Ahsanullah University of Science and Technology

Department of Electrical and Electronic Engineering

LABORATORY MANUAL

FOR

ELECTRICAL AND ELECTRONIC SESSIONAL COURSES

Student Name :

Student ID :

Course No. : **EEE 4134**

Course Title : **VLSI-I Lab**

For the students of

Department of Electrical and Electronic Engineering

4th Year, 1st Semester

Table of Contents

Lab-1: Combinational Logic Circuit Design in Verilog HDL

Introduction

A hardware description language (HDL) is similar to a typical computer programming language except that an HDL is used to describe hardware rather than a program to be executed on a computer. Two HDLs are IEEE standards: Verilog HDL and VHDL (Very High Speed Integrated Circuit Hardware Description Language). Verilog language can be used in several ways to model digital systems. In this lab, we will implement combinational logic circuits using **Gate level or structural modeling, Data flow modeling (DFM),** and **Behavioral modeling** styles of Verilog.

```
module module_name [(port_name{, port_name})];
    [parameter declarations]
    [input declarations]
    [output declarations]
    [inout declarations]
    [wire or tri declarations]
    [reg or integer declarations]
    [function or task declarations]
    [assign continuous assignments]
    [initial block]
    [always blocks]
    [gate instantiations]
    [module instantiations]
endmodule
```

Structure of a Verilog Module

Example 01

Example 01 demonstrates the Verilog HDL code of a Half Adder using the **Gate Level** abstraction.

```
module Half_Adder(s,c,x,y);
input  x,y;
output s,c;
xor s1(s,x,y);
and c1(c,x,y);
endmodule
```

Example 02

Example 02 demonstrates the Verilog HDL code of a Half Adder using the **Data Flow** modeling style.

```
module Half_Adder(s,c,x,y);
input  x,y;
output s,c;
assign s=x^y;
assign c=x&y;
endmodule
```

Example 03

Example 03 demonstrates the Verilog HDL code of a 2X1 multiplexer using the **Gate Level** abstraction.

```
module mux_2to1(s,Io,I1,Y);
input s,Io,I1;
output Y;
wire w1,w2,w3;
not g1(w1,s);
and g2(w2,Io,w1);
and g3(w3,s,I1);
or g4(Y,w2,w3);
endmodule
```

Example 04

Example 04 demonstrates the Verilog HDL code of a 2X1 multiplexer using the **Data Flow** modeling style.

```
module mux_2to1(s,Io,I1,Y);
input s,Io,I1;
output Y;
assign Y=(~s& Io)|(s&I1);
endmodule
```

Example 05

Example 02 demonstrates the Verilog HDL code of a 2X1 multiplexer using the **Behavioral** modeling style.

```
module mux_2to1(s,Io,I1,Y);
input s,Io,I1;
output reg Y;
always@ (s,Io,I1)  //if we use always @* The * operator will automatically identify all sensitive variables.
begin
        if(s==0)
                Y=Io;
        else if(s==1)
                Y=I1;
        else
                Y=0;
end
endmodule
```

Student Task

1. Verify the functionality of **Example-3** using the clock pulses shown in *figure-a*.

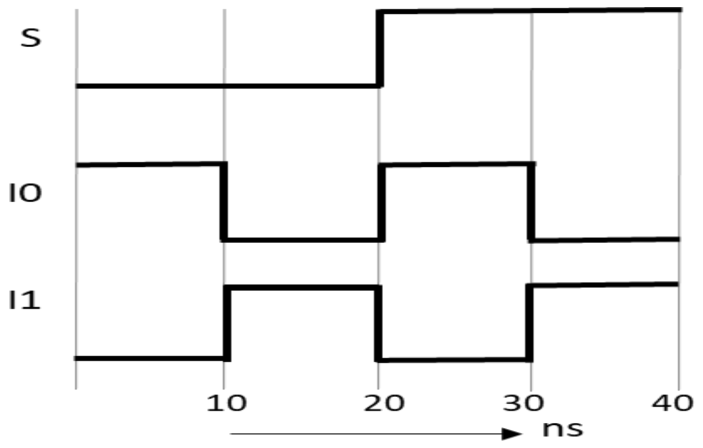


figure-a

2. Write a Verilog code to obtain the same logic circuit of *figure-b* in **'RTL Viewer'** and verify the functionality of the circuit using the clock pulses shown in *figure-c*.

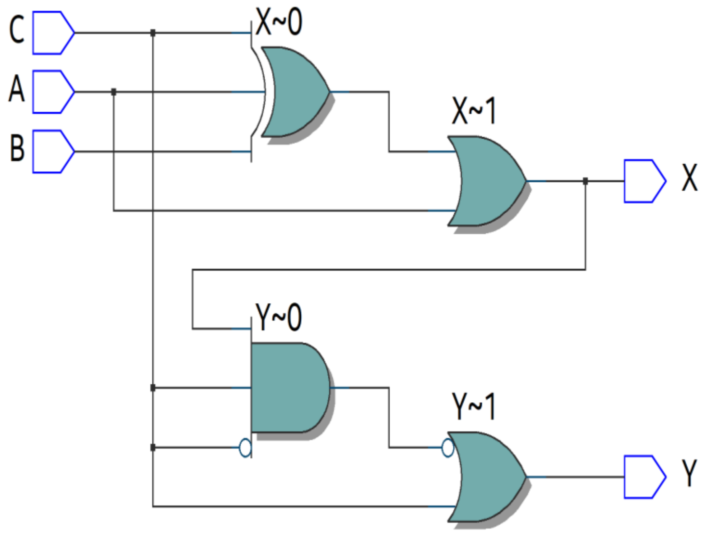


figure-b

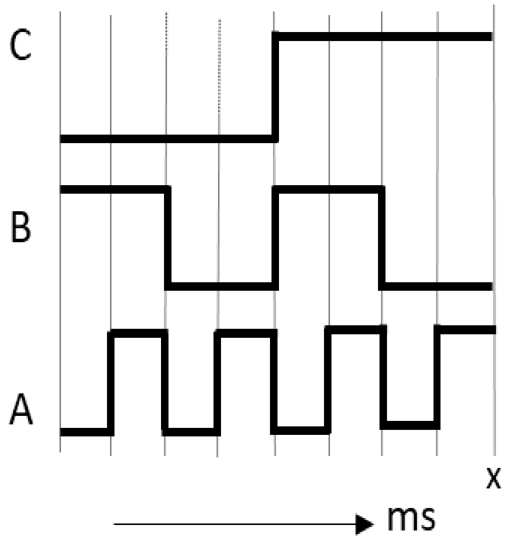
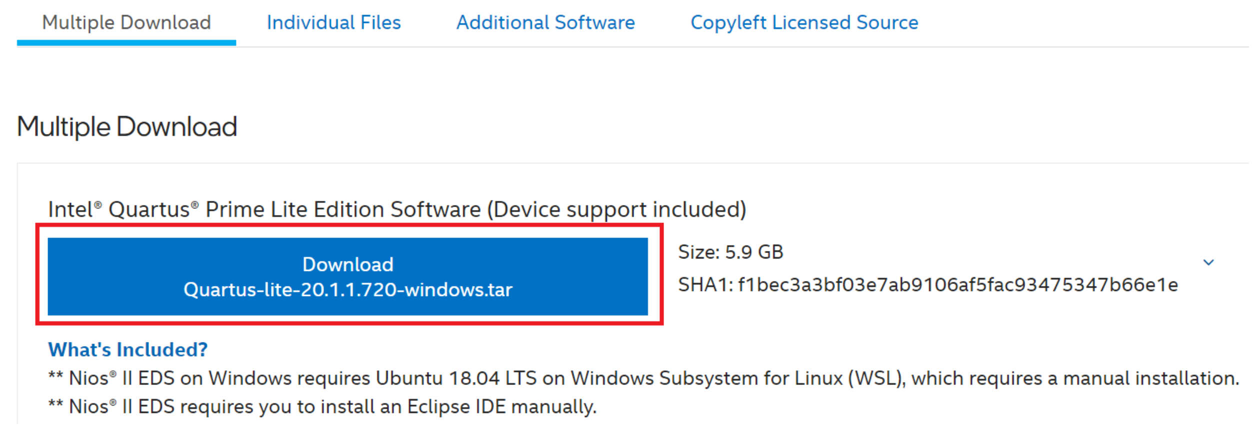


figure-c

Installation of Quartus Prime

1. Download the Intel® Quartus® Prime Lite Edition Design Software Version 22.1.2 from the following link

https://www.intel.com/content/www/us/en/software-kit/660907/intel-quartus-prime-lite-edition-design-software-version-20-1-1-for-windows.html



Multiple Download

Intel® Quartus® Prime Lite Edition Software (Device support included)

Size: 5.9 GB

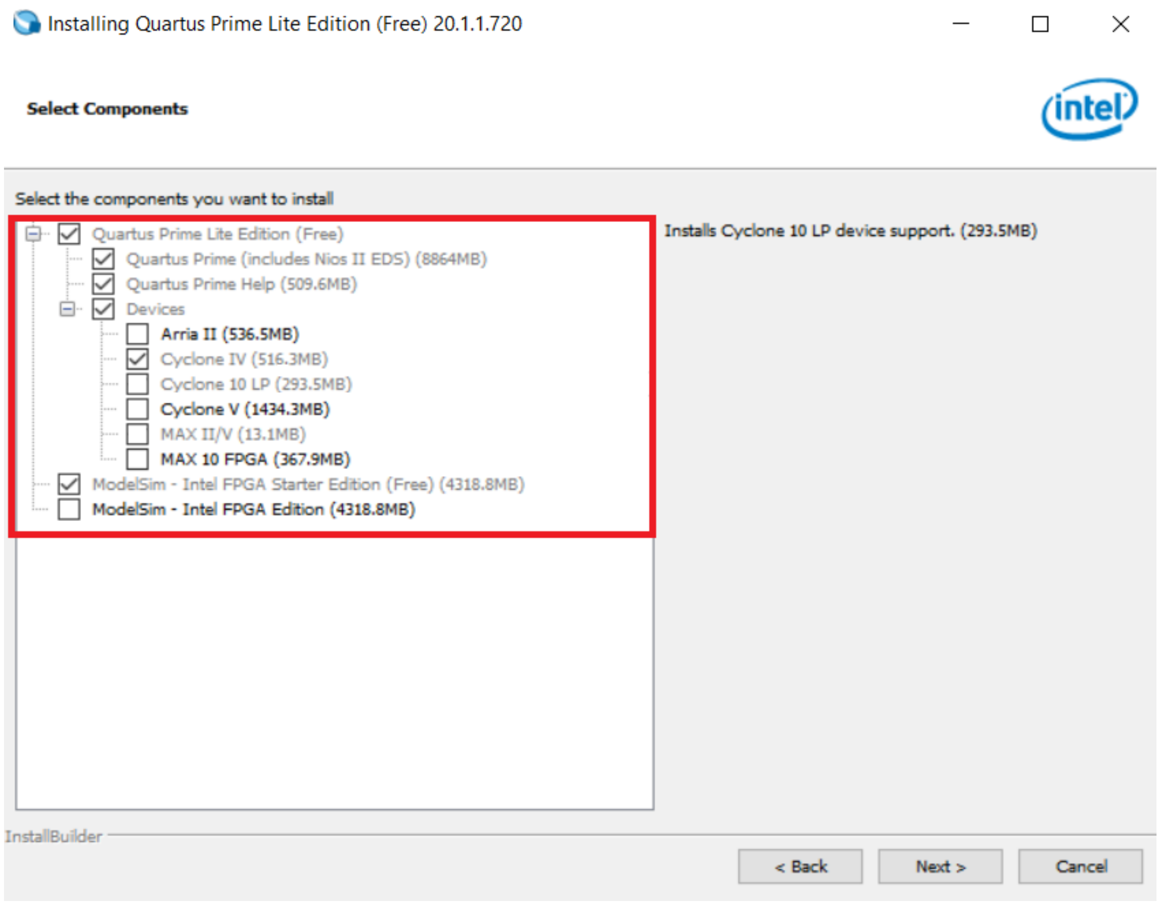
SHA1: f1bec3a3bf03e7ab9106af5fac93475347b66e1e

What's Included?

** Nios® II EDS on Windows requires Ubuntu 18.04 LTS on Windows Subsystem for Linux (WSL), which requires a manual installation.

** Nios® II EDS requires you to install an Eclipse IDE manually.

2. Extract the downloaded file.
3. Run the **setup.bat** file.
4. During installation make sure the following components are selected.



Installing Quartus Prime Lite Edition (Free) 20.1.1.720

Select Components

Select the components you want to install

- [✓] Quartus Prime Lite Edition (Free)
  - [✓] Quartus Prime (includes Nios II EDS) (8864MB)
  - [✓] Quartus Prime Help (509.6MB)
  - [✓] Devices
    - [ ] Arria II (536.5MB)
    - [✓] Cyclone IV (516.3MB)
    - [ ] Cyclone 10 LP (293.5MB)
    - [ ] Cyclone V (1434.3MB)
    - [ ] MAX II/V (13.1MB)
    - [ ] MAX 10 FPGA (367.9MB)
- [✓] ModelSim - Intel FPGA Starter Edition (Free) (4318.8MB)
- [ ] ModelSim - Intel FPGA Edition (4318.8MB)

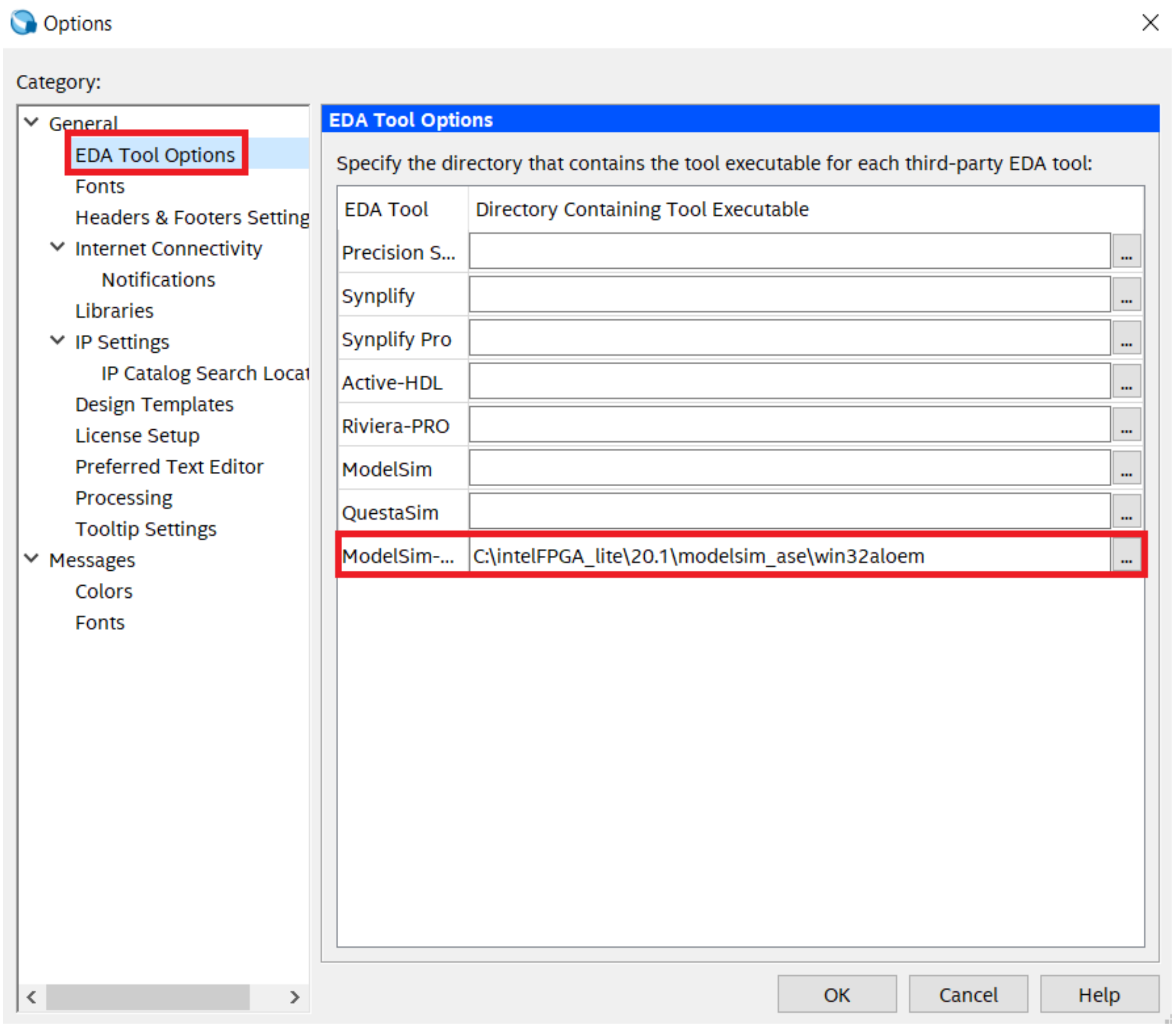
Installs Cyclone 10 LP device support. (293.5MB)

Simulating Verilog HDL using Quartus Prime

1. Find the following icon on your PC and run the software by double-clicking the icon.

2. For using the Quartus Prime very first time we need to locate the directory of ModelSim Altera. For that execute **Tool → Options → EDA Tool Options** and give the proper location of ModelSim Altera and click **OK**. Normally the location is

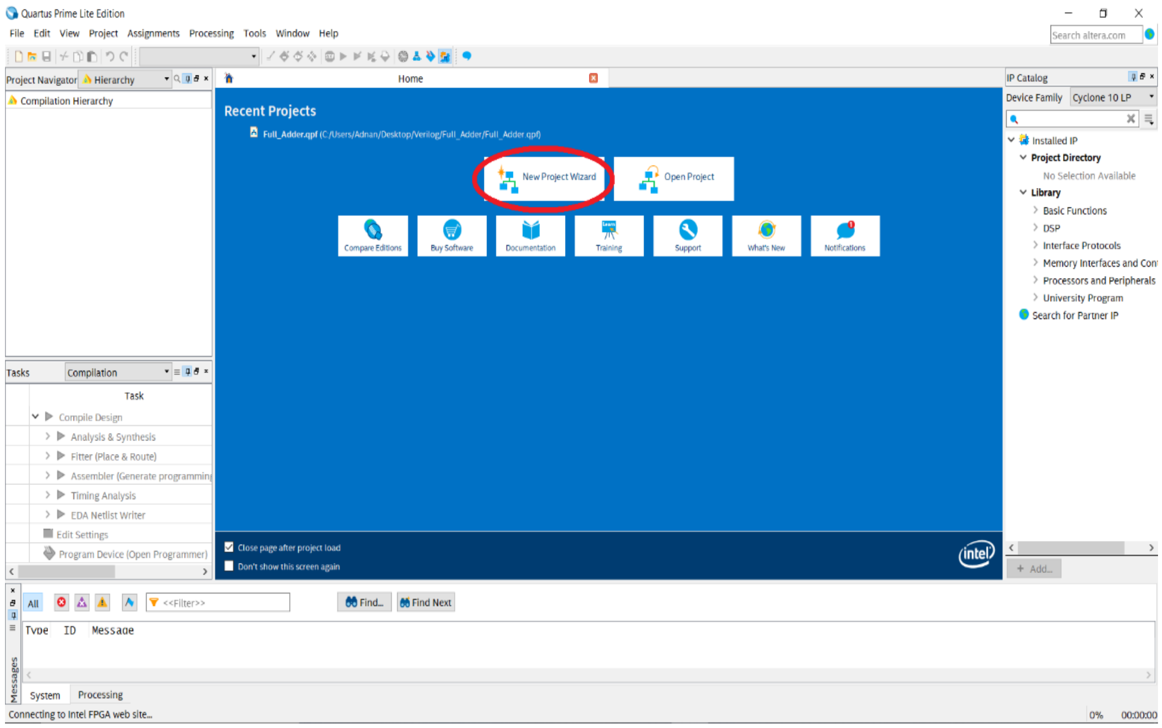
C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem
or
C:\intelFPGA\20.1\modelsim_ase\win32aloem



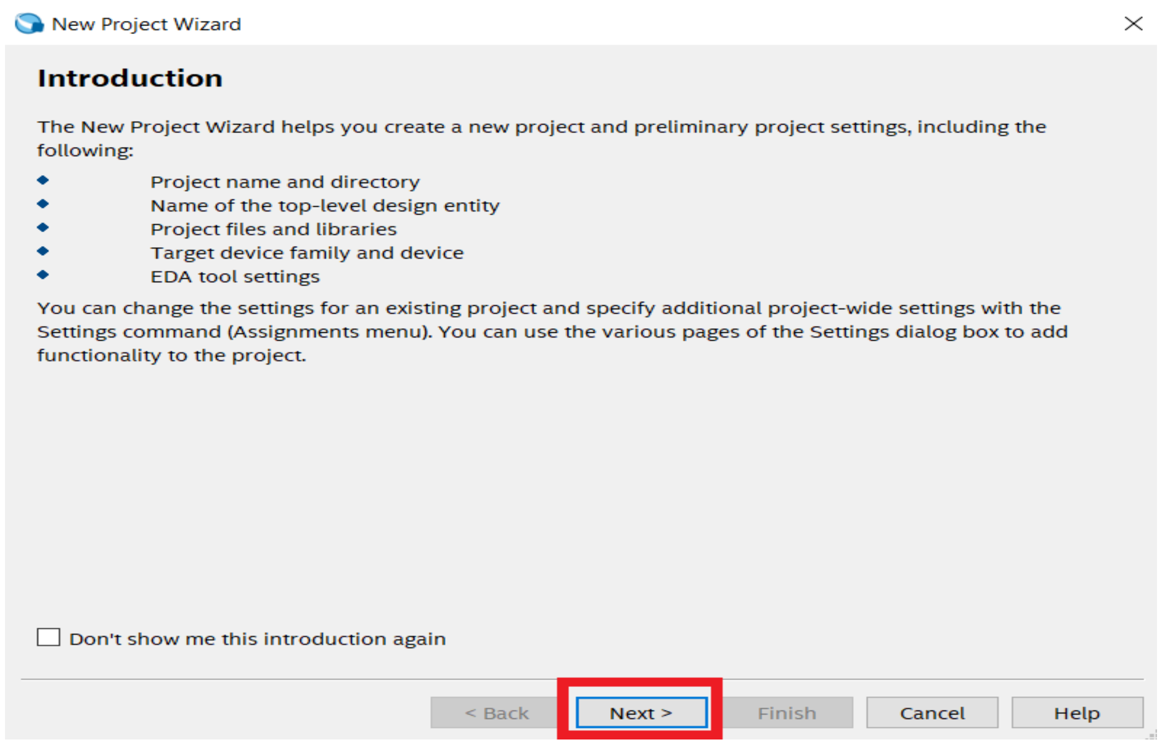
Specify the directory that contains the tool executable for each third-party EDA tool:

EDA Tool	Directory Containing Tool Executable
Precision S...	
Synplify	
Synplify Pro	
Active-HDL	
Riviera-PRO	
ModelSim	
QuestaSim	
ModelSim-...	C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem

3. The following window will pop up. Now click on the **New project wizard**.



4. Click **Next** on the following window.



New Project Wizard

Introduction

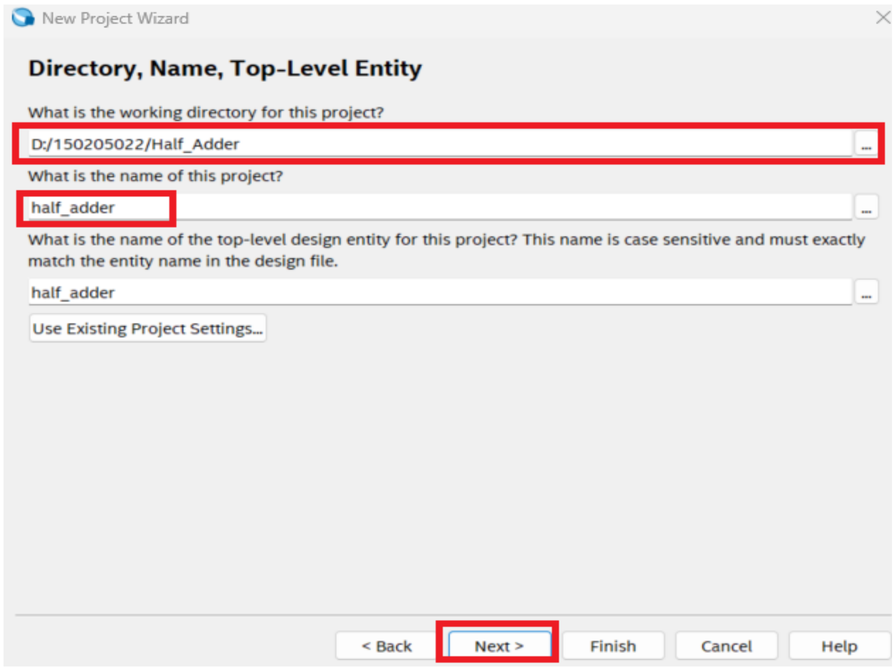
The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

5. In the following window change the working directory of the project to your directory (e.g. D:\150205022/Half_Adder) and give a name to the project as shown and click **Next** four times.

[Project name must be same as the top module]



New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

D:/150205022/Half_Adder

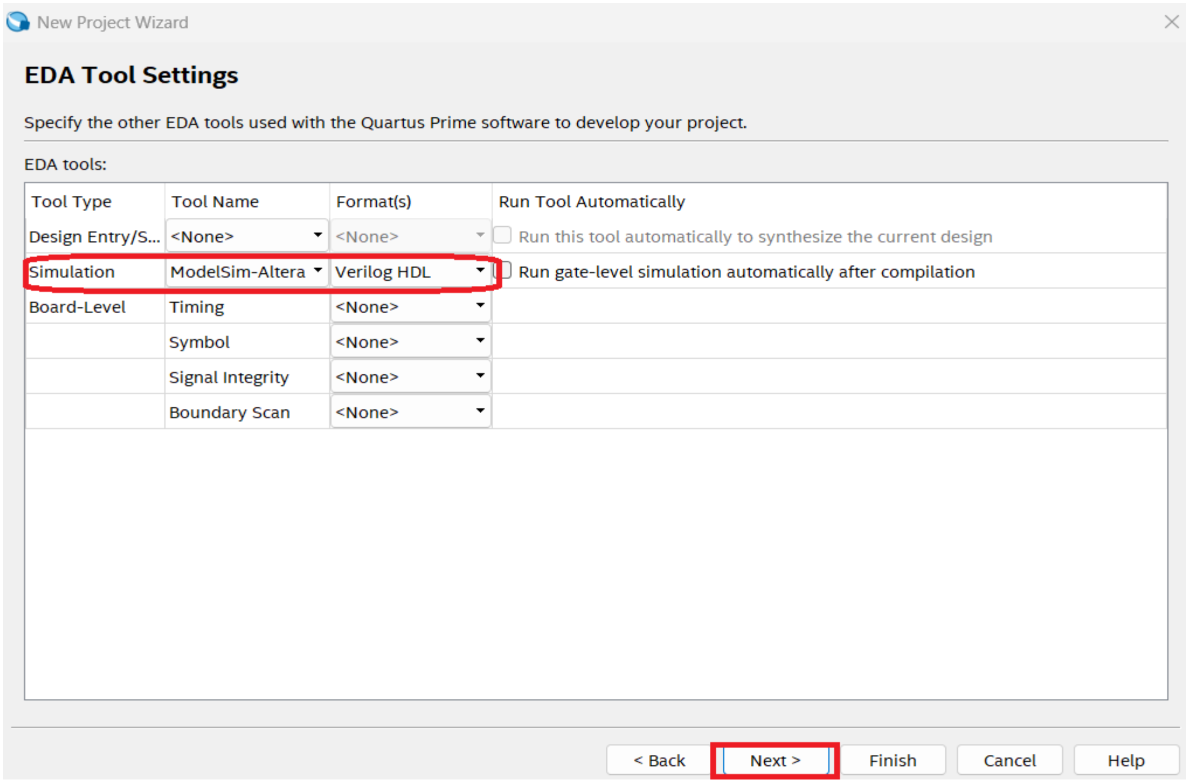
What is the name of this project?

half_adder

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

half_adder

6. After clicking **Next** four times the following window will pop up then set the simulation tool as **"Questa Intel FPGA"** , format as "**Verilog HDL**" and make sure the "**Run gate-level simulation automatically after compilation**" check box is not checked in the EDA tool settings window. After that click **Next.**



New Project Wizard

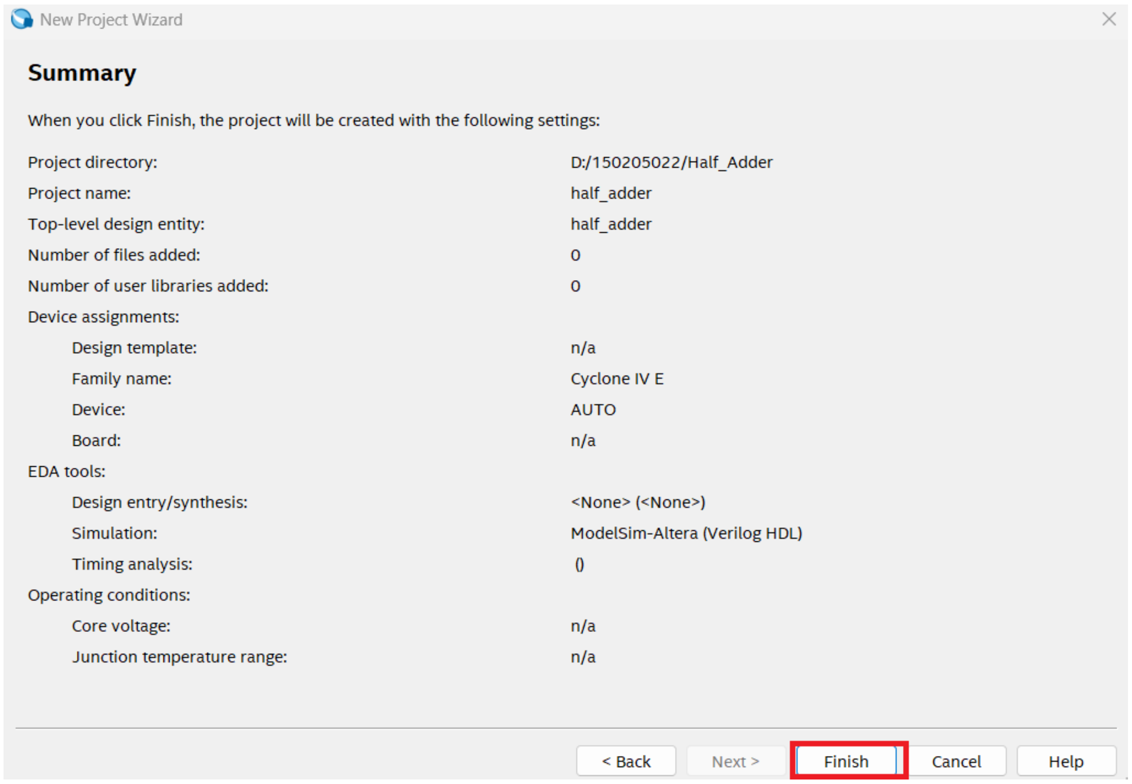
EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/S...	<None>	<None>	☐ Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	Verilog HDL	☐ Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

7. The summary window will appear and click **Finish.**

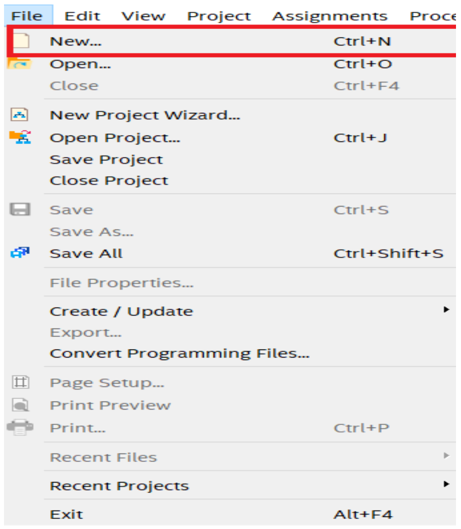

New Project Wizard

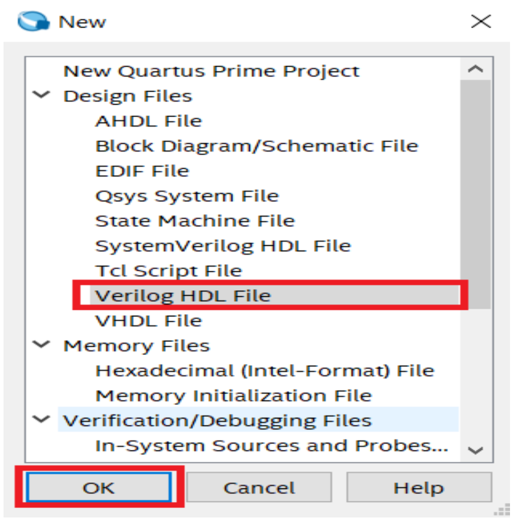
Summary

When you click Finish, the project will be created with the following settings:

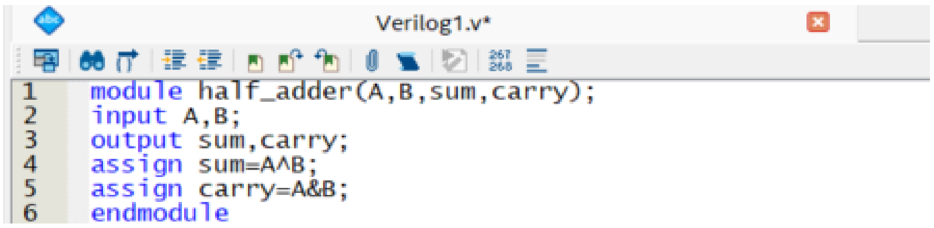
Project directory:	D:/150205022/Half_Adder
Project name:	half_adder
Top-level design entity:	half_adder
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Cyclone IV E
Device:	AUTO
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Altera (Verilog HDL)
Timing analysis:	0
Operating conditions:	
Core voltage:	n/a
Junction temperature range:	n/a

8. Execute **File → New**. In the **New** window select the **Verilog HDL file** and click ok. The editor window will appear.



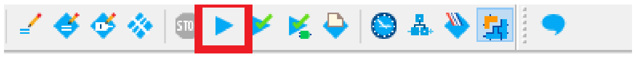

New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes...
OK
Cancel
Help

9. In the editor window write the Verilog module of your design.

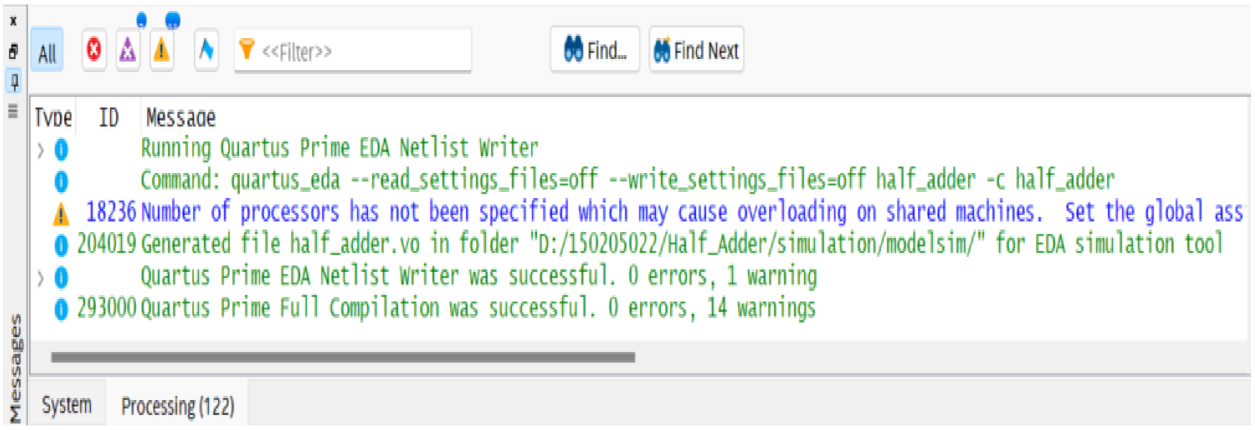


```
module half_adder(A,B,sum,carry);
input A,B;
output sum,carry;
assign sum=A^B;
assign carry=A&B;
endmodule
```

10. Now click on the compilation icon for compiling the design. In this step, the software will ask you to save the Verilog HDL file. Save it with the identical name of the project file.

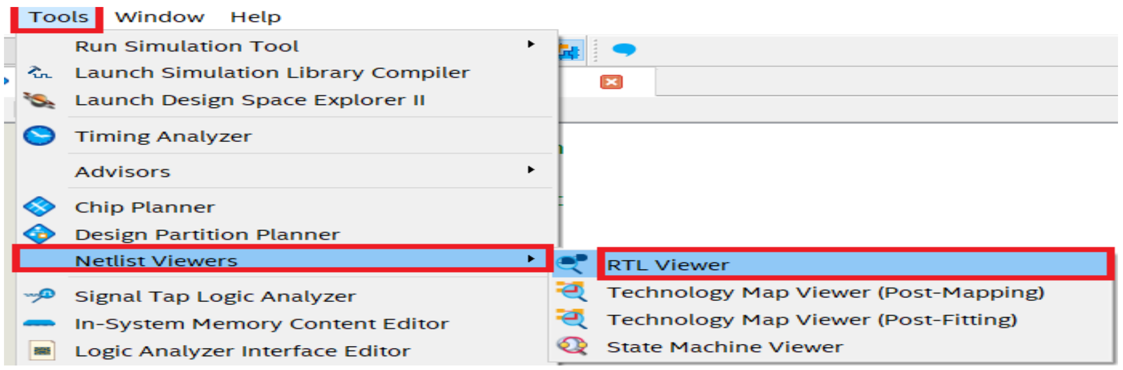


11. After successful compilation you will get the following message. Ignore the warnings.

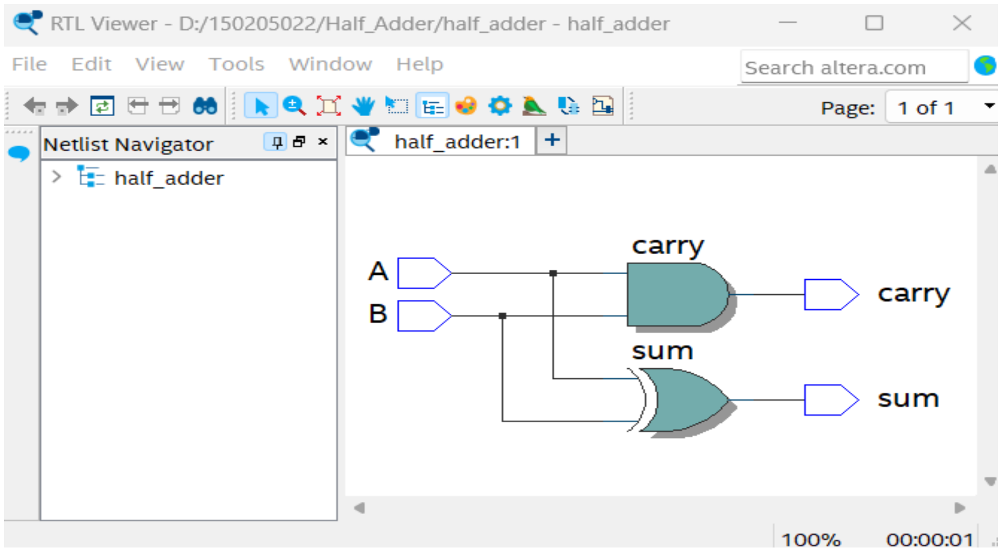


```
Type  ID      Message
      Running Quartus Prime EDA Netlist Writer
      Command: quartus_eda --read_settings_files=off --write_settings_files=off half_adder -c half_adder
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global ass
      204019 Generated file half_adder.vo in folder "D:/150205022/Half_Adder/simulation/modelsim/" for EDA simulation tool
      Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings
```

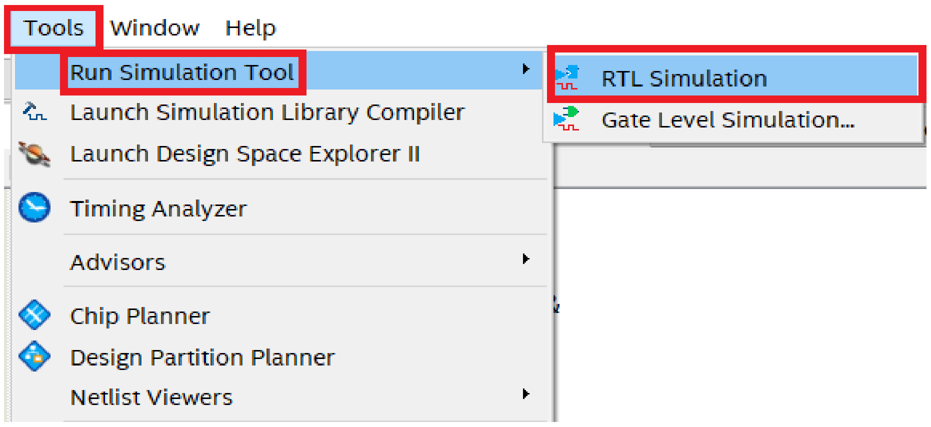
12. The RTL view of the Verilog module can be obtained in Quartus prime by executing **Tools → Netlist Viewers → RTL Viewer.**


Tools
Window
Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
Signal Tap Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
RTL Viewer
Technology Map Viewer (Post-Mapping)
Technology Map Viewer (Post-Fitting)
State Machine Viewer

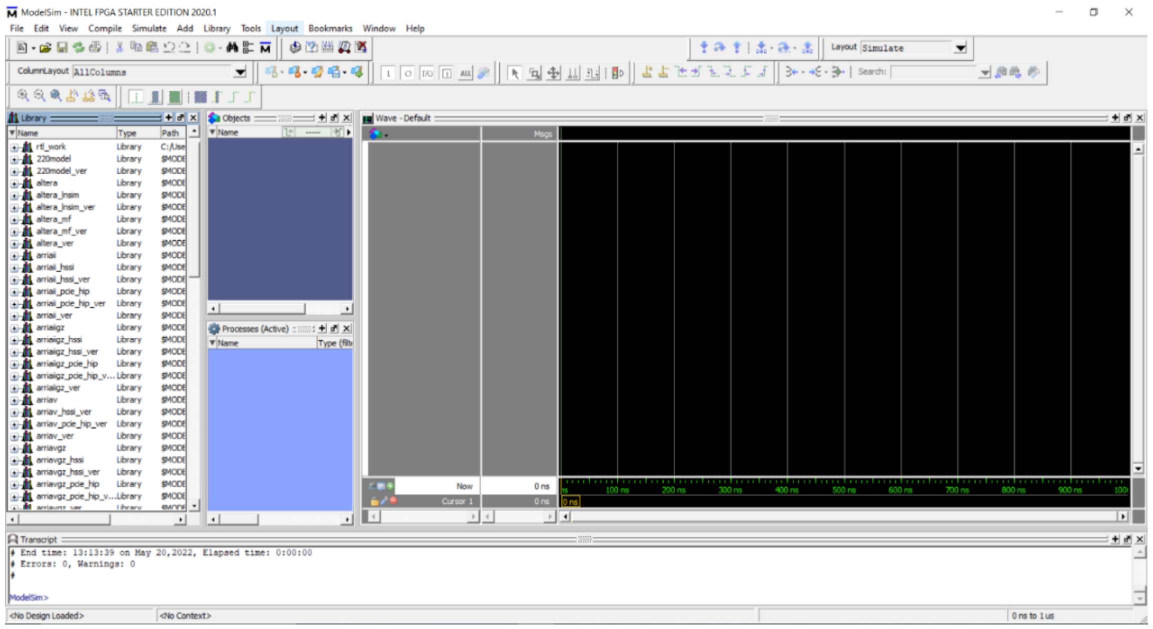
13. The RTL view of our design will be like the following figure.



14. Now to launch ModelSim, execute **Tool→ Run Simulation Tool → RTL Simulation.**



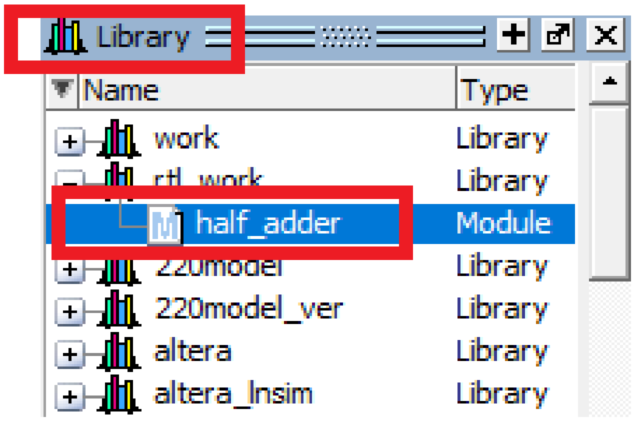
15. The ModelSim will be launched, and the following window will open.


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Library
Objects
Wave - Default
Processes (Active)
Transcript

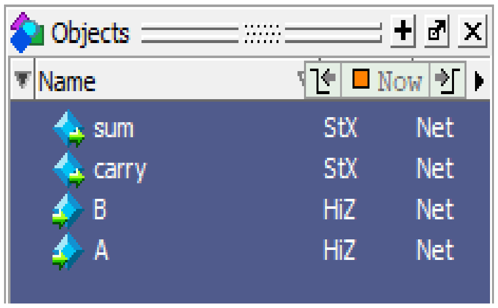
```
# End time: 13:13:39 on May 20,2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
ModelSim>
```

<No Design Loaded>   <No Context>

16. Now go to the **Library** window present on the left side of the ModelSim window and execute **rtl_work→ <double click on your project module name>**

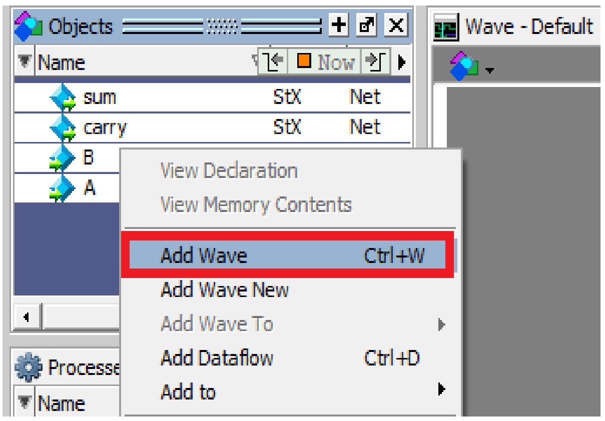


17. The input and output variables defined in the Verilog will appear in the **Objects** window.

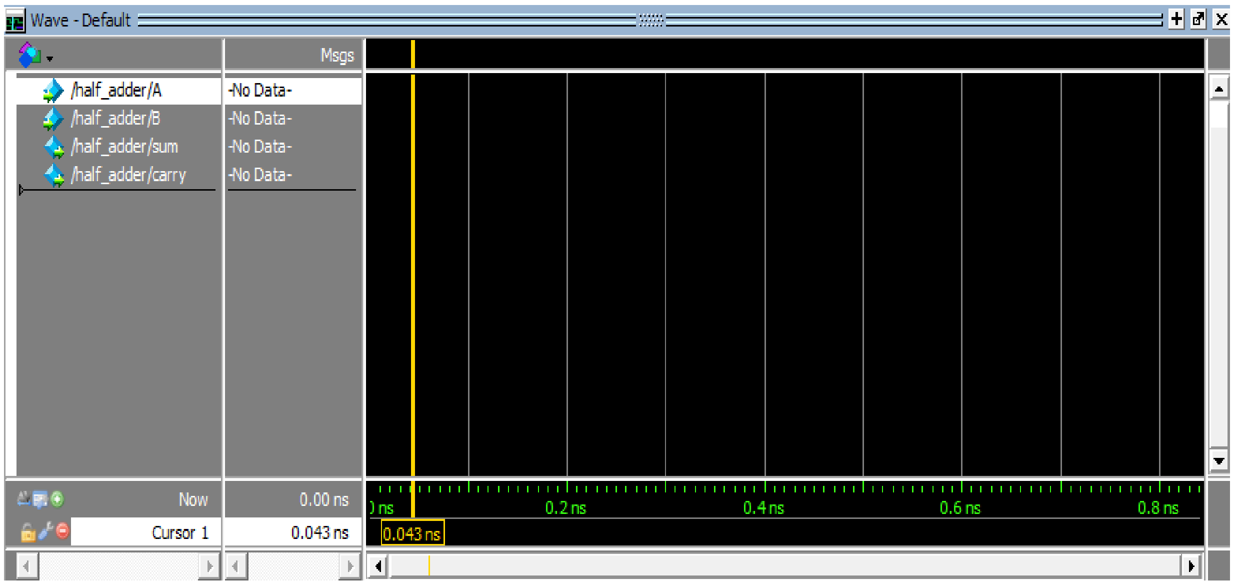


Name		
sum	StX	Net
carry	StX	Net
B	HiZ	Net
A	HiZ	Net

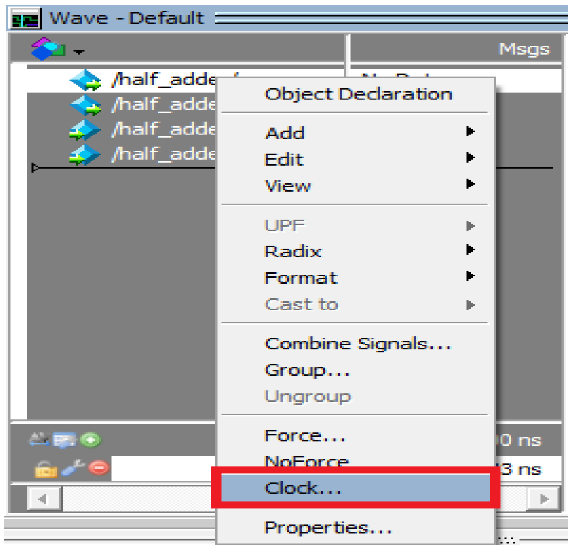
18. Now select all the input and output variables of the **Objects** window and by right-clicking on your mouse execute **Add Wave** to place them in the **Wave** window.



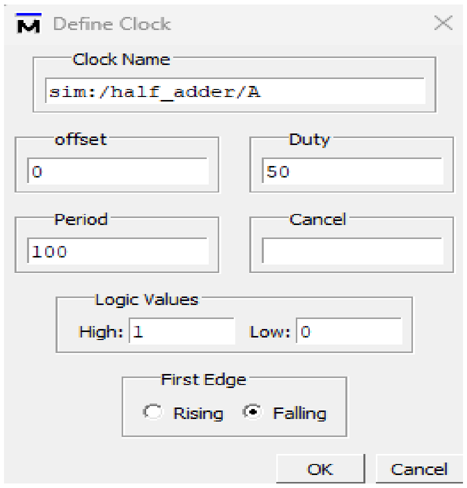
19. All the input and output variables will be placed on the **wave** window and the wave window will look like the following.



20. Now apply clocks to each input variable. Right-clicking any input variable and from the popped-up menu execute **Modify → Clock.**



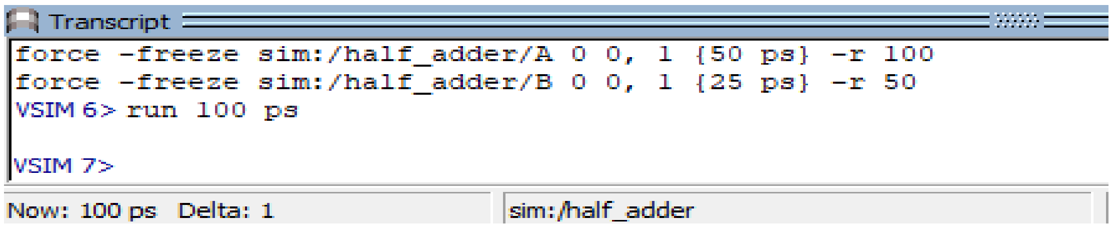
21. The **Define Clock** window will appear. Set parameters as per your requirement keep in mind all the units are in picoseconds by default.



Clock Name: sim:/half_adder/A
offset: 0
Duty: 50
Period: 100
Cancel:
Logic Values High: 1 Low: 0
First Edge: Rising [✓ Falling]

22. After defining all the input clocks, to evaluate the outputs write **run 100 ps** on the Transcript of ModelSim. Then the simulation will be performed for 100 ps.

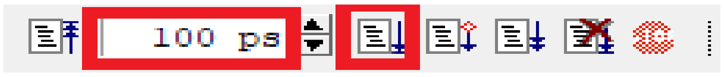
[Give run length according to your requirement.]



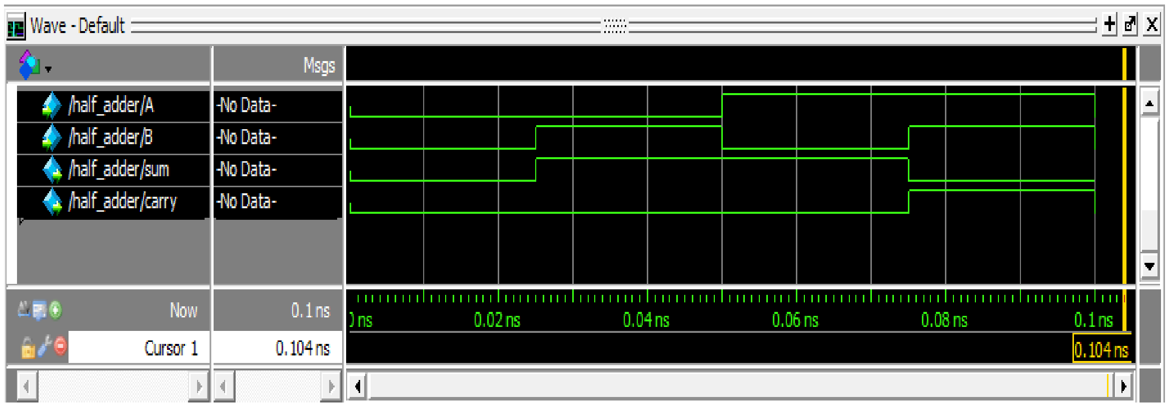
```
Transcript
force -freeze sim:/half_adder/A 0 0, 1 {50 ps} -r 100
force -freeze sim:/half_adder/B 0 0, 1 {25 ps} -r 50
VSIM 6> run 100 ps

VSIM 7>
Now: 100 ps  Delta: 1    sim:/half_adder
```

Alternatively, we can run the wave output using the **Run** icon by typing the **Run length**



23. The wave window will look like the following figure after simulation.

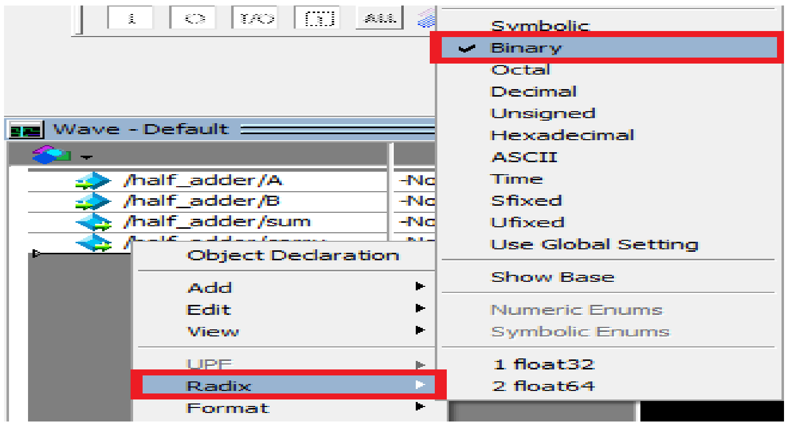


If you need to change the clock pulse you must reset all the clocks before changing clocks otherwise the inputs and outputs will change after the previous run time which is not a convenient way to represent the inputs and outputs. The command **"restart"** is used in the transcript for resetting all the clocks. Alternatively, restart can be performed by executing **Simulate → Restart**

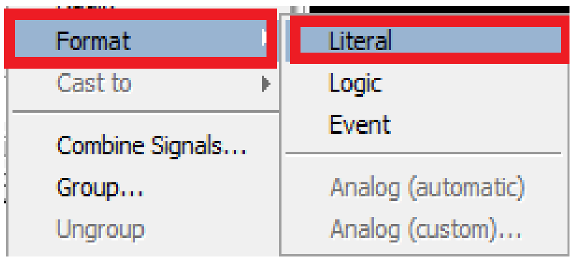
Showing Binary values on the Wave

Sometimes it is hard to verify the functionality of a digital system from the wave. For easy functional verification, we can read the binary values from the wave of ModelSim by doing the following steps.

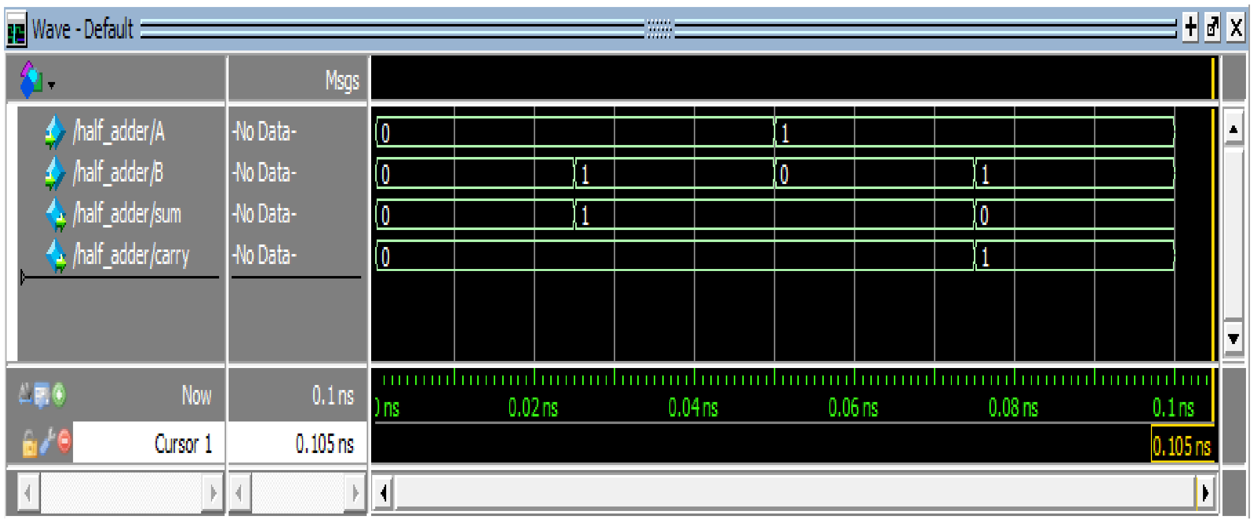
I. Select all the input and output variables on the clock and right-click on the mouse and execute **Radix → Binary**.



II. After changing the **Radix,** change the **Format** type similarly by selecting all input and output variables on the wave by right-clicking on the mouse and then executing **Format → Literal**.

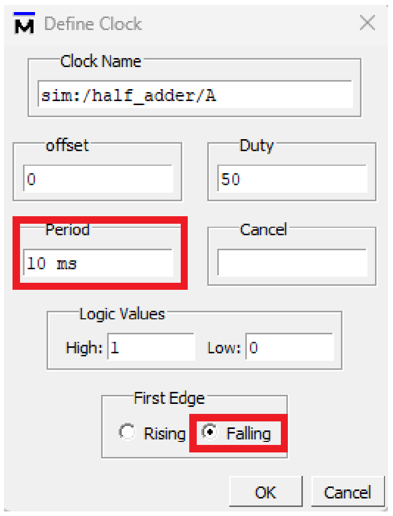


III. Now on the wave, binary values will be displayed which can be easily analyzed.



Changing Clock Unit

In step 21 it is mentioned that ModelSim's default timing unit is picosecond. But in some cases, we may need to define clocks in other units. Let us consider, that we need to define the period of a, b, and c as 10ms, 5ms, and 2.5ms respectively. Now define the clock a, b, and c as shown in the below figures.

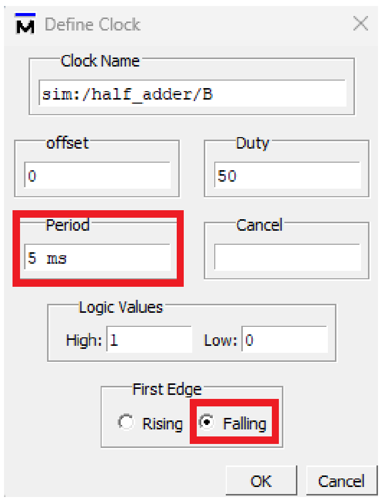

Define Clock

Clock Name: sim:/half_adder/A

offset: 0
Duty: 50
Period: 10 ms
Cancel:

Logic Values: High: 1 Low: 0

First Edge: Rising [✓ Falling]


Define Clock

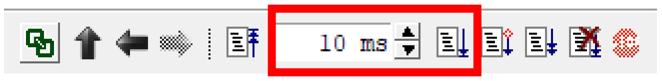
Clock Name: sim:/half_adder/B

offset: 0
Duty: 50
Period: 5 ms
Cancel:

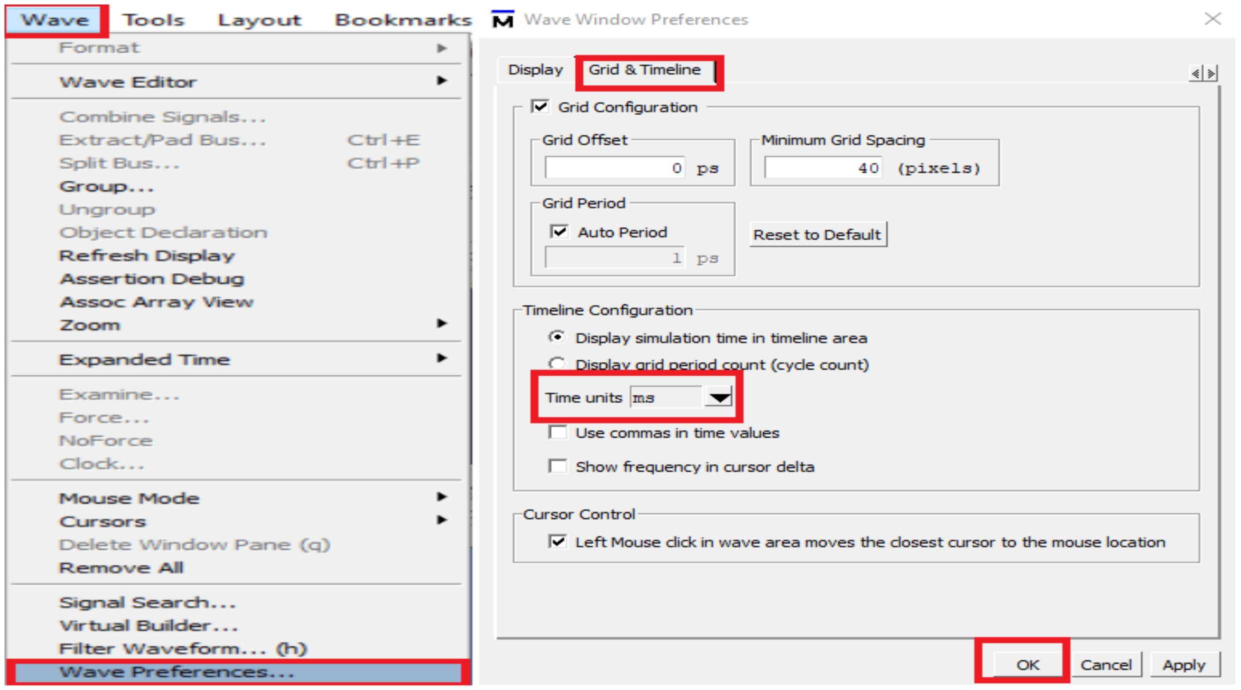
Logic Values: High: 1 Low: 0

First Edge: Rising [✓ Falling]

To view the output for all the input combinations the run length should be equal to the maximum period.

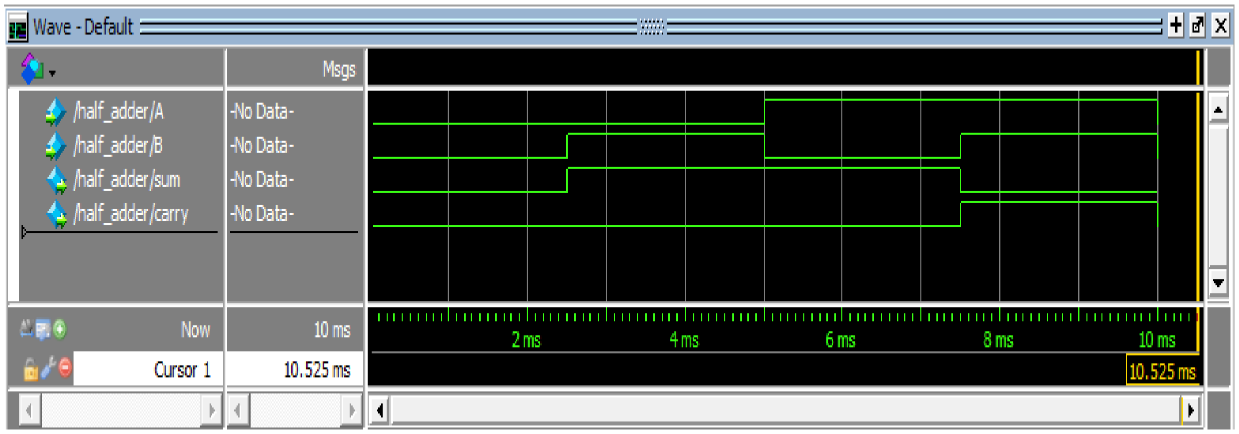

10 ms

As all the units are in milliseconds, for easy visualization we can change the time units of the wave grid by executing **Wave → Wave Preferences → Grid & Timeline → Time units → ms**.



[✓] Grid Configuration
Grid Offset: 0 ps
Minimum Grid Spacing: 40 (pixels)
Grid Period: [✓] Auto Period, 1 ps
Timeline Configuration
(●) Display simulation time in timeline area
( ) Display grid period count (cycle count)
Time units: ms
[ ] Use commas in time values
[ ] Show frequency in cursor delta
Cursor Control
[✓] Left Mouse click in wave area moves the closest cursor to the mouse location

Now the ModelSim wave window will look like the following figure.



Similarly, for femtoseconds, nanoseconds, and microseconds, we can use **fs**, **ns,** and **ms** respectively

Lab-2: Digital System Design Using Verilog HDL

Example 01

Example 01 demonstrates the Verilog HDL code of a full adder following the **Hierarchical Modeling** style. In the design, a half-adder module is constructed from the predefined logic gates and then the half-adder module is used twice to design the full adder.

```
module Full_Adder(A,B,Cin,sum,carry); // Top module
input A,B,Cin;
output sum,carry;
wire s1,c1,c2;
Half_Adder sm1(s1,c1,A,B);
Half_Adder sm2(sum,c2,s1,Cin);
or o1(carry,c1,c2);
endmodule

module Half_Adder(s,c,x,y);  // macro cell
input  x,y;
output s,c;
xor s1(s,x,y);   // predefined primitive or leaf cells
and c1(c,x,y);
endmodule
```

N.B. One module can be instantiated to another module without maintaining the I/O sequence using the **Name Wise Instantiation** or **Explicit method** (**.**Exact_Port(Port_to_be_Assigned)).

Example 02

Example 02 demonstrates the Verilog HDL code of a 4 to 2 priority encoder with a valid bit.

```
module p_encoder_4to2(D,Y,V);
input [3:0]D;          //declaring variable for input
output reg [1:0]Y;   //declaring variable for output
output reg V;        //declaring the variable for valid bit
always@ *
begin
        casex(D)
                4'b0001:
                begin
                        Y=2'b00; V=1;
                end
                4'b001x:
                begin
                        Y=2'b01; V=1;
                end
                4'b01xx:
                begin
                        Y=2'b10; V=1;
                end
                4'b1xxx:
                begin
                        Y=2'b11; V=1;
                end
                default:
                begin
                        Y=2'bx; V=0;
                end
        endcase
end
endmodule
```

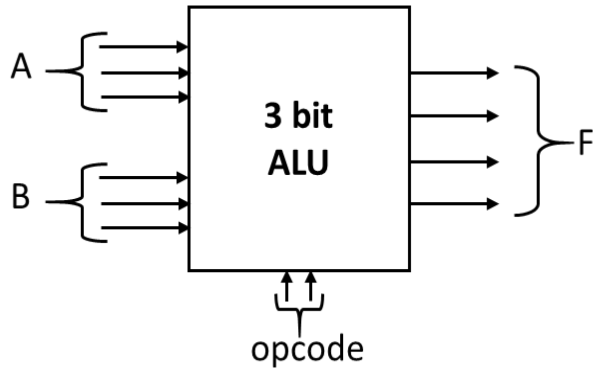
Example 03

Example 03 demonstrates the Verilog HDL code of a 2 to 4 decoder.

```
module decoder_2to4(s,e,y);
input [1:0] s;
input e;
output reg [3:0]y;
integer k;
always@ (s,e)
begin
        for (k=0;k<=3;k=k+1)
                begin
                if ((s==k) && (e==1))
                        y[k]=1;
                else
                        y[k]=0;
                end
end
endmodule
```

Example 04

Example 04 demonstrates the Verilog HDL code of a 3-bit arithmetic logic unit that works according to the function table mentioned in *Table-1*



Block representation of the ALU of Example 04

Table-1

Operation Code	**Function**
00	Binary sum of A and B
01	Adds 1 with B
10	2s complement of A
11	Bitwise XOR operation between A and B

```
module ALU_4bit(A,B,F,opcode);
input [2:0]A, B;
input [1:0]opcode;
output reg [3:0]F;
always@ (*)
begin
case(opcode)
            2'b00:F=A+B;
            2'b01:F=B+1;
            2'b10: F=-A;
            2'b11: F=A^B;
            default:F=0;
endcase
end
endmodule
```

Student Task:

1. Write a Verilog code to implement the system shown in *figure-a* using two submodules and a top module. Hence verify the output X and Y with the clock pulse shown in *figure-b*.

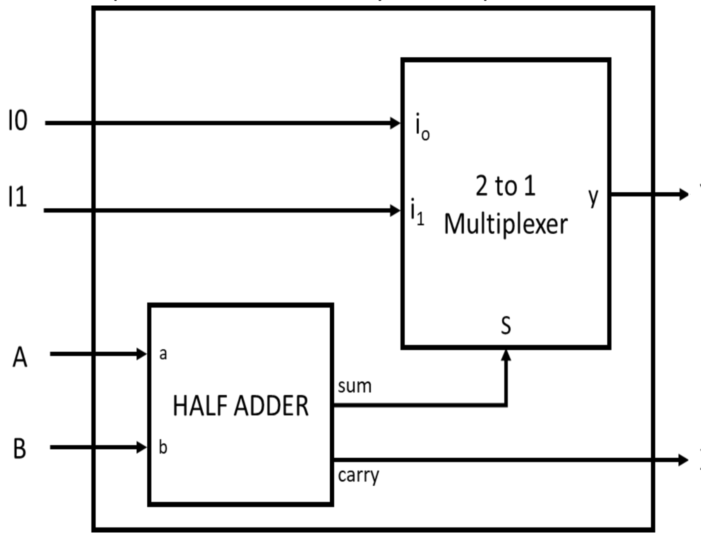


figure-a

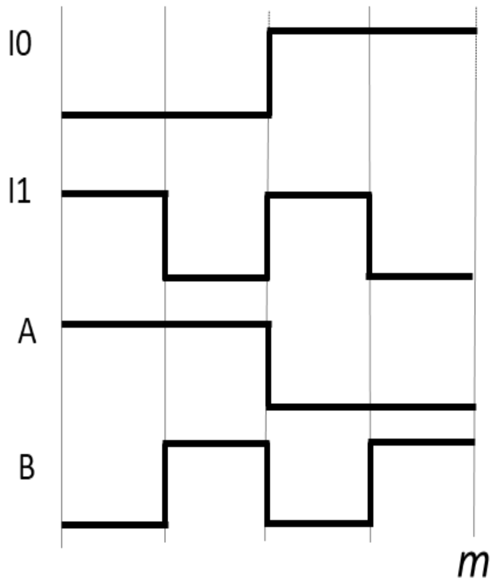


figure-b

2. Write a Verilog code to implement a 4x1 Multiplexer with the minimum number of 2x1 Multiplexers.
3. Write a Verilog code to implement a BCD adder using behavioral statements.
4. Write a Verilog code to implement a 4-bit magnitude comparator.
5. Write a Verilog code to implement a 4-bit ripple carry adder using full adders.

Lab-3: Sequential System Design Using Verilog HDL

Example 01

Example 01 demonstrates the Verilog HDL code of a SR Latch

```
module sr_latch(s,r,Q,clk);
input clk,s,r;
output reg Q;
always@ *
begin
if(clk==1)
        begin
                case ({s,r})
                2'b00:Q=Q;
                2'b01:Q=0;
                2'b10:Q=1;
                default:Q=1'bx;
                endcase
        end
end
endmodule
```

Example 02

Example 02 demonstrates the Verilog HDL code of a JK Latch.

```
module jk_latch(J,K,Q,clk);
input clk,J,K;
output reg Q;
always@ (*)
begin
        if(clk==1)
        begin
        case({J,K})
                2'b00:Q=Q;
                2'b01:Q=0;
                2'b10:Q=1;
                2'b11:Q=~Q;
                default:Q=1'bx;
        endcase
        end
end
endmodule
```

Example 03

Example 03 demonstrates the Verilog HDL code of a D Latch using behavioral statements.

```
module D_latch(D,Q,clk);
input clk,D;
output reg Q;
always@ (*)
begin
if (clk==1)
        Q=D;
end
endmodule
```

Example 04

Example 04 demonstrates the Verilog HDL code of a T Latch using behavioral statements.

```
module T_latch(T,Q,clk);
input T,clk;
output reg Q;
always@ (*)
begin
        if (clk==1)
        begin
                if(T==0)
                        Q=Q;
                else if(T==1)
                        Q=~Q;
        end
end
endmodule
```

Example 05

Example 05 demonstrates the Verilog HDL code of a positive edge triggered D flip-flop.

```
module D_FF(clk, D,Q);
input D,clk;
output reg Q;
always@(posedge clk)
        Q=D;
endmodule
```

Example 06

Example 06 demonstrates the Verilog HDL code of a positive edge triggered D flip-flop with synchronous set and reset.

```
module D_FF(clk, rst, set, D, Q);
input D,set,rst,clk;
output reg Q;
always@ (posedge clk)
begin
        if(rst==1)
                Q=0;
        else if(set==1)
                Q=1;
        else
                Q=D;
end
endmodule
```

Example 07

Example 07 demonstrates the Verilog HDL code of a positive edge triggered D flip-flop with asynchronous reset.

```
module D_FF(clk, rst, set, D, Q);
input D,set,rst,clk;
output reg Q;
always@ (posedge clk, posedge rst)
begin
        if(rst==1)
                Q=0;
        else if(set==1)
                Q=1;
        else
                Q=D;
end
endmodule
```

Example 08

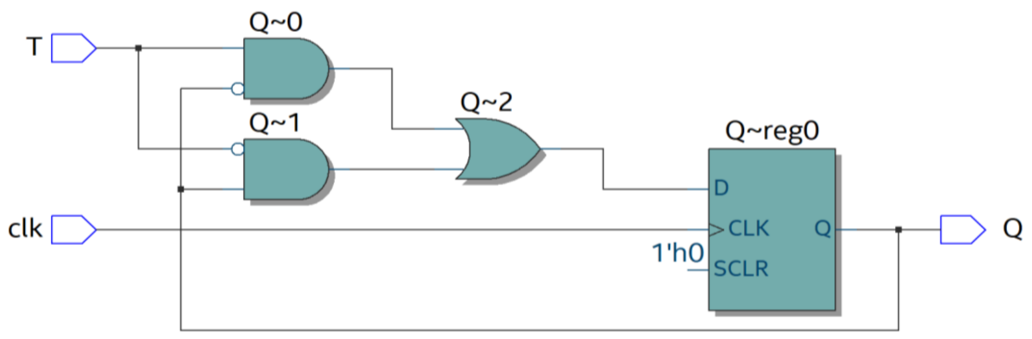


figure-a

Example 08 demonstrates the Verilog HDL code of the RTL view shown in figure-a.

```
module rtl1(T,clk,Q);
input T,clk;
output reg Q;
always@(posedge clk)
begin
  Q<=(T&(~Q))|(Q&(~T));
end
endmodule
```

Example 09

Example 09 demonstrates the Verilog HDL code of a 2-bit up counter with synchronous enable.

```
module up_count(clk,E,Q);
input clk,E;
output reg [1:0]Q;
always@(posedge clk)
if(!E)
        Q=0;
else
        Q=Q+1;
endmodule
```

Example 10

Example 10 demonstrates the Verilog HDL code of a 4-bit shift registrar.

```
module shift_reg(w,clk,Q);
input w,clk;
output reg [3:0]Q;
always@(posedge clk)
begin
            Q[3]<=w;
            Q[2]<=Q[3];
            Q[1]<=Q[2];
            Q[0]<=Q[1];
end
endmodule
```

Student Task

1. Verify the functionality of **Example-09** using the clock pulses shown in *figure-a*.

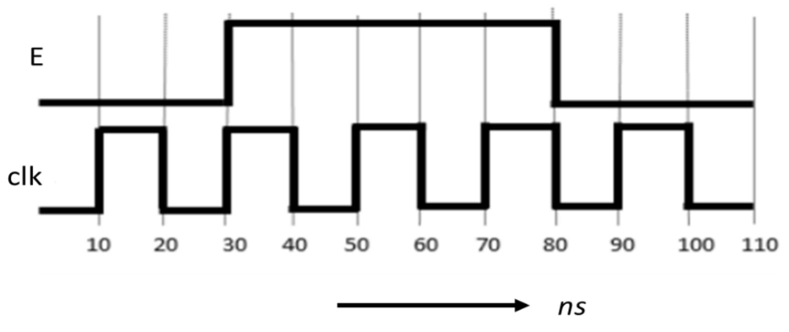


figure-a

2. Write a Verilog code to obtain the same logic circuit of *figure-b* in **'RTL Viewer'**.

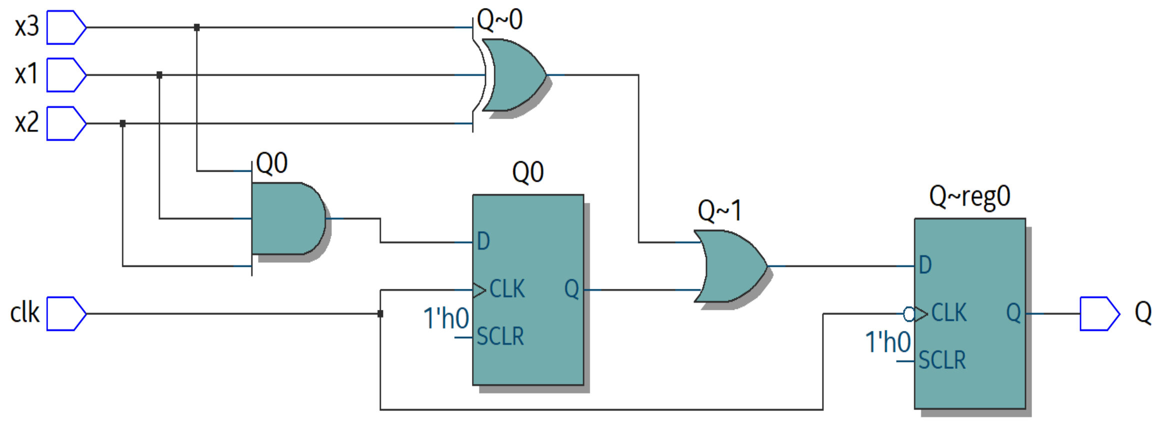


figure-b

3. What is the difference between the outputs of the two modules (test1 and test2) shown below?

```
module test1(a,b,y);
input a,b;
output reg y;
always@(*)
begin
        y=a;
        y=y|b;
        y=y^b;
end
endmodule
```

```
module test2(a,b,y);
input a,b;
output reg y;
always@(*)
begin
        y<=a;
        y<=y|b;
        y<=y^b;
end
endmodule
```

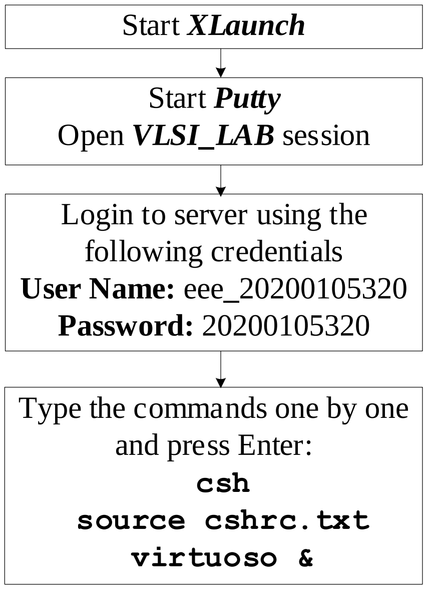
4. Write a Verilog code to design a BCD counter.
5. Write Verilog code for a universal shift register which can shift left/right and can load data both in parallel mode and serial mode.
6. Write the Verilog code for a logic circuit which counts to 3 if selection input (composed of 2 bits) S_1S_0 = 01, counts to 7 if S_1S_0 = 10, counts to 15 if S_1S_0 = 11. If S_1S_0 = 00, then it halts counting. Use positive edge-triggered counter.

Lab-4: CMOS Inverter Design Using Cadence Virtuoso

Objectives:

- To learn how to logging into the Cadence Virtuoso.
- To learn how to draw schematic of basic logic gates in Cadence Virtuoso.
- To learn how to create symbol view from schematic view

Steps to Login into the server



Start ***XLaunch***

Start ***Putty***
Open ***VLSI_LAB*** session

Login to server using the following credentials
User Name: eee_20200105320
Password: 20200105320

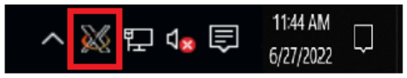
Type the commands one by one and press Enter:

```
csh
source cshrc.txt
virtuoso &
```

The detailed instructions are given below

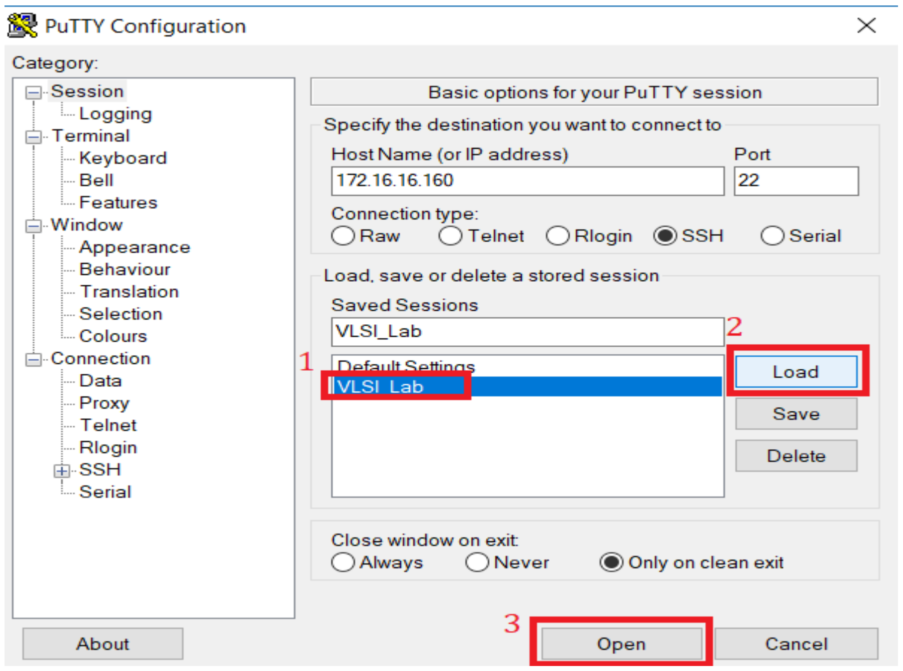
1. Find the Desktop shortcut icon for **XLaunch**. Double-click on it. Click **Next, Next, Next**, **Finish** (in that order) in the windows that pop up one after another.

   After it starts, you will see the Xming icon at the bottom right corner of your Desktop screen.

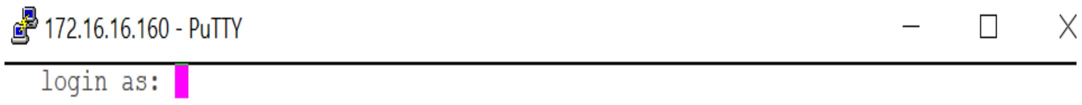
2. Find the icon for Putty. Double click on it to open it. ‘Putty Configuration’ window will pop-up

3. Select **VLSI_LAB** under the **‘Saved Sessions’** category. Click **Load** and then click **Open.**



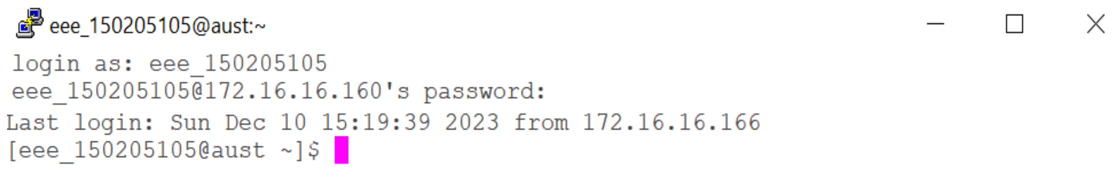
Host Name (or IP address): 172.16.16.160
Port: 22
Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial
Saved Sessions: VLSI_Lab
Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

4. Now you will see a Terminal window which prompts you for login.



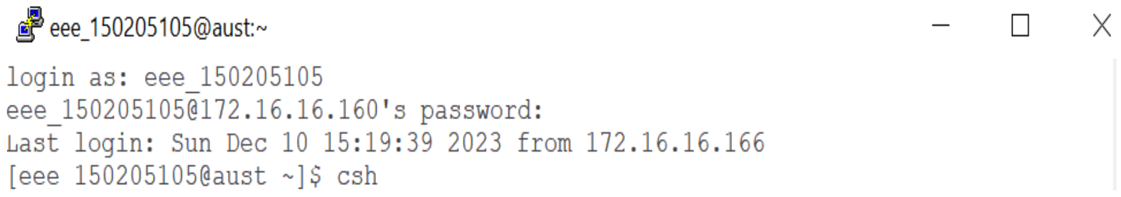
```
login as:
```

5. Log in to your workstation using your username and password. **Your username will be eee_<student id> and your password will be your student ID**. When you are typing your password, the command window will not display the characters you type in, so make sure you are typing the right password. After logging in to your account, the terminal window should look like the following:



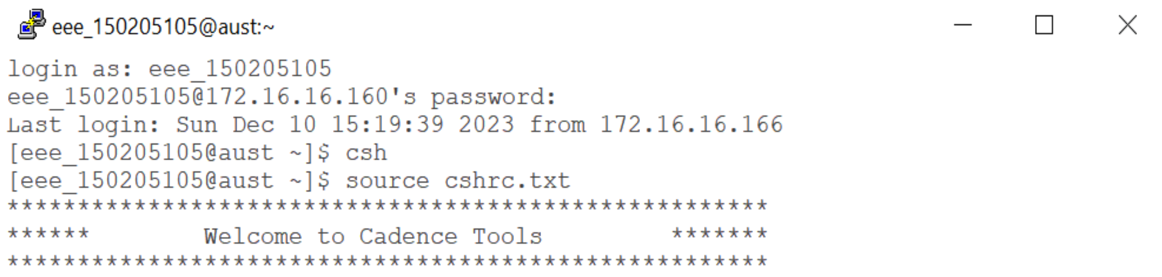
```
login as: eee_150205105
eee_150205105@172.16.16.160's password:
Last login: Sun Dec 10 15:19:39 2023 from 172.16.16.166
[eee_150205105@aust ~]$
```

6. Type **csh** and press the **‘Enter’** key.



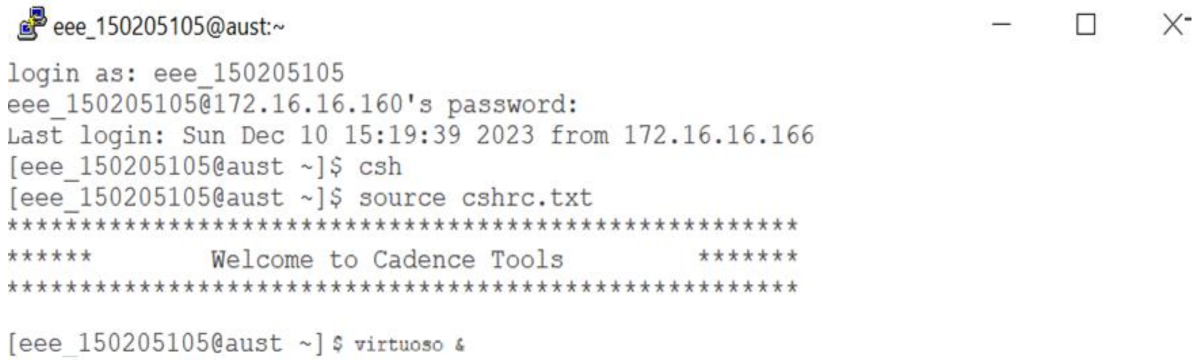
```
login as: eee_150205105
eee_150205105@172.16.16.160's password:
Last login: Sun Dec 10 15:19:39 2023 from 172.16.16.166
[eee_150205105@aust ~]$ csh
```

7. Then type **source cshrc_q** and press the **‘Enter’** key. The following message will be displayed in the Terminal window: Welcome to Cadence Tools That means you can use Cadence tools now.



```
eee_150205105@aust:~
login as: eee_150205105
eee_150205105@172.16.16.160's password:
Last login: Sun Dec 10 15:19:39 2023 from 172.16.16.166
[eee_150205105@aust ~]$ csh
[eee_150205105@aust ~]$ source cshrc.txt
*******************************************************
******        Welcome to Cadence Tools         *******
*******************************************************
```

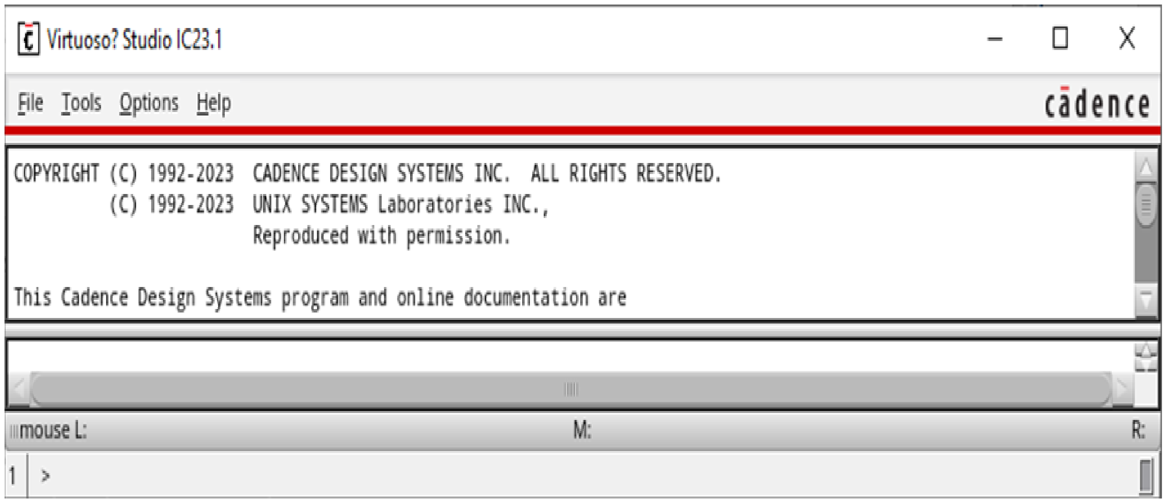
8. Then type **virtuoso &** in the command prompt as shown below.



```
eee_150205105@aust:~
login as: eee_150205105
eee_150205105@172.16.16.160's password:
Last login: Sun Dec 10 15:19:39 2023 from 172.16.16.166
[eee_150205105@aust ~]$ csh
[eee_150205105@aust ~]$ source cshrc.txt
*******************************************************
******        Welcome to Cadence Tools         *******
*******************************************************

[eee_150205105@aust ~] $ virtuoso &
```

9. Virtuoso® **Command Interpreter Window (CIW)** appears at the bottom of the screen. From the CIW menus, all Cadence main tools, online help and options can be accessed. In the window area, all kind of messages (info, errors, warnings, etc) generated by the different Cadence tools appear. You can also introduce commands.



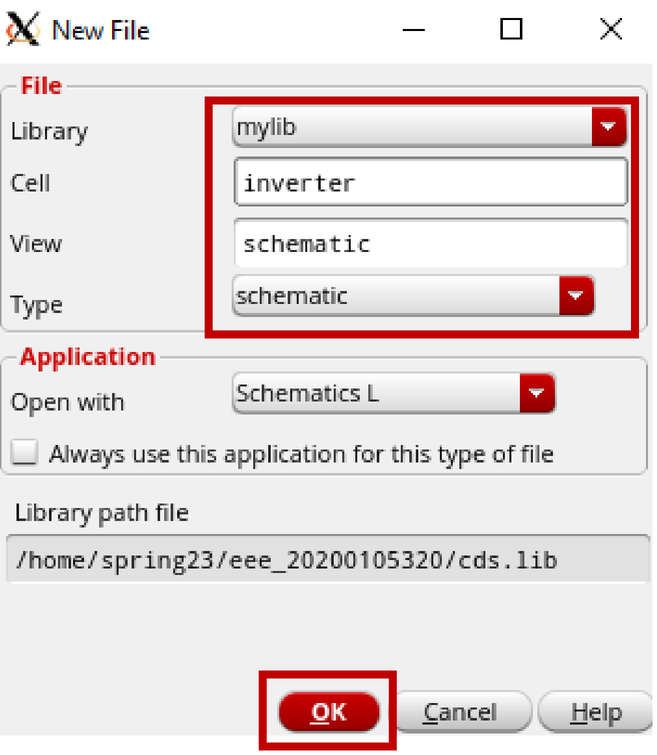
```
COPYRIGHT (C) 1992-2023  CADENCE DESIGN SYSTEMS INC.  ALL RIGHTS RESERVED.
          (C) 1992-2023  UNIX SYSTEMS Laboratories INC.,
                         Reproduced with permission.

This Cadence Design Systems program and online documentation are
```

Creating a Schematic

1. In the **Command Interpreter Window (CIW)**, execute ***File→New→Cellview***. Set up the '**New File**' form as follows:

Library: *mylib*, **Cell:** *inverter*, **View:** *schematic*, **Type:** *schematic*, **Application: Open with:** *Schematics L*



New File

File

Library	mylib
Cell	inverter
View	schematic
Type	schematic

Application

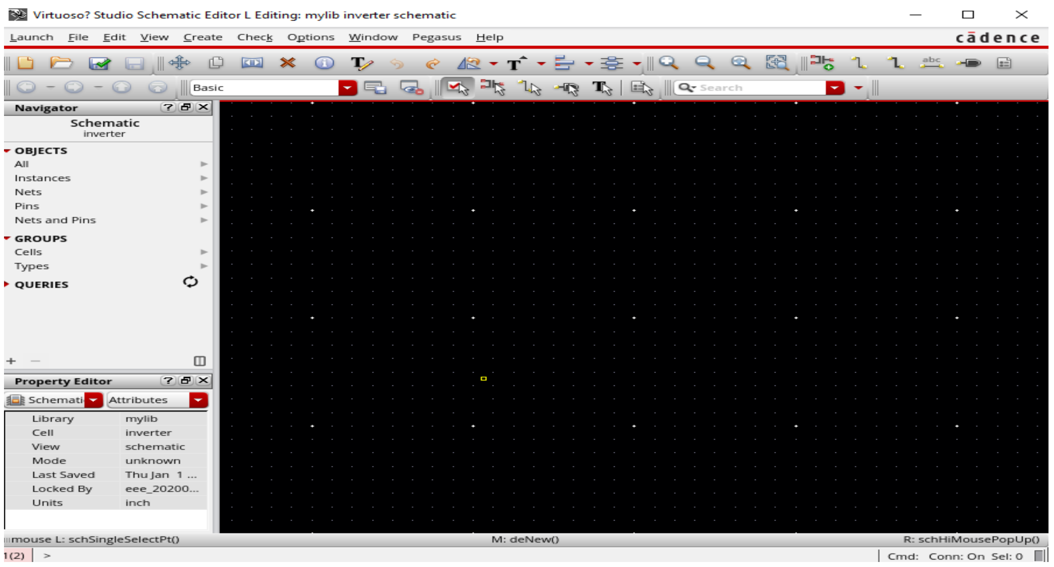
Open with: Schematics L

☐ Always use this application for this type of file

Library path file

```
/home/spring23/eee_20200105320/cds.lib
```

A blank schematic window for the inverter design appears.

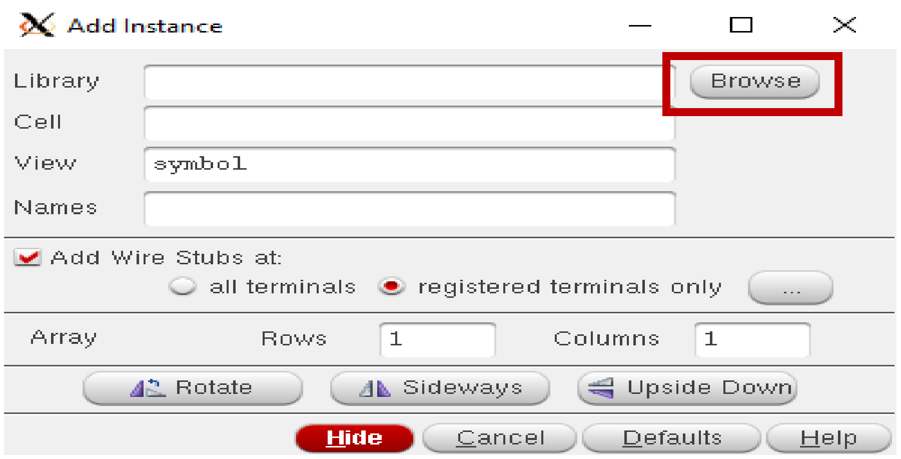


Virtuoso? Studio Schematic Editor L Editing: mylib inverter schematic

Library	mylib
Cell	inverter
View	schematic
Mode	unknown
Last Saved	Thu Jan 1 ...
Locked By	eee_20200...
Units	inch

```
mouse L: schSingleSelectPt()          M: deNew()          R: schHiMousePopUp()
1(2) | >                                          Cmd:   Conn: On  Sel: 0
```

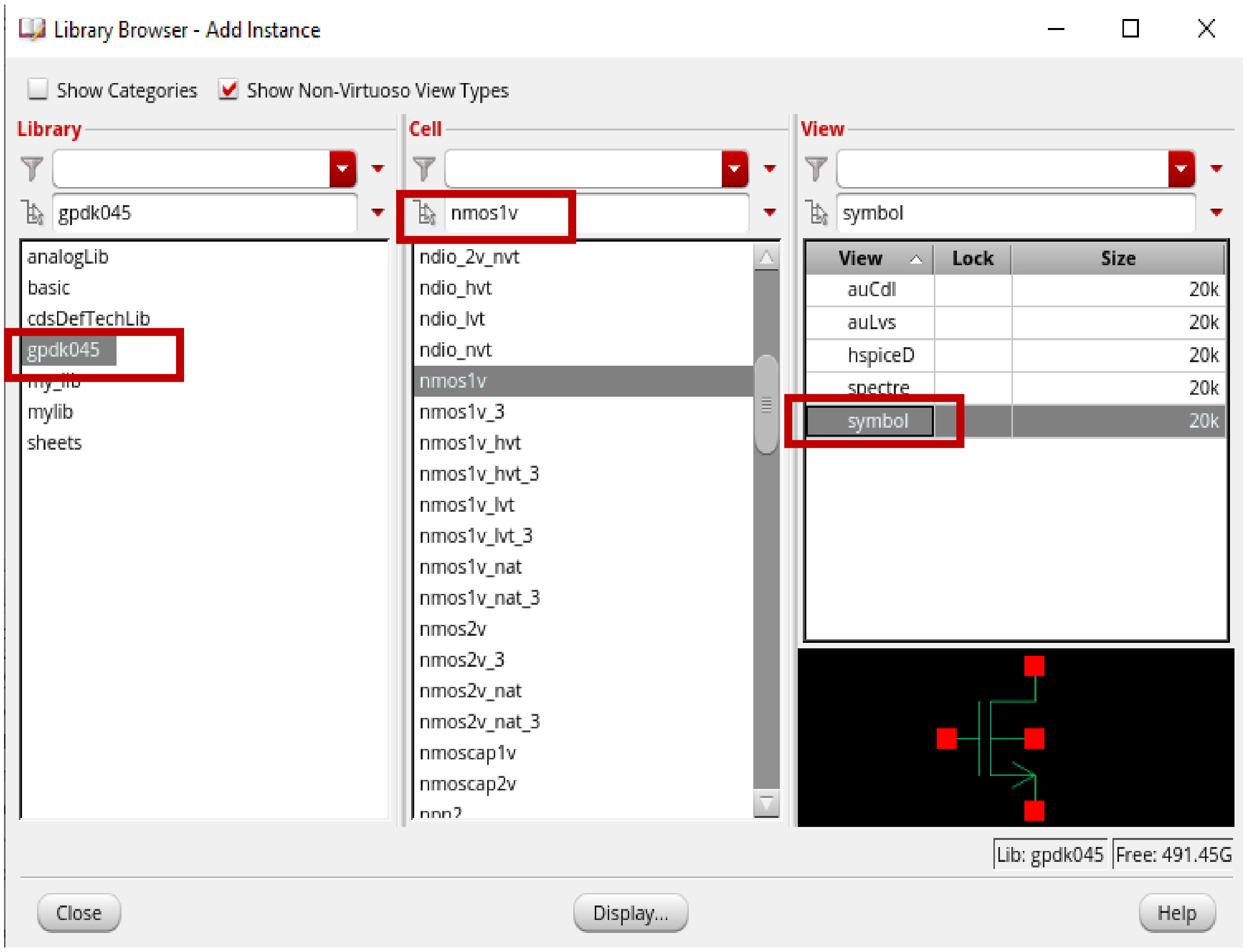
2. To create an instance, you can execute ***Create→ Instance*** in Virtuoso schematic editor window or simply use shortcut key "**i**". The following window will appear. Click **Browse** to select a library component.



View	symbol
Rows	1
Columns	1

Add Wire Stubs at: [✓] — all terminals [✓ registered terminals only]

3. Library Browser window will show up. Choose **Library:** *gpdk045*, **Cell:** *nmos1v*, **View:** *symbol*. (Note that while you are doing this, the '**Add Instance**' form is getting updated as well).



Library: gpdk045

analogLib
basic
cdsDefTechLib
gpdk045
my_lib
mylib
sheets

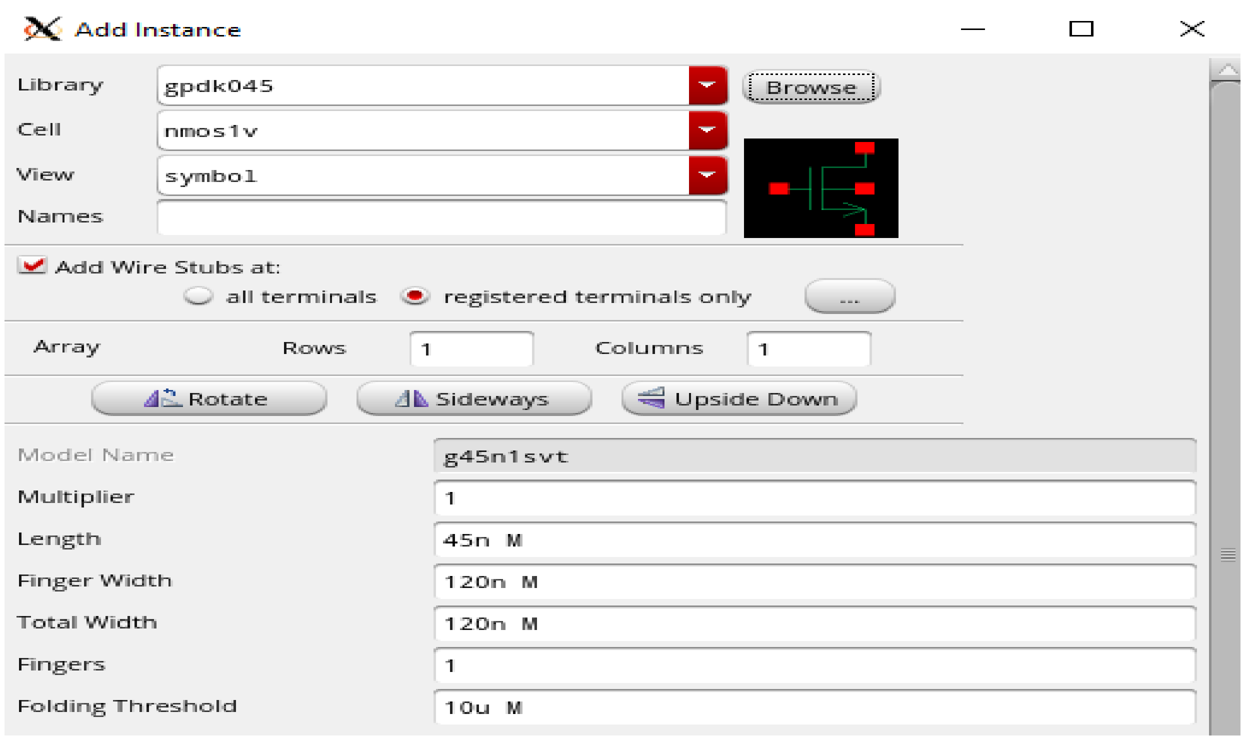
Cell: nmos1v

ndio_2v_nvt
ndio_hvt
ndio_lvt
ndio_nvt
nmos1v
nmos1v_3
nmos1v_hvt
nmos1v_hvt_3
nmos1v_lvt
nmos1v_lvt_3
nmos1v_nat
nmos1v_nat_3
nmos2v
nmos2v_3
nmos2v_nat
nmos2v_nat_3
nmoscap1v
nmoscap2v
npn2

View: symbol

View	Lock	Size
auCdl		20k
auLvs		20k
hspiceD		20k
spectre		20k
symbol		20k

Lib: gpdk045 Free: 491.45G


Add Instance

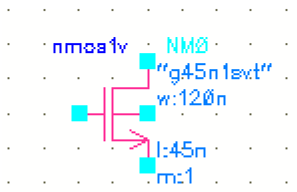
Library	gpdk045
Cell	nmos1v
View	symbol
Names	

[✓] Add Wire Stubs at: all terminals [● registered terminals only]

Array	Rows	1	Columns	1

Model Name	g45n1svt
Multiplier	1
Length	45n M
Finger Width	120n M
Total Width	120n M
Fingers	1
Folding Threshold	10u M

4. Make sure that the ***view*** name field in the form is set to ***symbol***. After you complete the form, move your cursor to the schematic window and click left button of mouse to place the component. After entering the components, click **Cancel** in the **Create Instance** form or press **Esc** keeping your cursor in the schematic window.



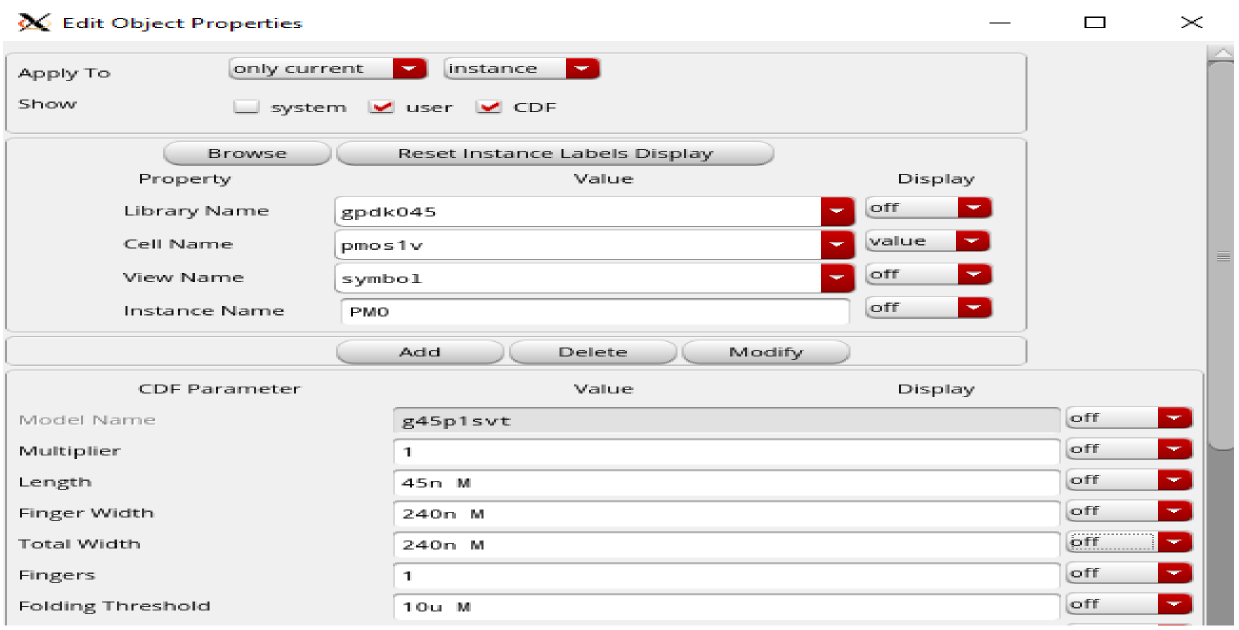
Similarly, add **pmos1v** cell.

If you place component in the wrong location, press '**m**' on keyboard, click once on the component to select it and move the mouse to move the component to your desired location.

5. Now we can adjust the sizes of the transistors by editing instance properties. Left click on the NMOS to select the component. Then, press "**q**" to modify its properties, or in schematic editor window, execute ***Edit*** **→** ***Properties*** **→** ***Object***.

You will update the Library Name, Cell Name, and the property values given in the table below as you place each component. The inverter design contains the following cells from the following libraries.

For example, while modifying the transistor width for PMOS, set **Total Width** to 240n, and then press '**Tab**' key and the **Finger Width** will be set to the same value. Click **OK**.


Edit Object Properties

Apply To: only current, instance

Show: [ ] system [✓] user [✓] CDF

Browse | Reset Instance Labels Display

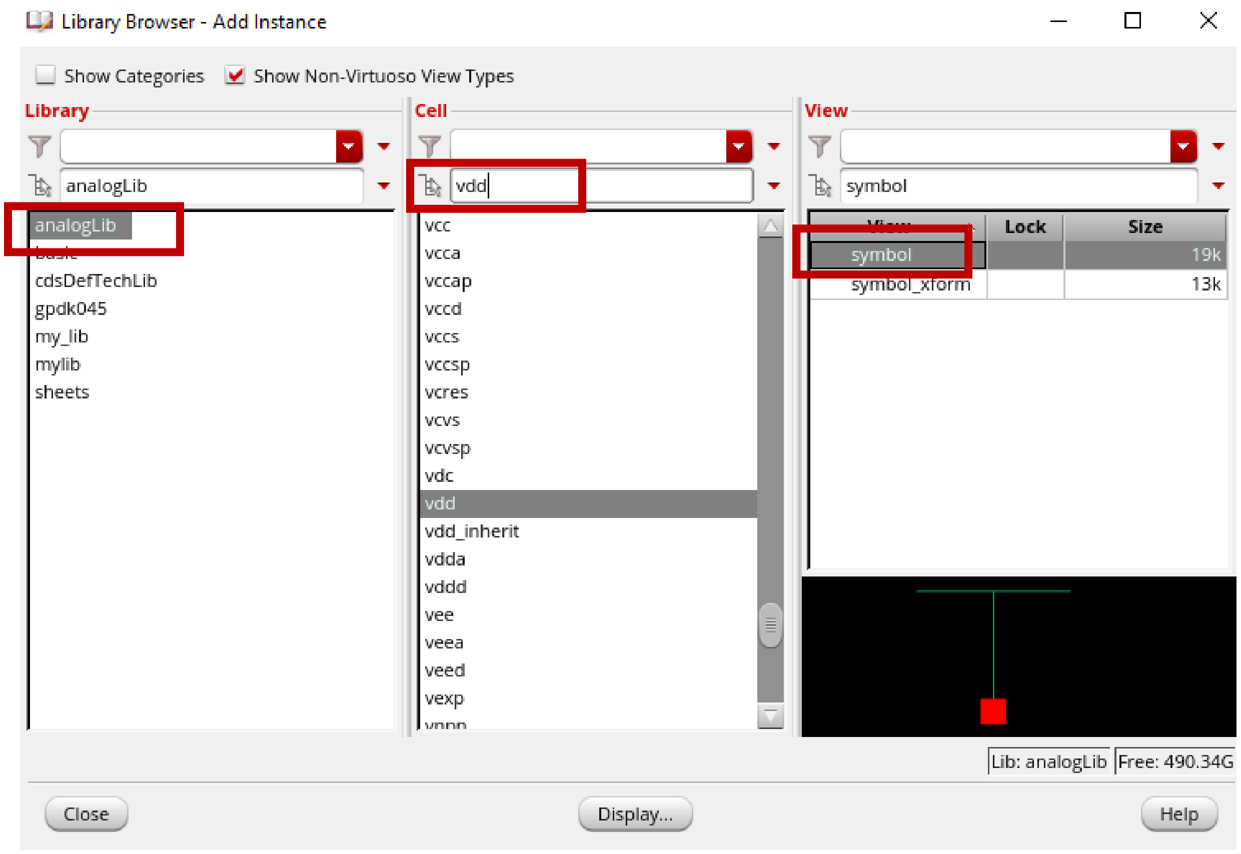
Property	Value	Display
Library Name	gpdk045	off
Cell Name	pmos1v	value
View Name	symbol	off
Instance Name	PM0	off

Add | Delete | Modify

CDF Parameter	Value	Display
Model Name	g45p1svt	off
Multiplier	1	off
Length	45n M	off
Finger Width	240n M	off
Total Width	240n M	off
Fingers	1	off
Folding Threshold	10u M	off

To deselect any object, press keyboard command "Ctrl+d".

Next, instantiate power nets (cell **vdd** and **gnd** from **analogLib** library).


Library Browser - Add Instance

[ ] Show Categories [✓] Show Non-Virtuoso View Types

Library: analogLib — analogLib, basic, cdsDefTechLib, gpdk045, my_lib, mylib, sheets

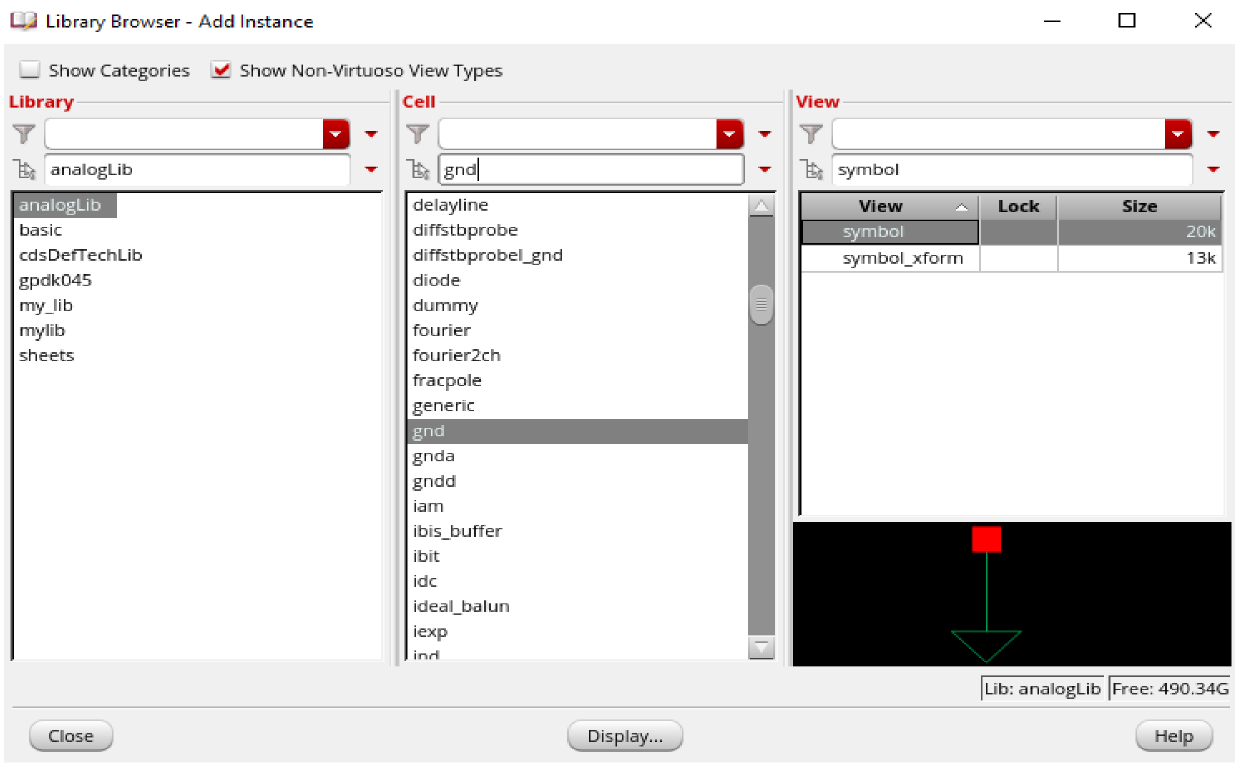
Cell: vdd — vcc, vcca, vccap, vccd, vccs, vccsp, vcres, vcvs, vcvsp, vdc, vdd, vdd_inherit, vdda, vddd, vee, veea, veed, vexp, vnnn

View: symbol

View	Lock	Size
symbol		19k
symbol_xform		13k

Lib: analogLib Free: 490.34G

Close | Display... | Help


Library Browser - Add Instance

Show Categories; ✓ Show Non-Virtuoso View Types

Library: analogLib
analogLib
basic
cdsDefTechLib
gpdk045
my_lib
mylib
sheets

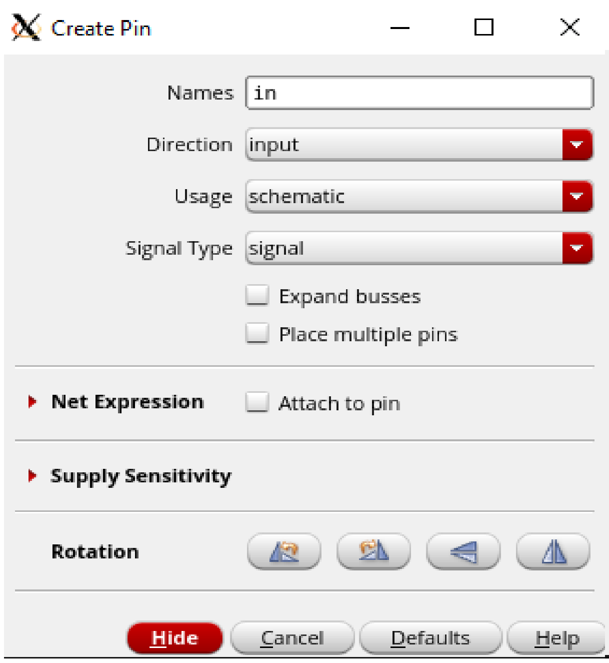
Cell: gnd
delayline
diffstbprobe
diffstbprobel_gnd
diode
dummy
fourier
fourier2ch
fracpole
generic
gnd
gnda
gndd
iam
ibis_buffer
ibit
idc
ideal_balun
iexp
ind

View: symbol

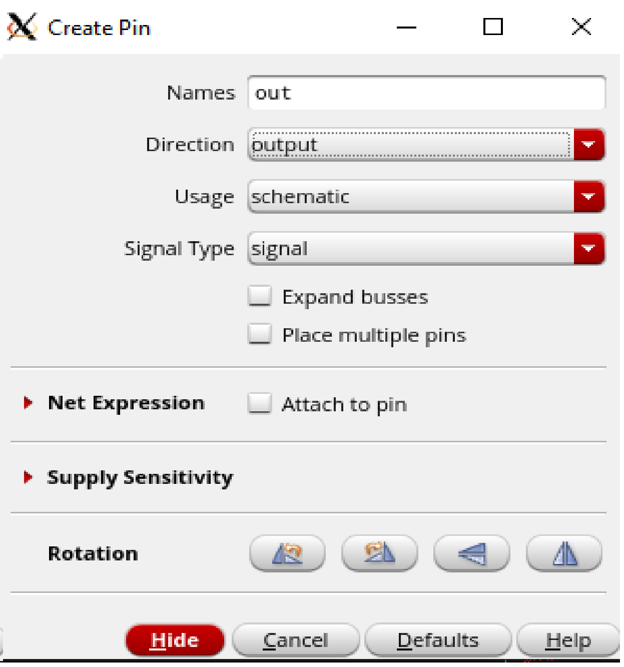
View	Lock	Size
symbol		20k
symbol_xform		13k

Lib: analogLib Free: 490.34G

6. Execute ***Create→ Pin*** or press '**p**' on keyboard. '**Add Pin**' form will appear. Enter the name of the pin and **Direction** of the pin. Add all the pins (**in**, **out**) to the schematic. For an inverter, gate input pin (e.g. in) is the input and output pin (e.g. out) at the common node between drains of NMOS and PMOS is output of the inverter. So, select **Direction** property as **input** for **in**, and **output** for **out**.


Create Pin

Names: in
Direction: input
Usage: schematic
Signal Type: signal
Expand busses
Place multiple pins
Net Expression — Attach to pin
Supply Sensitivity
Rotation

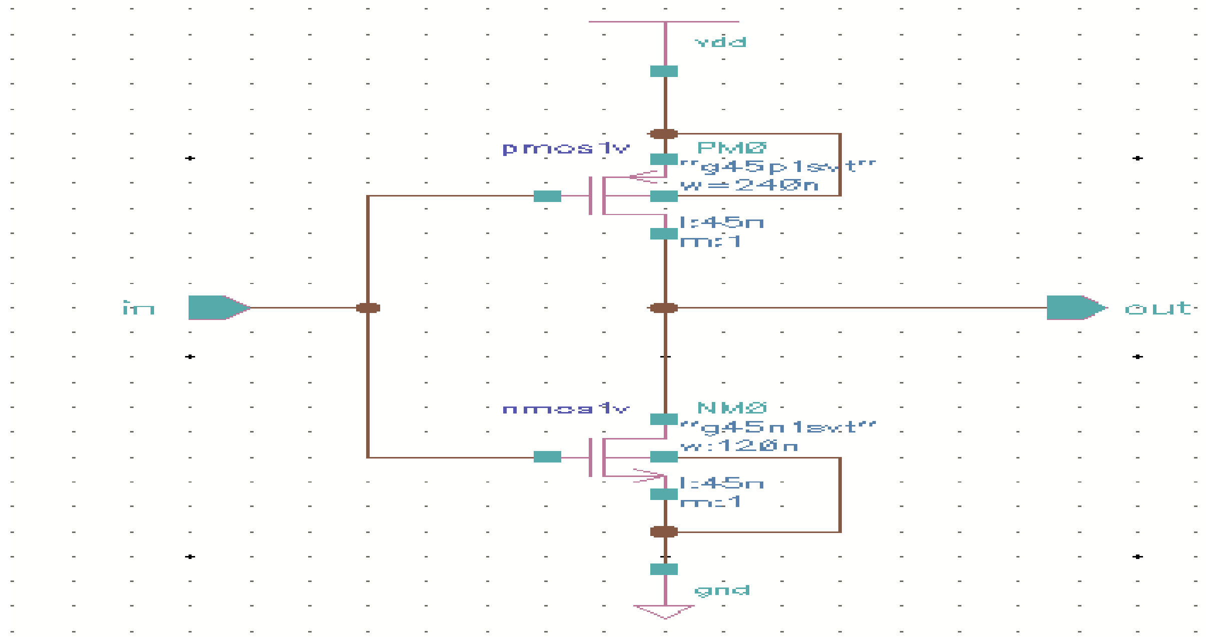

Create Pin

Names: out
Direction: output
Usage: schematic
Signal Type: signal
Expand busses
Place multiple pins
Net Expression — Attach to pin
Supply Sensitivity
Rotation

7. Use ***Add→ Wire*** menu or simply press ‘**w**’ key while staying on the schematic editor to enter wiring mode / **Esc** to exit. Click and release left button of mouse to start wire connections and click again at another point to draw wire connection.

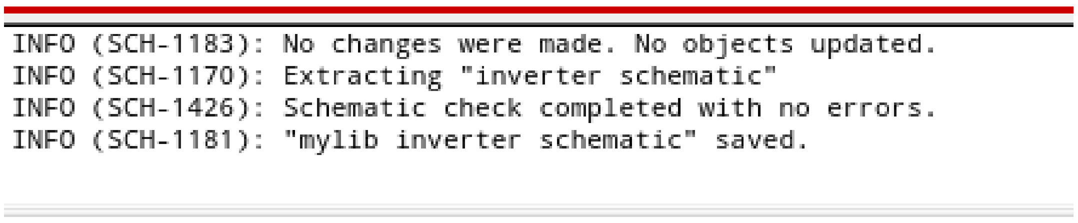
It is a good practice to periodically save your work by clicking on **Check and Save** button (the checkmark button just below the Tools menu). You can also save your work from the drop-down menu ***File→Save***.

The final schematic looks like the following one:



8. Click **Check and Save**.

9. Check **CIW** for errors or warnings. Some licence warnings may be ignored. If there are no error or design warning, you should see the following message:

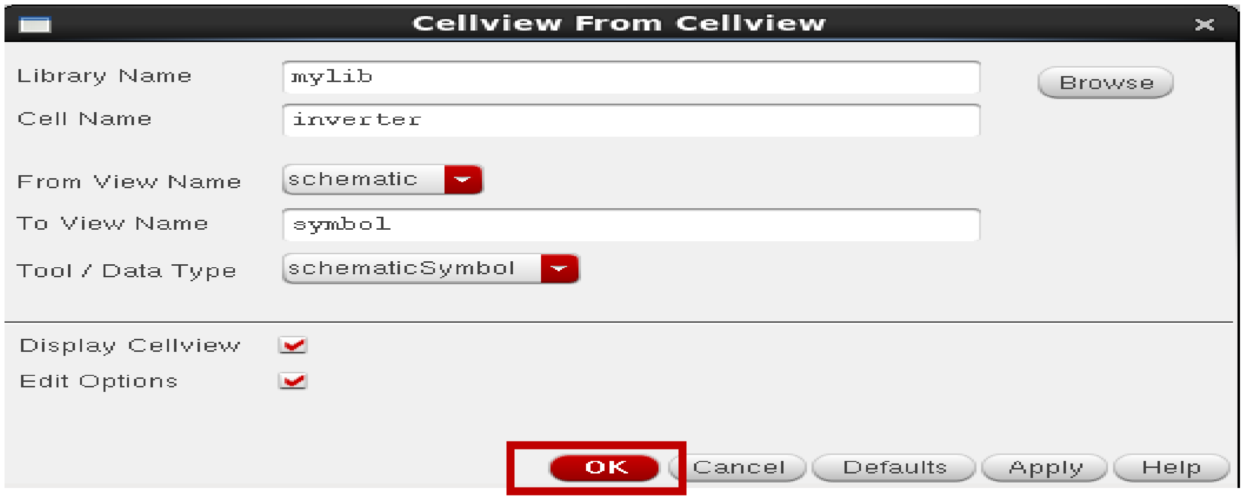


```
INFO (SCH-1183): No changes were made. No objects updated.
INFO (SCH-1170): Extracting "inverter schematic"
INFO (SCH-1426): Schematic check completed with no errors.
INFO (SCH-1181): "mylib inverter schematic" saved.
```

Symbol Creation

We will create a symbol for your inverter design so that we can use this symbol view for the schematic in a hierarchical design.

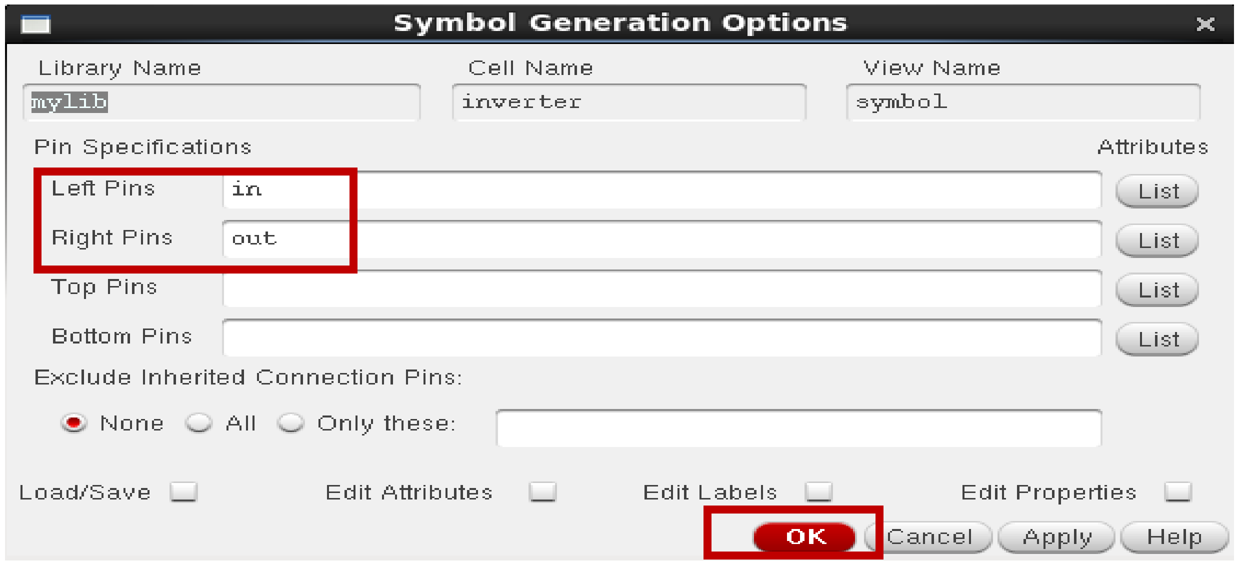
1. In the schematic editor window for ***inverter***, execute ***Create→Cellview→From Cellview***. ‘**Cellview from cellview**’ window appears. Click **OK**.



Cellview From Cellview

Library Name	mylib
Cell Name	inverter
From View Name	schematic
To View Name	symbol
Tool / Data Type	schematicSymbol
Display Cellview	✓
Edit Options	✓

2. In the ‘**Symbol Generation Options**’ window, you can choose the location of the pins.



Symbol Generation Options

Library Name	Cell Name	View Name
mylib	inverter	symbol

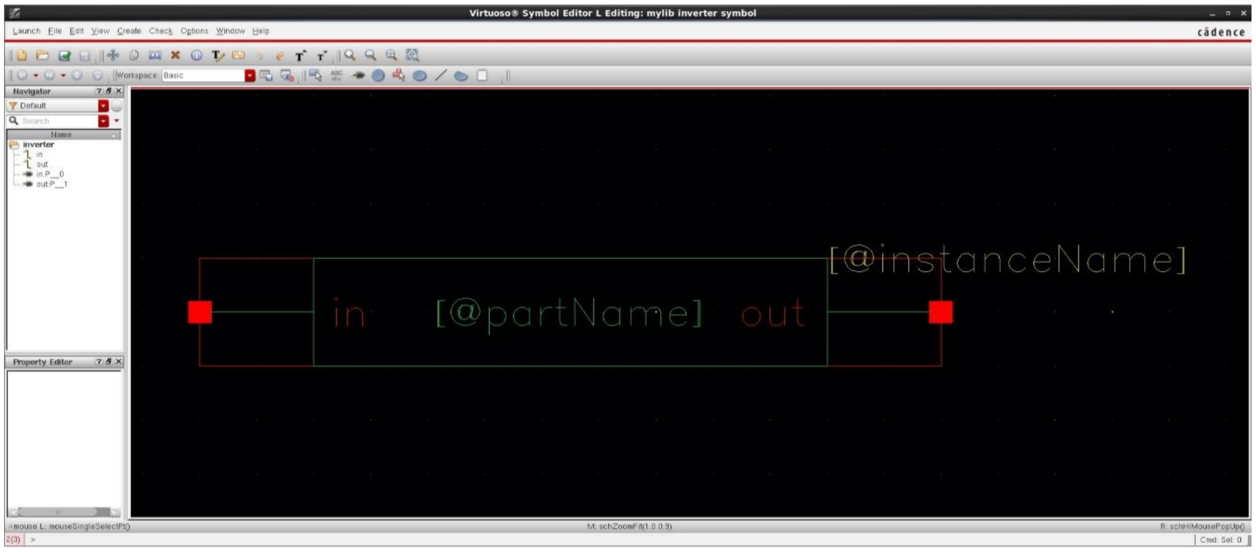
Pin Specifications

Left Pins	in
Right Pins	out
Top Pins	
Bottom Pins	

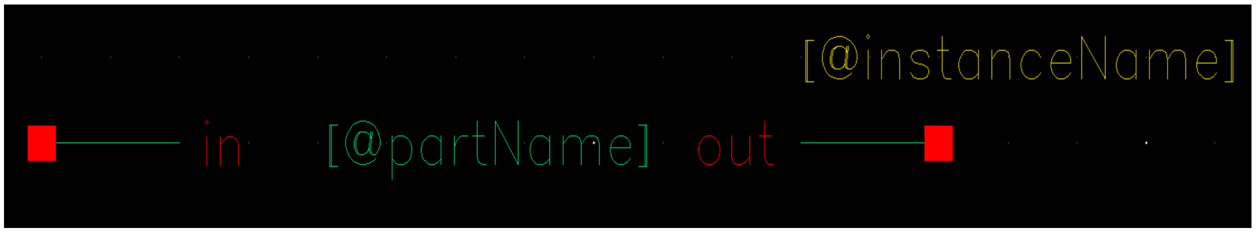
Exclude Inherited Connection Pins: [✓ None] All Only these:

Load/Save ☐ Edit Attributes ☐ Edit Labels ☐ Edit Properties ☐

3. Click **OK** on the ‘**Symbol Generation Options**’ window and the **Symbol Editor** window will open.



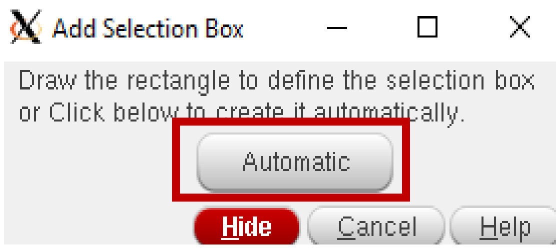
3. Click **Delete** icon in the symbol window, delete the outer red rectangle and green rectangle.



4. Execute ***Create →Shape→polygon***, and draw a shape similar to triangle. After creating the triangle, press **Esc** key.

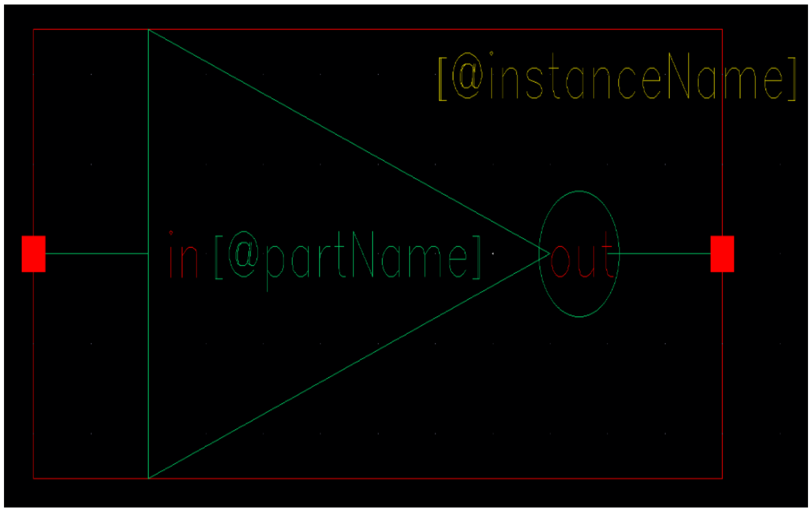
5. Execute ***Create→Shape→Circle*** to make a circle at the end of the triangle. You can move the pin names according to the location.

6. Execute ***Create→Selection Box***. In the '**Add Selection Box**' form, click '**Automatic**'. A new red selection box is automatically added.


Add Selection Box

Draw the rectangle to define the selection box or Click below to create it automatically.

7. After creating symbol, click on the save icon in the symbol editor window to save the symbol. In the symbol editor window, execute ***File→Check and Save***. Then close the symbol editor window.



Appendix: Cadence Virtuoso® Schematic Editor L Shortcuts

Shortcut key	Tasks performed
w	Add a wire
i	Add an instance
p	Add a pin
l	Add label to a wire
e	Display options
q	Select an object and press q to open 'Edit Object Property' dialogue box
[	Zoom out
]	Zoom in
c	Copy
m	Move
u	Undo
Shift+u	Redo
f	Fit the entire schematic in the window

Student Task

7. Implement 3 input NOR and NAND gate using CMOS logic family

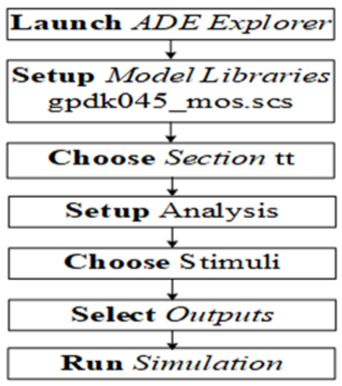
Lab-5: Measuring Different Performance Parameters of a CMOS Inverter

Objectives:

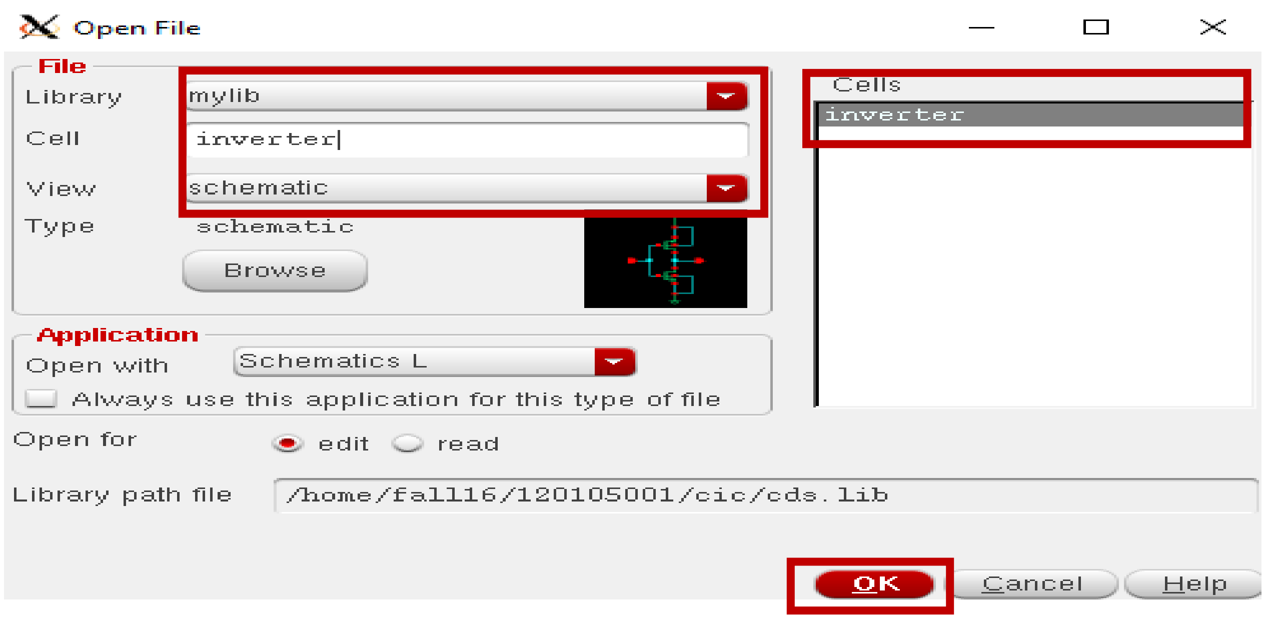
- To verify the functionality through transient simulation.
- To measure power dissipation, propagation delay, rise time and fall time of logic gates.
- To learn about process corners and their effects on delay and power dissipation

Functional Verification

The following flowchart shows the steps to be executed to simulate a design using ADE Explorer:



1. To simulate the schematic of inverter first open the schematic of inverter, execute ***File→ Open*** in **CIW**. In the '**Open File**' window, select the inverter schematic from the list. Click **OK**.


Open File

File
Library mylib
Cell inverter
View schematic
Type schematic

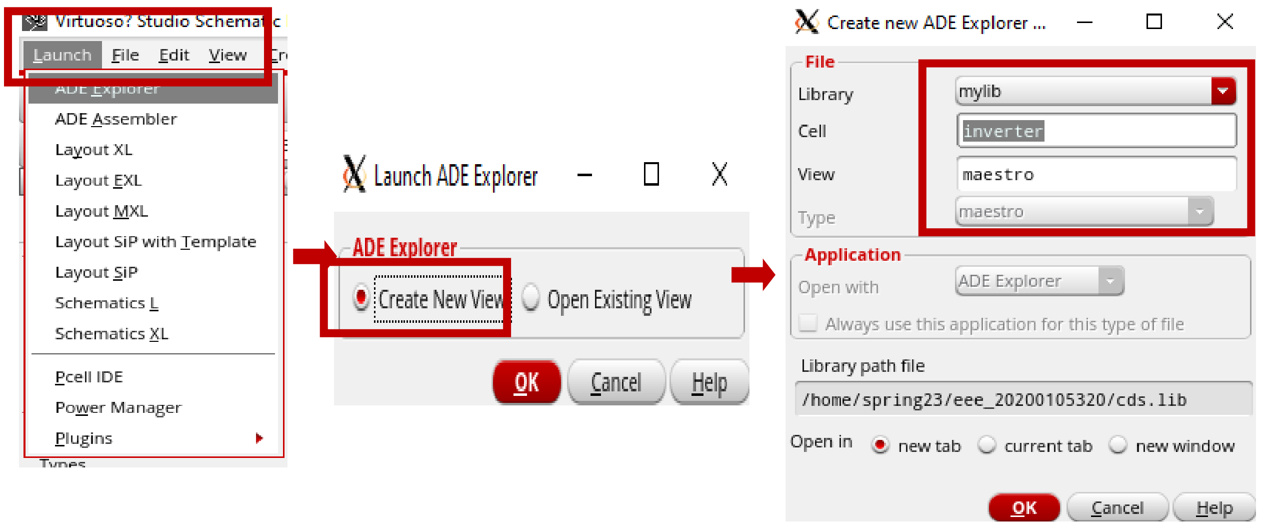
Application
Open with Schematics L
Always use this application for this type of file

Open for [✓ edit] read

Library path file /home/fall16/120105001/cic/cds.lib

Cells
inverter

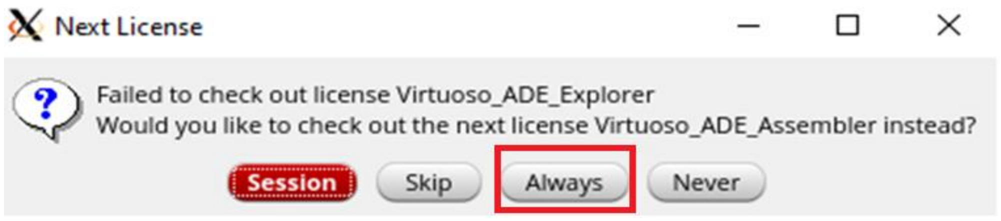
2. In the Schematic editor window, execute ***Launch→ ADE Explorer.*** The following window may appear.



Create new ADE Explorer:

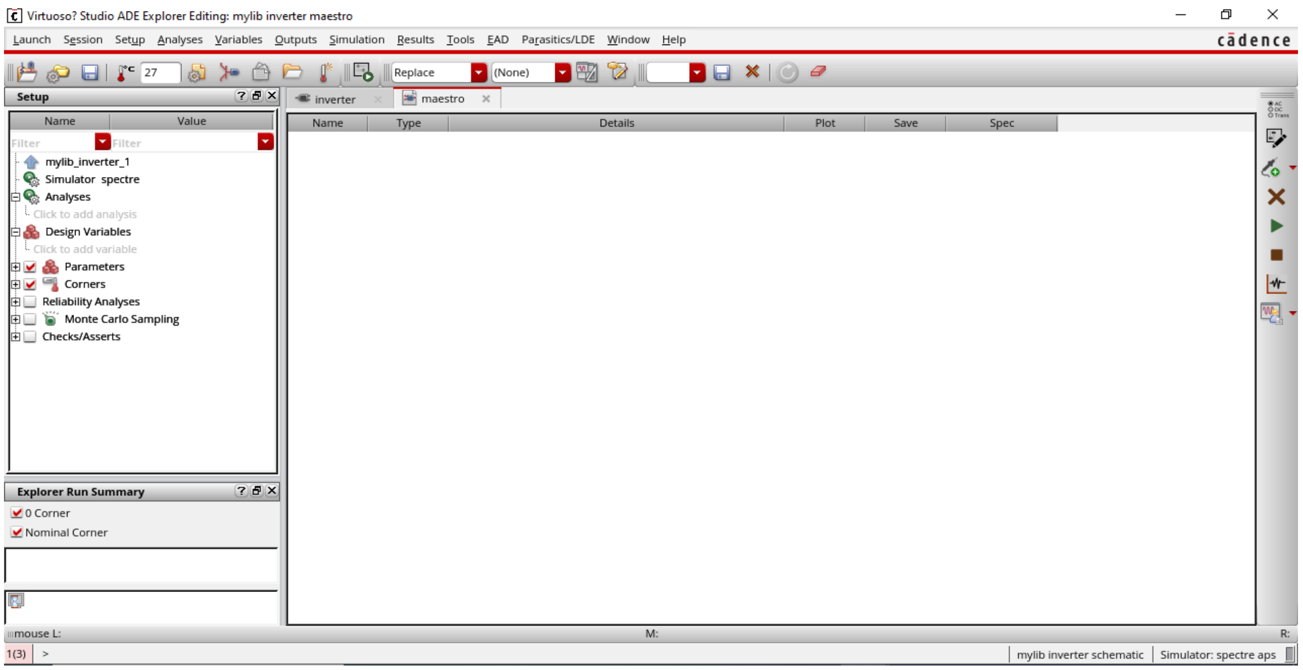
Library: mylib
Cell: inverter
View: maestro
Type: maestro
Open with: ADE Explorer
Always use this application for this type of file
Library path file: /home/spring23/eee_20200105320/cds.lib
Open in [✓ new tab] current tab new window

The following window may appear. Click **Always**.

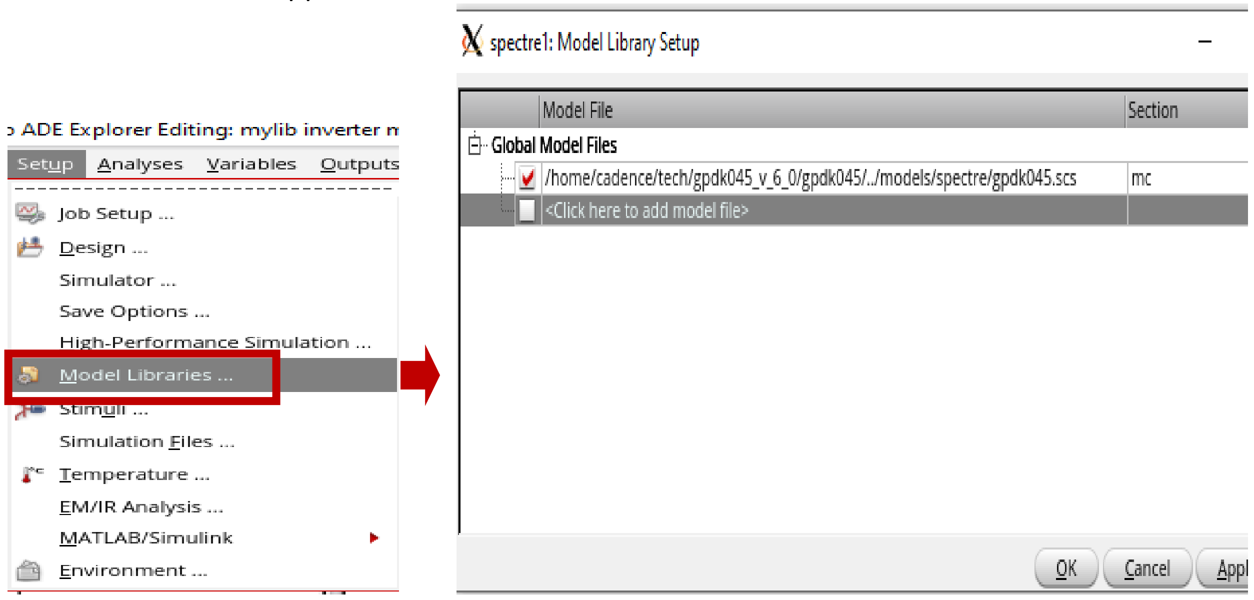


Failed to check out license Virtuoso_ADE_Explorer
Would you like to check out the next license Virtuoso_ADE_Assembler instead?

Analog Design Environment (ADE) Explorer window will appear.

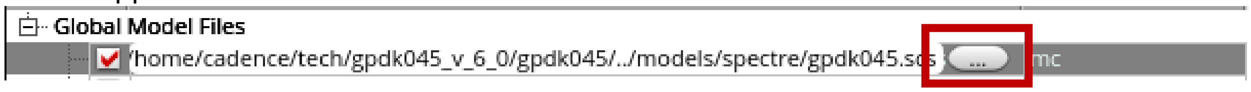


6. Set up the model libraries by executing ***Setup→Model Libraries***. '**Model Library Setup**' Window will appear:



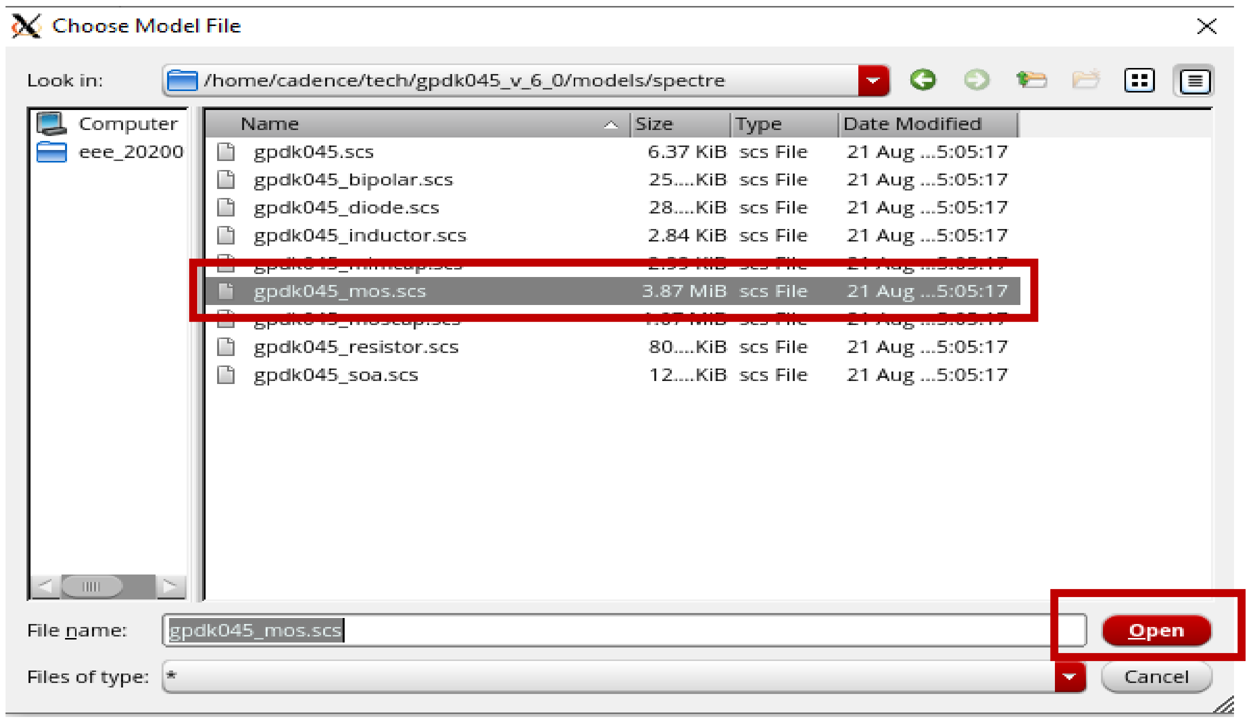
/home/cadence/tech/gpdk045_v_6_0/gpdk045/../models/spectre/gpdk045.scs mc

7. Click twice on the file name given under **Global Model Files**. An ash coloured button will appear.



/home/cadence/tech/gpdk045_v_6_0/gpdk045/../models/spectre/gpdk045.scs mc

Click on the button. '**Choose Model File**' window will appear.



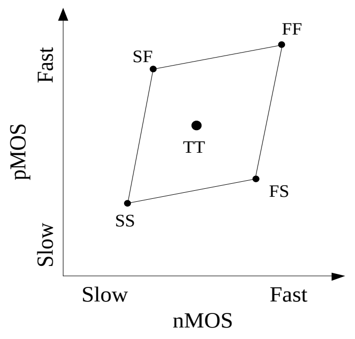
Look in: /home/cadence/tech/gpdk045_v_6_0/models/spectre

Name	Size	Type	Date Modified
gpdk045.scs	6.37 KiB	scs File	21 Aug ...5:05:17
gpdk045_bipolar.scs	25....KiB	scs File	21 Aug ...5:05:17
gpdk045_diode.scs	28....KiB	scs File	21 Aug ...5:05:17
gpdk045_inductor.scs	2.84 KiB	scs File	21 Aug ...5:05:17
gpdk045_mimcap.scs	2.33 KiB	scs File	21 Aug ...5:05:17
gpdk045_mos.scs	3.87 MiB	scs File	21 Aug ...5:05:17
gpdk045_moscap.scs	1.07 MiB	scs File	21 Aug ...5:05:17
gpdk045_resistor.scs	80....KiB	scs File	21 Aug ...5:05:17
gpdk045_soa.scs	12....KiB	scs File	21 Aug ...5:05:17

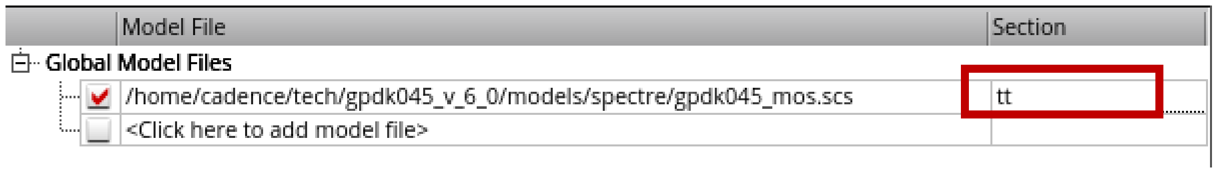
File name: gpdk045_mos.scs
Files of type: *

8. Select **gpdk045_mos.scs** from the list. Click **Open**.

In this model file, there are models to simulate various corners like fast-fast (FF), fast-slow (FS), typical-typical (TT) etc. These are called process corners, depending on the speed of MOS transistors (NMOS and PMOS). Refer to the following figure for the definition of process corners:

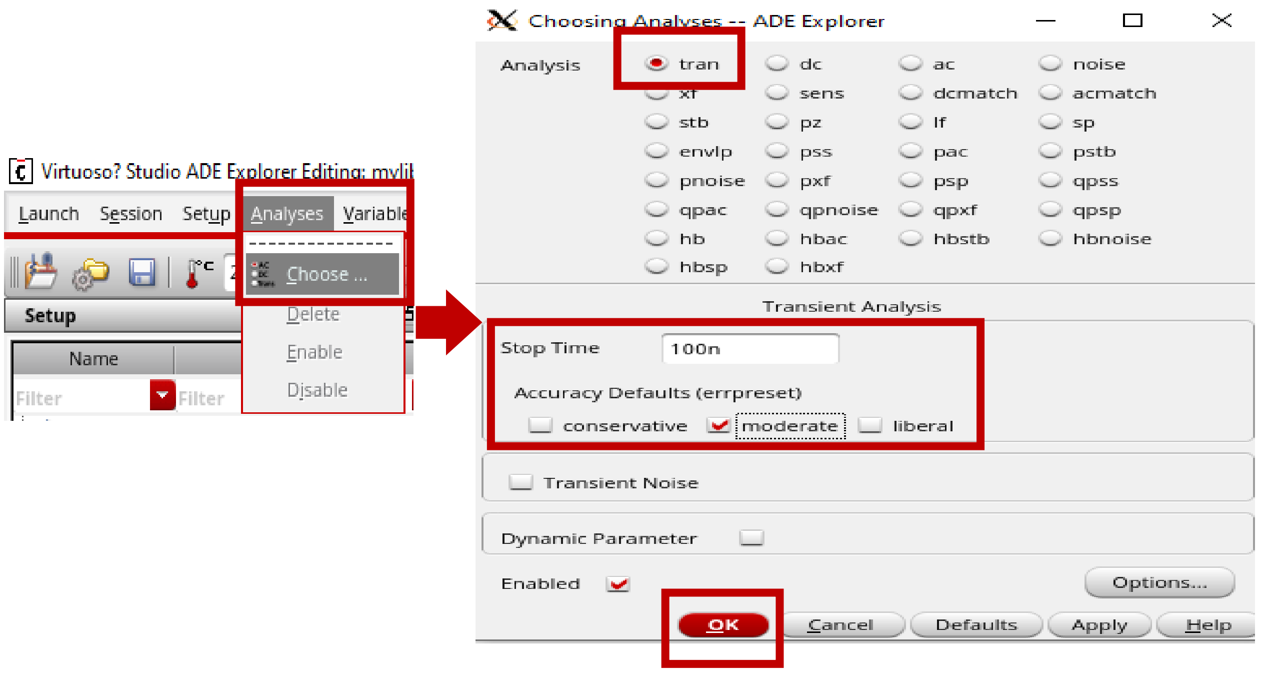


We will choose the section typical from the **Section** scroll bar and select the section '**tt**'. These will enable us to use the TT models of the 1.2 V MOS transistors. Only one **Global Model File** will be defined. Uncheck or delete any other model files that appear. Click **OK**.



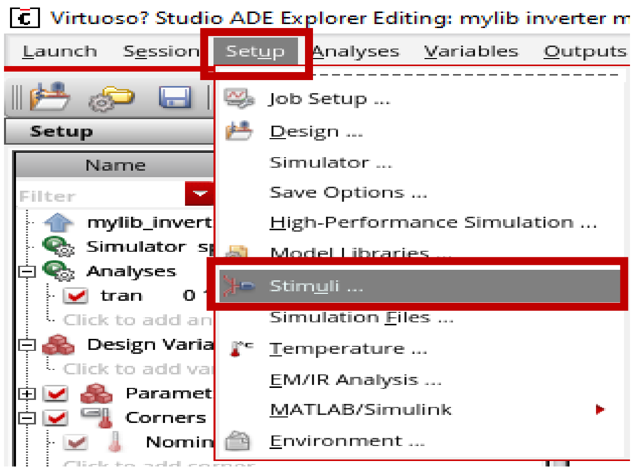
Model File	Section
Global Model Files	
✓ /home/cadence/tech/gpdk045_v_6_0/models/spectre/gpdk045_mos.scs	tt
<Click here to add model file>	

9. Now choose the analysis to be done from ***Analyses→Choose***. Select transient (**tran**) analysis to be done. Provide a reasonable value for 'stop time' to observe few periods of signals. (e.g. **Analysis:** *tran*, **Stop Time:** 100n, **Accuracy Defaults:** *moderate*). Click **OK**.

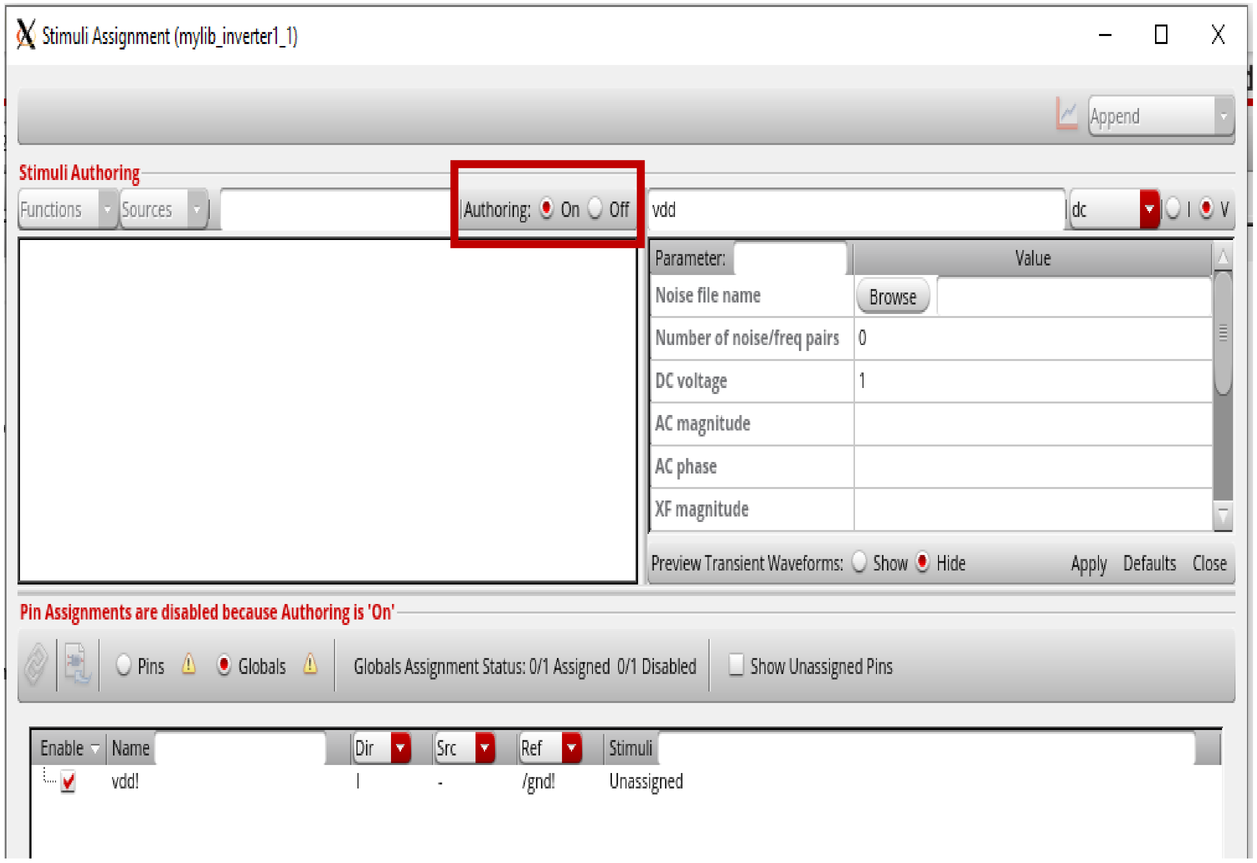


Stop Time: 100n
Accuracy Defaults (errpreset): conservative [✓ moderate] liberal

8. Now execute ***Setup*→ *Stimuli*** to assign signals to pins of the inverter.



8. In '**Stimuli Assignment**' window, select **Globals**. Now you can see global power net **vdd!**. Click on **Authoring → ON**.


Stimuli Assignment (mylib_inverter1_1)
Append
Stimuli Authoring
Functions
Sources
Apply
Defaults
Close
Pin Assignments are disabled because Authoring is 'On'
Show Unassigned Pins

Authoring: [✓ On] Off

vdd | dc

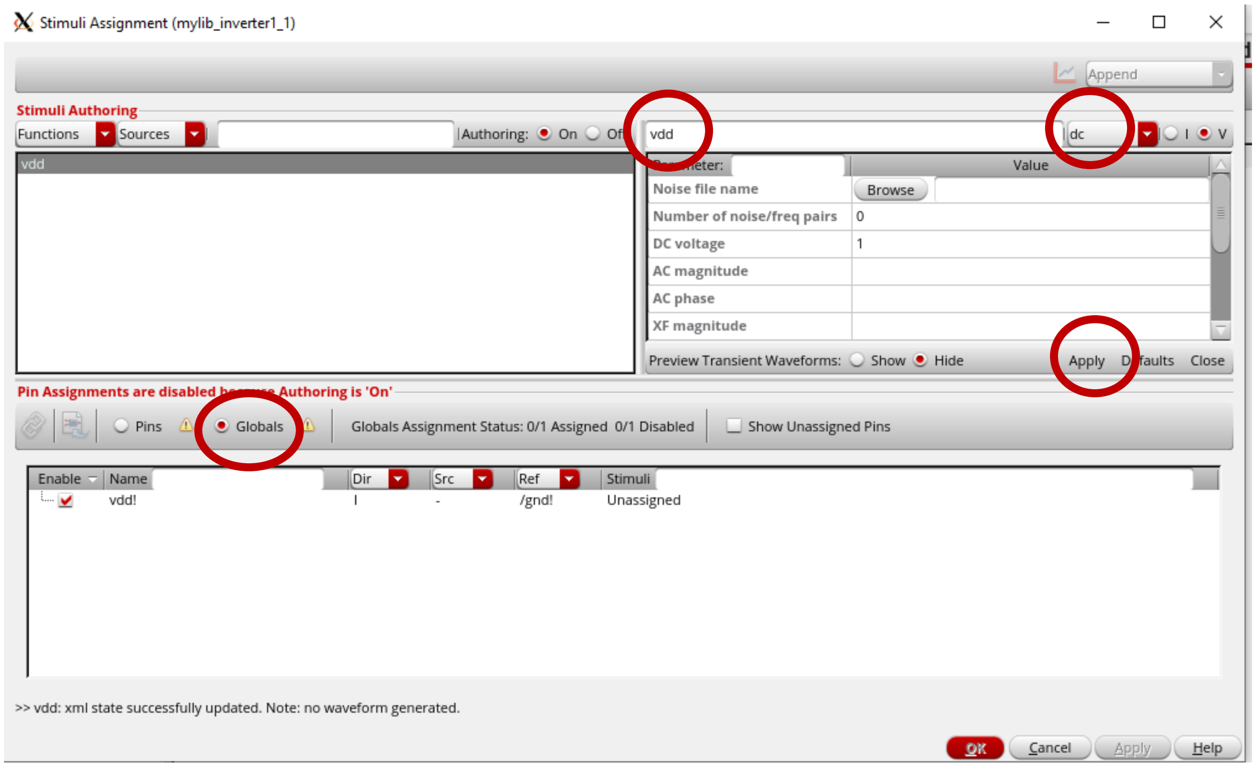
Parameter:	Value
Noise file name	Browse
Number of noise/freq pairs	0
DC voltage	1
AC magnitude	
AC phase	
XF magnitude	

Preview Transient Waveforms: Show [✓ Hide]

Pins [✓ Globals] Globals Assignment Status: 0/1 Assigned 0/1 Disabled

Enable	Name	Dir	Src	Ref	Stimuli
✓	vdd!	I	-	/gnd!	Unassigned

Write **vdd** in the box and select **dc** under the drop down menu. Put a value of 1 on the **DC voltage** box. The filled up form for '**vdd!**' will look like the one below. Click **Apply** (clicking **OK** will close the window and it will have to be reopened to setup inputs).


Stimuli Assignment (mylib_inverter1_1)
Append
Stimuli Authoring
Functions
Sources
Authoring: On Off
Preview Transient Waveforms: Show Hide
Apply
Defaults
Close
Pin Assignments are disabled because Authoring is 'On'
Pins
Globals
Show Unassigned Pins
OK
Cancel
Apply
Help

Authoring: [✓ On] Off

vdd — dc — [✓ V]

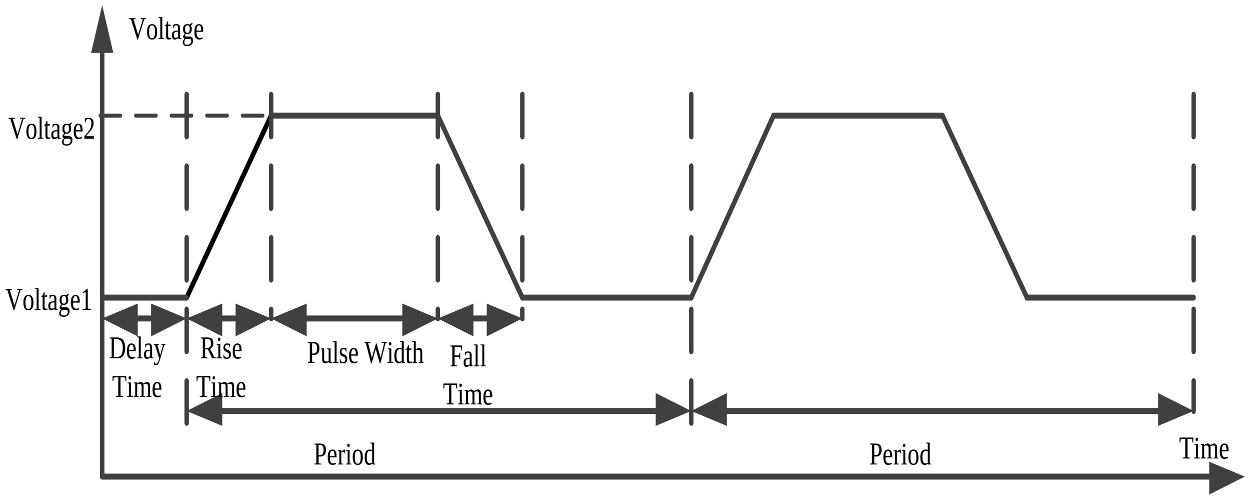
Parameter:	Value
Noise file name	Browse
Number of noise/freq pairs	0
DC voltage	1
AC magnitude	
AC phase	
XF magnitude	

Pins [✓ Globals] Globals Assignment Status: 0/1 Assigned 0/1 Disabled

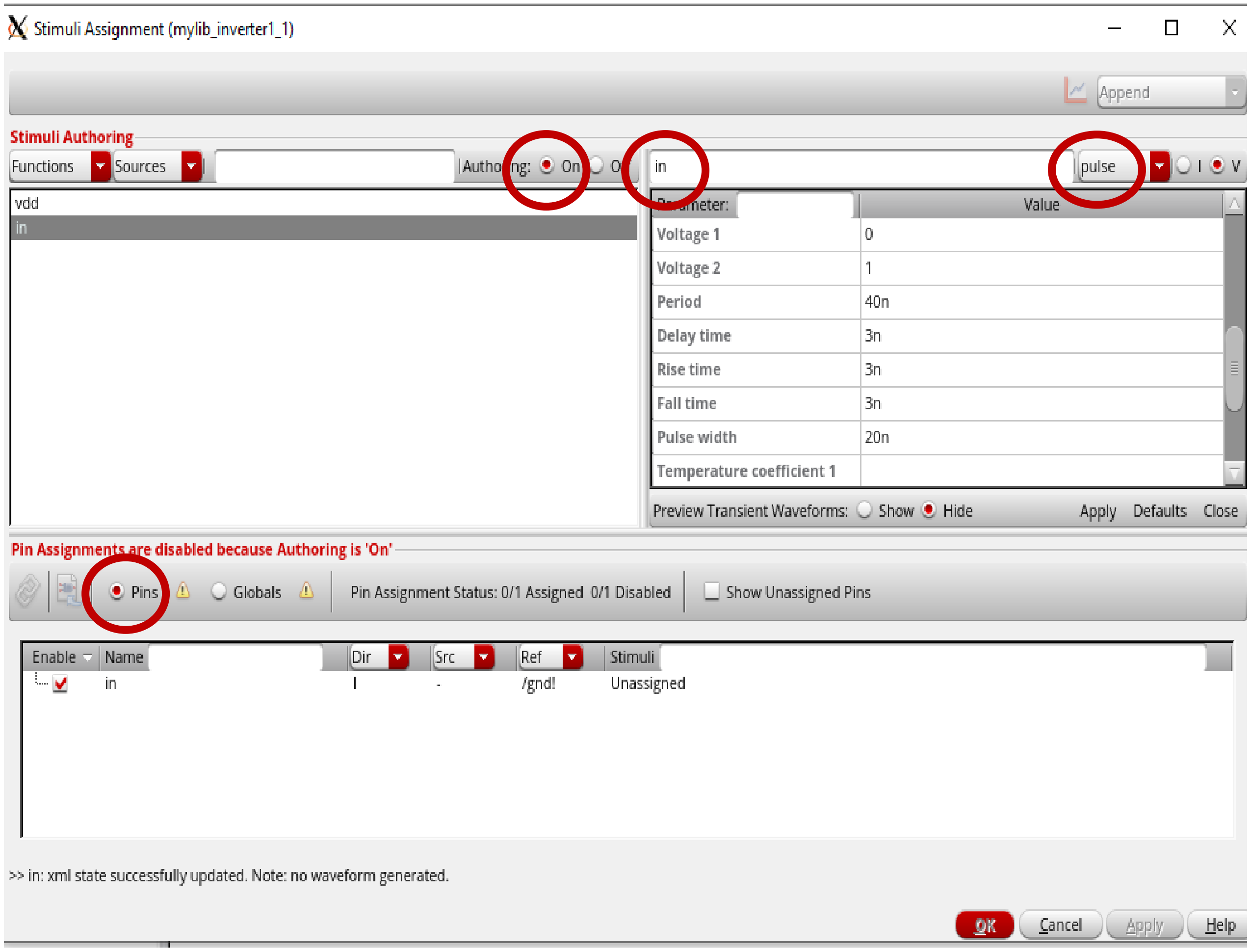
Enable	Name	Dir	Src	Ref	Stimuli
✓	vdd!	I	-	/gnd!	Unassigned

>> vdd: xml state successfully updated. Note: no waveform generated.

9. Select **Inputs**. For input pin '**in**', we have to set a pulse waveform. The following figure shows the definition of pulse parameters:



For setting signal to input pin '**in**', select **Pins** in **Stimuli Assignment** window. Click on **Authoring → ON**, write **in** in the box and select '***pulse***' under the drop down menu. Parameters for pulse source will be as follows: **Voltage1** = 0V, **Voltage2** = 1V, **Period** = 40n, **Delay time** = 3n, **Rise time** = 3n, **Fall time** = 3n, **Pulse width** = 20n. Click **Apply** and then click **OK**. (Delay, Rise time and Fall time can also be set at ps ranges for sharp transitions).

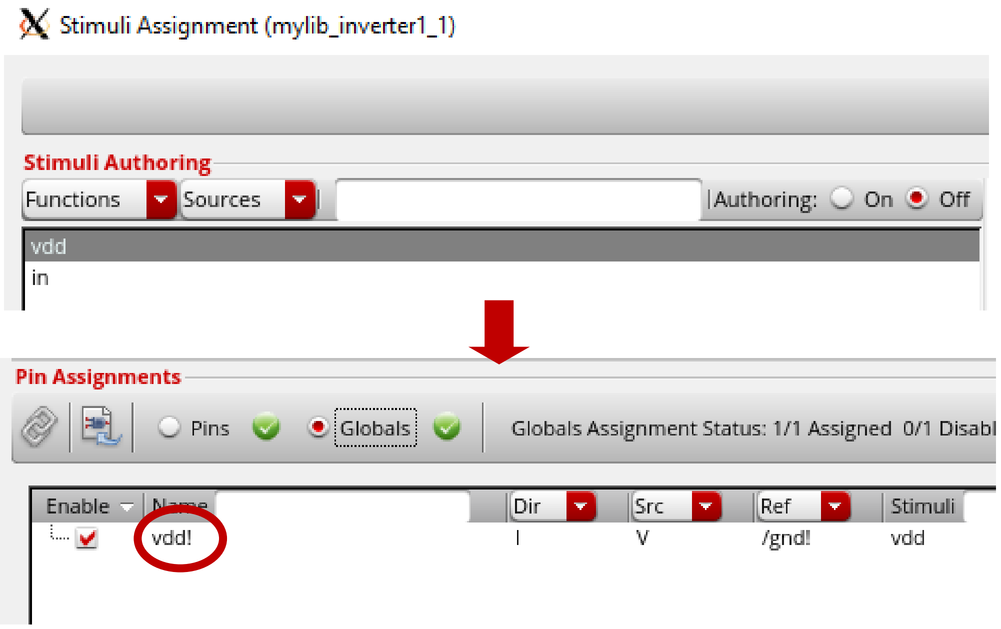

Stimuli Assignment (mylib_inverter1_1)
Append
Stimuli Authoring
Functions
Sources
Authoring: On Off
in
pulse
vdd
in
Preview Transient Waveforms: Show Hide
Apply
Defaults
Close
Pin Assignments are disabled because Authoring is 'On'
Pins
Globals
Pin Assignment Status: 0/1 Assigned 0/1 Disabled
Show Unassigned Pins
OK
Cancel
Apply
Help

Parameter	Value
Voltage 1	0
Voltage 2	1
Period	40n
Delay time	3n
Rise time	3n
Fall time	3n
Pulse width	20n
Temperature coefficient 1	

Enable	Name	Dir	Src	Ref	Stimuli
✓	in	I	-	/gnd!	Unassigned

```
>> in: xml state successfully updated. Note: no waveform generated.
```

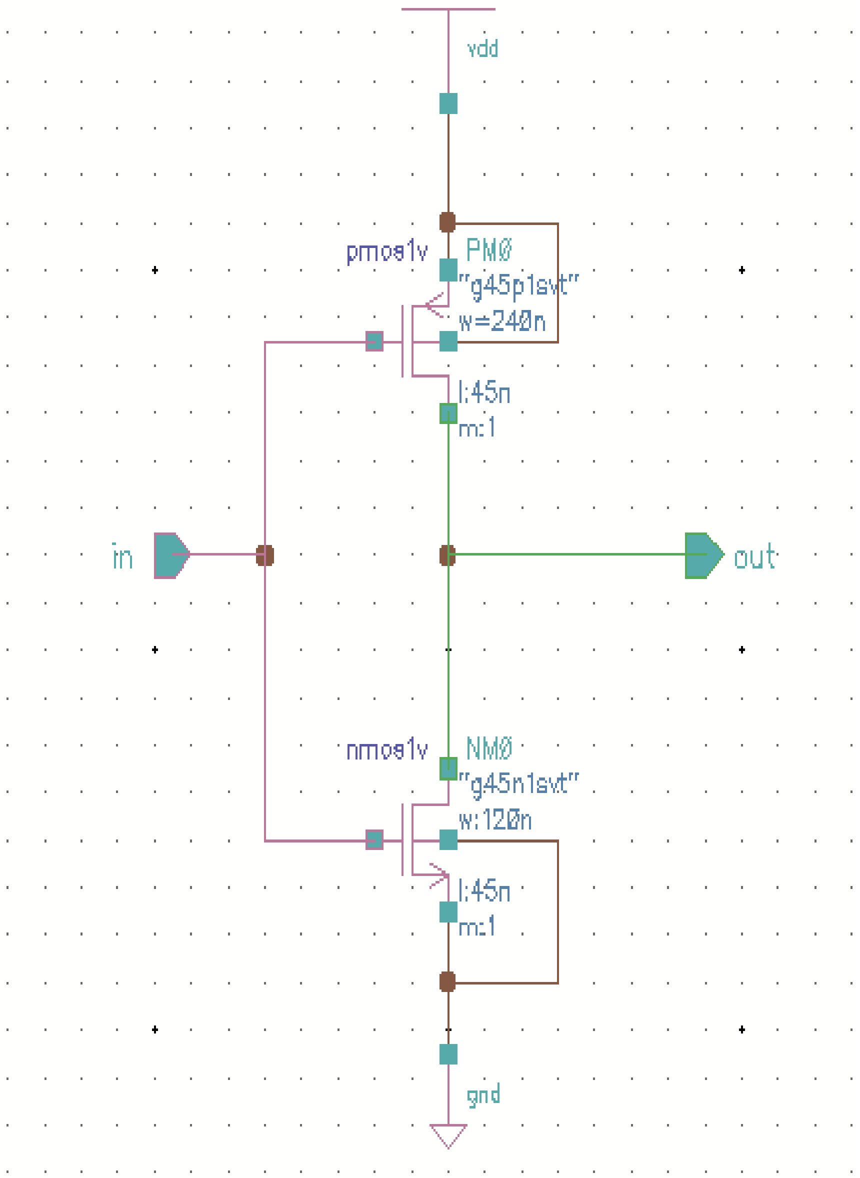
10. Now Click on **Authoring → OFF.** Then select the pin **vdd** in the **Stimuli Authoring window**. Go to **Pin Assignments →** ***Right Click on vdd!→ Assign Stimuli to selected Pin.***


Stimuli Assignment (mylib_inverter1_1)
Stimuli Authoring
Functions
Sources
Authoring: On Off
vdd
in
Pin Assignments
Pins
Globals
Globals Assignment Status: 1/1 Assigned 0/1 Disabl

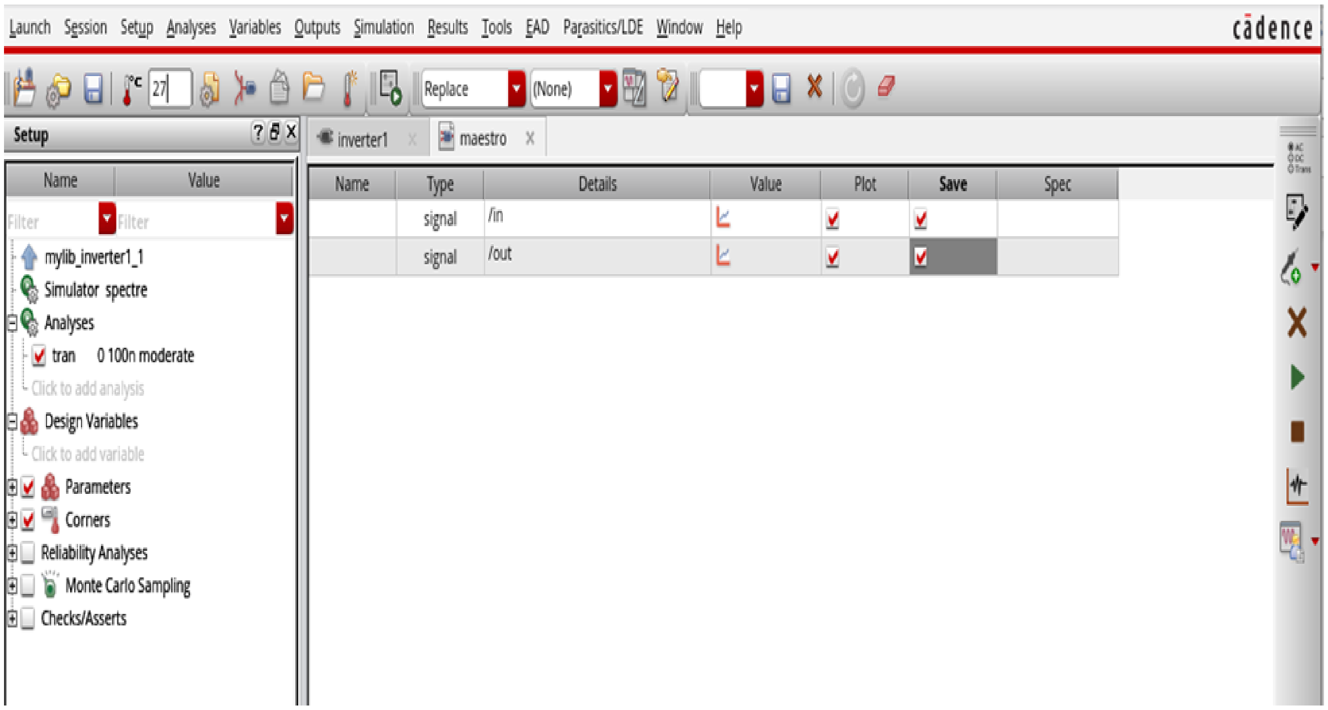
Enable	Name	Dir	Src	Ref	Stimuli
✓	vdd!	I	V	/gnd!	vdd

Do the same for other pins that has been defined previously.

10. Select the output to be plotted by executing ***Outputs→To be plotted→Select on Design*** in the ADE window. Schematic editor window will pop up, select '**out**' and '**in**' by clicking on the pins/terminals or selecting from the list on the left hand side as shown in the figure below. When you select them, you will see colours being assigned to these pins.



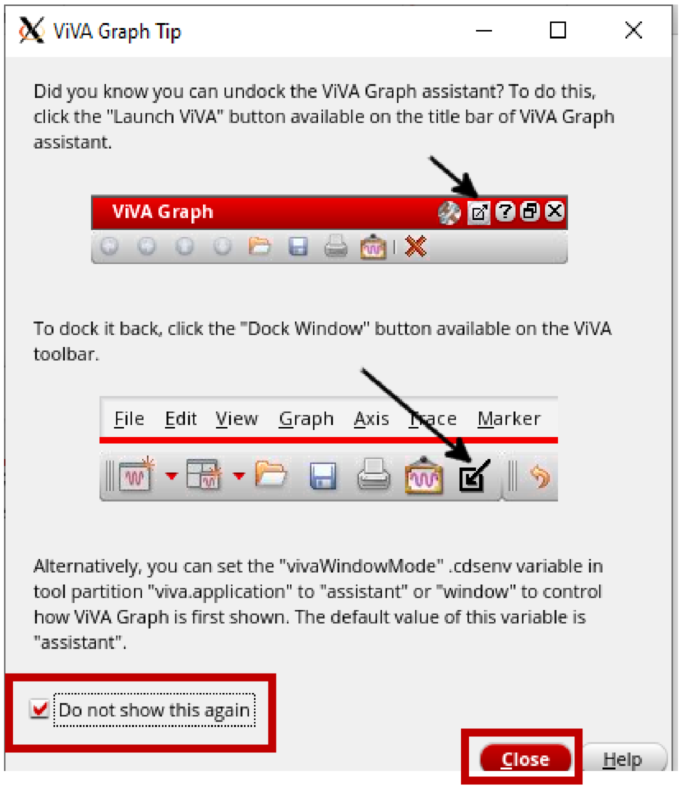
11. Your **Analog Design Environment** window should now look like the following:


Launch Session Setup Analyses Variables Outputs Simulation Results Tools EAD Parasitics/LDE Window Help
cādence
Replace
(None)
Setup
inverter1
maestro

Name	Value
mylib_inverter1_1	
Simulator spectre	
Analyses	
tran	0 100n moderate
Click to add analysis	
Design Variables	
Click to add variable	
Parameters	
Corners	
Reliability Analyses	
Monte Carlo Sampling	
Checks/Asserts	

Name	Type	Details	Value	Plot	Save	Spec
	signal	/in		✓	✓	
	signal	/out		✓	✓	

12. Now run the simulation by executing ***Simulation→Netlist and Run*** in the ADE window. The following window may appear. Click on **Do not show again→ Close.**


ViVA Graph Tip

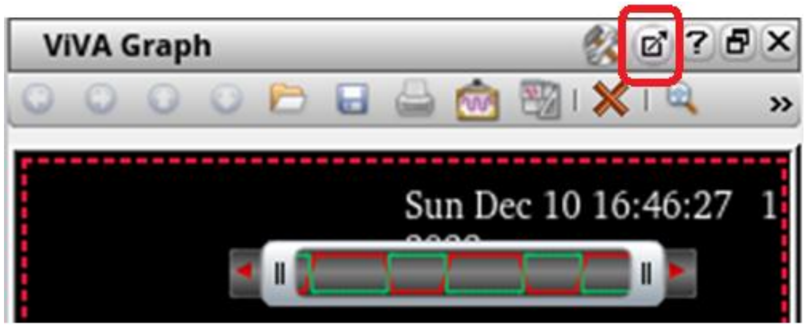
Did you know you can undock the ViVA Graph assistant? To do this, click the "Launch ViVA" button available on the title bar of ViVA Graph assistant.

To dock it back, click the "Dock Window" button available on the ViVA toolbar.

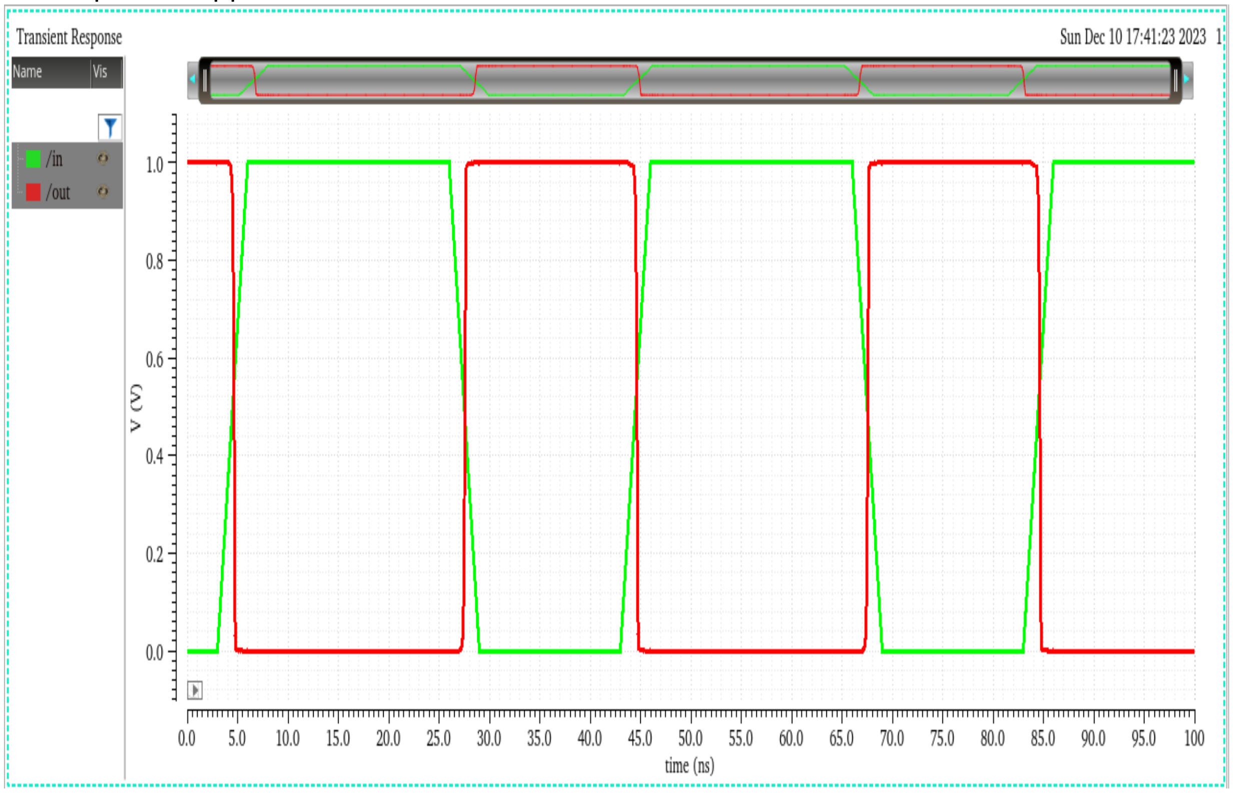
Alternatively, you can set the "vivaWindowMode" .cdsenv variable in tool partition "viva.application" to "assistant" or "window" to control how ViVA Graph is first shown. The default value of this variable is "assistant".

[✓] Do not show this again

The simulation will run and the VIVA Graph will appear in window as shown below. Click on the marked icon shown in figure.

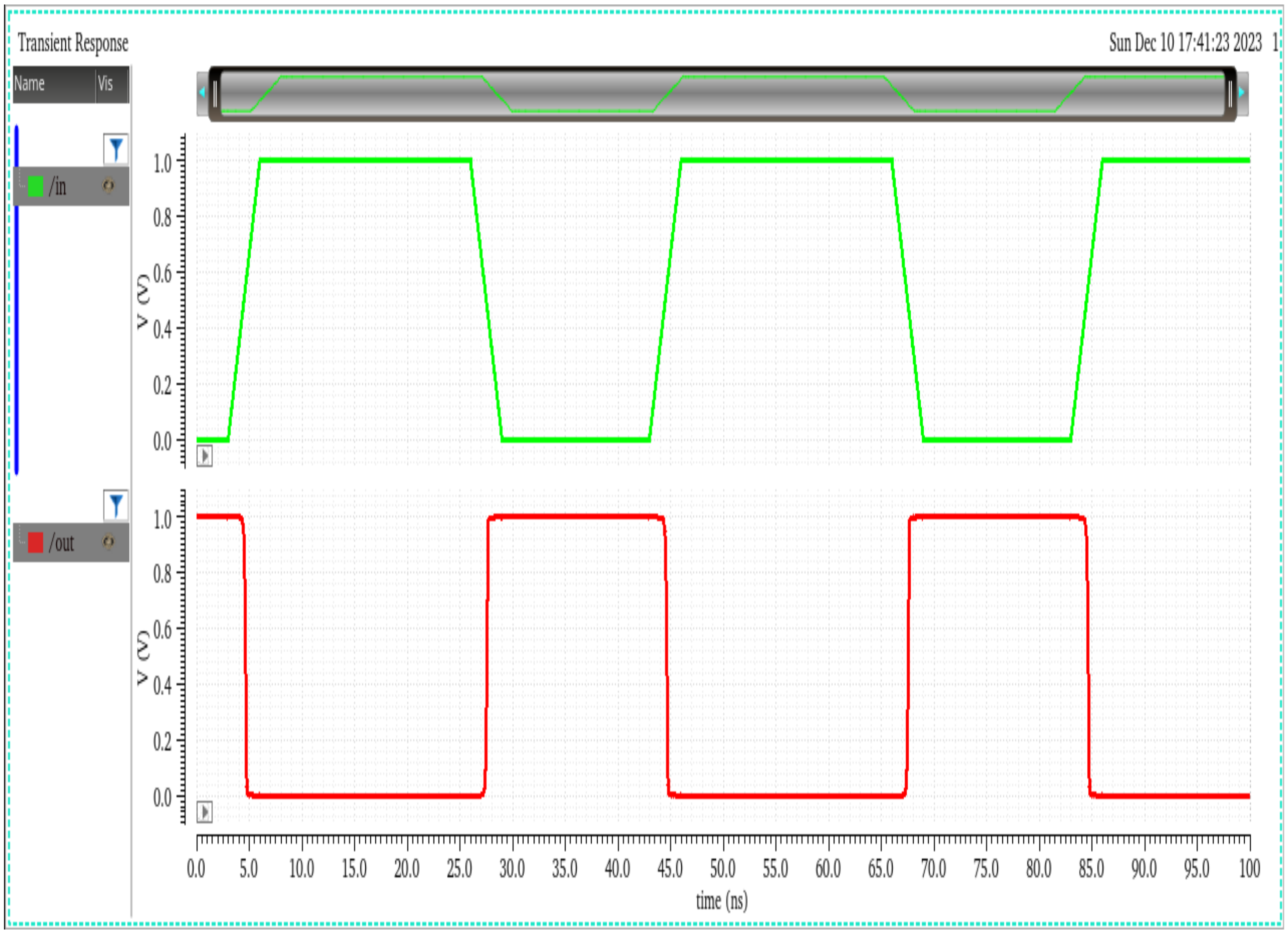


The output will appear in virtuoso visualization window as shown below.



13. Finally, we are going to separate the plots into two sub-graphs. Click on the following icon for splitting graphs.

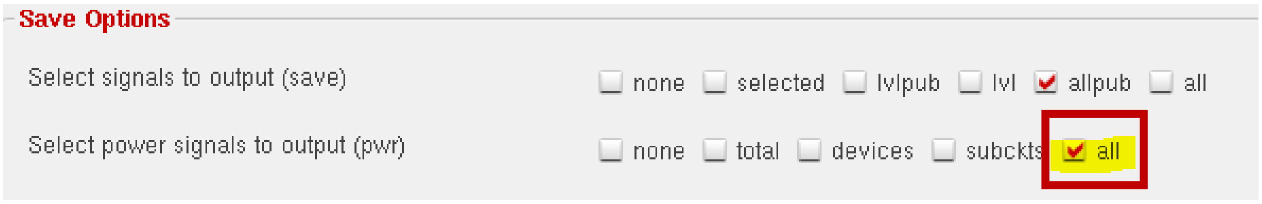
The final plot should look like the one shown below:



Measuring average power using Waveform calculator

We will compute the average power consumed in a circuit for the duration of transient simulation window.

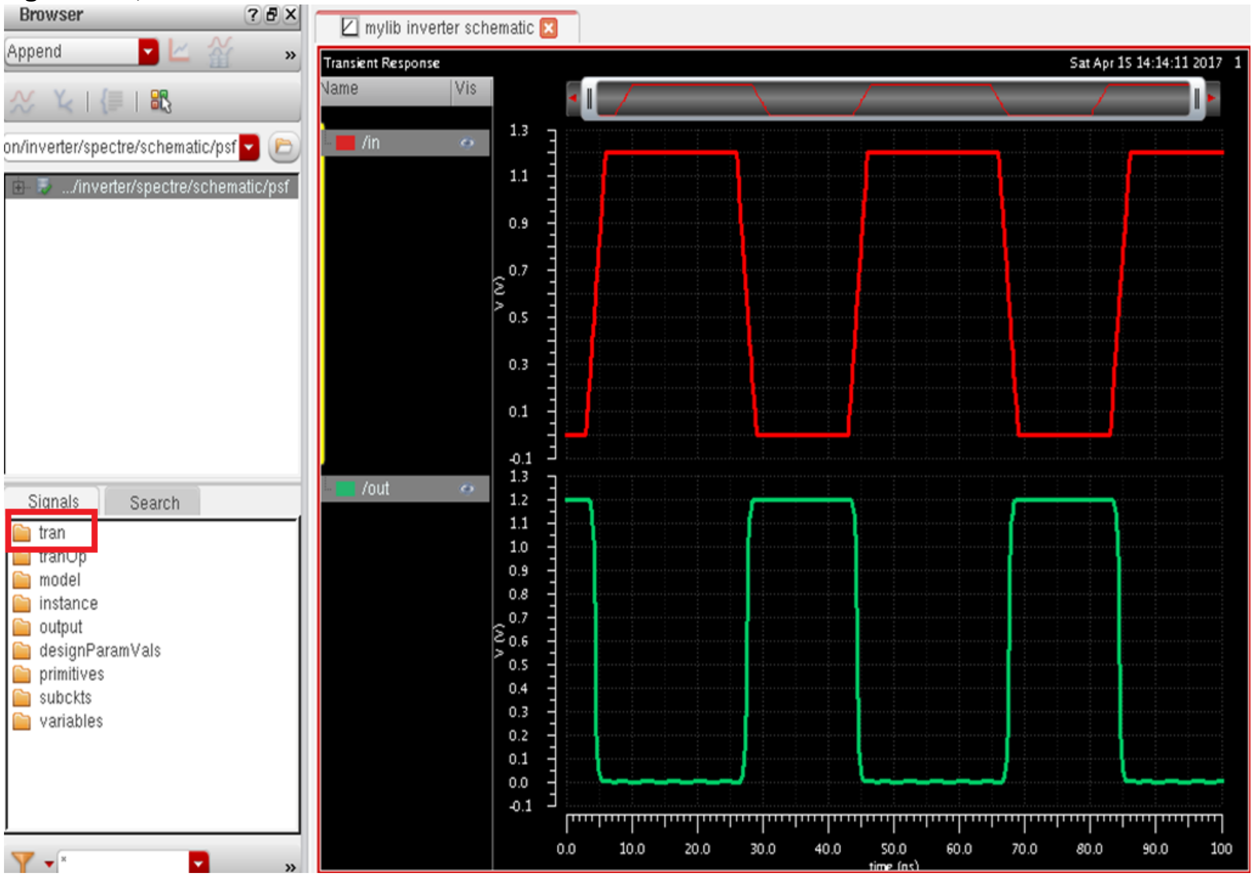
1. To do this, make sure that before running simulation you select the ***Outputs→Save All*** option in ADE L window. '**Save Options**' window will appear. Under '**Select power signals to output (pwr)**' option, put a tick mark in **all** option. Click **OK**.


Save Options

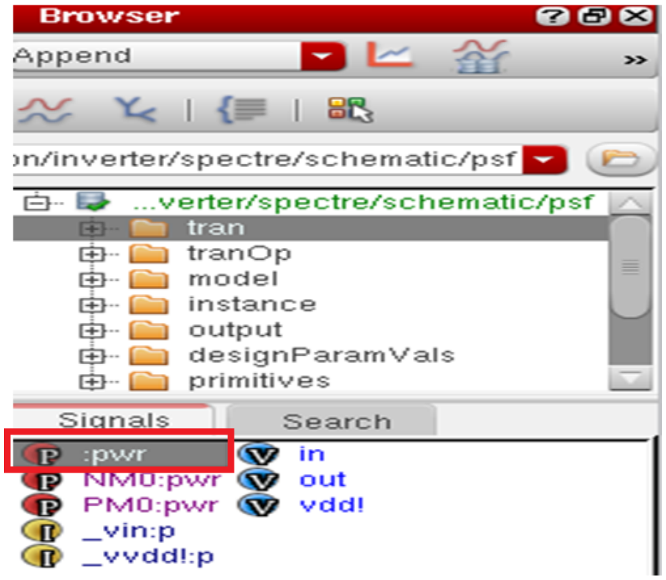
Select signals to output (save) none selected lvlpub lvl [✓ allpub] all

Select power signals to output (pwr) none total devices subckts [✓ all]

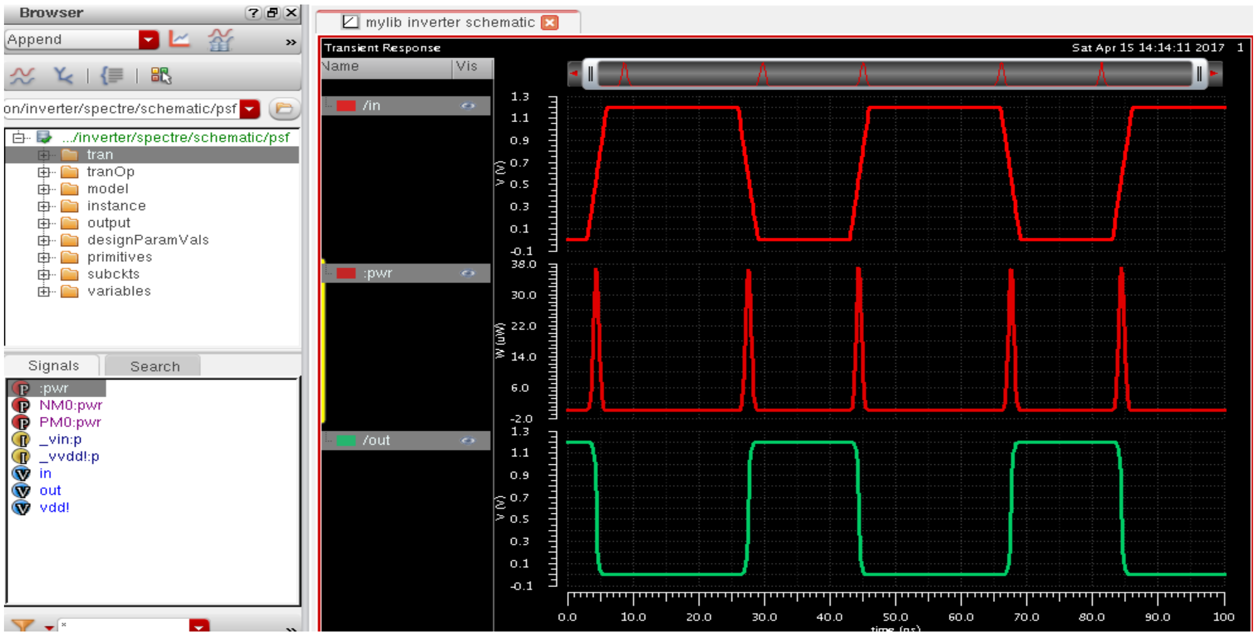
2. Then simulate the circuit as usual, by executing ***Simulation→Netlist and Run***.
Execute ***Tools→ Result Browser*** in ADE L window. '**Result Browser**' window will appear to the left side in '**Virtuoso Analysis and Visualization XL**' window. Double-click on **tran**. From the signals list,



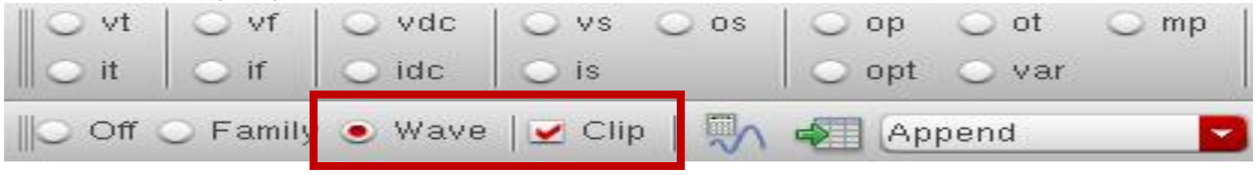
3. Now double-click on **:pwr**



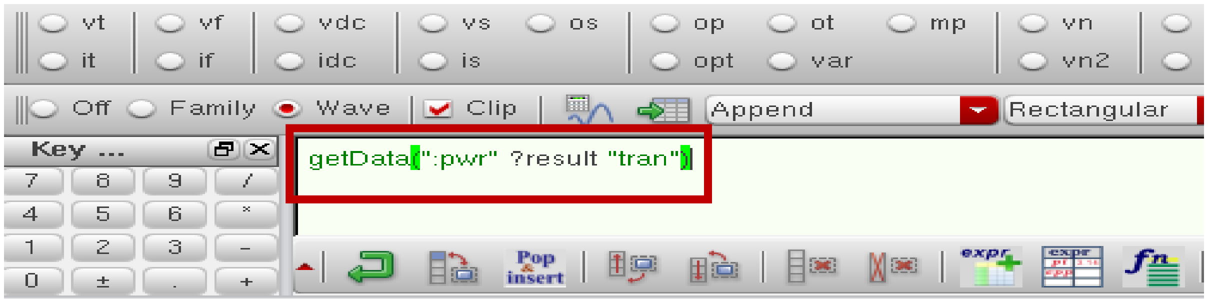
4. The waveform display window will show the "**:pwr**" (the instantaneous power consumed by the whole circuit) along with 'in' and 'out' signals.



5. Now, open **Waveform calculator** window. The calculator window appears. Make sure the "**Wave**" and "**Clip**" options are selected.

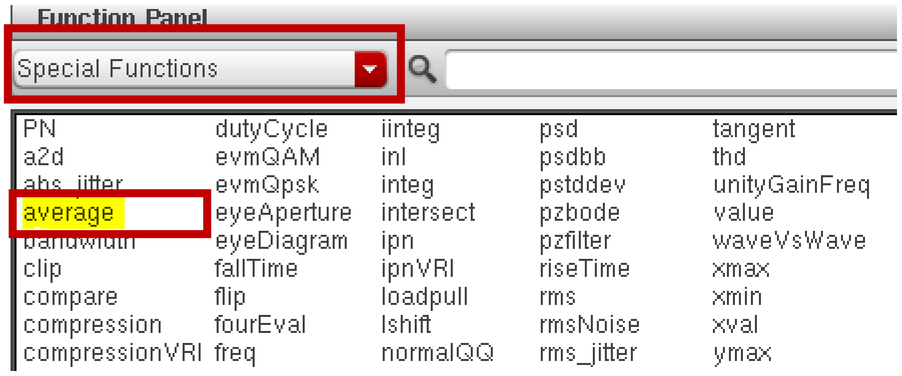


6. Now switch back to the waveform window and left click the mouse once on the power waveform. Then switch back to the calculator window. The buffer window should be filled in as follows:

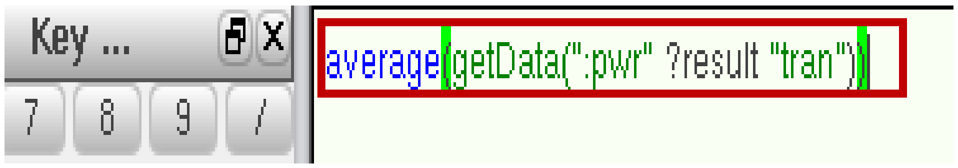


```
getData(":pwr" ?result "tran")
```

7. Now select '**average**' from '**Special Functions**' Menu and click apply.

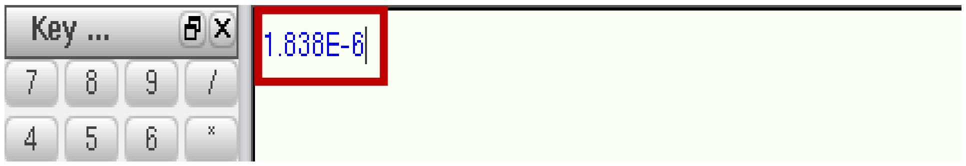


8. The buffer will now look like the following one:



```
average(getData(":pwr" ?result "tran"))
```

9. Click on **Evaluate the buffer** icon and the average power dissipation in that time window will be displayed (about 1.838 μW in this example).

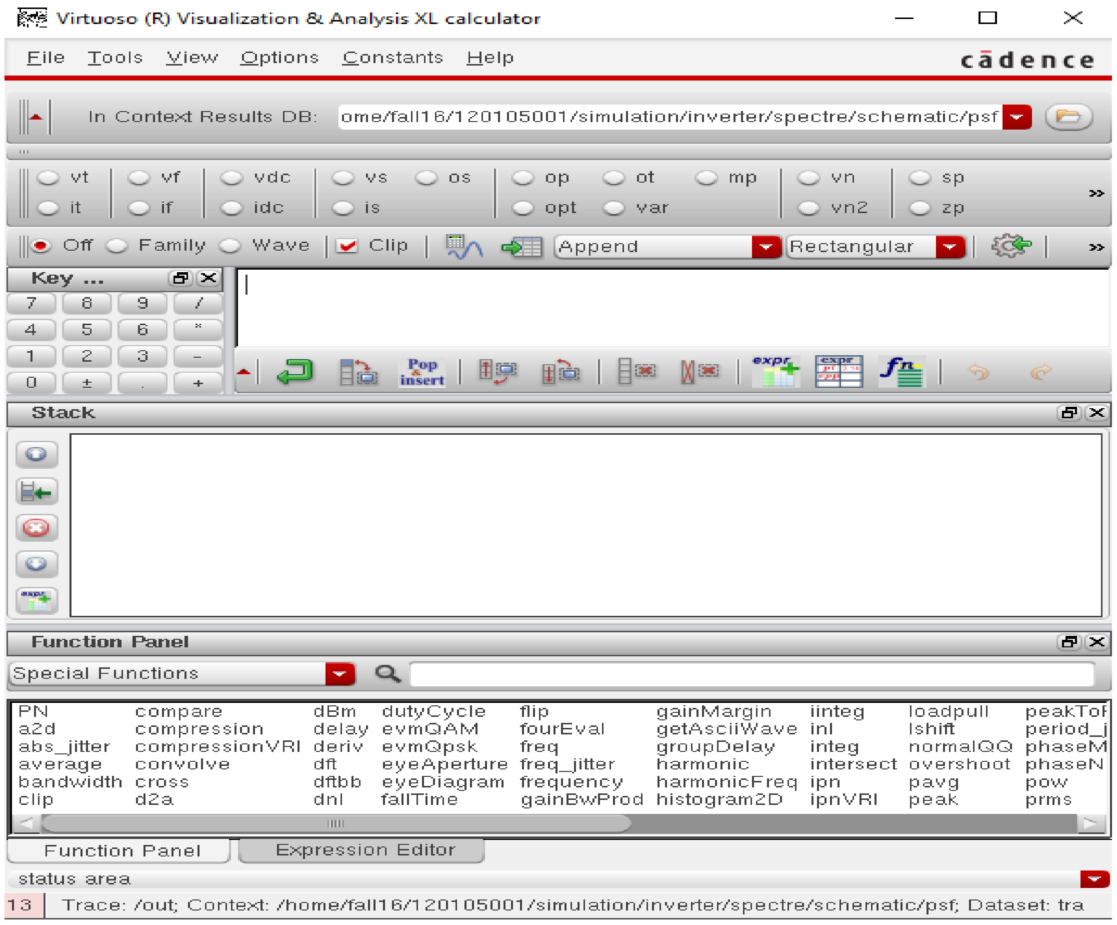


```
1.838E-6
```

Measuring propagation delay using Waveform calculator

Waveform calculator can be used to perform many different measurements and transformations on the waveforms displayed in the waveform window. This includes – computing the average of a waveform (e.g. power) over the entire length of the simulation or in a given period of time, finding the propagation delay of between input and output signals, or addition/subtraction/multiplication/division of waveforms, etc.

1. Execute ***Tools→Calculator*** in Virtuoso Visualization & Analysis XL window. '**Virtuoso Visualization & Analysis XL calculator**' window will pop-up:


Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
cadence
vt vf vdc vs os op ot mp vn sp
it if idc is opt var vn2 zp
Off Family Wave Clip
Key ...
7 8 9 /
4 5 6 *
1 2 3 -
0 ± . +
Pop insert
Stack
Function Panel
Special Functions
PN compare dBm dutyCycle flip gainMargin iinteg loadpull peakToP
a2d compression delay evmQAM fourEval getAsciiWave inl lshift period_j
abs_jitter compressionVRI deriv evmQpsk freq groupDelay integ normalQQ phaseM
average convolve dft eyeAperture freq_jitter harmonic intersect overshoot phaseN
bandwidth cross dftbb eyeDiagram frequency harmonicFreq ipn pavg pow
clip d2a dnl fallTime gainBwProd histogram2D ipnVRI peak prms
Function Panel
Expression Editor
status area

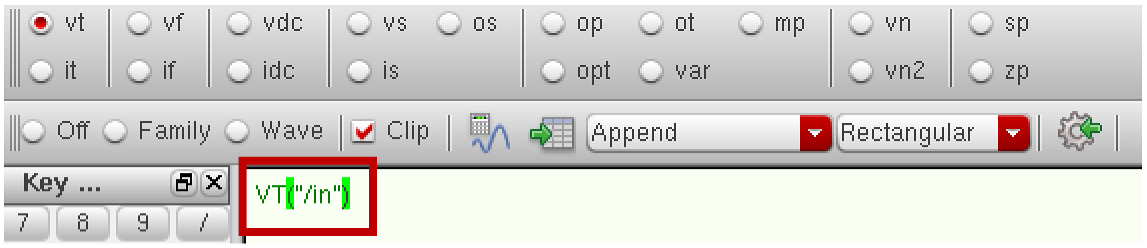
In Context Results DB: ome/fall16/120105001/simulation/inverter/spectre/schematic/psf

Append

Rectangular

13 Trace: /out; Context: /home/fall16/120105001/simulation/inverter/spectre/schematic/psf; Dataset: tra

2. Select '**vt**'. Go to Schematic editor window and click on input node '**in**'. An expression (e.g. **VT("/in")**) will appear. Copy the expression.

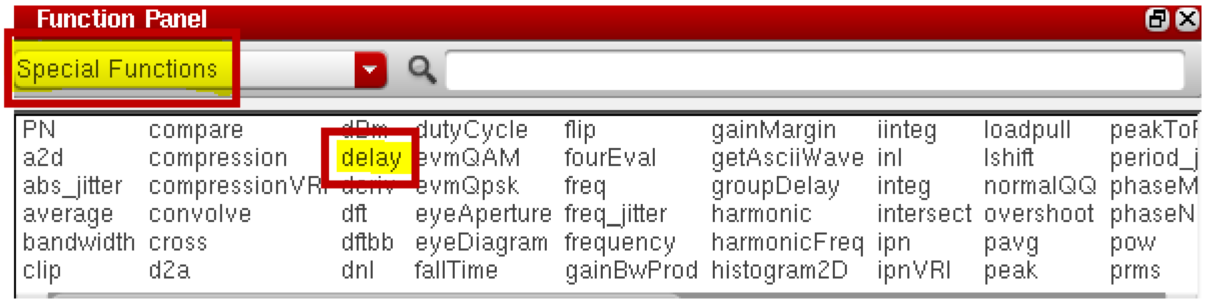


Append

Rectangular

VT("/in")

3. In the **Function Panel**, select '**Special functions**' and select '**delay**'.

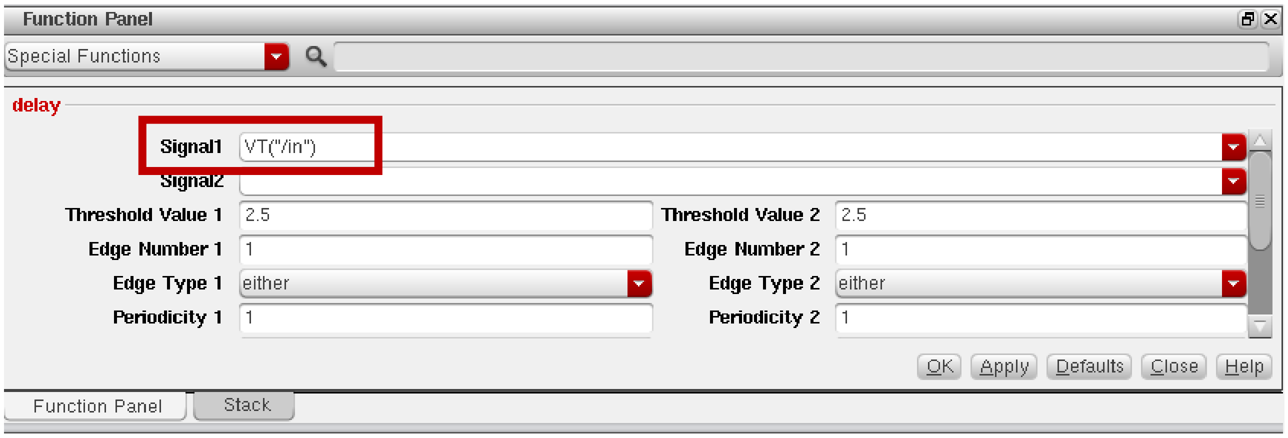


Function Panel

Special Functions

```
PN         compare        dBm    dutyCycle   flip        gainMargin     iinteg     loadpull   peakToF
a2d        compression    delay  evmQAM      fourEval    getAsciiWave   inl        lshift     period_j
abs_jitter compressionVRI deriv  evmQpsk     freq        groupDelay     integ      normalQQ   phaseM
average    convolve       dft    eyeAperture freq_jitter harmonic       intersect  overshoot  phaseN
bandwidth  cross          dftbb  eyeDiagram  frequency   harmonicFreq   ipn        pavg       pow
clip       d2a            dnl    fallTime    gainBwProd  histogram2D    ipnVRI     peak       prms
```

4. The following window will appear. Put the expression previously obtained in the field '**Signal1**'. Do the same for output signal '**out**' to fill in the field '**Signal2**'.



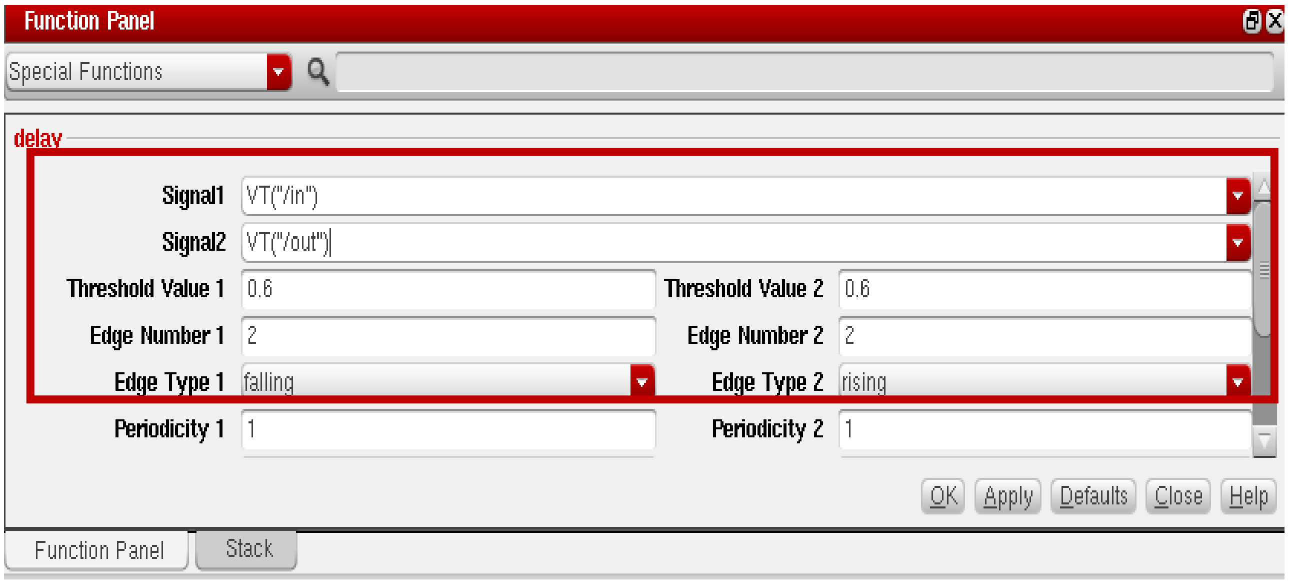
Function Panel

Special Functions

delay

Signal1	VT("/in")		
Signal2			
Threshold Value 1	2.5	Threshold Value 2	2.5
Edge Number 1	1	Edge Number 2	1
Edge Type 1	either	Edge Type 2	either
Periodicity 1	1	Periodicity 2	1

Fill up the rest of the form as follows:



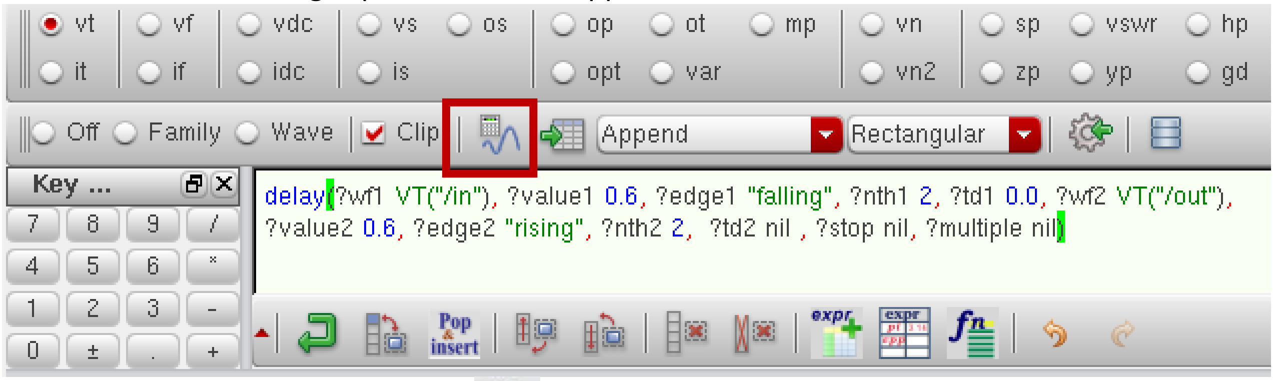
Function Panel

Special Functions

delay

Signal1	VT("/in")		
Signal2	VT("/out")		
Threshold Value 1	0.6	Threshold Value 2	0.6
Edge Number 1	2	Edge Number 2	2
Edge Type 1	falling	Edge Type 2	rising
Periodicity 1	1	Periodicity 2	1

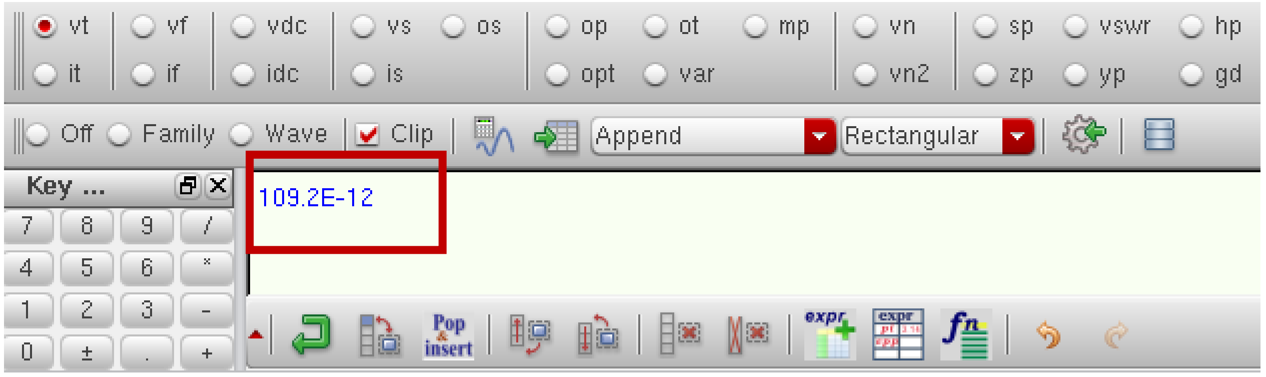
5.Click **OK**. The following expression should appear:



```
delay(?wf1 VT("/in"), ?value1 0.6, ?edge1 "falling", ?nth1 2, ?td1 0.0, ?wf2 VT("/out"),
?value2 0.6, ?edge2 "rising", ?nth2 2,  ?td2 nil , ?stop nil, ?multiple nil)
```

6. Click on **Evaluate the buffer** icon.

The propagation delay (in seconds) will be displayed in the window.



```
109.2E-12
```

Measuring rise time and fall time using Waveform calculator

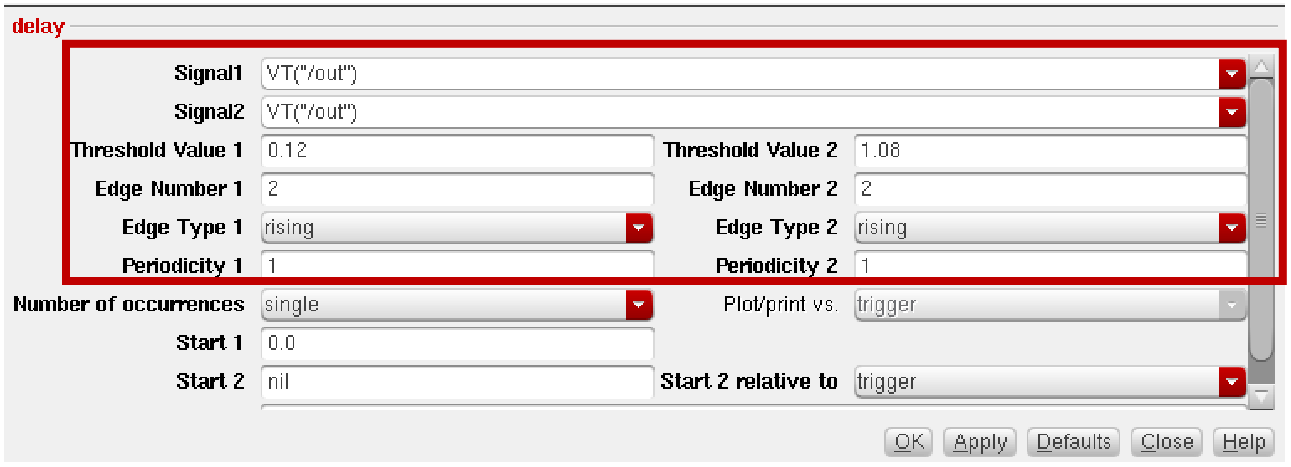
1. Open the '**delay**' function window under **Waveform calculator** in the same way that you followed for propagation delay measurement. This time both **Signal1** and **Signal2** will be **VT("/out")**.

2. Threshold value 1 and 2 should be **0.12** (10% of 1.2 V supply) and **1.08** (90% of 1.2 V supply) respectively for 10% to 90% rise time calculation. *These values should be swapped for fall time calculation.*

3. For rise time/fall time calculation, both the **Edge numbers** must be the same.

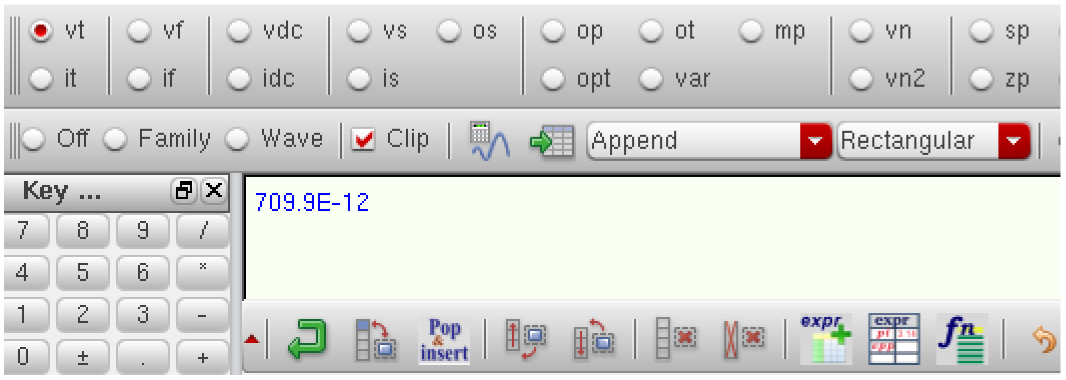
4. The **Edge type**s should be rising for rise time calculation and falling for fall time calculation.

Example: Rise time calculation of rising edge 2 for an inverter:


delay

Signal1	VT("/out")		
Signal2	VT("/out")		
Threshold Value 1	0.12	Threshold Value 2	1.08
Edge Number 1	2	Edge Number 2	2
Edge Type 1	rising	Edge Type 2	rising
Periodicity 1	1	Periodicity 2	1
Number of occurrences	single	Plot/print vs.	trigger
Start 1	0.0		
Start 2	nil	Start 2 relative to	trigger

5. Click **OK** after filling in the form as shown above. The following expression should appear:



709.9E-12

Student Task

1. Explain the nature of power consumption curve.
2. Perform SS, FF, SF and FS process corner simulations and compare the power consumption and propagation delays.

Lab-6: DC Sweep and Parametric Sweep in Cadence Virtuoso

Objectives:

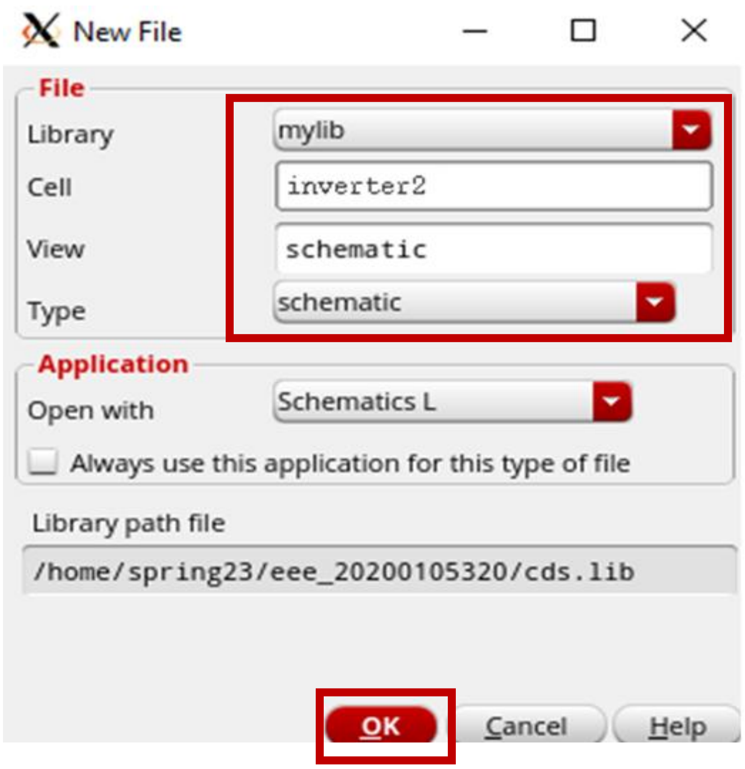
- To learn how to perform DC sweep and parametric simulation.
- To learn the voltage transfer characteristics of inverter.

Obtain the voltage transfer characteristics of inverter

We will perform both DC and parametric simulation at the same time on inverter schematic. We will obtain the transfer characteristic curve (TCC) of inverter from DC simulation and by varying the width of the PMOS transistor; we will observe its effect on transfer characteristics.

1. Open Cadence Virtuoso and create a new schematic by executing, ***File→ New→ Cellview***. Set up the '**New File**' in the **Command Interpreter Window (CIW)**

Library: *mylib*, **Cell:** *inverter2*, **View:** *schematic*, **Type:** *schematic*, **Application: Open with:** *Schematics L*



New File

File

Library: mylib

Cell: inverter2

View: schematic

Type: schematic

Application

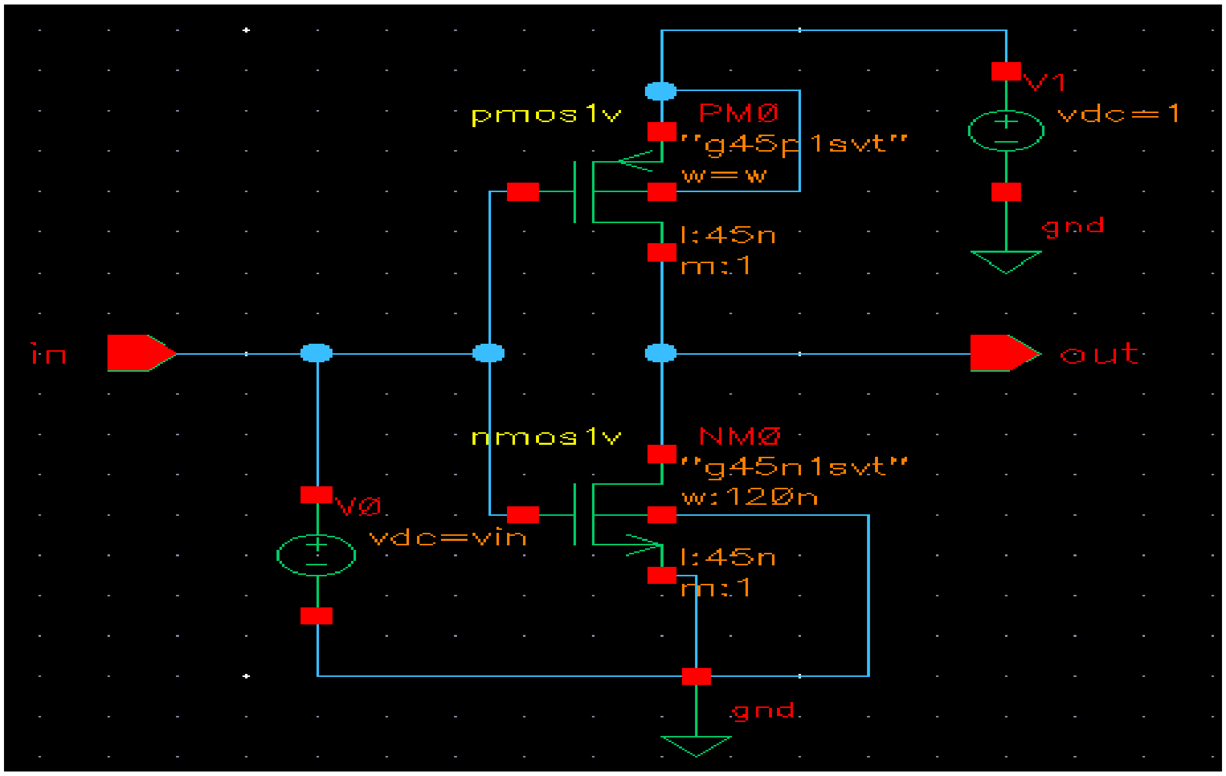
Open with: Schematics L

Always use this application for this type of file

Library path file

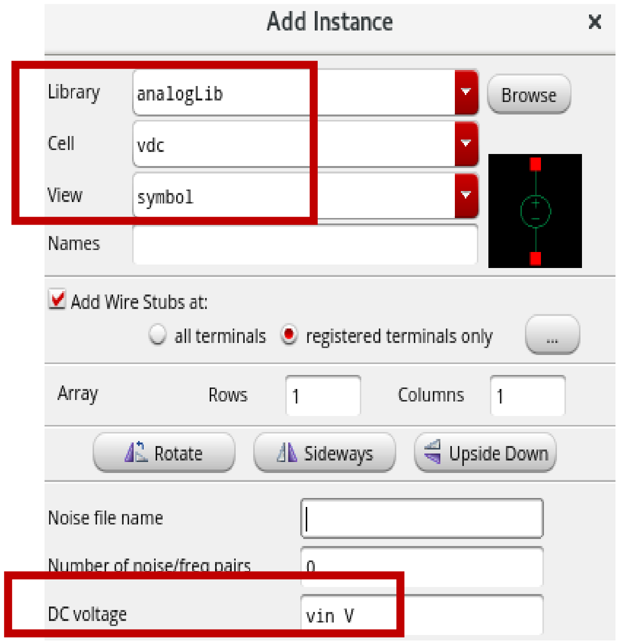
/home/spring23/eee_20200105320/cds.lib

2. In the schematic window design the following schematic.



Here the **V0** and **V1** is a **vdc** type source placed from the library **analogLib**. The **V0** source is connected between '**in**' and '**gnd!**' and the **V1** source is connected between **source of PMOS** and **gnd.** Now, select the **V0** source and click '**q**' , the '**Edit object properties**' window will open. Place **vin** under '**DC voltage**' field. Similarly, for the **V1** source place **1 v** as shown below.

For the source V0



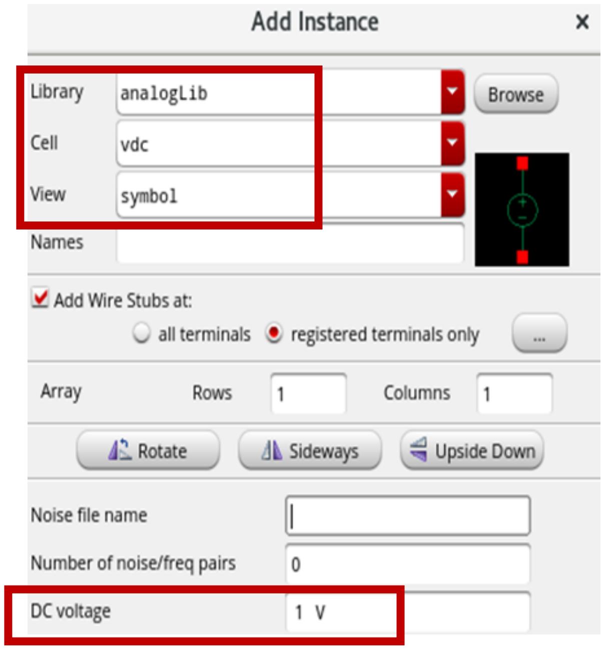
Library	analogLib
Cell	vdc
View	symbol
Names	

Add Wire Stubs at: [✓]
all terminals [✓ registered terminals only]

Array Rows 1 Columns 1

Noise file name	
Number of noise/freq pairs	0
DC voltage	vin V

For the source V1



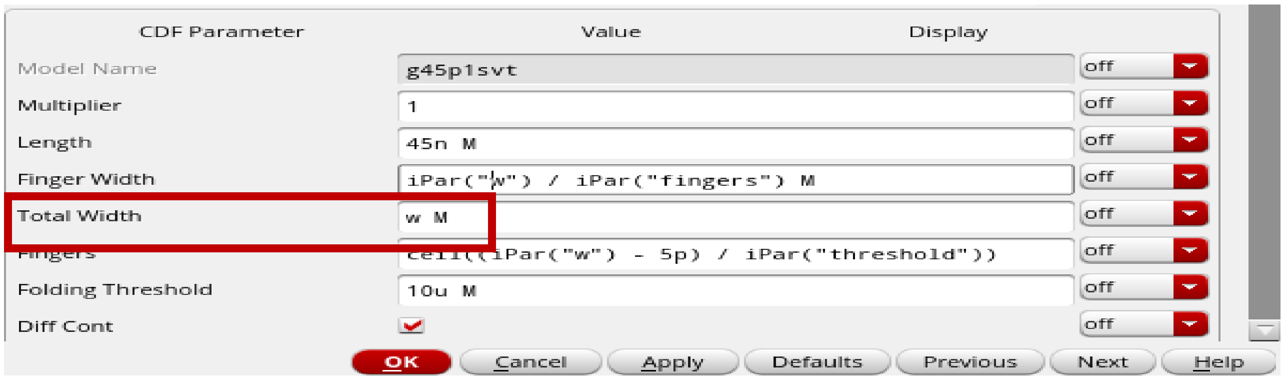
Library	analogLib
Cell	vdc
View	symbol
Names	

Add Wire Stubs at: [✓]
all terminals [✓ registered terminals only]

Array Rows 1 Columns 1

Noise file name	
Number of noise/freq pairs	0
DC voltage	1 V

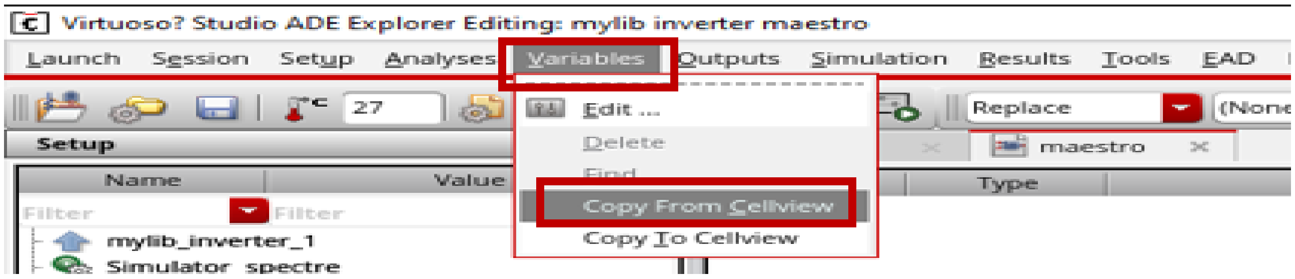
3. To sweep the width of PMOS transistor select the PMOS transistor in the schematic editor window, click '**q**' and '**Edit object properties**' window will open. Place ***w*** under '**Total Width**' and press **tab** on keyboard. The '**Finger Width**' field will be automatically changed as follows:



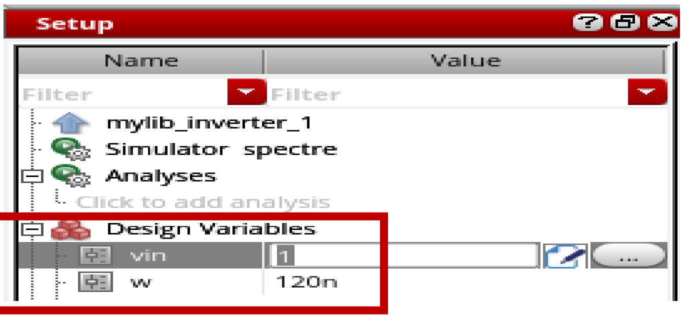
CDF Parameter	Value	Display
Model Name	g45p1svt	off
Multiplier	1	off
Length	45n M	off
Finger Width	iPar("w") / iPar("fingers") M	off
Total Width	w M	off
Fingers	ceil((iPar("w") - 5p) / iPar("threshold"))	off
Folding Threshold	10u M	off
Diff Cont	✓	off

4. Execute ***Launch→ADE Explorer*** and setup **Model Library** to **gpdk045_mos.scs** and section to **TT** similar to the way you did in **Lab 1**.

5. Execute ***Variables→Copy from Cellview***.



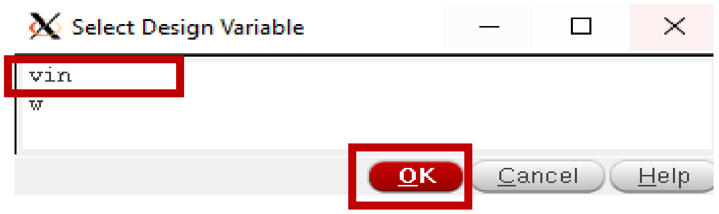
6. You will see '**w**' and '**vin**' appear in the '**Design Variables**' window. Click on '**w**' field, put a default value of **120n**.. Similarly click on '**vin**' field, put a default value of **1**.



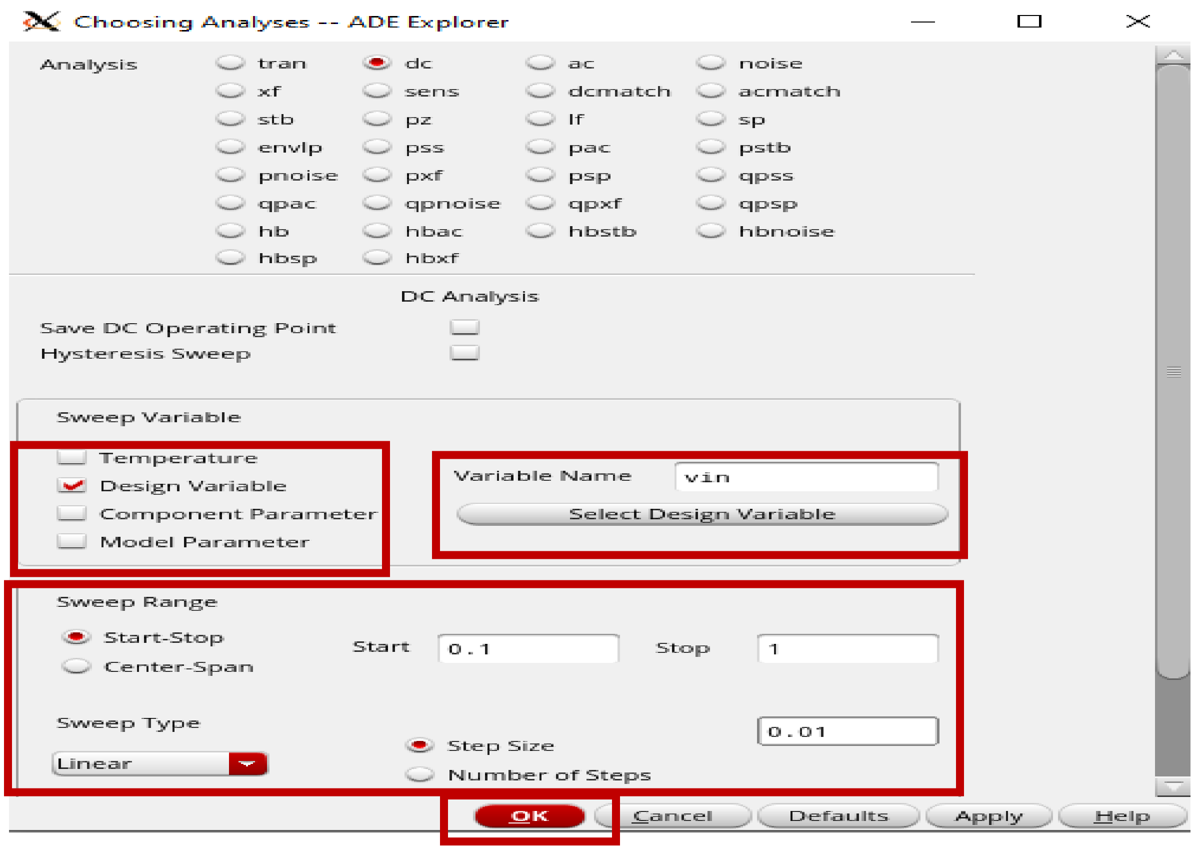
Setup

Name	Value
mylib_inverter_1	
Simulator spectre	
Analyses	
Click to add analysis	
Design Variables	
vin	1
w	120n

7. Execute ***Analyses →Choose*** and in the '**Choosing Analyses**' form, select **dc**. Under '**Sweep Variable**', select '**Design Variable**' and click on '**Select design variable**'. Select '**vin**' and click **OK**.



8. Under '**sweep range**', select '**start-stop**' and put a **start value** of **0.1** and a **stop value** of **1**. Select '**Sweep type**' to be '**linear**' and '**Step size**' to be 0.01. Click **OK**.
Your Window Should look like this.



Choosing Analyses -- ADE Explorer

Analysis	○ tran	[✓ ● dc]	○ ac	○ noise
	○ xf	○ sens	○ dcmatch	○ acmatch
	○ stb	○ pz	○ lf	○ sp
	○ envlp	○ pss	○ pac	○ pstb
	○ pnoise	○ pxf	○ psp	○ qpss
	○ qpac	○ qpnoise	○ qpxf	○ qpsp
	○ hb	○ hbac	○ hbstb	○ hbnoise
	○ hbsp	○ hbxf		

DC Analysis

Save DC Operating Point ☐
Hysteresis Sweep ☐

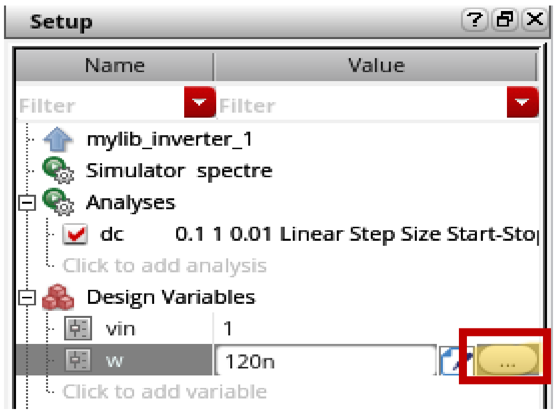
Sweep Variable
☐ Temperature
[✓ Design Variable]
☐ Component Parameter
☐ Model Parameter

Variable Name: vin
Select Design Variable

Sweep Range
[✓ ● Start-Stop] Start: 0.1 Stop: 1
○ Center-Span

Sweep Type: Linear
[✓ ● Step Size] 0.01
○ Number of Steps

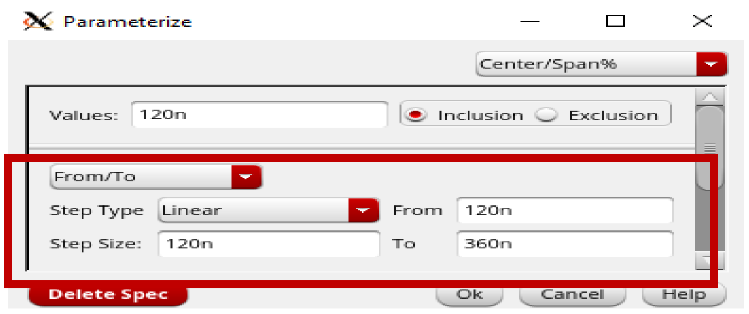
9. Execute ***Outputs → To be plotted*** and select '**out**' pin on the schematic.
Select **Netlist and run** to simulate a single TCC for the given default **PMOS width of 120nm**.
10. For Parametric Analysis, Click on, "w" under Design variable of the **ADE Explorer** window.



Setup

Name	Value
Filter	Filter
mylib_inverter_1	
Simulator	spectre
Analyses	
✓ dc	0.1 1 0.01 Linear Step Size Start-Sto
Click to add analysis	
Design Variables	
vin	1
w	120n
Click to add variable	

11. Click on the **3 dot icon**. The following window will appear.



Parameterize

Center/Span%

Values: 120n

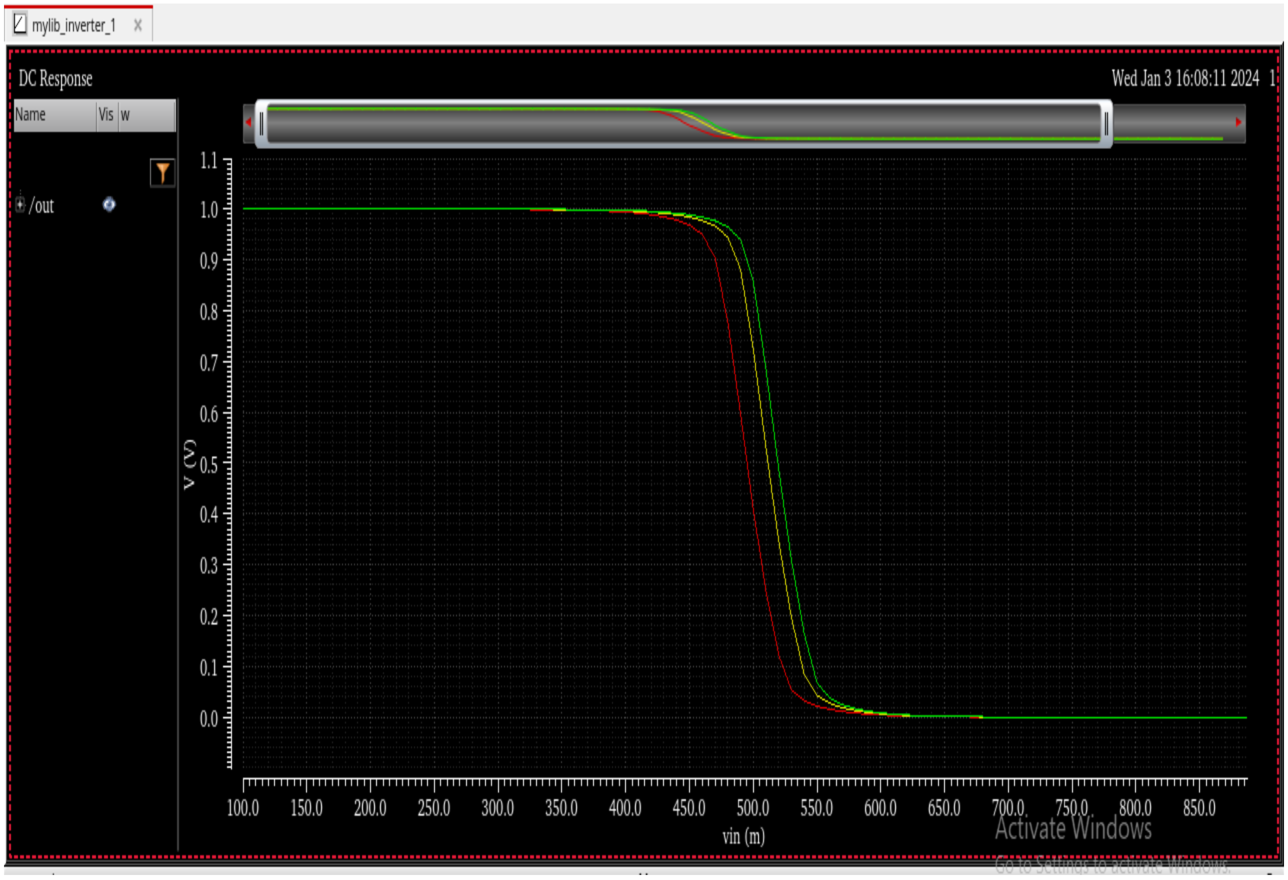
Inclusion [✓] Exclusion

From/To

Step Type: Linear — From: 120n

Step Size: 120n — To: 360n

12. Click on **"From/To"** From the drop down menu. Choose Step type as **liner**. Put **From:** 120n and **To: 360**n. and put **Step Size:** 120n. Click on **Ok.** Now run the simulation by executing ***Simulation → Netlist and Run***.The output should look like following

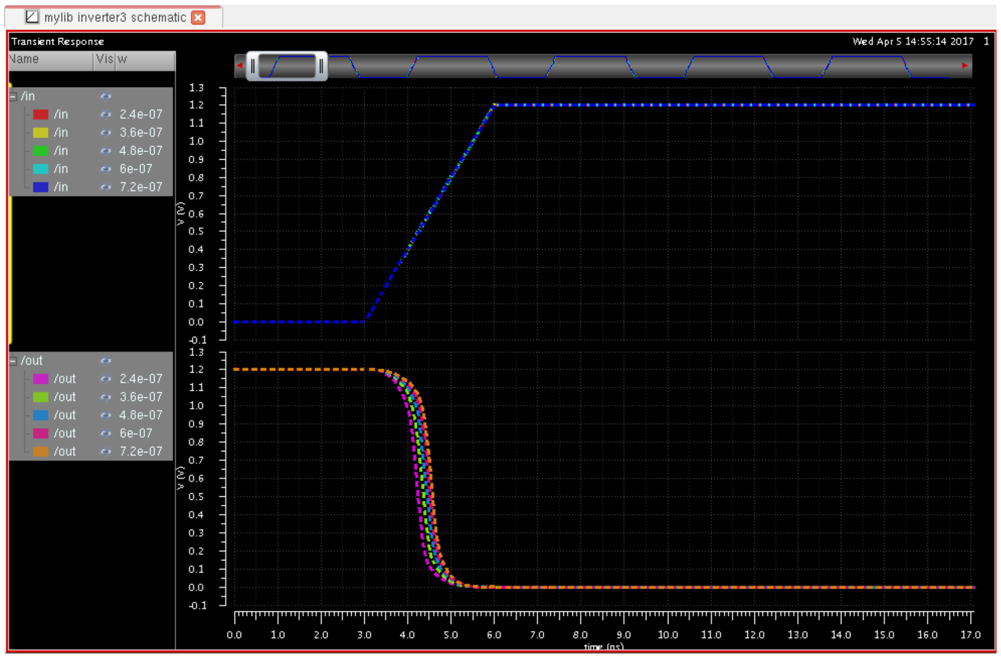


These transfer characteristics can be further explored to find Noise margin, and inversion voltage for different widths of PMOS.

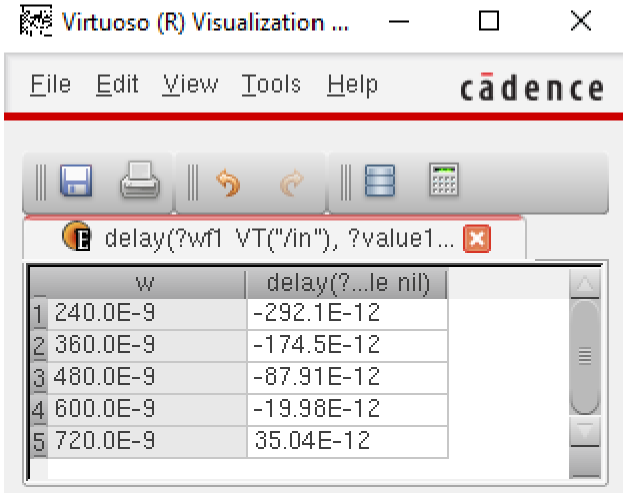
13. Save the state of the ADE Explorer window.

Student Task

1. Show the effect of changing NMOS width on the TCC of an inverter.
2. Perform a parametric analysis in transient simulation to show the effect of changing the PMOS width on the propagation delay. You are expected to obtain a similar graph as shown below:



Setup necessary settings for the **delay** function in the **waveform calculator.** Click **OK** and click on '**Evaluate the buffer and display the results in a table**' icon.
The results should be displayed in a table like below:



delay(?wf1 VT("/in"), ?value1...

	w	delay(?...le nil)
1	240.0E-9	-292.1E-12
2	360.0E-9	-174.5E-12
3	480.0E-9	-87.91E-12
4	600.0E-9	-19.98E-12
5	720.0E-9	35.04E-12

3. Some of the propagation delays are negative. Explain why?
4. Explain the change in propagation delay with the change in PMOS transistor width.
5. Obtain the output characteristic curve (ID vs VDS) and transfer characteristic curve (ID vs VGS) of NMOS and PMOS.

Lab-7: Custom Layout of an Inverter

Objectives:

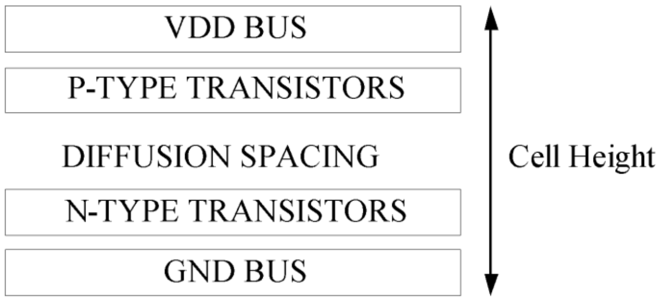
- To create a layout view of the basic inverter circuit from scratch in Virtuoso Layout Editor
- To design the layout keeping basic design rules in mind
- To design cell layout of a constant height for use in hierarchical design

Introduction to Layout, DRC and LVS

Layout is representation of a circuit in terms of planar geometric shapes (e.g. rectangles, polygons) showing the patterns of metal, polysilicon, oxide, or diffusion layers that make up the components (resistors, inductors, capacitors, transistors) of the integrated circuit.

When using a standard process (e.g. 45nm, 90nm or 180nm process available in our lab), the behaviour of the final integrated circuit depends significantly on the positions and interconnections of the geometric shapes due to parasitic resistances and capacitances contributed by them. While designing a layout, designer must keep in mind performance (e.g. power-delay product) and size (area occupied by the chip) criterion.

While designing digital circuits, one usually follows an ASIC design flow, where, the height of standard cells that are used is the same throughout the cell library, but their widths must vary according to their logical functions and drive strengths. The following figure shows a generalized standard cell height concept:



Although we will follow a full-custom IC design flow, we will maintain same cell height throughout our cell library.

The generated layout must pass a series of checks in a process known as physical verification. The most common checks in this verification process are:

- Design Rule Checking (DRC)
- Layout Versus Schematic (LVS) checking
- Parasitic extraction and post-layout simulation

Design Rule Check (DRC):

Design Rule Checking (DRC) is the process that determines whether the designed layout of a circuit satisfies a rules specified by the process being used.

Design Rules are a series of rules (e.g. area, width, overlap, enclosure, extension, spacing) provided by semiconductor manufacturers which are specific to a particular semiconductor manufacturing process. Design rules specify certain geometric and connectivity restrictions to ensure that the process can fabricate the device properly.

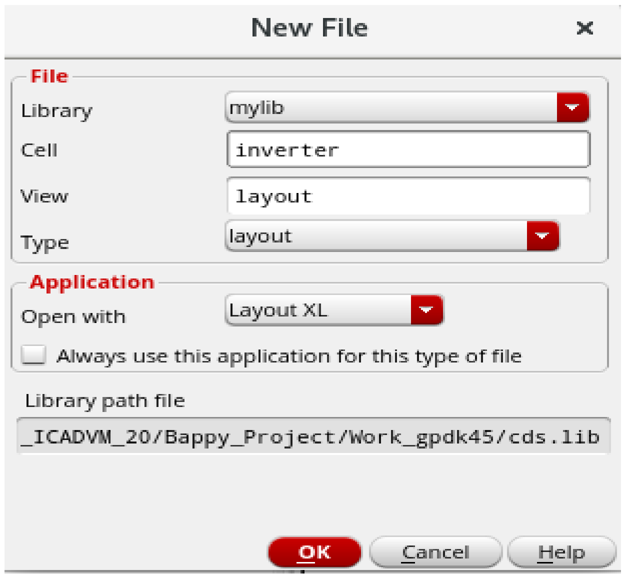
Layout versus Schematic (LVS) Check:

The Layout Versus Schematic (LVS) is the verification step to determine whether a particular integrated circuit layout corresponds to the original schematic or circuit diagram of the design. A successful Design rule check (DRC) ensures that the layout conforms to the rules designed/required for faultless fabrication. However, it does not guarantee if it really represents the circuit we desire to fabricate. This is why an LVS check is used.

Layout design using Virtuoso Layout Suite XL Editor

1. Invoke **Virtuoso Layout Suite XL Editor** from the **CIW** by executing ***File�New�Cellview***. The '**New File**' form appears. Fill it in as shown in the figure below:

Cell: *inverter*, **View:** *layout*. Click **OK**.


New File

File

Library: mylib
Cell: inverter
View: layout
Type: layout

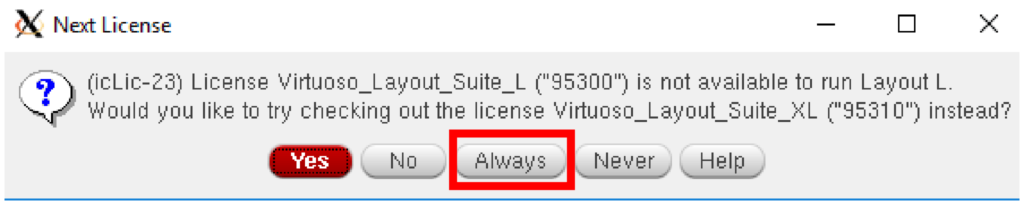
Application

Open with: Layout XL

☐ Always use this application for this type of file

Library path file: _ICADVM_20/Bappy_Project/Work_gpdk45/cds.lib

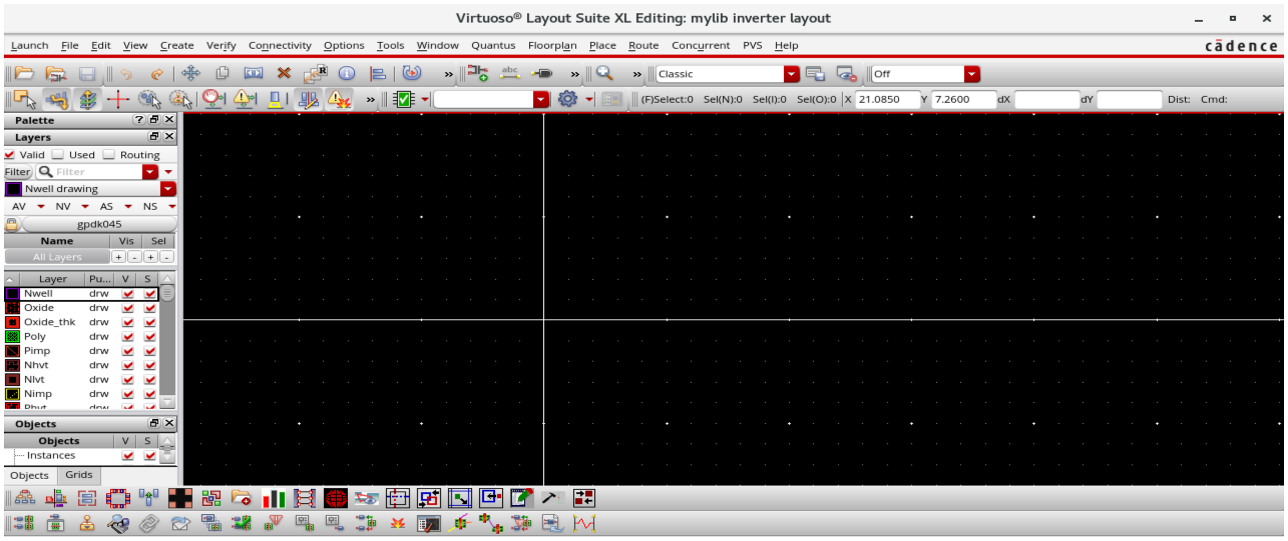
2. Click '**Always**' if the following window appears before Layout window appears.



Next License

(icLic-23) License Virtuoso_Layout_Suite_L ("95300") is not available to run Layout L.
Would you like to try checking out the license Virtuoso_Layout_Suite_XL ("95310") instead?

The following window of **Virtuoso Layout Suite XL Editor** will appear.



Virtuoso® Layout Suite XL Editing: mylib inverter layout

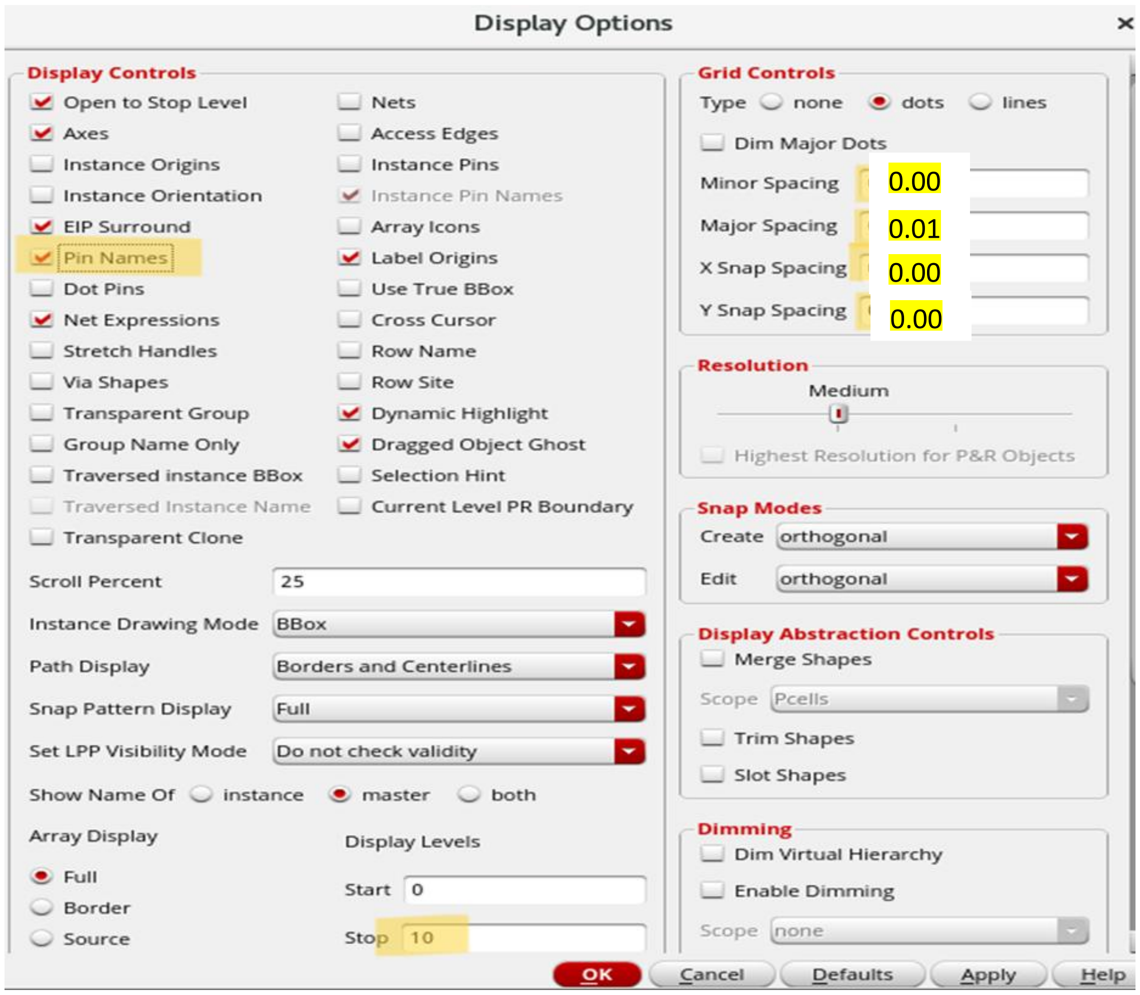
On the left side of the window, you will find a panel called 'Layers'. This panel is divided in three main categories which are: layer color, layer name and layer purpose. The details are described in the table below:

Color	Matches the color in the Editing window. Each layer has its own color and pattern. Each layer has two colors associated with it: a fill color and an outline color. These colors can be changed to fit your taste by editing the technology file.
Name	The type of layer (Nwell, Oxide, Poly, Metal1, etc)
Purpose	In gpdk045 the only purpose classifications are: drw = drawing, slot = slot Drawing is used in layout, slot is used to create a hole for metal stress relief

Verify that the layers display corresponds to the gpdk045layers shown in the GPDK 90 nm Mixed Signal Process Specification manual.

3. Before starting to design layout, you need to set the layout display configuration. Execute the following in the Virtuoso Layout Editor*: **Options→Display*** or press '**e**' on keyboard. Configure the form as shown in the figure below: You have to set the following parameters only:

Minor spacing	0.001
Major spacing	0.01
X snap spacing	0.005
Y snap spacing	0.005
Display Levels: Stop	10


Display Options

Display Controls

☑ Open to Stop Level
☑ Axes
☐ Instance Origins
☐ Instance Orientation
☑ EIP Surround
☑ Pin Names
☐ Dot Pins
☑ Net Expressions
☐ Stretch Handles
☐ Via Shapes
☐ Transparent Group
☐ Group Name Only
☐ Traversed instance BBox
☐ Traversed Instance Name
☐ Transparent Clone
☐ Nets
☐ Access Edges
☐ Instance Pins
☑ Instance Pin Names
☐ Array Icons
☑ Label Origins
☐ Use True BBox
☐ Cross Cursor
☐ Row Name
☐ Row Site
☑ Dynamic Highlight
☑ Dragged Object Ghost
☐ Selection Hint
☐ Current Level PR Boundary

Scroll Percent: 25
Instance Drawing Mode: BBox
Path Display: Borders and Centerlines
Snap Pattern Display: Full
Set LPP Visibility Mode: Do not check validity
Show Name Of: instance [✓ master] both

Array Display: [✓ Full] Border Source

Display Levels
Start: 0
Stop: 10

Grid Controls

Type: none [✓ dots] lines
☐ Dim Major Dots
Minor Spacing: 0.00
Major Spacing: 0.01
X Snap Spacing: 0.00
Y Snap Spacing: 0.00

Resolution

Medium
☐ Highest Resolution for P&R Objects

Snap Modes

Create: orthogonal
Edit: orthogonal

Display Abstraction Controls

☐ Merge Shapes
Scope: Pcells
☐ Trim Shapes
☐ Slot Shapes

Dimming

☐ Dim Virtual Hierarchy
☐ Enable Dimming
Scope: none

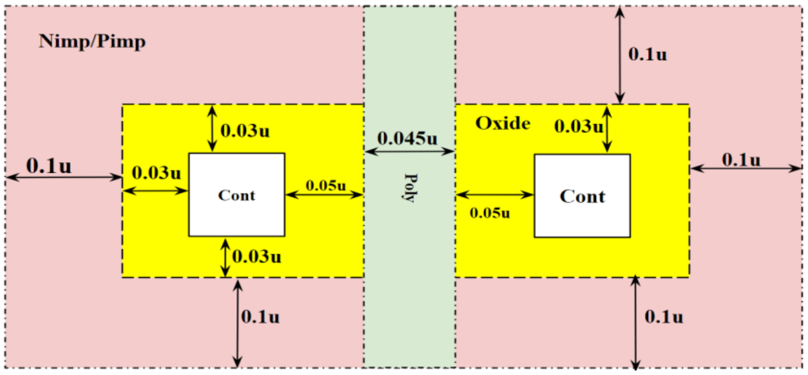
4. Now we are going to build the layout of the inverter. An inverter has an NMOS and a PMOS transistor. First we will build an NMOS transistor.

Layout of NMOS inverter consists of oxide, Nimp, Cont and Poly layers. Study the rules of these layers and calculate the minimum size of the Poly, Cont, Oxide and Nimp layer to create an NMOS transistor.

The rules related to the NMOS transistor can be summarised as follows:

Contact size	0.06 μm × 0.06 μm (Fixed)
Poly width (Minimum)	0.045 μm (Fixed MOS gate length)
Contact to poly spacing (Minimum)	0.05 μm
Contact to oxide enclosure (Minimum)	0.03 μm
Poly/Nimp extending from oxide (Minimum)	0.1 μm (gate side enclosure)
Nimpenclosing oxide (Minimum)	0.1μm (enclosure other than gate sides)
Minimum Metal 1 width	0.06 μm
Maximum Metal 1 width	6 μm
Minimum Metal 1 to Contact enclosure	0.03 μm (on at least two opposite sides)

The following figure illustrates some of the design rules mentioned above:



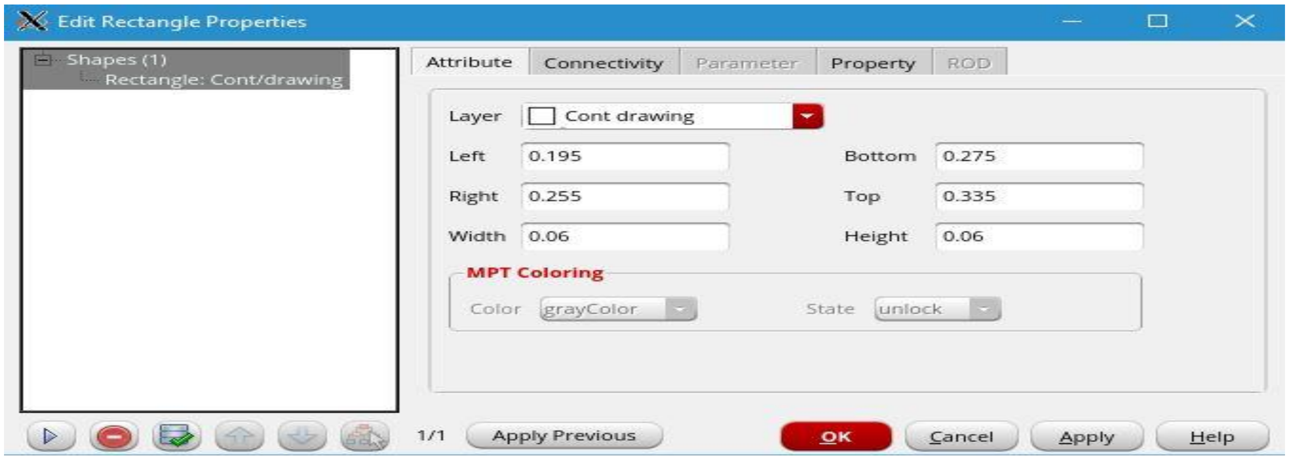
Now study the PMOS transistor structure in the GPDK 45 nm Mixed Signal Process Spec. The PMOS transistor consists of Oxide, Poly, Pimp, Cont and Nwell layer. Study the rules of these layers and calculate the minimum size of Poly, Cont, Oxide, Pimp and Nwell layer to create a PMOS transistor. The rules related to PMOS are same as NMOS except the there is an additional layer, the Nwell, whose rules are as follows:

Minimum Nwell width	0.3μm
Minimum Nwell spacing to Nwell (same potential)	0.3 μm
Minimum Nwell spacing to Nwell (different potential)	0.6 μm
Minimum Nwell spacing to N+ active area	0.16 μm
Minimum Nwell spacing to P+ active area	0.16 μm
Minimum Nwell enclosure to P+ active area	0.06 μm
Minimum Nwell enclosure to N+ active area	0.06 μm
Minimum N+ Active Area to P+ Active Area Spacing	0.1μm

Now we start building the NMOS and PMOS transistor layout. Look at the **LSW** and find the current drawing layer.

5. Click on the following icon in **Virtuoso Layout Suite XL Editor** window so that it notifies you anytime you make a violation of any design rule. When clicked, it will show '**DRD Notify ON**'. DRD stands for Design Rule Driven.

6. Select '**Cont (drw)**' (contact) layer from the '**Layers**' panel and draw a rectangle of '**Cont (drw)**' layer using ***Create⍰Shape⍰Rectangle*** or simple pressing '**r**'. Press '**Esc**' to stop '**create rectangle**' tool. In gpdk045 technology, **Cont** layers must be of dimension **0.06 μm x 0.06 μm**. So, if your rectangle is not of that dimension, click on the rectangle, press '**q**'. In the following window, check if the criterion has been met and change '**Width/Height**' if required.


Edit Rectangle Properties

Shapes (1)
Rectangle: Cont/drawing

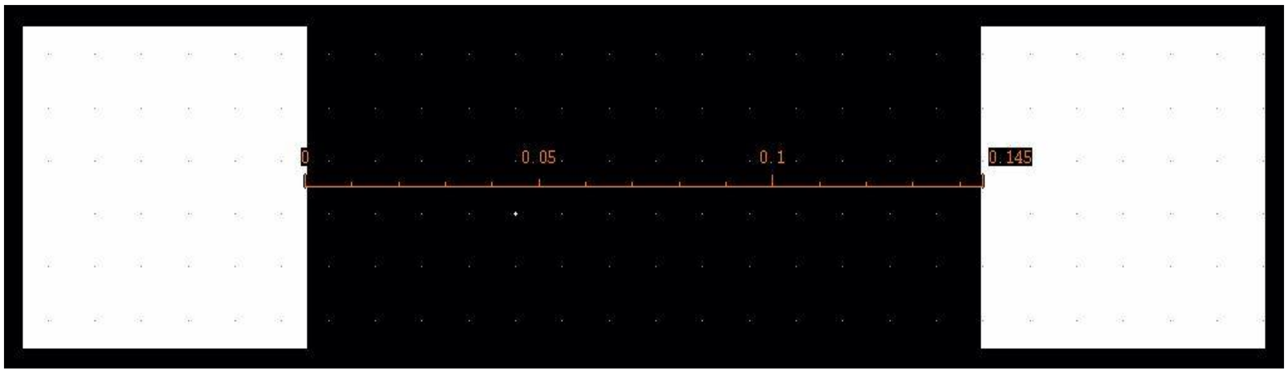
Layer: Cont drawing

Left	0.195	Bottom	0.275
Right	0.255	Top	0.335
Width	0.06	Height	0.06

MPT Coloring
Color: grayColor State: unlock

Contact to poly spacing must be **0.05 μm** in this technology and the **channel length** of NMOS/PMOS in our design is **0.045 μm**. So, we need a minimum space of **0.145 μm** between the contacts at source and drain.

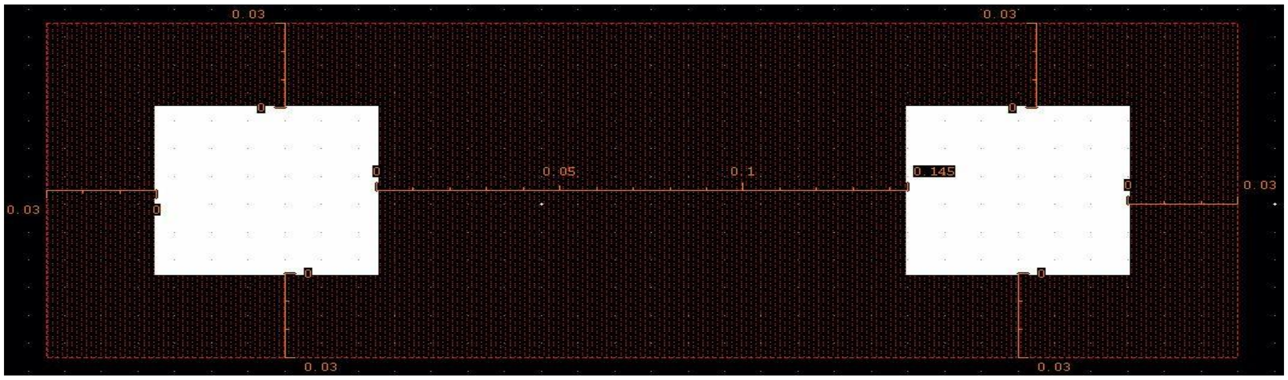
7. Press '**k**' to invoke the '**ruler**' tool. Use it to measure lengths whenever needed. To copy, press '**c**'. After placing two contacts, the layout looks like this:



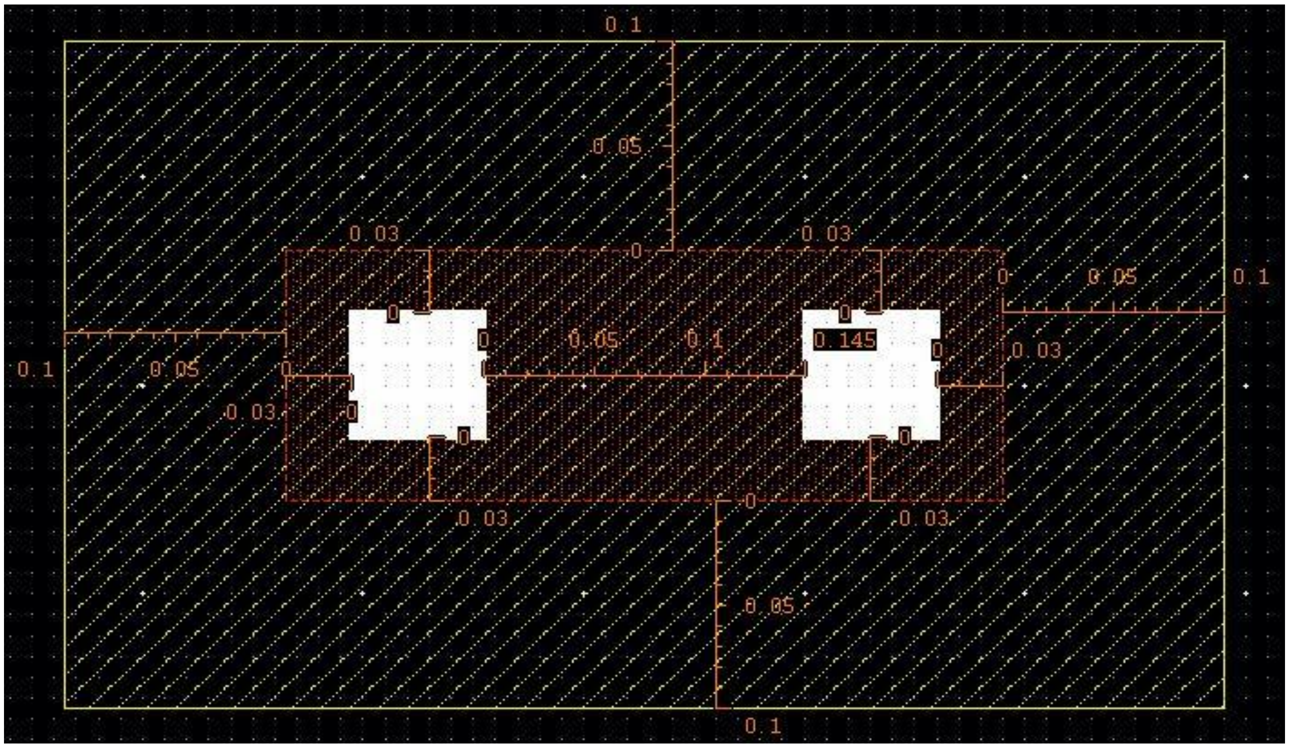
8. Now, **contact to oxide spacing** is minimum **0.03 μm**. So, draw a rectangle of '**Oxide (drw)**' layer so that it covers both the contacts and extends from each side by **0.03 μm**.

While drawing this, you will see Design rule violations when they are committed.

9. After drawing oxide layer, the layout should look like this:

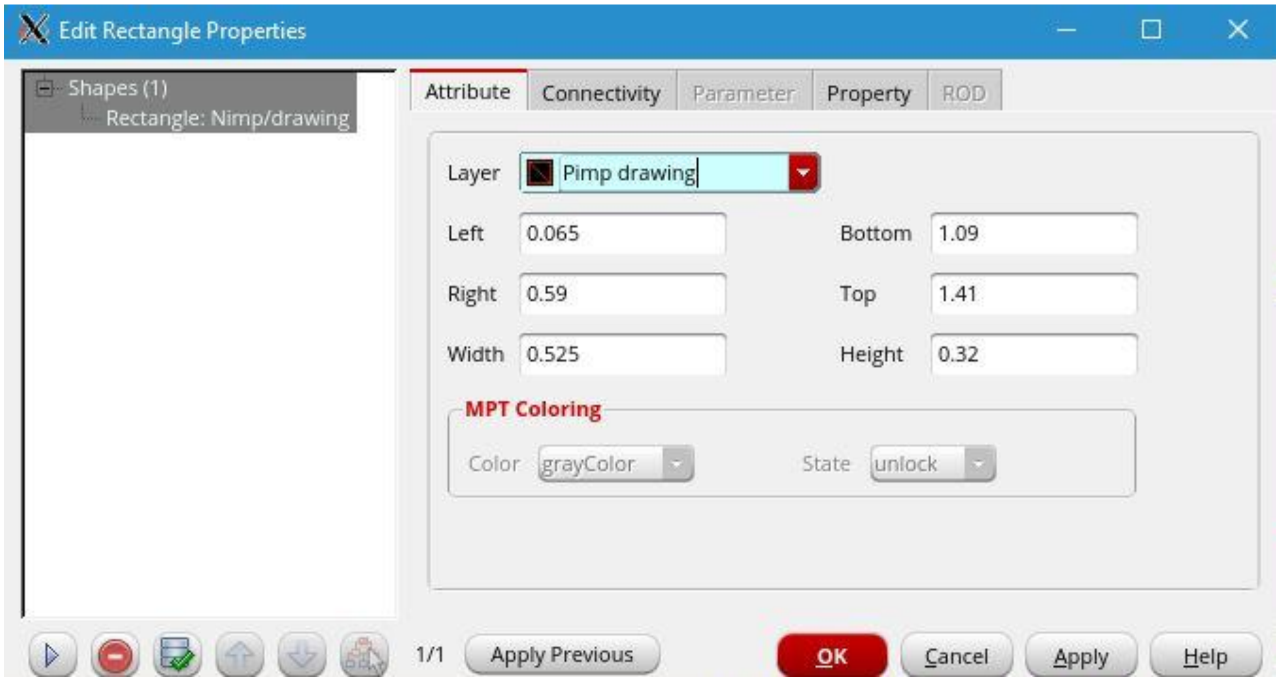


10. Now we will draw '**Nimp (drw)**' layer, which must extend from the oxide layer by a minimum of **0.1 μm**. First, draw a rectangle and then extend it to meet design rules. Use stretch tool by pressing '**s**'. Layout will look like the following:



11. Now, copy it and create another copy of all these layers by selecting all and pressing '**c**'.

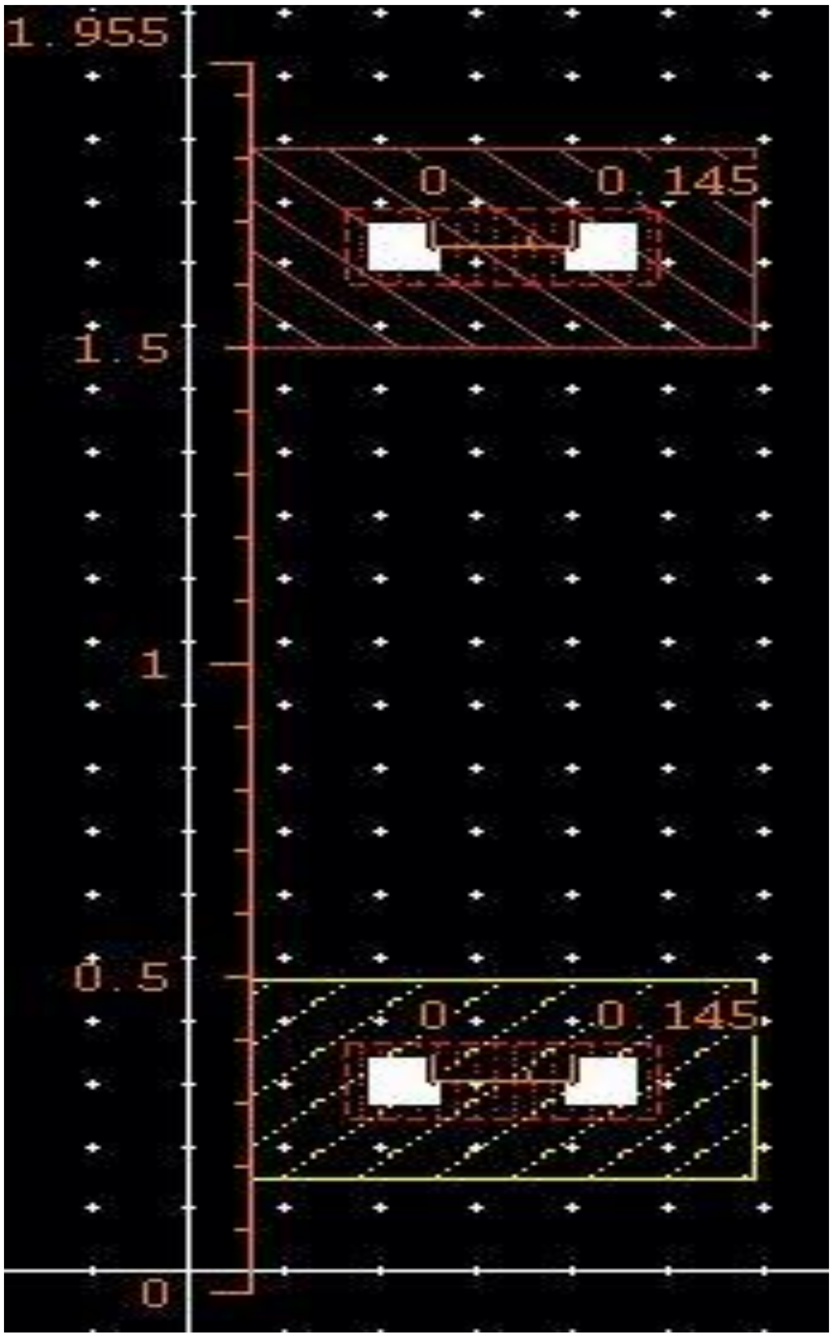
12. Click on the '**Nimp (drw)**' layer of the copy in the upper portion of the layout and press '**q**' to edit properties. From '**Edit Rectangle Properties**' window, select '**Pimp (drw)**' layer under '**Layer**' option. Click **OK**.



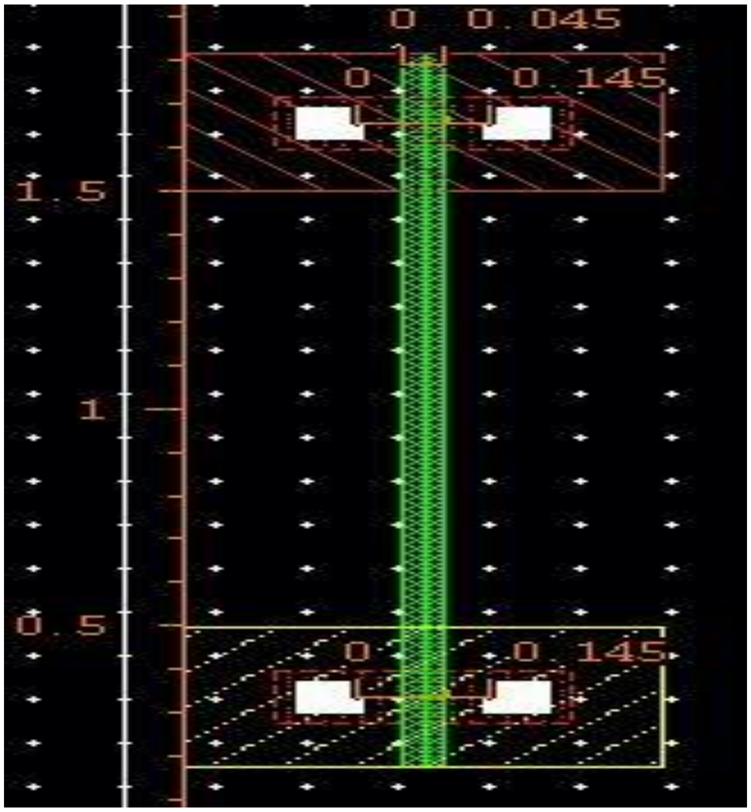
Layer: Pimp drawing
Left: 0.065
Bottom: 1.09
Right: 0.59
Top: 1.41
Width: 0.525
Height: 0.32
Color: grayColor
State: unlock

13. With more metal layers available in today's silicon processes, using the routing approach, such as first metal traverse vertically and second metal traverse horizontally, would be advantageous in standar cell physical design. Using this method, the second layer (e.g. Metal2) can be used for power and ground routing over internal standard cell transistors. In standard cell layout, it is preferable to use firt conducting layer, such as Metal1, as much as possible to make internal connections of NMOS and PMOS transitors within the cell. If there is a nedd to use other conducting layers, such as, Metal2, use of such layers must be kept to a minimum. It is desired to use first routing (e.g. Metal1) layer for standard cell ports.

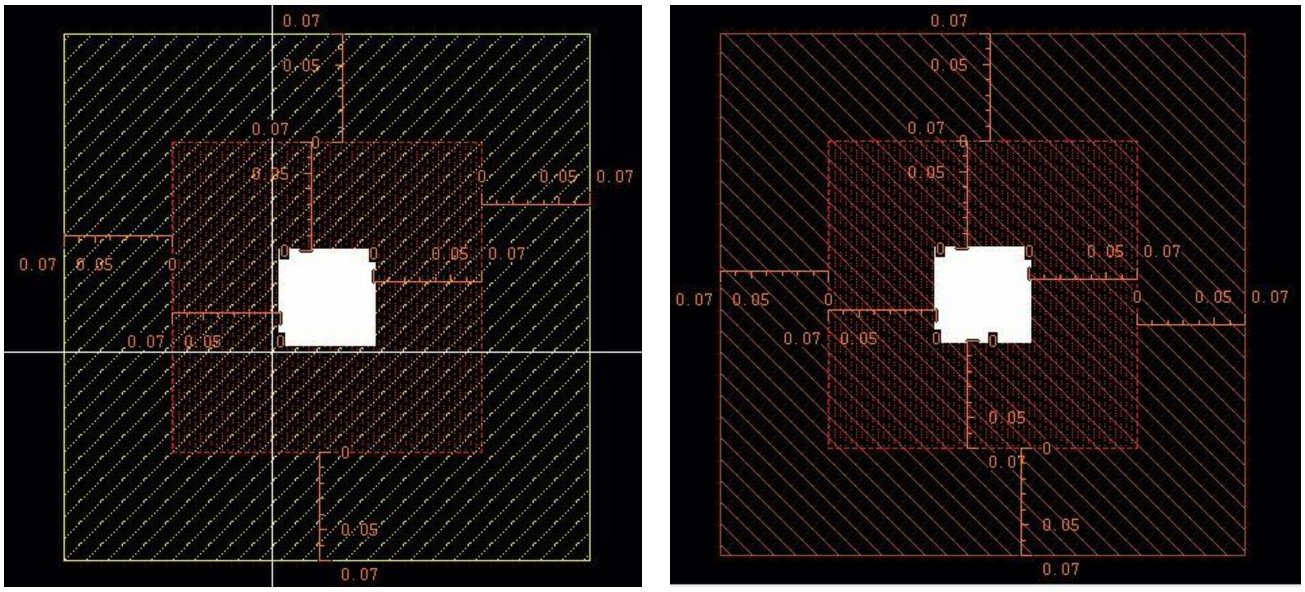
Our cells will have a height of **0.32 μm**. Place the two parts (NMOS and PMOS) **1 μm** apart, and create a ruler so that the cell height can be checked whenever needed and the separation between the NMOS and PMOS can be maintained properly. Now, the layout will look like the following:


1.955
1.5
1
0.5
0
0
0.145
0
0.145

14. Next, draw a '**Poly (drw)**' path by selecting '**Poly**' layer from the '**Layers**' panel and pressing '**p**' to invoke '**create path**' tool. This layer must be of **0.045 μm** in width and in between the two contacts, extending from the oxide layer by **0.1 μm** (at least, on both sides). After placing the '**poly**' gate, the layout will look like the following one:

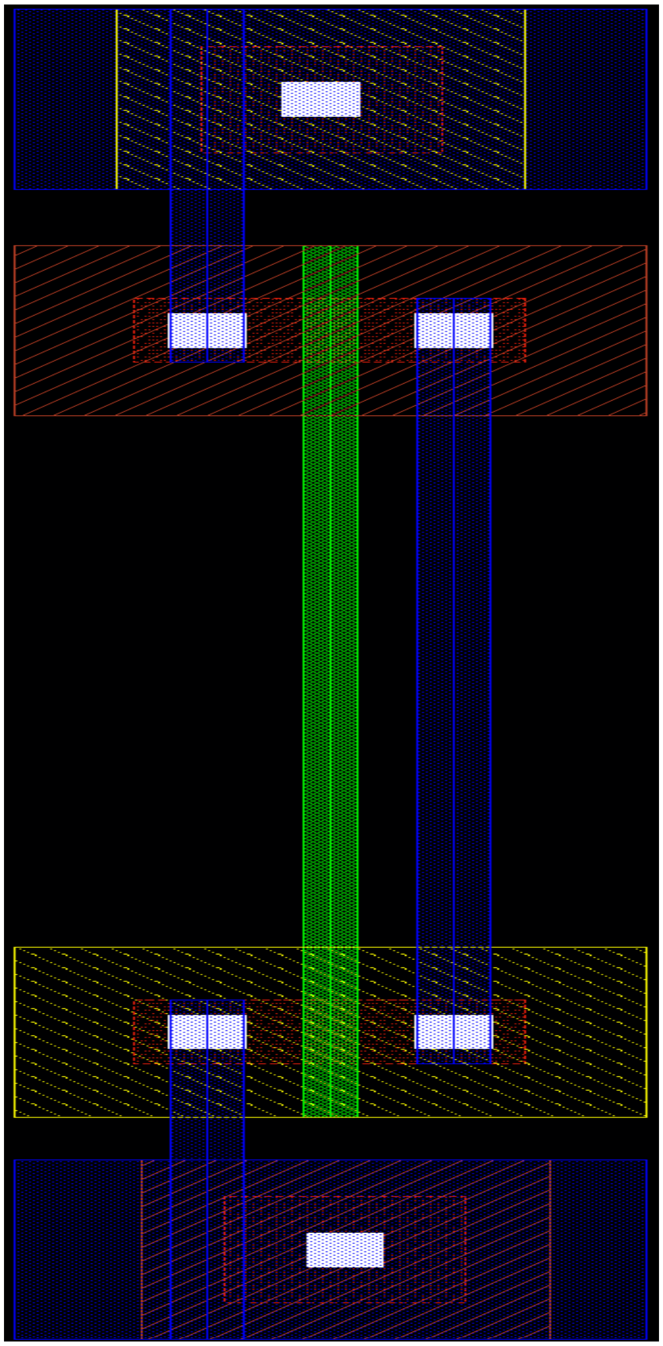


15. Now that you know most of the shortcuts and layers, draw contact for body terminals for NMOS and PMOS. These portions should consist of Cont, Oxide and Nimp (for body of PMOS) or Pimp (for body of NMOS). Check DRD notifications for design rule violations. The following figure shows a Psubstrate and an Nwell contact.

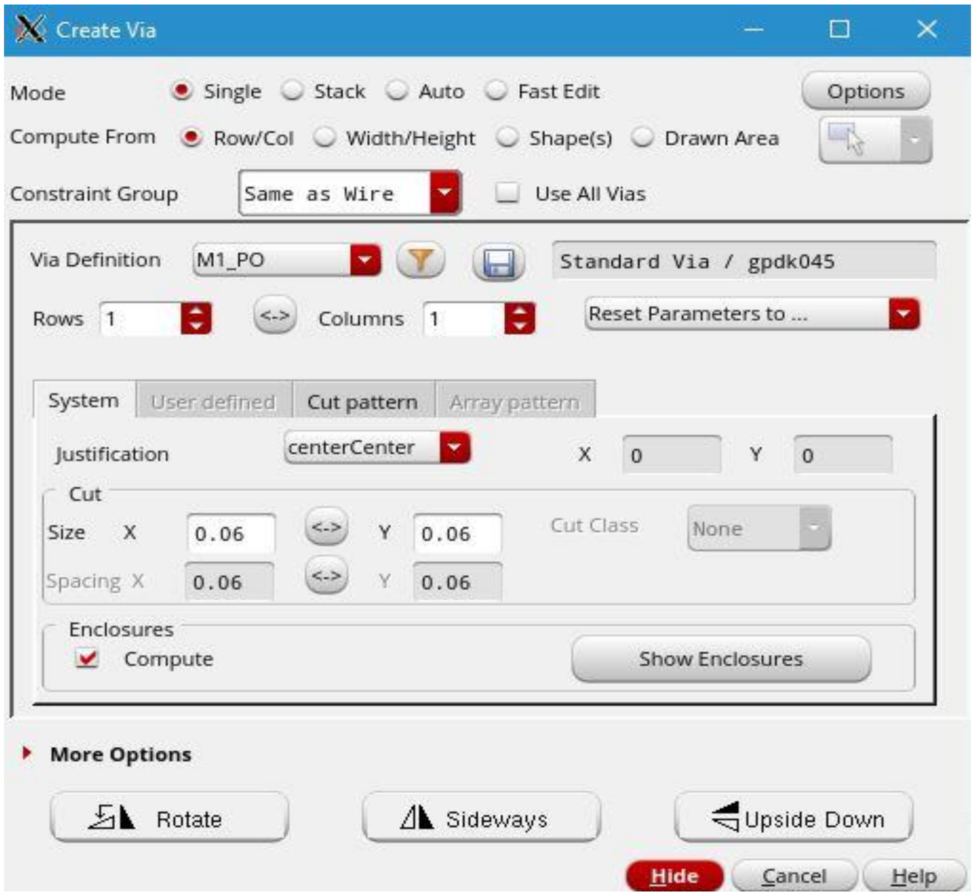


16. Connect the Drain regions of the NMOS and PMOS. Also connect the source of both MOS's to respective body terminals using '**Metal1 (drw)**' layer. Connect the drains of the MOS's using '**Metal1 (drw)**' layer.

17. PMOS should be in '**Nwell (drw)**'. So draw an '**Nwell (drw)**' rectangle surrounding both the PMOS and the body contact for PMOS. The layout will look like the following:



18. Now, we have to place pins. The gate is in ‘**poly (drw)**’ layer. Let’s bring it to ‘**Metal1(drw)**’ layer by extending the ‘**Poly (drw)**’ layer, creating a contact between ‘**Poly**’ and ‘**Metal1**’ layer by pressing ‘**o**’ to ‘**create via**’ and selecting ‘**M1_POv**’ under ‘**via definition**’ and placing it on layout.



Mode [✓ Single] Stack Auto Fast Edit

Compute From [✓ Row/Col] Width/Height Shape(s) Drawn Area

Constraint Group: Same as Wire; Use All Vias ☐

Via Definition: M1_PO; Standard Via / gpdk045

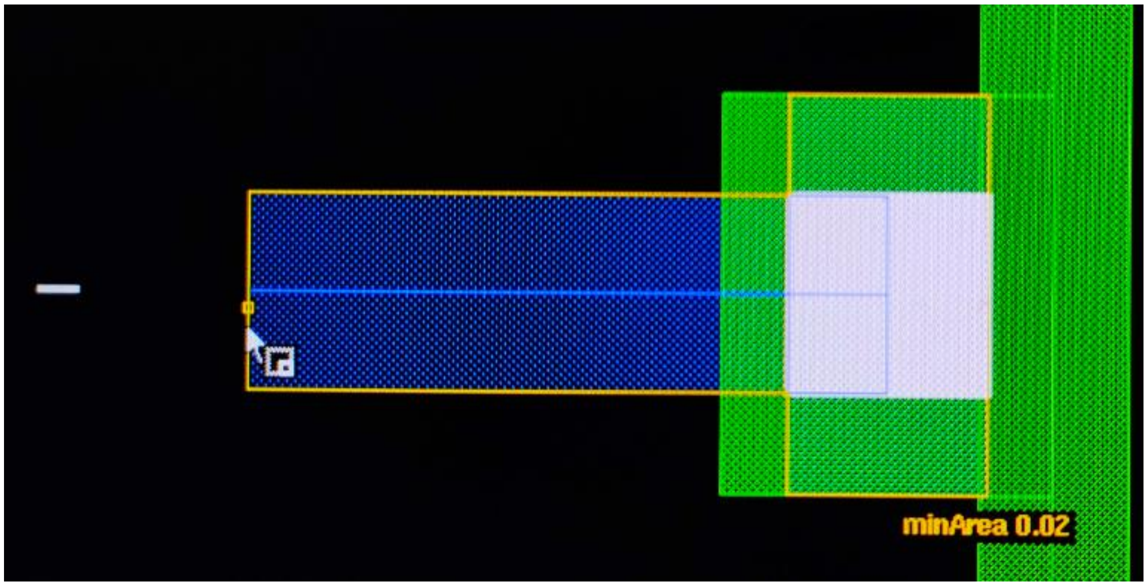
Rows: 1; Columns: 1

Justification: centerCenter; X: 0; Y: 0

Cut: Size X 0.06, Y 0.06; Spacing X 0.06, Y 0.06; Cut Class: None

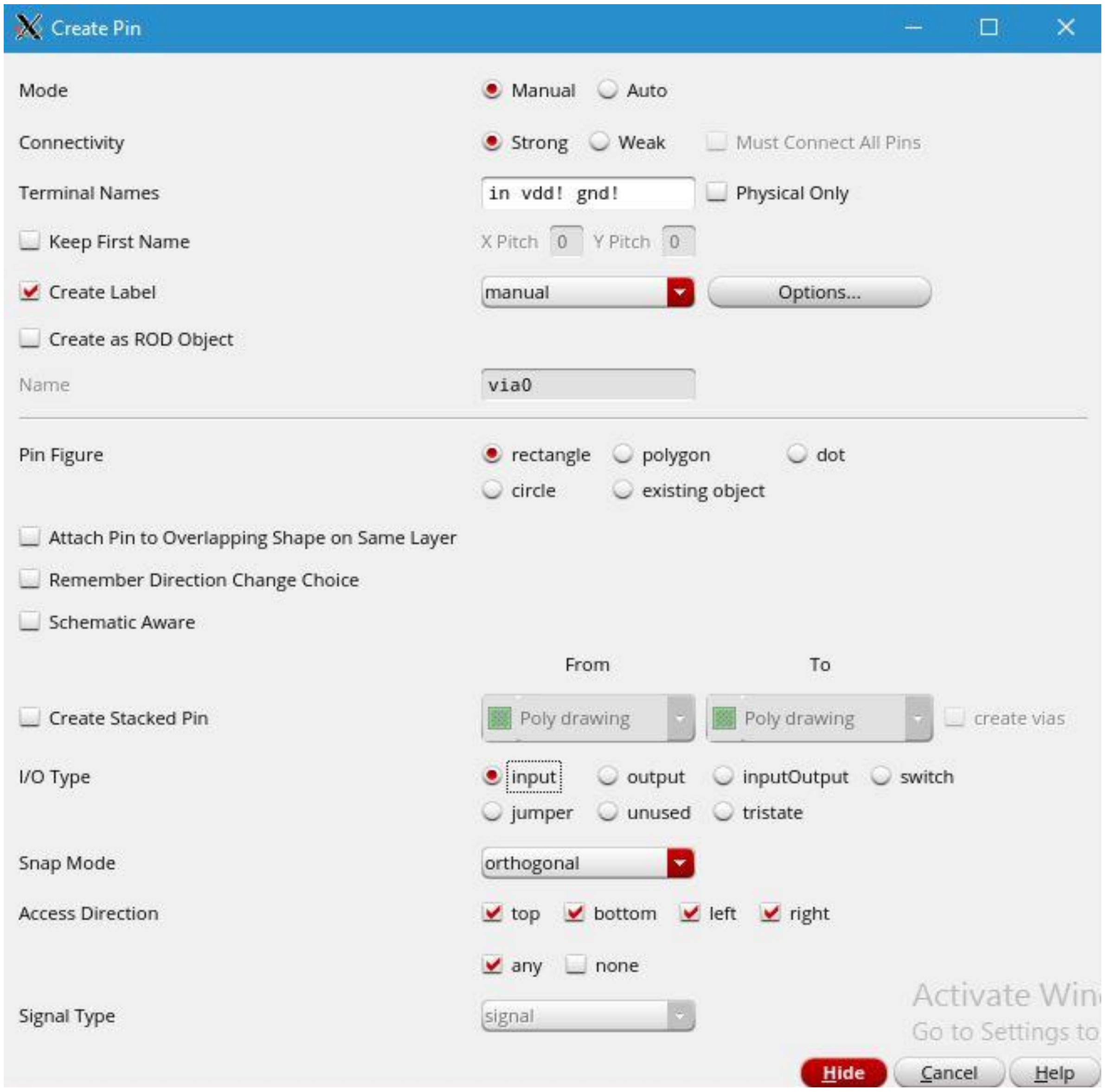
Enclosures: Compute ✓

19. Also draw a ‘**Metal1 (drw)**’ rectangle on the via, because the default **Metal1** rectangle area is less than the required minimum.



20. Now, Execute ***CreatePin*** to create pins for **vdd!**, **gnd!**, **in** and **out**.

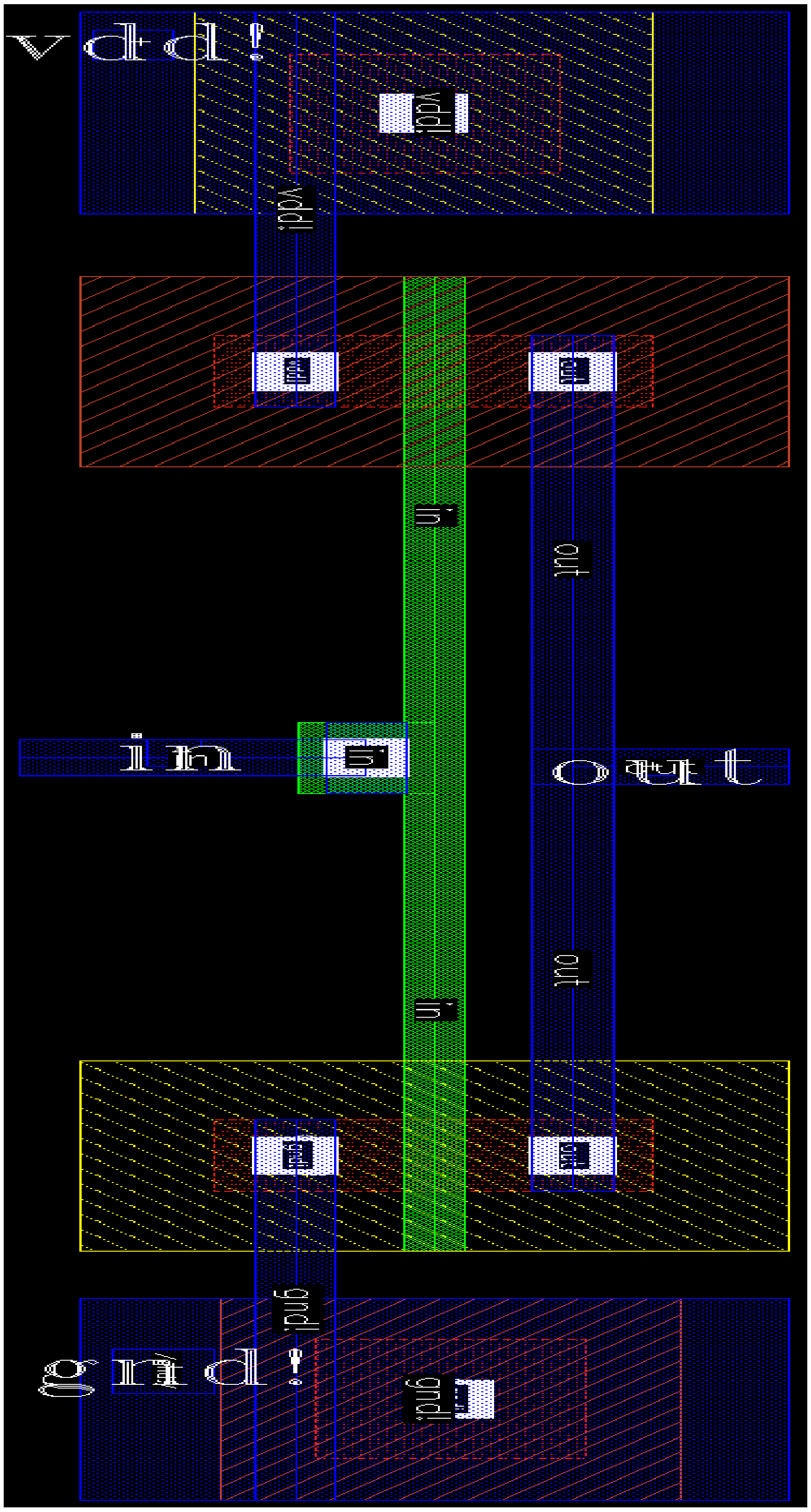
For **in**, **vdd!** and **gnd!** select ‘**input**’ as **‘I/O type’** and for **out** select ‘**output**’ as **‘I/O type’**. Now, draw rectangles on the **Poly-Metal1** via for ‘**in**’ pin, PMOS source-to-body ‘**Metal1**’ connection for ‘**vdd!**’ pin and NMOS source-to-body connection for ‘**gnd!**’ pin. For ‘**out**’ pin, draw the rectangle on the **Metal1** layer connecting the two drains of MOS’s.



Mode: [✓ Manual] Auto
Connectivity: [✓ Strong] Weak; Must Connect All Pins
Terminal Names: in vdd! gnd!; Physical Only
Keep First Name; X Pitch 0 Y Pitch 0
[✓ Create Label]: manual; Options...
Create as ROD Object
Name: via0
Pin Figure: [✓ rectangle] polygon dot circle existing object
Attach Pin to Overlapping Shape on Same Layer
Remember Direction Change Choice
Schematic Aware
Create Stacked Pin: From Poly drawing, To Poly drawing; create vias
I/O Type: [✓ input] output inputOutput switch jumper unused tristate
Snap Mode: orthogonal
Access Direction: [✓ top] [✓ bottom] [✓ left] [✓ right] [✓ any] none
Signal Type: signal

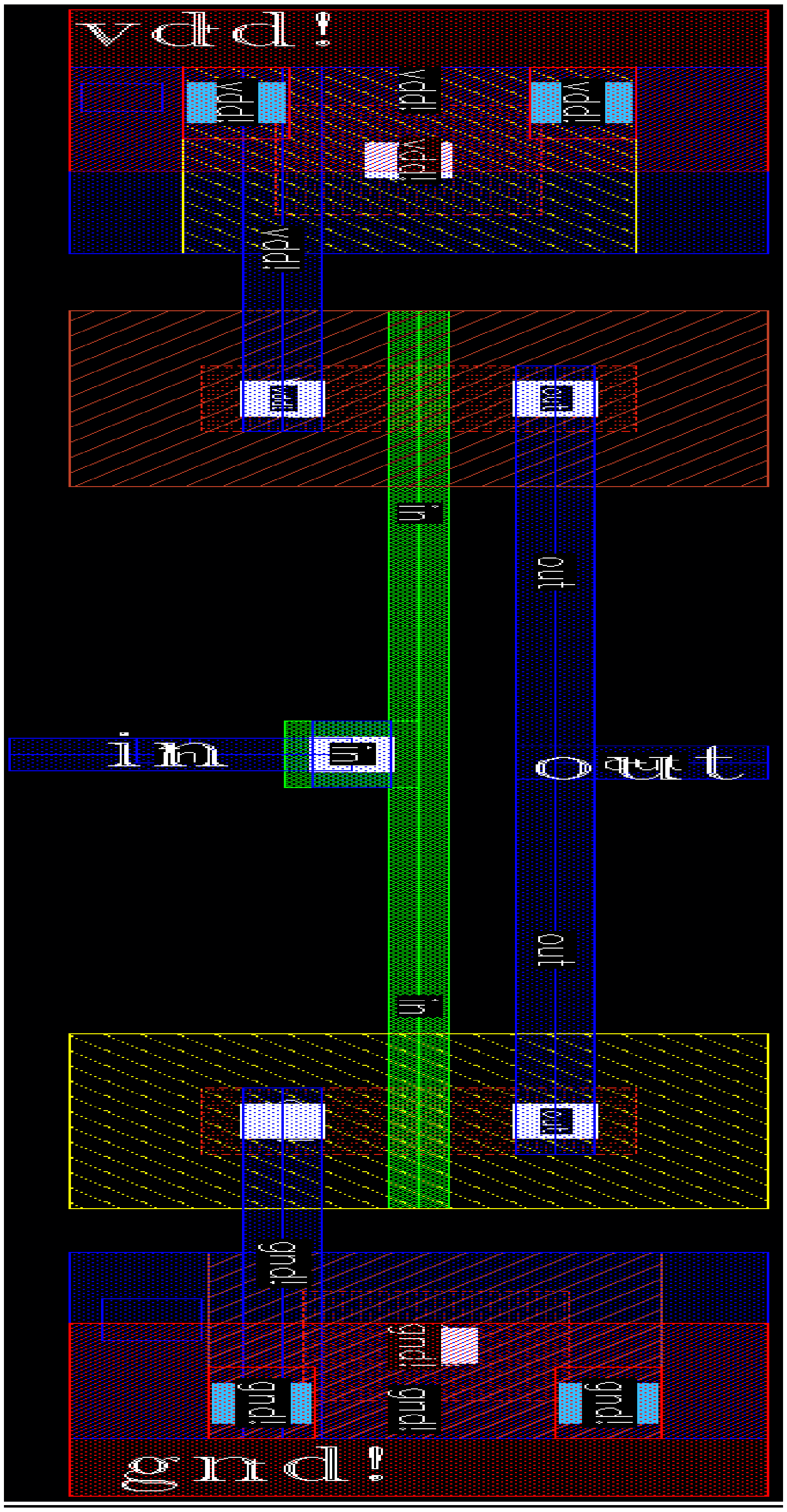
You may add label to pins.

Your Layout will look like this.



21. Finally, add **Metal2** paths of **0.5** μ width for power rails and connect them to power nets in **Metal1** by using **Metal1 to Metal2** via by invoking '**create via**'.

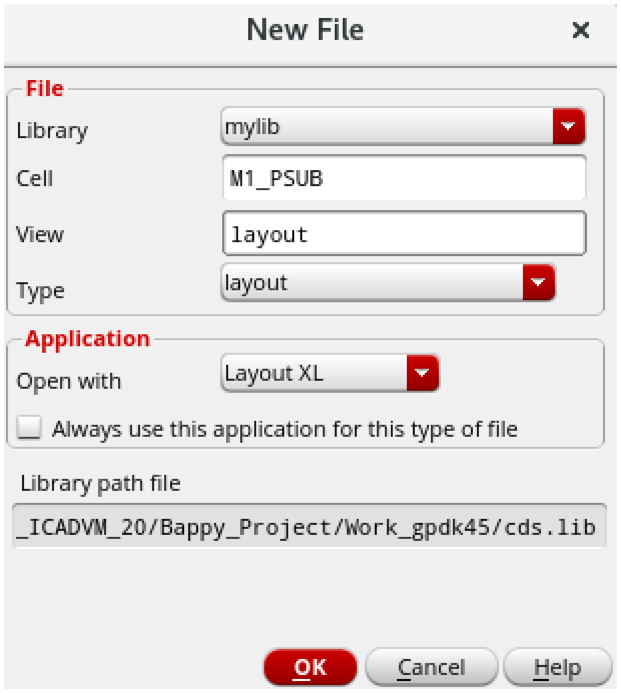
The final layout will look like the following:



Creating Body ties

Now you know what body tie is. We will now make two instances for body ties one for **Psub** and one for **Nwell**.

1. Execute ***File→New→Cellview*** and fill in the New File form as follows. Click **OK**.



New File

File

Library	mylib
Cell	M1_PSUB
View	layout
Type	layout

Application

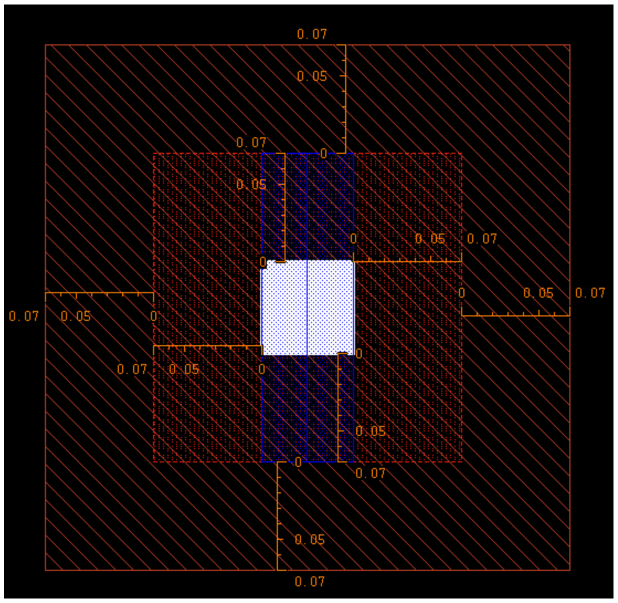
Open with	Layout XL

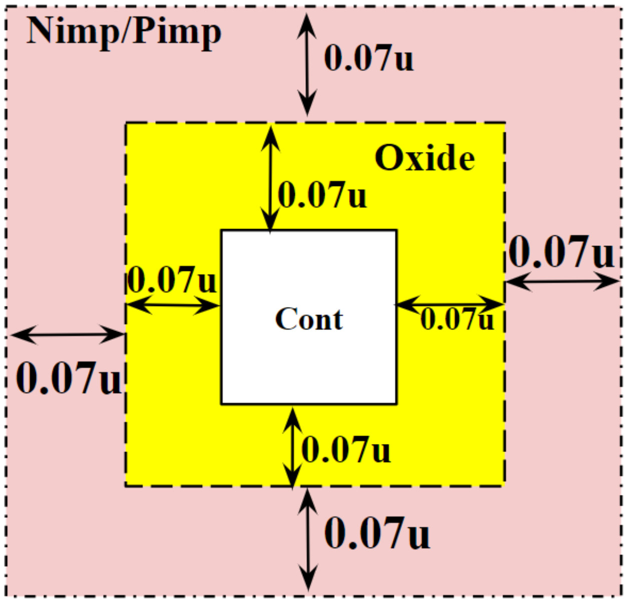
☐ Always use this application for this type of file

Library path file

_ICADVM_20/Bappy_Project/Work_gpdk45/cds.lib

2. Draw psubstrate contact in the same way as you have made it in Lab3.





3. Save it and make nwell contact similarly (name it **M1_NWELL**), just change the **Pimp** layer to **Nimp** and everything else is the same.

Save these two for later use.

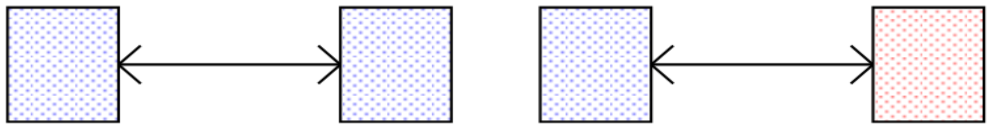
Appendix A (Shortcut keys for Cadence Virtuoso ® Layout Editor L)

Shortcut Key	Tasks performed
f	Fit display to window
r	Draw rectangle
q	Edit property of an object
p	Makes a min width path of the layer selected in LSW
Ctrl+a	Select all
Ctrl+d	Deselect all
c	Copy
m	Move
s	Stretch side of a rectangle
k	Invoke ruler tool
Shift+k	Delete all rulers
i	Add an instance
u	Undo
Shift+u	Redo
e	Display options
o	Add via between layers
l	Create a label

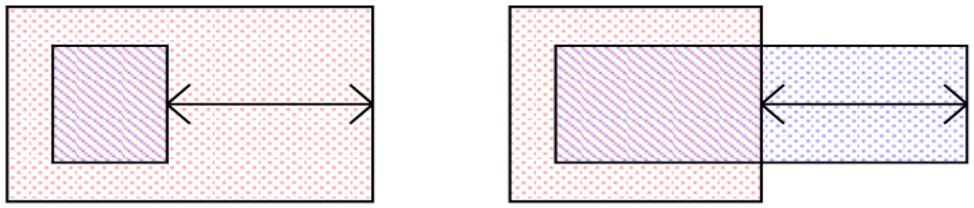
Appendix B (gpdk090 Design Rules Guide (Abridged Version for VLSI-I Lab))

Terminology Definitions

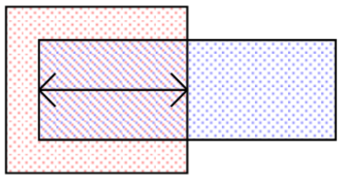
Spacing - distance from the outside of the edge of a shape to the outside of theedge of another shape.



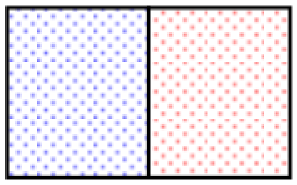
Enclosure - distance from the inside of the edge of a shape to the outside of theedge of another shape.



Overlap - distance from the inside of the edge of a shape to the inside of the edgeof another shape.



Butting - outside of the edge of a shape touching the outside of the edge of anothershape.



Lab-8: Post Layout Verification

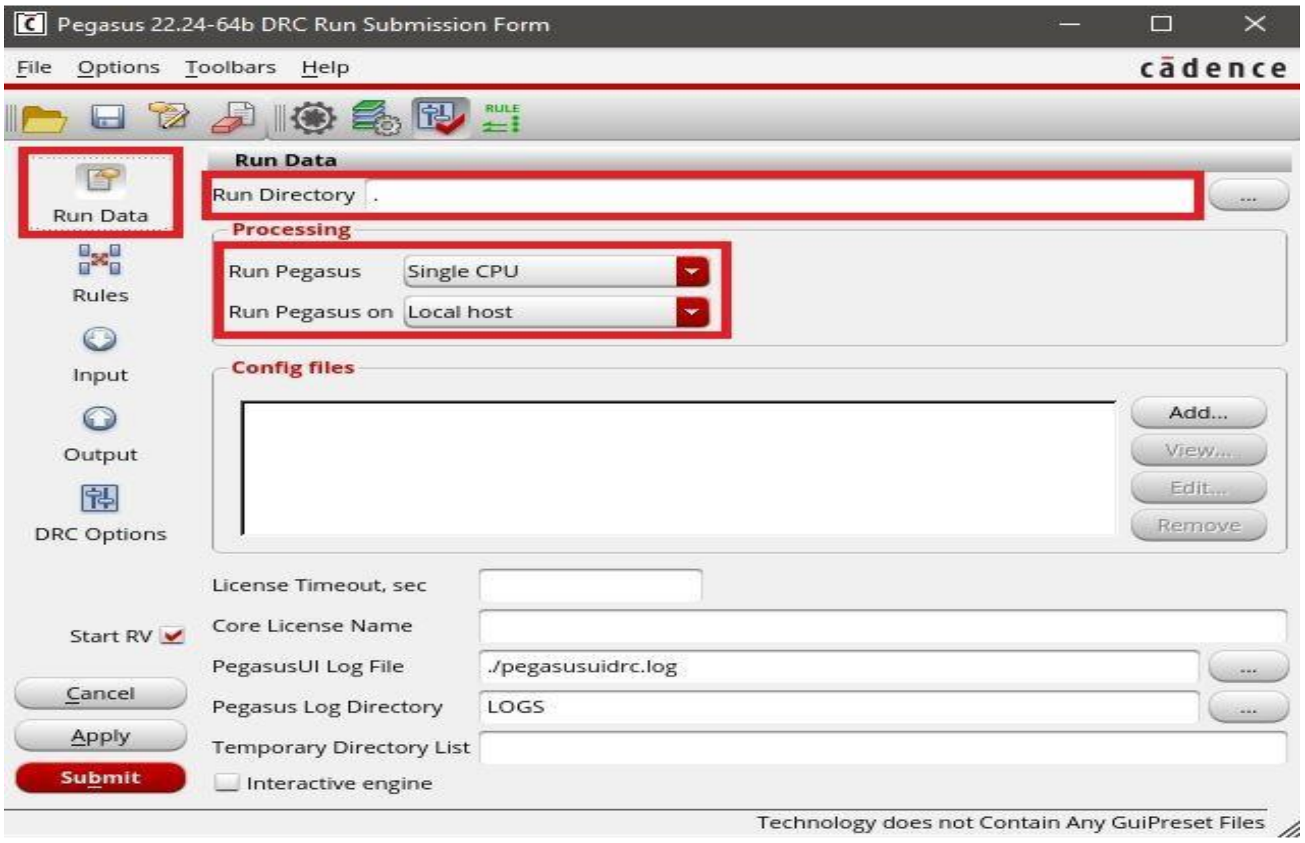
Objectives:

- To perform Design rules check (DRC), Layout vs. Schematic check (LVS) of inverter layout
- To extract parasitic resistance and capacitance from layout of designed inverter
- To perform transient simulation of extracted view
- To create layout views for body ties of NMOS and PMOS for further use

Design Rule Check (DRC)

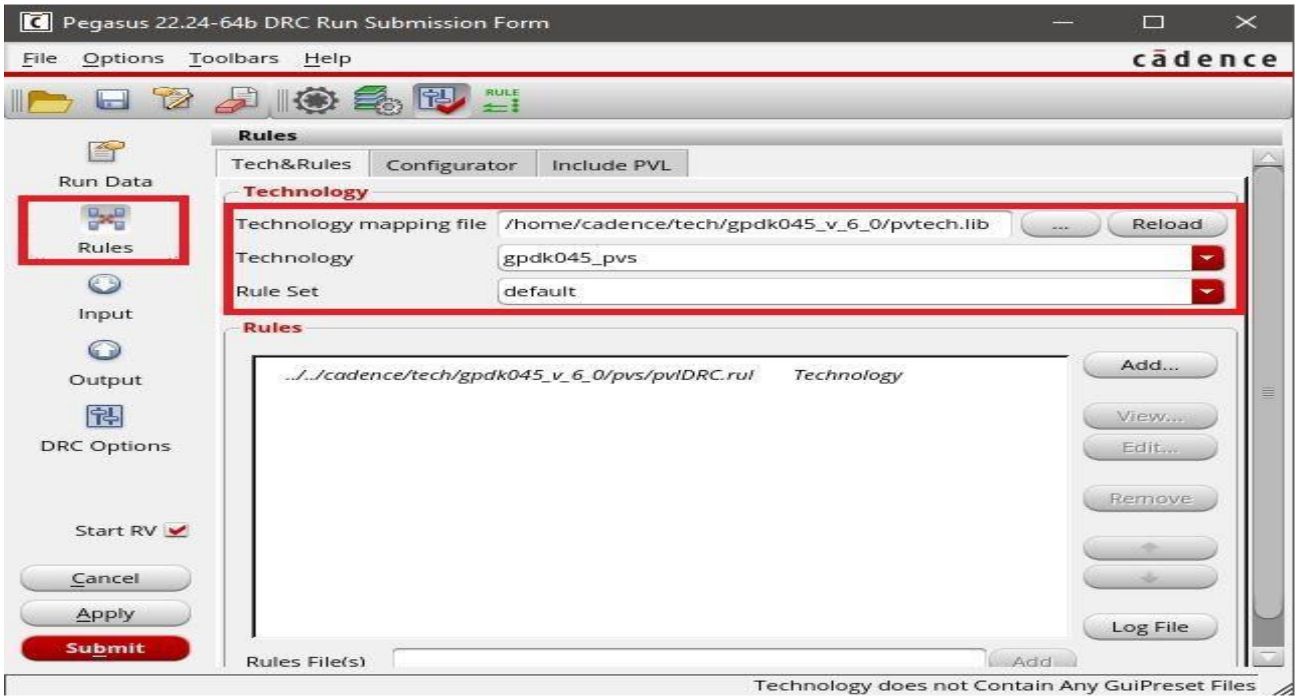
1. Now we would like to check the DRC rules by Pegasus. Execute **Pegasus DRC**

The following **DRC Run Submission Form** will appear. In the **Run Data** section, put dot(.) in the **Run Directory** and **Run Pegasus** in **Single CPU** and on **Local host** as shown below.



Run Directory .
Run Pegasus: Single CPU
Run Pegasus on: Local host
License Timeout, sec
Core License Name
PegasusUI Log File: ./pegasusuidrc.log
Pegasus Log Directory: LOGS
Temporary Directory List
Start RV ✓
Interactive engine ☐
Technology does not Contain Any GuiPreset Files

2.Now in the **Rules** section select the **Technology mapping file**, **Technology** and **Rule Set** as shown below.


Pegasus 22.24-64b DRC Run Submission Form

Rules – Tech&Rules

Technology

Technology mapping file: /home/cadence/tech/gpdk045_v_6_0/pvtech.lib

Technology: gpdk045_pvs

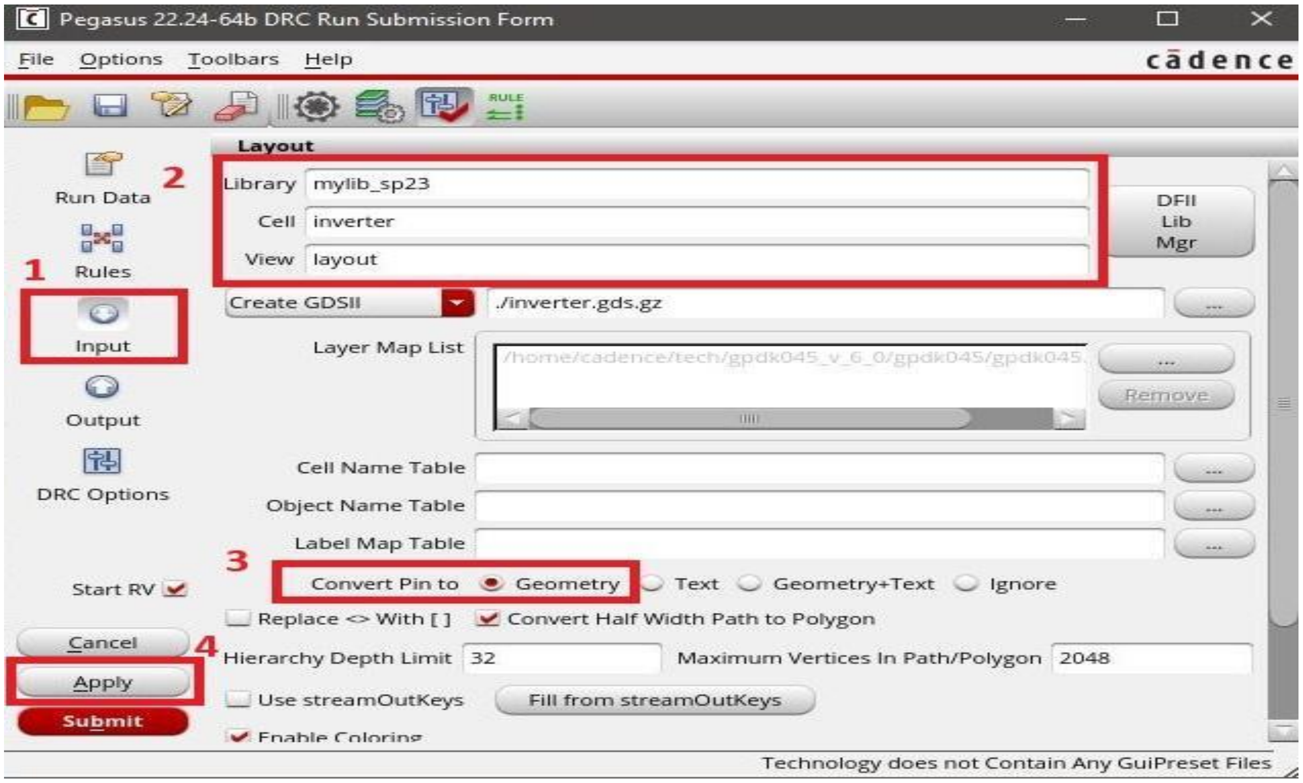
Rule Set: default

Rules

../../cadence/tech/gpdk045_v_6_0/pvs/pvlDRC.rul   Technology

Technology does not Contain Any GuiPreset Files

4. In the Input section select your **Library, Cell,** and **View**. Also, select **Convert Pin to Geometry.** Then click **Apply.**


Pegasus 22.24-64b DRC Run Submission Form

Layout

Library: mylib_sp23

Cell: inverter

View: layout

Create GDSII: ./inverter.gds.gz

Layer Map List: /home/cadence/tech/gpdk045_v_6_0/gpdk045/gpdk045

Cell Name Table:

Object Name Table:

Label Map Table:

Convert Pin to [✓ Geometry] Text Geometry+Text Ignore

Replace <> With [ ]

[✓] Convert Half Width Path to Polygon

Hierarchy Depth Limit: 32

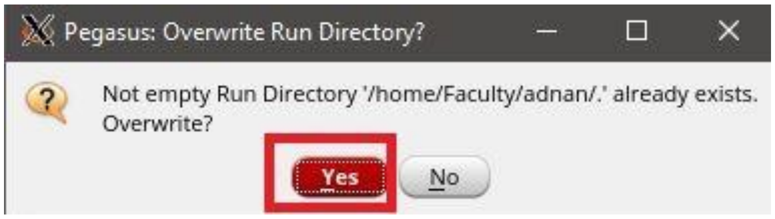
Maximum Vertices In Path/Polygon: 2048

Use streamOutKeys

[✓] Enable Coloring

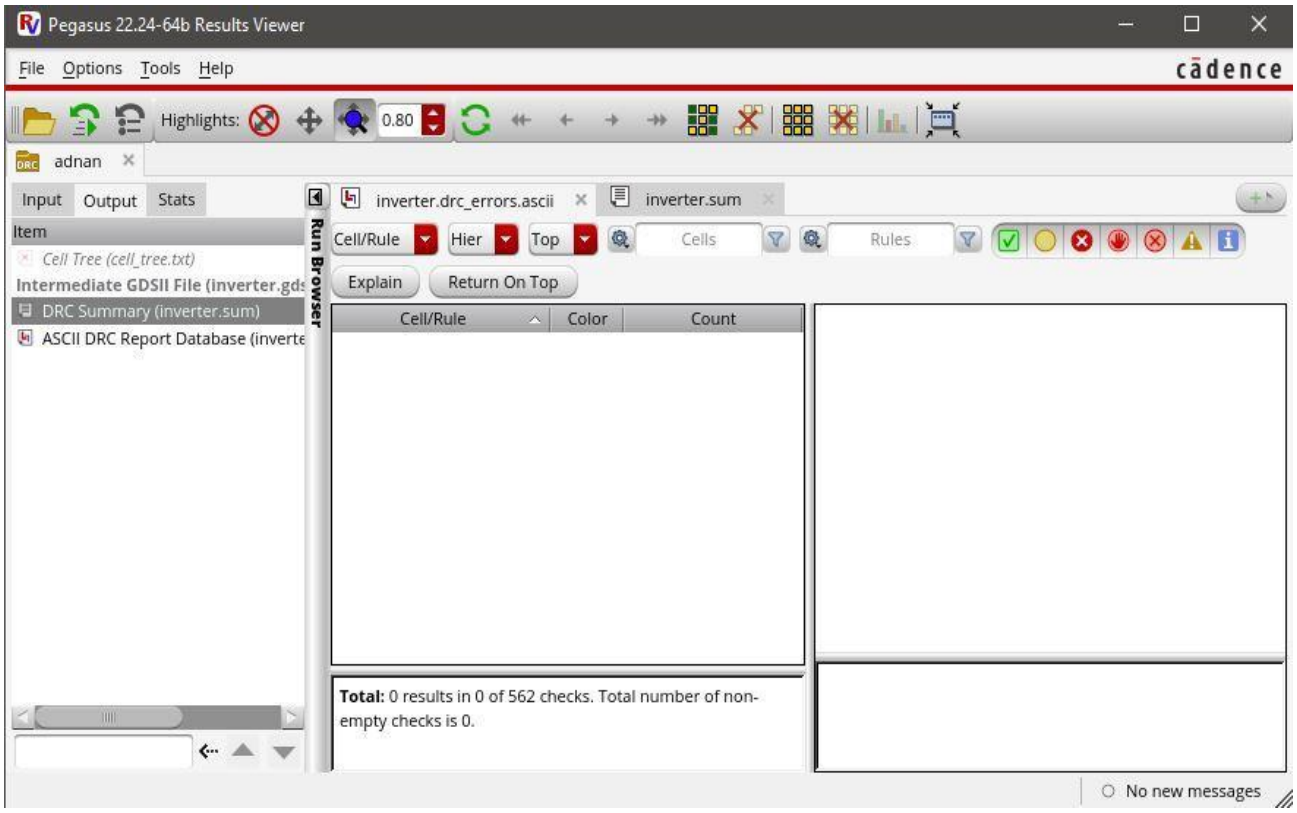
Technology does not Contain Any GuiPreset Files

5. Click **Yes** if the following window appears.


Pegasus: Overwrite Run Directory?

Not empty Run Directory '/home/Faculty/adnan/.' already exists. Overwrite?

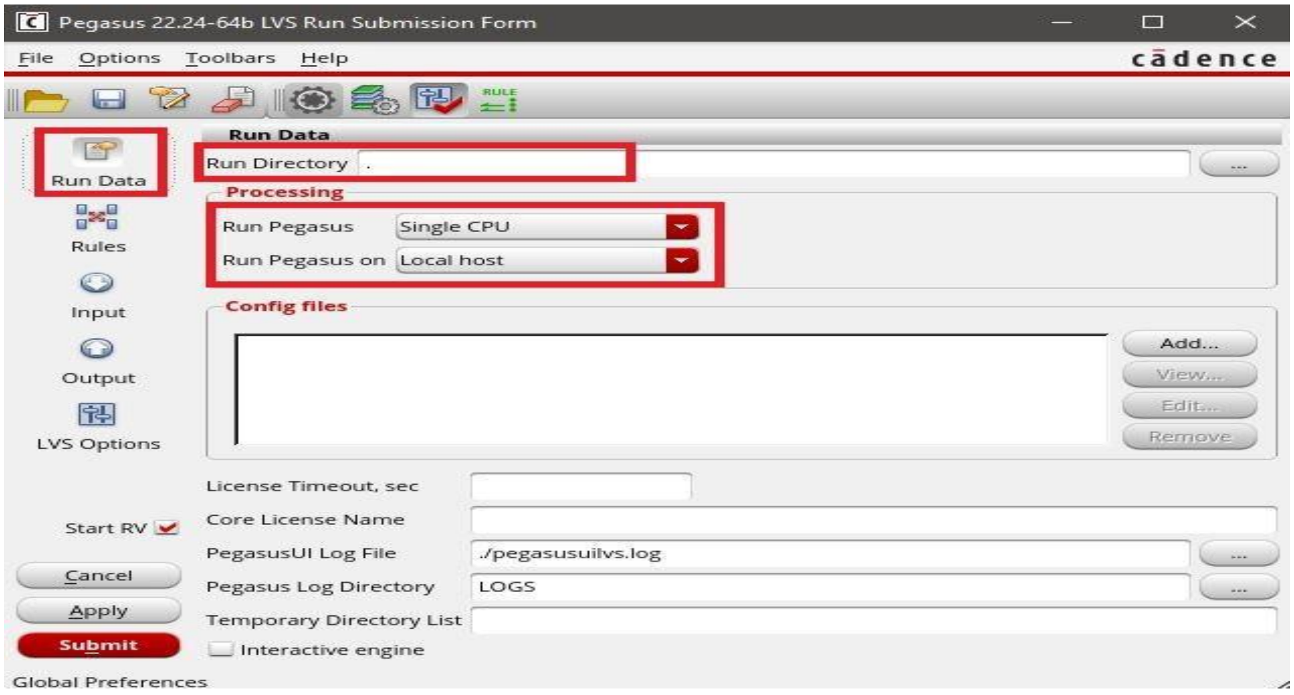
6.The following **Results Viewer** will appear. The **DRC Summary** will be empty if there is no error.


Pegasus 22.24-64b Results Viewer
File
Options
Tools
Help
cadence
Highlights:
0.80
adnan
Input
Output
Stats
Item
Cell Tree (cell_tree.txt)
Intermediate GDSII File (inverter.gds
DRC Summary (inverter.sum)
ASCII DRC Report Database (inverte
Run Browser
inverter.drc_errors.ascii
inverter.sum
Cell/Rule
Hier
Top
Cells
Rules
Explain
Return On Top
Cell/Rule
Color
Count
No new messages

Total: 0 results in 0 of 562 checks. Total number of non-empty checks is 0.

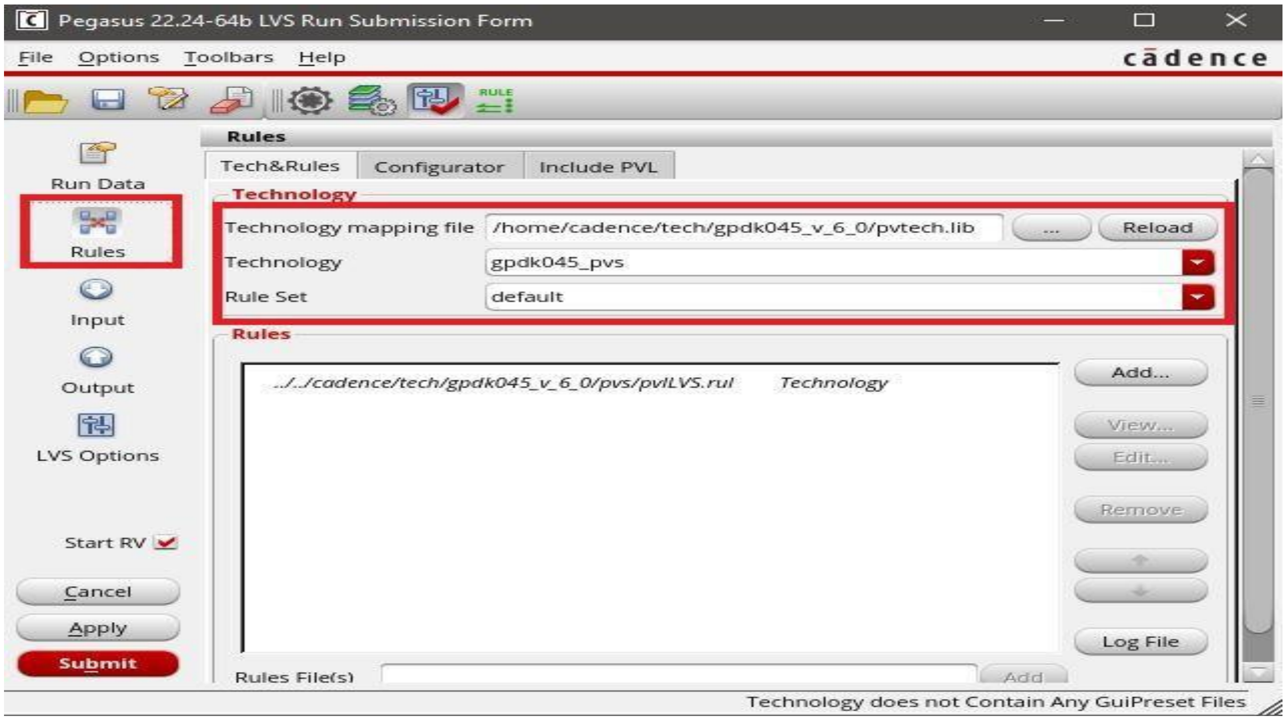
Layout Versus Schematic

1. Execute **Pegasus → LVS**
2. The following **LVS Run Submission Form** will appear. In the **Run Data** section put dot (.) in the **Run Directory** and **Run Pegasus** in **Single CPU** and on **Local host** as shown below.



Run Directory: .
Run Pegasus: Single CPU
Run Pegasus on: Local host
License Timeout, sec:
Core License Name:
PegasusUI Log File: ./pegasusuilvs.log
Pegasus Log Directory: LOGS
Temporary Directory List:
Interactive engine

3. Now in the **Rules** section select the Technology mapping file, Technology and Rule Set as shown below.



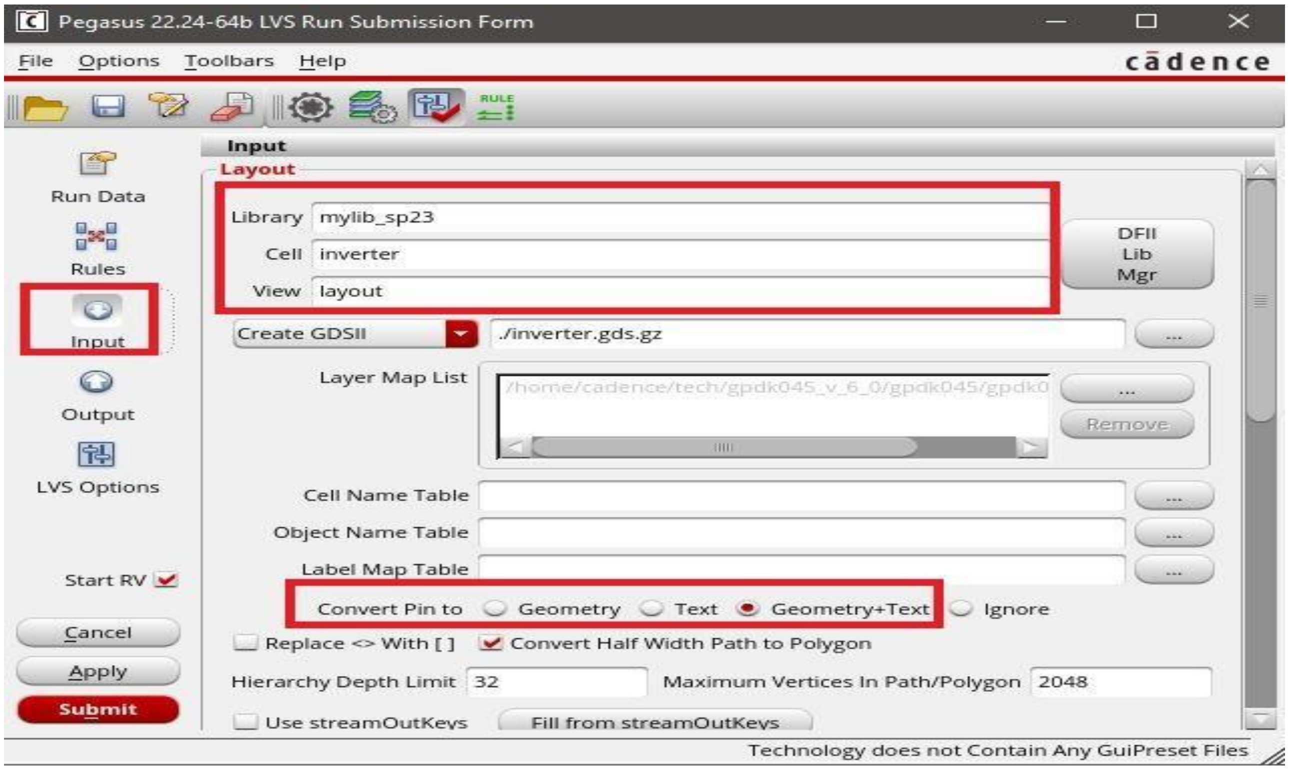
Technology mapping file: /home/cadence/tech/gpdk045_v_6_0/pvtech.lib
Technology: gpdk045_pvs
Rule Set: default

../../cadence/tech/gpdk045_v_6_0/pvs/pvlLVS.rul *Technology*

Rules File(s)

Technology does not Contain Any GuiPreset Files

4. In the Input section select your **Library, Cell,** and **View**. Also, select **Convert Pin to Geometry+Text.**



Pegasus 22.24-64b LVS Run Submission Form

File Options Toolbars Help

Input

Layout

Library: mylib_sp23
Cell: inverter
View: layout

Create GDSII: ./inverter.gds.gz

Layer Map List: /home/cadence/tech/gpdk045_v_6_0/gpdk045/gpdk0

Cell Name Table:
Object Name Table:
Label Map Table:

Convert Pin to: Geometry Text [✓ Geometry+Text] Ignore

Replace <> With [ ]: ☐
Convert Half Width Path to Polygon: ✓

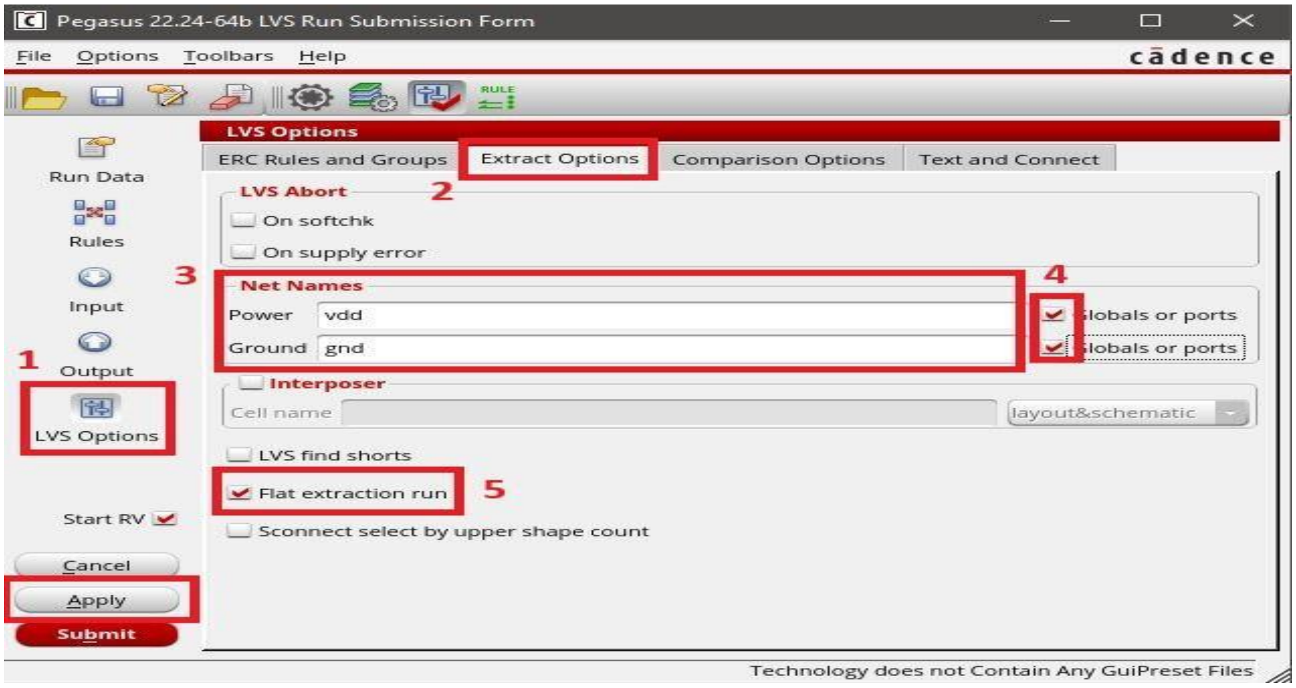
Hierarchy Depth Limit: 32
Maximum Vertices In Path/Polygon: 2048

Use streamOutKeys: ☐

Start RV: ✓

Technology does not Contain Any GuiPreset Files

5. Now in the **LVS Options** section, go to the **Extract Options** tab write the **Net Names** as shown below and **select** the **global or ports** option and **Flat extraction run** as shown below. Then click **Apply.**



Pegasus 22.24-64b LVS Run Submission Form

LVS Options

ERC Rules and Groups | Extract Options | Comparison Options | Text and Connect

LVS Abort
On softchk: ☐
On supply error: ☐

Net Names
Power: vdd — globals or ports ✓
Ground: gnd — globals or ports ✓

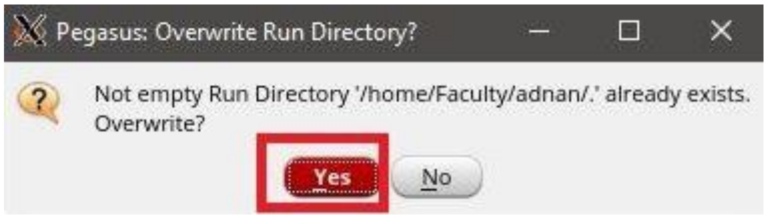
Interposer: ☐
Cell name: layout&schematic

LVS find shorts: ☐
Flat extraction run: ✓
Sconnect select by upper shape count: ☐

Start RV: ✓

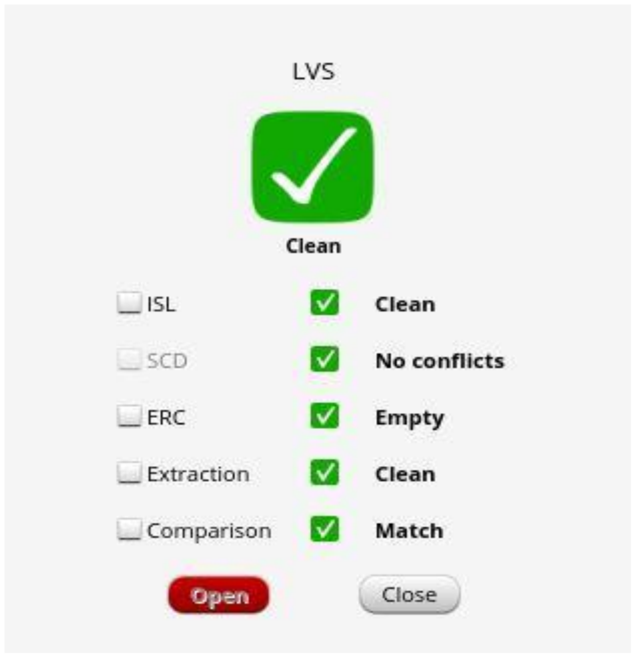
Technology does not Contain Any GuiPreset Files

6. Click **Yes** if the following window appears.


Pegasus: Overwrite Run Directory?

Not empty Run Directory '/home/Faculty/adnan/.' already exists. Overwrite?

7. If the layout matches with schematic the following window will appear.


LVS

Clean

ISL	✓	Clean
SCD	✓	No conflicts
ERC	✓	Empty
Extraction	✓	Clean
Comparison	✓	Match

Lab-9: Schematic Driven Layout Design

Objectives:

- To be familiar with schematic-driven layout with the example of a 2-input NAND gate.
- To perform Schematic Level Verification, Layout Design, DRC and LVS check and perform post-layout simulation from extracted view.

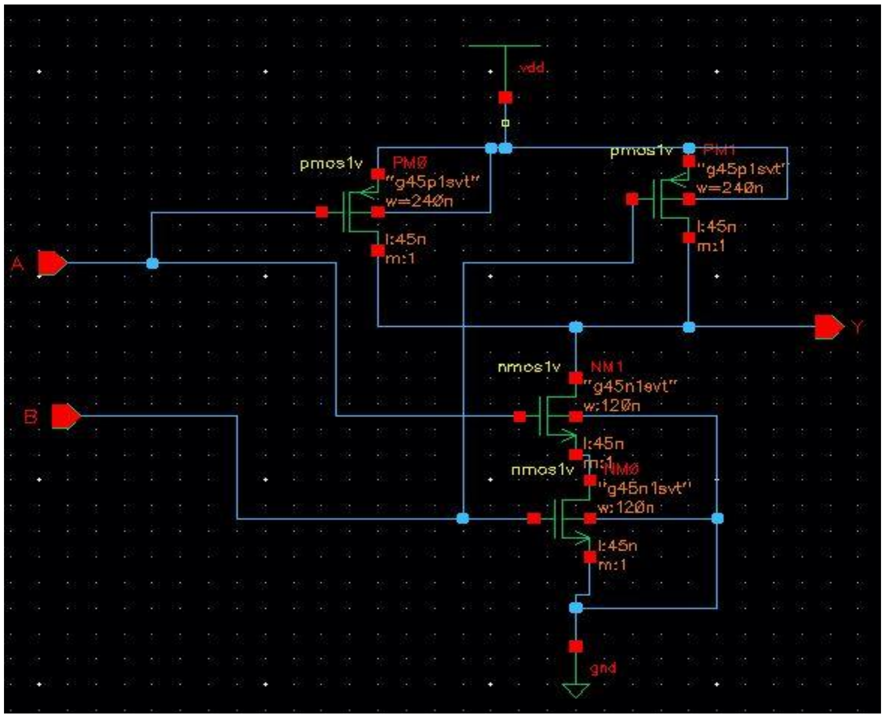
Creating Layout using Virtuoso Layout Editor XL

1. Virtuoso Layout Editor XL is a schematic-driven layout generation tool. To learn schematic driven layout, we will create the schematic view of a 2-input NAND gate cell which we named ***NAND2X1***.

2. Instantiate the following cells to your schematic.

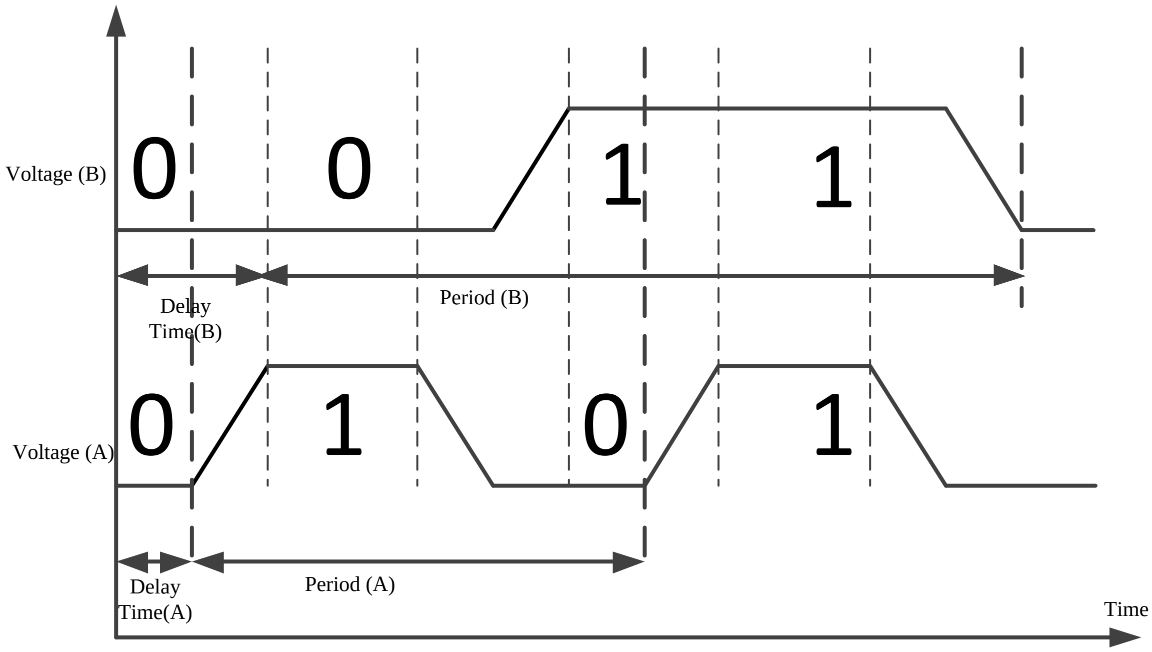
Library Name	Cell Name	Properties/Comment
gpdk090	*nmos1v*	For NM0 and NM1, Width = 240n
gpdk090	*pmos1v*	For PM0 and PM1, Width = 480n
analogLib	*vdd*	
analogLib	*gnd*	

Use your experience from Lab1 to draw the schematic diagram of the nand gate.

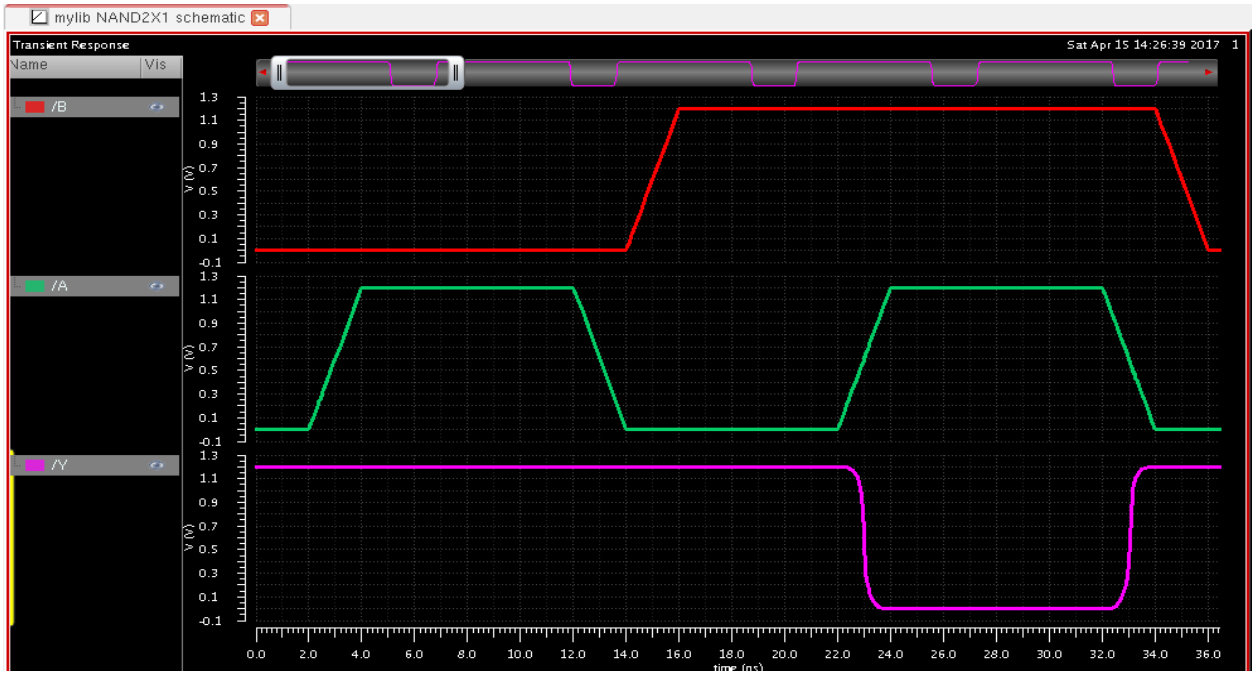


3. Launch **ADE L** and simulate the design to verify its functionality. Setup **Model library**, **Analysis** type and **Outputs to be plotted** as you have done in previous lab.

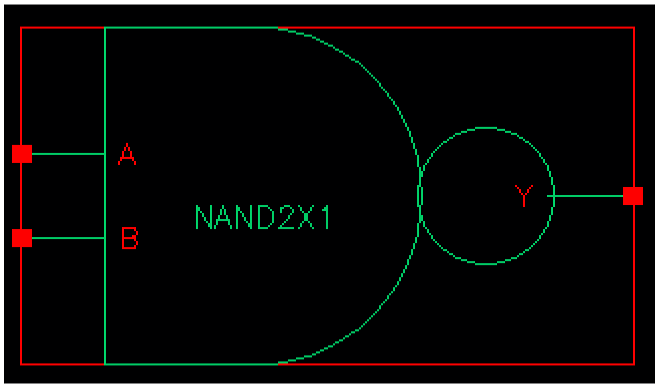
While setting inputs for signals A and B, you have to use different periods and delays for the two signals, so that you can observe all four cases (00, 01, 10, 11) of input signals. Also make sure 0→1 and/or 1→0 transitions for both input signals do not occur at the same time. The following figure shows a sample of two signals meeting these criteria:



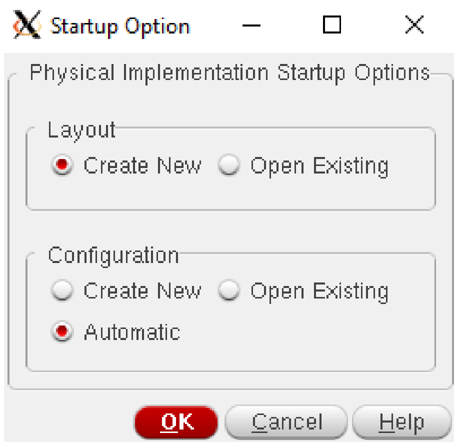
A sample waveform window would look like the following after simulation:



4. Then create a symbol in the same way you have made symbol of inverter in lab2. Make it look like a nand gate.



5. In schematic editor window, execute: ***Launch→Layout XL***. The following window will appear:

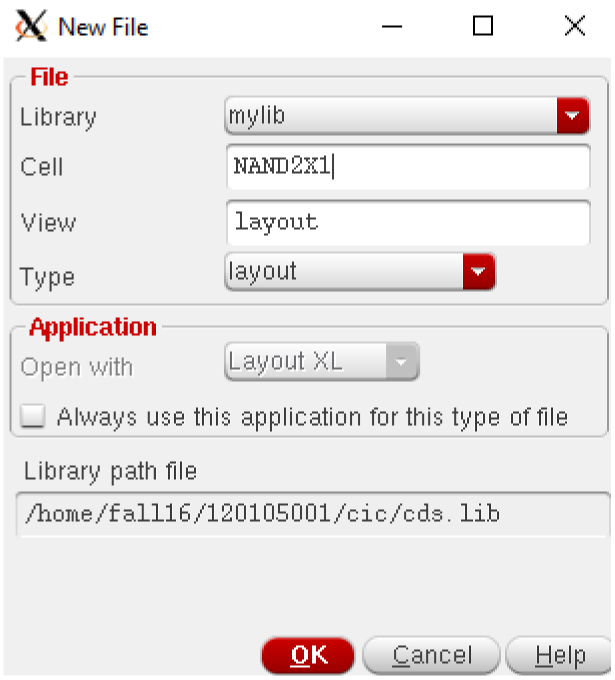


Physical Implementation Startup Options

Layout
[✓ Create New] Open Existing

Configuration
Create New Open Existing [✓ Automatic]

6. Click **OK**. '**New File**' window for layout will appear. Click **OK**.



File

Library	mylib
Cell	NAND2X1
View	layout
Type	layout

Application

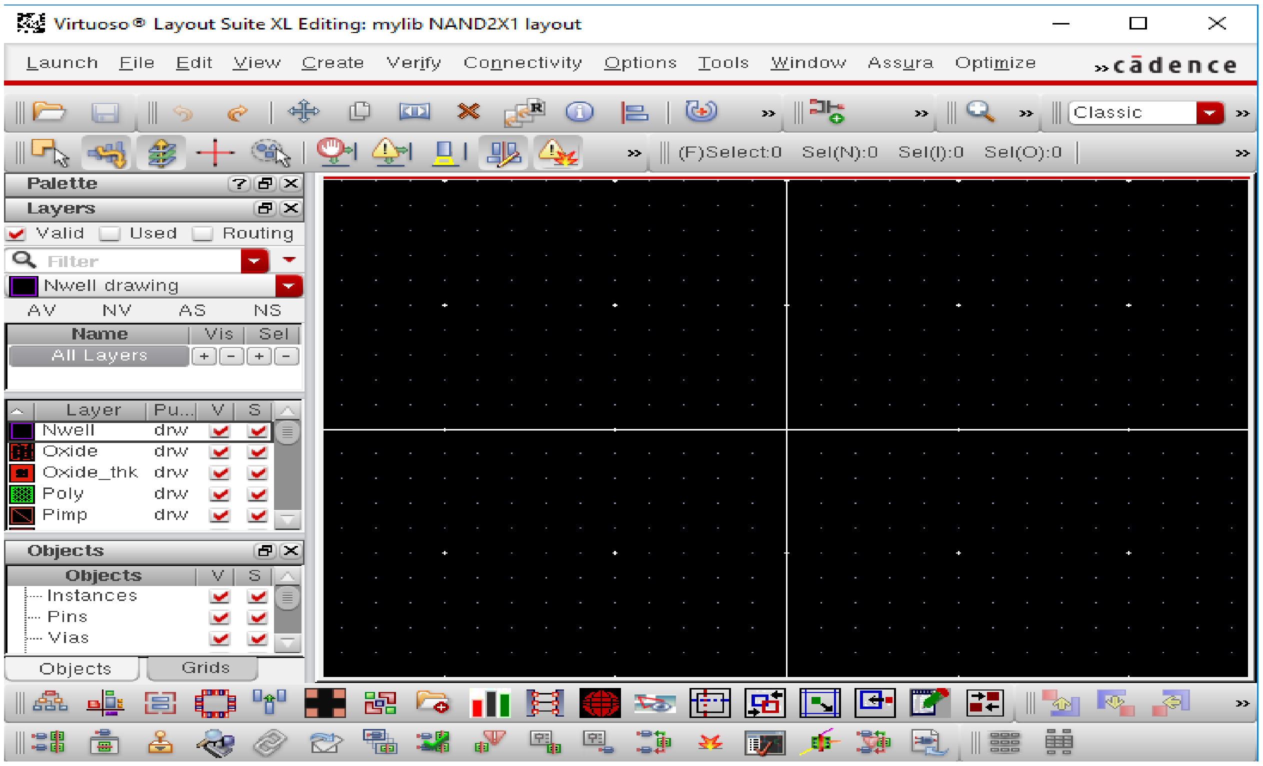
Open with: Layout XL

Always use this application for this type of file

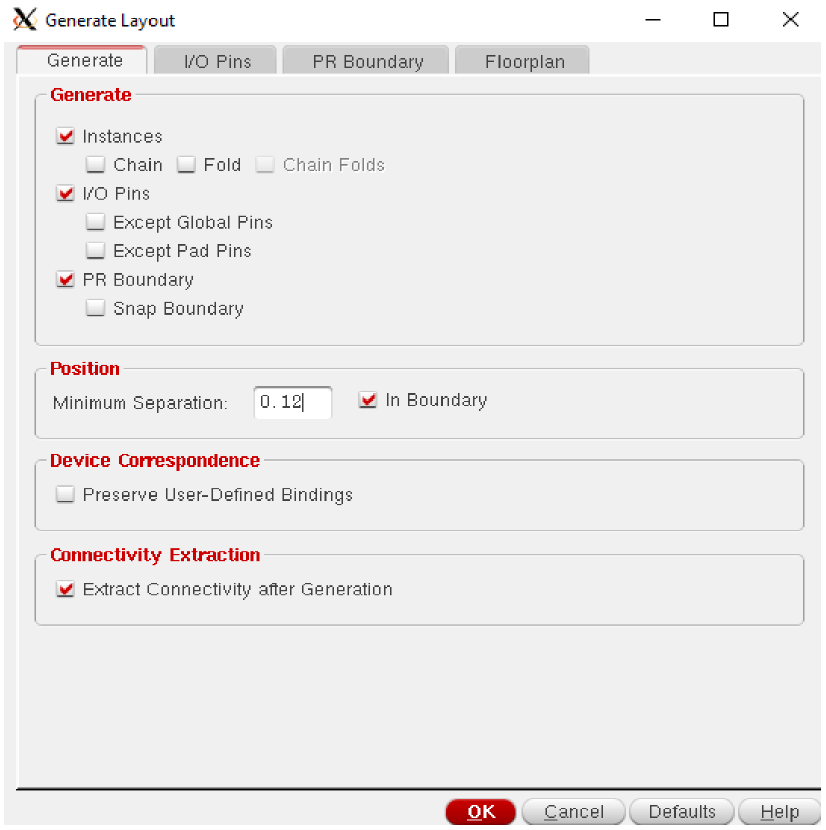
Library path file

/home/fall16/120105001/cic/cds.lib

'Virtuoso Layout Editor XL' window will appear.



7. Execute ***Connectivity→ Generate All From Source.*** The following pop-up window will appear:


Generate Layout

Generate | I/O Pins | PR Boundary | Floorplan

Generate

- [✓] Instances
  - [ ] Chain
  - [ ] Fold
  - [ ] Chain Folds
- [✓] I/O Pins
  - [ ] Except Global Pins
  - [ ] Except Pad Pins
- [✓] PR Boundary
  - [ ] Snap Boundary

Position

Minimum Separation: 0.12 [✓] In Boundary

Device Correspondence

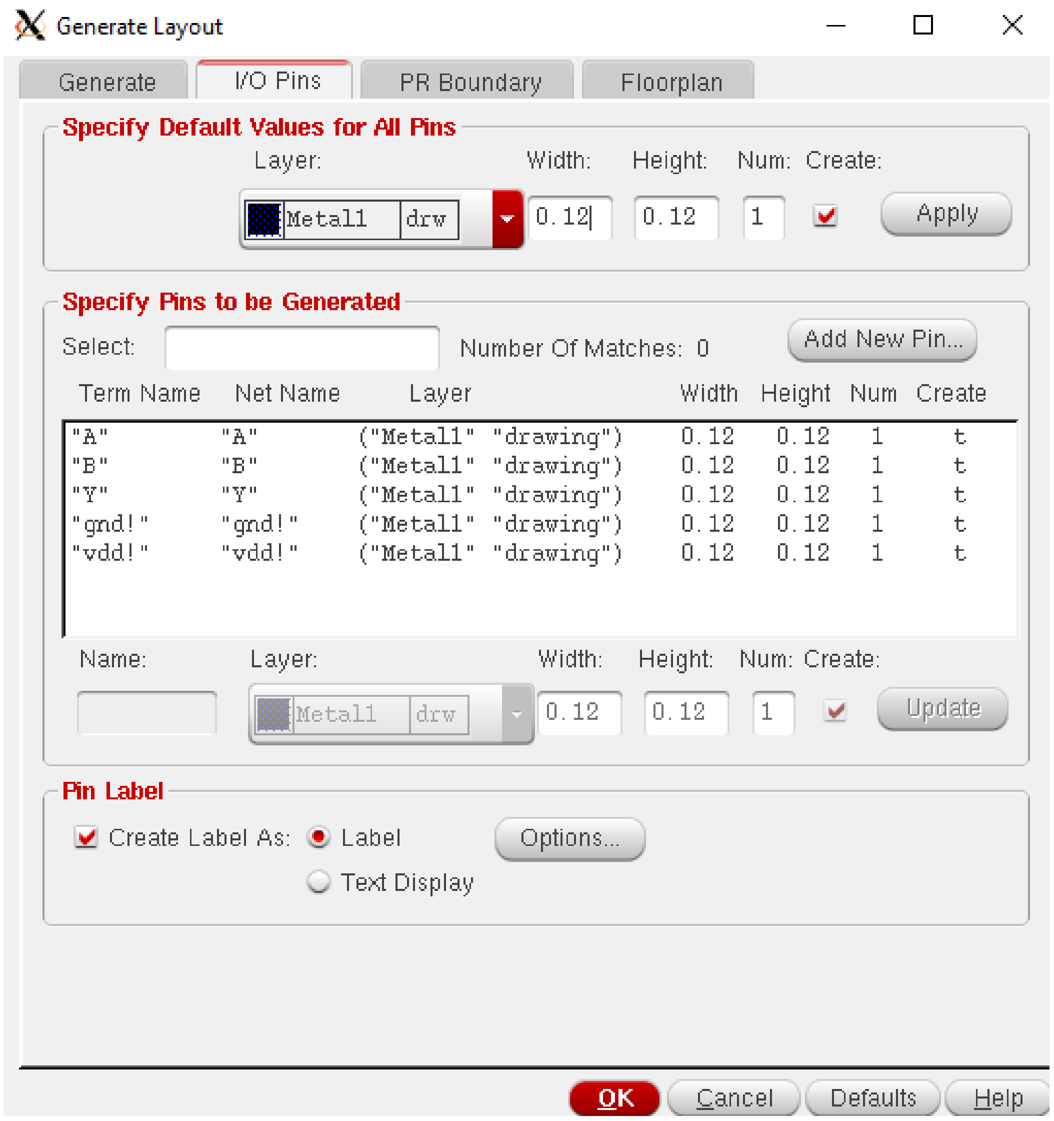
- [ ] Preserve User-Defined Bindings

Connectivity Extraction

- [✓] Extract Connectivity after Generation

OK | Cancel | Defaults | Help

8. Go to **I/O pins** tab. The dialog box shows that all I/O pins are in **Metal1** layer (**Metal1 drw**).



Specify Default Values for All Pins

Layer:	Width:	Height:	Num:	Create:
Metal1 drw	0.12	0.12	1	✓

Specify Pins to be Generated

Select: Number Of Matches: 0

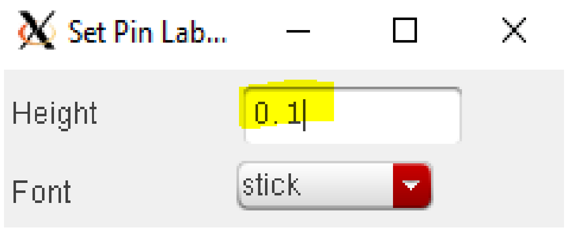
Term Name	Net Name	Layer	Width	Height	Num	Create
"A"	"A"	("Metal1" "drawing")	0.12	0.12	1	t
"B"	"B"	("Metal1" "drawing")	0.12	0.12	1	t
"Y"	"Y"	("Metal1" "drawing")	0.12	0.12	1	t
"gnd!"	"gnd!"	("Metal1" "drawing")	0.12	0.12	1	t
"vdd!"	"vdd!"	("Metal1" "drawing")	0.12	0.12	1	t

Name:	Layer:	Width:	Height:	Num:	Create:
	Metal1 drw	0.12	0.12	1	✓

Pin Label

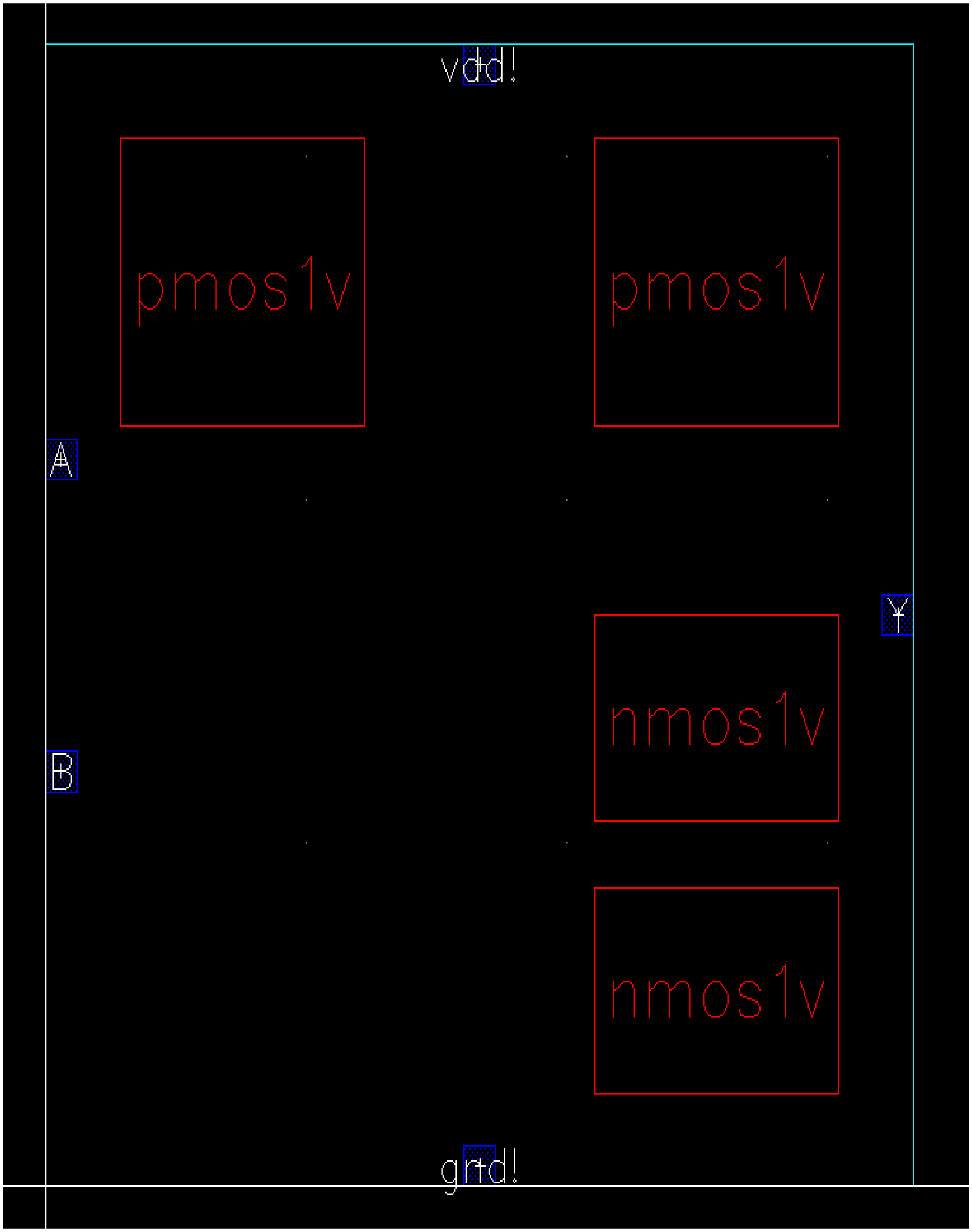
✓ Create Label As: [✓ Label] Text Display

Also put a tick mark on **Create label as Label** and click on **Options**. The following **Set Pin Label** window will appear. Set **Height** to 0.1. Then click **OK** on both the windows.

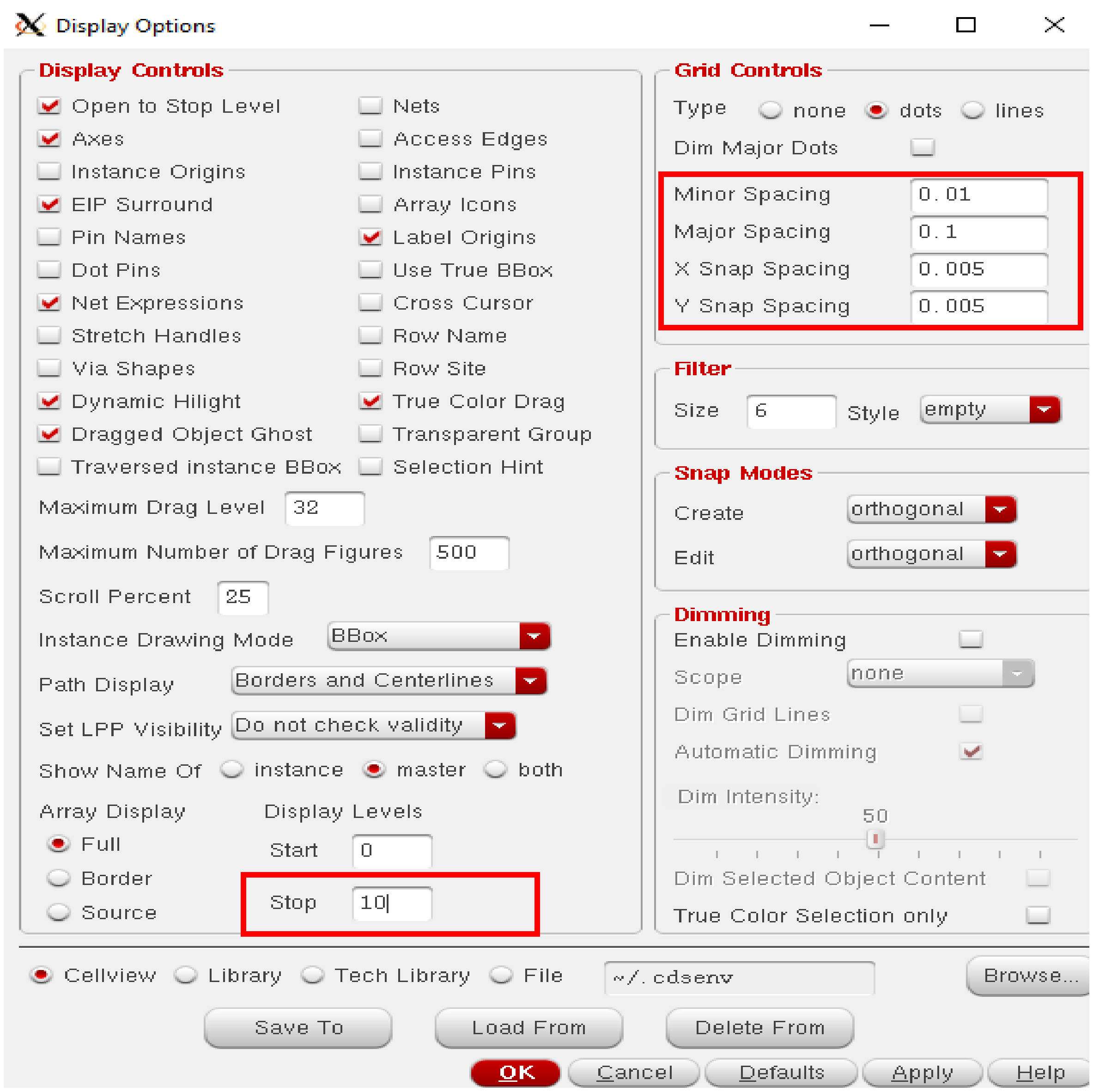


Height	0.1
Font	stick

9. The initial pin and transistor placement in layout will look like the following:



10. Execute ***Options→Display*** or Press ‘**e**’ on keyboard to open ‘**Display Options**’. Fill it in as shown:


Display Options

Display Controls

[✓] Open to Stop Level
[ ] Nets
[✓] Axes
[ ] Access Edges
[ ] Instance Origins
[ ] Instance Pins
[✓] EIP Surround
[ ] Array Icons
[ ] Pin Names
[✓] Label Origins
[ ] Dot Pins
[ ] Use True BBox
[✓] Net Expressions
[ ] Cross Cursor
[ ] Stretch Handles
[ ] Row Name
[ ] Via Shapes
[ ] Row Site
[✓] Dynamic Hilight
[✓] True Color Drag
[✓] Dragged Object Ghost
[ ] Transparent Group
[ ] Traversed instance BBox
[ ] Selection Hint

Maximum Drag Level: 32
Maximum Number of Drag Figures: 500
Scroll Percent: 25
Instance Drawing Mode: BBox
Path Display: Borders and Centerlines
Set LPP Visibility: Do not check validity
Show Name Of: instance [✓ master] both
Array Display: [✓ Full] Border Source
Display Levels: Start 0, Stop 10

Grid Controls

Type: none [✓ dots] lines
Dim Major Dots [ ]
Minor Spacing: 0.01
Major Spacing: 0.1
X Snap Spacing: 0.005
Y Snap Spacing: 0.005

Filter

Size: 6
Style: empty

Snap Modes

Create: orthogonal
Edit: orthogonal

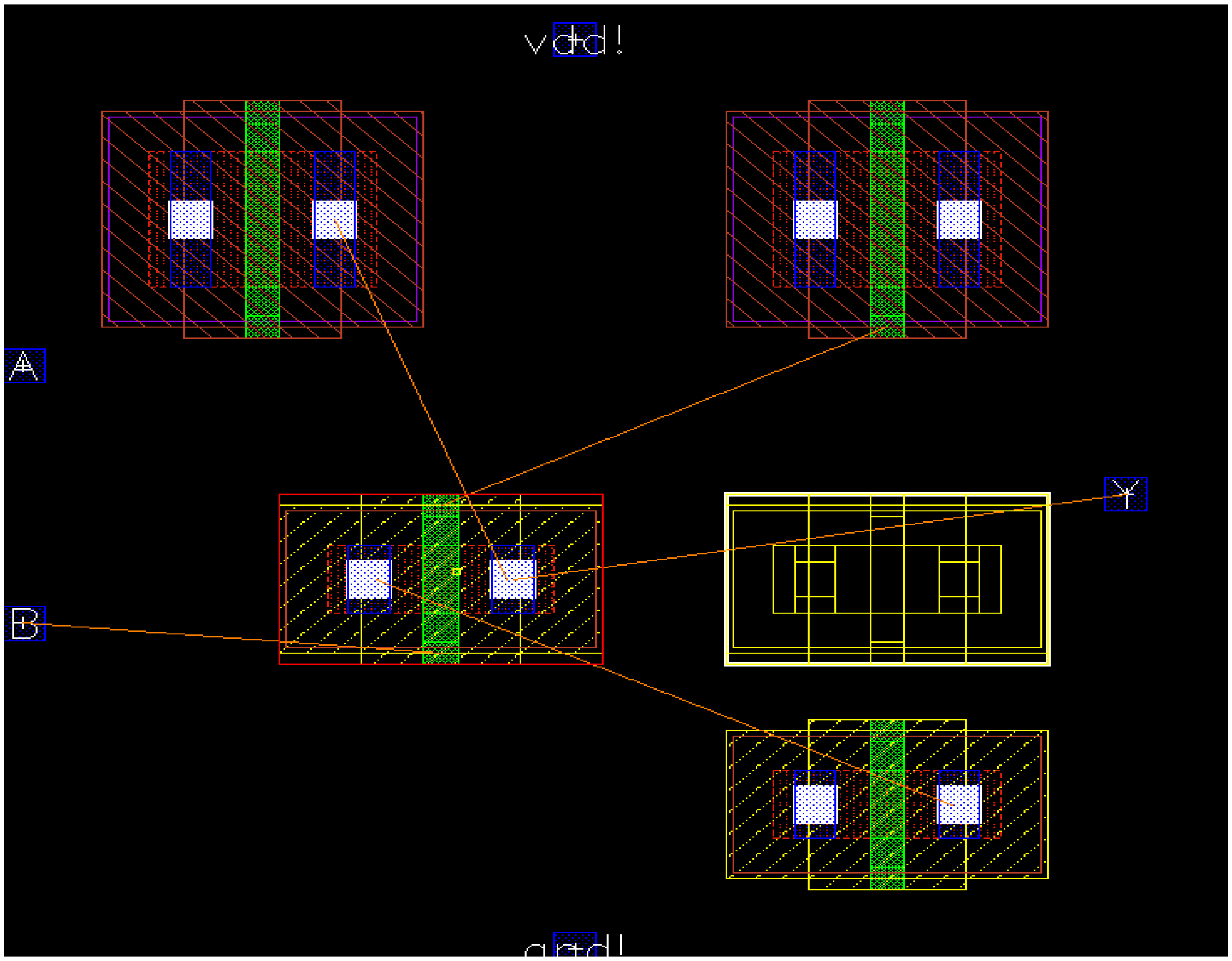
Dimming

Enable Dimming [ ]
Scope: none
Dim Grid Lines [ ]
Automatic Dimming [✓]
Dim Intensity: 50
Dim Selected Object Content [ ]
True Color Selection only [ ]

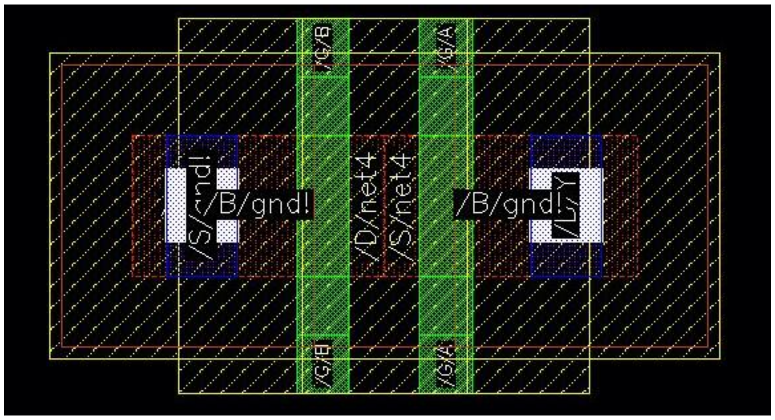
[✓ Cellview] Library Tech Library File: ~/.cdsenv

11. The transistors and pins are shown inside a bounding box, which is an estimate of the optimum size of the final layout. Automatic router will use the bounding box to constrain all routing to occur within the box. The bounding box may need to be re-sized to accommodate all components. An important concept to keep in mind during resizing is that standard cells typically have fixed height (so that power/ground rails line up correctly for routing purposes). **Delete** the **PR Boundary** for now.

Virtuoso Layout Editor XL (VXL) and gpdk090 allow us to create stacked transistors with shared source/drain areas. Zoom in to two transistors at the bottom (to zoom in, type “**z**” and draw a box around the transistors). Click on the transistor on the right and type “**m**” to move the object. As you start dragging the object to the left, fly-lines indicating connectivity will appear as shown below:

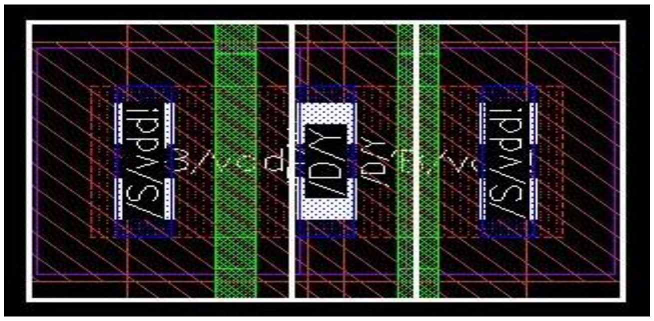


12. When the source/drain areas are overlapped, left-click to fix the position. You should see a transistor stack with shared source/drain areas like this (depending on how far you move, you may need to move left/right a bit):

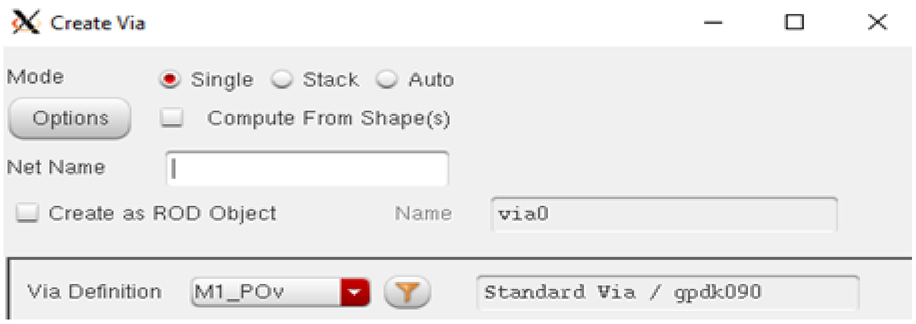


This is a nice NMOS stack for the NAND gate. As you can see, the source/drain contacts have disappeared. Back to the big picture, zoom to fit (press "**F**").

Let's do the same exercise for the PMOS transistors. The PMOS transistors in nand gate do have shared drain contacts because they work in parallel. Connectivity information is extracted from schematic by VXL. The pull-up network looks like the following:



13. Now, connect different layers using path tool (press '**p**' on keyboard), and fill areas by drawing rectangles where necessary (press '**r**' on keyboard). To connect one layer to another (e.g. **Poly to Metal1** or **Metal1 to Metal2**), create via by pressing '**o**' on keyboard and selecting proper **'Via Defintion'**.



Create Via

Mode: [✓ Single] Stack Auto

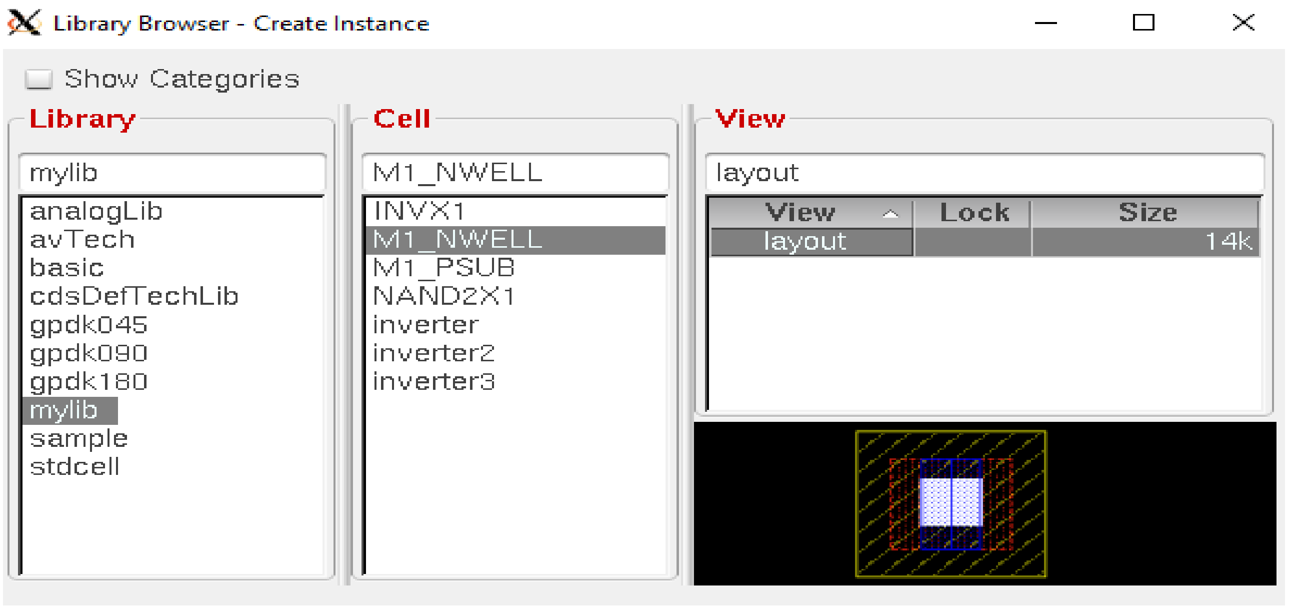
Options ☐ Compute From Shape(s)

Net Name

☐ Create as ROD Object Name via0

Via Definition M1_POv Standard Via / gpdk090

14. Instantiate **M1_PSUB** and **M1_NWELL** cells (that you have created earlier) by pressing '**i**' on keyboard and selecting the layout view from library browser.



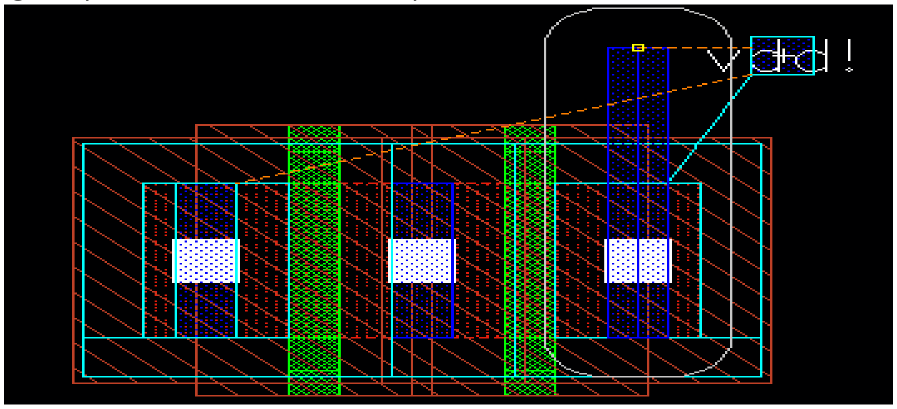
Library Browser - Create Instance

☐ Show Categories

Library	Cell	View
mylib	M1_NWELL	layout
analogLib	INVX1	
avTech	M1_NWELL	
basic	M1_PSUB	
cdsDefTechLib	NAND2X1	
gpdk045	inverter	
gpdk090	inverter2	
gpdk180	inverter3	
mylib		
sample		
stdcell		

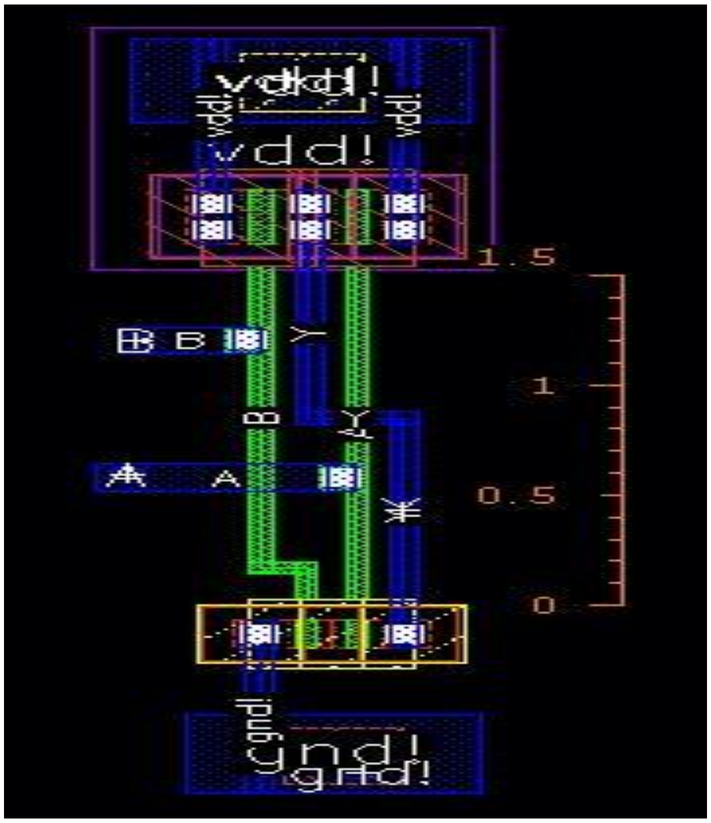
View	Lock	Size
layout		14k

15. Wire up the layout. When you do so, you may encounter multiple options for certain pins. For example, when you select the PMOS to connect its source to VDD, there are multiple Metal1 wires in the PMOS. The desired path will be highlighted and you'll see the fly-line. Continue until you finish routing all the signals. Move **vdd!** and **gnd!** pins to the power rails. As you are moving the pins around, notice the fly-lines that indicate the connections.



A practice that you can follow while wiring is to use **Metal1** for all vertical wiring and **Metal2** for all horizontal wiring inside the cell. Also make the cell height **5 μm**.

16. Your final layout will look something like the following:



17. Perform DRC, LVS and QRC for NAND2X1 as you have done in previous labs. Generate **av_extracted** view and simulate the circuit from that view to verify the functionality.

Lab-10: Introduction to Hierarchical Layout Design

Objectives:

- To be familiar with concept of hierarchical design
- To perform Schematic Level Verification, Layout Design, DRC and LVS check
- To perform post-layout simulation of top level design

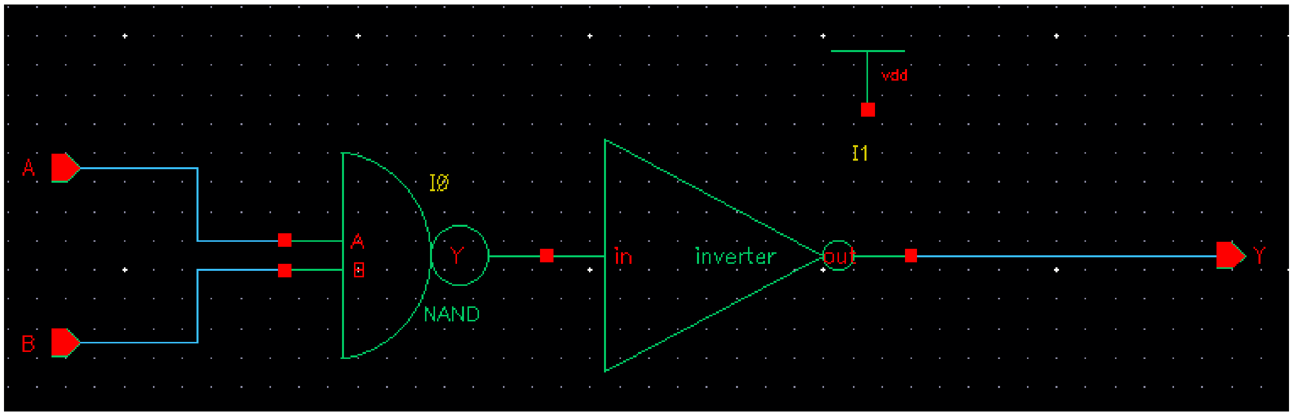
Introduction to Hierarchical Design

By this time, you should have completed layout of ***INVX1*** and ***NAND2X1***. Now, you will learn how to perform hierarchical design. (cell *INVX1* in this section is **inverter** in your case)

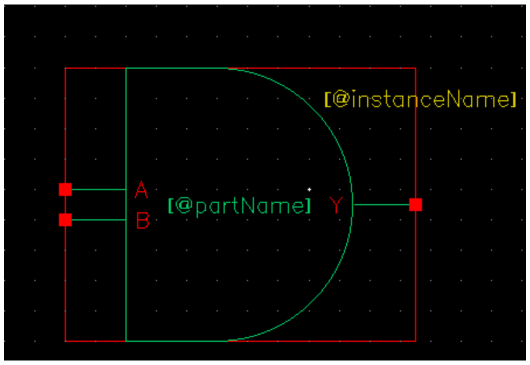
1. Create a new **cellview** of type schematic named *AND*.

2. Instantiate ***NAND*** and ***INVERTER*** symbol from your library ***mylib*** in that schematic.

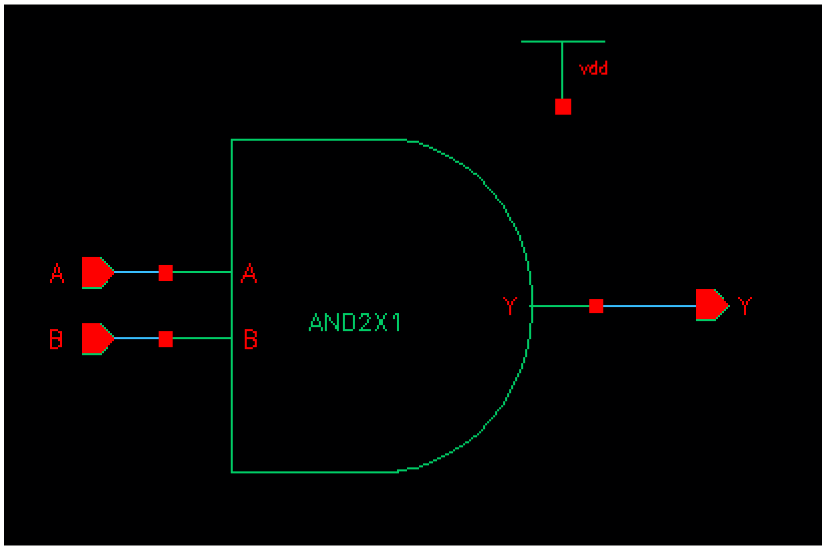
Your final schematic should look something like the following:



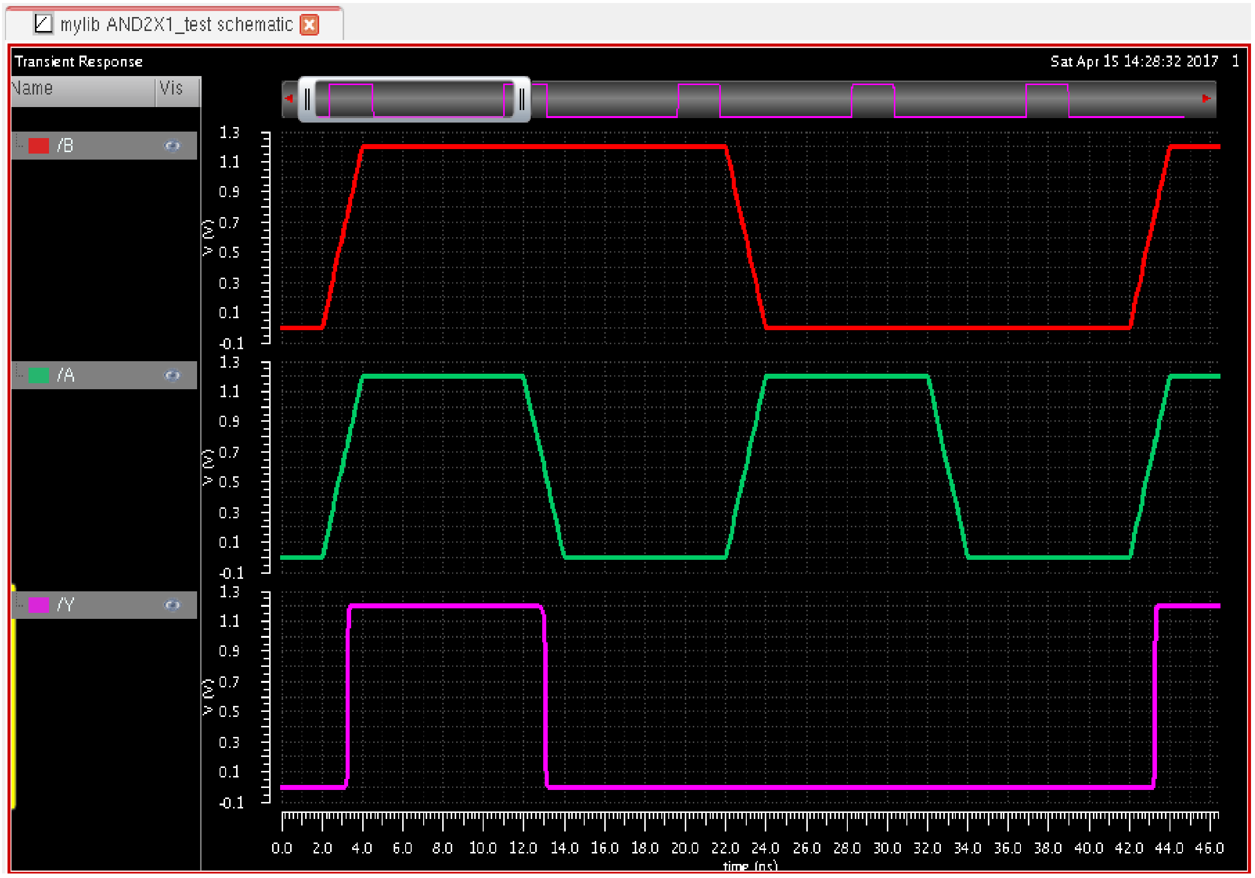
3. Now make a symbol of ***AND***.



4. Now create a new **cellview** named *AND_test*, instantiate *AND* and *vdd* in that cell and the final schematic should look like the following:

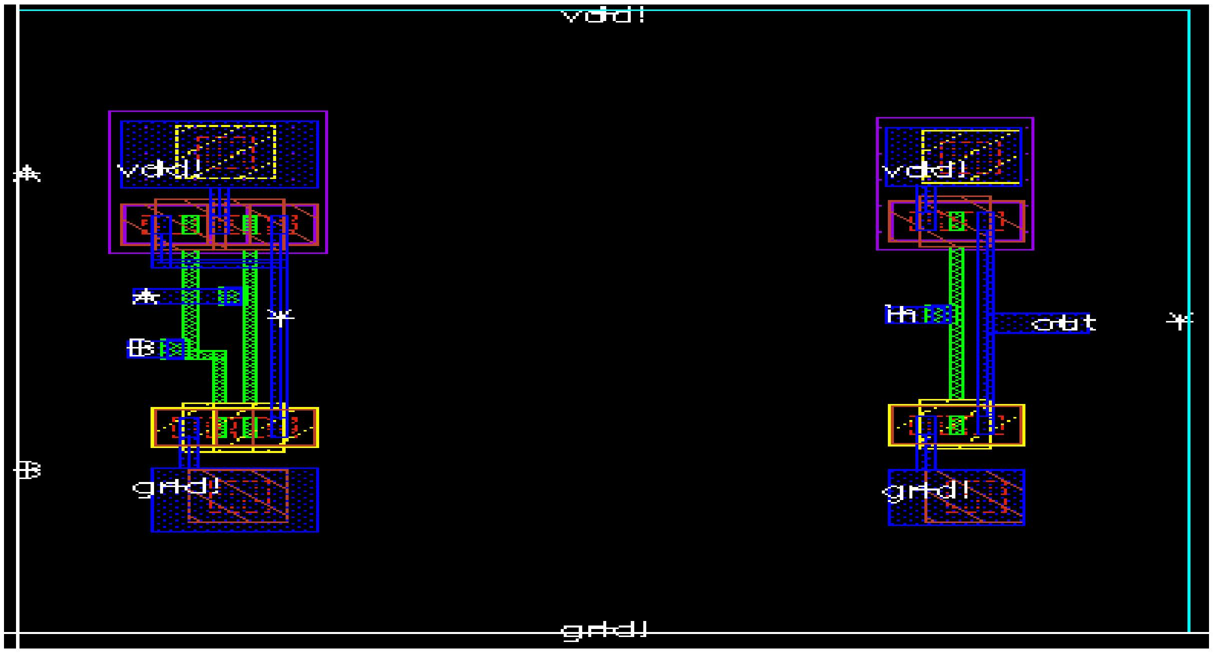


5. Now, launch **ADE L**, setup **Stimuli**, **Model library**, **Analysis** type and **Outputs to be plotted** in the same way as you have done in Lab 5. Run the simulation.

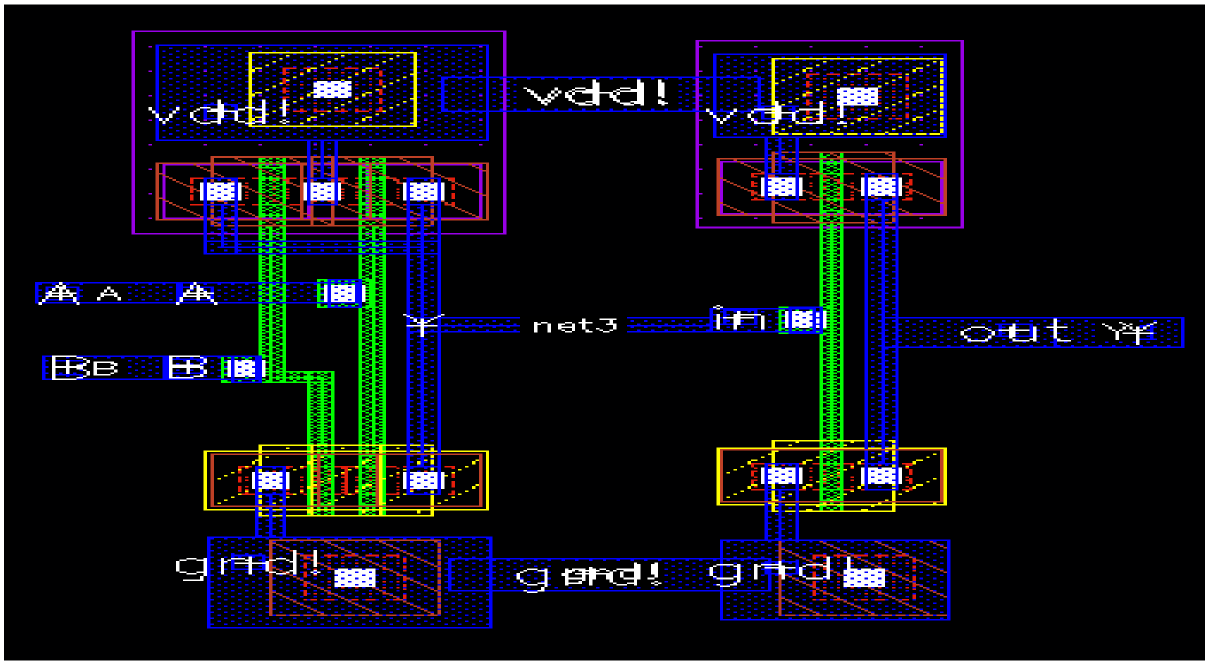


6. If functional verification is okay, then execute ***Launch→Layout XL*** from the schematic of **AND2X1**.

7. Follow procedure of Lab 5 to generate instances and set display options. You will get something like the following:



8. Connect them as required and the final layout should look like the following (probably better!)



9. Perform DRC, LVS and QRC as you have done in previous labs.

References and Acknowledgment

The following resources have been consulted while preparing the manual.

- "**Fundamentals of Digital Logic with Verilog Design**" *by Stephen Brown and Zvonko Vranesic*
- **"CMOS VLSI Design: A Circuits and Systems Perspective"** *by Neil Weste, David Harris.*
- **"Physical Design Essentials: An ASIC Design Implementation Perspective**" *by Khosrow Golshan.*
- **"Digital VLSI Chip Design with Cadence and Synopsys CAD tools"** *by Erik Brunvand.*
- **"Custom IC Design Manual"** *provided by University Support Team, Cadence® Design Systems, Bangalore, India*

Prepared by:

Mr. Adnan Amin Siddiquee
Lecturer, Dept. of EEE, AUST

Editorial Committee:

Dr. Monzurul Islam Dewan, Assistant Professor, Dept. of EEE, AUST
Ms. Oli Lowna Baroi, Assistant Professor, Dept. of EEE, AUST
Mr. Md. Shanian Moed, Lecturer, Dept. of EEE, AUST

Special Thanks to:

Dr. Satyendra Nath Biswas, Professor, Dept. of EEE, AUST
Mr. S. M. Ishraqul Huq, Assistant Professor, Dept. of EEE, AUST
Mr. Anupam Golder, Lecturer, Dept. of EEE, AUST

Published on: Spring 2025 Session