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

EEE 468

VLSI Laboratory

Final Project Report

Section: G2 Group: 01

16 Bit Ripple Carry Adder with Generate-Propagate Logic

Course Instructors:

Nafis Sadik
Rafid Hasan Palash

Signature of Instructor: _____

Academic Honesty Statement:

"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: Dipika Rani Nath** **Student ID: 1906092**	**Signature:** _____ **Full Name: Zahin Ikbal Zeal** **Student ID: 1906101**
Signature: _____ **Full Name: Jannatul Ferdous** **Student ID: 1906103**	**Signature:** _____ **Full Name: Fahim Ahmed** **Student ID: 1906110**

Table of Contents

1 Abstract

This project focuses on designing a 16-bit ripple carry adder (RCA) using generate-propagate logic implemented in SystemVerilog. The adder's functionality is verified through a self-checking testbench and a layered testbench methodology, ensuring robust verification. The circuit is synthesized using Cadence tools, followed by physical design (Place and Route, PnR) and Design Rule Check (DRC) validation. This comprehensive approach ensures the circuit's correctness & performance,showcasing a complete design and verification flow.

2 Introduction

Addition is a fundamental operation in digital systems, and the ripple carry adder (RCA) is one of the simplest and most commonly used adder architectures. This project aims to implement a 16-bit RCA optimized with generate-propagate logic to improve its efficiency. The use of SystemVerilog allows for a modular and scalable design, while verification through self-checking and layered testbenches ensures that the design meets functional requirements under various test scenarios. Post-RTL, the design undergoes synthesis, physical design (PnR), and Design Rule Check (DRC) using Cadence tools. These steps ensure the design's readiness for fabrication, adhering to performance constraints.

3 Design

3.1 Design Method

In Generate-Propagate logic.for ith Bit,

$G_i = A_i \& B_i$
$P_i = A_i \wedge B_i$
$C_i = G_i \mid P_i \& C_{i-1}$
$S_i = P_i \wedge C_{i-1}$

So the carry of ith stage is only available after the completion of (i-1)th carry, which is essentially the concept of carry ripple adder.

We introduced **clock** to **synchronize** the output and so our synthesized circuit will have both combinational and sequential parts.

3.2 System Verilog Codes

Design Code

```
module rippleadder(a,b,cin,sum,cout,clk);

  input [15:0]a,b;
  input cin,clk;
  output reg [15:0]sum;
  output reg cout;
  reg [15:0] g,p,c;

  always@(posedge clk)

              begin

                 g=a&b;
                 p=a^b;

                 c[0]=g[0]|p[0]&cin;
                 c[1]=g[1]|p[1]&c[0];
                 c[2]=g[2]|p[2]&c[1];
                 c[3]=g[3]|p[3]&c[2];
                 c[4]=g[4]|p[4]&c[3];
                 c[5]=g[5]|p[5]&c[4];
                 c[6]=g[6]|p[6]&c[5];
                 c[7]=g[7]|p[7]&c[6];
                 c[8]=g[8]|p[8]&c[7];
                 c[9]=g[9]|p[9]&c[8];
                 c[10]=g[10]|p[10]&c[9];
                 c[11]=g[11]|p[11]&c[10];
                 c[12]=g[12]|p[12]&c[11];
                 c[13]=g[13]|p[13]&c[12];
                 c[14]=g[14]|p[14]&c[13];
                 cout=g[15]|p[15]&c[14];

                 sum[0]=p[0]^cin;
                 sum[1]=p[1]^c[0];
                 sum[2]=p[2]^c[1];
                 sum[3]=p[3]^c[2];
                 sum[4]=p[4]^c[3];
                 sum[5]=p[5]^c[4];
                 sum[6]=p[6]^c[5];
                 sum[7]=p[7]^c[6];
                 sum[8]=p[8]^c[7];
                 sum[9]=p[9]^c[8];
                 sum[10]=p[10]^c[9];
                 sum[11]=p[11]^c[10];
                 sum[12]=p[12]^c[11];
                 sum[13]=p[13]^c[12];
                 sum[14]=p[14]^c[13];
                 sum[15]=p[15]^c[14];

              end

endmodule
```

Self-Checking Testbench Code

```
module adderstimulus;
  reg clk, cin;
  reg [15:0] a, b;
  wire [15:0] sum;
  wire cout;

  initial
    forever #5 clk = ~clk;

  adder DUT (.clk (clk),
             .a (a),
             .b (b),
             .sum (sum),
             .cin (cin),
             .cout (cout)
            );
```

```
  initial begin
    clk = 1'b0;
    a = 120;
    b = 245;
    cin = 1;

    #10
    result_checker(cin, a, b, sum, cout);

    #10
    a = 12555;
    b = 45895;
    cin = 0;

    #10
    result_checker(cin, a, b, sum, cout);

    #10
    a = 55562;
    b = 20511;
    cin = 1;

    #10
    result_checker(cin, a, b, sum, cout);

    #10
    a = 251;
    b = 1548;
    cin = 0;

    #10
    result_checker(cin, a, b, sum, cout);

    $finish;
  end

  task result_checker(input cin, input [15:0] a, b, input [15:0] sum, input cout);
    reg [15:0] expected_sum;
    reg expected_cout;
    begin
      {expected_cout, expected_sum} = a + b + cin;
      if (expected_cout == cout && expected_sum == sum)
        $display("Passed : a=%d b=%d cin=%d sum=%d expected sum=%d cout=%d expected cout=%d",
                 a, b, cin, sum, expected_sum, cout, expected_cout);
      else
        $display("Failed : a=%d b=%d cin=%d sum=%d expected sum=%d cout=%d expected cout=%d",
                 a, b, cin, sum, expected_sum, cout, expected_cout);
    end
  endtask

  initial begin
    $dumpfile("aludump.vcd");
    $dumpvars;
  end
endmodule
```

Layered Testbench Code

transaction.sv

```
class transaction;
  rand bit [15:0] a;
  rand bit [15:0] b;
  rand bit cin;
  bit [15:0] sum;
  bit cout;

endclass:transaction
```

interface.sv

```
interface adder_if(input clk);
  logic [15:0] a,b;
  logic cin;
  logic [15:0] sum;
  logic cout;

  clocking driver_cb @(negedge clk);
    default input #1 output #1;
    output a,b,cin;
  endclocking

  clocking mon_cb @(negedge clk);
    default input #1 output #1;
    input a,b,cin;
    input sum,cout;
  endclocking

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

endinterface
```

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
```

driver.sv

```
class driver;
  mailbox gen2driv, driv2sb;
  virtual adder_if.DRIVER adderif;
  transaction d_trans;
  event driven;
  function new(mailbox gen2driv, driv2sb , virtual adder_if.DRIVER adderif, event driven);
    this.gen2driv=gen2driv;
    this.adderif=adderif;
    this.driven=driven;
    this.driv2sb=driv2sb;
  endfunction
  task main(input int count);
    repeat(count) begin
      d_trans=new();
      gen2driv.get(d_trans);

