always @(posedge clock or posedge reset) begin

if (reset) begin

state <= reset\_pc;

program\_counter <= 8'b00000000;

register\_A <= 16'b0000000000000000;

out <= 16'h0 ;

random <= 4'b0101;

end

// Execute the RANDOM instruction

execute\_random :

begin

random <= (random << 1);

random[0] <= (random[3] ~^ random[2]);

register\_A <= random;

state <= fetch;

end

// Execute the ADDIND instruction

execute\_addind :

begin

state <= execute\_add;

end

// Execute the ADDPCR instruction

execute\_addpcr :

begin

state <= execute\_add;

end

execute\_random: memory\_address\_register <= program\_counter;

execute\_addind: memory\_address\_register <= memory\_data\_register;

execute\_addpcr: memory\_address\_register <= instruction\_register[7:0] + program\_counter;

instruction\_decoder.v-----------------------------------------------------------------------------------------------------------------------------------------------------

// State Encodings

parameter

fetch = 5'h1,

execute\_add = 5'h3,

execute\_store = 5'h4,

execute\_load = 5'h7,

execute\_jump = 5'h8,

execute\_jump\_n = 5'h9,

execute\_out = 5'ha,

execute\_xor = 5'hb,

execute\_or = 5'hc,

execute\_and = 5'hd,

execute\_jpos = 5'he,

execute\_jzero = 5'hf,

execute\_addi = 5'h10,

execute\_shl = 5'h11,

execute\_shr = 5'h12,

execute\_sub = 5'h13,

execute\_random = 5'h14,

execute\_addind = 5'h15,

execute\_addpcr = 5'h16;

8'b01000101 :

state <= execute\_random;

8'b01000110 :

state <= execute\_addind;

8'b01000111 :

state <= execute\_addpcr;

clock\_divider.v-----------------------------------------------------------------------------------------------------------------------------------------------

module clock\_divider (clk, rst\_n, clk\_o, sw);

input sw;

else

begin

if (sw == 0) begin

div <= div + 1;

end

else if (sw == 1) begin

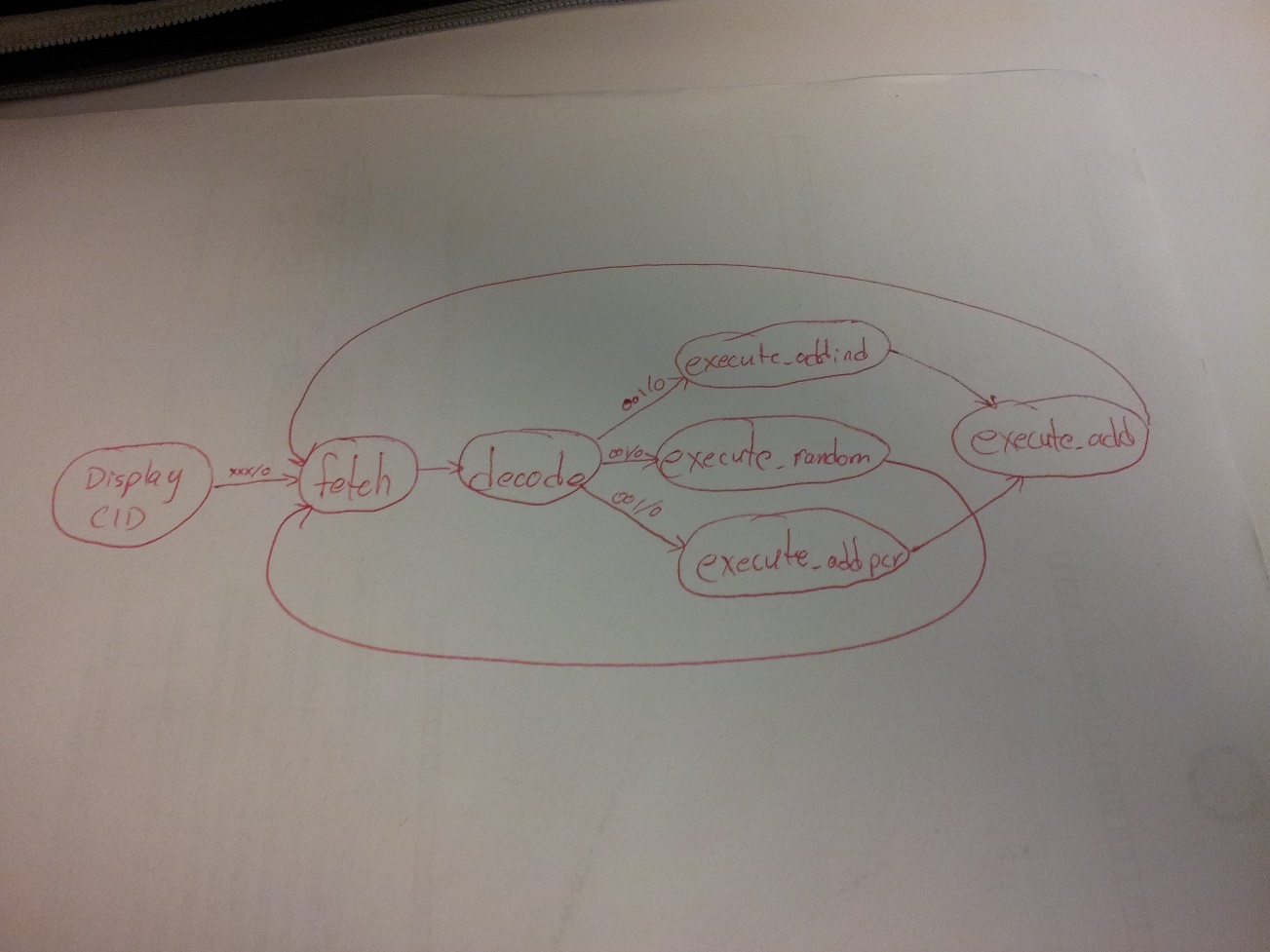
div <= div + 5;

end

en <= 0;

end

1. State Diagram:



1. Compilation Report:

