ASIC Project Report

Shivam Gupta, 2024PVL0120
Ankit Verma, 2024PVL0119
Asgar Ali, 2024PVL094
Satvik Jain, 2022 M.Tech VLSI 2024-2026, Electrical Department
Indian Institute of Technology, Jammu

Abstract

The RTL-to-GDSII flow is the fundamental process for translating high-level hardware descriptions into physical layouts for ASIC fabrication. This report outlines the key stages, including RTL design and verification, synthesis, placement and routing, and crucial physical verification steps like DRC. It emphasizes the transformation from abstract logic to a manufacturable GDSII file, highlighting the essential methodologies for realizing integrated circuits.

I Fast Fourier Transform

I-A RTL Design

Register Transfer Level (RTL) is a design abstraction used in digital circuit design that describes the flow of data between registers and the logical operations performed on that data. It forms the foundation for hardware description languages like Verilog and VHDL. RTL design enables precise control over hardware behavior and is essential for synthesizing circuits into gate-level implementations, making it a crucial step in the development of digital systems and ASICs.

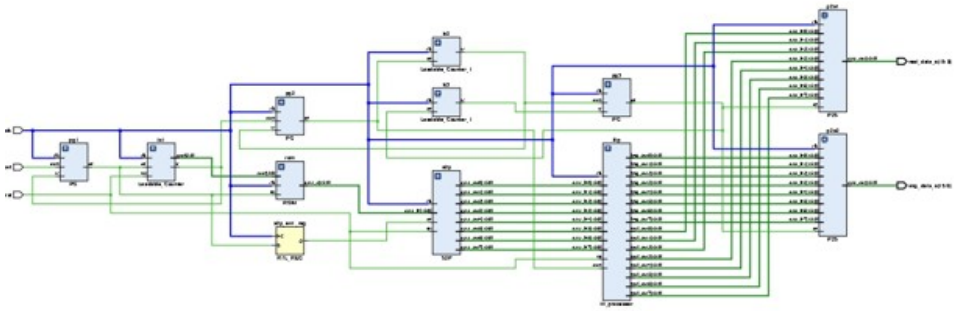


Fig. 1: RTL DESIGN

I-B Verilog code

```
    `timescale 1ns / 1ps
module TOP(
    input clk, start, rst,
    output signed [15:0] real_data_o, img_data_o
);
    wire tc1, tc2, tc3;
    wire en1, en2, en3;
    wire signed [15:0] rom_out;
    reg s2p_en1 , s2p_en2;

    wire [3:0] addr ;

    // Declare unpacked arrays as separate signals
    wire signed [15:0] s2p_out0, s2p_out1, s2p_out2, s2p_out3;
    wire signed [15:0] s2p_out4, s2p_out5, s2p_out6, s2p_out7;

    wire signed [15:0] fftp_real_out0, fftp_real_out1, fftp_real_out2, fftp_real_out3;
```

```
wire signed [15:0] fftp_real_out4, fftp_real_out5, fftp_real_out6, fftp_real_out7;

wire signed [15:0] fftp_img_out0, fftp_img_out1, fftp_img_out2, fftp_img_out3;
wire signed [15:0] fftp_img_out4, fftp_img_out5, fftp_img_out6, fftp_img_out7;



localparam integer fft_delay = 4;

// Control logic
PG pg1(clk, start, tc1, en1);
Loadable_Counter lc1(clk, en1, rst , addr, tc1);
ROM rom(clk, en1, addr, rom_out);


always @(posedge clk) begin
    if (en1) s2p_en1 <= 1;
    else s2p_en1 <= 0;
end



// S2P instantiation
S2P s2p(
    clk, rst, s2p_en1 , rom_out
    ,s2p_out0, s2p_out1, s2p_out2, s2p_out3,
               s2p_out4, s2p_out5, s2p_out6, s2p_out7
);

// Stage 2 PG and Counter
PG pg2(clk, tc1, tc2, en2);
Loadable_Counter_1 lc2(clk, en2  , tc2);

// FFT Processor instantiation
fft_processor fftp(
    .clk(clk), .start(en2), .rst(rst),
    .data_in0(s2p_out0), .data_in1(s2p_out1), .data_in2(s2p_out2), .data_in3(s2p_out3),
    .data_in4(s2p_out4), .data_in5(s2p_out5), .data_in6(s2p_out6), .data_in7(s2p_out7),
    .real_out0(fftp_real_out0), .real_out1(fftp_real_out1), .real_out2(fftp_real_out2), .re
    .real_out4(fftp_real_out4), .real_out5(fftp_real_out5), .real_out6(fftp_real_out6), .re
    .img_out0(fftp_img_out0), .img_out1(fftp_img_out1), .img_out2(fftp_img_out2), .img_out
    .img_out4(fftp_img_out4), .img_out5(fftp_img_out5), .img_out6(fftp_img_out6), .img_out
);

// P2S instances for real and imaginary outputs
P2S p2s1(
    .clk(clk), .en(en3),
    .data_in0(fftp_real_out0), .data_in1(fftp_real_out1), .data_in2(fftp_real_out2), .data_
    .data_in4(fftp_real_out4), .data_in5(fftp_real_out5), .data_in6(fftp_real_out6), .data_
    .data_out(real_data_o)
);

P2S p2s2(
    .clk(clk), .en(en3),
    .data_in0(fftp_img_out0), .data_in1(fftp_img_out1), .data_in2(fftp_img_out2), .data_in
    .data_in4(fftp_img_out4), .data_in5(fftp_img_out5), .data_in6(fftp_img_out6), .data_in
    .data_out(img_data_o)
);
```

```
    // Stage 3 PG and Counter
    PG pg3(clk, tc2, tc3, en3);
    Loadable_Counter_1 lc3(clk, en3  , tc3);

endmodule

////////////PG
module PG(input clk , input start , input tc , output reg en);
always @(posedge clk)begin
    if(start == 1)en <= 1 ;
    else en <= 0 ;
end
endmodule//////////PG


////////////Loadable_Counter
module Loadable_Counter(input clk , en , rst , output reg [3:0]addr , output reg tc);

reg en_internal;
always @(posedge clk)begin
    if(rst)begin
        addr <= 4'b0111 ;
        tc <= 0 ;
        en_internal <= 0 ;
    end
    else if(en || en_internal)begin
        if(addr == 0)begin
            tc <= 1 ;
            en_internal <= 0 ;

        end
        else begin
            en_internal <= 1 ;
            addr <= addr - 1 ;
        end
    end
    else begin
        tc <= 0 ;
        addr <= 4'b0111 ;
        en_internal <= 0 ;
    end
end

endmodule//////////Loadable_Counter+%

module Loadable_Counter_1(input clk , en  , output reg tc);
    reg en_internal ;
    integer count ;
    always @(posedge clk)begin
        if(en)begin
            en_internal <= 1'b1 ;
                count <= 6 ;
                tc <= 1'b0 ;
        end
        else if(en_internal == 1)begin
            if(count == 0)begin
                en_internal <= 1'b0 ;
```

```
                count <= 7 ;
                tc <= 1'b1 ;
            end
            else begin
                en_internal <= 1'b1 ;
                count <= count - 1 ;
                tc <= 1'b0 ;
            end
        end
        else begin
            count <= 7 ;
            tc <= 1'b0 ;
            en_internal <= 1'b0 ;
        end
    end
endmodule

module BRAM (
    input clk,

    input [3:0] read_address,
    output reg [15:0] read_data );

    // BRAM: 16 locations, 16 bits wide
    reg [15:0] bram [0:15];

    // Initialize memory from file
//    initial $readmemh("mem_init0.mem", bram);
    // hard coding initial values for simulation

    // Test - 1
   //initial begin
 //       bram[0] = 16'h0000 ;
 //       bram[1] = 16'h0100 ;
 //       bram[2] = 16'h0200 ;
 //       bram[3] = 16'h0300 ;
 //       bram[4] = 16'h0400 ;
 //       bram[5] = 16'h0500 ;
 //       bram[6] = 16'h0600 ;
 //       bram[7] = 16'h0700 ;
   // end



//    // Test - 2
//    initial begin
//        bram[0] = 16'h0000 ;
//        bram[1] = 16'h03A6 ;
//        bram[2] = 16'hF9A6 ;
//        bram[3] = 16'h07CB ;
//        bram[4] = 16'h01C2 ;
//        bram[5] = 16'hFCA3 ;
//        bram[6] = 16'h0642 ;
//        bram[7] = 16'hFFD3 ;
//    end
    integer i;

    always @(posedge clk) begin
```

```
        read_data <= bram[read_address];
    end
endmodule
module ROM (
    input clk,
   input rst ,
    input [3:0] addr,  // 4-bit address (16 locations)
    output signed [15:0] data_o
);

    BRAM bram_inst(clk , addr , data_o) ;
endmodule


module S2P(
    input clk, rst, en,
    input [15:0] data_in,
    output reg signed [15:0] data_out0, data_out1, data_out2, data_out3,
                                data_out4, data_out5, data_out6, data_out7
);
    reg en_internal;
    reg [2:0] counter = 3'b111;

    // Internal shift register to hold 8 values
    reg signed [15:0] shift_reg [7:0];

    always @(posedge clk) begin : block_name
    integer i ;
        if (rst) begin
            counter <= 3'b111;
            en_internal <= 0;
            for ( i = 0; i < 8; i = i + 1) begin
                shift_reg[i] <= 0;
            end
        end
        else begin
            if (en || en_internal) begin
                if (counter == 0) begin
                    en_internal <= 0;
                    counter <= 3'b111;
                end
                else begin
                    en_internal <= 1;
                    counter <= counter - 1;
                end

                // Shift operation
                shift_reg[0] <= data_in;
                for ( i = 1; i < 8; i = i + 1) begin
                    shift_reg[i] <= shift_reg[i-1];
                end
            end
            else begin
                en_internal <= 0;
                counter <= 3'b111 ;
            end
        end
    end
```

```
    // Assigning the outputs explicitly
    always @(*) begin
        data_out0 = shift_reg[0];
        data_out1 = shift_reg[1];
        data_out2 = shift_reg[2];
        data_out3 = shift_reg[3];
        data_out4 = shift_reg[4];
        data_out5 = shift_reg[5];
        data_out6 = shift_reg[6];
        data_out7 = shift_reg[7];
    end

endmodule


//////////BF1
module BF1(input clk , input signed [15:0]in1 , in2 , output reg signed [15:0]out1 , out2) ;
    always@(posedge clk)begin
        out1 <= in1 + in2 ;
        out2 <= in1 - in2 ;
    end
endmodule//////////BF1

//////////BF2
module BF2(input clk , input signed [15:0]in1 , in2 , output reg signed [15:0]out1 , out2, out
    always @(posedge clk)begin
        out1 <= in1 ;
        out2 <= -in2 ;
        out3 <= in1 ;
        out4 <= in2 ;
    end
endmodule//////////BF2

module CMX1(input clk, input signed [15:0]in1 , in2 , output reg signed [15:0]out1 , out2) ;
    wire signed [15:0] sum, diff;
    wire signed [31:0] mult_sum, mult_diff;

    // Compute sum and difference
    assign sum = in1 + in2;  // (a + b)
    assign diff = in1 - in2; // (a - b)

    // Multiply by 0.7071 (approx 181 in Q8.8)
    assign mult_sum = sum * 16'sd181;  // 0.7071 * (a + b)
    assign mult_diff = diff * 16'sd181; // 0.7071 * (a - b)

    always @(posedge clk) begin
        // Adjust for fixed-point by shifting right by 8 bits
        out1 <= mult_sum >>> 8; // Real part
        out2 <= -(mult_diff >>> 8); // Imaginary part (negation for -j term)
    end

endmodule

module CMX2(input clk, input signed [15:0]in1 , in2 , output reg signed [15:0]out1 , out2) ;
    wire signed [15:0] sum, diff;
    wire signed [31:0] mult_sum, mult_diff;
```

```
    // Compute sum and difference
    assign sum = in1 + in2;  // (a + b)
    assign diff = in1 - in2; // (a - b)

    // Multiply by 0.7071 (approx 181 in Q8.8)
    assign mult_sum = sum * 16'sd181;  // 0.7071 * (a + b)
    assign mult_diff = diff * 16'sd181; // 0.7071 * (a - b)

    always @(posedge clk) begin
        // Adjust for fixed-point by shifting right by 8 bits
        out2 <= -(mult_sum >>> 8); //imaginary part
        out1 <= -(mult_diff >>> 8); // Real part
    end

endmodule

module delay(input clk , input signed [15:0]in , output reg signed [15:0] out);
    always@(posedge clk)begin
        out <= in ;
    end
endmodule



module fft_processor (
    input clk, start, rst,
    input signed [15:0] data_in0, data_in1, data_in2, data_in3,
                        data_in4, data_in5, data_in6, data_in7,
    output reg signed [15:0] real_out0, real_out1, real_out2, real_out3,
                             real_out4, real_out5, real_out6, real_out7,
    output reg signed [15:0] img_out0, img_out1, img_out2, img_out3,
                             img_out4, img_out5, img_out6, img_out7
);

    reg en_internal ;
    reg [2:0] counter ;

    ////////////////////////////stage1
    wire [15:0] stage1_o[7:0];

    BF1 stage1_1(clk, data_in0, data_in4, stage1_o[0], stage1_o[1]);
    BF1 stage1_2(clk, data_in2, data_in6, stage1_o[2], stage1_o[3]);
    BF1 stage1_3(clk, data_in1, data_in5, stage1_o[4], stage1_o[5]);
    BF1 stage1_4(clk, data_in3, data_in7, stage1_o[6], stage1_o[7]);

    ////////////////////////////stage2
    wire [15:0] stage2_o[11:0];
    BF1 stage2_1(clk, stage1_o[0], stage1_o[2], stage2_o[0], stage2_o[1]);
    BF2 stage2_2(clk, stage1_o[1], stage1_o[3], stage2_o[2], stage2_o[3], stage2_o[4], stage2_
    BF1 stage2_3(clk, stage1_o[4], stage1_o[6], stage2_o[6], stage2_o[7]);
    BF2 stage2_4(clk, stage1_o[5], stage1_o[7], stage2_o[8], stage2_o[9], stage2_o[10], stage2_

    ////////////////////////////stage3
    wire [15:0] stage3_o[11:0];
    delay stage3_1(clk, stage2_o[0], stage3_o[0]);
    delay stage3_2(clk, stage2_o[1], stage3_o[1]);
    delay stage3_3(clk, stage2_o[2], stage3_o[2]);
    delay stage3_4(clk, stage2_o[3], stage3_o[3]);
```

```
    delay stage3_5(clk, stage2_o[4], stage3_o[4]);
    delay stage3_6(clk, stage2_o[5], stage3_o[5]);
    delay stage3_7(clk, stage2_o[6], stage3_o[6]);
    delay stage3_8(clk, stage2_o[7], stage3_o[7]);
    CMX1 cmx1(clk, stage2_o[8], stage2_o[9], stage3_o[8], stage3_o[9]);
    CMX2 cmx2(clk, stage2_o[10], stage2_o[11], stage3_o[10], stage3_o[11]);

    ////////////////////////////stage4
    wire [15:0] stage4_real_o[7:0], stage4_imag_o[7:0];

    ////////////////////real part
    BF1 stage4_real_1(clk, stage3_o[0], stage3_o[6], stage4_real_o[0], stage4_real_o[4]);
    BF1 stage4_real_2(clk, stage3_o[2], stage3_o[8], stage4_real_o[1], stage4_real_o[5]);
    BF2 stage4_real_3(clk, stage3_o[1], stage3_o[7], stage4_real_o[2], stage4_imag_o[2], stage
    BF1 stage4_real_4(clk, stage3_o[4], stage3_o[10], stage4_real_o[3], stage4_real_o[7]);

    ////////////////////imaginary part
    assign stage4_imag_o[0] = 0;
    assign stage4_imag_o[4] = 0;
    BF1 stage4_imag_2(clk, stage3_o[3], stage3_o[9], stage4_imag_o[1], stage4_imag_o[5]);
    BF1 stage4_imag_4(clk, stage3_o[5], stage3_o[11], stage4_imag_o[3], stage4_imag_o[7]);

    always @(posedge clk) begin
        if (rst) begin
            en_internal <= 0;
            counter <= 0;
            real_out0 <= 0; real_out1 <= 0; real_out2 <= 0; real_out3 <= 0;
            real_out4 <= 0; real_out5 <= 0; real_out6 <= 0; real_out7 <= 0;
            img_out0 <= 0; img_out1 <= 0; img_out2 <= 0; img_out3 <= 0;
            img_out4 <= 0; img_out5 <= 0; img_out6 <= 0; img_out7 <= 0;
        end
        else begin
            real_out0 <= stage4_real_o[0];
            real_out1 <= stage4_real_o[1];
            real_out2 <= stage4_real_o[2];
            real_out3 <= stage4_real_o[3];
            real_out4 <= stage4_real_o[4];
            real_out5 <= stage4_real_o[5];
            real_out6 <= stage4_real_o[6];
            real_out7 <= stage4_real_o[7];

            img_out0 <= stage4_imag_o[0];
            img_out1 <= stage4_imag_o[1];
            img_out2 <= stage4_imag_o[2];
            img_out3 <= stage4_imag_o[3];
            img_out4 <= stage4_imag_o[4];
            img_out5 <= stage4_imag_o[5];
            img_out6 <= stage4_imag_o[6];
            img_out7 <= stage4_imag_o[7];
        end
    end

endmodule

module P2S (input clk , en ,
input [15:0] data_in0 , data_in1 , data_in2 , data_in3 , data_in4 , data_in5, data_in6 , data_
output reg [15:0] data_out) ;
```

```
    integer count ;
    reg en_internal ;
    always @(posedge clk)begin
        if(en )begin
            count <= 0;
            en_internal <= 1'b1 ;
        end
        else if(en_internal)begin
            count <= count + 1 ;
            case(count)
                0 : data_out <= data_in0 ;
                1 : data_out <= data_in1 ;
                2 : data_out <= data_in2 ;
                3 : data_out <= data_in3 ;
                4 : data_out <= data_in4 ;
                5 : data_out <= data_in5 ;
                6 : data_out <= data_in6 ;
                7 : data_out <= data_in7 ;
            endcase
            en_internal <= (count == 7 ) ? 1'b0 : 1'b1 ;
        end
        else begin
        data_out <= 0 ;
            en_internal <= 0 ;
            count <= 0 ;
        end

    end

endmodule
```

I-C Pre-Synthesis Simulation

Pre-synthesis simulation is the process of testing your RTL design (in Verilog or VHDL) before it undergoes synthesis. It verifies the functional correctness of the code using a testbench and simulation tool. This step helps identify logical errors early, ensures the design behaves as intended, and enables quicker debugging. Since it uses high-level HDL, simulations are faster and more readable compared to post-synthesis or gate-level simulations.

testbench:

```
module tb ;
    reg clk , start ,rst ;
    wire signed [15:0] real_data_o , img_data_o;
    TOP top(clk , start ,rst , real_data_o , img_data_o) ;
    always #5 clk = ~clk ;
    initial begin
        rst = 0 ;
        clk = 0 ;
        start = 0 ;
        #10;
        start = 1 ;
        #10 ;
        start = 0 ;
    end
    initial begin
        #400;$finish;
    end
endmodule
```

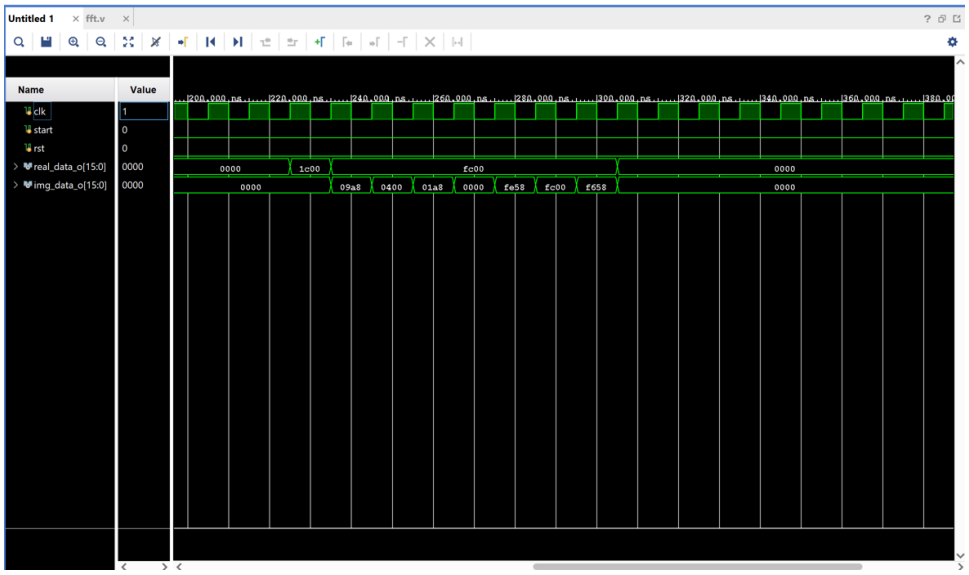


Fig. 2: OUTPUT WAVEFORM

I-D Synthesis

Synthesis is the process of converting RTL code (written in Verilog or VHDL) into a gate-level netlist using synthesis tools like Cadence Genus. It maps the high-level design to technology-specific gates and flip-flops, optimizing for performance, area, and power. The output is a netlist, ready for place-and-route steps in ASIC or FPGA design. Synthesis also involves applying constraints like timing and area requirements to ensure the design meets specifications.
After synthesis, Genus generates several important files, including the gate-level netlist (.v), updated constraints file (.sdc), and various reports such as area (area.rpt), timing (timing.rpt), and power (power.rpt). These files are used for verification, analysis, and passed to place-and-route tools for further implementation.

