PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

'Elevator Controller'

Submitted by

Keertiraj K V (PES1UG21EC129)
Nishita N Joshi (PES1UG21EC170)
P Anurup (PES1UG21EC906)

August - December 2023

For the partial fulfillment of course Digital System Design – UE21EC342AB1

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
PES UNIVERSITY
BENGALURU - 560 085

FACULTY OF ENGINEERING
PROGRAM - B.TECH

Table of contents

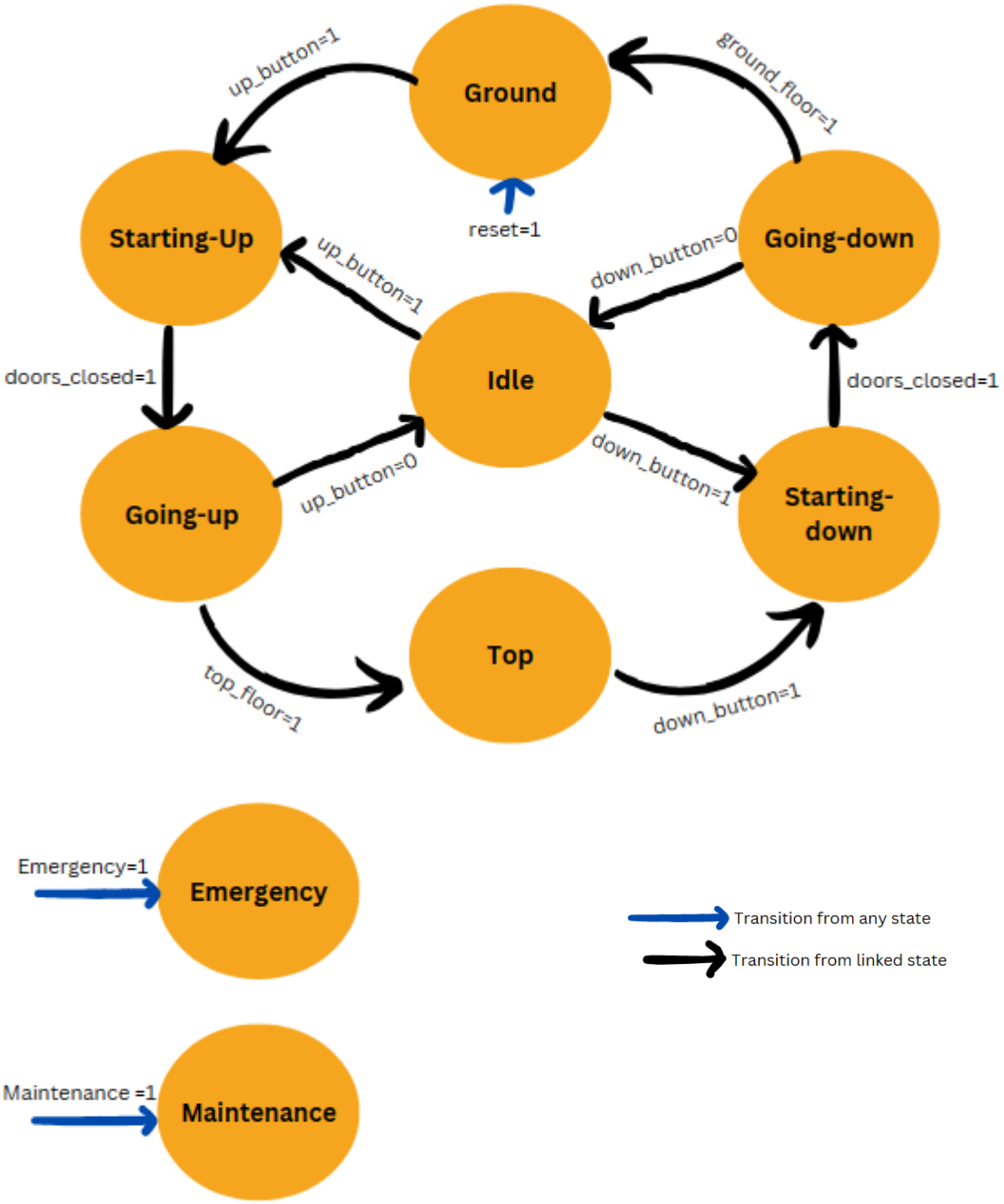
1. Introduction

Elevators, integral to modern urban life, have evolved from simple mechanical systems to sophisticated, digitally-driven entities. This project zeroes in on the software facet of elevator control, steering away from traditional relay-based mechanisms in favor of digital advancements. Through the development of a meticulous design and testbench file, our focus extends from initial design to the synthesis phase, utilizing Cadence tools to generate critical metrics such as area and power reports.

The objective is clear: to seamlessly integrate advanced software solutions into the elevator control domain. By delving into the intricacies of software design and synthesis, this project aligns with the trajectory of elevators as intelligent, digital entities poised to redefine standards in vertical transportation. The synthesis phase, facilitated by Cadence tools, not only provides quantitative insights into system performance but also serves as a compass for optimization and refinement.

Our aspiration is to contribute to the broader narrative of digital innovation in urban infrastructure, where precision, efficiency, and safety converge. As we navigate the synthesis process, our project becomes a stepping stone towards elevators that are not merely conveyors of people but intelligent, digital companions in the ever-evolving landscape of urban mobility.

2. Block Diagram



3. Code:

(i)Design code:

```
module lift_con(input wire clk, reset, up_button, down_button, doors_closed,
top_floor, ground_floor, emergency, maintenance,output reg [3:0] state,
output reg motor_on, output reg motor_direction, output reg fan_on);

  // State encoding
parameter Ground = 4'b0000, Starting_up = 4'b0001,
Going_up = 4'b0010, Idle = 4'b0011, Top = 4'b0100,
Starting_down = 4'b0101, Going_down = 4'b0110,
Emergency = 4'b0111, Maintenance = 4'b1000;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      state <= Ground;
      motor_on <= 0;
      fan_on <= 0;
    end
    else begin
      case (state)
        Ground: begin
          if (up_button) begin
            state <= Starting_up;
            motor_on <= 1;
            motor_direction <= 1; // Upward
            fan_on <= 1;
          end
          else state <= Ground;
        end
        Starting_up: begin
          if (doors_closed) state <= Going_up;
          else state <= Starting_up;
        end
```

```
        Going_up: begin
          if (top_floor) begin
            state <= Top;
            motor_on <= 0;
          end
          else if (!up_button) begin
            state <= Idle;
            motor_on <= 0;
          end
          else state <= Going_up;
        end
        Idle: begin
          if (up_button) begin
            state <= Starting_up;
            motor_on <= 1;
            motor_direction <= 1; // Upward
            fan_on <= 1;
          end
          else if (down_button) begin
            state <= Starting_down;
            motor_on <= 1;
            motor_direction <= 0; // Downward
            fan_on <= 1;
          end
          else state <= Idle;
        end
        Top: begin
          if (down_button) begin
            state <= Starting_down;
            motor_on <= 1;
            motor_direction <= 0; // Downward
            fan_on <= 1;
```

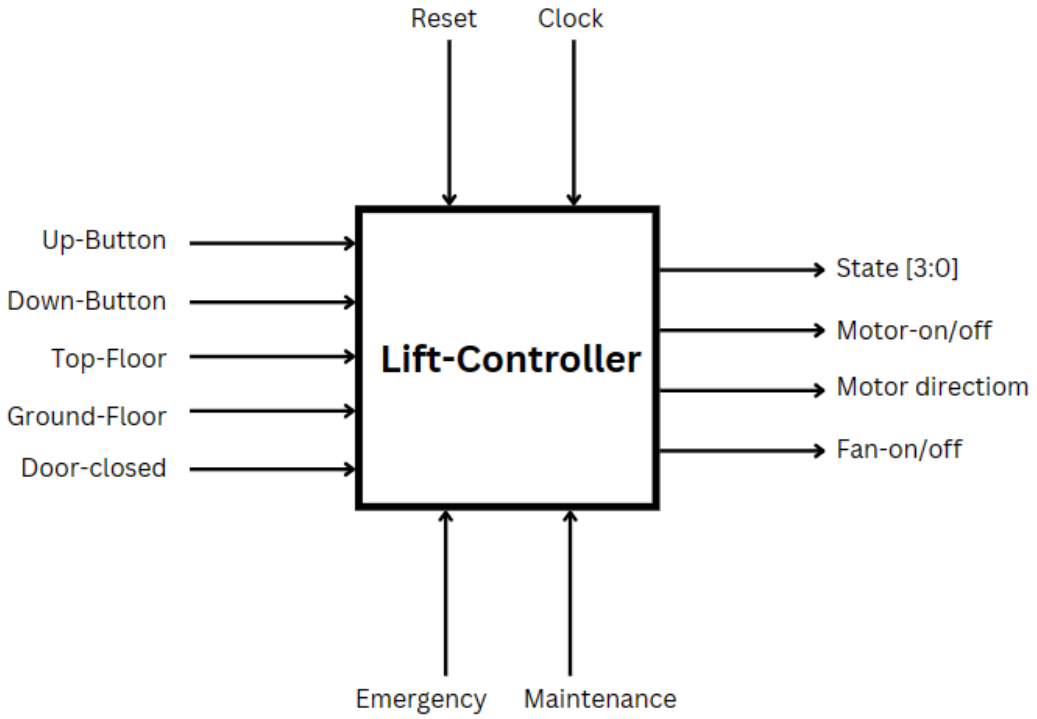
```
        end
        Starting_down: begin
          if (doors_closed) state <= Going_down;
          else state <= Starting_down;
        end
        Going_down: begin
          if (ground_floor) begin
            state <= Ground;
            motor_on <= 0;
          end
          else if (!down_button) begin
            state <= Idle;
            motor_on <= 0;
          end
          else state <= Going_down;
        end
        Emergency: begin
          state <= Ground;
          motor_on <= 0;
          fan_on <= 0;
        end
        Maintenance: begin
          state <= Maintenance;
          motor_on <= 0;
          fan_on <= 0;
        end
      endcase
    end
    if (emergency) state <= Emergency;
    if (maintenance) state <= Maintenance;
  end
endmodule
```

(ii)Test bench:

```
module lift_con_tb;
    reg clk;
    reg reset;
    reg up_button;
    reg down_button;
    reg doors_closed;
    reg top_floor;
    reg ground_floor;
    reg emergency;
    reg maintenance;

    wire [3:0] state;
    wire motor_on;
    wire motor_direction;
    wire fan_on;

    // Instantiate the FSM
    lift_con u1 (
        .clk(clk), .reset(reset), .up_button(up_button), .down_button(down_button),
        .doors_closed(doors_closed),.top_floor(top_floor), .ground_floor(ground_floor),
        .emergency(emergency), .maintenance(maintenance),.state(state),
        .motor_on(motor_on), .motor_direction(motor_direction), .fan_on(fan_on)
    );

    // Clock generator
    always begin
        #1 clk = 0;
        #5 clk = ~clk;
    end
        always #2 reset <= $random;
        always #2 up_button <= $random;
        always #2 down_button <= $random;
        always #2 doors_closed <= $random;
        always #2 top_floor <= $random;
        always #2 ground_floor <= $random;
        always #2 emergency <= $random;
        always #2 maintenance <= $random;

endmodule
```

Code Explanation:

- Firstly we define all the inputs and outputs present in our design file “lift_con.v” The inputs and outputs as follows:



- We have given each state a separate 4 bit encoding. We have 9 states so we represented from 4’b0000 to 4’b1000.
- Now using an asynchronous reset we start our FSM:
    1) Encounters reset being high => immediately goes to Ground state. Internally motor and fan are turned off
    2) If reset is not high it goes through following cases:
        a) Lift is presently in ground state now if the lift encounters a push button of **“up_button”** to be high it quickly shifts its state to => Starting_up. Internally the motor is turned on and fan is turned on and so is the motor_direction is on.
        b) Lift is presently in Starting_up state let’s say. If the lift encounters **“doors_closed”** is high i.e, doors getting shut It changes its state to => Going_up.
        c) Lift is presently in Going_up state. For the sake of covering all possibilities let’s say lift reaches the top

floor, so if **“top_floor”** is set to high the lift moves to a state=>Top. Motor_on is reset to 0. This is basically the extreme case.

Now if **“up_button” = 0** it will go to Idle state.

d) Now the Lift has reached the Idle state. That is it stays in the state wherever it is. Now if **“up_button”** is set to 1, the state starts to move upwards. State=>Starting_up. Internally the motor is set to 1 and motor direction is also high (upward direction) and the fan is also turned on.

   Now if **“down_button”** is set to 1, the state starts to move upwards. State=>Starting_down. Internally the motor is set to 1 and motor direction is reset to 0 or is low (downward direction) and the fan is also turned on.

e) Lift is now encountering Top state.It has reached to the extreme floor. Now if it senses **“down_button”** as 1 state it should go to => Starting_down, the motor has started and the direction is downward. Fan is always on anyway except for Ground.

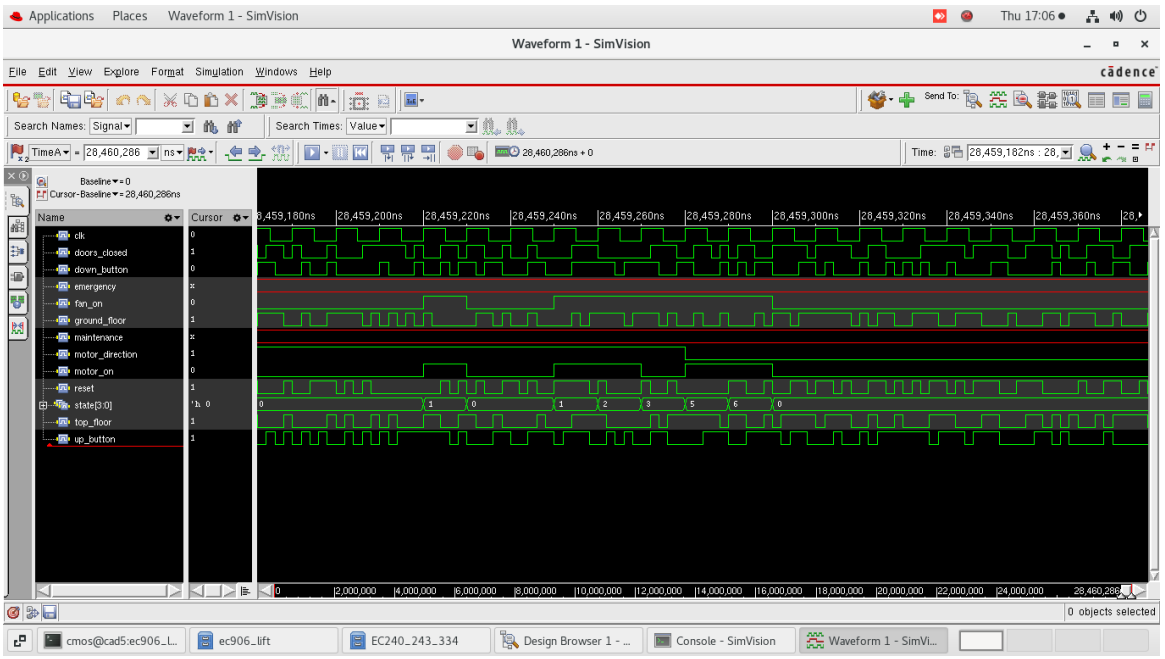
f) Once the lift is in Starting_down state it is in motion it has just come to that stage, that means it should close the doors.
   If it senses **“doors_closed”** as high the state that should be updated to is Going_down state.

g) In the Going_down state again it should check for the most extreme case which is the ground_floor. If that **“ground_floor”** is set to 1 it goes to ground state and switches off the motor.

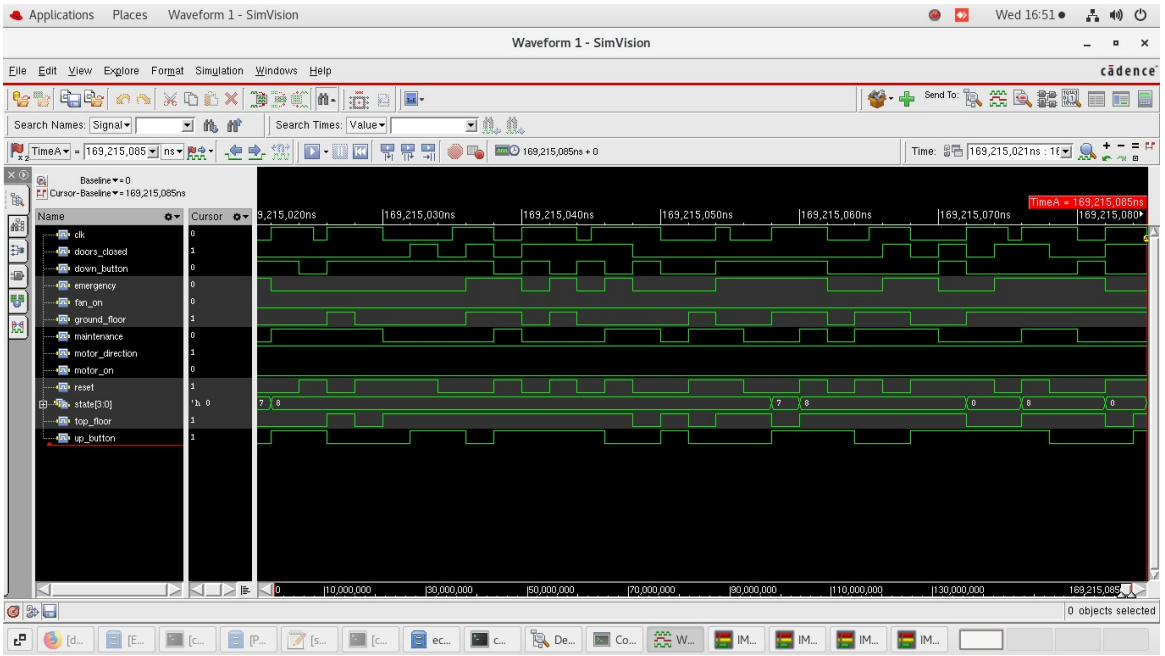
h) There are 2 more States which are Emergency and Maintenance. These 2 have separate inputs to trigger them and Emergency state is when encountered the entire code rushes to Ground state.

