TSMC PDK usage guide:

Introduction of the usage of IC6.1 process design kits (PDK)

Release 0.1

June 2009

DISCLAIMER

The information contained herein is provided by TSMC on an "AS IS" basis without any warranty, and TSMC has no obligation to support or otherwise maintain the information. TSMC disclaims any representation that the information does not infringe any intellectual property rights or proprietary rights of any third parties. There are no other warranties given by TSMC, whether express, implied or statutory, including, without limitation, implied warranties of merchantability and fitness for a particular purpose.

STATEMENT OF USE

This information contains confidential and proprietary information of TSMC. No part of this information may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of TSMC. This information was prepared for informational purpose and is for use by TSMC's customers only. TSMC reserves the right to make changes in the information at any time and without notice.

Table of Contents

Chapter 1 Introduction

The major purpose of this document is to introduce the basic usage of a TSMC's PDK for those users who are completely new to TSMC PDK or never use TSMC's PDKs before as a reference. To ease the overall introduction, we use a simple design as an example to go through the whole design flow: starting from the schematic capture and ending at the physical verification and post-layout simulation. Moreover, we divide the whole flow into several phases to match general logic or Mix-signal design flows and they are "Schematic capture", "Pre-layout simulation", "Layout creation", "Physical verification" and "Post-layout simulation".

- ⌘ **Schematic capture**
  - *Environment setup*
  - *Creating a library*
  - *Creating a design*
  - *Creating a symbol*
  - *Creating a test fixture*
- ⌘ **Pre-layout simulation**
  - *Using spectre for simulation*
  - *Using hspice for simulation*
- ⌘ **Layout creation**
  - *Schematic-drive-layout*
  - *Components placement*
  - *Auto-route and manual route*
- ⌘ **Physical verification**
  - *Physical verification using Assura*
    - A. Assura DFII flow
    - B. Assura CDL flow
  - *Physical verification using Calibre*
    - A. Calibre CDL flow
    - B. Calibre GUI flow

- **Post-layout simulation**
  - *Simulating with the extracted netlist*
  - *Simulating with the extracted view*

Each phase may contain various kinds of flows corresponding to different tools. Users can choose those flows according to their needs. For example, in the pre-layout simulation phase, two kinds of simulators can be chose to carry out the simulation. You only need to choose one simulator for your simulation.

> ***This document doesn't focus on the tool usage. For the detail usage of the related tools, please contact the tool vendors.***

Chapter 2 Schematic capture

After you have finished the installation of a TSMC's PDK, we will start to create a new design based on the installed PDK.

Environment setup

Before we start to create a new design, some environment setups should be done. First, we have to set the environment variable of "CDS_Netlisting_Mode" to "Analog". This can be archived by the following UNIX command:

```
setenv    CDS_Netlisting_Mode
```

Then, try to find a directory to create your project by:

```
%mkdir ~/my_project

%cd ~/my_project
```

All the further designs created by users should be performed in this directory.

The next steps are to copy some necessary files from the PDK installed directory to your project directory. These include copying the 'display' file and linking the 'models' and 'stream' directories and others to your project directory.

```
%cp <pdk_install_direcotry>/display.drf    .
%ln –s <pdk_install_directory>/models    .
%ln –s <pdk_install_directory>/stream    .
%cp -f    <pdk_install_directory>/assura_tech.lib    .    (For Assura tool)
%cp -rf    <pdk_install_directory>/Assura    .             (For Assura tool)
%cp -rf    <pdk_install_directory>/Calibre    .            (For Calibre tool)
```

The last step for environment setup is to create a "cds.lib" file in your project directory. Users can use a text editor to add the following line into the "cds.lib" file located in you project directory:

```
INCLUDE    <pdk_install_directory>/cds.li
```

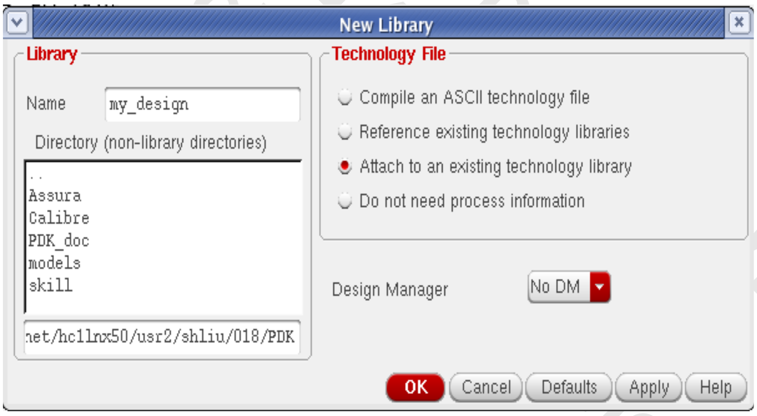
Note:

1. The installation procedures of a TSMC’s PDK can be found in the document of “TSMC PDK reference manual ” released along with the corresponding PDK.

2. The <pdk_install_directory> is referred to the path where the TSMC’s PDK was installed.

Creating a Library

After completing the environment setup, we can start to create a new library. This can be archive by using the command “File->New->Library” from either the CIW (Command Interpreter Window) or the Library Manager and select the “Attach to an existing techfile” option. When the Library Manager asked for the name of the “Attach to Technology Library”, please select the name of PDK library that you installed for this design. Here we have an example for TSMC 0.18um MM/RF PDK (fig1~2).


New Library

Library

Name: my_design

Directory (non-library directories)

..
Assura
Calibre
PDK_doc
models
skill

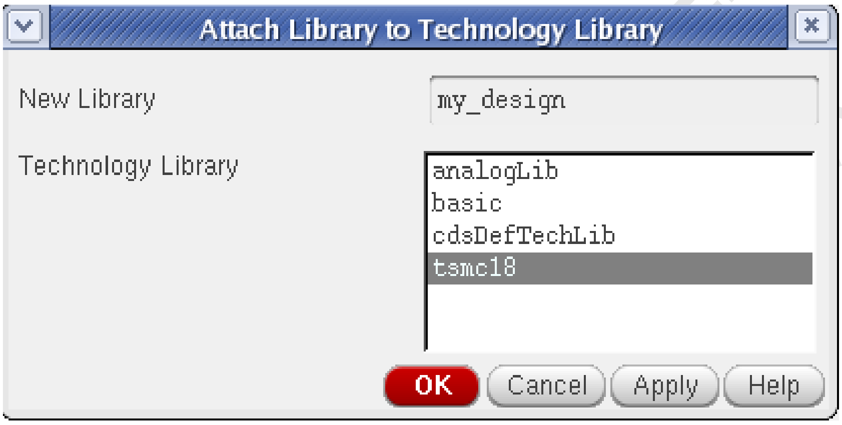
net/hc1lnx50/usr2/shliu/018/PDK

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

FIG. 1


Attach Library to Technology Library

New Library: my_design

Technology Library:

analogLib
basic
cdsDefTechLib
tsmc18

FIG. 2

Creating a design

When the library creation is completed, we can now start to create our design circuit under Cadence Composer environment. The basic steps to capture a new schematic including: open a new schematic design, select and place components, edit component properties, wire components…and so on.

From now on, we will use a two-stage invert (a buffer) as an example, which was designed based on TSMC 0.18um MM/RF PDK, to continue the following introductions in this document.

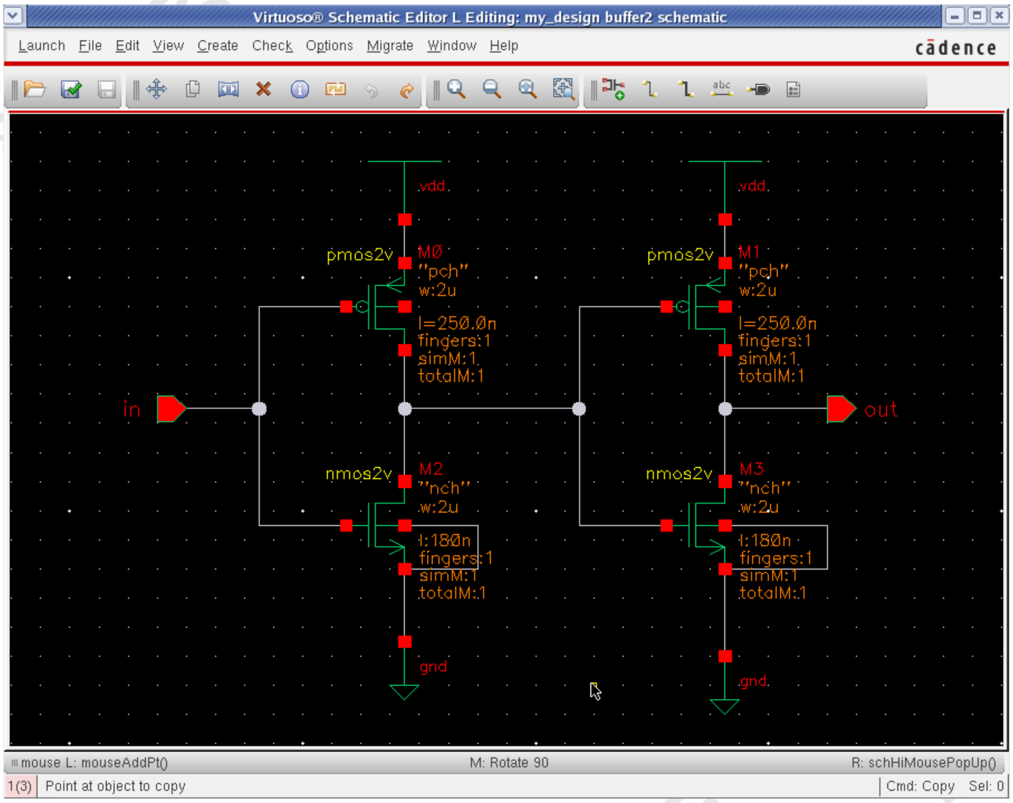


FIG. 3 schematic-capture of a buffer

Creating a symbol

After completing the creation of schematic-capture, we have to create its corresponding symbol for the subsequence simulation steps.

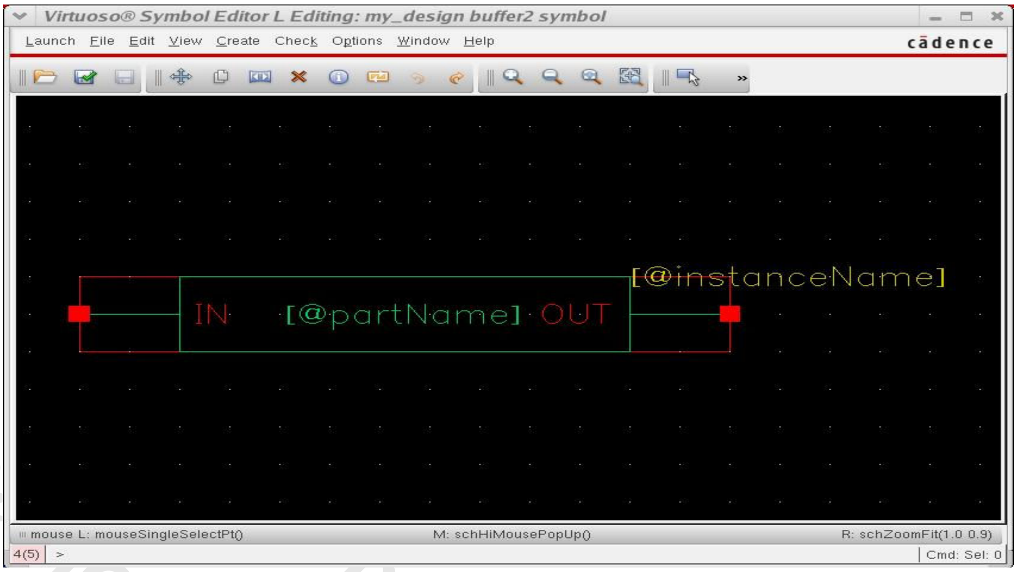

Virtuoso® Symbol Editor L Editing: my_design buffer2 symbol
Launch File Edit View Create Check Options Window Help
cādence
[@instanceName]
IN
[@partName]
OUT
mouse L: mouseSingleSelectPt()
M: schHiMousePopUp()
R: schZoomFit(1.0 0.9)
4(5)
>
Cmd: Sel: 0

FIG. 4 symbol for the buffer

Creating a test fixture

The final step before we start the simulation is to create a test fixture for our design. The creation of test fixture is similar to the creation of a design. Furthermore, you also have to prepare the voltage source and determine the output loading. Generally, a test fixture will consist of the following components: a design (the buffer in our case), DC voltage source, ground, vdd, voltage source(vdc/vpluse), and output loading. The test fixture that we used for our design is namely 'TB_buffer2' and can be seen in FIG5.

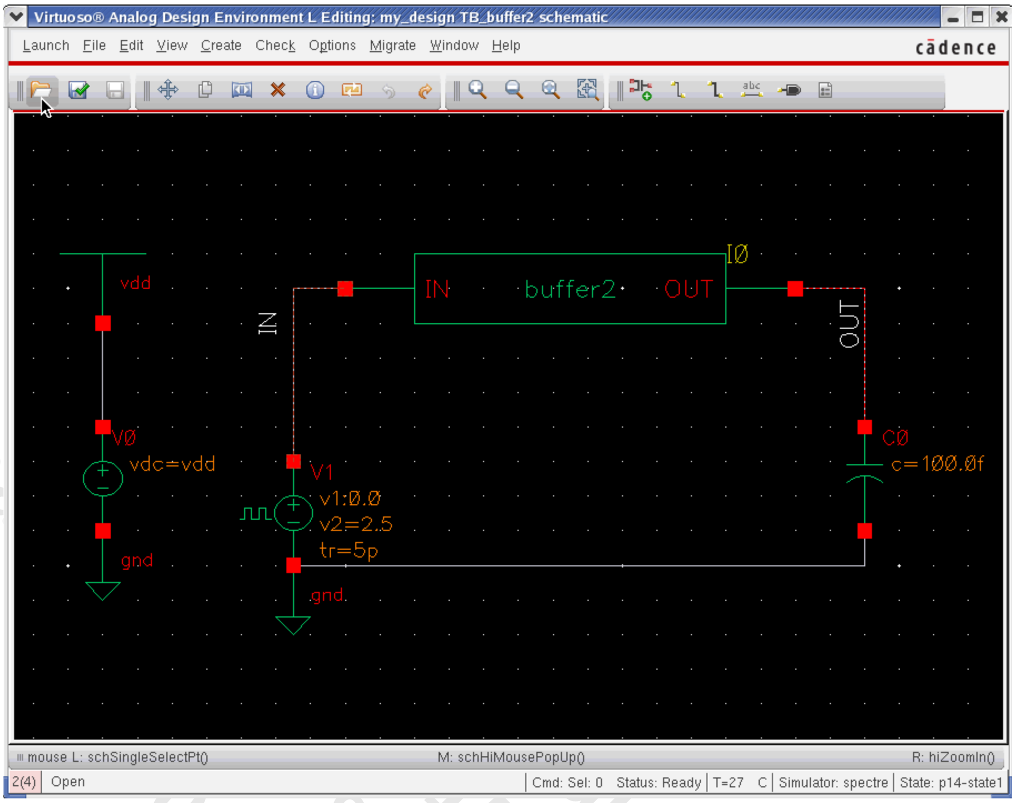


FIG. 5 test fixture schematic for the buffer

Chapter 3 Pre-layout simulation

Performing pre-simulation

After the creation of our design is completed, we have to verify the electrical performance and the functionality of our design using a simulation tool. In this chapter, we will have introductions on two simulators: Spectre and Hspice.

Using spectre for simulation

In this section, we will start to simulate our design with spectre simulator under Analog Artist environment. The simulation steps for spectre is as follows:

1. Open Analog Artist window

This can be archived by “Launch->ADE L” in the menu banner of schematic view (Fig. 6).

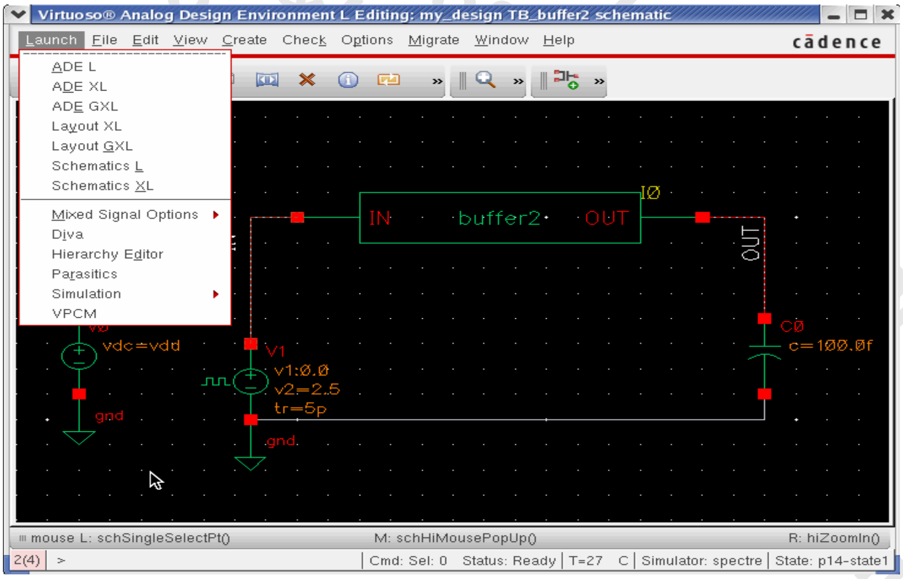
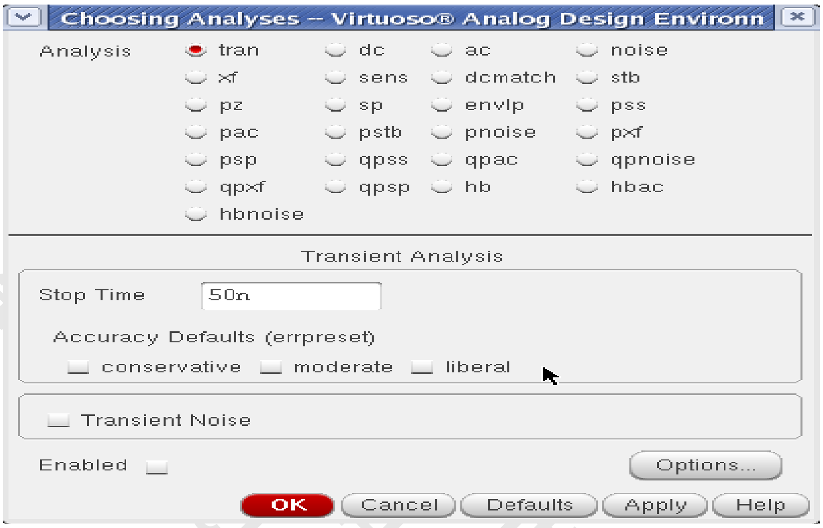


FIG. 6 open Analog Artist window

2. Select analysis type and fill in parameters for simulation

In the Analog Artist window, there are many analysis options that you can choose. Since we want to analysis the delay information (performance) of our design, we choose the transient analysis for our design (Fig 7). Some designers may want to see the OP point and they can also include the DC op point analysis.