      @(adderif.driver_cb);
      adderif.driver_cb.a <= d_trans.a;
      adderif.driver_cb.b <= d_trans.b;
      adderif.driver_cb.cin <= d_trans.cin;
      driv2sb.put(d_trans);
      -> driven;
    end
  endtask:main
endclass:driver
```

monitor.sv

```
class monitor;
  mailbox mon2sb;
  virtual adder_if.MONITOR adderif;
  transaction m_trans;
  event driven;

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

  task main(input int count);
    @(driven);
    @(adderif.mon_cb);
    repeat(count) begin
      m_trans=new();
      @(posedge adderif.clk);
      m_trans.sum=adderif.mon_cb.sum;
      m_trans.cout=adderif.mon_cb.cout;
      mon2sb.put(m_trans);
    end
  endtask:main
endclass:monitor
```

scoreboard.sv

```
class scoreboard;
  mailbox driv2sb;
  mailbox mon2sb;

  transaction d_trans;
  transaction m_trans;

  event driven;

  function new(mailbox driv2sb, mon2sb);
    this.driv2sb=driv2sb;
    this.mon2sb=mon2sb;
  endfunction

  task main(input int count);
    $display("------------------Scoreboard Test Starts--------------------");
    repeat(count) begin
      m_trans=new();
      mon2sb.get(m_trans);
      report();

      if((m_trans.sum != d_trans.sum)||(m_trans.cout != d_trans.cout))
        $display("Failed : a=%d b=%d cin=%d  Expected sum=%d  Resulted sum=%d Expected cout=%d  Resulted
cout=%d",d_trans.a,d_trans.b, d_trans.cin,d_trans.sum, m_trans.sum,d_trans.cout, m_trans.cout);
      else
        $display("Passed : a=%d b=%d cin=%d  Expected sum=%d  Resulted sum=%d Expected cout=%d  Resulted
cout=%d",d_trans.a,d_trans.b, d_trans.cin,d_trans.sum, m_trans.sum,d_trans.cout, m_trans.cout);

    end
    $display("------------------Scoreboard Test Ends--------------------");
  endtask:main

  task report();
    d_trans=new();
    driv2sb.get(d_trans);

    {d_trans.cout,d_trans.sum}=d_trans.a+d_trans.b+d_trans.cin;


  endtask:report


endclass:scoreboard
```

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 adder_if adderif;

  function new(virtual adder_if adderif);
    this.adderif=adderif;
    gen2driv=new();
    driv2sb=new();
    mon2sb=new();

    gen=new(gen2driv);
    drv=new(gen2driv,driv2sb,adderif.DRIVER,driven);
    mon=new(mon2sb,adderif.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

endclass:environment
```

testcase01.sv

```
`include "environment.sv"

program test(input int count, adder_if adderif);
  environment env;

  class testcase01 extends transaction;

  endclass:testcase01

  initial begin
    testcase01 testcase01handle;
    testcase01handle=new();

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


endprogram:test
```

testbench.sv

```
`include "testcase01.sv"
`include "interface.sv"
module testbench;
  bit clk;
  initial begin
    forever #5 clk =~clk;
  end
  int count=15;
  adder_if adderif(clk);
  test test01(count,adderif);
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
  end
  rippleadder DUT (
    .a(adderif.a),
    .b(adderif.b),
    .cin(adderif.cin),
    .sum(adderif.sum),
    .cout(adderif.cout),
    .clk(clk)
  );
endmodule
```

Code for coverage

```
module tb();
  reg [15:0] a, b;
  reg cin;
  wire [15:0] sum;
  wire cout;
  reg clk;
  rippleadder DUT (
    .cout(cout),
    .a(a),
    .b(b),
    .sum(sum),
    .cin(cin),
    .clk(clk)
  );
  initial begin
    forever #5 clk = ~clk;
  end
  covergroup cg;

        option.per_instance = 1;
        A : coverpoint a;
        B : coverpoint b;
        C : coverpoint cin;
        S : coverpoint sum;
        Co: coverpoint cout;
  endgroup: cg
  cg cg_h;
  initial begin
    cg_h = new();
    clk = 0;
    repeat(400)
      begin
        a = $random;
        b = $random;
        cin = $random;
        #10;
        cg_h.sample();
        reportt(a, b, sum, cout, cin);
        #10;
      end
  end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
  end
  initial begin
    #8000;
    $finish;
  end
  task reportt(input [15:0] a, input [15:0] b, input [15:0] sum, input cout, input cin);
    reg [15:0] expected_sum;
    reg expected_cout;
    {expected_cout, expected_sum} = a + b + cin;
    if (cout == expected_cout && sum == expected_sum)
      $display("Passed : time=%t a= %d b=%d cin=%d sum=%d expected sum=%d cout=%d expected cout=%d", $time, a, b, cin, sum,
expected_sum, cout, expected_cout);
    else
      $display("Failed : time=%t a= %d b=%d cin=%d sum=%d expected sum=%d cout=%d expected cout=%d", $time, a, b, cin, sum,
expected_sum, cout, expected_cout);
  endtask
endmodule
```

4 Results from Testbench

4.1 Self-Checking Testbench

```
[vlsi21@CadenceServer3 ~/cds_digital]$ irun adder.v alutest.v
irun(64): 15.10-s015: (c) Copyright 1995-2016 Cadence Design Systems, Inc.
Recompiling... reason: file './alutest.v' is newer than expected.
        expected: Thu Nov 28 11:39:46 2024
        actual:   Thu Nov 28 11:51:52 2024
file: alutest.v
        module digital_lib.adderstimulus:v
                errors: 0, warnings: 0
                Caching library 'digital_lib' ....... Done
        Elaborating the design hierarchy:
        Top level design units:
                adderstimulus
        Building instance overlay tables: ..................... Done
        Generating native compiled code:
                digital_lib.adderstimulus:v <0x2a3bc2c6>
                        streams:   4, words:  9162
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                         Instances  Unique
                Modules:          2       2
                Registers:       16      16
                Scalar wires:     3       -
                Vectored wires:   3       -
                Always blocks:    1       1
                Initial blocks:   3       3
        Writing initial simulation snapshot: digital_lib.adderstimulus:v
Loading snapshot digital_lib.adderstimulus:v ..................... Done
ncsim> source /home/eda/cadence/lnx/INCSIVE/icd/icdcm_t1b_016/flow/INCISIV/INCISIV151/15.10.015/lnx86/tools/inca/files/ncsimrc
ncsim> run
ncsim: *W,DVEXACC: some objects excluded from $dumpvars due to access restrictions, use +access+r on command line for access to all objects.
           File: ./alutest.v, line = 72, pos = 8
          Scope: adderstimulus
           Time: 0 FS + 0

Passed : a=  120 b=  245 cin= 1 sum=   366 expected sum =   366 cout= 0 expected cout=0
Passed : a=12555 b=45895 cin= 0 sum= 58450 expected sum = 58450 cout= 0 expected cout=0
Passed : a=55562 b=20511 cin= 1 sum= 10538 expected sum = 10538 cout= 1 expected cout=1
Passed : a=  251 b= 1548 cin= 0 sum=  1799 expected sum =  1799 cout= 0 expected cout=0
Simulation complete via $finish(1) at time 70 NS + 0
./alutest.v:47 $finish;
ncsim> exit
[vlsi21@CadenceServer3 ~/cds_digital]$
```