TIMING REPORT

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 30 2025  10:08:18 am
  Module:                 TOP
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (92111 ps) Setup Check with Pin fftp/cmx2/out1_reg[14]/CK->D
          Group: clk
     Startpoint: (R) fftp/stage2_4/out1_reg[0]/CK
          Clock: (R) clk
       Endpoint: (R) fftp/cmx2/out1_reg[14]/D
          Clock: (R) clk

                    Capture       Launch
       Clock Edge:+  100000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=  100000            0

            Setup:-     228
      Uncertainty:-      50
```

```
     Required Time:=   99722
      Launch Clock:-       0
         Data Path:-    7611
             Slack:=   92111

#-------------------------------------------------------------------------------------------
#                    Timing Point                     Flags    Arc   Edge   Cell     Fanout Lo
#                                                                                        (f
#-------------------------------------------------------------------------------------------
  fftp/stage2_4/out1_reg[0]/CK                        -       -      R     (arrival)   1181
  fftp/stage2_4/out1_reg[0]/Q                         -       CK->Q  F     DFFQX1         6 12
  fftp/cmx2/sub_310_23/g453/Y                         -       A->Y   R     CLKINVX1       2  4
  fftp/cmx2/sub_310_23/g451/Y                         -       B->Y   F     NAND2XL        2  7
  fftp/cmx2/sub_310_23/g448/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g447/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g446/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g445/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g444/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g443/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g442/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g441/CO                        -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/sub_310_23/g440/S                         -       CI->S  R     ADDFX1         7 29
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1949/Y  -       A->Y   F     CLKINVX1       2  9
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1921/Y  -       A1->Y  R     OAI22X1        4  8
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1901/Y  -       A->Y   F     INVX1          3  6
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1859/Y  -       B->Y   R     MXI2XL         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1813/S  -       CI->S  F     ADDFXL         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1793/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1792/CO -       CI->CO F     ADDFXL         1  2
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1791/Y  -       A->Y   R     INVX1          1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1790/CO -       CI->CO R     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1788/CO -       CI->CO R     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1786/CO -       CI->CO R     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1784/CO -       CI->CO R     ADDFX1         1  2
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1783/Y  -       A->Y   F     INVX1          1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1781/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1780/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1779/CO -       CI->CO F     ADDFX1         2  6
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1778/Y  -       B0->Y  R     OAI2BB1X1      1  2
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1776/Y  -       B0->Y  F     OAI21X1        1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1774/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1773/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1772/CO -       CI->CO F     ADDFX1         1  4
  fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1771/S  -       CI->S  R     ADDFXL         1  1
  fftp/cmx2/out1_reg[14]/D                            <<<     -      R     DFFQX1         1
#-------------------------------------------------------------------------------------------
```

POWER REPORT

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 30 2025  10:41:48 am
  Module:                 TOP
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================
```

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
TOP	3715	197605.129	846080.271	1043685.401
fftp	2609	150730.444	726450.669	877181.112
cmx2	472	16853.845	193979.027	210832.872
csa_tree_m..17_pad_groupi	200	5503.289	72964.547	78467.836
csa_tree_m..17_pad_groupi	186	5399.193	72622.952	78022.145
sub_310_23	35	1452.946	14215.660	15668.606
add_309_22	19	1334.104	11711.179	13045.282
cmx1	381	15778.396	146311.460	162089.856
csa_tree_m..17_pad_groupi	186	5399.193	59339.680	64738.874
const_mul_293_27	109	4427.840	44293.417	48721.257
sub_290_23	35	1452.946	11678.326	13131.271
add_289_22	19	1334.104	10418.838	11752.941
stage1_1	86	5903.722	15427.243	21330.965
sub_270_21	35	1426.997	3012.155	4439.152
add_269_21	19	1312.412	1717.554	3029.965
stage1_2	86	5903.722	17031.460	22935.182
sub_270_21	35	1426.997	3343.889	4770.886
add_269_21	19	1312.412	2746.243	4058.655
stage1_3	86	5903.722	21782.230	27685.952
sub_270_21	35	1426.997	5027.545	6454.542
add_269_21	19	1312.412	3907.081	5219.493
stage1_4	86	5903.722	18057.351	23961.073
sub_270_21	35	1426.997	3598.301	5025.298
add_269_21	19	1312.412	2869.319	4181.731
stage2_1	86	5903.722	22561.804	28465.526
sub_270_21	35	1426.997	5104.032	6531.029
add_269_21	19	1312.412	4260.368	5572.780
stage2_3	86	5903.722	24452.511	30356.233
sub_270_21	35	1426.997	6021.301	7448.298
add_269_21	19	1312.412	4715.226	6027.638
stage4_imag_2	86	5903.722	17651.665	23555.388
sub_270_21	35	1426.997	4812.925	6239.922
add_269_21	19	1312.412	3415.214	4727.626
stage4_imag_4	86	5903.722	16771.368	22675.090
sub_270_21	35	1426.997	4838.929	6265.926
add_269_21	19	1312.412	3512.545	4824.956
stage4_real_1	86	5903.722	26591.556	32495.278
sub_270_21	35	1426.997	7123.309	8550.306
add_269_21	19	1312.412	6100.331	7412.743
stage4_real_2	86	5903.722	17412.108	23315.830
sub_270_21	35	1426.997	5062.343	6489.340
add_269_21	19	1312.412	3511.930	4824.342
stage4_real_4	86	5903.722	17386.279	23290.001
sub_270_21	35	1426.997	5123.190	6550.187
add_269_21	19	1312.412	3542.282	4854.693
stage2_4	90	5303.011	23011.965	28314.975
stage2_2	89	5289.996	19747.443	25037.439
stage4_real_3	89	5289.996	16543.871	21833.867
stage3_1	16	1582.157	6986.256	8568.413
stage3_2	16	1582.157	6003.117	7585.274
stage3_3	16	1582.157	4562.966	6145.123
stage3_4	16	1582.157	4987.951	6570.107
stage3_5	16	1582.157	4562.966	6145.123
stage3_6	16	1582.157	4526.342	6108.498

```
    stage3_7                        16   1582.157   6591.226    8173.383
    stage3_8                        16   1582.157   6026.245    7608.402
  s2p                              399  18361.907  58613.005   76974.912
  p2s1                             234   8776.775  14539.915   23316.690
    inc_add_432_28                  76   1382.530    710.860    2093.389
  p2s2                             218   8774.791  13809.678   22584.469
    inc_add_432_28                  76   1382.530    869.438    2251.967
  lc2                              126   6853.889   9883.324   16737.213
    sub_139_32                      62   1875.065    523.929    2398.995
  rom                              108   3025.172   2645.432    5670.604
    bram_inst                      108   3025.172   2645.432    5670.604
  lc1                               17    686.613   1482.970    2169.583
  pg1                                1     98.885    206.498     305.383
  pg2                                1     98.885    214.307     313.192
  pg3                                1     98.885    136.057     234.941
```

