



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

Sl.no.	Title	Page no.
1.	Introduction	3
2.	Block Diagram	4
3.	Code	5
4.	Schematic & Simulation results	9
5.	Physical Design	17
6.	References	23



1. Introduction

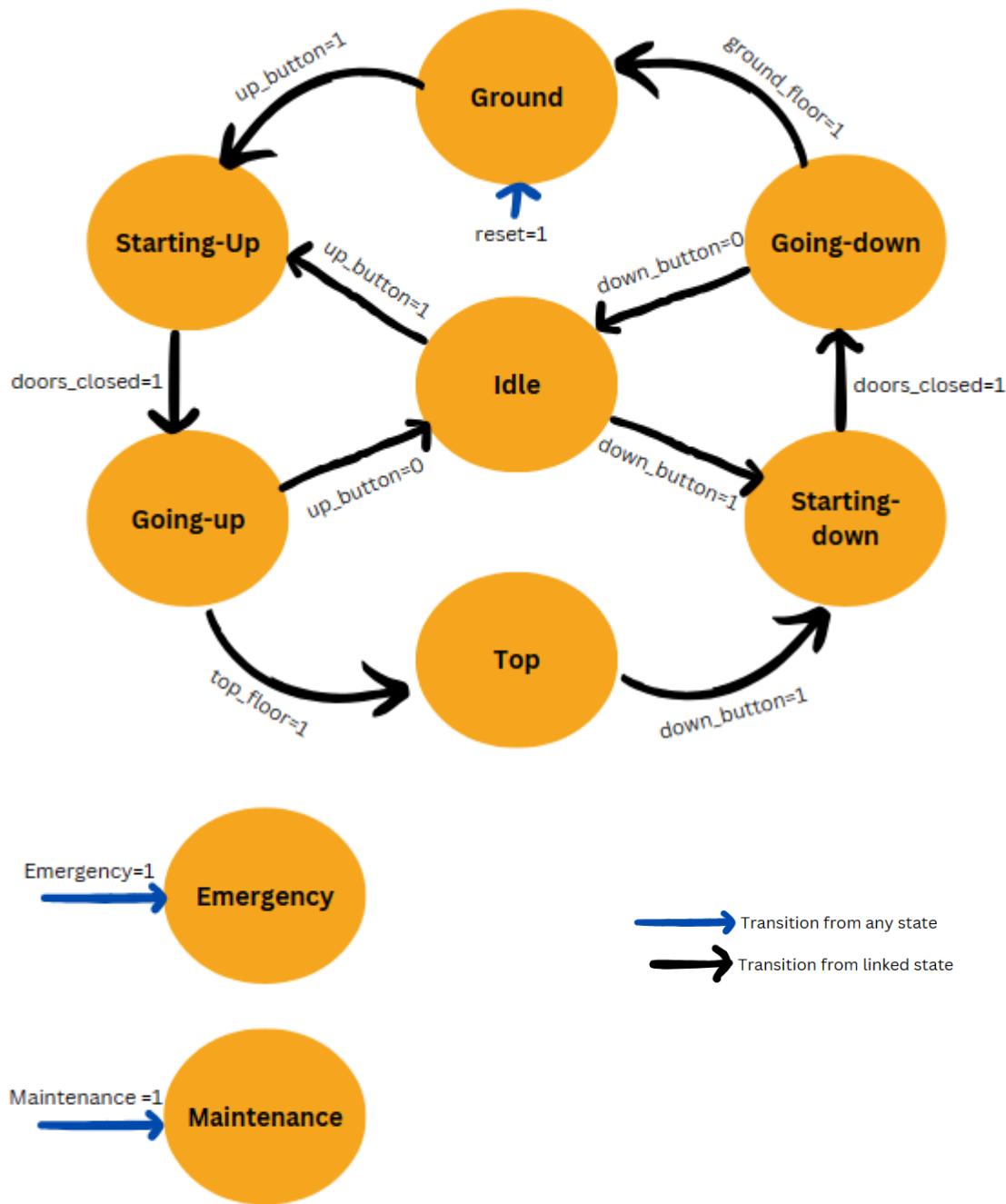
Elevators, integral to modern urban life, have evolved from simple mechanical systems to sophisticated, digitally-driven entities. This project zeroes in on the software facet of elevator control, steering away from traditional relay-based mechanisms in favor of digital advancements. Through the development 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
                end
            endcase
        end
    end
endmodule

```



```

    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;

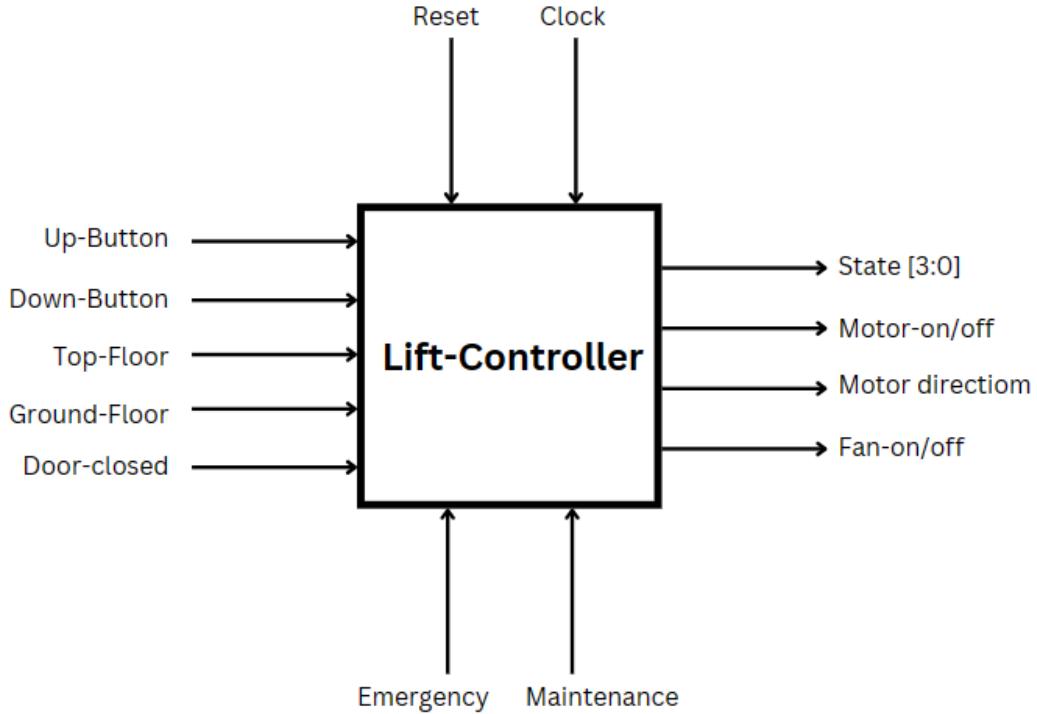
  // Instantiate the FSM
  lift_con u1 (
    .clk(clk), .reset(reset), .up_button(up_button), .down_button(down_button),
    .doors_closed(doors_closed), .top_floor(top_floor), .ground_floor(ground_floor),
    .emergency(emergency), .maintenance(maintenance), .state(state),
    .motor_on(motor_on), .motor_direction(motor_direction), .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:
 - Encounters reset being high => immediately goes to Ground state. Internally motor and fan are turned off
 - If reset is not high it goes through following cases:
 - 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.
 - 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.
 - 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.