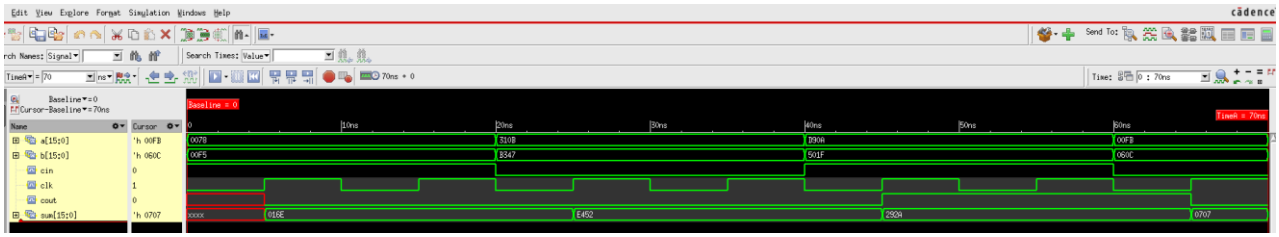

Figure 1: Waveforms from Testbench

4.2 Layered Testbench

```
.. Done
        Design hierarchy summary:
                                  Instances  Unique
                Modules:                  2       2
                Interfaces:               1       1
                Programs:                 1       1
                Registers:               57      84
                Scalar wires:             5       -
                Vectored wires:           5       -
                Named events:             2       6
                Always blocks:            1       1
                Initial blocks:           4       4
                Clocking blocks:          2       2
                Clocking items:           8       8
                Parallel blocks:          1       1
                Pseudo assignments:       6       6
                Assertions:               1       1
                Compilation units:        1       1
                SV Class declarations:    7       7
                SV Class specializations: 7       7
        Writing initial simulation snapshot: worklib.testben
ch:sv
```

```
ncsim> run
------------------Scoreboard Test Starts--------------------
Passed : a=55922 b= 1686 cin=1  Expected sum=57609  Resulted
 sum=57609 Expected cout=0  Resulted cout=0
Passed : a=63267 b=14694 cin=1  Expected sum=12426  Resulted
 sum=12426 Expected cout=1  Resulted cout=1
Passed : a=50935 b=40541 cin=1  Expected sum=25941  Resulted
 sum=25941 Expected cout=1  Resulted cout=1
Passed : a=55368 b= 9315 cin=0  Expected sum=64683  Resulted
 sum=64683 Expected cout=0  Resulted cout=0
Passed : a=29737 b=26602 cin=1  Expected sum=56340  Resulted
 sum=56340 Expected cout=0  Resulted cout=0
Passed : a=25821 b=65446 cin=0  Expected sum=25731  Resulted
 sum=25731 Expected cout=1  Resulted cout=1
Passed : a=32982 b=40250 cin=1  Expected sum= 7697  Resulted
 sum= 7697 Expected cout=1  Resulted cout=1
Passed : a=31103 b=27189 cin=0  Expected sum=58292  Resulted
 sum=58292 Expected cout=0  Resulted cout=0
Passed : a=26015 b=29607 cin=0  Expected sum=55622  Resulted
 sum=55622 Expected cout=0  Resulted cout=0
Passed : a=25189 b=53238 cin=0  Expected sum=12891  Resulted
 sum=12891 Expected cout=1  Resulted cout=1
Passed : a=18295 b= 6623 cin=1  Expected sum=24919  Resulted
 sum=24919 Expected cout=0  Resulted cout=0
Passed : a=   25 b=51694 cin=0  Expected sum=51719  Resulted
 sum=51719 Expected cout=0  Resulted cout=0
Passed : a=59471 b=39366 cin=1  Expected sum=33302  Resulted
 sum=33302 Expected cout=1  Resulted cout=1
Passed : a=44985 b=24816 cin=1  Expected sum= 4266  Resulted
 sum= 4266 Expected cout=1  Resulted cout=1
Passed : a=23428 b=21101 cin=1  Expected sum=44530  Resulted
 sum=44530 Expected cout=0  Resulted cout=0
------------------Scoreboard Test Ends--------------------
Simulation complete via $finish(1) at time 165 NS + 1
../testbench/environment.sv:39     $finish;
ncsim> exit
[vlsi21@CadenceServer3 run]$
```

4.3 Coverage Results

For 16 bit variables, the interval of 1024 was taken and checked if testcase hit within the range or not.If testcases hit all the ranges/intervals, coverage is 100%.

```
CUMULATIVE SUMMARY
=============================================
|    Coverage Type    | Weight | Hits/Total |
=============================================
| Covergroup Coverage |      1 |   100.000% |
|---------------------|--------|------------|
| Types               |        |      1 / 1 |
=============================================
CUMULATIVE DESIGN-BASED COVERAGE: 100.000%
COVERED DESIGN UNITS: 1 / 1
FILES: 1
```

```
SUMMARY
=================================================================================
|    Coverage Type    | Weight | Local Hits/Total | Recursive Hits/Total |
=================================================================================
| Covergroup Coverage |      1 |         100.000% |             100.000% |
|---------------------|--------|------------------|----------------------|
| Types               |        |            1 / 1 |                1 / 1 |
=================================================================================
WEIGHTED AVERAGE LOCAL: 100.000%
WEIGHTED AVERAGE RECURSIVE: 100.000%
```

```
COVERGROUP COVERAGE
============================================================
|        Covergroup         |   Hits   |  Goal /  | Status  |
|                           |          | At Least |         |
============================================================
| TYPE /tb/cg               | 100.000% | 100.000% | Covered |
============================================================
| INSTANCE <UNNAMED1>       | 100.000% | 100.000% | Covered |
|---------------------------|----------|----------|---------|
| COVERPOINT <UNNAMED1>::A  | 100.000% | 100.000% | Covered |
|---------------------------|----------|----------|---------|
```