CELL REPORT

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 30 2025  10:41:48 am
  Module:                 TOP
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================
```

Gate	Instances	Area	Library
ADDFX1	394	7753.683	slow
ADDFXL	122	2400.887	slow
ADDHXL	2	24.221	slow
AND2X1	24	108.994	slow
AND2X4	1	8.326	slow
AND3X1	3	18.166	slow
AO21X1	6	40.873	slow
AOI211XL	1	5.298	slow
AOI21X1	112	508.637	slow
AOI21XL	16	72.662	slow
AOI222XL	15	124.889	slow
AOI22X1	71	429.919	slow
AOI22XL	110	666.072	slow
AOI2BB1X1	21	127.159	slow
AOI2BB1XL	3	18.166	slow
CLKINVX1	76	172.573	slow
CLKXOR2X1	9	74.933	slow
DFFQX1	1087	17277.756	slow
DFFTRXL	91	1859.703	slow
INVX1	257	583.570	slow
INVXL	32	72.662	slow
MXI2XL	214	1295.813	slow
NAND2BX1	21	95.369	slow
NAND2BXL	3	13.624	slow
NAND2XL	147	445.057	slow
NAND3BX1	1	6.055	slow
NAND3X1	16	72.662	slow

NAND4XL	34	180.142	slow
NOR2BX1	227	1030.898	slow
NOR2BXL	2	9.083	slow
NOR2X1	3	11.354	slow
NOR2XL	279	844.700	slow
NOR3BX1	1	6.055	slow
NOR3X1	6	27.248	slow
NOR4X1	7	42.386	slow
OA21X1	3	20.436	slow
OA21XL	22	149.866	slow
OAI211X1	1	5.298	slow
OAI21X1	92	417.809	slow
OAI21XL	37	168.032	slow
OAI22X1	14	84.773	slow
OAI2BB1X1	16	84.773	slow
OAI2BB1XL	33	174.844	slow
OAI32X1	2	13.624	slow
OR2X1	31	140.783	slow
OR2XL	3	13.624	slow
OR4X1	1	6.812	slow
OR4XL	24	163.490	slow
SDFFQX1	3	61.309	slow
XNOR2X1	16	133.214	slow
XNOR2XL	1	8.326	slow
XOR2XL	2	16.652	slow
total	3715	38093.263	

Type	Instances	Area	Area %
sequential	1181	19198.769	50.4
inverter	365	828.806	2.2
logic	2169	18065.689	47.4
physical_cells	0	0.000	0.0
total	3715	38093.263	100.0

To view the schematic after synthesis in Cadence Genus, you typically use the GUI mode. To Launch Genus in GUI Mode we use : show-gui

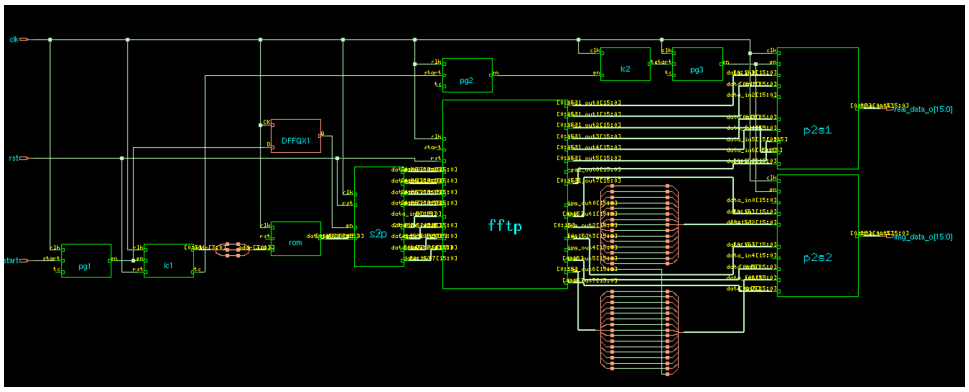


Fig. 3: GENUS RTL DIGRAM

I-E Post-Synthesis

Post-synthesis refers to the stage following the synthesis of a digital design, where the RTL code has been transformed into a gate-level netlist. This netlist is now mapped to specific technology libraries and optimized for area, power, and timing. Post-synthesis analysis involves verifying that the synthesized design meets the required timing and functionality through simulations and timing reports. Tools are used to perform static timing analysis (STA), power estimation, and logic equivalence checking. This step ensures that the design is ready for physical implementation, such as placement and routing. Any violations detected here must be addressed before proceeding further.

STEPS TO NCLAUNCH

Step 1: In the Terminal write the nclaunch to open the tool.
Step 2: Copy the file in the download folder of slow.v from the directory saved in vlog folder.
Step 3: Send the slow.v, synthesis after netlist.v and the testbench .v file in the compiler hdl and the result store in the Worklib file.
Step 4: Open the workLib file and select the module name file in .v format same for testbench .v format and add the elaborater on each .v file.
Step 5: Open the snapshots, select the generated testbench file and open the gui graphics.

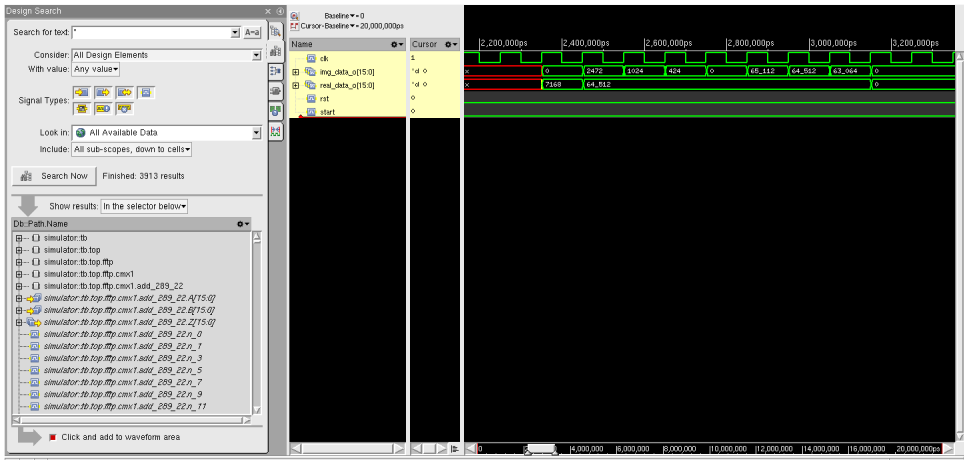


Fig. 4: POST SYNTHESIS WAVEFORM- netlist output

I-F Physical design

Physical design is the process of transforming a synthesized netlist into a physical layout that can be manufactured on a silicon chip. It is a crucial stage in ASIC design where the logical representation of the circuit is converted into geometric representations of standard cells, wires, and vias. The process includes several key steps: floorplanning, placement, clock tree synthesis (CTS), routing, and design rule checking (DRC). During floorplanning, the chip area is defined, and major functional blocks are positioned. In placement, standard cells are placed optimally based on timing and connectivity. CTS ensures balanced clock signal distribution across the chip. Routing connects all components electrically while adhering to spacing rules. The final layout is verified against the design rules of the fabrication process. Tools like Cadence Innovus or Synopsys IC Compiler are commonly used. Physical design significantly impacts performance, power, and area, and is essential for creating reliable, manufacturable chips.

Step 1 : In the terminal type the command innovus to open the tool,then a new window of cadence open.
Step 2 : Go to the file, open the design, now fill all the essential files ,LEF files , netkist after synthesis.v files, and in MMMC add the max Timing with slow .lib and min timing with fast.lib, max delay with max timing and vice versa,add the setup and hold condition,in last all the netlist and design floor occur on the screen. The format will be of .view and .global.
Step 3 : Open the specify floorplan fill the necessary information like core utilization and add the aspect ratio and core to IO boundary to 10,click ok.
Step 4 :Do the power planning add the strips, rings with appropriate metals selection both horizontal and vertical with high metals.
Step 5 : Add the nets,by clicking on route and then special route and click on all metal layers.
Step 6: Go to place,physical cell, add encp. select the filler and add later the well tap instance with appropriate setting to rows to get the design.
Step 7 : Again Go to place,std. cell, then place io pins and click on ok, all the std. cell come into the die are.

Step 8 : In digital analysis, Go to report timing , preCTS to setup all information occur on the screen, then same will done for the hold .
Step 9: Go to the Clock Debugger click on CTS and apply ,path will be generated and save the design with .enc format.
Step 10:Go to Route ,Nano Route ,route add the timing driven add congestion click on terminal to get the result.
Step 11 : Go to placement add the physical cell add the filler(Select all filters) the fillers.

I-G Imported Design

After completing synthesis in Cadence Genus, the next step is to import the synthesized design into Cadence Innovus for physical implementation. This involves loading the synthesized netlist, SDC constraints, and technology files. Innovus uses this data to begin floorplanning, placement, and routing, forming the foundation for physical design.

Fig. 5: IMPORTED DESIGN

I-H Floor planing

Floorplanning is the initial stage of physical design in Cadence Innovus where the layout of the chip is defined. It involves placing macros, defining core and I/O areas, setting power planning, and creating placement blockages. A good floorplan ensures optimal area utilization, routing efficiency, and better performance in later stages.

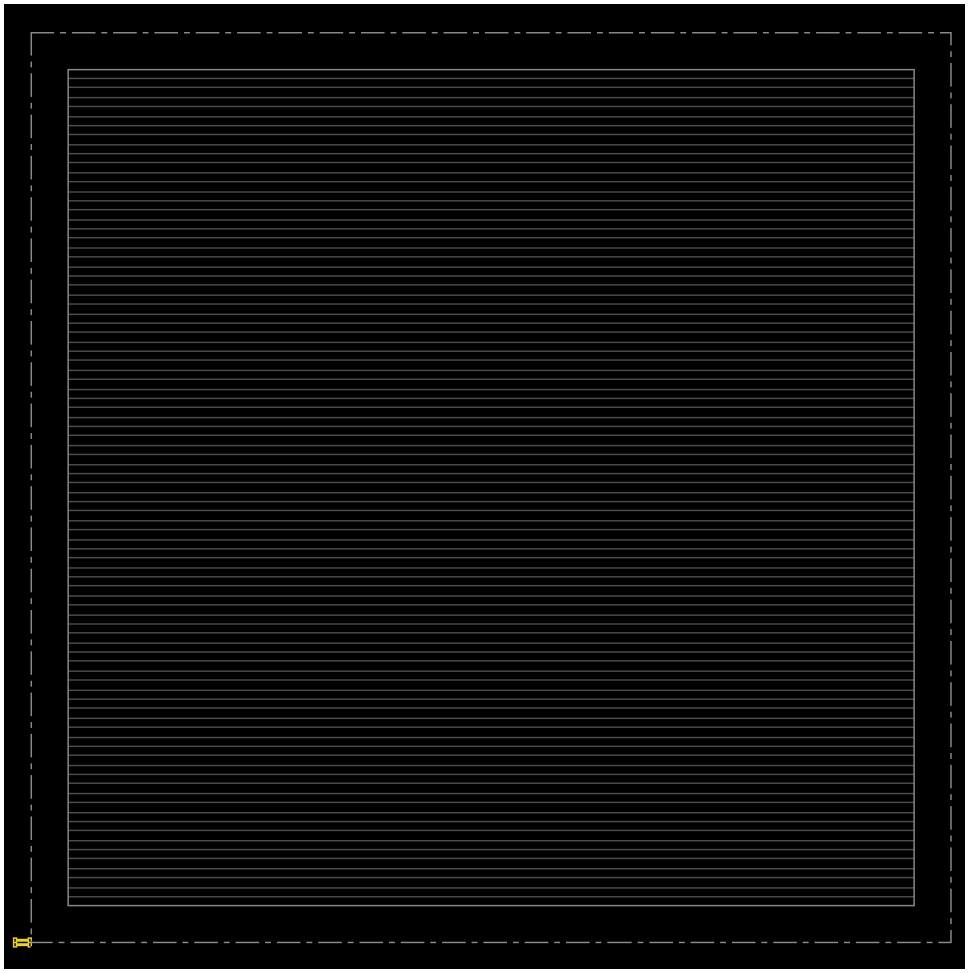


Fig. 6: FLOOR PLANING

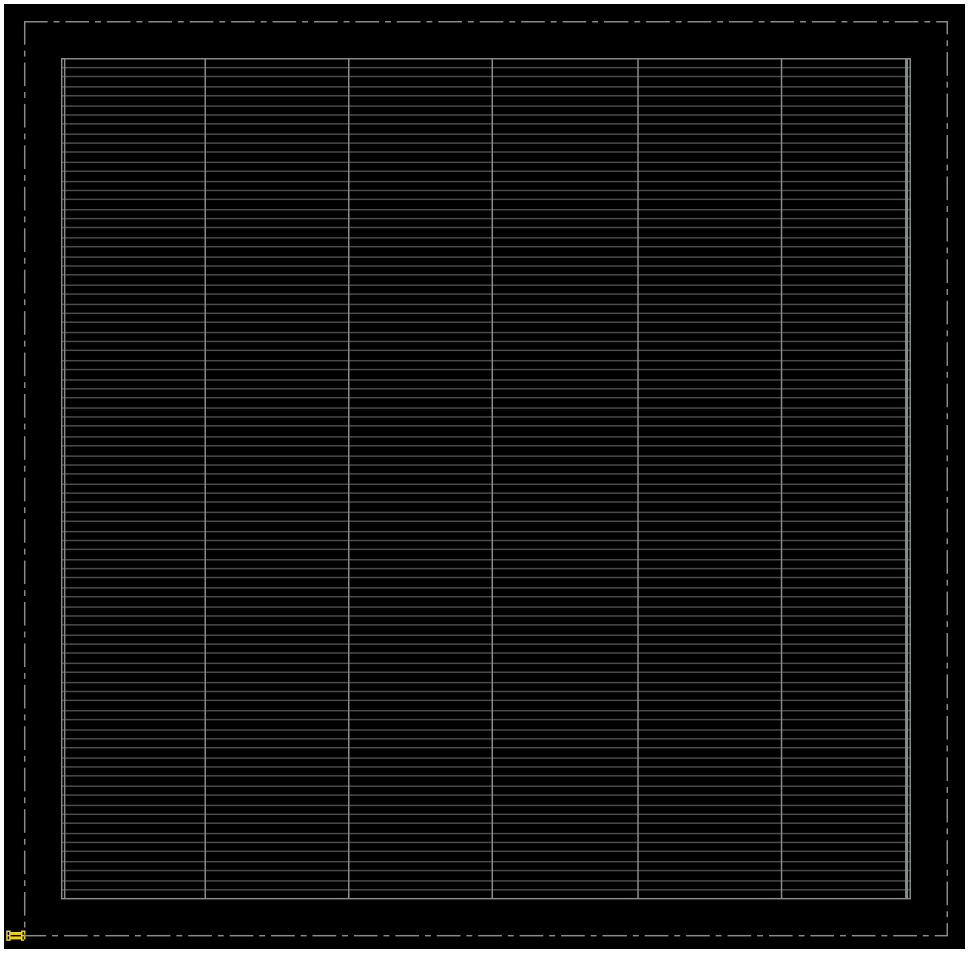


Fig. 7: FILLER

I-I Power Planning

Power planning is a crucial step in physical design where the power distribution network is created. It involves generating power and ground rings, straps, and vias to deliver stable power across the chip. Proper power planning ensures voltage integrity, minimizes IR drop, and supports the chip's performance and reliability requirements.

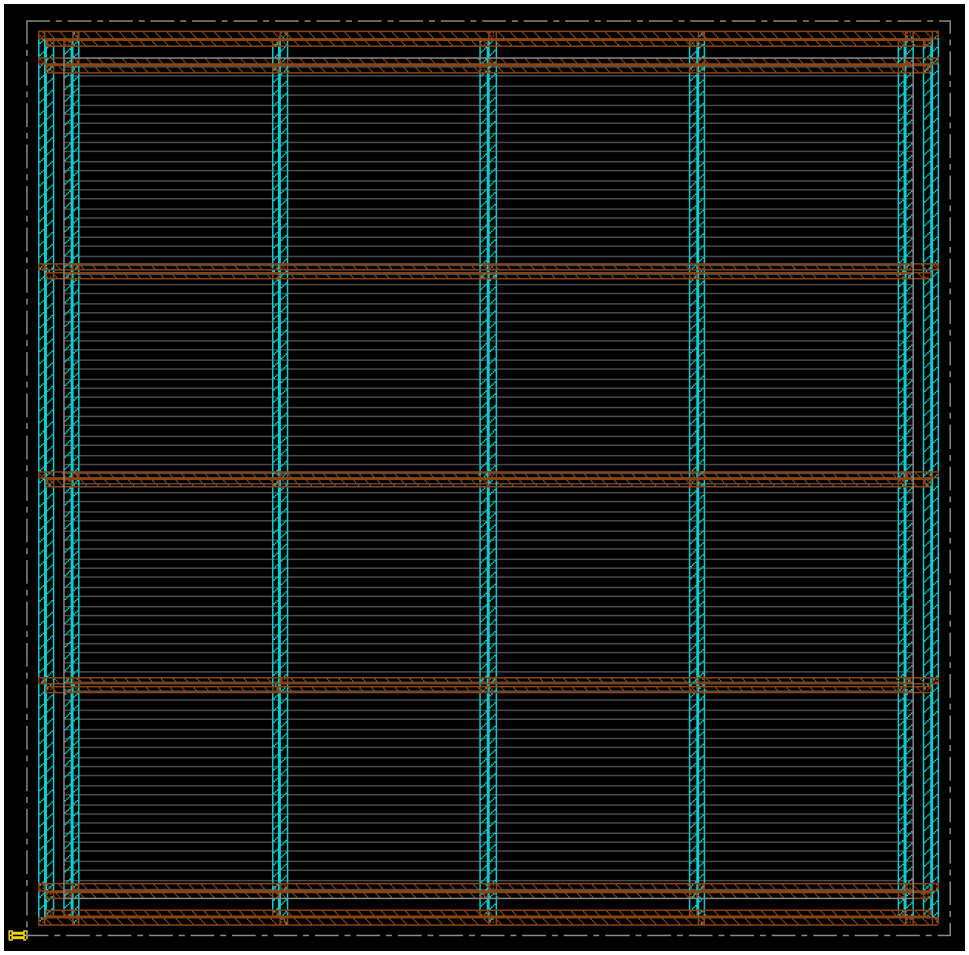


Fig. 8: POWER PLANNING

I-J Routing

Routing is the stage where electrical connections between placed cells are created using metal layers. It includes global and detailed routing to connect nets while avoiding design rule violations. Proper routing ensures signal integrity, meets timing constraints, and minimizes crosstalk, congestion, and resistance-capacitance (RC) delays across the chip layout.

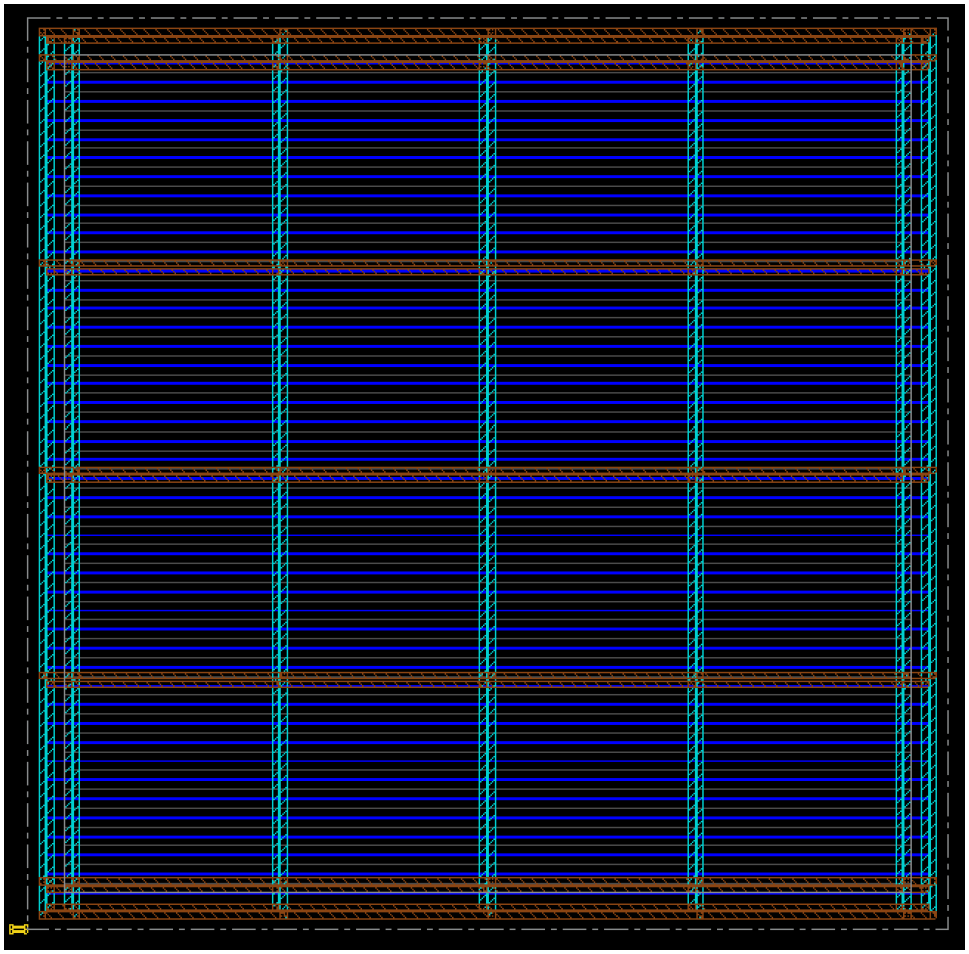


Fig. 9: ROUTING

I-K Placement

Placement is the process of positioning standard cells within the floorplan after power planning. It aims to optimize timing, area, and power while minimizing wirelength and congestion. During placement, no routing is done, but logical connections are considered to ensure efficient layout and prepare the design for successful routing.

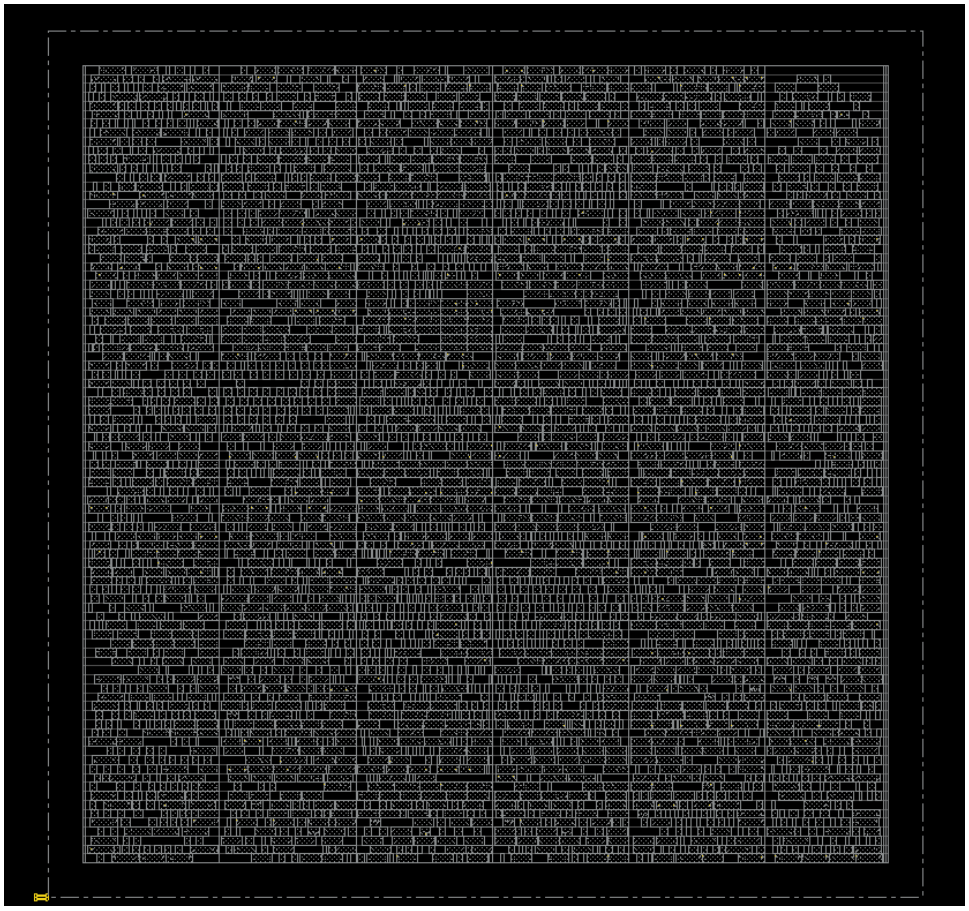


Fig. 10: PLACEMENT OF STANDARD CELL

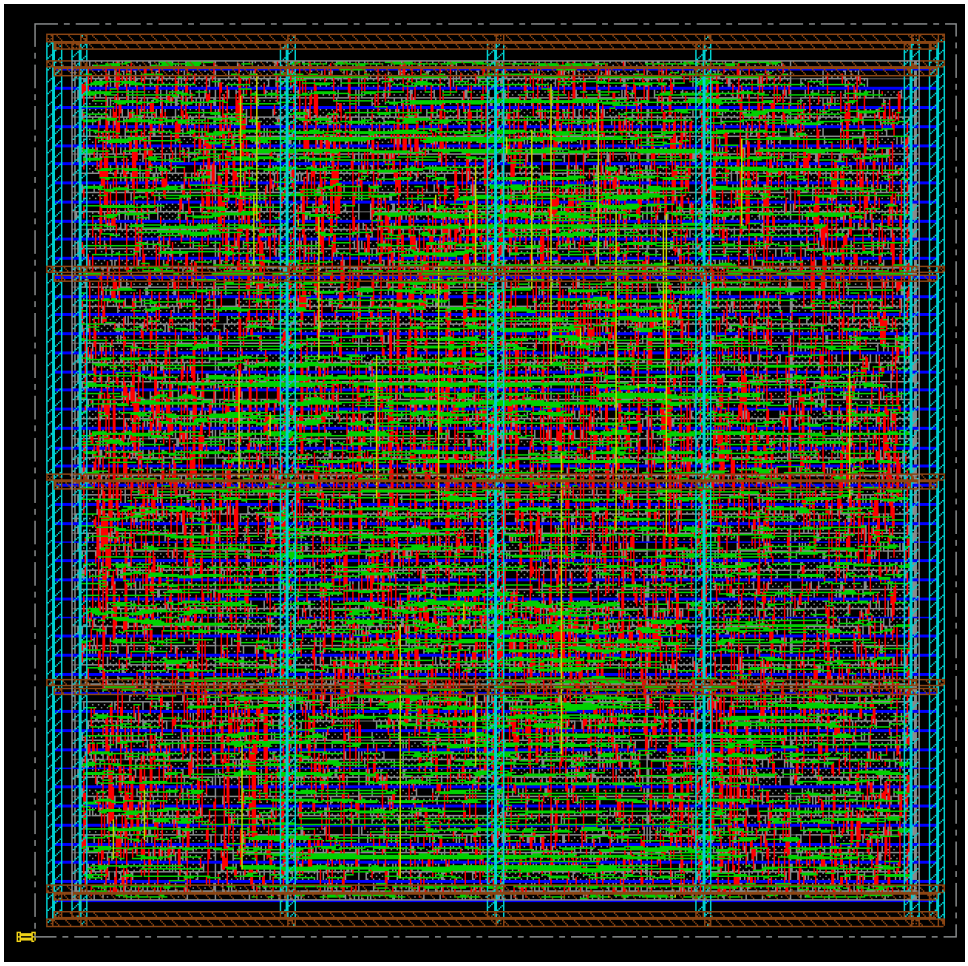


Fig. 11: STANDARD CELL

CELL INFORMATION

Cell information provided includes details about standard cells such as their dimensions, pin locations, functionality, and power requirements. This data comes from the technology library and is essential for placement, routing, and timing analysis. Accurate cell information ensures correct logical and physical implementation throughout the digital design flow.

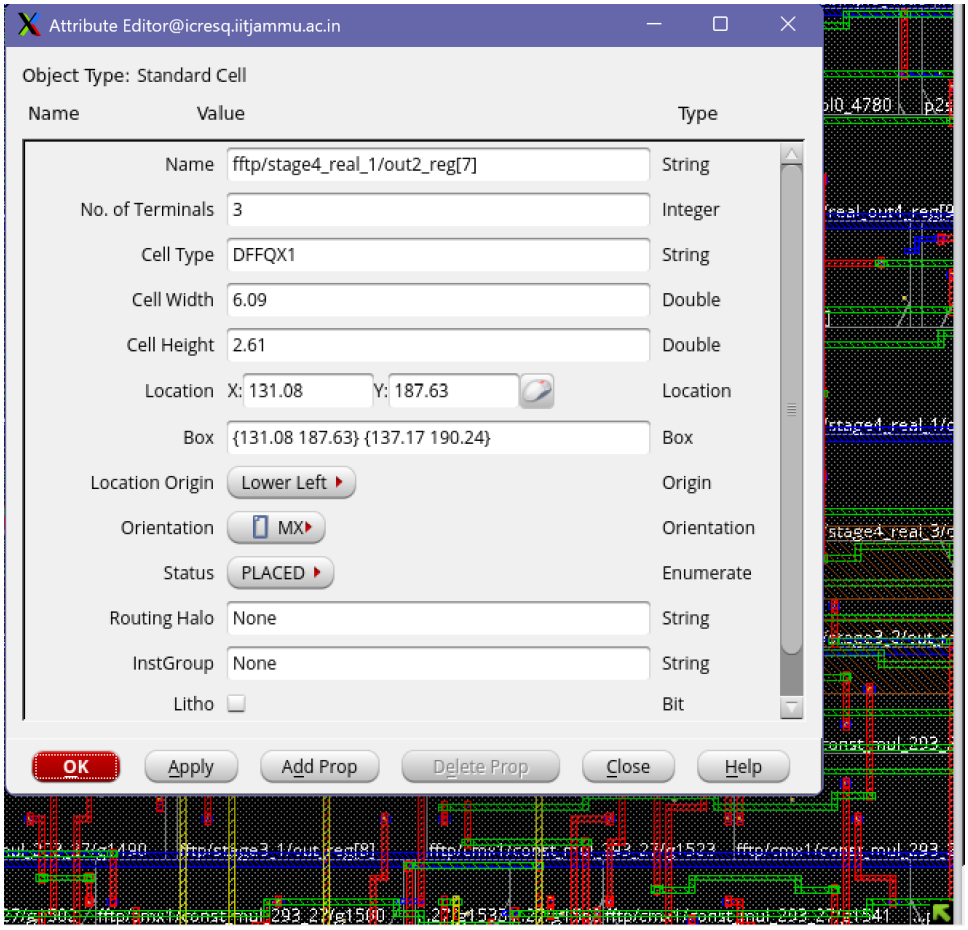


Fig. 12: CELL INFORMATION

METAL WIRE INFORMATION

Metal wire information includes details about the various metal layers used for routing, such as width, spacing, resistance, capacitance, and routing direction. This data is defined in the technology file and is crucial for ensuring signal integrity, minimizing delay, and avoiding design rule violations during the physical design process.

