ASSIGNMENT-5

LALITHA MATHI; UFID-29341966

VHDL CODE FOR 8-BIT SEQUENCE DETECTOR:

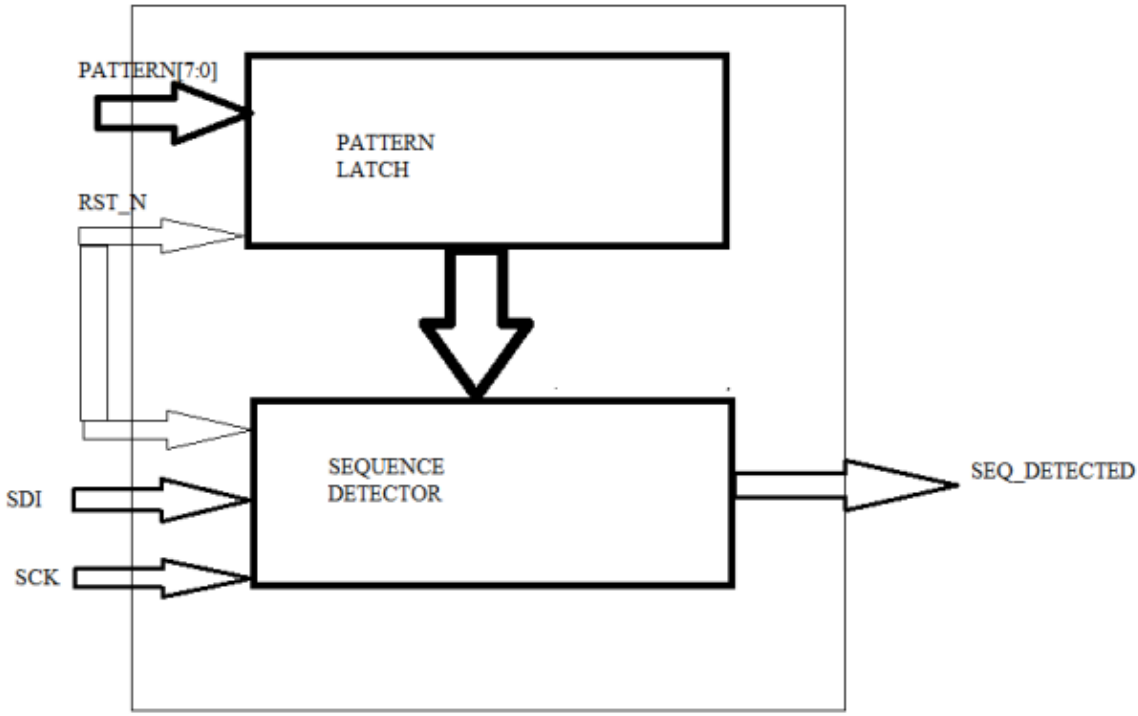
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FSM is
Port ( PATTERN : in  STD_LOGIC_VECTOR(7 DOWNTO 0);
RST_N : in  STD_LOGIC;
SDI : in  STD_LOGIC;
SCK : in  STD_LOGIC;
SEQ_DETECTED : out  STD_LOGIC);
end FSM;

architecture FSM of FSM is
type state_type is(STATE_0, STATE_1, STATE_2, STATE_3, STATE_4, STATE_5, STATE_6,
STATE_7, STATE_8);
signal present_state: state_type;
signal next_state: state_type;
signal pattern_n: STD_LOGIC_Vector(7 downto 0);
begin
process (RST_N, SCK)
begin
        if (RST_N = '0') then
        present_state <= STATE_0;
        elsif (rising_edge(SCK)) then
        present_state <= next_state;
        end if;
end process;
process(RST_N)
begin
        if(RST_N = '1' and RST_N'event) then
        pattern_n <= pattern;
        end if;
end process;
process(present_state, SDI,pattern_n)
begin
next_state <= present_state;
SEQ_DETECTED <= '0';
case present_state is
when STATE_0 => if( SDI = pattern_n(7)) then
                                        next_state <= STATE_1;
                                        end if;
```

ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

```
when STATE_1 => if( SDI = pattern_n(6)) then
                    next_state <= STATE_2;
                    elsif (SDI = pattern_n(7)) then
                    next_state <= STATE_1;
                    else
                    next_state <= STATE_0;
                     end if;

when STATE_2 => if( SDI = pattern_n(5)) then
                    next_state <= STATE_3;
                     elsif (SDI = pattern_n(7)) then
                    next_state <= STATE_1;
                     else
                    next_state <= STATE_0;
                    end if;
when STATE_3 => if( SDI = pattern_n(4)) then
                    next_state <= STATE_4;
                     elsif (SDI = pattern_n(7)) then
                    next_state <= STATE_1;
                     else
                    next_state <= STATE_0;
                     end if;
when STATE_4 => if( SDI = pattern_n(3)) then
                    next_state <= STATE_5;
                     elsif (SDI = pattern_n(7)) then
                    next_state <= STATE_1;
                     else
                    next_state <= STATE_0;
                     end if;
when STATE_5 => if( SDI <= pattern_n(2)) then
                    next_state <= STATE_6;
                     elsif (SDI <= pattern_n(7)) then
                    next_state <= STATE_1;
                     else
                    next_state <= STATE_0;
                     end if;
when STATE_6 => if( SDI = pattern_n(1)) then
                    next_state <= STATE_7;
                     elsif (SDI = pattern_n(7)) then
                    next_state <= STATE_1;
                     else
                    next_state <= STATE_0;
                     end if;
when STATE_7 => if( SDI = pattern_n(0)) then
```