COVERPOINT <UNNAMED1>::A	100.000%	100.000%	Covered
bin auto[0:1023]	9	1	Covered
bin auto[1024:2047]	6	1	Covered
bin auto[2048:3071]	10	1	Covered
bin auto[3072:4095]	5	1	Covered
bin auto[4096:5119]	7	1	Covered
bin auto[5120:6143]	6	1	Covered
bin auto[6144:7167]	4	1	Covered
bin auto[7168:8191]	6	1	Covered
bin auto[8192:9215]	8	1	Covered
bin auto[9216:10239]	4	1	Covered
bin auto[10240:11263]	7	1	Covered
bin auto[11264:12287]	7	1	Covered
bin auto[12288:13311]	4	1	Covered
bin auto[13312:14335]	7	1	Covered
bin auto[14336:15359]	9	1	Covered
bin auto[15360:16383]	4	1	Covered
bin auto[16384:17407]	4	1	Covered
bin auto[17408:18431]	4	1	Covered
bin auto[18432:19455]	4	1	Covered
bin auto[19456:20479]	7	1	Covered
bin auto[20480:21503]	6	1	Covered
bin auto[21504:22527]	8	1	Covered
bin auto[22528:23551]	7	1	Covered
bin auto[23552:24575]	7	1	Covered
bin auto[24576:25599]	12	1	Covered
bin auto[25600:26623]	6	1	Covered
bin auto[26624:27647]	7	1	Covered
bin auto[35840:36863]	6	1	Covered
bin auto[36864:37887]	2	1	Covered
bin auto[37888:38911]	5	1	Covered
bin auto[38912:39935]	10	1	Covered
bin auto[39936:40959]	6	1	Covered
bin auto[40960:41983]	2	1	Covered
bin auto[41984:43007]	10	1	Covered
bin auto[43008:44031]	6	1	Covered
bin auto[44032:45055]	7	1	Covered
bin auto[45056:46079]	3	1	Covered
bin auto[46080:47103]	10	1	Covered
bin auto[47104:48127]	4	1	Covered
bin auto[48128:49151]	11	1	Covered
bin auto[49152:50175]	8	1	Covered
bin auto[50176:51199]	3	1	Covered
bin auto[51200:52223]	6	1	Covered
bin auto[52224:53247]	1	1	Covered
bin auto[53248:54271]	8	1	Covered
bin auto[54272:55295]	6	1	Covered
bin auto[55296:56319]	9	1	Covered
bin auto[56320:57343]	3	1	Covered
bin auto[57344:58367]	8	1	Covered
bin auto[58368:59391]	8	1	Covered
bin auto[59392:60415]	5	1	Covered
bin auto[60416:61439]	6	1	Covered
bin auto[61440:62463]	6	1	Covered
bin auto[62464:63487]	6	1	Covered
bin auto[63488:64511]	7	1	Covered
bin auto[64512:65535]	7	1	Covered

COVERPOINT <UNNAMED1>::C	100.000%	100.000%	Covered
bin auto[0]	185	1	Covered
bin auto[1]	215	1	Covered

COVERPOINT <UNNAMED1>::B	100.000%	100.000%	Covered
bin auto[0:1023]	12	1	Covered
bin auto[1024:2047]	8	1	Covered
bin auto[2048:3071]	4	1	Covered
bin auto[3072:4095]	14	1	Covered
bin auto[4096:5119]	8	1	Covered
bin auto[5120:6143]	6	1	Covered
bin auto[6144:7167]	8	1	Covered
bin auto[7168:8191]	4	1	Covered
bin auto[8192:9215]	4	1	Covered
bin auto[9216:10239]	10	1	Covered
bin auto[10240:11263]	1	1	Covered
bin auto[11264:12287]	7	1	Covered
bin auto[12288:13311]	5	1	Covered
bin auto[13312:14335]	5	1	Covered
bin auto[14336:15359]	4	1	Covered
bin auto[15360:16383]	4	1	Covered
bin auto[16384:17407]	9	1	Covered
bin auto[17408:18431]	5	1	Covered
bin auto[18432:19455]	5	1	Covered
bin auto[19456:20479]	11	1	Covered
bin auto[20480:21503]	8	1	Covered
bin auto[21504:22527]	5	1	Covered
bin auto[22528:23551]	6	1	Covered
bin auto[23552:24575]	8	1	Covered
bin auto[24576:25599]	7	1	Covered
bin auto[25600:26623]	8	1	Covered
bin auto[26624:27647]	6	1	Covered
bin auto[35840:36863]	4	1	Covered
bin auto[36864:37887]	12	1	Covered
bin auto[37888:38911]	4	1	Covered
bin auto[38912:39935]	6	1	Covered
bin auto[39936:40959]	6	1	Covered
bin auto[40960:41983]	6	1	Covered
bin auto[41984:43007]	5	1	Covered
bin auto[43008:44031]	7	1	Covered
bin auto[44032:45055]	8	1	Covered
bin auto[45056:46079]	6	1	Covered
bin auto[46080:47103]	5	1	Covered
bin auto[47104:48127]	5	1	Covered
bin auto[48128:49151]	3	1	Covered
bin auto[49152:50175]	5	1	Covered
bin auto[50176:51199]	5	1	Covered
bin auto[51200:52223]	3	1	Covered
bin auto[52224:53247]	4	1	Covered
bin auto[53248:54271]	2	1	Covered
bin auto[54272:55295]	6	1	Covered
bin auto[55296:56319]	4	1	Covered
bin auto[56320:57343]	6	1	Covered
bin auto[57344:58367]	8	1	Covered
bin auto[58368:59391]	8	1	Covered
bin auto[59392:60415]	4	1	Covered
bin auto[60416:61439]	6	1	Covered
bin auto[61440:62463]	7	1	Covered
bin auto[62464:63487]	9	1	Covered
bin auto[63488:64511]	6	1	Covered
bin auto[64512:65535]	6	1	Covered

COVERPOINT <UNNAMED1>::Co	100.000%	100.000%	Covered
bin auto[0]	208	1	Covered
bin auto[1]	192	1	Covered

