Virtuoso® Analog Design Environment User Guide

Product Version 5.1.41
September 2006

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

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

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

Contents

B
Waveform Tools in ADE

Preface

This manual describes how to use the Virtuoso® Analog Design Environment to simulate analog designs.

The preface discusses the following:

- Related Documents on page 23
- Typographic and Syntax Conventions on page 24

Related Documents

The Virtuoso® Analog Design Environment is documented in a series of online manuals. The following documents give you more information.

- *Virtuoso® Advanced Analysis Tools User Guide* gives information about Monte Carlo, optimization, and statistical analysis.
- *Analog Waveform User Guide* describes how to use the AWD waveform display tool.
- *WaveScan User Guide* provides more information about the WaveScan waveform display tool.
- *Cadence® Distributed Processing User Guide* describes how to use multiple hosts to distribute simulations between a collection of different machines.
- *Virtuoso® Analog Mixed-Signal Simulation Interface Option User Guide* gives information about how to set up and run mixed-signal simulations.
- *Virtuoso® Parasitic Simulation User Guide* describes how to analyze parasitics.
- *Waveform Calculator User Guide* gives detailed information about using the waveform calculator.
- *Virtuoso® Schematic Composer User Guide* describes connectivity and naming conventions for inherited connections and how to add and edit net expressions in a schematic or symbol cellview.
- *SpectreRF Help* describes how to use the RF option.

- *Spectre Circuit Simulator Reference* and *Spectre Circuit Simulator User Guide* describe the Virtuoso® analog circuit simulator in detail.
- *Virtuoso® UltraSim Simulator User Guide* provides detailed information about the UltraSim simulator.
- *Virtuoso® AMS Environment User Guide* provides detailed information about the AMS simulator.

Typographic and Syntax Conventions

This list describes the syntax conventions used in this manual.

`literal` Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names.

`argument (z_argument)`
Words in italics indicate user-defined arguments for which you must substitute a name or a value. (The characters before the underscore (_) in the word indicate the data types that this argument can take. Names are case sensitive. Do not type the underscore (*z_*) before your arguments.)

`[ ]` Brackets denote optional arguments.

… Three dots (...) indicate that you can repeat the previous argument. If you use them with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.

`argument`… Specify at least one, but more are possible.

`[argument]`… Specify zero or more.

`,`… A comma and three dots together indicate that if you specify more than one argument, you must separate those arguments by commas.

If a command line or SKILL expression is too long to fit inside the paragraph margins of this document, the remainder of the expression is put on the next line, indented.

When writing the code, put a backslash (\) at the end of any line that continues on to the next line.

SKILL Syntax Examples

The following examples show typical syntax characters used in SKILL.

Example 1

```
list( g_arg1 [g_arg2] ...) => l_result
```

Example 1 illustrates the following syntax characters.

`list` Plain type indicates words that you must enter literally.

`g_arg1` Words in italics indicate arguments for which you must substitute a name or a value.

`( )` Parentheses separate names of functions from their arguments.

`_` An underscore separates an argument type (left) from an argument name (right).

`[ ]` Brackets indicate that the enclosed argument is optional.

=> A right arrow points to the return values of the function. Also used in code examples in SKILL manuals.

`...` Three dots indicate that the preceding item can appear any number of times.

Example 2

```
needNCells(
s_cellType | st_userType
x_cellCount
)
=> t/nil
```

Example 2 illustrates two additional syntax characters.

`|` Vertical bars separate a choice of required options.

`/`	Slashes separate possible return values.

Form Examples

Each form shows you the system defaults:

- Filled-in buttons are the default selections.
- Filled-in values are the default values.

1

Features of the Virtuoso® Analog Design Environment

This chapter describes the features of the Virtuoso® Analog Design Environment. This is an overview. Detailed information is available in later chapters.

Consistent User Interface

The Virtuoso® design framework II environment is the foundation on which a wide range of Cadence tools is built. Using this architecture, you can go from one tool to another without tedious data conversion. The consistent user interface makes it easy to apply your knowledge of one Cadence tool to many other Cadence tools.

The design framework II environment is an open system. You can integrate third party tools and enter your own design data with industry-standard EDIF and Virtuoso® GDSII Stream formats. Simulators can be integrated using a programmable netlister and a general waveform processor.

Analog Design Entry

You enter designs into the Virtuoso® Analog Design Environment using a hierarchical schematic editor. This editor uses a set of simulation environment commands in combination with a library.

In addition to letting you enter a schematic, these commands let you place circuit variables or design equations directly on the appropriate elements of the schematic.

The equations can be any arbitrary algebraic expressions and can include popular scientific functions, such as log, exp, or cos. When you run a simulation, all expressions are automatically evaluated, and any modified circuit variables are automatically passed down through the schematic hierarchy. Because the schematic can contain both design equations and circuit topology details, you can use it to archive the most important aspects of a design. The expression capability also makes the design more general and reusable.

Design Hierarchy

In Analog Design Environment, you can start your design by building up a large circuit or system using high-level functional blocks (in the form of analog macromodels) and, as the design progresses, gradually fill in details of the blocks. When the design is finished, you can efficiently run large simulations using a mix of the high-level models and more detailed transistor-level models. You use detailed models where the highest accuracy is necessary in the simulation.

Simulators Supported

The Virtuoso® Analog Design Environment allows you to choose and work with any of the following simulators:

Annotation

Analog Design Environment lets you annotate and display DC voltages and transistor operating points directly on the schematic. You can also print out a hardcopy of any of the various outputs including annotated schematics and complex waveforms.

Interactive Simulation

Interactive circuit simulation lets you quickly enter, change, analyze, display, and manipulate simulation results. For example, after starting a circuit simulation, you can interrupt it, probe through the design hierarchy to check node voltages and currents, and then continue simulation.

Simulation Output and Analysis

The Analog Design Environment supports advanced analog/mixed-signal waveform display and post-processing tools (*WaveScan and Analog Waveform Display)*, which feature:

- Outputs overlaid from different simulations
- Multiple strip or superimposed plots
- Linear and log plots
- Smith charts
- Single or multiple Y axes
- Multiple windows
- Pan and zoom capability
- A built-in waveform calculator lets you display algebraic expressions composed of any combination of input or output voltages or currents. Such expressions can be plotted against any variable, including other algebraic expressions.
- Prepackaged waveform measurement tools are also included so you can get accurate numbers quickly. These tools let you automatically measure delay time, rise time, overshoot, settling time, slew rate, phase and gain margins, and other common analog characteristics.

To learn more about the two waveform display tools, refer to the *WaveScan User Guide* and the *Analog Waveform User Guide* respectively.

Advanced Analysis

The Virtuoso® Analog Design Environment supports advanced analysis features such as Parametric analysis, Statistical analysis, Corners analysis and the Optimizer. For details see, *Analysis Tools*.

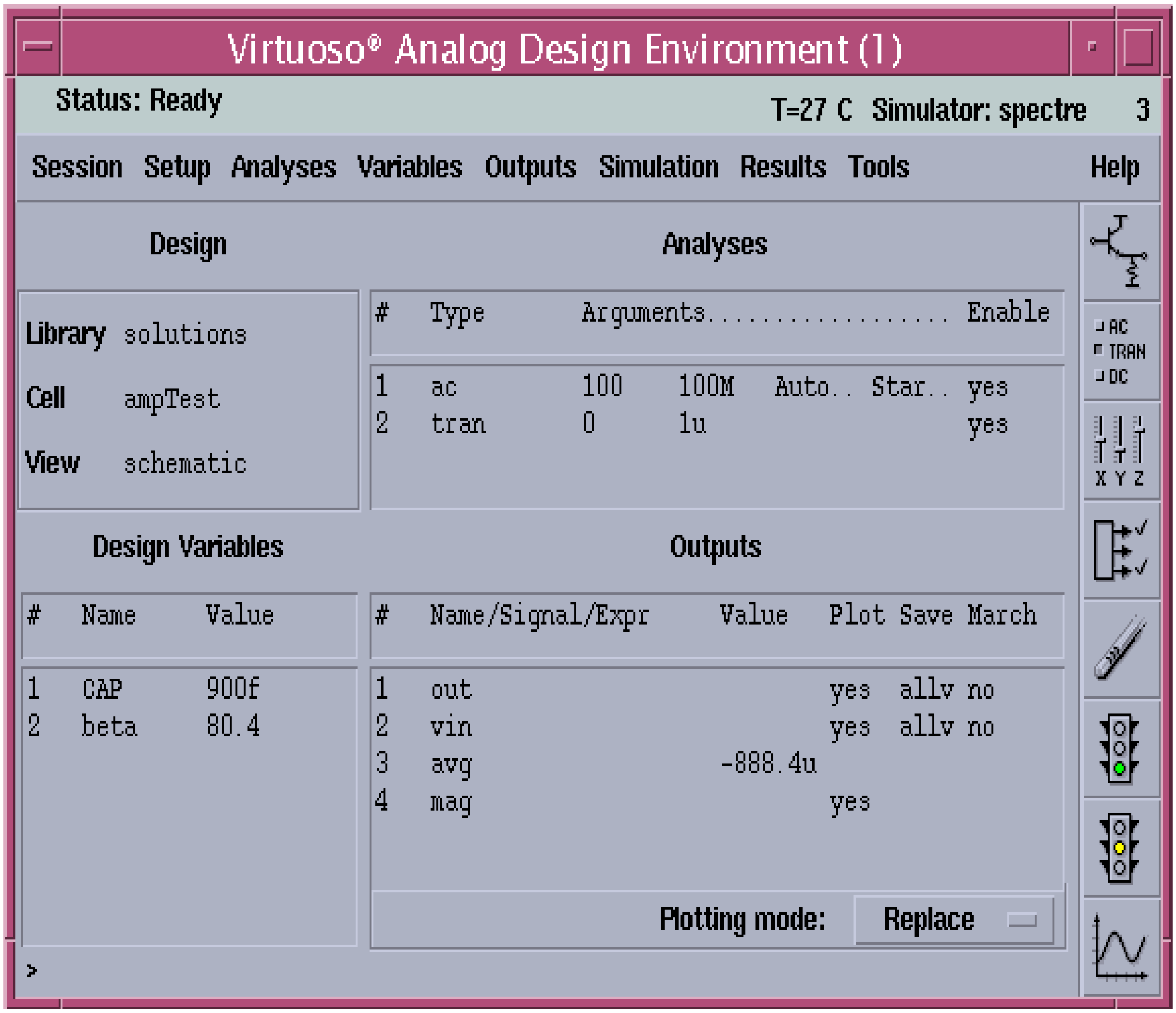
2

Environment Setup

This chapter describes the features of the Virtuoso® Analog Design Environment and tells you how to set these features during your sessions. This chapter also describes how you use Cadence simulators and third-party simulators in the Analog Design Environment.

About the Simulation Window

For help on the Simulation window menus, click on the following figure and choose a command from the pop-up menus. For help on the icons, or on any region of the form, click on the figure. Refer to the Virtuoso® *Analog Design Environment SKILL Language Reference* for instructions on customizing any of the banner menus.



Status: Ready T=27 C Simulator: spectre 3

Design

Library solutions
Cell ampTest
View schematic

Analyses

#	Type	Arguments................				Enable
1	ac	100	100M	Auto..	Star..	yes
2	tran	0	1u			yes

Design Variables

#	Name	Value
1	CAP	900f
2	beta	80.4

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	out		yes	allv	no
2	vin		yes	allv	no
3	avg	-888.4u			
4	mag		yes		

Plotting mode: Replace

Displaying the Simulation Window

There are two ways to open the Simulation window:

- From the Schematic window
- From the Command Interpreter Window (CIW)

To start the analog circuit design environment and open the Simulation window from the Schematic window,

1. Open the Schematic window.

2. Choose *Tools – Analog Environment* from the Schematic window menu.

The *Analog Environment* and *Mixed-Signal* menus are added to the main menu bar. If the Simulation window is the session base, it opens.

3. Choose *Analog Environment – Simulation* from the Schematic window menu.

If the Schematic window is the session base, the Simulation window opens and the simulation environment is initialized.

Note: When you open the Simulation window from a Schematic window, the current cellview is considered the design to be simulated.

To open the Simulation window from the CIW,

➤ Choose *Tools – Analog Environment – Simulation*.

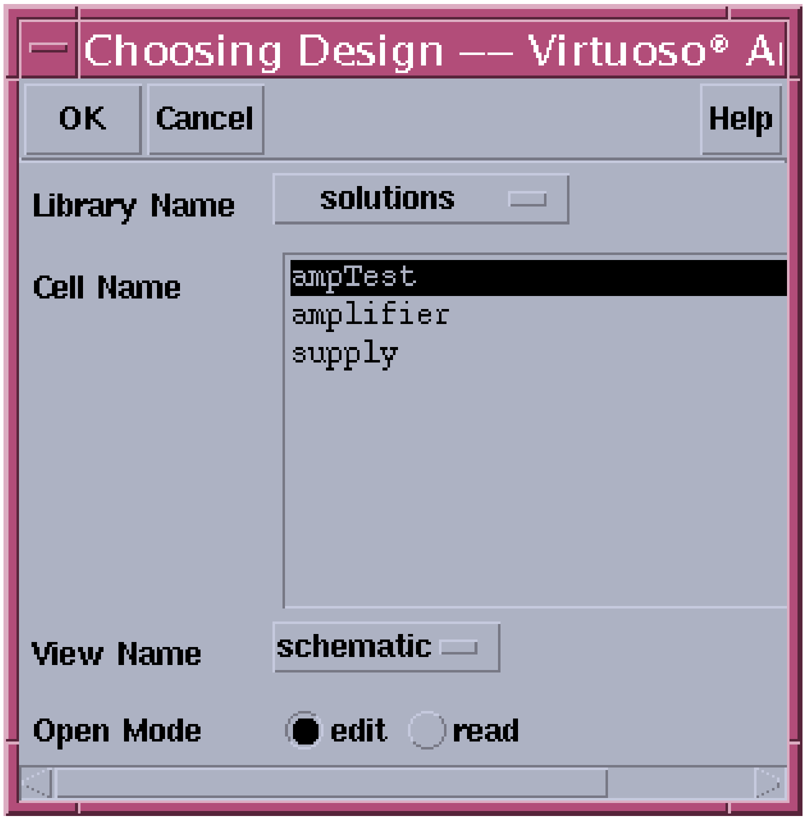
The Simulation window opens and the simulation environment is initialized.

Choosing the Design

To open a design or to select a different design,

1. In the Simulation window, choose *Setup – Design*.

The Choosing Design form appears.



Library Name: solutions

Cell Name:
- ampTest
- amplifier
- supply

View Name: schematic

Open Mode: ● edit ○ read

2. Choose a library name, cell name, and view name.
3. Choose either *edit* or *read* mode, and click *OK*.

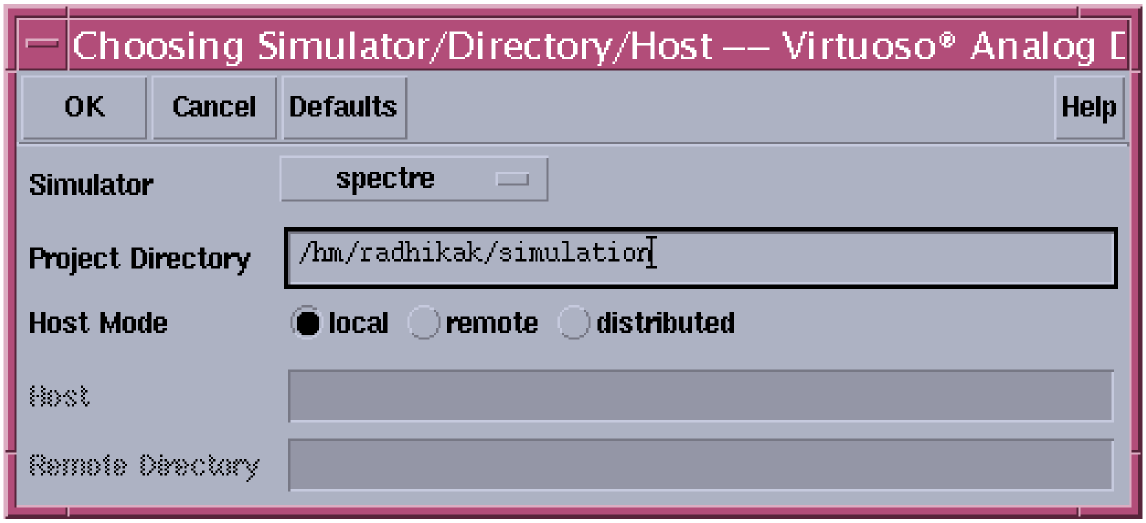
Note: To open a selected design in a different mode, you have to first re-set the session using the option, *Session – Reset*.

Choosing a Simulator

To choose a simulator,

1. In the Simulation window or the Schematic window, choose *Setup – Simulator/Directory/Host*.

The Choosing Simulator/Directory/Host form appears.



Simulator: spectre

Project Directory: /hm/radhikak/simulation

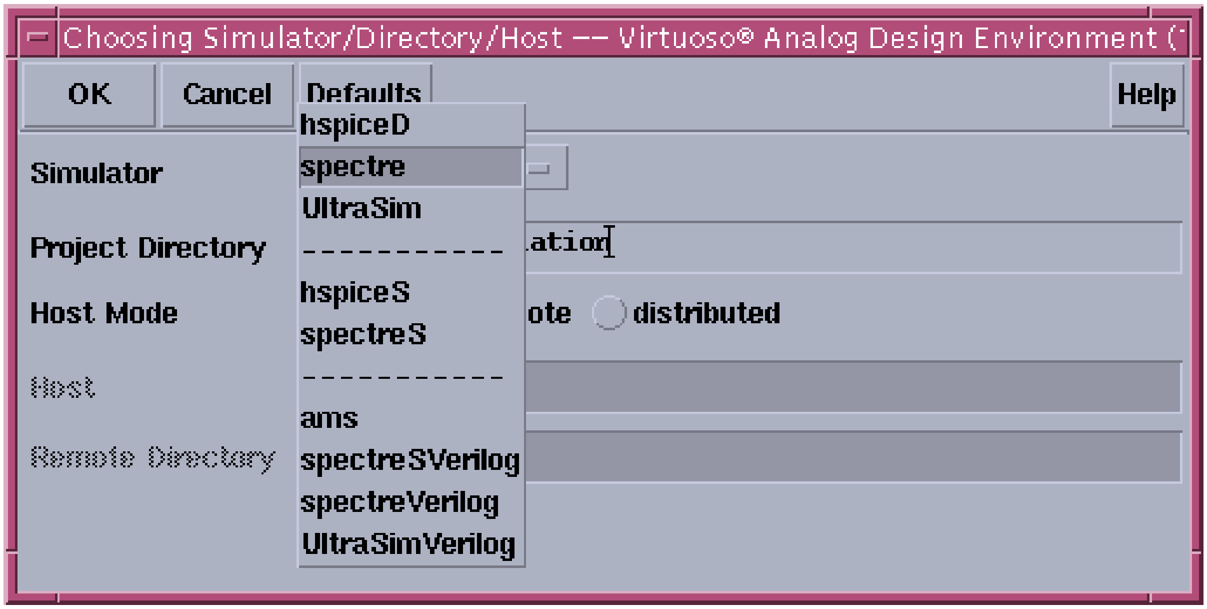
Host Mode: ● local ○ remote ○ distributed

Host

Remote Directory

For detailed information about the form, see “Choosing Simulator/Directory/Host” on page 93.

2. Choose a simulator from the Simulator cyclic list. For more information about these simulators, see “Simulator Interfaces” on page 43.



Simulator

Project Directory: ...ation

Host Mode: ...ote ○ distributed

Host

Remote Directory

3. The default *Host Mode* setting is *local*.

For information on the *remote* host mode, see the topic Setting Up a Remote Simulation on page 75.

When the *Host Mode* is *Distributed*, the *Choosing Simulator/Directory/Host* form re-displays to show the *Auto Job Submit, E-mail Notify*, *Check setup* and the *Stop setup check* buttons. For details, refer to the Setup Requirements section in Chapter 2 of the Virtuoso® *Analog Distributed Processing Option User Guide*.

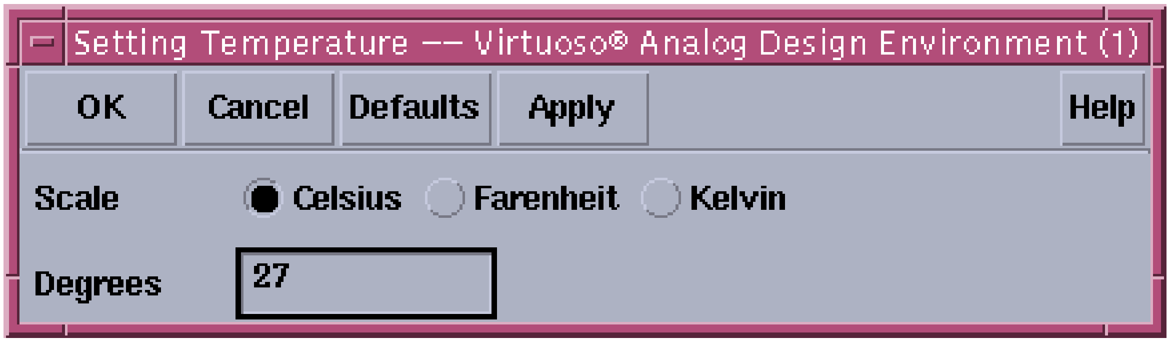
4. Check the path in *Project Directory* for simulation data, and change it if necessary.

Setting the Simulation Temperature

To set the simulation temperature,

1. In the Simulation window or the Schematic window, choose *Setup – Temperature*.

   The Setting Temperature form appears.


Setting Temperature –– Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help

Scale ● Celsius ○ Farenheit ○ Kelvin

Degrees 27

2. Choose the units you want to use for temperature.
3. Type a value in degrees, and click *OK*.

Setting the Model Path

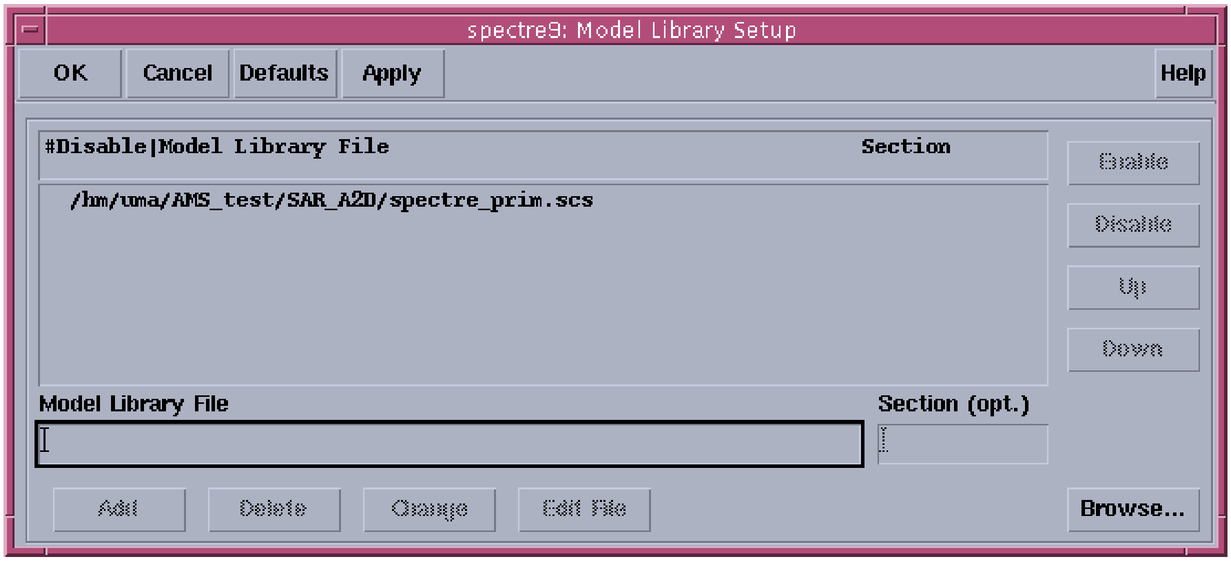
A model path is a list of directories that is searched in sequence to locate models for instances in your design. For different simulators, the model path is set differently as described in the following sections.

Setting the Model Path for Direct Simulation and AMS

To set up models for simulator interfaces that use the direct simulation approach, such as UltraSim, spectre and hspiceD.

1. In either the Simulation window or the Schematic window, choose *Setup – Model Libraries*.

The Model Library Setup form appears.



```
#Disable|Model Library File                                        Section
  /hm/uma/AMS_test/SAR_A2D/spectre_prim.scs
```

For detailed information about the form, see "Model Library Setup" on page 96.

2. Fill in the *Model File* field with the path to the model file you want to use.

   This can be the full UNIX path or the name of one or more files. The model files contain all model definitions referred to by your design and not defined within the Virtuoso® library. Unless you specify a full path, the simulator assumes that you are using the project directory specified in the form.

   Example:

   The model file for a direct interface simulation of the schematic view of the lowpass cell of the aExamples library can be found in *your_install_dir*/tools/dfII/samples/artist/models/spectre/definitions.scs

```
simulator lang=spectre

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

model pnp bjt type=pnp is=3.28e-16 va=30 bf=35 \
br=6 nc=2 ikr=100m rc=1 \
cjc=1e-12 fc=0.5 cje=0.7e-12 \
tr=200e-12 tf=65e-12 itf=0.03 vtf=7 xtf=2
```

   The models npn and pnp are referenced within the opamp schematic cell-view of the aExamples library. The device Q25 (connected to the pin inp) references the model npn with the parameter model (Model Name).

3. You may select a section from the pull-down list of the optional field, *Section*.

   This field can have one or more values. The model file can have one or more model library definitions. For each of the values specified, the model files are included with the directive to use the library definition desired.

   Example:

   For the model file

   ```
   #ifdef fast
   ... models ..
   #endif
   #ifdef slow
   ... models ...
   #endif
   ```

   the value `slow` in the *Model Sections* field causes the second part of the file to be used. For more details on modeling, see the online manual *Direct Simulation Modeling User Guide*.

You can enable or disable model files by using SKILL functions as follows:

```
asiSetEnvOptionVal(asiGetTool(’spectre) "modelFiles"
list(
   list("/usr1/models/model1.scs")
   list("/usr1/models/model2.scs")
   list("#" "/usr1/models/model3.scs")
))
```

The # sign is used to diable the `model3.scs` file.

When the *Section* field for a particular model file is defined, the netlist will contain the statement,

```
.LIB “<modelLibraryFile>” <section>
```

When the *Section* field is not defined, the netlist will contain the statement,

```
.INCLUDE “<modelLibraryFile>”
```

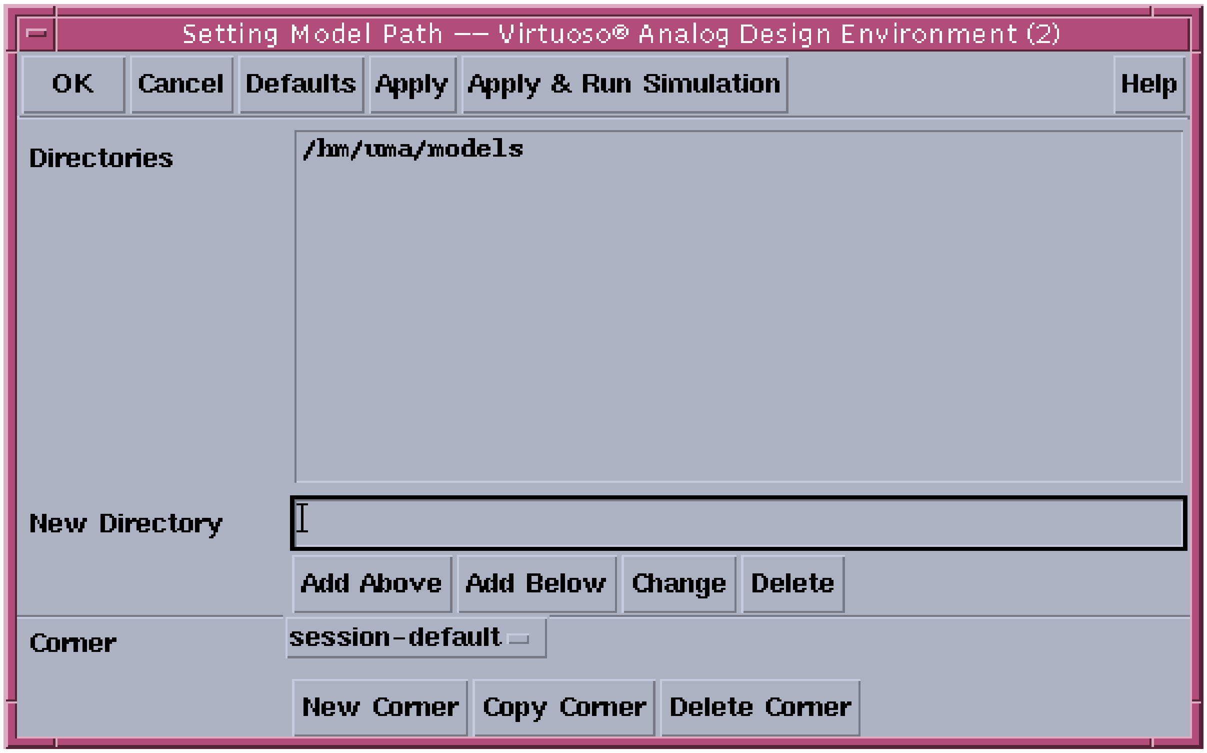
Setting the Model Path for a Socket Simulation

Use the following method to set the model path for socket simulations (for example, simulations using the spectreS and hspiceS interfaces).

Note: This method is not available for the spectre interface and other interfaces that use the direct simulation approach.

1. In either the Simulation window or the Schematic window, choose *Setup – Model Path*.

   The Setting Model Path form appears.



Directories /hm/uma/models

New Directory

Corner session-default

   For detailed information about the form, see "Setting Model Path" on page 95.

   Each list of directories is given a unique name, or corner, and provides model information for one corner of a design. You can simulate a design multiple times using different corners to test the reliability of the design with different specifications.

2. Select a corner.

   Click the corner selection box and select a corner. This changes directories to display the list associated with the corner you selected.

3. If an appropriate corner does not exist, create a new one.

   a. Click *New Corner* or *Copy Corner*.

   b. Type the new name in the Adding New Corner or Copying Corner form.

   c. Click *OK*.

   The form redisplays with the new name in *Corner*.

If you are adding a new corner, no directories are displayed in the *Directories* display area.

If you are copying a corner, the directories from the previous corner are displayed in the *Directories* display area.

4. If needed, make changes to the displayed model path.

   There are several ways to change a model path.

   a. Add a new entry.

      Select the directory that will be above or below the new entry. Type the directory in *New Directory* and click *Add Above* or *Add Below*.

   b. Change an entry.

      Double-click an entry to display it in *New Directory*. Make changes to the path and click *Change*.

   c. Delete an entry.

      Select the entry and click *Delete*.

5. If needed, rearrange the directories to provide the proper search sequence.

   a. Double-click a directory to display it in *New Directory*.

   b. Click *Delete*.

      This removes the directory from the directories list but leaves it in the *New Directory* field.

   c. Click the entry that will be above or below the one you are moving.

   d. Click *Add Above* or *Add Below* to move the directory.

6. Apply your changes.

   There are several ways to apply your changes:

- Click *OK* or *Apply* to accept the changes.

  OK accepts the change and closes the form.

  Apply accepts the change but keeps the form displayed for further changes.

- Click *Apply & Run Simulation* when you need to run multiple simulations using different model paths.

  The form remains open for you to run a subsequent simulation with another corner.

7. Click *Defaults* to set the environment variable to the current corner of the default model path.
8. To save this setting for future sessions, choose *Options – Save Defaults* from the CIW.

A file called `.mpu-selection-file` is created at the location `<project directory>/<cellName>/<simulator>` when an ADE session ends. Information about corners, the model paths defined for those corners and the currently selected corner for a design is stored in `.mpu-selection-file`. When the same design is opened again in ADE using the same simulator and the same project directory, the Setting Model Path dialog box takes default values from this file.

Choosing a User Interface Path

The analog circuit design environment provides two interface paths to the simulation environment:

- Through the Simulation window

  The Simulation window displays the main simulation environment at a glance. It is designed to display and manipulate environment variables and settings.

  Its focus is on simulation when circuit topology is fixed and simulations are run to tune design variables.

- Through the Schematic window

  Analog circuit design environment menus can be added to the Schematic window.

  This allows full access to simulation functions from the design environment.

  Its focus is on simulation during the early design phase, when the circuit topology changes often.

Either path allows you to adjust inputs and outputs, run simulations, and select data to be plotted or saved. You can select which window to open automatically at startup by specifying your default user model as described in “Setting Basic Session Defaults” on page 83.

Using the Simulation Window

In a single view, the Simulation window displays

- Simulator name and status
- Design selection

- Design variable values
- Selected analyses and their settings
- Outputs and the method of presenting results
- Simulation temperature

You can make adjustments quickly by choosing the appropriate menu selection or by double-clicking on a displayed item. Changes are immediately updated in the Simulation window segments.

If necessary, you can always display the schematic of the current design by choosing it from the *Session* menu.

Using the Schematic Window

The Virtuoso® Schematic window can be appended with simulation menus so that setup and run choices are readily available. These menus provide choices for

- Analog environment session controls
- Setup form access
- Simulation run commands
- Result disposition
- Simulation tools for evaluation and optimization
- Mixed-signal simulation access

The simulation menus do not interfere with your use of the Virtuoso schematic window. All of the menus normally provided on the window are still available.

If necessary, you can always get a look at your entire environment setup by redisplaying the Simulation window through the *Analog Environment* menu choice.

Simulator Interfaces

ADE has a variety of simulators integrated in it. The following sections describe these.

Socket Versus Direct Interfaces

Cadence has integrated simulators into the analog circuit design environment in two ways:

- Interfaces that use the direct simulation approach

  This is the preferred approach. With direct simulation, the netlist uses the syntax of the simulator you are using, without any processing to evaluate expressions. The passed parameters, design variables, functions, and so on are all resolved by the simulator. The netlist is a direct reflection of the design.

- Interfaces that use cdsSpice for processing of the netlist

  With socket interfaces, the netlist is processed by Cadence SPICE to evaluate all expressions and resolve passed parameters. You would use the socket methodology to integrate your simulator if the simulator you use cannot handle expressions or parameter passing.

Spectre Simulator

The analog circuit design environment provides the spectre and spectreS interfaces to the Spectre® analog simulator. Using the spectre interface is the preferred method. It takes advantage of the new features added to the Spectre simulator, and it uses the direct simulation approach.

The Spectre simulator is integrated into the analog circuit design environment with the Open Analog Simulation Integration Socket (OASIS).

Spectre-specific information appears in several other places in this document. For more information about the Spectre simulator, consult these topics:

- “Spectre Options” on page 297
- “Setting Up a Spectre Analysis” on page 192

For information about Spectre, read the *Spectre Circuit Simulator User Guide* and the *Spectre Circuit Simulator Reference* manual.

Note: Spectre Direct allows the user to view partial plots while the simulation is running. Click the *Plot* icon at the lower right corner of the Artist window. The *March* option is a leftover from Spectre Socket. The forms containing this option are common to all simulators therefore, this extra option has been left in place.

Running the Spectre Simulator Outside of the Virtuoso® Analog Design Environment

To run the Spectre simulator outside of the Virtuoso® Analog Design Environment but later view the results in the circuit design environment,

1. Set up the simulation in the analog circuit design environment.
2. Choose *Simulation – Netlist – Create* in the Simulation window to generate a netlist.

   The netlist file is named `netlist` and is written to the netlist directory.
3. Run the Spectre simulator with these options:

   ```
   spectre -f psfbin [-raw ../psf] [other_arguments] <cell>.scs
   ```

With the `-f psfbin` option, the simulator creates the PSF data files you need to view and manipulate the results in the analog circuit design environment. The `-raw` option determines where the file is written. With this option, the files are written to the directory

```
../psf
```

Note that the `-I` options for the spectre interface include path might be needed as well. When a simulation is run from the analog circuit design environment, the file run is created in the netlist directory. This is a shell script that can be used as well.

Viewing Standalone Results in the Virtuoso® Analog Design Environment

To view standalone Spectre simulation results in the Analog Design Environment,

1. Open the Calculator.
2. Choose *Browse*.
3. To name the project directory, choose the directory above the one containing the Spectre data.

   Do not choose the data directory itself.
4. Place the cursor over the data directory.
5. Press the middle mouse button and choose *Create ROF* from the pop-up menu.

   The system creates the data structure the analog circuit design environment needs to read the Spectre PSF data.

WaveScan does not require manual ROF creation. PSF, WSF and SST2 data formats are supported. To learn about the *WaveScan Results Browser*, see Accessing Data.

The spectreS interface uses the socket simulation approach, which involves processing of the netlist by the Cadence SPICE simulator.

If you are using the spectreS interface, consider migrating to the spectre interface.

Virtuoso UltraSim Simulator Interface

The Virtuoso® analog design environment (ADE) provides the interface to the Virtuoso® UltraSim™ simulator.

To run Virtuoso UltraSim 64-bit software,

1. Use the `-debug3264 -V` command to check your system configuration:

   ```
   $your_install_dir/tools/bin/ultrasim -debug3264 -V
   ```

   You can use the information provided by the command to verify if the 64-bit version is applicable to your platform, if the 64-bit software is installed, and whether or not it is selected.

2. Install the Virtuoso UltraSim 64-bit software to the same location as your 32-bit software.

3. Verify that all required software patches are installed by running `checkSysConf` (system configuration checking tool script). The script is located in your local installation of Cadence software:

   ```
   $your_install_dir/tools/bin/checkSysConf MMSIM6.0
   ```

   The script is also available on the SourceLinkSM online customer support system.

4. Set the `CDS_AUTO_64BIT` environment variable `{ALL|NONE|"list"|INCLUDE:"list"|EXCLUDE:"list"}` to select 64-bit executables.

   ❑ **`ALL`** invokes all applications as 64-bit.

   The list of available executables is located at:

   ```
   $your_install_dir/tools/bin/64bit
   ```

   ❑ **`NONE`** invokes all applications as 32-bit.

   ❑ **`"list"`** invokes only the executables included in the list as 64-bit.

   `“list”` is a list of case-sensitive executable names delimited by a comma (`,`), semicolon (`;`), or colon (`:`).

   ❑ **`INCLUDE:”list"`** invokes all applications in the list as 64-bit.

- ❑ **`EXCLUDE:”list"`** invokes all applications as 64-bit, except the applications contained in the list.

Note: If `CDS_AUTO_64BIT` is not set, the 32-bit executable is invoked by default.

Example

```
setenv CDS_AUTO_64BIT ultrasim
setenv CDS_AUTO_64BIT "EXCLUDE:si"
```

5. Launch the executables through the wrapper.

   All 64-bit executables are controlled by a wrapper executable. The wrapper invokes the 32-bit or 64-bit executables depending on how the `CDS_AUTO_64BIT` environment variable is set, or whether the 64-bit executable is installed. The wrapper also adjusts the paths before invoking the 32-bit or 64-bit executables. The wrapper you use to launch the executables is located at *`your_install_dir`/tools/bin.*

   Note: Do not launch the executables directly from the *`your_install_dir`/tools/bin/64bit* or *`your_install_dir`/tools/bin/32bit* directory.

   Example

   ```
   $your_install_dir/tools/bin/ultrasim
   ```

6. Start Virtuoso UltraSim 64-bit by choosing *Setup – Simulator/Directory/Host – Simulator – UltraSim* in the Simulator window.

For more information about setting Virtuoso UltraSim simulator options, refer to the *Virtuoso® UltraSim Simulator User Guide*.

Virtuoso AMS Simulator Interface

The Virtuoso® Analog Design Environment (ADE) provides a seamless integration of the Virtuoso® AMS simulator. The integration of the AMS simulator and ADE creates a design environment with the look and feel expected by the analog and mixed-signal designers who already use ADE. When using this integration, you can access designs using the same tools you currently use for pure analog and mixed signal designs.

For more information on the AMS environment, refer to the *Virtuoso® AMS Environment User Guide*.

The integration of the AMS simulator and ADE has the following features:

- Connect Rules

You can point to existing connect rules or create your own by parameterizing existing rules. The form allows you to work with multiple connect rules and auto-compiles all the built in or modified rules. For more information, see Setting Connect Rules on page 58.

- Matlab/Simulink.

  The ability to run a consimulation using Matlab/Simulink with AMS is now avaialble in ADE. You can start MATLAB® before AMS starts by setting specified waiting time and run the cosimulation with general analog simulation flow in ADE. You can also start MATLAB independently in ADE and run the cosimulation just like the EDEN flow in previous release. For more information, see Using Matlab/Simulink on page 69

- Global Signals

  You can use the Global Signals form to declare a signal that is used as an out-of-module signal reference. For more information, see Working with Global Signals in AMS on page 310.

- SimVision Integration

  You can run simulations interactively using the SimVision debugger, by changing the run mode to interactive. Cross-probing from schematics works with the SimVision integration. For more information, see Using the SimVision Debugger with AMS on page 340.

- Logfile Utility

  You can look at all the individual log files, brought up in an xterm window, or you can use the NCBrowse logfile utility that matches a particular error in a logfile back to the original source. For more information, see Viewing the Output Log for AMS on page 337.

- Error Explanation

  You can view detailed explanation of the error for AMS in the Error Explanation form. To view an error, you need to enter the error string. However, the errors displayed for AMS are the ones that are present in the log files that are created in the psf directory while a session is being run. For more information, see Viewing the Error Explanation for AMS on page 339.

- Pack-N-Go

  With this utility you can pack up the minimal needed information to reproduce an issue in ADE, the simulator or both. For more information, see Packing a Design in AMS on page 189.

- Default Disciplines

You can specify disciplines on a library, cell, cell terminal, instance, instance terminal and net from the Composer UI. You can autocreate a discrete discipline and ADE auto-compiles the discipline for you. For more information, see Default Digital Discipline Selection on page 347.

- Advanced Analyses

  Advanced Analyses such as parametric analysis, corners and optimization works with AMS.

  Note: Monte Carlo currently does not work with AMS.

- Available Analyses

  Transient and dc op analyses are available when you use the AMS simulator with ADE. AC is available in AMS only when you use spectre as the solver.

- State Files from Other Simulators

  When you use the AMS simulator with ADE, you can load and use any state file that ADE has saved, regardless of the simulator ADE was running when the state file was saved. Other simulators include Virtuoso® Spectre, Virtuoso® UltraSim, spectreVerilog and UltraSimVerilog.

- Outputs

  Use the ADE *Output* options to *Save All Signals* or to *Select Specific Signals*. You cannot save AC currents.

- Single Button *Netlist and Run*

  Netlist files must be compiled and elaborated before they can be simulated. When you press *Netlist and Run* a sequence of tools is invoked including the simulator.

  - ❑ Schematics are translated to Verilog-AMS netlists
  - ❑ Netlists are compiled
  - ❑ Elaboration runs
  - ❑ Simulation runs

  Each tool produces a separate log file. When any tool fails, the CIW displays a failure message. Look in the tool’s log file for descriptive error messages. For details, refer to the section Viewing the Output Log for AMS on page 337.

- Full Support for ADE Display Tools

  The integration of the AMS simulator and ADE makes the full set of ADE tools available. In particular, you can examine waveforms with the WaveScan and AWD display tools and take advantage of their superior performance for large mixed-signal designs as well as their analog-centric capabilities. You can also bring up SimVision as a simple waveform tool after simulation as well. You can back annotate DC and transient operating point information to the schematic. You can use ADE data access features such as the Calculator and Results Browser, and the ADE Direct Plot form.

- OCEAN

  Ocean provides full support for the AMS simulator including several new commands such as `connectRules()`. For details of the commands, refer to the *OCEAN Reference*.

- Distributed and Remote Simulations

  Use network mode for distributed AMS simulations.

- Visual Display of Signal Domains

  In the schematic window, you can now highlight analog nets and digital nets in different colors, with indicators showing the location of automatically inserted connect modules. Visualization of a mixed-signal design can help you tune the design for desired characteristics.

- Display Partition

  The Display Partition capability of the AMS environment and simulator in ADE is similar to the display partition capability used in verimix.

To see some frequently asked questions about AMS-in-ADE, choose *Session – FAQ* from the ADE window.

Differences to Expect When Using AMS in ADE

For ADE Users

Some significant differences introduced with the AMS integration and ADE are the following.

- AMS brings a cell-based netlister to ADE. The cell-based netlister adds increased speed, flexibility and capacity to ADE. It maintains netlisting compatibility between ADE and AMS. For more information on the AMS Netlister, refer to the *Virtuoso® AMS Environment User Guide.*

- Both the *Netlist* command and the *Netlist and Run* command can potentially call several tools to

  - ❑ Compile updated text views

- ❑ Netlist updated cellviews
- ❑ As necessary, call several additional tools
  - ❍ *ncvlog* or *ncvhdl* to compile
  - ❍ *ncelab* to elaborate
  - ❍ *ncsim* to simulate

  Messages in the CIW indicate which tool is running. Each tool writes its own log file.

- ■ The *NCBrowse* utility helps pinpoint issues in the logfiles created during compilation, elaboration and simulation. Use *Simulation – Output Log* to activate the *NCBrowse* utility.
- ■ By default ADE does not save outputs. In the ADE Simulation window, do one of the following
  - ❑ Select *Outputs – Save All* to save all output signals.
  - ❑ Select *Outputs – To Be Saved – Select On Schematic* and select specific output signals.
- ■ You cannot directly use the following in designs:
  - ❑ Parameters declared in model files. This is due to language boundary issues in Spice and Spectre.
  - ❑ Parameters defined in definition files. This is due to language boundary issues in VerilogAMS.
- ■ You can specify a text block as the top level in your configuration as follows:

  a. Compile your top-level module by hand, for example, by running this command:

  ```
  ncvlog -use5x -ams -view verilogams textTop.vams
  ```

  This creates the `myLib/textTop/verilogams` directory as follows:

  ```
  master.tag       pc.db          verilog.vams
  ```

  b. From the Library Manager, open the cell for editing, make a change and save the view. This adds the following types of files:

  ```
  myLib/textTop/verilogams/verilogAMS.cdb
  myLib/textTop/symbol
  myLib/textTop/prop.xx
  ```

  c. In the Hierarchy Editor, create a config view with this text as top.

  d. Open ADE, and set the schematic to that config.

Note: Do not turn the ncelab `-update` flag off.

For AMS Users

- The ADE default mechanism is used. This is includes ADE state files and the `.cdsenv` file, which provides all ADE defaults. In the AMS Environment, `ams.env` is the default mechanism.
- To use the `hdl.var` file, select *Simulation – Options – Compiler – hdl.var*.

Specifying Results Directory

Before running the simulation, you can specify results directory to save the simulation data using the variable AMS_RESULTS_DIR. When you specify this variable, all the files and scripts present in or read from the psf directory will be saved/read from directory specified through AMS_RESULTS_DIR.

This variable can be set using one of the following methods:

- When you set the simulation directory in ADE using, *Setup – Simulatior/Directory/Host form,* the variable will by default be set to `<user_simulator_dir>/<cell>/<simulator>/<view>/psf` and all the data will be stored in this location.
- You can set a UNIX environment variable `AMS_RESULTS_DIR` at the unix prompt. When the variable is set on unix prompt, the variable will be identified, honoured and the simulation data will be stored in `$AMS_RESULTS_DIR/psf`.
- You can also set this variable in the OCEAN script. You would required to set it through OCEAN command `resultsDir(“path_for_results_to_store”)`. This will store the simulation data to `path_for_results_to_store/psf`.

cds_alias

cds_alias is a simple cell which is defined in the library `"basic"`. This cell is required only if your design contains the cds_alias instances. If the design that you are using contains `cds_alias` and you have not defined a library "basic" in your cds.lib, then a default cds_alias cell is created under the top design.

For example, if the cell, cds_alias is in library `"basic"`, you need not do anything. It will be compiled under implicit_tmp_dir/basic/cds_alias/functional. However, if the library `"basic"` is not present in cds.lib, AMS-ADE will itself compile it in `implicit_tmp_dir/<design_lib>/cds_alias/functional`.

More Information on the AMS Simulator

- ❑ The *Virtuoso® AMS Simulator User Guide*
- ❑ The *Virtuoso® AMS Environment User Guide*
- ❑ The *Virtuoso® Mixed-Signal Circuit Design Environment User Guide*
- ❑ The *Cadence Verilog-AMS Language Reference*
- ❑ The *Cadence Hierarchy Editor User Guide*
- ❑ The *Cadence Application Infrastructure User Guide*
- ❑ The *NC Verilog Simulator Help*
- ❑ The *NC VHDL Simulator Help*
- ❑ The *SimVision User Guide*
- ❑ The *Spectre Circuit Simulator Reference*
- ❑ The *Spectre Circuit Simulator User Guide*

Mixed-Signal Simulators

The following mixed-signal simulators are also provided:

- UltraSimVerilog
- spectreVerilog
- spectreSVerilog

For details about UltraSimVerilog, see Chapter 12, “UltraSimVerilog,”. For details about spectreVerilog and spectreSVerilog, see *Virtuoso® Analog Mixed-Signal Simulation Interface Option User Guide*.

Hspice Direct Interface

The *Analog Design Environment* (ADE) contains a direct integration of the *Hspice* simulator. ADE’s *Hspice* integration (*HspiceS*) used to be based on the socket methodology, which required edit permissions to the schematic being simulated, and caused usage issues such as subcircuit name mapping. There are several advantages of the direct simulator integration approach over the socket simulator integration approach, namely:

- Improved Performance in Netlisting

- Better Readability of Netlists
- Read-only Designs can be Simulated, Provided they are Extracted
- Advanced Evaluation of Operators

For more information, see Appendix 11, “Hspice Direct Support,”.

HSPICE Socket Interface

The HSPICE simulator is integrated into the analog circuit design environment with the Open Analog Simulation Integration Socket (OASIS).

For more information about the HSPICE simulator, see “HSPICE Socket Options” on page 322.

Running the HSPICE Simulator Outside of the Virtuoso® Analog Design Environment

To run the HSPICE simulator outside of the Analog Design Environment,

1. Set up the simulation in the Virtuoso® Analog Design Environment.

   Be sure to set correctly the HSPICE options that are specific to the Virtuoso® Analog Design Environment. The settings depend on where you want to view the results.

2. Choose *Simulation – Netlist – Create Final* to generate a netlist.

   The netlist file is named `hspiceFinal` and is written to the netlist directory.

Viewing Standalone Results in the Virtuoso® Analog Design Environment

To view standalone HSPICE simulation results in the analog circuit design environment, set the following HSPICE options when you run the simulation:

- ARTIST=2
- INGOLD=2
- PSF=2

To view the results,

1. Open the Calculator.
2. Click *Browse*.

3. To name the project directory, choose the directory above the one containing the HSPICE data.

   Do not choose the data directory itself.

4. Place the cursor over the data directory.
5. Press the middle mouse button and choose *Create ROF* in the pop-up menu.

   The system creates the data structure the analog design environment needs to read the HSPICE PSF data.

Note: WaveScan does not require manual ROF creation. PSF, WSF and SST2 data formats are supported. To learn about the *WaveScan Results Browser*, see *Accessing Data*.

Cadence SPICE Simulator

The Cadence® SPICE simulator plays an important role for socket interfaces such as spectreS and cdsSpice. This simulator is not in any way involved in the direct simulation approach that is used by the spectre interface. The Cadence SPICE simulator is an enhanced version of the University of California at Berkeley SPICE. The Cadence SPICE simulator has a modified architecture that allows efficient operation in various interactive modes, such as simulation stop and restart or quick change and resimulate.

The Cadence SPICE simulator also uses advanced convergence algorithms that are automatically started when there are convergence difficulties in a problem circuit. Once the convergence is reached, the solution can be stored and restored for future simulations.

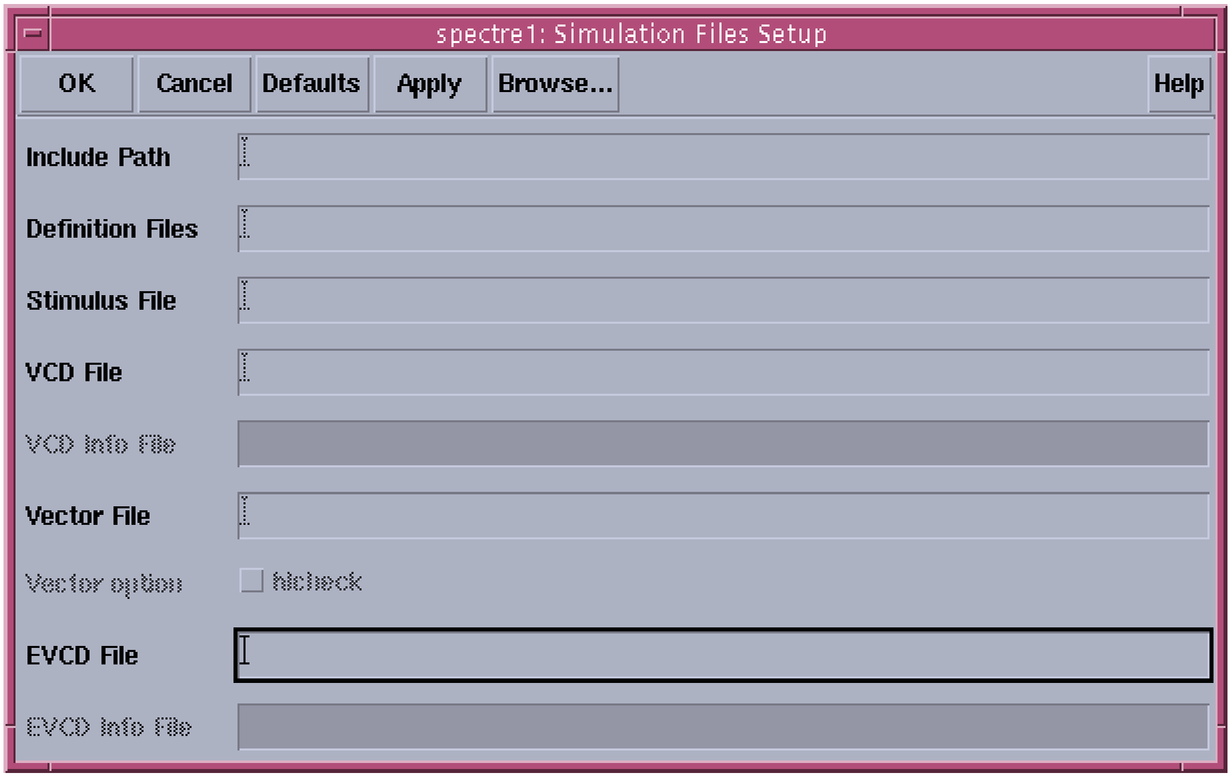
For information about Cadence SPICE device models and analysis modes that are available in the analog circuit design environment, see the *Cadence SPICE Reference Manual*.

Setting Up Simulation Files

Before you run a simulation, you must set up the simulation files.

1. Choose *Setup – Simulation Files*.

The Simulation Files Setup form appears.



2. In the *Include Path* field, type the relative path to the simulation files.

   The simulator resolves a relative filename by first looking in the netlist directory from where the simulator is run.

   Note: A file name starting with a `.` symbol is also resolved by first looking in the `netlist` directory, then in each of the directories specified by the include path, from left to right. The `.` does not mean the current directory.

3. In the *Definition Files* field, type the full UNIX path or the name of one or more files. A definitions file contains function definitions and definitions of parameters that are not displayed in the *Design Variables* section of the simulation window.

   Example:

   You can find the definitions file for a Spectre simulation of the schematic view of the `lowpass` cell of the `aExamples` library in *`your_install_dir`*`/tools/dfII/samples/artist/models/spectre/definitions.scs`:

   ```
   simulator lang=spectre
   parameters PiRho=2500 PbRho=200
   ```

```
 function Rpb(l,w)=(PbRho*l/w)
 function Rpi(l,w)=(PiRho*l/w)
```

The parameters `PiRho` and `PbRho` are referenced by included models and are not referenced from any part of the design in the Cadence library (`lowpass` or `opamp`).

4. In the *Stimulus File* field, type the full path to the directory where the stimulus file resides.
5. Type the VCD file information into the *VCD File* and *VCD Info File* fields.
6. Type the digital vector file name into the *Vector File* field. For spectre, you may also enable or disable the Vector *hlcheck* option.
7. Type the EVCD file information into the *VCD File* and *VCD Info File* fields.
8. Click *OK*.

You can use the *Browse* button to locate a file or path for any of the selected fields on the form. The *Browse* button opens a general purpose file browser. In case of fields that require only one path/file, the new path/file selected in the browser simply replaces the current value in the field. However, in case of fields that accept multiple file names, the browser works by simply appending to whatever is already specified in the field. If the required path/file has already been specified in the field earlier, then it is not appended again.

For more information about VCD and EVCD files, see Chapter 11 of the *Virtuoso® UltraSim Simulator User Guide*.

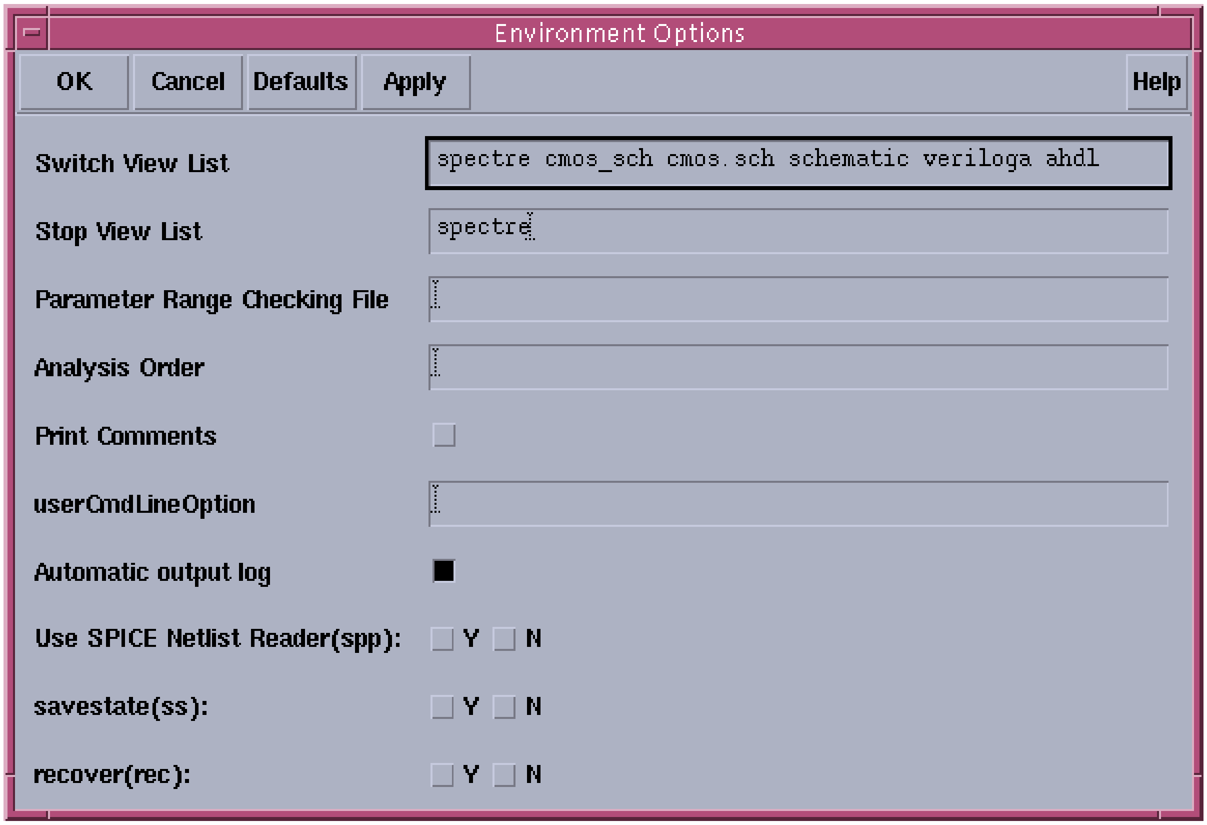
Setting Simulation Environment Options

Setting Simulation Environment Options for Direct Simulation

To open the Environment Options form

1. In either the Simulation window or the Schematic window, choose *Setup – Environment*.

The Environment Options form appears.



Switch View List	spectre cmos_sch cmos.sch schematic veriloga ahdl
Stop View List	spectre
Parameter Range Checking File	
Analysis Order	
Print Comments	□
userCmdLineOption	
Automatic output log	■
Use SPICE Netlist Reader(spp):	□ Y □ N
savestate(ss):	□ Y □ N
recover(rec):	□ Y □ N

For detailed information about the form, see “Environment Options” on page 98.

The display varies depending on which simulator you are using and whether you are using a config view for the design. Instance-based view switching is supported only for purely analog designs, not for mixed-signal designs.

2. Check that the path to the parameter range-checking file is correct. For the Spectre simulator, this file contains the parameter range limits. You do not need to enter the full path for the file if the file is in the directory specified in the include path on the Model Setup form.

   Note: A period (.) in a UNIX path specification is interpreted relative to the directory from which you started the analog circuit design environment.

3. (Optional) If you are not using a config view, check and set the options for view switching to control how the system netlists hierarchical designs.

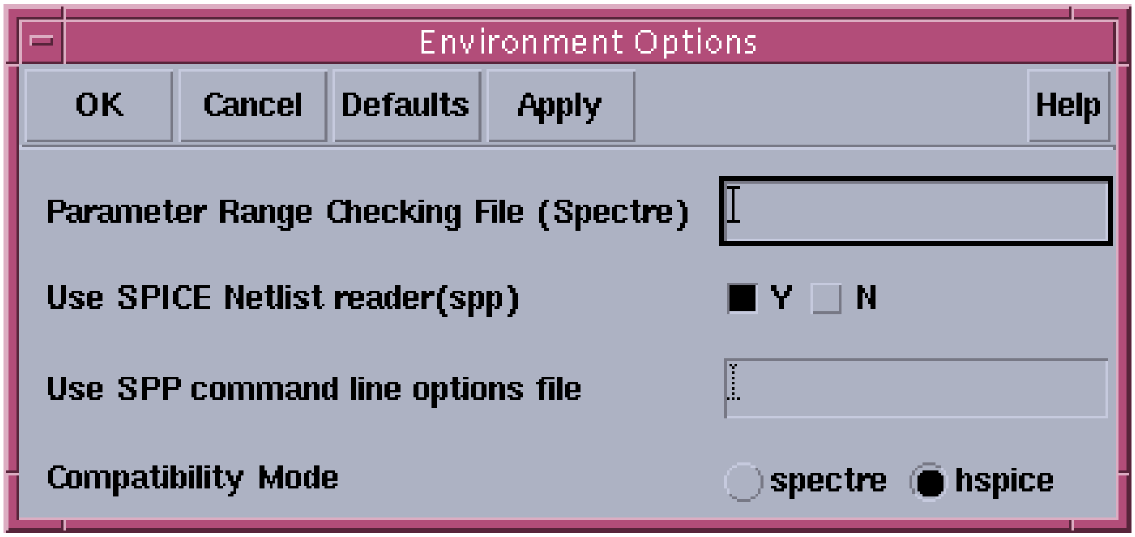
4. Set other options as needed, and click *OK*.

Setting Environment Options for AMS

To set the environment options,

1. In the Simulation window, choose *Setup – Environment*.

   The Environment Options form appears.



Parameter Range Checking File (Spectre)

Use SPICE Netlist reader(spp) ■ Y □ N

Use SPP command line options file

Compatibility Mode ○ spectre ● hspice

   For detailed information about the form, see “Environment Options” on page 98.

2. Specify a parameter range checking file.
3. Set the remaining options as needed.
4. Click *OK.*

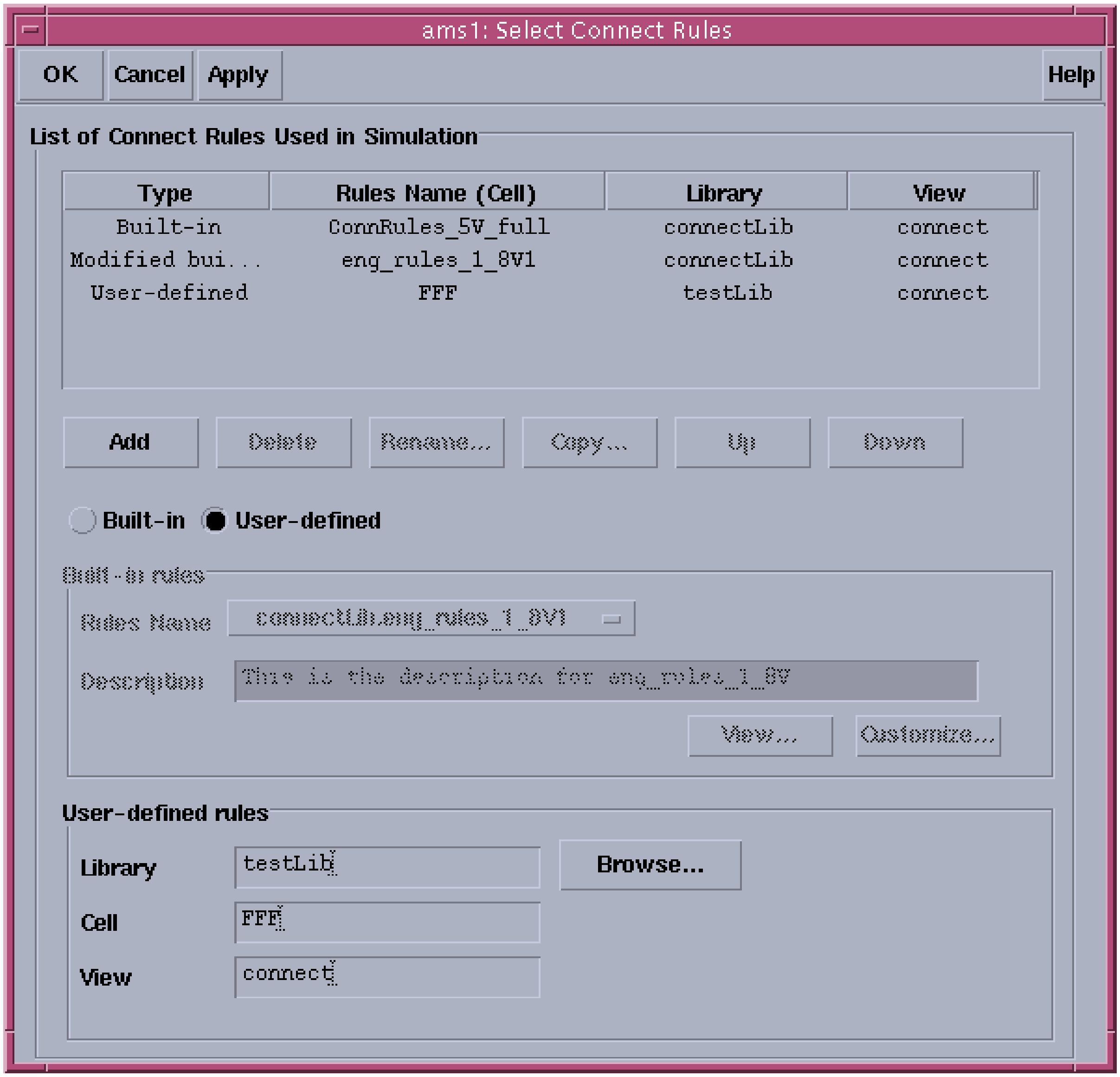
Setting Connect Rules

A connect rule specification is used is used to insert selected connect modules in mixed ports. A connect module is a module that is automatically or manually inserted to connect the continuous and discrete disciplines (mixed-nets) of the design hierarchy together. For more information, see the *Using Connect Modules* section in the *Mixed-Signal Aspects of Verilog-AMS* chapter of the *Cadence Verilog-AMS Language Reference*.

To set connect rules,

1. In the Simulation window, choose *Setup – Connect Rules.*

The Select Connect Rules form appears.



List of Connect Rules Used in Simulation

Type	Rules Name (Cell)	Library	View
Built-in	ConnRules_5V_full	connectLib	connect
Modified bui...	eng_rules_1_8V1	connectLib	connect
User-defined	FFF	testLib	connect

○ Built-in ● User-defined

Built-in rules

Rules Name	connectLib.eng_rules_1_8V1
Description	This is the description for eng_rules_1_8V

User-defined rules

Library	testLib
Cell	FFF
View	connect

The form shows a list of connect rules used in the simulation and that need to be passed to the ncelab command line. Each row has the following details:

- ❑ *Type*—Displays the type of the connect rule as `Built-in`, `User-defined` or `Modified built-in`.
- ❑ *Rules Name*—Displays the name of the rule, which would be a cell name.
- ❑ *Library*—Displays the name of the library the rule is in.
- ❑ *View*—Displays the view name for the rules.

If one of these rules is selected and it is of the type `Built-in` or `Modified built-in`, the *Built-in* option button is selected and the *Built-in rules* group box is enabled. If the type of the selected rule is `User-defined`, the *User-defined* option button is selected and the *User-defined rules* group box enabled. The built-in rules in the UI come from the connectRules.il File.

2. You can add built-in or user-defined connect rules to this list.

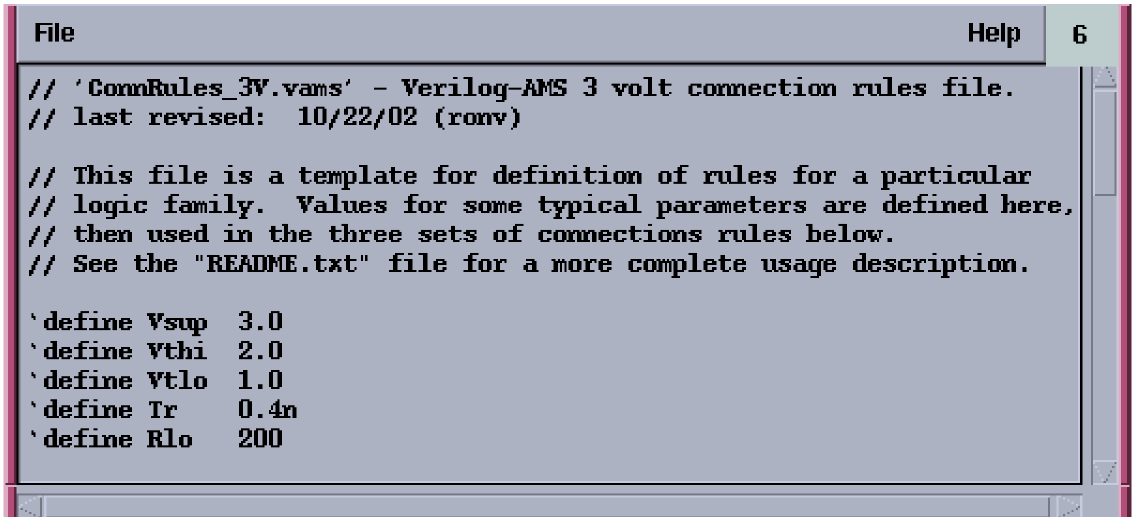
❑ To add a built-in connect rules,

a. Select the *Built-in* option button. This enables the items in the *Built-in rules* group box.

b. Select a rule from the *Rules Names* pull-down list.

Its description appears in the non-editable *Description* field below it.

c. If you want to see the rule, click the *View* button. This brings up the connect rules file.



```
// 'ConnRules_3V.vams' - Verilog-AMS 3 volt connection rules file.
// last revised:  10/22/02 (ronv)

// This file is a template for definition of rules for a particular
// logic family.  Values for some typical parameters are defined here,
// then used in the three sets of connections rules below.
// See the "README.txt" file for a more complete usage description.

`define Vsup  3.0
`define Vthi  2.0
`define Vtlo  1.0
`define Tr    0.4n
`define Rlo   200
```

d. If you want to customize the rule, click the *Customize* button.

This brings up the Customize Built-in Rules form using which you may customize the rule. For more details, see Customizing Built-in Rules on page 65.

Note: When you customize a built-in rule, its type appears as *Modified built-in* and its name is modified with a post-fixed number and added as a separate item in the list of rules. For example, a built-in rule `ConnRules_3V_mid`, when customized, would be automatically renamed as `ConnRules_3V_mid1`.

e. Click the *Add* button below the table to add the rule to the list.

- ❑ To add user-defined connect rules,

    a. Select the *User-defined* option button. This enables the fields in the *User-defined rules* group box.

    b. Select a rule from a lib:cell:view structure by using the *Browse* button, which brings up the Library Manager.

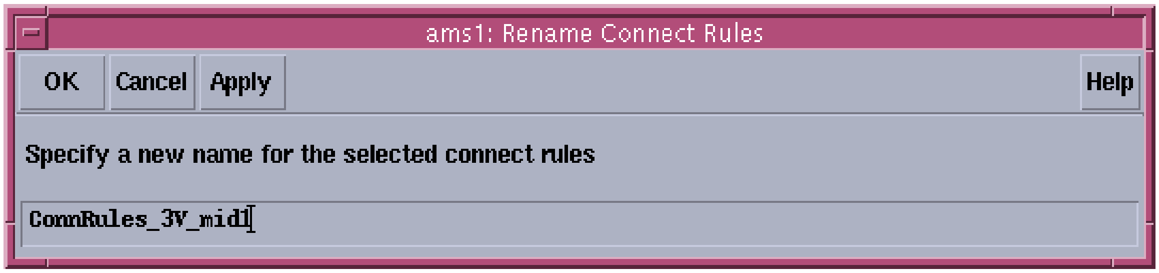
    c. Click the *Add* button below the table to add the rule to the list.

3. To rename a rule,

a. Select a rule from the table.

b. Click the *Rename* button.

The Rename Connect Rules pop-up box appears.



Specify a new name for the selected connect rules

ConnRules_3V_mid1

c. Specify a unique name for the selected connect rule and click *OK* or *Apply*.

If a rule by the same name exists in the Connect Rules table, an error message appears and the pop-up remains open.

The list of rules in the Select Connect Rules from also shows the modified connect rule name.

4. To delete a rule,

a. Select one or more rules from the table.

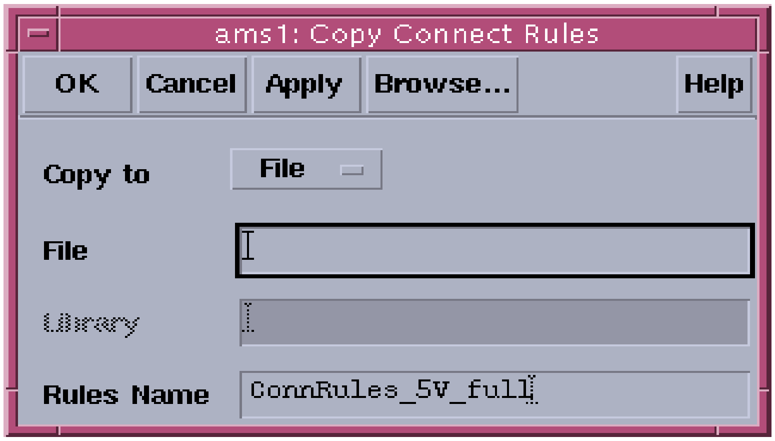
b. Click the *Delete* button.

5. To copy a rule,

a. Select a rule.

b. Click the *Copy* button.

The Copy Connect Rules form appears.



Copy to: File

File:

Library:

Rules Name: ConnRules_5V_full

c. Select either *Library* or *File* in the *Copy to* pull-down box to indicate where you want the rule copied to.

   If your choice is *Library*, the *Library* and *Rules Name* fields are enabled and you need to specify the relevant values in them. You may either type in these values or select them using the *Browse* button. If your choice is *File*, the *File* field is enabled and you need to specify a filename for the rule to be copied into. You can specify a filename indicating the path. If you specify a filename without a path, it implies that it is in the current working directory. If you specify a name for a file on which you do not have read permission, you will see an error message saying so.

   The Browse button brings up the library browser if you select *Library* and the file browser if you select *File*.

d. Click *OK*.

   The new copied rule appears in the connect rules table. This is a convenient way to copy your modified connect rule to a permanent location in your library. The connect rules will be saved in your state files, but you may want to save it to a permanent library as well.

6. To change the sequence of rules in the table,

   a. Select a rule.

   b. Click the *Up* or *Down* buttons.

   Note: All the selected rules are auto-compiled. They are passed to the ncelab command line. If two rows have a matching statement, ncelab uses the first of these rows. If two rules in a particular row have a matching statement, the last of these statements is used.

7. Click *OK* to save the settings in the current session.

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Connect Modules* option in the Saving State form and save the current ADE state. You can later load the state using the Loading State form with the *Connect Modules* option selected. For more information, see Saving and Restoring the Simulation Setup on page 77.

connectRules.il File

All Cadence-supplied connect rules are built-in. The parser `genConnRuleFile.pl` automatically compiles built-in and customized built-in connect rules and stores them in the `connectRules.il` file. By default, this file exists in the `connectLib` library in the `dfII/bin` stream.

ADE searches for the `connectRules.il` file in this order:

- All libraries defined in the `cds.lib` file
- Current work directory
- Home directory
- `$CDS_HIER/share/cdssetup/ams/`
- Path specified by the cdsenv variable `connectRulesPath`

If multiple `connectRules.il` files are located, their contents are concatenated and shown in the Select Connect Rules form.

1. To get the parser to auto-compile your user-defined connect rules, run the parser by using the following syntax:

   ```
   genConnRulesFile [-help] [-destpath <destination path>] -lib <lib1> <file1>
   [file2 ...] [-lib <lib2> <file3> [file4 ...]]
   ```

   where

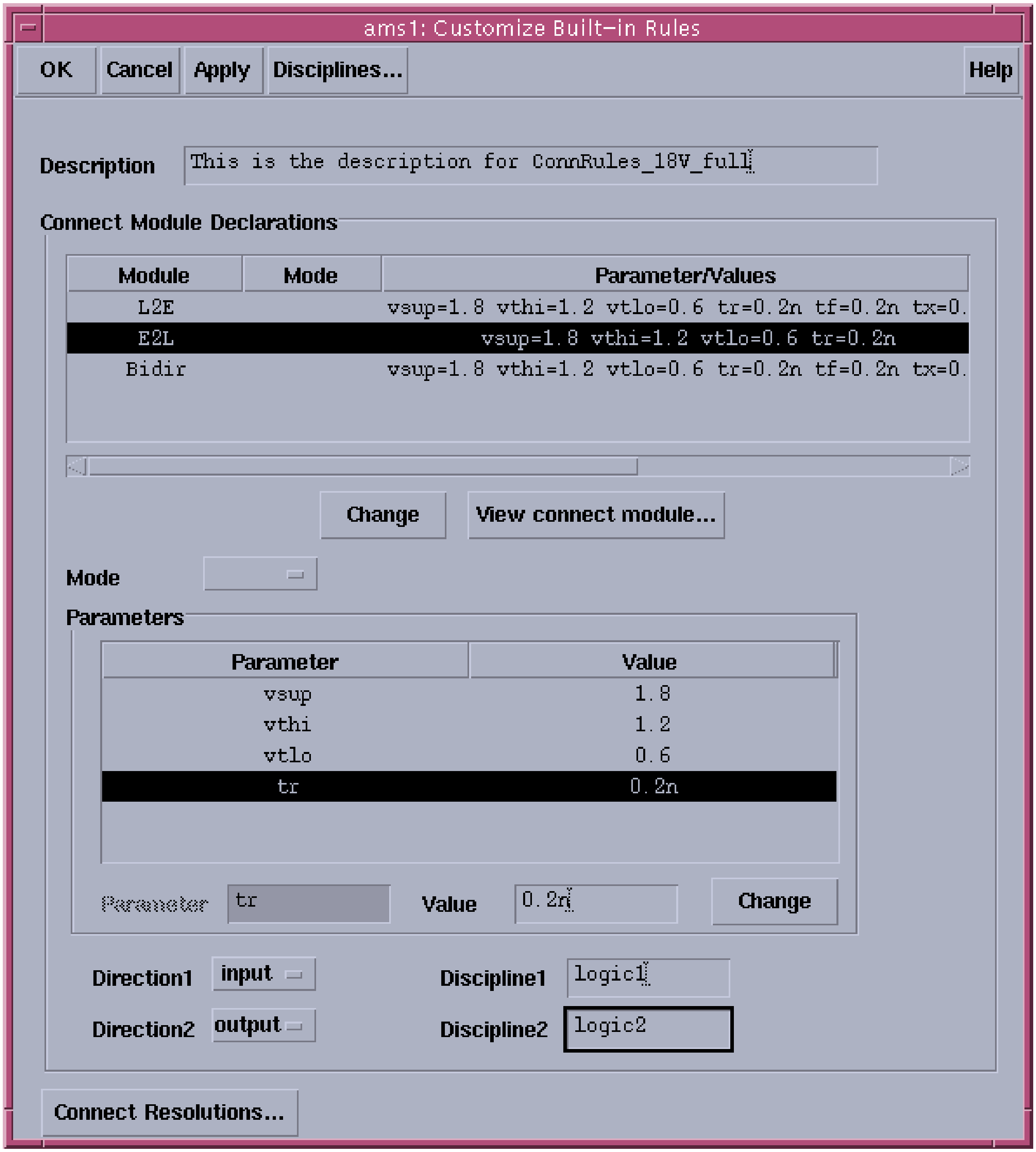
   - `-help` prints the help message for the parser.
   - `-destpath` *`<destination path>`* specifies the directory to hold the `connectRules.il` file.
   - `-lib` *`<lib1>`* *`<file1>`* [*`file2`* ...] specifies the libraries that have the connect rule files that need be parsed.

2. When the parser is run, the connect rules in the specified files are included in the `connectRules.il` file by running the parser.

3. When you open the Select Connect Rules form, the selected connect rules appear as built-in rules along with the other built-in rules in the design and are auto-compiled as usual.

Customizing Built-in Rules

This form appears when you click the *Customize* button in the Select Connect Rules form.



Description: This is the description for ConnRules_18V_full

Connect Module Declarations

Module	Mode	Parameter/Values
L2E		vsup=1.8 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n tx=0.
E2L		vsup=1.8 vthi=1.2 vtlo=0.6 tr=0.2n
Bidir		vsup=1.8 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n tx=0.

Mode

Parameters

Parameter	Value
vsup	1.8
vthi	1.2
vtlo	0.6
tr	0.2n

Parameter: tr   Value: 0.2n

Direction1: input   Discipline1: logic1

Direction2: output   Discipline2: logic2

You use this form as follows:

1. Specify a description for the rule in the *Description* field. This replaces the contents of the corresponding non-editable field in the Select Connect Rules form.
2. The *Connect Module Declarations* table displays the following information about the modules in the selected connect rule:

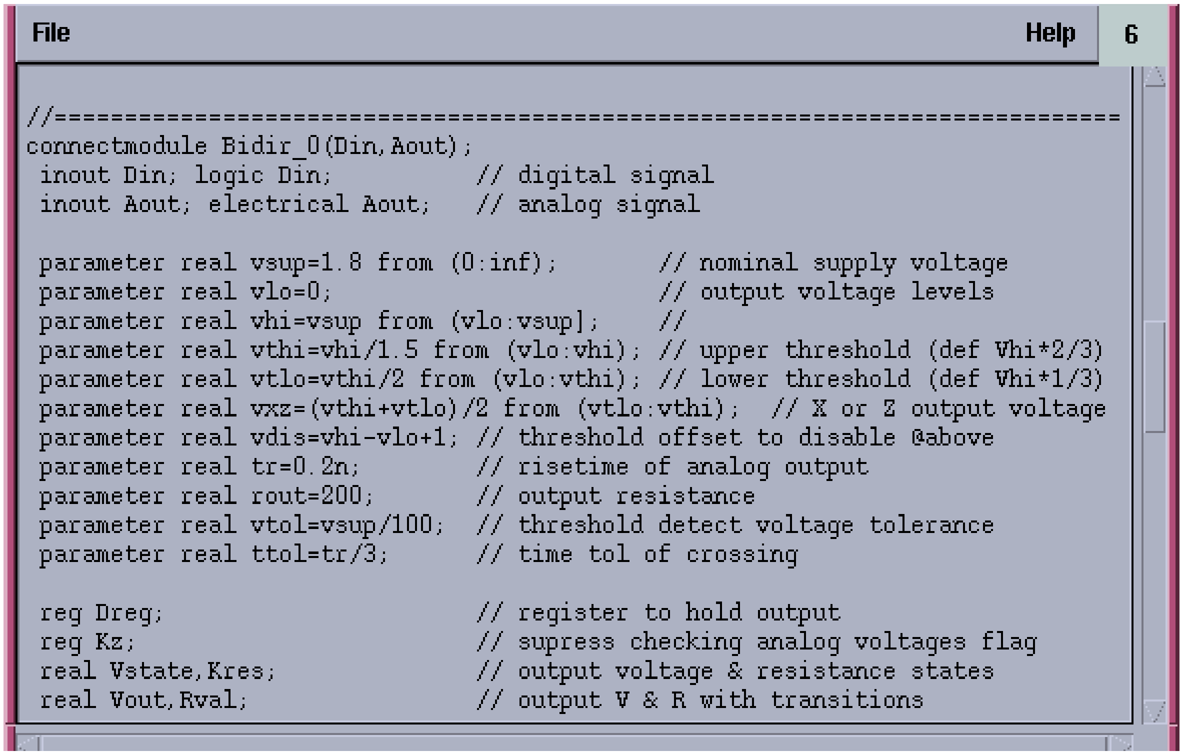
   - *Module* shows the name of a connect module.
   - *Mode* can be blank or have either of the values `merged` or `split`. When it is blank, it indicates that there is only one port of discrete discipline on the signal. The `split` value indicates that there should be one connect module inserted for each port. The `merged` value, which is the default, specifies that only one connect module should be inserted for all the ports on a signal.
   - *Parameter/Values* shows a list of parameter values.
   - The direction of the first port appears in the *Direction1* column and for the second port in the *Direction2* column. These columns can be blank or have one of these values: `input`, `output` or `inout`.

     You may need to scroll to the right to see these and the remaining columns.
   - The discipline information for the first port appears in the *Discipline1* column and for the second port under *Discipline2*. These columns may be blank for modules that do not have them specified.

   Select a module by clicking on the related row.
3. When a module is selected, these fields get populated with the related values: *Mode*, *Direction1*, *Direction2*, *Discipline1*, and *Discipline2*. After you modify one or more values, click the *Change* button just below the *Connect Module Declarations* table. The changes are reflected in the table. If you select multiple rows, this *Change* button appears disabled.

   When you select a row, the *Parameters* table is also populated. You can change values by selecting a row in this table, modifying the *Value* and clicking the *Change* button next to it.
4. If you want to see the connect module file, click the *View connect module* button.

This brings up the related connect modules file as shown below.



```
//==============================================================================
connectmodule Bidir_0(Din,Aout);
 inout Din; logic Din;           // digital signal
 inout Aout; electrical Aout;    // analog signal

 parameter real vsup=1.8 from (0:inf);       // nominal supply voltage
 parameter real vlo=0;                       // output voltage levels
 parameter real vhi=vsup from (vlo:vsup];    //
 parameter real vthi=vhi/1.5 from (vlo:vhi); // upper threshold (def Vhi*2/3)
 parameter real vtlo=vthi/2 from (vlo:vthi); // lower threshold (def Vhi*1/3)
 parameter real vxz=(vthi+vtlo)/2 from (vtlo:vthi);  // X or Z output voltage
 parameter real vdis=vhi-vlo+1; // threshold offset to disable @above
 parameter real tr=0.2n;        // risetime of analog output
 parameter real rout=200;       // output resistance
 parameter real vtol=vsup/100;  // threshold detect voltage tolerance
 parameter real ttol=tr/3;      // time tol of crossing

 reg Dreg;                      // register to hold output
 reg Kz;                        // supress checking analog voltages flag
 real Vstate,Kres;              // output voltage & resistance states
 real Vout,Rval;                // output V & R with transitions
```

If a corresponding file does not exist, an error message appears.

5. When you select only one module in the *Connect Module Declarations* table, the related parameters and values are listed in the *Parameters* table. You can select a parameter and change its value by typing over it in the *Value* field and clicking the *Change* button next to it.

   When you select multiple rows in the *Connect Module Declarations* table, a union of all the parameters is shown in the *Parameters* table with their values. If a parameter has different values in different modules, its value is shown as blank. You can specify a value to be used for a particular parameter in the *Value* field and click the *Change* button next to it. The specified value applies to all the selected modules.

6. The *Direction1* and *Direction2* fields are blank by default. The possible combinations of values you may specify are:

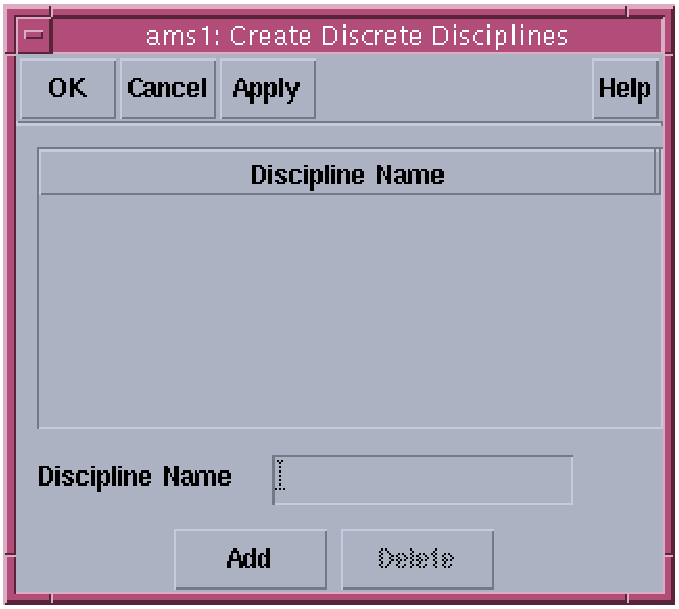
   ❑ *input*, *output*

   ❑ *output*, *input*

   ❑ *inout*, *inout*

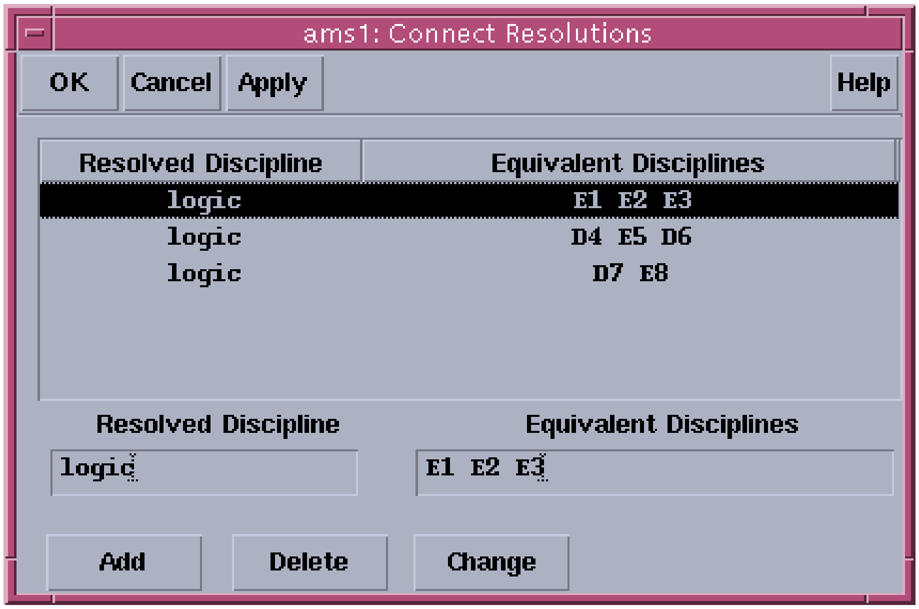
7. The *Discipline1* and *Discipline2* fields are also blank by default. They may remain blank or you may specify a discipline-pair for the selected module.

You can also create discrete disciplines and specify them in these fields by clicking the *Disciplines* button, which brings up the Create Discrete Disciplines form.



It shows a list of disciplines. You may type in a name in the *Discipline Name* field and click *Add* to create a new discipline. The name should be a legal verilog identifier. You may select one or more discipline names listed in the table and click the *Delete* button to delete them. This information is saved for the session when you click *OK* and is included in the connect modules information saved in a state file. For information on states, see Saving and Restoring the Simulation Setup on page 77.

8. The *Connect Resolutions* button brings up the Connect Resolutions form.



Resolved Discipline	Equivalent Disciplines
logic	E1 E2 E3
logic	D4 E5 D6
logic	D7 E8

Resolved Discipline: logic

Equivalent Disciplines: E1 E2 E3

It shows a list of *Resolved Disciplines* and *Equivalent Disciplines*. You can specify a discipline to be used when multiple nets with compatible disciplines are part of the same mixed net. For example, in the form shown above, *logic* is the discipline to be used in the discipline resolution process for the *E1*, *E2* and *E3* disciplines.

- ❑ You can add a new association by specifying a *Resolved Discipline* and an *Equivalent Discipline* and clicking the Add button.
- ❑ You can modify an association by selecting a row, modifying the values in the *Resolved Discipline* and *Equivalent Discipline* fields and clicking the *Change* button.
- ❑ You can select one or more rows and delete them by clicking the *Delete* button.

9. Click the *OK* button to update the connect rules information with the changes made for connect resolutions.

10. Click the *OK* button in the Customize Built-in Rules form to update the connect rules information for the session.

Using Matlab/Simulink

The ability to run a cosimulation using Matlab®/Simulink ®with AMS is now available. There are three use models to choose from:

- From ADE: You can start Matlab from ADE.

- Separate: DFII/ADE is started and then Matlab is separately started in standalone mode and then the two communicate for the co-simulation.
- From MATLAB: AMS is started from Matlab via the runSimulation script.

This User Guide describes the ADE integration of Matlab. For details on other two flows, refer to AMS Designer Simulator User Guide.

Note: MATLAB and Simulink are registered trademarks of The MathWorks, Inc.

Setting up the AMS/MATLAB Cosimulation

In order to consimulate between AMS with Simulink®, coupler modules are required for both AMS and Simulink. For AMS, the coupler module, will be placed as part of your design in the schematic. Associated with each coupler is a verilog.vams file that contains the coupler module. This coupler module contains the system calls: `$couple_init` which calls the VPI code that will be used to communicate with Simulink. The system task `$set_access_readwrite` ensures that the proper read/write access is set for the coupler module..

There are two types of coupler cells to choose from: fixed cell coupler or pcell coupler. The pcell coupler is located in analogLib. For more information see, description for simulinkCoupler in Analog Library User Guide. You can place the pcell coupler and configure the number of input and output pins along with other options described below. You can create the matching `verilog.vams` file that contains the correct inputs/outputs as configured in the pcell on the schematic by using the GUI in ADE as described below.

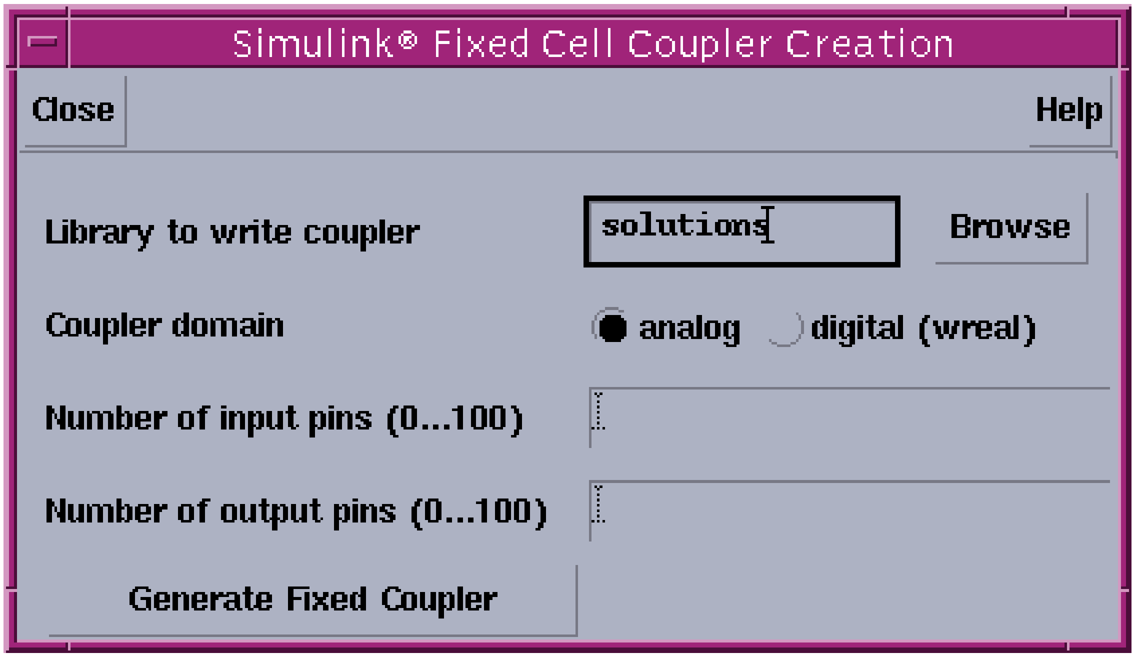
If there are specific input/output configurations that you know you will use frequently, you can create a fixed cell coupler and use that fixed cell in your schematic rather than a pcell. The `verilog.vams` file will be created for the fixed cell when the fixed cell is created. To create a fixed cell coupler:

1. In the schematic, choose *Tools - AMS Opts*.

   This will result in the AMS menu being added to the Composer banner.

2. Select *AMS - Simulink Coupler Creation*.

The Simulink Coupler Fixed Cell Creation form opens.



Library to write coupler: solutions

Coupler domain: ● analog ○ digital (wreal)

Number of input pins (0...100):

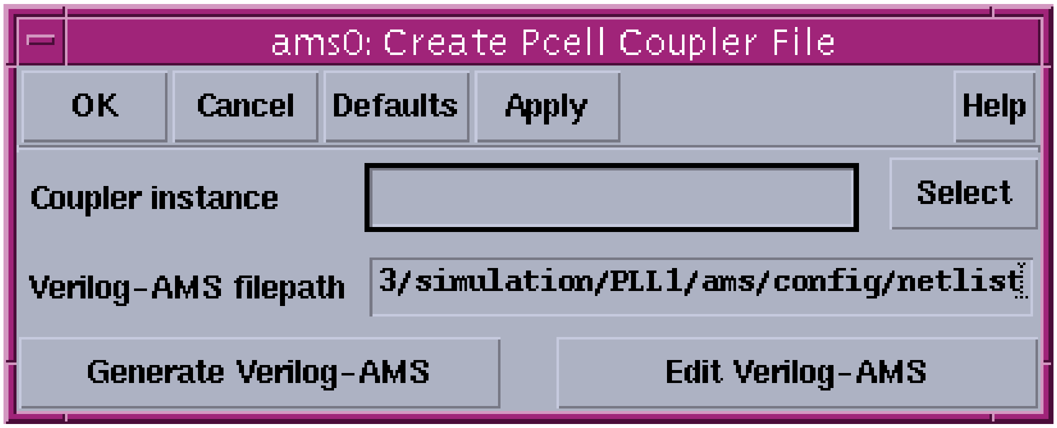
Number of output pins (0...100):

3. Enter the Library name where you want the fixed coupler to be placed. You may want to create a coupler library for all the fixed couplers that you would require or create them in your design libraries. Enter the coupler domain, Number of input and output pins and click *Generate Fixed Coupler* button to generate a fixed coupler that can be used in any future design. A fixed cell coupler is created in the library that you specify. The name of the fixed cell coupler that gets created is:
   `coupler_<numInputPorts>_<numOutputPorts>_[d,a]`

Starting MATLAB

You can start Matlab directly from ADE. If your design is using pcell coupler blocks, then you need to create a verilog.ams file for the pcell coupler block before netlisting. In the simulation

window, choose Setup – *Matlab – Create Pcell Coupler File...* The Create Pcell Coupler File form appears.
:



Coupler instance

Verilog-AMS filepath 3/simulation/PLL1/ams/config/netlist

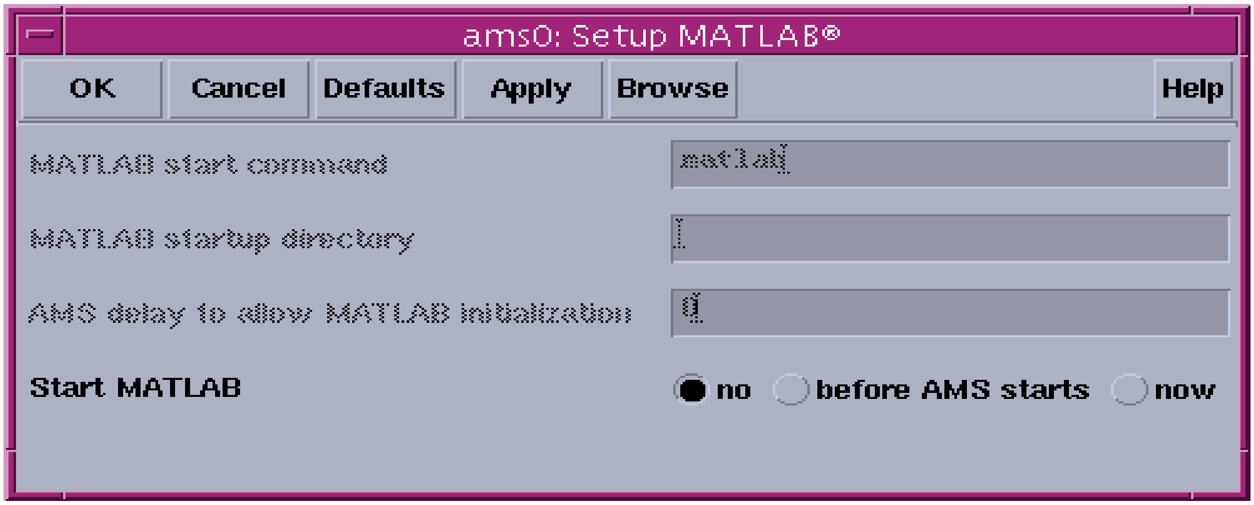
- You can either directly enter the Coupler instance or select the same from the schematic by clicking the Select button.

Note: If the block selected on the schematic is not a pcell coupler block, the message about an incorrect selection appears in the CIW.

- When you click on Generate Verilog-AMS button, a Verilog-AMS file is created for pcell coupler block.
- The Verilog-AMS file generated from this form is placed in the netlist directory by default. You can specify another location in the Verilog-AMS filename field.
- You can also Edit the Verilog-AMS file and save it at its original location.

After creating the verilog.vams file for the pcell coupler, or if you are using a fixed cell coupler, choose *Setup – MATLAB – Start...*

The Setup Matlab form appears.



MATLAB start command: matlab

MATLAB startup directory:

AMS delay to allow MATLAB initialization: 0

Start MATLAB ● no ○ before AMS starts ○ now

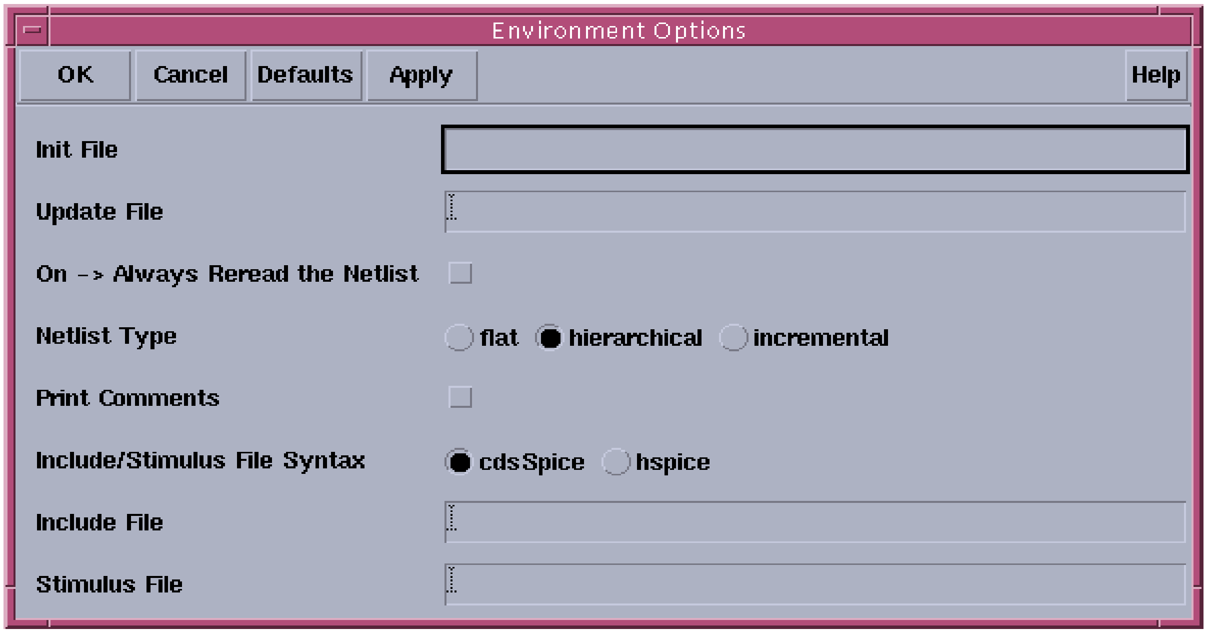
1. Enter the command for starting Matlab using MATLAB start command field. The default command to start Matlab is “matlab”. You can enter any other command along with the command line argument.
2. Enter the path for the directory where matlab is launched in the Matlab startup directory field.
3. Enter simulation initialization timeout in AMS delay to allow MATLAB initialization. It is required to give Simulink a chance to start up before AMS. After Simulink is initialized, you can run the simulation.
4. If you select no option in Start MATLAB, Matlab will not be started. This may be useful if you want to keep same form setup but do not want to run cosimulation or if you wnat to start Matlab yourself, independent of ADE.
5. If you want to start MATLAB automatically, select the before AMS starts checkbox in Start Matlab section. When you select the before AMS starts option, all the fileds in the form are enabled. To run Simulink simulation automatically when AMS starts, you need to have a startup.m file that contains the sim() command in the directory, which is specified in the Matlab startup directory field.
6. If you want to launch MATLAB manually, select now option in start MATLAB. The Start button appears and the AMS delay to allow MATLAB initialization is disabled. Click Start button to launch MATLAB.

Setting Simulation Environment Options for Socket Simulation

To open the Environment Options form,

1. In the Simulation or Schematic window, choose *Setup – Environment*.

   The Environment Options form appears.



Init File	
Update File	
On -> Always Reread the Netlist	□
Netlist Type	○ flat ● hierarchical ○ incremental
Print Comments	□
Include/Stimulus File Syntax	● cdsSpice ○ hspice
Include File	
Stimulus File	

   For detailed information about the form, see “Environment Options” on page 98.

   The display varies depending on which simulator you are using. Instance-based view switching is supported only for purely analog designs, not for mixed-signal designs.

2. Check that the paths to these files are correct:

   - Init file, typically for setting parameters and defining functions
   - Update file, typically for updating model variables
   - Stimulus or include files, containing other simulator information

   You do not need to enter the full path for the file if the file is in the directory specified in the model path.

   Note: A period (.) in a UNIX path specification is interpreted relative to the directory from which you started the Cadence tools.

3. (Optional) If you are not using a config view, check and set the options for view switching to control how the system netlists hierarchical designs.

4. Choose flat or incremental hierarchical netlisting.

5. Set other options as needed, and click *OK*.

Setting Up a Remote Simulation

To run your Spectre, spectreS or AMS simulation on a remote system where the analog circuit design environment is completely installed,

1. As described in the topic Choosing a Simulator on page 34, choose *Setup – Simulator/ Directory/Host*.

   The Choosing Simulator/Directory/ Host form appears.

2. Select the simulator you want to use for this simulation.
3. Type the path to your project directory.

   The path specified in *Project Directory* should be the path from your local machine to your project directory.

4. Set *Host Mode* to *remote*.
5. Type the name of the host system that will run the simulation.
6. Type the path to your project directory relative to the remote system.

   The path specified in *Remote Directory* (accessed from the remote machine) is the absolute path from the remote machine to your project directory.

Note: The Analog Design Environment creates the simulation input file (`input.scs`) and the simulation command file (`runSimulation`) and runs it on the local/remote hosts through `ipcBeginProcess` API. This IPC call is similar to running commands on the remote host through `rsh`.

Basic Requirements for Running a Remote Simulation

- The directory on the host where the Analog Design Environment is running should have the exact file system path on the remote host also.
- Users should be able to perform an `rsh` on the remote host without any login/password.
- Once `rsh` is successful, the login SHELL run command file (`.cshrc` for CSH) should contain appropriate path settings for the Cadence hierarchy. This implies that all the executables should be in the path and LICENCE should be set to the appropriate licence server.
- The `cdsServIpc` process should be running on the local as well remote hosts.

Using a Third-Party Simulator for Remote Simulations

To set up a remote simulation with a third-party simulator (or with Spectre if the remote system has only the Spectre simulator and its license server installed),

1. Check that you have a home directory on the remote system.

   The directory must be in the same location as on your local system, and you must have write permission. For example, if your local home directory is `/home/fred`, the remote machine must also have a home directory called `/home/fred`.

2. In the Choosing Simulator/Directory/Host form, set *Host Mode* to *local*.

   The script to run the simulation is a local file.

Scripts for Using Third-Party Simulators in Remote Simulations

Before you can run remote simulations with a third-party simulator (or with the Spectre simulator if the remote system has only the Spectre simulator and its license server installed), your system administrator must set up a script.

1. Move to the `bin` directory in the Cadence hierarchy.

   ```
   cd your_install_dir/bin
   ```

2. Move the script called `hspiceArtRem` to another directory.

   Choose a directory that comes before the Cadence `bin` directory in your UNIX path. For example, use

   ```
   mv hspiceArtRem ~/spectre
   ```

   for running the Spectre simulator remotely.

Note: Do not move or delete the executable for your simulator. The script name, however, needs to match your simulator name.

3. Check that the script comes before the simulator executable in your search path.

   For example,

   ```
   which spectre
   ```

   needs to return the path to the script, not to the executable.

4. Edit the script and make these changes:

   a. Change the machine name from `cds8715` to the name of the remote host running the simulator.

b. Change `/usr/meta/bin/hspice` to the directory where the simulator is located on the remote machine.

c. On HP systems only, because the `rsh` command is called `remsh`, remove the comment character from the line

```
remshell=remsh
```

When you run a simulation, the script you copied runs the simulator on the remote machine. The PSF files are written to your home directory on the remote machine. The script then copies these PSF files back to the PSF directory on the local machine. The analog design environment reads these PSF files for waveforms and backannotation.

About the Simulation Environment

Configuration of the Virtuoso® analog design simulation environment depends on several kinds of configuration settings:

- Settings you choose in the Editing Session Defaults form
- Settings in your personal and site `.cdsinit` files
- Settings in your personal and site `.cdsenv` files
- UNIX environment variables you set in your `.cshrc` file

Note: These options determine the configuration of the analog circuit design environment. You configure the simulator itself with the Environment Options form.

Saving and Restoring the Simulation Setup

You can save and restore all or part of the simulation environment setup with the *Session – Save State* and *Session – Load State* commands.

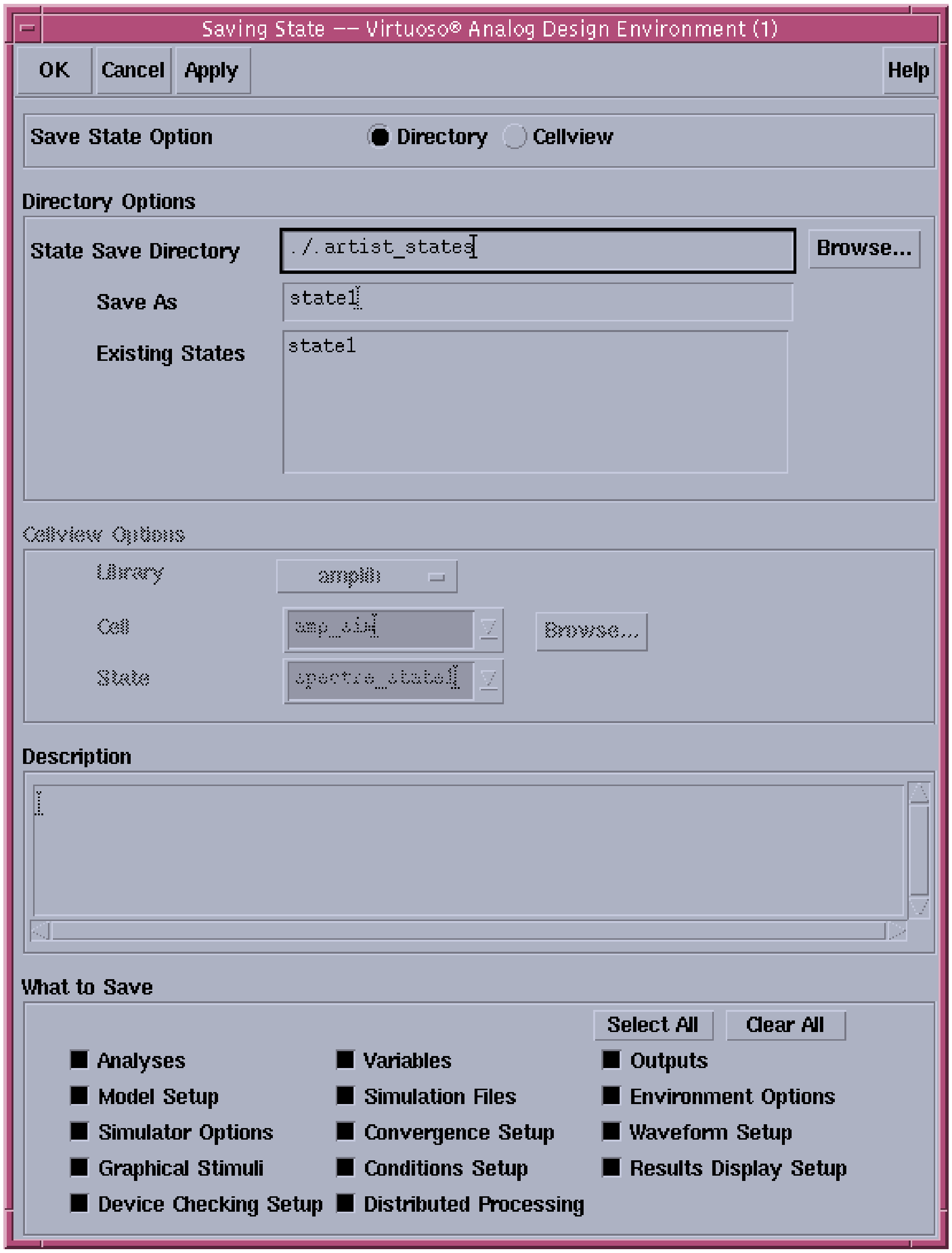
Saving the waveform setup using the *Saving State* form saves the same information as the *File – Save as* command in the waveform window (if *WaveScan* is your waveform viewer). If AWD (A*nalog Waveform Display*) is your waveform viewer, *Window – Save* is the command.

Saved states are simulator dependent for analyses, simulator options, and convergence setup (if the convergence commands you saved are not supported by the other simulator). You can restore saved states from different simulators. The analog circuit design environment restores as much as possible despite simulator-dependent settings.

Saving the Simulation Setup

Once you set up the analog circuit design environment to run a simulation, you can save most of the simulation setup with the *Session – Save State* command.

1. Set up the Virtuoso® analog design simulation environment.
2. In the Simulation window, choose *Session – Save State*, or from the Schematic window, choose *Analog Environment – Save State*. The *Saving State* form appears.



Save State Option ● Directory ○ Cellview

Directory Options

State Save Directory ./.artist_states

Save As state1

Existing States state1

Cellview Options

Library ampLib

Cell amp_sim

State spectre_state1

Description

What to Save

■ Analyses
■ Model Setup
■ Simulator Options
■ Graphical Stimuli
■ Device Checking Setup
■ Variables
■ Simulation Files
■ Convergence Setup
■ Conditions Setup
■ Distributed Processing
■ Outputs
■ Environment Options
■ Waveform Setup
■ Results Display Setup

For detailed information about the form, see “Saving State” on page 102.

3. Select either of the option buttons *Directory* or *Cellview* to indicate that you want to save the state into a directory or as a cellview. The default option is *Directory*.

   a. When the *Directory* option is selected, the fields in the *Directory Options* group box appear enabled. You need to specify a path in the *State Save Directory* by typing it in or by using the *Browse* button to locate it. The *Existing States* list shows all the states saved in that location. You need to also specify a name for the new state in the *Save As* field by selecting one of existing state names to overwrite it or by typing a new name.

   b. When you select the *Cellview* option, the fields in the *Cellview Options* group box appear enabled and those in the *Directory Options* group box appear disabled. You need to specify a *Library* and *Cell* where you want to save the state you specify in the *State* field. You may either type these in or specify them using the *Browse* button.

      If you choose an existing state from a library that is data-managed, and click *OK* and if your *Auto Checkin* and *Auto Checkout Preferences* are enabled, the existing state is checked out and the state being saved is checked in. For more information, see about, see *Chapter 10* of the *Cadence Design Framework II User Guide*.

      ADE states saved as a cellview can be seen in the list of views in the Library Manager. You can apply the same operations on them as you can on views, such as copy, delete, check out, check in and so on.

4. Use the *Description* field to enter a short description about the current state.

5. The substates displayed in the *What to Save* section are also enabled or disabled according to the substates in the selected state. You can enable or disable any of the substates individually by selecting the checkboxes next to them. You may also select all of them or none of them collectively by using the *Select All* or *Clear All* checkboxes at the upper left corner of the *What to Save* group box.

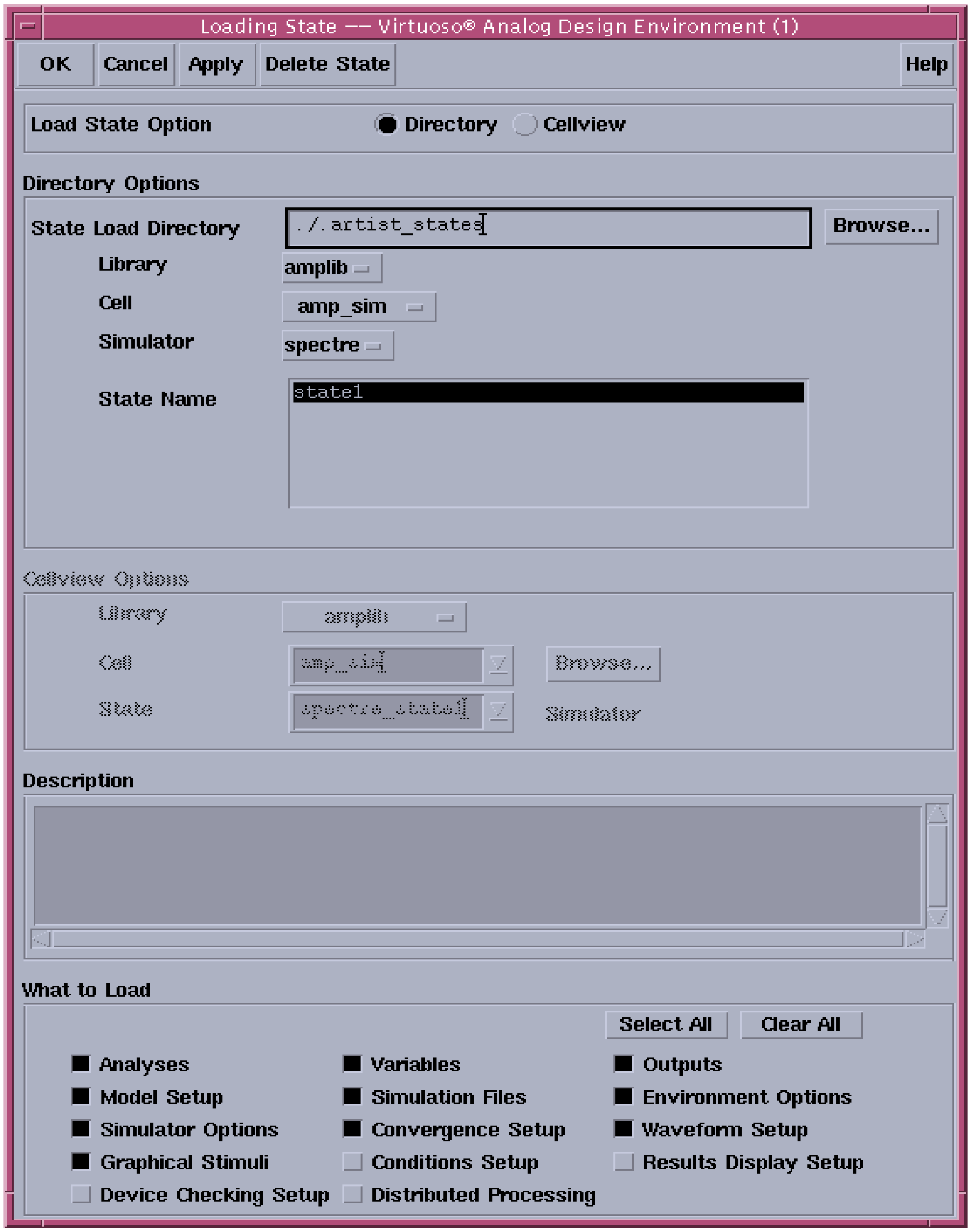
6. Click *OK* to save the specified state.

Restoring the Simulation Setup

To restore all or part of a saved setup,

1. From the Simulation window, choose *Session – Load State*, or from the Schematic window, choose *Analog Environment – Load State*.

The Loading State form appears.



Loading State — Virtuoso® Analog Design Environment (1)

Load State Option ● Directory ○ Cellview

Directory Options

State Load Directory ./.artist_states
Library amplib
Cell amp_sim
Simulator spectre
State Name state1

Cellview Options

Library amplib
Cell amp_sim
State spectre_state1
Simulator

Description

What to Load

■ Analyses
■ Model Setup
■ Simulator Options
■ Graphical Stimuli
□ Device Checking Setup
■ Variables
■ Simulation Files
■ Convergence Setup
□ Conditions Setup
□ Distributed Processing
■ Outputs
■ Environment Options
■ Waveform Setup
□ Results Display Setup

For detailed information about the form, see “Loading State” on page 104.

2. Select either of the option buttons *Directory* or *Cellview* to indicate that you want to load the state from a directory or from a cellview. The default option is *Directory*.

   a. When the *Directory* option is selected, the fields in the *Directory Options* group box appear enabled. You need to specify a path in the *State Load Directory* by typing it in or by using the *Browse* button to locate it. Select the desired values in the *Library*, *Cell*, and *Simulator* fields. *State Name* specifies all the saved states for the cell and simulator combination that you specified. Select the state name you want to load.

b. When you select the *Cellview* option, the fields in the *Cellview Options* group box appear enabled and those in the *Directory Options* group box appear disabled. You need to specify a *Library*, *Cell* and *State*ou want to load. You may either type these in or specify them using the *Browse* button.

3. The *Description* field shows a short description about the selected state.
4. The substates displayed in the *What to Load* section are based on the substates in the selected state. You can enable or disable any of the substates individually by selecting the checkboxes next to them. You may also select all of them or none of them collectively by using the *Select All* or *Clear All* checkboxes at the upper left corner of the *What to Save* group box.
5. You can click *OK* to load the specified state.

   The system restores as much of the information as possible, ignoring settings from other simulators that are incompatible with the simulator that you have selected.
6. To delete a selected state, click the *Delete State* button.

Note: While loading a state from one simulator to other, the variable:

- may have the same name and value type. In this case, the variable is loaded.
- may not exist. In this case, no warning appears.
- is the same but the value type may be different or the value is not in the displayed list. In this case, an error is issued.

Saving a Script

The Open Command Environment for Analysis (OCEAN) lets you set up, simulate, and analyze circuit data. OCEAN is a text-based process that you can run from a UNIX shell or from the Command Interpreter Window (CIW). You can type in OCEAN commands in an interactive session, or you can create scripts containing your commands, then load those scripts into OCEAN. OCEAN can be used with any simulator integrated into the analog circuit design environment.

OCEAN lets you

- Create scripts that you can run repeatedly to verify circuit performance

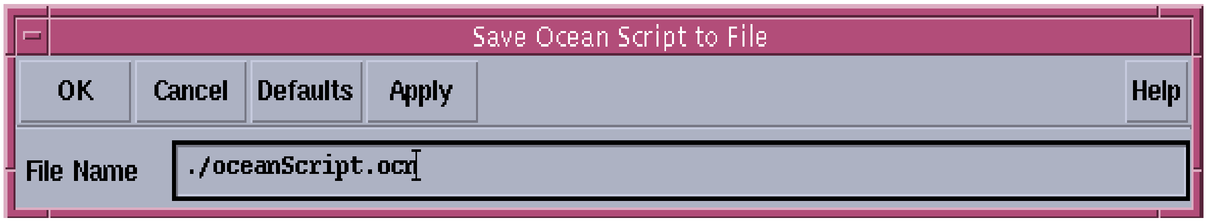
- Run longer analyses such as parametric analyses, corners analyses, and Monte Carlo simulation, more effectively
- Run long simulations in OCEAN without starting the analog circuit design environment graphical user interface
- Run simulations from a nongraphic, remote terminal

From the Simulation window, you can set up your simulation environment, choose plotting options, and save them in a script.

To create a script from the analog circuit design environment,

1. Make the setup and plot processing selections that you want to capture in the script.
2. Choose *Session – Save Script*.

   The Save Ocean Script to File form appears.



File Name ./oceanScript.ocn

3. Click *OK* to accept the default filename (`~/oceanScript.ocn`), or change the name of the file and click *OK*.

   The design environment creates and displays a script containing the OCEAN setup and plotting tasks you performed. You can edit the script to add simulation or postprocessing commands as needed.

   Note: The *`spectreFormatter`* and associated methods are defined in the *spectreinl* context. This does not get loaded automatically by *asiGetTool('spectre*). To load relevant contexts, you need to edit your code by adding *spectreinl* to the list of contexts it loads. This is illustrated in the following example:

```
; Need to have these contexts loaded - in the right order
; - so that that the spectreFormatter class can be extended
(foreach context '("oasis" "asimenv" "analog" "spectrei" "spectreinl")
  (unless (isContextLoaded context)
          (loadContext
           (prependInstallPath (strcat "etc/context/" context ".cxt")))
          (callInitProc context)
          )
  )
```

For more information about OCEAN commands and scripts, see the *OCEAN Reference*.