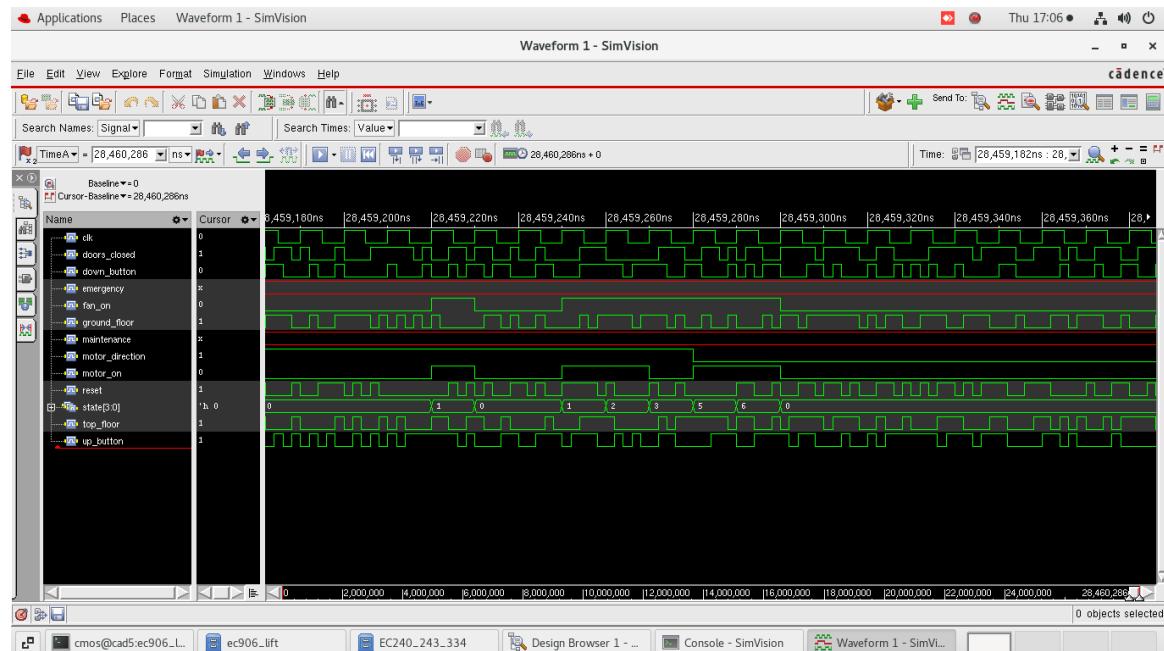
Now if “**down_button**” is set to 1, the state starts to move upwards. State=>Starting_down. Internally the motor is set to 1 and motor direction is reset to 0 or is low (downward direction) and the fan is also turned on.

- 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.

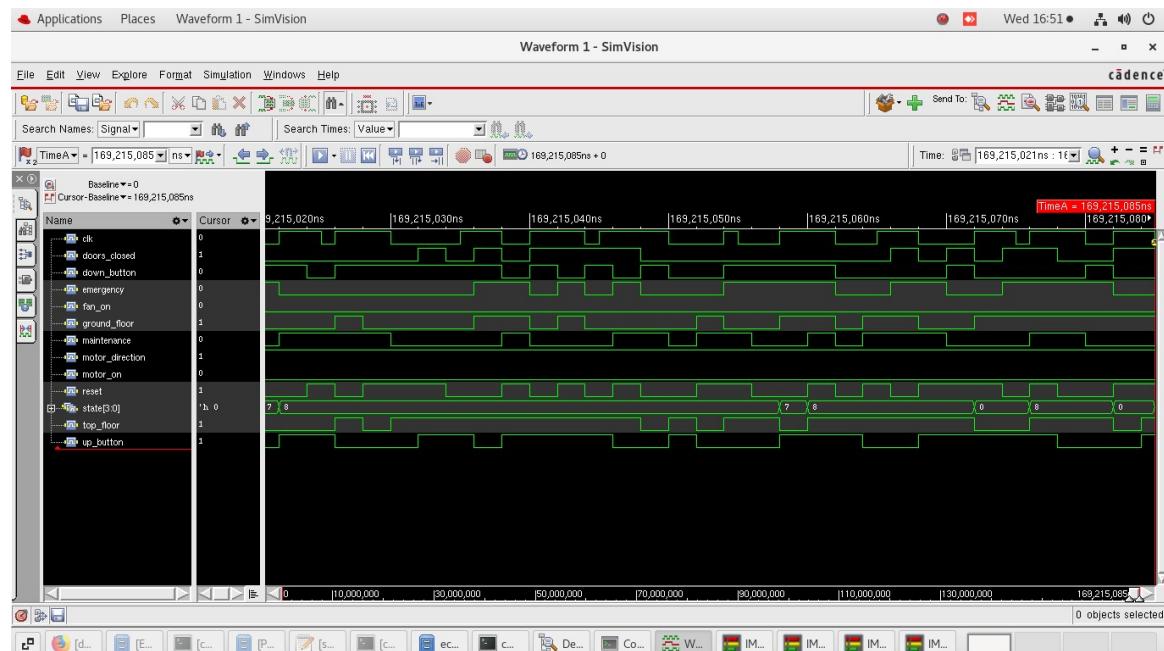


4. Schematic and Simulation Results

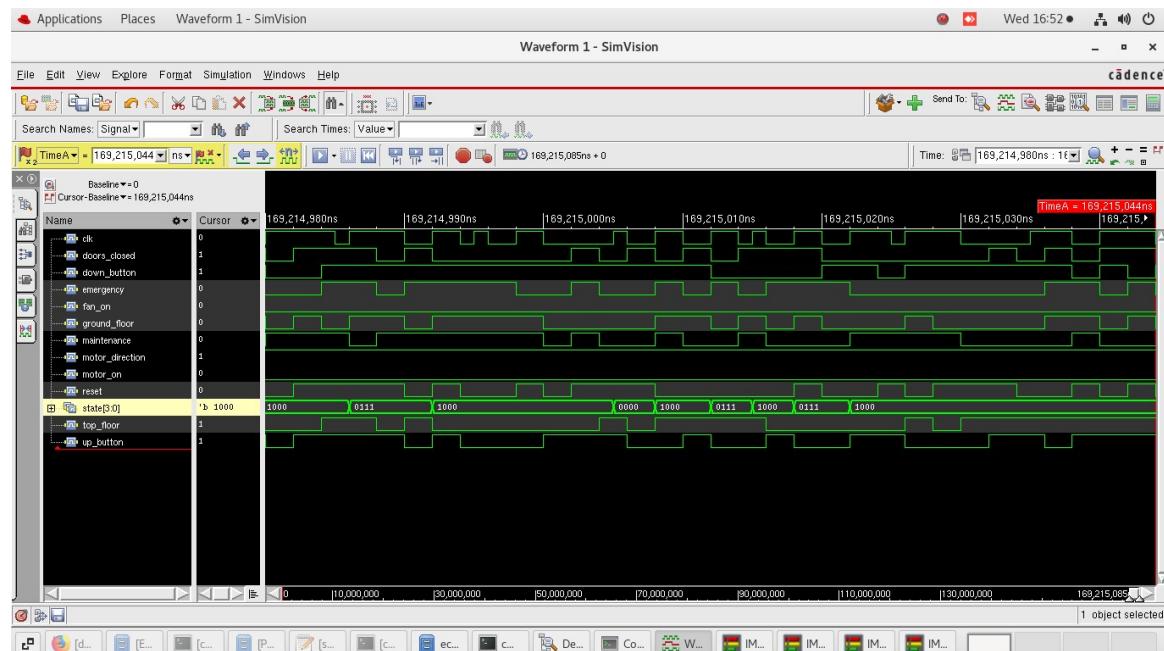
(i) Simulation Result:



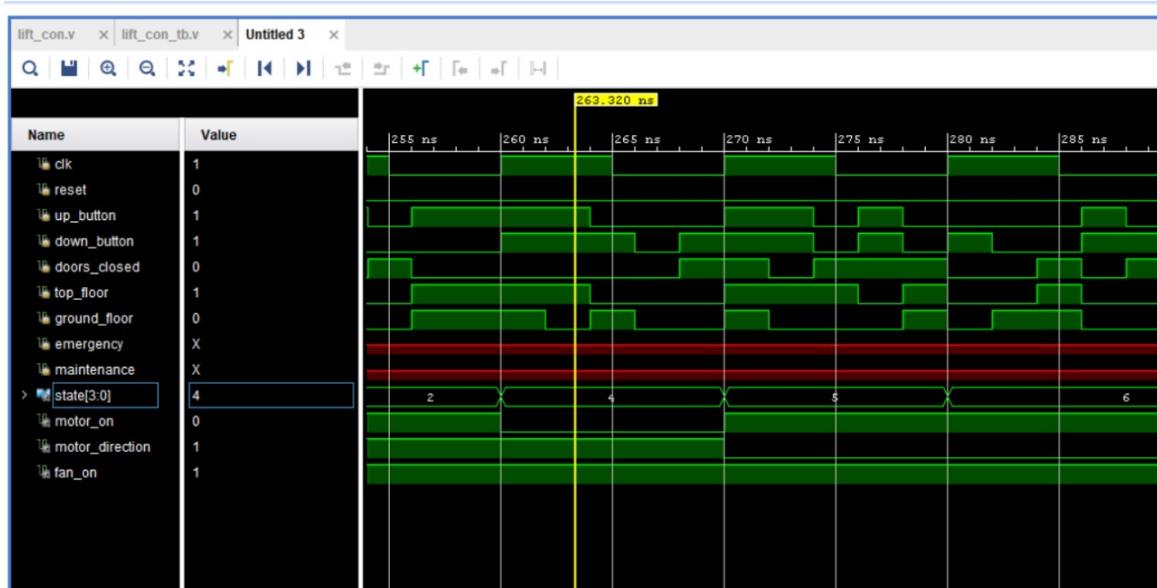
[Figure 1]



[Figure 2]



[Figure 3]



[Figure 4]

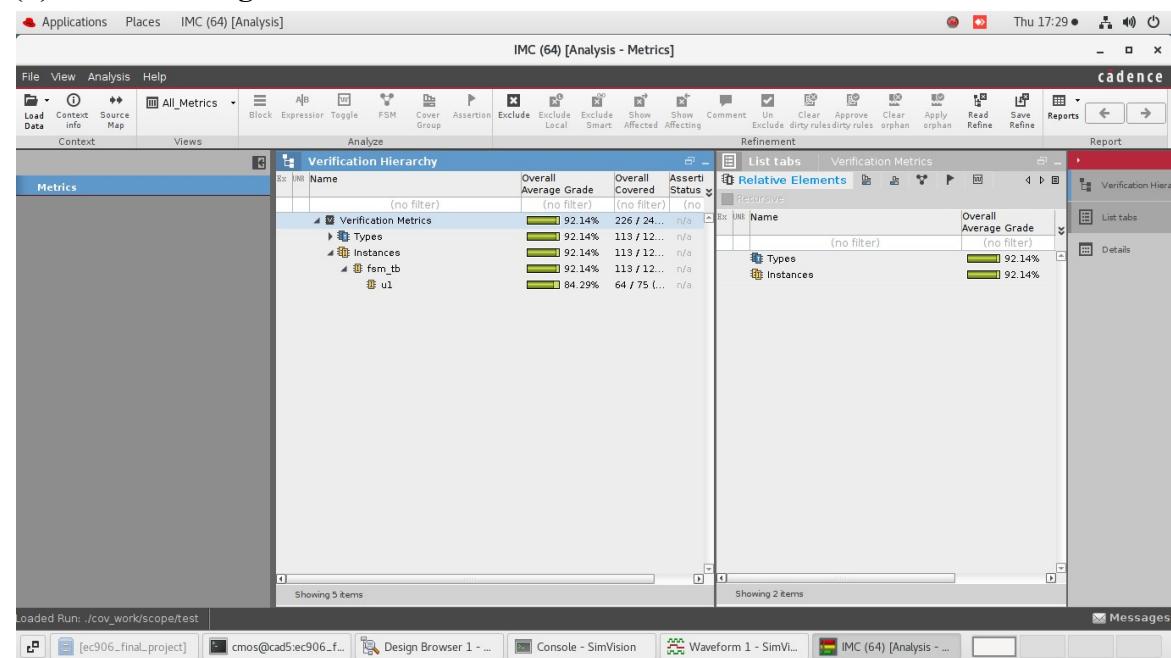
Explanation of Simulation graph:

For example, if it is in state 4 (Top):

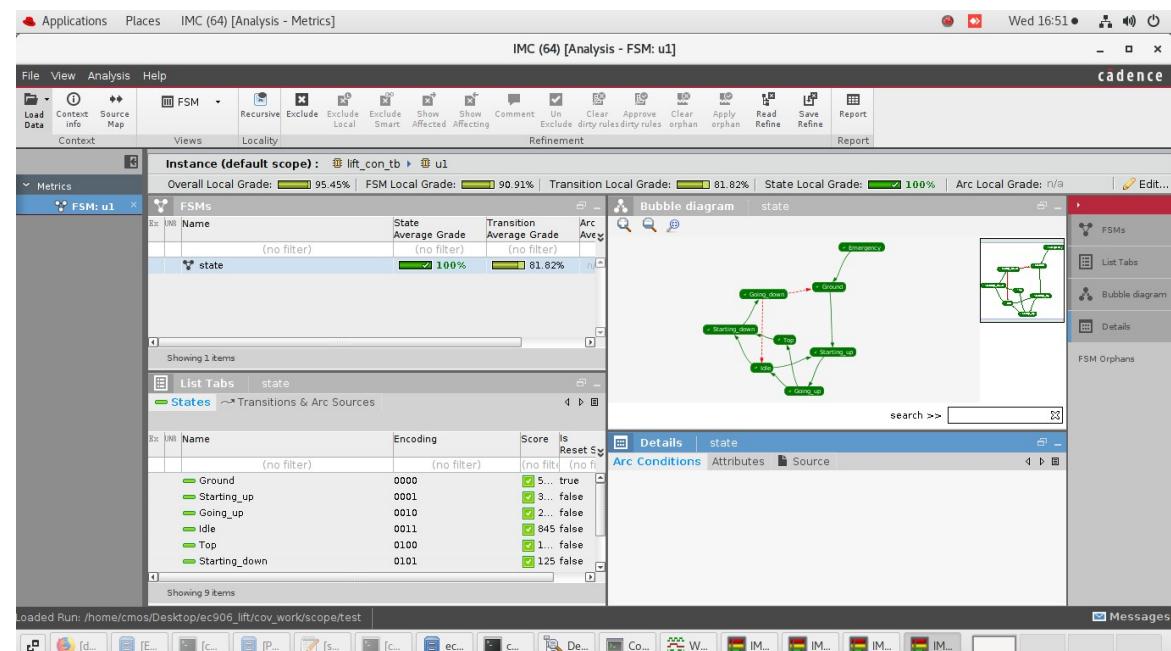
It will go to Top state if top_floor = 1, reset = 0 and clk = 1.



