BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 468 (January 2024)
VLSI Laboratory

Final Project Report

Section: G1 Group: 01

12-bit Rotator

Course Instructors:

Nafis Sadik, Lecturer
Rafid Hassan Palash, Adjunct Lecturer

Signature of Instructor: ____________________

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: ____________ **Full Name: Anudwaipaon Antu** **Student ID: 1906001**	**Signature:** ____________ **Full Name: Md. Sabbir Ahmed** **Student ID:1906004**
Signature: ____________ **Full Name: Junayet Hossain** **Student ID:1906026**	**Signature:** ____________ **Full Name: Vivek Chowdhury** **Student ID:1906031**

1 Table of Contents

1 Objectives:

The goals of our project are:

- ❖ To understand how RTL code works.
- ❖ Understand several testbench methods.
- ❖ To get familiar with the concepts of coverage.
- ❖ Synthesis of the proposed design in Cadence.
- ❖ To get familiar with physical design and PnR process.

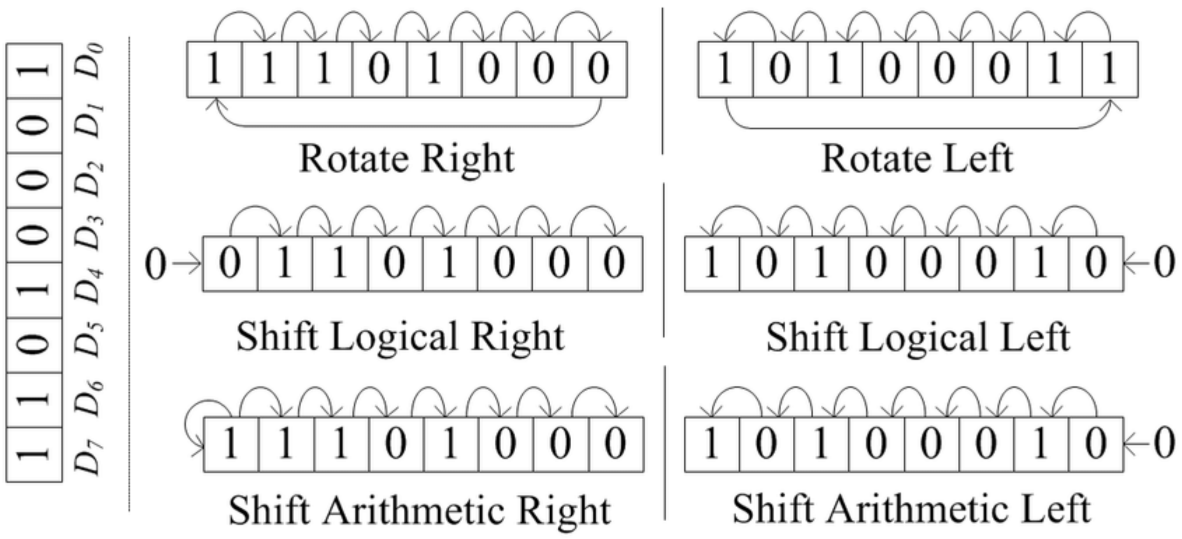
2 Design Steps:

We followed the four following steps:

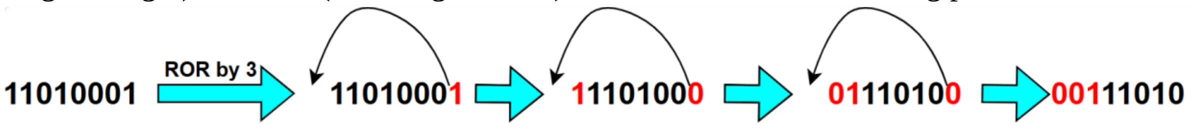
- ❖ Developing the design code and directed testbench with waveforms.
- ❖ Implementing layered testbench with maximum coverage.
- ❖ Doing the optimized synthesis process.
- ❖ Doing the optimized PnR process with maximized chip density with respect to die area, pin planning and power rails.

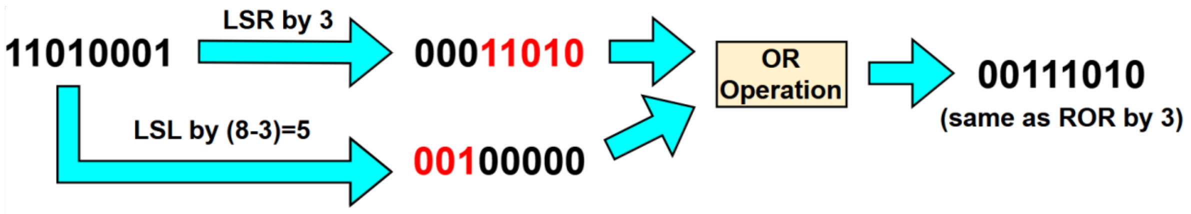
3 Theory:

The rotating operation works as follows:



In order to perform ROR (Rotate Right) and ROL (Rotate left) operation with LSR (Shift Logical Right) and LSL (shift Logical Left), we can follow the following process:





Same explanation goes for ROL. So, if we use a 12-bit number and rotation(i.e-right) has to be done for it, at first, we need to perform LSR by the given amount of rotation and then we need to perform LSL by 12 minus rotation amount and finally by doing OR operation we can have the desired result. As the rotate amount is defined as 4-bit, any number between 0 to 15 can be used. On the other hand, the input is defined as 12-bit. So, a reduced rotate amount variable can be defined using modulo operation by 12 for faster operation.

4 Step-1 (Design Code & Directed Testbench):

Design code for 12-bit rotator:

4.1 design.sv

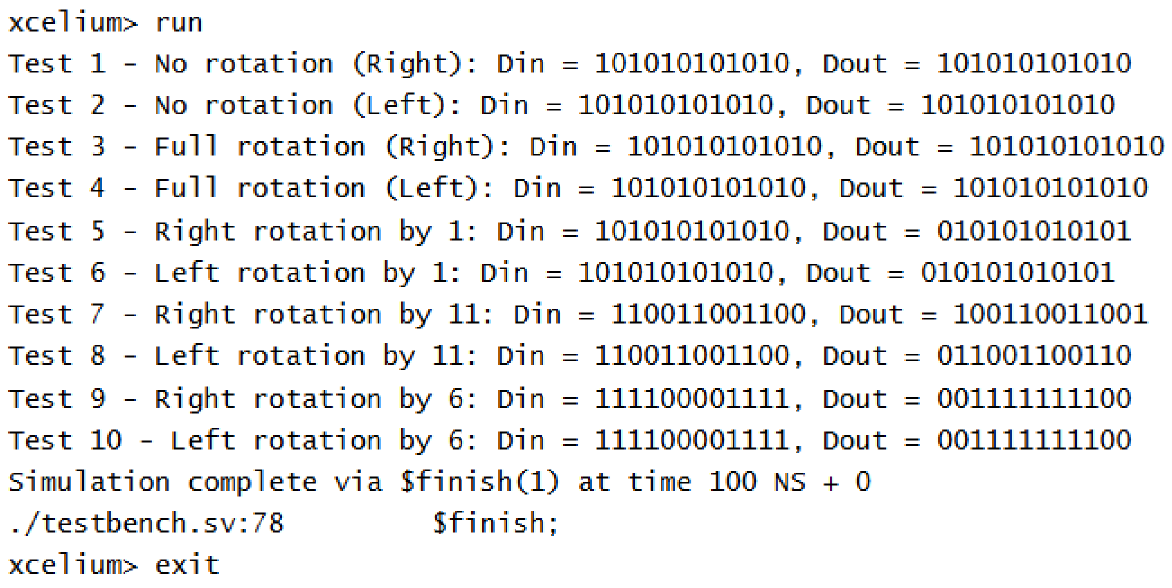
```
module bit_rotator(
    input wire [11:0] Din,
    input wire [3:0] Ramt,
    input wire L, // Direction: 0 = right rotate, 1 = left rotate
    output reg [11:0] Dout
);
  wire [3:0]Ramt_reduced;
  assign Ramt_reduced = Ramt%12;
    always @(*) begin
        if (L == 1) begin
            // Left rotate
            Dout = (Din << Ramt_reduced) | (Din >> (12 - Ramt_reduced));
        end else begin
            // Right rotate
            Dout = (Din >> Ramt_reduced) | (Din << (12 - Ramt_reduced));
        end
    end

endmodule
```

The directed testbench with waveforms for our design code is given below:

4.2 testbench.sv:

```
module testbench;
    reg [11:0] Din;
    reg [3:0] Ramt;
    reg L;
    wire [11:0] Dout;

    bit_rotator uut (
        .Din(Din),
        .Ramt(Ramt),
        .L(L),
        .Dout(Dout)
    );

    initial begin
      $dumpfile("dumpfile.vcd");
        $dumpvars;

        Din = 12'b101010101010;
        Ramt = 4'd0;
        L = 0;
        #10;
        $display("Test 1 - No rotation (Right): Din = %b, Dout = %b", Din, Dout);

        Din = 12'b101010101010;
        Ramt = 4'd0;
        L = 1;
        #10;
        $display("Test 2 - No rotation (Left): Din = %b, Dout = %b", Din, Dout);

        Din = 12'b101010101010;
        Ramt = 4'd12;
        L = 0;
        #10;
        $display("Test 3 - Full rotation (Right): Din = %b, Dout = %b", Din,
Dout);

        Din = 12'b101010101010;
        Ramt = 4'd12;
        L = 1;
        #10;
        $display("Test 4 - Full rotation (Left): Din = %b, Dout = %b", Din,
Dout);

        Din = 12'b101010101010;
        Ramt = 4'd1;
        L = 0;
        #10;
        $display("Test 5 - Right rotation by 1: Din = %b, Dout = %b", Din, Dout);

        Din = 12'b101010101010;
        Ramt = 4'd1;
        L = 1;
        #10;
        $display("Test 6 - Left rotation by 1: Din = %b, Dout = %b", Din, Dout);
        Din = 12'b110011001100;
        Ramt = 4'd11;
        L = 0;
        #10;
        $display("Test 7 - Right rotation by 11: Din = %b, Dout = %b", Din,
Dout);

        Din = 12'b110011001100;
        Ramt = 4'd11;
        L = 1;
        #10;
        $display("Test 8 - Left rotation by 11: Din = %b, Dout = %b", Din, Dout);

        Din = 12'b111100001111;
        Ramt = 4'd6;
        L = 0;
        #10;
        $display("Test 9 - Right rotation by 6: Din = %b, Dout = %b", Din, Dout);

        Din = 12'b111100001111;
        Ramt = 4'd6;
        L = 1;
        #10;
        $display("Test 10 - Left rotation by 6: Din = %b, Dout = %b", Din, Dout);

        $finish;
    end
endmodule
```

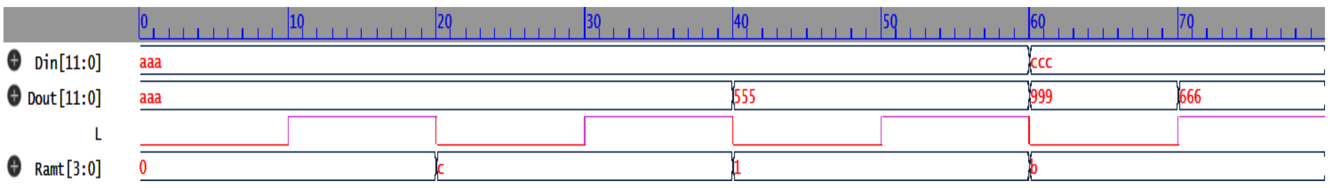
4.3 Result:



```
xcelium> run
Test 1 - No rotation (Right): Din = 101010101010, Dout = 101010101010
Test 2 - No rotation (Left): Din = 101010101010, Dout = 101010101010
Test 3 - Full rotation (Right): Din = 101010101010, Dout = 101010101010
Test 4 - Full rotation (Left): Din = 101010101010, Dout = 101010101010
Test 5 - Right rotation by 1: Din = 101010101010, Dout = 010101010101
Test 6 - Left rotation by 1: Din = 101010101010, Dout = 010101010101
Test 7 - Right rotation by 11: Din = 110011001100, Dout = 100110011001
Test 8 - Left rotation by 11: Din = 110011001100, Dout = 011001100110
Test 9 - Right rotation by 6: Din = 111100001111, Dout = 001111111100
Test 10 - Left rotation by 6: Din = 111100001111, Dout = 001111111100
Simulation complete via $finish(1) at time 100 NS + 0
./testbench.sv:78         $finish;
xcelium> exit
```

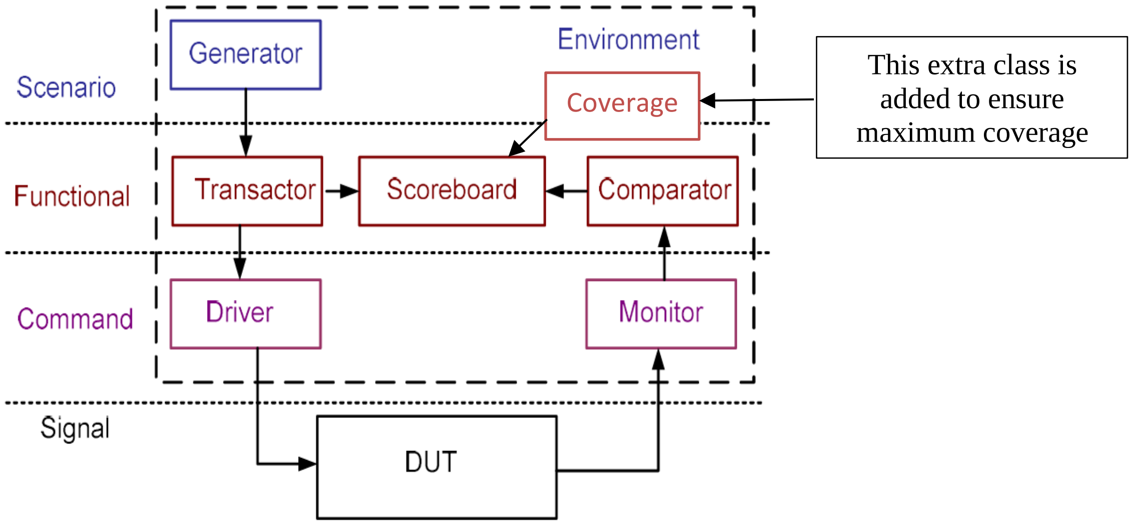
Here, 10 test cases are successfully verified. It can also be verified by the waveforms.

4.4 Waveforms:



The waveforms also verify the testcases. For example, for Ramt=h'11, we will get h'999 for right rotation and h'666 for left rotation which are justified from the waveforms.

5 Step-2 (Layered Testbench with Maximized Coverage):



The necessary .sv files are given according to the schematic:

5.1 environment.sv:

```
`include "generator.sv"
`include "driver.sv"
`include "monitor.sv"
`include "scoreboard.sv"

class environment;
  mailbox gen2driv;
  mailbox driv2sb;
  mailbox mon2sb;

  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;

  event driven;

  virtual rot_if rotif;

  function new(virtual rot_if rotif);
    this.rotif=rotif;
    gen2driv=new();
    driv2sb=new();
    mon2sb=new();

    gen=new(gen2driv);
    drv=new(gen2driv,driv2sb,rotif.DRIVER,driven);
    mon=new(mon2sb,rotif.MONITOR,driven);
    scb=new(driv2sb,mon2sb);

  endfunction

  task main(input int count);
    fork  gen.main(count);
          drv.main(count);
          mon.main(count);
          scb.main(count);
    join
    $finish;
  endtask:main

  function int get_pass_count();
    return scb.pass_count;
  endfunction

  function int get_fail_count();
    return scb.fail_count;
  endfunction
endclass
```

5.2 interface.sv:

```
interface rot_if(input clk);

  logic [11:0] Din;
  logic [3:0] Ramt;
  logic L;
  logic  [11:0] Dout;

  clocking driver_cb @(negedge clk);
    default input #1 output #1;
    output Din, Ramt, L;
  endclocking

  clocking mon_cb @(negedge clk);
    default input #1 output #1;
    input Din, Ramt, L;
    input Dout;
  endclocking

  modport DRIVER (clocking driver_cb, input clk);
  modport MONITOR (clocking mon_cb, input clk);

endinterface
```

5.3 generator.sv:

```
`include "transaction.sv"

class generator;
  mailbox gen2driv;
  transaction g_trans, custom_trans;

  function new(mailbox gen2driv);
    this.gen2driv=gen2driv;
  endfunction

  task main(input int count);
    repeat(count) begin
      g_trans=new();
      g_trans=new custom_trans;
      assert(g_trans.randomize());
      gen2driv.put(g_trans);
    end
  endtask:main

endclass:generator
```

5.4 transaction.sv:

```
class transaction;

  rand bit [11:0] Din;
  rand bit [3:0] Ramt;
  //rand bit  S;
  rand bit L;
  bit [11:0] Dout;

endclass:transaction
```

5.5 driver.sv:

```
class driver;
  mailbox gen2driv, driv2sb;
  virtual rot_if.DRIVER rotif;
  transaction d_trans;
  event driven;

  function new(mailbox gen2driv, driv2sb , virtual rot_if.DRIVER rotif, event
driven);
    this.gen2driv=gen2driv;
    this.rotif=rotif;
    this.driven=driven;
    this.driv2sb=driv2sb;
  endfunction
  task main(input int count);
    repeat(count) begin
      d_trans=new();
      gen2driv.get(d_trans);

      @(rotif.driver_cb);
      rotif.driver_cb.Din <= d_trans.Din;
      rotif.driver_cb.Ramt <= d_trans.Ramt;
         rotif.driver_cb.L <= d_trans.L;
      driv2sb.put(d_trans);
      -> driven;
    end

  endtask:main

endclass:driver
```

5.6 monitor.sv:

```
class monitor;
  mailbox mon2sb;
  virtual rot_if.MONITOR rotif;
  transaction m_trans;
  event driven;

  function new(mailbox mon2sb, virtual rot_if.MONITOR rotif, event driven);
    this.mon2sb=mon2sb;
    this.rotif=rotif;
    this.driven=driven;
  endfunction

  task main(input int count);
    @(driven);
    @(rotif.mon_cb);
    repeat(count) begin
      m_trans=new();
      @(posedge rotif.clk);
      m_trans.Dout=rotif.mon_cb.Dout;
       //m_trans.Cout=counterif.mon_cb.Cout;
      mon2sb.put(m_trans);
    end
  endtask:main


endclass:monitor
```

5.7 coverage.sv:

```
class rot_coverage;
  // Variables to sample
  bit [11:0] Din, Dout;
  bit L;
  bit [3:0] Ramt;
  covergroup cg;
    //option.per_instance = 1;
    //option.comment = "Rot coverage";

    // Cover basic input values
    Din_cp: coverpoint Din {
      bins min_edge = {12'h000};
      bins max_edge = {12'hFFF};
      bins others[4] = {[12'h001:12'hFFE]};  // Split other values into 4 bins
    }

    L_cp: coverpoint L;
    Ramt_cp: coverpoint Ramt;


    // Cross coverage
    LDin_cross: cross L_cp, Din_cp {
      option.weight = 2;  // Important to verify Load with different A values
    }


  endgroup

  function new();
    cg = new;
    cg.start();  // Explicitly start coverage collection
  endfunction


  function void sample(transaction trans);
    this.Din = trans.Din;
    this.Dout = trans.Dout;
    this.L = trans.L;
    this.Ramt = trans.Ramt;
    cg.sample();
  endfunction
endclass
```

5.8 scoreboard.sv:

```
`include "coverage.sv"

class scoreboard;
  mailbox driv2sb;
  mailbox mon2sb;

  transaction d_trans;
  transaction m_trans;

  event driven;

  int pass_count, fail_count;
  bit test_result;  // 1 for pass, 0 for fail

  // Add state tracking
  bit [11:0] expected_count;
  rot_coverage coverage;

  function new(mailbox driv2sb, mon2sb);
```

```
    this.driv2sb = driv2sb;
    this.mon2sb = mon2sb;
    this.pass_count = 0;
    this.fail_count = 0;
    this.expected_count = 0; // Initialize counter
    coverage = new();             // Initialize coverage
  endfunction

  task main(input int count);
    $display("------------------Scoreboard Test Starts--------------------");
    repeat(count) begin
      d_trans = new();
      m_trans = new();

      // Get both transactions
      driv2sb.get(d_trans);
      mon2sb.get(m_trans);

      // Calculate expected value based on inputs
      if(d_trans.L == 1)
      expected_count = (d_trans.Din << ((d_trans.Ramt)%12)) | (d_trans.Din >> (12
- ((d_trans.Ramt)%12)));
    else
      expected_count = (d_trans.Din >> ((d_trans.Ramt)%12)) | (d_trans.Din << (12
- ((d_trans.Ramt)%12)));

      // Compare with actual value
      test_result = (m_trans.Dout == expected_count);

      // Sample coverage after each transaction
      coverage.sample(d_trans);

      if(test_result) begin
        pass_count++;
        $display("||Passed||Din = %b || Ramt = %d || L = %d || Expected_out = %b
|| Resulted_out =
%b",d_trans.Din,d_trans.Ramt,d_trans.L,expected_count,m_trans.Dout );
      end else begin
        fail_count++;
        $display("||Failed||Din = %b || Ramt = %d || L = %d || Expected_out = %b
|| Resulted_out =
%b",d_trans.Din,d_trans.Ramt,d_trans.L,expected_count,m_trans.Dout );
      end
    end
    $display("------------------Scoreboard Test Ends--------------------");
    $display("Total Passes: %0d, Total Fails: %0d", pass_count, fail_count);

    $display("------------------Coverage Summary--------------------");
    $display("Coverage: %.2f%%", coverage.cg.get_coverage());
  endtask

endclass
```

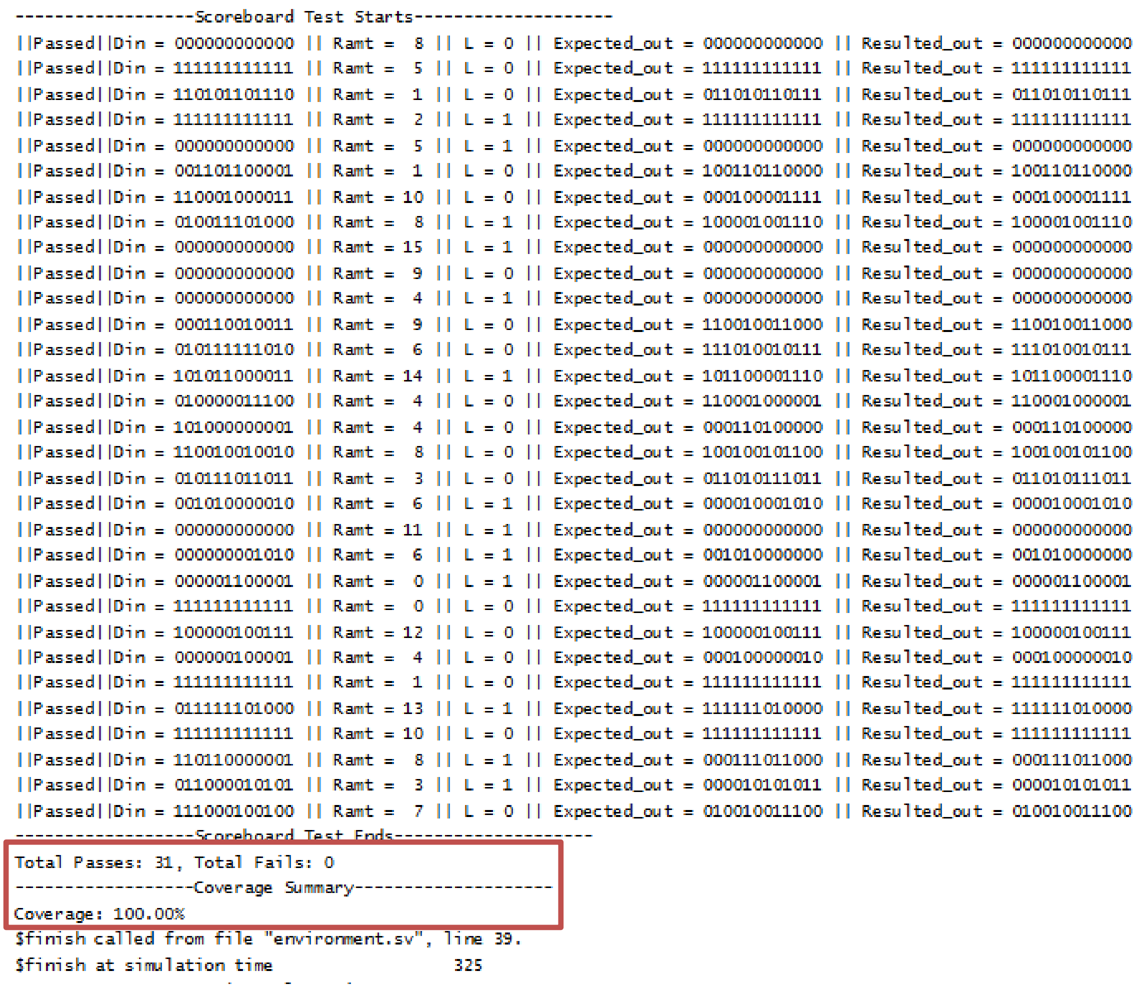
5.9 testcases.sv:

```
`include "environment.sv"

program test(input int count, rot_if rotif, output int pass_count, output int
fail_count, output bit test_done);
  environment env;

  class testcase01 extends transaction;
    constraint c_A {
      Din dist {
        12'h000 :/ 20,
        12'hFFF :/ 20,
        [12'h001:12'hFFE] :/ 60
      };
    }

    constraint c_edge {
      Ramt inside {[0:15]};
      L inside {[0:1]};
    }

  endclass

  initial begin
    testcase01 testcase01handle;
    testcase01handle = new();

    env = new(rotif);
    env.gen.custom_trans = testcase01handle;
    env.main(count);

    pass_count = env.get_pass_count();
    fail_count = env.get_fail_count();
    test_done = 1'b1;
  end

endprogram:test
```

5.10 testbench.sv:

```
`include "testcases.sv"
`include "interface.sv"

module testbench;
  bit clk;
  int pass_count, fail_count;
  bit test_done;

  always #5 clk = ~clk;

  int count = 31;  // Increased test count for better coverage
  rot_if rotif(clk);

  test test01(count, rotif, pass_count, fail_count, test_done);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
  end

  bit_rotator DUT (
    .Din(rotif.Din),
    .Ramt(rotif.Ramt),
    .L(rotif.L),
    .Dout(rotif.Dout)
  );

  initial begin
    #4;
  end

endmodule
```

