

## **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.

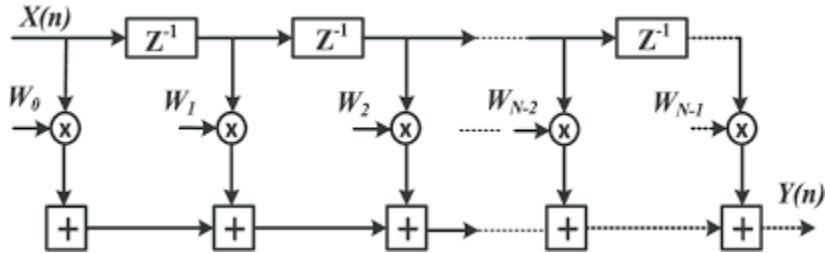


Figure 1: FIR Filter with fixed coefficients (direct form)

## **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:

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

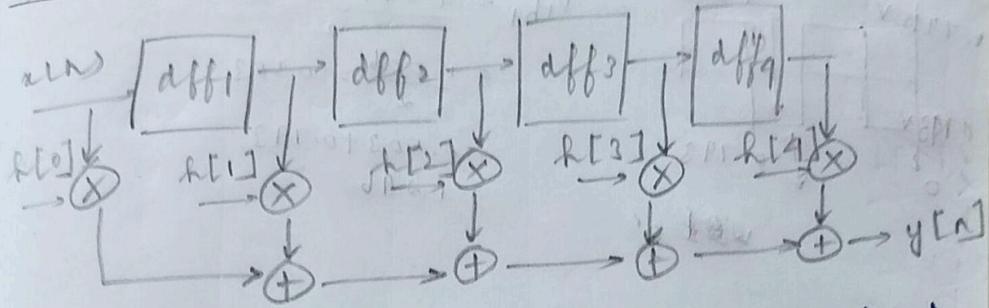
- Where:
    - $x[n]$  is the input at time n
    - $h_0$  to  $h_5$  are the FIR coefficients (here, 3, 4, 5, 4, 3, 3)
    - Input samples are delayed using D flip-flops, and the resulting values are multiplied by the respective coefficients using shift-and-add logic. The products are then summed up to give the output.

## **MANUAL CALCULATIONS:**

Using the above formula for direct form 6 tap FIR filter :

We get the following calculations

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	0	0	0	0	0	0	0
1	10	10	0	0	0	0	$3 \times 10 = 30$
2	20	10	10	0	0	0	$3 \times 10 + 4 \times 10 = 70$
3	20	20	10	10	0	0	$60 + 40 + 50 = 150$
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	$120 + 160 + 150 + 120 + 60 + 60 = 670$
8	50	40	40	30	30	20	$140 + 160 + 200 + 120 + 90 + 60 = 780$
9	50	50	40	40	30	30	$180 + 200 + 200 + 160 + 90 + 90 = 890$
10	60	50	50	40	40	30	$180 + 200 + 200 + 250 + 160 + 120 + 90 = 1000$

Likewise, all the values can be computed.

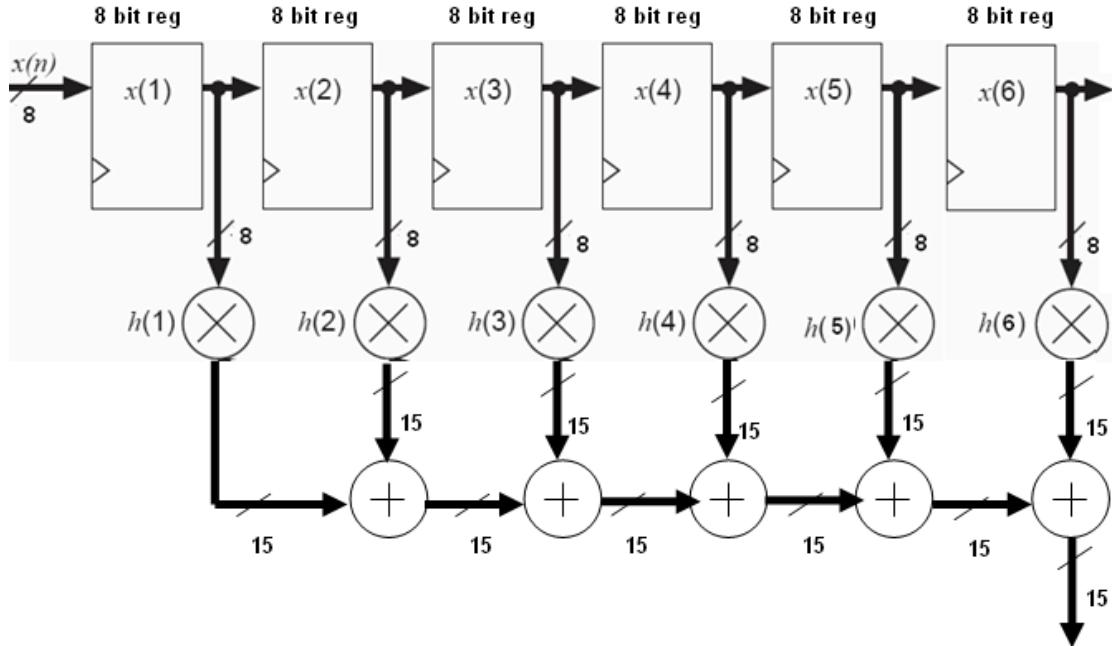


Figure 2: 6-tap FIR Filter with fixed coefficients (direct form)