```
                                        next_state <= STATE_8;
                                         elsif (SDI = pattern_n(7)) then
                                        next_state <= STATE_1;
                                         else
                                        next_state <= STATE_0;
                                        end if;
when STATE_8 => if (SDI = pattern_n(7)) then
                                        next_state <= STATE_1;
                                        else next_state <= STATE_0;
                                        end if;
SEQ_DETECTED <= '1';
end case;
end process;
end FSM;
```

BLOCK DIAGRAM:

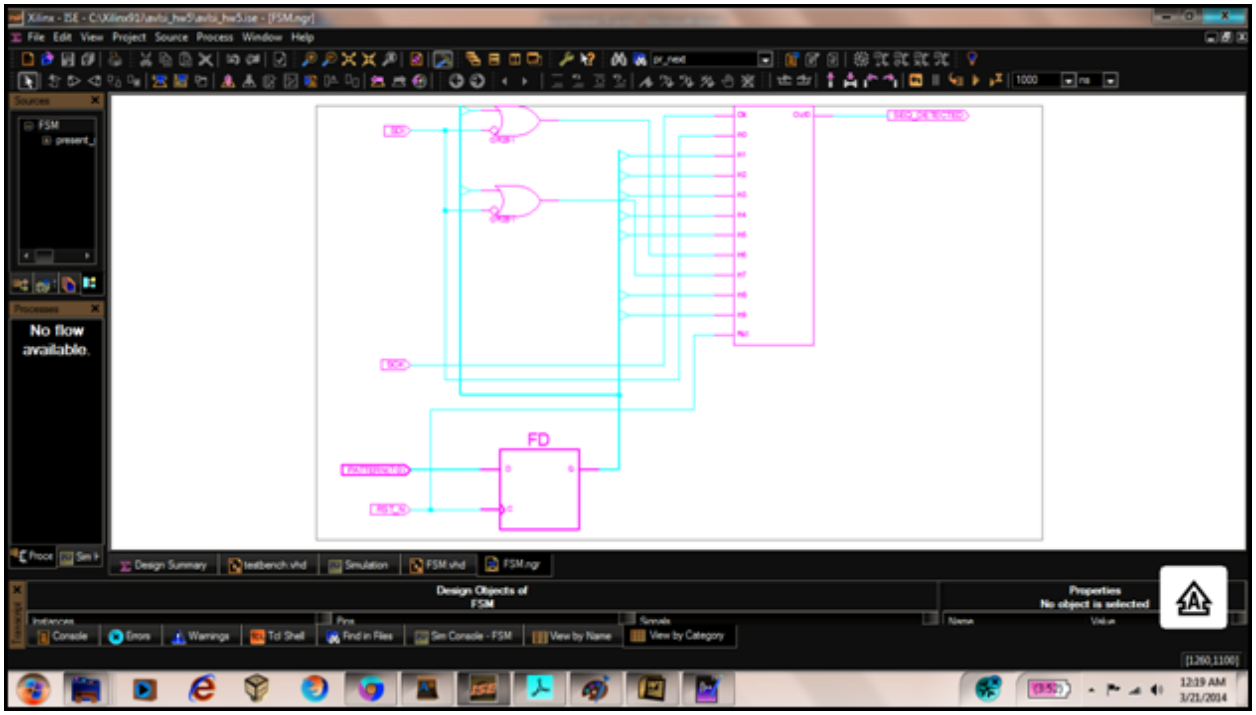


DESCRIPTION:

The basic function of this 8-bit sequence detector is to monitor the serial incoming data and assert the output flag signal when a certain sequence (data pattern) is detected. Here PATTERN LATCH has an input of 7 bit pattern whose operation is controlled by the RST_N(Reset) signal. Thus if RST_N is '1' or high, the 8 bit sequence will be detected and if RST_N is '0' or low, system will be reset. SDI is the sequence detector input given which has controller signals, RST_N and SCK(Clock). Here sequence will be detected only at the rising edge of the SCK and only when the RST_N is high. A new data pattern will

only be detected at the rising edge of the RST_N. When the pattern is detected the SEQ_DETECTED signal will go high for one clock cycle showing that the sequence has been detected successfully.

The Synthesis of the figure in Xilinx Simulator gives the following RTL Schematic showing the pattern Latch and Sequence detector.



TEST BENCH:

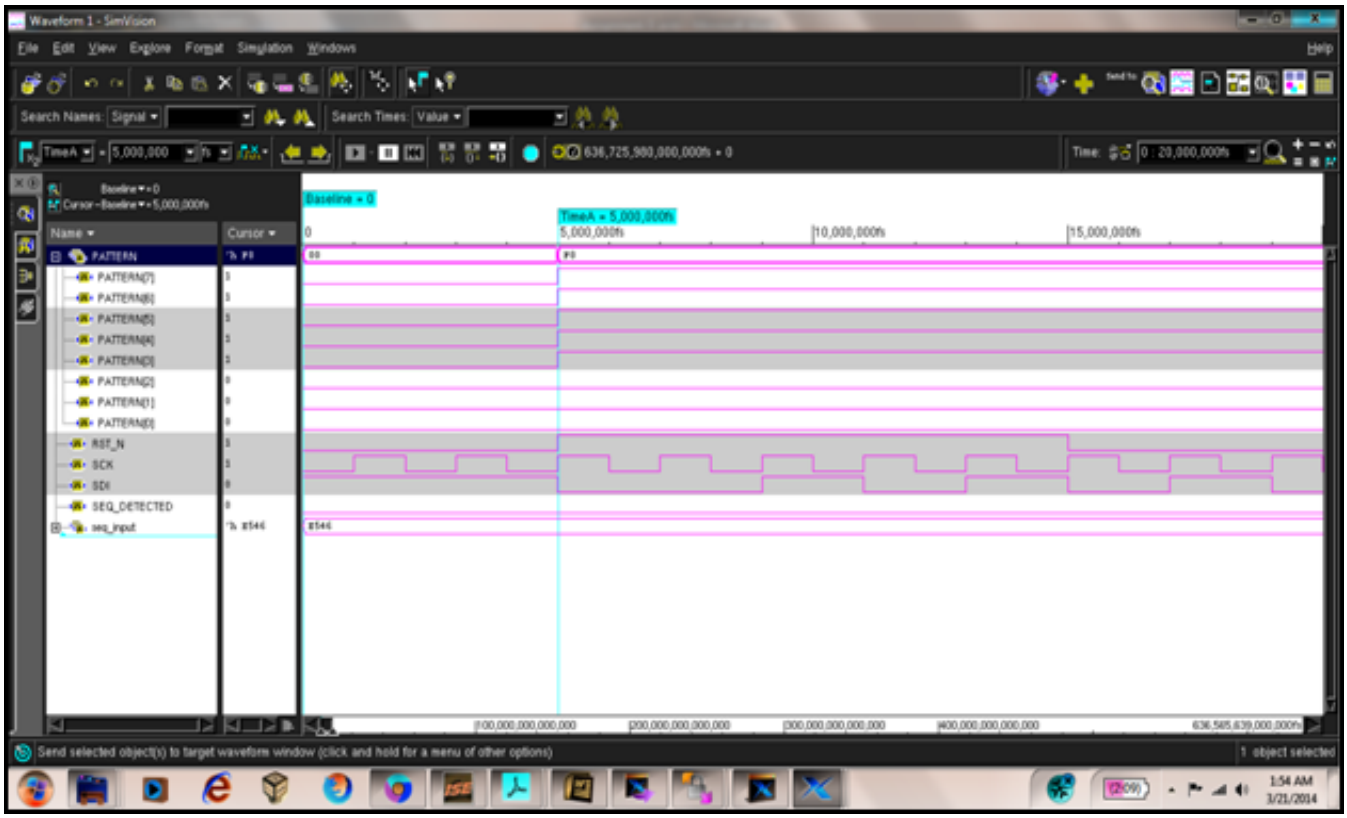
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY testbench_vhd IS
END testbench_vhd;

ARCHITECTURE behavior OF testbench_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT FSM
        PORT(
                PATTERN : IN std_logic_vector(7 downto 0);
                RST_N : IN std_logic;
                SDI : IN std_logic;
                SCK : IN std_logic;
                SEQ_DETECTED : OUT std_logic
                );
```

```
        END COMPONENT;

        --Inputs
        SIGNAL RST_N :  std_logic := '0';
        SIGNAL SDI :  std_logic := '0';
        SIGNAL SCK :  std_logic := '0';
        SIGNAL PATTERN :  std_logic_vector(7 downto 0) := (others=>'0');

        --Outputs
        SIGNAL SEQ_DETECTED :  std_logic;
  --Variables
        SIGNAL seq_input:std_logic_vector(15 downto 0):="1110010101000110";

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: FSM PORT MAP(
                PATTERN => PATTERN,
                RST_N => RST_N,
                SDI => SDI,
                SCK => SCK,
                SEQ_DETECTED => SEQ_DETECTED
        );
 SCK <= not SCK after 1 ns;
        tb : PROCESS
  BEGIN
          wait until SCK'event and SCK = '1';
   wait until SCK'event and SCK = '1';
          wait until SCK'event and SCK = '1';
                                RST_N <= '1';
                                pattern <= "11111000";

                wait for 10 ns;
                                RST_N <= '0';
                wait for 80 ns;
                                RST_N <= '1';
                                pattern <="11100101";

                wait for 20 ns;
                                RST_N <='1' ;
                                wait;
        END PROCESS;
  PROCESS
        BEGIN
                for i in 15 downto 0 loop
```

```
            SDI <= seq_input(i);
            wait until SCK'event and SCK='1';
            end loop;
    END PROCESS;
    END;
```