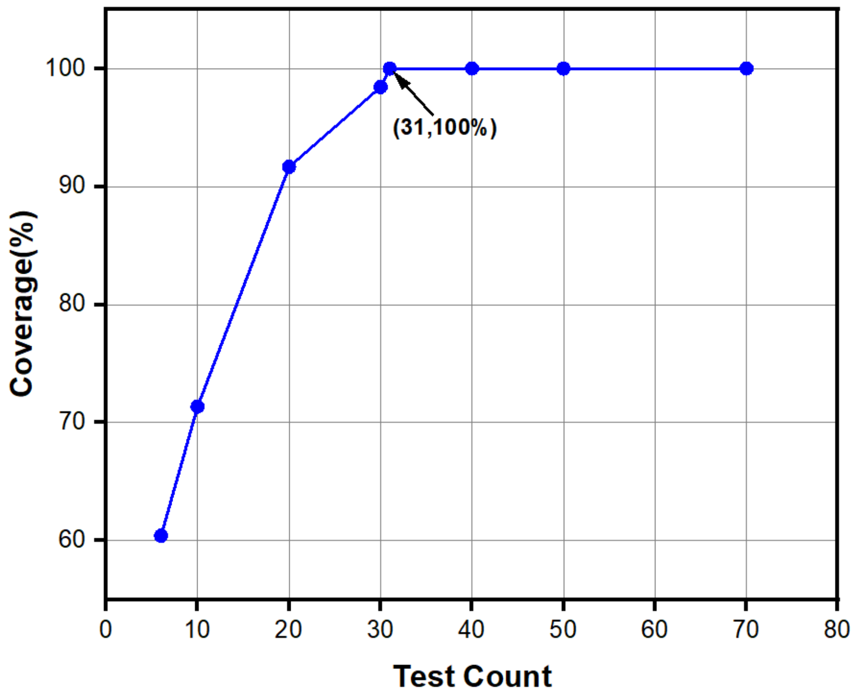
5.11 Result:



```
------------------Scoreboard Test Starts--------------------
||Passed||Din = 000000000000 || Ramt =  8 || L = 0 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 111111111111 || Ramt =  5 || L = 0 || Expected_out = 111111111111 || Resulted_out = 111111111111
||Passed||Din = 110101101110 || Ramt =  1 || L = 0 || Expected_out = 011010110111 || Resulted_out = 011010110111
||Passed||Din = 111111111111 || Ramt =  2 || L = 1 || Expected_out = 111111111111 || Resulted_out = 111111111111
||Passed||Din = 000000000000 || Ramt =  5 || L = 1 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 001101100001 || Ramt =  1 || L = 0 || Expected_out = 100110110000 || Resulted_out = 100110110000
||Passed||Din = 110001000011 || Ramt = 10 || L = 0 || Expected_out = 000100001111 || Resulted_out = 000100001111
||Passed||Din = 010011101000 || Ramt =  8 || L = 1 || Expected_out = 100001001110 || Resulted_out = 100001001110
||Passed||Din = 000000000000 || Ramt = 15 || L = 1 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 000000000000 || Ramt =  9 || L = 0 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 000000000000 || Ramt =  4 || L = 1 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 000110010011 || Ramt =  9 || L = 0 || Expected_out = 110010011000 || Resulted_out = 110010011000
||Passed||Din = 010111111010 || Ramt =  6 || L = 0 || Expected_out = 111010010111 || Resulted_out = 111010010111
||Passed||Din = 101011000011 || Ramt = 14 || L = 1 || Expected_out = 101100001110 || Resulted_out = 101100001110
||Passed||Din = 010000011100 || Ramt =  4 || L = 0 || Expected_out = 110001000001 || Resulted_out = 110001000001
||Passed||Din = 101000000001 || Ramt =  4 || L = 0 || Expected_out = 000110100000 || Resulted_out = 000110100000
||Passed||Din = 110010010010 || Ramt =  8 || L = 0 || Expected_out = 100100101100 || Resulted_out = 100100101100
||Passed||Din = 010111011011 || Ramt =  3 || L = 0 || Expected_out = 011010111011 || Resulted_out = 011010111011
||Passed||Din = 001010000010 || Ramt =  6 || L = 1 || Expected_out = 000010001010 || Resulted_out = 000010001010
||Passed||Din = 000000000000 || Ramt = 11 || L = 1 || Expected_out = 000000000000 || Resulted_out = 000000000000
||Passed||Din = 000000001010 || Ramt =  6 || L = 1 || Expected_out = 001010000000 || Resulted_out = 001010000000
||Passed||Din = 000001100001 || Ramt =  0 || L = 1 || Expected_out = 000001100001 || Resulted_out = 000001100001
||Passed||Din = 111111111111 || Ramt =  0 || L = 0 || Expected_out = 111111111111 || Resulted_out = 111111111111
||Passed||Din = 100000100111 || Ramt = 12 || L = 0 || Expected_out = 100000100111 || Resulted_out = 100000100111
||Passed||Din = 000000100001 || Ramt =  4 || L = 0 || Expected_out = 000100000010 || Resulted_out = 000100000010
||Passed||Din = 111111111111 || Ramt =  1 || L = 0 || Expected_out = 111111111111 || Resulted_out = 111111111111
||Passed||Din = 011111101000 || Ramt = 13 || L = 1 || Expected_out = 111111010000 || Resulted_out = 111111010000
||Passed||Din = 111111111111 || Ramt = 10 || L = 0 || Expected_out = 111111111111 || Resulted_out = 111111111111
||Passed||Din = 110110000001 || Ramt =  8 || L = 1 || Expected_out = 000111011000 || Resulted_out = 000111011000
||Passed||Din = 011000010101 || Ramt =  3 || L = 1 || Expected_out = 000010101011 || Resulted_out = 000010101011
||Passed||Din = 111000100100 || Ramt =  7 || L = 0 || Expected_out = 010010011100 || Resulted_out = 010010011100
------------------Scoreboard Test Ends--------------------
Total Passes: 31, Total Fails: 0
------------------Coverage Summary--------------------
Coverage: 100.00%
$finish called from file "environment.sv", line 39.
$finish at simulation time                   325
```

5.12 Test Count vs Coverage(%) Plot:

Test Count	Coverage (%)
6	60.42
10	71.35
20	91.67
30	98.44
31	100
40	100
50	100
70	100



This plot shows that after 30 test counts, coverage will reach its maximized value(100%).

6 Step-3 (Synthesis with Optimized Power, Performance & Area):

Three files (bit_rotator.v, bit_rotator.tcl, bit_rotator.sdc) are required for the synthesis process.

bit_rotator.v:

```
module bit_rotator(
    input wire [11:0] Din,
    input wire [3:0] Ramt,
    input wire L, // Direction: 0 = right rotate, 1 = left rotate
    output reg [11:0] Dout
);
  wire [3:0]Ramt_reduced;
  assign Ramt_reduced = Ramt%12;
    always @(*) begin
        if (L == 1) begin
            // Left rotate
            Dout = (Din << Ramt_reduced) | (Din >> (12 - Ramt_reduced));
        end else begin
            // Right rotate
            Dout = (Din >> Ramt_reduced) | (Din << (12 - Ramt_reduced));
        end
    end

endmodule
```

6.1 bit_rotator.tcl:

```
#run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false / ;
if {[file exists /proc/cpuinfo]} {
sh grep "model name" /proc/cpuinfo
sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"
##############################################################################
### Preset global variables and attributes
##############################################################################
set DESIGN bit_rotator
set SYN_EFF high
set MAP_EFF high
set OPT_EFF high
# Directory of PDK
#set pdk_dir /home/wsadiq/buet_flow/GPDK045
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
#set_attribute init_lib_search_path $pdk_dir/gsclib045/timing
#set_attribute init_hdl_search_path /home/wsadiq/buet_flow/rtl
set_attribute init_lib_search_path $pdk_dir

#set_attribute init_hdl_search_path ../../rtl
##Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
#set_attribute library "\
slow_vdd1v0_basicCells_hvt.lib \
slow_vdd1v0_basicCells.lib \
slow_vdd1v0_basicCells_lvt.lib"
set_attribute library "\
slow_vdd1v0_basicCells.lib"
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]
# If you dont want to use LVT uncomment this line
#set_dont_use [get_lib_cells *LVT*]

####################################################################
### Load Design
#####################################################################
###source verilog_files.tcl
read_hdl "\
${DESIGN}.v"
elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
time_info Elaboration
check_design -unresolved
####################################################################
### Constraints Setup
#####################################################################
read_sdc bit_rotator.sdc

report timing -encounter >> reports/${DESIGN}_pretim.rpt


################################################################################
###################
### Synthesizing to generic
################################################################################
###################
syn_generic
puts "Runtime & Memory after 'syn_generic'"
time_info GENERIC
report datapath > reports/${DESIGN}_datapath_generic.rpt
generate_reports -outdir reports -tag generic
write_db -to_file ${DESIGN}_generic.db
report timing -encounter >> reports/${DESIGN}_generic.rpt


##This synthesizes your code
synthesize -to_mapped

## This writes all your files
write -mapped > bit_rotator_synth.v

## THESE FILES ARE NOT REQUIRED, THE SDC FILE IS A TIMING FILE
write_script > script
```

6.2 bit_rotator.sdc:

```
# Setting up time units
set_units -time 1ns -capacitance pF

# Setting the clock period to 10ns, as period = 1/freq, here, freq = 100MHz
set clock_period 10;

# Set the top module
set top_module "bit_rotator"

# Clock port (if applicable)
set clock_port {clk}

# Reset port (if applicable)
set reset_port {rst_n}

# Setting the input ports in a list
set input_ports {Din[11:0], Ramt[3:0], L}

# Setting the output port in a list
set output_ports {Dout[11:0]}

# Define clocks
create_clock -period ${clock_period} -waveform {0 5} -name func_clk [get_ports
${clock_port}]

# Setting up constraints for the reset signal (if reset is synchronous and
multicycle)
set_multicycle_path -setup 3 -from [get_ports ${reset_port}]
set_multicycle_path -hold 2 -from [get_ports ${reset_port}]

# Define input delays (e.g., max propagation delay to input ports)
set_input_delay 0.4 -clock [get_clocks {func_clk}] ${input_ports}

# Define output delays (e.g., max propagation delay from output ports)
set_output_delay 0.6 -clock [get_clocks {func_clk}] ${output_ports}
```

Now, we need to optimize the synthesis process with respect to setup and hold time, clock frequency, input and output delay for different efforts (low, medium, high).

6.3 Data for Low & Medium Effort:

setup time(clock)	leakage power(nW)	internal power(nW)	net power(nW)	switching power (nW)	total area
0.1	3.62	2142.62	1039.08	3181.7	229.82
1	3.62	2142.62	1039.08	3181.7	229.82
3	3.62	2142.62	1039.08	3181.7	229.82
5	3.62	2142.62	1039.08	3181.7	229.82
8	3.62	2142.62	1039.08	3181.7	229.82

hold time(clock)	leakage power(nW)	internal power(nW)	net power(nW)	switching power (nW)	total area
0.1	3.62	2142.62	1039.08	3181.7	229.82
1	3.62	2142.62	1039.08	3181.7	229.82
2	3.62	2142.62	1039.08	3181.7	229.82
5	3.62	2142.62	1039.08	3181.7	229.82
8	3.62	2142.62	1039.08	3181.7	229.82

clock freq (MHz)	leakage power(nW)	internal power(nW)	net power(nW)	switching power (nW)	total area
20	3.62	2142.62	1039.08	3181.7	229.82
50	3.62	2142.62	1039.08	3181.7	229.82
100	3.62	2142.62	1039.08	3181.7	229.82
200	3.62	2142.62	1039.08	3181.7	229.82
500	3.62	2142.62	1039.08	3181.7	229.82
1000	3.62	2142.62	1039.08	3181.7	229.82

input delay (ns)	leakage power(nW)	internal power(nW)	net power(nW)	switching power (nW)	total area
0.1	3.62	2142.62	1039.08	3181.7	229.82
0.4	3.62	2142.62	1039.08	3181.7	229.82
0.8	3.62	2142.62	1039.08	3181.7	229.82
1.5	3.62	2142.62	1039.08	3181.7	229.82
3	3.62	2142.62	1039.08	3181.7	229.82
5	3.62	2142.62	1039.08	3181.7	229.82

output delay (ns)	leakage power(nW)	internal power(nW)	net power(nW)	switching power (nW)	total area
0.1	3.62	2142.62	1039.08	3181.7	229.82
0.3	3.62	2142.62	1039.08	3181.7	229.82
0.6	3.62	2142.62	1039.08	3181.7	229.82
2	3.62	2142.62	1039.08	3181.7	229.82
5	3.62	2142.62	1039.08	3181.7	229.82

6.4 Data for High Effort:

setup time(clock)	leakage power(nW)	internal power(nW)	net power(nW)	switching power(nW)	total area
0.1	3.61	2147.81	1038.45	3186.25	229.14
1	3.61	2147.81	1038.45	3186.25	229.14
3	3.61	2147.81	1038.45	3186.25	229.14
5	3.61	2147.81	1038.45	3186.25	229.14
8	3.61	2147.81	1038.45	3186.25	229.14