Resetting the Default Environment

You can reset the simulation environment to its initial default state.

Note: If you want any of the data from your current session, you need to save it using the *Session – Save State* before you use *Session – Reset*. When you use *Reset*, any data currently displayed is overwritten with the default data.

To reset the simulation environment to its default state,

➤ In the Simulation window, choose *Session – Reset*.

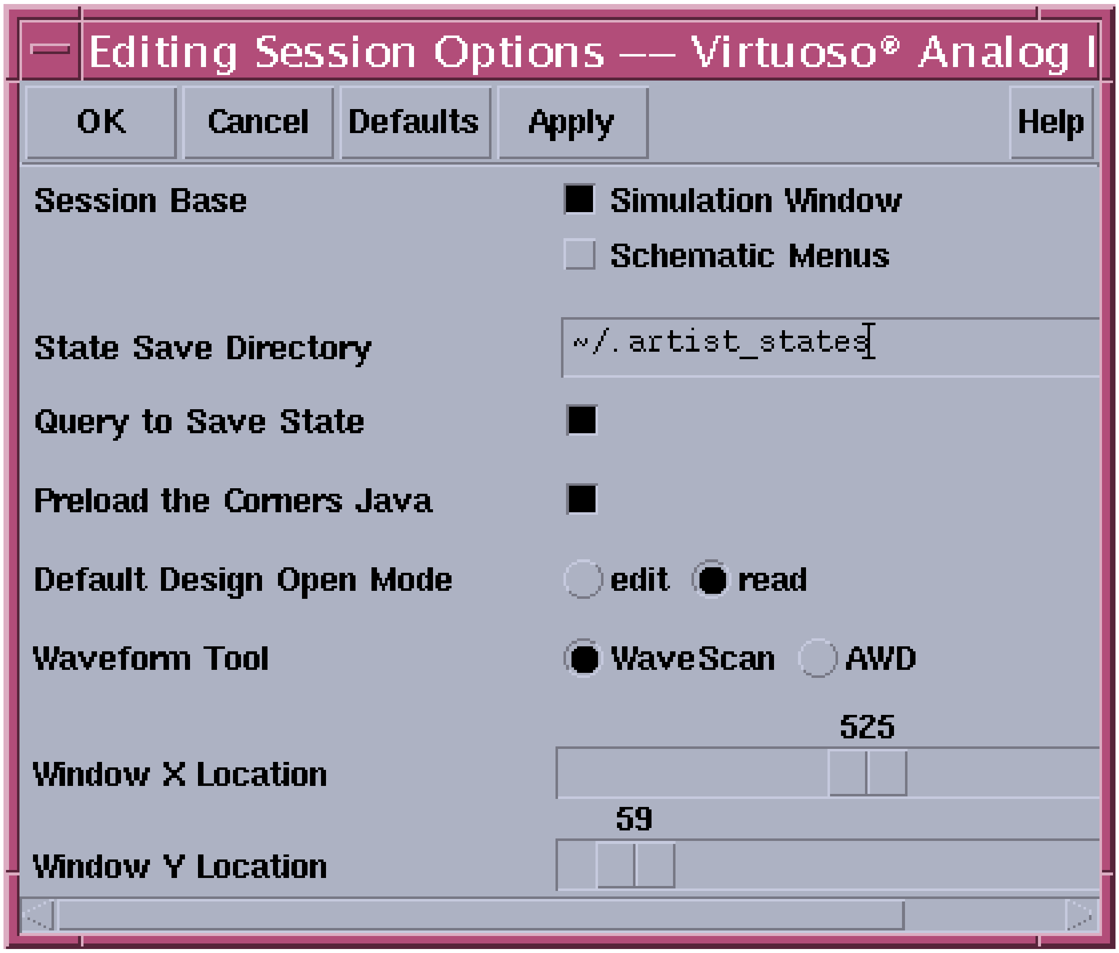
This command restores the original defaults. If you have data in this form, it is overwritten with default settings. Use the *Session – Save* command to save this information before resetting the defaults.

Setting Basic Session Defaults

To set the basic defaults, including your window preference,

1. From the Simulation window, choose *Session – Options*, or from the Schematic window, choose *Analog Environment – Options*.

The Editing Session Options form appears.



Session Base	■ Simulation Window □ Schematic Menus
State Save Directory	~/.artist_states
Query to Save State	■
Preload the Corners Java	■
Default Design Open Mode	○ edit ● read
Waveform Tool	● WaveScan ○ AWD
Window X Location	525
Window Y Location	59

For detailed information about the form, see "Editing Session Options" on page 106.

Note: Changes take effect the next time you start the Analog Design Environment. They do not take effect immediately.

2. Choose a session base.

   If you want a Simulation window to open when you start the analog circuit design environment, choose the *Simulation Window* option.

   If you want the *Analog Environment* menus to appear on the Schematic window when you start the Virtuoso® analog design simulator, choose the *Schematic Menus* option.

3. To save the state of your environment in a specific location, type the path for this location in the *Save State Directory* field.

4. To be prompted to save the state of your environment each time you close an Virtuoso® analog design simulation window, choose *Query to Save State*.
5. To set the default mode whenever you open your Schematic window, choose *edit* or *read* in *Default Design Open Mode*.
6. To specify the default location for your Simulation window, use the *Window X Location* and *Window Y Location* fields to adjust the positioning.

Netlisting Control Variables

You can customize ADE netlisting by setting the following variables.

Variable	Description
nlReNetlistAll	When set to t, all cellviews in the design are re-netlisted, irrespective of the setting specified using the ADE GUI options *Create* or *Recreate* in the *Simulation – Netlist* submenu. If `nlReNetlistAll` is not set and has its default value, `nil`, the ADE GUI netlisting options are used. You can set this variable in .simrc file.
globalGroundNet	When you use this variable to specify a string, the value specified is considered to be the global ground net and overrides the default analogLib global ground net (gnd!). You can set this variable in .simrc file. This variable is valid for socket direct netlister only.
simPropValueNotation	When set to `Engineering`, all property values are displayed in engineering notation in the netlist. For Example, a resistance value of `1m` is displayed as `1e-3`. If `simPropValueNotation` is set to `Scientific`, all property values are displayed in scientific notation in the netlist. For Example, a resistance value of `1m` is displayed as `1.000000000e-03.` You can set this variable in si.env file. This variable is valid for socket netlisters only.

Customizing Your .cdsinit File

You can customize your analog circuit design environment by adding SKILL commands to your `.cdsinit` file, the initialization file for the Cadence software.

There is a sample `.cdsinit` file for the analog circuit design environment in the following location:

`your_install_dir``/tools/dfII/samples/artist/.cdsinit`

Customizing Your .cdsenv File

You can set the default values for fields in analog circuit design environment forms by setting variables in your `~/.cdsenv` file.

There is a sample `.cdsenv` file for the analog circuit design environment in the following location:

`your_install_dir``/tools/dfII/etc/tools/`*`simulator_name`*

Customizing Your Menus File

The menus file is a simple SKILL file, therefore you can customize the same menu file for different releases by adding skill code within the *if-then-else* statement.

```
(if (equal curVersion 44X) then
   ;;; 44X menus customization here
else
   ;;; 44Y menus customization here
)
```

Alternatively, you can create the `simui.menus` file like this:

```
(if (equal curVersion 44X) then
    load "44X_file"
)
(if (equal curVersion 44X) then
    load "44Y_file"
)
```

where `44X_file` and `44Y_file` are path to the menus file for different releases.

Setting UNIX Environment Variables

UNIX environment variables help configure the Virtuoso® analog design simulation environment.

The `CDS_Netlisting_Mode` variable controls

- How parameter values on components that use CDF and AEL are interpreted during netlisting
- Which LVS tool the system uses

The syntax for this variable in a `.cshrc` file is

```
setenv CDS_Netlisting_Mode "{Analog|Digital|Compatibility}"}
```

When the `CDS_Netlisting_Mode` variable is set to `Analog` or `Compatibility`, the component parameter evaluation takes CDF and AEL into account. When the variable is set to `Digital`, CDF and AEL are not taken into account, which results in better netlisting performance. When the variable is set to `Analog`, the Analog LVS tool (`auLvs`) is used. For more details on auLvs, see Appendix E, "auLvs Netlisting,"

Use these rules to set `CDS_Netlisting_Mode`.

- When you use the analog circuit design environment, set this variable to an appropriate value. If your design depends on CDF information, then set `CDS_Netlisting_Mode` to `Analog`. If your circuit does not use CDF information, then set `CDS_Netlisting_Mode` to `Digital` or `Compatibility`.
- When you use socket simulation in the analog circuit design environment and this variable is not set or is set to `Digital`, you see a dialog box that lets you set the value to `Analog` for the current session. (The socket netlister requires that the variable be set to `Analog`.) In this situation, the design environment uses the Analog LVS (`auLVS`) tool.
- If you use the analog circuit design environment for LVS, set `CDS_Netlisting_Mode` to `Analog`.
- If you use CDF and the Circuit Designer's Workbench but do not have the analog circuit design environment, set `CDS_Netlisting_Mode` to `Analog`.
- If you do not have the `auLVS` tool and do not use CDF or AEL expressions, set the variable to `Digital`. In this mode, you get the fastest netlisting speed and run `iLVS`, which uses OSS NLP expression syntax.
- If you want CDF compatibility with `iLVS`, set the variable to `Compatibility` for faster netlisting than with `Analog`. However, note that `Compatibility` mode has the following limitations:
  - ❑ Connection of terminals by properties is not supported. A typical use of this capability in the analog circuit design environment is connecting the bulk pin of four-terminal transistors to a net.
  - ❑ Analog circuit design environment features other than expression evaluation are not supported.

If you do not set `CDS_Netlisting_Mode`, the default depends on which command you use to start the Cadence tools.

Command	Default for CDS_Netlisting_Mode
`icms`	`Analog`
`msfb`	`Analog`
all others	`Digital`

Reserved Words

Reserved Words in Direct Simulation

Each simulator has reserved words you cannot use as names for design variables or passed parameters of a subcircuit (use in a pPar expression):

- Simulator command or function names
- Simulator global variables, including
  - ❑ Simulator options
  - ❑ Names on fixed-form fields

  For example, CDF parameters on CDF forms, or properties on property forms, are all reserved words.

All Spectre simulator reserved words can be found in the *Spectre Circuit Simulator Reference*.

Reserved Words in Socket Simulations

Each simulator has reserved words you cannot use, such as net, component, or property names:

- Simulator command or function names
- Simulator global variables
  - ❑ Simulator options
  - ❑ Names on fixed-form fields

For example, CDF parameters on CDF forms, or properties on property forms, are all reserved words

- Miscellaneous words, including
    - ❑ others
    - ❑ loadSave
    - ❑ simInfo
    - ❑ othersFG
    - ❑ rep

All Cadence SPICE reserved words can be found in the *Cadence SPICE Reference Manual*. All Spectre reserved words can be found in the *Spectre Circuit Simulator Reference*.

What Are Bindkeys?

Bindkeys are keys you can program to run commands instead of using the mouse to choose the command from a menu. You can set bindkeys temporarily during a design session, or you can set them for all subsequent sessions in your `.cdsinit` file.

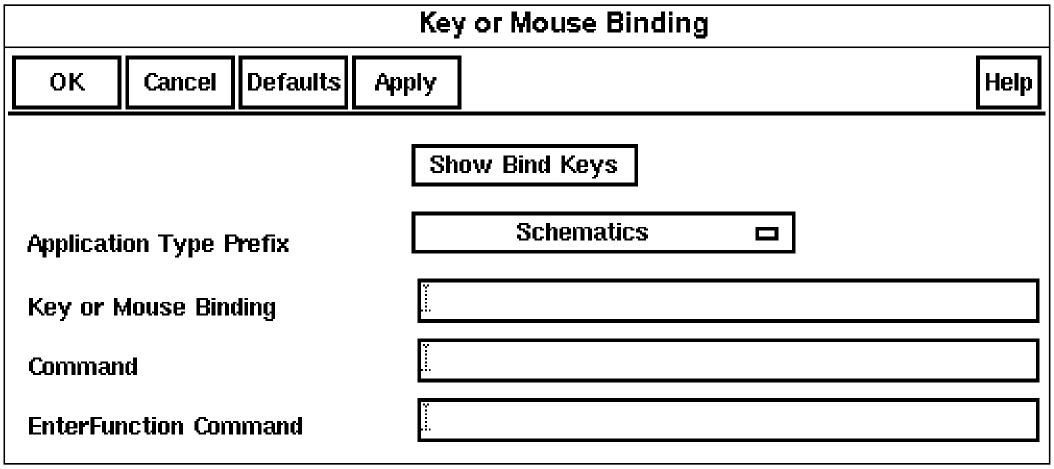
For more information about setting bindkeys, see the *Design Framework II Help*.

Checking Bindkey Assignments

To see if any bindkeys are already defined for your Virtuoso® analog design system,

1. In the CIW, choose the *Options – Bindkey* command.

The Key or Mouse Binding form appears.



Application Type Prefix: Schematics

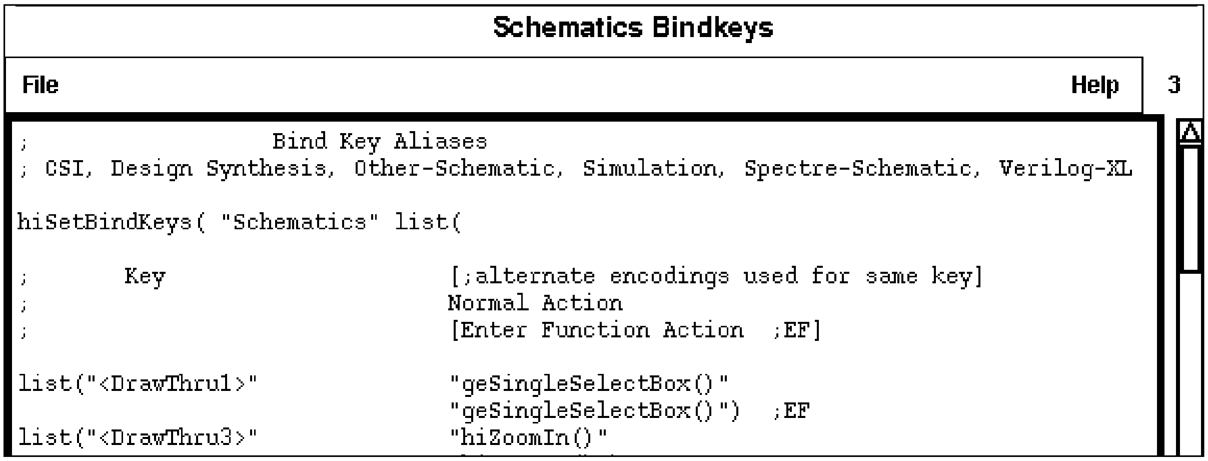
Key or Mouse Binding

Command

EnterFunction Command

2. From *Application Type Prefix*, choose *Schematics*.
3. Click *Show Bind Keys*.

   A text window appears showing the bindkeys defined for your system.



```
;                    Bind Key Aliases
; CSI, Design Synthesis, Other-Schematic, Simulation, Spectre-Schematic, Verilog-XL

hiSetBindKeys( "Schematics" list(

;       Key                       [;alternate encodings used for same key]
;                                 Normal Action
;                                 [Enter Function Action  ;EF]

list("<DrawThru1>"                "geSingleSelectBox()"
                                  "geSingleSelectBox()")  ;EF
list("<DrawThru3>"                "hiZoomIn()"
```

In this example, the system has one bindkey defined: the key sequence `Control-c` performs the same function as the *Simulation – Interrupt* command.

4. Choose the *File – Close* command to close the window.

Assigning Bindkeys

There are three ways to bind a key or mouse button to a function (command) in the Virtuoso® analog design simulation environment. You can

- Use the Key or Mouse Binding form called by the *Options – Bindkey* command in the CIW
- Type the SKILL command to set the bindkey directly in the CIW
- Type the SKILL command to set the bindkey in your `.cdsinit` file

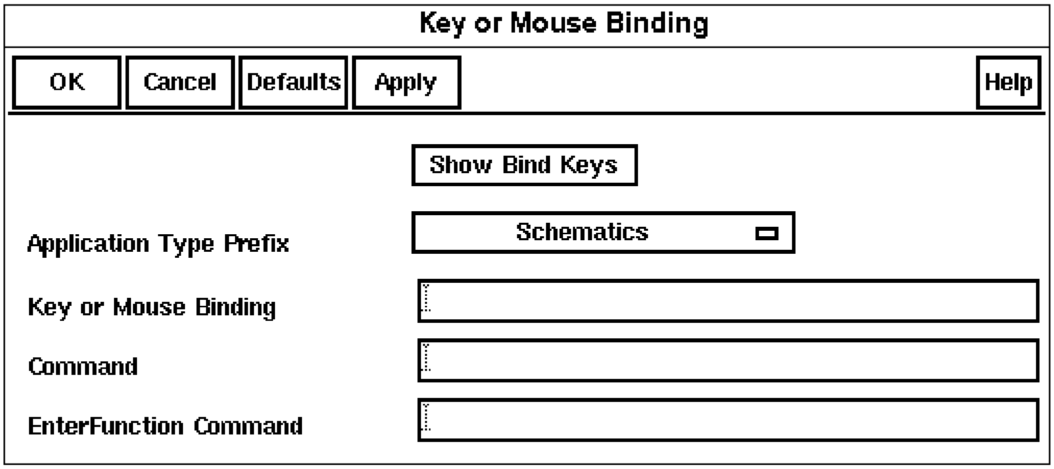
Bindkeys set by the first two methods are valid for the current session only. To set bindkeys you want for every session, you must enter them in your `.cdsinit` file.

Using the Key or Mouse Binding Form

To program an Virtuoso® analog design simulation environment command to a bindkey,

1. In the CIW, choose the *Options – Bindkey* command.

   The Key or Mouse Binding form appears.



Application Type Prefix: Schematics
Key or Mouse Binding:
Command:
EnterFunction Command:

2. In the *Key or Mouse Binding* field, type in the keys to which you want to bind the command.

   You can bind a function to a key (`m`, for example), a combination of keys (`Ctrl<Key>m`, for example) or a mouse button (`<Btn1Down>` for the left mouse button, for example).

3. In the *Command* field, type the SKILL function corresponding to the command.

   Set the CIW Log Filter to display the SKILL functions for the menu commands you enter.

4. Click *OK* or *Apply*.

You can click *Show Bind Keys* in the Key or Mouse Binding form to see the command.

Using the CIW

To program an Virtuoso® analog design simulation environment command to a bindkey through the CIW,

➤ Type the following in the CIW:

```
hiSetBindKey( "Schematics" "<Key>x"
    "SKILL_command" )
```

x is the name of the key to which you want to bind the mouse function.

SKILL_command is the command you type into the CIW to call the function.

For example, to bind the *Setup – Environment* command to the `Control-m` key, type this in the CIW:

```
hiSetBindKey( "Schematics"
  "Ctrl<Key>m" "sevChooseEnvironmentOptions(hiGetCurrentWindow()->sevSession)")
```

To bind the *Setup – Environment* command to the `Shift-s` key, type this in the CIW:

```
hiSetBindKey( "Schematics"
  "Ctrl<Key>s" "sevChooseEnvironmentOptions(hiGetCurrentWindow()->sevSession)")
```

Note: Your cursor must be in the Schematics window when you use the bindkey. Also, bindkeys are no longer supported in the simulation window and work only if you use ADE in the Composer window.

Using Your .cdsinit File

To program an Virtuoso® analog design simulation environment command to a bindkey in the `.cdsinit` file,

1. In a UNIX window in your home directory, type

```
vi .cdsinit
```

2. In the file that appears, type `i` (for insert mode), then type the following

```
hiSetBindKey( "Schematics" "<Key>x"
    "SKILL_command" )
```

x is the name of the key to which you want to bind the mouse function.

SKILL_command is the command you type to call the function.

3. To save your changes and quit, type

```
:wq
```

To avoid running commands inadvertently, typically you want to bind functions to a combination of keys that you do not ordinarily use.

Note: Your cursor must be in the Simulation window when you use the bindkey.

Form Field Descriptions

Choosing Simulator/Directory/Host

Simulator lets you specify the simulator you want by choosing from the options in the cyclic field.

Project Directory lets you specify the run directory.

Host Mode lets you choose local or remote simulation by clicking on the appropriate radio button.

Host lets you specify a path to the host computer for remote simulation. You must specify a full path.

Remote Directory lets you specify a path to the run directory for remote simulation. You must specify a full path.

Digital Host Mode (for Verilog options only) lets you choose local or remote digital simulation by clicking on the appropriate radio button.

Digital Host (for Verilog options only) lets you specify a path to the host computer for digital remote simulation. You must specify a full path.

Create New File

Library Name lets you browse and select the name of the library where the cell for the new file resides.

Cell Name lets you specify the name of the cell for which you are creating the file.

View Name lets you specify the name of the view that you are creating.

Tool specifies the tool that you will be using to create the new file.

Library path file specifies the location of the `cds.lib` file that contains the paths to your libraries.

Setting Model Path

Defaults displays the default directories list and uses it in the current session.

Apply accepts the current directories list as the list to use for the current session.

Apply & Run Simulation accepts the current directories list as the list to use for the current session and starts the simulator.

Directories is the list of paths to model files. The paths are checked in sequence.

New Directory lets you type a new path to be added to the directories list.

Add Above adds the new directory above the selected item in the directories list.

Add Below adds the new directory below the selected item in the directories list.

Change displays the selected directory in the *New Directory* field so that you can change it.

Delete removes the selected directory from the directories list.

Corner provides alternate groups of directories that can be substituted for the current directories list.

New Corner lets you name the current directories list and access it from the *Corner* cyclic field. You name the new corner by typing the name in the *New Directory*.

Copy Corner lets you copy the directories list of the corner specified in the *Corner* cyclic field into the displayed directories list.

Delete Corner lets you delete the corner specified in the *Corner* cyclic field.

Model Library Setup

List box lists all the model files to be included. You enter the model file's name into the *Model File* field. You can include an optional *Section* field.

Add adds the value of the *Model File* field (and *Section* field) to the end of the list of files in the list box.

If a file is selected in the list box, the *Model File* field (with optional *Section*) entry is added above the highlighted item.

Delete removes any items that are selected in the list box. An item in the list box has to be selected for this button to remove the file.

Change replaces the selected entry with the value of the *Model File* field and the optional *Section* field.

Edit File opens the model file named in the *Model File* field in the default editor. If the *Model File* field is a directory or a nonexistent file, an error is reported in the CIW.

Browse calls the Browser, so you can find model files easily and place them in the *Model File* field.

If there is an entry in the *Model File* field and the directory listed there is valid, the Browser starts (lists the contents of) that directory.

If the entry in the *Model File* field is a file (with a path), the Browser opens at the directory part of the path.

If there is nothing in the *Model File* field, or the path listed in the field is invalid, the Browser opens at the directory from which `icms` was started.

OK stores the *Model File* list. When the Model Library Setup form is called up again, the stored list appears in the *Model File* list box.

Apply stores the *Model File* list. When the Model Library Setup form is called up again, the stored list appears in the *Model File* list box. Using *Apply* rather than *OK* causes the form to remain open.

Cancel closes the form. Any entries added after the last *Apply* are stored and any additions or modifications to the list box are discarded.

Defaults loads the default values.

Enable selects highlighted model file(s) for a particular run.

> **Note:** In case any of the highlighted model files are prefixed with a #, clicking the *Enable* button removes the #.

Disable de-selects the highlighted model files. This implies that once the *Disable* button is clicked, the highlighted model files in the *List box* get disabled and a # is added before the model path.

Up moves a highlighted model file upwards. This button gets disabled if more than one model file is highlighted/selected. If this button is clicked with the first file in the list box highlighted/ selected, nothing happens and the *Down* button is activated.

Down moves a highlighted model file downwards. This button gets disabled if more than one model file is selected.If this button is clicked with the last file in the list box highlighted/ selected, nothing will happen.

The order of model files can be re-arranged using the *Up* and *Down* buttons.

The table below summarizes the state (Enabled/Disabled) of buttons in the *Model Library Setup* form depending on the model file selected:

Button	**No Selection**	**One file selected**	**Multiple Files Selected**
Add	Enabled	Enabled	Enabled
Delete	Disabled	Enabled	Enabled
Change	Disabled	Enabled	Disabled
Edit File	Disabled	Enabled	Disabled
Enable	Disabled	Enabled	Enabled
Disable	Disabled	Enabled	Enabled
Up	Disabled	Enabled	Disabled
Down	Disabled	Enabled	Disabled

Environment Options

Init File sets the path to the `init.s` simulation file. The system appends `.s` to the filename you enter.

Update File sets the path to the `update.s` simulation file. The system appends `.s` to the filename you enter.

Parameter Range-Checking File lets you enter the path to a file containing the correct ranges for component parameters. If this path is present, the simulator checks the values of all component parameters in the circuit against the parameter range-checking file and prints out a warning if any parameter value is out of range.

Recover from Checkpoint File (spectreS option only) lets you restart the simulation using the data generated by a spectreS simulation that was interrupted before it could finish. You can set spectreS to print simulation results to a checkpoint file automatically after a time period you specify in one of two places:

- In the *ckptclock* field in the Simulator Options form called by the *Simulate – Options – SpectreS* command.

  The default value is 1800 seconds. Under default conditions, spectreS writes a checkpoint file at 1800 seconds and at every 1800-second interval after the first write. The default state for this option is *on*.

- In the *ckptperiod* field in the Transient Options form called by the *Simulate – Options – Transient* command.

  The default state for this option is *off.*

The SpectreS simulator creates a checkpoint file in the netlist/raw directory under the following name:

```
design.analysis.ckpt
```

Netlist Type lets you select *flat*, *hierarchical*, or *incremental* netlisting. The available options vary depending on the simulator you use.

flat netlists each primitive and its path.

hierarchical netlists each subcircuit and its models.

incremental netlists only those instances that need renetlisting.

Switch View List is a list of the views that the software switches into when searching for design variables. The software searches through the hierarchical views in the order shown in the list.

Stop View List is a list of views that identify the stopping view to be netlisted. This list does not require a particular sequence.

Instance-Based View Switching is available for analog-only designs that do not use a config view.

When off, disables instance-based view switching during netlisting. The netlister uses the values in the *Switch View List* and ignores the values in the *Instance View List Table* and *Instance Stop List Table*.

When selected, it enables instance-based view switching during netlisting. The netlister uses the values in the *Instance View List Table* and *Instance Stop List Table* to override, on a per-instance basis, the default simulation view selection algorithm imposed by the *Switch View List*.

Instance View List Table is a list of sublists. Each sublist is made up of the character string that is the list name followed by a string containing a list of view names. The list name must be the same as the name of the *instViewList* property that you added to an instance with the *Edit Object Properties – Add Property* command.

Instance Stop List Table is a list of sublists. Each sublist is made up of the character string that is the list name followed by a string containing a list of view names. The list name must be the same as the name of the *instStopList* property that you added to an instance with the *Edit Object Properties – Add Property* command.

Analysis Order lets you enter the order for writing analysis statements in the input file passed to the simulator. You can specify any number of analyses in the field without duplication. Any missing analysis are put in the end.

Print Comments

When off, comments are not printed.

When on, extra comments are placed in the netlist regarding component location and name.

UserCmdLineOption helps you specify options that cannot otherwise be specified using the UI. You can enter commands that are valid for the selected simulator. ADE appends these commands as they are to the list of commands specified using the UI for the simulator. If you provide invalid commands through this option, ADE does not validate them; the simulator may or may not fail depending upon the simulator strategy applied to invalid commands.

Include/Stimulus File Syntax specifies the syntax used for an include or stimulus file.

cdsSpice specifies that the file is in Cadence SPICE syntax or that an include file in Cadence SPICE syntax is used to include a second file in the syntax of the target simulator. If you change the include or stimulus file and you select *cdsSpice*, the design is renetlisted.

The simulator-specific option specifies that the file is in the correct syntax for the target simulator. If you change the include or stimulus file and you select the name of the target simulator, the design is not renetlisted.

Include File adds statements to the netlist that are simulator specific. For example, you might want to add special `.model` or `.lib` statements to the netlist for an HSPICE simulation.

Stimulus File sets up a special analog stimulus file.

Automatic output log

When on, the output log opens and displays simulator messages as they are generated.

Use SPICE Netlist Reader(spp)

Y runs spectre with the `+spp` option, which runs the SPICE netlist reader on the input file.

N runs spectre with the `-spp` option, which does not run the SPICE netlist reader on the input file.

Use SPP command-line options file helps you specify the path and name of the SPP file.

Compatibility Mode sets the default netlist type to `hspice` or `spectre`. This option works for both solvers - UltraSim and Spectre. The default values for the simulator options *Temperature* and *Tnom* change according to the choice made.

savestate (ss)

Y runs spectre with the `+ss` option, which saves circuit information at set intervals or at multiple points during a transient analysis.

N runs spectre with the `-ss` option, which does not save circuit information at set intervals during a transient analysis.

Note: This option does not work as intended with advanced analysis tools, such as MonteCarlo, Optimizer, Corners, and parametric analysis. Ensure that it is set to N, it's default setting, before using these tools.

recover (rec) allows the simulation to be restarted from a specified checkpoint.

Y runs spectre with the `+rec` option, which restarts a transient simulation based on conditions specified in a saved-state file.

N runs spectre with the `-rec` option, which does not restart a transient simulation, even if conditions for this have been specified in a saved-state file.

Generate Map File (for mixed-signal simulation)

When off, a node map file is not generated.

When on, a node map file is generated. The node map file maps the schematic names of nets to names used by the simulators in text format.

Interactive instructs the simulator to continue running in the background and to wait for new simulation requests after completing a simulation. The noninteractive default is to run in batch mode, which causes the simulator to exit after completion. The *Interactive* option allows faster response to simulator requests but also locks a simulator license when in use.

Mixed Signal Netlist Mode (for mixed-signal simulation) specifies whether a flat or hierarchical netlist is created. The default is a flat netlist.

Verilog Netlist Option (for mixed-signal simulation) displays either the Flat Netlist Options form or the Hierarchical Netlist Options form.

Saving State

Save State Options lets you specify that you want to save the state within a directory or within a cellview. If you select *Directory*, the fields in the *Cellview Options* group box are disabled. If you select *Cellview*, the fields in the *Directory Options* group box are disabled.

Directory Options lets you specify the *State Save Directory* in which you want to save the state. You may either type it in or specify it by using the *Browse* button. You use the *Save As* field to specify a state name. You may populate it by selecting one of the *ExistingStates* to overwrite or by specifying a new state name.

Cellview Options lets you specify a *Library* and *Cell* where you want to save the state you specify in the *State* field. You may either type these in or specify them using the *Browse* button.

Description lets you specify a short note about the state being saved.

What to Save controls the type of information saved.

> **Select All** saves all the types of information shown.
>
> **Clear All** does not save any of the types of information shown.
>
> **Analyses** saves the Choosing Analyses form settings (but not simulator options that you set with the options buttons in these forms). Analysis form settings are simulator specific, so you cannot restore them from a different simulator.
>
> **Variables** saves design variables.
>
> **Outputs** saves the saved, plotted, and marched output settings. Output settings are design specific, but if the signal names match, you can restore them from a different design. You can restore expressions from any design.
>
> **Model Setup** saves the model setup of the session.
>
> **Simulation Files** saves simulation-specific files specific files and other information. For spectre, these files can be stimulus files, definition files, and include paths.
>
> **Environment Options** saves the environment option settings. If you are using a different simulator in another session, only those options that are applicable to the current simulator can be retrieved. The information you can get to by clicking *Verilog Netlist Option* in the Environment Options form is among the information that you can save.
>
> **Simulator Options** saves all simulator option settings. This information is simulator-specific and can be retrieved only if the same simulator is being used.

Convergence Setup saves the *Node Set*, *Initial Condition*, and *Force Node* settings, as well as restored DC and transient solutions. Convergence setup information is simulator-specific, but if both simulators support the node set functions you saved, it is possible to restore them to a different simulator.

Waveform Setup saves the Waveform window settings (for AWD and WaveScan).

Graphical Stimuli saves the setup of the graphical stimulus form.

Conditions Setup saves the settings for the Circuit Conditions form.

Results Display Setup saves the state of the *Results Display* window.You can do this by enabling the *Results Display Setup* without closing the *Results Display* window. Later, if you run the simulation again and load back the window, the new data can be loaded back and displayed as you specified it when you saved the state.

Device Checking Setup saves the device checks set up using the Device Checking Setup form during the current session.

Distributed Processing saves the DRMS commands associated with the state if you are using the distributed mode with the command option selected in the Job Submit form.

The following options appear only when the selected simulator is ams.

Solver Option saves the solver specification for the current session.

Run Option restores the run option setting as batch or interactive as specified in the Choose Run Options form.

Connect Modules saves connect rules and related information about modules, disciplines and resolution as specified in the Connect Rules Selection form.

Discipline Selection saves disciplines as specified in the Default Digital Discipline Selection form.

Global Signals saves global signals specified in the Global Signals form.

Library Files saves the library file and directory specifications made through the *Simulation – Options – Compiler – Library Files/Directories* option from ADE.

Loading State

Load State Options lets you specify that you want to load the state from a directory or from a cellview. If you select *Directory*, the fields in the *Cellview Options* group box are disabled. If you select *Cellview*, the fields in the *Directory Options* group box are disabled.

Directory Options lets you specify the *State Load Directory* from which you want to load the state. You may either type it in or specify it by using the *Browse* button. You then specify the *Library*, *Cell*, and *Simulator* that the state used. You use the *Save As* field to specify a state name. You may populate it by selecting one of the *ExistingStates* to overwrite or by specifying a new state name.

Cellview Options lets you specify a *Library* and *Cell* where you want to save the state you specify in the *State* field. You may either type these in or specify them using the *Browse* button.

What to Load controls the type of information restored.

> **Select All** restores all the types of information shown.
>
> **Clear All** does not restore any of the types of information shown.
>
> **Analyses** restores the Choosing Analyses form settings (but not simulator options that you set with the Options buttons in these forms). Analysis settings are simulator specific, so you cannot restore them from a different simulator.
>
> **Variables** restores design variables.
>
> **Outputs** restores the saved, plotted, and marched output settings. Output settings are design specific, but if the signal names match, you can restore them from a different design. You can also restore expressions from any design.
>
> **Model Setup** restores the model setup of the session that was saved.
>
> **Simulation Files** restores simulation-specific files and other information. For spectre, these files can be stimulus files, definition files, and include paths.
>
> **Environment Options** restores only the environment option settings. If you are using a different simulator in another session, you can restore only those options that are applicable to the current simulator.
>
> **Simulator Options** restores all Simulator Options form settings. This information is simulator specific.
>
> **Convergence Setup** restores the *Node Set*, *Initial Condition*, and *Force Node* settings, as well as DC and transient solutions.
>
> **Waveform Setup** restores the Waveform window settings (for AWD and WaveScan).

Graphical Stimuli restores the setup of the graphical stimulus form.

Conditions Setup restores the settings for the Circuit Conditions form.

Results Display Setup restores the state of the *Results Display* window.

Device Checking Setup restores the device checks set up using the Device Checking Setup form.

Distributed Processing restores the DRMS commands associated with the state only when you switch to the distributed mode with the command option selected in the Job Submit form. When you do this, the DRMS commands stored in the state are listed and you can select one of them.

The following options appear only when the selected simulator is *ams*.

Solver Option restores the solver specification for the current session.

Run Option restores the run option setting as batch or interactive as specified in the Choose Run Options form.

Connect Modules restores connect rules and related information about modules, disciplines and resolution as specified in the Connect Rules Selection form.

Discipline Selection restores disciplines as specified in the Default Digital Discipline Selection form.

Global Signals restores global signals specified through the Global Signals form.

Library Files restores library file and directory specifications made through the *Simulation – Options – Compiler – Library Files/Directories* option from ADE.

Editing Session Options

Session Base lets you choose the way the Virtuoso® analog design software starts up your session: open the Simulation window, display the analog circuit design environment menus on the Virtuoso Schematic window, or both.

State Save Directory identifies the directory in which the saved state file is to be copied. By default, saved state files are to be kept in the `.artist_states` directory in your home directory. You can change this path to another directory as needed.

Query to Save State lets you choose whether you want to be queried to save the state of your environment before making a change. If the option is on, you are prompted to save the state before your environment is changed. If the option is off, you can save the state manually by choosing *Session – Save State*, but you will not be prompted to do so.

Default Design Open Mode lets you choose the default open mode for your designs. If you select *edit*, your designs are opened in edit mode. If you select *read*, your designs are opened in read-only mode. You can change the mode manually by selecting *edit* or *read* for the *Open Mode* option on the Choosing Design form.

Waveform Tool lets you choose either *WaveScan* or *AWD* as the waveform viewer.

Window X Location lets you set the horizontal position of the left side of the Simulation window. A selection of 1 (the default) positions the window flush with the left side of your screen. Higher numbers move the Simulation window further to the right.

Window Y Location lets you set the vertical position of the top of the Simulation window. A selection of 1 positions the top of the window flush with the top of your screen. Higher numbers move the Simulation window further down the screen. The default positioning places the window about one third of the way down the screen.

3

Design Variables and Simulation Files for Direct Simulation

This chapter describes how you set design variables. You also learn about simulation files. Click an item in the following list for more information about that topic. This chapter is specific to direct simulations using the Spectre® circuit simulator interface.

About Direct Simulation

In the 4.4.3 and later releases, Cadence includes a simplified form of analog simulation, referred to as *direct simulation*, in its Virtuoso® Analog Design Environment tools.

In the 4.4.2 and earlier releases, the analog circuit design environment depends on socket interfaces to analog simulators. These socket interfaces use Cadence SPICE to compensate for simulator limitations. The most important of these limitations is lack of an expression evaluation capability in the simulator. Now, most analog simulators no longer have these limitations. Direct simulation is intended for these advanced analog simulators.

For the 4.4.3 and later releases, direct simulation is preferred over socket simulation. Cadence's development is focused on direct simulation. Socket simulation is given minimal development. This means that only a limited number of enhancements are made to these products, and only important bugs are fixed. Inherited connections, for example, is available for both spectre and spectreS. Direct simulation is available in the 4.4.3 release for spectre, and, of course, spectreVerilog.

Note: An important difference between direct and socket simulation is that in case of socket netlisting (with simulators such as spectreS, hspiceS and cdsSpice), AEL keywords and constants such as `boltzman`, `charge`, `degPerRad`, `epp0`, `pi`, `sqrt2`, `twoPi` and so on are evaluated and passed to the final netlist via Cadence SPICE. In case of direct netlisting (with simulators such as spectre, hspiceD and UltraSim), these keywords are treated as design variables and the netlister expects values for these variables to be defined during netlisting.

Important Benefits of Direct Simulation

Direct simulation is not identical to socket simulation. Cadence believes that the benefits justify the change:

- Improved performance in netlisting

  For a test case with 18 K components, a 5x speed improvement for first-time netlist was observed. Because netlisting for direct simulation takes full advantage of incremental netlisting, even higher improvements can be seen for second-time netlisting.

- Improved performance of simulation for spectre

  The simulator input file (equivalent of final netlist for spectreS) is not recreated for every simulation, and cdsSpice is not running. The Spectre simulator is started once and design variable changes are sent to spectre interactively. This saves simulator startup time and license checks between simulations. As a result, parametric analysis is much faster.

- Readable netlists

  In spectre direct, the netlist is truly hierarchical. The subcircuits are no longer unfolded. Subcircuit names are no longer mapped unless necessary. All numeric values in the netlist are more readable.

- Support of the preferred modeling approach that facilitates the use of standard foundry-model files

  For detailed information about the preferred modeling approach for direct simulations, see the *Direct Simulation Modeling User Guide*.

- The non-CDF libraries used in the Composer/Spectre Circuit Simulation Solution can easily be used in the analog circuit design environment, and the CDF libraries can easily be used in Composer/Spectre Circuit Simulation Solution

  If CDF libraries are used in the Composer/Spectre Circuit Simulation Solution, the compatibility flag needs to be switched on (using the UNIX environment variable `CDS_Netlist_Mode`).

- Read-only designs can be simulated, provided that they are extracted

- Improved support of standalone netlisting

  Most of the information entered in the design such as expressions and passed parameters are found in the netlist. As a result, the netlist is more useful when directly used with the Spectre circuit simulator. For example, the user can add an AC analysis to the netlist that sweeps a design variable instead of frequency. Because direct simulation results in a closer resemblance between the graphical user interface and the simulator, the Spectre manual is more useful to analog circuit design environment users.

- With the direct simulation approach, many problems are solved that could not be solved in socket simulation

  For example, multiple uses of SpectreHDL modules with subcircuits needed, SPSS analyses with hierarchical circuits, problems with sweeping model parameters in analysis statements have all been resolved. There is no longer an 8-character length limitation of design variables.

Using Direct Simulation

To use direct simulation, choose *Tools – Analog Environment – Simulation* from the Command Interpreter Window (CIW). On the simulation window set the simulator to spectre (this is now the Cadence default).

To perform a quick simulation, use the design `lowpass` in the `aExamples` library. You can find spectre model files in `*your_install_dir*/tools/dfII/samples/artist/models/spectre`.

Important Use-Model Differences between spectreS and spectre

Differences have been kept to a minimum. There are two important differences that existing spectreS users need to be aware of:

- All model files are specified by the user through the Model Library Setup form. This facilitates Cadence's preferred modeling approach for analog simulation. For more information about modeling in direct simulations, see the *Direct Simulation Modeling User Guide*.
- Schematic *Check and Save* is now recommended over schematic *Save*. Failure to use *Check and Save* may cause problems during netlisting. When the design is netlisted, the design is not extracted automatically. The benefit of automatic extraction as provided by the spectreS interface is limited. Most of a designer's schematics are read-only and must be extracted by those with adequate permissions. Hierarchical extraction may also have drawbacks. An extraction of an individual cellview with its graphical feedback on essential problems helps the user avoid many aggravating mistakes. When the user netlists in the background, automatic extraction is not possible because the executable that is in foreground has a lock on the design. This use model does enable you to simulate read-only designs. This is one of the long-term problems that has been resolved with direct simulation.

Migration from spectreS to spectre

To perform a simulation of one of your old designs, use the conversion tools. Access the conversion tools from the *Tools – Conversion Toolbox* menu in the CIW.

Design Variables and Simulation

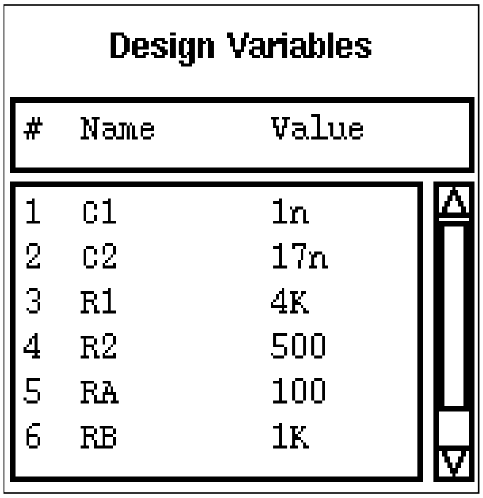
The following section describes how to work with design variables.

Setting Values

You can use design variables and CDF parameters to set component values.

Design variable values are always global to the design. The scope of CDF parameter values, however, depends on which Analog Expression Language (AEL) functions you use to refer to the parameter.

You set values for design variables in the Editing Design Variables form. Selected variables and their values appear in the lower left corner of the Simulation window. Up to 999 variables can be displayed.



Design Variables

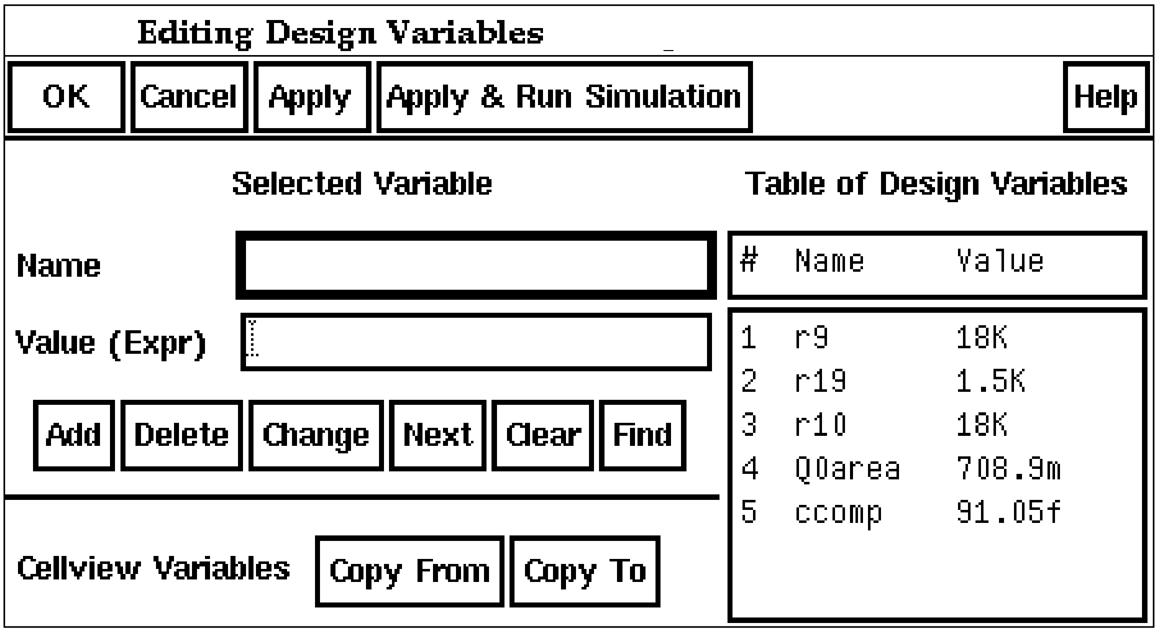
#	Name	Value
1	C1	1n
2	C2	17n
3	R1	4K
4	R2	500
5	RA	100
6	RB	1K

Adding a New Variable

To add a new variable,

1. In the Simulation window, choose *Variables – Edit*, or in the Schematic window, choose *Setup – Variables*.

The Editing Design Variables form appears.



#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

For detailed information about the form, see “Editing Design Variables” on page 137.

2. If the *Name* field already contains the name of a variable, click *Clear*.
3. Enter the variable name in the *Name* field.

   The name must begin with a letter and can contain only letters and numbers.
4. Enter a number or an expression in *Value (Expr)*.

   The expression can be an equation, a function, or another variable. Expression syntax follows AEL syntax. These expressions are evaluated by the simulator. If the value does not have any number before the decimal point, the AMS simulator inserts a leading zero before the decimal point.
5. Click *Add*.

   The new variable appears in the table on the right side of the form.

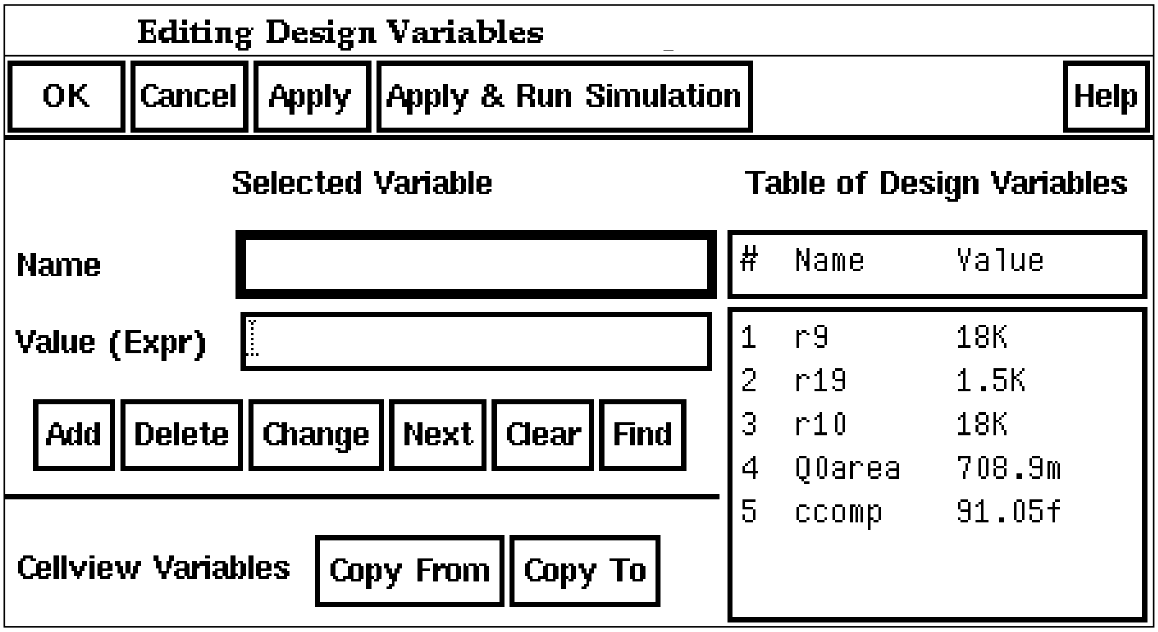
Note: You can also define variables in the definitions file.

Changing Values

To change the value of a design variables,

1. In the Simulation window, choose *Variables – Edit*, or in the Schematic window, choose *Setup – Variables*.

   The Editing Design Variables form appears.



#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

   For detailed information about the form, see “Editing Design Variables” on page 137.

2. Click the variable name in the *Table of Design Variables* list box.

   The value or expression appears in the *Value (Expr)* field.

3. Edit the value or expression.

4. Click *Change*.

5. Click *Apply* or *Apply & Run Simulation*.

Deleting Values

To delete a design variable,

1. In the Simulation window, click in the *Design Variables* list to select the variable.

   To select more than one variable, hold down the `Control` key while you click the variables, or click and drag the cursor.

   To deselect a highlighted variable, hold down the `Control` key while you click the variable.

2. Choose *Variables – Delete*.

Saving Variable Values

You can save the current variable values and later load these values back into either the simulation environment or the schematic.

To save the variable values,

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Environment – Save State*.

   The Saving State form appears.

2. Type a name for the saved simulation state.

3. Check that the *Variables* box is selected, and click *OK*.

Restoring Variable Values

To restore saved variable values,

1. In the Simulation window, choose *Session – Load State*, or in the Schematic window, choose *Analog Environment – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Select a run directory with the *Cell* and *Simulator* fields.

   The list box shows the saved states for the cell and simulator combination.

3. Click an entry in *State Name*.

4. Choose the *What to Load* options you need and click *OK*.

Copying Values between the Schematic and the Simulation Environment

If you change variables in the Schematic window and want to use these values in your next simulation,

1. Choose *Variables – Edit* in the Simulation window.

2. Click *Copy From*.

If you change variables in the simulation window and want to copy the values back to the cellview before you save the schematic,

1. Choose *Variables – Edit* in the Simulation window.
2. Click *Copy To*.

Note: Make sure that the schematic and simulation environment variables are consistent with each other.

Displaying Values on the Schematic

To display the values of instance parameters that are design variables on the schematic,

1. Edit the component's CDF with the CDF Editor.
2. Set *paramEvaluate* to *full*.

Adding Setup Files for Direct Simulation

You can add additional information to a netlist using three kinds of input files:

- The definition files, where you typically define functions and set values for global variables not part of the analog circuit design environment variables

  These files are specified through the Simulation Files Setup form.

- The model files, where you define models referenced by your design

  These files are specified through the Model Library Setup form.

- The stimulus files, used as an alternative to entering sources on a schematic or through the *Setup – Stimuli* menu

  These files are specified through the Simulation Files Setup form.

The syntax used is determined by the simulator.

In the stimulus file, you can type node names or component names in Open Simulation System (OSS) syntax `[#name]` to have the system substitute the corresponding node numbers in the netlist. You can use a backslash (\) to escape a square bracket. For more information about OSS syntax, refer to the *Open Simulation System Reference* manual, or view a sample stimulus file at *`your_install_dir`*`/tools/dfII/samples/artist/models/spectre/opampStimuli.scs`.

Note: The OSS syntax is not allowed in a definition file or model file.

Using a Definitions File

The definitions files are intended for the definition of functions and global variables that are not design variables. Examples of such variables are model parameters or internal simulator parameters.

The definitions file is loaded by the simulator when it starts.

Note: The definitions file `*your_install_dir*/tools/dfII/samples/artist/models/spectre/defaults.scs` has a number of function definitions for the Spectre circuit simulator. For example, the function `GAUSS` is defined to return the nominal value.

To specify the definitions file,

1. Create the file in the directory you specify in the include path field on the Model Setup form.
2. Choose *Setup – Simulation Files*.

   The Simulation files form opens.
3. Enter the full UNIX path of the definitions file, including any extension, in the *Definitions Files* field.

   For example, type `/cds/tools/dfII/samples/artist/spectre/models/default.scs` in the *Definitions Files* field. The simulator searches for the corresponding definitions file.

Syntax

Note: The syntax for defining functions in the definition files for the Spectre simulator is described in the *Spectre Circuit Simulator Reference*. Below is an example from the file `defaults.scs`:

```
real gauss( real mn, real std, real n) {
return mn;
}
```

A definition file might contain

- Simple passing parameters

```
real R(real l, real w) {
return (500*l/w);
}
```

- Functions returning constant values

```
real PiRho() {
return 2500;
}
real Rpi(real l, real w) {
return PiRho()*l/w;
}
```

For example, to define a poly resistor function of temperature, you can use the function definition

```
real rpoly(real value, real tdc) {
value*(1+.01*(tdc-25)+.002*(tdc-25)**2);
}
```

You can use this function when defining resistor values within your circuit. For example, the value of a resistor might be

```
rpoly(1k,tempdc)
```

You can set resistor properties `tc1` and `tc2` so that the system automatically models resistor temperature effects, rather than defining your own functions.

Definition File Example

Here is the sample `definitions.scs` file in `*your_install_dir*/tools/dfII/samples/artist/models/spectre`:

```
simulator lang=spectre
real PiRho() {
    return 2500;
}
real PbRho() {

    return 200
}
real Rpb(real l, realw) {

    return PbRho()*l/w;
}
real Rpi(l,w) {
    return PiRho()*l/w;
}
```

Stimuli Setup

There are three ways to set up stimuli in the analog circuit design environment simulator:

- Add source symbols to the schematic
- Use the Setup Analog Stimuli form
- Specify a stimulus file

Using the Setup Analog Stimuli Form

You can add stimuli to the simulator input file through the Setup Analog Stimuli form.

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!).

All sources, whether used for stimulus or for a power supply, are assumed to come from the `analogLib` library, a library supplied by Cadence. If your sources are located in a different library, you must add the *refLibs* property to your design library to identify where to find the source information. Note that global signals should be set to only DC sources.

The following procedure sets up the simulation environment for external stimuli, creates a simulator input file, and generates a stimulus file containing input and power source stimuli in the proper syntax for your simulator.

1. To access other libraries for sources, set the *refLibs* property to specify the library search sequence.

   The `analogLib` library is the default library for global sources. You do not need to set this property to use `analogLib`. If you want to use other libraries, however, use the following procedure to create the *refLibs* property and list the libraries you want to access in the appropriate sequence.

   a. From the CIW, choose *Tools – Library Manager*.

      The Library Manager: Directory form appears.

   b. Choose the library name of the current design.

   c. Choose *Edit – Properties*.

      The Library Property Editor form appears.

   d. Verify that the *refLibs* property has been set to the appropriate library search sequence.

The property appears in the lower section of the form. The libraries are searched in sequence from left to right.

e. If there is no *refLibs* entry, click *Add* on the Library Property Editor form, add the data specified below, and click *OK*.

In the Add Property form, specify the following property name and characteristics.

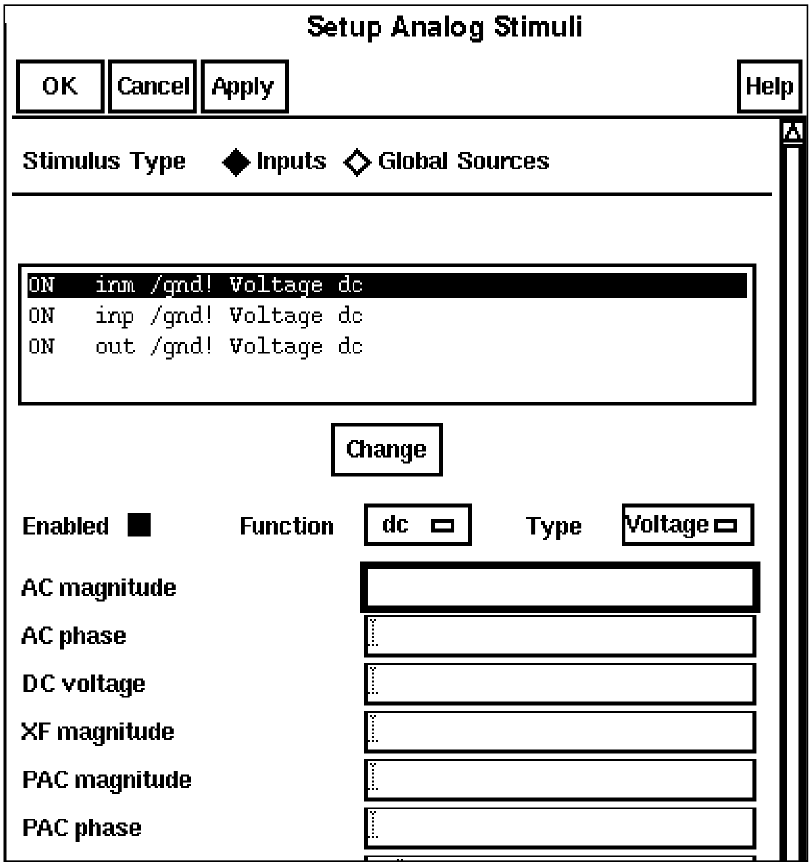
Name	*refLibs*
Type	*string*
Value	list of one or more libraries in search sequence

The *refLibs* property and the search list are displayed in the parameter list on the Library Property Editor form.

f. Click *OK* to return to the Library Manager form.

2. Choose *Setup – Stimuli* in the Simulation window to add stimuli.

The Setup Analog Stimuli form appears.



Stimulus Type ◆ Inputs ◇ Global Sources

```
ON   inm /gnd! Voltage dc
ON   inp /gnd! Voltage dc
ON   out /gnd! Voltage dc
```

Enabled ■ Function dc Type Voltage

AC magnitude

AC phase

DC voltage

XF magnitude

PAC magnitude

PAC phase

For detailed information about the form, see “Setup Analog Stimuli Form” on page 135.

3. Select the stimulus for an input signal:
   - a. Click an input pin in the list box.
   - b. Choose the appropriate *Function*.

dc=	Direct current
sin=	Sinusoidal waveform
pulse=	Pulse waveform
exp=	Exponential waveform
pwl=	Piecewise linear waveform
pwlf=	Piecewise linear waveform file
sffm=	Single frequency FM source waveform

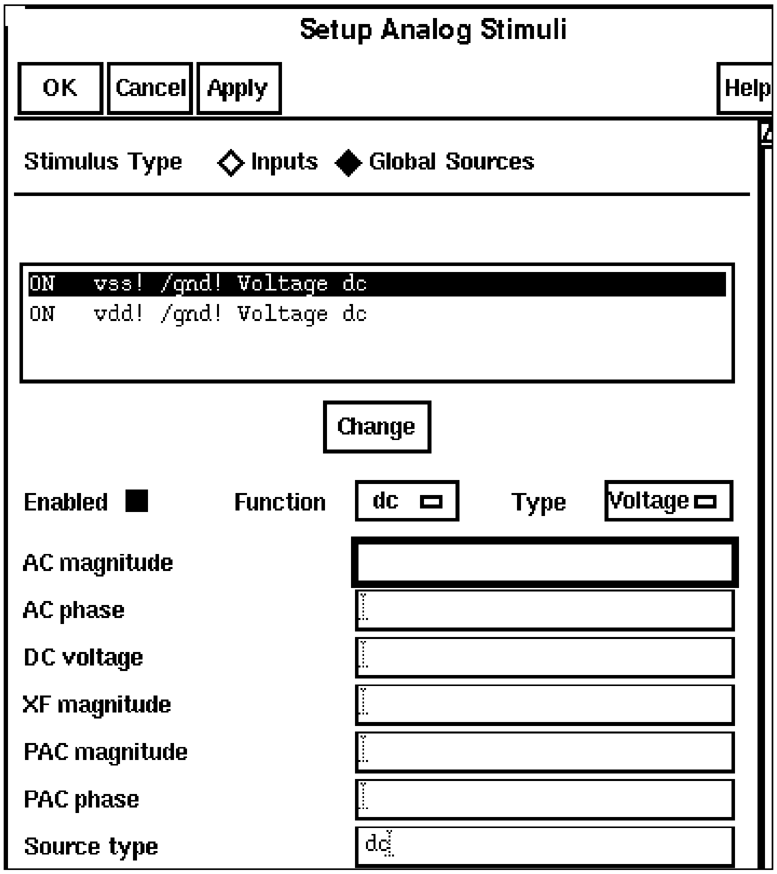
- **c.** To specify a voltage or current stimulus, select the appropriate value in the *Type* field.
- **d.** Enter new parameter values as needed in the fields below the *Function* and *Type* fields.

  The parameters displayed in this list depend on the simulator you are using. Refer to your simulator documentation for details on setting these parameters.
- **e.** Click *Change*.

  The list box displays the signal and the proper stimulus syntax for your simulator.
- **f.** Click another input pin, and repeat these steps for each pin you want to edit.
- **g.** Click *OK*.

4. Assign DC voltages to global sources:

- **a.** Select *Global Sources*.



Setup Analog Stimuli

Stimulus Type ○ Inputs ● Global Sources

```
ON   vss! /gnd! Voltage dc
ON   vdd! /gnd! Voltage dc
```

Enabled ■ Function dc Type Voltage

AC magnitude
AC phase
DC voltage
XF magnitude
PAC magnitude
PAC phase
Source type dc

All sources in your schematic that are global sources (excluding the ground signal, gnd!) are displayed in the list box.

Only DC source values should be set here.

b. Click a source in the list box.

c. Select *dc* for *Function*.

d. Enter new values as needed in the parameter fields.

The parameters displayed in this list depend on the simulator you are using. Refer to your simulator documentation for details on setting these parameters.

e. Click *Change*.

The list box displays the signal and the proper stimulus syntax for your simulator.

f. Click another source, and repeat these steps for each source you want to set.

g. To remove the voltage source for a particular global signal, select the signal in the list box and click *Enabled* to toggle it off.

The status displayed in the list box changes from `On` to `Off`.

Note that a signal that is not enabled is still used in the simulation and its connectivity is still honored by the netlister.

h. Click *OK* to close the Setup Analog Stimuli form.

The stimulus file is created automatically from the details you have entered.

Specifying a Stimulus File

Stimulus files let you add lines of code to the simulator input file that the analog circuit design environment generates. The stimulus file can be used for including input and power supply stimuli, initializing nodes, or for including estimated parasitics in the netlist.

You can specify a stimulus file on the Simulation Files Setup form.

Example of a spectre Stimulus File

The file `opampStimuli.scs` in `tools/dfII/samples/artist/models/spectre` is an example of a stimulus file that can be used for the `opamp` example in the `aExample` library.

```
simulator lang=spectre
```

```
_v1 ([#inp] 0) type=sin freq=1k ampl=1
_v2 ([#inm] 0) type=dc dc=0

TODO: show a vpwl with square bracquets
```

Model Files in the Virtuoso® Analog Design Environment

The standard way to define models is by using the simulator's native language. This is described in more detail in the *Direct Simulation Modeling User Guide*.

You can include one or more model files using the Model Library Setup form.

By convention, if the parameter `model` (with prompt *Model Name*) is set, the value is used as the component name. For example, the `Q25` component in the `opamp` schematic cellview in the `aExamples` library has a model parameter with the value `npn`. As a consequence, the netlist entry is

```
Q25 1 2 3 npn ...
```

Note that in this example, `npn` can be the name of a model definition or a subcircuit definition. Therefore, there is no distinction between components referencing models or subcircuits (also called macros).

Updating cell CDF for Direct Simulation

The CDF for a device referencing a model definition is identical to that referencing a subcircuit definition. To update the stopping cellview CDF to pass parameters into the subcircuit and set the order of the input terminals

➤ Choose *Tools – CDF – Edit* in the CIW and modify the *simInfo* section of the component CDF as follows:

Field	Value
netlistProcedure	`nil`
instParameters	The names of any CDF parameters on the component that you need to pass into the subcircuit or model
otherParameters	`model`

Field	**Value**
termOrder	The names of the symbol’s terminals, in the order you want them netlisted (the order must match the node order on the *subckt* line or that of the model referenced)

Model File Libraries

This capability is also known as .include or .lib Commands. This is handled through the Model Library Setup form.

Referencing Textual Subcircuits or Models

Textual subcircuit definitions can be referenced easily. Depending on the terminals and passed parameters used, a single cell can be used to reference either a model definition or a subcircuit definition.

To netlist the subcircuit correctly, the analog circuit design environment requires a symbol cellview, a stopping cellview, and an appropriate CDF on the cell.

Updating the Component CDF

To update the component CDF of the cell,

1. Choose *Tools – CDF – Edit* from the CIW.
2. Select *Add*.
3. Set the parameter name and attributes as follows:

Field	**Value**
name	*model*
type	*string*
prompt	*Model name*
parseAsNumber	*no*
parseAsCEL	*no*
defValue	

Note: Do not use the *Units* attribute.

4. Click *Edit* on the Simulator Information section of the Edit CDF form. The Edit Simulator Information form appears. Fill this form out according to the following table:

Field	Value
netlistProcedure	nil
instParameters	The names of any CDF parameters on the component that you need to pass into the subcircuit
otherParameters	`model`
termOrder	The order of the terminal names required for the subcircuit definition
componentName	

For example, here are the default values of the cell nmos in the analogLib library:

Field	Value
netlistProcedure	nil
instParameters	`w l as ad ps pd nrd nrs ld ls m trise region`
otherParameters	`model`
termOrder	
componentName	

Creating a Stopping Cellview

To create a stopping cellview for your simulator,

1. Edit the symbol cellview.
2. Choose *Design – Save As*.
3. Keep the same cell name, but choose a new cellview name to match your simulator. For example, for the Spectre simulator, use the cellview name *spectre*.

Using the Component

The component is used by placing an instance in the design. For example, the `analogLib` `nmos` component has many instances in the schematic named `foldedCascode` in the `aExamples` library. Note that the *Model name* field is a special field. It is the name of the subcircuit referenced. For this design, it is `nmos24`.

Including the Subcircuit File in the Netlist

The file that contains the subcircuit definition is specified through the Model Library Setup form. The syntax of the file depends on the simulator you use.

Below is a section of the spectre netlist generated for the `aExamples` `foldedCascode` example:

```
M5 (vout vref3 net32 0) nmos24 w=20u l=1.8u
M13 (vref3 vref1 vdd! vdd!) pmos24 w=40u l=3u
```

The subcircuit file `externalMos.scs` in `tools/dfII/samples/artist/models/spectre` is

```
inline subckt nmos24 (c b e s)
parameters w=1 l=1
nmos24 (c b e s) _mos l=nmos24LengthCorrection( l )
+ w=nmos24WidthCorrection( w )
model _mos mos2 type=n vto = 0.775 tox = 400e-10 nsub = 8e+15
+ xj = 0.15u ld = 0.20u uo = 650 ucrit = 0.62e+5 uexp = 0.125
+ vmax = 5.1e+4 neff = 4.0 delta = 1.4 rsh = 36 cgso = 1.95e-10
+ cgdo = 1.95e-10 cj = 195u cjsw = 500p mj = 0.76 mjsw = 0.30
+ pb = 0.8
ends
```

Note: The parameters are identical to that of a `mos2` spectre model. The same `analogLib` `nmos` cell can be used to reference a simulator model definition or a simulator subcircuit definition.

Scope of Parameters

You can use design variables and CDF parameters to set component values.

Note: Do not use callbacks on parameters whose values are expressions, particularly expressions that use pPar. The expressions might not be evaluated correctly and the system does not detect the errors. In general, try to avoid callbacks whenever possible.

Inheriting from the Same Instance: iPar()

When a parameter value must depend on the value of another parameter on the current instance, use the iPar function.

```
iPar( "CDF_parameter_name" )
```

The value of this expression is the value of this parameter on the current instance, or its value on the cell's effective CDF.

For example, suppose the parameter `AD` of a MOS transistor is a function of its channel width. You could define `AD` in the Schematic window using the *Edit – Properties* command as

```
iPar("w")*5u
```

The resulting value is the value of `w` on the instance times 5u.

The iPar expression is substituted with the value of the parameter, enclosed by parentheses, during netlisting. If no value is found, the system reports an error.

Passed Parameter Value of One Level Higher: pPar()

When a parameter expression must depend on the value of a passed parameter, use the pPar function.

```
pPar( "CDF_parameter_name" )
```

The value of this expression is the value of the passed parameter.

For example:

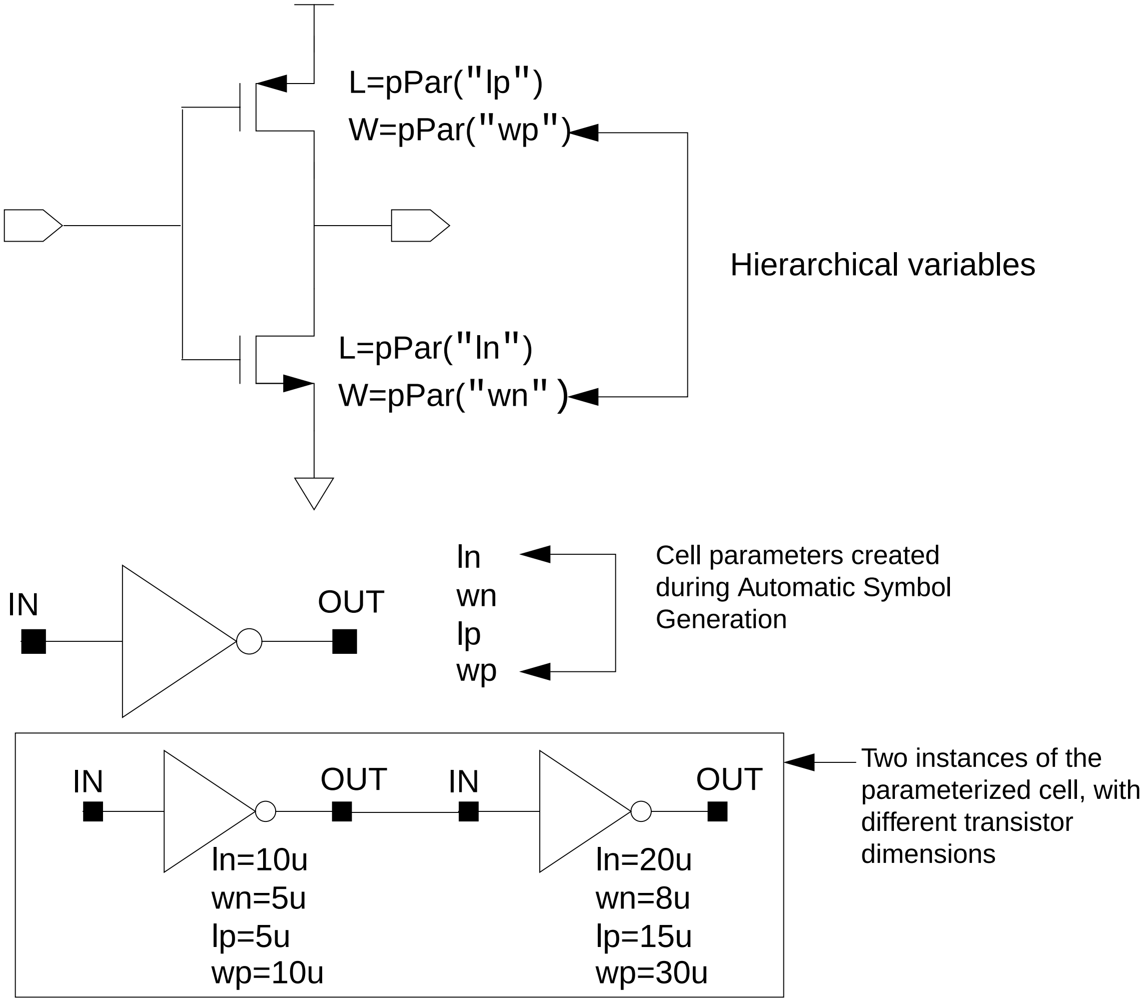
```
pPar("vss")
```

value of the "DC Voltage" parameter on the `v27` instance in the `aExamples opamp` schematic is specified for the `aExamples lowpass` schematic as `15`.

When you create new symbols using automatic symbol generation (the *Create Cellview – From Cellview* command in the Schematic window), the system creates component parameters for the parameters you defined with pPar. The following illustration gives an example of the automatic symbol generation process.

During netlisting, the pPar expression is substituted by the name of the parameter.

Using Ppar



Passed Parameters from Any Higher Level: atPar()

You should avoid using atPar. Use pPar instead.

Inheriting from the Instance Being Netlisted: dotPar()

You should avoid using dotPar. Use iPar instead.

Table of Functions

Parameters can be inherited by algebraic expressions that are used as component values. The Analog Expression Language (AEL) provides functions to control how parameters are inherited. The AEL inheritance functions are compatible with the corresponding NLP functions shown in the following table.

Functions		**Meaning**	**Scope Rules**
AEL	**NLP (OSS)**		
iPar	[~	Instance parameter	Search the instance carrying iPar, then the effective cell CDF
pPar	[+	Parent parameter	Search the parent instance, then the effective cell CDF of the parent instance

Nesting Functions

Arguments to inheritance functions can have values determined by other inheritance functions. The identity of the current instance and the parent instance are determined relative to the instance on which the current expression is stored.

For example, if an expression uses `iPar("w")` and the value of `w` is an expression that uses `iPar("l")`, `w` and `l` must both be on the same component as the original expression.

Consider the expression

```
pPar("slewRate") + 100.0
```

The value of `slewRate` might depend on inheritance functions:

```
iPar("a") + pPar("b")
```

AEL searches for `a` on the same instance (or in the same effective cell CDF) where it found `slewRate`. Thus, the search takes place in the parent of the instance where the `pPar("slewRate") + 100.0` expression was used. In turn, the system evaluates `pPar("b")` by looking for `b` on the grandparent instance.

The system detects circular references during netlisting and reports an error.

Using Inheritance Functions in Input Files

You can use inheritance functions like iPar and pPar in conjunction with built-in functions or user-defined functions.

How the Netlister Expands Hierarchy

While netlisting a hierarchical design, the analog circuit design environment expands every cell (instance) into lower level cells until it reaches a cell designated as a primitive. The primitive is then added to the netlist. This process is called design hierarchy expansion or view selection.

At each level in the hierarchy, there can be several views of each cell. You use a view list to specify which view the design environment selects and descends into. View lists can be global to the entire design or specific to an instance as specified by its property values.

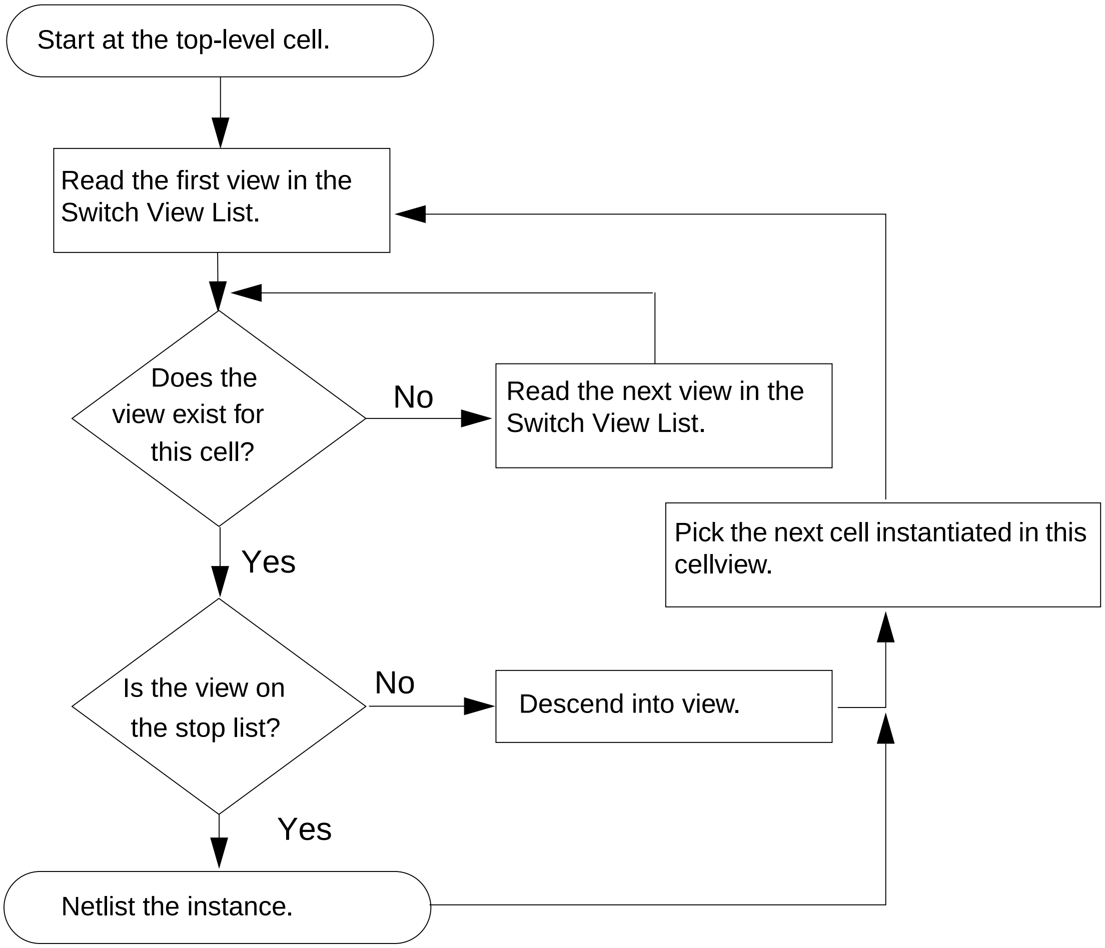
For analog simulation, you specify the global or default view list in the *Switch View List* field of the Environment Options form, when the cellview selected is not a configuration. If an instance does not have any of the views listed in the view list, the netlister reports an error.

The netlister identifies primitives with a stop list. When the netlister reaches a view that is listed in both the view list and stop list, the instance is netlisted and no further expansion occurs below this level. The global stop list is also specified on the environment options form.

For more information, see the *Hierarchy Editor User Guide*.

Note: Parasitic simulation and mixed-signal simulation use different processes for creating view lists and stop lists. Refer to Chapter 1 of the *Virtuoso Parasitic User Guide* and Chapter 7 of the *Virtuoso Mixed-Signal Simulation Interface Option User Guide* for details.

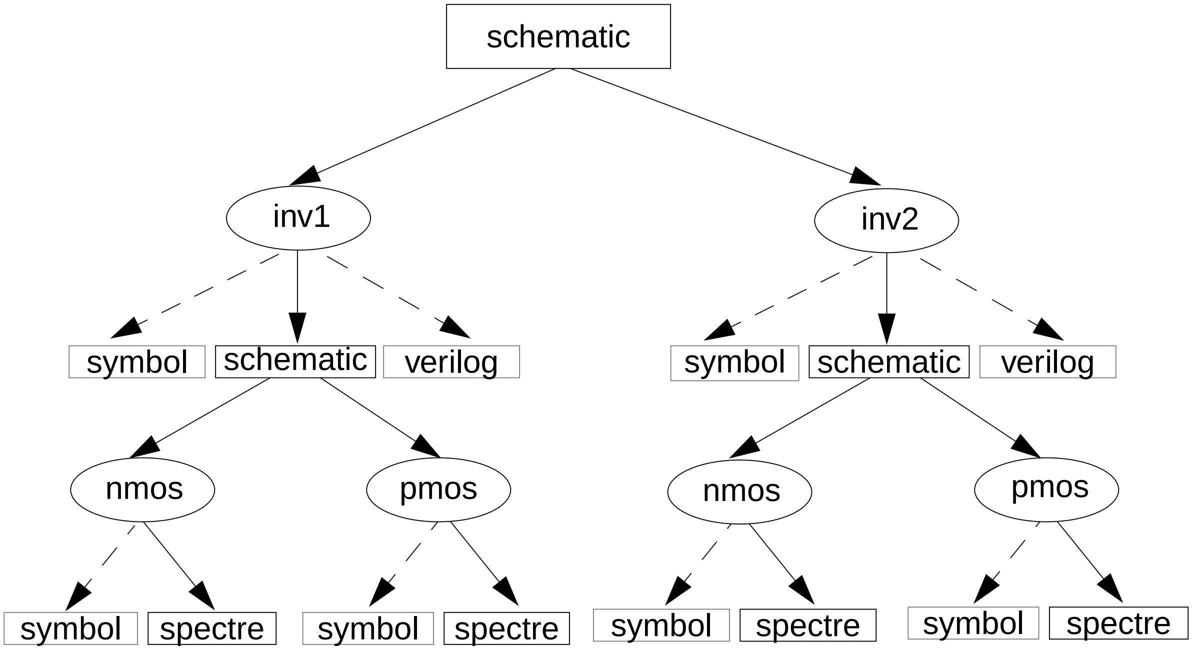
The flowchart shows how a netlister like OSS expands a design.


Start at the top-level cell.
Read the first view in the Switch View List.
Does the view exist for this cell?
No
Read the next view in the Switch View List.
Yes
Pick the next cell instantiated in this cellview.
Is the view on the stop list?
No
Descend into view.
Yes
Netlist the instance.

Netlisting Sample for Spectre

The following figure illustrates how hierarchy expansion is performed on a simple design. The solid lines show the view selection and design expansion based on the Switch View List and Stop View List that are provided.

Switch View List:	spectre schematic cmos.sch verilog
Stop View List:	spectre verilog

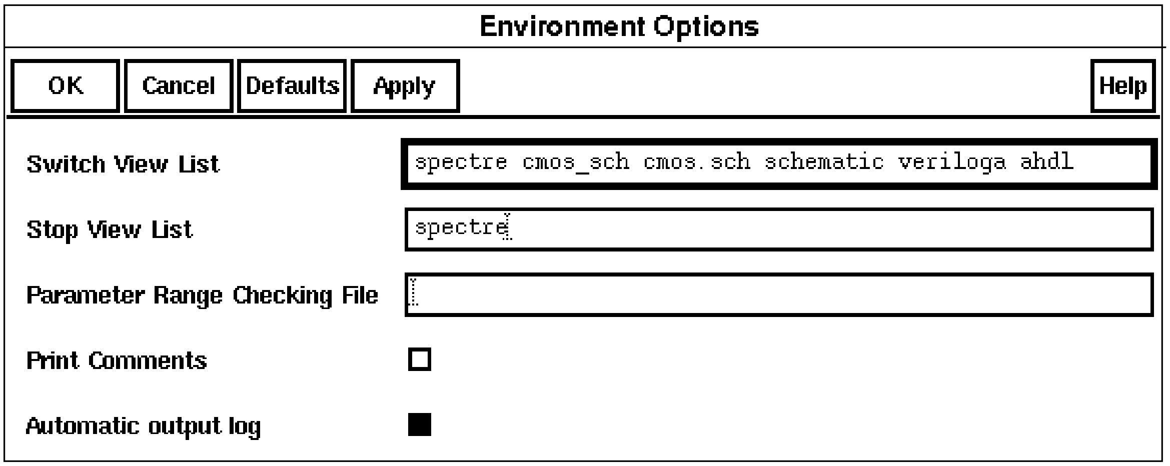


Modifying View Lists and Stop Lists

To modify a switch view list or stop view list,

1. Choose *Setup – Environment* from the Simulation window.

The Environment Options form appears.



Switch View List: spectre cmos_sch cmos.sch schematic veriloga ahdl

Stop View List: spectre

Parameter Range Checking File:

Print Comments □

Automatic output log ■

For detailed information about the form, see “Environment Options” on page 98.

2. Modify entries to the *Switch View List* field, as needed.

   The switch view list informs the netlister how to descend into different views in the design. Sequence is important in the switch view list. Refer to the flowchart to see how the netlister selects appropriate views.

3. Modify entries to the *Stop View List* field, as needed.

   In most cases, you can control netlisting adequately using the switch view list. If necessary, you can add entries to the end of the stop view list. Sequence is not important in the stop view list.

4. Click *OK* or *Apply* to add your changes.

About Netlists

The analog circuit design environment creates or updates the simulator input file automatically when you give the command to run a simulation.

You do not need to use the *Netlist – Create* command unless

- You want to use analog circuit design environment to create a netlist but run the simulator in standalone mode
- You want to modify the netlist, perhaps to take advantage of features that the design environment interface to your simulator does not support
- You want to read the netlist before starting the simulation

There are two kinds of files generated for simulation:

- The netlist, which contains component information but no simulation control data
- The simulator input file, which contains both the netlist and the simulator control information required (this file is passed to your simulator)

Netlists are hierarchical. They are created incrementally, re-netlist only the changed schematics in a design. All schematics can be forced to re-netlist by choosing *Simulation – Netlist – Recreate* from the Simulation window.

The .simrc File

You can use the `.simrc` file with Spectre during interface initialization to customize netlisting. This file helps you set or override defaults for simulation variables in such a way that the changes affect only your own simulations. The `.simrc` file is a way to set defaults on a per-user or per-system basis, and no other designer is affected by this file. This file is optional and is loaded if it exists. It is searched for in the following order:

```
$SIMRC/.simrc
$ossSimUserSiDir/.simrc
dfII/local/.simrc
Current UNIX directory/.simrc
~/.simrc
```

The search stops at the first place where the file is found. No other `.simrc` files are loaded unless the load is done from within the first one it finds, allowing for tiered loading or for the local CAD group to alter or disallow the search mechanism. If it does not exist, no error is generated. You can delete the `.simrc` file if you do not want it to be taken into account while netlisting.

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.

Creating and Displaying a Netlist

To create and display a new netlist,

➤ Choose *Simulation – Netlist – Create*.

The simulator input file is created in a file. The name of the file is `input` with the simulator-specific extension. For Spectre, the extension is `.scs`. This file is put in the following directory:

```
projectDirectory/topCellName/simulatorName/view/netlist/
        input.ext
```

projectDirectory
is the directory you specified in the Choosing Simulator/Directory/Host form.

topCellName is the root level cell name of the design.

simulatorName
is the name of the simulator.

view is the view name being netlisted.

ext is the simulator-specific extension.

To display an existing simulator input file,

➤ Choose *Simulation – Netlist – Display*.

Note: For information on variables that you can set to customize netlisting options, see the topic *Netlisting Options* in *Chapter 4* of *Virtuoso Analog Design Environment SKILL Language Reference*.

Form Field Descriptions

Setup Analog Stimuli Form

Stimulus Type

Inputs sets the stimulus for the signals with input pins in the schematic.

Global Sources lets you assign DC voltages to global signals that represent power supplies in the design.

Name identifies the signal name that is currently selected.

Library identifies the library where the selected signal or global source model was found.

Change recalculates the input voltage or current for the selected signal based on the function, type, and property values specified in the lower portions of the form. The calculated value is specified in the list box in the appropriate syntax.

Enabled lets you specify whether each signal is ON or OFF.

Function lets you choose the function for the selected signal.

> **dc** displays the direct current stimulus option properties and values.
>
> **pulse** displays the pulse stimulus option properties and values.
>
> **sin** displays the sinusoidal stimulus option properties and values.
>
> **exp** displays the exponential stimulus option properties and values.
>
> **pwl** displays the piecewise linear stimulus option properties and values.
>
> **pwlf** displays the name of the file containing piecewise linear stimulus option properties and values.
>
> **sffm** displays the single frequency FM stimulus option properties and values.

Type lets you select the voltage or current for the signal highlighted in the list box.

Parameters and their values identify the simulator-specific parameters required by your simulator. The parameters list here will vary depending on the simulator you are using. Refer to your simulator documentation for information on setting or changing these parameter values.

The form lets you set inputs and global sources for your design.

The list box contains the current netlist values of the input or bidirectional pins. Each line contains the proper syntax for your simulator.

The fields displayed below the *Function* and *Type* cyclic fields (*AC magnitude*, *AC phase*, *DC voltage*, and so forth in this example) provide parameter input specific to your simulator.

When the form is first displayed, fields in this section could be blank, could contain default values, or could contain initial values that you specified at another time.

The form changes dynamically when you select a different input pin, function, or type.

Editing Design Variables

Name is an optional name for the variable, which appears in the *Table of Design Variables* list box.

Value (Expr) is the variable value, either a number or an expression.

Add creates the variable you have specified in the *Selected Variable* area.

Delete removes a highlighted variable. Click in the list box to highlight a variable.

Change updates the highlighted variable with the new information from the *Selected Variable* area.

Next highlights the following signal or expression in the *Table of Design Variables* list box.

Clear empties the *Selected Variable* area so you can enter a new variable.

Find locates the highlighted variable in your design.

Cellview Variables lets you keep variables consistent in the simulation environment and the cellview design database by copying them back and forth.

> **Copy From** copies the variable values in the schematic cellview into the simulation environment.
>
> **Copy To** copies the variable values in the simulation environment to the schematic cellview.

Table of Design Variables identifies the name and value of each design variable in the design. Each entry is numbered for easy reference.

4

Design Variables and Simulation Files for Socket Simulation

This chapter describes how you set design variables. You also learn about simulation files. This chapter is specific to simulations using simulators integrated in the Cadence® SPICE socket.

Schematic Variables and Simulation

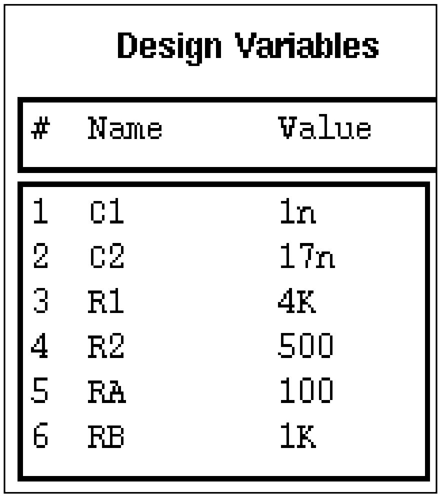
The following section describes how to work with design variables.

Setting Values

You can use design variables and CDF parameters to set component values.

Design variable values are always global to the design. The scope of CDF parameter values, however, depends on which Analog Expression Language (AEL) functions you use to refer to the parameter.

You set values for design variables in the Editing Design Variables form. Selected variables and their values appear in the lower left corner of the Simulation window. Up to 999 variables can be displayed.



Design Variables

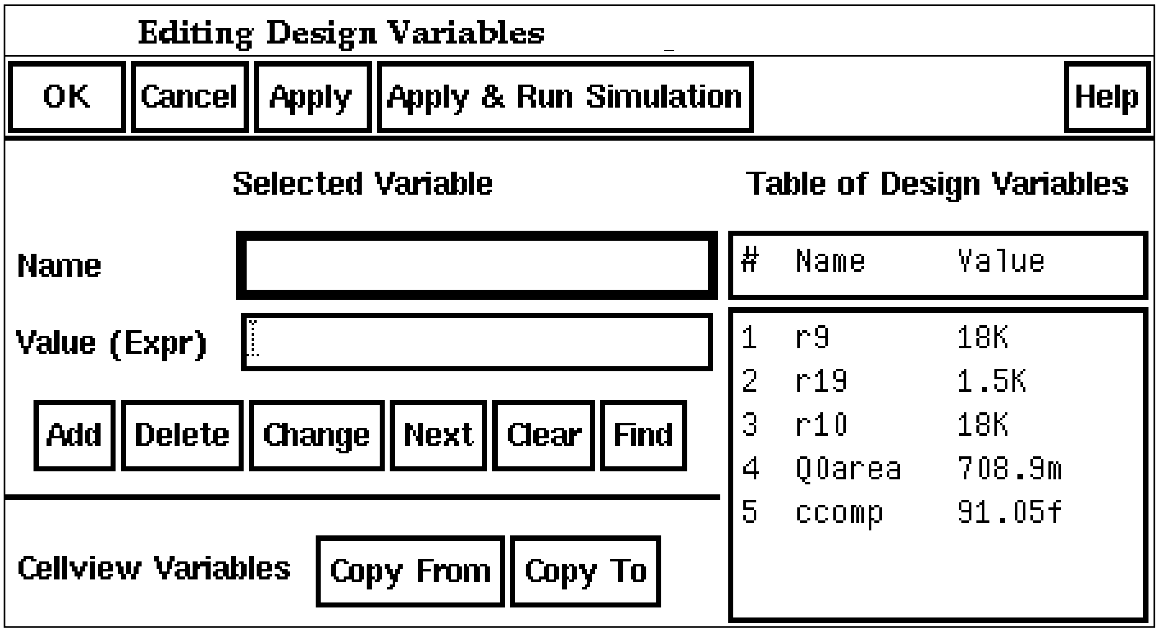
#	Name	Value
1	C1	1n
2	C2	17n
3	R1	4K
4	R2	500
5	RA	100
6	RB	1K

Adding a New Variable

To add a new variable,

1. In the Simulation window, choose *Variables – Edit*, or in the Schematic window, choose *Setup – Variables*.

The Editing Design Variables form appears.



#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

For detailed information about the form, see “Editing Design Variables” on page 184.

2. If *Name* already contains the name of a variable, click *Clear*.
3. Type the variable name in the *Name* field.

   The name must begin with a letter, contain only letters and numbers, and be no longer than eight (8) characters.
4. Type a number or an expression in *Value (Expr)*.

   The expression can be an equation, a function, or another variable. Expression syntax follows AEL syntax. The expressions are evaluated by the Cadence SPICE simulator.
5. Click *Add*.

   The new variable appears in the table on the right side of the form.

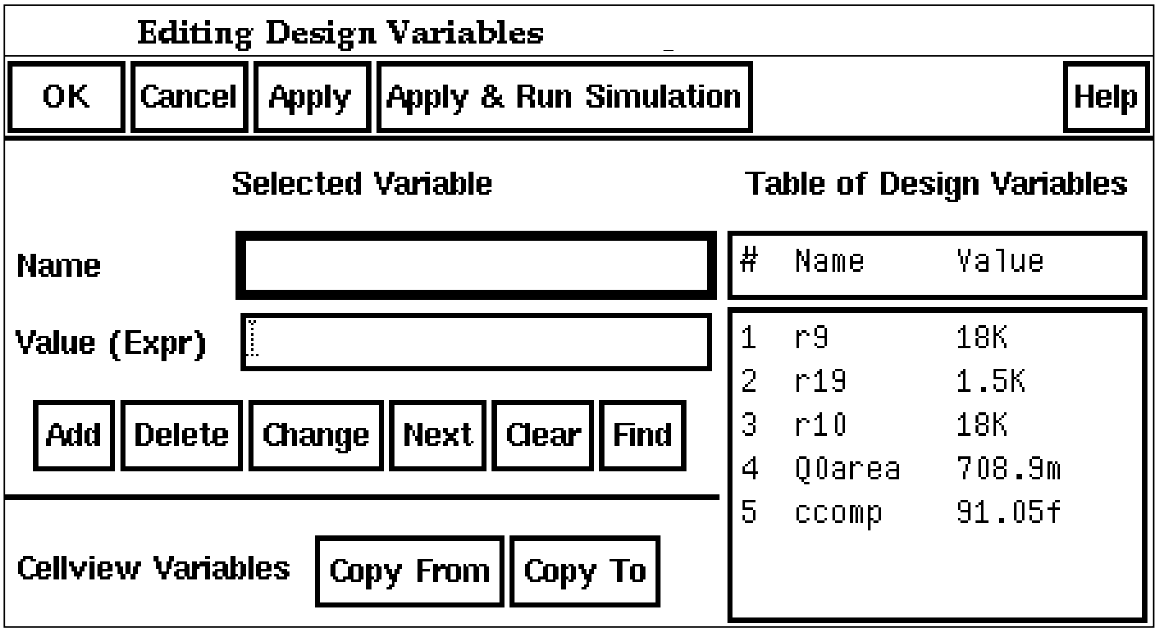
Note: You can also define variables in the Update or Init file.

Changing Values

To change the value of a design variable,

1. In the Simulation window, choose *Variables – Edit*, or in the Schematic window, choose *Setup – Variables*.

The Editing Design Variables form appears.



#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

For detailed information about the form, see “Editing Design Variables” on page 184.

2. Click the variable name in the *Table of Design Variables* list box.

   The value or expression appears in the *Value (Expr)* field.

3. Edit the value or expression.
4. Click *Change*.
5. Click *Apply* or *Apply & Run Simulation*.

Deleting Values

To delete a design variable

1. In the Simulation window, click in the *Design Variables* list to select the variable.

   To select more than one variable, hold down the `Control` key while you click the variables, or click and drag the cursor.

   To deselect a highlighted variable, hold down the `Control` key while you click the variable.

2. Choose *Variables – Delete*.

Saving Variable Values

You can save the current variable values and later load these values back into either the simulation environment or the schematic.

To save the variable values,

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Artist – Save State*.

   The Saving State form appears.

2. Type a name for the saved simulation state.
3. Check that the *Variables* box is selected, and click *OK*.

Restoring Variable Values

To restore saved variable values,

1. In the Simulation window, choose *Session – Load State*, or in the Schematic window, choose *Analog Artist – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Select a run directory with the *Cell* and *Simulator* fields.

   The list box shows the saved states for the cell and simulator combination.

3. Click an entry in *State Name*.
4. Choose one of the following restore modes:
   - ❑ Choose *append* to merge the current and saved variables, overwriting any that are duplicated in both sets.
   - ❑ Choose *overwrite* to delete all the current variables that are not in the saved set.

Copying Values between the Schematic and the Simulation Environment

If you change variables in the Schematic window and want to use these values in your next simulation,

1. Choose *Variables – Edit* in the Simulation window.
2. Click *Copy From*.

If you change variables in the simulation window and want to copy the values back to the cellview before you save the schematic

1. Choose *Variables – Edit* in the Simulation window.
2. Click *Copy To*.

Note: Make sure that the schematic and simulation environment variables are consistent with each other.

Displaying Values on the Schematic

To display the values of instance parameters that are design variables on the schematic,

1. Edit the component's CDF with the CDF Editor.
2. Set *paramEvaluate* to *full*.

Adding Analysis Commands to Netlist

You can add additional information to a netlist using three kinds of input files:

- The init file, where you typically define functions
- The update file, where you typically set values for global design variables
- The include or stimulus file, which can include any additional information

You can use Cadence SPICE syntax in init and update files. Cadence SPICE translates the netlist information into the native syntax of the destination simulator. You can use your target simulator language in include and stimulus files.

You can type node names or component names in OSS syntax `[#name]` to have the system substitute the corresponding node numbers in the netlist.

Note: For compatibility with previous releases, you can use two levels of include files to bypass the Cadence SPICE interface and add information in native syntax for another simulator.

Note: Socket Netlister adds a + sign by default for design variables and changes this only when the user specifies a - sign for them. This is done irrespective of the type and use model of design variables.

Using an Init File

The init file is where you typically define functions that appear in design values, model parameters, or internal simulator parameters.

The commands in the init file are run when you initialize the simulation or when you change the path or init filename.

To specify the init filename,

1. Create the file in the directory you specify on the *Model Path* in the Environment Options form.
2. The filename must end in `.s`, such as `init.s`.
3. Choose *Setup – Environment*.

   The Environment Options form appears.
4. Type the filename, without the `.s` suffix, in the *Init File* field.

   For example, type `init` in *Init File*. The system searches for the corresponding `init.s` file.

Syntax

The syntax for defining functions in the init file is

```
FUNCTION function_name(passing values)=Mathematical Expression
with values
```

If you do not want to pass variables into the user-defined function, the syntax is

```
FUNCTION function_name(0)=Mathematical Expression
```

Note: The `FUNCTION` command is defined in the *Cadence SPICE Reference Manual*. You can include any Cadence SPICE functions in the init file.

Example Functions

An init file might contain:

- Simple passing parameters

  ```
  FUNCTION   R(l,w)=(500*l/w)
  ```

- Passing parameters with a variable from an update file or with design variables

```
FUNCTION   Rpi(l,w)=(PiRho*l/w)
```

- Parameters with mathematical expressions using variables from the Simulation Options forms

```
FUNCTION   Nbf(0)=NpnBf*(TNOM/TEMPDC)**1.5
```

For example, to define a poly resistor function of temperature, you can create an `init.s` file containing the following command:

```
FUNCTION rpoly(value)=value*(1+.01*(tempdc-
    25)+.002*(tempdc-25)**2)
```

You can use this function when defining resistor values within your circuit. For example, the value of a resistor might be

```
rpoly(1k)
```

Init Example

Note: You can set resistor properties *tc1* and *tc2* so that the system automatically models resistor temperature effects, rather than defining your own functions.

Here is a sample `init.s` file:

```
FUNCTION Rpb(l,w)=(PbRho*l/w)*GAUSS(1,0.01,1)
FUNCTION Rpi(l,w)=(PiRho*l/w)*GAUSS(1,0.01,1)
FUNCTION Nbf(0)=NpnBf*GAUSS(1,0.02,1)
FUNCTION Pbf(0)=PnpBf*GAUSS(1,0.02,1)
FUNCTION Nis(0)=NpnIs*GAUSS(1,0.03,1)
FUNCTION Pis(0)=PnpIs*GAUSS(1,0.03,1)
FUNCTION CGA(c,sd)=GAUSS(c,sd,1)
```

Using an Update File

The update file contains commands that are run each time you run a simulation. You can use this file to set values for global variables, but it is better to set the values in the Editing Design Variables form or in the *Design Variables* section of the Simulation window. The update file is the last file the simulator reads before each simulation.

Note: Variable values in the update file override values you set in the Editing Design Variables form.

To specify the update filename,

1. Put the file in the directory you specify on the *Model Path* in the Environment Options form.

   The filename must end in `.s`, such as `update.s`

2. Choose *Setup – Environment*.

3. Type the filename without the `.s` suffix in the *Update File* field.

   For example, type `update` in the *Update File* field. The system searches for the corresponding `update.s` file.

Update Examples

Instead of using the `rpoly` function as shown in the example for the init file, you can describe resistor temperature dependence in the update file with a resistor temperature coefficient variable `rtempco`:

```
set rtempco=1+.01*(tempdc-25)+.002*&(tempdc-25)**2
```

You can assign a value of *`rtempco`*`*1k` to a resistor in your schematic. `tempdc` is a built-in Cadence SPICE variable.

Here are some other examples:

```
set PiRho=GAUSS(2500,2500*0.1,1)
set vgain=100meg
```

Setting Spectre Options

When you set Spectre simulator options in an init or update file, you use the cdsSpice `ptprop` or `psprop .s` files. This is an exception to the normal syntax rules for these files.

The syntax for specifying numerical options is

```
ptprop keyword option_name value
```

The syntax for specifying string options is

```
keyword option_name "value"
```

The keywords are

Keyword	Type of Spectre Option
`Spectre_Opt`	General options
`Spectre_Tran`	Transient options

Keyword	Type of Spectre Option
`Spectre_DC`	DC options
`Spectre_AC`	AC options
`Spectre_DC_op`	DC operating point options

Options Examples

Below are some examples showing how to set Spectre options in an init or update file.

```
ptprop Spectre_Opt reltol .001
psprop Spectre_Opt flow "I"
ptprop Spectre_Tran cmin 0
ptprop Spectre_DC maxiters 150
psprop Spectre_AC annotate "status"
ptprop Spectre_Tran maxsteps 10000
```

Making Init or Update Files Compatible with Other Simulators

You can make your init and update files compatible with both Spectre and other simulators by including both Spectre and Cadence SPICE option commands in the file.

This example sets the `reltol` option for both the Cadence SPICE and Spectre simulators:

```
set reltol=1e-4
ptprop Spectre_Opt reltol .0001
```

Stimuli Setup

There are three ways to set up stimuli in the Virtuoso® Analog Design Environment simulator:

- Add source symbols to the schematic
- Use the Setup Analog Stimuli form
- Specify a stimulus file

Stimulus and include files let you add lines of code to the netlist that the analog circuit design environment generates. The system treats information in either type of file the same way. The stimulus file can be used for initializing nodes or for including input and power supply stimulus or estimated parasitics in the netlist.

You can create a stimulus file in the analog circuit design environment using the *Setup – Stimulus – Edit Analog* menu selection from the Simulation window. This option displays the Edit Stimulus File form that prompts you for the method of implementation. You can choose to create the stimulus file yourself or to fill in a form and have the file created automatically.

To create an include file, choose *Setup – Include – Edit Analog* in the Simulation window. The procedure for creating and modifying an include file is similar to that for creating a stimulus file using a text editor.

The Analog Stimuli Form

Through the 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!).

All sources, whether used for stimulus or for a power supply, are assumed to come from `analogLib`, a library supplied by Cadence. If your sources are located in a different library, you must add the *refLibs* property to your design library to identify where to find the source information. Note that global signals should be set to only DC sources.

The following procedure sets up the simulation environment for external stimuli, creates a netlist, and generates a stimulus file containing input and power source stimuli in the proper syntax for your simulator.

1. Make sure that you specify the file syntax in the Environment Options form.

    a. In the Simulation window, choose *Setup – Environment*.

       The Environment Options form appears.

    b. In the *Include/Stimulus File Syntax* field, choose the target simulator.

    c. For any simulator that uses the Cadence SPICE socket (like hspiceS or spectreS), you can select either the target simulator or cdsSPICE syntax.

    d. Click *OK*.

2. To access other libraries for sources, set the *refLibs* property to specify the library search sequence.

The `analogLib` library is the default library for global sources. You do not need to set this property to use `analogLib`. If you want to use other libraries, however, use the following procedure to create the *refLibs* property and list the libraries you want to access in the appropriate sequence.

a. From the CIW, choose *Tools – Library Manager*.

The Library Manager: Directory form appears.

b. Choose the library name of the current design.

c. Choose *Edit – Properties*.

The Library Property Editor form appears.

d. Verify that the *refLibs* property is set to the appropriate library search sequence.

The property appears in the lower section of the form. The libraries are searched in sequence from left to right.

e. If there is no *refLibs* entry, click *Add* on the Library Property Editor form, add the data specified below, and click *OK*.

In the Add Property form, specify the following property name and characteristics.

Name	*refLibs*
Type	*string*
Value	List of one or more libraries in search sequence

The *refLibs* property and the search list are displayed in the parameter list on the Library Property Editor form.

f. Click *OK* to return to the Library Manager form.

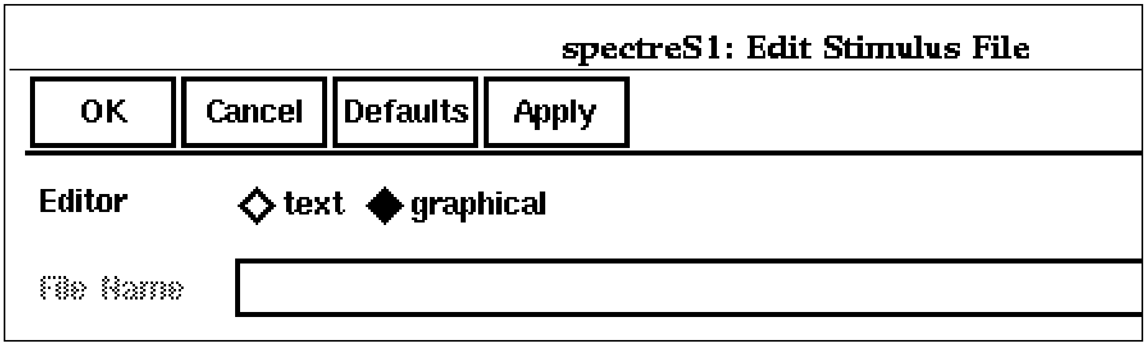
3. Create a raw or final netlist.

In the Simulation window, choose *Simulation – Netlist – Create*.

For socket interfaces, in the Simulation window, choose *Simulation – Netlist – Create Raw* or *Simulation – Netlist – Create Final*.

4. Choose *Setup – Stimulus – Edit Analog* in the Simulation window to create the stimulus file.

The Edit Stimulus File form appears.



spectreS1: Edit Stimulus File

Editor ◇ text ◆ graphical

File Name

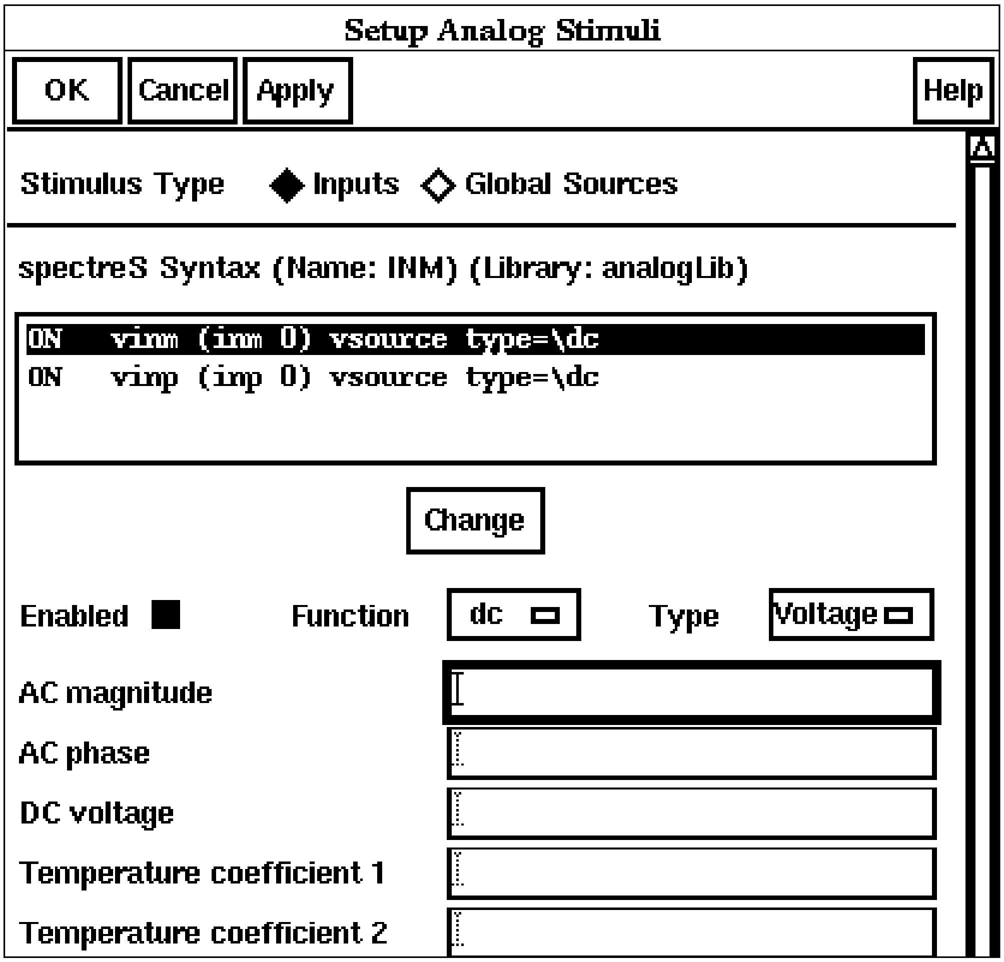
Text mode opens a text editor window where you can type in input commands in the appropriate syntax for your simulator.

Graphical mode creates the stimulus file automatically from details you supply in the Setup Analog Stimuli form.

5. Choose *graphical* and click *OK*.

Note: The *Setup Analog Stimuli* form can also be invoked by selecting *graphical* and clicking on *Apply*. In this case, the *Edit Stimulus File* form will remain open even after you click on *OK* in the *Setup Analog Stimuli* form. You need to click on *Cancel* to close it.

The Setup Analog Stimuli form appears.



Setup Analog Stimuli

Stimulus Type ◆ Inputs ◇ Global Sources

spectreS Syntax (Name: INM) (Library: analogLib)

```
ON   vinm (inm 0) vsource type=\dc
ON   vinp (inp 0) vsource type=\dc
```

Enabled ■ Function dc Type Voltage

AC magnitude

AC phase

DC voltage

Temperature coefficient 1

Temperature coefficient 2

For detailed information about the form, see “Setup Analog Stimuli” on page 182.

6. Select the stimulus for an input signal:

a. Click an input pin in the list box.

b. Choose the appropriate *Function*.

dc=	Direct current
sin=	Sinusoidal waveform
pulse=	Pulse waveform
exp=	Exponential waveform
pwl=	Piecewise linear waveform
pwlf=	Piecewise linear waveform file

sffm= Single frequency FM source waveform

c. To specify a voltage or current stimulus, choose the appropriate value in the *Type* field.

d. Type new parameter values as needed in the fields below the *Function* and *Type* fields.

The parameters displayed in this list depend on the simulator you are using. Refer to your simulator documentation for details on setting these parameters.

e. Click *Change*.

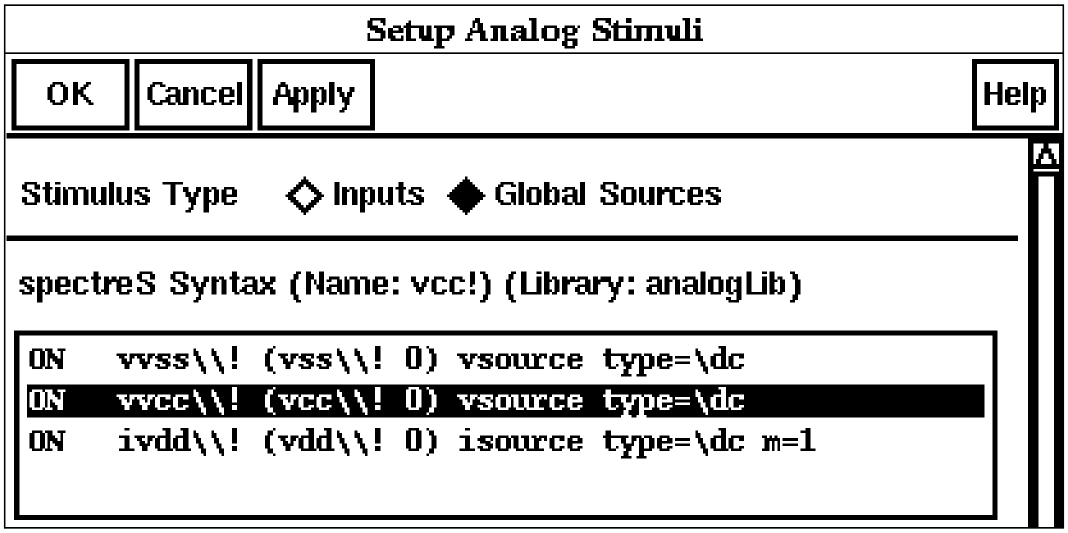
The list box displays the signal and the proper stimulus syntax for your simulator.

f. Click another input pin, and repeat these steps for each pin you want to edit.

g. Click *OK*.

7. Assign DC voltages to global sources:

a. Choose *Global Sources*.



Stimulus Type ◇ Inputs ◆ Global Sources

spectreS Syntax (Name: vcc!) (Library: analogLib)

```
ON   vvss\\! (vss\\! 0) vsource type=\dc
ON   vvcc\\! (vcc\\! 0) vsource type=\dc
ON   ivdd\\! (vdd\\! 0) isource type=\dc m=1
```

All sources in your schematic that are global sources (excluding the ground signal, gnd!) are displayed in the list box. Set only DC source values here.

b. Click a source in the list box.

c. Choose *dc* for *Function*.

d. Type new values as needed in the parameter fields.

The parameters displayed in this list depend on the simulator you are using. Refer to your simulator documentation for details on setting these parameters.

e. Click *Change*.

The list box displays the signal and the proper stimulus syntax for your simulator.

f. Click another source, and repeat these steps for each source you want to set.

g. To remove the voltage source for a particular global signal, select the signal in the list box and click *Enabled* to switch it off.

The status displayed in the list box changes from *On* to *Off*.

Note that a signal that is not enabled is still used in the simulation and its connectivity is still honored by the netlister.

h. Click *OK* to close the Setup Analog Stimuli form.

The stimulus file is created automatically from the details you enter.

Creating a Stimulus File Using a Text Editor

To create a stimulus or include file using the default text editor,

1. Make sure the target simulator is specified in the Environment Options form.

a. In the Simulation window, choose *Setup – Environment*.

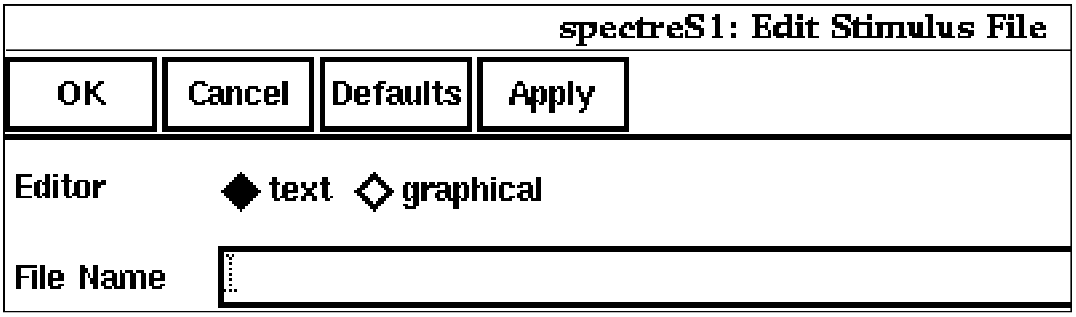
The Environment Options form appears.

b. Choose the target simulator in the *Include/Stimulus File Syntax* field, and click *OK*.

For any simulator that uses the Cadence SPICE socket (like hspiceS or spectreS), you can choose either the target simulator or cdsSPICE.

2. In the Simulation window, choose *Setup – Stimulus – Edit Analog* or *Setup – Simulation Files – Include File*.

3. If you are creating a stimulus file, click *text* for the text editor format.



Editor ◆ text ◇ graphical

File Name

4. Type a filename in the *File Name* field.

   To put the file in the standard location in the run directory, type in just the filename. (It is not necessary to type in a path.)

   To put the file in a different location, type in the complete path.

5. Click *OK* to open the file in your default editor.

6. Write the file in the appropriate syntax.

   Use the netlist syntax for the target simulator selected on the Environment Options form.

   For any simulator that uses the Cadence SPICE socket (like hspiceS or spectreS), you can type in commands in your target simulator syntax or in cdsSpice syntax. The syntax must match the target simulator specified on the Environment Options form.

   If you choose cdsSpice for the include or stimulus file syntax, Cadence SPICE processes the include or stimulus file.

   The system translates the statements into your simulator netlist format, but you can type in node names using the format `[#name]` to have the system substitute the corresponding node numbers.

7. Quit the text editor.

Managing Edited Files

Path Specification

If you type in just a filename without a path when you create a stimulus or include file, the system puts the file in the following simulation run directory:

`projectDirectory/topCellName/simulatorName/schematic/netlist/fileName`

projectDirectory

The directory you specified in the Choosing Simulator/Directory/Host form.

topCellName The root-level cell name of the design.

simulatorName The name of the simulator.

schematic The view name being netlisted.

fileName The default name for your stimulus or include file. You can also specify a name for the file here.

If you want to put a stimulus or include file somewhere other than in the simulation run directory, you can specify an absolute path, such as your home directory:

~/fileName

Note: If you put a stimulus or include file somewhere other than in your simulation run directory, you must specify the path to that file in the *Include File* field of the Environment Options form.

Setup Considerations

If you choose *cdsSpice* in the *Include/Stimulus File Syntax* field and your file is in the target simulator syntax, you need to have two levels of include files. This choice is compatible with previous versions of the analog circuit design environment.

The first file contains a single `include` command for Cadence SPICE that specifies the name of the second file. This `include` command is added to the raw netlist.

The second file is made up of commands using the target simulator syntax. The commands in this file are added to the final netlist that is sent to the target simulator.

Examples of Edited Files

Using Analysis Commands the Virtuoso® Design Environment Does Not Support

Your simulator might use commands that the analog circuit design environment interface does not support. If your simulator is integrated through the Cadence SPICE socket, you can specify the target simulator in the *Include/Stimulus File Syntax* field on the Environment Options form.

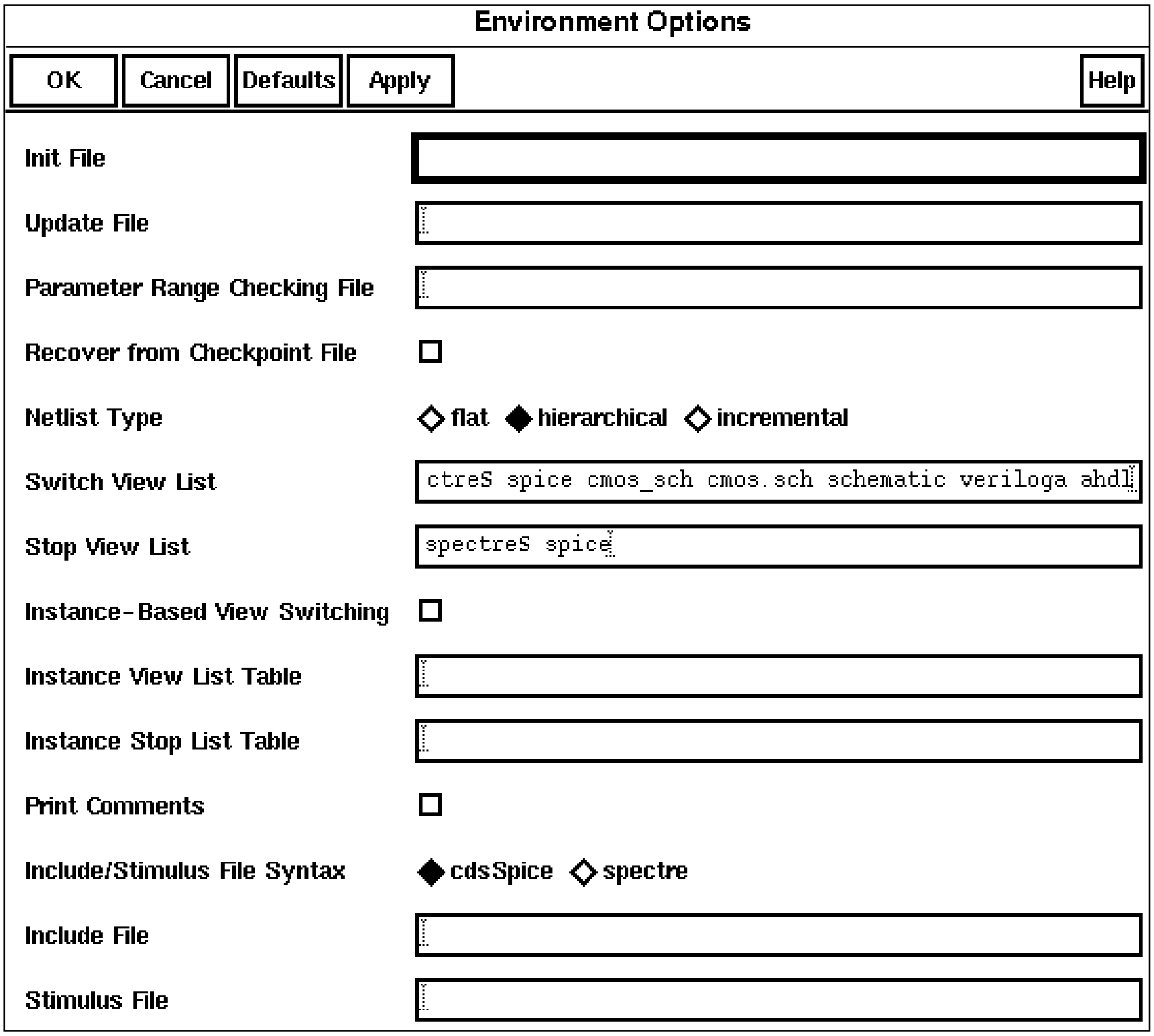
1. Create one or two files in the directory

   projectDirectory/topCellName/simulatorName/schematic/
   ```
       netlist
   ```

   The target simulator file contains the commands in the syntax of the simulator you are using.

2. Choose *Setup – Environment*.

The Environment Options form appears.



Environment Options

Init File	
Update File	
Parameter Range Checking File	
Recover from Checkpoint File	□
Netlist Type	○ flat ● hierarchical ○ incremental
Switch View List	ctreS spice cmos_sch cmos.sch schematic veriloga ahdl
Stop View List	spectreS spice
Instance-Based View Switching	□
Instance View List Table	
Instance Stop List Table	
Print Comments	□
Include/Stimulus File Syntax	● cdsSpice ○ spectre
Include File	
Stimulus File	

3. Select the target simulator name in the *Include/Stimulus File Syntax* field.
4. Type the name of the file in *Include File*.

A Stimulus File for the Spectre Interface

Stimulus files let you add lines of code to the simulator input file that the analog circuit design environment simulator generates. You can use the stimulus file for including input and power supply stimuli, initializing nodes, or for including estimated parasitics in the netlist.

You can specify a stimulus file on the Simulation Files Setup form.

For example:

```
vxx1 [#/B0] [#/gnd!] vsource type=pwl
file="/usr/mnt/user/pwlf1"
```

```
vxx2  [#/B2_] [#/gnd!] vsource type=pwl
file="/usr/mnt/user/pwlf2"
```

Using Node Set Commands in Subcircuits

A stimulus or include file can contain node set commands. In this hspiceS example, the `~/lowpass_nodeset` commands are entered in the include file.

```
.nodeset v(XI1.net433)=-12.98
.nodeset v(out)=1.0
.nodeset v(net15)=.999
```

For flat netlisting, the first line is modified to

```
.nodeset v[#I1/net433]=-12.98
```

Using Spectre Syntax in a Stimulus File

If you want cdsSpice to parse a file in Spectre syntax in a stimulus file, use two backslashes (\\) before all words that the Cadence SPICE simulator should not evaluate as variables.

