

# **Spectre® Simulator Fundamentals**

## **Section 1: Spectre Basics**

**Course Version SPECTRE 18.1**

**Lab Manual**

**Revision 1.0**

© 1990–2019 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

**Trademarks:** Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

**Restricted Print Permission:** This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

- The publication may be used solely for personal, informational, and noncommercial purposes;

- The publication may not be modified in any way;

- Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and

- Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence customers in accordance with, a written agreement between Cadence and the customer.

Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

**Restricted Rights:** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

## Table of Contents

### Spectre Simulator Fundamentals Section 1: Spectre Basics

<b>Module 1:</b>	<b>About This Course</b>	
	There are no labs in this module. ....	7
<b>Module 2:</b>	<b>Introduction to the Spectre® Simulator</b>	
	There are no labs in this module. ....	11
<b>Module 3:</b>	<b>Spectre Netlist Language</b>	
<b>Lab 3-1</b>	<b>Working with a Spectre Netlist .....</b>	<b>15</b>
	Logging In.....	15
	Opening a Terminal Window.....	15
	Examining the Amplifier Design and Testbench for LPF.....	16
	Examining the Netlist and Circuit Models .....	17
	Troubleshooting the Netlist.....	18
	Reviewing the Spectre Output Files.....	18
	Analyzing the Results Using Virtuoso Visualization and Analysis .....	19
	Creating a User-Defined Function .....	21
<b>Lab 3-2</b>	<b>Exploring the Netlist Control Statements .....</b>	<b>23</b>
	Printing Detailed Simulation Statistics .....	23
	Using the <i>info</i> Statement.....	24
	Using the <i>alter</i> Statement.....	25
	Using the <i>print</i> Statement .....	26
<b>Lab 3-3</b>	<b>Examining the Localized Temperature Settings (Optional).....</b>	<b>29</b>
	Support of TRISE/DTEMP and TEMP for Local Blocks.....	29
<b>Module 4:</b>	<b>Spectre Command-Line Interface</b>	
<b>Lab 4-1</b>	<b>Controlling the Spectre Outputs and Protecting Proprietary Information .....</b>	<b>35</b>
	Using a Simulation Configuration File to Run a Spectre Simulation.....	35
	Using the Environment Default Setting .....	36
	Overriding Using the <i>spectre</i> Command at Run Time.....	37
	Creating an Encrypted Netlist .....	38
<b>Lab 4-2</b>	<b>Exploring SPICE Compatibility .....</b>	<b>39</b>
	Performing a Spectre Simulation on a SPICE Netlist .....	39
	Enabling the <i>+spice</i> Option .....	40
<b>Module 5:</b>	<b>Spectre Analog Simulations in the ADE Explorer</b>	
<b>Lab 5-1</b>	<b>Setting Up and Running Spectre Simulation in ADE Explorer. ....</b>	<b>43</b>
	Starting the Cadence® Environment.....	43
	Examining the Amplifier Design and Testbench for LPF.....	44
	Opening the <i>maestro</i> View in Virtuoso ADE Explorer.....	46
	Choosing a Simulator.....	48
	Setting the Model Libraries.....	49
	Choosing Analyses.....	49
	Setting Design Variables.....	51
	Saving and Plotting Simulation Outputs .....	53
	Creating the Netlist .....	54

Running the Simulation.....55

# **Module 1: About This Course**



**There are no labs in this module.**





## **Module 2: Introduction to the Spectre<sup>®</sup> Simulator**



**There are no labs in this module.**



## **Module 3: Spectre Netlist Language**



## Lab 3-1 Working with a Spectre Netlist

**Objective:** To troubleshoot a netlist and create user-defined functions.

---

In this lab, you troubleshoot a netlist and create a user-defined function, and then review the simulation results of the amplifier design used in Low Pass filters.

### Logging In

1. At the login prompt, enter:

```
user1
```

or

```
whichever user login is enabled on your workstation.
```

2. At the password prompt, enter:

```
training
```

or

```
whichever password your instructor specifies
```

Your instructor or local system administrator might use a different user account name and password. Check before proceeding.

### Opening a Terminal Window

1. To open a new terminal window in Linux, **right**-click on an empty portion of the desktop and select **New Terminal**.
2. To open a new terminal window in Solaris, **right**-click on an empty portion of the desktop and select **Tools**, then **Terminal**.

A New Terminal window appears and you can start your lab exercises.

3. Before you start your lab session, check that the software is installed using the following command:

```
spectre -W
```

The version reported should be SPECTRE 18.1.0.421.isr9 or later.

```
virtuoso -W
```

The version reported should be IC6.1.8-64b.500.5 or later.

4. Now, you can start the labs by moving to the `~/SSFS1` directory in your terminal window.

```
cd ~/SSFS1
```

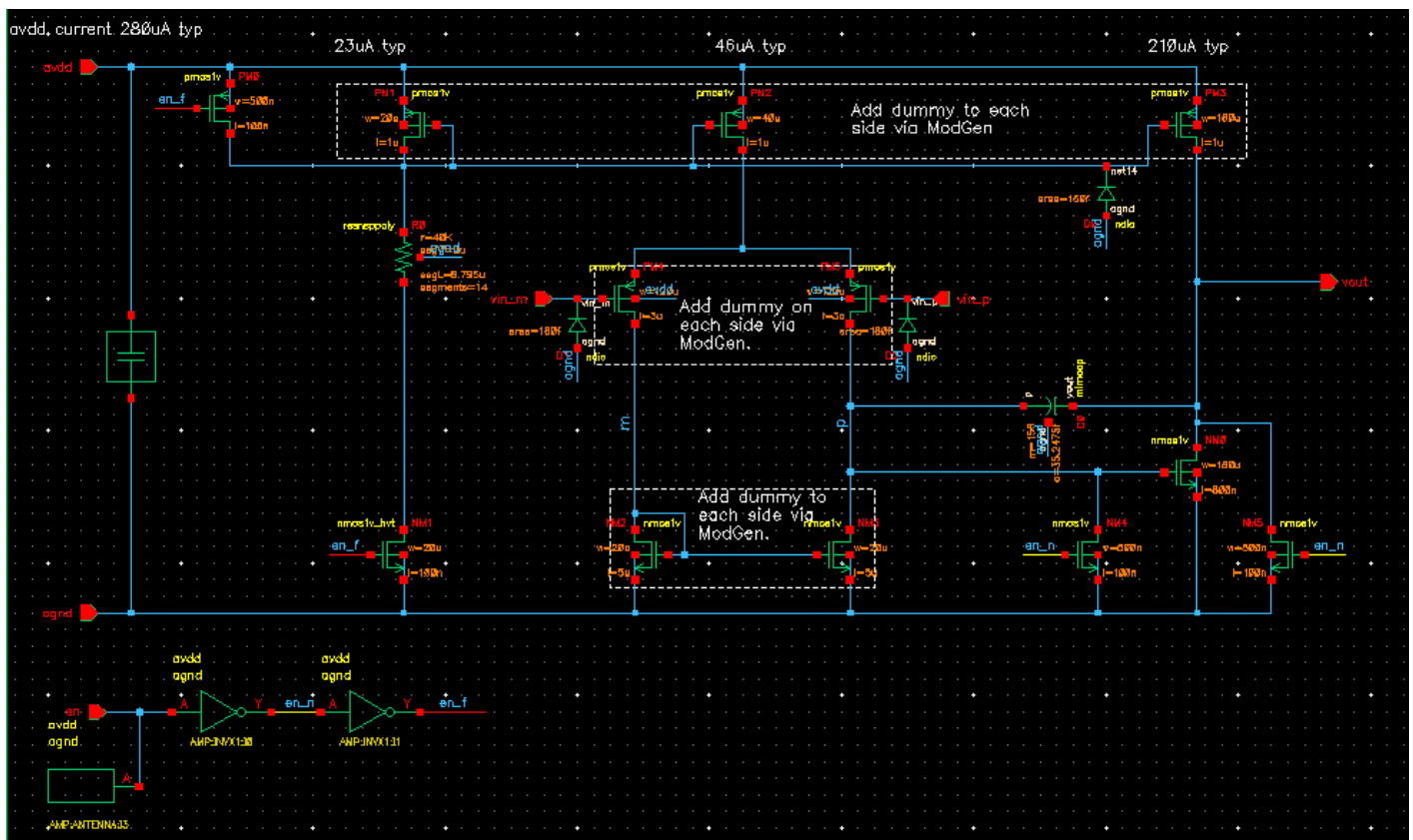
5. In the *SSFS1* directory, you have all *README*, source files (*cshrc*), and *copyright* files. Take a look at them before starting the lab exercises.
6. Run the clean script in that directory by entering:

```
./cleanDB
```

This command clears the simulation directory, *log*, *raw* and other result files that are already present in the whole lab database and ensures that you get fresh results while doing the lab.

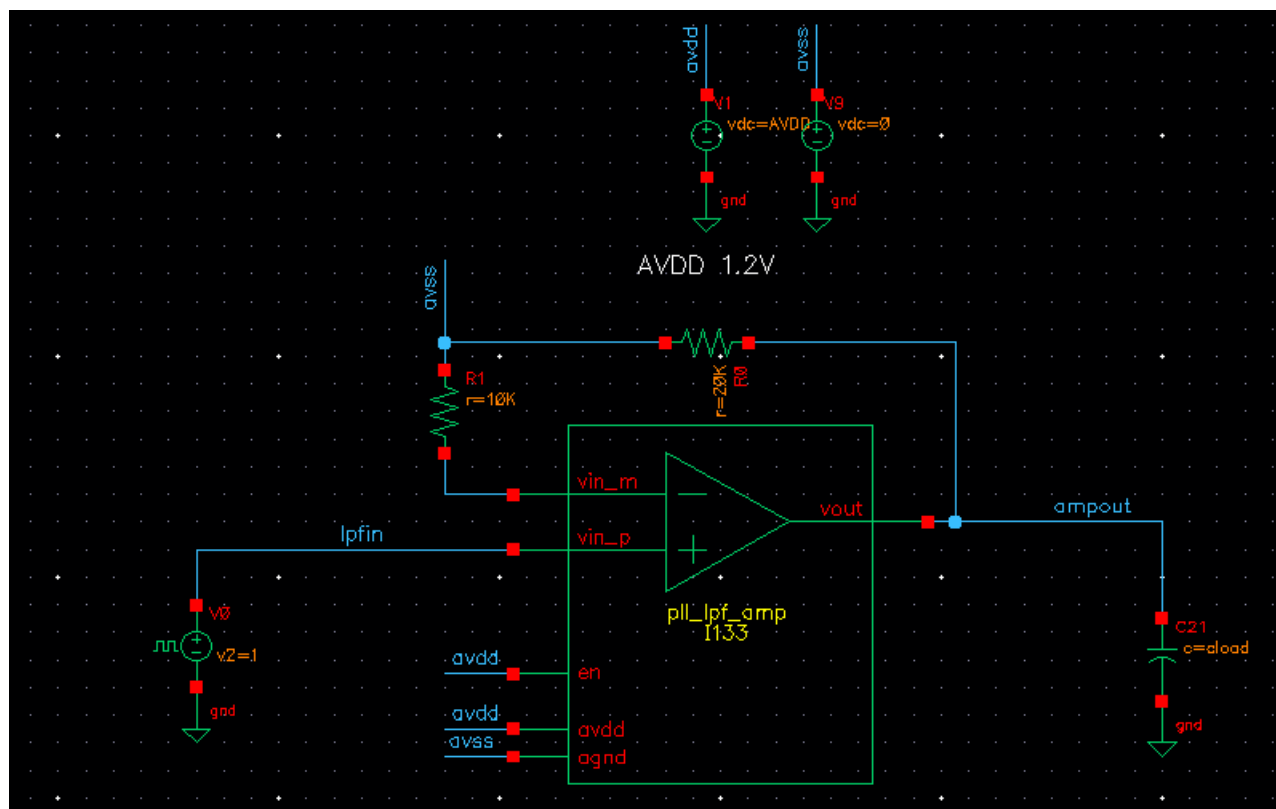
## Examining the Amplifier Design and Testbench for LPF

The design used in this lab is an amplifier design used in low-pass filter circuit of a PLL design with GPDK45 Technology. It uses MOS devices for biasing and output and have their model names displayed. In addition, the MOS devices have sizing information displayed.





The top-level *AMP\_Test* schematic testbench design is as shown below:



In this top-level testbench, you have the *pll\_lpf\_amp* I133 instance. The voltage supplied at the *vin\_put* terminal (**vin\_p**) is a PWL source that changes between a high (*val1=1.0V*) and low (*val0=0V*) value with timing characteristics (*period=5u*) defined.

## Examining the Netlist and Circuit Models

The netlist of the above amplifier design and testbench is already created and available in the *~SSFS1/SpectreNetlists/lpf\_amplifier* directory.