==========

COVERPOINT <UNNAMED1>::S	100.000%	100.000%	Covered
bin auto[0:1023]	4	1	Covered
bin auto[1024:2047]	6	1	Covered
bin auto[2048:3071]	6	1	Covered
bin auto[3072:4095]	5	1	Covered
bin auto[4096:5119]	5	1	Covered
bin auto[5120:6143]	3	1	Covered
bin auto[6144:7167]	6	1	Covered
bin auto[7168:8191]	11	1	Covered
bin auto[8192:9215]	4	1	Covered
bin auto[9216:10239]	6	1	Covered
bin auto[10240:11263]	6	1	Covered
bin auto[11264:12287]	4	1	Covered
bin auto[12288:13311]	4	1	Covered
bin auto[13312:14335]	9	1	Covered
bin auto[14336:15359]	7	1	Covered
bin auto[15360:16383]	8	1	Covered
bin auto[16384:17407]	4	1	Covered
bin auto[17408:18431]	8	1	Covered
bin auto[18432:19455]	6	1	Covered
bin auto[19456:20479]	12	1	Covered
bin auto[20480:21503]	9	1	Covered
bin auto[21504:22527]	4	1	Covered
bin auto[22528:23551]	7	1	Covered
bin auto[23552:24575]	11	1	Covered
bin auto[24576:25599]	6	1	Covered
bin auto[25600:26623]	5	1	Covered
bin auto[26624:27647]	12	1	Covered
bin auto[35840:36863]	5	1	Covered
bin auto[36864:37887]	2	1	Covered
bin auto[37888:38911]	6	1	Covered
bin auto[38912:39935]	6	1	Covered
bin auto[39936:40959]	3	1	Covered
bin auto[40960:41983]	4	1	Covered
bin auto[41984:43007]	8	1	Covered
bin auto[43008:44031]	6	1	Covered
bin auto[44032:45055]	10	1	Covered
bin auto[45056:46079]	5	1	Covered
bin auto[46080:47103]	7	1	Covered
bin auto[47104:48127]	11	1	Covered
bin auto[48128:49151]	6	1	Covered
bin auto[49152:50175]	5	1	Covered
bin auto[50176:51199]	5	1	Covered
bin auto[51200:52223]	5	1	Covered
bin auto[52224:53247]	3	1	Covered
bin auto[53248:54271]	13	1	Covered
bin auto[54272:55295]	9	1	Covered
bin auto[55296:56319]	3	1	Covered
bin auto[56320:57343]	7	1	Covered
bin auto[57344:58367]	2	1	Covered
bin auto[58368:59391]	4	1	Covered
bin auto[59392:60415]	6	1	Covered
bin auto[60416:61439]	5	1	Covered
bin auto[61440:62463]	4	1	Covered
bin auto[62464:63487]	10	1	Covered
bin auto[63488:64511]	5	1	Covered
bin auto[64512:65535]	6	1	Covered

A total of **400 test case** was required for 100% coverage in that testbench.

5 Synthesis & Optimization of Area, Power Consumption

5.1 Synthesis

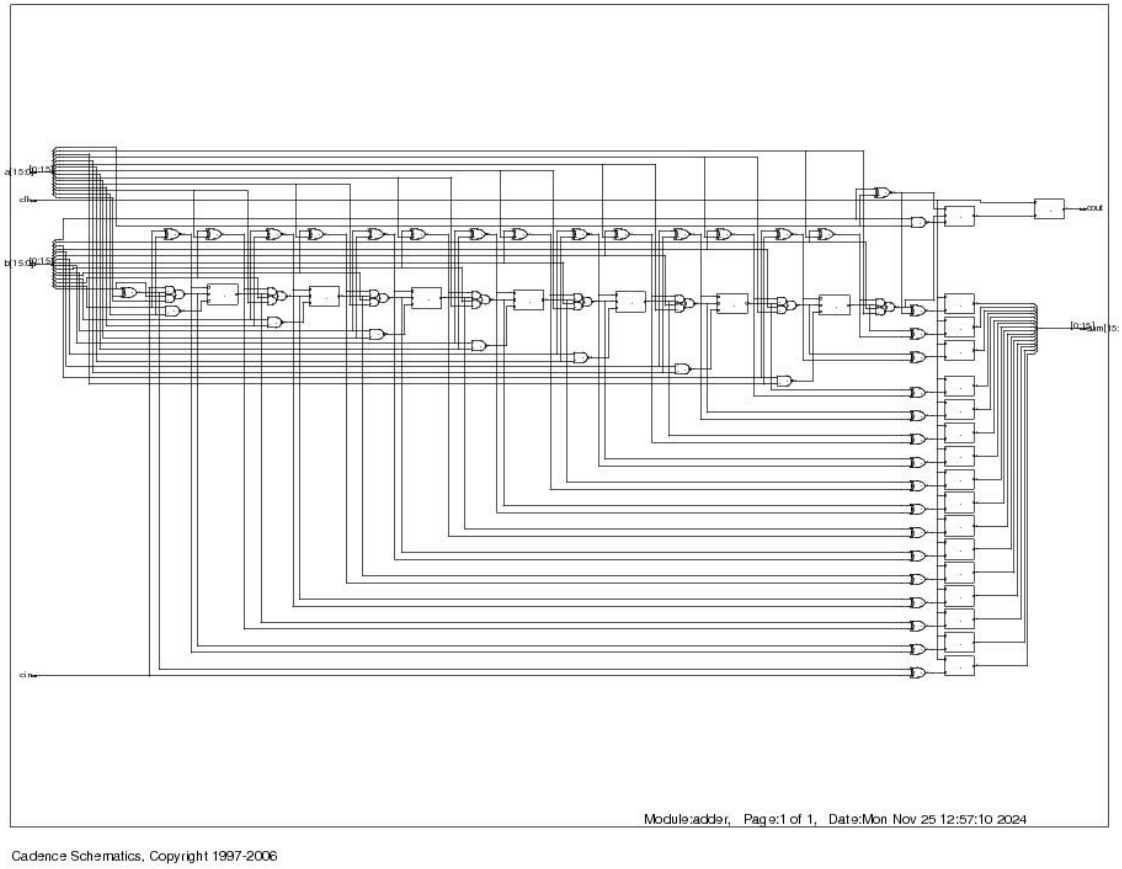

Figure 2: Synthesized Circuit from Cadence

5.2 Optimization

We optimized our design in medium effort as it provides the best optimization in both area and power consumption and delay. Setup time, hole time, clock period, input & output delay was varried for optimization.

