QMP 7.1 D/F

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Electronics & CommunicationEngineering

VLSI LAB

Sub Code: 21ECL66

B.E - VI Semester

Lab Manual 2023-24

Name : ______________________________________

USN : ______________________________________

Batch :__________________Section : _____________

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Electronics & CommunicationEngineering

VLSI LAB

Version 2.0

APRIL 2024

Prepared by:

Mrs. Bhavya B
Assistant Professor
Dept. of ECE

Reviewed by:

Dr. Sanjeevakumar Harihar
Associate Professor
Dept. of ECE

Approved by:

Dr. Thejaswini R,
Professor & Head, Dept.
of ECE

INSTITUTION VISION
To create centers of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership.

INSTITUTION MISSION

1. To provide high quality technical and professionally relevant education in a diverse learning environment.
2. To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.
3. To prepare the next generation of skilled professionals to successfully compete in the diverse global market.
4. To promote a campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.
5. To offer a wide variety of off-campus education and training programmes to individuals and groups.
6. To stimulate collaborative efforts with industry, universities, government and professional societies.
7. To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

VISION OF THE DEPARTMENT

"To Nurture and Develop Competent Electronics and Communication Engineering Skilled Professionals with values for the betterment of Society"

MISSION OF THE DEPARTMENT

1. To nurture the technical/professional/engineering and entrepreneurial skills for overall self and societal upliftment through co-curricular and extra-curricular events.
2. To orient the Faculty/Student community towards the higher education, research and development activities.
3. To create the Centres of Excellence in the field of electronics and communication in collaboration with industries/Universities by training the faculty through latest technologies.
4. To impart quality technical education in the field of electronics and communication engineering to meet over the current/future global industry requirements.

PROGRAM SPECIFIC OUTCOMES

PSO1: Build Analog and Digital Electronic systems for Multimedia Applications, VLSI and Embedded Systems in Interdisciplinary Research / Development.

PSO2: Design and Develop Communication Systems as per Real Time Applications and Current Trends.

PROGRAM EDUCATIONAL OBJECTIVES

PEO1 : Provide technical solutions to real world problems in the areas of electronics and communication by developing suitable systems.

PEO2 : Pursue engineering career in Industry and/or pursue higher education and Research.

PEO3: Acquire and follow best professional, ethical practices in Industry relevant to the Society.

PEO4 : Communicate effectively and Develop an ability to work in team/Individually

PROGRAMME OUTCOMES

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12**. Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

LIST OF LAB EXPERIMENTS

8	**a)** Capture schematics of two-stage operational amplifier and measure the following: a. UGB b. dB Bandwidth c. Gain Margin and phase margin with and without coupling capacitance d. Use the op-amp in the inverting and non-inverting configuration and verify its functionality. e. Study the UGB, 3dB bandwidth, gain and power requirement in op-amp by varying the stage wise transistor geometries and record the observations. **b)** Draw layout of two-stage operational amplifier with minimum transistor width set to 300 (in 180/90/45 nm technology), choose appropriate transistor geometries as per the results obtained in part a. Use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.	40 – 46
Demonstration Experiments (For CIE)		
9	UART • Write Verilog Code • Verify the Functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timingconstraints • Tabulate the Area, Power and Delay for the Synthesized netlist Identify Critical path	47 – 54
10	For synthesized netlist carry out the following: • Floor planning • Placement and Routing • Record the parameters such as no. of metal layers used for routing, flip method for placement of standard cells • Physical Verification and record the DRC and LVS reports • Generate GDSII	55 – 84

SYLLABUS

VLSI Laboratory			
Course Code	**21ECL66**	CIE Marks	50
Teaching Hours/Week (L: T: P: S)	0:0:2:0	SEE Marks	50
Credits	1	Exam Hours	3

Sl.No.	**Experiments**
	ASIC Digital Design
1	4-Bit Adder • Write Verilog Code • Verify the Functionality using Test-bench • Synthesize the design by setting proper constraints and obtain the netlist. From the report generated identify Critical path, Maximum delay, Total number of cells, Powerrequirement and Total area required
2	4-Bit Booth Multiplier • Write Verilog Code • Verify the Functionality using Test-bench • Synthesize the design by setting proper constraints and obtain the netlist. From the report generated identify Critical path, Maximum delay, Total number of cells, Power requirement and Total area required
3	32-Bit ALU Supporting 4-Logical and 4-Arithmetic operations, using case and if statement for ALUBehavioral Modeling • Write Verilog Code • Verify functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timing constraints • Tabulate the Area, Power and Delay for the Synthesized netlist • Identify Critical path
4	Latch and Flip-Flop • Synthesize the design and compare the synthesis report (D, SR, JK)
	ASIC Analog Design
5	a) Capture the schematic of CMOS inverter with load capacitance of 0.1pF and set the widths ofInverter with Wn = Wp, Wn = 2Wp, Wn = Wp/2 and length at selected technology. Carry out the following:
	i. Set the input signal to a pulse with rise time, fall time of 1ns and pulse width of 10ns and the time period of 20ns and plot the input voltage and output voltage of designed inverter? ii. From the simulation result compute tpHL, tpLH and td for all three geometrical settings of width? iii. Tabulate the results of delay and find the best geometry for minimum delay for CMOS inverter? b) Draw layout of inverter with Wp/Wn = 40/20, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.
6	a) Capture the schematic of 2-input CMOS NAND gate having similar delay as that of CMOS inverter computed in experiment above. Verify the functionality of NAND gate and also find out the delay td for all four possible combinations of input vectors. Table the results. Increase the drive strength to 2X and 4X and tabulate the results. b) Draw the layout of NAND with Wp/Wn = 40/20, use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre- layout simulations. Record the observations.
7	a) Capture schematic of Common Source Amplifier with PMOS Current Mirror Load and find its transient response and AC response? Measure the Unit Gain Bandwidth (UGB), amplification factor by varying transistor geometries, study the impact of variation in width to UGB. b) Draw Layout of common source amplifier, use optimum layout methods. Verify for DRC & LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

8	a) Capture schematics of two-stage operational amplifier and measure the following: i. UGB ii. dB Bandwidth iii. Gain Margin and phase margin with and without coupling capacitance iv. Use the op-amp in the inverting and non-inverting configuration and verify its functionality. v. Study the UGB, 3dB bandwidth, gain and power requirement in op-amp by varying the stage wise transistor geometries and record the observations. b) Draw layout of two-stage operational amplifier with minimum transistor width set to 300 (in 180/90/45 nm technology), choose appropriate transistor geometries as per the results obtained in part a. Use optimum layout methods. Verify for DRC and LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.
	Demonstration Experiments (For CIE)
9	UART • Write Verilog Code • Verify the Functionality using Test-bench • Synthesize the design targeting suitable library and by setting area and timing constraints • Tabulate the Area, Power and Delay for the Synthesized netlist, Identify Critical path
10	For synthesized netlist carry out the following: • Floor planning • Placement and Routing • Record the parameters such as no. of metal layers used for routing, flip method for placement of standard cells • Physical Verification and record the DRC and LVS reports • Generate GDSII
11	Design and characterize 6T binary SRAM cell and measure the following: • Read Time, Write Time, SNM, Power • Draw Layout of 6T SRAM, use optimum layout methods. Verify for DRC & LVS, extract parasitic and perform post layout simulations, compare the results with pre-layout simulations. Record the observations.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubricsfor the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of thepractical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up willbe evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th weekof the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course is 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners).

Change of experiment is allowed only once and 15% Marks allotted to the procedure part to be made zero.

The duration of SEE is 03 hours.

Rubrics suggested in Annexure-II of Regulation book

Department of Electronics and Communication Engineering

VLSI LAB
Course Objectives and Outcomes

Course objectives: This course will enable students to:

- Design, model, simulate and verify digital circuits.
- Design layouts and perform physical verification of CMOS digital circuits.
- Perform ASIC design flow and understand the process of synthesis, synthesis constraints andevaluating the synthesis reports to obtain optimum gate level netlist.
- Perform RTL-GDSII flow and understand the stages in ASIC.

Course outcomes: After studying this course, students will be able to:

- Design and simulate combinational and sequential digital circuits using Verilog HDL.
- Understand the synthesis process of digital circuits using EDA tool.
- Perform ASIC design flow and understand the process of synthesis, synthesis constraints andevaluating the synthesis reports to obtain optimum gate level netlist.
- Design and simulate basic CMOS circuits like inverter, common source amplifier, differentialamplifier, SRAM.
- Perform RTL_GDSII flow and understand the stages in ASIC design.

INSTRUCTIONS TO THE STUDENT

1. Come prepared to the lab with relevant theory about the experiment you are conducting.

2. Each experiment will be evaluated for 20 marks. More weightage will be given for preparation and understanding.

3. Handle the desktop system and interfacing boards with care

4. Do not delete or change the system settings and files.

5. For any missing items, penalty will be imposed on the respective batch.

6. Maintain professional attitude and discipline during lab sessions.

INTRODUCTION

VLSI stands for Very Large Scale Integration, which is a technology used to create integrated circuits (ICs) by combining thousands or millions of transistors into a single chip. VLSI technology has revolutionized the electronics industry by enabling the production of compact, powerful and low-cost microprocessors, memory chips, digital signal processors, and other advanced electronic devices.

As of 2020, the scale of integration in VLSI technology has reached nanometer level processes, with feature sizes on the order of 7-5 nm, allowing for the integration of billions of transistors on a single chip. The latest processors and integrated circuits are being manufactured using 7nm and 5nm process nodes, enabling greater performance and power efficiency.

ASIC Design Flow

(Digital Design)

PART - A

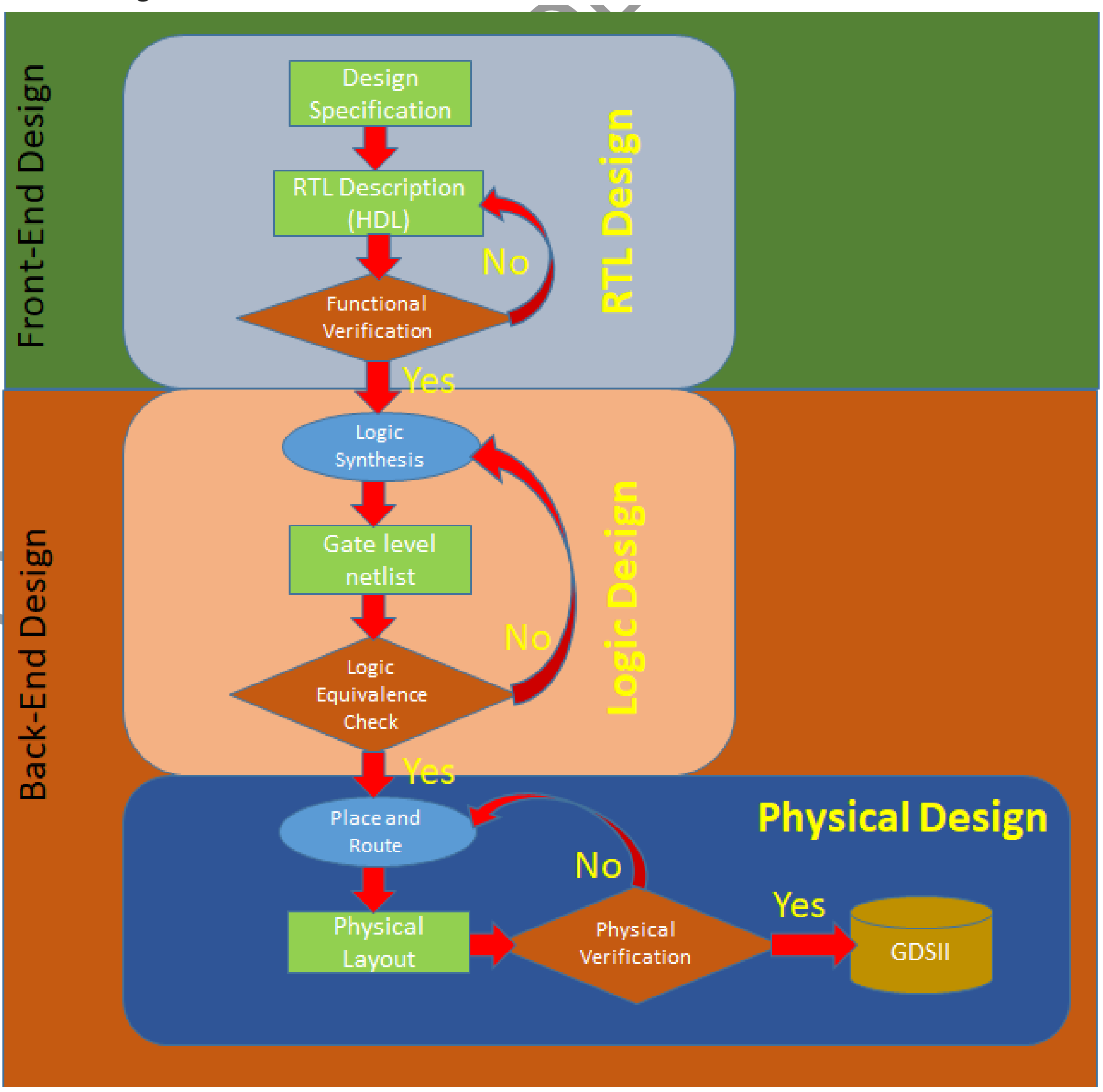
What Is an ASIC ?

An ASIC (Application-Specific Integrated Circuit) is a type of integrated circuit (IC) that is designed to perform a specific task or function. It is customized for a particular application, unlike general-purpose ICs such as microprocessors and memory chips.

ASICs are typically used in high-performance applications where specific processing requirements need to be met, such as in networking, telecommunications, and consumer electronics. ASICs are designed and manufactured for a specific customer or application and can include digital, analog, and mixed-signal components on a single chip.

The complete ASIC design process can be divided into two parts.

1. Front End Design
2. Back End Design



Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

Lab 1: 4-Bit Adder

Aim: To write a verilog code for 4bit adder and verify the functionality using Test bench.

- Synthesize, Analyse Reports and Netlist, Critical Path and Max Operating Frequency.
- From the report generated find the total number of cells, power requirement and total area requirement.

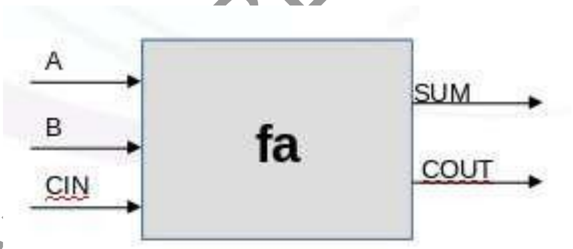
Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

Design Information and Bock Diagram:

A full adder is a combinational circuit that performs the arithmetic sum of three input bits Ai, addend Bi and carry in C in from the previous adder. Its results contain the sum Si and the carryout, C out to the next stage. So to design a 4-bit adder circuit we start by designing the 1 –bit full adder then connecting the four 1-bit full adders to get the 4-bit adder as shown in the diagram below. For the 1-bit full adder, the design begins by drawing the Truth Table for the three input and the corresponding output SUM and CARRY.

Fig: Diagram and truth table of full adder



A	B	C	SUM OUT	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

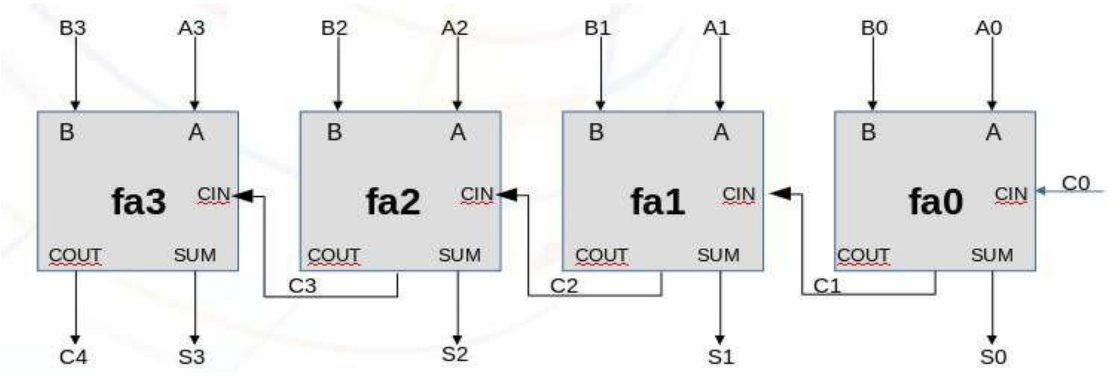


Fig: Diagram of 4 Bit Adder

Creating a Work space :

- Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

a) Verify the Functionality

- Three Codes shall be written for implementation of 4-bit Adder as follows,
  - fa.v → Single Bit 3-Input Full Adder [Sub-Module / Function]
  - fa_4bit.v → Top Module for Adding 4-bit Inputs.
  - fa_test.v → Test bench

Source Code – fa.v :-

```
 module full_adder( A,B,CIN,S,COUT);
 input A,B,CIN;
output S,COUT;
assign S = A^B^CIN;
assign COUT = (A&B) | (CIN&(A^B));
 endmodule
```

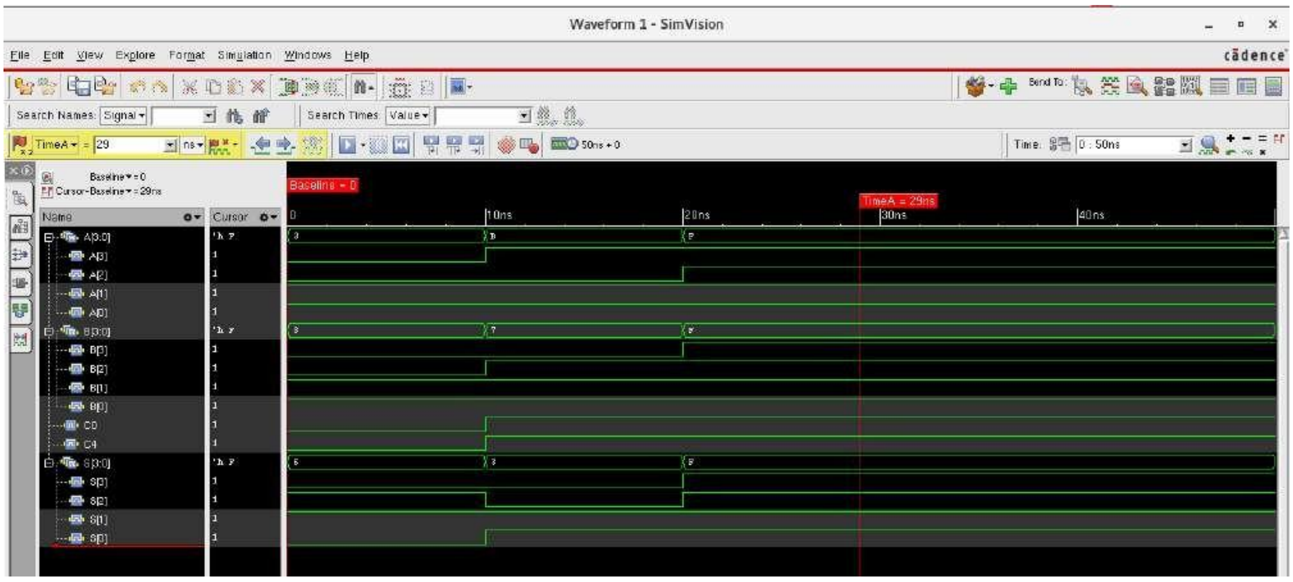
Source Code – fa_4bit.v :-

```
module four_bit_adder(A,B,C0,S,C4);
input [3:0] A,[3:0] B,C0;
output [3:0] S,C4;
wire C1,C2,C3;
   full_adder fa0 (A[0],B[0],C0,S[0],C1);
   full_adder fa1 (A[1],B[1],C1,S[1],C2);
   full_adder fa2 (A[2],B[2],C2,S[2],C3);
   full_adder fa3 (A[3],B[3],C3,S[3],C4);
   endmodule
```

Test Bench – fa_test.v :-

```
module test_4_bit;
     reg [3:0] A;
     reg [3:0] B;
     reg C0;
     wire [3:0] S;
     wire C4;
     four_bit_adder dut(A,B,C0,S,C4);
     initial begin
     A = 4'b0011;B=4'b0011;C0 = 1'b0; #10;
     A = 4'b1011;B=4'b0111;C0 = 1'b1; #10;
     A = 4'b1111;B=4'b1111;C0 = 1'b1; #10;
     endinitial
     #50 $finish;
     endmodule
```

Waveform :

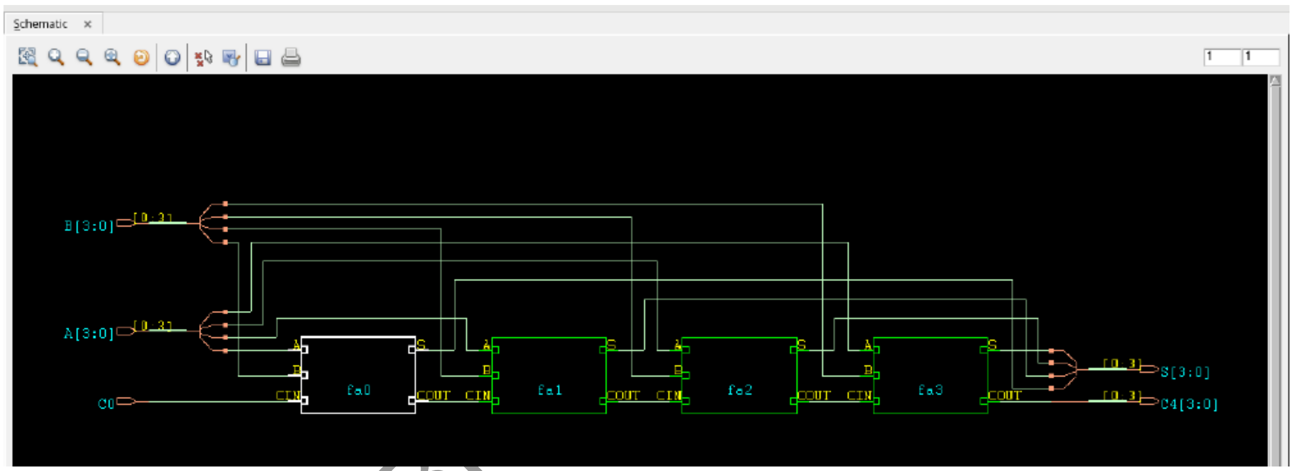


b) Synthesis and Report/Output Analysis

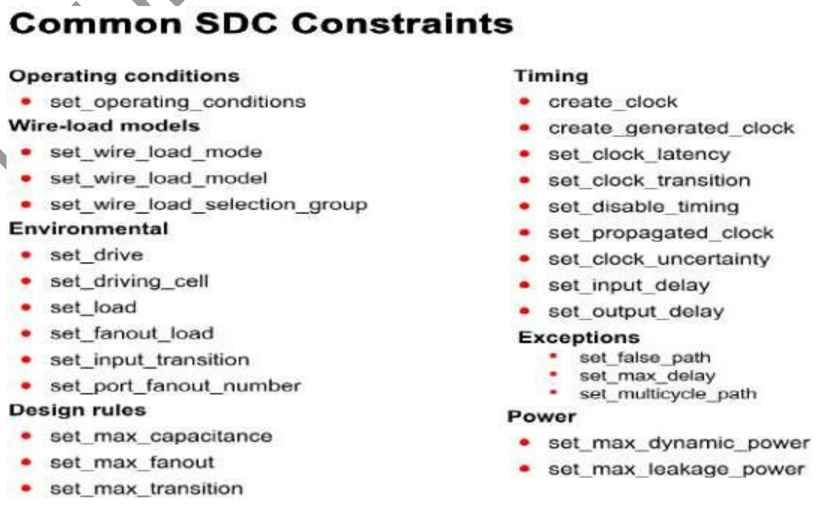
Note 1:-

1. The Cells given in the netlist can be checked in the .lib files for their properties.
2. The Max Operating Frequency does not apply for Purely Combinational Circuit.

Synthesis RTL Schematic :



Some Common Constraints are given below for reference



Common SDC Constraints

Operating conditions
- set_operating_conditions

Wire-load models
- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group

Environmental
- set_drive
- set_driving_cell
- set_load
- set_fanout_load
- set_input_transition
- set_port_fanout_number

Design rules
- set_max_capacitance
- set_max_fanout
- set_max_transition

Timing
- create_clock
- create_generated_clock
- set_clock_latency
- set_clock_transition
- set_disable_timing
- set_propagated_clock
- set_clock_uncertainty
- set_input_delay
- set_output_delay

Exceptions
- set_false_path
- set_max_delay
- set_multicycle_path

Power
- set_max_dynamic_power
- set_max_leakage_power

Note 2:-

1. You can tabulate Area, Power and Timing Constraints using any of the SDC Constraints asinstructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports donot overwrite the earlier ones.

Lab 2: 4 – Bit Booth Multiplier

Aim: To write a verilog code for 4bit Booth Multiplier and verify the functionality using Test bench.

- Synthesize, Analyse Reports and Netlist, Critical Path and Max Operating Frequency.
- From the report generated find the total number of cells, power requirement and total area requirement.

Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

Design Information and Bock Diagram:

In Booth's multiplier works on Booth's Algorithm that does the multiplication of 2's complement notation of two signed binary numbers.

Advantages:

1. Less complexity
2. Faster Multiplication
3. Consecutive additions are replaced
4. Ease in scaling

Disadvantage:

1. High power consumption
2. Large chip area

Algorithm:

Registers used: A, M, Q, Qres (Qres is the residual bit after a right shift of Q), n (counter)

Step 1: Load the initial values for the registers.

A = 0 (Accumulator), Qres = 0, M = Multiplicand, Q = Multiplier and n is the count value which equals the number of bits of multiplier.

Step 2: Check the value of {Q0, Qres}. If 00 or 11, goto step 5. If 01, goto step 3. If 10, goto step 4.

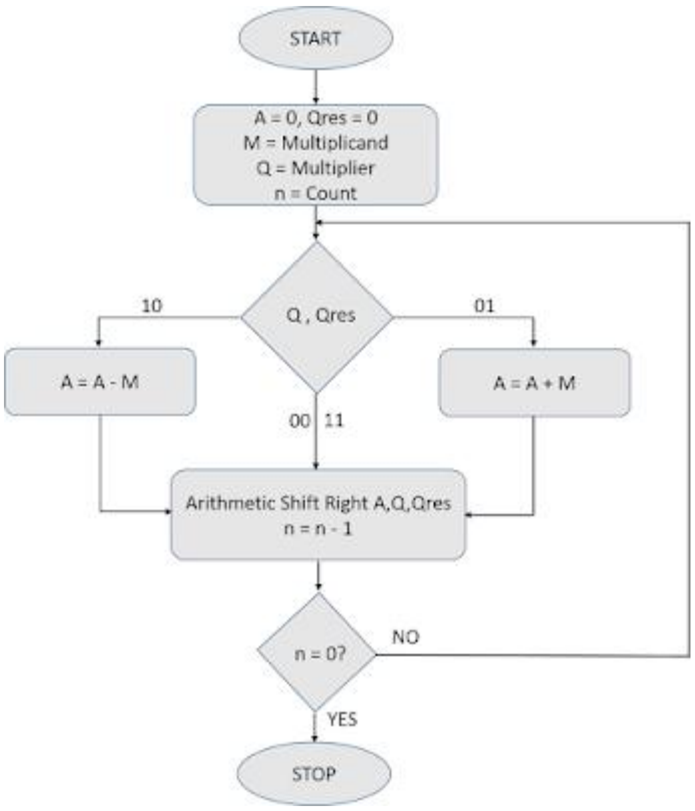
Step 3: Perform A = A + M. Goto step 5.

Step 4: Perform A = A - M.

Step 5: Perform Arithmetic Shift Right of {A, Q, Qres} and decrement count.

Step 6: Check if counter value n is zero. If yes, goto next step. Else, goto step 2.

Step 7: Stop



Creating a Work space :

- Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

a) To Verify the Functionality using Test Bench

Source Code – boothmul.v :-

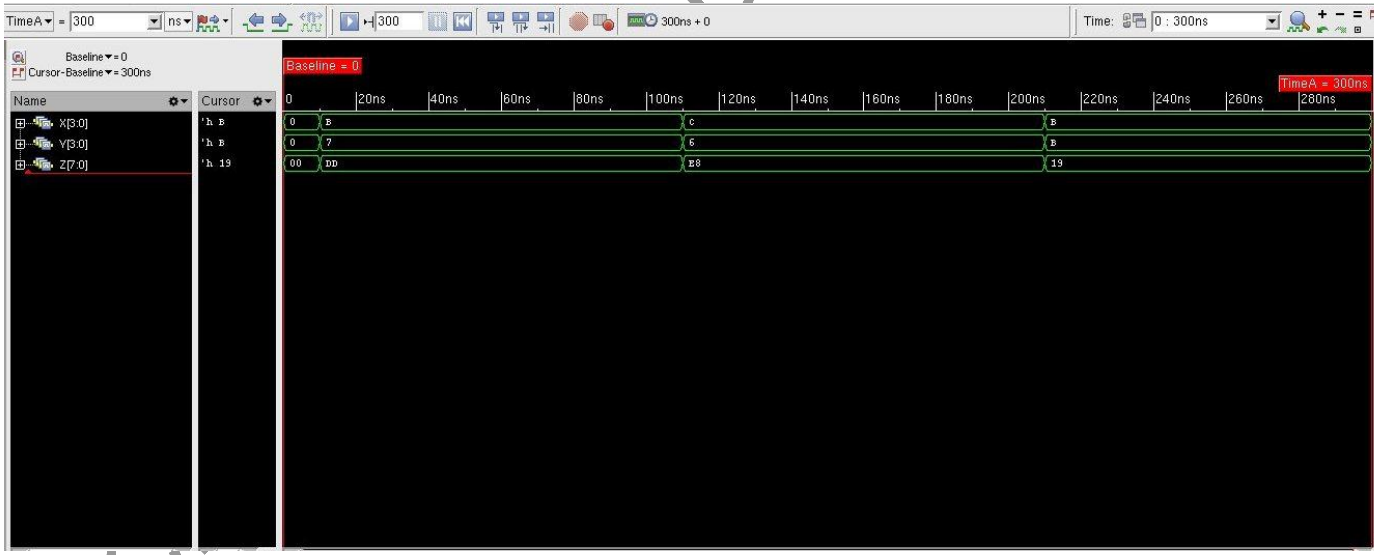
```
 module BoothMulti (X, Y, Z);
input signed [3:0] X, Y;
 output signed [7:0] Z;
 reg signed [7:0] Z;
 reg [1:0] temp;
 integer i;
 reg E1;
 reg [3:0] Y1;
 always @ (X, Y)
       begin
       Z = 8'd0;
       E1 = 1'd0;
       for (i = 0; i < 4; i = i + 1)
                    begin
                      temp = {X[i], E1};
                    Y1 = - Y;
                    case (temp)
                       2'd2 : Z [7 : 4] = Z [7 : 4] + Y1;
                      2'd1 : Z [7 : 4] = Z [7 : 4] + Y;
                     default : begin end
                   endcase
 Z = Z >> 1;
 Z[7] = Z[6];
 E1 = X[i];
     end
 if (Y == 4'd8)
      begin
         Z = - Z;
     end
end
endmodule
```

Test Bench – boothmul test.v :-

```
module BoothTB;
   // Inputs
   reg [3:0] X;
   reg [3:0] Y;
   // Outputs
   wire [7:0] Z;
   // Instantiate the Unit Under Test (UUT)
   BoothMulti uut (
      .X(X),
      .Y(Y),
      .Z(Z)
   );
   initial begin
   // Initialize Inputs
      X = 0;
      Y = 0;
  // Wait 100 ns for global reset to finish
```

```
        #10;
        X=-5;
        Y=7;
  #100;
        X=-4;
        Y=6;
#100;
X=-5;
Y=-5;
        // Add stimulus here
    end
```

Waveform :

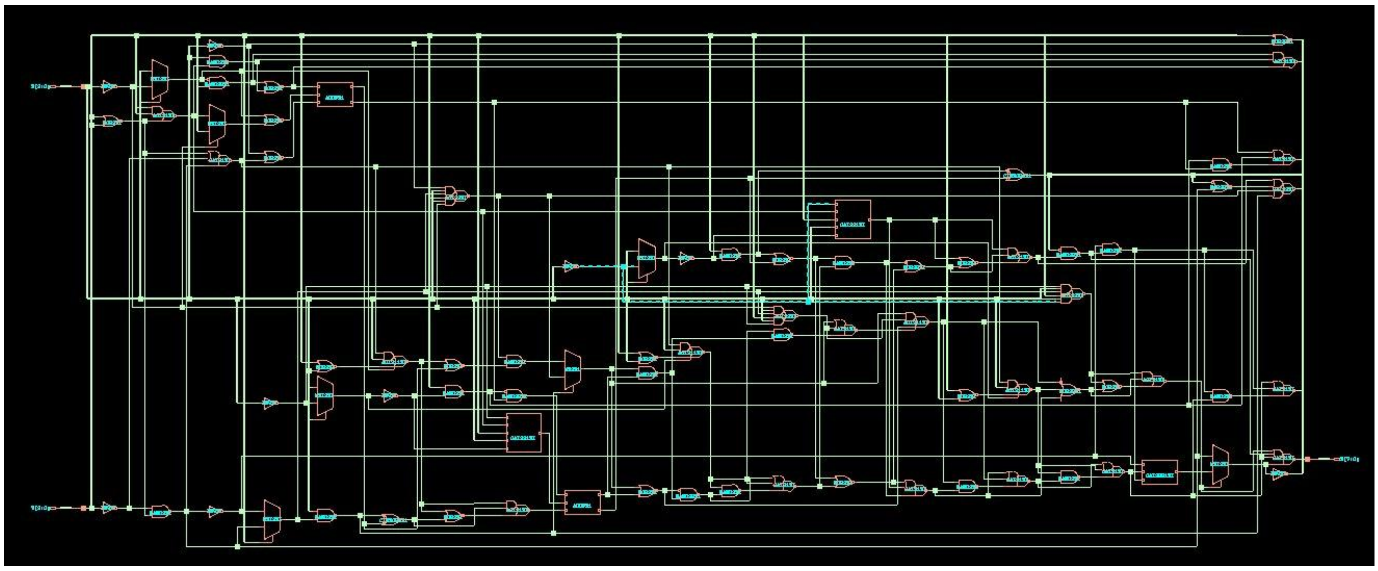


b) Synthesis and Report/Output Analysis

Note 1:-

1. The Cells given in the netlist can be checked in the .lib files for their properties.
2. The Max Operating Frequency does not apply for Purely Combinational Circuit.

Synthesis RTL Schematic :



Some Common Constraints are given below for reference

Lab 3 : 32-bit ALU

Aim: Write a verilog code for 32 bit ALU supporting four logical and four arithmetic operations,use case statement and if statement for ALU behavioral modeling.

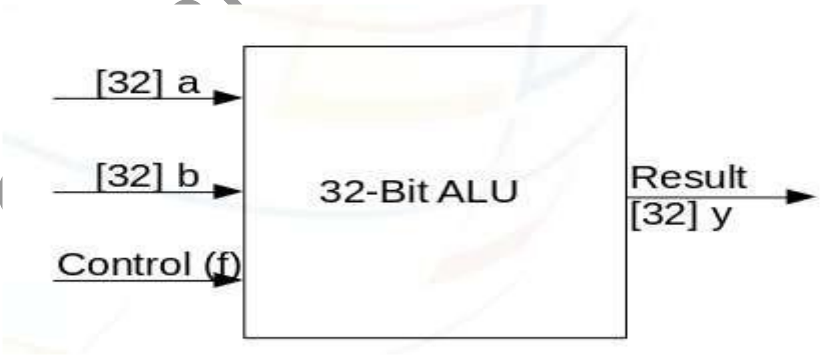
- To Verify the Functionality using Test Bench
- Synthesize and compare the results using if and case statements
- Identify Critical Path and constraints

Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

Design Information and Bock Diagram:

The ALU will take in two 32-bit values, and control line. An Arithmetic unit does the following task like addition subtraction, multi-fiction and logical operations. As the input is given in32 bit we get 32 bit output. The arithmetic will show only one output at a time so a selector is necessary to select one of the operator.



Creating a Work space :

- Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

To Verify the Functionality using Test Bench

Source Code – Using Case Statement :

```
module 32bit_alu_case(a, b, f, y);
input [31:0]a;
input [31:0]b;
input  [2:0]f;
output reg [31:0]y;
always @ (*)
begin
case (f)
   3'b000: y=a&b;          //AND Operation
   3'b001: y=a|b;          //OR Operation
   3'b010: y=~(a&b);       //NAND Operation
   3'b011: y=~(a|b);       //NOR Operation
   3'b100: y=a+b;          //Addition
   3'b101: y=a-b;          //Subtraction
   3'b110: y=a*b;          //Multiply
   default: y=32'bx;
    endcase
 end
 endmodule
```

<u>Test Bench :</u>

```
module 32bit_alu_case_tb;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
32bit_alu_case test2(.a(a),.b(b),.f(f),.y(y));
initial
begin
a=32'h00000000;
b=32'hFFFFFFFF;
#10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b011;
#10 f=3'b100;
#10 f=3'b101;
#10 f=3'b110;
end
initial
#50 $finish;
endmodule
```

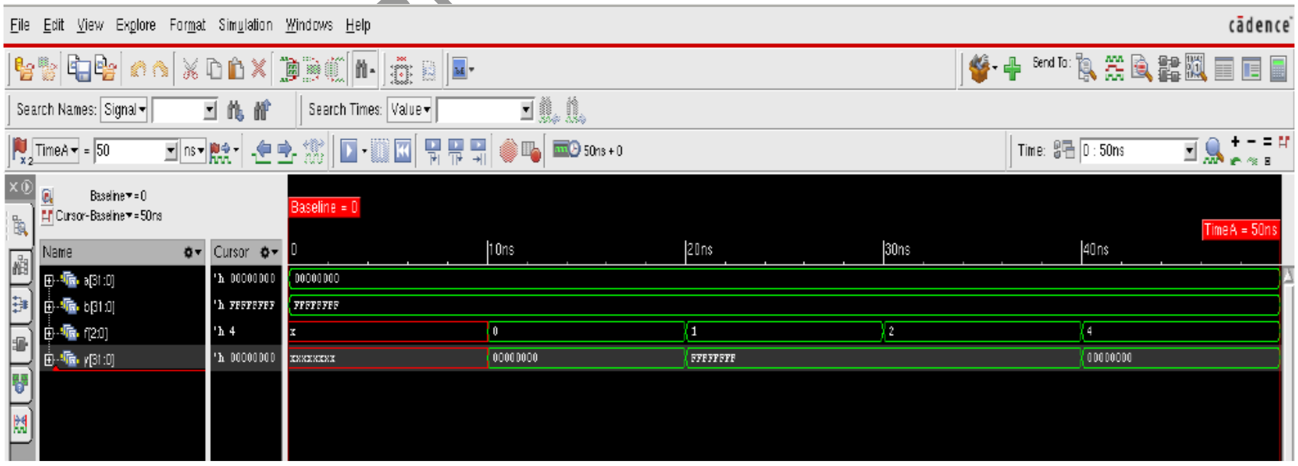
<u>Source Code - Using If Statement :</u>

```
module 32bit_alu_if(y,a,b,f);
input [31:0]a;
input [31:0]b;
input  [2:0]f;
output reg [31:0]y;
always@(*)
begin
 if(f==3'b000)
  y=a&b;                    //AND Operation
  else
  if (f==3'b001)
  y=a|b;                    //OR Operation
  else
  if (f==3'b010)
  y=a+b;                    //Addition
  else
  if (f==3'b011)
  y=a-b;                    //Subtraction
  else
  if (f==3'b100)
  y=a*b;                    //Multiply
  else
  y=32'bx;
  end
  endmodule
```

<u>Test bench :</u>

```
module 32bit_alu_tb_if;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
32bit_alu_if  test(.y(y),.a(a),.b(b),.f(f));
initial
begin
a=32'h00000000;
b=32'hFFFFFFFF;
#10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b100;
end
 initial
#50 $finish;
endmodule
```

Wave Forms :



a) Synthesize Design

- Run the synthesis Process one time for each code and make sure the output File names are changed accordingly.

Synthesis RTL Schematic :

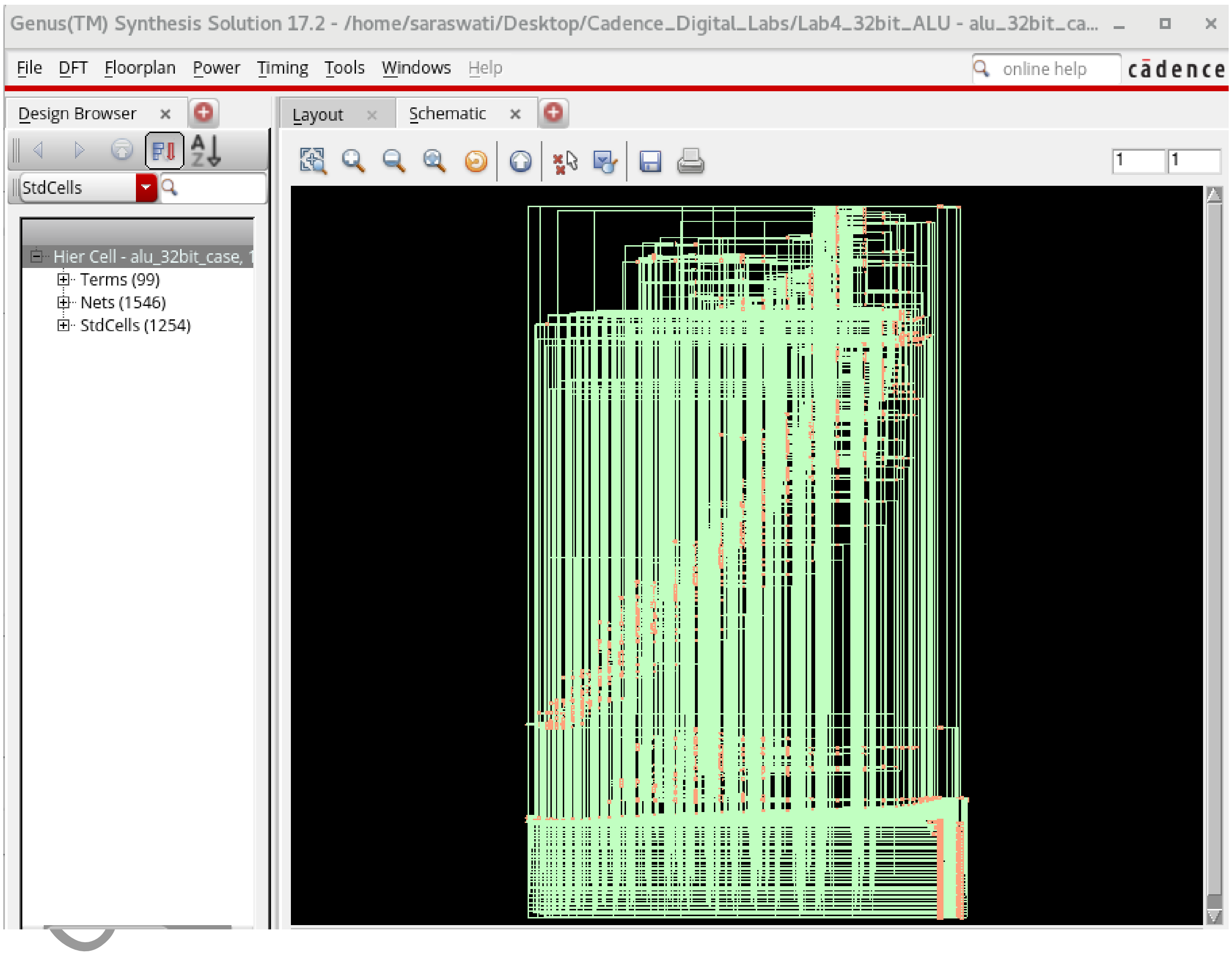

Genus(TM) Synthesis Solution 17.2 - /home/saraswati/Desktop/Cadence_Digital_Labs/Lab4_32bit_ALU - alu_32bit_ca...
File DFT Floorplan Power Timing Tools Windows Help
online help
cadence
Design Browser
Layout
Schematic
StdCells
Hier Cell - alu_32bit_case,
Terms (99)
Nets (1546)
StdCells (1254)

Fig: Schematic Capture of 32 Bit ALU

Note :-

1. You can tabulate Area, Power and Timing Constraints using any of the SDC Constraints asinstructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports donot overwrite the earlier ones.

Lab 4: Latches and Flip Flops

Aim : Write a verilog code for Latch and Flip-flops (D, SR, JK), Synthesize the design and compare the synthesis report.

Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

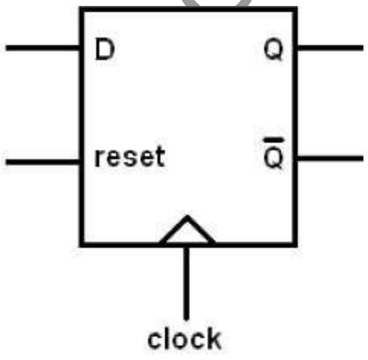
Design Information and Bock Diagram:

Latches and flip-flops are the basic elements for storing information. One latch or flip-flopcan store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted.

In other words, when they are enabled, their content changes immediately when their inputschange. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

There are basically four main types of latches and flip-flops: SR, D, and JK. The major differences in these flip-flop types are the number of inputs they have and how they change state.For each type, there are also different variations that enhance their operations.

Example: D-Flip-flop



Input			Output	
D	reset	clock	Q	Q'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Fig: Block Diagram and truth table of D-Flip Flop

Creating a Work space :

- Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

a) Verilog Codes for D-Flip Flop, JK-Flip Flop and SR-Flip Flop.

