IC DESIGN LAB PROJECT

FIR FILTER IMPLEMENTATION

TOOL USED: CADENCE GENUS, NCLAUNCH, INNOVUS STYLUS, EDA PLAYGROUND (ICARUS VERILOG)

REPORTED BY:

R VRSHA SOLACHI

EC24M2013

SEMI CUSTOM DESIGN PROJECT
FIR Filter Design Implementation

AIM:

- Design and implementation of an 8-bit, 6-tap FIR filter using Verilog.
- Perform RTL simulation, synthesis, and physical implementation using industry-standard EDA tools.

TOOLS USED:

- Cadence Genus (Synthesis)
- Cadence NCLaunch (Simulation)
- Cadence Innovus (Place & Route)
- Cadence Stylus (Physical Verification)
- EDA Playground (Icarus Verilog for functional simulation)

PROCEDURE:

- **RTL Design & Verification (Functional Simulation)**
  - Write the Verilog code for a 6-tap FIR filter with coefficients {3, 4, 5, 4, 3, 3}.
  - Create a testbench in Verilog to provide input stimuli and observe the output.
  - Simulate the design using EDA Playground (Icarus Verilog) or NCLaunch (Cadence simulator).
  - Verify the waveform output and ensure the filter is functioning as expected.

- **Synthesis using Cadence Genus**
  - Load the Verilog source files into Cadence Genus.
  - Perform RTL synthesis targeting a standard cell library (GPDK 90nm).
  - Analyze reports for area, power, and timing.
  - Generate netlist and SDC files for backend flow.

- **Floorplanning, Placement, and Routing using Innovus**
  - Import synthesized netlist and constraints into Cadence Innovus.
  - Perform floorplanning, standard cell placement, and routing.
  - Run Design Rule Check (DRC) and Layout vs Schematic (LVS).
  - Optimize for timing closure and generate final GDSII.

- **Parasitic Extraction & Post-Layout Simulation**

  - Extract parasitics using QRC or integrated Stylus extraction flow.
  - Perform post-layout simulation using Spectre or NCLaunch.

- Compare pre-layout and post-layout simulation waveforms.

CIRCUIT DIAGRAM:

The FIR filter consists of a series of delay registers, multipliers (with constant coefficients), and an adder tree to sum the products.

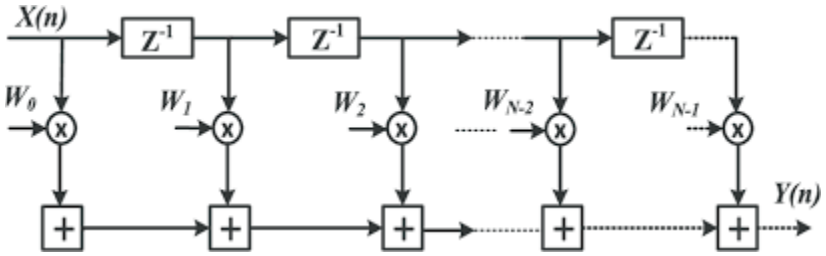


Figure 1: FIR Filter with fixed coefficients (direct form)

OPERATION:

- The FIR filter works on the principle of convolving a finite set of previous input values with fixed coefficients.
- The output at time n is given by:

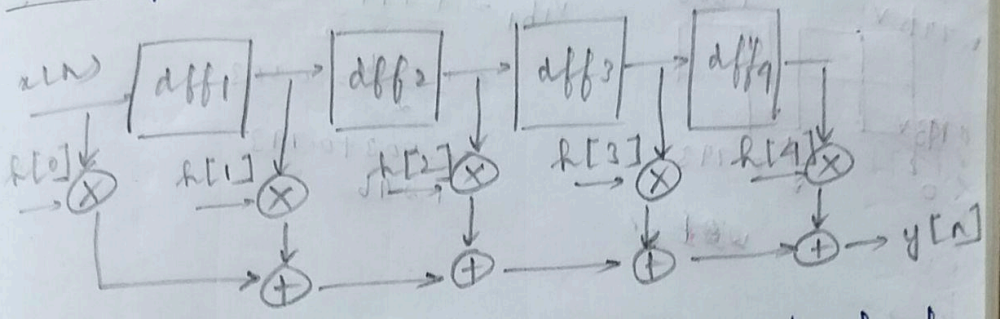
  y[n] = h0*x[n] + h1*x[n-1] + h2*x[n-2] + h3*x[n-3] + h4*x[n-4]

- Where:
  - x[n] is the input at time n
  - h0 to h5 are the FIR coefficients (here, 3, 4, 5, 4, 3, 3)
  - Input samples are delayed using D flip-flops, and the resulting values are multiplied by the respective coefficients using shift-and-add logic. The products are then summed up to give the output.

MANUAL CALCULATIONS:

Using the above formula for direct form 6 tap FIR filter :
We get the following caluclations

Direct form FIR filter using a tapped delay line structure



The structure implements the standard FIR filter equation

$$y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2] + h_3 x[n-3] + h_4 x[n-4]$$

n	x(n)	x(n-1)	x(n-2)	x(n-3)	x(n-4)	x(n-5)	y(n)
0	10	0	0	0	0	0	1X10 = 10
1	10	10	0	0	0	0	1X10+4X10 = 70
2	20	10	10	0	0	0	60+40+50 = 150
3	20	20	10	10	0	0	60+80+50+40 = 230
4	30	20	20	10	10	0	90+80+100+40+30 = 340
5	30	30	20	20	10	10	90+120+100+80+30+30 = 450
6	40	30	30	20	20	10	120+120+150+80+60+30 = 560
7	40	40	30	30	20	20	120+160+180+120+60+60 = 670
8	50	40	40	30	30	20	100+160+200+120+90+60 = 780
9	50	50	40	40	30	30	150+200+200+160+90+90 = 890
10	60	50	50	40	40	30	150+200+250+160+120+90 = 1000

Likewise, all the values can be computed.