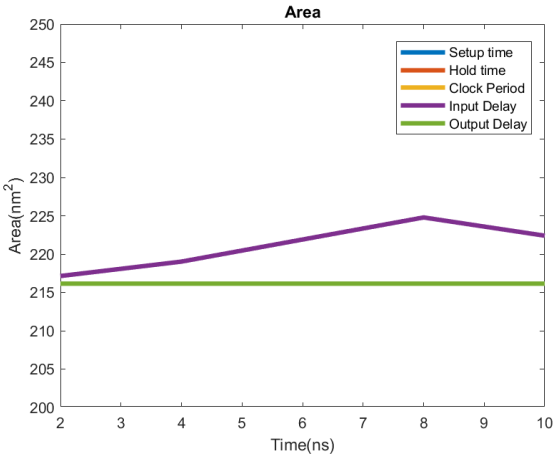

Figure 3: Area Optimization

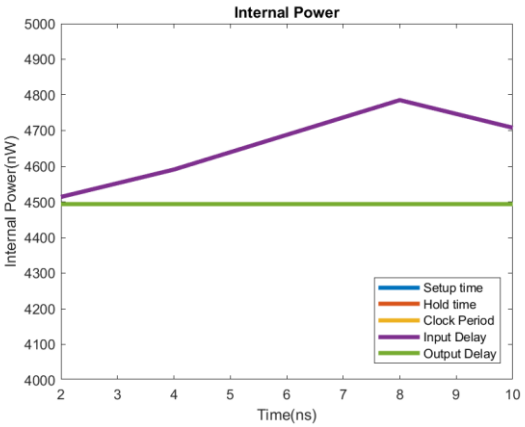

Figure 4: Internal Power Optimization

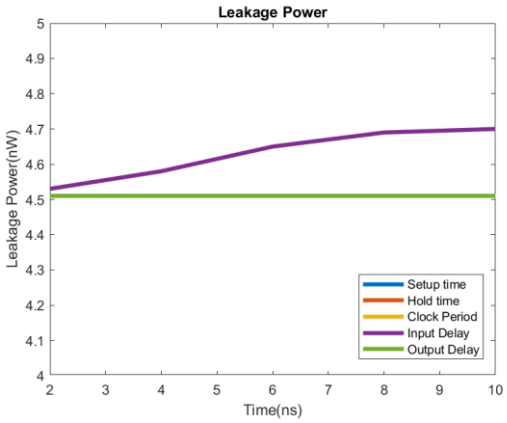

Figure 5: Leakage Power Optimization

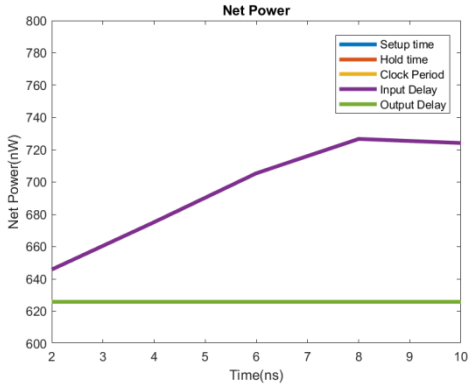

Figure 6: Net Power Optimization

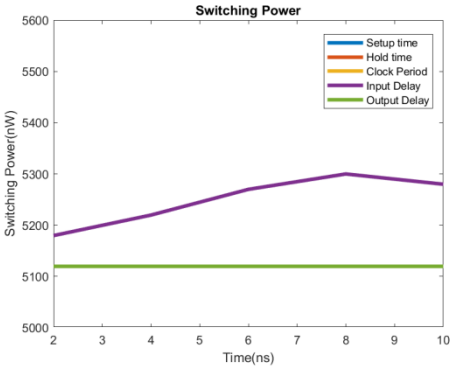

Figure 7: Switching Power Optimization

5.3 Optimized Results

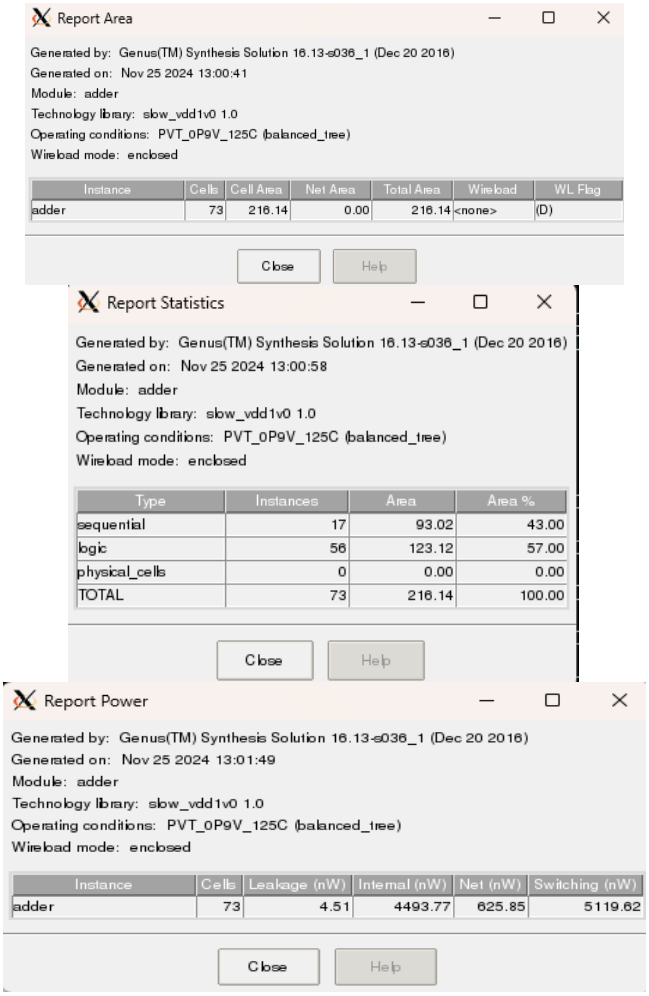

Figure 8: Area & Power after Optimization

6 PnR ,CTS, Density & DRC Test

6.1 CTS

```
+-------------+-------+---------+---------+
| Setup mode  | all   | reg2reg | default |
+-------------+-------+---------+---------+
| WNS (ns):   | 0.000 | 0.95    | 0.001   |
| TNS (ns):   | 0.000 | 0       | 0.000   |
| Violating Paths: | 0 | 0     | 0       |
| All Paths:  | 17    | 17      | 0       |
+-------------+-------+---------+---------+

+-------------+-------+---------+---------+----------+
| DRVs        | Real  |         |         | Total    |
+-------------+-------+---------+---------+----------+
|             | Nr nets(terms) | Worst Vio | Nr nets(terms) |
+-------------+----------------+-----------+----------------+
| max_cap     | 0 (0)          | 0.000     | 0 (0)          |
| max_tran    | 0 (0)          | 0.000     | 0 (0)          |
| max_fanout  | 0 (0)          | 0.000     | 0 (0)          |
| max_length  | 0 (0)          | 0.000     | 0 (0)          |
+-------------+----------------+-----------+----------------+
```

We have total 17 sequential path as there is total 17 outputs (1 Cout and 16m bits) from 17 Flip Flops.