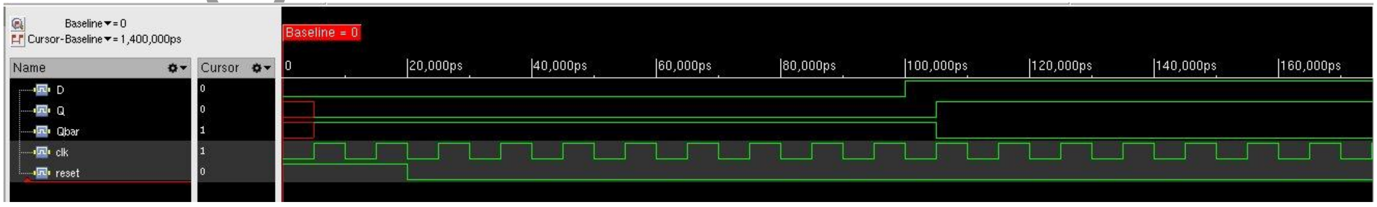
Source code for D-Flip Flop :

```
`timescale 1ns/1ps
module DFF( Q, Qbar, D, Clk, Reset);
output  reg  Q;
output  Qbar;
input D, Clk, Reset;
always @(posedge Clk)
begin
 if (Reset == 1'b1)              //If at reset
 Q <= 1'b0;
 Else
 Q <= D;
end
assign Qbar = ~Q;
endmodule
```

Test bench :

```
`timescale 1ns/1ps
module DFF_test;
reg clk, reset, D;
wire q, qbar;
initial
begin
clk=0;
reset=1; #20;
reset=0;
end
initial
begin
D=0; #100;
D=1; #100;
D=0;
end
dff uut(Q, Qbar, D,clk, reset);
always  #5  clk=~clk;                // Inverting clk every 5ns
Initial
#1400 $finish;                       // Finishing Simulation at t=1400ns
endmodule
```

Wave Forms :



Source code for D-Latch :

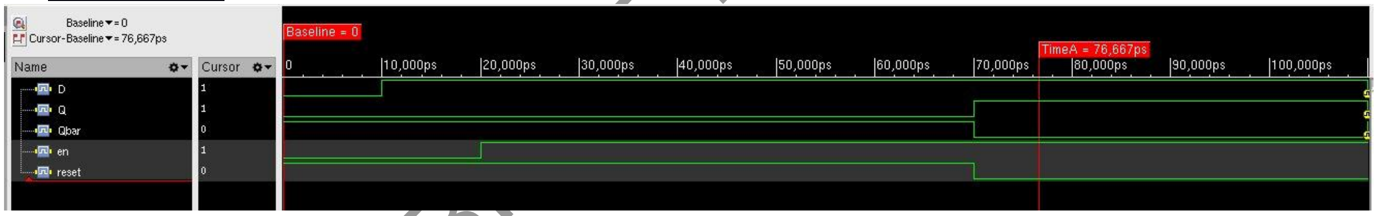
```
`timescale 1ns/1ps
module Dlatch ( Q, Qbar, D, en, Reset);
output  reg  Q;
output   Qbar;
input D, en, Reset;
always @(en or Reset or D)
begin
 if (Reset == 1)                 //If at reset
 Q <= 1'b0;
 else
  if (en)
  Q <= D;
end
assign Qbar = ~Q;
endmodule
```

Test bench :

```
`timescale 1ns/1ps
module Dlatch_test;
reg en, reset, D;
wire Q, Qbar;
initial
begin
en=0;
```

```
reset=1; #20;
en=1; #50;
reset=0;
end
initial
begin
D=0; #10;
D=1; #100;
D=0;
end
Dlatch uut (Q, Qbar, D, en, reset);
endmodule
```

Wave Forms :



Source code for SR Flip Flop :

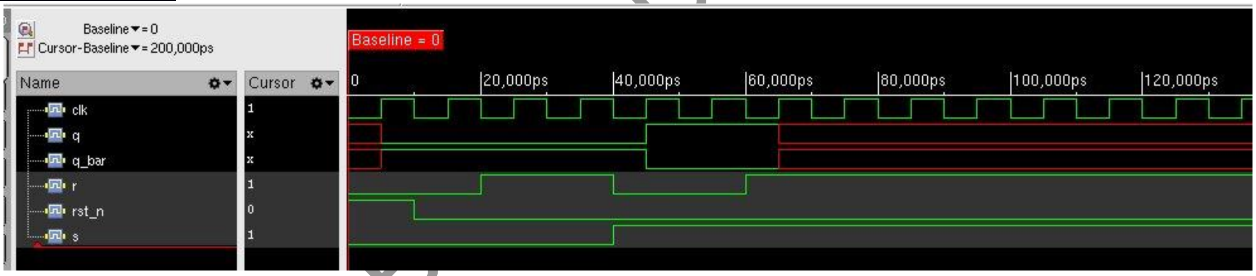
```
 `timescale 1ns/1ps
 module srff (clk, rst_n, s,r, q, q_bar);
   input clk, rst_n;
   input s,r;
   output reg q;
   output q_bar;
     // always@(posedge clk or negedge rst_n)        // for asynchronous reset
   always@(posedge clk)
  begin                                               // for synchronous reset
     if(rst_n)
        q <= 0;
     else
 begin
       case({s,r})
         2'b00: q <= q;           // No change
         2'b01: q <= 1'b0;        // reset
         2'b10: q <= 1'b1;        // set
         2'b11: q <= 1'bx;         // Invalid inputs
       endcase
     end
   end
   assign q_bar = ~q;
  endmodule
```

Test bench :

```
  `timescale 1ns/1ps
  module srff_test;
    reg clk, rst_n;
    reg s, r;
    wire q, q_bar;
    initial begin
     clk = 0;
      rst_n = 1; #10; rst_n = 0;
    end
```

```
   initial begin
  s=0; r=0; #20;
  s=0; r=1; #20;
  s=1; r=0; #20;
  s=1; r=1;
  end
  srff  uut (clk, rst_n, s, r, q, q_bar);
  always #5 clk = ~clk;
 initial
 #1400 $finish;
 endmodule
```

Wave Forms :



Source Code for SR Latch :

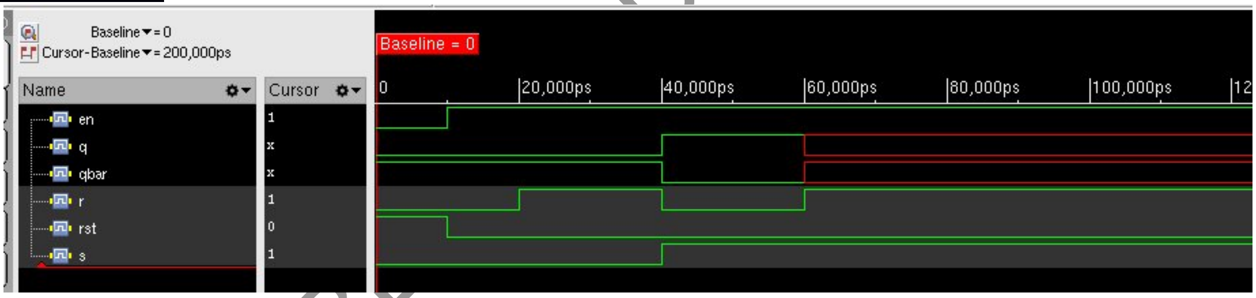
```
module srlatch (q, qbar, s, r, en, rst);
output reg q;
output q;
input s, r, en, rst;
always @ (en or rst or s or r)
      begin
      if (rst)
      q<=0;
      else
if (en)
      begin
      case ({s,r})
      2'b00: q<=q;
      2'b01: q<=1'b0;
      2'b10: q<=1'b1;
      2'b11: q<=1'bx;
      endcase
      end
end
assign qbar=~q;
endmodule
```

Test bench :

```
`timescale 1ns/1ps
module srlatch_test;
reg en, rst;
reg s, r;
wire q, qbar;
initial
      begin
      en=0; rst=1; #10;
      en=1; rst=0; #10;
      rst =0;
```

```
        end
initial begin
s=0; r=0; #20;
s=0; r=1; #20;
s=1; r=0; #20;
s=1; r=1; #20;
end
srlatch uut (q, qbar, s, r, en, rst);
initial
#1400 $finish;
endmodule
```

Wave Forms :



Source Code for JK Flip Flop :

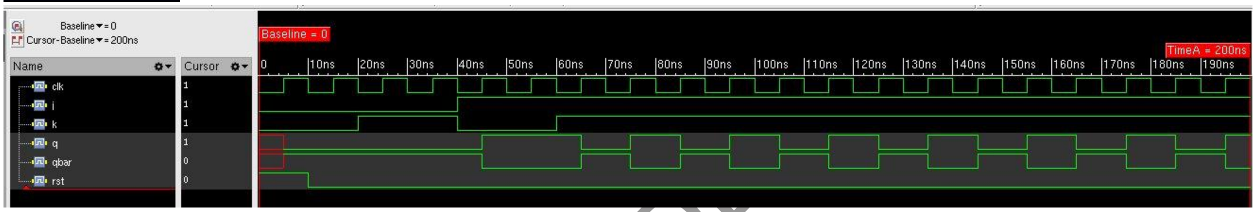
```
module jkff (clk, rst, J, K, q, qbar);
input J, K, clk, rst;
output reg Q;
output qbar;
always @(posedge clk)
        begin
        if (rst)
        q<=0;
        else
                begin
                case ({j,k})
                2'b00: q<= q;
                2'b01: q<= 1'b0;
                2'b10: q<= 1'b1;
                2'b11: q<= ~q;
                endcase
                end
        end
assign qbar= ~q;
endmodule
```

Test bench :

```
Module jkff_test;
reg clk, rst, j, k;
wire q, qbar;
initial begin
clk =0; rst =1; #10;
rst =0;
end
initial begin
j=0; k=0; #20;
j=0; k=1; #20;
j=1; k=0; #20;
```

```
j=1; k=1; #20;
end
jkff  uut (clk, rst, j, k, q, qbar);
always #5 clk =~clk;
endmodule
```

Wave Forms :



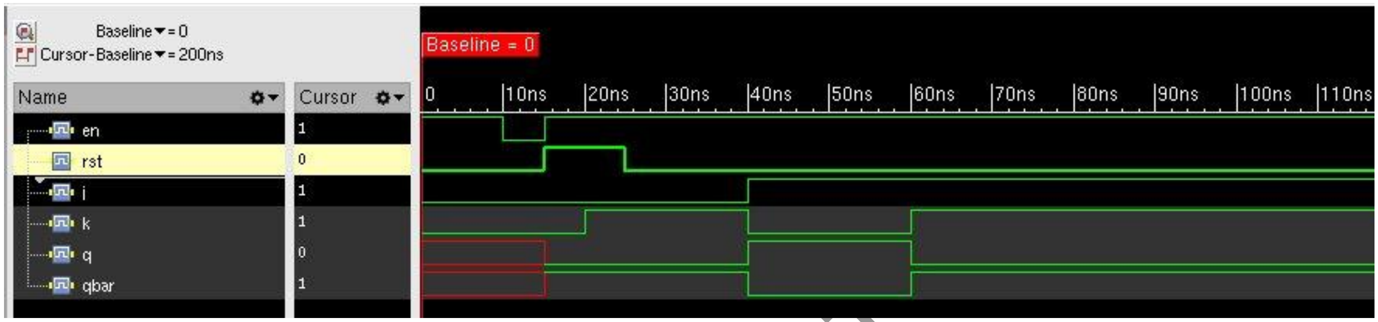
Source of JK Latch :

```
module jkff(en, rst, j, k, q, qbar);
input en, rst, j, k;
output reg q;
output qbar;
always @(en or rst or j or k)
       begin
       if (rst)
       q<=0;
       else
       if (en)
             begin
             case ({j, k})
             2'b00: q<=q;
             2'b01: q<=1'b0;
             2'b10: q<=1'b1;
             2'b11: q<=~q;
             endcase
             end
      end
assign qbar=~q;
endmodule
```

Test bench :

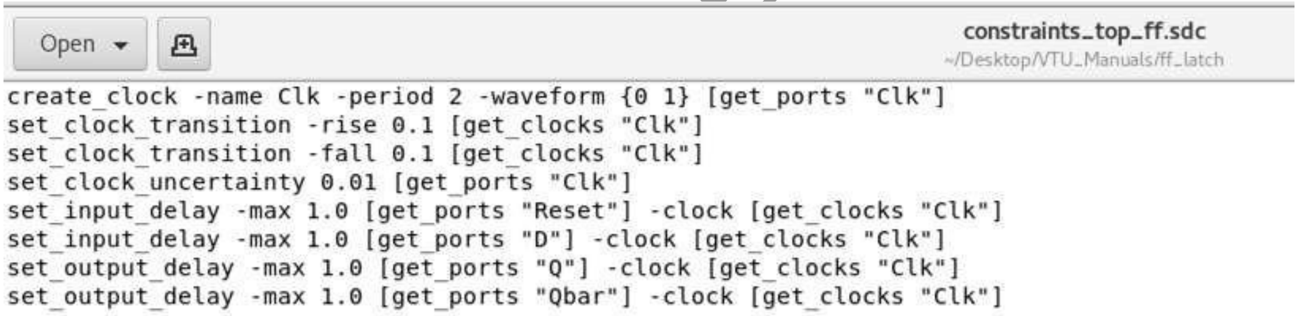
```
Module jklatch_test;
reg en, rst, j, k;
wire q, qbar;
initial begin
en=1; rst=0; #10;
en=0; #5;
en=1;
rst=1; #10;
rst=0;
end
initial begin
j=0; k=0; #20;
j=0; k=1; #20;
j=1; k=0; #20;
j=1; k=1; #20;
end
jklatch uu (en, rst, j, k, q, qbar);
endmodule
```

Wave Forms



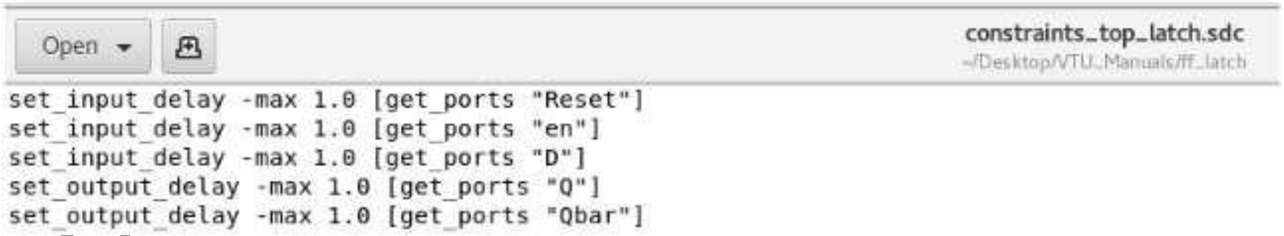
a) Synthesize the Design & Comparing Reports

<u>**Example for SDC:**</u>



```
create_clock -name Clk -period 2 -waveform {0 1} [get_ports "Clk"]
set_clock_transition -rise 0.1 [get_clocks "Clk"]
set_clock_transition -fall 0.1 [get_clocks "Clk"]
set_clock_uncertainty 0.01 [get_ports "Clk"]
set_input_delay -max 1.0 [get_ports "Reset"] -clock [get_clocks "Clk"]
set_input_delay -max 1.0 [get_ports "D"] -clock [get_clocks "Clk"]
set_output_delay -max 1.0 [get_ports "Q"] -clock [get_clocks "Clk"]
set_output_delay -max 1.0 [get_ports "Qbar"] -clock [get_clocks "Clk"]
```

The Above is the example SDC for a D Flip Flop



```
set_input_delay -max 1.0 [get_ports "Reset"]
set_input_delay -max 1.0 [get_ports "en"]
set_input_delay -max 1.0 [get_ports "D"]
set_output_delay -max 1.0 [get_ports "Q"]
set_output_delay -max 1.0 [get_ports "Qbar"]
```

The above is an Example for SDC File for **D Latch**.

<u>Synthesis RTL Schematic :</u>

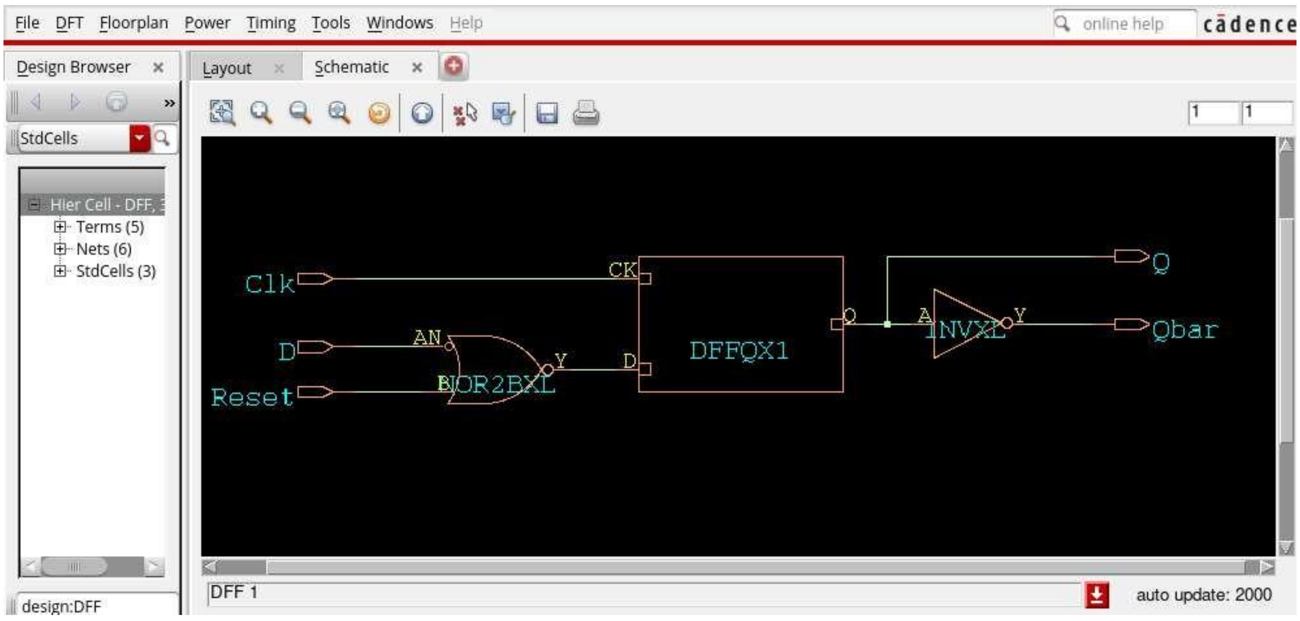


Fig: Schematic of D-Flip Flop

<u>**Note :-**</u>

1. You can tabulate Area, Power and Timing Constraints using any of the SDC Constraints as instructed.
2. Make sure, during synthesis the Report File Names are changed sothat the latest reports do not overwrite the earlier ones.

Custom IC Design Flow (ASIC Analog Design)

PART – B

LAB – 01: CMOS INVERTER

Objective:

(a) Capture the Schematic of a CMOS Inverter with Load Capacitance of 0.1 pF and set the Widths of Inverter with

(i) $W_N = W_P$

(ii) $W_N = 2\ W_P$

(iii) $W_N = W_P / 2$

and Length at selected Technology. Carry out the following:

1. Set the Input Signal to a pulse with Rise Time, Fall Time of 1 ps and Pulse Width of 10 ns, Time Period of 20 ns and plot the input voltage and output voltage of the designed Inverter
2. From the Simulation Results, compute tp_{HL}, tp_{LH} and t_{PD} for all the three geometrical settings of Width
3. Tabulate the results of delay and find the best geometry for minimum delay for CMOS Inverter

Solution:

(i) SCHEMATIC CAPTURE FOR THE CMOS INVERTER

For the CMOS Inverter circuit, PMOS and NMOS transistors are required. The parameters for the devices as given in the requirement are considered as in Table – 1, Table – 2 and Table – 3.

Table – 1: Length and Width of NMOS and PMOS Transistors for the condition WN = WP

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 850 n Length, L = 180 n

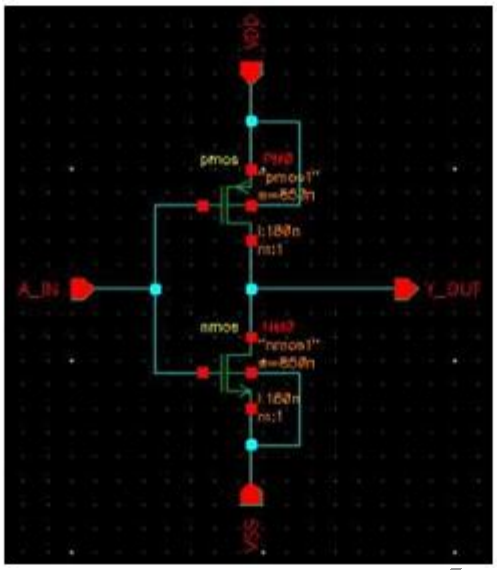
Table – 2: Length and Width of NMOS and PMOS Transistors for the conditionWN = 2 * WP

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, WN = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 1.7 u Length, L = 180 n

Table – 3: Length and Width of NMOS and PMOS Transistors for the conditionWN = WP / 2

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 850 n Length, L = 180 n
gpdk180	Pmos	Width, W_P = 425 n Length, L = 180 n

.



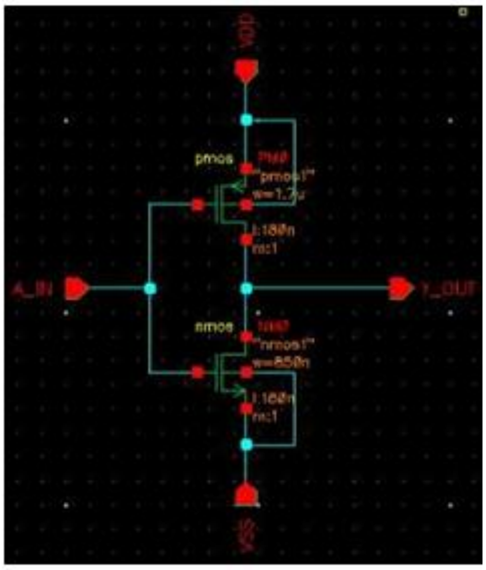


Figure – 1.25(a): WN = WP **Figure – 1.25(b): WN = 2 * WP**

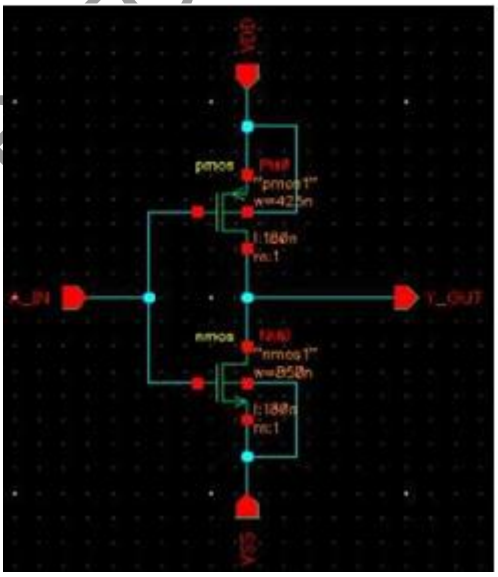


Figure – 1.25(c): WN = WP / 2

SYMBOL CREATION:

.

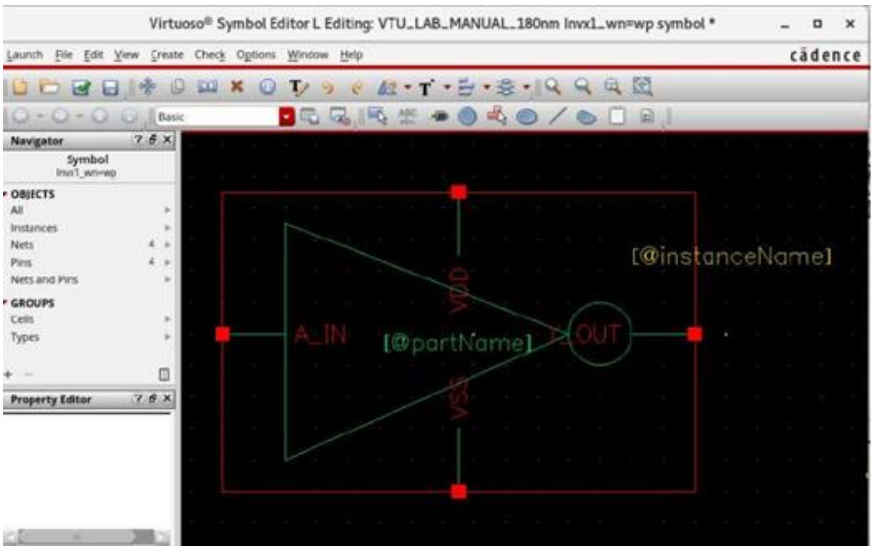


Figure – 1.35: Customized Symbol of an Inverter

(1) TEST CIRCUIT FOR SIMULATION:

The complete circuit after instantiating all the devices and interconnections is shown in Figure – 1.42.

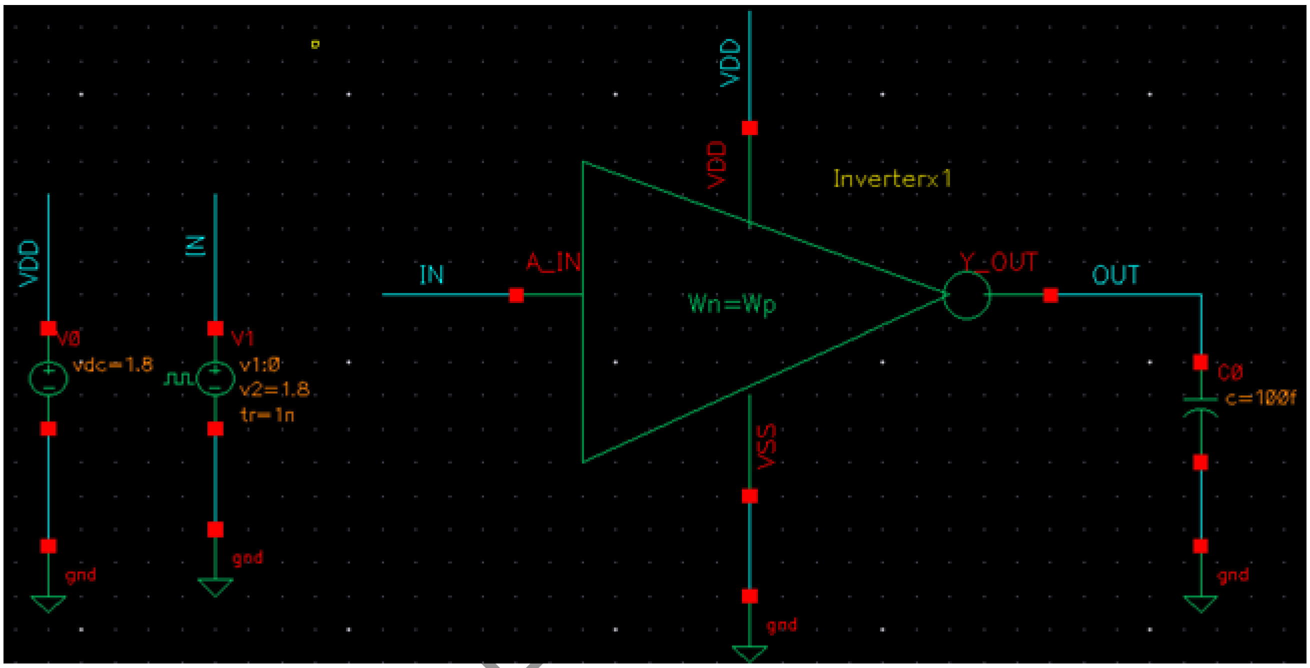


Figure – 1.42: Complete Test Schematic

FUNCTIONAL SIMULATION WITH SPECTRE:

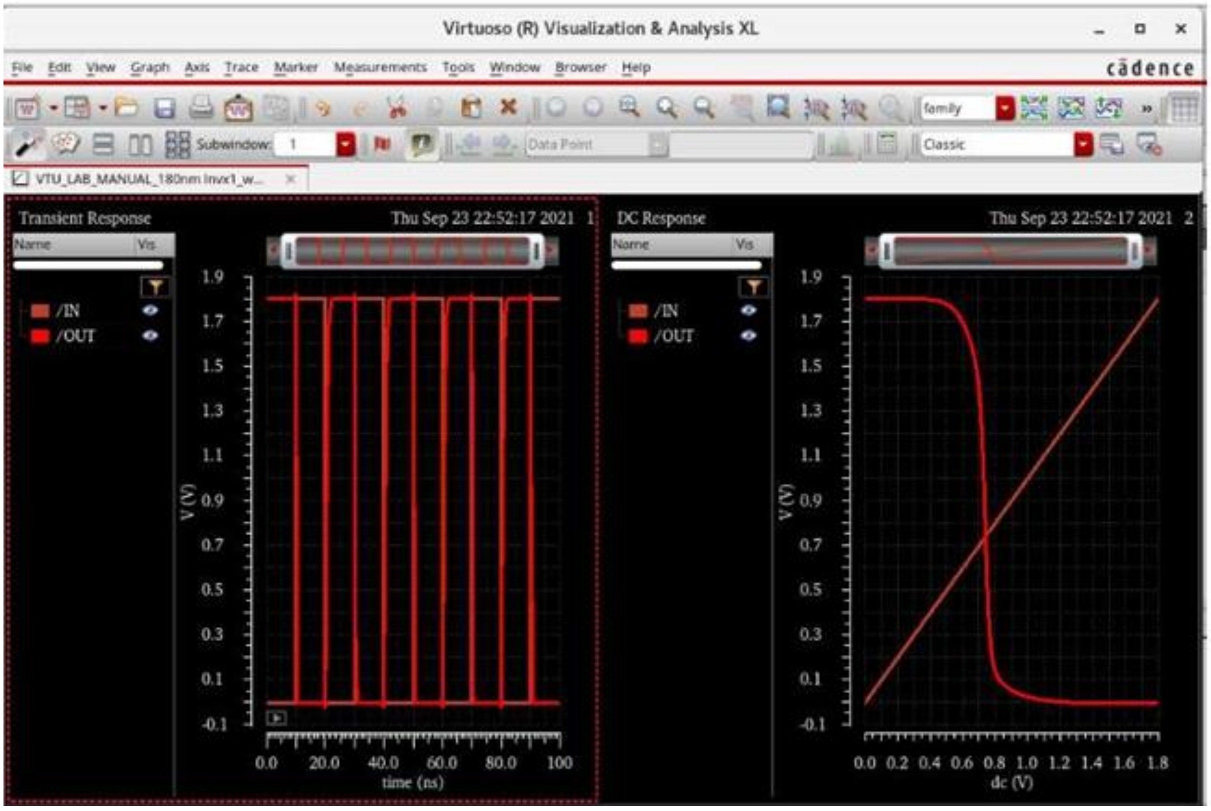

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cadence
Subwindow: 1
Data Point
family
Classic
VTU_LAB_MANUAL_180nm Invx1_w...
Transient Response
Thu Sep 23 22:52:17 2021 1
Name Vis
/IN
/OUT
V (V)
time (ns)
DC Response
Thu Sep 23 22:52:17 2021 2
dc (V)

Figure – 1.70: Simulated waveforms

The Input and Output Signals can be split up by selecting "**Graph** —>**Split All Strips**" as in Figure – 1.71.

(1) CALCULATION OF tpHL, tpLH AND tPD:

To calculate the Propagation Delay (t_{PD}), the formula used is

$$tPD = \frac{(tpLH + tpHL)}{2}$$

where, tp_{LH} —> Low – High Propagation Delay and tp_{HL} —> High – Low Propagation Delay.

To calculate tp_{LH} and tp_{HL} use the Calculator option from the "**Virtuoso (R) Visualization and Analysis**" window. So, select "**Tools** —> **Calculator**" or click on the icon as shown in Figure – 1.78.

(2) TABULATED VALUES OF DELAY:

The results of tp_{HL}, tp_{LH} and t_{PD} for all the required geometrical settings are tabulated belowin Table – 5.

Table – 5: Values of tp_{HL}, tp_{LH} and t_{PD} for different geometries

Width Settings	MOSFET	Width	tpLH	tpHL	tpd
Wp =Wn	PMOS	850n	3.233E-10	7.049E-10	5.141E-10
	NMOS	850n			
Wn =2Wp	PMOS	850n	3.337E-10	4.700E-10	4.019E-10
	NMOS	1.7u			
Wn = Wp/2	PMOS	425n	1.141E-09	3.154E-10	7.282E-10
	NMOS	850n			

(a) Layout of CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$

Objective:

To draw the Layout of CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$ using optimum Layout Methods. Verify for DRC and LVS, extract the Parasitics and perform the Post-Layout Simulations, comparethe results with Pre-Layout Simulations and record the observations.

SCHEMATIC CAPTURE:

Create a New Library, Create a Cellview and instantiate the required devices through "Create Instance" option. The parameter for PMOS and NMOS Transistors are listed in Table – 6 shown below.

Table – 6: Parameters for NMOS and PMOS Transistors

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 20 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 40 u Length, L = 180 n

Follow the techniques demonstrated in Lab – 01 to complete the Schematic. The completed CMOS Inverter circuit is shown in Figure – 1.85.

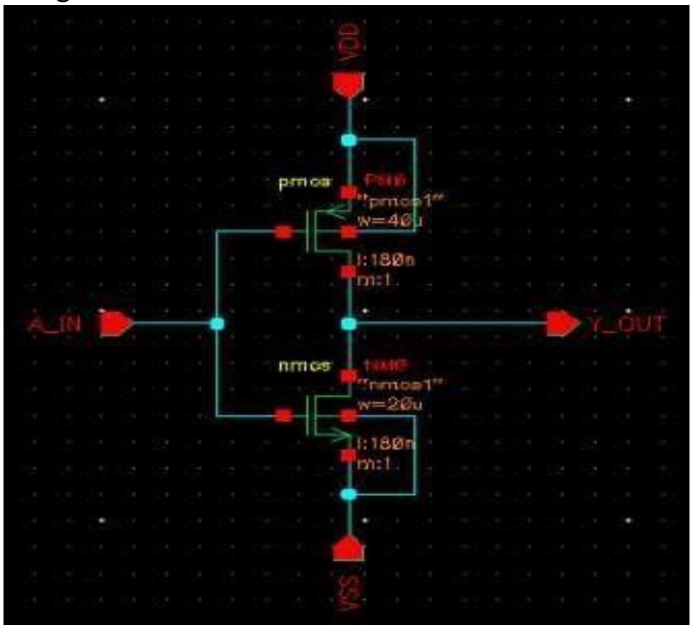


Figure – 1.85: Schematic for CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$

The symbol for the CMOS Inverter is shown in Figure – 1.86.

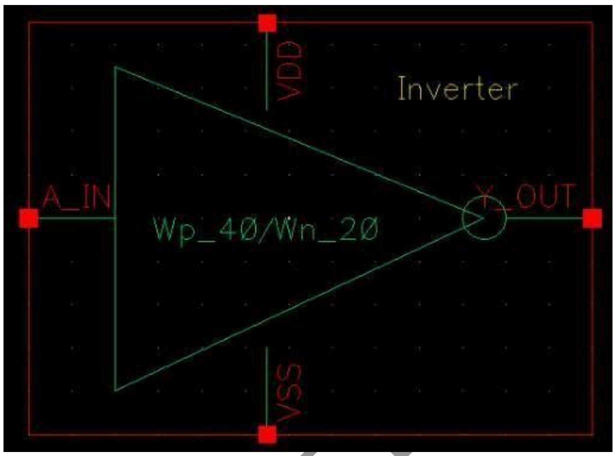


Figure – 1.86: Symbol for CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$

SIMULATION:

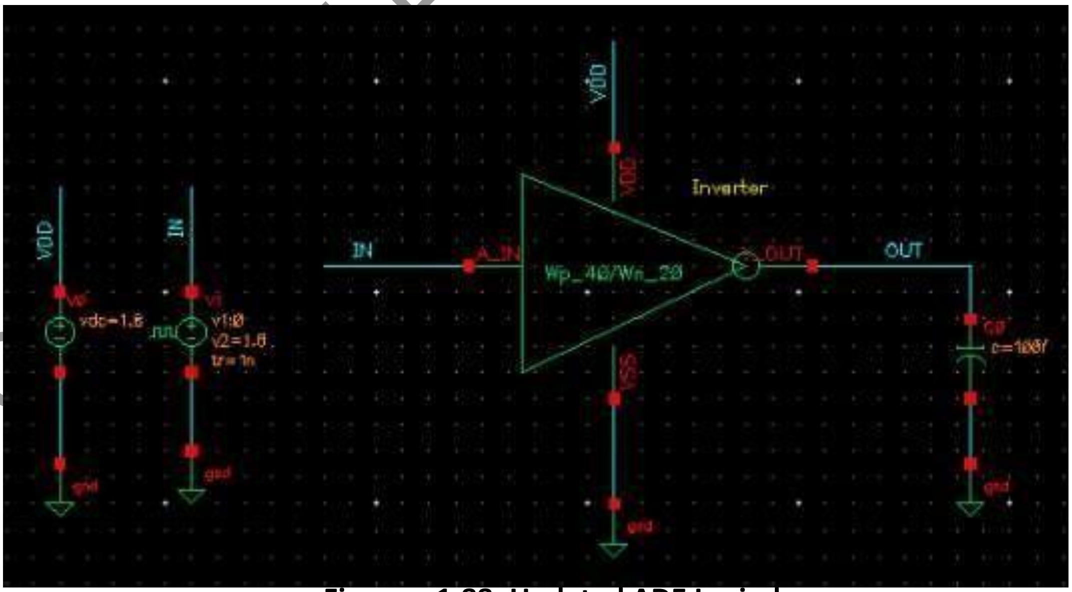


Figure – 1.88: Updated ADE L window

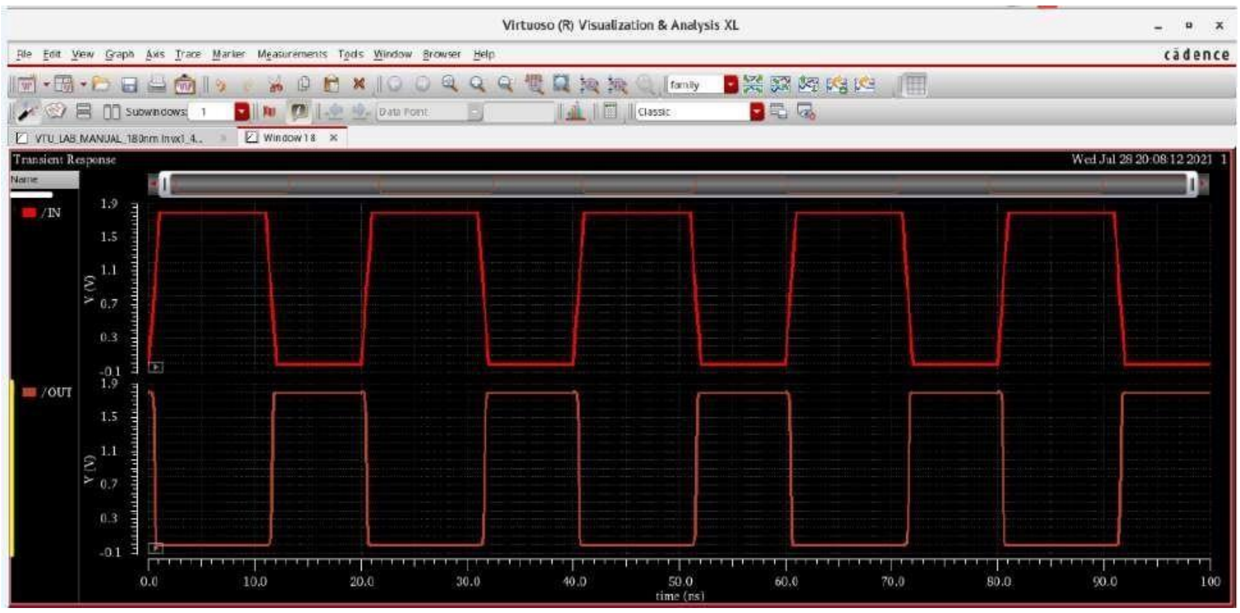


Figure – 1.89: Transient Analysis for CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$

Similarly, the signals plotted after DC Analysis are shown in Figure – 1.90.

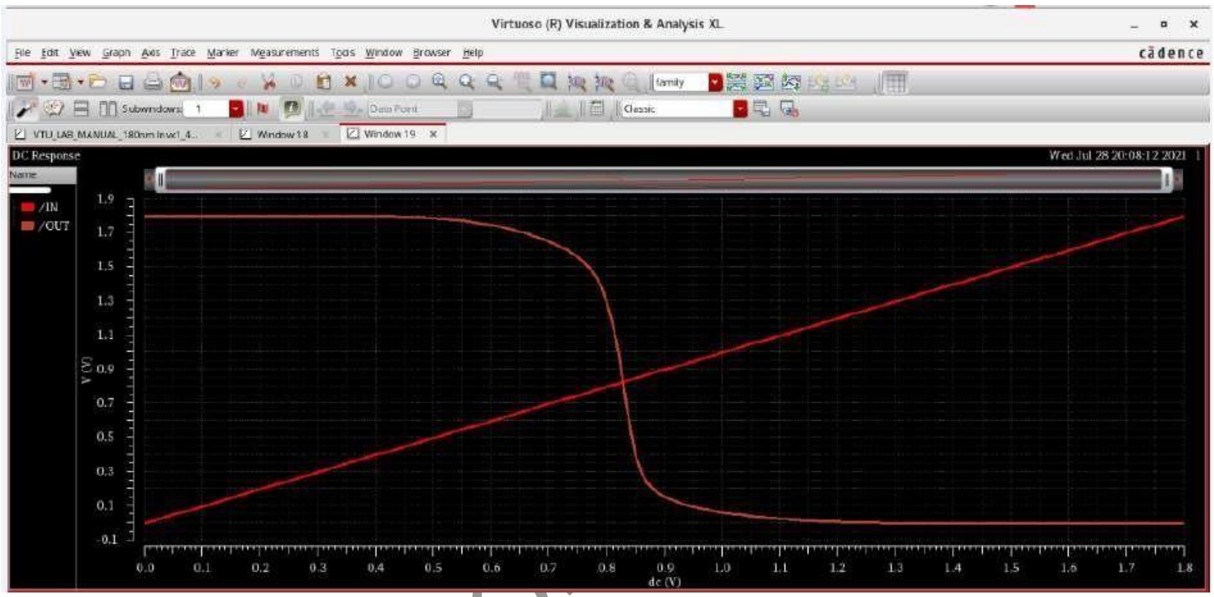

Virtuoso (R) Visualization & Analysis XL
cādence
File Edit View Graph Axes Trace Marker Measurements Tools Window Browser Help
VTU_LAB_MANUAL_180nm Invx1_4...
Window 18
Window 19
DC Response
Wed Jul 28 20:08:12 2021
/IN
/OUT
V (V)
dc (V)

Figure – 1.90: DC Analysis for CMOS Inverter with $\frac{W_p}{W_n} = \frac{40}{20}$

VALUES OF tp_{HL}, tp_{LH} AND t_{PD}:

Obtain the values of tp_{HL}, tp_{LH} and t_{PD} by referring to "**CALCULATION OF tpHL, tpLH AND tPD**" in the previous section. The values are tabulated as shown in Table – 7

Table – 7: Values tp_{HL}, tp_{LH} **AND** ***for CMOS Inverter with*** $\frac{W_p}{W_n} = \frac{40}{20}$

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	1.228E-10	3.550E-11	7.920E-11
NMOS	180n	20u			

LAB – 02: 2 – INPUT CMOS NAND GATE

Objective:

(a) Capture the Schematic of a 2 – input CMOS NAND Gate having similar delay as that of CMOS Inverter computed in Lab – 01. Verify the functionality of the NAND Gate and also find out the delay for all the four possible combinations of input vectors. Tabulate the results. Increase the drive strength to 2X and 4X and tabulate the results.

(b) Draw the layout of NAND with $\frac{W_p}{W_n} = \frac{40}{20}$ use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare theresults with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATIC CAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library using the option "File —> New —> Library", create a New Cell View upon selecting the newly created library using the option "File —> New —> Cell View" and instantiate the required devices using the "Create —> Instance" option.

The device parameters are listed in Table – 9.

Table – 9: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gate

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 1.7 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 1.275 u Length, L = 180 n

Similarly, the device parameters for the 2 – input CMOS NAND Gate with drive strength 2 and drive strength 4 are listed in Table – 10 and Table – 11.

Table – 10: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gate *with Drive Strength "2"*

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 3.4 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 2.55 u Length, L = 180 n

Table – 11: Width and Length of NMOS and PMOS Transistors for CMOS NAND Gate *with Drive Strength "4"*

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 6.8 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 5.1 u Length, L = 180 n

The completed Schematic for all the three dimensions are shown in Figure – 2.1, Figure – 2.2 and Figure – 2.3.

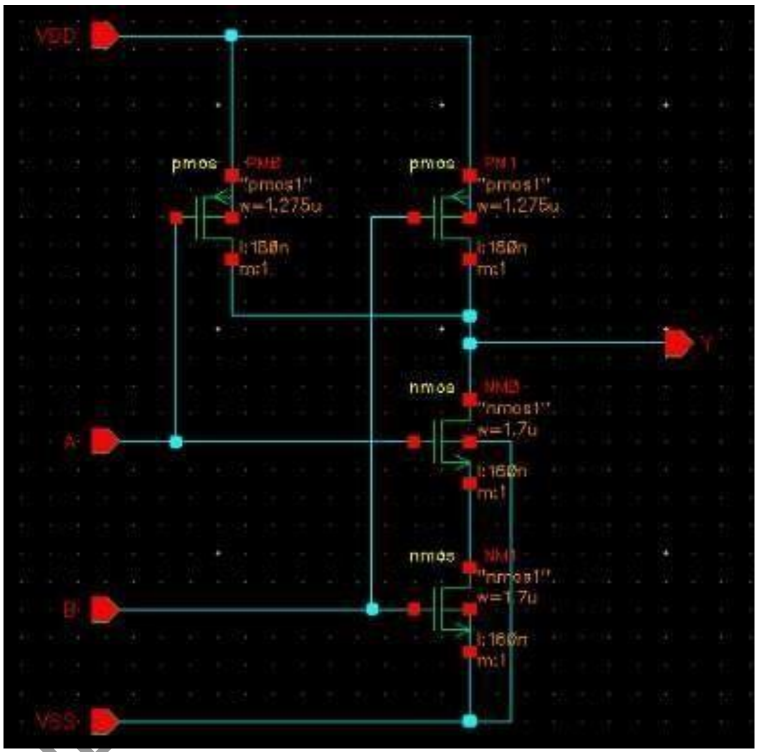


Figure – 2.1: Schematic Capture of 2 – input CMOS NAND Gate (NAND2X1)

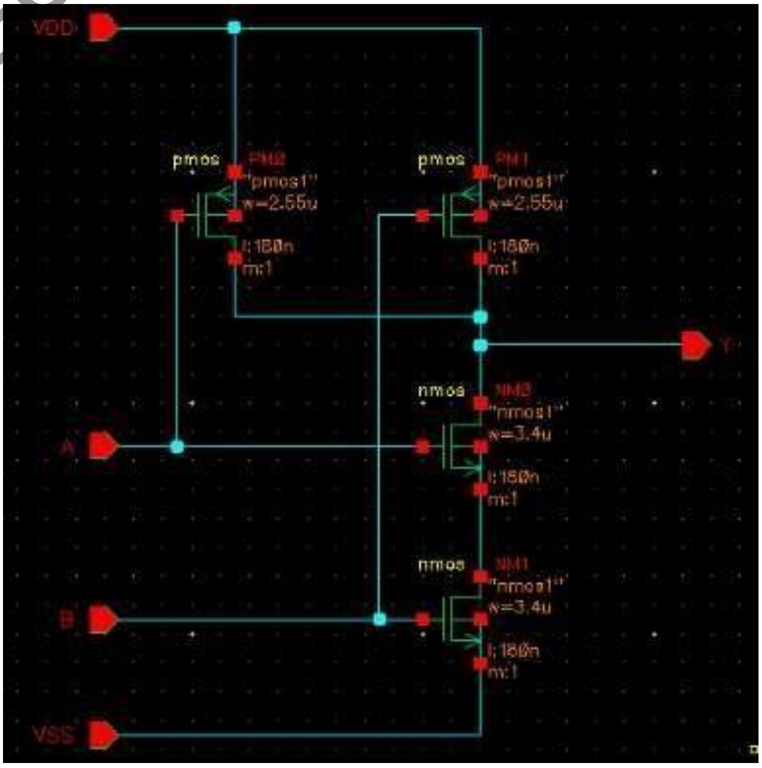


Figure – 2.2: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2(NAND2X2)

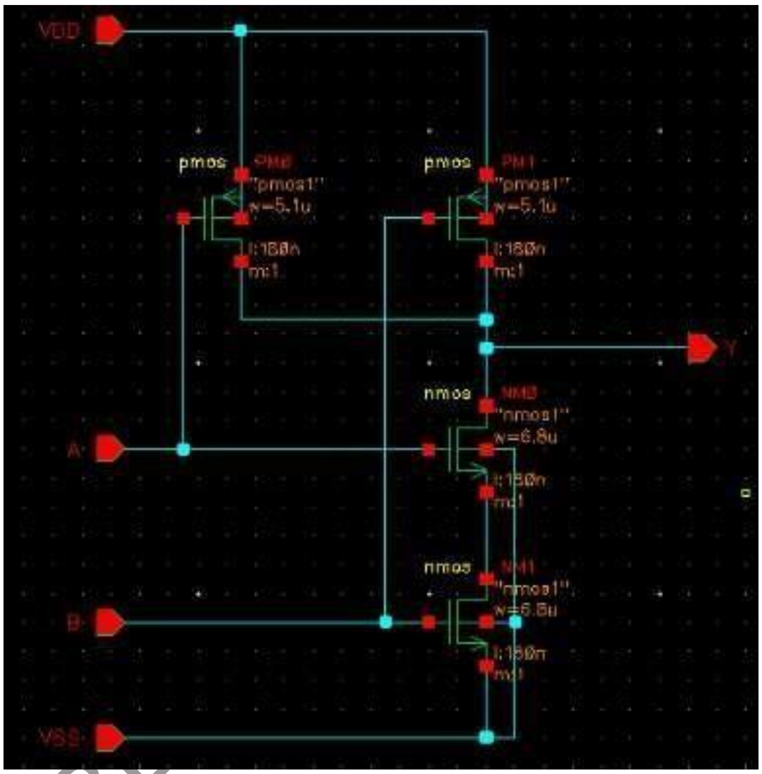


Figure – 2.3: Schematic Capture of 2 – input CMOS NAND Gate with drive strength 2(NAND2X4)

The symbol for the CMOS NAND Gate is shown in Figure – 2.4.

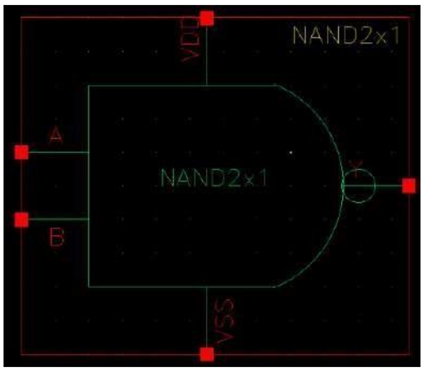


Figure – 2.4: Symbol of 2 – input NAND Gate

FUNCTIONAL SIMULATION:

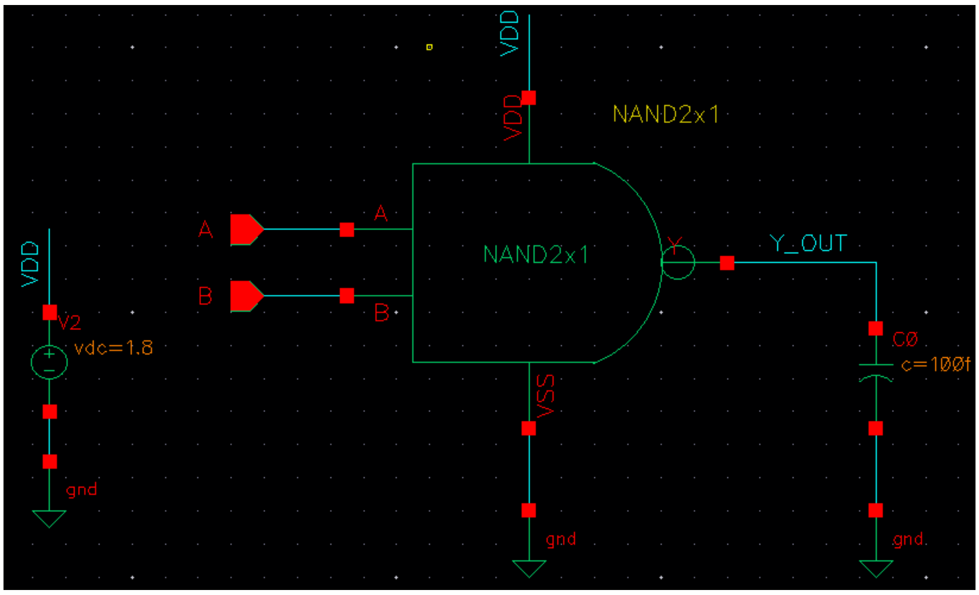


Figure – 2.5: Test Schematic for 2 – input CMOS NAND Gate

The Simulated waveforms can be seen as shown in Figure – 2.9.