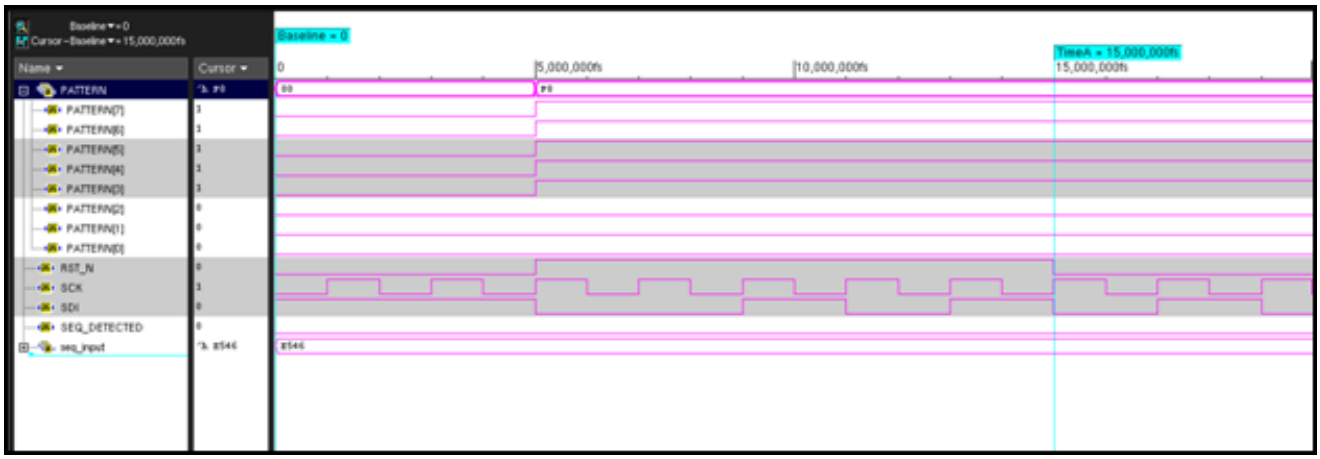
RESULTS:

In **case 1** Time =5 ns
RST_N becomes 1
Pattern changes to F8 i.e., '11111000'
SEQ_DETECTED signal is '0'

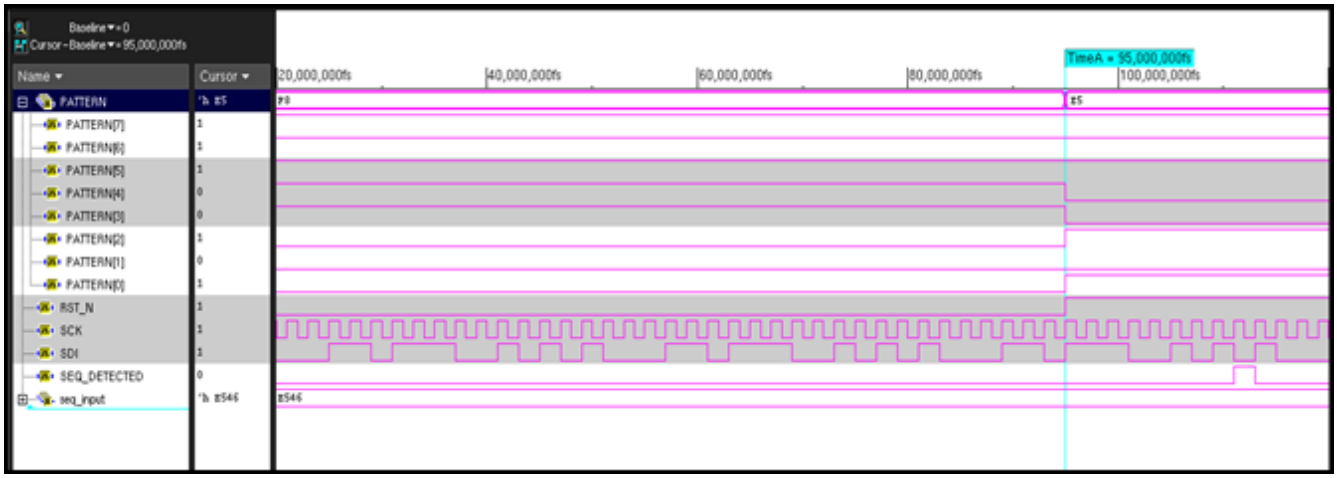


In **case 2** Time =15 ns
RST_N becomes 0
Pattern changes to F8 i.e., '11111000'
SEQ_DETECTED signal is '0'