6.2 Density

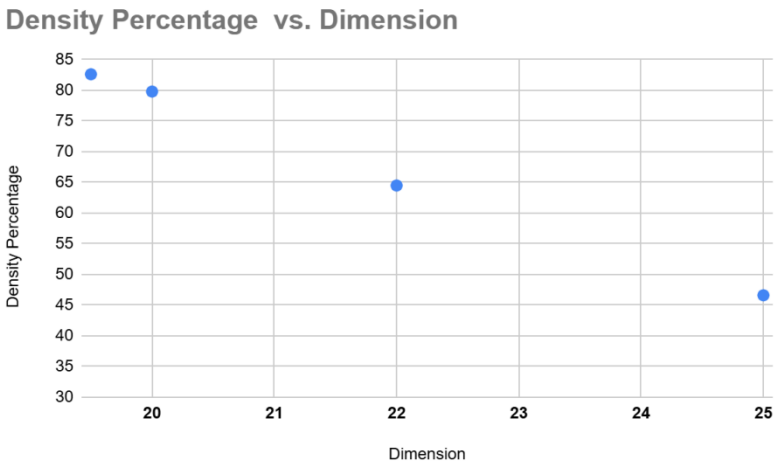

Figure 9: Density vs. Dimension

```
--------+

Density: 82.614%
Total number of glitch violations: 0
-------------------------------------------------------------
**optDesign ... cpu = 0:00:08, real = 0:00:09, mem = 1621.
, totSessionCpu=0:07:33 **
 ReSet Options after AAE Based Opt flow
*** Finished optDesign ***
Removing temporary dont_use automatically set for cells wi
 technology sites with no row.
```

Figure 10: Optimized Density

This is the maximum density we achieved by reducing dimension with 0 DRC error. If we lower the dimension more, DRC errors appear.