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 50n

Accuracy Defaults (errpreset)

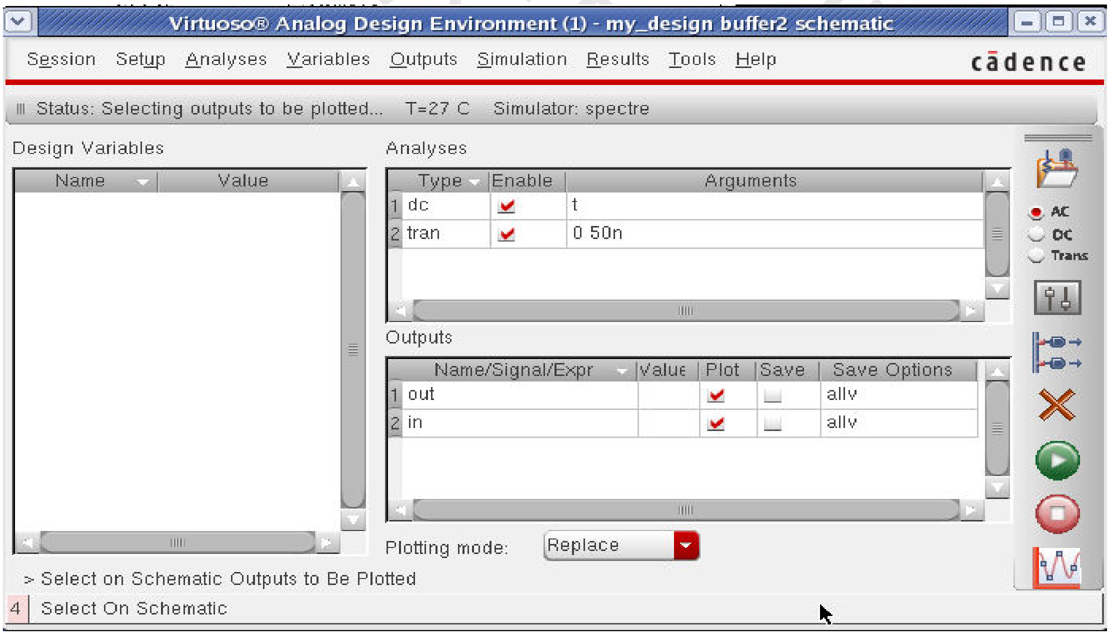
□ conservative □ moderate □ liberal

□ Transient Noise

Enabled □

FIG. 7 choosing analysis type and filling in parameters

In addition to choosing the analysis type, we also need to select the nodes that we want to observe as simulation results from the schematic view. In our test fixture, we choose the "IN" and "OUT" nodes of our buffer circuit for the result browsing.



Virtuoso® Analog Design Environment (1) - my_design buffer2 schematic

Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence

Status: Selecting outputs to be plotted... T=27 C Simulator: spectre

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	dc	✓	t
2	tran	✓	0 50n

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	out		✓	□	allv
2	in		✓	□	allv

Plotting mode: Replace

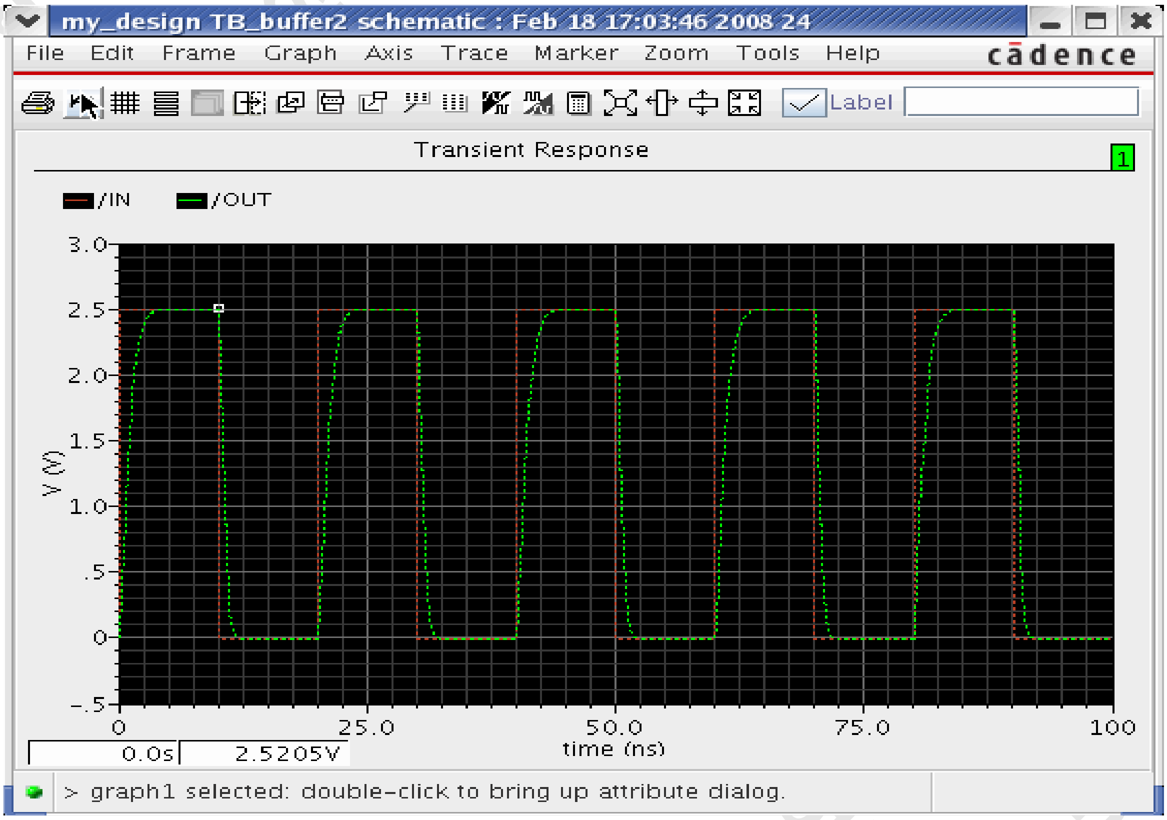
> Select on Schematic Outputs to Be Plotted

4 Select On Schematic

FIG. 8 select browsing nodes

3. Run simulation and observe the simulation results

To start the simulation, you can click “simulation->netlist and run” from the Analog Artist menu banner. After the simulation is completed, the wave form window appears. You can see the simulation result that corresponding to the chose nodes (in step 2). In our case, the waveform of nodes “IN” and “OUT” of our design circuit (buffer) is shown (Fig 9).


my_design TB_buffer2 schematic : Feb 18 17:03:46 2008 24
File Edit Frame Graph Axis Trace Marker Zoom Tools Help
cādence
Label
Transient Response
1
/IN
/OUT
3.0
2.5
2.0
1.5
1.0
.5
0
-.5
V (V)
0
25.0
50.0
75.0
100
time (ns)

0.0s 2.5205V

> graph1 selected: double-click to bring up attribute dialog.

FIG. 9 simulation waveform

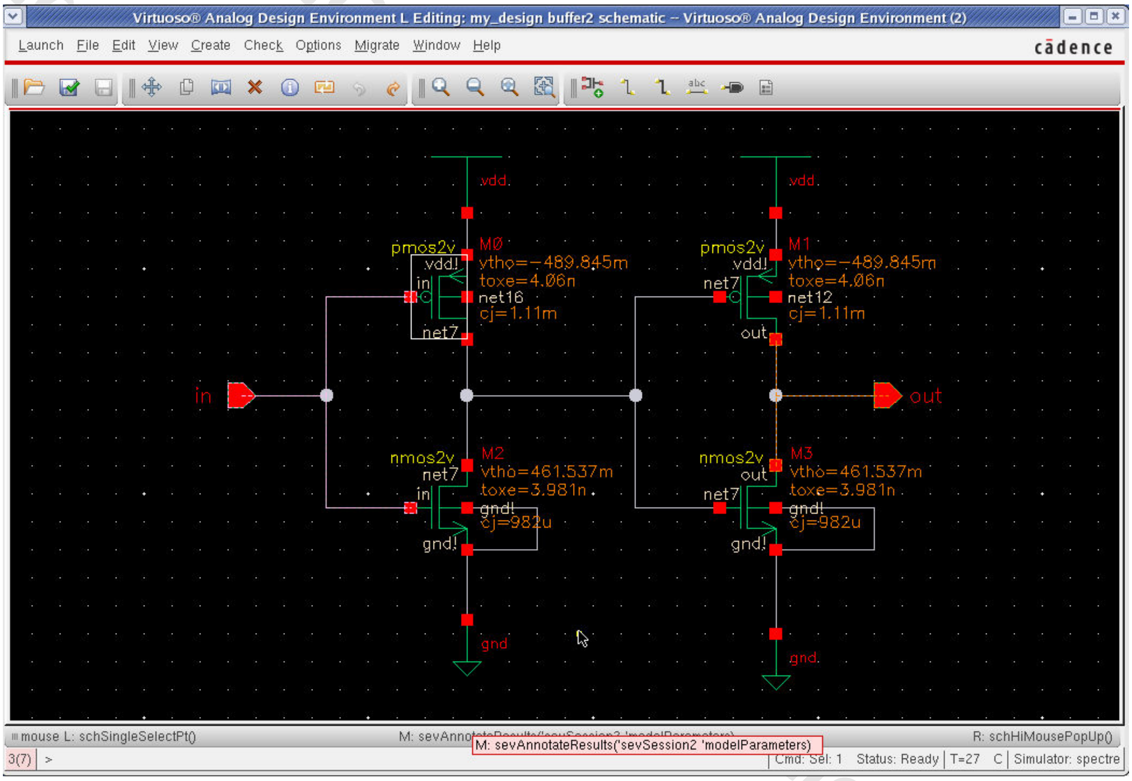
According to the waveform in Fig 9, we found that the functionality of this buffer is correct and the performance of this buffer is as below (with 1pF load):

Delay time: Td (rise) = 0.565ns; Td (fall) = 0.52ns

Rising/Falling time: Tr = 0.996ns; Tf = 0.72ns

4. Annotating simulation results back to schematic

Sometime, you may want to see more detail simulation results such as operating points, DC voltages, and model parameters on the schematic of your design. You can back annotate the simulation data to the schematic by choose “Results-> Annotate->…” from the menu banner of Artist simulation window. The available choices of annotated data are “DC node voltages”, “DC operating points”, “Transient node voltages”, ”Transient operating points”, “Model parameters”, “Component parameters”…and so on. Here we select to back annotate the “model parameters” inside our buffer circuit (Fig 10).



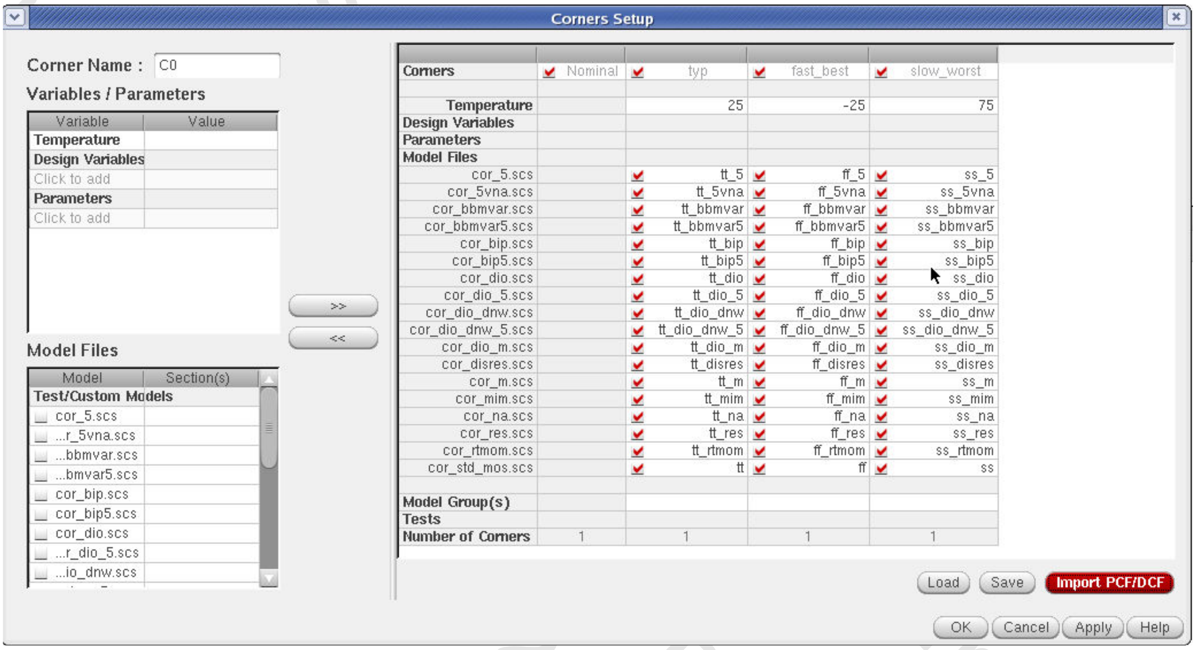
mouse L: schSingleSelectPt()　M: sevAnnotateResults('sevSession2 'modelParameters)　R: schHiMousePopUp()

3(7) >　M: sevAnnotateResults('sevSession2 'modelParameters)　Cmd: Sel: 1　Status: Ready　T=27 C　Simulator: spectre

FIG. 10 back annotate model parameter data

5. Running corner analysis to cover the process variation

In addition to cover the typical case, we may sometime want to simulate our design to cover the process variations in different corners. This can let us know whether the circuit performance specifications will still meet even when the process variation shift to different corner. Furthermore, this can also improve the product yield of our design. In our case, we simulate our design in three different corners: the typical case, the fast_best case (all devices in FF) and slow_worst (all devices in SS) case. By loading the well-defined PCF file released along with TSMC's PDK, you can find the corner analysis window as FIG 11.