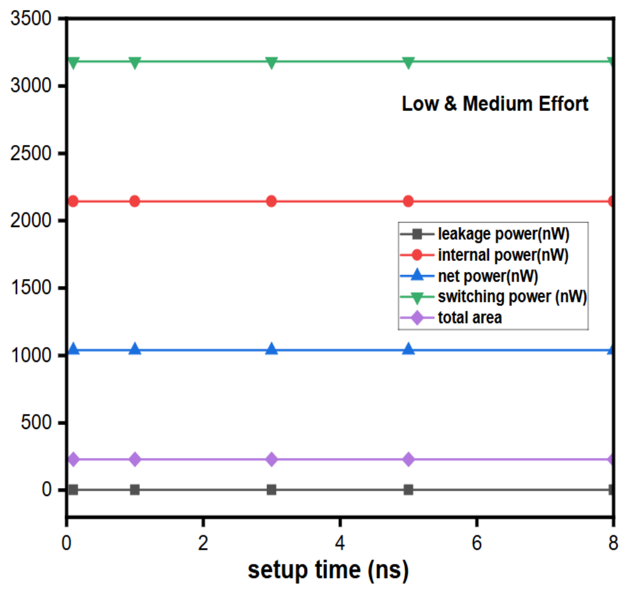
hold time(clock)	leakage power(nW)	internal power(nW)	net power(nW)	switching power(nW)	total area
0.1	3.61	2147.81	1038.45	3186.25	229.14
1	3.61	2147.81	1038.45	3186.25	229.14
2	3.61	2147.81	1038.45	3186.25	229.14
5	3.61	2147.81	1038.45	3186.25	229.14
8	3.61	2147.81	1038.45	3186.25	229.14

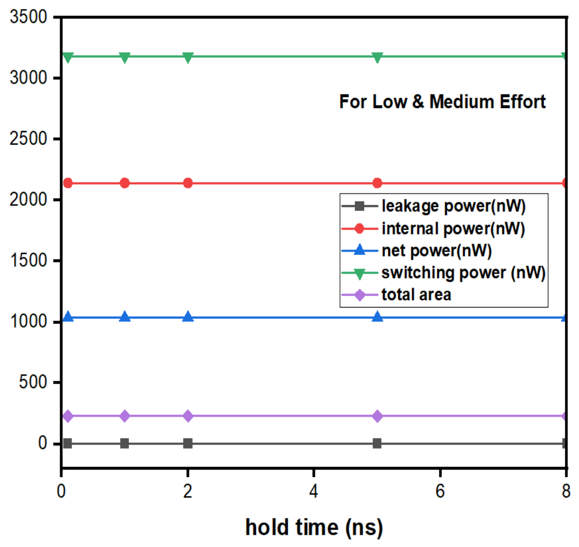
clock freq (MHz)	leakage power(nW)	internal power(nW)	net power(nW)	switching power(nW)	total area
20	3.61	2147.81	1038.45	3186.25	229.14
50	3.61	2147.81	1038.45	3186.25	229.14
100	3.61	2147.81	1038.45	3186.25	229.14
200	3.61	2147.81	1038.45	3186.25	229.14
500	3.61	2147.81	1038.45	3186.25	229.14
1000	3.61	2147.81	1038.45	3186.25	229.14

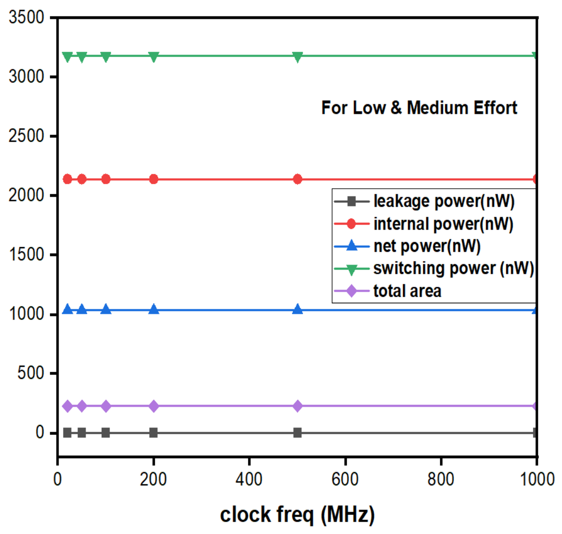
input delay (ns)	leakage power(nW)	internal power(nW)	net power(nW)	switching power(nW)	total area
0.1	3.61	2147.81	1038.45	3186.25	229.14
0.4	3.61	2147.81	1038.45	3186.25	229.14
0.8	3.61	2147.81	1038.45	3186.25	229.14
1.5	3.61	2147.81	1038.45	3186.25	229.14
3	3.61	2147.81	1038.45	3186.25	229.14
5	3.61	2147.81	1038.45	3186.25	229.14

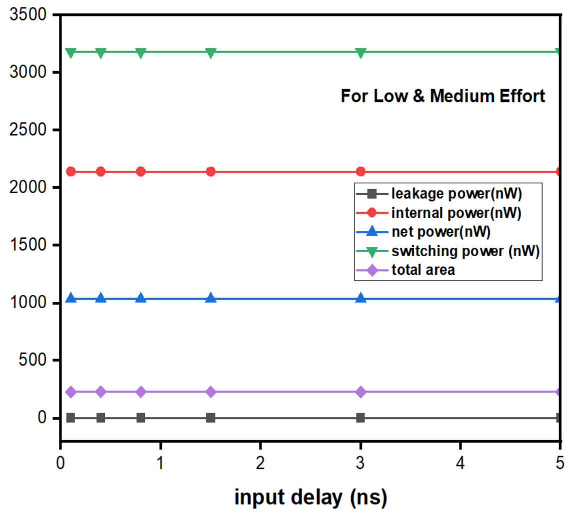
output delay (ns)	leakage power(nW)	internal power(nW)	net power(nW)	switching power(nW)	total area
0.1	3.61	2147.81	1038.45	3186.25	229.14
0.3	3.61	2147.81	1038.45	3186.25	229.14
0.6	3.61	2147.81	1038.45	3186.25	229.14
2	3.61	2147.81	1038.45	3186.25	229.14
5	3.61	2147.81	1038.45	3186.25	229.14

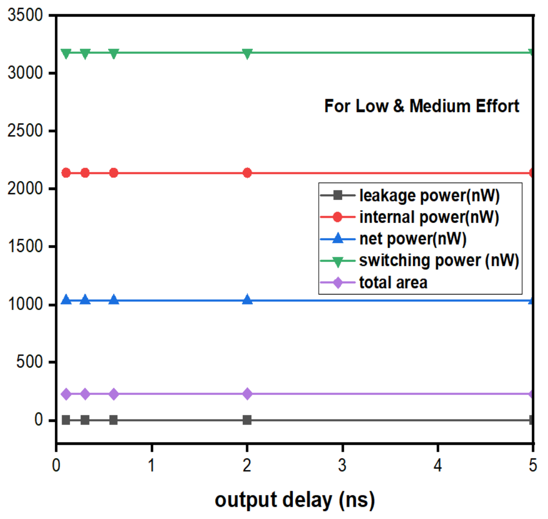
6.5 Plots:

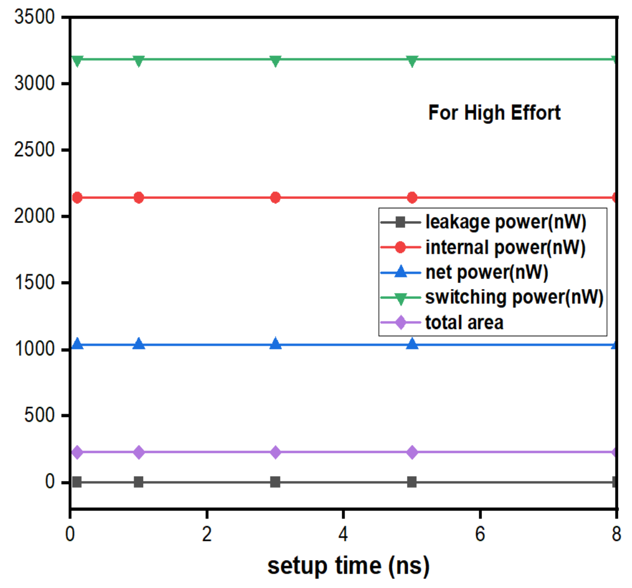

Low & Medium Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power (nW)
total area
setup time (ns)
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8

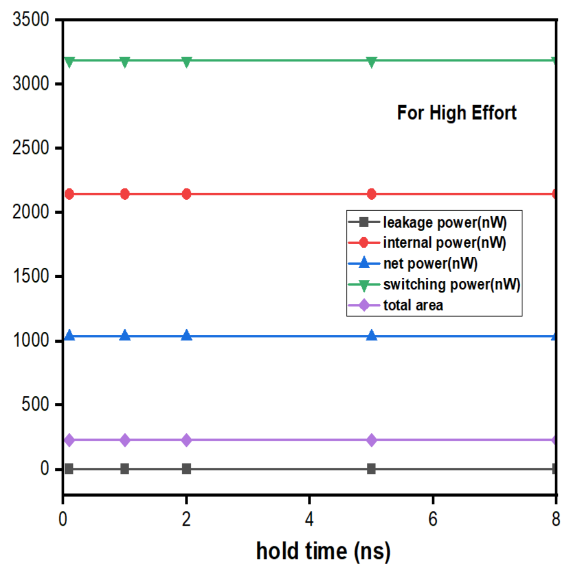

For Low & Medium Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power (nW)
total area
hold time (ns)

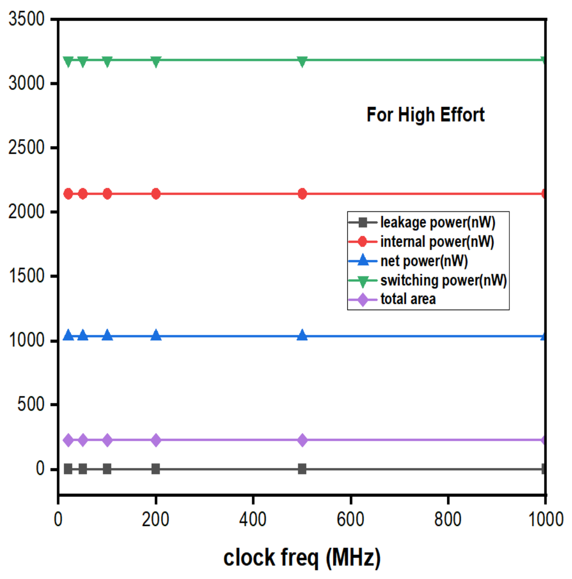

For Low & Medium Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power (nW)
total area
clock freq (MHz)
0
200
400
600
800
1000

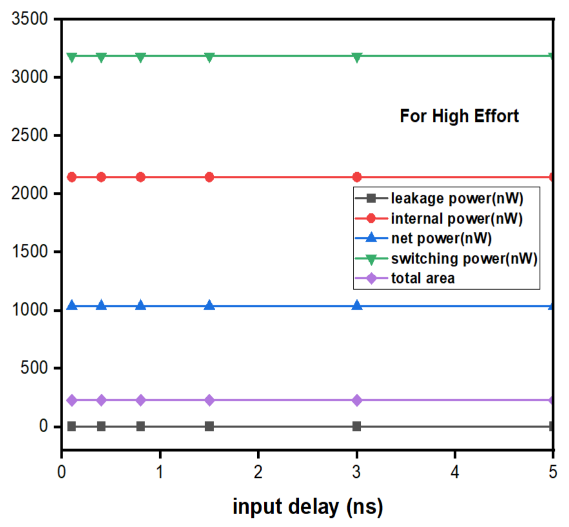

For Low & Medium Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power (nW)
total area
input delay (ns)

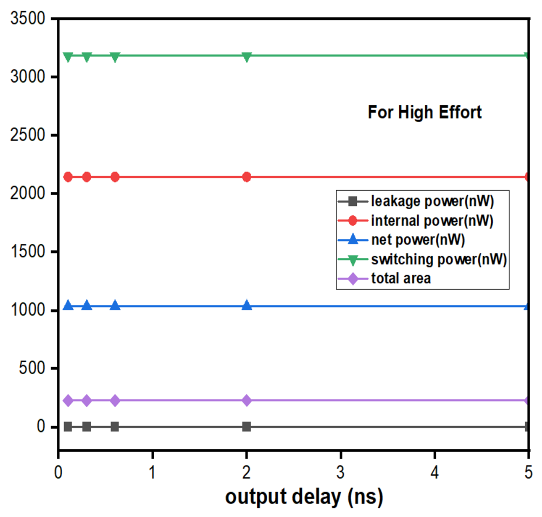

For Low & Medium Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power (nW)
total area
output delay (ns)


3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8
setup time (ns)
For High Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power(nW)
total area