4. Schematic and Simulation Results

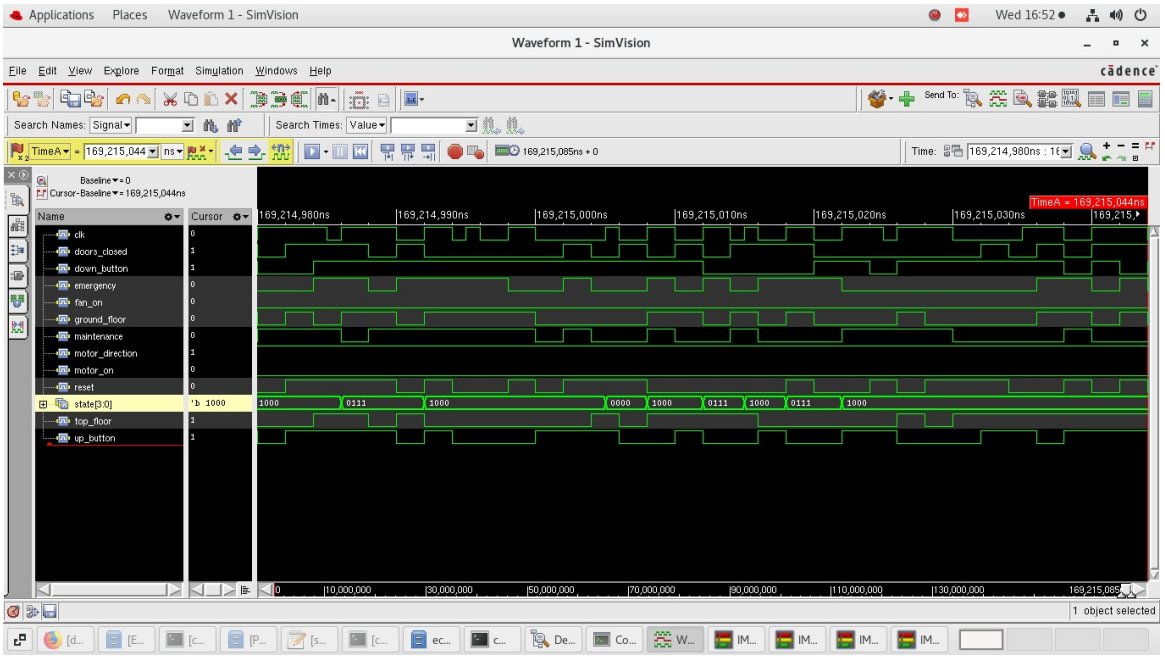
(i) Simulation Result:



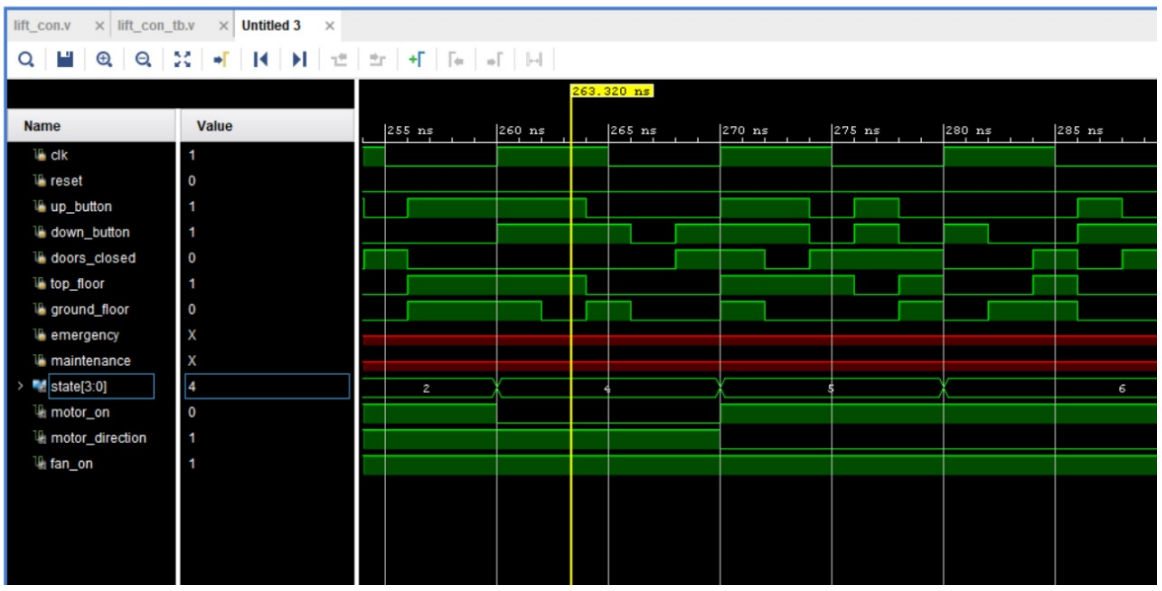
[Figure 1]



[Figure 2]



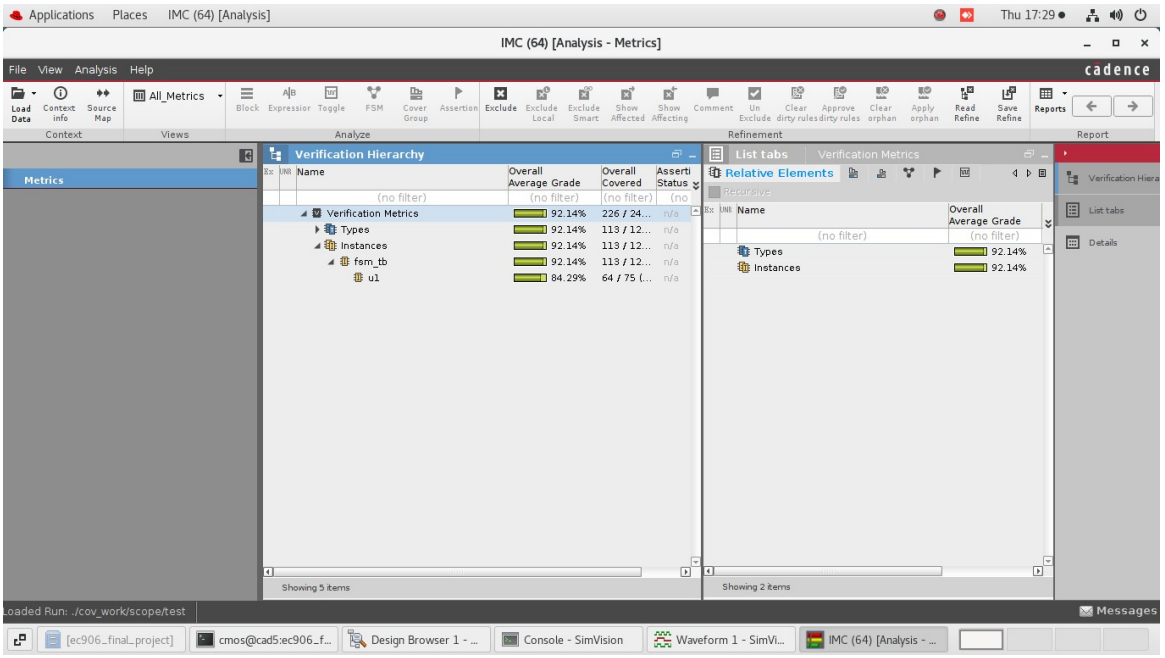
[Figure 3]



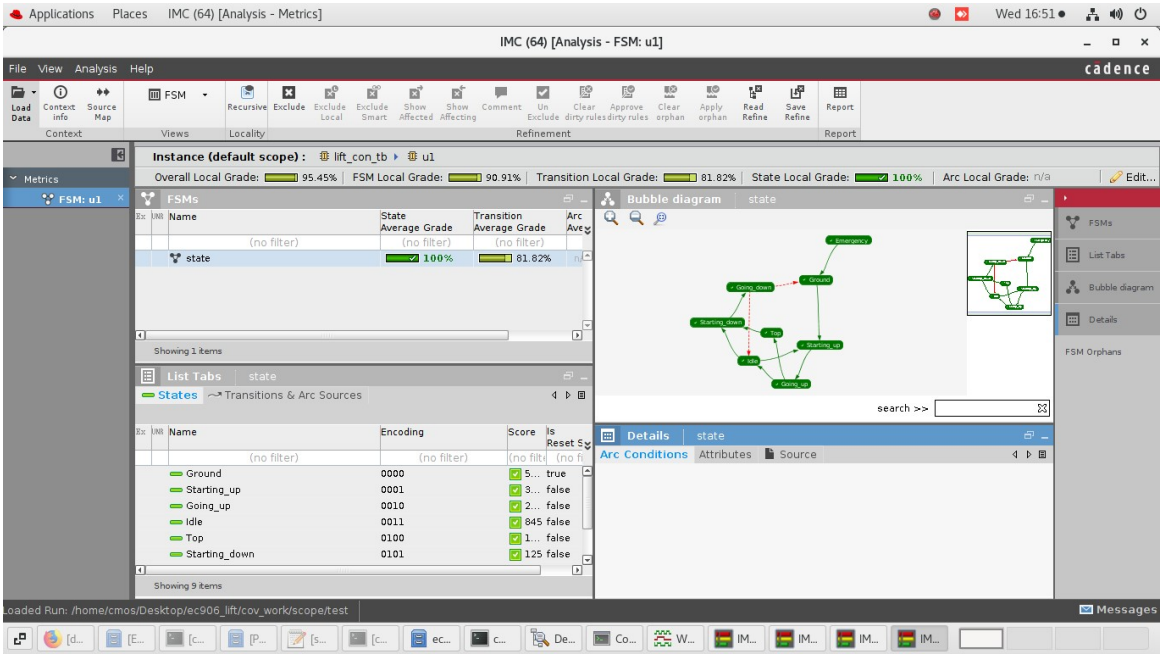
[Figure 4]

Explanation of Simulation graph:
For example, if it is in state 4 (Top):
It will go to Top state if top_floor = 1, reset = 0 and clk = 1.