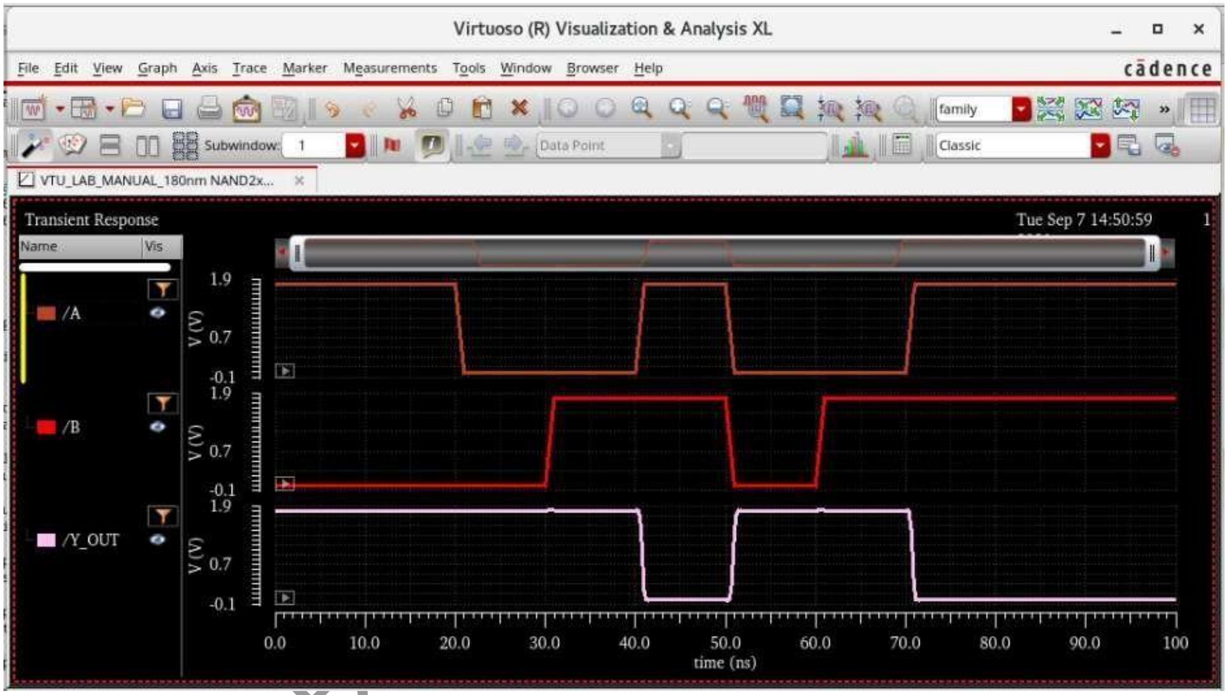


Figure – 2.9: Transient Analysis of 2 – input CMOS NAND Gate

The delay values are obtained using the "**Calculator**" option as demonstrated in Lab – 01. Theresults are tabulated as shown in Table – 12.

Table – 12: Values of Delay for 2 – input CMOS NAND2X1, NAND2X2 and NAND 2X4

NAND Type	MOSFET	Length	Width	tpLH	tpHL	tpd
NAND2X1	PMOS	180n	1.5 * 850n = 1.275u	3.720E-10	3.340E-10	3.530E-10
	NMOS		2 * 850n = 1.7u			
NAND2X2	PMOS	180n	1.5 * 850n * 2 = 2.55u	2.610E-10	2.200E-10	2.400E-10
	NMOS		2 * 850n * 2 = 3.4u			
NAND2X4	PMOS	180n	1.5 * 850n * 4 = 5.1u	1.900E-10	1.650E-10	1.780E-10
	NMOS		2 * 850n * 4 = 6.8u			

Solution – (b):

SCHEMATIC CAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library, a New Cell View and instantiate the devices as per the Schematic of 2 – input CMOS NAND Gate. The device parameters for the NMOS and PMOS Transistors are listed in Table – 13.

Table – 13: Device parameters for 2 – input CMOS NAND Gate with $\frac{W_p}{W_n} = \frac{40}{20}$

Library Name	Cell Name	Comments / Properties
gpdk180	Nmos	Width, W_N = 20 u Length, L = 180 n
gpdk180	Pmos	Width, W_P = 40 u Length, L = 180 n

The Schematic as per the dimensions of NMOS and PMOS transistors listed above is shown in Figure – 2.10.

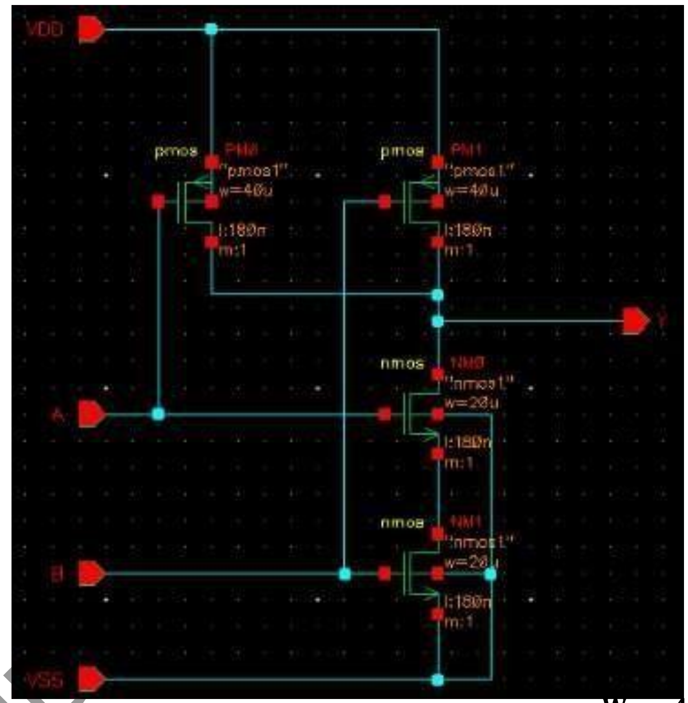

VDD
pmos PM0 "pmos1" w=40u l:180n m:1
pmos PM1 "pmos1" w=40u l:180n m:1
Y
nmos NM0 "nmos1" w=20u l:180n m:1
A
nmos NM1 "nmos1" w=20u l:180n m:1
B
VSS

Figure – 2.10: Schematic for 2 – input CMOS NAND with $\frac{W_p}{W_n} = \frac{40}{20}$

Symbol for the Schematic in Figure – 2.10 is shown in Figure – 2.11.

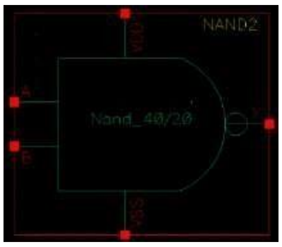


Figure – 2.11: Symbol for 2 – input CMOS NAND with $\frac{W_p}{W_n} = \frac{40}{20}$

FUNCTIONAL SIMULATION:

The Test Schematic for the functionality check of the 2 – input CMOS NAND Gate is shownin Figure – 2.12.

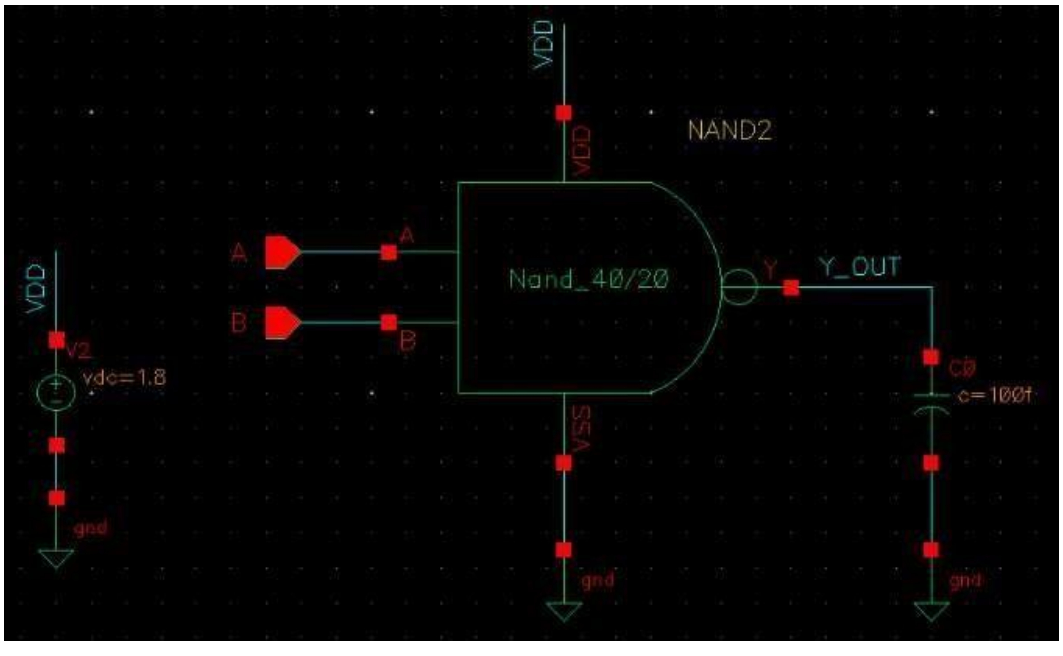


Figure – 2.12: Test Schematic for 2 – input CMOS NAND with $\frac{W_p}{W_n} = \frac{40}{20}$

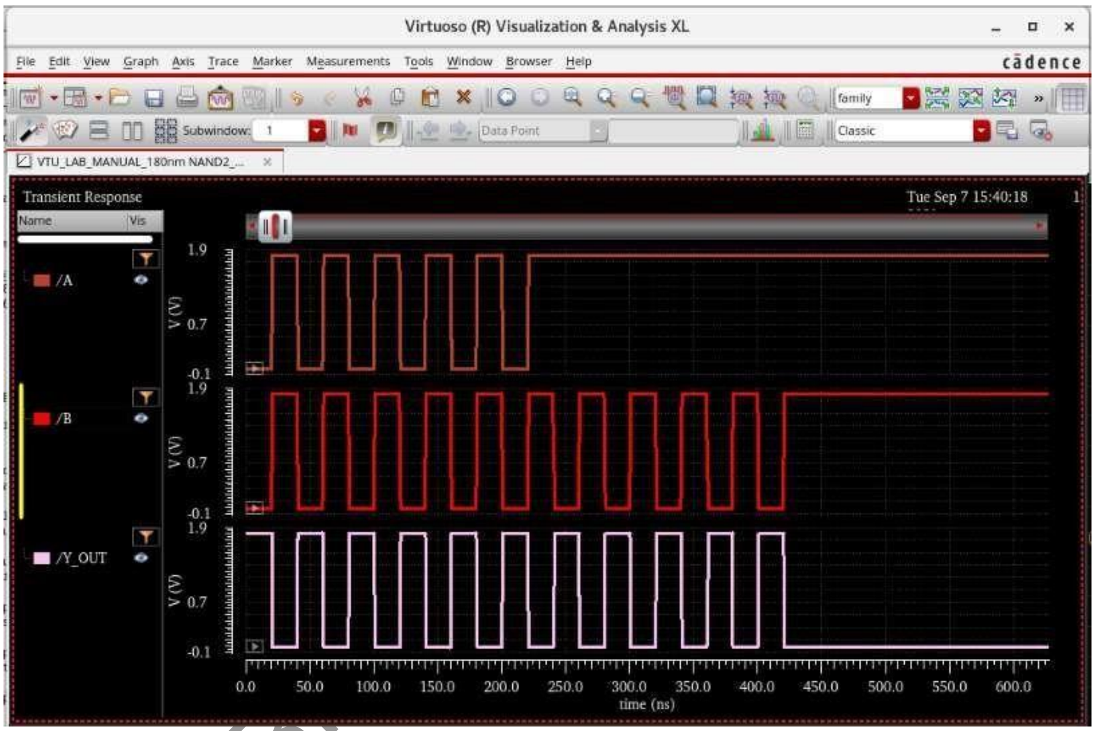


Figure – 2.14: Simulated Waveforms for 2 – input CMOS NAND with $\frac{W_p}{W_n} = \frac{40}{20}$

The values of delay elements are tabulated in Table – 14.

Table – 14: Delay Elements for 2 – input CMOS NAND Gate with $\frac{W_p}{W_n} = \frac{40}{20}$ (Pre Layout *Simulation)*

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	3.64E-11	1.55E-10	9.57E-11
NMOS	180n	20u			

LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devicesfrom the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 2.15.

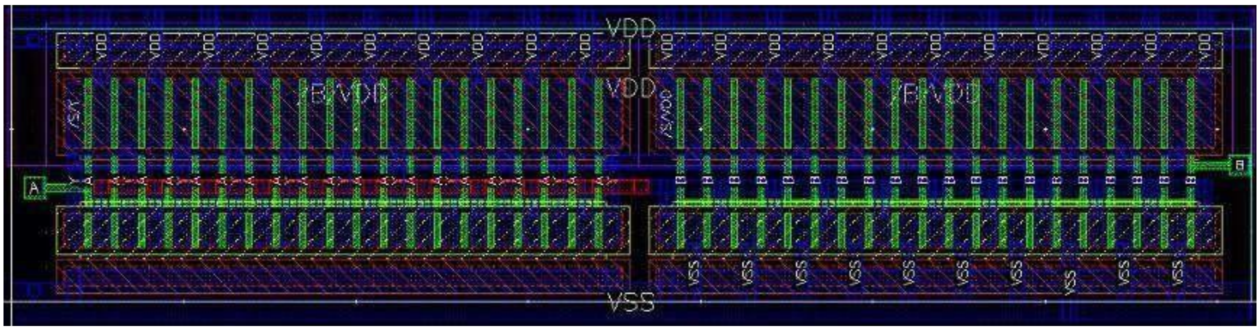


Figure – 2.15: Layout for 2 – input CMOS NAND Gate with $\frac{W_p}{W_n} = \frac{40}{20}$

DRC:

To check for the DRC violations, browse the "**assura_tech.lib**" file, select "**Assura —> Run DRC**", verify the Layout Design Source, mention a "**Run Name**", select "**Technology —> gpdk180**" and click on "**OK**" as demonstrated in Lab – 01.

LVS:

To check for the LVS violations, select “**Assura** —> **Run LVS**”, verify the Schematic Design Source and the Layout Design Source, mention a “**Run Name**”, select “**Technology** —> **gpdk180**” and click on “**OK**” as demonstrated in Lab – 01.

QRC:

To extract the Parasitics, select “**Assura** —> **Quantus**”, select “**Technology** —> **gpdk180**”,“**Output** —> **Extracted View**” from the “**Setup**” option, select “**Extraction Type** —> **RC**” and“**Ref Node** —> **VSS**” from the “**Extraction**” and click on “**OK**” as demonstrated in Lab – 01. The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

The values of delay are shown in Table – 15.

Table – 15: Delay Elements for 2 – input CMOS NAND Gate with $\frac{W_p}{W_n} = \frac{40}{20}$ (Post Layout ***Simulation)***

MOSFET	Length	Width	tpLH	tpHL	tpd
PMOS	180n	40u	3.64E-11	1.55E-10	9.57E-11
NMOS	180n	20u			

LAB – 03: COMMON SOURCE AMPLIFIER WITH PMOS CURRENT MIRROR LOAD

Objective:

(a) Capture the Schematic of a Common Source Amplifier with PMOS Current Mirror Load and find its Transient Response and AC Response. Measure the UGB and Amplification Factor by varying transistor geometries, study the impact of variation in width to UGB.

(b) Draw the layout of Common Source Amplifier, use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATICCAPTURE:

Following the techniques demonstrated in Lab – 01, Create a New Library using the option“ File →New → Library”, create a New Cell View upon selecting the newly created library using the option “File → New → Cell View” and instantiate the required devices using the “Create → Instance” option.

The device parameters are listed in Table–16.

Table –16: Width and Length of NMOS and PMOS Transistors

LibraryName	CellName	Comments/Properties
gpdk180	Nmos	Width,W_N =6 u Length, L=180 n
gpdk180	Pmos	Width, W_P= 8.85u Length, L=180 n

The completed Schematic is shown in Figure–3.1.

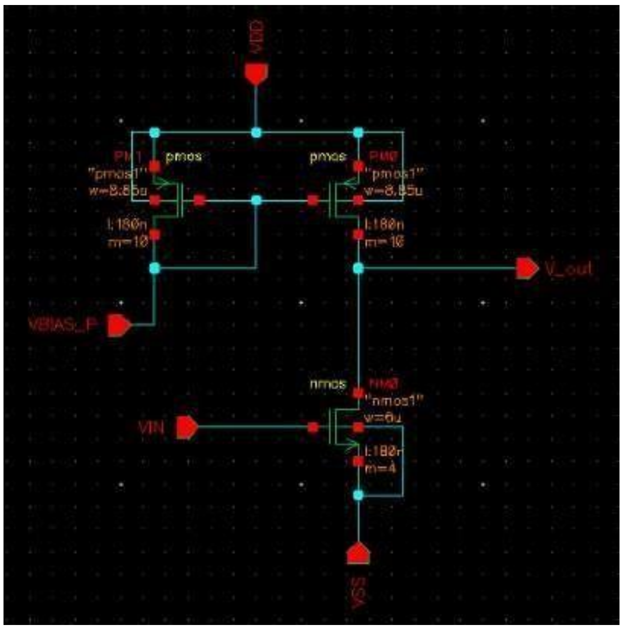


Figure – 3.1: Schematic of Common Source Amplifier with PMOS Current Mirror Load

The symbol for the Common Source Amplifier with PMOS Current Mirror Load is shown in Figure – 3.2.

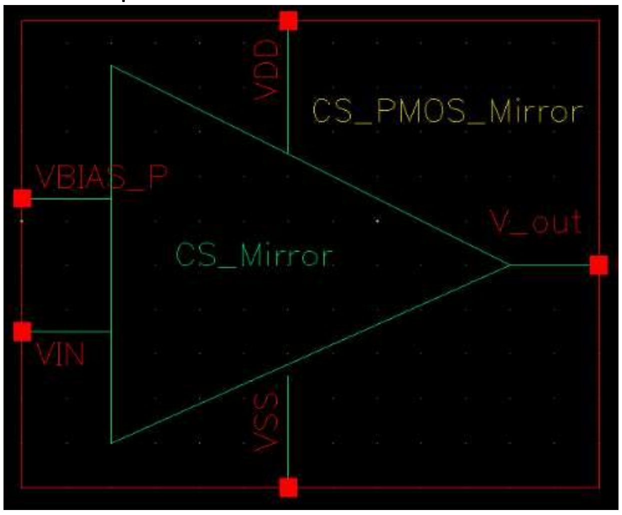


Figure – 3.2: Symbol of Common Source Amplifier with PMOS Current Mirror Load

FUNCTIONAL SIMULATION:

Using the symbol created, build the Test Schematic. Create a New Cell View, instantiate the symbol of Common Source Amplifier with PMOS Current Mirror Load, DC Voltage Source, Current Source, AC Voltage Source, Capacitance, Resistance and Ground, connect the using wires

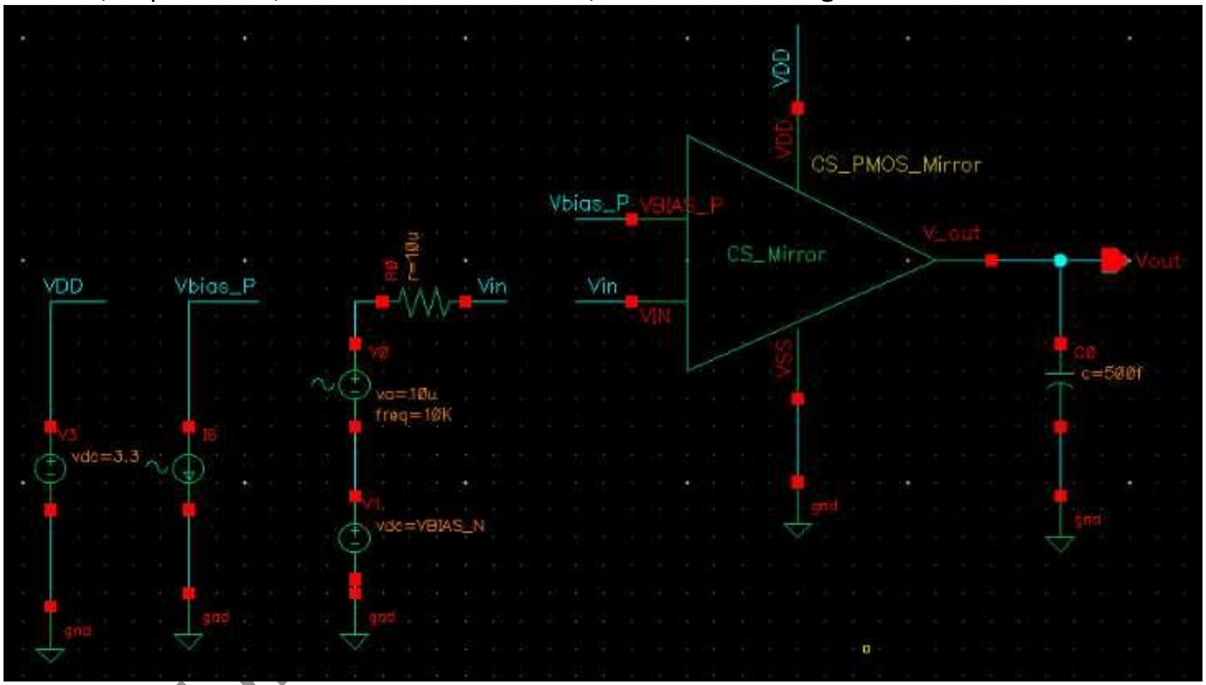


Figure – 3.3: Test Schematic for 2 – input CMOS NAND Gate

The parameters for remaining devices are shown in Table – 17.

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = 3.3 V (VDD)
analogLib	vdc	DC voltage = VBIAS_N V (Vin)
analogLib	isin	DC current = 100u A (Vbias_P)
analogLib	vsin	AC Magnitude = 1 V, Amplitude = 10u V, Frequency = 10K Hz (Vin)
analogLib	cap	Capacitance = 500f F
analogLib	res	Resistance = 10u Ohms
analogLib	gnd	

Table – 17: Parameters for the devices used in Test Schematic

The Simulated waveforms can be seen as shown in Figure – 3.5 and Figure – 3.6.

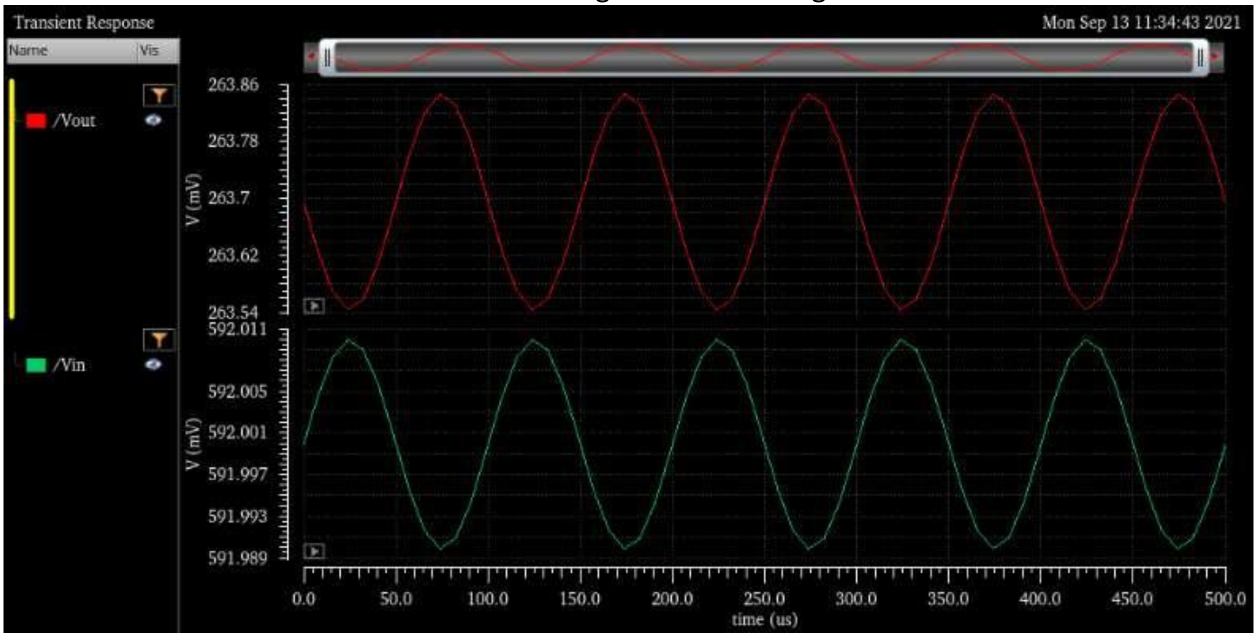


Figure – 3.5: Transient Analysis

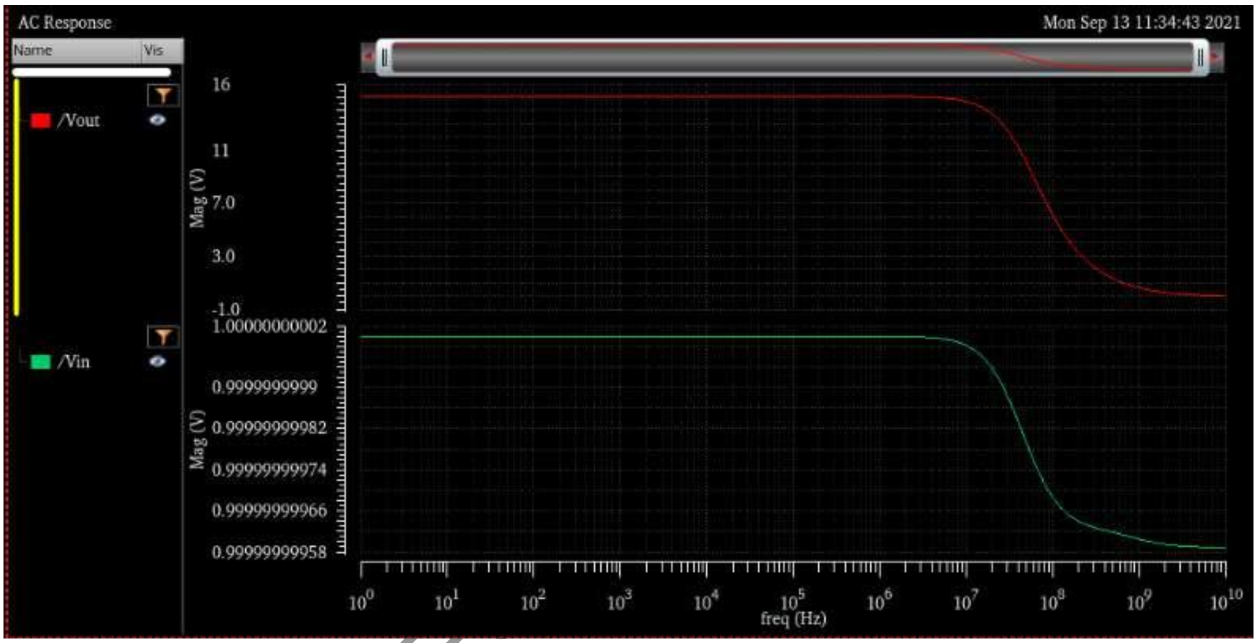


Figure – 3.6: AC Analysis

To measure the Gain and Unity Gain Bandwidth, go back to the ADE L window, select "**Results → Direct Plot → AC Magnitude & Phase**" as shown in Figure – 3.7.

The Test Schematic window pops up, select the output net as shown in Figure – 3.8 and click on "**Esc**" key on the keyboard.

The waveform can be seen as shown in Figure – 3.9. The marker placed on the low frequency part of the response gives the DC Gain, use the bind key "**M**" to place the marker.

Place a horizontal cursor at "**0 dB**" and the crossing frequency gives the Unity Gain Bandwidth (UGB) as shown in Figure 3.9.

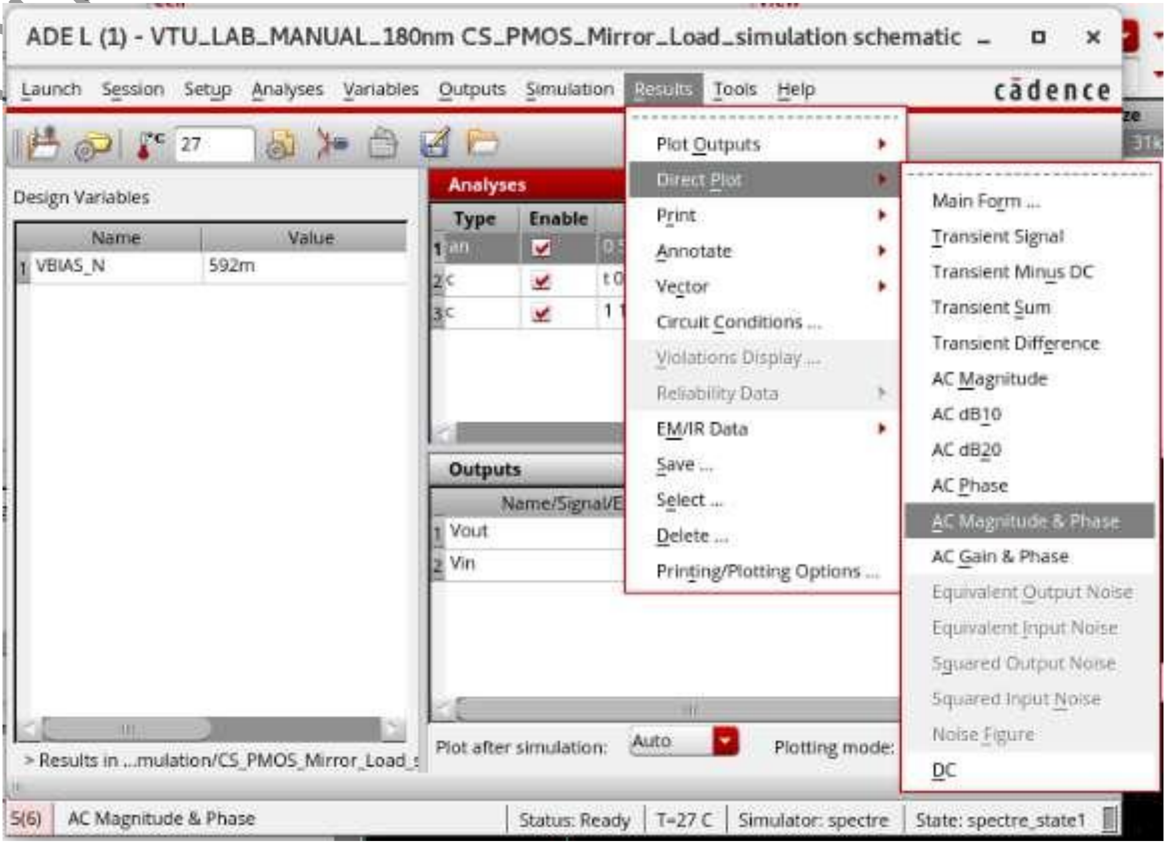


Figure – 3.7: Results → Direct Plot → AC Magnitude & Phase

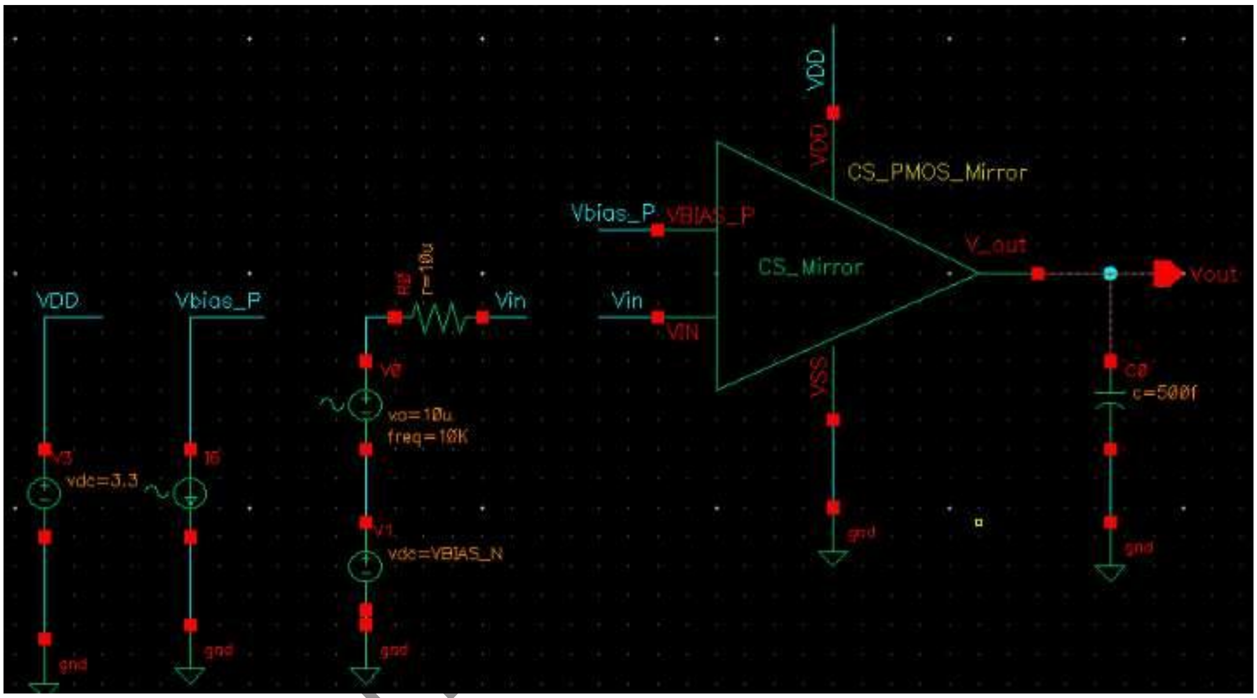


Figure – 3.8: Selecting Output Net from the Test Schematic

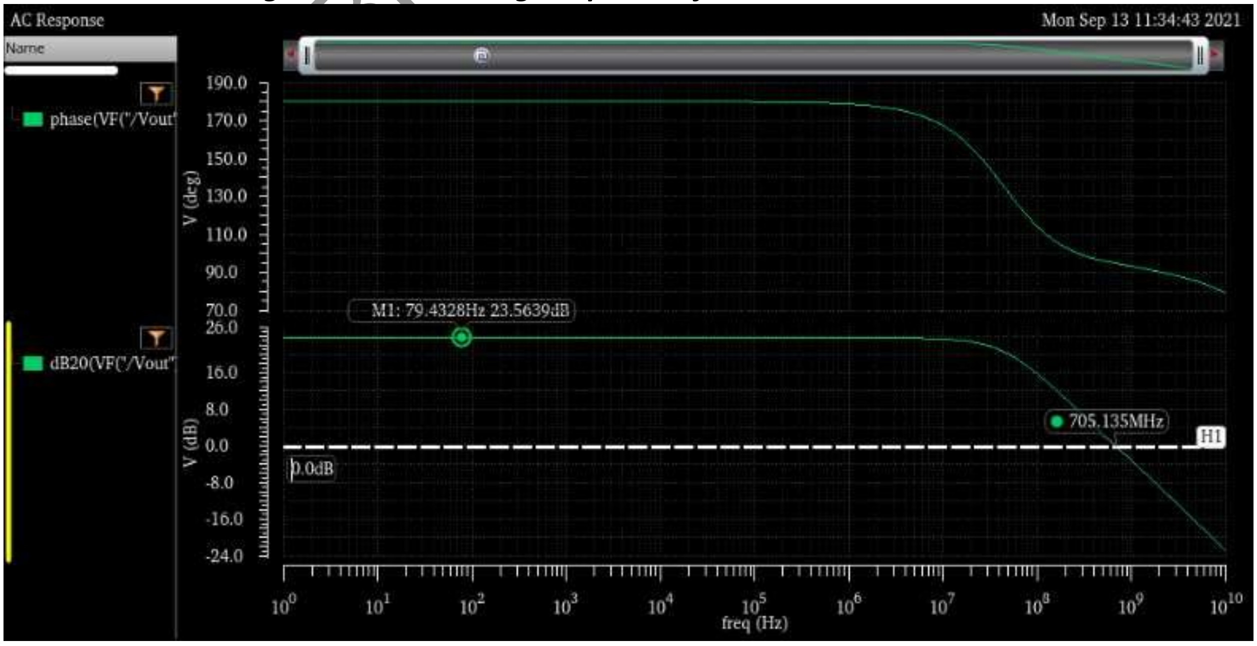


Figure – 3.9: Gain and Phase plot

Solution – (b): LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devices from the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 3.10.

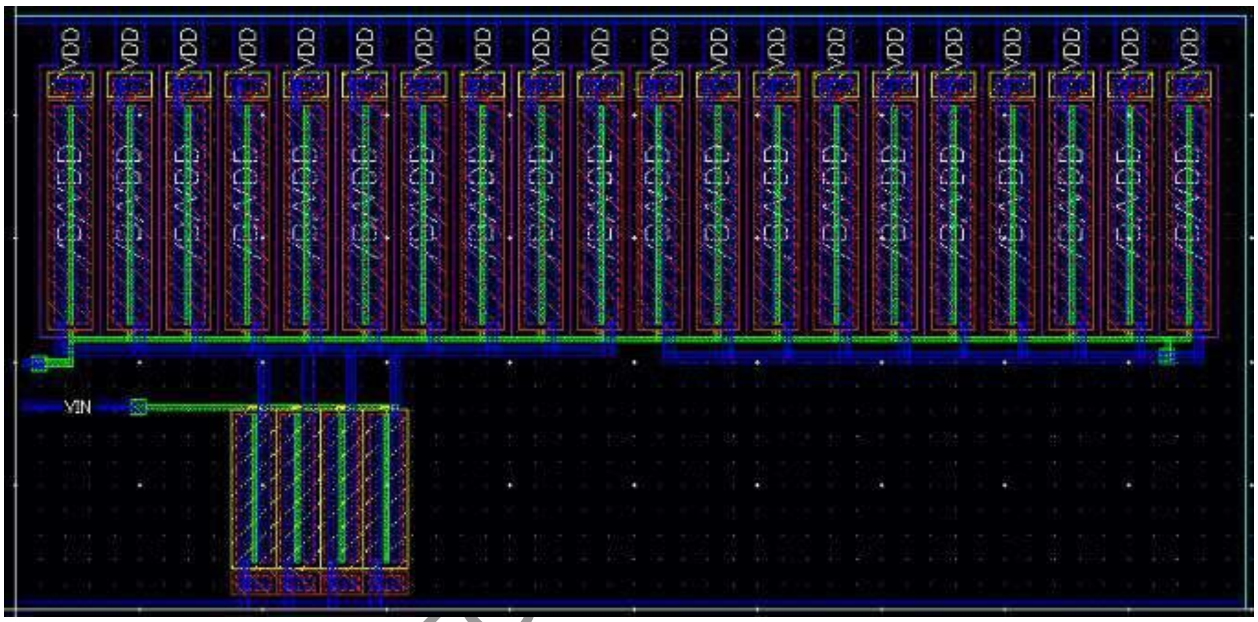


Figure – 3.10: Layout for Common Source Amplifier with PMOS Current Mirror Load

DRC:

To check for the DRC violations, browse the "**assura_tech.lib**" file, select "**Assura → Run DRC**", verify the Layout Design Source, mention a "**Run Name**", select "**Technology → gpdk180**" and click on "**OK**" as demonstrated in Lab – 01.

LVS:

To check for the LVS violations, select "**Assura → Run LVS**", verify the Schematic Design Source and the Layout Design Source, mention a "**Run Name**", select "**Technology → gpdk180**" and click on "**OK**" as demonstrated in Lab – 01.

QRC:

To extract the Parasitics, select "**Assura → Quantus**", select "**Technology → gpdk180**", "**Output → Extracted View**" from the "**Setup**" option, select "**Extraction Type → RC**" and "**Ref Node → VSS**" from the "**Extraction**" and click on "**OK**" as demonstrated in Lab – 01. The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

LAB – 04: 2 – STAGE OPERATIONAL AMPLIFIER

Objective:

a) Capture the Schematic of a 2 – Stage Operational Amplifier and measure the following:

1. UGB
2. dB Bandwidth
3. Gain Margin and Phase Margin with and without coupling capacitance
4. Use the Op-Amp in the Inverting and Non-Inverting configuration and verify its functionality
5. Study the UGB, 3 dB Bandwidth, Gain and Power Requirement in Op-Amp by varying the stage wise transistor geometries and record the observations

b) Draw the layout of 2 – stage Operational Amplifier with the maximum transistor width set to 300 (in 180 / 90/ 45n m Technology), choose appropriate transistor geometries as per the results obtained in 4(a). Use optimum layout methods. Verify DRC and LVS, extract the parasitics and perform the post layout simulation, compare the results with pre layout simulations. Record the observations.

Solution – (a):

SCHEMATIC CAPTURE:

Table – 18: Device Parameters for 2 – Stage Operational Amplifier

Library Name	Transistor	Cell Name	Comments / Properties
gpdk045	M0, M1	pmos2v	Width, W = 465 n Length, L = 150 n
gpdk045	M3, M4	nmos2v	Width, W = 490 n Length, L = 150 n
gpdk045	M5, M7	nmos2v	Width, W = 1.09 u Length, L = 150 n
gpdk045	M2	pmos2v	Width, W = 10 u Length, L = 150 n
gpdk045	M6	nmos2v	Width, W = 6.88 u Length, L = 150 n
gpdk045	M8	pmoscap2v	Calculated Parameter = Capacitance Capacitance = 250.043 f

The completed Schematic as per the dimensions mentioned in Table – 18 is shown in Figure – 4.2.

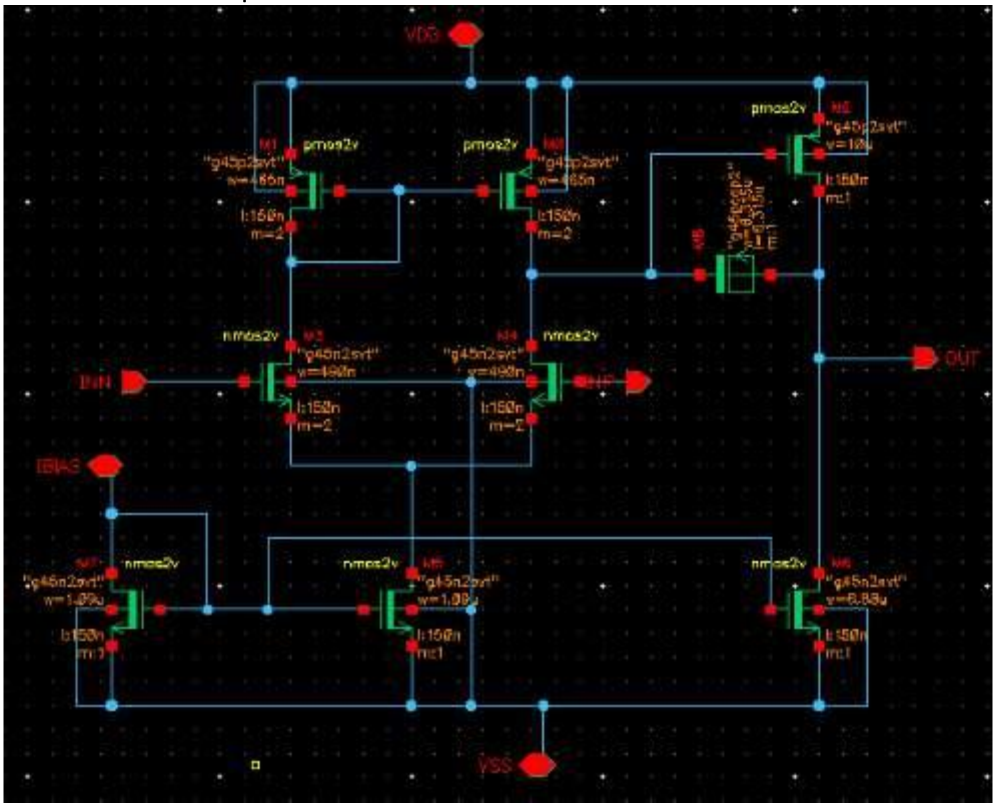


Figure – 4.2: Schematic of 2 – Stage Operational Amplifier

The Symbol created according to the Techniques demonstrated in Lab – 01 is shown in Figure – 4.3.

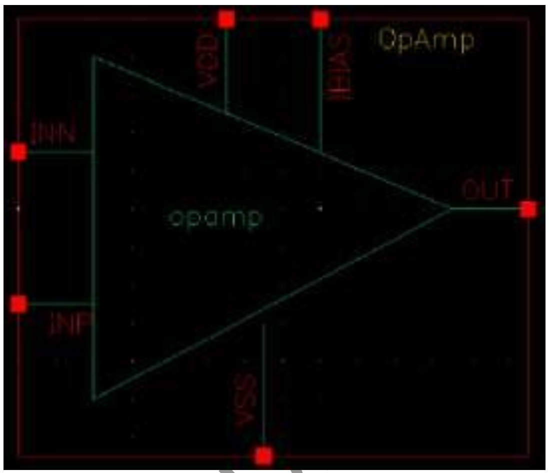


Figure – 4.3: Symbol for 2 – Stage Operational Amplifier

FUNCTIONAL SIMULATION USING ADE EXPLORER AND ASSEMBLER:

Table – 19: Device Parameters for 2 – Stage Operational Amplifier Test Schematic

Library Name	Cell Name	Comments / Properties
analogLib	vdc	DC voltage = vdd V
analogLib	vdc	DC voltag
analogLib	vpulse	Voltage 1 = vdc + 0.3 V, Voltage 2 = vdc - 0.3 V, Period = 10u s, Rise time = 10p s, Fall time = 10p s
analogLib	idc	DC current = ibias A
analogLib	cap	Capacitance = CL F
analogLib	gnd	

The Test Schematic after completion of all the interconnections can be seen as shown in Figure – 4.4.

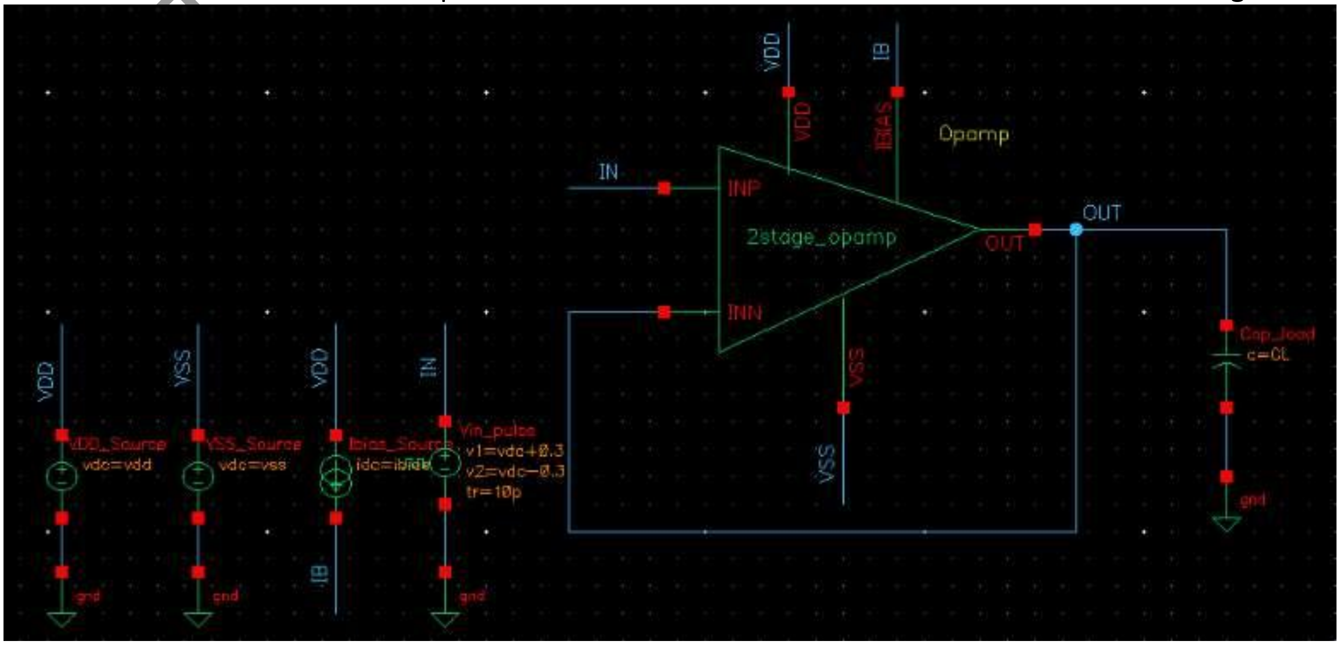


Figure – 4.4: Test Schematic for 2 – Stage Operational Amplifier

The specification that has to be achieved on simulating the design are as follows:

- Slew Rate >= 50 MV/s
- DC Open Loop Gain >= 60 dB (1000 V/V)
- Unity Gain Bandwidth >= 50 MH
- Output Offset <= ± 10 mV
- Settling Time <= 50 ns

The steps to be carried out are listed below:

Step – 1:

Select "**Launch → ADE Explorer**" as shown in Figure – 4.5.