6.3 Layout

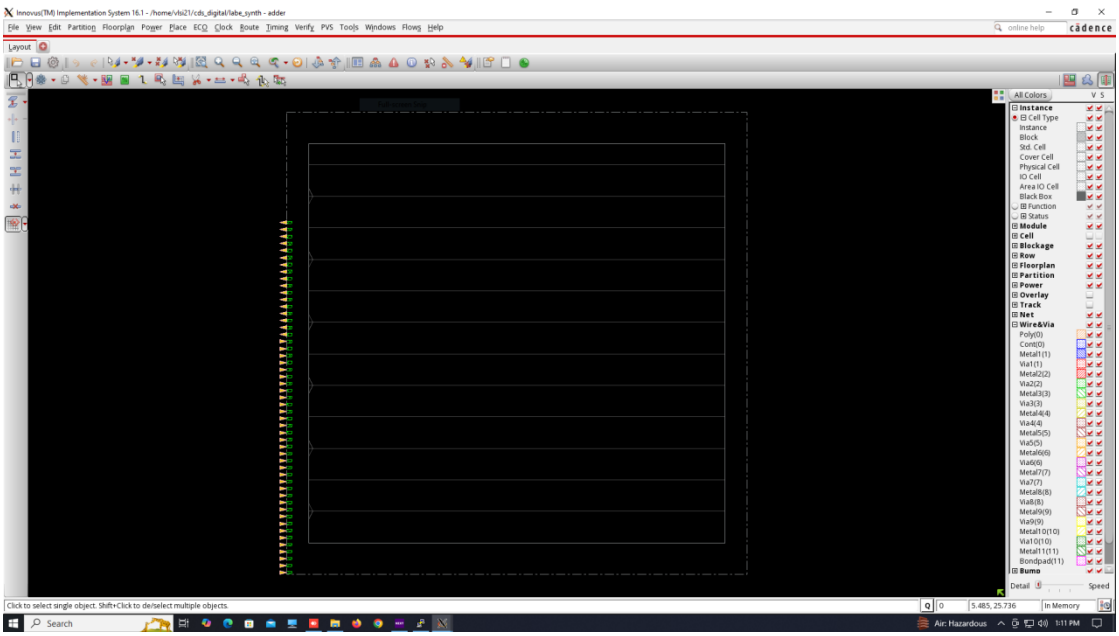

Figure 11: After Adding Pins

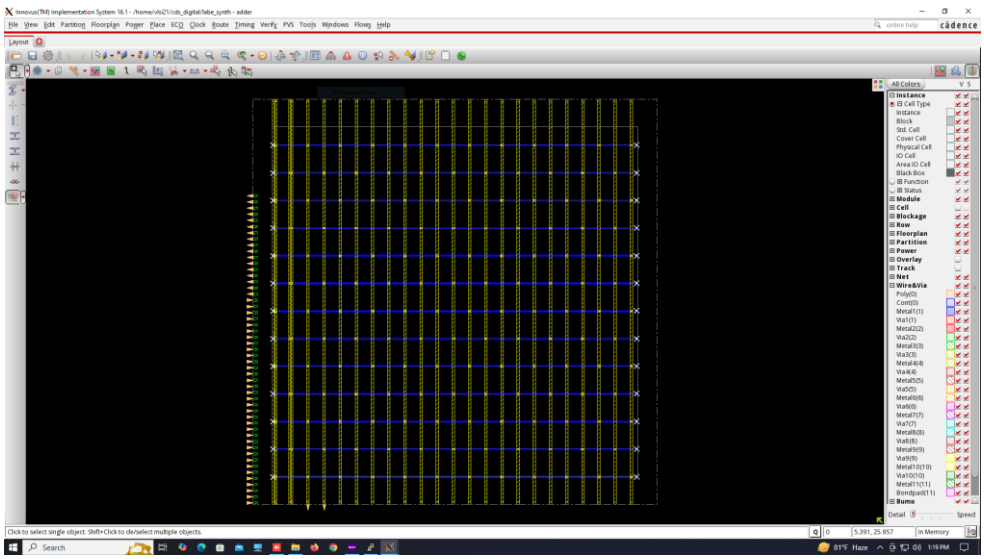

Figure 12: After Adding Power Rails

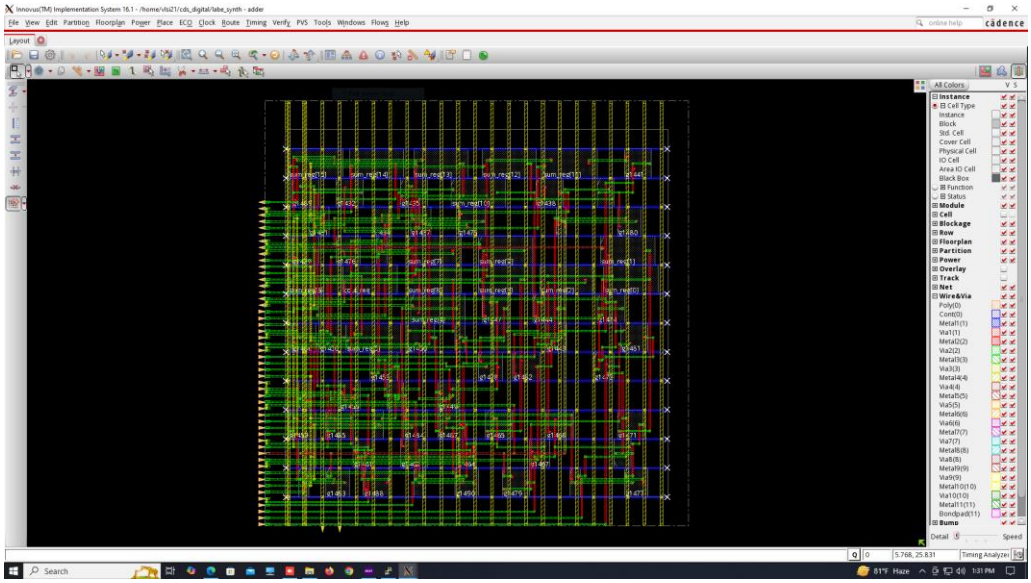

Figure 13: After Cell Placements

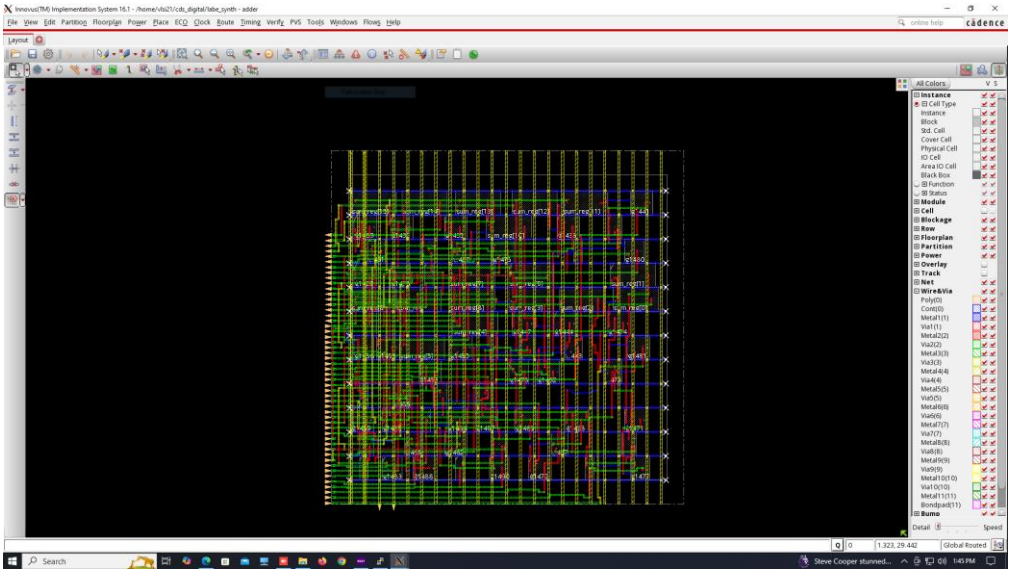

Figure 14: After Nano Routing

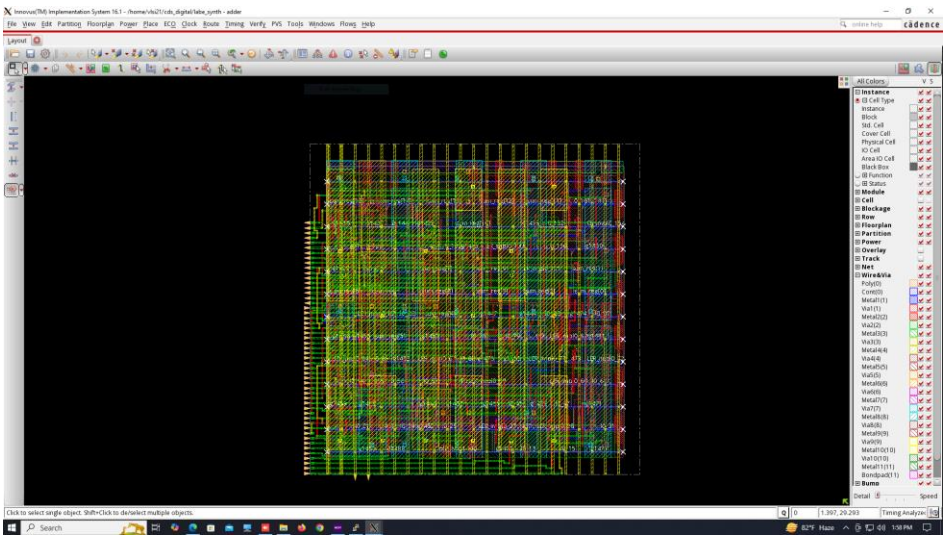

Figure 15: After Metal Fill

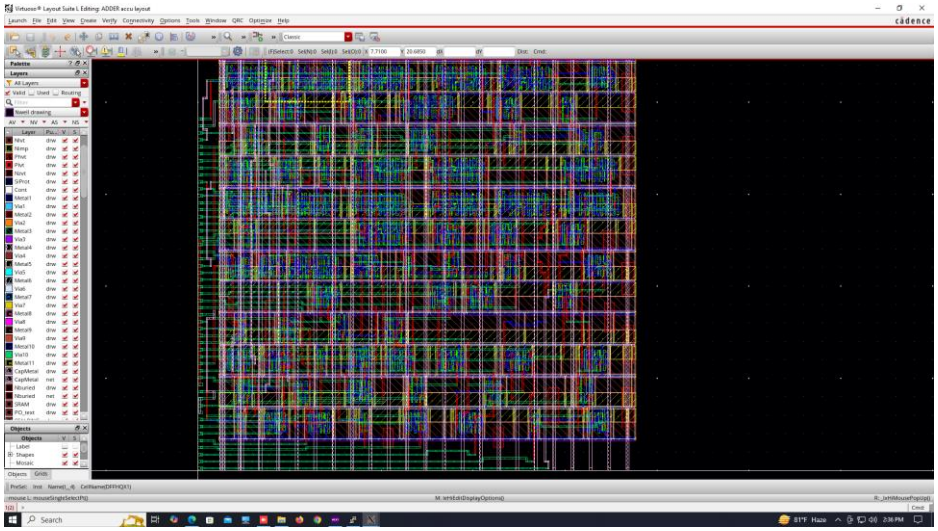

Figure 16: Final Layout

6.4 DRC Test

```
Total CPU Time                  : 1(s)
Total Real Time                 : 1(s)
Peak Memory Used                : 20(M)
Total Original Geometry         : 1626(12405)
Total DRC RuleChecks            : 562
Total DRC Results               : 0 (0)
Summary can be found in file accu.sum
ASCII report database is /home/vlsi21/accu.drc_errors.ascii
Checking in all SoftShare licenses.
```

Figure 17: DRC Test

7 Log Book of Project Implementation

Date	Milestone achieved	Individual Role	Team Role	Comments
25/10/2024	Design Code & Self-Checking Verification Code	1906110		Successful
01/11/2024	Layered Testbench Code	1906103		Successful
08/11/2024	Synthesis	1906092		Successful
18/11/2024	PnR with DRC Test	1906101		Successful
24/11/2024	Coverage Maximization	1906110		Successful
02/12/2024	Optimization of PPA		Whole Team	Successful

8 References

1) CMOS VLSI Design: A Circuits and Systems Perspective 4th Edition by Neil Weste & David Harris

2) https://www.sciencedirect.com/topics/computer-science/ripple-carry-adder