The equation simplifies to

$$y[n] = h_0 \cdot x[n] + h_1 \cdot x[n-1] + h_2 \cdot x[n-2] + h_3 \cdot x[n-3] + h_4 \cdot x[n-4] + h_5 \cdot x[n-5]$$

#### 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 program.

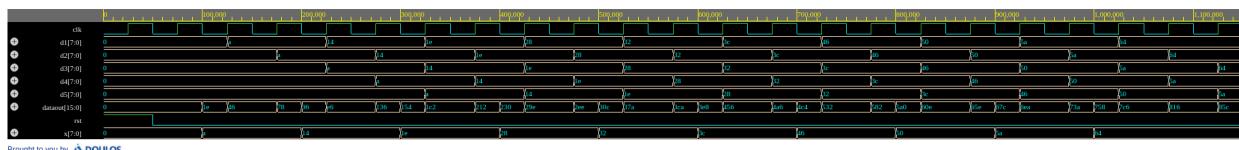


Figure 3: waveform of 6 tap FIR filter (Icarus Verilog)

- The output of the above code is shown below :

Log Share

```

# SLP: 44 (100.00%) signals in SLP and 0 interface signals
# ELAB2: Elaboration final pass complete - time: 0.1 [s].
# KERNEL: SLP loading done - time: 0.0 [s].
# KERNEL: Warning: You are using the Riviera-PRO EDU Edition. The performance of simulation is reduced.
# KERNEL: Warning: Contact Aldec for available upgrade options - sales@aldec.com.
# KERNEL: SLP simulation initialization done - time: 0.0 [s].
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 4681 kB (elbread=427 elab2=4118 kernel=136 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: Time=25000 ns | x= 0 | d1= 0 | d2= 0 | d3= 0 | d4= 0 | d5= 0 | dataout= 0
# KERNEL: Time=75000 ns | x= 0 | d1= 0 | d2= 0 | d3= 0 | d4= 0 | d5= 0 | dataout= 0
# KERNEL: Time=125000 ns | x= 10 | d1= 0 | d2= 0 | d3= 0 | d4= 0 | d5= 0 | dataout= 30
# KERNEL: Time=175000 ns | x= 10 | d1= 10 | d2= 0 | d3= 0 | d4= 0 | d5= 0 | dataout= 70
# KERNEL: Time=225000 ns | x= 20 | d1= 10 | d2= 10 | d3= 0 | d4= 0 | d5= 0 | dataout= 150
# KERNEL: Time=275000 ns | x= 20 | d1= 20 | d2= 10 | d3= 10 | d4= 0 | d5= 0 | dataout= 230
# KERNEL: Time=325000 ns | x= 30 | d1= 20 | d2= 20 | d3= 10 | d4= 10 | d5= 0 | dataout= 340
# KERNEL: Time=375000 ns | x= 30 | d1= 30 | d2= 20 | d3= 20 | d4= 10 | d5= 10 | dataout= 450
# KERNEL: Time=425000 ns | x= 40 | d1= 30 | d2= 30 | d3= 20 | d4= 20 | d5= 10 | dataout= 560
# KERNEL: Time=475000 ns | x= 40 | d1= 40 | d2= 30 | d3= 30 | d4= 20 | d5= 20 | dataout= 670
# KERNEL: Time=525000 ns | x= 50 | d1= 40 | d2= 40 | d3= 30 | d4= 30 | d5= 20 | dataout= 780
# KERNEL: Time=575000 ns | x= 50 | d1= 50 | d2= 40 | d3= 40 | d4= 30 | d5= 30 | dataout= 890
# KERNEL: Time=625000 ns | x= 60 | d1= 50 | d2= 50 | d3= 40 | d4= 40 | d5= 30 | dataout= 1000
# KERNEL: Time=675000 ns | x= 60 | d1= 60 | d2= 50 | d3= 50 | d4= 40 | d5= 40 | dataout= 1110
# KERNEL: Time=725000 ns | x= 70 | d1= 60 | d2= 60 | d3= 50 | d4= 50 | d5= 40 | dataout= 1220
# KERNEL: Time=775000 ns | x= 70 | d1= 70 | d2= 60 | d3= 60 | d4= 50 | d5= 50 | dataout= 1330
# KERNEL: Time=825000 ns | x= 80 | d1= 70 | d2= 70 | d3= 60 | d4= 60 | d5= 50 | dataout= 1440
# KERNEL: Time=875000 ns | x= 80 | d1= 80 | d2= 70 | d3= 70 | d4= 60 | d5= 60 | dataout= 1550
# KERNEL: Time=925000 ns | x= 90 | d1= 80 | d2= 80 | d3= 70 | d4= 70 | d5= 60 | dataout= 1660
# KERNEL: Time=975000 ns | x= 90 | d1= 90 | d2= 80 | d3= 80 | d4= 70 | d5= 70 | dataout= 1770
# KERNEL: Time=1025000 ns | x=100 | d1= 90 | d2= 90 | d3= 80 | d4= 80 | d5= 70 | dataout= 1880
# KERNEL: Time=1075000 ns | x=100 | d1=100 | d2= 90 | d3= 90 | d4= 80 | d5= 80 | dataout= 1990
# KERNEL: Time=1125000 ns | x=100 | d1=100 | d2=100 | d3= 90 | d4= 90 | d5= 80 | dataout= 2070
# RUNTIME: Info: RUNTIME_0068 design.sv (56): $finish called.
# KERNEL: Time: 1150 ns, Iteration: 0, Instance: /fir_filter_tb, Process: @INITIAL#24_5@.
# KERNEL: stopped at time: 1150 ns
# VSIM: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.