```
#---------------------------
#        Total           4237
#---------------------------
#
#
#  Congestion Analysis: (blocked Gcells are excluded)
#
#                OverCon
#                 #Gcell    %Gcell
#    Layer           (1)   OverCon
#  --------------------------------
#  Metal1        0(0.00%)   (0.00%)
#  Metal2        0(0.00%)   (0.00%)
#  Metal3        0(0.00%)   (0.00%)
#  Metal4        0(0.00%)   (0.00%)
#  Metal5        0(0.00%)   (0.00%)
#  Metal6        0(0.00%)   (0.00%)
#  Metal7        0(0.00%)   (0.00%)
#  Metal8        0(0.00%)   (0.00%)
#  Metal9        0(0.00%)   (0.00%)
#  --------------------------------
#    Total      0(0.00%)   (0.00%)
#
#  The worst congested Gcell overcon (routing demand over resource in number of tracks) = 1
#  Overflow after GR: 0.00% H + 0.00% V
#
[hotspot] +------------+---------------+---------------+
[hotspot] |            |   max hotspot | total hotspot |
[hotspot] +------------+---------------+---------------+
[hotspot] | normalized |          0.00 |          0.00 |
[hotspot] +------------+---------------+---------------+
Local HotSpot Analysis: normalized max congestion hotspot area = 0.00, normalized total congestion hotspot area = 0.00 (area is in unit o
f 4 std-cell row bins)
#Complete Global Routing.
#Total wire length = 63971 um.
#Total half perimeter of net bounding box = 63767 um.
#Total wire length on LAYER Metal1 = 9 um.
#Total wire length on LAYER Metal2 = 33630 um.
#Total wire length on LAYER Metal3 = 28719 um.
#Total wire length on LAYER Metal4 = 1614 um.
#Total wire length on LAYER Metal5 = 0 um.
#Total wire length on LAYER Metal6 = 0 um.
#Total wire length on LAYER Metal7 = 0 um.
#Total wire length on LAYER Metal8 = 0 um.
#Total wire length on LAYER Metal9 = 0 um.
#Total number of vias = 19626
```

Fig. 13: Congestion or Hotspot

```
#Total memory = 1091.64 (MB)
#Peak memory = 1104.85 (MB)
#
#Start Detail Routing..
#start initial detail routing ...
#   number of violations = 2
#
#    By Layer and Type :
#                MetSpc   Totals
#       Metal1        1        1
#       Metal2        1        1
#       Totals        2        2
#cpu time = 00:00:12, elapsed time = 00:00:12, memory = 1117.72 (MB), peak = 1117.89 (MB)
#start 1st optimization iteration ...
#   number of violations = 0
#cpu time = 00:00:00, elapsed time = 00:00:00, memory = 1118.67 (MB), peak = 1118.69 (MB)
#Complete Detail Routing.
#Total wire length = 71024 um.
#Total half perimeter of net bounding box = 63767 um.
#Total wire length on LAYER Metal1 = 3592 um.
#Total wire length on LAYER Metal2 = 33633 um.
#Total wire length on LAYER Metal3 = 29043 um.
#Total wire length on LAYER Metal4 = 4756 um.
#Total wire length on LAYER Metal5 = 0 um.
#Total wire length on LAYER Metal6 = 0 um.
#Total wire length on LAYER Metal7 = 0 um.
#Total wire length on LAYER Metal8 = 0 um.
#Total wire length on LAYER Metal9 = 0 um.
#Total number of vias = 22297
#Up-Via Summary (total 22297):
#
#-----------------------
# Metal1          12975
# Metal2           8878
# Metal3            444
#-----------------------
#                 22297
#
#Total number of DRC violations = 0
#Cpu time = 00:00:12
#Elapsed time = 00:00:12
#Increased memory = 1.00 (MB)
#Total memory = 1092.66 (MB)
#Peak memory = 1118.70 (MB)
#
#start routing for process antenna violation fix ...
```

Fig. 14: Wire length

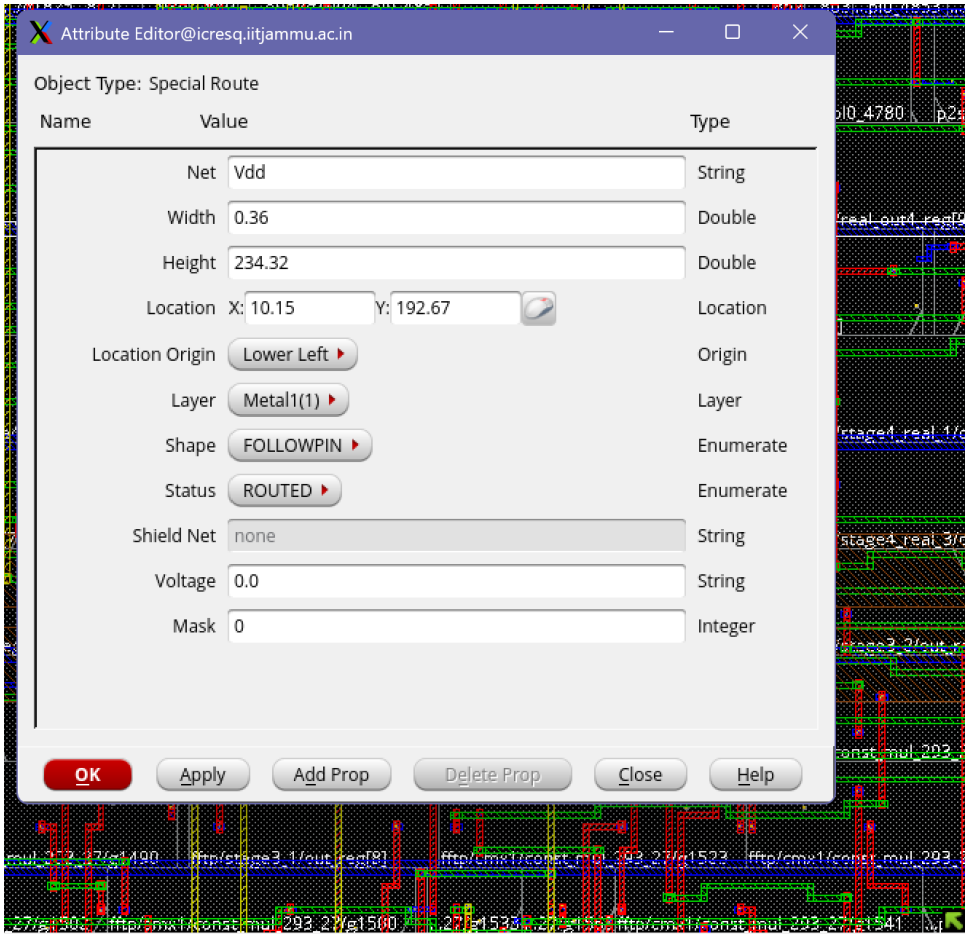
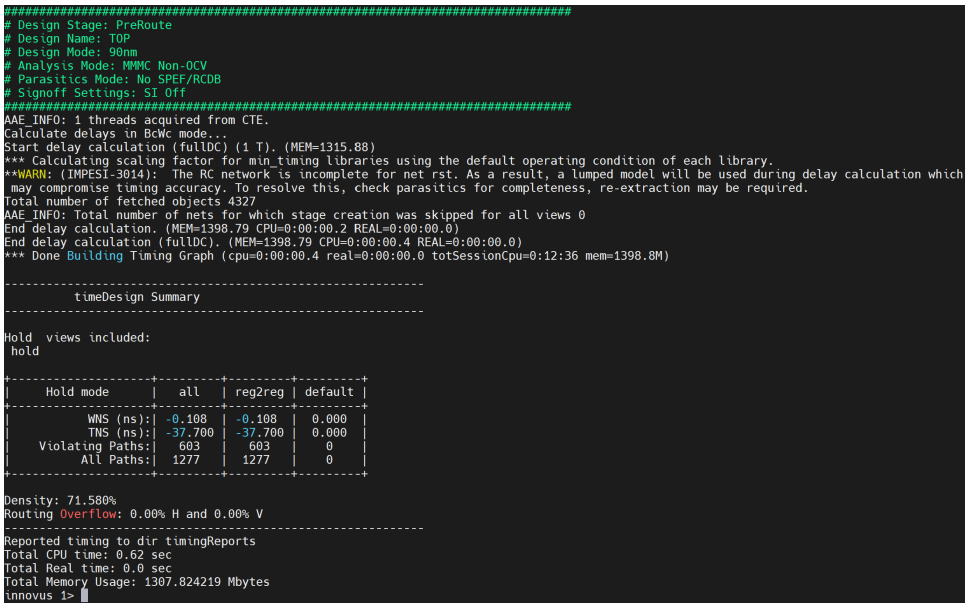


Fig. 15: WIRE INFORMATION

PRE CLOCK TREE SYNTHESIS HOLD ANALYSIS

Pre-CTS hold analysis is performed before clock tree synthesis to ensure that data paths are not violating hold time requirements. At this stage, ideal clocks are used, and the goal is to identify and fix early data arrival issues. Resolving hold violations early helps prevent timing problems after CTS.



```
#############################################################################
# Design Stage: PreRoute
# Design Name: TOP
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#############################################################################
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1315.88)
*** Calculating scaling factor for min_timing libraries using the default operating condition of each library.
**WARN: (IMPESI-3014):  The RC network is incomplete for net rst. As a result, a lumped model will be used during delay calculation which
 may compromise timing accuracy. To resolve this, check parasitics for completeness, re-extraction may be required.
Total number of fetched objects 4327
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1398.79 CPU=0:00:00.2 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1398.79 CPU=0:00:00.4 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.4 real=0:00:00.0 totSessionCpu=0:12:36 mem=1398.8M)

------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 hold

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.108  | -0.108  |  0.000  |
|           TNS (ns):| -37.700 | -37.700 |  0.000  |
|    Violating Paths:|   603   |   603   |    0    |
|          All Paths:|  1277   |  1277   |    0    |
+--------------------+---------+---------+---------+

Density: 71.580%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.62 sec
Total Real time: 0.0 sec
Total Memory Usage: 1307.824219 Mbytes
innovus 1>
```

Fig. 16: PRE-CTS-HOLD

PRE CLOCK TREE SYNTHESIS SETUP ANALYSIS

Pre-CTS setup analysis checks if data paths meet setup time requirements before clock tree synthesis, using ideal clock assumptions. This helps identify paths where data may arrive too late relative to the clock edge. Early detection allows designers to optimize placement and buffering to improve timing before building the clock tree.

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 setup

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 91.362  | 91.362  | 98.360  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  1310   |  1277   |   377   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |     14 (14)      |   -0.033   |     14 (14)      |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 71.580%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.28 sec
Total Real time: 0.0 sec
Total Memory Usage: 1333.519531 Mbytes
innovus 1>
```

Fig. 17: PRE-CTS-SETUP

LAYOUT

Pre-CTS ECO (Engineering Change Order) refers to design modifications made before the Clock Tree Synthesis stage in ASIC flow. These changes typically involve logic fixes, functional updates, or RTL corrections. Since clock structures aren't built yet, pre-CTS ECOs are easier to implement and verify compared to post-CTS or post-layout ECOs.

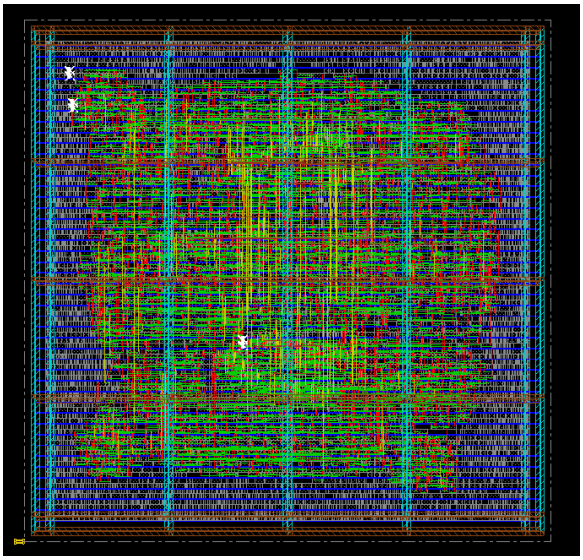


Fig. 18: PRE-CTS-ECO

I-L Clock tree synthesis

Clock Tree Synthesis (CTS) is the process of building a balanced clock distribution network to deliver the clock signal to all sequential elements with minimal skew and delay. CTS replaces ideal clocks with buffers and inverters, ensuring timing integrity, reducing clock skew, and improving setup and hold time closure.

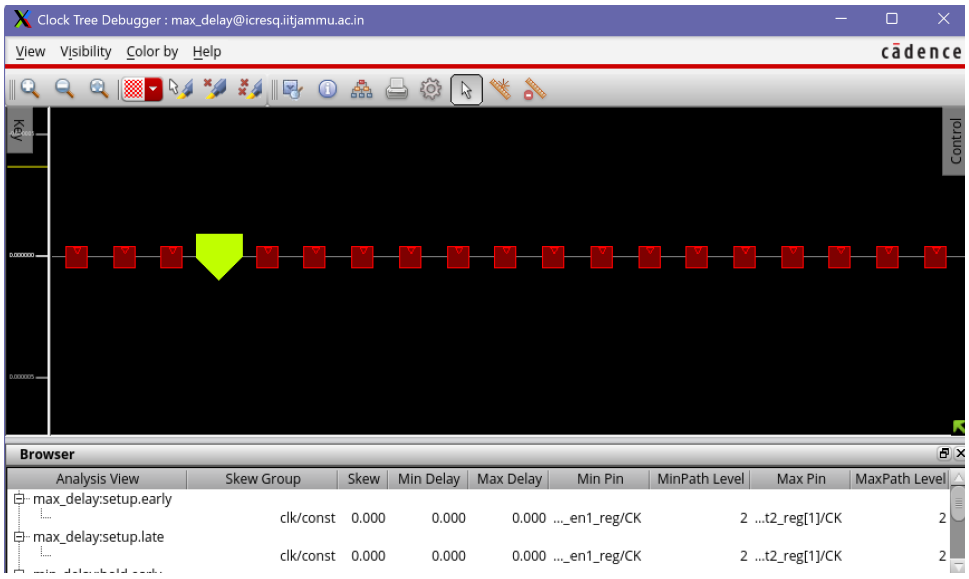


Fig. 19: CLOCK DEBUGGING

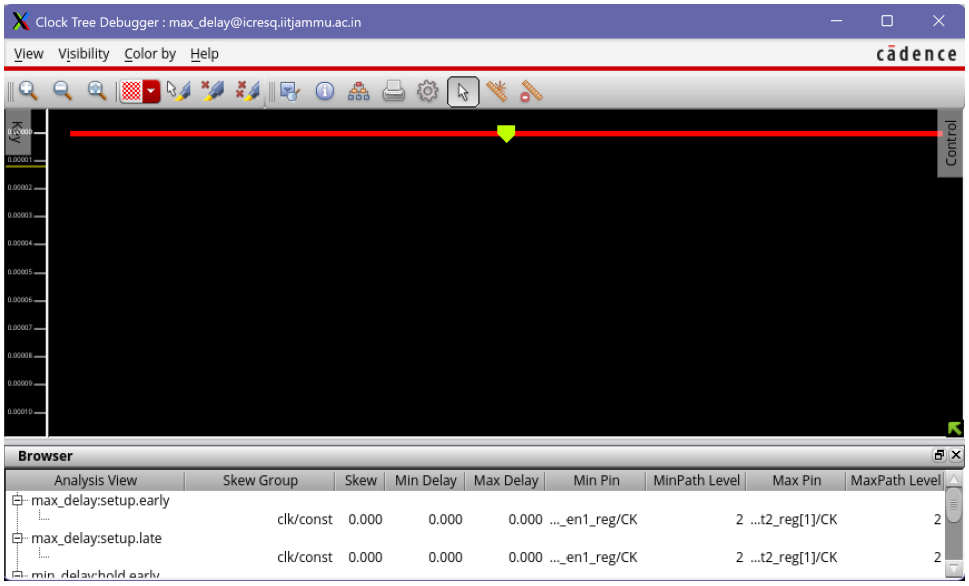


Fig. 20: CLOCK DEBUGGING

```
#############################################################################
# Design Stage: PreRoute
# Design Name: TOP
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#############################################################################
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1388.91)
*** Calculating scaling factor for min_timing libraries using the default operating condition of each library.
**WARN: (IMPESI-3014):  The RC network is incomplete for net rst. As a result, a lumped model will be used during delay calcul
 may compromise timing accuracy. To resolve this, check parasitics for completeness, re-extraction may be required.
Total number of fetched objects 4327
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1471.82 CPU=0:00:00.2 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1471.82 CPU=0:00:00.4 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.4 real=0:00:00.0 totSessionCpu=0:13:19 mem=1471.8M)

------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 hold

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.108  | -0.108  |  0.000  |
|           TNS (ns):| -37.700 | -37.700 |  0.000  |
|    Violating Paths:|   603   |   603   |    0    |
|          All Paths:|  1277   |  1277   |    0    |
+--------------------+---------+---------+---------+

Density: 71.580%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.62 sec
Total Real time: 0.0 sec
Total Memory Usage: 1380.105469 Mbytes
innovus 1>
```

Fig. 21: HOLD ANALYSIS – POST CTS HOLD

```
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1388.93)
*** Calculating scaling factor for max_timing libraries using the default operating condition of each library.
**WARN: (IMPESI-3014):  The RC network is incomplete for net rst. As a result, a lumped model will be used during delay calculation which
 may compromise timing accuracy. To resolve this, check parasitics for completeness, re-extraction may be required.
Total number of fetched objects 4327
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1471.84 CPU=0:00:00.3 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1471.84 CPU=0:00:00.3 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.4 real=0:00:00.0 totSessionCpu=0:13:10 mem=1471.8M)

------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 setup

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 91.362  | 91.362  | 98.360  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  1310   |  1277   |   377   |
+--------------------+---------+---------+---------+

+---------------+-------------------------------+------------------+
|               |             Real              |       Total      |
|    DRVs       +-----------------+-------------+------------------|
|               |  Nr nets(terms) | Worst Vio   |  Nr nets(terms)  |
+---------------+-----------------+-------------+------------------+
|   max_cap     |     14 (14)     |   -0.033    |     14 (14)      |
|   max_tran    |      0 (0)      |   0.000     |      0 (0)       |
|   max_fanout  |      0 (0)      |     0       |      0 (0)       |
|   max_length  |      0 (0)      |     0       |      0 (0)       |
+---------------+-----------------+-------------+------------------+

Density: 71.580%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.65 sec
Total Real time: 0.0 sec
Total Memory Usage: 1405.058594 Mbytes
innovus 1>
```