For example:

```
vxx1 [#/B0] [#/gnd!] \\vsource type=\\pwl
file="/usr/mnt/user/pwlf1"
```

```
vxx2  [#/B2_] [#/gnd!] \\vsource type=\\pwl
file="/usr/mnt/user/pwlf2"
```

Model Files in Socket Simulations

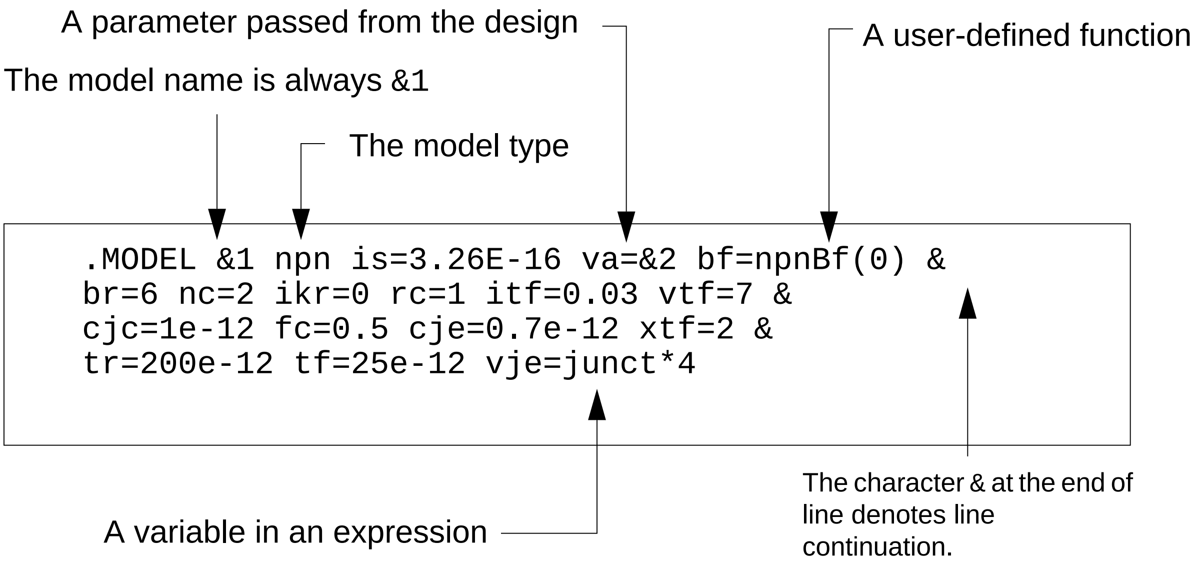
The analog circuit design environment simulator expects to find model statements that follow Cadence SPICE syntax in files named `device.m` in one of the directories in the model path. You specify the model directory search path in the Environment Options form.

Cadence SPICE syntax for models differs from Berkeley SPICE syntax in three important ways:

- The ampersand (&) continuation character appears at the end of the line.

- The `&1` argument passes in the model name. This syntax lets the analog circuit design environment generate multiple models from the same model file, as is necessary for Monte Carlo analyses.
- You can use variables and functions in the model statement.

You can use global variables, Cadence SPICE functions, or user-defined functions to set device parameter values in model files. (Monte Carlo analyses have several examples.) The example model statement below demonstrates each of these capabilities.



A parameter passed from the design

A user-defined function

The model name is always `&1`

The model type

```
.MODEL &1 npn is=3.26E-16 va=&2 bf=npnBf(0) &
br=6 nc=2 ikr=0 rc=1 itf=0.03 vtf=7 &
cjc=1e-12 fc=0.5 cje=0.7e-12 xtf=2 &
tr=200e-12 tf=25e-12 vje=junct*4
```

The character & at the end of line denotes line continuation.

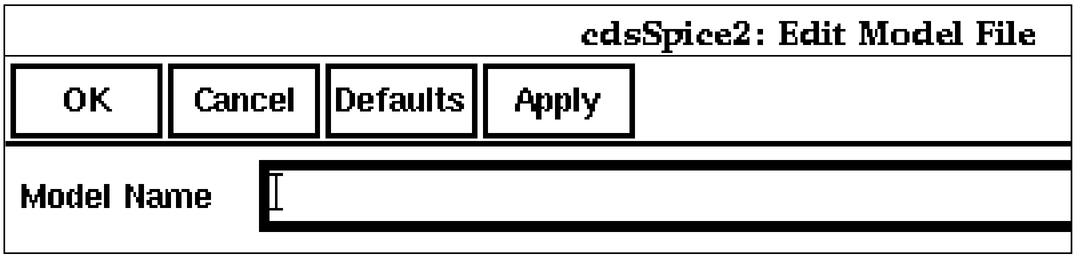
A variable in an expression

Editing Model Files in the Models Directory

To edit a model file,

1. Choose *Setup – Simulation Files – Model File*.

The Edit Model File form appears.



Model Name is the name of a file that contains the model of a nonlinear device. The name always ends with the `.m` extension.

2. Type the name of the model file you want to edit.

   Model filenames must end in `.m`.

   The system looks for the file in the directories you entered in the *Model Path* field of the Environment Options form. You specify the model directory search path in the Environment Options form.

   If the file does not exist, the system creates it.

3. Click *OK* to begin editing the file in `vi`.

4. Write the file and quit the text editor.

Example Model File

Below is the model file for a device named `nmos`. The file is named `nmos.m`.

```
.model &1 nmos &
```

```
level=2 vto=0.9 kp=3.0e-5 gamma=1.36 phi=0.747 &
cgso=5.2e-10 cgdo=5.2e-10 cgbo=1.47e-10 cj=3.2e-4 &
cjsw=9e-10 js=1e-4 tox=5e-8 nsub=5e16 tps=1 &
xj=4e-7 ld=4e-7 uo=450 ucrit=8e4 uexp=0.15 mj=0.5 & rsh=20 utra=0.3
kf=6e-24 lambda=0.02 vmax=5e4 & xqc=0.49
```

Note: Editing a model file during an analog circuit design environment session is not recommended. The netlister does not recreate the netlist if a model is changed.

Using Model Files in Native Syntax In Socket Simulations

You can also use models written in the native syntax of a simulator. There are two ways to accomplish this:

- Using `.lib` or `.include` commands in an include file
- Using backslashes (\) to prevent Cadence SPICE from evaluating parameters as variables.

Nested Include Files

If your simulator is integrated through the Cadence SPICE socket, you do not need to convert the models into Cadence SPICE syntax.

1. Create a file in the directory.

   `runDirectory/designName/simulatorName/``schematic/netlist`

   The file contains the model statements (or any other netlist information) for your simulator.

2. Choose *Setup – Environment*.

   The Environment Options form appears.

3. Type the name of the file in *Include File*.

Using .include or .lib Commands in Include Files

You can also use a single include file that contains `.lib` or `.include` commands (if your simulator supports them).

1. Create a file in the following directory:

   `runDirectory/designName/simulatorName/schematic/``netlist`

2. Add `.lib` or `.include` commands as needed for your models.

   You can put the following commands in a single include file:

   ```
   .lib '/home/user3/myModels/nom' nmos
   .lib '/home/user3/myModels/nom' pmos
   ```

3. Choose *Setup – Environment*.

   The Environment Options form appears.

4. Type the name of the file in *Include File*.

The SPICE socket interface expects to find models in `.m` files. When you run the simulation, you see error messages about missing model files, but the simulation still runs. If the models are in the include file, you can ignore the error messages. To prevent these messages, create empty `.m` files for each model and put them in the models directory.

Using Backslashes

When Cadence SPICE interprets model files in the native syntax for other simulators, it treats unrecognized words (commands, options, and values) as variables and tries to evaluate them. Putting a backslash (\) in front of a word prevents Cadence SPICE from evaluating the word. To do this, you must edit the model statement.

For example, if you are running the Spectre® circuit simulator in the Cadence SPICE socket, this Spectre model statement

```
.model myMod myName type=n
```

yields the error message

```
PARAMETER HAS NOT BEEN SET: n
```

If you add a backslash before the n

```
.model myMod myName type=\n
```

Cadence SPICE passes the correct model statement to the Spectre simulator:

```
.model myMod myName type=n
```

About Subcircuits and Macros

The analog circuit design environment can netlist hierarchical blocks in two ways:

- Add a user-written macro (subcircuit) file to the netlist
- Generate a subcircuit netlist from the schematic that corresponds to the block

You can control how the analog circuit design environment netlists each block and each instance of the block.

Note: This material applies only to simulators integrated through the Cadence SPICE socket.

How Subcircuits Are Named

Whenever possible during hierarchical netlisting, subcircuit names are generated in the form

```
_libName_cktName_schematic
```

For example:

```
.SUBCKT _mylib_bicomp_schematic n0 n1 HSBIAS OUT
```

This name generation is not possible when a subcircuit is parameterized (that is, when it has parameter values passed in through macroArguments in the stopping cellview CDF). In this situation, subcircuits are named `sub`*X*, where *X* is a unique number. For example:

```
.SUBCKT sub3 D G S
```

If you want to have nonparameterized subcircuits named in the `sub`*X* style, add a dummy parameter to the *macroArgument* field of the CDF for the stopping cellview.

Creating the Component CDF and a Stopping Cellview

To netlist the subcircuit correctly, the analog circuit design environment requires a symbol and stopping cellview with appropriate CDF. Follow these steps to update the symbol cellview and create a stopping cellview:

1. Update the component CDF by using the CDF Editor.
2. Copy the symbol cellview to create a stopping cellview for your simulator.
3. Modify the stopping cellview CDF to pass parameters to the subcircuit and tell the analog circuit design environment to netlist it as a SPICE-type subcircuit. You make these changes to the *simInfo* section of the CDF for the destination simulator.

Updating the Component CDF

To update the component CDF of the symbol cellview,

1. Choose *Tools – CDF – Edit* from the CIW.
2. Choose *Add*.
3. Set the parameter name and attributes as follows:

Field	**Value**
name	macro
type	string
prompt	MacroName
parseAsNumber	no
defValue	*filename_without_ext*

Do not use the `.s` filename extension in the attribute value. For example, if the filename is `opamp.s`, type `opamp` as the *defValue*.

Note: Do not use the Units attribute.

Creating a Stopping Cellview

To create a stopping cellview for your simulator,

1. Edit the symbol cellview.
2. Choose *Design – Save As*.
3. Keep the same cell name, but choose a new view name to match your simulator.

   By convention, the view name is the simulator name, such as cdsSpice.

Updating the Stopping Cellview CDF

To update the stopping cellview CDF to pass parameters into the subcircuit and set the order of the input terminals,

➤ Choose *Tools – CDF – Edit* in the CIW and modify the *simInfo* section of the component CDF as follows:

Field	Value
netlistProcedure	`ansSpiceSubcktCall`
macroArguments	The names of any CDF parameters on the component that you need to pass into the subcircuit. The parameters are mapped to the `&2, &3,` … arguments in the subcircuit file based on the order of the macroArguments.
otherParameters	`macro`
namePrefix	`X`
termOrder	The names of the symbol's terminals, in the order you want them netlisted. The order must match the node order on the `.SUBCKT` line.
componentName	subcircuit

Including the Subcircuit File in the Netlist

There are two ways to include your subcircuit file in the netlist. The syntax of the file depends on which method you choose.

- You can use the Cadence SPICE socket model file inclusion mechanism, in which case your subcircuit file must follow Cadence SPICE rules for specifying the subcircuit name and parameters. In all other respects, the subcircuit file follows the simulation engine's native syntax. You can place the subcircuit file in any of the directories in the model path.
- You can use include files, in which case your subcircuit file must strictly follow the simulation engine's native syntax.

As a Model File

Subcircuit files included using the model path must follow certain Cadence SPICE syntax rules in specifying the subcircuit name and any CDF parameters being passed into the subcircuit. In every other respect, the subcircuit file must follow the simulation engine's native syntax. Follow these syntax rules while creating your subcircuit file:

- Use `&1` as the name of the subcircuit on the `.SUBCKT` line. For example:

  ```
  .SUBCKT &1 34 19 56
  ```

  The analog circuit design environment passes the subcircuit name into the file to replace `&1`.
- To pass CDF parameters into the subcircuit, use the variables `&2`, `&3`, … *`&n`*. The order of these variables must match the CDF parameters you define in the *macroArgument* field of the *simInfo* section of the CDF.
- The last line of the file must be

  ```
  .ENDS &1
  ```

The filename must end in `.s` if the file begins with a lowercase letter, or in `.S` if it begins with an uppercase letter. Names must be no longer than eight (8) characters. For example, if your component is named `opamp`, name the file

```
opamp.s
```

Nested Include Files

Subcircuits that you include using include files must follow the destination simulator syntax strictly.

Use the following parameter format in the HSPICE netlist:

```
X1 in out1 inva wp=7u wn=5u
X2 out1 out2 inva wp=14u wn=10u m=2
```

The subcircuit file `inva_macro` is

```
.SUBCKT inva a y
MP7 y a 2 2 pmos w=wp l=2u
MN8 y a 0 0 nmos w=wn l=2u
.ENDS inva
```

You then add `inva_macro` using *Setup – Env – Include* and set the *Include/Stimulus Syntax* field to *hspiceS*.

HSPICE CDF Examples

This table shows example CDF values for a width parameter named `wn` being passed in to an HSPICE subcircuit.

CDF Parameter	Macro	macroArgumentStyle	Parameter wn
ParamType	string	cyclic	string
ParseAsNumber	no	--	yes
Units	do not use	do not use	lengthMetric
StoreDefault	no	no	no
name	macro	macroArgumentStyle	wn
Prompt	subcircuit name	Macro Arg Style	nmos width
defValue	inva	hspiceS	leave blank
choices	--	default hspiceS	--

This table shows example CDF parameters for the HSPICE multiplier parameter on a subcircuit.

CDF Parameter	Macro	macroArgumentStyle	Parameter m
ParamType	string	cyclic	string
ParseAsNumber	no	--	yes
Units	do not use	do not use	do not use
StoreDefault	no	no	no
name	macro	macroArgumentStyle	m
Prompt	subcircuit name	Macro Arg Style	multiplier

CDF Parameter	Macro	macroArgumentStyle	Parameter m
defValue	inva	hspiceS	--
choices	--	default hspiceS	--

Scope of Parameters

You can use design variables and CDF parameters to set component values.

Design variable values are always global to the design. The scope of CDF parameter values, however, depends on which Analog Expression Language (AEL) functions you use to refer to the parameter.

Note: Do not use callbacks on parameters whose values are expressions, particularly expressions that use pPar. The expressions might not be evaluated correctly and the system does not detect the errors. Use callbacks only to establish dependencies between parameters whose values are numeric constants.

Inheriting from the Same Instance: iPar()

When a parameter value must depend on the value of another parameter on the current instance, use the iPar function.

```
iPar( "CDF_parameter_name" )
```

The value of this expression is the value of this parameter on the current instance, or its value on the cell’s effective CDF.

For example, suppose the parameter AD of a MOS transistor should be a function of its channel width. You can define AD in the Schematic window using the *Edit – Properties* command as

```
iPar("w")*5u
```

The resulting value is the value of `w` on the instance times 5u.

The `iPar` expression is substituted with the value of the parameter, enclosed by parentheses during netlisting. If no value is found, the system reports an error.

Passed Parameter Value from the Parent Instance: pPar()

When a parameter must depend on the value of a passed parameter, use the pPar function.

`pPar( "`*`CDF_parameter_name`*`" )`

The value of this expression is the value of the passed parameter.

For example:

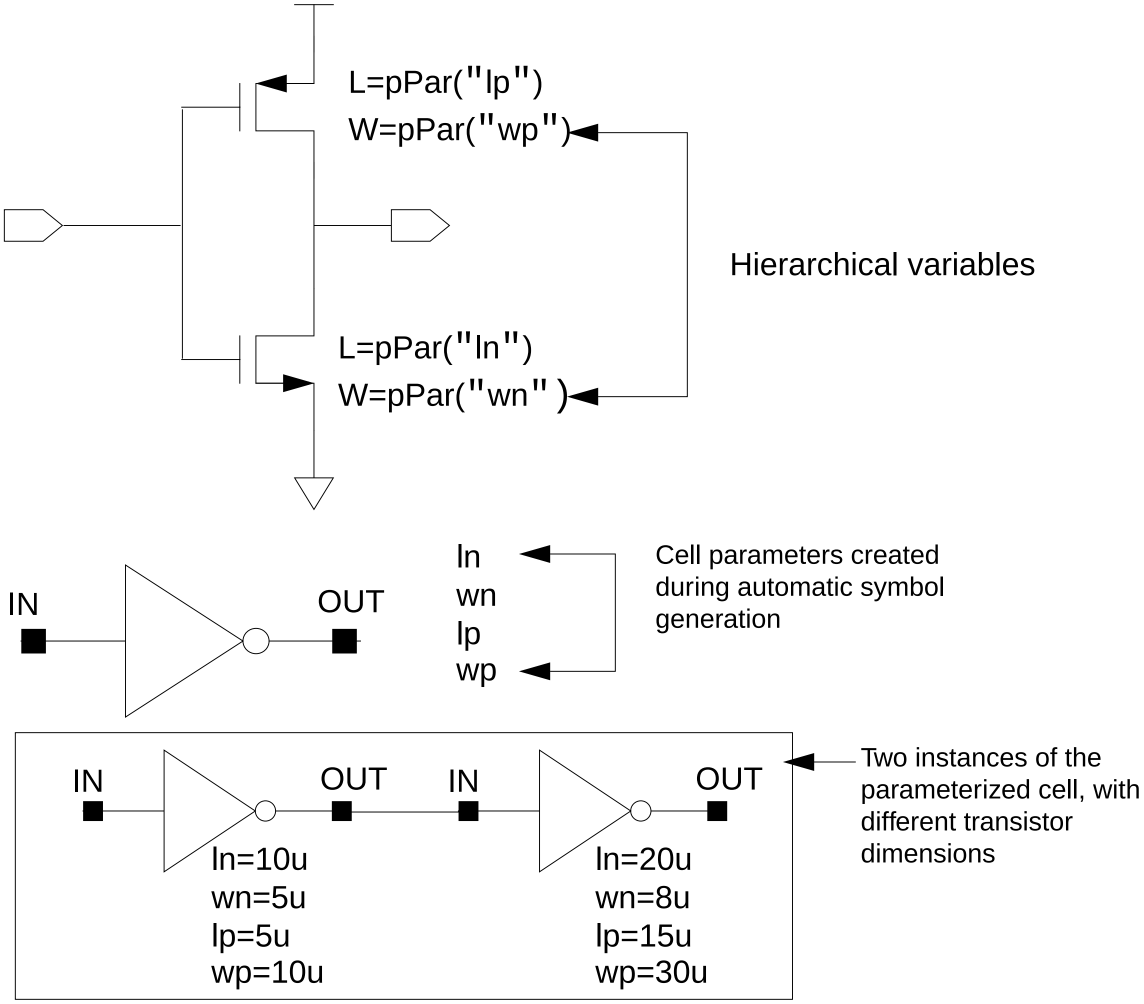
```
pPar("vss")
```

The value of the “DC Voltage” parameter on the `v27` instance in the `aExamples` `opamp` schematic is specified for the `aExamples lowpass` schematic as `15`.

When you create new symbols using automatic symbol generation (the *Create Cellview – From Cellview* command in the Schematic window), the system creates component parameters for the parameters you defined with pPar. The following illustration gives an example of the automatic symbol generation process.

During netlisting, the pPar expression is substituted by the name of the parameter.

Using Ppar



Inheriting from the Current or Any Higher Level: atPar()

When a parameter must depend on a parameter at the current level of the hierarchy or anywhere above the current level (and not just the parent instance), use the atPar function.

```
atPar( "CDF_parameter_name" )
```

First, the flat netlister searches on the instance currently being sent to the netlist. Then it searches on the instance of the cell that contains this instance, and so on up the instance hierarchy. Next, it searches for the property on the current instance master. The master is usually the simulation primitive for this device in the library. Finally, it searches the global cellview for your simulator (that is, the `nlpglobals` cellview). The property search is halted, and the value sent to the netlist when the specified property is found.

Note: Avoid using atPar. Use pPar instead.

Inheriting from the Instance being Netlisted: dotPar()

When a parameter must depend on a parameter of the instance being netlisted, use the dotPar function.

`dotPar( "`*`CDF_parameter_name`*`" )`

Click here to see an example that uses the dotPar function.

Table of Functions

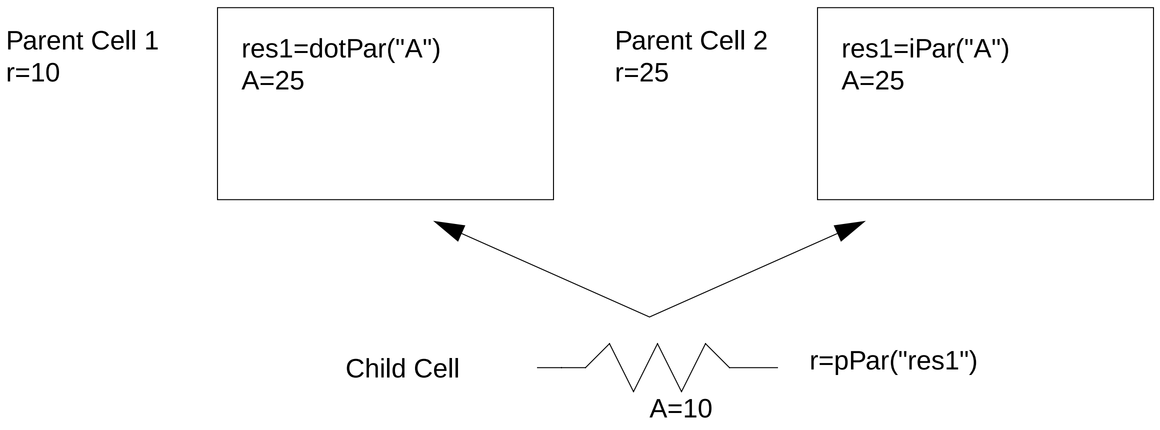
Parameters can be inherited by algebraic expressions that are used as component values. The Analog Expression Language (AEL) provides functions to control how parameters are inherited. (The AEL inheritance functions are compatible with the corresponding NLP functions.)

Functions		**Meaning**	**Scope rules**
AEL	**NLP (OSS)**		
`iPar`	[~	Instance parameter	Search the instance carrying `iPar`, then the effective cell CDF
`pPar`	[+	Parent parameter	Search the parent instance, then the effective cell CDF of the parent instance
`atPar`	[@	At or above the instance	Search the instance currently being written to the netlist, then its effective cell CDF, then continue to the top of the instance hierarchy **Note:** atPar does not have a default value.
`dotPar`	[.	At the instance being netlisted	Search instance currently being written to the netlist, then its effective cell CDF.

Determining the Current Instance in Expression Evaluation

The atPar and dotPar functions identify the current instance differently than do pPar and iPar. For atPar and dotPar, the current instance is always the instance being netlisted. For pPar and iPar, the current instance is the one carrying the pPar or iPar expression. When component values depend on parameter inheritance expressions in another cellview, different functions choose a different current instance.

In the example below, the value of *r* for two instances of a resistor is different. The dotPar function looks for the value of *A* on the resistor instance being netlisted, whereas iPar looks in the cell containing the iPar function.



Nesting Functions

Arguments to inheritance functions can have values determined by other inheritance functions. The identity of the current instance and the parent instance are determined relative to the instance on which the current expression is stored.

Suppose you have a hierarchical instance with a CDF parameter of `resVal = 1K`, and the corresponding schematic has some resistor instances on it. With the *Edit – Properties* command, you can set the value of `r` on a resistor instance to be

```
pbase(pPar("resVal"))
```

When you create a raw netlist using hierarchical netlisting, the resistor line in the raw netlist `.s` file shows a value similar to

```
pbase((&2))
```

The value of `N` in `&N` depends on how many CDF parameters were being inherited into that subcircuit. If you create a final netlist, you see a value of 1 K (in scientific notation) on the resistor.

If you now set `TEMPDC` to 125 and rerun final netlisting, the resistor has a value of 1.2 K because it was scaled by the piecewise-linear function `TABLE`.

If you use flat netlisting, the raw netlist `.c` file shows the resistor value to be `pbase((1K))` because the netlister can fully evaluate the pPar function.

Note: You cannot use iPar or pPar in the initialization file because the simulator cannot evaluate these AEL expressions.

How the Netlister Expands Hierarchy

While netlisting a hierarchical design, the analog circuit design environment expands every cell (instance) into lower level cells until it reaches a cell designated as a primitive. The primitive is then added to the netlist. This process is called design hierarchy expansion or view selection.

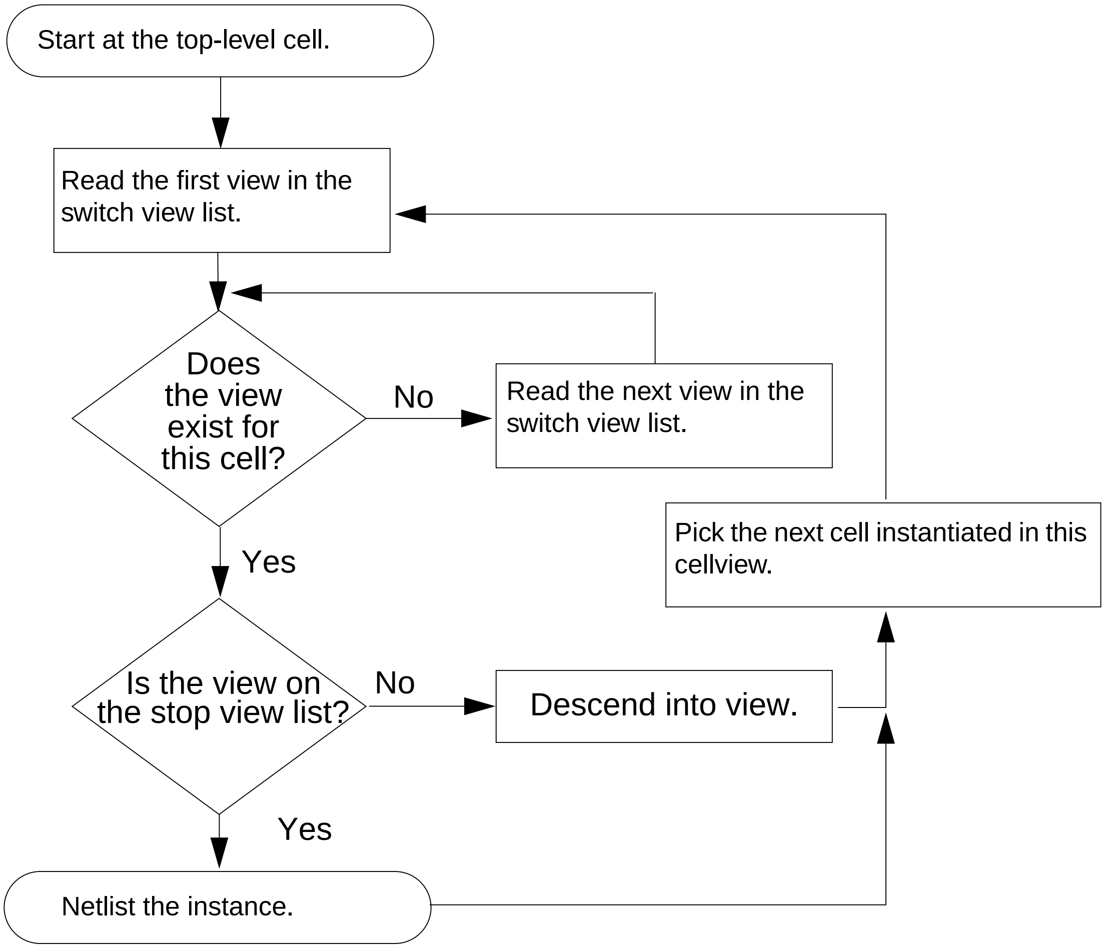
At each level in the hierarchy, there can be several views of each cell. You use a view list to specify which view the analog circuit design environment selects and descends into. View lists can be global to the entire design or specific to an instance as specified by its property values.

For analog simulation, you specify the global or default view list in the *Switch View List* field of the Environment Options form. If an instance does not have any of the views listed in the view list, the netlister reports an error.

The netlister identifies primitives with a stop list. When the netlister reaches a view that is listed in both the view list and stop list, the instance is netlisted and no further expansion occurs below this level. The global stop list is set internally by the analog circuit design environment.

Note: Parasitic simulation uses different processes for creating view lists and stop lists.

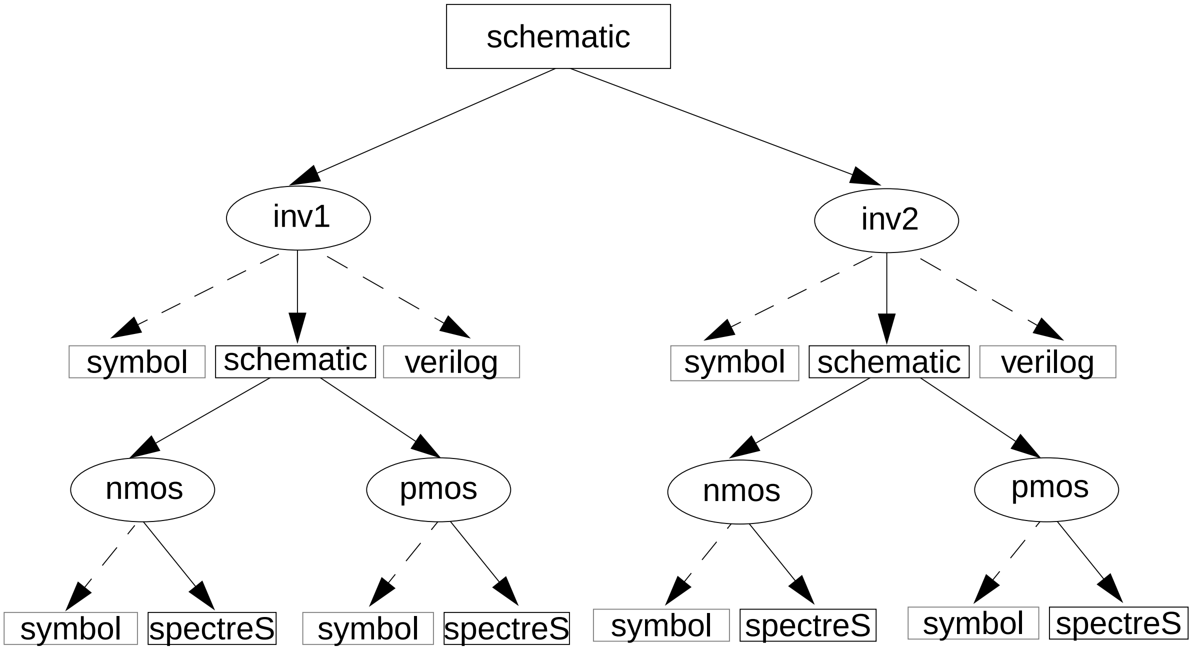
The flowchart shows how a netlister like OSS expands a design.


Start at the top-level cell.
Read the first view in the switch view list.
Does the view exist for this cell?
No
Read the next view in the switch view list.
Yes
Pick the next cell instantiated in this cellview.
Is the view on the stop view list?
No
Descend into view.
Yes
Netlist the instance.

Netlisting Sample for SpectreS

The following figure illustrates how hierarchy expansion is performed on a simple design. The solid lines show the view selection and design expansion based on the *Switch View List* and *Stop View List* that are provided.

Switch View List:	spectreS schematic cmos.sch verilog
Stop View List:	spectreS verilog

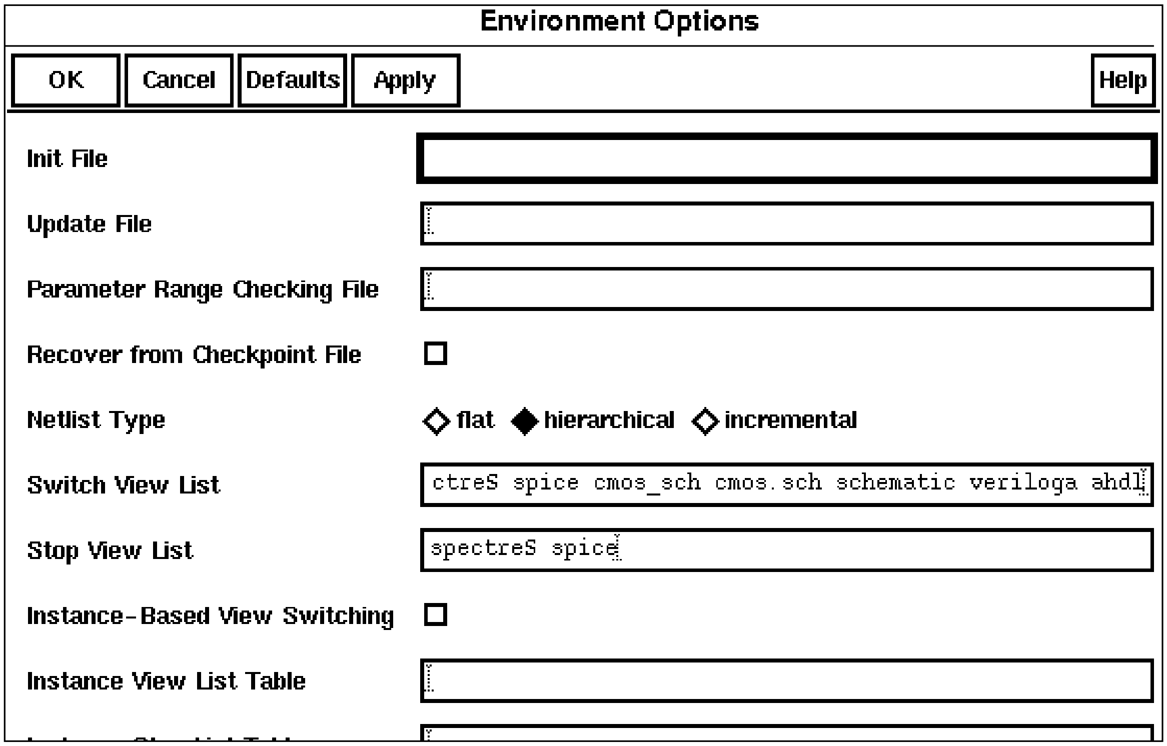


Modifying View Lists and Stop View Lists

To modify a *Switch View List* or stop view list,

1. Choose *Setup – Environment* from the Simulation window.

The Environment Options form appears.



Init File

Update File

Parameter Range Checking File

Recover from Checkpoint File □

Netlist Type ◇ flat ◆ hierarchical ◇ incremental

Switch View List `ctreS spice cmos_sch cmos.sch schematic veriloga ahdl`

Stop View List `spectreS spice`

Instance-Based View Switching □

Instance View List Table

For detailed information about the form, see “Environment Options” on page 98.

2. Modify entries to the *Switch View List* field, as needed.

   The switch view list informs the netlister how to descend into different views in the design. Sequence is important in the switch view list. Refer to the flowchart to see how the netlister selects appropriate views.

3. Modify entries to the *Stop View List* field, as needed.

   In most cases, you can control netlisting adequately using the switch view list. If necessary, you can add entries to the end of the stop view list. Sequence is not important in the stop view list.

4. Click *OK* or *Apply* to add your changes.

Using Instance-Based View Switching

It might be necessary to change the view list or stop list for specific instances in your design. This is called instance-based view switching. Instance-based view switching is supported only for purely analog designs, not for mixed-signal designs. You can also use config views and the Hierarchy Editor to switch instance views.

For analog designs, you enable instance-based view switching through a toggle switch on the Environment Options form. When *Instance-Based View Switching* is enabled, the netlister checks each instance for either an *instViewList* property or an *instStopList* property. If one of these properties is found, its value is read to identify the appropriate list to be used as the view list or stop list during netlisting.

To allow creation of alternate view lists and stop lists, the Environment Options form contains two fields: the *Instance View List Table* and the *Instance Stop List Table*. You create these tables by entering names and views for view lists and stop lists that you will use for your design. A specific syntax is needed to create these tables. Refer to the procedures in the following sections for details on creating a view list table and a stop list table.

To use instance-based view switching for analog designs you need to

- Turn on the toggle switch for instance-based view switching in the Environment Options form
- Add list names and entries in the proper syntax in the Environment Options form to create the *View List Table* or the *Stop List Table*
- Set a property on each instance that provides the name of the list to use for the instance

In the following sample of an *Instance View List Table*

```
(("analog1" "spectreS" "cdsSpice" "behavioral" "schematic")
("analog2" "spectreS" "coms.sch" "schematic"))
```

the list named `analog1` is made up of the views `spectreS`, `cdsSpice`, `behavioral`, and `schematic`. The list named `analog2` is made up of the views `spectreS`, `coms.sch`, and `schematic`.

Setting Up View List and Stop List Tables

To set up list tables for instance-based view switching,

1. Choose *Setup – Environment* in the Simulation window.

   The Environment Options form appears.

2. Click *Instance-Based View Switching*.

   This tells the netlister that you will be using a customized view list or stop list for some of the instances in your design.

3. Type an entry for *Instance View List Table* or for *Instance Stop List Table*.

   Each table is a list of sublists. Each sublist (distinguished by the parentheses) is made up of character strings: the first string contains the list name and the rest contain the views associated with the name.

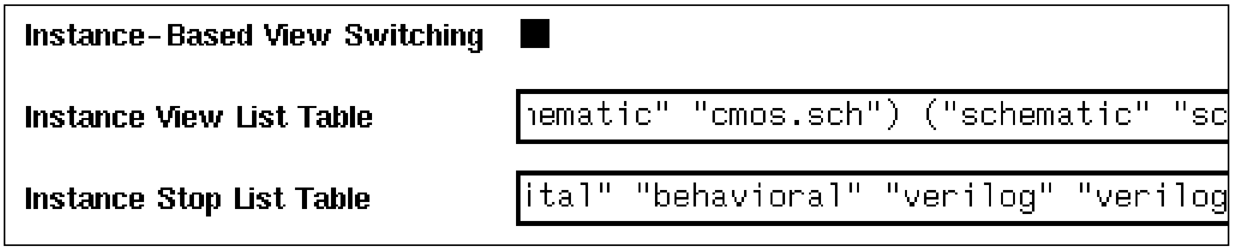
   The format for the table is

   ```
   (("name1" "spectreS" "schematic")("name2" "cdsSpice" "spectreS"
   "schematic"))
   ```

   Note: On the Environment Options form, you type in one continuous line for the entries of the table. Enclose each list in parentheses and separate the lists with a single space.

   On the Environment Options form, you can drag the cursor over the left or right edge of the table data field to scroll to the undisplayed portion of the list.

   The sequence of views is significant for view lists, but it is not significant for stop lists. The following figure shows filled-in entries for instance-based view-switching fields on the Environment Options form.



Instance-Based View Switching ■

Instance View List Table `hematic" "cmos.sch") ("schematic" "sc`

Instance Stop List Table `ital" "behavioral" "verilog" "verilog`

Assigning Properties to Instances in the Schematic

When instance-based view switching is active, the netlister checks each instance for an instViewList or instStopList property. If the property exists, the netlister interprets the property value as the name of the list in the *Instance View List Table* or the *Instance Stop List Table* field.

To create a view list or stop list property for an instance,

1. Choose the instance in the schematic.
2. In the Virtuoso Schematic window, choose *Edit – Properties – Objects*.

   The Edit Properties form appears.
3. In the user properties section of the form, choose *Add*.

   The Add Property form appears.

4. Set *Name* to *instViewList* or *instStopList*.
5. Set *Property Type* to *string*.
6. Set *Property Value* to the name of the appropriate list.

   For example, if you use the previous sample for your table entries, you can specify either *name1* or *name2* as the value of this property.

About Netlists

The analog circuit design environment creates or updates the netlist automatically when you give the command to run a simulation.

You do not need to use the *Netlist – Create* command unless

- You want to use the analog circuit design environment to create a netlist, but run the simulator in standalone mode
- You want to modify the netlist, perhaps to take advantage of features that the analog circuit design environment interface to your simulator does not support
- You want to read the netlist before starting the simulation

There are two kinds of netlists:

- Raw netlists, which contain component information but no simulation control data (the simulator performs postprocessing on the raw netlist file before generating the final netlist)
- Final netlists, which are complete and ready to be input to your simulator

Netlists can be flat or hierarchical, and you can tell the system to incrementally renetlist only the changed schematics in a design.

Incremental Netlisting

You specify whether to generate flat or incremental hierarchical netlists in the Environment Options form with the *Netlist Type* field.

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.

Incremental netlisting works with Cadence SPICE and Spectre simulators and simulators in the Cadence SPICE socket.

Whenever incremental netlisting is impractical, the system automatically re-netlists every schematic. This happens (for the next netlist compilation only) when

- You change the CDF of an instance using the CDF Editor
- You add a new CDF parameter to a parameterized subcircuit schematic with the pPar function. Typically, this means you have added pPar to the expression for a component value, and the argument toPar is the new parameter.

Creating and Displaying a Raw Netlist for Socket Simulations

To create and display a new raw netlist,

➤ Choose *Simulation – Netlist – Create Raw*.

The raw netlist is created in a file with the `.c` extension and resides in the following directory:

`projectDirectory/topCellName/simulatorName/schematic/``netlist/`
`topCellName.c`

`projectDirectory`

The directory you specified in the Choosing Simulator/Directory/Host form.

`topCellName` The root level cell name of the design.

`simulatorName` The name of the simulator.

`schematic` The view name being netlisted.

If you netlist a hierarchical design using Cadence SPICE, this command creates a `topCellName.c` file containing information about the netlist file and a `topCellName.s` file containing the component descriptions of the elements in the schematic.

To display an existing raw netlist,

➤ Choose *Simulation – Netlist – Display Raw*.

Creating and Displaying a Final Netlist for Socket Simulations

To create and display a new final netlist,

➤ Choose *Simulation – Netlist – Create Final*.

This file is always named `m6File` and resides in the following directory:

`projectDirectory/topCellName/simulatorName/schematic/``netlist/m6File`

`projectDirectory`

The directory you specified in the Choosing Simulator/Directory/Host form

`topCellName` The root level cell name of the design.

`simulatorName` The name of the simulator.

`schematic` The view name being netlisted.

To display an existing netlist,

➤ Choose *Simulation – Netlist – Display Final*.

Form Field Descriptions

Setup Analog Stimuli

Stimulus Type

Inputs sets the stimulus for the signals with input pins in the schematic.

Global Sources lets you assign DC voltages to global signals that represent power supplies in the design.

Language Syntax specifies the simulator being used and lists the signals with input pins in the design. The entries in the list box are displayed in the syntax appropriate to the specified simulator.

Name identifies the signal name that is currently selected.

Library identifies the library where the selected signal or global source model was found.

Change recalculates the input voltage or current for the selected signal based on the function, type, and property values specified in the lower portions of the form. The calculated value is specified in the list box in the appropriate syntax.

Enabled lets you specify whether each signal is on or off.

Function lets you choose the function for the selected signal.

dc displays the direct current stimulus option properties and values.

pulse displays the pulse stimulus option properties and values.

sin displays the sinusoidal stimulus option properties and values.

exp displays the exponential stimulus option properties and values.

pwl displays the piecewise linear stimulus option properties and values.

pwlf displays the name of the file containing piecewise linear stimulus option properties and values.

sffm displays the single frequency FM stimulus option properties and values.

Type lets you select the **voltage** or **current** for the signal highlighted in the list box.

Parameters and their values identify the simulator-specific parameters required by your simulator. The parameters list here varies depending on the simulator you are using. Refer to your simulator documentation for information on setting or changing these parameter values.

The form lets you set inputs and global sources for your design.

The text above the list box specifies the language syntax used for the input statements, the name of the highlighted input pin preceded by `v` (for voltage) or `i` (for current) depending on its stimulus type, and the library from which the source came. If you are using the *refLibs* property to specify a library search sequence, check this field to verify that the appropriate library is being used.

The list box contains the current netlist values of the input or bidirectional pins. Each line contains the proper syntax for your simulator.

The fields displayed below the *Function* and *Type* options (*AC magnitude*, *AC phase*, *DC voltage*, and so forth in this example) provide parameter input specific to your simulator.

When the form is first displayed, fields in this section can be blank, can contain default values, or can contain initial values that you specified at another time.

The form changes dynamically when you select a different input pin, function, or type.

Editing Design Variables

Name is an optional name for the variable, which appears in the *Table of Design Variables* list box.

Value (Expr) is the variable value, either a number or an expression.

Add creates the variable you specified in the *Selected Variable* area.

Delete removes a highlighted variable. Click in the list box to highlight a variable.

Change updates the highlighted variable with the new information from the *Selected Variable* area.

Next highlights the following signal or expression in the *Table of Design Variables* list box.

Clear empties the *Selected Variable* area so you can type in a new variable.

Find locates the highlighted variable in your design.

Cellview Variables lets you keep variables consistent in the simulation environment and the cellview design database by copying them back and forth.

> **Copy From** copies the variable values in the schematic cellview into the simulation environment.
>
> **Copy To** copies the variable values in the simulation environment to the schematic cellview.

Table of Design Variables identifies the name and value of each design variable in the design. Each entry is numbered for easy reference.

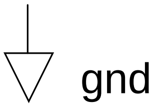
5

Setting Up for an Analysis

This chapter shows you how to set up to run an analysis.

Required Symbol

You must include an instance of the cell `gnd` from the `analogLib` library in the schematic. Analog simulators need this cell to recognize the DC path to ground.



Setting Up with Different Simulators

To set up analyses,

1. From the Simulation window, choose *Analyses – Choose*, or from the Schematic window, choose *Setup – Analyses*.

   The Choosing Analyses form for your simulator appears.

   For help setting up a particular analysis, see “Setting Up a Spectre Analysis” on page 192, or refer to your simulator manual.

2. Select an analysis.

   The Choosing Analyses form redraws to show the parameters for the new analysis.

3. Set the analysis options.

4. Click *Apply*.

   The analysis you selected displays in the *Analyses* section of the Simulation window.

The next step is usually selecting the outputs you want to save.

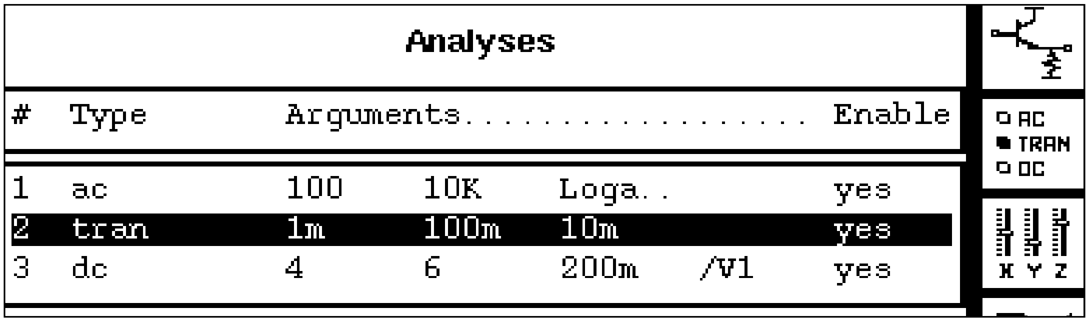
Tip

If you do any select operation from the schematic after you have already invoked the *Choosing Analysis* form, the control does not return back to the analysis form. This can be inconvenient if several windows are open. The control can be made back to the analysis form, by clicking on *Analyses – Choose* once more, in the *Virtuoso Analog Design Environment* window.

Deleting an Analysis

To delete an analysis,

1. In the Simulation window, click the analysis to highlight it.



Analyses

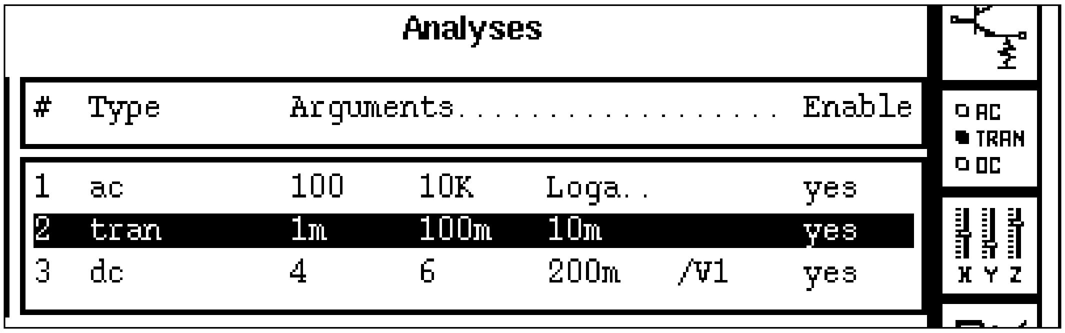
#	Type	Arguments..................				Enable
1	ac	100	10K	Loga..		yes
2	tran	1m	100m	10m		yes
3	dc	4	6	200m	/V1	yes

2. Choose *Analyses – Delete* or click the delete icon.

Enabling or Disabling an Analysis

To temporarily disable an analysis without deleting it from the environment,

1. In the Simulation window, click the analysis to highlight it.



Analyses

#	Type	Arguments..................				Enable
1	ac	100	10K	Loga..		yes
2	tran	1m	100m	10m		yes
3	dc	4	6	200m	/V1	yes

2. Choose *Analyses – Disable*.

To enable a disabled analysis,

1. In the Simulation window, click the analysis to highlight it.
2. Choose *Analyses – Enable*.

Note: You can also enable and disable analyses with the *Enabled* option in each Choosing Analyses form. Click to change the option and then click *Apply*.

Saving the Analysis Setup

You can save the current settings in the Choosing Analyses forms and later restore these analyses.

To save the analysis setup,

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Environment – Save State*.

   The Saving State form appears.

2. Enter a name for the saved simulation state.
3. Check that the *Analyses* box is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring a Saved Analysis Setup

To restore a saved analysis setup,

1. In the Simulation window, choose *Session – Load State*, or from the Schematic window, choose *Analog Environment – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Choose a run directory with the *Cell* and *Simulator* fields.

   The list box shows the saved states for the cell and simulator combination.

3. Click a state name.
4. Check that *Analyses* is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Packing a Design in AMS

AMS offers a design packing tool, Pack-Files, to give you a standard way of packaging design runs so that they can be sent to others, who can then see it exactly the way you see it. This capability is useful, for example, when you encounter a bug, program crash, or slow performance and need to have Cadence support personnel reproduce the problem.

Note: The design packing tool is not supported for the AIX operating system.

To pack your design,

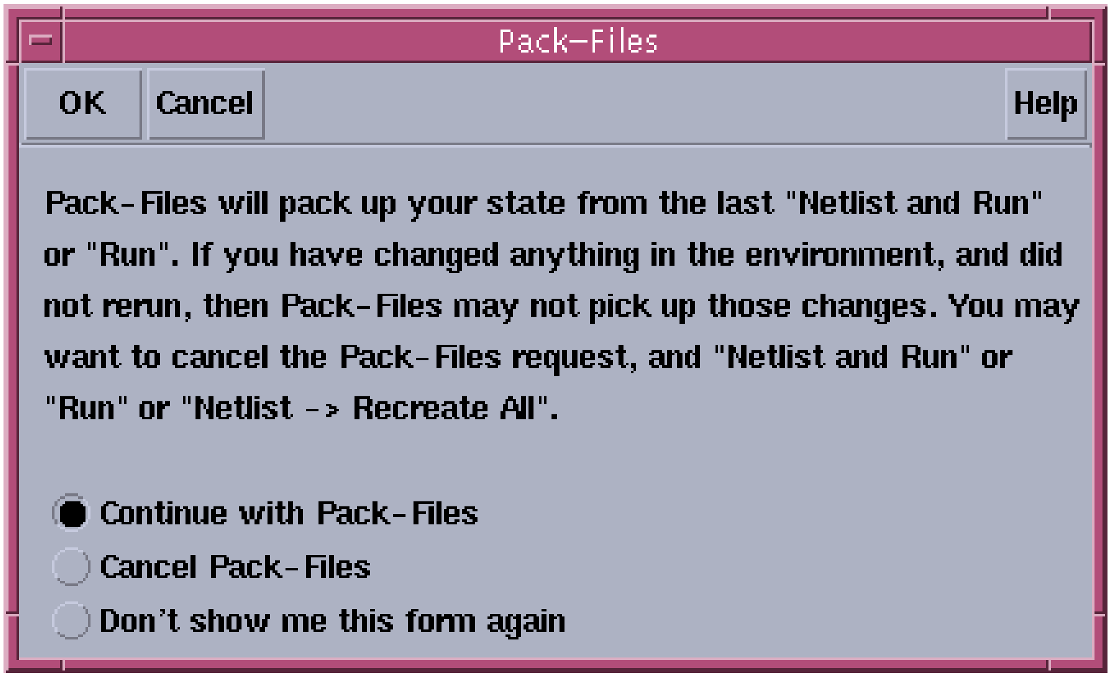
1. Choose *Tools – Pack-Files*.

   The Pack-N-Go form appears notifying you that the state after the last *Netlist and Run* or *Run* commands will be used to create the Pack-Files data. You have the following choices:

   ❑ *Continue with Pack-N-Go* – if you do not want to regenerate the netlist.

   ❑ *Cancel Pack-Files* – if you want to regenerate the netlist.

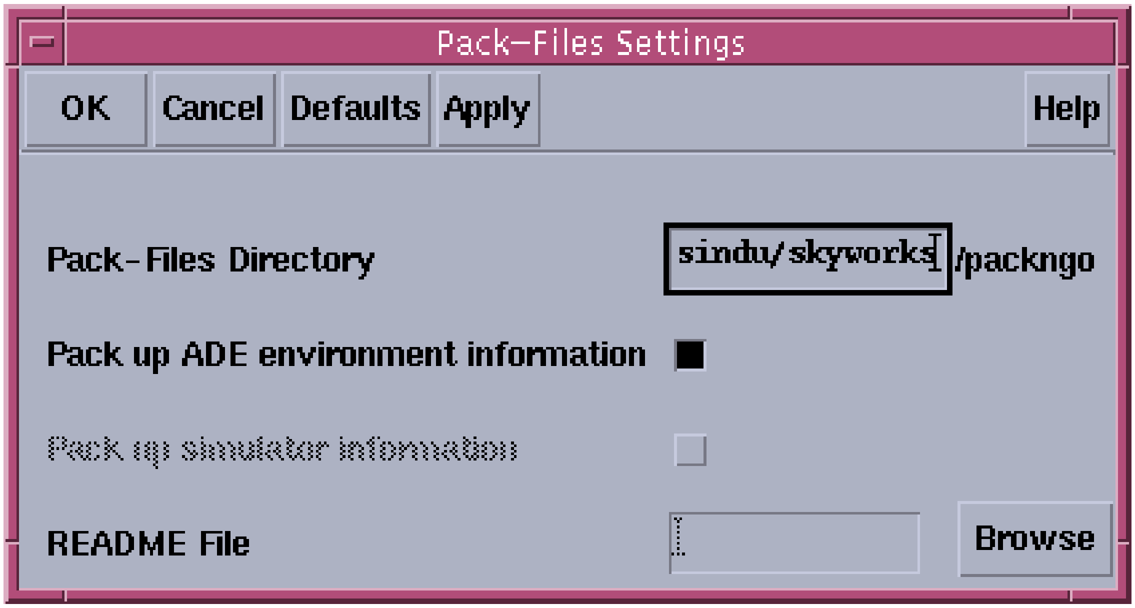
   ❑ *Don't show me this form again* – if you would prefer to work with Pack-Files without seeing this pop-up again.



Pack-Files will pack up your state from the last "Netlist and Run" or "Run". If you have changed anything in the environment, and did not rerun, then Pack-Files may not pick up those changes. You may want to cancel the Pack-Files request, and "Netlist and Run" or "Run" or "Netlist -> Recreate All".

● Continue with Pack-Files
○ Cancel Pack-Files
○ Don't show me this form again

If you select the first option button and continue the operation, the Pack-Files Settings form appears.



Pack-Files Directory sindu/skyworks /packngo

Pack up ADE environment information ■

Pack up simulator information □

README File

2. Your current working directory appears by default in the *Pack-Files Directory* field. You can specify another directory if you want. A subdirectory called `packngo` is created in the directory specified.

3. The *Pack up ADE environment information* option button is selected by default. If you do not want to pack ADE information, deselect it.

4. The *Pack up simulator information* option button is selected by default. If you do not want to pack simulator information, you may deselect it.

   Note: On the IBM platform, this option is not supported. For other platforms, ensure that you select either this option or the *Pack up ADE environment information* option, or both.

5. Specify a *README File* by typing it in or using the *Browse* button. You can specify information and instructions about the design in a readme file.

6. Click *OK* or *Apply*.

   The Pack-Files tool copies data into the directory. A series of messages appear in the CIW showing what the tool does. A record file, `ade.record`, is created including all information about the design the way it is in the active ADE session. This information includes files such as `CDS.log`, netlist files or files in the `psf` directory, `.cdsenv`, `.cdsinit`, `cds.lib`, and so on. The utility runs and prints the output into the icms log file. It also writes a duplicate output into a specified log file. Finally, a tar file is created, which you may e-mail to your Cadence support person.

Caution

Check that the data you send out does not have any sensitive or confidential information that you do not want going out of your company.

This form has a corresponding script, *packngo*, which is located in the dfII/bin directory. It can be run standalone to pack up environment and/or simulator files. For information about this script, see *Appendix F*, *Using the Design Packing Tool* of the *Virtuoso AMS Simulator User Guide*.

Setting Up a Spectre Analysis

To set up analyses for the Spectre simulator,

1. Choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. Choose an analysis.

   The Choosing Analyses form redisplays to show the parameters for the new analysis.

3. Set the options and click *Apply*.
4. Choose another analysis to set up.

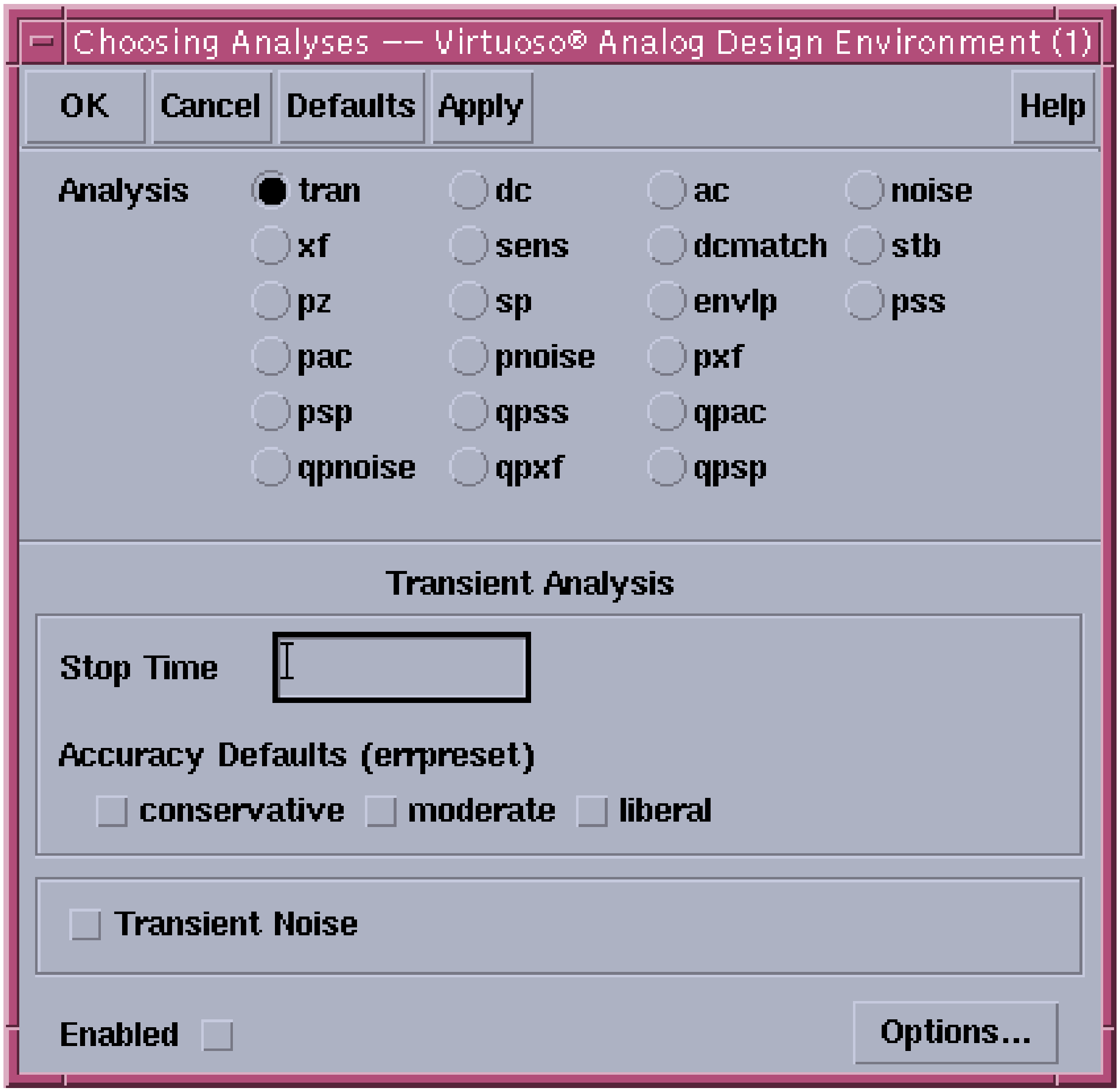
The next step is usually selecting the outputs you want to save.

For help setting up a particular analysis, refer to the *Spectre Circuit Simulator Reference* manual.

Transient Analysis

The transient analysis computes the transient response of a circuit over an interval. The initial condition is taken to be the DC steady-state solution.

To set up a transient analysis,


Choosing Analyses -- Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help
Options...

Analysis ● tran ○ dc ○ ac ○ noise
○ xf ○ sens ○ dcmatch ○ stb
○ pz ○ sp ○ envlp ○ pss
○ pac ○ pnoise ○ pxf
○ psp ○ qpss ○ qpac
○ qpnoise ○ qpxf ○ qpsp

Transient Analysis

Stop Time

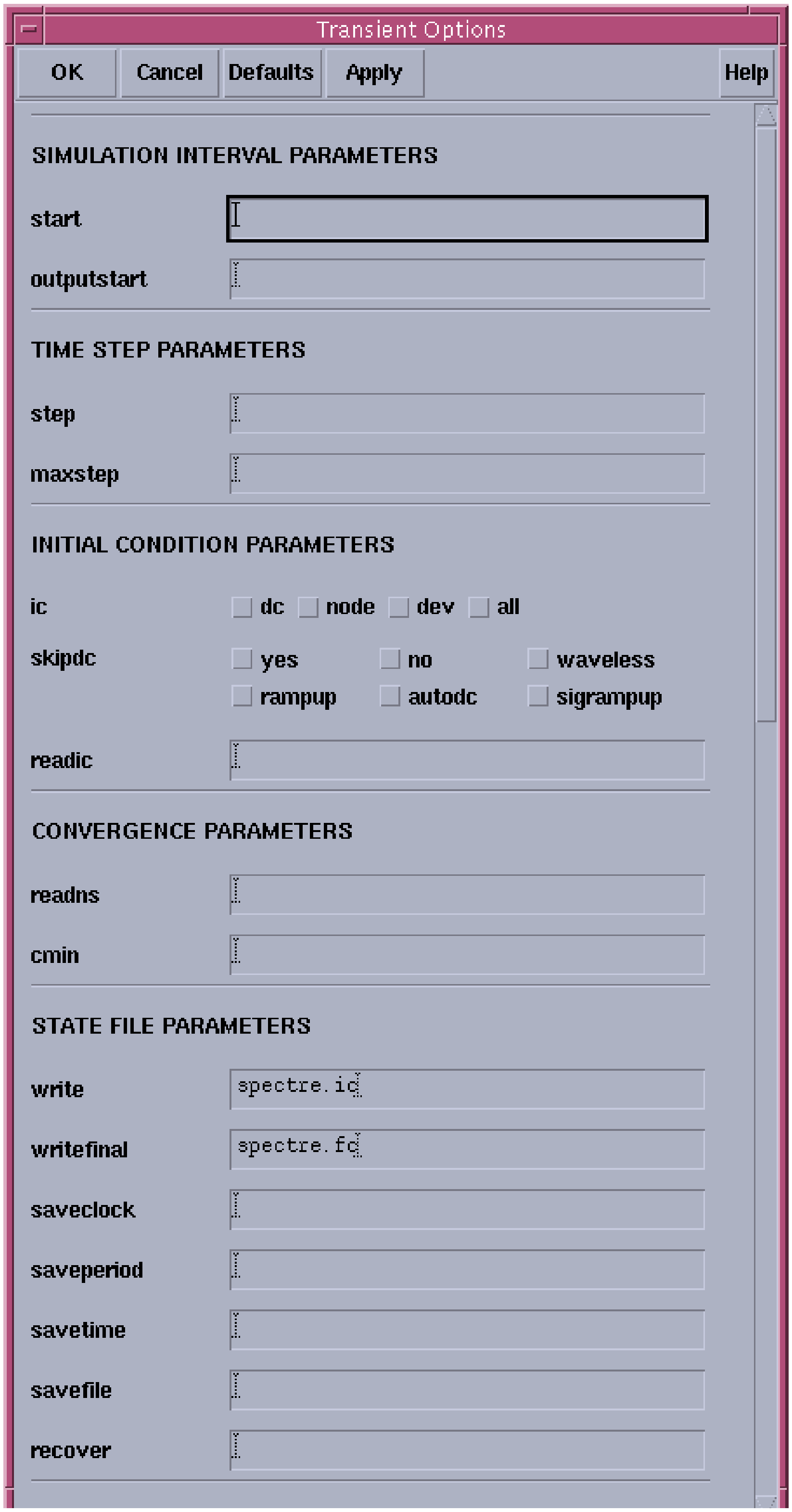
Accuracy Defaults (errpreset)
□ conservative □ moderate □ liberal

□ Transient Noise

Enabled □

1. In the Choosing Analyses form, enter the *Stop Time*.

2. Click *Options* to bring up the Transient Options form.



SIMULATION INTERVAL PARAMETERS

start

outputstart

TIME STEP PARAMETERS

step

maxstep

INITIAL CONDITION PARAMETERS

ic □ dc □ node □ dev □ all

skipdc □ yes □ no □ waveless □ rampup □ autodc □ sigrampup

readic

CONVERGENCE PARAMETERS

readns

cmin

STATE FILE PARAMETERS

write spectre.ic

writefinal spectre.fc

saveclock

saveperiod

savetime

savefile

recover

3. Click *Apply*.

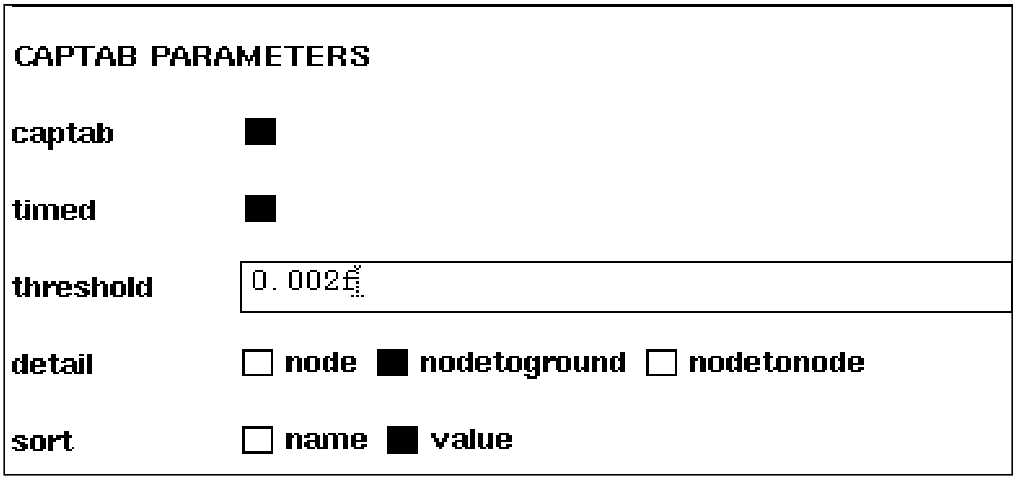
Transient Noise Analysis

Beginning with 5.1.41 USR1, you have the option of obtaining Spectre from the MMSIM release stream. While Spectre will continue to ship with 5.1.41, new features and all but the most critical bug fixes will be provided exclusively with the MMSIM stream. MMSIM6.0 will be released at the same time as the 5.1.41 USR1 update, but must be downloaded and installed separately. To use these releases together, put the `<path_to_mmsim_release>/tools/bin` directory before any dfII paths in your $path.

In this release, the current transient analysis has been extended to support transient noise analysis.

CAPTAB Parameters

You can generate capacitive loading information about a circuit, after a Spectre simulation. For transient analysis, you can specify specific transient timepoints at which to create a capacitance table, using the *infotimes* parameters of Spectre's transient analysis. If you do not specify these parameters, the capacitance table is generated for the final time point. You can specify these parameters using the following components available on the *transient options* window in the section *CAPTAB PARAMETERS.*



CAPTAB PARAMETERS

captab ■

timed ■

threshold 0.002f

detail □ node ■ nodetoground □ nodetonode

sort □ name ■ value

captab indicates if you have specified captab parameters. (Enabled or Disabled).

timed indicates if *infotimes* will be used for the purpose of storing captabs instead of operating points (Enabled or Disabled).

threshold indicates the threshold value in real numbers.Results below this value are omitted from the output. The default value is 0.0.

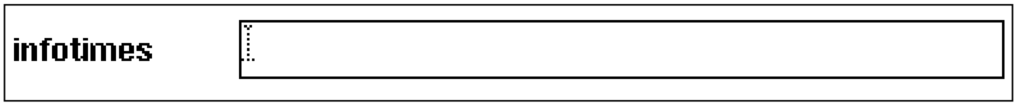
detail can be set to *node*, *nodetoground*, and *nodetonode*. The default option is *node.* To determine the *node to ground* capacitance for *DC* and *Transient* analysis, enable the *nodetoground* button on the appropriate *Options* form. When you run the simulation, the node to ground capacitance for all the nodes is displayed in the log file (`spectre.out`).

sort can be set to *name* and *value*. This can be set in order to sort the entries in the table by their value, or alphabetically by name. The default option is *name*.

For details on Transient Analysis*,* refer to the *Analysis Statements* chapter of the *Spectre Circuit Simulator Reference*.

Infotimes

Operating-point data can be saved by Spectre during a transient analysis. To control the amount of data produced for operating-point parameters, you can use the infotimes option to specify at which time points you would like to save operating point output for all devices.



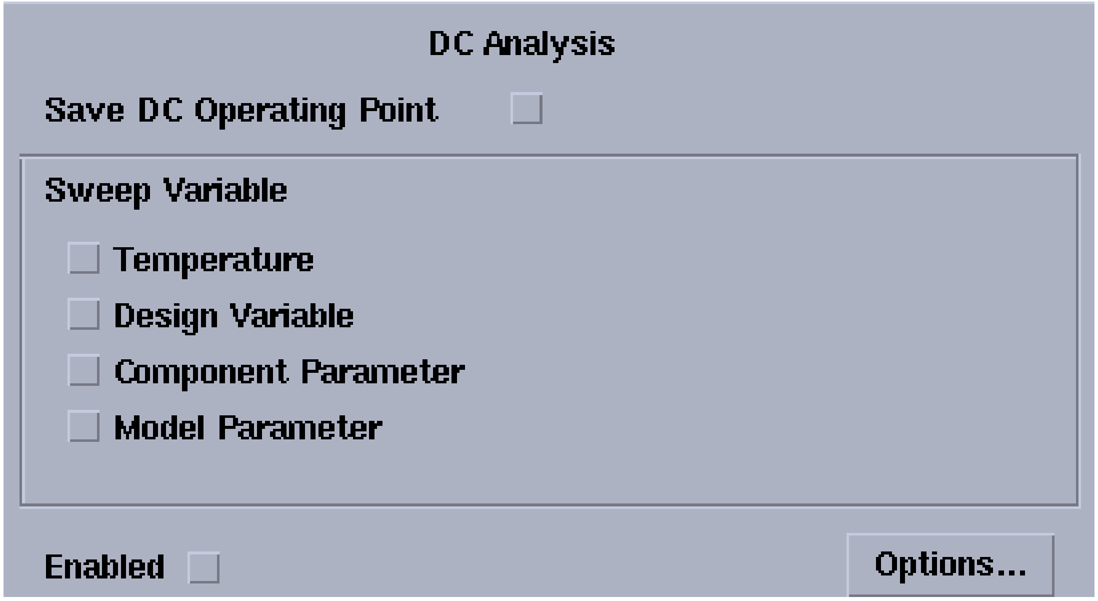
infotimes

infotimes indicates a vector of numbers specifying specific times at which operating-point data is to be collected. 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 in the simulation output window. Once infotime values are specified and simulated successfully, clicking the *Results – Print – Transient Operating Point* menu and then selecting a device on schematic, will print the operating point data for all the timepoints saved. If nothing is specified in this field then the infotimes option is not used.

If the user specifies infotimes and then simulates successfully, clicking the *Results – Annotate – Transient Operating Point* menu will bring up the Annotating Transient Operating Points Results form.

DC Analysis

The DC analysis finds the DC operating point or DC transfer curves of the circuit. To generate transfer curves, specify a parameter and a sweep range. The parameter can be a temperature, a device instance parameter, or a device model parameter.



DC Analysis

Save DC Operating Point □

Sweep Variable

□ Temperature
□ Design Variable
□ Component Parameter
□ Model Parameter

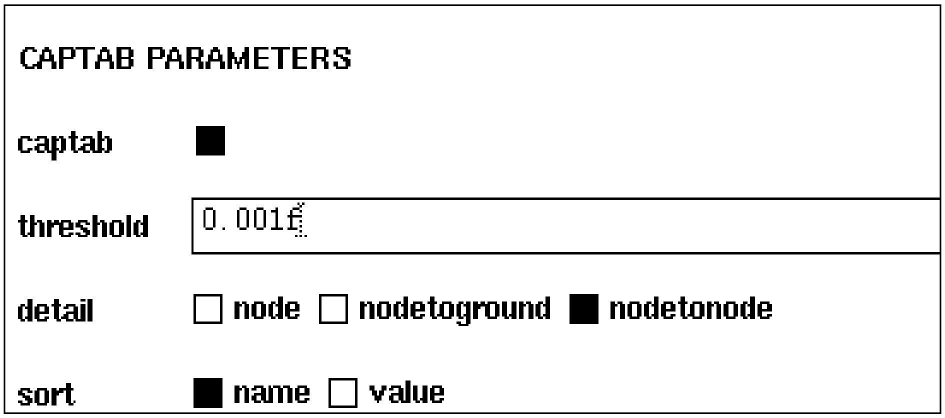
Enabled □

To save the DC operating point,

➤ Click *Save DC Operating Point*, click *Enabled*, and click *Apply*.

CAPTAB Parameters

You can generate capacitive loading information about a circuit after a Spectre simulation. The following additional components are available on the *dc options* window:



CAPTAB PARAMETERS

captab ■

threshold 0.001f

detail □ node □ nodetoground ■ nodetonode

sort ■ name □ value

captab indicates if you have specified captab parameters. (Enabled or Disabled)

threshold indicates the threshold value in real numbers.Results below this value are omitted from the output. The default value is 0.0

detail includes *node*, *nodetoground*, and *nodetonode*. The default option is *node*

sort includes *name* and *value*. This can be set in order to sort the entries in the table by their value, or alphabetically by name. The default option is *name*.

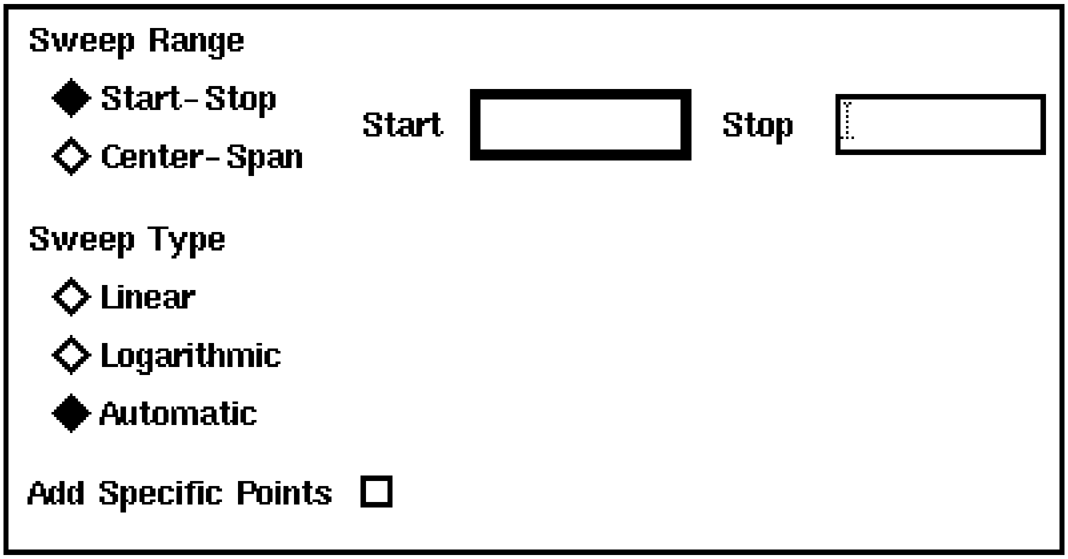
For details on DC Analysis refer to the *Analysis Statements* chapter of the *Spectre Circuit Simulator Reference*.

Sweeping a Variable

To run a DC transfer curve analysis and sweep a variable,

1. Choose a sweep variable.

   The Choosing Analyses form redisplays to show additional fields.



Sweep Range

◆ Start-Stop　　Start □　　Stop □

◇ Center-Span

Sweep Type

◇ Linear

◇ Logarithmic

◆ Automatic

Add Specific Points □

2. Specify the necessary parameters.

   - ❑ If you sweep a design variable, fill out the name of the design variable, or choose from the list box after pressing the select button.
   - ❑ To sweep a component, specify the component name and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.

- ❑ To sweep a model parameter, enter the model and parameter names.

3. Specify the sweep range and type.

The sweep type options are mapped to Spectre statements:

- ❑ Linear + Step Size = step
- ❑ Linear + Number of Steps = lin
- ❑ Logarithmic + Points Per Decade = dec
- ❑ Logarithmic + Number of Steps = log
- ❑ Add Specific Points = values=[…]

4. Click *Options* to set the spectreS options controlling DC simulation.

5. Click *Apply*.

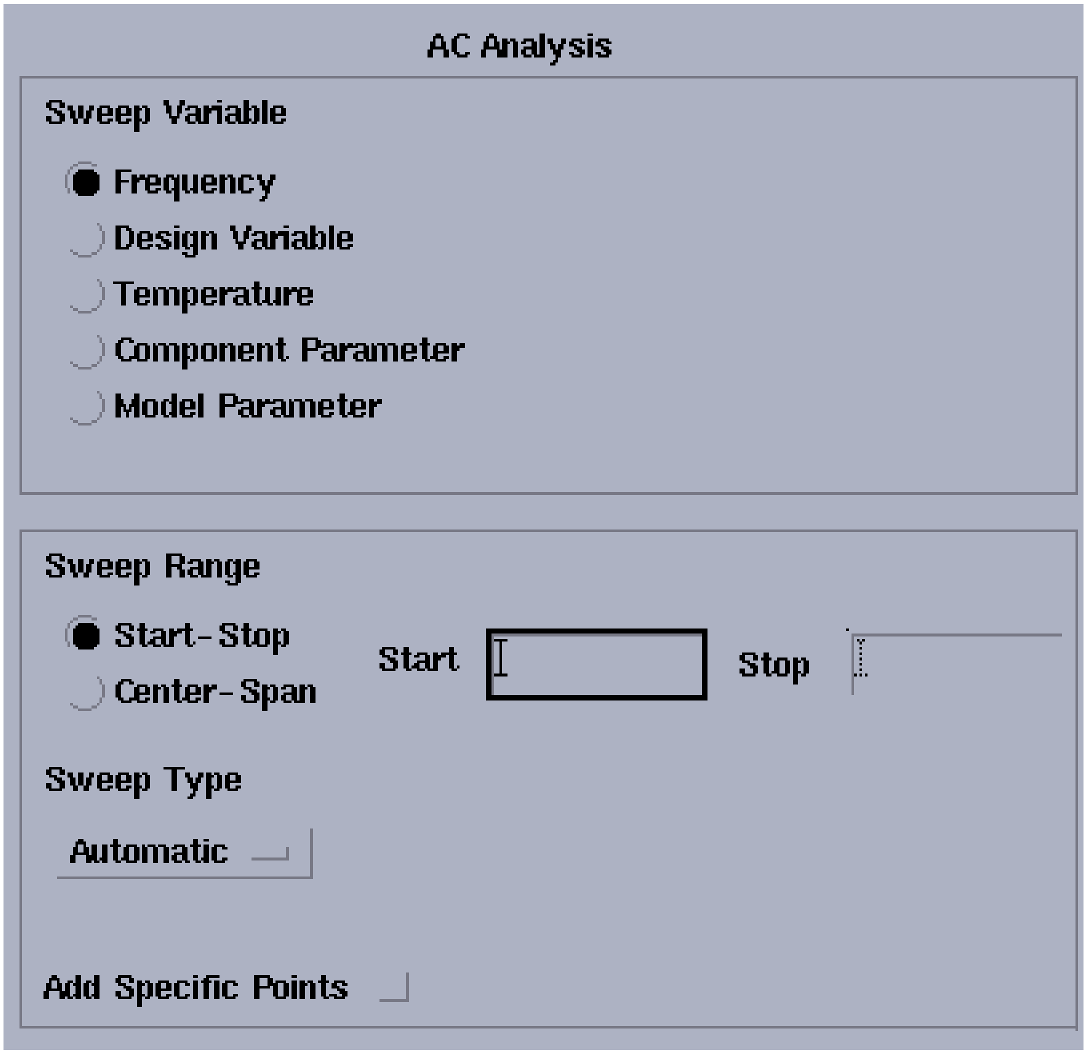
AC Small-Signal Analysis

AC small-signal analysis linearizes the circuit about the DC operating point and computes the response to a given small sinusoidal stimulus. Spectre can perform the analysis while sweeping a parameter.

The parameter can be a frequency, a design variable, temperature, a component instance parameter, or a component model parameter. If changing a parameter affects the DC operating point, the operating point is recomputed on each step.

To set up an AC small-signal analysis,

1. Choose *ac* from the Choosing Analyses form to display the appropriate options.



AC Analysis

Sweep Variable

● Frequency
○ Design Variable
○ Temperature
○ Component Parameter
○ Model Parameter

Sweep Range

● Start-Stop
○ Center-Span

Start
Stop

Sweep Type

Automatic

Add Specific Points □

2. Choose a sweep variable option and specify any necessary parameters.

   - If you do not sweep the frequency, specify the frequency at which to sweep the variable.
   - If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
   - If you sweep a component, specify the parameter to sweep. Click *Select Component* to click in the Schematic window and select the component.
   - If you sweep a model parameter, enter the model and parameter names.

3. Specify the sweep range and type.

Enter the start and stop points of the range or the center and span of the range.

The sweep type options are mapped to Spectre statements:

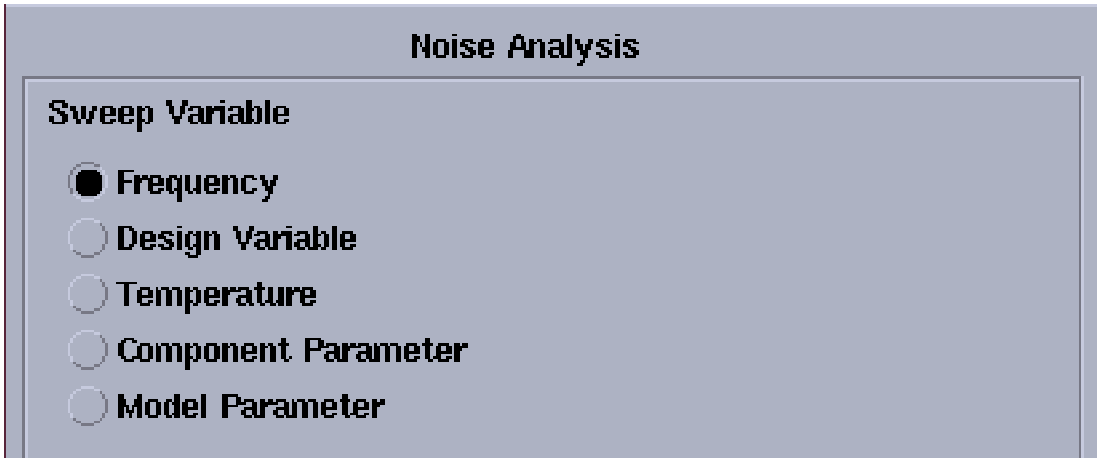
- ❑ Linear + Step Size = step
- ❑ Linear + Number of Steps = lin
- ❑ Logarithmic + Points Per Decade = dec
- ❑ Logarithmic + Number of Steps = log
- ❑ Add Specific Points = values=[…]

4. Click *Options* to select the Spectre options controlling the simulation.
5. Click *Enabled* and *Apply*.

Noise Analysis

The noise analysis linearizes the circuit about the DC operating point and computes the total-noise spectral density at the output. If you specify an input probe, the transfer function and the input-referred noise for an equivalent noise-free network is computed. To set up a noise analysis,

1. Choose a sweep variable option and specify any necessary parameters.



Noise Analysis

Sweep Variable

● Frequency
○ Design Variable
○ Temperature
○ Component Parameter
○ Model Parameter

- ❑ If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- ❑ If you sweep a design variable, fill out the name of the design variable, or choose from the list box after pressing the select button.

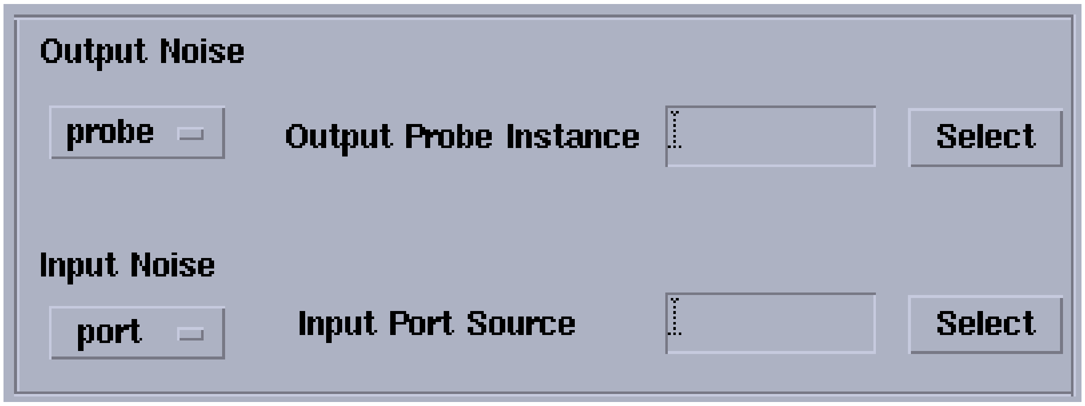
- If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- If you sweep a model parameter, enter the model and parameter names.

2. Specify the sweep range and type.

   The sweep type options are mapped to Spectre statements:

   - Linear + Step Size = step
   - Linear + Number of Steps = lin
   - Logarithmic + Points Per Decade = dec
   - Logarithmic + Number of Steps = log
   - Add Specific Points = values=[…]

3. Choose an *Output Noise* option.



Output Noise

probe Output Probe Instance

Input Noise

port Input Port Source

   - To measure the output noise voltage, click *voltage* in the cyclic field, and specify values for *Positive Output Node* and *Negative Output Node*, and click a net in the schematic.
   - To measure the output noise probe, click *probe* in the cyclic field, and click *Select* opposite *Output Probe Instance*, and click a voltage source in the schematic.

   Note: While selecting nodes, select the nodes/nets around the desired instance.

4. Optionally, choose an *Input Noise* option.

   - Choose *voltage*, *current*, or *port*.
   - Click *Select* for *Input Voltage Source* or *Input Current Source* or *Input Port Source*.

- ❑ Click a source or port in the schematic.
- ❑ Click *Apply*.

5. Click *Options* to set the spectre options controlling noise simulation.
6. Click *Apply*.

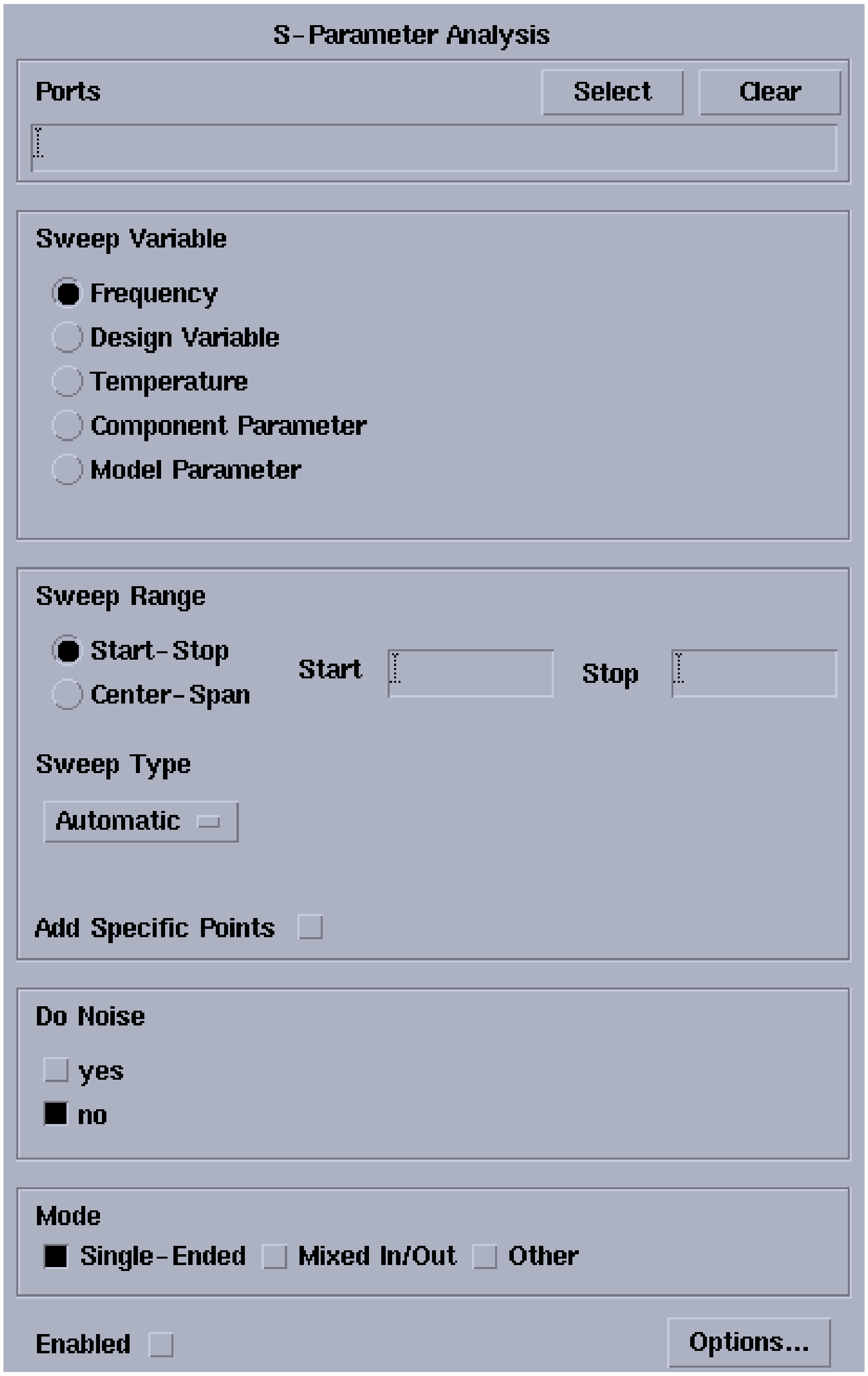
S-Parameter Analysis

The S-parameter analysis linearizes the circuit about the DC operating point and computes S-parameters of the circuit taken as an N-port. The psin instances (netlist-to-Spectre port statements) define the ports of the circuit. Each active port is turned on sequentially, and a linear small-signal analysis is performed. The Spectre simulator converts the response of the circuit at each active port into S-parameters and prints these parameters. There must be at least one active port (analogLib psin instance) in the circuit.

The parameter can be a frequency, a design variable, temperature, a component instance parameter, or a component model parameter. If changing a parameter affects the DC operating point, the operating point is recomputed on each step.

To set up an S-parameter analysis,

1. Choose *sp* from the *Choosing Analyses* form to display the appropriate options.



Ports

Sweep Variable
● Frequency
○ Design Variable
○ Temperature
○ Component Parameter
○ Model Parameter

Sweep Range
● Start-Stop
○ Center-Span
Start
Stop

Sweep Type
Automatic

Add Specific Points □

Do Noise
□ yes
■ no

Mode
■ Single-Ended □ Mixed In/Out □ Other

Enabled □

2. Specify the list of active *Ports*. In this field, the ports are numbered sequentially beginning with one, in the order given. Otherwise, all ports present in the circuit are active and the port numbers used are those that were assigned on the port statements.

3. Choose a sweep variable option and specify any necessary parameters.

- ❑ If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- ❑ If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
- ❑ If you sweep a component, specify the parameter to sweep. Click *Select Component* to select the component in the Schematic window.
- ❑ If you sweep a model parameter, enter the model and parameter names.

4. Specify the sweep range and type.

Enter the start and stop points of the range or the center and span of the range.

The sweep type options are mapped to Spectre statements:

- ❑ Linear + Step Size = step
- ❑ Linear + Number of Steps = lin
- ❑ Logarithmic + Points Per Decade = dec
- ❑ Logarithmic + Number of Steps = log
- ❑ Add Specific Points = values=[…]

5. Click *Options* to select the Spectre options controlling the simulation.

6. Click the *Do Noise* radio button to perform noise analysis.

7. Select the Mode. The available modes are:

- ❑ Single-Ended
- ❑ Mixed In/Out
- ❑ Other

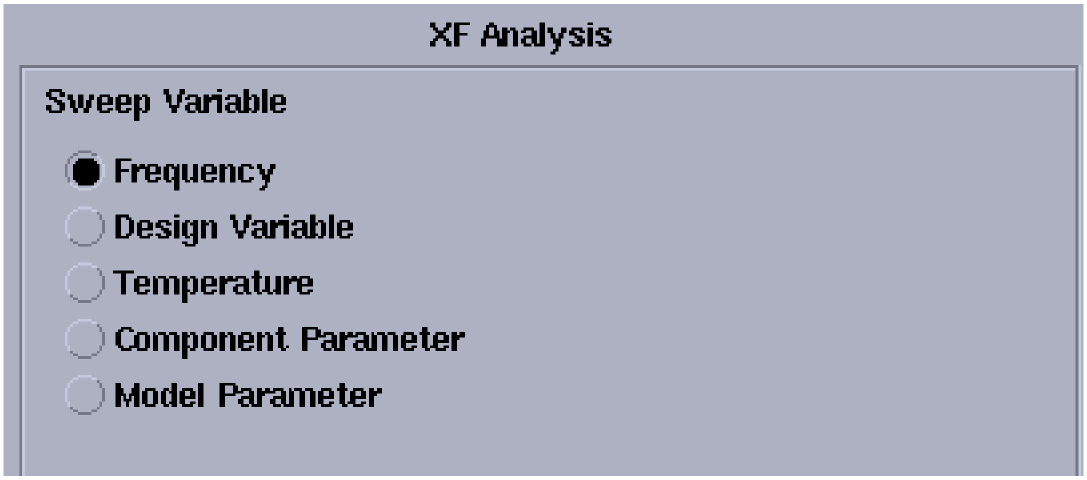
Single-Ended is the default mode. Select Mixed In/Out, if the required sp data is in mixed mode. If you have customized data, specify the Other option.

8. Click *Enabled* and *Apply*.

Transfer Function Analysis

The transfer function, or xf, analysis linearizes the circuit about the DC operating point and performs a small-signal analysis that calculates the transfer function from every independent source or instance terminal in the circuit to a designated output. The variable of interest at the output can be voltage or current.

1. Select a sweep variable option and specify any necessary parameters.



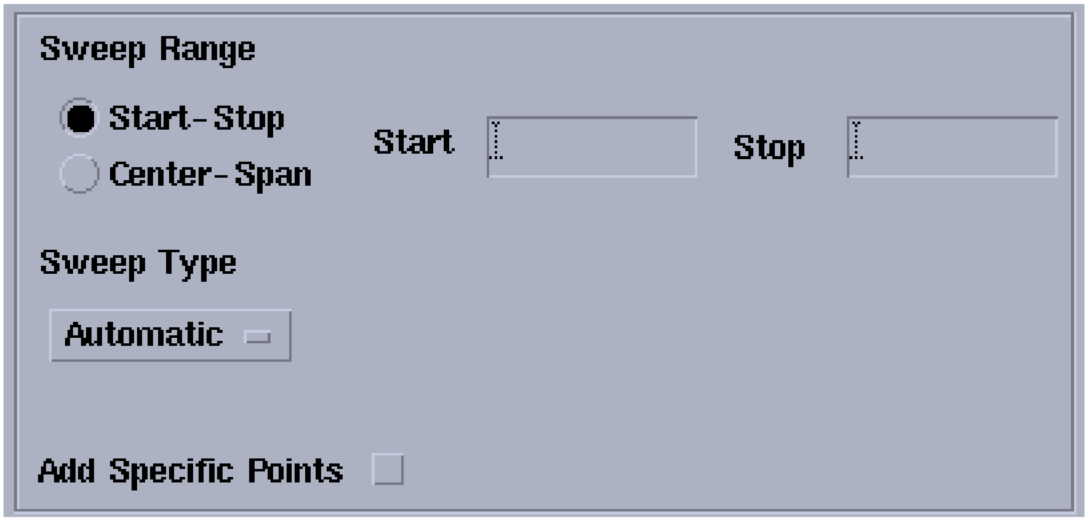
XF Analysis

Sweep Variable

● Frequency
○ Design Variable
○ Temperature
○ Component Parameter
○ Model Parameter

- If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
- If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- If you sweep a model parameter, enter the model and parameter names.

2. Specify the sweep range and type.



Sweep Range

● Start-Stop

○ Center-Span

Start

Stop

Sweep Type

Automatic

Add Specific Points □

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[…]

3. Choose *voltage* or *probe* for *Output*.

- To measure the output voltage, click *Select* opposite *Positive Output Node* and click a net in the schematic.
- To measure the output probe, click *probe*, click *Select* opposite *Output Probe Instance,* and click an instance in the schematic.

Note: While selecting nodes, select the nodes/nets around the desired instance.

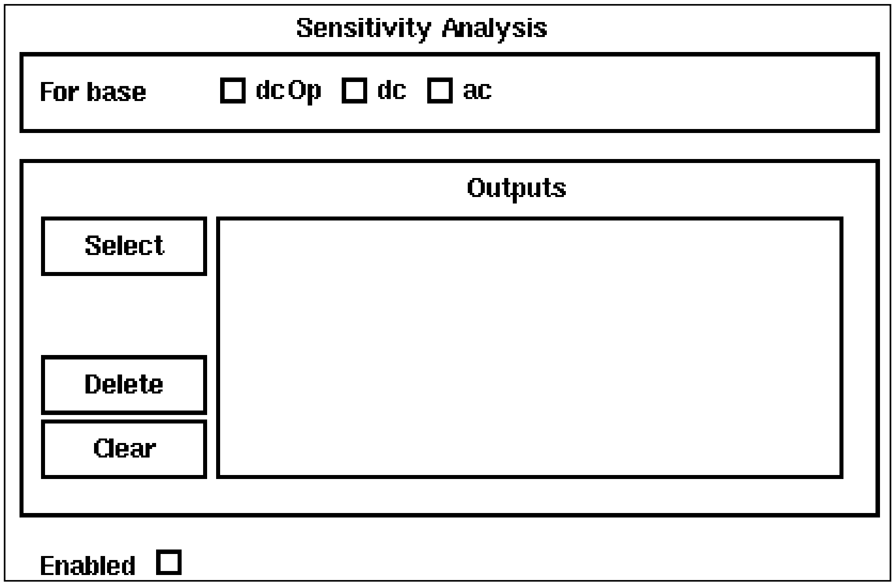
4. Click *Options* to set the spectre options controlling transfer function simulation.

5. Click *Apply*.

Sensitivity Analysis

Sensitivity analysis lets a designer see which parameters in a circuit most affect the specified outputs. It is typically used to tune a design to increase or decrease certain design goals. You might run a sensitivity analysis to determine which parameters to optimize using the optimizer.

1. Choose the *sens* radio button on the Choosing Analyses form. The form redraws:



2. Choose which types of sensitivities you want to calculate.

   In the *For base* field, choose any of the analyses on which you want to perform a sensitivity analysis. The available analyses are *dcOp* (DC operating point), *dc*, and *ac*.

   Before you run a sensitivity analysis, you must run the corresponding base analysis.

3. Click *Select* to select the outputs you want to measure.

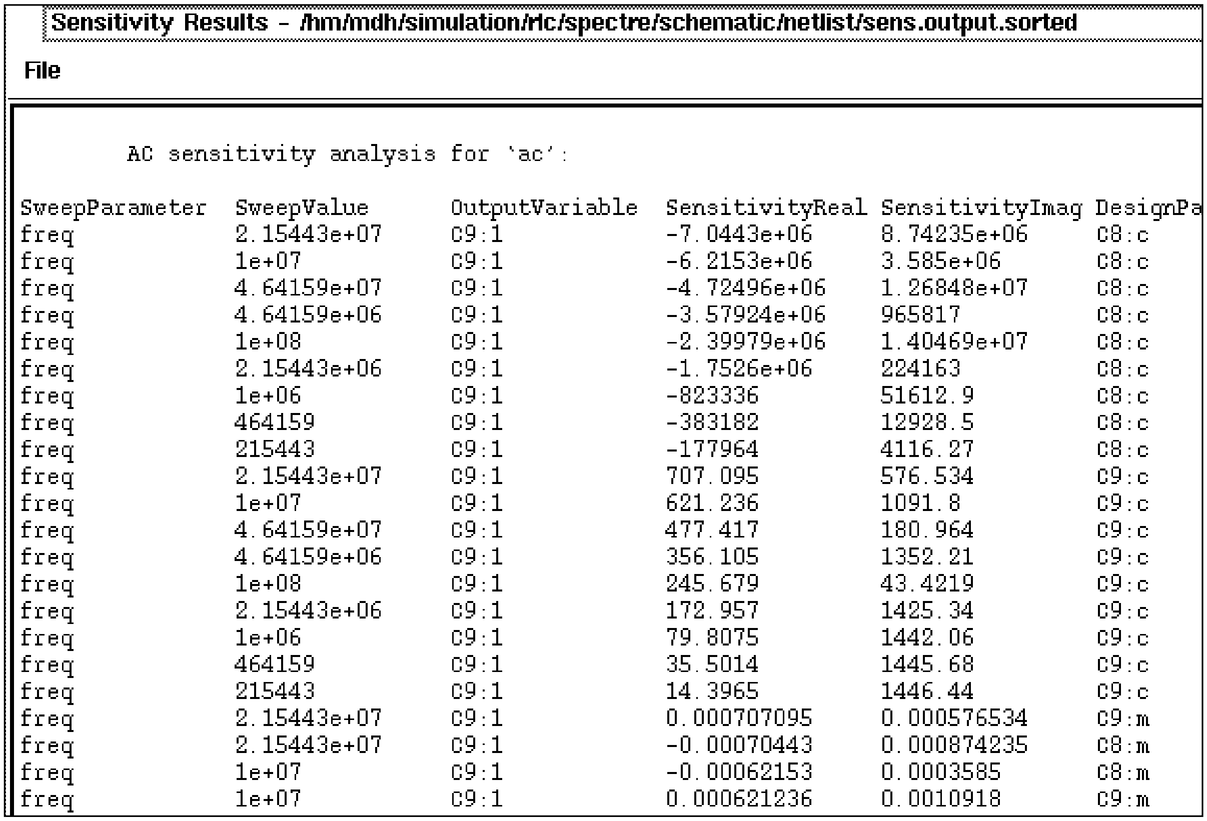
   Select prompts you to select outputs by clicking on their instance in the schematic. Outputs can be any nets or ports. When you click *Select*, the Schematic window moves to the front of the screen. The Schematic window must be open before you can select any outputs. Use the `Esc` key to end selection.

4. (Optional) In the Simulation window, choose *Simulation – Options – Analog* to open the Simulator Options form. Scroll down in the form to find the sensitivity options.

   Type a filename in the *sensfile* field to specify a filename for the Spectre sensitivity results. This file is in ASCII format, and is generated in the psf directory. If you do not specify a value, the file is named `sens.output` by default.

5. View your results.

From the simulation window, choose *Results – Print – Sensitivity*. The results display in a print window.


Sensitivity Results - /hm/mdh/simulation/rlc/spectre/schematic/netlist/sens.output.sorted
File

```
        AC sensitivity analysis for `ac':

SweepParameter  SweepValue      OutputVariable  SensitivityReal SensitivityImag DesignPa
freq            2.15443e+07     C9:1            -7.0443e+06     8.74235e+06     C8:c
freq            1e+07           C9:1            -6.2153e+06     3.585e+06       C8:c
freq            4.64159e+07     C9:1            -4.72496e+06    1.26848e+07     C8:c
freq            4.64159e+06     C9:1            -3.57924e+06    965817          C8:c
freq            1e+08           C9:1            -2.39979e+06    1.40469e+07     C8:c
freq            2.15443e+06     C9:1            -1.7526e+06     224163          C8:c
freq            1e+06           C9:1            -823336         51612.9         C8:c
freq            464159          C9:1            -383182         12928.5         C8:c
freq            215443          C9:1            -177964         4116.27         C8:c
freq            2.15443e+07     C9:1            707.095         576.534         C9:c
freq            1e+07           C9:1            621.236         1091.8          C9:c
freq            4.64159e+07     C9:1            477.417         180.964         C9:c
freq            4.64159e+06     C9:1            356.105         1352.21         C9:c
freq            1e+08           C9:1            245.679         43.4219         C9:c
freq            2.15443e+06     C9:1            172.957         1425.34         C9:c
freq            1e+06           C9:1            79.8075         1442.06         C9:c
freq            464159          C9:1            35.5014         1445.68         C9:c
freq            215443          C9:1            14.3965         1446.44         C9:c
freq            2.15443e+07     C9:1            0.000707095     0.000576534     C9:m
freq            2.15443e+07     C9:1            -0.00070443     0.000874235     C8:m
freq            1e+07           C9:1            -0.00062153     0.0003585       C8:m
freq            1e+07           C9:1            0.000621236     0.0010918       C9:m
```

DC Mismatch Analysis

The *dcmatch* analysis option performs DC device mis-matching analysis for a given output. It computes the deviation in the DC operating point of the circuit caused by mismatch in the devices. Users need to specify mismatch parameters in their model cards for each device contributing to the deviation. The analysis uses the device mismatch models to construct equivalent mismatch current sources to all the devices that have mismatch modeled. These current sources will have zero mean and some variance. The variance of the current sources is computed according to mismatch models. The analysis computes the 3-sigma variance of dc voltage or current due to the mismatch current sources.

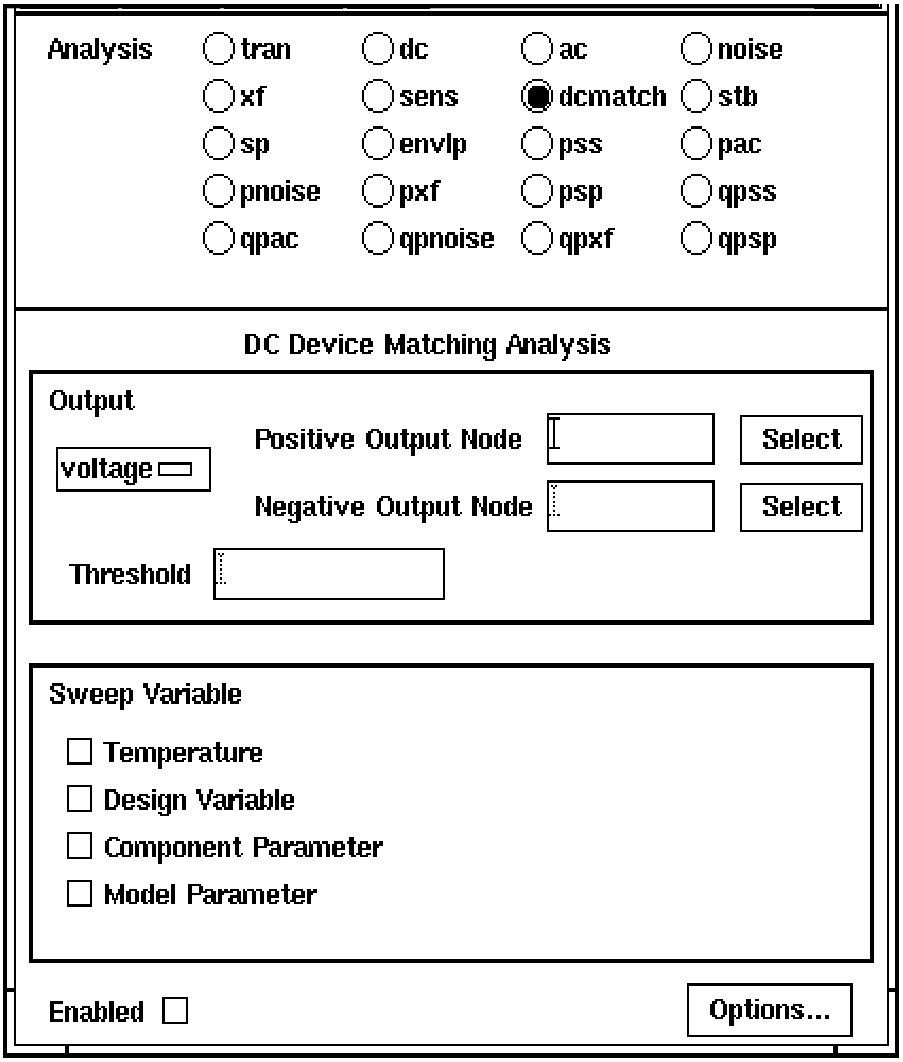
To set up a DC Mismatch analysis for the Spectre simulator,

1. Select *Analyses – Choose* from the Virtuoso® *Analog Design Environment* window.

   The *Choosing Analyses* form appears.

2. Choose *dcmatch*.

The *Choosing Analyses* form re-displays to show the fields that are required for DC mismatch analysis.



Analysis ○ tran ○ dc ○ ac ○ noise
○ xf ○ sens ● dcmatch ○ stb
○ sp ○ envlp ○ pss ○ pac
○ pnoise ○ pxf ○ psp ○ qpss
○ qpac ○ qpnoise ○ qpxf ○ qpsp

DC Device Matching Analysis

Output

voltage

Positive Output Node

Negative Output Node

Threshold

Sweep Variable

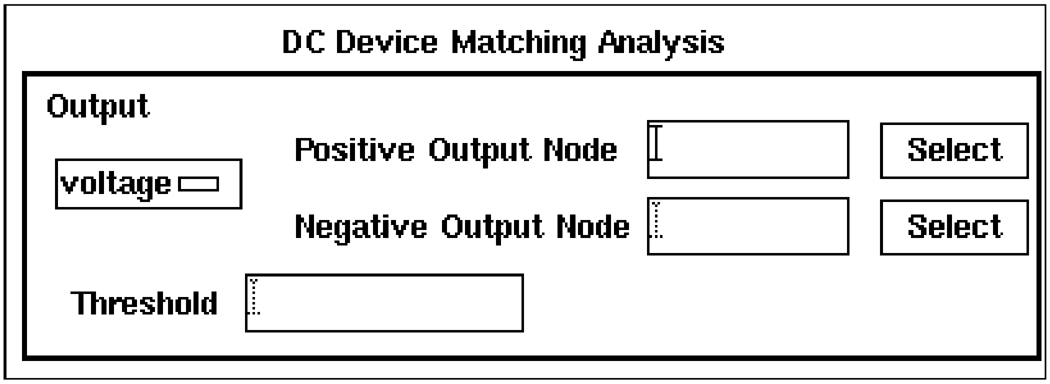
□ Temperature
□ Design Variable
□ Component Parameter
□ Model Parameter

Enabled □

3. Specify the output in the *Output* section of the form. You can choose either *Voltage* or *Probe* in the cyclic drop down field.

   To specify a *Voltage* output,

a. Choose *Voltage* in the cyclic drop down field

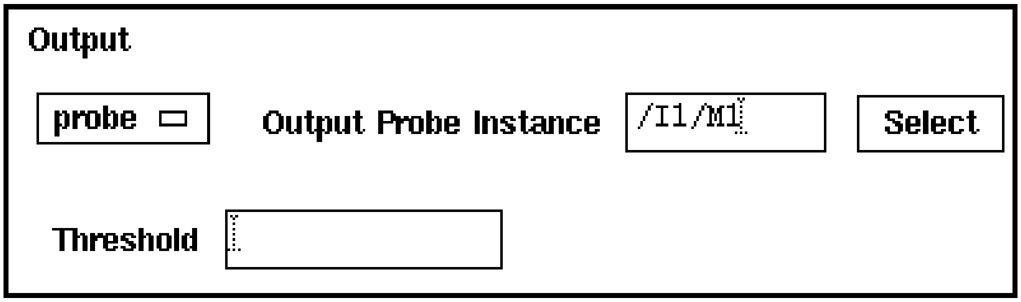


b. Click *Select* opposite *Positive Output Node* and click a net in the schematic for the positive output node. Optionally, click *Select* opposite *Negative Output Node* and click a net in the schematic.

To specify a current output,

a. Choose *Probe* in the cyclic drop down field.

b. Click *Select* opposite O*utput Probe Instance* and click a probe in the schematic.



Output Probe Instance: /I1/M1

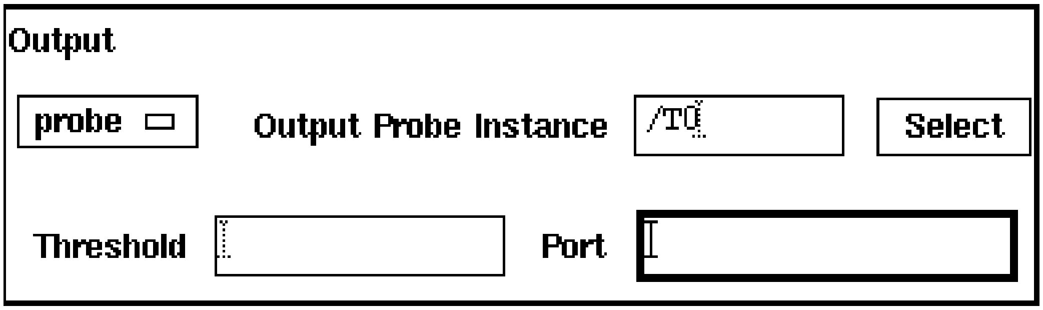
Note: This probe device selected needs to have its terminal currents as network variables. For any other device selection, a warning will be displayed in the CIW stating that the selected object is not a valid type.

Valid Spectre Devices and corresponding analogLib Cells.

Device	Corresponding analogLib Cells
inductor	ind, pinductor
vsource	vdc, vpulse, vpwl, vpwlf, vsin, vexp, vsource
switch	sp1tswitch, sp2tswitch, sp3tswitch, sp4tswitch

Device	Corresponding analogLib Cells
tline	tline
controlled voltage source	vcvs, ccvs, sccvs, svcvs, zccvs, zvcvs, pvcvs, pvcvs2, pvcvs3, pccvs
iprobe	iprobe

If the selected probe has multiple ports (for example tline), you can specify the port number in the *Port* field.



Output

probe ▭ Output Probe Instance /T0 Select

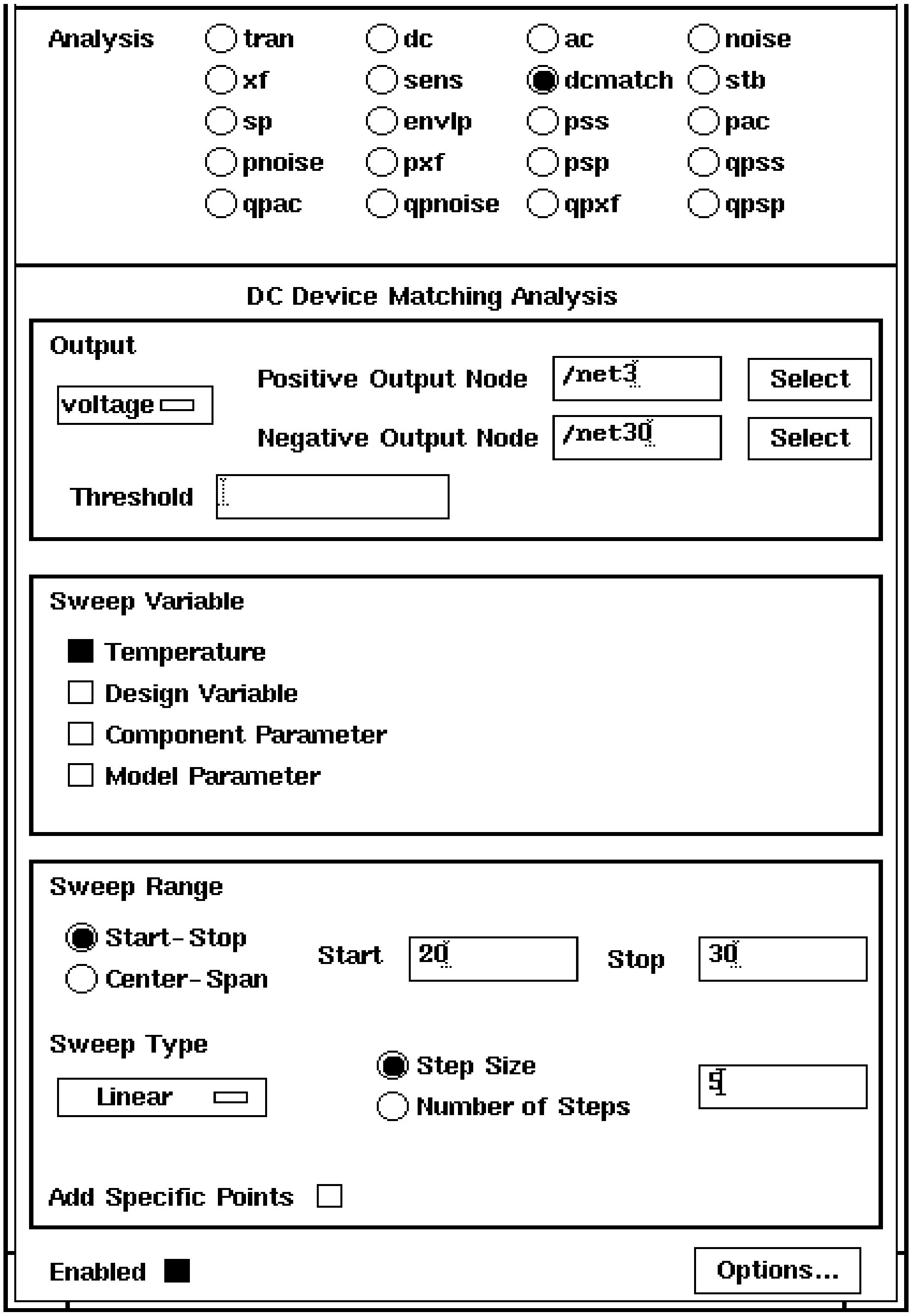
Threshold Port

Refer to the Component Description Format User Guide for information on creating more library components selectable for an analysis.

4. Specify a value in the *Threshold* field to control the number of devices displayed in the output log. The value should be a positive number less than or equal to 1. All devices whose relative contribution falls below the threshold specified are not displayed in the output log.

5. Choose a parameter to sweep in the analysis. The parameters that you can select are *Temperature*, *Design Variable*, *Component Parameter* and *Model Parameter*.

When any of these parameters are selected, the *Sweep Range* section is displayed. Also, the form re-displays according to the parameter that is selected.



Analysis ○ tran ○ dc ○ ac ○ noise
○ xf ○ sens ● dcmatch ○ stb
○ sp ○ envlp ○ pss ○ pac
○ pnoise ○ pxf ○ psp ○ qpss
○ qpac ○ qpnoise ○ qpxf ○ qpsp

DC Device Matching Analysis

Output

voltage

Positive Output Node /net3 Select

Negative Output Node /net30 Select

Threshold

Sweep Variable

■ Temperature
□ Design Variable
□ Component Parameter
□ Model Parameter

Sweep Range

● Start-Stop
○ Center-Span

Start 20 Stop 30

Sweep Type

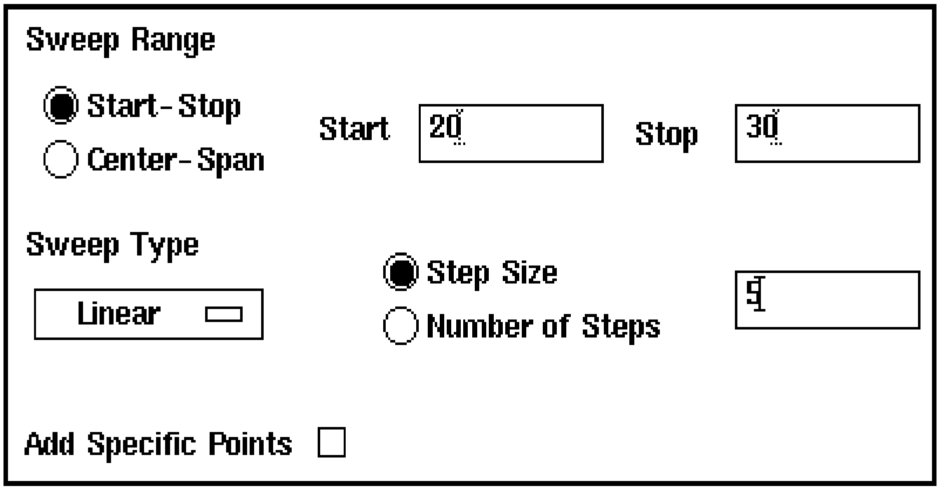
Linear

● Step Size 5
○ Number of Steps

Add Specific Points □

Enabled ■ Options...

6. Specify the *Sweep Range and Sweep Type* for the swept parameter.



Sweep Range

● Start-Stop Start 20 Stop 30

○ Center-Span

Sweep Type

Linear

● Step Size 5

○ Number of Steps

Add Specific Points □

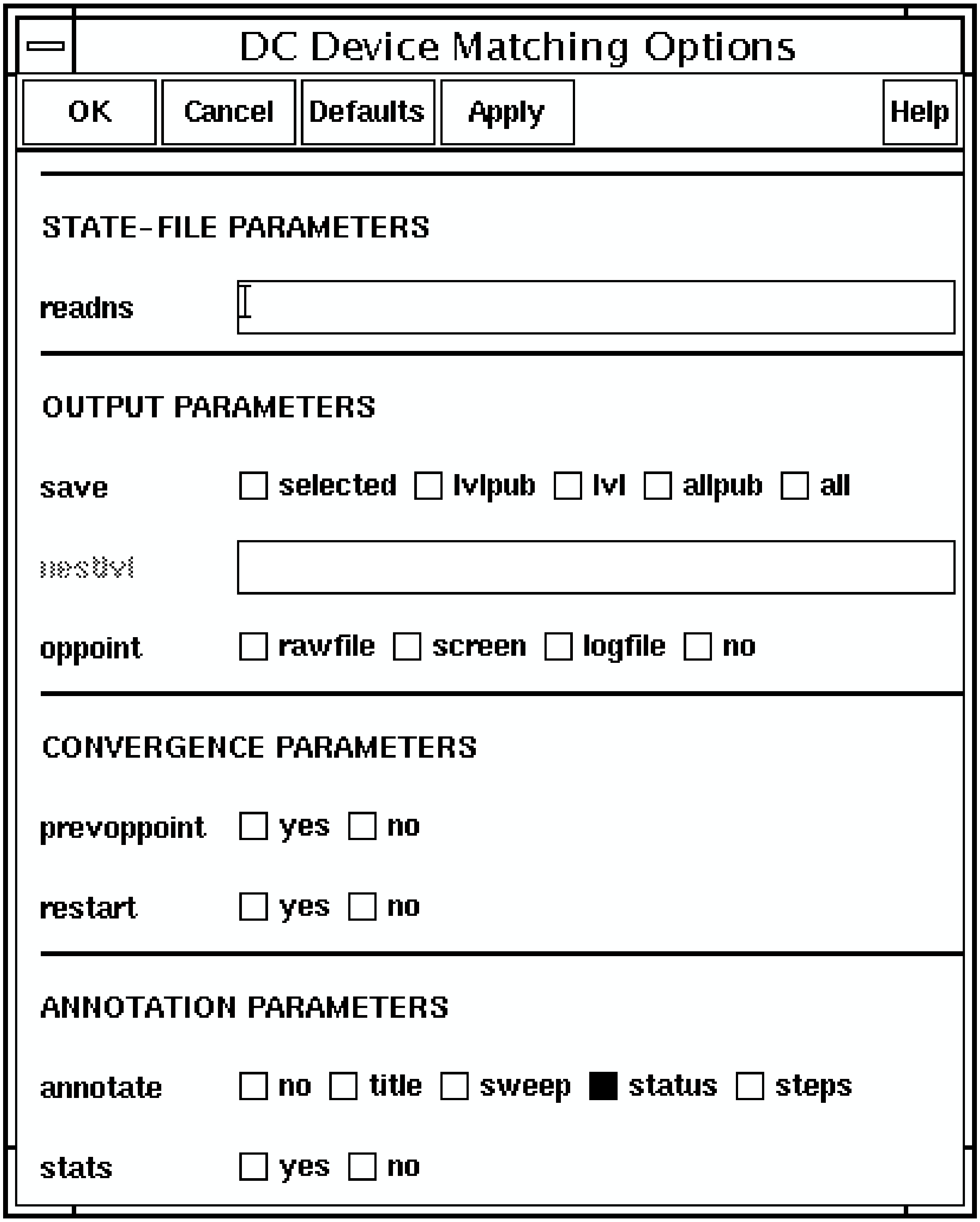
Enter the start and stop points of the range or the center and span of the range.