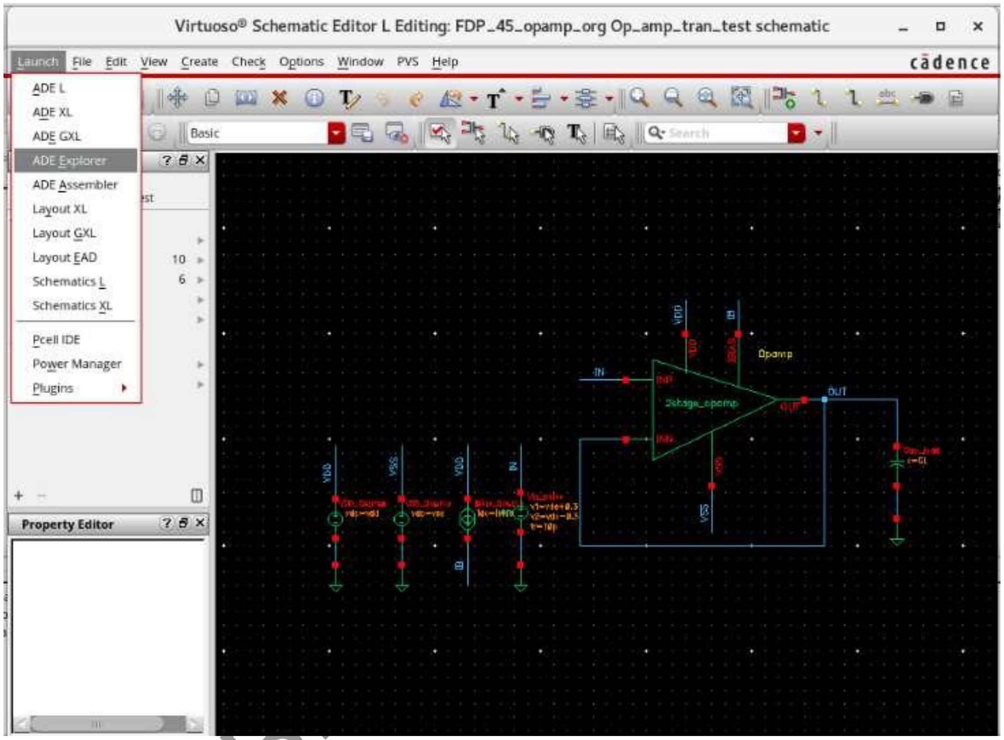


Figure – 4.5: Launch → ADE Explorer

Name of the Variable	Value
CL	1p
ibias	10u
vdc	1
vdd	2
vss	0

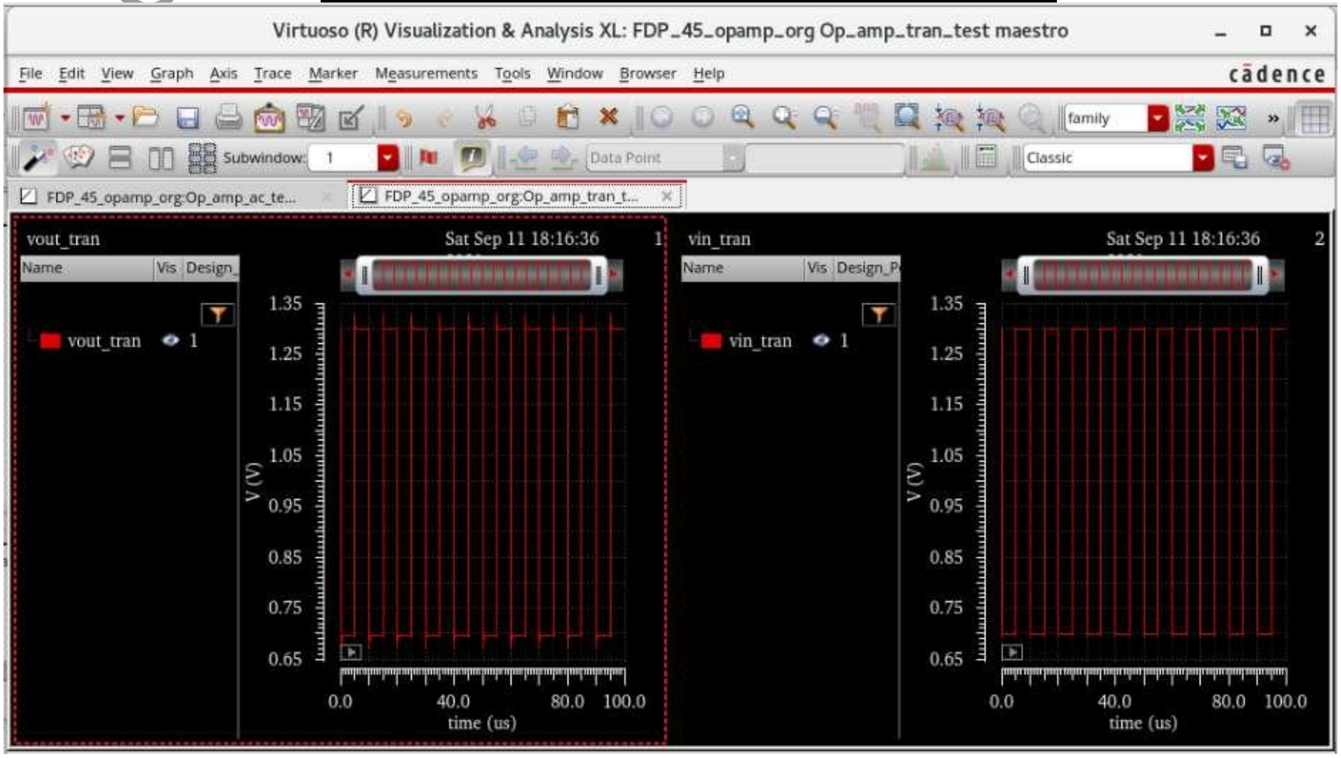

Virtuoso (R) Visualization & Analysis XL: FDP_45_opamp_org Op_amp_tran_test maestro
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
Subwindow: 1
Data Point
Classic
family
FDP_45_opamp_org:Op_amp_ac_te...
FDP_45_opamp_org:Op_amp_tran_t...
vout_tran
Sat Sep 11 18:16:36
Name Vis Design
vout_tran 1
V (V)
1.35
1.25
1.15
1.05
0.95
0.85
0.75
0.65
0.0 40.0 80.0 100.0
time (us)
vin_tran
Sat Sep 11 18:16:36
vin_tran 1
time (us)

Figure – 4.40: Plotted Waveforms

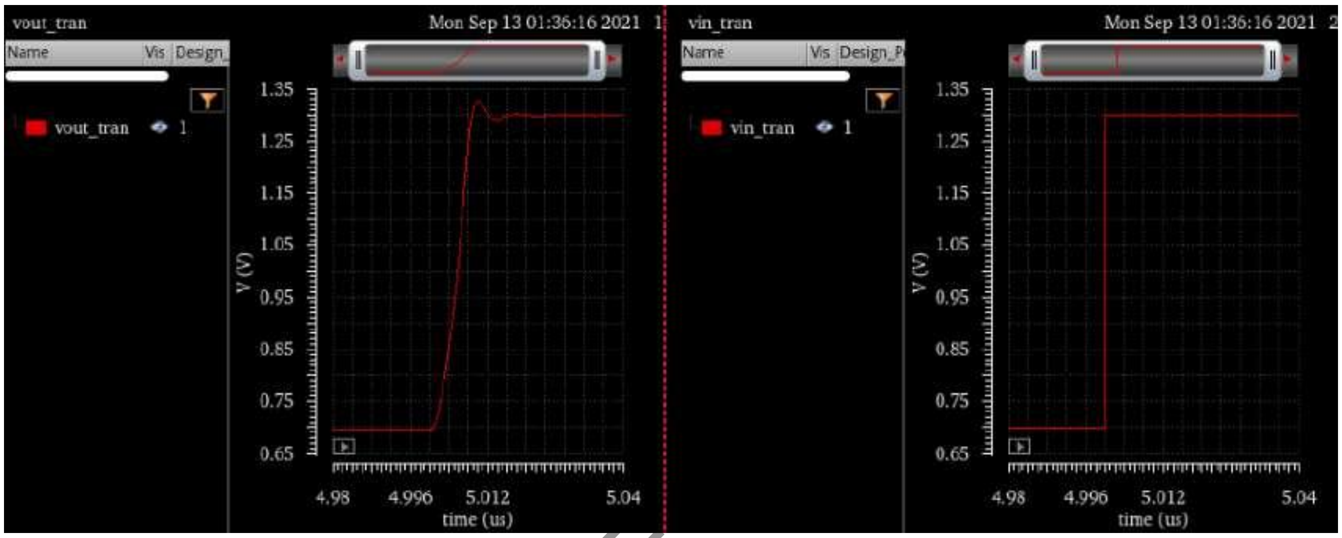


Figure – 4.44: Isolated Waveforms

Use left mouse click and drag and drop to combine the waveforms as shown in Figure – 4.45.

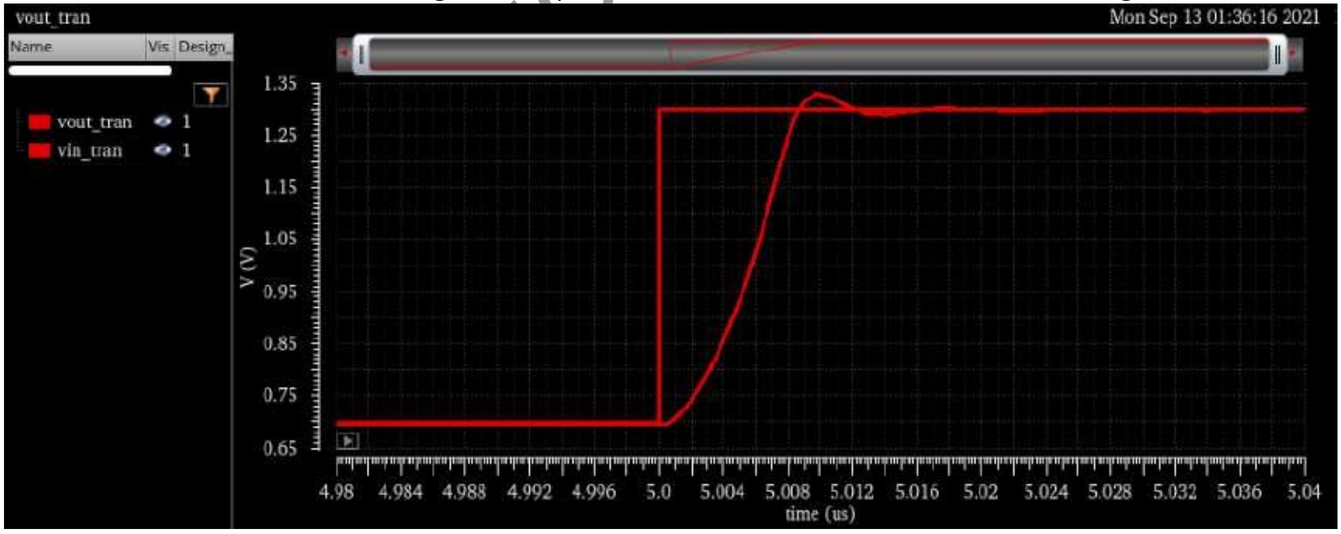


Figure – 4.45: Combined Waveforms

Use the bind key "**M**" to setup a Marker at the required time instance as shown in Figure – 4.46.

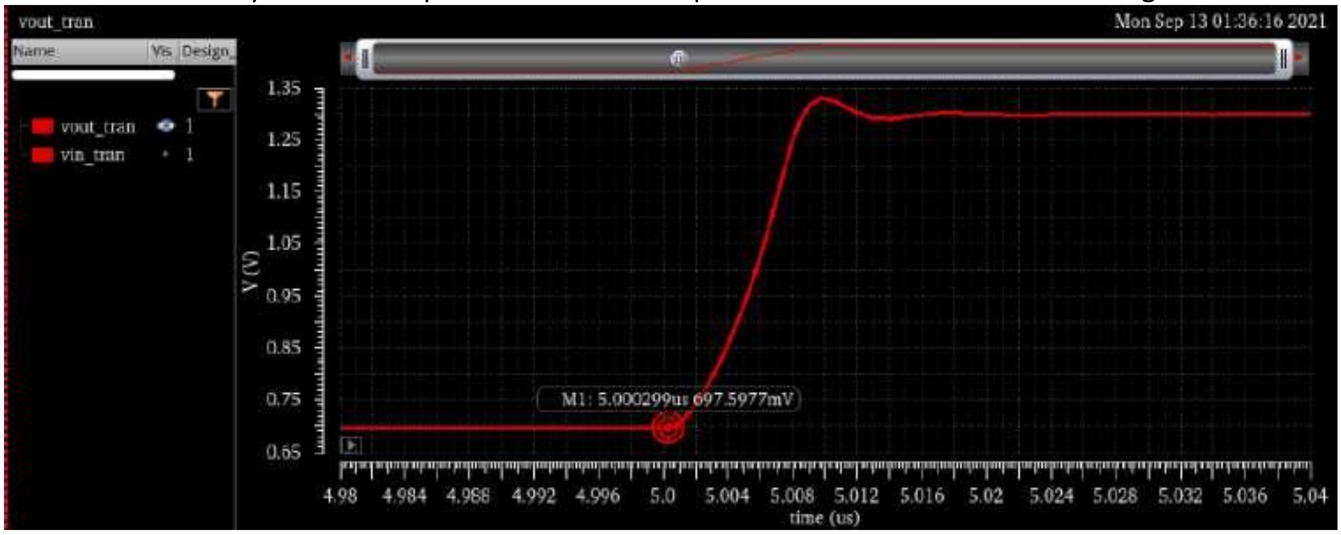


Figure – 4.46: Waveform with Marker

Use the bind key "**H**" to setup horizontal cursors at **1.294 V** and **1.306 V** as shown in Figure – 4.47.

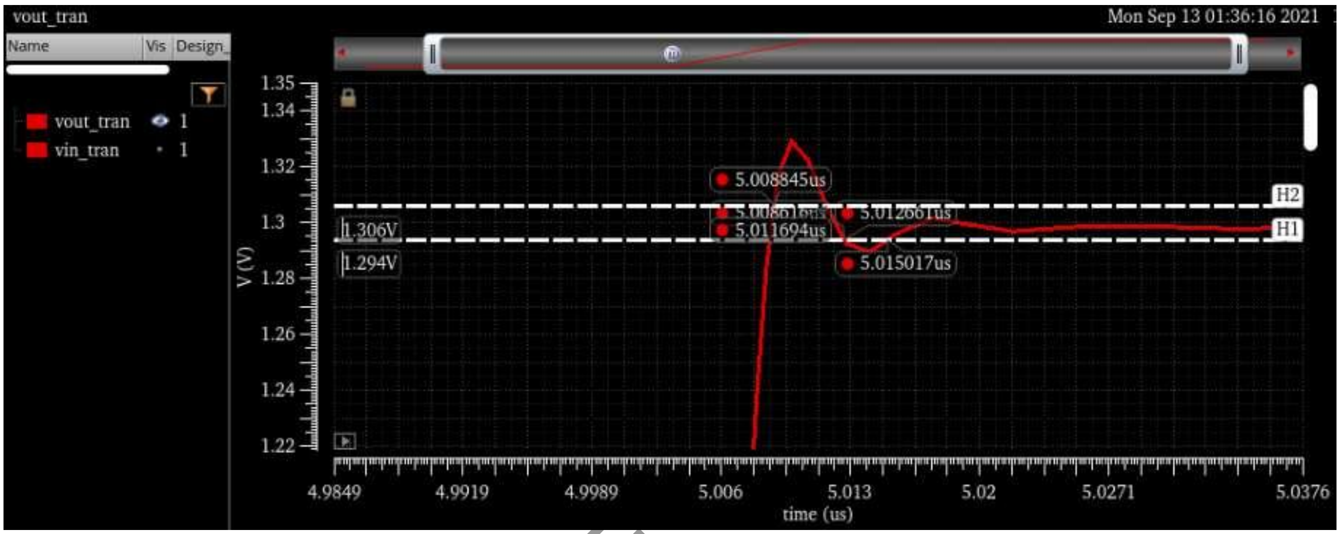


Figure – 4.47: Horizontal cursors at 1.294 V and 1.306 V

Use the zooming options to zoom-in and zoom-out as and when required. Setup a marker on the lower horizontal cursor as shown in Figure – 4.48.

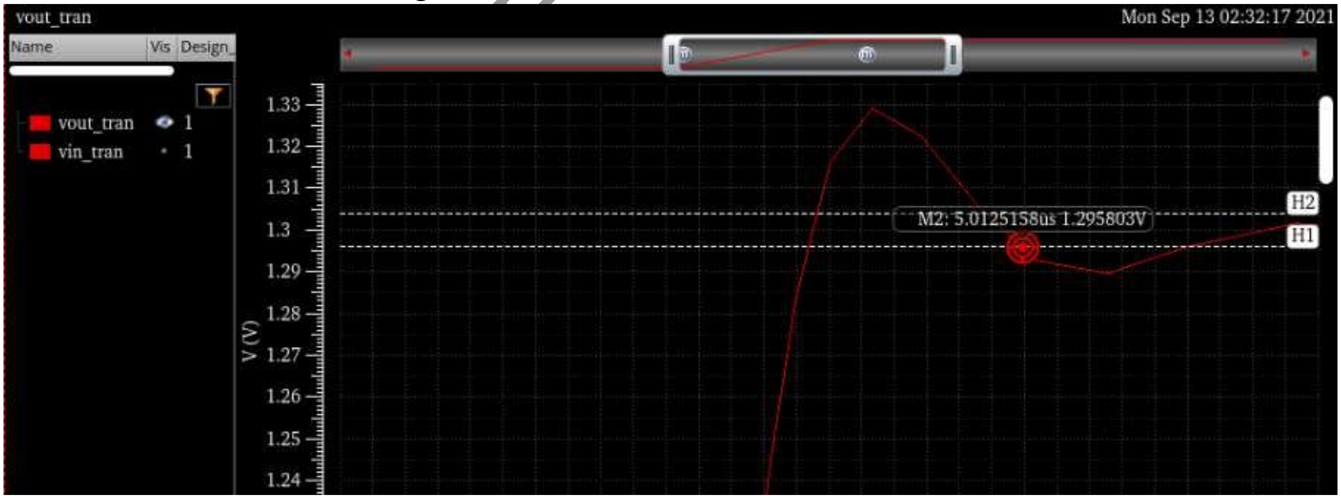


Figure – 4.48: Marker on the 2nd horizontal cursor

The difference between the timing instances gives the **Settling Time** as **12.5n s**. Without closing the waveform window, open the "**maestro**" in the ADE Assembler. For this simulation, the output dc value is 1.298 V and the input dc value is 1.3 V. The difference gives the **DC Offset (1.298 V – 1.3 V = 2m V)**.

From the AC Analysis curve, set the marker on the low frequency portion of the signal as shown in Figure – 4.49.

The marker reading gives the **DC Open Loop Gain** which is **50.98 dB**.

Setup a horizontal cursor at **0 dB** as shown in Figure – 4.50. The point of intersection of the cursor with the AC Analysis curve gives the Unity Gain Bandwidth.

The **Unity Gain Bandwidth** is measured as **84.51M Hz**.

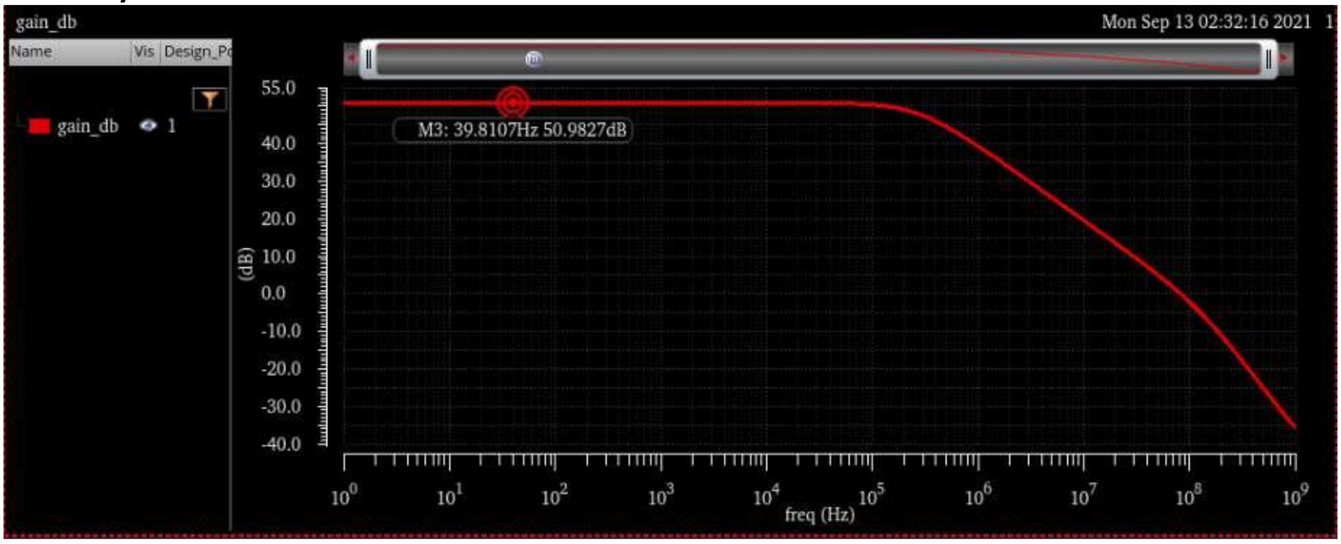


Figure – 4.49: DC Open Loop Gain – 50.98 dB

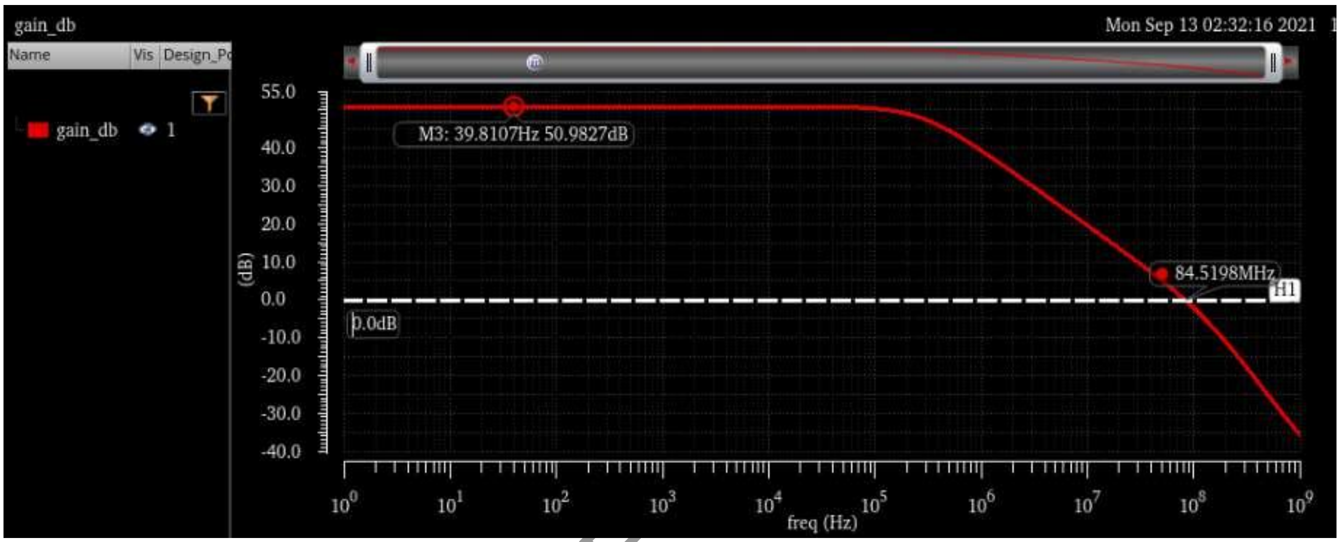


Figure – 4.50: Unity Gain Bandwidth – 84.51M Hz

GAIN MARGIN AND PHASE MARGIN:

The Schematic for measuring the Gain Margin and Phase Margin is shown in Figure – 4.68.

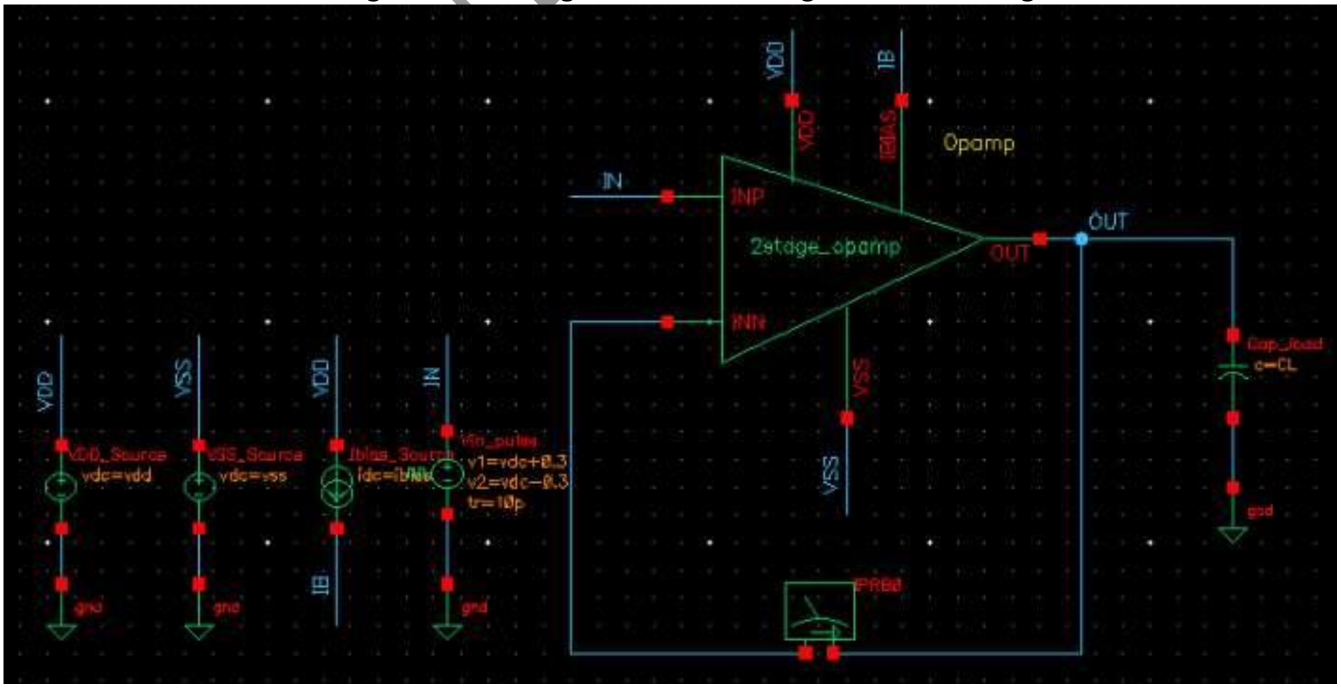


Figure – 4.68: Schematic for Gain Margin and Phase Margin measurement

The "**iprobe**" (available in "**analogLib**") acts as a signal source for the stability analysis. Create a Test Copy for the Stability Analysis as shown in Figure – 4.69.

LAYOUT:

Follow the techniques demonstrated in Lab – 01 to open the Layout Editor, import the devices from the Schematic, place the devices as per the requirement and complete the routing. The completed layout can be seen as shown in Figure – 4.76.

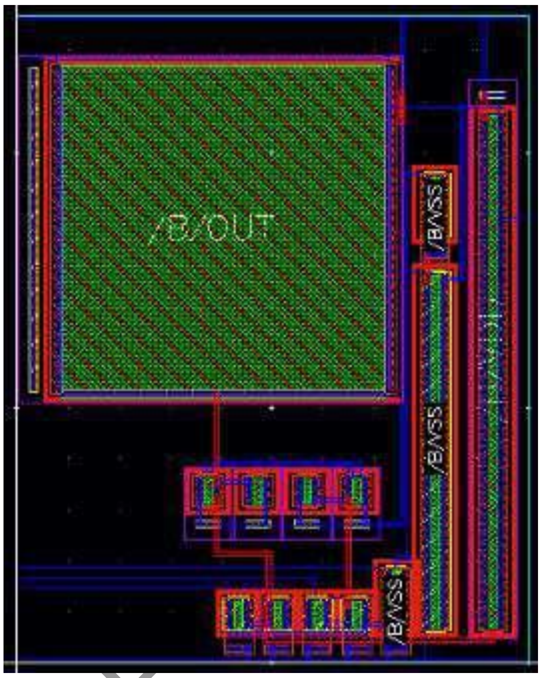


Figure – 4.76: Layout for 2 – Stage Operational Amplifier

DRC:

To check for the DRC violations, browse the "**assura_tech.lib**" file, select "**Assura** → **Run DRC**", verify the Layout Design Source, mention a "**Run Name**", select "**Technology** → **gpdk045**" and click on "**OK**" as demonstrated in Lab – 01.

LVS:

To check for the LVS violations, select "**Assura** → **Run LVS**", verify the Schematic Design Source and the Layout Design Source, mention a "**Run Name**", select "**Technology** → **gpdk045**" and click on "**OK**" as demonstrated in Lab – 01.

QRC:

To extract the Parasitics, select "**Assura** → **Quantus**", select "**Technology** → **gpdk180**", "**Output** → **Extracted View**" from the "**Setup**" option, select "**Extraction Type** → **RC**" and "**Ref Node** → **VSS**" from the "**Extraction**" and click on "**OK**" as demonstrated in Lab – 01. The result can be checked from the Library Manager.

BACKANNOTATION:

Import the parasitics into the Test Schematic and re-run the simulation to check their impact by calculating the delay elements as demonstrated in Lab – 01.

Demonstration Experiments

Lab 9: UART

Aim: Write a verilog code for UART and carry out the following:

- To Verify the Functionality using test Bench
- Synthesize Design using constraints
- Tabulate Reports using various Constraints
- Identify Critical Path and calculate Max Operating Frequency

Tool Required:

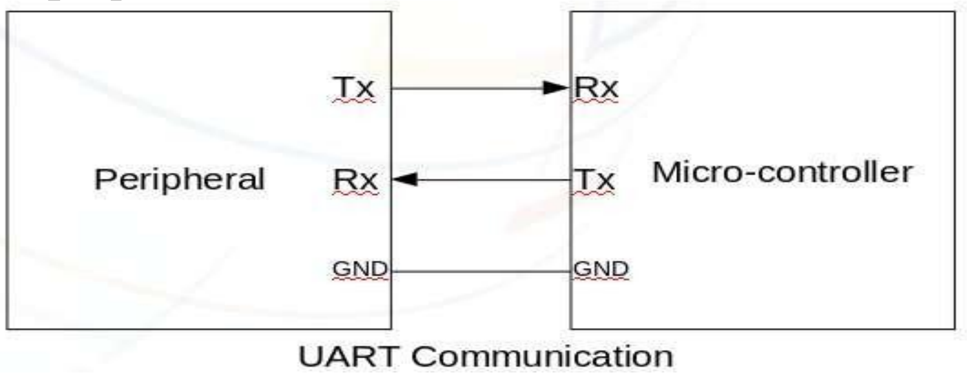
- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus

Design Information and Bock Diagram:

The **UART** is "Universal Asynchronous Receiver/Transmitter", and it is an inbuilt IC within a micro-controller but not like a communication protocol (I2C & SPI). The main function of UART is to serial data communication. In UART, the communication between two devices can be done in two ways namely serial data communication and parallel data communication.

The transmitter section includes three blocks namely transmit hold register, shift register and also control logic. Likewise, the receiver section includes a receive hold register, shift register, and control logic. These two sections are commonly provided by a baud-rate-generator. This generator is used for generating the speed when the transmitter section & receiver section has to transmit or receive the data.

Fig: UART



Creating a Workspace :

- Create a new sub-Directory for the Design and open a terminal from the Sub-Directory.

a) Functional Verification using Test Bench

Source Code – Transmitter :

```
// This code contains the UART Transmitter.  This transmitter is able
// to transmit 8 bits of serial data, one start bit, one stop bit,
// and no parity bit.  When transmit is complete o_Tx_done will be
// driven high for one clock cycle.
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217

module UART_TX
  #(parameter CLKS_PER_BIT = 217)
  (
   input       i_Clock,
   input       i_TX_DV,
   input  [7:0] i_TX_Byte,
   output      o_TX_Active,
   output reg o_TX_Serial,
   output      o_TX_Done
   );

  parameter IDLE  = 3'b000;
  parameter TX_START_BIT = 3'b001;
  parameter TX_DATA_BITS = 3'b010;
  parameter TX_STOP_BIT = 3'b011;
  parameter CLEANUP      = 3'b100;
  reg [2:0] r_SM_Main      = 0;
  reg [7:0] r_Clock_Count = 0;
  reg [2:0] r_Bit_Index  = 0;
  reg [7:0] r_TX_Data     = 0;
  reg      r_TX_Done       = 0;
  reg      r_TX_Active  = 0;

always @(posedge i_Clock)
begin
case (r_SM_Main)
IDLE :
       begin
         o_TX_Serial   <= 1'b1;          // Drive Line High for Idle
         r_TX_Done      <= 1'b0;
        r_Clock_Count <= 0;
         r_Bit_Index  <= 0;
         if (i_TX_DV == 1'b1)
         begin
               r_TX_Active <= 1'b1;
               r_TX_Data  <= i_TX_Byte;
              r_SM_Main  <= TX_START_BIT;
           end
         else
        r_SM_Main <= IDLE;
       end //
      case: IDLE
// Send out Start Bit. Start bit = 0
      TX_START_BIT :
       begin
         o_TX_Serial <= 1'b0;

         // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
         if (r_Clock_Count < CLKS_PER_BIT-1)
         begin
           r_Clock_Count <= r_Clock_Count + 1;
           r_SM_Main    <= TX_START_BIT;
           end
           else
           begin
           r_Clock_Count <= 0;
           r_SM_Main      <= TX_DATA_BITS;
           end
       end                // case: TX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
      TX_DATA_BITS :
       begin
```

```
          o_TX_Serial <= r_TX_Data[r_Bit_Index];

    if (r_Clock_Count < CLKS_PER_BIT-1)
    begin
            r_Clock_Count <= r_Clock_Count + 1;
            r_SM_Main    <= TX_DATA_BITS;
            end
            else
            begin

           r_Clock_Count <= 0;            // Check if we have sent out all bits
       if (r_Bit_Index < 7)

       begin
              r_Bit_Index <= r_Bit_Index + 1;
               r_SM_Main <= TX_DATA_BITS;
              end
            else
            begin
              r_Bit_Index <= 0;
              r_SM_Main  <=
            TX_STOP_BIT;
              end
          end
        end                          // case: TX_DATA_BITS

       // Send out Stop bit. Stop bit = 1
       TX_STOP_BIT :
         begin
           o_TX_Serial <= 1'b1;

           // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
           if (r_Clock_Count < CLKS_PER_BIT-1)
           begin
             r_Clock_Count <= r_Clock_Count + 1;
             r_SM_Main     <= TX_STOP_BIT;
             end
             else
             begin
             r_TX_Done     <= 1'b1; r_Clock_Count <= 0;
             r_SM_Main     <= CLEANUP;
             r_TX_Active  <= 1'b0;
           end
         end // case: TX_STOP_BIT

       // Stay here 1 clock
       CLEANUP :
         begin
           r_TX_Done <= 1'b1;
           r_SM_Main <= IDLE;
            end
       default :
         r_SM_Main <= IDLE;
         endcase
     end
```

```
  assign o_TX_Active = r_TX_Active;

  assign o_TX_Done  = r_TX_Done;
  endmodule
```

Source Code – Receiver :

```
// This file contains the UART Receiver.  This receiver is able to
// receive 8 bits of serial data, one start bit, one stop bit,
// and no parity bit.  When receive is complete o_rx_dv will be
// driven high for one clock cycle.
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217

module UART_RX
  #(parameter CLKS_PER_BIT =
  217)(
   input        i_Clock,
   input        i_RX_Serial,
   output       o_RX_DV,
   output [7:0] o_RX_Byte
   );

  parameter IDLE         = 3'b000;
  parameter RX_START_BIT = 3'b001;
  parameter RX_DATA_BITS = 3'b010;
  parameter RX_STOP_BIT = 3'b011;
  parameter CLEANUP  = 3'b100;

  reg [7:0] r_Clock_Count = 0;
  reg [2:0] r_Bit_Index = 0; //8 bits total
  reg [7:0] r_RX_Byte      = 0;
  reg    r_RX_DV      = 0;
  reg [2:0] r_SM_Main    = 0;

  // Purpose: Control RX state machine
  always @(posedge i_Clock)
  begin
    case (r_SM_Main)
    IDLE :
   Begin
   r-RX_DV  <=1'b0;
   r_Clock_Count <= 0;
   r_Bit_Index  <= 0;

    if (i_RX_Serial == 1'b0)           // Start bit detected

          r_SM_Main <= RX_START_BIT;
        else
          r_SM_Main <= IDLE;
          end
      // Check middle of start bit to make sure it's still low
      RX_START_BIT :
        begin
```

```
          if (r_Clock_Count == (CLKS_PER_BIT-1)/2)
          begin
            if (i_RX_Serial == 1'b0)
            begin
              r_Clock_Count <= 0;  // reset counter, found the middle
              r_SM_Main     <= RX_DATA_BITS;
              end
              else
              r_SM_Main <= IDLE;
              end
              else
              begin
          r_Clock_Count <= r_Clock_Count + 1;
          r_SM_Main     <= RX_START_BIT;
          end
        end              // case: RX_START_BIT

      // Wait CLKS_PER_BIT-1 clock cycles to sample serial data
      RX_DATA_BITS :
        begin
          if (r_Clock_Count < CLKS_PER_BIT-1)
          begin
            r_Clock_Count <= r_Clock_Count + 1;
            r_SM_Main     <= RX_DATA_BITS;
            end
           else
            begin
            r_Clock_Count    <= 0;
            r_RX_Byte[r_Bit_Index] <= i_RX_Serial;
            // Check if we have received all bits
            if (r_Bit_Index < 7)
              begin
              r_Bit_Index <= r_Bit_Index + 1;
              r_SM_Main  <= RX_DATA_BITS;
              end
              else
                  begin
                            r_Bit_Index <= 0;
                            r_SM_Main  <= RX_STOP_BIT;
                            end
       end
    end                    // case: RX_DATA_BITS
      // Receive Stop bit. Stop bit = 1
      RX_STOP_BIT :
        begin
          // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
          if (r_Clock_Count < CLKS_PER_BIT-1)
          begin
            r_Clock_Count  <= r_Clock_Count + 1;
            r_SM_Main   <= RX_STOP_BIT;
              end
              else
              begin
                r_RX_DV        <= 1'b1;
            r_Clock_Count <= 0;
            r_SM_Main    <= CLEANUP;
            end
        end                 // case: RX_STOP_BIT
      // Stay here 1 clock
      CLEANUP :
        begin
          r_SM_Main <= IDLE;
          r_RX_DV  <= 1'b0;
        end
      default :
        r_SM_Main <= IDLE;
              endcase
              end
    assign o_RX_DV = r_RX_DV;
    assign o_RX_Byte = r_RX_Byte;
    endmodule // UART_RX
```

Test bench :

```
// This testbench will exercise the UART RX.
// It sends out byte 0x37, and ensures the RX receives it correctly.
`timescale 1ns/10ps
`include "uart_tx.v"
`include "uart_rx.v"
module UART_TB ();
  // Testbench uses a 25 MHz clock
  // Want to interface to 115200 baud UART
  // 25000000 / 115200 = 217 Clocks Per Bit.
  parameter c_CLOCK_PERIOD_NS = 40;
  parameter c_CLKS_PER_BIT        = 217;
  parameter c_BIT_PERIOD         = 8600;
  reg r_Clock = 0;
  reg r_TX_DV = 0;
  wire w_TX_Active, w_UART_Line;
  wire w_TX_Serial;
  reg [7:0] r_TX_Byte = 0;
  wire [7:0] w_RX_Byte;

  UART_RX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_RX_Inst
    (.i_Clock(r_Clock),
     .i_RX_Serial(w_UART_Line),
     .o_RX_DV(w_RX_DV),
     .o_RX_Byte(w_RX_Byte)
     );

  UART_TX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_TX_Inst
    (.i_Clock(r_Clock),
     .i_TX_DV(r_TX_DV),
     .i_TX_Byte(r_TX_Byte),
     .o_TX_Active(w_TX_Active),
     .o_TX_Serial(w_TX_Serial),
     .o_TX_Done()
     );
  // Keeps the UART Receive input high (default) when
  // UART transmitter is not active
  assign w_UART_Line = w_TX_Active ? w_TX_Serial : 1'b1;
  always
    #(c_CLOCK_PERIOD_NS/2) r_Clock <= !r_Clock;
 // Main Testing:
initial
    begin
      // Tell UART to send a command (exercise TX)
      @(posedge r_Clock);
      @(posedge r_Clock);
      r_TX_DV  <= 1'b1;
      r_TX_Byte <= 8'h3F;
      @(posedge r_Clock);
      r_TX_DV <= 1'b0;
  end
  endmodule
```

Waveform :

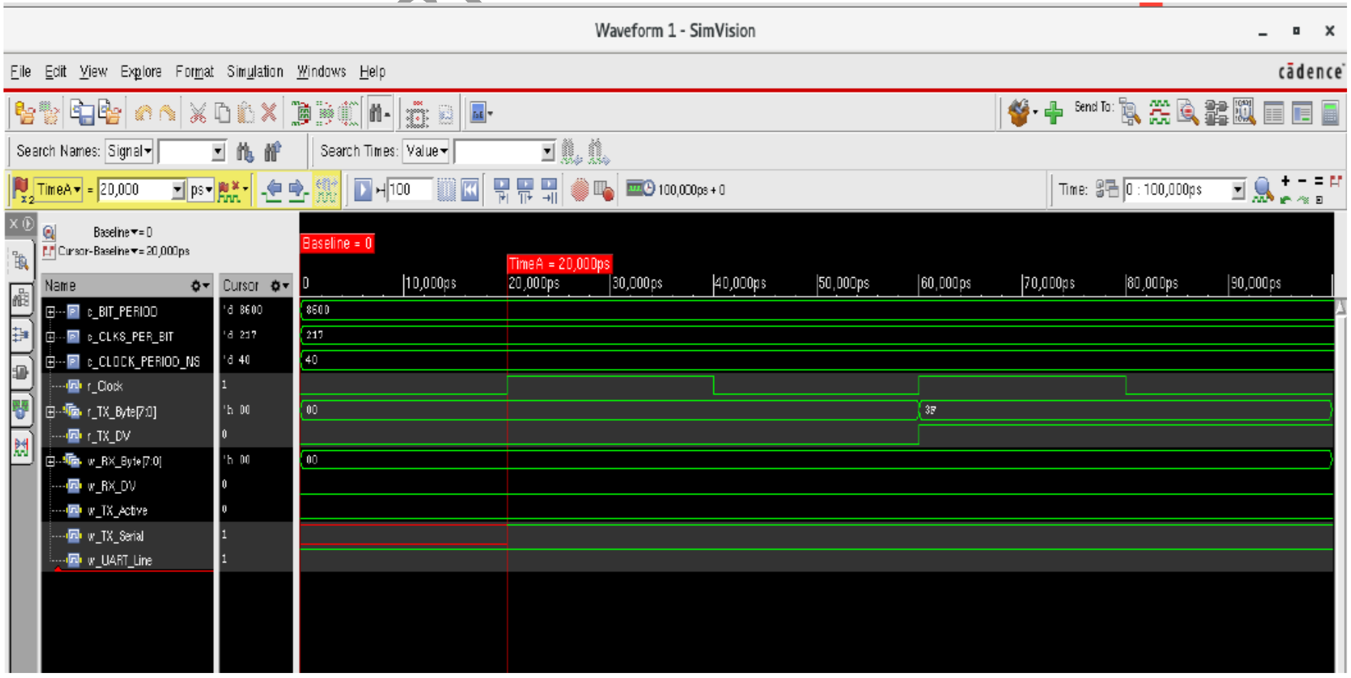
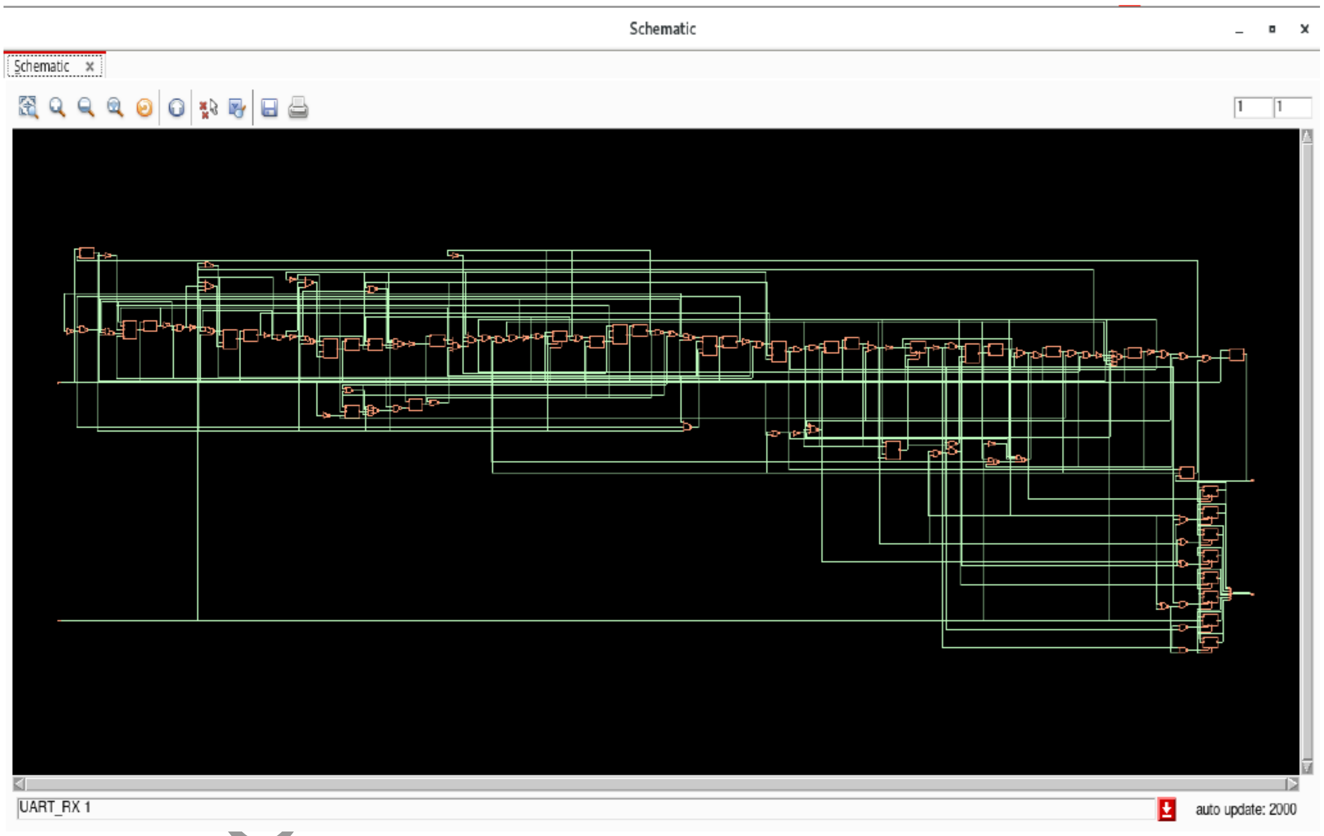


Fig: Simulation Waveform for UART

b) Synthesize the Design

```
1.  read_libs /home/install/FOUNDRY/digital/90nm/dig/lib/slow.lib
2.  read_hdl {uart_tx.v / uart_rx.v}    //Choose any one
3.  elaborate
4.  read_sdc constraints_top.sdc        //Reading Top Level SDC
5.  set_db syn_generic_effort medium //Setting effort medium
6.  set_db syn_map_effort medium
7.  set_db syn_opt_effort medium
8.  syn_generic
9.  syn_map
10. syn_opt     //Performing Synthesis Mapping and Optimisation
11. report_timing > uart_timing.rep  //Generates Timing report for worst datapath and dumps into file
12. report_area > uart_area.rep       //Generates Synthesis Area report and dumps into a file
13. report_power > uart_power.rep  //Generates Power Report [Pre-Layout]
14. report_qor > uart_qor.rep
15. write_hdl > uart_netlist.v           //Creates readable Netlist File
16. write_sdc > uart_sdc.sdc             //Creates Block Level SDC
```

Synthesis RTL Schematic :



Note :-

1. You can tabulate Area, Power and Timing Constraints using any of theSDC Constraints as instructed.
2. Make sure, during synthesis the Report File Names are changed so that the latest reports donot overwrite the earlier ones.

