EXPERIMENT No. 01: Write Verilog Code for the following circuits and their Test Bench for verification, observe the waveform and synthesize the code with technological library with given constraints*. Do the initial timing verification with gate level simulation.

i) An inverter

1.1 Objectives:

To write a Verilog and test bench code for an inverter gate and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Basic knowledge of verilog coding, CMOS (Complementary Metal Oxide Semiconductor), MOSFET.

1.4 Introduction:

It consists of PMOS and NMOS FET. The input A serves as the gate voltage for both transistors. The NMOS transistor has an input from Vss (ground) and PMOS transistor has an input from Vdd. The terminal Y is output. When a high voltage (~ Vdd) is given at input terminal (A) of the inverter, the PMOS becomes open circuit and NMOS switched OFF so the output will be pulled down to Vss.
When a low-level voltage (<Vdd, ~0v) applied to the inverter, the NMOS switched OFF and PMOS switched ON. So the output becomes Vdd or the circuit is pulled up to Vdd.

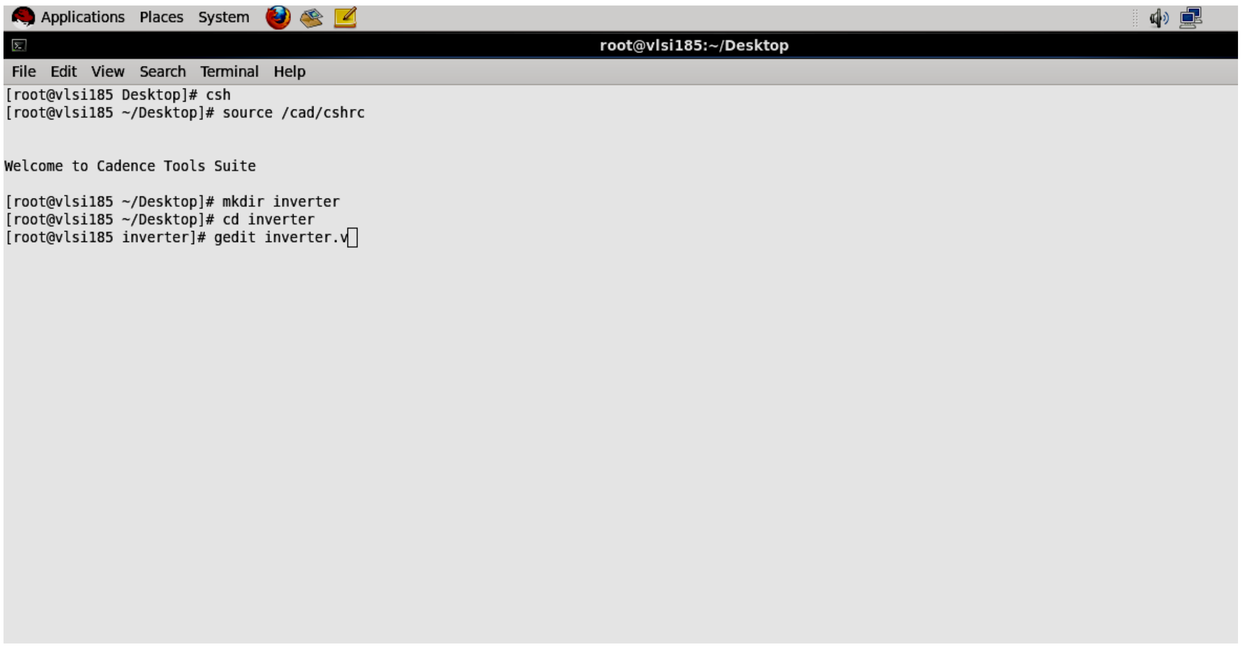
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.

4. To source the cadence tools type **source /cshrc/cad** in the command line.

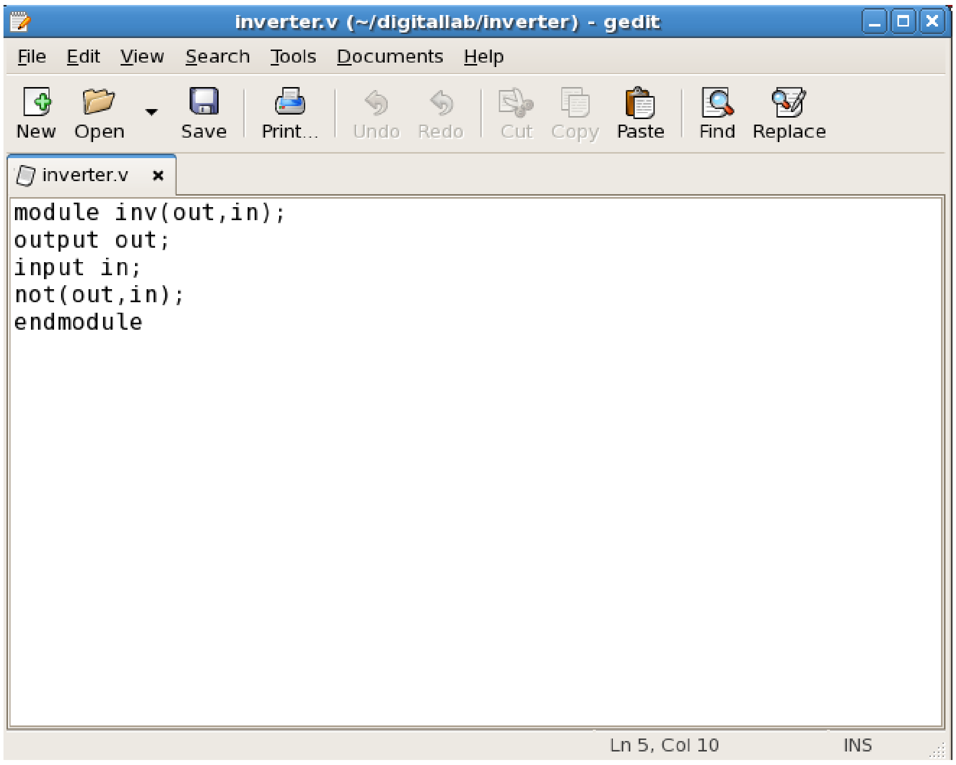
STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES

5. The Verilog code and test bench code should be written inside the respected directory

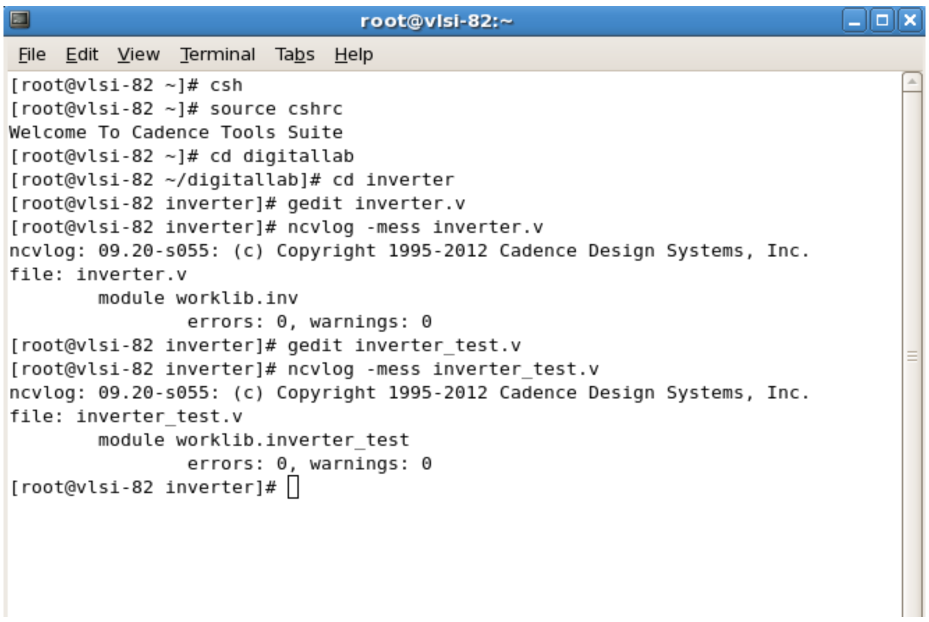

Applications Places System
root@vlsi185:~/Desktop
File Edit View Search Terminal Help

```
[root@vlsi185 Desktop]# csh
[root@vlsi185 ~/Desktop]# source /cad/cshrc


Welcome to Cadence Tools Suite

[root@vlsi185 ~/Desktop]# mkdir inverter
[root@vlsi185 ~/Desktop]# cd inverter
[root@vlsi185 inverter]# gedit inverter.v
```

6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit inverter.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. inverter_test.v.**
8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.

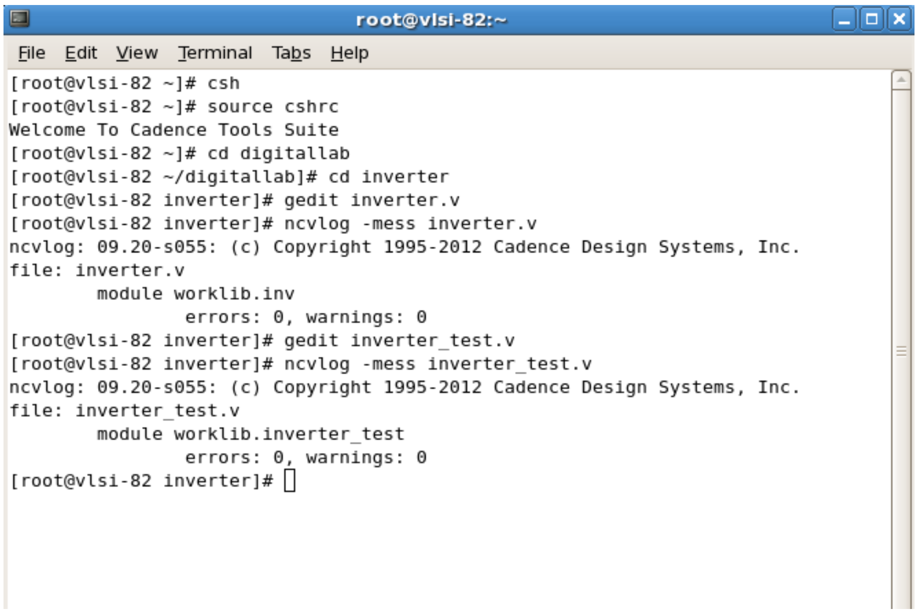

inverter.v (~/digitallab/inverter) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
inverter.v
Ln 5, Col 10
INS

```
module inv(out,in);
output out;
input in;
not(out,in);
endmodule
```



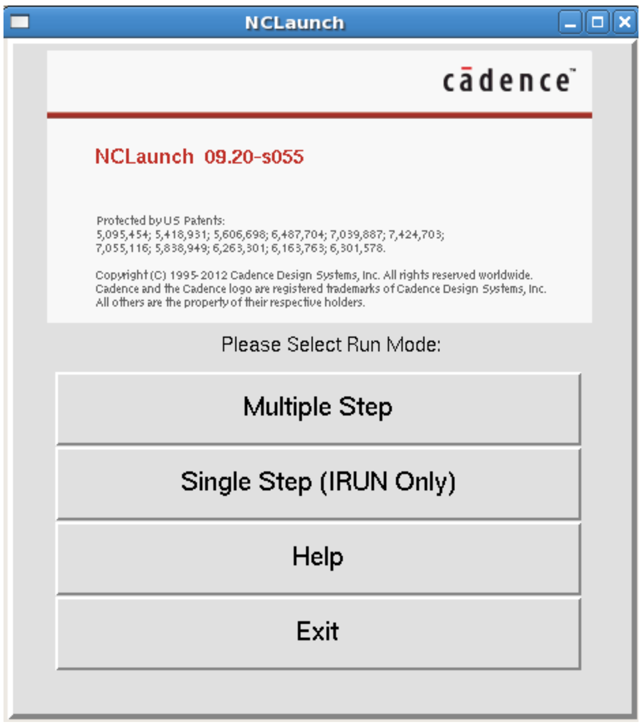
```
[root@vlsi-82 ~]# csh
[root@vlsi-82 ~]# source cshrc
Welcome To Cadence Tools Suite
[root@vlsi-82 ~]# cd digitallab
[root@vlsi-82 ~/digitallab]# cd inverter
[root@vlsi-82 inverter]# gedit inverter.v
[root@vlsi-82 inverter]# ncvlog -mess inverter.v
ncvlog: 09.20-s055: (c) Copyright 1995-2012 Cadence Design Systems, Inc.
file: inverter.v
        module worklib.inv
                errors: 0, warnings: 0
[root@vlsi-82 inverter]# gedit inverter_test.v
[root@vlsi-82 inverter]# ncvlog -mess inverter_test.v
ncvlog: 09.20-s055: (c) Copyright 1995-2012 Cadence Design Systems, Inc.
file: inverter_test.v
        module worklib.inverter_test
                errors: 0, warnings: 0
[root@vlsi-82 inverter]# 
```

10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window



```
[root@vlsi-82 ~]# csh
[root@vlsi-82 ~]# source cshrc
Welcome To Cadence Tools Suite
[root@vlsi-82 ~]# cd digitallab
[root@vlsi-82 ~/digitallab]# cd inverter
[root@vlsi-82 inverter]# gedit inverter.v
[root@vlsi-82 inverter]# ncvlog -mess inverter.v
ncvlog: 09.20-s055: (c) Copyright 1995-2012 Cadence Design Systems, Inc.
file: inverter.v
        module worklib.inv
                errors: 0, warnings: 0
[root@vlsi-82 inverter]# gedit inverter_test.v
[root@vlsi-82 inverter]# ncvlog -mess inverter_test.v
ncvlog: 09.20-s055: (c) Copyright 1995-2012 Cadence Design Systems, Inc.
file: inverter_test.v
        module worklib.inverter_test
                errors: 0, warnings: 0
[root@vlsi-82 inverter]# 
```

11. Select **Single Step (IRUN only)**

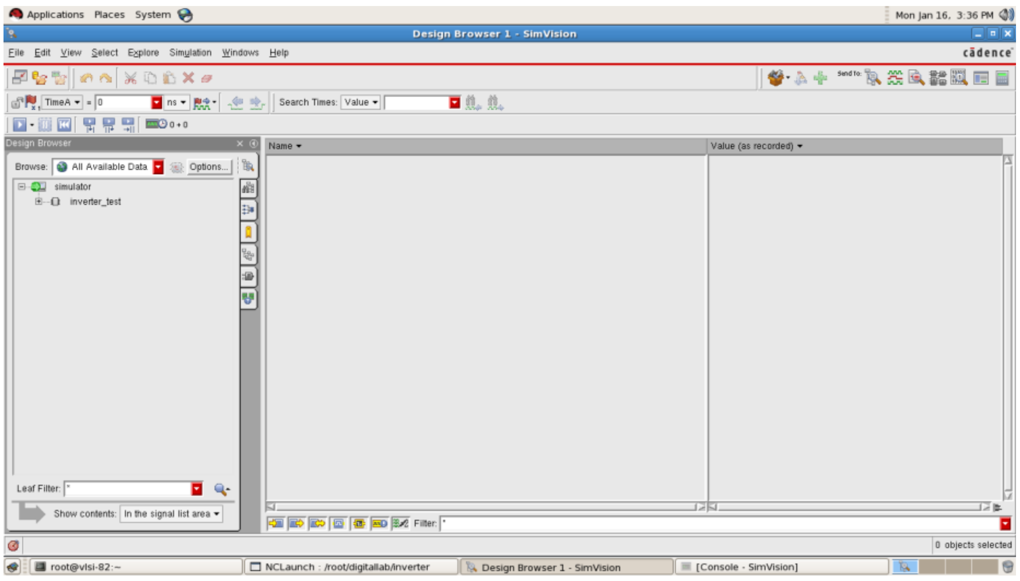


NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:


Applications Places System
Mon Jan 16, 3:36 PM
Design Browser 1 - SimVision
File Edit View Select Explore Simulation Windows Help
cadence
Design Browser
Browse: All Available Data
Options...
simulator
inverter_test
Name
Value (as recorded)
Leaf Filter:
Show contents: In the signal list area
Filter:
0 objects selected
root@vlsi-82:~
NCLaunch : /root/digitallab/inverter
Design Browser 1 - SimVision
[Console - SimVision]

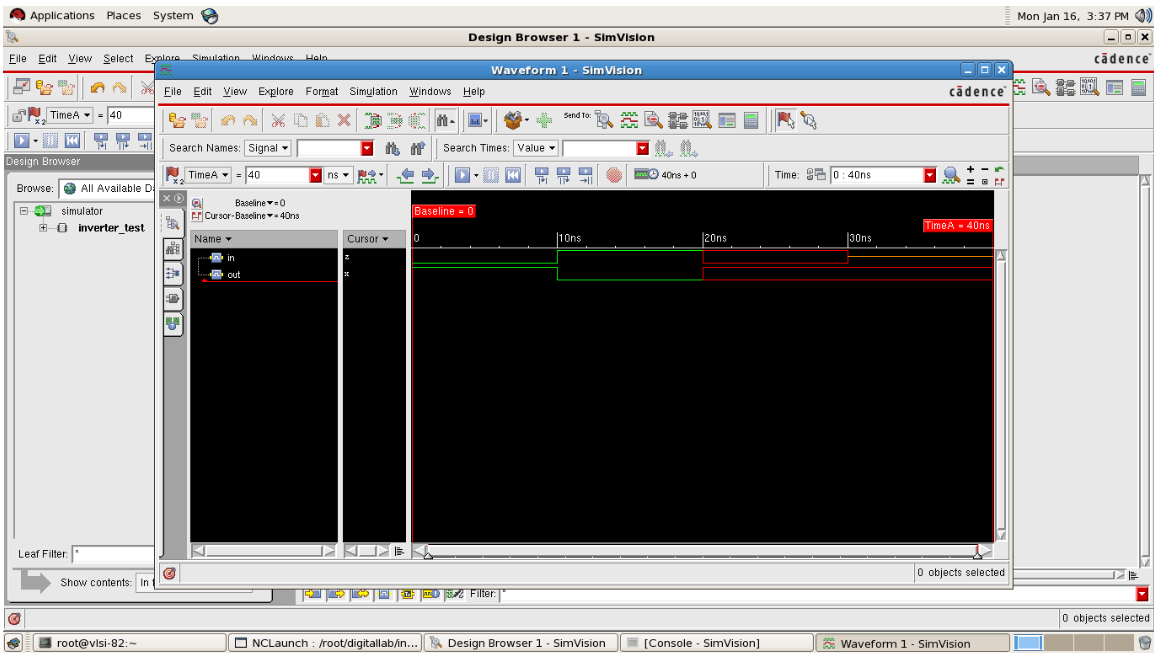
12. Now the tool will verify the codes one by one and open a simulation window if there is no error.

13. Double click on the files both test and code files to check for errors.

14. Double click on the launch IRUN with current selection.

15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**

16. Verify the output.



17. To synthesize the verilog code type **rc –gui** in terminal window.

18. Once enter into gui window type the following commands.

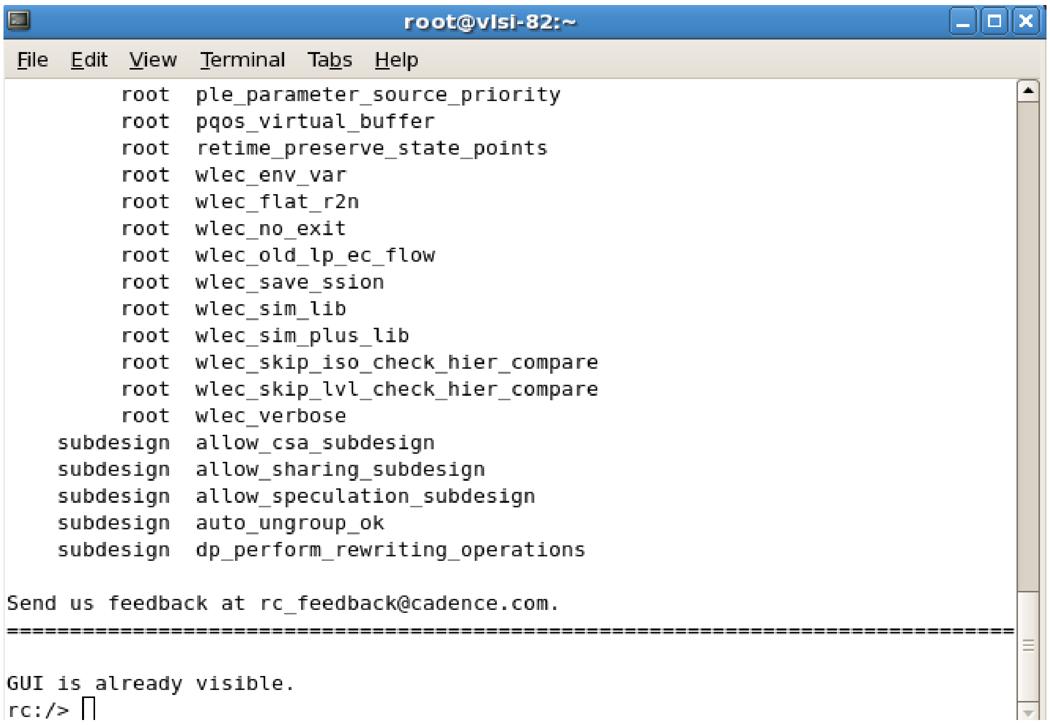
19. set_attr lib_search_path <copy and paste the library folder path>

20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**

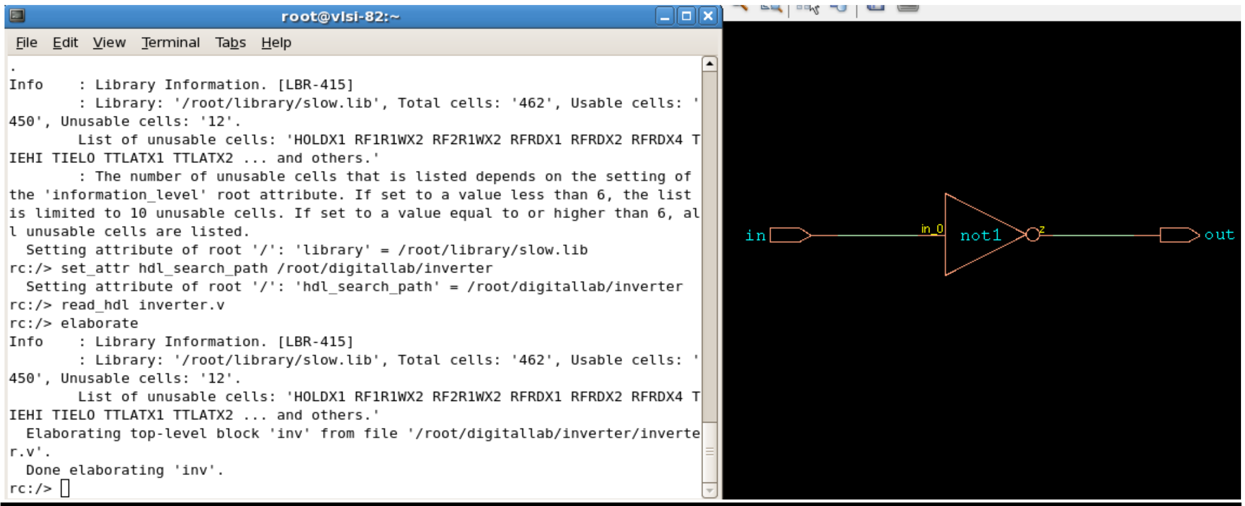
21. set_attr hdl_search_path <copy and paste the verilog folder path>

22. read_hdl <filename.v>

23. elaborate



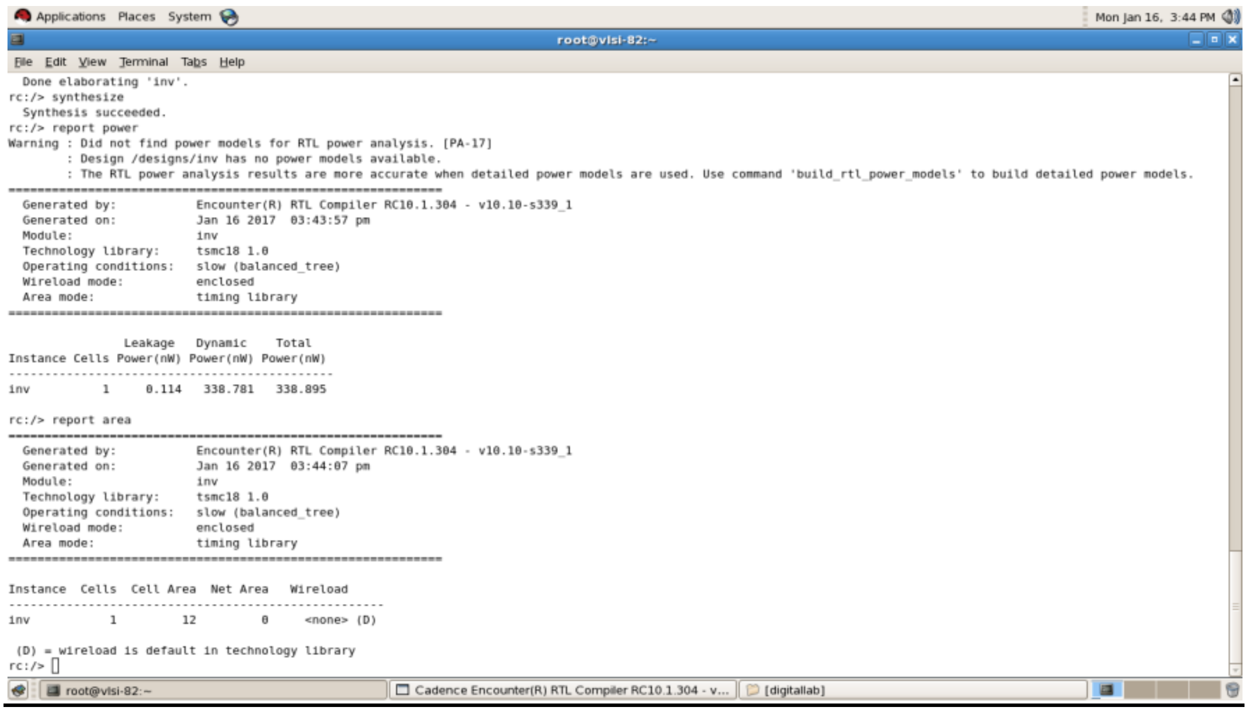
```
          root   ple_parameter_source_priority
          root   pqos_virtual_buffer
          root   retime_preserve_state_points
          root   wlec_env_var
          root   wlec_flat_r2n
          root   wlec_no_exit
          root   wlec_old_lp_ec_flow
          root   wlec_save_ssion
          root   wlec_sim_lib
          root   wlec_sim_plus_lib
          root   wlec_skip_iso_check_hier_compare
          root   wlec_skip_lvl_check_hier_compare
          root   wlec_verbose
     subdesign   allow_csa_subdesign
     subdesign   allow_sharing_subdesign
     subdesign   allow_speculation_subdesign
     subdesign   auto_ungroup_ok
     subdesign   dp_perform_rewriting_operations

Send us feedback at rc_feedback@cadence.com.
================================================================================

GUI is already visible.
rc:/>
```



```
.
Info    : Library Information. [LBR-415]
        : Library: '/root/library/slow.lib', Total cells: '462', Usable cells: '
450', Unusable cells: '12'.
        List of unusable cells: 'HOLDX1 RF1R1WX2 RF2R1WX2 RFRDX1 RFRDX2 RFRDX4 T
IEHI TIELO TTLATX1 TTLATX2 ... and others.'
        : The number of unusable cells that is listed depends on the setting of
the 'information_level' root attribute. If set to a value less than 6, the list
is limited to 10 unusable cells. If set to a value equal to or higher than 6, al
l unusable cells are listed.
  Setting attribute of root '/': 'library' = /root/library/slow.lib
rc:/> set_attr hdl_search_path /root/digitallab/inverter
  Setting attribute of root '/': 'hdl_search_path' = /root/digitallab/inverter
rc:/> read_hdl inverter.v
rc:/> elaborate
Info    : Library Information. [LBR-415]
        : Library: '/root/library/slow.lib', Total cells: '462', Usable cells: '
450', Unusable cells: '12'.
        List of unusable cells: 'HOLDX1 RF1R1WX2 RF2R1WX2 RFRDX1 RFRDX2 RFRDX4 T
IEHI TIELO TTLATX1 TTLATX2 ... and others.'
  Elaborating top-level block 'inv' from file '/root/digitallab/inverter/inverte
r.v'.
  Done elaborating 'inv'.
rc:/>
```

24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.

25. Once synthesize done check for report power, area and gate



```
  Done elaborating 'inv'.
rc:/> synthesize
  Synthesis succeeded.
rc:/> report power
Warning : Did not find power models for RTL power analysis. [PA-17]
        : Design /designs/inv has no power models available.
        : The RTL power analysis results are more accurate when detailed power models are used. Use command 'build_rtl_power_models' to build detailed power models.
============================================================
  Generated by:           Encounter(R) RTL Compiler RC10.1.304 - v10.10-s339_1
  Generated on:           Jan 16 2017  03:43:57 pm
  Module:                 inv
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
---------------------------------------------
inv          1     0.114   338.781   338.895

rc:/> report area
============================================================
  Generated by:           Encounter(R) RTL Compiler RC10.1.304 - v10.10-s339_1
  Generated on:           Jan 16 2017  03:44:07 pm
  Module:                 inv
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance  Cells  Cell Area  Net Area   Wireload
----------------------------------------------------
inv           1         12         0     <none> (D)

 (D) = wireload is default in technology library
rc:/> 
```

26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

VERILOG CODE:

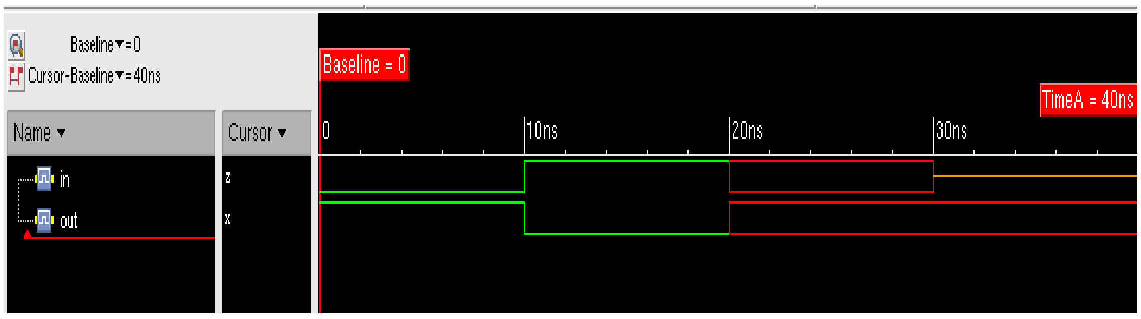
```
module inv(out,in);
output out;
input in;
not(out,in);
endmodule
```

TEST BENCH CODE:

```
// Declare all inputs as register and output as wire
module inv_test;
wire out;
reg in;
inv a1(out,in); // Instead of a1 can give any names u like other than module name.
initial // All stimulus must be inside initial block, clock must be inside always block.
begin
```

```
#10 in=1'b0;
#10 in=1'b1;
#10 in=1'bx;
#10 in=1'bz;
#10 $finish;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. What is MOSFET.
2. Explain the working of MOSFET
3. What is CMOS.
4. Explain the working of CMOS.
5. What is pmos and nmos

1.10 Post – Experimentation Questions:

1. Explain the working of inverter.
2. Explain the characteristics of inverter.
3. Write the truth table of inverter.
4. Write the waveform of inverter.
5. Explain FETs.

ii) A BUFFER

1.1Objectives:

To write a Verilog and test bench code for buffer and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Concept of Buffer.

1.4 Introduction:

A **buffer**, is a basic logic gate that passes its input, unchanged, to its output. Its behavior is the opposite of a NOT gate. The main purpose of a buffer is to regenerate the input, usually using a strong high and a strong low. A buffer has one input and one output; its output always equals its input. Buffers are also used to increase the propagation delay of circuits by driving the large capacitive loads.

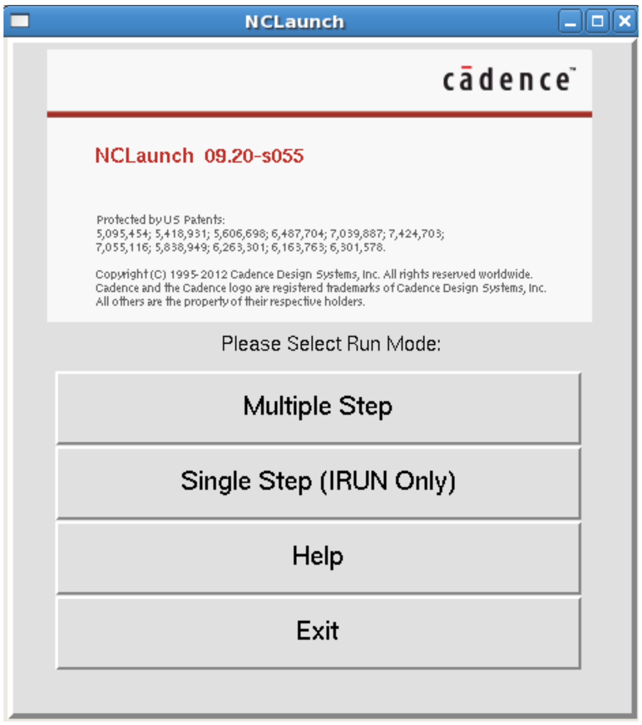
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit buffer.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. buffer_test.v.**

8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>
20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>

22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

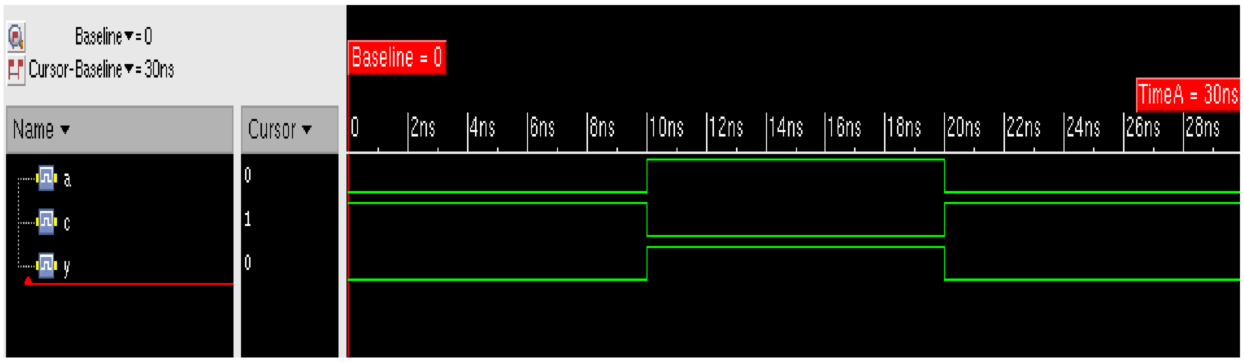
VERILOG CODE:

```
module buffer(a,c,y);
input a;
output y;
inout c;
assign c=~a;
assign y=~c;
endmodule
```

TEST BENCH CODE:

```
module buffer_test;
reg a;
wire c;
wire y;
buffer a1(a,c,y);
initial
begin
$monitor ($time,"c=%d,y=%d"c,y);
#10 a=1'b0;
#10 a=1'b1;
#10 a=1'b0;
#10 $finish;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. What is buffer.
2. Working of PMOS
3. Explain working of NMOS
4. Explain the working of inverter
5. Write the symbol of inverter

1.10 Post – Experimentation Questions:

1. Explain the working of Buffer.
2. Explain the characteristics of Buffer.
3. Write the symbol of Buffer.
4. Write the waveform for buffer.
5. Explain the working of each stage in Buffer.

iii) Transmission Gate

1.1 Objectives:

To write a Verilog and test bench code for Transmission gate and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Knowledge of PMOS, NMOS, CMOS.

1.4 Introduction:

A **transmission gate** (**TG**) it is analog gate similar to a relay that can conduct in both directions or block by a control signal with almost any voltage potential. It is a CMOS-based switch, in which PMOS passes a strong 1 but poor 0, and NMOS passes strong 0 but poor 1. Both PMOS and NMOS work simultaneously.

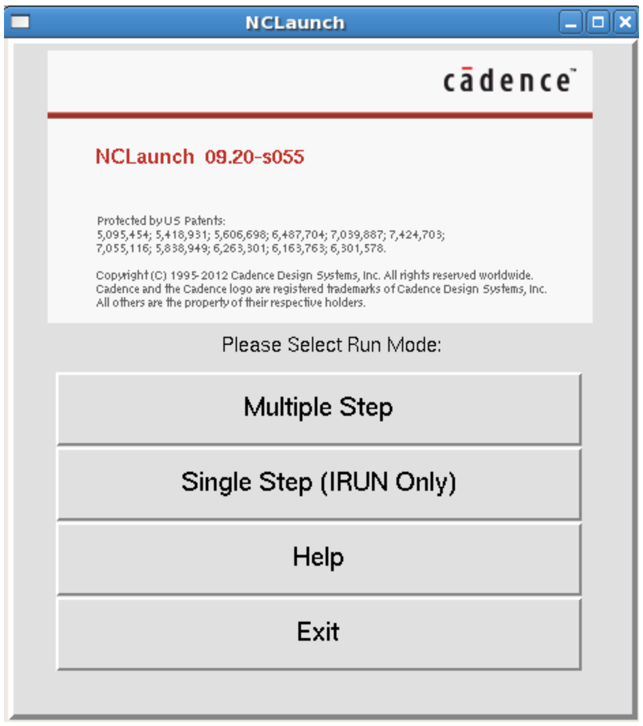
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit transmission.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. transmission_test.v.**
8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.

9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>
20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>

22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

<u>VERILOG CODE</u>:

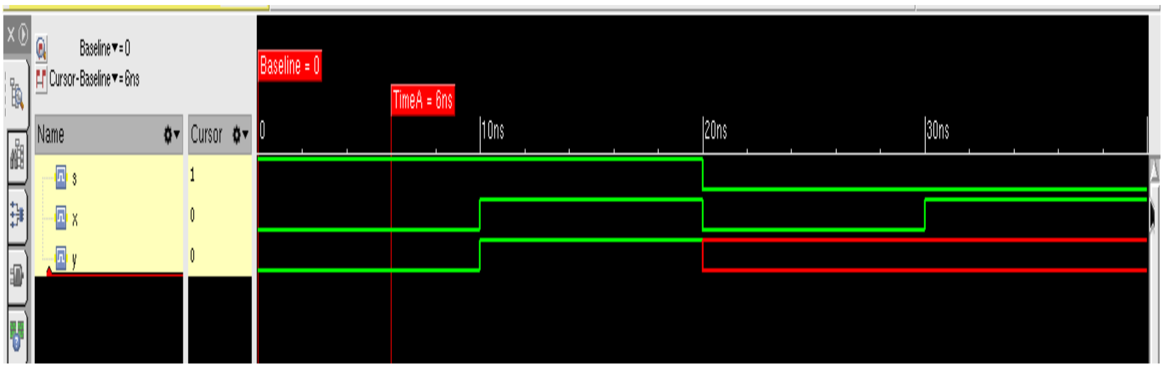
```
module transgate(s,x,y);
input s,x;
output y;
reg y;
always@ (x or s)
begin
if(x==0 && s==1)
y =1'b0;
if(x==1 && s==1)
y =1'b1;
if( s == 0)
y=1'bx;
end
endmodule
```

<u>TEST BENCH CODE</u>:

```
module transgate_test;
reg x,s;
wire y;
transgate a1(s,x,y);
initial
begin
```

```
$monitor($time,"y=%d",y);
x=1'b0;
s=1'b1;
#10 x=1'b1;
s=1'b1;
#10 x=1'b0;
s=1'b0;
#10 x=1'b1;
s=1'b0;
#10 $finish;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Explain the working of NMOS and PMOS.
2. Discuss the theoretical working.
3. Explain the working of buffer.
4. Draw the circuit for TG.
5. Analyze the theoretical working of TG

1.10 Post – Experimentation Questions:

1. Explain the functioning of TG
2. Discuss TG characteristics.
3. Explain the verilog code written for TG
4. Write the symbol for TG.
5. Explain the waveform of TG.

iv) Basic/Universal Gate

1.1 Objectives:

To write a Verilog and test bench code for Basic/Universal gate and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Basic knowledge of digital logic gates and Boolean algebra.

1.4 Introduction:

Digital systems are said to be constructed by using logic **gates**. These **gates** are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR **gates**. NAND gate and NOR gate are universal logic gates because any boolean function can be implemented using one of these gates.

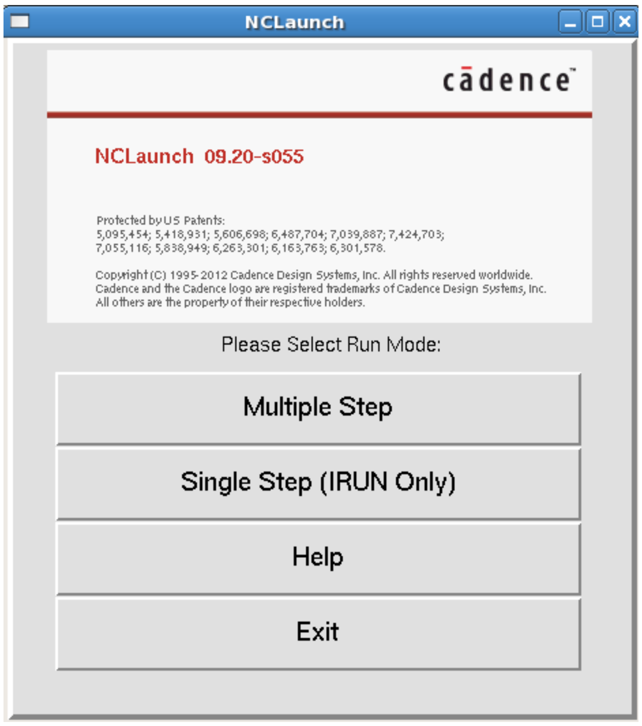
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

<u>STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES</u>

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit basic.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. basic_test.v.**
8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.

9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>
20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>

22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

VERILOG CODE:

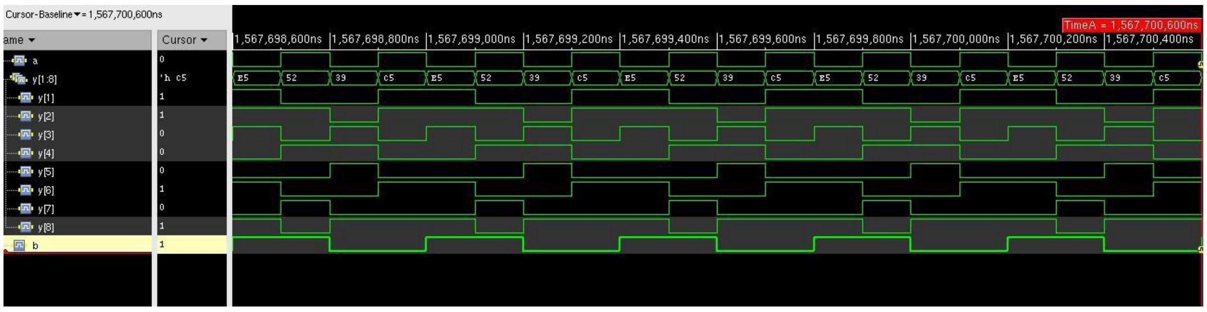
```
module allgate (a,b,y);
input a,b;
output[1:8]y;
assign y[1]=a+b;
assign y[2]=a|b;
assign y[3]=~a;
assign y[4]=~(a+b);
assign y[5]=~(a|b);
assign y[6]=a^b;
assign y[7]=a&b;
assign y[8]=~(a&b);
endmodule
```

TEST BENCH CODE:

```
module tb_allgate;
reg a,b;
wire [1:8]y;
allgate a1(a,b,y);
initial
begin
a=1'b0;
forever #100 a=~a;
end
```

```
initial
begin
b=1'b0;
forever #200 b=~b;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Define logic gates.
2. Explain the Boolean algebra.
3. Write the truth table for AND gate.
4. Write the truth table for OR gate.
5. Write the truth table for NOR and NAND gate.

1.10 Post – Experimentation Questions:

1. Explain working of basic gates and universal gates.
2. Write the symbol of basic gates.
3. List the universal gates and draw the symbols
4. Explain the verilog code for the same.
5. What is wire in verilog.

v) Flip Flops

1.1 Objectives:

To write a Verilog and test bench code for RS, D, JK, MS,T Flip Flops and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Basic knowledge of Latch, memory.

1.4 Introduction:

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic.

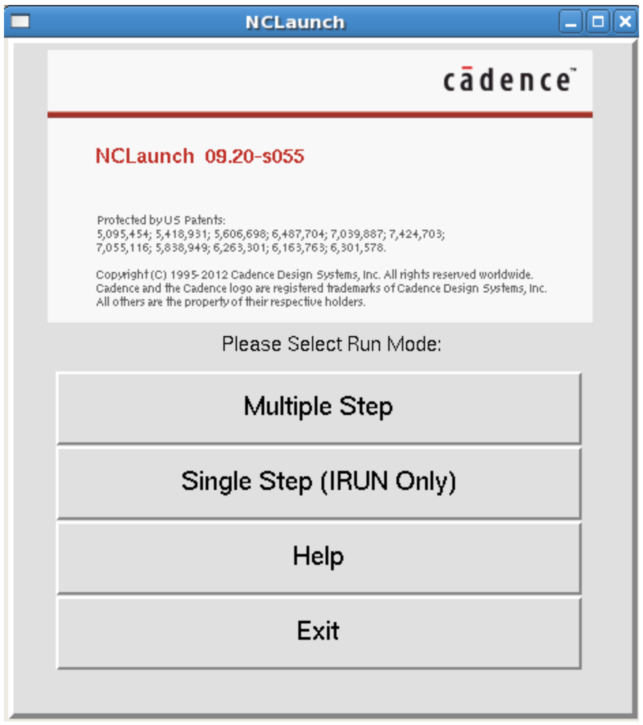
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

<u>STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES</u>

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit flop.v.**

7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. flop_test.v.**
8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch

cādence

NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>

20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>
22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

RS FLIPFLOP

VERILOG CODE:

```
module srff(q,qb,s,r,clk,rst);
output q,qb;
input clk,s,r,rst;
reg tq;
always@ (posedge clk or posedge rst)
begin
if(rst==1)
tq<=1'b0;
else
begin
if (s==1'b0 && r==1'b0)
tq<=tq;
else if(s==1'0 && r==1'b1)
tq<=1'b0;
else if(s==1'b1 && r==1'b0)
tq<=1'b1;
else if(s==1'b1 && r==1'b1)
tq<=1'bx;
end
```

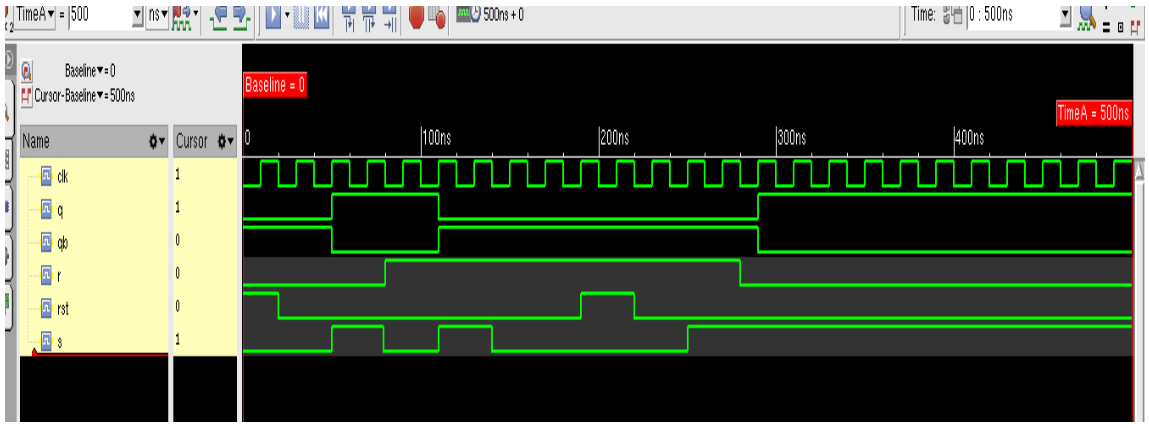
```
end
assign q=tq;
assign qb=~tq;
endmodule
```

<u>TESTBENCH CODE</u>:

```
module srff_test;
reg clk,s,r,rst;
wire q,qb;
srff sr1(q,qb,s,r,clk,rst);
initial
clk=1'b0;
always
#10 clk=~clk;
initial
begin
rst=1'b1;
s=1'b1;
r=1'b0;
#10 s=1'b0;
#10 r=1'b1;
#10 rst=1'b0;
#10 s=1'b1;
r=1'b0;
#10 s=1'b0;
r=1'b0;
#10 s=1'b0;

r=1'b1;
#10 rst=1'b1;
#10 s=1'b1;
r=1'b0;
end
initial
```

```
#500 $finish;
endmodule
```

1.7 Results :



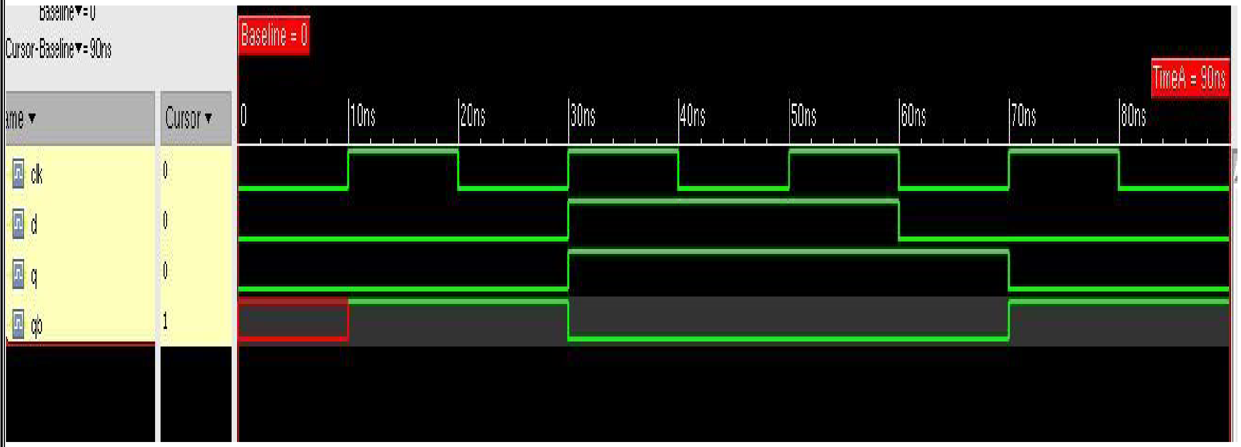
D FLIPFLOP

VERILOG CODE:

```
module dff(d,q,qd,clk);
input d,clk;
output q,qd;
reg q,qd;
initial
begin
q=0;
end
always@(posedge clk)
begin
if(d==0)
q=0;
else
q=1;
qd=~q;
end
endmodule
```

TESTBENCH CODE:

```
module dff_test;
reg d,clk;
wire q,qd;
dff dff1(d,q,qd,clk);
initial
clk=1'b0;
always
#10 clk=~clk;
Initial
begin
$monitor($time,"q=%d","qd=%d",q,qd);
d=1'b0;
#30 d=1'b1;
#30 d=1'b0;
#30 $finish;
end
endmodule
```



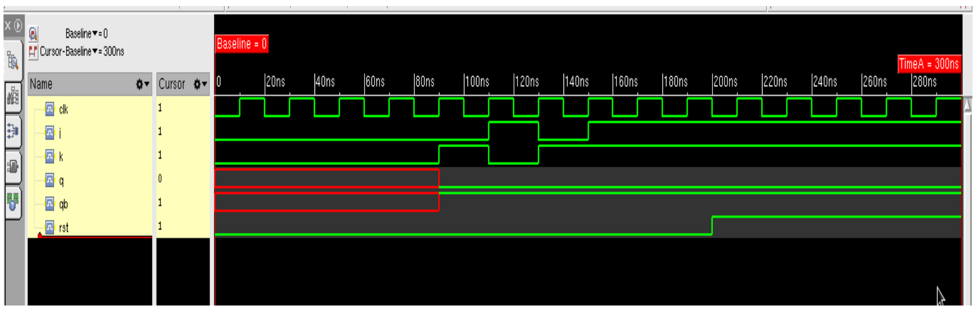
JK FLIPFLOP

VERILOG CODE:

```
module jkff(q,qb,clk,j,k,rst);
input clk,rst,j,k;
output q,qb;
reg q,tq;
always@ (posedge clk or posedge rst)
begin
if(rst==1)
begin
q<=1'b0;
tq<=1'b0;
end
else
begin
if(j==1'b1 && k==1'b0)
q<=j;
else if(j==1'b0 && k==1'b1)
q<=1'b0;
else if(j==1'b1 && k==1'b1)
begin
tq<=~tq;
q<=tq;
end
end
end
assign
qb=~q;
endmodule
```

TESTBENCH CODE:

```
module jk_test;
reg clk, rst,j,k;
wire q,qb;
jkff jk1(q,qb,clk,rst,j,k);
initial
clk =1'b0;
always
#10 clk=~clk;
initial
begin
j=1'b0;k=1'b0;rst=1'b0;
#30 rst=1'b0;
#60 j=1'b0;k=1'b1;
#20 j=1'b1;k=1'b0;
#20 j=1'b0;k=1'b1;
#20 j=1'b1;k=1'b1;
#50 rst=1'b1;
End
initial
#300 $finish;
endmodule
```



MS FLIPFLOP

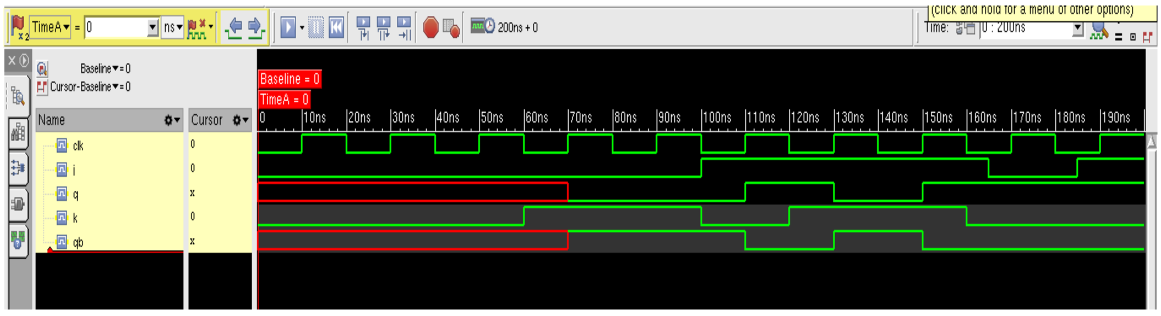
VERILOG CODE:

```
module msff(j,k,clk,q,qb);
input clk,j,k;
output q,qb;
reg q,qb,tq;
always@(clk)
begin
if(!clk)
begin
if(j==1'b0 && k==1'b1)
tq<=1'b0;
else if(j==1'b1 && k==1'b0)
tq<=1'b1;
else if(j==1'b1 && k==1'b1)
tq<=~tq;
end
if(clk)
begin
q<=tq;
qb<=~tq;
end
end
endmodule
```

TESTBENCH CODE:

```
module msff_test;
reg j,k,clk;
wire q,qb;
msff ms(j,k,clk,q,qb);
initial
clk=1'b0;
always
```

```
#10 clk=~clk;
initial
begin
j=1'b0;k=1'b0;
#60 j=1'b0;k=1'b1;
#40 j=1'b1;k=1'b0;
#20 j=1'b1;k=1'b1;
#40 j=1'b1;k=1'b0;
#5 j=1'b0;k=1'b0;
#20 j=1'b1;
#10;
end
initial
#200 $finish;
endmodule
```


TimeA = 0
Baseline = 0
Cursor-Baseline = 0
Name
Cursor
clk
j
q
k
qb
0 10ns 20ns 30ns 40ns 50ns 60ns 70ns 80ns 90ns 100ns 110ns 120ns 130ns 140ns 150ns 160ns 170ns 180ns 190ns

T FLIPFLOP

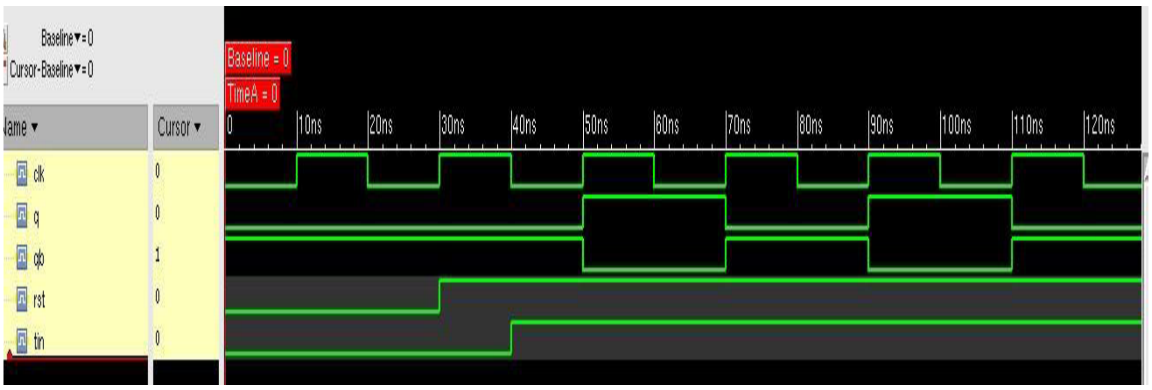
VERILOG CODE:

```
module tff(q,qd,clk,rst,tin);
input tin,clk,rst;
output q,qd;
reg tq;
always@(posedge clk or posedge rst)
begin
if(rst==1)
tq<=1'b0;
else
```

```
begin
if(tin)
tq<=1'b1;
end
end
assign q=tq;
assign qb=~tq;
endmodule
```

TESTBENCH CODE:

```
module tff_test;
reg tin,rst,clk;
wire q,qd;
tff abc(q,qd,clk,rst,tin);
initial
clk=1'b0;
always
#10 clk=~clk;
initial
begin
rst=1'b1;tin=1'b0;
#300 rst=1'b0;
#100 tin=1'b1;
#205 tin=1'b0;
#300 tin=1'b1;
#175 tin=1'b0;
#280 rst=1'b1;
#200 rst=1'b0;
#280 tin=1'b1;
#100;
end
initial
#2000 $finish;
Endmodule
```



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. What is Latch?
2. Discuss different types of memory.
3. Write the truth table for D & JK flip flop.
4. Write the truth table for MS, T flip flop.
5. Write the truth table for RS flip flop.

1.10 Post – Experimentation Questions:

1. Working of T flip-flops.
2. Discuss the difference of all the flip-flop mentioned.
3. Explain the working of D flip flop.
4. Explain the working of RS flip flop.
5. Explain the working of T and MS flip flop.

vi) Serial & Parallel Adder

1.1 Objectives:

To write a Verilog and test bench code for Adder and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Knowledge of full adder, half adder, binary addition.

1.4 Introduction:

The serial binary adder or bit-serial adder is a digital circuit that performs binary addition bit by bit. The serial full adderhas three single-bit inputs for the numbers to be added and the carry in. Parallel adder is a combinatorial circuit (not clocked, does not have any memory and feedback) adding every bit position of the operands in the same time. Thus it is requiring number of bit-adders(full adders + 1 half adder) equal to the number of bits to be added.

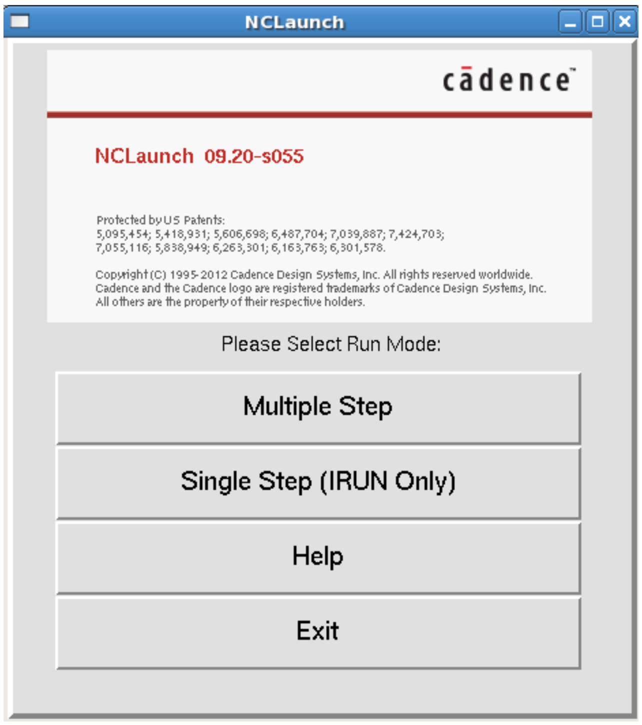
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

<u>STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES</u>

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit adder.v.**

7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. adder_test.v.**
8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch

NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>

20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>
22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

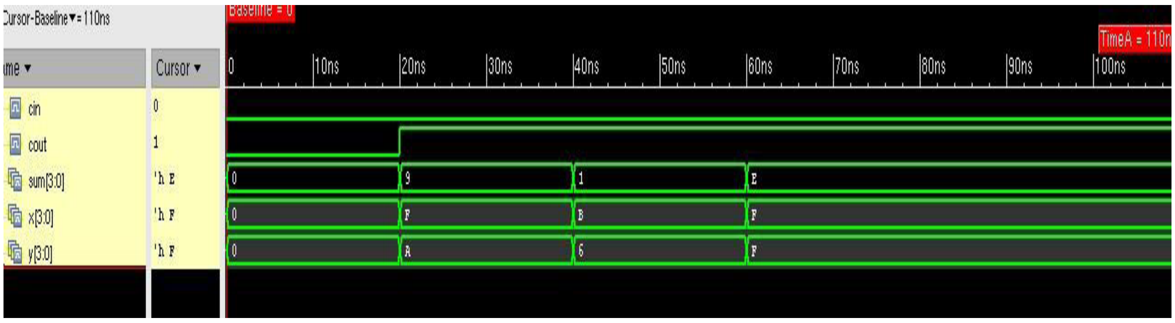
VERILOG CODE:

```
module par(cin,x,y,sum,cout);
input cin;
input [3:0]x,y;
output[3:0]sum;
output cout;
fulladd g0(cin,x[0],y[0],sum[0],c0);
fulladd g1(c0,x[1],y[1],sum[1],c1);
fulladd g2(c1,x[2],y[2],sum[2],c2);
fulladd g3(c2,x[3],y[3],sum[3],cout);
endmodule
module fulladd(cin,x,y,sum,cout);
input cin ,x,y;
output sum ,cout;
assign sum=x^y^cin;
assign cout=((x&y)||(x&cin)||(y&cin));
endmodule
```

TESTBENCH CODE:

```
module par_test;
reg [3:0]x,y;
reg cin;
wire [3:0]sum;
wire cout;
par abc(cin,x,y,sum,cout);
initial
begin
$monitor($time,"sum=%d",sum);
x=4'b0000;y=4'b0000;
cin=1'b0;
#20 x=4'b1111;y=4'b1010;
#20 x=4'b1011;y=4'b0110;
#20 x=4'b1111;y=4'b1111;
#50 $finish;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Discuss half adder.
2. Discuss full adder.
3. Explain the binary addition.
4. Explain 2's complement
5. Explain binary subtraction.

1.10 Post – Experimentation Questions:

1. Explain working of serial adder
2. Explain working of parallel adder.
3. Explain the waveform generated by the code.
4. Explain the difference between serial and parallel adder.
5. Explain the verilog code for the same.

vii) 4 bit Counter (Synchronous and Asynchronous Counter)

1.1 Objectives:

To write a Verilog and test bench code for Counter and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Basic knowledge of binary counters and difference between synchronous and asynchronous.

1.4 Introduction:

In an **asynchronous counter**, an external event is used to directly SET or CLEAR a flip-flop when it occurs. In a **synchronous counter** however, the external event is used to produce a pulse that is synchronised with the internal clock. An example of an **asynchronous counter** is a **ripple counter**.

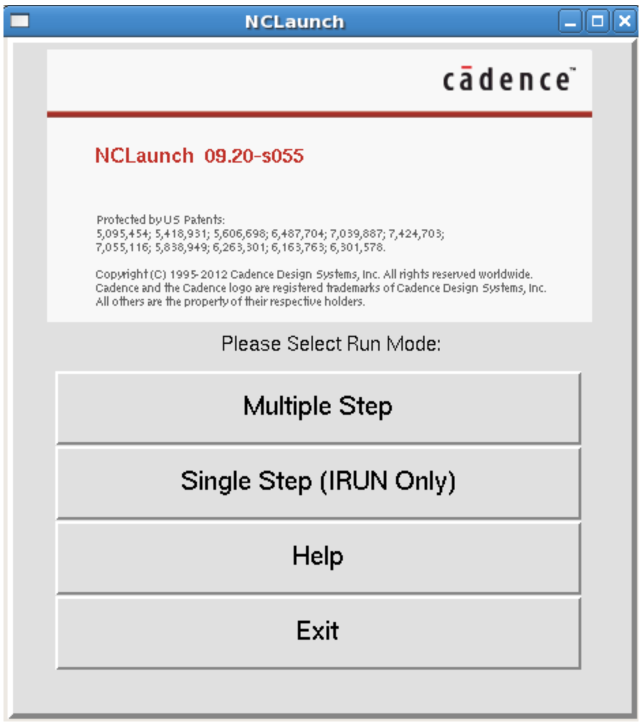
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

<u>STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES</u>

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit counter.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. counter_test.v.**

8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window.
11. Select **Single Step (IRUN only)**



NCLaunch

NCLaunch 09.20-s055

Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

Please Select Run Mode:

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>
20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**
21. set_attr hdl_search_path <copy and paste the verilog folder path>

22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

SYNCHRONOUS COUNTER

VERILOG CODE:

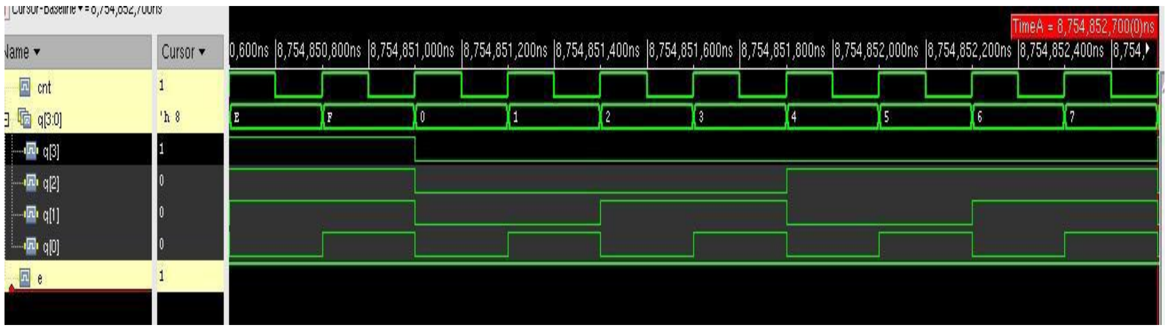
```
module syncnt(cnt,e,q);
input cnt,e;
output [3:0]q;
wire e1,e2,e3;
tff aa(e,cnt,q[0]);
and1 go(q[0],e,e1);
tff bb(e1,cnt,q[1]);
and2 g1(q[1],q[0],e,e2);
tff cc(e2,cnt,q[2]);
and3 g2(q[2],q[1],q[0],e,e3);
tff dd(e3,cnt,q[3]);
endmodule
module tff(t,clk,q);
input t,clk;
output q;
reg q;
initial
q=1'b0;
always@(posedge clk)
begin
if(t==0)
```

```
q=q;
else
q=~q;
end
endmodule
module and1(a,b,c);
input a,b;
output c;
assign c=a&b;
endmodule
module and2(d,e,f,g);
input d,e,f;
output g;
assign g=d&e&f;
endmodule
module and3(p,q,r,s,t);
input p,q,r,s;
output t;
assign t=p&q&r&s;
endmodule
```

TESTBENCH CODE:

```
module syncnt_test;
reg cnt,e;
wire[3:0]q;
syncnt a1(cnt,e,q);
initial
begin
cnt=0;
e=1;
end
always #10cnt=~cnt;
endmodule
```

1.7 Results :



ASYNCHRONOUS COUNTER

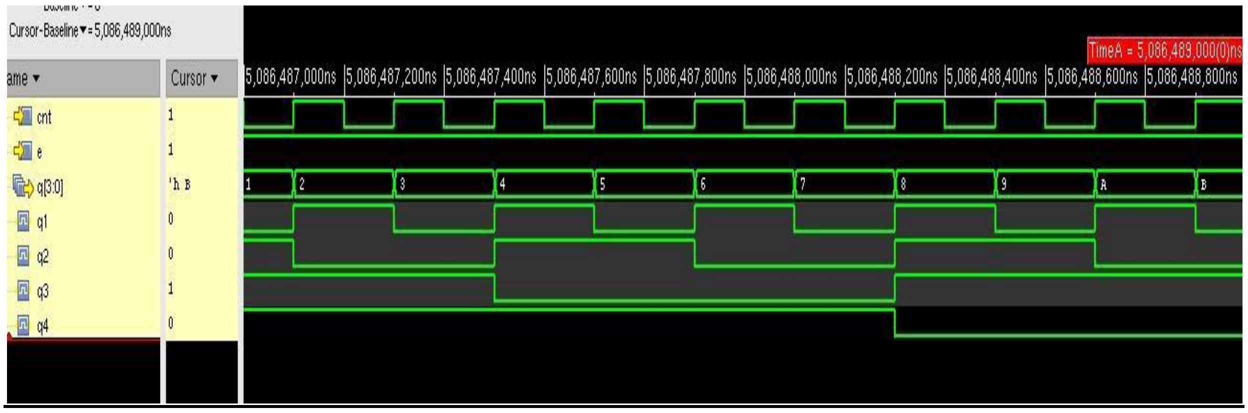
VERILOG CODE:

```
module account(cnt,e,q);
input cnt,e;
output [3:0]q;
wire q1,q2,q3,q4;
tff aaa(e,cnt,q[0],q1);
tff bbb(e,q1,q[1],q2);
tff ccc(e,q2,q[2],q3);
tff ddd(e,q3,q[3],q4);
endmodule
module tff(t,clk,q,qb);
input t,clk;
output q,qb;
reg q,qb;
initial
begin
q=1'b0;
qb=1'b1;
end
always@(posedge clk)
begin
if(t==0)
q=q;
```

```
else
q=~q;
qb=~q;
end
endmodule
```

TESTBENCH CODE:

```
module account_test;
reg cnt,e;
wire [3:0]q;
account ac(cnt,e,q);
initial
begin
cnt=0;
e=1;
end
always
#100 cnt=~cnt;
Endmodule
```



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Explain ripple counter.
2. Explain binary counter.
3. Explain modulo counters.
4. What is the difference between ripple and binary counter.
5. What is the difference between synchronous and asynchronous in digital circuit.

1.10 Post – Experimentation Questions:

1. Explain the working of synchronous counter.
2. Explain the working of asynchronous counter.
3. Explain the waveform generated by synchronous counter.
4. Explain the waveform generated by asynchronous counter.
5. Explain the verilog code for the same.

viii) Successive Approximation Register (SAR)

1.1 Objectives:

To write a Verilog and test bench code for SAR and verify the output waveform.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Knowledge of ADC, Registers.

1.4 Introduction:

Successive Approximation ADC. Successive approximation ADC is the advanced version of Digital ramp type ADC which is designed to reduce the conversion and to increase speed of operation. The successive approximation register counts by changing the bits from MSB to LSB according to input.

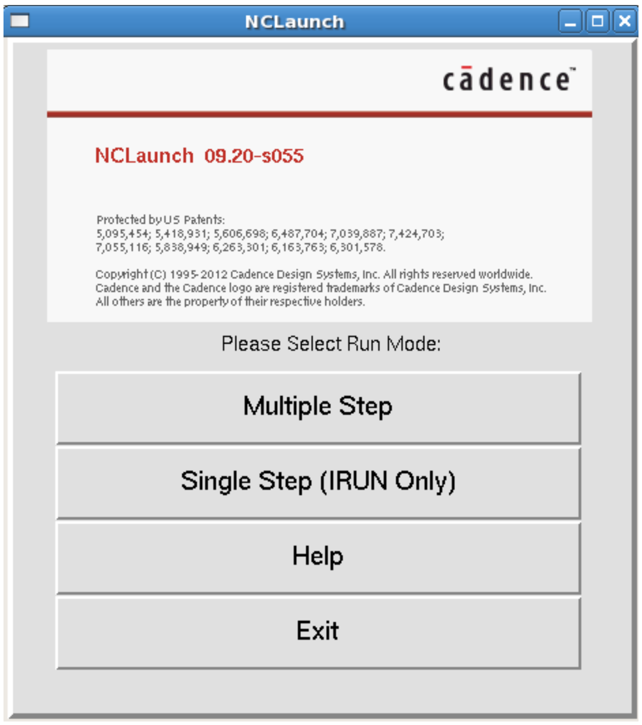
1.5 Procedure:

1. Login to **redhat** operating system with the respective login name and password provided in the lab.
2. Right click on the screen and select **Open Terminal** to open the command prompt.
3. To enter into csh shell type **csh** in the command line.
4. To source the cadence tools type **source /cshrc/cad** in the command line.

STEPS TO COMPLETE AND SIMULATE THE VERILOG CODES

5. The Verilog code and test bench code should be written inside the respected directory
6. To write the Verilog code open the editor by typing **gedit<filename.v>** Eg:**gedit sar.v.**
7. To write the test_bench code open the editor by typing **gedit<filename_test.v>** Eg:**gedit. sar_test.v.**

8. To compile the verilog program **type ncvlog –mess <filename.v>**, check for errors.
9. To compile the test bench program **type ncvlog –mess <filename_test.v>**, check for errors.
10. To verify and run the verilog code in the cadence tool type **nclaunch –new** in terminal window
11. Select **Single Step (IRUN only)**



Protected by US Patents:
5,095,454; 5,418,931; 5,606,698; 6,487,704; 7,039,887; 7,424,703;
7,055,116; 5,838,949; 6,263,301; 6,163,763; 6,301,578.

Copyright (C) 1995-2012 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc.
All others are the property of their respective holders.

12. Now the tool will verify the codes one by one and open a simulation window if there is no error.
13. Double click on the files both test and code files to check for errors.
14. Double click on the launch IRUN with current selection.
15. A new window will appear and in that select in and out right click on it and select the option **send to waveform.**
16. Verify the output.
17. To synthesize the verilog code type **rc –gui** in terminal window.
18. Once enter into gui window type the following commands.
19. set_attr lib_search_path <copy and paste the library folder path>
20. **set_attr library <copy and paste the slow.lib file path inside the library folder>**

21. set_attr hdl_search_path <copy and paste the verilog folder path>
22. read_hdl <filename.v>
23. elaborate
24. Once elaboration done synthesize the code by typing **synthesize** in terminal window.
25. Once synthesize done check for report power, area and gate
26. Optimize the program using command **synthesize –to_mapped**, and check for report power, area and gate. Compare the report of after and before optimization. Repeat same steps for all the programs.

1.6 Program:

VEROLIG CODE:

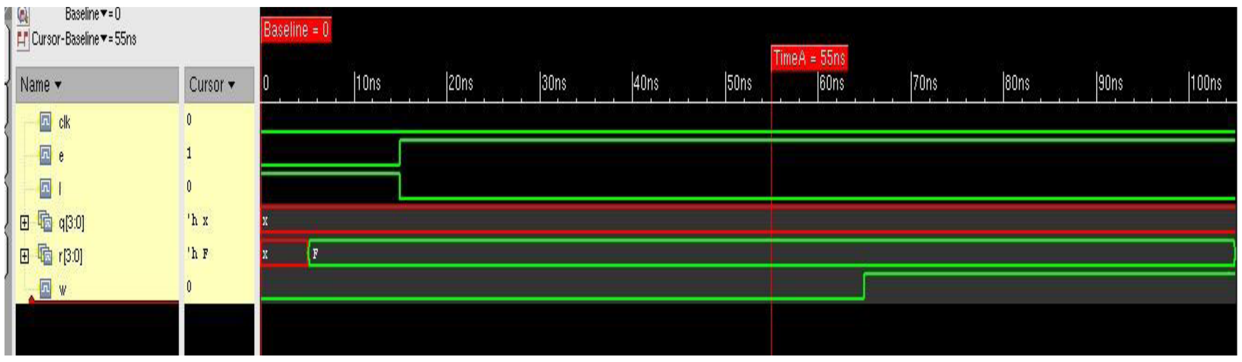
```
module sar(l,e,w,clk,r,q);
input l,e,w,clk;
input [3:0]r;
output [3:0]q;
reg [3:0]q;
integer k;
always@(posedge clk)
if(1)
q=r;
else if(e)
begin
for(k=0;k<3;k=k+1)
q[k]=q[k+1];
q[3]=w;
end
endmodule
```

TESTBENCH CODE:

```
module sar_test;
reg l,e,w,clk;
reg [3:0]r;
wire [3:0]q;
```

```
sar abc(l,e,w,clk,r,q);
initial
clk=1'b0;
always
#10 clk=-clk;
initial
begin
l=1'b1;
w=1'b0;
e=1'b0;
#5 r=4'b1111;
#10 l=1'b0;e=1'b1;w=1'b0;
#10 w=1'b0;
#10 w=1'b0;
#10 w=1'b0;
#10 w=1'b0;
#10 w=1'b1;
#10 w=1'b1;
#10 w=1'b1;
#10 w=1'b1;
#10 $finish;
end
endmodule
```

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. What is sampling.
2. What is sampling theorem.
3. Explain R-2R DAC.
4. Explain working of ADC
5. Explain working of DAC.

1.10 Post – Experimentation Questions:

1. Discuss successive approximation register.
2. Explain functioning of ADC in detail.
3. Explain the verilog code for SAR.
4. Explain the waveforms generated by the same.
5. Define Nyquist rate.

EXPERIMENT No. 01: Design an Inverter with given specifications**, completing the design flow mentioned below:

a. Draw the schematic and verify the following

i) DC Analysis ii) Transient Analysis

b. Draw the Layout and verify the DRC, ERC

c. Check for LVS

d. Extract RC and back annotate the same and verify the Design

e. Verify & Optimize for Time, Power and Area to the given constraint*

1.1 Objectives:

Design an Inverter with given specifications and complete design till layout and verify the parameters DRC, ERC, QRC. Extract RC and back annotate for the verification.

1.2 Apparatus Required:

Cadence Tools.

1.3 Pre-Requisite:

Basic Knowledge of CMOS and MOSFET. W/L ratio.

1.4 Introduction:

It consists of PMOS and NMOS FET. The input A serves as the gate voltage for both transistors. The NMOS transistor has an input from Vss (ground) and PMOS transistor has an input from Vdd. The terminal Y is output. When a high voltage (~ Vdd) is given at input terminal (A) of the inverter, the PMOS becomes open circuit and NMOS switched OFF so the output will be pulled down to Vss.

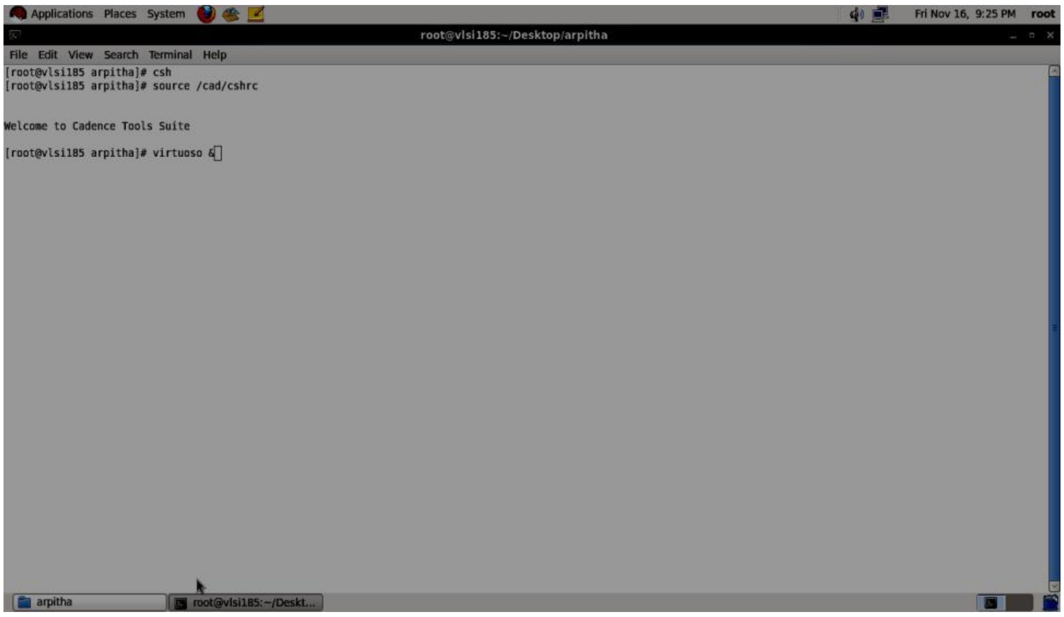
When a low-level voltage (<Vdd, ~0v) applied to the inverter, the NMOS switched OFF and PMOS switched ON. So the output becomes Vdd or the circuit is pulled up to Vdd.

A CMOS circuit is composed of two MOSFETs. The top FET is a PMOS type device while the bottom FET is an NMOS type. The body effect is not present in either device since the body of each device is directly connected to the device’s source. Both gates are connected to the input line. The output line connects to the drains of both FETs.

1.5 Procedure:

STEPS FOR THE ANALOG LAB

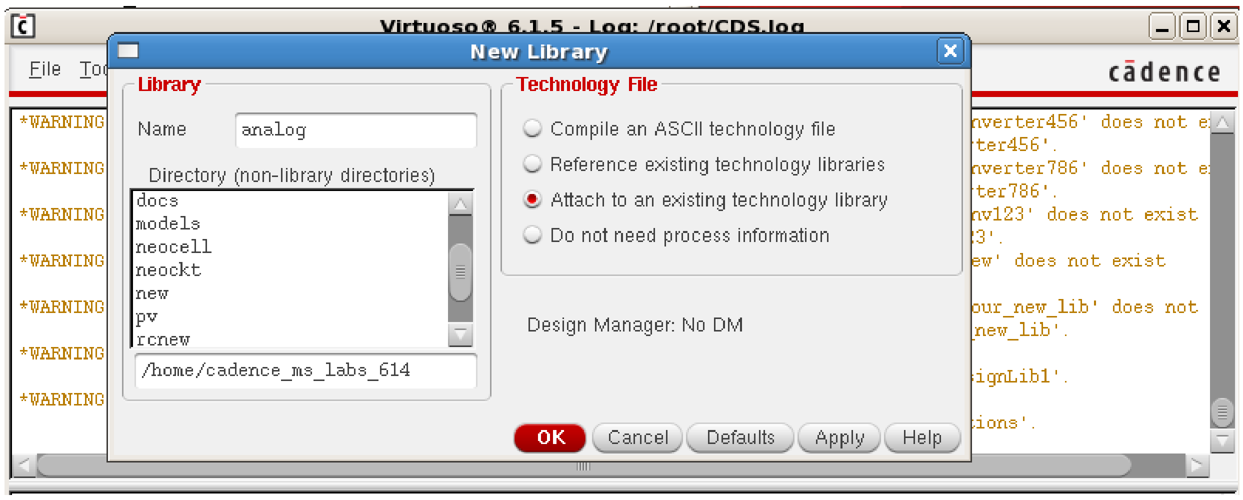
- Login to **redhat** operating system with the respective login name and password provided in the lab.
- Right click on the screen and select **Open Terminal** to open the command prompt.
- To enter into csh shell type **csh** in the command line.
- To source the cadence tools type **source /cshrc/cad** in the command line.
- To open the tool type **virtuoso &** in the command line.


Applications Places System
Fri Nov 16, 9:25 PM root
root@vlsi185:~/Desktop/arpitha
File Edit View Search Terminal Help
arpitha
root@vlsi185:~/Deskt...

```
[root@vlsi185 arpitha]# csh
[root@vlsi185 arpitha]# source /cad/cshrc


Welcome to Cadence Tools Suite

[root@vlsi185 arpitha]# virtuoso &
```

Creating the Library

- Creating a new library by going to the link **File -> New -> Library**
- Give the library name and click ok.
- Select **attach to an existing technology library** and click **ok.**



New Library

Library

Name: analog

Directory (non-library directories)

```
docs
models
neocell
neockt
new
pv
rcnew
```

/home/cadence_ms_labs_614

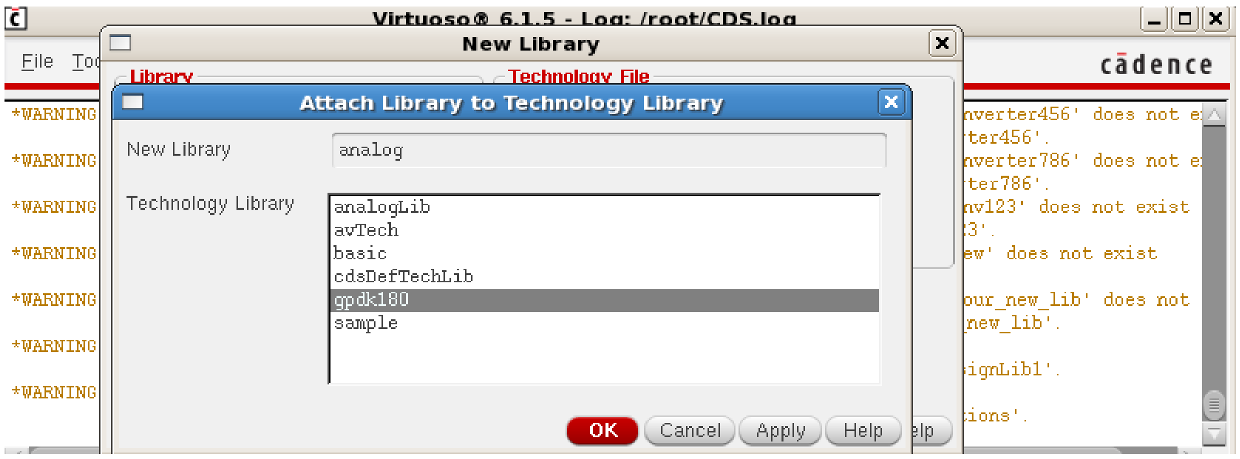
Technology File

- ( ) Compile an ASCII technology file
- ( ) Reference existing technology libraries
- [✓ Attach to an existing technology library]
- ( ) Do not need process information

Design Manager: No DM

```
*WARNING   nverter456' does not e
           ter456'.
*WARNING   nverter786' does not e
           ter786'.
*WARNING   nv123' does not exist
           3'.
*WARNING   ew' does not exist

*WARNING   our_new_lib' does not
           new_lib'.
*WARNING   ignLib1'.

*WARNING   ions'.
```

- Select **gpdk180** and click **ok.**



Attach Library to Technology Library

New Library: analog

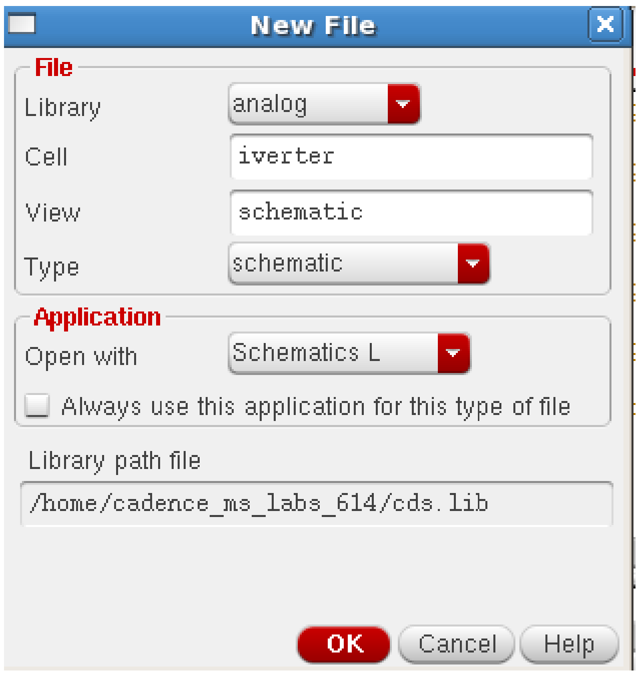
Technology Library

```
analogLib
avTech
basic
cdsDefTechLib
gpdk180
sample
```

```
*WARNING   nverter456' does not e
           ter456'.
*WARNING   nverter786' does not e
           ter786'.
*WARNING   nv123' does not exist
           3'.
*WARNING   ew' does not exist

*WARNING   our_new_lib' does not
           new_lib'.
*WARNING   ignLib1'.

*WARNING   ions'.
```

SCHEMATIC AND SYMBOL CREATION

- In the Library manager select given library
- In Library manager select **File -> New -> Cell view.**
- Give the cell name as inverter



New File

File

Library: analog

Cell: iverter

View: schematic

Type: schematic

Application

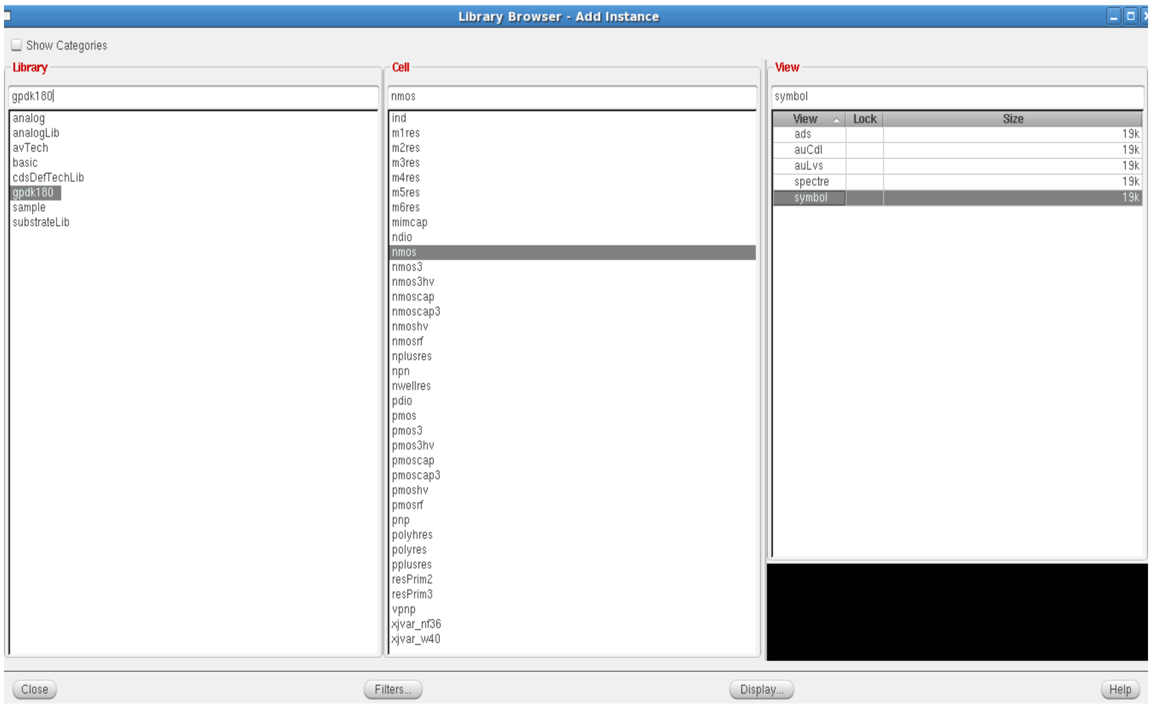
Open with: Schematics L

☐ Always use this application for this type of file

Library path file

/home/cadence_ms_labs_614/cds.lib

- Virtuoso Schematic Editor will open.
- Select Create -> Instance, and select the required components (pmos & nmos symbols)



Library Browser - Add Instance

☐ Show Categories

Library: gpdk180

analog
analogLib
avTech
basic
cdsDefTechLib
gpdk180
sample
substrateLib

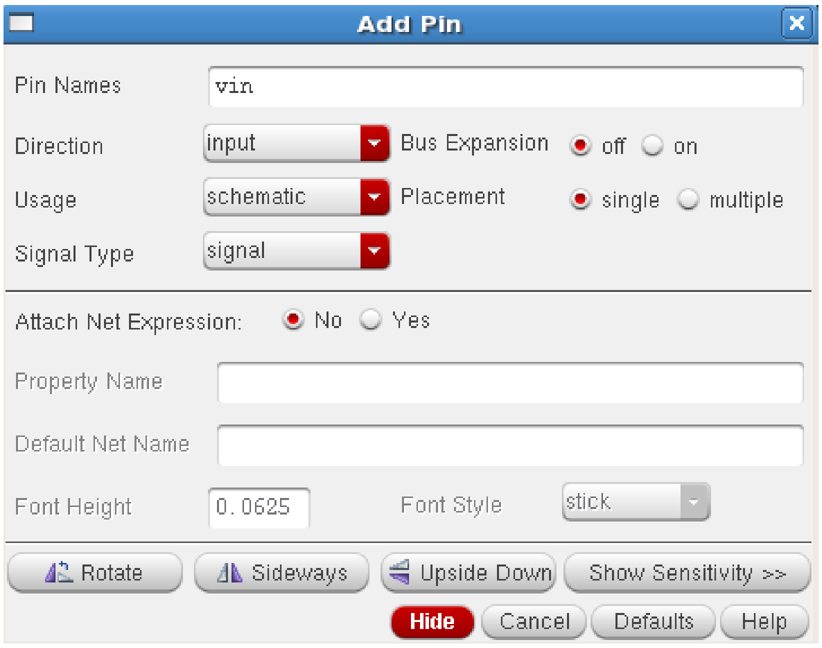
Cell: nmos

ind
m1res
m2res
m3res
m4res
m5res
m6res
mimcap
ndio
nmos
nmos3
nmos3hv
nmoscap
nmoscap3
nmoshv
nmosrf
nplusres
npn
nwellres
pdio
pmos
pmos3
pmos3hv
pmoscap
pmoscap3
pmoshv
pmosrf
pnp
polyhres
polyres
pplusres
resPrim2
resPrim3
vpnp
xjvar_nf36
xjvar_w40

View: symbol

View	Lock	Size
ads		19k
auCdl		19k
auLvs		19k
spectre		19k
symbol		19k

- Select **Create -> pins**, and place the ports (vin, vdd, gnd, vout) on the editor.



Add Pin

Pin Names: vin

Direction: input | Bus Expansion: [✓ off] on

Usage: schematic | Placement: [✓ single] multiple

Signal Type: signal

Attach Net Expression: [✓ No] Yes

Property Name:

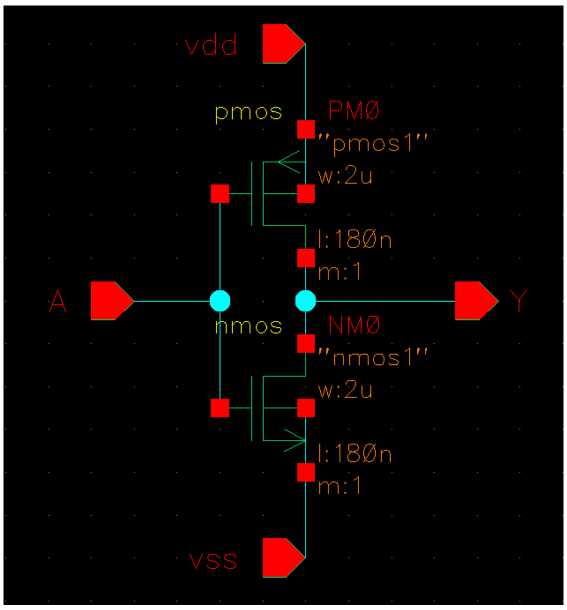
Default Net Name:

Font Height: 0.0625 | Font Style: stick

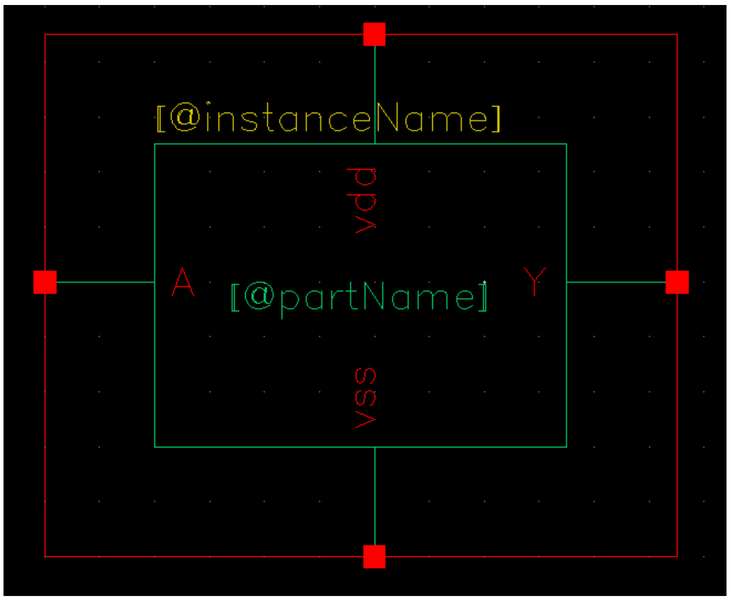
- Make the complete connection by using wires by selecting **create -> wire (narrow)**
- After complete of the schematic, save the file by **file -> check & save**

1.6 Observations:

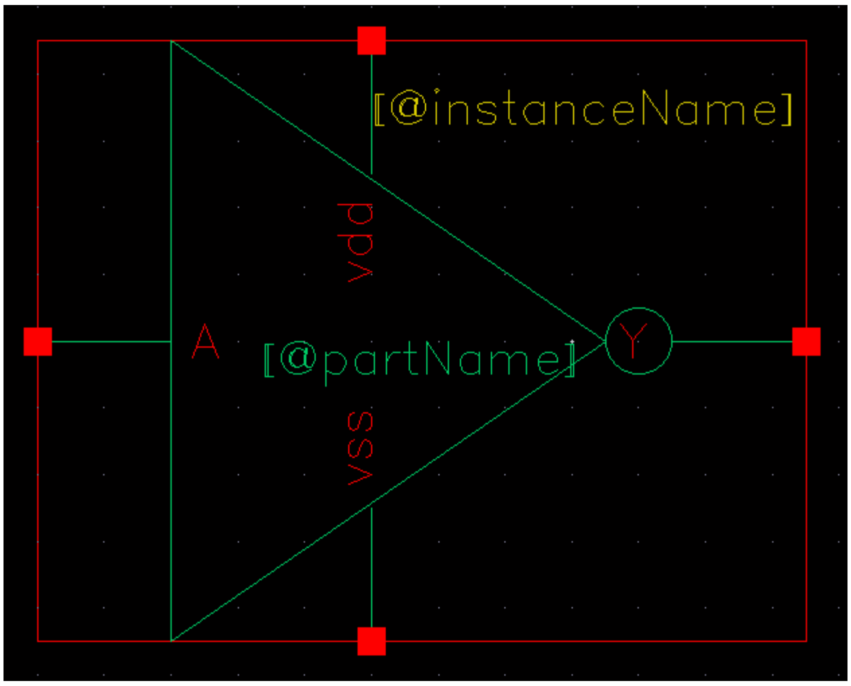
<u>SCHEMATIC</u>:



SYMBOL:



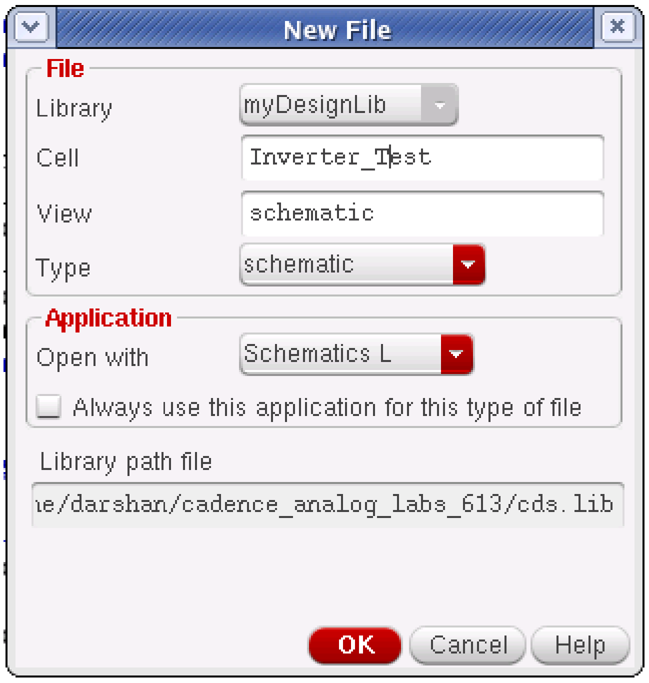
- To create symbol **Create -> Cell View -> From Cell view.**
- Virtuoso symbol editor opens.
- Place the right, left, bottom and top pins.
- Create shape of the cell using option in **Create -> Shapes.**



BUILDING THE TEST DESIGN FOR THE CELL

- In the library manager select given library

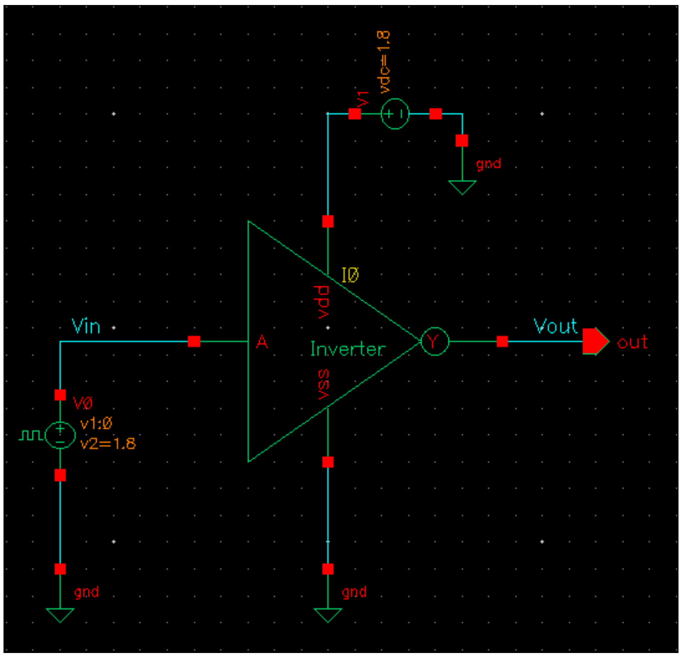
- Select **File -> New -> Cell view.**
- Give the cell name as **inverter_test.**



Library: myDesignLib
Cell: Inverter_Test
View: schematic
Type: schematic
Open with: Schematics L
Always use this application for this type of file
Library path file: ıe/darshan/cadence_analog_labs_613/cds.lib

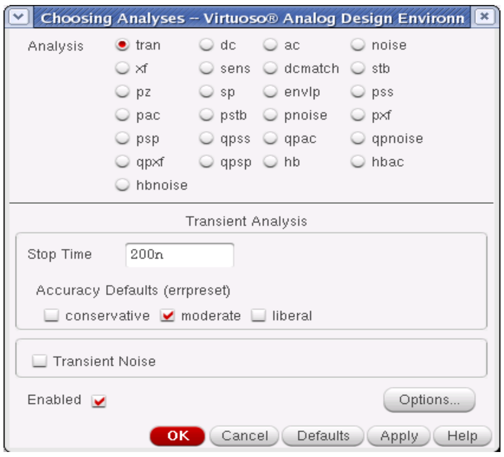
- Virtuoso Schematic editor window will open.
- Select **create -> instance,** and select the design to test.
- To apply inputs, **create -> instance.** In the library select **analoglib** and select appropriate inputs (vpulse, vdc, gnd).

Library name	**Cellview name**	**Properties/Comments**
Scem	Inverter	Symbol
analogLib	vpulse	v1=0, v2=1.8,td=0 tr=tf=1ns, ton=10n, T=20n
analogLib	vdc, gnd	vdc=1.8



STEPS TO SIMULATE

- In the Virtuoso Schematic Editor after completing the test circuit design **launch->ADEL**
- Virtuoso Analog Design Environment window opens.
- Go to **Analyses -> choose,** give appropriate inputs for **tran** for transient analyses and **dc** for dc analyses.


Choosing Analyses -- Virtuoso® Analog Design Environn

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

Transient Analysis

Stop Time 200n

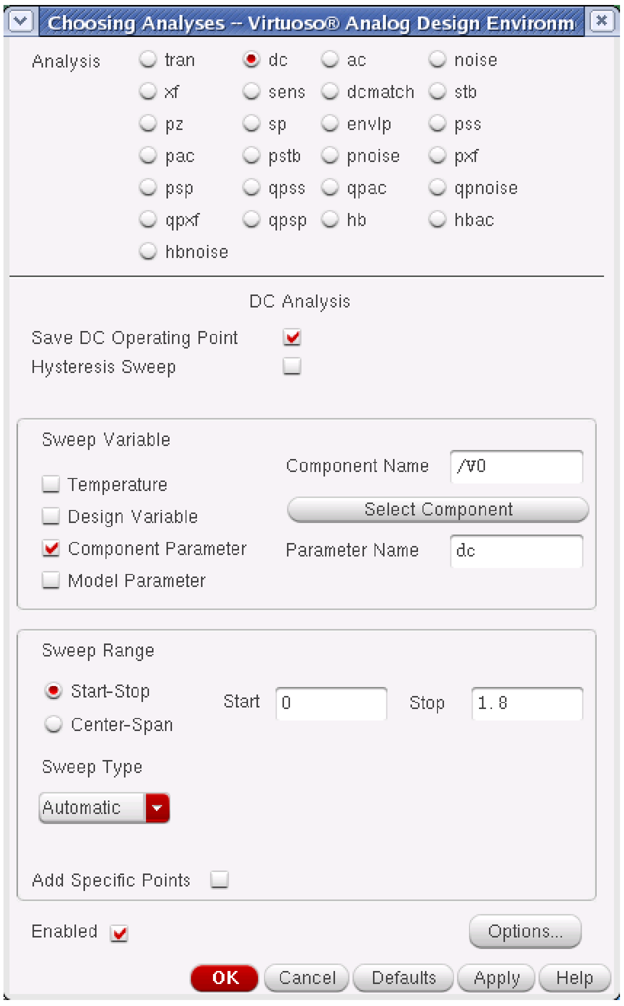
Accuracy Defaults (errpreset)

conservative [✓ moderate] liberal

Transient Noise

Enabled ✓

Options... OK Cancel Defaults Apply Help



Choosing Analyses -- Virtuoso® Analog Design Environm

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

DC Analysis

Save DC Operating Point ✓
Hysteresis Sweep ☐

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

Component Name: /V0
Select Component
Parameter Name: dc

Sweep Range
[✓ Start-Stop] ○ Center-Span
Start: 0 Stop: 1. 8

Sweep Type: Automatic

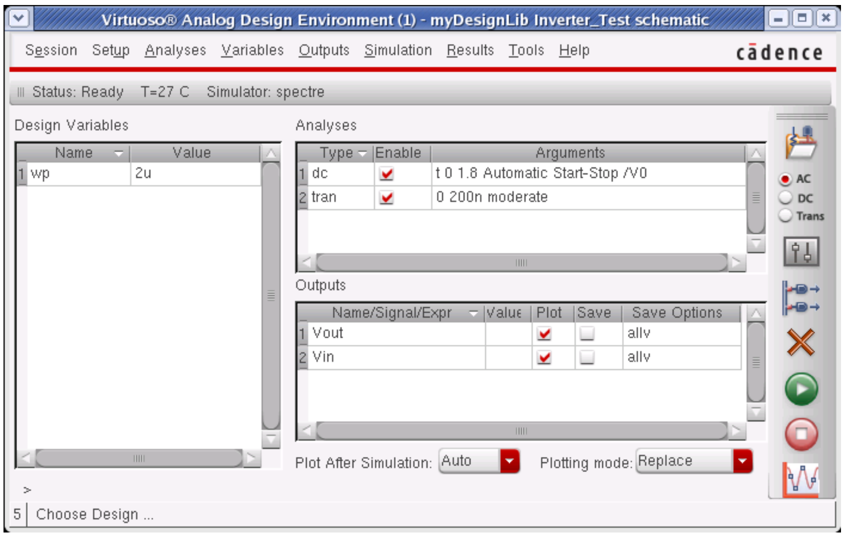
Add Specific Points ☐

Enabled ✓

Options...

OK Cancel Defaults Apply Help

- Go to **outputs -> to be plotted -> select on schematic** (input and output wires).
- Go to **simulation -> netlist and run**
- When simulation finishes, the Transient, DC plots automatically will be popped up along with log file.



Virtuoso® Analog Design Environment (1) - myDesignLib Inverter_Test schematic

Session Setup Analyses Variables Outputs Simulation Results Tools Help cādence

Status: Ready T=27 C Simulator: spectre

Design Variables

	Name	Value
1	wp	2u

Analyses

	Type	Enable	Arguments
1	dc	✓	t 0 1.8 Automatic Start-Stop /V0
2	tran	✓	0 200n moderate

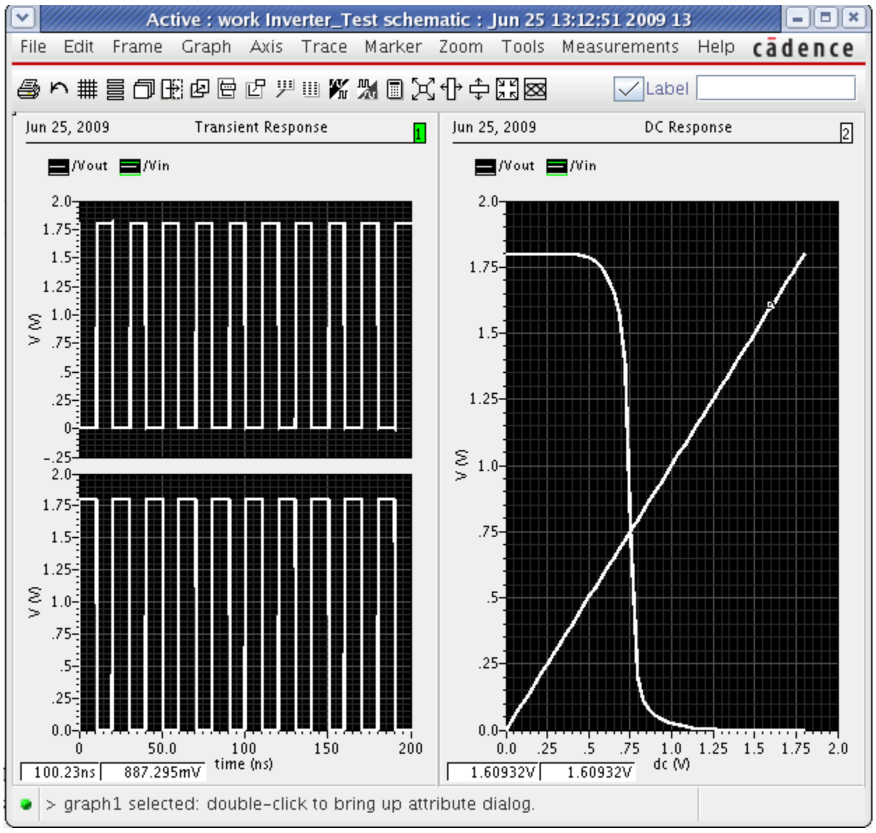
Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	Vout		✓		allv
2	Vin		✓		allv

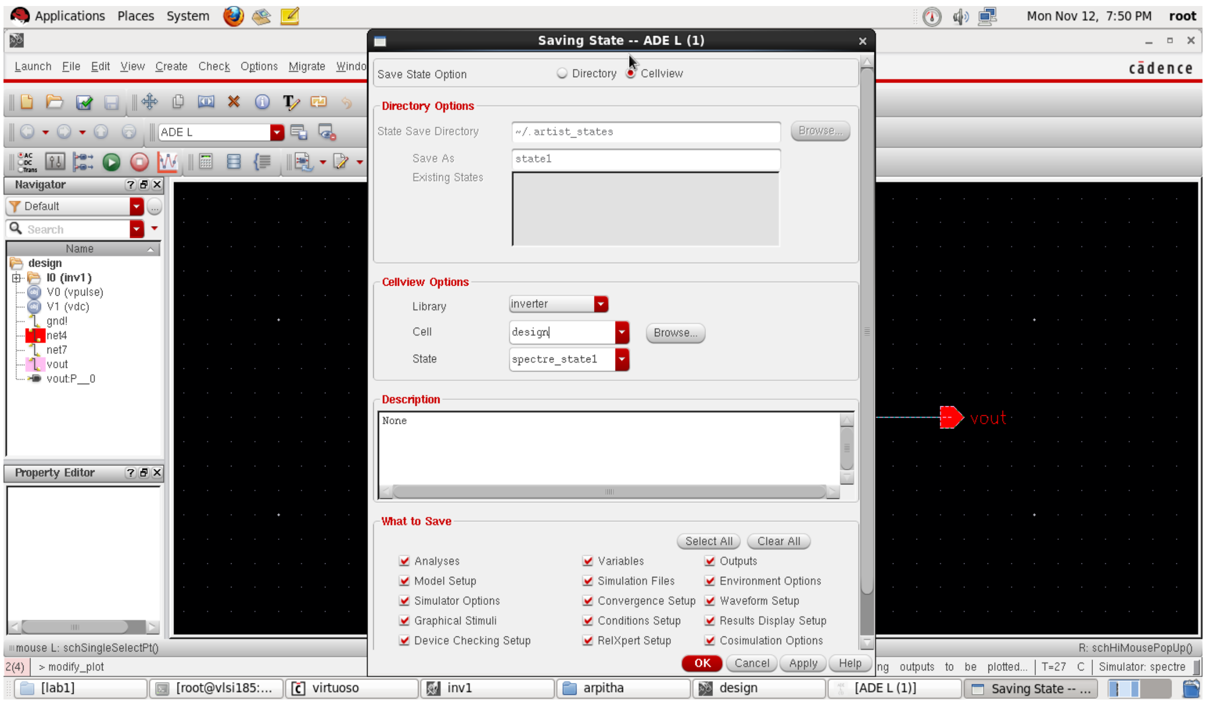
Plot After Simulation: Auto Plotting mode: Replace

[✓ AC] DC Trans

5 Choose Design ...



- To save the waveform, go to **session** in the ADEL window and in session select **save state**, under it click on **cell view** and then okay.


Applications Places System
Mon Nov 12, 7:50 PM root
Launch File Edit View Create Check Options Migrate Windo
cādence
ADE L
Navigator
Default
Search
Name
design
I0 (inv1)
V0 (vpulse)
V1 (vdc)
gnd!
net4
net7
vout
vout:P__0
Property Editor
vout
mouse L: schSingleSelectPt()
R: schHiMousePopUp()
2(4) > modify_plot
T=27 C Simulator: spectre
[lab1] [root@vlsi185:... virtuoso inv1 arpitha design [ADE L (1)] Saving State -- ...

Saving State -- ADE L (1)

Save State Option ○ Directory [● Cellview]

Directory Options

State Save Directory ~/.artist_states Browse...

Save As state1

Existing States

Cellview Options

Library inverter

Cell design Browse...

State spectre_state1

Description

None

What to Save Select All Clear All

- ✓ Analyses
- ✓ Variables
- ✓ Outputs
- ✓ Model Setup
- ✓ Simulation Files
- ✓ Environment Options
- ✓ Simulator Options
- ✓ Convergence Setup
- ✓ Waveform Setup
- ✓ Graphical Stimuli
- ✓ Conditions Setup
- ✓ Results Display Setup
- ✓ Device Checking Setup
- ✓ RelXpert Setup
- ✓ Cosimulation Options

OK Cancel Apply Help

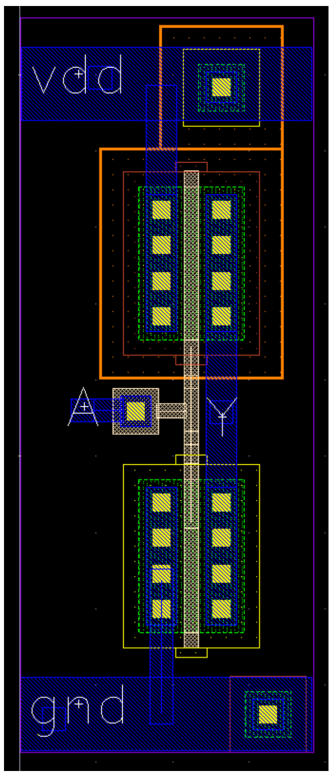
STEPS FOR CREATING LAYOUT VIEW

- From the **Inverter** schematic window menu execute **Launch – Layout XL**. A **Startup Option** form appears.
- Select **Create New** option. This gives a New Cell View Form
- Check the Cellname **(Inverter)**, Viewname **(layout).**
- Click **OK** from the New Cellview form.
- LSW and a blank layout window appear along with schematic window.

Adding Components to Layout

- Execute **Connectivity – Generate – All from Source** or click the icon in the layout editor window, **Generate Layout** form appears. Click **OK** which imports the schematic components in to the Layout window automatically.
- Re arrange the components with in PR-Boundary as shown in the next page.
- Press Shift –f in the layout window to display all the levels.
- To rotate a component, Select the component and execute **Edit –Properties**. Now select the degree of rotation from the property edit form.

- To Move a component, Select the component and execute **Edit -Move** command.



PHYSICAL VERIFICATION

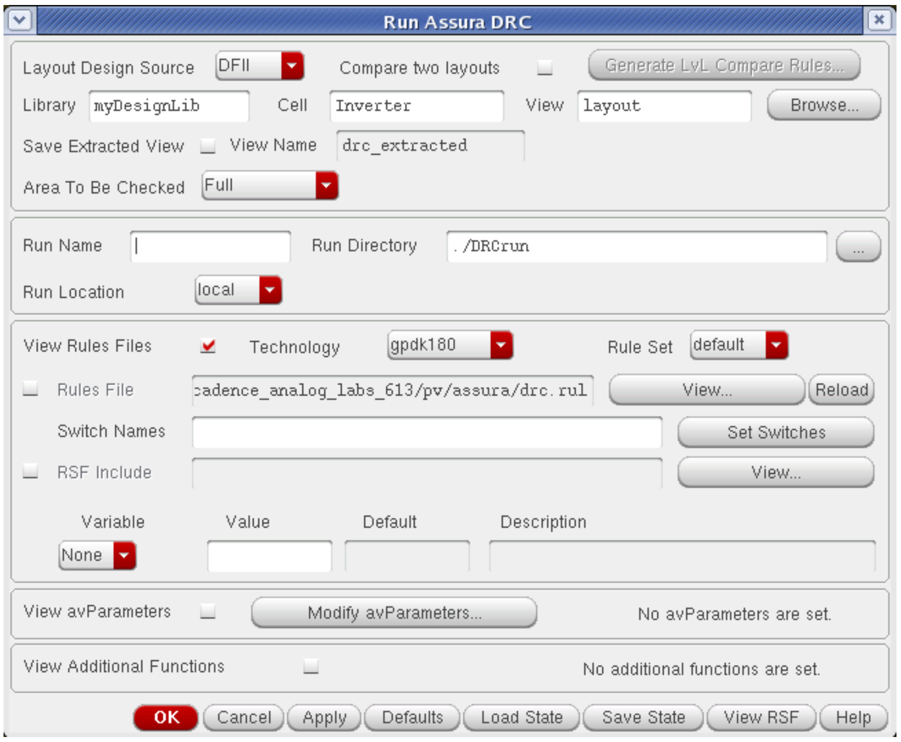
Select **Assura**⟶ **technology** ⟶**Browse**⟶ click on previous directory**(..)** until we get a **cad** option ⟶ double click on cad ⟶**foundary**⟶**analog** ⟶**180nm**⟶**assura.tech.lib**⟶**okay.**

Assura DRC

Running a DRC

- Open the Inverter layout form the CIW or library manger if you have closed that.
- Select **Assura - Run DRC** from layout window.
- The DRC form appears. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as **gpdk180.** This automatically loads the rule file.

Your DRC form should appear like this



Run Assura DRC

Layout Design Source: DFII
Compare two layouts: ☐
Library: myDesignLib
Cell: Inverter
View: layout
Save Extracted View: ☐
View Name: drc_extracted
Area To Be Checked: Full
Run Name:
Run Directory: ./DRCrun
Run Location: local
View Rules Files: ✓
Technology: gpdk180
Rule Set: default
Rules File: ☐ cadence_analog_labs_613/pv/assura/drc.rul
Switch Names:
RSF Include: ☐
Variable: None
View avParameters: ☐ — No avParameters are set.
View Additional Functions: ☐ — No additional functions are set.

- Click **OK** to start DRC.
- A Progress form will appears. You can click on the watch log file to see the log file.
- When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click **Yes** to view the results of this run.
- If there any DRC error exists in the design **View Layer Window** (VLW) and **Error Layer Window** (ELW) appears. Also the errors highlight in the design itself.
- Click **View – Summary** in the ELW to find the details of errors.
- You can refer to rule file also for more information, correct all the DRC errors and **Re – run** the DRC.
- If there are no errors in the layout then a dialog box appears with **No DRC errors found** written in it, click on **close** to terminate the DRC run.

ASSURA LVS

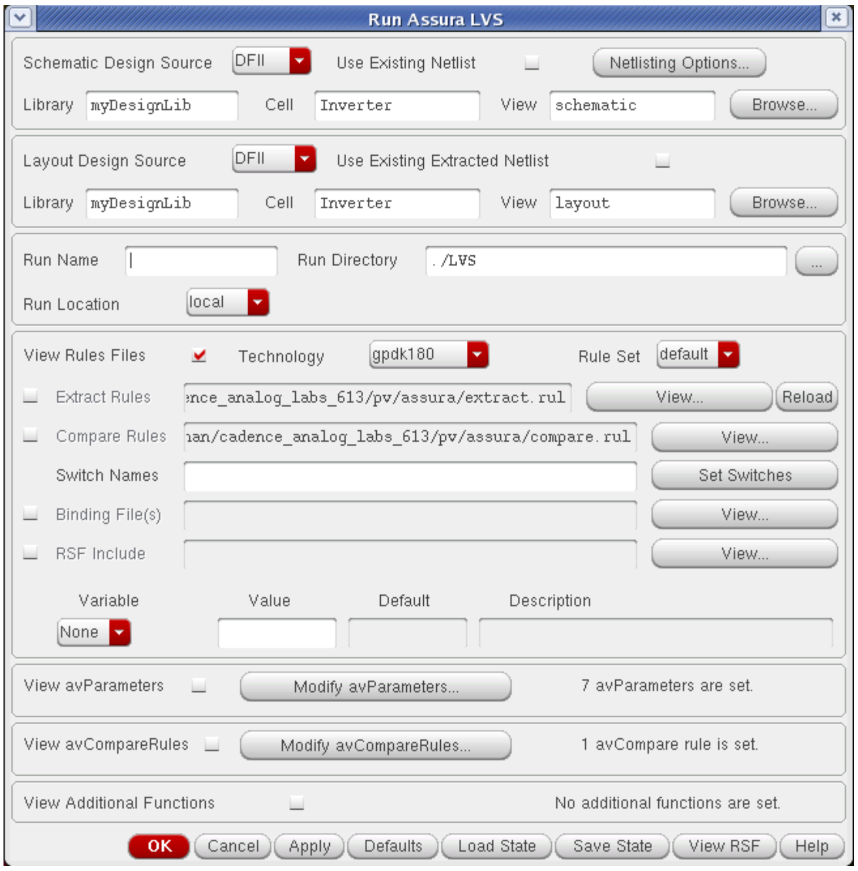
In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS

- Select **Assura – Run LVS** from the layout window.

The Assura Run LVS form appears. It will automatically load both the schematic and layout view of the cell.

- Change the following in the form and click OK.



Run Assura LVS

Schematic Design Source	DFII	Use Existing Netlist	☐	Netlisting Options...	
Library	myDesignLib	Cell	Inverter	View	schematic
Layout Design Source	DFII	Use Existing Extracted Netlist	☐		
Library	myDesignLib	Cell	Inverter	View	layout
Run Name		Run Directory	./LVS		
Run Location	local				
View Rules Files	✓	Technology	gpdk180	Rule Set	default
☐ Extract Rules	ence_analog_labs_613/pv/assura/extract.rul				
☐ Compare Rules	ian/cadence_analog_labs_613/pv/assura/compare.rul				
Switch Names					
☐ Binding File(s)					
☐ RSF Include					

Variable	Value	Default	Description
None			

View avParameters ☐ — 7 avParameters are set.

View avCompareRules ☐ — 1 avCompare rule is set.

View Additional Functions ☐ — No additional functions are set.

- The LVS begins and a Progress form appears.
- If the schematic and layout matches completely, you will get the form displaying **Schematic and Layout Match**.
- If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.
- Click **Yes** in the form.

LVS debug form appears, and you are directed into LVS debug environment.

- In the **LVS debug form** you can find the details of mismatches and you need to correct all those mismatches and **Re – run** the LVS till you will be able to match the schematic with layout.

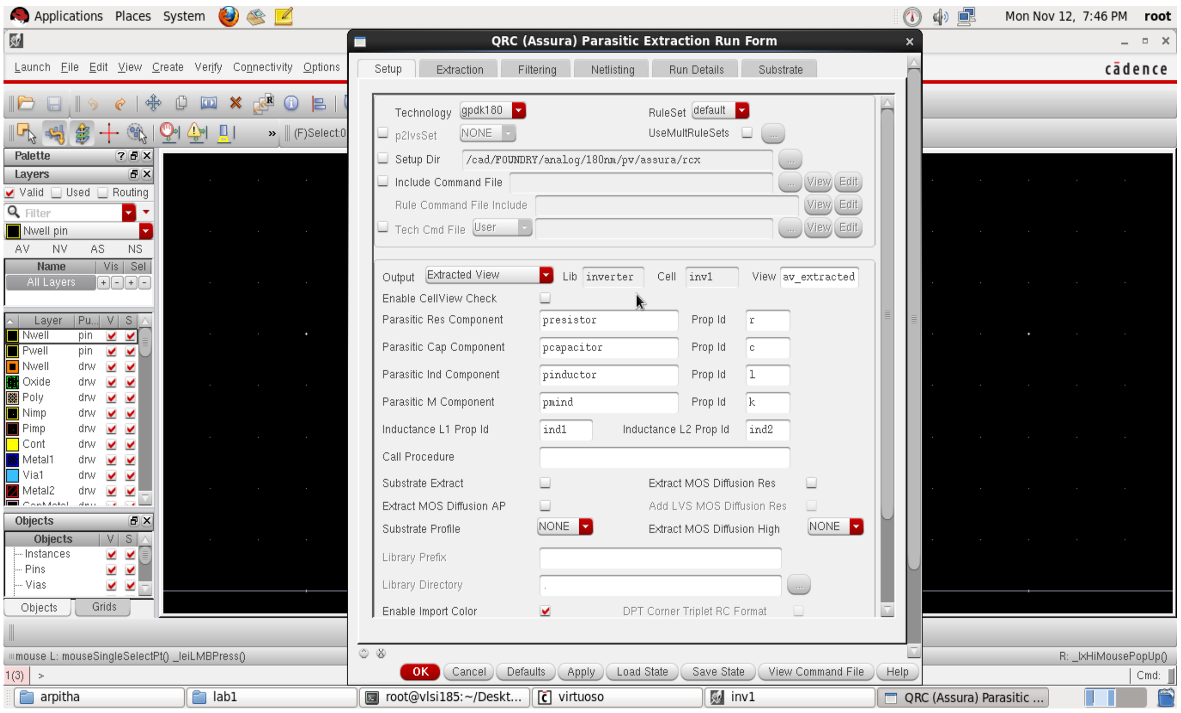
ASSURA RCX

In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.

Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

Running QRC

- From the layout window execute **Assura – Run QRC**
- In QRC window,under **set up** option select **output as extracted view**,
- Under **Extraction** option,keep **extracted type as RC** and **ref node as gnd! Or vss!**

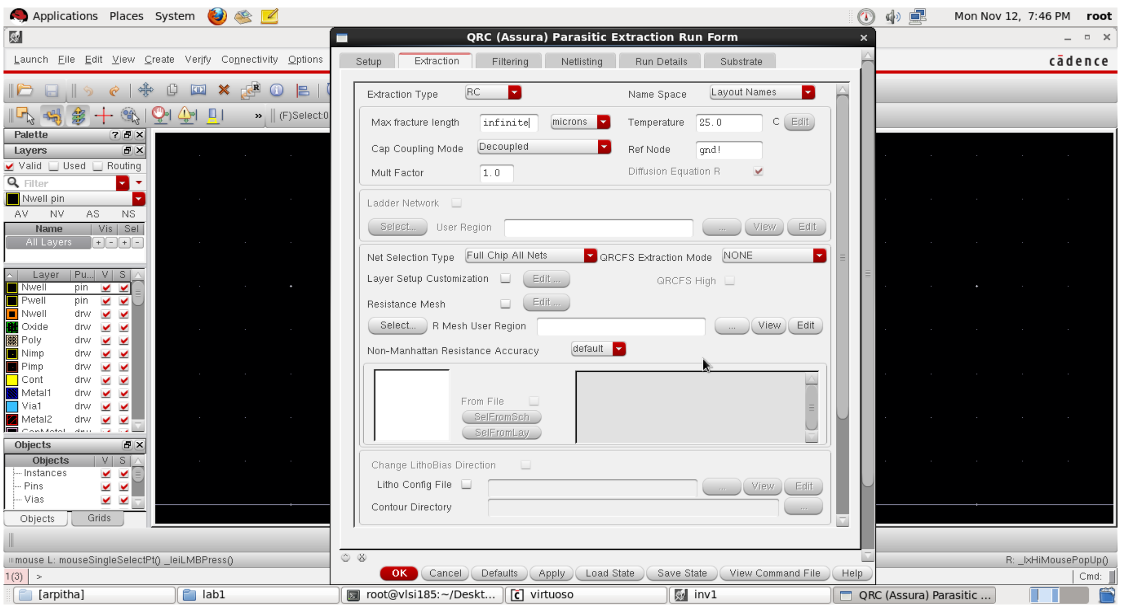


QRC (Assura) Parasitic Extraction Run Form

Setup | Extraction | Filtering | Netlisting | Run Details | Substrate

Technology: gpdk180 RuleSet: default
p2lvsSet: NONE UseMultRuleSets
Setup Dir: /cad/FOUNDRY/analog/180nm/pv/assura/rcx
Include Command File
Rule Command File Include
Tech Cmd File: User

Output: Extracted View Lib: inverter Cell: inv1 View: av_extracted
Enable CellView Check
Parasitic Res Component: presistor Prop Id: r
Parasitic Cap Component: pcapacitor Prop Id: c
Parasitic Ind Component: pinductor Prop Id: l
Parasitic M Component: pmind Prop Id: k
Inductance L1 Prop Id: ind1 Inductance L2 Prop Id: ind2
Call Procedure
Substrate Extract Extract MOS Diffusion Res
Extract MOS Diffusion AP Add LVS MOS Diffusion Res
Substrate Profile: NONE Extract MOS Diffusion High: NONE
Library Prefix
Library Directory: .
Enable Import Color ✓ DPT Corner Triplet RC Format

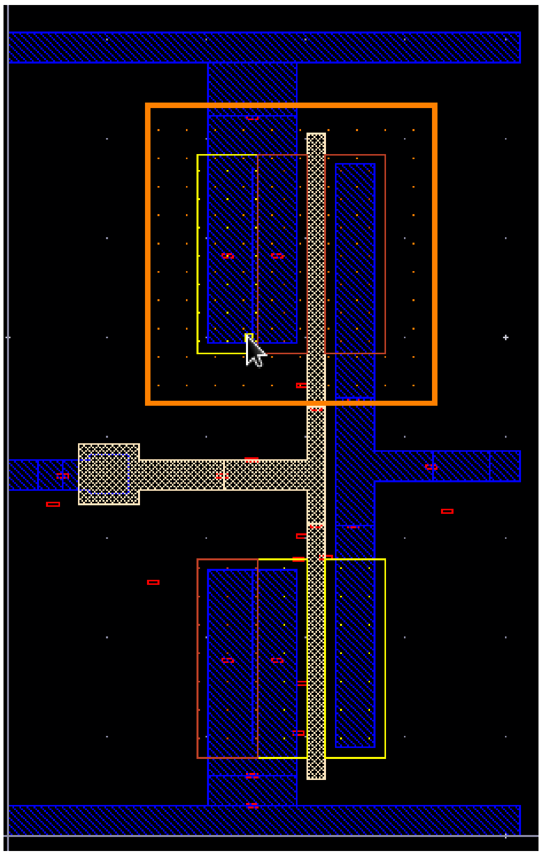


QRC (Assura) Parasitic Extraction Run Form

Setup | Extraction | Filtering | Netlisting | Run Details | Substrate

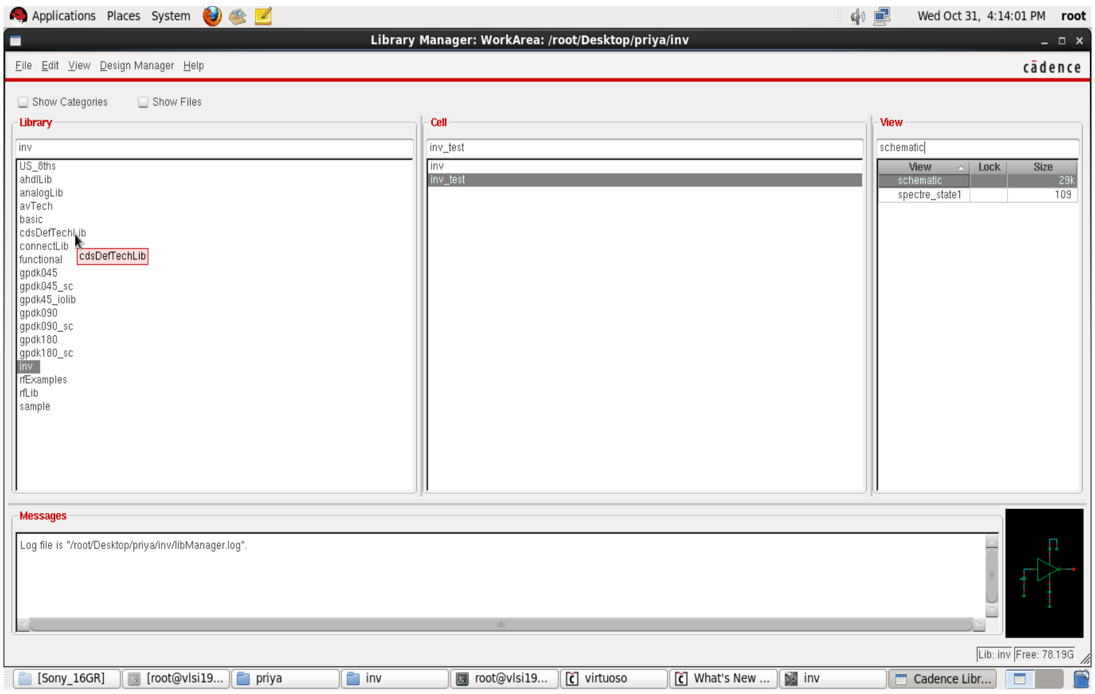
Extraction Type: RC Name Space: Layout Names
Max fracture length: infinite microns Temperature: 25.0 C
Cap Coupling Mode: Decoupled Ref Node: gnd!
Mult Factor: 1.0 Diffusion Equation R ✓
Ladder Network
User Region
Net Selection Type: Full Chip All Nets QRCFS Extraction Mode: NONE
Layer Setup Customization QRCFS High
Resistance Mesh
R Mesh User Region
Non-Manhattan Resistance Accuracy: default
From File
Change LithoBias Direction
Litho Config File
Contour Directory

AvExtracted View



Steps to obtain Back Annotate Wave

- Select the given library name, and select the schematic test.



Applications Places System Wed Oct 31, 4:14:01 PM root

Library Manager: WorkArea: /root/Desktop/priya/inv

File Edit View Design Manager Help cādence

Show Categories Show Files

Library: inv

US_8ths
ahdlLib
analogLib
avTech
basic
cdsDefTechLib
connectLib
functional
gpdk045
gpdk045_sc
gpdk45_iolib
gpdk090
gpdk090_sc
gpdk180
gpdk180_sc
inv
rfExamples
rfLib
sample

cdsDefTechLib

Cell: inv_test

inv
inv_test

View: schematic

View	Lock	Size
schematic		29k
spectre_state1		109

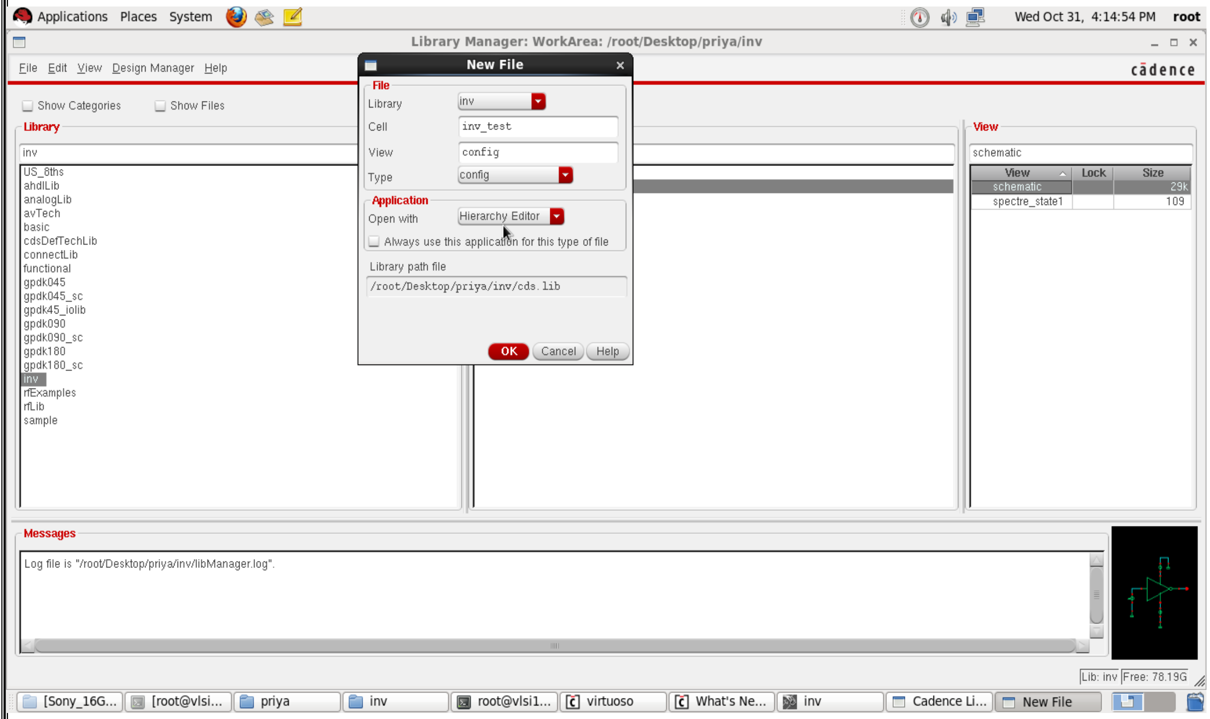
Messages

```
Log file is "/root/Desktop/priya/inv/libManager.log".
```

Lib: inv Free: 78.19G

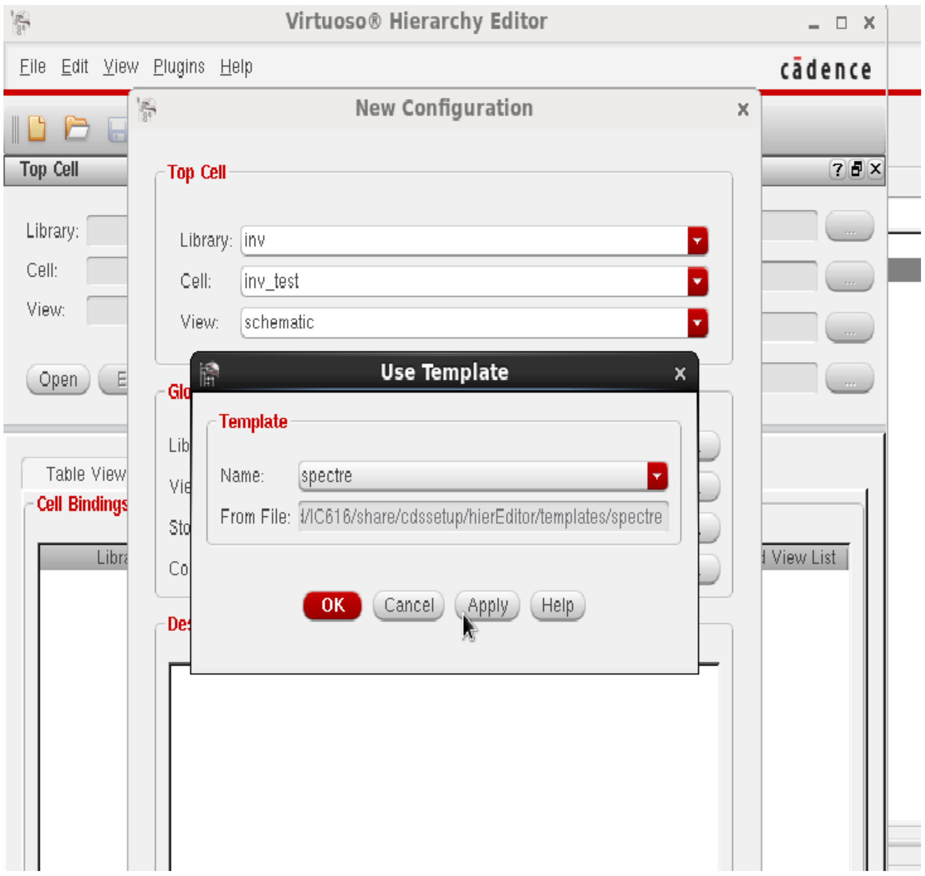
[Sony_16GR] [root@vlsi19... priya inv root@vlsi19... virtuoso What's New ... inv Cadence Libr...

- Go to library manager, click on **New** →**cell view** →keep type as **config**

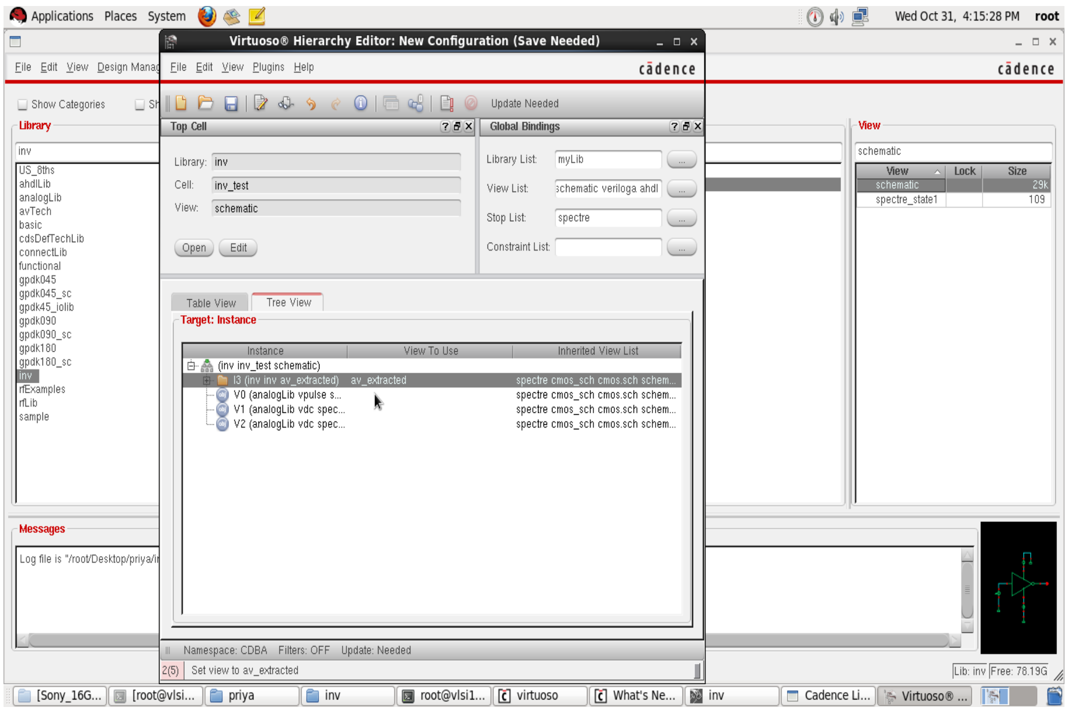

Applications Places System
Wed Oct 31, 4:14:54 PM root
Library Manager: WorkArea: /root/Desktop/priya/inv
File Edit View Design Manager Help
cādence
Show Categories
Show Files
Library
View
New File
File
Library inv
Cell inv_test
View config
Type config
Application
Open with Hierarchy Editor
Always use this application for this type of file
Library path file
/root/Desktop/priya/inv/cds.lib
OK Cancel Help
Messages
Lib: inv Free: 78.19G

Log file is "/root/Desktop/priya/inv/libManager.log".

- Below window will appear, **select view as schematic** and **select use template**, under it **select spectre** → **okay**



- In Hierarchy editor window click on **Tree View**, right click on the folder, click on **av_extracted.**



Virtuoso® Hierarchy Editor: New Configuration (Save Needed)

Top Cell
Library: inv
Cell: inv_test
View: schematic

Global Bindings
Library List: myLib
View List: schematic veriloga ahdl
Stop List: spectre
Constraint List:

Target: Instance

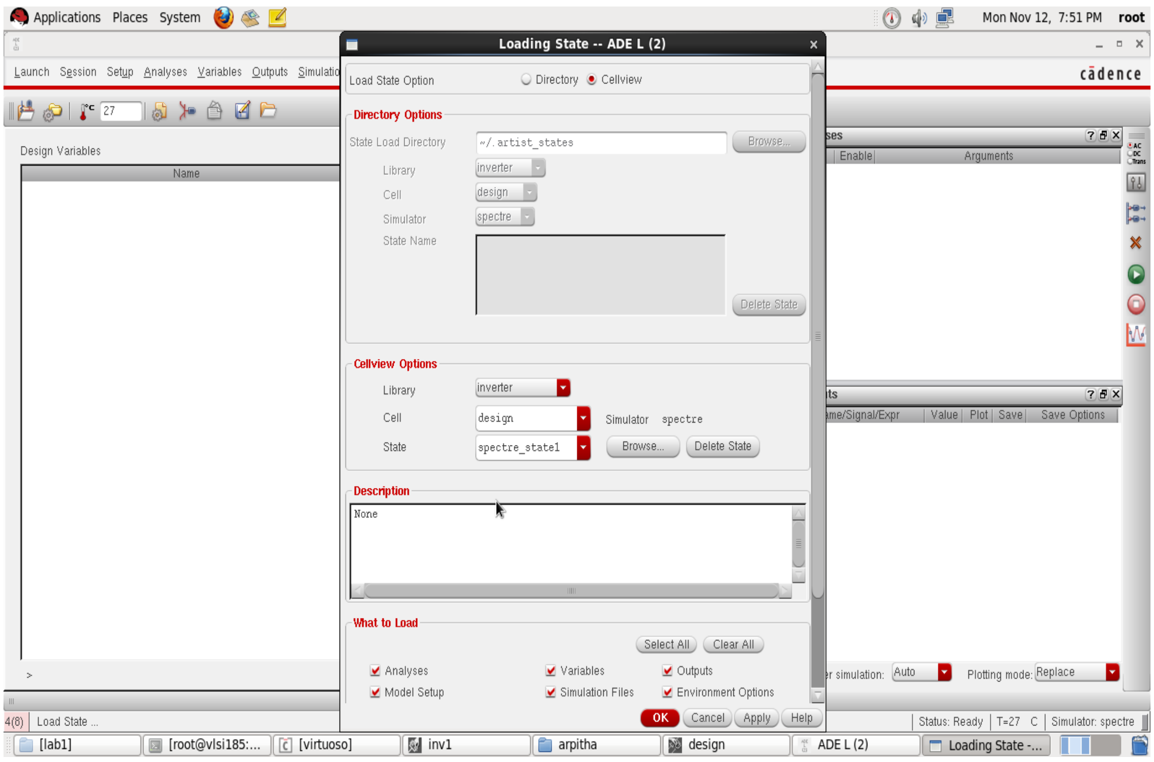
Instance	View To Use	Inherited View List
(inv inv_test schematic)		
I3 (inv inv av_extracted)	av_extracted	spectre cmos_sch cmos.sch schem...
V0 (analogLib vpulse s...		spectre cmos_sch cmos.sch schem...
V1 (analogLib vdc spec...		spectre cmos_sch cmos.sch schem...
V2 (analogLib vdc spec...		spectre cmos_sch cmos.sch schem...

Namespace: CDBA Filters: OFF Update: Needed
2(5) Set view to av_extracted

Log file is "/root/Desktop/priya/i

View	Lock	Size
schematic		29k
spectre_state1		109

- Click on Open→(schematic test window will be opened)→launch ADEL
- To load the simulated graph ,In ADEL Click on session →load state →cell view →okay



Loading State -- ADE L (2)

Load State Option ( ) Directory (●) Cellview

Directory Options
State Load Directory: ~/.artist_states
Library: inverter
Cell: design
Simulator: spectre
State Name

Cellview Options
Library: inverter
Cell: design
Simulator: spectre
State: spectre_state1

Description
None

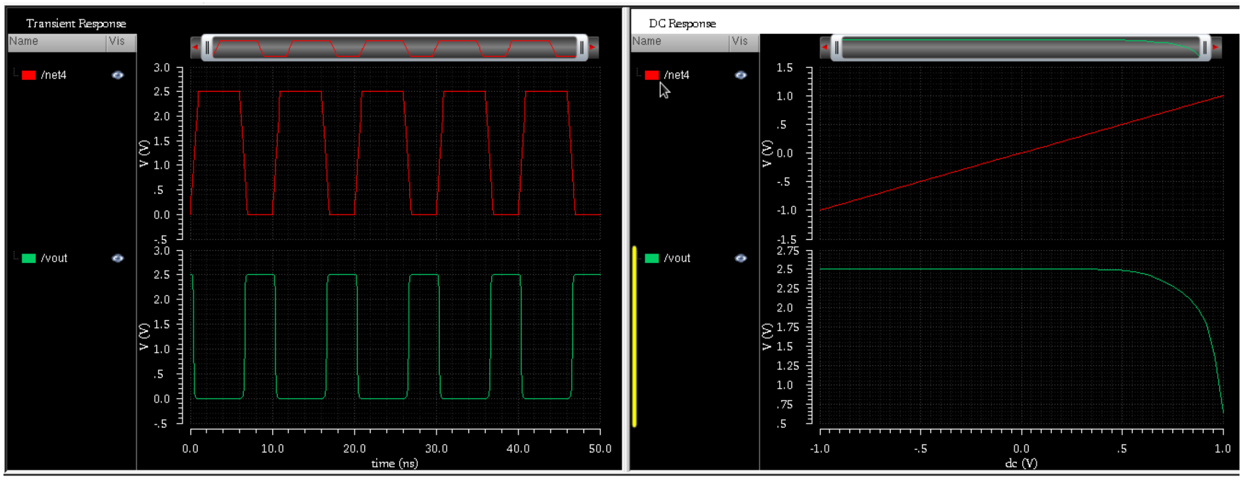
What to Load
✓ Analyses ✓ Variables ✓ Outputs
✓ Model Setup ✓ Simulation Files ✓ Environment Options

OK Cancel Apply Help

4(8) Load State ...
Status: Ready T=27 C Simulator: spectre

- Once the schematic waveform is loaded, Click on the run symbol in ADEL window to obtain back annotate wave.

1.7 Results :



1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Explain MOSFET, CMOS
2. Explain W/L ratio.
3. What is gpdk.
4. What is body effect.
5. What are the different rules followed in LAYOUT design and what are the importance of the same.

1.10 Post – Experimentation Questions:

1. Explain the inverter characteristics transient response.
2. Explain the DC analysis of inverter.
3. What is the importance of DRC
4. What is ERC.
5. Discuss the need of back annotation.

EXPERIMENT No. 02: Design the (i) Common source and Common Drain amplifier and (ii) A Single Stage differential amplifier, with given specifications**, completing the design flow mentioned below:

a. Draw the schematic and verify the following

i) DC Analysis ii) AC Analysis iii) Transient Analysis

b. Draw the Layout and verify the DRC, ERC

c. Check for LVS

d. Extract RC and back annotate the same and verify the Design.

2.1 Objectives:

Design Common source, common drain and single stage differential amplifier with given specifications and complete design till layout and verify the parameters DRC, ERC, QRC. Extract RC and back annotate for the verification.

2.2 Apparatus Required:

Cadence Tools.

2.3 Pre-Requisite:

Knowledge of tool flow of cadence, design rules and working of amplifiers.

2.4 Introduction:

The common-source amplifier is one of the basic amplifiers in CMOS analog circuits. Because of its very high input impedance, relatively high gain, low noise, speed, and simplicity, common-source amplifiers find different applications from sensor signal amplification to RF low-noise amplification. Good understanding of this amplifier is essential not only for good design of one but also for analysis of other advanced circuits such as differential amplifiers.

A common-drain amplifier is one in which the input signal is applied to the gate and the output is taken from the source, making the drain common to both. Because it is common, there is no need for a drain resistor. A common-drain amplifier is shown below. A common-drain amplifier is also called a source-follower. Self-biasing is used in this particular circuit. The input signal is applied to the gate through a coupling capacitor, and the output signal is coupled to the load resistor through the other capacitor.

The differential amplifier boosts the difference between the two input signals demonstrated as V+ and V- and rejects any common signals from the two sources. Its ideal characteristics are infinite bandwidth, gain, and common mode rejection ratio (CMRR); and high input impedance, less distortion, and low output admittance. It is also characterized to have less harmonic distortion and high output voltage swing. Its characteristics test the effective performance of the circuit. These amplifiers are broadly used in linear amplification circuits to obtain less distortion at the output. They can be designed in multiple ways, providing a result where output may be single or double ended. However, the most commonly used in designing a differential amplifier is the double ended, wherein two inputs provide two outputs, called a fully differential amplifier. Its advantage over single ended is simple biasing, high linearity, and high immunity to noise. But it has a large area to cover.

2.5 Procedure:

STEPS FOR THE ANALOG LAB

- Login to **redhat** operating system with the respective login name and password provided in the lab.
- Right click on the screen and select **Open Terminal** to open the command prompt.
- To enter into csh shell type **csh** in the command line.
- To source the cadence tools type **source /cshrc/cad** in the command line.
- To open the tool type **virtuoso &** in the command line.

Creating the Library

- Creating a new library by going to the link **File -> New -> Library**
- Give the library name and click ok.
- Select **attach to an existing technology library** and click **ok.**
- Select **gpdk180** and click **ok.**

SCHEMATIC AND SYMBOL CREATION

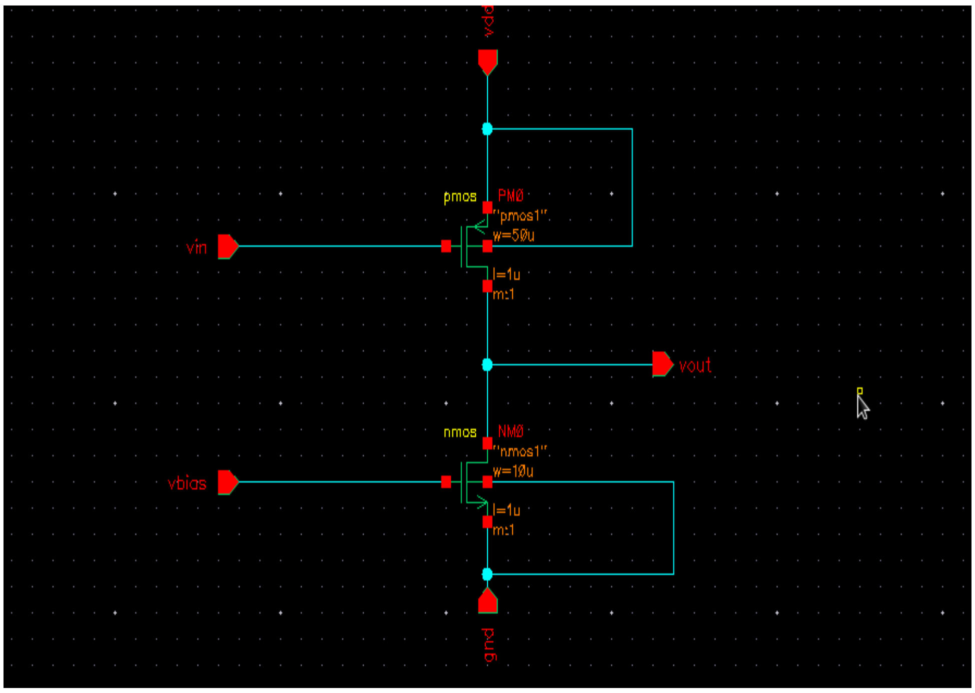
- In the Library manager select given library
- In Library manager select **File -> New -> Cell view.**
- Give the cell name as csamp.
- Virtuoso Schematic Editor will open.
- Select Create -> Instance, and select the required components (pmos & nmos symbols)
- Select **Create -> pins**, and place the ports (vin, vdd, gnd, vout) on the editor.
- Make the complete connection by using wires by selecting **create -> wire (narrow)**
- After complete of the schematic, save the file by **file -> check & save**

2.6 Observations:

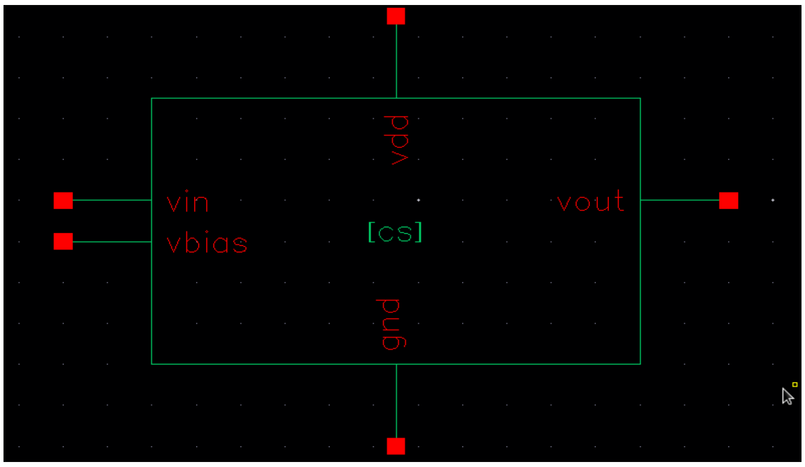
i) COMMON SOURCE AMPLIFIER & COMMON DRAIN AMPLIFIER

COMMON SOURCE AMPLIFIER

COMMON SOURCE SCHEMATIC



COMMON SOURCE SYMBOL



- To create symbol **Create -> Cell View -> From Cell view.**
- Virtuoso symbol editor opens.
- Place the right, left, bottom and top pins.
- Create shape of the cell using option in **Create -> Shapes.**

BUILDING THE TEST DESIGN FOR THE CELL

- In the library manager select given library
- Select **File -> New -> Cell view.**
- Give the cell name as **csamp_test.**
- Virtuoso Schematic editor window will open.
- Select **create -> instance,** and select the design to test.
- To apply inputs, **create -> instance.** In the library select **analoglib** and select appropriate inputs (vpulse, vdc, gnd).

Library name	Cell Name	Properties/Comments
gpdk180	pmos	Model Name = pmos1; W= 50u ; L= 1u
gpdk180	nmos	Model Name =nmos1; W= 10u ; L= 1u

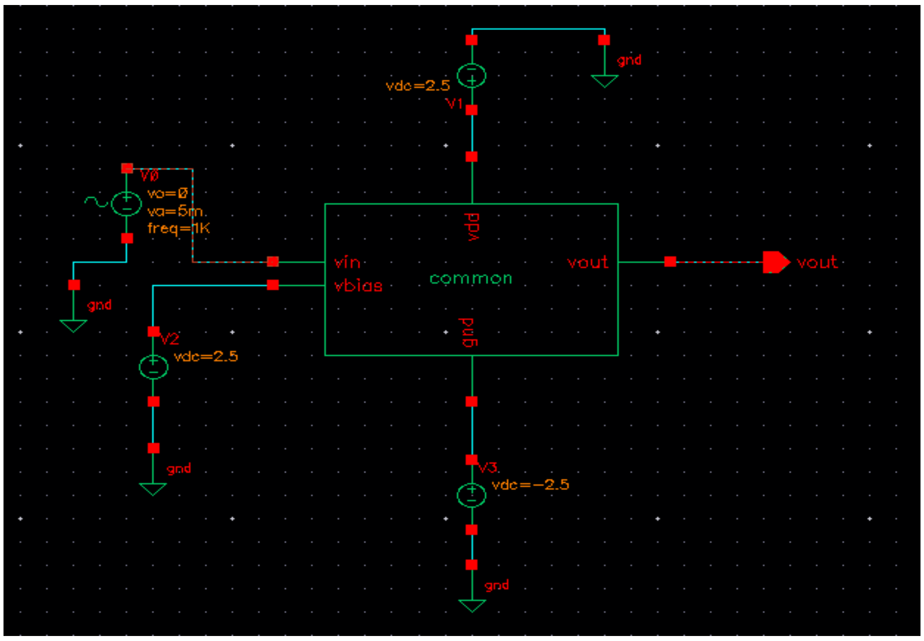
If the width is 50u or more than that, then divide the mos i.e transistors into parts so for that click on the nmos or pmos properties, and keep

Total Width=50u

Finger Width=25u

Fingers=2

COMMON SOURCE SYMBOL TEST



STEPS TO SIMULATE

- In the Virtuoso Schematic Editor after completing the test circuit design **launch->ADEL**
- Virtuoso Analog Design Environment window opens.
- Go to **Analyses -> choose,** give appropriate inputs for **tran** for transient analyses, **ac** for ac analyses and **dc** for dc analyses.
- Go to **outputs -> to be plotted -> select on schematic** (input and output wires).
- Go to **simulation -> netlist and run**
- When simulation finishes, the Transient, DC plots automatically will be popped up along with log file
- To save the waveform, go to **session** in the ADEL window and in session select **save state**, under it click on **cell view** and then okay.

STEPS FOR CREATING LAYOUT VIEW

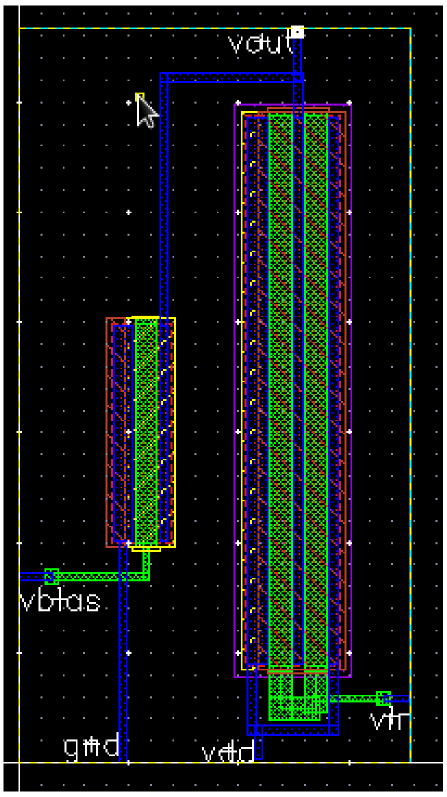
- From the **csamp** schematic window menu execute **Launch – Layout XL**. A **Startup Option** form appears.
- Select **Create New** option. This gives a New Cell View Form
- Check the Cellname **(csamp)**, Viewname **(layout).**

- Click **OK** from the New Cellview form.
- LSW and a blank layout window appear along with schematic window.

Adding Components to Layout

- Execute **Connectivity – Generate – All from Source** or click the icon in the layout editor window, **Generate Layout** form appears. Click **OK** which imports the schematic components in to the Layout window automatically.
- Re arrange the components with in PR-Boundary as shown in the next page.
- Press Shift –f in the layout window to display all the levels.
- To rotate a component, Select the component and execute **Edit –Properties**. Now select the degree of rotation from the property edit form.

- To Move a component, Select the component and execute **Edit -Move** command.



PHYSICAL VERIFICATION

Select **Assura**⟶ **technology** ⟶**Browse** ⟶ click on previous directory**(..)** until we get a **cad** option ⟶ double click on cad ⟶**foundary**⟶**analog** ⟶**180nm**⟶**assura.tech.lib**⟶**okay.**

Assura DRC

Running a DRC

- Open the Inverter layout form the CIW or library manger if you have closed that.
- Select **Assura - Run DRC** from layout window.
- The DRC form appears. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as **gpdk180.** This automatically loads the rule file.

Your DRC form should appear like this

- Click **OK** to start DRC.
- A Progress form will appears. You can click on the watch log file to see the log file.
- When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click **Yes** to view the results of this run.
- If there any DRC error exists in the design **View Layer Window** (VLW) and **Error Layer Window** (ELW) appears. Also the errors highlight in the design itself.
- Click **View – Summary** in the ELW to find the details of errors.
- You can refer to rule file also for more information, correct all the DRC errors and **Re – run** the DRC.
- If there are no errors in the layout then a dialog box appears with **No DRC errors found** written in it, click on **close** to terminate the DRC run.

ASSURA LVS

In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS

- Select **Assura – Run LVS** from the layout window.

The Assura Run LVS form appears. It will automatically load both the schematic and layout view of the cell.

- Change the following in the form and click OK.
- The LVS begins and a Progress form appears.
- If the schematic and layout matches completely, you will get the form displaying **Schematic and Layout Match**.

- If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.
- Click **Yes** in the form.

LVS debug form appears, and you are directed into LVS debug environment.

- In the **LVS debug form** you can find the details of mismatches and you need to correct all those mismatches and **Re – run** the LVS till you will be able to match the schematic with layout.

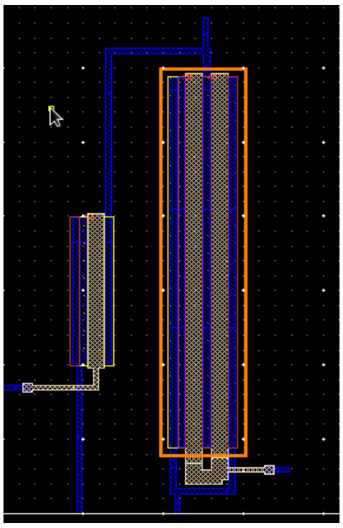
ASSURA RCX

In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.

Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

Running QRC

- From the layout window execute **Assura – Run QRC**
- In QRC window,under **set up** option select **output as extracted view**,
- Under **Extraction** option,keep **extracted type as RC** and **ref node as gnd! Or vss!**



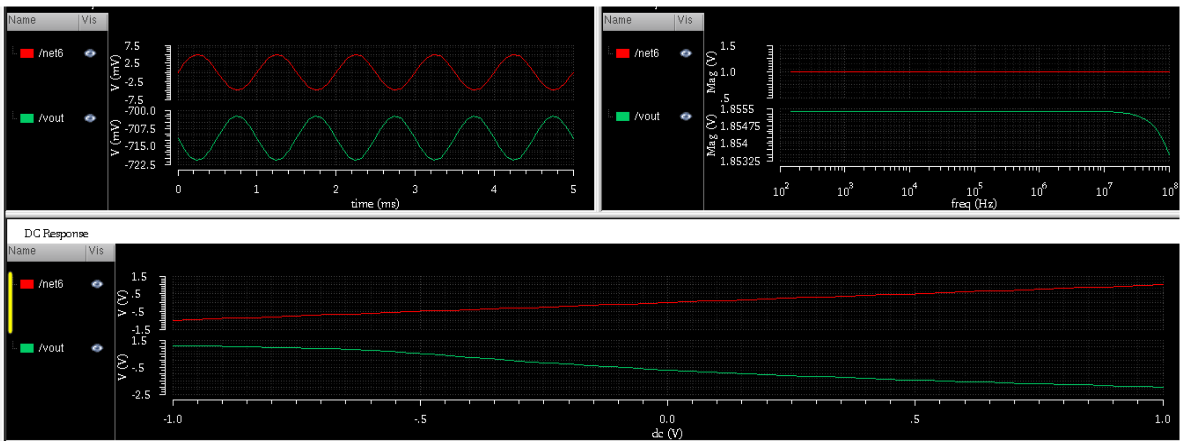
Steps to obtain Back Annotate Wave

- Select the given library name, and select the schematic test.
- Go to library manager, click on **New** →**cell view** →keep type as **config**
- Below window will appear, **select view as schematic** and **select use template**, under it **select spectre** → **okay**

- In Hierarchy editor window click on **Tree View**, right click on the folder, click on **av_extracted.**
- Click on Open→(schematic test window will be opened)→launch ADEL
- To load the simulated graph ,In ADEL Click on session →load state →cell view →okay
- Once the schematic waveform is loaded, Click on the run symbol in ADEL window to obtain back annotate wave.

2.7 Results :

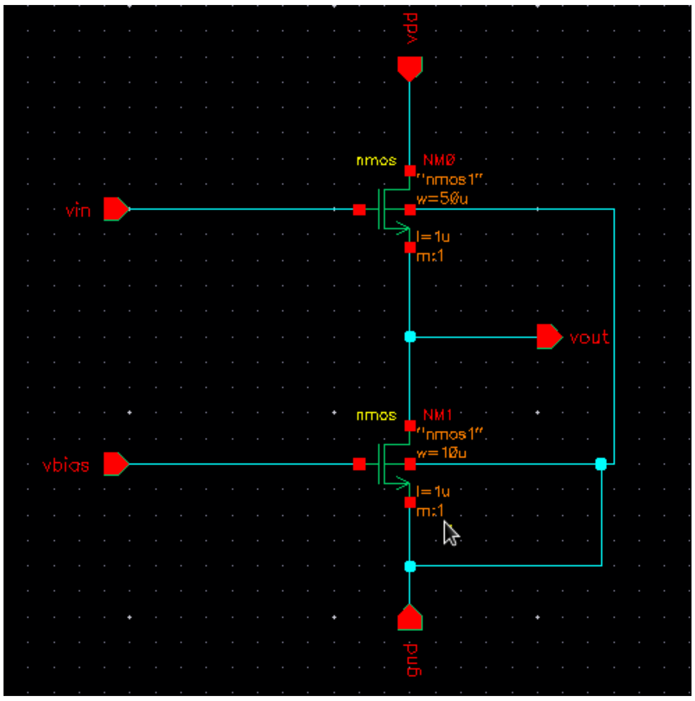
For AC analysis of amplifier, select Sweep Variable as Frequency and Sweep range, start=150 and stop=100M and Sweep type as logarithmic that is 20db.



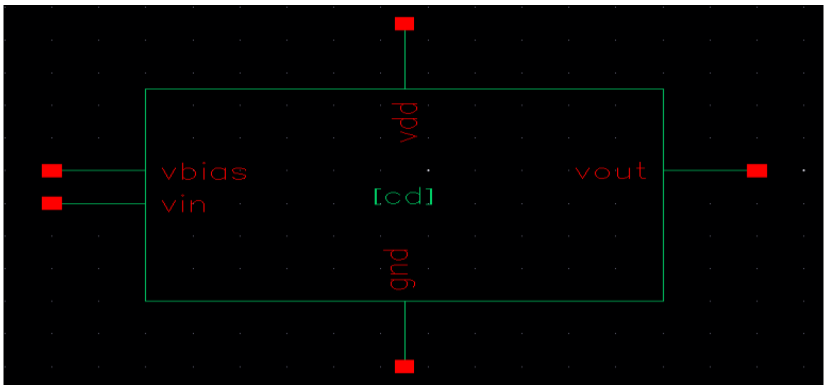
COMMON DRAIN AMPLIFIER

Library name	Cell Name	Properties/Comments
gpdk180	nmos	Model Name = nmos1; W= 50u ; L= 1u
gpdk180	nmos	Model Name = nmos1; W= 10u ; L= 1u

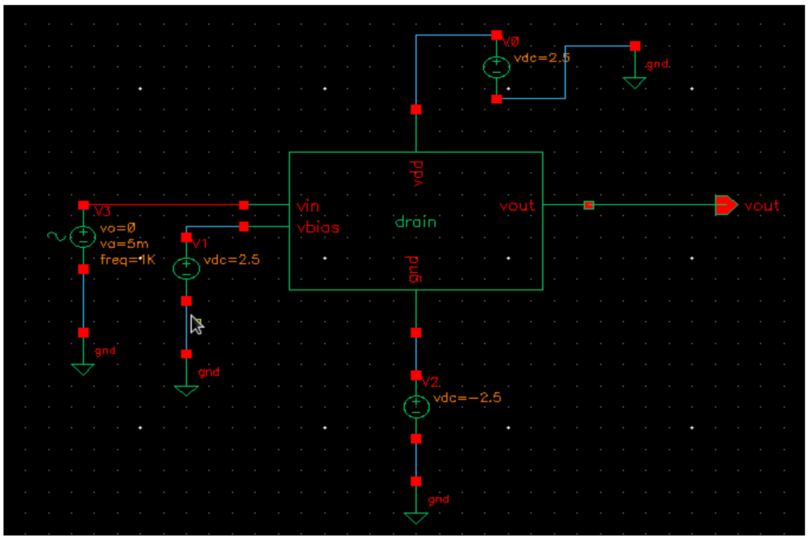
COMMON DRAIN SCHEMATIC



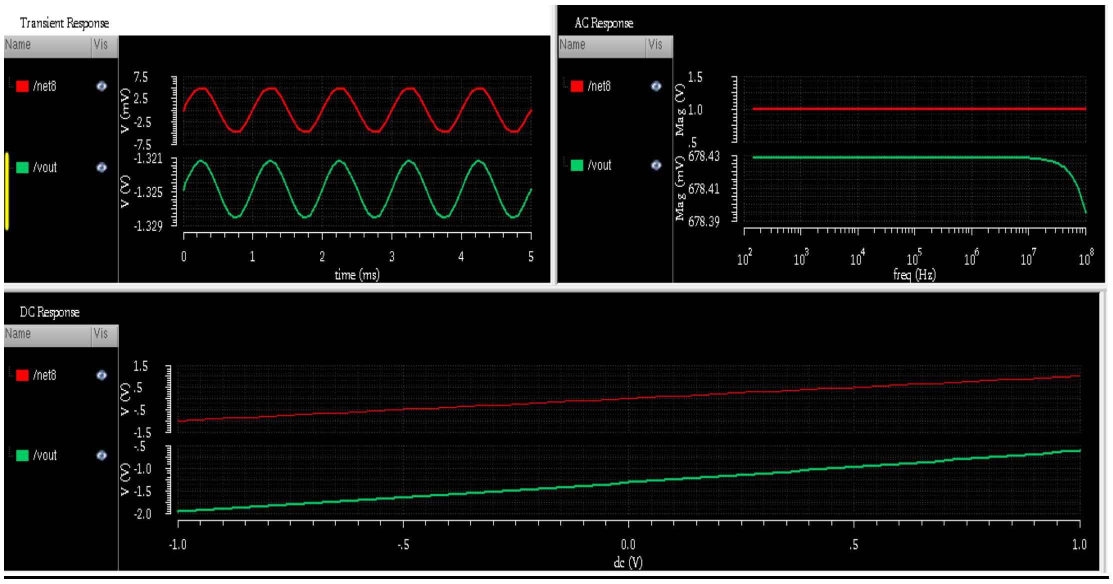
COMMON DRAIN SYMBOL



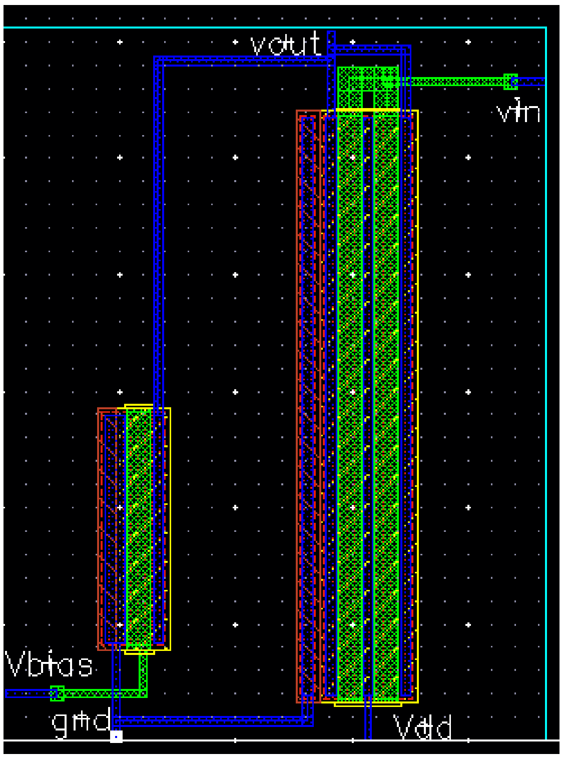
COMMON DRAIN TEST SYMBOL



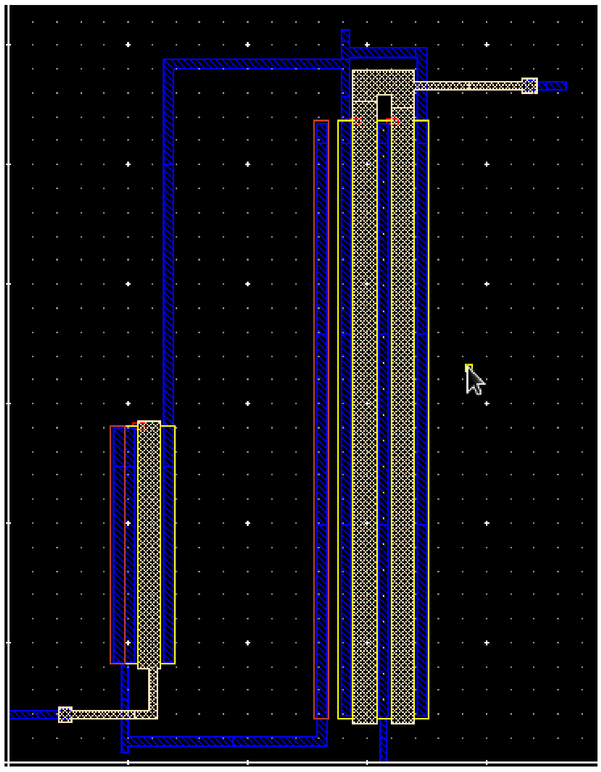
COMMON DRAIN WAVEFORM



COMMON DRAIN LAYOUT



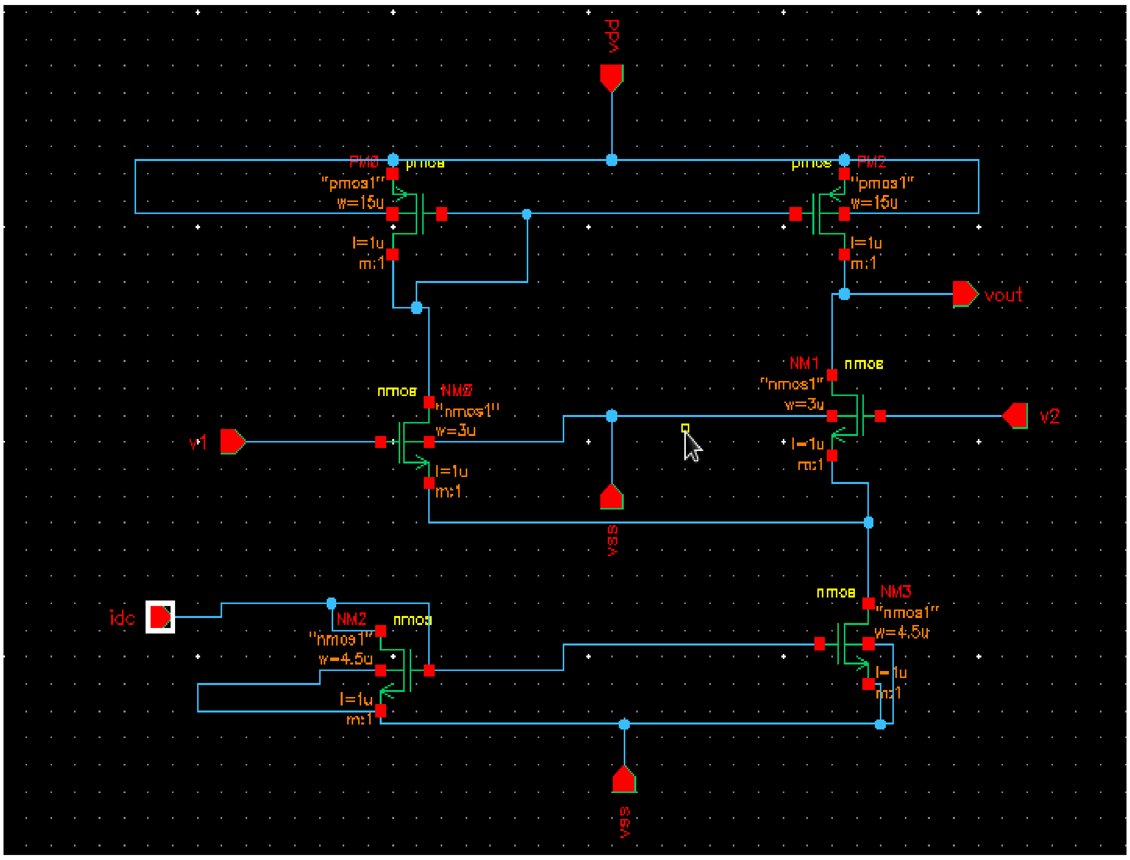
COMMON DRAIN AV EXTRACTED



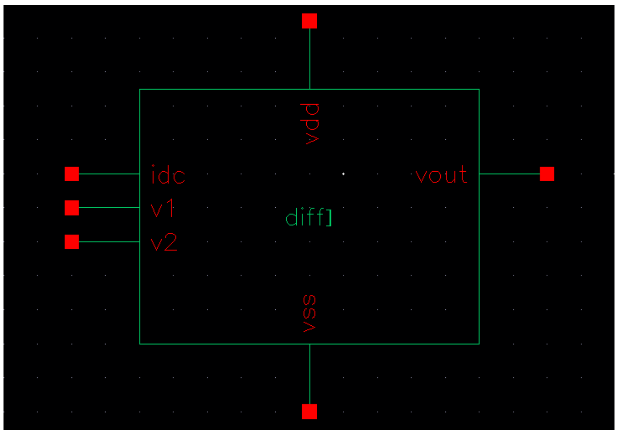
ii) A SINGLE STAGE DIFFERENTIAL AMPLIFIER

Library name	Cell Name	Properties/Comments
gpdk180	nmos	Model Name = nmos1 (NM0, NM1) ; W= 3u ; L= 1u
gpdk180	nmos	Model Name =nmos1 (NM2, NM3) ; W= 4.5u ; L= 1u
gpdk180	pmos	Model Name =pmos1 (PM0, PM1); W= 15u ; L= 1u

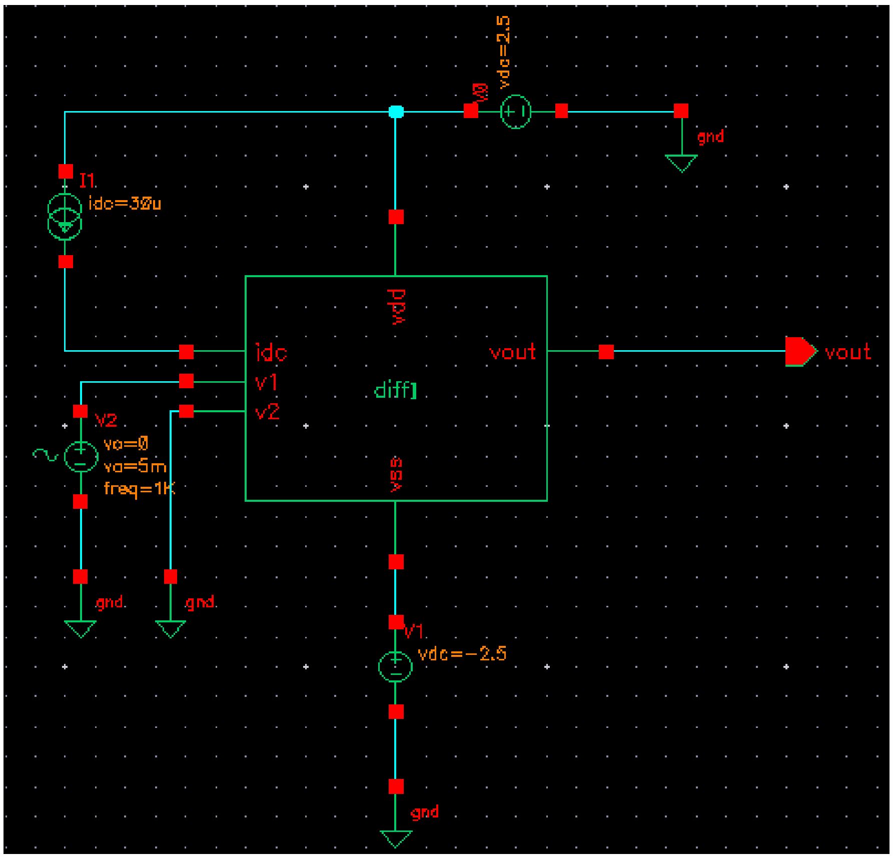
DIFFERENTIAL SCHEMATIC



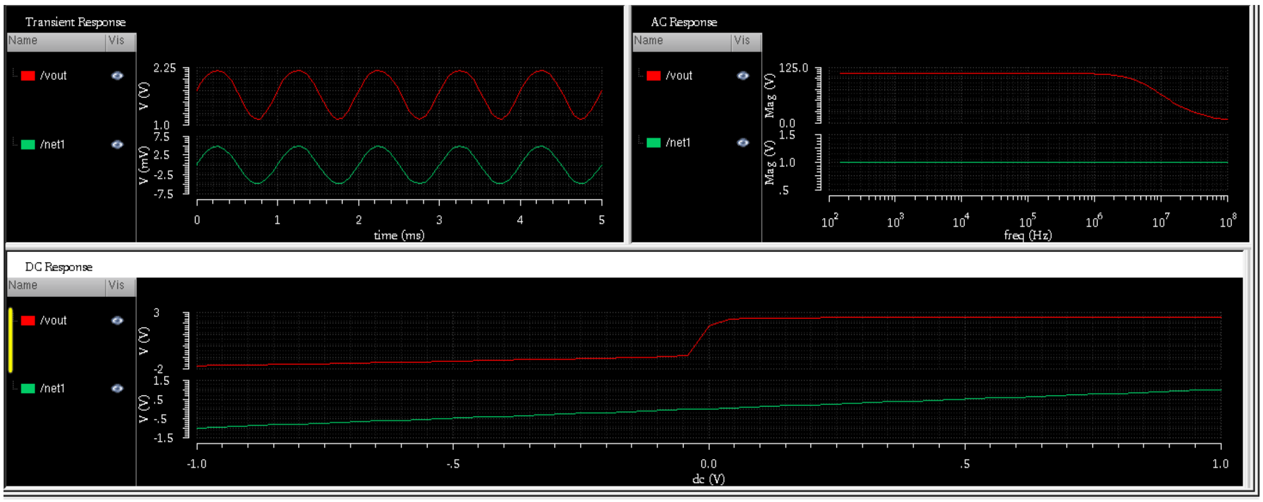
DIFFERENTIAL SYMBOL



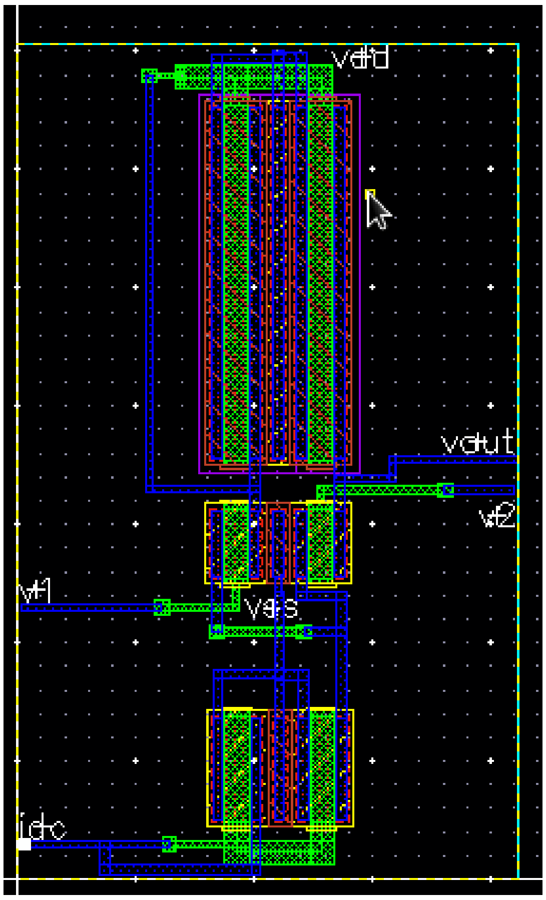
DIFFERENTIAL TEST SYMBOL



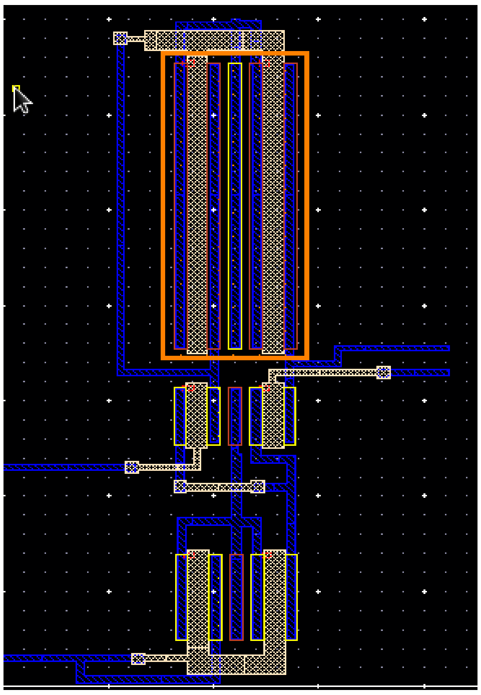
DIFFERENTIAL WAVEFORM



DIFFERENTIAL LAYOUT



DIFFERENTIAL AV EXTRACTED



2.8 Discussions:

2.9 Pre – Experimentation Questions:

1. What are LVS and DRL tool?
2. What is amplifier.
3. Why W/L ratio is important.
4. What is DC analysis of the circuit.
5. What is AC analysis.

2.10 Post – Experimentation Questions:

1. Explain the functioning of Common Source.
2. Explain the functioning of Common Drain amplifier.
3. Explain the functioning of differential amplifier.
4. Discuss the difference between CS amplifier, CD amplifier and Differential amplifier.
5. What is the necessity of DRC, ERC, QRC.

EXPERIMENT No. 03: Design an op-amp with given specification** using given differential amplifier Common source and Common Drain amplifier in library*** and completing the design flow mentioned below:

a. Draw the schematic and verify the following

i) DC Analysis ii) AC Analysis iii) Transient Analysis

b. Draw the Layout and verify the DRC, ERC

c. Check for LVS

d. Extract RC and back annotate the same and verify the Design.

3.1 Objectives:

Design an op-amp with given specifications and complete design till layout and verify the parameters DRC, ERC, QRC. Extract RC and back annotate for the verification.

3.2 Apparatus Required:

Cadence Tools.

3.3 Pre-Requisite:

Knowledge of Differential amplifier and common source amplifier.
Creating the layout in cadence.

3.4 Introduction:

Op-amps are linear devices which has nearly all the properties required for not only ideal DC amplification but is used extensively for signal conditioning, filtering and for performing mathematical operations such as addition, subtraction, integration, differentiation etc . Generally an Operational Amplifier is a 3-terminal device.

It consists mainly of an Inverting input denoted by a negative sign, ("-") and the other a Non-inverting input denoted by a positive sign ("+") in the symbol for op-amp. Both these inputs are very high impedance. The output signal of an Operational Amplifier is the magnified difference between the two input signals or in other words the amplified differential input. Generally the input stage of an Operational Amplifier is often a differential amplifier.

An operational amplifier is a DC-coupled differential input voltage amplifier with an rather high gain. In most general purpose op-amps there is a single ended output. Usually an op-amp produces an output voltage a million times larger than the voltage difference across its two input terminals.

3.5 Procedure:

STEPS FOR THE ANALOG LAB

- Login to **redhat** operating system with the respective login name and password provided in the lab.
- Right click on the screen and select **Open Terminal** to open the command prompt.
- To enter into csh shell type **csh** in the command line.
- To source the cadence tools type **source /cshrc/cad** in the command line.
- To open the tool type **virtuoso &** in the command line.

Creating the Library

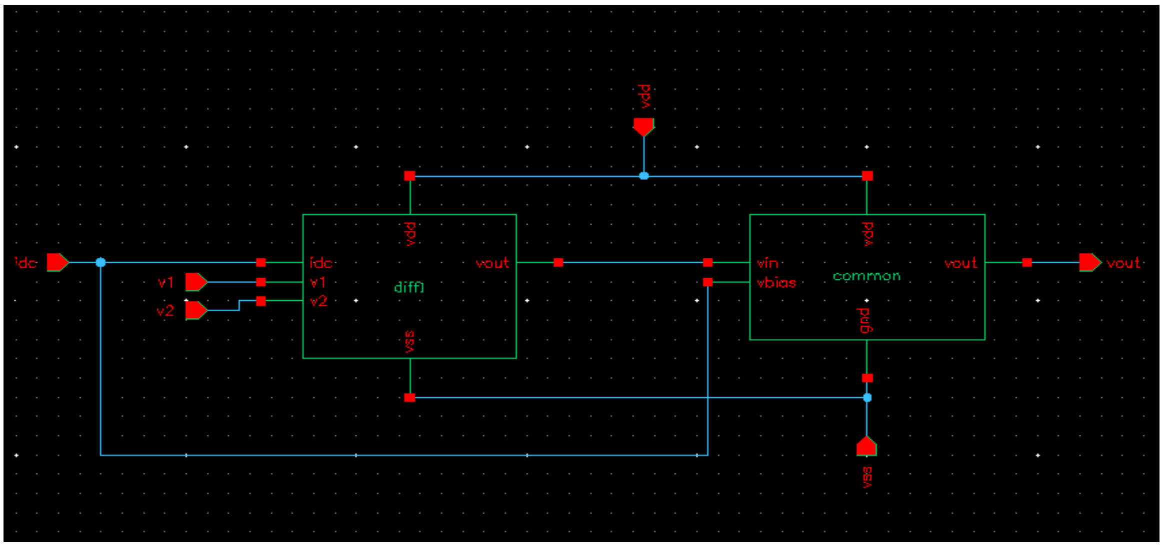
- Creating a new library by going to the link **File -> New -> Library**
- Give the library name and click ok.
- Select **attach to an existing technology library** and click **ok.**
- Select **gpdk180** and click **ok.**

SCHEMATIC AND SYMBOL CREATION

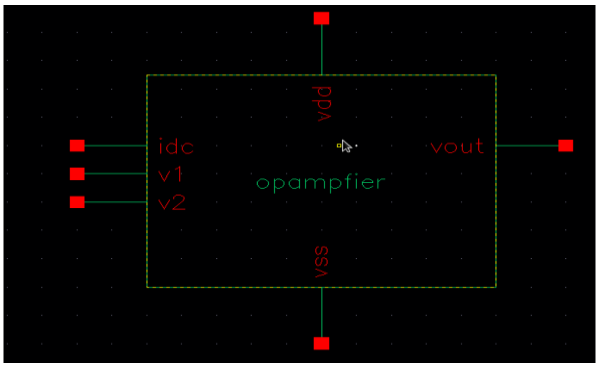
- In the Library manager select given library
- In Library manager select **File -> New -> Cell view.**
- Give the cell name as opamp
- Virtuoso Schematic Editor will open.
- Select Create -> Instance, and select the required components (pmos & nmos symbols)
- Select **Create -> pins**, and place the ports (vin, vdd, gnd, vout) on the editor.
- Make the complete connection by using wires by selecting **create -> wire (narrow)**
- After complete of the schematic, save the file by **file -> check & save**

3.6 Observations:

OPAMP SCHEMATIC



OPAMP SYMBOL



- To create symbol **Create -> Cell View -> From Cell view.**
- Virtuoso symbol editor opens.
- Place the right, left, bottom and top pins.
- Create shape of the cell using option in **Create -> Shapes.**

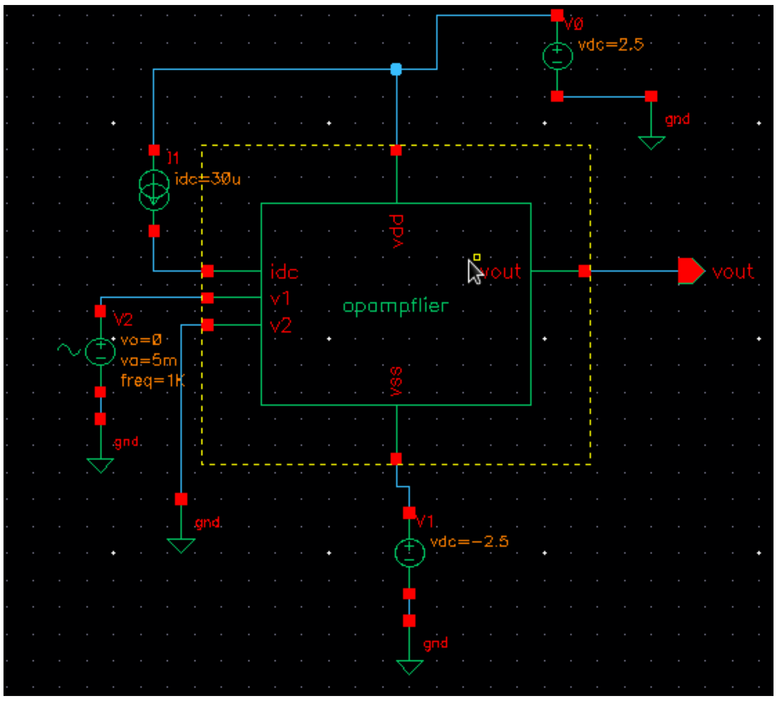
BUILDING THE TEST DESIGN FOR THE CELL

- In the library manager select given library
- Select **File -> New -> Cell view.**
- Give the cell name as **opamp_test.**
- Virtuoso Schematic editor window will open.

- Select **create -> instance,** and select the design to test.
- To apply inputs, **create -> instance.** In the library select **analoglib** and select appropriate inputs (vpulse, vdc, gnd).

Library name	Cell Name	Properties/Comments
myDesignLib	Diff_amplifier	Symbol
myDesignLib	cs_amplifier	Symbol

OPAMP TEST SYMBOL



STEPS TO SIMULATE

- In the Virtuoso Schematic Editor after completing the test circuit design **launch->ADEL**
- Virtuoso Analog Design Environment window opens.
- Go to **Analyses -> choose,** give appropriate inputs for **tran** for transient analyses and **dc** for dc analyses.
- Go to **outputs -> to be plotted -> select on schematic** (input and output wires).
- Go to **simulation -> netlist and run**

- When simulation finishes, the Transient, DC plots automatically will be popped up along with log file.
- To save the waveform, go to **session** in the ADEL window and in session select **save state**, under it click on **cell view** and then okay.

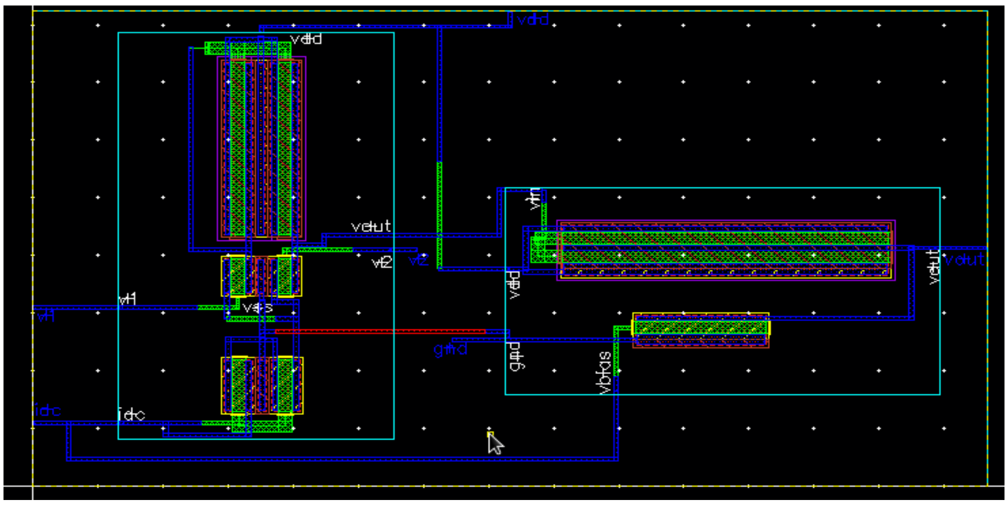
STEPS FOR CREATING LAYOUT VIEW

- From the **Inverter** schematic window menu execute **Launch – Layout XL**. A **Startup Option** form appears.
- Select **Create New** option. This gives a New Cell View Form
- Check the Cellname **(Inverter)**, Viewname **(layout).**
- Click **OK** from the New Cellview form.
- LSW and a blank layout window appear along with schematic window.

Adding Components to Layout

- Execute **Connectivity – Generate – All from Source** or click the icon in the layout editor window, **Generate Layout** form appears. Click **OK** which imports the schematic components in to the Layout window automatically.
- Re arrange the components with in PR-Boundary as shown in the next page.
- Press Shift –f in the layout window to display all the levels.
- To rotate a component, Select the component and execute **Edit –Properties**. Now select the degree of rotation from the property edit form.

- To Move a component, Select the component and execute **Edit -Move** command.



PHYSICAL VERIFICATION

Select **Assura** ⟶ **technology** ⟶ **Browse** ⟶ click on previous directory**(..)** until we get a **cad** option ⟶ double click on cad ⟶**foundary** ⟶ **analog** ⟶ **180nm** ⟶ **assura.tech.lib** ⟶ **okay.**

Assura DRC

Running a DRC

- Open the Inverter layout form the CIW or library manger if you have closed that.
- Select **Assura - Run DRC** from layout window.
- The DRC form appears. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as **gpdk180.** This automatically loads the rule file.

Your DRC form should appear like this

- Click **OK** to start DRC.
- A Progress form will appears. You can click on the watch log file to see the log file.
- When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click **Yes** to view the results of this run.
- If there any DRC error exists in the design **View Layer Window** (VLW) and **Error Layer Window** (ELW) appears. Also the errors highlight in the design itself.
- Click **View – Summary** in the ELW to find the details of errors.
- You can refer to rule file also for more information, correct all the DRC errors and **Re – run** the DRC.
- If there are no errors in the layout then a dialog box appears with **No DRC errors found** written in it, click on **close** to terminate the DRC run.

ASSURA LVS

In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS

- Select **Assura – Run LVS** from the layout window.

The Assura Run LVS form appears. It will automatically load both the schematic and layout view of the cell.

- Change the following in the form and click OK.

- The LVS begins and a Progress form appears.
- If the schematic and layout matches completely, you will get the form displaying **Schematic and Layout Match**.
- If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.
- Click **Yes** in the form.

LVS debug form appears, and you are directed into LVS debug environment.

- In the **LVS debug form** you can find the details of mismatches and you need to correct all those mismatches and **Re – run** the LVS till you will be able to match the schematic with layout.

ASSURA RCX

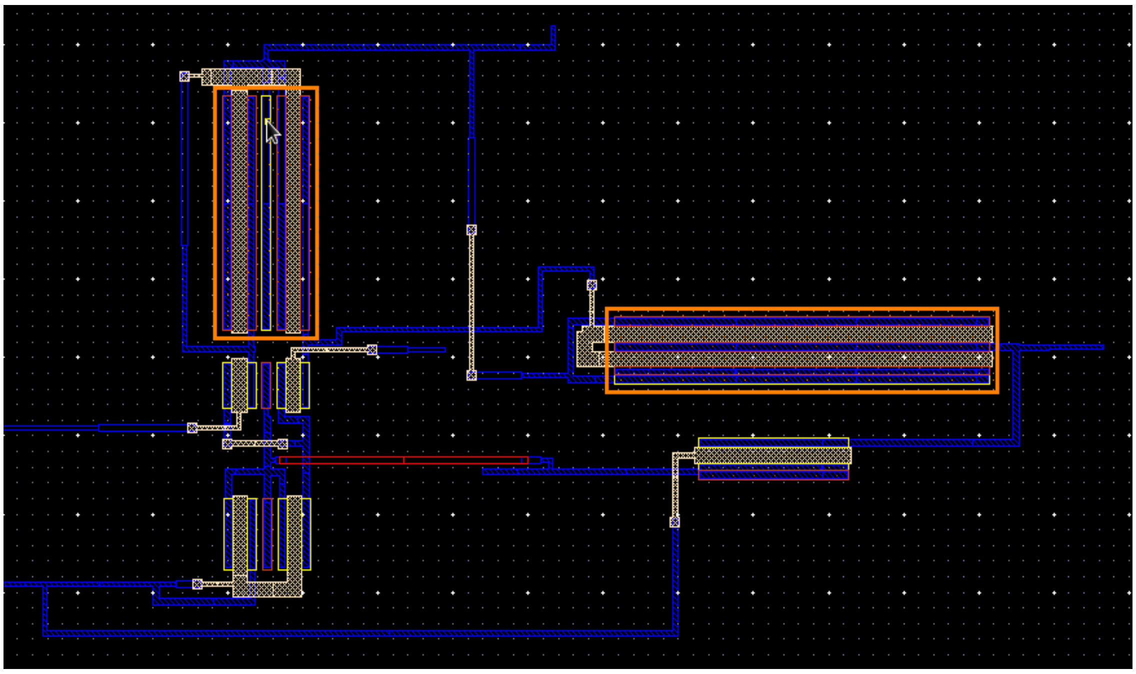
In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.

Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

Running QRC

- From the layout window execute **Assura – Run QRC**
- In QRC window,under **set up** option select **output as extracted view**,
- Under **Extraction** option,keep **extracted type as RC** and **ref node as gnd! Or vss!**

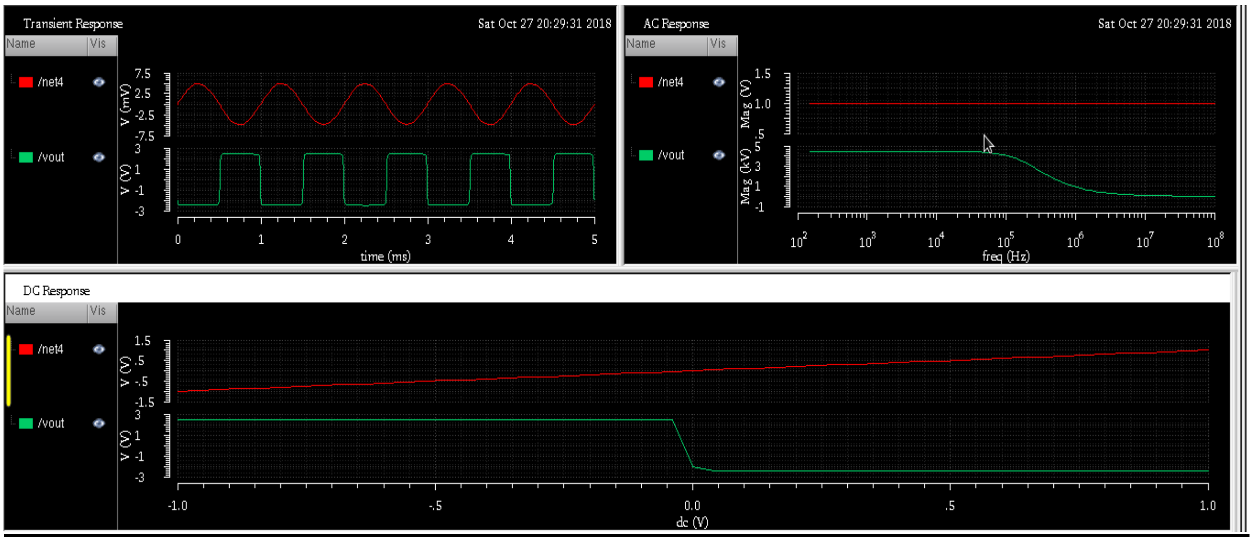
OPAMP AV EXTRACTED



Steps to obtain Back Annotate Wave

- Select the given library name, and select the schematic test.
- Go to library manager, click on **New** →**cell view** →keep type as **config**
- Below window will appear, **select view as schematic** and **select use template**, under it **select spectre** → **okay**
- In Hierarchy editor window click on **Tree View**, right click on the folder, click on **av_extracted.**
- Click on Open→(schematic test window will be opened)→launch ADEL
- To load the simulated graph ,In ADEL Click on session →load state →cell view →okay
- Once the schematic waveform is loaded, Click on the run symbol in ADEL window to obtain back annotate wave.

3.7 Results :



3.8 Discussions:

3.9 Pre – Experimentation Questions:

1. Define op-amp
2. Discuss the application of op-amp.
3. Explain different parameters of op-amp.
4. Explain the working of differential amplifier.
5. Explain the working of common source amplifier.

3.10 Post – Experimentation Questions:

1. Discuss the application of op-amp.
2. Explain the characteristics of the op-amp.
3. How inverting and non-inverting op-amp can be designed.
4. Define CMRR for an op-amp
5. Discuss the waveforms generated.

EXPERIMENT No. 04: Design a 4 bit R-2R based DAC for the given specification and completing the design flow mentioned using given op-amp in the library***.

a. Draw the schematic and verify the following

i) DC Analysis

ii) AC Analysis

iii) Transient Analysis

b. Draw the Layout and verify the DRC, ERC

4.1 Objectives:

Design a 4 bit R-2R DAC with given specifications and complete design till layout and verify the parameters DRC,ERC.

4.2 Apparatus Required:

Cadence Tools.

4.3 Pre-Requisite:

Knowledge of Differential amplifier, common source amplifier and op-amp. Creating the layout in cadence.

4.4 Introduction

R-2R Ladder, which uses fewer unique resistor values thus does not require precision resistors. A disadvantage of the former DAC design was its requirement of several different precise input resistor values: one unique value per binary input bit. Its advantage comparing to the binary weighted is it only has two values of resistors, thus the actual values used is relatively less important if it is extremely large values. The staircase voltage result is more likely to be monotonic as the effect of the MSB resistor is not many times greater than that for LSB resistor.

4.5Procedure:

STEPS FOR THE ANALOG LAB

- Login to **redhat** operating system with the respective login name and password provided in the lab.
- Right click on the screen and select **Open Terminal** to open the command prompt.
- To enter into csh shell type **csh** in the command line.
- To source the cadence tools type **source /cshrc/cad** in the command line.
- To open the tool type **virtuoso &** in the command line.

Creating the Library

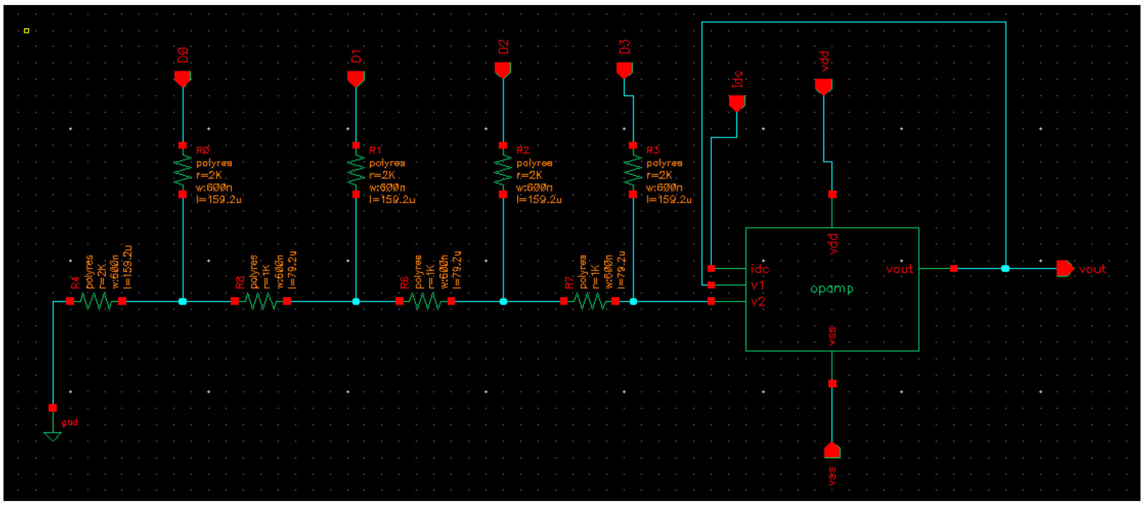
- Creating a new library by going to the link **File -> New -> Library**
- Give the library name and click ok.
- Select **attach to an existing technology library** and click **ok.**
- Select **gpdk180** and click **ok.**

SCHEMATIC AND SYMBOL CREATION

- In the Library manager select given library
- In Library manager select **File -> New -> Cell view.**
- Give the cell name as R-2R.
- Virtuoso Schematic Editor will open.
- Select Create -> Instance, and select the required components (pmos & nmos symbols)
- Select **Create -> pins**, and place the ports (vin, vdd, gnd, vout) on the editor.
- Make the complete connection by using wires by selecting **create -> wire (narrow)**
- After complete of the schematic, save the file by **file -> check & save**

4.6 Observations:

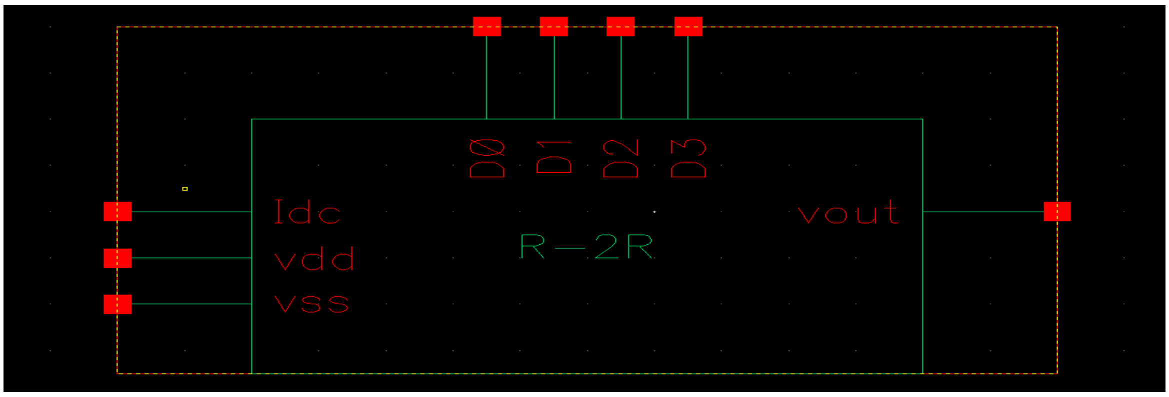
R-2R DAC SCHEMATIC


D0
D1
D2
D3
idc
vdd
R0 polyres r=2K w:600n l=159.2u
R1 polyres r=2K w:600n l=159.2u
R2 polyres r=2K w:600n l=159.2u
R3 polyres r=2K w:600n l=159.2u
R4 polyres r=2K w:600n l=159.2u
R5 polyres r=1K w:600n l=79.2u
R6 polyres r=1K w:600n l=79.2u
R7 polyres r=1K w:600n l=79.2u
idc
v1
v2
opamp
vout
vss
gnd

Library name	Cell Name	Properties/Comments
gpdk180	Polyres	R = 2k
gpdk180	Polyres	R = 1k
MyDesignLib	op-amp	Symbol
AnalogLib	Idc, gnd	idc = 30u

Pin Names	Direction
D0 D1 D2 D3	Input
Vout	Output
vdd, vss	Input

R-2R DAC SYMBOL



- To create symbol **Create -> Cell View -> From Cell view.**
- Virtuoso symbol editor opens.
- Place the right, left, bottom and top pins.

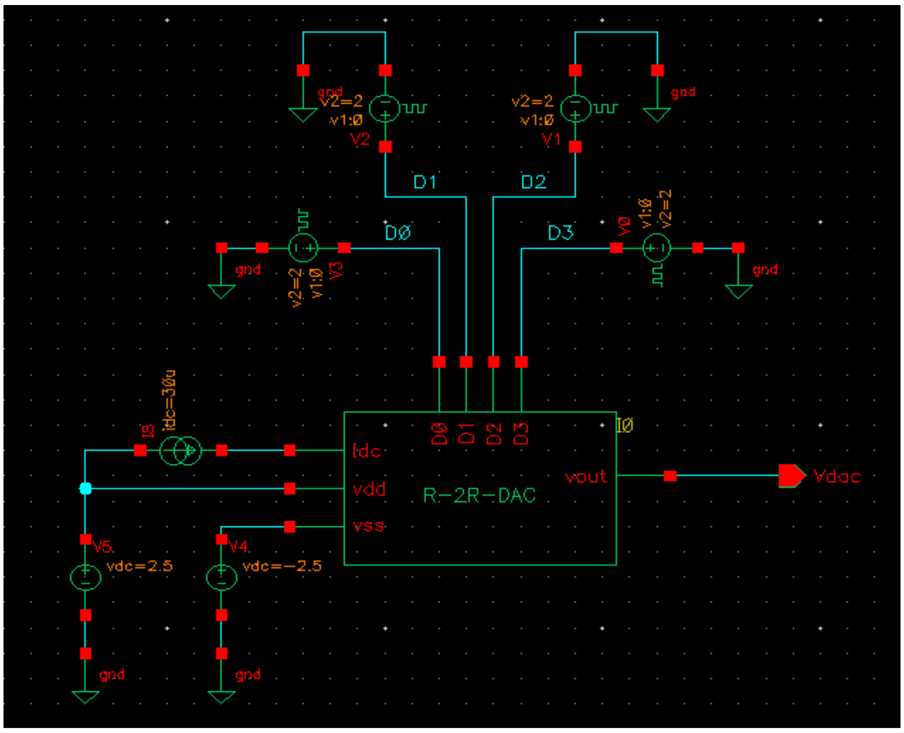
Create shape of the cell using option in **Create -> Shapes**

BUILDING THE TEST DESIGN FOR THE CELL

- In the library manager select given library
- Select **File -> New -> Cell view.**
- Give the cell name as **R-2R_test.**
- Virtuoso Schematic editor window will open.
- Select **create -> instance,** and select the design to test.
- To apply inputs, **create -> instance.** In the library select **analoglib** and select appropriate inputs (vpulse, vdc, gnd).
-

Library name	Cellview name	Properties/Comments
myDesignLib	R-2R-DAC	Symbol
analogLib	Vpulse	For V0: v1= 0 v2 = 2 Ton = 5n T = 10n For V1: v1= 0 v2 = 2 Ton = 10n T = 20n For V2: v1= 0 v2 = 2 Ton = 20n T = 40n For V3: v1= 0 v2 = 2 Ton = 40n T = 80n
analogLib	vdc, gnd	vdd = 2.5 vss = -2.5
analogLib	Idc	Idc=30ua

R-2R TEST SYMBOL



STEPS TO SIMULATE

- In the Virtuoso Schematic Editor after completing the test circuit design **launch->ADEL**
- Virtuoso Analog Design Environment window opens.
- Go to **Analyses -> choose,** give appropriate inputs for **tran** for transient analyses and **dc** for dc analyses.
- Go to **outputs -> to be plotted -> select on schematic** (input and output wires).
- Go to **simulation -> netlist and run**
- When simulation finishes, the Transient, DC plots automatically will be popped up along with log file.
- To save the waveform, go to **session** in the ADEL window and in session select **save state**, under it click on **cell view** and then okay.

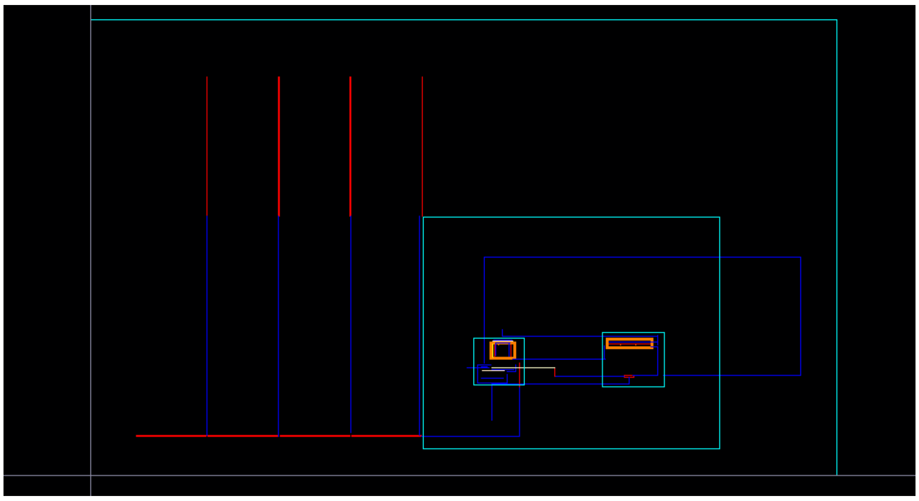
STEPS FOR CREATING LAYOUT VIEW

- From the **Inverter** schematic window menu execute **Launch – Layout XL**. A **Startup Option** form appears.
- Select **Create New** option. This gives a New Cell View Form
- Check the Cellname **(Inverter)**, Viewname **(layout).**

- Click **OK** from the New Cellview form.
- LSW and a blank layout window appear along with schematic window.

Adding Components to Layout

- Execute **Connectivity – Generate – All from Source** or click the icon in the layout editor window, **Generate Layout** form appears. Click **OK** which imports the schematic components in to the Layout window automatically.
- Re arrange the components with in PR-Boundary as shown in the next page.
- Press Shift –f in the layout window to display all the levels.
- To rotate a component, Select the component and execute **Edit –Properties**. Now select the degree of rotation from the property edit form.



PHYSICAL VERIFICATION

Select **Assura** ⟶ **technology** ⟶ **Browse** ⟶ click on previous directory**(..)** until we get a **cad** option ⟶ double click on cad ⟶ **foundary** ⟶ **analog** ⟶ **180nm** ⟶ **assura.tech.lib** ⟶ **okay.**

Assura DRC

Running a DRC

- Open the Inverter layout form the CIW or library manger if you have closed that.
- Select **Assura - Run DRC** from layout window.
- The DRC form appears. The Library and Cellname are taken from the current design window, but rule file may be missing. Select the Technology as **gpdk180.** This automatically loads the rule file.

Your DRC form should appear like this

- Click **OK** to start DRC.

- A Progress form will appears. You can click on the watch log file to see the log file.
- When DRC finishes, a dialog box appears asking you if you want to view your DRC results, and then click **Yes** to view the results of this run.
- If there any DRC error exists in the design **View Layer Window** (VLW) and **Error Layer Window** (ELW) appears. Also the errors highlight in the design itself.
- Click **View – Summary** in the ELW to find the details of errors.
- You can refer to rule file also for more information, correct all the DRC errors and **Re – run** the DRC.
- If there are no errors in the layout then a dialog box appears with **No DRC errors found** written in it, click on **close** to terminate the DRC run.

ASSURA LVS

In this section we will perform the LVS check that will compare the schematic netlist and the layout netlist.

Running LVS

- Select **Assura – Run LVS** from the layout window.

The Assura Run LVS form appears. It will automatically load both the schematic and layout view of the cell.

- Change the following in the form and click OK.
- The LVS begins and a Progress form appears.
- If the schematic and layout matches completely, you will get the form displaying **Schematic and Layout Match**.
- If the schematic and layout do not matches, a form informs that the LVS completed successfully and asks if you want to see the results of this run.
- Click **Yes** in the form.

LVS debug form appears, and you are directed into LVS debug environment.

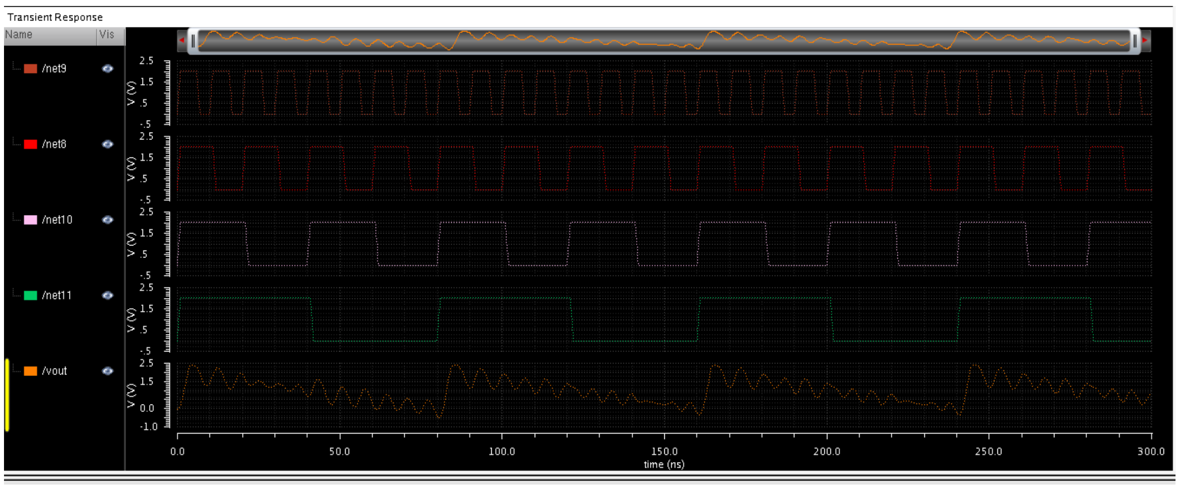
- In the **LVS debug form** you can find the details of mismatches and you need to correct all those mismatches and **Re – run** the LVS till you will be able to match the schematic with layout.

ASSURA RCX

- In this section we will extract the RC values from the layout and perform analog circuit simulation on the designs extracted with RCX.

- Before using RCX to extract parasitic devices for simulation, the layout should match with schematic completely to ensure that all parasites will be backannoted to the correct schematic nets.

4.7 Results :



4.8 Discussions

4.9 Pre-Requisite Question

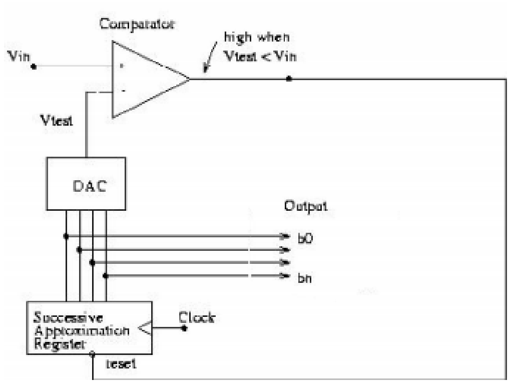
1. What are the design rules and explain them
2. Discuss the functioning of op-amp.
3. What is R-2R based DAC.
4. What is gpdk90.
5. What are the different technologies used for designing in Cadence.

4.10 Post-Requisite Question

1. Explain the working R-2R DAC.
2. Discuss the circuit design of R-2R based DAC.
3. Discuss the waveforms generated for the same.
4. Explain sampling and hold method.
5. Explain the technology gpdk180.

EXPERIMENT No. 05: For the SAR based ADC mentioned in the figure below draw the mixed signal schematic and verify the functionality by completing ASIC Design FLOW.

[Specifications to GDS-II]



5.1 Objectives:

Design an op-amp with given specifications and complete design till layout and verify the parameters DRC, ERC, QRC. Extract RC and back annotate for the verification.

5.2 Apparatus Required:

Cadence Tools.

5.3 Pre-Requisite:

Knowledge of Differential amplifier and common source amplifier, op-amp. Creating the layout in cadence.

5.4 Introduction:

A SAR ADC uses a series of comparisons to determine each bit of the converted result. Therefore, a SAR ADC needs at least n+1 clock cycles to convert an analog input to the ADC to a result, where n is the number of bits of the ADC. The analog input is tracked by the SAR ADC, then sampled and held during the conversion.

5.5 Project:

****THIS EXPERIMENT WILL BE GIVEN AS MINI-PROJECT****

5.6 Discussions:

5.7 Results

5.8 Pre-Requisite Question

1. Discuss the functioning of CS amplifier.
2. What is differential amplifier and explain the functioning.
3. Explain the functioning of op-amp.
4. Explain the characteristics of op-amp.
5. Draw the waveform generated by op-amp and analyze.

5.9 Post-Requisite Question

1. Explain sampling and hold.
2. Discuss the working of ADC
3. What is the application of SAR.
4. Explain the waveform generated by SAR.
5. What is the function of ADC.