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[…]

Note: By default, when no sweep variable is selected, the *Sweep Range* section is not displayed.

7. Click the *Options* button to open the *Options* form corresponding to the dcmatch analysis. The *DC Device Matching Options* form appears.



DC Device Matching Options

STATE-FILE PARAMETERS

readns

OUTPUT PARAMETERS

save □ selected □ lvlpub □ lvl □ allpub □ all

nestlvl

oppoint □ rawfile □ screen □ logfile □ no

CONVERGENCE PARAMETERS

prevoppoint □ yes □ no

restart □ yes □ no

ANNOTATION PARAMETERS

annotate □ no □ title □ sweep ■ status □ steps

stats □ yes □ no

Refer to the Spectre Circuit Simulator Reference for details.

8. Click *Apply*.

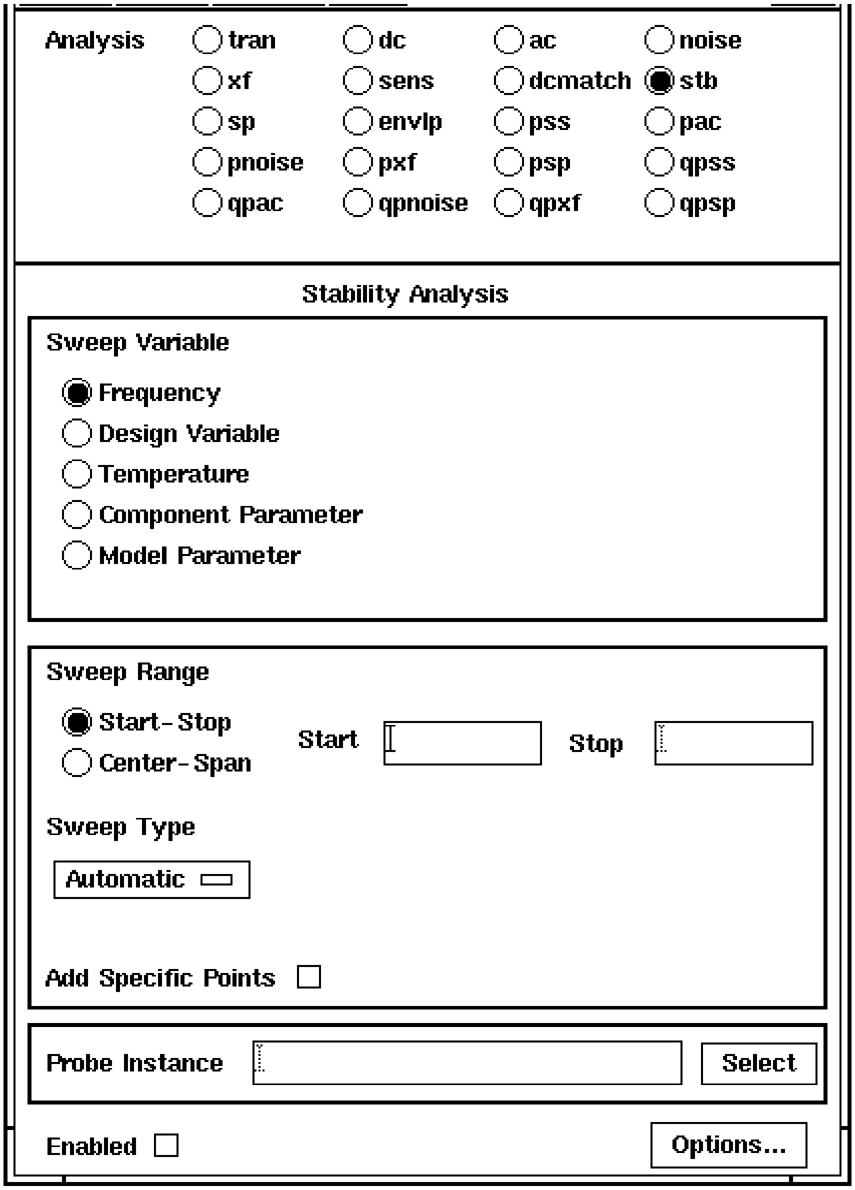
To access the results post simulation, choose *Results – Print – Mismatch Summary.*

Note: To print these results from OCEAN, use the OCEAN command, `dcmatchSummary`.

Stability Analysis

Stability analysis outputs the loop gain for the feedback loop or a gain device. To set up a stability analysis for the Spectre simulator,

1. Select *Analyses – Choose* from the Virtuoso® *Analog Design Environment* window. The *Choosing Analyses* form appears.
2. Choose *stb*. The *Choosing Analyses* form re-displays to show the fields that are required for the stability analysis.



Analysis ○ tran ○ dc ○ ac ○ noise
○ xf ○ sens ○ dcmatch ● stb
○ sp ○ envlp ○ pss ○ pac
○ pnoise ○ pxf ○ psp ○ qpss
○ qpac ○ qpnoise ○ qpxf ○ qpsp

Stability Analysis

Sweep Variable

● Frequency
○ Design Variable
○ Temperature
○ Component Parameter
○ Model Parameter

Sweep Range

● Start-Stop
○ Center-Span

Start □ Stop □

Sweep Type

Automatic

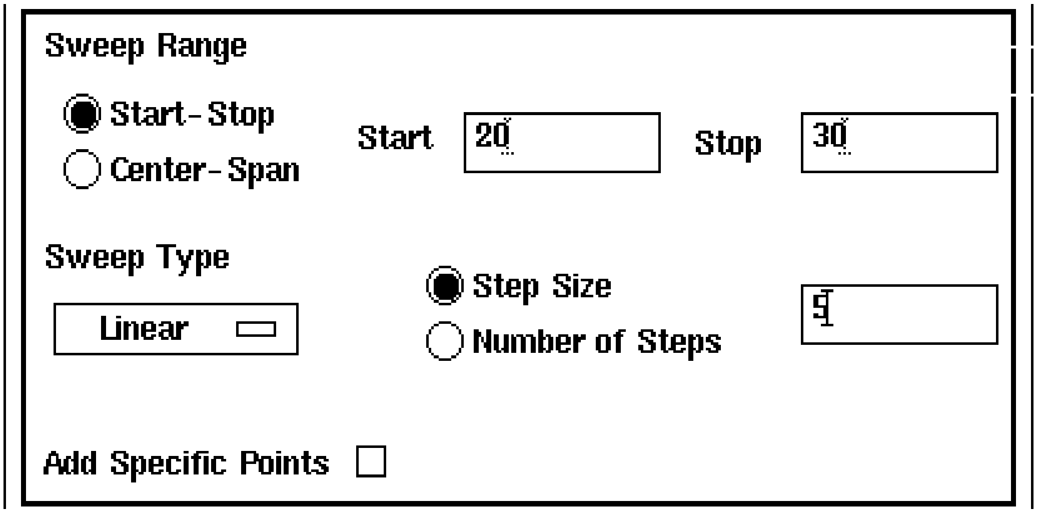
Add Specific Points □

Probe Instance □ Select

Enabled □ Options...

3. Choose a parameter to sweep in the analysis. The parameters that you can select are *Frequency*, *Design Variable*, *Temperature*, *Component Parameter* and *Model Parameter*. For any parameter other than frequency, you need to specify the frequency at which the analysis is to be performed. When the swept parameter is frequency, it also outputs the phase and gain margins if they can be calculated from the loop gain curve within the swept frequency values.

4. Specify the *Sweep Range and Sweep Type* for the swept parameter.



Sweep Range

● Start-Stop Start 20 Stop 30

○ Center-Span

Sweep Type

Linear ▭

● Step Size 5

○ Number of Steps

Add Specific Points □

Enter the start and stop points of the range or the center and span of the range.

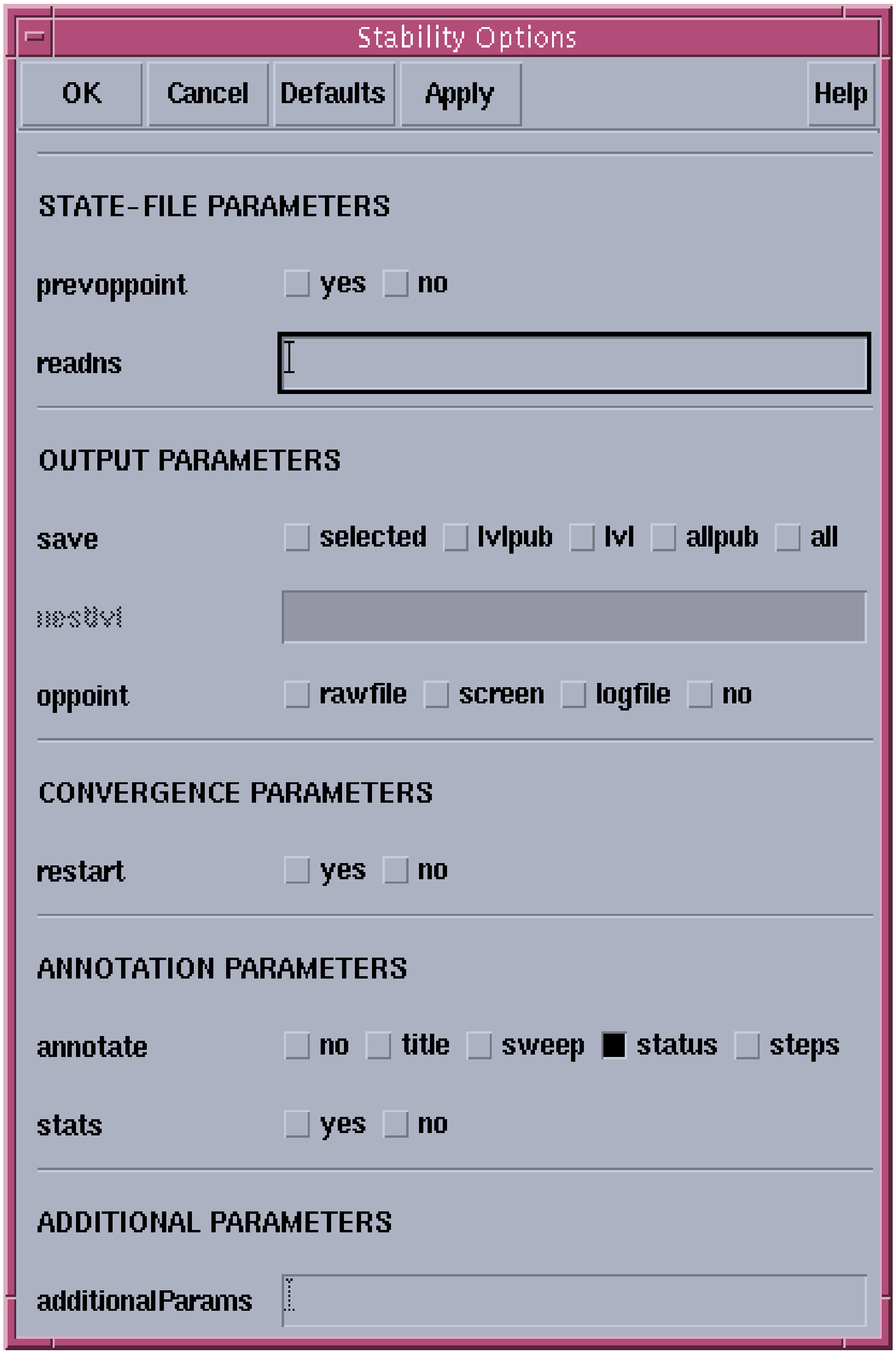
The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[…]

The form changes dynamically as per the current selection.

5. Specify a value in the *Probe Instance* text field. You can use the *Select* button to select the instance from the schematic and the name for the selected instance automatically appears in the text field.

6. Click the *Options* button to open the options form corresponding to the stability analysis. The *Stability Options* form appears.
.



STATE-FILE PARAMETERS

prevoppoint □ yes □ no

readns

OUTPUT PARAMETERS

save □ selected □ lvlpub □ lvl □ allpub □ all

nestlvl

oppoint □ rawfile □ screen □ logfile □ no

CONVERGENCE PARAMETERS

restart □ yes □ no

ANNOTATION PARAMETERS

annotate □ no □ title □ sweep ■ status □ steps

stats □ yes □ no

ADDITIONAL PARAMETERS

additionalParams

You can refer to the Spectre Circuit Simulator Reference for details.

7. Click *Apply*.

To access the results post simulation, choose *Results – Print – Stability Summary*

You can also access these results from *Results-Direct Plot*.

Pole Zero Analysis

Pole Zero Analysis is a useful method for studying the behavior of linear time invariant networks and can be applied to the design of analog circuits. Therefore, it can be used for determining stability of designs.

In *Pole Zero* analysis, a network is described by its network transfer function. For any linear time invariant network, it can be written in the general form:

$$H(S) = \frac{N(S)}{D(S)} = \frac{a_0 S^m + a_1 S^{m-1} + \ldots + a_m}{b_0 S^n + b_1 S^{n-1} + \ldots + b_n}$$

Similarly, in the factorized form:

$$H(S) = \frac{N(S)}{D(S)} = \frac{a_0}{b_0} \cdot \frac{(S+Z_1)(S+Z_2)}{(S+P_1)(S+P_2)} \cdots \frac{(S+Z_i)}{(S+P_i)} \cdots \frac{(S+Z_m)}{(S+P_m)}$$

Here, the roots of the numerator ***N(S)*** (that is, ***Z***) are called *zeros* of the network function. The roots of the denominator ***D(S)*** (that is, ***P***) are called the *poles* of the network function. ***S*** is the complex frequency.

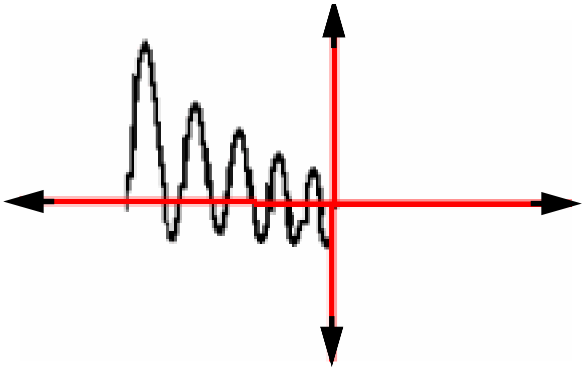
The behavior of the network depends upon the location of the poles and zeros on the complex S-plane. The poles are called natural frequencies of the network.

For example:

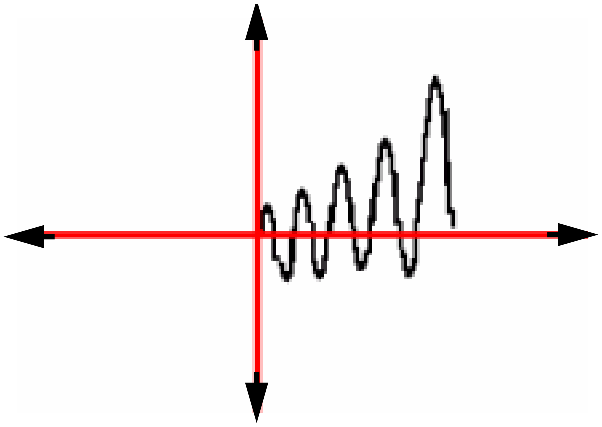
$$H(S) = \frac{(S-2)\cdot(S+1)}{S}$$

Here, the *zeros* are the values of ***H(S)*** which make it zero (***S***`=2` and ***S***`=-1`). The *poles* make ***H(S)*** go to infinity (the pole is at ***S***`=0`)

When all the poles have negative real parts, the poles are located on the left hand side of the XY plane. In this situation, the circuit is considered *stable*. The following diagram illustrates the behavior of a stable circuit:



In case there are poles present on the right hand side of the XY plane, the circuit is considered *unstable*. The following diagram illustrates the behavior of an unstable circuit.

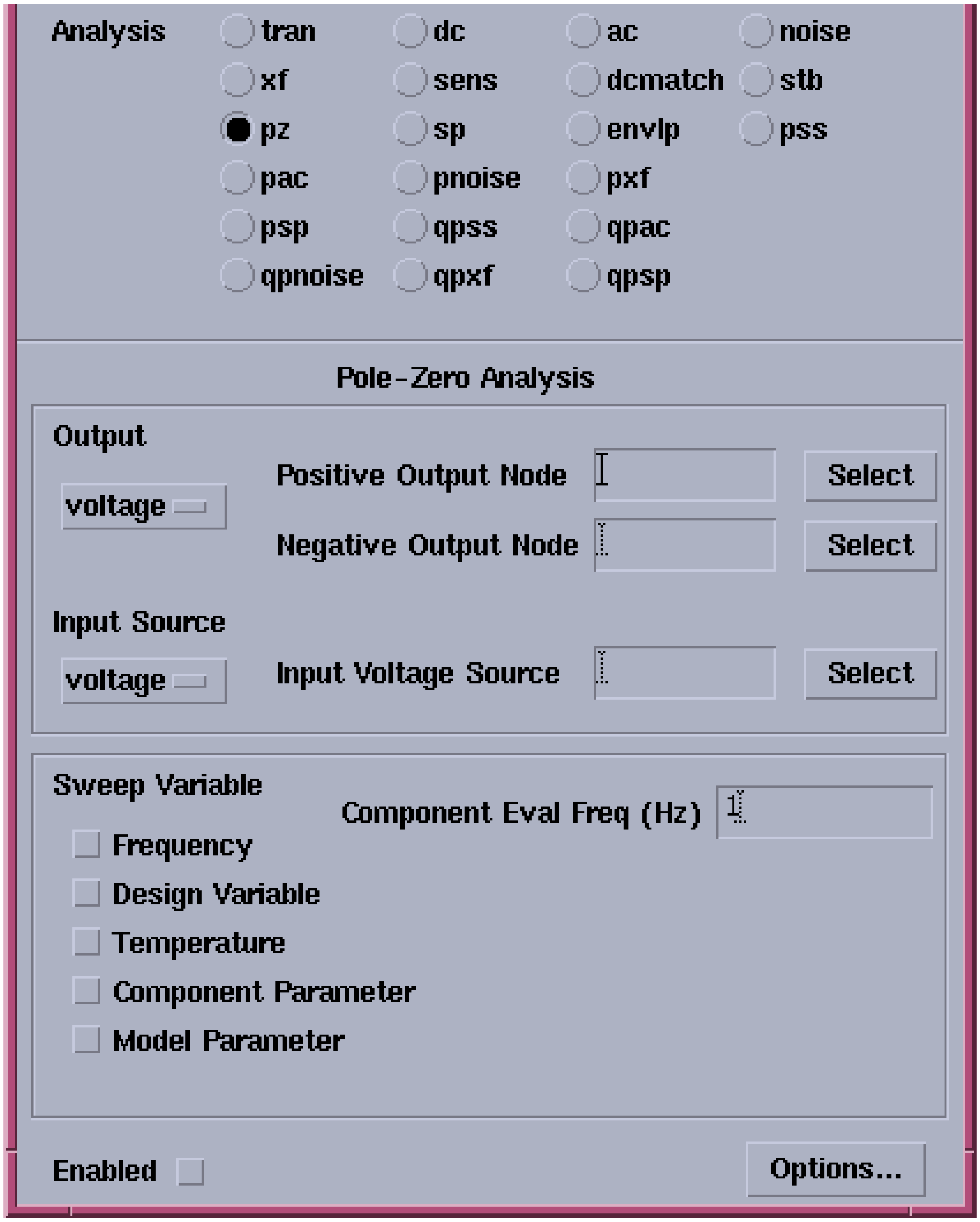


For absolute stability, there can be no poles with positive real parts. If there are poles with positive real parts the output signal may become unbounded.

To set up a *Pole Zero* analysis for the Spectre simulator,

1. Select *Analyses – Choose* from the Virtuoso® *Analog Design Environment* window. The *Choosing Analyses* form appears.

2. Select *pz.* The *Choosing Analyses* form re-displays to show the fields that are required for a *Pole Zero* analysis.



Analysis: ● pz

Output: voltage

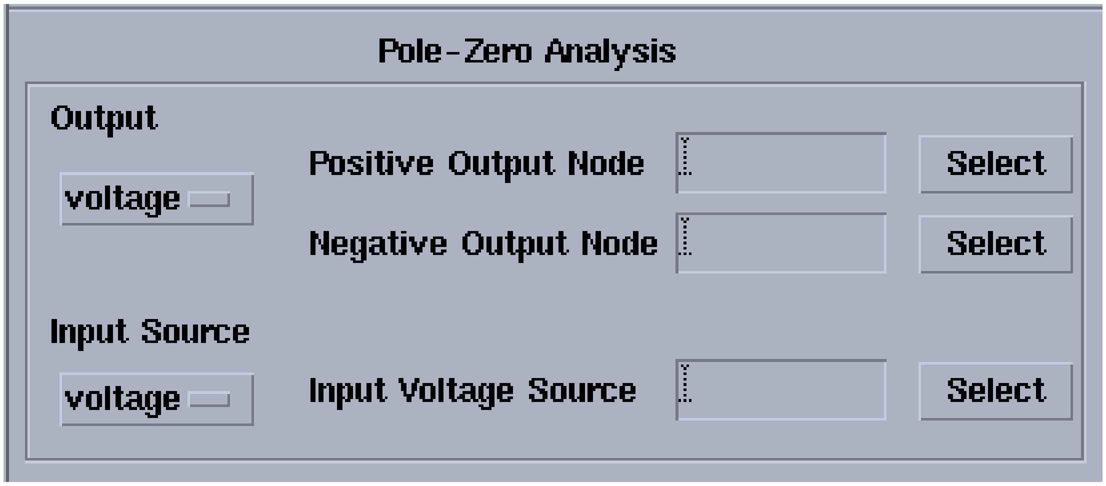
Input Source: voltage

Component Eval Freq (Hz): 1

3. Specify the output in the *Output* section of the form. You can choose either *Voltage* or *Probe* in the cyclic drop down field.

   To specify a *Voltage* output,

a. Choose *Voltage* in the cyclic drop down field.



Pole-Zero Analysis

Output

voltage

Positive Output Node

Negative Output Node

Input Source

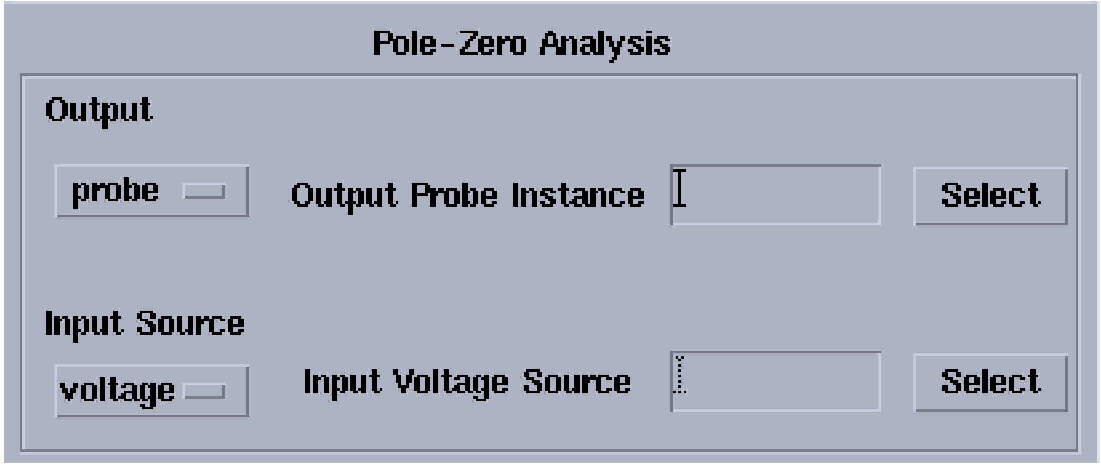
voltage

Input Voltage Source

b. Click *Select* opposite *Positive Output Node* and click a net in the schematic for the positive output node. Also, click *Select* opposite *Negative Output Node* and click a net in the schematic.

To specify a current output,

a. Choose *Probe* in the cyclic drop down field.



Pole-Zero Analysis

Output

probe

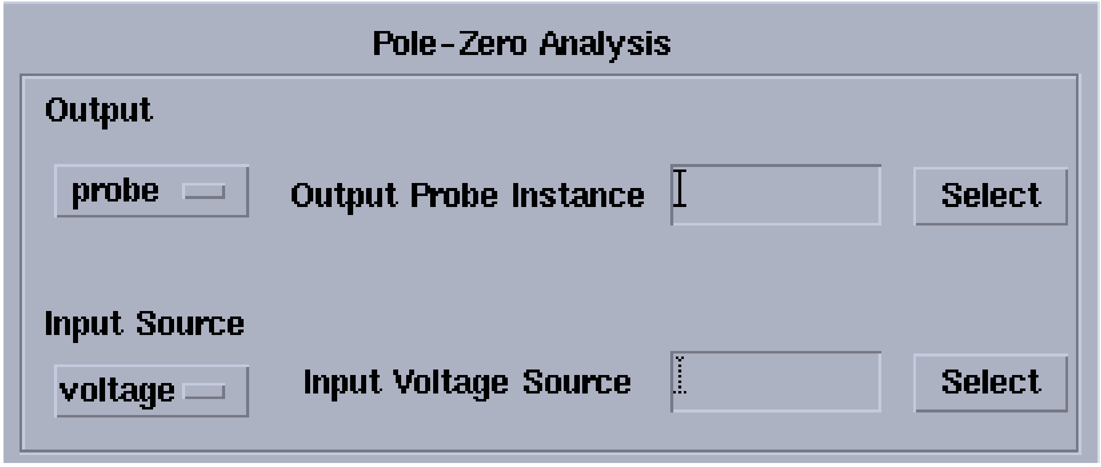
Output Probe Instance

Input Source

voltage

Input Voltage Source

b. Click *Select* opposite O*utput Probe Instance* and click an instance (with terminal currents as network variables) in the schematic.



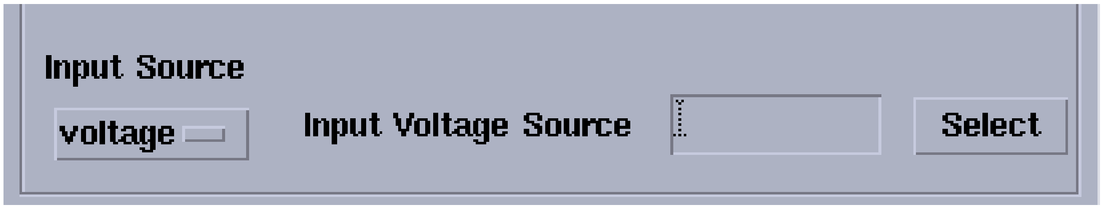
For any other device selection, a warning will be displayed in the CIW stating that the selected object is not a valid type.

Valid Spectre Devices and corresponding analogLib cells:

Device	Corresponding analogLib Cells
inductor	ind, pinductor
vsource	vdc, vpulse, vpwl, vpwlf, vsin, vexp, vsource
switch	sp1tswitch, sp2tswitch, sp3tswitch, sp4tswitch
tline	tline
controlled voltage source	vcvs, ccvs, sccvs, svcvs, zccvs, zvcvs, pvcvs, pvcvs2, pvcvs3, pccvs
iprobe	iprobe

When *tline* is the device selected, the *Output* section of the form re-displays to show the *porti* field. This parameter lets you specify a current output that is defined by the device terminal current. Since all of these are two-terminal devices, the current through one of the device terminals would be the same as through the other. The *tline* device is the only one that has more than two terminals.

4. Specify the input voltage or current source by selecting either *voltage* or *current* in the cyclic drop down *Input Source* field of the same form.

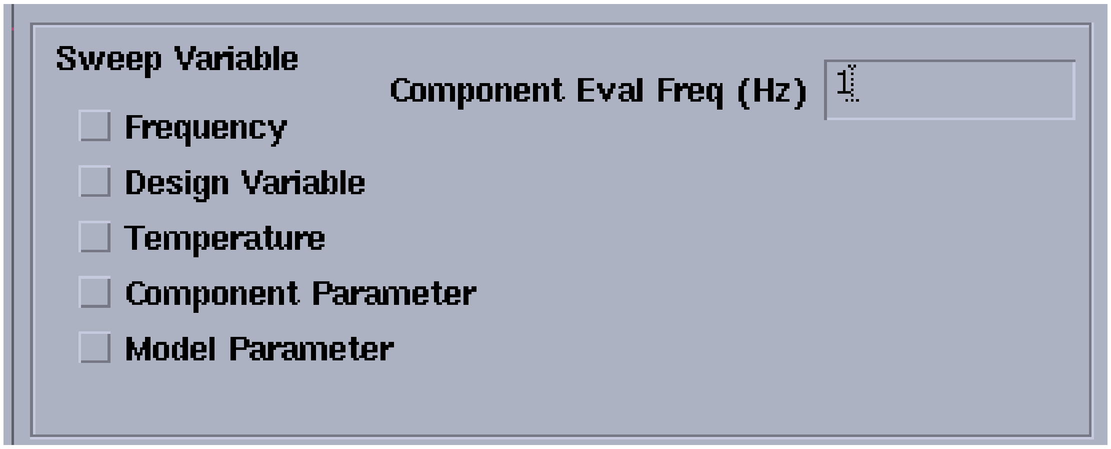


Input Source

voltage

Input Voltage Source

5. If you want to sweep a variable in conjunction with the Pole-Zero analysis, choose a parameter to sweep. The parameters that you can select are *Frequency, Design Variable*, *Temperature, Component Parameter* and *Model Parameter*.

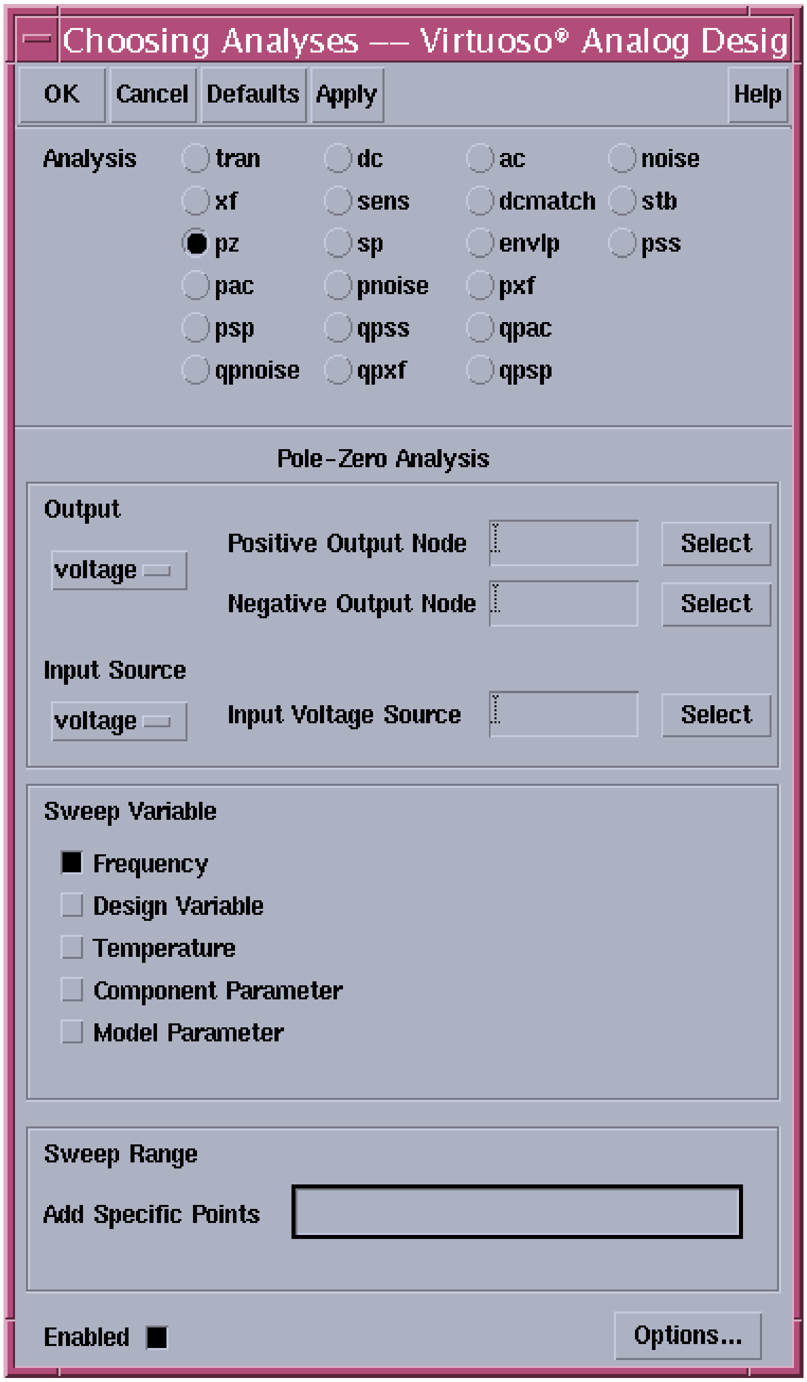


Sweep Variable

□ Frequency
□ Design Variable
□ Temperature
□ Component Parameter
□ Model Parameter

Component Eval Freq (Hz) 1

When any of these parameters are selected, the form re-displays according to the parameter that is selected:



Choosing Analyses –– Virtuoso® Analog Desig

OK | Cancel | Defaults | Apply | Help

Analysis

○ tran ○ dc ○ ac ○ noise
○ xf ○ sens ○ dcmatch ○ stb
● pz ○ sp ○ envlp ○ pss
○ pac ○ pnoise ○ pxf
○ psp ○ qpss ○ qpac
○ qpnoise ○ qpxf ○ qpsp

Pole-Zero Analysis

Output

voltage

Positive Output Node Select

Negative Output Node Select

Input Source

voltage

Input Voltage Source Select

Sweep Variable

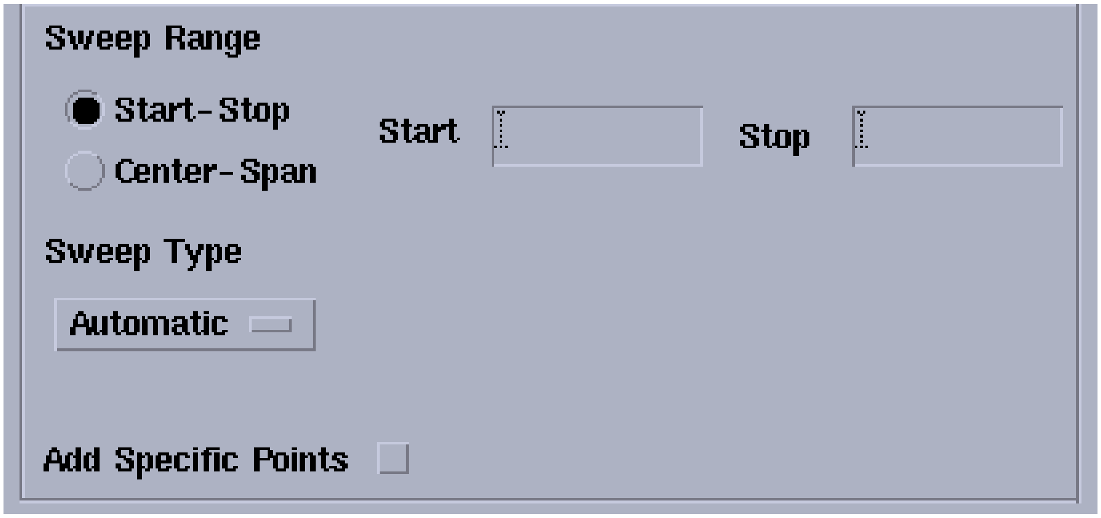
☒ Frequency
□ Design Variable
□ Temperature
□ Component Parameter
□ Model Parameter

Sweep Range

Add Specific Points

Enabled ☒ Options...

6. Specify the *Sweep Range and Sweep Type* for the swept parameter (*Design Variable*, *Temperature, Component Parameter* or *Model Parameter*).



Sweep Range

● Start-Stop
○ Center-Span

Start
Stop

Sweep Type

Automatic

Add Specific Points □

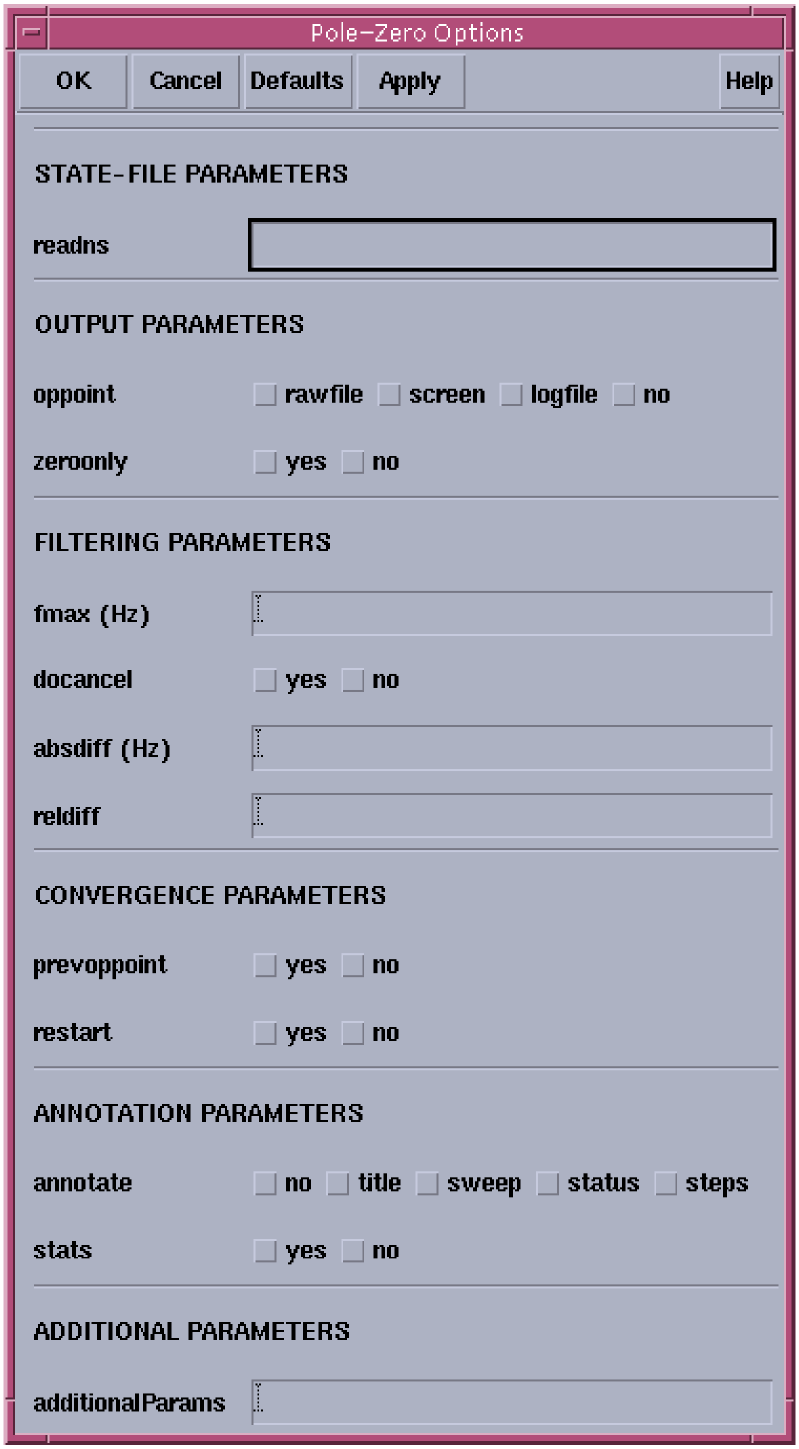
Enter the start and stop points of the range or the center and span of the range.

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[…]

By default, when no sweep variable is selected, the *Sweep Range* section is not displayed.

7. Click the *Options* button to open the *Options* form corresponding to the pz analysis. The *Pole-Zero Options* form appears.



STATE-FILE PARAMETERS

readns

OUTPUT PARAMETERS

oppoint □ rawfile □ screen □ logfile □ no

zeroonly □ yes □ no

FILTERING PARAMETERS

fmax (Hz)

docancel □ yes □ no

absdiff (Hz)

reldiff

CONVERGENCE PARAMETERS

prevoppoint □ yes □ no

restart □ yes □ no

ANNOTATION PARAMETERS

annotate □ no □ title □ sweep □ status □ steps

stats □ yes □ no

ADDITIONAL PARAMETERS

additionalParams

For details about this form, refer to the Spectre Circuit Simulator Reference.

Tip

You can also type `spectre -h pz` in the shell, for help on *Pole Zero* options.

8. Click *Apply*.

To print the results post simulation, choose *Results – Print – Pole Zero Summary*

You can also plot results from *Results – Direct Plot – Main Form.*

To plot and print these results from OCEAN, use the OCEAN command, `pzPlot` and `pzSummary`.

Other Spectre Analyses

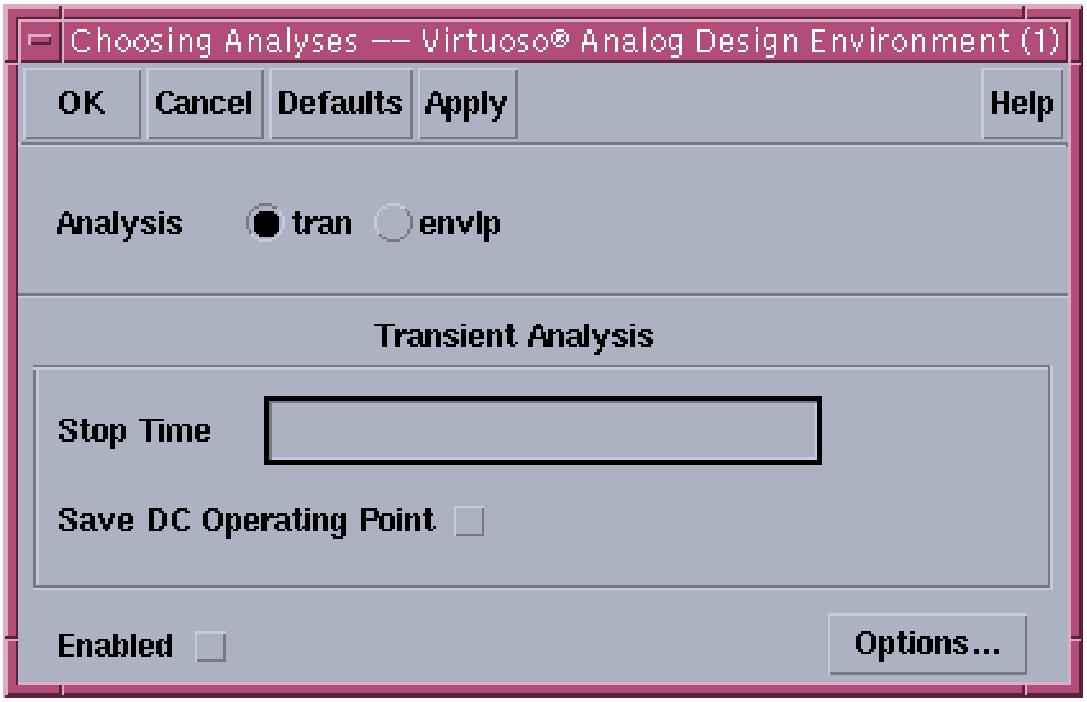
For information on the Spectre analyses available, see the *Spectre Circuit Simulator Reference* manual.

Setting Up an UltraSim Analysis

To set up the Virtuoso® UltraSim™ simulator for analysis,

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.


Choosing Analyses –– Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help
Transient Analysis
Options...

Analysis ● tran ○ envlp

Stop Time

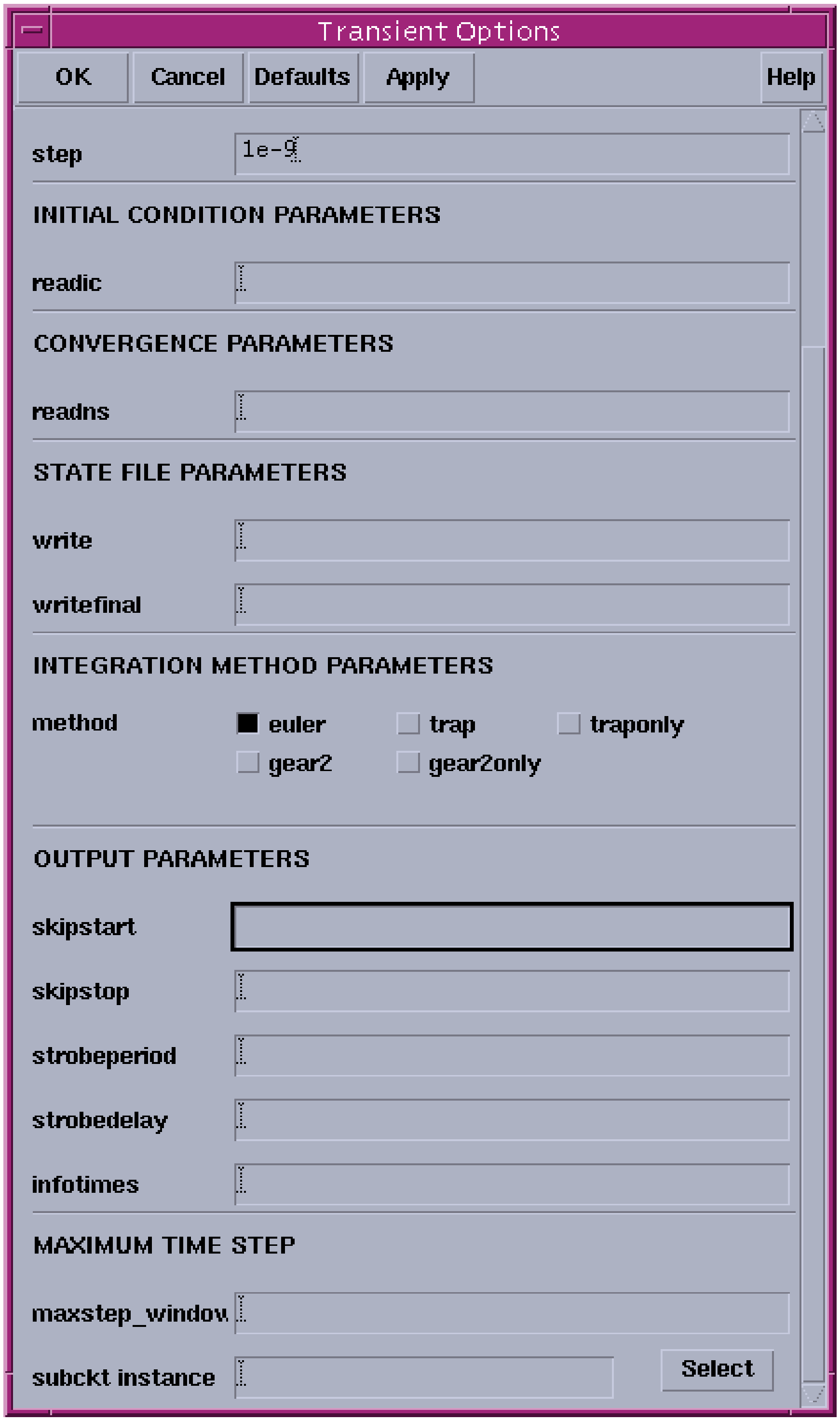
Save DC Operating Point □

Enabled □

The *tran* option is selected.

2. Click *Options*.

The Transient Options form appears.



step 1e-9

INITIAL CONDITION PARAMETERS

readic

CONVERGENCE PARAMETERS

readns

STATE FILE PARAMETERS

write

writefinal

INTEGRATION METHOD PARAMETERS

method ■ euler □ trap □ traponly □ gear2 □ gear2only

OUTPUT PARAMETERS

skipstart

skipstop

strobeperiod

strobedelay

infotimes

MAXIMUM TIME STEP

maxstep_windov

subckt instance

3. Set the transient simulation options as needed.

- ❑ **start=0 s** transient start time.
- ❑ **outputstart** output is saved after this time is reached.
- ❑ **step** minimum time step used by the simulator to maintain the aesthetics of the computed waveforms.
- ❑ **readic** initial condition is contained in this file (`readic` file is specified relative to the `/netlist` directory).
- ❑ **readns** estimate of the DC solution (nodeset) is contained in this file.
- ❑ **write** initial transient solution is written to this file.
- ❑ **writefinal** final transient solution is written to this file.
- ❑ **method** integration method. Values are `euler` (default), `trap`, `traponly`, `gear2` or `gear2only`.
- ❑ **skipstart=starttime s** time to start skipping output data.
- ❑ **skipstop=stoptime s** time to stop skipping output data.
- ❑ **strobeperiod (s)** output strobe interval (in seconds of transient time).
- ❑ **strobedelay=0 s** delay (phase shift) between the skipstart time and the first strobe point.
- ❑ **infotimes=[...] s** times when info analysis specified by `infoname` is performed.
- ❑ **maxstep_window** maximum time step (setting `maxstep` to a smaller value can improve simulation accuracy). You can set global or local maxstep option for one instance. However, if you want to add maxstep options for different instances or subckts, you can add a column in maxstep_window and choose string and input in HED. For example, `time1 value1 time2 value 2....`
- ❑ **subcircuits** circuit blocks for maximum time step.

By default, the output format of the output of an UltraSim transient analysis is SST2.

For more information about the transient simulation options, refer to Chapter 2, “Netlist Formats” (*Virtuoso® UltraSim Simulator User Guide)*.

4. Click *OK*.

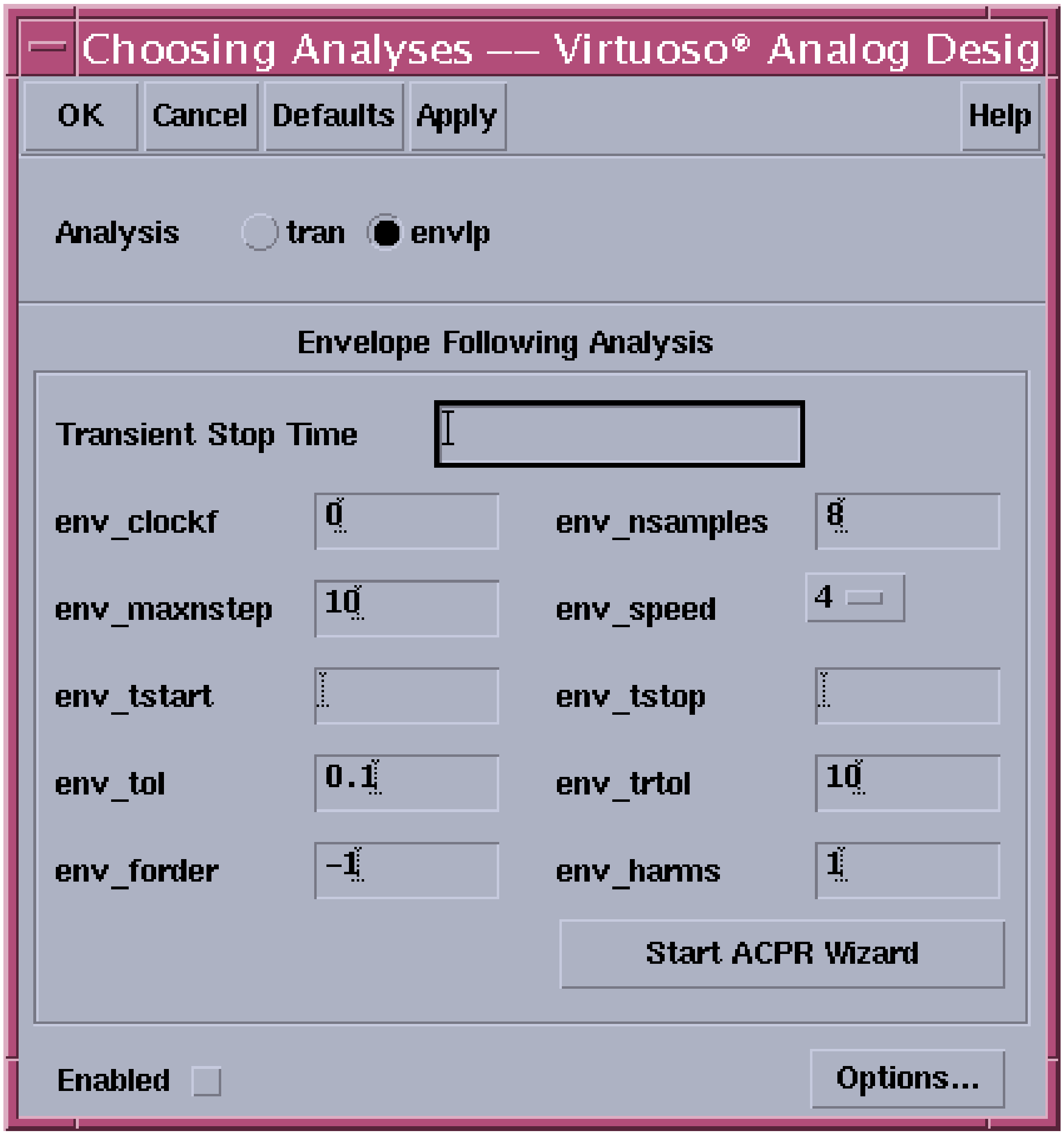
Fast Envelope Analysis for RF Circuit Simulation

Fast envelope simulation is an RF circuit simulation technique developed to reduce the large number of time steps and high computational costs associated with conventional transient analysis. It can be used for specific classes of circuits that operate with multiple, proportionate fundamentals. In this case, the greatest common denominator of all fundamental frequencies should be used as the clock frequency.

To run a fast envelope analysis,

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.



Analysis ○ tran ● envlp

Envelope Following Analysis

Transient Stop Time			
env_clockf	0	env_nsamples	8
env_maxnstep	10	env_speed	4
env_tstart		env_tstop	
env_tol	0.1	env_trtol	10
env_forder	-1	env_harms	1

Enabled □

2. Click the *envlp* button.
3. Choose the appropriate form settings.

❑ **env_clockf** specifies the carrier (clock) for fast envelope analysis.

❑ **env_nsamples** specifies the number of sample (Fourier collocation) points in one carrier (clock) period for fast envelope analysis. The default is eight sample points. The value for *env_nsamples* must be an even integer number greater than or equal to 4.

❑ **env_maxnstep** specifies the maximum number of steps for envelope solve (one point per clock period). The default value is 10 steps. The greater the value of *env_maxnstep*, the higher the speed of the fast envelope analysis over transient analysis, and the lower the accuracy.

❑ **env_speed** specifies the speed setting for fast envelope analysis (settings range from 1-4; default value is 4). A value closer to 1 increases accuracy, but decreases speed with respect to transient analysis.

The *env_speed* settings are as follows:

env_speed = 1 (env_tol = 0.0001; env_trtol = 40)
env_speed = 2 (env_tol = 0.001; env_trtol = 30)
env_speed = 3 (env_tol = 0.01; env_trtol = 20)
env_speed = 4 (env_tol = 0.1; env_trtol = 10)

❑ **env_tstart** specifies the start time for fast envelope analysis (default is three clock periods). For circuits with waveforms that have significant transient changes at the beginning of the analysis, set *env_tstart* to a time point after the transient changes have subsided.

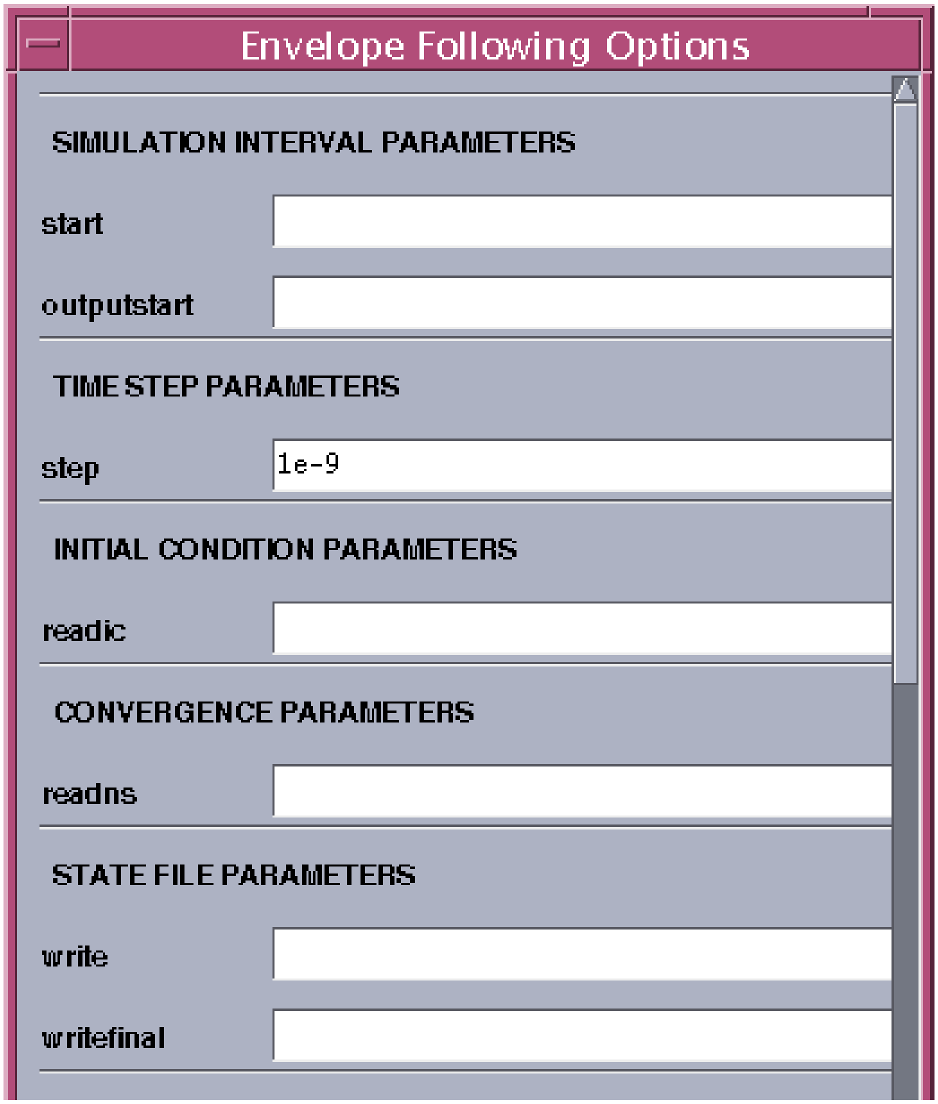
❑ **env_tstop** specifies the stop time for fast envelope analysis. *env_tstop* is useful for fast envelope analysis of a specific time interval and subsequent transient analysis of the remaining simulation time.

❑ **env_tol** controls the accuracy of envelope solve for fast envelope analysis. The range of values for *env_tol* is between 0 and 1. A value closer to 0 increases accuracy, but decreases speed with respect to transient analysis. Its default value is set according to *env_speed*.

❑ **env_trtol** multiplies *env_tol* for local truncation error (LTE) checking of envelope solve in fast envelope analysis. The value of *env_trtol* must be greater than or equal to 1. A value closer to 1 increases accuracy, but decreases speed with respect to transient analysis. Its default value is set according to *env_speed*.

❑ **env_forder** specifies the filtering order for spectral filtering in fast envelope analysis. A value greater than or equal to 1 can be selected for *env_forder*. The lower the value selected, the greater the amount of filtering applied.

- ❑ **env_harms** specifies the number of clock frequency harmonics used to calculate the time varying Fourier coefficients (default is `env_harms=1`). Due to sampling criteria, *env_harms* must be less than or equal to `env_nsamples/2`. If a larger number is specified, *env_harms* is reset to `env_nsamples/2`.
- ❑ **Start ACPR Wizard** opens the ACPR Wizard. Use the wizard to help you fill in the Choosing Analyses form. For more information, refer to "Using the ACPR Wizard" on page 237.
- ❑ **Enabled** runs the analysis with the next simulation.
- ❑ **Options** opens the Envelope Following Options form.



Envelope Following Options

SIMULATION INTERVAL PARAMETERS

start	
outputstart	

TIME STEP PARAMETERS

step	1e-9

INITIAL CONDITION PARAMETERS

readic	

CONVERGENCE PARAMETERS

readns	

STATE FILE PARAMETERS

write	
writefinal	

The fast envelope analysis options are the same as the transient simulation options (refer to the transient simulation options section for more information).

4. In the Simulation window, choose *Simulation – Netlist and Run*.

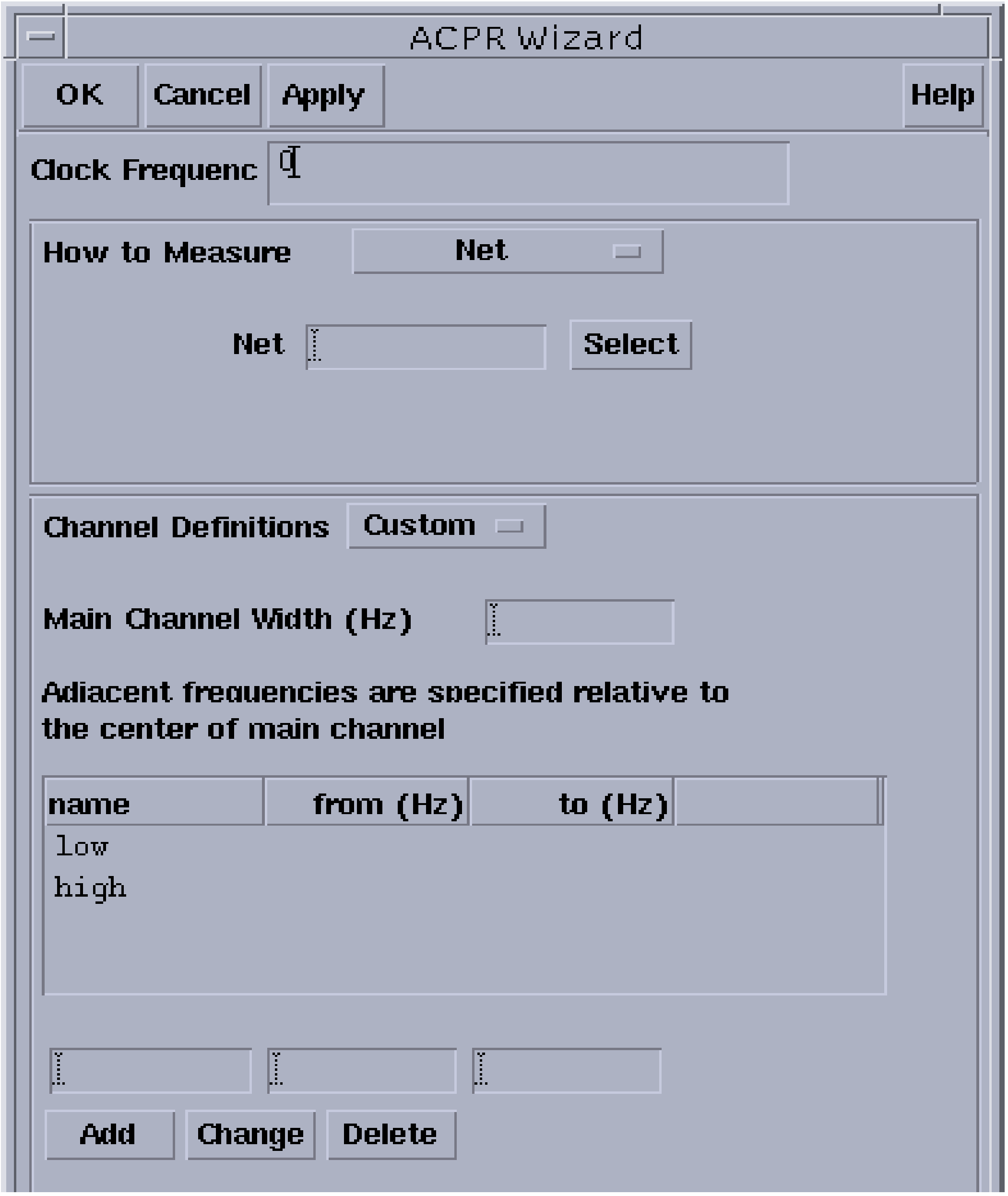
   Check the CIW for messages stating that the simulation has started and finished successfully (information is also written to a log file as the simulation runs).

Using the ACPR Wizard

The adjacent channel power ratio (ACPR) wizard simplifies the complicated calculations needed to measure ACPR. It determines the appropriate simulation parameters and function arguments from the information you supply to the ACPR wizard.

1. In the Choosing Analyses form, click *Start ACPR Wizard*.

The ACPR Wizard form appears.



How to Measure: Net

Channel Definitions: Custom

Adjacent frequencies are specified relative to the center of main channel

name	from (Hz)	to (Hz)	
low			
high			

2. Choose the appropriate form settings.

- ❑ **Clock Frequency** identifies the source of the modulated signal.
- ❑ **How to Measure** specifies the measurement for a single net or between differential nets.
- ❑ **Channel Definitions** specifies frequencies for the channel.

- ❑ **Add**, **Change**, and **Delete** modifies channel definitions.
- ❑ **Stabilization Time (Sec)** specifies the length of time in seconds to wait before using the data for analysis.
- ❑ **Repetitions** designates the number of times to repeat the discrete Fourier transform for averaging.
- ❑ **Resolution Bandwidth (Hz)** specifies the spacing of data points on the resulting power density curve, in Hz.
- ❑ **Windowing Function** tapers the signal before performing the discrete Fourier transform (DFT) to reduce the effect of any edge discontinuities.

3. Click *OK*.

   Values are calculated and appear in the Choosing Analyses form.

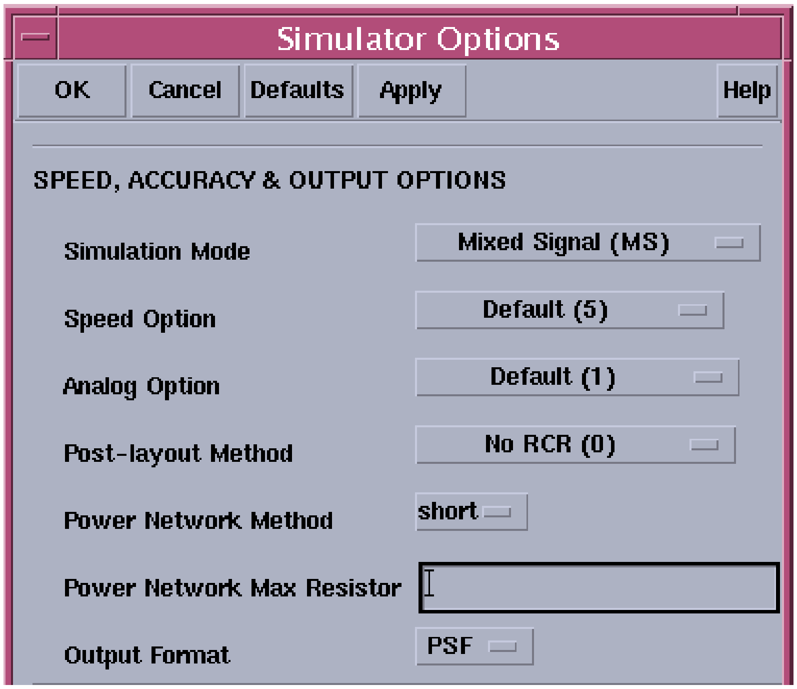
4. Run the simulation.

Running Advanced Analysis UltraSim Simulations

To run a Virtuoso UltraSim simulator advanced analysis,

1. In the Simulation window, choose *Simulation – Options – Analog*.

The Simulator Options form appears.



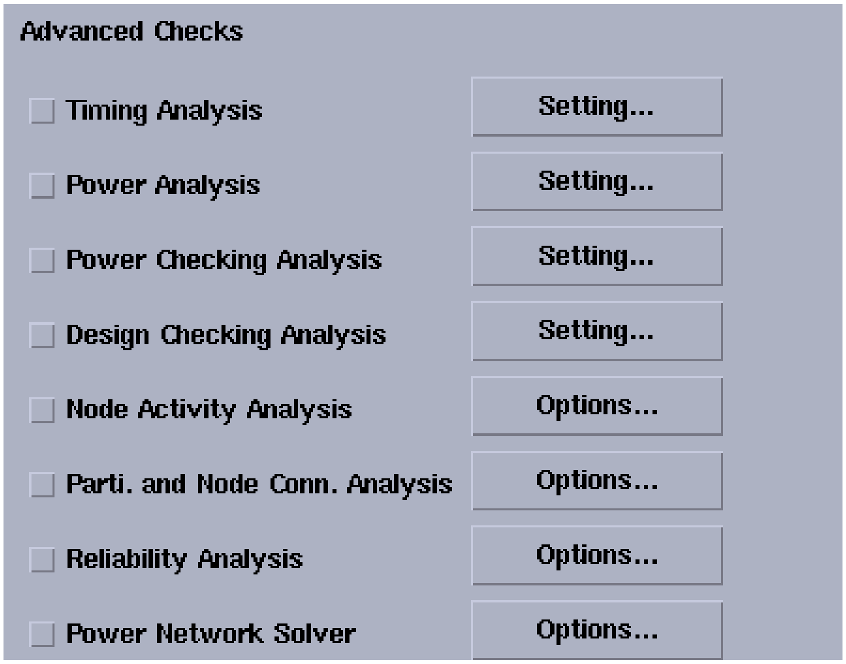
SPEED, ACCURACY & OUTPUT OPTIONS

Simulation Mode	Mixed Signal (MS)
Speed Option	Default (5)
Analog Option	Default (1)
Post-layout Method	No RCR (0)
Power Network Method	short
Power Network Max Resistor	
Output Format	PSF

There are eight advanced analysis options in the *Advanced Checks* section:

- Timing Analysis on page 241
- Power Analysis on page 246
- Power Checking Analysis on page 246
- Design Checking Analysis on page 247
- Node Activity Analysis on page 248
- Parti. and Node Conn. Analysis on page 249
- Reliability Analysis on page 250

- ❑ Power Network Solver on page 251


Advanced Checks

□ Timing Analysis

□ Power Analysis

□ Power Checking Analysis

□ Design Checking Analysis

□ Node Activity Analysis

□ Parti. and Node Conn. Analysis

□ Reliability Analysis

□ Power Network Solver

2. Choose the appropriate advanced analysis and set the options in the corresponding analysis forms.

Timing Analysis

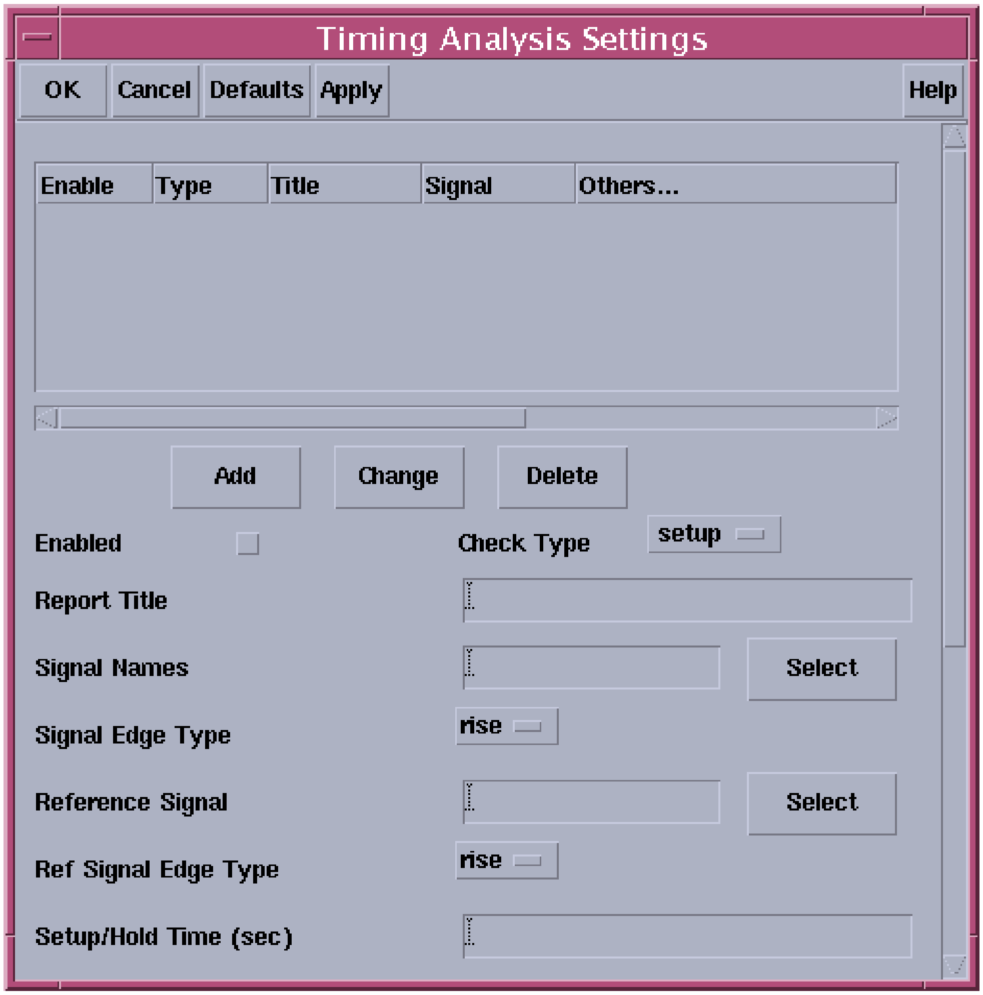
In timing analysis, you can perform the following checks on signals:

- Setup Check on page 242
- Hold Check on page 243
- Pulse Width Check on page 244
- Timing Edge Check on page 245

For more information about the timing analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso® UltraSim Simulator User Guide)*.

To view a timing analysis, choose *Results – Print – Advanced Analysis Results – Timing Analysis* in the Simulation window.

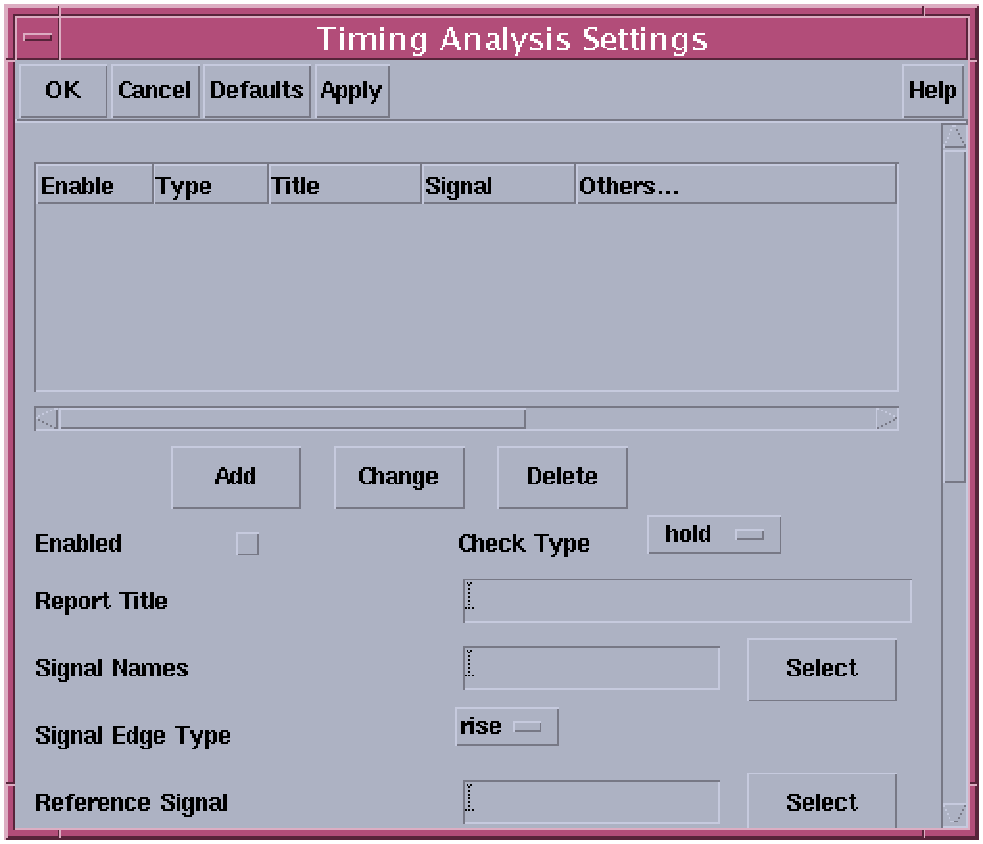
Setup Check



Enabled □
Check Type: setup
Report Title
Signal Names
Signal Edge Type: rise
Reference Signal
Ref Signal Edge Type: rise
Setup/Hold Time (sec)

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add a setup check.

Hold Check



Enabled □

Check Type hold

Report Title

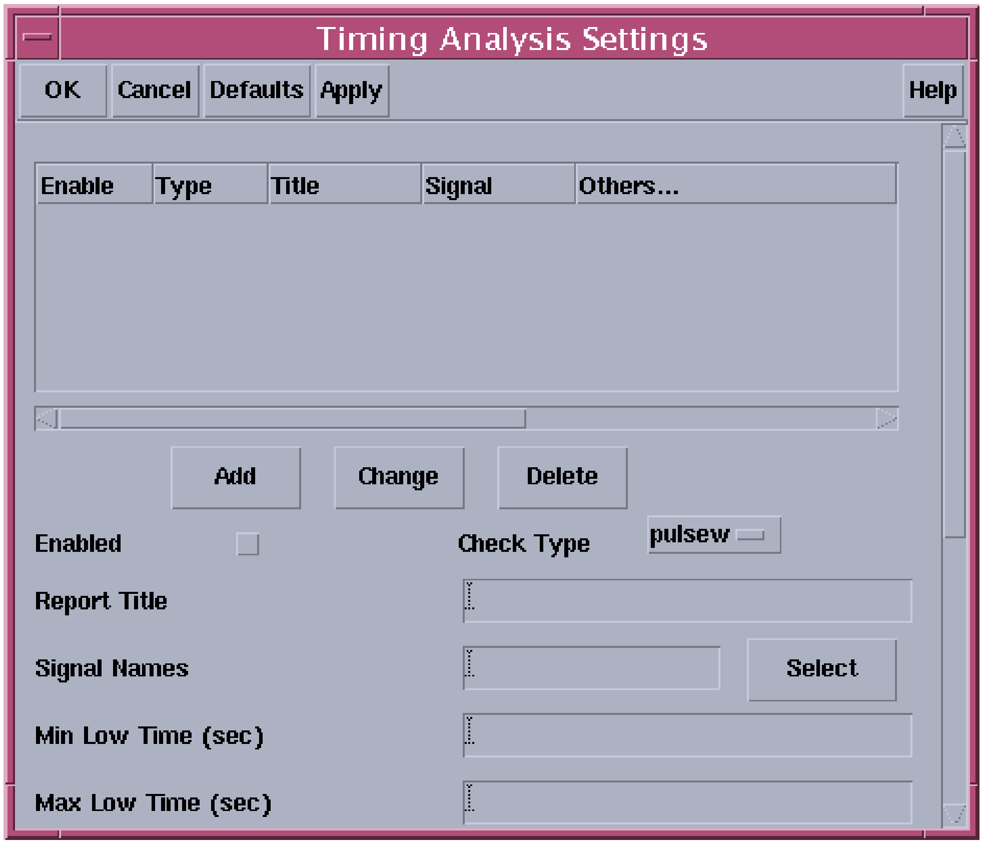
Signal Names

Signal Edge Type rise

Reference Signal

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add a hold check.

Pulse Width Check

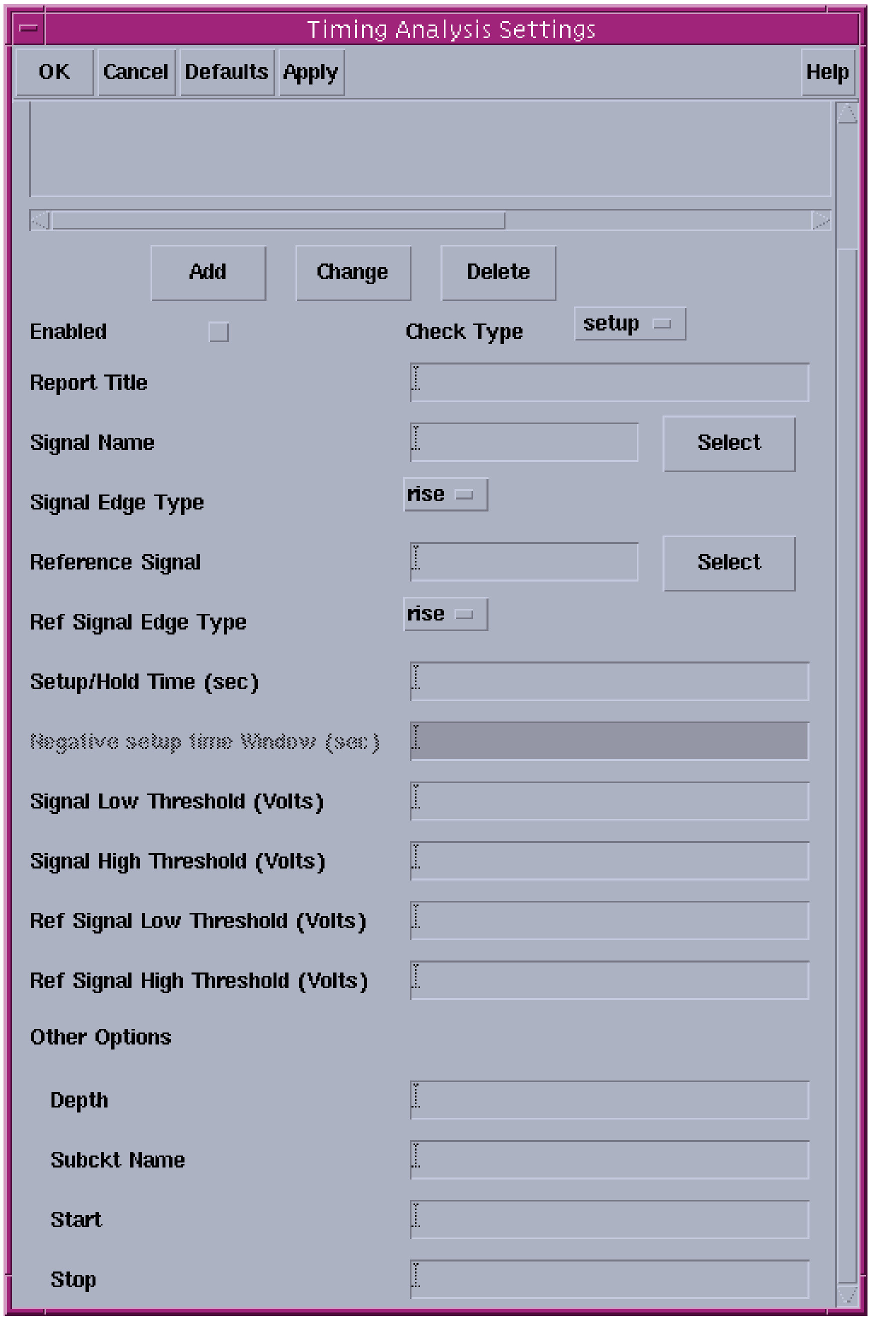


Timing Analysis Settings

Enabled □
Check Type pulsew
Report Title
Signal Names
Min Low Time (sec)
Max Low Time (sec)

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add a pulse width check.

Timing Edge Check



Enabled □ Check Type setup

Report Title

Signal Name

Signal Edge Type rise

Reference Signal

Ref Signal Edge Type rise

Setup/Hold Time (sec)

Negative setup time Window (sec)

Signal Low Threshold (Volts)

Signal High Threshold (Volts)

Ref Signal Low Threshold (Volts)

Ref Signal High Threshold (Volts)

Other Options

Depth

Subckt Name

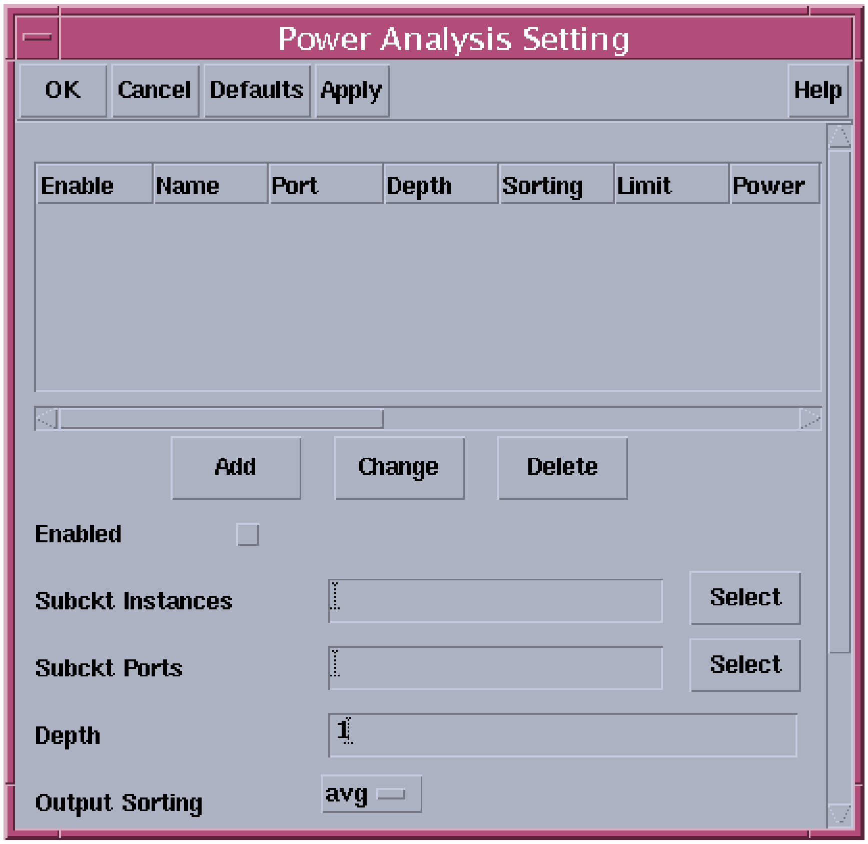
Start

Stop

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add an timing edge check.

Power Analysis

The power analysis reports the power consumed by each element and subcircuit in the circuit. Power analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Power Analysis*.



Enable	Name	Port	Depth	Sorting	Limit	Power

Enabled □

Subckt Instances

Subckt Ports

Depth 1

Output Sorting avg

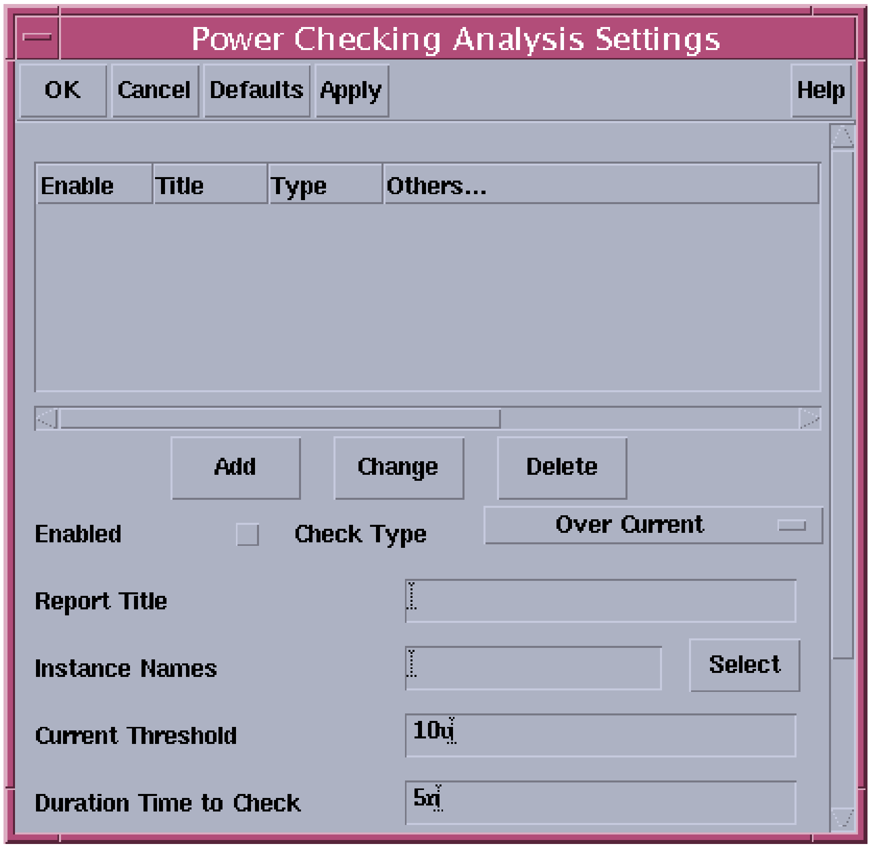
1. Adjust the power analysis settings as needed.
2. Click *Add* to add a power analysis check.

For more information about the power analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso® UltraSim Simulator User Guide)*.

Power Checking Analysis

Based on the specified element list (current threshold, over current duration time, and checking windows), the Virtuoso UltraSim simulator reports in a `.pcheck` file which elements over what time period have current over the threshold for a time period equal to or greater than the specified duration. If no window is specified, the whole simulation period is used.

Power checking results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Power Check Report*.



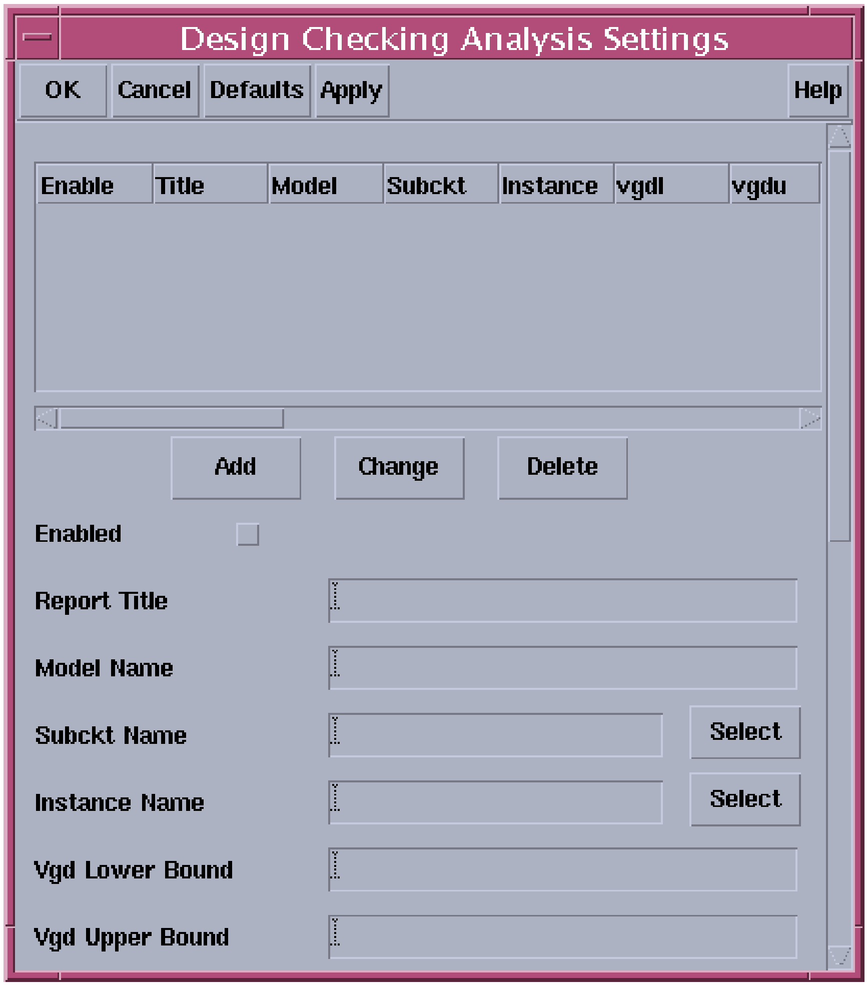
Enabled □
Check Type: Over Current
Report Title:
Instance Names:
Current Threshold: 10u
Duration Time to Check: 5n

1. Adjust the power checking analysis settings as needed.
2. Click *Add* to add a power check.

For more information about the power checking analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso UltraSim Simulator User Guide*).

Design Checking Analysis

This command allows you to monitor device voltages during a simulation run, and generates a report if the voltages exceed the specified upper and lower bounds. Design checking results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Device Voltage Check Report*.



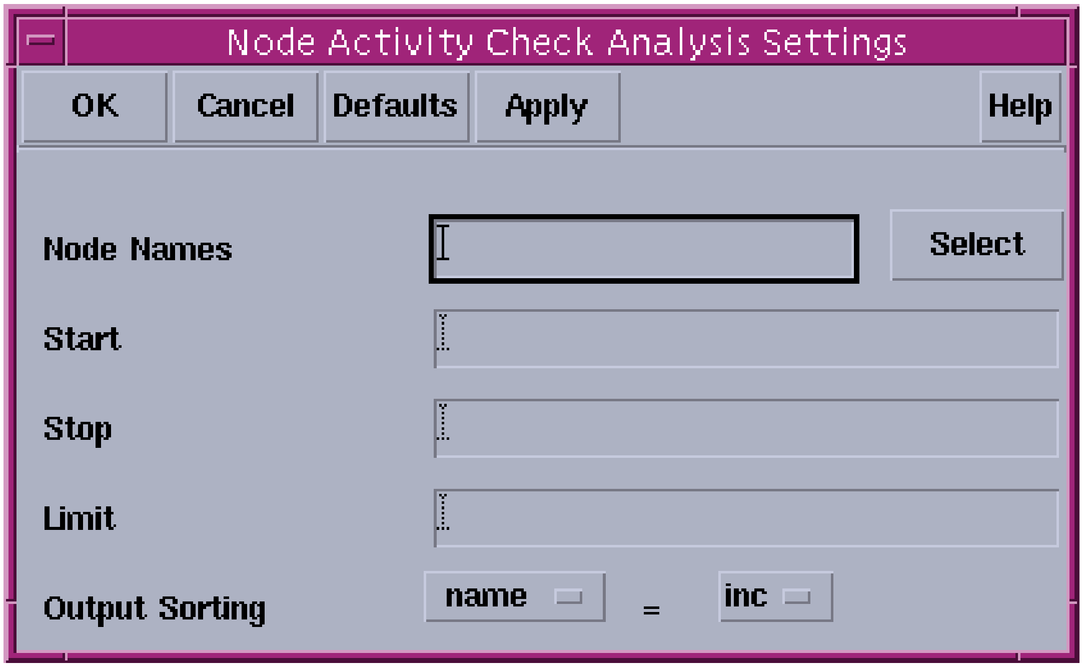
1. Adjust the design checking analysis settings as needed.
2. Click *Add* to add a design check.

For more information about the design checking analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso UltraSim Simulator User Guide*).

Node Activity Analysis

The node activity analysis provides information about the nodes and monitors activities such as voltage overshoots (VOs) and voltage undershoots (VUs), maximum and minimum rise/fall times, signal probability of being high or low, node capacitance, and number of toggles.

Node activity analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Node Activity Analysis*.



1. Enter the Node Name or select the same from the schematic.
2. Enter the Start and Stop time.
3. Adjust the node activity analysis settings as needed.
4. Click *OK* to add a node activity analysis check.

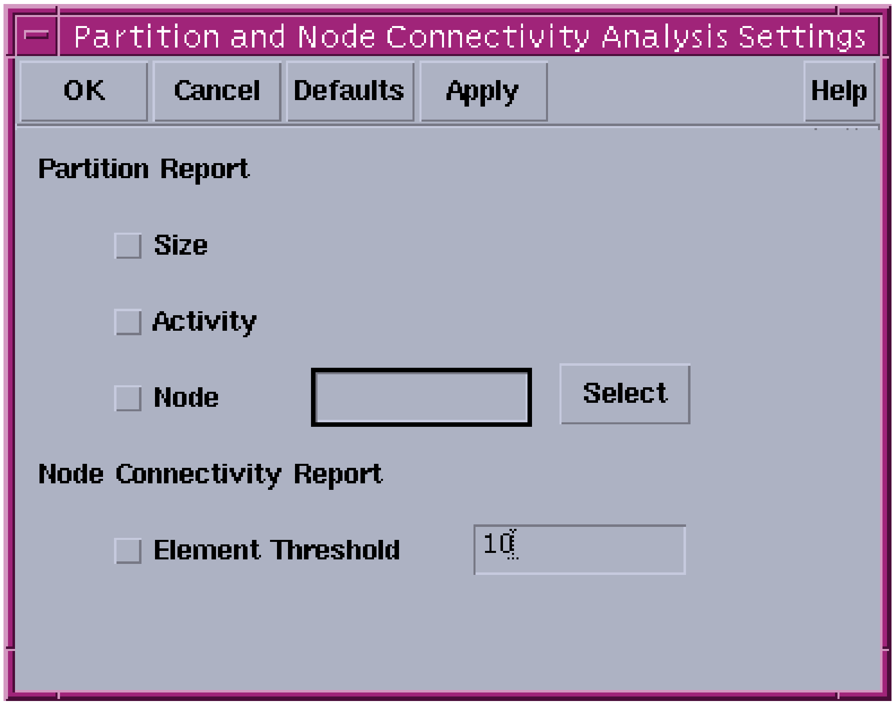
For more information about the node activity analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso® UltraSim Simulator User Guide)*.

Parti. and Node Conn. Analysis

The Virtuoso UltraSim simulator lets you perform partition and node connectivity analysis using .usim_report commands. The information is reported in a .pr file. For example, if the netlist name is circuit.sp, then the report is named circuit.pr.

The .usim_report commands are useful for debugging simulations. For example, checking the size of partitions and their activities, as well as checking node activity to verify bus nodes.

Partition and Node Connectivity Analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Parti. and Node Conn. Analysis*.



Partition Report

□ Size

□ Activity

□ Node

Node Connectivity Report

□ Element Threshold 10

1. Adjust the partition and node connectivity settings as needed.
2. Click *OK* to add a partition and node connectivity analysis check.

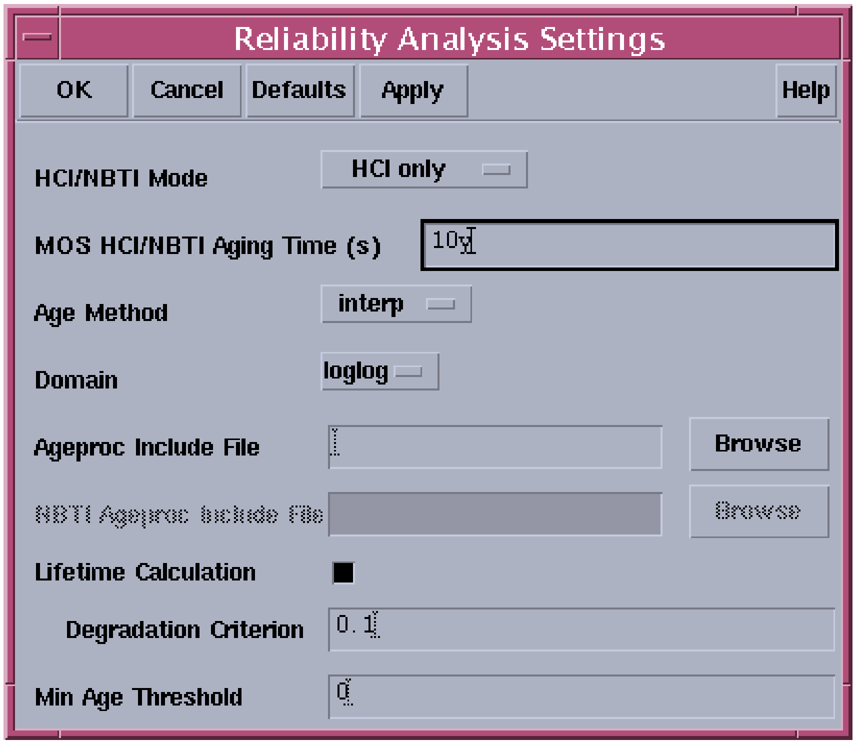
For more information about the partition and node connectivity analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso® UltraSim Simulator User Guide)*.

Reliability Analysis

The reliability analysis simulates circuit aging due to hot carrier injection (HCI) induced degradation and negative bias thermal instability (NBTI). It can also simulate degraded circuit performance for the above effects, after a specified amount of circuit operation.

To run a reliability analysis, you need reliability models which are usually provided by your modeling group. Reliability models can be added using *Setup – Model Libraries* in the Simulation window.

Note: Reliability analysis is only available with HSPICE netlist format. Reliability analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Reliability Analysis*.



HCI/NBTI Mode	HCI only
MOS HCI/NBTI Aging Time (s)	10y
Age Method	interp
Domain	loglog
Ageproc Include File	
NBTI Ageproc Include File	
Lifetime Calculation	■
Degradation Criterion	0.1
Min Age Threshold	0

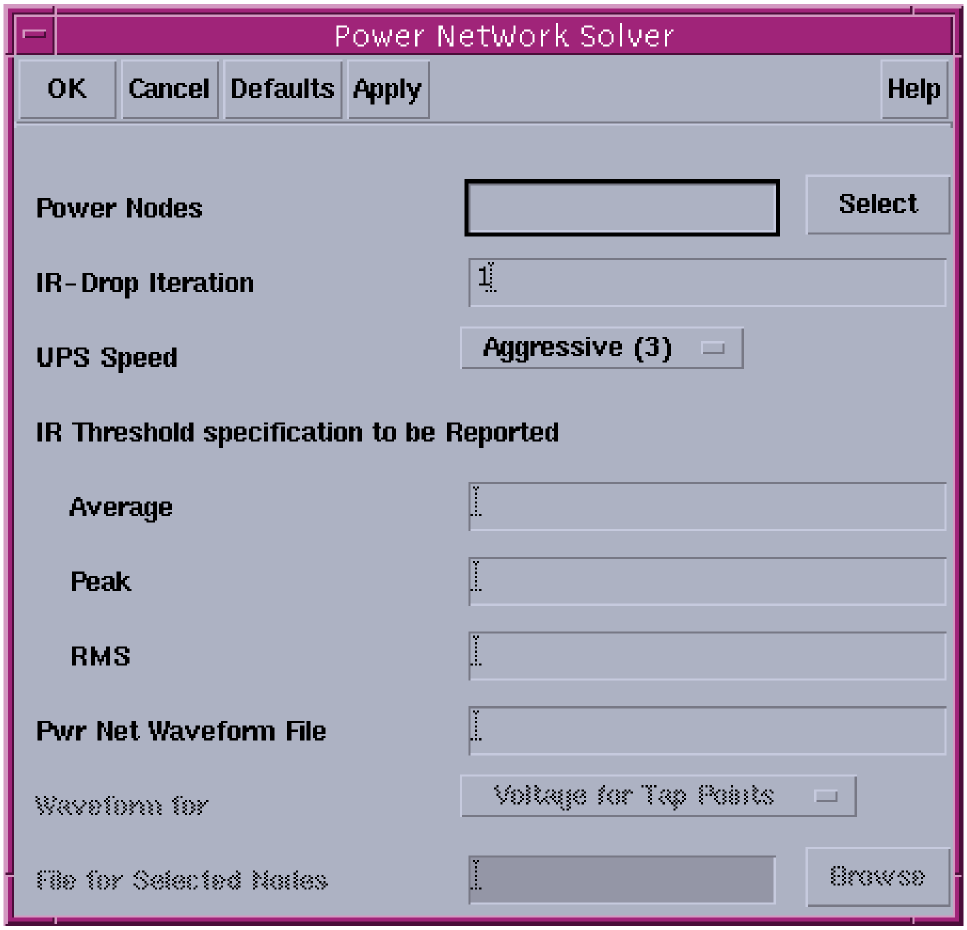
1. Adjust the reliability activity analysis settings as needed.
2. Click *OK* to add a reliability analysis check.

For more information about the reliability activity analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso® UltraSim Simulator User Guide)*.

Power Network Solver

The Power network solver is an optimized solver designed to analyze linear power networks. The solver is integerated into Virtuoso Ultrasim Simulator and together with the Virtuoso Ultrasim engine, lets you calculate the IR drop in power networks and analyze the effects of IR drop on circuit behaviour.

Power Network solver can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – IR Drop Report*.



Power Nodes

IR-Drop Iteration 1

UPS Speed Aggressive (3)

IR Threshold specification to be Reported

Average

Peak

RMS

Pwr Net Waveform File

Waveform for Voltage for Tap Points

File for Selected Nodes

1. Adjust the network power solver settings as needed.
2. Click *OK* to add a this check.

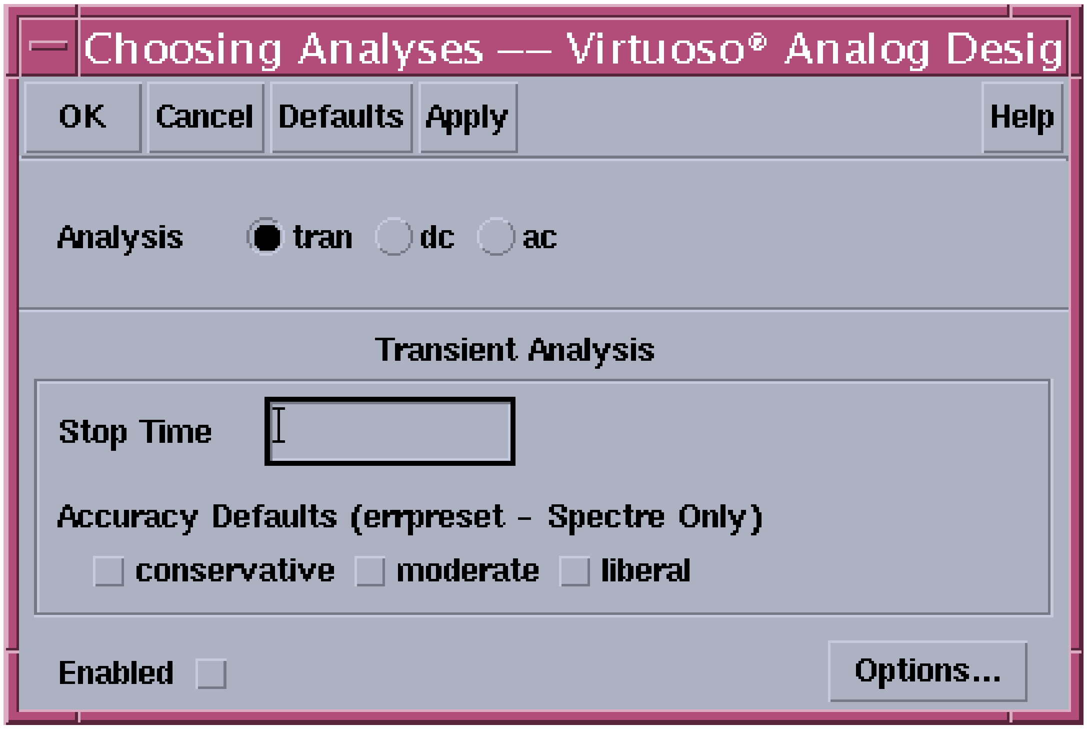
For more information about the Power Network Solver, refer to Chapter 6, “Power Network Solver” (*Virtuoso® UltraSim Simulator User Guide)*.

Setting Up an AMS Analysis

To set up the Virtuoso® AMS simulator for analysis,

1. In the Simulation window, choose *Analyses – Choose*.

   The Choosing Analyses form appears.



Analysis ● tran ○ dc ○ ac

Stop Time

Accuracy Defaults (errpreset - Spectre Only)

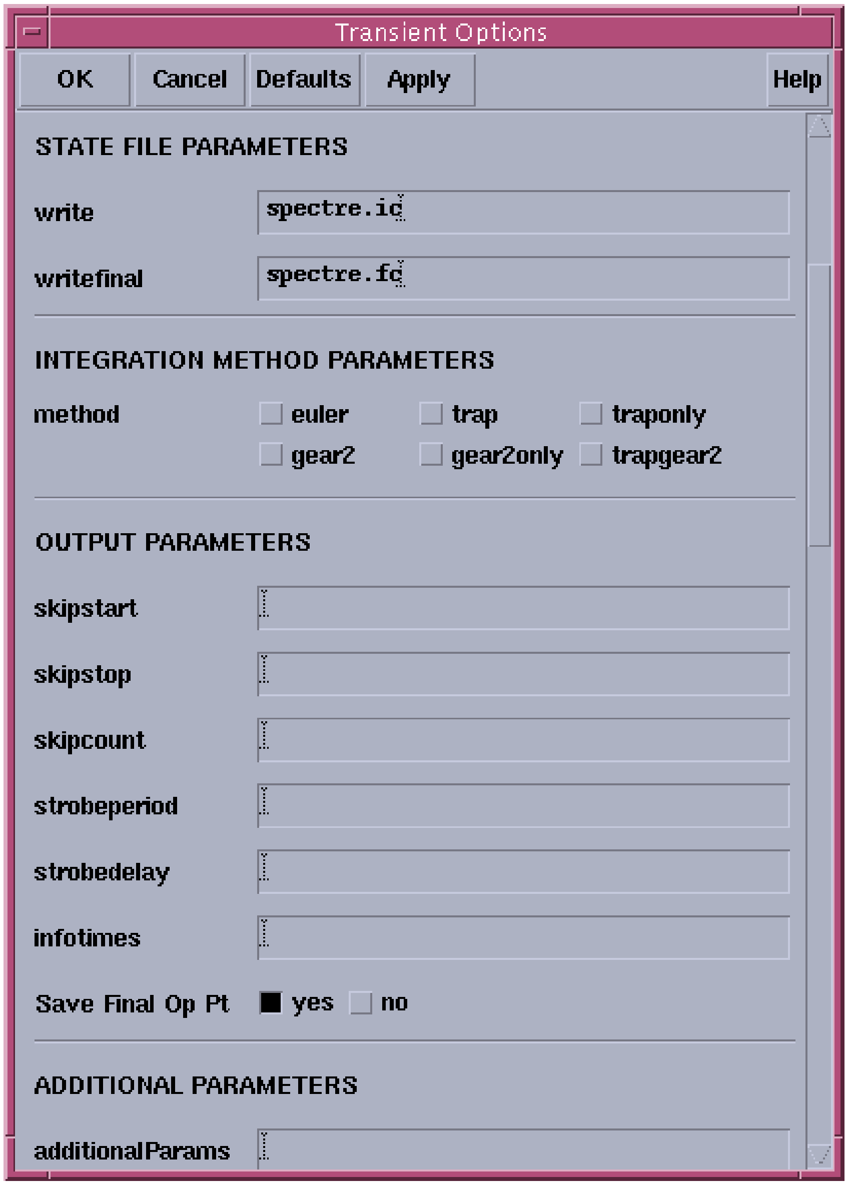
□ conservative □ moderate □ liberal

Enabled □

The Virtuoso® AMS simulator supports transient and dc analyses. It supports ac analysis only when spectre is selected as the solver. In this example, the *tran* option is selected. This computes the transient response of the circuit over a specified time interval. You can adjust transient analysis parameters in several ways to meet the needs of your simulation. You can influence the speed of the simulation by setting parameters that control accuracy requirements and the number of data points saved.

2. Click *Options*. The *Transient Options* form appears. In addition to common options, it contains a section *ULTRASIM TRANSIENT PARAMETERS* with options applicable

only to the UltraSim solver and a section *SPECTRE TRANSIENT PARAMETERS* with options applicable only to the Spectre solver.



STATE FILE PARAMETERS

write: spectre.ic

writefinal: spectre.fc

INTEGRATION METHOD PARAMETERS

method: □ euler □ trap □ traponly □ gear2 □ gear2only □ trapgear2

OUTPUT PARAMETERS

skipstart:

skipstop:

skipcount:

strobeperiod:

strobedelay:

infotimes:

Save Final Op Pt ■ yes □ no

ADDITIONAL PARAMETERS

additionalParams:

3. Set the transient simulation options as needed.

Simulation Interval Parameters

- **start=0 s** transient start time.

Time Step Parameters

- ❑ **step=1e-9 s** minimum time step used by the simulator to maintain the aesthetics of the computed waveforms.

Initial Condition Parameters

- ❑ **readic** initial condition is contained in this file (`readic` file is specified relative to the `/netlist` directory).

State File Parameters

- ❑ **write** initial transient solution is written to this file.
- ❑ **writefinal** final transient solution is written to this file.

Integration Method Parameters

- ❑ **method** integration method. Values are `euler` (default), `trap`, `traponly`, `gear2` `gear2only`, `trapgear2`

Output Parameters

- ❑ **skipstart=starttime s** time to start skipping output data.
- ❑ **skipstop=stoptime s** time to stop skipping output data.
- ❑ **strobeperiod (s)** output strobe interval (in seconds of transient time).
- ❑ **strobedelay=0 s** delay (phase shift) between the skipstart time and the first strobe point.
- ❑ **infotimes=[...] s** times when info analysis specified by `infoname` is performed.
- ❑ **Save Final Op Pt.** generates the info statements for final operating point into the control file and related data into in psf file. If you do not want the results to be saved, select *No*. When this option is disabled, no final operating point data is generated.

Time Step Parameters

- ❑ **maxstep** largest time step permitted

Initial Condition Parameters

- ❑ **ic** Values are `dc, node, dev, all`
- ❑ **skipdc** Values are `no, yes, waveless, rampup, autodc`

Convergence Parameters

- ❑ **cmin** minimum capacitance to ground at each node set to a physically reasonable nonzero value

Accuracy Parameters

- ❑ Set multiple accuracy-speed trade-off parameters (`relref, lteratio, fastbreak`) at once

Newton Parameters

- ❑ **maxiters** maximum iterations

Annotation Parameters

- ❑ Specify the degree of annotation. Values are no, `title`, `sweep`, `status`, steps. Print a summary of which runs did not converge or had problems evaluating statements.

Additional Parameters

- ❑ Specify any additional analysis parameters that you want to be used while netlisting.

Setting Up a SpectreS Analysis

To set up analyses for the spectreS interface,

1. Choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. Select an analysis.

   The Choosing Analyses form redisplays to show the parameters for the new analysis.

3. Set the options and click *Apply*.

4. Select another analysis to set up.

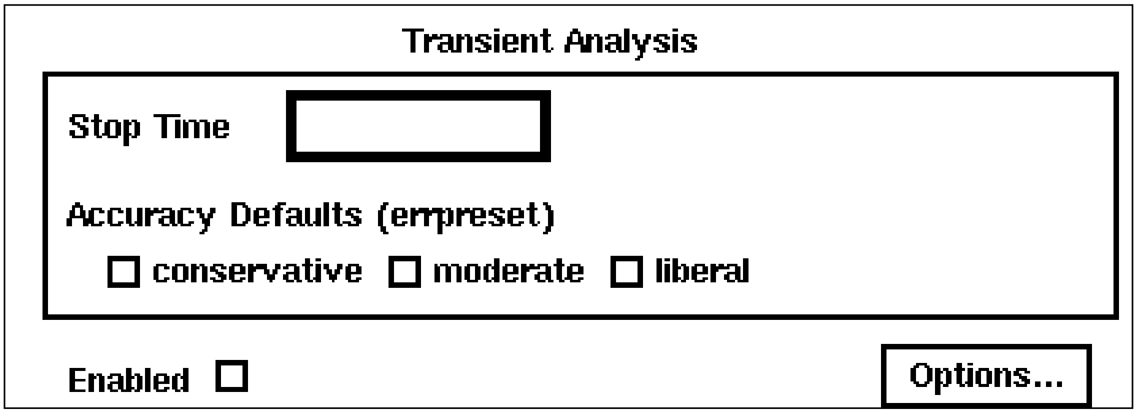
The next step is usually selecting the outputs you want to save.

For help setting up a particular analysis, click an analysis at the left or refer to the *Spectre Circuit Simulator Reference*.

Transient Analysis

The transient analysis computes the transient response of a circuit over an interval. The initial condition is taken to be the DC steady-state solution.

To set up a transient analysis,



Transient Analysis

Stop Time

Accuracy Defaults (errpreset)

□ conservative □ moderate □ liberal

Enabled □

1. In the Choosing Analyses form, type the *Stop Time*.
2. Click *Options* to
   - ❑ Set Spectre options controlling the time step
   - ❑ Set any other options related to transient simulation
3. Click *Apply*.

AC Small-Signal or S-Parameter Analysis

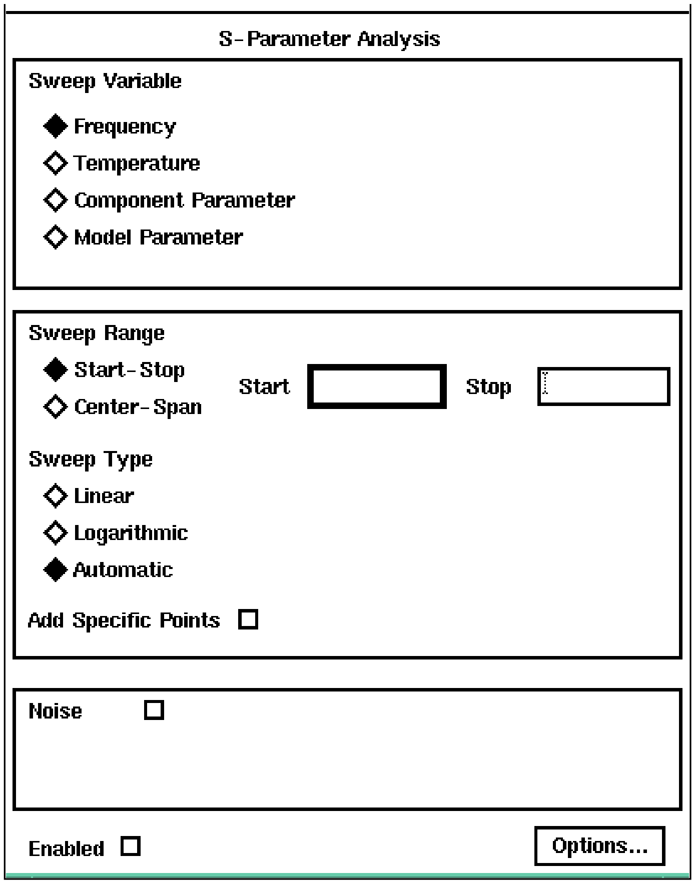
AC small-signal analysis linearizes the circuit about the DC operating point and computes the response to a given small sinusoidal stimulus. SpectreS can perform the analysis while sweeping a parameter.

The parameter can be a frequency, a temperature, a component instance parameter, or a component model parameter. If changing a parameter affects the DC operating point, the operating point is recomputed on each step.

The S-parameter analysis computes the S-parameters between one or more ports in a circuit. These S-parameters describe a linear N-port.

To set up an AC small-signal or S-parameter analysis,

1. Choose *ac* or *sp* from the Choosing Analyses form to display the appropriate options.



S-Parameter Analysis

Sweep Variable

◆ Frequency
◇ Temperature
◇ Component Parameter
◇ Model Parameter

Sweep Range

◆ Start-Stop
◇ Center-Span

Start □ Stop □

Sweep Type

◇ Linear
◇ Logarithmic
◆ Automatic

Add Specific Points □

Noise □

Enabled □

2. Select a sweep variable option and specify any necessary parameters.

   ❑ If you do not sweep the frequency, specify the frequency at which to sweep the variable.

   ❑ If you sweep a component, specify the parameter to sweep. Click *Select Component* to click in the Schematic window and select the component.

   ❑ If you sweep a model parameter, enter the model and parameter names.

3. Specify the sweep range and type.

   Enter the start and stop points of the range or the center and span of the range.

   The sweep type options are mapped to Spectre statements:

   ❑ Linear + Step Size = step

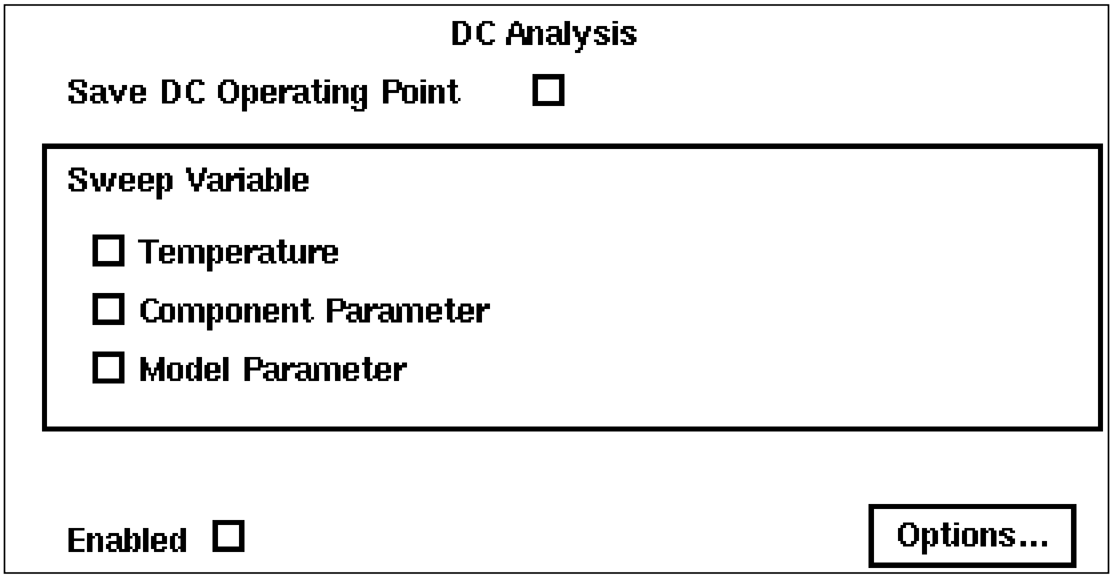
   ❑ Linear + Total Points = lin

- ❑ Logarithmic + Points Per Decade = dec
- ❑ Logarithmic + Total Points = log
- ❑ Add Specific Points = values=[…]

4. Click *Options* to select the spectreS options controlling the simulation.
5. Click the *Noise* radio button to perform noise analysis (for S-parameter analysis only).
6. Click *Enabled* and *Apply*.

DC Analysis

The DC analysis finds the DC operating point or DC transfer curves of the circuit. To generate transfer curves, specify a parameter and a sweep range. The parameter can be a temperature, a device instance parameter, or a device model parameter.



DC Analysis

Save DC Operating Point □

Sweep Variable

□ Temperature
□ Component Parameter
□ Model Parameter

Enabled □

To save the DC operating point,

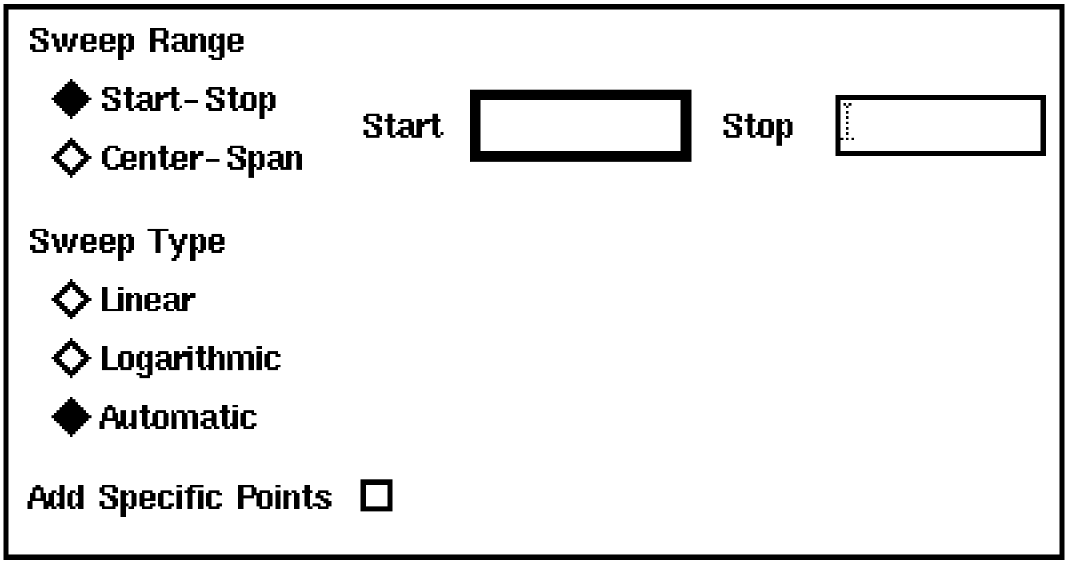
➤ Click *Save DC Operating Point*, click *Enabled*, and click *Apply*.

Sweeping a Variable

To run a DC transfer curve analysis and sweep a variable,

1. Select a sweep variable.

The Choosing Analyses form redisplays to show additional fields.



Sweep Range

◆ Start-Stop

◇ Center-Span

Start □ Stop □

Sweep Type

◇ Linear

◇ Logarithmic

◆ Automatic

Add Specific Points □

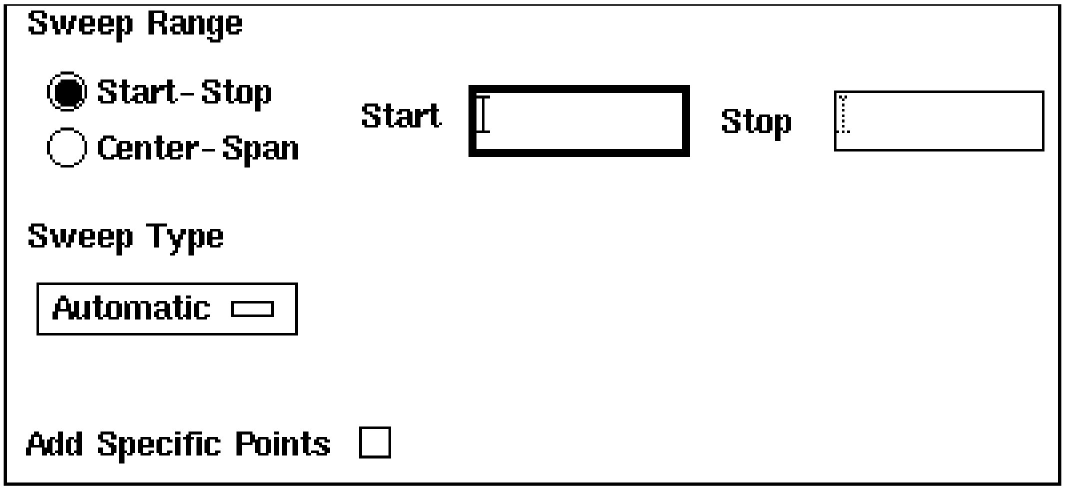
2. Specify the necessary parameters.
   - ❑ To sweep a component, specify the component name and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
   - ❑ To sweep a model parameter, enter the model and parameter names.
3. Specify the sweep range and type.