```

Done

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  $\text{shift\_reg}[0]$ .
- Old values shift right ( $\text{shift\_reg}[i] = \text{shift\_reg}[i-1]$ ).
- The output (dataout) is computed using the equation above.

## AREA REPORT

The screenshot shows a terminal window titled "report\_area.rpt" with the following content:

```
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

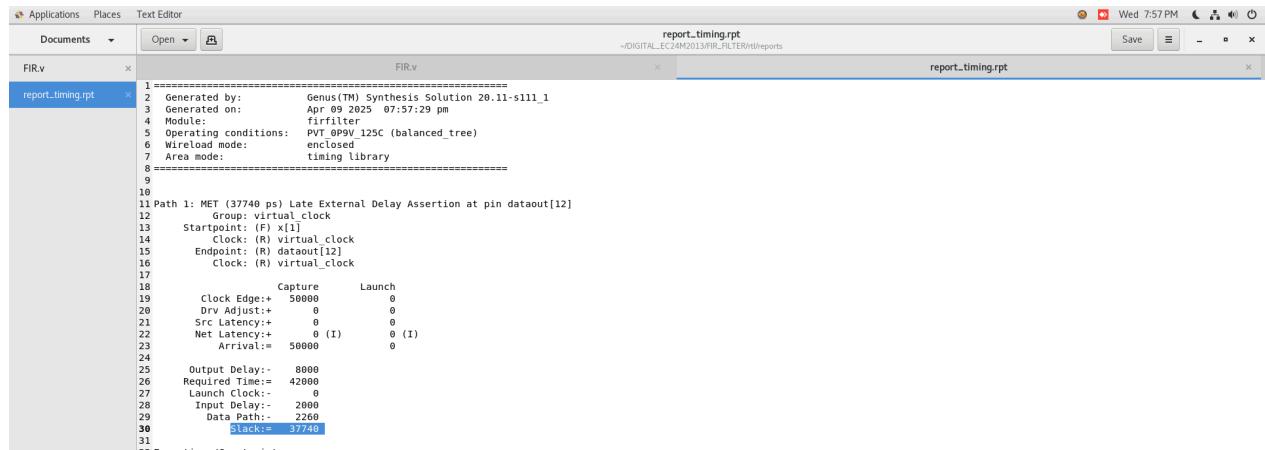
The screenshot shows a terminal window titled "report\_power.rpt" with the following content:

```
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



The screenshot shows the Cadence Genus Synthesis Solution Tool interface. The window title is "report\_timing.rpt". The content displays a timing report for an FIR filter design. The report includes header information, a summary of operating conditions, and a detailed analysis of a specific timing path. The path is labeled "Path 1: MET (37740 ps) Late External Delay Assertion at pin dataout[12]". The report provides capture and launch details for each stage of the path, including clock edges, driver adjustments, and arrival times. It also lists output delays, required times, launch clocks, input delays, and data paths. A final entry shows a positive slack of 37740 ps.

```
1 ====== Genus(TM) Synthesis Solution 20.11-s111_1
2 Generated by:          Genus(TM) Synthesis Solution 20.11-s111_1
3 Generated on:        April 10, 2025 07:57:29 pm
4 Mode:                firfilter
5 Operating conditions: PVT 0.99V 125C (balanced_tree)
6 Wireload mode:       enclosed
7 Area mode:           timing library
8 ======
9
10
11 Path 1: MET (37740 ps) Late External Delay Assertion at pin dataout[12]
12   Group: virtual_clock
13     Startpoint: (F) x[1]
14       Clock: (R) virtual_clock
15     Endpoint: (R) dataout[12]
16       Clock: (R) virtual_clock
17
18             Capture      Launch
19             Clock Edge:+ 50000    0
20             Drv Adjust:+ 0        0
21             Src Latency:+ 0        0
22             Net Latency:+ 0 (I)   0 (I)
23             Arrival:= 50000    0
24
25             Output Delay:- 8000
26             Required Time:= 42000
27             Launch Clock:- 0
28             Input Delay:- 2000
29             Data Path:- 2260
30             Slack:= 37740
31
```

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.