Corner Name : C0

Variables / Parameters

Variable	Value
Temperature	
Design Variables	
Click to add	
Parameters	
Click to add	

Model Files

Model	Section(s)
Test/Custom Models	
□ cor_5.scs	
□ ...r_5vna.scs	
□ ...bbmvar.scs	
□ ...bmvar5.scs	
□ cor_bip.scs	
□ cor_bip5.scs	
□ cor_dio.scs	
□ ...r_dio_5.scs	
□ ...io_dnw.scs	

Corners	✓ Nominal	✓ typ	✓ fast_best	✓ slow_worst
Temperature		25	-25	75
Design Variables				
Parameters				
Model Files				
cor_5.scs		✓ tt_5	✓ ff_5	✓ ss_5
cor_5vna.scs		✓ tt_5vna	✓ ff_5vna	✓ ss_5vna
cor_bbmvar.scs		✓ tt_bbmvar	✓ ff_bbmvar	✓ ss_bbmvar
cor_bbmvar5.scs		✓ tt_bbmvar5	✓ ff_bbmvar5	✓ ss_bbmvar5
cor_bip.scs		✓ tt_bip	✓ ff_bip	✓ ss_bip
cor_bip5.scs		✓ tt_bip5	✓ ff_bip5	✓ ss_bip5
cor_dio.scs		✓ tt_dio	✓ ff_dio	✓ ss_dio
cor_dio_5.scs		✓ tt_dio_5	✓ ff_dio_5	✓ ss_dio_5
cor_dio_dnw.scs		✓ tt_dio_dnw	✓ ff_dio_dnw	✓ ss_dio_dnw
cor_dio_dnw_5.scs		✓ tt_dio_dnw_5	✓ ff_dio_dnw_5	✓ ss_dio_dnw_5
cor_dio_m.scs		✓ tt_dio_m	✓ ff_dio_m	✓ ss_dio_m
cor_disres.scs		✓ tt_disres	✓ ff_disres	✓ ss_disres
cor_m.scs		✓ tt_m	✓ ff_m	✓ ss_m
cor_mim.scs		✓ tt_mim	✓ ff_mim	✓ ss_mim
cor_na.scs		✓ tt_na	✓ ff_na	✓ ss_na
cor_res.scs		✓ tt_res	✓ ff_res	✓ ss_res
cor_rtmom.scs		✓ tt_rtmom	✓ ff_rtmom	✓ ss_rtmom
cor_std_mos.scs		✓ tt	✓ ff	✓ ss
Model Group(s)				
Tests				
Number of Corners	1	1	1	1

FIG. 11 Loading PCF file for corner analysis

After finished the corner analysis, the simulation waveforms of those inspect nodes in different corners are shown in FIG 12. From Fig 12, we find that the circuit performances of our circuit in different corners are as follows:

In slow_worst case:

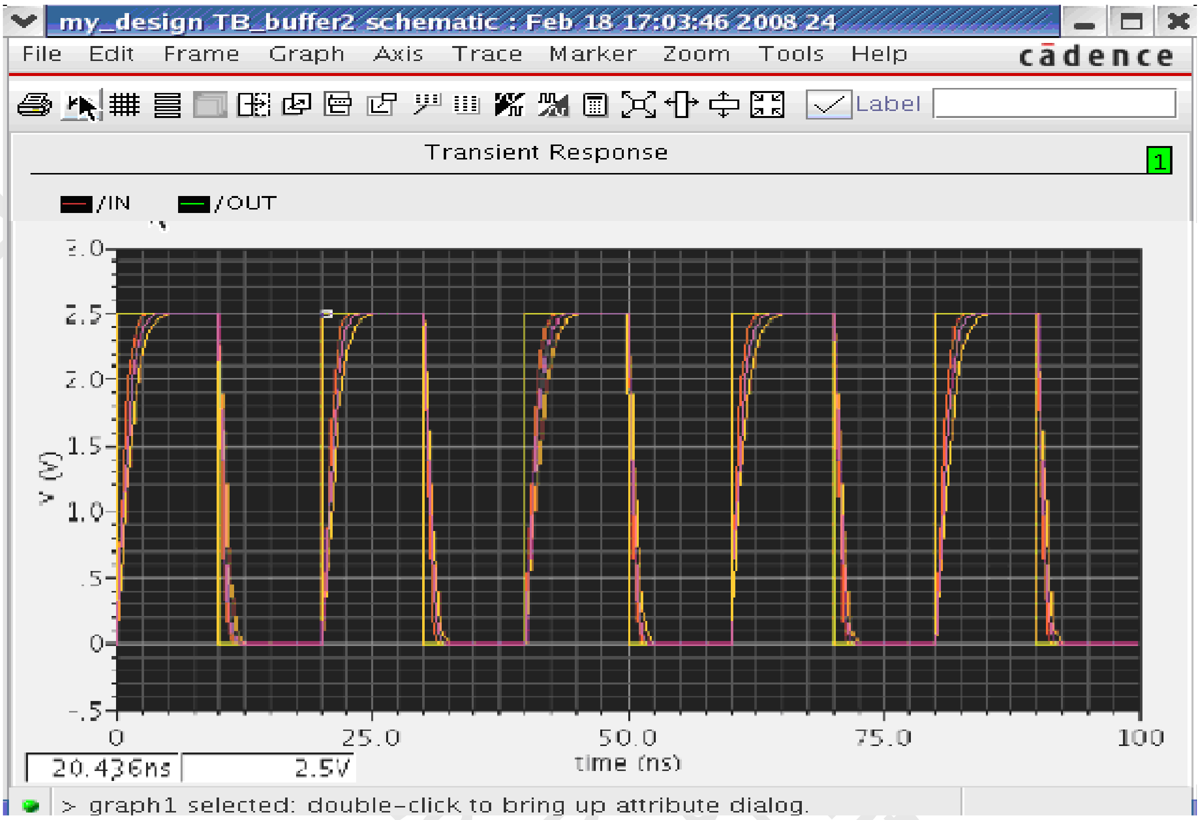
Delay time: Td (rise) = 0.709ns; Td (fall) = 0.643ns

Rising/Falling time: Tr = 1.226ns; Tf = 0.925ns

In Fast_best case:

Delay time: Td (rise) = 0.446ns; Td (fall) = 0.424ns

Rising/Falling time: Tr = 0.765ns; Tf = 0.574ns



> graph1 selected: double-click to bring up attribute dialog.

FIG. 12 simulation waveform for corner analysis

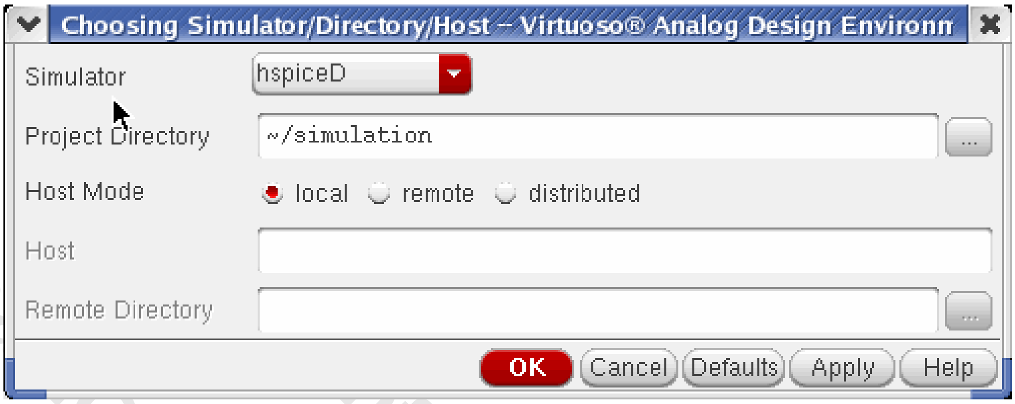
Using hspice for simulation

If you are not comfortable in using spectre for simulation, you can also choose other simulators (such as HspiceD, UltraSim, and so on) for simulation. In this section, we will introduce the simulation steps to use Hspice as the simulator.

The simulation steps for Hspice are as follows:

1. *Write out the hspice netlist from Artist*

The first step to run the Hspice simulation is to obtain a hspice netlist. This can be archived by "setup->Simulator/Directory/Host…" and set the simulator to "hsipceD" (Fig 13).



Choosing Simulator/Directory/Host -- Virtuoso® Analog Design Environn

Simulator: hspiceD

Project Directory: ~/simulation

Host Mode: ● local ○ remote ○ distributed

Host:

Remote Directory:

FIG. 13 choose "hspiceD" as simulator

In addition to choose "hspiceD" as simulator, we also need to fill in some necessary parameters for simulation such as choosing analysis type and deciding simulation periods (same as the step 2 for Spectre simulation).

After the previous works are all done, we can start to write out the hspice simulation netlist from Artist by click "Simulation->Netlist->Create Final" in Artist simulation window (Fig 14). The final simulation netlist window appeared after the netlist transfer procedure completed. You can save the simulation netlist from the simulation netlist window by click "File->Save As". In our case, we save the Hspice simulation netlist as "test_buffer.sp".

2. Add model include section and other control statements

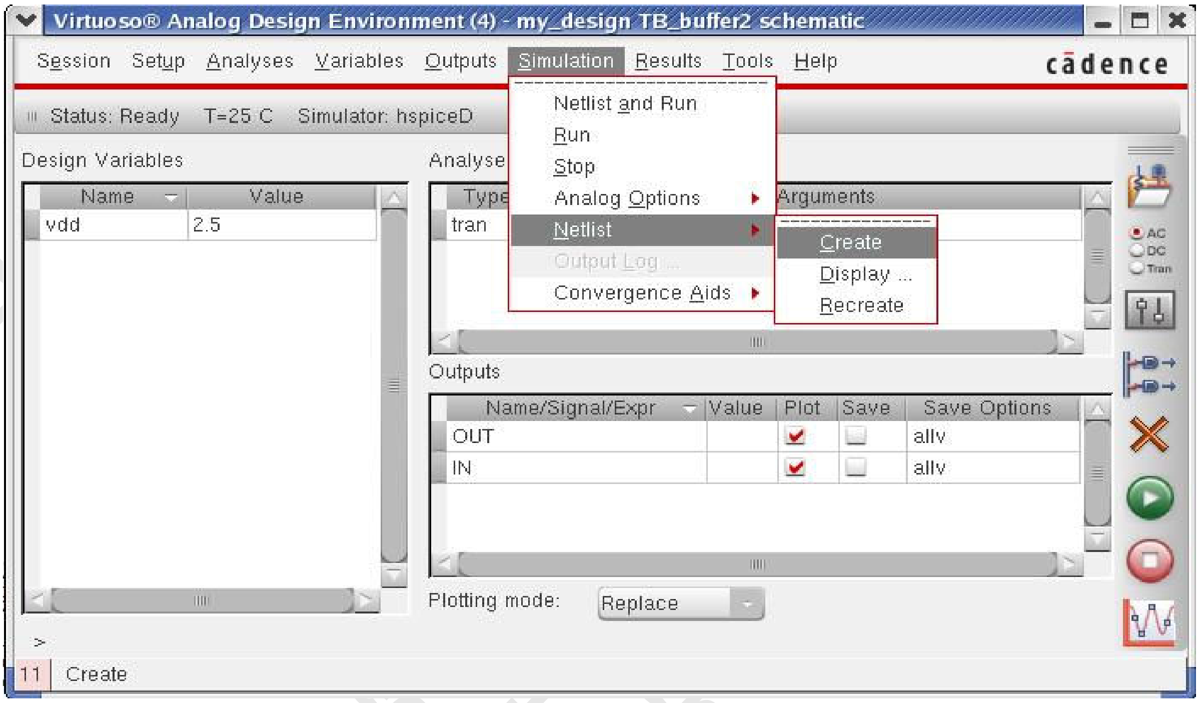
After obtained the Hspice simulation netlist, we still need to add the information that is need for later simulation into the netlist. The extra lines that we added into our netlist includes model include section, and output control section (fig 15).

3. Run Hspice simulation and check the simulation results

We can start to run the Hspice simulation under UNIX command prompt with the netlist that we obtained from step 2.

```
%hspice test_buffer.sp
```

And the simulation result can be obtained (fig 15) after the simulation completed.



Status: Ready T=25 C Simulator: hspiceD

Design Variables

Name	Value
vdd	2.5

Analyses

Type	Arguments
tran	

Outputs

Name/Signal/Expr	Value	Plot	Save	Save Options
OUT		☑	□	allv
IN		☑	□	allv

Plotting mode: Replace

```
>
11  Create
```

FIG. 14 Write out hspice simulation netlist

```
*INCLUDE FILES
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" probe
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_bbmvar5
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_5vna
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_dio_m
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_rtmom
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_bbmvar
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_disres
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_mim
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_dio_5
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_bip5
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_dio_dnw
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_m
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_5
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_res
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_bip
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_dio_dnw_5
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_dio
.lib "/net/hc1lnx50/usr2/shliu/018/PDK/models/hspice/cmn018_assp_5v_v1d1.l" tt_na
.PLOT V(IN) V(OUT)
```

FIG. 15 Add model-include and output control statement to hspice netlist

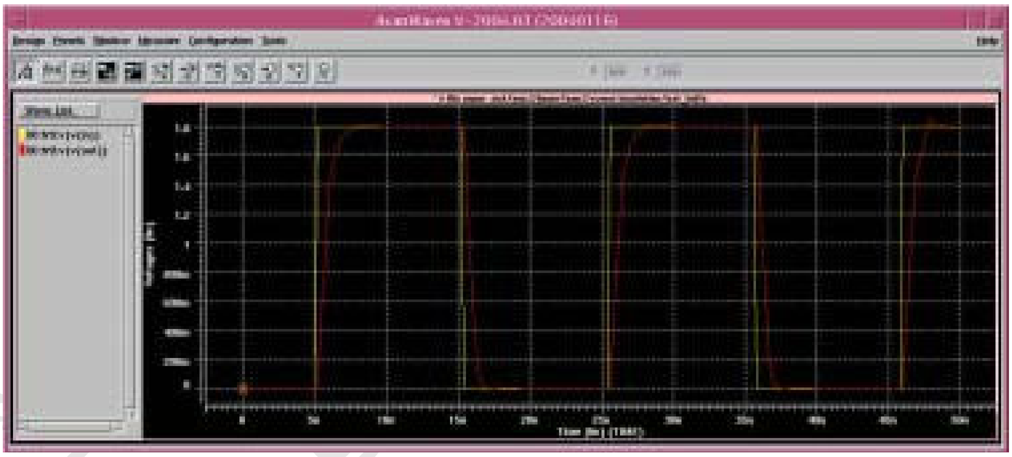


FIG. 16 Hspice simulation results

Chapter 4 Layout creation

After completed the pre-layout simulation and make sure the functionality and the circuit performance are all correct and in the design specifications, we can now start to create the corresponding layout for our design. There are many ways to create the layout including the tradition manual way and other automatic ways. Generally, the tradition layout creation method is to draw the layer geometries based on schematic manually, which is usually laboured and time-consuming. To take the advantage of using a PDK, we use another more efficient way to create the layout for our design. The layout creation procedures for our automatic way are partitioned into three parts: “Schematic-Driven-Layout”, “Components Placement” and “Auto route and Manual route”.

Schematic- Driven- Layout

Steps for Schematic-Driven-Layout method:

1. Open the schematic view of our design

2. From the schematic menu select “Launch -> Layout XL”

After selecting this option, a small dialog box will first open to let users select the cell name and view name for the layout. Upon finished the selection of the cell name and view name, a Virtuoso XL layout window popup for layout generation.

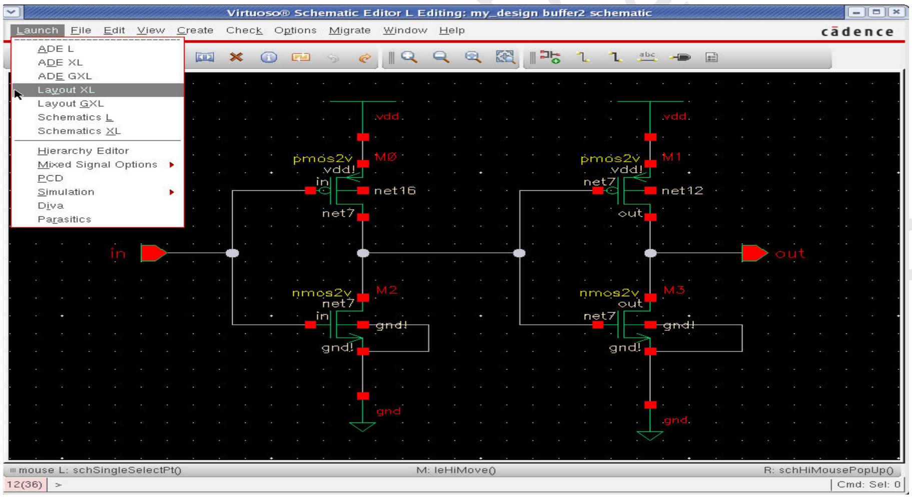
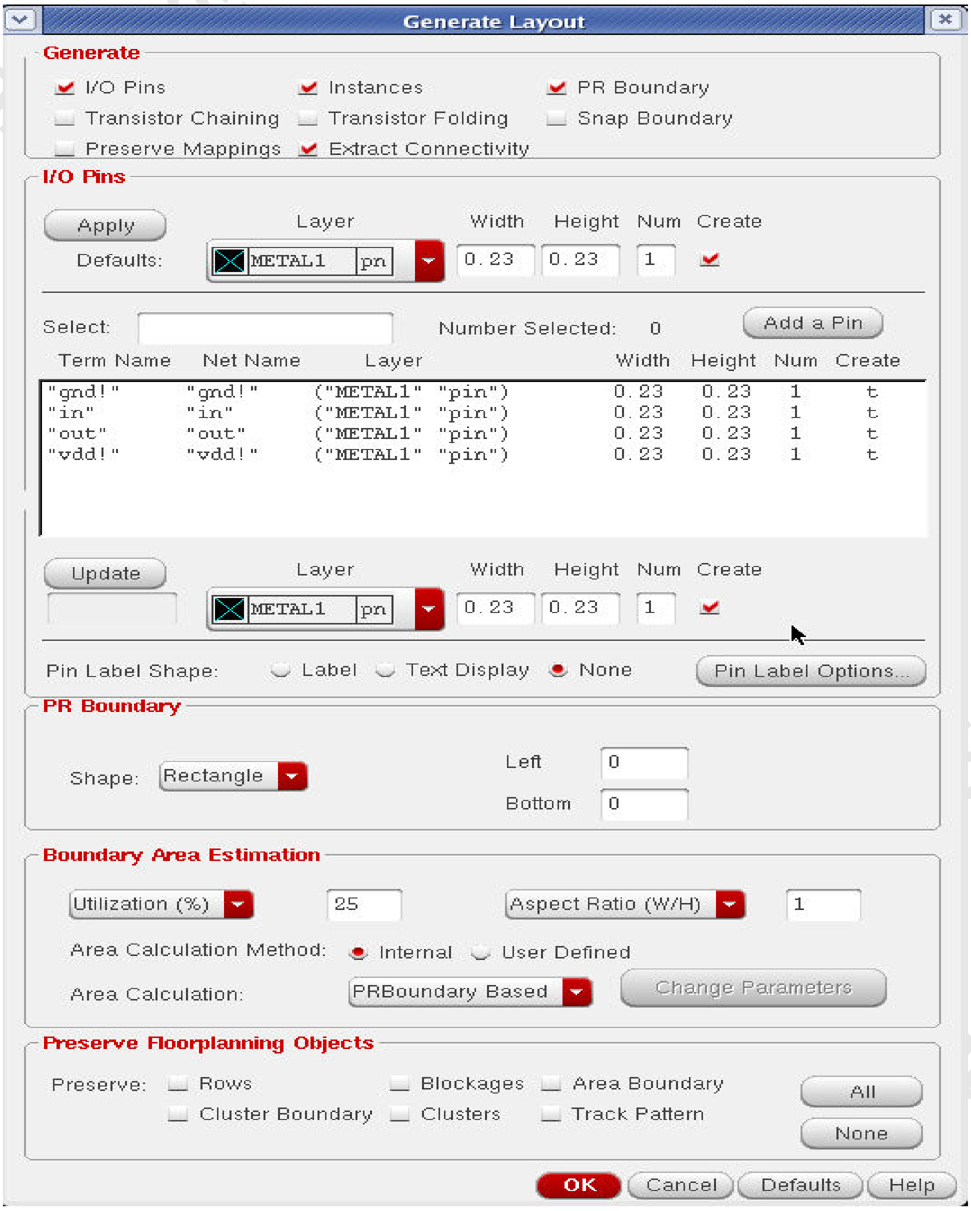


FIG. 17 invoke Virtuoso XL for schematic-driven-layout

3. From the Virtuoso XL layout menu select “Connectivity -> Generate -> All From Source …”

A layout generation options window appeared and then users can set pin layers, pin width, pin height, boundary layer …and so on for layout generation. Click on “Apply” after specifying the I/O pin layers to M1/pin, then click “OK”..