   The sweep type options are mapped to spectreS statements:
   - ❑ Linear + Step Size = step
   - ❑ Linear + Total Points = lin
   - ❑ Logarithmic + Points Per Decade = dec
   - ❑ Logarithmic + Total Points = log
   - ❑ Add Specific Points = values=[…]
4. Click *Options* to set the spectreS options controlling DC simulation.
5. Click *Apply*.

Transfer Function Analysis

The transfer function, or xf, analysis linearizes the circuit about the DC operating point and performs a small-signal analysis that calculates the transfer function from every independent source or instance terminal in the circuit to a designated output. The variable of interest at the output can be voltage or current.

1. Select a sweep variable option and specify any necessary parameters.
   - If you do not sweep the frequency, specify the frequency at which to sweep the variable.
   - If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
   - If you sweep a model parameter, type the model and parameter names.
2. Specify the sweep range and type.



Sweep Range

● Start-Stop Start Stop

○ Center-Span

Sweep Type

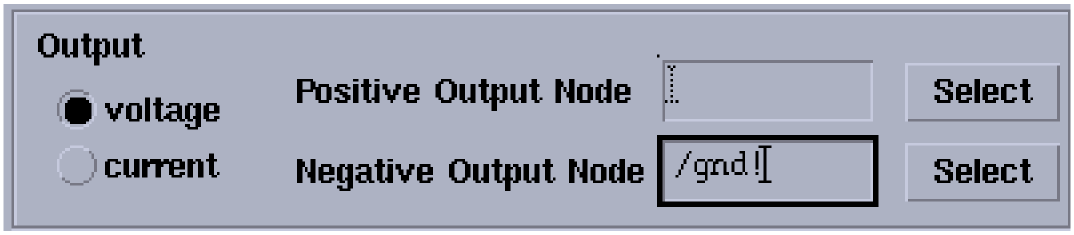
Automatic

Add Specific Points □

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Total Points = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Total Points = log
- Add Specific Points = values=[…]

3. Choose output *voltage* or *current*.



Output
● voltage
○ current
Positive Output Node:
Negative Output Node: /gnd!

- ❑ To measure the output voltage, click *Select* opposite *Positive Output Node* and click a net in the schematic.
- ❑ To measure the output probe, click *probe*, click *Select* opposite *Negative Output Node*, and click an instance in the schematic.

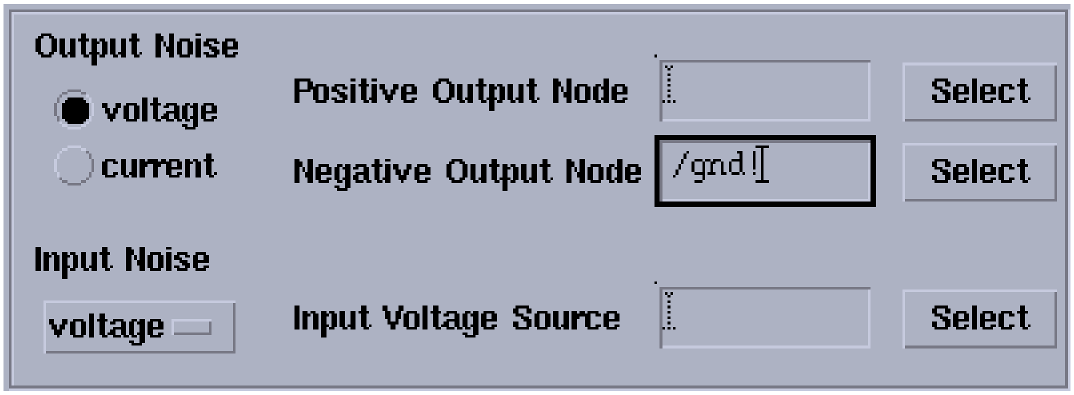
Note: While selecting nodes, select the nodes/nets around the desired instance.

4. Click *Options* to set the spectreS options controlling transfer function simulation.
5. Click *Apply*.

Noise Analysis

The noise analysis linearizes the circuit about the DC operating point and computes the total-noise spectral density at the output. If you specify an input probe, the transfer function and the input-referred noise for an equivalent noise-free network is computed. To set up a noise analysis,

1. Choose a sweep variable option and specify any necessary parameters.



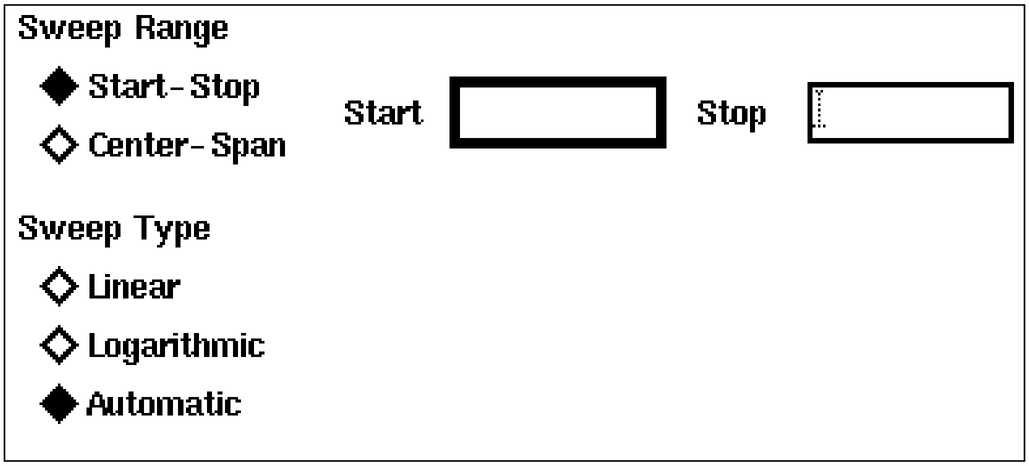
Output Noise
● voltage
○ current
Positive Output Node:
Negative Output Node: /gnd!

Input Noise
voltage
Input Voltage Source:

- ❑ If you do not sweep the frequency, specify the frequency at which to sweep the variable.

- ❑ If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- ❑ If you sweep a model parameter, enter the model and parameter names.

2. Specify the sweep range and type.



Sweep Range

◆ Start-Stop Start [ ] Stop [ ]

◇ Center-Span

Sweep Type

◇ Linear

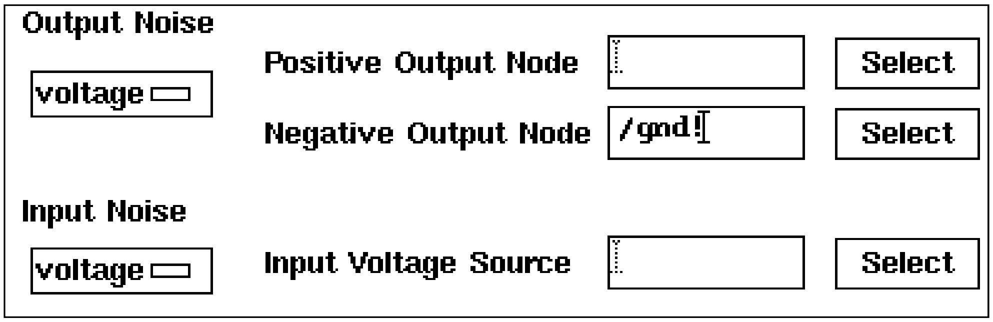
◇ Logarithmic

◆ Automatic

The sweep type options are mapped to Spectre statements:

- ❑ Linear + Step Size = step
- ❑ Linear + Total Points = lin
- ❑ Logarithmic + Points Per Decade = dec
- ❑ Logarithmic + Total Points = log
- ❑ Add Specific Points = values=[…]

3. Choose an *Output Noise* option.



Output Noise

voltage ▭ Positive Output Node [ ] Select

Negative Output Node [/gnd!] Select

Input Noise

voltage ▭ Input Voltage Source [ ] Select

- ❑ To measure the output noise voltage, click *Select* opposite *Positive Output Node* and click a net in the schematic.
- ❑ To measure the output noise probe, click *probe* in the cyclic field and click *Select* opposite *Output Probe Instance*, and click a voltage source in the schematic.

Note: While selecting nodes, select the nodes/nets around the desired instance.

4. Optionally, choose an *Input Noise* option.

- ❑ Choose *voltage*, *current*, or *port*.
- ❑ Click *Select Input Voltage Source* or *Input Current Source* or *Input* Port *Source*.
- ❑ Click a source or port in the schematic.
- ❑ Click *Apply*.

5. Click *Options* to set the spectreS options controlling noise simulation.

6. Click *Apply*.

Current Probing For SpectreS Macro Models

You can enable probing for device terminals in Analog Artist SpectreS when the device is represented by a macro model subcircuit file. A device in a schematic is represented by a SpectreS macromodel subcircuit in a `*.s` file.

You can probe the device’s terminals with the calculator’s IT() command and plot the correct current. The current is a summation of several different devices inside the subcircuit, but the subcircuit is stored outside the Artist environment. The CDF SimInfo section’s termMapping field can be used to create expressions that represent the total current flowing through the subcircuit devices.

Consider that the subcircuit you want to use with the Analog Artist model path looks like this:

```
.subckt &1 in out
    *Subcircuit
     Xcapb in out anyOldName2
  .ends &1
.subckt anyOldName2 in out
  *Capacitor
  c1 in out 5p
.ends anyOldName2
```

If the subcircuit were included with an include file, it would look like this:

```
.subckt anyOldName in out
    *Subcircuit
     Xcapb in out anyOldName2
  .ends anyOldName
.subckt anyOldName2 in out
  *Capacitor
  c1 in out 5p
.ends anyOldName2
```

- The mappedRoot() command creates the fully qualified hierarchical name to the current instance. In other words, if you place down an instance called `cap1` in your schematic, mappedRoot() returns the full path including the c1. Your term mapping would then create the following save command in the final netlist. The netlist would require the two subcircuits, either from using the Model Path or from using an include file. The final netlist would show this line:

  ```
  save  xcap1.xcapb.c1:1
  ```

- You can use the root() command instead of mappedRoot() if you are interested in basing one terminal on another terminal’s current. For example, a two-terminal device could specify the current explicitly for the first terminal, then specify that the second terminal is simply the minus() of the first. You could go one step further and specify that in a three or four terminal device, one of the terminals is the minus() of the sum of all of the other terminals (since D+G+S+B=0, this means B=-(D+G+S)). The termMapping below calculates the PLUS terminal as the current through the plus terminal (terminal 1) of subcircuit device "c1", then calculates the MINUS terminal to be the minus() of whatever was calculated for the whole subcircuit's PLUS terminal:

```
termMapping (nil
    PLUS "(FUNCTION mappedRoot('.xcapb.c1:1'))"
    MINUS "(FUNCTION minus(root('PLUS')))"
  )
```

This will not work if you have current leaving the subcircuit other than through the terminals. For example, if you have body-effect diodes or parasitic capacitors that connect to ground, then the sum of the current entering the subcircuit through the terminals is not truly the total current used by the device.

- In order to use subcircuit mapping in 4.4.1 or later, you need to use the Spectre hierarchy delimiter ".", not the spice delimiter "^". In 4.3.4, the spice delimiter could be used because the "save" command appeared in a section of the final netlist that was in Spectre's Spice compatibility mode. In 4.4, the save statement appears in a Spectre native-mode section, so you must use the Spectre native delimiter. Again, this term mapping would require BOTH subcircuits anyOldName and anyOldName2 from either the .s (model path) file or the include file mentioned above. The first subcircuit (either the &1 or the anyOldName) will instantiate the Xcapb device. The second subcircuit (anyOldName2) will instantiate the c1 device which is inside the subcircuit for the Xcapb instance.

- You must use lower-case in your mapping command if the subcircuit is in Spectre's Spice compatibility mode. For the model path subcircuit file, the subcircuit is always in Spice compatibility mode. For an include file that has the Spectre syntax mode selected on the *Setup – Environment* form, the file will be in Spectre native mode. If you use an include file and you use cdsSpice as the syntax mode, then the file will be in Spice compatibility mode. If the save command appears in the Spectre native section of the netlist as it does in 4.4.x, Spectre will treat the save requests in a case-sensitive manner. The problem is that your subcircuit may be in a Spice compatibility mode section and Spectre ALWAYS converts the lines in spice compatibility section lines to lower case.

  If your subcircuit is in Spectre native format in the final netlist, with the appropriate `simulator lang=spectre` commands before it, you must use the case-sensitive device name the way you used it in the subcircuit.

  If your subcircuit is in a Spice compatibility section, then you will need to use the lower-case names for the devices.

- You MUST dump and edit the CDF file for the cell to include the *noPortDelimiter* field. The *noPortDelimiter* option suppresses the insertion of a `:` at the end of the mappedRoot() value before the argument to the mappedRoot() is appended. Looking at these two save statements will make it clearer.

  ```
  save xcap1:.Xcapb.c1:1
  save xcap1.Xcapb.c1:1
  ```

- In the first case, noPortDelimiter was either set to `nil` or was missing from the simInfo section. In the second case, noPortDelimiter was set to t so the netlister didn't insert the port delimiter ":" after the instance name but before the argument to mappedRoot().

  To edit the simInfo section of the CDF:

  a. Type the following in the CIW to dump the CDF to a file:

  In 4.3,

  ```
  CdfDumpCellCDF("LibName" "CellName")
  ```

  A new window will appear. Save the content to a file.

  In 4.4,

  ```
  cdfDump("LibName" "fileToDumpTo"
  ?cellName "CellName")
  ```

  b. Now, edit the `fileToDumpTo` and look for the spectreS simInfo section. It will look something like this:

  ```
  cdfId->simInfo->spectreS = ’( nil
  current           port
  propMapping       nil
  termMapping       (nil
  In
  "FUNCTION(mappedRoot('.c1:1')+mappedRoot(\
  ".c2:2'))"
              Out
  "FUNCTION(minus(root('In')))"
           )
          modelArguments    nil
          instParameters    (cap1 cap2)
          otherParameters   (macro)
          namePrefix        "X"
          termOrder         ("In" "Out")
          componentName     subcircuit
          macroArguments    (cap1 cap2)
          netlistProcedure  ansSpiceSubcktCall
  ```

```
        noPortDelimiter   t
    )
```

Notice that you probably don’t have the `noPortDelimiter` item. You will need to add it to be able to probe subcircuits.

c. Once the `noPortDelimiter` item has been added, save your work, go back to the CIW and type:

```
load("fileToDumpTo")
```

- For a two-terminal device, Spectre will save whatever terminal you specify and the framework will correctly calculate the total current from the `termMapping`, but the DRL Browser will only assume the first terminal is saved and it will only display the first terminal in the Browser.

  In other words, if you choose to save the second terminal of a two-terminal device, Spectre will save it, the IT and other functions will correctly sum the value, but the Browser will show you the first terminal in the Browser window (:1 at the end instead of :2). In other words, you are better off saving the first terminal and using either a math operator or the minus() function to negate the first terminal’s value. You don’t need to worry about this for any three or more terminal devices.

- For a multi-terminal device, you can use either the numeric terminal name (1 2 3 4 etc.) or the names given in the `spectre -help [device]` device synopsys. This means you can refer to the gate of a mos device either with ".m23:g" or with ".m23:2" since the gate terminal is terminal 2. If you don’t provide a terminal number, Spectre saves ALL of the terminals for the device AND Spectre saves all of the transient operating information for the device (vgs, vds, ids, etc.). However, if you want the current summation to work, you must explicitly specify a terminal name (1 2 3 4 or D G S B) as this is the only way the summation will know which terminal to use in summing the current. It would seem that Spectre is saving the name (e.g. ends in `:g`), but the summation is looking for either no ending or a `:1` ending.

- You can use the save statement to your advantage in some ways by creating a *dummy* pin on a device and assigning the dummy pin’s current to be a particular parameter like vdsat. If you do this, you can now see the vdsat parameter’s value as a function of time, something that the Operating Point analysis cannot do. Operating Point analysis only saves the DC Operating point and a snapshot of operating points at the very last transient timestep. Use this termMapping to create a terminal VDSAT and have IT(`“VDSAT"`) show you the vdsat value of device m23 as a function of time. termMapping (VDSAT "(FUNCTION mappedRoot('.m23:vdsat'))")

Note: The above information does not apply to Hspice termMapping.

Setting Up a Cadence SPICE Analysis

To set up analyses for Cadence® SPICE,

1. Choose *Analyses – Choose*.

   The Choosing Analyses form appears.

2. Choose an analysis.

   The Choosing Analyses form redraws to show the parameters for the new analysis.

3. Set the options and click *Apply*.

4. Choose another analysis to set up.

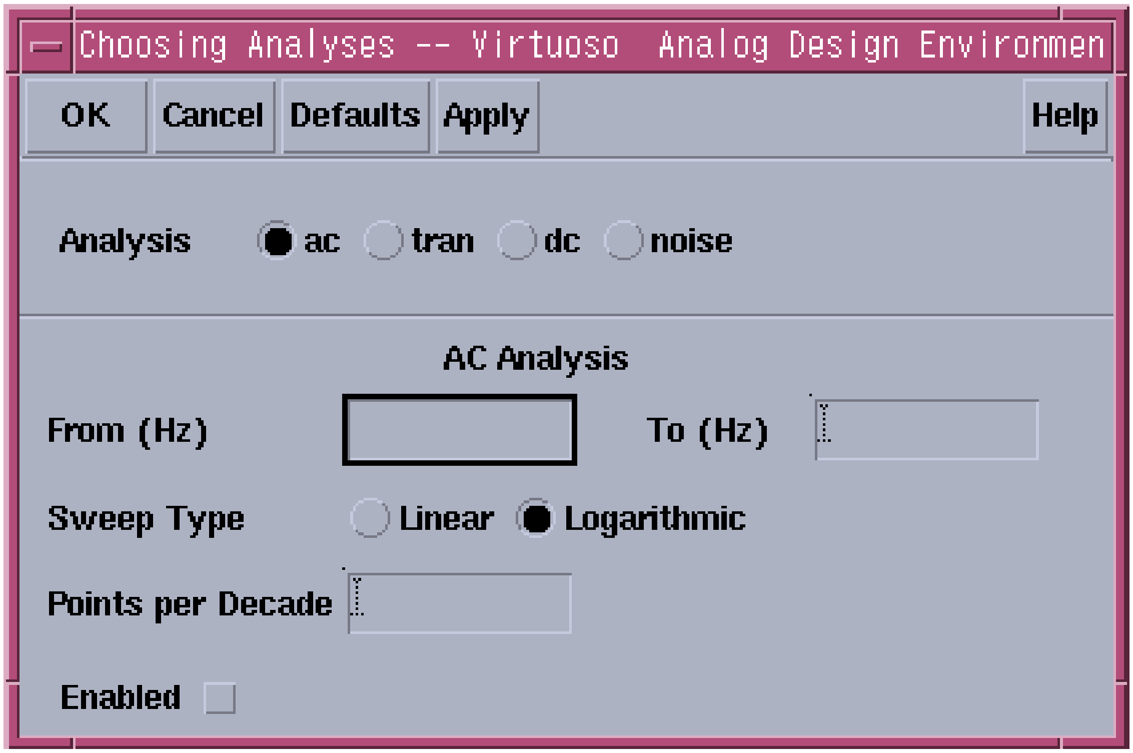
The next step is usually selecting the outputs you want to save.

For help setting up a particular analysis, refer to the *Cadence SPICE Reference Manual*.

AC Analysis

To set up an AC small-signal analysis around the DC operating point,

1. In the Choosing Analyses form, choose *ac* for *Analysis*.



Analysis ● ac ○ tran ○ dc ○ noise

AC Analysis

From (Hz)
To (Hz)

Sweep Type ○ Linear ● Logarithmic

Points per Decade

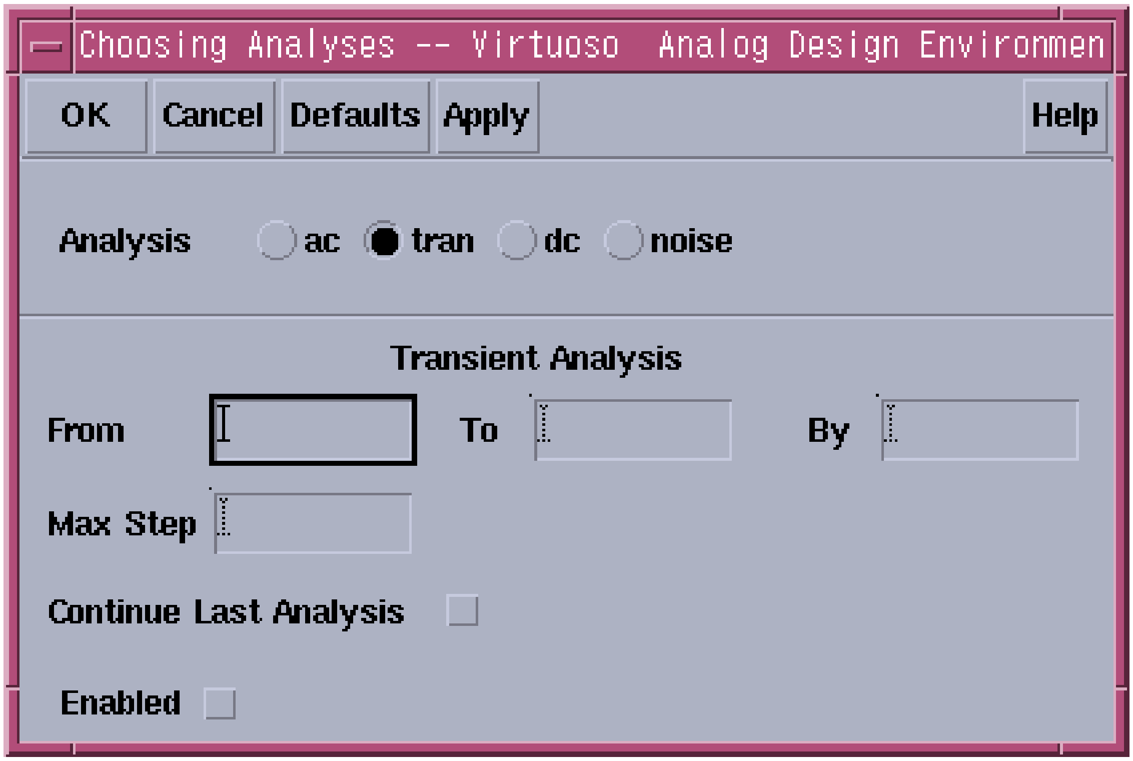
Enabled □

2. Type a starting and stopping frequency in *From* and *To*.
3. Choose a sweep type option and enter the frequency increment in hertz or points per decade.
4. Click *Apply*.

Transient Analysis

To set up a transient analysis,

1. In the Choosing Analyses form, choose *tran* for *Analysis*.



Analysis ○ ac ● tran ○ dc ○ noise

Transient Analysis

From ▭ To ▭ By ▭

Max Step ▭

Continue Last Analysis □

Enabled □

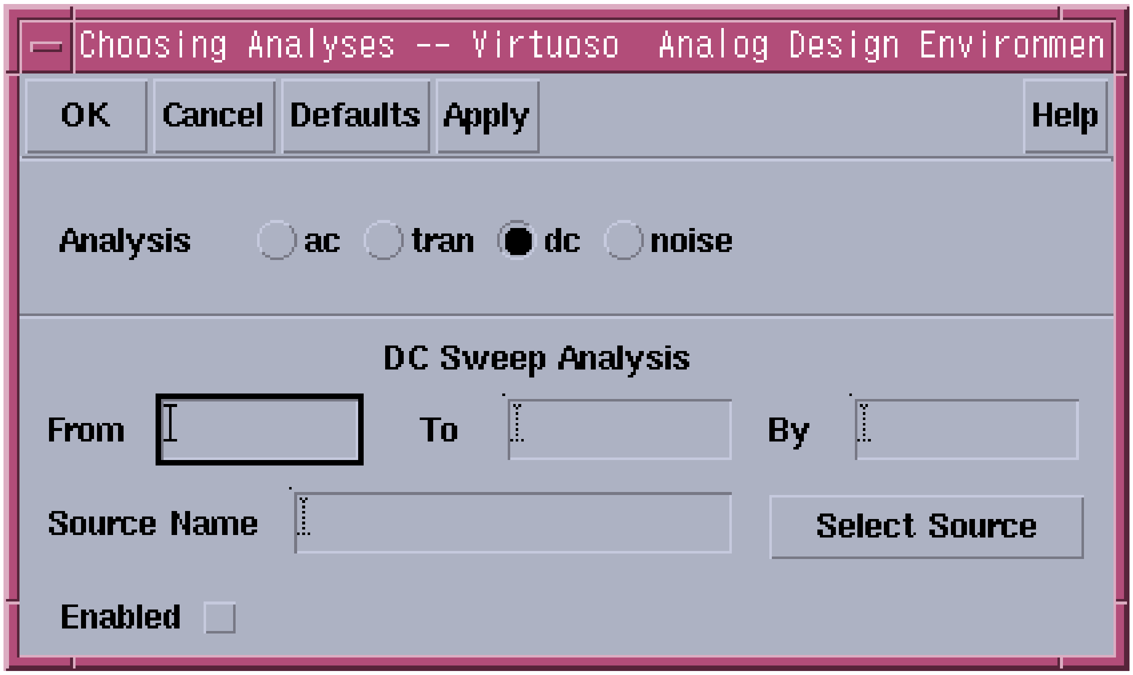
2. Type the starting and stopping times in *From* and *To*.
3. Type the time increment in *By*.
4. Set the maximum timestep (DELMAX) in *Max Step*, if desired.
5. Click *Apply*.

Note: Use *Continue Last Analysis* to finish an interrupted analysis, to extend one that finished normally, or to continue one that was interrupted with a cdsSpice *break* parameter.

DC Analysis

To set up a source-sweep (DC) analysis,

1. In the Choosing Analyses form, choose *dc* for *Analysis*.



Analysis ○ ac ○ tran ● dc ○ noise

DC Sweep Analysis

From
To
By

Source Name

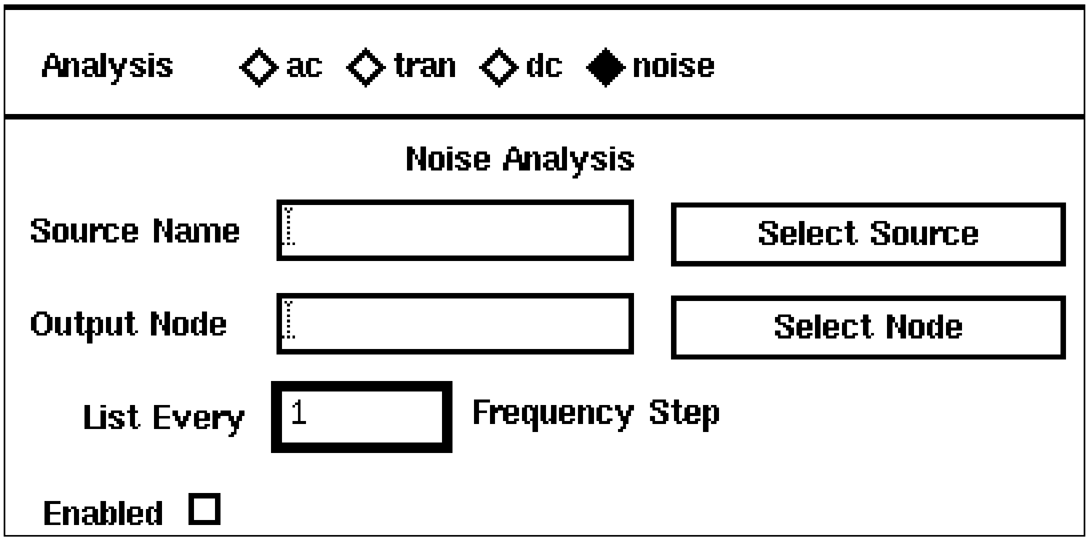
Enabled □

2. Type the starting and stopping times in *From* and *To*.
3. Type the time increment in *By*.
4. Type the name of the current or voltage source in *Source Name*, or click *Select Source* and click the source in the Schematic window.
5. Click *Apply*.

Noise Analysis

To set up a noise analysis,

1. In the Choosing Analyses form, choose *noise* for *Analysis*.



Analysis ◇ ac ◇ tran ◇ dc ◆ noise

List Every 1 Frequency Step

Enabled □

2. Set up an AC analysis.
3. Specify the noise source by typing the source name or by clicking *Select Source* and clicking on the source in the Schematic window.
4. Specify *Output Node* where the total noise contribution is completed, or click *Select Node* and click a net in the Schematic window.

Note: When you select a noise analysis, the system turns off the *Save All Node Voltages* option for cdsSpice. Use the *Outputs – To Be Saved – Select on Schematic* command to save the specific nodes you want to look at.

6

Selecting Data to Save, Plot, or March

This chapter shows you how to select data that you want to save, plot, or march.

About the Saved, Plotted, and Marched Sets of Outputs

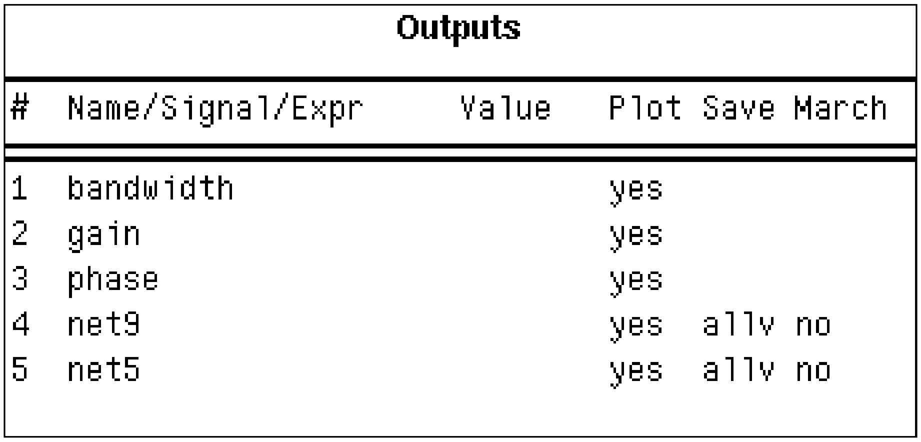
The Virtuoso® Analog Design Environment keeps track of three sets of nets and terminals:

- The saved set, for which simulation data is written to disk
- The plotted set, which is automatically plotted after simulation in the Waveform window

  The plotted set can also contain expressions.
- The marched set, which is plotted in the Marching Waveform window during simulation

The contents of all three sets of outputs are listed in the *Outputs* section of the Simulation window and in the Setting Outputs form. Up to 999 outputs can be displayed in the Simulation window.

In the figure below, all five signals will be plotted and two will be saved after simulation. None will be marched during simulation.



Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	bandwidth		yes		
2	gain		yes		
3	phase		yes		
4	net9		yes	allv	no
5	net5		yes	allv	no

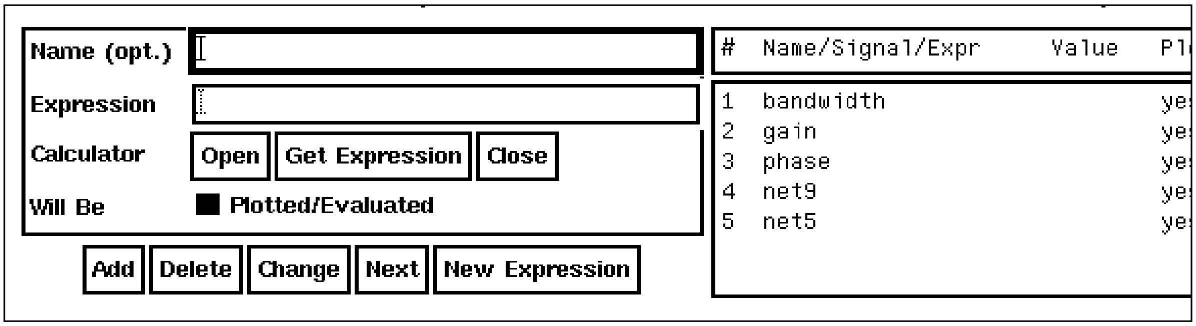
Note: If you click a net (*Name/Signal/Expr*) repeatedly, the highlighting will toggle on and off, and the signal will appear and disappear from the *Outputs* list.

Opening the Setting Outputs Form

You set up the saved, plotted, and marched sets of outputs with the Setting Outputs form.

➤ In the Simulation window, choose *Outputs – Setup*, or from the Schematic window, choose *Setup – Outputs*.

The Setting Outputs form appears.



Name (opt.)

Expression

Will Be ■ Plotted/Evaluated

#	Name/Signal/Expr	Value	Pl
1	bandwidth		ye
2	gain		ye
3	phase		ye
4	net9		ye
5	net5		ye

For detailed information about the form, see “Setting Outputs” on page 292.

Deciding which Outputs to Save

Saving all the node voltages and terminal currents for a large design produces an enormous data set. The analog circuit design environment lets you save a selected set of voltages and currents from the schematic.

Once you select a set of output nodes and terminals, you can save their names to a file using the *Save State* command. Then, if you resimulate the design and want to view the same voltages and currents, you can load the set from the state file.

After you select the outputs you want to save, the next step is generally to start the simulation.

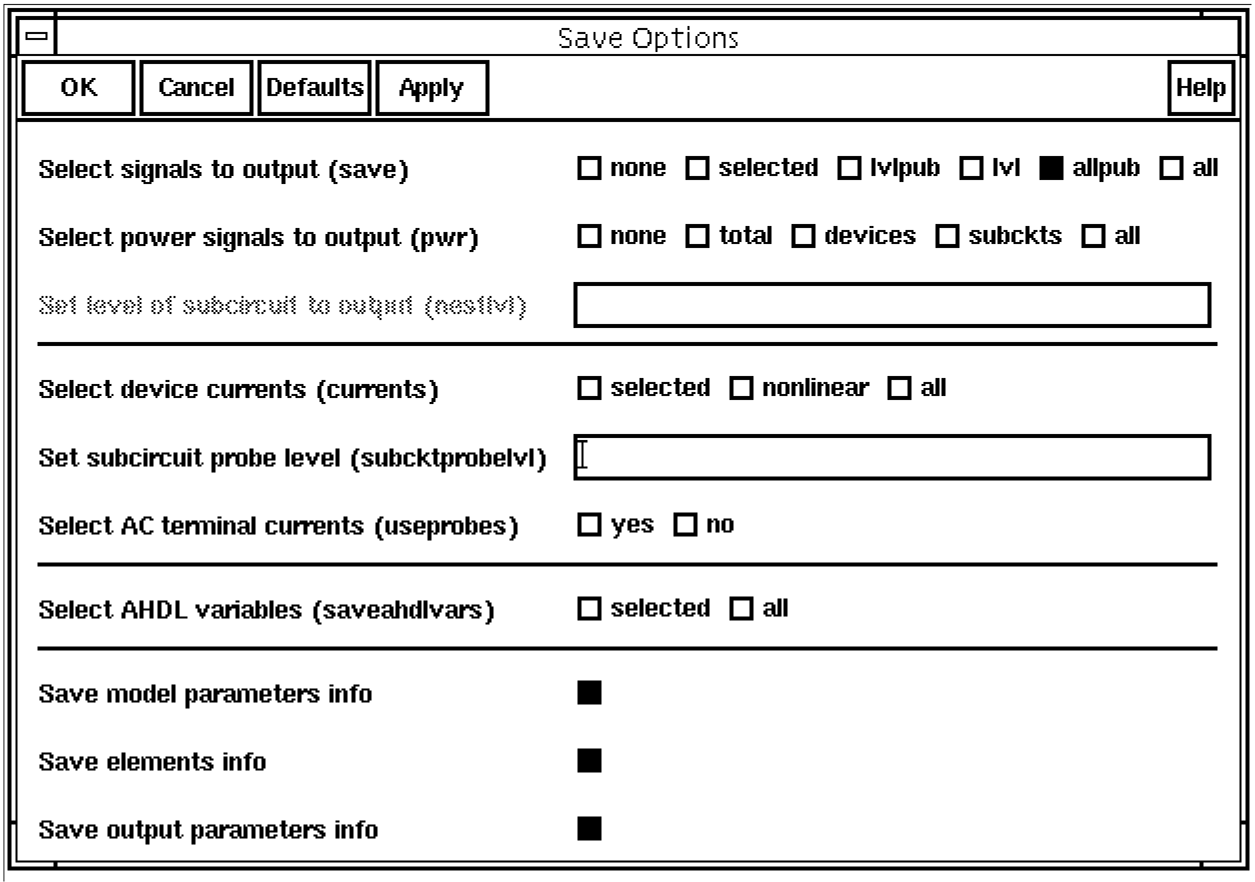
Saving All Voltages or Currents

To save all of the node voltages and terminal currents,

1. In the Simulation window, choose *Outputs – Save All*, or in the Schematic window, choose *Setup – Save All*.

   A form appears that varies according to the simulator you use. For example, if you use the Spectre simulator, the Save Options form appears with the following format.

For detailed information about the form, see “Save Options and Keep Options” on page 293.



Select signals to output (save) □ none □ selected □ lvlpub □ lvl ■ allpub □ all

Select power signals to output (pwr) □ none □ total □ devices □ subckts □ all

Set level of subcircuit to output (nestlvl)

Select device currents (currents) □ selected □ nonlinear □ all

Set subcircuit probe level (subcktprobelvl)

Select AC terminal currents (useprobes) □ yes □ no

Select AHDL variables (saveahdlvars) □ selected □ all

Save model parameters info ■

Save elements info ■

Save output parameters info ■

With AMS, the *Save All* command is not on by default. Save (or probe) statements are placed in the `amsControl.tcl` file and not in the `amsControl.scs` file. For information about this, see the “Tcl-Based Debugging” appendix chapter of the *Virtuoso AMS Simulator User Guide.*

2. Select the values you want to save and click *OK*.

Note: When you set up a noise analysis with cdsSpice only, the system turns these options off. If you later deactivate the noise analysis, the system reactivates the *Select all* options.

Note: For AMSUltra, the options—*Save model parameters info*, *Save elements info*, and *Save output parameters info*—appear deselected by default. If you select them, the speed with which AMSUltra typically works may be compromised especially if you have a big design.

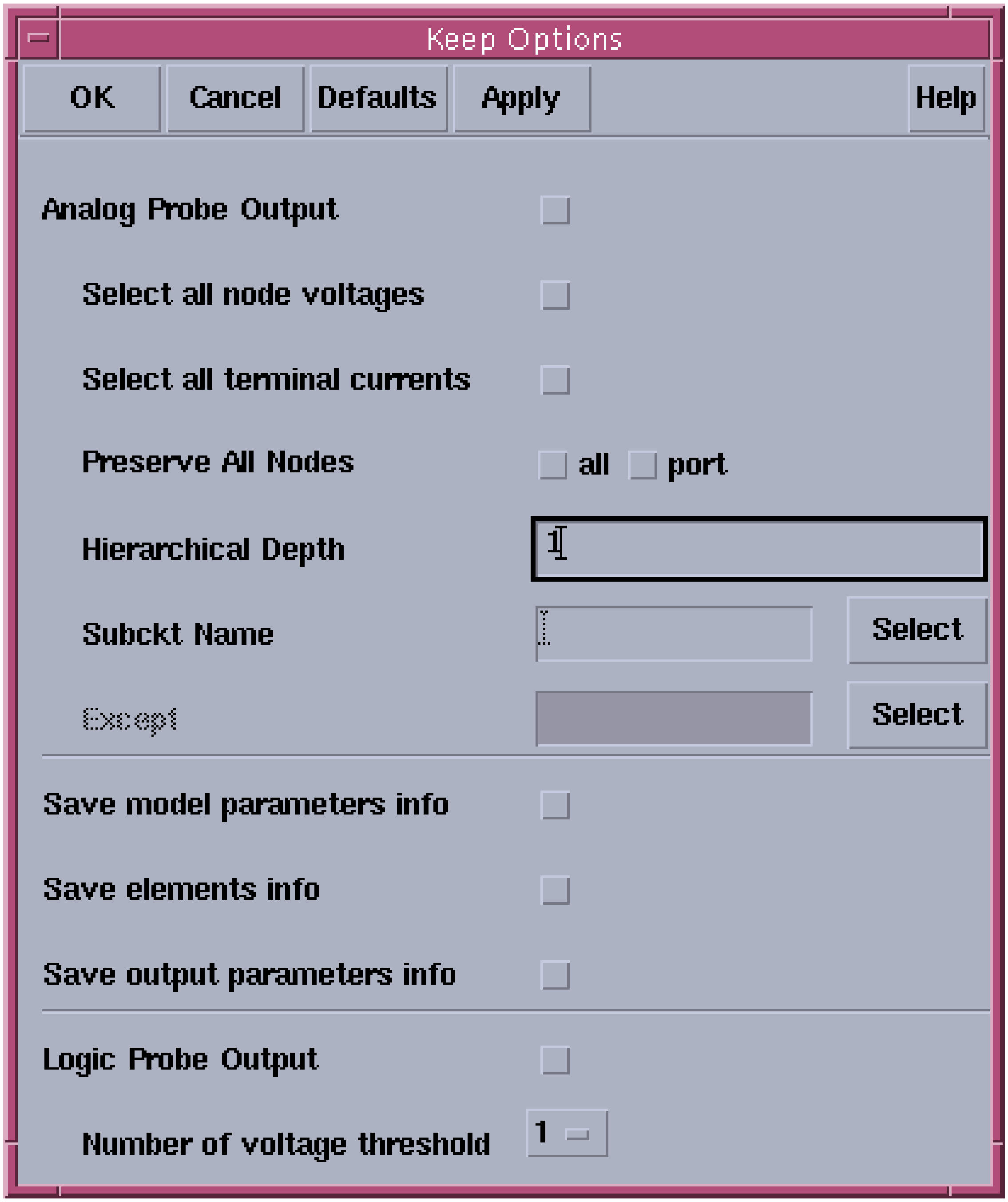
For details about selecting device currents (*currents*), setting subcircuit probe level (*subcktprobelvl*) and selecting AC terminal currents (*useprobes*), refer to the “Specifying Output Options” chapter of the *Spectre Circuit Simulator User Guide.*

Saving Outputs for UltraSim Simulations

To view outputs for waveform data:

1. In the Simulation window, choose *Outputs – Save All*.

   The Keep Options form appears.



Analog Probe Output □

Select all node voltages □

Select all terminal currents □

Preserve All Nodes □ all □ port

Hierarchical Depth 1

Subckt Name

Except

Save model parameters info □

Save elements info □

Save output parameters info □

Logic Probe Output □

Number of voltage threshold 1

Analog Probe Output

To output waveform data for an analog probe, choose the appropriate settings.

❑ **Select all node voltages** use to output all node voltages.

❑ **Select all terminal currents** use to output all terminal currents. The Virtuoso UltraSim simulator outputs the first terminal current of each device.

❑ **Preserve All Nodes** use to preserve either all or port RC node voltages. RC nodes are not reduced, allowing the nodes to be saved in simulation.

❑ **Hierarchical Depth** use to save and display more than one level of hierarchical results.

❑ **Except** specifies the nodes to be excluded from the analog probe. A node name, element name, or wildcard (*) can be used.

❑ **Save model parameters info** specifies that input parameters for models of all components be saved.

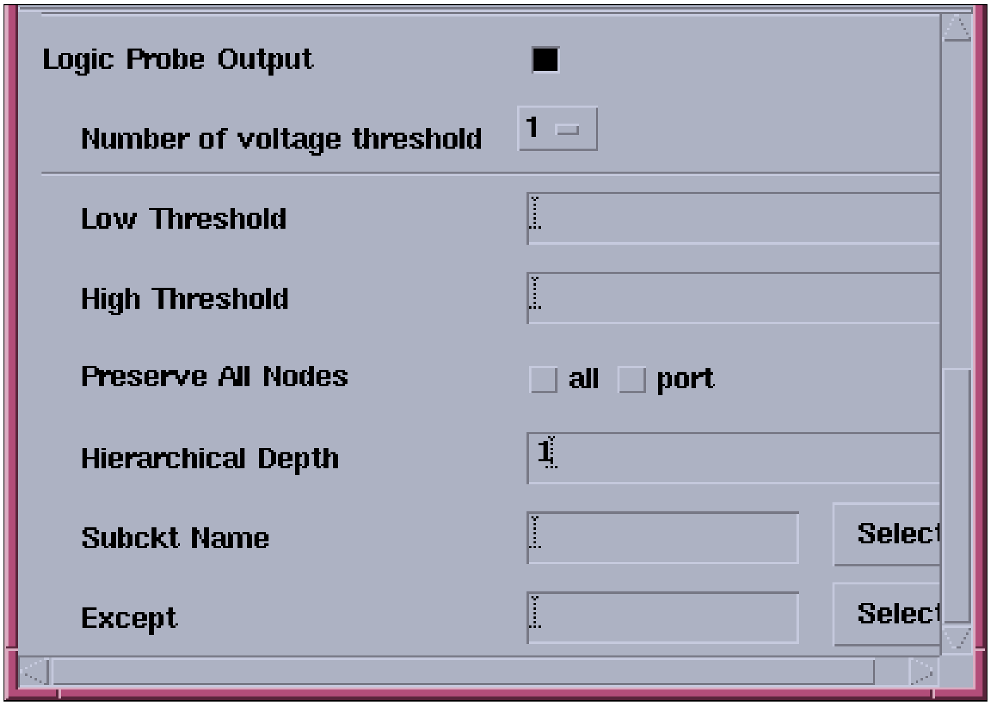
❑ **Save elements info** specifies that input parameters for instances of all components be saved.

❑ **Save output parameters info** specifies that effective and temperature-dependent parameter values be saved.

Logic Probe Output

To output waveform data for a logic probe, choose the appropriate settings (the Keep Options form expands to show the *Logic Probe Output* settings).

Note: The *Logic Probe Output* option is available only for SST2 and FSDB waveform output formats.



Logic Probe Output ■

Number of voltage threshold 1

Low Threshold

High Threshold

Preserve All Nodes □ all □ port

Hierarchical Depth 1

Subckt Name

Except

- **Number of voltage threshold** use to indicate the number of voltage thresholds for each logic probe.
- **Low Threshold** specify for each logic probe the low threshold value which corresponds to the digital `0` value.
- **High Threshold** specify for each logic probe the high threshold value which corresponds to the digital `1` value.
- **Preserve All Nodes** use to preserve all or only port RC nodes from the RC reduction.
- **Hierarchical Depth** use to save and display more than one level of hierarchical results.
- **subckts** use to indicate the subcircuit name for the logic probe. If a name is not entered into the `subckts` field, the simulator applies the logic probe to all blocks.

2. Click *OK*.

Saving Selected Voltages or Currents

To save the simulation data for particular nodes and terminals,

1. In the Simulation window, choose *Outputs – To Be Saved – Select on Schematic*, or in the Schematic window, choose *Setup – Select on Schematic – Outputs to be Saved*.
2. In the Schematic window, choose one or more nodes or terminals.

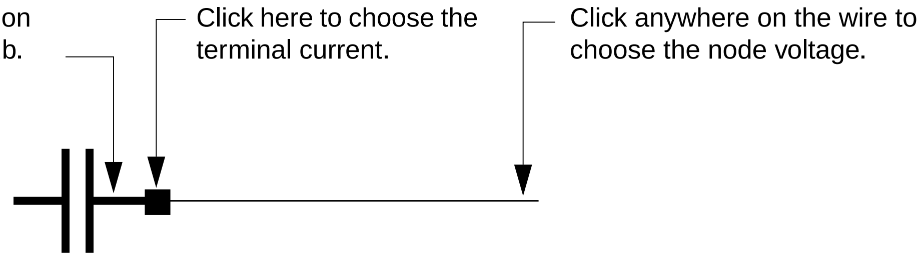
   The system circles pins when you choose a current and highlights wires when you choose a net.

   ❑ Click on an instance to choose all instance terminals.

   ❑ Click on the square pin symbols to choose currents.

   ❑ Click on wires to choose voltages.

   ❑ Click and drag to choose voltages by area.


Never click on the wire stub.
Click here to choose the terminal current.
Click anywhere on the wire to choose the node voltage.

3. Press the `Esc` key when you finish.

Adding a Node or Terminal to a Set

To add a node or terminal to the saved, marched, or plotted sets,

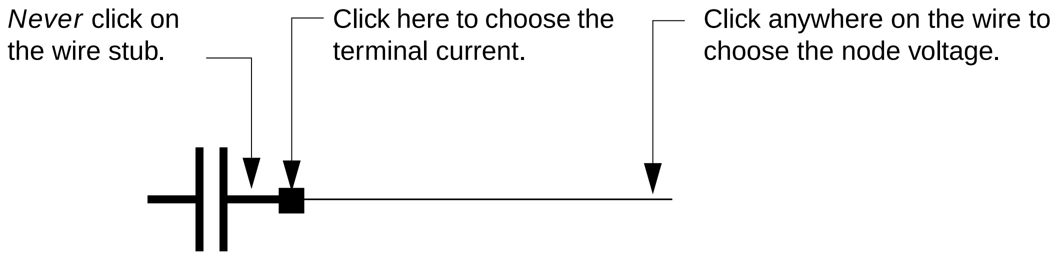
1. Choose one of the *Outputs – To Be – Select on Schematic* commands in the Simulation window, or choose *Setup – Select on Schematic – Outputs to be* in the schematic.
2. In the Schematic window, choose one or more nodes or terminals.

   The system circles pins when you choose a current and highlights wires when you choose a net.

   ❑ Click on the square pin symbols to choose currents.

   ❑ Click on wires to choose voltages.

- ❑ Click and drag to choose voltages by area.


Never click on the wire stub.
Click here to choose the terminal current.
Click anywhere on the wire to choose the node voltage.

3. Press the `Esc` key when you finish.

To select nodes and terminals in lower-level schematics to be plotted, saved, or marched,

1. In the Simulation window, choose *Outputs – To Be – Select on Schematic*, or in the Schematic window, choose *Setup – Select on Schematic – Outputs to be*.
2. In the Schematic window, choose *Design – Hierarchy – Descend Edit* and click on an instance.
3. Click *OK* in the form that appears.
4. In the Schematic window, choose one or more nodes or terminals.
5. Press the `Esc` key when you finish.

Note: For cdsSpice only, data for nets and terminals in lower-level schematics is not saved when you use the Keep Options form to save all currents. You must explicitly choose each terminal with the *Outputs – To Be Saved – Select on Schematic* command.

Adding a Saved Node to the Plot or March Set

To add an output in your saved set to the plotted or marched set,

1. In the Simulation window, click in the *Outputs* list to choose the output.

   To select more than one output, hold down the `Control` key while you click on the outputs, or click and drag.

   To deselect a highlighted output, hold down the `Control` key while you click on it.

2. Choose one of the following:

- *Outputs – To Be Marched – Add To*
- *Outputs – To Be Plotted – Add To*

The outputs table is updated to show the outputs have been added to the saved or marched sets.

Note: You can use the *Outputs – To Be – Remove From* commands to remove highlighted outputs from a set.

Removing Nodes and Terminals from a Set

To remove a node or terminal from the saved, plotted, or marched set,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. Double-click on the node or terminal in the *Table Of Outputs* list box.

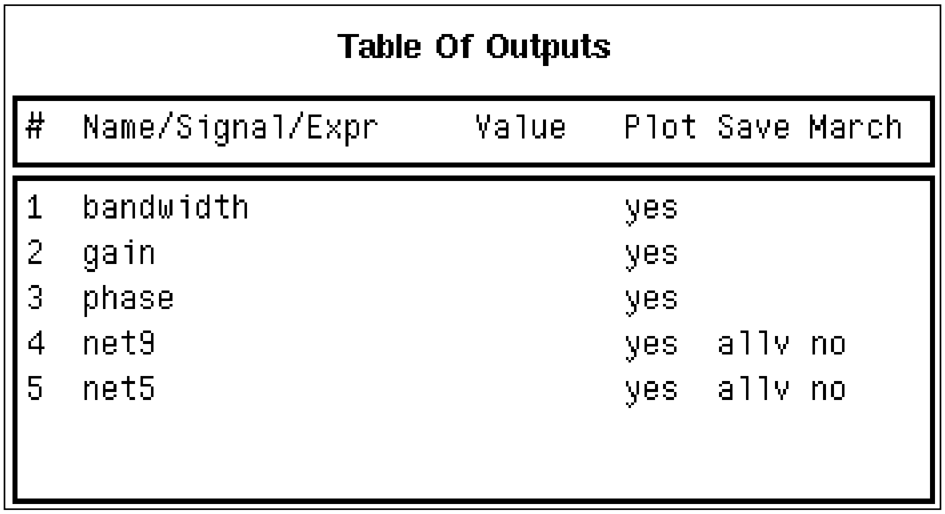


Table Of Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	bandwidth		yes		
2	gain		yes		
3	phase		yes		
4	net9		yes	allv	no
5	net5		yes	allv	no

3. Click to deselect the appropriate *Will Be* boxes.
4. Click *Change*.

Note: To remove a node from all three sets (delete it), highlight the node in the Simulation window and choose *Outputs – Delete*.

Saving a List of Outputs

You can save both

- The data for a set of outputs
- The list of saved, marched, and plotted outputs itself

The saved list includes output expressions.

To save the list of saved, marched, and plotted outputs,

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Environment – Save State*.

   The Saving State form appears.

2. Type a name for the saved simulation state.
3. Check that the *Outputs* box is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring a Saved List of Outputs

To restore a saved set of outputs,

1. In the Simulation window, choose *Session – Load State*, or in the Schematic window, choose *Analog Environment – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Choose a run directory with the *Cell* and *Simulator* fields.

   The display shows the saved states for the cell and simulator combination.

3. Click on a state name.
4. Check that *Outputs* is selected and click *OK*.

Conditional Search for Results

After running a simulation, you can search the results for components in the saturation region, breakdown region, or any user-defined region. To do a conditional search for results, choose *Results – Circuit Conditions* from the *Simulation* menu. Follow the procedure below to search for circuit conditions.

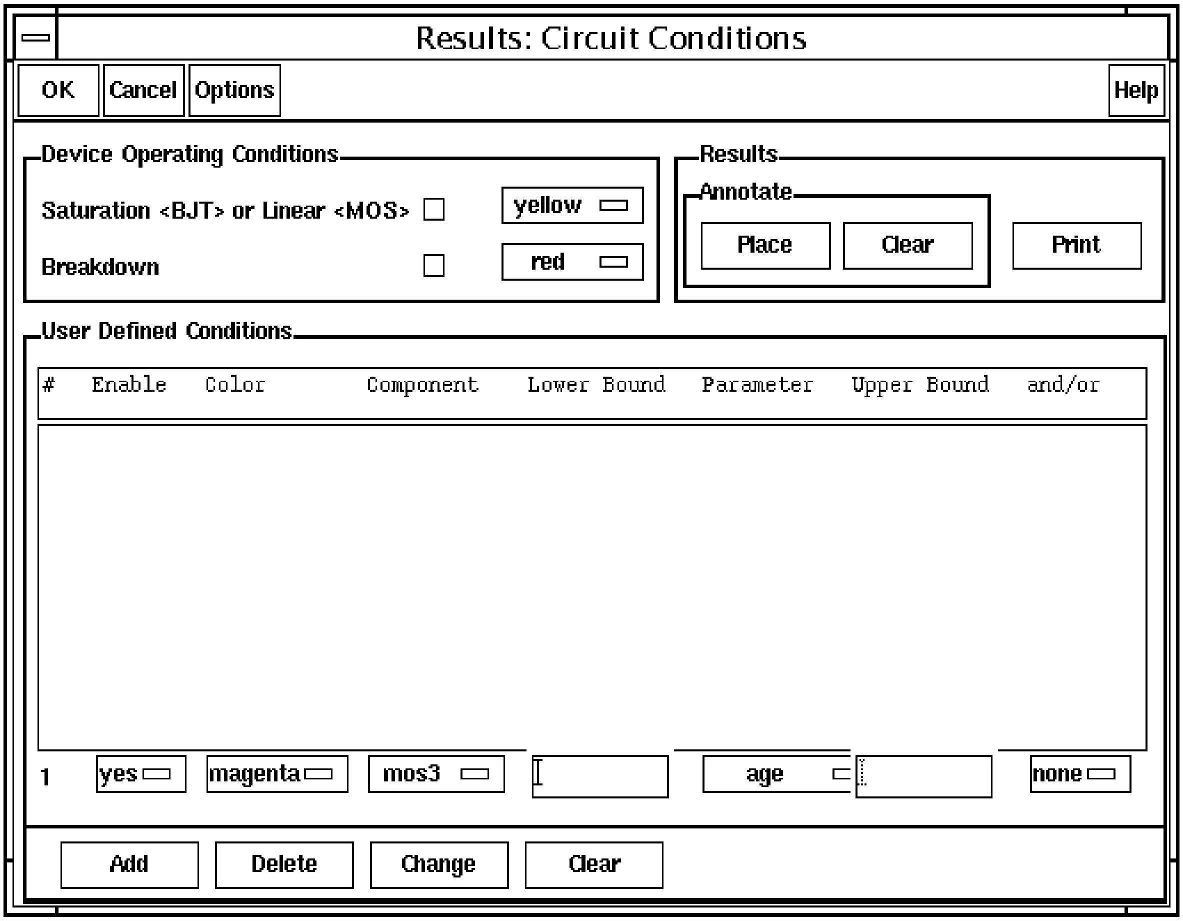
1. Run a simulation.

   Note: You must run a DC operating-point analysis to use the circuit conditions capability.

2. Choose *Results – Select* and indicate the results that you wish to search.

3. Choose *Results – Circuit Conditions* from the Simulation window.

   The Circuit Conditions form appears:



Results: Circuit Conditions

Device Operating Conditions

Saturation <BJT> or Linear <MOS> □ yellow

Breakdown □ red

User Defined Conditions

#	Enable	Color	Component	Lower Bound	Parameter	Upper Bound	and/or
1	yes	magenta	mos3		age		none

   For detailed information about the form, see “Circuit Conditions” on page 289.

4. Choose device operating conditions.

   You can choose to view components in the saturation (for BJT devices), linear (for MOS devices), or breakdown region.

   Note: The appropriate model parameters must be set for the simulator to calculate these conditions. These features might not be available for simulators other than spectre or cdsSpice.

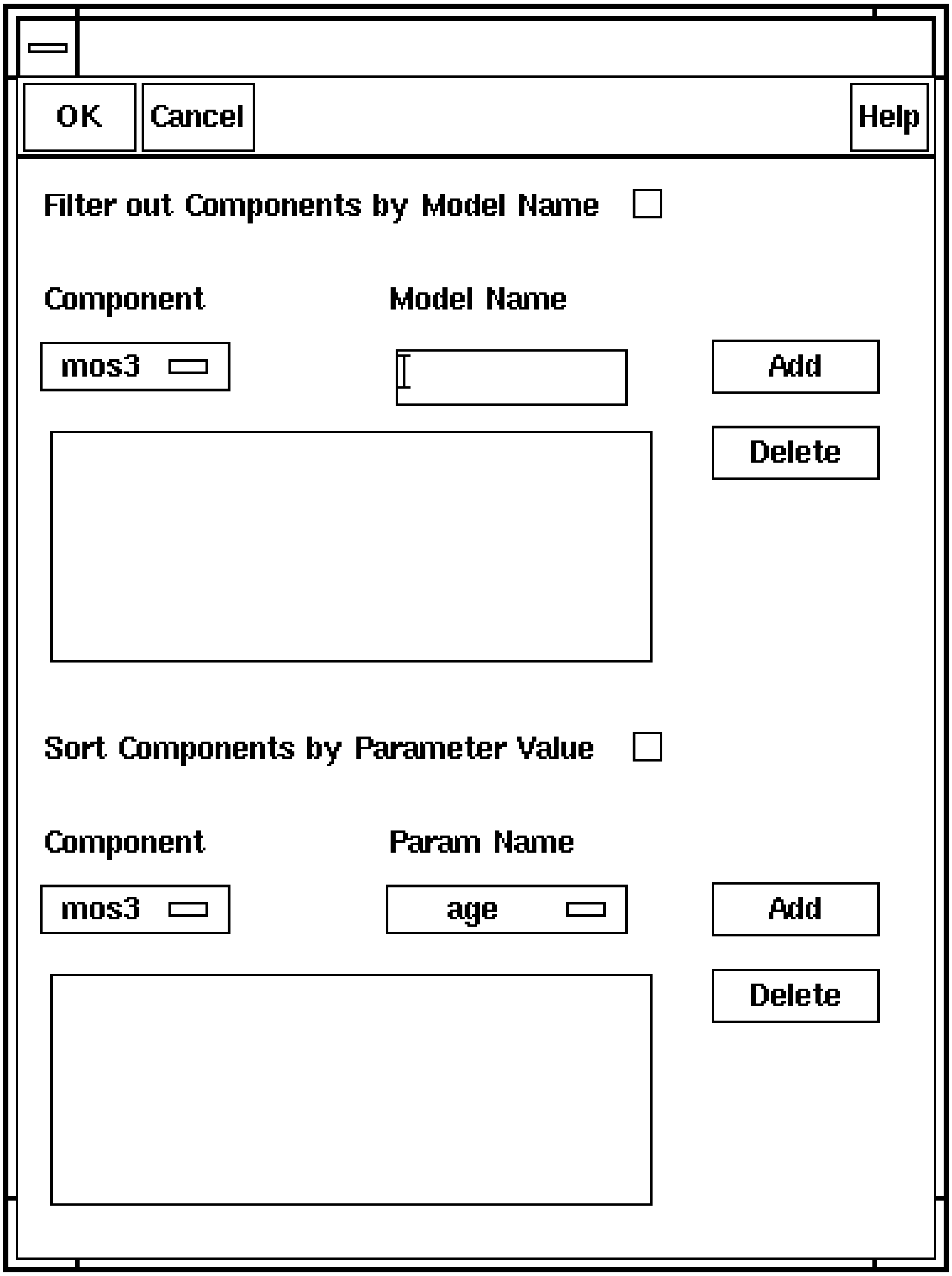
5. Set up *User Defined Conditions*.

   You use the cyclic and type-in fields to create the custom conditions you want to search for.

6. View the results of the conditions you chose by doing the following.

- ❑ Click *Place* to highlight the instances that meet the specified conditions on the schematic.
- ❑ Click *Print* to print the values of instances that meet the specified conditions in a print window.

7. Clicking on the *Options* button will bring up a form where you can specify filter and sort conditions.



Filter out Components by Model Name □

Component: mos3
Model Name:

Sort Components by Parameter Value □

Component: mos3
Param Name: age

In the *Filter out Components by Model Name* section, you can enter filters using the cyclic field displaying all the component types and the text entry field to type in model names. After you have selected the component type and entered a model name, press *Add* to add the filter to list of filters. You can select one or more filters in the list and then click *Delete* to delete the filters. The filters are active only when the *Boolean* button is on. When the filters are active, any component that matches a filter will be filtered out from the output of *Print* button.

The next section is *Sort components by Parameter Value*. Users can use the two cyclic fields to enter sorting criteria for a component type. When this section is active (Boolean is on) the output from *Print* for user defined conditions will be sorted according to the sort variable for given component type.

Form Field Descriptions

Circuit Conditions

Device Operating Conditions

These checkboxes let you highlight components in saturation and in breakdown. When the *Annotate Place* button is pressed, components in breakdown, saturation, or both are highlighted on the schematic with a colored box. The color of the box is chosen by the color cyclic field next to each field.

Saturation

For Spectre saturation, an instance is highlighted if

- For BJT: If the operating point parameter region=3
- For MOS/bsim: If the operating point parameter region=2

For cdsSpice saturation, an instance is highlighted if

- The operating point sat=0

Breakdown

For Spectre breakdown, an instance is highlighted if

- For BJT: If the operating point parameter region=4

For cdsSpice breakdown, an instance is highlighted if

- The operating point bkdwn=0

Note: For the simulator to calculate breakdown or saturation, the appropriate model parameters need to be set.

User-Defined Conditions

Enable uses the cyclic field to select yes or no to enable or disable a condition.

Color shows the color with which you want to highlight instances meeting a condition.

Component shows the type of component for which you want to create conditions.

Lower Bound specifies the lower boundary of a parameter’s value.

Upper Bound specifies the upper boundary of a parameter’s value.

Parameter is an operating-point parameter you choose from the cyclic field. The *Lower Bound* and *Upper Bound* values apply to the selected parameter.

and/or sets Boolean arguments to a condition. When *and* is used, both conditions must be met for an instance to be highlighted. When *or* is used, either condition must be met for an instance to be highlighted. Both operators have the same precedences.

Add adds another compound condition to the existing entries in the table. When this button is clicked, a new row is added to the bottom of the table so that a designer can specify another search condition.

Delete removes a condition from the table. When this button is clicked the selected entries in the table are removed. You select entries by clicking on a row in the *User Defined Conditions* box.

Change lets you modify a user-defined condition. You must select the condition before modifying it.

Clear lets you clear all the entries from the *User Defined Conditions* box.

Results

Annotate Place uses the conditions specified in the form above to search through the simulation’s DC operating point data. The data matching the conditions specified in the table are filtered and the instances are highlighted. Components are highlighted with boxes. The Circuit Conditions form remains open until you click either the *OK* or *Cancel* button. In hierarchical designs, if the component that needs to be highlighted is within a block, the block is highlighted. When you push into the highlighted block so that the component is displayed, the component is then highlighted. Whenever *Annotate* is selected, the currently highlighted instances are cleared and the new ones redrawn. If a component meets more than one condition, it is highlighted with a third color and a message prints in the CIW.

Annotate Clear, when selected, clears all of the highlighted instances from the Schematic window.

Print, when selected, prints the results to the print window, which displays the results as a table. The results are defined as the components that match the conditions specified in the Circuit Conditions form with their state.

Setting Outputs

Name (opt.) is an optional name for the signal, which appears in the *Table Of Outputs* list box and in the Waveform window.

Expression is the calculator expression to plot, save, or march.

Calculator buttons are displayed only while no net or terminal is selected in the *Table Of Outputs* list box.

> **Open** opens the calculator.
>
> **Get Expression** copies the expression in the calculator buffer into the *Expression* field.
>
> **Close** dismisses the calculator window.

Will Be changes depending on whether an expression or a signal is selected.

> **Plotted/Evaluated** plots or prints the value of the expression after each simulation.

Add creates the output you set up in the *Selected Output* area.

Delete removes the highlighted output. Click in the *Table Of Outputs* list box to highlight an output.

Change updates the highlighted output with the new settings in the *Selected Output* area.

Next moves the highlight to the next signal or expression in the *Table Of Outputs* list box. This allows you to make changes to consecutive entries in the *Table Of Outputs* list box without clicking on each entry.

New Expression clears the *Selected Output* area so you can enter a new output.

Save Options and Keep Options

The title of this form and the options displayed vary depending on the simulator you use. If you are using direct simulation, see the following section for information. If you are using a socket simulator, see “Keep Options Form for Socket Simulation” on page 294.

Save Options Form for Direct Simulation (Spectre)

For detailed information about the fields in the following table, follow the cross- references. All of the cross-references are to sections in the “Specifying Output Options” chapter of the *Spectre Circuit Simulator User Guide*.

Field	**For more information, see**
Select signals to output (save)	The save Parameter Options
Select power signals to output (pwr)	Saving Power of the *Virtuoso Spectre Circuit Simulator User Guide.*
Set level of subcircuit to output (nestlvl)	Saving Groups of Signals of *Virtuoso Spectre Circuit Simulator User Guide*.
Select device currents (currents)	Saving Groups of Currents of *Virtuoso Spectre Circuit Simulator User Guide.*
Set subcircuit probe level (subcktprobelvl)	Saving Subcircuit Terminal Currents of *Virtuoso Spectre Circuit Simulator User Guide.*
Select AC terminal currents (useprobes)	Setting Multiple Current Probes of *Virtuoso Spectre Circuit Simulator User Guide.*
Select AHDL variables (saveahdlvars)	Saving All AHDL Variables of *Virtuoso Spectre Circuit Simulator User Guide.*

Caution

The *all* ***buttons (*** *Select signals to output (save)* ***) are global buttons but can be locally overriden while specifying options for an particular analysis.***

The other fields in the Save Options form are described below.

Save model parameters info specifies that input parameters for models of all components be saved.

Save elements info specifies that input parameters for instances of all components be saved.

Save output parameters info specifies that effective and temperature-dependent parameter values be saved.

Keep Options Form for Socket Simulation

Select all node voltages specifies that all node voltages be maintained for plotting.

Select all terminal currents specifies that all terminal currents be saved for plotting.

Select all DC/Transient terminal currents specifies that all DC and transient terminal currents be saved for plotting.

Select all AC terminal currents (allAnalogACTC) specifies that all AC terminal currents will be saved for plotting by setting the spectreS `allAnalogACTC` option. Selecting this option significantly increases transient analysis simulation time.

Select all digital node voltages specifies that all digital node voltages be saved for plotting.

Save All AHDL Module Variables specifies that all the AHDL variables belonging to all the AHDL instances in the design be saved.

7

Running a Simulation

This chapter describes how to run a simulation that you have set up.

Prerequisites to Simulation

Before running a simulation, you need to

- Start the Virtuoso® Analog Design Environment and set up the simulation environment
- Specify analyses
- Specify which outputs to save

After you finish simulating, you can plot results.

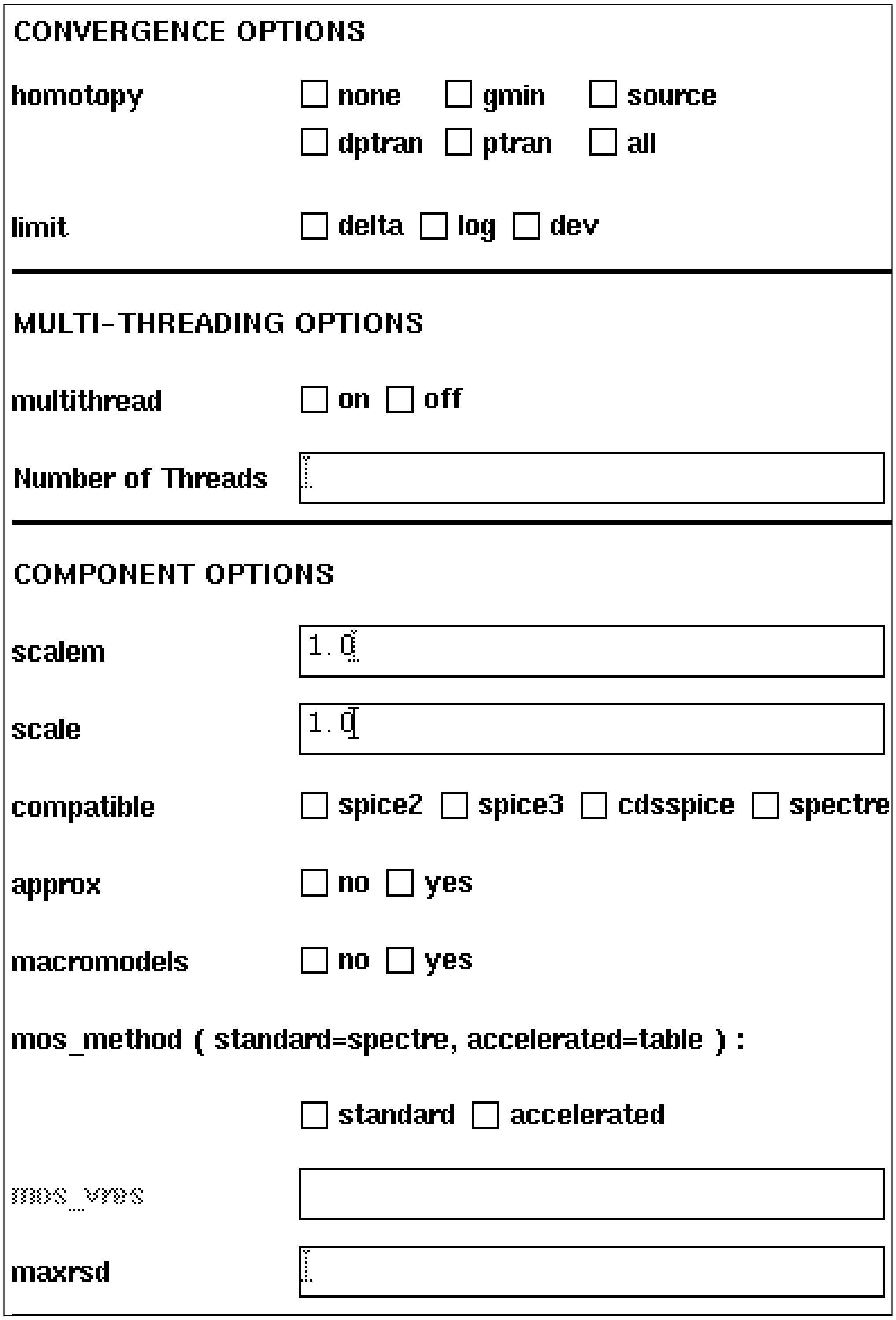
Setting Simulator Options

You set simulator-specific options and variables in two places:

- The *Simulation – Options – Analog* command lets you set simulator options and variables that apply to all analyses.
- For the spectre, spectreS, and some other simulator interfaces, the *Options* buttons in each Choosing Analyses form let you set options that apply to the specific analysis.

Each simulator has a different set of options.

Spectre Options



CONVERGENCE OPTIONS

homotopy □ none □ gmin □ source □ dptran □ ptran □ all

limit □ delta □ log □ dev

MULTI-THREADING OPTIONS

multithread □ on □ off

Number of Threads

COMPONENT OPTIONS

scalem 1.0

scale 1.0

compatible □ spice2 □ spice3 □ cdsspice □ spectre

approx □ no □ yes

macromodels □ no □ yes

mos_method (standard=spectre, accelerated=table) :

□ standard □ accelerated

mos_vres

maxrsd

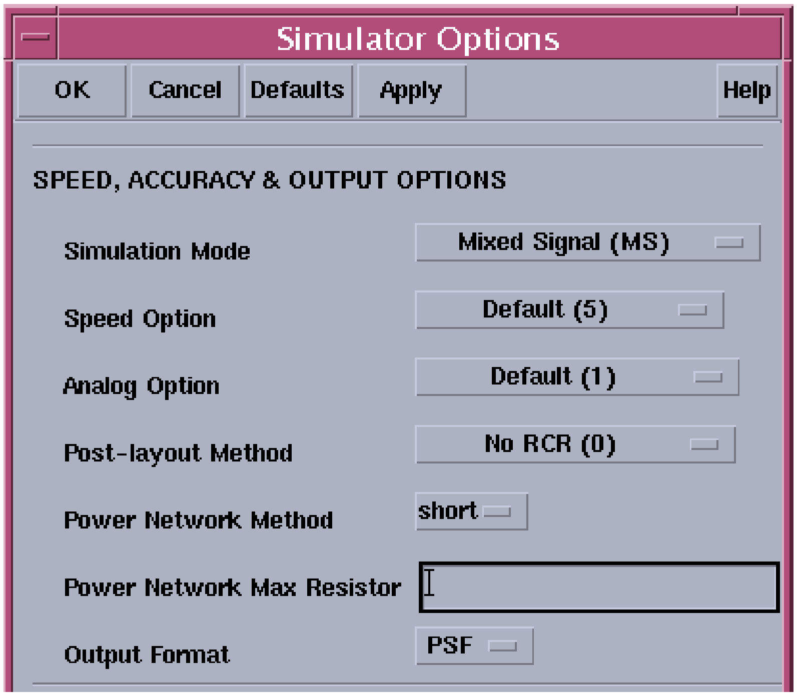
For help on Spectre options, refer to the *Immediate Set Options (options)* section in the *Analysis Statements* chapter of the *Spectre Circuit Simulator Reference* manual.

UltraSim Options

To set the Virtuoso UltraSim simulator options,

1. In the Simulation window, choose *Simulation – Options – Analog*.

   The Simulator Options form appears.



Simulator Options

SPEED, ACCURACY & OUTPUT OPTIONS

Simulation Mode	Mixed Signal (MS)
Speed Option	Default (5)
Analog Option	Default (1)
Post-layout Method	No RCR (0)
Power Network Method	short
Power Network Max Resistor	
Output Format	PSF

2. Set the simulator options as needed.

   For more details about the Virtuoso UltraSim simulator options, refer to Chapter 3, “Simulation Options” (*Virtuoso UltraSim Simulator User Guide*).

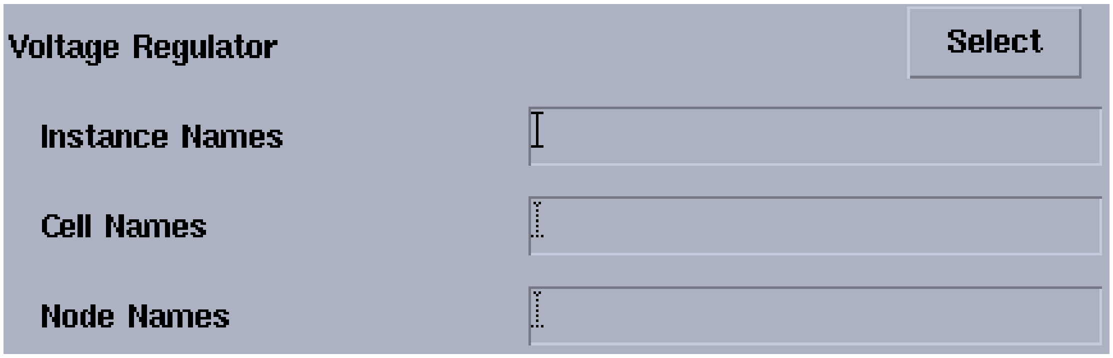
3. Click *OK*.

Setting Voltage Regulator Options

To set the voltage regulator options:

1. In the Simulation window, choose *Simulation – Options – Analog*.
2. The Simulator Options form appears, choose *OTHER SIMULATOR OPTIONS*.

The Ultrasim Other Simulator Options form appears. Set the voltage regulator options as required in the Voltage Regulator Section.



You can enter multiple Instance, Cell and Node names in the respective fields.

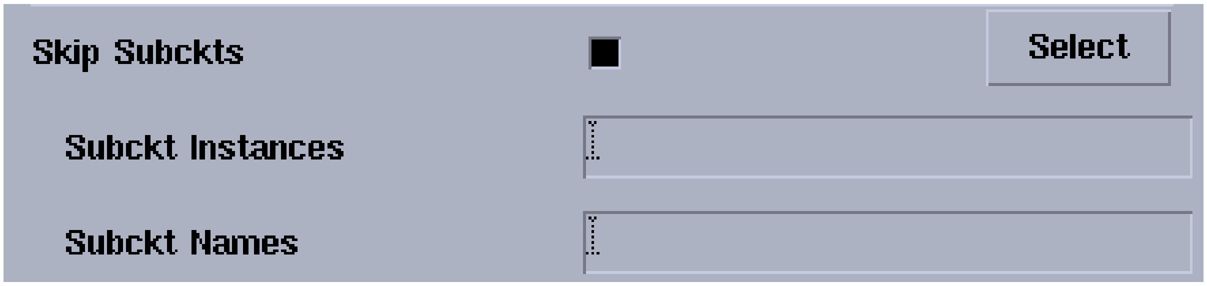
Note: You can directly set the voltage regulator options from the schematic, using the Select button.

Setting Skip Options in Ultrasim

To set the Skip options in Ultrasim:

1. In the Simulation window, choose *Simulation – Options – Analog*.
2. The Simulator Options form appears, choose *OTHER SIMULATOR OPTIONS*.

The Ultrasim Other Simulator Options form appears. Set the skip options as required in the Skip Subckts section.



You can enter multiple subckt instances and names in respective fields. You can also directly select instances from the schematic, using the Select button.

Setting Block-Based usim_opt Options

Schematic

The block-based *usim_opt on schematics* option lets you set speed, accuracy, and functionality of the Virtuoso UltraSim simulation for local subcircuit instances. Any Virtuoso UltraSim simulator options, including *usim_opt*, can be set in instance properties on the schematic. The most commonly used options are

- `sim_mode` (df/da/ms/a/s)
- `speed` (1-8)
- `analog` (1/2/3)

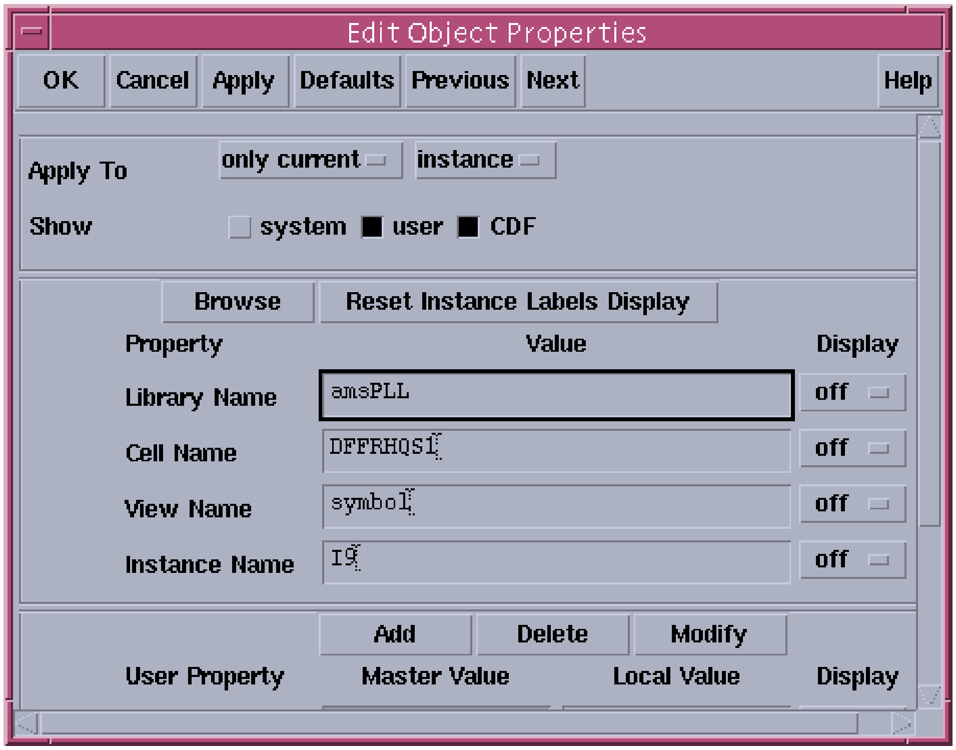
Note: The *usim_opt* option is valid only for block-level (subcircuit instances) settings.

For more details, refer to Chapter 3, “Simulation Options” (*Virtuoso® UltraSim Simulator User Guide)*.

To set block-based *usim_opt on schematic* options

1. Click on an instance in the schematic.
2. In the schematic window, choose *Edit – Properties – Objects*.

The Edit Object Properties form appears.



Apply To: only current, instance

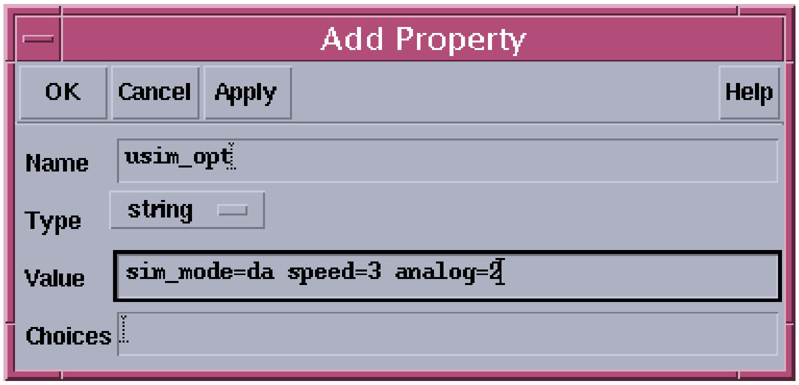
Show: □ system ■ user ■ CDF

Property	Value	Display
Library Name	amsPLL	off
Cell Name	DFFRHQS1	off
View Name	symbol	off
Instance Name	I9	off

User Property	Master Value	Local Value	Display

3. Click *Add*.

   The Add Property form appears.



Name: usim_opt

Type: string

Value: sim_mode=da speed=3 analog=2

Choices:

4. In the *Name* field, type `usim_opt`.
5. Type `sim_mode=da speed=3 analog=2` into the *Value* field.
6. Click *OK* to save the settings and close the Add Property form.
7. In the Edit Object Properties form, click *OK*.

8. In the schematic, choose *Design – Check and Save*.
9. In the Cadence® Analog Design Environment simulation window, choose *Simulation – Options – Analog*.

   The Simulator Options form appears.
10. Turn on *Allow usim_opt on schematics*.
11. Click *OK* to save the settings and close the Simulator Options form.
12. Run netlisting or the simulation from the Simulation window.

    The *usim_opt* settings are set locally for the instance block in the netlist.

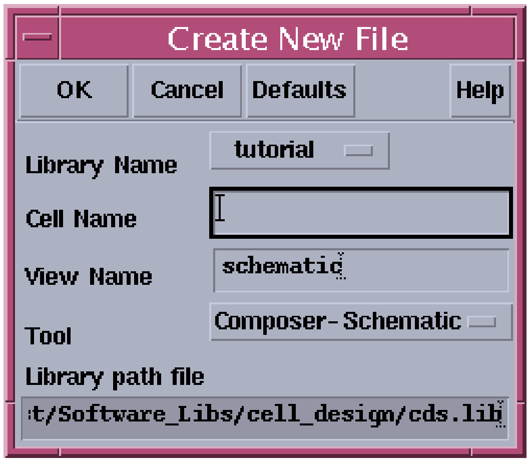
    Note: Virtuoso Spectre and HSPICE netlist formats are supported in IC 5.0 and later releases.

Hierarchy Editor

You can also set block level *usim_opt* options, such as speed and accuracy, using the Cadence® Hierarchy Editor (HED). For more information about setting Virtuoso UltraSim simulator options, refer to Chapter 3, “Simulation Options” (*Virtuoso® UltraSim Simulator User Guide)*.

1. From the CIW, choose *File – New – Cellview*.

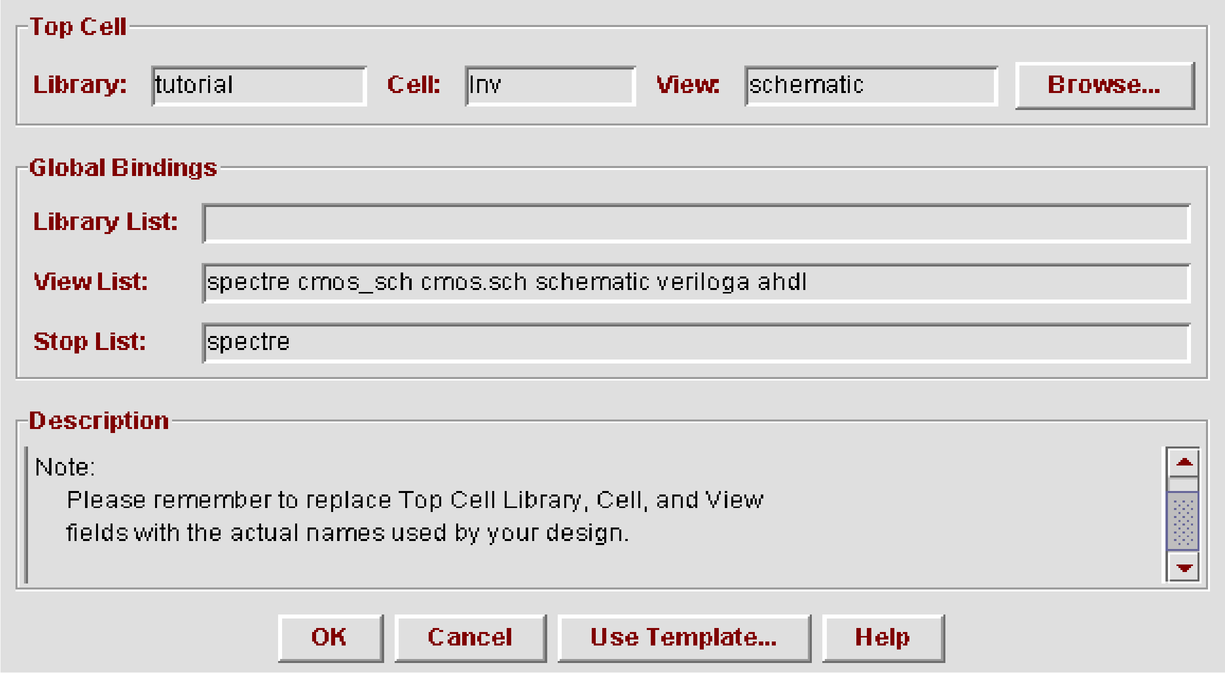
   The Create New File form appears.



Library Name	tutorial
Cell Name	
View Name	schematic
Tool	Composer-Schematic
Library path file	:t/Software_Libs/cell_design/cds.lib

2. Choose a library, cell, and view.
3. Choose *Hierarchy-Editor* from the *Tool* drop-down list box.
4. Click *OK*.

The New Configuration form appears.



Top Cell

Library: tutorial
Cell: Inv
View: schematic

Global Bindings

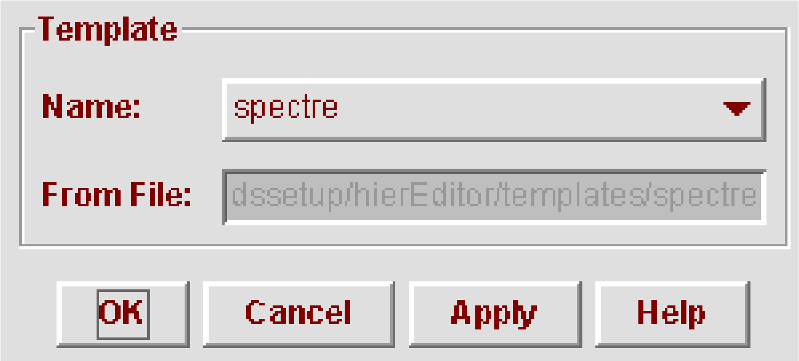
Library List:
View List: spectre cmos_sch cmos.sch schematic veriloga ahdl
Stop List: spectre

Description

Note:
Please remember to replace Top Cell Library, Cell, and View
fields with the actual names used by your design.

5. Click on the *Use Template* button located at the bottom of the form.

The Use Template form appears.



Template

Name: spectre
From File: dssetup/hierEditor/templates/spectre

6. Choose *spectre* from the *Name* drop-down list box.

7. Click *OK*.

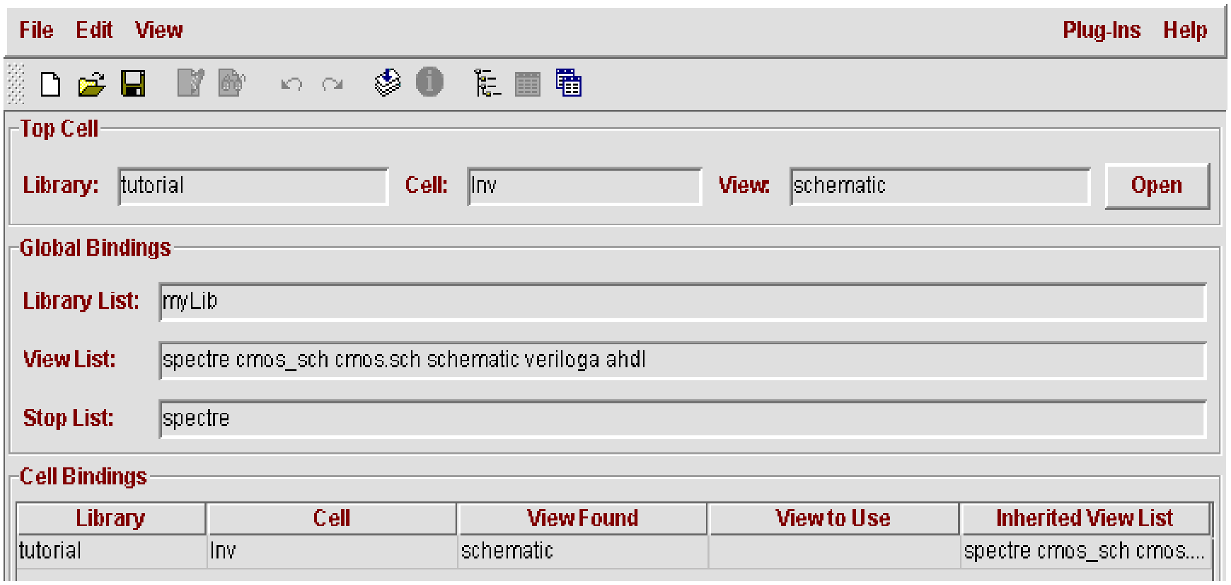
The New Configuration form redisplays with default data for the *Top Cell* and *Global Bindings* sections.

8. In the *Top Cell* section, enter the desired library, cell name, and schematic view.

9. In the *Global Bindings* section, remove `myLib` from the *Library List* field.

10. Click *OK*.

The Cadence hierarchy editor form displays your data.



Top Cell

Library: tutorial
Cell: Inv
View: schematic

Global Bindings

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

Cell Bindings

Library	Cell	View Found	View to Use	Inherited View List
tutorial	Inv	schematic		spectre cmos_sch cmos....

Note: The hierarchy editor form configures the design by using a default *View List* and *Stop List* in the *Global Bindings* section. You need to modify these lists for your design.

11. Choose *View – Properties*.

The *sim_mode* and *speed* columns appear in the *Cell Bindings* section.

12. Choose *Edit – Add Property Column* to add additional cell- or occurrence-based properties to the *Cell Bindings* section.

Note: Instance-based properties are not supported by the Virtuoso UltraSim simulator in ADE.

13. Set options for cell- or occurrence-based properties.

To set options for cell-based properties,

a. Right-click in a property column.

A popup menu appears.

b. Choose *Set “property name” Cell Property* to set the property.

To set options for occurrence-based properties,

a. Choose *View – Tree*.

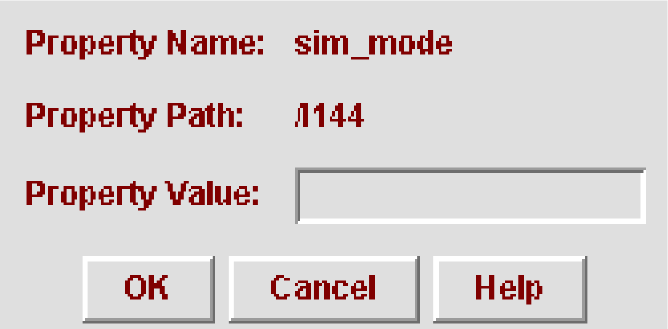
b. Select an instance and right-click on a property column in the *Cell Bindings* section.

A popup menu appears.

Note: Do not left-click on a property column to set a property (left-click creates an instance-based property, which is not supported by the Virtuoso UltraSim simulator in ADE).

c. Choose *Set "name" Occurrence Property*.

The Set Occurrence Property form appears.



Property Name: sim_mode

Property Path: /I144

Property Value:

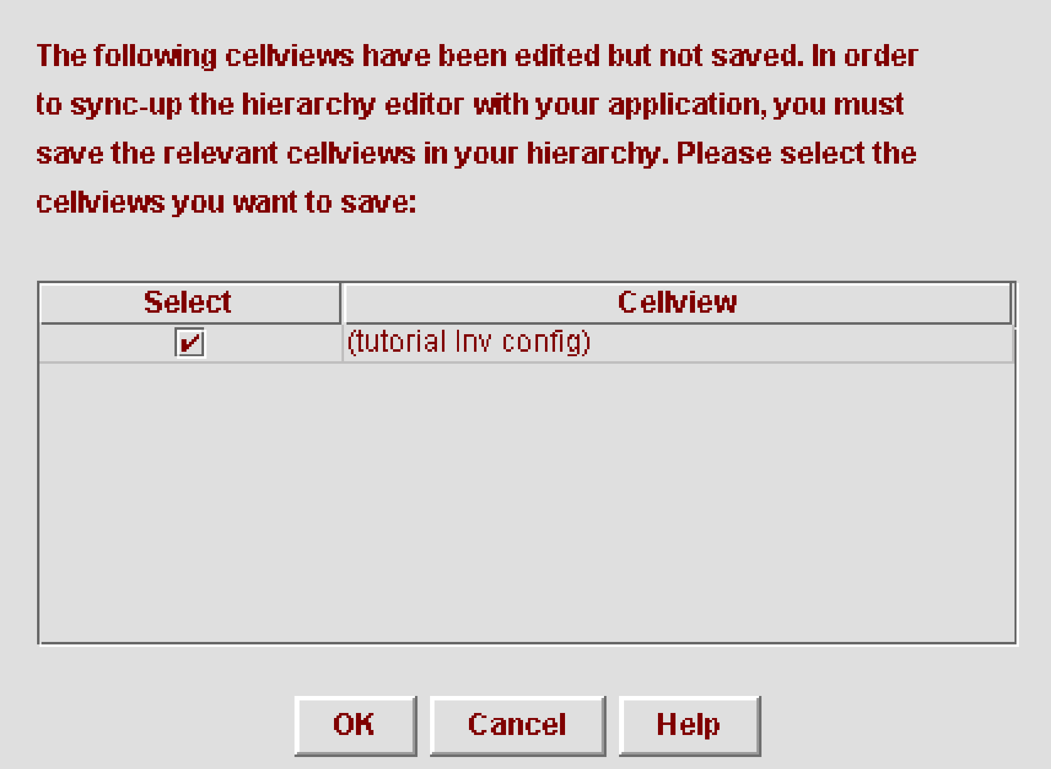
d. Choose the appropriate property value.

e. Click *OK*.

The instance is marked with an o symbol.

f. Choose *View – Update*.

The Update Sync-up form appears.



The following cellviews have been edited but not saved. In order to sync-up the hierarchy editor with your application, you must save the relevant cellviews in your hierarchy. Please select the cellviews you want to save:

Select	Cellview
☑	(tutorial Inv config)

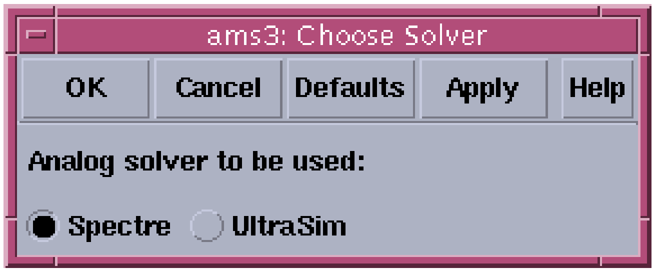
g. Click *OK* to save the cellview.

AMS Options

When you select ams as your simulator, the Simulation menu offers you an option to select a solver for the simulation. After doing that, you can set Virtuoso® AMS simulator options by choosing the appropriate option from the *Simulation – Options* submenu.

Choosing a Solver

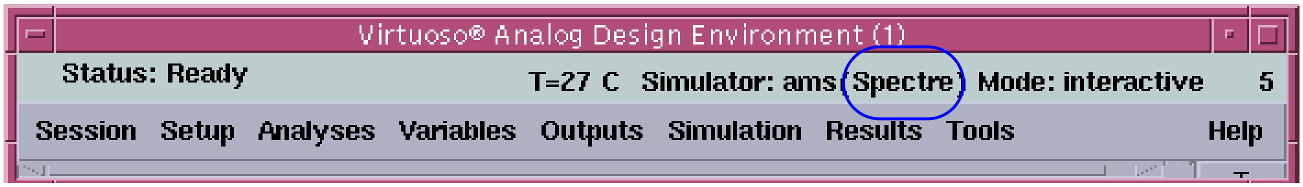
Choose *Simulation – Solver* to bring up the Choose Solver form, in which you can select either *Spectre* or *UltraSim* as the solver.



Analog solver to be used:

● Spectre ○ UltraSim

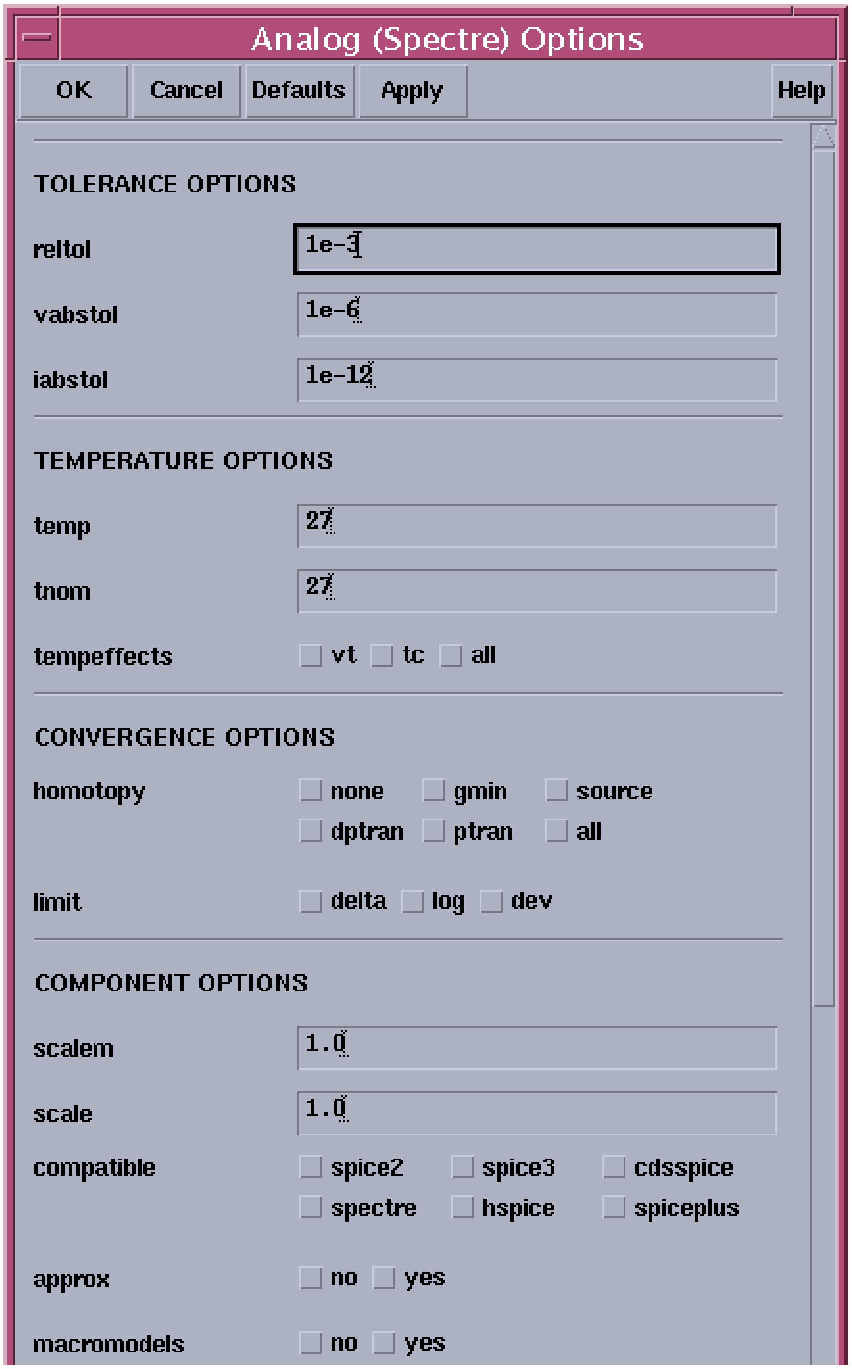
Your choice appears next to the name of the selected simulator below the title bar as highlighted in the snapshot below.



Status: Ready T=27 C Simulator: ams(Spectre) Mode: interactive 5

Analog (Spectre)

Choose *Simulation – Options – Analog (Spectre).*



TOLERANCE OPTIONS

reltol 1e-3

vabstol 1e-6

iabstol 1e-12

TEMPERATURE OPTIONS

temp 27

tnom 27

tempeffects □ vt □ tc □ all

CONVERGENCE OPTIONS

homotopy □ none □ gmin □ source □ dptran □ ptran □ all

limit □ delta □ log □ dev

COMPONENT OPTIONS

scalem 1.0

scale 1.0

compatible □ spice2 □ spice3 □ cdsspice □ spectre □ hspice □ spiceplus

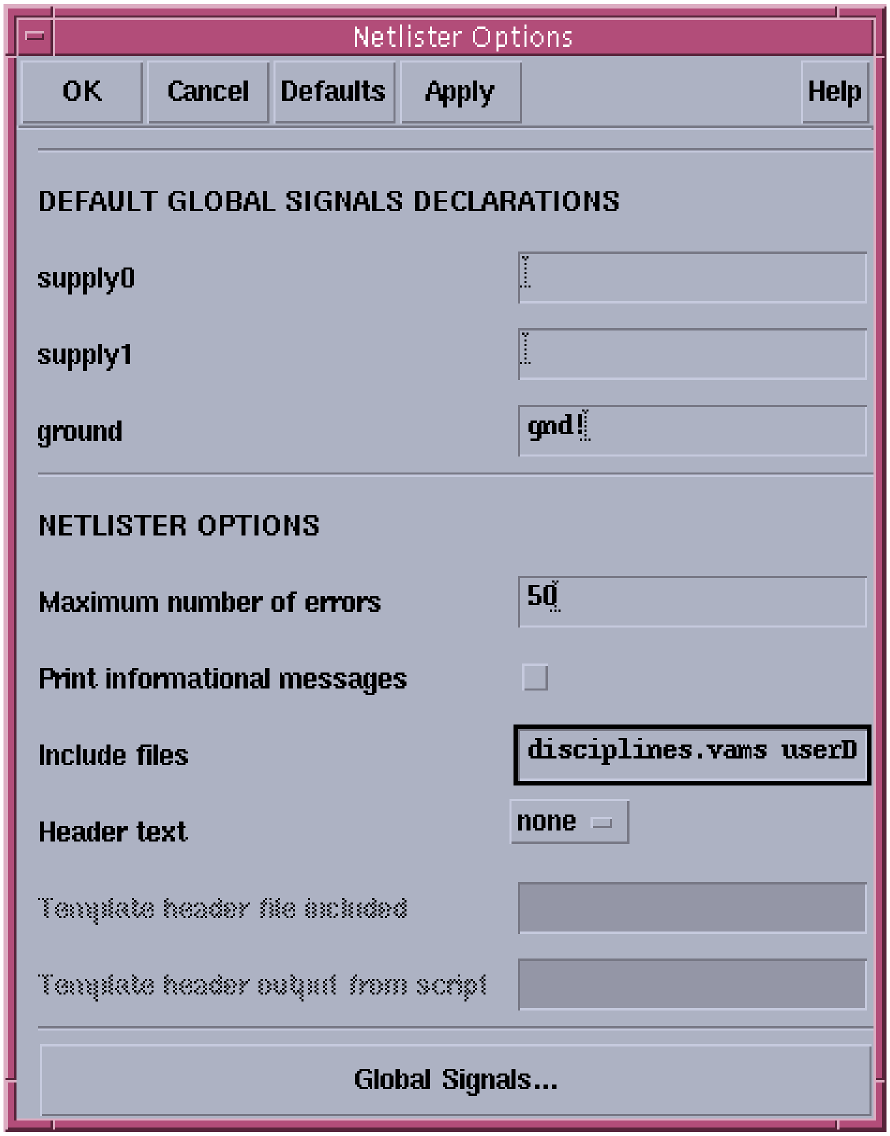
approx □ no □ yes

macromodels □ no □ yes

For details refer to Chapter 7 of the *Spectre Circuit Simulator User Guide*.

Netlister

Choose *Simulation – Options – Netlister.* For details refer to the *Virtuoso® AMS Environment User Guide.*



DEFAULT GLOBAL SIGNALS DECLARATIONS

supply0
supply1
ground: gnd!

NETLISTER OPTIONS

Maximum number of errors: 50
Print informational messages: □
Include files: disciplines.vams userD
Header text: none
Template header file included
Template header output from script

The **Default Global Signals Declarations** field allows you to specify default global signals in the corresponding fields.

The **Maximum number of errors field** allows you to set the maximum number of errors the netlister can encounter before it stops processing the design.

The **Print informational messages** requests more numerous and extensive messages.

The **Global Signals** button brings up the Global Signals form, which is described in the next topic.

For detailed description of the Netlister, see the *Netlisting* chapter of the *Virtuoso® AMS Environment User Guide*

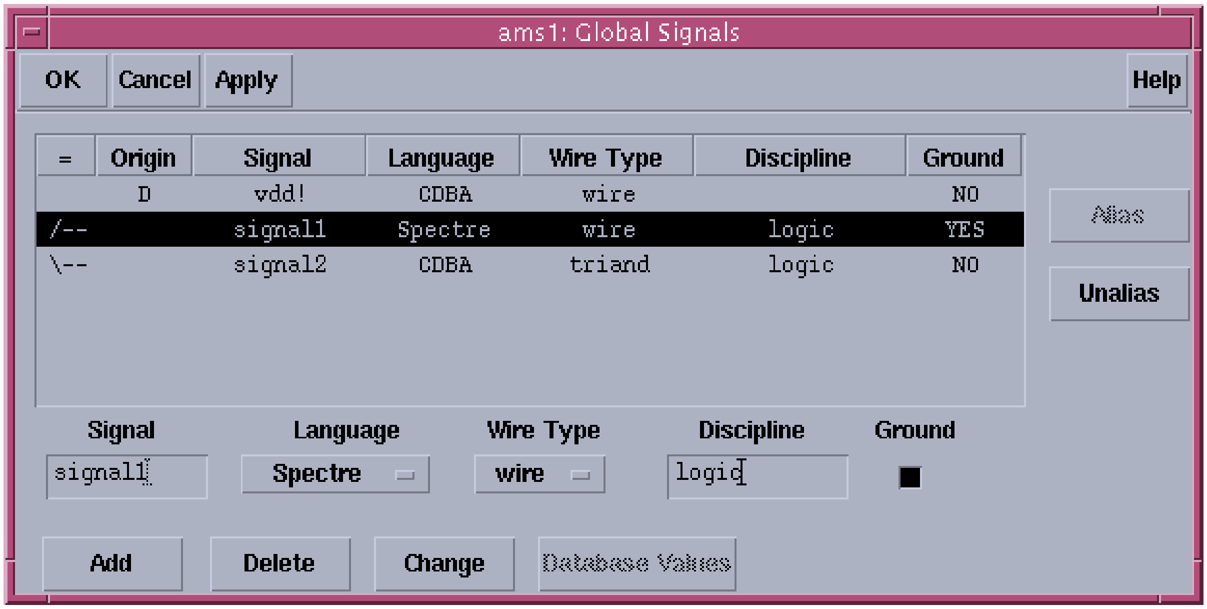
Working with Global Signals in AMS

A global signal is a signal that is connected by name across all levels of a design hierarchy without using pins. Global signals can come from schematic data or from text modules. AMS is aware of only global signals that come from schematic data. You can use the Global Signals form to declare a signal that is used as an out-of-module signal reference.

To add a global signal,

1. In the Netlister Options form, click the *Global Signals* button.

   The Global Signals form appears. If the design had not been netlisted after recent changes, you are prompted to netlist the design so that the Global Signals form can display the latest data.



=	Origin	Signal	Language	Wire Type	Discipline	Ground
	D	vdd!	CDBA	wire		NO
/--		signal1	Spectre	wire	logic	YES
\--		signal2	CDBA	triand	logic	NO

Signal	Language	Wire Type	Discipline	Ground
signal1	Spectre	wire	logic	■

2. Global signals are displayed in a tabular format. It includes the following information.

   - The first column indicates the alias of aliased signals. They represent:
     - The beginning of the aliased set: `/--`
     - The aliased signals in between: `|--`

    - The end of the aliased set: `\--`

- The *Origin* column is either blank if the signal was added using the Global Signals form or it has the value `D` to indicate that a signal has been extracted from the design.
- The *Signal* column shows the name of the signal.
- The *Language* column displays the language in which the signal was created.
- The *Wire Type* column shows the wire type of the signal.
- The *Discipline* column shows the discipline of the signal.
- The *Ground* column indicates if the global signal is used as a ground reference.

Note: Netlisting extracts the signals in the design and merges them with the signals created using the Global Signals form. If you open the form without netlisting, it would be empty.

3. The input fields below the report get populated by signal details when you select any signal. To change these values, you can either type over them or select values from the cyclic lists and then click the *Change* button.
4. To add new global signals,

   a. Type a unique name for the signal in the *Signal* field. You can specify a range such as <5:8> by post-fixing it to the name.

   b. Select the language as *CDBA*, *Spectre*, *Spice* or *Verilog-AMS* from the *Language* cyclic list.

   c. Select one of these from the *Wire Type* cyclic list: *wire, supply0, supply1, tri, tri0, tri1, triand, trior, trireg, wand, wor,* or *wreal.*

   Note: If the signal name you specify is included in the *supply0* or *supply1* field of the Netlister Options form, the default value for wire type would be changed to *supply0* and *supply1* accordingly.

   d. Type a discipline name for the signal in the *Discipline* field.

   e. If you want to use the global signal as a ground reference, select the *Ground* option. You can select this option only for signals that have the wire type *wire* or *tri*.

   Note: If the signal name you specify is included in the *ground* field of the Netlister Options form, this field appears selected by default.

   f. Click the *Add* button.

The new global signal appears in the list of global signals.

Note: You can create a new global signal from an existing one by selecting one, modifying its values and clicking *Add*.

5. To delete one or more signals, select them and click the *Delete* button.

   You cannot delete a global signal that is extracted from the design. If you select such a signal, the Language and Name fields appear non-editable. If you change any of the other values, the *Database Values* button is enabled, using which you can set the fields back to their original values from the database.

6. You can alias global signals into groups. Aliased signals in a group are electrically equivalent, as if they are joined by a wire. To alias global signals, select the signals to be aliased and click the Alias button.

   To select signals listed consecutively, hold down the `shift` key while you click the signal names to be aliased. To select signals that are not listed sequentially, hold down the `control` key while you click on the signal names.

   When you alias signals, they redisplay consecutively in the global signal list, joined by a vertical connecting bar. If you alias signals belonging to separate aliased signal groups, all of the signals in the groups are aliased.

7. To unalias signals, select the signals to be unaliased from the group, and click the *Unalias* button. If an alias set has only two signals, and you unalias one, the other also gets automatically unaliased.

8. When you have finished editing the list of global signals, click *OK*.

You need to regenerate the netlist so that the changes made in this form reflect in the netlist and cds_globals module is regenerated. If you try to create or re-create the netlist without applying the changes in the Global Signals form, your changes get overwritten by the netlist. This is the reason a prompt appears as follows:

```
While netlisting, the globals would again be extracted from the design and the
globals form would be updated. Unsaved changes, if any on the globals form would
be lost. Proceed with netlisting?
```

Netlisting includes the following actions:

1. It extracts information about all the global signals and design variables from the design.

2. Information about variables is merged with the design and written as it is to the `verilog.vams` file.

3. The global signals information is updated with any new extracted global signals that you modified in the current session.

4. If an extracted global signals exists in the global signals information and its origin is marked as D, and it has not been modified in the design, it is not copied.
5. If an extracted global signals exists in the global signals information and its origin is marked as D, and you modified it using the Global Signals form in the current session, the values are copied over and a message appears saying so.
6. If it exists in the global signals information for the session and the Origin is not marked as ‘D’, a prompt appears as follows:

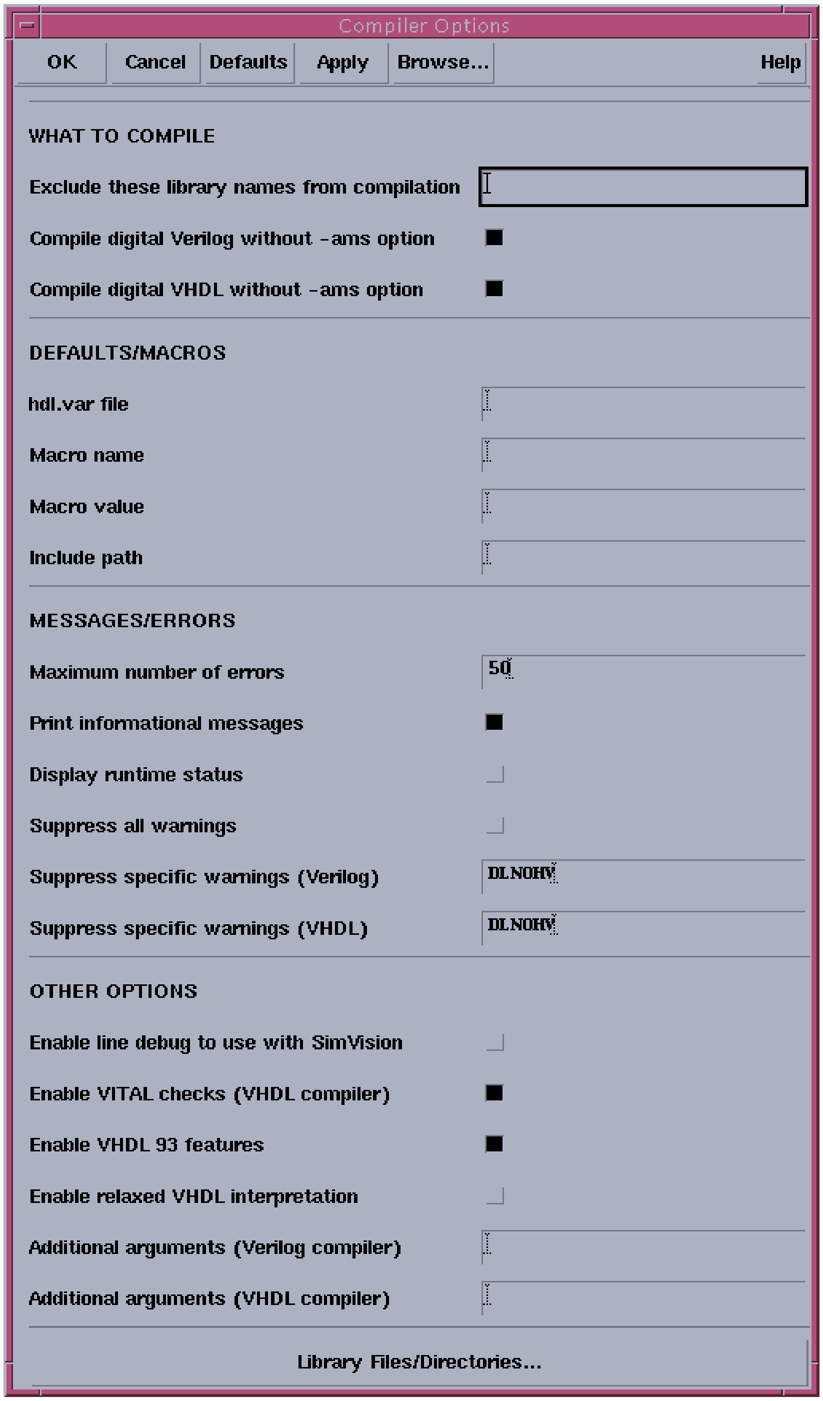
   ```
   A global with the name ‘%s’ already exists in the session and is now also found
   in the design. Using the one that exists in the session.
   ```

The Global Signals form appears updated the next time it is opened.

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Global Signals* option in the Saving State form and save the current ADE state. You can later load the state using the Loading State form with the *Global Signals* option selected. For more information, see Saving and Restoring the Simulation Setup on page 77.

Compiler

Choose *Simulation – Options – Compiler.*



WHAT TO COMPILE

Exclude these library names from compilation

Compile digital Verilog without -ams option ■

Compile digital VHDL without -ams option ■

DEFAULTS/MACROS

hdl.var file

Macro name

Macro value

Include path

MESSAGES/ERRORS

Maximum number of errors 50

Print informational messages ■

Display runtime status □

Suppress all warnings □

Suppress specific warnings (Verilog) DLNOHV

Suppress specific warnings (VHDL) DLNOHV

OTHER OPTIONS

Enable line debug to use with SimVision □

Enable VITAL checks (VHDL compiler) ■

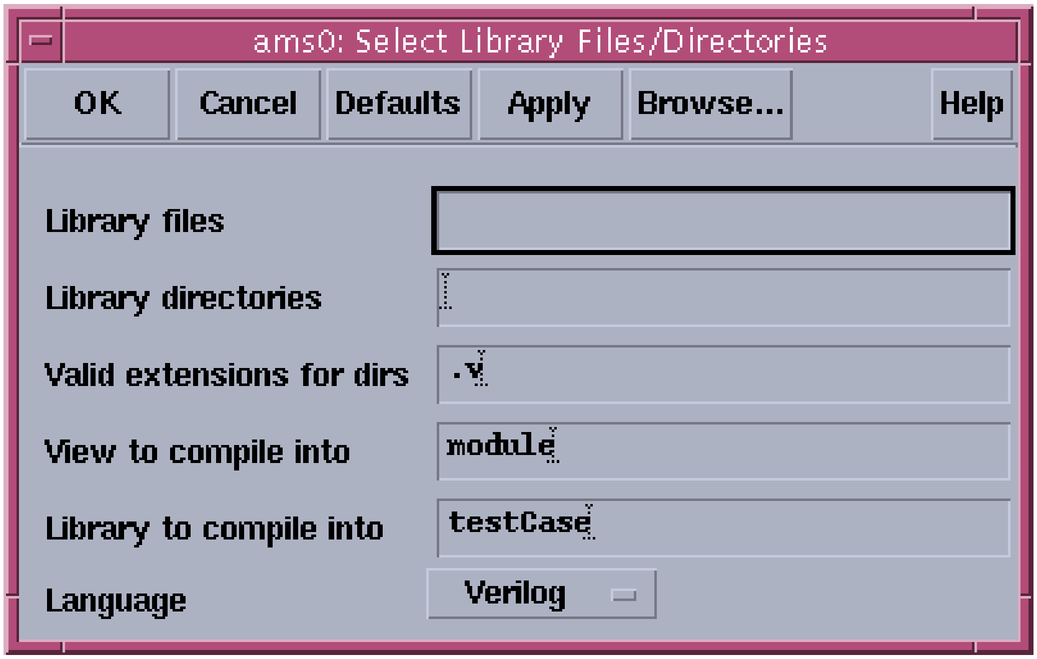
Enable VHDL 93 features ■

Enable relaxed VHDL interpretation □

Additional arguments (Verilog compiler)

Additional arguments (VHDL compiler)

1. Use the *WHAT TO COMPILE* options to specify the library names that can be excluded from compilation and if digital Verilog and VHDL should be compiled without the -ams option.
2. Use the *DEFAULTS/MACROS* options to specify an `hdl.var` file and the libraries to be excluded from compilation.
3. Use the *MESSAGES/ERRORS* options to specify the error and display options.
4. Use the *OTHER OPTIONS* to enable VHDL/Verilog options and to specify other arguments.
5. The *Library Files/Directories* button brings up the Select Libary Files/Directories form using which you can specify files that you want compiled.



ams0: Select Library Files/Directories

Library files	
Library directories	
Valid extensions for dirs	.v
View to compile into	module
Library to compile into	testCase
Language	Verilog

You can specify the following information in this form.

- **Library files**: Paths of library files, separated by spaces. The extensions of these files must be one of these: `.v`, `.va`, `.vams`, `.vhd`, `.vhms`. The paths are relative to the netlist directory.
- **Library directories**: Paths of directories, separated by spaces. Files in the specified directories are considered if they have these extensions: `.v`, `.va`, `.vams`, `.vhd`, `.vhms`.
- **Valid extensions for dirs**: Valid extensions for library directories, separated by spaces.

- ❑ **View to compile into**: The view in which you want the files to be compiled. The default view is `module`.
- ❑ **Library to compile into**: The library into which you want the files to be compiled. This library should exist in `cds.lib`. The default is the design library.

  Note: Please ensure that for ncelab to pick the compiled files, the library in which the files get compiled is specified in Library List Global Bindings in the Hierarchy Editor.
- ❑ **Language**: The language you want to complie the files into. You can select any of these: *Verilog*, *VerilogA*, *VerilogAMS*, *VHDL* or *VHDLAMS*.

You can use the *Browse* button to populate these fields. This button brings up the UNIX browser for the first two fields. For the last field, it brings up the Library Browser.

These settings can also be collectively saved by using the Library Files check box in the Saving State form.

6. Click *OK*.

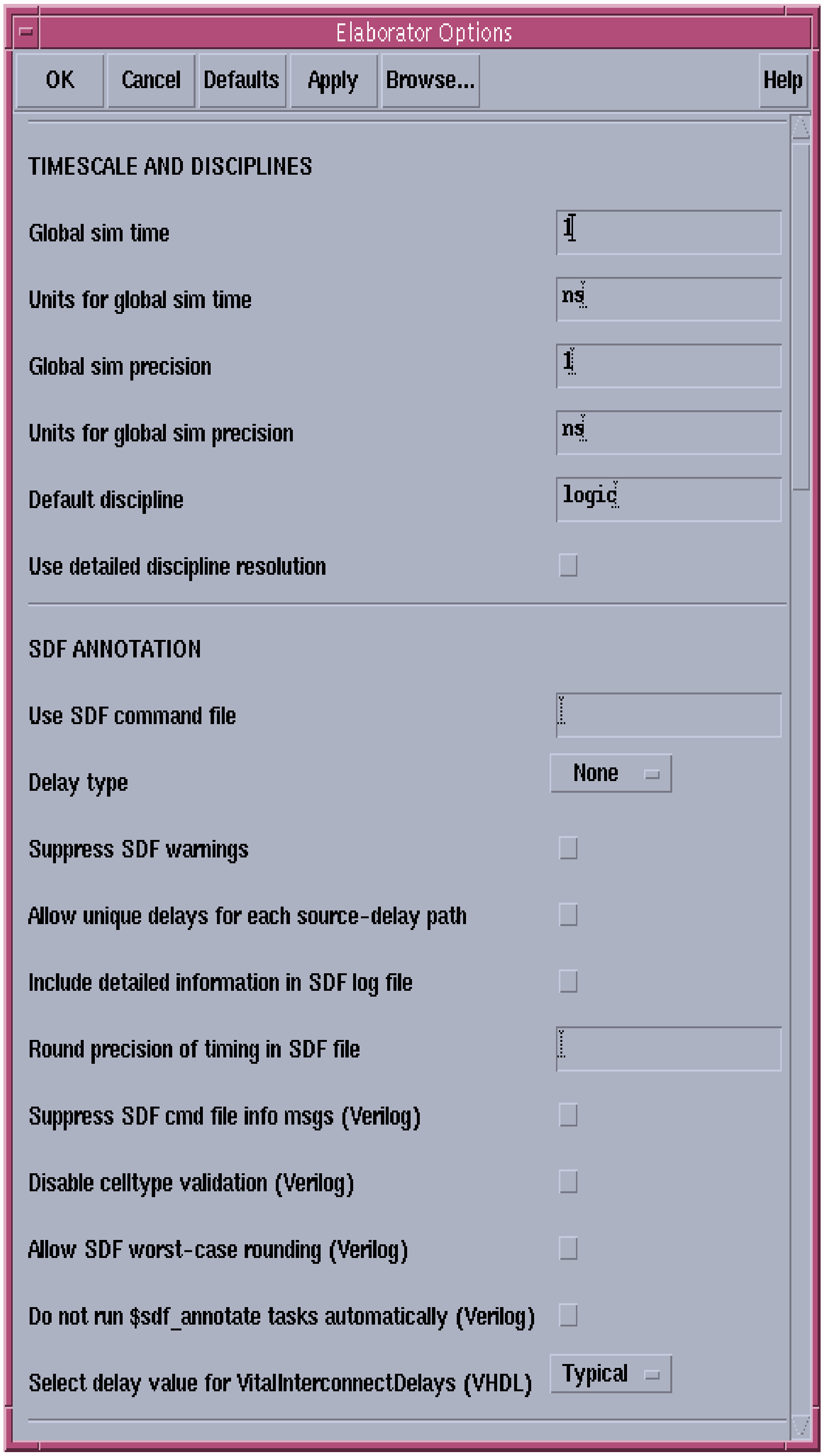
   When the simulation is run, the specified files are compiled into the implicit library.