Fig. 22: SETUP ANALYSIS – POST CTS SETUP

I-M NANOROUTE AND LAYOUT

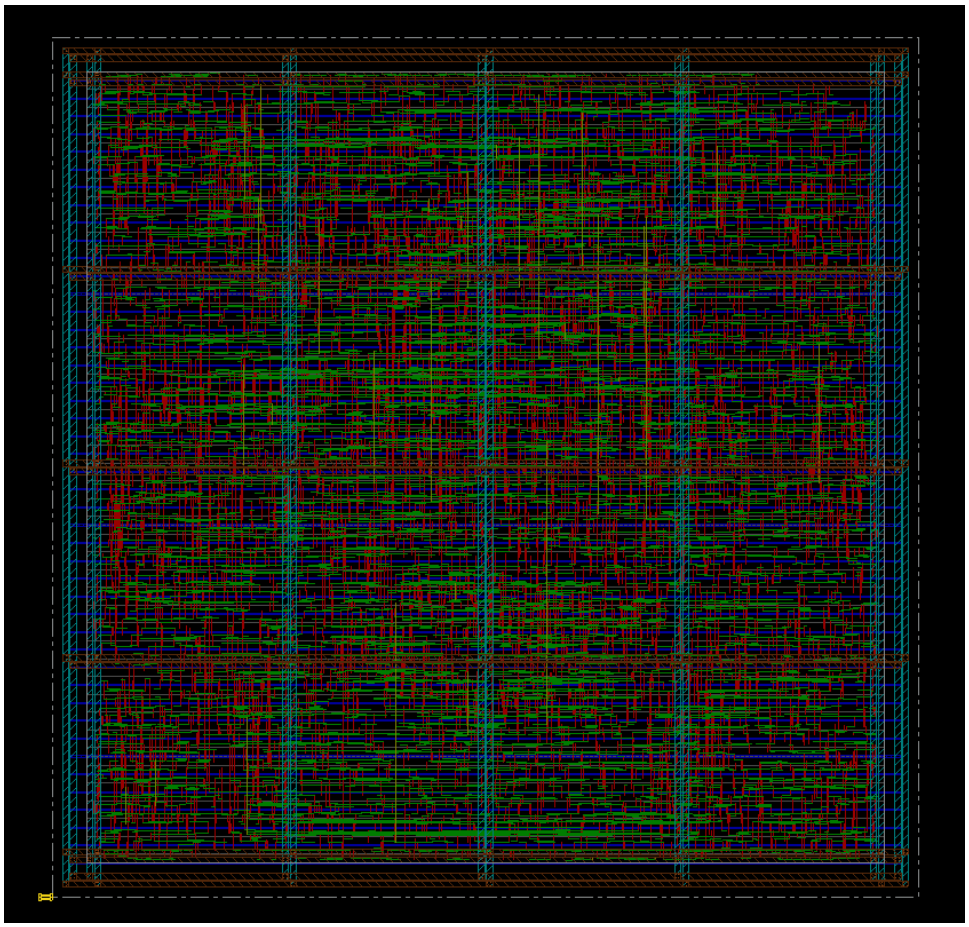


Fig. 23: Nanoroute

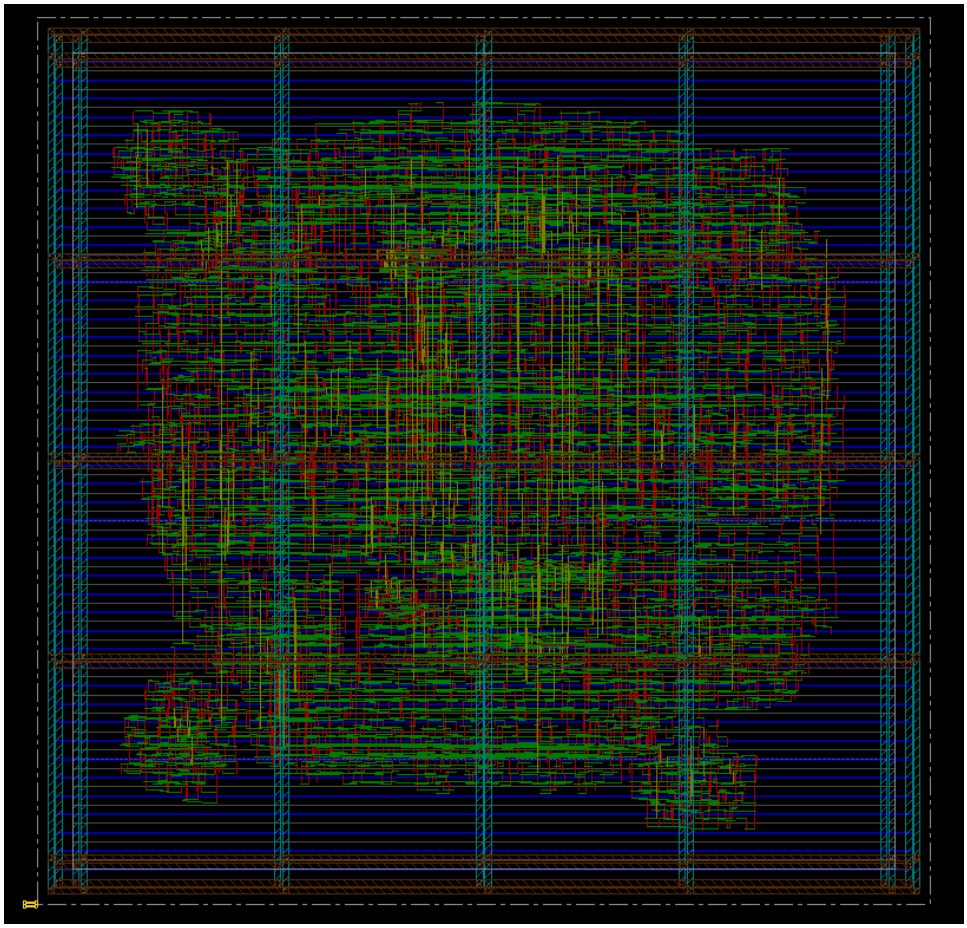
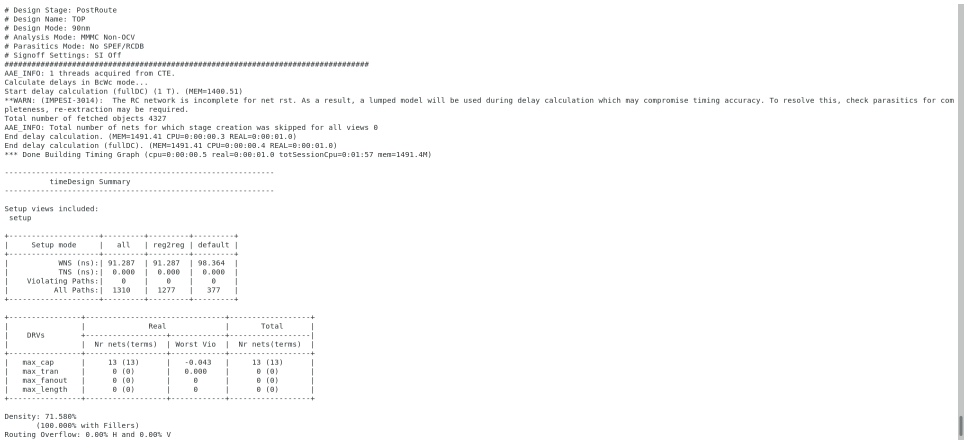


Fig. 24: ECO optimised

I-N Post Synthesis Timing and Simulation

```
#############################################################################
# Design Stage: PostRoute
# Design Name: TOP
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#############################################################################
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1407)
*** Calculating scaling factor for min_timing libraries using the default operating condition of each library.
**WARN: (IMPESI-3014):  The RC network is incomplete for net rst. As a result, a lumped model will be used during delay calculation which may compromise timing accuracy. To resolve this, check parasitics for com
pleteness, re-extraction may be required.
Total number of fetched objects 4327
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1489.91 CPU=0:00:00.3 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1489.91 CPU=0:00:00.4 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.5 real=0:00:00.0 totSessionCpu=0:01:59 mem=1489.9M)

------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 hold

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.108  | -0.108  |  0.000  |
|           TNS (ns):| -37.395 | -37.395 |  0.000  |
|    Violating Paths:|   598   |   598   |    0    |
|          All Paths:|  1277   |  1277   |    0    |
+--------------------+---------+---------+---------+

Density: 71.580%
         (100.000% with Fillers)
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.66 sec
Total Real time: 0.0 sec
Total Memory Usage: 1402.214844 Mbytes
innovus 1> Start to check current routing status for nets...
All nets are already routed correctly.
End to check current routing status for nets (mem=1402.2M)
#############################################################################
```

Fig. 25: Post Route Hold Analysis



```
# Design Stage: PostRoute
# Design Name: TOP
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#################################################################################
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1400.51)
**WARN: (IMPESI-3014):  The RC network is incomplete for net rst. As a result, a lumped model will be used during delay calculation which may compromise timing accuracy. To resolve this, check parasitics for com
pleteness, re-extraction may be required.
Total number of fetched objects 4327
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1491.41 CPU=0:00:00.3 REAL=0:00:01.0)
End delay calculation (fullDC). (MEM=1491.41 CPU=0:00:00.4 REAL=0:00:01.0)
*** Done Building Timing Graph (cpu=0:00:00.5 real=0:00:01.0 totSessionCpu=0:01:57 mem=1491.4M)

------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

Setup views included:
 setup

+--------------------+--------+---------+---------+
|     Setup mode     |   all  | reg2reg | default |
+--------------------+--------+---------+---------+
|          WNS (ns):| 91.287 | 91.287  | 98.364  |
|          TNS (ns):|  0.000 |  0.000  |  0.000  |
|    Violating Paths:|    0   |    0    |    0    |
|         All Paths:|  1310  |  1277   |   377   |
+--------------------+--------+---------+---------+

+----------------+-----------------------------+------------------+
|                |             Real            |       Total      |
|    DRVs        +------------------+----------+------------------|
|                |  Nr nets(terms)  | Worst Vio |  Nr nets(terms)  |
+----------------+------------------+-----------+------------------+
|   max_cap      |      13 (13)     |   -0.043  |     13 (13)      |
|   max_tran     |       0 (0)      |   0.000   |      0 (0)       |
|   max_fanout   |       0 (0)      |     0     |      0 (0)       |
|   max_length   |       0 (0)      |     0     |      0 (0)       |
+----------------+------------------+-----------+------------------+

Density: 71.580%
        (100.000% with Fillers)
Routing Overflow: 0.00% H and 0.00% V
```

Fig. 26: Post Route Setup Analysis

Once the design is completed, and being optimised using ECO, netlist is generated, and that same netlist is tested by NClaunch using same testbench

Fig. 27: Post Synthesis Output Waveform

I-O Post Layout and Verification and Extraction

DRC Viloations

DRC (Design Rule Check) violation in Cadence Innovus indicates that some part of your design does not comply with the foundry-defined physical layout rules.

- **Spacing violations:** Insufficient distance between metal layers or features.
- **Width violations:** Metal lines narrower than the minimum allowed.
- **Enclosure violations:** Vias not properly enclosed by surrounding metal.
- **Antenna violations:** Long unconnected wires that can damage transistors.
- **Via issues:** Missing or improperly stacked vias.

```
MAR: ( Minimum Area ) Regular Wire of Net s2p_out2[4]  ( Metal2 )
Bounds : ( 86.780, 126.165 ) ( 86.930, 126.425 )


OFFGRID: ( Off Grid or Wrong Way ) Regular Wire of Net fftp/stage1_3/sub_270_21/n_14  ( Metal2 )
Bounds : ( 50.903, 125.875 ) ( 51.042, 127.005 )


MAR: ( Minimum Area ) Regular Wire of Net s2p/n_85  ( Metal2 )
Bounds : ( 44.730, 126.165 ) ( 44.880, 126.425 )


MAR: ( Minimum Area ) Regular Wire of Net s2p_out5[0]  ( Metal2 )
Bounds : ( 47.050, 126.165 ) ( 47.200, 126.425 )


MAR: ( Minimum Area ) Regular Wire of Net s2p_out5[0]  ( Metal2 )
Bounds : ( 51.690, 126.165 ) ( 51.840, 126.425 )


 Total Violations : 1000 Viols.
```

Fig. 28: DRC Violation

PowerVIA

This report indicates that the verifyPowerVia command in Cadence Innovus was run successfully for the design TOP, and no issues or warnings were found regarding the power via connections.

verifyPowerVia checks the connectivity and integrity of vias used to connect power nets (like VDD, VSS) across metal layers.

```
#################################################################
#  Generated by:      Cadence Innovus 17.12-s095_1
#  OS:                Linux x86_64(Host ID icresq.iitjammu.ac.in)
#  Generated on:      Thu May  1 07:54:54 2025
#  Design:            TOP
#  Command:           verifyPowerVia
#################################################################
Verify Power Via report created on Thu May  1 07:54:54 2025


Begin Summary
    Found no problems or warnings.
End Summary
```

Fig. 29: Verification of Power through via

Standard Delay Format (SDF)

Standard Delay Format (SDF) is an IEEE standard textual format used to represent timing information for digital designs. It is widely used for back-annotating delays into gate-level simulations after synthesis or place-and-route. SDF provides data such as:

- Cell delays and interconnect delays
- Timing checks (e.g., setup, hold, recovery)
- Path-specific delays
- Conditional timing (e.g., based on clock edges)

```
(DELAYFILE
  (SDFVERSION "3.0")
  (DESIGN "TOP")
  (DATE "Thu May  1 08:04:05 2025")
  (VENDOR "Cadence Design Systems, Inc.")
  (PROGRAM "Innovus")
  (VERSION "v17.12-s095_1 ((64bit) 11/09/2017 12:10 (Linux 2.6.18-194.el5))")
  (DIVIDER /)
  (VOLTAGE 0.900000::0.900000)
  (PROCESS "1.000000::1.000000")
  (TEMPERATURE 125.000000::125.000000)
  (TIMESCALE 1.0 ns)


  (CELL
    (CELLTYPE  "MXI2XL")
    (INSTANCE  fftp/cmx2/csa_tree_minus_318_17_pad_groupi/g1869)
      (DELAY
    (ABSOLUTE
    (IOPATH A Y  (0.196::0.196) (0.164::0.164))
    (IOPATH B Y  (0.174::0.174) (0.141::0.141))
    (IOPATH (posedge S0) Y  (0.202::0.202) (0.124::0.124))
    (IOPATH (negedge S0) Y  (0.147::0.147) (0.199::0.199))
    )
      )
  )

```

Fig. 30: Delay of an instance or cell

DELAY Block (ABSOLUTE):

This lists pin-to-pin delays under specific transition conditions:

From	To	Rise Delay (ns)	Fall Delay (ns)
`A`	`Y`	0.196	0.164
`B`	`Y`	0.174	0.141
`posedge S0`	`Y`	0.202	0.124
`negedge S0`	`Y`	0.147	0.199

Fig. 31: Delay

`SPEF (Standard Parasitic Exchange Format) file`

A SPEF (Standard Parasitic Exchange Format) file is used in VLSI design flows to describe the parasitic resistance and capacitance of interconnects in a digital circuit. These parasitics are extracted from the layout after the place and route (P//R) stage and are critical for accurate timing analysis, signal integrity, and power estimation.

```
23635  *D_NET *37 0.00300378
23636
23637  *CONN
23638  *I *5311:A I *C 213 105 *L 0.00668 *D ADDFXL
23639  *I *5330:A I *C 213 103 *L 0.00668 *D ADDFXLs
23640  *I *7789:Q O *C 212 115 *L 0 *D DFFQX1
23641  *I *8151:B I *C 214 114 *L 0.00164 *D MXI2XL
23642  *I *8077:A I *C 213 118 *L 0.0017 *D MXI2XL
23643
23644  *CAP
23645  1 *37:1 1.06566e-05
23646  2 *37:2 2.01714e-05
23647  3 *37:3 9.5148e-06
23648  4 *37:4 2.16714e-05
23649  5 *37:5 2.16714e-05
23650  6 *37:6 0.00084007
23651  7 *37:7 0.00066223
23652  8 *37:8 0.000187596
23653  9 *37:9 0.000173371
23654  10 *37:10 9.75654e-06
23655  11 *37:11 0.0001517
23656  12 *37:12 2.16714e-05
23657  13 *37:14 1.82684e-05
23658  14 *37:15 0.000110372
23659  15 *37:16 1.8649e-05
23660  16 *37:17 0.000242818
23661  17 *37:18 3.80592e-07
23662  18 *37:19 0.000132446
23663  19 *37:22 2.16714e-05
23664  20 *37:23 2.16714e-05
23665  21 *37:24 0.000110372
23666  22 *37:25 0.000110372
23667  23 *37:26 4.33427e-05
23668  24 *37:27 4.33427e-05
23669
23670  *RES
23671  1 *37:28 *8077:A 1.4
23672  2 *37:27 *37:28 1.4
23673  3 *37:26 *37:27 0.248571
23674  4 *37:25 *37:26 1.4
23675  5 *37:24 *37:25 0.621429
23676  6 *37:23 *37:24 1.4
23677  7 *37:22 *37:23 0.124286
23678  8 *37:19 *37:22 1.4
23679  9 *37:18 *8151:B 1.4
```

Fig. 32: Capacitance and Resistance Details of Interconnect

Total capacitance of D NET = 0.00300378 pF

- `*CONN`

This section lists **connections** to the net. Each line has the format:

```php-template
*I <pin> <direction> *C <x> <y> *L <load> *D <celltype>
```

For example:

```mathematica
*I *5311:A I *C 213 105 *L 0.00668 *D ADDFXL
```

Fig. 33: How to Read file

- `*CAP`

This section lists all **lumped capacitance values** at various nodes in the net. Format:

```php-template
<index> <node> <capacitance>
```

Example:

```
1 *37:1 1.06566e-05
```

Fig. 34: Capacitance Interconnect

- `*RES`

This is the **RC tree**—resistances between nodes. Format:

```php-template
<index> <node1> <node2> <resistance>
```

Example:

```less
1 *37:28 *8077:A 1.4
```

Fig. 35: Resistance Details of Interconnect

SPF (standard parsitic format) file

This snippet is from a DSPF (Detailed Standard Parasitic Format) file generated by Innovus, used for RC extraction—an important step in physical design verification where parasitic resistance and capacitance values are extracted for accurate timing and signal integrity analysis.

```
* RC extraction data for design TOP
*
*
*
*|DSPF 1.0
*
*|VENDOR "Cadence Design Systems, Inc."
*|PROGRAM "FirstEncounter System"
*|DIVIDER /
*|DELIMITER :
*|BUSBIT []
*
.SUBCKT TOP
+ clk start rst real_data_o[15] real_data_o[14] real_data_o[13] real_data_o[12] real_data_o[11] real_data_o[10] real_data_o[9]
+ real_data_o[8] real_data_o[7] real_data_o[6] real_data_o[5] real_data_o[4] real_data_o[3] real_data_o[2] real_data_o[1] real_data_o[0] img_data_o[15]
+ img_data_o[14] img_data_o[13] img_data_o[12] img_data_o[11] img_data_o[10] img_data_o[9] img_data_o[8] img_data_o[7] img_data_o[6] img_data_o[5]
+ img_data_o[4] img_data_o[3] img_data_o[2] img_data_o[1] img_data_o[0]

* Net Section
*
*
*|GROUND_NET 0
*
*|NET clk 0.833488PF
*|P (clk I 0.000000PF 0.000 0.000)
*|I (s2p/shift_reg_reg\[7\]\[10\]:CK s2p/shift_reg_reg\[7\]\[10\] CK I
*+ 0.001744PF 171.535 134.415)
*|I (s2p/shift_reg_reg\[3\]\[10\]:CK s2p/shift_reg_reg\[3\]\[10\] CK I
*+ 0.001744PF 170.955 131.805)
*|I (fftp/stage1_3/out1_reg\[9\]:CK fftp/stage1_3/out1_reg\[9\] CK I
*+ 0.001744PF 164.865 131.805)
*|I (fftp/stage1_4/out1_reg\[9\]:CK fftp/stage1_4/out1_reg\[9\] CK I
```

Fig. 36: C Extraction

NET CLK total Capacitance 0.833488 pf shown in the image above. It is showing total capacitance of entire clk net.

```
R1_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_12
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1814:CO 1.400000
R2_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_11
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1806:B 1.400000
R3_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_9
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_12 1.400000
R4_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_8
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_11 0.124286
R5_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_7
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1801:A1 1.400000
R6_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_6
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_9 4.101429
R7_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_6
```

Fig. 37: R Extraction

```
R1_2197 fftp/cmx2/csa_tree_minus_319_17_pad_groupi/n_157_12
+ fftp/cmx2/csa_tree_minus_319_17_pad_groupi/g1814:CO 1.400000
```

Represents resistor between two node is 1.4 ohms

Difference Between SPEF And SPF SPEF is a standardised version of IEEE 1481, whereas SPF is a general version that contains less information. SPEF can be used by Multiple tools and SPF only by cadence.

I-P Layout vs Schematic

LVS can be performed using built-in tool in Innovus called PVS(physical verification system), or Calibre(Siemens EDA) tool can be used.
TO perform LVS we required:
1. Netlist created by genus(.v or .spice
2. Netlist in .v or spice format created by Innovus(PD)
3. .rul file provided by the boundary
4. .lib file from the foundary

II Systolic Matrix Multiplier

II-A RTL Design

Register Transfer Level (RTL) is a key abstraction in digital circuit design that outlines how data moves between registers and the logic operations applied to it. It serves as the basis for hardware description languages such as Verilog and VHDL. RTL design provides detailed control over hardware functionality and plays a vital role in converting designs into gate-level implementations, making it an essential phase in developing digital systems and ASICs.