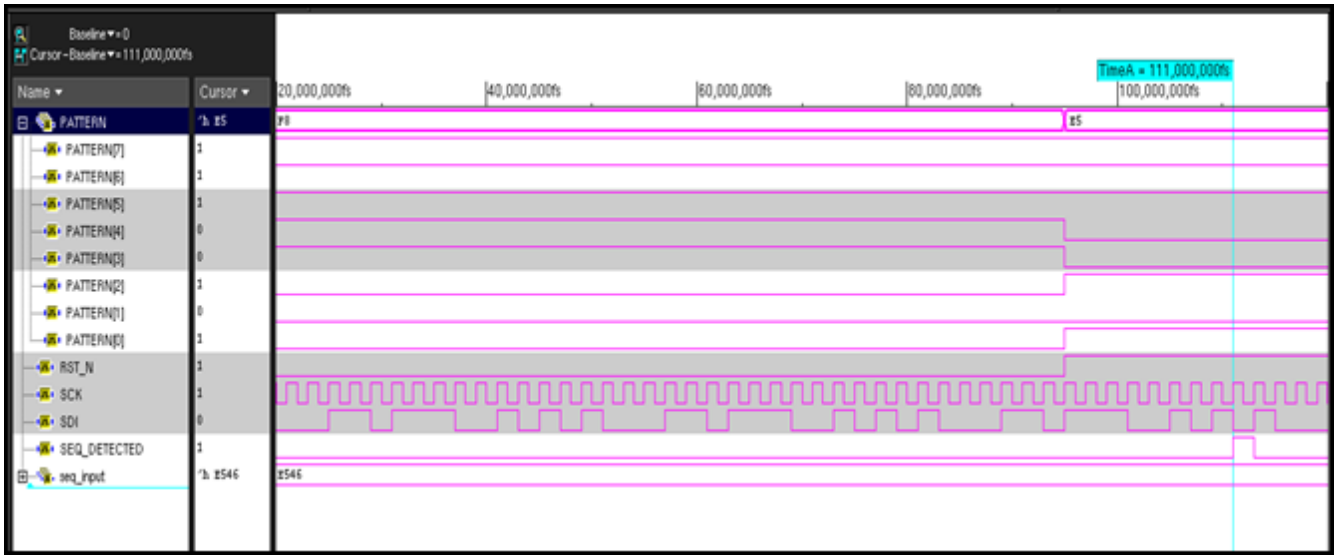
ASSIGNMENT-5
LALITHA MATHI; UFID-29341966



In **case 3** Time =95 ns
RST_N becomes 1
Pattern changes to E5 i.e., '11100011'
SEQ_DETECTED signal is '0'