(ii) Code Coverage:



[Figure 5]

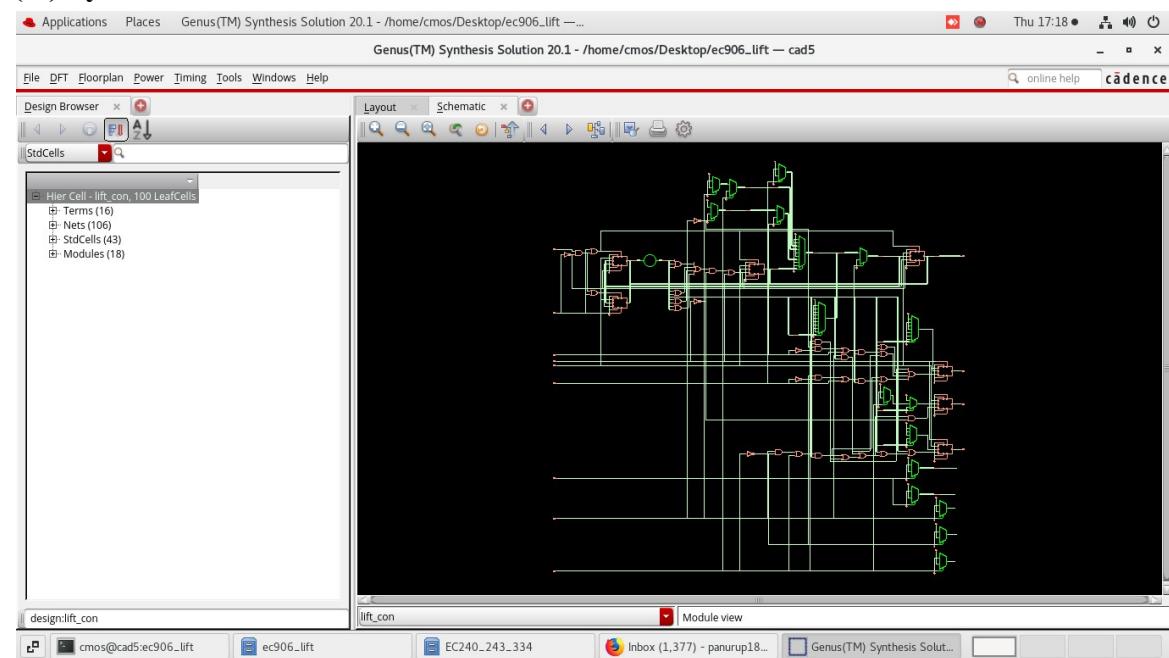


[Figure 6]

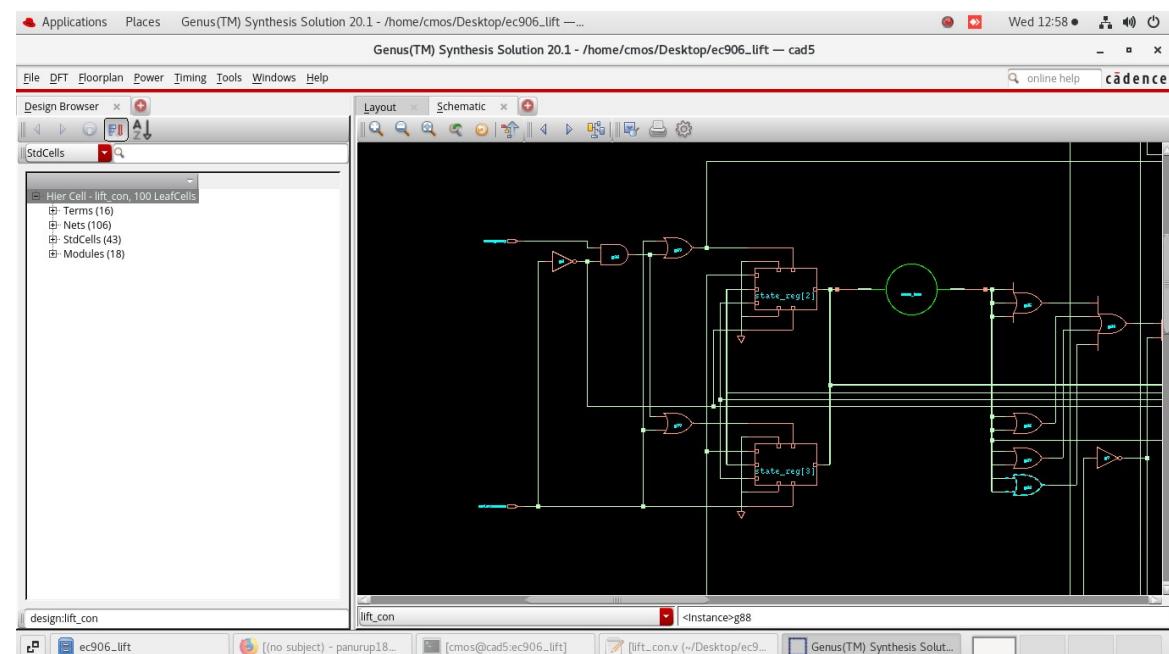
- We have got a code coverage of 92% and the FSM diagram has been shown.
- All the states in the FSM have been covered.



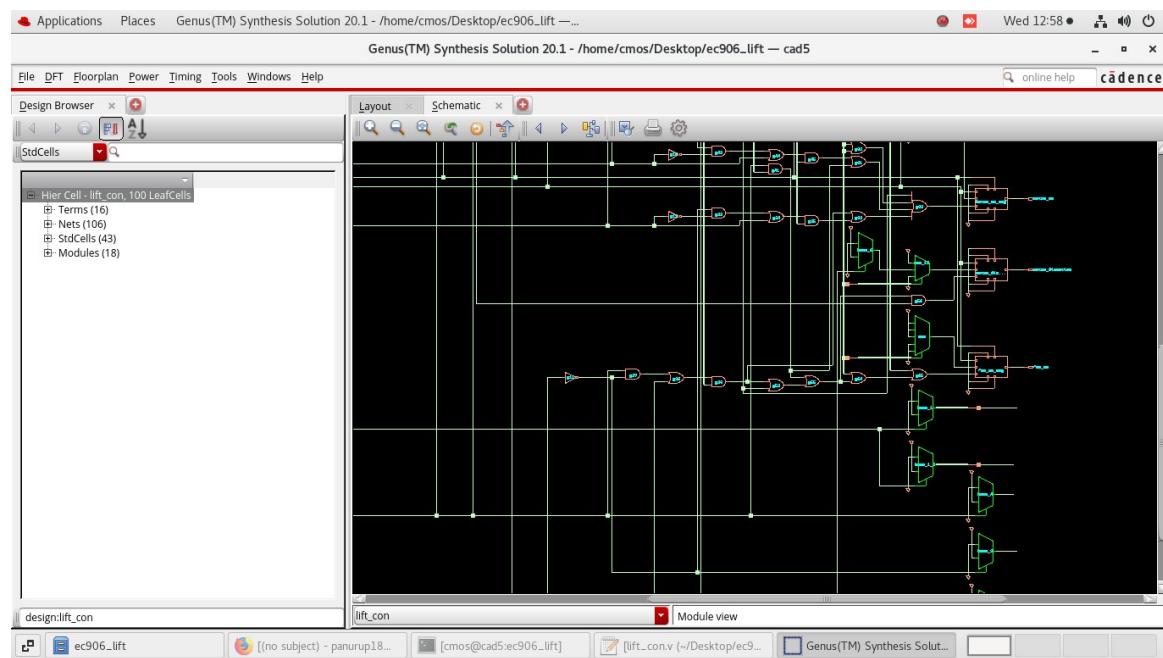
(iii) Synthesis:



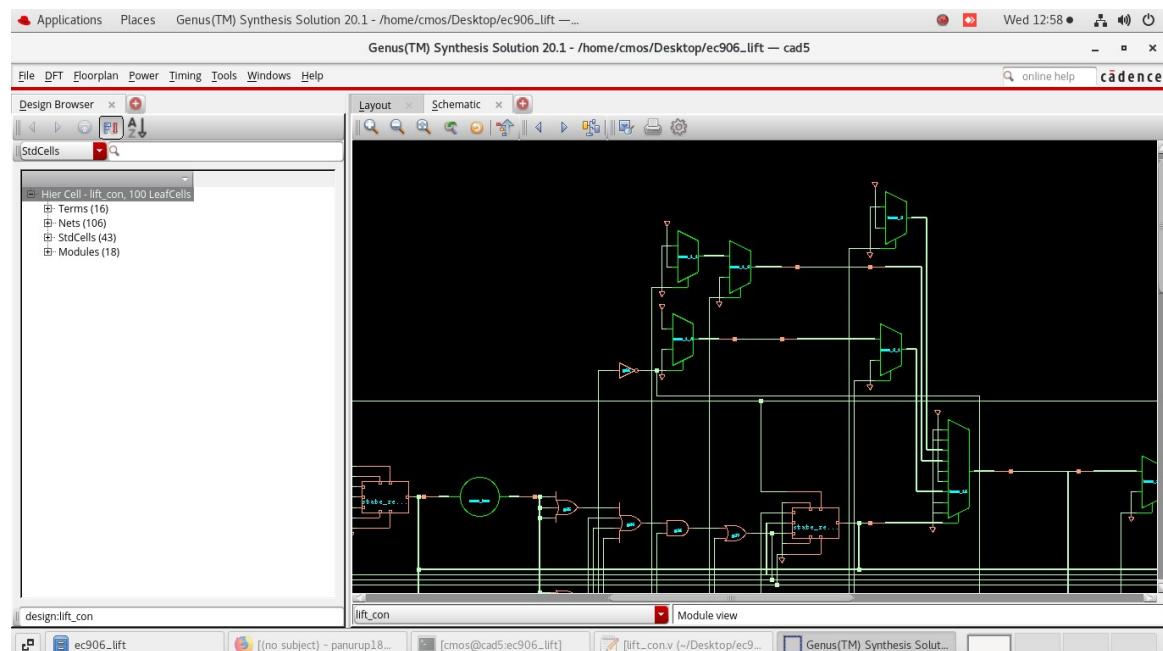
[Figure 7]



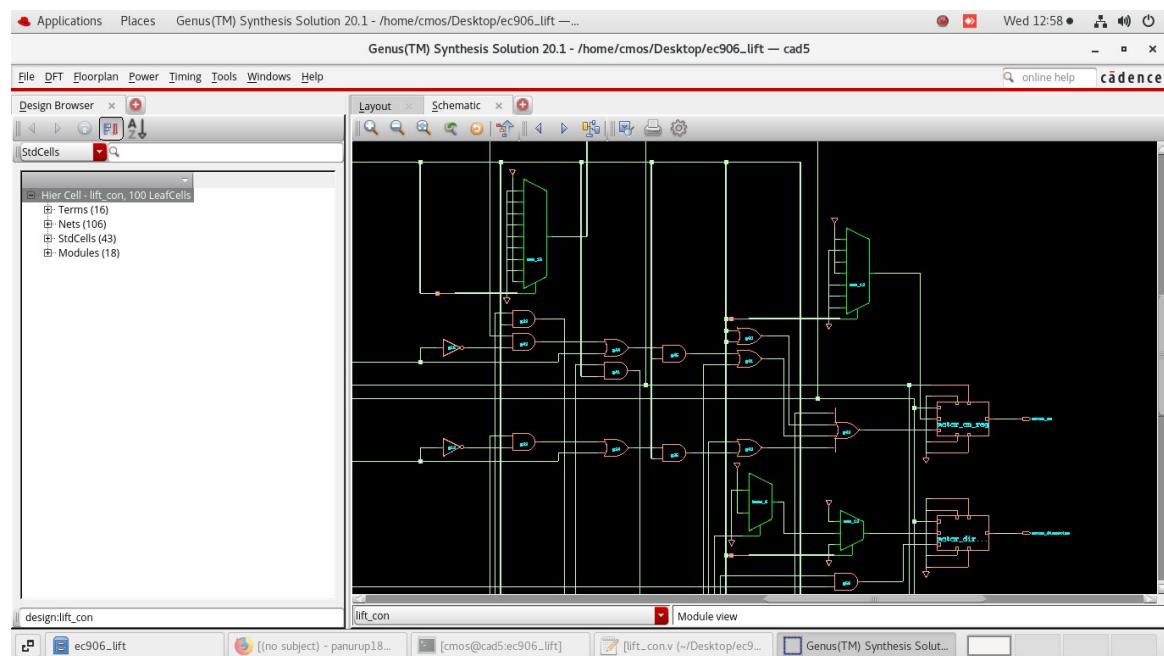
[Figure 8]



[Figure 9]



[Figure 10]



[Figure 11]

(iv) Timing report:

```

Applications Places Terminal
cmos@cad5:ec906_lift
File Edit View Search Terminal Help
which have a clock already defined 'port:lift_con/clk'.
: The current version does not support this SDC command option. However, future versions may be enhanced to support this option.
Statistics for commands executed by read_sdc:
"all_inputs"           - successful      1 , failed      0 (runtime 0.00)
"all_outputs"          - successful      1 , failed      0 (runtime 0.00)
"create_clock"         - successful      1 , failed      0 (runtime 0.00)
"get_clocks"           - successful      2 , failed      0 (runtime 0.00)
"get_ports"            - successful      1 , failed      0 (runtime 0.00)
"set_clock_transition" - successful      2 , failed      0 (runtime 0.00)
"set_input_delay"      - successful      1 , failed      0 (runtime 0.00)
"set_output_delay"     - successful      1 , failed      0 (runtime 0.00)
read_sdc completed in 00:00:00 (hh:mm:ss)
@genus:root:8> report_timing
=====
Generated by:          Genus(TM) Synthesis Solution 20.11-s111_1
Generated on:          Nov 23 2023 04:18:19 pm
Module:                lift_con
Operating conditions: _nominal_ (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library
=====

Path 1: MET (8467 ps) Setup Check with Pin state_reg[2]/clk->d
Group: clk
Startpoint: (F) down_button
Clock: (R) clk
Endpoint: (R) state_reg[2]/d
Clock: (R) clk

    Capture          Launch
    Clock Edge:+ 10000      0
    Dly Adjust:+ 0          0
    Src Latency:+ 0          0
    Net Latency:+ 0 (I)      0 (I)