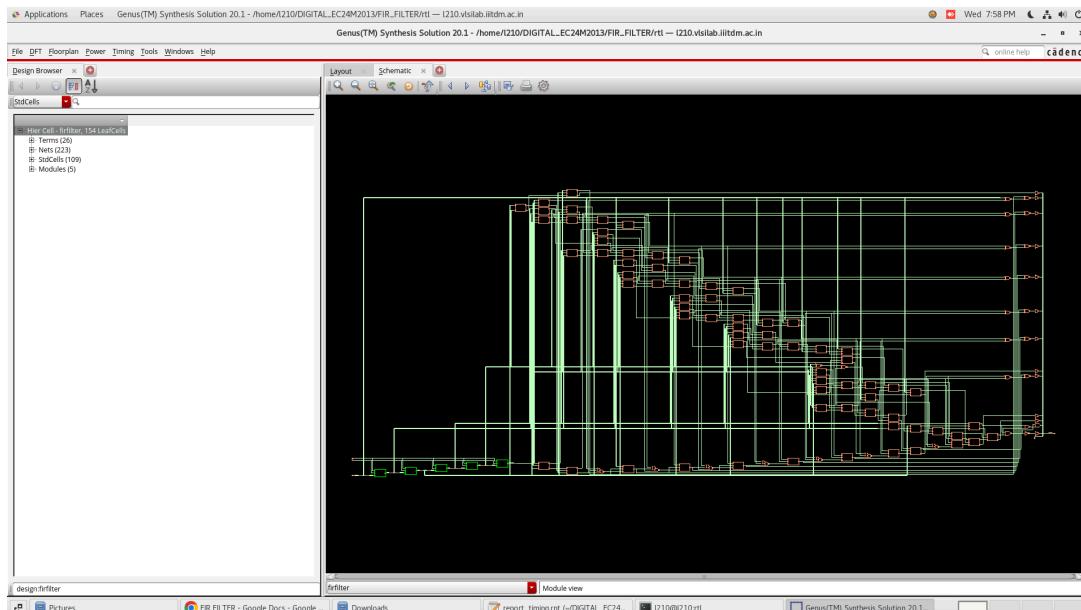


Figure 4: gui\_show command output screen

## 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.

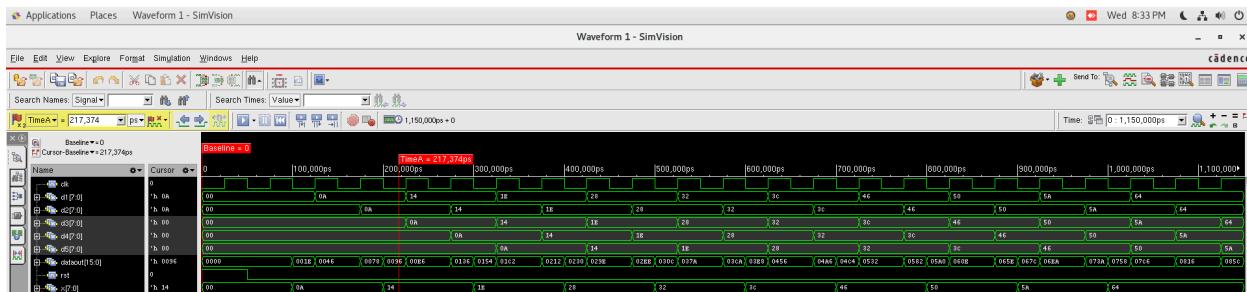


Figure 5: Nclaunch output waveform

```
Applications Places Console - SimVision
Console - SimVision
File Edit View Simulation Windows Help
Text Search: 125000 ns | x= 10 | 
1,150,000ps + 0

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= 0
Time=175000 ns | x= 10 | d1= 10 | d2= 0 | d3= 0 | d4= 0 | d5= 0 | dataout= 30
Time=225000 ns | x= 20 | d1= 10 | d2= 5 | d3= 0 | d4= 0 | d5= 0 | dataout= 70
Time=275000 ns | x= 20 | d1= 20 | d2= 10 | d3= 0 | d4= 0 | d5= 0 | dataout= 150
Time=325000 ns | x= 20 | d1= 20 | d2= 10 | d3= 10 | d4= 0 | d5= 0 | dataout= 230
Time=375000 ns | x= 30 | d1= 20 | d2= 20 | d3= 10 | d4= 10 | d5= 0 | dataout= 340
Time=425000 ns | x= 30 | d1= 30 | d2= 20 | d3= 20 | d4= 10 | d5= 10 | dataout= 450
Time=475000 ns | x= 40 | d1= 30 | d2= 30 | d3= 20 | d4= 20 | d5= 10 | dataout= 560
Time=525000 ns | x= 50 | d1= 40 | d2= 40 | d3= 30 | d4= 30 | d5= 20 | dataout= 670
Time=575000 ns | x= 50 | d1= 50 | d2= 40 | d3= 40 | d4= 30 | d5= 30 | dataout= 780
Time=625000 ns | x= 60 | d1= 50 | d2= 50 | d3= 40 | d4= 40 | d5= 30 | dataout= 890