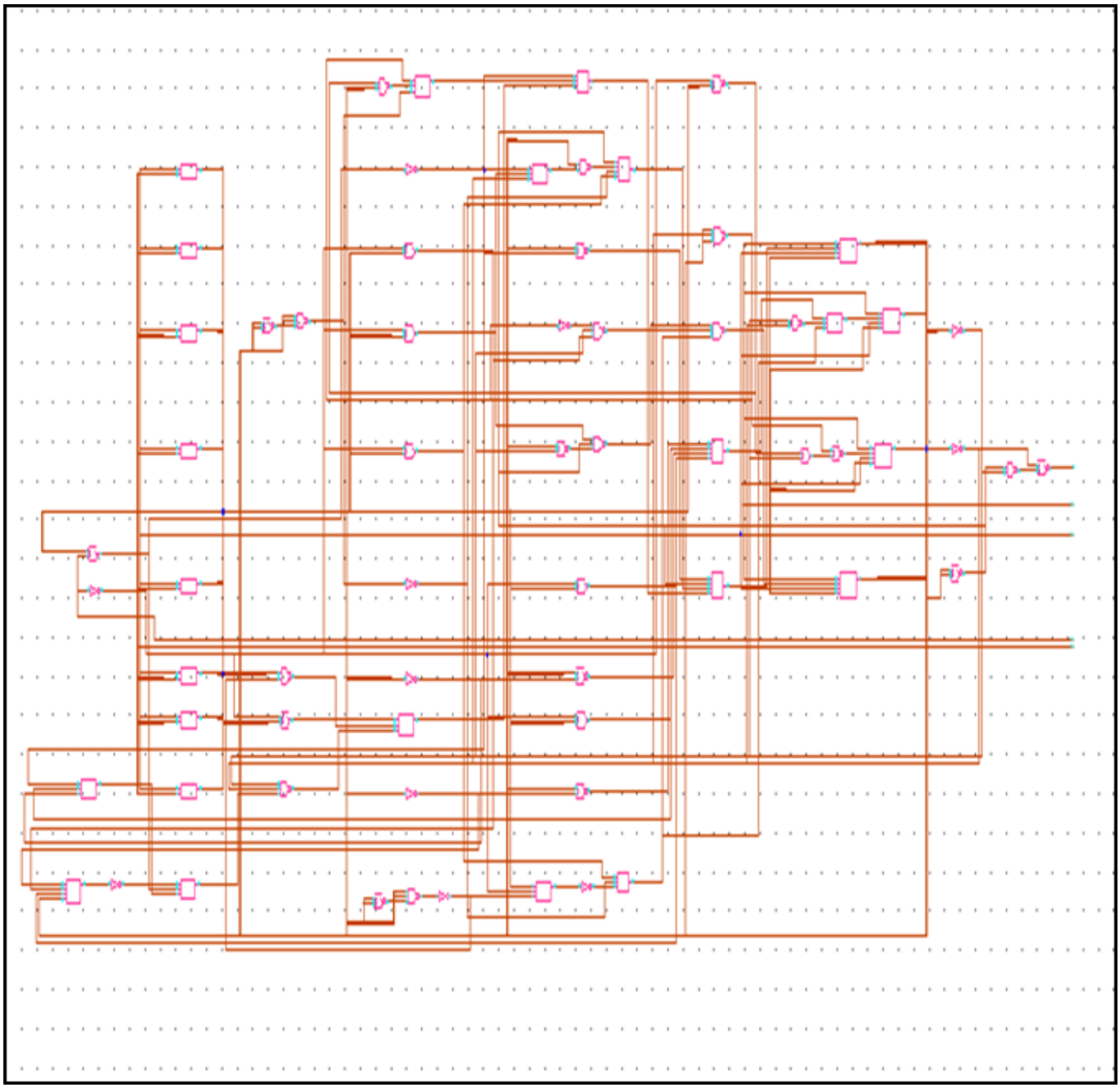


Case4
After 7 cycles the SEQ_DETECTED signal becomes '1' after detecting the sequence pattern at 115 ns.



ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

GATE LEVEL SCHEMATIC:



GATE LEVEL NETLIST(Final.v):

```
/*
##################################################################
#  Generated by:      Cadence Encounter 10.13-s292_1
#  OS:                Linux x86_64(Host ID wks18.ecel.ufl.edu)
#  Generated on:      Thu Mar 20 17:34:46 2014
#  Design:            FSM
#  Command:           saveNetlist -excludeLeafCell final.v
##################################################################
```

ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

```
*/
module FSM (
        PATTERN,
        RST_N,
        SDI,
        SCK,
        SEQ_DETECTED);
  input [7:0] PATTERN;
  input RST_N;
  input SDI;
  input SCK;
  output SEQ_DETECTED;

  // Internal wires
  wire n5;
  wire n6;
  wire n7;
  wire n8;
  wire n9;
  wire n10;
  wire n11;
  wire n12;
  wire n13;
  wire n14;
  wire n15;
  wire n16;
  wire n17;
  wire n18;
  wire n19;
  wire n20;
  wire n21;
  wire n22;
  wire n23;
  wire n24;
  wire n25;
  wire n26;
  wire n27;
  wire n28;
  wire n29;
  wire n30;
  wire n31;
  wire n32;
  wire n33;
  wire n34;
  wire n35;
  wire n36;
  wire n37;
  wire n38;
  wire n39;
  wire n40;
  wire n41;
```

```
wire n42;
wire n43;
wire n44;
wire n45;
wire n46;
wire n47;
wire n48;
wire n49;
wire n50;
wire n51;
wire n52;
wire n53;
wire n54;
wire n55;
wire n56;
wire n57;
wire [3:0] pr_state;
wire [7:0] pr_next;

DFFPOSX1 pr_next_reg_7_ (.Q(pr_next[7]),
        .D(PATTERN[7]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_6_ (.Q(pr_next[6]),
        .D(PATTERN[6]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_5_ (.Q(pr_next[5]),
        .D(PATTERN[5]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_4_ (.Q(pr_next[4]),
        .D(PATTERN[4]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_3_ (.Q(pr_next[3]),
        .D(PATTERN[3]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_2_ (.Q(pr_next[2]),
        .D(PATTERN[2]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_1_ (.Q(pr_next[1]),
        .D(PATTERN[1]),
        .CLK(RST_N));
DFFPOSX1 pr_next_reg_0_ (.Q(pr_next[0]),
        .D(PATTERN[0]),
        .CLK(RST_N));
DFFSR pr_state_reg_3_ (.S(1'b1),
        .R(RST_N),
        .Q(pr_state[3]),
        .D(n54),
        .CLK(SCK));
DFFSR pr_state_reg_1_ (.S(1'b1),
        .R(RST_N),
        .Q(pr_state[1]),
```

```
        .D(n56),
        .CLK(SCK));
DFFSR pr_state_reg_0_ (.S(1'b1),
        .R(RST_N),
        .Q(pr_state[0]),
        .D(n57),
        .CLK(SCK));
DFFSR pr_state_reg_2_ (.S(1'b1),
        .R(RST_N),
        .Q(pr_state[2]),
        .D(n55),
        .CLK(SCK));
OR2X2 U7 (.Y(n17),
        .B(n10),
        .A(n18));
OR2X2 U8 (.Y(n39),
        .B(pr_next[2]),
        .A(n15));
INVX2 U9 (.Y(n5),
        .A(n29));
INVX2 U10 (.Y(n6),
        .A(n46));
INVX2 U11 (.Y(n7),
        .A(n49));
INVX2 U12 (.Y(n8),
        .A(n16));
INVX2 U13 (.Y(n9),
        .A(pr_state[3]));
INVX2 U14 (.Y(n10),
        .A(pr_state[2]));
INVX2 U15 (.Y(n11),
        .A(n45));
INVX2 U16 (.Y(n12),
        .A(pr_state[1]));
INVX2 U17 (.Y(n13),
        .A(pr_state[0]));
INVX2 U18 (.Y(n14),
        .A(n43));
INVX2 U19 (.Y(n15),
        .A(SDI));
NOR2X1 U20 (.Y(n54),
        .B(n17),
        .A(n16));
NAND3X1 U21 (.Y(n55),
        .C(n21),
        .B(n20),
        .A(n19));
NAND3X1 U22 (.Y(n20),
        .C(n22),
        .B(n10),
        .A(n8));
```