Lab 6: Physical Design

Aim: For the synthesized netlist carry out the following any two above experiments:

- Floor planning, identify the placement of pads, placement and Routing

Tool Required:

- Functional Simulation: Incisive Simulator (ncvlog, ncelab, ncsim)
- Synthesis: Genus
- Physical Design: Innovus

Mandatory Inputs for PD:

1. Gate Level Netlist [Output of Synthesis]
2. Block Level SDC [Output of Synthesis]
3. Liberty Files (.lib)
4. LEF Files (Layer Exchange Format)

Expected Outputs from PD:

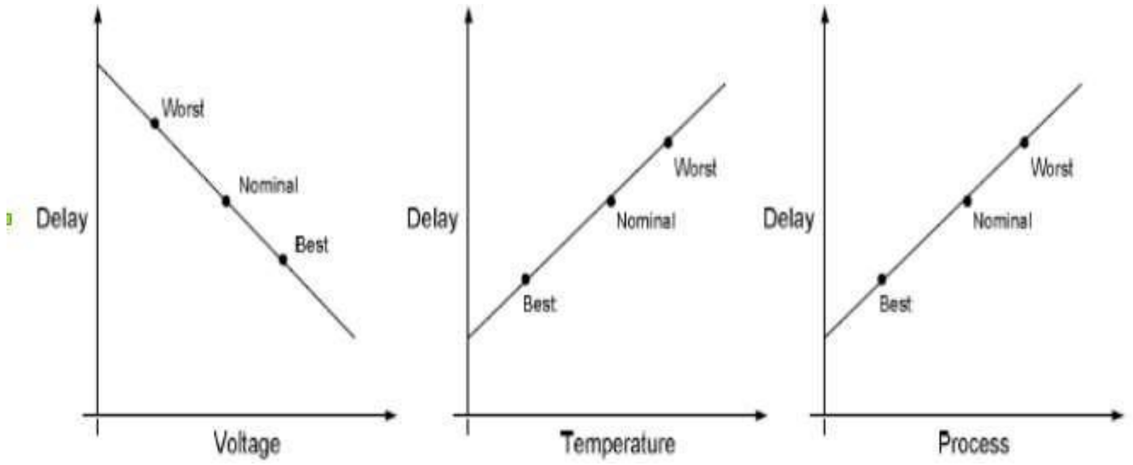
1. GDS II File (Graphical Data Stream for Information Interchange – Feed In for FabricationUnit).
2. SPEF, SDF
   - Make sure the Synthesis for the target design is done and open a terminal from the corresponding workspace.
   - Initiate the Cadence tools and **cmd :**innovus (Press Enter)
   - For Innovus tool, a GUI opens and also the terminal enters into innovus command prompt where in the tool commands can be entered.

Physical Design involves 5 stages as following :
After Importing Design,
→ Floor Planning
→ Power Planning
→ Placement
→ CTS (Clock Tree Synthesis)
→ RoutingModule

Importing Design
To Import Design, all the Mandatory Inputs are to be loaded and this can be done either using script files named with .globals and .view/.tcl or through GUI as shown below.
The target design considered here is counter design
The procedure shall remain the same for any other design from the above discussed experiments.

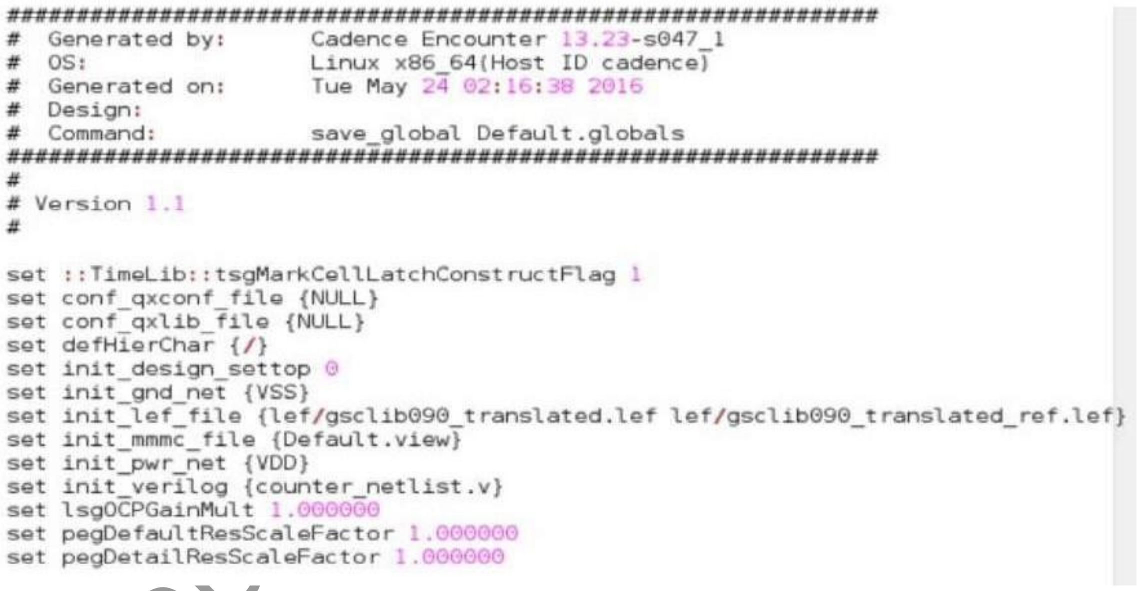
Note :

1. For Synthesis, slow.lib was read as input. Each liberty file contains a pre-defined Process,Voltage and Temperature (PVT) values which impact the ease of charge movement.
2. Process, Voltage and Temperature individually affect the ease of currents as depicted below.



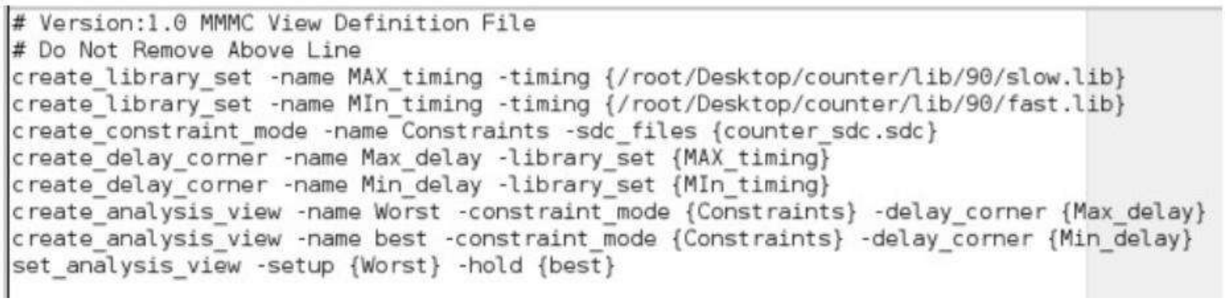
3. Hence, slow.lib contains PVT combination (corner) with **slow** charge movement =>

Maximum Delay => **Worst** Performance
4. Similarly, fast.lib contains PVT Combination applicable across its designs to give **Fast** charge movement => **Minimum** Delay => **Best** Performance.
5. When these corners are collaborated with the sdc, they can be used to analyse timing forsetup in the worst case and hold in the best case.
6. All these analysis views are to be manually created either in the form of script or using theGUI.



```
#####################################################################
#   Generated by:       Cadence Encounter 13.23-s047_1
#   OS:                 Linux x86_64(Host ID cadence)
#   Generated on:       Tue May 24 02:16:38 2016
#   Design:
#   Command:            save_global Default.globals
#####################################################################
#
# Version 1.1
#

set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set conf_qxconf_file {NULL}
set conf_qxlib_file {NULL}
set defHierChar {/}
set init_design_settop 0
set init_gnd_net {VSS}
set init_lef_file {lef/gsclib090_translated.lef lef/gsclib090_translated_ref.lef}
set init_mmmc_file {Default.view}
set init_pwr_net {VDD}
set init_verilog {counter_netlist.v}
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
```

Script under Default.globals file

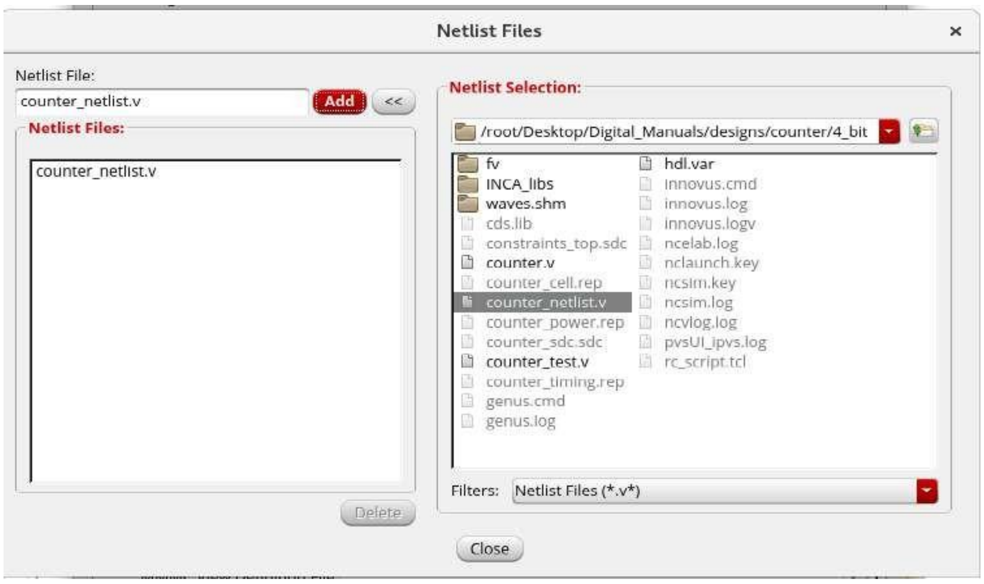


```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_library_set -name MAX_timing -timing {/root/Desktop/counter/lib/90/slow.lib}
create_library_set -name MIn_timing -timing {/root/Desktop/counter/lib/90/fast.lib}
create_constraint_mode -name Constraints -sdc_files {counter_sdc.sdc}
create_delay_corner -name Max_delay -library_set {MAX_timing}
create_delay_corner -name Min_delay -library_set {MIn_timing}
create_analysis_view -name Worst -constraint_mode {Constraints} -delay_corner {Max_delay}
create_analysis_view -name best -constraint_mode {Constraints} -delay_corner {Min_delay}
set_analysis_view -setup {Worst} -hold {best}
```

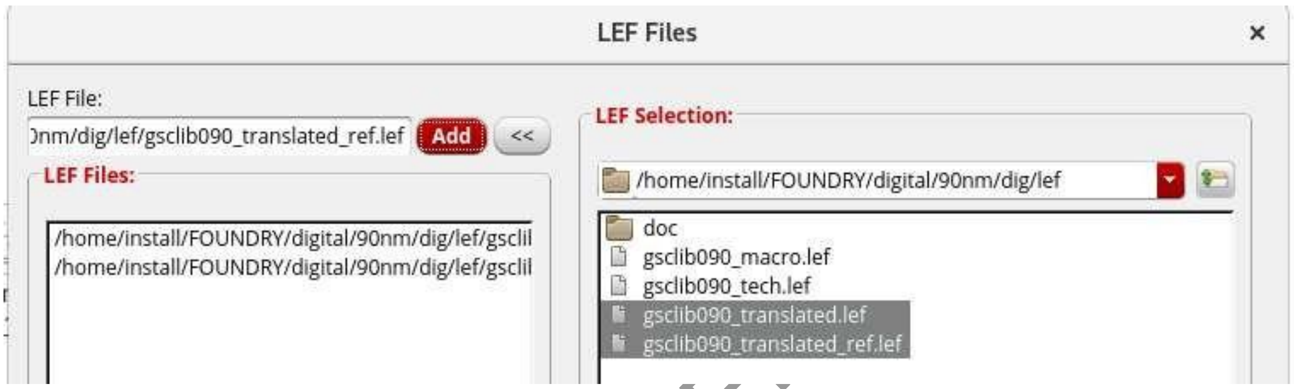
Script under Default.view (or) Default.tcl file

Note : Check the paths to properly read in the input files.

- Else, if you would like to import your design using GUI, open the Innovus tool and from the
    - GUI, go to File → Import Design.
- A new pop-up window appears.
- First load the netlist. You can browse for the file and select "Top cell : Auto Assign".



Similarly select your lef files from /home/install/FOUNDRY/digital/90nm/dig/lef/ as shown below.

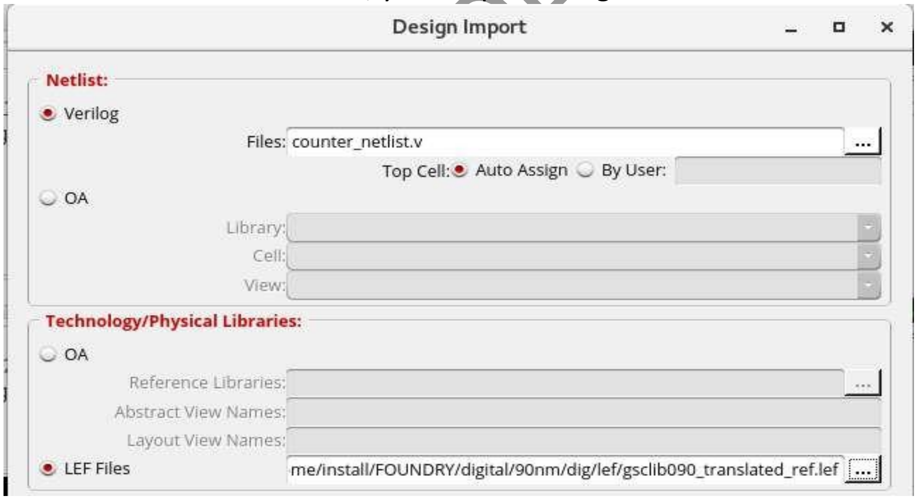


LEF File:)nm/dig/lef/gsclib090_translated_ref.lef

LEF Files:
/home/install/FOUNDRY/digital/90nm/dig/lef/gsclil
/home/install/FOUNDRY/digital/90nm/dig/lef/gsclil

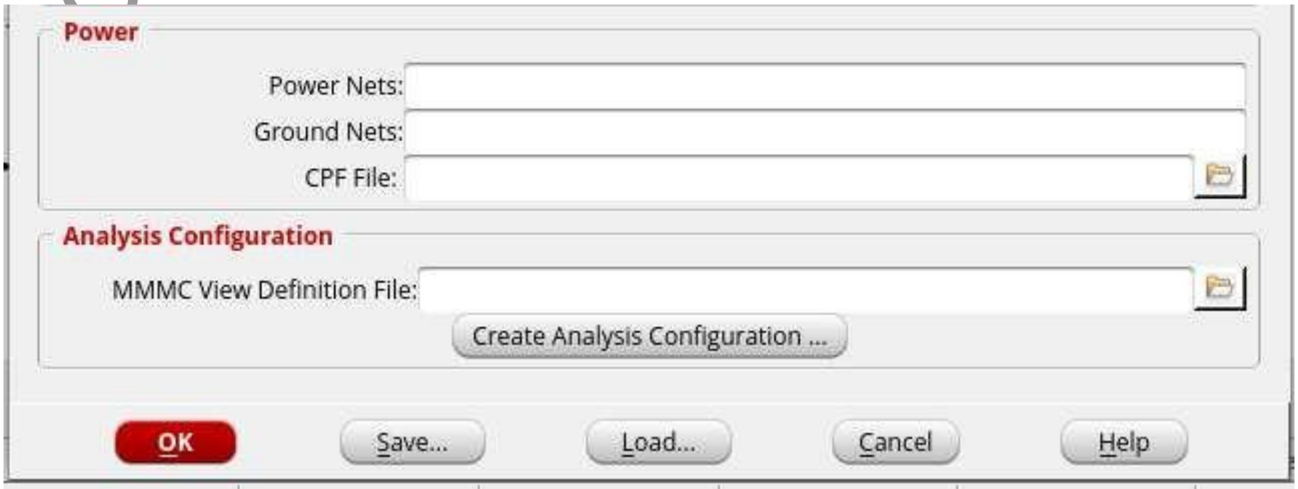
LEF Selection: /home/install/FOUNDRY/digital/90nm/dig/lef
gsclib090_macro.lef
gsclib090_tech.lef
gsclib090_translated.lef
gsclib090_translated_ref.lef

Once both the Netlist and LEF Files are loaded, your import design window is as follows.

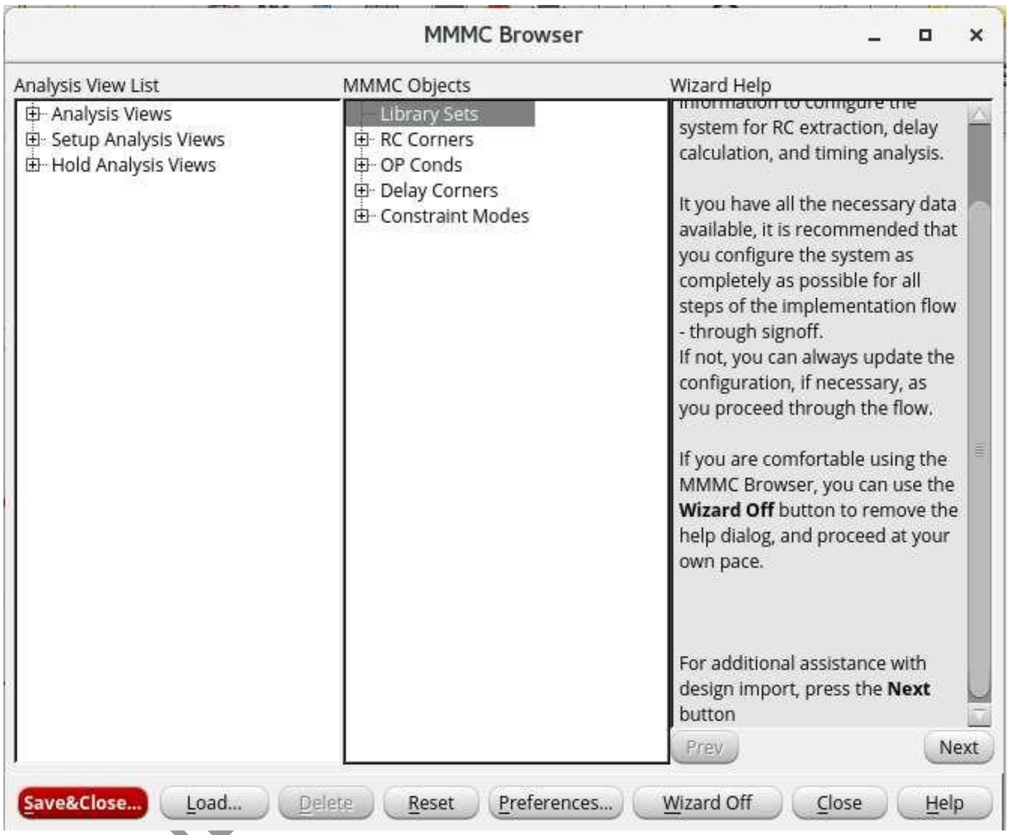


[✓ Verilog]
Files: counter_netlist.v
Top Cell: [✓ Auto Assign] By User:
OA
Library:
Cell:
View:
OA
Reference Libraries:
Abstract View Names:
Layout View Names:
[✓ LEF Files] me/install/FOUNDRY/digital/90nm/dig/lef/gsclib090_translated_ref.lef

- In order to load the Liberty File and SDC, create delay corners and analysis view, select the “Create Analysis Configuration” option at the bottom.



- An MMMC browser Pops Up.



information to configure the system for RC extraction, delay calculation, and timing analysis.

It you have all the necessary data available, it is recommended that you configure the system as completely as possible for all steps of the implementation flow - through signoff.
If not, you can always update the configuration, if necessary, as you proceed through the flow.

If you are comfortable using the MMMC Browser, you can use the **Wizard Off** button to remove the help dialog, and proceed at your own pace.

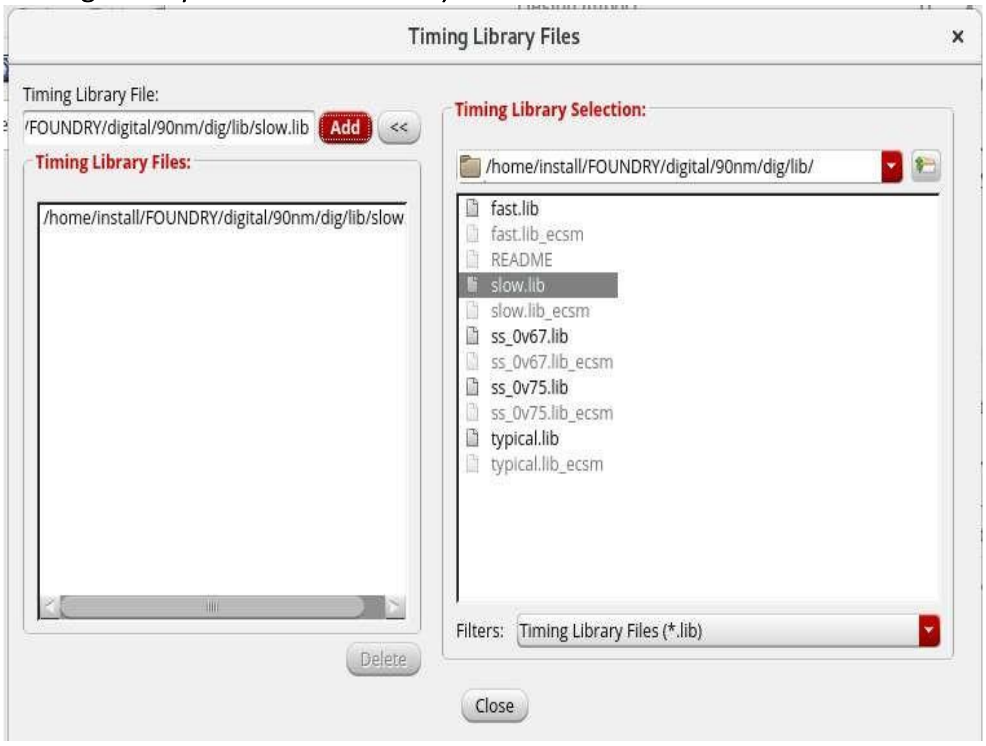
For additional assistance with design import, press the **Next** button

The order of adding the MMMC Objects is as follows.

1. Library Sets
2. RC Corners
3. Delay Corners
4. Constraints (SDC)

Once all of them are added, Analysis Views are created and assigned to Setup and Hold.
In order to add any of the objects, make a right click on the corresponding label → Select New.

- Adding Liberty Files under "Library Sets"



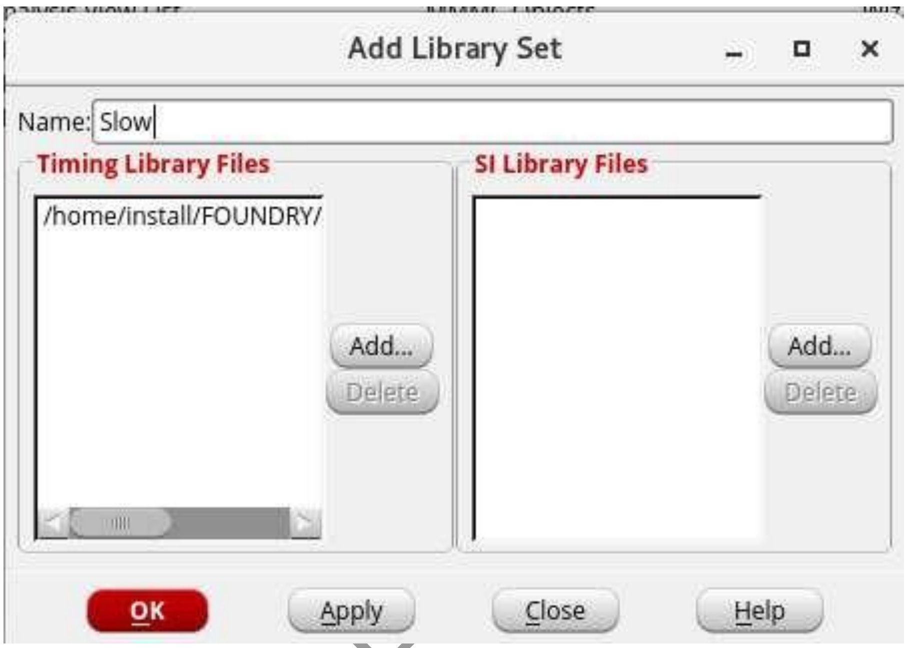
/FOUNDRY/digital/90nm/dig/lib/slow.lib

/home/install/FOUNDRY/digital/90nm/dig/lib/slow

/home/install/FOUNDRY/digital/90nm/dig/lib/

fast.lib
fast.lib_ecsm
README
slow.lib
slow.lib_ecsm
ss_0v67.lib
ss_0v67.lib_ecsm
ss_0v75.lib
ss_0v75.lib_ecsm
typical.lib
typical.lib_ecsm

Timing Library Files (*.lib)



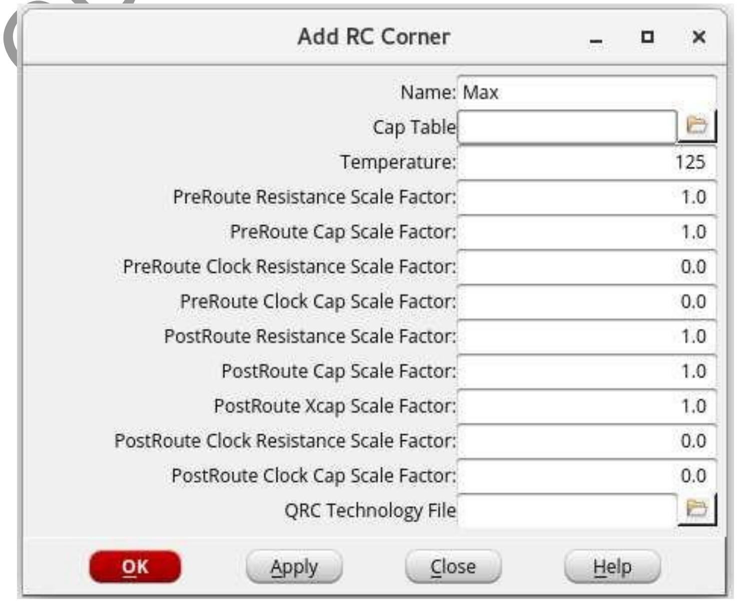
Add Library Set

Name: Slow

Timing Library Files: /home/install/FOUNDRY/

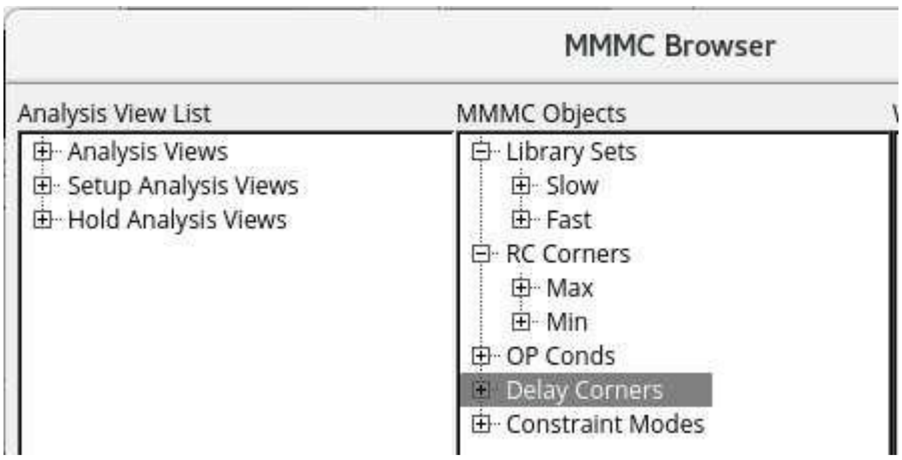
SI Library Files:

- Similarly, add fast.lib with a label Fast or any identifier of your own.
- Adding RC Corners can also be done in a similar process. The temperature value can be found under the corresponding liberty file. Also, cap table and RC Tech files can be addedfrom Foundry where available.



Add RC Corner

Name:	Max
Cap Table	
Temperature:	125
PreRoute Resistance Scale Factor:	1.0
PreRoute Cap Scale Factor:	1.0
PreRoute Clock Resistance Scale Factor:	0.0
PreRoute Clock Cap Scale Factor:	0.0
PostRoute Resistance Scale Factor:	1.0
PostRoute Cap Scale Factor:	1.0
PostRoute Xcap Scale Factor:	1.0
PostRoute Clock Resistance Scale Factor:	0.0
PostRoute Clock Cap Scale Factor:	0.0
QRC Technology File	

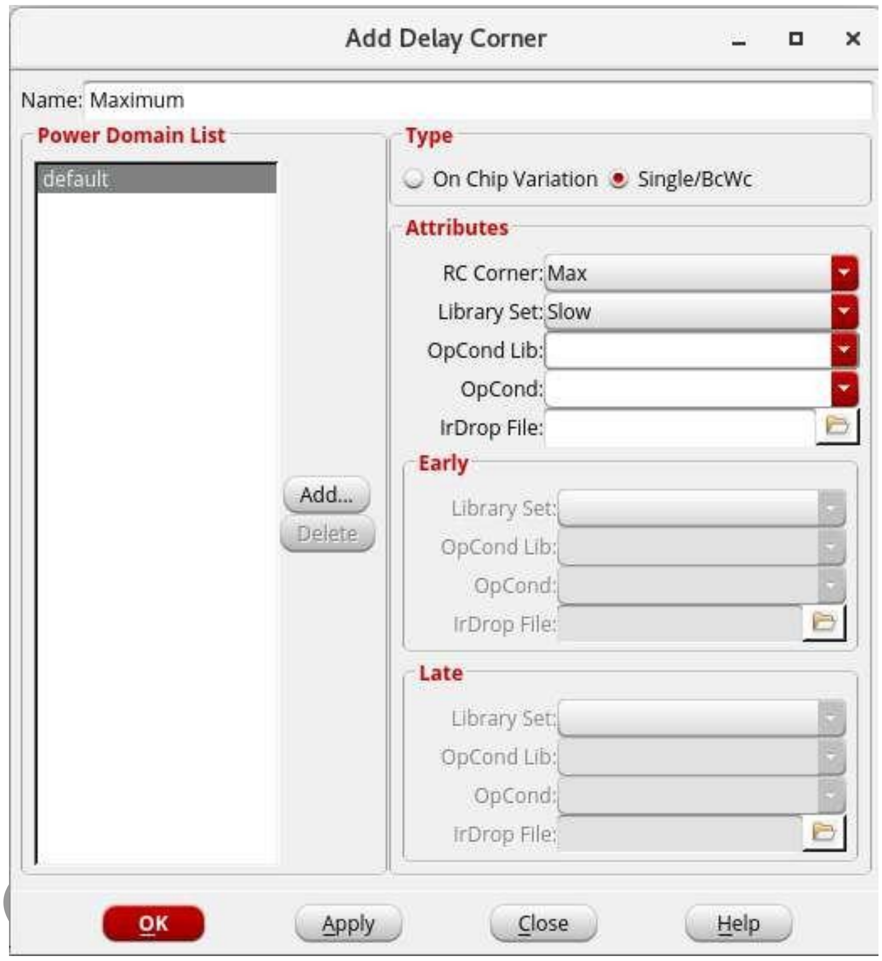


MMMC Browser

Analysis View List
- Analysis Views
- Setup Analysis Views
- Hold Analysis Views

MMMC Objects
- Library Sets
  - Slow
  - Fast
- RC Corners
  - Max
  - Min
- OP Conds
- Delay Corners
- Constraint Modes

- Delay Corners are formed by combining Library Sets with RC Corners.
- An example is shown below.



Name: Maximum

Type: On Chip Variation [✓ Single/BcWc]

Attributes

RC Corner: Max

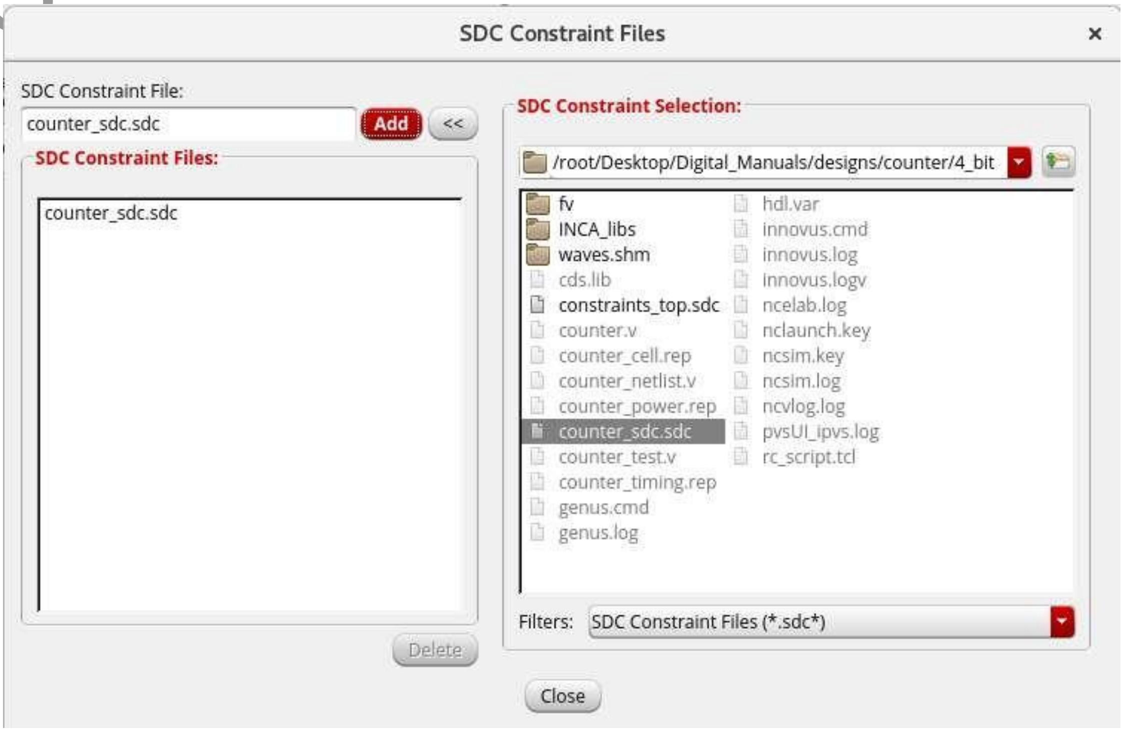
Library Set: Slow

OpCond Lib:

OpCond:

IrDrop File:

- Similarly, SDC can be read in under the MMMC Object of "Constraints".



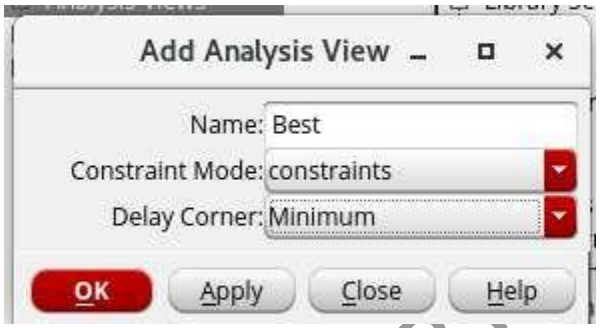
SDC Constraint File: counter_sdc.sdc

SDC Constraint Files: counter_sdc.sdc

SDC Constraint Selection: /root/Desktop/Digital_Manuals/designs/counter/4_bit

Filters: SDC Constraint Files (*.sdc*)

- Analysis Views are formed from combinations of SDC and Delay Corner.

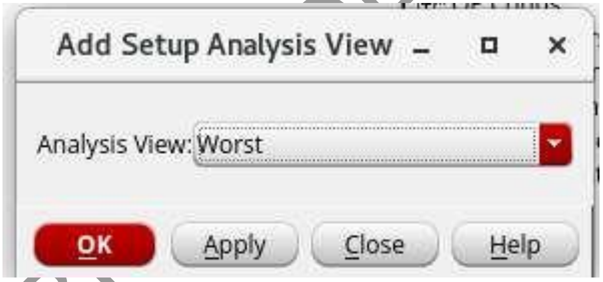

Add Analysis View

Name: Best

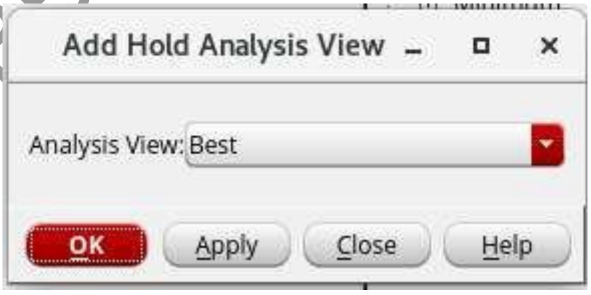
Constraint Mode: constraints

Delay Corner: Minimum

- Once “Best” and “Worst” Analysis views are created, assign them to Setup and Hold.

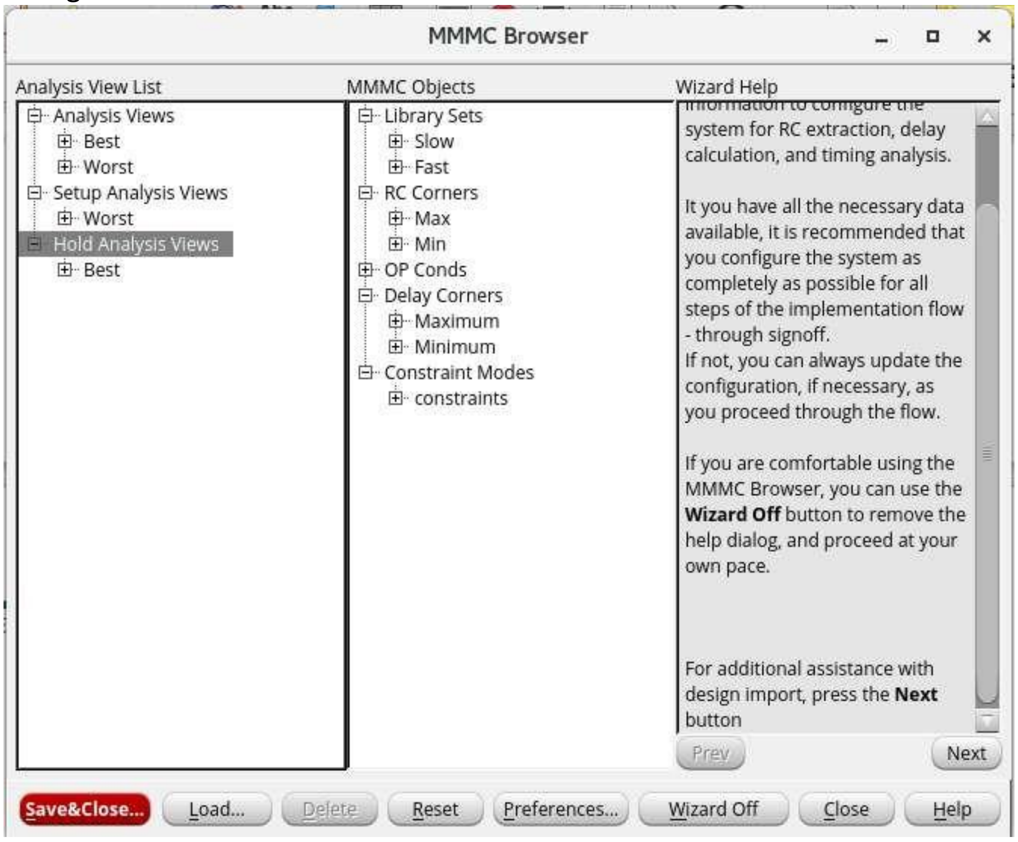

Add Setup Analysis View

Analysis View: Worst


Add Hold Analysis View

Analysis View: Best

- Once all the process is done, Click on “Save&Close” and save the script generated with any name of your choice.
- Make sure the file extension remains .view or .tcl
- After saving the script, go back to Import Design window and Click “OK” to load your design.


MMMC Browser

Analysis View List
- Analysis Views
  - Best
  - Worst
- Setup Analysis Views
  - Worst
- Hold Analysis Views
  - Best

MMMC Objects
- Library Sets
  - Slow
  - Fast
- RC Corners
  - Max
  - Min
- OP Conds
- Delay Corners
  - Maximum
  - Minimum
- Constraint Modes
  - constraints

Wizard Help

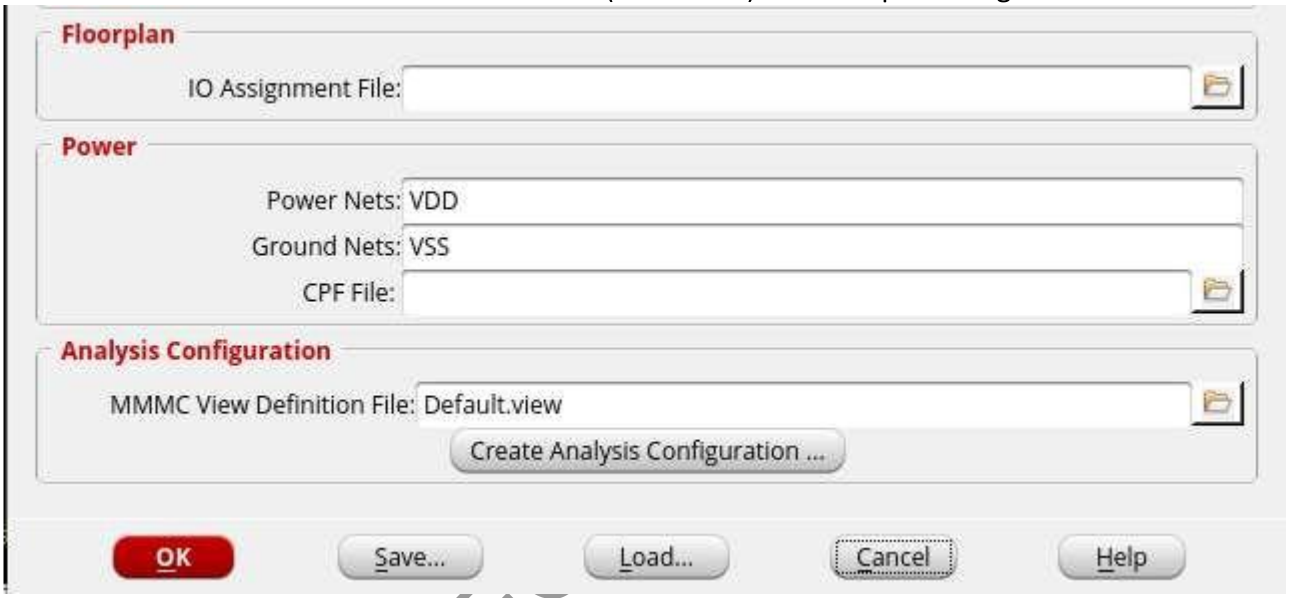
information to configure the system for RC extraction, delay calculation, and timing analysis.

It you have all the necessary data available, it is recommended that you configure the system as completely as possible for all steps of the implementation flow - through signoff.
If not, you can always update the configuration, if necessary, as you proceed through the flow.

If you are comfortable using the MMMC Browser, you can use the **Wizard Off** button to remove the help dialog, and proceed at your own pace.

For additional assistance with design import, press the **Next** button

- Add Power and Ground Net names (Identifiers) under Import design window.



Floorplan

IO Assignment File:

Power

Power Nets: VDD

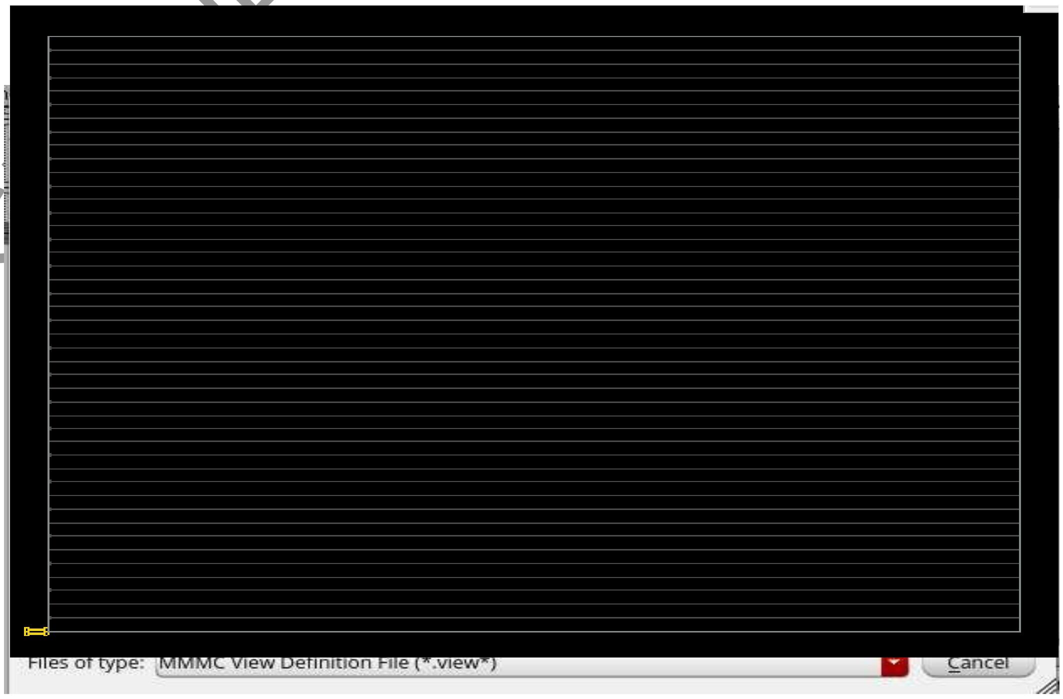
Ground Nets: VSS

CPF File:

Analysis Configuration

MMMC View Definition File: Default.view

- A rectangular or square box appears in your GUI if and only if all the inputs are readproperly.
- If the box does not appear, check for errors in your log (Either on terminal or log file from pwd)

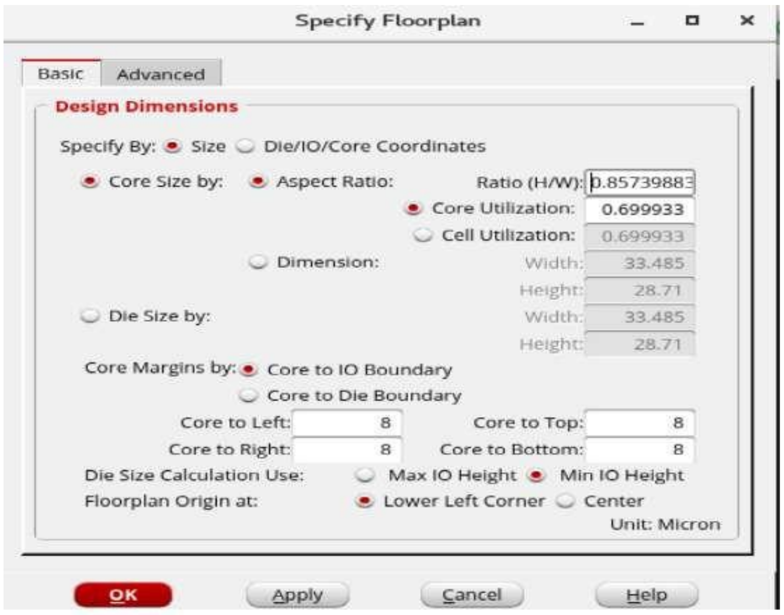


- The internal area of the box is called "Core Area".
- The horizontal lines running along the width of Core are "Standard Cell Rows". Every alternate of them are marked indicating alternate VDD and VSS rows.
- This setup is called "**Flipped Standard Cell Rows**".