(ii) Code Coverage:



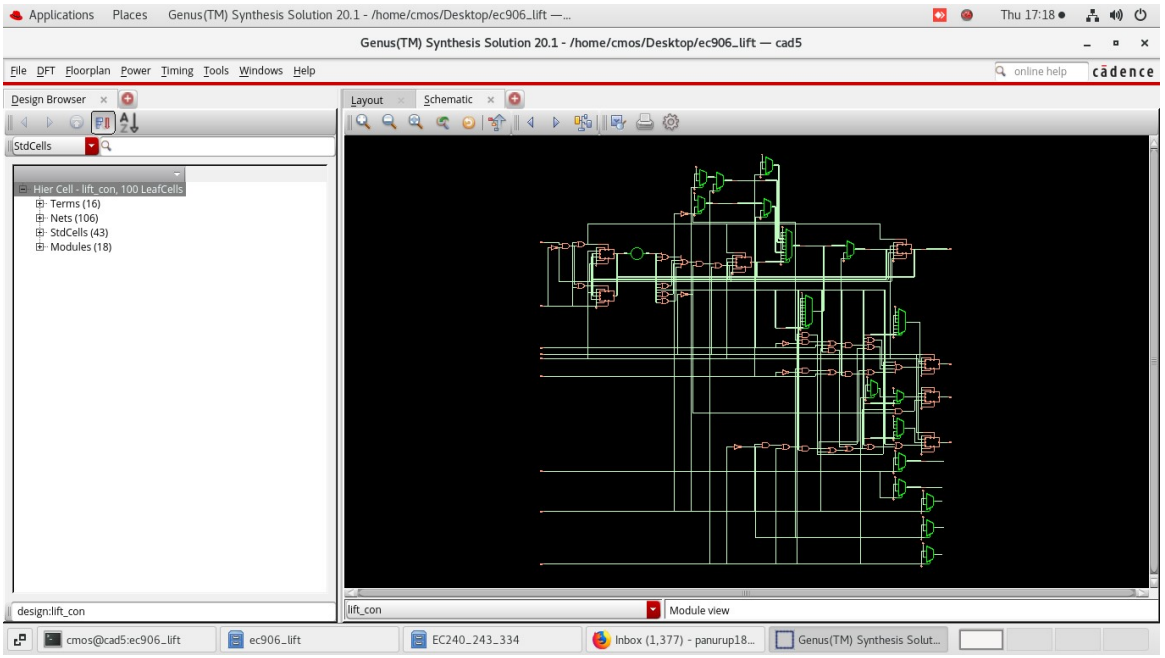
[Figure 5]



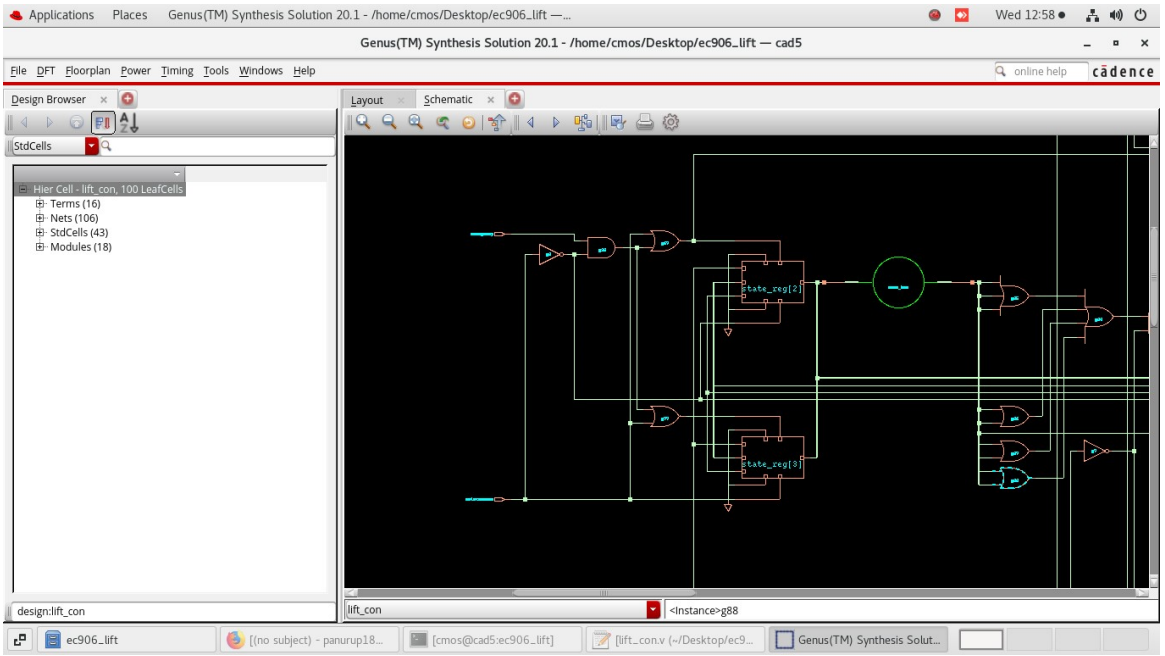
[Figure 6]

- We have got a code coverage of 92% and the FSM diagram has been shown.
- All the states in the FSM have been covered.

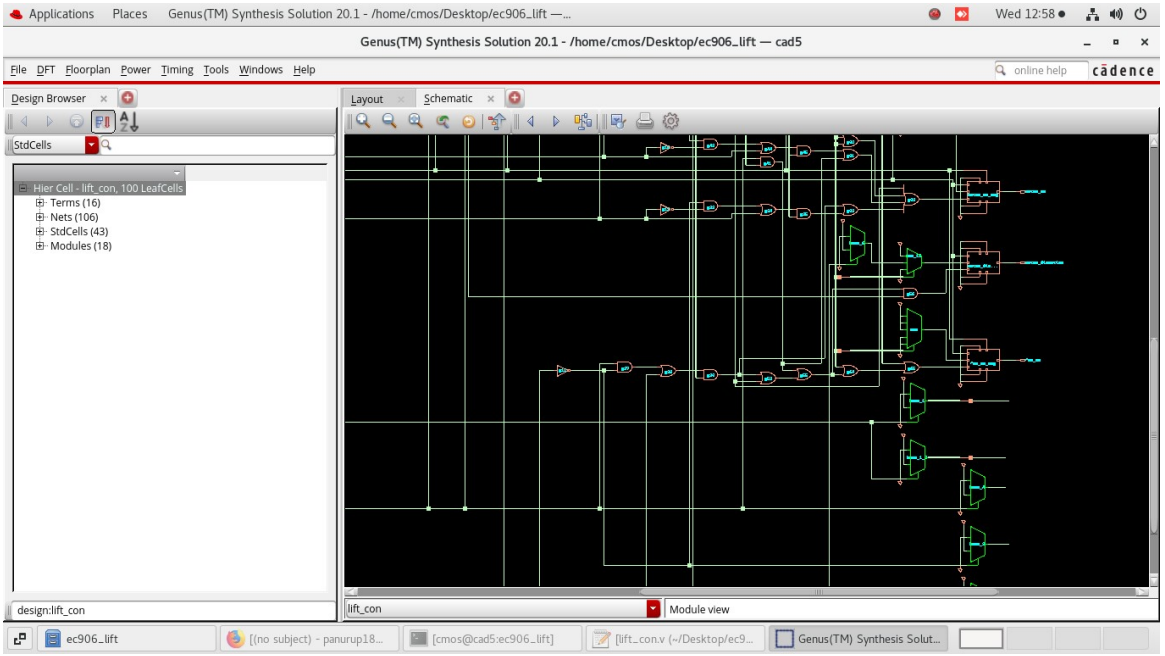
(iii) Synthesis:



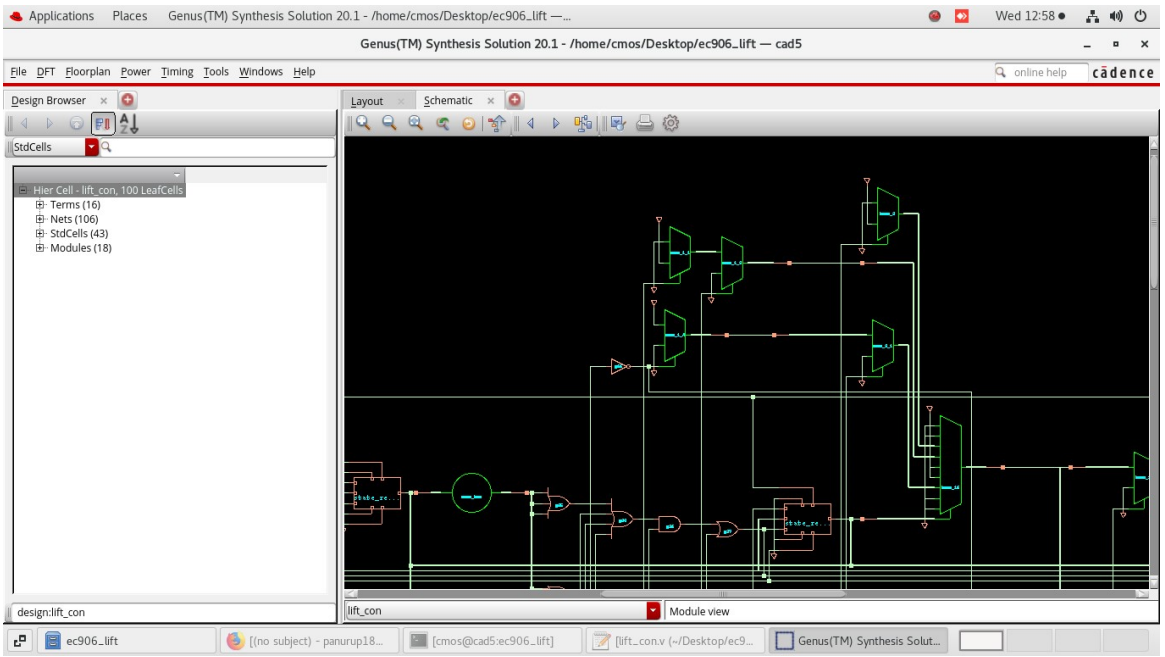
[Figure 7]



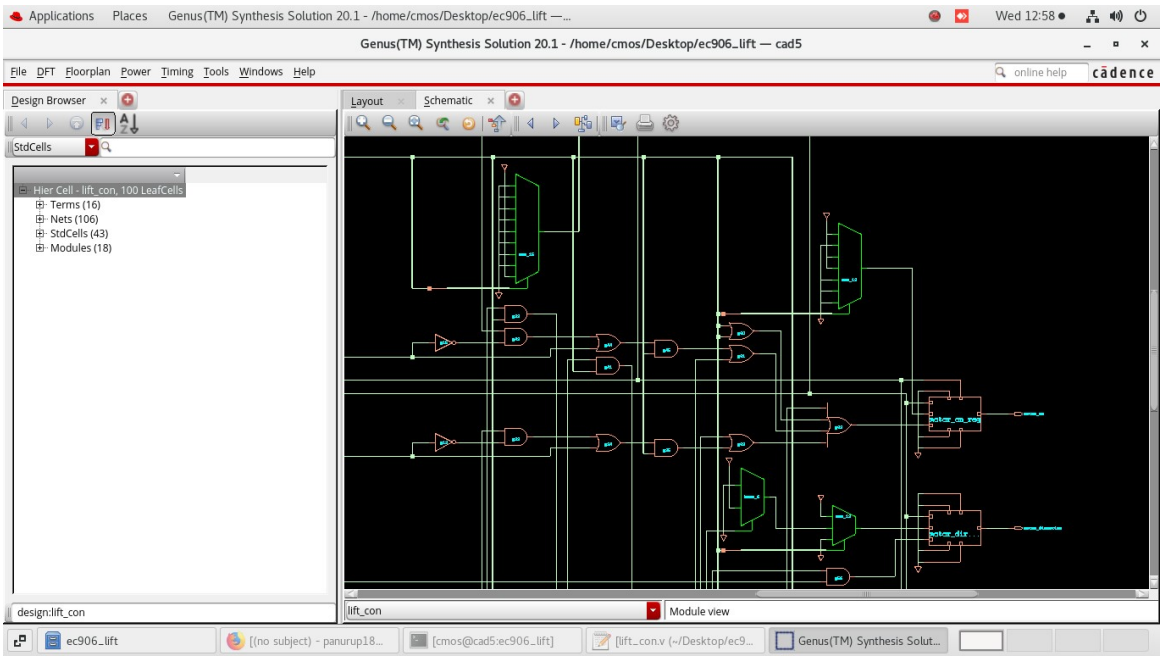
[Figure 8]



[Figure 9]

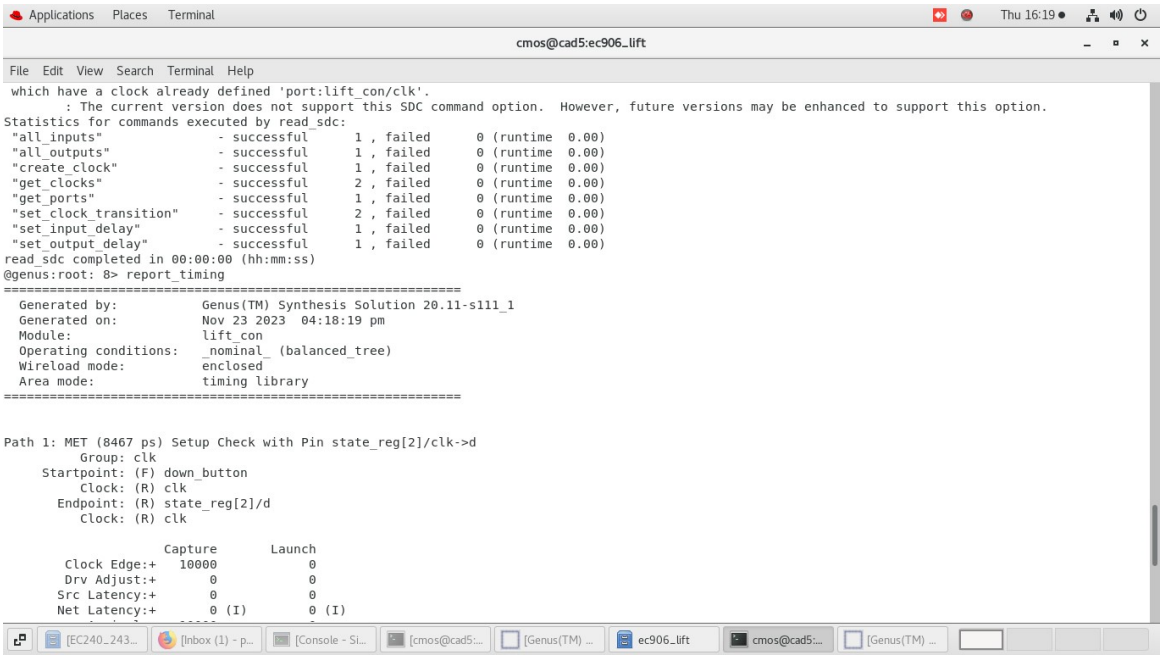


[Figure 10]

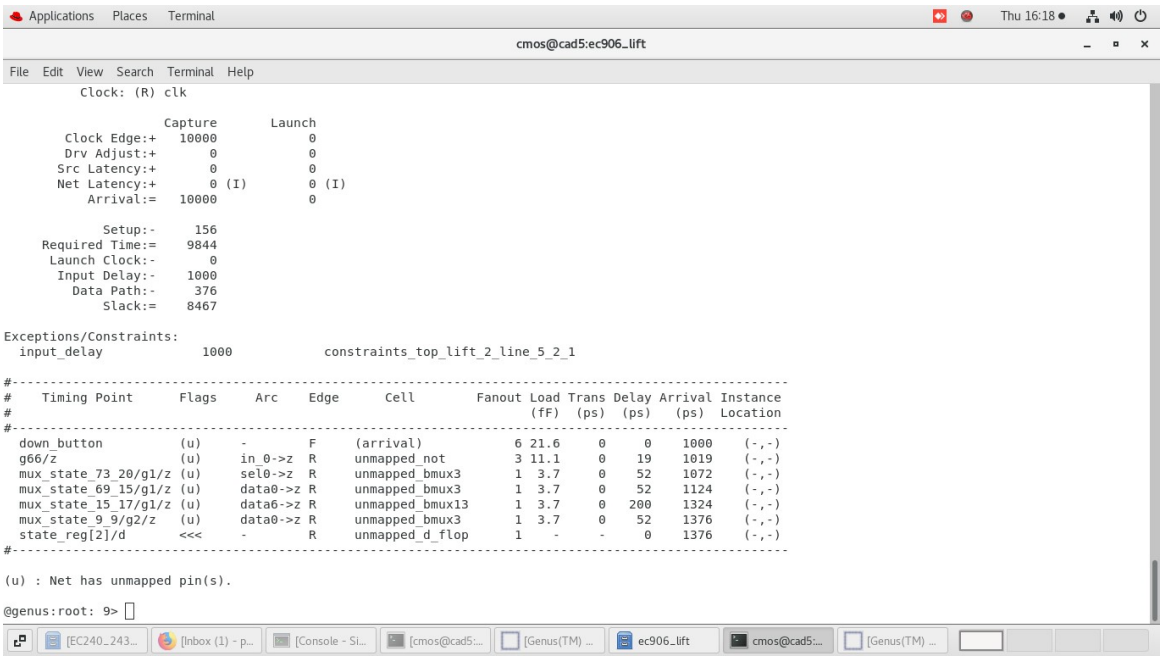


[Figure 11]