Note: Some of the fields in the Compiler form will be hidden if you chhose the run mode as ncverilog.

For details, refer to the *Virtuoso® AMS Environment User Guide*

Elaborator

Choose *Simulation – Options – Elaborator.*



TIMESCALE AND DISCIPLINES

Global sim time: 1

Units for global sim time: ns

Global sim precision: 1

Units for global sim precision: ns

Default discipline: logic

Use detailed discipline resolution □

SDF ANNOTATION

Use SDF command file:

Delay type: None

Suppress SDF warnings □

Allow unique delays for each source-delay path □

Include detailed information in SDF log file □

Round precision of timing in SDF file:

Suppress SDF cmd file info msgs (Verilog) □

Disable celltype validation (Verilog) □

Allow SDF worst-case rounding (Verilog) □

Do not run $sdf_annotate tasks automatically (Verilog) □

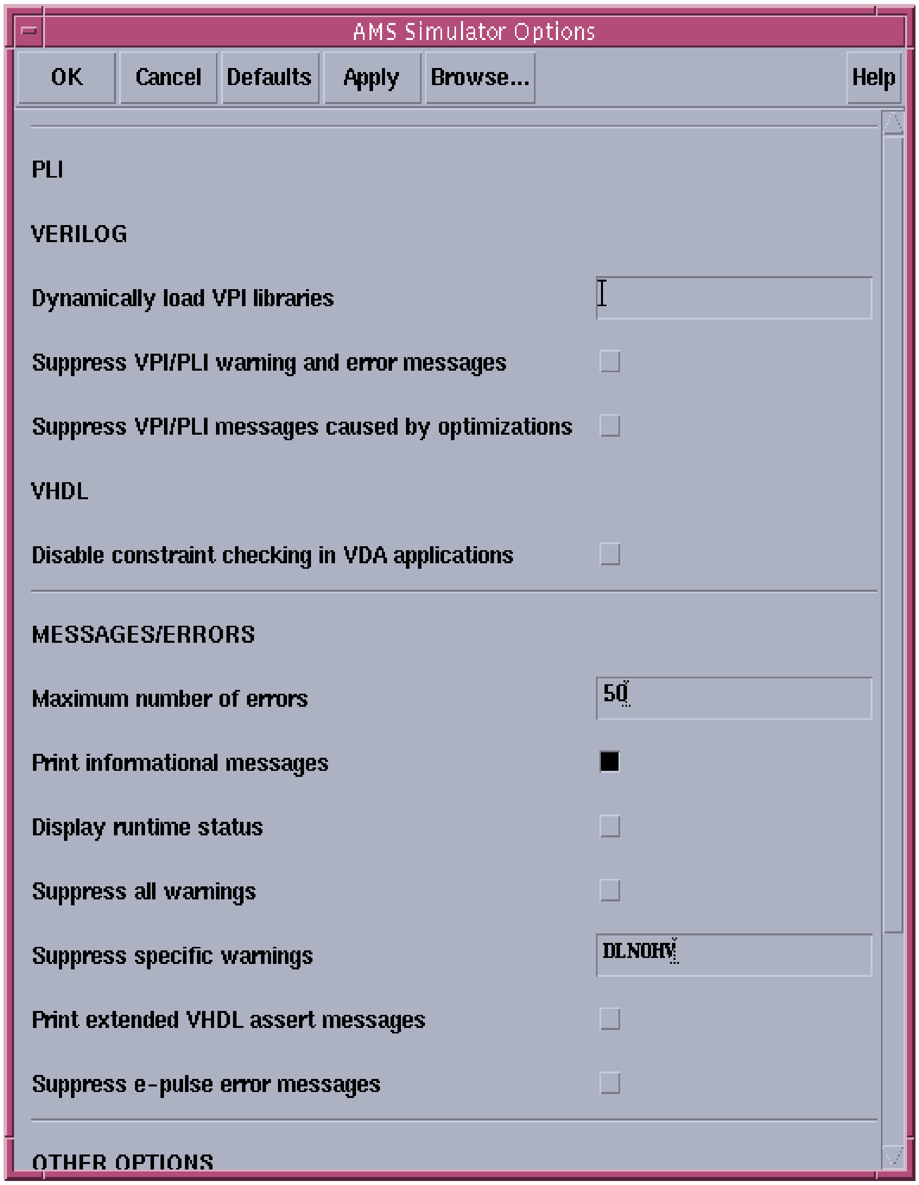
Select delay value for VitalInterconnectDelays (VHDL): Typical

This form enables you to set the following options: *Timescale and Disciplines*, *SDF Annotation, Timing Options, PLI, Access, Messages/Errors, Other Options*. For details refer to the *Virtuoso® AMS Environment User Guide* (Specifying the Behavior of the Elaborator, Simulator, and Waveform Viewer section).

Note: Some of the fields in the Elaborator form will be hidden if you chhose the run mode as ncverilog.

AMS Simulator

Choose *Simulation – Options – AMS Simulator.*



PLI

VERILOG

Dynamically load VPI libraries

Suppress VPI/PLI warning and error messages □

Suppress VPI/PLI messages caused by optimizations □

VHDL

Disable constraint checking in VDA applications □

MESSAGES/ERRORS

Maximum number of errors 50

Print informational messages ■

Display runtime status □

Suppress all warnings □

Suppress specific warnings DLNOHV

Print extended VHDL assert messages □

Suppress e-pulse error messages □

OTHER OPTIONS

In the Verilog and VHDL sections specify the following options:

Dynamically load VPI libraries: Dynamically loads the specified VPI application.

Suppress VPI/PLI warning and error messages: Disables printing of PLI warning and error messages.

Suppress VPI/PLI messages caused by optimizations: Prints a warning message only the first time that a PLI read, write, or connectivity access violation is detected.

Disable constraint checking in VDA applications: Disables constraint checking in VHDL Design Access (VDA) functions, for increased performance.

In the Messages/Errors section specify the following options:

Maximum number of errors: Stops compilation if the number of errors reaches the specified maximum limit.

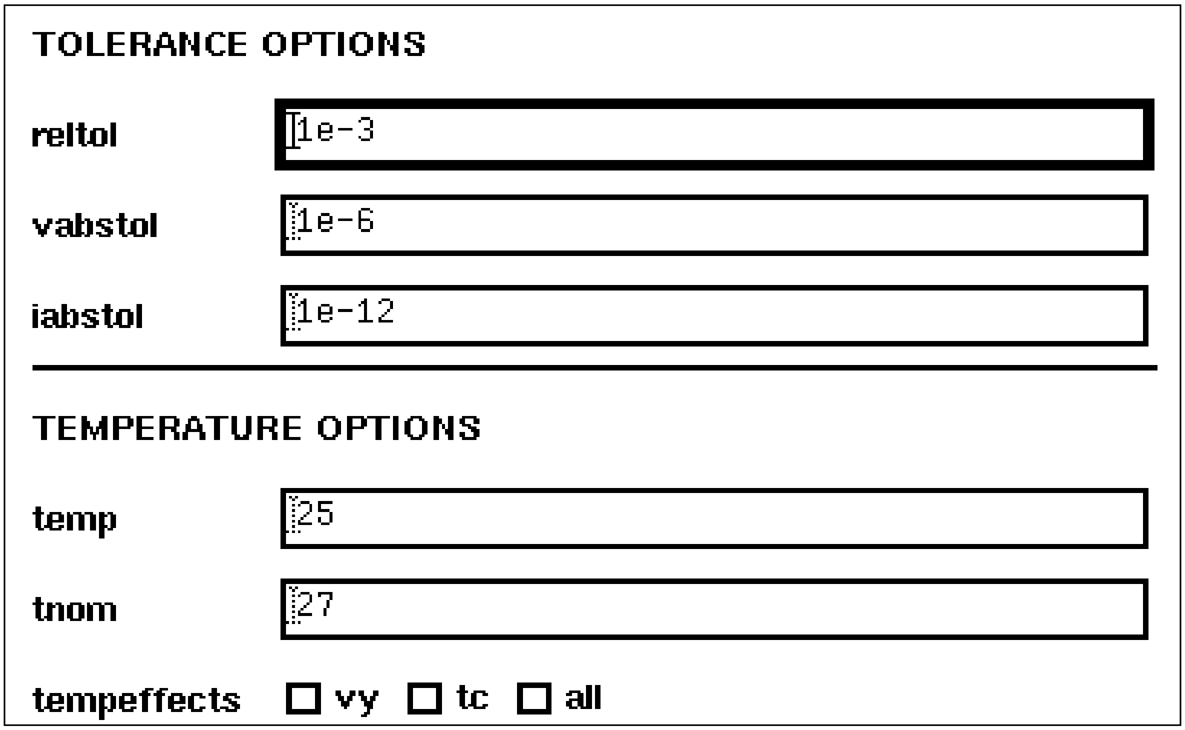
In the Other Options section, you can also specify if you want to print information or VHDL assert messages, display runtime status, generate a runtime profile or allow profiling of threaded processes by enabling the corresponding buttons.

Note: Some of the fields in the Simulator Options form will be hidden if you chhose the run mode as ncverilog.

For details, refer to the *Virtuoso® AMS Environment User Guide.*

SpectreS Options

For the spectreS simulator, the *Simulator – Options – Analog* command displays a scrolling list of simulator options. Also, each Choosing Analyses form has an *Options* button to let you set options specific to that analysis.



TOLERANCE OPTIONS

reltol	1e-3
vabstol	1e-6
iabstol	1e-12

TEMPERATURE OPTIONS

temp	25
tnom	27
tempeffects	□ vy □ tc □ all

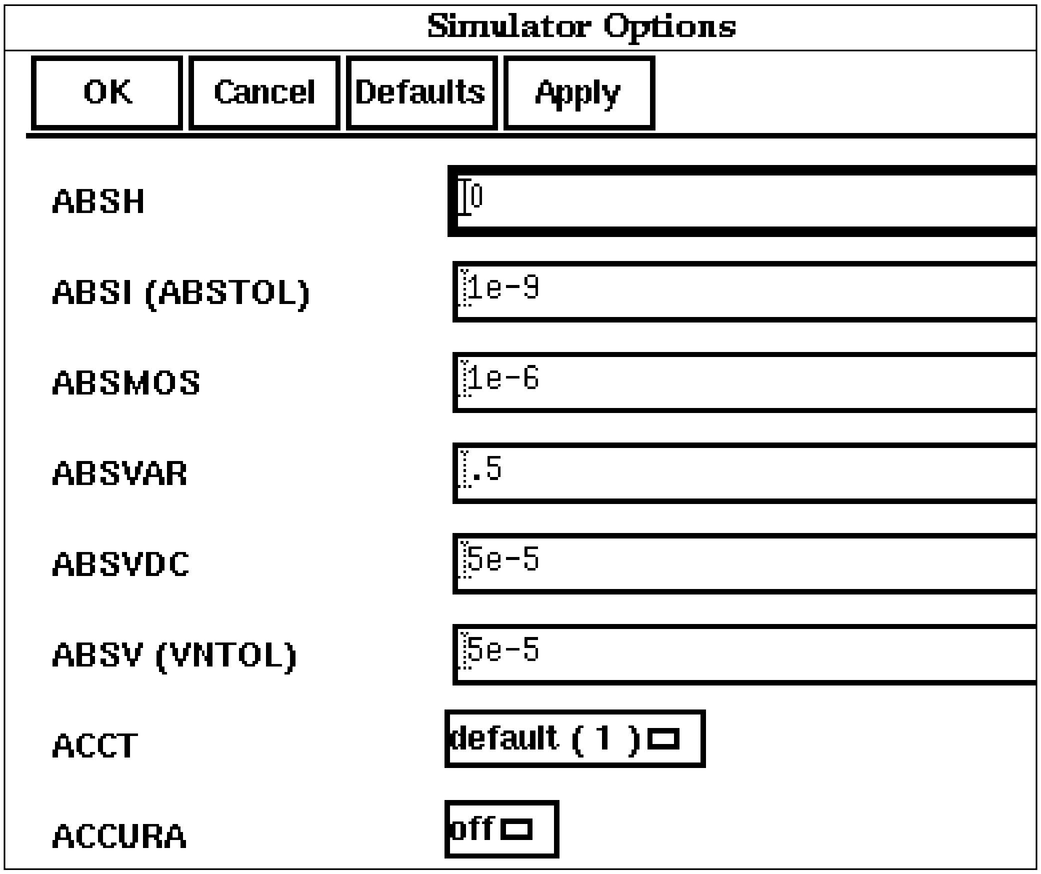
For help on spectreS options, refer to the *Spectre Circuit Simulator Reference* manual.

HSPICE Socket Options

To set the simulator options for the HSPICE socket simulator,

1. Choose *Simulation – Options – Analog*.

The HSPICE Simulator Options form appears.



ABSH	0
ABSI (ABSTOL)	1e-9
ABSMOS	1e-6
ABSVAR	.5
ABSVDC	5e-5
ABSV (VNTOL)	5e-5
ACCT	default (1)
ACCURA	off

2. Check that the following options are set to provide the analog circuit design environment compatibility:

   ```
   ARTIST=2
   INGOLD=2
   PSF=2
   ```

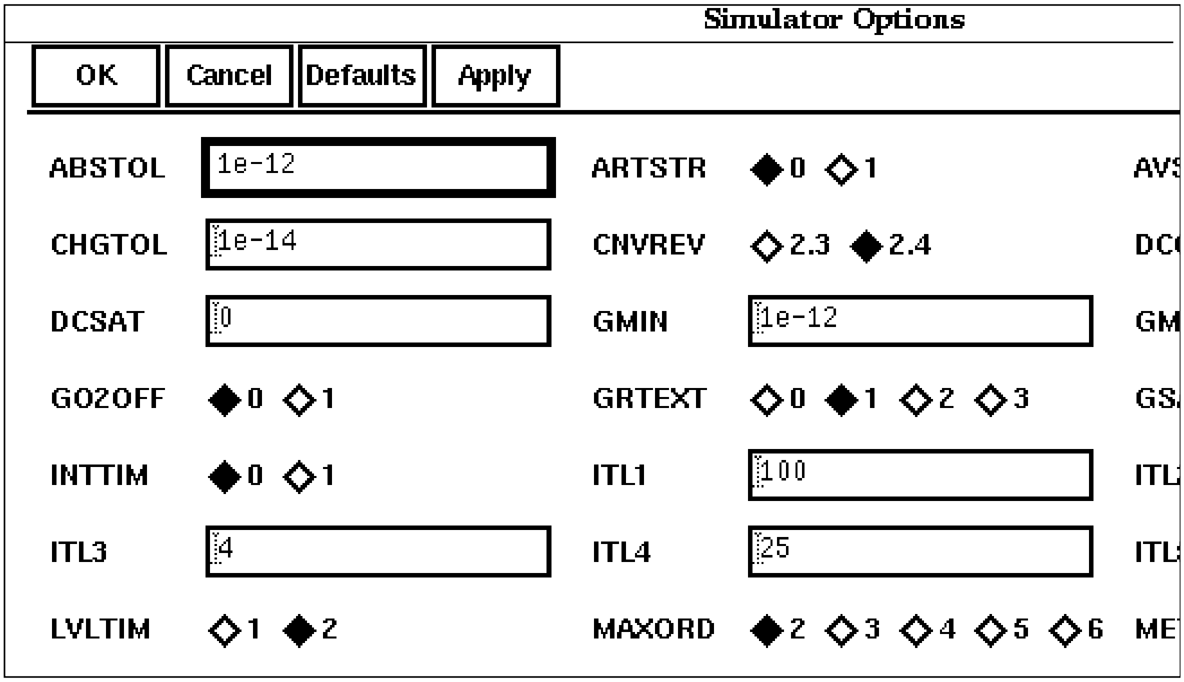
   If you run the HSPICE simulator in standalone mode, rather than under the analog circuit design environment, you might want to change these values.

3. Set `UIC=1` to include the `UIC` entry in the `.TRAN` command.

For help setting other options, refer to your HSPICE documentation.

Cadence SPICE Options

For the Cadence[®] SPICE simulator, the *Simulator – Options – Analog* command displays a scrolling list of simulator options.



ABSTOL	1e-12	ARTSTR	◆0 ◇1	AV
CHGTOL	1e-14	CNVREV	◇2.3 ◆2.4	DC
DCSAT	0	GMIN	1e-12	GM
GO2OFF	◆0 ◇1	GRTEXT	◇0 ◆1 ◇2 ◇3	GS
INTTIM	◆0 ◇1	ITL1	100	ITL
ITL3	4	ITL4	25	ITL
LVLTIM	◇1 ◆2	MAXORD	◆2 ◇3 ◇4 ◇5 ◇6	ME

For help on any of these options, refer to the *Cadence SPICE Reference Manual*.

OSS-based AMS Netlister

Starting from IC5141USR4, the OSS-based netlister, used by Spectre, Ultrasim and other simulators is also available to be used by AMS in ADE. This netlister uses the same Spectre CDF as used by Spectre, Ultrasim, SpectreVerilog, and UltrasimVerilog. Therefore, the ams simInfo and PDK conversions are not necessary. The OSS-based netlister is hierarchical and netlists the entire hierarchy, as opposed to each cell independently. Since the OSS-based netlister is hierarchical, you cannot share individually netlisted or compiled cells. However, since this netlister is incremental, it will not re-netlist the parts of your design that have not changed.

Important Benefits of OSS-based AMS Netlister

Some of the important benefits of OSS-based AMS netlister are:

- It uses spectre CDFs and spectre netlist procedures to netlist the Spectre primitives. You do not need to add ams siminfo, create ams netlist procedures or convert your PDKs. The netlister also works with verilog views similar to SpectreVerilog and Verilog netlister.
- It does not write into the 5X library. Therefore, you do not require writable master libraries, explicit tmps, or implicit tmps.
- Debugging and the ability to run standalone simulation is expected to be simplified as the design is primarily in one netlist file, similar to Spectre, rather than spread within the 5X library or explicit or implicit library. Additionally, any netlister messages/errors will be consistent with other SpectreVerilog netlister messages/errors. For example, if you understand the messages from the Spectre netlister, you will understand the messages from the OSS-based AMS netlister as the netlisters are same.
- Since the OSS-based netlister does not use the 5X structure, compilation is expected to be faster.
- The OSS-based AMS netlister works with the new 'ncverilog' one step method. This use model is similar to the Verilog-XL use model and it enables functionality such as -y/-v inclusions, similar to Verilog-XL. This approach is more consistent with the digital use model allowing design information to be shared easily.

Choosing the Netlister

If you are using AMS for the first time or migrating from SpectreVerilog/UltraSimVerilog or any other solution, use the OSS-based netlister. If you use the OSS-based AMS netlister, the PDK conversion will not be required and therefore, the transition will be smooth while moving from another solution to OSS-based AMS netlister.

If you are an existing AMS user, you can continue with the current netlister i.e the cell-based netlister. However, if you want to move to OSS-based netlister, refer to Important Benefits of OSS-based AMS Netlister on page 324 and Limitations on page 325 for more details.

Limitations

Listed below are some of the limitations for OSS-based AMS netlister, which would be resolved in a future release:

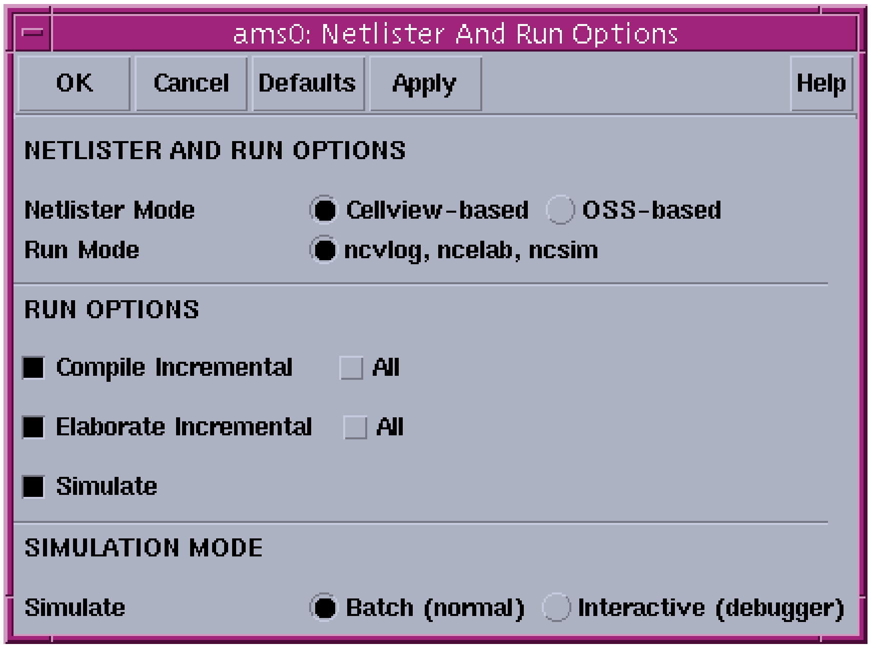
- Like SpectreVerilog, text files need to be imported to dfII with VerilogIn.
- The OSS based netlister uses the spectre simInfo and spectre netlist proc, and netlists in spectre syntax. This spectre syntax needs be translated to verilogAMS syntax for the AMS simulator. This adds an extra step in the process.

- For inherited connections, the OSS netlister creates pseudo-ports which are extra ports propogated up the hierarchy until they resolve in the form of a net or term. However, this behaviour will result in all netSet properties being removed during the port drilling. Therefore, the sideways netSet will not work.
- Supply sensitivity inherited connections have the same limitation as inherited connections listed above, and this will also be resolved in a future release.
- Occurance based and instance based binding is not supported. The repercussions of this is that you can not have two different text views of the same cell instantiated in the same design.
- If the config changes, the design will need to be renetlisted to account for the config changes. However, with the cellview-based netlister, this is not necessary , assuming a verilog.vams file exists for all possible configurations. The configuration changes are accounted for by the simulator with the cellview-based netlister and not the netlister.

Selecting the Netlister

You can select the netlister that you want to use by choosing *Simulation – Netlister and Run Options ...*

The Netlister and Run Options form opens.



NETLISTER AND RUN OPTIONS

Netlister Mode ● Cellview-based ○ OSS-based

Run Mode ● ncvlog, ncelab, ncsim

RUN OPTIONS

■ Compile Incremental □ All

■ Elaborate Incremental □ All

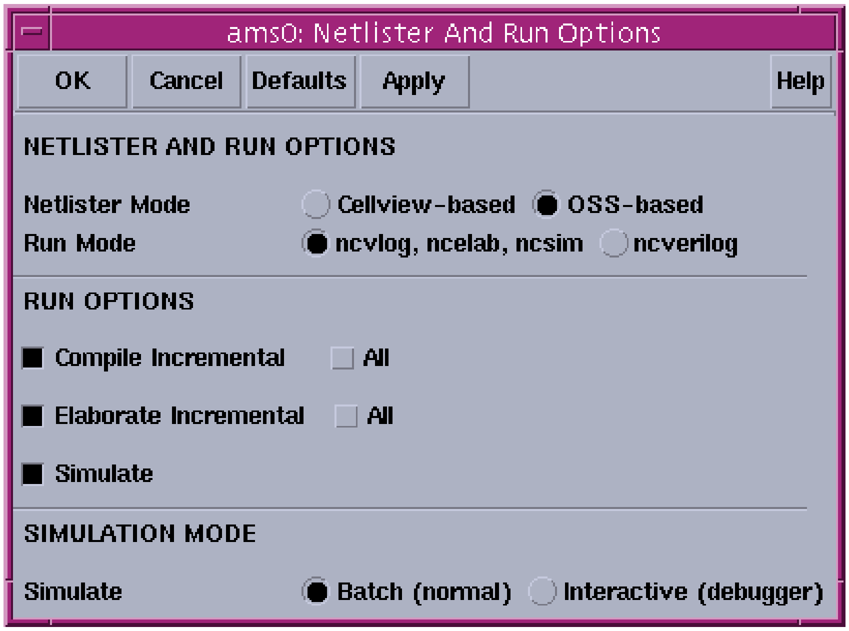
■ Simulate

SIMULATION MODE

Simulate ● Batch (normal) ○ Interactive (debugger)

The Cellview-based netlister is selected by default. If you choose Cellview-based netlister, you will not be able to choose single step (ncverilog) run mode.

However, if you select OSS-based netlister, both the run modes: three step and single step (ncverilog) are avaliable to you. You can select the required run mode and click OK.



NETLISTER AND RUN OPTIONS

Netlister Mode ○ Cellview-based ● OSS-based
Run Mode ● ncvlog, ncelab, ncsim ○ ncverilog

RUN OPTIONS

■ Compile Incremental □ All

■ Elaborate Incremental □ All

■ Simulate

SIMULATION MODE

Simulate ● Batch (normal) ○ Interactive (debugger)

If you choose single step (ncverilog) as the run mode, the ‘AMS RUN OPTIONS’ section will allow you to specify only the feasible combinations that are supported by NC-Verilog options. For example:

- If you select Compile Incremental/all, you will be able to compile.
- If you select Elaborate Incremental/all, you will be able to compile and elaborate both. Even if the Compile Incremental/all is not selected, you will be able to Compile and Elaborate. There is no option in ncverilog that will allow only elaboration to work.
- If you select all the options, you will be able to compile, elaborate and simulate. If only simulate is selected, Compile and Elaborate will not be done.

NC-Verilog

NC-Verilog is a one step run mode for netlisting any design. You can choose the run mode from Netlist and Run Options form if the netlister mode that you have selected is OSS-based.

By selecting the run mode as `ncverilog`, you will be able to specify the command line options from the ncverilog options form. If you need to specify additional advanced options, you can open the Compiler options, Elaborator options and the Simulator options forms respectively from the NC-Verilog form. `ncverilog` provides many arguments and added options that correspond to individual compiler/elaborator/simulator options.

If you want to set an option that is not available in the GUI, you can set it in the Additional arguments field of the NC-verilog options form. Following ncverilog arguments exist using which the remaining options can be passed to ncverilog command line:

+ncvlogargs+<string>	Arguments to pass on to underlying Parser(compiler)
+ncelabargs+<string>	Arguments to pass on to underlying Elaborator
+ncsimargs+<string>	Arguments to pass on to underlying Simulator

To specify an option that takes an argument, enclose the option and argument in quotation marks. To specify more than one option, enclose the list of options in quotation marks. For example, you can specify one or more ncelab options as follows:

```
% ncverilog -f verilog.vc +ncelabargs+-nostdout
% ncverilog -f verilog.vc +ncelabargs+”-errormax 10”
% ncverilog -f verilog.vc +ncelabargs+”-access +r+w -mindelays”
```

If you select ncverilog as an option in the run mode, some of the fields on the Compiler Options, Elaborator Options, and Simulator Options form will be hidden. You can select the respective options button from the NC-Verilog form. For details on these forms, refer to sections Compiler on page 314, Elaborator on page 317 and AMS Simulator on page 320.

Note: If ncverilog is selected as the runmode, the Pack-Files option in the Tools menu is disabled.

In AMS in ADE, ncverilog flow, the verilog AMS compilation (`+ncams`) is used instead of normal digital compilation. Digital Verilog can contain names that are considered keywords in AMS. For example `"s-in"`, in digital verilog modules is the keyword `sin (sine)` in AMS. If you use `"sin"` with ncverilog `+ncams`, you will get an error if it is not used as the `sine` function. To resolve this, you can specify the `‘begin_keywords "1364-2001"` directive around the digital verilog and this will result in AMS not interpretting ams keywords in those modules with the `begin_keyword` directive. For more infromation refer to AMS Designer Simulator User Guide.

If you are using IUS55 or below, the variables defined below are always set to nil as `‘begin_keywords "1364-2001"` directive is not supported. Starting with IUSS57 users, these variables hold true and can be customized as per requirements.

In AMS in ADE ncverilog flow, there are three places where you can have digital blocks and digital files. Therefore, you may need `‘begin_keywords "1364-2001"` directives at such places. The places are:

- Netlist

In netlist, when you include digital blocks, it should be preceded with `‘begin_keywords "1364-2001"` directive. Printing of begin keywords in netlist for digital blocks can be controlled by an environment variable ams.envOpts SpecialHandleForDigitalBlock. If set to t, the `begin_keywords` is printed in the netlist for digital blocks. If set to nil, the `begin_keywords` is not printed in the netlist. The default for this environment variable is t. The syntax for the variable is:

```
ams.envOpts SpecialHandleForDigitalBlock boolean t
```

- Digital files specified through -v option

In the run simulation script, the digital files specified through -v option should be preceded with `‘begin_keywords "1364-2001"` directive. For this, the `‘begin_keywords "1364-2001"` directive is specified in a file amskeyb.v and the corresponding end_keywords are printed in amskeye.v file. These files are prefixed and sufixed respectively to the digital file while printing the ncverilog command line. This is controlled by an environment variable ams.envOpts SpecialHandleForDigLibFiles. If set to t, flow described above holds true. If set to nil, the amskeyb.v and amskeye.v will not be printed in the command line. The default for this environment variable is t. The syntax for the variable is:

```
ams.envOpts SpecialHandleForDigLibFiles boolean t
```

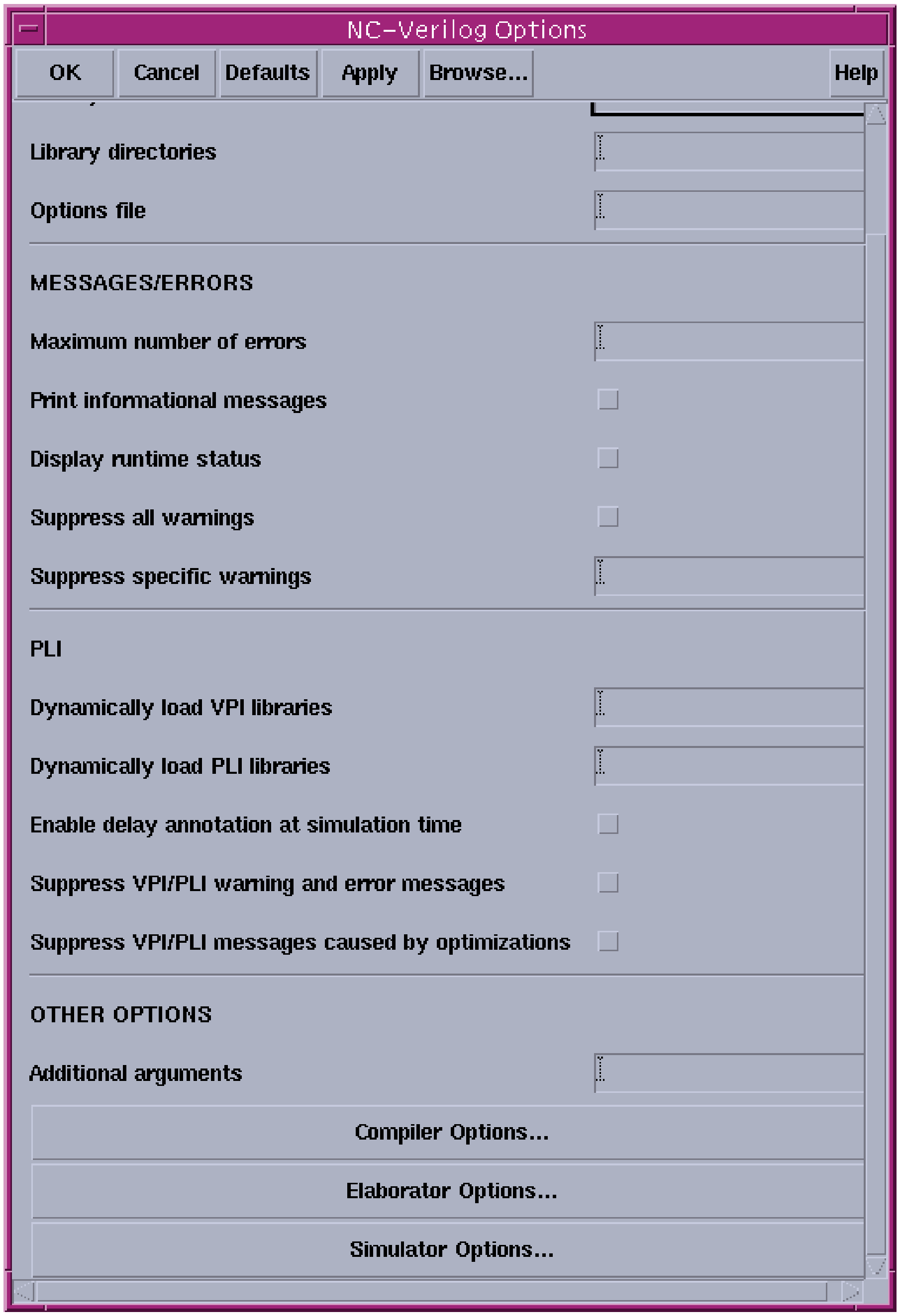
- Digital library directory specified through -y option

In the run simulation script, the digital library directory specified through -y option should be preceded with `‘begin_keywords "1364-2001"` directive. For this, the `‘begin_keywords "1364-2001"` directive is specified in a file amskeyb.v and the corresponding end_keywords are printed in amskeye.v file. These library directories are prefixed and sufixed respectively to the digital library directory name while printing the ncverilog command line. This is controlled by an environment variable ams.envOptsSpecialHandleForDigLibDirs. If set to t, flow described above holds true. If set to nil, the amskeyb.v and amskeye.v will not be printed in the command line. The default for this environment variable is t. The syntax for the variable is:

```
ams.envOptsSpecialHandleForDigLibDirs boolean t
```

NC-Verilog Options form

The ncverilog options can be set in NC-Verilog Options form. To open the form, choose *Simulation – Options – NC-Verilog...*.



Library directories

Options file

MESSAGES/ERRORS

Maximum number of errors

Print informational messages □

Display runtime status □

Suppress all warnings □

Suppress specific warnings

PLI

Dynamically load VPI libraries

Dynamically load PLI libraries

Enable delay annotation at simulation time □

Suppress VPI/PLI warning and error messages □

Suppress VPI/PLI messages caused by optimizations □

OTHER OPTIONS

Additional arguments

In the Include Options section specify the following options:

Library files: Include the specified library file.

Library directories: Include the specified library directory.

Options file: Read the command-line arguments contained in the specified file.

The MESSAGS/ERROS section controls the messages to be displayed.This section is common for ncvlog, ncelab and ncsim. It contains the following fields:

Maximum number of errors: Number of error messages for ncvlog, ncelab and ncsim.

Print informational messages: If selected, displays the meesage from the tool.

Display runtime status: If selected, displays the runtime status.

Suppress all warnings: Supresses all warnings. If this field is selected, the supress all warnings field is disabled.

Suppress specific warnings: Supresses warnings with a code

The OTHER OPTIONS section contains the following fields:

Additional arguments: These options would be as is passed to ncverilog.

Compiler/Elaborator and Simulator Options forms can be opened from this form using the following buttons:

Compiler Options button: Open the compiler options form for NC-Verilog.

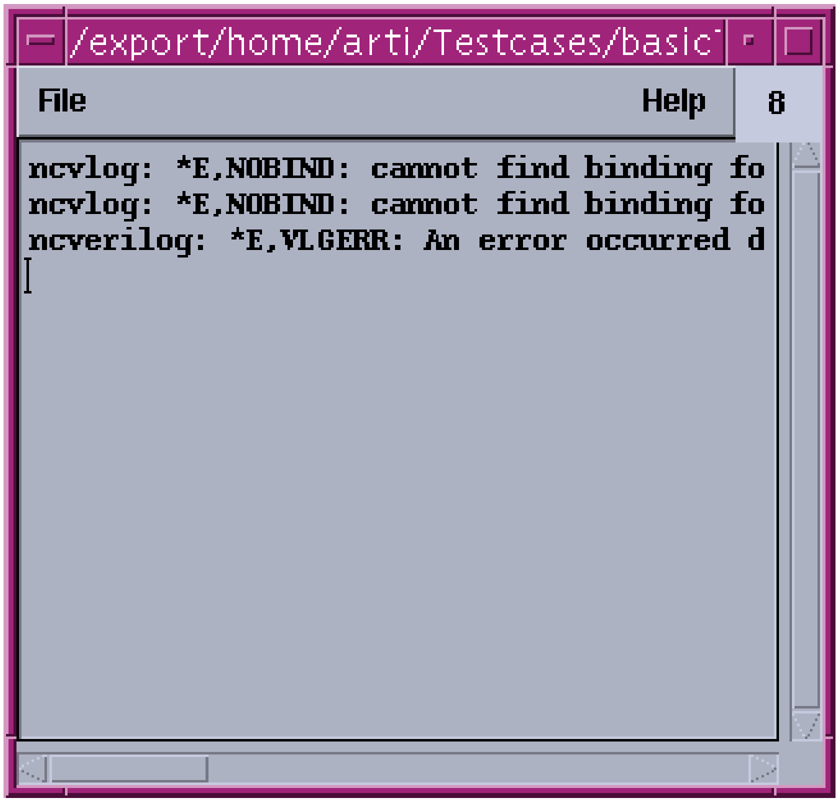
Elaborator Options button: Opens the elaborator options form for NC-Verilog.

Simulator Options button: Opens the simulator options form for NC-Verilog.

Netlist and Run Command

The Netlist and Run command generates the ncverilog command using the compiler, elaborator, simulator and additional ncverilog options.

After running the netlist and run command, you can view the Log file using, *Simulation - Output Log - NC-Verilog Log ...*



```
ncvlog: *E,NOBIND: cannot find binding fo
ncvlog: *E,NOBIND: cannot find binding fo
ncverilog: *E,VLGERR: An error occurred d
```

Note: All the errors generated for compile, elaborator and AMS simulator can be viewed in this log file.

Starting a Simulation

To start a simulation,

➤ Choose *Simulation – Run*.

Alternatively, click the yellow traffic light icon on the simulation window. With this approach, the netlist is assured to reflect any design changes.

To start a simulation using the existing netlist,

➤ Choose *Simulation – Netlist and Run*.

Alternatively, click the green traffic light icon on the simulation window. With this approach, the design is not netlisted if a netlist is already available. This is faster than the

Netlist and Run command, and it can be useful in situations where no design manipulations have been made. The resulting simulation reflects simulation setup modifications such as analysis setup changes, design variable changes, and simulator option changes. It does not reflect design changes such as a change on the edit properties form and a change of the stop and switch view lists on the environment options form.

Note: If data is purged for the current session, you can exit dfII via *File – Exit* or can continue to work in the session. To do this, just reset the purged session and invoke a new session. Also, error messages are generated in the CIW if you select *Simulation – Run* (or *Netlist – Create/Recreate*) in a purged dfII session. The messages will be displayed in the CDS.log file for both `icms` and OCEAN (`icxx`), as follows:

- ❑ **icms**

```
*WARNING* You do not have the required cellViews or properties
open for this session.

*WARNING* You may have purged the data from virtual memory or
the schematic data has been closed.

*WARNING* Reset the ADE session (via Session->Reset) or quit
and re-invoke ADE and other application(s) you are using.
```

- ❑ **OCEAN (CIW or icxx)**

```
You do not have the required cellViews or properties open for
this session. You may have purged the data from virtual memory
or the schematic data has been closed. You can type:
simulator('simulatorName) to reset the session or quit the
application that you are using.
```

Starting a Socket Simulation

To start a simulation,

➤ Choose *Simulation – Run*.

Interrupting or Stopping a Simulation

To stop a simulation that is running,

➤ Choose *Simulation – Stop*.

The system saves any simulation results that were calculated.

The stopped simulation cannot be continued (except for Cadence SPICE transient simulations).

Important

> The *Simulation – Stop* option is not only used to stop a running simulation, it is also used to release the *Spectre* license.

Continuing a Cadence SPICE Transient Simulation

To finish an interrupted Cadence SPICE transient simulation or to continue beyond a completed transient simulation,

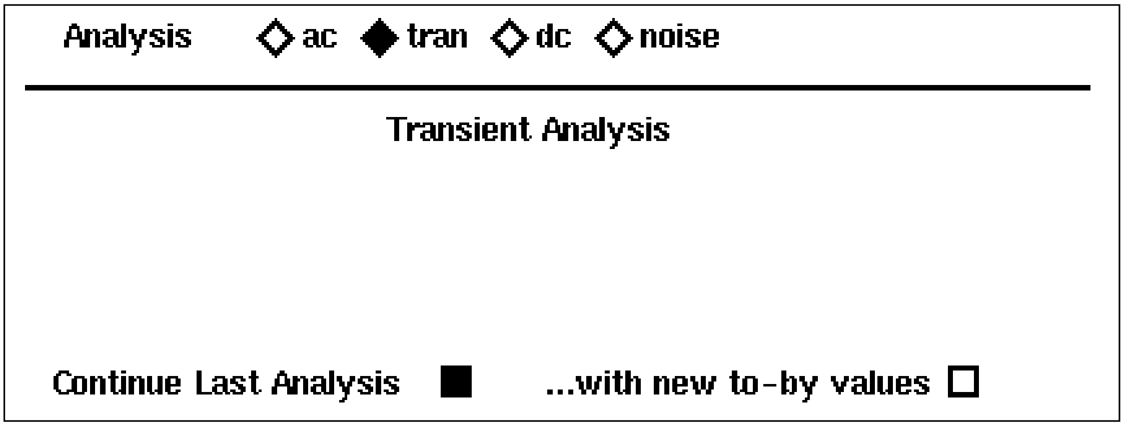
1. In the Simulation window, choose *Analyses – Choose*, or in the Schematic window, choose *Analog Environment – Analyses*.

   The Choosing Analyses form appears.

2. Choose *tran* for a Transient Analysis.

3. Choose *Continue Last Analysis*.

   The Choosing Analyses form is redrawn as shown below.



Analysis ◇ ac ◆ tran ◇ dc ◇ noise

Transient Analysis

Continue Last Analysis ■ ...with new to-by values □

4. Do one of the following:

   - ❑ To continue an interrupted simulation to its previously specified end point, click *OK*.
   - ❑ To specify a new *To*, *By*, or *Max Step value*, click *with new to-by values*, enter new values, and click *OK*.

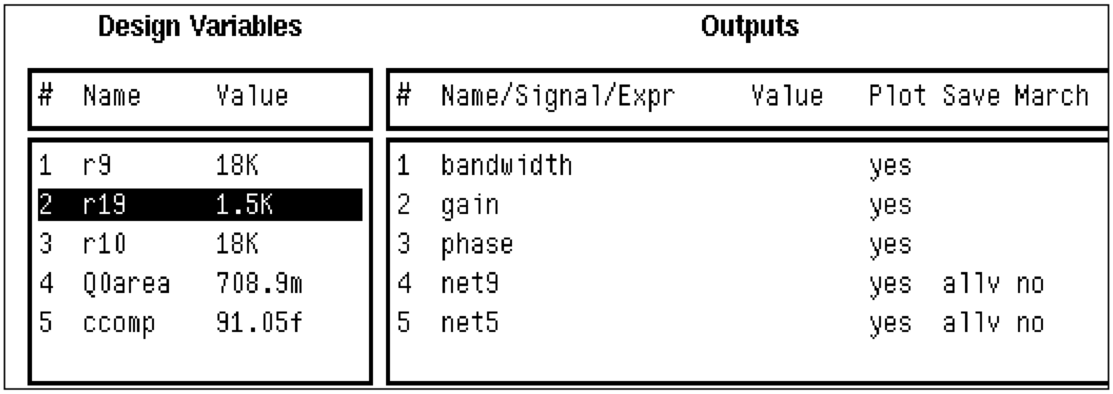
The fields for these options do not appear until you click *with new to-by values*.

The simulation continues.

Updating Variables and Resimulating

To change design variable values and run another simulation,

1. In the Simulation window, double-click the variable you want to change, or in the Schematic window, choose *Setup – Variables*.



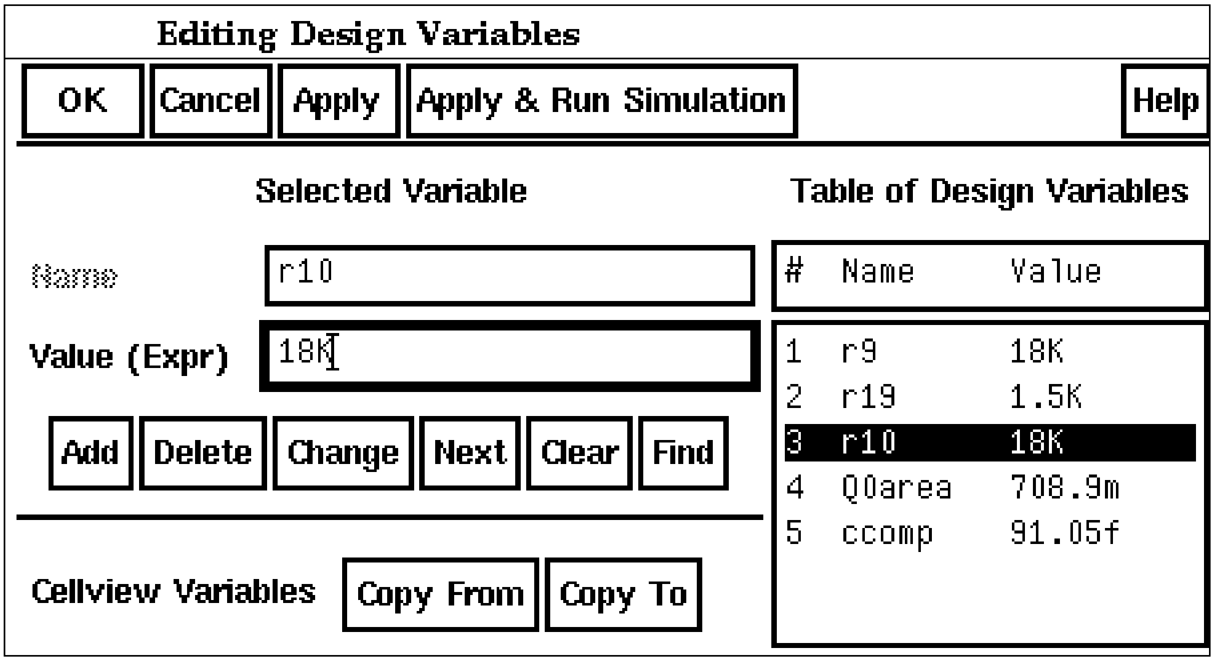
Design Variables

#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	bandwidth		yes		
2	gain		yes		
3	phase		yes		
4	net9		yes	allv	no
5	net5		yes	allv	no

The Editing Design Variables form appears, and the variable you clicked is highlighted.



Name: r10

Value (Expr): 18K

#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	Q0area	708.9m
5	ccomp	91.05f

2. Change the *Value (Expr)* field.
3. Click *Apply & Run Simulation*.

Saving Simulator Option Settings

You can save the current simulator option settings and later restore these settings.

To save the simulator options,

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Environment – Save State*.

   The Saving State form appears.

2. Type a name for the saved simulation state.
3. Check that the *Simulation Options* box is on and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring Saved Settings

To restore saved simulator options,

1. In the Simulation window, choose *Session – Load State*, or in the Schematic window, choose *Analog Environment – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Choose a run directory with the *Cell* and *Simulator* fields.

   The display shows the saved states for the cell and simulator combination.

3. Click a state name.
4. Check that *Simulation Options* is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Viewing the Simulation Output

Simulators integrated in the Cadence SPICE socket create a log file.

To read the log file,

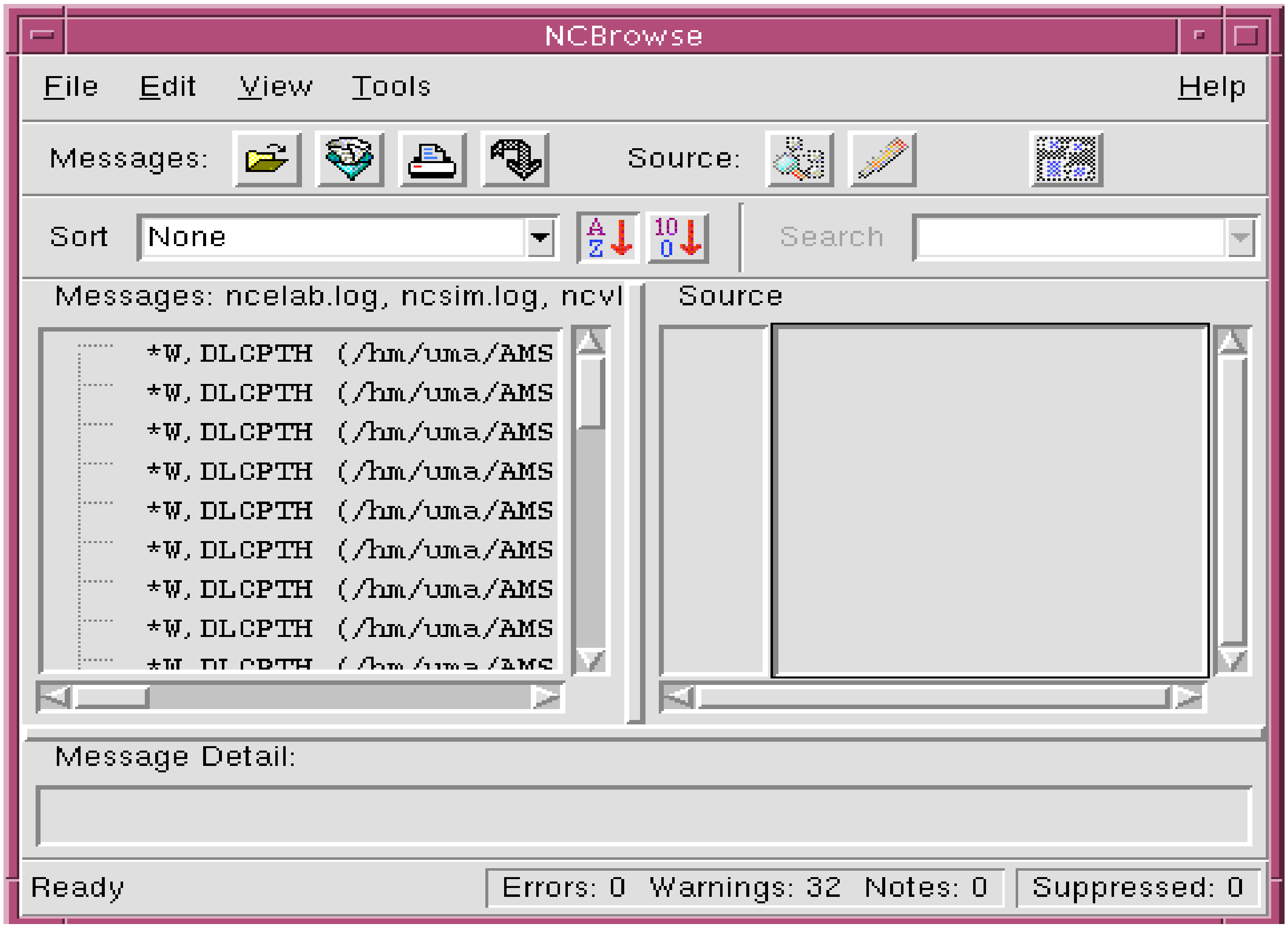
➤ Choose *Simulation – Output Log*.

Some simulators write a second output file. To read this file,

➤ Choose *Simulation – Textual Output*.

Viewing the Output Log for AMS

You can view the output logs (Netlister, Compiler, Elaborator, Simulator) from the *Simulation – Output Log* submenu. Choose *Output Log – LogFile utility* to launch the NCBrowse message browser.



Sort: None

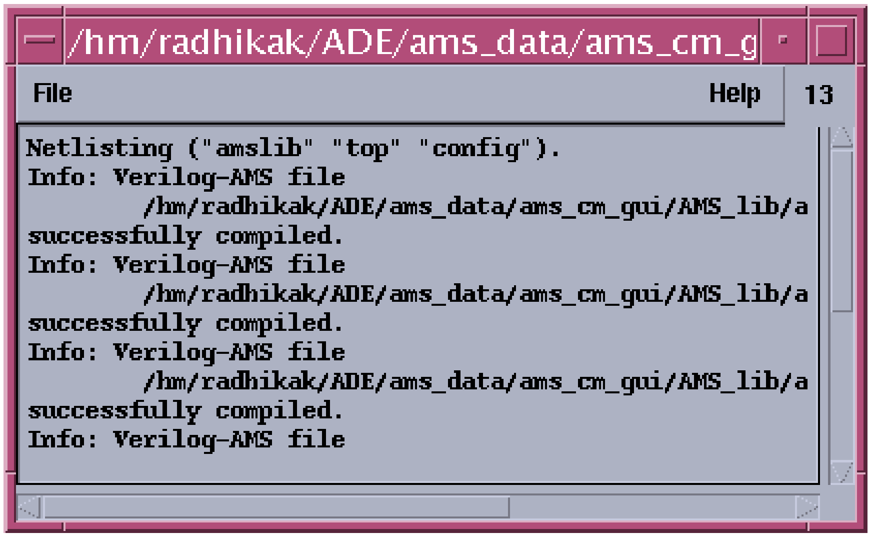
Messages: ncelab.log, ncsim.log, ncvl

```
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
*W,DLCPTH (/hm/uma/AMS
```

Ready   Errors: 0 Warnings: 32 Notes: 0   Suppressed: 0

You can analyze log files with this utility. The NCBrowse message browser lets you select a log file message and view the source code that caused the message. It includes sorting and filtering tools that let you view only those messages that are important to you. You can also use the message browser to print formatted output reports. To know more about the NC Browse utility, refer to the *NC Browse Message Browser User Guide.*

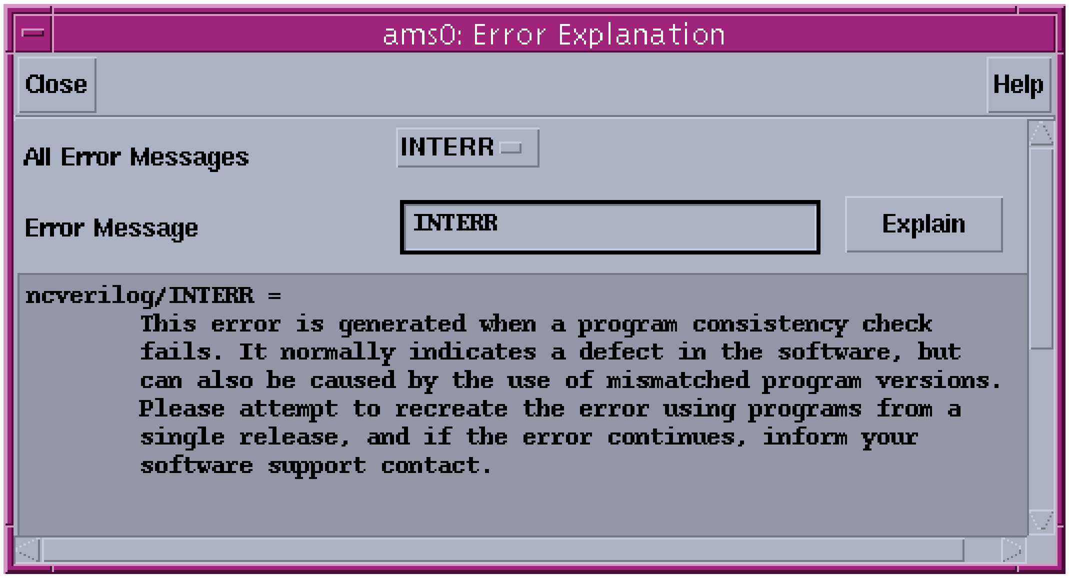
Similarly, choose the *Netlister Log*, *Compiler Log*, *Elaborator Log* and *Simulator Log* options to view the respective logs in the *Results Browser*.



```
Netlisting ("amslib" "top" "config").
Info: Verilog-AMS file
        /hm/radhikak/ADE/ams_data/ams_cm_gui/AMS_lib/a
successfully compiled.
Info: Verilog-AMS file
        /hm/radhikak/ADE/ams_data/ams_cm_gui/AMS_lib/a
successfully compiled.
Info: Verilog-AMS file
        /hm/radhikak/ADE/ams_data/ams_cm_gui/AMS_lib/a
successfully compiled.
Info: Verilog-AMS file
```

Viewing the Error Explanation for AMS

You can view detailed explanation of the error for AMS in the Error Explanation form. Choose *Simulation – Output Log – Error Explanation...* to launch the Error Explanation form.



All Error Messages: INTERR

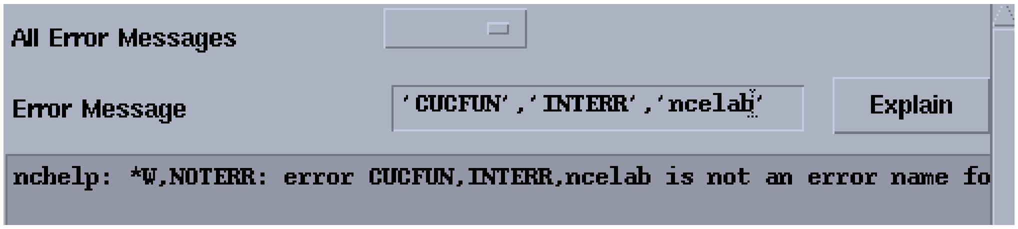
Error Message: INTERR

```
ncverilog/INTERR =
        This error is generated when a program consistency check
        fails. It normally indicates a defect in the software, but
        can also be caused by the use of mismatched program versions.
        Please attempt to recreate the error using programs from a
        single release, and if the error continues, inform your
        software support contact.
```

All the error messages that are present in the log files that are created in the psf directory while a session is being run are listed in the All Error Messages field. To view details about an error message, you can select the Error from All Error Messages field and then click the Explain button. The description for the error appears as shown in the figure above.

You can also enter the error in the Error Message field. If you enter a string, it will be treated as an error string. If you enter two strings, the first string is treated as the tool name and the second string is treated as an error message. If you enter more than two strings, you will get a warning. For example, if you enter "`CUCFUN INTERR ncelab`" as the error string, you will get a warning

```
nchelp: *W,NOTERR: error CUCFUN,INTERR,ncelab is not an error name
for nceverilog.
```



All Error Messages:

Error Message: 'CUCFUN','INTERR','ncelab'

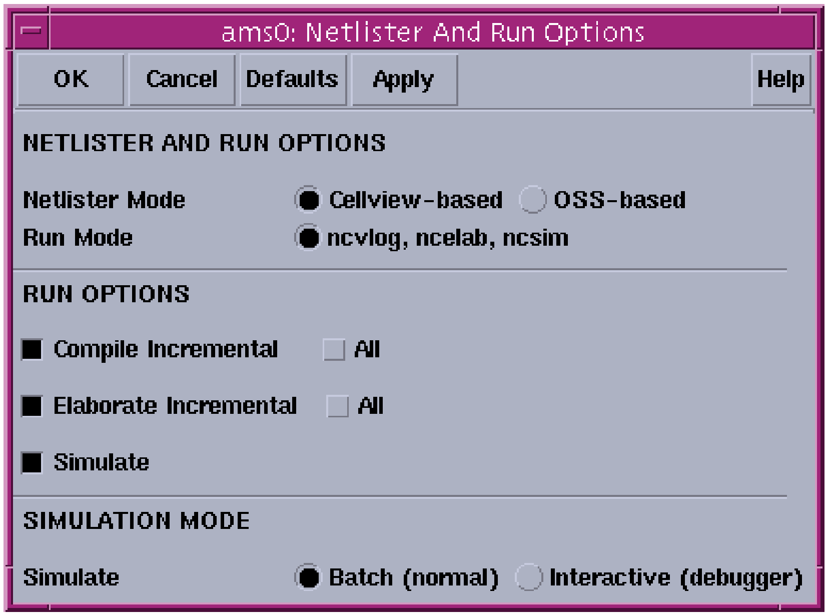
```
nchelp: *W,NOTERR: error CUCFUN,INTERR,ncelab is not an error name fo
```

The error message description consists of the following:

- For a growing number of messages, special messages are available to describe the problem and if known, a solution for the same. The number of messages for which there will be additional help is expected to grow with each release.
- For all error names, the nchelp for that error name is printed.

Using the SimVision Debugger with AMS

You can choose AMS run options and simulation run options by choosing *Simulation – Netlist and Run Options...*. This brings up the Netlist and Run Options form.



NETLISTER AND RUN OPTIONS

Netlister Mode ● Cellview-based ○ OSS-based

Run Mode ● ncvlog, ncelab, ncsim

RUN OPTIONS

■ Compile Incremental □ All

■ Elaborate Incremental □ All

■ Simulate

SIMULATION MODE

Simulate ● Batch (normal) ○ Interactive (debugger)

You can choose to only compile, elaborate or simulate the design, although by default all three are selected as shown in the screenshot. For example, if you select only *Elaborate*, the *Simulation – Netlist* command only elaborates the design. Compilation and elaboration can be incremental or for the whole design together. A simulation would fail if you choose both *Compile* and *Simulate*. You would need to either deselect *Simulate* or select *Elaborate* as well.

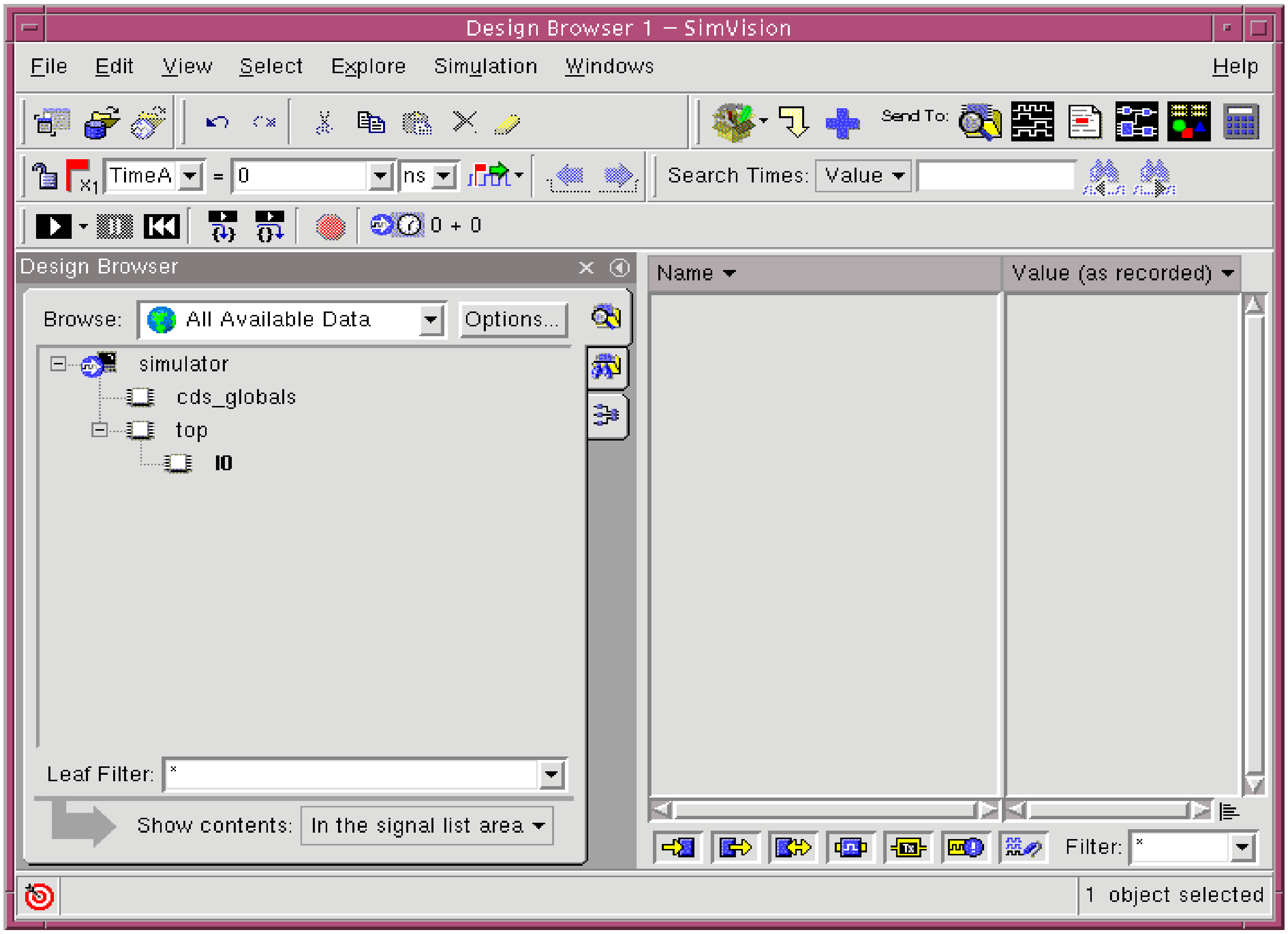
The *Batch* mode is the default mode. In this mode, all signal plotting occurs in the default analog waveform plotting tools – AWD or WaveScan.

The *Interactive* mode launches the complete SimVision debug environment on top of current design. SimVision is a graphical user interface for Cadence simulators and related debugging tools. For details, refer to the *SimVision User Guide* and the *Cadence AMS Simulator User Guide.*

In the interactive mode,

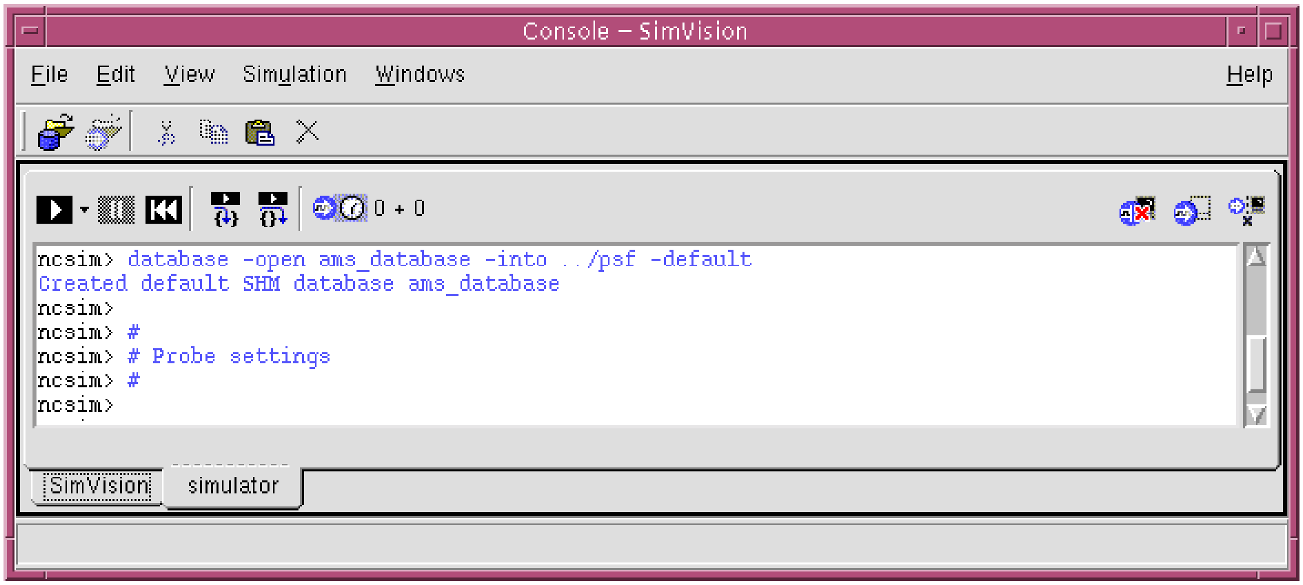
- The *Parametric Analysis*, *Corners*, *Optimization* and *Pack-N-Go commands* in the *Tools* menu cannot be used. If you attempt to use them, a prompt appears saying that the option is not enabled for the *interactive* mode and suggests that you switch to the *batch* mode to be able to use it. It also warns you that if you attempt to rerun the simulation without exiting SimVision, the simulation results may get corrupted.
- The saved currents and voltages can be plotted in SimVision Waves through the ADE selection mechanism. The ADE plot outputs signals go to SimVision Waves if it is open. If it is not open, the output goes to the selected waveform tool – AWD or WaveScan.
- If you select new nets for the plot list, they take effect in the next SimVision run.
- If you select within SimVision’s browser, the selected information may be placed in the ADE Outputs pane.

Direct Plot and plotting from the calculator or browser go to the selected waveform tool, regardless of the run mode or whether or not SimVision is up.



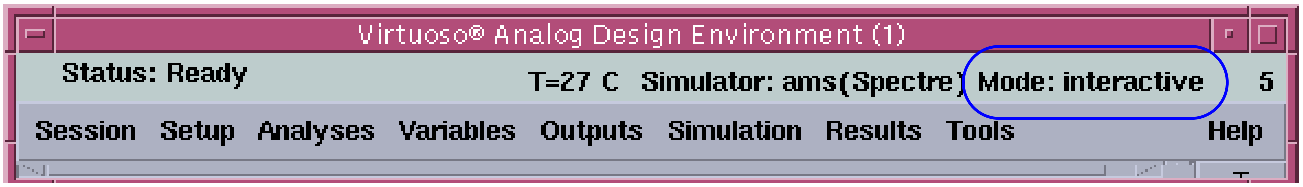
ADE sends Tcl commands to SimVision for the simulation run and to plot the signals listed in the ADE outputs pane. This mode lets you control ncsim and simvision from the console window.

The status of SimVision plotting commands can be seen at the SimVision prompt in the console window.



```
ncsim> database -open ams_database -into ../psf -default
Created default SHM database ams_database
ncsim>
ncsim> #
ncsim> # Probe settings
ncsim> #
ncsim>
```

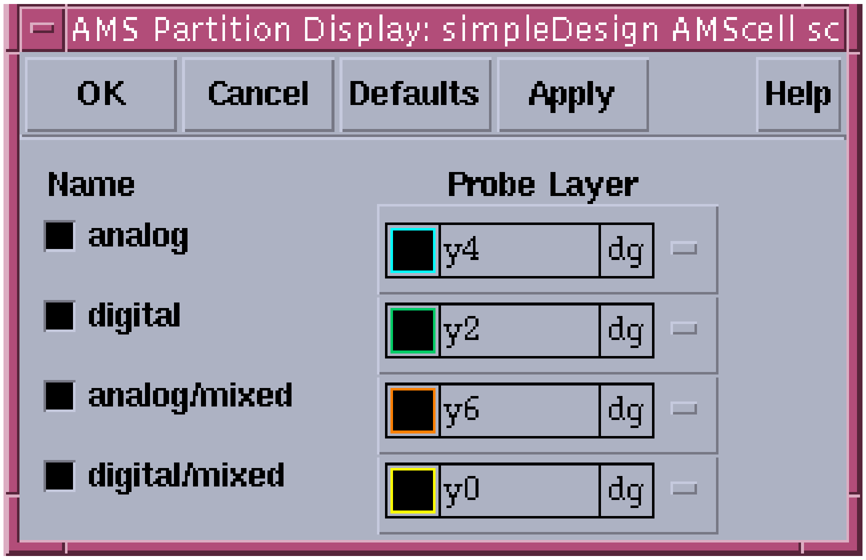
The ADE window displays the selected mode below the title bar as highlighted in this snapshot.



Display Partition

Once you run a simulation, you can view your data and distinguish between analog instances or nets, digital instances or nets, and mixed instances or nets by the color associated with each partition.

1. To access this feature, choose *Tools – AMS Opts* in your schematic window. The *AMS* option appears on the menu bar.
2. Run a simulation and choose *AMS – Display Partition – Initialize* so that the Display Partition menu commands are enabled.
3. Choose *AMS – Display Partition – Interactive* to view the *Partition Display* window.



Name	Probe Layer
■ analog	y4 dg
■ digital	y2 dg
■ analog/mixed	y6 dg
■ digital/mixed	y0 dg

This feature highlights the analog and digital parts of a schematic in different colors.

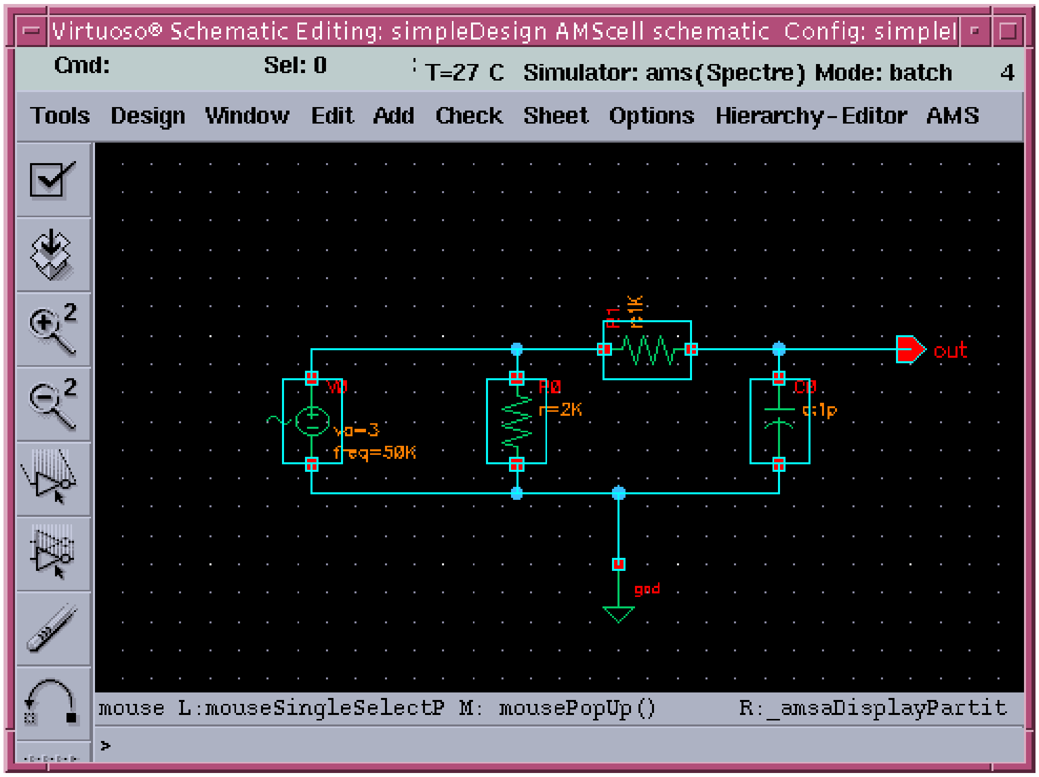
analog indicates that the net or instance is analog in nature. A net is analog throughout a hierarchy if everything under that instance is analog.

digital indicates that the net or instance is digital in nature. A net is digital throughout a hierarchy if everything under that instance is digital

analog/mixed indicates that the net is mixed in nature. It means that some segments of the net are analog and some are digital in the hierarchy.

digital/mixed indicates the same as analog/mixed, the difference being that at the current level, the net is digital.

The schematic reflects these color preferences.

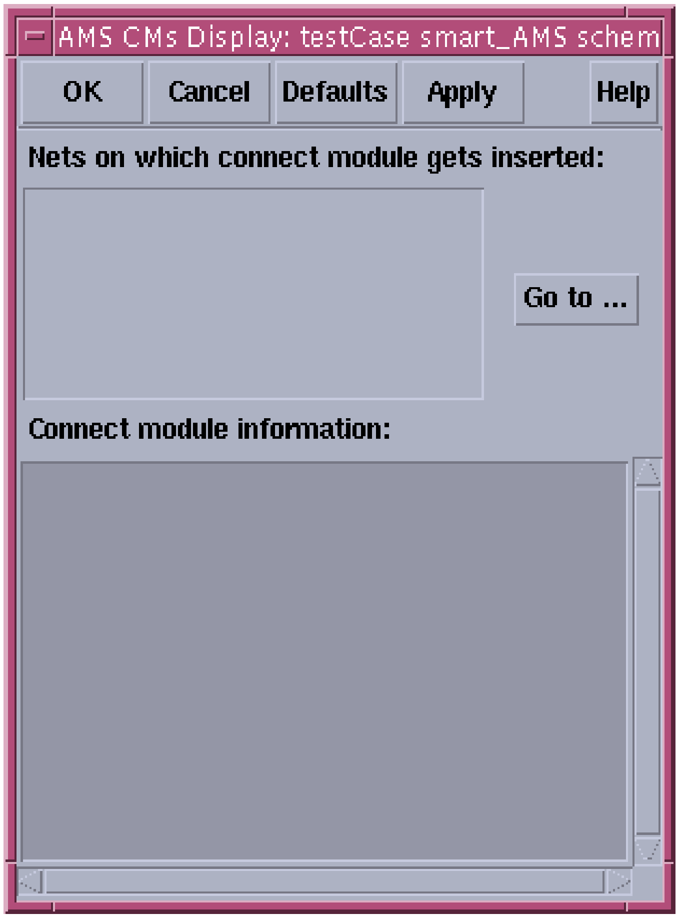


Virtuoso® Schematic Editing: simpleDesign AMScell schematic Config: simpleI

Cmd: Sel: 0 T=27 C Simulator: ams(Spectre) Mode: batch 4

```
mouse L:mouseSingleSelectP M: mousePopUp()      R:_amsaDisplayPartit
>
```

4. You can see all the IEs in the configuration by choosing *AMS – Display Partition – IE Information*. This brings up the CMs Display dialog box displaying all IEs.


AMS CMs Display: testCase smart_AMS schem
OK
Cancel
Defaults
Apply
Help
Nets on which connect module gets inserted:
Go to ...
Connect module information:

You can select an IE and click *Go to* to see it zoomed-in on the schematic. When you select an IE, information pertaining to it and related connect module information appears in the box below.

Default Digital Discipline Selection

Disciplines denote an object as analog (with an electrical discipline, for example) or digital (with a logic discipline, for example). You can define the default disciplines for design objects by using the *AMS – Default Digital Discipline Selection* command in the Composer window.

Default digital discipline selection indirectly controls the selection of connect module (IE) on mixed nets. A discipline denotes an object as analog or digital based on whether it is electrical or logic, respectively. When objects of different disciplines are connected, connect rules determine which connect modules are inserted on mixed nets.

The inserted connect modules then convert signals to values that are appropriate for each discipline. To customize the conversions for your design, you can use the connect rules to override parameters, such as supply voltage or rise time, that are used in the connect modules.

For more information, see the *Mixed-Signal Aspects of Verilog-AMS* chapter of the *Cadence Verilog-AMS Language Reference*.

To specify a default digital discipline for design objects,

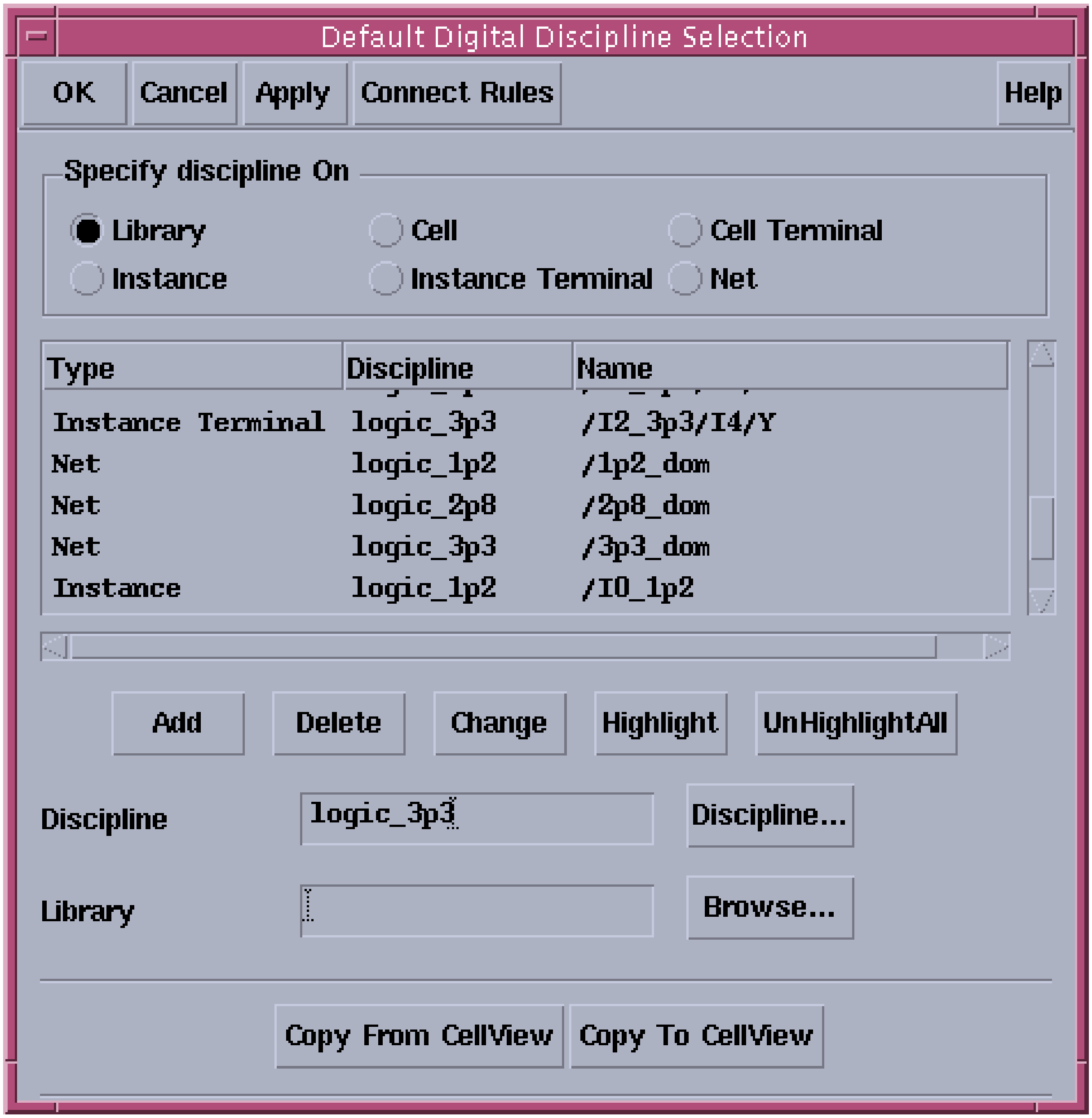
1. Choose *Tools – AMS Opts.* in your schematic window.

   The *AMS* menu appears on the menu bar.

2. Choose *AMS – Default Digital Discipline Selection*.

   A submenu appears showing six options: *Library*, *Cell*, *Cell Terminal*, *Net*, *Instance*, *Instance Terminal*.

3. Select any of the submenu options to bring up the Default Digital Discipline Selection form. For example, if you select *Library*, the form comes up as shown here. You can specify disciplines on libraries in this form.



Specify discipline On

● Library ○ Cell ○ Cell Terminal
○ Instance ○ Instance Terminal ○ Net

Type	Discipline	Name
Instance Terminal	logic_3p3	/I2_3p3/I4/Y
Net	logic_1p2	/1p2_dom
Net	logic_2p8	/2p8_dom
Net	logic_3p3	/3p3_dom
Instance	logic_1p2	/I0_1p2

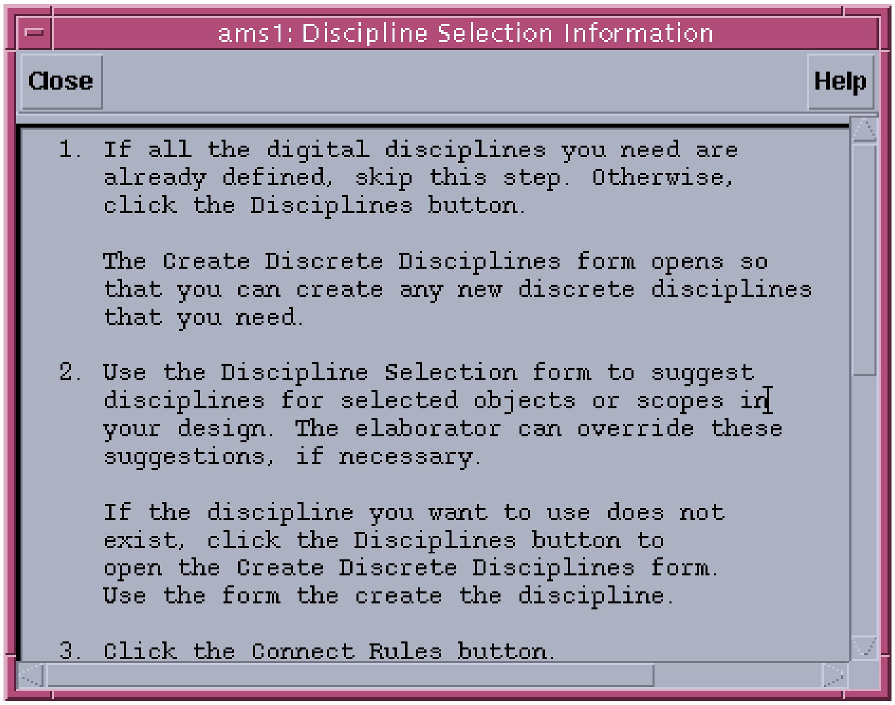
Discipline logic_3p3

Library

Alternatively, you can select any of the other options from the *Specify discipline on* group box. The fields below the *Discipline* field change as follows depending on the design object selected.

Design Object	**Related fields**	**How to specify**
Library	*Library*	Type in a valid value or use the *Browse* button to specify a library from the Library Browser.
Cell	*Library* *Cell*	Type in valid values or use the *Browse* button to specify a library and cell from the Library Browser.
Cell Terminal	*Library* *Cell* *Terminal*	Type in valid values or use the *Select* button to select a cell terminal from the schematic.
Net	*Net*	Type in a valid value or use the *Select* button to select a net from the schematic.
Instance	*Instance*	Type in a valid value or use the *Select* button to select an instance from the schematic.
Instance Terminal	*Instance Terminal*	Type in a valid value or use the *Select* button to select an instance terminal from the schematic.

When you open the form for the first time, a file showing discipline selection information pops up with information about disciplines.



```
  1. If all the digital disciplines you need are
     already defined, skip this step. Otherwise,
     click the Disciplines button.

     The Create Discrete Disciplines form opens so
     that you can create any new discrete disciplines
     that you need.

  2. Use the Discipline Selection form to suggest
     disciplines for selected objects or scopes in
     your design. The elaborator can override these
     suggestions, if necessary.

     If the discipline you want to use does not
     exist, click the Disciplines button to
     open the Create Discrete Disciplines form.
     Use the form the create the discipline.

  3. Click the Connect Rules button.
```

4. The table lists the types of design objects, their default disciplines and their names. You can sort this table on *Type* or *Discipline*.

5. To create a new discipline,

a. Click the *Discipline* button.

The Create Discrete Disciplines form appears in which you can create a discipline. The new discipline appears in the *Discipline* field.

b. Click the *Add* button.

6. To delete one or more disciplines specified on an object,

a. Select one or more rows in the table.

b. Click the *Delete* button.

7. To change a discipline specified on an object,

a. Select a row in the table.

The form refreshes to show the fields pertaining to the selected kind of object.

b. Change the values as required and click the *Change* button.

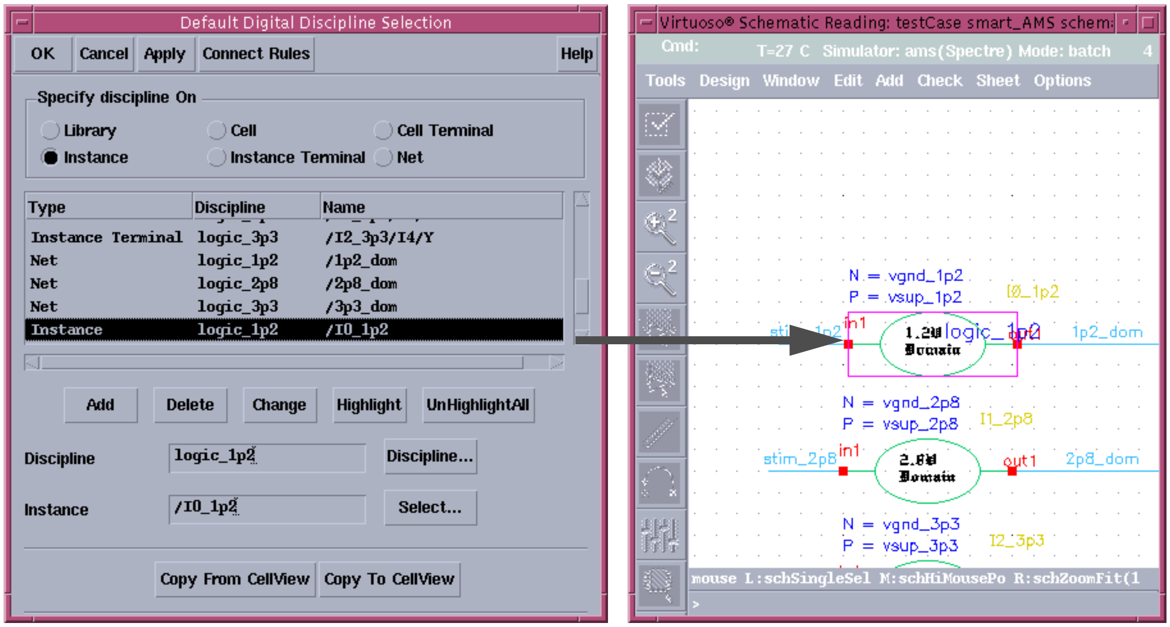
8. To view a discipline on the schematic,

   a. Select a row from the table.

      Note that only nets, instances and instance terminals can be highlighted.

   b. Click the *Highlight* button.

      As shown in the illustration below, the selected discipline, in this case `logic_1p2`, appears highlighted in the schematic. You can select and highlight multiple objects this way. You can click the *Unhighlight All* button to remove the highlights.



Default Digital Discipline Selection

Specify discipline On

○ Library ○ Cell ○ Cell Terminal
● Instance ○ Instance Terminal ○ Net

Type	Discipline	Name
Instance Terminal	logic_3p3	/I2_3p3/I4/Y
Net	logic_1p2	/1p2_dom
Net	logic_2p8	/2p8_dom
Net	logic_3p3	/3p3_dom
Instance	logic_1p2	/I0_1p2

Discipline: logic_1p2

Instance: /I0_1p2

Virtuoso® Schematic Reading: testCase smart_AMS schem:

Cmd: T=27 C Simulator: ams(Spectre) Mode: batch 4

mouse L:schSingleSel M:schHiMousePo R:schZoomFit(1

9. To copy disciplines from the cellview, click the *Copy from Cellview* button. Conversely, to copy disciplines from the form to the cellview, click the *Copy to Cellview* button.

   After copying over disciplines from a cellview, when you click the *Discipline* button, a form may pop up listing disciplines that have not been defined and asking if you would like to define them. If you select *Yes*, the Create Discrete Disciplines form appears showing a list of the undefined disciplines. If you select *No*, it appears blank.

10. You can specify connect rules for a selected discipline by using the *Connect Rules* button to bring up the Select Connect Rules form. You can use this form to create, modify or delete connect rules. This form does appears only if ADE is up and the simulator set to ams. Otherwise, an error message appears.

11. Click *OK*.

The disciplines created are automatically compiled.

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Discipline Selection* option in the Saving State form and save the current ADE state. You can later load the state using the Loading State form with the *Discipline Selection* option selected. For more information, see Saving and Restoring the Simulation Setup on page 77.

Entering Simulator Commands in the Type-In Window

Through the Type-In window, you can send commands directly to Cadence SPICE, type SKILL commands, and, in mixed-signal simulation, send commands to the Verilog®-XL simulator.

Note: This is available only for socket interfaces.

To enter Cadence SPICE commands or SKILL commands,

1. Choose *Simulation – Command Type-In*.
2. Type into the field at the bottom of the window.

   To enter SKILL commands, precede the commands with the at (@) character.

To enter Verilog-XL commands during mixed-signal simulation,

➤ Set the *Stop After Compilation* option on the Verilog-XL Simulation Options form before starting the simulation.

   This halts Verilog-XL processing after compilation and displays the following message in the CIW:

```
Verilog/spectre Mixed-Signal Interface
Type ? for help.
```

During a mixed-signal simulation, do *not* enter commands in the CIW. Instead, use the following steps.

1. When the message appears, choose *Simulation – Command Type-In* from the Simulation window.

   This displays the Virtuoso® analog circuit Type-In window.

2. At the bottom of the Type-In window, enter a Verilog-XL command preceded by a dollar sign ($).

   The dollar sign ($) identifies the command as input to the Verilog-XL simulator.

For example, you enter the Verilog-XL `$display` in the Type-In window as

```
$display
```

with the appropriate arguments.

You can use all Verilog interactive debugging commands. Refer to the *Verilog Reference Manual* for commands that are available.

To plot a digital waveform in SimVision, type the following commands:

```
$shm_open("my.db")
$shm_probe("AS")
```

This saves the signal waveform database in `my.db` in the netlist digital directory.

You need to start SimVision in standalone mode from a UNIX window by typing `simvision -waves` at the prompt. When SimWave starts, load the waveform database `my.db` and use the *Tools – Browser* to choose signals to plot.

Note: Do not use the Verilog-XL commands `$restart` or `$save` from a mixed-signal simulation. Using these interactive job control commands within an active mixed-signal simulation can affect the interprocess communication and end your simulation.

3. When you want to resume mixed-signal simulation, enter the `.` command preceded by a dollar sign this way: `$.`

Running a Parametric Analysis

For information on how to select range specifications, start, interrupt, re-start and close a parametric analysis run, see Running a Parametric Analysis.

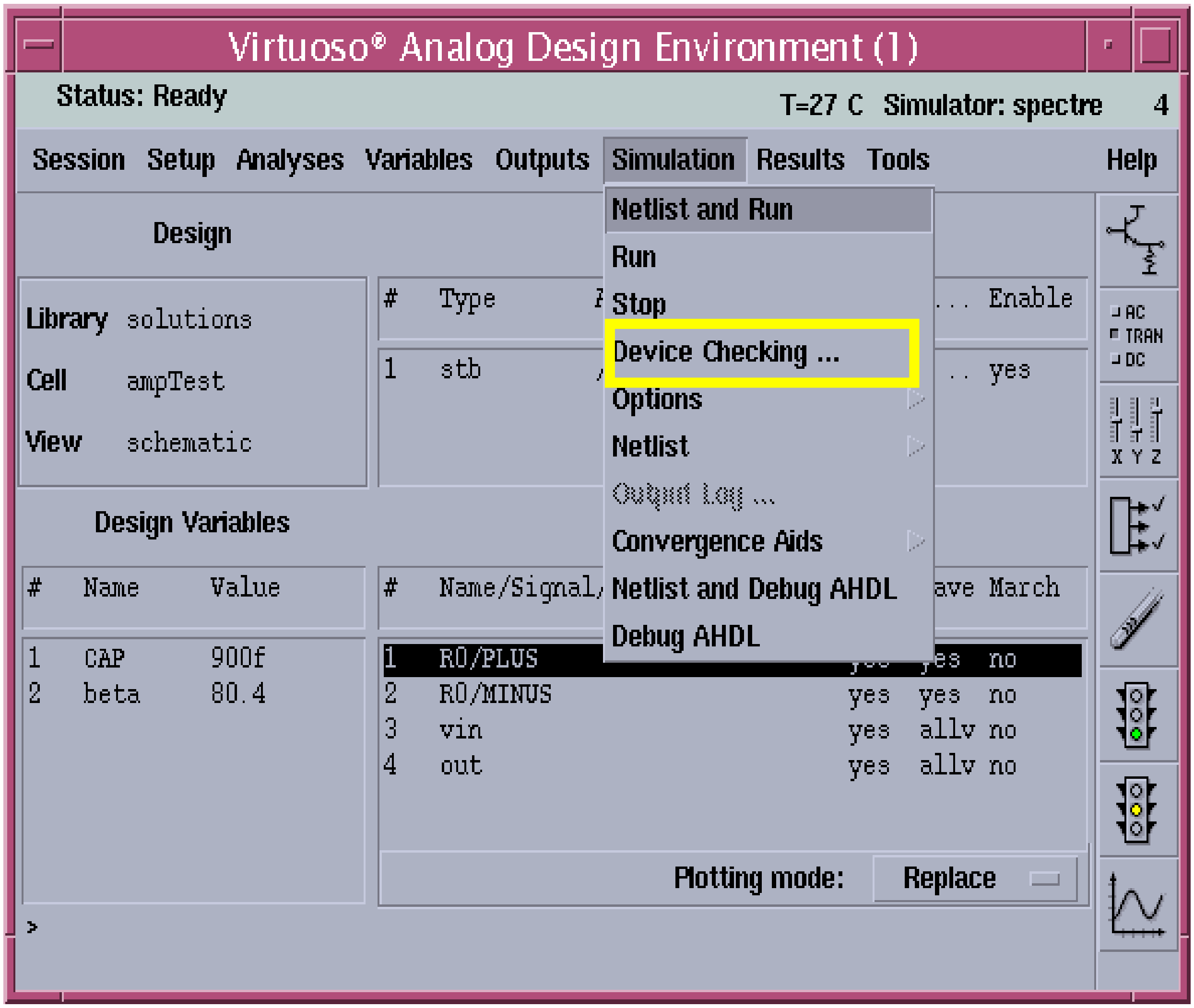
Setting Up and Running Statistical Analyses

For information about running statistical analyses, see the *Virtuoso Advanced Analysis Tools User Guide*.

Device Checking

The Analog Design Environment is now enhanced to support Spectre's device checking capability. This capability allows you to determine if elements in your circuit are violating predefined safe operating areas. The rules can include checks of operating point parameters, as well as expressions that combine these parameters. These checks can be part of your model card, netlist, or any other file that is included for simulation as a part of the design. A graphical user interface is available for you to add/modify device checks, run *Spectre* with device checks and see violations. You can print a summary of all the violations, highlight violating devices in the schematic and obtain all violations for any device in the schematic. Filters can be used to reduce the number of violations printed/devices highlighted.

To access this feature, click *Simulation – Device Checking* in the simulation window.



Status: Ready   T=27 C Simulator: spectre   4

Design

Library solutions
Cell ampTest
View schematic

#	Type	...	Enable
1	stb	...	yes

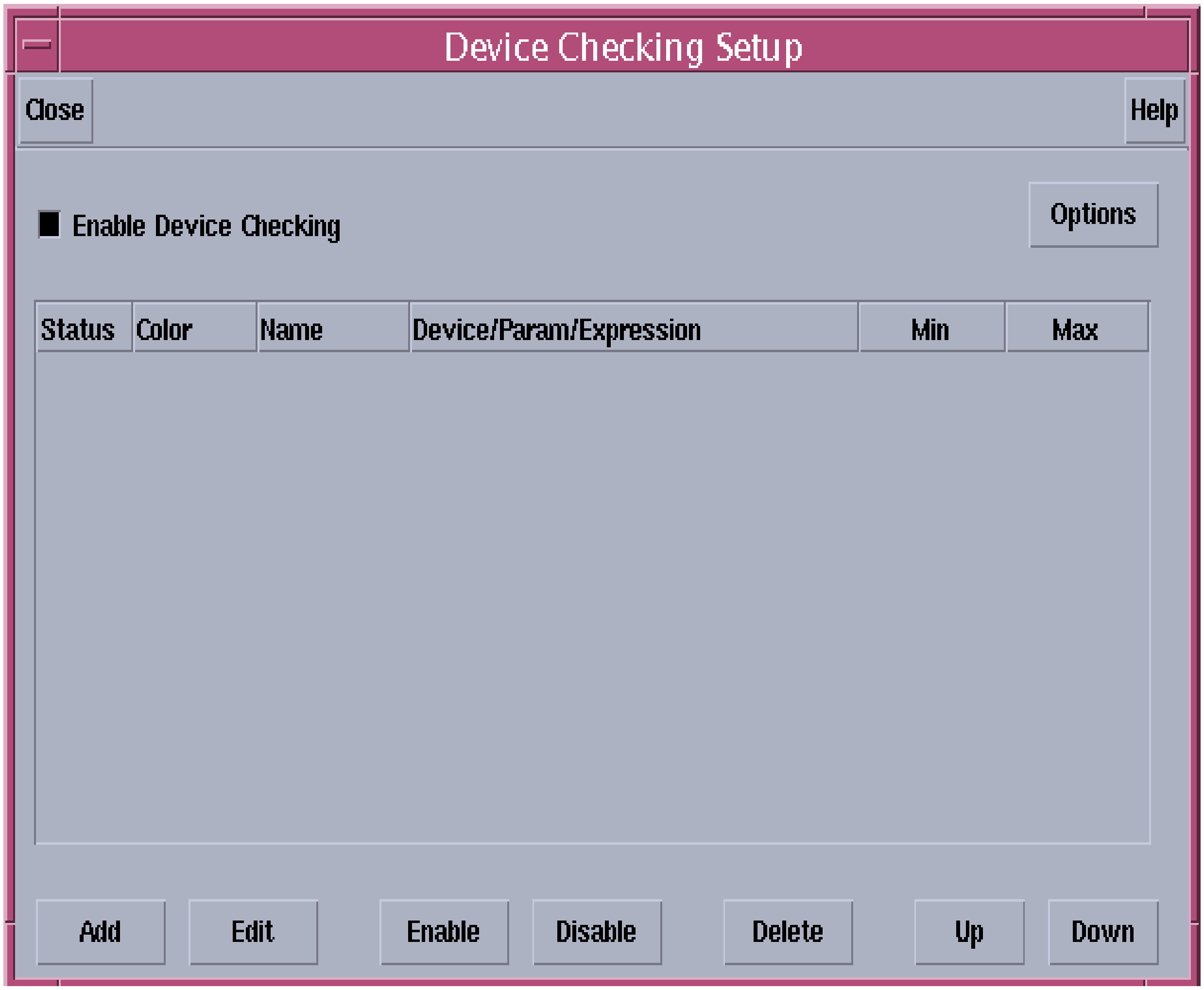
Design Variables

#	Name	Value
1	CAP	900f
2	beta	80.4

#	Name/Signal/		ave	March
1	R0/PLUS		es	no
2	R0/MINUS	yes	yes	no
3	vin	yes	allv	no
4	out	yes	allv	no

Plotting mode: Replace

The *Device Checking Setup* form is displayed:



■ Enable Device Checking

This form is used to enter and manage asserts.

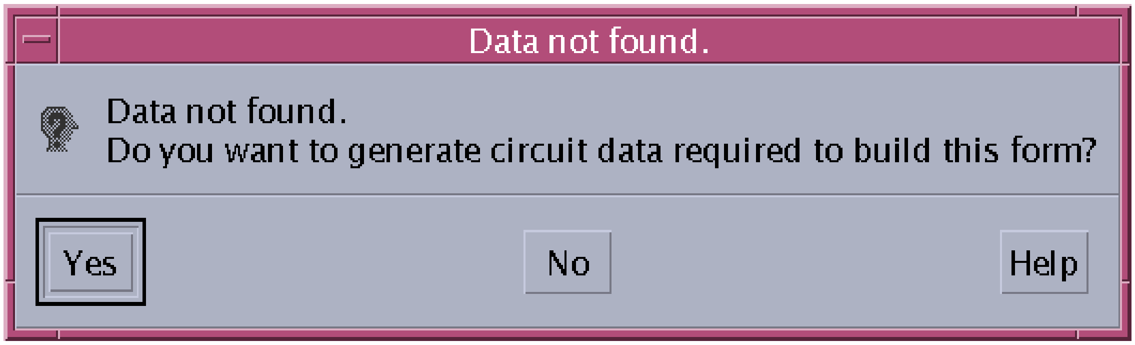
The **Enable Device Checking** button is on by default. You can write asserts (entered using the graphical user interface) to the netlist only when this button is enabled.

The **List box** field displays all the asserts that have been entered.The words ON/OFF under the *Status* column indicate whether the assert is enabled or disabled.

The **Add** button brings up the *Edit Device Check* form which is used to add new asserts.

The **Edit** button brings up the *Edit Device Check* form which is used to update a selected assert.

Note: When you click on *Add* or *Edit* (after loading an existing state) for the first time, a *Data not found* dialog box requesting confirmation for generating circuit data appears:



Data not found.

Data not found.
Do you want to generate circuit data required to build this form?

The **Enable** button is used to enable one or more asserts selected in the list. The status of an assert is indicated by the words ON/OFF preceding the assert.

The **Disable** button is used to enable one or more asserts selected in the list of asserts.This disables the assert for all analyses.

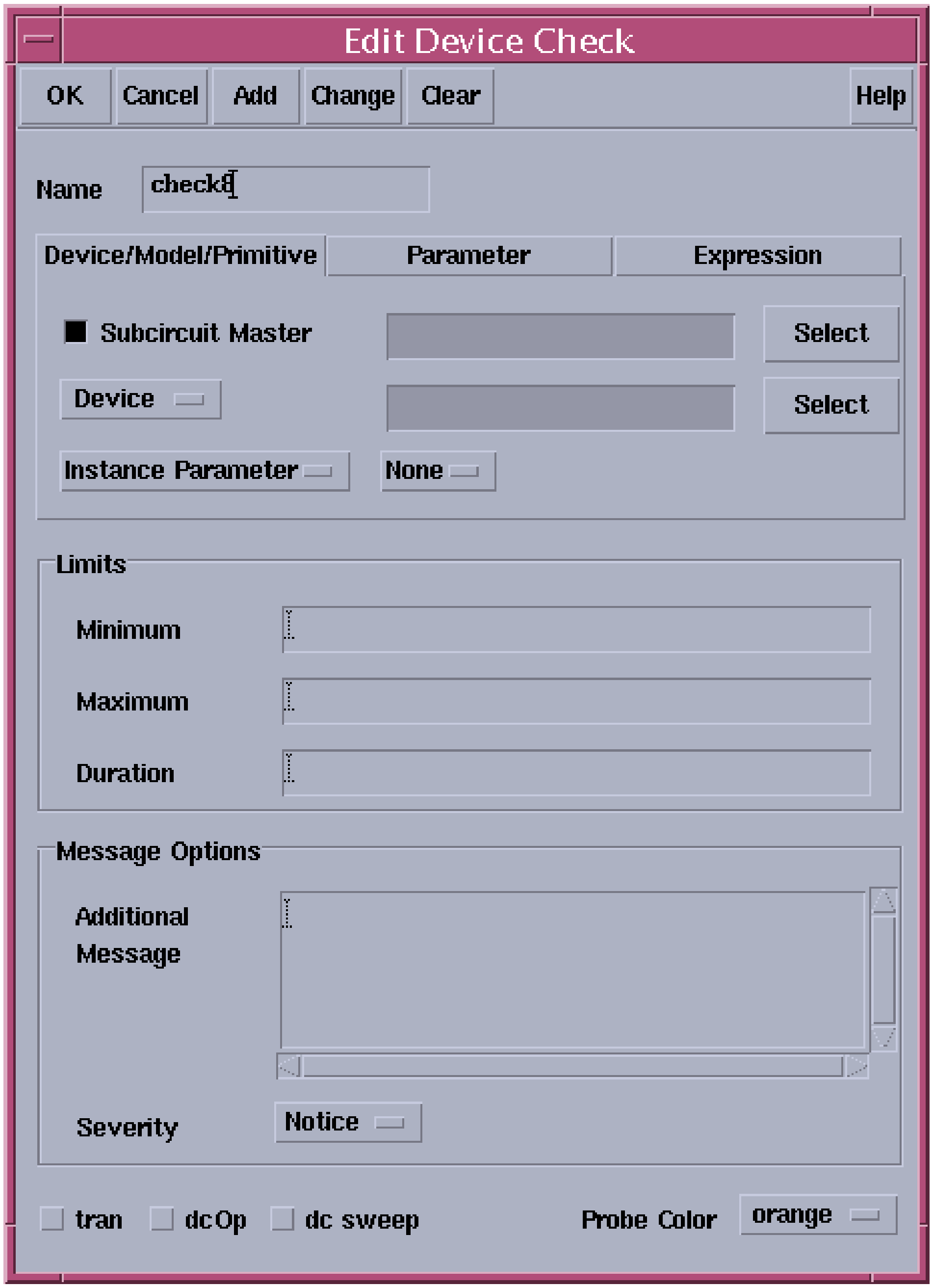
The Delete button is used to delete one or more asserts selected in the list.

The **Up/Down** buttons are used to move a selected assert one position up or one position down.

The **Options** button brings up a Device Checking Options form where you can specify options such as *start* and *stop* times and severity for a Transient, DC Op and DC analysis.

Editing Asserts

You can add new asserts or change a selected assert using the *Edit Device Check* form.



Name: check8

■ Subcircuit Master

Device

Instance Parameter: None

Severity: Notice

□ tran □ dcOp □ dc sweep

Probe Color: orange

You can create three types of asserts by clicking the corresponding tab page(*Device/ Primitive/Model*, *Parameter* or *Expression)*. Each tab page contains fields applicable for that type of assert.

The **Name** field, allows you to specify names for an assert.These asserts apply to Instances, Models or Primitives.

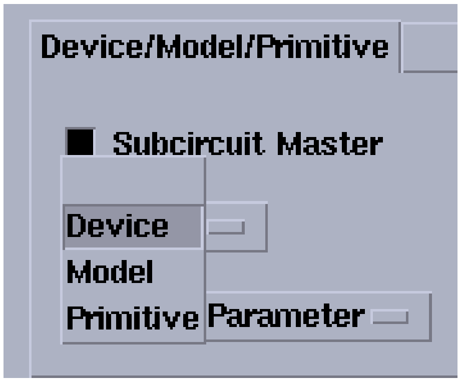
- The *Device/Primitive/Model* tab page

  The **Subcircuit Master** field is a boolean field. This field can be used to add `sub=subckt` to the assert statement. To the right side of this is a field that becomes active only when the *Subcircuit Master* button is enabled. You can click on the *Select* button and select an instance in the schematic window. The subcircuit master name for that instance is displayed in this field. In the following example, the text in bold is created by this field.

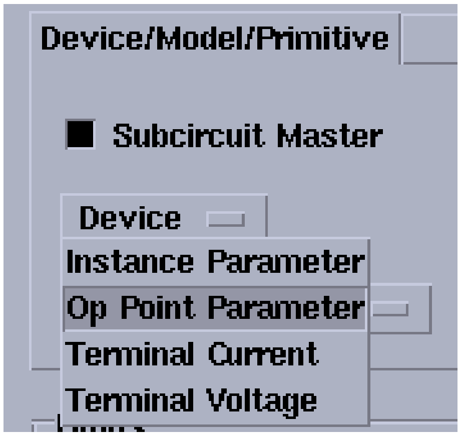
```
assert0 assert sub=myAmp dev=M1 param=Vgs mix=0.0 max=2.5
message=”Vgs....
```

  The **Device/Model/Primitive** is a cyclic field providing the options Device, Model and Primitive. If you select Device, click on the Select button to select a device in the schematic. If you select Model or Primitive, the tab page re-displays to show a cyclic field where you can select models (trpmos, trnpn, trpnp, trnmos) and primitives (bjt, isource, resistor, vsource, mos2, capacitor).

```
assert1 assert sub=myAmp dev=M1 param=Vgs min=0.0 max=2.5
message=”Vgs....
assert2 assert sub=myAmp mod=trnmos modelparam=Vtho  min=0.0
max=0.8 ....
assert3 assert sub=myAmp primitive=bsim3 param=Vgs  min=0.0
max=2.5 .....
```



The **Instance Parameter/Op Point Parameter/ Model Parameter/Terminal Current/ Terminal Voltage** field is a cyclic field. When any of these is selected, the tab page re-displays to show cyclic fields containing a corresponding list of parameters.

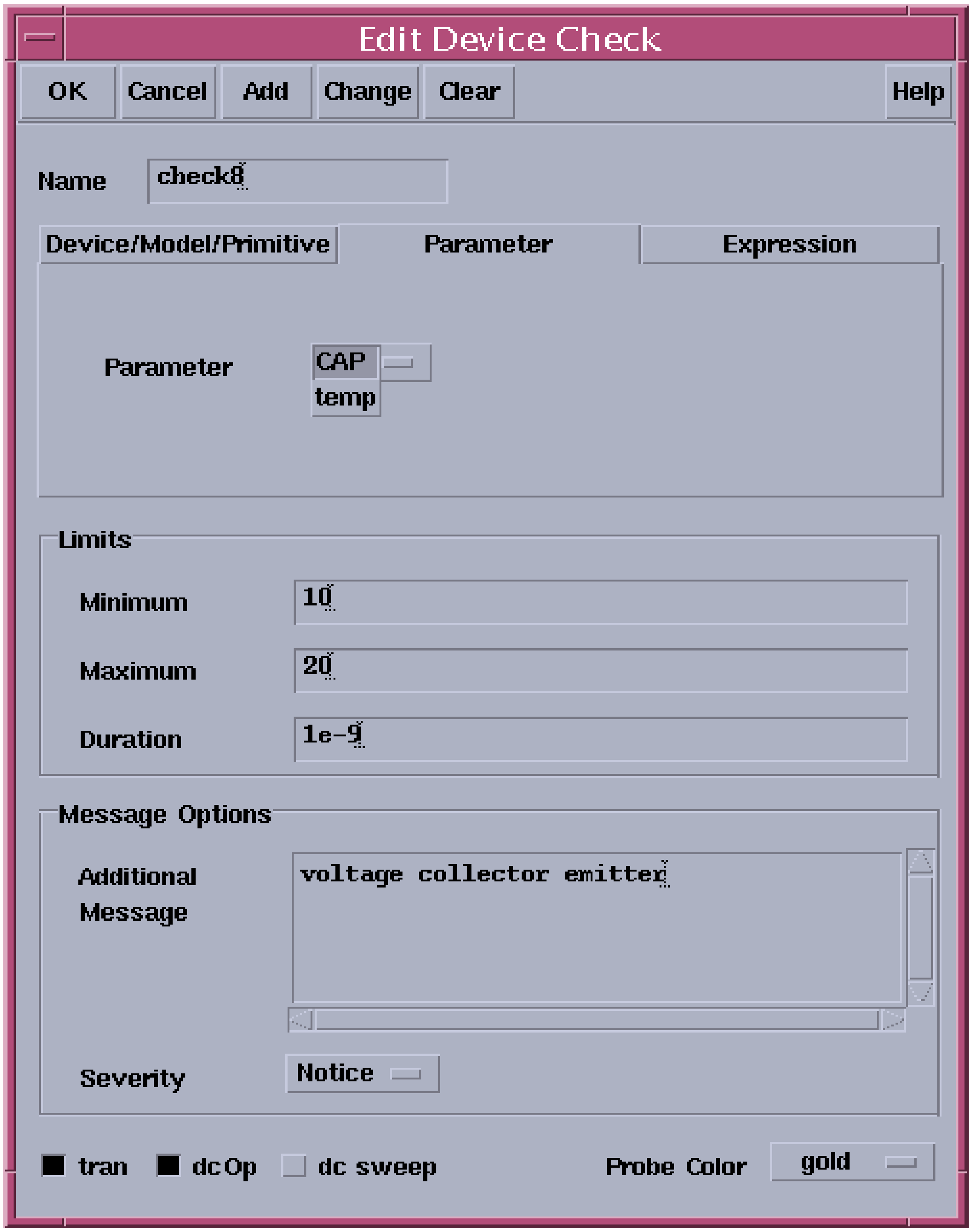


```
assert4 assert sub=myAmp dev=M1 param=Vgs min=0.0 max=2.5
message=”Vgs....
assert5 assert sub=myAmp mod=trnmos modelparam=Vtho  min=0.0
max=0.8 ....
```

- The *Parameter* page

  This is a cyclic field. You can select a top level netlist parameter whose value is to be checked. The choices for this cyclic field would be the design variables defined in the

Analog Design Environment plus the Spectre reserved parameters `temp, tnom, scale, scalem, freq, time`.



Name: check8

Parameter: CAP
temp

Limits

Minimum: 10

Maximum: 20

Duration: 1e-9

Message Options

Additional Message: voltage collector emitter

Severity: Notice

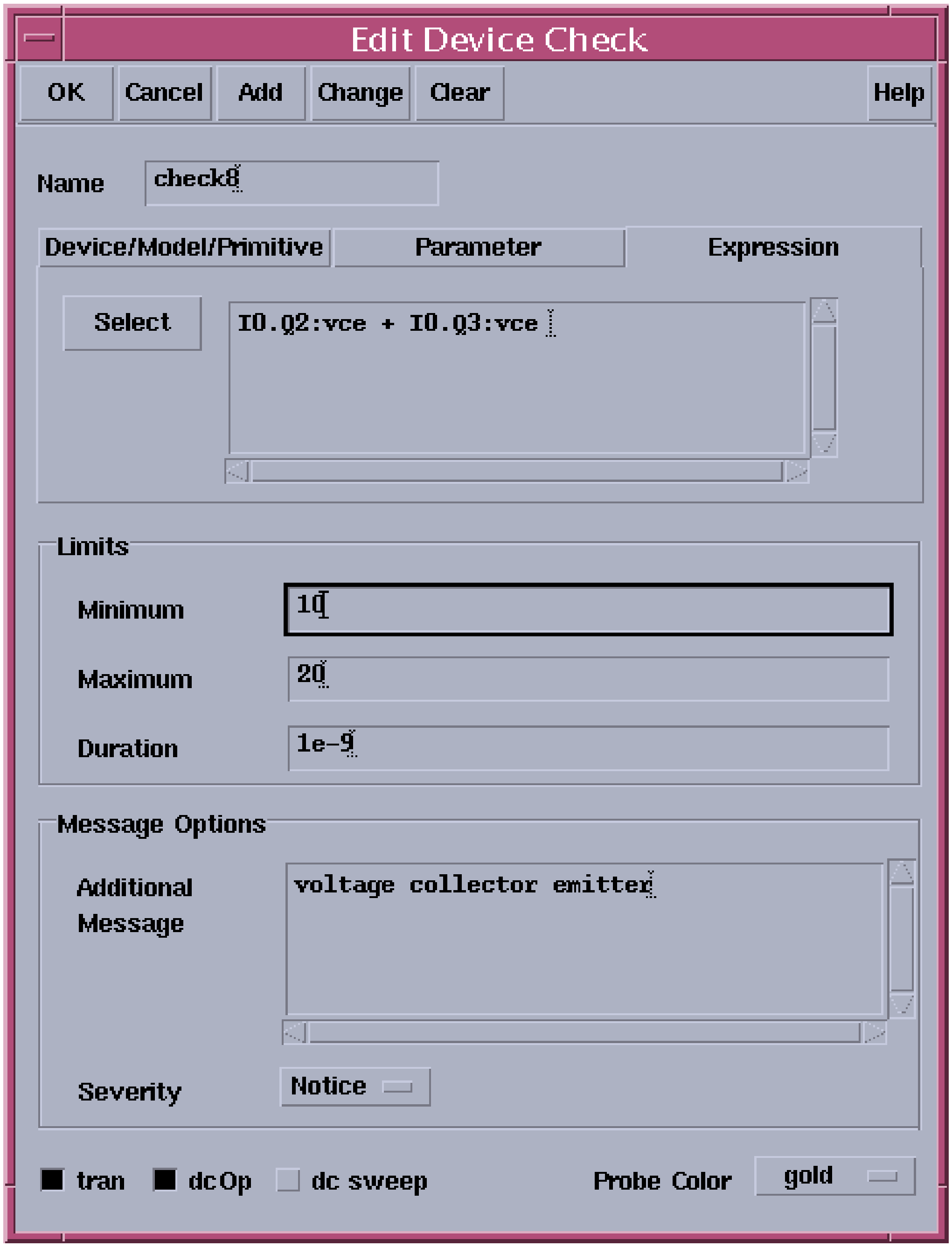
■ tran ■ dcOp □ dc sweep

Probe Color: gold

- The *Expression* page

  This page can be used to specify the three types of asserts mentioned earlier. The *expression* field is a text entry field. You can enter a *mdl* expression in this field. The *select* button is used to select a device in the schematic. When a device is selected a form appears in which you can select one of the displayed parameters or type in a name. The full schematic name for that instance followed by :paramname is appended to the existing text in the field. You can then add operators and functions to create an expression.

```
assert6 assert expr=(i1.x1:id - i2.x2:id ) min= - 0.001u
max=0.001u duration=1n
```



Name: check8

Expression: I0.Q2:vce + I0.Q3:vce

Limits

Minimum: 10

Maximum: 20

Duration: 1e-9

Message Options

Additional Message: voltage collector emitter

Severity: Notice

■ tran ■ dcOp □ dc sweep

Probe Color: gold

- The **Limit** section contains the *Minimum and Max*imum fields used to specify minimum and maximum values for the assert statements.
  `assert7 assert sub=myAmp dev=M1 param=Vgs` **`min=0.0 max=2.5`** `message=”Vgs..`
  The **Duration** field is a text field that can be used to specify a duration for the assert statement.
  `assert8 assert expr=(m1:ids > m2:ids) min=0.0 max=2.5` **`duration=1`**

  The **Message Options** section the Additional *Message* field, a text field that is used to specify the message to be printed when this assert fails. [`message=“message string”`]

  `assert9 assert sub=myAmp dev=M1 param=Vgs min=0.0 max=2.5` **`message=”Vgs`**

  It also contains the *Severity* field a cyclic field that is used to specify the error level. Choices are `Notice, Warning` and `Error and None`.

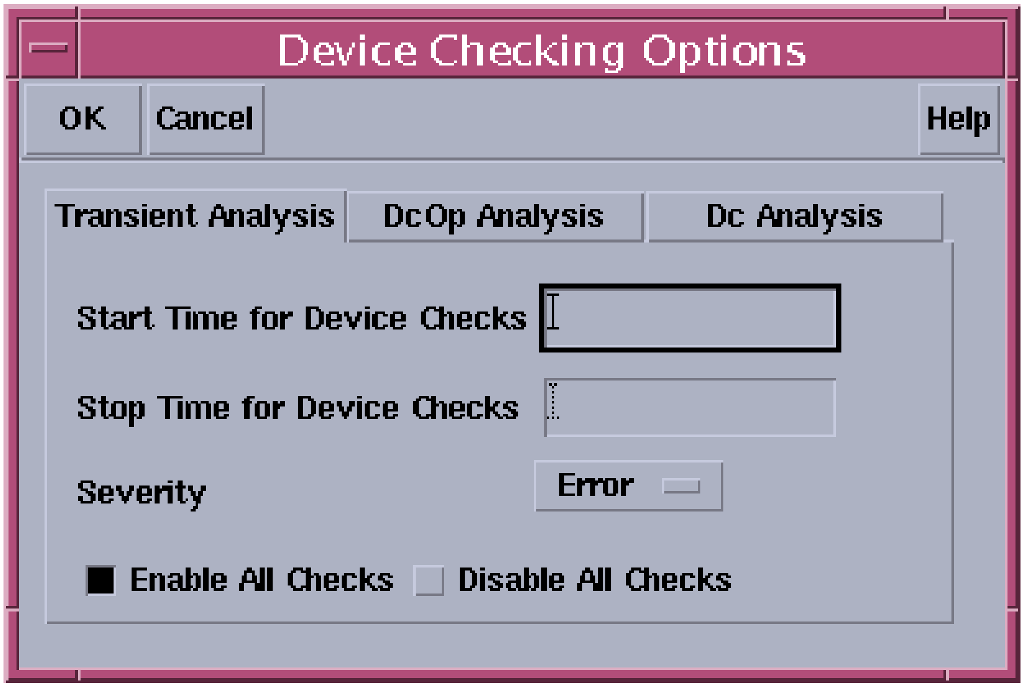
  `assert10 assert sub=myAmp dev=M1 param=Vgs min=0.0 max=2.5 message=”Vgs > 2.5”` **`level=’warn`**

The **tran**, **dcOp and dc sweep** buttons are used to specify if the assert will be enabled during dc and/or transient analysis. These buttons are disabled by default.

The **Probe Color** cyclic button can be used to specify colors for probes. You can change the default colors by choosing *Tools – Display Resource Tool Box* from the ICW and editing the properties of layers y0-y9.