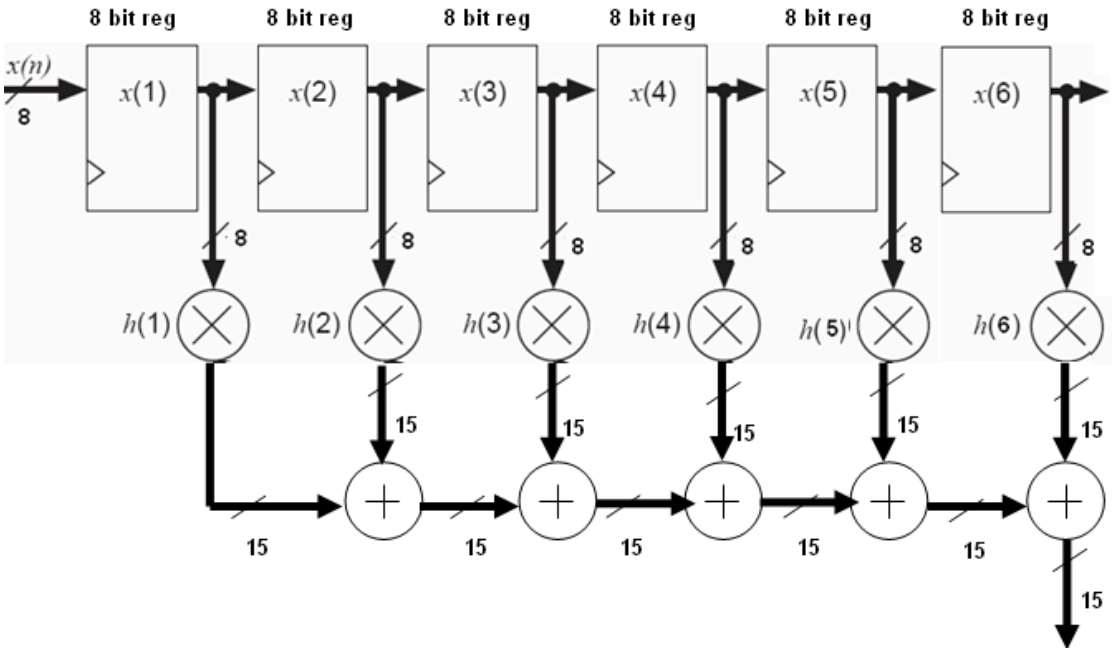


Figure 2: 6-tap FIR Filter with fixed coefficients (direct form)

The equation simplifies to

y[n] = h0·x[n] + h1·x[n-1] + h2·x[n-2] + h3·x[n-3] + h4·x[n-4] + h5·x[n-5]

CODE:

```
`timescale 1ns/1ps

module firfilter (
   input clk, rst,
   input [7:0] x,
   output [15:0] dataout
);
   wire [7:0] d1, d2, d3, d4, d5;  // Delay elements
   wire [15:0] m1, m2, m3, m4, m5, m6;  // Multiplication results

   // FIR filter coefficients
   parameter h0 = 3;
   parameter h1 = 4;
   parameter h2 = 5;
   parameter h3 = 4;
   parameter h4 = 3;
   parameter h5 = 3;
```

```
    // Instantiate D flip-flops for delay elements
    dff dff1(clk, rst, x, d1);
    dff dff2(clk, rst, d1, d2);
    dff dff3(clk, rst, d2, d3);
    dff dff4(clk, rst, d3, d4);
    dff dff5(clk, rst, d4, d5);

    // Multiply inputs with coefficients
    assign m1 = h0 * x;
    assign m2 = h1 * d1;
    assign m3 = h2 * d2;
    assign m4 = h3 * d3;
    assign m5 = h4 * d4;
    assign m6 = h5 * d5;

    // Sum of all terms (FIR filter output)
    assign dataout = m1 + m2 + m3 + m4 + m5 + m6;

endmodule

// D Flip-Flop Module (8-bit)
module dff (
    input clk, rst,
    input [7:0] d,
    output reg [7:0] q
);
    always @(posedge clk or posedge rst) begin
        if (rst)
            q <= 8'b0;
        else
            q <= d;
    end
endmodule
```

TESTBENCH:

```
`timescale 1ns/1ps

module fir_filter_tb;

    reg clk;
    reg rst;
```

```
reg [7:0] x;
wire [15:0] dataout;

firfilter uut (
    .clk(clk),
    .rst(rst),
    .x(x),
    .dataout(dataout)
);

// Tap wires (connected to internal delay registers)
wire [7:0] d1 = uut.d1;
wire [7:0] d2 = uut.d2;
wire [7:0] d3 = uut.d3;
wire [7:0] d4 = uut.d4;
wire [7:0] d5 = uut.d5;

initial begin
    $dumpfile("fir_filter.vcd");
    $dumpvars(0, fir_filter_tb);

    clk = 0;
    rst = 1;
    x = 0;

    #50 rst = 0;

    // Apply input samples every 50 ns
    #50 x = 10;
    #50 x = 10;
    #50 x = 20;
    #50 x = 20;
    #50 x = 30;
    #50 x = 30;
    #50 x = 40;
    #50 x = 40;
    #50 x = 50;
    #50 x = 50;
    #50 x = 60;
    #50 x = 60;
    #50 x = 70;
    #50 x = 70;
```

```
        #50 x = 80;
        #50 x = 80;
        #50 x = 90;
        #50 x = 90;
        #50 x = 100;
        #50 x = 100;

        #100 $finish;
    end

    always #25 clk = ~clk;

    // Print output every posedge of clk
    always @(posedge clk) begin
        $display("Time=%0t ns | x=%3d | d1=%3d | d2=%3d | d3=%3d | d4=%3d | d5=%3d |
dataout=%5d",
                 $time, x, d1, d2, d3, d4, d5 dataout);
    end

endmodule
```

WAVEFORM ANALYSIS:

- Observe clk, rst, x, and dataout signals in the waveform.
- Confirm that after initial delays, the dataout signal reflects the weighted sum of the last 5 input samples.
- Check internal signals like d1, d2, d3, and d4 to verify prgoram.

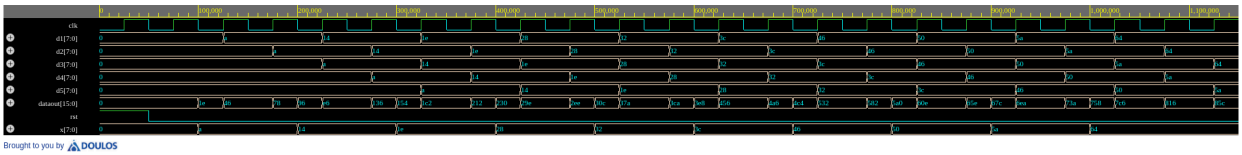


Figure 3: waveform of 6 tap FIR filter (Icarus Verilog)

- The output of the above code is shown below :