Time=675000 ns | x= 60 | d1= 60 | d2= 50 | d3= 50 | d4= 40 | d5= 40 | dataout= 1000
Time=725000 ns | x= 70 | d1= 60 | d2= 60 | d3= 50 | d4= 50 | d5= 40 | dataout= 1110
Time=775000 ns | x= 70 | d1= 70 | d2= 60 | d3= 60 | d4= 50 | d5= 50 | dataout= 1220
Time=825000 ns | x= 80 | d1= 70 | d2= 70 | d3= 60 | d4= 60 | d5= 50 | dataout= 1330
Time=875000 ns | x= 80 | d1= 80 | d2= 70 | d3= 70 | d4= 60 | d5= 60 | dataout= 1440
Time=925000 ns | x= 80 | d1= 80 | d2= 80 | d3= 70 | d4= 70 | d5= 60 | dataout= 1550
Time=975000 ns | x= 90 | d1= 90 | d2= 80 | d3= 80 | d4= 70 | d5= 70 | dataout= 1660
Time=1025000 ns | x=100 | d1= 90 | d2= 90 | d3= 80 | d4= 80 | d5= 70 | dataout= 1770
Time=1075000 ns | x=100 | d1=100 | d2= 90 | d3= 90 | d4= 80 | d5= 80 | dataout= 1880
Time=1125000 ns | x=100 | d1=100 | d2=100 | d3= 90 | d4= 90 | d5= 80 | dataout= 1990
Time=1150000 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.

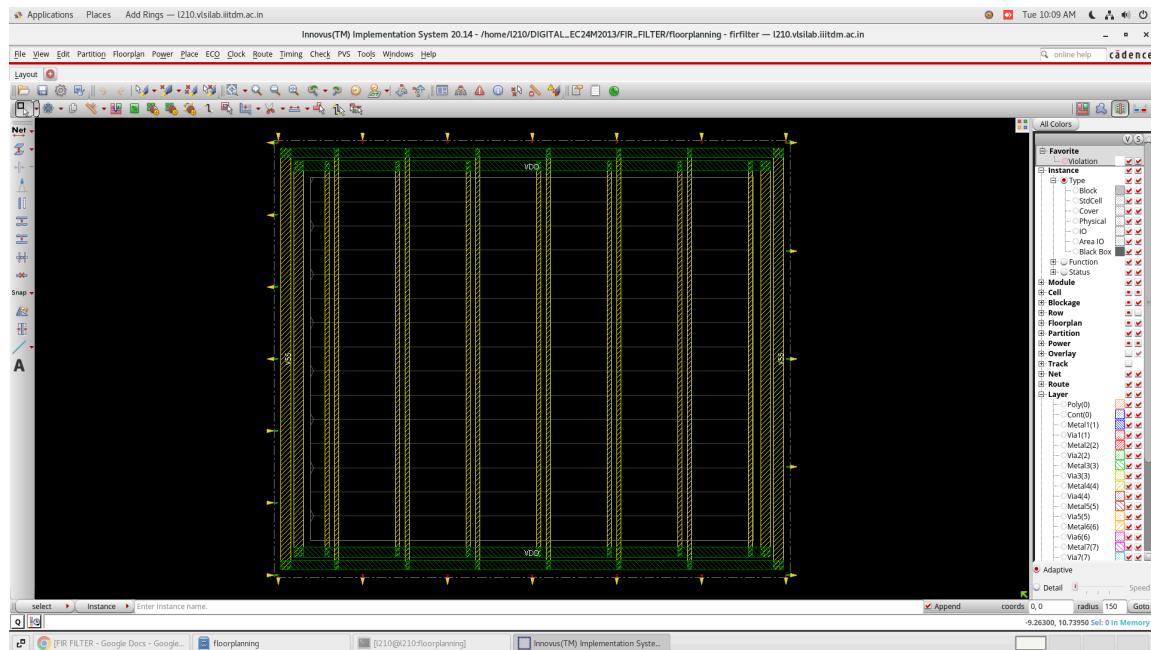


Figure 7: add stripes

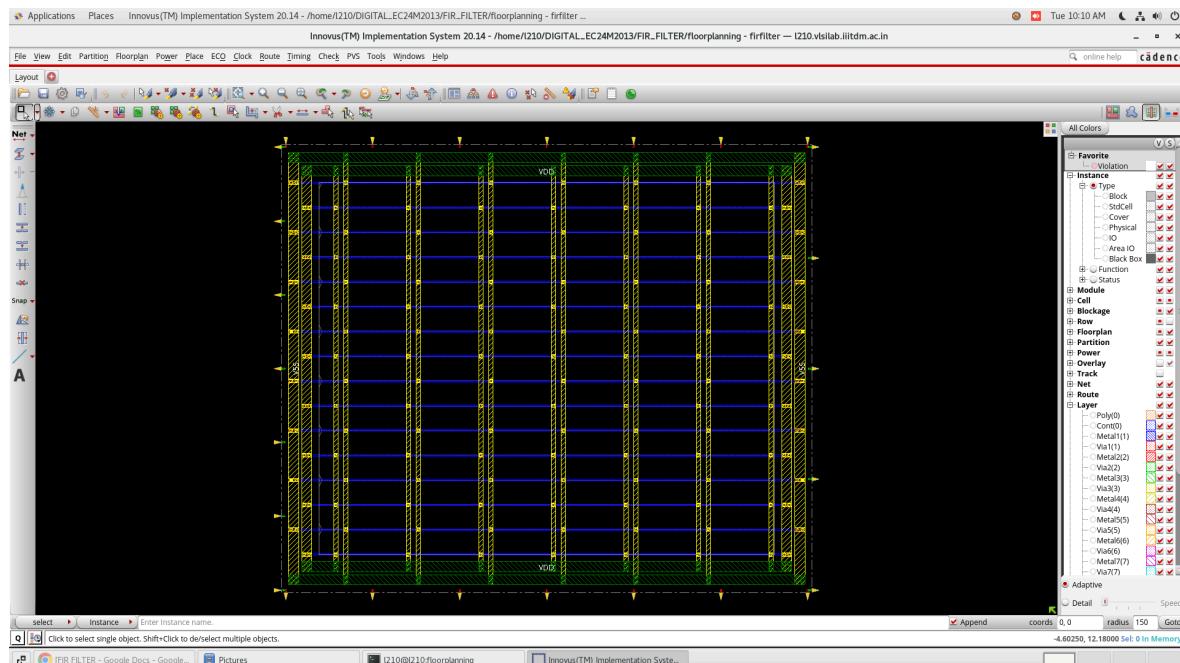


Figure 8: physical view on the followpin routes.

- **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.

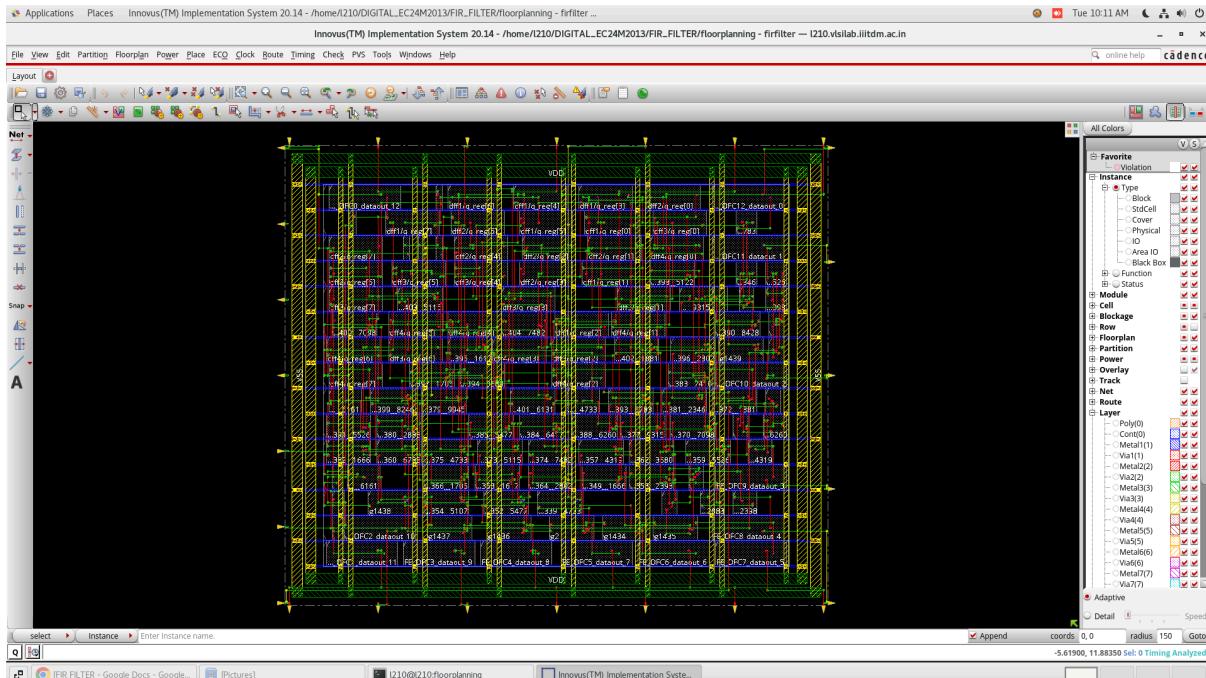


Figure 10: place\_opt\_design

- **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.