```

[Figure 12]



```

Applications Places Terminal Thu 16:18 ● 100% ○
cmos@cad5:ec906_lift - - x

File Edit View Search Terminal Help
Clock: (R) clk

Capture Launch
Clock Edge:+ 10000 0
Drv Adjust:+ 0 0
Src Latency:+ 0 0
Net Latency:+ 0 (I) 0 (I)
Arrival:= 10000 0

Setup:- 156
Required Time:= 9844
Launch Clock:- 0
Input Delay:- 1000
Data Path:- 376
Slack:= 8467

Exceptions/Constraints:
input _delay 1000 constraints_top_lift_2_line_5_2_1

#-----#
# Timing Point Flags Arc Edge Cell Fanout Load Trans Delay Arrival Instance
# (fF) (ps) (ps) (ps) Location
#-----#
down_button (u) - F (arrival) 6 21.6 0 0 1000 (-,-)
g66/z (u) in_0->z R unmapped_not 3 11.1 0 19 1019 (-,-)
mux_state_73_20/g1/z (u) sel0->z R unmapped_bmx3 1 3.7 0 52 1072 (-,-)
mux_state_69_15/g1/z (u) data0->z R unmapped_bmx3 1 3.7 0 52 1124 (-,-)
mux_state_15_17/g1/z (u) data0->z R unmapped_bmx13 1 3.7 0 200 1324 (-,-)
mux_state_9_9/g2/z (u) data0->z R unmapped_bmx3 1 3.7 0 52 1376 (-,-)
state_reg[2]/d <<< - R unmapped_d_flop 1 - - 0 1376 (-,-)

#-----#
(u) : Net has unmapped pin(s).

@genus:root: 9> [Figure 13]

```

[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:

```

Applications Places Terminal Thu 17:15 ● 100% ○
cmos@cad5:ec906_lift - - x

File Edit View Search Terminal Help
Wireload mode: enclosed
Area mode: timing library
=====

Instance Module Cell Count Cell Area Net Area Total Area Wireload
-----+
lift_con 100 1921.075 0.000 1921.075 <none> (D)
mux_state_17_15 bmx 1 20.275 0.000 20.275 <none> (D)
mux_state_26_15 bmx 1 40.550 0.000 40.550 <none> (D)
mux_state_34_20 bmx 1 20.275 0.000 20.275 <none> (D)
mux_state_30_15 bmx 3 60.826 0.000 60.826 <none> (D)
mux_state_47_20 bmx 1 40.550 0.000 40.550 <none> (D)
mux_state_41_15 bmx 1 40.550 0.000 40.550 <none> (D)
mux_state_56_15 bmx 1 20.275 0.000 20.275 <none> (D)
mux_state_65_15 bmx 1 40.550 0.000 40.550 <none> (D)
mux_state_73_20 bmx 1 40.550 0.000 40.550 <none> (D)
mux_state_69_15 bmx 3 60.826 0.000 60.826 <none> (D)
ctl_state_15_13 case_box 26 309.197 0.000 309.197 <none> (D)
mux_fan_on_15_13 mux 1 50.688 0.000 50.688 <none> (D)
mux_motor_direction_41_15 bmx 1 20.275 0.000 20.275 <none> (D)
mux_motor_direction_15_13 mux 12 30.413 0.000 30.413 <none> (D)
mux_motor_on_15_13 mux 13 70.963 0.000 70.963 <none> (D)
mux_state_15_13 mux 1 91.238 0.000 91.238 <none> (D)
mux_state_15_17 bmx 16 273.715 0.000 273.715 <none> (D)
mux_state_9_9 bmx 17 81.101 0.000 81.101 <none> (D)
(D) = wireload is default in technology library
@genus:root: S> report_power
Info : Joules engine is used. [RPT-16]
      : Joules engine is being used for the command report_power.
Info : ACTP-0001 [ACTPInfo] Activity propagation started for stim#0 netlist
      : lift_con
Info : ACTP-0001 [ACTPInfo] Activity Propagation Progress Report : 100%
Info : ACTP-0001 Activity propagation ended for stim#0
Info : PWR-0001 [PwrInfo] compute_power effective options
      : -mode : vectorless