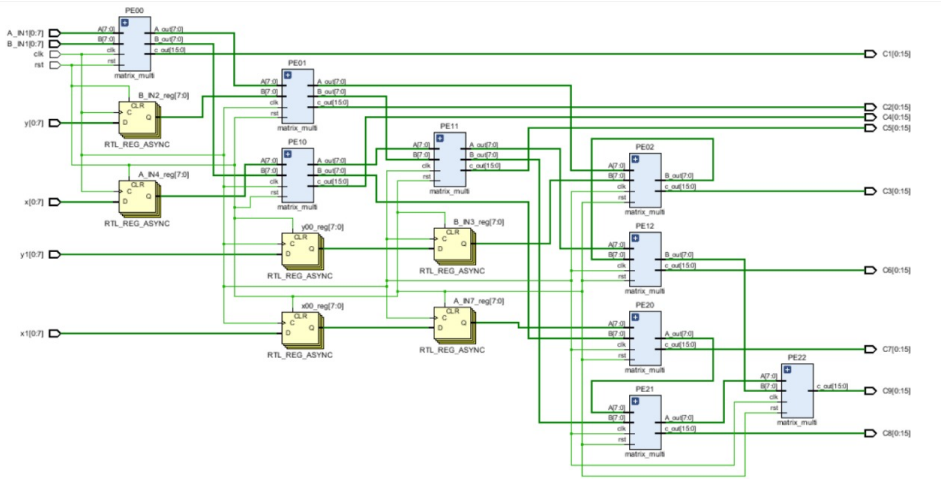


Fig. 38: RTL DESIGN

II-B Verilog code

```
    timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 23.04.2025 21:18:51
// Design Name:
// Module Name: systolic_array_3x3
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module systolic_array_3x3(
input wire clk,
    input wire rst,
    input wire [0:7] x,x1,y,y1,
    input wire [0:7] A_IN1,
    input wire [0:7] B_IN1,
    output wire [0:15] C1,C2,C3,C4,C5,C6,C7,C8,C9
);
wire [0:7] A_OT1,A_OT2,B_OT1,A_OT4,B_OT2,A_OT5,B_OT3,B_OT4,A_OT7,B_OT5,A_OT8,B_OT6;
```

```
reg [0:7] A_IN4,B_IN2,B_IN3;
reg [0:7] A_IN7,x00,y00;
always @(posedge clk or posedge rst) begin
if (rst) begin
            A_IN4 <= 7'b0;
            A_IN7 <= 7'b0;
            x00 <= 7'b0;
            B_IN2 <= 7'b0;
            B_IN3 <= 7'b0;
            y00 <= 7'b0;

        end else begin
            A_IN4 <= x;
             A_IN7 <= x00;
             x00 <= x1;
             B_IN2 <= y;
             B_IN3 <= y00;
             y00 <= y1;
end
end
               matrix_multi PE00 (
                   .clk(clk),
                   .rst(rst),
                   .A(A_IN1),
                   .B(B_IN1),
                   .A_out(A_OT1),
                   .B_out(B_OT1),
                   .c_out(C1)
               );

               matrix_multi PE01 (
                    .clk(clk),
                    .rst(rst),
                    .A(A_OT1),
                    .B(B_IN2),
                    .A_out(A_OT2),
                    .B_out(B_OT2),
                    .c_out(C2)
                );



              matrix_multi PE02 (
                       .clk(clk),
                       .rst(rst),
                       .A(A_OT2),
                       .B(B_IN3),
                       .B_out(B_OT3),
                       .c_out(C3)
                   );



           matrix_multi PE10 (
                       .clk(clk),
                       .rst(rst),
                       .A(A_IN4),
                       .B(B_OT1),
```

```
            .A_out(A_OT4),
            .B_out(B_OT4),
            .c_out(C4)
        );


matrix_multi PE11 (
            .clk(clk),
            .rst(rst),
            .A(A_OT4),
            .B(B_OT2),
            .A_out(A_OT5),
            .B_out(B_OT5),
            .c_out(C5)
         );



matrix_multi PE12 (
            .clk(clk),
            .rst(rst),
            .A(A_OT5),
            .B(B_OT3),
            .B_out(B_OT6),
            .c_out(C6)
         );



matrix_multi PE20 (
            .clk(clk),
            .rst(rst),
            .A(A_IN7),
            .B(B_OT4),
            .A_out(A_OT7),
            .c_out(C7)
        );


matrix_multi PE21 (
            .clk(clk),
            .rst(rst),
            .A(A_OT7),
            .B(B_OT5),
            .A_out(A_OT8),
            .c_out(C8)
        );


 matrix_multi PE22 (
            .clk(clk),
            .rst(rst),
            .A(A_OT8),
```

```
                    .B(B_OT6),
                    .c_out(C9)
                );

        endmodule
```

II-C Pre-Synthesis Simulation

Pre-synthesis simulation involves validating the RTL design, written in Verilog or VHDL, before it is synthesized into gate-level hardware. Using a testbench and simulation tools, this stage checks the functional accuracy of the design. It helps catch logical issues early in the development cycle, making it easier to debug and refine the design. Because the simulation runs on the higher-level HDL representation, it is generally faster and easier to interpret than gate-level or post-synthesis simulations.
Testbench

```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 23.04.2025 21:20:54
// Design Name:
// Module Name: systolic_array_3x3_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module systolic_array_3x3_tb;
reg clk;
reg rst;
reg [0:7] x, x1, y, y1;
reg [0:7] A_IN1, B_IN1;
wire [0:15] C1, C2, C3, C4, C5, C6, C7, C8, C9;

systolic_array_3x3 uut (
    .clk(clk),
    .rst(rst),
    .x(x),
    .x1(x1),
    .y(y),
    .y1(y1),
    .A_IN1(A_IN1),
    .B_IN1(B_IN1),
    .C1(C1),
    .C2(C2),
    .C3(C3),
    .C4(C4),
    .C5(C5),
    .C6(C6),
    .C7(C7),
```

```
    .C8(C8),
    .C9(C9)
);

always #5 clk = ~clk;

initial begin
    clk = 0;
    rst = 1;
    x = 8'd0;
    x1 = 8'd0;
    y = 8'd0;
    y1 = 8'd0;
    A_IN1 = 8'd0;
    B_IN1 = 8'd0;
    #10;
    rst = 0;
    A_IN1 = 8'd1;
    B_IN1 = 8'd1;
    x = 8'd1;
    x1 = 8'd1;
    y = 8'd1;
    y1 = 8'd1;
    #10
     A_IN1 = 8'd1;
     B_IN1 = 8'd1;
     x = 8'd1;
     x1 = 8'd1;
     y = 8'd1;
     y1 = 8'd1;

    #10

    A_IN1 = 8'd1;
    B_IN1 = 8'd1;
    x = 8'd1;
    x1 = 8'd1;
    y = 8'd1;
    y1 = 8'd1;
      #10;

      A_IN1 = 8'd0;
      B_IN1 = 8'd0;
      x = 8'd0;
      x1 = 8'd1;
      y = 8'd0;
      y1 = 8'd1;
      #40
       A_IN1 = 8'd0;
       B_IN1 = 8'd0;
       x = 8'd0;
       x1 = 8'd0;
       y = 8'd0;
       y1 = 8'd0;

    $stop;
end
```

```
endmodule
```

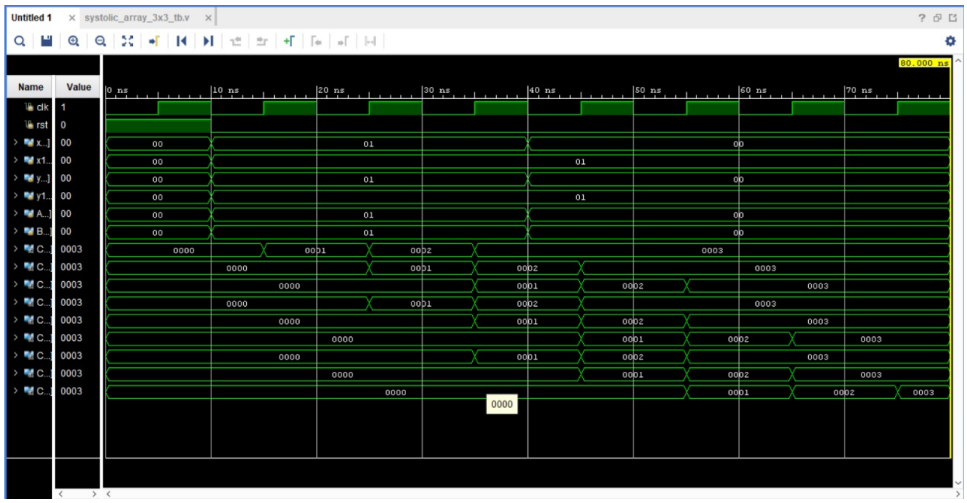


Fig. 39: OUTPUT WAVEFORM

II-D Synthesis

Synthesis is a crucial phase in the digital design flow where RTL code (written in Verilog or VHDL) is translated into a gate-level representation using Electronic Design Automation (EDA) tools such as **Cadence Genus Synthesis Solution**. Genus maps the high-level functional design to a network of technology-specific logic gates and flip-flops, optimizing it for key parameters like area, power, and timing.

Cadence Genus is widely used in ASIC and SoC design flows due to its fast and scalable synthesis engine, support for hierarchical and physical-aware synthesis, and advanced optimization capabilities. It also supports incremental synthesis, allowing efficient updates to large designs without full re-synthesis.

Input Files for Cadence Genus

To perform synthesis using Genus, the following input files are required:

RTL Design Files (.v or .vhdl): The Verilog or VHDL files describing the design at the Register Transfer Level.
Standard Cell Library Files (.db): Technology-specific cell library files used for mapping RTL to physical gates.
Design Constraints File (.sdc): Specifies timing, clock definitions, input/output delays, area constraints, and other design rules.
TCL Script (.tcl): A script that automates the synthesis flow, including reading files, setting constraints, and generating reports.

Output Files Generated by Genus

After successful synthesis, Genus generates several important output files used in downstream design steps:

***Gate-Level Netlist (.v)*:** The synthesized design represented using standard cells from the technology library.
Updated constraint file (.sdc): May include updated timing constraints reflecting synthesis results.
Area Report(area.rpt): A detailed summary of the area used by each module and the entire design.
Timing Report(timing.rpt): Provides information about timing paths, setup and hold violations, and clock frequencies.
Power Report(power.rpt): Estimates dynamic and static power consumption of the synthesized design.
Log File(.log): Contains a complete log of the synthesis process, warnings, and error messages for debugging.
Mapping Report(mapping.rpt): Shows how RTL components were mapped to standard cells.
Utilization Report (utilization.rpt): Summarizes resource utilization, including the number of gates, flip-flops, and other components used.

These output files are essential for verifying functional correctness (via gate-level simulation), performing static timing analysis, and moving the design into the physical implementation phase (place-and-route).

Timing report

```
  ============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 24 2025  03:44:21 pm
  Module:                 systolic_array_3x3
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (8746 ps) Setup Check with Pin x00_reg[4]/CK->D
          Group: clk
     Startpoint: (R) x1[4]
          Clock: (R) clk
       Endpoint: (R) x00_reg[4]/D
          Clock: (R) clk

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

             Setup:-     204
       Uncertainty:-      50
     Required Time:=    9746
      Launch Clock:-       0
       Input Delay:-    1000
         Data Path:-       0
             Slack:=    8746

Exceptions/Constraints:
  input_delay             1000            in_del_12_1

#-------------------------------------------------------------------------------------------
# Timing Point   Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival Instance
#                                                    (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------
  x1[4]          -       -     R     (arrival)      1  2.3     0     0    1000    (-,-)
  x00_reg[4]/D   -       -     R     DFFRHQX1       1    -     -     0    1000    (-,-)
#-------------------------------------------------------------------------------------------
```

POWER REPORT

```
    ====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 24 2025  03:44:21 pm
  Module:                 systolic_array_3x3
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

                          Leakage    Dynamic     Total
     Instance      Cells Power(nW)  Power(nW)  Power(nW)
--------------------------------------------------------
systolic_array_3x3    49  4830.453 122571.643 127402.096
  PE00                 0     0.000   6440.512   6440.512
  PE01                 0     0.000   6440.512   6440.512
  PE02                 0     0.000   6440.512   6440.512
  PE10                 0     0.000   6440.512   6440.512
  PE11                 0     0.000   6440.512   6440.512
  PE12                 0     0.000   6440.512   6440.512
  PE20                 0     0.000   6440.512   6440.512
  PE21                 0     0.000   6440.512   6440.512
  PE22                 0     0.000   6440.512   6440.512
```

AREA REPORT

```
    ====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 24 2025  03:44:21 pm
  Module:                 systolic_array_3x3
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================


  Gate    Instances   Area   Library
-------------------------------------
DFFRHQX1         48  980.942    slow
INVXL             1    2.271    slow
-------------------------------------
total            49  983.213


     Type      Instances   Area  Area %
----------------------------------------
sequential            48 980.942   99.8
inverter               1   2.271    0.2
unresolved             9   0.000    0.0
physical_cells         0   0.000    0.0
----------------------------------------
total                 58 983.213  100.0
```

On running command ”gui show” opens a Graphical User Interface (GUI) window that can display various design views. One of the key things you can view there is the standard cell library used during synthesis.
Standard cells are the basic building blocks of digital circuits in ASIC (Application-Specific Integrated Circuit) design. They are pre-designed, pre-characterized logic gates and other components that are used to implement digital logic.

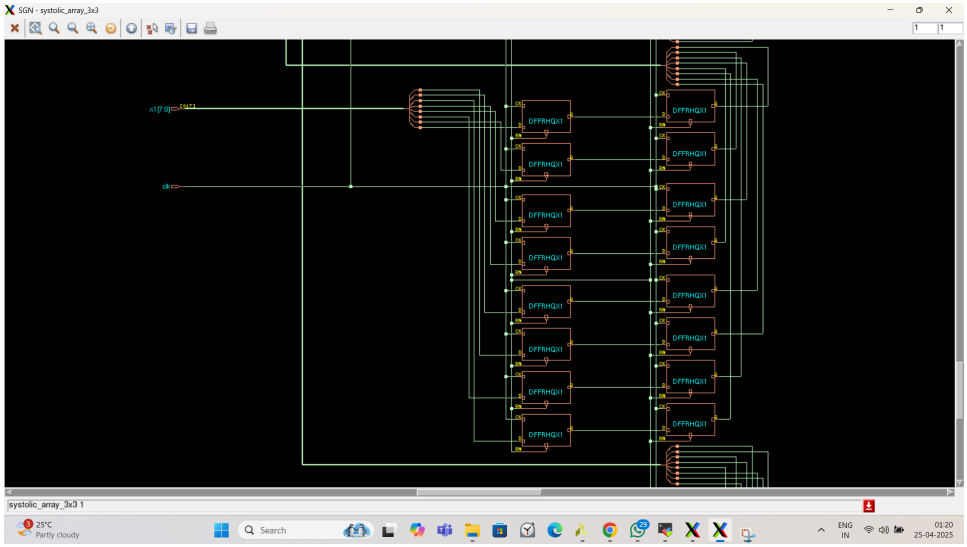


Fig. 40: GENUS STANDARD CELLS

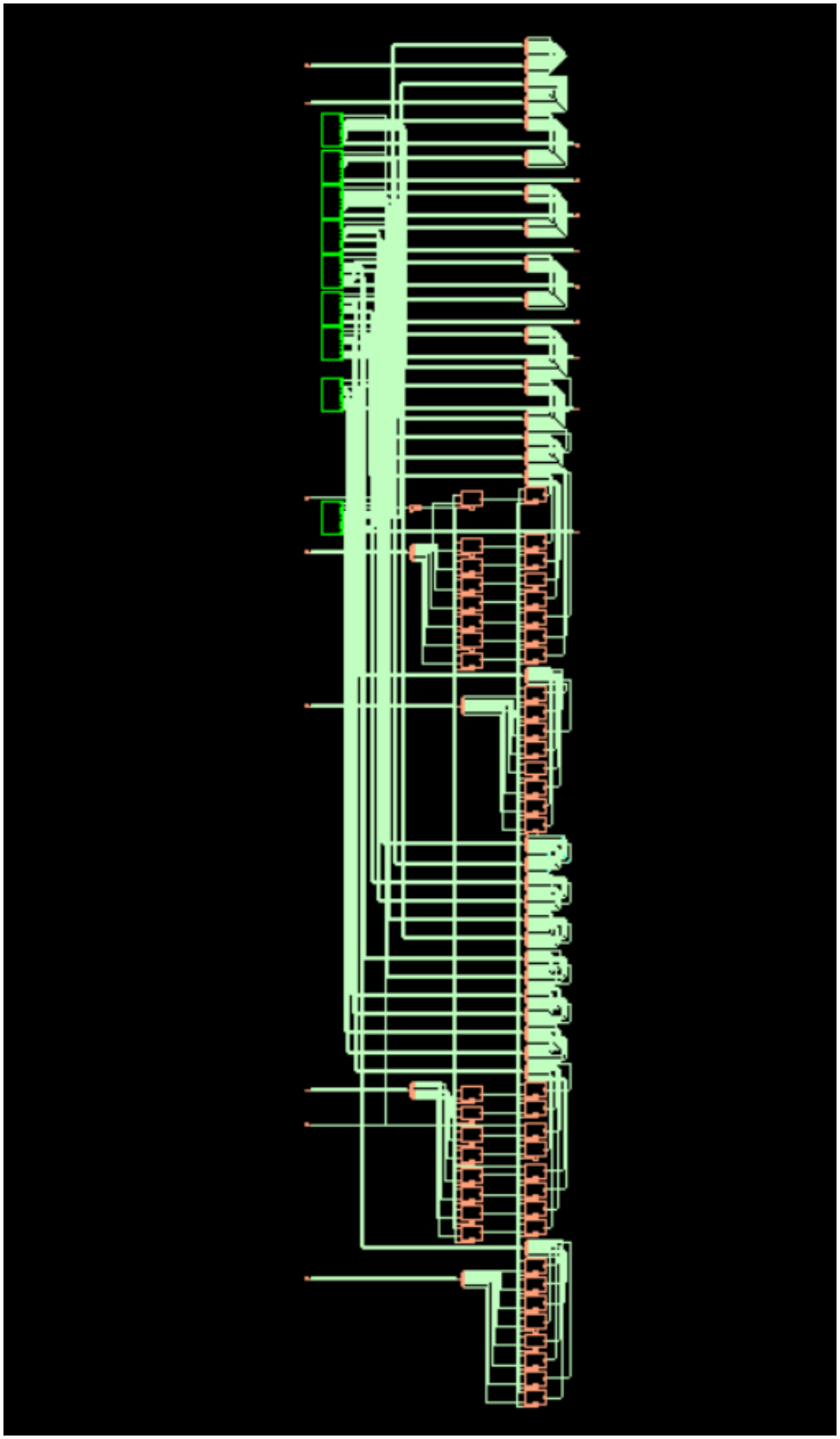


Fig. 41: GENUS STANDARD CELLS

II-E Post-Synthesis

Post-synthesis refers to the stage following the synthesis of a digital design, where the RTL code has been transformed into a gate-level netlist. This netlist is now mapped to specific technology libraries and optimized for area, power, and timing. Post-synthesis analysis involves verifying that the synthesized design meets the required timing and functionality through simulations and timing reports. Tools are used to perform static timing analysis (STA), power estimation, and logic equivalence checking. This step ensures that the design is ready for physical implementation, such as placement and routing. Any violations detected here must be addressed before proceeding further.

STEPS TO NCLAUNCH

Step 1: In the Terminal write the nclaunch to open the tool.
Step 2: Copy the file in the download folder of slow.v from the directory saved in vlog folder.
Step 3: Send the slow.v, synthesis after netlist.v and the testbench .v file in the compiler hdl and the result store in the Worklib file.
Step 4: Open the workLib file and select the module name file in .v format same for testbench .v format and add the elaborater on each .v file.
Step 5: Open the snapshots, select the generated testbench file and open the gui graphics.

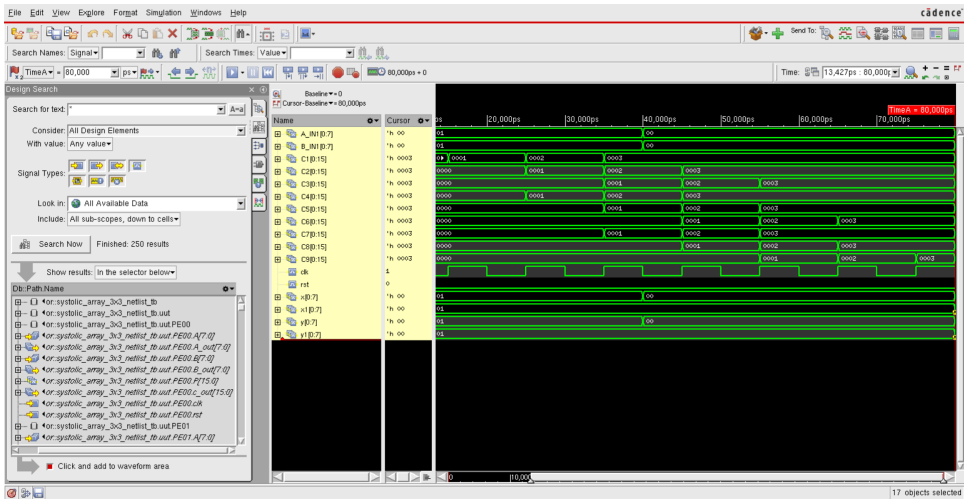


Fig. 42: POST SYNTHESIS WAVEFORM(netlist tb)

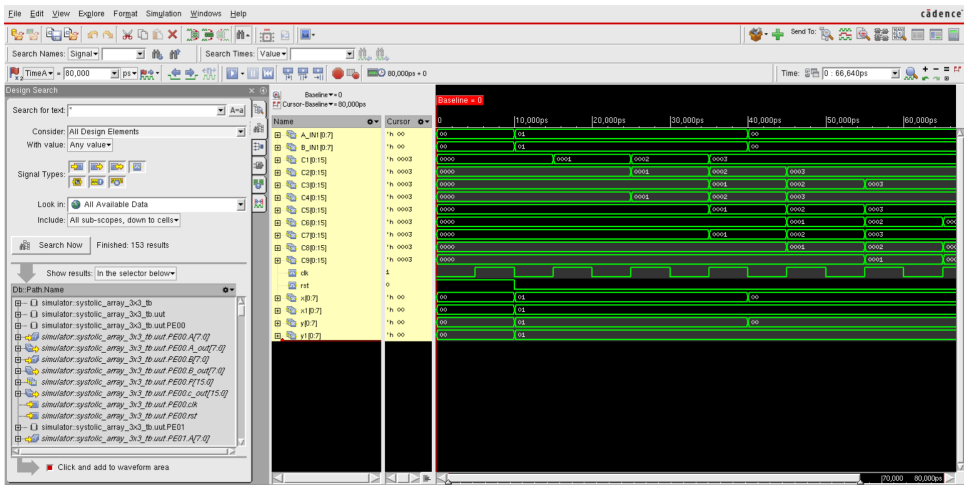


Fig. 43: POST SYNTHESIS WAVEFORM(SIMPLE tb

III PHYSICAL DESIGN

Physical design is a vital stage in the ASIC (Application-Specific Integrated Circuit) design flow, where the synthesized netlist—representing the logical behavior of the circuit—is transformed into a physical layout suitable for silicon fabrication. This step bridges the gap between digital logic design and actual hardware implementation.