Setting Options

You setup options using the following form:



Start Time for Device Checks

Stop Time for Device Checks

Severity: Error

■ Enable All Checks □ Disable All Checks

The Transient Analysis page is used to specify options such as start and stop times and severity of violations. Default value for severity fields is none.

```
ch1 checklimit enable=[assert1 assert2 ... assertn] start=1u
stop=2u
```

```
tran tran stop=1u write="spectre.ic" writefinal="spectre.fc"
```

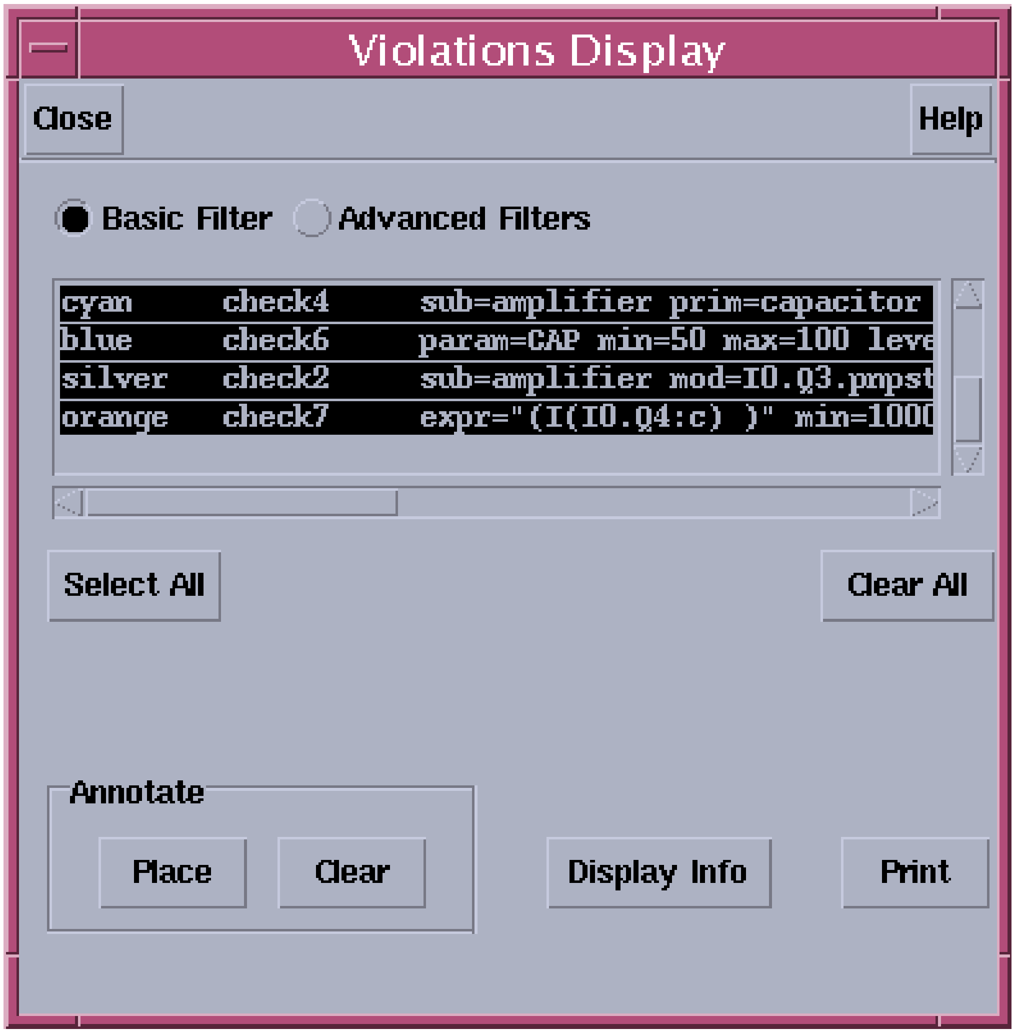
```
checklimit  enable=[assert1 assert2 assert5 assert6 assert7]
severity='notice'
```

```
dc dc oppoint=rawfile maxiters=150
```

The check boxes **Enable All Checks** and **Disable All Checks** are used to enable or disable all checks for transient or dc analysis.

Violations Display

Filters can be used to reduce the amount of violations messages displayed. This allows you to focus on critical parts of results and filter out the rest. The Violations Display form is used to display violations and to specify various filters to reduce the violation data that is displayed.



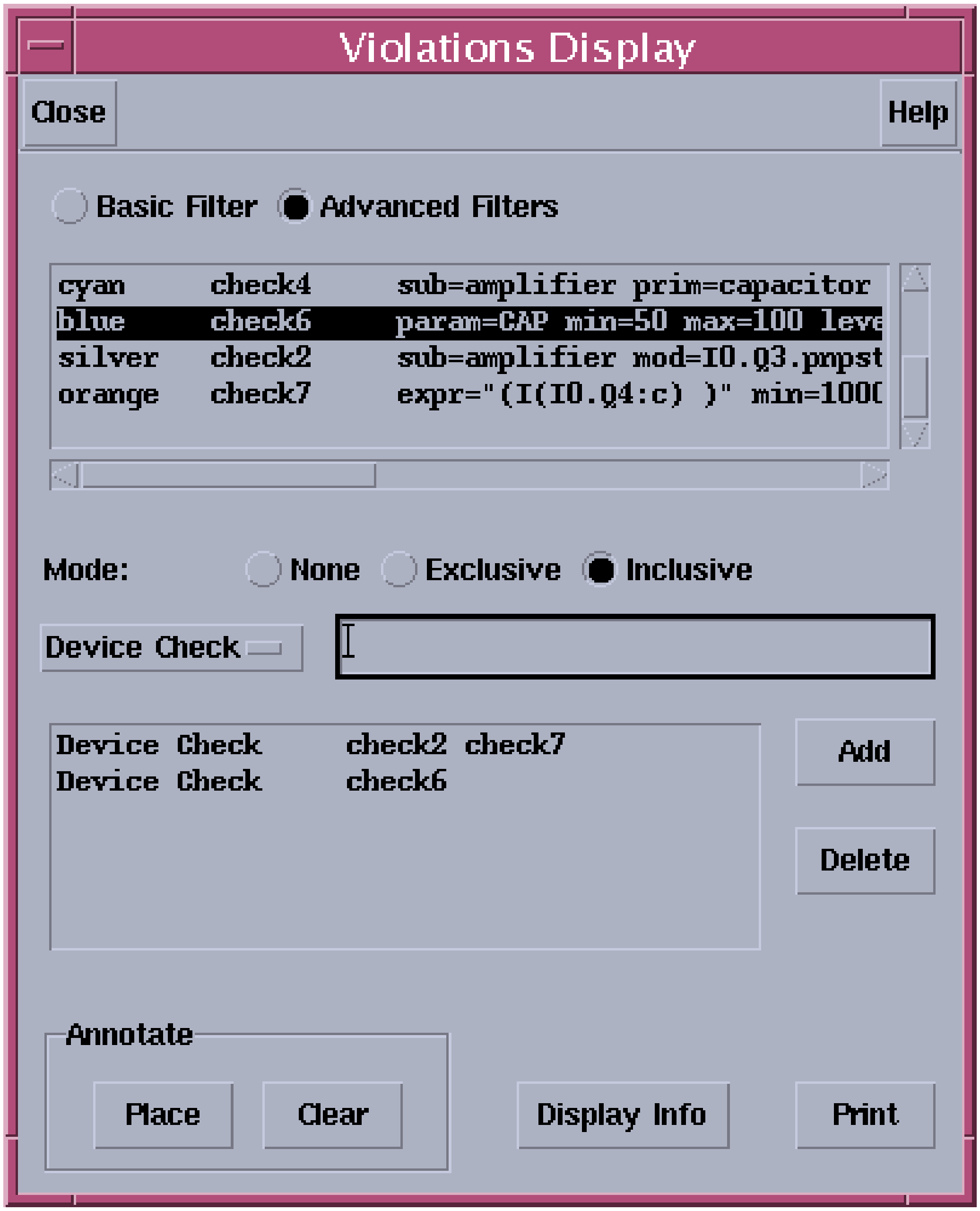
● Basic Filter ○ Advanced Filters

```
cyan     check4     sub=amplifier prim=capacitor
blue     check6     param=CAP min=50 max=100 leve
silver   check2     sub=amplifier mod=I0.Q3.pnpst
orange   check7     expr="(I(I0.Q4:c) )" min=1000
```

Asserts are displayed in the list box. These include asserts entered by you as well as asserts included through model files and other include files.

Two types of filters are available, Basic and Advanced. You can specify filters based on device names, device types, model names, device check names, error levels and the analysis during which a violation occurred.

The **Advanced** filter provides you with the option of filtering violations in the *Inclusive* or *Exclusive* mode.



○ Basic Filter ● Advanced Filters

```
cyan     check4     sub=amplifier prim=capacitor
blue     check6     param=CAP min=50 max=100 leve
silver   check2     sub=amplifier mod=I0.Q3.pnpst
orange   check7     expr="(I(I0.Q4:c) )" min=100
```

Mode: ○ None ○ Exclusive ● Inclusive

Device Check

```
Device Check     check2 check7
Device Check     check6
```

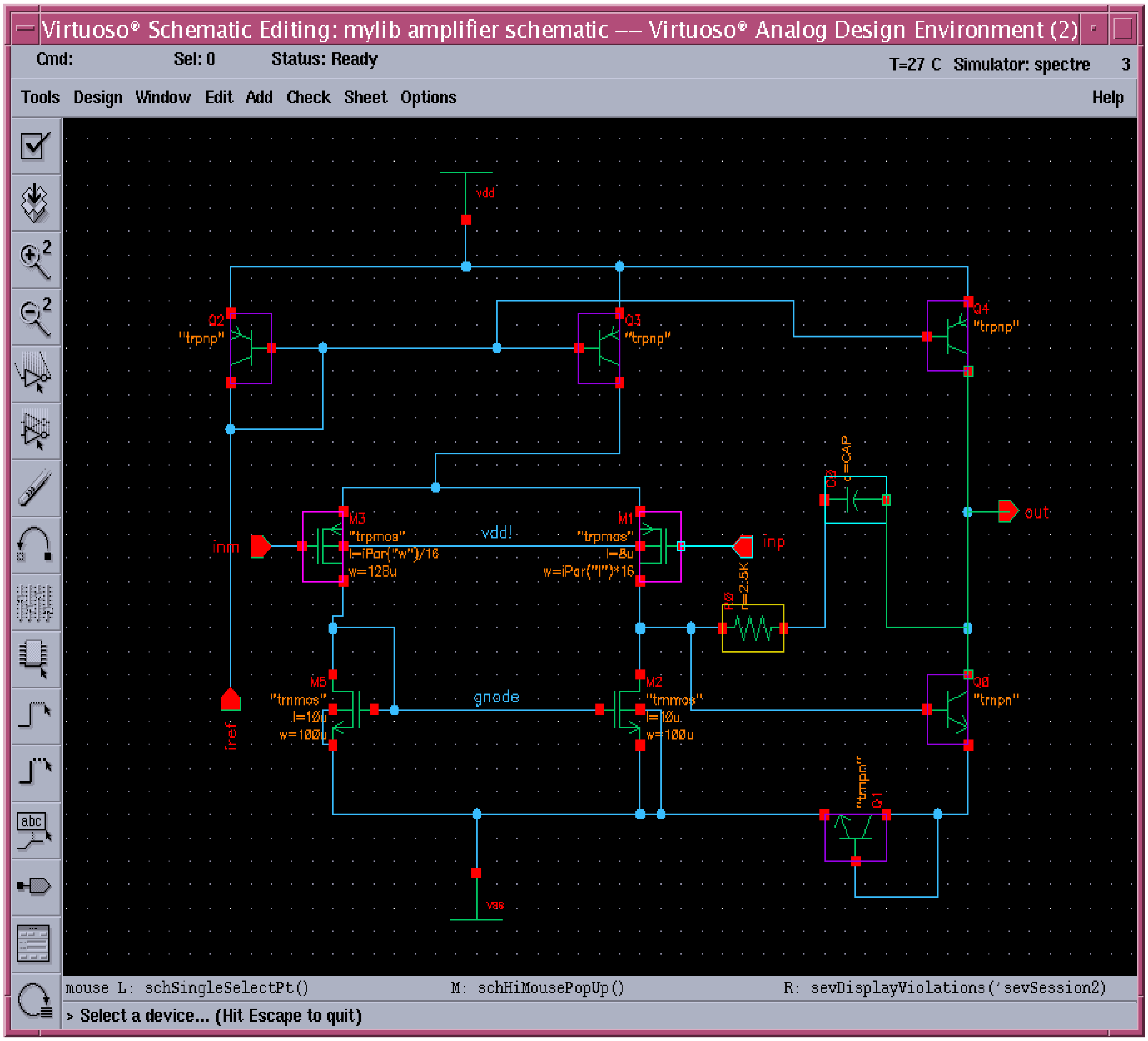
If the **Inclusive** radio button is enabled, all violations that match the filters are displayed.

If the **Exclusive** radio button is enabled, all violations that match the filter are filtered out. You can switch between the two filtering options.

Add button is used to add a filter to the list.

Delete button is used to delete one or more filters from list.

Place button when pressed, opens the schematic and highlights all the devices that failed the asserts. You can change the default colors by choosing *Tools – Display Resource Tool Box* from the ICW and editing the properties of layers y0-y9.



Virtuoso® Schematic Editing: mylib amplifier schematic –– Virtuoso® Analog Design Environment (2)

Cmd: Sel: 0 Status: Ready T=27 C Simulator: spectre 3

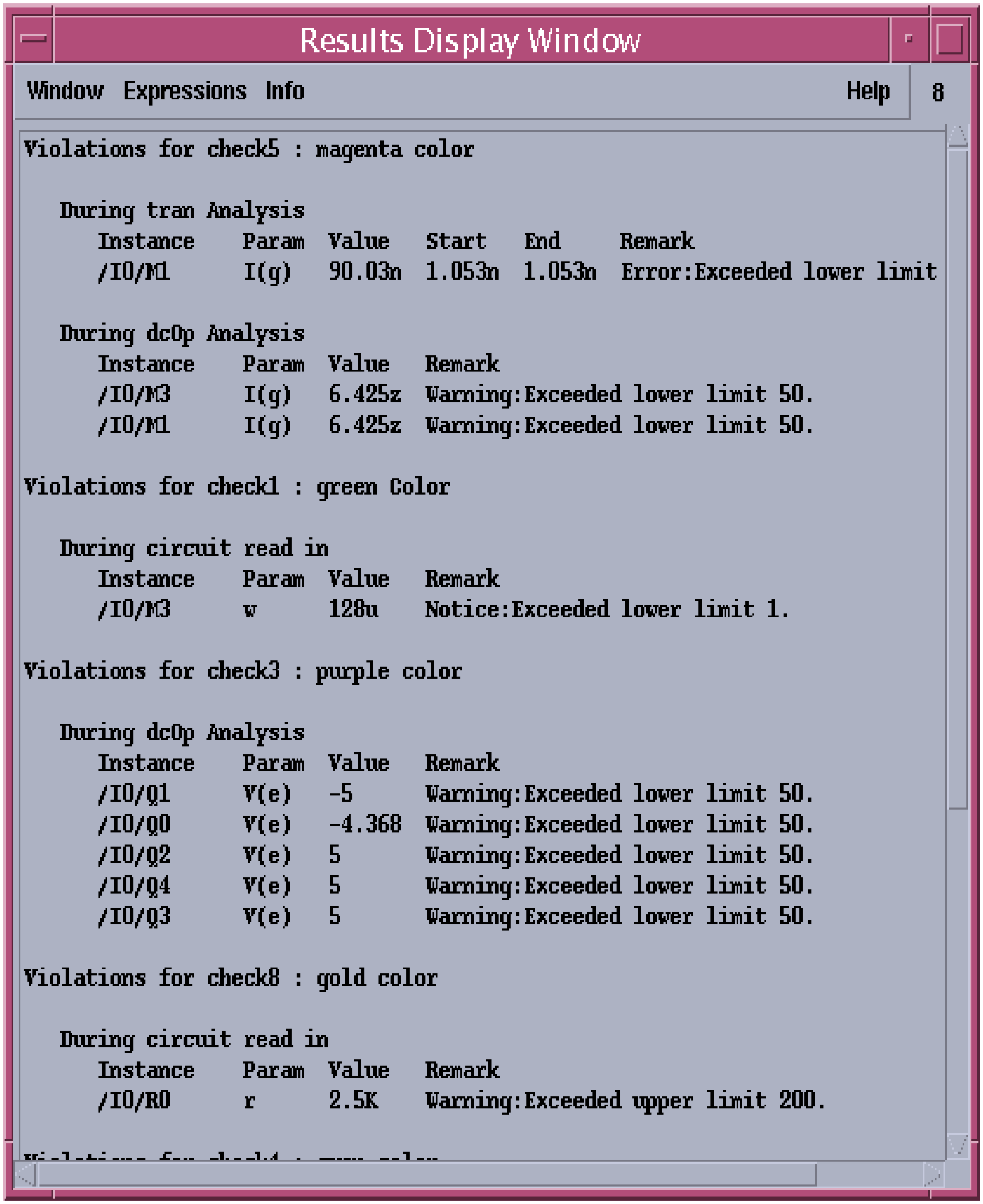
mouse L: schSingleSelectPt() M: schHiMousePopUp() R: sevDisplayViolations('sevSession2)

> Select a device... (Hit Escape to quit)

Clear button clears the annotations in schematic window.

Display Info display the violation information for any device highlighted in the schematic. You have to click on this button and then select one or more highlighted devices.

Print prints a summary of all the violations.



```
Violations for check5 : magenta color

   During tran Analysis
      Instance    Param  Value   Start   End     Remark
      /I0/M1      I(g)   90.03n  1.053n  1.053n  Error:Exceeded lower limit

   During dcOp Analysis
      Instance    Param  Value   Remark
      /I0/M3      I(g)   6.425z  Warning:Exceeded lower limit 50.
      /I0/M1      I(g)   6.425z  Warning:Exceeded lower limit 50.

Violations for check1 : green Color

   During circuit read in
      Instance    Param  Value   Remark
      /I0/M3      w      128u    Notice:Exceeded lower limit 1.

Violations for check3 : purple color

   During dcOp Analysis
      Instance    Param  Value   Remark
      /I0/Q1      V(e)   -5      Warning:Exceeded lower limit 50.
      /I0/Q0      V(e)   -4.368  Warning:Exceeded lower limit 50.
      /I0/Q2      V(e)   5       Warning:Exceeded lower limit 50.
      /I0/Q4      V(e)   5       Warning:Exceeded lower limit 50.
      /I0/Q3      V(e)   5       Warning:Exceeded lower limit 50.

Violations for check8 : gold color

   During circuit read in
      Instance    Param  Value   Remark
      /I0/R0      r      2.5K    Warning:Exceeded upper limit 200.
```

8

Helping a Simulation to Converge

This chapter describes how you can help a troublesome simulation to converge. Select topics from the following list to view more information.

Commands for Forcing Convergence

You use the commands in the *Simulation – Convergence Aids* menu to help the simulator find a solution when it fails to achieve convergence. Once you get the simulation to converge, you can save the DC and Transient solutions. When you resimulate, you can save time by restoring the saved solutions.

There are three commands to help the simulator find a solution:

- *Node Set*, which provides an initial guess for nodes in any DC analysis or the initial condition calculation for the transient analysis
- *Initial Condition*, which provides initial conditions for nodes in the transient analysis

- *Force Node*, which sets the voltage on a node and locks it at that voltage during the entire simulation

Note: Refer to the simulator manual for specific details about the commands that help circuits converge. Not all simulators support these three commands, and simulators implement these methods differently.

Node Set

To set an initial DC voltage on selected nodes, use the *Simulation – Convergence Aids – Node Set* command.

For Spectre, the node set is used to provide an initial guess for nodes in any DC analysis or the initial condition calculation for the transient analysis. It netlists to

```
nodeset node=value
```

For more information, see the *Spectre Circuit Simulator Reference* manual.

For other simulators, the *Node Set* command is equivalent to

```
.NODESET v(node)=value
```

Initial Conditions

To set an initial transient voltage on selected nodes, use the *Simulation – Convergence Aids – Initial Condition* command.

For Spectre, initial conditions are used to provide initial conditions for nodes in the transient analysis. Initial conditions are accepted only for inductor currents and node voltages where the nodes have a path of capacitors to ground. This is netlisted to

```
ic node=value
```

For more information, see the *Spectre Circuit Simulator Reference* manual.

For other simulators, the *Initial Condition* command is equivalent to

```
.IC node=value
```

Force Node

To set a node to a specific voltage throughout the simulation, use the *Simulation – Convergence Aids – Force Node* command.

For details on this command, see the reference manual for the simulator that you use.

One way to use this feature is to store the DC solution from a simulation with *Force Node* active, remove the *Force Node* setting, restore the DC solution, and run another simulation.

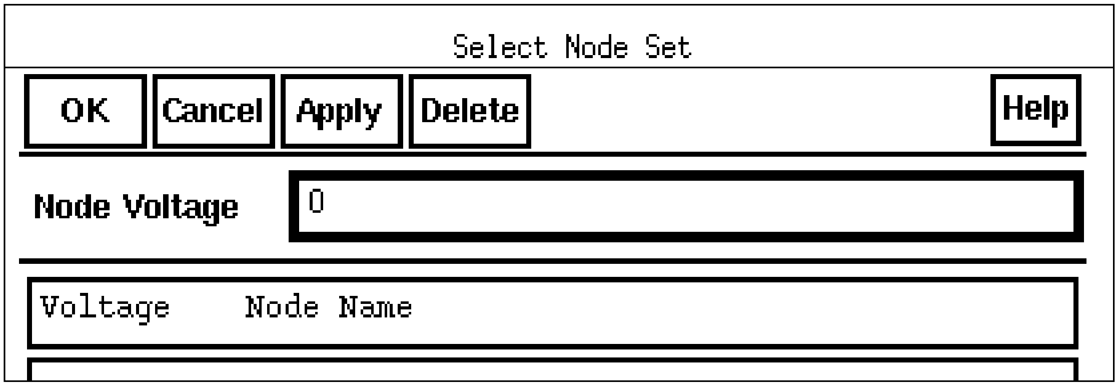
Note: The Spectre simulator and some other simulators do not support this command.

Selecting Nodes and Setting Their Values

To select a node and set its voltage,

1. Choose *Simulation – Convergence Aids* and a convergence command (*Node Set, Initial Condition,* or *Force Node*).

   A form appears to enter the voltage. Each command displays a different form. The Select Node Set form is shown here.



Select Node Set

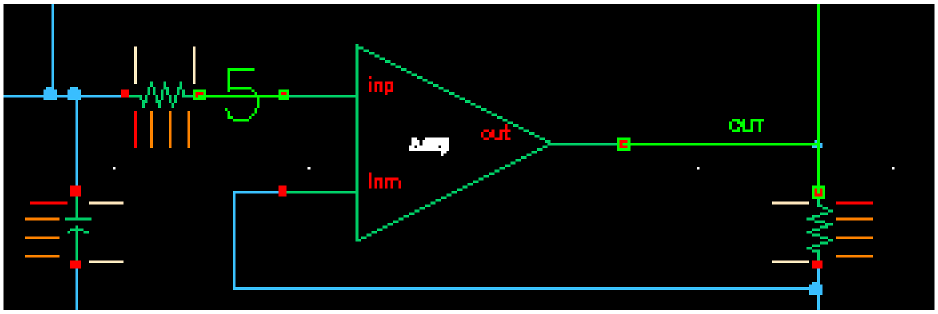
Node Voltage: 0

Voltage   Node Name

2. Type the voltage.
3. Click in the Schematic window to select the first node.

   The node name appears in the form.

In the Schematic window, the node is highlighted and the voltage appears. For split nets, the system labels only the driving cell.



4. To set other nodes to the same voltage, select them.
5. To set other nodes to a different voltage, change the voltage, and select other nodes.
6. Click *OK* or *Cancel* when you are finished selecting nodes.

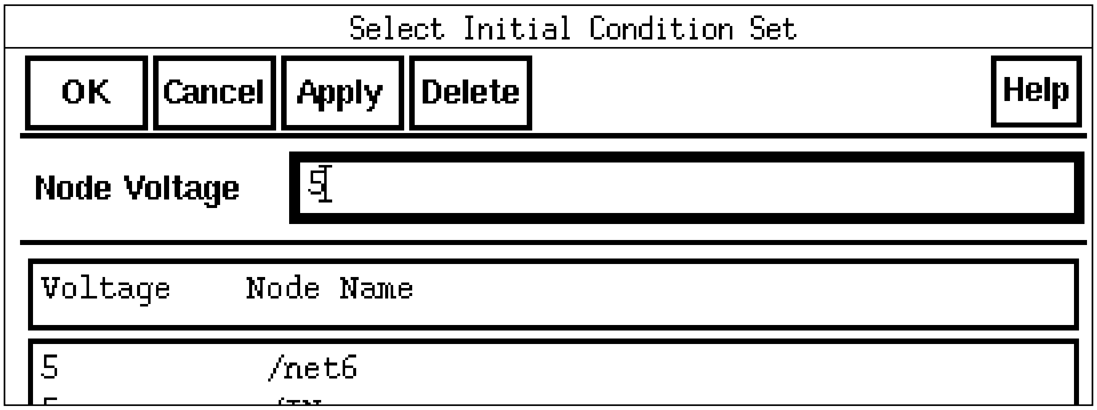
You can select nodes at any level of the hierarchy. When you select an interface node at a lower level, the node is highlighted, but the voltage value appears only on the higher-level schematic.

Releasing Voltages

To release the node set, initial condition, or force node voltage settings,

1. Choose *Simulation – Convergence Aids* and a convergence command (*Node Set*, *Initial Condition*, or *Force Node*).

   A form appears listing all of the nodes that have been set. The Select Initial Condition Set form is shown here.



Node Voltage 5

```
Voltage     Node Name
5             /net6
```

2. Click on the net in the schematic to release it, or click on the net name in the form and click *Delete*.

   On the form, you can select several nodes to release by holding down the left mouse button and dragging through the list box.

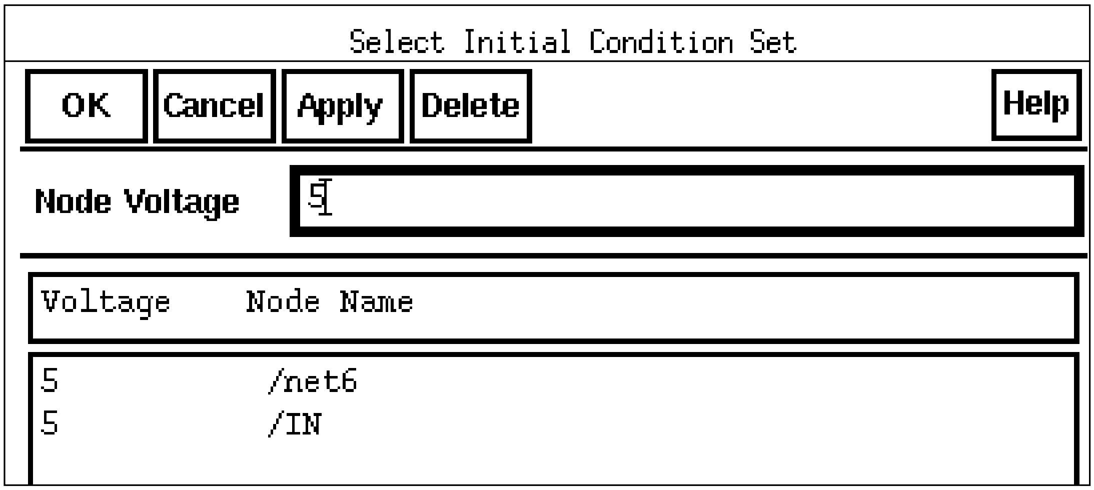
3. Click *OK* or *Cancel* when you are finished releasing nodes.

Changing Voltages

To change the value of the voltage set on a node,

1. Choose *Simulation – Convergence Aids* and a convergence command (*Node Set*, *Initial Condition*, or *Force Node*).

   A form appears listing all of the nodes that have been set. The Select Initial Condition Set form is shown here.



Node Voltage 5

Voltage	Node Name
5	/net6
5	/IN

2. Click on the node name to highlight it.
3. Type the new voltage value, and click *Apply*.
4. Click *OK* or *Cancel* when you are finished changing voltages.

Saving and Restoring Node Voltages

To save a list of nodes and their node set, initial condition, or force node voltage settings,

1. In the Simulation window, choose *Session – Save State*.

The Saving State form appears.

2. Type in a name for the saved simulation state.
3. Check that the *Convergence Setup* box is selected, and click *OK*.

To restore the saved settings,

1. Choose *Session – Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Choose a run directory with the *Cell* and *Simulator* fields.

   The list box shows the saved states for the cell and simulator combination.

3. Click on a *State Name*.
4. Check that *Convergence Setup* is selected, and click *OK*.

Highlighting Set Nodes

While the *Select* commands are active, the system highlights selected nodes and labels them with the voltages you set. After you close the form, the system removes the highlighting and the labels.

To redisplay the highlighting and labels,

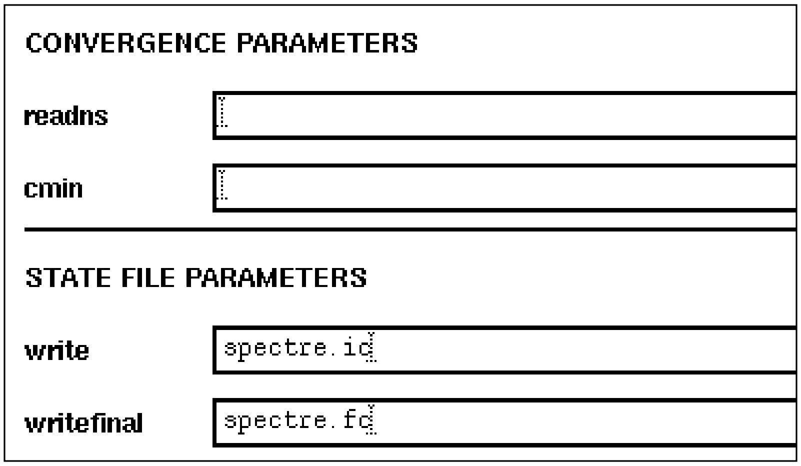
➤ Choose *Simulation – Convergence Aids* and a convergence command (*Node Set*, *Initial Condition*, or *Force Node*).

To remove the highlighting and labels,

➤ Close the active Select form.

Storing a Solution

To store a solution for the Spectre simulator, use the *write* and *writefinal* fields in the *State File Parameters* section of any analysis options form.



CONVERGENCE PARAMETERS

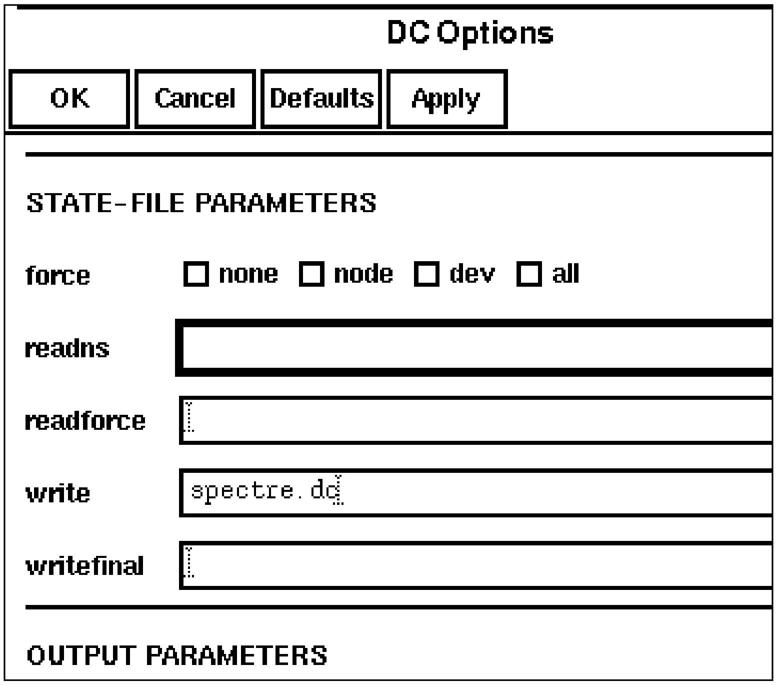
readns	
cmin	

STATE FILE PARAMETERS

write	spectre.ic
writefinal	spectre.fc

This causes the Spectre simulator to write out the initial and final conditions of a transient analysis. By default, the conditions are written to the files `spectre.ic` and `spectre.fc` in the netlist directory.

To store the solution of an analysis, use the *write* and *writefinal* fields of the DC Options form for the analysis.



STATE-FILE PARAMETERS

force □ none □ node □ dev □ all

readns

readforce

write spectre.dc

writefinal

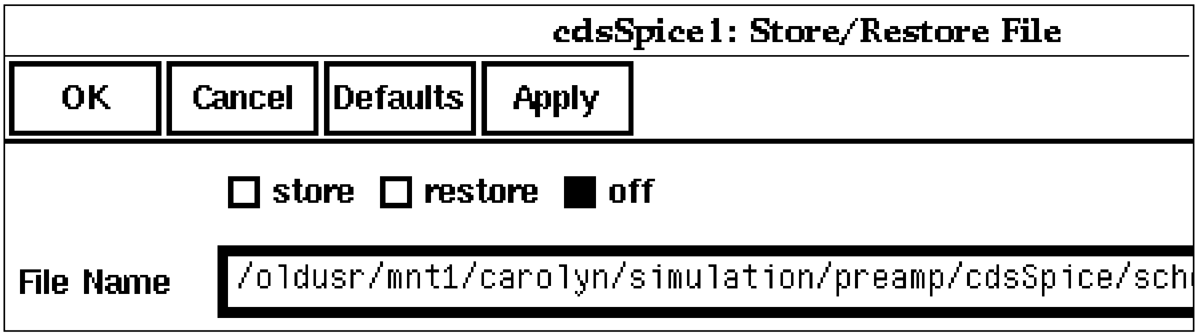
OUTPUT PARAMETERS

Note: This does not apply to spectre or spectreS. Use the analysis options instead.

To store a DC or Transient solution to restore later,

1. After simulation, choose *Simulation – Convergence Aids* and either *Store/Restore* or *Transient Store/Restore*.

   The Store/Restore File form appears.



□ store □ restore ■ off

File Name /oldusr/mnt1/carolyn/simulation/preamp/cdsSpice/sch

   For detailed information about the form, see “Store/Restore File” on page 379.

2. Choose *store*.

3. If you want to restore this solution the next time you simulate, also choose *restore*.
4. Check the filename and change it if necessary.

   The system writes the solution to the file you specify. The default file is

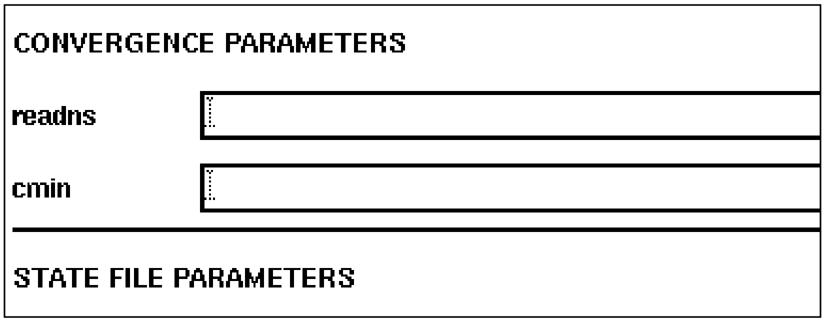
   `project_dir/design/simulator/storeRestoreFile`

   The system does not overwrite the file the next time you simulate. To save a new solution, you must explicitly store again.

   For spectreS, use the Spectre *write* and *writefinal* analysis options.

Restoring a Solution for Spectre

To restore a saved transient solution, use the *readns* field in the *Convergence Parameters* section of the transient options form. Use the name of the file that was previously used in the *write* or *writefinal* sections of the same form.



CONVERGENCE PARAMETERS

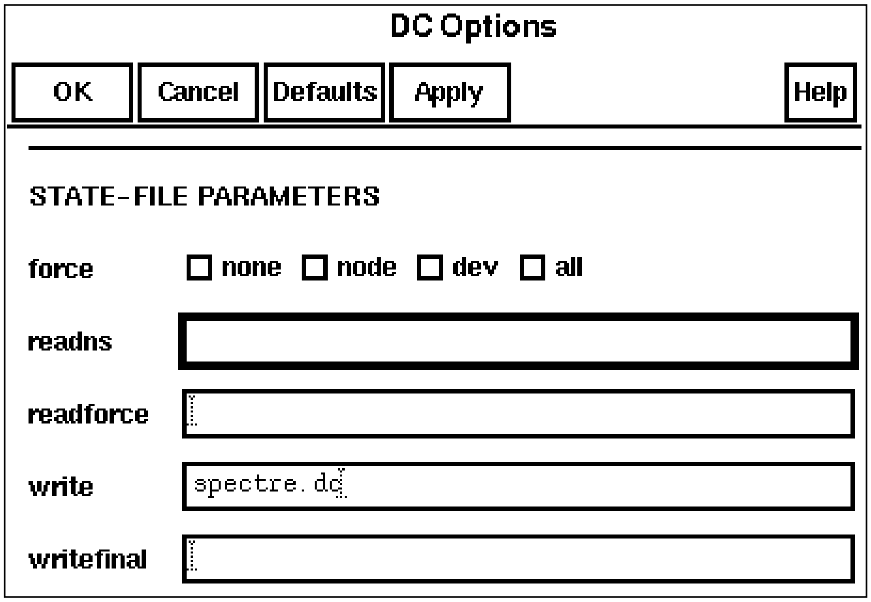
readns

cmin

STATE FILE PARAMETERS

To stop using the solution, clear the field.

To restore a saved DC solution, use the *readns* field in the *State-File Parameters* section of the DC Options form. Use the name of the file that was previously used in the *write* or *writefinal* sections of the same form.



STATE-FILE PARAMETERS

force □ none □ node □ dev □ all

readns

readforce

write spectre.dc

writefinal

To stop using the solution, clear the field.

Restoring a Solution for cdsSpice

To restore a saved DC or Transient solution,

1. Choose *Simulation – Convergence Aids* and either *Store/Restore* or *Transient Store/Restore*.
2. Choose *restore* and enter the filename.

   The solution will be restored for the next and all subsequent simulations, until you turn off the restore feature.

To stop using the restored solution,

➤ Choose *off* in the Store/Restore File form.

This does not delete the stored solution.

For spectreS, use the Spectre *readns* and *readic* analysis options.

Form Field Descriptions

Store/Restore File

store stores the DC analysis node voltages in the file specified in the *File Name* field.

restore restores the DC analysis node voltages from the file specified in the *File Name* field.

off turns off the *store* and *restore* buttons.

File Name is the name of the file into which the node voltages are saved. The default filename is `storeRestoreFile`*.*

9

Analysis Tools

This chapter contains information about the advanced analysis tools available in the Virtuoso® Analog Design Environment.

About Parametric Analysis

The parametric analysis feature (parametric plotting) lets you assign values to components and other parameters in a circuit and sweep the circuit over the ranges of specified values. Parametric analysis can be a useful tool during the design phase of a circuit or during verification and lets you specify ranges and pairs of values for components, semiconductor parameters, and other circuit parameters, and then analyze the circuit over these specified values. This is called sweeping parameters. The values you sweep in a parametric analysis must be identified as variables, as opposed to fixed values, on the schematic.

Parametric analysis, when used with the display features of the waveform window (WaveScan or AWD) the waveform calculator, and the Results Browser, enables you to see the effect of systematically altering circuit values.

For example, after running a parametric analysis, you can plot a group of curves for any waveform object in the netlist in a single display window. Each curve represents the results for a particular value in the sweep range, and you can compare the different curves to choose the best value.

Sweeps on Multiple Variables

If you sweep more than one variable in a parametric analysis, the first variable (Sweep 1 position) is assigned its first value while subsequent variables (Sweep 2, Sweep 3, …) cycle through all their values. The first variable then moves to its next value, and the subsequent variables again cycle through all their values, and so on.

For example, you might set these sweep specifications.

- Sweep values .01 and .02 for the sweep 1 variable
- Sweep values .03 and .04 for the sweep 2 variable
- Sweep values .05 and .06 for the sweep 3 variable

With these specifications, you perform eight analyses as shown below:

	Sweep 1 value	**Sweep 2 value**	**Sweep 3 value**
Analysis 1	.01	.03	.05
Analysis 2	.01	.03	.06
Analysis 3	.01	.04	.05
Analysis 4	.01	.04	.06
Analysis 5	.02	.03	.05
Analysis 6	.02	.03	.06
Analysis 7	.02	.04	.05
Analysis 8	.02	.04	.06

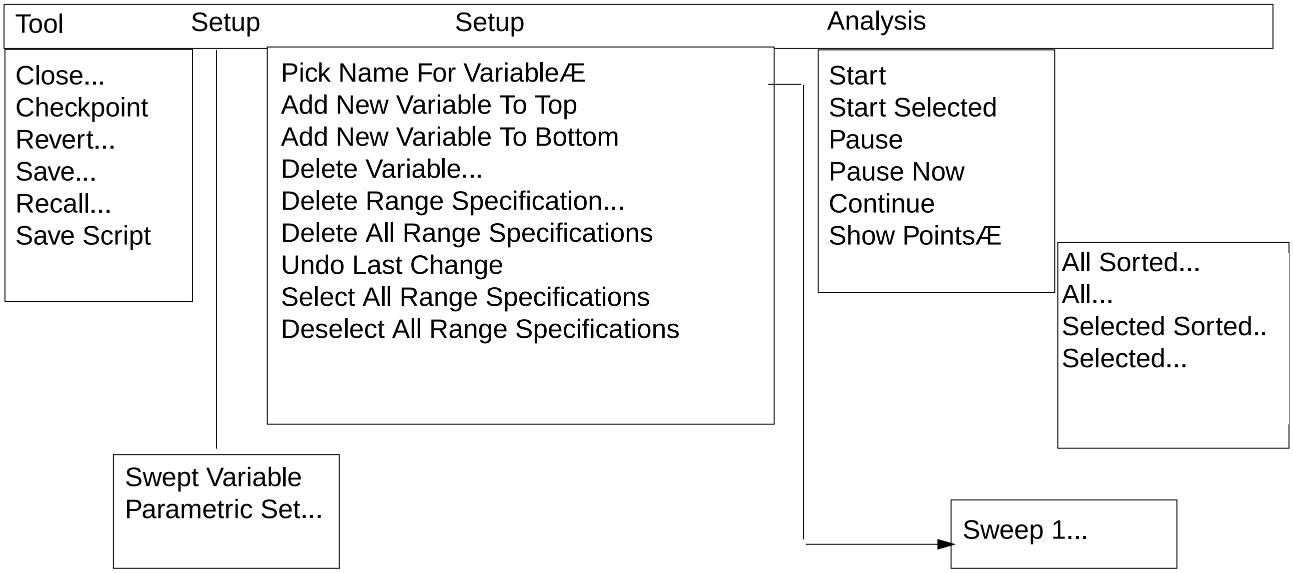
Overview of Analysis Specification

The following is an overview of the steps required to perform a parametric analysis. Click on any highlighted area to go to more information about a particular step or feature.

- In most cases, you perform a simulation of a circuit before you use parametric analysis. This practice helps ensure that the simulation runs successfully before the parametric analysis runs multiple simulations.
- Call up the Parametric Analysis window to start a parametric analysis.

- Specify the sweep variables, the sweep ranges, and the step values for a parametric analysis run by filling in values in the Parametric Analysis window.
- View your specifications before you run the analysis by examining the selected points (optional).
- Save the parametric analysis specifications to either temporary or permanent storage, and then recall the specifications later (optional).
- At runtime, select specifications for that run (optional). You can interrupt and restart a parametric analysis run.
- Finally, plot the results of the analysis.

The Parametric Analysis window has the following menu options.

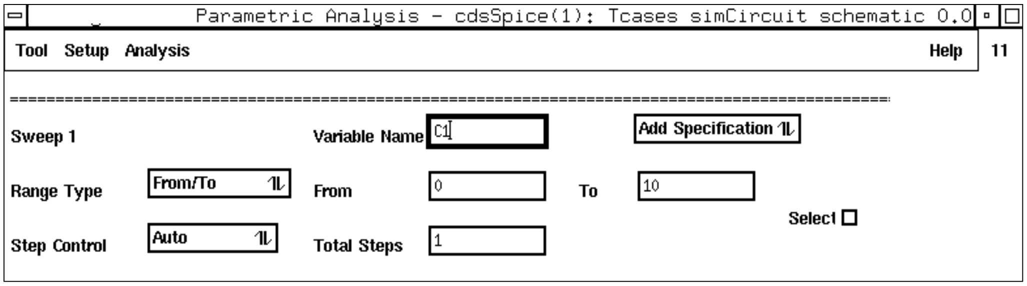


Getting Started with Parametric Analysis

You can use the following procedure to call up the Parametric Analysis window:

➤ Choose *Tools – Parametric Analysis* in the Virtuoso® analog circuit simulation window.

The Parametric Analysis window appears. You use this window to specify values for the parametric analysis. You can enter many specifications, and you can choose options from three main menus at the top of the window. These menus are *Tool*, *Setup*, and *Analysis*.


Parametric Analysis - cdsSpice(1): Tcases simCircuit schematic 0.0
Tool
Setup
Analysis
Help
11
Add Specification

Sweep 1

		Variable Name	C1		
Range Type	From/To	From	0	To	10
					Select □
Step Control	Auto	Total Steps	1		

Caution

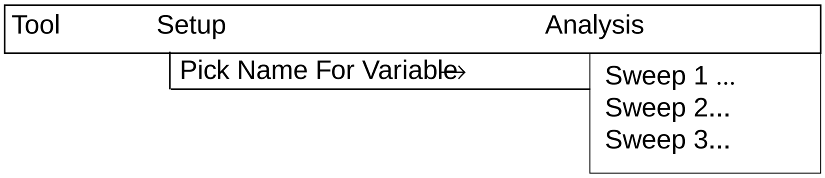
Parametric analyses are interactive. That is, if you make a change to your design during the simulation the design is re-netlisted. Data in the successive runs (after the design change) may be different from results of previous runs.

Specifying Sweep Variables

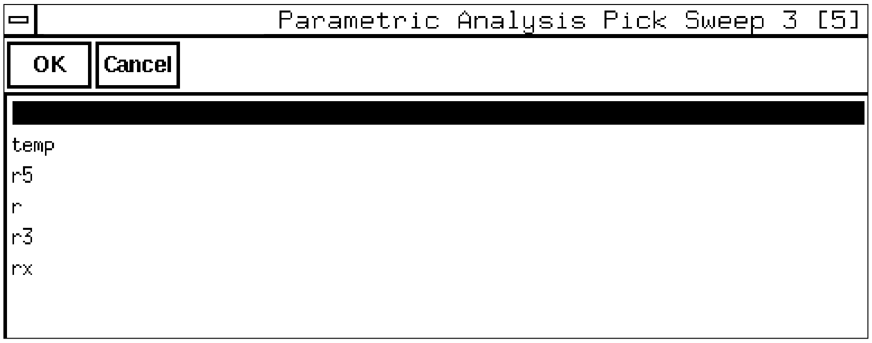
To specify a variable,

1. Choose *Setup – Pick Name For Variable – Sweep N* in the Parametric Analysis window.

   If you are choosing your first variable, *Sweep 1* is the only choice. If you have already specified sweep variables, a menu item appears for each sweep variable, and you can change the specifications for any of these variables.



A Pick Sweep window displays a list of variables you can sweep.

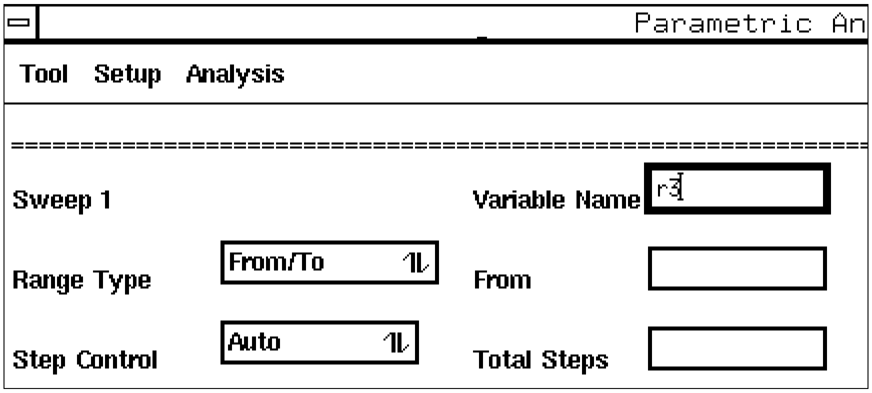


temp
r5
r
r3
rx

Note: Components or parameters that have fixed values in the original simulation do not appear in the Pick Sweep window.

2. Click on a variable in the list in the Pick Sweep window, and click *OK* on the window.

   The window disappears and the name of the variable appears in the *Variable Name* field of the Parametric Analysis window.



Sweep 1
Variable Name r3
Range Type From/To
From
Step Control Auto
Total Steps

You can also type a valid variable name directly into the *Variable Name* field.

Note: You can sweep the temperature in a parametric analysis. Choose *Tools – Parametric Analysis* in the *Virtuoso Analog Design Environment* window. The *Parametric Analysis* window appears. You use this window to specify values for the parametric analysis. Specify `temp` as the *Variable Name*. Also specify the *sweep range* and *number of steps*. The `temp` variable is built-in; therefore you do not have to add your own design variables.

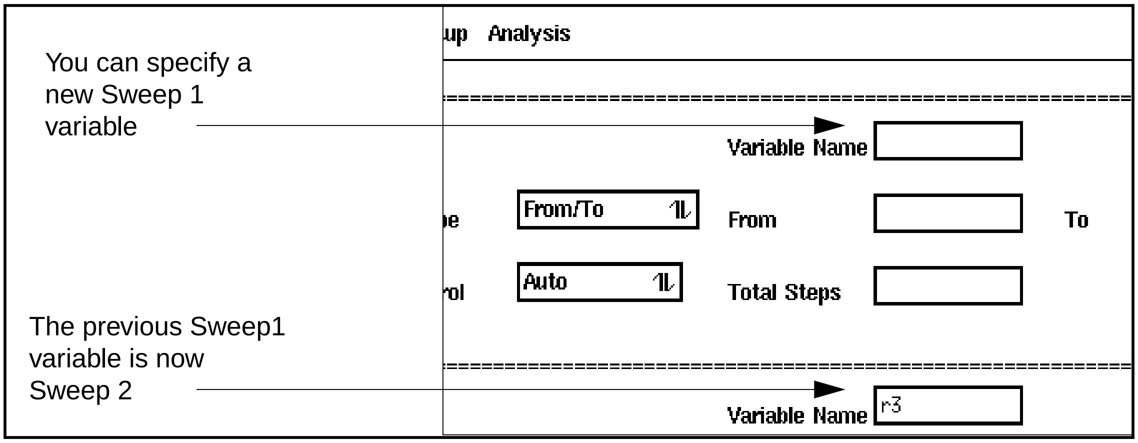
Adding Variables

You can add a new variable either as a Sweep 1 variable or as a Sweep *n* variable, where *n* is the highest number of sweeps enabled. If you add a new Sweep 1 variable, the other variables move down one number. See the overview for more information about the sweep levels in parametric analysis.

To specify a new Sweep 1 variable,

➤ Choose *Setup – Add New Variable To Top* in the Parametric Analysis window.

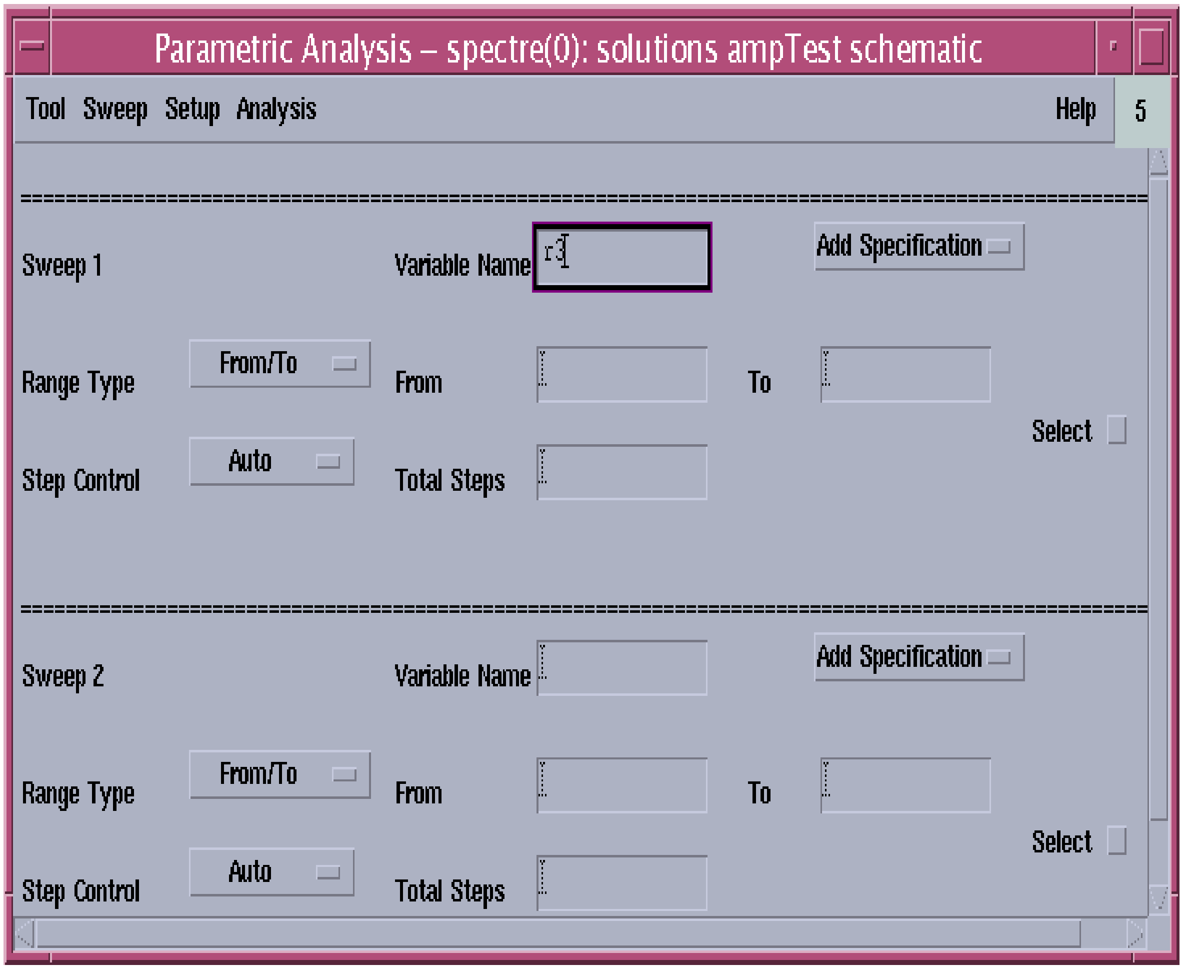
The window changes to let you specify a new Sweep 1 variable. (In this example, `r3` was the previous Sweep 1 variable.)


You can specify a new Sweep 1 variable
up Analysis
Variable Name
e
From/To
From
To
rol
Auto
Total Steps
The previous Sweep1 variable is now Sweep 2
Variable Name
r3

To specify a new Sweep *n* variable,

➤ Choose *Setup – Add New Variable To Bottom* in the Parametric Analysis window.

The window changes to let you add a new variable at the bottom of the window. (In this example, a new Sweep 2 variable is added below `r3`, the Sweep 1 variable.)



Parametric Analysis – spectre(0): solutions ampTest schematic

Sweep 1 Variable Name r3
Range Type From/To From To
Step Control Auto Total Steps

Sweep 2 Variable Name
Range Type From/To From To
Step Control Auto Total Steps

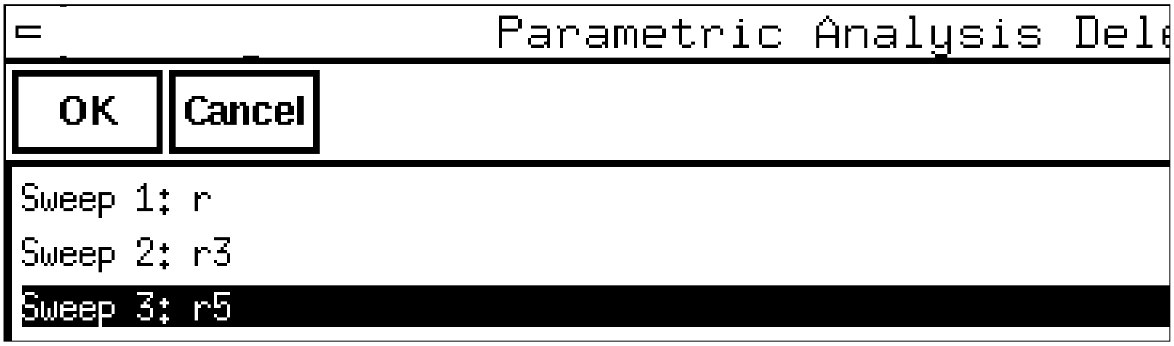
Note: When you add a variable to a window, information is added to the window but the window itself does not grow larger. You might need to enlarge the window to see all the information.

Deleting and Restoring Variables

To delete a sweep variable,

1. Choose *Setup – Delete Variable* in the Parametric Analysis window.

The Parametric Analysis Delete Variable form appears.



Sweep 1: r
Sweep 2: r3
Sweep 3: r5

2. Click on the variable you want to delete and click *OK.*

The variable is deleted and the sweep number for each higher numbered sweep decreases by one. (For example, if you delete Sweep 2, the previous Sweep 3 becomes the new Sweep 2, and the previous Sweep 4 becomes Sweep 3.)

To restore the last variable you deleted,

➤ Choose *Setup – Undo Last Change* in the Parametric Analysis window.

Note: This command reverses deletions of range specifications and deletions of variables. You can restore only the last deletion. If your last deletion is a *Delete Range Specification* or a *Delete All Range Specifications* command, you cannot restore a previous variable deletion.

Specifying Ranges

In this section, you will learn

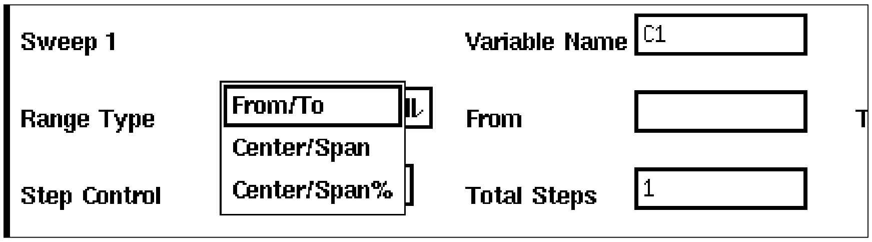
- How to specify the range for a sweep and how to choose a range of specification options
- How to specify multiple sweep ranges for a given variable
- How to delete range specifications

If you want to learn how to select some range specifications and ignore others for a given parametric analysis run, see “Setting Up and Running Statistical Analyses” on page 353.

Specifying Range Limits and Range Types

To specify range limits for a sweep,

1. Choose a range type from the *Range Type* cyclic field of the Parametric Analysis window.



Sweep 1

Variable Name: C1

Range Type: From/To

From:

Step Control

Total Steps: 1

   For detailed information about the form, see “Parametric Analysis” on page 410.

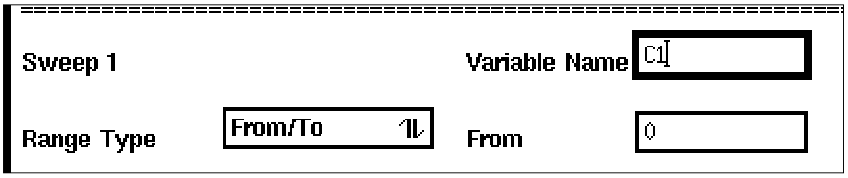
2. Type in appropriate range limits.

   Depending on the range type you choose, the fields where you type these limits are labelled *From – To* or *Center – Span*.

Specifying Multiple Ranges for a Variable

To specify an additional sweep range for a variable,

1. Choose *Range* from the *Add Specification* cyclic field in the Parametric Analysis window.



Sweep 1

Variable Name: C1

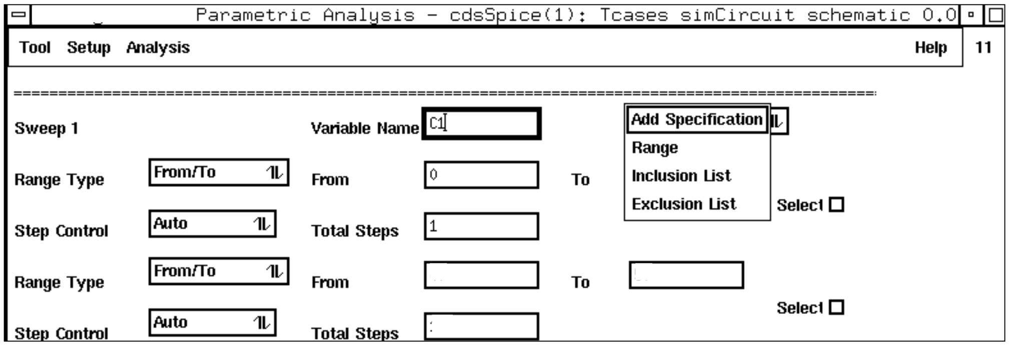
Range Type: From/To

From: 0

   See “Form Field Descriptions” on page 410 for more information.

   The form changes to let you specify an additional sweep range.

2. Type in appropriate limits.



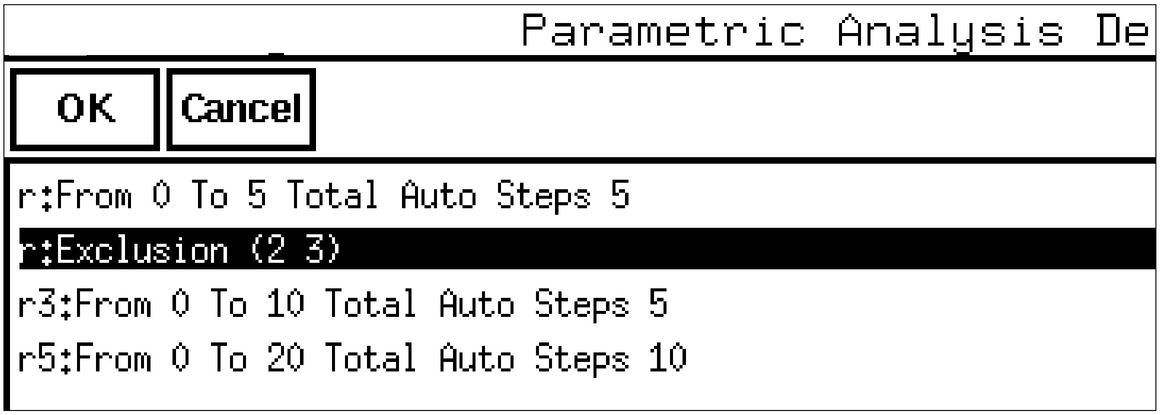
Sweep 1
Variable Name C1
Range Type From/To
From 0
To
Step Control Auto
Total Steps 1
Range Type From/To
From
To
Step Control Auto
Total Steps

Deleting Range Specifications

To delete a single range specification,

1. Choose *Setup – Delete Range Specification* in the Parametric Analysis window.

   The Delete Settings form appears.



```
r:From 0 To 5 Total Auto Steps 5
r:Exclusion (2 3)
r3:From 0 To 10 Total Auto Steps 5
r5:From 0 To 20 Total Auto Steps 10
```

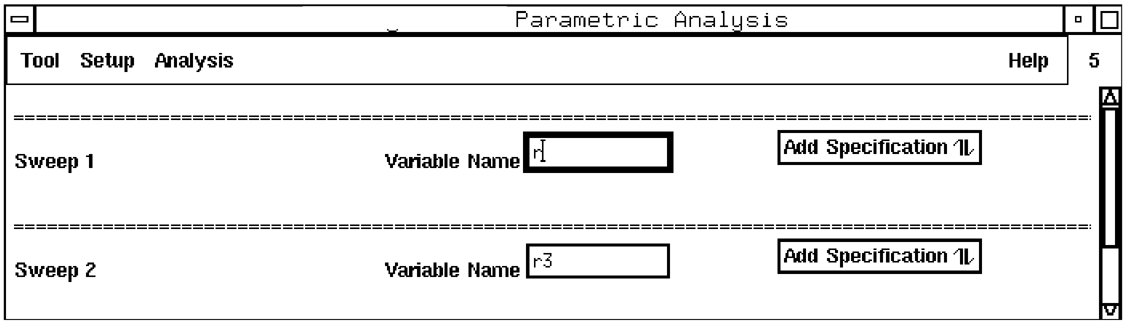
2. Click on the setting you want to delete and click *OK*.

   The setting is deleted from the Parametric Analysis window.

To delete all the range specifications,

➤ Choose *Setup – Delete All Range Specifications* in the Parametric Analysis window.

All range specifications, except the sweep fields, are deleted from the Parametric Analysis window, as shown below:



Sweep 1 Variable Name r

Sweep 2 Variable Name r3

Note: To add new range specifications to the window, use the *Add Specification* field.

To restore the last specification that you deleted with a *Delete Range Specification* or *Delete All Range Specifications* command,

➤ Choose *Setup – Undo Last Change* in the Parametric Analysis window.

Note: This command reverses deletions of variables and range specifications. You can restore only the last deletion. If your last deletion is a *Delete Variable* command, you cannot restore a previous range specification.

You can sweep the temperature in a parametric analysis.

1. Choose *Tools – Parametric Analysis* in the *Virtuoso Analog Design Environment* window.
2. The *Parametric Analysis* window appears. You use this window to specify values for the parametric analysis. Specify `temp` as the *Variable Name*. Also specify the *sweep range* and *number of steps*. The `temp` variable is built-in, therefore you do not have to add your own design variables.

Storing Specifications

The Parametric Analysis window lets you store sweep settings in two ways:

- You can store settings temporarily in a buffer. With the temporary storage option, you can revert to the last saved state. These settings are lost when you close the Parametric Analysis window.
- You can store settings permanently in a file that you name. With the permanent storage option, you can recall the settings at any time.

Temporary Storage

To store sweep settings temporarily in a buffer,

➤ Choose *Tool – Checkpoint* in the Parametric Analysis window.

Your sweep settings are stored in a buffer.

To revert to the sweep settings stored in a buffer, use the following procedure.

Caution

The Revert menu command eliminates your current settings and replaces them with settings from the buffer.

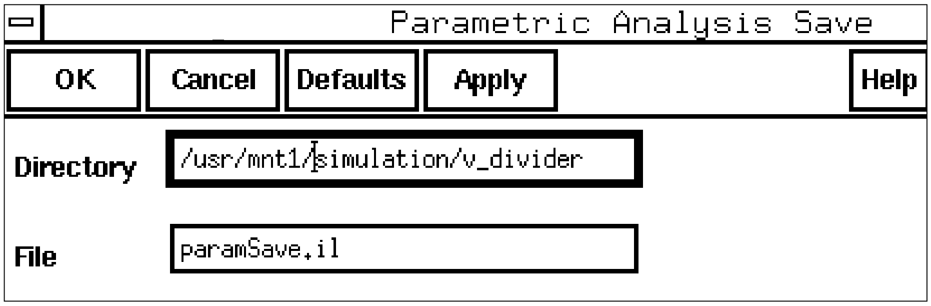
1. Choose *Tool – Revert* in the Parametric Analysis window.
2. Click *Yes* in the dialog box.

Permanent Storage

To save sweep settings permanently in a file,

1. Choose *Tool – Save* in the Parametric Analysis window.

   The Parametric Analysis Save form appears.



Directory /usr/mnt1/simulation/v_divider

File paramSave.il

2. Type in the directory path and filename you want.
3. Click *OK* or *Apply*.

   If you click *OK*, the Parametric Analysis Save form disappears.

Important

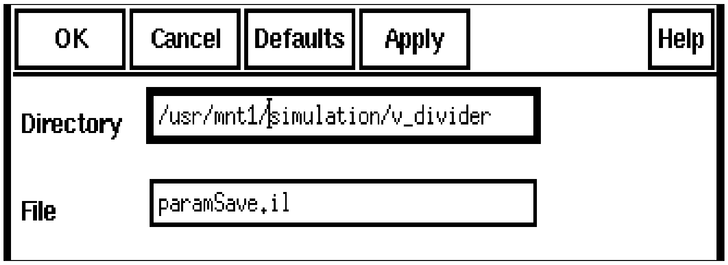
> Once you invoke the *Save* form and click *OK* or *Apply*, a subsequent *Recall* invocation will display the same values used in the *Save* form.

Note: If you click on *Defaults*, the directory path and filename revert to the settings that were current when you brought up the Parametric Analysis window.

To recall sweep settings from a file, use the following procedure:

1. Choose *Tool – Recall* in the Parametric Analysis window.

   The Parametric Analysis Recall form appears.



Directory /usr/mnt1/simulation/v_divider

File paramSave.il

2. Type in the directory path and filename you want to recall.
3. Click *OK* or *Apply*.

   If you click *OK*, the Parametric Analysis Recall form disappears.

Note: Clicking on *Defaults* recalls settings from the directory path and filename that were current when you brought up the Parametric Analysis window.

Important

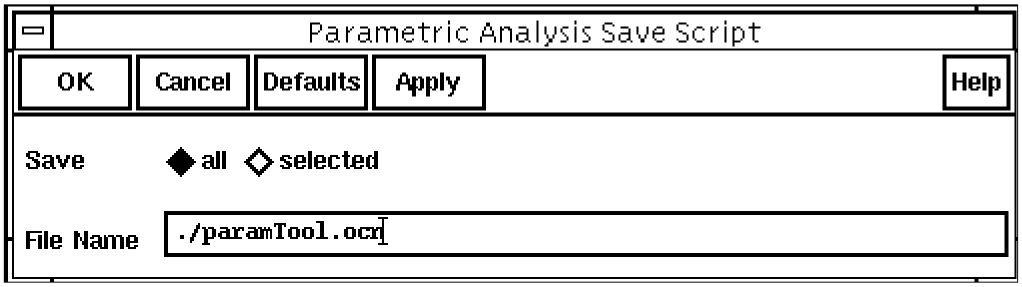
> If you invoke the *Recall* form (without invoking the *Save* form earlier) you will see the default values. If you then change the values (*Directory*, *FileName* or both) and then click *OK* or *Apply*, subsequent invocation of *Recall* would display the changed values.

Saving a Script

To save a script for later use in the OCEAN environment,

1. Choose *Tool – Save Script* in the Parametric Analysis window.

The Parametric Analysis Save Script form appears.



Save ◆ all ◇ selected

File Name ./paramTool.ocn

2. Choose which simulations to save in the script. Turning on *all* saves a script that runs every simulation specified in the Parametric Analysis window. Turning on *selected* saves a script that runs only the simulations identified with the *select* button in the Parametric Analysis window.
3. Type in the path and file where you want the script to be saved.
4. Click *OK* or *Apply*.

   The script is saved in the file. For additional information about using the saved script with OCEAN, see the *OCEAN Reference*.

Viewing Specifications

Parametric analysis helps you edit your sweep specifications by letting you view the data points you selected. You have four viewing options, which you can select from the *Analysis* menu of the Parametric Analysis window.

To view all your data points in the order they are specified in the Parametric Analysis window,

➤ Choose *Analysis – Show Points – All...*

To view all your data points in the order they are simulated in a Parametric Analysis run,

➤ Choose *Analysis – Show Points – All Sorted...*

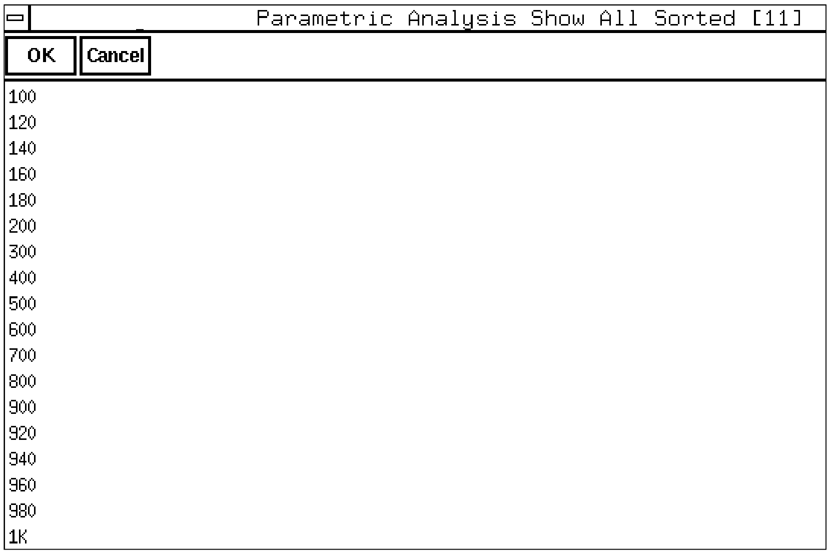
To view selected data points in the order they are specified in the Parametric Analysis window,

➤ Choose *Analysis – Show Points – Selected...*

To view selected data points in the order they are simulated in a Parametric Analysis run,

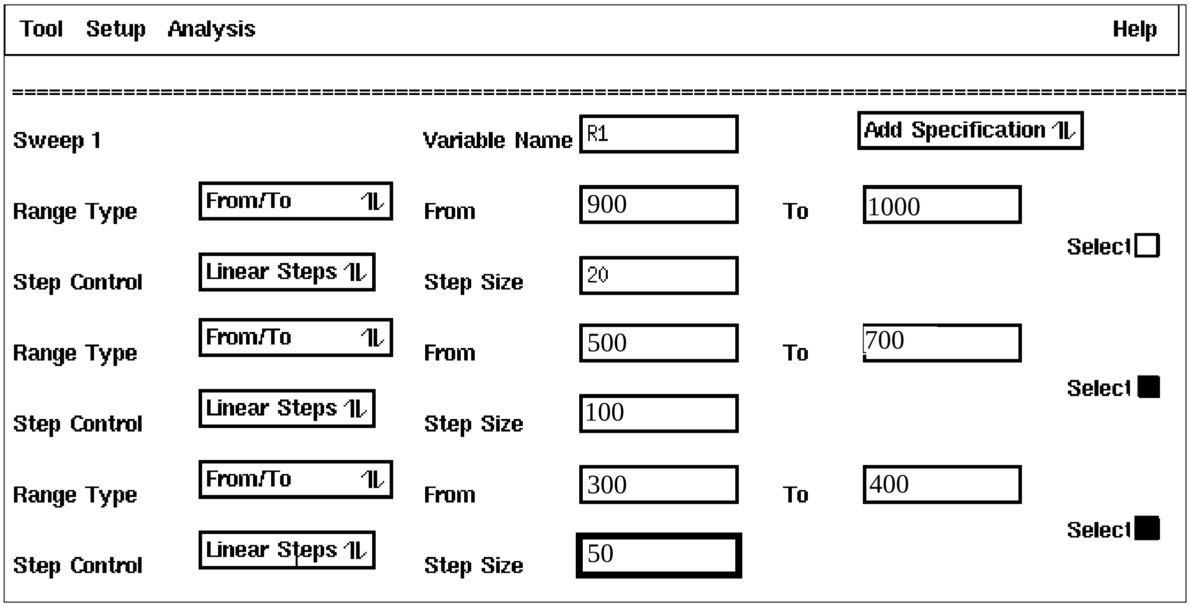
➤ Choose *Analysis – Show Points – Selected Sorted...*

For each of the four options, a window appears with a list of the data points you requested.



```
100
120
140
160
180
200
300
400
500
600
700
800
900
920
940
960
980
1K
```

For example, consider the following range and step definitions.



Sweep 1 Variable Name R1

Range Type From/To From 900 To 1000 Select □
Step Control Linear Steps Step Size 20

Range Type From/To From 500 To 700 Select ■
Step Control Linear Steps Step Size 100

Range Type From/To From 300 To 400 Select ■
Step Control Linear Steps Step Size 50

You can view the data points for this window in the four ways shown below.

All – 900, 920, 940, 960, 980, 1000, 500, 600, 700, 300, 350, 400

All Sorted – 300, 350,400, 500, 600, 700, 900, 920, 940, 960, 980, 1000

Selected – 500, 600, 700, 300, 350, 400

Selected Sorted – 300, 350, 400, 500, 600, 700

Specifying Step Values and Types

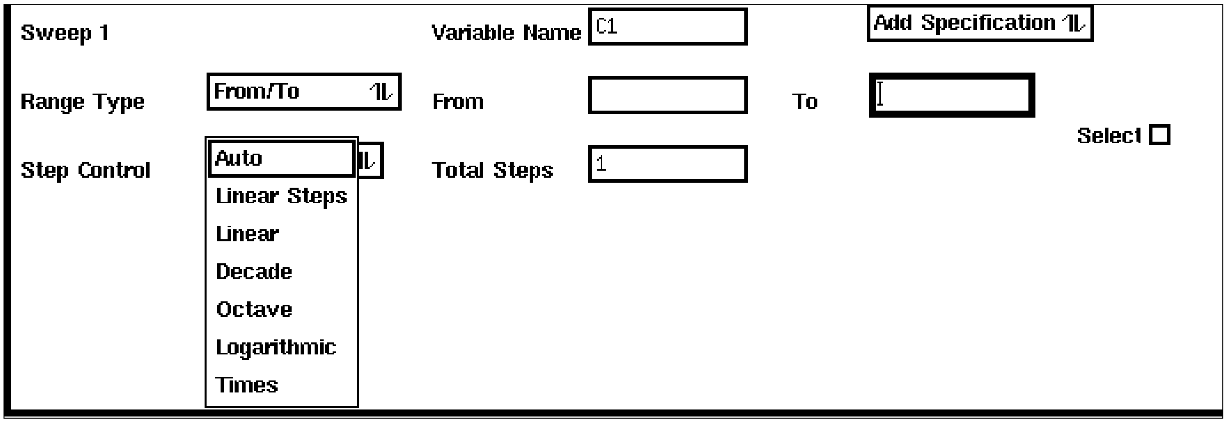
To specify appropriate step values for a sweep,

1. In the Parametric Analysis window, select the step-value type from the *Step Control* cyclic field.

   The step-value type determines the interval between step values. You can select from among a number of options. When you choose a step value, the *Total Steps* field (to the right) changes to fit your selection.

2. Specify the number of steps or the step size in the *Total Steps* field. (The default is 5 steps if you leave the field blank.)

   To learn more about a particular step-value type, click on that type in the window segment below. To learn more about all the step-value types, see “Parametric Analysis” on page 410.



Sweep 1

Variable Name C1

Range Type From/To

From

To

Select □

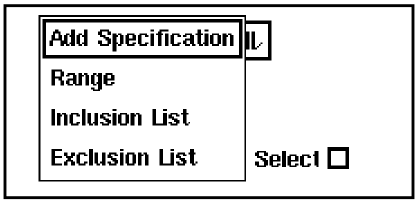
Step Control Auto

Total Steps 1

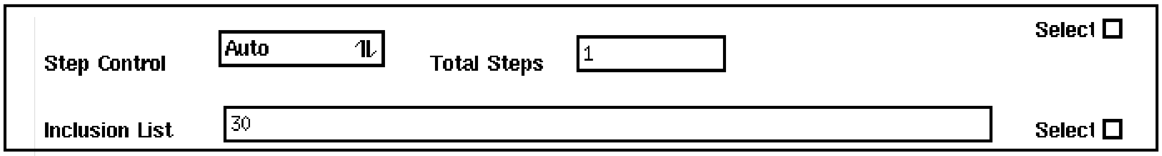
Adding Individual Step Values

To add individual step values,

1. Choose *Add Specification – Inclusion List* in the Parametric Analysis window.



The field is added to the Parametric Analysis window.



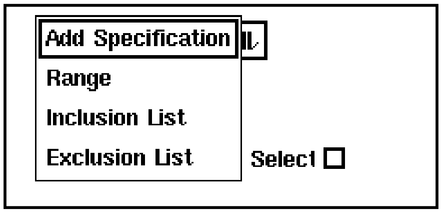
Step Control: Auto
Total Steps: 1
Select □
Inclusion List: 30
Select □

2. Type in the values that you want to add.
3. If necessary, request another inclusion list from the *Add Specification* field.

Deleting Individual Step Values

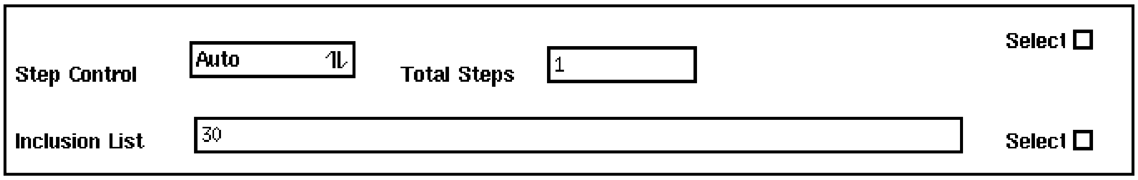
To delete individual step values,

1. Choose *Add Specification – Exclusion List* in the Parametric Analysis window.



For detailed information about the form, see "Parametric Analysis" on page 410.

The field is added to the Parametric Analysis window.



Step Control: Auto
Total Steps: 1
Select □
Inclusion List: 30
Select □

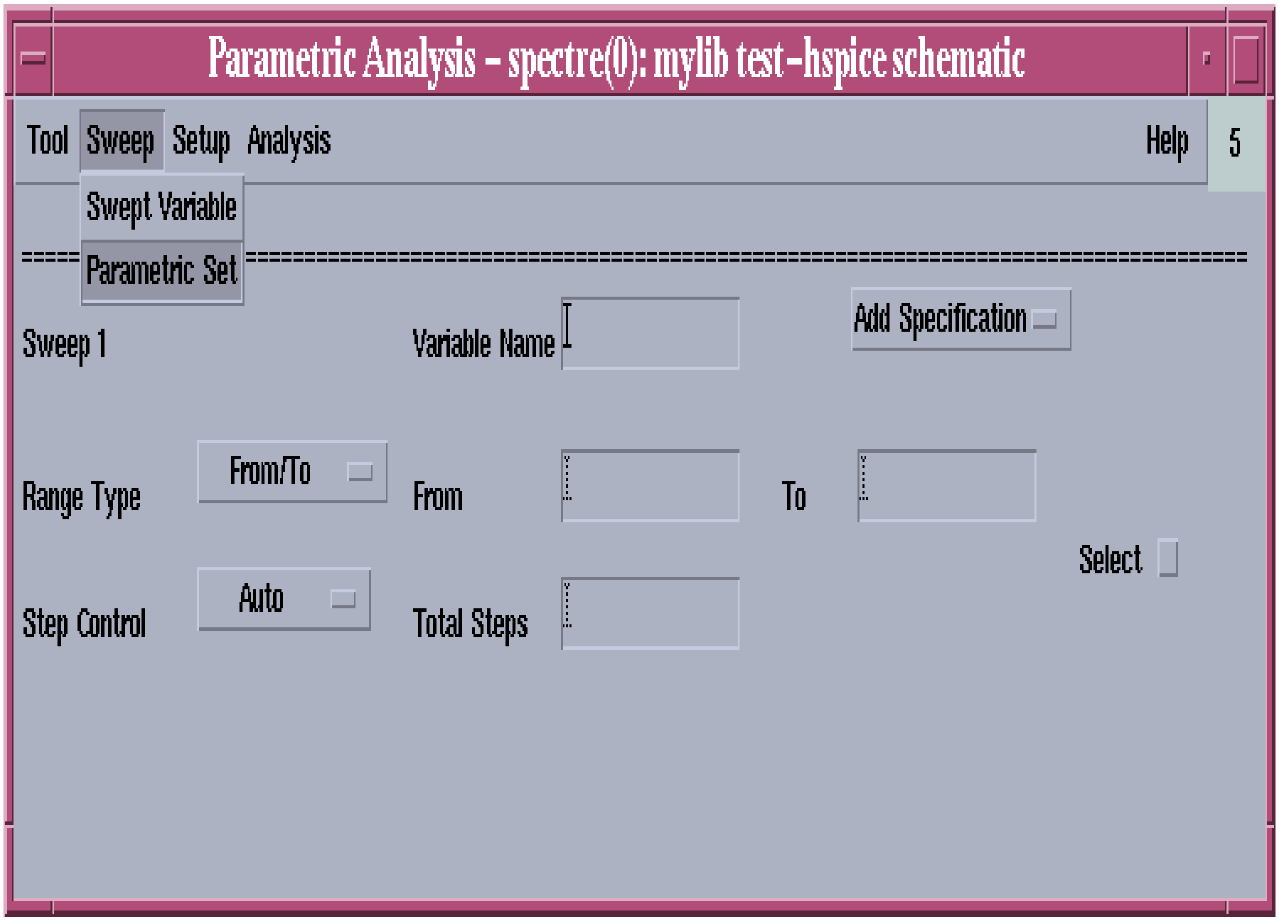
2. Type in the values that you want to delete.
3. If necessary, request another exclusion list from the *Add Specification* field.

Parametric Set Sweep

The *Parametric Set Sweep* feature has been added to the *Parametric Analysis* tool. This allows you to sweep parameter groupings and allows the specification of multiple lists of parameters. The tool then picks the first parameter value from each list for the first iteration, the second value of each list for the second iteration, and so on.

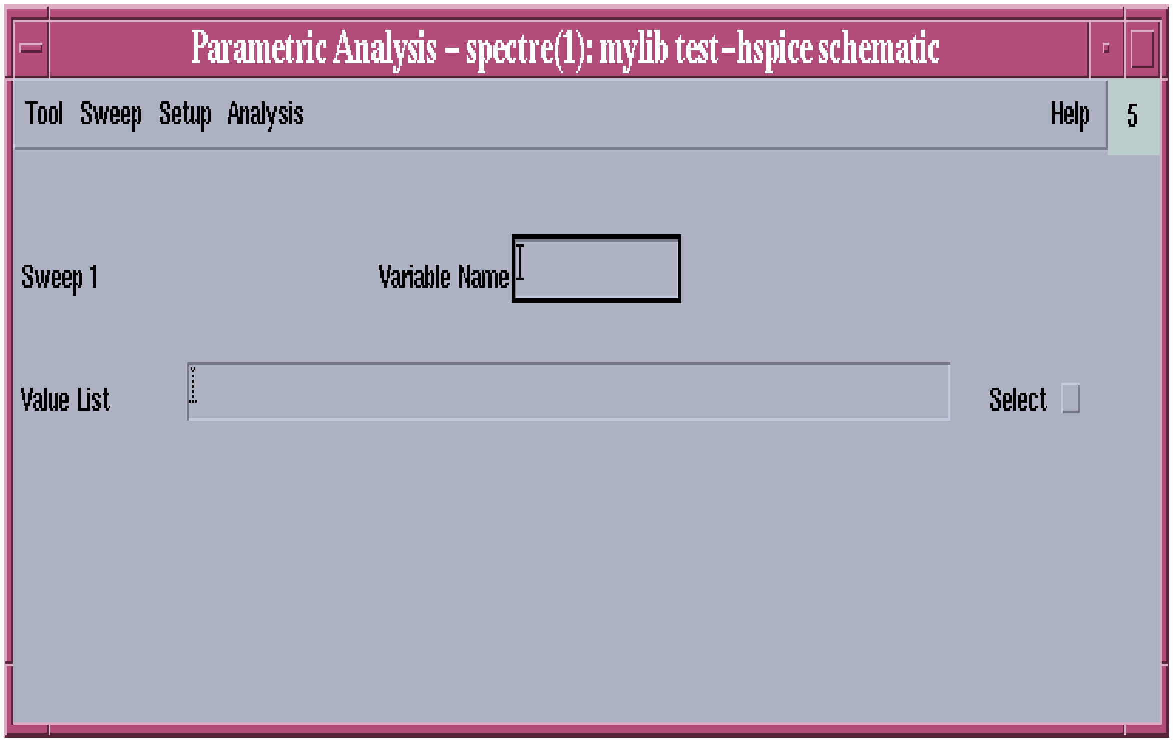
Note: Please see `spectre -h paramset` for more information.

The sweep types supported by the *Parametric Analysis* tool are *Swept Variable* and *Parametric Set*, where *Swept Variable* is the default mode. You can select these using the *Sweep* menu. You can also switch between the supported sweep types without losing the contents of the form.



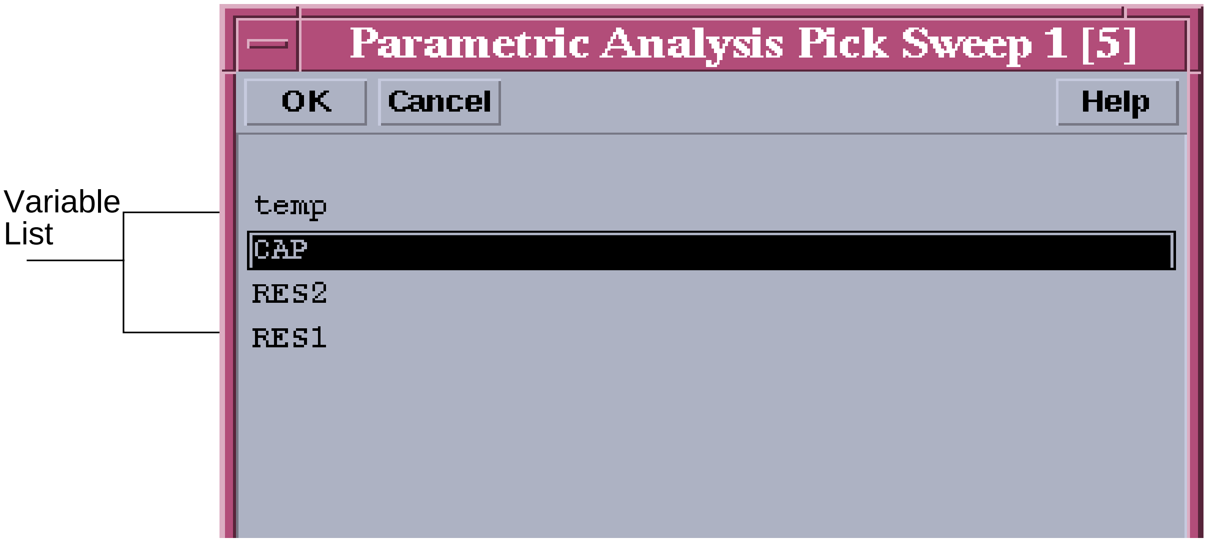
To perform a *Parametric Set* sweep,

1. Choose *Sweep – Parametric Set.*


Parametric Analysis - spectre(1): mylib test-hspice schematic
Tool Sweep Setup Analysis
Help
5
Sweep 1
Variable Name
Value List
Select

Specify the *Variable Name* for the sweep. Also specify the list of values used for the *Parametric* simulation.

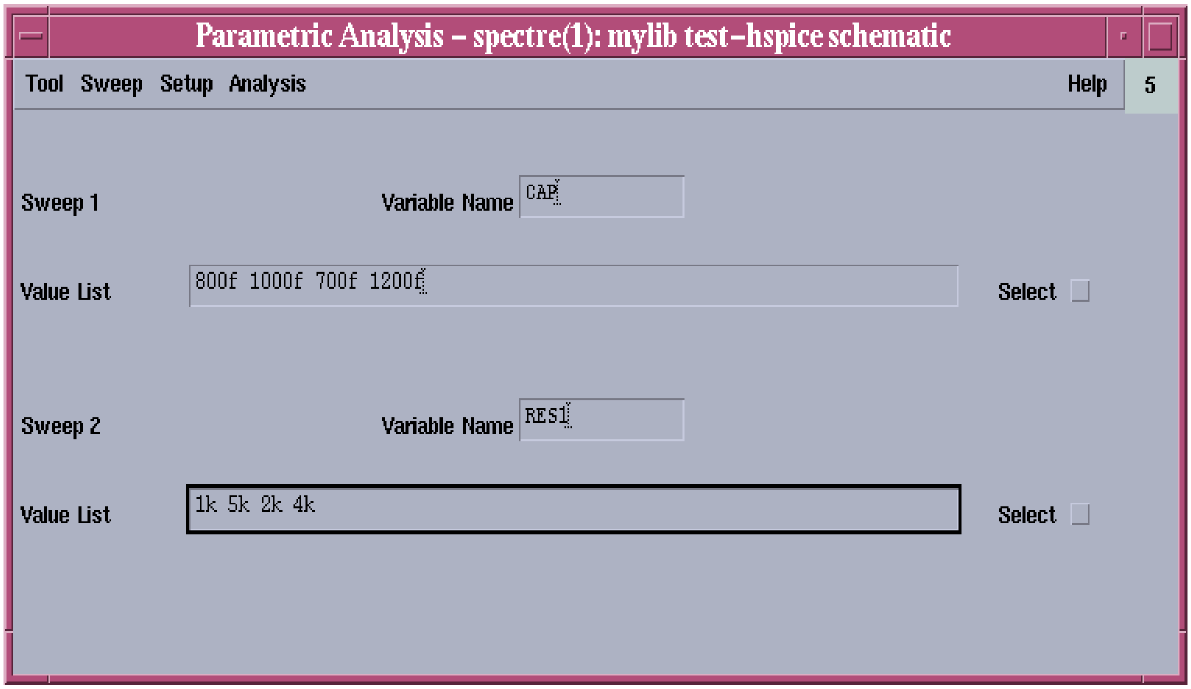
You can also select the *Variable Name* using *Setup – PickName For Variable*. For details see Specifying Sweep Variables. A *Pick Sweep* window displays a list of variables you can sweep.



temp
CAP
RES2
RES1

You can also add additional sweep variables using the *Setup – Add New Variable To Top* or the *Setup – Add New Variable To Bottom* options. For details, see Adding Variables. The window re-displays and reflects the added sweeps and variables.

For example, for *Variable Name:* `CAP and RES and` *Value List* of `800f 1000f 700f 1200f and 1K 5K 2K 4K,` the specified *Parametric Set* Sweep is as follows.


Parametric Analysis – spectre(1): mylib test–hspice schematic
Tool
Sweep
Setup
Analysis
Help
5

Sweep 1 Variable Name CAP

Value List 800f 1000f 700f 1200f Select □

Sweep 2 Variable Name RES1

Value List 1k 5k 2k 4k Select □

This example causes the following pairs of values to be simulated:

800f 1K

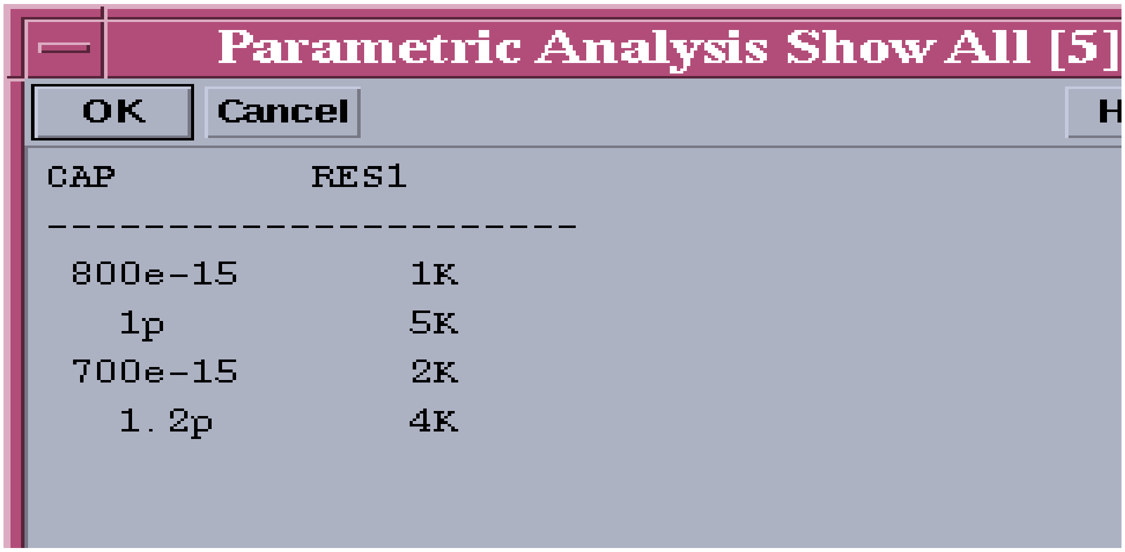
1000f 5K

700f 2K

1200f 4K

Please note that the length of the number of values is same for both the variables. This is a requirement for *Parametric Set Sweep*.

To view the *Parametric Set Sweep* sets specified select *Analysis – Show Sweep Sets – All.* For details, see Viewing Specifications. Data is represented in a tabular format in the *Parametric Analysis Show All* window.



```
CAP          RES1
----------------------
 800e-15        1K
   1p           5K
 700e-15        2K
   1.2p         4K
```

To run the run the Parametric Simulation for the data displayed, select *Analysis – Start* or *Analysis – Start Selected*. For details, see Viewing Specifications.

Running a Parametric Analysis

For information on how to select range specifications, start, interrupt, re-start and close a parametric analysis run, see

This section contains information about the following:

- How to select among your range specifications at run time

  These run-time modifications let you conveniently choose specifications for a single parametric analysis run.

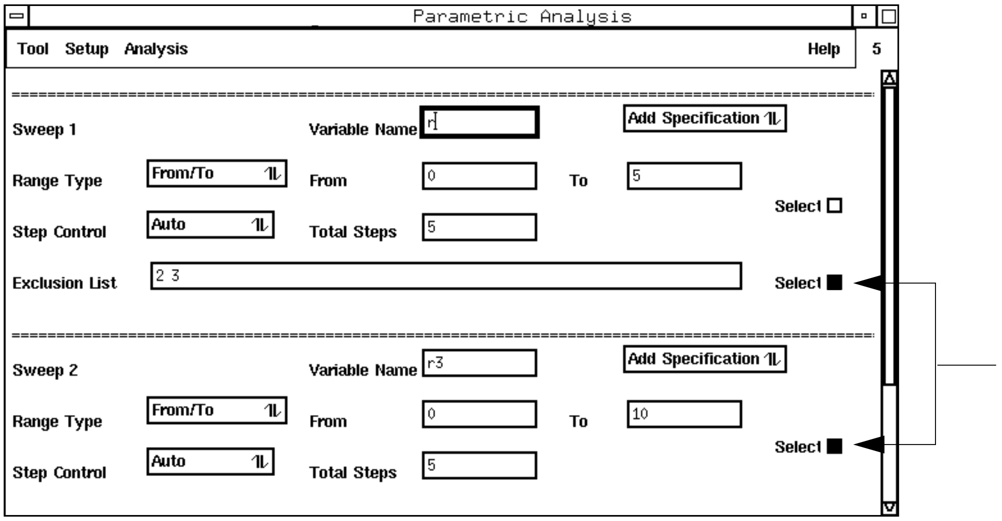
- How to start a parametric analysis run
- How to interrupt and restart a parametric analysis
- How to close the Parametric Analysis window when you finish running parametric analyses

Run-Time Modifications

To specify which range specifications you want to simulate

➤ Click the *Select* button to the right of each range specification you want in the Parametric Analysis window.

You must choose at least one range specification for each analysis.



Parametric Analysis

Sweep 1 Variable Name r
Range Type From/To From 0 To 5
Step Control Auto Total Steps 5 Select □
Exclusion List 2 3 Select ■

Sweep 2 Variable Name r3
Range Type From/To From 0 To 10
Step Control Auto Total Steps 5 Select ■

Selected buttons

To choose all range specifications in the Parametric Analysis window,

➤ Choose *Setup – Select All Range Specifications* in the *Setup* menu in the Parametric Analysis window.

To deselect all previously selected range specifications,

➤ Choose *Setup – Deselect All Range Specifications* in the Parametric Analysis window.

Starting the Run

To start a Parametric Analysis run,

➤ Choose *Analysis – Start* in the Parametric Analysis window.

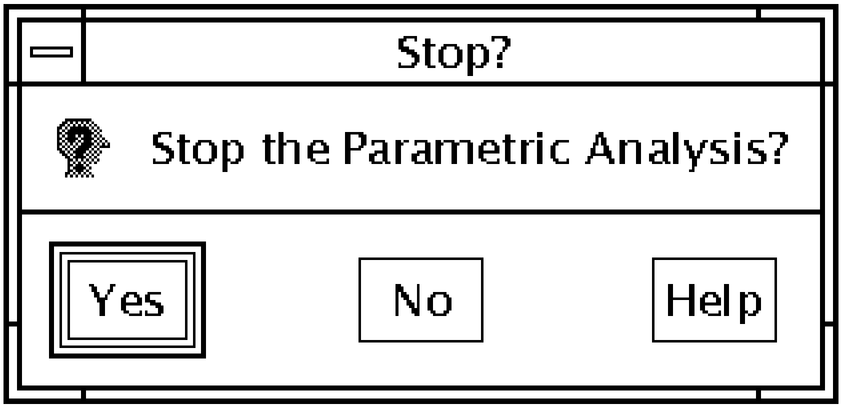
To start a Parametric Analysis run that performs only those simulations selected at run time,

➤ Choose *Analysis – Start Selected* in the Parametric Analysis window.

Interrupt and Restart

To stop the parametric analysis,

➤ Choose *Analysis – Stop* in the Parametric Analysis window. The *Stop* window appears.



Stop the Parametric Analysis?

To specify an interrupt after completion of the currently running analysis,

➤ Choose *Analysis – Pause* in the Parametric Analysis window.

To specify an immediate interrupt of an analysis,

➤ Choose *Analysis – Pause Now* in the Parametric Analysis window.

Caution

Pause Now interrupts any currently running simulation, and it might invalidate the results of that simulation.

Typically, you use *Pause Now* to interrupt a long simulation when you decide that continuing the parametric analysis is not productive.

To restart an interrupted analysis,

➤ Choose *Analysis – Continue* in the Parametric Analysis window.

Closing the Window

To close the Parametric Analysis window,

➤ Choose *Tool – Close* in the Parametric Analysis window.

Statistical Analysis

Statistical analysis is a powerful method for estimating parametric yields. The manufacturing variations in components affect the production yield of any design that includes them. Statistical analysis allows you to study this relationship in detail.

To prepare for a statistical analysis, you create a design that includes devices or device models that are assigned statistically varying parameter values. The shape of each statistical distribution represents the manufacturing tolerances on a device. During the analysis, the statistical analysis option performs multiple simulations, with each simulation using different parameter values for the devices based upon the assigned statistical distributions.

When the simulations finish, you can use the data analysis features of the statistical analysis option to examine how manufacturing tolerances affect the overall production yield of your design. If necessary, you can then switch to different components or change the design to improve the yield.

For information about running a statistical analysis, consult the Virtuoso® *Advanced Analysis Tools User Guide*.

Using the Optimizer

Analog circuit design remains a highly manual iterative process. Starting with a behavioral description, you adjust the circuit topology and component parameters until the circuit satisfies your specifications. At each iteration, you change component parameters, such as resistor values and transistor areas, and then resimulate.

The optimizer takes a given topology and automatically adjusts component parameters to meet your specifications. The optimizer uses powerful numerical algorithms to guide the changes in component values in the correct direction. Often the optimizer can take a design that is close to meeting the specifications, run a series of simulations, and generate new component values that push the design into the acceptable performance range.

For more information about the optimizer, consult the Virtuoso® *Advanced Analysis Tools User Guide*.

Corners Analysis

The corners tool provides a convenient way to measure circuit performance while simulating a circuit with sets of parameter values that represent the most extreme variations in a manufacturing process.

With the tool, you can compare the results for each set of parameter values with the range of acceptable values. By revising the circuit, if necessary, so that all the sets of parameters produce acceptable results, you can ensure the largest possible yield of circuits at the end of the manufacturing process.

For more information about running Corners analyses, consult the Virtuoso® *Advanced Analysis Tools User Guide*.

UltraSim Power Network Solver

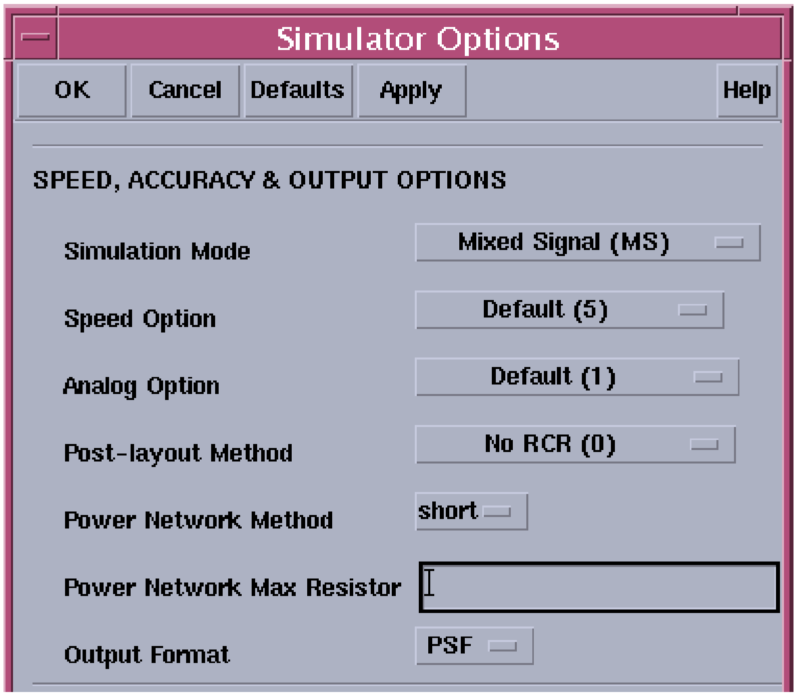
This section describes how to detect and analyze power networks using the Virtuoso® UltraSim™ power network solver (UPS) in the analog design environment.

To analyze IR drop effects and their influences on circuit behavior, parasitics in the power and ground net of a circuit design need to be extracted and analyzed together with the circuit. Parasitic elements, such as resistors, capacitors, and inductors, build the power network. These elements need to be simulated, so the parasitic effects on circuit behavior can be analyzed.

To use UPS to detect and analyze power networks:

1. Choose *Simulation – Options – Analog*.

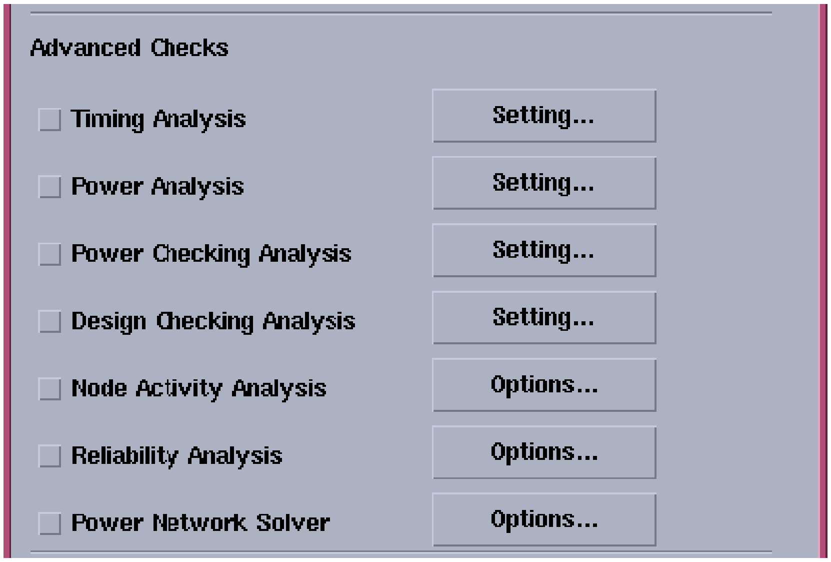
The Simulator Options form appears.



SPEED, ACCURACY & OUTPUT OPTIONS

Simulation Mode	Mixed Signal (MS)
Speed Option	Default (5)
Analog Option	Default (1)
Post-layout Method	No RCR (0)
Power Network Method	short
Power Network Max Resistor	
Output Format	PSF

2. Choose *ups* from the *Power Network Method* cyclic pulldown button.
3. Click the *Power Network Solver* check box in the *Advanced Checks* section.

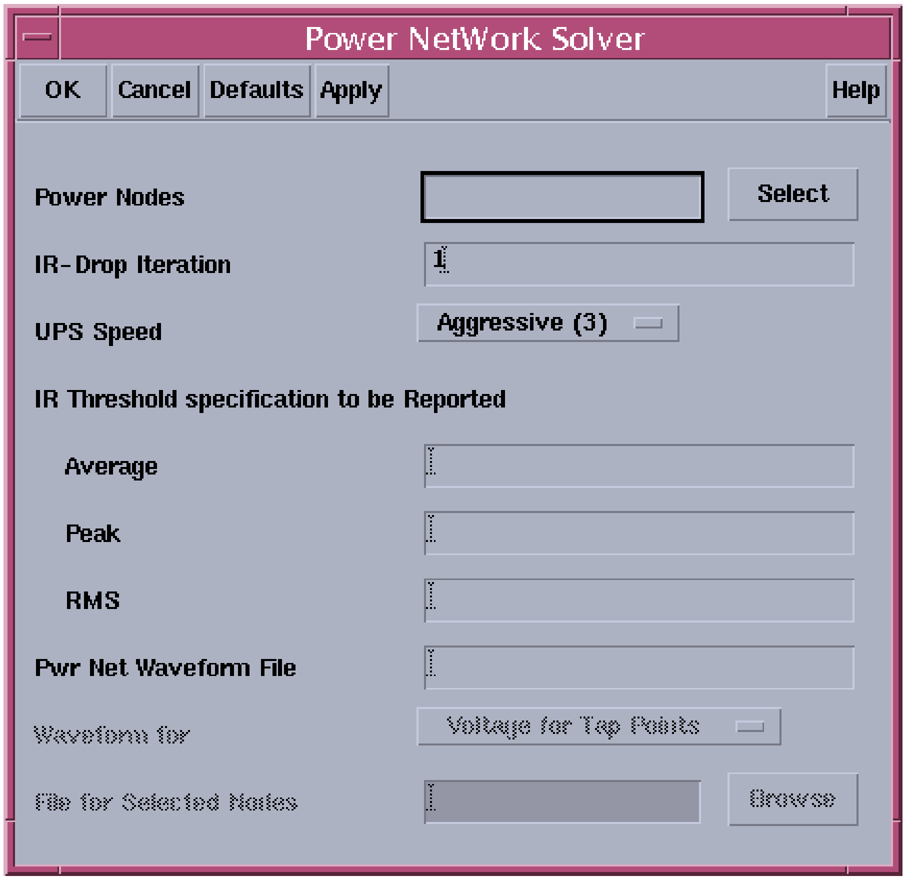


Advanced Checks

- □ Timing Analysis
- □ Power Analysis
- □ Power Checking Analysis
- □ Design Checking Analysis
- □ Node Activity Analysis
- □ Reliability Analysis
- □ Power Network Solver

4. Click the *Options* button.

The Power Network Solver window appears.



IR-Drop Iteration 1

UPS Speed Aggressive (3)

Waveform for Voltage for Tap Points

5. Adjust the power network solver options as needed.
6. Click *OK*.

For more information, refer to Chapter 5, "Power Network Solver" in the *Virtuoso UltraSim Simulator User Guide*.

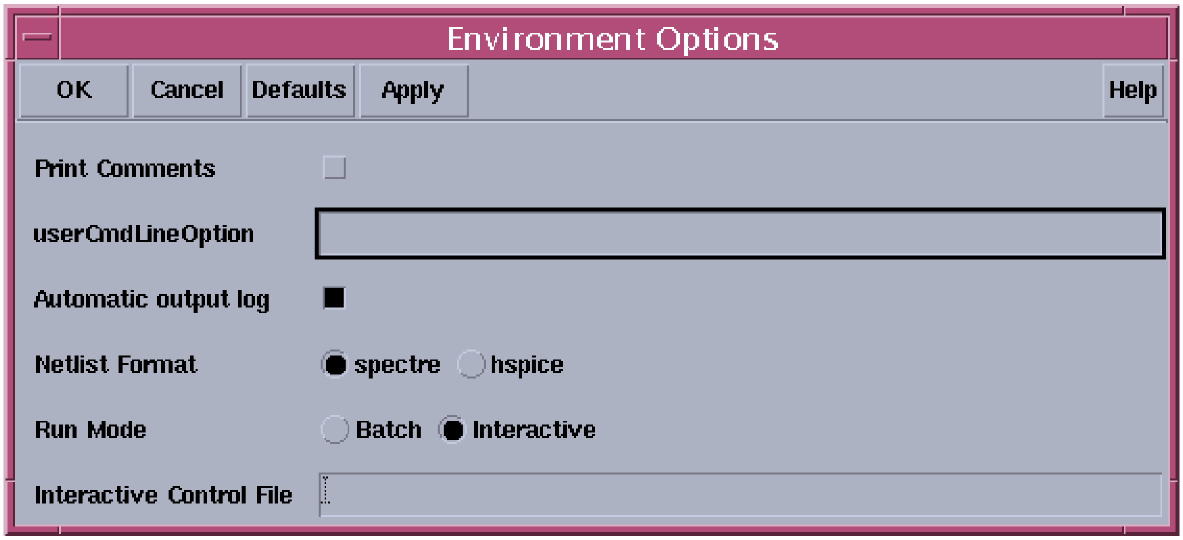
UltraSim Interactive Simulation Debugging

The Virtuoso UltraSim simulator interactive circuit debugging mode allows you to obtain design data, such as circuit elements and parameters, circuit topology, and instantaneous signal values. It can also be used to probe dynamic circuit behavior, including voltage and current waveforms simulated to the current time step.

To use the interactive simulation debugging mode:

1. Choose *Setup – Environment* in the simulation window.

The Environment Options window appears.



Print Comments □

userCmdLineOption

Automatic output log ■

Netlist Format ● spectre ○ hspice

Run Mode ○ Batch ● Interactive

Interactive Control File

2. Choose the *Interactive* radio button.

3. Type in the *Interactive Control File* name.

4. Click *OK.*

For more information about interactive debugging, refer to Chapter 6, "Interactive Simulation Debugging" in the *Virtuoso UltraSim Simulator User Guide*.

Form Field Descriptions

Parametric Analysis

Add Specification adds or deletes values from your original range specification. The Parametric Analysis window expands to accept the new specifications.

Range adds another set of range specification fields. You can reset all the range and step control options for each new range specification.

Inclusion List adds specific points or expressions to plot in the range you have specified. You can specify any number of additional points. If you need more space to type in values, request more inclusion lists.

Exclusion List deletes specific points or expressions from the list of points to be plotted. You can specify any number of points. If you need more space to type in values, request more exclusion lists.

Range Type gives you options for specifying the sweep range. Depending on the range type you choose, the two data entry fields to the right of the *Range Type Menu* (with default values *From* and *To*) change to match the range type you select.

From/To lets you specify the limits of the sweep range with numerical values.

Center/Span lets you specify a center point and the range of values around the center you want to sweep. For example,
center = 100, *span* = 20 is equivalent to *from* = 90, *to* = 110.

Center/Span% also lets you specify a center point and a range around the center. With this option, you specify range limits as a percentage of the center value. For example, *center* = 100, *span%* = 40 is equivalent to *from* = 80, *to* = 120.

Step Control gives you options for specifying the size and number of steps to be used during the simulation.

Auto sweeps five steps between the start and stop values you specify. If the ratio between the start and stop values is greater than 1:50, the system uses logarithmic steps and sweeps powers of 10 with equidistant exponents. Otherwise, the sweep steps are equidistant and linear. With this option, you do not have to enter data in the *Total Steps* field.

For example, if you enter a start value of 2 and a stop value of 100, a start:stop ratio of 1:50, the parametric analyzer sweeps the following linear step values:

2	26.5	51	75.5	100

If you enter a start value of 2 and a stop value of 101, a start:stop ratio greater than 1:50, the parametric analyzer uses the following logarithmic step values.

2 5.33154 14.2127 37.8877 101

These steps, with exponents rounded to two decimal places, are $10^{.30}$, $10^{.73}$, $10^{1.15}$, $10^{1.58}$, and $10^{2.00}$.

Linear Steps sweeps a number of equidistant steps determined by the size of the step you specify.

For example, if you enter a start value of 1.0, a stop value of 2.1, and a *Step Size* value of 0.2, the parametric analyzer simulates at the following values:

1.0 1.2 1.4 1.6 1.8 2.0

Linear simulates the number of steps you specify and automatically assigns equal intervals between the steps. With this option, there is always a simulation at both the start and stop values. The number of steps must be an integer value greater than 0.

For example, if you enter a start value of 0.5, a stop value of 2.0, and a *Total Steps* value of 4, the parametric analyzer simulates at the following values:

0.5 1.0 1.5 2.0

Decade assigns the number of steps you specify between the starting and stopping points using the following formula:

$\textit{decade multiplier} = 10^{1/\text{steps per decade}}$

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps per decade is 5.

For example, if you specify a start value of 1, a stop value of 10, and a *Steps/Decade* value of 5, the parametric analyzer simulates at the following values:

1 1.58489 2.51189 3.98107 6.30957 10

The values are 10^{0}, $10^{.2}$, $10^{.4}$, $10^{.6}$, $10^{.8}$, and 10^{1}.

Octave assigns the number of steps you specify between the starting and stopping points using the following formula:

$\textit{octave multiplier} = 2^{1/\text{steps per octave}}$

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps per octave is 5.

For example, if you specify a start value of 2, a stop value of 4, and a *Steps/Octave* value of 5, the parametric analyzer simulates at the following values:

2 2.2974 2.63902 3.03143 3.4822 4

These values are 2^{1}, $2^{1.2}$, $2^{1.4}$, $2^{1.6}$, $2^{1.8}$, and 2^{2}.

Logarithmic assigns the number of steps you specify between the starting and stopping points at equal-ratio intervals using the following formula:

$$\mathit{log\ multiplier} = (\mathit{To/From})^{(\mathrm{steps-1})}$$

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps is 5.

For example, if you use a start value of 3, a stop value of 15, and a *Total Steps* value of 5, the parametric analyzer simulates at the following values:

3 4.48605 6.7082 10.0311 15

The ratios of consecutive values are equal, as shown below.

3/4.48605 = 4.48605/6.7082 = 6.7082/10.0311 = 10.0311/15 = .67

Times simulates at points between the start and stop values that are consecutive multiples of the value you enter in the *Multiplier* field.

For example, if you enter a start value of 1, a stop value of 1000, and a *Multiplier* value of 2, the parametric analyzer simulates at the following values:

1 2 4 8 16 32 64 128 256 512

10

Plotting and Printing

This chapter shows you how to print and plot simulation data.

Selecting the Waveform Tool

You can plot simulation results in the waveform tool of your choice. The *Analog Design Environment* supports both WaveScan and AWD.

Although WaveScan is the default, you can dynamically change the waveform tool and override the default in one of the following ways:

- Specify *WaveScan* or *AWD* as the *Waveform Tool* in the *Editing Session Options* window *(Session – Option).* For details, see *Setting Basic Session Defaults.*
- Specify the *cdsenv* variable *`cds_ade_wftool`* as follows:

  `asimenv.startup` *`cds_ade_wftool`* `string  "awd"`

  This variable has two possible values `awd` and `wavescan` with `wavescan` being the default.
- Specify *`envSetVal()`* as follows:

  `envSetVal``( “asimenv.startup” “cds_ade_wftool” ’string  "awd" )`

  The value changed by *`envSetVal()`* is applied only in the executable and the entry in the `cdsenv` file is not updated.
- Use the function *awvSetWaveformToolName()* as follows:
  `awvSetWaveformToolName( "awd" )`
- Use the OCEAN function `ocnWaveformTool`*()* as follows:
  `ocnWaveformTool(‘awd)`

Once you change the waveform tool, the change is applicable to all the sessions and the CIW running from the executable.

The following table illustrates the postprocessing tools available with *WaveScan* and AWD as the chosen waveform viewers:

WaveScan	**AWD**
WaveScan Graph	AWD Graph
WaveScan Calculator	AWD Calculator
WaveScan Results Browser	AWD Results Browser
ADE Results Display (Print)	ADE Results Display (Print)

Note: The AMS simulator works with AWD and WaveScan. AMS uses SST2 for transient data and PSF for all other analyses.

Overview of Plotting

There are several ways to select simulation results and plot them in the Waveform window:

- From the WaveScan Calculator, use a the plot icon to plot waveforms. (If AWD is your waveform viewer, use the *plot* or *erplot* command).
- From the Results Browser, click right on a node that contains waveforms.
- From the Simulation window, use the *Outputs – To Be Plotted – Select On Schematic* command, or from the Schematic window, use the *Setup – Select On Schematic – Outputs To Be Plotted* command to select nets and terminals in the schematic. Use commands in the *Results – Plot Outputs* menu to display the curves.
- From the Simulation window, use the *Outputs – To Be Marched – Select On Schematic* command, or from the Schematic window, use the *Setup – Select On Schematic – Outputs To Be Marched* command to select nets and terminals to plot during simulation.
- From the Simulation window or the Schematic window, use the *Results – Direct Plot* command to select nets and terminals in the schematic and to plot a function immediately.

Note: When you click on a terminal, it gets selected first and then the wire gets selected. Therefore, you can now alternate between the two.

Before you can plot results, you need to run a simulation or select results. To select results,

1. Choose *Results – Select* in the Simulation window
2. Choose the current data file
3. Click *OK*

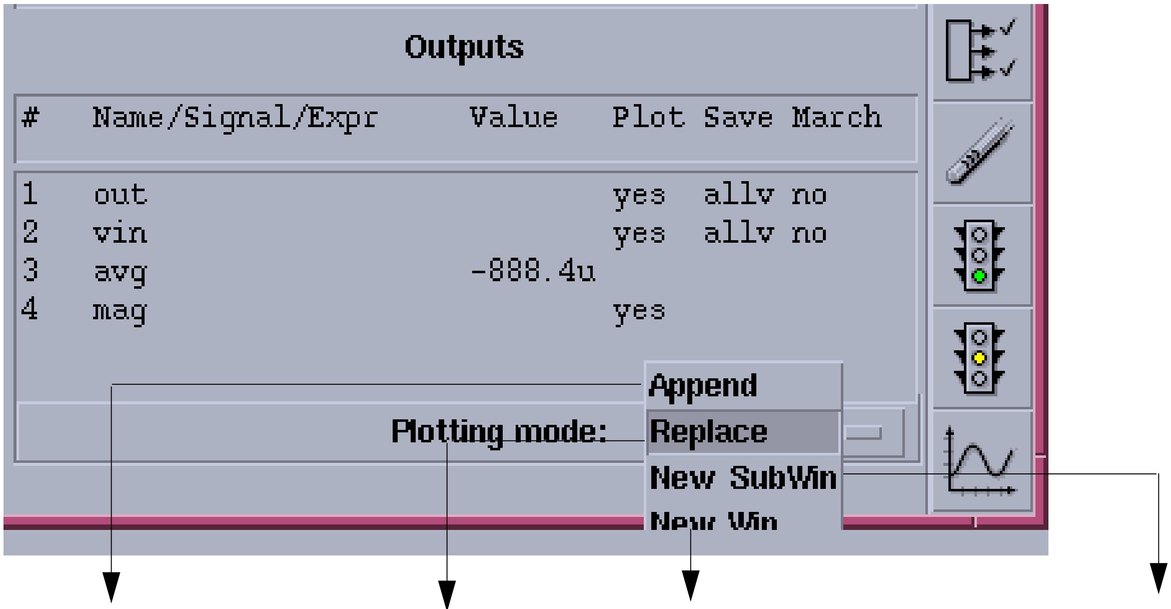
Note: The ability to plot during a simulation run is also termed as *Snapshot*.

If you set up the *Outputs* section in the Virtuoso Analog Design Environment, with nets to be plotted, and click the *Plot Outputs* icon during an analysis run, the waveform window will pop up and plot the outputs.

Therefore, you get a snapshot of the simulation run upto that time point. You can use also the *Calculator* or the *Results Browser* to plot outputs.

Plot Selector

The Analog Design Environment now supports four plotting styles that you can choose from:



Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	out		yes	allv	no
2	vin		yes	allv	no
3	avg	-888.4u			
4	mag		yes		

Appends the new signal to the current graph.

Replaces the current graph with the new signal. (Default option)

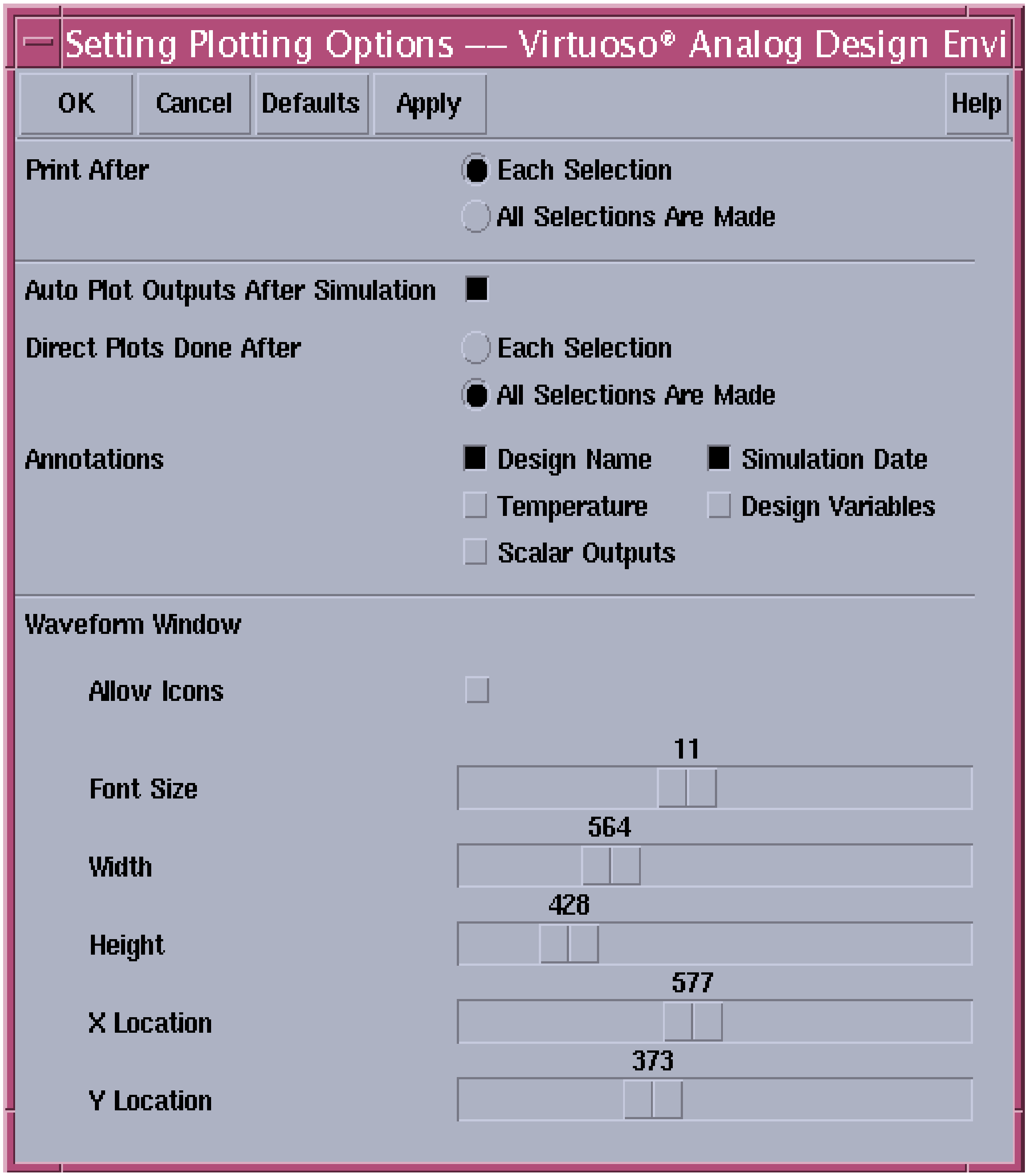
Plots the signal in a new window.