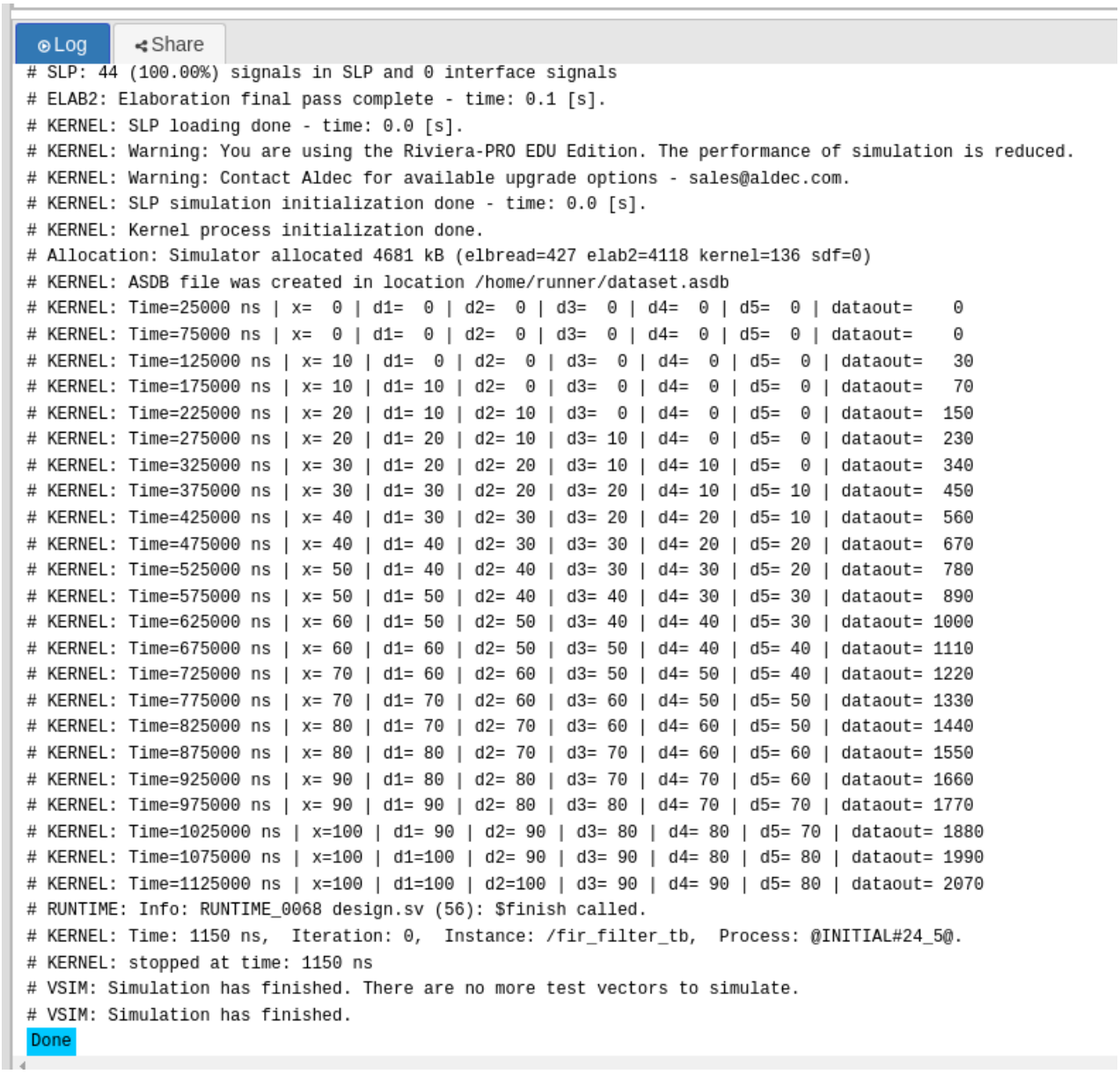


Figure 4: Console output of Icarus Verilog

EXPLANATION

- The shift register stores the last 5 input values. At each clock cycle:
- New input (x[n]) is stored in shift_reg[0].
- Old values shift right (shift_reg[i] = shift_reg[i-1]).
- The output (dataout) is computed using the equation above.

AREA REPORT

Applications Places Text Editor

Documents Open report_area.rpt

~/DIGITAL_EC24M2013/FIR_FILTER/synthesis/reports

report_area.rpt

```
1 ============================================================
2  Generated by:           Genus(TM) Synthesis Solution 20.11-s111_1
3  Generated on:           Apr 09 2025  07:57:29 pm
4  Module:                 firfilter
5  Operating conditions:   PVT_0P9V_125C (balanced_tree)
6  Wireload mode:          enclosed
7  Area mode:              timing library
8 ============================================================
9
10  Instance Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
11 ------------------------------------------------------------------------------
12 firfilter               154    703.152     0.000      703.152 <none> (D)
13   dff1    dff             9     49.932     0.000       49.932 <none> (D)
14   dff2    dff_32          9     49.932     0.000       49.932 <none> (D)
15   dff3    dff_31          9     49.932     0.000       49.932 <none> (D)
16   dff4    dff_30          9     49.932     0.000       49.932 <none> (D)
17   dff5    dff_29          9     49.932     0.000       49.932 <none> (D)
18   (D) = wireload is default in technology library
```

The total area of 703.152 units were covered

POWER REPORT

Applications Places Text Editor

Documents Open report_power.rpt

~/DIGITAL_EC24M2013/FIR_FILTER/synthesis/reports

report_power.rpt

```
1 Instance: /firfilter
2 Power Unit: W
3 PDB Frames: /stim#0/frame#0
4   ------------------------------------------------------------------------
5     Category         Leakage     Internal    Switching        Total    Row%
6   ------------------------------------------------------------------------
7       memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
8     register     5.38336e-09  3.37013e-07  3.83940e-08  3.80790e-07   3.97%
9        latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
10       logic     1.28157e-08  1.76188e-06  7.42917e-06  9.20386e-06  96.03%
11        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
12       clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
13         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
14          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
15  ------------------------------------------------------------------------
16    Subtotal     1.81990e-08  2.09889e-06  7.46756e-06  9.58465e-06 100.00%
17  Percentage           0.19%       21.90%       77.91%      100.00% 100.00%
18  ------------------------------------------------------------------------
```

The different categories of power are shown with memory, register, latch, bbox, clock, pad and pm were taken into consideration.

IMPLEMENTATION :