Tools Commonly Used: Cadence Innovus, Synopsys IC Compiler II, Mentor Graphics Caliber (for DRC / LLS).

Importance of Physical Design: Directly influences chip performance, power consumption, area efficiency, and yield.

Plays a crucial role in meeting stringent specifications in advanced technology nodes (e.g., 5nm, 3nm).

Requires coordination across multiple domains: digital logic, analog IPs, clocking, power integrity, and signal integrity.

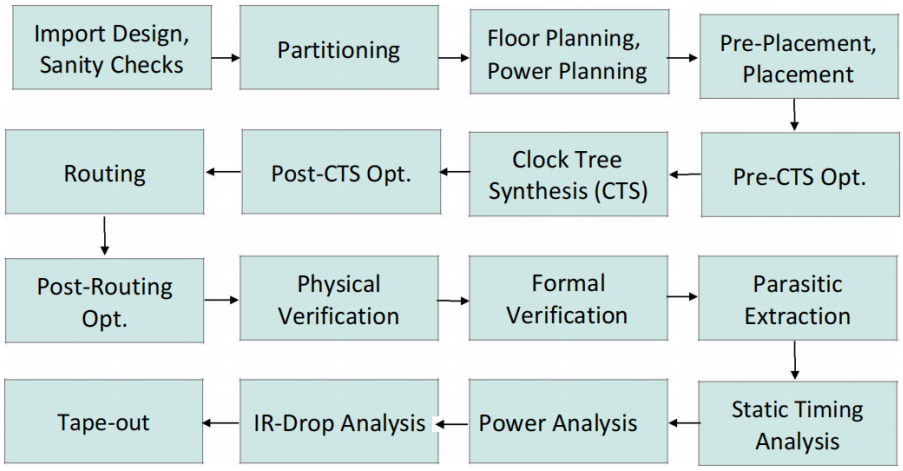


Fig. 44: PHYSICAL DESIGN FLOW

Step 1 : In the terminal type the command innovus to open the tool,then a new window of cadence open.
Step 2 : Go to the file, open the design, now fill all the essential files ,LEF files , netkist after synthesis.v files, and in MMMC add the max Timing with slow .lib and min timing with fast.lib, max delay with max timing and vice versa,add the setup and hold condition,in last all the netlist and design floor occur on the screen. The format will be of .view and .global.
Step 3 : Open the specify floorplan fill the necessary information like core utilization and add the aspect ratio and core to IO boundary to 10,click ok.
Step 4 :Do the power planning add the strips, rings with appropriate metals selection both horizontal and vertical with high metals.
Step 5 : Add the nets,by clicking on route and then special route and click on all metal layers.
Step 6: Go to place,physical cell, add encp. select the filler and add later the well tap instance with appropriate setting to rows to get the design.
Step 7 : Again Go to place,std. cell, then place io pins and click on ok, all the std. cell come into the die are.
Step 8 : In digital analysis, Go to report timing , preCTS to setup all information occur on the screen, then same will done for the hold .
Step 9: Go to the Clock Debugger click on CTS and apply ,path will be generated and save the design with .enc format.
Step 10:Go to Route ,Nano Route ,route add the timing driven add congestion click on terminal to get the result.
Step 11 : Go to placement add the physical cell add the filler(Select all filters) the fillers.

III-A IMPORTED DESIGN

The following input files information are loaded to the tool:
Netlist (.v)
Physical Libraries (.lef)
Timing Libraries (.lib)
Technology Files
Constraints (.sdc)
The core area is approximately calculated by the tool from the Netlist. While Importing, we first have to load the lef files and then the lib files.

Netlist consists of: Ports of Standard Cells and Macros and Interconnection details.

Synopsys Design Constraints: (SDC)
—Timing Constraints : Clock Definition (Time Period, Duty Cycle) , Timing Exceptions (False Paths, Asynchronous Paths) Non-Timing Constraints: Operating conditions , Wire load models , System interface, Design rule constraints (DRVs - Max. Cap./ Transition/Fanout) Area constraint, Multi-voltage and Power optimization constraints
Logic assignments:
Liberty Timing File contains:
Cell Type and Functionality , Delay Models (WLD/ NLDM/ CCS) , Pin/ Cell Timings and design rules ,PVT Conditions , Power Details (Leakage and Dynamic)
Library Exchange Format (LEF) contains:Cell Name, Shape, Size, Orientation Class , Port/Pin Name, Direction and Layout Geometries , Obstruction/ Blockages ,Antenna Diff. Area

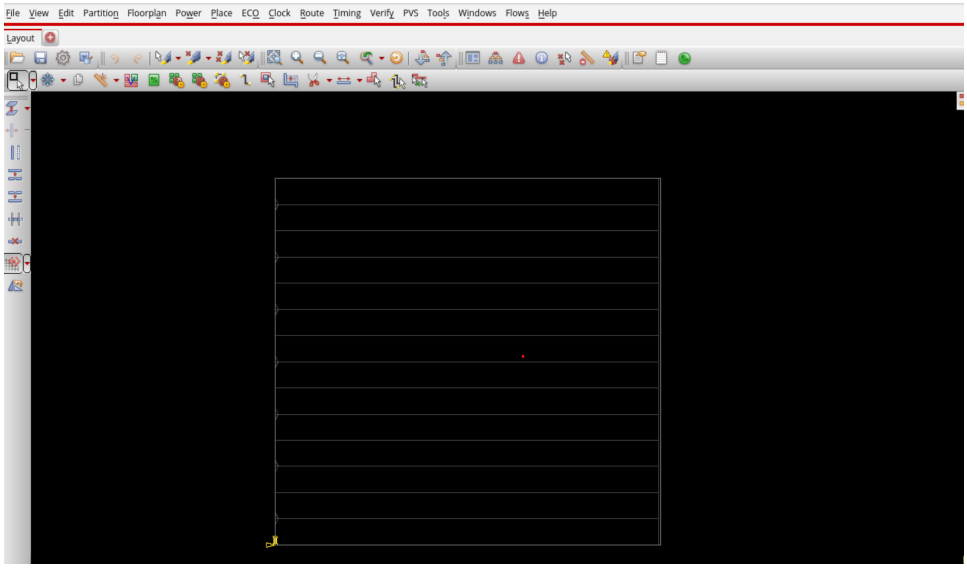


Fig. 45: imported design

III-B FLOOR PLAN

Initialize with Chip and Core Aspect Ratio (AR)
— Initialize with Core Utilization
— Initialize Row Configuration and Cell Orientation:Slanting lines in the side of the cell rows denote the Cell Orientation.
— Provide the Core to Pad/ IO spacing (Core to IO clearance):–Core to IO clearance Used for Placing IOs and Power Ring.
— Pins/ Pads Placement,
— Macro Placement by Fly-line Analysis,
— Macro Placement requirements are also need to consider
— Blockage Management (Placement/ Routing)

III-C POWER PLANNING

To connect Power to the Chip.Levels of Power Distribution:
Rings:
—VDD and VSS Rings are formed around the Core and Macro
Stripes:
—Carries VDD and VSS around the chip
—Carries VDD and VSS from Rings across the chip
—Power Stripes are created in the Core Area to tap power from Core Rings to the core area.

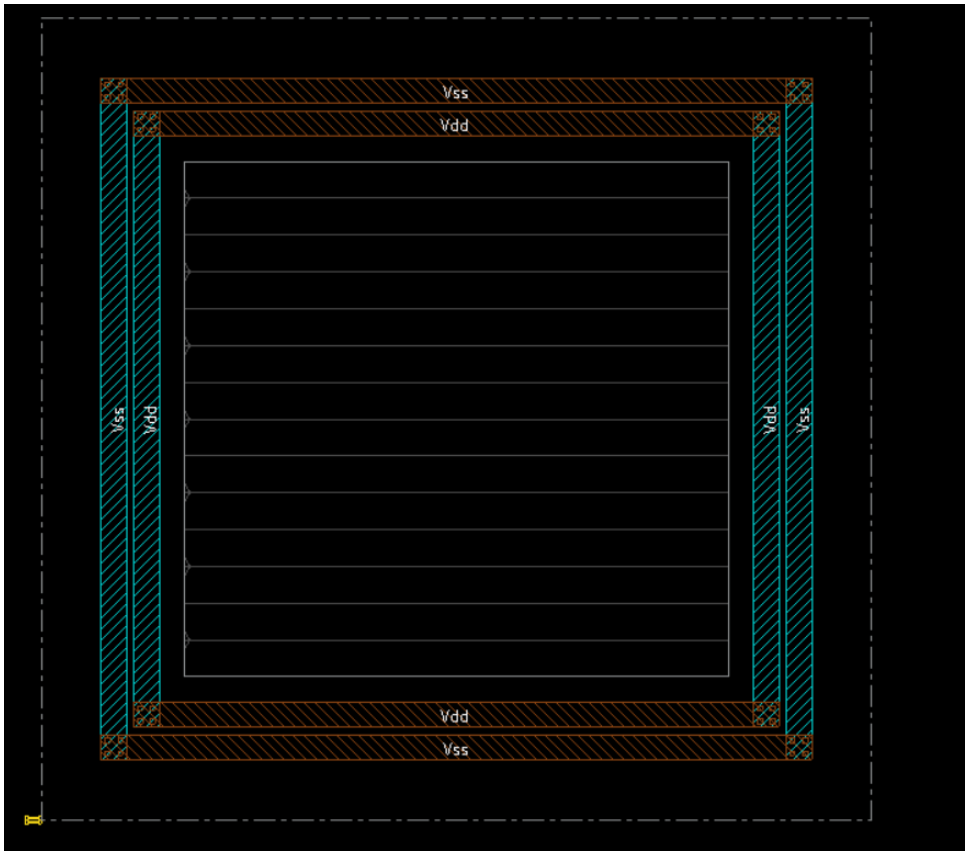


Fig. 46: POWER RINGS

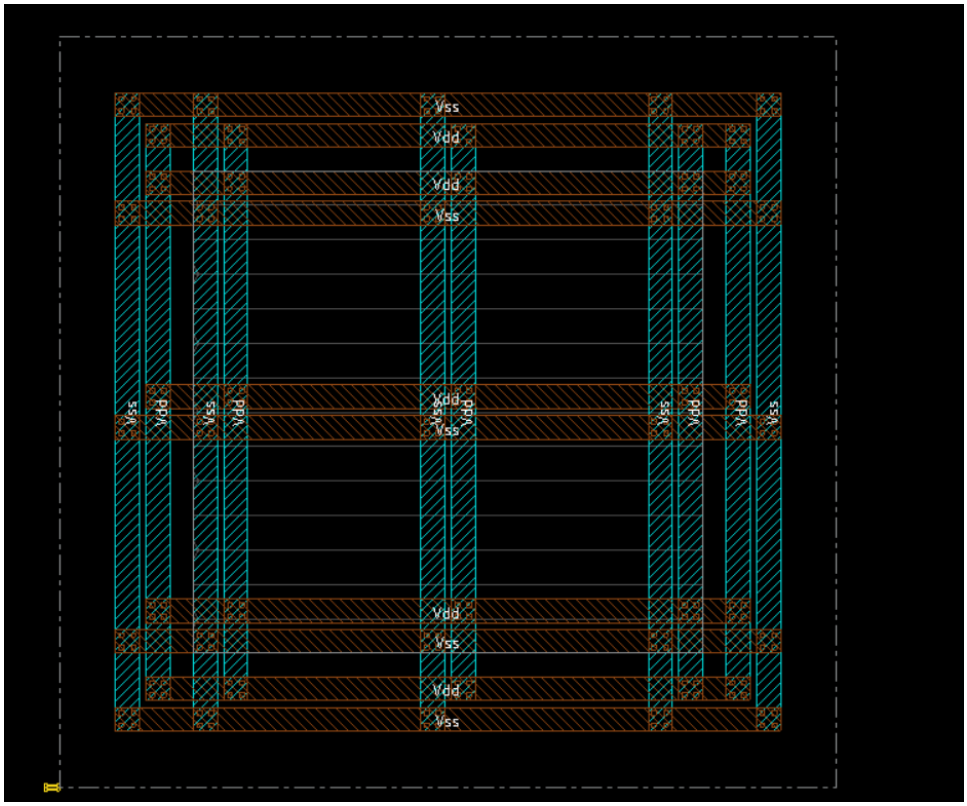


Fig. 47: Add Stripes

III-D Routing

Routing is the stage in physical design where electrical interconnections between placed cells are created using multiple metal layers. It is divided into global routing, which plans approximate paths, and detailed routing, which assigns exact tracks and vias. Proper routing ensures timing closure, maintains signal integrity, and avoids design rule violations. It also minimizes crosstalk, RC delays, and routing congestion. Techniques like shielding, non-default rules, and layer optimization are used to enhance performance. Efficient routing is critical for achieving high-speed, low-power, and manufacturable chip layouts.

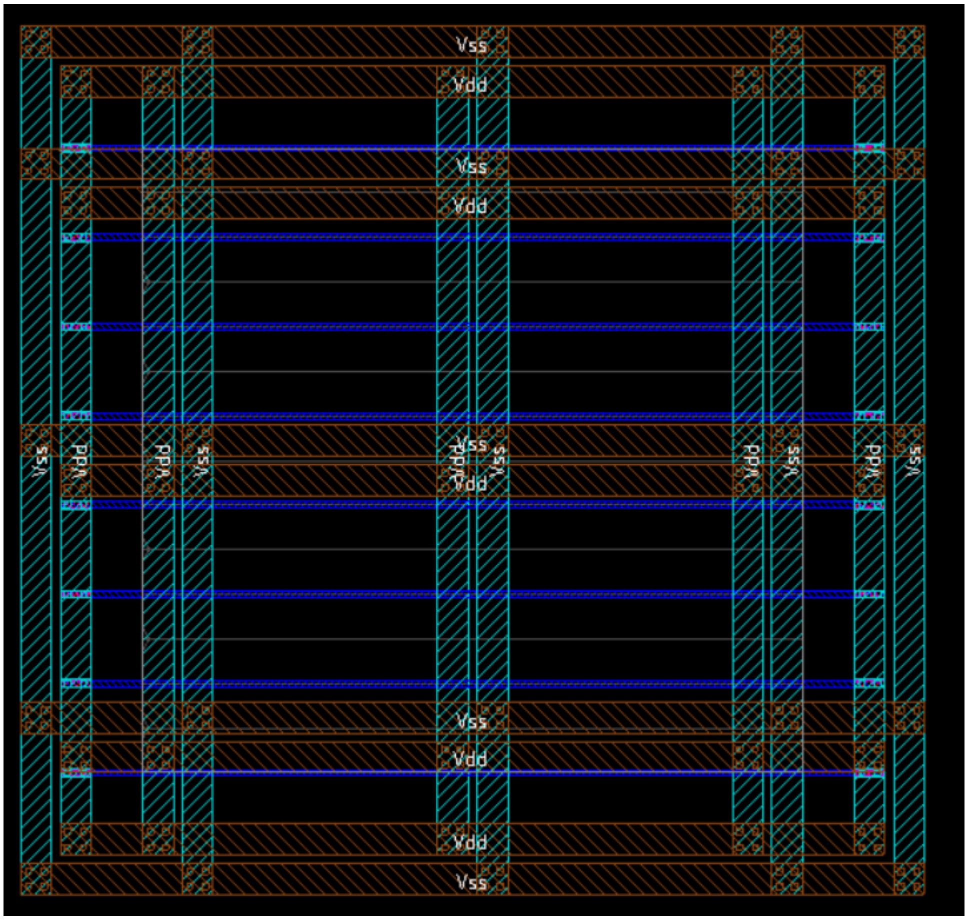


Fig. 48: Routing

III-E Pre -Placement

End Cap cells:

1. These cells prevent the cell damage during fabrication.
2. Used for row connectivity and specifying row ending.
3. To avoid drain and source short.
4. These are used to address boundary N-Well issues for DRC cleanup.

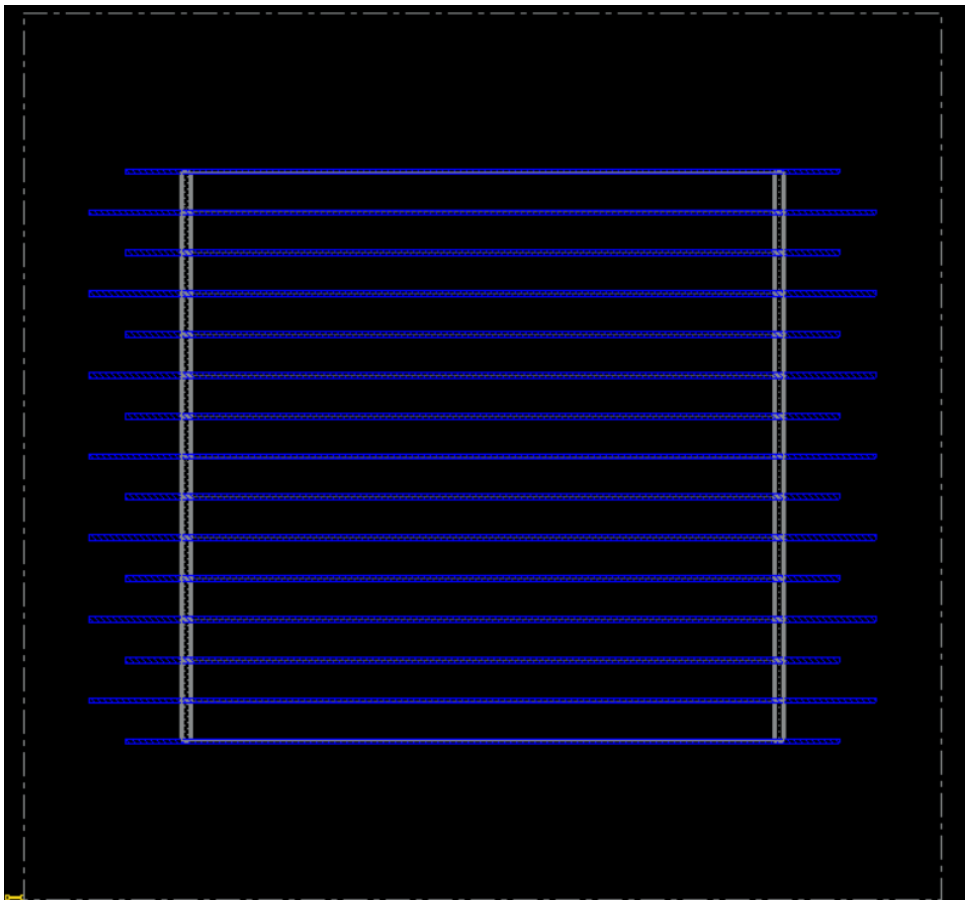


Fig. 49: End Cap

Well Tap cells:

1. These are used to connect VDD and GND to substrate and N-Well respectively because it results in lesser drift to prevent latch-up.
2. If we keep well taps according to the specified distances, N-Well potential leads to proper electrical functioning.
3. To limit the resistance between power and ground connections to wells of the substrate.