```

@innovus 6b> set db check drc.disable_rules {} ; set db check drc.ndr spacing auto ; set db check drc.check only default ; set db check_drc_inside_via def false ; set db check_drc_exclude_pg_net false ; set db check drc_limit 1000
@innovus 6b> check_drc
#-check_drc ignore_trial_route false ; set_db check_drc_ignore_cell_blockage false ; set_db check_drc_use_min_spacing_on_block_obs auto ; set_db check_drc_report_firfilter.drc.rpt ; set_db check_drc_limit 1000
#-report firfilter.drc.rpt           # enums={true false auto}, default=auto, user setting
*** Starting Verify DRC (MEM: 2181.5) ***
VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Checking Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area: {0.000 0.000 36.600 30.970} 1 of 1
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:00.0 ELAPSED TIME: 0.00 MEM: 8.0M) **

@innovus 6b> set db check drc.area {0 0 0}
@innovus 6b> check_connectivity -type all -error 1000 -warning 50
VERIFY_CONNECTIVITY uses new engine.

***** Start: VERIFY_CONNECTIVITY *****
Start Time: Tue Apr  8 10:22:57 2025

Design Name: firfilter
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (36.6000, 30.9700)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Tue Apr  8 10:22:57 2025
Time Elapsed: 0:00:00.0