Generate

☑ I/O Pins ☑ Instances ☑ PR Boundary

☐ Transistor Chaining ☐ Transistor Folding ☐ Snap Boundary

☐ Preserve Mappings ☑ Extract Connectivity

I/O Pins

	Layer		Width	Height	Num	Create
Defaults:	METAL1	pn	0. 23	0. 23	1	✓

Select: Number Selected: 0

Term Name	Net Name	Layer	Width	Height	Num	Create
"gnd!"	"gnd!"	("METAL1" "pin")	0. 23	0. 23	1	t
"in"	"in"	("METAL1" "pin")	0. 23	0. 23	1	t
"out"	"out"	("METAL1" "pin")	0. 23	0. 23	1	t
"vdd!"	"vdd!"	("METAL1" "pin")	0. 23	0. 23	1	t

Layer		Width	Height	Num	Create
METAL1	pn	0. 23	0. 23	1	✓

Pin Label Shape: ○ Label ○ Text Display ● None

PR Boundary

Shape: Rectangle

Left 0

Bottom 0

Boundary Area Estimation

Utilization (%) 25 Aspect Ratio (W/H) 1

Area Calculation Method: ● Internal ○ User Defined

Area Calculation: PRBoundary Based

Preserve Floorplanning Objects

Preserve: ☐ Rows ☐ Blockages ☐ Area Boundary

☐ Cluster Boundary ☐ Clusters ☐ Track Pattern

FIG. 18 layout generation options window

After finished the selection of above information, some rectangles that represent the transistors and I/O pins will show up in the bottom of the layout window.

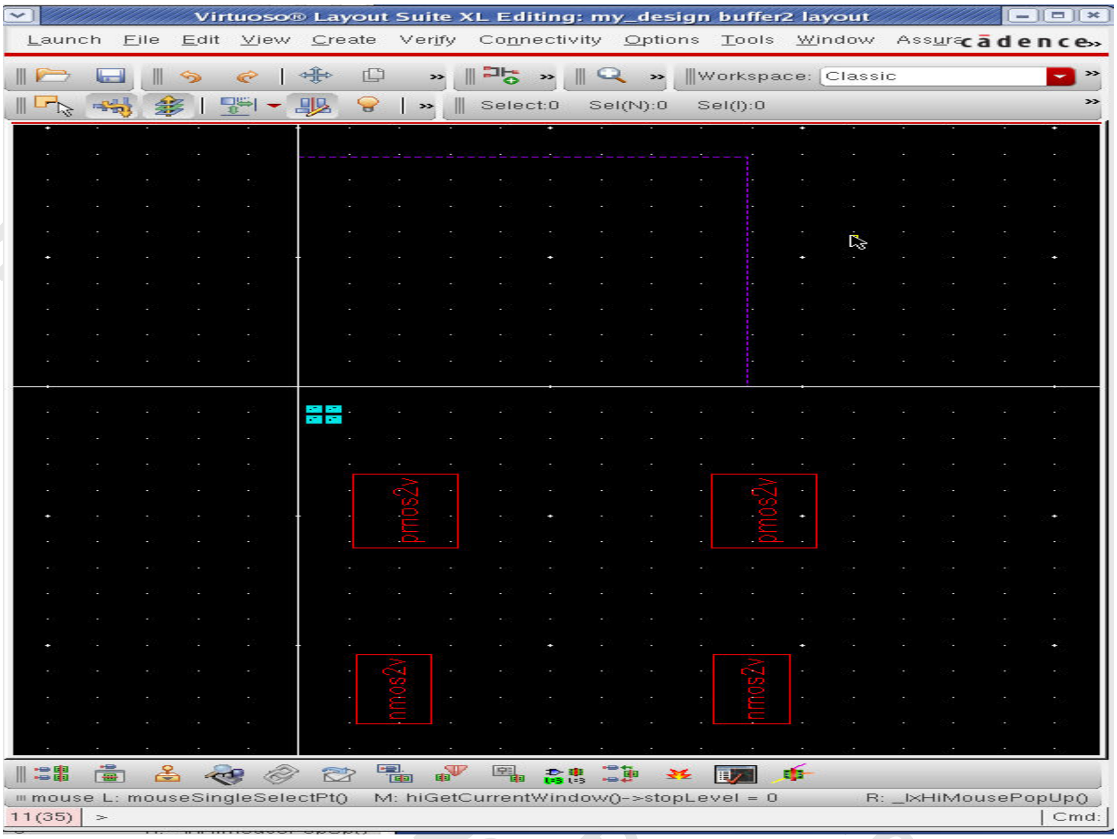

Virtuoso® Layout Suite XL Editing: my_design buffer2 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura
cadence
Workspace: Classic
Select:0 Sel(N):0 Sel(I):0
pmos2v
pmos2v
nmos2v
nmos2v
mouse L: mouseSingleSelectPt() M: hiGetCurrentWindow()->stopLevel = 0 R: _lxHiMousePopUp()
11(35) >
Cmd:

FIG. 18 I/O pins and devices generated in layout window

Components placement

The next step is to do the device placement. The only one thing that you need to do is to place all the components and I/O pins in the layout window into the design area (cell boundary). By selecting devices/IO pins and dragging them to proper locations inside the design area, we can complete the component placement. During the device movement and placement, the lines represent the connections of select object to other objects will show up. This can help you to decide where to properly locate the selected object (Fig 19).

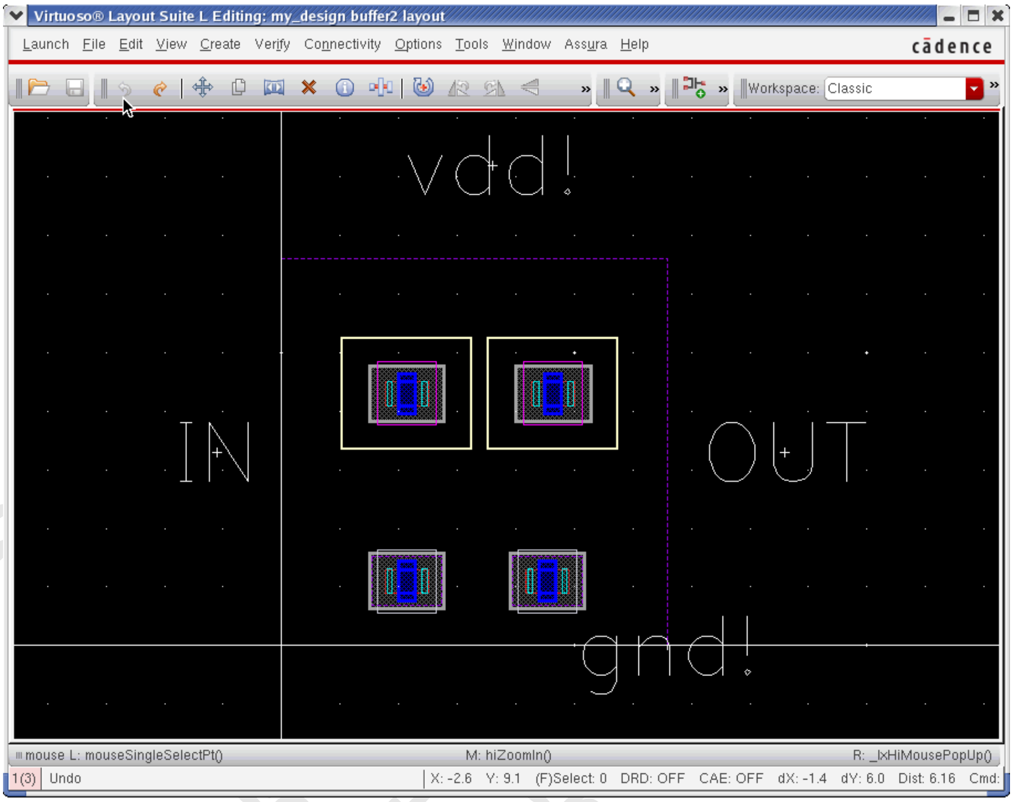

Virtuoso® Layout Suite L Editing: my_design buffer2 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura Help
cādence
Workspace: Classic
vdd!
IN
OUT
gnd!
mouse L: mouseSingleSelectPt()
M: hiZoomIn()
R: _lxHiMousePopUp()
1(3) Undo
X: -2.6 Y: 9.1 (F)Select: 0 DRD: OFF CAE: OFF dX: -1.4 dY: 6.0 Dist: 6.16 Cmd:

FIG. 19 Place all devices into design area in layout window

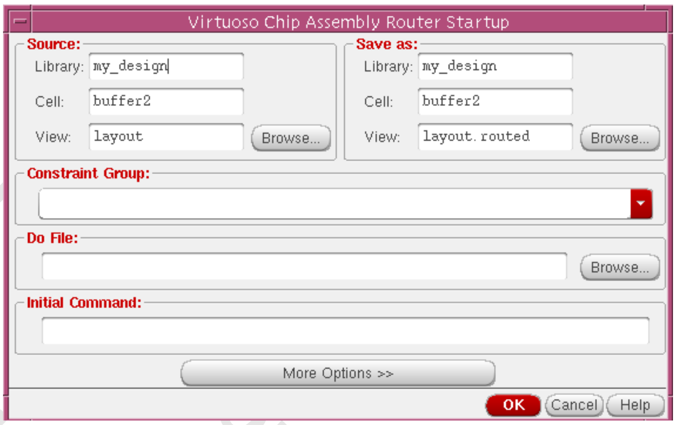
Auto-route and manual route

When the component placement is completed, the next step is to perform the device routing. Since TSMC's PDK also support the ICC rules for Virtuoso Chip Assembly Router (VCAR), we will use the VCAR to take full advantage of automated routing capability. After that, if we don't feel comfortable with the routing results, we can perform some manual routes to fix the routing results.

The steps for auto route are as follows:

1. To invoke the router,
   (a) Select Launch – Layout GXL.
   (b) From the toolbar, select Window – Assistants and turn on Task Assistant.
   The Task Assistant appears on the right side of the design window.
   (c) Turn on Chip Assembly Router.
   (d) From the toolbar, select Routing – Start Router.

2. Specify the routing view name in the popup window to initiate the VCAR (Fig 20).
3. Click "Autoroute->Detail route->Detail router" to complete the auto routing (Fig 21).


Virtuoso Chip Assembly Router Startup

Source:
Library: my_design
Cell: buffer2
View: layout

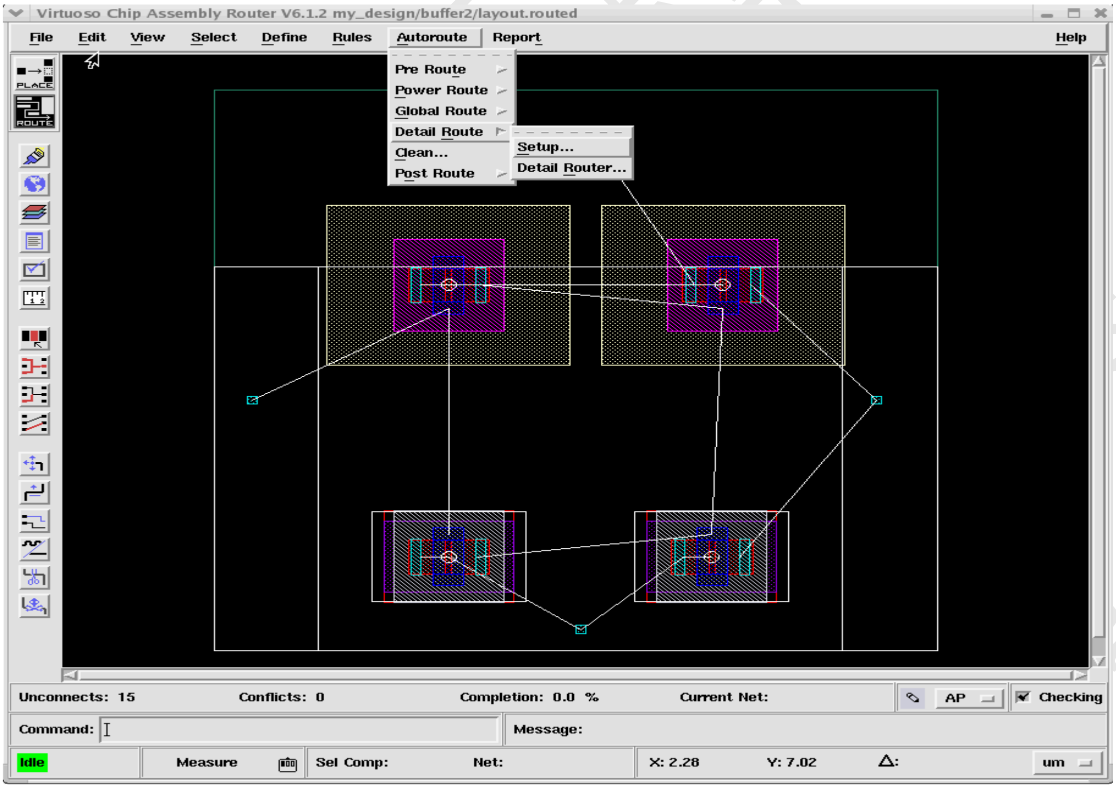
Save as:
Library: my_design
Cell: buffer2
View: layout.routed

Constraint Group:

Do File:

Initial Command:

FIG. 20 Setup for VCR


Virtuoso Chip Assembly Router V6.1.2 my_design/buffer2/layout.routed

Unconnects: 15
Conflicts: 0
Completion: 0.0 %
Current Net:
Command:
Message:
Idle
Measure
Sel Comp:
Net:
X: 2.28
Y: 7.02
Δ:
um

FIG. 21 Perform automatic routing

If you don’t feel comfortable with the results from VCR, you can manually fix the routing results in the layout window. For our design, the final layout after manually fixed is shown in Fig 22.

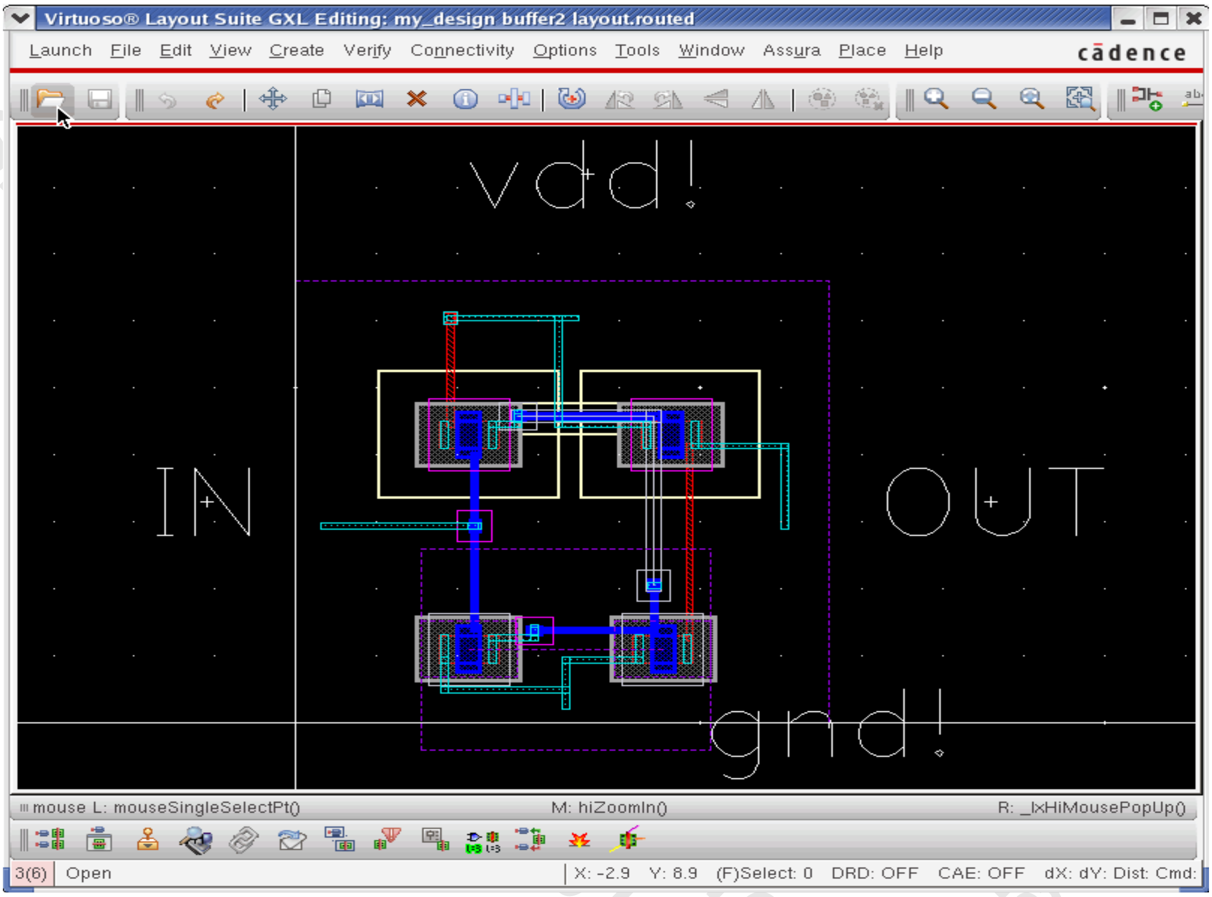

Virtuoso® Layout Suite GXL Editing: my_design buffer2 layout.routed
Launch File Edit View Create Verify Connectivity Options Tools Window Assura Place Help
cādence
vdd!
IN
OUT
gnd!
mouse L: mouseSingleSelectPt()
M: hiZoomIn()
R: _lxHiMousePopUp()
3(6)
Open
X: -2.9 Y: 8.9 (F)Select: 0 DRD: OFF CAE: OFF dX: dY: Dist: Cmd:

FIG. 22 Final layout

Chapter 5 Physical verification

After the layout creation is completed, we have to start the physical verification to make sure this layout is DRC free and each device in the layout is completely match to its corresponding component in original schematic. After that, the parasitic extraction is also need to be performed for post-layout simulation to make sure our design still work well after taking the parasitic R & C effects into account. Generally, the physical verification procedures can be divided into three parts: the design rule check (DRC), layout V.S. schematic check (LVS) and parasitic extraction (RCX). Furthermore, based on the differences on running methods and supported tools, they can be classified into different flows. In order to satisfy most of users, TSMC's PDK supports varied kinds of decks to be used for different kinds of flows. Currently, the supported tools are Assura and Calibre, and the supported flows are "Assura DFII flow", "Assura CDL flow", "Calibre CDL flow" and "Calibre interactive flow". In real design, users only need to choose one of these physical verification flows and have no need to go through all the flows.

Physical verification using Assura

In this section, we will introduce the "Assura DFII flow" and "Assura CDL flow" for Assura users.

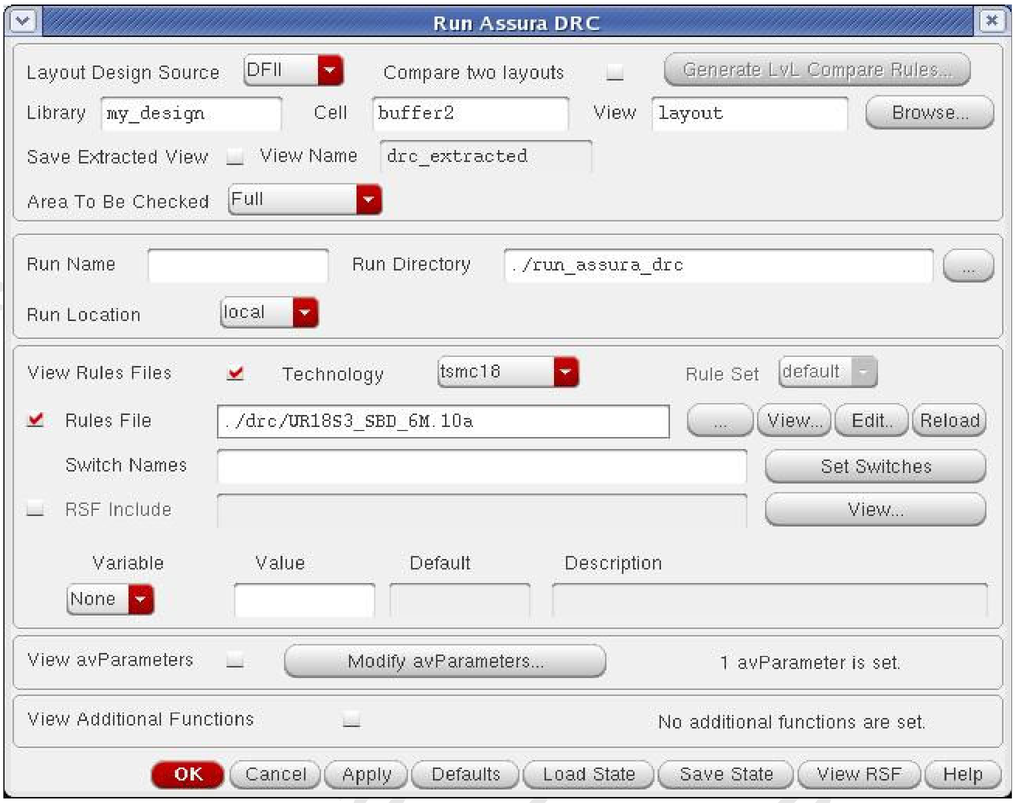
Assura DFII flow

When choosing Assura as the physical verification tools, most of people will go through their physical verification with DFII flow because of its conveniency and user friendly. The physical verification procedures for Assura DFII flow are as follows:

Assura DFII DRC

1. Click "Assura->Run DRC…" in layout window to invoke Assura DRC graphic user interface.
2. Fill in the "Assura run directory" and select the "Technology" field to "tsmc18" (Fig. 23) in Assura DRC window. You can also set the Assura DRC switches and other parameters if needed.

3. Click “OK” to run the Assura DRC and see the result. If the layout isn’t DRC free, you have to manually re-edit the layout and re-run the DRC check to make it DRC free.



Run Assura DRC

Layout Design Source: DFII

Compare two layouts □

Library: my_design Cell: buffer2 View: layout

Save Extracted View □ View Name: drc_extracted

Area To Be Checked: Full

Run Name:

Run Directory: ./run_assura_drc

Run Location: local

View Rules Files ☑ Technology: tsmc18 Rule Set: default

☑ Rules File: ./drc/UR18S3_SBD_6M.10a

Switch Names:

□ RSF Include

Variable	Value	Default	Description
None			

View avParameters □ 1 avParameter is set.

View Additional Functions □ No additional functions are set.

FIG. 23 Assura DRC setup window

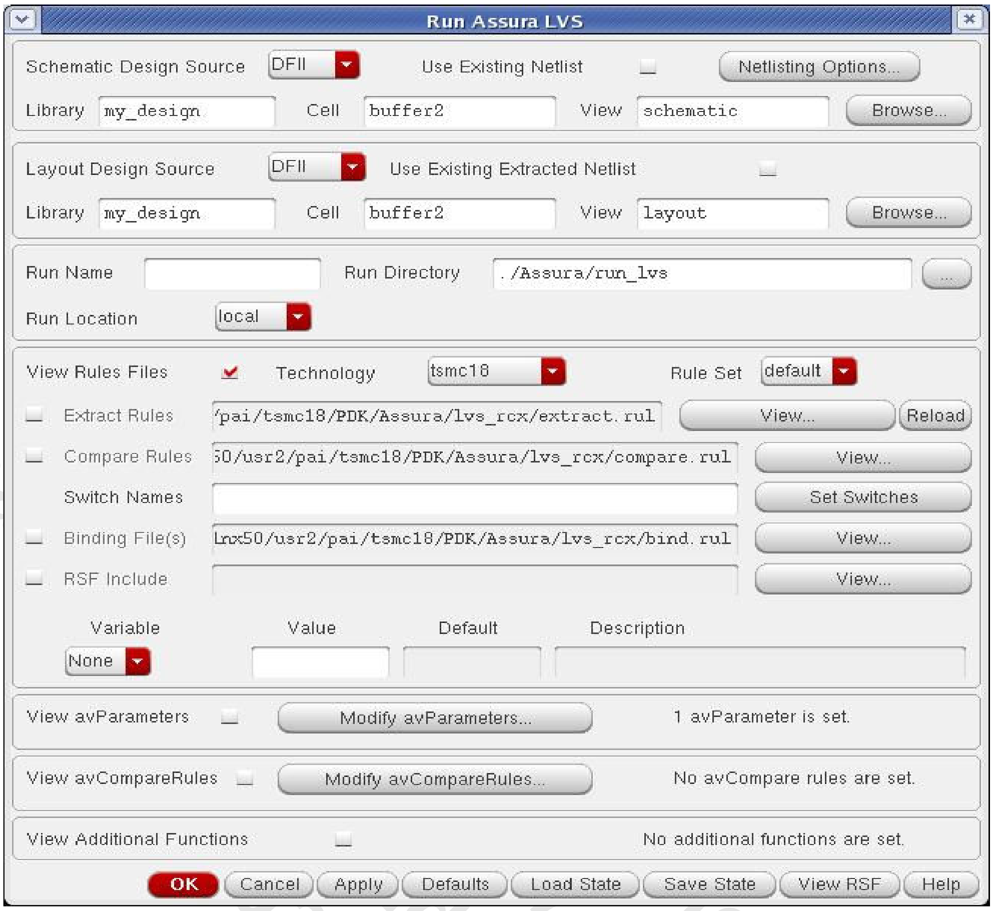
Assura DFII LVS

After the layout has no DRC violations (DRC free), the next step is to run the LVS check to make sure the layout is totally match to the schematic.

1. Click “Assura->Run LVS…” in layout window to invoke Assura LVS graphic user interface.

2. Fill in the “Assura run directory” and select the “Technology” field to “tsmc18” (Fig. 24) in Assura LVS window. You can also set the Assura LVS switches and other parameters if needed.

3. Click “OK” to run the Assura LVS and see the result. If the layout isn’t matched to schematic, you have to manually re-edit the layout and re-run the LVS check to make the LVS result matched.



Run Assura LVS

Schematic Design Source: DFII
Use Existing Netlist □
Library: my_design Cell: buffer2 View: schematic

Layout Design Source: DFII
Use Existing Extracted Netlist □
Library: my_design Cell: buffer2 View: layout

Run Name:
Run Directory: ./Assura/run_lvs
Run Location: local

View Rules Files ☑ Technology: tsmc18 Rule Set: default
□ Extract Rules: /pai/tsmc18/PDK/Assura/lvs_rcx/extract.rul
□ Compare Rules: 50/usr2/pai/tsmc18/PDK/Assura/lvs_rcx/compare.rul
Switch Names:
□ Binding File(s): lnx50/usr2/pai/tsmc18/PDK/Assura/lvs_rcx/bind.rul
□ RSF Include:

Variable: None Value: Default: Description:

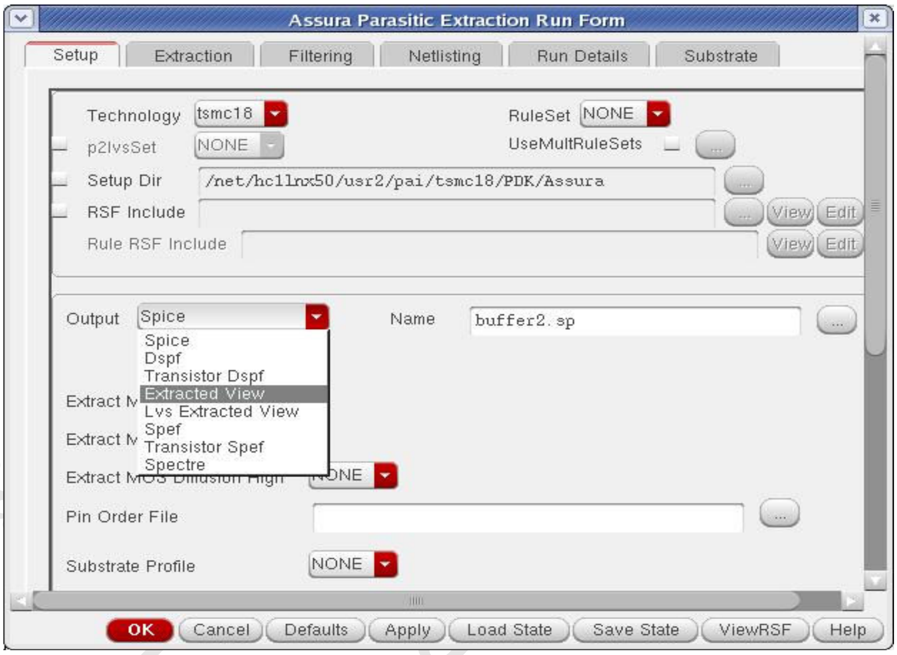
View avParameters □ 1 avParameter is set.
View avCompareRules □ No avCompare rules are set.
View Additional Functions □ No additional functions are set.

FIG. 24 Assura LVS setup window

Assura DFII RCX

When the layout is DRC free and LVS clean, the next step is to perform the RC extraction. This step is to prepare the layout extracted netlist for post-layout simulation.

1. Click “Assura->Run RCX…” in layout window to invoke Assura RCX graphic user interface.
2. Select “Output” to “Extracted View” in “setup” folder of Assura RCX window to output extract result to “av_extracted” view (Fig 25).
3. In the “Extraction” folder of Assura RCX window, select the “extraction mode” to “C only” (if you want to extract only C), set the “Name space” to “Schematic Names” and fill in the “Ref Node”(here we use “gnd!”)(Fig 26).
4. Click “OK” to start the Assura RC extraction. After the RC extraction is completed, a new view (“av_extracted” view, Fig 27) which contains not only the original components but also the parasitic devices will be generated and then can be used for post-layout simulation.



Technology tsmc18

RuleSet NONE

☐ p2lvsSet NONE

UseMultRuleSets ☐

☐ Setup Dir /net/hc11nx50/usr2/pai/tsmc18/PDK/Assura

☐ RSF Include

Rule RSF Include

Output Spice

- Spice
- Dspf
- Transistor Dspf
- Extracted View
- Lvs Extracted View
- Spef
- Transistor Spef
- Spectre

Name buffer2.sp

Extract M

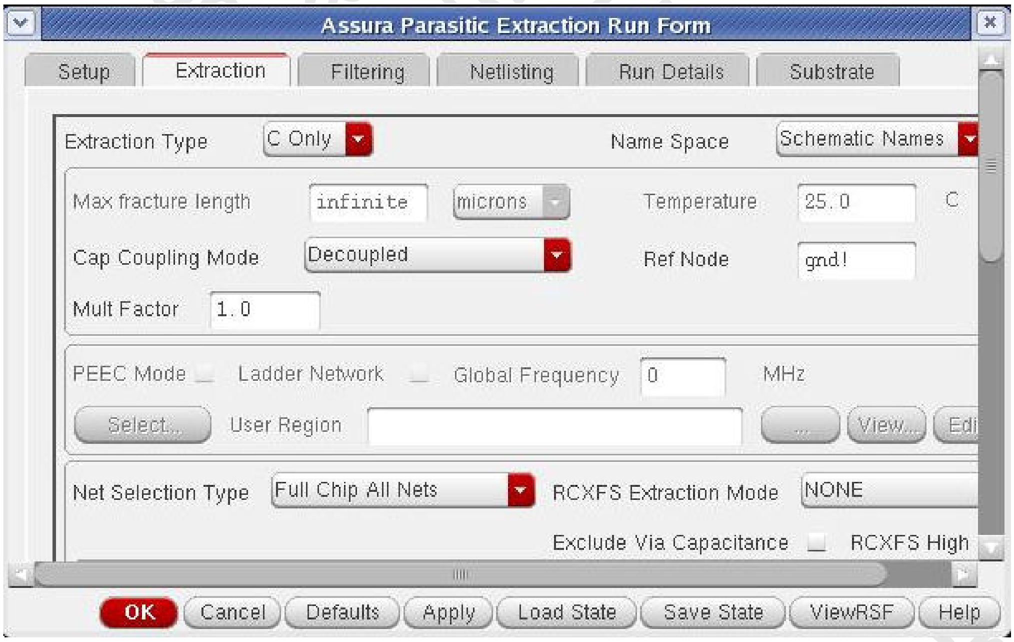
Extract M

Extract MOS Diffusion High NONE

Pin Order File

Substrate Profile NONE

FIG. 25 Setup folder of Assura RCX window



Extraction Type C Only

Name Space Schematic Names

Max fracture length infinite microns

Temperature 25.0 C

Cap Coupling Mode Decoupled

Ref Node gnd!

Mult Factor 1.0

PEEC Mode ☐ Ladder Network ☐ Global Frequency 0 MHz

User Region

Net Selection Type Full Chip All Nets

RCXFS Extraction Mode NONE

Exclude Via Capacitance ☐ RCXFS High

FIG. 26 Extraction folder of Assura RCX window

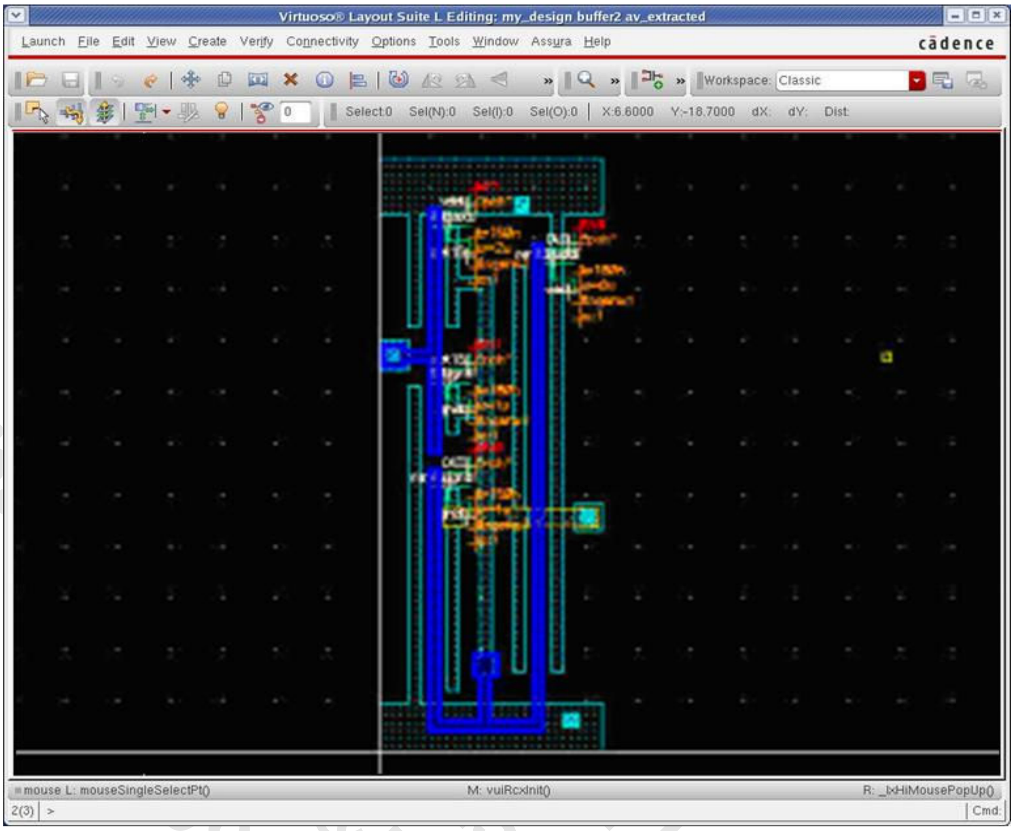

Virtuoso® Layout Suite L Editing: my_design buffer2 av_extracted
Launch File Edit View Create Verify Connectivity Options Tools Window Assura Help
cādence
Workspace: Classic
Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:6.6000 Y:-18.7000 dX: dY: Dist:
mouse L: mouseSingleSelectPt()
M: vuiRcxInit()
R: _lxHiMousePopUp()
2(3) >
Cmd:

FIG. 27 Assura av_extracted view

Assura CDL flow

In addition to the DFII flow, some users may use Assura CDL flow for their physical verification. Currently, TSMC's PDK only supports the Assura CDL flow in DRC and LVS verification.

Prepare Data

Before starting the Assura CDL verification flow, users have to prepare the CDL netlist and the GDS file.

Export CDL netlist:

1. Select "File->Export->CDL…" from the CIW window to export the schematic netlist

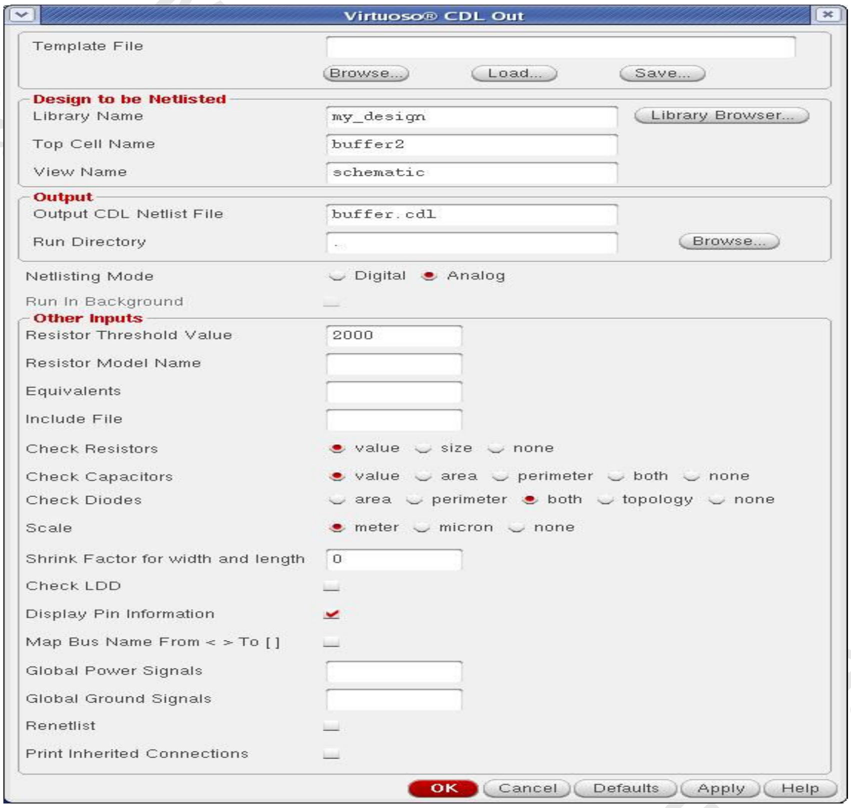
2. Select the top cell name to your design, make sure the netlist mode is "Analog", and fill in the output file name of the netlist (Fig 28).

3. **If the "source_added" file (empty sub-circuit file) is provided along with the LVS deck, you also have to attach this file to the output netlist file (obtained from step #2).**

Export GDS file:

1. Select "File->Export->Stream…" from the CIW window to export the GDS file.

2. Select the top cell name to your design, fill in the "layer map table" and the output file name of the GDS file (Fig 29).



Virtuoso® CDL Out

Template File:

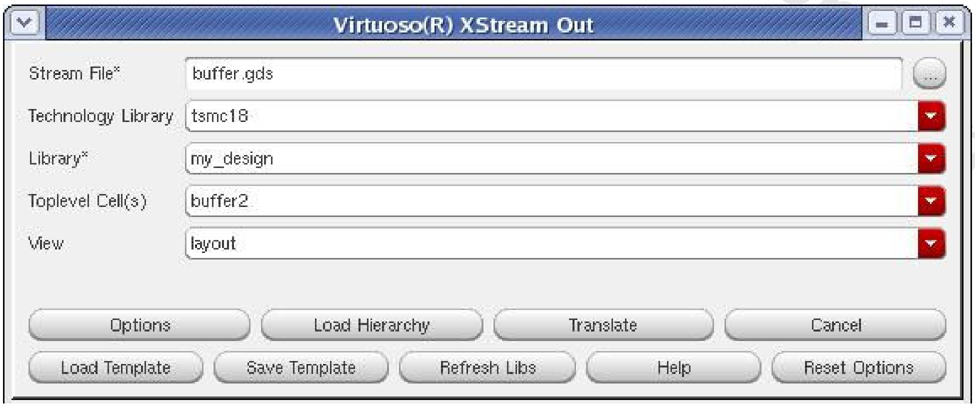
Design to be Netlisted
Library Name: my_design
Top Cell Name: buffer2
View Name: schematic

Output
Output CDL Netlist File: buffer.cdl
Run Directory: .

Netlisting Mode: ○ Digital ● Analog
Run In Background: □

Other Inputs
Resistor Threshold Value: 2000
Resistor Model Name:
Equivalents:
Include File:
Check Resistors: ● value ○ size ○ none
Check Capacitors: ● value ○ area ○ perimeter ○ both ○ none
Check Diodes: ○ area ○ perimeter ● both ○ topology ○ none
Scale: ● meter ○ micron ○ none
Shrink Factor for width and length: 0
Check LDD: □
Display Pin Information: ☑
Map Bus Name From < > To [ ]: □
Global Power Signals:
Global Ground Signals:
Renetlist: □
Print Inherited Connections: □

FIG. 28 Export CDL netlist



Virtuoso(R) XStream Out

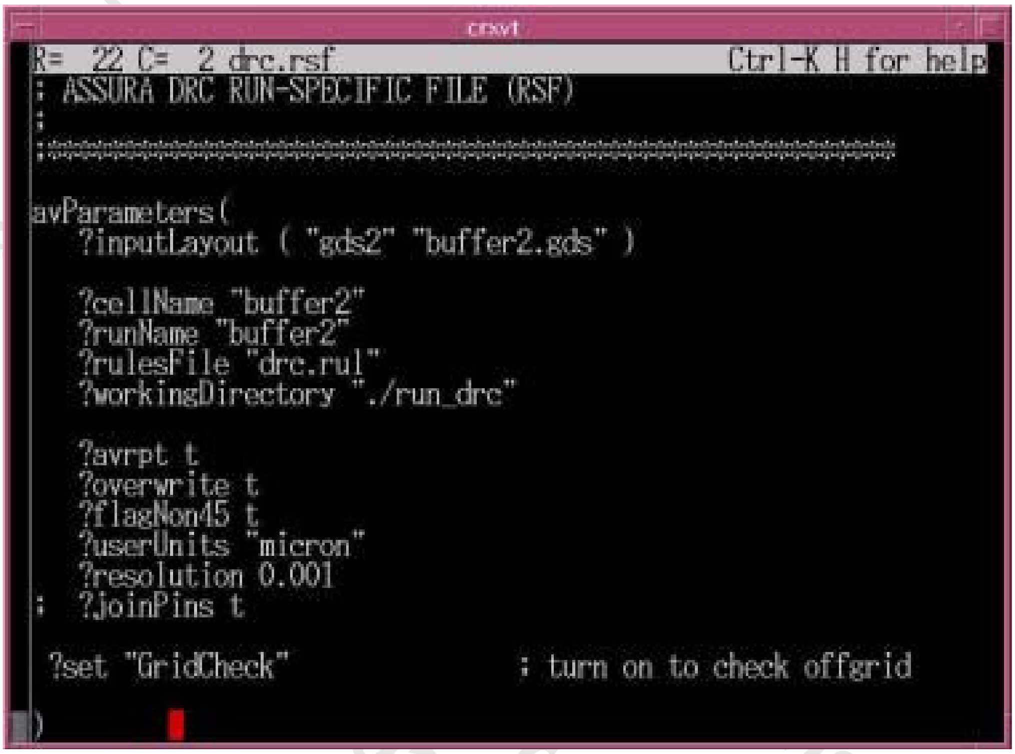
Stream File*: buffer.gds
Technology Library: tsmc18
Library*: my_design
Toplevel Cell(s): buffer2
View: layout

FIG. 29 Export GDS layout

Assura CDL DRC

After the GDS file is exported, you have to edit the Assura ASCII run specific file (RSF) to specify the input data, output data and types of checks to be run in Assura DRC check.

1. Edit Assura DRC RSF file to specify the necessary information (Fig 30).



```
R=  22 C=  2 drc.rsf                                  Ctrl-K H for help
; ASSURA DRC RUN-SPECIFIC FILE (RSF)
;
;****************************************************************

avParameters(
   ?inputLayout ( "gds2" "buffer2.gds" )

   ?cellName "buffer2"
   ?runName "buffer2"
   ?rulesFile "drc.rul"
   ?workingDirectory "./run_drc"

   ?avrpt t
   ?overwrite t
   ?flagNon45 t
   ?userUnits "micron"
   ?resolution 0.001
;  ?joinPins t

 ?set "GridCheck"               ; turn on to check offgrid

)
```

FIG. 30 Edit Assura DRC RSF file

2. Run Assura DRC checking in UNIX command prompt and check the results.

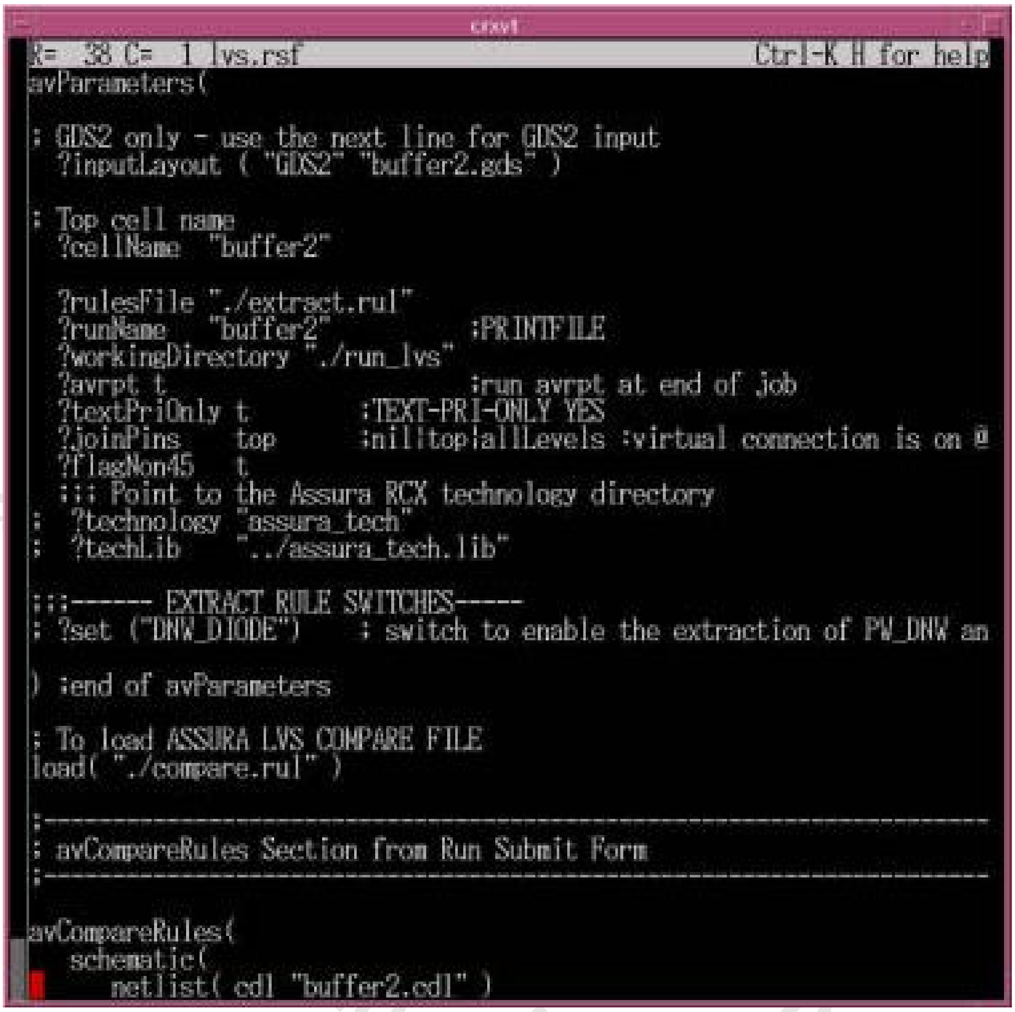
```
%asssura drc.rsf
```

3. If the GDS isn't DRC free, you have to go back to fix you layout and re-stream out the GDS for DRC check again.

Assura CDL LVS

When the GDS file is DRC free, you can continue to run the Assura CDL LVS.

1. Edit Assura LVS RSF file to specify the necessary information (Fig 31).



```
R=  38 C=  1 lvs.rsf                                        Ctrl-K H for help
avParameters(

; GDS2 only - use the next line for GDS2 input
  ?inputLayout ( "GDS2" "buffer2.gds" )

; Top cell name
  ?cellName  "buffer2"

  ?rulesFile "./extract.rul"
  ?runName   "buffer2"           ;PRINTFILE
  ?workingDirectory "./run_lvs"
  ?avrpt t                       ;run avrpt at end of job
  ?textPriOnly t        ;TEXT-PRI-ONLY YES
  ?joinPins    top      ;nil|top|allLevels ;virtual connection is on @
  ?flagNon45   t
  ;;; Point to the Assura RCX technology directory
;  ?technology "assura_tech"
;  ?techLib    "../assura_tech.lib"

;;;------ EXTRACT RULE SWITCHES-----
; ?set ("DNW_DIODE")    ; switch to enable the extraction of PW_DNW an

) ;end of avParameters

; To load ASSURA LVS COMPARE FILE
load( "./compare.rul" )

;------------------------------------------------------------------------
; avCompareRules Section from Run Submit Form
;------------------------------------------------------------------------

avCompareRules(
  schematic(
     netlist( cdl "buffer2.cdl" )
```

FIG. 31 Edit Assura LVS RSF file

2. Run Assura LVS checking in UNIX command prompt and check the results.

   %asssura lvs.rsf

3. If the GDS isn't LVS clean, you have to go back to fix you layout and re-stream out the GDS for LVS check again.

Assura CDL RCX

Currently, TSMC's PDK doesn't support the Assura RC extraction with CDL flow.

Physical verification using Calibre

In addition to support using Assura for physical verification, TSMC's PDK also supports using Calibre for physical verification. In this section, we will introduce the "Calibre CDL flow" and "Calibre interactive flow" for Calibre users.

Calibre CDL flow

Before starting the Calibre CDL physical verification flow, users have to prepare the CDL netlist and the GDS file first (please refer to the "prepare data" section in Assura CDL flow).

Calibre CDL DRC

After the GDS file is ready, you have to:

1. Edit the Calibre DRC deck and specify the GDS file name and top cell name for the deck (Fig 32).

2. Check the switches in the deck and correctly set these switches if need.

3. Run Calibre CDL flow DRC in UNIX command prompt and check the DRC results.

```
%calibre –drc –hier calibre.drc
```

4. If the layout isn't DRC free, you have to fix all DRC violations, re-stream out the GDS file and run the DRC check again.

Calibre CDL LVS

When the DRC checking is completed and the layout is DRC free, the next step is to perform the LVS checking.

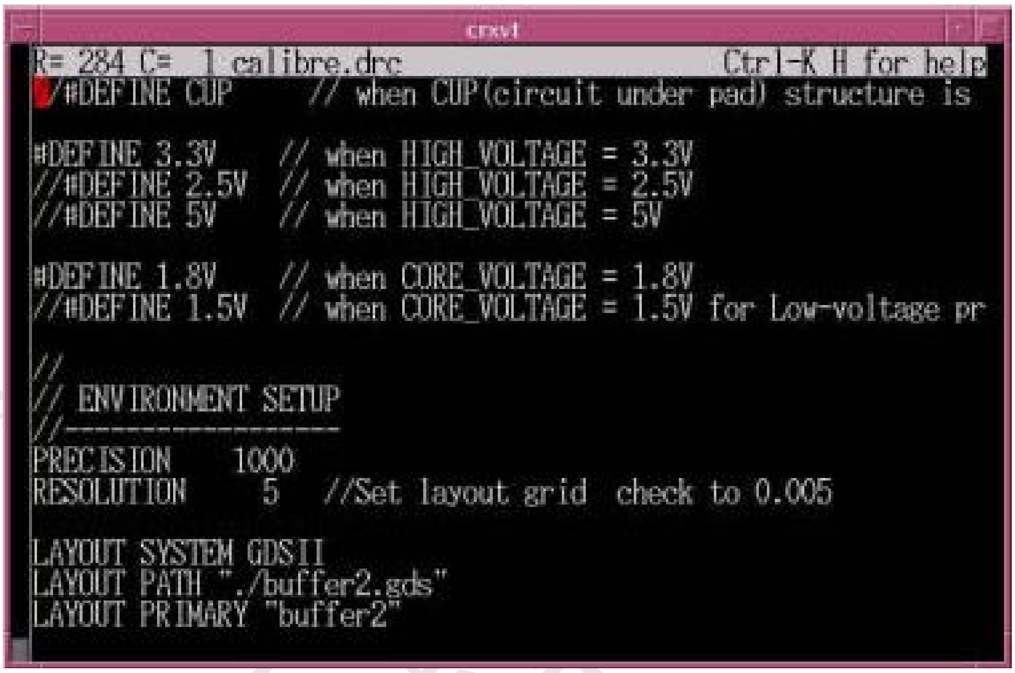
1. Edit the Calibre LVS deck and specify the GDS file name, top cell name and CDL netlist file for the deck (Fig 33).

2. Check the switches in the deck and correctly set these switches if need.

3. Run Calibre CDL flow LVS in UNIX command prompt and check the LVS results.

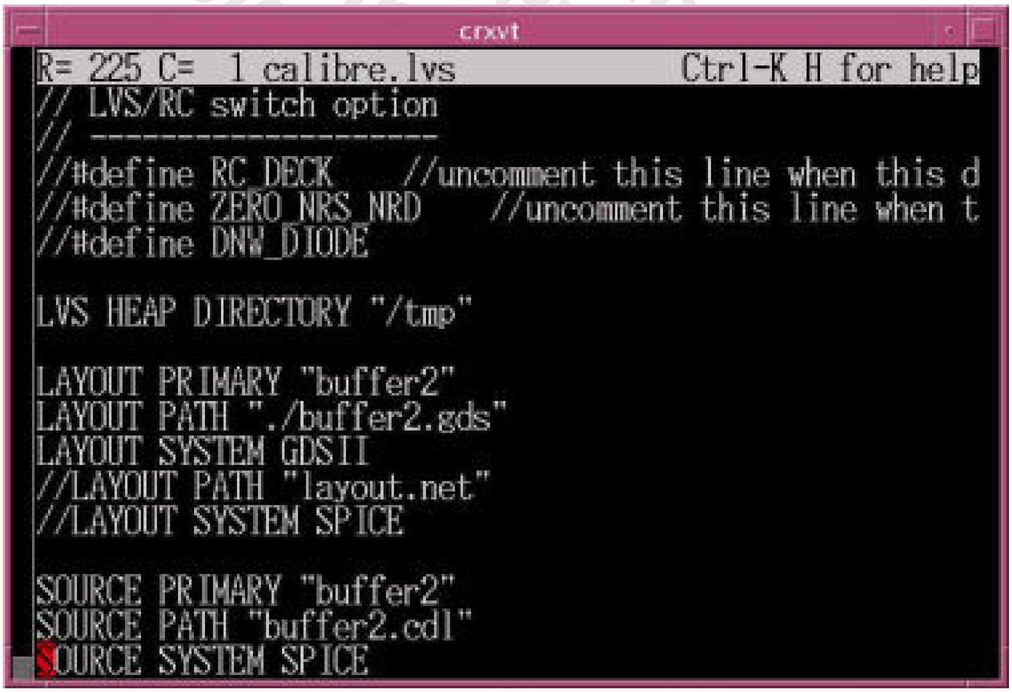
```
%calibre –lvs –hier –spi layout.net calibre.drc
```

4. If the layout isn't LVS clean, you have to fix the layout, re-stream out the GDS file and run the LVS check again.



```
R= 284 C=  1 calibre.drc                      Ctrl-K H for help
//#DEFINE CUP     // when CUP(circuit under pad) structure is

#DEFINE 3.3V    // when HIGH_VOLTAGE = 3.3V
//#DEFINE 2.5V  // when HIGH_VOLTAGE = 2.5V
//#DEFINE 5V    // when HIGH_VOLTAGE = 5V

#DEFINE 1.8V    // when CORE_VOLTAGE = 1.8V
//#DEFINE 1.5V  // when CORE_VOLTAGE = 1.5V for Low-voltage pr

//
// ENVIRONMENT SETUP
//-----------------
PRECISION    1000
RESOLUTION     5   //Set layout grid  check to 0.005

LAYOUT SYSTEM GDSII
LAYOUT PATH "./buffer2.gds"
LAYOUT PRIMARY "buffer2"
```

FIG. 32 Edit Calibre DRC deck



```
R= 225 C=  1 calibre.lvs                  Ctrl-K H for help
// LVS/RC switch option
// ---------------------
//#define RC_DECK     //uncomment this line when this d
//#define ZERO_NRS_NRD     //uncomment this line when t
//#define DNW_DIODE

LVS HEAP DIRECTORY "/tmp"

LAYOUT PRIMARY "buffer2"
LAYOUT PATH "./buffer2.gds"
LAYOUT SYSTEM GDSII
//LAYOUT PATH "layout.net"
//LAYOUT SYSTEM SPICE

SOURCE PRIMARY "buffer2"
SOURCE PATH "buffer2.cdl"
SOURCE SYSTEM SPICE
```

FIG. 33 Edit Calibre LVS deck

Calibre CDL RCX

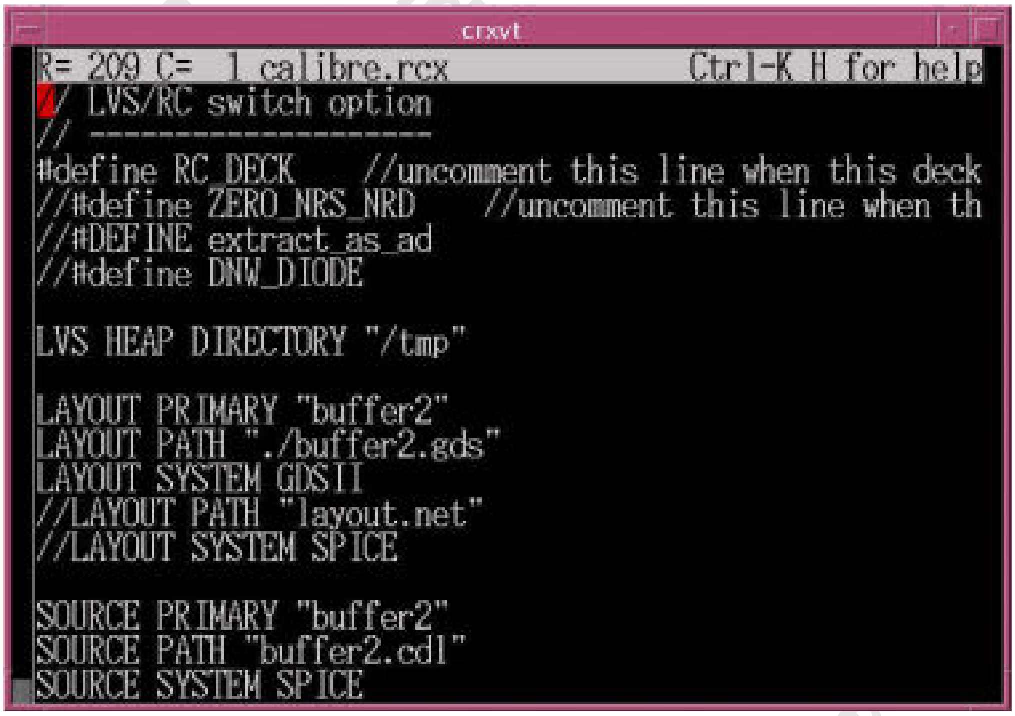
After the layout is DRC free and LVS clean, the next step is to perform the RC extraction. This step is to prepare the layout extracted netlist for post-layout simulation.

1. Edit the Calibre RC deck and specify the GDS file name, top cell name and CDL netlist file for the deck (Fig 34).

2. Check the switches in the deck and correctly set these switches if need.

3. Run Calibre CDL RCX in UNIX command prompt and obtain the extracted netlist. (Use extract C only for illustration)

```
%calibre –xrc –phdb calibre.drc
%calibre –xrc –pdb –c calibre.drc
%calibre –xrc –fmt -c calibre.drc
```



```
R= 209 C=  1 calibre.rcx                    Ctrl-K H for help
// LVS/RC switch option
// --------------------
#define RC_DECK    //uncomment this line when this deck
//#define ZERO_NRS_NRD    //uncomment this line when th
//#DEFINE extract_as_ad
//#define DNW_DIODE

LVS HEAP DIRECTORY "/tmp"

LAYOUT PRIMARY "buffer2"
LAYOUT PATH "./buffer2.gds"
LAYOUT SYSTEM GDSII
//LAYOUT PATH "layout.net"
//LAYOUT SYSTEM SPICE

SOURCE PRIMARY "buffer2"
SOURCE PATH "buffer2.cdl"
SOURCE SYSTEM SPICE
```

FIG. 34 Edit Calibre RCX deck

Calibre GUI flow (with Calibre-view)

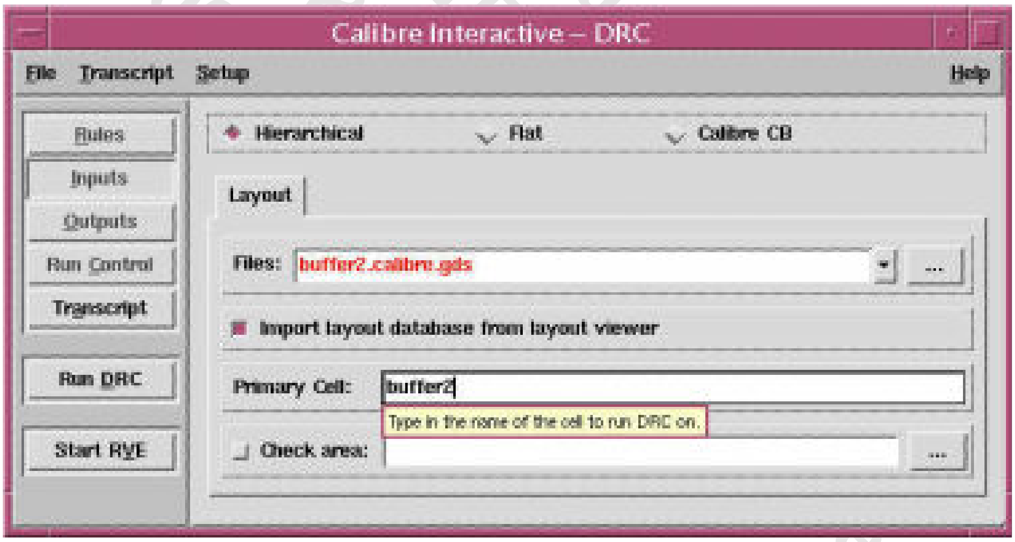
If Calibre users don't like the CDL flow, TSMC's PDK also support the Calibre GUI mode flow which integrate the Calibre interface into Cadence Virtuoso environment.

Calibre GUI DRC

Calibre GUI mode flow is working under Cadence Virtuoso environment and its DRC verification procedures are as follows:

1. Click "Calibre->Run DRC" in layout window to invoke Calibre DRC graphic user interface.
2. Specify the "Calibre-DRC rules file" and the "Primary cell" in Calibre interactive-DRC window (Fig 35). If you need to change some DRC switches, you have to edit the Calibre DRC deck first.
3. Click "Run DRC" button to start the Calibre DRC verification and check the result in RVE.

4. If the layout isn't DRC free, you have to re-edit the layout and re-run the DRC check to make it DRC free.



Files: buffer2.calibre.gds

■ Import layout database from layout viewer

Primary Cell: buffer2

Type in the name of the cell to run DRC on.

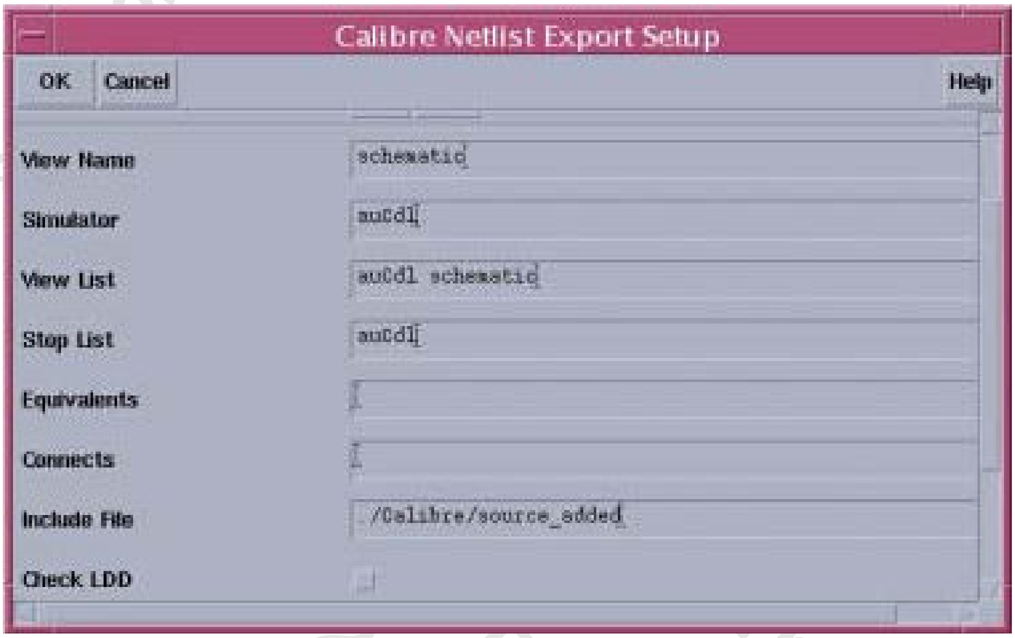
□ Check area:

FIG. 35 Calibre interactive DRC window

Calibre GUI LVS

After the layout has no DRC violations, the next step is to run the LVS check to make sure the layout is completely match to the schematic.

1. If the “source_added” file (empty sub-circuit file) is provided along with the LVS deck, you have to specify this file as an include file for netlist export by click “Calibre->Setup->Netlist Export…”(fig 36).



View Name: schematic

Simulator: auCdl

View List: auCdl schematic

Stop List: auCdl

Equivalents:

Connects:

Include File: /Calibre/source_added

Check LDD: □

FIG. 36 Calibre netlist export setup

2. Click “Calibre->Run LVS” in layout window to invoke Calibre LVS graphic user interface.
3. Specify the “Calibre-LVS rules file“, working directory and “Primary cell” in Calibre LVS window. If you need to change some LVS switches, you have to edit the Calibre LVS deck first.

4. Click “OK” to run the Calibre GUI LVS and see the result. If the layout isn’t matched to schematic, you have to fix the layout and re-run the LVS check to make the LVS result matched.

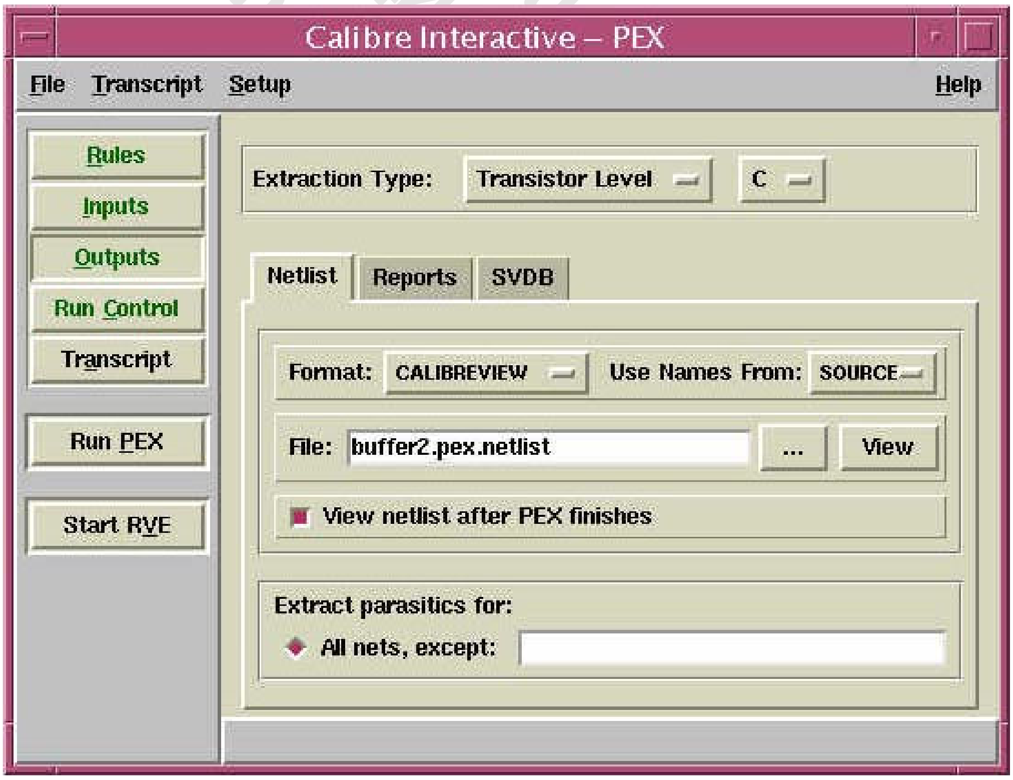
Calibre GUI RCX (with Calibre-view)

When the layout is verified to be DRC free and LVS clean, the next step is to perform the RC extraction. In addition to directly obtain the ascii extracted netlist file from Calibre GUI mode extraction (like Calibre CDL RCX), you can also choose to obtain an extracted view 'calibre-view' which is somewhat like the Assura av_extracted view. In this calibre-view, you can see not only the original components and but also the parasitic devices and their connectivity. Furthermore, this calibre-view can also be used in Cadence Analog Artist environment for post-layout simulation.