Synthesis Using GENUS Synthesis Solution Tool

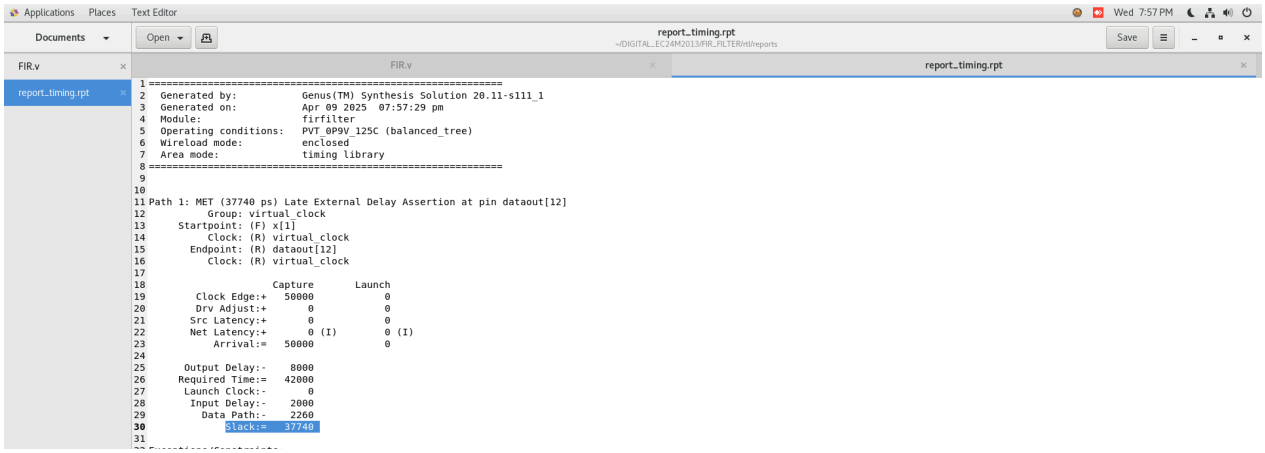


Figure 5: Timing report in GENUS synthesis

The 6-tap FIR filter design was synthesized using **Cadence Genus Synthesis Solution**. During the synthesis, **timing analysis** was performed, and the design achieved a **positive slack of 37,740 ps**, indicating that all timing requirements were comfortably met and the design is **timing clean**.

GUI

The gui_show command in Genus was used to **visualize timing paths**, critical nets, and to debug any potential delay issues graphically in the GUI environment.

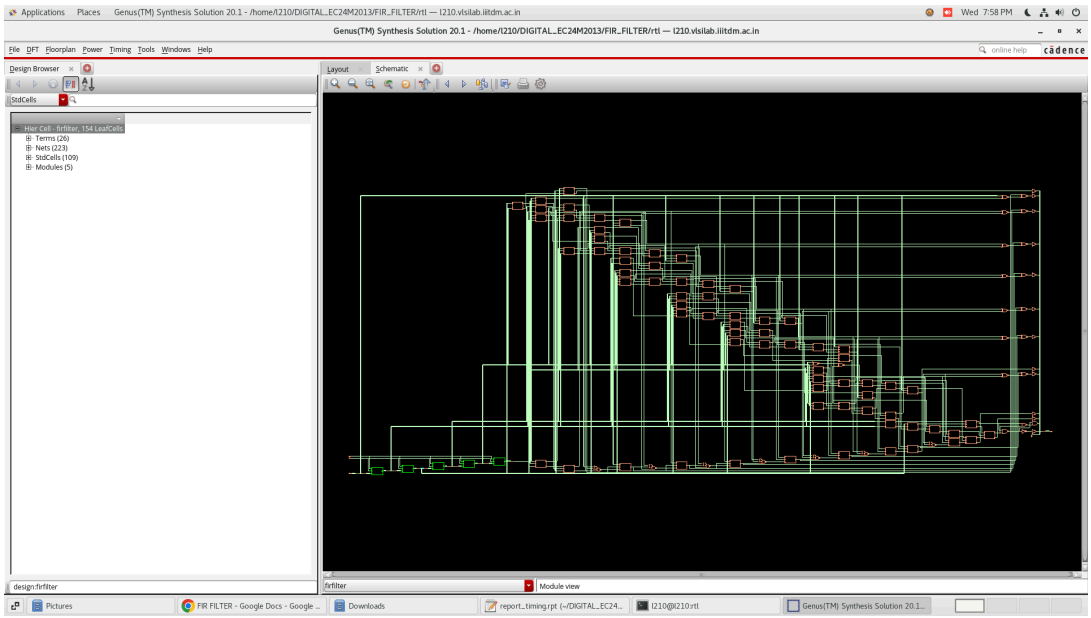


Figure 4: gui_show command output screen

NCLAUNCH

- In NCLaunch, the simulation environment allowed us to observe both the console output and the waveform results of the FIR filter.
- The waveform shows the behavior of input signal x, intermediate delayed outputs (d1 through d5), and the final filter output dataout.
- As the input values change over time, the FIR filter processes these samples through delay registers and multiply-accumulate stages.
- The output waveform of dataout illustrates the smoothing effect of the filter, showing how past and current inputs are combined based on the tap coefficients (h0 to h5).
- This confirms the functional correctness of the FIR implementation, matching expected filtered results for each clock cycle.

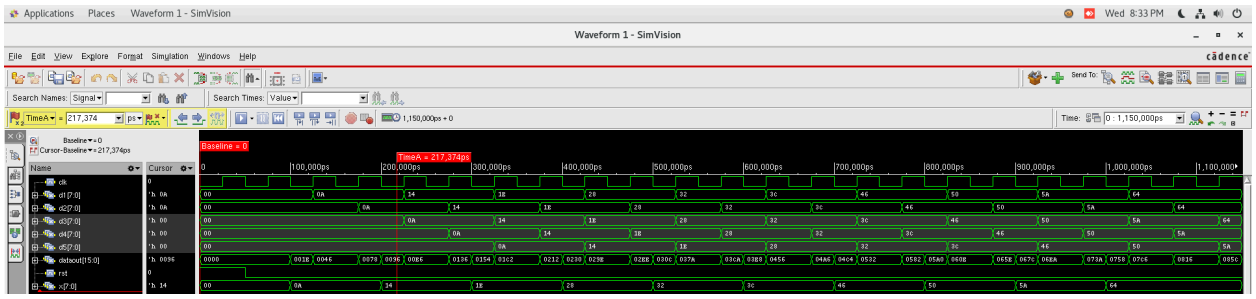
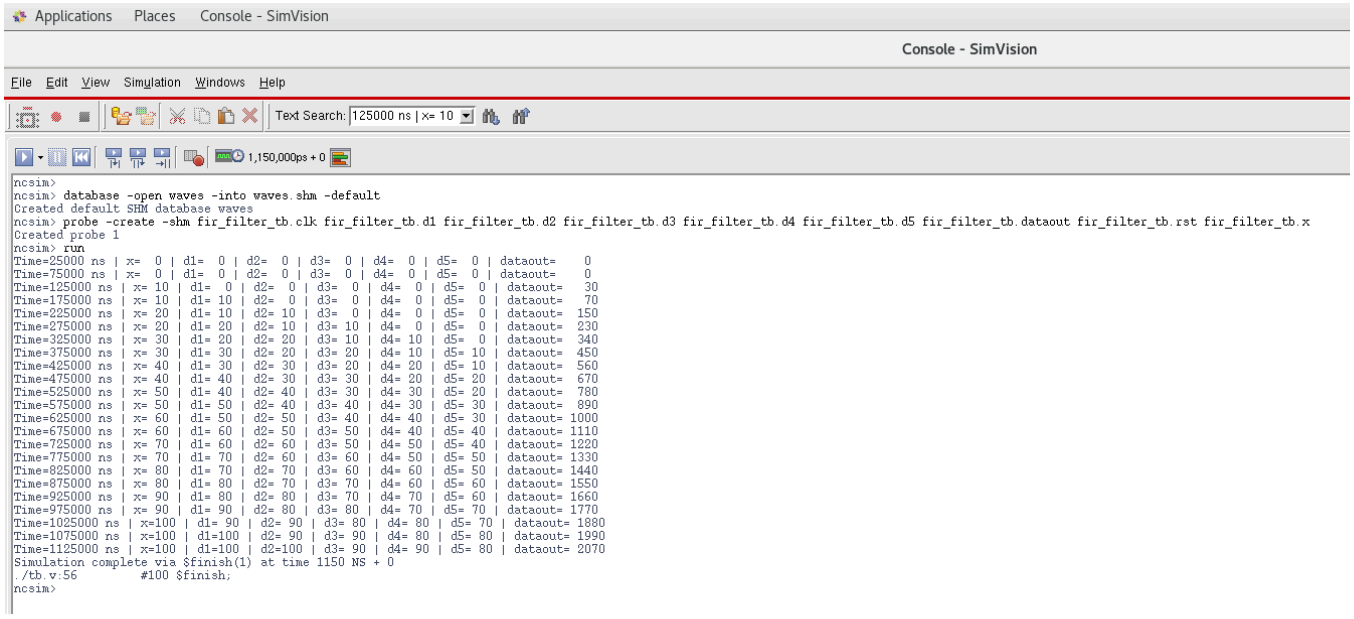