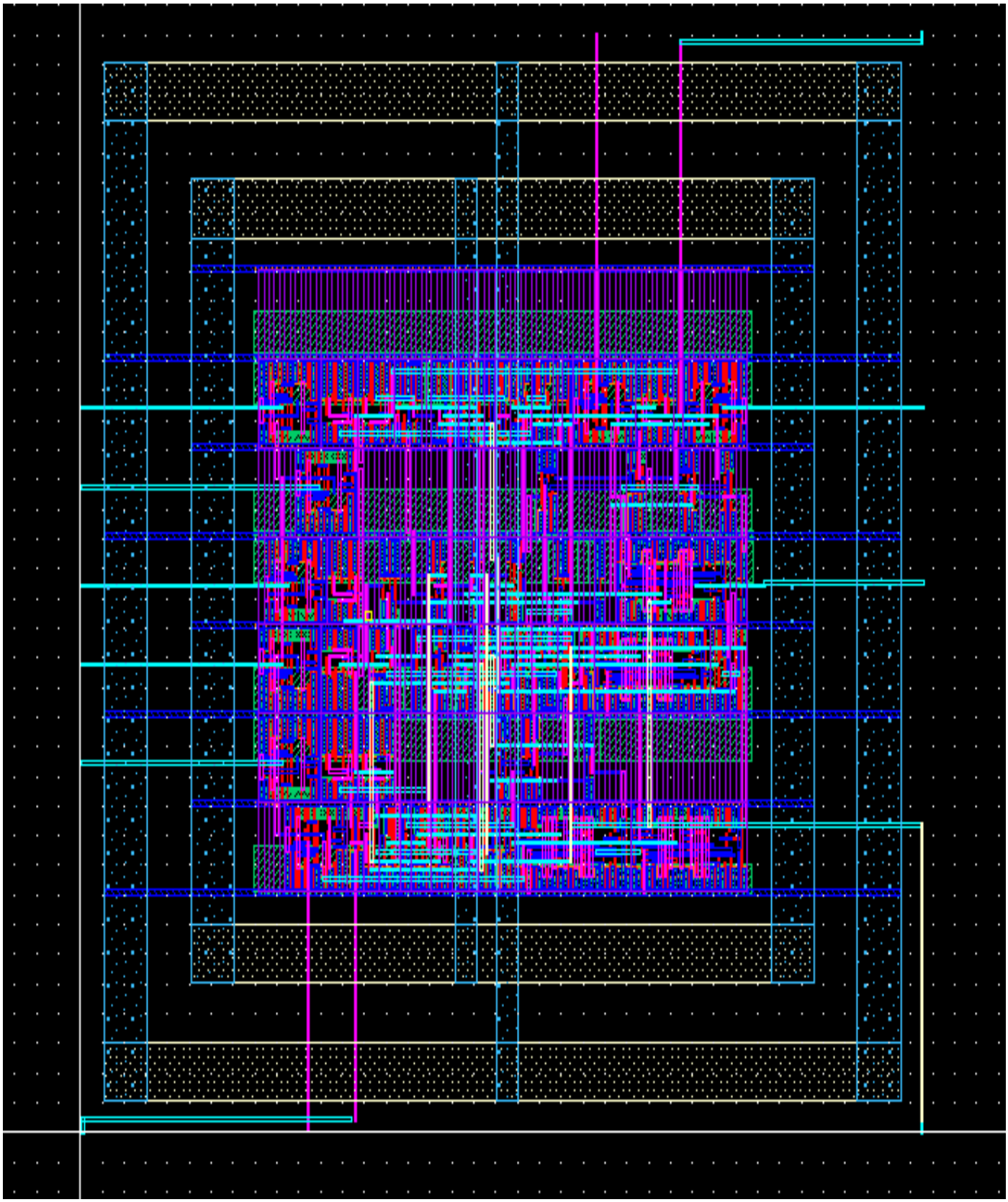
```
NAND3X1 U23 (.Y(n19),
        .C(n25),
        .B(n24),
        .A(n23));
NOR2X1 U24 (.Y(n25),
        .B(n10),
        .A(pr_state[3]));
OAI21X1 U25 (.Y(n56),
        .C(n21),
        .B(n27),
        .A(n26));
AOI21X1 U26 (.Y(n21),
        .C(n5),
        .B(n7),
        .A(n28));
OAI21X1 U27 (.Y(n29),
        .C(n6),
        .B(n15),
        .A(pr_next[2]));
NAND2X1 U28 (.Y(n27),
        .B(n9),
        .A(n10));
AOI22X1 U29 (.Y(n26),
        .D(n33),
        .C(n32),
        .B(n31),
        .A(n30));
NOR2X1 U30 (.Y(n32),
        .B(n13),
        .A(pr_state[1]));
NOR2X1 U31 (.Y(n30),
        .B(n12),
        .A(pr_state[0]));
AOI22X1 U32 (.Y(n57),
        .D(n36),
        .C(n35),
        .B(n24),
        .A(n34));
AOI21X1 U33 (.Y(n36),
        .C(n38),
        .B(n37),
        .A(n14));
OAI21X1 U34 (.Y(n38),
        .C(n41),
        .B(n40),
        .A(n39));
NAND3X1 U35 (.Y(n41),
        .C(n42),
        .B(n9),
        .A(n10));
OAI21X1 U36 (.Y(n42),
```

```
        .C(n44),
        .B(n43),
        .A(n11));
OAI21X1 U37 (.Y(n44),
        .C(n13),
        .B(n31),
        .A(n14));
XOR2X1 U38 (.Y(n31),
        .B(pr_next[5]),
        .A(n15));
OAI22X1 U39 (.Y(n45),
        .D(pr_state[1]),
        .C(n33),
        .B(n22),
        .A(n12));
XOR2X1 U40 (.Y(n33),
        .B(pr_next[6]),
        .A(n15));
XOR2X1 U41 (.Y(n22),
        .B(pr_next[4]),
        .A(n15));
NAND2X1 U42 (.Y(n40),
        .B(n6),
        .A(pr_next[7]));
NAND3X1 U43 (.Y(n46),
        .C(n47),
        .B(pr_state[2]),
        .A(pr_state[0]));
NOR2X1 U44 (.Y(n47),
        .B(pr_state[1]),
        .A(pr_state[3]));
OAI21X1 U45 (.Y(n37),
        .C(n49),
        .B(n48),
        .A(n16));
NAND2X1 U46 (.Y(n48),
        .B(pr_state[2]),
        .A(n18));
XNOR2X1 U47 (.Y(n18),
        .B(pr_next[0]),
        .A(n15));
NAND3X1 U48 (.Y(n16),
        .C(pr_state[1]),
        .B(n9),
        .A(pr_state[0]));
AOI22X1 U49 (.Y(n35),
        .D(n28),
        .C(n7),
        .B(n50),
        .A(n24));
XOR2X1 U50 (.Y(n28),
```

```
        .B(pr_next[1]),
        .A(n15));
  NAND3X1 U51 (.Y(n49),
        .C(n51),
        .B(pr_state[2]),
        .A(pr_state[1]));
  NOR2X1 U52 (.Y(n51),
        .B(pr_state[0]),
        .A(pr_state[3]));
  NAND2X1 U53 (.Y(n50),
        .B(n52),
        .A(pr_state[2]));
  OAI21X1 U54 (.Y(n52),
        .C(n9),
        .B(n23),
        .A(n14));
  XOR2X1 U55 (.Y(n23),
        .B(pr_next[3]),
        .A(n15));
  NOR2X1 U56 (.Y(n34),
        .B(n14),
        .A(pr_state[2]));
  XOR2X1 U57 (.Y(n43),
        .B(SDI),
        .A(pr_next[7]));
  NOR2X1 U58 (.Y(SEQ_DETECTED),
        .B(n53),
        .A(n9));
  NAND2X1 U59 (.Y(n53),
        .B(n10),
        .A(n24));
  NOR2X1 U60 (.Y(n24),
        .B(pr_state[1]),
        .A(pr_state[0]));
endmodule
```