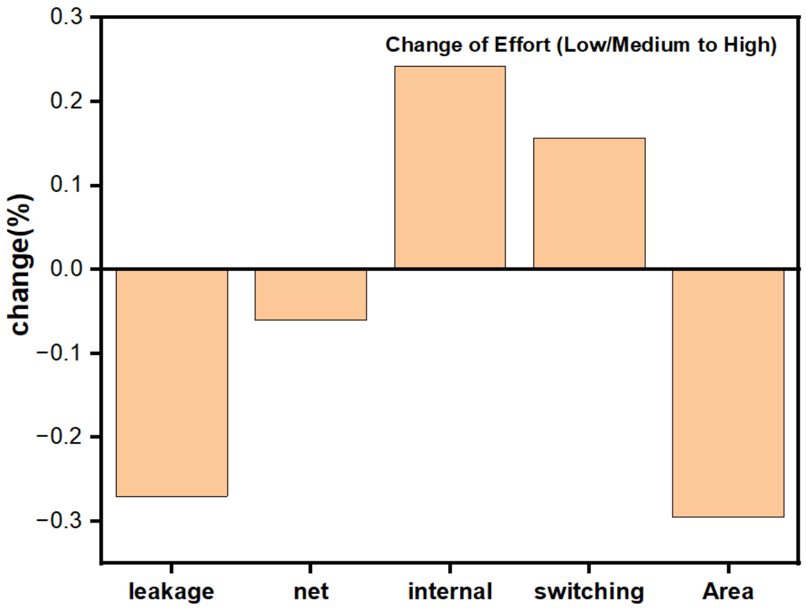

hold time (ns)
For High Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power(nW)
total area


clock freq (MHz)
For High Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power(nW)
total area


input delay (ns)
For High Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power(nW)
total area


output delay (ns)
For High Effort
leakage power(nW)
internal power(nW)
net power(nW)
switching power(nW)
total area

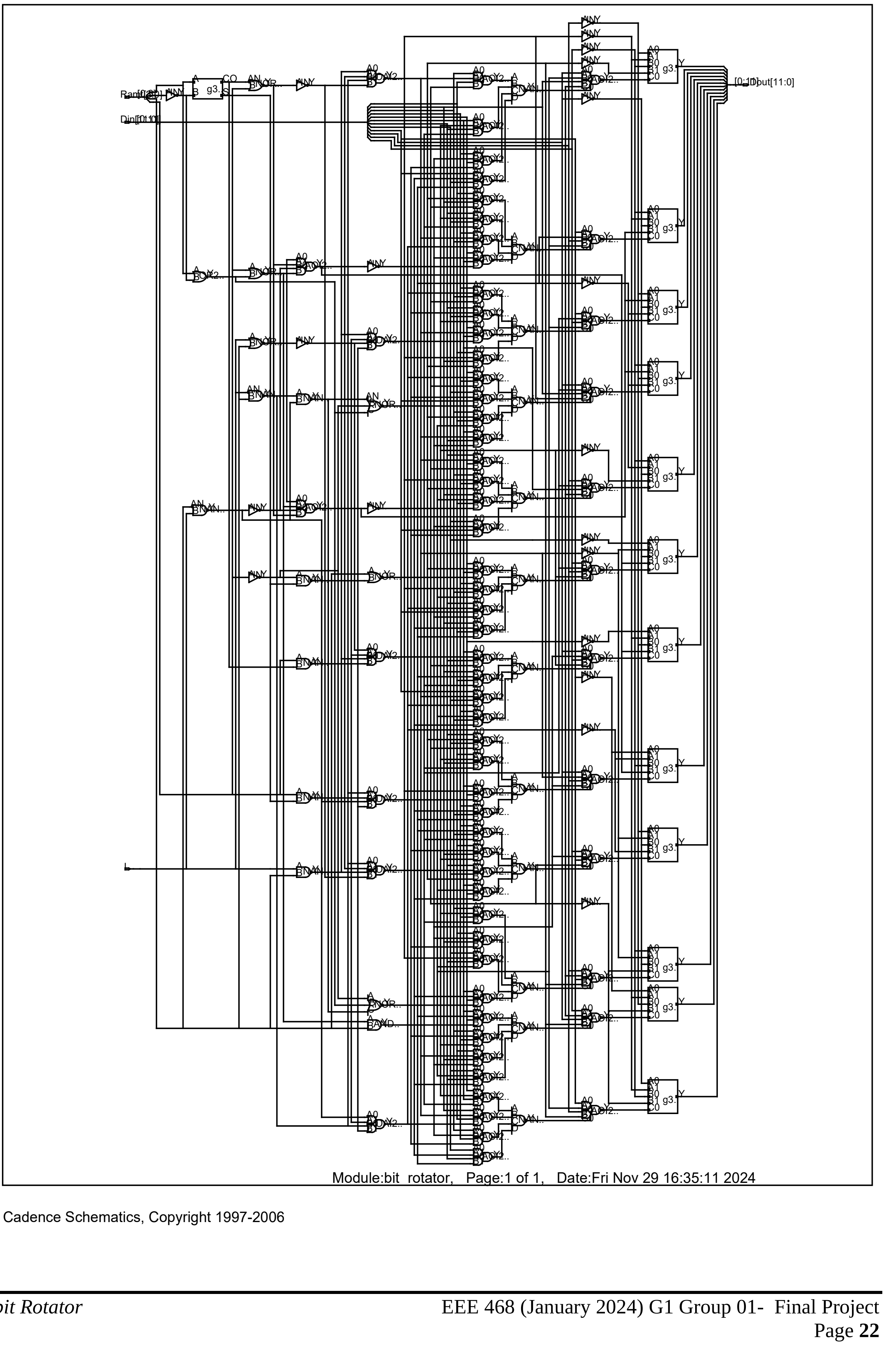
Here setup and hold time are scaled over $\frac{1}{T_{Period}}$. Here, we can observe that P_{switch} = P_{net} + $P_{internal}$. $P_{leakage}$ is independent. No change is occurring in every plot due to the absence of clock frequency. But change is observed when effort is changed from low/medium to high.



Here, we can see that in high effort, $P_{interal}$ increases and that's why P_{switch} also increases. But the other parameters (P_{leak}, P_{net}, Area) decrease in high effort. That's why we chose the high effort **bit_rotator_synth.v** for physical design.

6.6 Synthesized Design

Our Synthesized Design:


Rot[0:0]
Din[11:0]
Dout[11:0]
L
g3
Module:bit_rotator, Page:1 of 1, Date:Fri Nov 29 16:35:11 2024

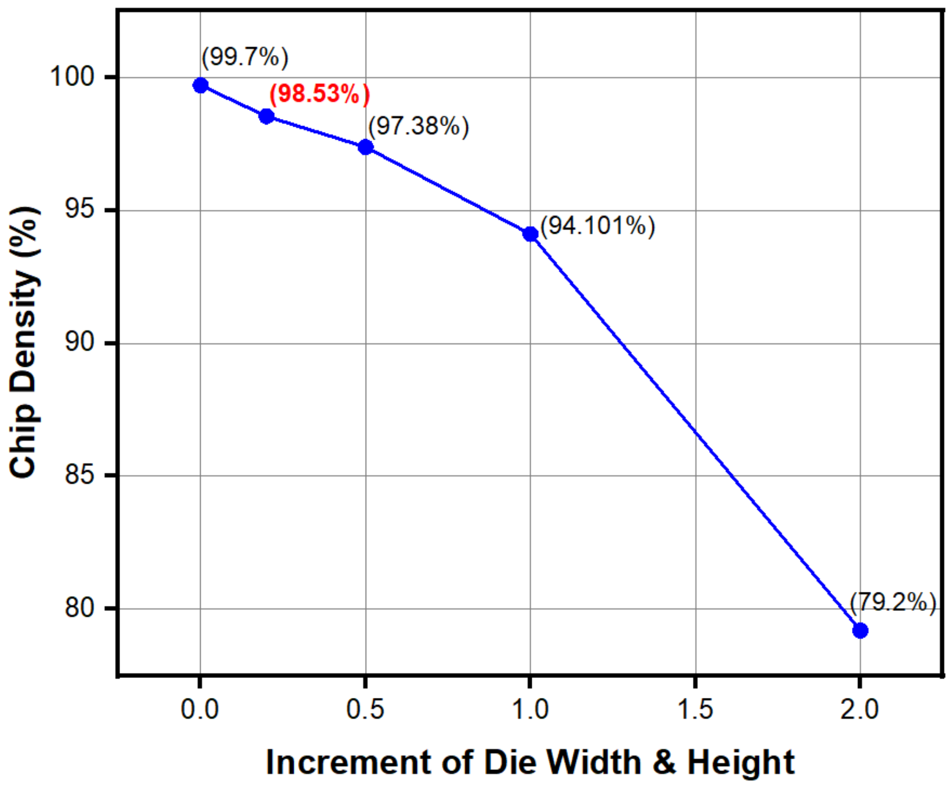
Cadence Schematics, Copyright 1997-2006

7 Step-4 (Optimized Physical Design with respect to Die Size, Pin Planning, and Power Rails):

Here we focused on maximum chip density by varying die size, pin planning and power rails. The suggested die **width** and **height** are **19.145** and **17.1**. We equally incremented these values by an equal amount and listed a table.

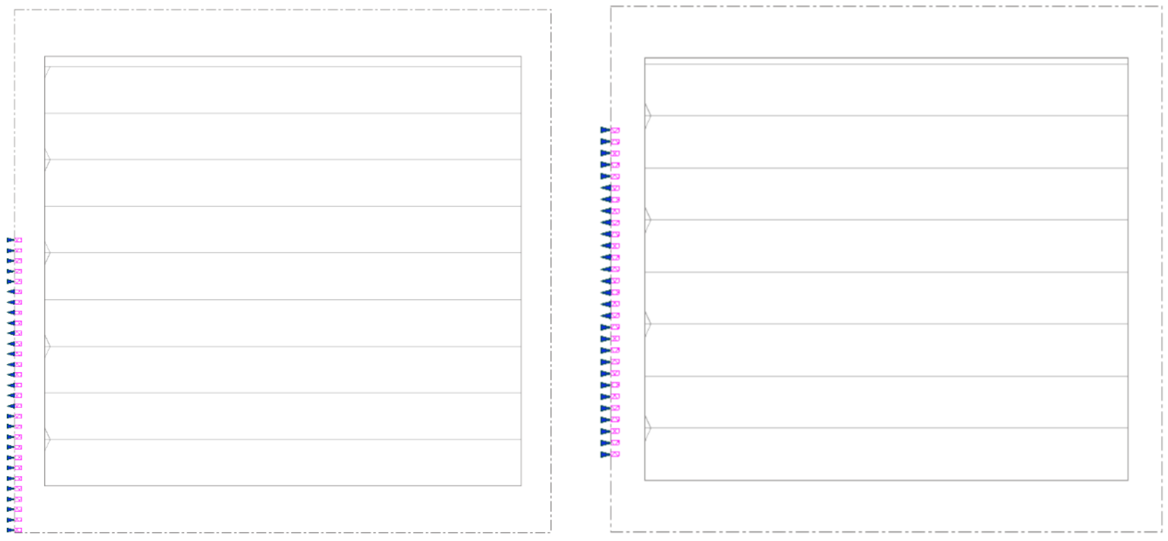
Increment Value	Die Width	Die Height
+2	21.145	19.1
+1	20.145	18.1
+0.5	19.645	17.6
+0.2	**19.345**	**17.3**
0	19.145	17.1

For this, we got the following chip density (%) vs increment value plot:

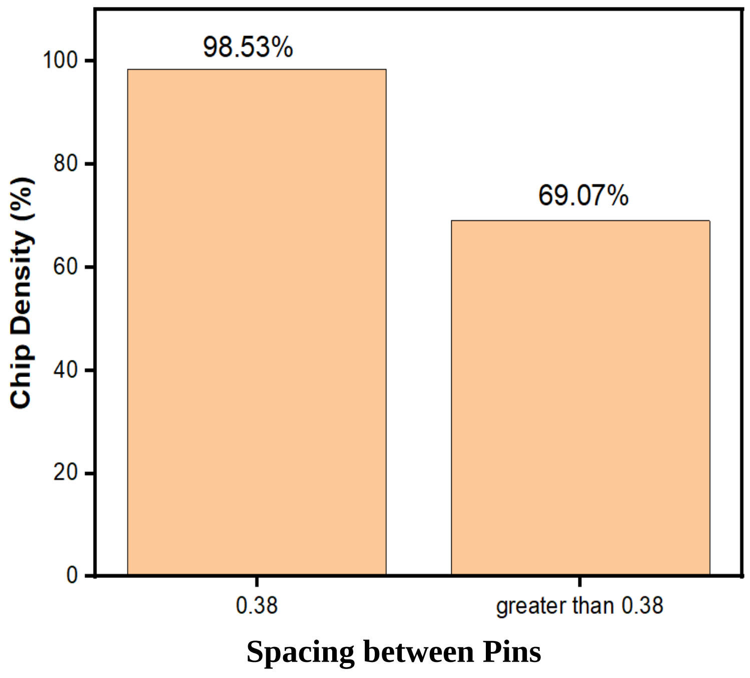


Although the maximum density is 99.7% for our design, we used the combination where we get **98.53%** because the first one fails in DRC checking and all other combinations pass in DRC. For our design, we chose die width = 19.345 and die height = 17.3.

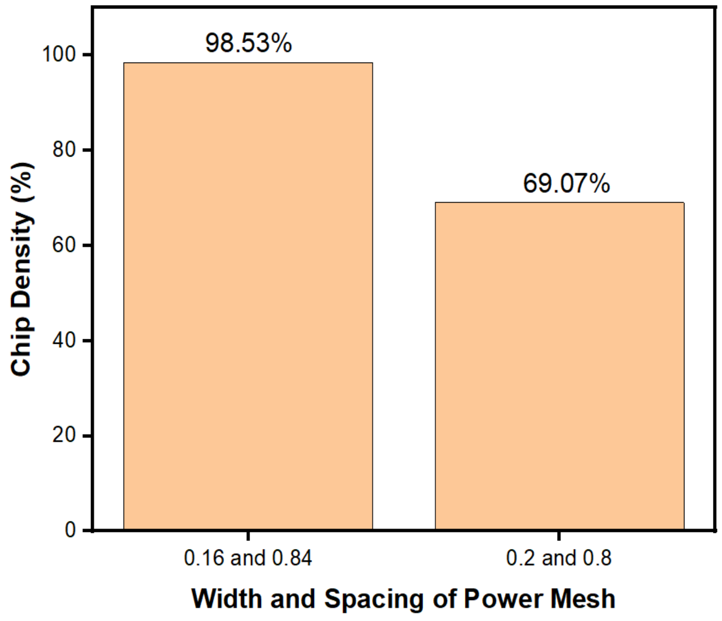
In case of pin planning position, it has no effect on chip density. For example, the following pictures of pin planning introduce zero effect on chip density.