```

[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

```

Applications Places Terminal Thu 17:15 ● 11:00 ⌂
cmos@cad5:ec906_lift - - x
File Edit View Search Terminal Help
: -frequency_scaling_factor : 1.0
: -use_clock_freq : stim
: -stim :/stim#
: -fromGenus : 1
Info : ACTP-0001 Timing initialization started
Info : ACTP-0001 Timing initialization ended
Info : PWRA-0002 [PwrInfo] Skipping activity propagation due to -skip_ap
: option...
Warning: PWRA-0302 [PwrWarn] Frequency scaling is not applicable for vectorless
: flow. Ignoring frequency scaling.
Warning: PWRA-0304 [PwrWarn] -stim option is not applicable with vectorless mode
: of power analysis, ignored this option.
Info : PWRA-0002 Started 'vectorless' power computation.
Info : PWRA-0002 Finished power computation.
Info : PWRA-0007 [PwrInfo] Completed successfully.
: Info=6, Warn=2, Error=0, Fatal=0
Instance: /lift_con
Power Unit: W
PDB Frames: /stim#0/frame#0
-----
Category Leakage Internal Switching Total Row%
-----
memory 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
register 9.13762e-07 0.0000e+00 0.0000e+00 9.13762e-07 4.56%
latch 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
logic 5.22344e-06 1.38903e-05 0.0000e+00 1.91137e-05 95.44%
bbox 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
clock 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
pad 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
pm 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.00%
-----
Subtotal 6.13720e-06 1.38903e-05 0.0000e+00 2.00275e-05 100.00%
Percentage 30.64% 69.36% 0.00% 100.00% 100.00%
-----
@genus:root: 6> █
[Figure 15]

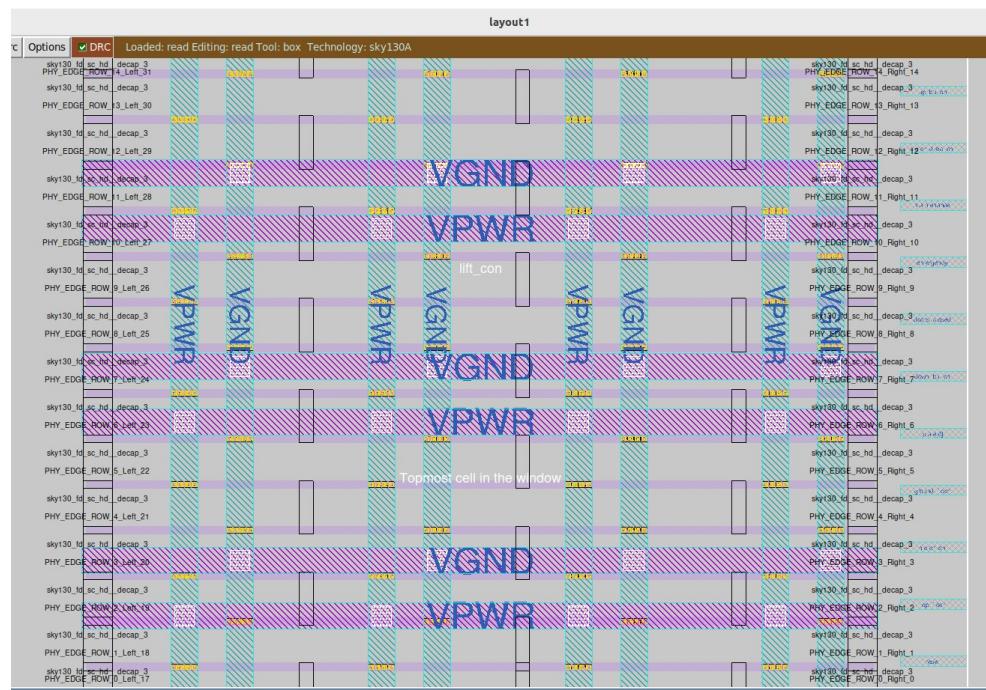
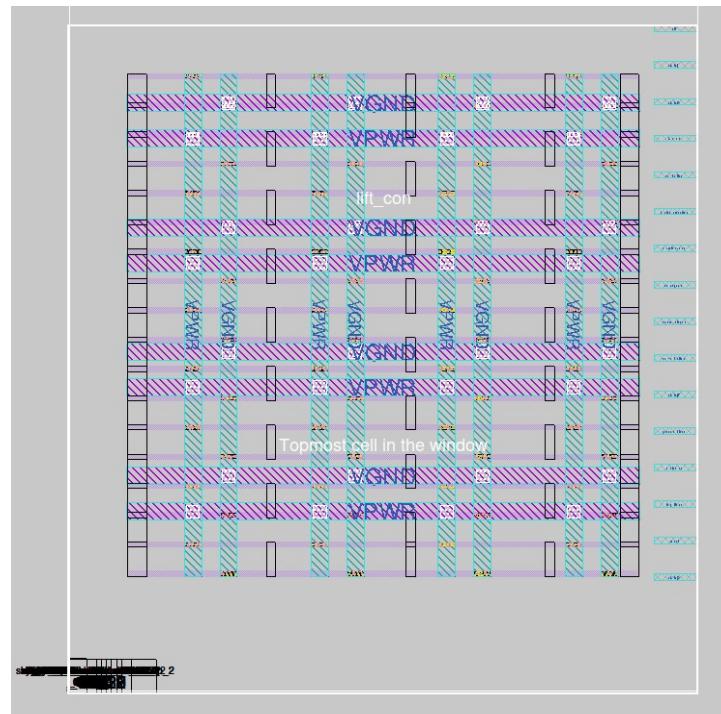
```

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:

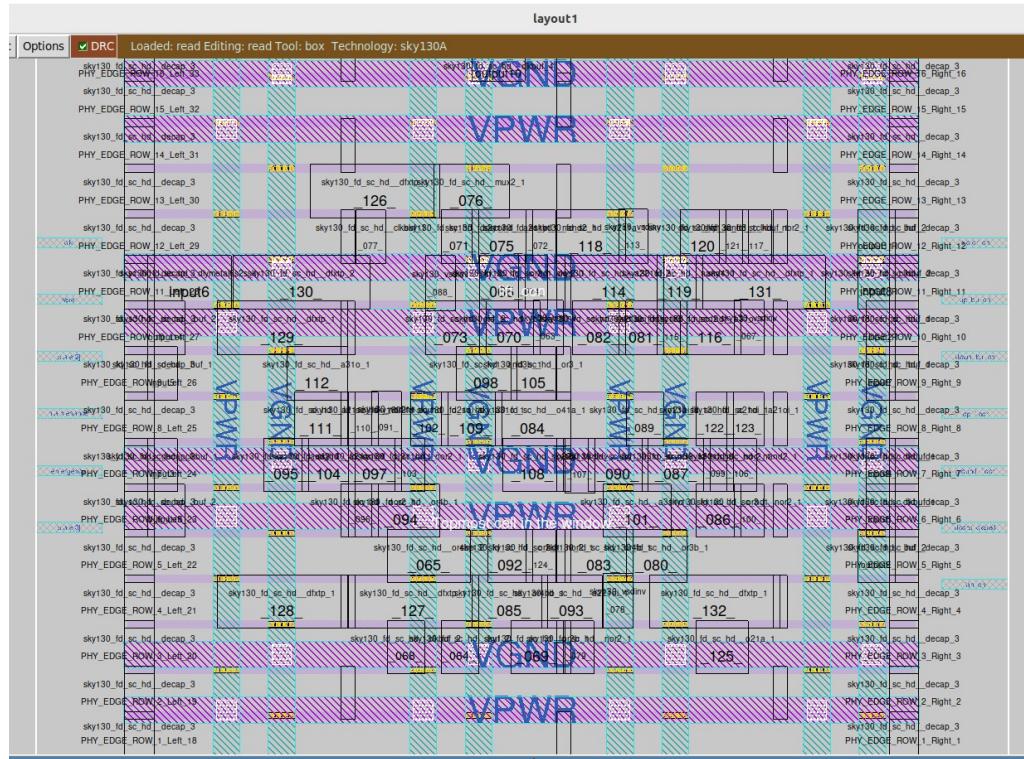
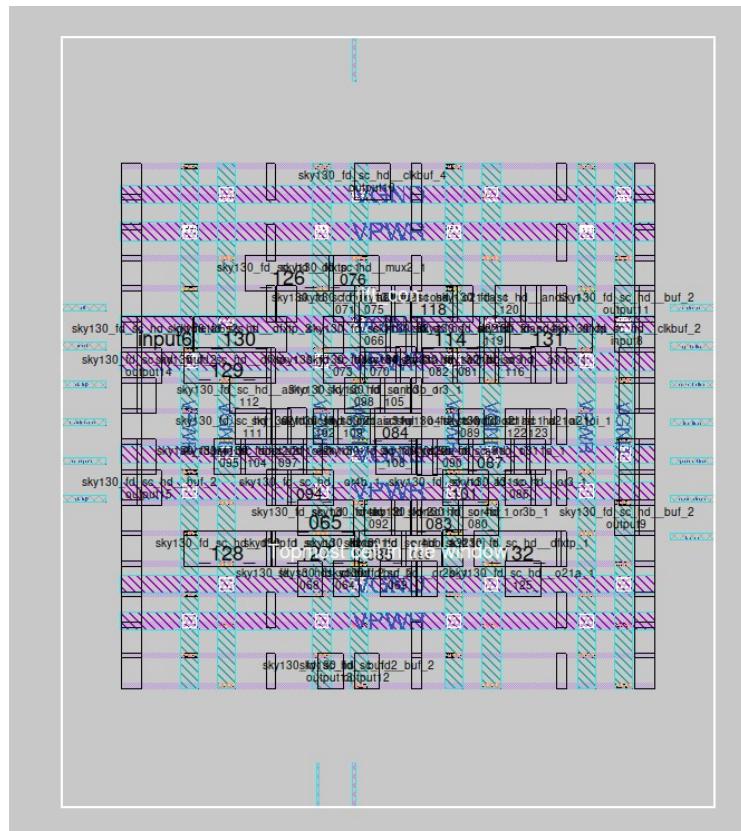
FLOORPLAN:



- 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

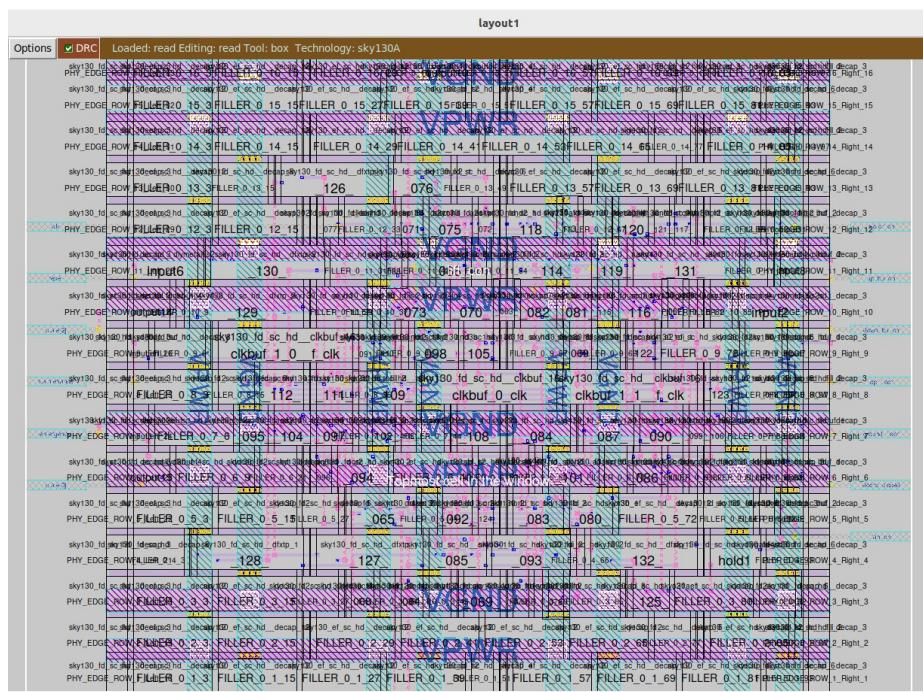
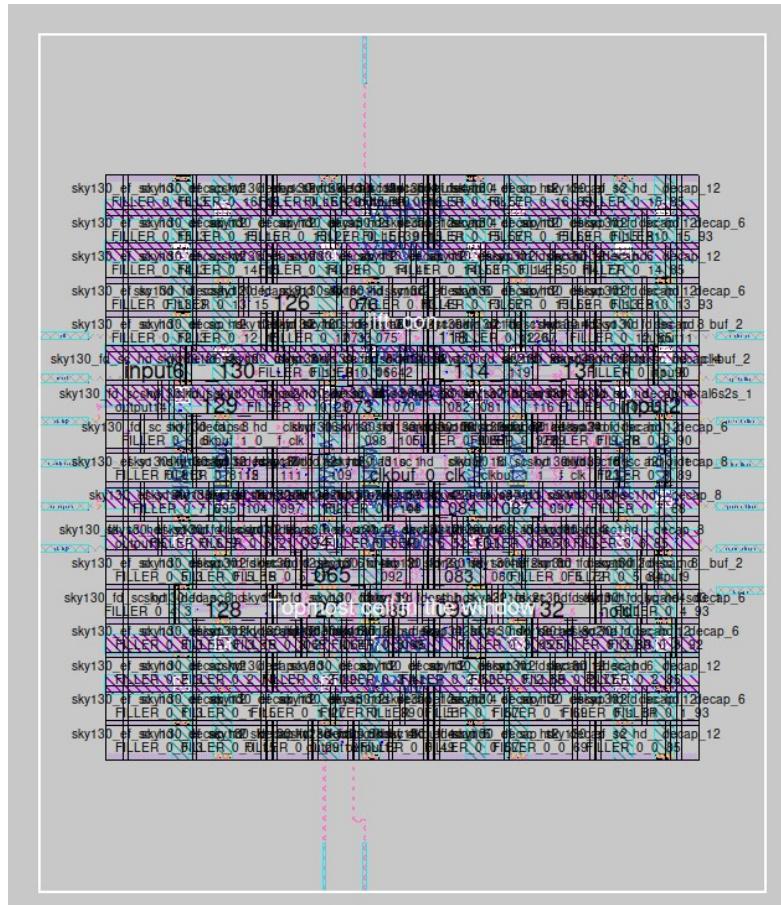


PLACEMENT:



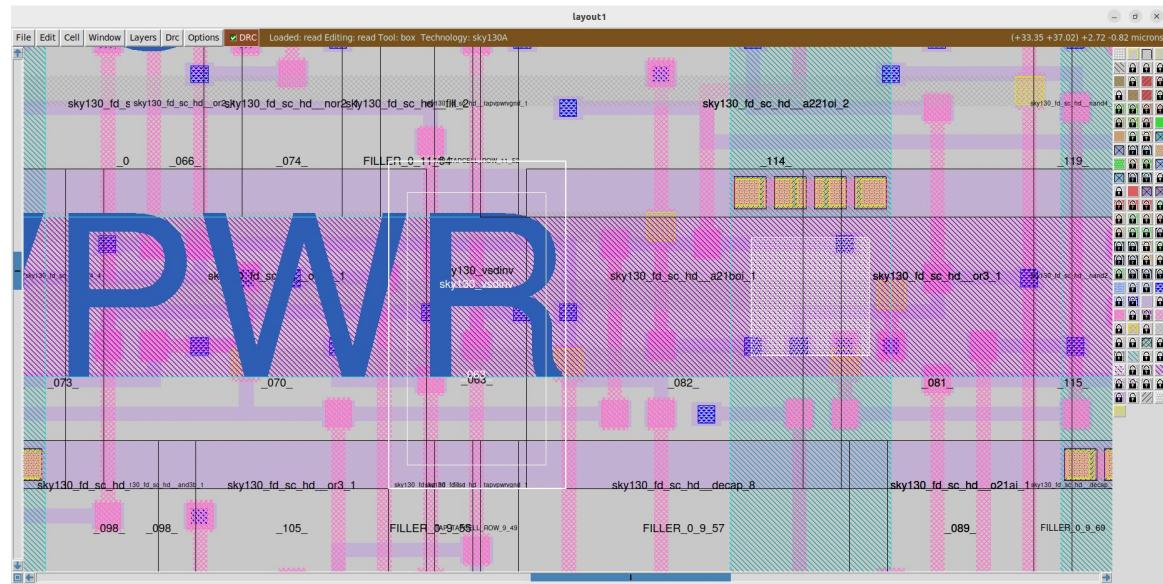


ROUTING:





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_a21bot_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  2.00  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 data arrival time
09
10           0.15  10.00 10.00  clock clk (rise edge)
11           0.00  10.00 9.75  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