Figure 5: Nclaunch output waveform




```
ncsim>
ncsim> database -open waves -into waves.shm -default
Created default SHM database waves
ncsim> probe -create -shm fir_filter_tb.clk fir_filter_tb.d1 fir_filter_tb.d2 fir_filter_tb.d3 fir_filter_tb.d4 fir_filter_tb.d5 fir_filter_tb.dataout fir_filter_tb.rst fir_filter_tb.x
Created probe 1
ncsim> run
Time=25000 ns | x=  0 | d1=  0 | d2=  0 | d3=  0 | d4=  0 | d5=  0 | dataout=    0
Time=75000 ns | x=  0 | d1=  0 | d2=  0 | d3=  0 | d4=  0 | d5=  0 | dataout=    0
Time=125000 ns | x= 10 | d1=  0 | d2=  0 | d3=  0 | d4=  0 | d5=  0 | dataout=   30
Time=175000 ns | x= 10 | d1= 10 | d2=  0 | d3=  0 | d4=  0 | d5=  0 | dataout=   70
Time=225000 ns | x= 20 | d1= 10 | d2= 10 | d3=  0 | d4=  0 | d5=  0 | dataout=  150
Time=275000 ns | x= 20 | d1= 20 | d2= 10 | d3= 10 | d4=  0 | d5=  0 | dataout=  230
Time=325000 ns | x= 30 | d1= 20 | d2= 20 | d3= 10 | d4= 10 | d5=  0 | dataout=  340
Time=375000 ns | x= 30 | d1= 30 | d2= 20 | d3= 20 | d4= 10 | d5= 10 | dataout=  450
Time=425000 ns | x= 40 | d1= 30 | d2= 30 | d3= 20 | d4= 20 | d5= 10 | dataout=  560
Time=475000 ns | x= 40 | d1= 40 | d2= 30 | d3= 30 | d4= 20 | d5= 20 | dataout=  670
Time=525000 ns | x= 50 | d1= 40 | d2= 40 | d3= 30 | d4= 30 | d5= 20 | dataout=  780
Time=575000 ns | x= 50 | d1= 50 | d2= 40 | d3= 40 | d4= 30 | d5= 30 | dataout=  890
Time=625000 ns | x= 60 | d1= 50 | d2= 50 | d3= 40 | d4= 40 | d5= 30 | dataout= 1000
Time=675000 ns | x= 60 | d1= 60 | d2= 50 | d3= 50 | d4= 40 | d5= 40 | dataout= 1110
Time=725000 ns | x= 70 | d1= 60 | d2= 60 | d3= 50 | d4= 50 | d5= 40 | dataout= 1220
Time=775000 ns | x= 70 | d1= 70 | d2= 60 | d3= 60 | d4= 50 | d5= 50 | dataout= 1330
Time=825000 ns | x= 80 | d1= 70 | d2= 70 | d3= 60 | d4= 60 | d5= 50 | dataout= 1440
Time=875000 ns | x= 80 | d1= 80 | d2= 70 | d3= 70 | d4= 60 | d5= 60 | dataout= 1550
Time=925000 ns | x= 90 | d1= 80 | d2= 80 | d3= 70 | d4= 70 | d5= 60 | dataout= 1660
Time=975000 ns | x= 90 | d1= 90 | d2= 80 | d3= 80 | d4= 70 | d5= 70 | dataout= 1770
Time=1025000 ns | x=100 | d1= 90 | d2= 90 | d3= 80 | d4= 80 | d5= 70 | dataout= 1880
Time=1075000 ns | x=100 | d1=100 | d2= 90 | d3= 90 | d4= 80 | d5= 80 | dataout= 1990
Time=1125000 ns | x=100 | d1=100 | d2=100 | d3= 90 | d4= 90 | d5= 80 | dataout= 2070
Simulation complete via $finish(1) at time 1150 NS + 0
./tb.v:56         #100 $finish;
ncsim>
```

Figure 6: Nclaunch Console

INNOVUS

- **Power Planning**

The power panning was used to insert **power stripes** (VDD/VSS) across the core area. These ensure proper power distribution to standard cells and macros, minimizing **IR drop** and enabling robust rail integrity for the entire chip.