(iv) Timing report:



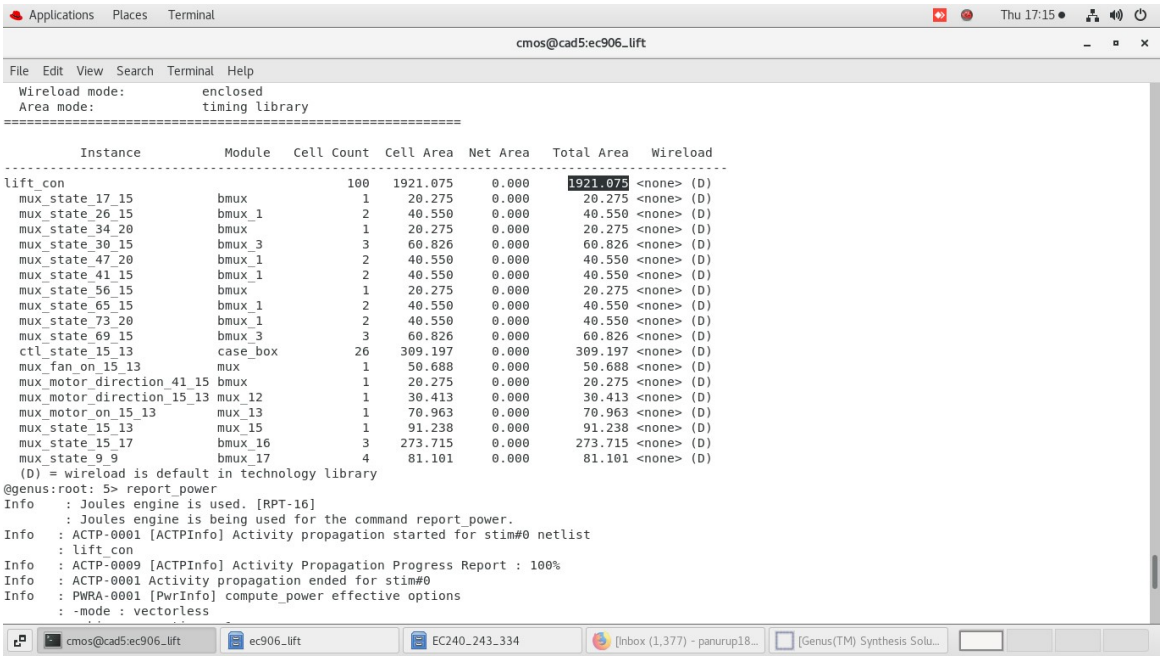
[Figure 12]



[Figure 13]

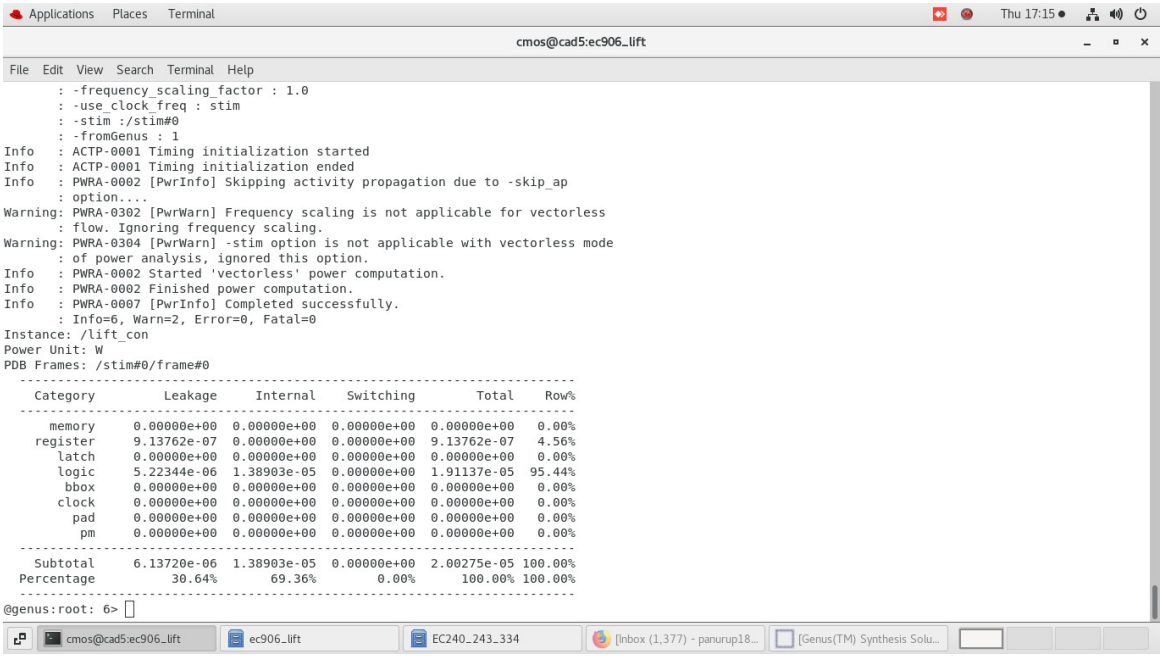
- Met the timing constraints with the constraints given as clk transition for rise as 0.1 and fall transition as 0.1 as well.
- Maximum input delay given as 0.1 and maximum output delay also given as 0.1ns
- Clk period given as 10ns
- Slack produced is 8467ps

(v) Area and Power report:



[Figure 14]

The different combinational blocks of mux taken from the library have the areas as listed above and they all add up to the final area of the lift controller = 1921.875 units

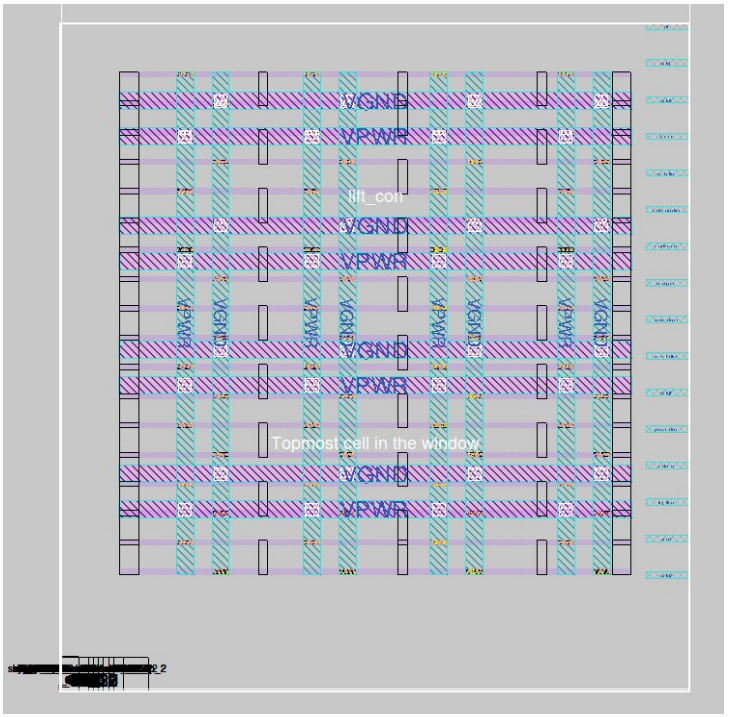


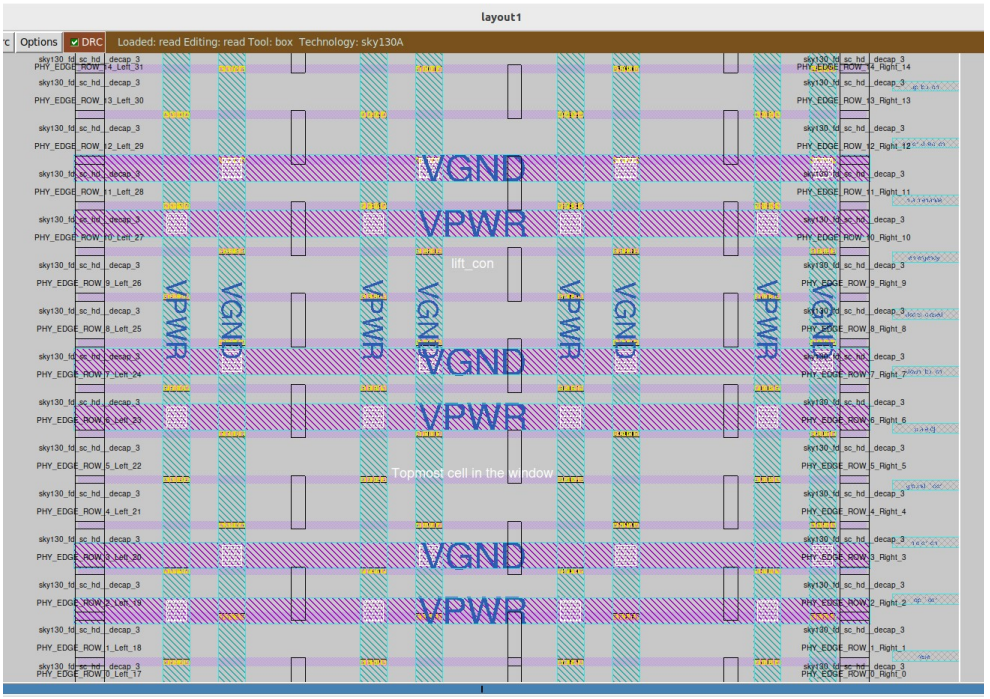
[Figure 15]

The different memory, register and logic units draw the power as listed above. They all add up to the subtotal of 2.00275e-05 Watts.