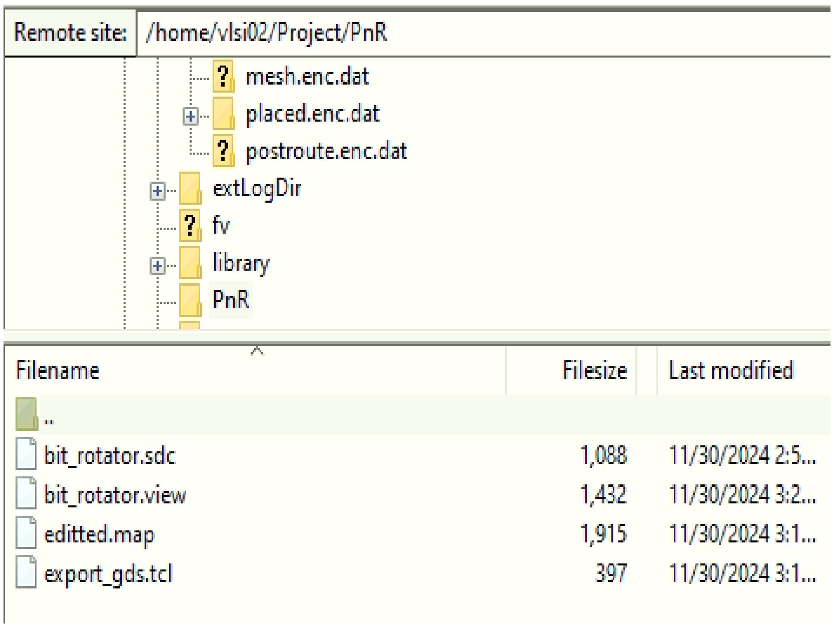
Now if spacing between the pins in the pin planning is greater than **0.38**, a value of **0.57** is auto-adjusted in the setting, and it introduces a huge drop in chip density. So, we chose **0.38** as the minimum spacing.



If the width and spacing in the power mesh are taken as **0.2** and **0.8** instead of **0.16** and **0.84** , a significant drop in chip density occurs. So, in power mesh, **0.16** and **0.84** are taken as power mesh width and spacing between them.



Now step by step processes are given with figures during the physical design. The **bit_rotator_synth.v** file is in the **/home/vlsi02/Project** directory. In the **/home/vlsi02/Project/PnR** directory, these files will be used in the PnR process.



```
Remote site: /home/vlsi02/Project/PnR
        mesh.enc.dat
        placed.enc.dat
        postroute.enc.dat
    extLogDir
    fv
    library
    PnR

Filename                          Filesize   Last modified
..
bit_rotator.sdc                      1,088   11/30/2024 2:5...
bit_rotator.view                     1,432   11/30/2024 3:2...
editted.map                          1,915   11/30/2024 3:1...
export_gds.tcl                         397   11/30/2024 3:1...
```

The .sdc file will be the same that we used in step-3 and editted.map file will also be same that we used in our experiment 4. Modifications are required in the **bit_rotator.view** and **export_gds.tcl**

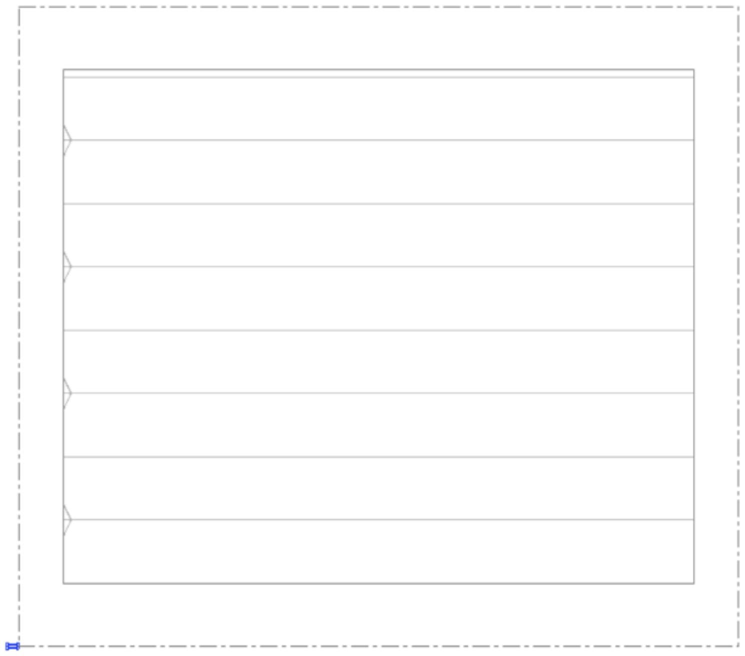
bit_rotator.view:

```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_rc_corner -name rcbest0 -T {0} -preRoute_res {1.0} -preRoute_cap {1.0} -
preRoute_clkres {1.0} -preRoute_clkcap {1.0} -postRoute_res {1.0} -postRoute_cap
{1.0} -postRoute_xcap {1.0} -postRoute_clkres {1.0} -postRoute_clkcap {1.0} -
qx_tech_file {/home/cad/VLSI2Lab/Digital/library/gpdk045.tch}
create_rc_corner -name rcworst125 -T {125} -preRoute_res {1.0} -preRoute_cap
{1.0} -preRoute_clkres {1.0} -preRoute_clkcap {1.0} -postRoute_res {1.0} -
postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres {1.0} -
postRoute_clkcap {1.0} -qx_tech_file
{/home/cad/VLSI2Lab/Digital/library/gpdk045.tch}
create_library_set -name BC -timing
{/home/cad/VLSI2Lab/Digital/library/fast_vdd1v0_basicCells.lib}
create_library_set -name WC -timing
{/home/cad/VLSI2Lab/Digital/library/slow_vdd1v0_basicCells.lib}
create_constraint_mode -name func -sdc_files
{/home/vlsi02/Project/PnR/bit_rotator.sdc}
create_delay_corner -name BC_rcbest0.hold -library_set {BC} -rc_corner {rcbest0}
create_delay_corner -name WC_rcworst125.setup -library_set {WC} -rc_corner
{rcworst125}
create_analysis_view -name func@BC_rcbest0.hold -constraint_mode {func} -
delay_corner {BC_rcbest0.hold}
create_analysis_view -name func@WC_rcworst125.setup -constraint_mode {func} -
delay_corner {WC_rcworst125.setup}
set_analysis_view -setup {func@BC_rcbest0.hold} -hold {func@BC_rcbest0.hold}
```

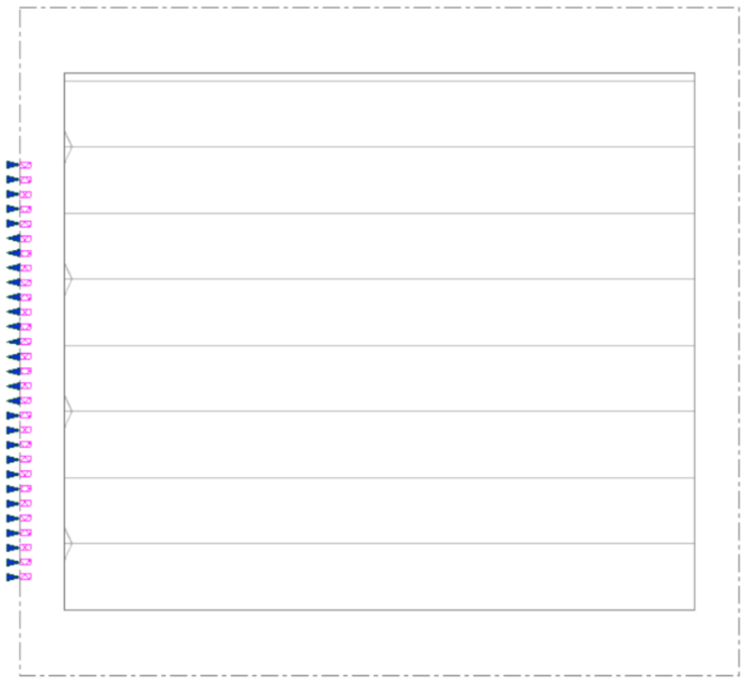
export_gds.tcl:

```
# Write Netlist
saveNetlist [dbGet [dbgTopCell].name].lvs_netlist.vg \
    -flat \
    -includePowerGround \
    -phys \
    -excludeLeafCell

# Write GDS
streamOut bit_rotator.merged.gds -mapFile /home/vlsi02/Project/PnR/editted.map -
mode ALL -units 2000 -merge /home/cad/VLSI2Lab/Digital/library/gsclib045.gds -
dieAreaAsBoundary -stripes 1 -structureName bit_rotator -libName DesignLib
```

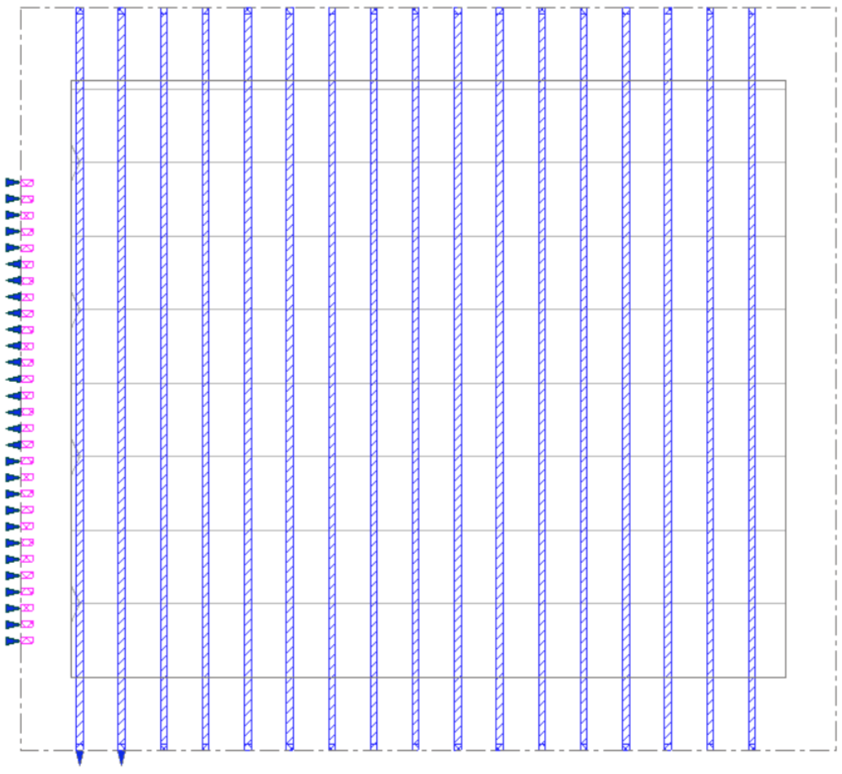
Using these, after floorplanning we got the following figure:



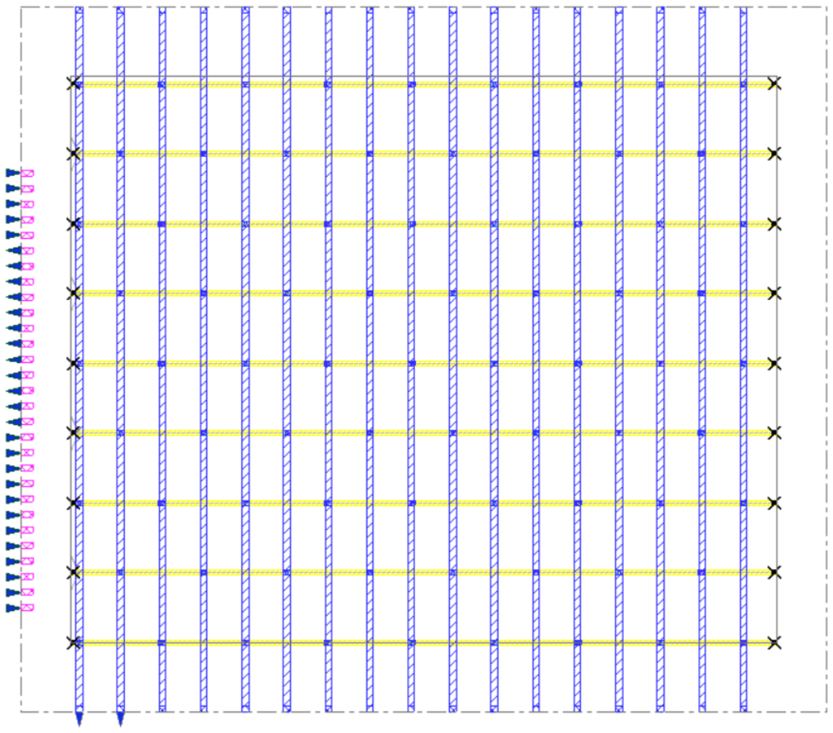
After that, using proper settings in pin editor as we discussed before, the figure will look like:



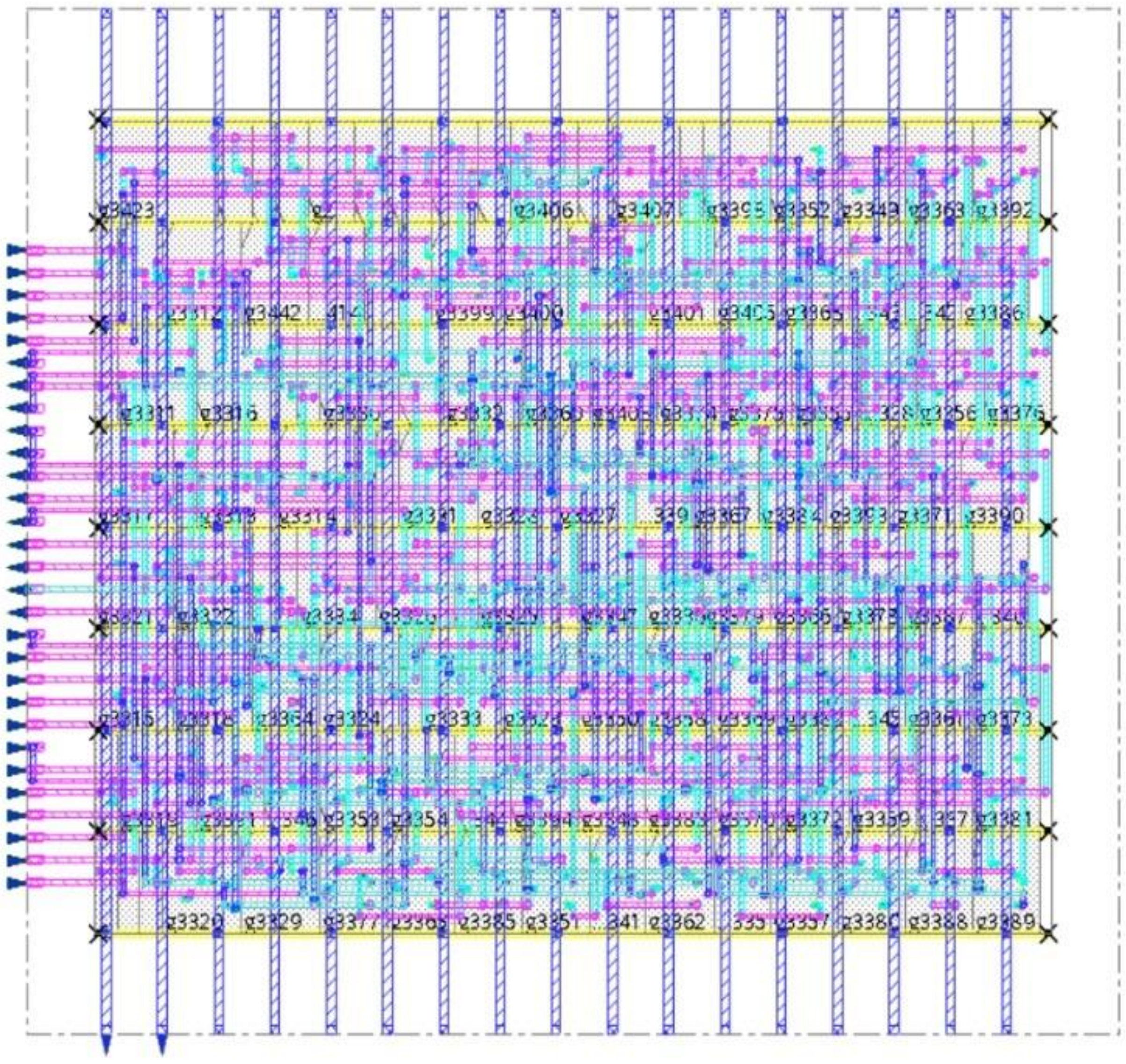
After creating the power mesh using proper settings:



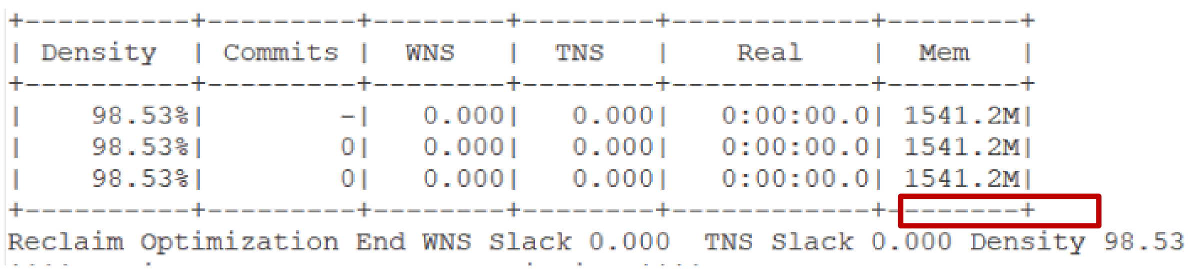
After adding M1 layer rail in the design we got:



Then after placement process:

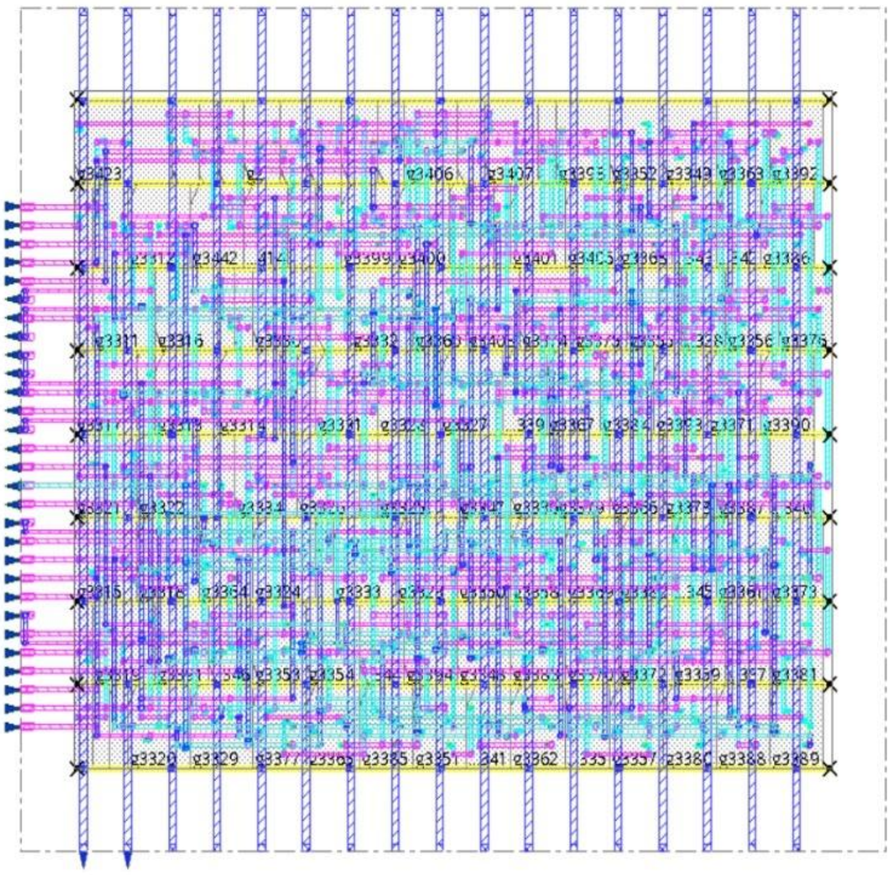


Density Data:

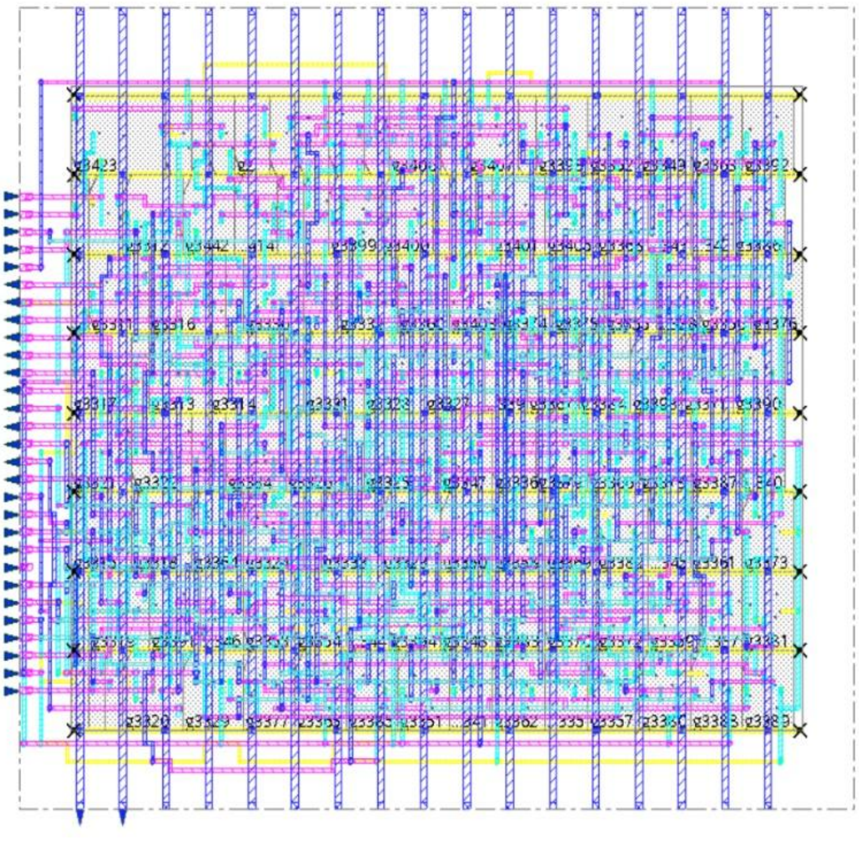


```
+----------+---------+--------+--------+------------+--------+
| Density  | Commits |  WNS   |  TNS   |    Real    |  Mem   |
+----------+---------+--------+--------+------------+--------+
|    98.53%|        -|   0.000|   0.000|   0:00:00.0| 1541.2M|
|    98.53%|        0|   0.000|   0.000|   0:00:00.0| 1541.2M|
|    98.53%|        0|   0.000|   0.000|   0:00:00.0| 1541.2M|
+----------+---------+--------+--------+------------+--------+
Reclaim Optimization End WNS Slack 0.000  TNS Slack 0.000 Density 98.53
```

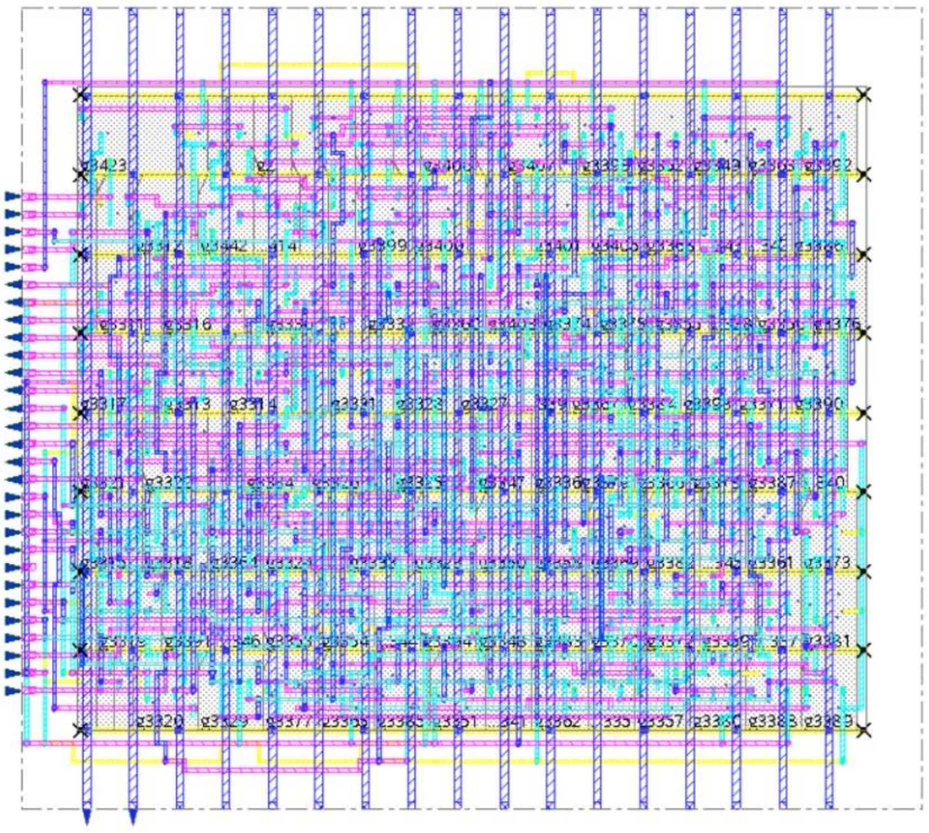
After CTS:



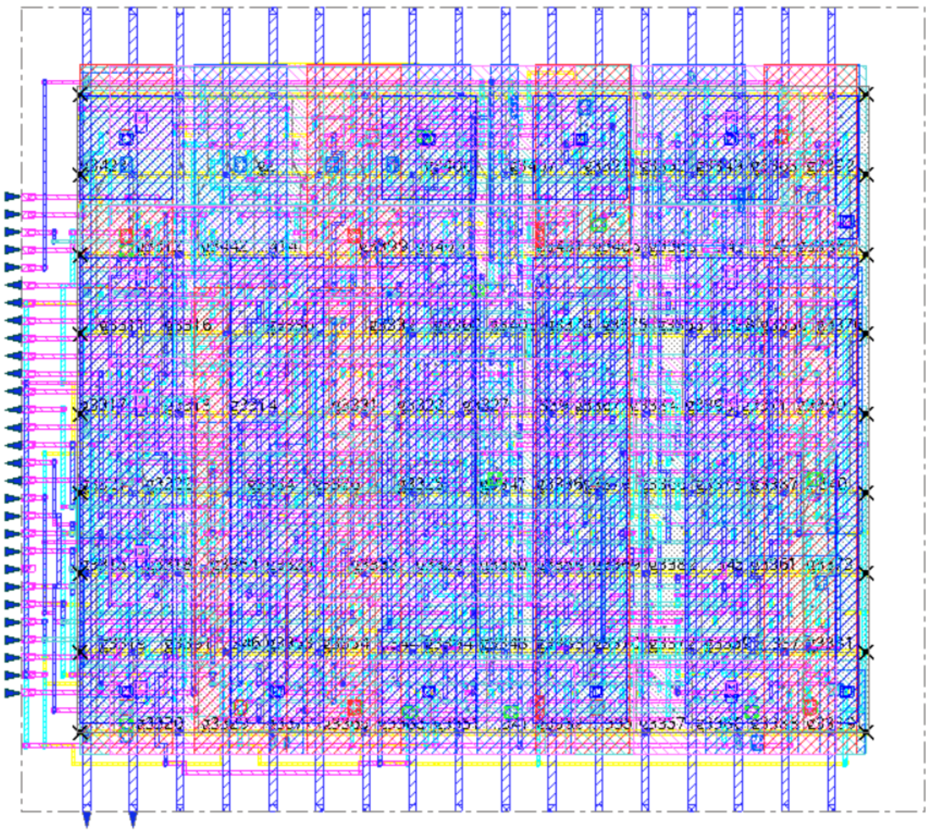
Then using nanorouting:



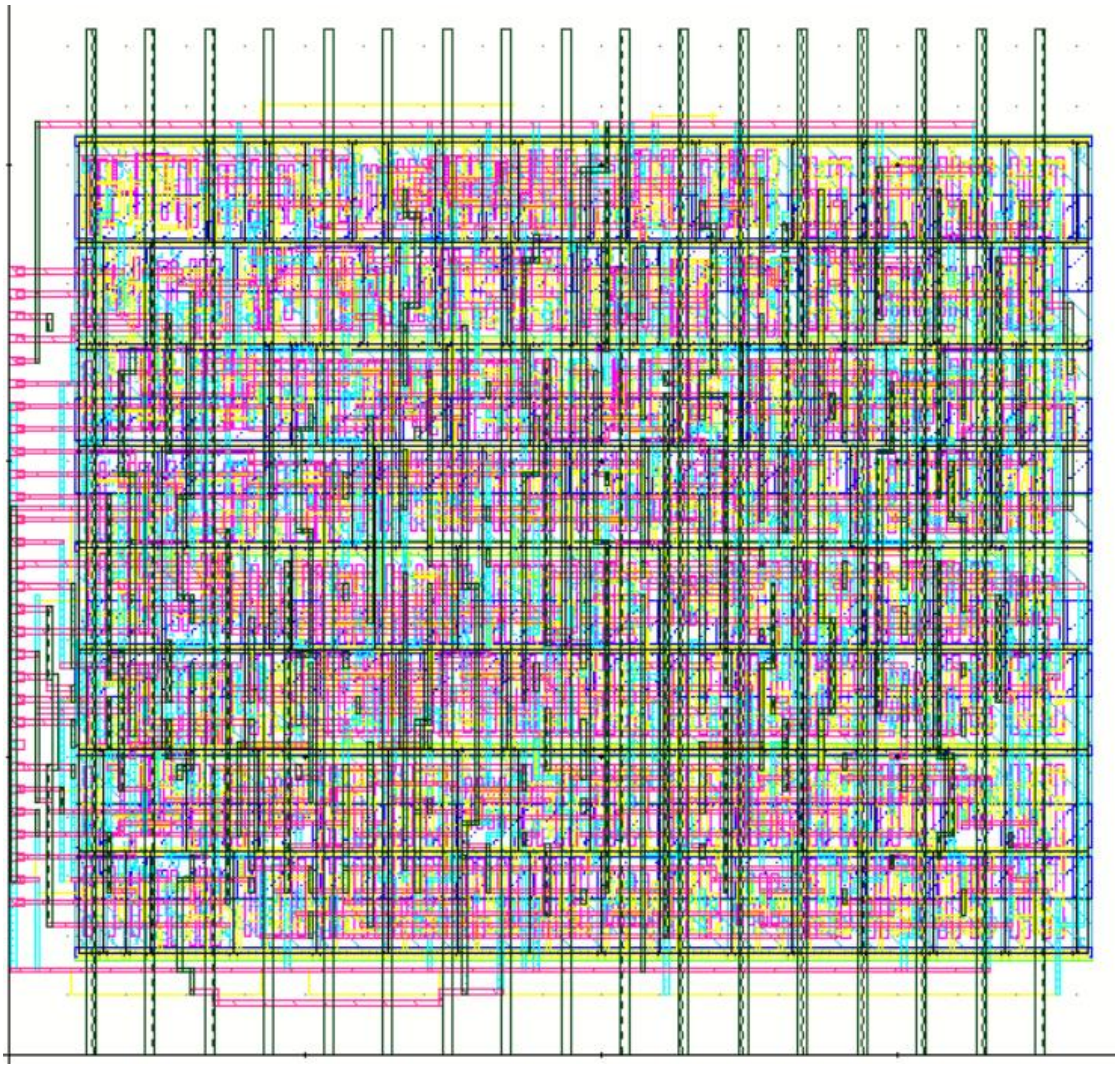
After post route timing and SI optimization, we got:



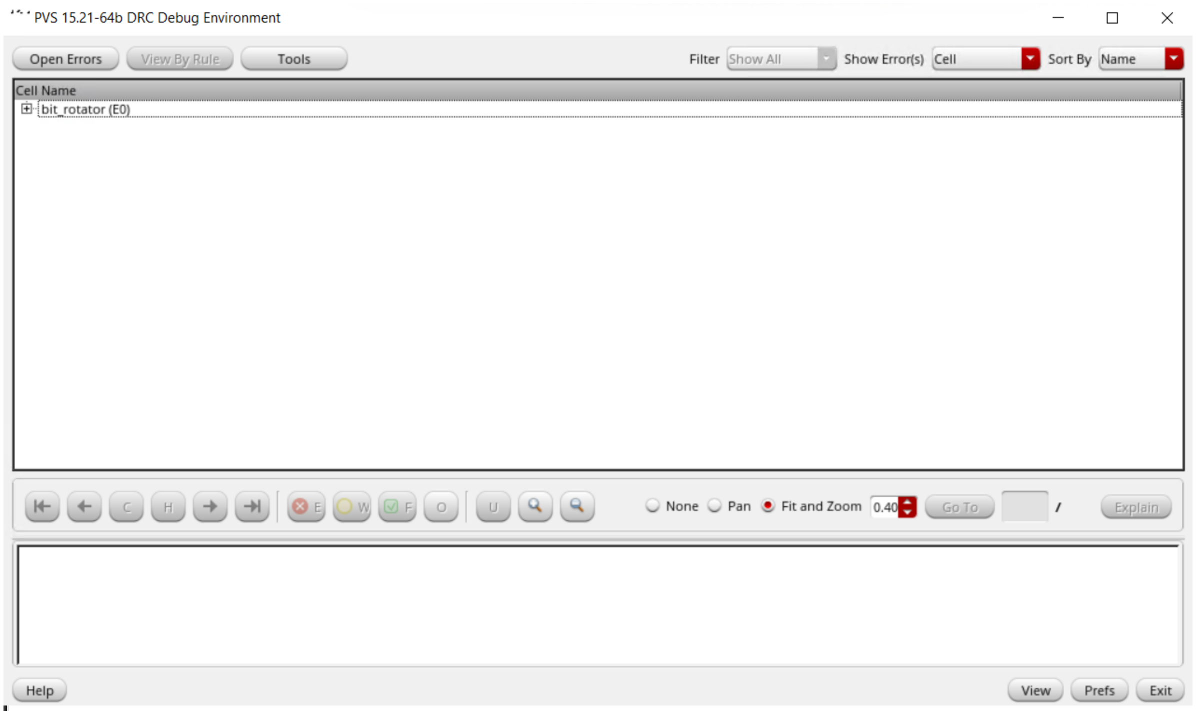
After adding metal fill:



7.1 Final design:



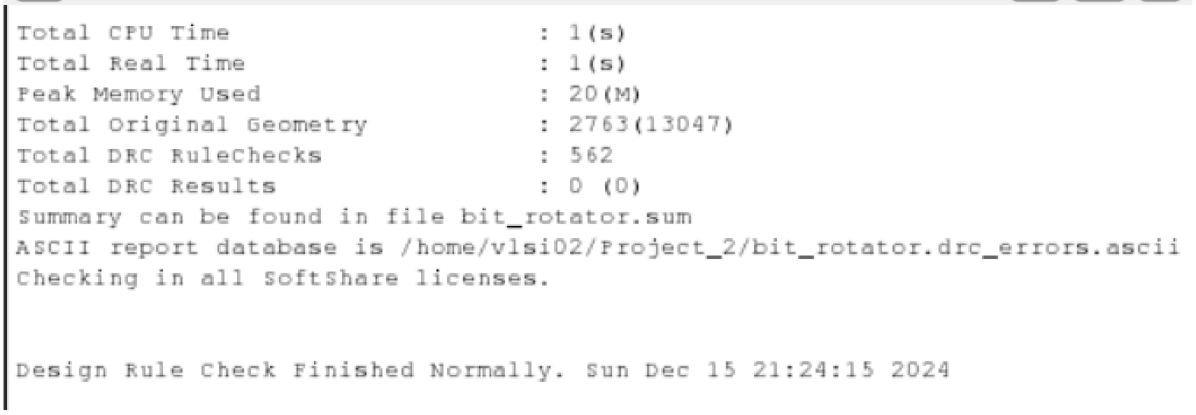
7.2 DRC result:



Filter: Show All Show Error(s): Cell Sort By: Name

Cell Name
bit_rotator (E0)

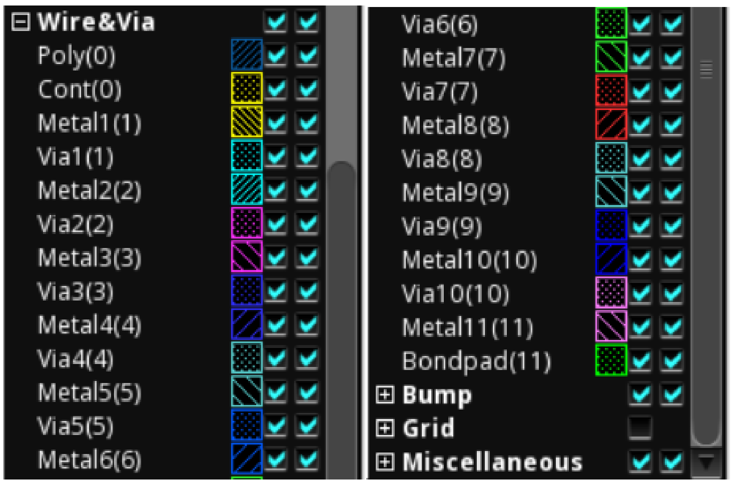
None Pan Fit and Zoom 0.40



```
Total CPU Time                    : 1(s)
Total Real Time                   : 1(s)
Peak Memory Used                  : 20(M)
Total Original Geometry           : 2763(13047)
Total DRC RuleChecks              : 562
Total DRC Results                 : 0 (0)
Summary can be found in file bit_rotator.sum
ASCII report database is /home/vlsi02/Project_2/bit_rotator.drc_errors.ascii
Checking in all SoftShare licenses.


Design Rule Check Finished Normally. Sun Dec 15 21:24:15 2024
```

Thus, the physical design was completed successfully.

7.3 Layers with different colors that are used:



8 Conclusion:

In this project, we successfully implemented the concepts of RTL coding, directed and layered testbench with coverage, optimization of synthesis and physical design. We understood the various parameters that influence the synthesis and physical design process. We have got a maximum chip density of **98.53%** which can conclude that our design is an optimized design.

9 Links of Codes and Files:

Step-1: https://edaplayground.com/x/BwGa
Step-2: https://edaplayground.com/x/8rfp
Step-3 & Step-4: /home/vlsi02/Project directory, /home/vlsi02/Project_2 directory

10 Contribution:

1906001:

- Module Design
- Cross-checking in directed and layered testbench
- Adding coverage block to check coverage

1906004:

- Physical design of synthesized design with Innovus
- Optimizing design area to achieve more chip density
- Optimizing pin planning and power mesh to achieve more chip density

1906026:

- Optimizing hold time, input delay
- Collaborating in physical design, testbench.
- Report Writing, Presentation making

1906031:

- Synthesis with Cadence Genus
- Optimizing setup time, hold time, input delay, output delay, clock frequency
- Report writing and generating the plots