1. Click "Calibre->Run PEX" in layout window to invoke Calibre RCX graphic user interface.

2. Specify the "Calibre-PEX rules file", working directory and "Primary cell" in Calibre RCX window. If you need to change some RCX switches, you have to edit the Calibre RCX deck first.

3. Select the "Extraction Type", specify the output format to be "CALIBREVIEW" and "Use Names From" to be "SOURCE" (fig 37).



Calibre Interactive – PEX

Extraction Type: Transistor Level | C

Format: CALIBREVIEW

Use Names From: SOURCE

File: buffer2.pex.netlist

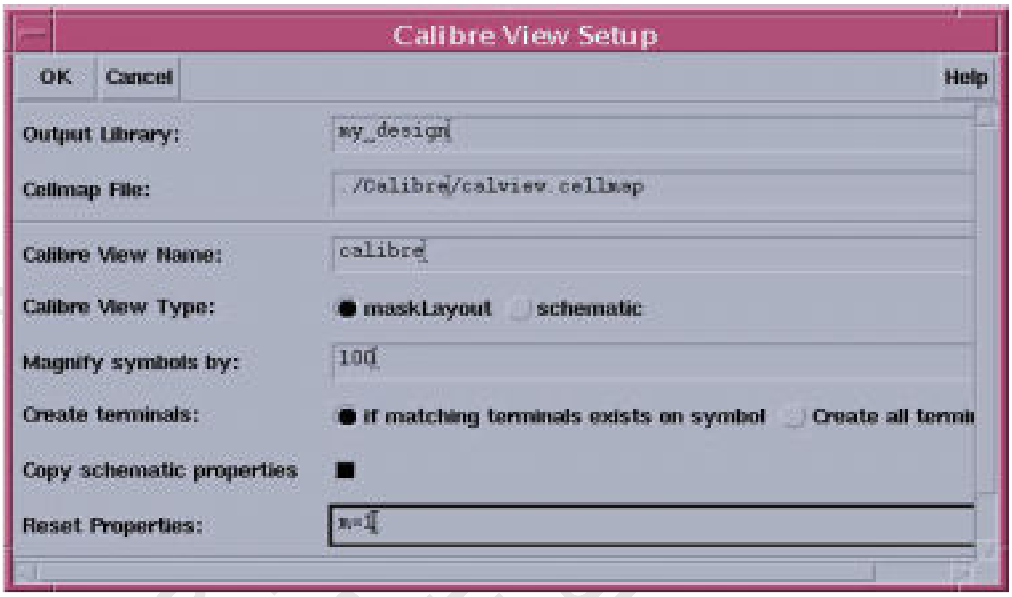
■ View netlist after PEX finishes

Extract parasitics for:

◆ All nets, except:

FIG. 37 Calibre interactive RCX window

4. Click “OK” to run the Calibre GUI RC extraction. After the extraction run is completed, a calibre view setup window pops up. Specify the “Cellmap File”, “Magnify symbols by” and make sure the “Copy schematic properties” button is on (fig 38).



Output Library: my_design

Cellmap File: ./Calibre/calview.cellmap

Calibre View Name: calibre

Calibre View Type: ● maskLayout ○ schematic

Magnify symbols by: 100

Create terminals: ● if matching terminals exists on symbol ○ Create all termi

Copy schematic properties ■

Reset Properties: m=1

FIG. 38 Calibre view setup window

5. Click “OK” in the Calibre view setup window to create the Calibre view. The generated Calibre-view can be seen in Fig 39.

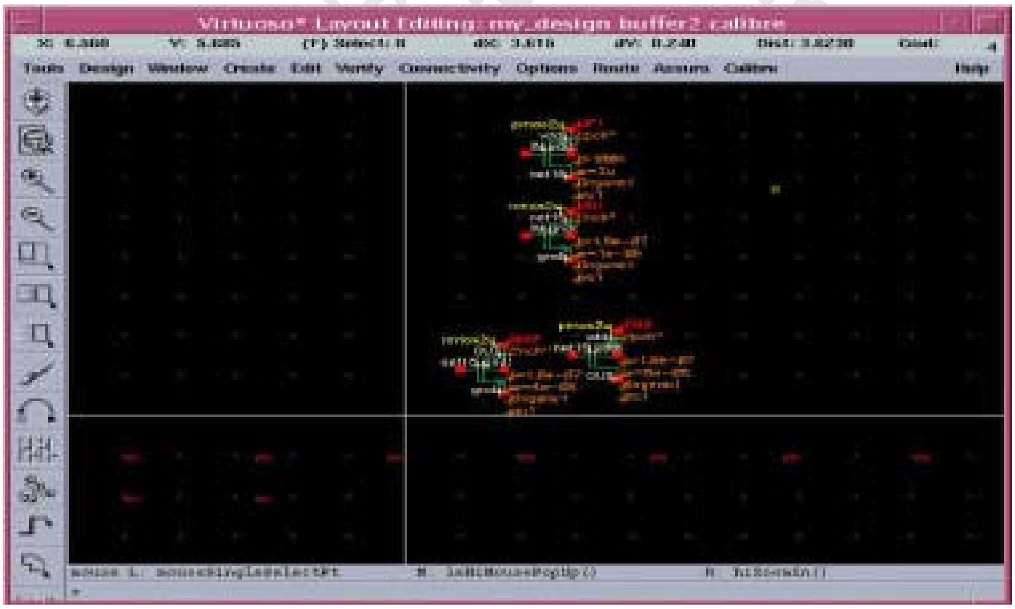

Virtuoso® Layout Editing: my_design buffer2 calibre
X: 6.360 Y: 5.485 (F) Select: 0 dX: 3.610 dY: 0.240 Dist: 3.6230 Cmd:
Tools Design Window Create Edit Verify Connectivity Options Route Assura Calibre Help
mouse L: mouseSingleSelectPt M: leHiMousePopUp() R: hiZoomIn()

FIG. 39 Calibre view result

Chapter 6 Post-layoutsimulation

When accomplished the physical verification, the last step to tape-out is to perform the post-layout simulation on the extracted netlist/view. During the post-layout simulation, not only the original components but also the parasitic R & C (depends on what you have extracted in RCX stage) of the interconnections are taking into consideration. Therefore, we can say that the post-layout simulation result is more approach to the real silicon comparing with the original pre-layout simulation result. Furthermore, base on the difference of RC extraction flows you chose you would run your post-layout simulation in different ways.

Simulating with the extracted netlist

If you choose to output an ASCII extracted netlist during the RC extraction phase (such as go through “Calibre CDL RCX” flow or select to export hspice/spectre netlist in “Calibre GUI RCX”/”Assura DFII RCX” flows), you may plan to run the post-layout simulation with batch mode in UNIX command prompt. Here, we use the extracted netlist, which is obtained from “Calibre CDL RCX” as an example, and run the post-layout simulation with Hspice simulator in UNIX command prompt.

1. Edit ASCII extracted netlist file to add model include section and other statements

Before starting the post-layout simulation, we have to add some information that is need for later simulation into the netlist. The extra lines that we added into the extracted netlist includes “model include section”, and “output control section” (the same as what we done in “Using Hspice for pre-layout simulation”).

2. Run Hspice simulation and check the simulation results (Fig40).

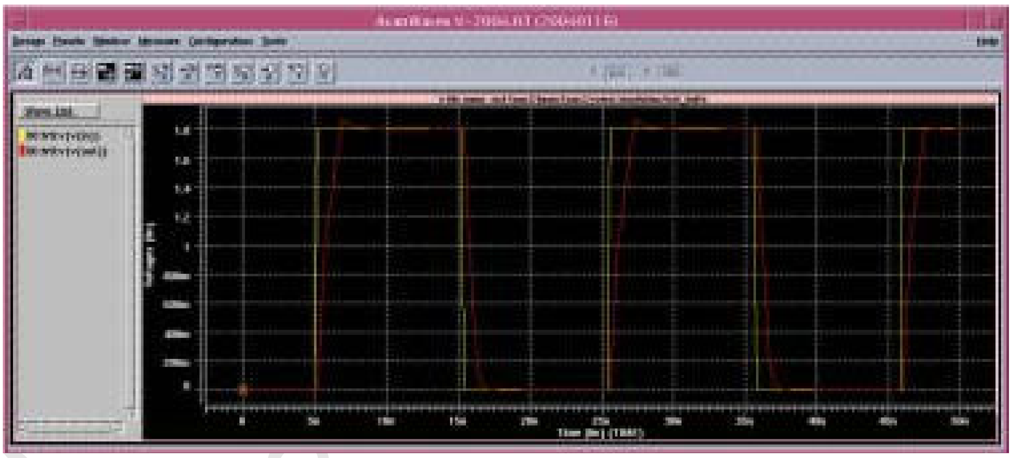


FIG. 40 Post-layout simulation result for extracted netlist

Simulating with the extracted view

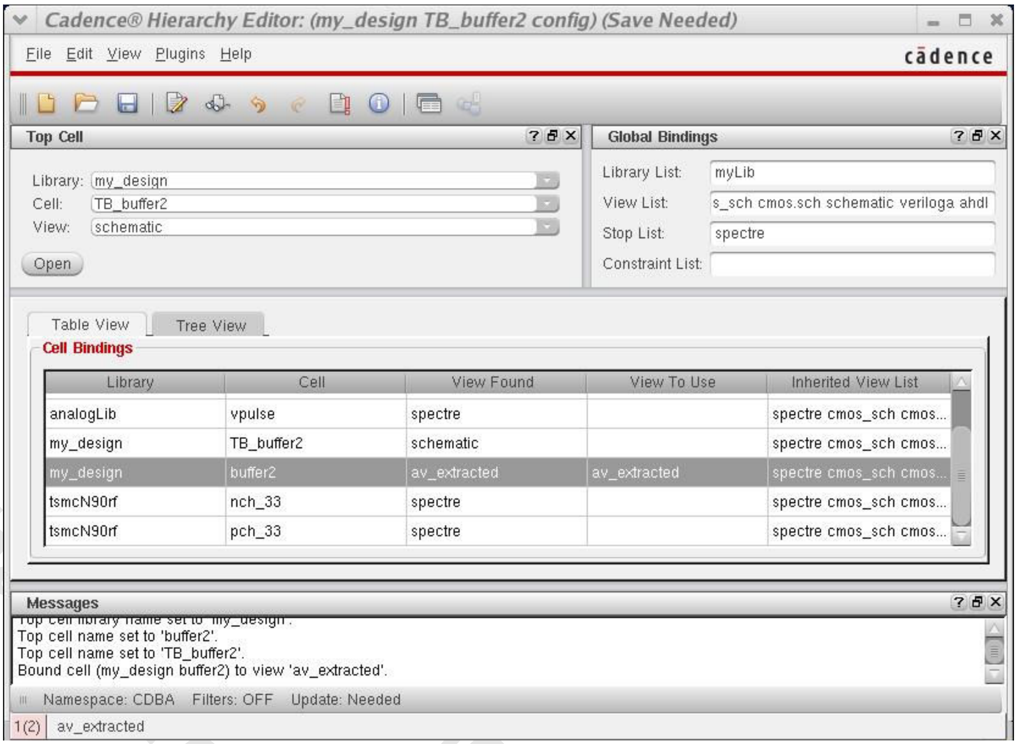
If you choose to output an extracted view during the RC extraction phase (such as go through “Calibre GUI RCX” flow or “Assura DFII RCX” flow), you may plan to run the post-layout simulation with GUI mode in Cadence Analog Artist environment. Here, we use the “av_extracted” view, which is obtained from “Assura DFII RCX” as an example, and run the post-layout simulation with Spectre simulator in Cadence analog Artist environment.

1. Create a ***config*** view for the test fixture schematic (the ‘test_buffer2’ in our example) and switch the ***av_extracted (or calibre)*** view of our design (buffer2) for the configuration (Fig 41).

Note: The detail on creating a *config* view in Cadence Virtuoso environment should be refered to Cadence Analog Design Environment manual.

2. Open the Analog Artist environment L and use the config view of the test fixture for simulation (fig 42). You can also output the simulation netlist to make sure the ‘av_extracted’ view of our design in the configuration file is used for simulation rather then the ‘schematic’ view.

3. Run the post-layout simulation and check the simulation result.


Cadence® Hierarchy Editor: (my_design TB_buffer2 config) (Save Needed)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open
Table View
Tree View
Messages

Library: my_design
Cell: TB_buffer2
View: schematic

Library List: myLib
View List: s_sch cmos.sch schematic veriloga ahdl
Stop List: spectre
Constraint List:

Cell Bindings

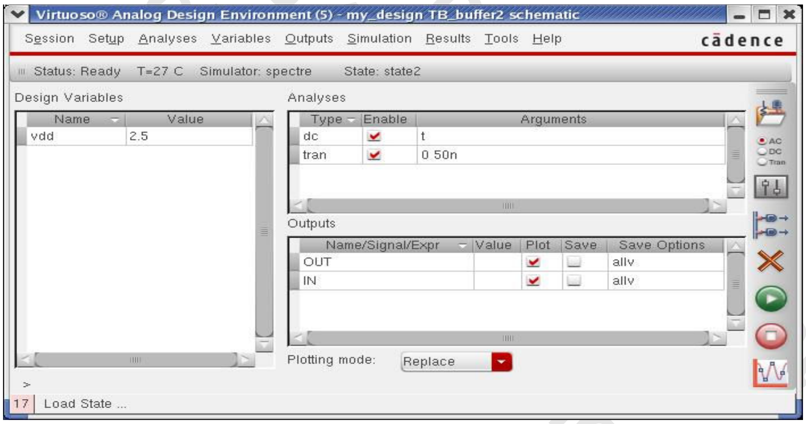
Library	Cell	View Found	View To Use	Inherited View List
analogLib	vpulse	spectre		spectre cmos_sch cmos...
my_design	TB_buffer2	schematic		spectre cmos_sch cmos...
my_design	buffer2	av_extracted	av_extracted	spectre cmos_sch cmos...
tsmcN90rf	nch_33	spectre		spectre cmos_sch cmos...
tsmcN90rf	pch_33	spectre		spectre cmos_sch cmos...

```
Top cell library name set to 'my_design'.
Top cell name set to 'buffer2'.
Top cell name set to 'TB_buffer2'.
Bound cell (my_design buffer2) to view 'av_extracted'.
```

Namespace: CDBA Filters: OFF Update: Needed

1(2) av_extracted

FIG. 41 switch design view to 'av_extracted' view


Virtuoso® Analog Design Environment (5) - my_design TB_buffer2 schematic
Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
AC
DC
Tran

Status: Ready T=27 C Simulator: spectre State: state2

Design Variables

Name	Value
vdd	2.5

Analyses

Type	Enable	Arguments
dc	✓	t
tran	✓	0 50n

Outputs

Name/Signal/Expr	Value	Plot	Save	Save Options
OUT		✓	□	allv
IN		✓	□	allv

Plotting mode: Replace

17 Load State ...

FIG. 42 Open Artist and use config view for post-sim

/home/hpai/simulation/buffer2/spectre/av_extracted/netlist/input.scs
File Help
cādence

```
// Library name: my_design
// Cell name: buffer2
// View name: av_extracted
c1 (vdd! 0) capacitor c=2.027e-15
c2 (in 0) capacitor c=1.325e-15
c3 (out 0) capacitor c=1.439e-15
c4 (net7 0) capacitor c=2.452e-15
M0 (net7 in vdd! vdd!) pch l=2.5e-07 w=2e-06 m=1 nf=1 sd=540n ad=0.96p \
        as=0.96p pd=4.96u ps=4.96u nrd=0.24 nrs=0.24 sa=4.8e-07 sb=4.8e-07 \
        sca=3.74002 scb=0.000795474 scc=4.42301e-06
M1 (out net7 vdd! vdd!) pch l=2.5e-07 w=2e-06 m=1 nf=1 sd=540n ad=0.96p \
        as=0.96p pd=4.96u ps=4.96u nrd=0.24 nrs=0.24 sa=4.8e-07 sb=4.8e-07 \
        sca=3.74002 scb=0.000795474 scc=4.42301e-06
M2 (net7 in 0 0) nch l=1.8e-07 w=2e-06 m=1 nf=1 sd=540n ad=0.96p as=0.96p \
        pd=4.96u ps=4.96u nrd=0.24 nrs=0.24 sa=4.8e-07 sb=4.8e-07 \
        sca=0.239679 scb=3.62102e-13 scc=4.3775e-25
M3 (out net7 0 0) nch l=1.8e-07 w=2e-06 m=1 nf=1 sd=540n ad=0.96p as=0.96p \
```

10 Open...

FIG. 43 Output the netlist to make sure a correct view is used for post-simulation