5. Physical Design:

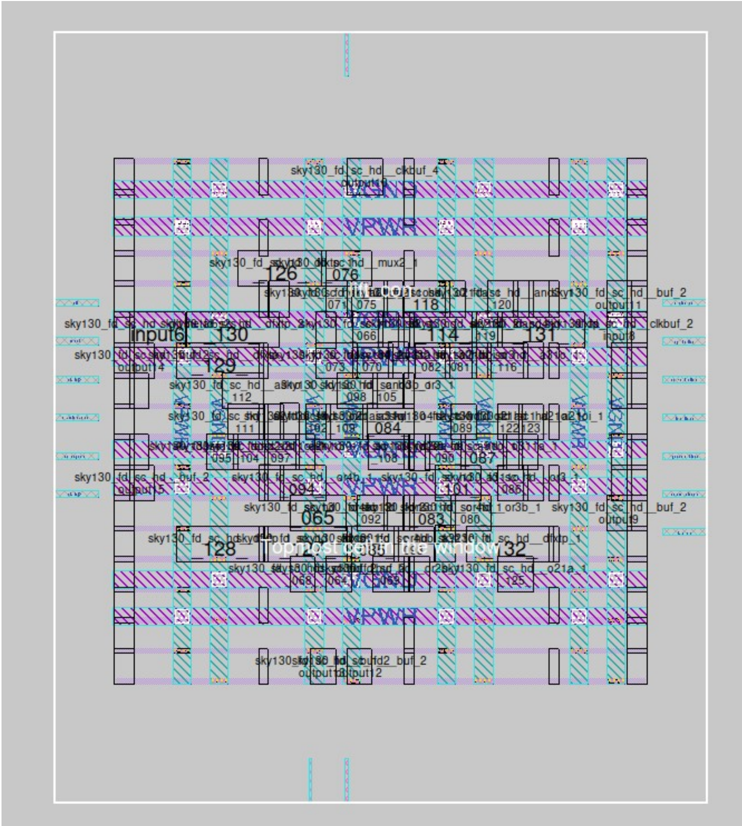
FLOORPLAN:

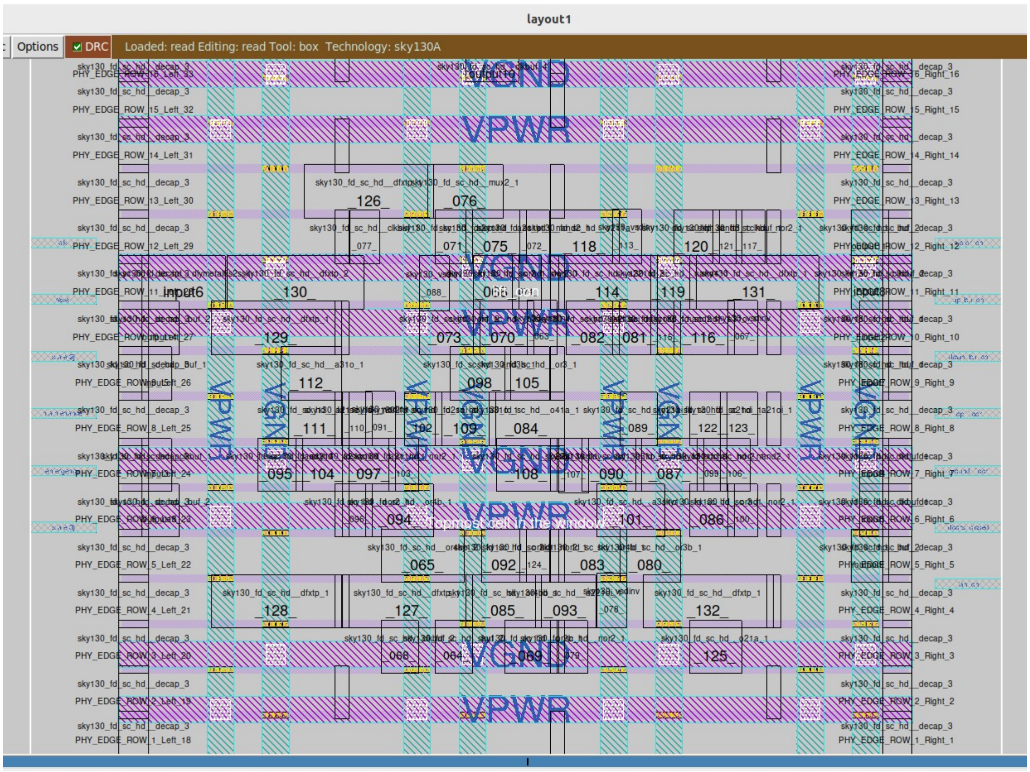




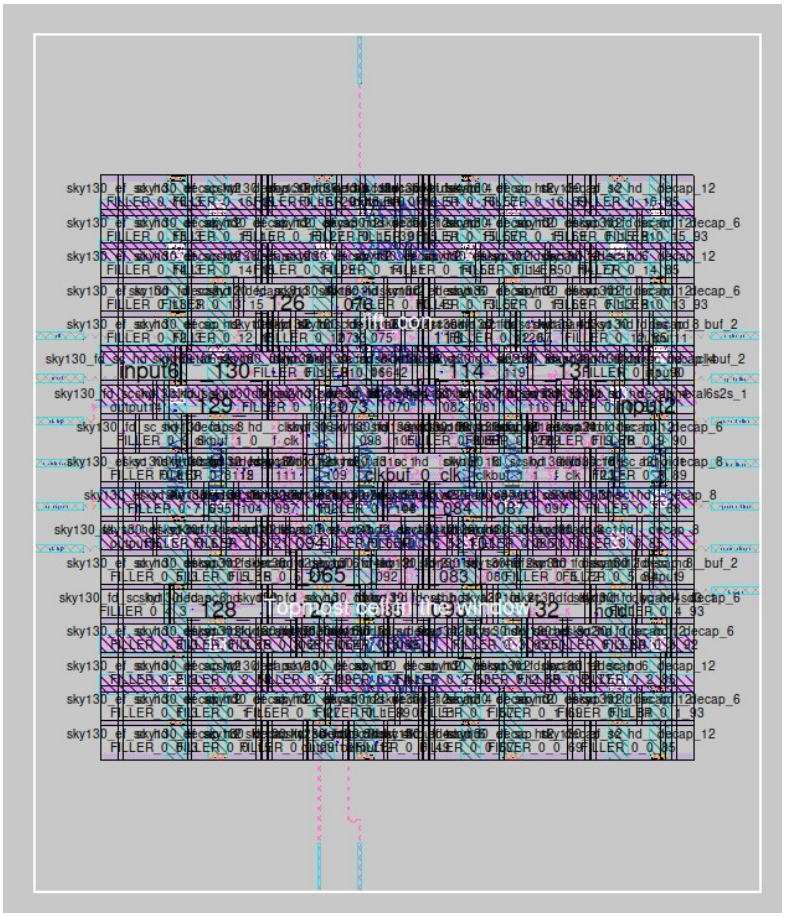
- Defines the chip's physical layout by specifying the core, periphery, and macroblock areas.
- Allocate space for standard cells, I/O pads, and other component

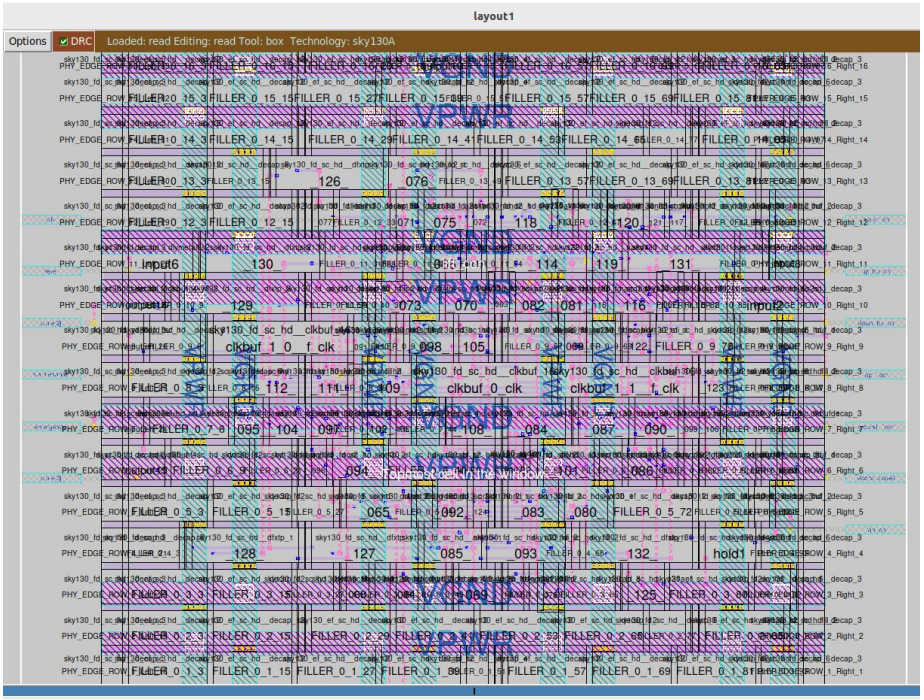
PLACEMENT:


VPWR
VGND
input6
126
130
129
128
065
132

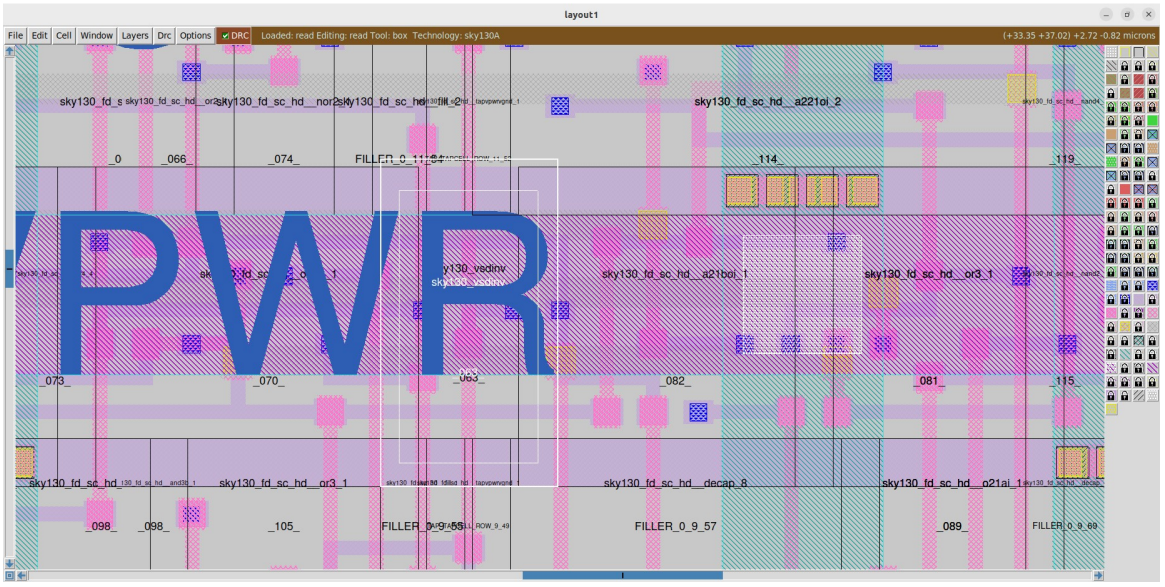

layout1
Options
DRC
Loaded: read Editing: read Tool: box Technology: sky130A
VGND
VPWR
126
076
130
129
112
128
127
132
125

ROUTING:




layout1
Options
DRC
Loaded: read Editing: read Tool: box Technology: sky130A

CELL INSTANCE:



(i) Area and Power report:

```
1271 area_report
1272
1273 ========================================
1274 report_design_area
1275 ========================================
1276 Design area 746 u^2 34% utilization.
1277 area_report_end
```

```
1217 power_report
1218
1219 ===========================================================================
1220  report_power
1221 ===========================================================================
1222 ======================== Typical Corner ===================================
1223
1224 Group                  Internal  Switching    Leakage      Total
1225                            Power      Power      Power      Power (Watts)
1226 ----------------------------------------------------------------
1227 Sequential             3.22e-05   3.27e-06   5.91e-11   3.55e-05  26.9%
1228 Combinational          6.58e-05   3.08e-05   3.74e-10   9.66e-05  73.1%
1229 Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
1230 Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
1231 ----------------------------------------------------------------
1232 Total                  9.81e-05   3.40e-05   4.33e-10   1.32e-04 100.0%
1233                           74.2%      25.8%       0.0%
1234
1235 power_report_end
```

```
61. Printing statistics.

=== lift_con ===

   Number of wires:                 76
   Number of wire bits:             79
   Number of public wires:          13
   Number of public wire bits:      16
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 70
     sky130_fd_sc_hd__a211o_2        1
     sky130_fd_sc_hd__a211oi_2       3
     sky130_fd_sc_hd__a21boi_2       1
     sky130_fd_sc_hd__a21oi_2        3
     sky130_fd_sc_hd__a221oi_2       2
     sky130_fd_sc_hd__a311o_2        1
     sky130_fd_sc_hd__a31o_2         4
     sky130_fd_sc_hd__and2_2         1
     sky130_fd_sc_hd__and3_2         1
     sky130_fd_sc_hd__and3b_2        1
     sky130_fd_sc_hd__buf_1          4
     sky130_fd_sc_hd__dfxtp_2        7
     sky130_fd_sc_hd__mux2_2         1
     sky130_fd_sc_hd__nand2_2        3
     sky130_fd_sc_hd__nand4_2        1
     sky130_fd_sc_hd__nor2_2         9
     sky130_fd_sc_hd__o211a_2        1
     sky130_fd_sc_hd__o21a_2         1
     sky130_fd_sc_hd__o21ai_2        2
     sky130_fd_sc_hd__o22a_2         1
     sky130_fd_sc_hd__o311a_2        1
     sky130_fd_sc_hd__o31a_2         1
     sky130_fd_sc_hd__o41a_2         1
     sky130_fd_sc_hd__or2_2          2
     sky130_fd_sc_hd__or2b_2         1
     sky130_fd_sc_hd__or3_2          3
     sky130_fd_sc_hd__or3b_2         1
     sky130_fd_sc_hd__or4_2          1
     sky130_fd_sc_hd__or4b_2         2
     sky130_fd_sc_hd__or4bb_2        2
     sky130_vsdinv                   7

   Chip area for module '\lift_con': 661.884800
```

[Figure 16]

(ii) Timing report:

```
1236 skew_report
1237
1238 ===========================================================================
1239 report_clock_skew
1240 ===========================================================================
1241 Clock clk
1242 Latency      CRPR       Skew
1243 _127_/CLK ^
1244    0.27
1245 _131_/CLK ^
1246    0.24      0.00       0.03
1247
1248 skew_report_end
1249 summary_report
1250
1251 ===========================================================================
1252 report_tns
1253 ===========================================================================
1254 tns 0.00
1255
1256 ===========================================================================
1257 report_wns
1258 ===========================================================================
1259 wns 0.00
1260
1261 ===========================================================================
1262 report_worst_slack -max (Setup)
1263 ===========================================================================
1264 worst slack 6.43
1265
1266 ===========================================================================
1267 report_worst_slack -min (Hold)
1268 ===========================================================================
1269 worst slack 0.21
1270 summary_report_end
```

```
79 max_report
80
81 ===========================================================================
82 report_checks -path_delay max (Setup)
83 ===========================================================================
84 ======================== Typical Corner ===================================
85
86 Startpoint: emergency (input port clocked by clk)
87 Endpoint: _127_ (rising edge-triggered flip-flop clocked by clk)
88 Path Group: clk
89 Path Type: max
90
91 Fanout     Cap    Slew   Delay    Time   Description
92 ---------------------------------------------------------------------------
93                   0.15    0.00    0.00   clock clk (rise edge)
94                           0.00    0.00   clock network delay (ideal)
95                           2.00    2.00 v input external delay
96      2    0.00    0.01    0.01    2.01 v emergency (in)
97                                           emergency (net)
98                   0.01    0.00    2.01 v _094_/A (sky130_fd_sc_hd__or4b_2)
99      4    0.01    0.15    0.76    2.77 v _094_/X (sky130_fd_sc_hd__or4b_2)
00                                           _037_ (net)
01                   0.15    0.00    2.77 v _095_/B (sky130_fd_sc_hd__and2_2)
02      3    0.01    0.06    0.28    3.04 v _095_/X (sky130_fd_sc_hd__and2_2)
03                                           _038_ (net)
04                   0.06    0.00    3.04 v _097_/A2 (sky130_fd_sc_hd__a211oi_2)
05      1    0.00    0.13    0.20    3.24 ^ _097_/Y (sky130_fd_sc_hd__a211oi_2)
06                                           _001_ (net)
07                   0.13    0.00    3.24 ^ _127_/D (sky130_fd_sc_hd__dfxtp_2)
08                           3.24    3.24   data arrival time
09
10                   0.15   10.00   10.00   clock clk (rise edge)
11                           0.00   10.00   clock network delay (ideal)
12                          -0.25    9.75   clock uncertainty
13                           0.00    9.75   clock reconvergence pessimism
14                                   9.75 ^ _127_/CLK (sky130_fd_sc_hd__dfxtp_2)
15                          -0.06    9.69   library setup time
16                                   9.69   data required time
17 ---------------------------------------------------------------------------
18                                   9.69   data required time
19                                  -3.24   data arrival time
20 ---------------------------------------------------------------------------
21                                   6.44   slack (MET)
```

[Figure 17]

6. References

- Wikipedia
- Github
- Youtube videos
- Teacher
- Friends

SCAN THE ABOVE CODE TO GET ALL THE FILES RELATED TO THIS PROJECT