1. Move to the **lpf\_amplifier** directory to perform this lab exercise.

```
cd ~SSFS1/SpectreNetlists/lpf_amplifier
```

2. Before you start with the lab session, run the clean script in that directory by entering:

```
./clean
```

This command clears the *log*, *raw*, and other result files already present to ensure that you get fresh results while doing the lab.

3. Open the **lpf\_amp.scs** netlist in your text editor and review its contents.

The **.scs** suffix directs the Spectre® parser to expect the native Spectre syntax.

The netlist starts with the *simulation language* command that sets the language context of the netlist, followed by *global* statement.

4. Review the other statements of the netlist like the *subcircuit* definition of an amplifier design with its *instance* statements, and below that is the top-level testbench details with its instance definitions and following it are the analysis and other simulator settings.

## Troubleshooting the Netlist

1. Scan the netlist **lpf\_amp.scs** to see whether you can locate any problems. (This netlist contains an error.) If you can find the error, try fixing it. Go to the next step if you cannot spot the problem.

2. Run the Spectre simulation with the following command:

```
spectre lpf_amp.scs
```

The Spectre diagnostics give explicit hints as to the source of the errors. You can even look at the *lpf\_amp.log* file to check the simulation diagnostics.

**Important:** You have instances referencing an **undefined model** and **missing cload parameter** value.

3. Locate and fix the errors.

- a. The **model file** is included from the path *./models/spectre/gpdk045.scs* for the **typical** process corner.

**Tip:** Use the *include* statement with **tt section**.

- b. Add the **cload parameter** value as 1pF.

Do not proceed to the next step until the simulation runs successfully complete.

**Note:** You can ignore the notice related to ANTENNA instance I133 where the terminals are connected together to node *avdd*.

## Reviewing the Spectre Output Files

The two main output files mainly generated after a Spectre run are:

- ◆ Simulation logfile (*lpf\_amp.log*).
- ◆ The waveform database (*lpf\_amp.raw*) in the default *psfbin* format, where these simulation results can be analyzed using the Virtuoso® Visualization and Analysis (ViVA) tool.

**Note:** Also, all the shared objects during compilation of models are stored in a directory with extension *ahdlSimDB* (*lpf\_amp.ahdlSimDB*).

1. Looking at the Simulation log file (*lpf\_amp.log*), answer the following.

*What is the number of accepted tran steps for this run?*

Answer: \_\_\_\_\_

2. From the **Instance parameter values `element'** section, identify the length of NM2 and NM3.

*What is the length of I133.NM2 and I133.NM3 of g45n1svt?*

Answer: \_\_\_\_\_

**Note:** These messages are printed in the log file because of the *info* statements at the end of this netlist file.

3. At the end of the log file, notice the time taken and memory used.

*What is the time taken and memory used for this Spectre simulation?*

Answer: \_\_\_\_\_

4. Close all the files.

## Analyzing the Results Using Virtuoso Visualization and Analysis

1. Start the Visualization and Analysis tool from the command line by entering this command:

```
viva &
```

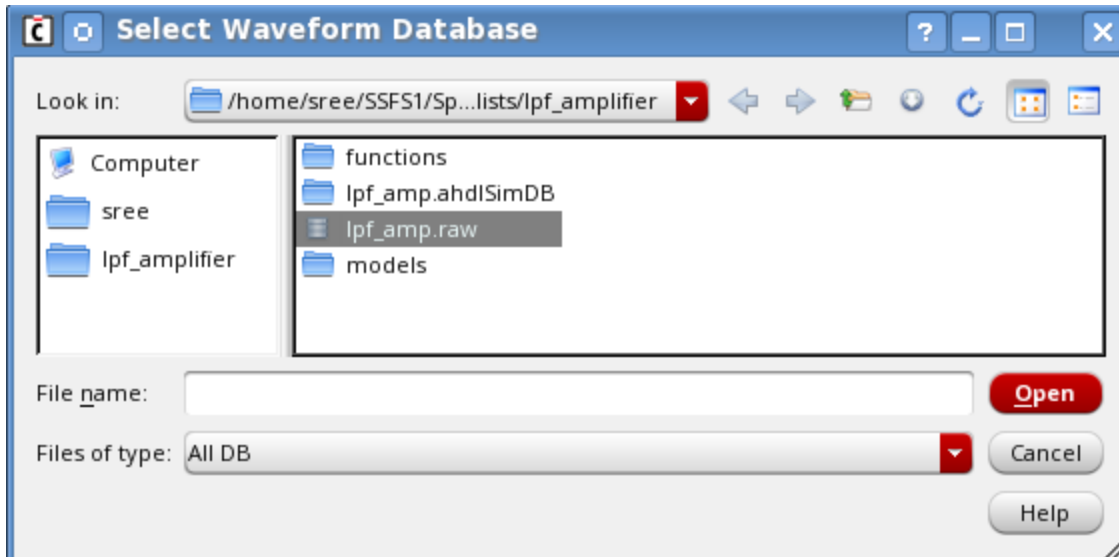
After a few seconds, the Virtuoso Visualization and Analysis window appears in the **Basic** workspace, along with a Log window that is similar to the CIW.

This Visualization and Analysis window consists of the Browser assistant on the left side, while an empty Qt- graph window is on its right side.

**Note:** The *Basic* workspace is the default for standalone Visualization and Analysis. You can change the workspaces to Browser, Classic, Marker Table, TM or Basic from the Workspace cyclic field. **Classic** workspace is the default for ADE Visualization and Analysis.

2. Choose **File – Open Results** from the pull-down menu of the Visualization and Analysis window.

The Select Waveform Database window appears.

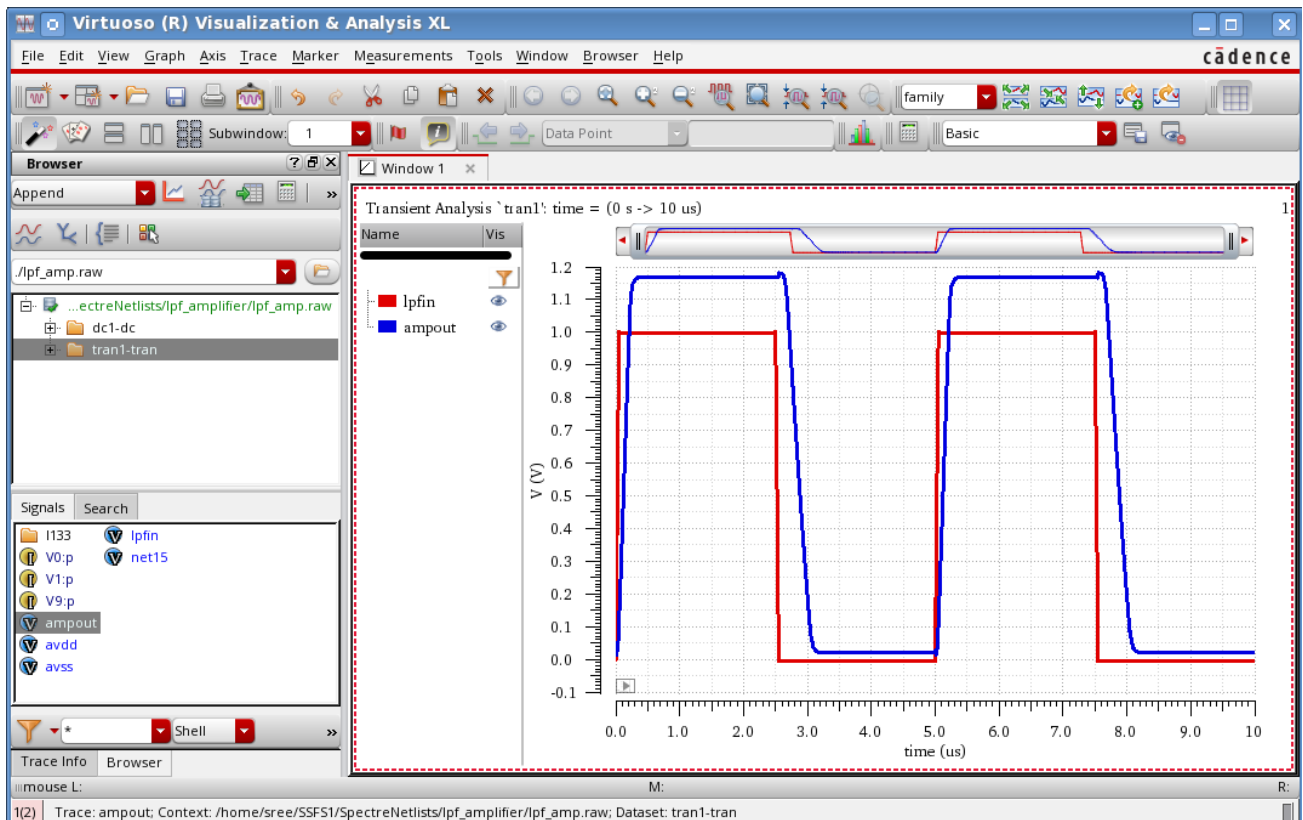


3. Select **lpf\_amp.raw** from the *lpf\_amplifier* folder and click **Open**.

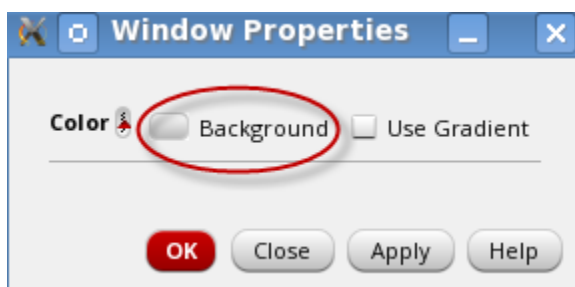
On the left side of the Visualization and Analysis XL window, you have the Browser with the path to *lpf\_amp.raw* on its upper pane while its lower pane displays numerous folders, including the *dc* and *tran* results folders.



4. Expand **lpf\_amp.raw** from the upper left pane by clicking the **+** sign. Click to highlight the *tran1-tran* folder. Plot the **lpfin** and **ampout** signals of the low pas filter amplifier design.



**Tip:** To modify the background color from black to white, choose **File** – **Window Properties** in the ViVA XL window and change the Background color to white.



5. You can also note down the DC information from the (*dcl-dc*), available in this raw directory.

*What is the value of ampout signal?*

Answer: \_\_\_\_\_

6. After analyzing the simulation results, close the Graph window and choose **File** – **Exit** to close the Visualization and Analysis tool.

## Creating a User-Defined Function

The user-defined function capability of Spectre allows you to build upon the provided set of built-in mathematical and trigonometric functions. You can write your own functions, and call these functions from within any expression. Arguments to user-defined functions are taken as **real** values, and the functions return **real** values.

In this section, you will create user-defined functions for the length of NM2 and NM3 MOS devices.

1. In the `~SSFS1/SpectreNetlists/lpf_amplifier/functions` directory, create a file called **mos\_length.scs**:

```
vi mos_length.scs
```

2. In this file, create a function named **mos\_length** that accepts two real arguments, *w* and *mos\_ratio*. The return value needs to reflect the length of the MOS device, as *w* divided by *mos\_ratio*.

**Tip:** Refer to Creating User-Defined Functions in the lecture slide.

3. Open the **lpf\_amp.scs** netlist in the `~SSFS1/SpectreNetlists/lpf_amplifier` directory.
  - a. Write a statement to include the *mos\_length.scs* file in the netlist.
  - b. In the subcircuit definition, change the length (l) value of NM2 and NM3 from 5u to the function *mos\_length*.

- c. Use *width* as 100u and *ratio* as 20, in the length function call for NM2 and NM3 defining them as parameters.

4. Save and close the **lpf\_amp.scs** file.

5. Ensure that you are in the `~SSFS1/SpectreNetlists/lpf_amplifier` directory and simulate your **lpf\_amp.scs** netlist using Spectre software from the command line.

```
spectre lpf_amp.scs
```

Check the *lpf\_amp.log* file for the calculated lengths of NM2 and NM3.

**Tip:** Your calculated length value should be 5um, if *ratio*=20.

6. Try different values for *ratio* (*ratio*=10) and rerun the simulation.

**Note:** Make sure the value for *width* stays at **100u** to be consistent with the instance statement.

7. After each successful simulation run, check the *lpf\_amp.log* file for the calculated lengths. The *lpf\_amp.log* file is rewritten on each simulation.

8. Close all the files.

**Note:** Solutions for this lab are in the `~SSFS1/SpectreNetlists/lpf_amplifier/.solutions` directory.



## Lab 3-2 Exploring the Netlist Control Statements

**Objective:** To control the flow of the Spectre simulation using netlist control statements.

---

The Spectre simulator lets you place a sequence of control statements in the netlist. You can use the same control statement more than once.

In this lab, you explore the use of some of the commonly used control statements like *options*, *alter*, *info* and *print* statements with an *opamp* design.

### Printing Detailed Simulation Statistics

You can control printing of detailed simulation statistics report using the *simstat=detailed* parameter setting in the *options* statement.

1. Move to the **control\_stat** directory to perform this lab exercise.

```
cd ~SSFS1/SpectreNetlists/control_stat
```

2. Before you start with the lab session, run the clean script in that directory by entering:

```
./clean
```

This command clears the *log*, *raw*, and other result files already present to ensure that you get fresh results while doing the lab.

3. Open and view the **bip\_opamp.scs** netlist, which is an opamp design composed mainly of bipolar transistors. Use any text editor.

```
vi bip_opamp.scs
```

There is an external compensation capacitor, *c\_comp*, which is set to 3pF, and a *vcvs* element that looks at the open loop gain of the amplifier.

4. Add the *simstat=detailed* parameter setting to the **options** statement.
5. Run the simulation with the following command:

```
spectre bip_opamp.scs
```

6. Check the *bip\_opamp.log* for the **detailed** simulation statistics report with virtual memory usage, current and peak physical memory usage and available physical/swap memory in addition to the *basic* setting information.

```

*****
Post-Transient Simulation Summary
*****
- To further speed up simulation, consider
  add ++aps on command line
- The circuit contains signals of the voltage > 10V, consider to set highvoltage=yes to get better accuracy and convergence ability
*****

During simulation, the CPU load for active processors is :
Spectre 0 (88.6 %)
Other
Initial condition solution time: CPU = 2 ms, elapsed = 5.28812 ms.
Intrinsic tran analysis time: CPU = 11.998 ms, elapsed = 14.0049 ms.
Total time required for tran analysis 'tran1': CPU = 13.998 ms, elapsed = 22.383 ms.
Time accumulated: CPU = 305.953 ms, elapsed = 394.104 ms.
Peak resident memory used = 73.1 Mbytes.
Resident physical memory = 73.1 Mbytes.
Peak virtual memory used = 579 Mbytes.
Virtual memory using = 579 Mbytes.
Physical memory available = 678 Mbytes.
Swap memory available = 3.99 Gbytes.

Aggregate audit (9:41:26 PM, Tue Aug 20, 2019):
Time used: CPU = 309 ms, elapsed = 398 ms, util. = 77.5%.
Time spent in licensing: elapsed = 68.1 ms, percentage of total = 17.1%.
Peak memory used = 73.5 Mbytes.
Simulation started at: 9:41:25 PM, Tue Aug 20, 2019, ended at: 9:41:26 PM, Tue Aug 20, 2019, with elapsed time (wall clock): 398 ms.
spectre completes with 0 errors, 0 warnings, and 1 notice.

```

7. Close all the files.

## Using the *info* Statement

You use the *info* statement to print the circuit information analysis outputs.

1. Open the the **bip\_opamp.scs** netlist.
2. Add an *info* analysis statement named **outputInfo** to print all the DC analysis **output** information into the **logfile** before the transient analysis statement.
3. Rerun the Spectre simulation.



4. Check the logfile *bip\_opamp.log* for the following output parameter values.

Total Power Dissipation = 414.218 mW.

Convergence achieved in 862 iterations.  
 DC simulation time: CPU = 57.992 ms, elapsed = 81.877 ms.  
 Total time required for dc analysis `dcl`: CPU = 57.992 ms, elapsed = 81.9521 ms.  
 Time accumulated: CPU = 283.956 ms, elapsed = 357.396 ms.  
 Peak resident memory used = 72.1 Mbytes.  
 Resident physical memory = 72.1 Mbytes.  
 Peak virtual memory used = 579 Mbytes.  
 Virtual memory using = 579 Mbytes.  
 Physical memory available = 678 Mbytes.  
 Swap memory available = 3.99 Gbytes.

\*\*\*\*\*

Output parameter values `outputInfo':

\*\*\*\*\*

Parameters measured at T = 27 C:

Instance: xi0.q0 of DOA

Model: bmen

Primitive: bjt

tempeff = 27 C

meff = 1

ibeff = 1 pOhm

ibceff = 1 pOhm

iseeff = 0 Ohm

isceff = 0 Ohm

isseff = 0 Ohm

Instance: xi0.q1 of DOA

Model: bmen

Primitive: bjt

tempeff = 27 C

meff = 1

ibeff = 1 pOhm

ibceff = 1 pOhm

iseeff = 0 Ohm

isceff = 0 Ohm

isseff = 0 Ohm

## Using the *alter* Statement

You use the *alter* statement to change the parameters of circuits, sub-circuits, and individual models or components.

1. Open the **bip\_opamp.scs** netlist.
2. Add an **alter** statement named **Change1** to change the circuit **temperature** to **100C** for the transient analysis.

**Tip:** Refer to the lecture material for syntax and more information.

3. Rerun the Spectre simulation.

4. Check the log file *bip\_opamp.log* and observe the change in temperature before transient analysis.

Change1: `temp' set to 100 C.

```
*****
Transient Analysis `tran1': time = (0 s -> 1 us)
*****
Trying `homotopy = gmin' for initial conditions.
DC simulation time: CPU = 25.996 ms, elapsed = 36.104 ms.
```

Opening the PSF file bip\_opamp.raw/tran1.tran.tran ...

Important parameter values:

```
start = 0 s
outputstart = 0 s
stop = 1 us
step = 1 ns
maxstep = 20 ns
ic = all
useprevic = no
skipdc = no
reltol = 1e-03
abstol(V) = 1 uV
abstol(I) = 1 pA
temp = 100 C
tnom = 27 C
tempeffects = all
errpreset = moderate
method = traponly
literation = 3.5
relref = sigglobal
cmin = 0 F
gmin = 1 pS
```

Output and IC/nodeset summary:

```
save 4 (current)
save 70 (voltage)
others 1
```

```
.....9.....8.....7.....6.....5.....4.....3.....2.....1.....0
Number of accepted tran steps = 54
```

## Using the *print* Statement

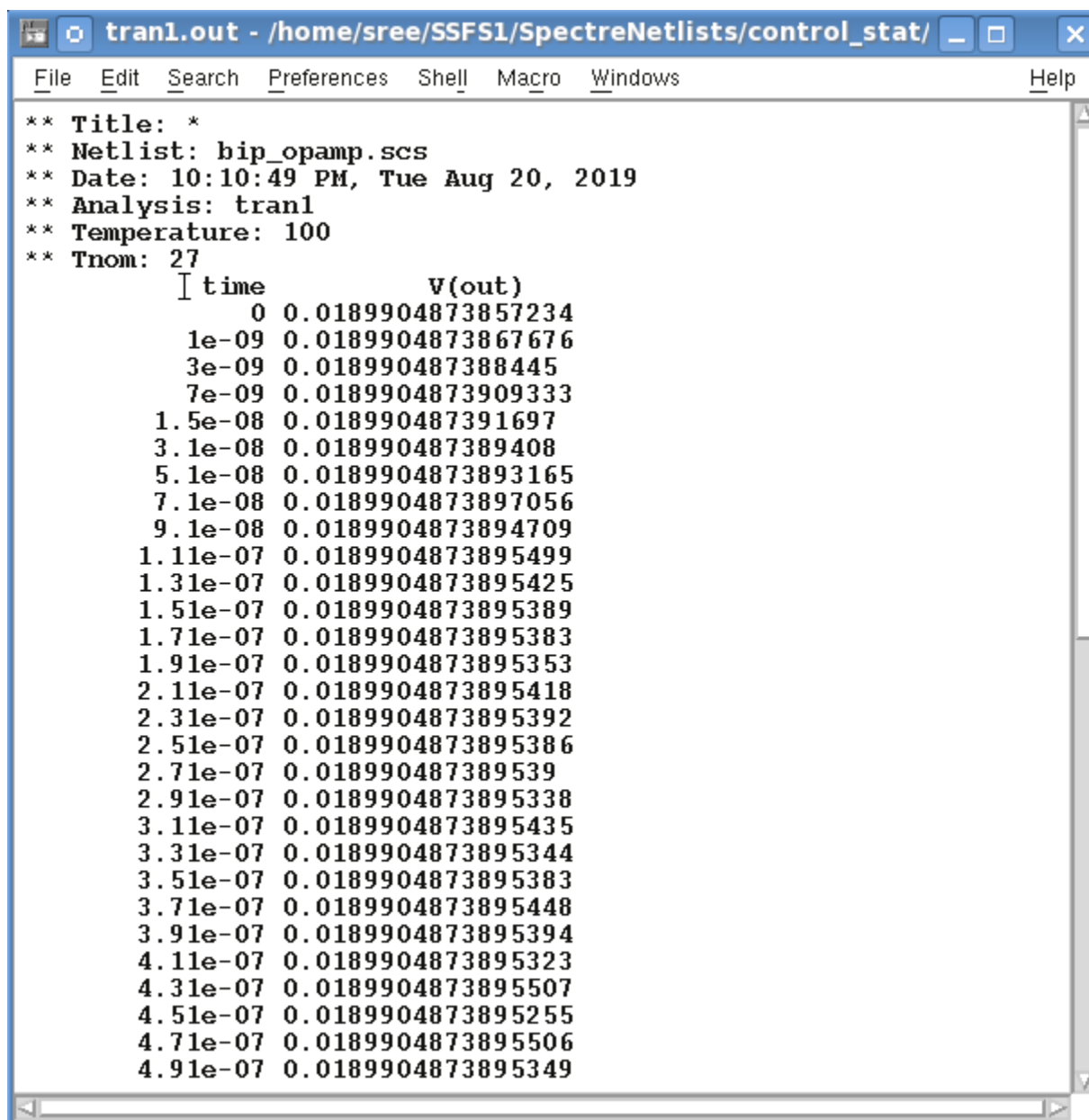
You use the *print* statement to print signal and instance data to an output file. You can use the *print* statement for AC, DC, transient, noise, and sweep analyses.

1. Open the the **bip\_opamp.scs** netlist.
2. Add a *print* statement after the transient analysis statement to print the **out** voltage value of transient analysis **tran1** to an output file named “*tran1.out*”. Specify a precision of **%15.15g** to output 15 decimal digits in mantissa.

**Tip:** Refer to the lecture material for syntax and more information.

3. Rerun the Spectre simulation.

4. Along with other output files *bip\_opamp.log* and *bip\_opamp.raw*, you notice a new output file named *tran1.out* is generated because of the **print** statement. This file contains the **out** signals information during every timestep.

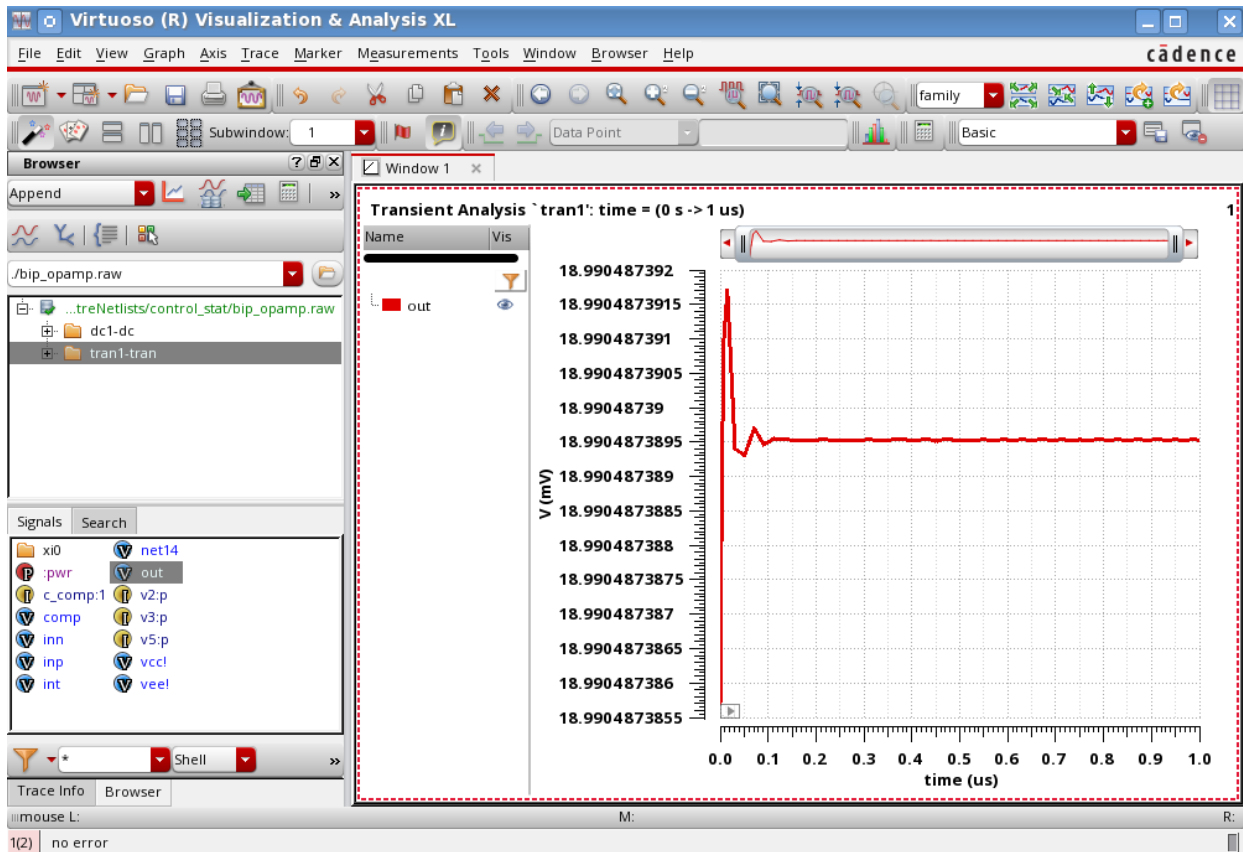


```

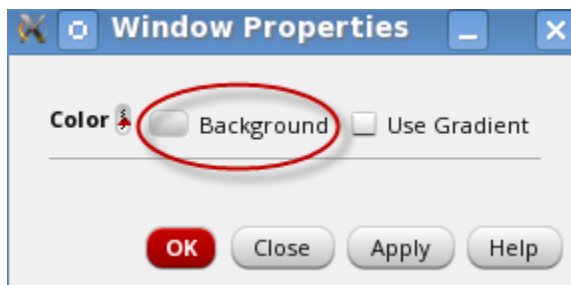
** Title: *
** Netlist: bip_opamp.scs
** Date: 10:10:49 PM, Tue Aug 20, 2019
** Analysis: tran1
** Temperature: 100
** Tnom: 27
    time          V(out)
    0 0.0189904873857234
    1e-09 0.0189904873867676
    3e-09 0.018990487388445
    7e-09 0.0189904873909333
    1.5e-08 0.018990487391697
    3.1e-08 0.018990487389408
    5.1e-08 0.0189904873893165
    7.1e-08 0.0189904873897056
    9.1e-08 0.0189904873894709
    1.1e-07 0.0189904873895499
    1.3e-07 0.0189904873895425
    1.5e-07 0.0189904873895389
    1.7e-07 0.0189904873895383
    1.9e-07 0.0189904873895353
    2.1e-07 0.0189904873895418
    2.3e-07 0.0189904873895392
    2.5e-07 0.0189904873895386
    2.7e-07 0.018990487389539
    2.9e-07 0.0189904873895338
    3.1e-07 0.0189904873895435
    3.3e-07 0.0189904873895344
    3.5e-07 0.0189904873895383
    3.7e-07 0.0189904873895448
    3.9e-07 0.0189904873895394
    4.1e-07 0.0189904873895323
    4.3e-07 0.0189904873895507
    4.5e-07 0.0189904873895255
    4.7e-07 0.0189904873895506
    4.9e-07 0.0189904873895349

```

- Plot the **out** signal from the *bip\_opamp.raw* waveform database using the Virtuoso Visualization and Analysis tool and compare it with the *tran1.out* file values.



**Tip:** To modify the background color from black to white, choose **File – Window Properties** in the ViVA XL window and change the Background color to white.



- After examining the results, close all the files, exit the Virtuoso Visualization and Analysis tool and proceed to the next lab exercise.

**Note:** Solutions for this lab are in the *~SSFS1/SpectreNetlists/control\_stat/.solutions* directory.

**End of Lab**

## Lab 3-3 Examining the Localized Temperature Settings (Optional)

**Objective:** To examine the effects of **TRISE/DTEMP** and **TEMP** parameters set on subckts.

The **TRISE/DTEMP** parameter accumulates the temperature for hierarchical or nested sub-circuits while the **TEMP** parameter set in the sub-circuit instance will exclude all previous temperature settings. In this section of the lab, you examine the support of **TRISE/DTEMP** and **TEMP** options for local blocks in a Spectre netlist.

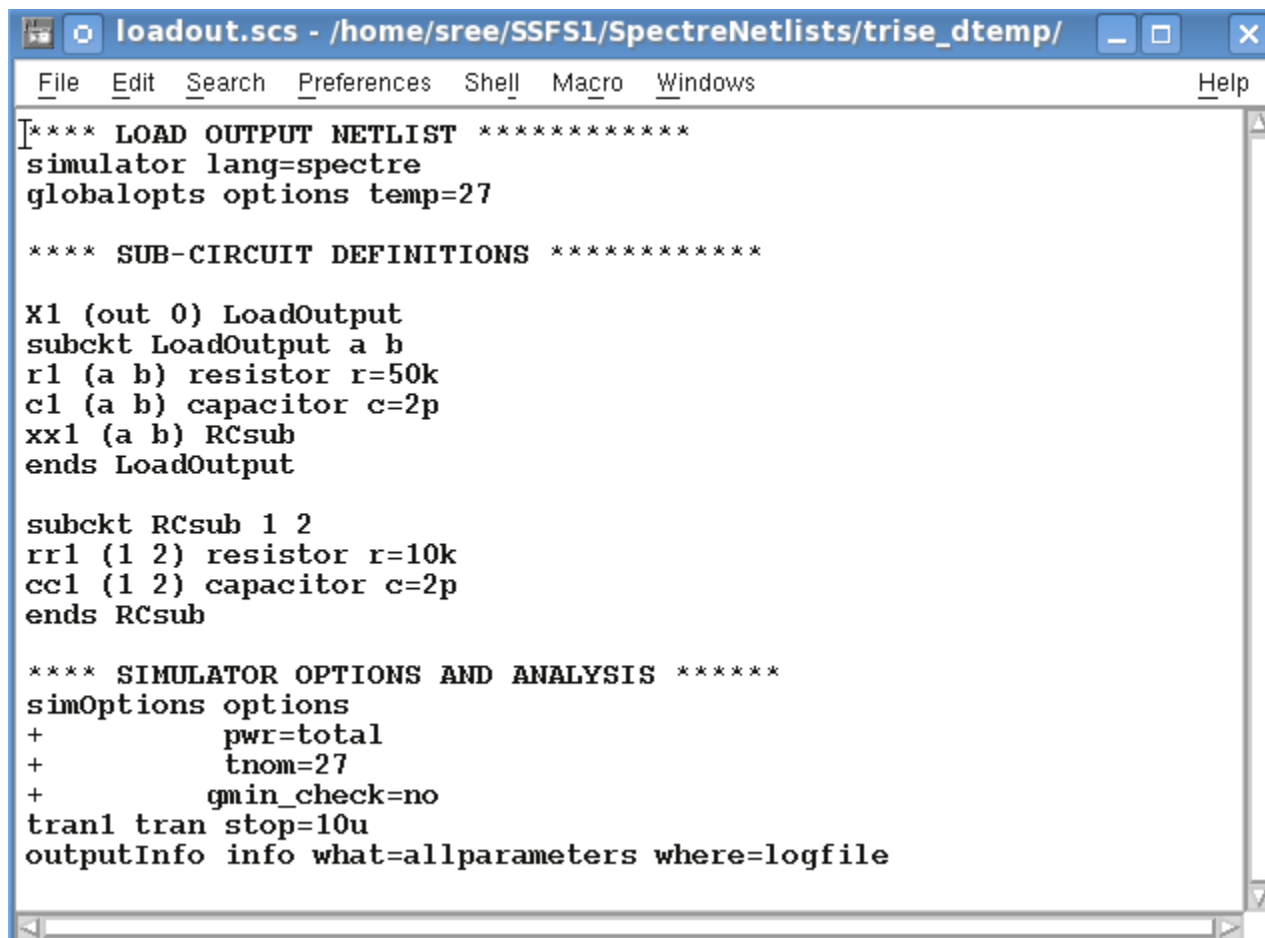
### Support of TRISE/DTEMP and TEMP for Local Blocks

1. Move to the `~SSFS1/SpectreNetlists/trise_dtemp` directory to perform this lab exercise.

```
cd ~SSFS1/SpectreNetlists/trise_dtemp
```

2. Open and view the `loadout.scs` netlist file using any of the text editors:

```
vi loadout.scs
```



```
loadout.scs - /home/sree/SSFS1/SpectreNetlists/trise_dtemp/
File Edit Search Preferences Shell Macro Windows Help
[**** LOAD OUTPUT NETLIST ****]
simulator lang=spectre
globalopts options temp=27

**** SUB-CIRCUIT DEFINITIONS ****

X1 (out 0) LoadOutput
subckt LoadOutput a b
r1 (a b) resistor r=50k
c1 (a b) capacitor c=2p
xx1 (a b) RCsub
ends LoadOutput

subckt RCsub 1 2
rr1 (1 2) resistor r=10k
cc1 (1 2) capacitor c=2p
ends RCsub

**** SIMULATOR OPTIONS AND ANALYSIS ****
simOptions options
+      pwr=total
+      tnom=27
+      gmin_check=no
tran1 tran stop=10u
outputInfo info what=allparameters where=logfile
```

Notice that the netlist has a global options statement with *temp=27* defined and then you have two subcircuit definitions *RCsub* and *LoadOutput* in this netlist with instance statements *xx1* and *X1* for them.

Also, note at the end of the file are the simulator options and analysis statements.

3. Spectre has been enhanced to enable to directly use the temperature rise (*trise/dtemp*) parameter as subcircuit instance parameter. So, modify the netlist **loadout.scs** with following actions to examine the TRISE effects.

- a. Add **trise=43** to the *X1, LoadOutput* instance statement.

- b. Add **trise=5** to the *xx1, RCsub* instance statement.

4. Save and close the netlist **loadout.scs** after adding these contents.

5. Run the Spectre simulation for the *loadout.scs* netlist.

6. Check your *loadout.log* file and note down the effective temperature for each of these instances.

*The effective temperature (tempeff) of LoadOutput Instance X1 is \_\_\_\_\_.*

*The effective temperature (tempeff) of RCsub Instance X1.xx1 is \_\_\_\_\_.*

7. Modify the netlist **loadout.scs** once again with following actions to examine the TEMP effects.

- a. Add **temp=43** to the *X1, LoadOutput* instance statement.

- b. Add **temp=5** to the *xx1, RCsub* instance statement.

8. Rerun the Spectre simulation on the *loadout.scs* netlist and note down the effective temperature of this run from *loadout.log* file.

*The effective temperature (tempeff) of LoadOutput Instance X1 is \_\_\_\_\_.*

*The effective temperature (tempeff) of RCsub Instance X1.xx1 is \_\_\_\_\_.*

9. Modify the netlist **loadout.scs** once again with following actions to examine both the TRISE/DTEMP and TEMP effects.

- a. Add **temp=100** to the *X1, LoadOutput* instance statement.

- b. Add **trise=25** to the *xx1, RCsub* instance statement.

10. Rerun the Spectre simulation on the *loadout.scs* netlist and note down the effective temperature of this run from the *loadout.log* file.

*The effective temperature (tempeff) of LoadOutput Instance X1 is \_\_\_\_\_.*

*The effective temperature (tempeff) of RCsub Instance X1.xx1 is \_\_\_\_\_.*

11. Close all the files and proceed to the next lab session.

**Note:** Solutions for this lab are in the *~SSFS1/SpectreNetlists/trise\_dtemp/.solutions* directory.







## **Module 4: Spectre Command-Line Interface**



## Lab 4-1 Controlling the Spectre Outputs and Protecting Proprietary Information

**Objective:** To explore the Spectre® options for specifying output formats and encrypting a netlist to protect its proprietary information.

---

This lab takes you through the process of running a Spectre simulation using an external configuration file containing the simulator settings, controlling the output formats using the environment setting and **spectre** command at run time and encrypting a netlist to protect the proprietary parameters, sub-circuits, models, and netlists, and release your libraries to your customers without revealing sensitive information.

### Using a Simulation Configuration File to Run a Spectre Simulation

In this section, you run a default Spectre simulation using an external configuration file containing the simulator settings.

1. In a terminal window, enter the following line and move to the respective directory:

```
cd ~SSFS1/SpectreCmdLine/output_format
```

2. Before you start with the lab session, run the clean script in that directory:

```
./clean
```

This clears the *log*, *raw*, and other result files from possible previous simulations and ensures that you obtain fresh results while doing the lab.

3. Open and view the netlist by entering the following command from your lab directory:

```
vi inverter_ring.scs
```

This is a ring oscillator design for inverter ring with no fanouts. The model definitions are available in the *mymodel.scs* file included in this netlist.

**Note:** The *.scs* file extension directs the Spectre parser to expect the native Spectre syntax. You can use any editor that you are comfortable with to open the inverter netlist. An alternative to using *.scs* is to make this statement the first line in the netlist: `simulator lang=spectre`.

Notice that the *global* statement includes three nodes: **0**, **vdd**, and **vss**. Note that these nodes can be used throughout the circuit hierarchy, without needing to pass the signals into subcircuits.

4. Close the **inverter\_ring.scs** file that you had opened for viewing.

5. Examine the configuration file *cfg*:

```
cat cfg
```

This configuration file *cfg* has the voltage source settings, simulator option settings, and the analysis statements.

6. Run the Spectre simulation enabling the configuration file during the runtime:

```
spectre +config cfg inverter_ring.scs
```

The two output files generated during this Spectre run are *inverter\_ring.log* and *inverter\_ring.raw* in the default *psfbin* format.

7. Open the output log file **inverter\_ring.log** and search for the following notice:

```
Notice from spectre during circuit read-in.
```

```
Configuration file used: cfg
```

This notice confirms that the simulation has used the *cfg* file present in the current working directory.

**Note:** If the configuration file is any other path, then give the complete path of the file and run the simulation.

The other notice is related to the Spectre initial condition statements defined for the five nodes. So you can ignore that notice.

8. Close all the files.

## Using the Environment Default Setting

The output formats supported by the Spectre simulator can be controlled using one of the following options:

- ◆ The environment default setting
- ◆ The **options** statement in the netlist to override an environment default setting
- ◆ The **spectre** command at run time to override any settings in the netlist

1. Set the following command-line options in the environment variable *SPECTRE\_DEFAULTS*.

```
setenv SPECTRE_DEFAULTS "+l %C:r.out -f psfascii"
```

What does each of the two options do?

*+l %C:r.out*

Answer: \_\_\_\_\_

*-f psfascii*

Answer: \_\_\_\_\_

2. Run a Spectre simulation on the *inverter\_ring.scs* netlist file and create a different output raw directory named *inverter\_ring.ascii*.

```
spectre +config cfg inverter_ring.scs -raw inverter_ring.ascii
```

The two output files generated during this Spectre run are *inverter\_ring.out* and *inverter\_ring.ascii*, in the *psfascii* format specified as an environment default setting.

3. Locate the ASCII file (*inverter\_ring.ascii/tran1.tran*) and find the approximate value for the following nodes at these (approximate) time points:

time=0 node "1": \_\_\_\_\_

time=3.207e-12 node "3": \_\_\_\_\_

time=5.757e-12 node "9": \_\_\_\_\_

4. Close all the files.

## Overriding Using the *spectre* Command at Run Time

1. Rerun a Spectre simulation on the *inverter\_ring.scs* netlist file, creating a different output raw directory named *mydata.raw* in *psfbin* format and a log file named *myout.log*.

```
spectre +config cfg inverter_ring.scs -format psfbin -raw mydata.raw
+log myout.log
```

This setting in the command line overrides any settings in the netlist or environment default setting.

2. Now, you can observe a new set of outputs *mydata.raw* and *myout.log*. Compare them with previous results and identify what's happening with respect to different output formats.

## Creating an Encrypted Netlist

In this lab section, you protect your proprietary parameters, subcircuits, models, and netlists with the encrypt feature in the Spectre simulator.

1. Open the **inverter\_ring.scs** netlist and add a **protect** at the beginning of the INVERTER sub-circuit and **unprotect** at the end of the sub-circuit definition.

**Tip:** Refer to the lecture material for any help in this step.

2. Save and close the netlist after adding these contents.
3. Create an encrypted netlist (*ringinv\_encrypt.scs*) from the *inverter\_ring.scs* netlist by using the following command providing appropriate files.

```
spectre_encrypt -i inverter_ring.scs -o ringinv_encrypt.scs
```

4. Open and view the encrypted netlist *ringinv\_encrypt.scs*. Notice that the INVERTER sub-circuit definition is encrypted using *pragma* statements. Also, only the content inside the *protect/unprotect* block is encrypted and the content outside this block remains the same as the original netlist. Close the encrypted netlist file after viewing its contents.
5. Create another encrypted netlist (*ringinv\_encryptall.scs*) using the appropriate command to encrypt the **entire netlist**, ignoring the *protect* and *unprotect* keywords from the *inverter\_ring.scs* file.

```
spectre_encrypt -i inverter_ring.scs -o ringinv_encryptall.scs -all
```

6. Now, open and view the encrypted netlist **ringinv\_encryptall.scs**. Notice that the entire netlist is encrypted ignoring the *protect* and *unprotect* keywords. This netlist can now be used to perform Spectre simulations without disclosing the proprietary information.
7. Close all the files and proceed to the next lab exercise.

**Note:** Solutions for this lab are in `~SSFS1/SpectreCmdLine/output_format/.solutions` directory.



## Lab 4-2 Exploring SPICE Compatibility

**Objective:** To examine the SPICE netlist compatibility and usage of the **+spice** Option in the Spectre command line.

---

The Spectre circuit simulator also provides SPICE netlist compatibility, eliminating the need for the Spice Pre-Parser (SPP) in your flow.

### Performing a Spectre Simulation on a SPICE Netlist

1. In a terminal window, change to the following directory:

```
cd ~SSFS1/SpectreCmdLine/spice_comp
```

2. Set the following command-line options in the environment variable **SPECTRE\_DEFAULTS**:

```
setenv SPECTRE_DEFAULTS "+l %C:r.log -f psfbin"
```

3. Open the **ampdata.sp** SPICE netlist file with any of the text editors:

```
vi ampdata.sp
```

This is a simple amplifier subcircuit with a top-level testbench. Notice that the complete netlist is in SPICE syntax. Toward the end of the file, you can see the .DATA setup with *sweep* statements for *temp* and *vdd*.

4. Close the netlist file after viewing its contents.

5. Run a Spectre simulation on the *ampdata.sp* SPICE netlist file:

```
spectre ampdata.sp
```

The two output files generated during this Spectre run are *ampdata.log* and *ampdata.raw*, in the *psfbin* format.

6. View the output of this simulation and answer the following questions:

*What type of simulations were run with the .DATA option?*

Answer: \_\_\_\_\_

*What parameters were swept?*

Answer: \_\_\_\_\_

*How many sweeps were run?*

Answer: \_\_\_\_\_

*What is the sweep called in the Results Browser?*

Answer: \_\_\_\_\_

*How much does the output vary over the sweep range?*

Answer: \_\_\_\_\_

*What is the nominal temperature during DC sweep?*

Answer: \_\_\_\_\_

## Enabling the +spice Option

1. Rerun the Spectre simulation on the **ampdata.sp** SPICE netlist file enabling the +spice option. This time, create different output files named **ampdata\_spice.log** and **ampdata\_spice.raw**.

```
spectre +spice ampdata.sp +log ampdata_spice.log -raw ampdata_spice.raw
```

The two output files generated during this Spectre run are *ampdata\_spice.log* and *ampdata\_spice.raw*.

2. Review the **ampdata\_spice.log** file and identify the two notices:

```
Notice from spectre at iteration_sweep1 = 2 during IC analysis `sweep1-001_srcSweep', during Sweep analysis `sweep1'.
```

```
Bad pivoting is found during DC analysis. Option dc_pivot_check=yes is recommended for possible improvement of convergence.
```

```
.....
```

```
Notice from spectre at iteration_sweep1 = 3 during IC analysis `sweep1-002_srcSweep', during Sweep analysis `sweep1'.
```

```
Bad pivoting is found during DC analysis. Option dc_pivot_check=yes is recommended for possible improvement of convergence.
```

*Why are these notices getting generated with +spice option enabled?*

Answer: \_\_\_\_\_

3. Compare the two log files *ampdata.log* and *ampdata\_spice.log* and answer the following.

*What is the nominal temperature in ampdata\_spice.log and why is this?*

Answer: \_\_\_\_\_

4. After analyzing the results, close all the files and proceed to the next lab exercise.





## **Module 5: Spectre Analog Simulations in the ADE Explorer**



## Lab 5-1 Setting Up and Running Spectre Simulation in ADE Explorer.

**Objective:** To set up and run Spectre® Simulations in ADE Explorer.

---

This lab takes you through the process of setting up and running a Spectre simulation in the ADE Explorer Environment. In this section, you will:

- ◆ Open the *maestro* view in Virtuoso® ADE Explorer
- ◆ Set up the Simulator (spectre) for simulation
- ◆ Include model files
- ◆ Choose Analyses
- ◆ Set design variables
- ◆ Save and plot simulation outputs
- ◆ Create a netlist and run the simulation

### Starting the Cadence® Environment

1. Change to the working directory in a terminal window by entering:

```
cd ~SSFS1/ADEExplorerLab
```

2. Run the clean script to clean any stale files or results and start the labs:

```
./clean
```

3. Start the Virtuoso software by entering:

```
virtuoso &
```

The Virtuoso environment starts, the Command Interpreter Window (CIW) opens, and the Library Manager appears. You can close the What's New window. The Library Manager also opens at startup because the *ddsOpenLibManager()* SKILL® function is used in the *.cdsinit* file.

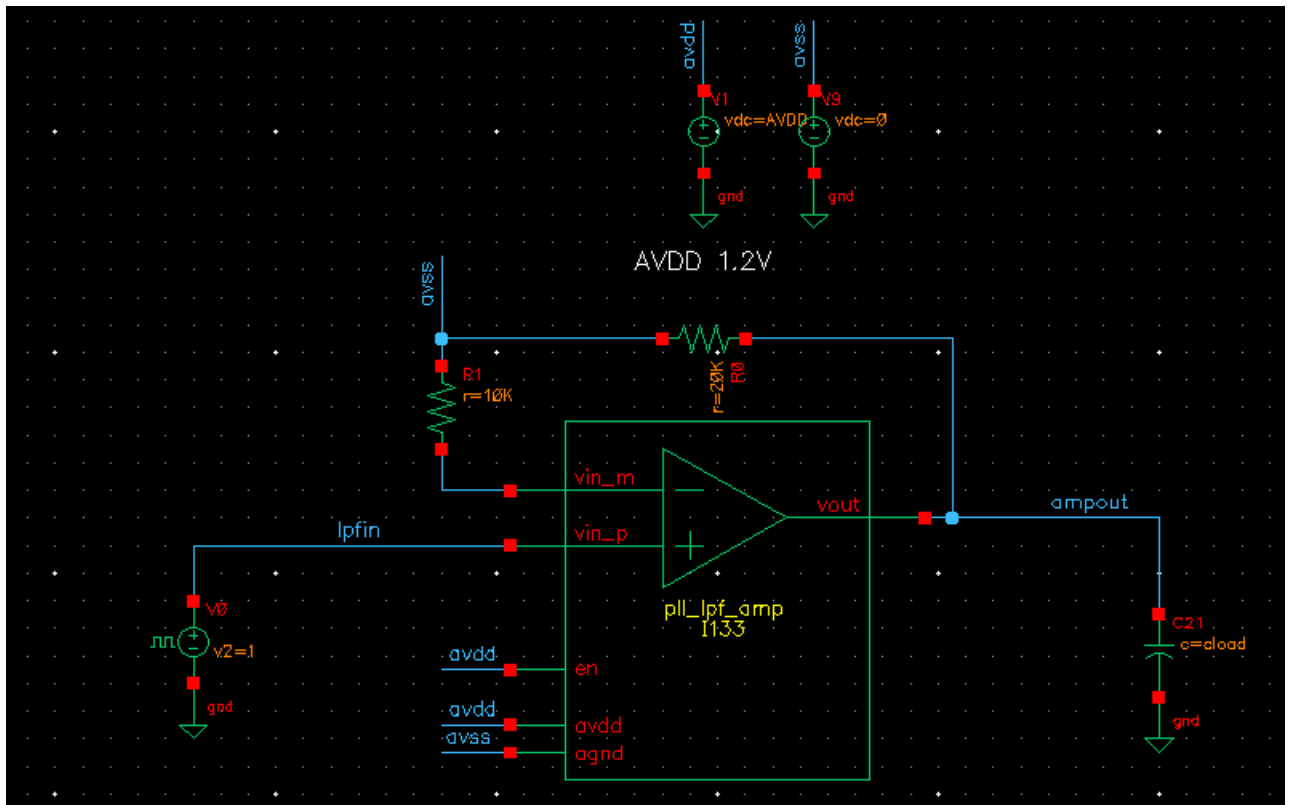
## Examining the Amplifier Design and Testbench for LPF

1. In the Library Manager, open the following design schematic:

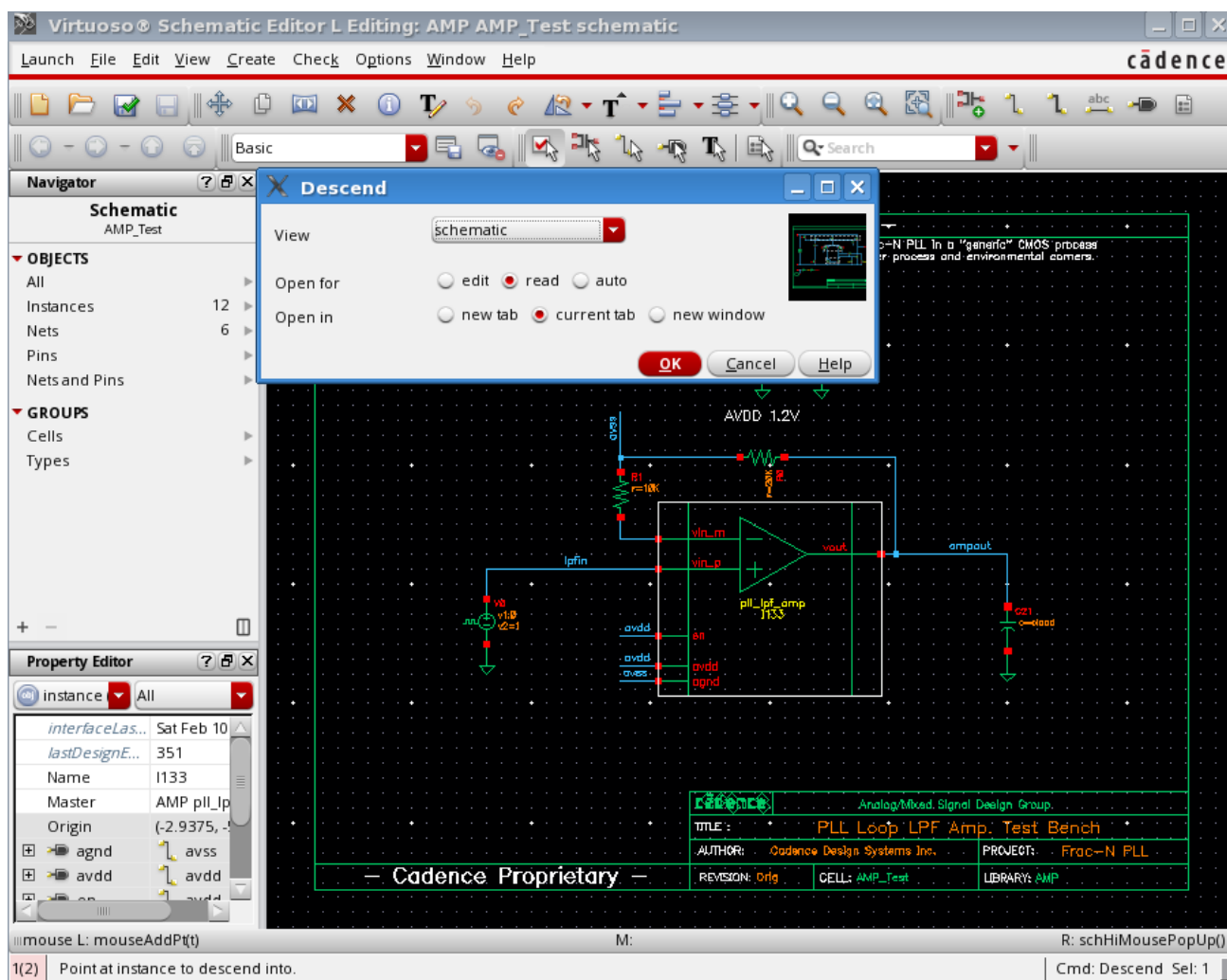
<b>Library</b>	AMP
<b>Cell</b>	AMP_Test
<b>View</b>	schematic

The design is a simple amplifier testbench used for low pass filter circuits in PLL. Notice the two assistants, **Navigator** and **Property Editor**, docked to the left of the Schematic Editor.

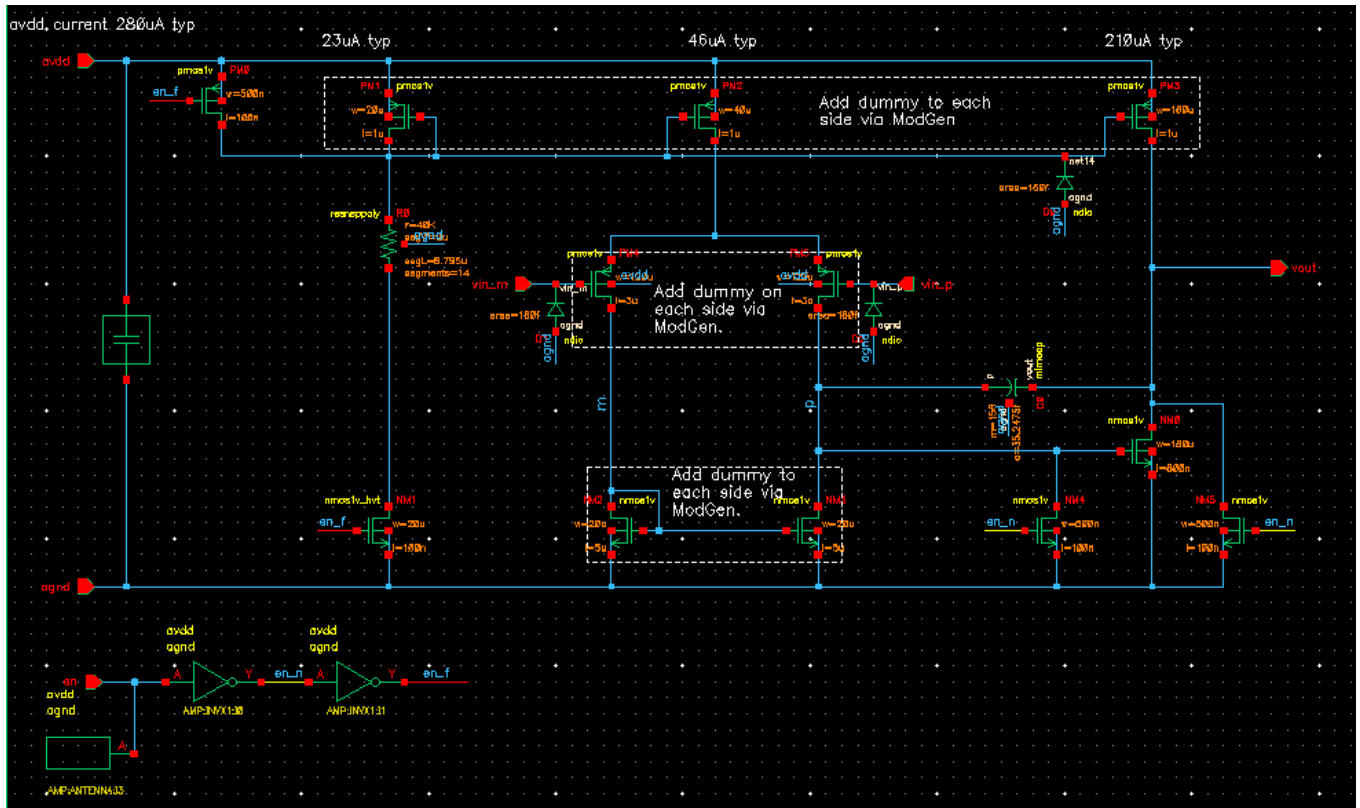
In this top-level testbench, you have the *pll\_lpf\_amp* I133 instance. The voltage supplied at the *vin\_put* terminal (**vin\_p**) is a PWL source that changes between a high (*val1*=1.0V) and low (*val0*=0V) value with timing characteristics (*period*=5u) defined.



- To examine the general circuit architecture, select the **pll\_lpf\_amp** I133 instance and **descend** into the schematic by pressing the Bindkey “E” or “Shift + E” and open the design in the **current** tab.



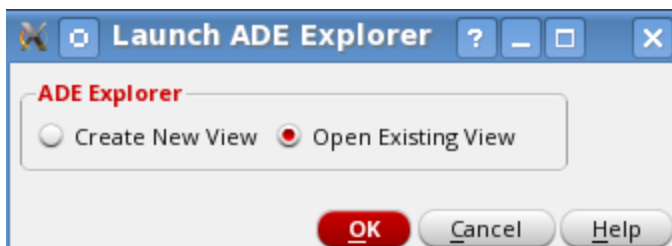
- Examine the **pll\_lpf\_amp** schematic design with MOS devices for biasing and output and have their model names displayed. In addition, the MOS devices have sizing information displayed.



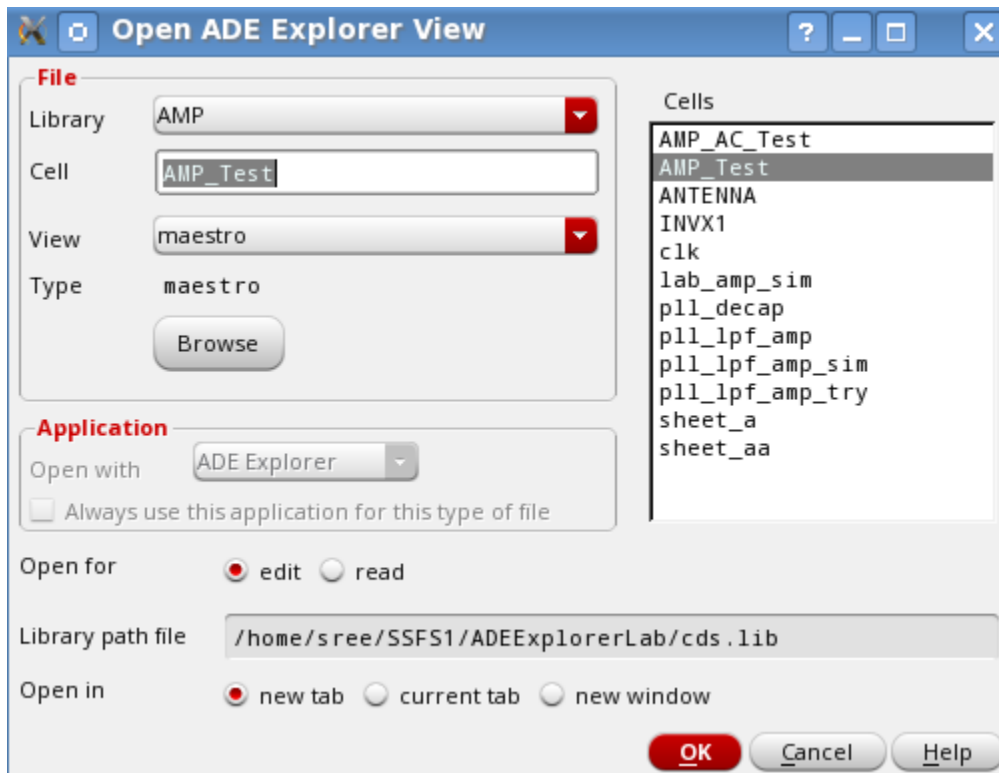
- After examining the schematic, return to the top of the testbench *AMP\_Test* in the Schematic Editor by choosing **Edit – Hierarchy – Return to Top**.

## Opening the *maestro* View in Virtuoso ADE Explorer

- In the *AMP\_Test* schematic window, choose **Launch – ADE Explorer** and choose **Open Existing View**.



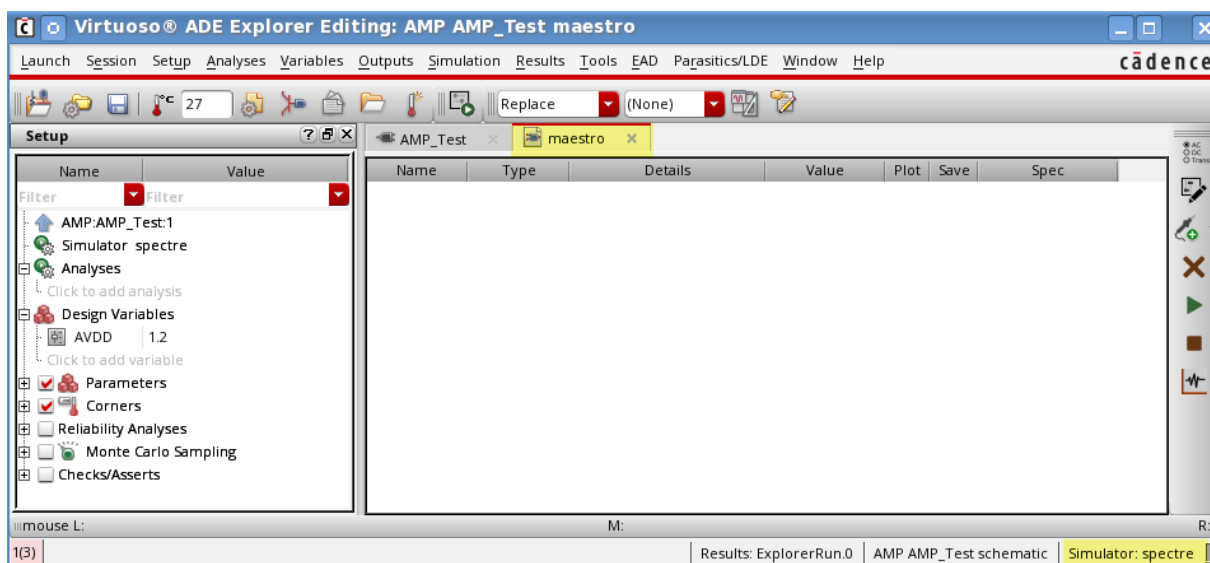
- Then in the below Open ADE Explorer View window, select the **maestro** view of the *AMP\_Test* cell from the *AMP* library and click **OK** to open the ADE Explorer in the **Edit** mode in a **new** tab.



**Tip:** Alternatively, open the existing **maestro** view directly from the Library Manager.

<b>Library</b>	AMP
<b>Cell</b>	AMP_Test
<b>View</b>	maestro

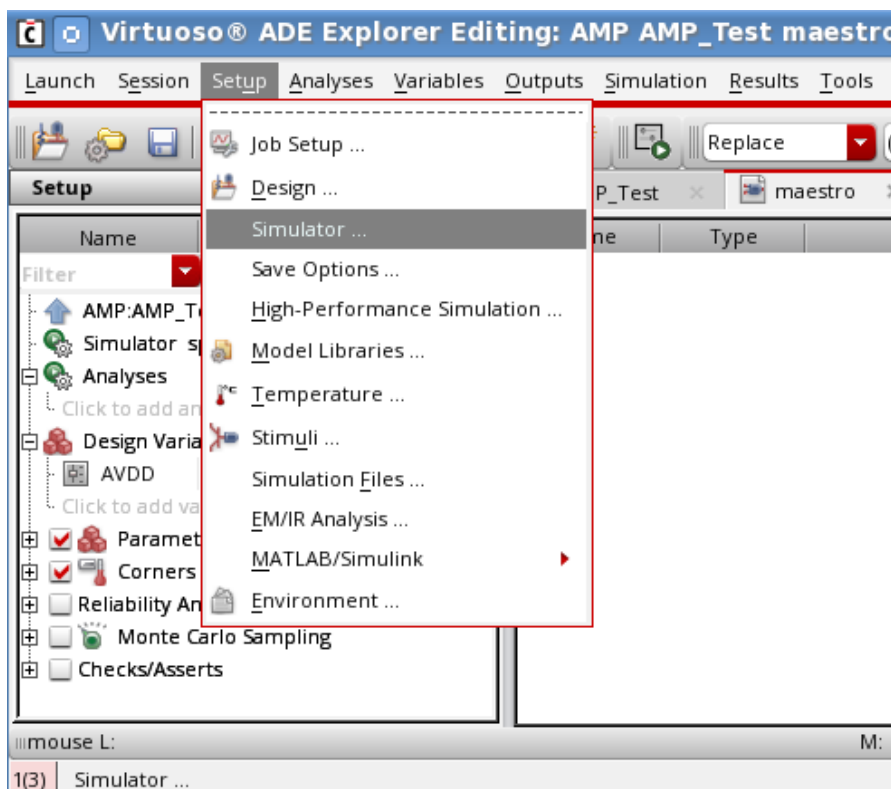
**Note:** You can open a design in multiple ways: **double**-clicking the **maestro** view, **right**-clicking the **maestro** view and selecting **Open** from the pop-up menu, or choosing **Open** from the Library Manager or CIW file menu.



## Choosing a Simulator

You now see how to set up and use the Spectre Circuit Simulator with the *AMP\_Test* design, which is made up of analog components.

1. In the ADE Explorer status bar, verify that the Simulator is set to **spectre**. If it is not, then choose **Setup – Simulator** from the menu bar, change the simulator to **spectre** and click **OK** to accept your setup.





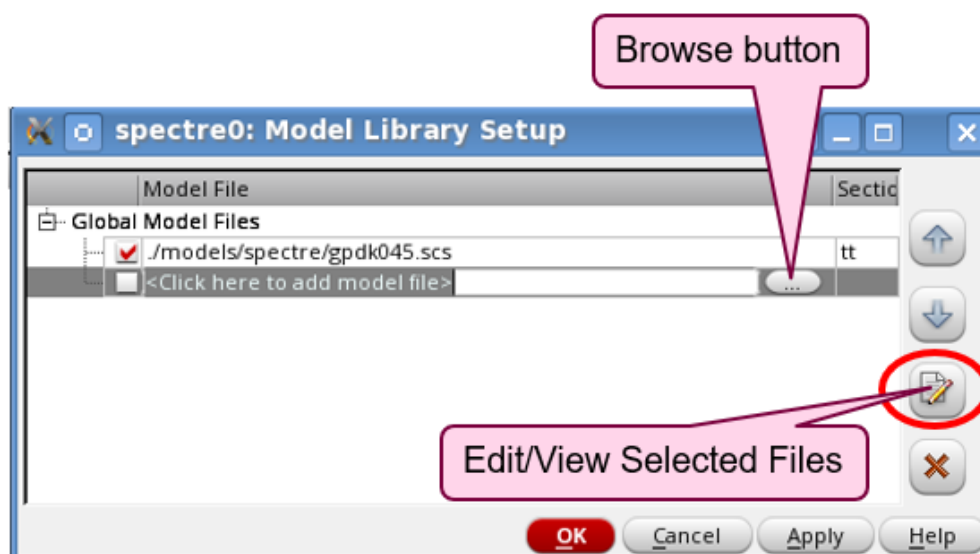
**Note:** The simulator should already be set to **spectre** by the *.cdsinit* file, which was read when invoking *virtuoso*.

## Setting the Model Libraries

The model library file contains the model files that describe the *nmos1v*, *pmos1v* and other devices during simulation.

1. Choose **Setup – Model Libraries**.


The Model Library Setup form appears. Notice that you have a *gpd045.scs* model file with the typical (**tt**) section. If not, select the model file from appropriate path using the **Browse (...)** button.



2. To view the model file, select the model file. Click the **Edit/View Selected Files** button.
3. With the Model Library Setup form, you can include multiple models. Once you are done, click **OK** or **Cancel** in this form.

## Choosing Analyses

This part of the lab demonstrates how to set up a transient analyses to complete your circuit during the simulation. You select and run multiple analyses on the *AMP\_Test* design.

1. In the Simulation window, choose **Analyses – Choose** or click  the **Choose Analysis** icon.

The Choosing Analyses form appears.

Take a few moments to become familiar with this form because it is used often throughout the lab activities. It is also a dynamic form: the layout changes based on selected analyses.

2. Set up a Transient analysis.

a. In the Analysis section, select **tran**.

b. Set the Stop Time field to **10u**.

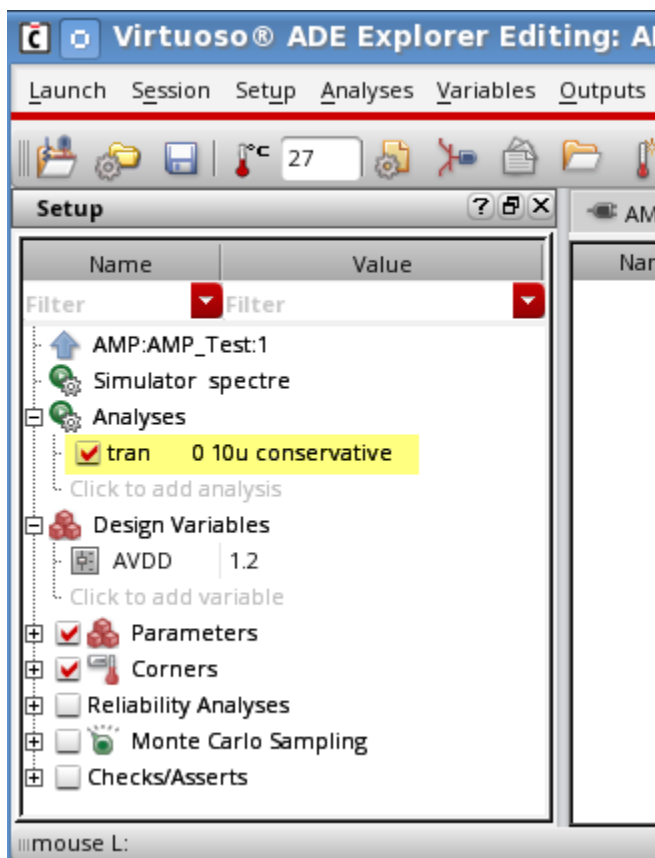
c. Set *errpreset* to **Conservative**.

**Tip:** **Moderate** is the default. For more accurate but speed-compensated solutions, choose conservative.

d. Select the **Enabled** check box and ensure the form is as shown below:

The screenshot shows the 'Choosing Analyses -- ADE Explorer' dialog box. The 'Analysis' section contains a grid of radio buttons for various analysis types: tran (selected), dc, ac, noise, xf, sens, dcmatch, acmatch, stb, pz, lf, sp, envlp, pss, pac, pstb, pnoise, pxf, psp, qpss, qpac, qpnoise, qpxf, qpsp, hb, hbac, hbstb, hbnoise, hbsp, and hbxf. Below this, the 'Transient Analysis' section is expanded, showing a 'Stop Time' field set to '10u'. Under 'Accuracy Defaults (errpreset)', the 'conservative' checkbox is checked, while 'moderate' and 'liberal' are unchecked. The 'Transient Noise' checkbox is unchecked. The 'Dynamic Parameter' checkbox is also unchecked. At the bottom left, the 'Enabled' checkbox is checked. An 'Options...' button is located to the right of the 'Enabled' checkbox. At the very bottom, there are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

- e. Click **Apply** and view the ADE Explorer window. The *tran* analysis is added to the Analyses section of the Setup assistant.



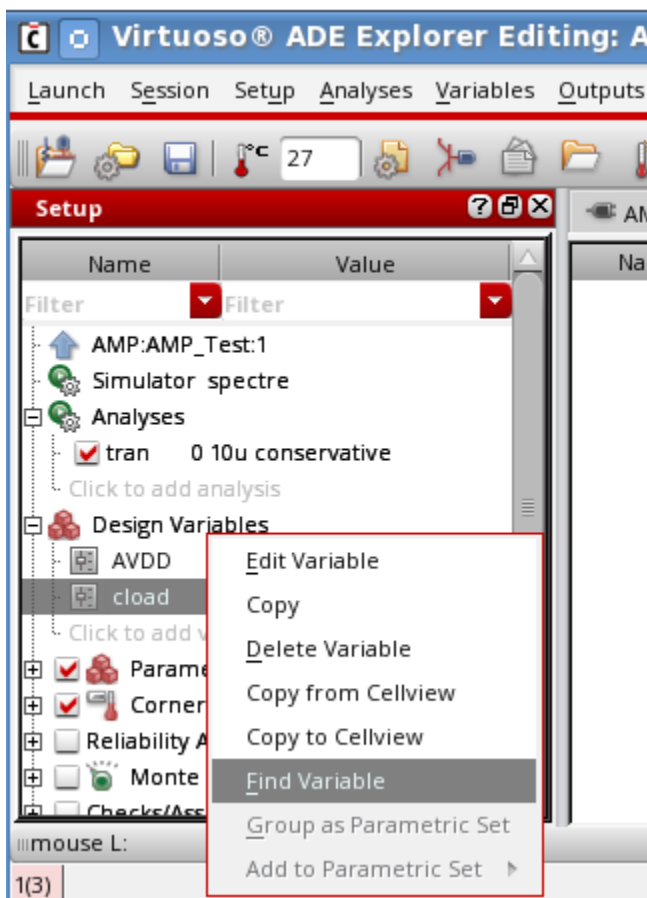
## Setting Design Variables

The values of any design variables in the circuit must be set before running a simulation. If values have been set in the past, they appear in the Value column of the Design Variables section of the Setup assistant in the ADE Explorer form.

1. To retrieve any of the variables in the hierarchy, choose **Variables – Copy from Cellview** on the ADE Explorer menu.

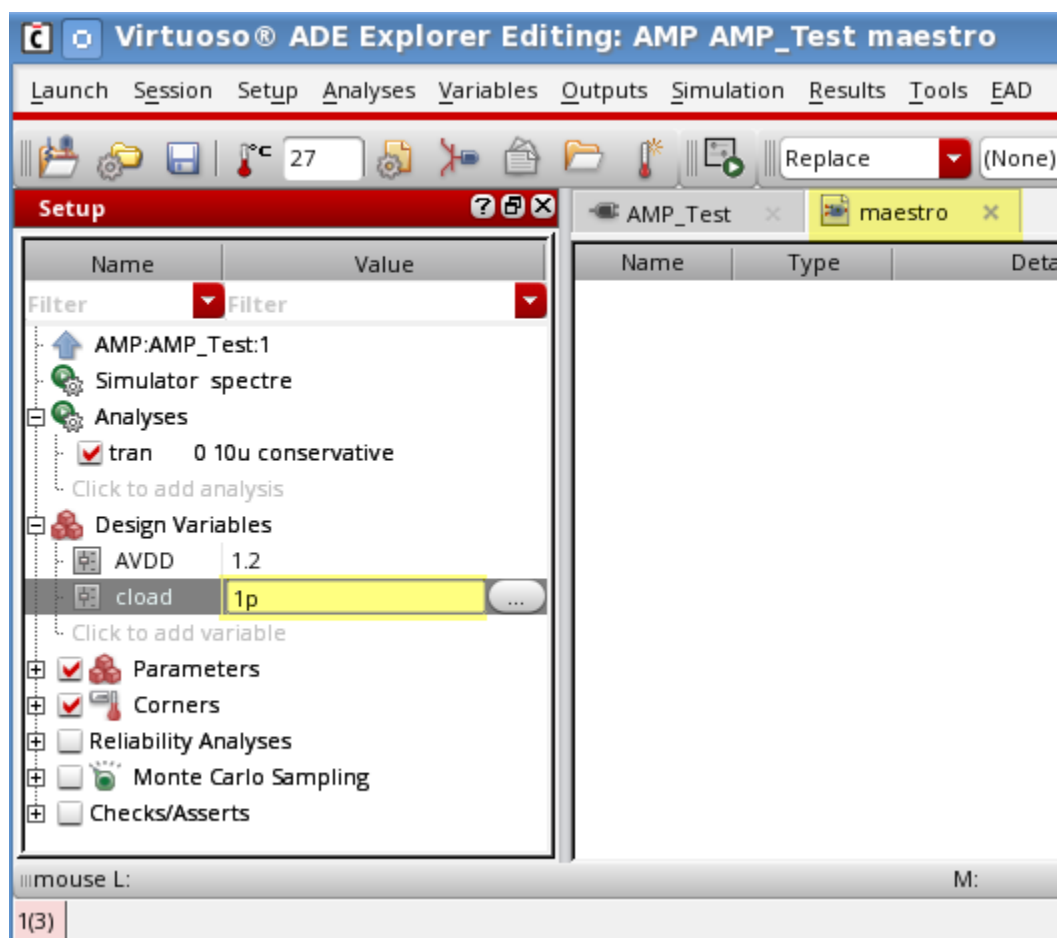
The *cloud* variable appears in a table in the Design Variables section.

- To locate the *CAP* variable in the *AMP\_Test* design, **right-click** the *CAP* name and select **Find Variable**.



The Find Variable operation takes you to the *AMP\_Test* schematic tab, descends into the hierarchy and zooms to highlight the *capacitor* instance.

- a. Move to the **maestro** tab, change the value of the *cload* variable to **1pF** by clicking the *Value* field after the variable *cload* and entering the new value. Press **Enter** to accept the change.



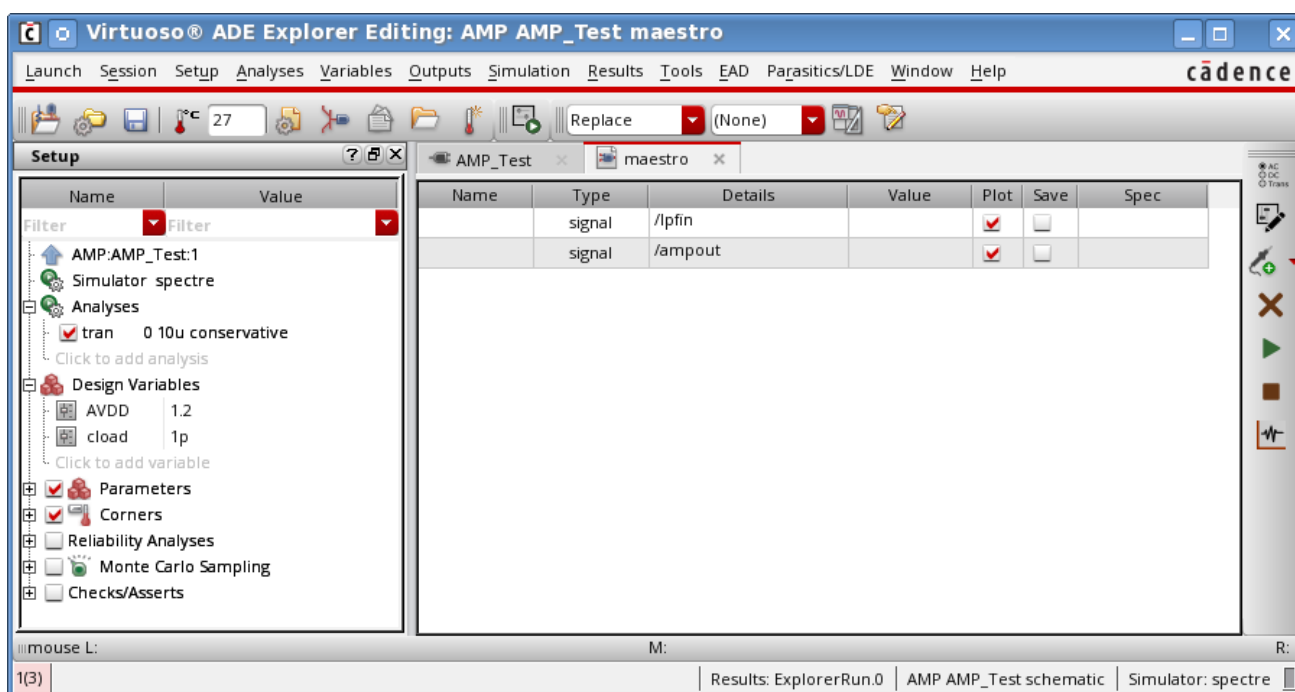
## Saving and Plotting Simulation Outputs

The simulation environment is configured to save all public node voltages by default. It also saves currents you select for plotting and saving. You can modify the default setup to save all terminal currents as well.

In larger designs, where saving all of the data requires too much disk space, you can select just a specific set of nodes to save. In this exercise, you select two terminals for which the signal current is saved.

1. In the Simulation window, choose **Outputs – Save All** to open the Save Options form.
  - a. In the *Select signals to output (save)* section of the Save Options form, select **allpub** if it is not already selected.
  - b. At the bottom of the form, ensure that the *Output Format* is **psfxl**.

- c. Click **OK** in the Save Options form.
2. To select specific signals for saving and plotting, choose **Outputs – To Be Plotted – Select On Design** in the ADE Explorer window. This brings to the front the schematic window for your selections.
  - a. Select the **lpfin** and **ampout** signals for plotting. After selections, the wires gets highlighted.
  - b. Press **Esc** with your cursor in the schematic window to stop the selection process.
3. Move to the **maestro** tab and verify whether it looks similar to the one in the following snapshot:



## Creating the Netlist

The netlist is generated automatically when the simulation is started by clicking the **Run Simulation**

button  or by choosing **Simulation – Netlist and Run**. However, netlists can also be created separately, without running a simulation.

- ♦ **Simulation – Netlist – Create** generates an incremental netlist where only changes are modified. This is the same netlist that is created when the Netlist and Run button is pressed.
- ♦ **Simulation – Netlist – Recreate** generates a clean netlist starting over from the beginning, which is useful when schematic changes have been made.

- ◆ **Simulation – Netlist – Display** displays the completed netlist.
- ◆ **Simulation – Run** runs the simulation when the netlist has been created.


1. In ADE Explorer, choose **Simulation – Netlist – Create**.

A window appears that displays the netlist. Notice how all of the parameters, design variables, and simulation settings that you entered appear in the netlist. If there are any errors encountered during this step, check the messages in the CIW for clues to the problem areas.

This is the netlist and is called the *input.scs* file. It is located in your *~SSFS1/ADEExplorerLab/simulation/AMP/AMP\_Test/maestro/results/maestro/.tmpADEDir\_XXXX/AMP:AMP\_Test:1/AMP\_AMP\_Test\_schematic\_spectre/netlist* directory.

2. Choose **File – Close Window** in the netlist window to close it.

## Running the Simulation

1. Choose **Simulation – Netlist and Run** or click the **Run Simulation** button  to recreate the netlist and run the simulation.

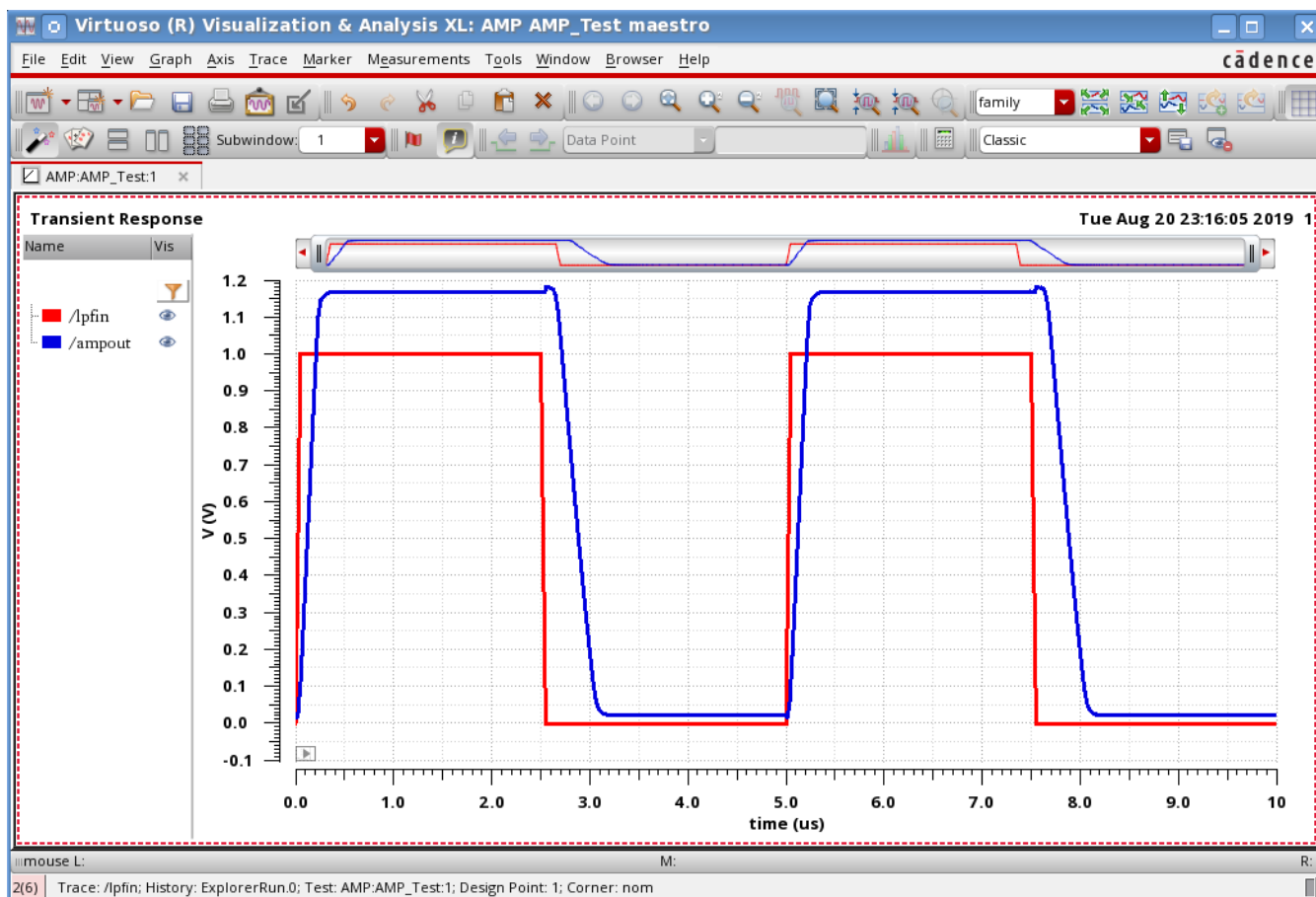
As an alternative to the previous step, you can choose **Simulation – Run** from the menu if you've already created your netlist.

A log file shows the progress of the simulation for all three analyses. This is called the *spectre.out* file and is stored in a hierarchical psf directory, inside your *./simulation* directory.

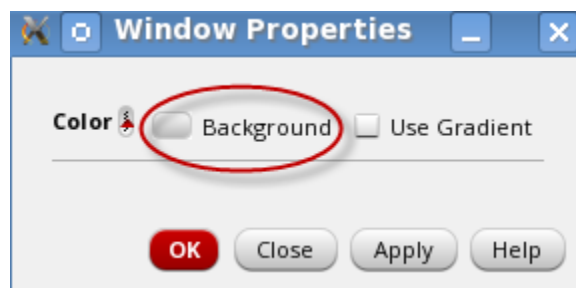
The log file also lists warnings and errors if there are problems in the input data, which often is missing model files.

**Tip:** If you close this log file, you can open it again by going to **Simulation – Output Log**.

2. When the simulation finishes, the transient analysis plots are plotted automatically for selected signals *lpfin* and *ampout* in ViVA Graph window with *Classic* workspace.



**Tip:** To modify the background color from black to white, choose **File – Window Properties** in the ViVA XL window and change the background color to white.



3. To close the log, choose **File – Close Window**.

Simulation results and related netlist information are stored in the *psf* and *netlist* directories, in a hierarchical structure inside your specified simulation directory.

Examples for the current run would be:

~SSFS1/ADEExplorerLab/simulation/AMP/AMP\_Test/maestro/results/maestro/Explorer Run.0/1/AMP:AMP\_Test:1/netlist/input.scs

~SSFS1/ADEExplorerLab/simulation/AMP/AMP\_Test/maestro/results/maestro/Explorer Run.0/1/AMP:AMP\_Test:1/psf/spectre.out



4. Close the ViVA Graph window after analyzing the results. Close all the windows except the schematic, ADE Explorer, Library Manager and CIW.