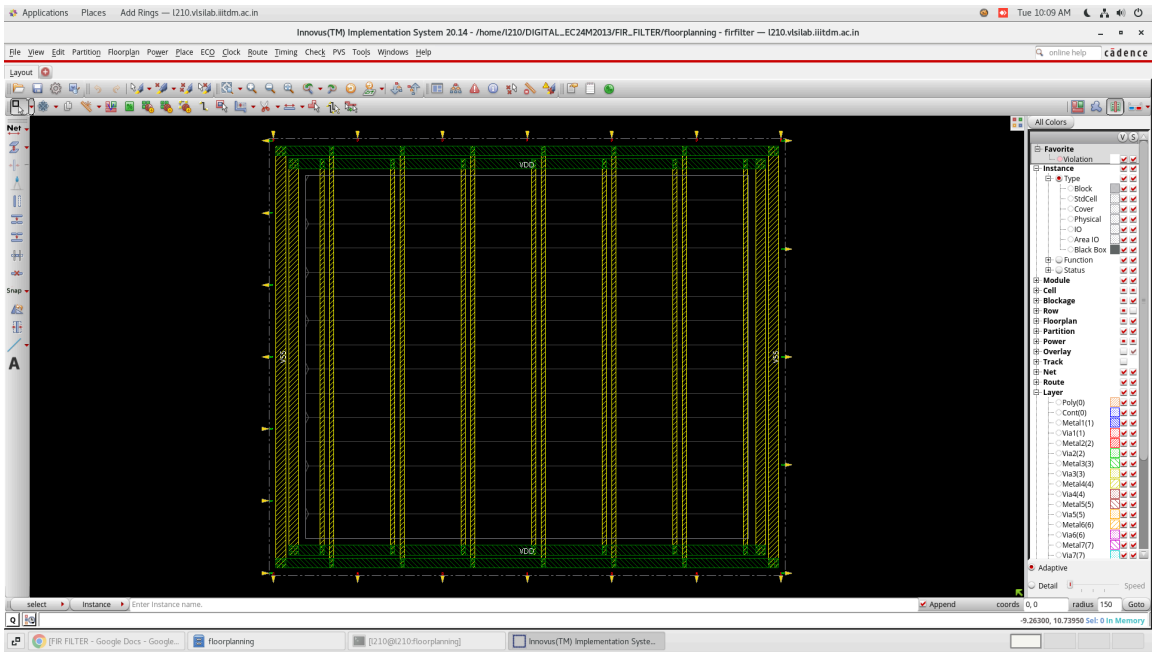


Figure 7: add stripes

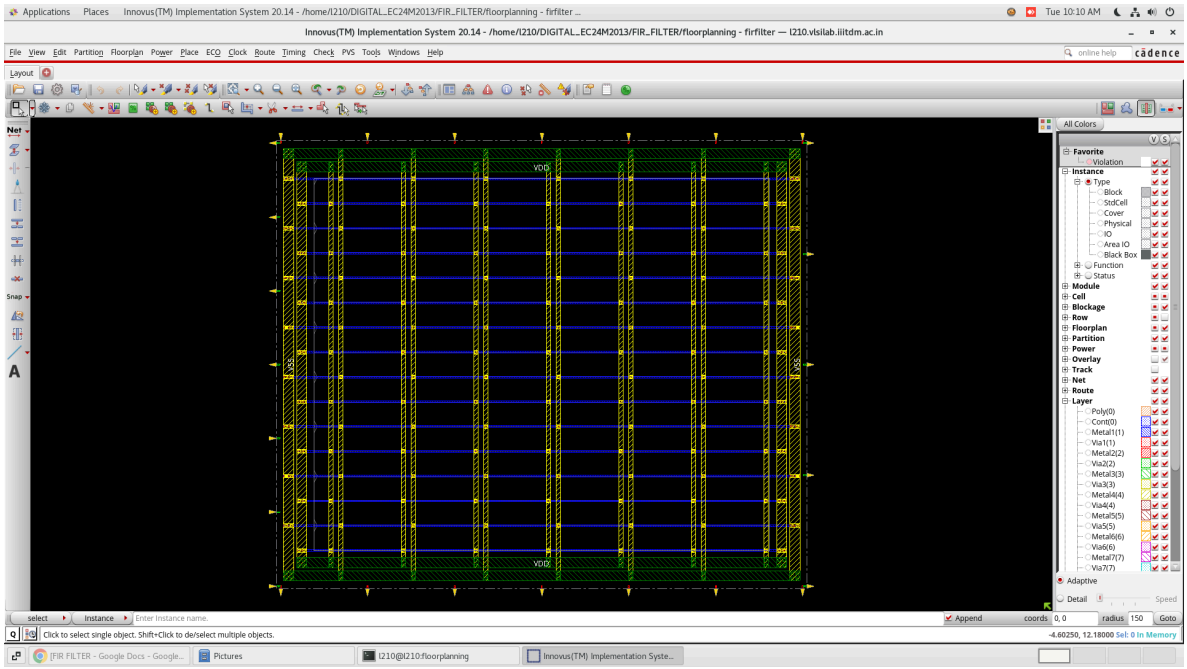


Figure 8: physical view on the followpin routes.

- **opt design summary – Placement Optimization Summary**

This summary reports metrics after placement optimization:

- **Total negative slack (TNS)** and **worst negative slack (WNS)** (both of them are positive).
- **Violating paths** are nil.
- **Congestion** levels are checked for routability.

```
------------------------------------------------------------------------
     opt_design Final Summary
------------------------------------------------------------------------

Setup views included:
 wc

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 37.469  |   N/A   | 37.469  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|   13    |   N/A   |   13    |
+--------------------+---------+---------+---------+
```

Figure 9: opt design final summary

- **Running place_opt_design**

The place_opt_design command performs **standard cell placement** with **timing-driven optimization**. It ensures optimal cell arrangement to meet setup and hold timing constraints.

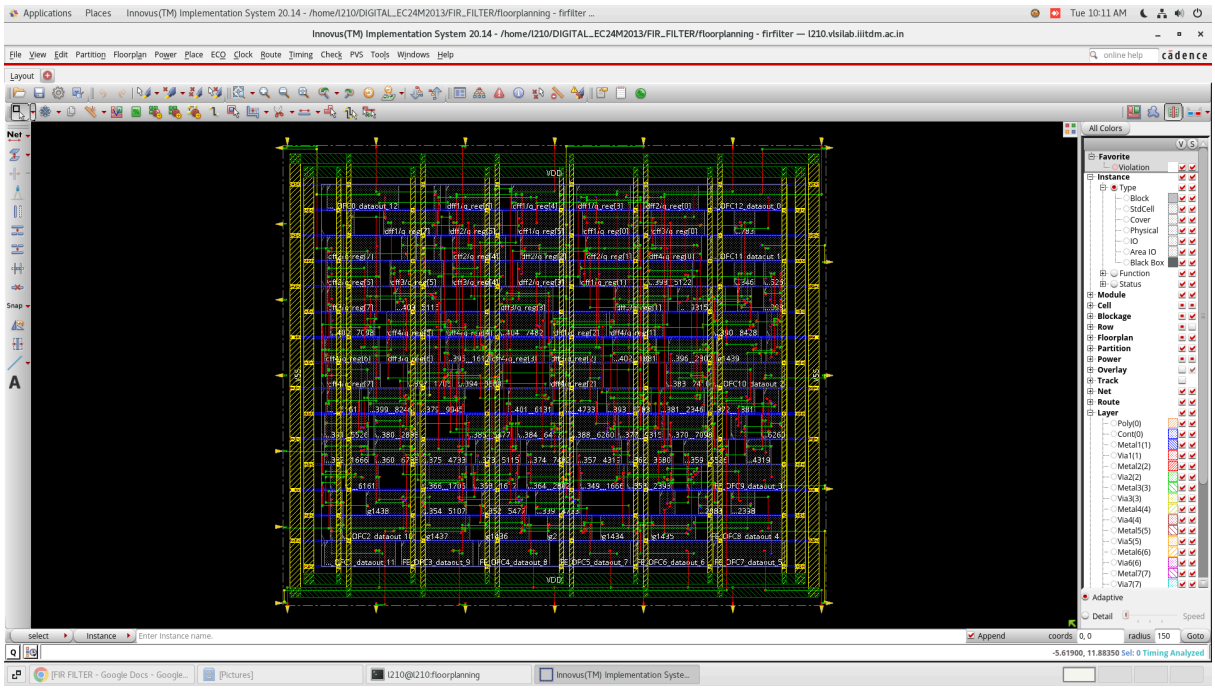


Figure 10: place_opt_design

- **Routing Rule Table**

The routing rule table defines:

- Preferred routing directions for each metal layer
- Width and spacing constraints.
  Via rules and DRC-safe margins.