→ **Floorplan**

Steps under Floorplan :

1. Aspect Ratio [Ratio of Vertical Height to Horizontal Width of Core]
2. Core Utilisation [The total Core Area % to be used for Floor Planning]
3. Channel Spacing between Core Boundary to IO Boundary
    - Select Floorplan → Specify Floorplan to modify/add concerned values to the above Factors. On adding/modifying the concerned values, the core area is also modified



Specify Floorplan

Basic | Advanced

Design Dimensions

Specify By: [✓ Size] Die/IO/Core Coordinates

[✓ Core Size by:] [✓ Aspect Ratio:] Ratio (H/W): 0.85739883

[✓ Core Utilization:] 0.699933

Cell Utilization: 0.699933

Dimension: Width: 33.485 Height: 28.71

Die Size by: Width: 33.485 Height: 28.71

Core Margins by: [✓ Core to IO Boundary] Core to Die Boundary

Core to Left: 8 Core to Top: 8

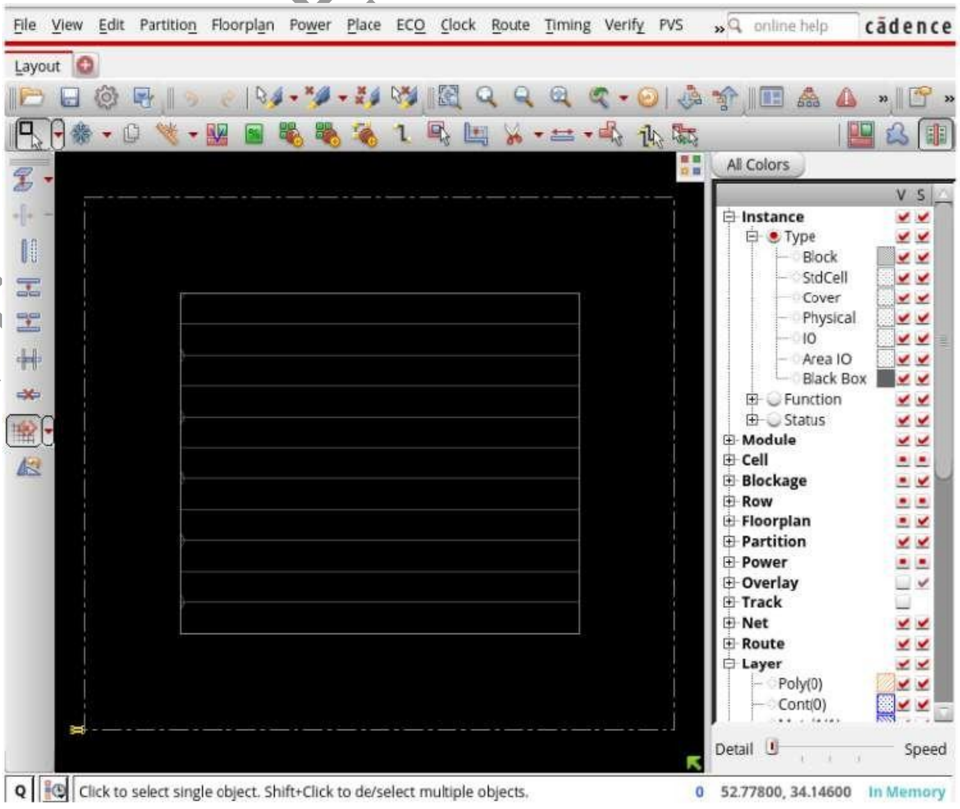
Core to Right: 8 Core to Bottom: 8

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

Unit: Micron

OK Apply Cancel Help



Click to select single object. Shift+Click to de/select multiple objects. 0 52.77800, 34.14600 In Memory

- The Yellow patch on the Left Bottom are the group of "Unassigned pins" which are to be placed along the IO Boundary along with the Standard Cells [Gates].

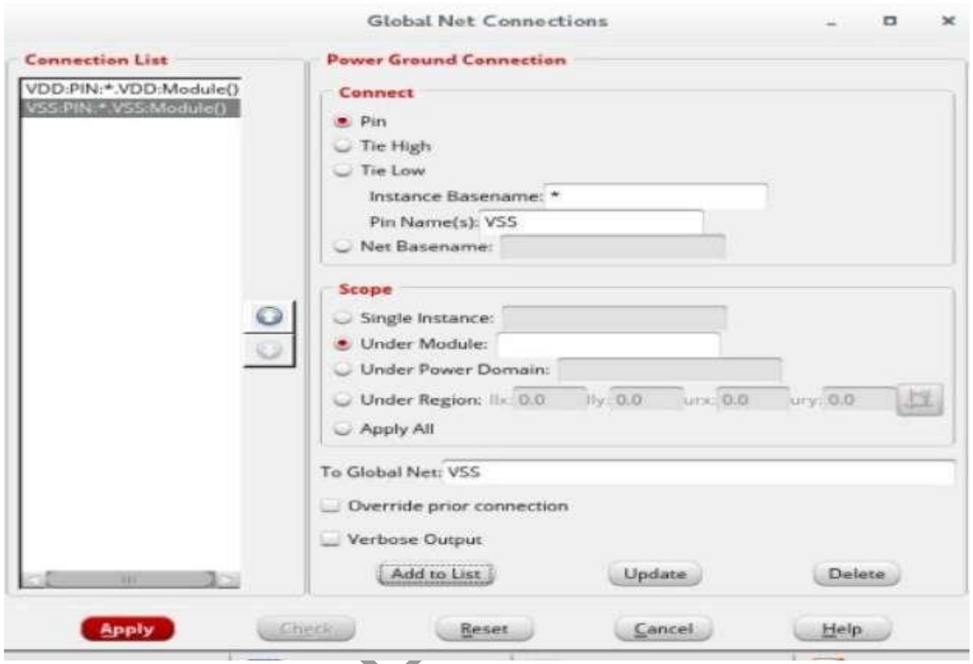
→ Power Planning

Steps under Power Planning :

1. Connect Global Net Connects
2. Adding Power Rings
3. Adding Power Strings
4. Special Route

Under Connect Global Net Connects, we create two pins, one for VDD and one for VSS connecting them to corresponding Global Nets as mentioned in Globals file / Power and Ground Nets.

1. Select Power → Connect Global Nets.. to create "Pin" and "Connect to Global Net" asshown and use "Add to list".
2. Click on "Apply" to direct the tool in enforcing the Pins and Net connects to Design and then Close the window.



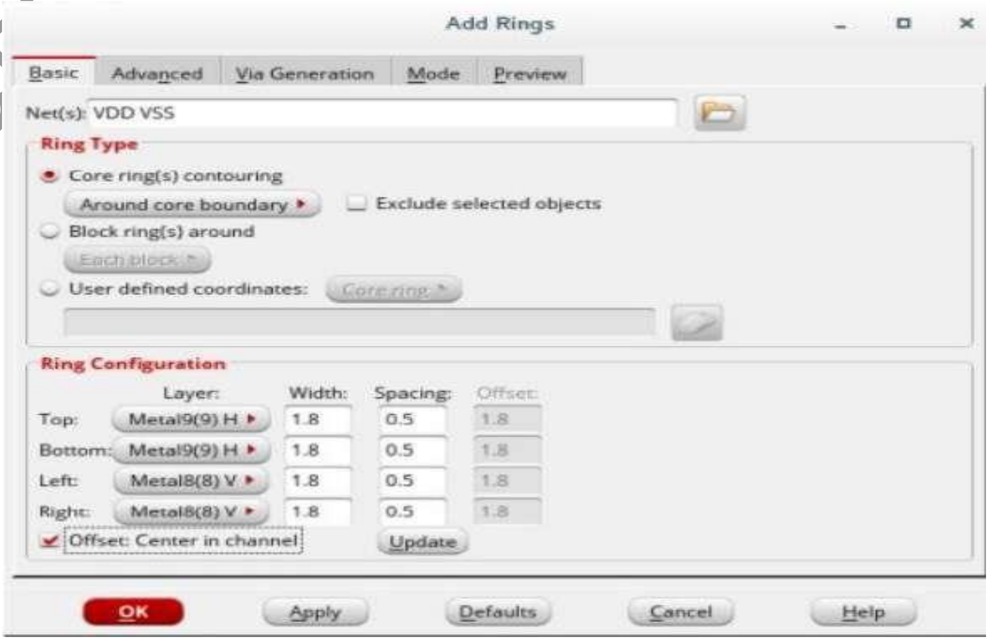
Connection List: VDD:PIN:*.VDD:Module() ; VSS:PIN:*.VSS:Module()

Connect: [✓ Pin] Tie High / Tie Low
Instance Basename: *
Pin Name(s): VSS
Net Basename:

Scope: Single Instance: / [✓ Under Module:] / Under Power Domain: / Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0 / Apply All

To Global Net: VSS
Override prior connection
Verbose Output

- In order to Tap in Power from a distant Power supply, Wider Nets and Parallel connectionsimprove efficiency. Moreover, the cells that would be placed inside the core area are expected to have shorter Nets for lower resistance.
- Hence Power Rings [Around Core Boundary] and Power Stripes [Across Core Boundary]are added which satisfies the above conditions.
- Select Power → Power Planning → Add Rings to add Power rings 'around Core Boundary'.



Net(s): VDD VSS

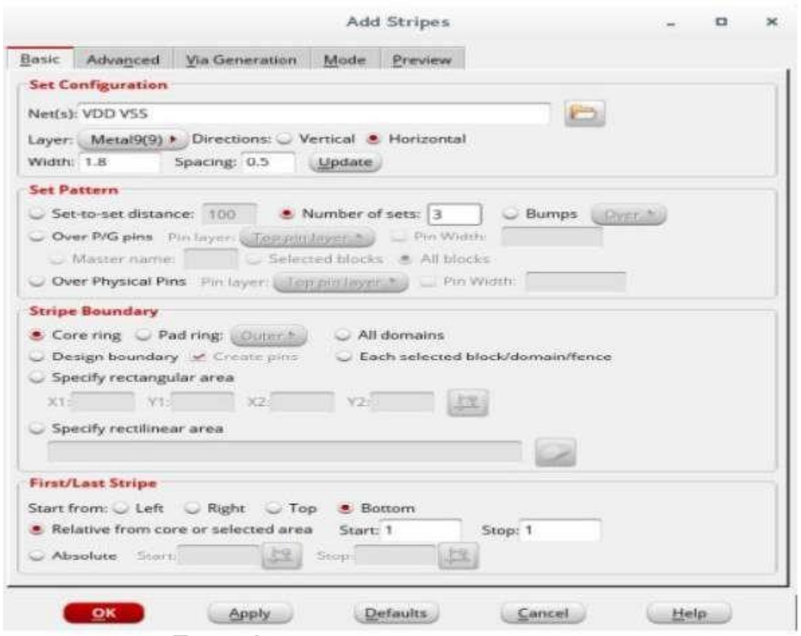
Ring Type: [✓ Core ring(s) contouring] Around core boundary / Exclude selected objects / Block ring(s) around: Each block / User defined coordinates: Core ring

	Layer:	Width:	Spacing:	Offset:
Top:	Metal9(9) H	1.8	0.5	1.8
Bottom:	Metal9(9) H	1.8	0.5	1.8
Left:	Metal8(8) V	1.8	0.5	1.8
Right:	Metal8(8) V	1.8	0.5	1.8

✓ Offset: Center in channel

- Select the Nets from Browse option OR Directly type in the Global Net Names separated by a space being Case and Spelling Sensitive.
- Select the Highest Metals marked 'H' [Horizontal] for Top and Bottom and Metals marked 'V' [Vertical] for Right and Bottom. This is because Highest metals have Highest Widths and thus Lowest Resistance.
- Click on Update after the selection and "Set Offset : Centre in Channel" in order to get the Minimum Width and Minimum Spacing of the corresponding Metals and then Click "OK".

- Similarly, Power Stripes are added using similar content to that of Power Rings


Add Stripes

Basic | Advanced | Via Generation | Mode | Preview

Set Configuration
Net(s): VDD VSS
Layer: Metal9(9) Directions: ○ Vertical ● Horizontal
Width: 1.8 Spacing: 0.5 Update

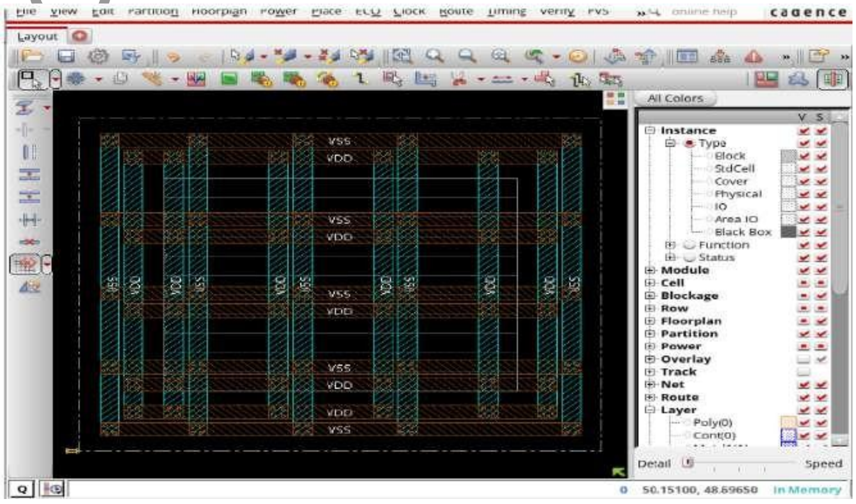
Set Pattern
○ Set-to-set distance: 100 ● Number of sets: 3 ○ Bumps Over
○ Over P/G pins Pin layer: Top pin layer Pin Width:
Master name: ○ Selected blocks ● All blocks
○ Over Physical Pins Pin layer: Top pin layer Pin Width:

Stripe Boundary
● Core ring ○ Pad ring: Outer ○ All domains
○ Design boundary ✓ Create pins ○ Each selected block/domain/fence
○ Specify rectangular area
X1: Y1: X2: Y2:
○ Specify rectilinear area

First/Last Stripe
Start from: ○ Left ○ Right ○ Top ● Bottom
● Relative from core or selected area Start: 1 Stop: 1
○ Absolute Start: Stop:

OK Apply Defaults Cancel Help

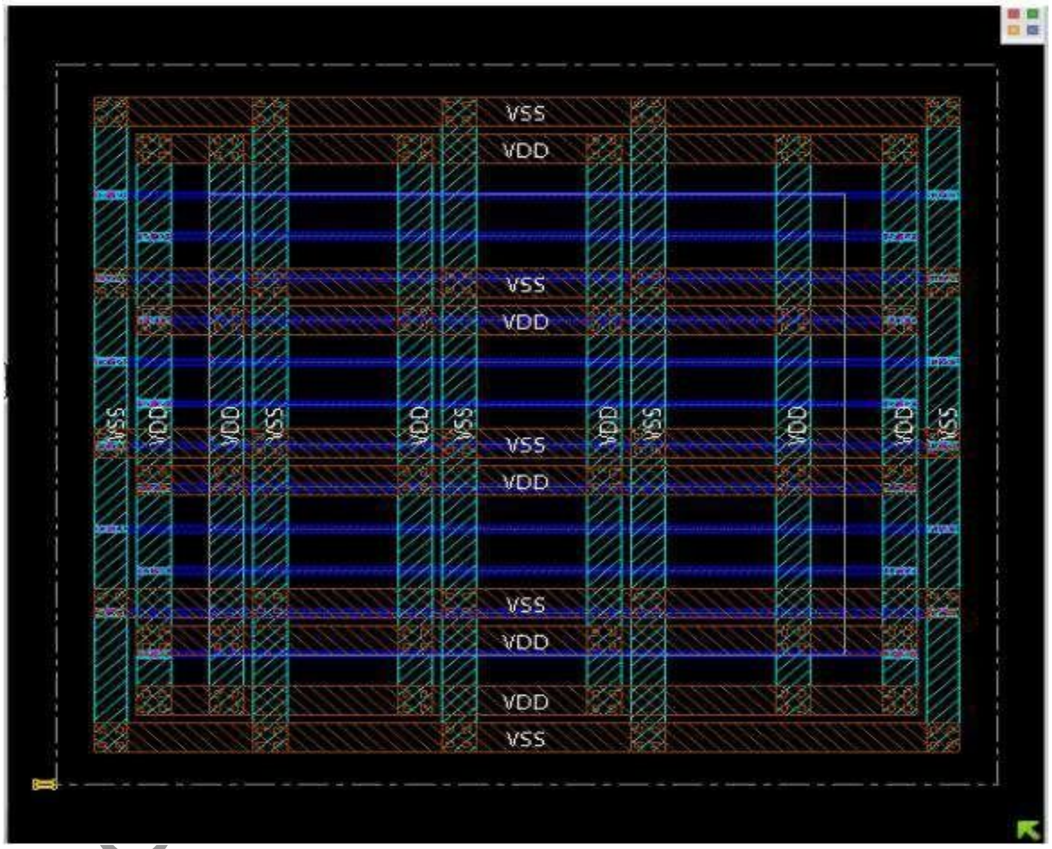
Factors to be considered under Power Stripes :
→ Nets
→ Metal and It’s Direction
→ Width and Spacing [Updated]
→ Set to Set Distance = (Minimum Width of Metal + Min. Spacing) x 2



- On adding Power Stripes, The Power mesh setup is complete as shown. However, There are no Vias that could connect Metal 9 or Metal 8 directly with Metal 1 [VDD or VSS of Standard Cells are generally made up of Metal 1].
- The connection between the Highest and Lowest Metals is done through Stacking of Vias done using “Special Route”.
- To perform Special Route, Select Route → Special Route → Add Nets → OK.
- After the Special Route is complete, all the Standard Cell Rows turn to the Color coded forMetal 1 as shown below.

The complete Power Planning process makes sure Every Standard Cell receives enough power tooperate smoothly.

→ **Pre – Placement** :

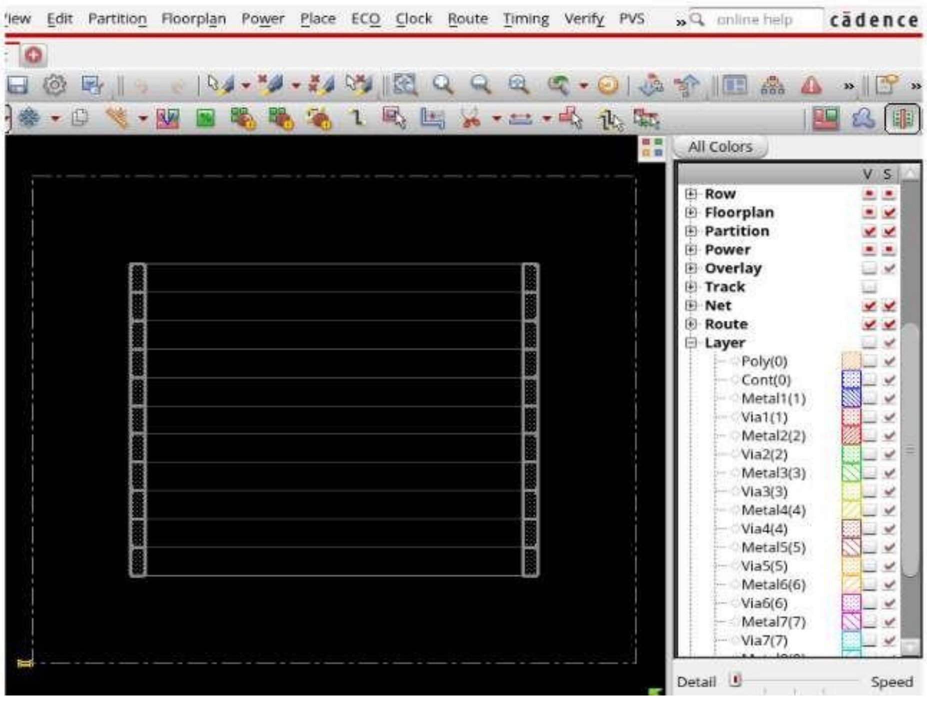


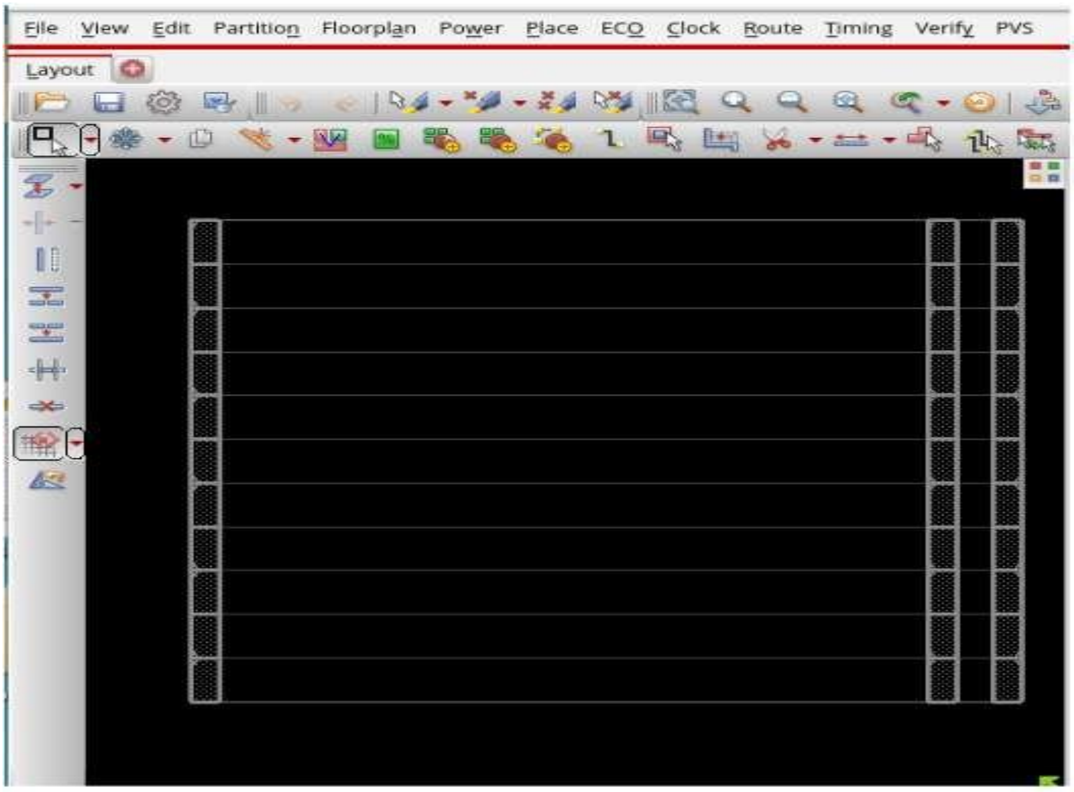
- After Power Planning, a few Physical Cells are added namely, End Caps and Well Taps.
- <u>End Caps :</u> They are Physical Cells which are added to the Left and Right Core Boundariesacting as blockages to avoid Standard Cells from moving out of boundary.
- <u>Well Taps :</u> They act like Shunt Resistance to avoid Latch Up effects.

To add End Caps, Select Place → Physical Cell → Add End Caps and "Select" the FILL's from the available list.

- Higher Fills have Higher Widths. As shown Below, The End Caps are added below yourPower Mesh.

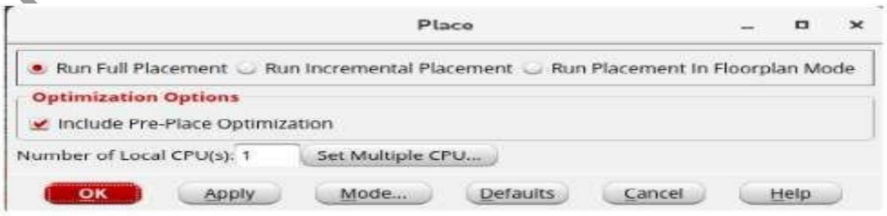
- To add Well Taps, Select Place → Physical Cell → Add Well Tap → Select →FillX [X → Strength of Fill = 1,2,4 etc] → Distance Interval [Could be given in range of 30-45u] → OK




File View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS
Layout

→ **Placement**

1. The Placement stage deals with Placing of Standard Cells as well as Pins.
2. Select Place → Place Standard Cell → Run Full Placement → Mode → Enable 'Place I/O Pins' → OK → OK .



Place

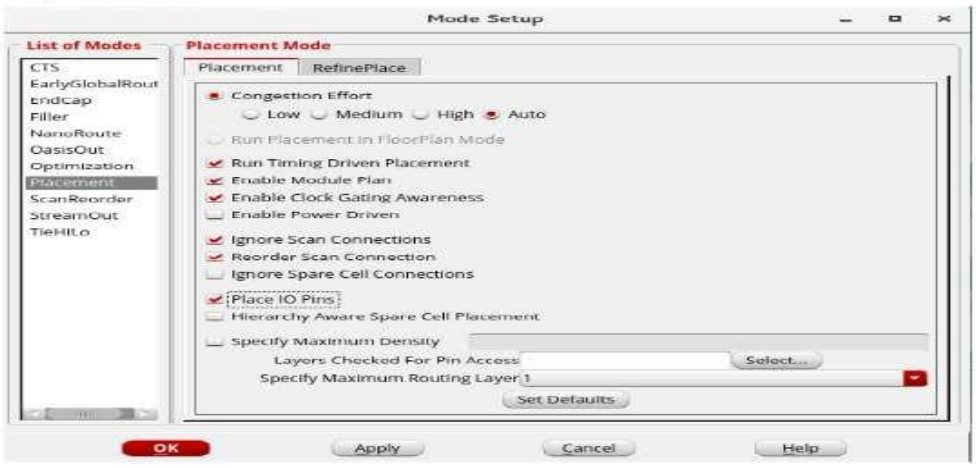
[✓ Run Full Placement] Run Incremental Placement Run Placement In Floorplan Mode

Optimization Options

[✓] Include Pre-Place Optimization

Number of Local CPU(s): 1 Set Multiple CPU...

OK Apply Mode... Defaults Cancel Help



Mode Setup

List of Modes: CTS, EarlyGlobalRout, EndCap, Filler, NanoRoute, OasisOut, Optimization, Placement, ScanReorder, StreamOut, TieHiLo

Placement Mode

Placement | RefinePlace

[✓] Congestion Effort
Low Medium High [✓ Auto]

Run Placement In FloorPlan Mode

[✓] Run Timing Driven Placement
[✓] Enable Module Plan
[✓] Enable Clock Gating Awareness
[ ] Enable Power Driven

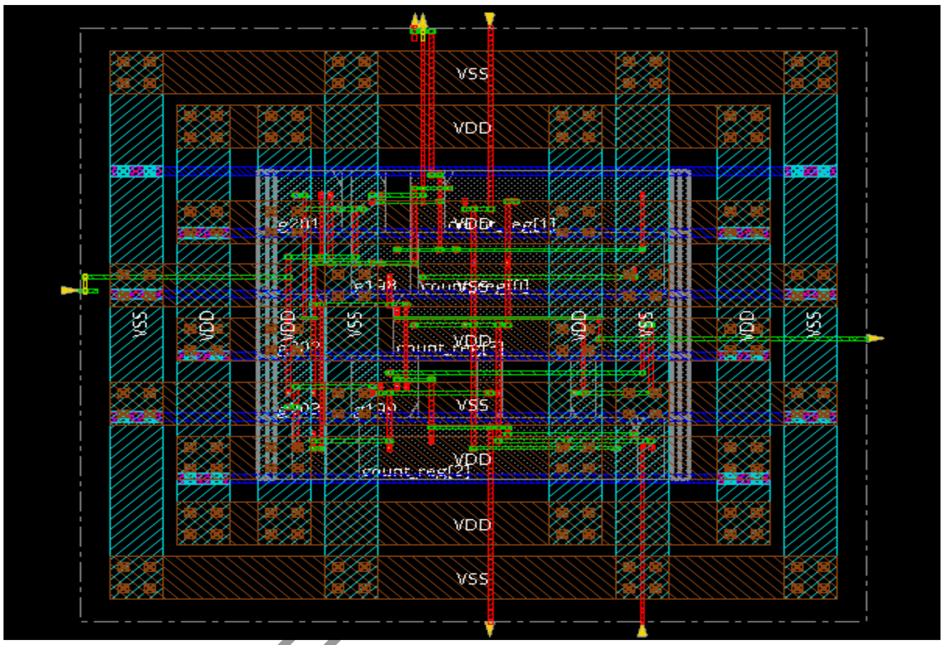
[✓] Ignore Scan Connections
[✓] Reorder Scan Connection
[ ] Ignore Spare Cell Connections

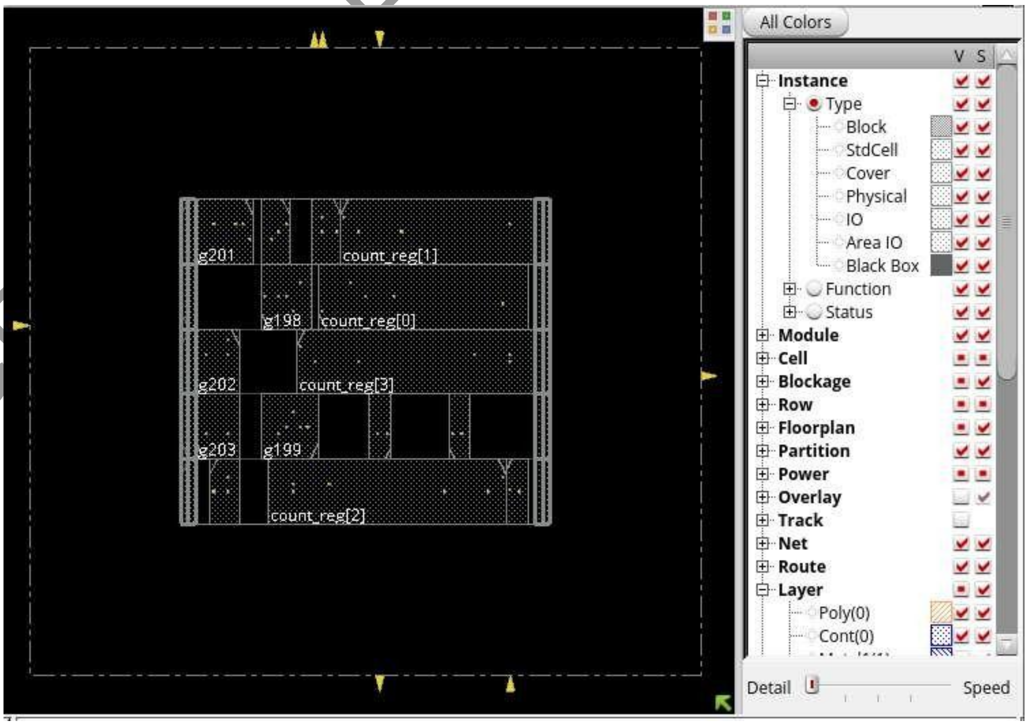
[✓] Place IO Pins
[ ] Hierarchy Aware Spare Cell Placement

[ ] Specify Maximum Density
Layers Checked For Pin Access Select...
Specify Maximum Routing Layer 1
Set Defaults

OK Apply Cancel Help

- All the Standard Cells and Pins are placed as per the communication between them, i.e., Two communicating Cells are placed as close as possible so that shorter Net lengths can be used for connections as Shorter Net Lengths enable Better Timing Results.



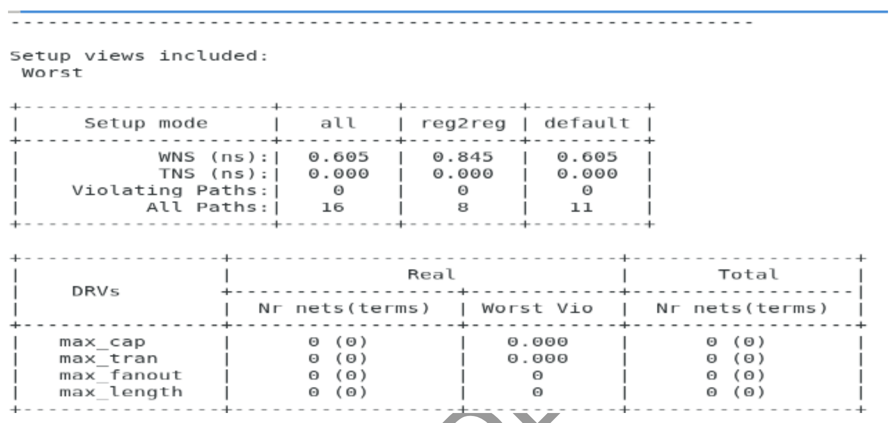


- You can toggle the Layer Visibility from the list on the Right. The List of Layers available are shown on the right under "Layer" tab with colour coding.

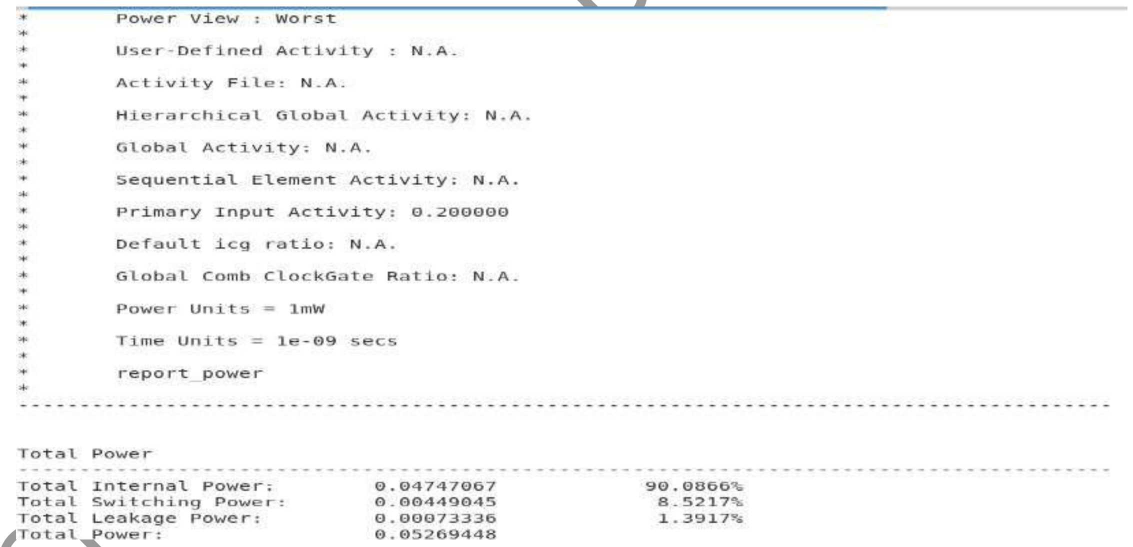
→ <u>Report Generation and Optimization</u> :

1. Timing Report :
   1. To generate Timing Report, Timing → Report Timing → Design Stage – PreCTS
   2. Analysis Type – Setup → OK
   3. The Timing report Summary can be seen on the Terminal.
2. Area Report :
   1. **cmd :** report_area
3. Power Report :
   1. **cmd :** report_power

```
innovus 9> innovus 9> report_area
Depth  Name       #Inst  Area (um^2)
------------------------------------
0      counter    15     134.7282
```

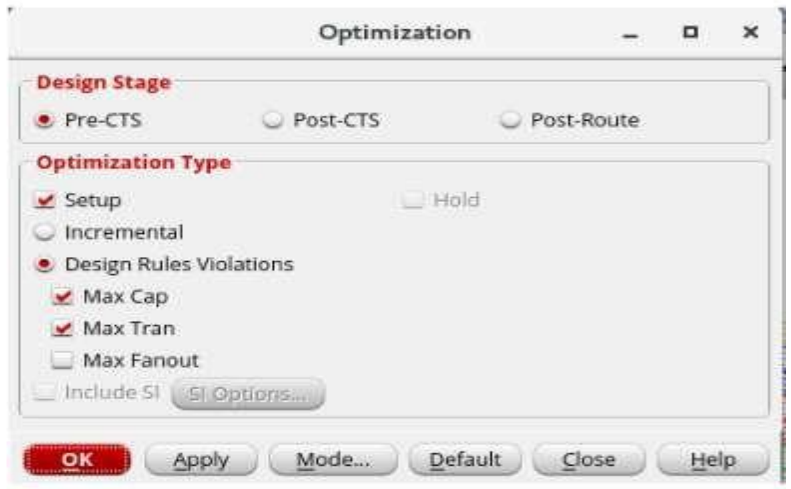


```
------------------------------------------------------------

Setup views included:
 Worst

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.605  |  0.845  |  0.605  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   16    |    8    |   11    |
+--------------------+---------+---------+---------+

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



```
*        Power View : Worst
*
*        User-Defined Activity : N.A.
*
*        Activity File: N.A.
*
*        Hierarchical Global Activity: N.A.
*
*        Global Activity: N.A.
*
*        Sequential Element Activity: N.A.
*
*        Primary Input Activity: 0.200000
*
*        Default icg ratio: N.A.
*
*        Global Comb ClockGate Ratio: N.A.
*
*        Power Units = 1mW
*
*        Time Units = 1e-09 secs
*
*        report_power
*
-----------------------------------------------------------------------------------------

Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:        0.04747067            90.0866%
Total Switching Power:       0.00449045             8.5217%
Total Leakage Power:         0.00073336             1.3917%
Total Power:                 0.05269448
```

- In case of any Violating paths, the design could be optimized in the following way.
- To optimize the Design, **Select ECO → Optimize Design → Design Stage [PreCTS] →** Optimization Type – Setup → OK



Optimization

Design Stage
[✓ Pre-CTS] Post-CTS Post-Route

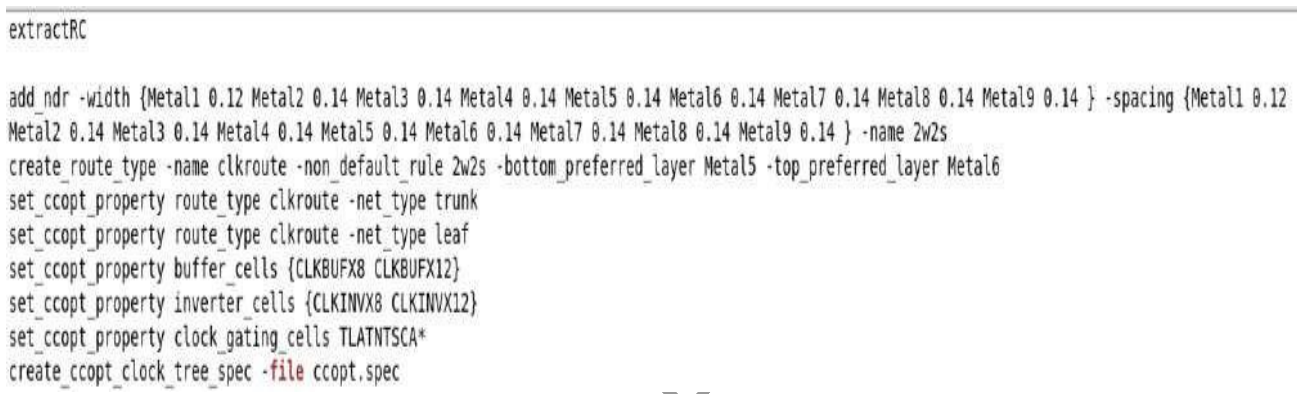
Optimization Type
[✓ Setup] Hold
Incremental
[✓ Design Rules Violations]
[✓ Max Cap]
[✓ Max Tran]
Max Fanout
Include SI SI Options...

OK Apply Mode... Default Close Help

- After you run the optimization, the terminal displays the latest Timing report and updated area and power reports can be checked.
- This step Optimizes your design in terms of Timing, Area and Power. You can GenerateTiming, Area, Power in similar way as above report Post – Optimization to compare theReports.

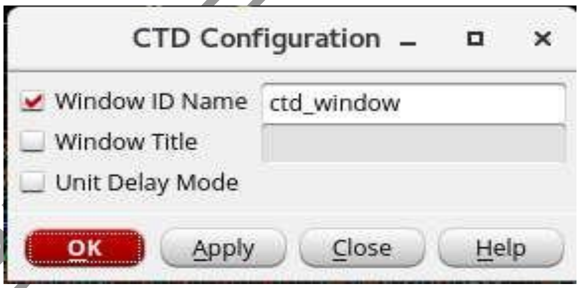
Clock Tree Synthesis

- The CTS Stage is meant to build a Clock Distribution Network such that every Register (Flip Flop) acquires Clock at the same time (Atleast Approximately) to keep them in propercommunication.
- A Script can be used to Build the Clock Tree as follows :

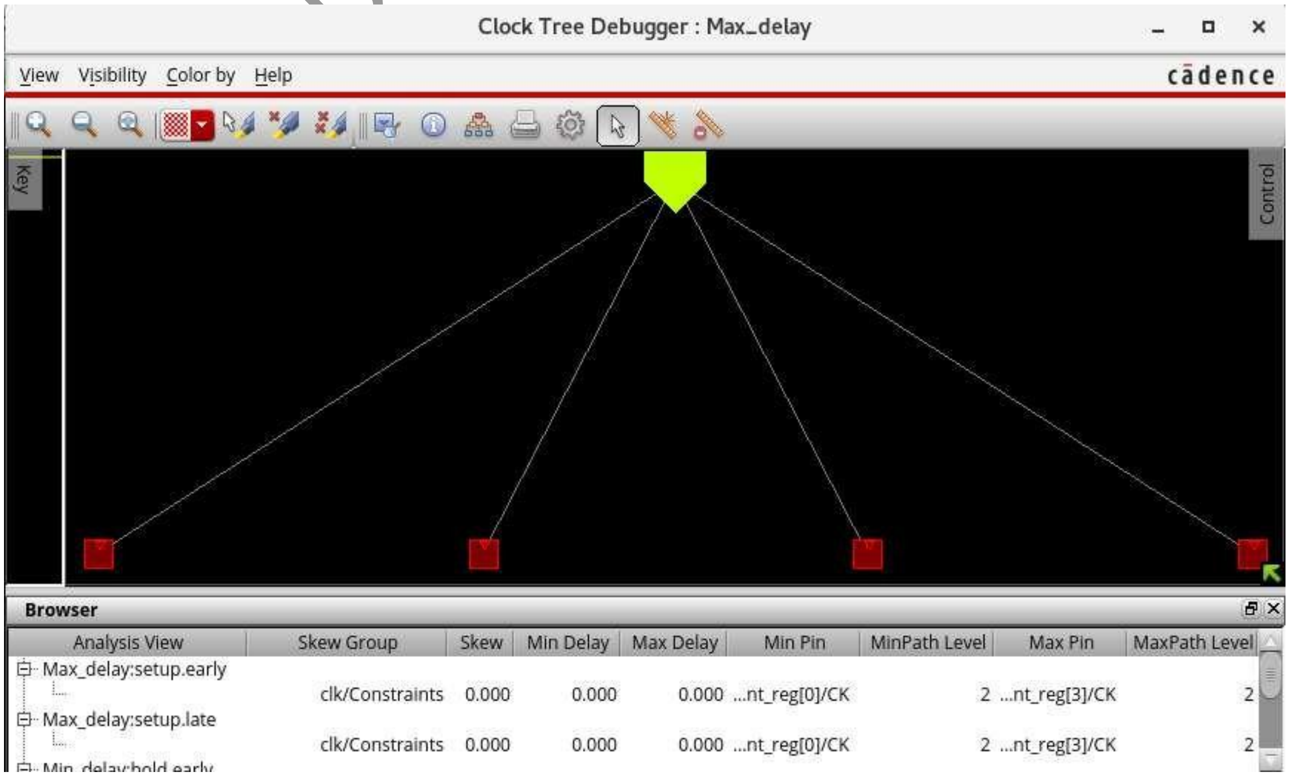


```
extractRC

add_ndr -width {Metal1 0.12 Metal2 0.14 Metal3 0.14 Metal4 0.14 Metal5 0.14 Metal6 0.14 Metal7 0.14 Metal8 0.14 Metal9 0.14 } -spacing {Metal1 0.12
Metal2 0.14 Metal3 0.14 Metal4 0.14 Metal5 0.14 Metal6 0.14 Metal7 0.14 Metal8 0.14 Metal9 0.14 } -name 2w2s
create_route_type -name clkroute -non_default_rule 2w2s -bottom_preferred_layer Metal5 -top_preferred_layer Metal6
set_ccopt_property route_type clkroute -net_type trunk
set_ccopt_property route_type clkroute -net_type leaf
set_ccopt_property buffer_cells {CLKBUFX8 CLKBUFX12}
set_ccopt_property inverter_cells {CLKINVX8 CLKINVX12}
set_ccopt_property clock_gating_cells TLATNTSCA*
create_ccopt_clock_tree_spec -file ccopt.spec
```

- Source the Script as shown in the above snapshot through the Terminal and then Select Clock → CCOpt Clock Tree Debugger → OK to build and view clock tree.



Window ID Name [✓] ctd_window
Window Title
Unit Delay Mode


Clock Tree Debugger : Max_delay
View Visibility Color by Help
cadence
Key
Control
Browser

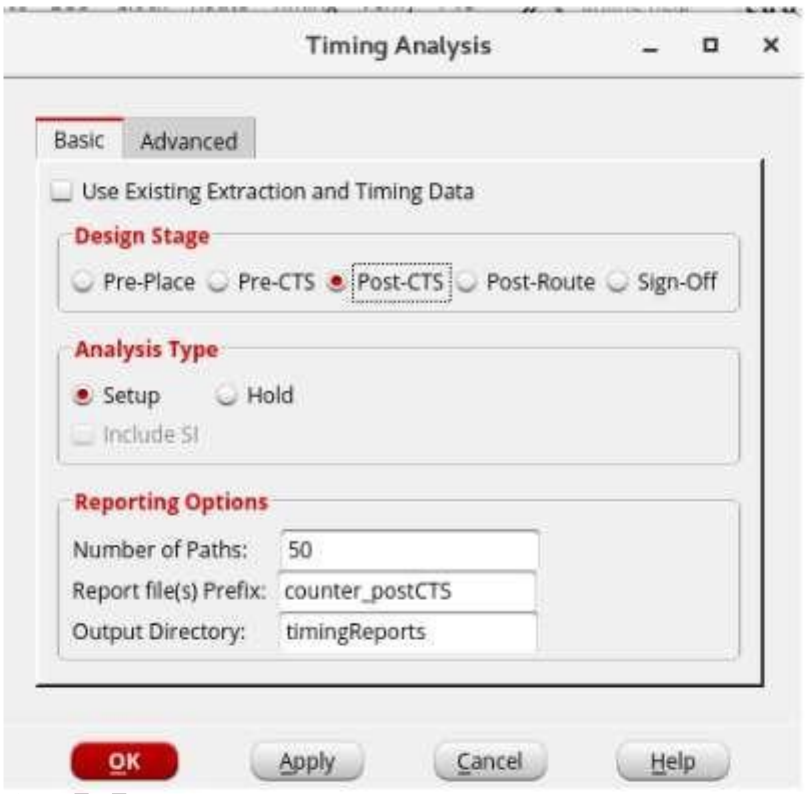
```
Analysis View            Skew Group       Skew   Min Delay  Max Delay  Min Pin          MinPath Level  Max Pin          MaxPath Level
Max_delay:setup.early
                         clk/Constraints  0.000  0.000      0.000      ...nt_reg[0]/CK  2              ...nt_reg[3]/CK  2
Max_delay:setup.late
                         clk/Constraints  0.000  0.000      0.000      ...nt_reg[0]/CK  2              ...nt_reg[3]/CK  2
Min_delay:hold.early
```

- The Red Boxes are the Clock Pins of various Flip Flops in the Design while Yellow Pentagon on the top represents Clock Source.
- The Clock Tree is built with Clock Buffers and Clock Inverters added to boost up the ClockSignal.

<u>Report Generation and Design Optimization :</u>

- CTS Stage adds real clock into the Design and hence "Hold" Analysis also becomes prominent. Hence, Optimizations can be done for both Setup & Hold, Timing Reports are tobe Generated for Setup and Hold Individually.

Setup Timing Analysis :


Timing Analysis

Basic | Advanced

Use Existing Extraction and Timing Data

Design Stage
Pre-Place Pre-CTS [✓ Post-CTS] Post-Route Sign-Off

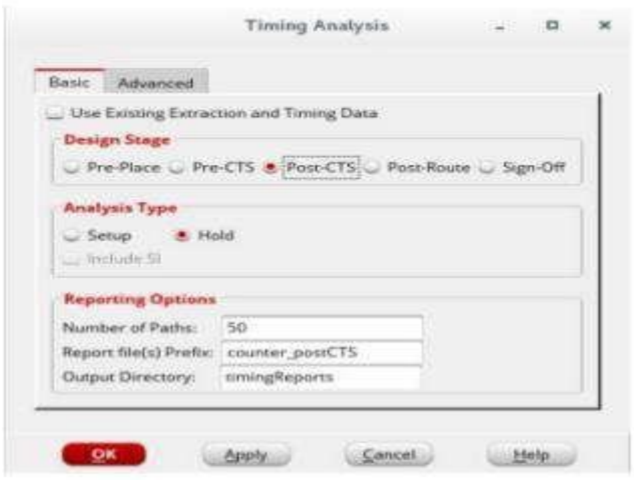
Analysis Type
[✓ Setup] Hold
Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postCTS
Output Directory:	timingReports

OK Apply Cancel Help

Hold Timing Analysis :


Timing Analysis

Basic | Advanced

Use Existing Extraction and Timing Data

Design Stage
Pre-Place Pre-CTS [✓ Post-CTS] Post-Route Sign-Off

Analysis Type
Setup [✓ Hold]
Include SI

Reporting Options

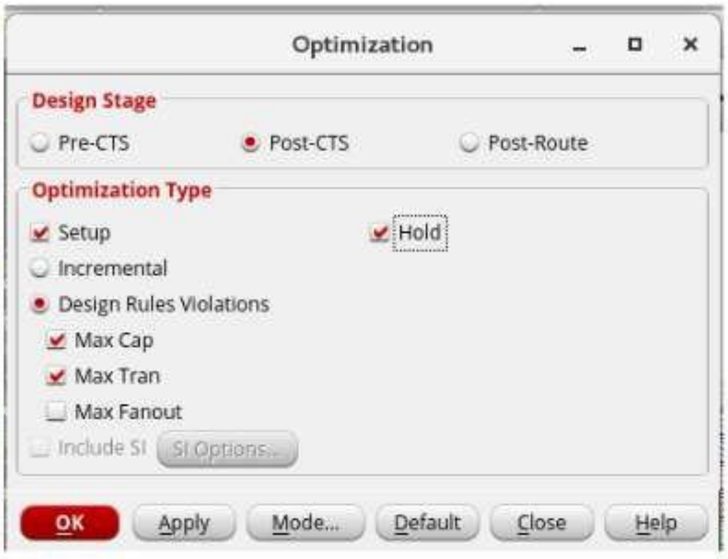
Number of Paths:	50
Report file(s) Prefix:	counter_postCTS
Output Directory:	timingReports

OK Apply Cancel Help

For Area and Power Report Generation,

report_area & report_power commands can be used.

Design Optimizations :


Optimization

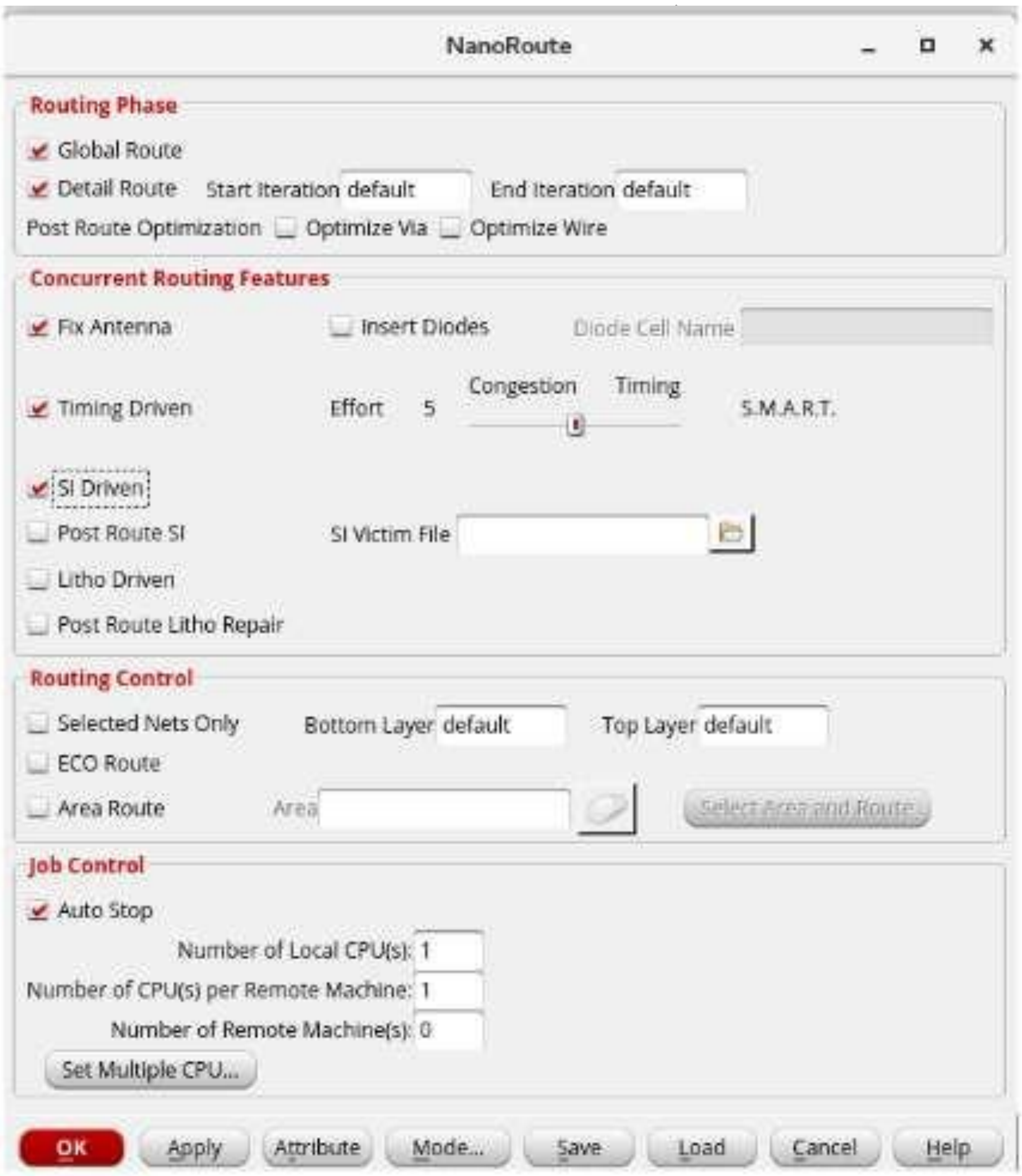
Design Stage
Pre-CTS [✓ Post-CTS] Post-Route

Optimization Type
[✓ Setup] [✓ Hold]
Incremental
[✓ Design Rules Violations]
[✓ Max Cap]
[✓ Max Tran]
Max Fanout
Include SI SI Options...

OK Apply Mode... Default Close Help

Routing :

1. All the net connections shown in the GUI till CTS are only based onthe Logical connectivity.
2. These connections are to be replaced with real Metals avoiding Opens, Shorts, Signal Integrity [Cross Talks], Antenna Violations etc.
3. To run Routing, Select Route → Nano Route → Route and enable Timing Driven and SI Driven for Design Physical Efficiency and Reliability.



NanoRoute

Routing Phase
- [✓] Global Route
- [✓] Detail Route — Start Iteration: default — End Iteration: default
- Post Route Optimization: [ ] Optimize Via [ ] Optimize Wire

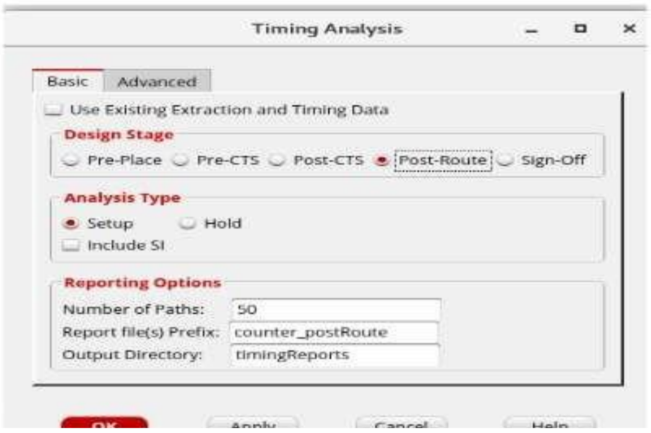
Concurrent Routing Features
- [✓] Fix Antenna — [ ] Insert Diodes — Diode Cell Name:
- [✓] Timing Driven — Effort 5 — Congestion / Timing — S.M.A.R.T.
- [✓] SI Driven
- [ ] Post Route SI — SI Victim File:
- [ ] Litho Driven
- [ ] Post Route Litho Repair

Routing Control
- [ ] Selected Nets Only — Bottom Layer: default — Top Layer: default
- [ ] ECO Route
- [ ] Area Route — Area: — Select Area and Route

Job Control
- [✓] Auto Stop
- Number of Local CPU(s): 1
- Number of CPU(s) per Remote Machine: 1
- Number of Remote Machine(s): 0
- Set Multiple CPU...

Report Generation and Design Optimization :

Setup Report :


Timing Analysis

Basic | Advanced

☐ Use Existing Extraction and Timing Data

Design Stage: Pre-Place Pre-CTS Post-CTS [✓ Post-Route] Sign-Off

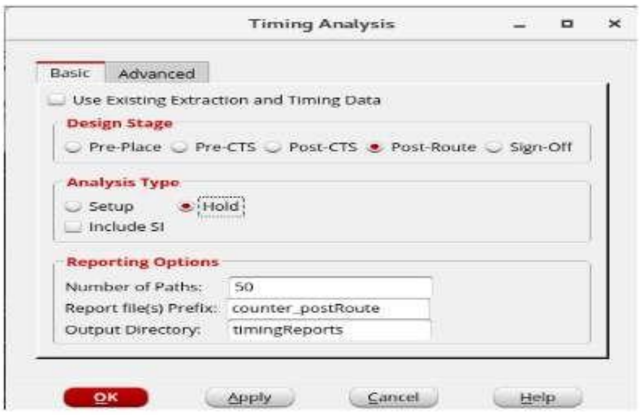
Analysis Type: [✓ Setup] Hold

☐ Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postRoute
Output Directory:	timingReports

Hold Report :


Timing Analysis

Basic | Advanced

☐ Use Existing Extraction and Timing Data

Design Stage: Pre-Place Pre-CTS Post-CTS [✓ Post-Route] Sign-Off

Analysis Type: Setup [✓ Hold]

☐ Include SI

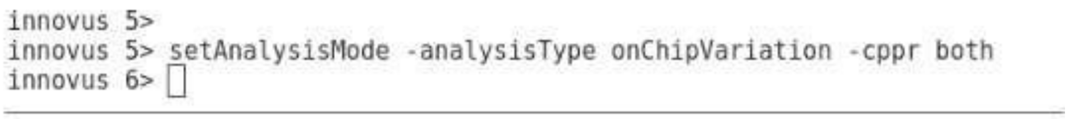
Reporting Options

Number of Paths:	50
Report file(s) Prefix:	counter_postRoute
Output Directory:	timingReports

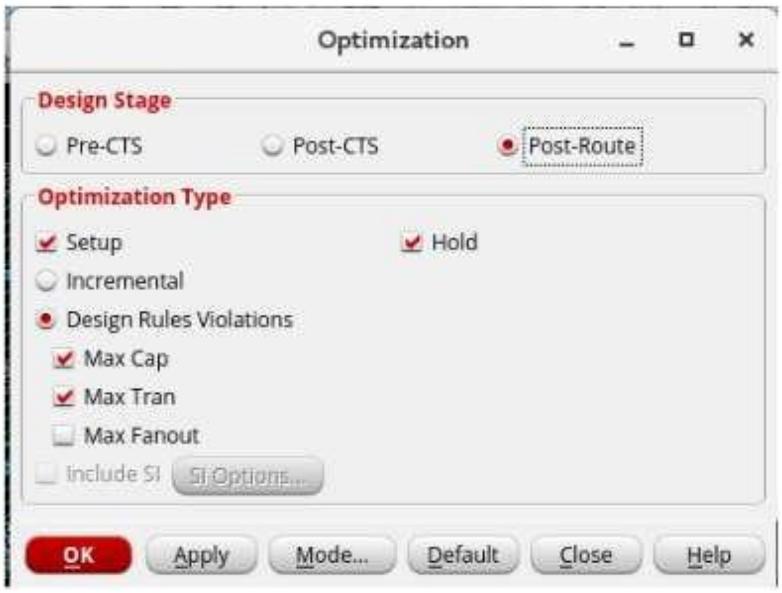
Area and Power Reports :

Use the commands report_area and report_power for Area and PowerReports respectively.

Design Optimization :


```
innovus 5>
innovus 5> setAnalysisMode -analysisType onChipVariation -cppr both
innovus 6>
```

Enter the above shown command in the Terminal in order to run the Design Optimization first Post-Route.


Optimization

Design Stage: Pre-CTS Post-CTS [✓ Post-Route]

Optimization Type

[✓ Setup] [✓ Hold]

Incremental

[✓ Design Rules Violations]

[✓ Max Cap]

[✓ Max Tran]

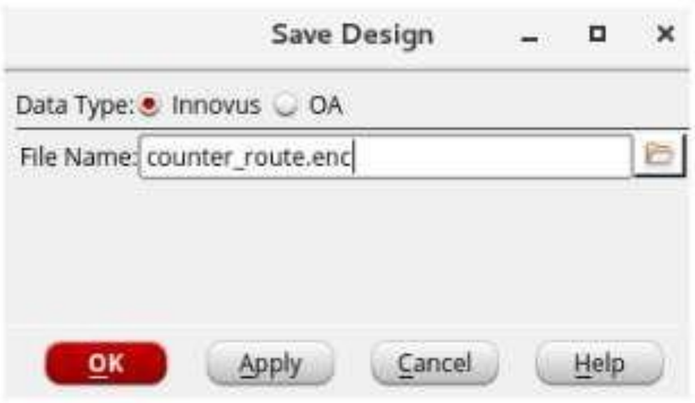
☐ Max Fanout

☐ Include SI

- As an alternate to the setAnalysisMode command, you can use the GUI at Tools → SetMode → Set Analysis Mode → Select On-Chip-Variation and CPPR.
- The Report generation is same as shown prior to Design Optimization.

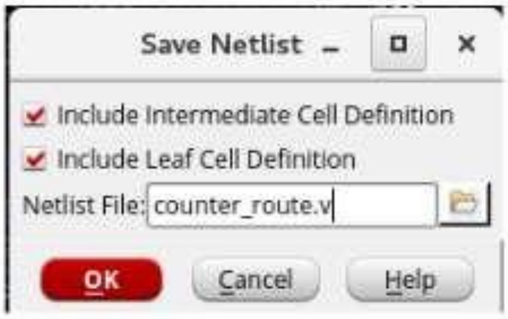
Saving Database :

1. **Saving Design => File → Save Design → Data Type : Innovus → <DesignName>.enc → OK**



Data Type: [✓ Innovus] OA
File Name: counter_route.enc

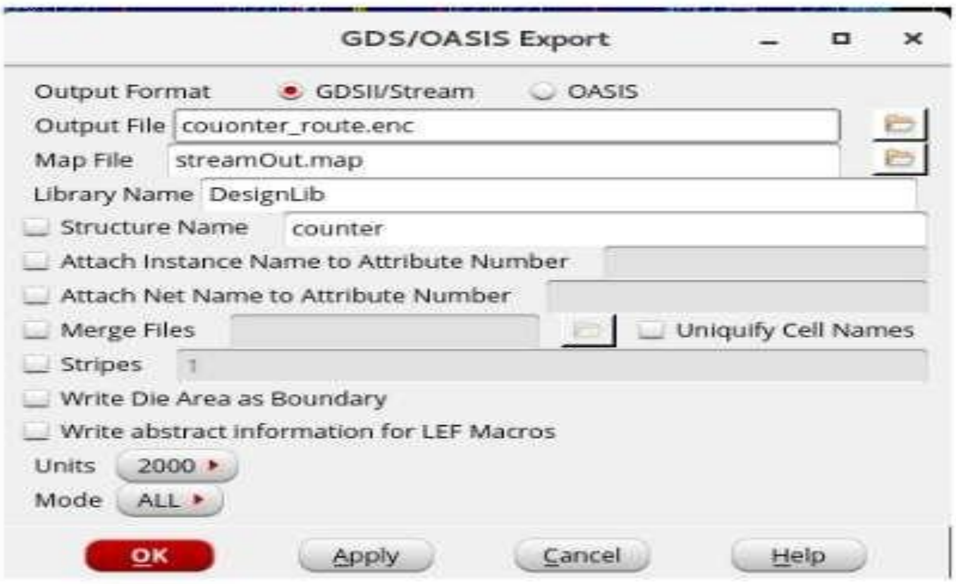
2. **Saving Netlist => File → Save → Netlist → <NetlistName>.v → OK**



✓ Include Intermediate Cell Definition
✓ Include Leaf Cell Definition
Netlist File: counter_route.v

It is recommended to save Netlist and Design at every stage. To restore a Design Data Base, type source <DesignName>.enc in the terminal.

3. **Saving GDS => File → Save → GDS/OASIS → <FileName>.gds → OK**



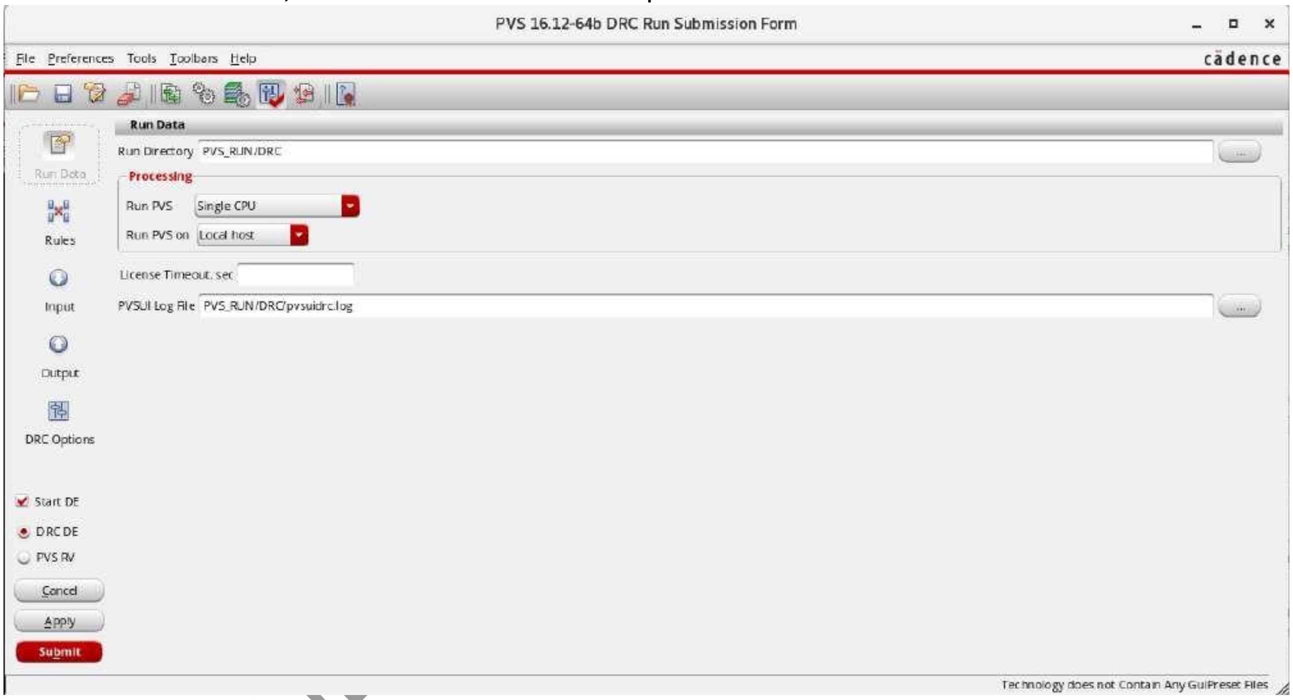
Output Format [✓ GDSII/Stream] OASIS
Output File couonter_route.enc
Map File streamOut.map
Library Name DesignLib
Structure Name counter
Attach Instance Name to Attribute Number
Attach Net Name to Attribute Number
Merge Files
Uniquify Cell Names
Stripes 1
Write Die Area as Boundary
Write abstract information for LEF Macros
Units 2000
Mode ALL

<u>Physical Verification – Capturing DRC and LVS :</u>

- After saving the routed Database, you can proceed for Physical Verification and capture the DRC and LVS reports.
- Inputs Required – DRC :
  - Technology Library and Rule Set
  - GDS format giles of all Standard Cells (Given by Cadence at /home/install/FOUNDRY/90nm/dig/gds for 90nm Tech node)
- Outputs – DRC :

- DRC Violation Report
- Physical Netlist (Optional)

From the Innovus GUI, select PVS → Run DRC to open the "DRC Submission Form".


PVS 16.12-64b DRC Run Submission Form
File Preferences Tools Toolbars Help
cādence
Run Data
Rules
Input
Output
DRC Options
Cancel
Apply
Submit

Run Data

Run Directory PVS_RUN/DRC

Processing

Run PVS Single CPU

Run PVS on Local host

License Timeout, sec

PVSUI Log File PVS_RUN/DRC/pvsuidrc.log

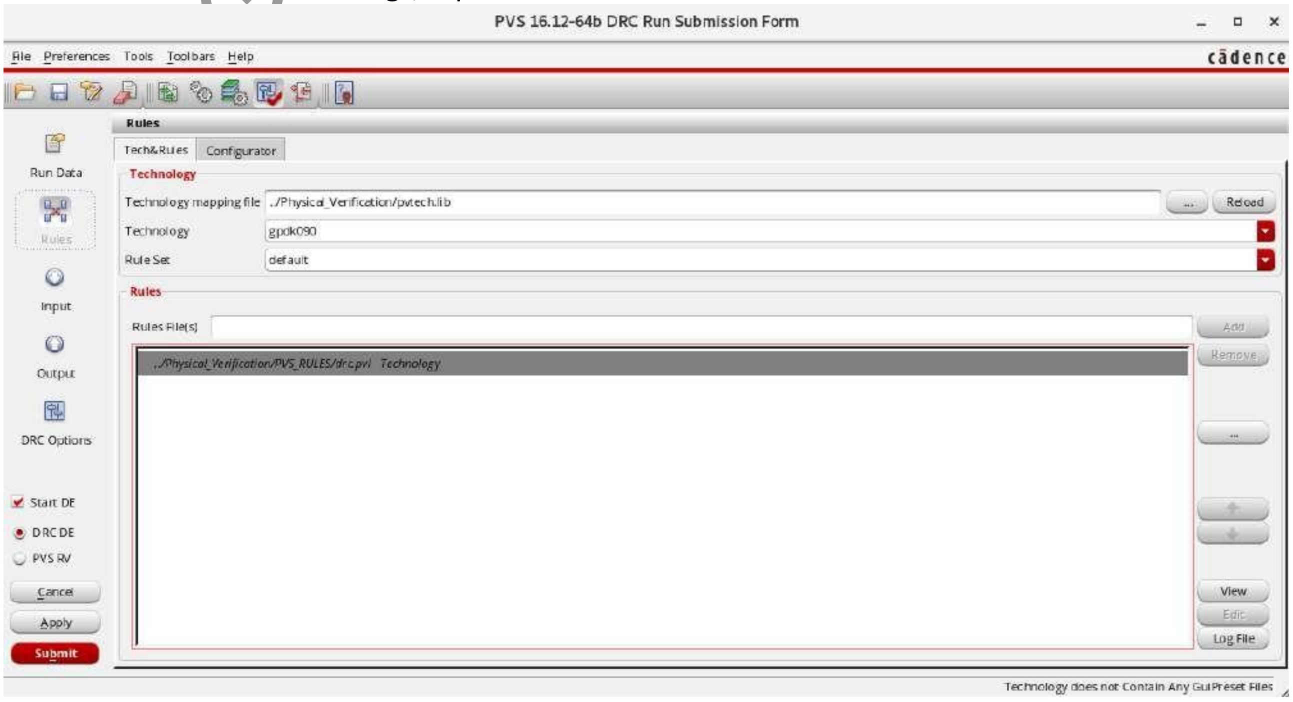
Start DE ✓

DRC DE ●

PVS RV

Technology does not Contain Any GuiPreset Files

The DRC Run Submission Form begins with mentioning the Run Directory. The Run Directory isthe location where all the logs, reports and other files concerned with PVS are saved.



Rules

Technology

Technology mapping file ./Physical_Verification/pvtech.lib

Technology gpdk090

Rule Set default

Rules

Rules File(s)

../Physical_Verification/PVS_RULES/drc.pvl Technology

Start DE ✓

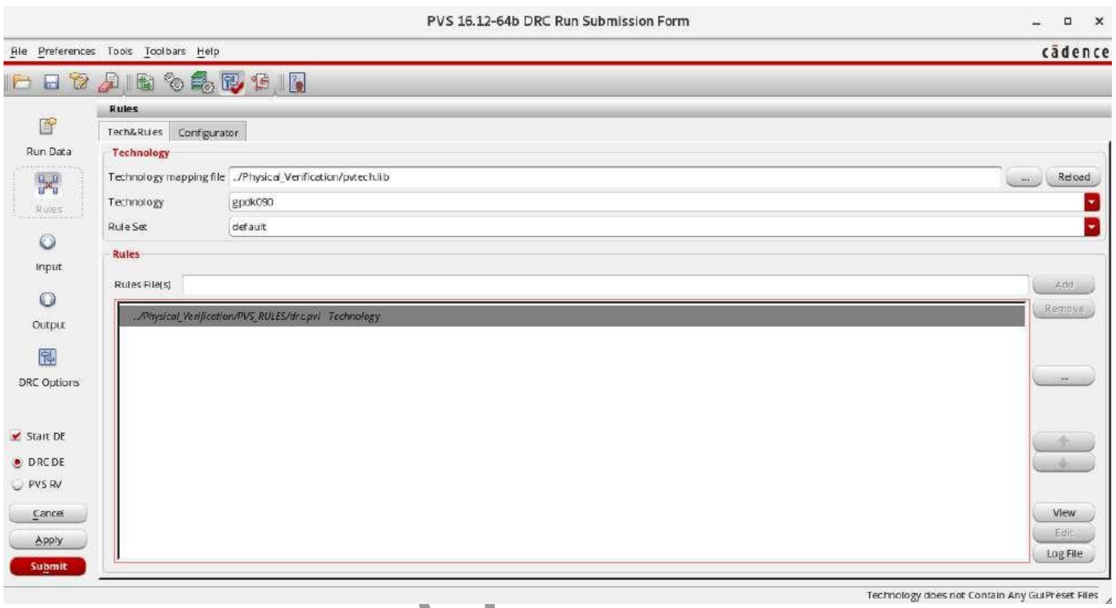
DRC DE ●

PVS RV

Technology does not Contain Any GuiPreset Files

The technology Library is to be loaded under "Rules tab".

- The Technology Library is specific for PVS Tool and technology node on which the designis created.
- On reading the tech lib, the rule set is loaded and the corresponding fabrication rules areread in to be checked against the design.



Rules

Technology

Technology mapping file ./Physical_Verification/pvtech.lib

Technology gpdk090

Rule Set default

Rules

Rules File(s)

../Physical_Verification/PVS_RULES/drc.pvl Technology

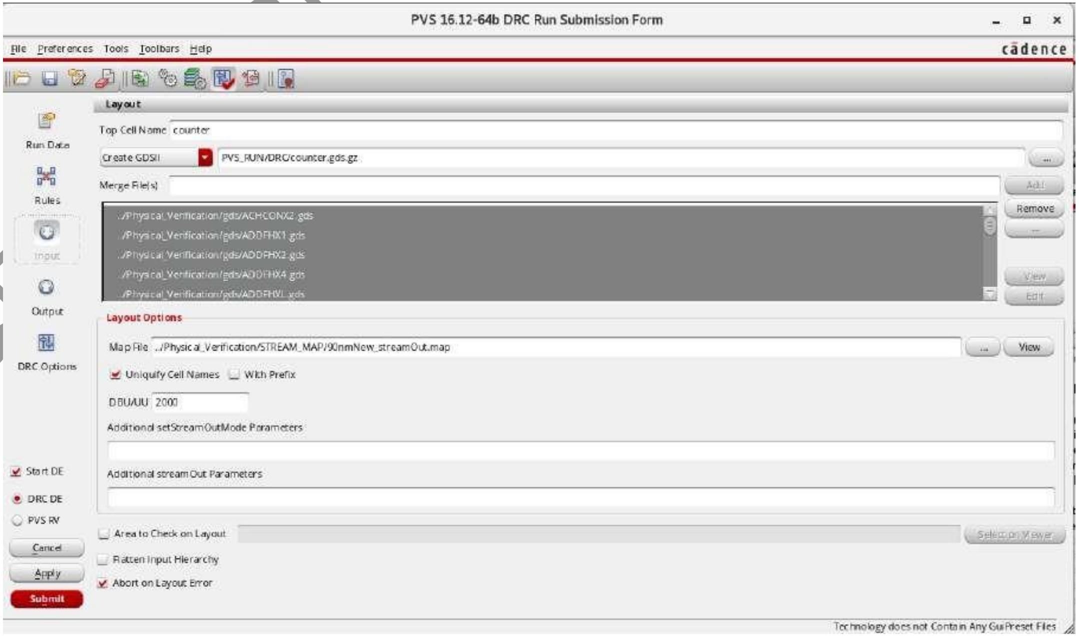
[✓] Start DE

(●) DRC DE

( ) PVS RV

Technology does not Contain Any GuiPreset Files

- The GDS format files of all standard cells available with the corresponding technology nodeare also provided by the vendor. Select all of them to add.



Layout

Top Cell Name counter

Create GDSII PVS_RUN/DRC/counter.gds.gz

Merge File(s)

./Physical_Verification/gds/ACHCONX2.gds
./Physical_Verification/gds/ADDFHX1.gds
./Physical_Verification/gds/ADDFHX2.gds
./Physical_Verification/gds/ADDFHX4.gds
./Physical_Verification/gds/ADDFHXL.gds

Layout Options

Map File ../Physical_Verification/STREAM_MAP/90nmNew_streamOut.map

[✓] Uniquify Cell Names [ ] With Prefix

DBU/UU 2000

Additional setStreamOutMode Parameters

Additional streamOut Parameters

[ ] Area to Check on Layout

[ ] Flatten Input Hierarchy

[✓] Abort on Layout Error

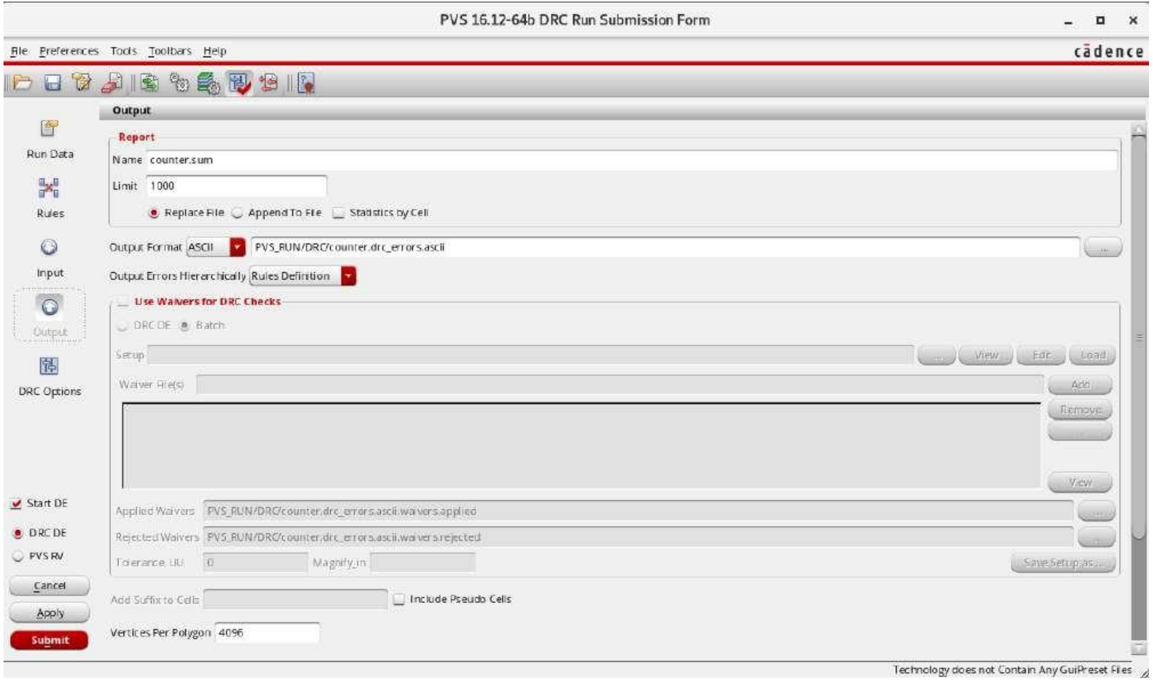
[✓] Start DE

(●) DRC DE

( ) PVS RV

Technology does not Contain Any GuiPreset Files

- The output report can be named and saved as shown.



Output

Report

Name counter.sum

Limit 1000

(●) Replace File ( ) Append To File [ ] Statistics by Cell

Output Format ASCII PVS_RUN/DRC/counter.drc_errors.ascii

Output Errors Hierarchically Rules Definition

[ ] Use Waivers for DRC Checks

( ) DRC DE (●) Batch

Setup

Waiver File(s)

Applied Waivers PVS_RUN/DRC/counter.drc_errors.ascii.waivers.applied

Rejected Waivers PVS_RUN/DRC/counter.drc_errors.ascii.waivers.rejected

Tolerance UU 0 Magnify_in

Add Suffix to Cells [ ] Include Pseudo Cells

Vertices Per Polygon 4096

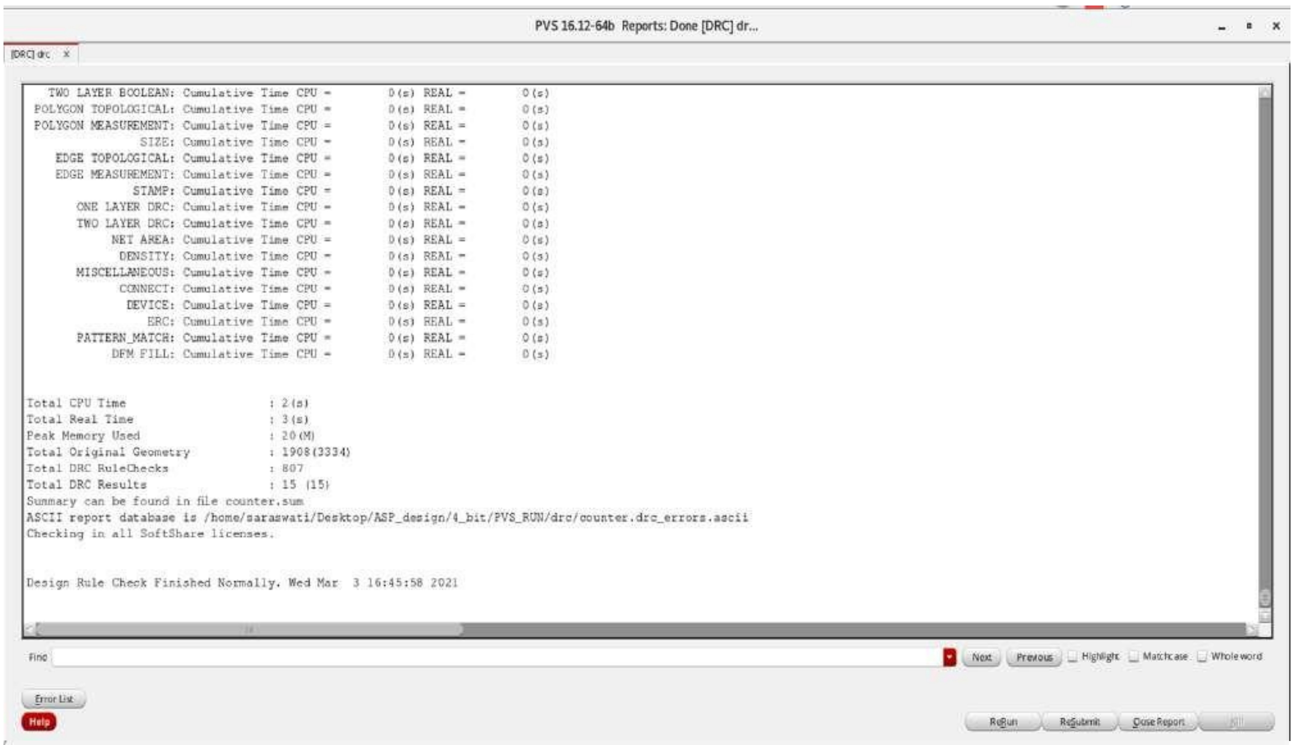
[✓] Start DE

(●) DRC DE

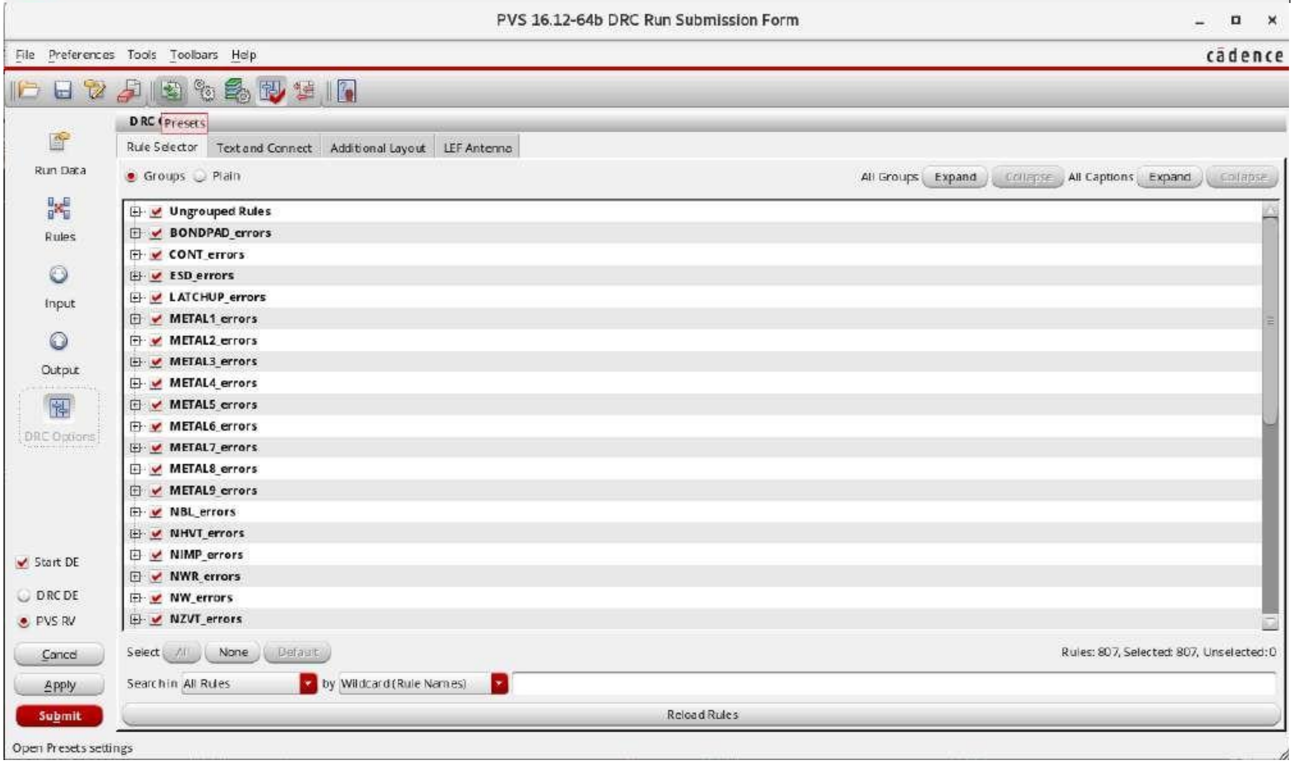
( ) PVS RV

Technology does not Contain Any GuiPreset Files

- Hit “Submit” to run the DRC and the following windows appear.



```
   TWO LAYER BOOLEAN: Cumulative Time CPU =        0(s) REAL =        0(s)
 POLYGON TOPOLOGICAL: Cumulative Time CPU =        0(s) REAL =        0(s)
 POLYGON MEASUREMENT: Cumulative Time CPU =        0(s) REAL =        0(s)
                SIZE: Cumulative Time CPU =        0(s) REAL =        0(s)
    EDGE TOPOLOGICAL: Cumulative Time CPU =        0(s) REAL =        0(s)
    EDGE MEASUREMENT: Cumulative Time CPU =        0(s) REAL =        0(s)
               STAMP: Cumulative Time CPU =        0(s) REAL =        0(s)
       ONE LAYER DRC: Cumulative Time CPU =        0(s) REAL =        0(s)
       TWO LAYER DRC: Cumulative Time CPU =        0(s) REAL =        0(s)
            NET AREA: Cumulative Time CPU =        0(s) REAL =        0(s)
             DENSITY: Cumulative Time CPU =        0(s) REAL =        0(s)
       MISCELLANEOUS: Cumulative Time CPU =        0(s) REAL =        0(s)
             CONNECT: Cumulative Time CPU =        0(s) REAL =        0(s)
              DEVICE: Cumulative Time CPU =        0(s) REAL =        0(s)
                 ERC: Cumulative Time CPU =        0(s) REAL =        0(s)
       PATTERN_MATCH: Cumulative Time CPU =        0(s) REAL =        0(s)
            DFM FILL: Cumulative Time CPU =        0(s) REAL =        0(s)


Total CPU Time                  : 2(s)
Total Real Time                 : 3(s)
Peak Memory Used                : 20(M)
Total Original Geometry         : 1908(3334)
Total DRC RuleChecks            : 807
Total DRC Results               : 15 (15)
Summary can be found in file counter.sum
ASCII report database is /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/drc/counter.drc_errors.ascii
Checking in all SoftShare licenses.


Design Rule Check Finished Normally. Wed Mar  3 16:45:58 2021
```



● Groups ○ Plain

- ✓ Ungrouped Rules
- ✓ BONDPAD_errors
- ✓ CONT_errors
- ✓ ESD_errors
- ✓ LATCHUP_errors
- ✓ METAL1_errors
- ✓ METAL2_errors
- ✓ METAL3_errors
- ✓ METAL4_errors
- ✓ METAL5_errors
- ✓ METAL6_errors
- ✓ METAL7_errors
- ✓ METAL8_errors
- ✓ METAL9_errors
- ✓ NBL_errors
- ✓ NHVT_errors
- ✓ NIMP_errors
- ✓ NWR_errors
- ✓ NW_errors
- ✓ NZVT_errors

✓ Start DE

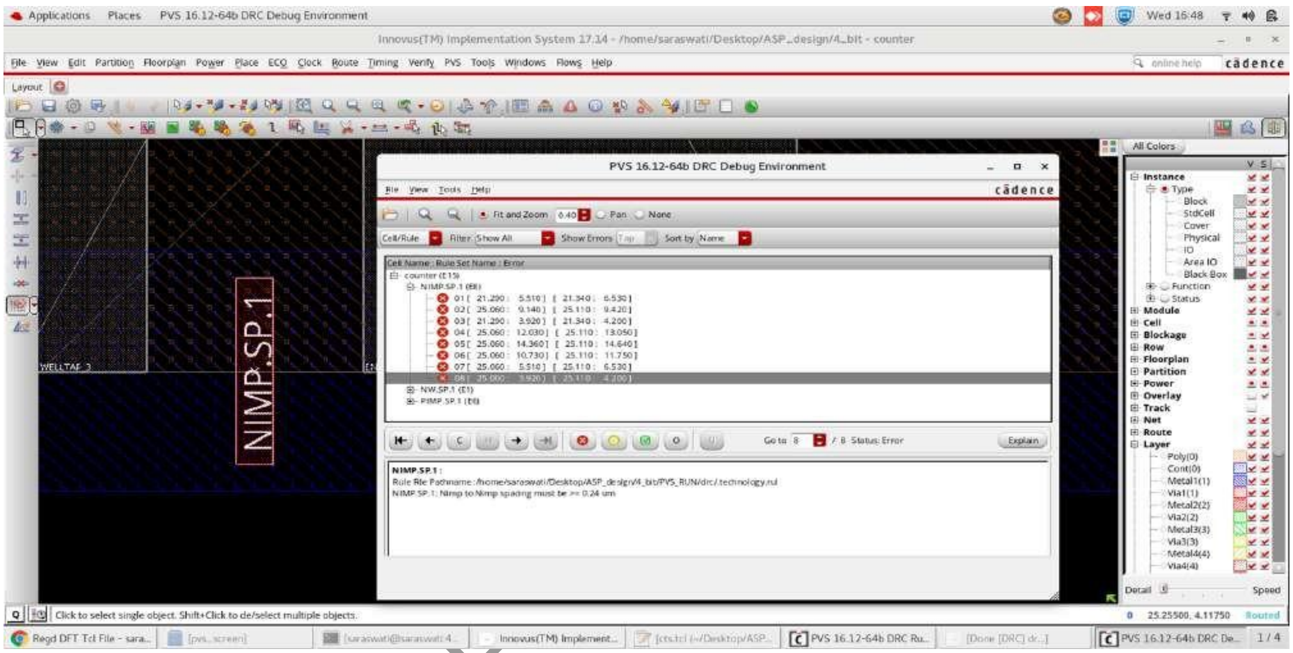
○ DRC DE

● PVS RV

Rules: 807, Selected: 807, Unselected: 0

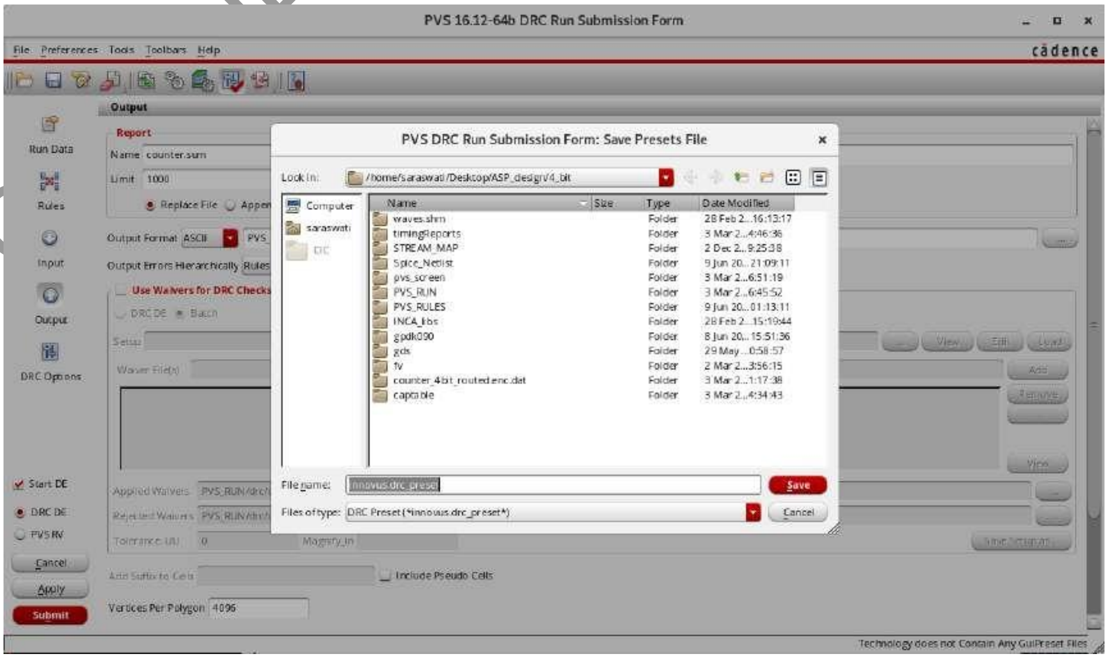
Search in All Rules by Wildcard(Rule Names)

- All the list of DRC Errors can be seen in the above window of which the location of the DRC Violation occuring can be highlighted dealing one to one.



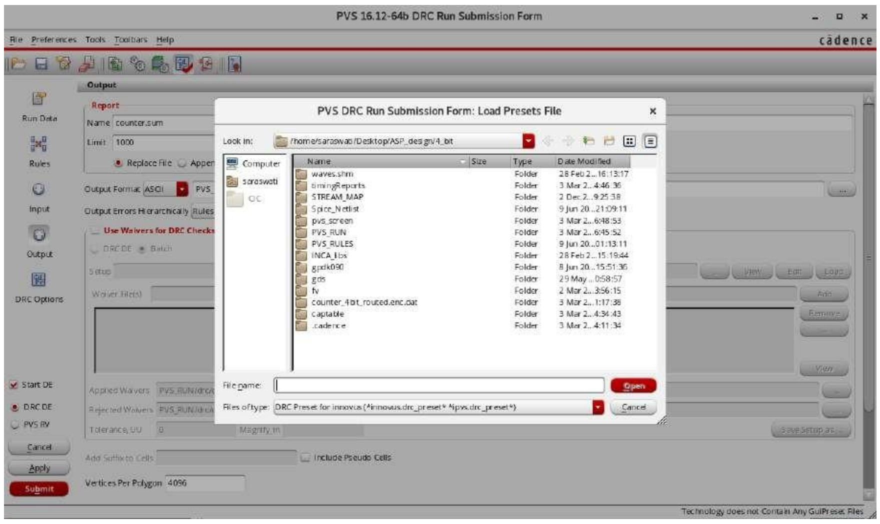
```
Cell Name : Rule Set Name : Error
counter (E15)
  NIMP.SP.1 (E8)
    01 [ 21.290 : 5.510 ] [ 21.340 : 6.530 ]
    02 [ 25.060 : 9.140 ] [ 25.110 : 9.420 ]
    03 [ 21.290 : 3.920 ] [ 21.340 : 4.200 ]
    04 [ 25.060 : 12.030 ] [ 25.110 : 13.050 ]
    05 [ 25.060 : 14.360 ] [ 25.110 : 14.640 ]
    06 [ 25.060 : 10.730 ] [ 25.110 : 11.750 ]
    07 [ 25.060 : 5.510 ] [ 25.110 : 6.530 ]
    08 [ 25.060 : 3.920 ] [ 25.110 : 4.200 ]
  NW.SP.1 (E1)
  PIMP.SP.1 (E6)