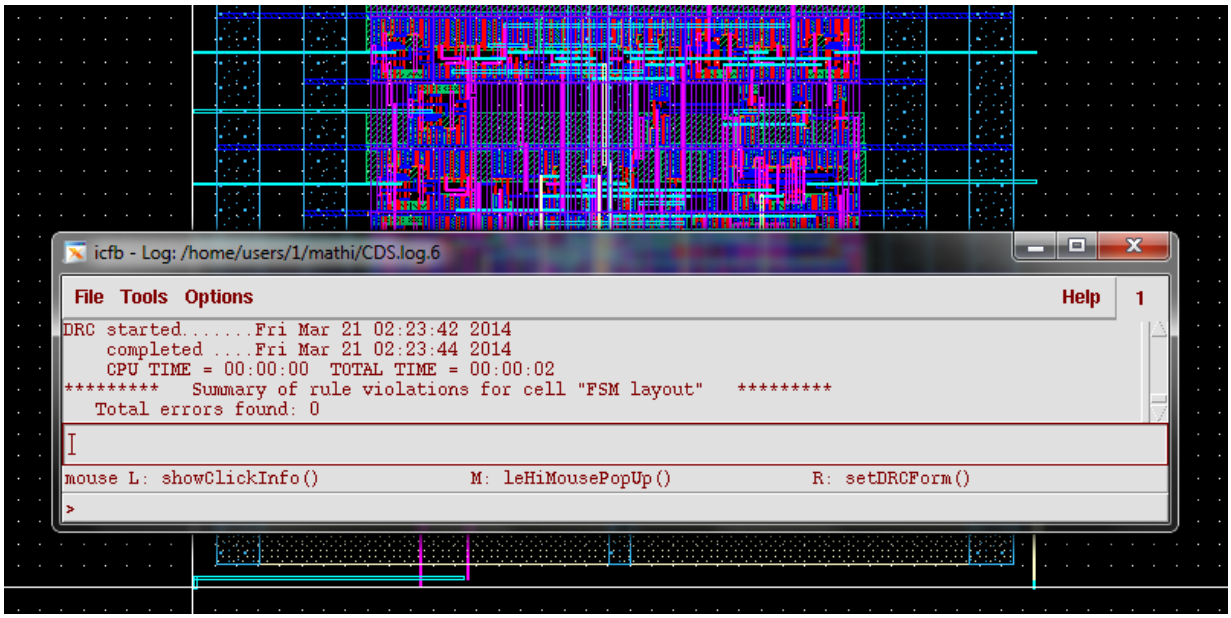
LAYOUT:

ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

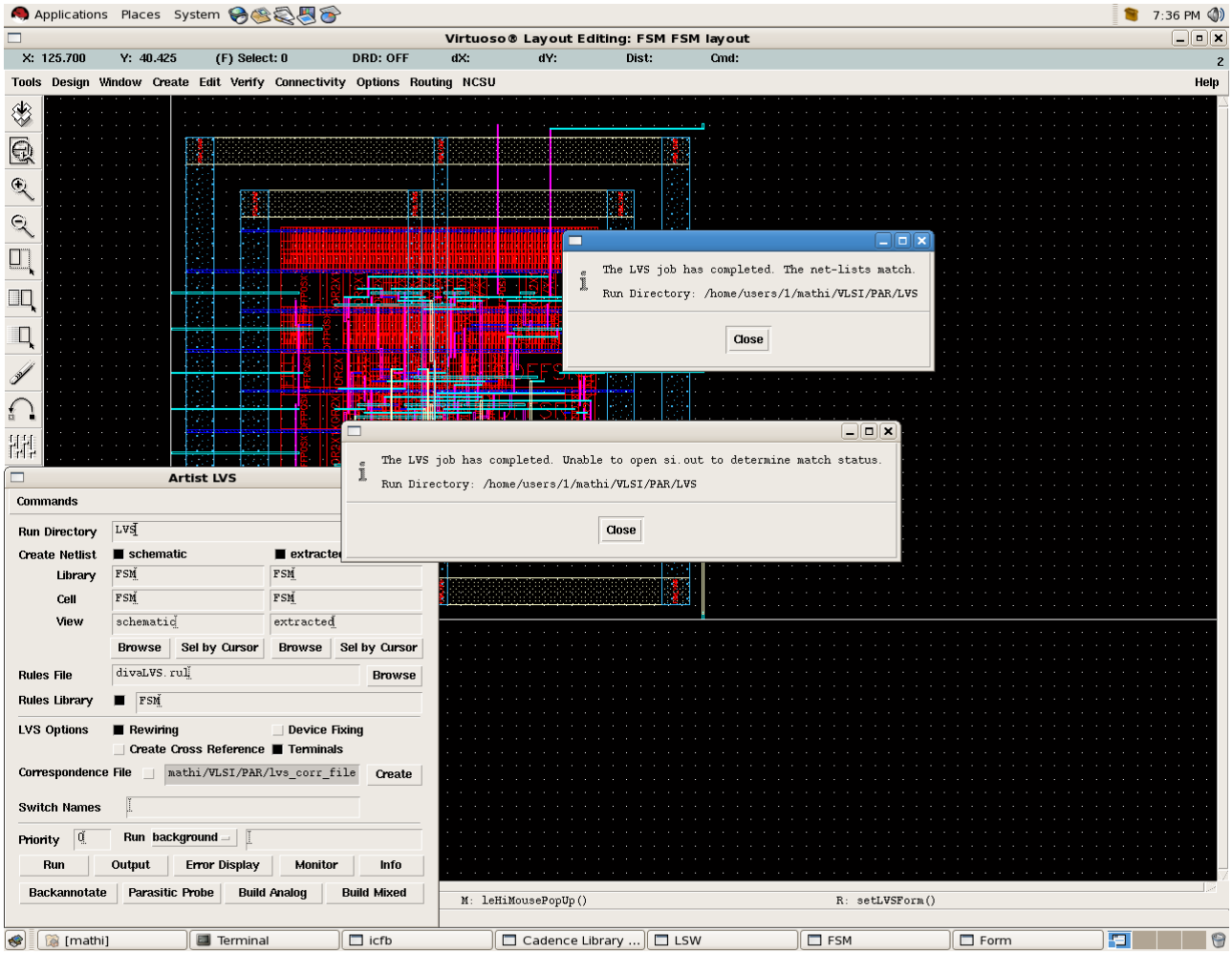


ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

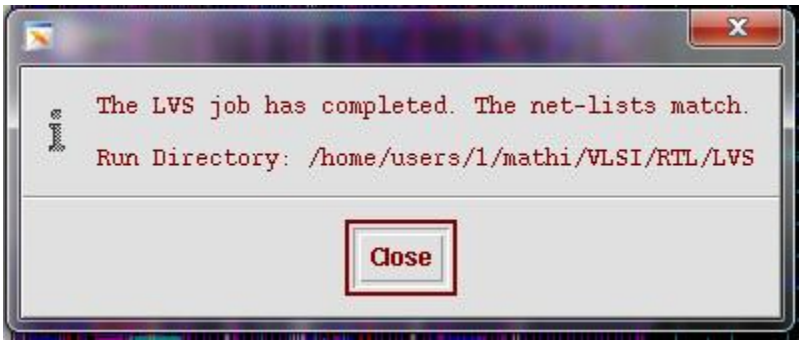
DRC CLEAN REPORT:

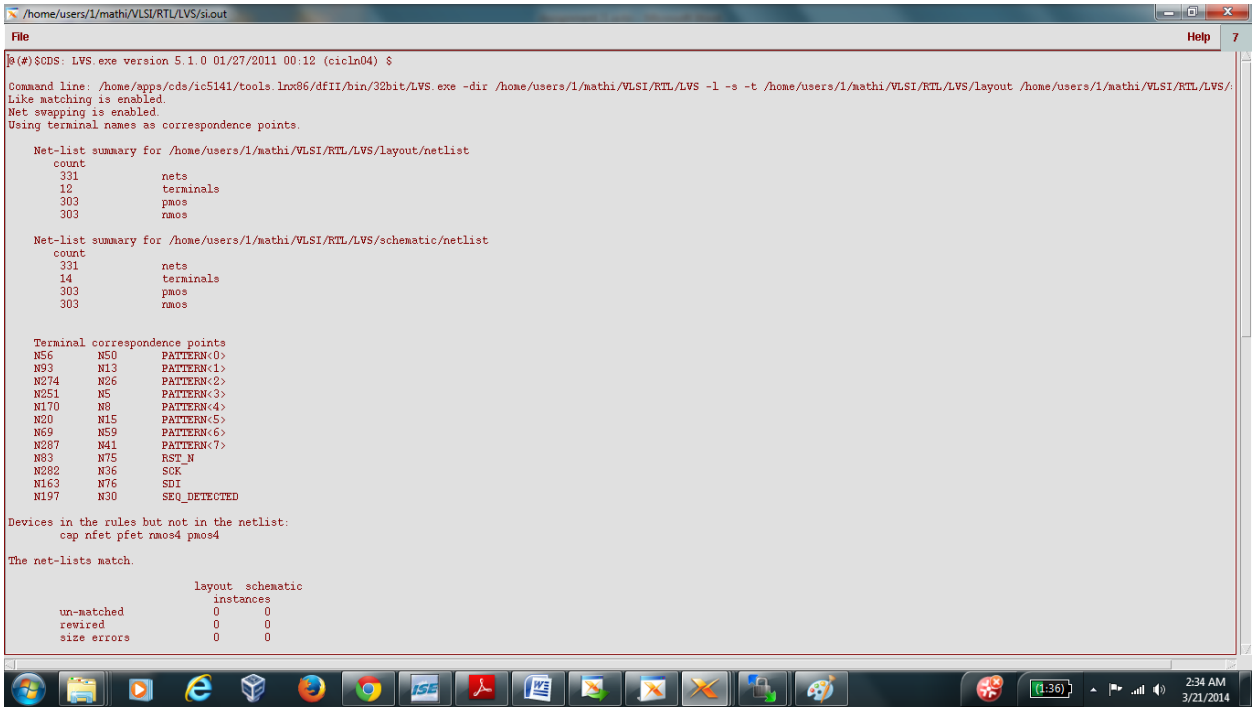


LVS REPORT:



ASSIGNMENT-5
LALITHA MATHI; UFID-29341966





SI.OUT FILE:

@(#)$CDS: LVS.exe version 5.1.0 01/27/2011 00:12 (cicln04) $

Command line: /home/apps/cds/ic5141/tools.lnx86/dfII/bin/32bit/LVS.exe -dir /home/users/1/mathi/VLSI/PAR/LVS -l -s -t /home/users/1/mathi/VLSI/PAR/LVS/layout /home/users/1/mathi/VLSI/PAR/LVS/schematic
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.

Net-list summary for /home/users/1/mathi/VLSI/PAR/LVS/layout/netlist

count	
331	nets
0	terminals
303	pmos
303	nmos

Net-list summary for /home/users/1/mathi/VLSI/PAR/LVS/schematic/netlist

ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

count	
331	nets
14	terminals
303	pmos
303	nmos

Devices in the rules but not in the netlist:
cap nfet pfet nmos4 pmos4

The net-lists match.

	layout	schematic
	instances	
un-matched	0	0
rewired	0	0
size errors	0	0
pruned	0	0
active	606	606
total	606	606
	nets	
un-matched	0	0
merged	0	0
pruned	0	0
active	331	331
total	331	331
	terminals	
un-matched	0	0
matched but different type	0	0
total	0	14

Probe files from /home/users/1/mathi/VLSI/PAR/LVS/schematic

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:

prunedev.out:

audit.out:

ASSIGNMENT-5
LALITHA MATHI; UFID-29341966

Probe files from /home/users/1/mathi/VLSI/PAR/LVS/layout

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:

prunedev.out:

audit.out: