Virtuoso Analog Design Environment: iModule 1 for iLS

Version 5.1.41

iLS LHS/TOC **February 10, 2005**

Schematic Entry and Simulation

cadence

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Cadence Trademarks

1st Silicon Success®
Allegro®
Assura™
BuildGates®
Cadence® (brand and logo)
CeltIC™
ClockStorm®
CoBALT™
Conformal®
Connections®
Design Foundry®
Diva®
Dracula®
Encounter™
Fire & Ice®
First Encounter®
FormalCheck®
HDL-ICE®
Incisive™
IP Gallery™
Nano Encounter™
NanoRoute™
NC-Verilog®
OpenBook® online documentation library
Orcad®
Orcad Capture®
Orcad Layout®
PacifIC™
Palladium™
Pearl®
PowerSuite™
PSpice®
QPlace®
Quest®
SeismIC™
SignalStorm®
Silicon Design Chain™
Silicon Ensemble®
SoC Encounter™
SourceLink® online customer support
Spectre®
TtME®
UltraSim®
Verifault-XL®
Verilog®
Virtuoso®
VoltageStorm®

Other Trademarks

All other trademarks are the exclusive property of their respective owners.

Confidentiality Notice

Table of Contents
Virtuoso Analog Design Environment iModule 1

Virtuoso Analog Design Environment iModule 1: Schematic Entry and Simulation

Introduction to the Analog Design Environment, Version 5.1.41

Module 1

February 10, 2005

Topics in this Module

- Course objectives
- Course outline
- Class schedule
- Getting help, technical support, and documentation
- What's new in 5.1.41
- The Design Framework II design environment
- Accessing design tools
- Creating a library
- Creating cells and cell views
- Schematic capture
- Analog simulation
- Analyses
- Summary

Terms and Definitions

CDSDoc	Cadence® online help tool that uses a Netscape browser interface.
CIW	Command interpreter window. Interface used to access DFII applications.
command line	A line buffer in the CIW that accepts SKILL-based commands.
text field	An area on a tool window where the user provides data.
cyclic field	Set of selectable options in a tool window, denoted by a small rectangle.
library	A set of design directories that includes ‘cells’ and ‘cellviews’.
Library Manager	A Cadence tool that allows user to browse and edit a design library.
Virtuoso Schematic Editor	Schematic editor and symbol generation tool in DFII.
cell	A basic unit of a design hierarchy described by cell views.
cell view	A specific view of a cell that includes schematic, symbol, or layout.
instance	A uniquely named placement of a symbol onto a schematic.
pin	A connection point on a schematic and symbol used for accessing signals.
bindkey	A predefined key on the keyboard that invokes a preselected command, sometimes called an “accelerator”.

Course Objectives

- Learn how to create schematics, symbols, and a design hierarchy
- Set up and run analog simulations
- Analyze simulation results
- Evaluate sensitivities and mismatches to improve circuit performance.
- Run Corners, Monte Carlo, and Optimization tools to improve yield
- Create and use OCEAN scripts and SKILL to set up and run simulations
- Understand the Component Description Format (CDF)
- Create configurations with the Hierarchy Editor (HED)
- Use subcircuits and macromodels
- Run the parasitic simulation flow
- Use advanced tools to solve special problems

The objective of this class is to provide both instruction and materials on using the Virtuoso Analog Design Environment that will enable you to use the entire front-to-back design flow.

Getting Help

You can get help with Cadence software from the following sources:

- Help button on forms and windows
- Cadence online documentation (CDSDoc)
- Education Services training manuals
- SourceLink® online customer support
- Customer Response Center (CRC)

Online Help

Cadence reference manuals and online help files for each product are installed automatically when installing the product. Hard copies of the reference manuals are available from Cadence. All these online documents are part of the online help system, which can be accessed as follows:

- View relevant product information by clicking the help button on windows and forms. Use this information to complete a form or what can be done in the window.
- Start the CDSDoc documentation from a UNIX shell by typing `cdsdoc&` at the command line and search through all Cadence reference manuals and online help systems installed with each product. Also, use CDSDoc to print the reference manuals entirely or just the relevant material.

Other Means of Getting Information

- With a software maintenance agreement, subscribe to the SourceLink online support system and view known problems and solutions or communicate with other users. The SourceLink system is accessible via the internet. To open an account, send email to support@cadence.com.
- Training manuals, like this one, can supplement reference manuals.
- When the above information is insufficient, call the Customer Response Center.
  - North America 1(877)CDS-4911 or 1(877)237-4911

What’s New in 5.1.41

- Spectre device checking interface
- WaveScan integration
- UltraSim integration
- Third-party OASIS integration
- Parasitic resimulation flow

Spectre Device Checking:

This feature allows engineers to specify rules governing safe operating areas of devices on a primitive, a model, or an instance basis.

WaveScan Integration:

WaveScan is a new waveform viewing tool that is a drop-in replacement for the Analog Waveform Display (AWD) tool. WaveScan is set as the default display and provides performance and capability improvements like trace-legend visibility, X-axis parametric sweep swapping, and the ability to export graphs in **.bmp**, **tiff**, and **.png** formats.

UltraSim Integration:

UltraSim is now directly integrated into ADE. UltraSim is a hierarchical simulator designed for verification of analog and mixed-signal circuits. UltraSim is not covered in this course.

Third-Party OASIS integration:

OASIS has been updated to allow third-party simulators to take advantage of Corners capability.

Parametric Resimulation Flow:

Greatly enhanced in 5.1.41, the parametric resimulation flow has been rewritten to improve usability and limit previous restrictions concerning Assura. You may now backannotate both node voltages and device operating points onto schematics. New capability for parasitic resistance annotation has been added as well.

Overview of Virtuoso Analog Design Environment

The Virtuoso Analog Design Environment is a software tool set within Design Framework II that is used to set up and run analog simulations. The Virtuoso Analog Design Environment also accesses and views the simulation results.

The Virtuoso Analog Design Environment allows you to:

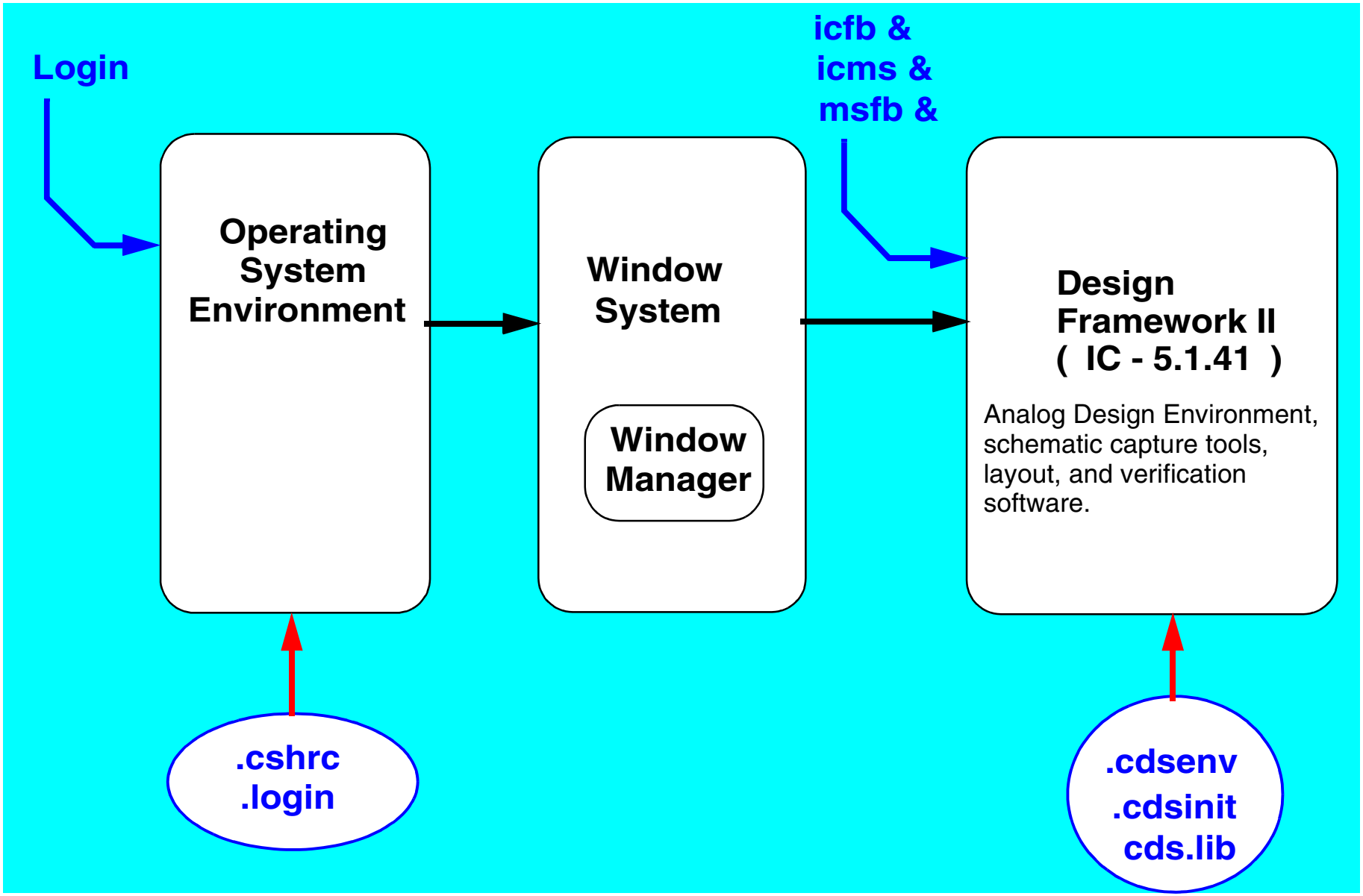
- Choose the simulator host
- Choose the type of analysis: ac, dc, transient, parametric, sensitivity, etc.
- Set design variables: Vdd, frequency, Cout, etc.
- Append model files and include files
- Netlist and run simulations
- Quickly alter the simulation setup and rerun the simulation
- Plot simulation results in the Waveform display tool
- Evaluate simulation results using waveform expressions
- Run multiple simulation tools: Corners, Monte Carlo, Optimizer, etc.
- Automatically set up, save, and run OCEAN scripts

The Virtuoso Analog Design Environment is a set of software design tools used to set up, control, and run circuit simulations. ADE allows you to choose the simulator host, set design variables, select model files, and to select analyses to add, modify, or delete from next simulation run.

The Virtuoso Analog Design Environment provides a user-friendly graphical interface that includes pull-down menus and icons for making fast and easy changes and also provides control for accessing the simulation results and displaying the results to the waveform display tool. The results can be entered into other tools for waveform processing or to obtain specific data using expressions.

The Virtuoso Analog Design Environment provides access to multiple simulation tools like Corners, Monte Carlo, and the Circuit Optimizer. ADE also allows you to automatically set up, save, and run OCEAN scripts.

Design System Initialization Files

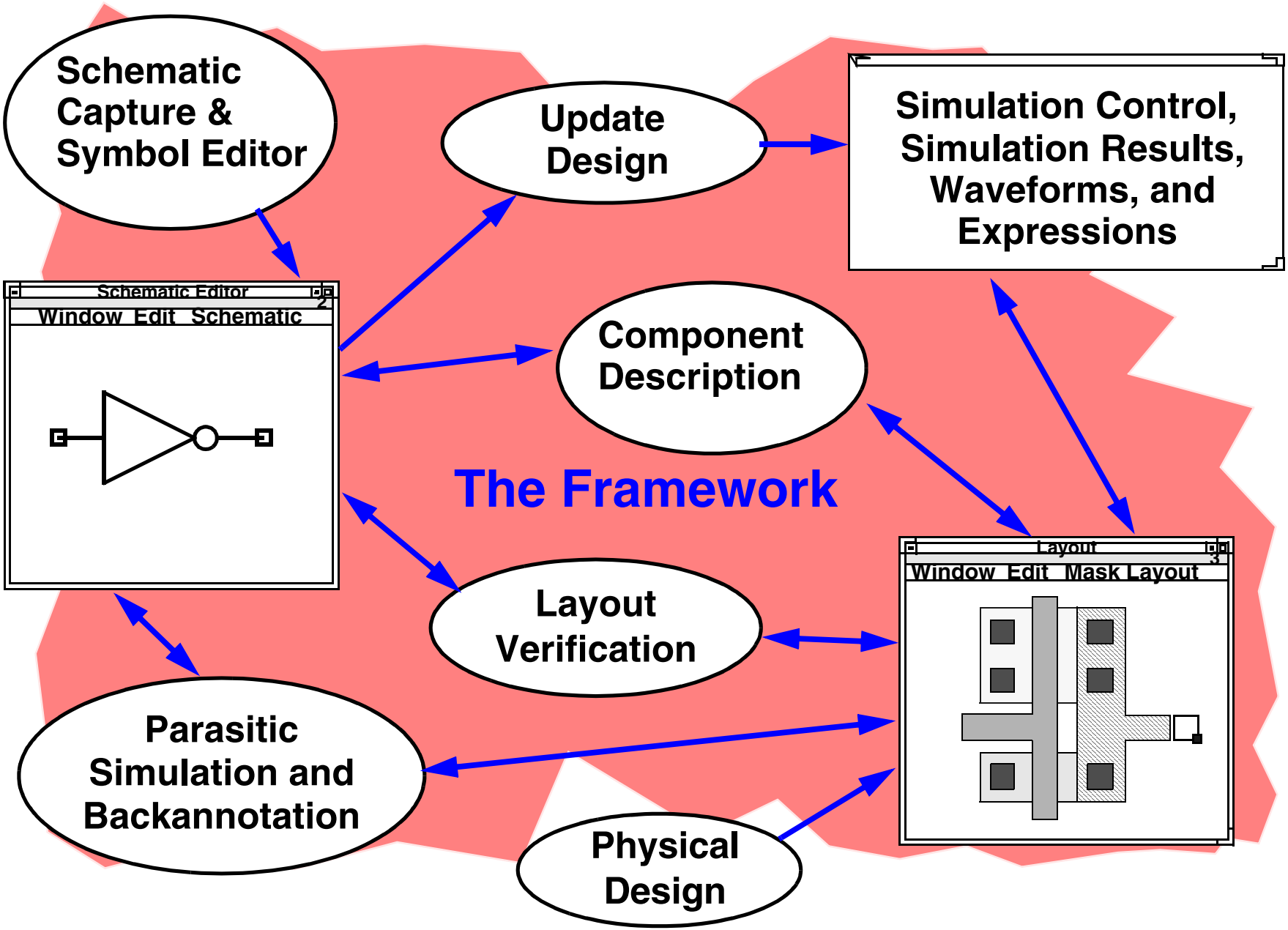


There are some design system initialization files that configure the operating system environment. For example, the *.cshrc* and *.login* files configure the UNIX environment when you log in and start a UNIX application.

The initialization file, *.cdsinit*, customizes the Virtuoso Analog Design Environment. The *cds.lib* file sets the paths to the libraries. These files, along with the *.cdsenv* file, are discussed later.

For more information on configuring your operating system environment for the Virtuoso Analog Design Environment, consult the *Cadence Design Framework II Configuration Information* guide.

Overview of the Design Framework II Environment



The Virtuoso Analog Design Environment is a set of design tools that operate within Design Framework II. Design Framework II is the underlying structure for Cadence design tools for schematic capture, analog simulation, and layout. It provides a single integrated environment for accessing all tools and design data, including the ability to:

- Access to the Command Interrupter Window (CIW) using ***icfb***, ***icms***, or ***msfb***.
- Use the Library Manager Tool to browse design libraries and open cell views.
- Create new libraries, cells, and cell views.
- Start or edit a schematic view or symbol view.
- Start or edit a layout design.
- Run layout verification.
- Start the Virtuoso Analog Design Environment and run simulations.
- Access simulation results directly using the Results Browser.
- Run OCEAN scripts.

Advantages of Using Design Framework II

- Common software environment for using schematic capture, simulation, layout, and design verification
- Easy-to-learn, consistent user interface
- Similar appearance between most forms and windows
- Communication between software tools within the DFII environment
- Tool windows remain open while running other applications
- Data can be “backannotated”
  - From layout to schematic
  - From simulation to schematic
  - From simulation to layout
- Applications may be customized or automated using SKILL or the OCEAN command language

The Cadence Design Framework II environment is an integrated design environment. An integrated environment means that numerous tools and applications operate together. For Design Framework II the environment provides schematic capture, simulation control, netlist generator, circuit simulator, waveform display, layout and verification tools.

For the design software tools, DFII provides:

- Consistent user interface

  Analog applications in the design framework have the same “look and feel.” Menu items are often in the same place in every application.

- Consistent database

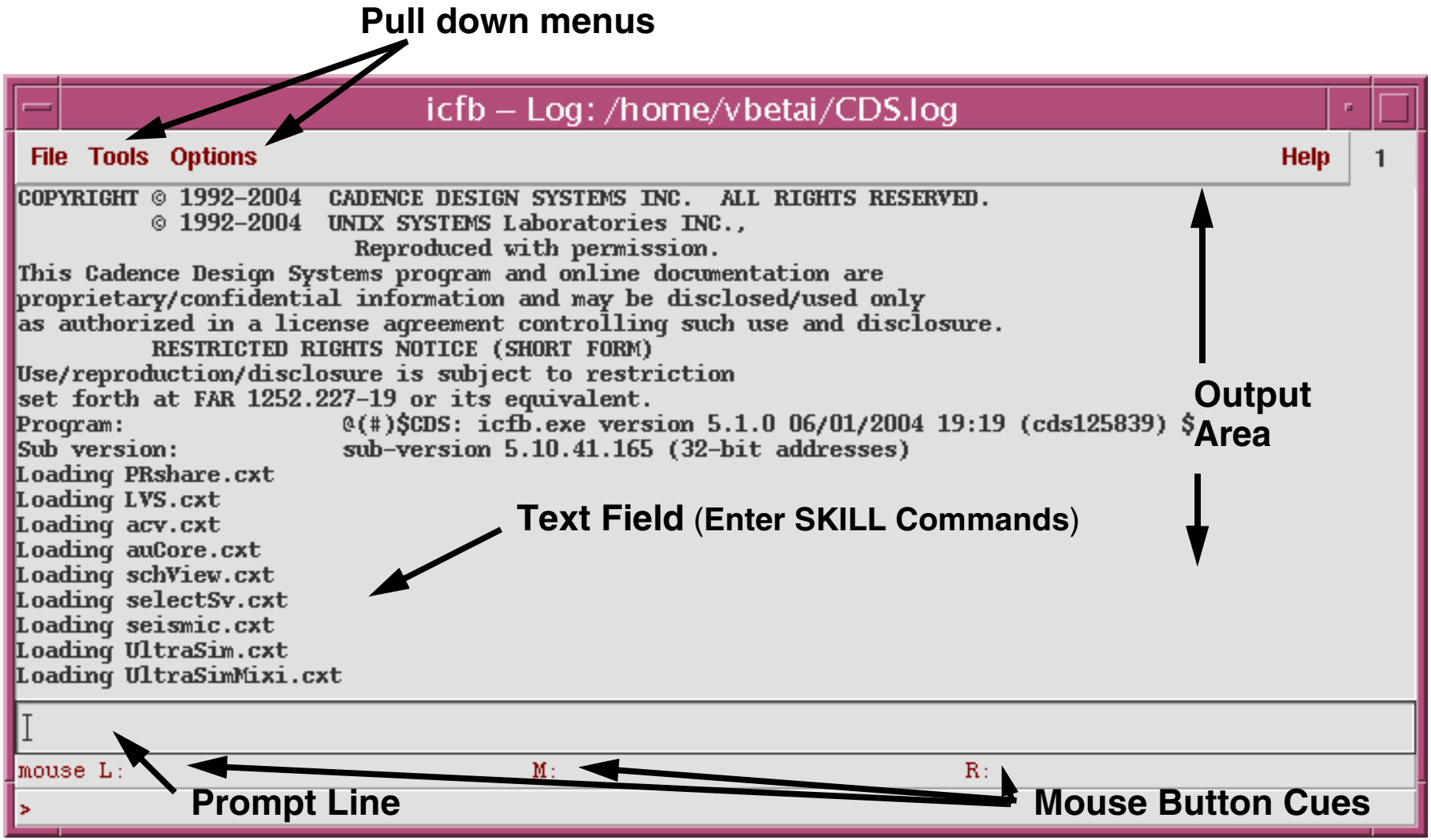
  A consistent database stores all design information. Tools share data in real time so long formalized translations between tools are not needed. The DFII environment also saves time during schematic to layout verification, because it updates layout geometries as the schematic component parameters change.

- Cooperating tools

  Applications run concurrently, with results available to all other tools, eliminating the need to open and close applications when changing tool contexts. For example, update and simulate a schematic without restarting the simulation environment. Updates are known to the simulation window as soon as they are made in the schematic entry window.

The Command Interpreter Window (CIW)

Enter: ***icfb &***, ***icms &***, *or* ***msfb &***



The Command Interpreter Window (CIW) is at the heart of the framework system. Use this window to access framework-based applications. System and error messages from applications are reported in this window.

- **Output Area**

  The output area displays a running history of the commands used with their results. For example, it issues a status message when a cell library is opened. This data is saved in the *Log File* whose path appears as the application title of the CIW. Use scroll bars to view previous output pane data without having to resize the CIW.

- **Text Entry Field**

  Enter Cadence SKILL commands in this area. Every pull-down menu command in the Design Framework II environment has an equivalent SKILL command. Advanced users can define and execute their own SKILL commands by entering them here.

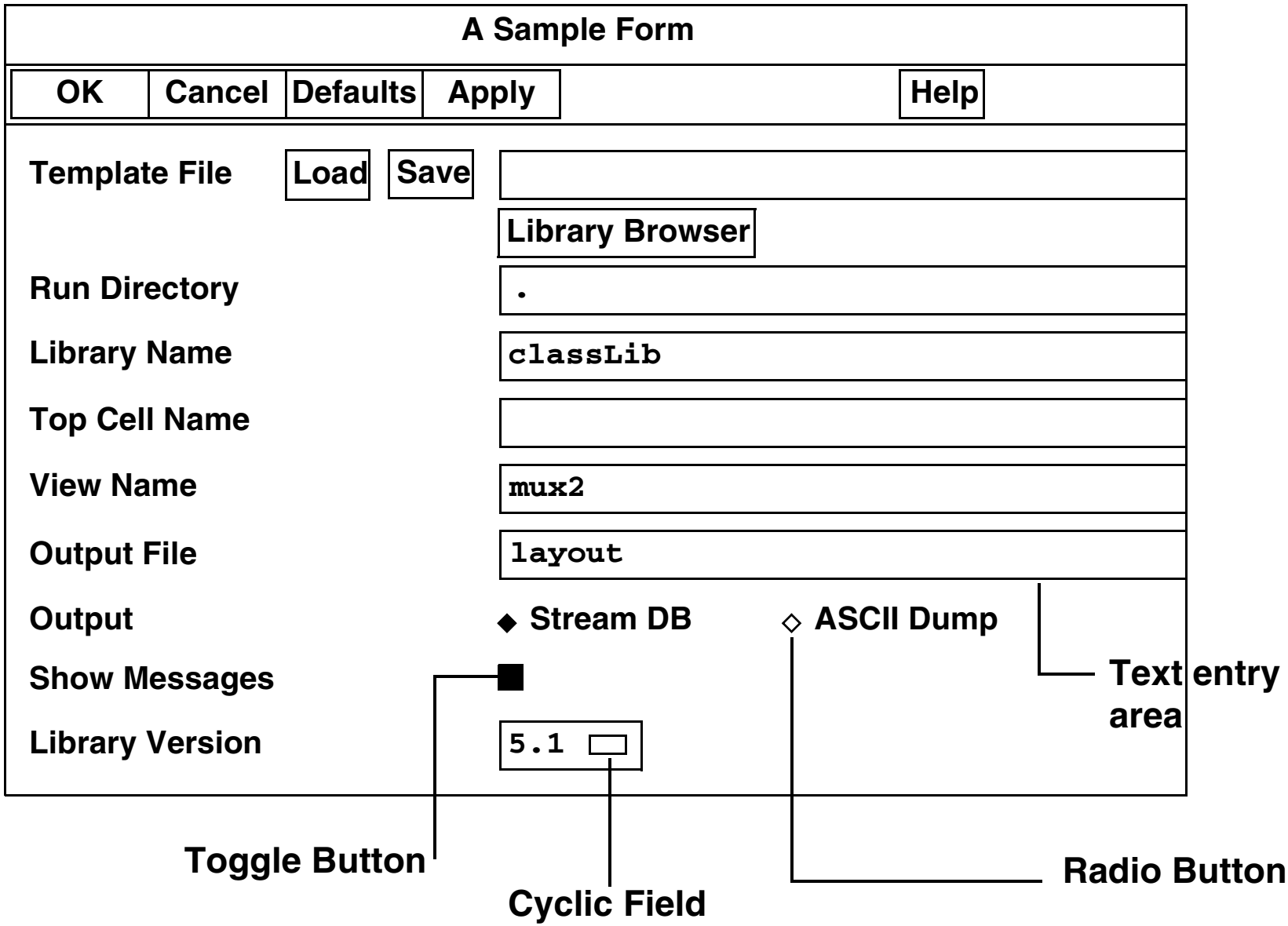
- **Prompt Line**

  The prompt line at the bottom of the CIW indicates the next step when carrying out a command executed in any Design Framework II application window.

- **Mouse Button Cues**

  Tells which mouse button to push to execute a command in a Design Framework II window.

Using a Form


A Sample Form
OK
Cancel
Defaults
Apply
Help
Template File
Load
Save
Library Browser
Run Directory
.
Library Name
classLib
Top Cell Name
View Name
mux2
Output File
layout
Output
◆ Stream DB
◇ ASCII Dump
Show Messages
Library Version
5.1
Text entry area
Toggle Button
Cyclic Field
Radio Button

Forms provide a place to enter the information required by a command.

The top of the form has a *title bar* and a set of *banner buttons*. The body of the form contains *prompts* that indicate which option is being set. Next to the prompt is one of the following:

- *radio button,* for choosing one of several options
- *text entry area*, for typing information
- *toggle button,* for turning options on or off
- *cyclic field,* for choosing one of many options. Initially only one option is shown. Move the pointer to the field and hold down the left mouse button, the other options appear.

The form might also have buttons such as *Browse*, which shows a browser window, or *More Options*, which displays another form.

Change of an entry on a form is disabled when the name appears in gray instead of black, and the text entry area is shaded.

- Press the **Tab** key or mouse to move to the next text entry field.
- Use the left and right arrows on the keyboard to move the cursor in a text entry field.

  Press **Control-a** to go to the beginning of a line; **Control-e** to go to the end of a line; **Control-u** to erase to the beginning of a line.

Initializing the Design Framework II Environment

The Design Framework II software reads your *.cdsinit* file at startup to set up your environment. The *.cdsinit* file:

- Sets user-defined bindkeys when the Design Framework II environment is started.
- Redefines system-wide defaults.
- Contains SKILL commands.

The search order for the *.cdsinit* file is:

- <install_dir>/tools/dfII/*local*
- the current directory
- the home directory

Here is the path to a sample *.cdsinit* file:

```
<install_dir>/tools/dfII/samples/artist/cdsinit
```

Start the Design Framework II environment, it reads the *.cdsinit* file to set up your configuration. The search order for the *.cdsinit* file is *<install_dir>/tools/dfII/local*, the current directory, and finally the home directory. When a *.cdsinit* file is found, the search stops unless a command in a *.cdsinit* file reads other user files.

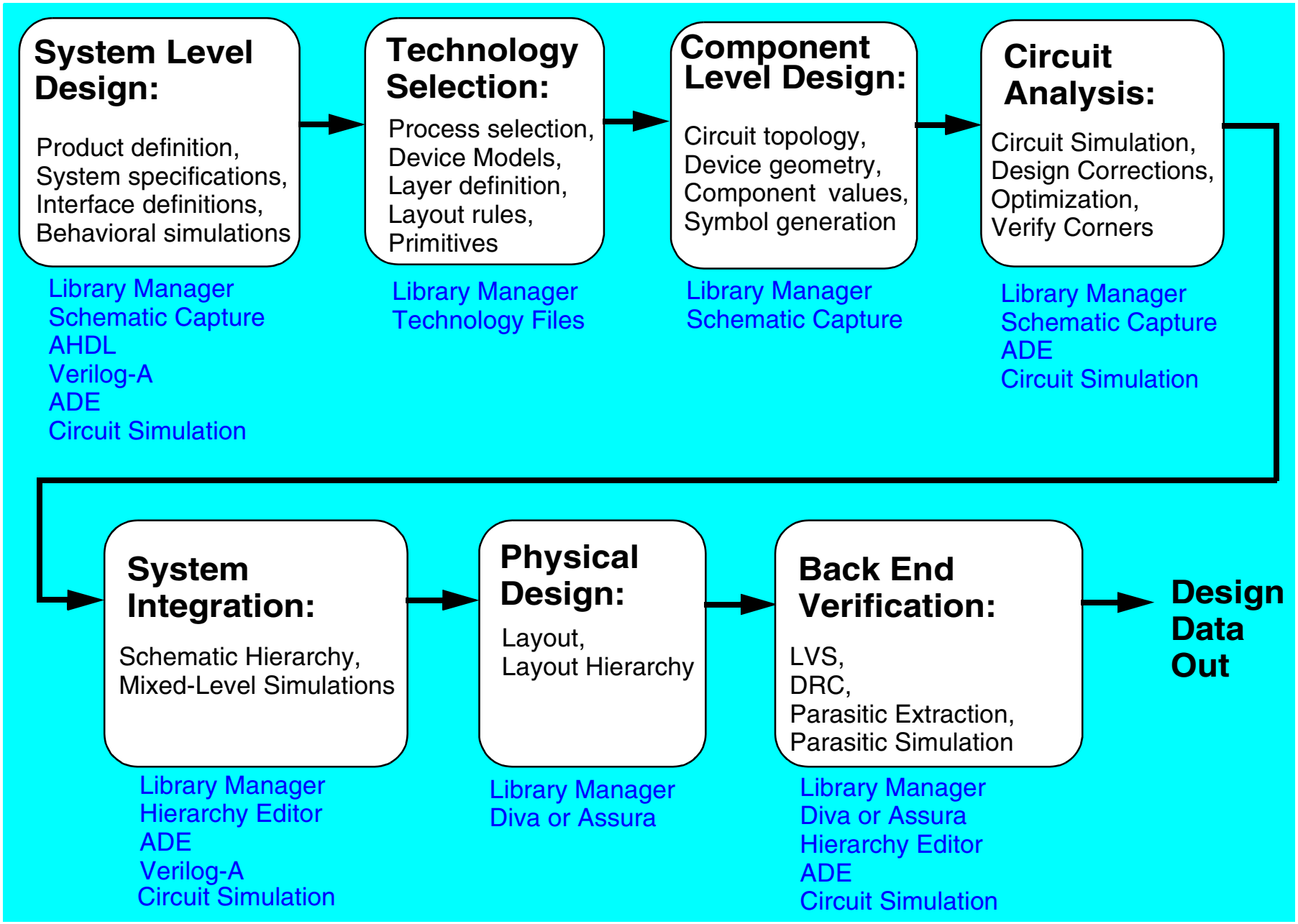
The *.cdsinit* file is a text file written in SKILL. A statement in a *.cdsinit* file can load user-defined bindkeys. Another statement might set Waveform Window defaults.

A sample *.cdsinit* file included with the software contains examples of statements to copy into your own *.cdsinit* files. It has very detailed comments about command usage. This sample is located at *<install_dir>/tools/dfII/cdsuser/.cdsinit*. An additional sample *.cdsinit* file exists for analog designers at *<install_dir>/tools/dfII/samples/artist/cdsinit*.

The Installation Path

The Design Framework II software product hierarchy is discussed in detail in the *Cadence Design Framework II Configuration Information* guide.

IC Design Flow, Front to Back

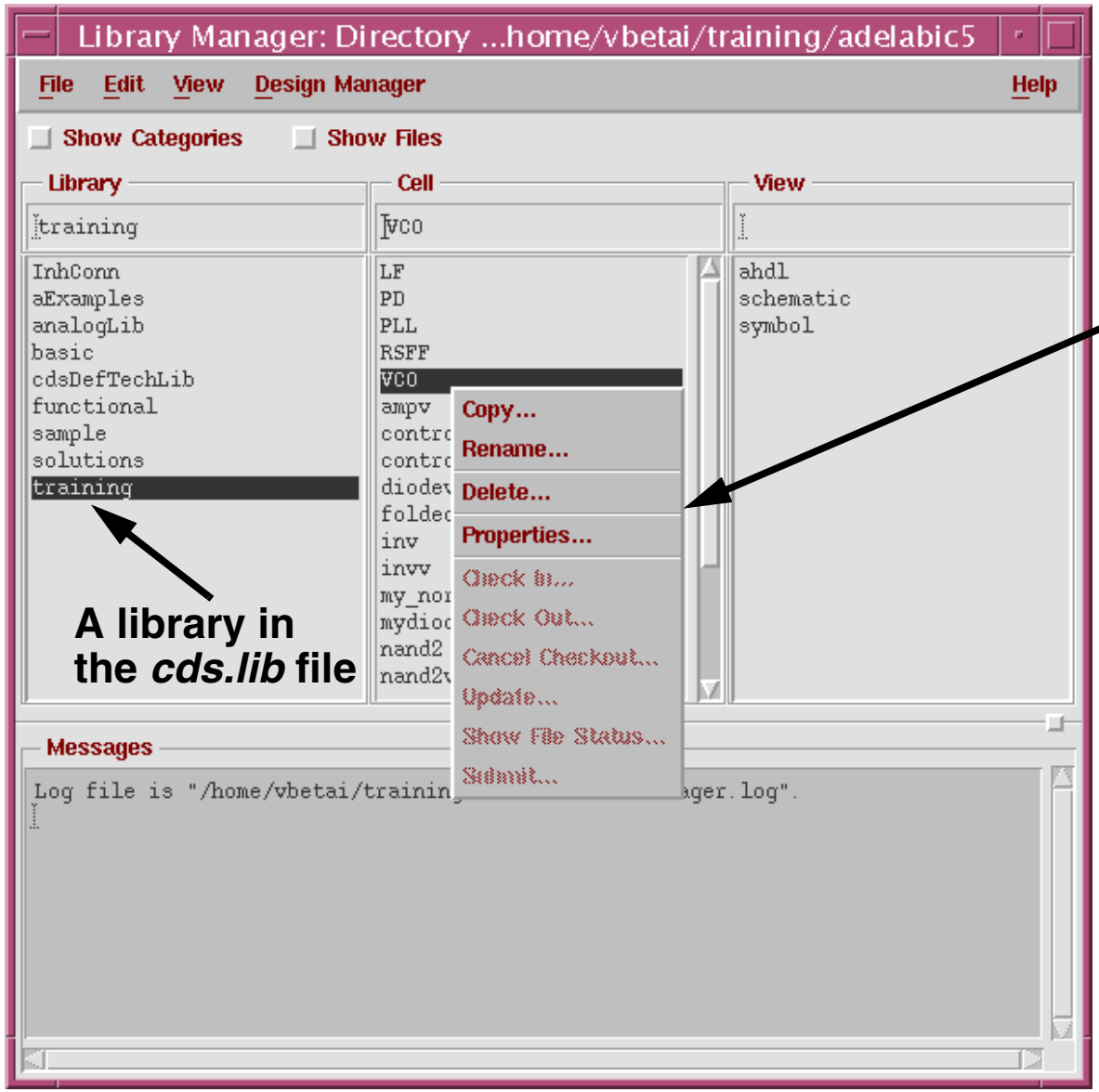


The graphical flow above shows a Front to Back design flow for integrated circuits or related system design. The blocks show the major steps or design categories. The text below each block shows the software tools used in the corresponding design steps.

- At the front end, the product, device or system is defined. The system-level specifications are used for behavioral simulation of the system.
- A fabrication process or technology is selected.
- A schematic of a specific block is captured.
- The design of the circuit is simulated. If needed, the circuit is redesigned to achieve specified goals.
- The circuit is integrated into a hierarchy. The hierarchy is then simulated.
- Physical design or layout capture of the circuit is completed. The layout of the hierarchy is then completed.
- Back end verification of the layout includes design rule checks, layout versus schematic checks, and parasitic extraction. The extracted parasitics are “backannotated” to the schematic for parasitic simulation using the circuit simulation software.

The Library Manager

The Library Manager is a graphical data management tool.



Object Sensitive Menus

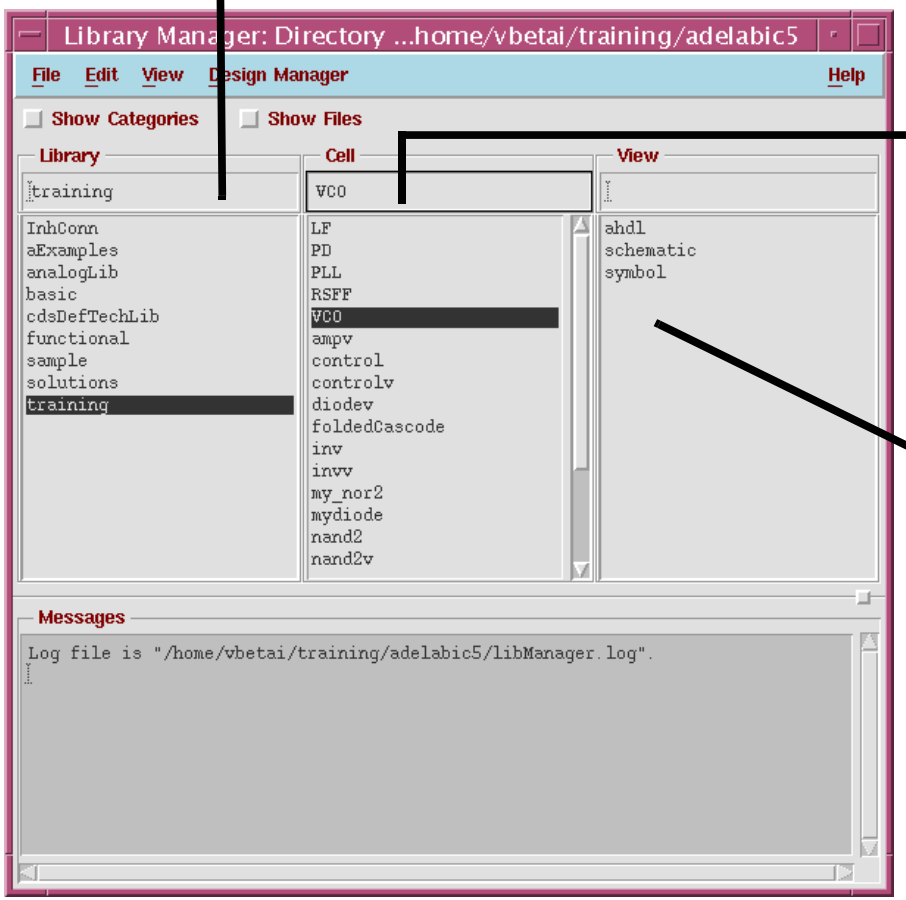
Tools—Library Manager

The Library Manager provides a convenient way to browse libraries containing cells and cell views. The most common use is to display the contents of libraries graphically. Other functionality includes renaming, copying, specifying permissions for, creating categories for, deleting, and viewing properties of design data. Use the Library Manager to create cells and views, to edit or read a design, and to access the design manager.

The illustration above shows a fully expanded library. Initially, the Library Manager lists only the library names that are set in the *cds.lib* file. This file contains the paths to the libraries used in the design session, including example libraries provided by Cadence, such as *analogLib* and *basic*.

Expand design data with Object Sensitive Menus (OSM) or with the mouse. To expand data, point at the word that represents the data in the Library Manager and choose the appropriate mouse button or menu command.

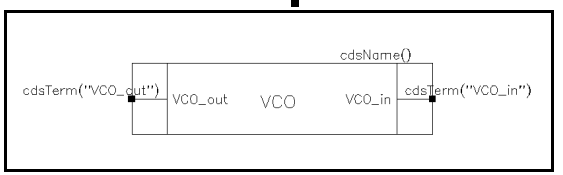
The Library Structure



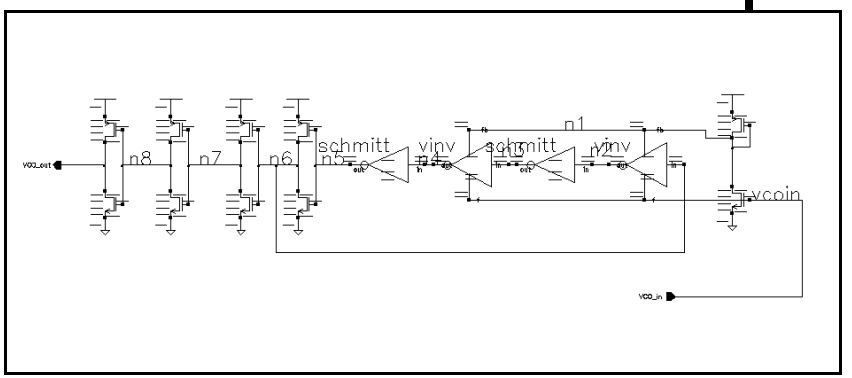
Tools—Library Manager...

Library Training

Cell VCO



Symbol View



Schematic View

- Library

  A *library* is a collection of cells. The library also contains all the different views associated with each of the cells. **Reference** libraries typically contain well-characterized cells that can be instantiated in many different designs. Examples are the *analogLib* and *basic* libraries. **Design** libraries contain cells currently under development by a particular user, group, or for a particular design project.

- Cells

  A *cell* is a logical component in your library. It can be a building block such as a *VCO* or *amplifier*. It can also be the top level chip name.

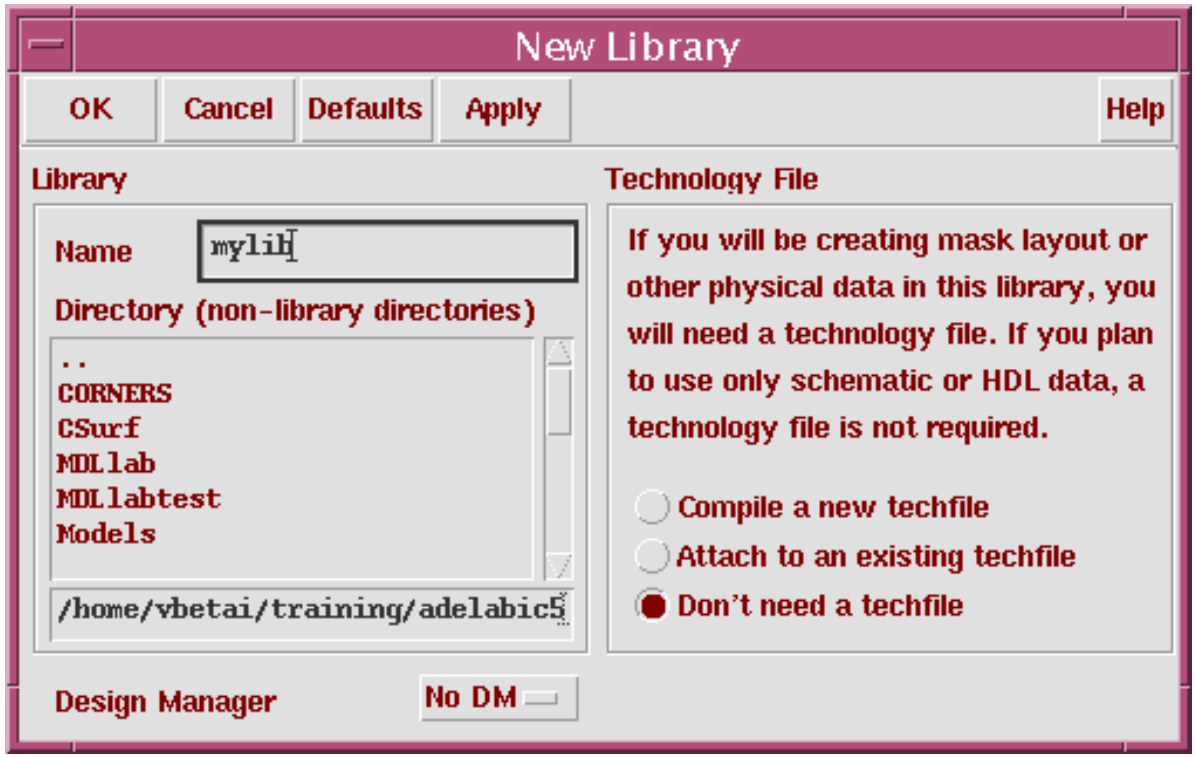
- Views

  A *view* is a particular representation of a cell such as a *layout*, *symbol*, or *schematic*. An application tool, such as *Virtuoso Schematic Editor*, creates a view. Although a chip can include many levels of cell hierarchy, none of the hierarchical complexity is reflected in the libraries.

  A library is a flat collection of cells. Details of the design hierarchy exist inside the views that contain instances of other cells. The library treats all cells the same.

Creating a New Library

In the CIW or the Library Manager, select **File—New—Library**.



- Specify the library name and path.
- Specify the design manager to use.
- For Physical Design and Verification, specify the ASCII technology file or technology file library to be attached to the new library.

  The new library is entered into the *cds.lib* file.

When creating library, use a form to specify the library name and path, the design manager to use, and the technology file to attach to the library.

- Technology File Contents

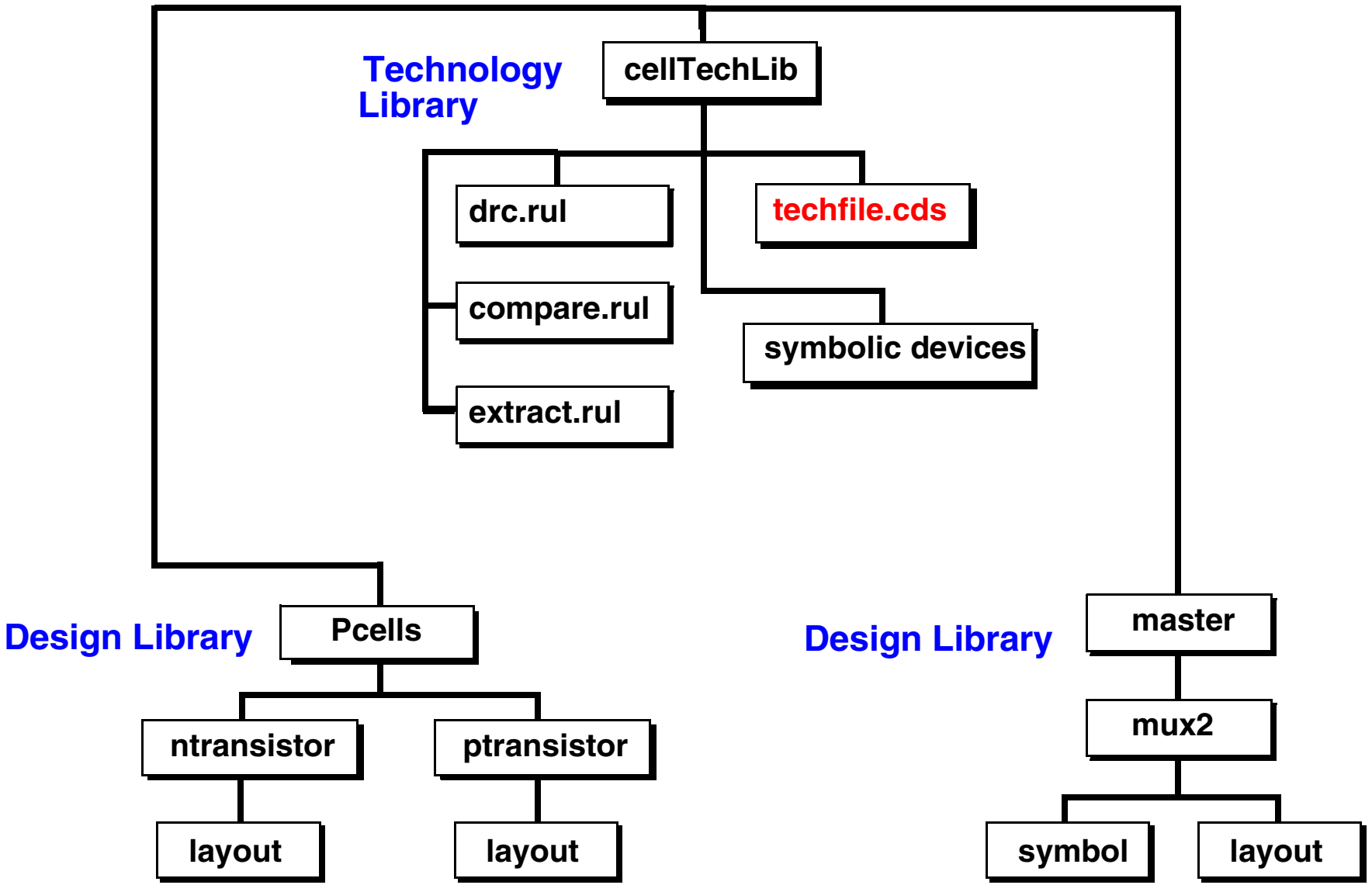
  The *technology file* is a large data file that specifies all of the technology-dependent parameters associated with that particular library. Design rules, symbolic device definitions, and parasitic values are some of the technology-specific parameters common to all cells in a library.

- *cds.lib* File

  The software automatically updates the *cds.lib* file when creating a library through the CIW's **File—New—Library** command, when copying one library to another name, or renaming a library. This file contains the paths to all of the libraries used in the design session, and can be accessed through CIW's **Tools—Library Path Editor** command.

Shared Technology Library

This example shows several libraries sharing the same technology file library.



Share technology file information between different libraries. Create a technology file library and attach your design libraries to the technology file library. Use the **Technology File—Attach To** command in the CIW to attach the design library to the technology file library. Sharing a technology file library with other libraries will share the same Assura® rules, layer information, and symbolic devices amongst a group of libraries. Sharing a technology file can help reduce the size of the design libraries, because the technology information is stored at only one location.

- *techfile.cds* file

  The *techfile.cds* file contains the binary technology file. This file name must be called *techfile.cds*.

- Assura Rules

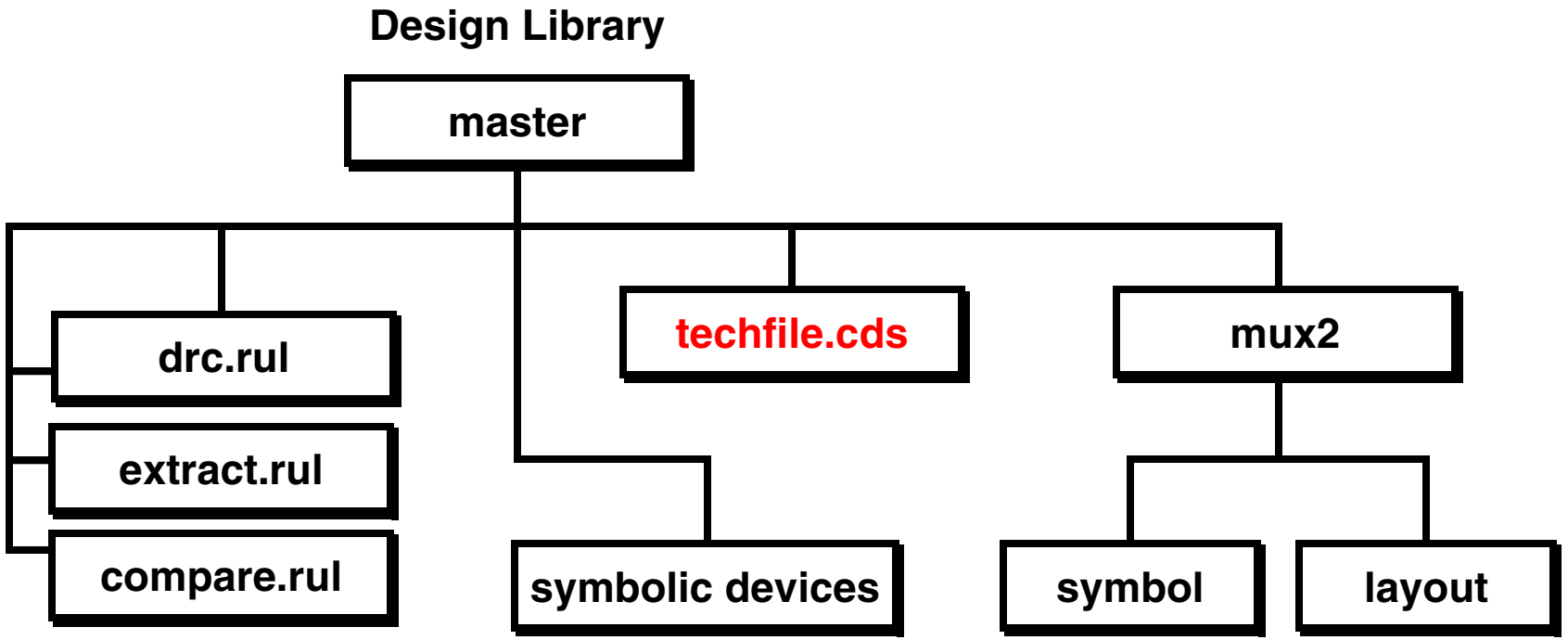
  The Assura rules are stored as separate ASCII files. For each type of rule (DRC, Extract, and Compare), there is an Assura rules file.

- Symbolic Devices

  The symbolic devices such as contacts, pins, transistors, and wire information can be shared between libraries.

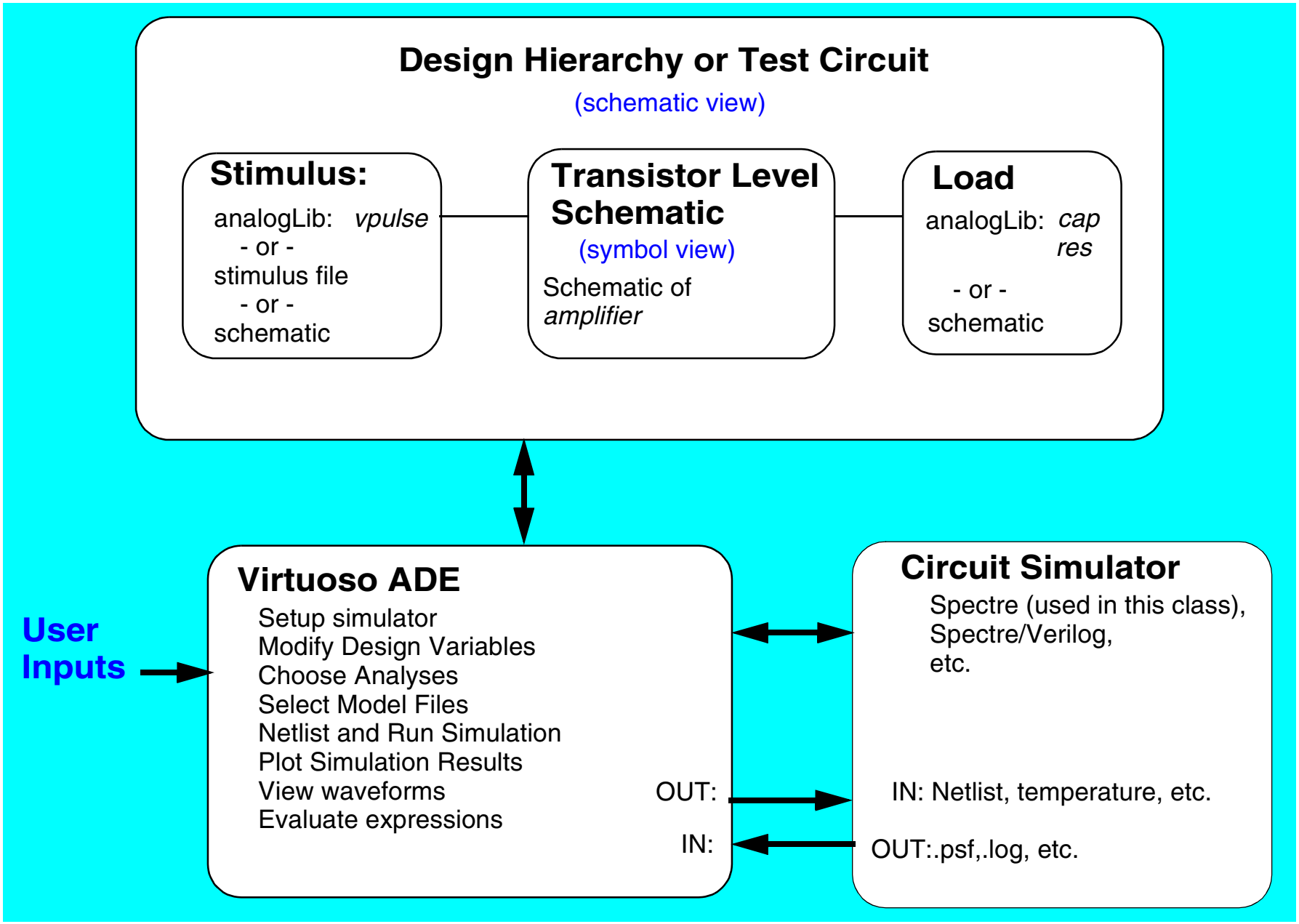
Technology File Stored in the Design Library

This example shows a technology file being stored inside a design library and not being shared with other libraries.



A library can have its own technology file information that is stored inside of the library.

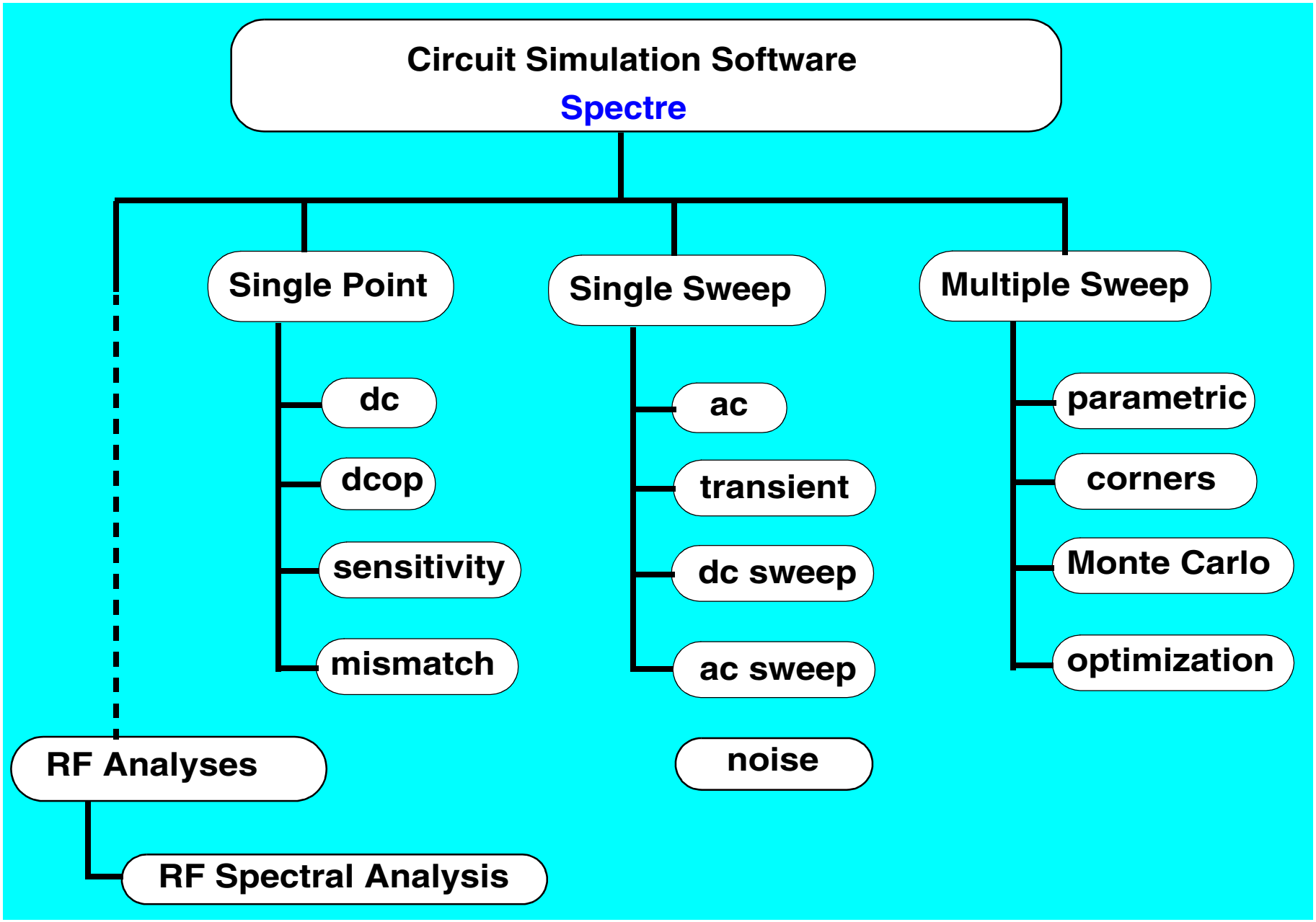
Overview of Circuit Simulation



The block diagram above shows an overview of the circuit simulation process.

- The circuit schematic is captured or edited.
- A symbol of the schematic is placed in a hierarchy or test circuit schematic.
- The Virtuoso Analog Design Environment is started.
- The user provides input to the Virtuoso Analog Design Environment to set up and control what information is netlisted and then sent to the circuit simulator.
- The user used the Virtuoso Analog Design Environment to run the circuit simulator.
- The user selects the information to be printed, plotted, or to be analyzed.
- The user modifies the setup or edits the schematic for the next simulation.

Types of Circuit Simulation Analyses



The diagram shows the variety of analyses available with analog circuit simulation.

Single-point analyses often include the steady state **dc** solution of the circuit. The operating point **dcop** solves the operating point device parameters and low frequency gain of the circuit.

Single-sweep analyses often include **ac** and **transient** analysis. The **ac** analysis is a frequency sweep of the circuit. The **transient** analysis is a time sweep of the circuit operation to a time domain stimulus. A **dc sweep** analysis is a multiple point dc analysis performed while stepping a parameter such as temperature, design variable, or a model parameter. Solving the dc gain of an amplifier as a function of temperature is often called a **temperature sweep**. Solving the gain of an amplifier as a function of a model parameter is called a parametric sweep. It is also possible to sweep the **ac** gain of an amplifier at a specified frequency of the amplifier over temperature. This is called an **ac temperature sweep**.

Multiple sweep analyses refer to sweeping one variable and then stepping another variable between successive sweeps. In **parametric** analysis, the two or more variables are altered at specified intervals. In the **Corners** analysis, variables are specified at named corners. In the **Monte Carlo** analysis, the parameters are altered using random number generators and a specified distribution. In **optimization** analysis, the parameter are altered using the results of the previous simulation and a search algorithm.

The Virtuoso Analog Design Environment and the Spectre circuit simulator are capable of performing the analyses shown above. In addition, the Spectre circuit simulator can perform special steady state **ac spectral analysis** on RF waveforms. These analyses are discussed in the Spectre RF classes.

Summary

In this module we discussed:

- Course objectives
- Getting Help, including CDSDoc
- Design Framework II environment
- Using forms
- Creating a library
- Creating cells and cell views
- Overview of schematic capture
- Overview of circuit simulation in the Virtuoso Analog Design Environment
- Types of simulation analyses

In this module we provided an introduction to the class, including:

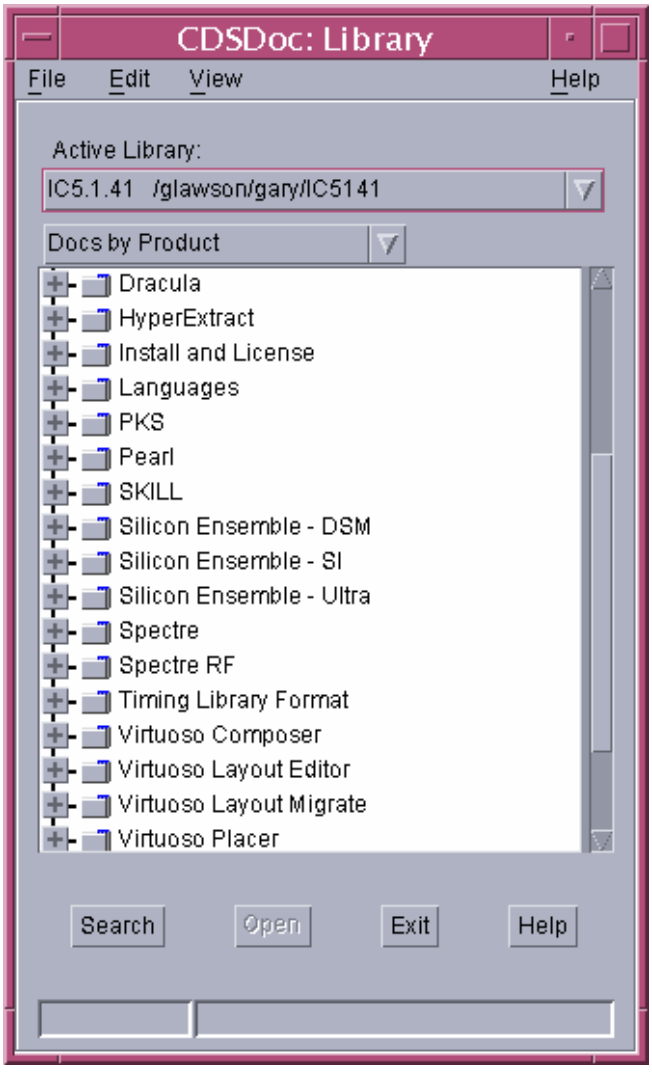
- Class objectives
- Online documentation

This module also provided discussion on the Design Framework II environment, including:

- Starting DFII with *icfb*, *icms*, or *msfb*.
- The Command Interpreter Window (CIW)
- Use of forms
- Front to Back design flow using Design Framework II
- Overview of schematic capture
- Overview of circuit simulation in the Virtuoso Analog Design Environment
- Types of simulation analyses

Labs for Module 1

Introduction



Lab 1-1 Setting Up the Database

Objective: To download and set up the course database.

1. Click the ADE_5_1_41.tar.Z link and download the compressed file to your home directory.

   Note: You can put the database where you like, but these instructions assume your home directory is the destination for the database.

2. After downloading the database file, you must uncompress it using the following command:

   ```
   uncompress ADE_5_1_41.tar.Z
   ```

   After you uncompress the course database file, you will have a tar file called *ADE_5_1_41.tar*.

3. Extract the *.tar* file by entering the following command in the window:

   ```
   tar xvf ADE_5_1_41.tar
   ```

   This command creates a directory called adelabic5.

4. You can now remove your tar file by entering:

   ```
   rm ADE_5_1_41.tar
   ```

5. Make sure that you read the *README* file in the *ADE* directory concerning the versions of the software to use with these labs.

of Lab

Lab 1-2 Getting Started

Objective: To login, start a shell tool, and use CDSDoc.

This lab activity is intended to help you begin using the Design Framework II design tools. You need to be familiar with the workstation and the operating system you will be working with.

This class was developed using the Solaris 2.8 operating system. If your operating system is different, your instructor or system administrator may provide additional instructions.

1. In a terminal window, type `csh` at the command prompt to invoke the C shell.

2. To verify that the path to the software is properly set in the .cshrc file, open a terminal window and enter:

```
> which icms
```

Starting the Online Help, CDSDoc

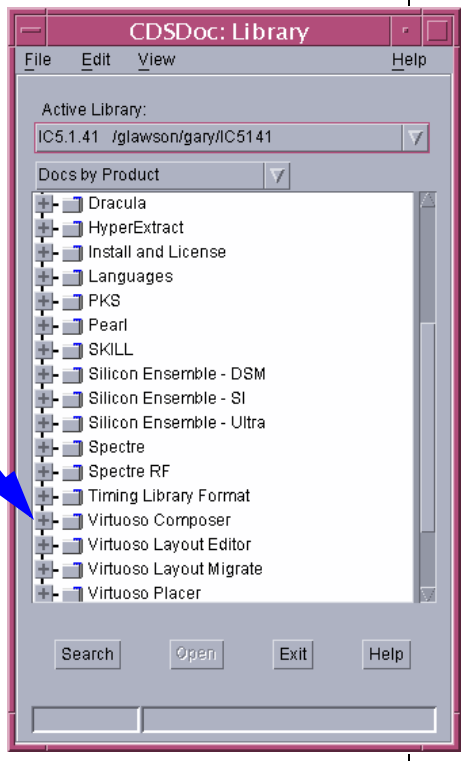
1. In the UNIX window enter the command:

   ```
   > cdsdoc &
   ```

2. The CDSDoc: Library window should appear.

   This is a graphical interface for selecting topics from the online documentation tool.

For step 3, move the cursor to the + in front of the Virtuoso Composer folder.



3. With the mouse, position the cursor on the + symbol in front of the **Virtuoso Composer** folder, and click **left**.

   The Composer folder expands to show available topics.

4. Click **left** in the subtopic field.

   The *Virtuoso Composer User Guide* appears.

5. Activate the selected topic with a **left** click on the open button.

6. After briefly viewing the information displayed, close the browser. Then click the **Exit** button on the **CDSDoc: Library** window.

The Cadence online documentation is available even without starting the Cadence design software. If the software is installed, you can access the documentation from any terminal window operating in a C shell with a path to the installation by typing `cdsdoc` at the command prompt.

Starting the Cadence Software

You will use the installed database to do your work.

1. Change to the course directory by entering this command:

   ```
   > cd ~/adelabic5
   ```

   You will start the Cadence Design Framework II environment from this directory because it contains cds.lib, which is the local initialization file. The library search paths are defined in this file.

If you start the Design Framework environment from another location, you will have to perform additional steps to access the training libraries.

2. In the same terminal window, enter:

   ```
   > icms &
   ```

   The Command Interpreter Window (CIW) appears at the bottom of the screen.

3. If the “What’s New ...” window appears, close it with the **File—Close** command.

4. Iconify other windows that are open at this time and make sure that the CIW is visible.

Lab 1-3 Top-Down System Modeling

Objective: To simulate a block-level design that uses AHDL modules.

In this first lab, you will simulate a top-down, block-level design. Assume that a design specification has been given to you, and you have used AHDL (analog behavioral) modules using the Verilog®-A language to describe the functions of your blocks.

> **Note:** All of the Verilog-A descriptions used in the modules in the *peakDetectv* design came from the Cadence hierarchy. They are included with your software installation.

The design is a peak detector circuit, which is already set up for you in a testbench for simulation. In this lab, you will run the simulation on this block-level design. For the rest of the course, you will then build the individual blocks that characterize the system, following the logical steps of a design flow.

This lab will give you a brief introduction to the simulation environment. You will learn about the simulation environment in great detail in the following modules.

Design Flow

The Design Flow diagram below represents the steps that will be taken in the front-to-back analog design methodology in this class. The current block is highlighted to indicate where you are in the flow.

Opening the Peak Detector Circuit

1. In the CIW, execute **Tools—Library Manager**.

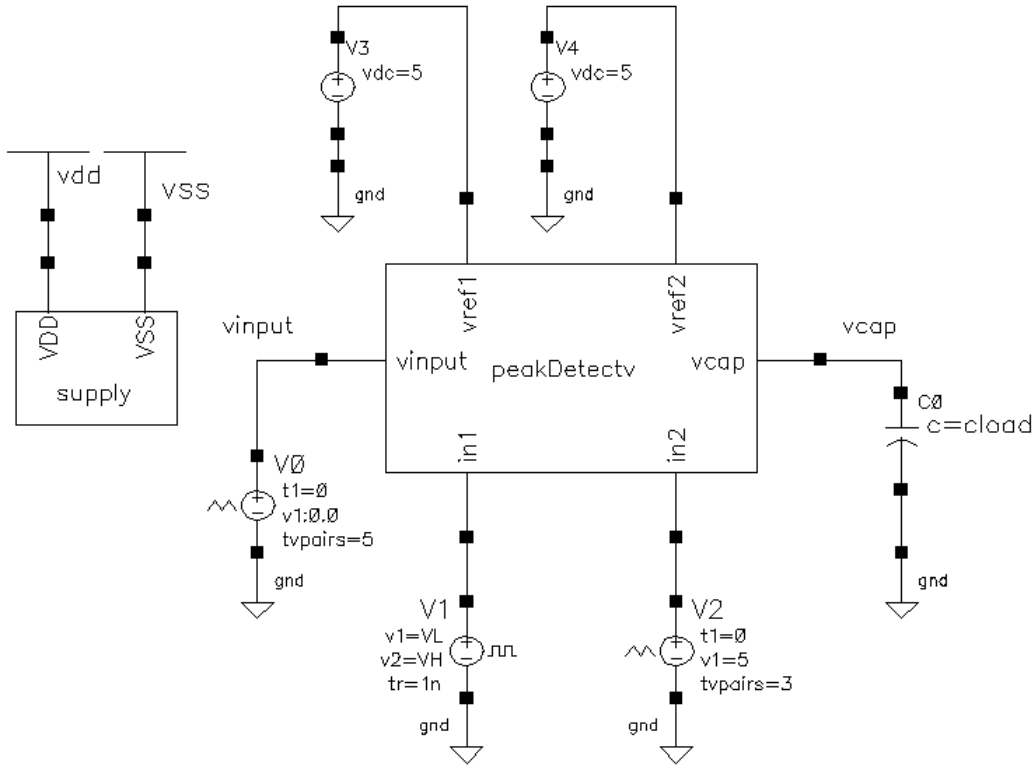
   The Library Manager appears.

2. In the Library Manager, click **left** to expand the *training* library and the cell named *peakTestv* (“v” is for Verilog-A).

   Only a *schematic* view exists for this cell.

3. With your **middle** mouse button, activate the Object Sensitive Menu (OSM) over the word *schematic* in the View section and select **Open**.

   The schematic of the testbench with supplies and stimuli for a peak detector circuit appears.



4. Run your cursor over the *peakDetectv* block until a rectangular box is highlighted around it. Click **left** to select the *peakDetectv* block, and the box will go from dashed lines to solid.

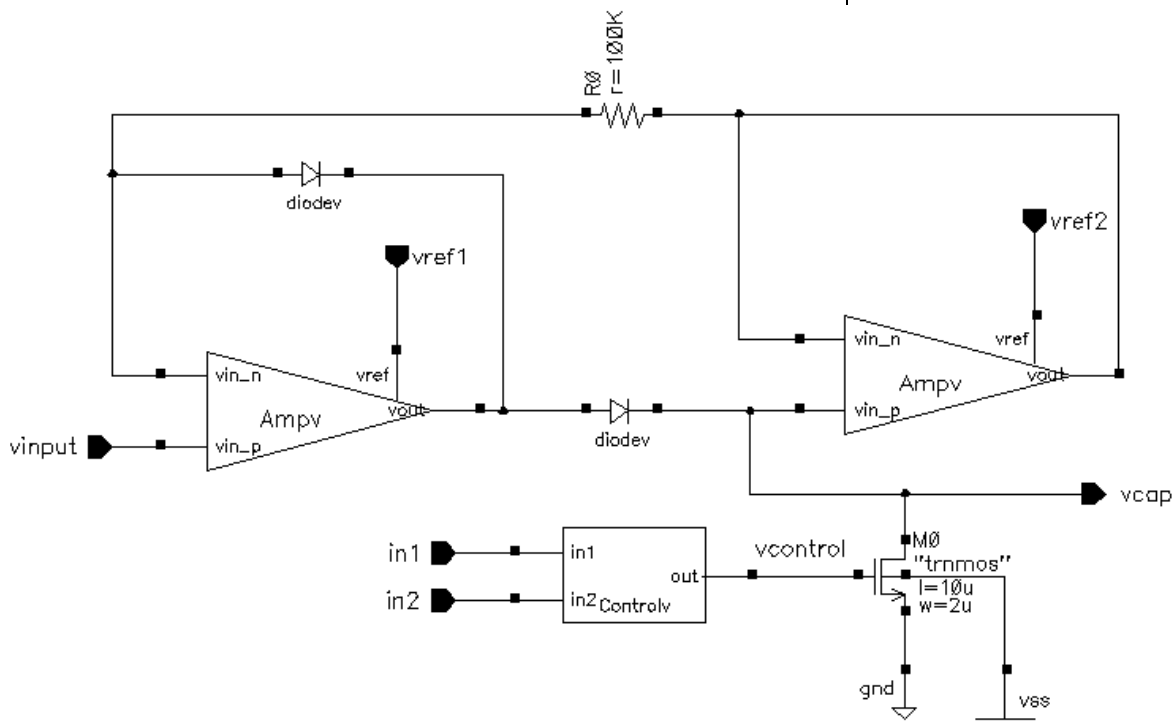
5. Execute **Design—Hierarchy—Descend Edit**.

   Tip: You can also press the **E** (Shift E) key.

6. In the Descend form that appears, set the View Name to schematic, and click **OK**.

   The *peakDetectv* schematic appears. You will study this design throughout the course. It consists of two operational amplifiers, two diodes, an NMOS transistor, and a resistor. All of the components except the resistor and NMOS device are modeled with Verilog-A, an analog HDL language.

   Using the Verilog-A language supports the top-down design methodology.



Viewing the AHDL Description

You can view the Verilog-A description of the operational amplifier (opamp), diode, or “digital” gates inside of the *control* block.

1. Run your cursor over one of the *Ampv* blocks in the *peakDetectv* schematic until a rectangular box is highlighted around it. Click **left** to select it.

2. Execute **Design—Hierarchy—Descend Edit**.

   Tip: You can also press the **E** (Shift E) key.

3. In the Descend form that appears, set the View Name to **veriloga** and click **OK**.

   A text window appears.

4. Put your cursor in the window, and press **Return**.

   The Verilog-A description of the opamp, called *ampv*, appears. Study the file and see if you can understand the behavior and structure of this description.

5. After viewing the file, exit the editor: For the vi editor, enter:

   Esc : q ! Return

   If you have used the Text Editor tool provided by the Common Desktop Environment (CDE), execute **File—Close**.

6. View the *veriloga* descriptions of the diode and logic gates if time permits.

Running Simulation

You will set up and run a simulation. You will learn more about the specifics of the simulation environment in the next section.

1. In the schematic window, execute **Tools—Analog Environment**.

   In a few moments, the Virtuoso Analog Design Environment simulation window appears, and the *peakTestv* schematic is in view again.

Choosing a Simulator

You will set the simulator to run the Spectre® tool, a high-speed, highly accurate analog simulator that is integrated directly into the Virtuoso Analog Design Environment.

1. In the Simulation window, execute **Setup—Simulator/Directory/Host**.

2. In the Choosing Simulator form, set the Simulator field to **spectre** and click **OK**.

Setting the Model Libraries

The Model Library File contains the models needed for simulation. For this simulation, the NMOS device in the *peakDetectv* design is the only one that is used.

1. Execute **Setup—Model Libraries**.

   The Models Library Setup form appears.

2. Place your cursor in the Model Library File field and enter the following value:

   ```
   ~/adelabic5/Models/myModels.scs
   ```

3. Click **Add**.

   The path moves to the Model Library File field. This file will now be included in the simulation environment and used to describe the NMOS transistor in the *peakDetectv* circuit.

4. Click **OK** in the Setting Model Path form.

Choosing Analyses

You will run a transient analysis on the *peakTestv* design.

1. In the Simulation window, click the **Choose Analyses** icon.

   Tip: You can also execute **Analyses—Choose**.

   The Choosing Analyses form appears.

2. To set up for Transient Analysis:

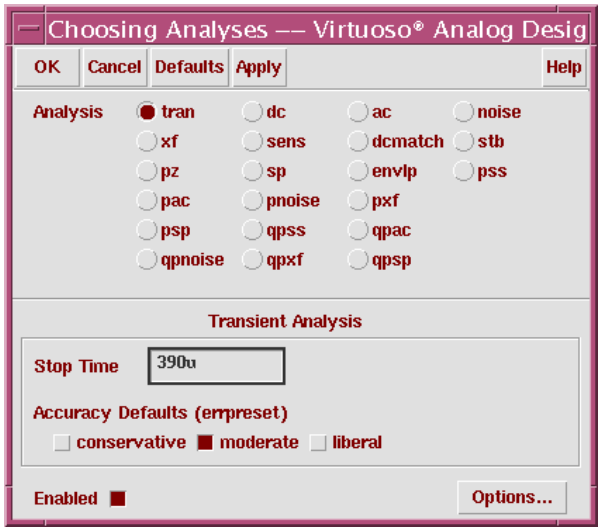
   a. In the Analysis section, select **tran**.

   b. Set the Stop Time field to **390u**.

   c. Select **Moderate** as the errpreset.

   d. Turn **On** the Enable button.

e. Click **OK** in the Choosing Analyses form.



Saving Outputs for Plotting

Select the nodes that will be plotted automatically when the simulation is complete. You will select your nodes in the *peakDetectv* schematic.

1. If the *peakDetectv* schematic is visible, proceed to Step 4.

2. If the *peakTestv* schematic is visible instead of the *peakDetectv* schematic, select the *peakDetectv* block and execute **Design—Hierarchy—Descend Edit**.

   Tip: You can also press **E** (Shift E).

3. In the Descend form that appears, set the View Name to **schematic**, and click **OK**.

   The *peakDetectv* schematic appears.

4. Execute **Outputs—To Be Plotted—Select On Schematic**.

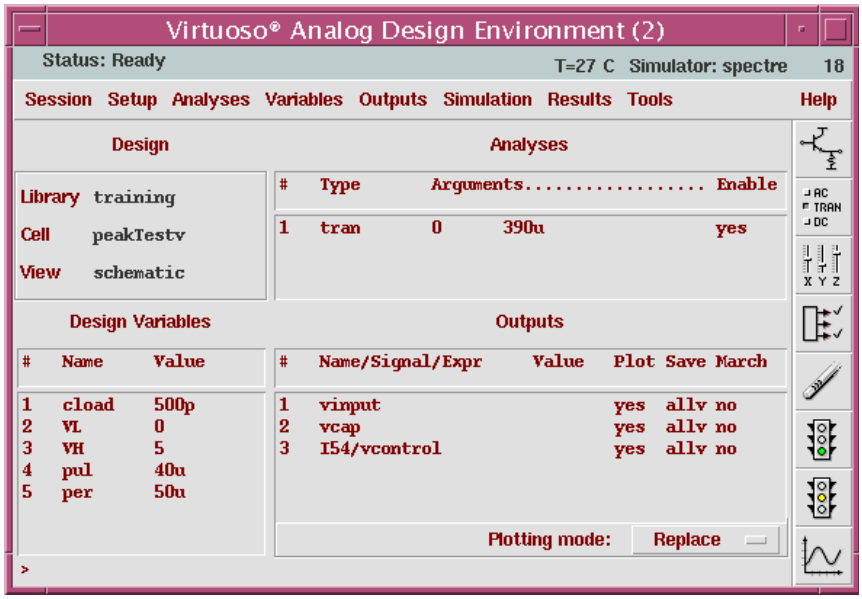
   Follow the prompts at the bottom of the schematic window and click on the wire or wires connected to the pins labeled *vinput*, *vcap*, and *vcontrol*. The nodes will highlight with unique colors.

5. With your cursor in the schematic window, press **Esc**.

   Note the updates in the Outputs section of the Simulation window. The signal *vcontrol* has the name *I54/vcontrol*.

   Do you know why?

   Does your simulation window look similar to this?



Note: At this point, you can save these environment settings using the **Session—Save State** command, enter **state1** in the *Save As* field then click **OK**. This will allow you to reload these settings next time you work with this schematic.

Running the Simulation

You will run the simulation and view the results. You will not need to generate and view a netlist before running simulation. If you choose to view a netlist, you can use the **Simulation—Netlist—Create** command.

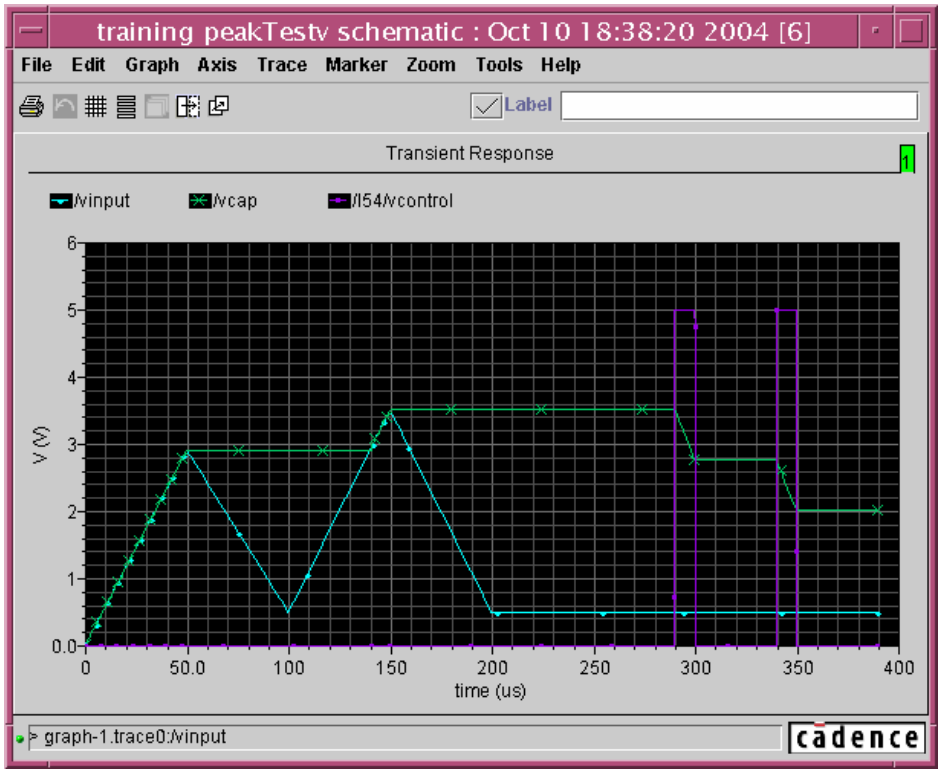
1. Execute **Simulation—Netlist and Run** to start the simulation or click the **Netlist and Run** icon in the Simulation window.

   The simulation starts, and a separate window appears and prints the simulation status as it runs. If there are errors in the setup, you will receive messages in this window directing you to take appropriate action. See if you can debug the errors, or check with the instructor.

   You might try creating a netlist and then clicking the **Run** icon to run the simulation, after you have corrected your errors.

   When the simulation completes, the Transient analysis results plot automatically.

   Does your Waveform Window look like this?



Note the following:

- The load capacitor stores the peak value of the input voltage as it is being charged (signal *vcap*).
- The *vcontrol* signal is applied to the gate of the NMOS device that discharges the load capacitor. It turns the transistor on or off.
- The pulse width of the *vcontrol* signal determines how much of the capacitor charge is dissipated.

Preparing for the Next Lab

1. In the Simulation window, execute **Session—Quit**.
2. In the schematic window, execute **Window—Close**. Click **No** in the Save Changes form if it comes up.
3. In the window that was started by the Simulation environment, execute **File—Close Window**.

Now that you have simulated a high-level, block description of your design by using AHDL, you will begin to build the schematics of each block in your design. The schematic-level blocks that you build will be used in several different designs in your labs.

In the next lab, you will start by creating a library, and then by building a schematic for the opamp that you simulated in this lab.

Schematic Entry

Module 2

February 10, 2005

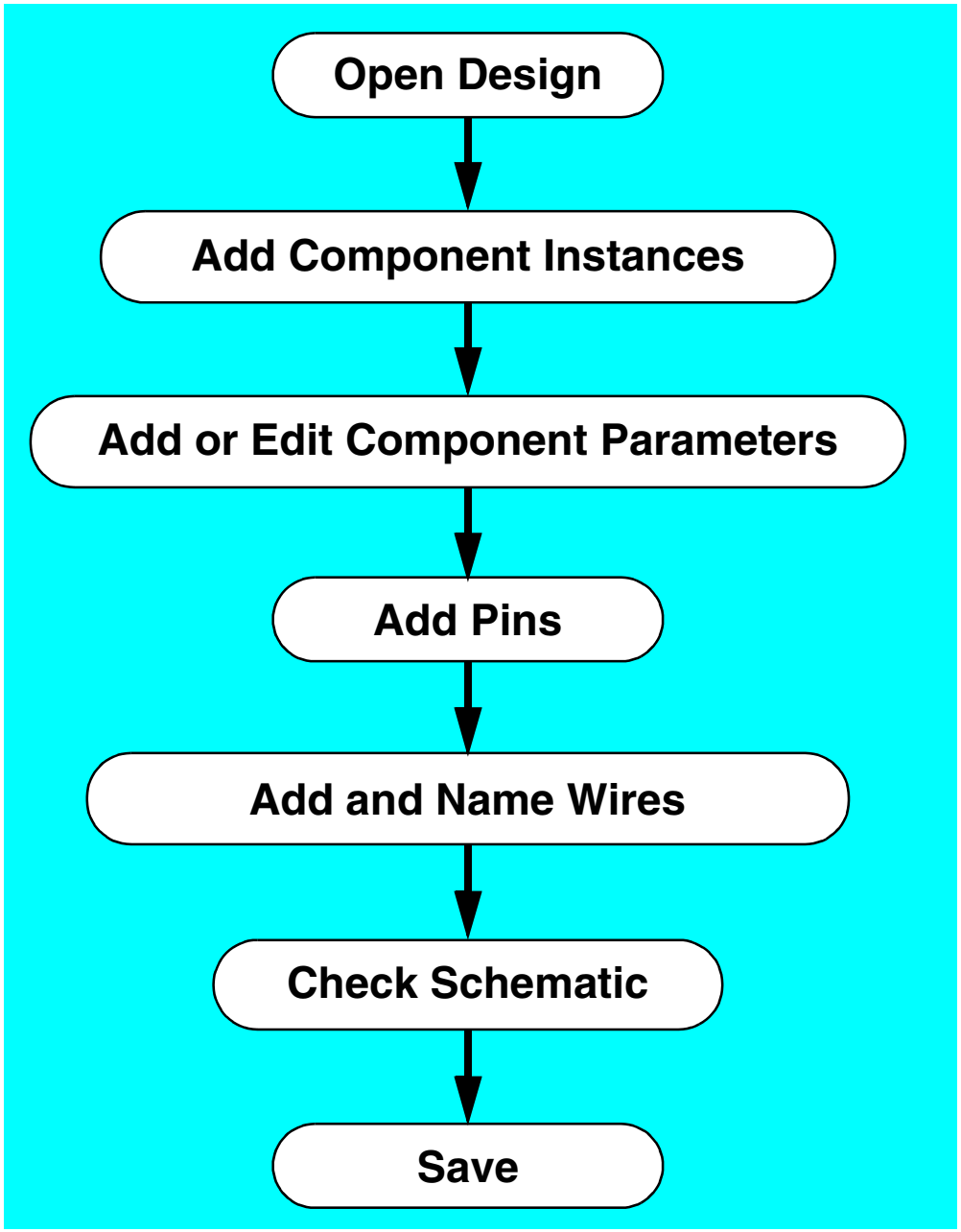
Topics in this Module

- The schematic capture flow
- Creating a schematic view
- Contents of a schematic
- Adding component instances
- Adding pins
- Adding wires
- Editing object properties
- Using Accelerator keys (also known as bindkeys) and schematic window icons
- Checking the schematic for errors
- Symbol generation and editing
- Using a design hierarchy

Terms and Definitions

library	A set of design directories that includes ‘cells’ and ‘cellviews’.
Library Manager	A Cadence tool that allows user to browse and edit a design library.
Virtuoso Schematic Editor	Schematic editor and symbol generation tool in DFII.
cell	A basic unit of a design hierarchy described by cell views.
cell view	A specific view of a cell that includes schematic, symbol or layout.
instance	A uniquely named placement of a symbol onto a schematic.
pin	A connection point on a schematic and symbol used for accessing signals.
bindkey	A predefined key on the keyboard that invokes a preselected command.

Schematic Entry Flow

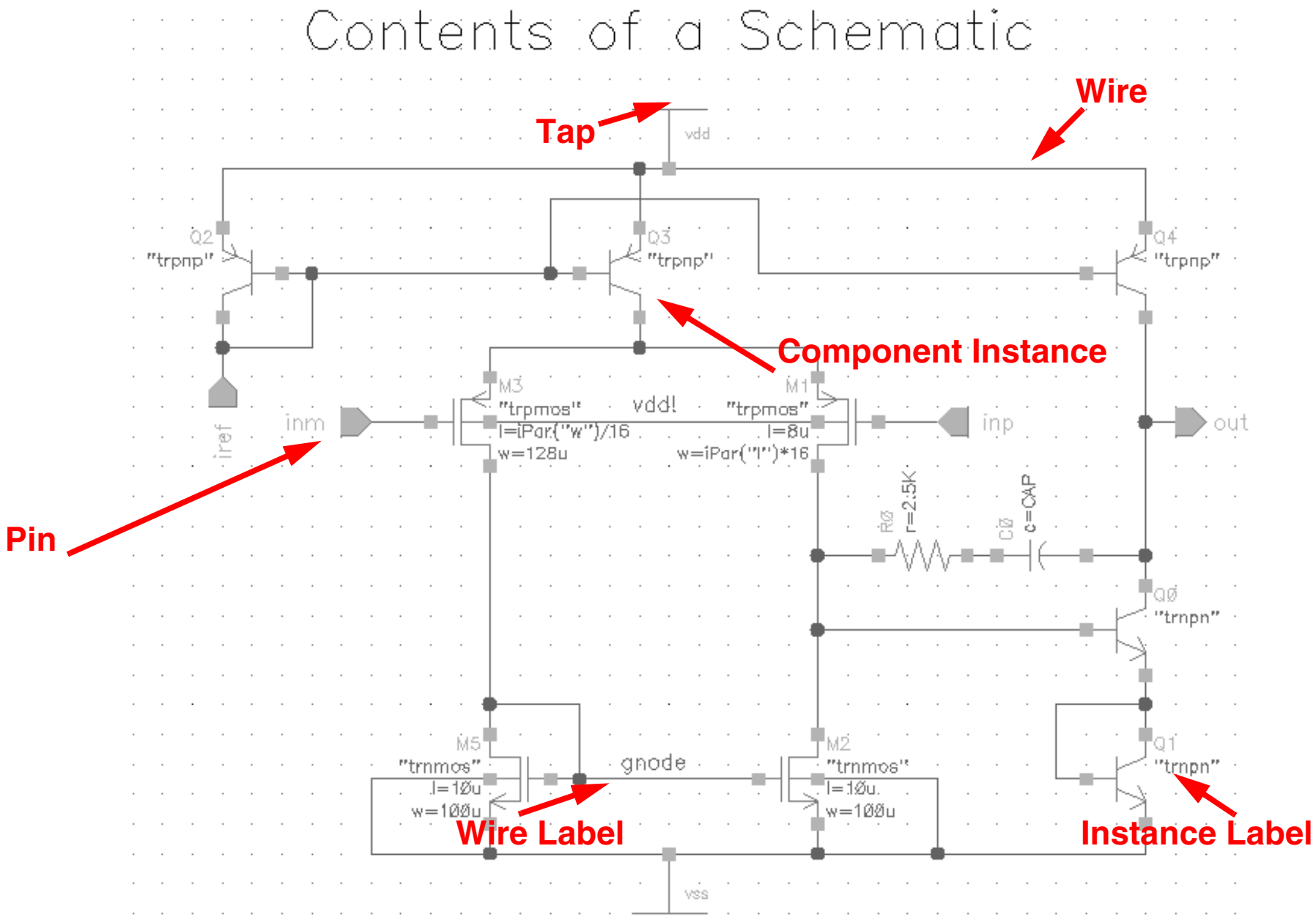


Perform the following steps when creating a schematic:

1. Open the design. Use the CIW or Library Manager tool.
2. Add component instances by placing cellviews from libraries.
3. Add or modify component parameters.
4. Add pins to indicate connections outside of this schematic.
5. Connect the components and pins.

   Use wires to do this. This step also includes giving meaningful names to signals in the design.
6. Check the design to ensure that it is correct.
7. Save the design.

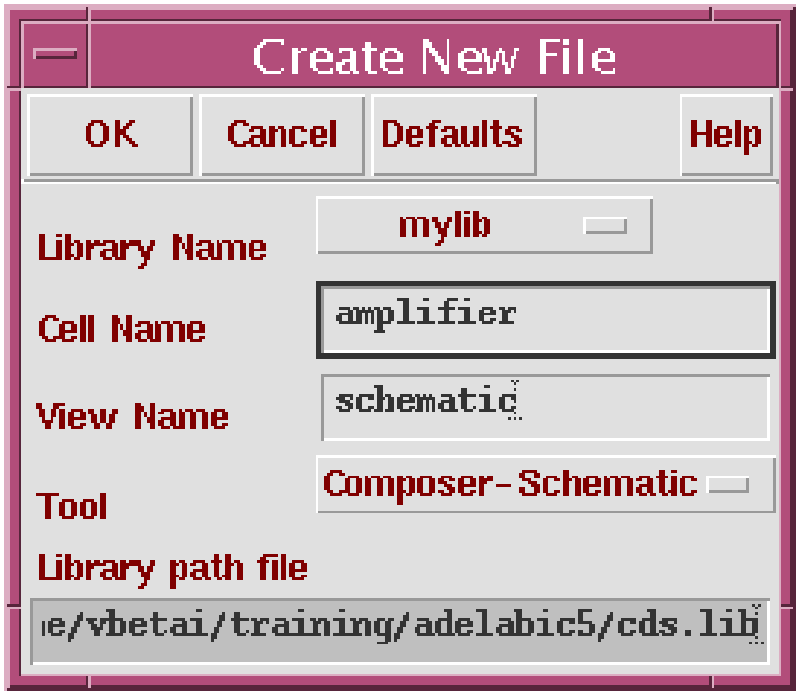
Contents of a Schematic



- Component instances represent instantiations of other cellviews in this cellview.
- Instance labels display component information in the design entry window.
- Pins can be inputs and outputs of a schematic or connection point when an instance is placed in another cellview.
- Wires can be drawn between pins to connect them.
- Wire labels provide meaningful signal identifiers for simulation results analysis.
- Analog taps and sources can be included directly in the design.

Creating a New Cellview

In the CIW or Library Manager, select **File—New—Cellview**.



Default View Name subject to override.

Select: Composer-Schematic

- Specify the Library Name, Cell Name, View Name, and Tool to use. The path to the *cds.lib* file will appear in the form and is not editable.
- Modify the Tool field to create a layout, verilog, symbol, schematic, vhdl, or ahdl view.

For an ADE schematic, select **Composer-Schematic** from the Tool cyclic field

Create a new cell views from the Library Manager or CIW.

Specify the Library Name, Cell Name, View Name, and Tool to use. The path to the *cds.lib* file will appear in the form and is not editable.

Modify the Tool field to create a *layout*, *verilog*, *symbol*, *schematic*, *vhdl*, or *ahdl* view. For a schematic, use *Composer-Schematic*. This will automatically enter *schematic* into the View Name text field.

Important

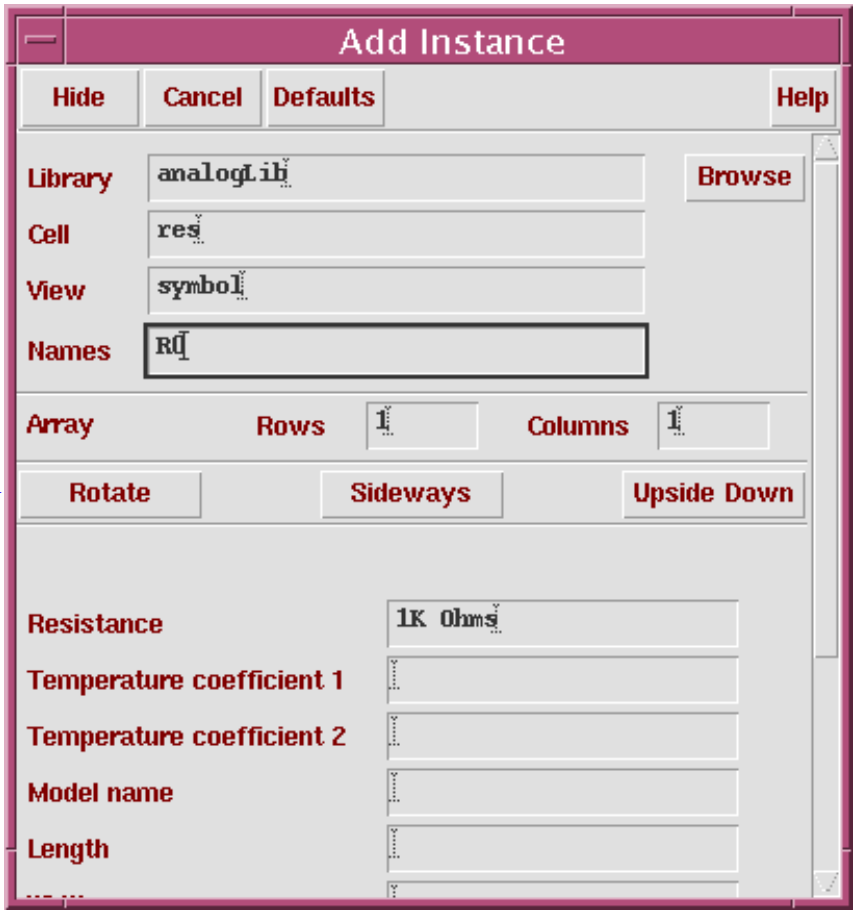
Although *schematic* is automatically entered into the View Name text field by default, you have the option to name the view anything you want. For example, the View Name may be altered to *schem1*, and the process repeated for *schem2*. As such, there are two or more schematic views for the same cell name. This allows parallel circuit designs within the same cell name. A common symbol for both schematics is then used within the hierarchy, until the final schematic is selected. This feature is useful in exploratory designs where the final circuit topology has not yet been finalized.

Adding Component Instances

Select **Add—Instance** or press the **i** key to display the Add Instance form.

- Attach multipliers to values. Enter **1k** (not 1 k) so that *k* is not mistaken as a variable.
- Parameter units, such as *ohms*, are implicit.

Use these buttons while placing components to control orientation.



Design components are generally instances of a *symbol* cellview and might be design primitives. Here are some properties associated with design component instances:

Parameter	Example Value
Library Name	analogLib
Cell Name	res
View Name	symbol
Instance Name	R2

The Instance Name is assigned automatically, unless explicitly specified.

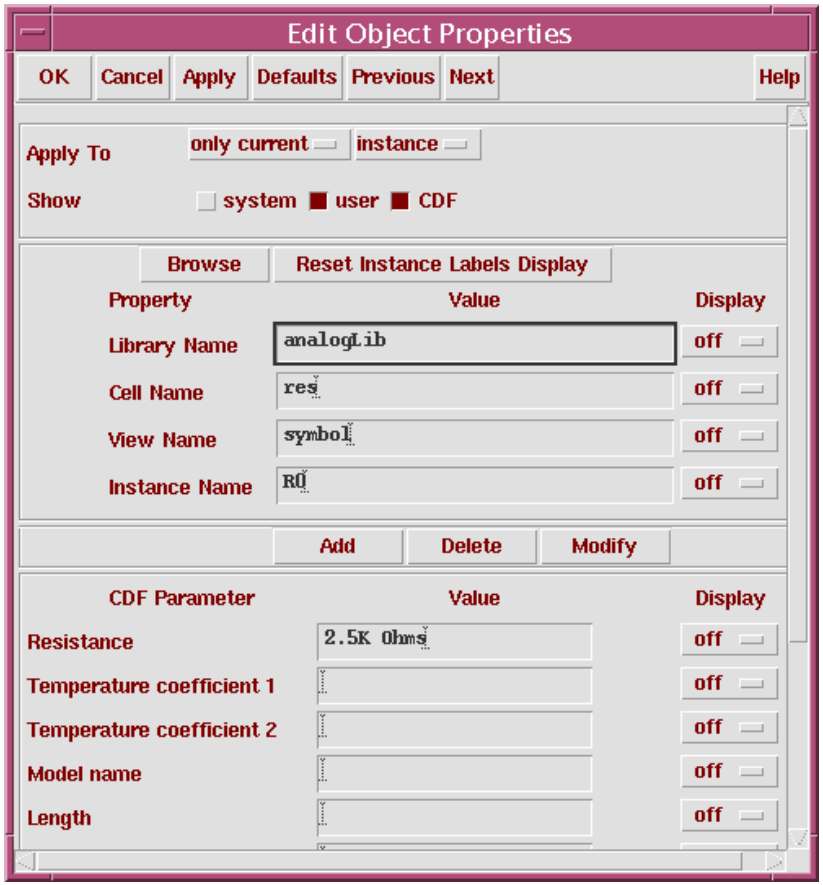
Find analog design primitives in the *analogLib* library. This library is included wherever the Virtuoso Analog Design Environment software is installed in the path *<install_dir>/tools/dfII/etc/cdslib/artist*. Include this path in your library search path to use *analogLib* components.

The system prompts for component parameters when instantiating the components. Attach multiplier suffixes, such as **k** for 1000, to numerical quantities.

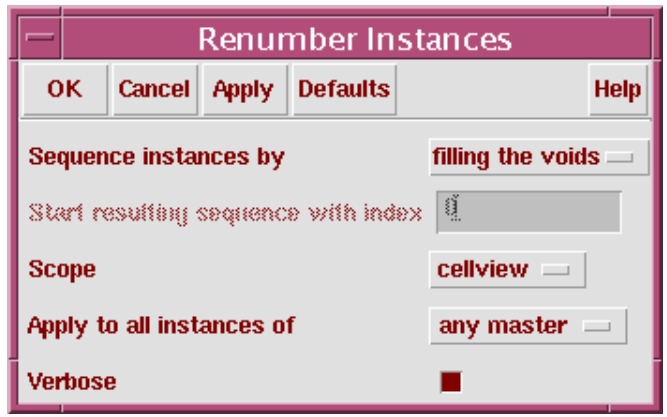
Use the **Rotate**, **Upsidedown**, and **Sideways** buttons to change the orientation of your components as they are placed in the schematic.

Updating Design Objects

- Select **Edit—Properties—Objects** or bindkey **q** to start the form.
  The **Next** and **Previous** buttons highlight single objects in a selected set.
- Use **Design—Renumber Instances** to renumber instances in a design.



Edit—Properties—Objects



Design—Renumber Instances

You can either update single or multiple objects in a selected set. Use the *Next* and *Previous* buttons on the Edit Object Properties form to scroll through a set of selected objects and update them. Only one object at a time will highlight in the schematic window. It is possible to modify most quantities that appear on the form.

The most common changes concern components parameters, pin name, and pin direction.

In addition, use the Stretch, Copy, Move, Delete, and Rotate commands to update your design. These commands are located under the **Edit** menu in the schematic window. To display an options form that is associated with any of these commands, use the **Cmd Options** icon or press the **F3** key while these commands are active.

Use the Renumber Instances form to renumber instances. This form renumbers component names in sequential order to make it easier to track component totals. In addition, adding and deleting components in a schematic during the design process can leave components labeled improperly.

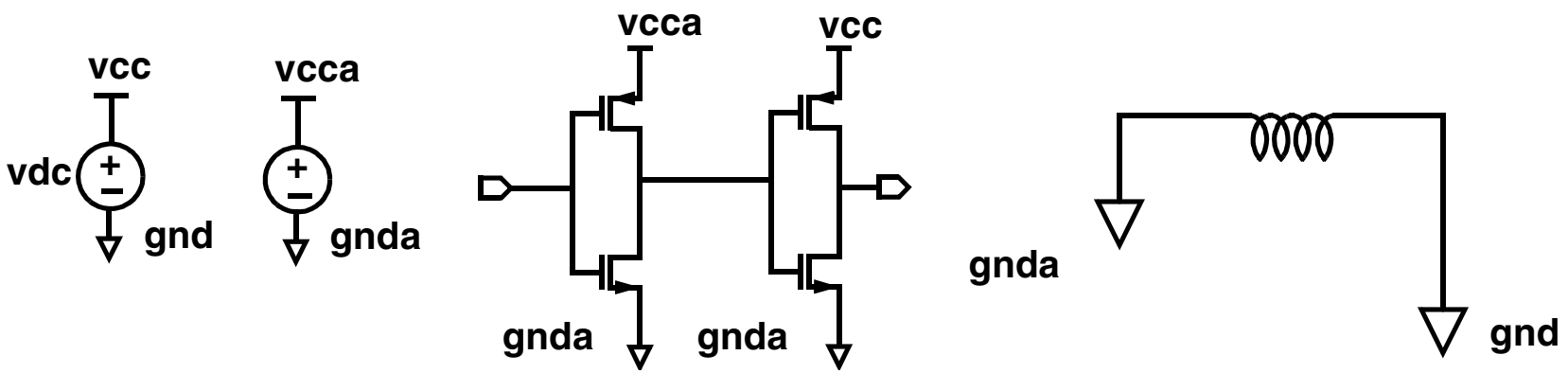
Note: The renumber sequence depends on the order that the symbols were added to the schematic.

Adding Sources and Ground

Sources, taps, and grounds are instances of cells.

Sample source cells are in the *analogLib* library.

- Choose from independent, dependent, and piece-wise linear (PWL) sources.
- Choose tap and ground cells, which are used to establish global nets.
- An instance of the cell *gnd* is required in the design for DC convergence.



- Ground

  Always include the symbol *gnd* (found in the *analogLib* library). Analog simulators require that all nodes in the circuit must have a DC path ground. This would be represented as *node 0* in the Cadence SPICE circuit simulator, for example. Use other ground symbols, such as *gnda*, for a ground that is connected to the reference ground through an analog circuit.

- Voltage Sources

  Include all of your DC and transient voltage and current sources in the schematic. There are many types of voltage sources in *analogLib*. For example, some of the independent voltage sources are *vdc*, *vsin*, *vpulse*, *vexp*, *vpwl*, and *vpwlf*. Each source has a current equivalent that begins with the letter ***i***. There are also equivalent dependent sources.

  All sources generate input waveforms except for *pwlf* sources, which stimulate a circuit using a text file of data tables. It is not necessary to include sources in the schematic, although this is often convenient. Attaching a stimulus file to the final netlist is discussed in the analog simulation section of this course.

- Voltage Taps

  Use tap symbols to transfer voltages and currents throughout the design without using wires. Voltage tap symbols, such as *vcc, vdd, vcca,* and *vccd,* are in the *analogLib* library.

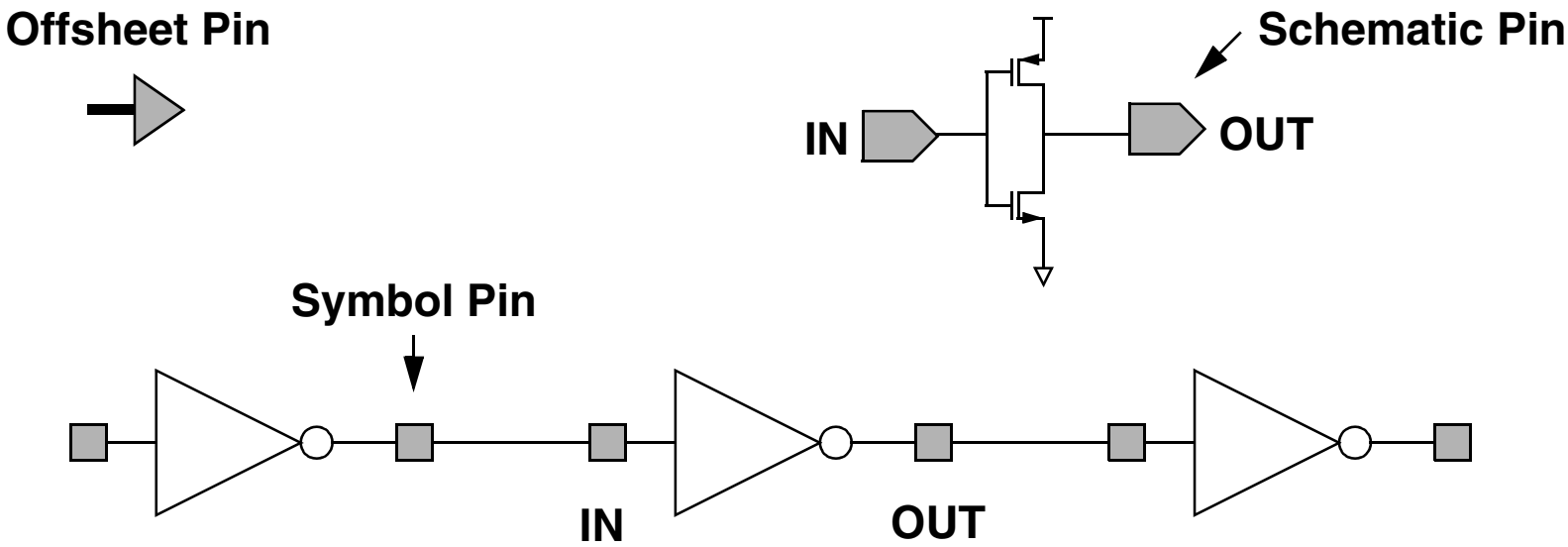
Pins

Pins have a user-defined *Name* and a *Direction* (input, output, or input/Output).

Pins are one of three *types*:

- *Schematic* pins provide ports to a schematic.
- *Symbol* pins provide ports to a symbol representing a schematic, and are connection points to the symbol in a hierarchical design.
- *Offsheet* pins are used in large designs without hierarchy.

Pin names and directions must match in all cellviews of a cell.



For analog designers, pins have two primary functions:

- Pins represent connection points between different cell views such as schematic, symbol, and layout representations. Using named pins identifies equivalent input, output, and I/O ports throughout the design environment.
- Pins provide connection points for objects that are hierarchically instantiated.

Pin Properties

Pins have a pin name, pin type, and pin direction. These should be consistent throughout your design.

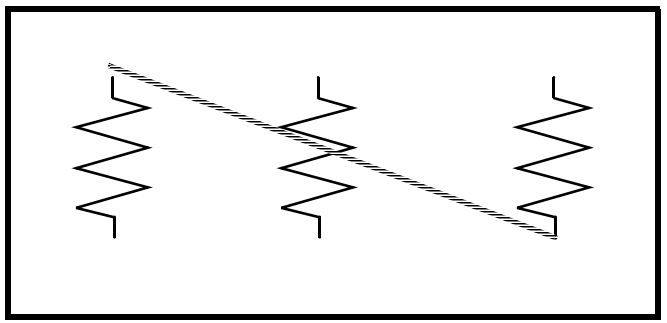
Multiple Sheet Design with Offsheet Pins

The *Virtuoso Schematic Editor User Guide* manual includes a section on multiple sheet design methodology and information on the *offsheet* pin type. You can get other help for the Schematic Editor software in the Cadence online help.

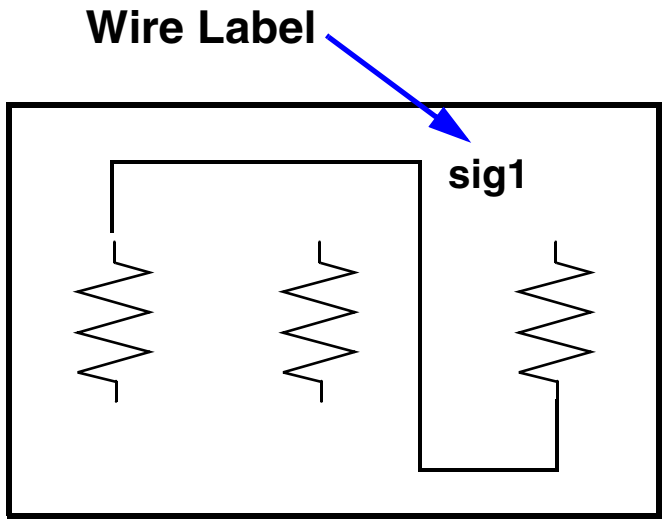
Pins (*ipin*, *opin*, *iopin*, *sympin*) now come from “basic” library.

Wires and Wire Labels

Automatic routing is the default mode.



Route Entered



The System Routes

When not labeling a wire, the system names the net formed by the wires.

If the router cannot find a path between two points,

- A dotted “flight line” is placed to establish connectivity only.
- Click on intermediate points to guide the router to yield a solid line of connectivity.
- Use the **Cmd Options** icon or **F3** key to modify the wiring options.

Draw wires between the instance pins and schematic pins to connect them. Use wide wires to indicate multiple signals on a wire, the system does not force or check this. Draw wires at any angle, but most designers frequently restrain wires to orthogonal lines.

- Using Route Methodology

  The route draw mode chooses two points in your design and then it automatically routes a wire around components. If a routed net remains dotted, it is because there was no clear routing channel. This can happen if the instances are too close or overlap the selection boxes. To solve this, move the components further apart to give a routing channel.

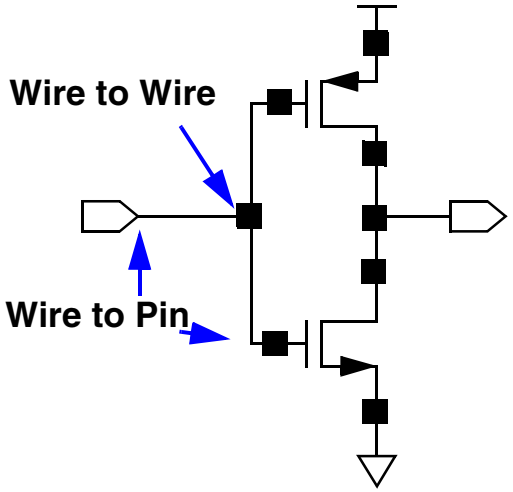
  Route method options exist to wire together two points immediately (the default) or indicate many points to route together later in a single step. More information on route methods is included in the design entry reference documentation.

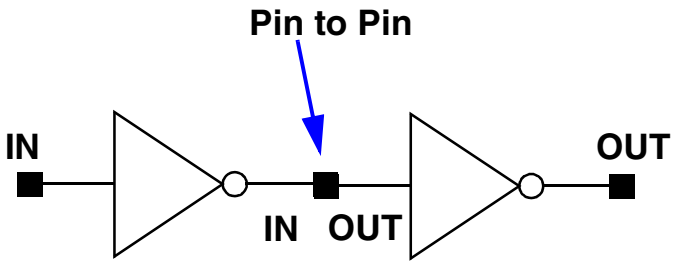
- Wire Labels

  Labeling wires gives the corresponding net a meaningful name in the simulation results data. Otherwise nets are system named. There is some control over the automatically generated names, but these may not be as meaningful as custom names.

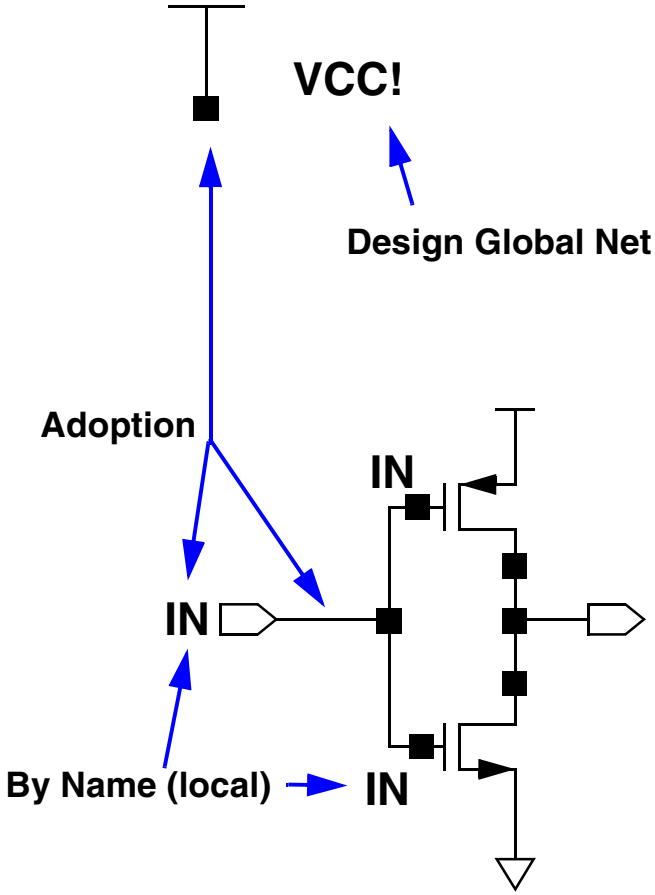
  Click the **Cmd Options** icon in the schematic window or press the **F3** key to change the default wiring setup.

Interconnecting Components





Avoid this when possible.



Schematic pins and global symbol pins name wires by adoption.

Note: Inherited connections (not shown) are discussed in Module 14 of iModule 5.

- Physical Connectivity

  All physical connections are made by wire-to-pin, wire-to-wire, or pin-to-pin connections.

- Connectivity by Name

  If two wires have been labeled with the same name, they become part of the same net when connectivity is established.

- System-Assigned Names

  If a net is unnamed, the system generates a name such as *net100* or *net7*. Optionally change the base name from *net* to something else. If a wire is connected to a schematic pin, then the pin name is used to name the net by adoption when connectivity is established.

- Global Nets

  Any net or pin name that ends in an exclamation point will be part of a global net when connectivity is established. Global nets are automatically connected through the hierarchy without the use of wires. For example, voltage taps have symbol pin names that end in an exclamation point. If a wire is connected to a pin that has a global name, the pin name is used to name the net by adoption. This is how voltage and ground signals are propagated throughout a design.

Note: A net named *net!* is not connected to a net named *net*.

Schematic Checking

During schematic checking, all of the following are performed by default:

- Update Connectivity

  This process associates wires and pins with logical connections called nets.

- Schematic Rules Check
  - Logical checks
  - Physical checks
  - Name checks

- Cross-View Checker

  This option checks for pin name and direction consistency between cellviews.

Execute **Check—Rules Setup** from a schematic window to edit the rules. Disable any or all of these schematic checking features, if not needed.

Schematic checking is a critical step in the design process.

Either check a single cellview or descend through the hierarchy to check all cellviews in your design.

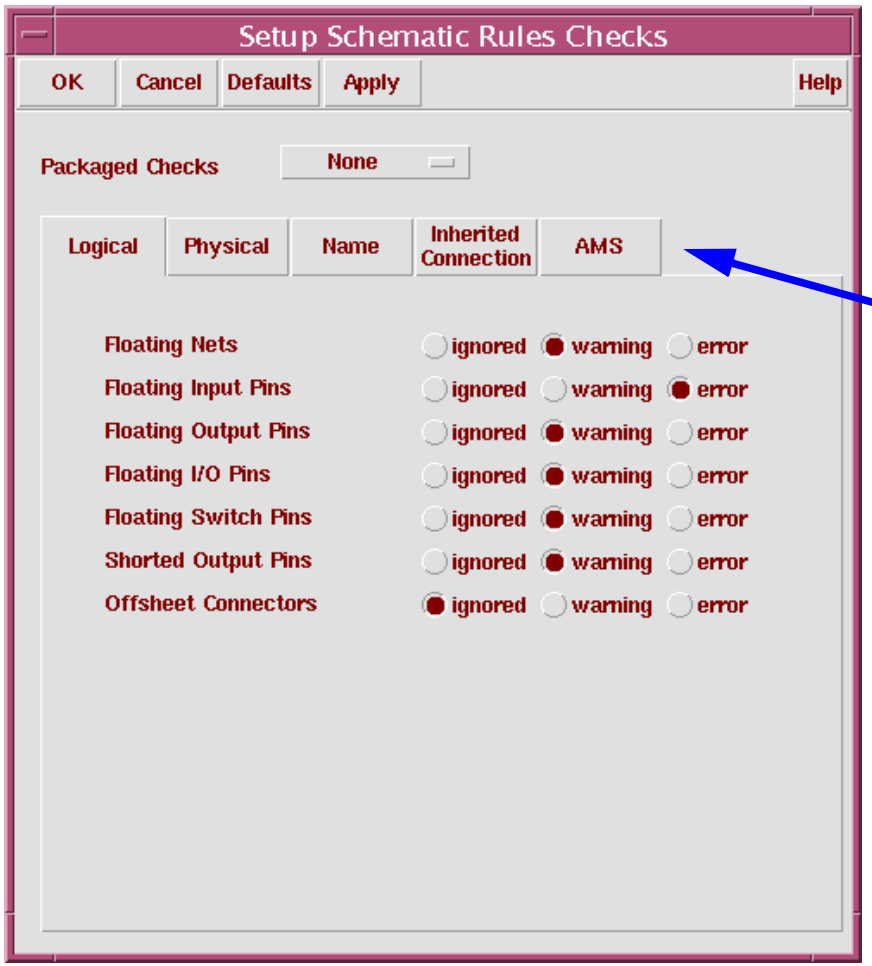
Checking a schematic accomplishes the following:

- Update Connectivity—When connectivity is established, wires and pins in the design entry window become associated with logical connections called nets. It is necessary to correct connectivity problems prior to going on to the next design phase.
- Schematic Rules Check—This process checks the schematic with a set of rules. Access them with the **Check—Rules Setup** command from the schematic window.

  The checks include:
  - Logical checks, such as *Floating Input Pins* and *Shorted Output Pins*.
  - Physical checks, such as *Unconnected Wires* and *Overlapping Instances.*
  - Name checks, such as *Instance Name Syntax.*
- Cross-View Checker—This option checks the pin consistency between different views of the cell. Pin names and directions must match between cellviews.

Schematic Checking Rules

The system sets the default schematic checking rules. The following set and selections are logical types of rules:



Note:
There are five sets of Rules Checks as indicated by the tabs.
- Logical
- Physical
- Name
- Inherited Connections
- AMS

Note: *Ignored* means do not check for a condition. It is permitted to generate a netlist and run a simulation with *warnings*, but not with *errors*.

The Schematic Rule Checker (SRC) performs schematic syntax checks. Select and override the default values of schematic rule checks. The defaults are acceptable for most applications. Select and set the severity level for SRCs. There are three levels for each check:

ignored	SRC does not perform the check.
warning	Warnings do not need to be corrected before continuing. SRC treats schematic connectivity as valid. The netlist program can still read the schematic.
error	Problems must be corrected before continuing. SRC treats schematic connectivity as invalid. The netlist program refuses to read the schematic.

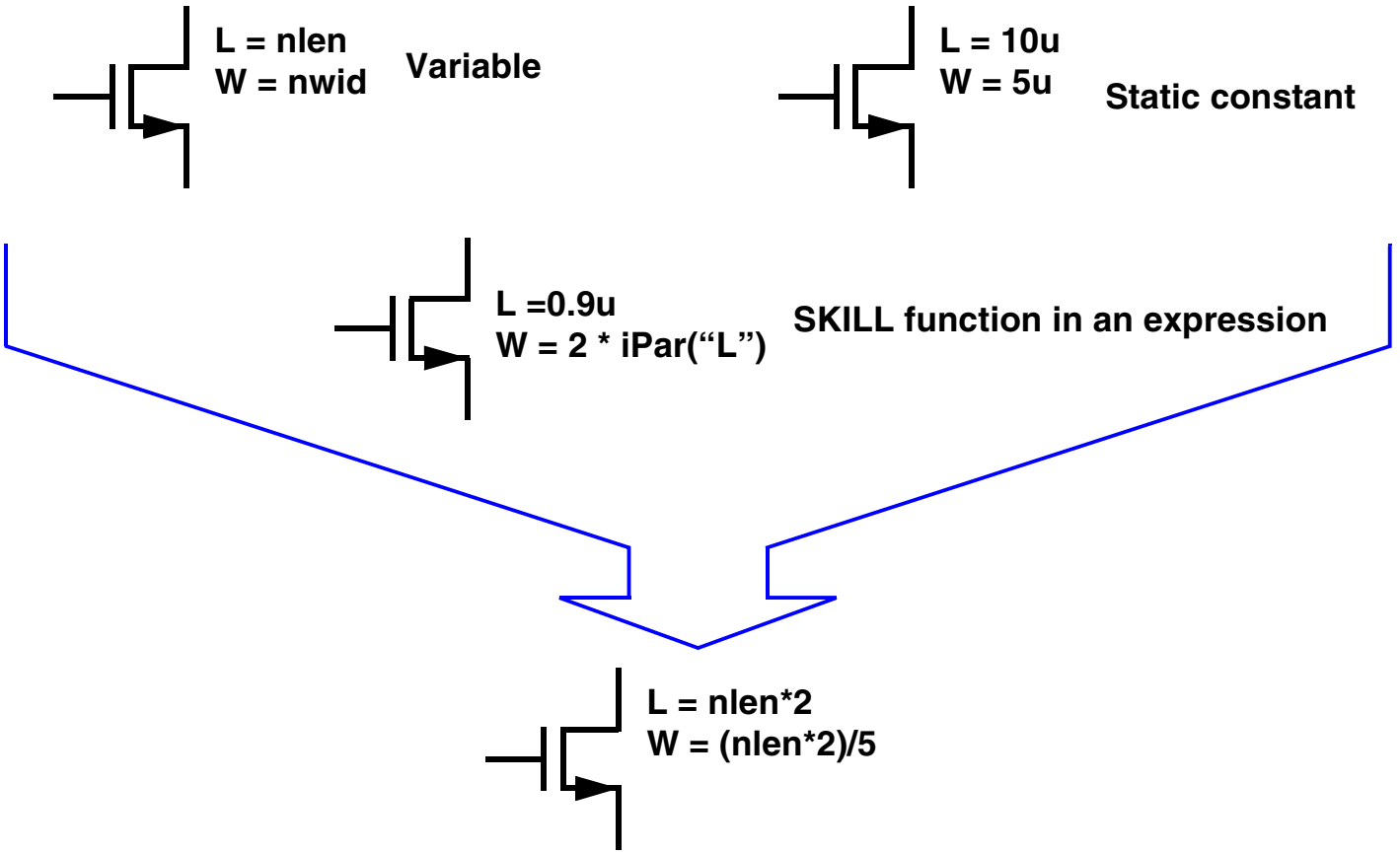
The checks are done for three different areas:

Logical checks	These checks consist of component connections that could affect the functionality of the circuit.
Physical checks	These checks consist of problems dealing with overlapping components, unconnected wires, and solder dots.
Name checks	These checks consist of problems associated with name syntax, behavioral model syntax, etc.

A description of the rules is in the *Virtuoso Schematic Editor User Guide*.

Component Parameter Types

Use user-defined functions to describe parameters.



Note: iPar is only used on the same instance where the iPar is.

Some basic types are static constants, global variables, and dependent variables. Use these in combination with a mathematical expression to create parameters.

To change a value during simulation, assign a variable name. Before running simulation, set design variables. All quantities assigned the same variable name will get the same value.

Hierarchical variables, user-defined functions, and user-defined constants are discussed later.

SKILL Functions

Use built-in SKILL functions that return design parameter values and use them in expressions to set component parameters. For example, to make the width of a MOS device a function of its length. In this example, if the length of a component is defined as *L*, then the width can be set as

```
w= 2 * iPar("L")
```

to make the width of the device twice the length.

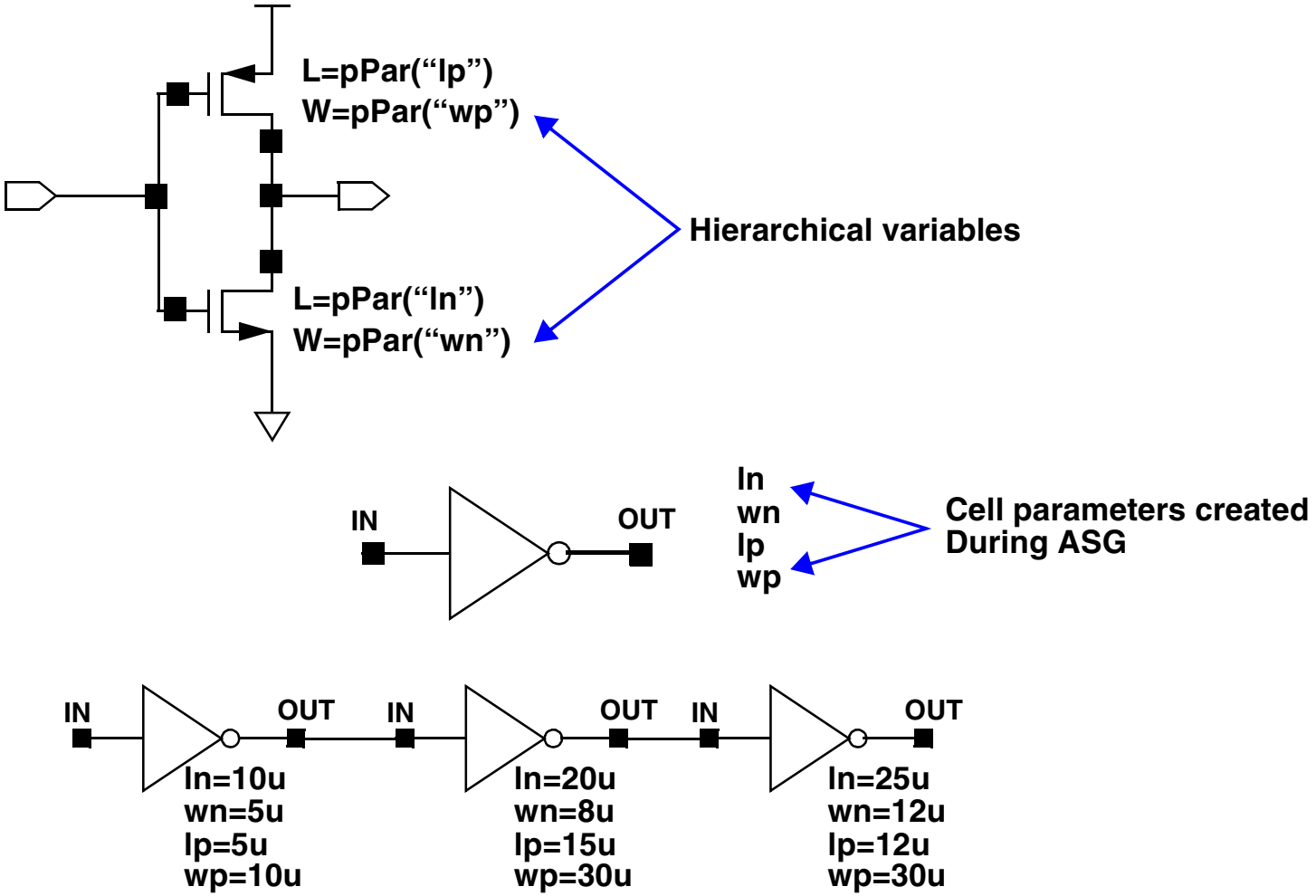
In general, the SKILL function

```
iPar("parameter name")
```

returns the value of a component parameter of the local cell.

More information on *pPar* and *iPar* is found in CDSDoc.

Passing Parameters Through the Hierarchy



During Automatic Symbol Generation (ASG), hierarchical variables are scanned. The system creates component parameters for the symbol from these variables.

Parameters become editable when the instances are selected.

You can place the same symbol many times and alter its schematic component parameter values at the instance level.

To accomplish this, assign expressions to schematic components using the following syntax:

```
pPar("variable")
```

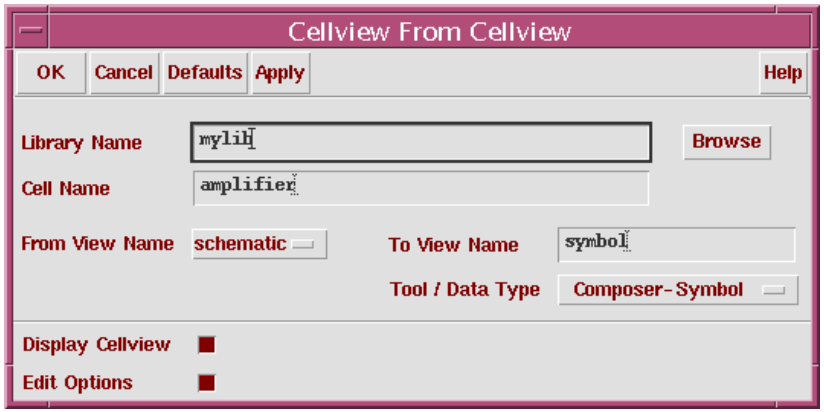
The value of the *variable* will be passed from the symbol level in the hierarchy to a component parameter in the schematic. Set the value of the *variable* when placing the symbol that was automatically generated from a schematic with hierarchical variables. During Automatic Symbol Generation (ASG), the system analyzes the hierarchical variables to determine what component parameters to prompt for when placing a symbol in another cellview.

Automatic Symbol Generation (ASG)

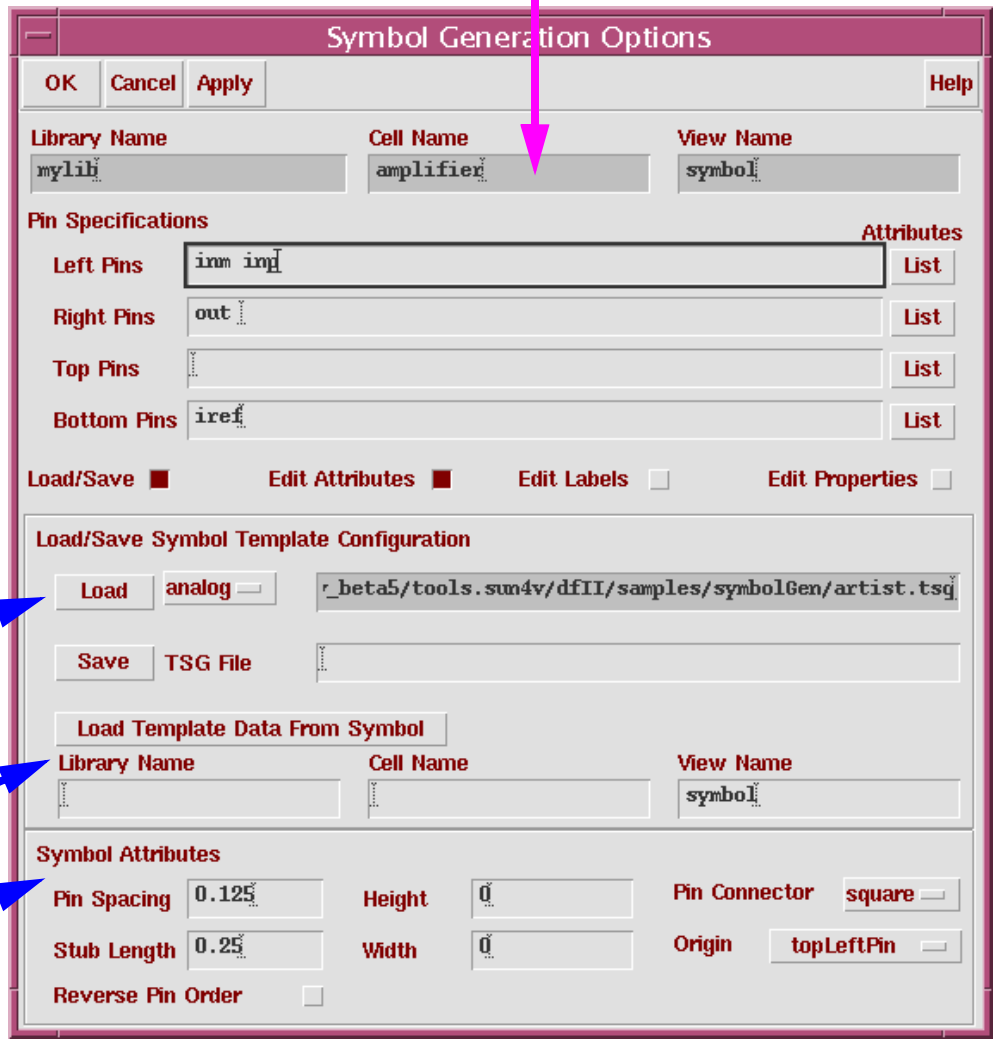
Automatic symbol generation assists in the creation of symbols. The quickest way to automatically create a symbol is from another cellview. Creating a symbol from an existing cellview also ensures that the pin properties will match between the cellviews. Other advantages to ASG are the automatic creation of symbol parameters from hierarchical variables in the schematic and the creation of CDF for the cell.

Symbol Generation

Design—Create Cellview—From Cellview



Select Apply or OK



This form opens with only the top portion, press these buttons to extend the symbol generation capabilities of the form.

Select a symbol generation template, other than the .cdsinit entry.

Select symbol generation attributes to control the symbol drawing.

Bring up ASG from the schematic window. A template file will be used for the symbol creation. There are different symbol template files for different tools in the Design Framework II environment.

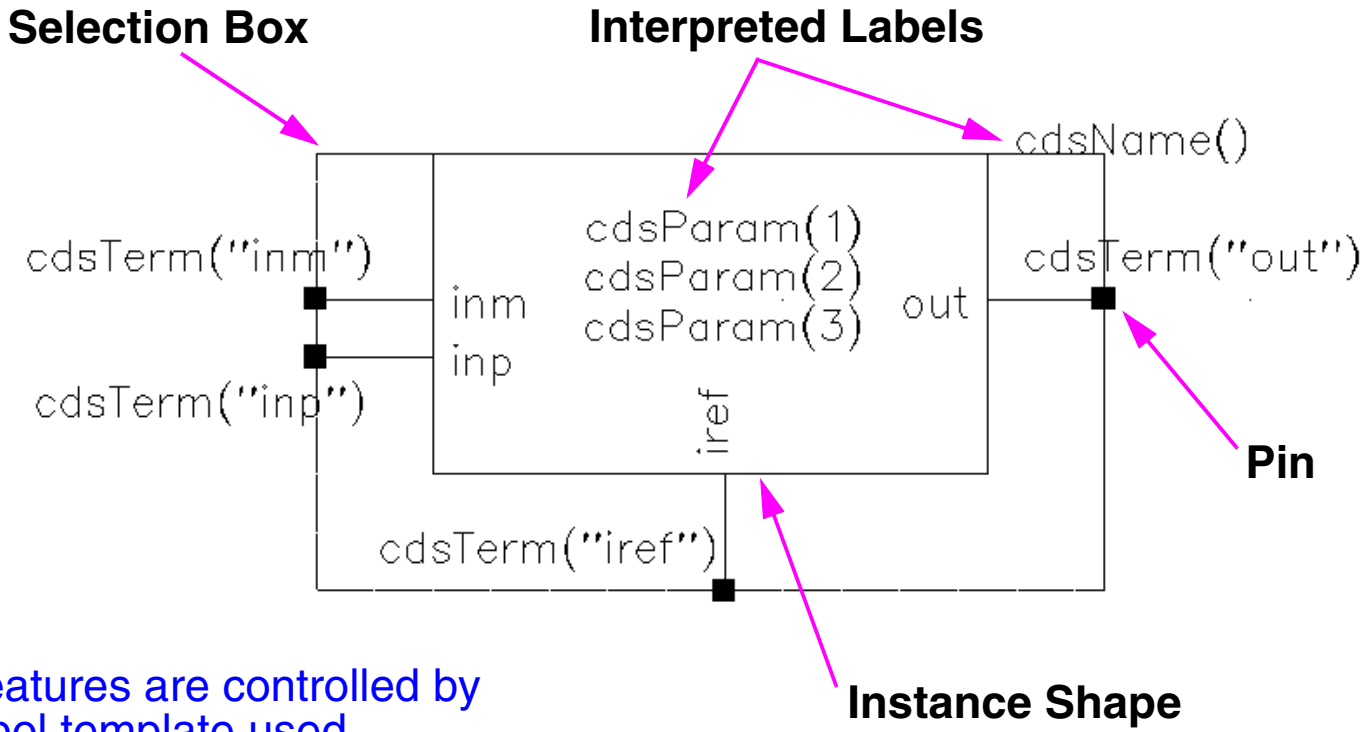
To make sure the *analog* symbol generation template is used in your design, put the following command in your *.cdsinit* file:

```
schSetEnv( "tsgTemplateType" "analog" )
```

Note: Failure to set this will result in a digital symbol generation.

Notice that the From View Name and To View Name fields can be modified in the Cellview from Cellview form. This provides a way to create other views, such as *behavioral* or *ahdl* from your schematics.

Characteristics of an Automatically Generated Symbol



These features are controlled by the symbol template used.

Interpreted labels on the symbol act as “placeholders” for different types of information to be displayed in the schematic.

- *cdsTerm()* labels display pin names or the net names.
- *cdsParam()* labels display parameters of an instance.
- *cdsName()* labels display the instance or cell name.

An automatically generated symbol cellview includes pins, a rectangular graphic and labels. It can be modified using the symbol editor.

Note: The generation depends on the symbol template selected either in .cdsinit or by the symbol generation form.

There is some control over how automatically generated symbols are drawn. By default, pins are placed at the left side of the symbol if their direction is input, at the right side of the symbol if their direction is output, and on top of the symbol if the direction is InputOutput. Options exist to change the pin appearance and order of the pins.

Interpreted Labels allow information to appear near the symbol after it is placed in another cellview. For the label generation template, the three label types are:

- *cdsTerm()* labels display pin names or the net names pins connect to.
- Each *cdsParam()* label can display a parameter of the instance. There can be multiples of this label.
- *cdsName()* labels display the instance or cell name.

All placeholder labels can be rearranged so that labels on instantiated instances are moved accordingly. These three labels have meaning in the Virtuoso Analog Design Environment. They display certain attributes, which are discussed later.

A selection box is drawn around the symbol and can be edited. It defines the symbol's selectable region after it is placed in another cellview.

Schematic Window Icons and Accelerator Keys

The Virtuoso Schematic Editor software provides both icons and “Accelerator” keys (bindkeys) to simplify schematic capture. The icons and Accelerator keys also reduce the time needed to capture and edit schematics.

- The icons appear on the left-hand side of the schematic editing window.
- An icon is activated by using a left click over the icon.
- Accelerator keys are activated by pressing specified keys on the keyboard.
- Accelerator keys are sometimes referred to as **bindkeys**.

The term “Accelerator” key and “Bindkey” are used interchangeably.

Schematic Editor Command Summary

Command	Bindkey	Command Sequence	Icon?
Add Component	i	Add—Instance	yes
Select Component(s)	LMB	left click, or drag LMB	
Copy	c	Edit—Copy	yes
Delete	Del	Edit—Delete	yes
Move	m	Edit—Move	
Stretch	M	Edit—Stretch	yes
Rotate	r	Edit—Rotate	
Repeat		RMB	yes
Modify Properties	q	Edit—Properties—Objects...	yes
Add Wire	w	Add—Wire	yes
Add Wire Name	l	Add—Wire—Name...	yes
Add Pin	p	Add—Pin	yes
Undo	u	Edit—Undo	yes
Redo	U	Edit—Redo	

Command	Bindkey	Command Sequence	Icon?
Zoom in by 2	]	Window—Zoom—zoom in by 2	yes
Zoom out by 2	[	Window—Zoom—zoom out by 2	yes
Zoom in	z	Window—Zoom—Zoom in	
Zoom out	Z	shift—RMB	
Fit	f	Window—Fit	
Redraw	f 6	Window—Redraw	
Check and Save	X	Design—Check and Save	yes
Save As	^s	Design—Save As	
Delete Marker	^g	Check—Delete Marker	
Descend Edit	E	Design—Hierarchy—Descend Edit	
Return	^e	Design—Hierarchy—Return	
Delete All Edits		Design—Discard Edits	
Close		Window—Close	

Bindkeys (Accelerator Keys)

Many of the schematic capture commands have alternative ways to be invoked.

1. A command sequence such as: **Edit—Properties—Objects**
2. An icon such as: 
3. A bindkey such as **q**

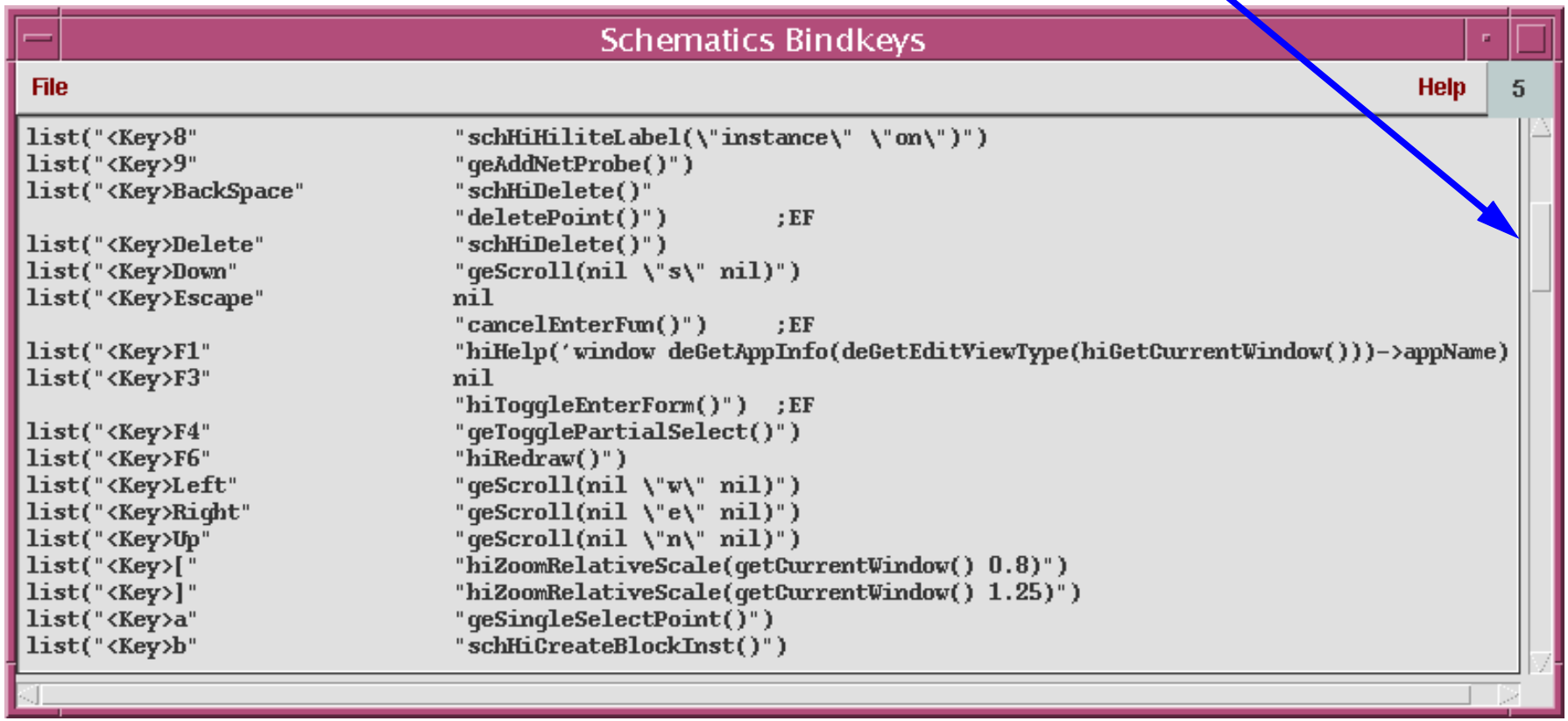
Bindkeys include the following features:

- Speed up schematic capture flow
- Default set of functions with installation
- Functions may be customized
- Full set of may be viewed or changed using **Options—Bindkeys** in the CIW

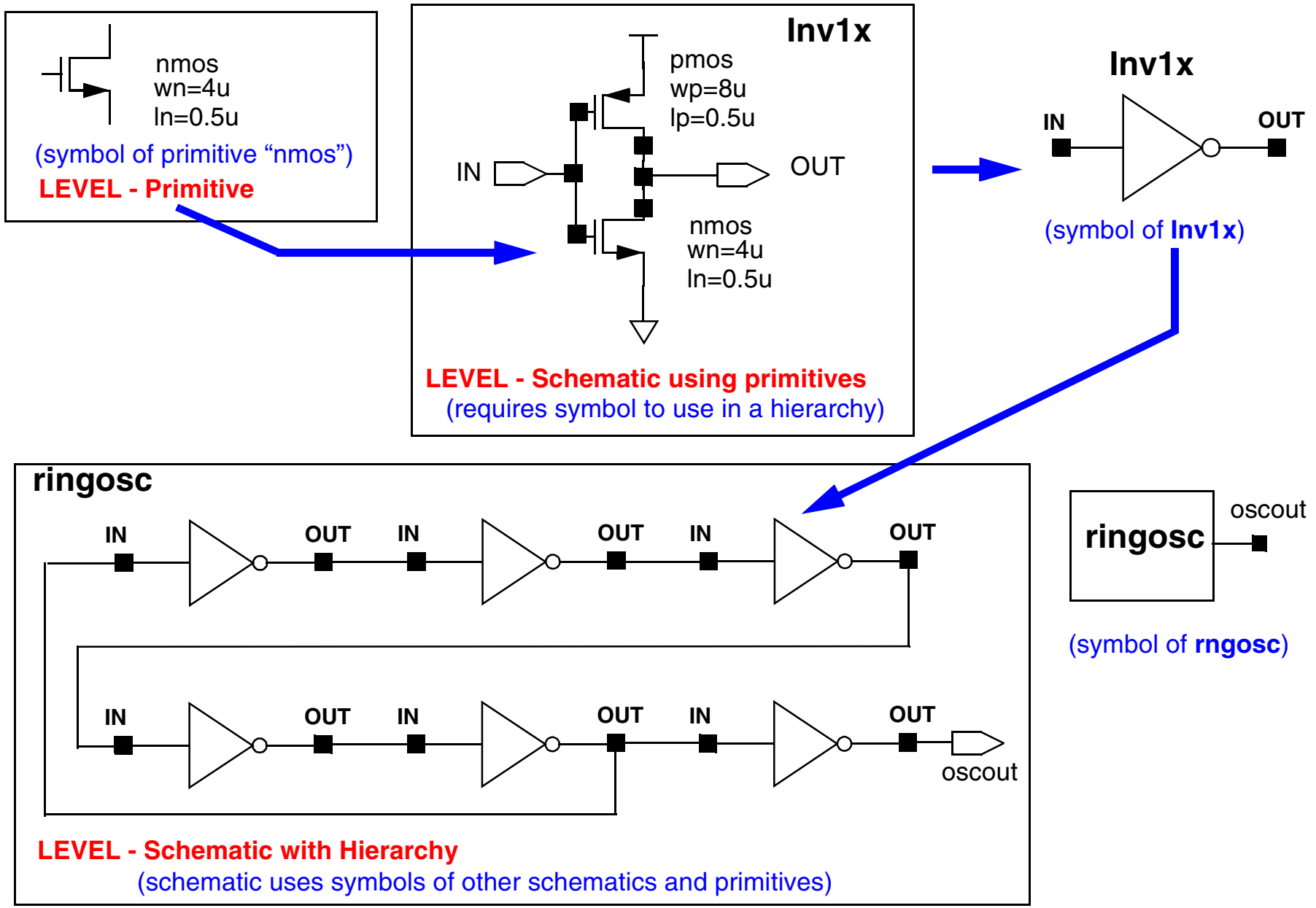
Bindkeys simplify the schematic capture flow. A default set of bindkeys is provided; however, the keys are programmable. To view the key settings and the corresponding SKILL syntax, select in the CIW: Options—Bindkeys. A **"Key or Mouse Binding Window"** appears. This window shows bindkeys for the schematic editor and other tools. In the Application Type Prefix cyclic field select *schematic*.

Then select the *Show Bind Keys* button. The Schematic Bindkeys window will appear.

Note: This is a partial list. To view all bindkeys use scroll bar.



Using a Hierarchy



A hierarchy is the design data of a complex system organized into simple and manageable data at different levels. A hierarchy simplifies the complex structure of a system. It often reduces the storage requirements for the data. It also simplifies and reduces the time to design the system.

Primitives are the basic design elements and exist at the bottom of the hierarchy. The design does not descend below this level. A schematic may consist entirely of primitives. Such a schematic is also referred to as a flat schematic or a primitive-level schematic. For large systems, for example a 16-bit analog-to-digital converter, it is difficult capture the design with a flat schematic.

A design of a complex system can consist entirely of a single schematic. The flat schematic can be simulated by including sources for power and stimulus. A design using only a flat schematic (without hierarchy) is inefficient when repeated structures are used. Such a schematic becomes difficult to manage as the circuit complexity increases.

A schematic can use symbols of primitives and symbols for other schematics. Such a schematic is more efficient to design repeated structures and complex systems. This use of symbols to represent schematics continues to higher and higher levels of hierarchy until the TOP level of the design is reached.

A symbol for a schematic view is only required when that schematic is used within a hierarchy.

Lab Reference Material: Mouse Buttons

Left Mouse Button—Select and Deselect	
Click	Select point
Double click	Extend select
Shift-click	Select point (add)
Control-click	Deselect point
Draw through	Select box or Direct Edit*
Shift draw through	Select box (add) or Direct Edit*
Control draw through	Deselect box or Direct Edit*
(EF)	Add point

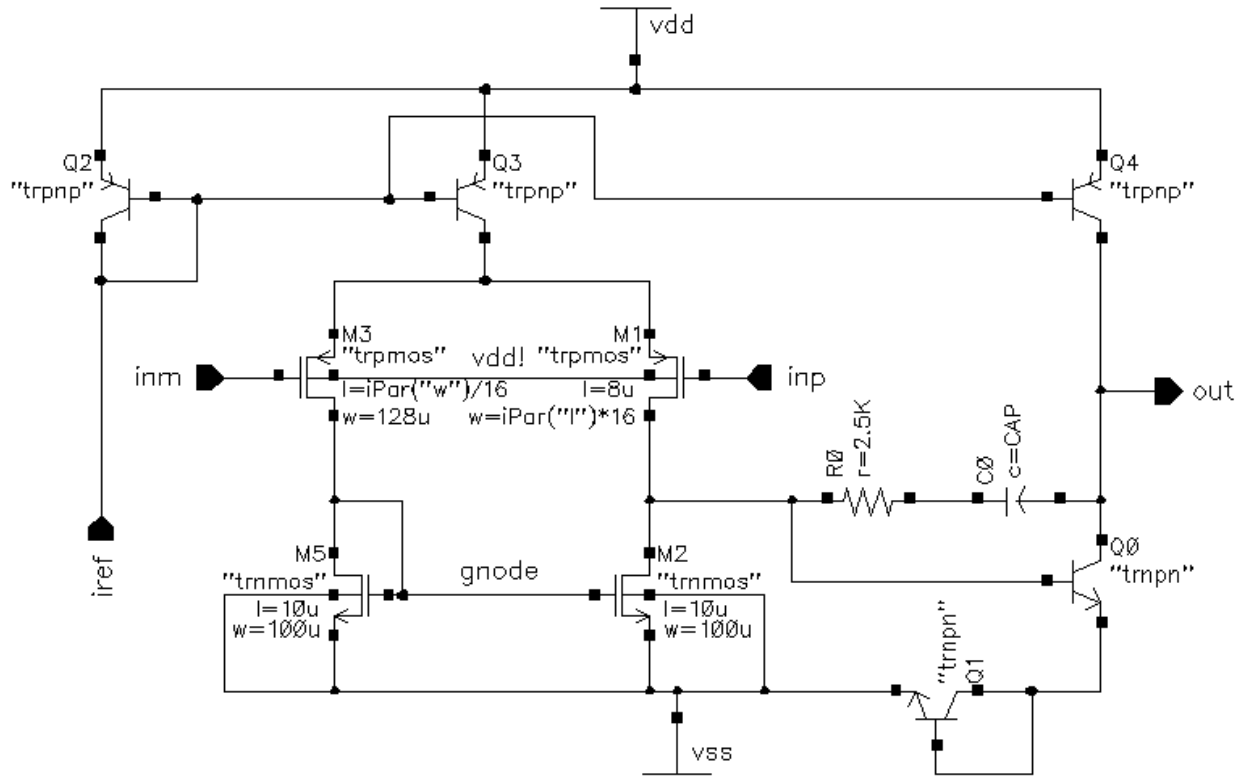
Middle Mouse Button Pop-Up Menus	
Click	Pop-up menus
(EF)	Pop-up menus

Right Mouse Button Repeat, Zoom, Options	
Click	Repeat last command
Draw through	Zoom in
Shift draw through	Zoom out
(EF)	Command options (command-specific bindings)

Note: EF (Enter Function) bindkeys used within an active command.
*Direct Edit applies only when over object.

Labs for Module 2

Schematic Capture



Lab 2-1 Schematic Entry

Objective: To create a library and build a Bi-CMOS Operational Amplifier.

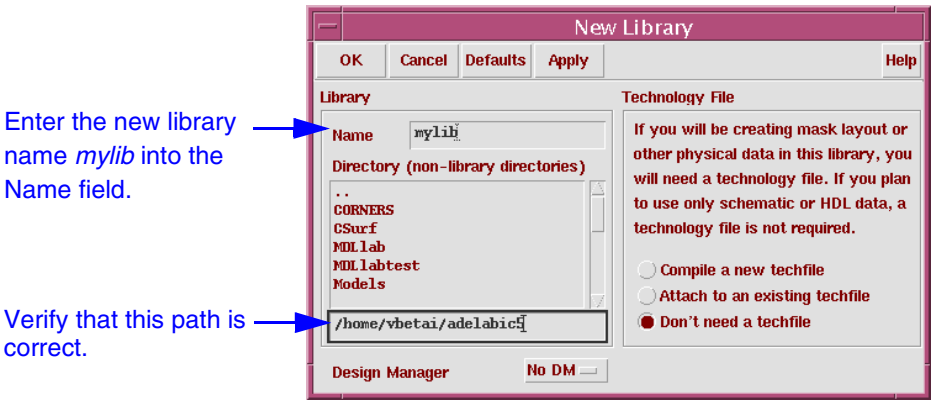
Create the *mylib* library to use throughout the course. Then build a Bi-CMOS operational amplifier in this library.

Creating a Library

1. In the CIW, execute **File—New—Library**.

   The New Library form appears.

2. In the New Library form, change **only** the following:



   In the field under the Directory section, verify that the path to the library is set to *~/adelab*ic5. (The ~ might be expanded to the full path name.) A technology file is not required for this lab, but can be attached after the library is created.

3. Click **OK** in the New Library form.

   Note: The lab exercises are written for a library name of *mylib*. If you choose a library name other than mylib, be aware of the inconsistencies as you work through the labs. It is possible, for example, to name your library *myLib*.

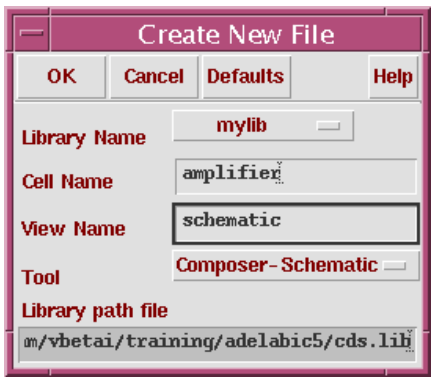
4. Look in the Library Manager window and verify that the *mylib* library is listed.

Creating a Schematic Cellview

Open a new schematic window in the *mylib* library and build the *amplifier* design. You will simulate this design later on.

1. In the CIW or Library Manager, execute **File—New—Cellview**.

2. Set up the Create New File form as follows:



   You cannot edit the **Library path file** and the one above might be different from the path shown in your form.

3. Click **OK** when done.

   A blank schematic window for the *amplifier* design appears.

Adding Components to a Schematic

Build the *amplifier* schematic shown below.

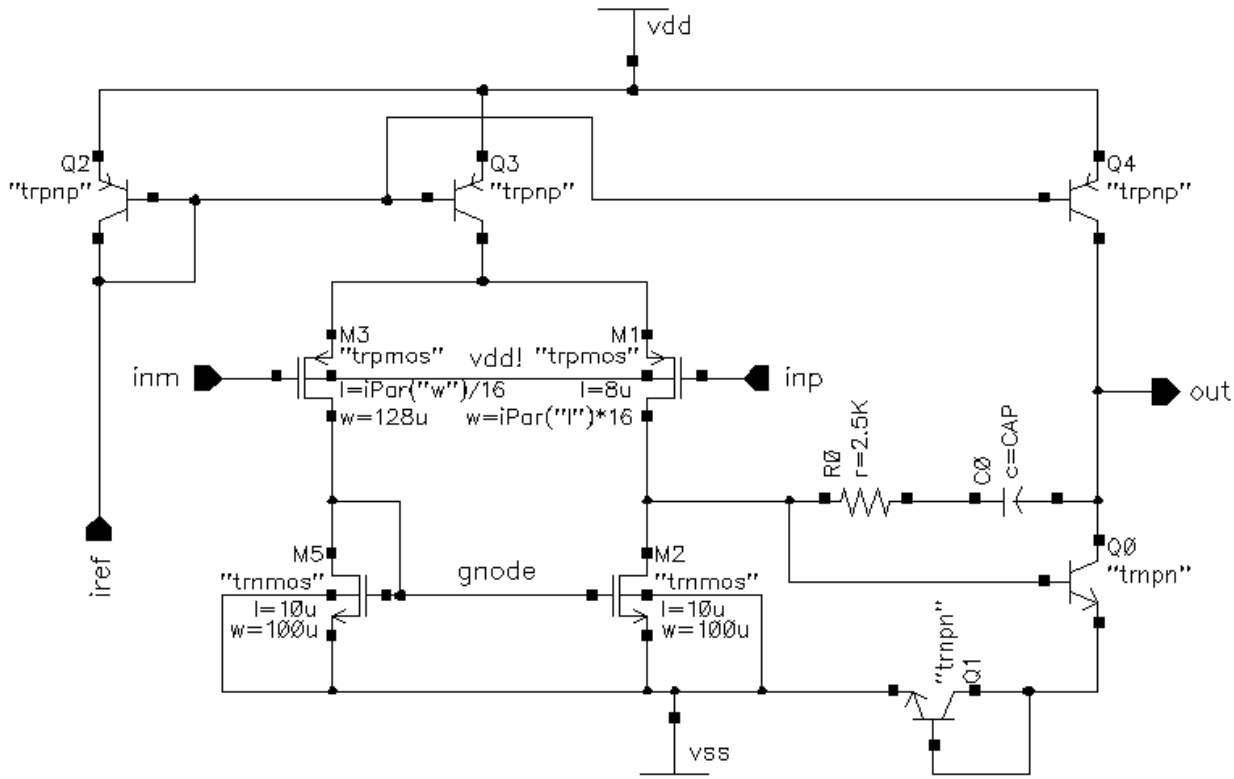
1. In the *amplifier* schematic window, click the **Instance** fixed menu icon to display the Add Instance form.

   Tip: You can also execute **Add—Instance** or press **i**.

   Make sure that the View Name field in the form is set to *symbol*. You will update the Library Name, Cell Name, and the property values given in the table on the next page as you place each component.

   After you complete the Add Instance form, move your cursor to the schematic window and click **left** to place a component.

   Another way to fill in the Add Instance form is to click on the **Browse** button. This opens up a Library Browser from which you can select components to place with your **left** mouse button.



Note: Instance Names in the picture, such as Q0, Q2, M1, and M2, might not match Instance Names in your schematic.

This is a table of components for building the *amplifier* schematic.

Library Name	Cell Name	Properties/Comments
analogLib	*pnp*	For Q2, Q3, Q4: Model Name=trpnp (no quotes)
analogLib	*npn*	For Q0, Q1: Model Name=trnpn (no quotes)
analogLib	*pmos4*	For M1: Model Name=trpmos (no quotes) Length=8u, Width=iPar("l")*16
analogLib	*pmos4*	For M3: Model Name=trpmos (no quotes) Length=iPar("w")/16, Width=128u
analogLib	*nmos4*	For M2, M5: Model Name=trnmos, Width=100u, Length=10u
analogLib	*res*	Resistance=2.5K
analogLib	*cap*	Capacitance=CAP (Design Variable)
analogLib	*vdd, vss*	

If you place a component with the wrong parameter values, use the **Edit—Properties—Objects** command to change the parameters. Use the **Edit—Move** command if you place components in the wrong location.

Note: Make sure you use the information shown above in the PROPERTIES/COMMENTS field. You might also need to rotate specific components as you place them in the schematic, especially transistor Q1. You can rotate components at the time you place them, or use the **Edit—Rotate** command after they are placed.

2. After entering components, click **Cancel** in the Add Instance form or press **Esc** with your cursor in the schematic window.

Adding Pins to a Schematic

Do not use the Add Instance form to place schematic pins. Use **Add—Pin** or the menu icon discussed next. Otherwise, the pin properties will not be correct.

1. Click the **Pin** fixed menu icon in the schematic window.

   Tip: You can also execute **Add—Pin** or press **p**.

   The Add Pin form appears.

2. Type the following in the Add Pin form, in the **exact** order shown, leaving spaces between the pin names:

Pin Names	`iref inp inm out`

   Make sure the Direction field is set to **input** and the Usage field is set to **schematic**.

3. Move your cursor into the *amplifier* schematic, and notice how an outline of a schematic input pin follows your cursor. Do **not** place the pin at this time.

4. Go back to the Add Pin window, and click on **Rotate**. Move your cursor into the schematic window. Notice that the pin is now rotated 90 degrees counterclockwise.

   You can also do this with your **right** mouse button.

   Note: The Add Instance form has options to rotate a component by 90°, or rotate the component around the X or Y axis. Before placing the component you can click right with the cursor in the design window to change the component orientation by 90°.

5. Refer to the diagram of the *amplifier* schematic, and click your **left** mouse button in the location in your schematic where you want the *iref* pin to be.

   Notice that *iref* is no longer in the Pin Names field of the Add Pin form.

6. You will now place the *inp* input pin.

   With your cursor in the schematic window, click **right** to rotate the next pin. Next, click **left** to place the *inp* pin in the proper location.

7. Next, place the *inm* input pin.

   Go back to the Add Pin form and click **Sideways**. Move your cursor into the schematic window and click **left** to place the *inm* pin in the proper location.

8. Finally, place the *out* output pin.

   In the Add Pin form, change the Direction field to **output**.

   Note: If you do not change the direction to output before placing the pin, you might encounter errors later on. You can always edit the properties of the pin if you make a mistake when placing it.

   Move your cursor into the schematic window and click **left** to place the *out* pin in the proper location.

9. In the schematic window, execute **Window—Fit** or press the **f** bindkey.

   Move or rearrange the location of your components if they are not in the right locations.

Adding Wires to a Schematic

Add wires to connect components and pins in the design.

1. Click the **Wire (narrow)** icon in the schematic window.

   Tip: You can also press the **w** key, or execute **Add—Wire (narrow)**.

2. In the schematic window, click on a pin of one of your components as the first point for your wiring.

   A diamond shape appears over the starting point of this wire.

3. Follow the prompts at the bottom of the design window and click **left** on the destination point for your wire.

   A wire is routed.

   Note: The wire router cannot always find a path between two points. If this happens, a dotted "flight line" is drawn between the points, and establishes connectivity. If flight lines occur in your lab, delete them and click on intermediate points to guide the router as your design is wired.

4. Continue wiring the schematic as shown in the example. The exact topology shown in the reference schematic is not as important as the connectivity.

   a. Use the **Cmd Options** fixed menu icon or press **F3** to see the Add Wire form. Modify the Draw Mode and Route Method fields for different effects.

   Tip: The fastest method is to set the Draw Mode to **route**, and the Route Method to **full**.

   b. With some design entry commands, options forms do not appear by default. Use the **Cmd Options** icon to view options, and make modifications.

   c. Vertically expand your schematic window if this icon is not visible.

5. When done wiring, press **Esc** with your cursor in the schematic window to cancel wiring.

Adding Net Names

You can add meaningful net names to specific wires in the design to help you identify signals in waveform plots and to debug your circuit. If you do not add net names, a name will be assigned automatically.

1. Click the **Wire Name** icon in the schematic window.

   Tip: You can also press **l** or execute **Add—Wire Name**.

   The Add Wire Name form appears.

2. In the Add Wire Name form, type the following in the Names field:

   ```
   vdd! gnode
   ```

3. Move the cursor into the *amplifier* schematic window.

   The label *vdd!* follows your cursor. Follow the prompts in the schematic window. The exclamation point (!) after the name *vdd* implies that this is a global node.

4. Click **left** just above the wire connecting the bulk nodes (opposite of the gates) of the *pmos4* transistors M3 and M1 as shown in the *amplifier* schematic diagram. Next, click on the wire itself.

5. Follow similar procedures to place the *gnode* label, which labels the node connecting the gates of the *nmos4* transistors M5 and M2.

6. With the cursor in the schematic window, press the **Esc** key.

Saving a Design

1. Click the **Check and Save** icon in the schematic editor window.

2. Observe the CIW output area.

Lab 2-2 Symbol Creation

Objective: To create a symbol for the Bi-CMOS operational amplifier.

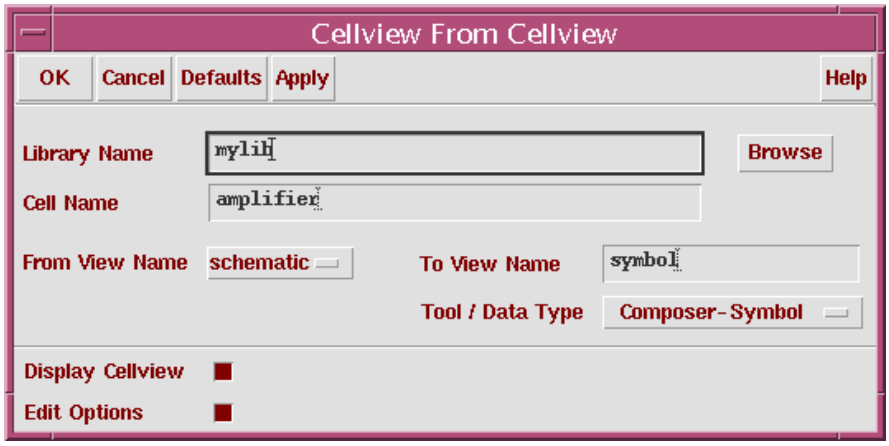
Creating a Symbol

In this lab activity, you will create a symbol for your *amplifier* design so you can place it in a test circuit for simulation. A symbol view is extremely important step in the design process. The symbol view must exist for the schematic to be used in a hierarchy. In addition, the symbol has attached properties (cdsParam) that facilitate the simulation and the design of the circuit.

1. In the *amplifier* schematic window, execute **Design—Create Cellview—From Cellview**.

   The Cellview From Cellview form appears. With the Edit Options function active, you can control the appearance of the symbol to generate.

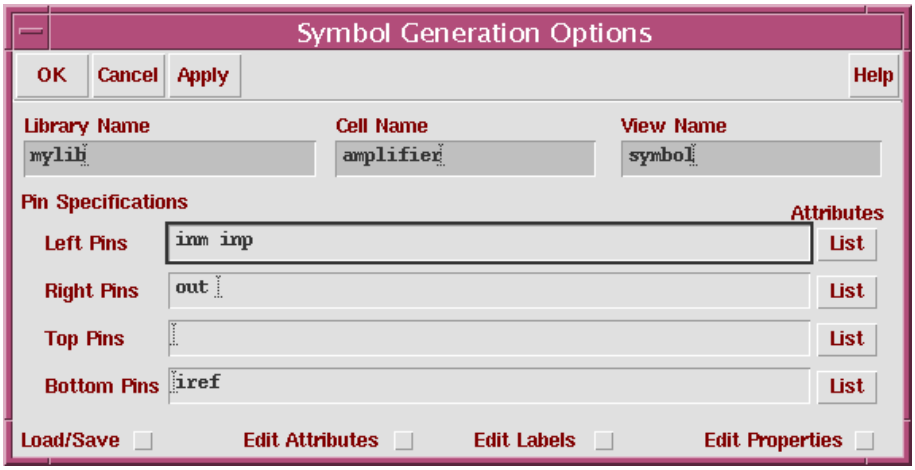
2. Verify that the From View Name field is set to **schematic**, and the To View Name field is set to **symbol**, with the Tool/Data Type set as **Composer-Symbol**.



3. Click **OK** in the Cellview From Cellview form.

   The Symbol Generation Form appears.

4. Modify the **Pin Specifications** as follows:

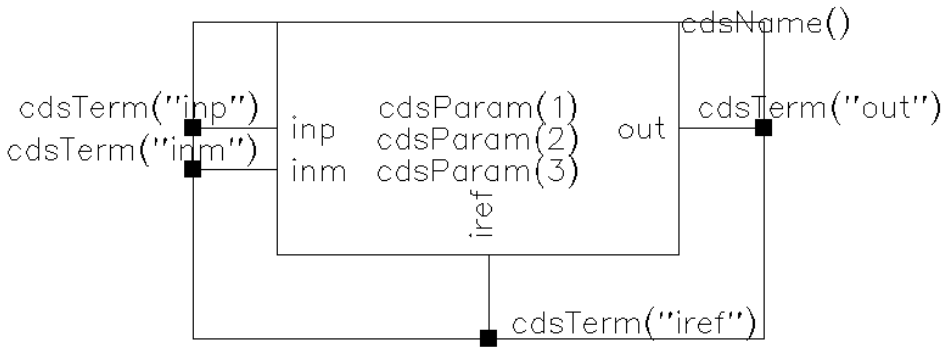


5. Click **OK** in the Symbol Generation Options form.

6. A new window displays an automatically created *amplifier* symbol as shown here.

   The resulting symbol uses an analog template that contains the proper interpreted labels and properties for analog simulation. You can use this template by default by modifying the local *.cdsinit* file using the directions in the sample *cdsinit* file. (In this class, the analog template is loaded through the *.cdsinit* file. View the *.cdsinit* file to see how this was done.)

   Note: You can also use the **Load/Save** option on the Symbol Generation form to load a particular template for the design session. The reference material has details.



7. Observe the CIW output pane and note the messages stating that an analog Component Description Format (CDF) was generated.

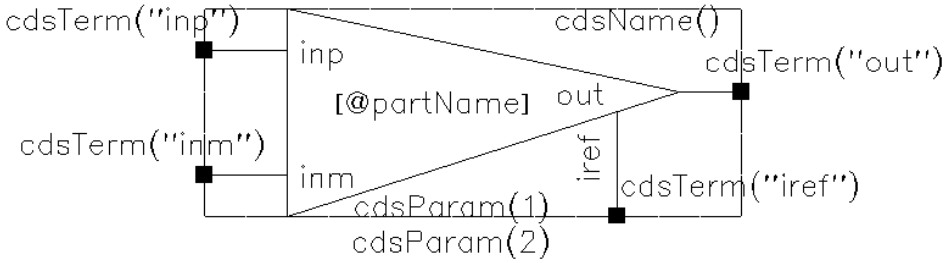
   In addition, you might see the following message:

   ```
   "Could not add or update the base lib CDF
   because it is not writable..."
   ```

   Ignore it for now.

Editing a Symbol

In the steps below, you will modify your *amplifier* symbol to look like this one. The shape of the amplifier shown below is more intuitive than the one that was automatically created.



1. Move your cursor over the symbol, until the entire green rectangle is highlighted (selected). Click **left** to select it.

2. Click the **Delete** icon in the symbol window.

3. Execute **Add—Shape—Polygon**. Follow the prompts at the bottom of the schematic, and draw the triangle shown in the final picture.

   Click left to place a side of your polygon. If you are unhappy with a line that you drew, use the **Backspace** key to undo the last line.

   Double click **left** when you have drawn the last point of your polygon to terminate the shape.

   The **u** bindkey will *undo* the entire shape if you make a mistake.

4. Draw a selection box around the *cdsTerm("inp")* label and attached pin by dragging the mouse with the **left** button depressed.

5. Click the **Move** icon in the schematic window.

   Follow the prompts at the bottom of the symbol window, and move the **inp** pin to its final destination.

6. Repeat steps 4 and 5 for the **inm** pin.

7. Repeat steps 4 and 5 for the **iref** pin.

   After moving the *iref* pin and wire, select the red, *iref* label. Use the **Edit—Rotate** command and follow the prompts to rotate the label.

   Once the label is rotated, **Move** it to the proper location.

8. Select the *cdsParam(3)* label and delete it.

9. Move your cursor over the symbol, until the entire red rectangle is highlighted (selected). Click **left** to select it. This is the selection box.

10. Click the **Delete** icon in the symbol window.

11. Click the **Selection Box** icon or execute **Add—Selection Box**.

12. In the Add Selection Box form, click **Automatic**.

    A new red selection box is automatically added.

13. Select and **Stretch** the *iref* pin to line up with the new selection box. You might need to move your *iref* and *cdsTerm("iref")* labels again.

14. Hold down the **Shift** key, and select both the *cdsParam(1)* and *cdsParam(2)* labels.

    Move the labels to the location shown in the diagram.

15. Select and **Move** the *cdsName()* label to its final destination.

Adding Text to a Symbol

You can add comments to any cellview for documentation.

1. Execute **Add—Note—Note Text**.

   The Add Note Text form appears.

2. Set these values only:

Note Text	`Amp`
Font Height	`0.0825`
Justification	`centerCenter`

   The Justification setting determines the origin of the text in relation to the cursor. In this case, the text is centered on the cursor and you can center the text on the symbol shape when you place it.

3. Move the cursor into the symbol window.

4. Place the label in the symbol window with your **left** mouse button, so that it is in the same location as in the final *amplifier* symbol diagram.

5. Click **Cancel** in the Add Note Text Form.

Saving a Symbol

1. Save your edited symbol view to disk.

2. Click the **Save** icon in the symbol editor window.

Preparing for the Next Lab

1. In the symbol editor, execute **Window—Close**.

2. If the *mylib* amplifier schematic window is still open, execute **Window—Close**, and click **No** in the Save Changes box if it appears.

Lab 2-3 Building the Supply Circuit

Objective: To build the supply cellview that will power the circuit.

You will combine both power sources needed to power the circuit into a single block named *supply*. You can set different values of **VDD** and **VSS** on this block when you place it in the design.

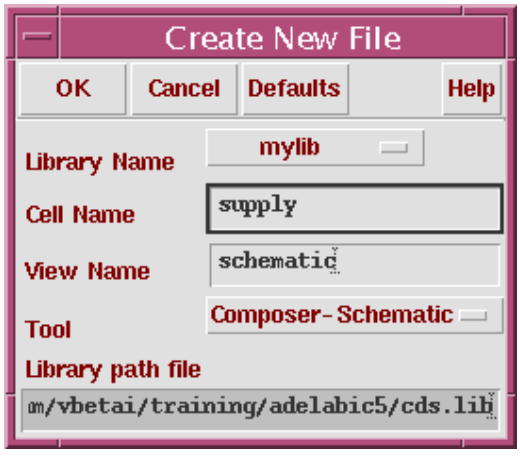
This method simplifies the process of adding power supplies to your circuit for simulation.

Creating the *supply* Cellview

You must create the *supply* cellview before adding components to the schematic.

1. In the CIW or Library Manager, execute **File—New—Cellview**.

2. Set up the Create New File form as follows:



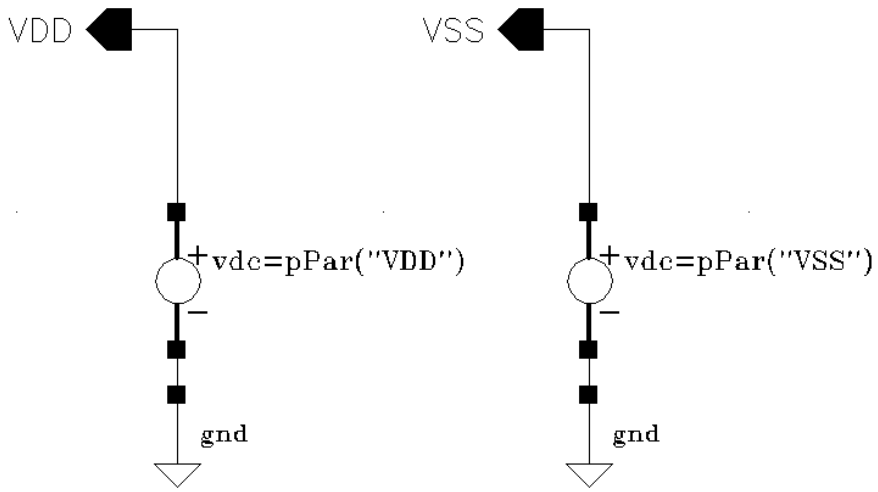
3. Click **OK** when done.

   A blank schematic window for the *supply* design appears.

Building the *supply* Circuit

Add components to build the *supply* schematic.

1. Using the component list in the table below, build this supply schematic.



Library Name	Cell Name	Properties/Comments
analogLib	vdc	For V0: DC voltage = pPar(“VDD”)
analogLib	vdc	For V2: DC voltage = pPar(“VSS”)
analogLib	gnd	

2. On the fixed menu in the schematic window, use the **Pin** icon to add the **output** pins *VDD* and *VSS*.

3. Click the **Wire (narrow)** icon and wire your schematic.

   Tip: You can also press the **w** key, or execute **Add—Wire(narrow)**.

4. **Check and Save** your *supply* design.

Creating a Symbol

Create a symbol of the *supply* circuit that you can place inside another schematic for simulation.

1. In the *supply* schematic window, execute **Design—Create Cellview—From Cellview**.

2. Click **OK** in the Cellview From Cellview form.

   The Symbol Generation Options form appears.

3. In the Pin Specifications field, leave all entries blank except for the following option:

Top Pins	`VDD VSS`

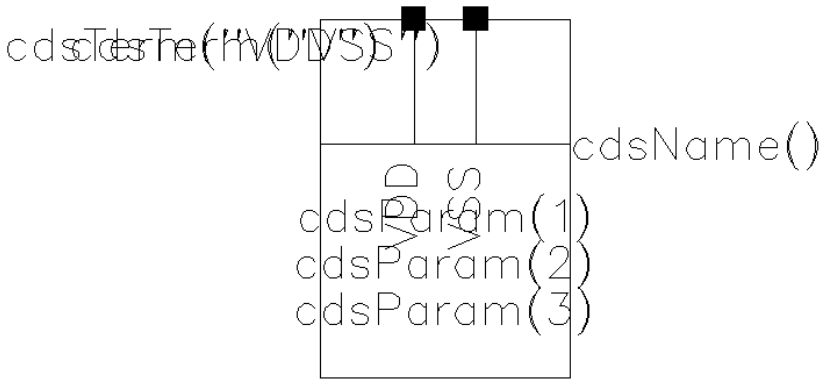
4. Click **OK** in the Symbol Generation Options form.

5. Observe the CIW output pane and note the messages stating that an analog CDF was generated.

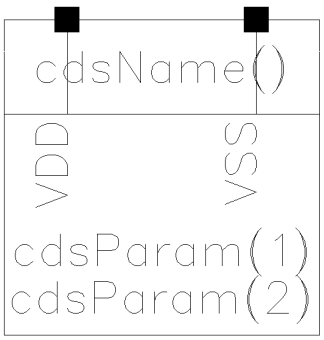
   During automatic symbol generation, hierarchical variables *pPar("VDD")* and *pPar("VSS")* are scanned and the system creates component parameters for the symbol based on these variables.

   This information is automatically recorded in the CDF for this cell.

   A window displays this supply symbol.



6. Modify the symbol to look like this one.



7. **Save** your symbol after completing your edits.

Notes on Symbol Updates

If you modify, add, or remove pins from a schematic, the existing symbol view will no longer match the schematic. You can use the Automatic Symbol Generator (ASG) to generate a new symbol.

In addition, you can use the ASG to update the CDF with additional or deleted parameters, based on modified hierarchical variables (*pPar()* expressions) in the schematic.

You can preserve the symbol that you created, by copying it to a temporary location. Run the ASG to regenerate the symbol, overwriting the existing one. Next, use the **Add—Import Symbol** command in the symbol editor to import the symbol copy (original symbol).

The reference material has details on the **Import Symbol** command. You might need to edit the CDF to remove unwanted component parameters or modify label display settings. You will learn about CDFs later.

Preparing for the Next Lab

1. In the symbol editor and *supply* schematic windows, execute **Window—Close**.

of Lab

Lab 2-4 Building the ampTest Design

Objective: To build a noninverting amplifier circuit using your Bi-CMOS operational amplifier.

Creating the *ampTest* Cellview

You will create the *ampTest* cellview that will contain an instance of the *amplifier* and *supply* cellviews. In the next section, you will run simulation on this design, with the amplifier set in a noninverting configuration with a gain of 3. You will measure output waveforms to verify the proper functionality of your amplifier.

1. In the CIW or Library Manager, execute **File—New—Cellview**.

2. Set up the Create New File form as follows:

3. Click **OK** when done.

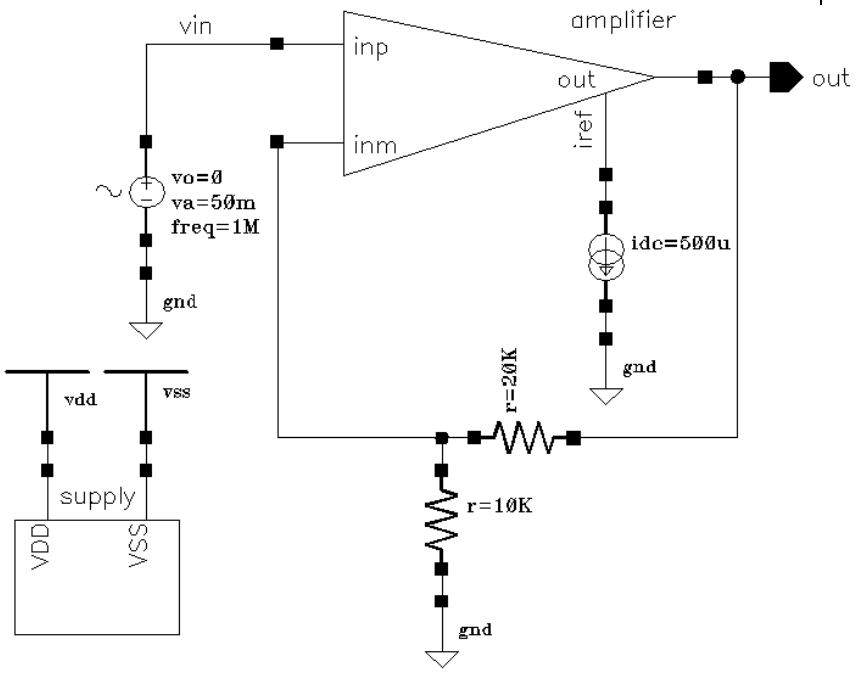
   A blank schematic window for the *ampTest* design appears.

Building the *ampTest* Circuit

In this part of the lab, you will build a **noninverting** amplifier with a **gain of 3**.

1. Using the component list and Properties/Comments in this table, build the *ampTest* schematic.

Library Name	Cell Name	Properties/Comments
mylib	amplifier	
mylib	supply	VDD = 5, VSS = -5
analogLib	vsin	For V2: AC Magnitude=1, Amplitude=50m, Frequency=1M, Offset voltage = 0
analogLib	idc	For I4: DC current = 500u
analogLib	res	For R1: Resistance = 20K
analogLib	res	For R0: Resistance = 10K
analogLib	vdd, vss	



Note: Remember to set the values for **VDD** and **VSS** in the *supply* block. Otherwise, your circuit will have no power.

2. In the schematic window, use the **Pin** icon to add the **output** pin *out*.

3. Click the **Wire (narrow)** icon and wire your schematic.

   Tip: You can also press the **w** key, or execute **Add—Wire(narrow)**.

4. In the schematic window, click the **Wire Name** function to access the form to label the node *vin*.

5. **Check and Save** your *ampTest* design.

Preparing for the Next Lab

1. Leave your *ampTest* schematic window open for the next section.

Analog Simulation

Module 3

February 10, 2005

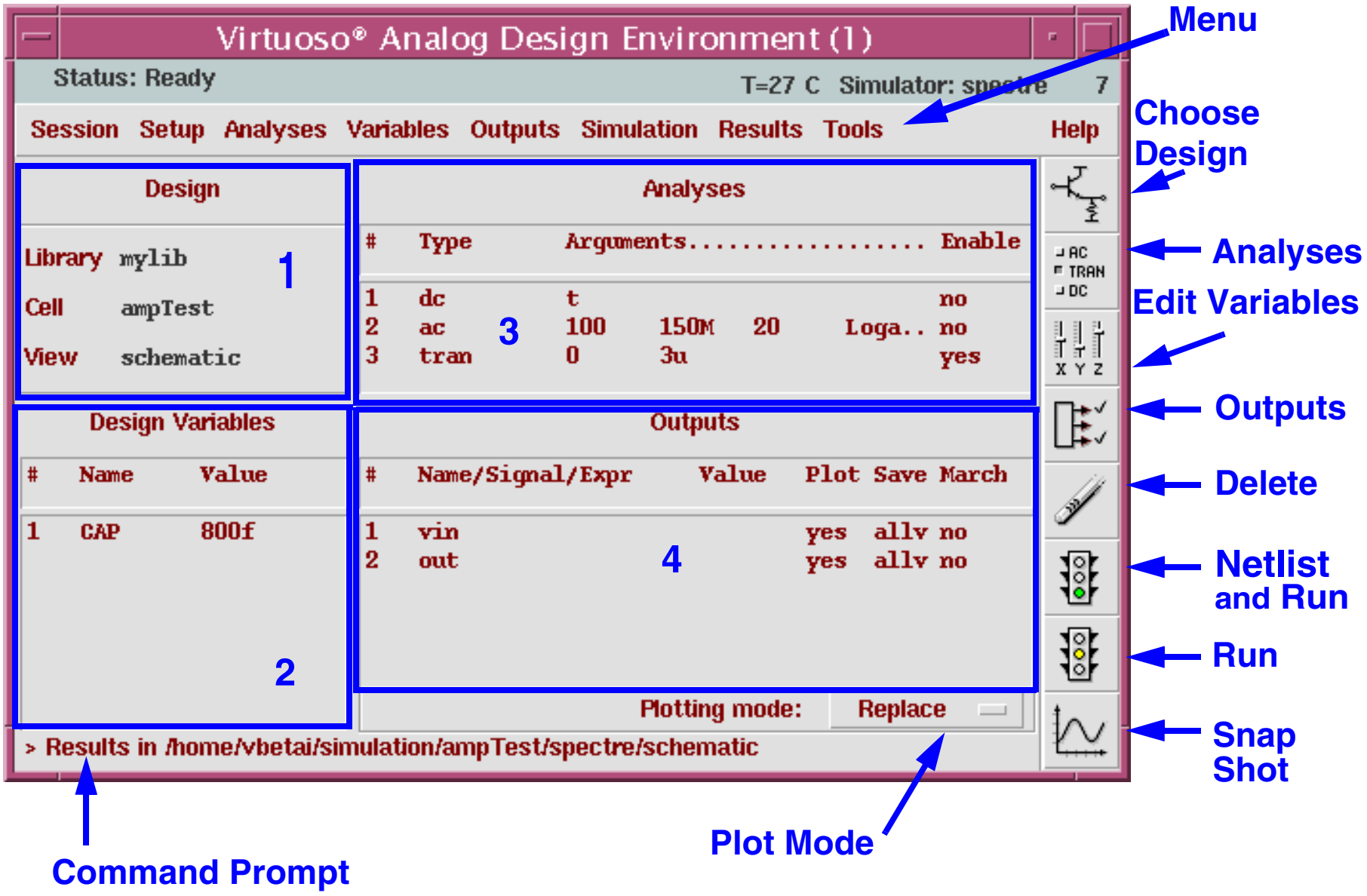
Topics in this Module

- Overview of the simulation environment
- Setting up the simulation environment
- Model files
- Design variables
- Choosing analyses
- Netlisting
- Running simulation
- Viewing simulation results with the Waveform display tool
- Saving simulator sessions

Terms and Definitions

simulation window	The ADE user interface to control and view simulations.
analyses field	A text field in the simulation widow indicating selected analyses.
output field	A text field in the simulation widow indicating selected output.
Spectre	A Cadence simulation tool for simulating analog circuits.
simulator host	Software tool such as Spectre, cdsSpice, etc. to be used for simulation.
model library	A text file having model description used by the simulator host.
stimulus template	A user interface used to establish signals used in simulation.
netlist	A textual description of a schematic used by the simulator host.
Waveform Window	A graphical interface used to plot simulation data.
direct plot	User command used for ‘special’ plots to the Waveform Window.
annotating	Process of displaying data back to another window or schematic.
label display	A label attached to a component for displaying information.
snapshot	A command to the Waveform Window to update intermediate results.

Important Features of the Simulation Window



The diagram shows the Virtuoso Analog Design Environment window. The simulation window is annotated with notes used to describe the important features.

The are four regions identified by numbered boxes 1, 2, 3, and 4.

- Region 1 is the “Design Field”. It indicates the library, cell, and cell view to be simulated.
- Region 2 is the “Design Variables Field”. It shows the design variables and set values.
- Region 3 is the “Analyses Field”. It shows the status, ranges, and types of analyses.
- Region 4 is the “Outputs Field”. It indicates the selected outputs and expressions.

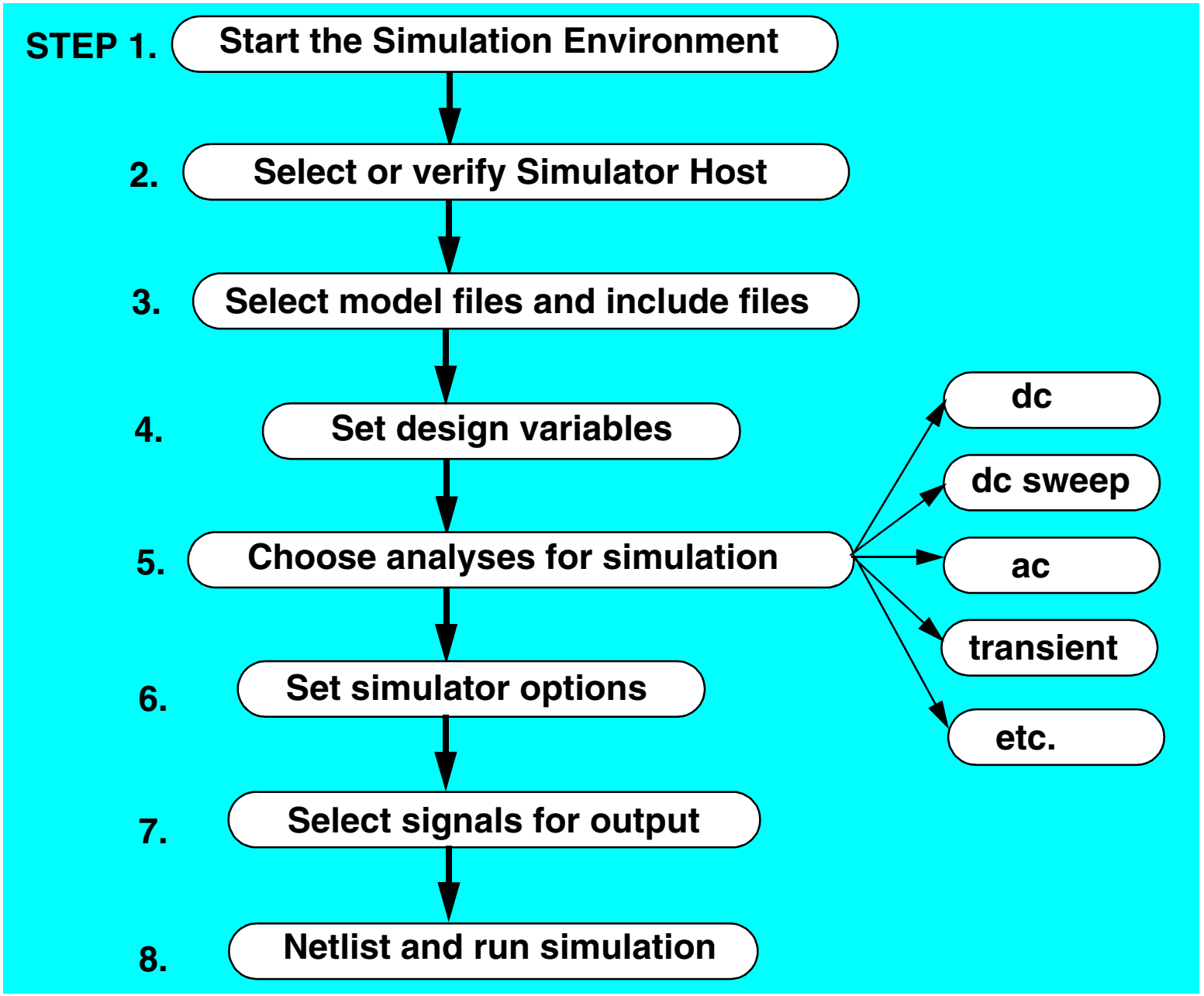
Important

The entries within Regions 2, 3, and 4 are accessible by executing a left mouse click. It is sometimes easier and faster to make changes within these fields than using the menu banner.

The simulation window also has a **Menu Banner** used to select analyses, models, outputs, and other simulation controls. The Menu Banner also has a Tools menu used to set up **advanced** simulation tools such as Corners, Monte Carlo, and Optimization.

The simulation window also has a set of icons for along the right-hand side for quick selection of specific commands.

Analog Simulation Flow



This diagram shows the major steps required to set up and run an analog circuit simulation in the simulation environment. To set up the simulation the first time, most of these steps are required. However, on all subsequent simulations, perhaps only one or two steps are needed.

Each of the steps numbered above shall be discussed within this module.

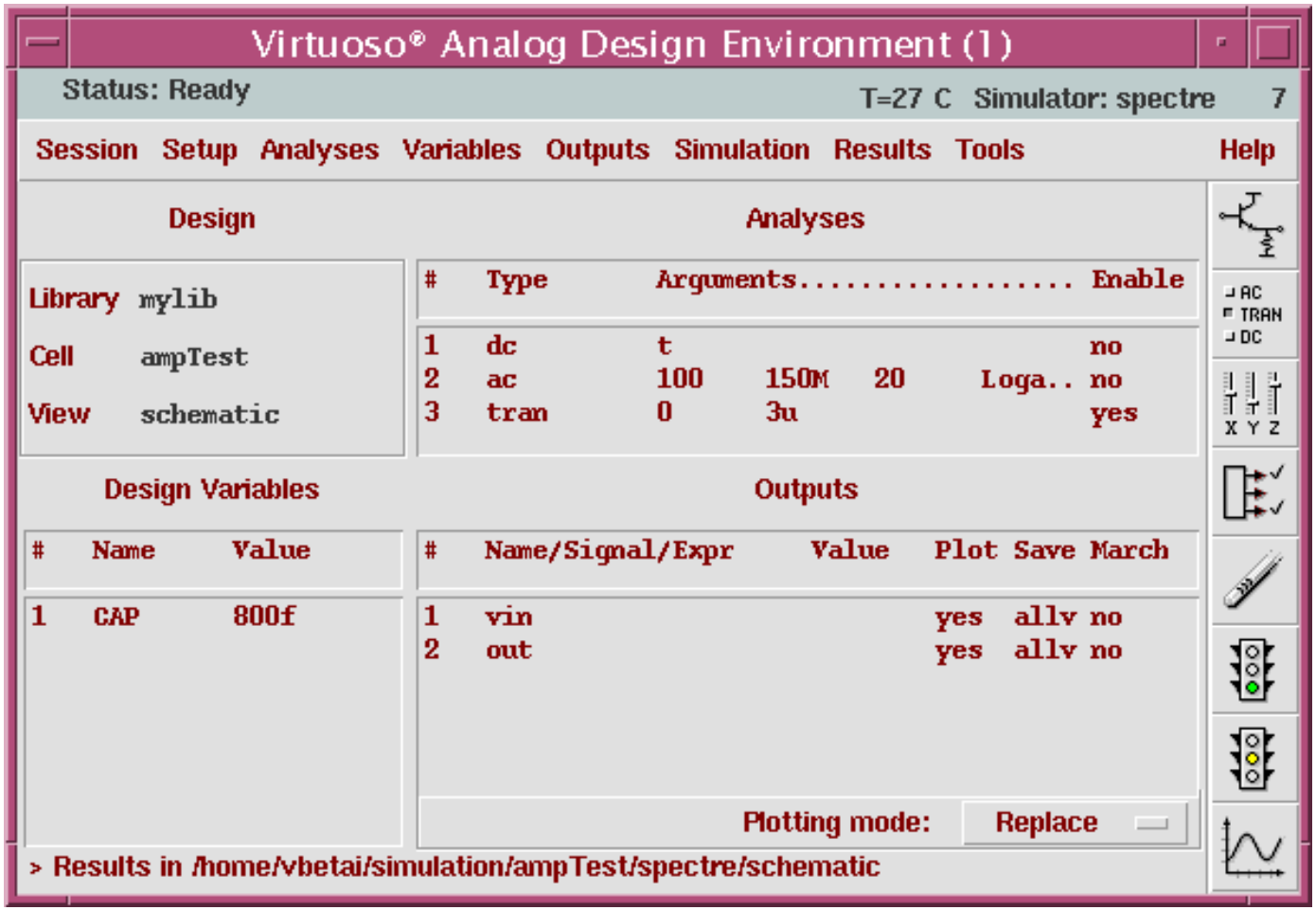
Important

If you are unfamiliar with the flow, this may appear to be a large amount of work to set up the simulation. However, after the initial setup is completed, the flow is greatly simplified. Once you are familiar with the flow, all of these steps are performed within a few minutes, or less.

Starting the Simulation Environment

STEP 1. Select **Tools—Analog Environment** from the schematic menu banner, or select **Tools—Analog Environment—Simulation** from the CIW.

Then the Virtuoso Analog Design Environment "**Simulation Window**" appears.



STEP 1.

In this class, use the Spectre simulator that is integrated directly into the Virtuoso Analog Design Environment.

Start the simulation environment from the schematic window or the CIW. If the simulation environment is started from the design window, the design in the window is understood to be the target of the analysis. Start the simulation environment from the CIW to simulate any design without viewing it.

Commands on pull-down menus in the Simulation window establish a simulation flow if used in order, starting at the top and left and ending with the last command on the right most pull-down menu.

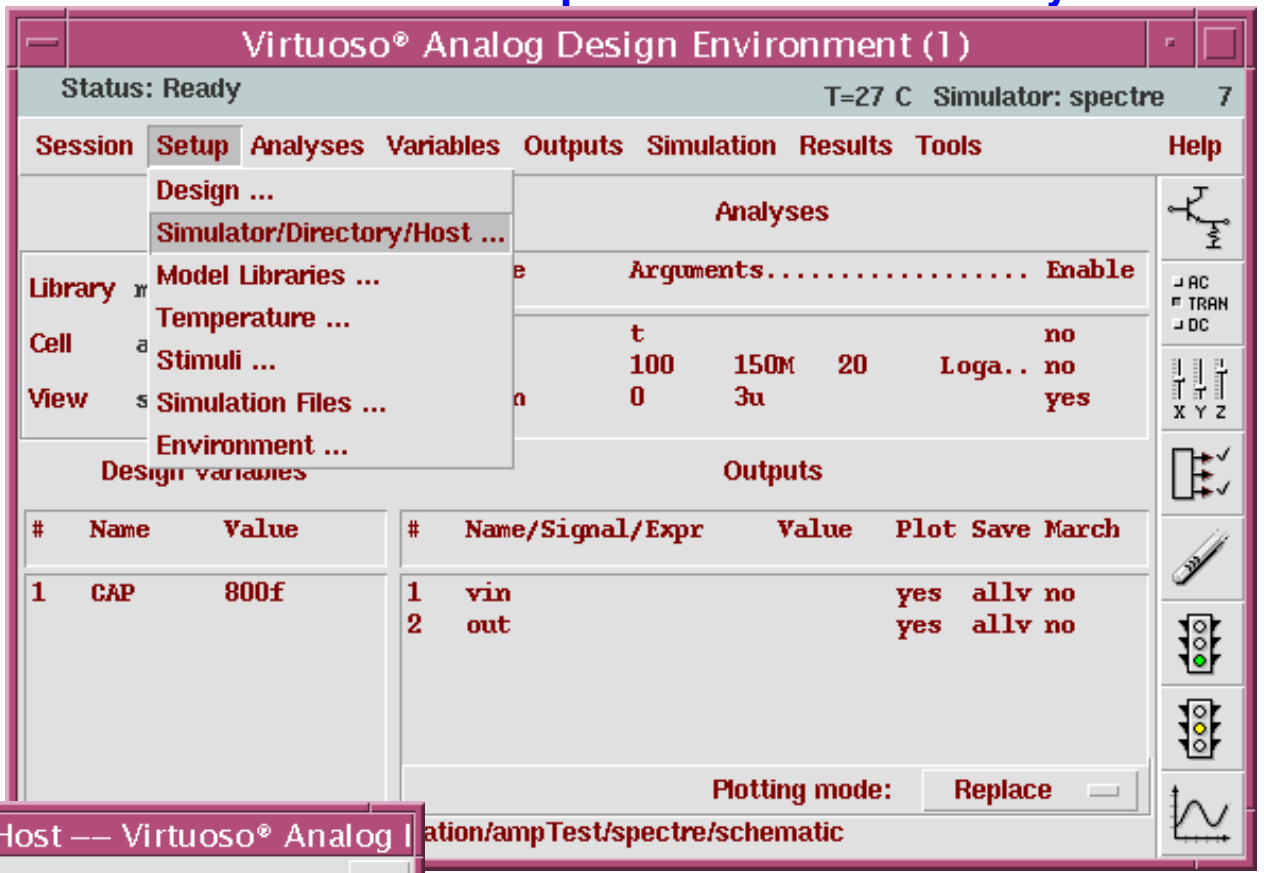
For analog analysis, include this statement in your *.cshrc* file before starting the Cadence Design Framework II environment:

```
setenv CDS_Netlisting_Mode "Analog"
```

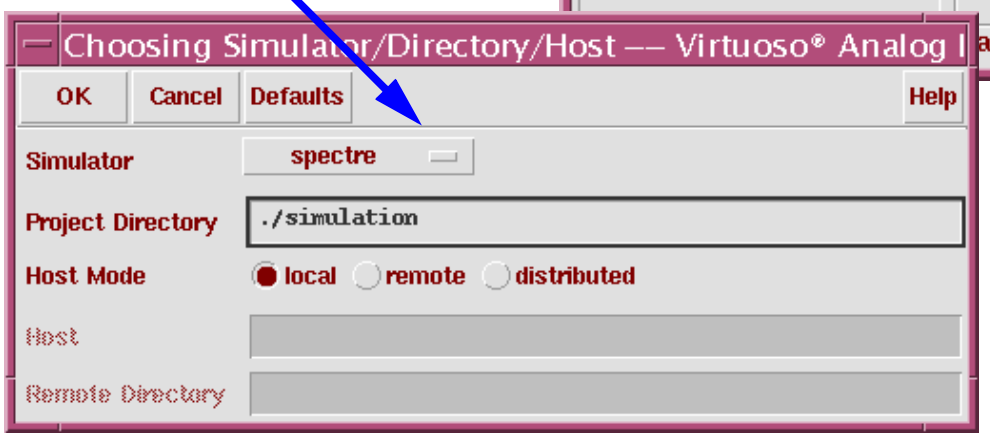
Setting the Simulator

STEP 2.
Select the simulator to be used.

Select: Setup—Simulator/Directory/Host



For this class select **spectre** in the cyclic field.



STEP 2.

The second step of the Analog Simulation Flow is to set the target simulator and Project Directory. The Project Directory is the location where the simulator puts all of the data, including netlists, waveforms, include files, and mapping information.

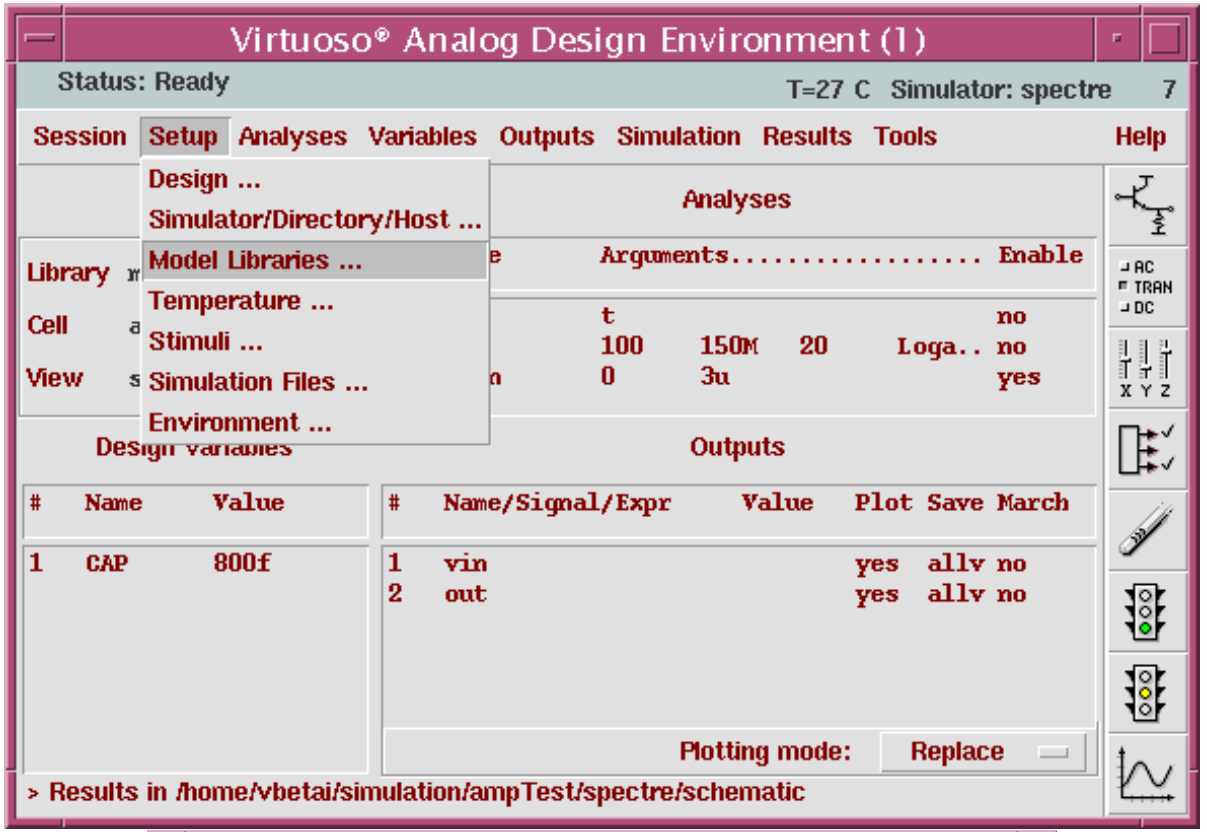
Set the simulator to **spectre** to start direct simulation with the Spectre tool. Note that a cyclic field appears that allows other host simulator choices, such as *spectreVerilog*.

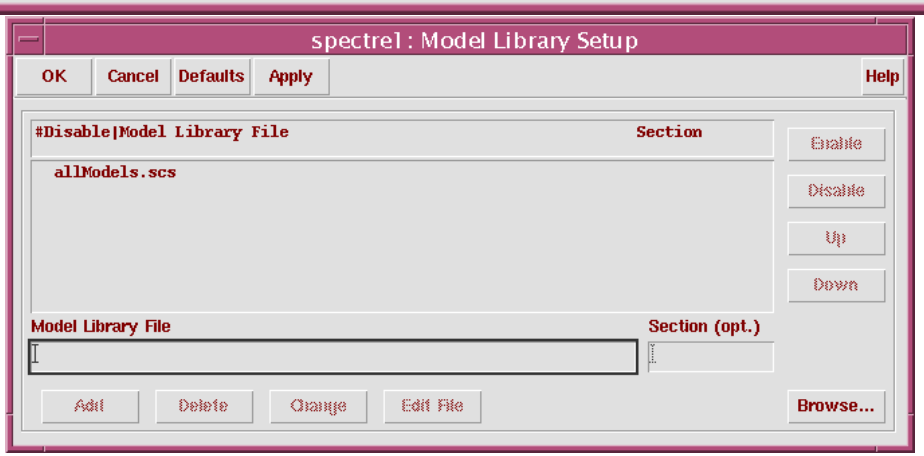
The Choosing Simulator/Directory/Host form has a button that provides a way to choose the **distributed** Host Mode. When this button is pressed, there are two additional fields added to your form: **Auto Job Submit**, and **E-Mail Notify**. This function uses multiple machines to handle large or computationally intensive jobs. Jobs are sent to the default queue named in your *.cdsenv* file or to the last setting in the Job Submit form. When **Auto Job Submit** is not selected, the Job Submit form appears whenever a simulation is run. More information can be found in CDSDoc.

Setting the Model Libraries

STEP 3.
Select the model files. In simulation window, select:

Setup—Model Libraries...





This example uses a relative path. The path is set by the Include Path in the Simulation Files Setup form.

STEP 3.

This is a critical step that must be performed to ensure that model files for active devices, such as transistors, are included in the netlist.

With Spectre Direct:

- *.scs* = Spectre circuit simulator. All files ending in *.scs* are implied to be Spectre syntax in the netlist unless **simulator lang=spice** is included in the file. (SPICE in this case means general Berkeley SPICE syntax and NOT cdsSpice). Spectre example:

```
model npn bjt type=npn is=3.26E-16 va=60 bf=100 \
br=6 nc=2 ikr=100m rc=1 vje=0.7 \
cjc=1e-12 fc=0.5 cje=0.7e-12 \
tr=200e-12 tf=25e-12 itf=0.03 vtf=7 xtf=2
```

- There are several modeling techniques to use. The main use model is one file with all of the models contained within.
- The contents of the model file do not appear in the netlist. The library model file is referenced as an include file.

There are samples of all of the models available at:

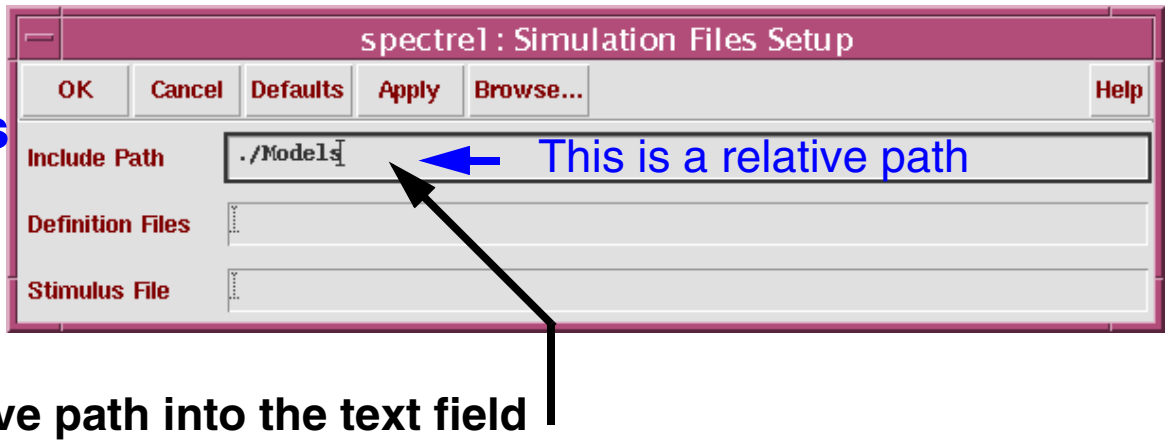
`<install_dir>/tools/dfII/samples/artist/models/spectre`

Simulation Files

STEP 3.

Setup—Simulation Files

For setting the path to other simulation files



Enter the absolute or relative path into the text field

Include Path	Path to location of any Model Library Files, Definition Files, and Stimulus Files (the paths in these fields can be relative paths)
Definitions Files	File or files that contain function definitions and parameter declarations not in the Design Variables section of the Simulation Window
Stimulus Files	Location of text-based stimulus file

- Each of the above fields can have one or more paths or file declarations separated by a space.
- Any type of Spectre include file can be included in the netlist by either using the Definitions Files field above or the Model Libraries Setup form.

STEP 3.

Include Path—Example of absolute path declaration:
"/usr/home/models/project/models"

The simulator resolves a relative file name by first looking in the directory where the file is located. Subsequently, it searches for the file in each of the directories specified by the include path, from left to right.

Definitions File—Defines functions and global variables that are not design variables. It is included in the netlist before model files. Example:

```
simulator lang=spectre
real PiRho() {
    return 2500;
} Functions returning constant values
real Rpb(real l, real w) {
    return PiRho()*l/w;
} Simple passing parameters
real rpoly(real value, real tdc) {
value*(1+.01*(tdc-25)+.002*(tdc-25)**2);
} poly resistor function of temperature
```

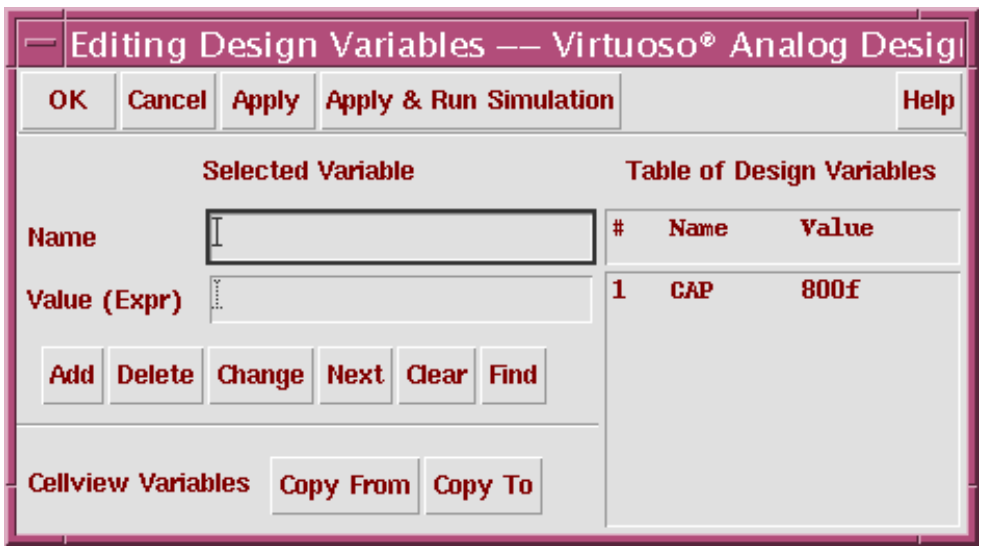
A sample Definitions File exists at:
`<install_dir>/tools/dfII/samples/artist/models/spectre/defaults.scs`

Stimulus File—Enter the full path to the directory where the stimulus file resides. A file name ending in *.*scs* defaults to Spectre. All other files default to SPICE syntax.

Setting Design Variables

STEP 4.

- Extract variables in a design with the **Copy From** button.
- Copy variable settings back to the design with the **Copy To** button.
- Add variables used in parameterized model files that are not extracted.
- Update a variable value and run simulation. Netlisting does not occur again.
- Extract new variables added after a simulation run.
- Use the **Find** button to locate the Selected Variable in your design.



Select **Variables—Edit** or click the **Edit Variables** icon.

STEP 4.

Use the Editing Design Variables form to:

- Enter design variable values. These values can be numbers or expressions.
- Copy variables from a design, or copy set variables back to the design to be saved.
- Add new variables. Add variable names to this form that control the simulation engine or appear in model files and then set their values.
- Find a variable in the design. This useful feature highlights the component to which the selected variable is attached. This can be very helpful in a large design when trying to locate a particular design variable.

When starting the simulator from this form, change a variable value, apply the change, and repeat simulations quickly. The system will not create a new netlist when a variable value is updated.

Spectre Direct has no character limit for the size of a variable. The following examples are legal statements:

```
parameters thisIsAReallyLongDesignVariableName=10000 desVar2=10p \
desVar3=2u desVar4=1.15
```

Design variables are not evaluated. They appear in the netlist as Spectre parameter statements.

Choosing Analyses

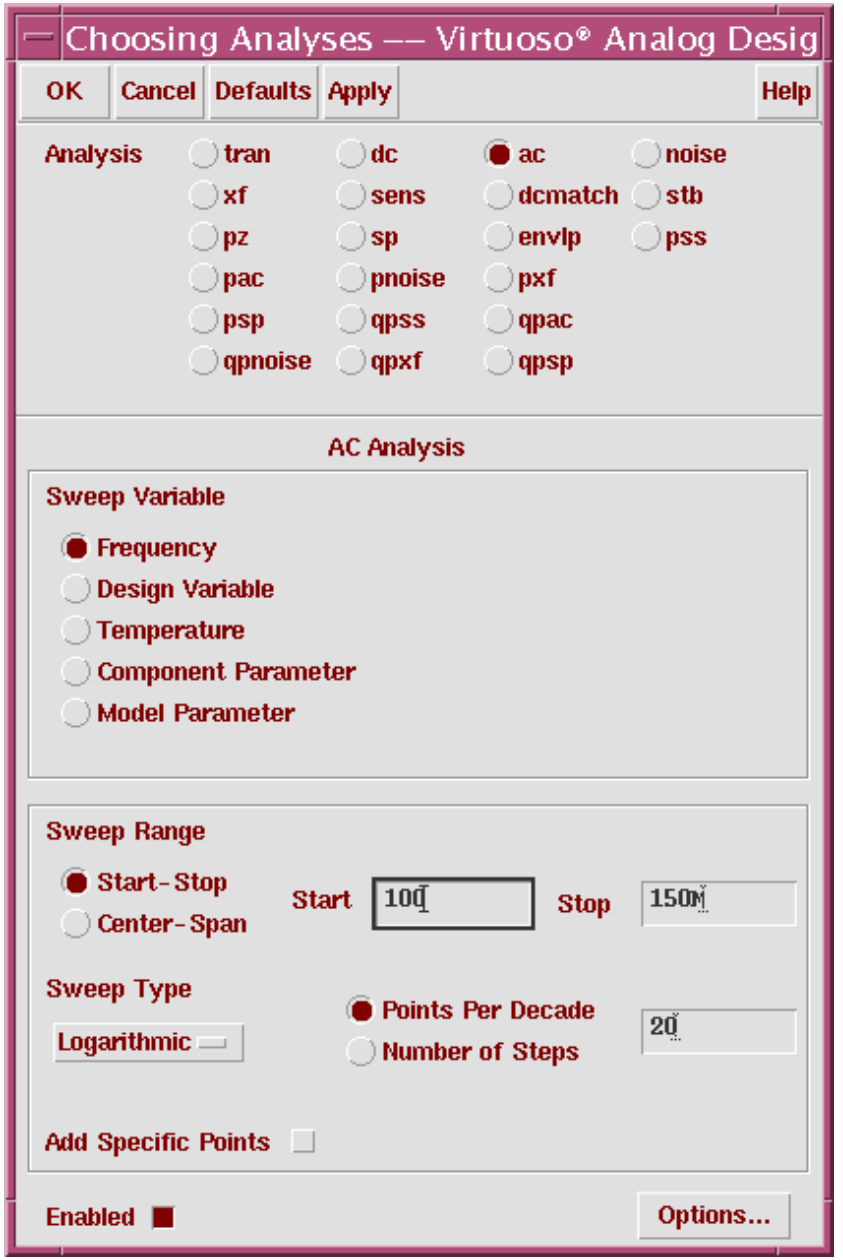
STEP 5.

Select

Analyses—Choose

or click the

Choose Analyses icon.

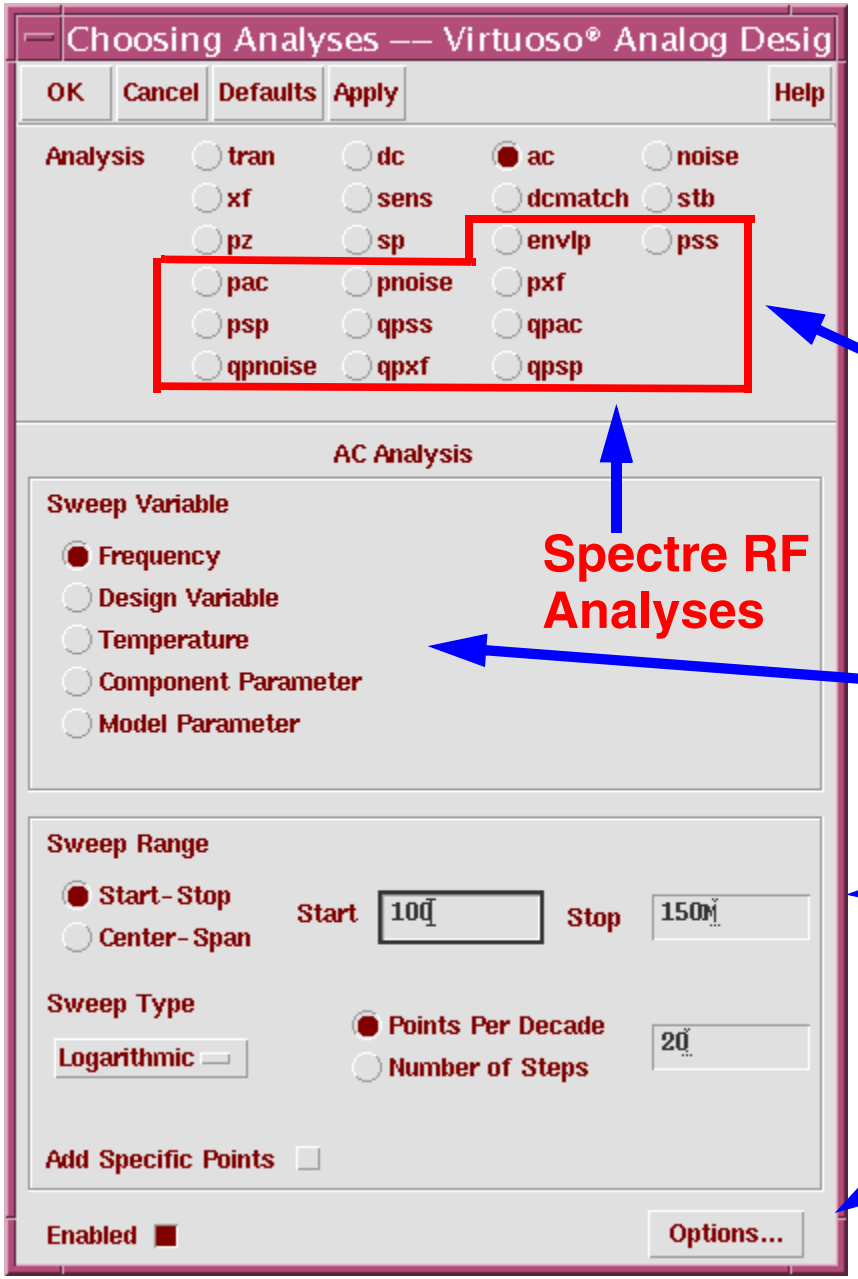


STEP 5.

Choosing the analyses to perform on the design is a straightforward process. Run analyses together, separately, and in any combination. Analysis units are determined by the simulation engine.

Output data is generated for each specified analysis during a simulation.

Choosing Analyses Details



The Choosing Analysis form dynamically changes based on the host simulator and the selections made on the form.

The types of analyses that are available depend on the host simulator.

Sweep Variables depend on the selected analysis.

Sweep Range

Analysis Options are entered by selecting the Options... button.

The Choosing Analysis form depends on the host simulator. The form above is used for the Spectre circuit simulator, which supports 15 types of analyses. These include **ac, dc, tran, stb, dcmatch**. In addition:

- sp S-Parameter Analysis

Also, the following types of RF analyses are supported by Spectre RF tool:

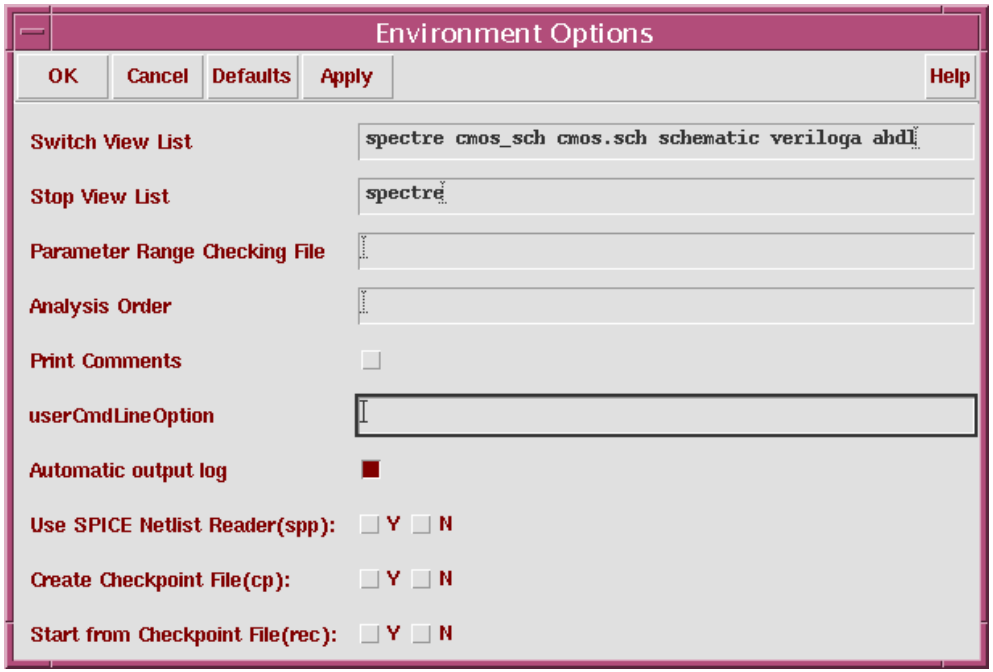
- envlp Envelope Following Analysis
- pss Periodic Steady State
- pac Periodic AC
- pnoise Phase Noise Analysis
- pxf Periodic XF Analysis
- psp Periodic S-Parameter
- qpss Quasi-Periodic Steady State Analysis
- qpac Quasi-Periodic AC
- qpnoise Quasi-Periodic Noise
- qpxf Quasi-Periodic XF
- qpsp Quasi-Periodic S-Parameter

For more information on Spectre RF analysis, see CDSDoc. In addition, Cadence Education Services provides training in Spectre RF usage.

Simulation Environment Options

STEP 6.

Select **Setup—Environment**



- Switch View List and Stop View List establish netlisting rules
- Parameter Range Checking File
- Use the SPICE Netlist Reader (spp)—Read in HSPICE/SPICE netlists and run with the Spectre simulator.
- Checkpoint and Restart

STEP 6.

The Switch View List specifies the order that the cellviews are netlisted.

The Stop View List specifies the view at which the netlist statements are generated.

For the Spectre simulator, the Parameter Range Checking File contains the parameter range limits. Use either the full path to the file, or a period (.) to specify a relative path to the directory where the Cadence tools were started.

(Optional) Check and set the options for view switching to control how the system netlists hierarchical designs. Normally, there is no need to modify the Switch View List and Stop View list.

The SPICE reader (*spp*) option is provided for include files or subcircuits that are in SPICE syntax. The *spp* automatically converts them to Spectre syntax, so a Spectre simulation can be run.

Starting in 5.1.41, SPP is no longer needed to read SPICE netlists or models into the spectre simulator. A new parser (csfe) has been added with better performance and compatibility with other SPICE simulators. csfe has not been made the default parser, but may be in future releases. Insert following line in *.cdsenv* file to invoke the new parser:

```
spectre.envOpts useCsfe boolean t
```

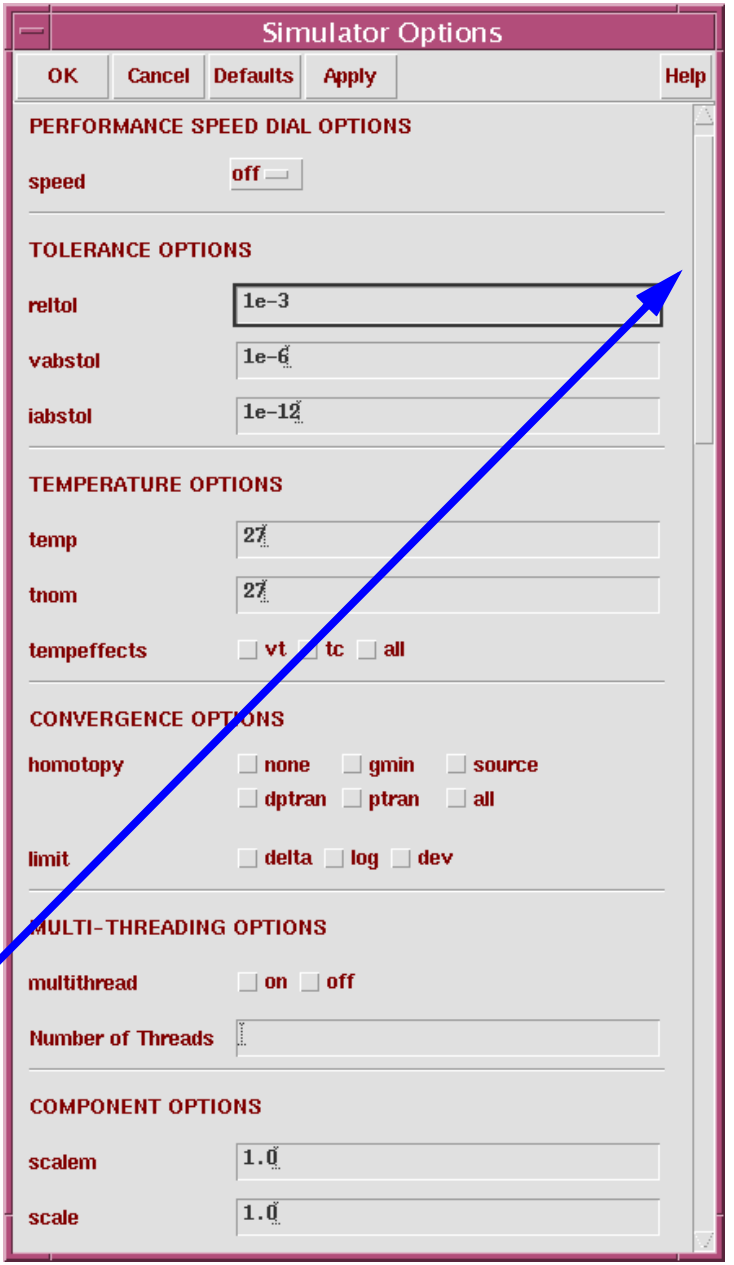
The **Create Checkpoint File (cp)** and **Start from Checkpoint File (rec)** options allow Spectre to save checkpoint files while a simulation is running, and then restart the analysis from this file. Currently, only transient analysis supports checkpoint and restart.

Simulator Options

STEP 6.

Select **Simulation—Options—Analog**

Use this form to set the simulator tolerance values, convergence options, and other settings.



NOTE:
This is a very long form. The scroll bar indicates the amount of the form that is visible.

STEP 6.

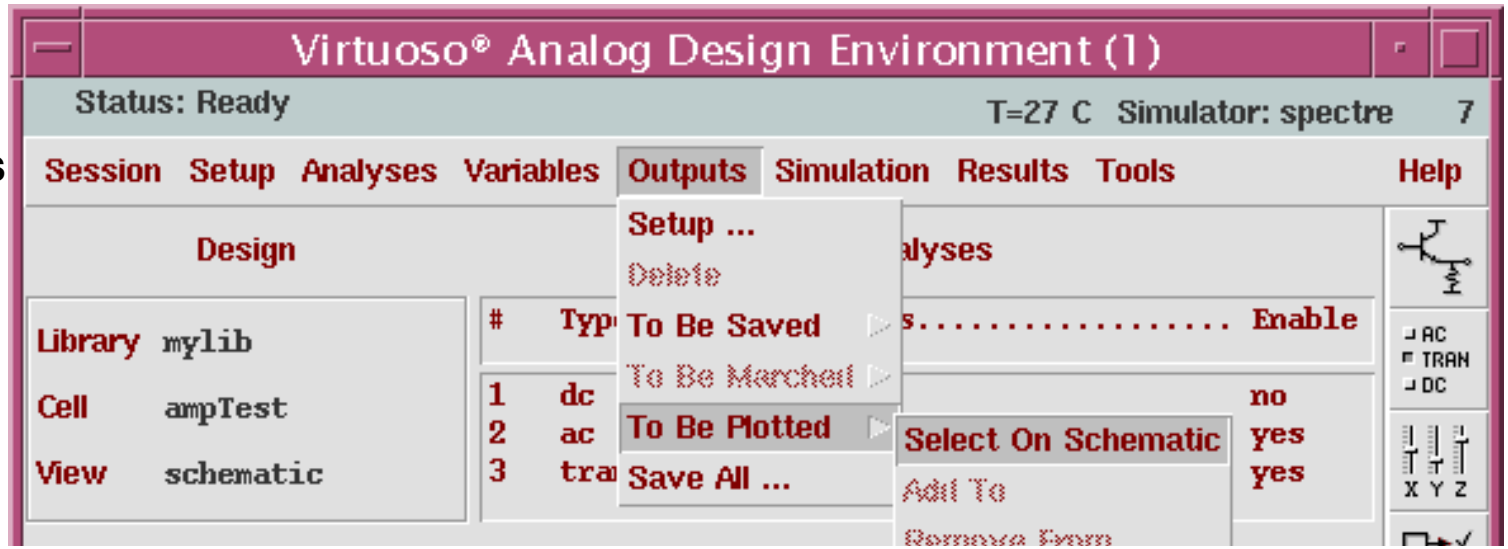
Select **Simulation—Options—Analog** to activate the Simulator Options form. Use this form to set convergence, tolerances, and other simulator settings.

In addition to this form, the Choosing Analyses form has an **Options** button for setting specific options for transient, ac, and dc analyses. For example, the **Infotimes** and **Captab** options are found by selecting the **tran** button in the Choosing Analysis form and then selecting the **Options** button.

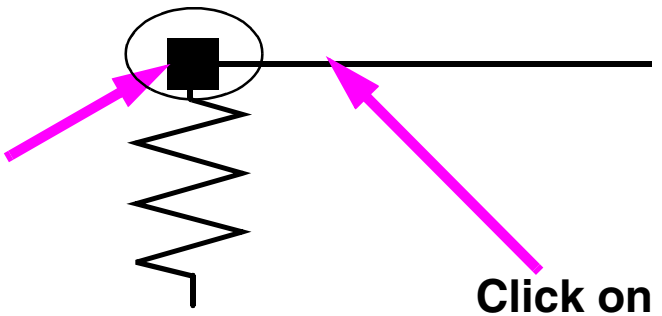
Probing the Schematic to Save Output Data

STEP 7.

Select signals output



Click on pins to save currents.



Click on wires to save voltages and frequency data.

Select: Outputs—To Be Plotted—Select On Schematic

- You **must** terminate this command by pressing the **Esc** key.
- Save and load data sets.
  Optionally, save quantities associated with all wires, all pins, or both.

STEP 7.

Specifying Outputs

Use **Outputs—To Be Plotted—Select On Schematic**

When using the command to select a set of outputs, click on device pins to save terminal currents or wires to save node voltage or frequency data. Data will be available for plotting and analysis on nodes or terminal pins that are saved before simulation.

Select the **Edit—Select—Filter** command in the schematic window to display the *Schematic Selection Filter* to choose which objects, such as wires and pins, are selectable.

Saving All Outputs

Save all quantities associated with wires or pins in a design.

Reminder to Terminate Select “Outputs...”

When doing the lab activities, it is extremely important to remember to terminate the command **Outputs—To be Plotted—Select On Schematic.**

This command allows you to select wires and terminals for plotting. The command continues to select wires and terminals until it is terminated by pressing the **Esc** key.

A common error when using the simulation environment is to continue work without terminating this command. The user will move the mouse to select Netlist and Run, or other commands in the menu banner. Then the user attempts to change a component parameter on a symbol in the schematic. The simulation environment responds by selecting all terminals of the component to be plotted and these appear in the “Outputs” field of the simulation window.

Now the user must unselect the terminals and delete the entries in “Outputs” field.

When using the Outputs—To Be Plotted—Select On Schematic command, it is important to terminate this command. It is terminated using the Esc key (also known as the escape key).

Important

If you forget to terminate (a common mistake) and later attempt to select a component on the schematic for edit, all terminals of the component will be selected for plotting and appear in the “Outputs” field the simulation window. At this point it is easy to recover. Select the component again. Fortunately, the Outputs—To Be Plotted—Select On Schematic command is a toggle command. The second mouse click is an unselect. The terminals of the component are unselected and also removed from the “Outputs” field. Now press the Esc key.

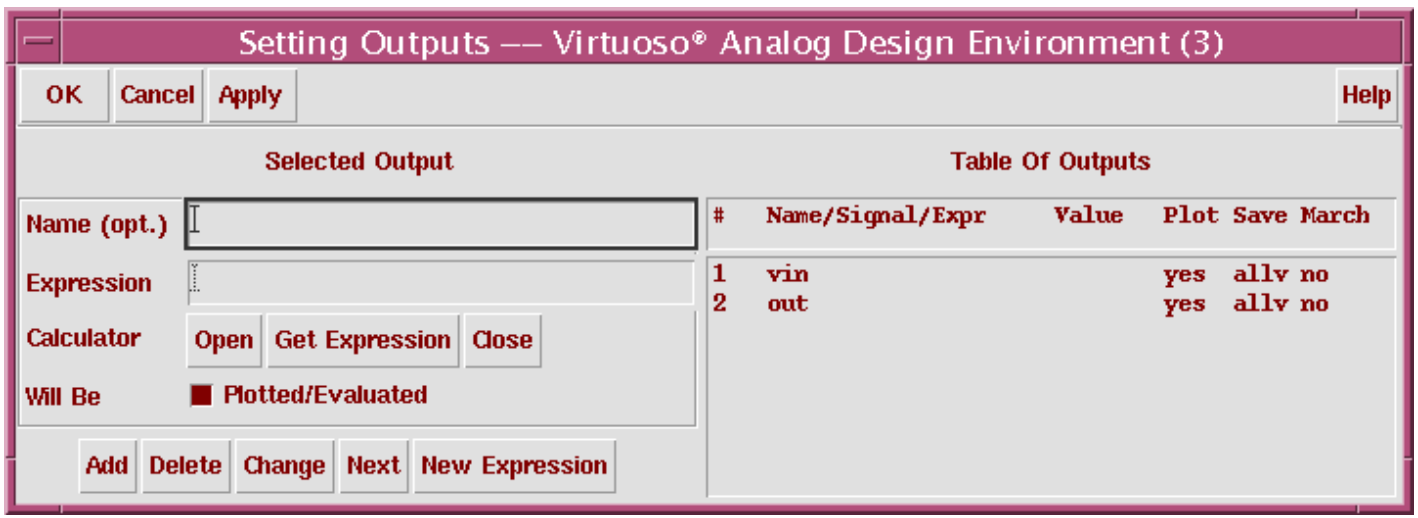
Outputs Section of Simulation Window

STEP 7.

#	Name/Signal/Expr	Value	Plot	Save	March
1	out	wave	yes	allv	yes
2	input	wave	yes	allv	no
3	VDC("/out")	-1.004m	yes		
4	phaseMargin	73.64	yes		
5	gainMargin	-16.00	yes		

Automatically Evaluated

Automatically Plotted



STEP 7.

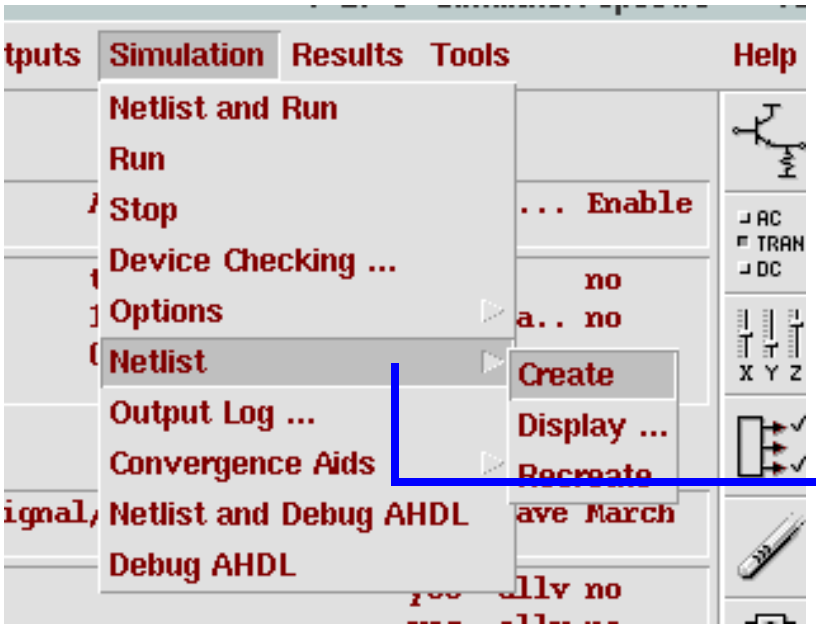
In the Outputs section of your Simulation window display the following:

- Signal names and evaluated expression values (created by the calculator)
- The signals to plot
- The signals to save
- The signals to march (this is an option in **spectreS** only.)

Use calculator expressions to return either a waveform or numeric value.

If signals and expressions are saved prior to running the simulation, the Virtuoso Analog Design Environment will automatically plot the waveforms and evaluate the numeric expressions and display their values when the simulation ends.

Netlisting



STEP 8.

- Netlists are hierarchical and created incrementally. Re-netlist only the modified schematics.
- Force all schematics to re-netlist with **Simulation—Netlist—Recreate**

STEP 8.

The system automatically creates netlists when running a simulation, but a netlist can be created and viewed before simulation.

Netlisting

Use the **Simulation—Netlist—Create** command to:

- Use the Virtuoso Analog Design Environment to create a netlist that is simulated in standalone mode.
- Modify the netlist, perhaps to take advantage of features that the interface to your simulator does not support.
- Read the netlist before starting the simulation.

Incremental Netlisting

Incremental netlisting is faster than full hierarchical netlisting because only the schematics that have changed since the previous netlist was generated are re-netlisted. This substantially speeds up netlisting of hierarchical designs containing many small schematics. The system keeps track of the status of each schematic during and between design sessions.

Running the Simulation

STEP 8.

After the netlist has been created or recreated, the simulation is ready to run.

To run the simulation:

- Select **Simulation—Run** or
- Select the **Run** icon on the right side of the simulation window.

If preselected outputs appear in the output field of the simulation window, then the Waveform display will automatically appear when the simulation is completed.

Running Additional Simulations

The purpose of running a simulation is to verify the operation and performance of the circuit. This often requires running additional simulations. Most steps of the simulation flow have now been completed. So running additional simulations is greatly simplified.

- To make changes to the simulation, simply modify the entries to the Design Variables, Analyses, or Output fields, and then press the **Run** icon.
- To change the temperature, select **Setup—Temperature** in the menu banner, enter the new value, and the press the **Run** icon.
- If you did not edit the schematic, you do not need to netlist the circuit.
- If you did edit the schematic, you must do a Check and Save in the schematic window.
- After the Check and Save, you must select **Netlist—Recreate** within the simulation window.
- The simulation setup can also be saved for running additional simulations at a later time, or even for running simulations on similar circuits.

Upon completion of the first simulation, the setup has been complete. It is now very simple to change the parameters of the next simulation, or set of simulations.

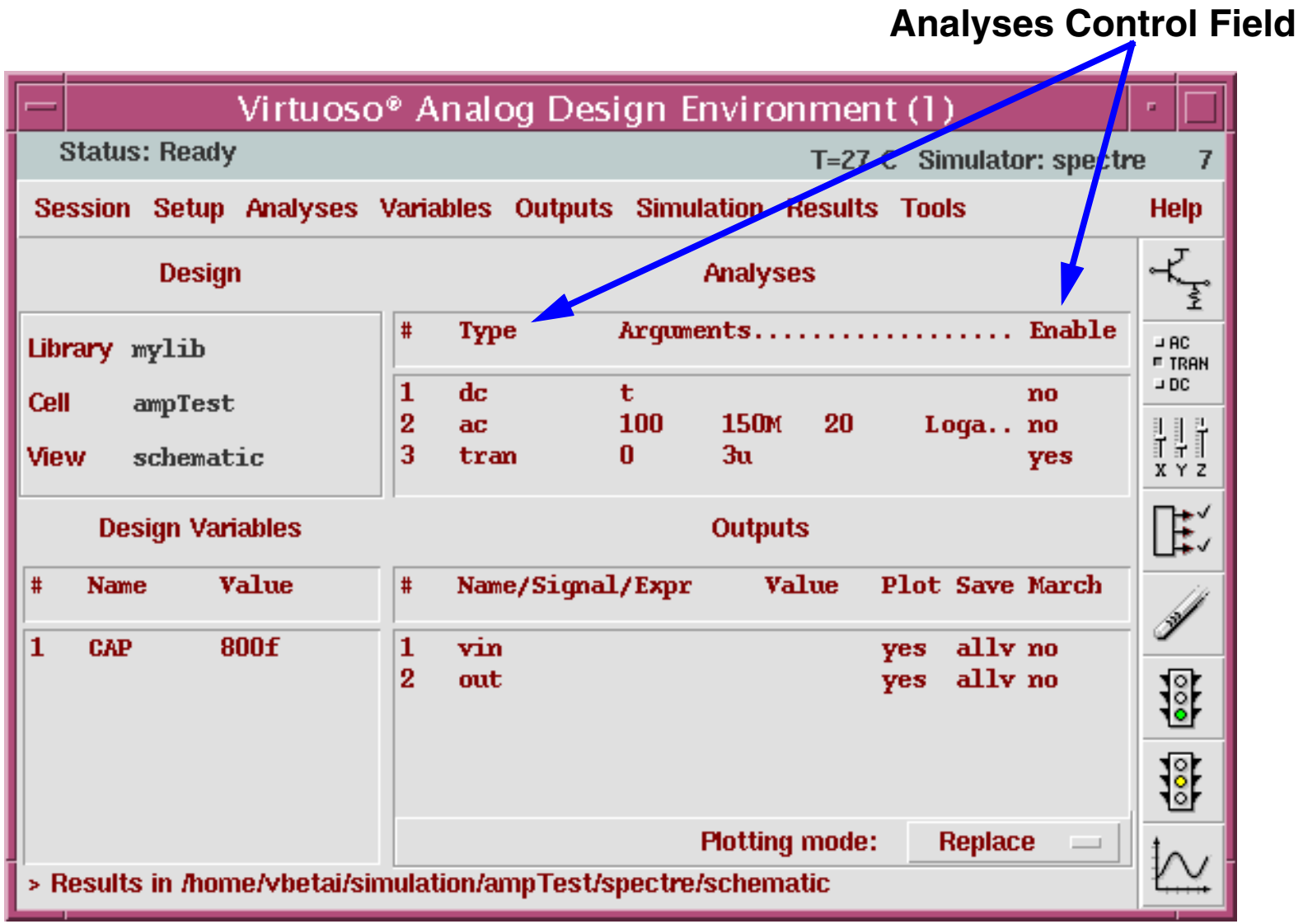
Just select by:

- a mouse sequence using the menu banner,
- selecting an icon on the right-hand side of the simulation window, or
- selecting the appropriate line within an editable field of the simulation window, such as the “Analyses” field.

If you make changes only to the simulation window, then you are able to run the new simulation without netlisting.

If you make changes to the schematic, then the schematic must be checked and saved, and a new netlist must be generated. However, the simulation window accesses the design database using the Design Framework. There is no requirement to open or close additional windows. A few simple clicks and the updated schematic is simulated.

Control of Analyses for Simulation



The Virtuoso Analog Design Environment provides additional control for running analyses. The Analyses Control Field lets you select which analyses to complete during the next simulation run. This field displays all analyses that have been activated by using the Choosing Analyses window.

To select a specific analysis, move the mouse into the Analyses Control Field and click **left** to select the specific analysis line. The selected line is highlighted.

The menu banner of the Simulation Window provides these options:

- Select **Analyses—Delete** to remove the analysis completely!
- Select **Analyses—Enable** to include the analysis in the next simulation.
- Select **Analyses—Disable** to deactivate analysis, but not delete from Analyses field.

The disable mode means the selected analysis does not run for net simulation. However, the **setup** from **Analyses—Choose** remains. Simply select, and then enable the analysis to run.

Additional Options Using ADE

The Virtuoso Analog Design Environment provides additional features that simplify running additional simulations, or modify the performance of the simulation.

These options are:

- Analog Default Options
- Save State
- Load State
- Stimulus Template
- Simulation Environment Options
- Infotimes
- Captab

Analog Default Options modifies the default appearance of the simulation window.

Save State is used to save the setup of the simulation window for reuse.

Load State is used the restore the setup of the simulation window.

Stimulus Template is an alternative method to provide stimulus to the circuit.

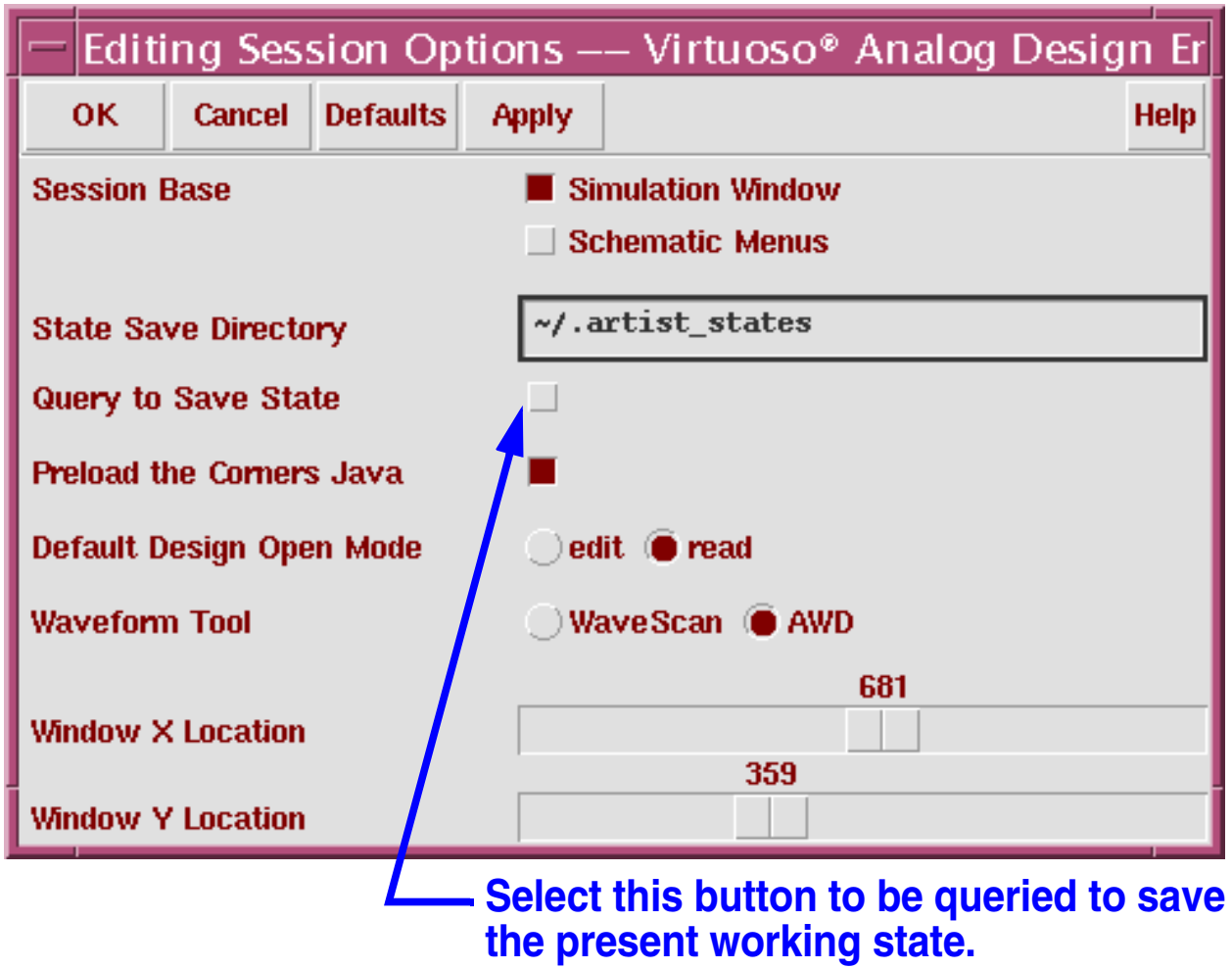
Simulation Environment Options alters the switch view list and the stop view list.

Save Options sets the default levels of signals to be saved.

Analog Default Options

In the Simulation window, select **Session—Options**.

Then an Editing Session Options window appears.



With this command, you set up certain options to affect the way the Simulation Environment looks. Choose which method you prefer: Simulation Window-based (default) or Virtuoso Schematic Editor-based (using schematic window menus) or both.

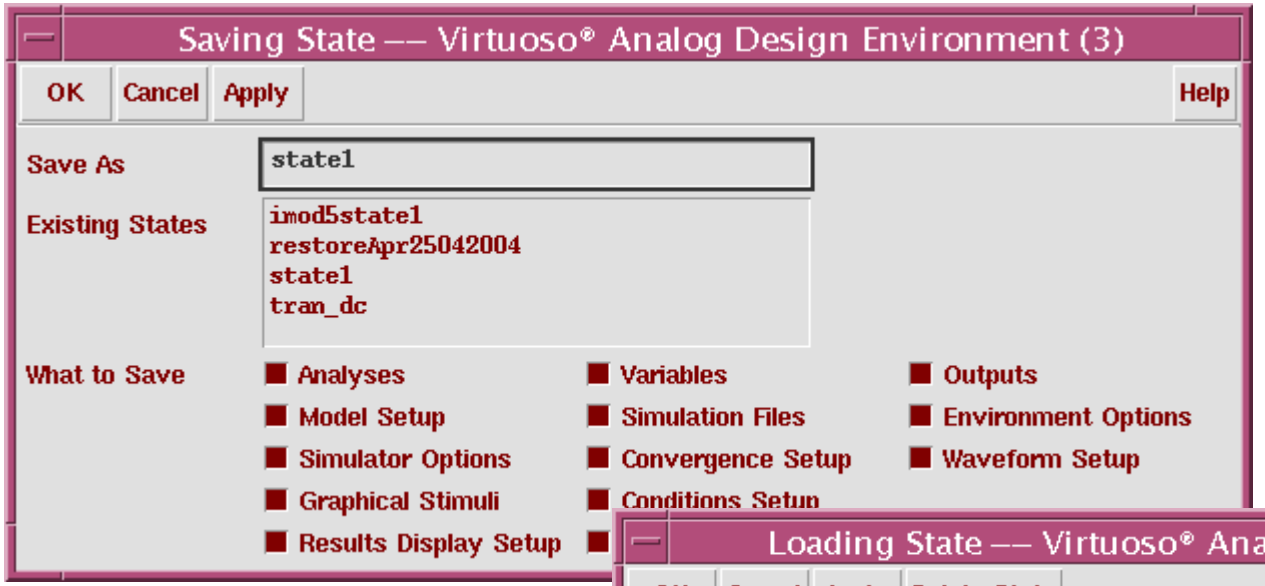
In addition, specify the directory to save state files in. State files characterize the Simulation Environment setup, including the Model Library File, convergence parameters, outputs, and design variables. Keep the default *./.artist_states* location (this will place the states in the project which launched the Cadence session), or choose one in your design database.

The option called **Preload the Corners Java** is a new feature that works with the Corners Analysis tool. It is turned on by default and can slow down your Simulation Environment start-up time. When not using the Corners tool, to turn it off or add the following statement to your *.cdsenv* file:

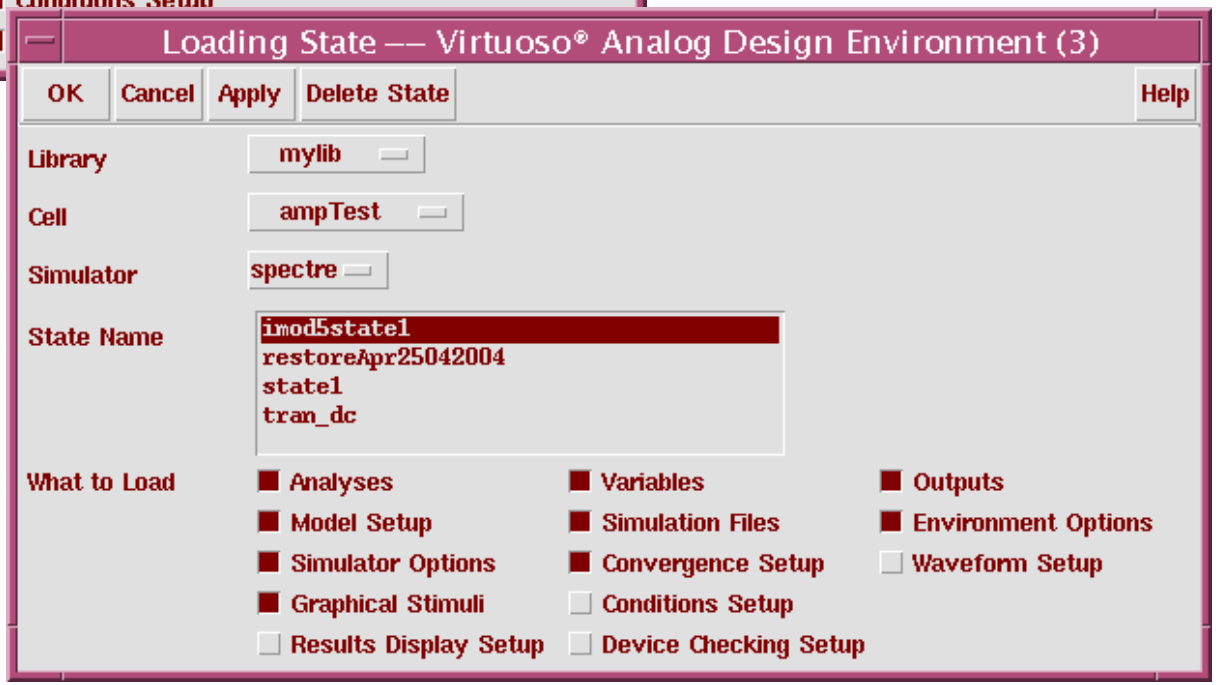
```
asimenv loadCorners     boolean nil
```

Also set the Default Design Open Mode and the location of the Simulation window.

Simulation States



Session—Save State



Session—Load State

Save State

This command saves simulation states during a session. Items that can be saved include Analyses, Variables and Outputs, and the Waveform Window. Save each of these items individually or in groups. Name the file anything by typing the name in the Save As field. The default name is **state1**.

By default, the files are saved under a directory called *./.artist_states*. Change the location of this directory by specifying a new directory in the State Save Directory field in the Editing Session Options form. Access this form with the **Session—Options** command in the Simulation window.

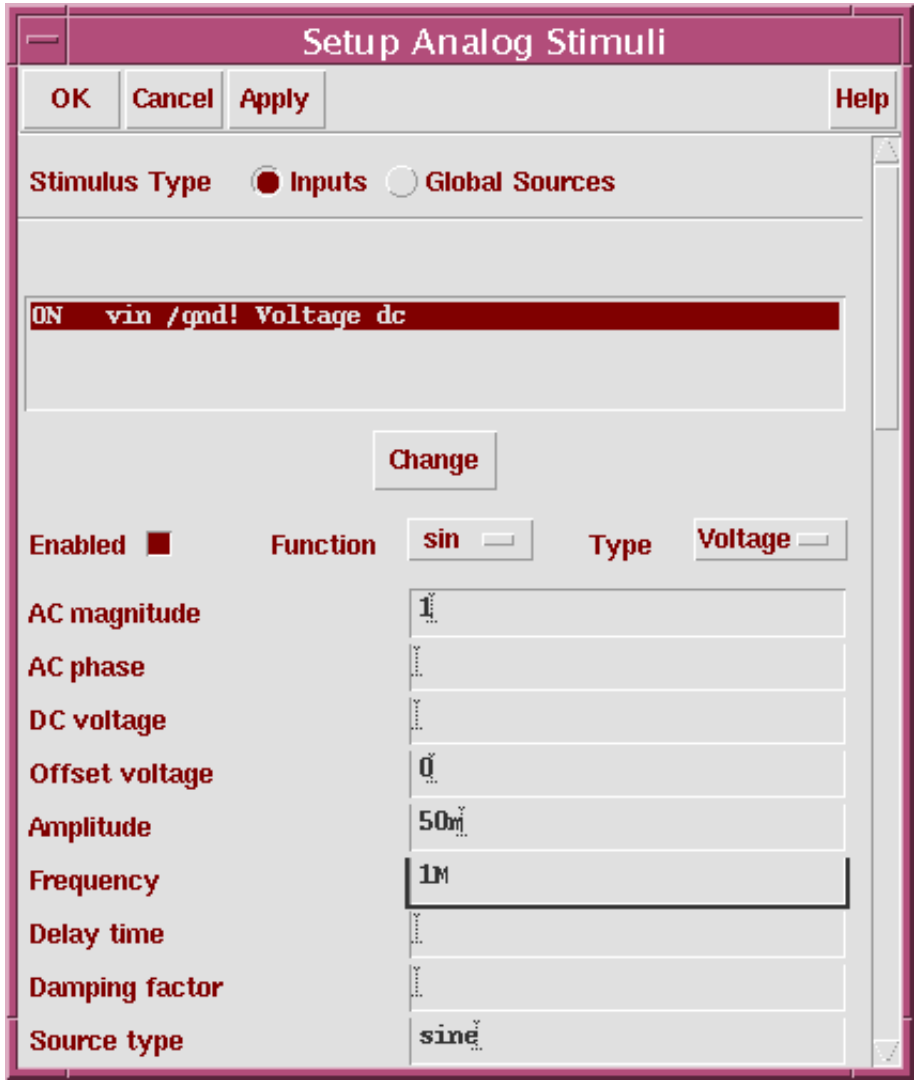
Load State

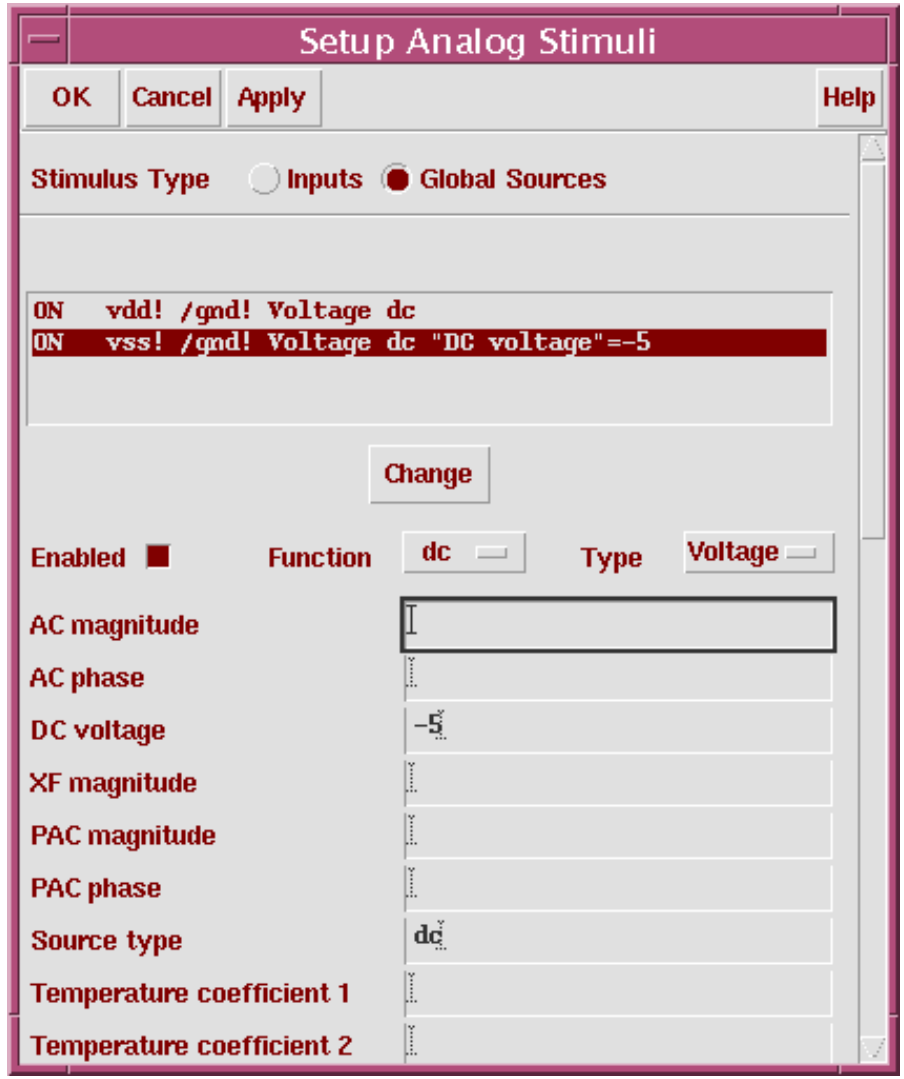
Use the **Load State** command to load saved states for a design. The cyclic fields next to Library and Cell are used to pick a particular design. The state files are simulator dependent, if Analyses are saved as part of the file. Use Outputs and Variables with any simulator. The State Name listbox will show all of the saved files for the design. Save individual objects.

This feature is particularly useful for adding items to the Outputs section. For example, the user might have specific equations that are always used for testing the AC stability of a circuit. Once these equations are defined, they can be loaded them into other designs very quickly.

Stimulus Template

Setup—Stimuli





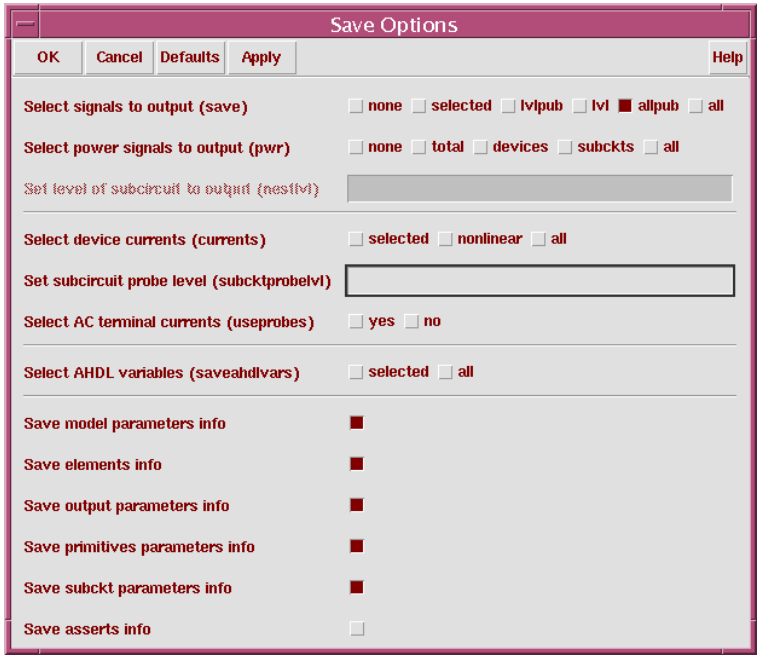
Use this graphical interface to create a stimulus file for specifying input stimuli and power supply stimuli to your design. Attach any type of source to the input pins or global pins that are in your design.

To use input stimuli, instantiate input pins into your top-level schematic for those stimuli signals. The power stimuli requires a defined global name on a signal (such as *vdd!*). Use this option to create designs that can run in multiple simulation scenarios, that do not require power sources and input stimuli that can clutter up the schematic window.

Note: To serve as optional stimuli, use both standard sources mixed with stimuli sources in your schematic. The stimulus template provides other options to creating stimuli for your circuit, depending on the analysis selected.

Save Options

Select: Outputs—Save All



Setting	Description
none	Does not save any data. (Currently saves one node chosen at random.)
selected	Saves only signals selected in schematic.
lvlpub	Saves all signals that are **normally useful** up to *nestlvl* deep in the subcircuit hierarchy. This option is equivalent to *allpub* for subcircuits.
lvl	Saves all signals up to *nestlvl* deep in subcircuit hierarchy. Relevant to subcircuits.
allpub	Saves only signals that are normally useful.
all	Saves all signals.

Specify which signals to save with the **save** parameter. Use the *nestlvl* parameter when saving signals in subcircuits. (Set **save** to *lvl* or *lvlpub*.) Signals that are “normally useful” include shared node voltages and currents through voltage sources and iprobes. If you use lvl or all instead of lvlpub or allpub, you will also get internal node voltages and currents through other components that happen to compute current. Thus, using *pub excludes internal nodes on devices (the internal collector, base, emitter on a BJT, the internal drain and source on a FET, etc). It also excludes the currents through inductors, controlled sources, transmission lines, transformers, etc.

To save power dissipated on a circuit, subcircuit, or device, use the *pwr* parameter. Power is calculated only during DC and transient analyses. The results are saved as a waveform, representing the instantaneous power dissipated in the circuit, subcircuit, or device.

The *nestlvl* parameter specifies how many levels deep into the subcircuit hierarchy to save signals. The default setting for *nestlvl* is infinity, which saves all levels.

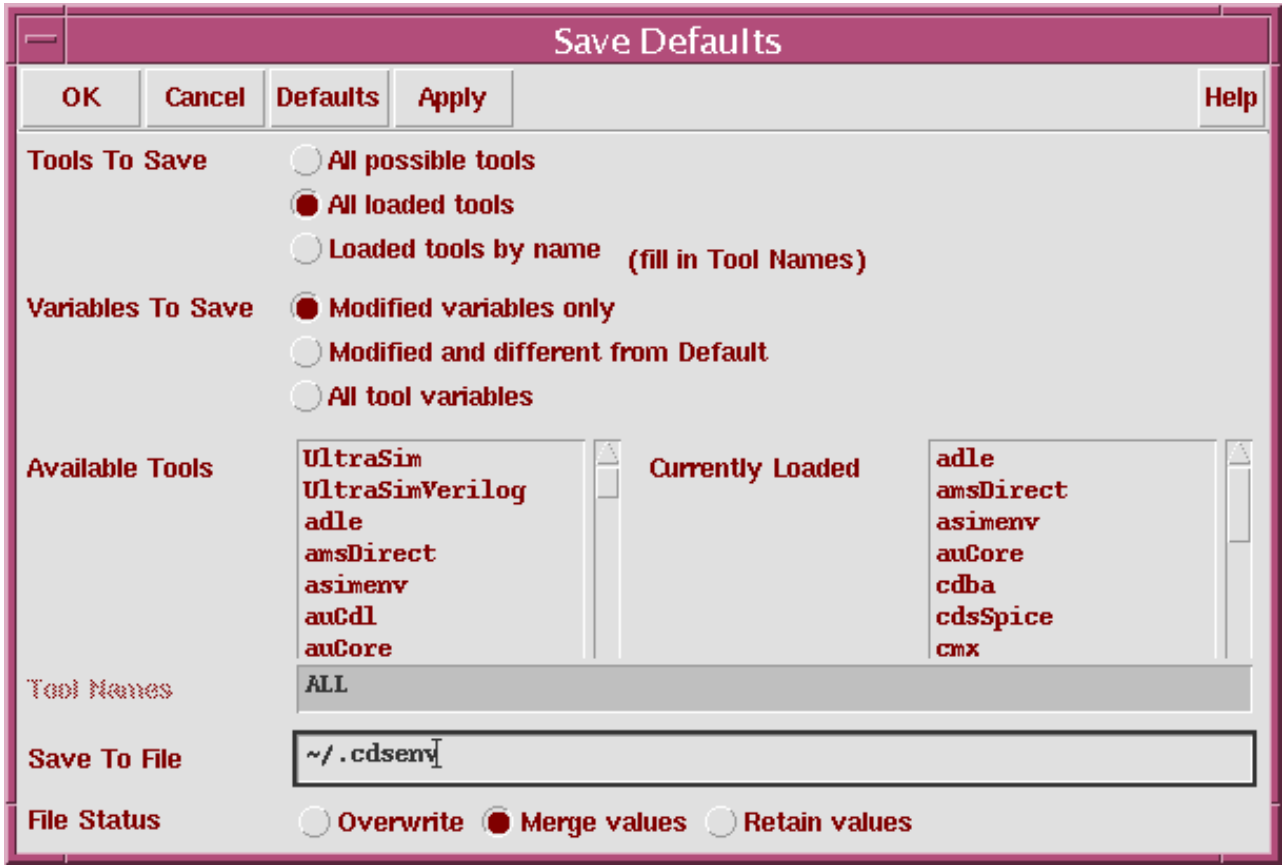
The **currents** parameter of the options statement computes and saves terminal currents. The **selected** parameter saves only currents select. The **nonlinear** parameter saves all terminal currents for nonlinear devices, naturally computed branch currents, and currents specified with save statements. The **all** parameter saves all currents. The **nonlinear** and **all** parameters can significantly increase simulation time.

Use the *subcktprobelvl* parameter to control the calculation of terminal currents for subcircuits. Current probes are added to the terminals of each subcircuit, up to *subcktprobelvl* deep. Specify all currents to be calculated with current probes by setting *userprobe* to yes. **Note that no AC currents will be saved if you do not userprobe to yes.** To save ahdl variables in ahdl instances, set *saveahdlvars* to **selected** or **all**.

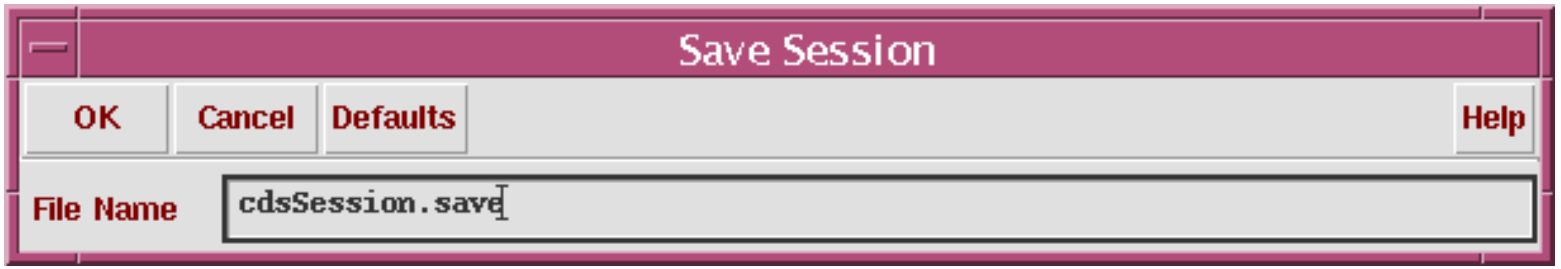
Sometimes there is a need to set a large number of current probes. This could happen, for example, if one needs to save a number of ACs. (Current probes can find such small signal currents when they are not normally computed.) Specify that all currents be calculated with current probes by placing *useprobes=yes* in an options statement.

To save all the ahdl variables belonging to all the ahdl instances in the design, set the *saveahdlvars* option to all using a Spectre options command. For example: *Saveahdl* options *saveahdlvars=all.*

Save Defaults and Save Session



In the CIW, select **Options—Save Defaults**.



In the CIW, select **Options—Save Session**.

The *.cdsenv* file contains default information for all of the tools in the executable. In the Virtuoso Analog Design Environment, the information that you set in the Editing Session Options form and the Setting Plotting Options form is saved in the *.cdsenv* file. Additionally, the Simulator and Project Directory are also saved.

File Status

- Overwrite: Stores the values entered in the Save Defaults form by overwriting the previous *.cdsenv* file in pointed to in the path.
- Merge values: Stores the modified values entered in the Save Defaults form in the previous *.cdsenv* file but does not delete pre-existing, unmodified values.
- Retain values: Stores the values entered in the Save Defaults form by creating another file. It is necessary to enter the name of the file in the Save to File field.

Save Session

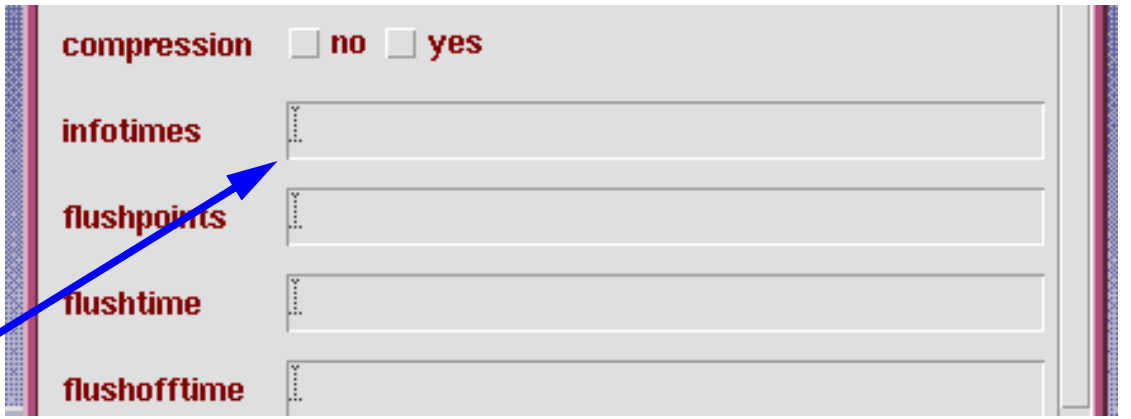
When starting the Design Framework II environment with a *-restore <sessionFileName>* option, the specified file is loaded. Windows that were active at the time of the session save will be restored, as well as each open form. Also restore sessions by typing *load ("<sessionFileName>")* in the CIW. Use caution when loading a session file in the CIW, because SKILL commands in the file might interfere with the current design session. The restored session will be added to the existing session, but no existing windows will close.

Infotimes

Infotimes is a transient analysis option that saves transient operating point information at specific timepoints in the simulation.

1. From the Simulation Window, select **Analyses—Choose**.
2. Select the **tran** button in the Choosing Analyses form.
3. Select **Options** at the bottom of the Choosing Analyses form.
4. A very long Transient Options form appears, scroll down to **infotimes**.
5. Enter infotimes, as shown.

The line on the Transient Options form for entering infotimes.



Enter your time points in this text field **in any order.**

Accessing the *infotimes* Text Field

Multiple values entered in this field should be separated by blank spaces. If invalid separators or non-numeric values are specified here, Spectre reports the error/warning in the simulation output window. For example, if you use "comma" between values, it works but gives you following warning:

```
Warning from spectre during circuit read-in.
"input.scs" 75: Use of the comma character in lists will not be
supported in future releases.
```

The infotimes netlist statements

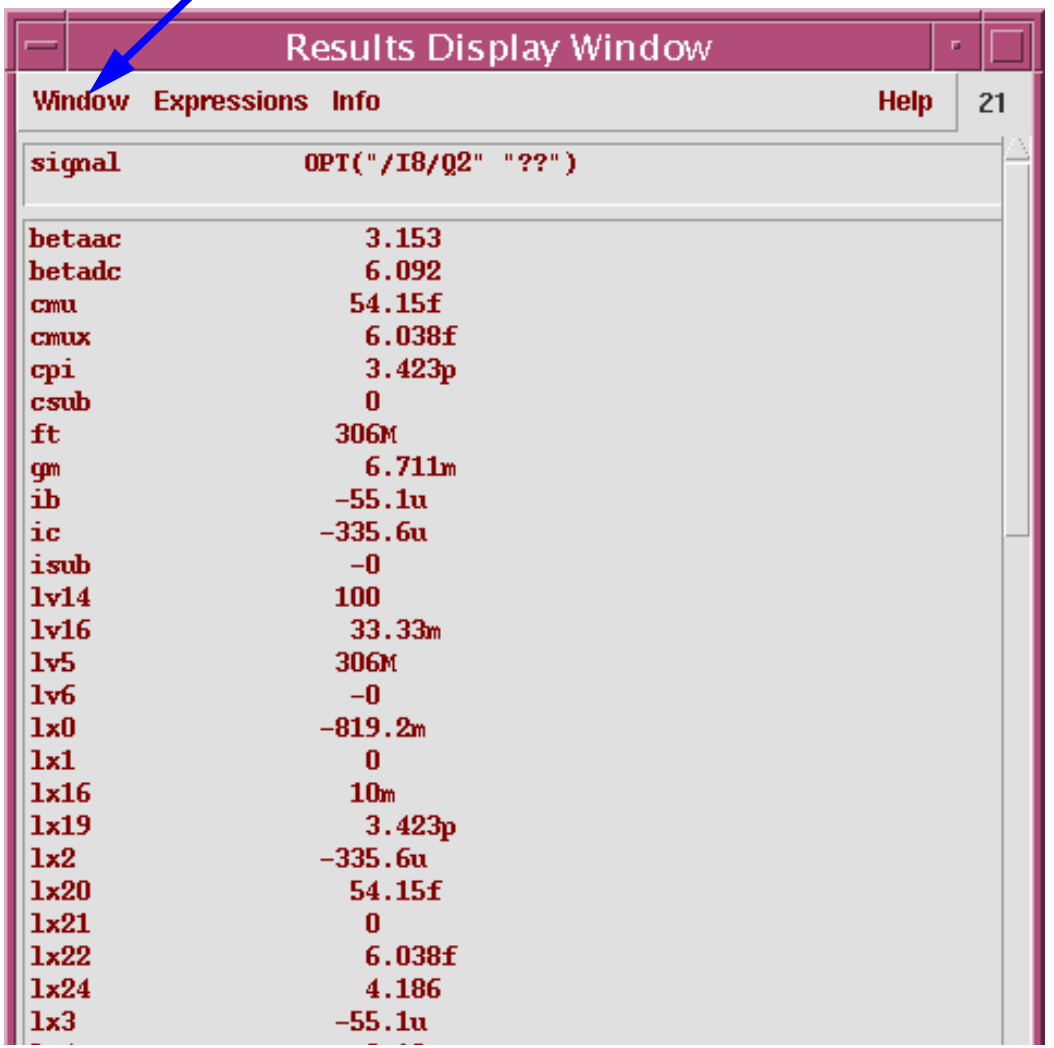
Netlist from ADE:

```
tran tran stop=100u write="spectre.ic" writefinal="spectre.fc" \
    annotate=status infotimes=[5u 29u 68u 87u] maxiters=5 \
    infoname=tran_Info
tran_Info info what=oppoint where=rawfile
```

Infotimes Results

Select **Results—Print—Transient Operating Points**, then click components on the schematic.

To print the data of this window as a file select: Window—Print



Scroll bar

Select additional components!

The scroll bar becomes smaller as additional devices are selected on the schematic. The file is getting larger, but the data is added to the bottom of the file.

- Once infotimes values are specified, netlist and run the simulation.
- When the simulation has been completed, select:

  Results—Print—Transient Operating Point
- A Results Display window appears. If you have not selected a component, the window will be empty. Select a component in the schematic.
- The operating point information appears in the window.
- Select another component, the data is added to the bottom. You may not see it; however, the scroll bar on the window gets smaller.
- To print the file, select **Window—Print** in the Results Display Window.

Captab

- Transient analysis option or dc analysis option
- Provides a table of node capacitances at specified times
- Has three node detail options: **node**, **nodetoground**, and **nodetonode**
- Has a threshold feature; default is 0.0F
- Similar to the CAPTAB option in HSPICE
- Used with infotimes in the transient analysis options form
- Simple to use

Captab provides a tabulation of node and device capacitance, either at the dc operating point or at the specified infotimes. The tabulation appears in the simulation output log file.

The CAPTAB parameters

detail = node

Provides details of the capacitance. Possible values are node, nodetoground, or nodetonode.

sort=name

How to sort the capacitance table. Possible values are name or value. If sort-by-value is selected, then the table will be sorted in a descending order of the *total node capacitance*. (The rows with the same “From_Node” will remain together.)

If sort-by-name is selected, then the table will be sorted in alphabetical order according to the “From_Node:To_Node” column.

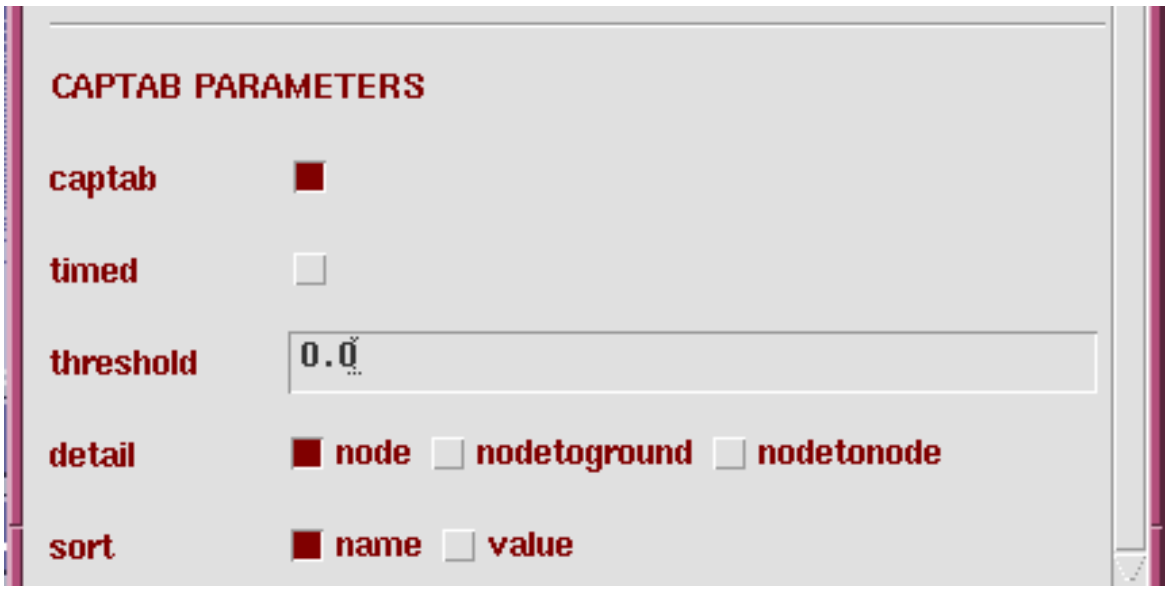
threshold=0 F

Threshold capacitance value for printing. This feature allows you to specify the threshold capacitance value. The nodes for which the *total node capacitance* is below the threshold value will not be included in the output.

Selecting the captab Option from ADE

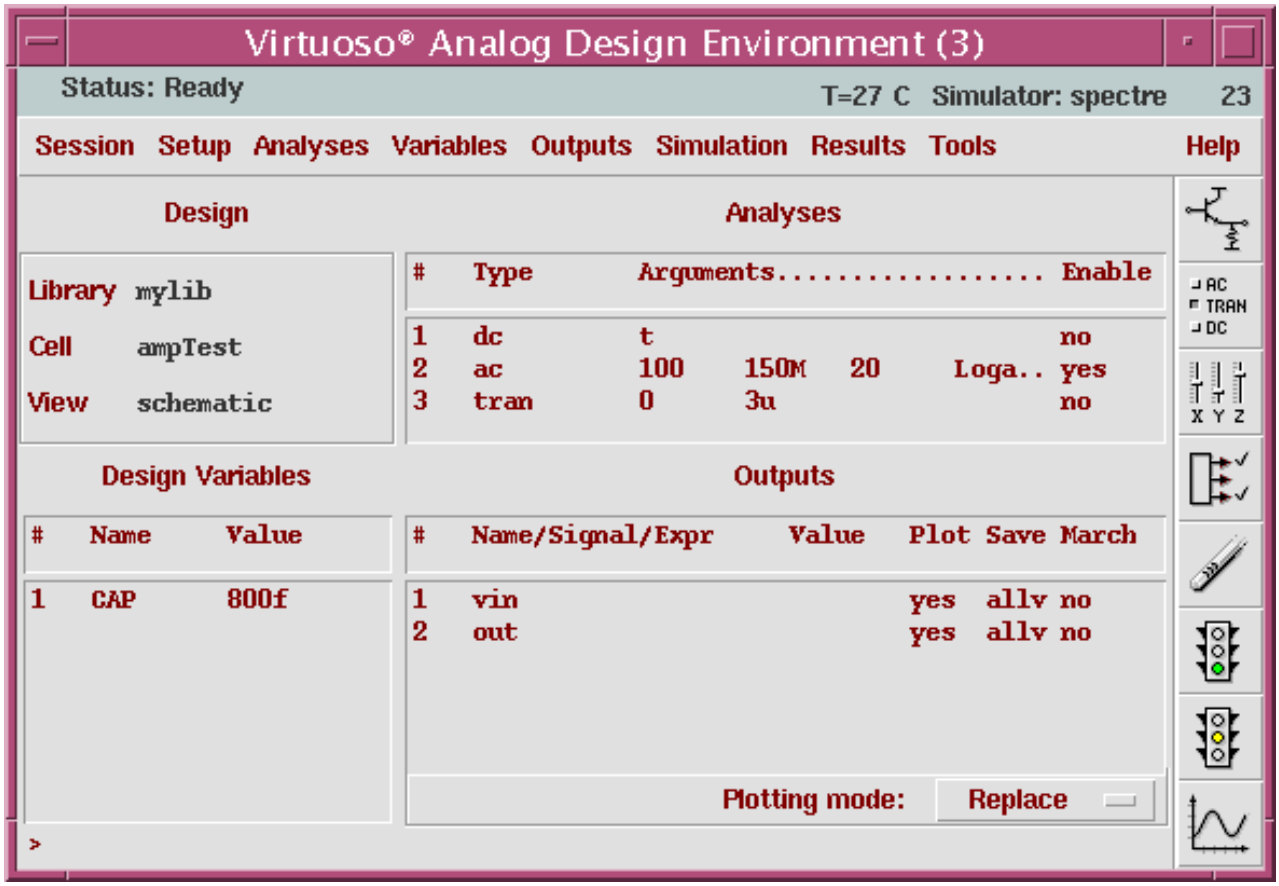
DC and Transient Analysis has CAPTAB PARAMETERS option. For transient analysis:

1. Select **Analyses—Choose**.
2. Select the **tran** button in the Choosing Analyses form.
3. Select **Options** at the bottom of the Choosing Analyses form.
4. On the Transient Options form scroll down to infotimes.
5. Enter infotimes.
6. Scroll down to CAPTAB PARAMETERS at the bottom of Transient Options form.
7. Enter CAPTAB selections.



Labs for Module 3

Analog Simulation with Spectre Direct



Lab 3-1 Running Simulation

Objective: To set up and run simulations on the ampTest design.

Before starting this lab, make sure that your *ampTest* schematic in the *mylib* library is open.

Starting the Simulation Environment

Start the Simulation Environment to run a simulation.

1. In the *ampTest* schematic window, execute **Tools—Analog Environment**.

   In a few moments, the Analog Design Environment simulation window appears.

Choosing a Simulator

Set the environment to use the Spectre® tool, a high speed, highly accurate analog simulator. Use this simulator with the *ampTest* design, which is made up of analog components.

1. In the simulation window, execute **Setup—Simulator/Directory/Host**.

2. In the Choosing Simulator form, set the Simulator field to **spectre** (not spectreS) and click **OK**. This should already be set in the *.cdsinit* file, which was read during invoking icms &.

Setting the Model Libraries

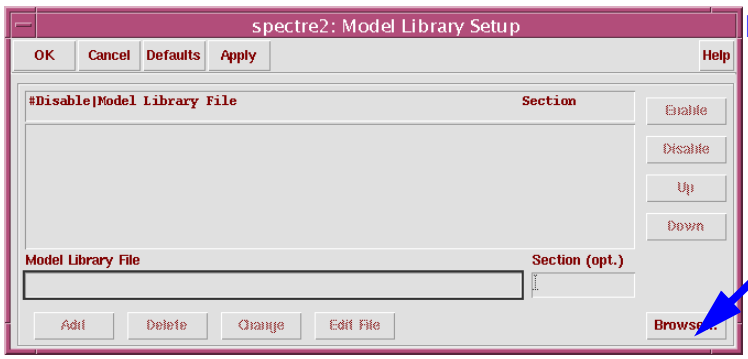
The Model Library file contains the model files that describe the *nmos4*, *pmos4*, *npn*, and *pnp* devices during simulation.

Important

*IMPORTANT! To avoid problems in completing this lab activity and the following lab activities, the next **7** steps must be performed exactly as stated.*

1. Execute **Setup—Model Libraries**.

   The Model Library Setup form appears. The first time you set up the simulation environment, this form is empty as shown.



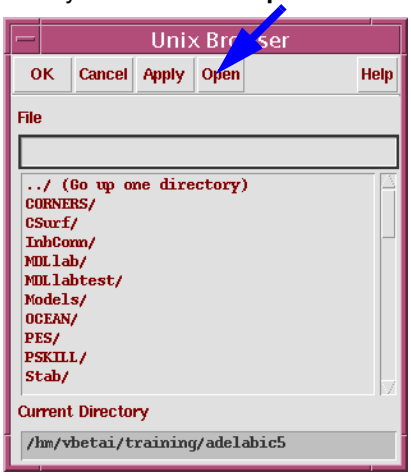
Form is empty!

For step 2, select the Browse... button.

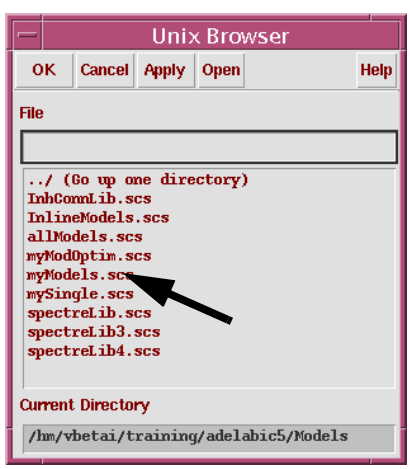
2. Place your cursor at the lower right corner of this form and click the **Browse** button.

3. A UNIX Browser form appears. This form shows the current directory. We are “browsing” for a model file, so move the cursor and select the **Models/** library, then click the **Open** button. Use the cursor to select **myModels.scs.** The model file appears in the text field for File.

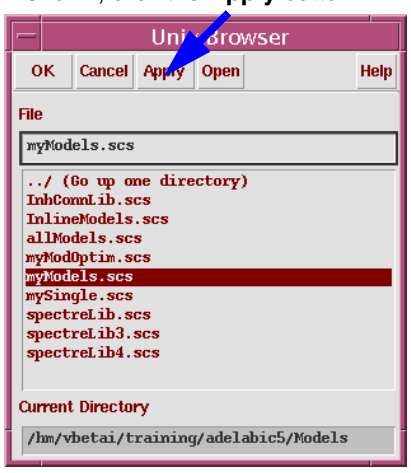
Use the cursor to select the library and then click **Open.**



The Model Files are Displayed. Use the cursor to select a file.



With the model file selected as shown, click the **Apply** button.

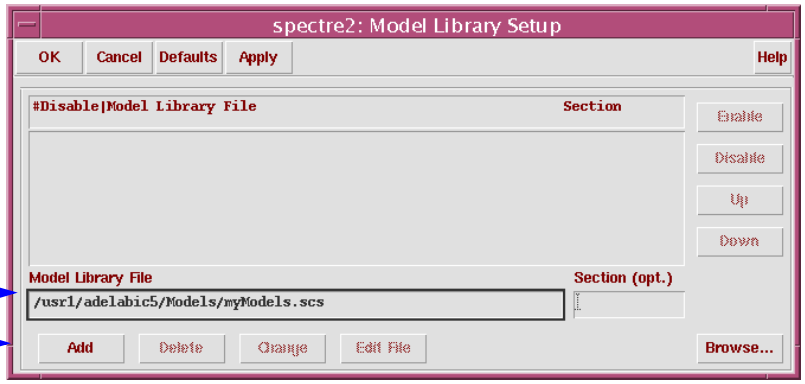


4. In the UNIX Browser form, click the **Apply** button, and then **OK**.

5. The path of the model library and the model file now appear in the text buffer of the Model Library Setup window.

Your path in the text buffer should look **about** like this!

Click **Add**

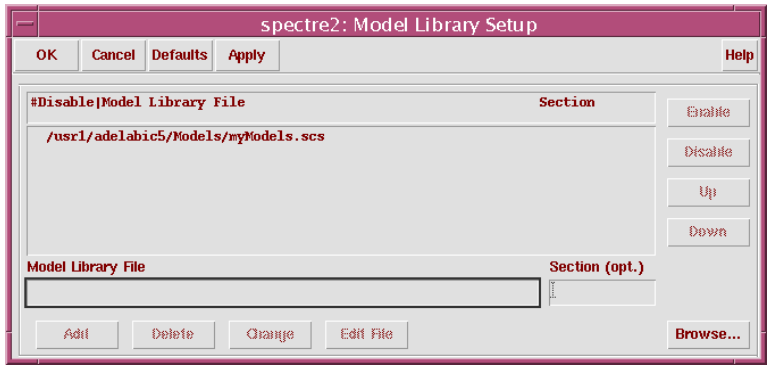


At this point the model file path is only in a text buffer. It needs to be moved into the active **Model Library File** field above.

6. Click the **Add** button.

   The path of the model file moves from the text buffer to the section above in the form. Your Model Library Setup window should now look the figure below.

Does your window look like this?



Note: **This section of this step is optional!** To view the model file, highlight the expression in the Model Library File field and click **Edit File**.

7. To complete the Model Library Setup, move the cursor and click **OK**.

   The Model Library Setup allows you to include multiple model files. It also allows you to use the Edit button to view the model file in the text buffer before moving it into the active field.

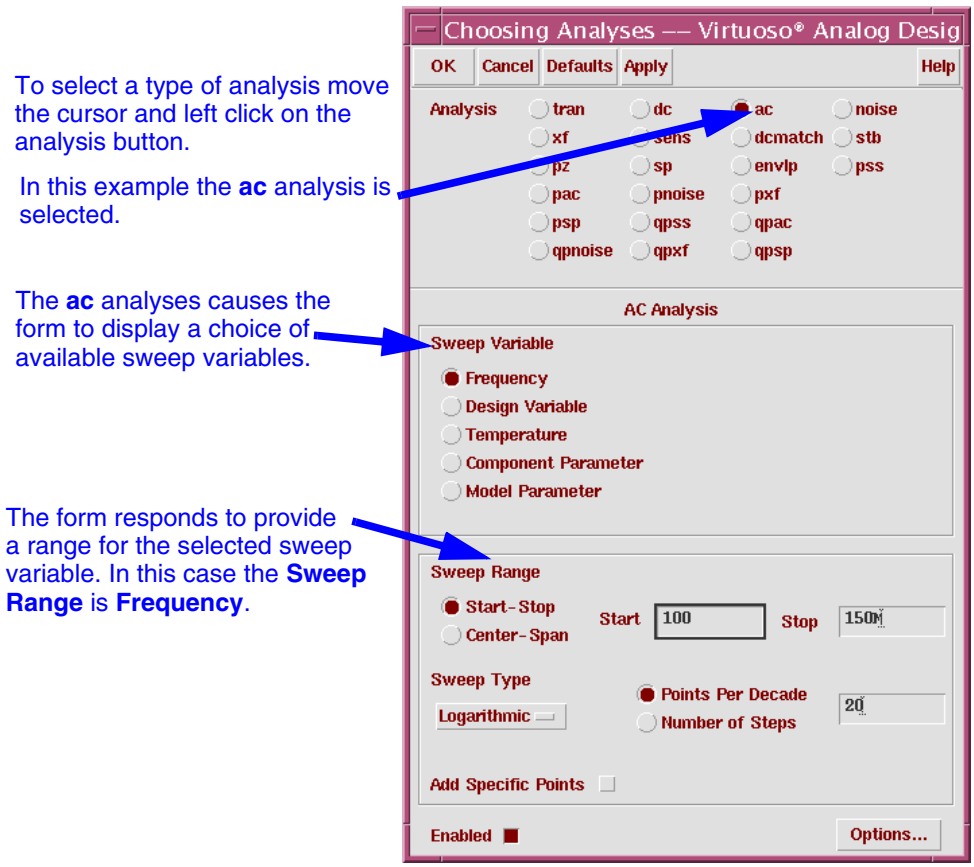
Choosing Analyses

This part of the lab activity demonstrates how to view and select the different types of analyses to complete your circuit when running the simulation. You will select and run multiple analyses on the *ampTest* design.

1. In the Simulation window, click the **Choose Analyses** icon.

   Tip: You can also execute **Analyses—Choose**.

   The Choosing Analyses form appears. Take a few moments to become familiar with the form. This form is used often throughout the lab activities. It is also a dynamic form: the bottom of the form changes based on selections above.



2. To set up for **ac** analysis:

   a. In the Analysis section select **ac**.

   b. Set the Sweep Variable to **Frequency**.

c. Set the Sweep Range **Start** to **100** and **Stop** to **150M**.

d. Set the Sweep Type to **Logarithmic**, and **20** Points Per Decade.

e. Select **Enabled**, and then **Apply**.

3. To set up for Transient Analysis:

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

   b. Set the Stop Time field to **3u**.

   c. Set errpreset to **Moderate**. (This will automatically enable the *Enabled* field.)

   d. Then click **Apply**.

4. To set up for DC Analysis:

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

   b. In the DC Analysis section, turn **on** Save DC Operating Point. (This will also enable the *Enabled* field.)

5. Click **OK** in the Choosing Analyses form. The DC analysis is applied and the form closes.

Setting Design Variables

Set the values of any design variables in the circuit before simulating. Otherwise, the simulation will not run.

1. In the Simulation window, click the **Edit Variables** icon.

   Tip: You can also execute **Variables—Edit**.

   The Editing Design Variables form appears.

2. Click **Copy From** at the bottom of the form.

   The design is scanned and all variables found in the design are listed.

   In a few moments, the *CAP* variable appears in the Table of Design Variables section.

3. Locate the *CAP* variable in the *ampTest* design:

   a. Highlight the *CAP* name in the Editing Design Variables window and click on **Find**.

      The *amplifier* schematic instance is highlighted.

   b. Execute **Design—Hierarchy—Descend Edit**.

   Tip: You can also press the **E** (Shift E) key.

   c. In the Descend form that appears, set the View Name to **schematic**, and click on **OK**. The *amplifier* schematic appears.

   d. Click **Find** again and note that the capacitor instance is highlighted.

      You have located the *CAP* design variable.

   e. Return to the *ampTest* design with the **Design—Hierarchy—Return** command.

   Tip: You can also press **^e** (**Control e**).

Notes About Find

The **Find** key is useful for finding variables in large designs with many levels of hierarchy.

If you notice extra variables in the Design Variables form, there are typographical errors in the component parameter values in your design.

For example, if you see a variable named *Ohms*, you might have entered *20K Ohms* instead of *20K* for the resistor value in the design. You do not need to enter units for components, such as resistors, capacitors, and voltage sources. They are added automatically.

To fix the problem, use the **Find** key to locate the erroneous variable. Use the **Edit—Properties—Objects** command to modify the incorrect property and **Apply** your changes.

Next, select the extra variable from the Table of Design Variables, **Delete** it, and save your design.

1. Set the value of the *CAP* variable:

   With the *CAP* variable highlighted in the Table of Design Variables, enter the following:

Value(Expr)	`0.8p`

   Click **Change** and notice the update in the Table of Design Variables.

2. To copy the value of *CAP* back to the schematic, click **Copy To** at the bottom of the form.

   Saving the schematic cellview will then keep the value of *CAP* with the schematic.

3. Click **OK** or **Cancel** in the Editing Design Variables window.

4. **Check and Save** your *ampTest* design.

Saving Simulation Data

The Simulation environment is configured to save all node voltages by default. You can modify the default to save all terminal currents as well.

In larger designs, where saving all of the data requires too much disk space, you can select 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, execute **Outputs—Save All**.

   The Save Options form appears.

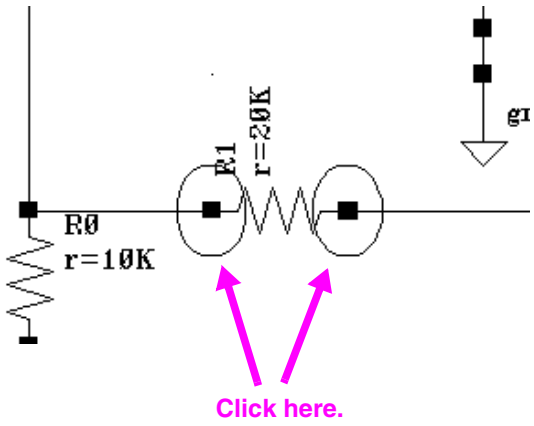
2. In the “Select signals to output (save)” section, make sure **allpub** is selected.

3. Click **OK** in the Save Options form.

4. To select specific terminals for which you save the signal current, execute **Outputs—To Be Saved—Select On Schematic** in the Simulation window.

   A prompt appears at the bottom of the *ampTest* schematic window.

5. In the schematic, click on the **terminal pins** on both sides of the 20K feedback resistor. An ellipse around each of the pins indicates that the signal current through the terminal will be saved.



6. Press **Esc** with your cursor in the schematic window to cancel the selection process.

Saving Outputs for Plotting

Select the nodes to plot when simulation is finished.

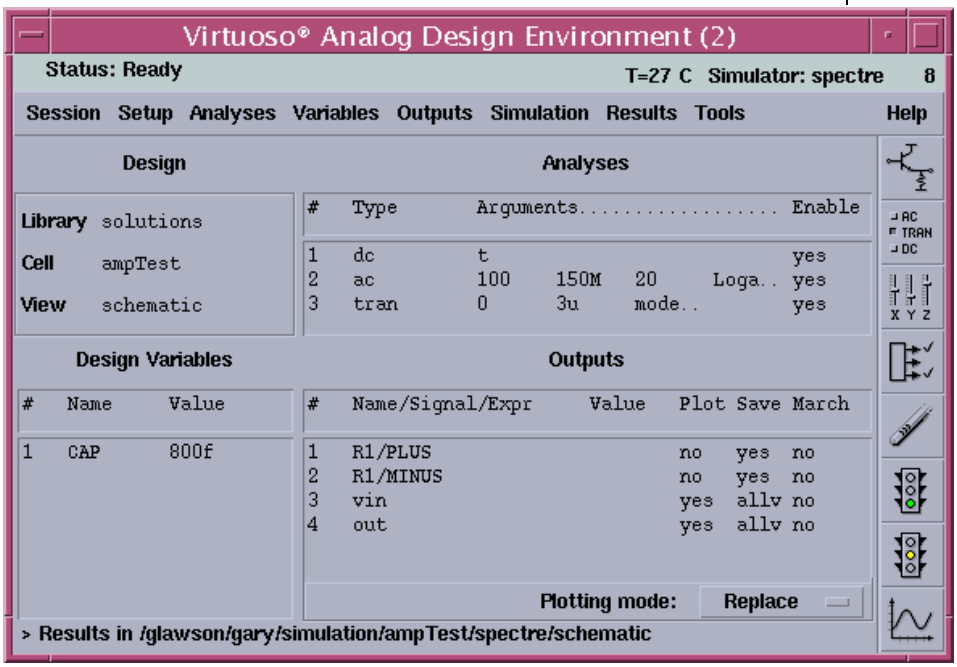
1. Execute **Outputs—To Be Plotted—Select On Schematic**.

2. Follow the prompts at the bottom of the schematic window. Click on the node (wire) labeled *vin* and *out*.

3. Press **Esc** with your cursor in the schematic window.

   Note the updates in the Outputs section of the Simulation window.

   Does your simulation window look like this?



Viewing the Netlist

The netlist is generated automatically when the simulation is run; however, you can view the netlist beforehand with the following procedure.

1. In the Simulation window, execute **Simulation—Netlist—Create**.

   A window appears that displays a hierarchical netlist. Note how all of the parameters 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.

   Note: To create a new netlist, execute **Simulation—Netlist—Recreate**. The recreate option updates any variable changes to the netlist.

2. Execute **File—Close Window** in the netlist window.

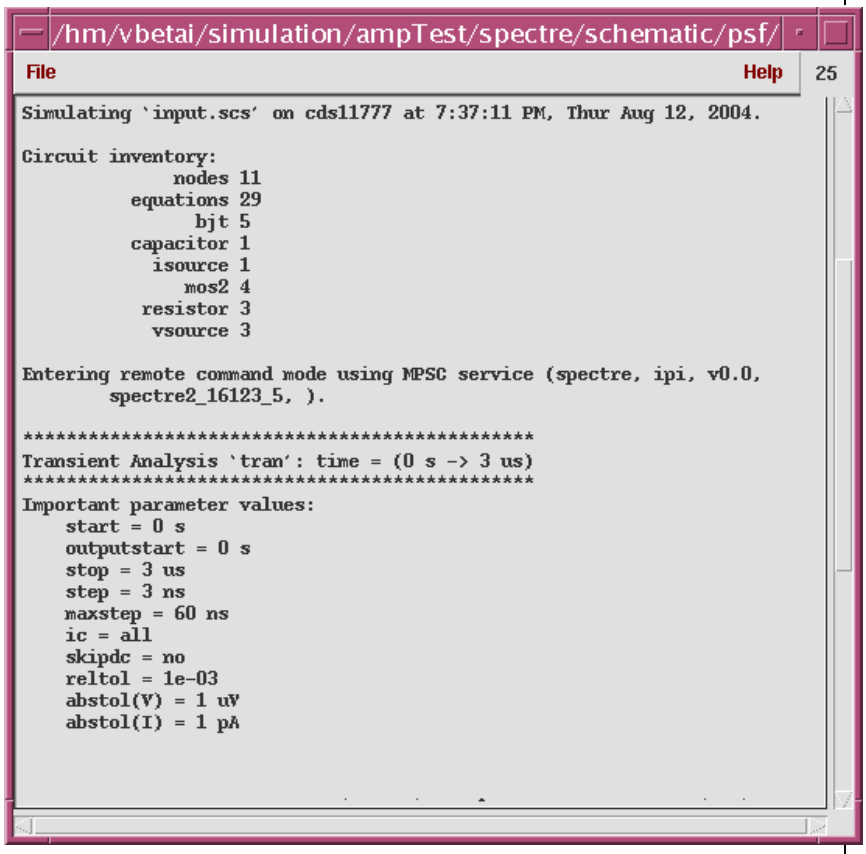
Running the Simulation

1. Execute **Simulation—Run** to start the simulation or click the **Run** icon in the Simulation window.

   (As an alternative, you can execute **Simulation—Netlist and Run** or click the **Netlist and Run** icon to create the netlist and run simulation from the start.)

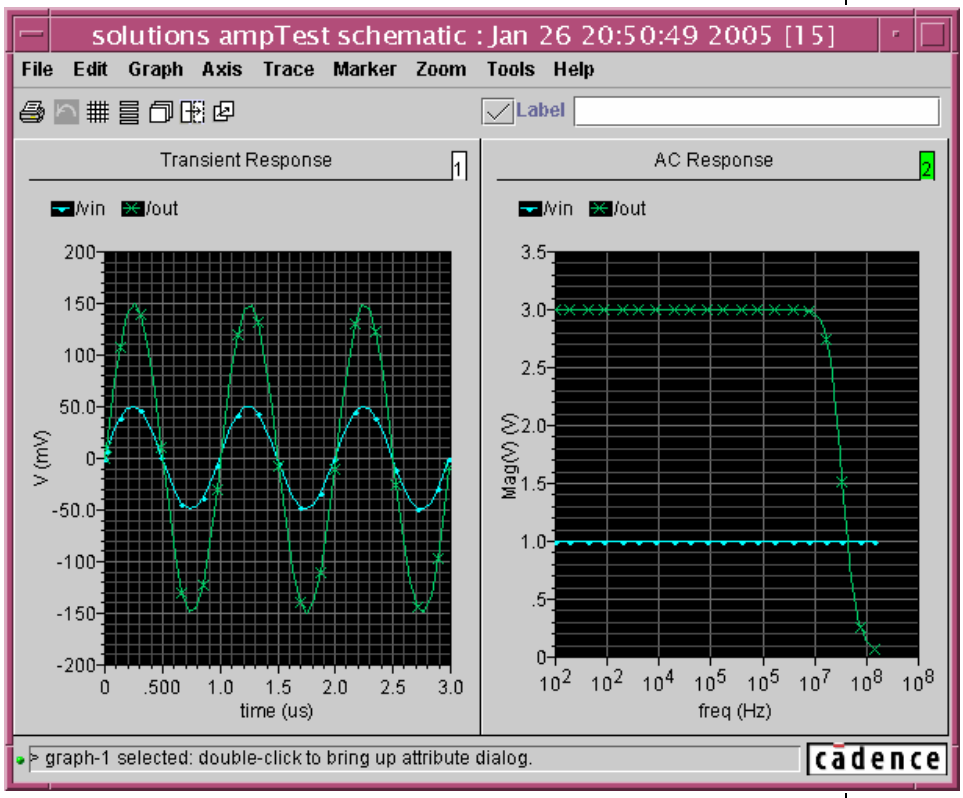
   As the simulation starts, messages appear in the CIW.

   A separate Spectre Output window like this one appears also.



This window is referred to as the simulation output window or the Spectre Output window. You can close it by executing **File—Close Window**.

2. When the simulation finishes, the Transient and AC Responses plot automatically.



Saving the Simulator State

You can save the simulator state, which stores information such as the Model Library file, outputs, analyses, environment options, and variables. This information restores the Simulation environment without having to type in all of the settings again.

1. In the Simulation window, execute **Session—Save State**.

   Note: This is an important step that allows you to restore your Simulation environment later on.

   The Saving State form appears.

2. Set the **Save As** field to **state1** and make sure all options are selected under the What to Save field.

3. Click **OK** in the Saving State form.

   The simulator state is saved.

   Optional: To check, use the mouse to select and the entries in the Output field, then execute **Outputs—Delete**. In the Design Variables field use the mouse to select the all entries, then execute **Variables—Delete** and close the Waveform Window. Now, execute: **Session—Load State**. The Load State window appears. Enter state1 if not yet selected, then left click **OK**. The simulator window with all entries, and the Waveform Window, are now restored.

Viewing Simulation Data with Snapshot

You can view simulation results before the entire run is complete. You will modify the maximum timestep taken by the simulator in order to slow down your simulation and make it more accurate. This will give enough time to take several "snapshots" of the data during the run.

In general, it is not recommended to change the maximum timestep taken by the Spectre simulator, because it is automatically adjusted for proper convergence and accuracy without manual interaction.

1. In the Analyses section of the Simulation window, highlight the lines for both the **ac** and **dc** analyses.

   Hold down the **Control** key to select both lines.

2. In the Simulation window, execute **Analyses—Disable**.

   The Enable field in the Analyses section of the window now lists *"no"* for the **ac** and **dc** analyses and lists *"yes"* for the **tran**sient analysis.

3. Double click the line for the **tran**sient analysis.

   The Choosing Analyses form appears.

4. At the bottom of the form, click the **Options** button.

   The Transient Options form appears.

5. Locate the TIME STEP PARAMETERS section, and set the following:

maxstep	`100p`

6. Click **Apply** in the Options form.

   This allows the transient analysis to run long enough for you to capture several “snapshots” before the simulation finishes.

7. Execute **Simulation—Run** to start the simulation or click on the **Run** icon in the Simulation window.

8. Look in the Spectre Output window, and note when the first time points in the Spectre simulation are taken. At this time, click the **Plot Outputs** icon or execute **Results—Plot Outputs—Transient** in the Simulation window.

   The Waveform Window plots the completed part of the simulation.

9. Click the **Plot Outputs** icon again, and watch the Waveform Window update.

   As the simulation ends, the full **3u** of simulation data is displayed in the Waveform Window.

10. Click **Cancel** in the Choosing Analyses form, if it is still open.

Preparing for the Next Lab

You will again restore the Simulation environment to the state that was created for running both the **ac** and **tran**sient analyses. This original state will be used for simulation in the next lab.

1. In the Simulation window, execute **Session—Load State**.

2. In the Loading State window, select

State Name	`state1`

3. Disable the button for **Waveform Setup** in the What to Load field.

4. Click **OK** in the Loading State form.

   Your previous simulation state is restored without the Waveform Window. If you attempt to restore the Waveform Window setup, you get messages indicating that the **ac** analysis results are not available in the data set that was created by the most recent simulation, which ran a transient analysis only.

5. In the *ampTest* schematic window, execute **Design—Probe—Remove All** to remove all colored probes that might remain from plotting the simulation results.

6. If the simulation output window is still open, close it by executing **File—Close Window**.

7. Leave your Simulation window, Waveform Window, and all other design windows as they are.

   You will explore the stimulus template in the next lab.

Lab 3-2 Using the Stimulus Template

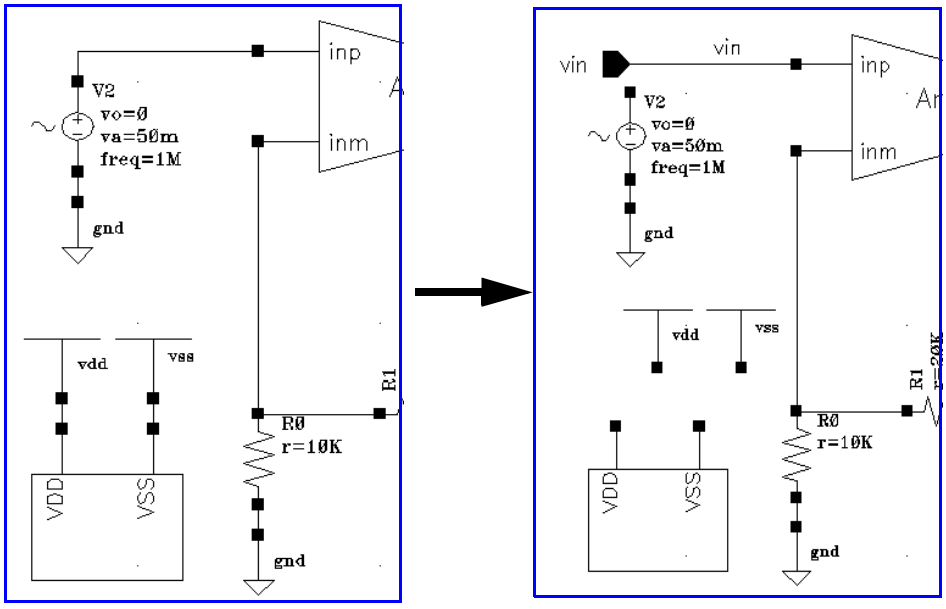
Objective: To use the graphical stimulus template and simulate the ampTest circuit.

Through a graphical interface, you can create a stimulus file that specifies input stimuli and power supply stimuli for your design. For input stimuli, your top-level schematic must contain input pins for the signals that you plan to set. To use the power stimuli, you must use a global name on a signal (such as *vdd!*). This allows you to create designs that can run in multiple simulation scenarios and that do not require power sources and input stimuli that can clutter up the schematic window.

You will modify the *ampTest* circuit by disconnecting the input sinusoidal voltage source and the supply block from the *vdd* and *vss* global symbols. You will create a stimulus file using the graphical interface to provide these signals. You will leave the input *vsin* voltage source and *supply* block in the window so that you can connect them again after this test. You can delete these two blocks and the design will run after you have created the stimulus file. However, you will use the *vsin* and *supply* blocks for later simulations.

Modifying the *ampTest* Design

1. Modify the left side of your *ampTest* schematic to look like this:



Before modifications **After modifications**

a. In the *ampTest* design, delete the wire connecting the *vsin* source to the **inp** terminal of the amplifier block. Also delete the wires connecting the *vdd* and *vss* symbols to the *supply* block.

b. Move the *vsin* source along with the attached *gnd* symbol and the *supply* block to the left as shown in the diagram.

c. Click the **Pin** icon in the schematic window and add an *input* pin with the name **vin** to your schematic.

d. Add a wire that connects the new pin to the **inp** port of the *amplifier*.

These actions complete the topographical modifications.

e. Click the **Check and Save** icon in the schematic editor window.

You will see a warning message in the Schematic Check box that appears. Note the flashing markers in the schematic window. These markers indicate your unconnected terminals.

f. Click **Close** in the Schematic Check box.

Ignore the warning messages for now.

Creating the Stimulus File

The graphical stimulus template finds all of the global signals and input pins in your design and allows you to create stimuli for them. You will create a stimulus file to replace the function of the *supply* and *vsin* components.

1. In the Simulation window, execute **Setup—Stimuli**.

   The Setup Analog Stimuli form appears.

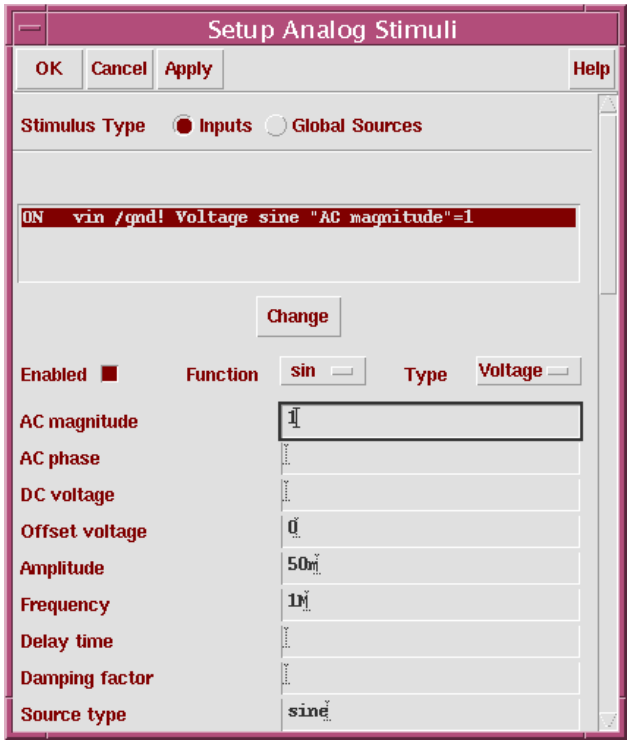
2. In the Setup Analog Stimuli form, make sure the Stimulus Type is set to **Inputs**.

   Note that an expression for the input pin, **vin**, appears near the top of the form.

3. Update the following options in the form and click **Change**.

Function	sin
AC Magnitude	1
Amplitude	50m
Frequency	1M

Does your form look like this?



These settings connect a sinusoidal voltage source to the pin, **vin**, with the same stimuli provided by the *vsin* source earlier.

4. Next, set the Stimulus type to **Global Sources**.

   Set the dc voltage sources that connect to the *vdd* and *vss* global signals.

5. In the Setup Analog Stimuli form, highlight the input for *vss*.

6. In the form, make sure the Function field is set to **dc** and change only the following:

DC voltage	-5

   Make sure that there are no entries for *AC magnitude* or *AC phase*.

7. Click **Change**.

8. Next, highlight the input for *vdd*.

9. In the form, make sure the Function field is set to **dc** and change only the following:

DC voltage	5

   Make sure that there are no entries for *AC magnitude* or *AC phase*.

10. Click **Change**.

11. Turn on **Enabled**.

12. Click **OK** in the Setup Analog Stimuli form.

    The stimuli for the *ampTest* design are available for simulation.

Running Simulation Using the Graphical Stimulus Template

Run the simulation and use the stimulus file that you created with the graphical interface. Before you run the simulation, you must recreate the netlist for the design because the circuit topology changed. If you do not create a netlist again, you will get errors when running simulation.

1. In the Simulation window, execute **Simulation—Netlist—Recreate**.

   In a few moments, the netlist appears. Study it to see how your stimuli appear in the netlist.

2. Execute **File—Close Window** in the netlist window.

3. Execute **Simulation—Run** to start the simulation or click on the **Run** icon in the Simulation window.

   When the simulation run is complete, the Waveform Window displays the results. The results are the same achieved using the *supply* and *vsin* components. If your results are different, check your setup.

4. If the simulation output window is still open, close it by executing **File—Close Window**.

Preparing for the Next Lab

You will restore the *ampTest* schematic to its original form for use in following labs. This will include turning off the stimuli that you created with the template, as well as recreating the netlist, since the topology has changed again.

If you do not turn off the stimuli that you created, they will appear as sources in parallel with your schematic sources when you restore them.

1. In the Simulation window, execute **Setup—Stimuli**.

   The Setup Analog Stimuli form appears.

2. In the Setup Analog Stimuli form, set the Stimulus Type to **Inputs**.

3. Highlight the entry for **vin**, turn *off* the **Enabled** field, and click **Apply**.

   The entry changes to OFF.

4. Next, set the Stimulus Type to **Global Sources**.

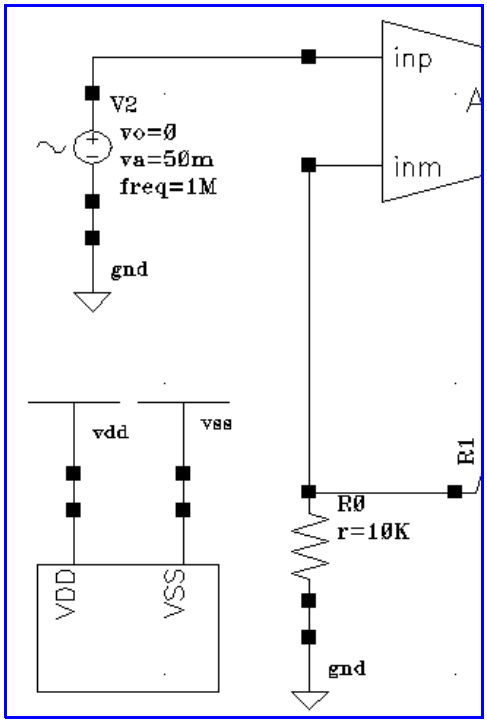
5. Highlight the entry for the **vss!** supply, turn *off* the **Enabled** field, and click **Apply**.

   The entry changes to OFF.

6. Highlight the entry for the **vdd!** entry and turn it *off*, too.

7. Click **OK** in the Setup Analog Stimuli form.

8. In the schematic, delete the input pin, *vin*, and modify the left side of your *ampTest* schematic to look like the following:



   a. Click the **Wire (narrow)** icon and wire up your design. Use the **Wire Name** icon to add the *vin* label, if necessary.

   b. Click the **Check and Save** icon in the schematic editor window.

      The flashing markers and warnings messages no longer appear, because the schematic is properly connected.

   c. In the *ampTest* schematic window, execute **Design—Probe—Remove All** to remove all colored probes that might remain from plotting the simulation results.

9. In the Simulation window, execute **Simulation—Netlist—Recreate**.

   In a few moments, the netlist appears.

10. Execute **File—Close Window** in the netlist window.

11. Execute **Simulation—Run** to start the simulation or click on the **Run** icon in the Simulation window.

    When the simulation is complete, the Waveform Window displays the correct results.

    If you have any error messages, check the setup before proceeding to the next lab. Possible errors might result from some of the graphical stimuli still being active.

12. If the simulation output window is still open, close it by executing **File—Close Window**.

13. Leave your Simulation window, Waveform Window, and all other design windows as they are.

    You will explore the Waveform Window in the next lab.

Lab 3-3 Transient Operating Point Analysis, “infotimes”

Objective: To use the Virtuoso Analog Design Environment to obtain parametric data during a transient simulation.

Overview of infotimes

During the process of designing circuits and then evaluating performance, it is often important to investigate the processes occurring within the circuit. These investigations are perhaps necessary perhaps due to critical timing requirements. Perhaps the investigation is due to unexpected circuit performance when the circuit is simulated. Whatever the reason, the processes occurring within the circuit need to be known.

The dc operating point information of a circuit provides detailed information on the state of the circuit prior to starting a transient analysis. As such, the processes occurring in the circuit at the beginning of the transient simulation are available. In a transient operating point analysis, the processes within the circuit are evaluated at a specific time. The values of individual device currents and voltages are then measured. This creates a “snapshot” of the circuit operation. Infotimes is a feature within the Virtuoso Analog Design Environment for invoking transient operating point analysis.

Starting the Environment for Transient Analysis

1. Start a design session. In a terminal window, change your current directory to ~/adelabic5. Then in the terminal window enter:

   ```
   icms &
   ```

   A CIW appears.

2. In the CIW, start the Library Manager by executing **Tools—Library Manager**.

3. In the Library Manager, select *mylib,* and the open the ampTest schematic.

4. In the schematic window, start the simulation window by executing **Tools—Analog Environment**.

5. Load the saved state for the ampTest circuit by executing **Session—Load State**.

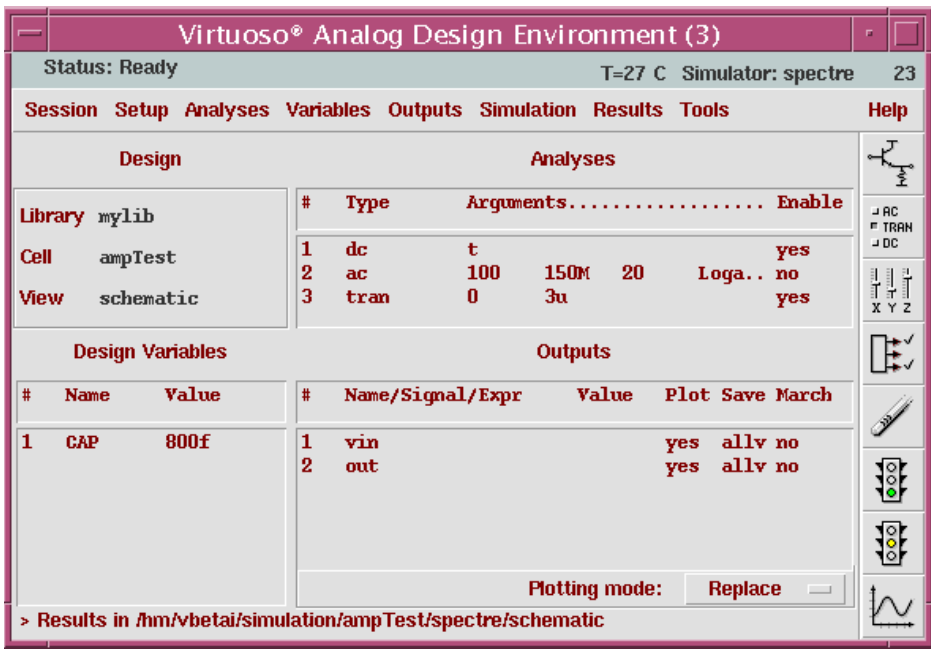
6. In the Loading State form, select **state1** and then click **OK**.

   The state that is loaded (state1) should return your simulation window to a condition of being “simulation ready”. All fields (Design, Variables, Analyses, and Outputs) of the simulation should be filled in. In addition, your model files have been also been set up.

7. You need to disable the ac analyses. Select the **ac** analysis line in the Analyses field to highlight it, then execute **Analyses—Disable**.

   The ac analysis is now disabled for the next simulation run. However, the setup of the ac analysis still exists, as shown in the Analyses field. Likewise, verify that the transient analysis is enabled.

8. Verify that your simulation window appears as shown. Then execute **Simulation—Netlist and Run**.



9. Verify that the transient simulation ran properly and that the output waveform shows a voltage gain of 3.

Entering the Time Points for Analysis

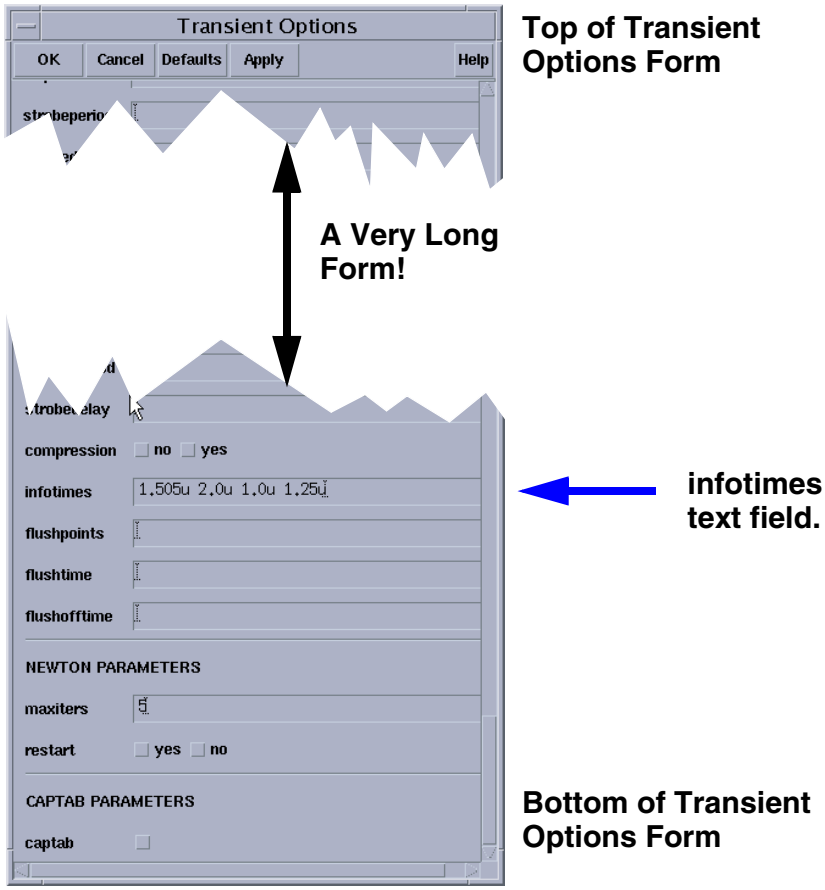
1. In the simulation window, execute **Analyses—Choose**.

2. Make certain that the Choosing Analyses form is set to **tran**.

3. At the lower right corner of the Choosing Analyses form, click Options.

4. The Transient Options form appears. Use the scroll bar to scroll down to the bottom of this form.

5. Locate the **infotimes** text field. The text field is about 9 lines from the bottom of the form, between **compression** and **flushpoints**.



6. In the infotimes text field, enter:

   1.505u 2.0u 1.00u 1.25u

Caution

Do not use commas or any other characters to separate the values. Commas will cause a warning to appear. ADE 5.1.41 no longer supports commas.

7. At the top of the Transient Options form, click **OK**.

8. In the Choosing Analyses form, click **OK**.

9. In the Simulation window, execute **Simulation—Netlist—Recreate**.

   The Netlist Output is displayed.

10. Search for infotimes and verify the time values are the same as those entered into the infotimes text field. Simulate the circuit by clicking the **Run** icon or by executing **Simulation—Run**.

11. When the simulation has completed, go to the schematic window and execute **Design—Hierarchy—Descend Edit**, then select the amplifier symbol of the ampTest schematic. Finally, select the schematic view in the Descend form.

    The schematic view opens. It will be used in the next step.

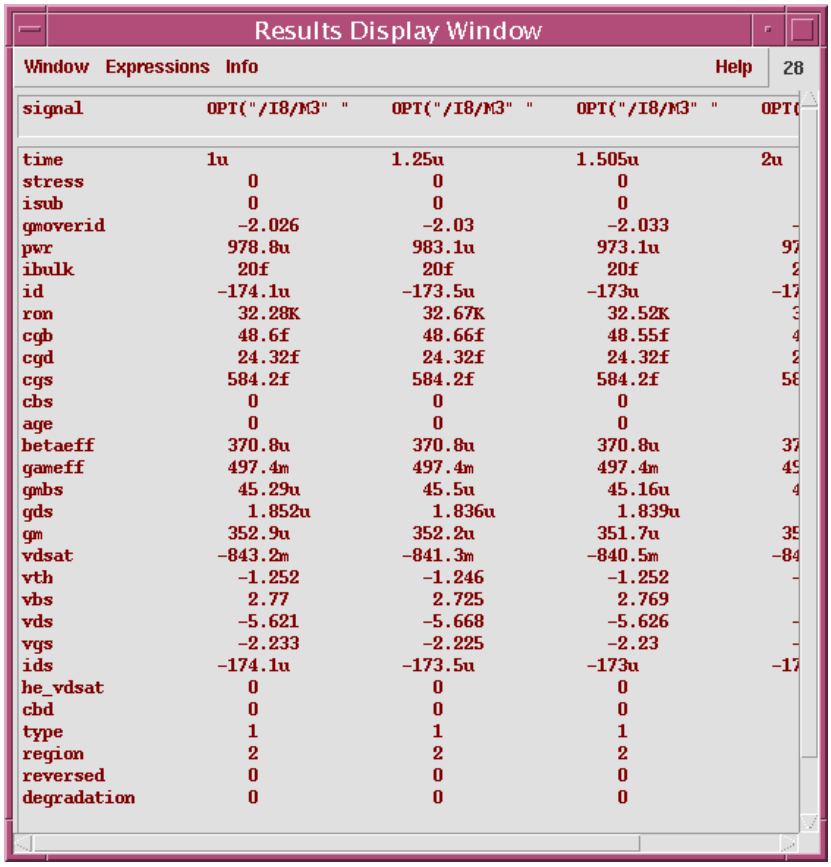
12. In the Simulation window, execute **Results—Print—Transient Operating Points**.

    A Results Display Window appears.

13. Move the cursor into the schematic window of the amplifier. Select the pmos transistor M1 in the schematic. Then select M3. Then press the **Esc** key.

    As the transistors are selected, the transient operating point information is appended to the Results Display Window.

14. In the Results Display Window, locate the values for **Ids** and **gm** for M1 and M3. Verify that **Ids** and **gm** remain nearly constant for the four sampled operating points.



signal	OPT("/I8/M3" "	OPT("/I8/M3" "	OPT("/I8/M3" "	OPT(
time	1u	1.25u	1.505u	2u
stress	0	0	0	
isub	0	0	0	
gmoverid	-2.026	-2.03	-2.033	-
pwr	978.8u	983.1u	973.1u	97
ibulk	20f	20f	20f	2
id	-174.1u	-173.5u	-173u	-17
ron	32.28K	32.67K	32.52K	3
cgb	48.6f	48.66f	48.55f	4
cgd	24.32f	24.32f	24.32f	2
cgs	584.2f	584.2f	584.2f	58
cbs	0	0	0	
age	0	0	0	
betaeff	370.8u	370.8u	370.8u	37
gameff	497.4m	497.4m	497.4m	49
gmbs	45.29u	45.5u	45.16u	4
gds	1.852u	1.836u	1.839u	
gm	352.9u	352.2u	351.7u	35
vdsat	-843.2m	-841.3m	-840.5m	-84
vth	-1.252	-1.246	-1.252	-
vbs	2.77	2.725	2.769	
vds	-5.621	-5.668	-5.626	-
vgs	-2.233	-2.225	-2.23	-
ids	-174.1u	-173.5u	-173u	-17
he_vdsat	0	0	0	
cbd	0	0	0	
type	1	1	1	
region	2	2	2	
reversed	0	0	0	
degradation	0	0	0	

Note: You can modify display format using **Expressions—Display Options**...

15. You will use your present lab setup for the next lab section. Leave all windows open.

End of Lab

Lab 3-4 Captab

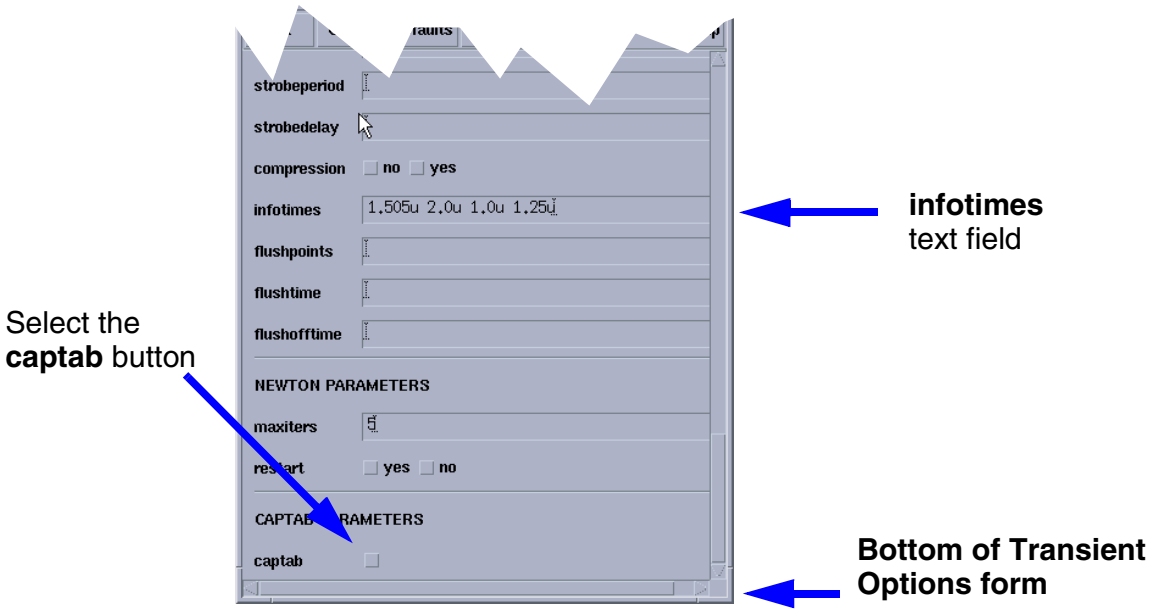
Objective: To demonstrate the Captab feature of the Virtuoso Analog Design Environment.

The Virtuoso Analog Design Environment has an option that provides a detailed tabulation of node and device capacitances during a transient simulation. This feature is selected from the Transient Options form.

Selecting the Captab Option

1. In the Simulation window, execute **Analyses—Choose**.
2. In the Choosing Analyses form, select **tran**.
3. At the bottom of the Choosing Analyses form, click **Options**.

   The Transient Options form appears.
4. Scroll down to the bottom of the form and select **captab**.



5. After the Transient Options form expands to display the captab parameters, set the following:

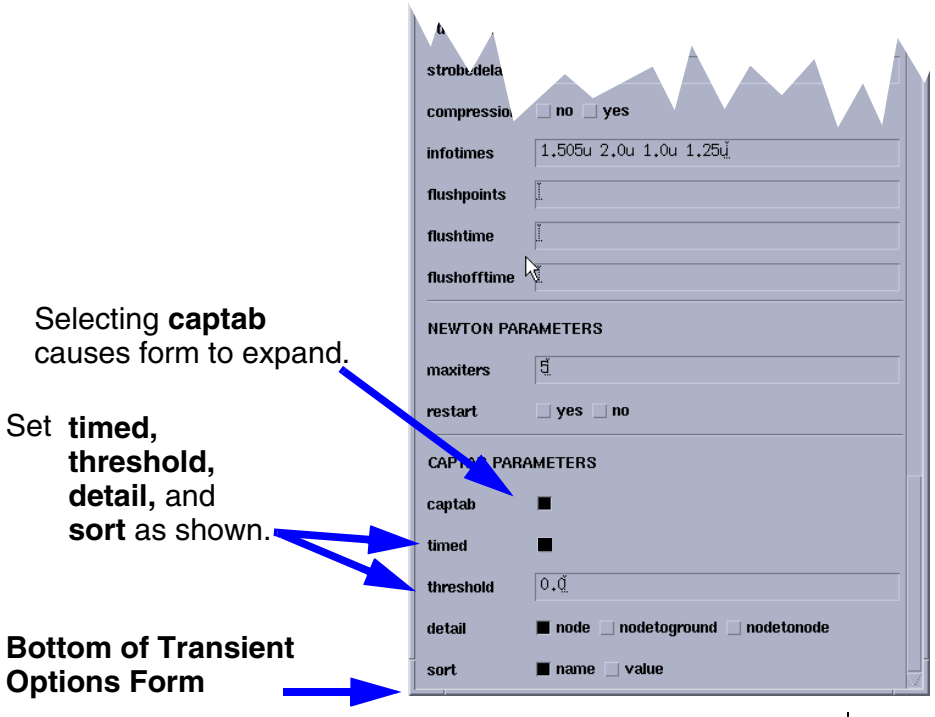
   a. Select the **timed** button to get capacitance values at infotimes; otherwise, it will show the cap values only at the final time point .

   b. Enter **0.0** in the **threshold** text field.

   c. Set **detail** to **node**.

   d. Set **sort** to **name**.

   The bottom of Transient Options form should appear as shown below.



6. At the top of the Transient Options form, click **OK**.

7. In the Choosing Analyses form, click **OK**.

8. Execute **Simulation—Netlist—Recreate** and notice the lines towards the bottom of the netlist that show how captab analysis is set up. Close the netlist after viewing.

9. In the simulation window, execute **Simulation—Netlist and Run**.

10. After the simulation runs, view the **captab** results.

11. Exit the design session. Do not save any results. Close all windows **except the CIW and the Library Manager**.