Go to 8 / 8 Status: Error

NIMP.SP.1 :
Rule File Pathname : /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/drc/.technology.rul
NIMP.SP.1: Nimp to Nimp spacing must be >= 0.24 um
```

- For example, in the above shown snapshot, the errors associated with N-Implant can be seen. (Select a error occurrence and click on the right arrow below to highlight/zoom in thelocation.)
- You can save the DRC Run as a "Preset" file to rerun the DRC if required at a later point of time.
- Saving/loading the Preset File is shown below.



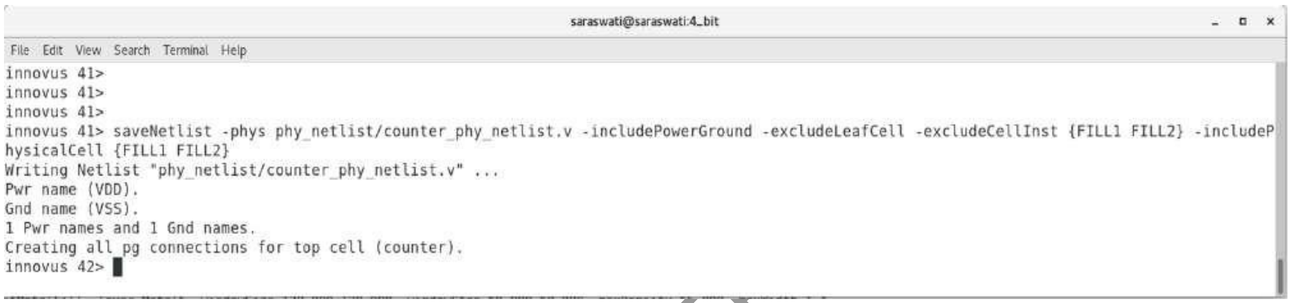
```
PVS DRC Run Submission Form: Save Presets File
Look In: /home/saraswati/Desktop/ASP_design/4_bit
File name: innovus.drc_preset
Files of type: DRC Preset (*innovus.drc_preset*)
```

- Loading a Preset file is shown below.

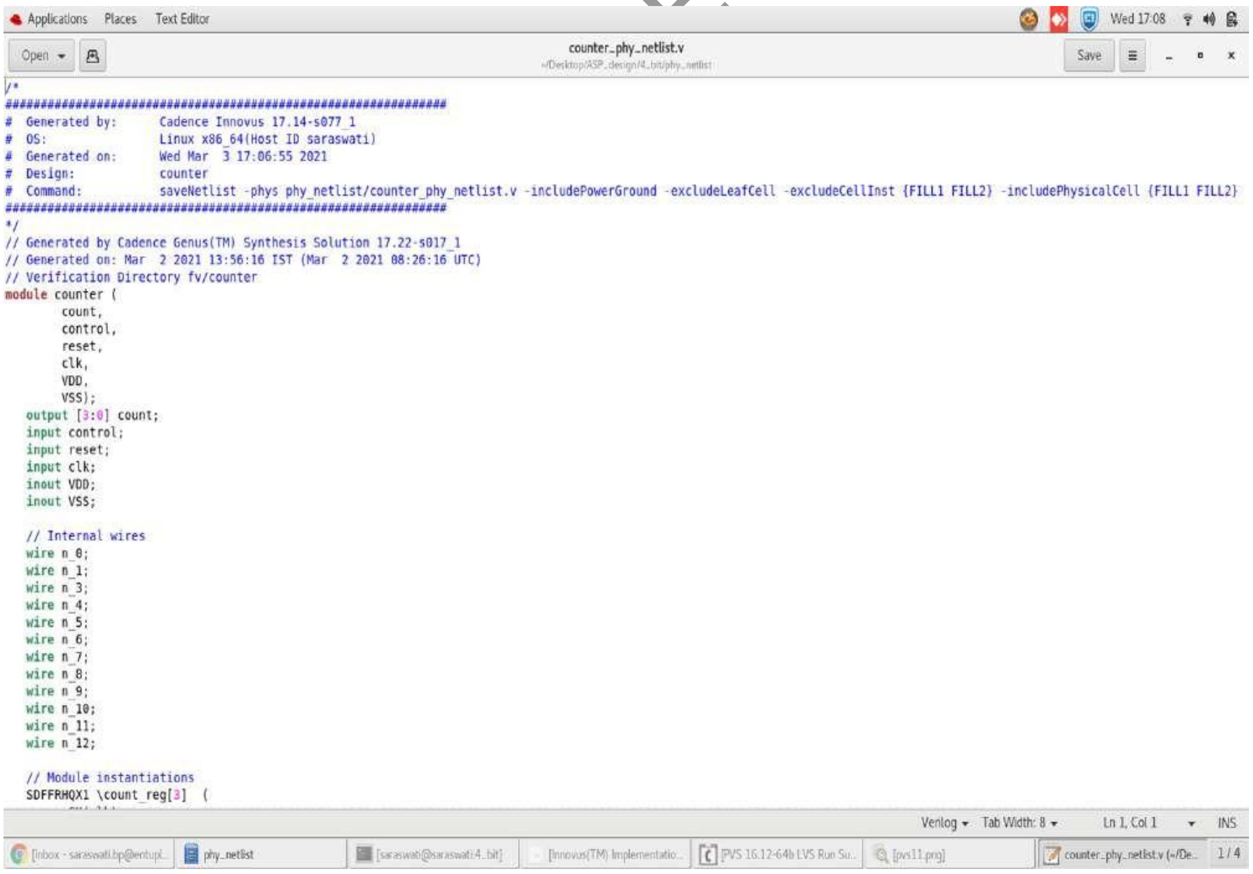


```
PVS DRC Run Submission Form: Load Presets File
Look In: /home/saraswati/Desktop/ASP_design/4_bit
Files of type: DRC Preset for innovus (*innovus.drc_preset* *pvs.drc_preset*)
```

Note : A Physical Netlist can be saved after the DRC Run as shown below.

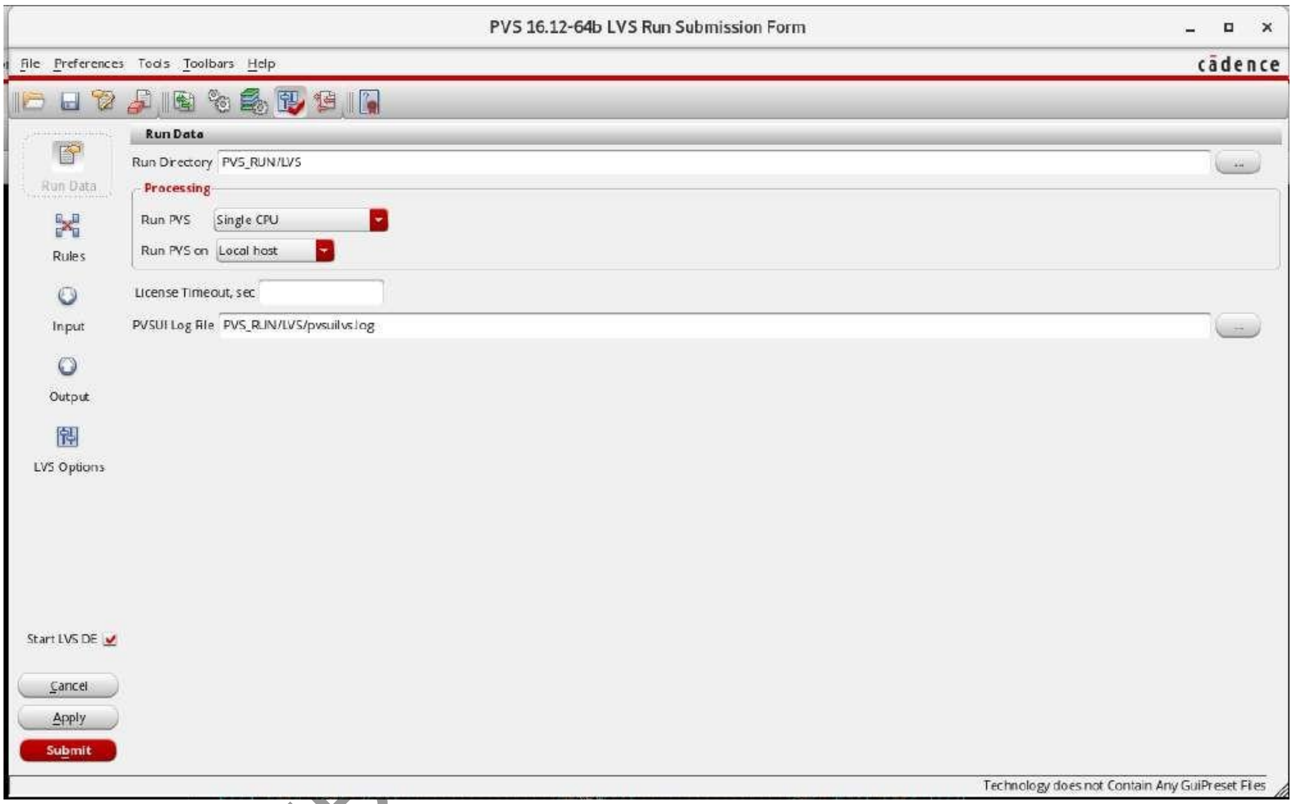


```
innovus 41>
innovus 41>
innovus 41>
innovus 41> saveNetlist -phys phy_netlist/counter_phy_netlist.v -includePowerGround -excludeLeafCell -excludeCellInst {FILL1 FILL2} -includeP
hysicalCell {FILL1 FILL2}
Writing Netlist "phy_netlist/counter_phy_netlist.v" ...
Pwr name (VDD).
Gnd name (VSS).
1 Pwr names and 1 Gnd names.
Creating all pg connections for top cell (counter).
innovus 42> █
```



```
/*
###################################################################
#  Generated by:      Cadence Innovus 17.14-s077_1
#  OS:                Linux x86_64(Host ID saraswati)
#  Generated on:      Wed Mar  3 17:06:55 2021
#  Design:            counter
#  Command:           saveNetlist -phys phy_netlist/counter_phy_netlist.v -includePowerGround -excludeLeafCell -excludeCellInst {FILL1 FILL2} -includePhysicalCell {FILL1 FILL2}
###################################################################
*/
// Generated by Cadence Genus(TM) Synthesis Solution 17.22-s017_1
// Generated on: Mar  2 2021 13:56:16 IST (Mar  2 2021 08:26:16 UTC)
// Verification Directory fv/counter
module counter (
        count,
        control,
        reset,
        clk,
        VDD,
        VSS);
   output [3:0] count;
   input control;
   input reset;
   input clk;
   inout VDD;
   inout VSS;

   // Internal wires
   wire n_0;
   wire n_1;
   wire n_3;
   wire n_4;
   wire n_5;
   wire n_6;
   wire n_7;
   wire n_8;
   wire n_9;
   wire n_10;
   wire n_11;
   wire n_12;

   // Module instantiations
   SDFFRHQX1 \count_reg[3]  (
```

- Inputs Required – LVS :
  - Technology Library
  - Standard Cell GDS Files
  - Spice Netlist of all Standard Cells (Provided by Library Vendor)
- Outputs – LVS :
  - LVS Match/Mismatch Report
- From the Innovus GUI, Select PVS → Run LVS to open the LVS run submission form.


PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Run Data
Rules
Input
Output
LVS Options
Start LVS DE
Cancel
Apply
Submit

Run Data

Run Directory PVS_RUN/LVS

Processing

Run PVS Single CPU

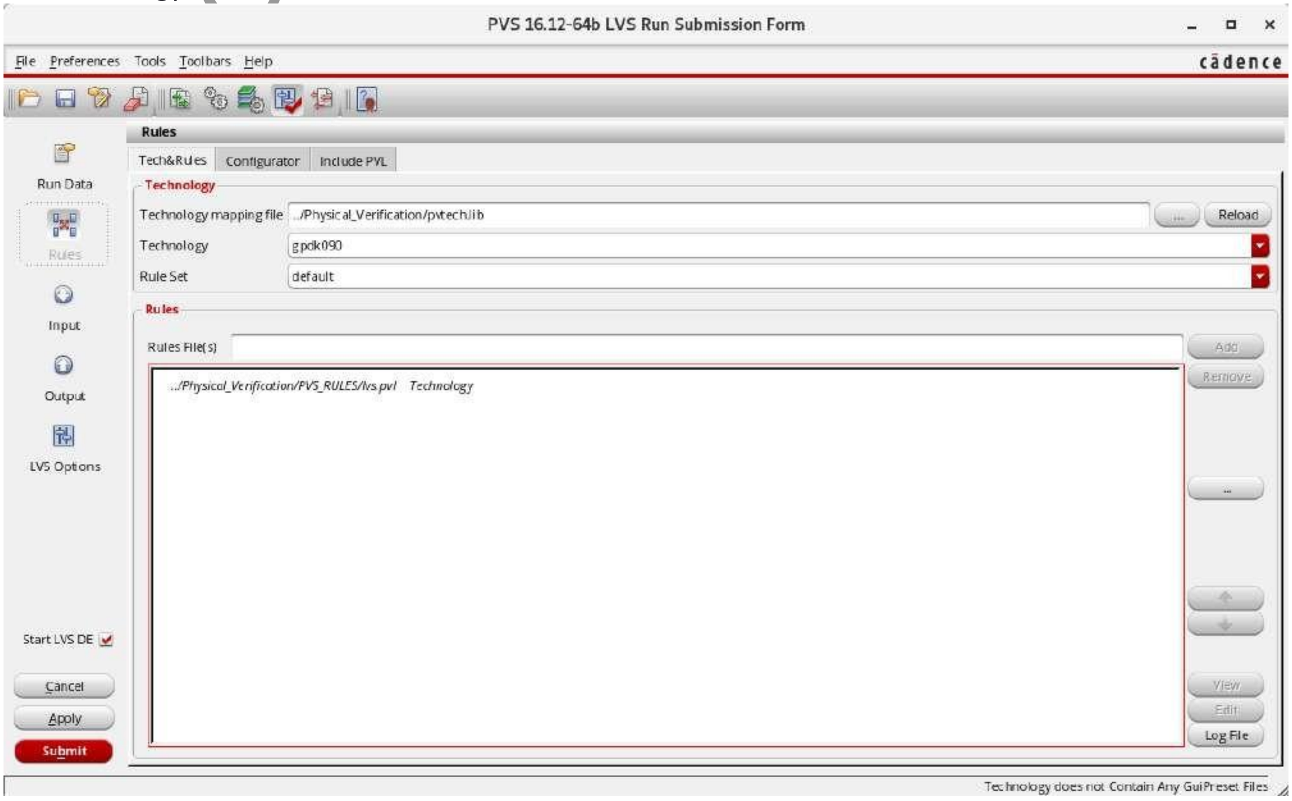
Run PVS on Local host

License Timeout, sec

PVSUI Log File PVS_RUN/LVS/pvsuilvs.log

Technology does not Contain Any GuiPreset Files

- Provide the Run directory and log file name (Along with path – Optional)
- Load the Tech Lib, GDS Files and Spice Netlist of all Standard Cells under the corresponding technology node.



Rules

Technology

Technology mapping file ../Physical_Verification/pvtech.lib

Technology gpdk090

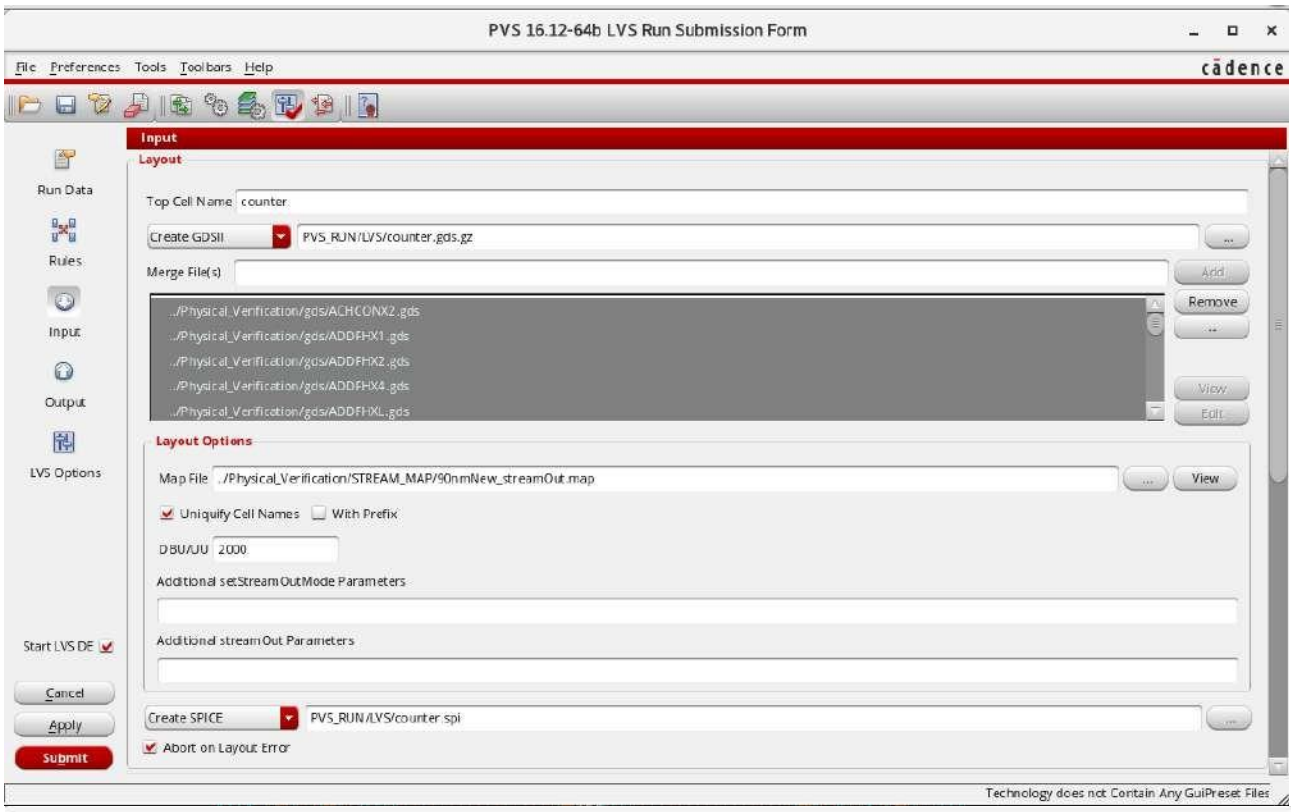
Rule Set default

Rules

Rules File(s)

../Physical_Verification/PVS_RULES/lvs.pvl Technology

Technology does not Contain Any GuiPreset Files



PVS 16.12-64b LVS Run Submission Form

File Preferences Tools Toolbars Help

Input

Layout

Top Cell Name: counter

Create GDSII: PVS_RUN/LVS/counter.gds.gz

Merge File(s):

```
../Physical_Verification/gds/ACHCONX2.gds
../Physical_Verification/gds/ADDFHX1.gds
../Physical_Verification/gds/ADDFHX2.gds
../Physical_Verification/gds/ADDFHX4.gds
../Physical_Verification/gds/ADDFHXL.gds
```

Layout Options

Map File: ./Physical_Verification/STREAM_MAP/90nmNew_streamOut.map

[✓] Uniquify Cell Names [ ] With Prefix

DBU/UU: 2000

Additional setStreamOutMode Parameters

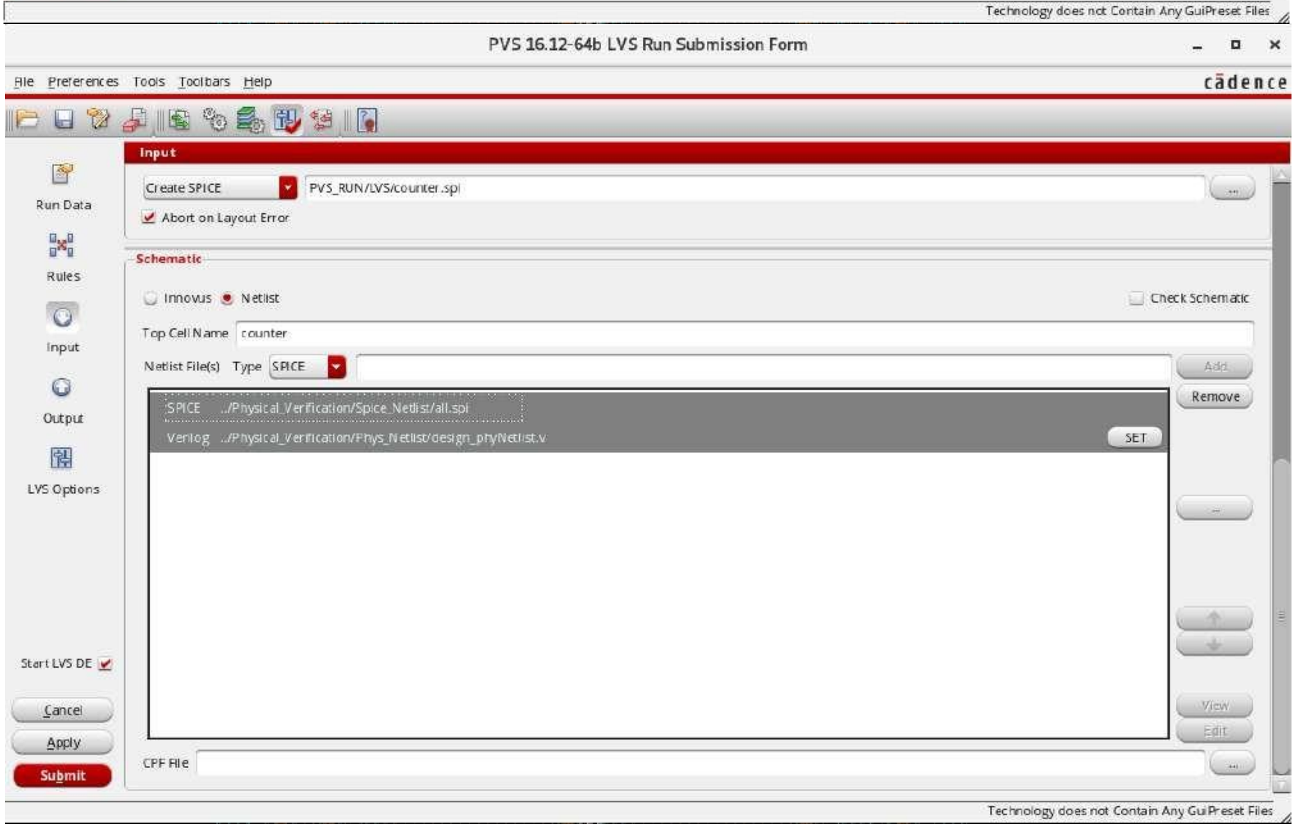
Additional streamOut Parameters

Create SPICE: PVS_RUN/LVS/counter.spi

[✓] Abort on Layout Error

Start LVS DE [✓]

Technology does not Contain Any GuiPreset Files



PVS 16.12-64b LVS Run Submission Form

File Preferences Tools Toolbars Help

Input

Create SPICE: PVS_RUN/LVS/counter.spi

[✓] Abort on Layout Error

Schematic

( ) Innovus (●) Netlist [ ] Check Schematic

Top Cell Name: counter

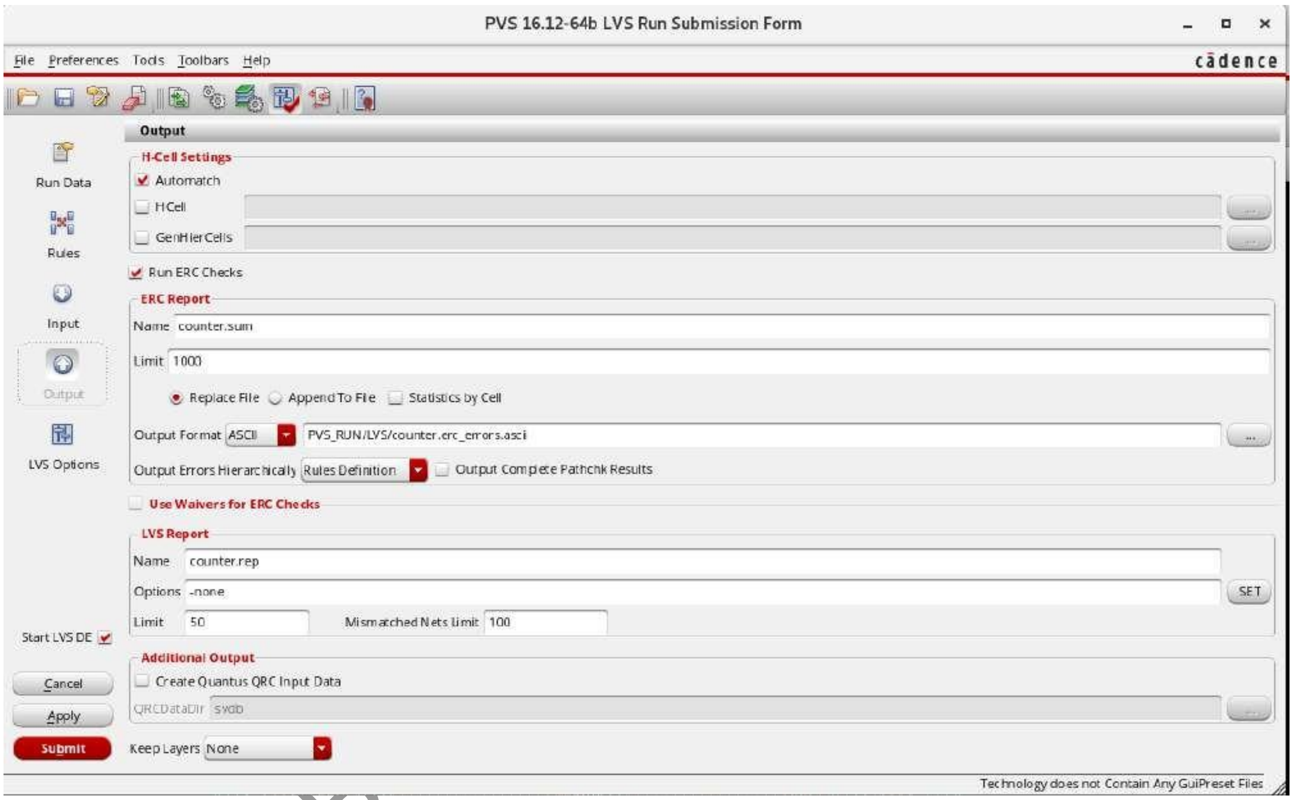
Netlist File(s) Type: SPICE

```
SPICE    ../Physical_Verification/Spice_Netlist/all.spi
Verilog  ../Physical_Verification/Phys_Netlist/design_phyNetlist.v
```

CPF File:

Start LVS DE [✓]

Technology does not Contain Any GuiPreset Files


PVS 16.12-64b LVS Run Submission Form
File Preferences Tools Toolbars Help
cadence
Run Data
Rules
Input
Output
LVS Options
Cancel
Apply
Submit

Output

H-Cell Settings
- [✓] Automatch
- [ ] HCell
- [ ] GenHierCells

[✓] Run ERC Checks

ERC Report
Name counter.sum
Limit 1000
[✓ Replace File] Append To File [ ] Statistics by Cell
Output Format ASCII PVS_RUN/LVS/counter.erc_errors.ascii
Output Errors Hierarchically Rules Definition [ ] Output Complete Pathchk Results

[ ] Use Waivers for ERC Checks

LVS Report
Name counter.rep
Options -none SET
Limit 50 Mismatched Nets Limit 100

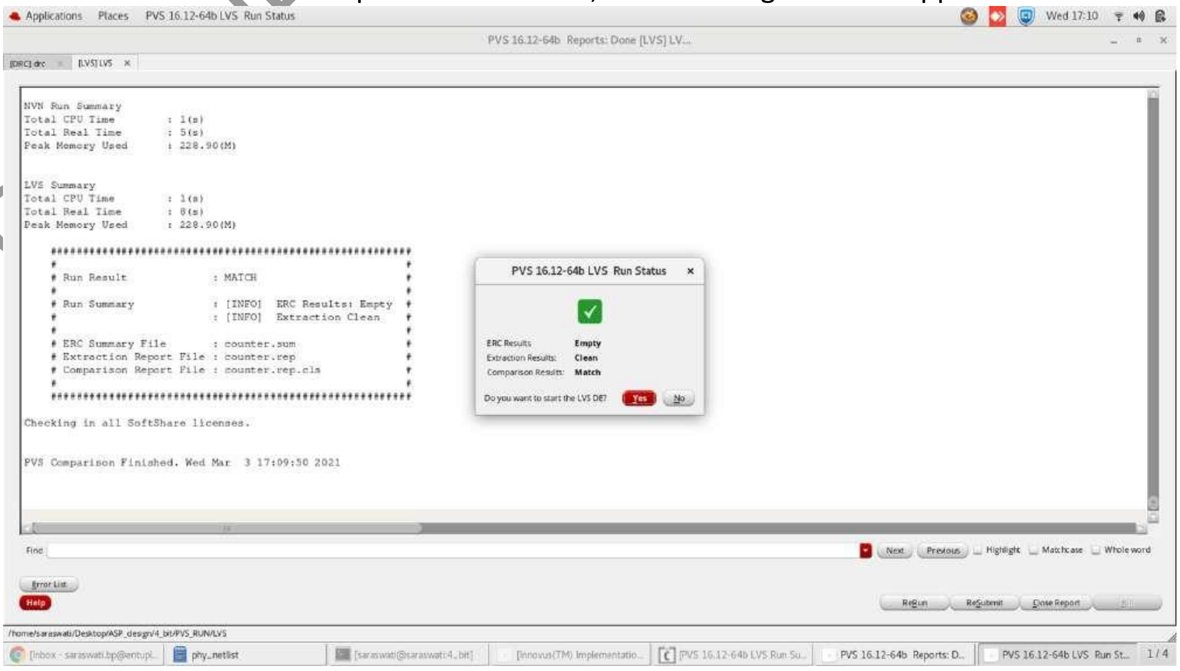
Start LVS DE [✓]

Additional Output
[ ] Create Quantus QRC Input Data
QRCDataDir svdb

Keep Layers None

Technology does not Contain Any GuiPreset Files

- On successful completion of LVS Run, the following windows appear.



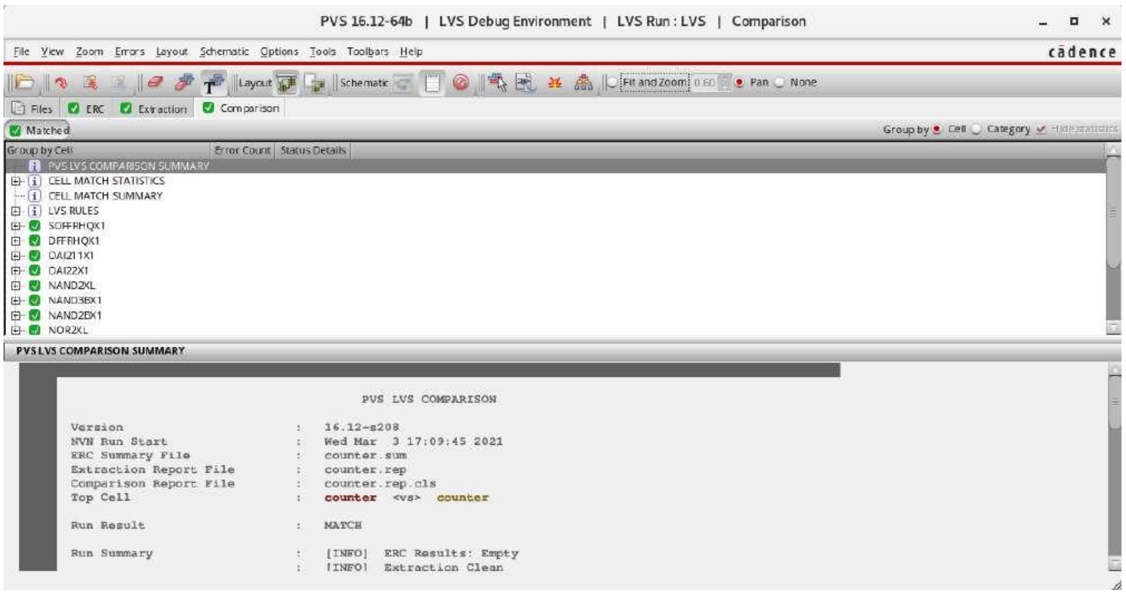
```
NVN Run Summary
Total CPU Time         : 1(s)
Total Real Time        : 5(s)
Peak Memory Used       : 228.90(M)


LVS Summary
Total CPU Time         : 1(s)
Total Real Time        : 8(s)
Peak Memory Used       : 228.90(M)

    ##########################################################
    #                                                        #
    # Run Result             : MATCH                         #
    #                                                        #
    # Run Summary            : [INFO]  ERC Results: Empty    #
    #                        : [INFO]  Extraction Clean      #
    #                                                        #
    # ERC Summary File       : counter.sum                   #
    # Extraction Report File : counter.rep                   #
    # Comparison Report File : counter.rep.cls               #
    #                                                        #
    ##########################################################
Checking in all SoftShare licenses.


PVS Comparison Finished. Wed Mar  3 17:09:50 2021
```

PVS 16.12-64b LVS Run Status

ERC Results:	Empty
Extraction Results:	Clean
Comparison Results:	Match

Do you want to start the LVS DE? Yes No

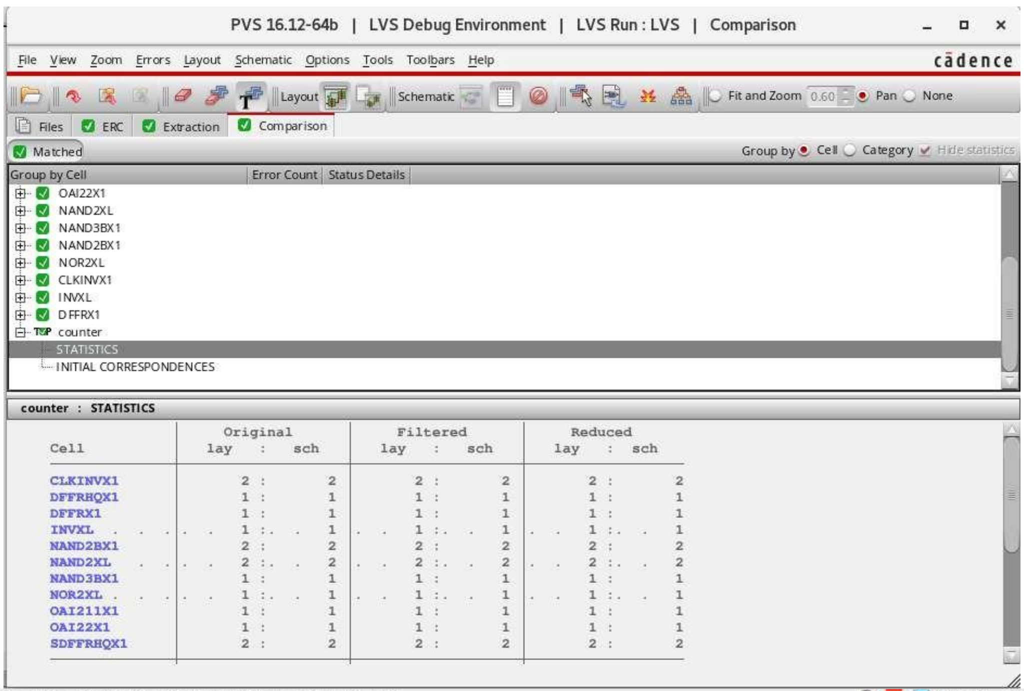
/home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS



PVS LVS COMPARISON SUMMARY
- CELL MATCH STATISTICS
- CELL MATCH SUMMARY
- LVS RULES
- SDFFRHQX1
- DFFRHQX1
- OAI211X1
- OAI22X1
- NAND2XL
- NAND3BX1
- NAND2BX1
- NOR2XL

PVS LVS COMPARISON SUMMARY

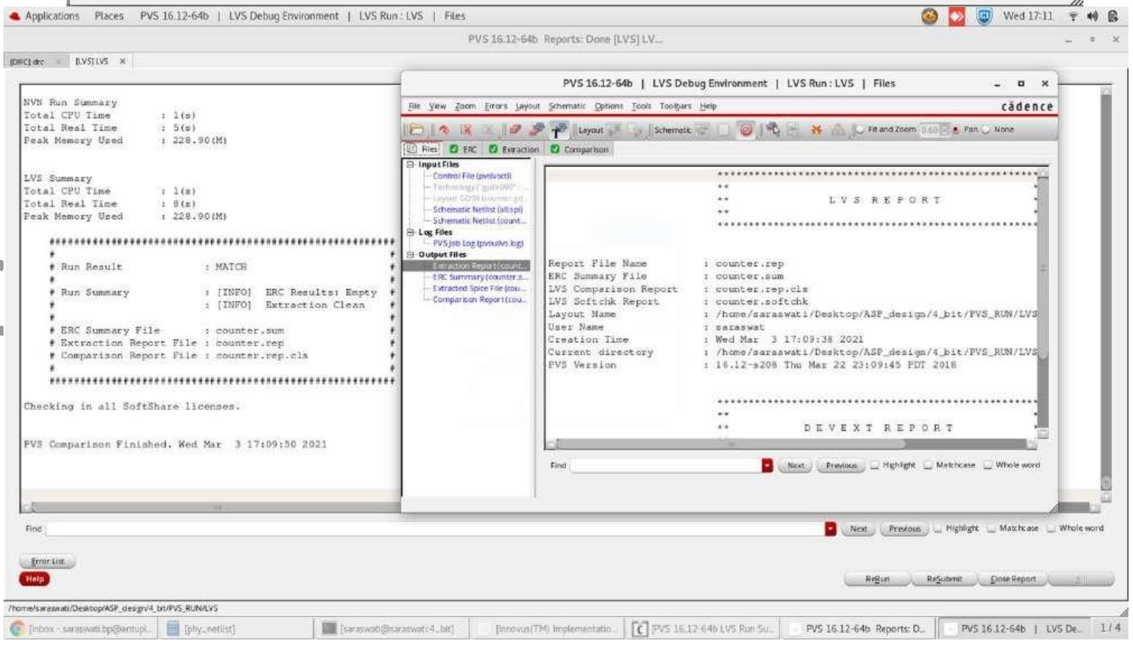
```
                         PVS LVS COMPARISON

Version                           :   16.12-s208
NVN Run Start                     :   Wed Mar  3 17:09:45 2021
ERC Summary File                  :   counter.sum
Extraction Report File            :   counter.rep
Comparison Report File            :   counter.rep.cls
Top Cell                          :   counter  <vs>  counter

Run Result                        :   MATCH

Run Summary                       :   [INFO]  ERC Results: Empty
                                  :   [INFO]  Extraction Clean
```


PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | Comparison
File View Zoom Errors Layout Schematic Options Tools Toolbars Help
cādence
Layout
Schematic
Fit and Zoom 0.60
Pan
None
Files
ERC
Extraction
Comparison
Matched
Group by Cell Category Hide statistics

Group by Cell	Error Count	Status Details
✓ OAI22X1		
✓ NAND2XL		
✓ NAND3BX1		
✓ NAND2BX1		
✓ NOR2XL		
✓ CLKINVX1		
✓ INVXL		
✓ DFFRX1		
TOP counter		
STATISTICS		
INITIAL CORRESPONDENCES		

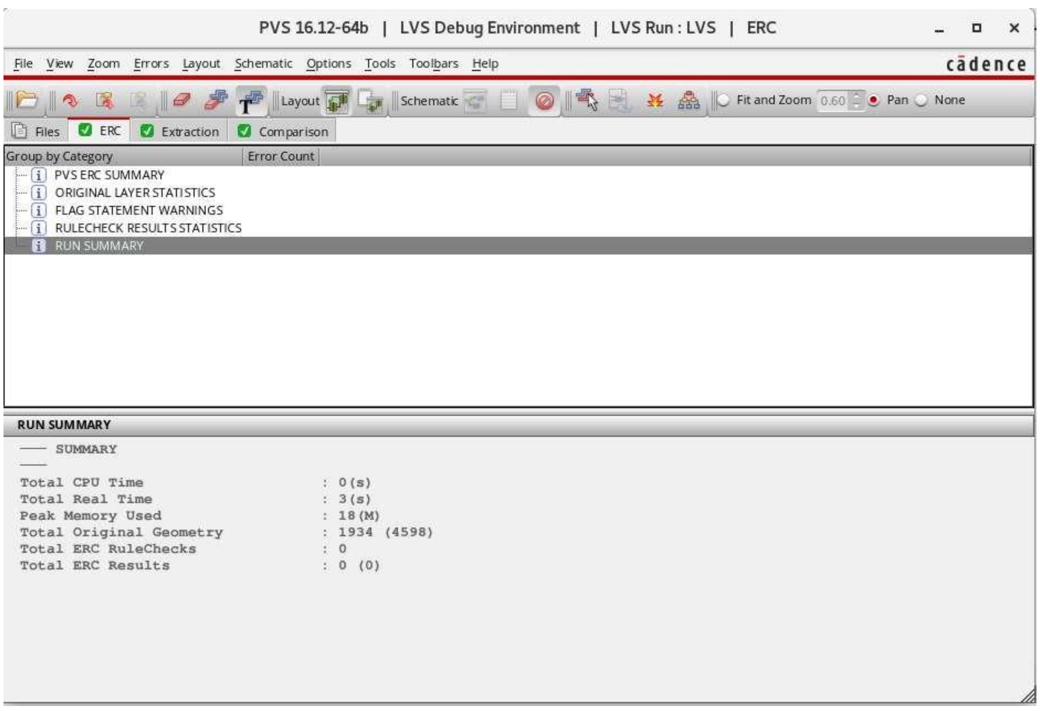
counter : STATISTICS

```
                    Original          Filtered          Reduced
Cell             lay   :   sch     lay   :   sch     lay   :   sch

CLKINVX1           2 :       2       2 :       2       2 :       2
DFFRHQX1           1 :       1       1 :       1       1 :       1
DFFRX1             1 :       1       1 :       1       1 :       1
INVXL  . . . . .   1 :.      1 . .   1 :.      1 . .   1 :. .    1
NAND2BX1           2 :       2       2 :       2       2 :       2
NAND2XL  . . . .   2 :. .    2 . .   2 :. .    2 . .   2 :. .    2
NAND3BX1           1 :       1       1 :       1       1 :       1
NOR2XL . . . . .   1 :.      1 . .   1 :. .    1 . .   1 :. .    1
OAI211X1           1 :       1       1 :       1       1 :       1
OAI22X1            1 :       1       1 :       1       1 :       1
SDFFRHQX1          2 :       2       2 :       2       2 :       2
```


Applications Places PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | Files
Wed 17:11
PVS 16.12-64b Reports: Done [LVS] LV...
[DRC] drc
[LVS] LVS
PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | Files
File View Zoom Errors Layout Schematic Options Tools Toolbars Help
cādence
Files ERC Extraction Comparison
Input Files
Control File (pvslvs.rul)
Technology ("gpdk090")
Layout GDSII (counter.gd...
Schematic Netlist (all.spi)
Schematic Netlist (count...
Log Files
PVS Job Log (pvslvs.log)
Output Files
Extraction Report (count...
ERC Summary (counter.s...
Extracted Spice File (cou...
Comparison Report (cou...
Find
Next
Previous
Highlight
Matchcase
Whole word
Error List
Help
ReRun
ReSubmit
Close Report
/home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS
[Inbox - saraswati.bp@entupl...
[phy_netlist]
[saraswati@saraswati:4_bit]
[Innovus(TM) Implementatio...
[PVS 16.12-64b LVS Run Su...
PVS 16.12-64b Reports: D...
PVS 16.12-64b | LVS De...
1 / 4

```
NVN Run Summary
Total CPU Time         : 1(s)
Total Real Time        : 5(s)
Peak Memory Used       : 228.90(M)


LVS Summary
Total CPU Time         : 1(s)
Total Real Time        : 8(s)
Peak Memory Used       : 228.90(M)

    #######################################################
    #                                                     #
    # Run Result             : MATCH                      #
    #                                                     #
    # Run Summary            : [INFO]  ERC Results: Empty #
    #                        : [INFO]  Extraction Clean   #
    #                                                     #
    # ERC Summary File       : counter.sum                #
    # Extraction Report File : counter.rep                #
    # Comparison Report File : counter.rep.cls            #
    #                                                     #
    #######################################################

Checking in all SoftShare licenses.


PVS Comparison Finished. Wed Mar  3 17:09:50 2021
```

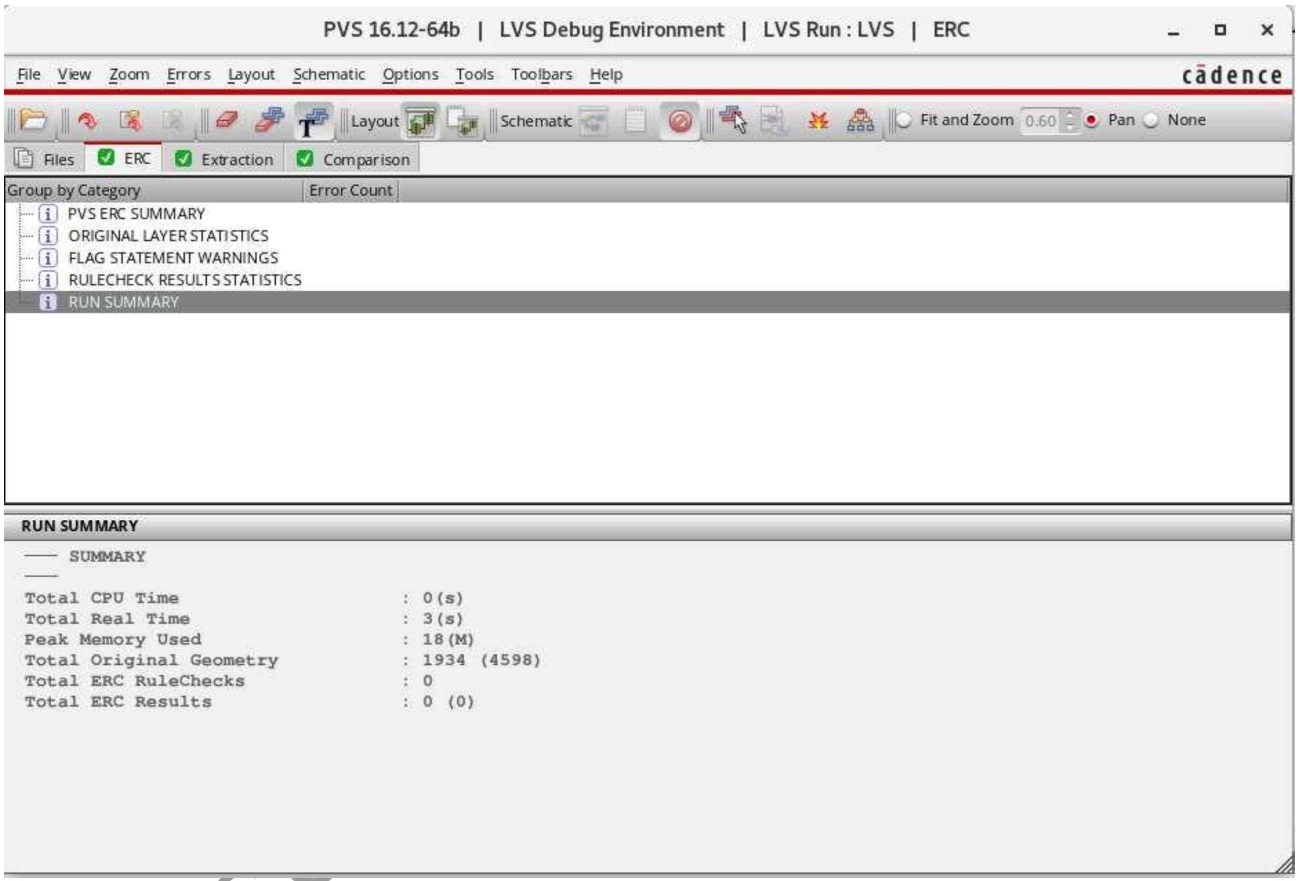
```
                         **************************************************
                         **
                         **               L V S   R E P O R T
                         **
                         **************************************************


Report File Name         : counter.rep
ERC Summary File         : counter.sum
LVS Comparison Report    : counter.rep.cls
LVS Softchk Report       : counter.softchk
Layout Name              : /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS
User Name                : saraswat
Creation Time            : Wed Mar  3 17:09:38 2021
Current directory        : /home/saraswati/Desktop/ASP_design/4_bit/PVS_RUN/LVS
PVS Version              : 16.12-s208 Thu Mar 22 23:09:45 PDT 2018


                         **************************************************
                         **
                         **          D E V E X T   R E P O R T
```


PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | ERC
File View Zoom Errors Layout Schematic Options Tools Toolbars Help
cādence
Layout
Schematic
Fit and Zoom 0.60
Pan
None
Files
ERC
Extraction
Comparison

Group by Category	Error Count
PVS ERC SUMMARY	
ORIGINAL LAYER STATISTICS	
FLAG STATEMENT WARNINGS	
RULECHECK RESULTS STATISTICS	
RUN SUMMARY	

RUN SUMMARY

```
—— SUMMARY
——

Total CPU Time                    : 0(s)
Total Real Time                   : 3(s)
Peak Memory Used                  : 18(M)
Total Original Geometry           : 1934 (4598)
Total ERC RuleChecks              : 0
Total ERC Results                 : 0 (0)
```


PVS 16.12-64b | LVS Debug Environment | LVS Run : LVS | ERC
File View Zoom Errors Layout Schematic Options Tools Toolbars Help
cādence
Layout
Schematic
Fit and Zoom
0.60
Pan
None
Files
ERC
Extraction
Comparison
Group by Category
Error Count
PVS ERC SUMMARY
ORIGINAL LAYER STATISTICS
FLAG STATEMENT WARNINGS
RULECHECK RESULTS STATISTICS
RUN SUMMARY

RUN SUMMARY

```
—— SUMMARY
——

Total CPU Time                    : 0(s)
Total Real Time                   : 3(s)
Peak Memory Used                  : 18(M)
Total Original Geometry           : 1934 (4598)
Total ERC RuleChecks              : 0
Total ERC Results                 : 0 (0)
```

- You can create a GDS file along with Stream out file either using the GUI as File → Save → GDS/Oasis or use the following command.
- **Cmd :** streamOut <GDSFileName>.gds -streamOut <streamOut>.map