***** End: VERIFY_CONNECTIVITY *****
Verification Complete : 0 Viols, 0 Wrngs.
(CPU Time: 0:00:00.0 MEM: 0.000M)

@innovus 7b> 

```

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

```

Applications Places Terminal Help
l210@l210:floortracing
File Edit View Search Terminal Help
Tue 10:12 AM
*** Finished place detail (0:00:46.7 mem=2027.1M) ***
[NR-eGR] Num Prerouted Nets = 0  Num Prerouted Wires = 0
[NR-eGR] Read numTotalNets=181 numIgnoredNets=0
[NR-eGR] ===== Routing rule table =====
[NR-eGR] Rule id: 0 Nets: 181
[NR-eGR] =====
[NR-eGR] Layer group 1: route 181 net(s) in layer range [2, 11]
[NR-eGR] Early Global Route congestion estimation runtime: 0.11 seconds, mem = 2027.1M
[NR-eGR] Early Global Route overflow after Early Global Route (0.00% H + 0.00% V, EstWL: 1.58688e+03um)
[NR-eGR] Early Global Route overflow after Early Global Route (0.00% H + 0.00% V)
[NR-eGR] Early Global Route congestion estimation runtime: 0.11 seconds, mem = 2027.1M
LocalHotspot Analysis: normalized max congestion hotspot area = 0.00, normalized total congestion hotspot area = 0.00 (area is in unit of 4 std-cell row bins)
[NR-eGR] =====
[NR-eGR] Metal1 (1F) length: 0.00000e+00um, number of vias: 480
[NR-eGR] Metal2 (2V) length: 8.93780e+02um, number of vias: 773
[NR-eGR] Metal3 (3H) length: 8.19025e+02um, number of vias: 21
[NR-eGR] Metal4 (4V) length: 3.00500e+02um, number of vias: 0