This ensures DRC-compliant and efficient signal routing.

Figure 11: No DRC violations and the connectivity is checked.

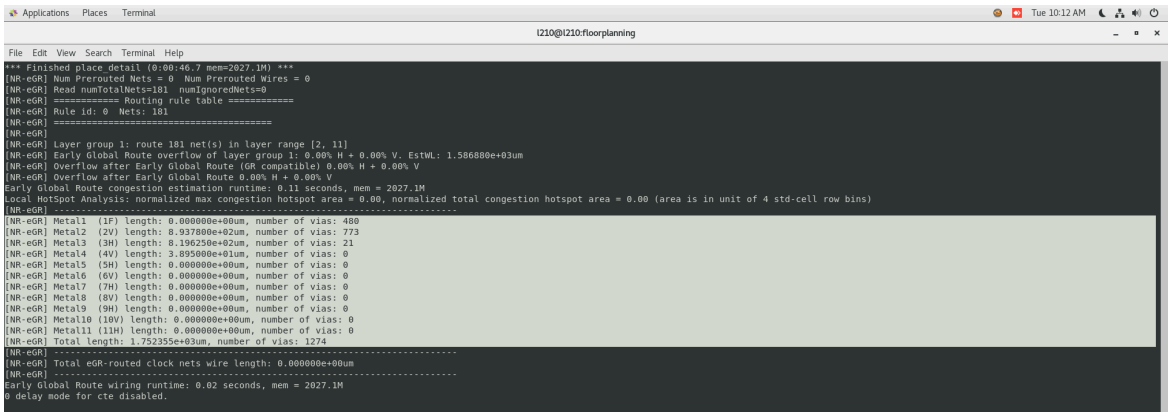


Figure 12: Routing rule table.

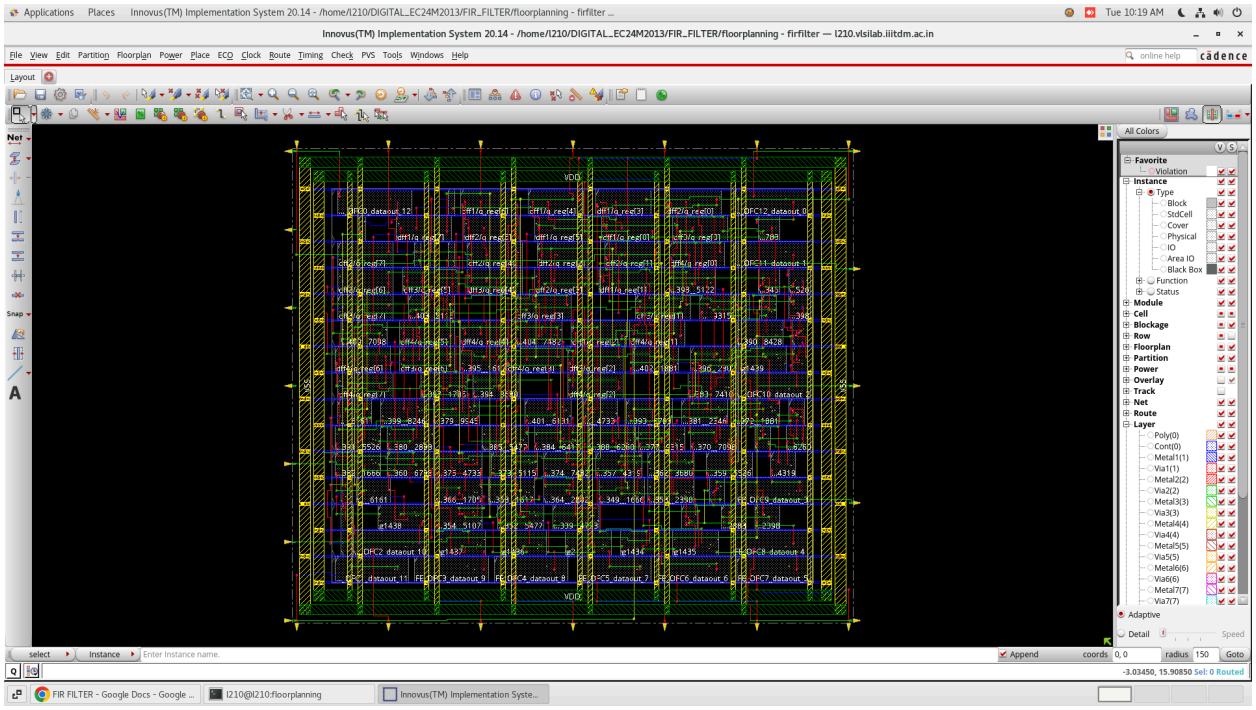


Figure 13: placement optimization

- **Final Timing Setup & Hold Summary**

After **clock tree synthesis (CTS)** and **global+detail routing**, final timing analysis is done:

- **Setup Summary**: Verifies that data arrives before the clock edge (no setup violations).
- **Hold Summary**: Confirms data is held stable after the clock edge
- Positive slack in both confirms **timing closure**.

```
------------------------------------------------------------
        time_design Summary
------------------------------------------------------------

Setup views included:
 wc

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 37.476  |   N/A   | 37.476  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|   13    |   N/A   |   13    |
+--------------------+---------+---------+---------+
```

Figure 14: time_design summary (setup)

```
---------------------------------------------------------
        time_design Summary
---------------------------------------------------------

Hold views included:
 bc

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 10.107  |   N/A   | 10.107  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|   13    |   N/A   |   13    |
+--------------------+---------+---------+---------+

Density: 74.650%
---------------------------------------------------------
```

Figure 15: time_design summary (hold)

- **Rail Analysis**

Rail analysis ensures the **power delivery network (PDN)** can handle voltage drops and current demands:

- Checks **IR drop** and **EM (Electro Migration)**.
- Validates power integrity across the chip.