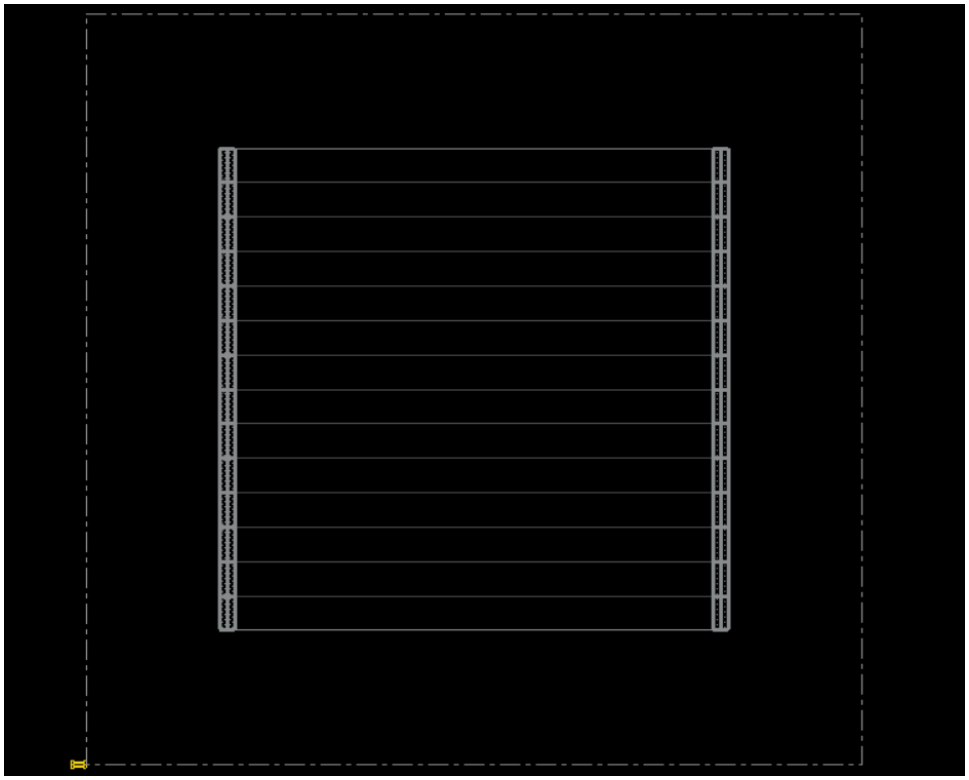


Fig. 50: Well Tap

III-F Placement

Placement for placing the Standard Cells in Placement Tracks.Placement Stages are:-

Global/ Coarse Placement:To get the approximate initial location,Cells are not legally placed and there can be overlapping.

Detail/ Legal Placement:To avoid cell overlapping,Cells have legalized locations, Legalize placement will place the cells in their legal position with no overlap.

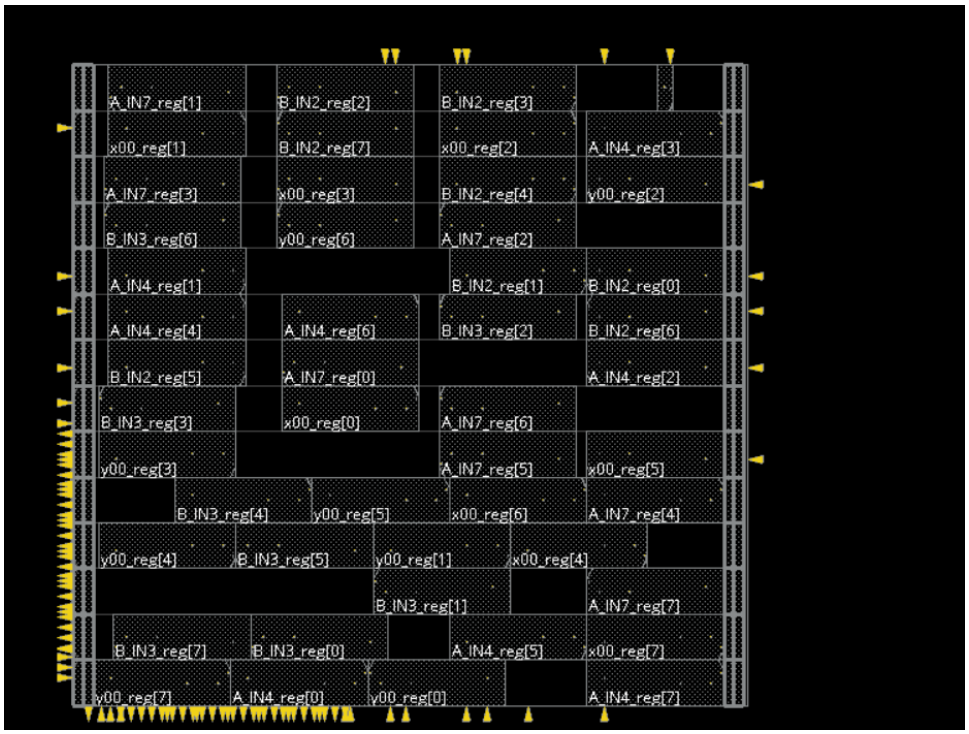


Fig. 51: Standard Cells

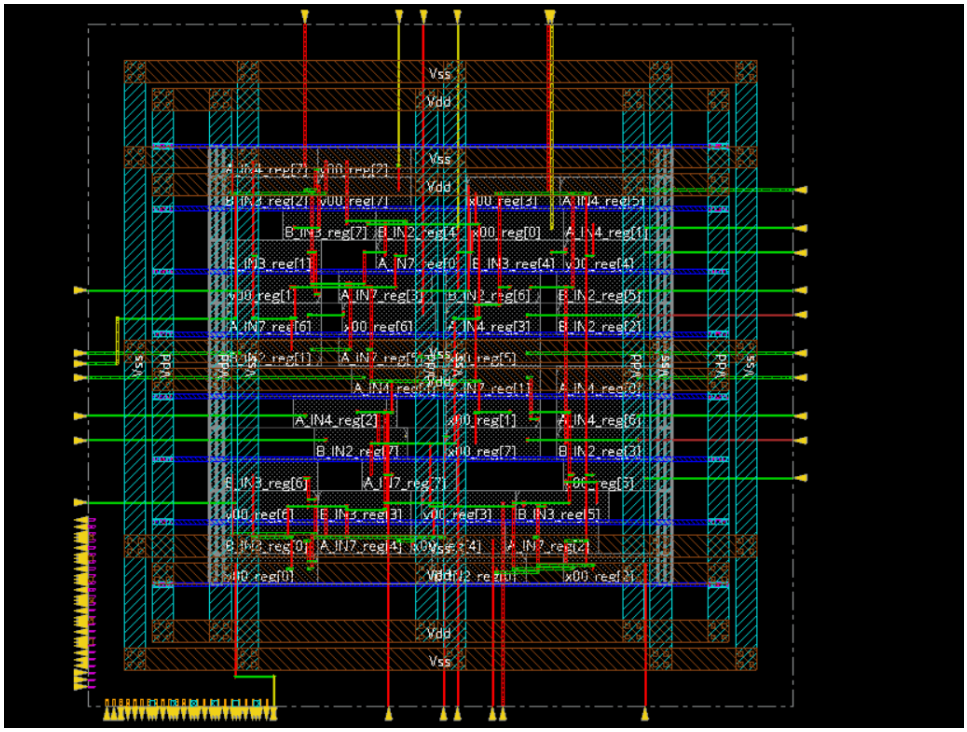


Fig. 52: Standard cell with routing

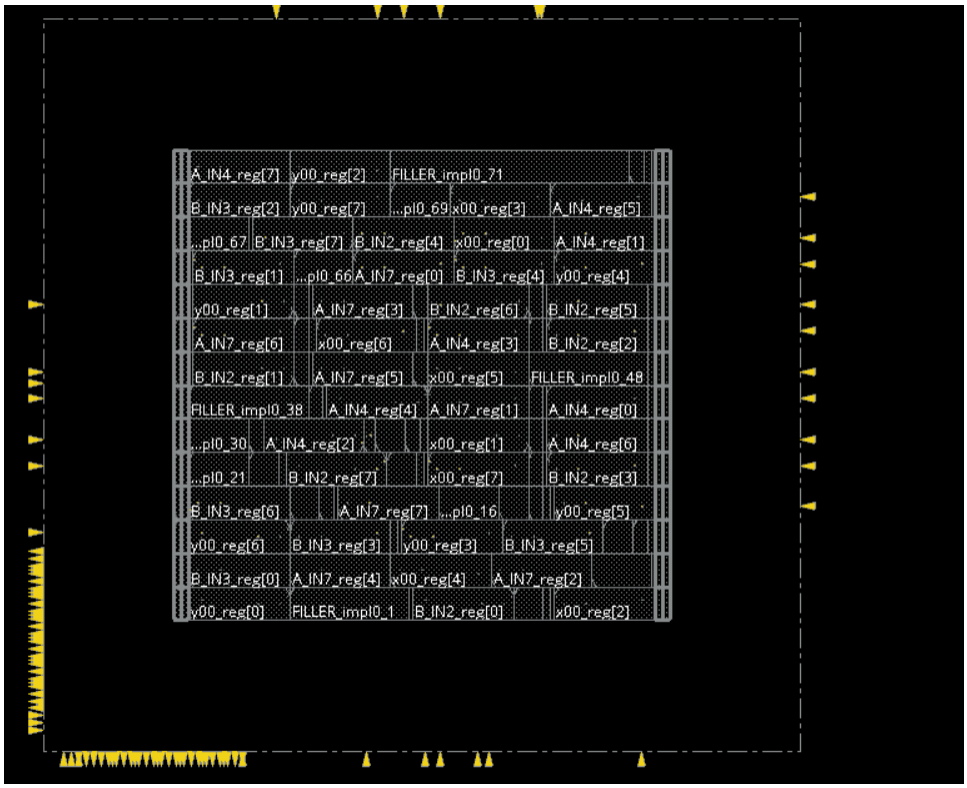


Fig. 53: Standard cell with fillers

Filler Cells: 1. To fill the empty space and provide connectivity of N-wells and implant layers.

III-G PRE CLOCK TREE SYNTHESIS HOLD ANALYSIS

Pre-CTS hold analysis is performed before clock tree synthesis to ensure that data paths are not violating hold time requirements. At this stage, ideal clocks are used, and the goal is to identify and fix early data arrival issues. Resolving hold violations early helps prevent timing problems after CTS.

```
#############################################################################
# Design Stage: PreRoute
# Design Name: systolic_array_3x3
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#############################################################################
AAE_INFO: 1 threads acquired from CTE.
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (1 T). (MEM=1404.07)
*** Calculating scaling factor for min_timing libraries using the default operating condition of each library.
Total number of fetched objects 83
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1486.97 CPU=0:00:00.0 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1486.97 CPU=0:00:00.1 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.1 real=0:00:00.0 totSessionCpu=0:02:20 mem=1487.0M)

------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

Hold  views included:
 hold

+--------------------+--------+---------+---------+
|     Hold mode      |  all   | reg2reg | default |
+--------------------+--------+---------+---------+
|           WNS (ns):| -0.041 | -0.041  |  0.000  |
|           TNS (ns):| -0.646 | -0.646  |  0.000  |
|    Violating Paths:|   16   |   16    |    0    |
|          All Paths:|   16   |   16    |    0    |
+--------------------+--------+---------+---------+

Density: 74.229%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.15 sec
Total Real time: 0.0 sec
Total Memory Usage: 1400.0625 Mbytes
innovus 1>
```

Fig. 54: PRE-CTS-HOLD

III-H PRE CLOCK TREE SYNTHESIS SETUP ANALYSIS

Pre-CTS setup analysis checks if data paths meet setup time requirements before clock tree synthesis, using ideal clock assumptions. This helps identify paths where data may arrive too late relative to the clock edge. Early detection allows designers to optimize placement and buffering to improve timing before building the clock tree.

```
Setup views included:
 setup

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  5.554  |  9.466  |  5.554  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   96    |   16    |   80    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      1 (1)       |   -0.153   |      1 (1)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 74.229%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.09 sec
Total Real time: 0.0 sec
Total Memory Usage: 1417.96875 Mbytes
innovus 1>
```

Fig. 55: PRE-CTS-SETUP

III-I CLOCK TREE SYNTHESIS

Clock Tree Synthesis is the process of creating this Clock Path from Clock Source to Clock Sinks.All Clock pins of flip Flop are considered as Clock Sinks (Leaf) where the Clock Tree Synthesis ends.

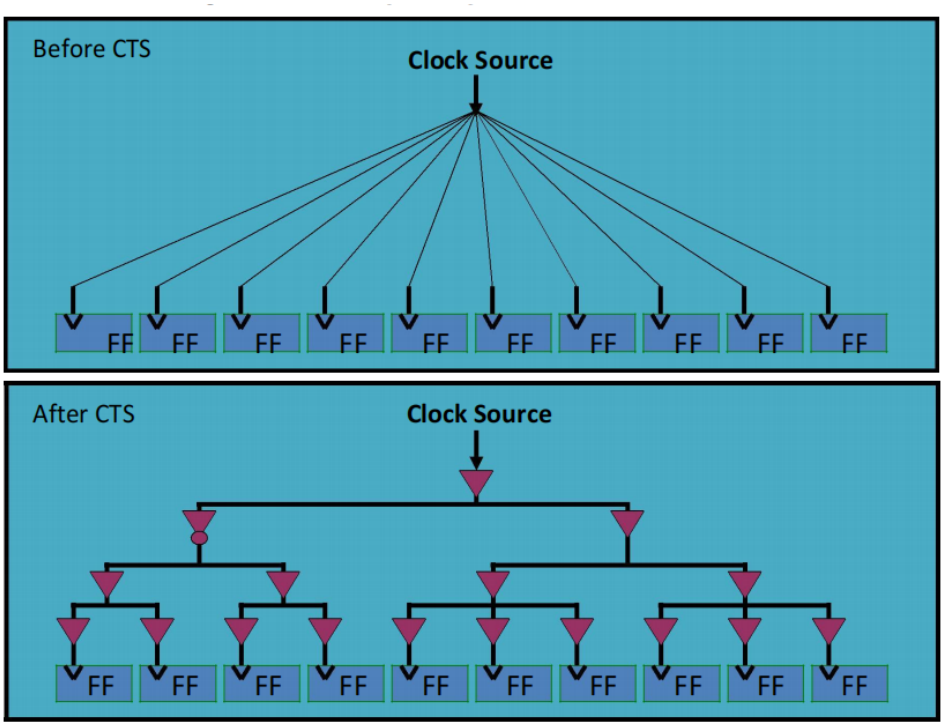


Fig. 56: CLOCK TREE SYNTHESIS

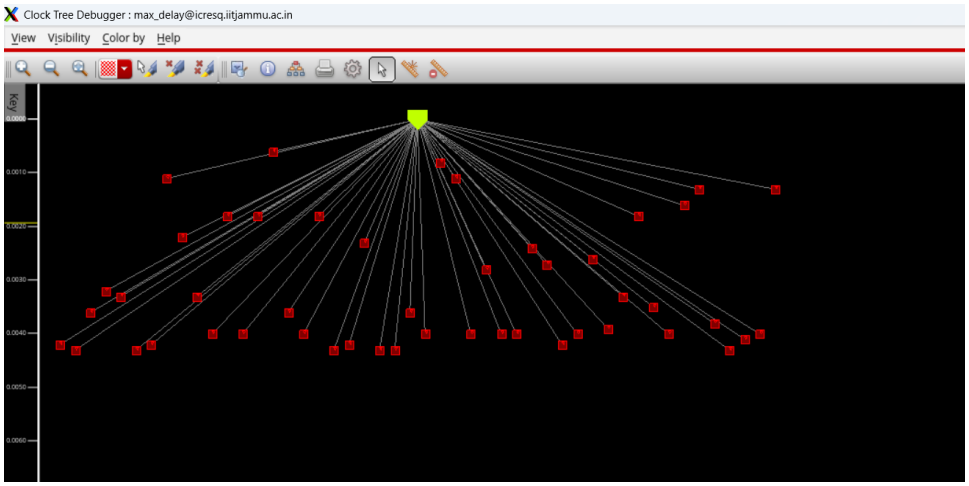


Fig. 57: CLOCK TREE
Clock Tree is a path from the Clock Source (Root) to Clock Sinks

III-J POS-CTS

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 hold

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.041  | -0.041  |  0.000  |
|           TNS (ns):| -0.646  | -0.646  |  0.000  |
|    Violating Paths:|   16    |   16    |    0    |
|          All Paths:|   16    |   16    |    0    |
+--------------------+---------+---------+---------+

Density: 74.229%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.13 sec
Total Real time: 0.0 sec
Total Memory Usage: 1402.796875 Mbytes
innovus 1>
```

Fig. 58: POST CTS HOLD

```
------------------------------------------------------------
        timeDesign Summary
------------------------------------------------------------

Setup views included:
 setup

--------------------+---------+---------+---------+
     Setup mode     |   all   | reg2reg | default |
--------------------+---------+---------+---------+
           WNS (ns):|  5.554  |  9.466  |  5.554  |
           TNS (ns):|  0.000  |  0.000  |  0.000  |
    Violating Paths:|    0    |    0    |    0    |
          All Paths:|   96    |   16    |   80    |
--------------------+---------+---------+---------+

----------------+-------------------------------+------------------+
                |             Real              |       Total      |
    DRVs        +------------------+------------+------------------|
                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
----------------+------------------+------------+------------------+
   max_cap      |      1 (1)       |   -0.153   |      1 (1)       |
   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
   max_fanout   |      0 (0)       |     0      |      0 (0)       |
   max_length   |      0 (0)       |     0      |      0 (0)       |
----------------+------------------+------------+------------------+

Density: 74.229%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.06 sec
Total Real time: 0.0 sec
Total Memory Usage: 1421.21875 Mbytes
```

Fig. 59: POST CTS SETUP

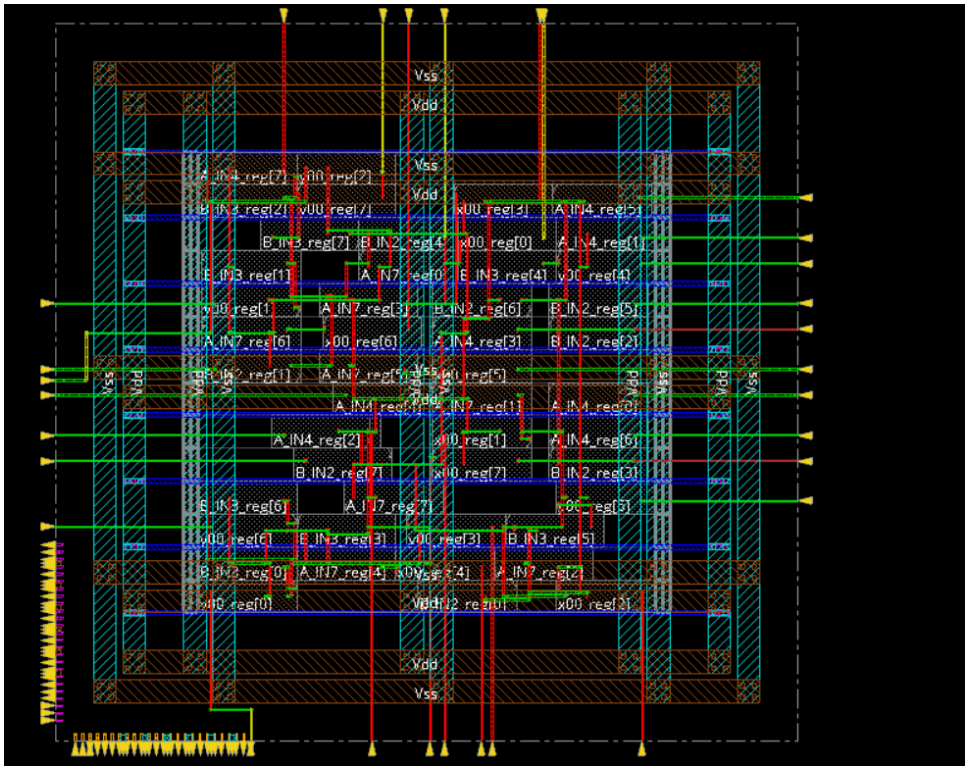


Fig. 60: Clock tree in design

Main concerns for Clock Design

Skew

Most important concern for clock networks For increased clock frequency, skew may contribute over 10Due to variations in trace length, metal width and height, coupling caps .

It can also be due to variations in local clock load, local power supply, local gate length and threshold, local temperature .

Power

Very important, as clock is a major power consumer.

It switches at every clock cycle.

— **Noise** Clock is often a very strong aggressor.

May need shielding.

Delay

Not really important.

But Slew Rate is important(sharp transition)