[NR-eGR] Metal5 (5H) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal6 (6V) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal7 (7H) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal8 (8V) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal9 (9H) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal10 (10V) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Metal11 (11H) length: 0.00000e+00um, number of vias: 0
[NR-eGR] Total length: 1.752355e+03um, number of vias: 1274
[NR-eGR] =====
[NR-eGR] Total eGR-routed clock nets wire length: 0.00000e+00um
[NR-eGR] =====
Early Global Route wiring runtime: 0.02 seconds, mem = 2027.1M
0 delay mode for cte disabled.

```

Figure 12: Routing rule table.

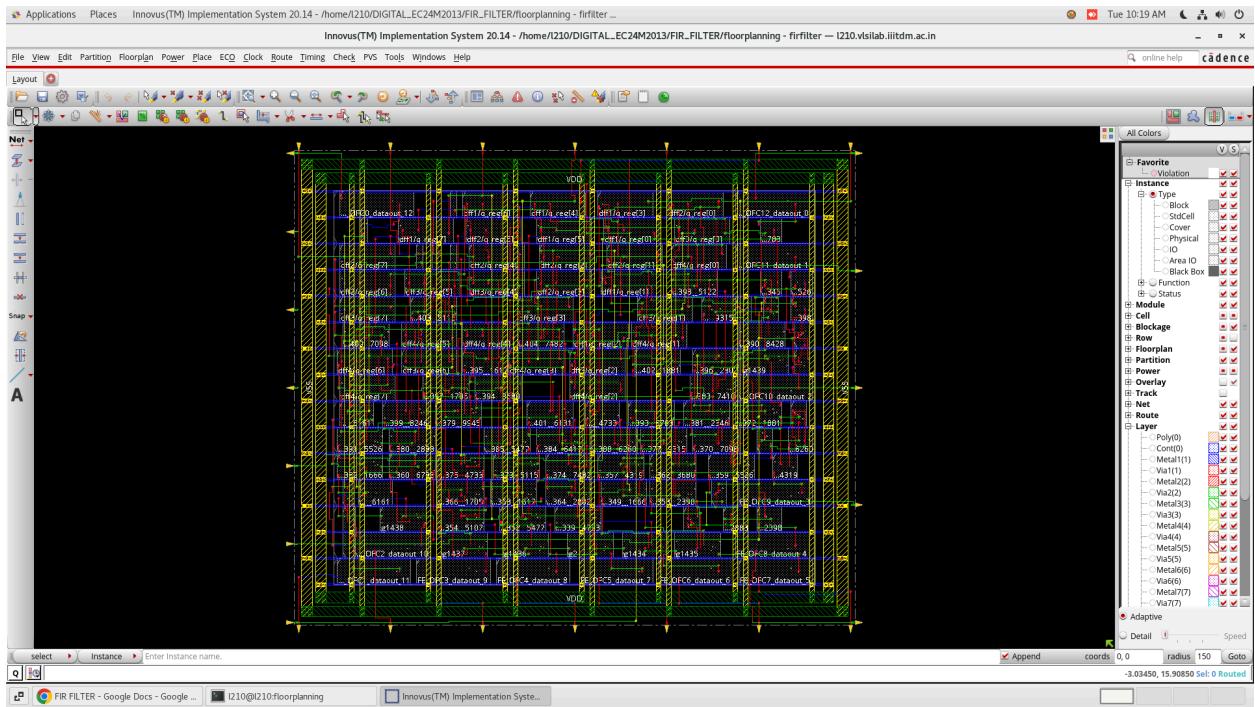


Figure 13: placement optimization

- **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	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.

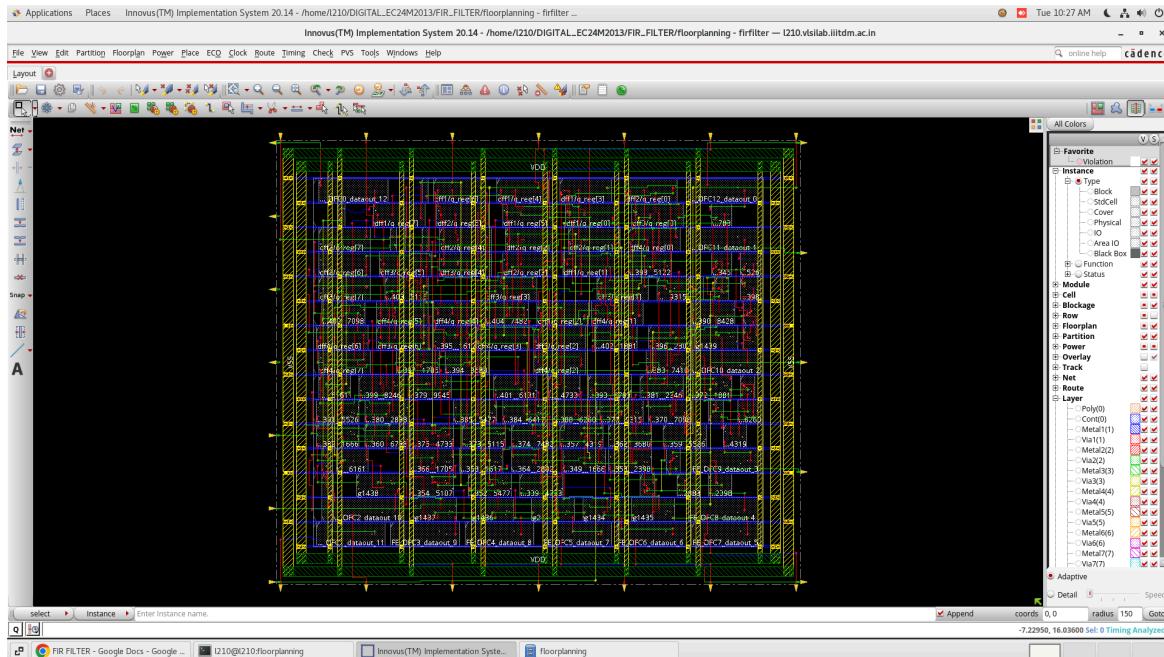


Figure 16: Rail Analysis

- **Power Analysis**

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

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

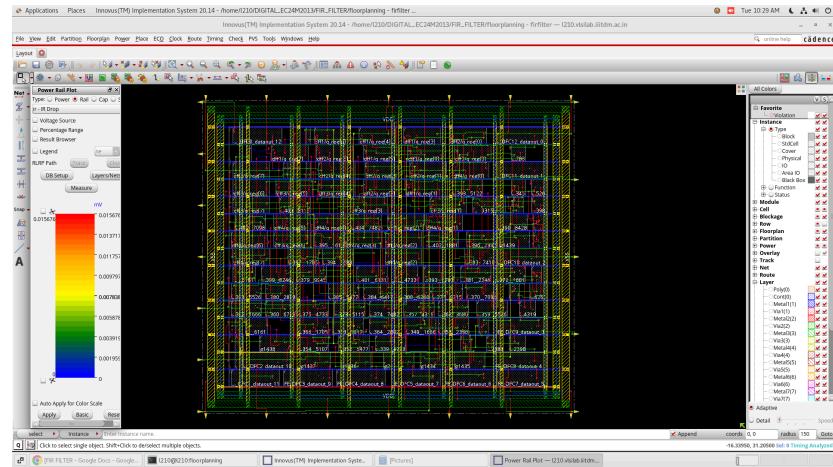


Figure 17: Power Analysis

- **Filler Insertion**

Filler cells are inserted between standard cells to:

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

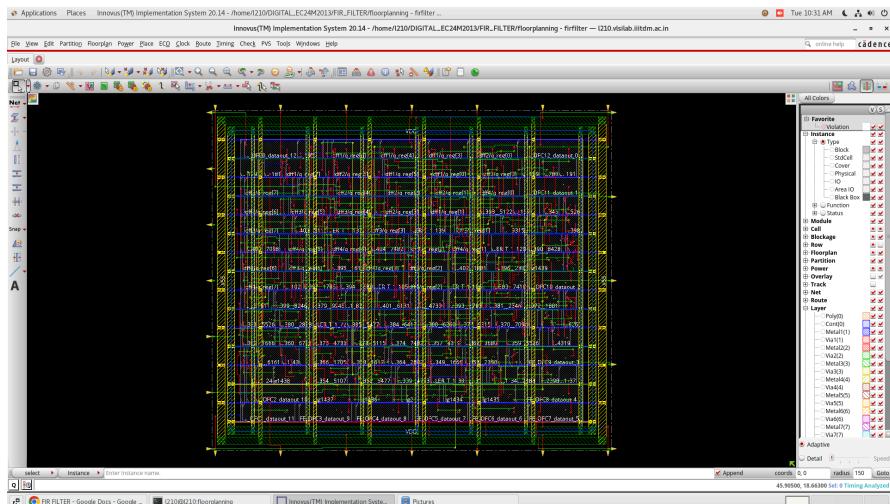


Figure 18: Filler Cells

## **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](http://www.ijraset.com))