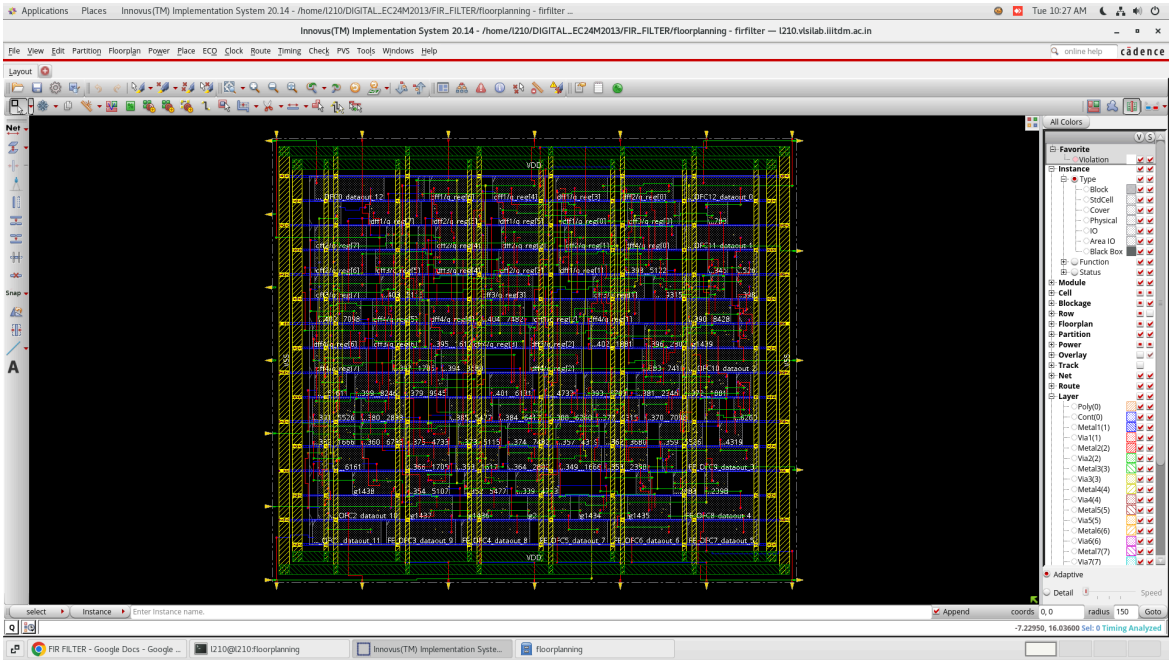


Figure 16: Rail Analysis

- **Power Analysis**

Reports **dynamic**, **leakage**, and **total power**:

- Useful to optimize for **low power**.
- Helps in checking peak power consumption for supply design.

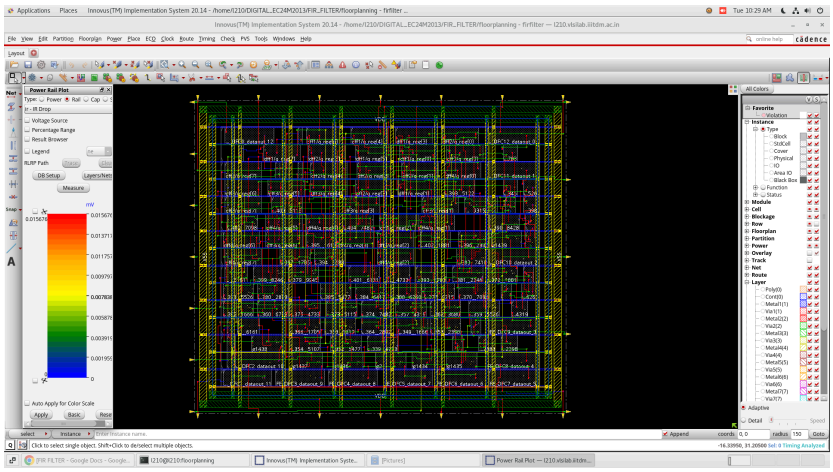


Figure 17: Power Analysis

- **Filler Insertion**

Filler cells are inserted between standard cells to:

- Maintain **well continuity**.
- Avoid DRC issues.
- Ensure proper connectivity of power rails.

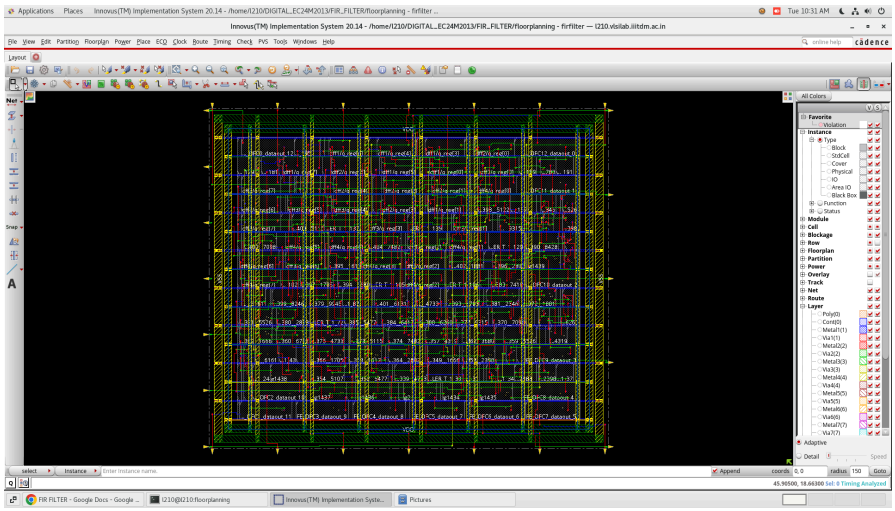


Figure 18: Filler Cells

INFERENCE

- In Genus Synthesis a positive slack of 37740 ps, indicating that the design meets all timing constraints was achieved .
- The synthesized netlist was structurally optimized for both area and performance.
- At NCLaunch Simulation The waveform output showed the correct filtering behavior.
- Each tap and delay element (d1 to d5) operated as expected, and the dataout reflected the weighted sum of current and delayed inputs, proving the FIR functionality.
- Power stripes were added to ensure even power distribution.
- Placement and optimization completed successfully with minimal congestion.
- Routing rule table guided clean routing with no DRC violations.
- Final timing analysis showed no setup or hold violations, confirming timing closure.
- Rail analysis and power analysis verified robust power integrity.
- Filler cells were added to maintain well connections and avoid gaps.

RESULT:

The FIR filter design met all functional and timing specifications from RTL to GDSII. The toolchain from Genus → NCLaunch → Innovus demonstrated a full ASIC flow implementation with waveform output, timing, area and power reports

REFERENCE

1. Design and Implementation of 6-Tap FIR Filter Using MAC for Low Power Applications (International Journal for Research in Applied Science & Engineering Technology (IJRASET)ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue VI June 2022- Available at www.ijraset.com)