TSMC PDK usage guide:

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

Release 0.1

Feb 2008

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-driven-layout*
  - ✧ *Components placement*
  - ✧ *Auto-route and manual route*
- **Physical verification**
  - ✧ *Physical verification using Assura*
    - *A. Assura DFII flow*
    - *B. Assura CDL 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    "Analog"
```

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" and "lib.defs" files 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.lib
```

And use a text editor to add the following line into the “lib.defs ” file located in you project directory:

INCLUDE <pdk_install_directory>/lib.defs

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 achieved 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 CRN65LP PDK (fig1~2).

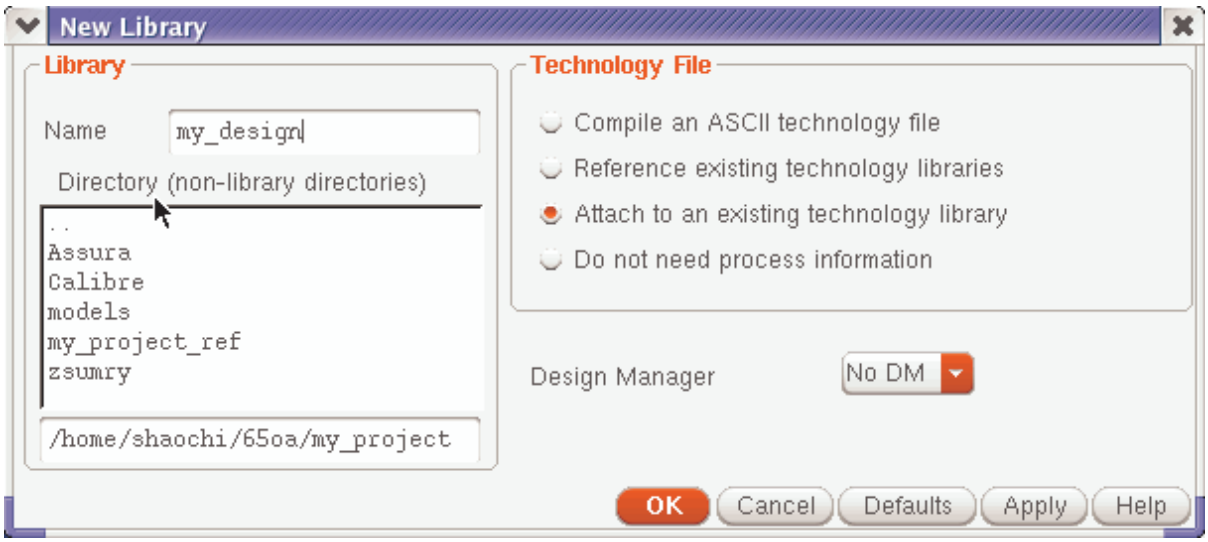


FIG. 1


Attach Library to Technology Library
New Library
my_design
Technology Library
analogLib
avTech
basic
cdsDefTechLib
sample1
tsmcN65
OK
Cancel
Apply
Help

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 CRN65LP PDK, to continue the following introductions in this document.

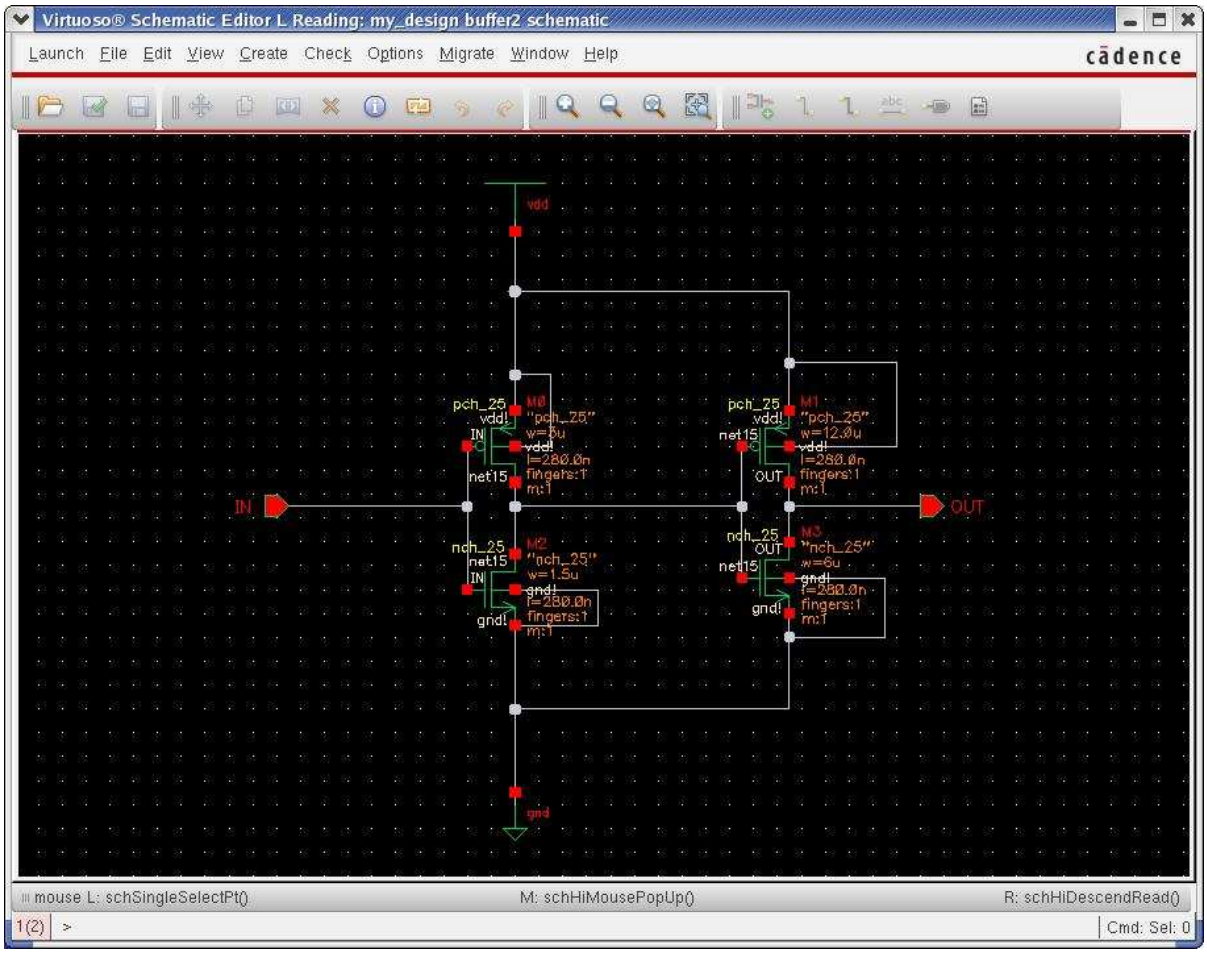


FIG. 3 schematic-capture of a buffer

Creating a symbol

After completing the creation of schematic-capture, we have to create its corresponding symbol for the subsequence simulation steps. On schematic view, execute **Create –> Cellviw –> From Cellview** to open the form then click OK to generate the corresponding symbol view.

FIG. 4

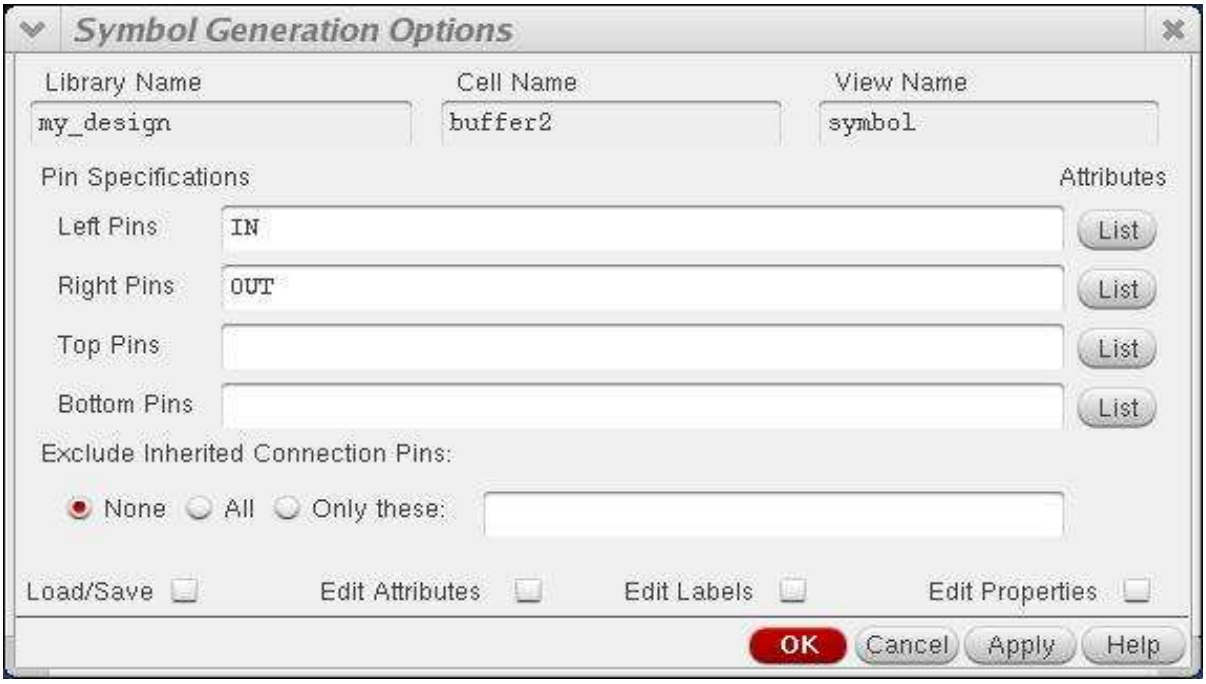


FIG. 5

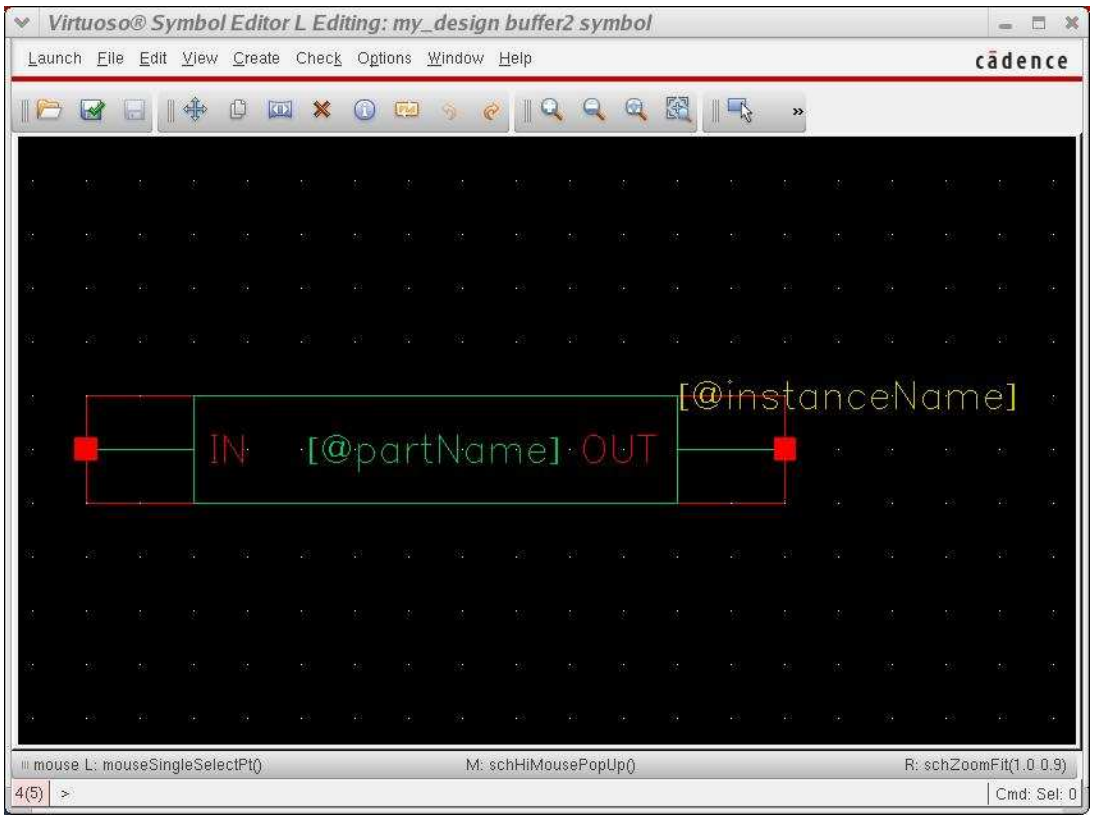


FIG. 6 symbol for the buffer

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, period is 20ns here), and output loading.

The test fixture that we used for our design is named ‘TB_buffer2’ and shown below.

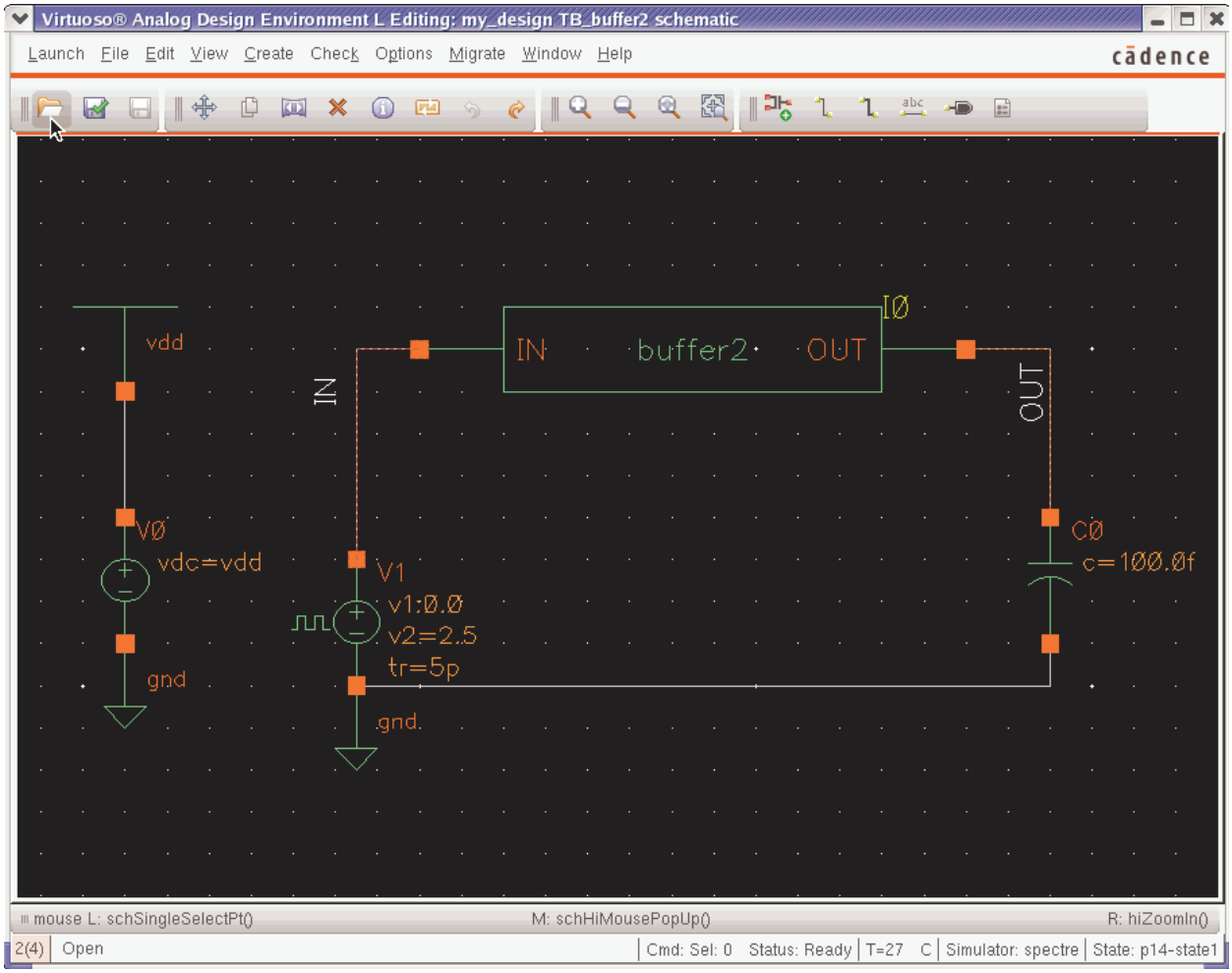


FIG. 7 text fixture schematic for the buffer

Chapter 3 Pre-layout 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 in ADE-L (Analog Design Environment L) . The simulation steps for spectre is as follows:

1. *Open Schematics L of testbench TB_buffer2 and launch ADE-L*

   This can be achieved by “Launch -> ADE L” in the menu banner of schematic view (FIG8).

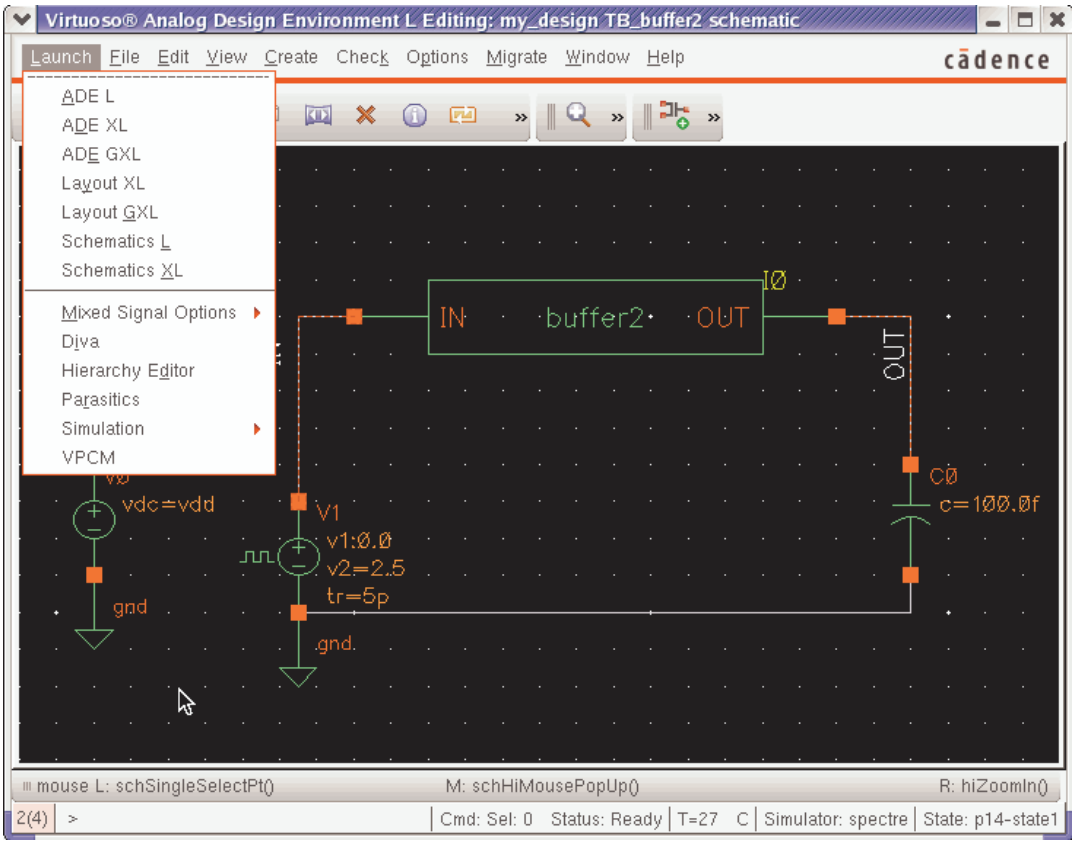


FIG. 8 Open ADE_L (Analog Design Environment L)

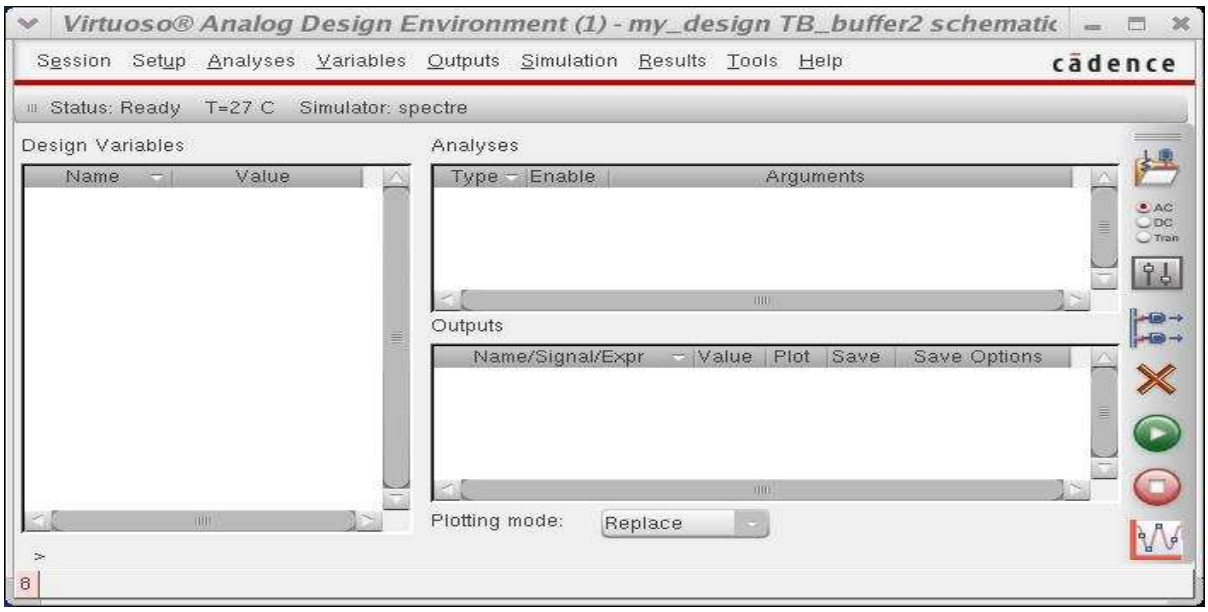


FIG. 9 ADE_L (Analog Design Environment L form)

2. Select analysis type and fill in parameters for simulation

In ADE-L (Analog Design Environment L) form, there are several analysis options to be chosen. Since we want to analysis the delay information (performance) of our design, we choose the transient analysis for our design. Some designers may want to see the OP point and they can also include the DC op point analysis.

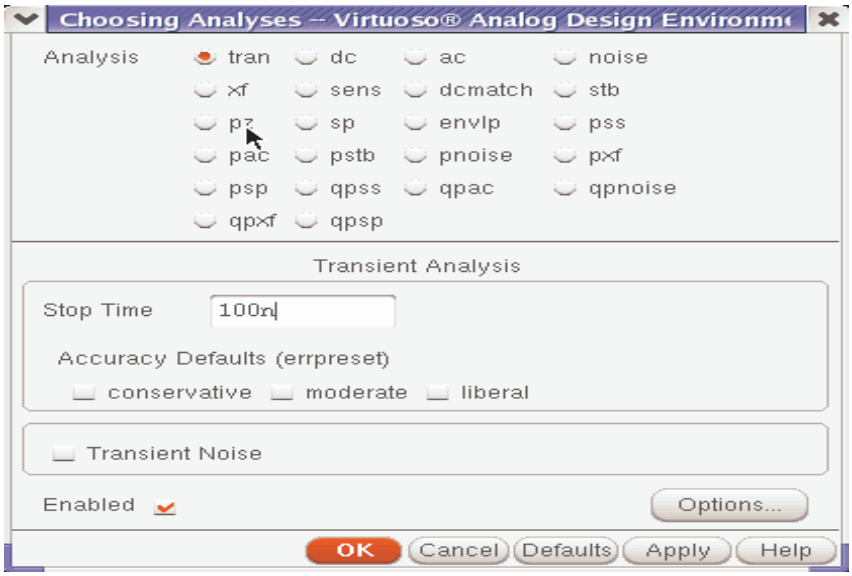


FIG. 10 choosing analysis type and filling in parameters

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

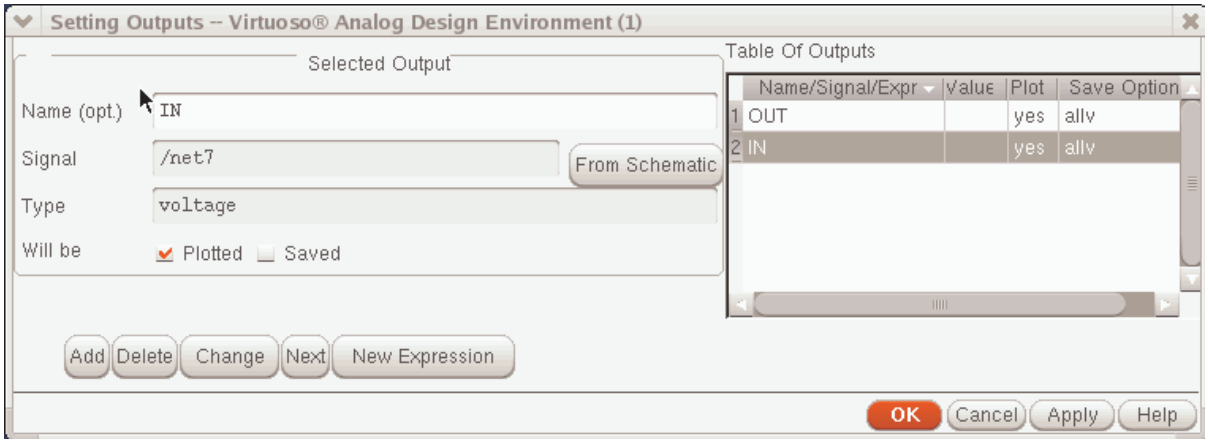


FIG. 11 select nodes

To set up the DC power supply to 2.5v, click on **Variables→ Copy From Cellview** and then fill in 2.5 in the Value field.

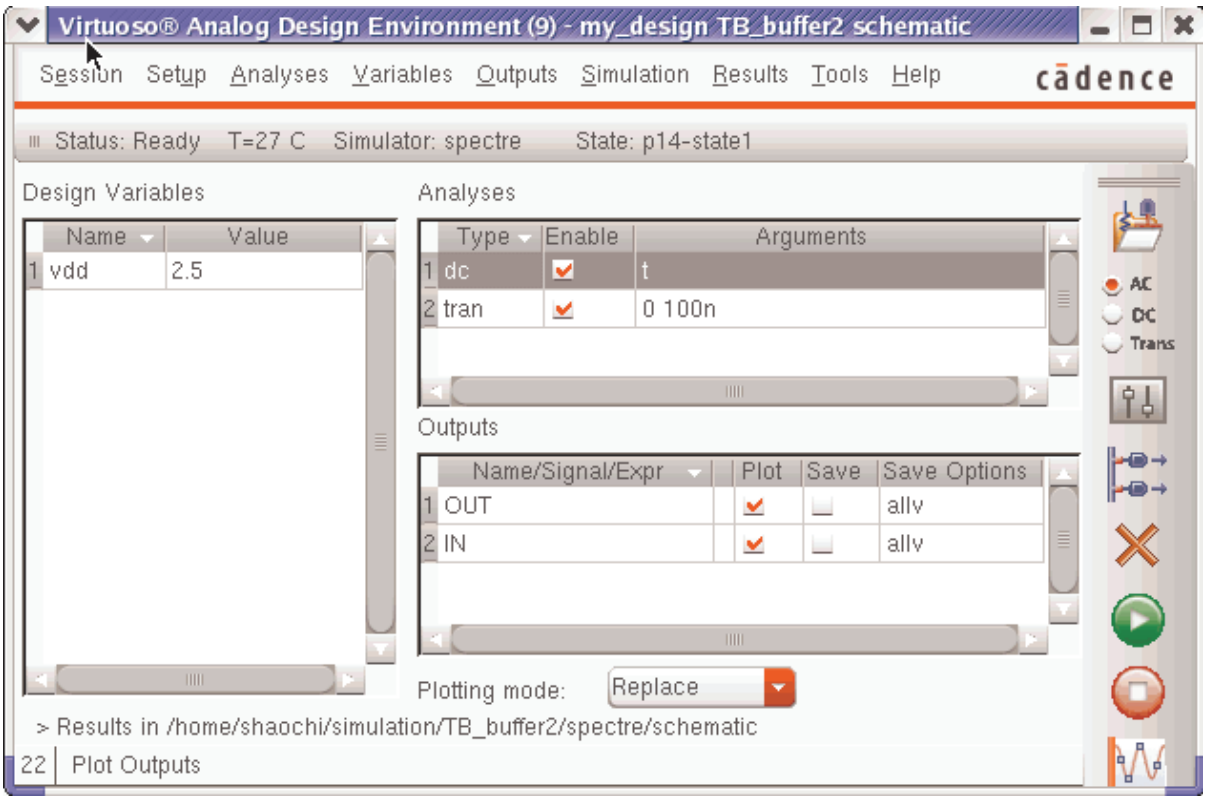


FIG. 12 selects browsing nodes and specifies supply voltage

3. Run simulation and observe the simulation results

To start the simulation, you can click “simulation->netlist and run” from ADE-L (Analog Design Environment L) toolbar or simply click the green (Netlist and Run) button. After the simulation is completed, the wave form window pops up automatically which presents all the simulation expression and measures defined in Outputs pane. In this case, the waveform of nodes “IN” and “OUT” of our design circuit block (buffer2) are shown below. (FIG13)

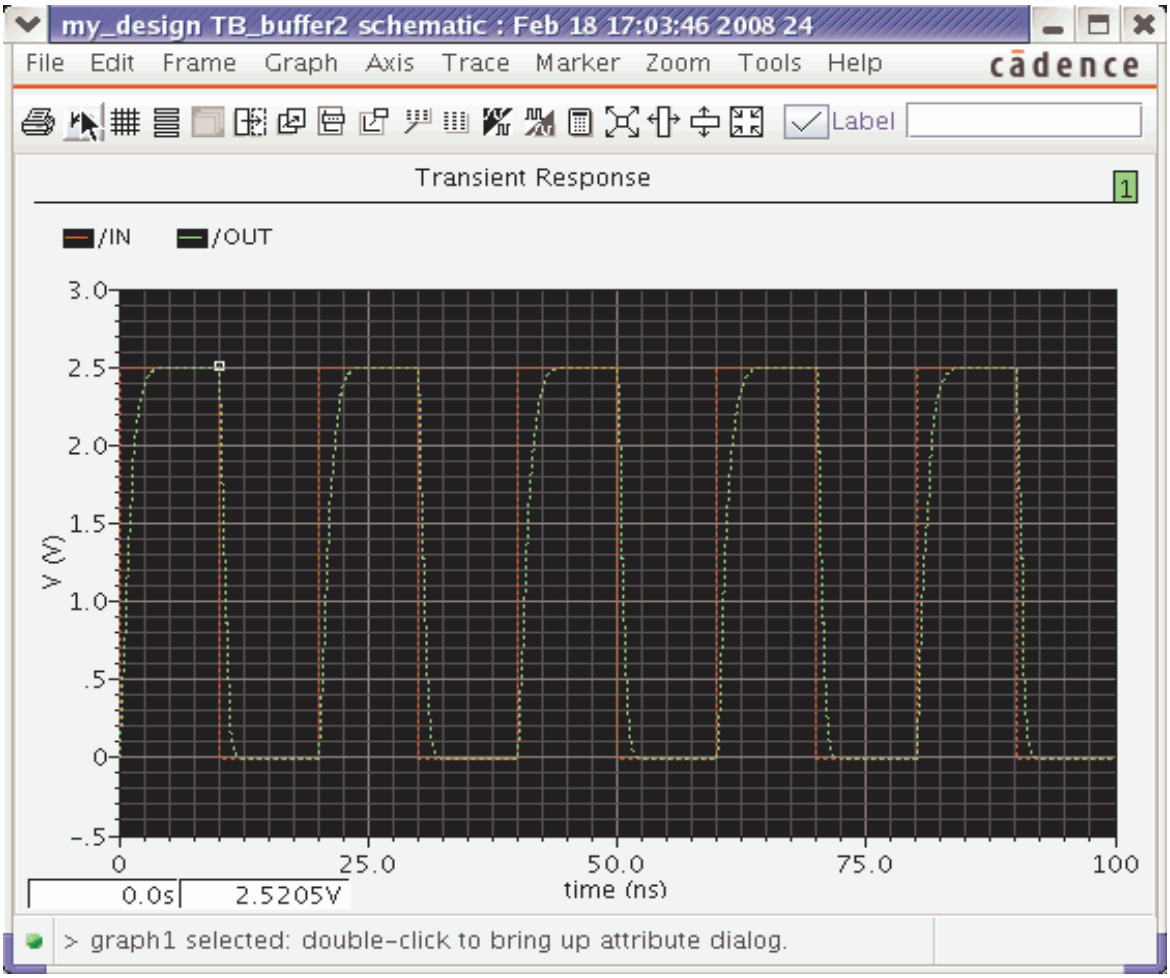


FIG. 13 simulation waveform

According to the waveform in Fig 13, we found that the functionality of this buffer is correct and the performance of this buffer is as below (with 100fF load):

Delay time: Td (rise) = 0.89 ns; Td (fall) = 0.55 ns

Rising/Falling time: Tr = 0.9 ns; Tf = 0.43 ns

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.

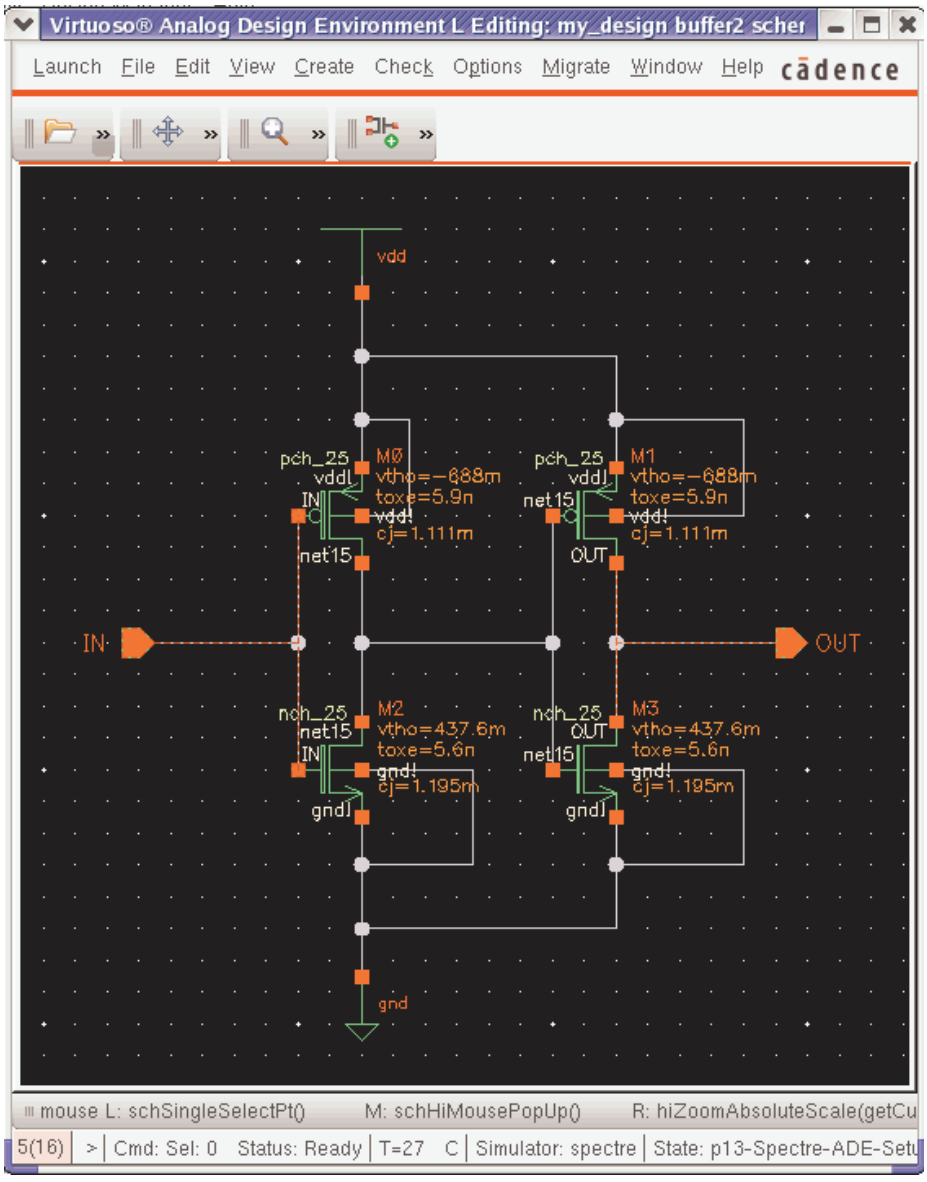


FIG. 14 back annotate model parameter data

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.

In the " Library manager" , open TB_buffer2 schematic view. After schematic view is opened , " Launch -> ADE-XL (Analog Design Environment XL) " , The ADE_XL (Analog Design Environment XL) GUI will appear like below.

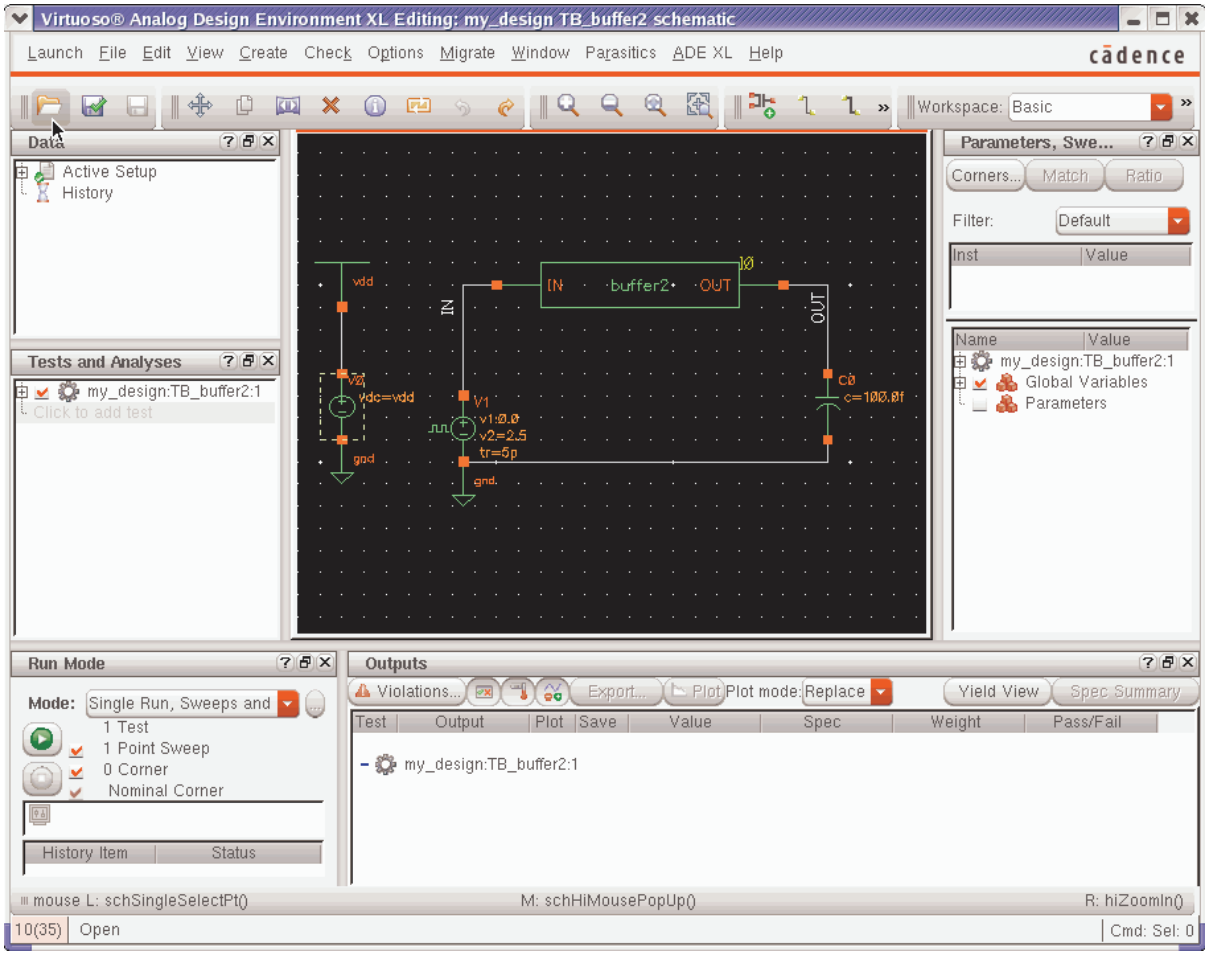
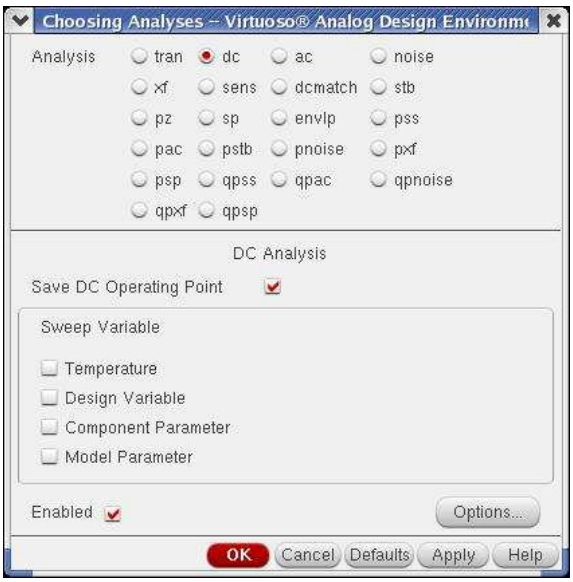


FIG. 15 Corner Analyses workspace

In the left panel “Tests and Analyses”, double click: my_design:TB_buffer2:1” and select **Analyses -> Choose** .We will choose “dc” and “tran” analyses.



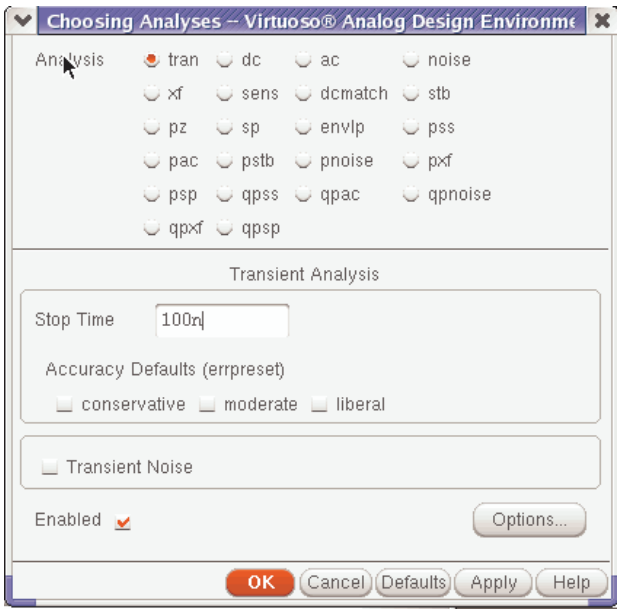


FIG. 16 Setting Analysis

In the right panel, “Parameters, Sweeps, and Corners Setup”, assign 2.5 to variable “vdd” by selecting **Variables > Edit**.

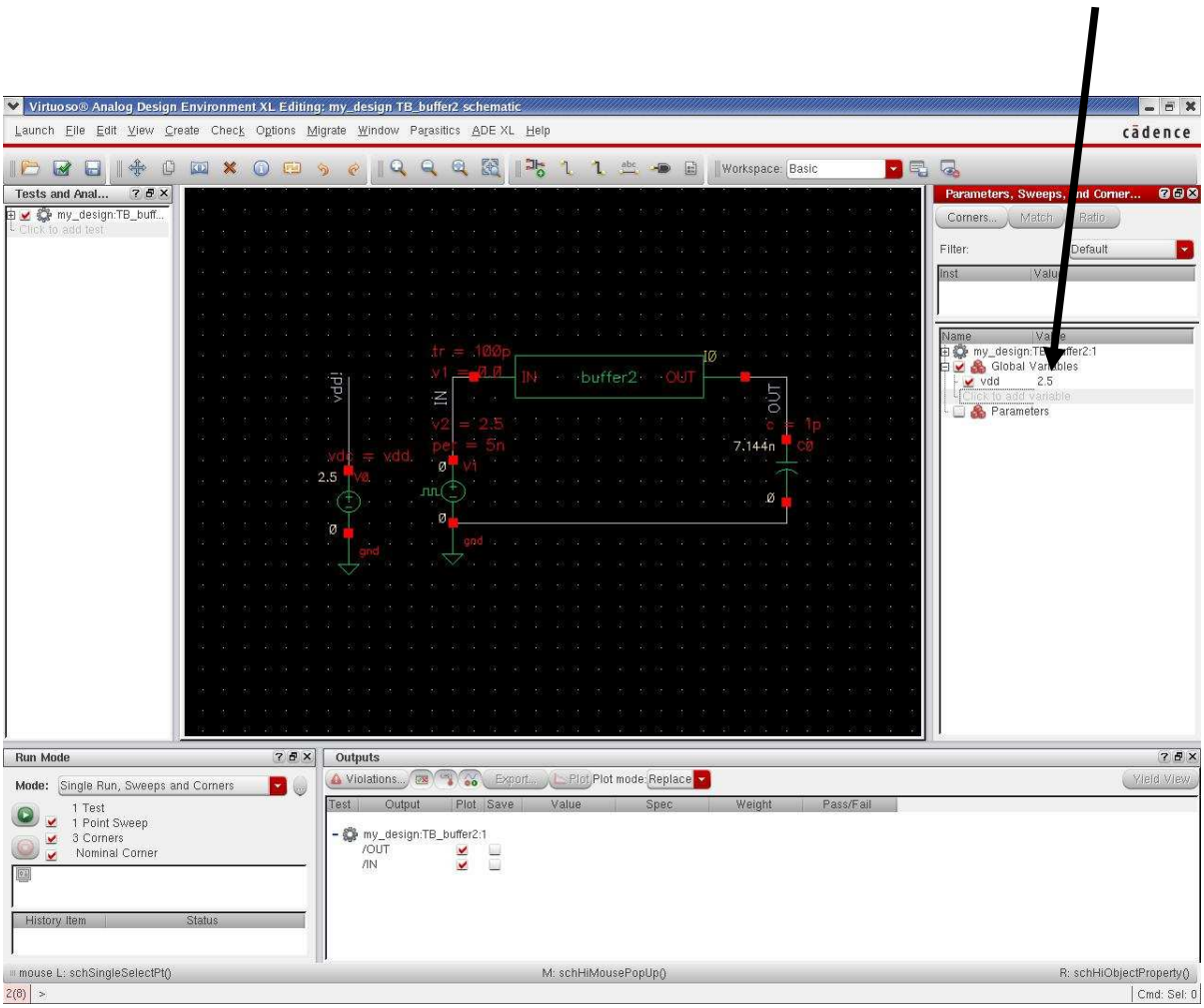


FIG. 17 setting variable for corner analysis

Then click “Corners” at the right panel. And “Corners Setup” form shows like below.

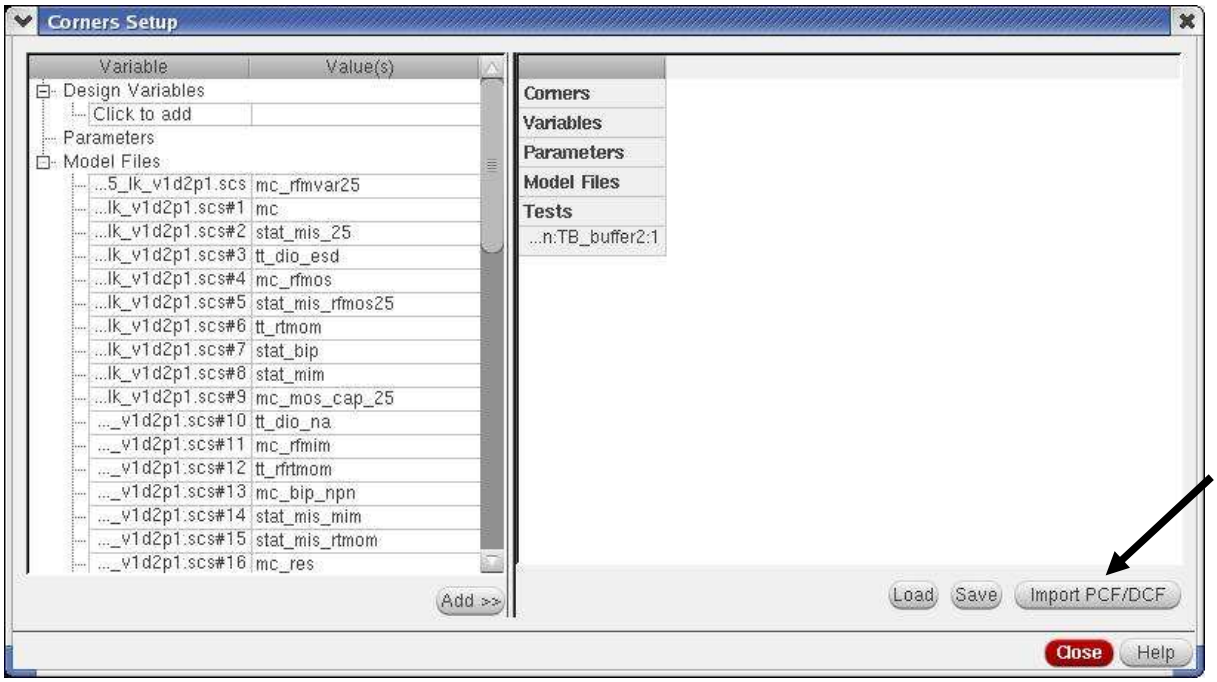


FIG. 18 Loading PCF file

Click “Import PCF/DCF” button , the file browser will pop up and please decent into directory <PDK_installDir>/models/spectre to open the sample PCF file ***tsmcN65.pcf*** then the Corner setup form will be defined accordingly.

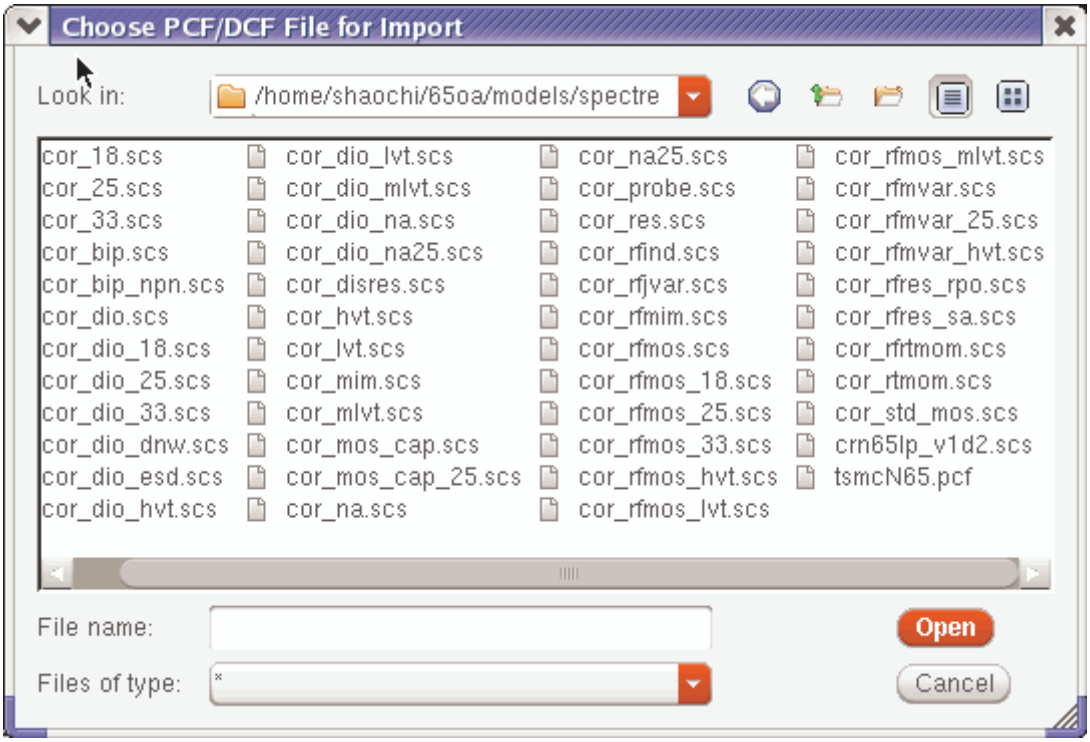


FIG. 19 Choosing PCF file

The final “Corners Setup” form looks like below after we import “tsmcN90rf.pcf”

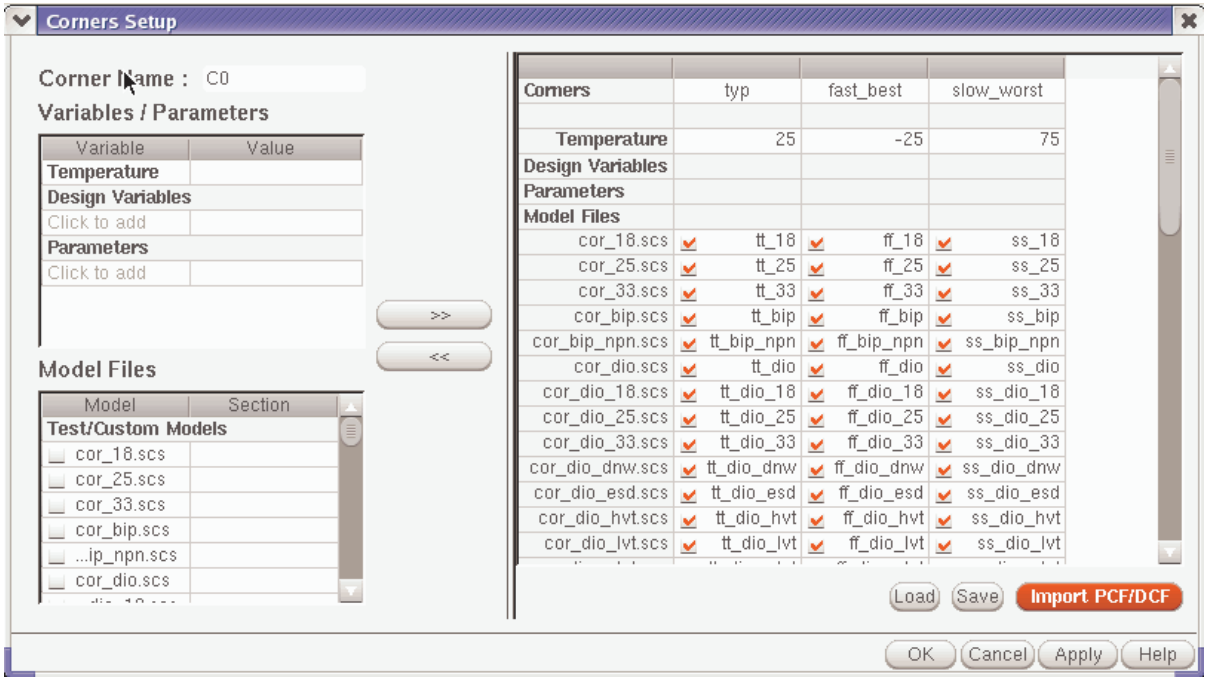


FIG. 20 Loading PCF file for corner analysis

In the bottom panel, “Outputs”, double click my_design:TB_buffer2:1 and set up which outputs we want to be saved and plotted.

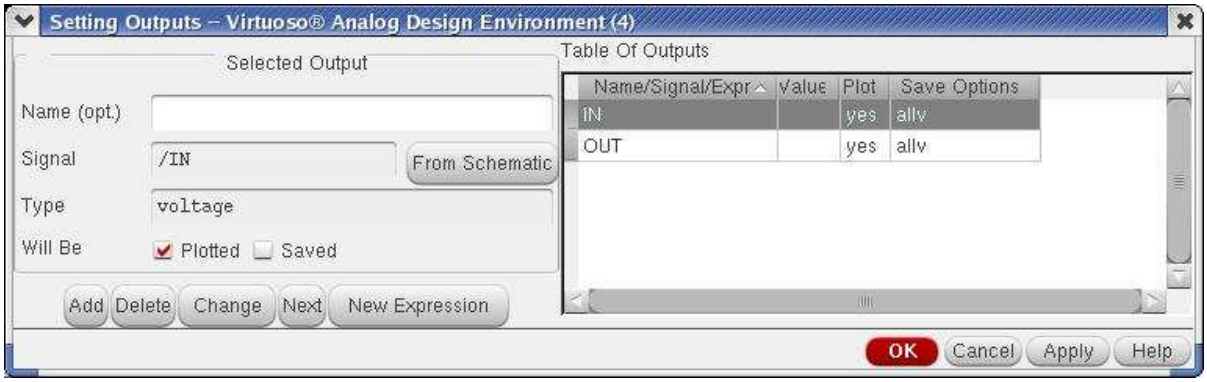


FIG. 21 setting output for corner analysis

In Run Mode pane, select *Single Run, Sweeps and Corners,* it shows 3 corners plus the nominal corner are checked to be available for simulation. Click the green button to start the corner simulation.

While the corner simulation was finished, the below output corner plot pops up and presents the data variation between different corners.

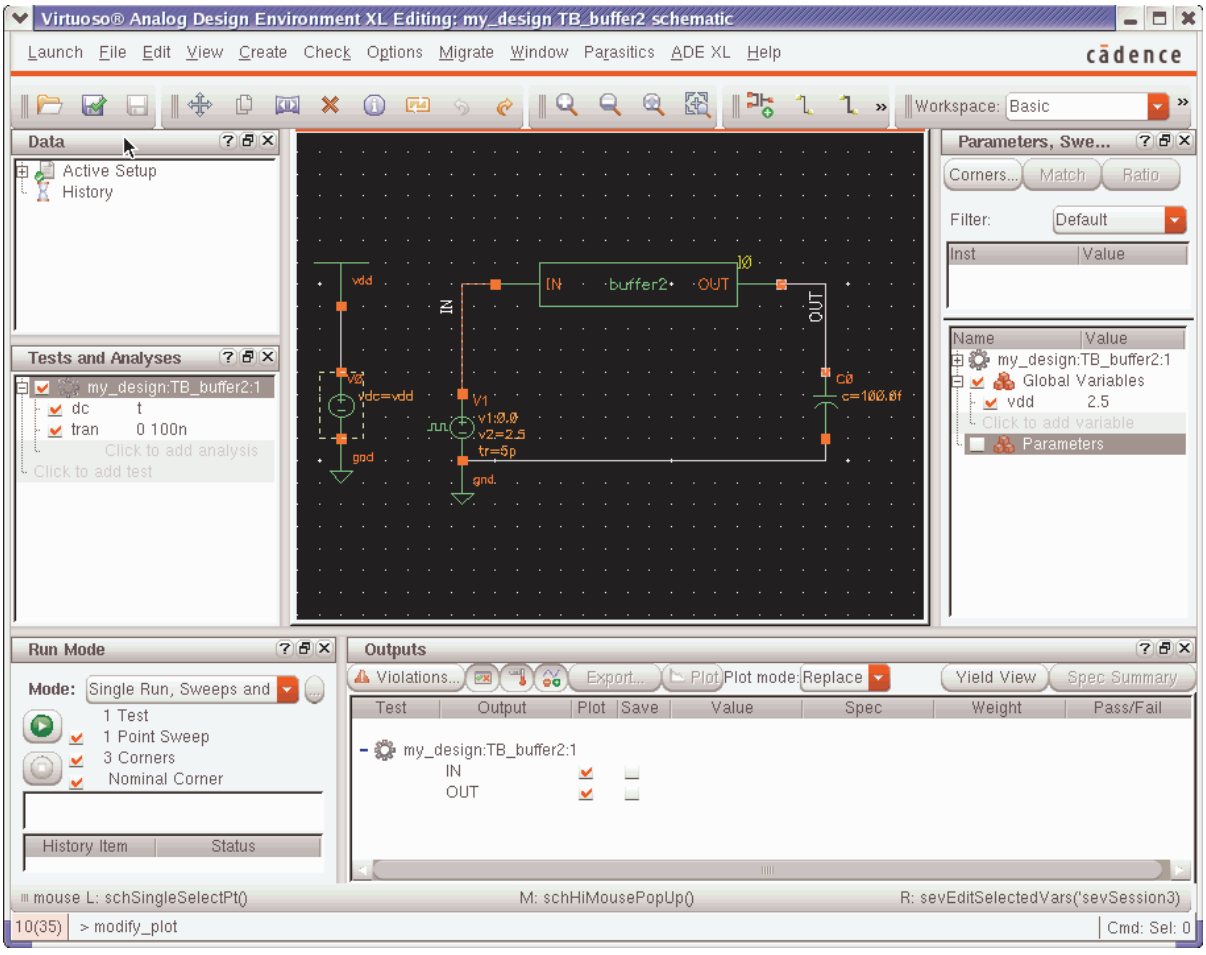


FIG. 22 Corner analysis setup form

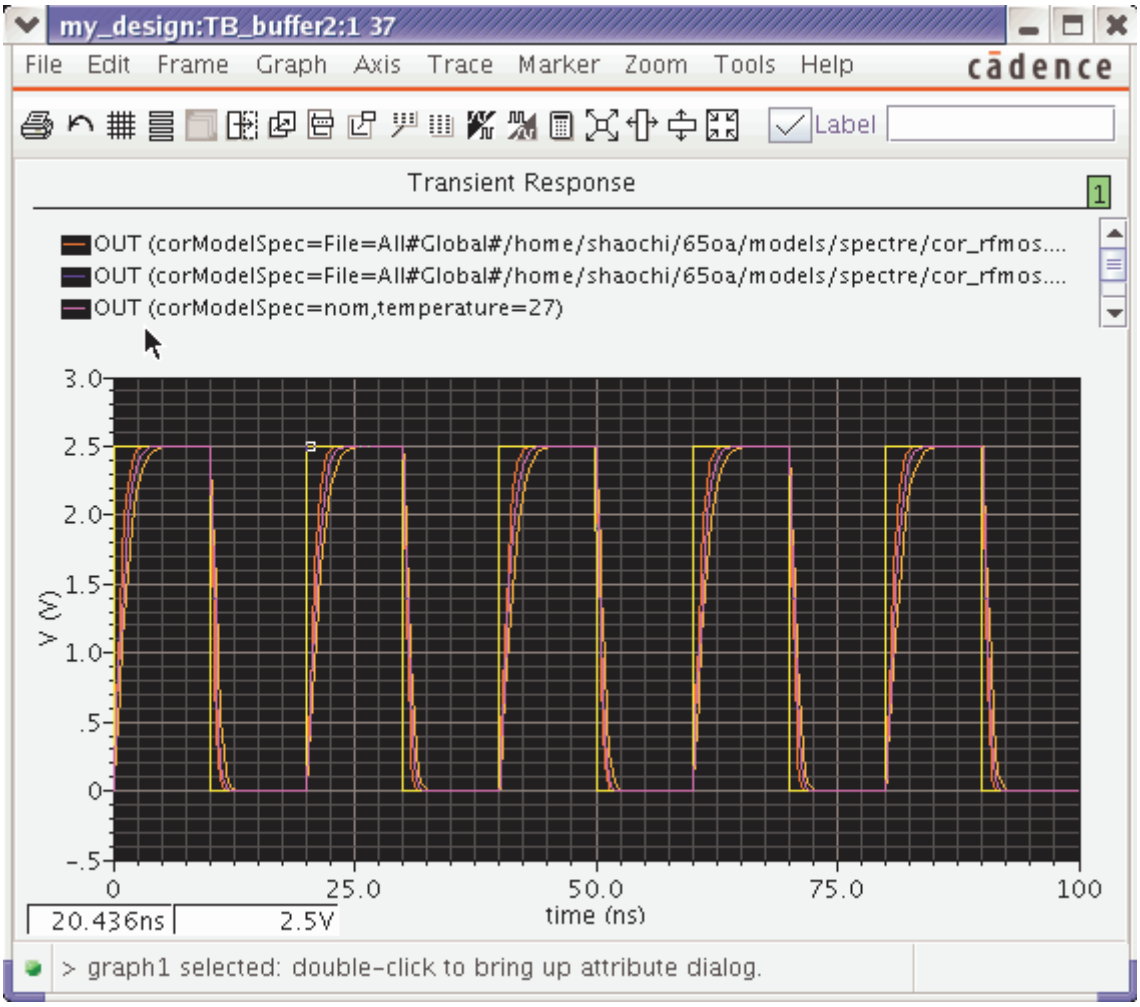


FIG. 23 simulation waveform for corner analysis

Using hspice for simulation

If you are not comfortable in using spectre for simulation, you can also choose other simulators (such as Hspice, UltraSim, or Ads … 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 ADE L (Analog Design Environment L)

The first step to run the Hspice simulation is to obtain a hspice netlist. This can be achieved by "setup->Simulator/Directory/Host…" and set the simulator to "hsipceD" (Fig 24).

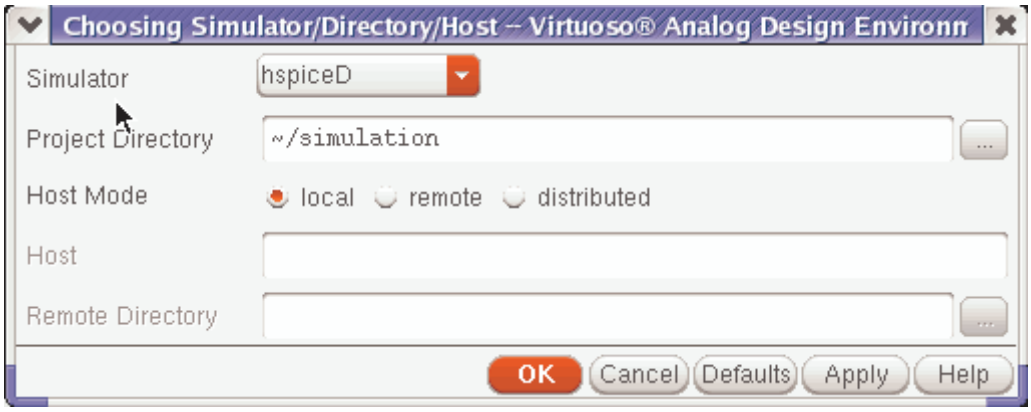


FIG. 24 Choose "hspiceD" as a simulator

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 steps for Spectre simulation).

After the previous works are all done, we can start to write out the hspice simulation netlist from ADE L by click "Simulation->Netlist->Create " in ADE L (Analog Design Environment L) window (Fig 25). 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".

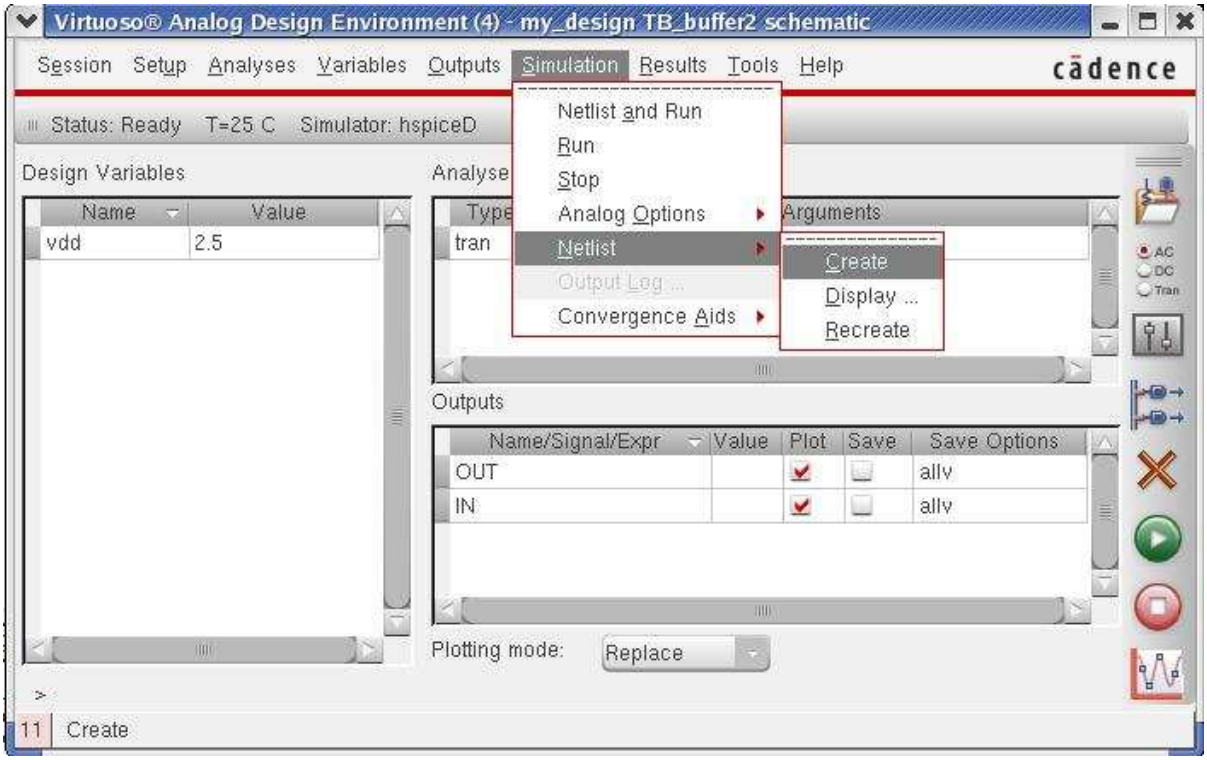


FIG. 25 Write out hspice simulation netlist

2. Add model include section and other control statements

After obtained the Hspice simulation netlist, we still need to add the information that is needed for later simulation into the netlist. The extra lines that we added into our netlist includes model include section, and output control section (fig 26).

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 27) after the simulation completed.

```
* INCLUDE FILES
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" probe
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_bip_npn
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfres_sa
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmvar
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_esd
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rtmom
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmvar_25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_disres
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_na
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_mim
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfrtmom
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_hvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfres_rpo
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmim
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_res
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_lvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_hvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_mlvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_bip
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_mos_cap
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_18
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_na
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_hvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfind
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_33
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_lvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_na25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_dnw
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_mlvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfjvar
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmvar_hvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_na25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_18
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_mos_cap_25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_33
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_33
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_mlvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_25
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_rfmos_lvt
.lib "tsmc_pdk_full_path/models/hspice/crn65lp_v1d2.l" tt_dio_18
.PLOT V(IN) V(OUT)
```

FIG. 26 add model-include and out control statement to hspice netlist

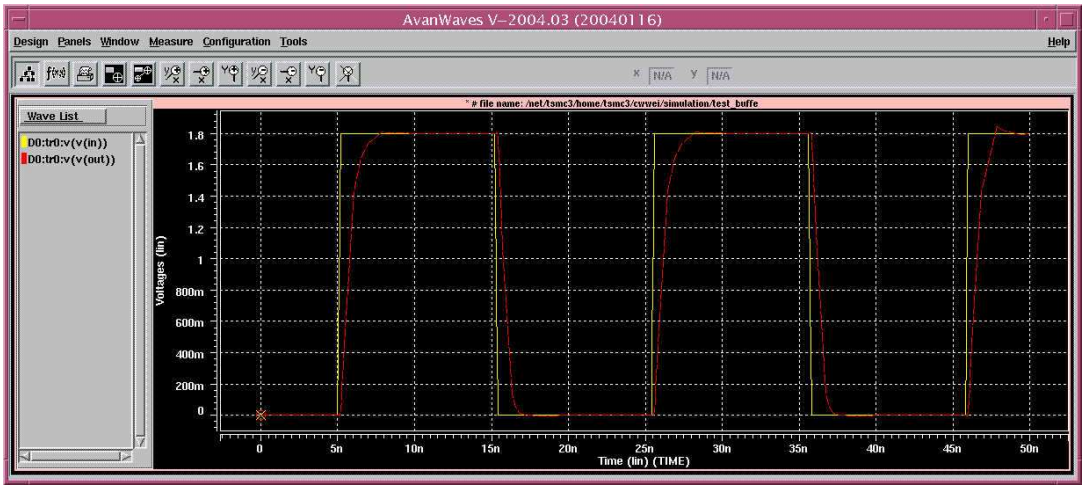


FIG. 27 Hspice simulation results

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.

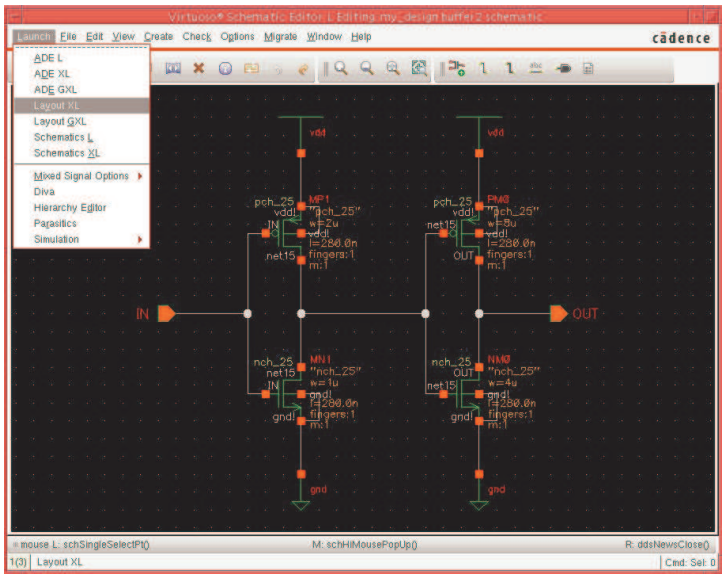


FIG. 28 invoke Virtuoso XL for schematic-driven-layout

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”.

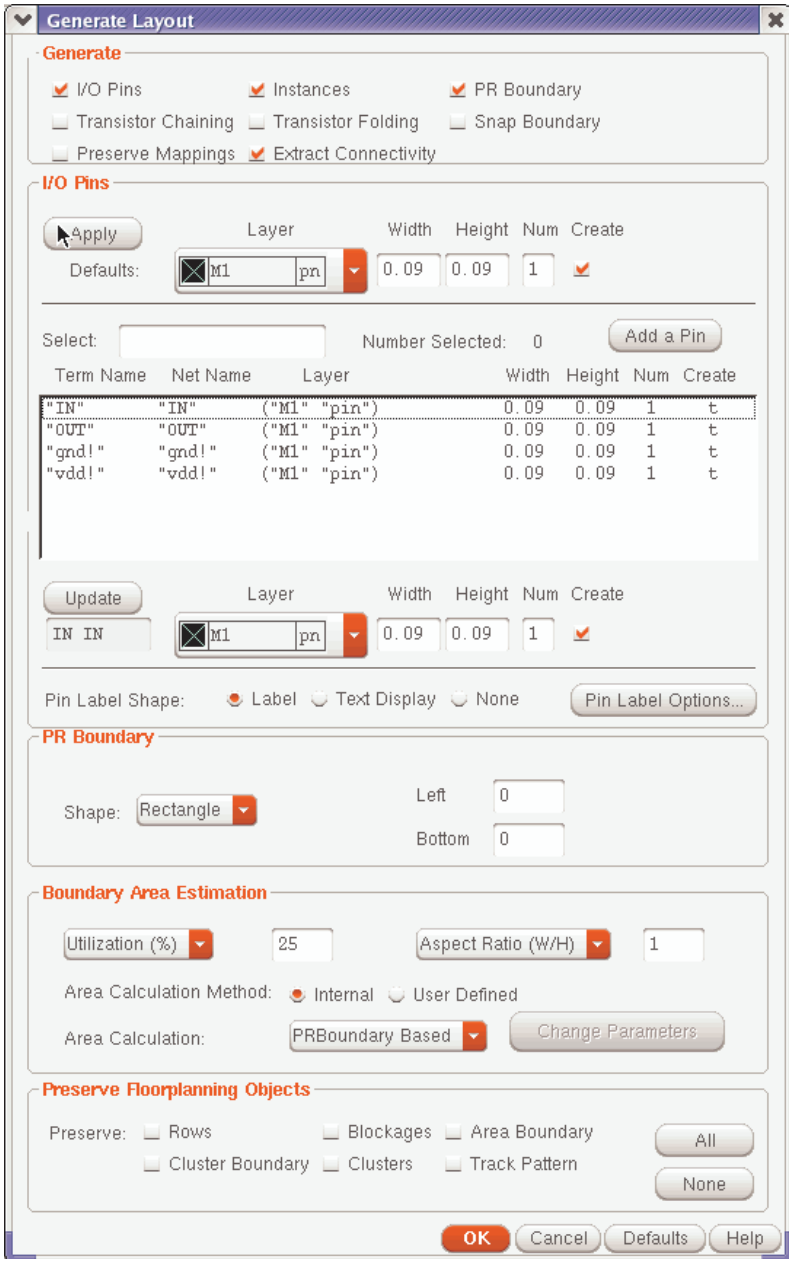


FIG. 29 layout generation options window

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.

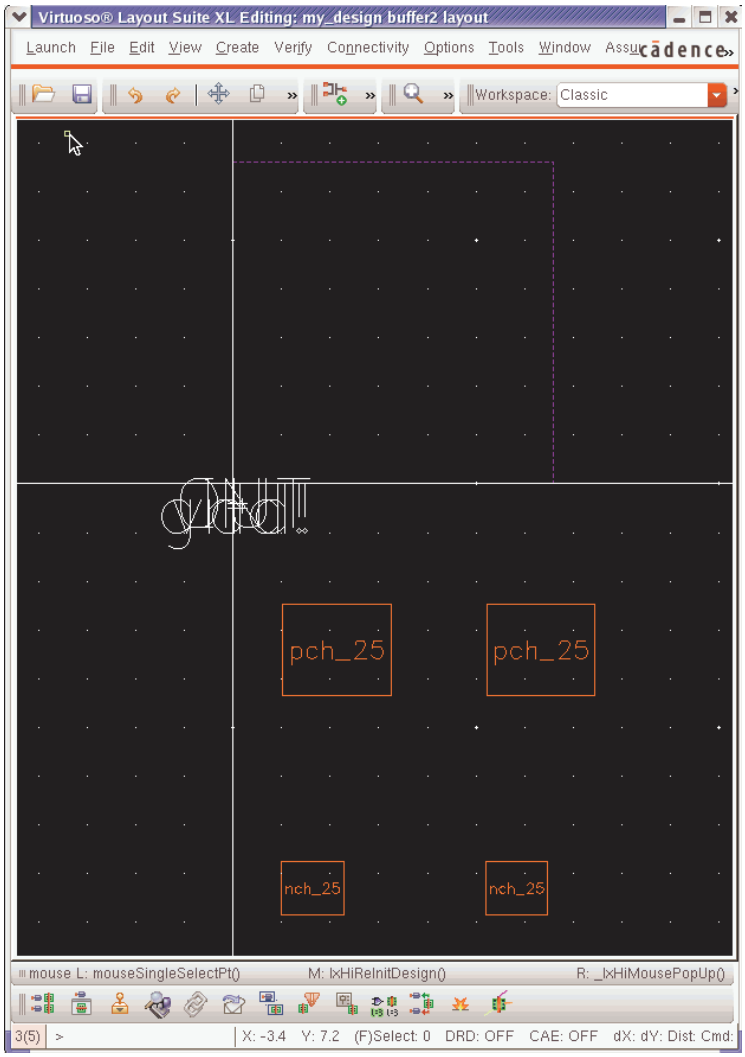


FIG. 30 I/O pins and devices generated in layout window

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 31).

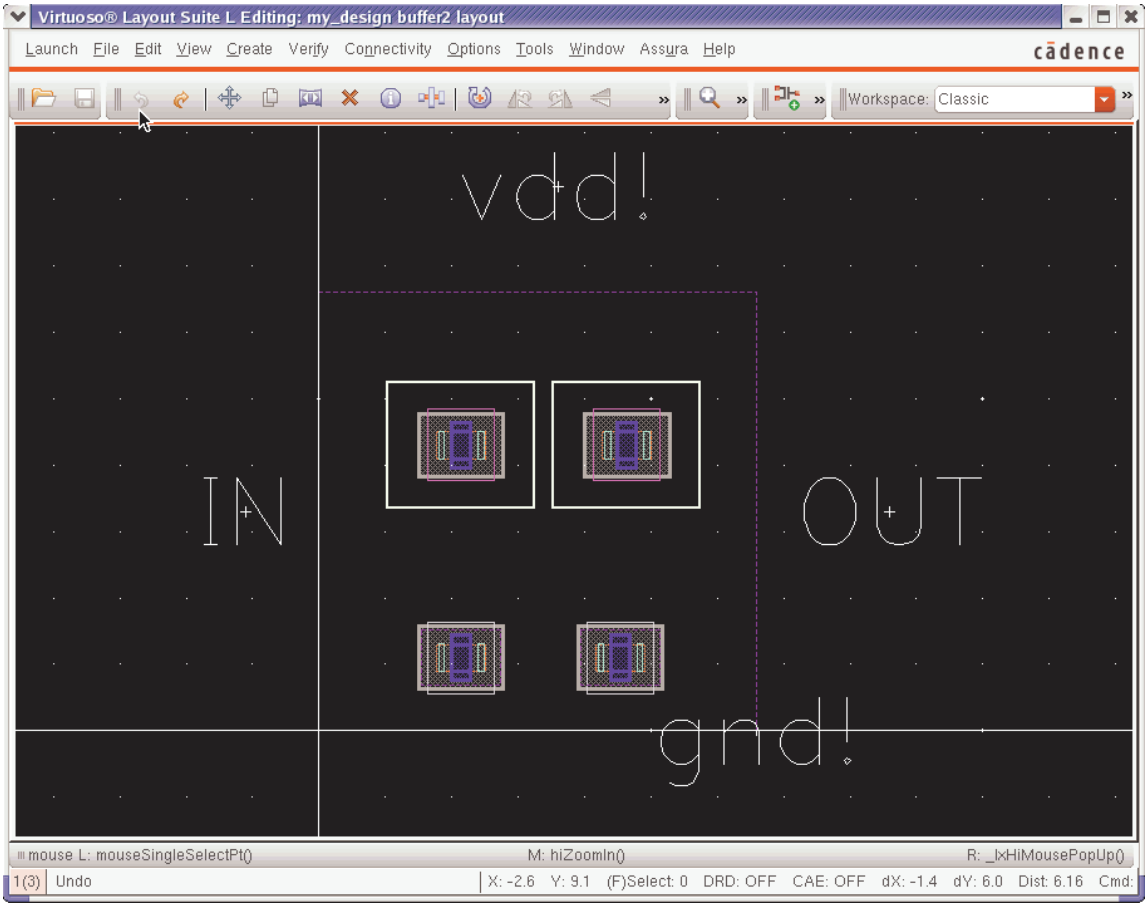


FIG. 31 Place all devices into design area in layout window

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. As Fig32 shows.

2. Specify the routing view name in the popup window to initiate the VCAR (Fig 33).
3. Click "Autoroute->Detail route->Detail router" to complete the auto routing (Fig 34).

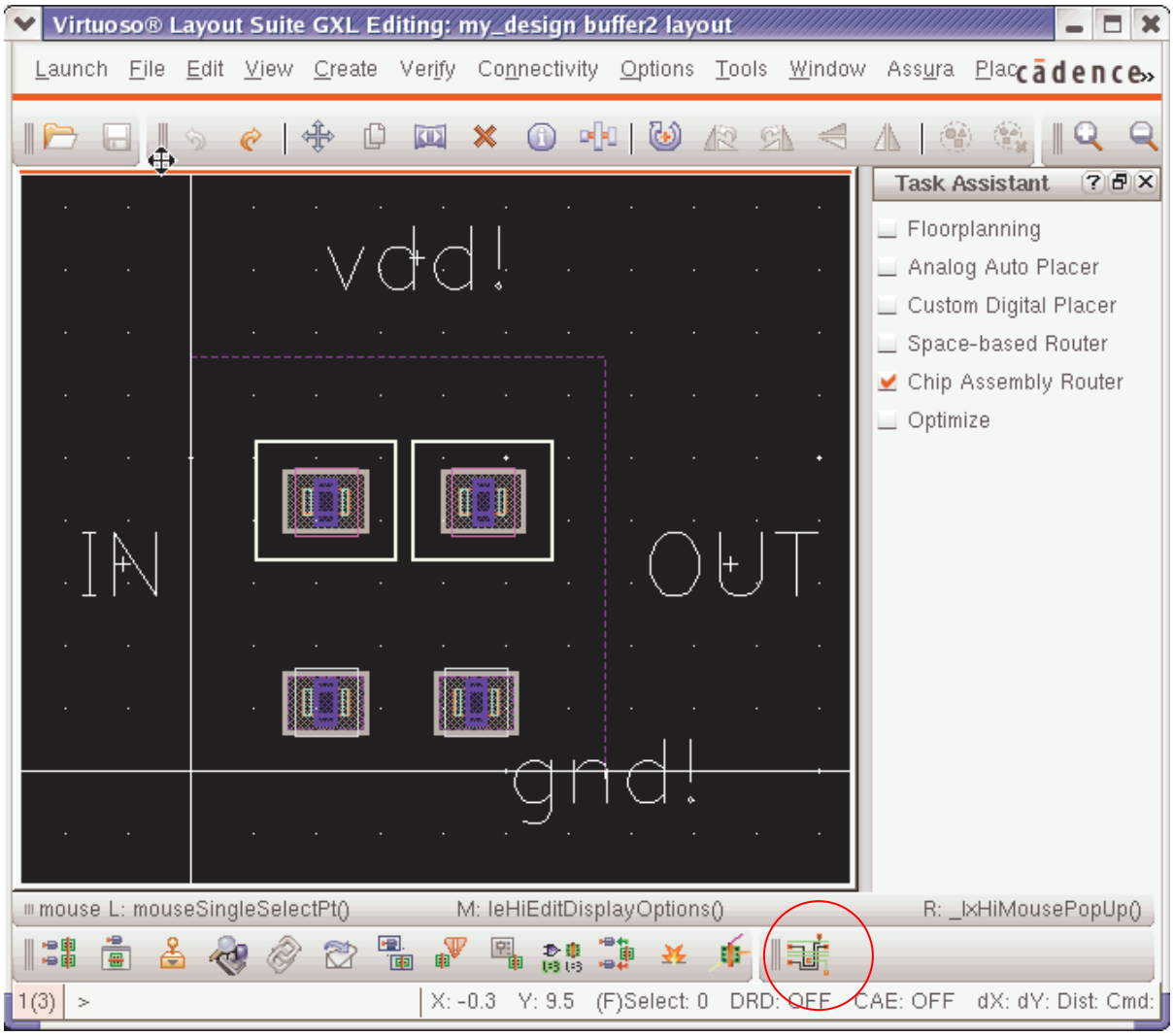


FIG. 32 Enable VCAR and Start router

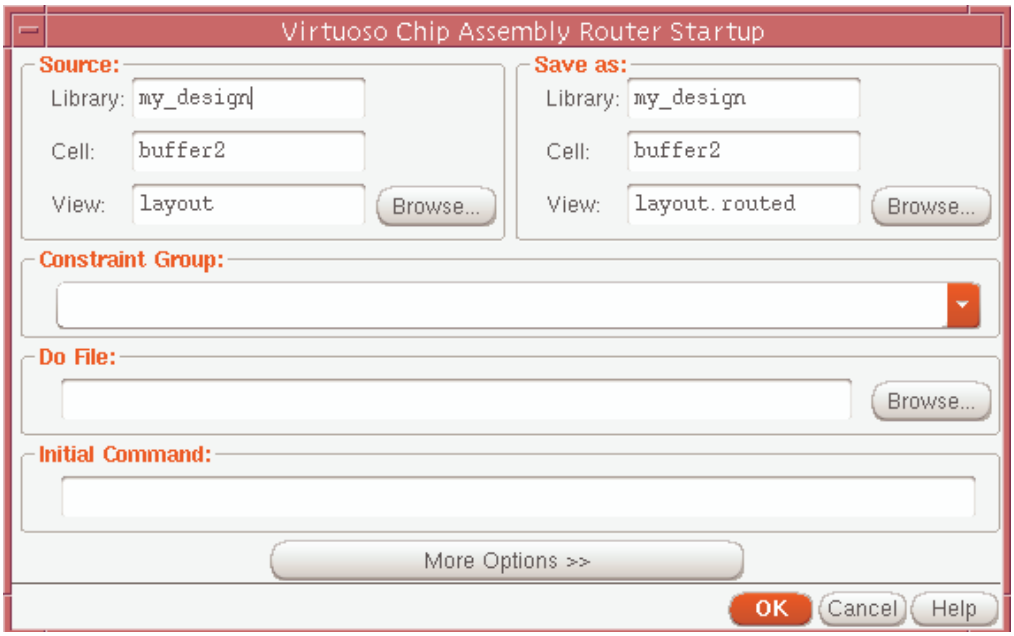


FIG. 33 Setup for VCAR

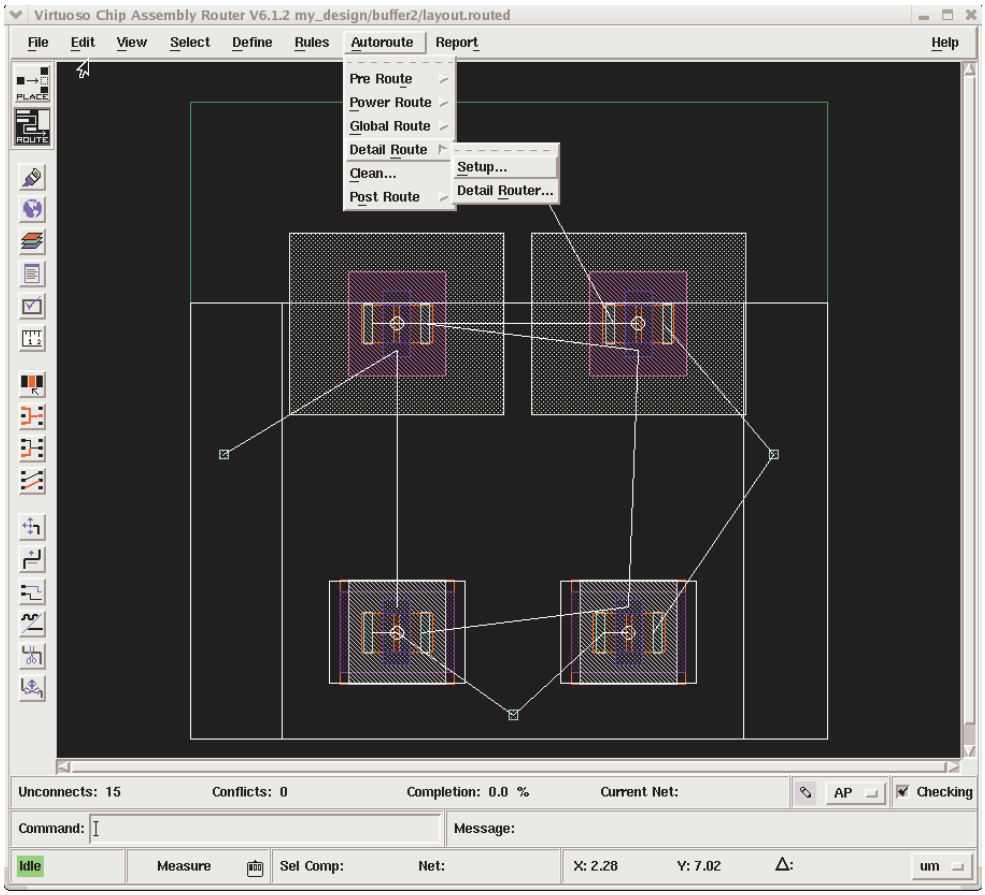


FIG. 34 Perform automatic routing

If you don’t feel comfortable with the results from VCAR, you can manually fix the routing results in the layout window. For our design, the final layout after manually fixed is shown in Fig 35.

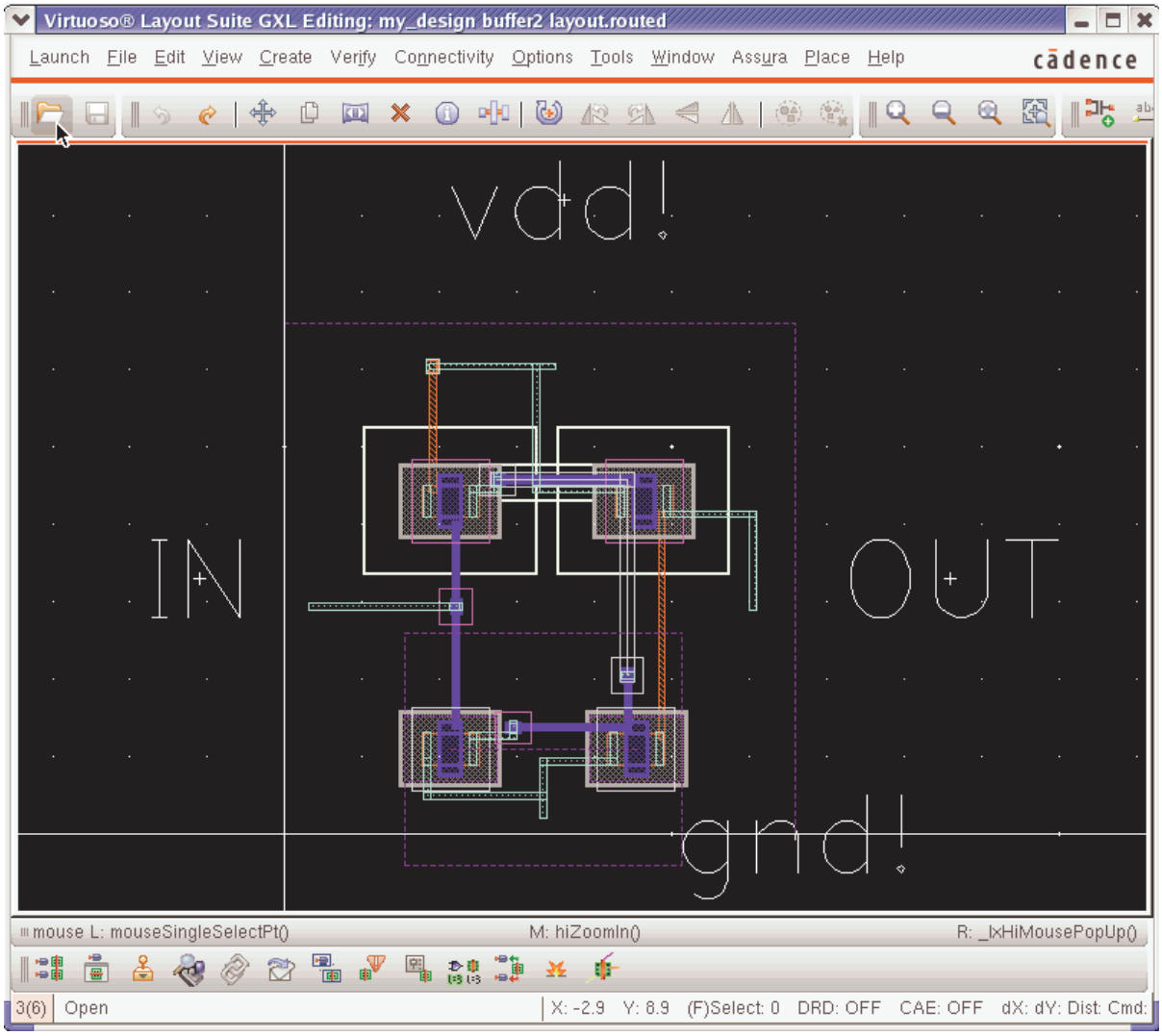


FIG. 35 Final layout

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 "assura_tech" (Fig. 36) 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.

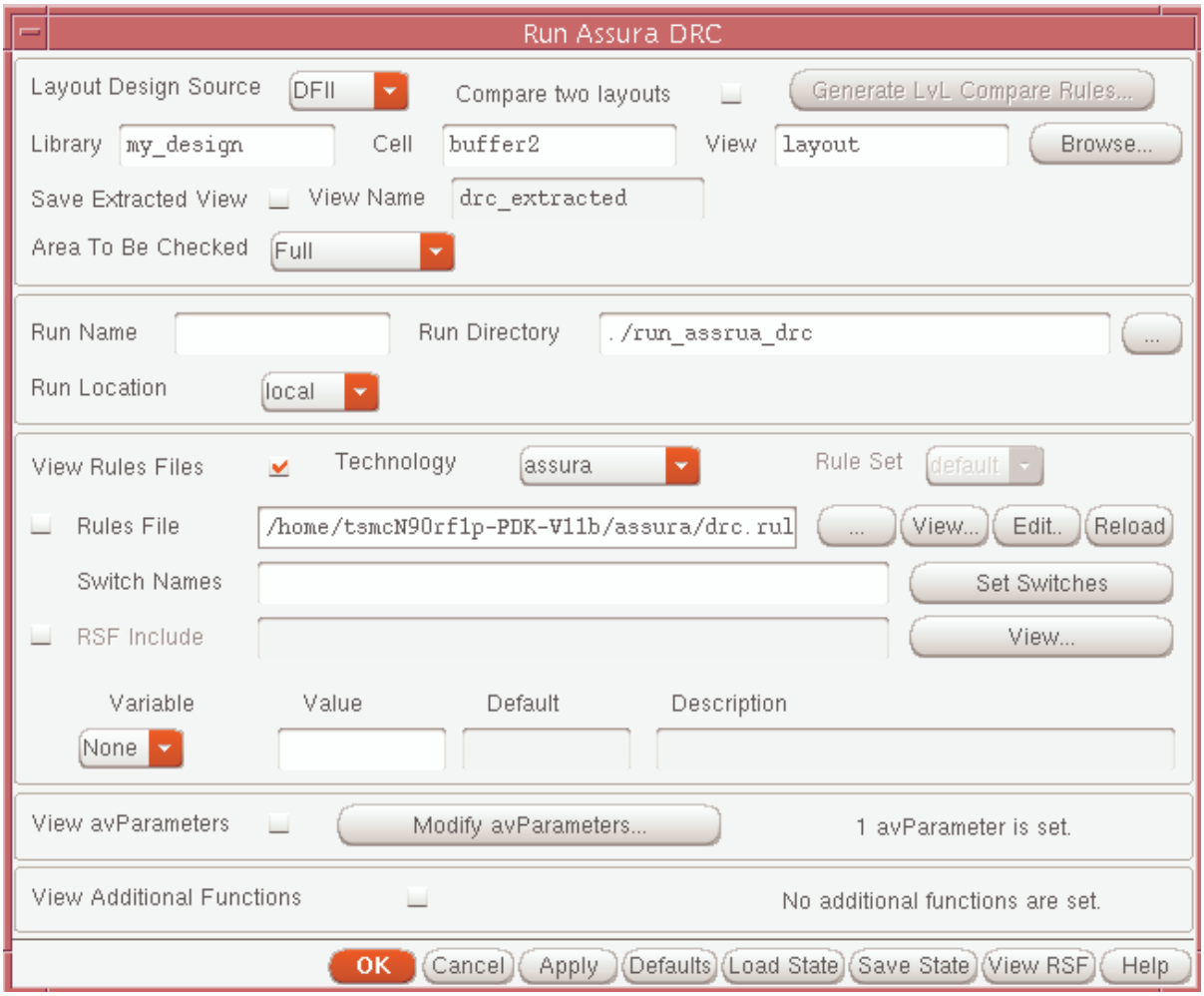


FIG. 36 Assura DRC setup window

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 “assura_tech” (Fig. 37) 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.

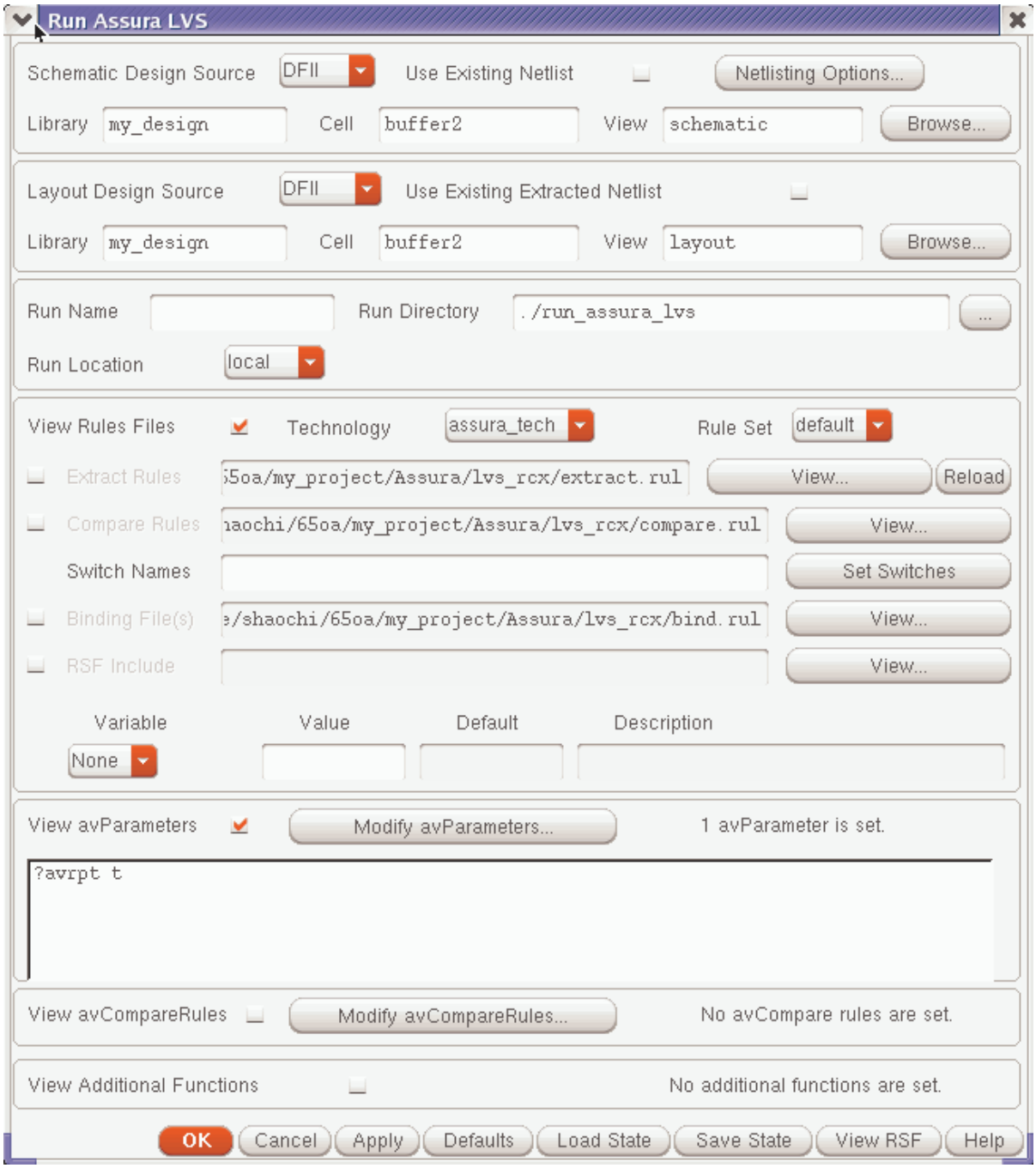


FIG. 37 Assura LVS setup window

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 38).
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 39).
4. Click “OK” to start the Assura RC extraction. After the RC extraction is completed, a new view (“av_extracted” view, Fig 40) which contains not only the original components but also the parasitic devices will be generated and then can be used for post-layout simulation.

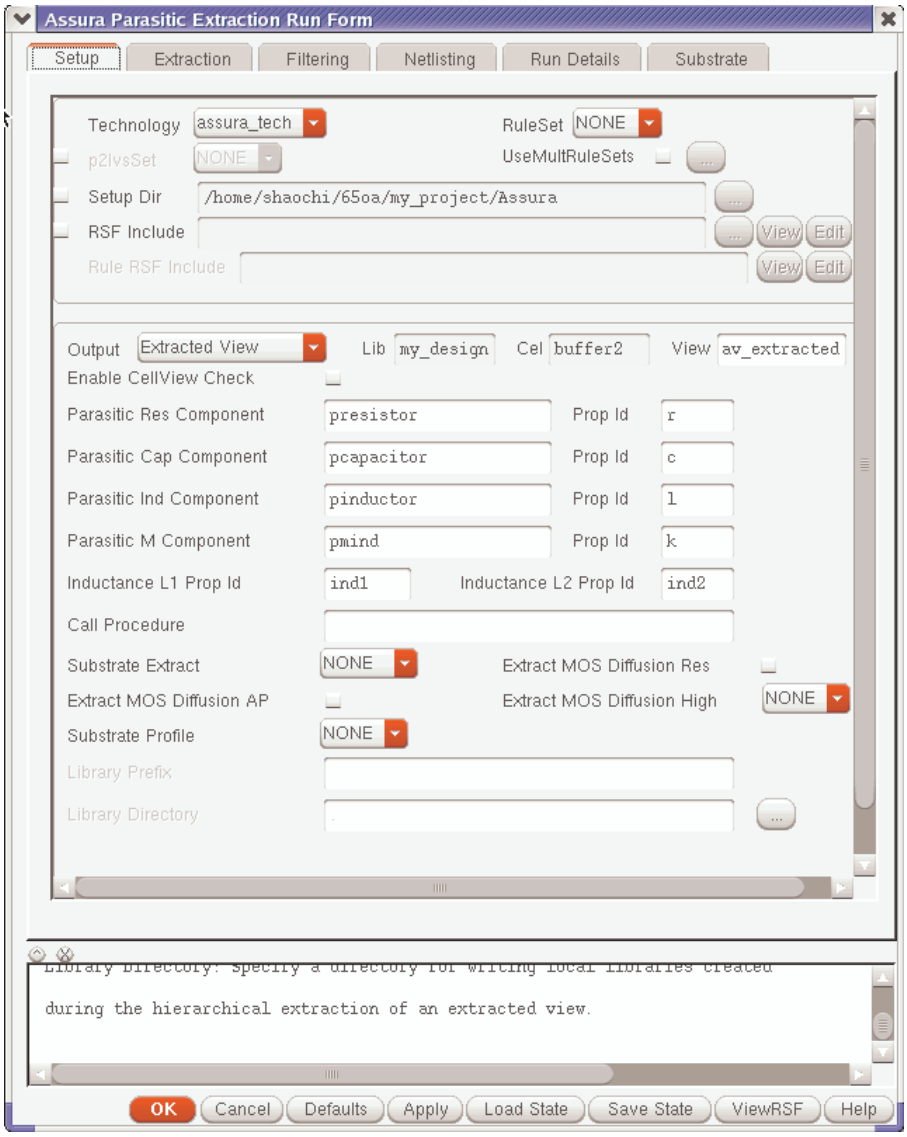


FIG. 38 Setup folder of Assura RCX window

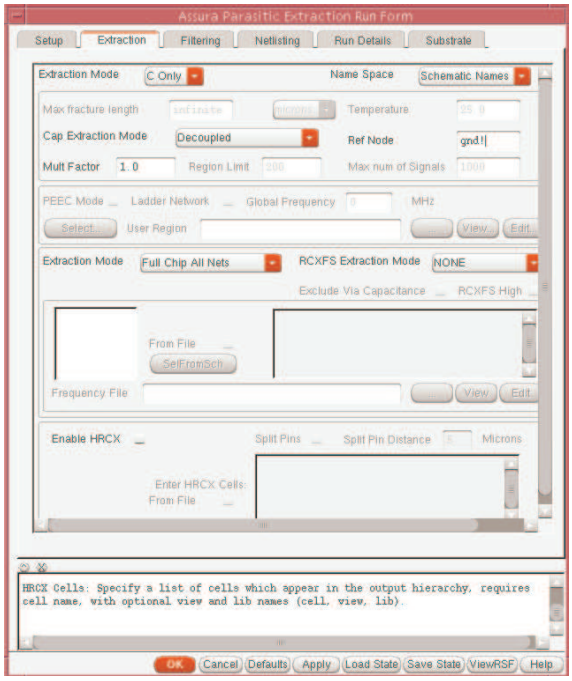


FIG. 39 Extraction folder of Assura RCX window

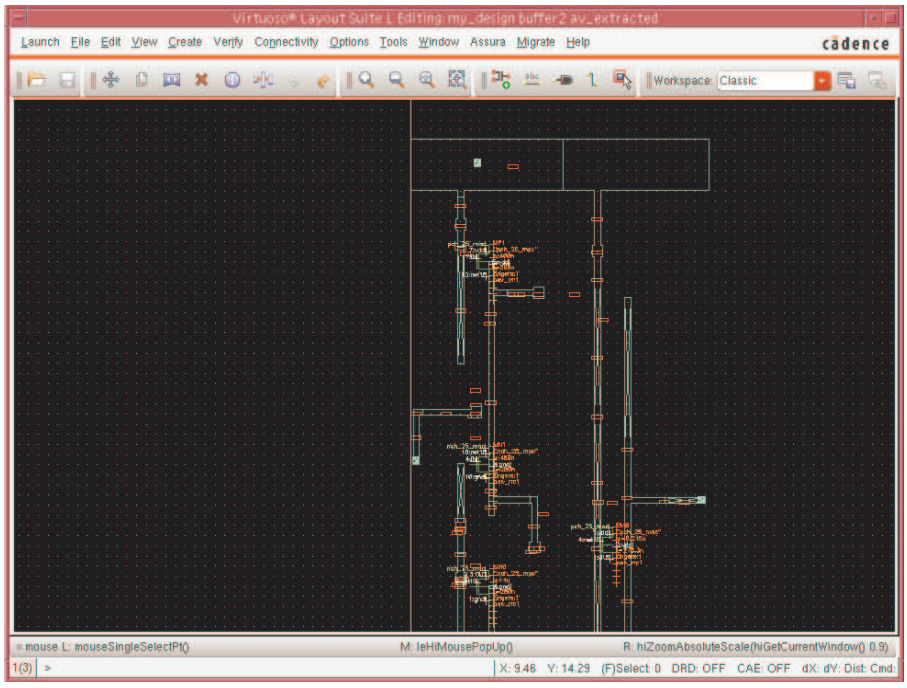


FIG. 40 Assura av_extracted view

Assura GDS/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 41).
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: (add Pin text before stream out)

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 42).

FIG. 41 Export CDL netlist

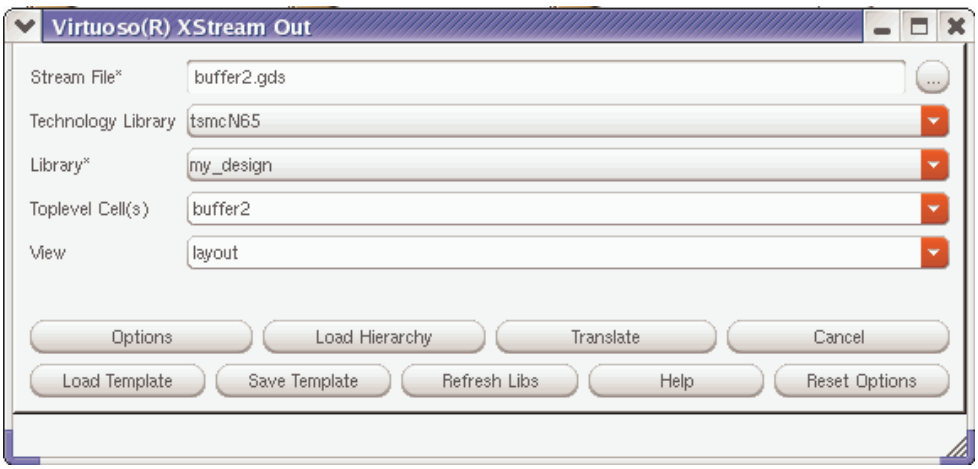


FIG. 42 Export GDS layout

Assura GDS 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 43).

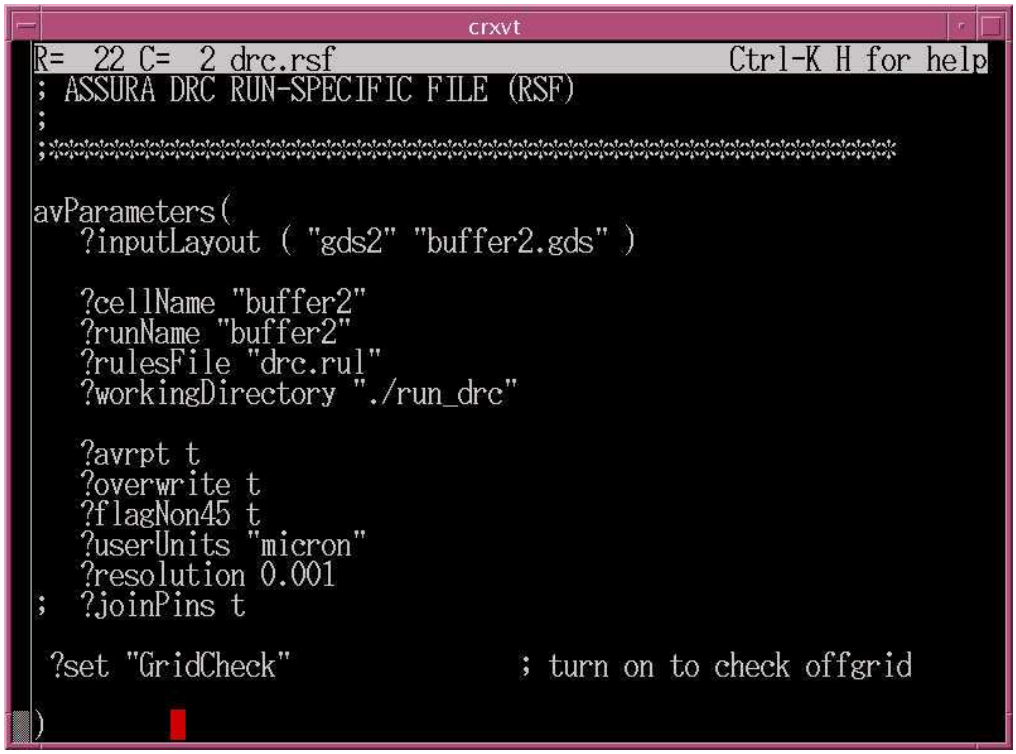


FIG. 43 Edit Assura DRC RSF file

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 GDS/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 44).

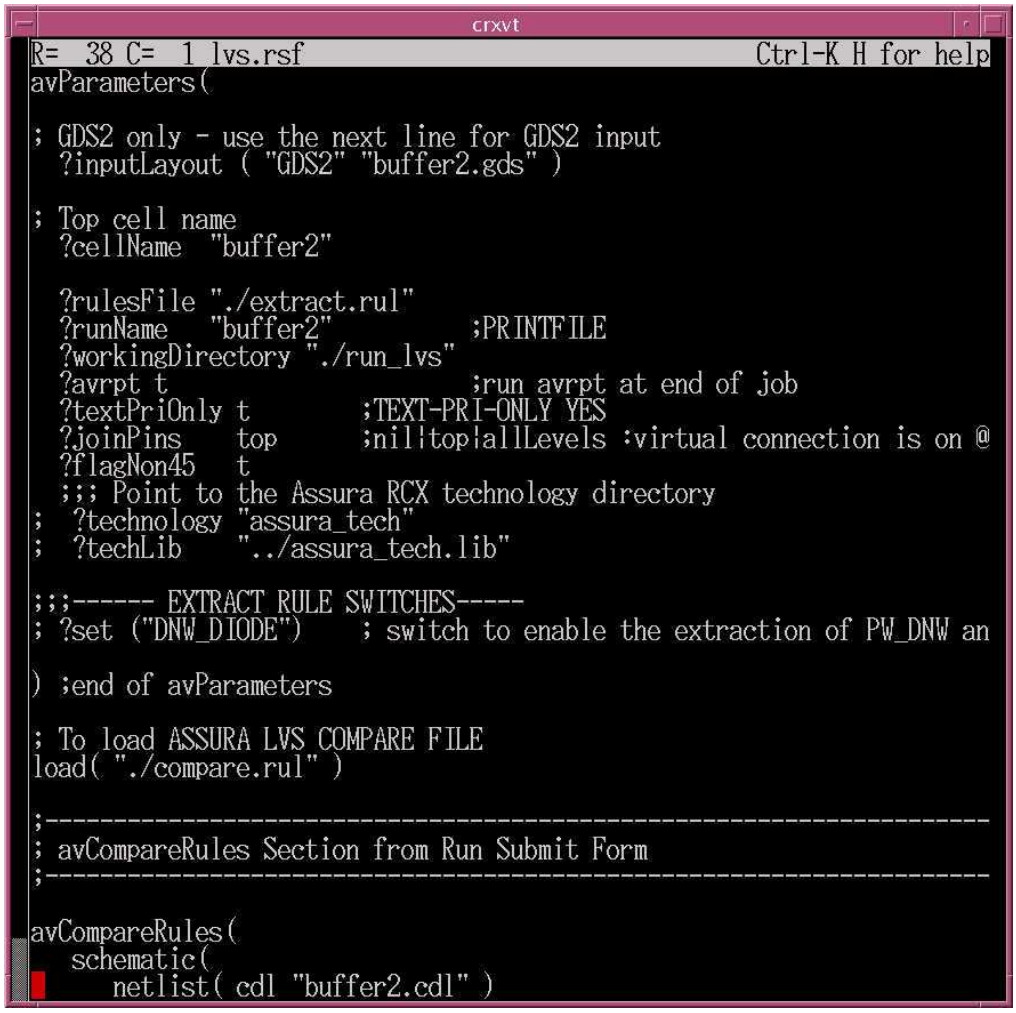


FIG. 44 Edit Assura LVS RSF file

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.

Chapter 6 Post-layout simulation

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.

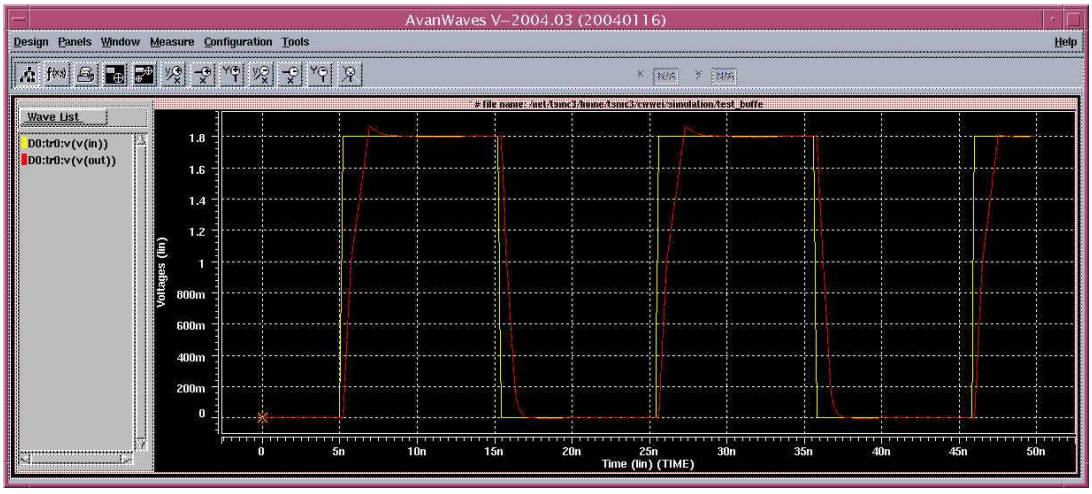
Simulating with the extracted netlist

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 as below.



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 'TB_buffer2' in this example) and switch on the ***av_extracted (or calibre)*** view for the cellView ***buffer2*** on the **Hierarchy Editor**.

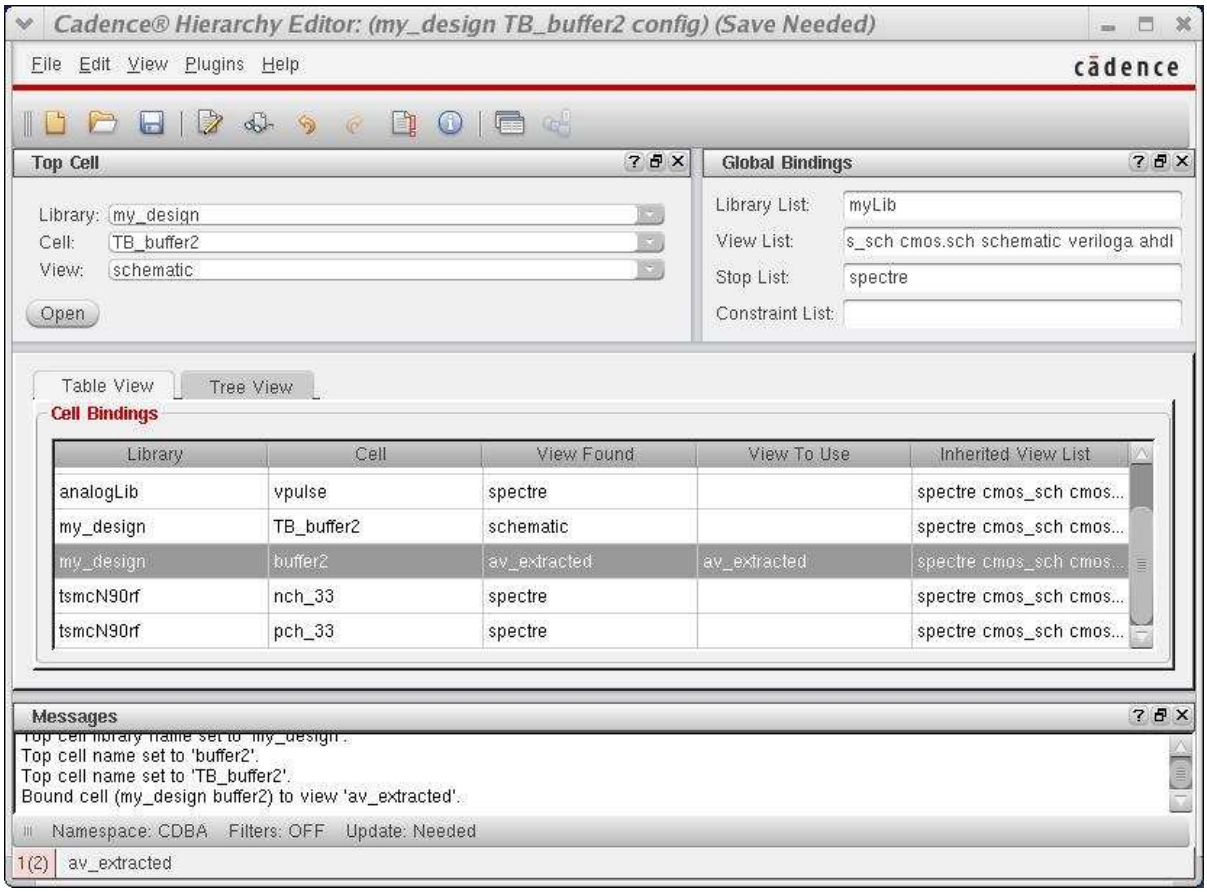


FIG. 45 switch design view to 'av_extracted' view

Note: The detail of creating a *config* view in Cadence Virtuoso environment please refer to Cadence Analog Design Environment manual.

2. Open the ADE L (Analog Artist environment L) and use the config view of the test fixture for simulation (fig 46). 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.

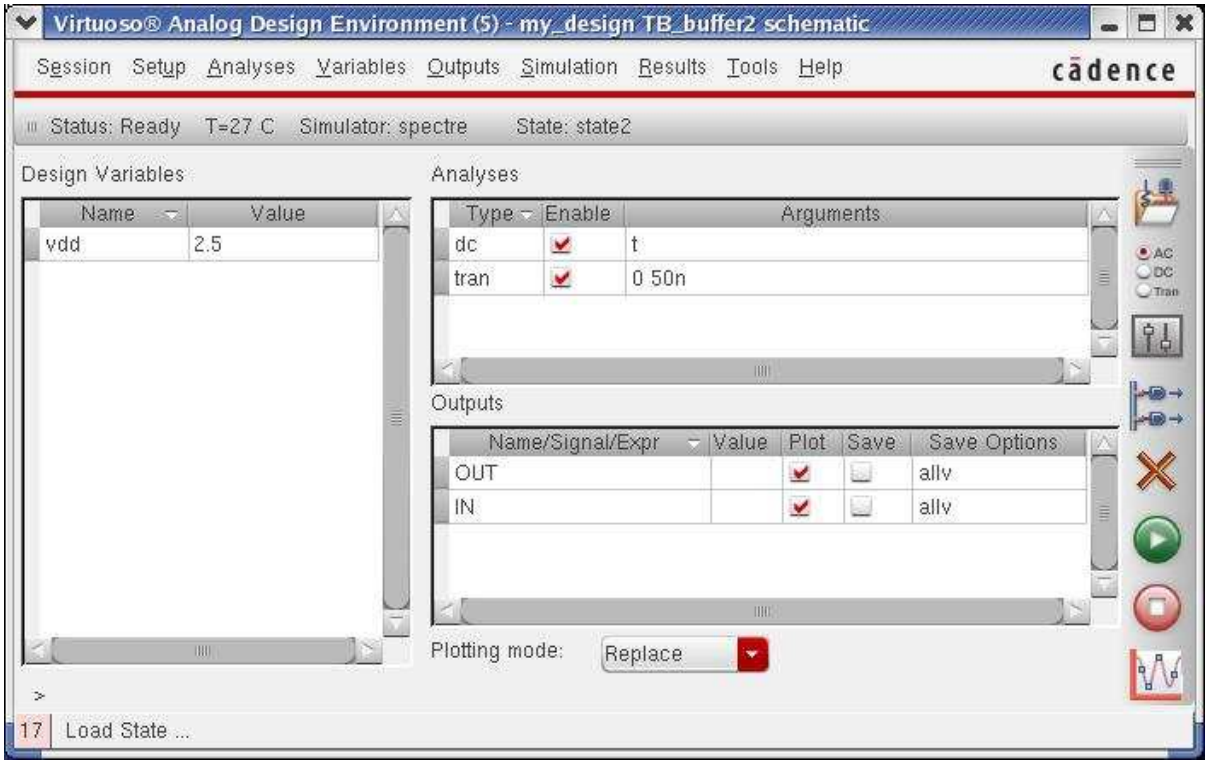


FIG. 46 Open ADE L (Analog Design Environment L) and use config view for post-sim

3. Run the post-layout simulation and check the simulation result shown as below.

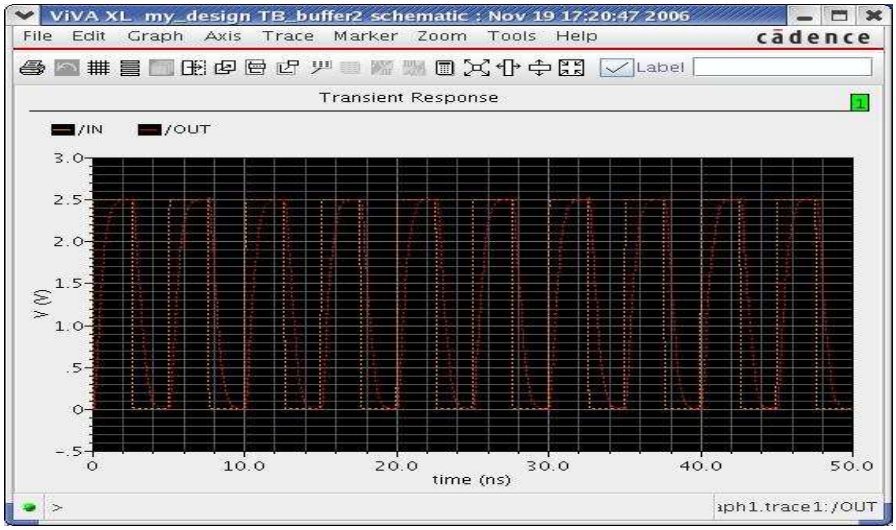


FIG. 47 Post-layout simulation result for extracted netlist