Plots the signal in a new subwindow.

These plotting styles can also be set from the *Direct Plot*, the *WaveScan Calculator* and the *WaveScanResults Browser* windows.

Setting Plotting and Display Options

You set plotting and Waveform window options with the Setting Plotting Options form.



Setting Plotting Options –– Virtuoso® Analog Design Envi

Print After ● Each Selection ○ All Selections Are Made

Auto Plot Outputs After Simulation ■

Direct Plots Done After ○ Each Selection ● All Selections Are Made

Annotations ■ Design Name ■ Simulation Date □ Temperature □ Design Variables □ Scalar Outputs

Waveform Window

Allow Icons □

Font Size 11

Width 564

Height 428

X Location 577

Y Location 373

For detailed information about the form, see “Setting Plotting Options” on page 488.

To preserve these settings in future design sessions,

- ➤ In the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

To use these settings in only the current design session,

- ➤ Click *OK*.

Note: Waveform window options you set here apply only to those windows opened by the Simulation window.

Saving and Restoring the Window Setup

You can save and restore a Waveform window setup with other setup options. For details, refer to the *WaveScan User Guide* or the *Analog Waveform User Guide*., depending on the waveform viewer you have selected.

1. In the Simulation window, choose *Session – Save State*, or in the Schematic window, choose *Analog Artist – Save State*.

   The Saving State form appears.

2. Type in a name for the saved simulation state.
3. Check that the *Waveform Setup* box is selected and click *OK*.

To restore the saved settings,

1. In the Simulation window, choose *Session – Load State*, or in the Schematic window, choose *Analog Environment– Load State*.

   The Loading State form appears. The form displays the state files in the run directory identified by the *Cell* and *Simulator* fields.

2. Select a run directory with the *Cell* and *Simulator* fields.
3. Click a *State Name* and choose *What to Load*.
4. Check that *Waveform Setup* is selected and click *OK*.

Using the Plot Outputs Commands

The five commands in the *Results – Plot Outputs* menu in the Simulation window plot each item in the plot set.

Transient	Plots the transient response for each node
AC	Plots the AC response for each node

DC	Plots the DC sweep response for each node
Noise	Plots the squared noise voltage for each node
Expressions	Plots the waveforms for expressions you define in the Setting Outputs form

Plotting the Current or Restored Results

To plot the most recent (or restored) results in the Waveform window,

1. In the Simulation window, choose *Outputs – To Be Plotted – Select on Schematic*, or in the Schematic window, choose *Setup – Select on Schematic – Outputs to be Plotted*.

   Nodes and terminals you have already selected are now highlighted.

2. In the Schematic window, select one or more nodes or terminals.
3. Press the `Escape` key when you finish selecting nodes and terminals.
4. Choose a *Results – Plot Output* command in the Simulation window.

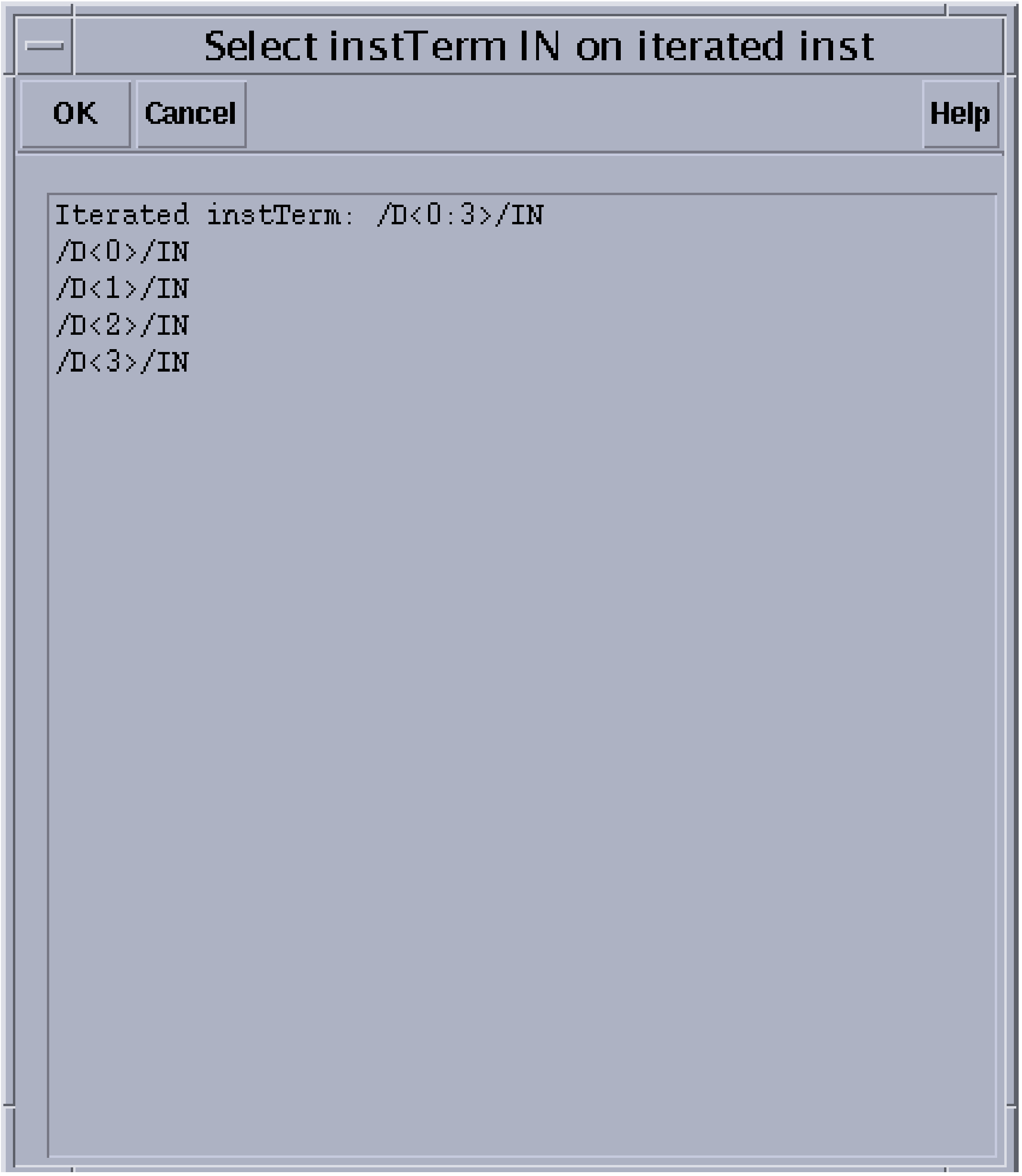
   The system plots the results you selected in the current Waveform window or opens a new Waveform window if one is not open.

To plot all of the available results at once,

➤ Click the *Plot Outputs* icon in the Simulation window.

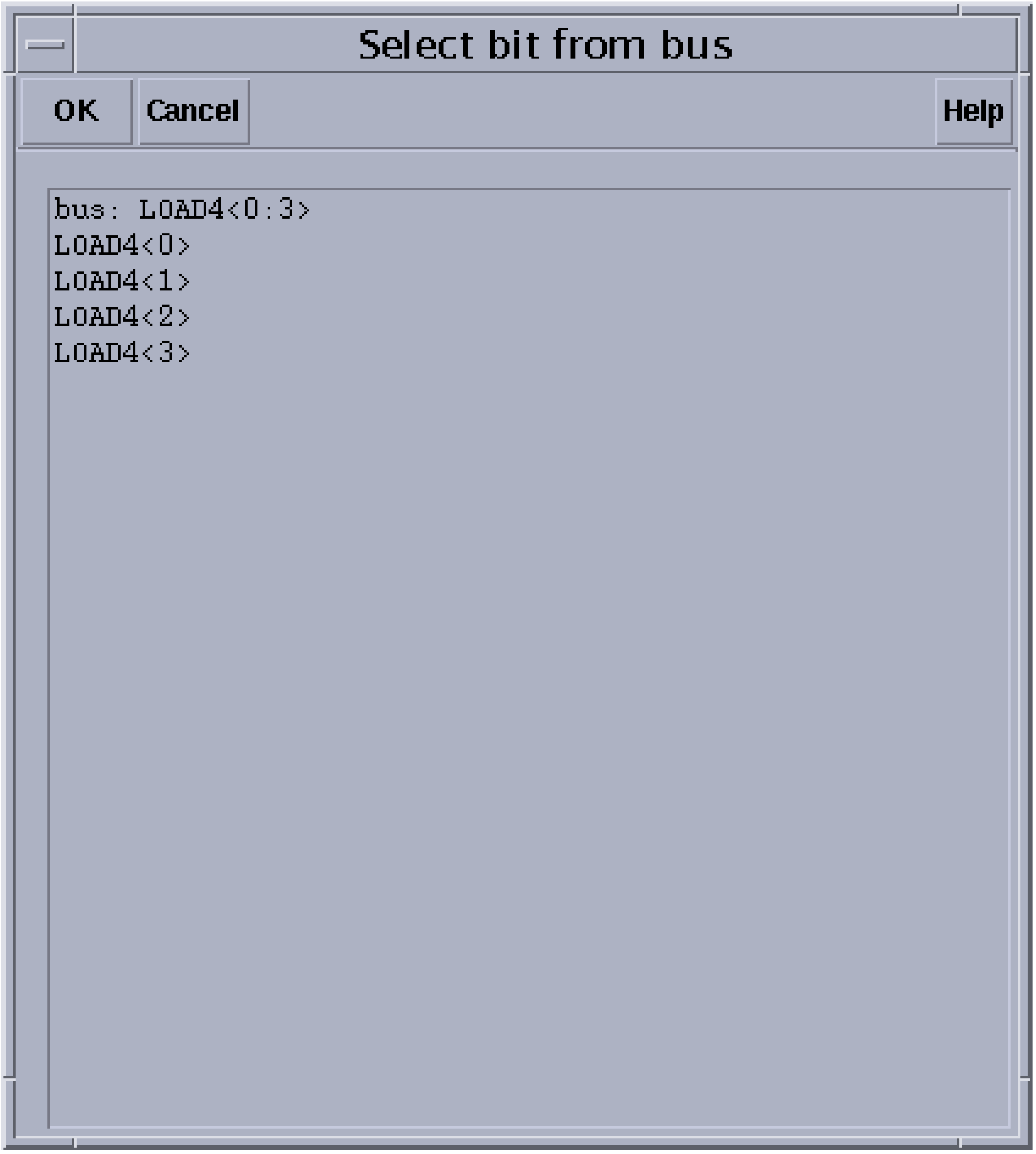
Note: You can plot only saved voltages and currents.

When you choose *Outputs – To be Plotted – Select* on schematic, and then select an iterated instance in the schematic, the *Select instTerm IN on iterated inst* form is displayed.



```
Iterated instTerm: /D<0:3>/IN
/D<0>/IN
/D<1>/IN
/D<2>/IN
/D<3>/IN
```

If you select a bus signal, the *Select bit from bus* form is displayed.



```
bus: LOAD4<0:3>
LOAD4<0>
LOAD4<1>
LOAD4<2>
LOAD4<3>
```

These forms enable you to select from one to all bits of an iterated item. When you select the top element in the listbox, all the individual bits are selected. You can also select an individual bit with the left mouse button.

Note: `Ctrl-Left` mouse will toggle selection of an item.`Shift-Left` mouse will select all items between the last selected item and the current item.

Removing Nodes and Terminals from the Plot List

To remove a node or terminal from the plotted set,

1. In the Simulation window, click in the *Outputs* list to select the output.

   To select more than one output, hold down the `Control` key while you click outputs, or click and drag.

   To deselect a highlighted output, hold down the `Control` key while you click the highlighted output.

2. Choose *Outputs – To Be Plotted – Remove From*.

Plotting Parasitic Simulation Results

When you plot the results of a parasitic simulation, only terminals and device pins can be mapped from the schematic to the extracted view.

To select results in the schematic while parasitic simulation is enabled,

1. From the Simulation window, choose *Outputs – To Be Plotted – Select on Schematic*, or in the Schematic window, choose *Setup – Select on Schematic – Outputs to be Plotted*.

2. In the schematic, select a terminal or pin, or a wire near a terminal or pin.

   If you select a point in the middle of a wire, the system chooses the nearest terminal or device pin and you might not get the right data.

   The system draws an X to mark the point you selected.

3. Choose a *Results – Plot Output* command.

   The color of the waveform matches the color of the X.

Note: You cannot probe nets that connect only sources and loads because these nets do not exist on the extracted view. You also cannot probe nets between parasitic components that were removed by selective annotation because these nets were removed when the selected view was built.

Using the Direct Plot Commands

You can plot common waveforms quickly in the Simulation window using the *Direct Plot* commands.With these commands, you do not need to use the calculator to create common expressions and you do not need to add the nets or terminals to the plot set.

To use Direct Plot,

➤ Choose *Results- Direct Plot - Main Form.* This brings up the unified *Direct Plot* main form that changes dynamically depending on the analysis that was performed.

or,

➤ Choose the *Results – Direct Plot* commands. The commands are as follows:

This option…	Plots this curve…
Transient Signal	Transient voltage or current waveforms
Transient Minus DC	Transient voltage or current waveforms without the DC offset
Transient Sum	Multiple signals added together and plotted; you are prompted for the signals
Transient Difference	Two signals subtracted (sig1- sig2) and plotted; you are prompted for two signals
AC Magnitude	AC voltage or current gain waveform
AC db10	The magnitude on a decibel scale 10log(V1)
AC db20	The magnitude of selected signals on a decibel scale 20log(V1)
AC Phase	AC voltage or current phase waveform
AC Magnitude & Phase	The db20 gain and phase of selected signals simultaneously
AC Gain & Phase	The differences between two magnitudes and two phases; you are prompted for two signals 20log(V2)-20log(V1) which is equivalent to 20log(V2/V1)
Equivalent Output Noise	Output noise voltage or current signals selected in the analysis form; the curve plots automatically and does not require selection

This option…	Plots this curve…
Equivalent Input Noise	Input noise waveform, which is the equivalent output noise divided by the gain of the circuit
Squared Output Noise	Squared output noise voltage or current signals selected in the analysis form; the curve plots automatically and does not require selection
Squared Input Noise	Input noise waveform, which is the equivalent output noise divided by the gain of the circuit squared
Noise Figure	Noise figure of selected signals according to the input, output, and source resistance
DC	DC sweep voltage or current waveform
S-Parameter	A parameter function plotted against frequency based on a pair of psin elements that define an input and an output circuit port
XF	Transfer function results
PSS	Periodic steady-state (PSS) simulation results
PDISTO	Periodic distortion (Pdisto) analysis
SPSS	Swept periodic steady-state (SPSS) simulation results
PAC	Periodic AC analysis; available after a PSS or SPSS analysis runs
PXF	Periodic transfer analysis; available after a PSS or SPSS analysis runs
PNoise	Periodic noise analysis; available after a PSS or SPSS analysis runs

To use *Direct Plot* commands,

1. Choose a command from the *Results – Direct Plot* menu.

   If necessary for the command, a form appears.

   The waveform window opens. If a waveform window was already open, it becomes active.

2. Look in the Schematic window for a prompt.

   The prompt tells you what to select in the schematic.

3. Select the signals necessary for the function and press the `Escape` key.

   The system plots the signals. The system shuffles windows automatically, so that the Waveform window is in front.

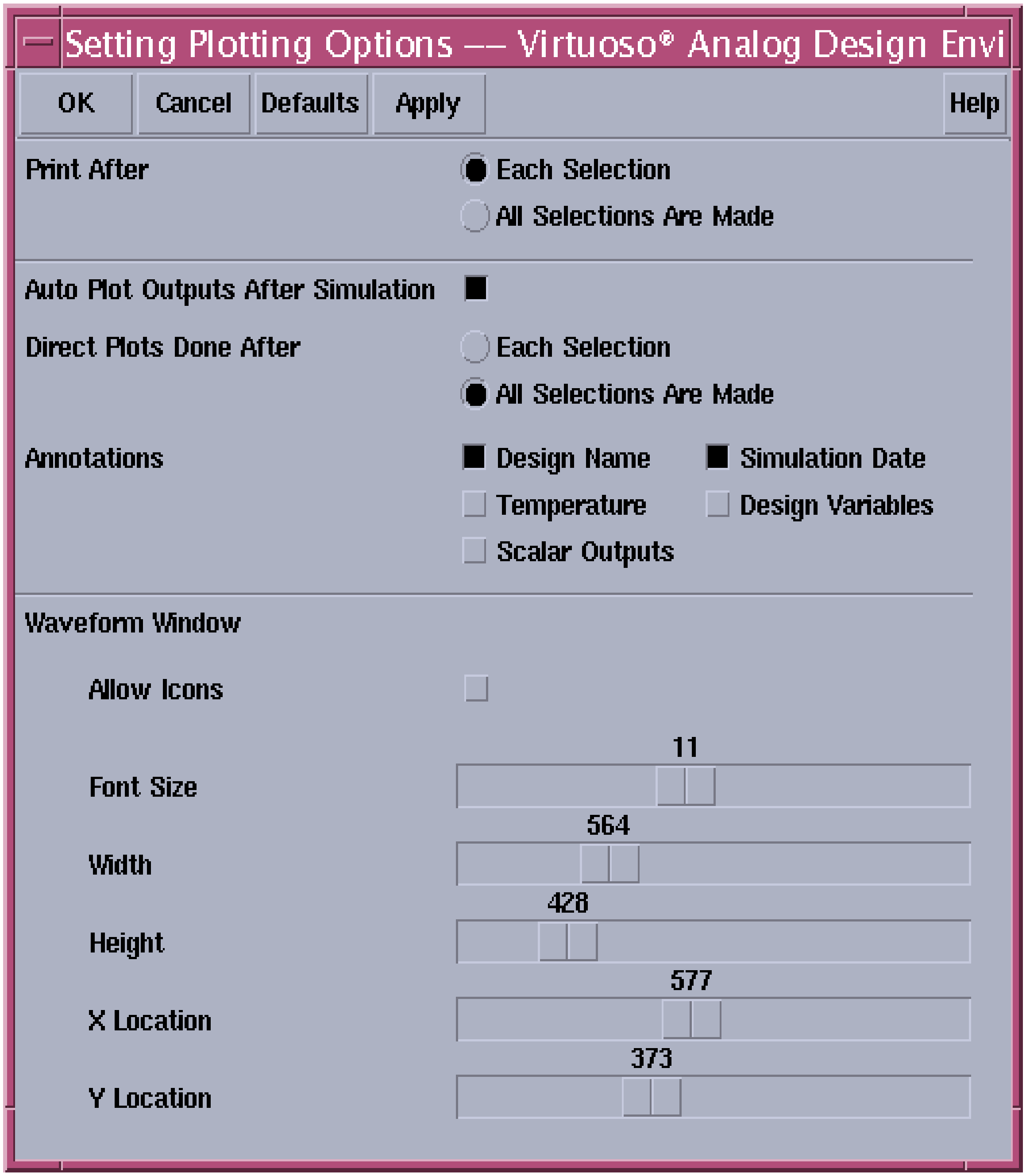
There are two modes for the *Direct Plot* commands:

- Plotting one signal at a time immediately after you select the signal
- Plotting several signals together after you press the `Escape` key

To choose the mode,

1. Choose *Results - Printing/Plotting Options*.

The Setting Plotting Options form appears.



Print After ● Each Selection ○ All Selections Are Made

Auto Plot Outputs After Simulation ☑

Direct Plots Done After ○ Each Selection ● All Selections Are Made

Annotations ☑ Design Name ☑ Simulation Date □ Temperature □ Design Variables □ Scalar Outputs

Waveform Window

Allow Icons □

Font Size 11

Width 564

Height 428

X Location 577

Y Location 373

For detailed information about the form, see "Setting Plotting Options" on page 488.

2. Choose one of the *Direct Plots Done After* options and click *OK*.

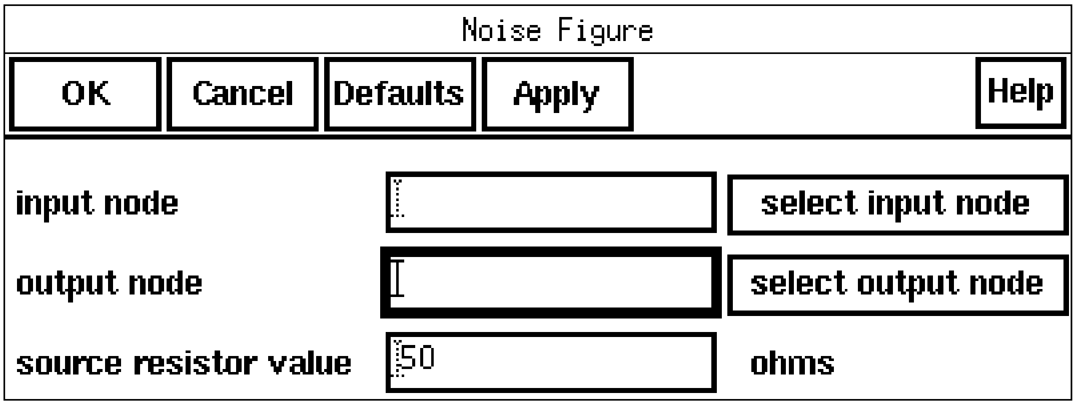
For Noise Figures

Note: When *Spectre* is the simulator, this form is not displayed. Noise figure is calculated by *Spectre* if a port is selected as the input source for noise analysis. If port is not selected as the input, noise figure data is not available.

To plot the noise figure,

1. Choose *Results – Direct Plot – Noise Figure*.

   The Noise Figure form appears.



input node		
output node		
source resistor value	50	ohms

2. Type the name of the input node, or click *select input node* and click the node in the schematic.
3. Type the name of the output node, or click *select output mode* and click the node in the schematic.
4. Click *OK*.

Note: The mathematical noise-figure expression is where:

VN2 =	The noise voltage
Vin =	The voltage at the input node
Vout =	The voltage at the output node
R =	The source resistor value
C =	1.61e-20 $\cong$ 4kTΔf

$$\mathbf{NF} = \mathbf{10\log\left(\frac{VN2*|Vin|}{C*|Vout|*R}\right)}$$

with

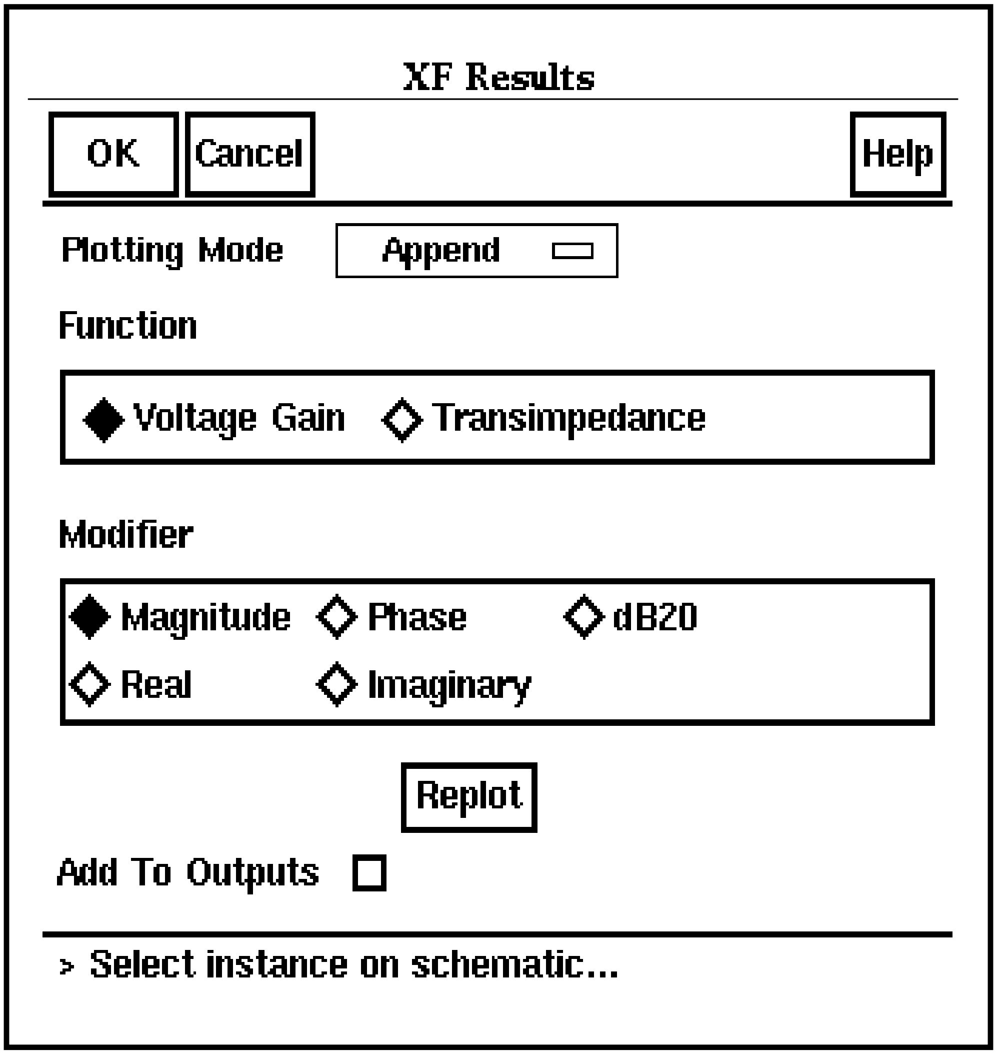
T = 291.5K and Δf = 1

For Transfer Functions

To plot the transfer function,

1. Choose *Results – Direct Plot – XF*.

The XF Results form appears.



Plotting Mode: Append

Function

◆ Voltage Gain ◇ Transimpedance

Modifier

◆ Magnitude ◇ Phase ◇ dB20
◇ Real ◇ Imaginary

Add To Outputs □

> Select instance on schematic...

For detailed information about the form, see "XF Results" on page 490.

2. Specify the *Plotting Mode*. You can specify:

- ❑ *Append* to append the new signal to the current graph.
- ❑ *Replace* to replace the current graph with the new signal. This is the default option.
- ❑ *New SubWin* to plot the signal in a new subwindow.
- ❑ *New Win* to plot the signal in a new window.

3. Choose either *Voltage Gain* or *Transimpedance* if you selected output voltage for the transfer analysis, or *Current Gain* or *Transconductance* if you selected output current for the transfer analysis.
4. Specify the modifiers as needed.
5. Select either the instance or instance terminal in the schematic.

   The Waveform window redisplays, showing the new plot.
6. To replot with modifications, make changes to the specifications on the XF Results form and click *Replot*.

For S-Parameters

A typical S-parameter direct plot shows a parameter function plotted against frequency, based on a pair of psin elements that define an input and an output circuit port.

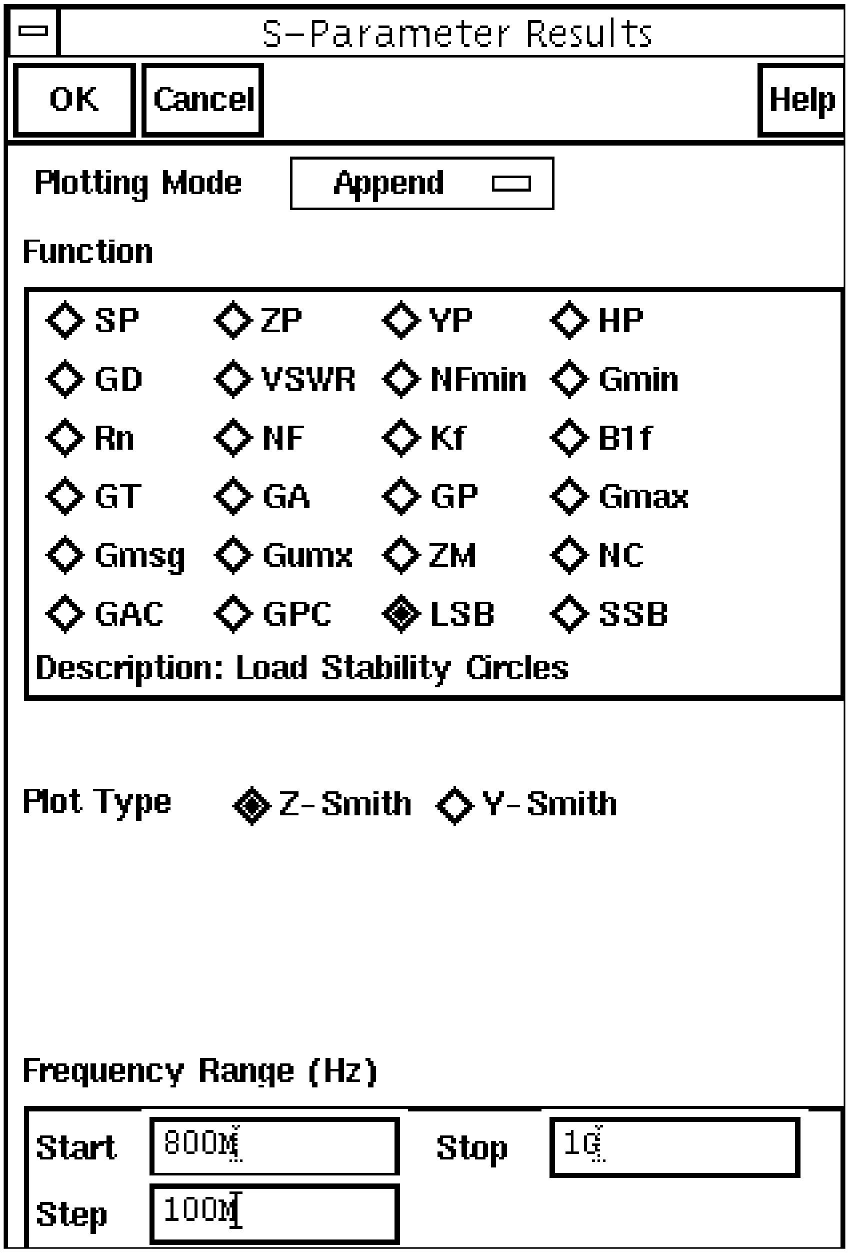
You define S-parameter direct plots with the S-Parameter Results form. If the form does not offer a plot you want to generate (for example, plots of complex computed results), use the waveform calculator.

The plots appear, by default, in the current Virtuoso® Analog Design Environment Waveform window or subwindow. The current subwindow has a rectangle around its window number (in the upper-right corner). To use a different subwindow, select it before beginning the direct plot procedure. If no Waveform window or subwindow is open, this plot function automatically opens one.

The *Results – Direct Plot – S-Parameter* command automatically opens

- A Waveform window (unless one is already open)
- The design schematic (unless it is already open)

- The S-Parameter Results form



Plotting Mode: Append

Function

○ SP	○ ZP	○ YP	○ HP
○ GD	○ VSWR	○ NFmin	○ Gmin
○ Rn	○ NF	○ Kf	○ B1f
○ GT	○ GA	○ GP	○ Gmax
○ Gmsg	○ Gumx	○ ZM	○ NC
○ GAC	○ GPC	● LSB	○ SSB

Description: Load Stability Circles

Plot Type ● Z-Smith ○ Y-Smith

Frequency Range (Hz)

Start 800M Stop 1G

Step 100M

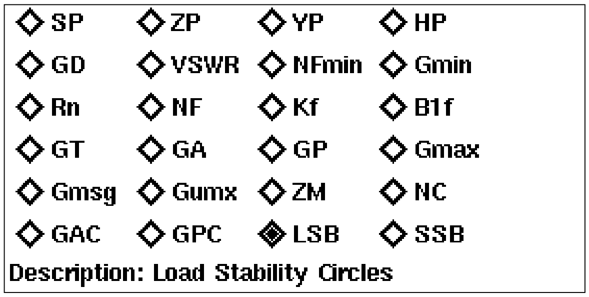
For detailed information about the form, see “S-Parameter Results” on page 491.

To plot S-parameter results,

1. Specify the *Plotting Mode*. You can specify:
   - ❑ *Append* to append the new signal to the current graph.

- ❑ *Replace* to replace the current graph with the new signal. This is the default option.
- ❑ *New SubWin* to plot the signal in a new subwindow.
- ❑ *New Win* to plot the signal in a new window.

2. Click the radio button for the S-parameter or noise-parameter function you want to plot.



Description: Load Stability Circles

A brief description of the function appears below the buttons, and the bottom of the form changes to show options for the function.

Note: Some functions are defined only for two-port circuits. If you choose a function that is not available for your circuit data set, a warning message appears at the bottom of the form. Click a button on the figure for information about a function. If you need an equation that is not represented on the form, use the waveform calculator to build, evaluate, and plot it.

3. Choose the appropriate *Plot Type* and *Modifier* to specify the plot type and the data or plot format.

4. Specify and draw the plot.

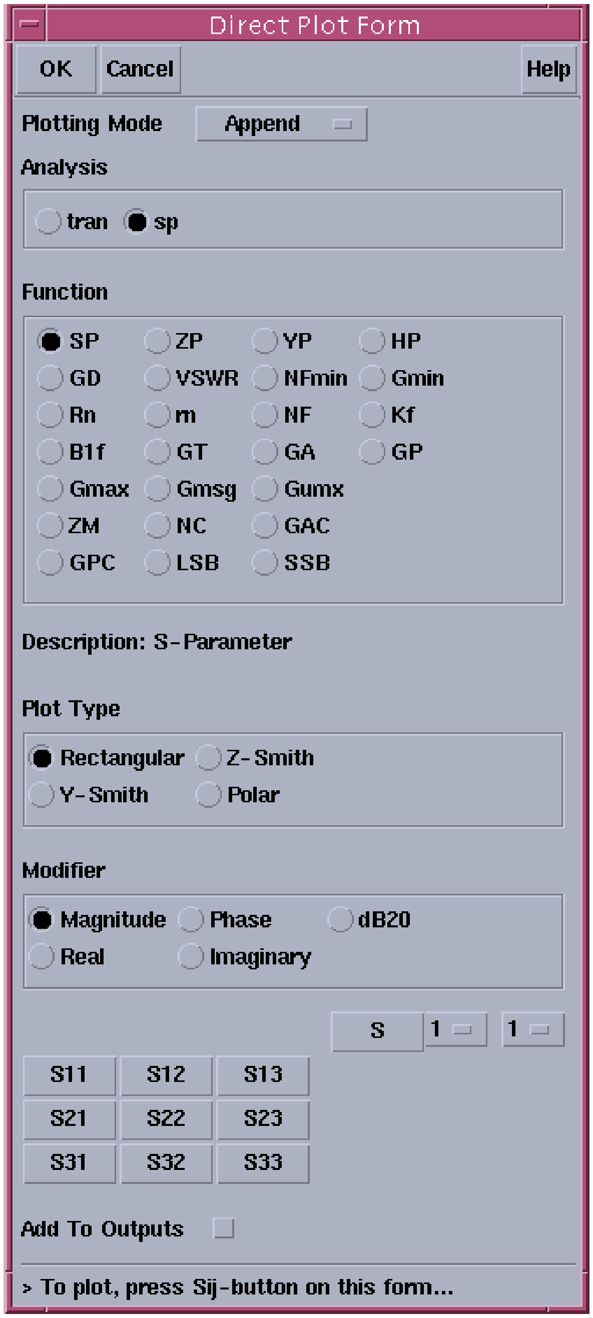
   For S, Z, Y, or H parameters (shown as *SP*, *ZP*, *YP*, and *HP* on the form), generate plots for ports 1 through 3 by clicking the appropriate parameter button at the bottom of the form. To generate plots for any higher-numbered ports, use the cyclic fields beside the buttons to specify the output and incident ports. Then click the *S*, *Y*, *Z*, or *H* button that is next to the cyclic field to plot.

Note: For circuits with three or fewer ports, the form has no cyclic fields.

Using the Direct Plot Main Form

For S-Parameter

Choose *Results – Direct Plot – Main Form*. If you select the Single-Ended mode in the S-parameter analysis form, the following *Direct Plot Form* appears. For Single-Ended mode, Rectangular, Z-Smith, Y-smith and Polar Plot types are available. At the bottom of the form, the available signal types are also available. For detailed information about the form, see “S-Parameter Results” on page 491.



Plotting Mode: Append

Analysis
○ tran ● sp

Function
● SP ○ ZP ○ YP ○ HP
○ GD ○ VSWR ○ NFmin ○ Gmin
○ Rn ○ rn ○ NF ○ Kf
○ B1f ○ GT ○ GA ○ GP
○ Gmax ○ Gmsg ○ Gumx
○ ZM ○ NC ○ GAC
○ GPC ○ LSB ○ SSB

Description: S-Parameter

Plot Type
● Rectangular ○ Z-Smith
○ Y-Smith ○ Polar

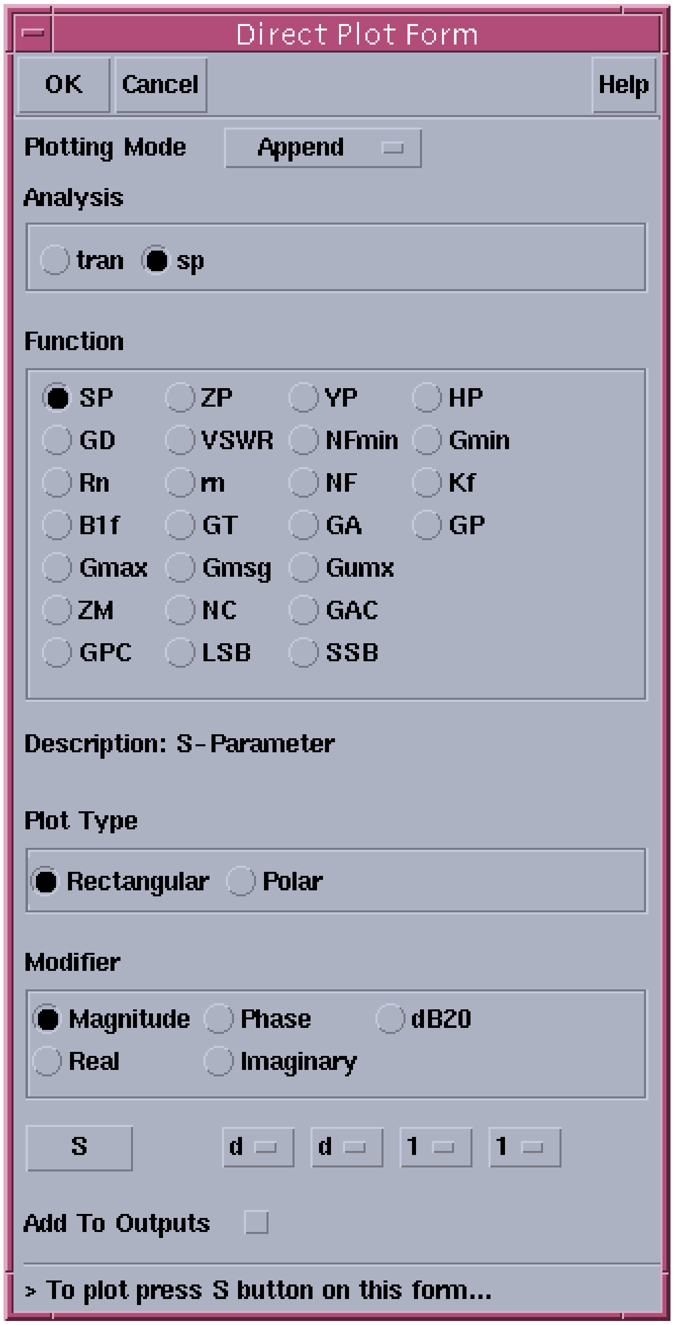
Modifier
● Magnitude ○ Phase ○ dB20
○ Real ○ Imaginary

S 1 1

Add To Outputs □

> To plot, press Sij-button on this form...

Choose *Results – Direct Plot – Main Form*. If you select the Mixed In/out mode in the S-parameter analysis form, the following *Direct Plot Form* appears. For Mixed In/outmode, only Rectangular and Polar Plot types are available. For detailed information about the form, see “S-Parameter Results” on page 491.



Direct Plot Form

Plotting Mode: Append

Analysis

○ tran ● sp

Function

● SP	○ ZP	○ YP	○ HP
○ GD	○ VSWR	○ NFmin	○ Gmin
○ Rn	○ rn	○ NF	○ Kf
○ B1f	○ GT	○ GA	○ GP
○ Gmax	○ Gmsg	○ Gumx	
○ ZM	○ NC	○ GAC	
○ GPC	○ LSB	○ SSB	

Description: S-Parameter

Plot Type

● Rectangular ○ Polar

Modifier

● Magnitude ○ Phase ○ dB20
○ Real ○ Imaginary

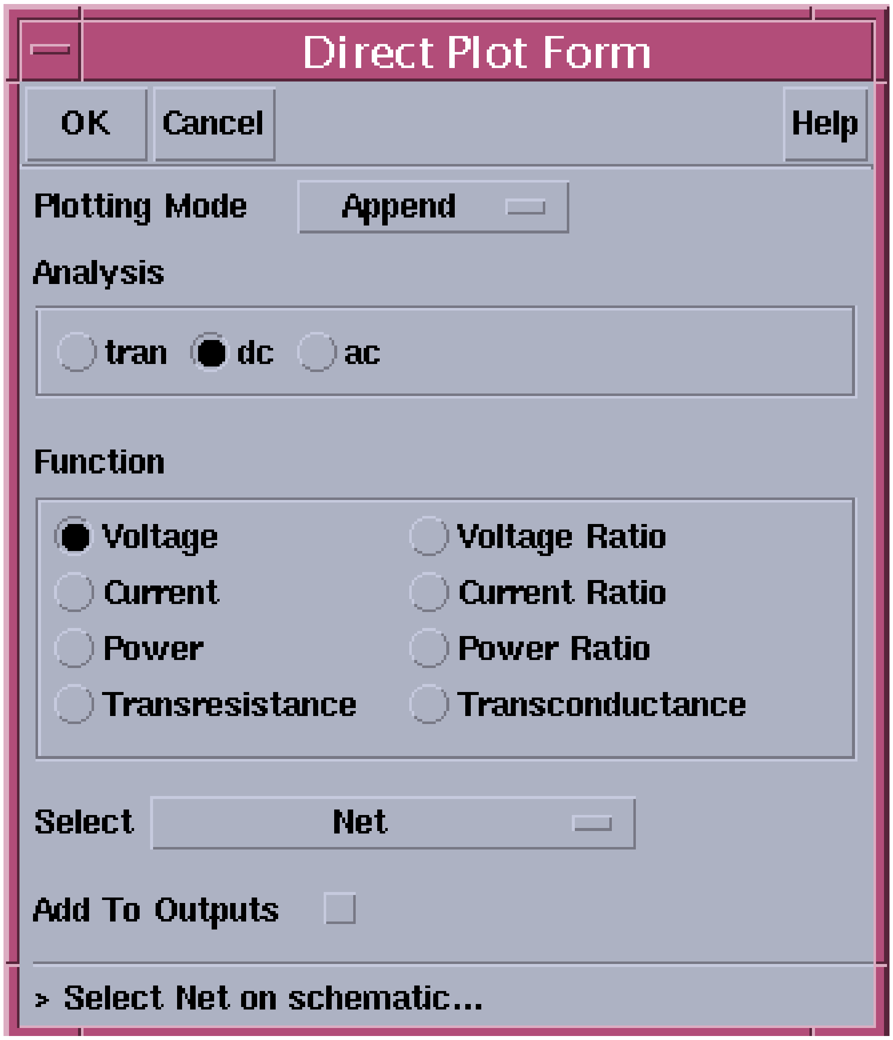
S d d 1 1

Add To Outputs □

> To plot press S button on this form...

For DC

1. Choose *Results – Direct Plot – Main Form*. The *Direct Plot Form* appears. Select the *dc* option in the *Analysis* section.



Direct Plot Form

OK | Cancel | Help

Plotting Mode: Append

Analysis

○ tran ● dc ○ ac

Function

● Voltage	○ Voltage Ratio
○ Current	○ Current Ratio
○ Power	○ Power Ratio
○ Transresistance	○ Transconductance

Select: Net

Add To Outputs □

> Select Net on schematic...

2. Specify the *Plotting Mode*. You can specify:

   - ❑ *Append* to append the new signal to the current graph.

     Note: The *Append* mode is not recommended for plots with different scales and units for the X axis. It can give strange results because *WaveScan* opens a new subwindow to plot any data that does not match the units and range of the existing traces.

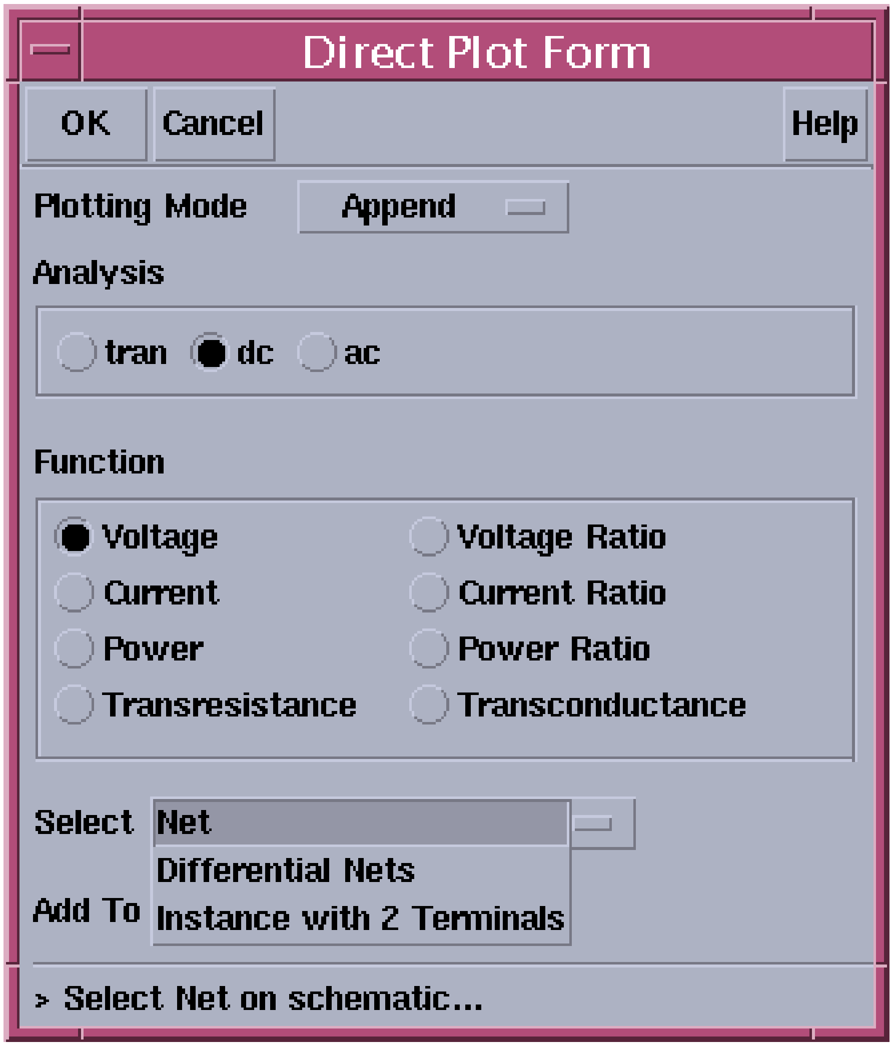
   - ❑ *Replace* to replace the current graph with the new signal. This is the default option.

   - ❑ *New SubWin* to plot the signal in a new subwindow.

- ❑ *New Win* to plot the signal in a new window.

3. The functions that are available are: *Voltage*, *Voltage Ratio*, *Current*, *Current Ratio*, *Power, Power Ratio, Transresistance* and *Transconductance*. Based on the selected function and available data, the form changes dynamically to display the applicable options.

4. Choose the nets and terminals to plot. You can select *Net*, *Differential Nets* or *Instance with 2 Terminals*.



Plotting Mode: Append

Analysis

○ tran ● dc ○ ac

Function

● Voltage ○ Voltage Ratio
○ Current ○ Current Ratio
○ Power ○ Power Ratio
○ Transresistance ○ Transconductance

Select: Net
Differential Nets
Instance with 2 Terminals

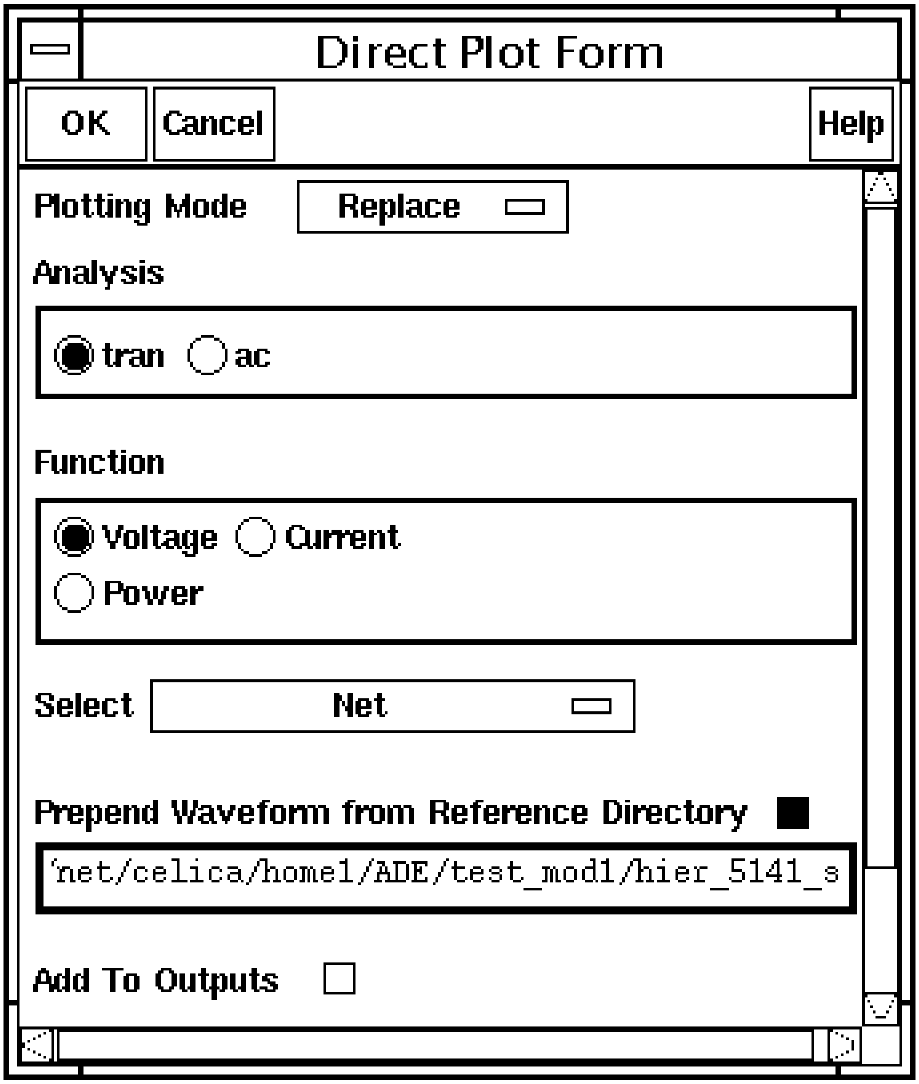
Add To

> Select Net on schematic...

5. Enable *Add To Outputs* to add expressions for the results to the outputs section and plot in the mode that you selected.

For Transient Results

1. Choose *Results – Direct Plot – Main Form*. The *Direct Plot Form* appears. Select the *tran* option in the *Analysis* section.



Plotting Mode: Replace

Analysis: ● tran ○ ac

Function: ● Voltage ○ Current ○ Power

Select: Net

Prepend Waveform from Reference Directory ■

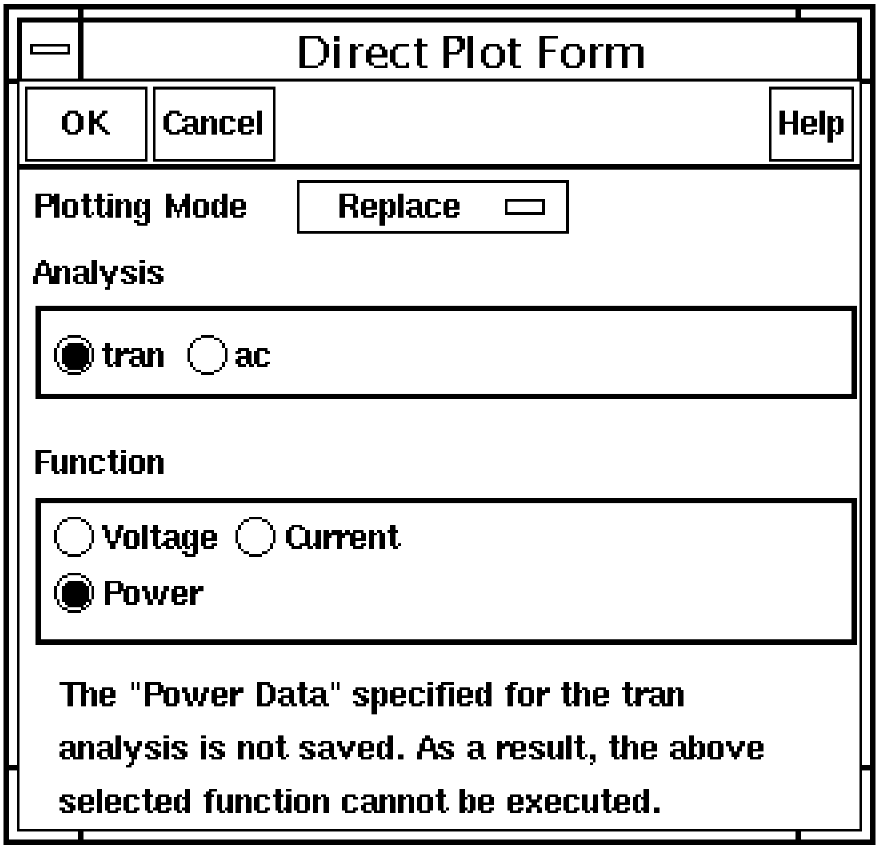
‘net/celica/home1/ADE/test_mod1/hier_5141_s

Add To Outputs □

2. Specify the *Plotting Mode*. You can specify:

   - *Append* to append the new signal to the current graph.
   - *Replace* to replace the current graph with the new signal. This is the default option.
   - *New SubWin* to plot the signal in a new subwindow.
   - *New Win* to plot the signal in a new window.

3. The functions that are available are: *Voltage*, *Current* and *Power*. Based on the selected function and available data, the form changes dynamically to display the applicable options. Most of the options are similar to those available in the *Direct Plot* form For DC analysis. If no power data is available, a corresponding message is displayed on the form:



Plotting Mode Replace

Analysis

● tran ○ ac

Function

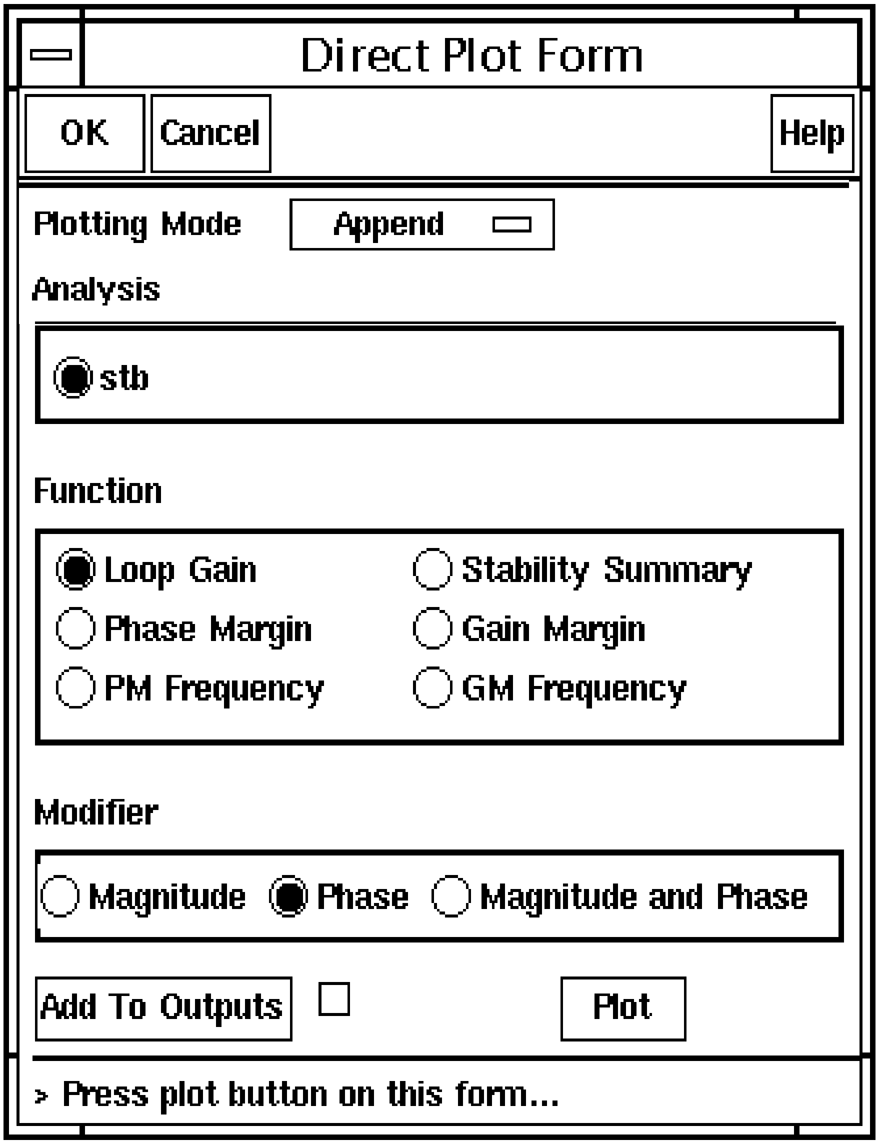
○ Voltage ○ Current
● Power

The "Power Data" specified for the tran analysis is not saved. As a result, the above selected function cannot be executed.

4. The *Prepend Waveform from Reference Directory* option can be used for appending multiple checkpoint/restart transient waveforms together to enable you to view complete waveforms. Specify the reference results directory(s) in the field. The signal you choose in your direct plot will then be accessed from the reference directory.

For Stability Results

1. Choose *Results – Direct Plot – Main Form*. The *Direct Plot Form* appears. Select the *stb* option in the *Analysis* section. The form re-displays accordingly.
.



Plotting Mode: Append

Analysis

● stb

Function

● Loop Gain ○ Stability Summary
○ Phase Margin ○ Gain Margin
○ PM Frequency ○ GM Frequency

Modifier

○ Magnitude ● Phase ○ Magnitude and Phase

Add To Outputs □

> Press plot button on this form...

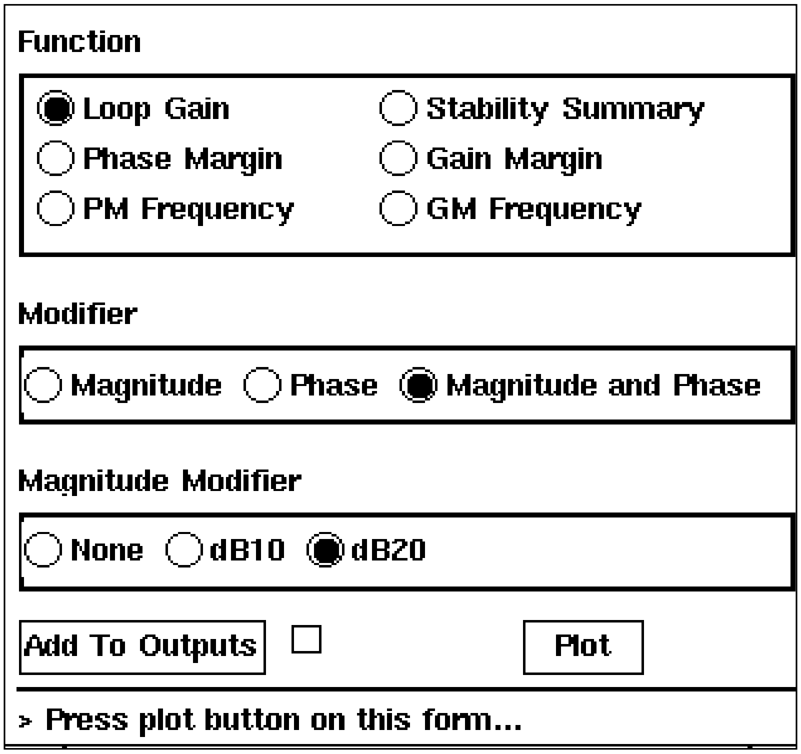
2. Specify the *Plotting Mode*. You can specify:

- ❑ *Append* to append the new signal to the current graph.
- ❑ *Replace* to replace the current graph with the new signal. This is the default option.
- ❑ *New SubWin* to plot the signal in a new subwindow.

- ❑ *New Win* to plot the signal in a new window.

3. The functions that are available are: *Loop Gain*, *Stability Summary*, *Phase Margin*, *Gain Margin*, *PM Frequency* and *GM Frequency*. Based on the selected function and available data, the form changes dynamically to display the applicable options.

a. When you select *Loop Gain*, the form re-displays to show the *Modifier* section.The loop gain output is a complex waveform and you can select it to plot *Magnitude*, *Phase* or both (*Magnitude and Phase*). Whenever you choose to plot *Magnitude*, the *Magnitude Modifier* section appears on the form. You can select *None*, *dB10* or *dB20*, as needed. Whenever you plot both the magnitude and phase, the waveform window changes to the strip mode. It reverts back to the composite mode for other plot operations



Function

● Loop Gain ○ Stability Summary
○ Phase Margin ○ Gain Margin
○ PM Frequency ○ GM Frequency

Modifier

○ Magnitude ○ Phase ● Magnitude and Phase

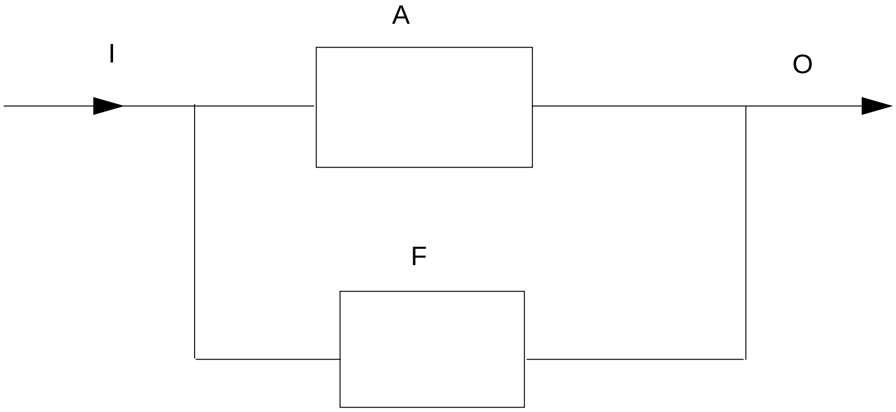
Magnitude Modifier

○ None ○ dB10 ● dB20

> Press plot button on this form...

Note: There is a difference between the Artist and Spectre definition of loop gain.

For the feedback circuit shown below:



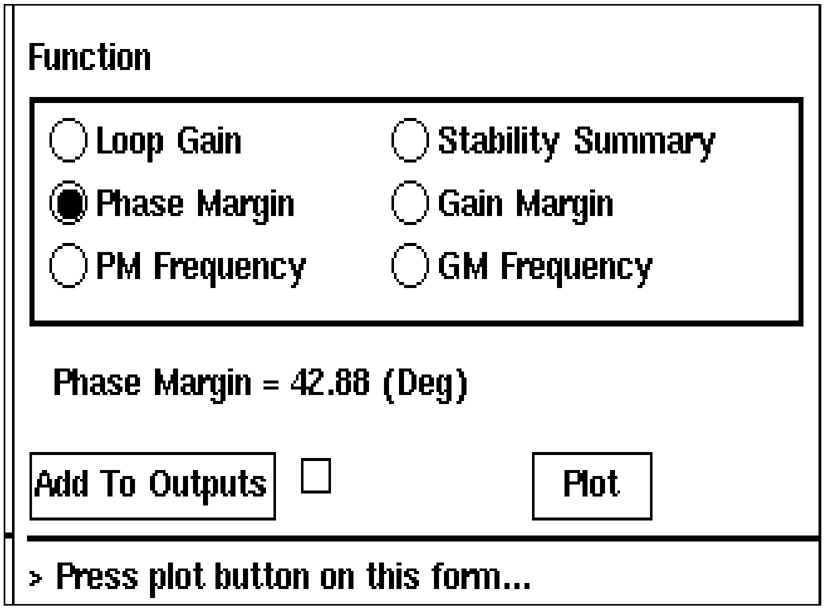
The closed loop gain is defined as:

$$\frac{O}{I} = \frac{A}{1 - AF}$$

The Spectre output defines loop gain as the product **AF**, while others (like the Artist Calculator `phaseMargin` and `gainMargin` functions) define **-AF** as the loopGain. Therefore, to obtain the same results from Artist, you need to negate the Spectre's loopGain as illustrated below:

```
gainMargin( -1 * getData( "loopGain" ?result “stb” ), 1)
phaseMargin( -1 * getData( "loopGain" ?result “stb” ) )
```

b. *Phase Margin*, *Gain Margin*, *PM Frequency* and *GM Frequency* constitute the margin data. This information is calculated from the loop gain data for the circuit. The information is only available when frequency is swept in the stability analysis and the swept range is sufficient to calculate the values. When the selected margin data is scalar the values are displayed on the form itself.



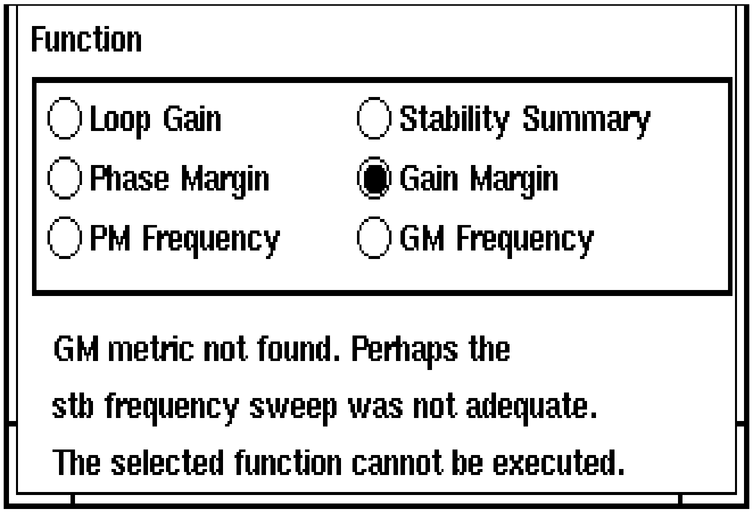
Function

○ Loop Gain ○ Stability Summary
● Phase Margin ○ Gain Margin
○ PM Frequency ○ GM Frequency

Phase Margin = 42.88 (Deg)

> Press plot button on this form...

When the swept frequency range is not sufficient to calculate the selected margin data an error is reported in the *Direct Plot Form* and the *Plot* and *Add to Outputs* button are not available.

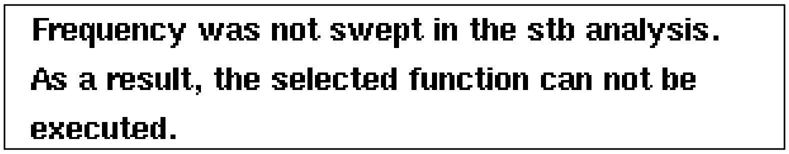


Function

○ Loop Gain ○ Stability Summary
○ Phase Margin ● Gain Margin
○ PM Frequency ○ GM Frequency

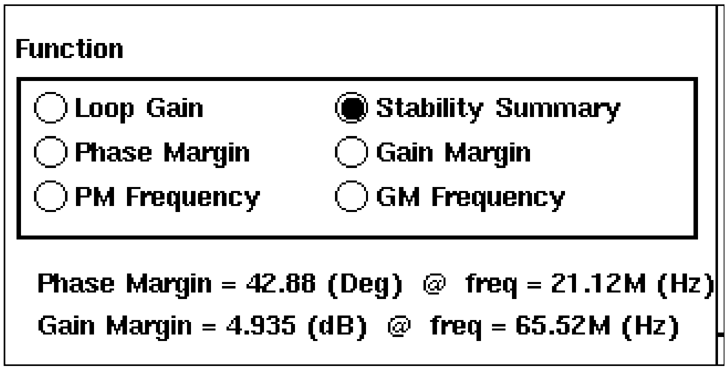
GM metric not found. Perhaps the
stb frequency sweep was not adequate.
The selected function cannot be executed.

When frequency is not swept in the stability analysis and you choose any of the margin data functions an error is reported in the *Direct Plot Form* and the P*lot* and *Add to Outputs* button are not available.



Frequency was not swept in the stb analysis.
As a result, the selected function can not be executed.

c. Selecting *Stability Summary* displays all the margin data collectively on the form, when the data is scalar. You do not have the facility to plot or add the four outputs when this function is chosen. Use the individual margin data function for this operation.

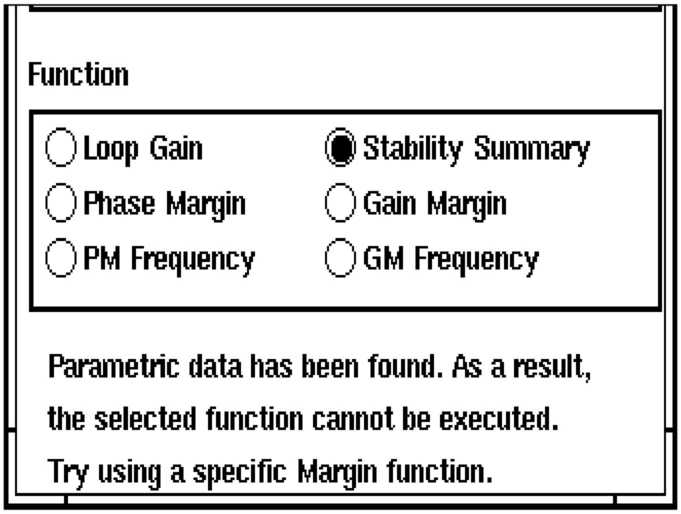


Function

○ Loop Gain	● Stability Summary
○ Phase Margin	○ Gain Margin
○ PM Frequency	○ GM Frequency

Phase Margin = 42.88 (Deg) @ freq = 21.12M (Hz)
Gain Margin = 4.935 (dB) @ freq = 65.52M (Hz)

When frequency was not swept or the margin data is not scalar an appropriate error is reported in the *Direct Plot Form* and the P*lot* and *Add to Outputs* button are not available.



Function

○ Loop Gain ● Stability Summary
○ Phase Margin ○ Gain Margin
○ PM Frequency ○ GM Frequency

Parametric data has been found. As a result, the selected function cannot be executed. Try using a specific Margin function.

4. Enable *Add To Output*s and plot in the mode that you selected.

Note: This option makes no checks for duplication in outputs.

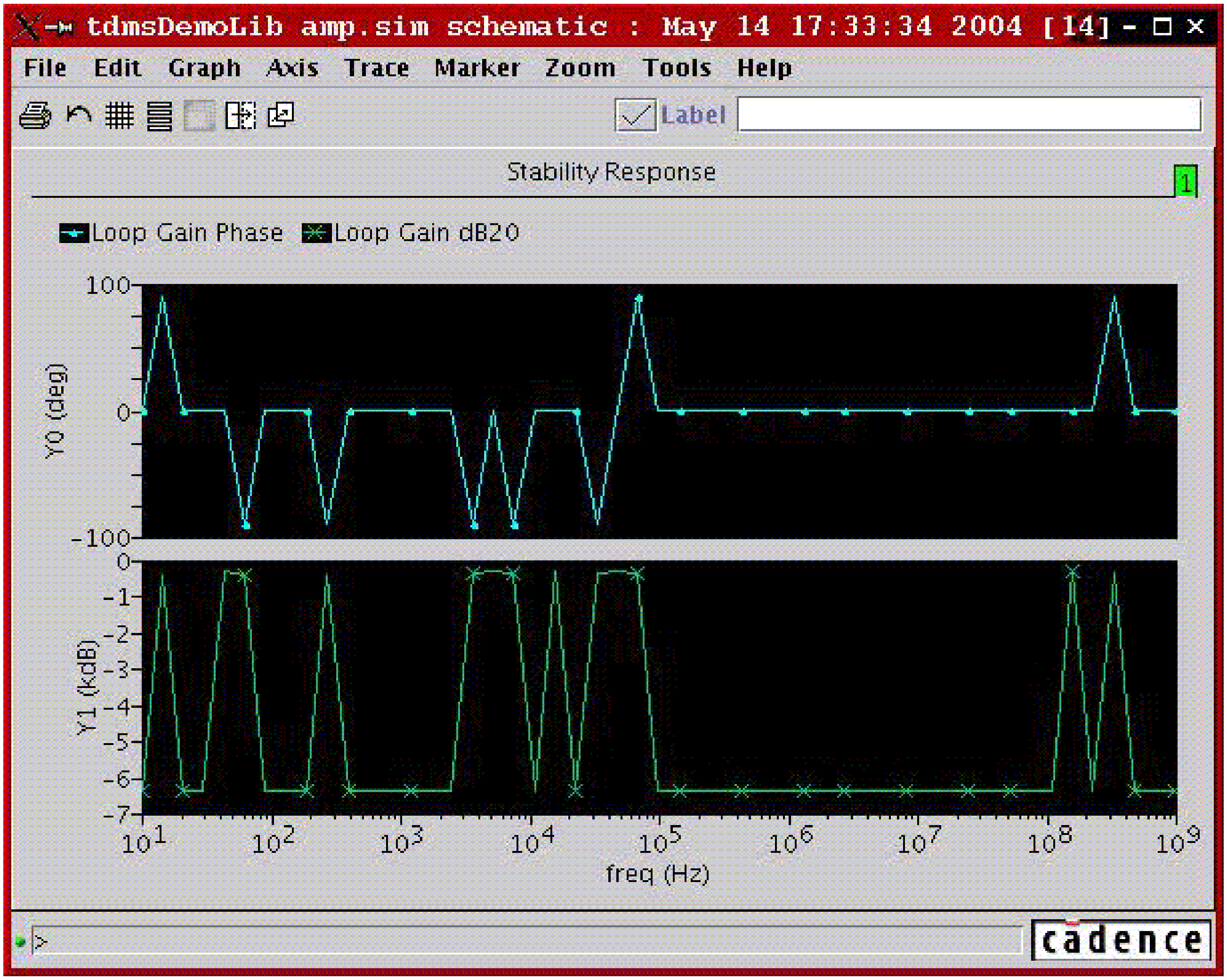
All other parts of the *Direct Plot* form work the same way as they do for other analyses. Refer to the *Spectre RF User Guide* for details.

This form handles parametric (family) data. The *Loop Gain* would be a set of curves for family data. Similarly for non-parametric data, *Phase Margin* and *Gain Margin* will be scalars. A horizontal straight line will be plotted for them.

Example

A wave form window when plotted with Magnitude and phase (dB20) for non-family data is displayed. The expression/waveform names created for the outputs are: `Loop Gain`, `Loop`

`Gain dB10`, `Loop Gain db20`, `Loop Gain Phase`, `Gain Margin`, `Phase Margin`, `Gain Margin Frequency` and `Phase Margin Frequency`.

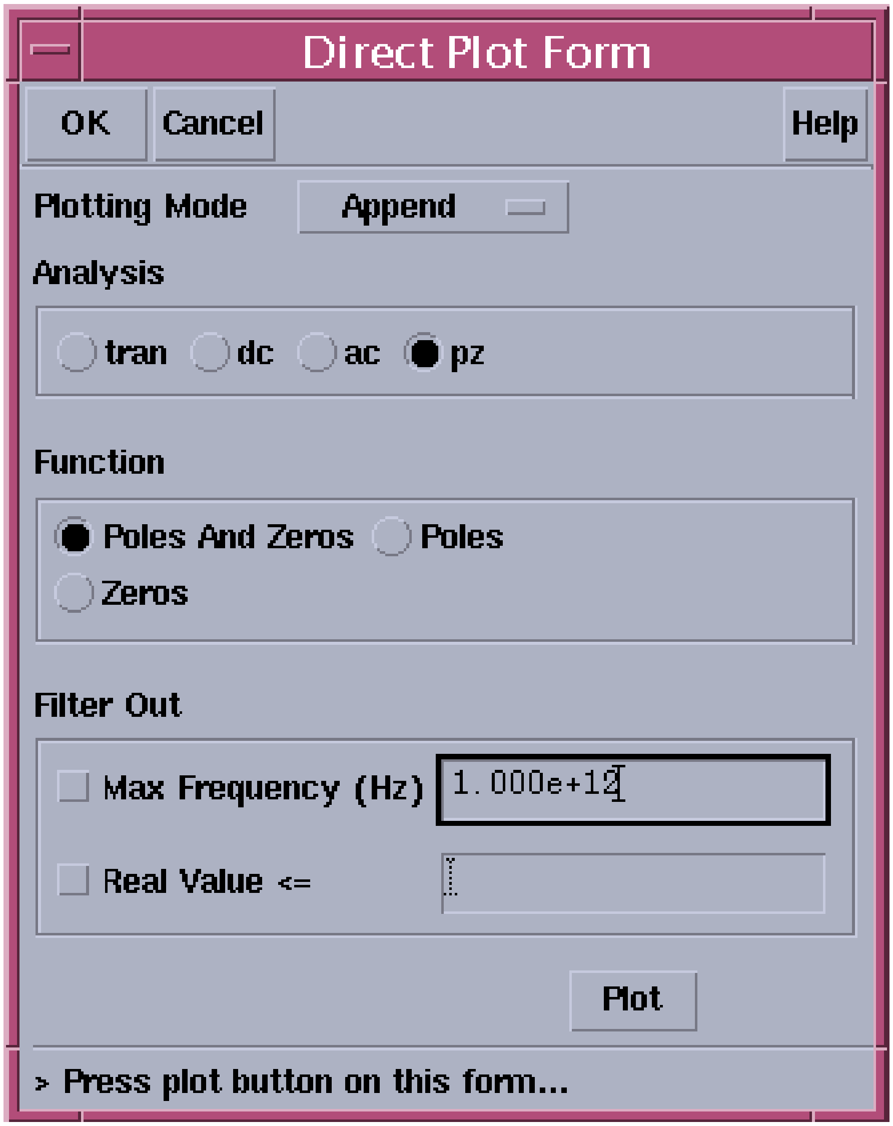


For Pole Zero Results

Once you run a simulation for *Pole Zero* analysis, you can use the *Direct Plot* main form to view the poles and zeros plotted on the real/imaginary plane in the *Analog Waveform Display* window.

1. To access the *Direct Plot* main form, select *Results – Direct Plot – Main Form*.

2. Select the *Pole Zero Analysis* option. The *Direct Plot Form* changes dynamically to display the applicable functions and options:



Direct Plot Form

Plotting Mode: Append

Analysis

○ tran ○ dc ○ ac ● pz

Function

● Poles And Zeros ○ Poles
○ Zeros

Filter Out

□ Max Frequency (Hz) 1.000e+12

□ Real Value <=

> Press plot button on this form...

3. Select the option, *Poles* if you want to plot only poles, *Zeros* if you want to plot only zeros and *Poles and Zeros* if you want to plot both poles and zeros.

   Note: By default, the option *Poles and Zeros* is selected.

4. Set the required options in the *Filter Out* section and click *OK*. This section provides a combination of filtering mechanisms that you can select in order to plot the poles and zeros. These are:

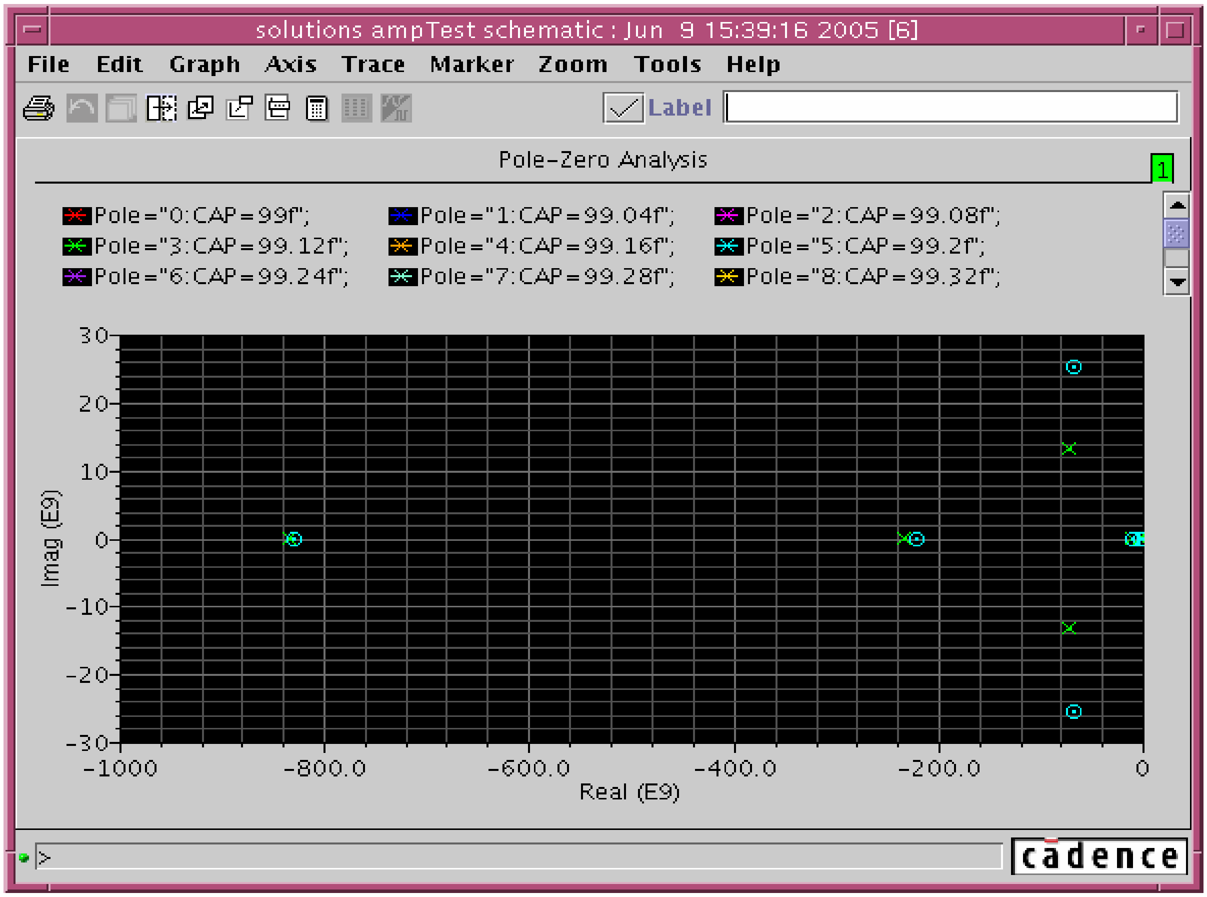
   ❑ *Max Frequency:*
   This option enables you to filter out poles and zeros that are outside the frequency

band of interest (FBOI) and that do not influence the transfer function in the FBOI. The default value is that specified in the *fmax* field in the *Pole-Zero Options* form. Only poles and zeros whose magnitudes exceed the frequency value specified are filtered out.

- *Real Value:*
  This option enables you to specify the real part of the frequency. Only poles and zeros whose real values are less than or equal to the real value specified are filtered out.

Note: By default, no filtering is selected. You can set the filtering criteria once you specify either poles or zeros or both to be plotted.

5. Click the *Plot* option to view the results in the *Waveform Window*:



Poles and zeros are plotted in *scatter* mode. This implies that poles and zeros are plotted individually but not connected. Poles are represented by the symbol **x** and zeros by the symbol **o**. The complex data is plotted with poles and zeros.

Non-Swept Parameters

For the non-swept case, the result of Pole Zero analysis will be two waveform objects, one representing poles and another representing the zeros. The two wave objects are plotted in the same color however, *poles* will be represented by the symbol **x** and *zeros* by the symbol **o**.

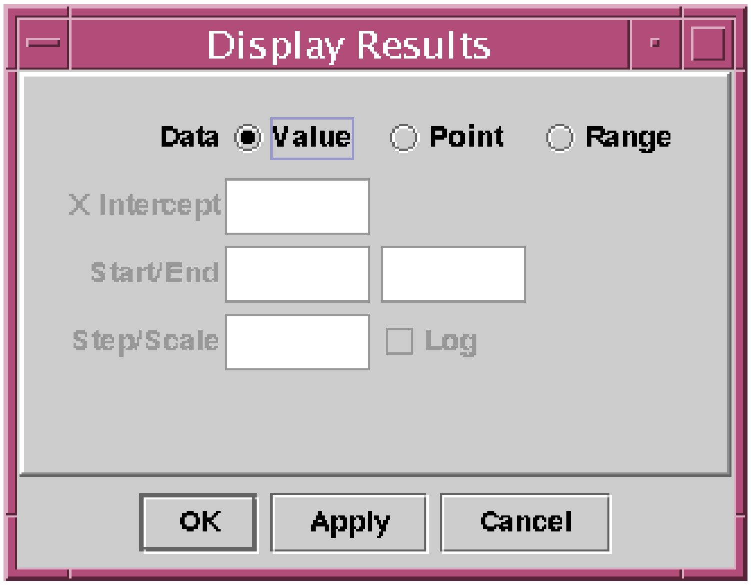
Swept Parameters

For swept parameter Pole Zero Analysis, it is possible to create the root-locus plot. Instead, the poles and zeros are plotted corresponding to each *Swept Parameter* value.

Overview of Printing

The following commands and tools within the Analog Design Environment print text results and reports to the Results Display Window.

- *Results – Print* menu commands in the Simulation window
- The *Tabular Results Display* (▦) button in *WaveScan Calculator.* This brings up the *Display Results* window:



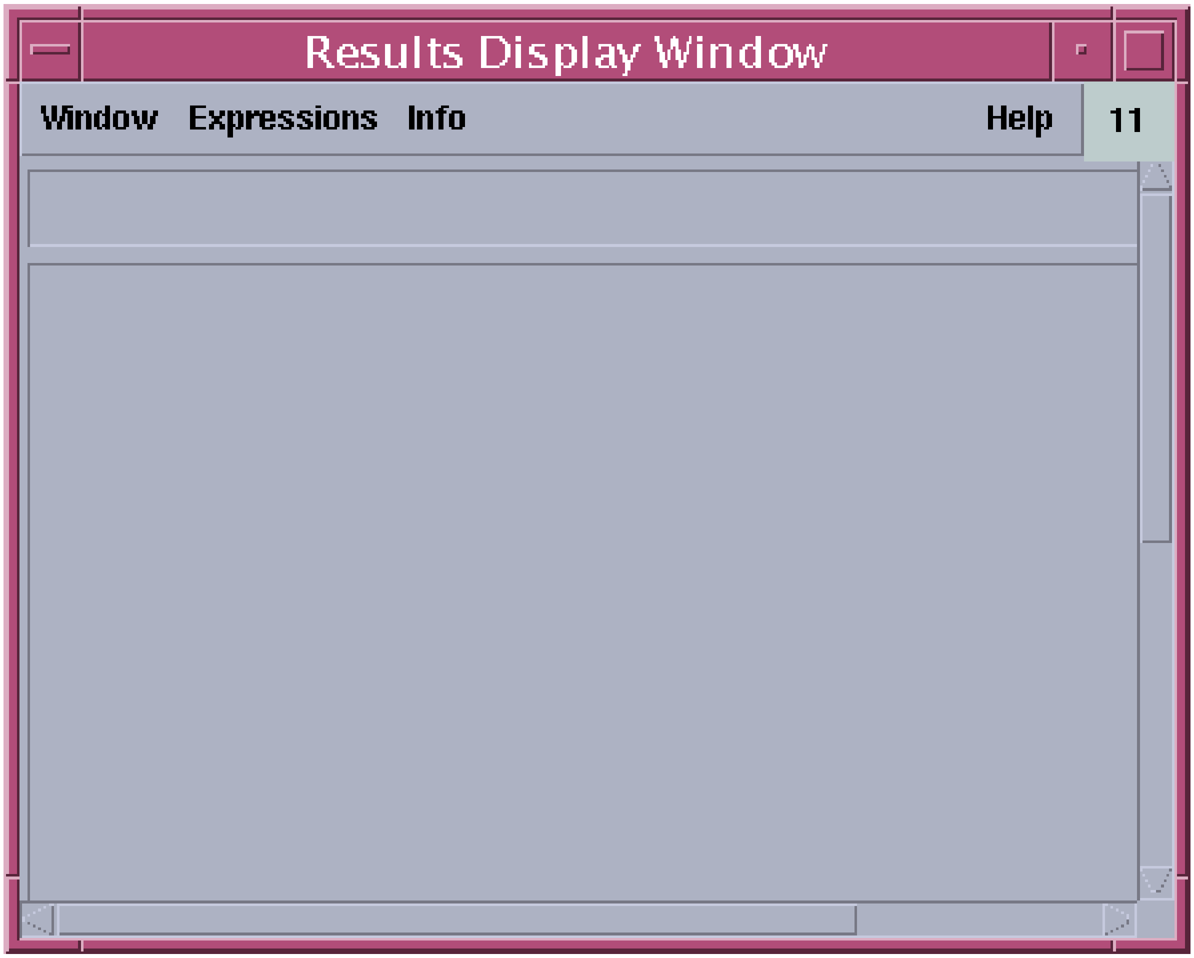
Data ● Value ○ Point ○ Range

If your waveform viewer is AWD, use the *Print* and *printvs* commands in Calculator

- *Statistics – Print* menu commands in the Statistics tool

- The *Markers* menu options in WaveScan Calculator. If your waveform viewer is AWD, use the *Display Intercept Data* (*Marker – Vertical Marker – Display Intercept Data)* commands in the AWD graph window.

Using any of these commands opens the Results Display Window.



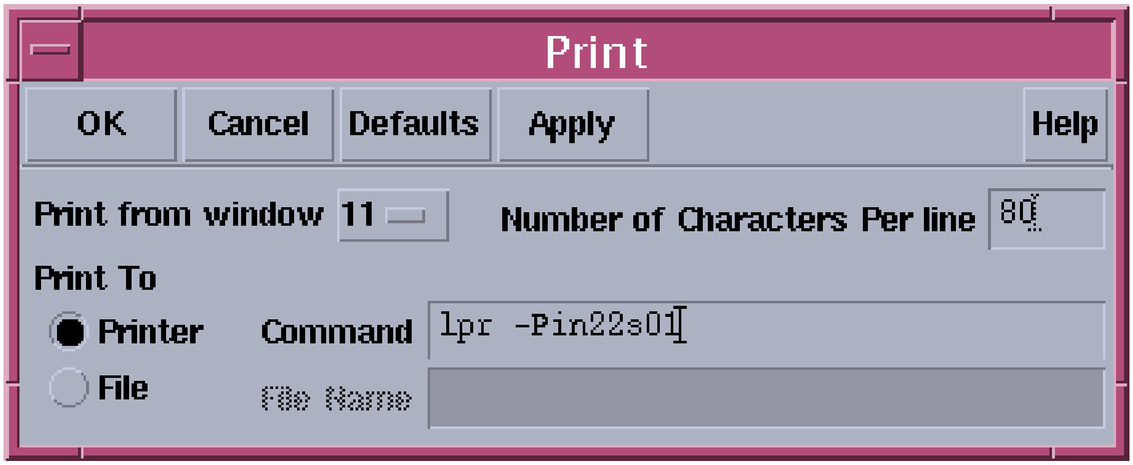
For guidance on using the Results Display Window to perform tasks, see the following sections.

Printing Results

To print the results in the Results Display Window either in hardcopy or to a file,

1. Choose *Window – Print*.

The Print form appears.



Print from window 11
Number of Characters Per line 80
Print To
● Printer Command lpr -Pin22s01
○ File File Name

2. Choose the correct window number from the *Print from window* cyclic field.

   This is the window containing the contents you want to print.

3. Type a value in the *Number of Characters Per Line* field.

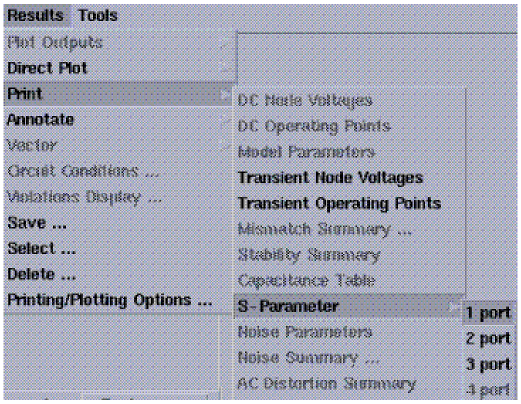
4. Choose either the *Printer* or *File* radio button in the *Print To* field.

   You must type a filename if you choose *File*.

5. Click *OK*.

S-parameter - Print

1. *Results – Print –* Parameter displays the print options available when single mode is selected in the S-parameter analysis form.



Similarly, when you select the Mixed In/Out mode for the S-parameter, print S-parameter sub menu shows 1, 2 and 3 ports are disabled. Only 4 port is enabled.

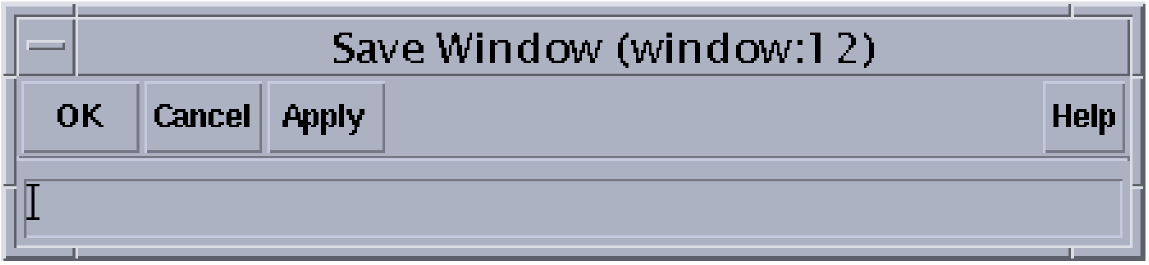
Saving State

You can use *Save State* and *Load State* capability to save the current setup of display options for printing waveforms such as printing format, setting a printing range if the amount of data is too large, printing at a certain interval, and changing the order of the display. You can save the state of the window into a file. Later if you run another simulation and do *Load State*, the new data can be loaded back and displayed as you specified when you saved the state. *Save State* and *Load State* are applicable only to waveforms (that is, expressions that can evaluate to a waveform). If you print out a single number, like a node voltage, these commands are disabled. You get a message stating this value is not a waveform and cannot be loaded back.

To save the contents and format of a Results Display Window,

1. Choose *Window – Save State*.

   The Save Window form appears.



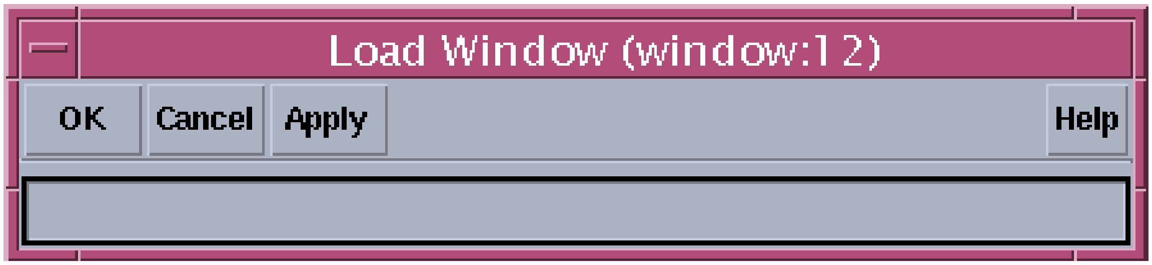
2. Type a filename in the field.
3. Click *OK*.

Loading State

To load a window state that you previously saved,

1. Choose *Window – Load State*.

The Load Window form appears.



2. Type the name of the saved file in the field.
3. Click *OK*.

Updating Results

To update the Results Display Window with results from a new simulation,

➤ Choose *Window – Update Results*.

This updates the data using the current window setup. *Update Results* is applicable only to waveforms (that is, expressions that can evaluate to waveforms). If you print out a single number, like a node voltage, this command is disabled.

Making a Window Active

There is no limit to the number of Results Display Windows you can have open, but only one window is active at a time. All printouts go to the active window.

To make a window active,

➤ Choose *Window – Make Active* in the window that you want to make active.

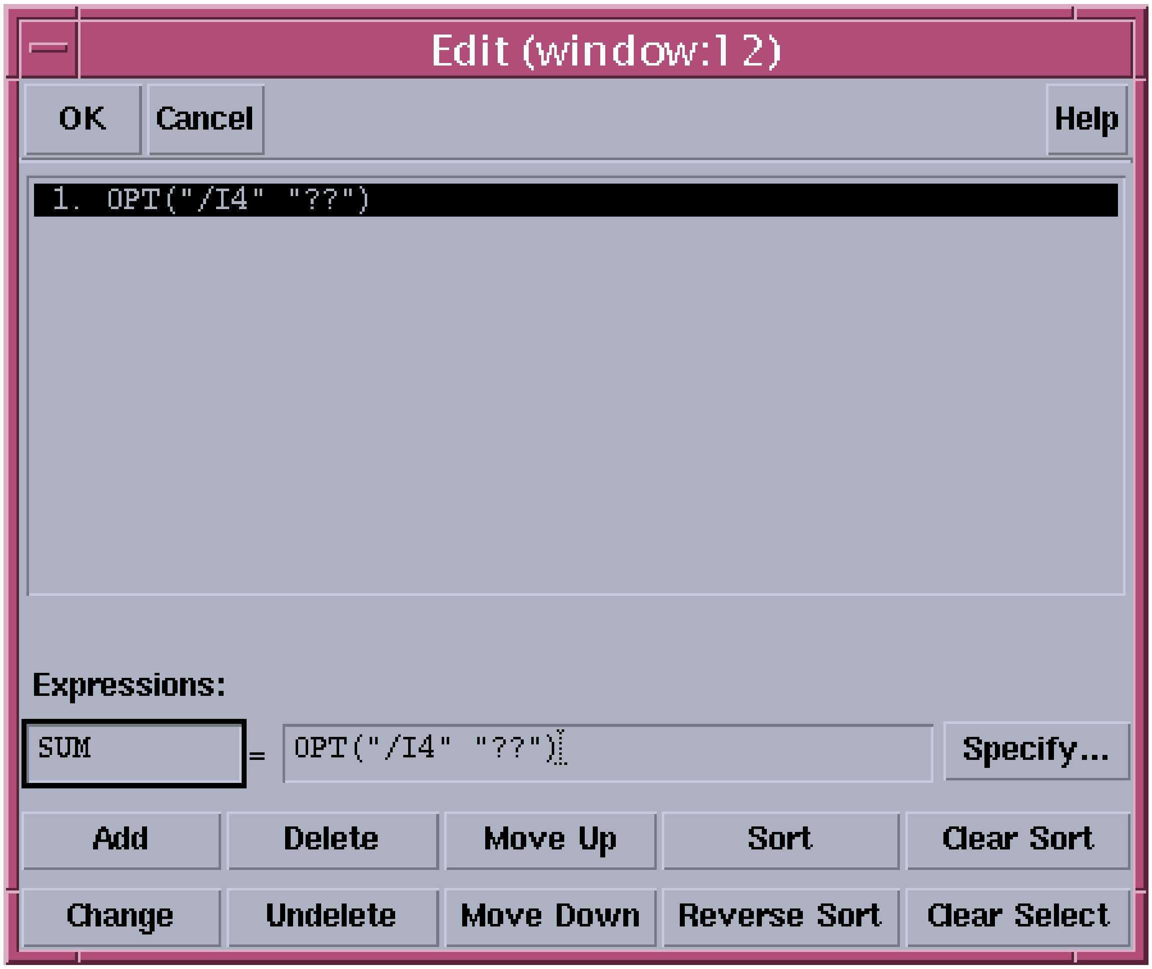
Editing Expressions

You can edit any expressions that evaluate to waveforms (for example, DC operating parameters, model parameters, and transient operating parameters). If you print only one value, the edit menu choices are not available. The editing commands operate on only the last table in the active Results Display Window.

To edit expressions in the print window,

1. Choose *Expressions – Edit*.

The Edit window appears.



```
1. OPT("/I4" "??")
```

```
SUM = OPT("/I4" "??")
```

2. Edit the expressions using the form buttons and fields.

Expressions The field to the left of the equal sign shows the aliased name of the expression to the right. Naming expressions is optional and the field might be blank. If aliases are used, they are shown in the list box and the title line of the print window list box.

Specify Retrieves the expression in the calculator buffer and places it into the edit field.

Add Adds the expression in the edit field to the list box.

Change Replaces the selected expression in the list box with the one in the edit field.

Delete Deletes the selected expression from the list box.

Undelete Lets you undo the last delete.

Move Up Moves the display of the selected expression one step to the left. If the expression is already the leftmost, it is moved to the rightmost.

Move Down Moves the display of the selected expression one step to the right. If the expression is already the rightmost, it is moved to the leftmost.

Sort Sorts the selected expression so that the value increases down the column.

Reverse Sort Sorts the selected expression so that the value decreases down the column.

Clear Sort Reverts to the default order.

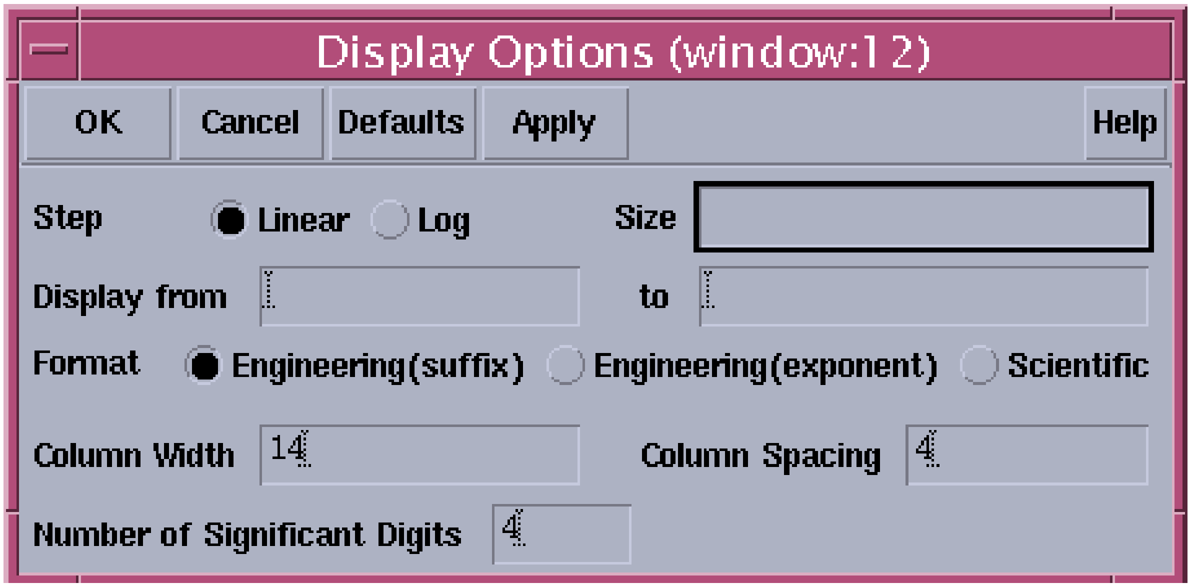
Clear Select Clears the selection in the list box. Also, you can clear entries from the list box by clicking on the entry while holding down the `Control` key.

Setting Display Options

To change the display options,

1. Choose *Expressions – Display Options*.

The Display Options form appears.



Step ● Linear ○ Log   Size

Display from   to

Format ● Engineering(suffix) ○ Engineering(exponent) ○ Scientific

Column Width 14   Column Spacing 4

Number of Significant Digits 4

2. Type the values into the form and select a format.

Step size
Specifies the interval for printing data.

Display from, to
Specifies the range of data to print. If *from* is left blank, the data is printed from the beginning. If *to* is blank, the data is printed to the end. You can set the print range only after printing data.

Format
Controls the format of the data printed. The possible formats are *Engineering Suffix* (default), *Engineering*, and *Scientific*. For example, you represent 0.0001 as 0.1m (engineering suffix), 0.1e-3 (engineering), or 1e-4 (scientific).

Linear/Log
Specifies whether the scale used for step size is linear or logarithmic.

Column width/spacing
Changes the number of characters allowed for column width and spacing. The default width is 14 characters. The allowed range is 4 to 20 characters. The default spacing is 4 and the allowed range is 1 to 10.

Number of significant digits
Specifies the number of significant digits to be printed. The default is 4 digits, and the allowed range is 2 to 10.

Note: If the Results Display Window contains more than one type of results, the *Display Options* commands apply only to the last result (if the last result can evaluate to a waveform). After the data is edited, only the last result appears in the window. If you want to preserve the previous results, you can open a new Results Display Window and print the results to be edited in the new window.

Displaying Output Information

To display output information,

➤ Choose *Info – Show Output*.

Output names are truncated to fit into columns if they are too long. The *Show Output* command shows the output names in full.

Specifying Results to Print

Before you can print results, you need to specify which results to print.

1. Do one of the following:
    - ❑ Run a simulation.
    - ❑ In the Simulation window, choose *Results – Select*, choose the desired data file, and click *OK*.
2. Make sure the Schematic window for the selected design is open.

To print results for the current simulation or for a selected data file,

1. Choose a print command from the *Results* menu.
2. Select a node in the Schematic window.

    The Results Display Window shows

    - ❑ The command syntax for the print option you selected
    - ❑ The results for the instance you selected

    Each time you click a node in the Schematic window, information about the node is added to the Results Display Window.

Printing DC Operating Points

To print the DC operating points of the components in your circuit,

1. Choose *Results – Print – DC Operating Points*.
2. Move your cursor into the Schematic window.

   The CIW prompts you to select instances for the operating point output.
3. Click an instance.

If the selected instance is a textual subcircuit, operating points for all devices in the subcircuit will be printed. It may take some time to search for all instances in a textual subcircuit. To disable the feature, set the following environment variable in your `.cdsenv`:

```
asimenv.printing  printInlines  boolean  nil
```

Printing Transient Operating Points

To print the final transient operating points of the nodes or components in your circuit,

1. Choose *Results – Print – Transient Operating Points*.
2. Move your cursor into the Schematic window.

   The CIW prompts you to select instances for the transient operating point (OPT) output.
3. Click an instance or node.

If the selected instance is a textual subcircuit, operating points for all devices in the subcircuit will be printed. It may take some time to search for all the instances in a textual subcircuit.To disable the feature, set the following environment variable in your `.cdsenv`:

```
asimenv.printing  printInlines  boolean  nil
```

Printing Model Parameters of Components

To print the model parameters of the nodes or components in your circuit,

1. Choose *Results – Print – Model Parameters*.
2. Move your cursor into the Schematic window.

   The CIW prompts you to select instances for the model parameter output.

3. Click an instance of a device.

If the selected instance is a textual subcircuit, model parameter for all devices in the subcircuit will be printed. It may take some time to search for all instance in a textual subcircuit. To disable the feature, set the following environment variable in your `.cdsenv`:

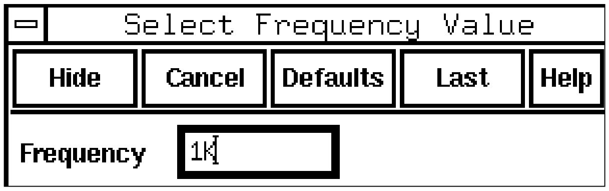
```
asimenv.printing  printInlines  boolean  nil
```

Printing Noise Parameters of Nodes or Components

To print the noise parameters of the nodes or components in your circuit,

1. Choose *Results – Print – Noise Parameters*.

   The Select Frequency Value form appears.



Frequency 1K

   If the form does not appear, press F3.

2. In the *Frequency* field, type the frequency value at which you want the noise parameters to print.

   The default frequency is 1K.

3. Move your cursor into the schematic window.

   The CIW prompts you to select instances for the VNP output.

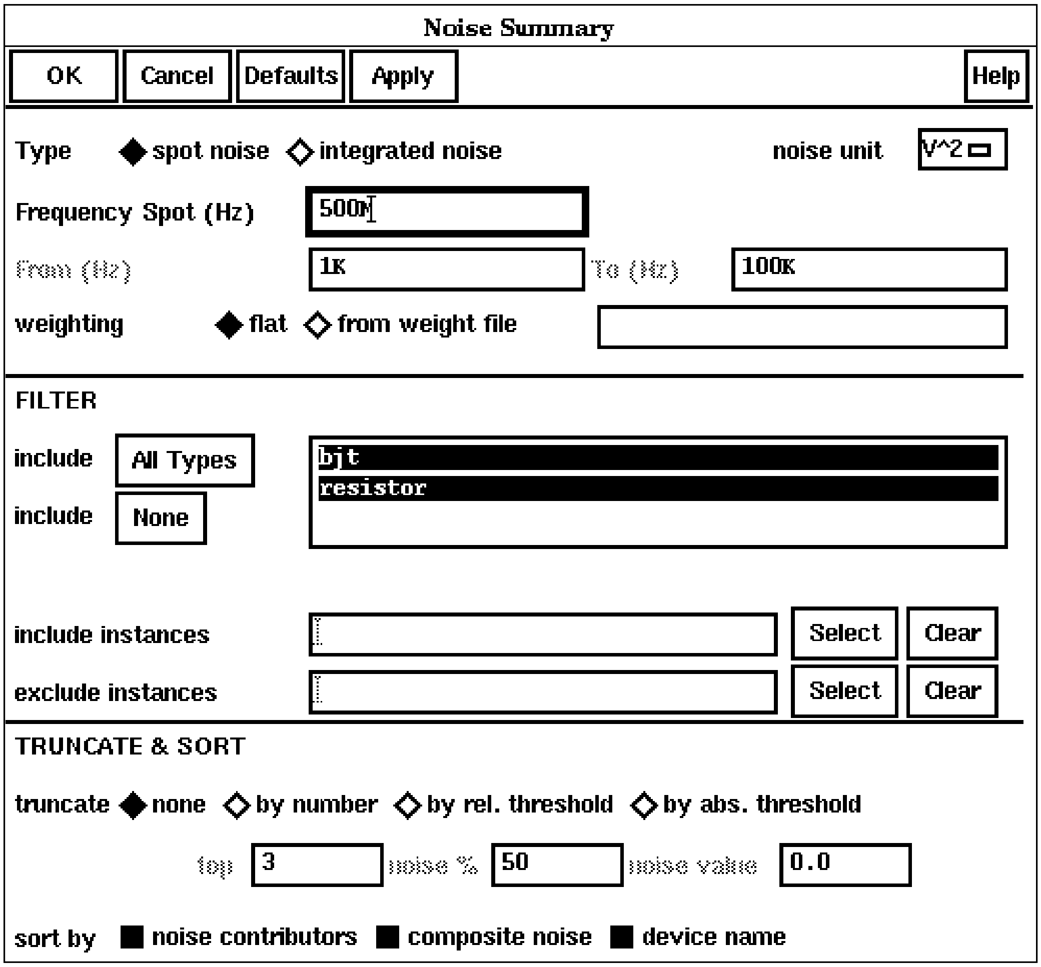
4. Click an instance or node.

Noise Summary

To display the noise contribution of the components in a circuit,

1. Run a noise analysis simulation.
2. Choose *Results – Print – Noise Summary*.

The Noise Summary form appears.



Noise Summary

Type ◆ spot noise ◇ integrated noise

noise unit V^2

Frequency Spot (Hz) 500M

From (Hz) 1K To (Hz) 100K

weighting ◆ flat ◇ from weight file

FILTER

include All Types

include None

bjt
resistor

include instances

exclude instances

TRUNCATE & SORT

truncate ◆ none ◇ by number ◇ by rel. threshold ◇ by abs. threshold

top 3 noise % 50 noise value 0.0

sort by ■ noise contributors ■ composite noise ■ device name

For detailed information about the form, see “Noise Summary” on page 496.

3. Choose either *spot noise* and its frequency or *integrated noise* and a range of frequencies.
4. If you choose *integrated noise*, you have the option of using a weighting factor.

   The *flat* weighting factor specifies that the integration be performed on the original unweighted waveform.

   The *from weight file* selection specifies that, before the integration is performed, the noise contributions of particular frequencies in the original waveform be weighted by factors supplied from an input file. The weighting file must have one of the following entries on the first line: `db`, `mag`, `dbl`, `DB`, `MAG`, `DBL`. Each additional line must contain a pair of X and Y values. All the pairs together must define a function. For example:

```
mag
1          .001641
60         .001641
100        .007499
200        .05559
```

5. Choose filtering details to include or exclude particular instances in your summary.

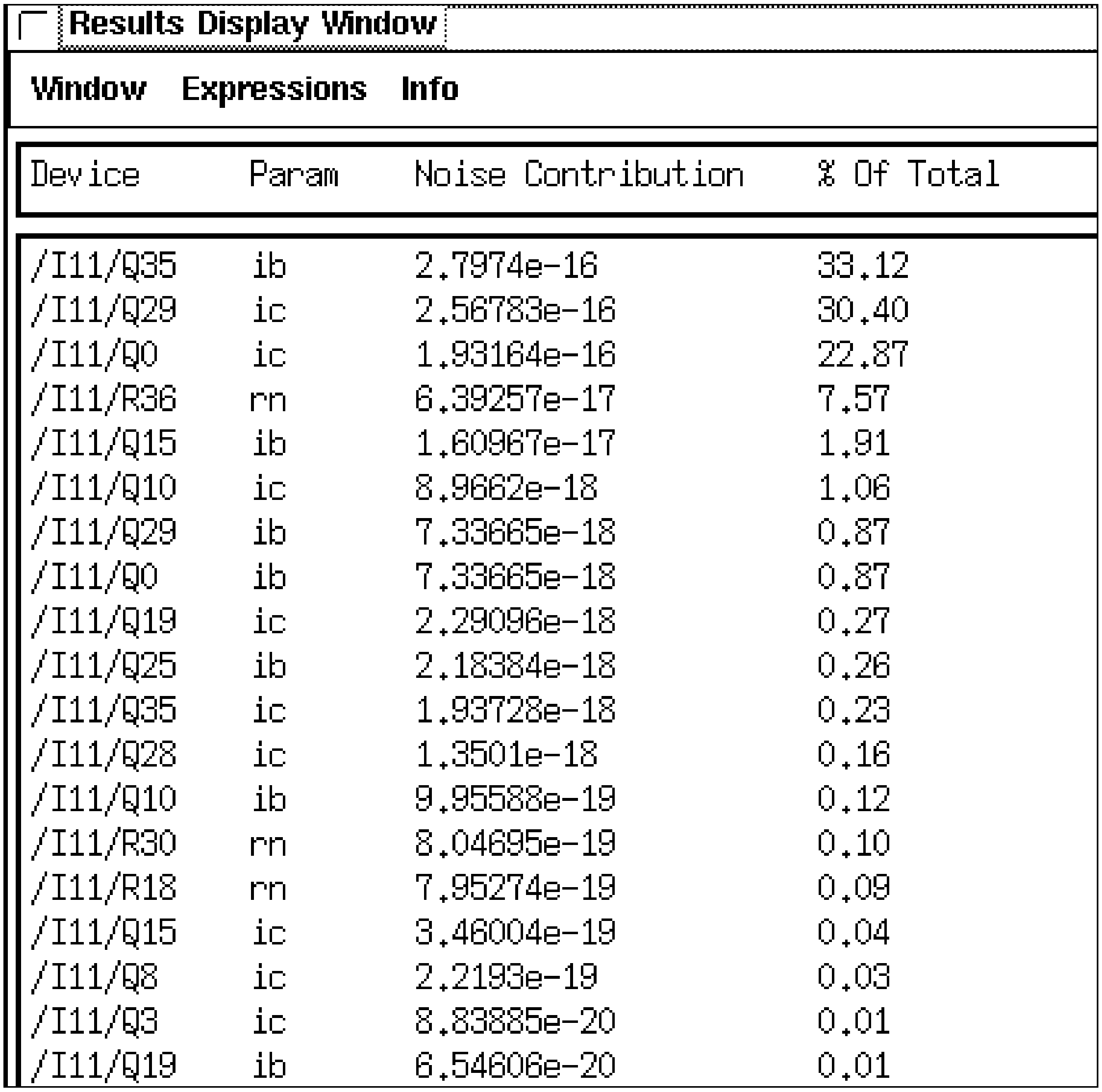
6. If needed, choose truncation details to shorten your summary.

   You can shorten your summary by specifying how many of the highest contributors to include in the summary, by specifying the percentage of noise a device must contribute to be included in the summary, or by specifying the level of noise a device must contribute to be included in the summary.

7. Choose a sorting method.

8. Click *OK*.

The Results Display window displays the noise summary using the criteria that you specified to shorten and order the list.



```
Device      Param    Noise Contribution    % Of Total

/I11/Q35    ib       2.7974e-16            33.12
/I11/Q29    ic       2.56783e-16           30.40
/I11/Q0     ic       1.93164e-16           22.87
/I11/R36    rn       6.39257e-17           7.57
/I11/Q15    ib       1.60967e-17           1.91
/I11/Q10    ic       8.9662e-18            1.06
/I11/Q29    ib       7.33665e-18           0.87
/I11/Q0     ib       7.33665e-18           0.87
/I11/Q19    ic       2.29096e-18           0.27
/I11/Q25    ib       2.18384e-18           0.26
/I11/Q35    ic       1.93728e-18           0.23
/I11/Q28    ic       1.3501e-18            0.16
/I11/Q10    ib       9.95588e-19           0.12
/I11/R30    rn       8.04695e-19           0.10
/I11/R18    rn       7.95274e-19           0.09
/I11/Q15    ic       3.46004e-19           0.04
/I11/Q8     ic       2.2193e-19            0.03
/I11/Q3     ic       8.83885e-20           0.01
/I11/Q19    ib       6.54606e-20           0.01
```

The precision of the noise data displayed in the *Results Display* window, can be controlled using the cdsenv variable `"digits"` of the tool[.partition] `"asimenv.noiseSummary"`. The default value for this variable is 6 and can be set to any other integer value using the following command on the CIW prompt:

Example

```
envSetVal("asimenv.noiseSummary" "digits" ’int 10)
```

```
This will set the value of the variable to 10.
```

The number of decimals printed for any relative contribution is controlled using the cdsenv variable *`"percentDecimals"`* of the tool[.partition] `"asimenv.noiseSummary"`. The default value for this variable is 2 and can be set to any other integer value using the following command on the CIW prompt:

Example

```
envSetVal("asimenv.noiseSummary" "percentDecimals" ’int 4)
```

```
This will set the value of the variable to 4.
```

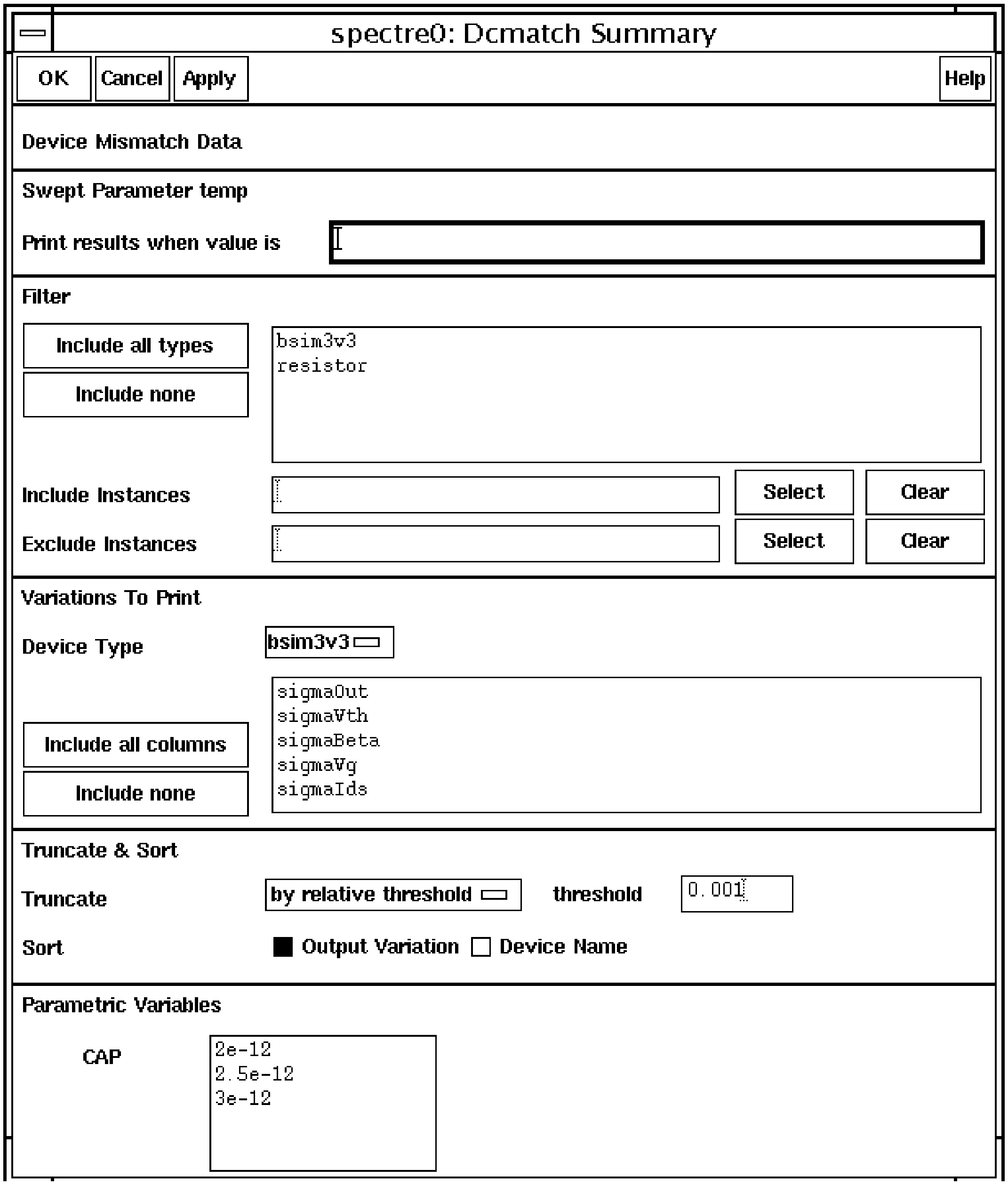
Printing DC Mismatch Summary

To print the DC Mismatch summary in your circuit,

1. Choose *Results – Print – DC Mismatch Summary.*

   Note: This menu option is enabled when ever dcmatch analysis is included in the last run or the results directory specifically selected through Artist, contains the results for dcmatch analysis.

The *DC Mismatch Summary* form appears



Device Mismatch Data

Swept Parameter temp

Print results when value is

Filter

bsim3v3
resistor

Include Instances

Exclude Instances

Variations To Print

Device Type bsim3v3

sigmaOut
sigmaVth
sigmaBeta
sigmaVg
sigmaIds

Truncate & Sort

Truncate by relative threshold threshold 0.001

Sort ■ Output Variation □ Device Name

Parametric Variables

CAP
2e-12
2.5e-12
3e-12

2. Specify a value in the *Print results when value is* field.

Note: The *Swept Parameter temp* section appears on the form only when a parameter was swept for the analysis and the swept values are numeric. The remaining swept parameters if any, appear in the list box(es) in the *Parametric Variables* section of the form, for you to select the available values.

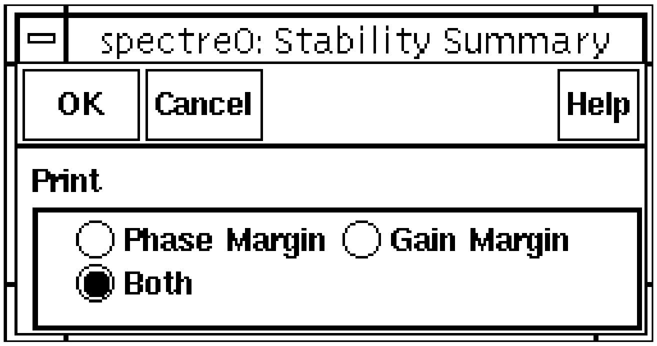
3. Specify the type of devices you need to print the results for, in the *Filter* section. The *Include all types* and *Include none* buttons can be used to include or exclude all types at a single click. You can include specific instances or exclude specific instances. You can either type the instance names or use the select buttons to pick them from schematics. The *Clear* button is used to clear the fields.
4. Specify the information to be made available for the various device types, in the *Variations to Print* section. The *Include all columns* and *Include none* buttons can be used for easier list box operation.
5. Truncate and sort data by top contributors and relative/absolute contribution. The default is relative contribution with the threshold being the value of the threshold parameter used on the analysis line. You can sort by variation or device name.
6. .The *Parametric Variables* section can display upto 6 swept variables in the results. Select the value for which the results are to be displayed.

Printing Stability Summary

To print the stability summary in your circuit,

1. Choose *Results – Print – Stability Summary.* The *Stability Summary* form appears.The form enables you to print *Phase Margin*, *Gain Margin* or *Both*.

   Note: This menu option will be enabled only when stability analysis was included in the last run or the results file exists in the results directory when you specifically selected an existing results directory through Artist.



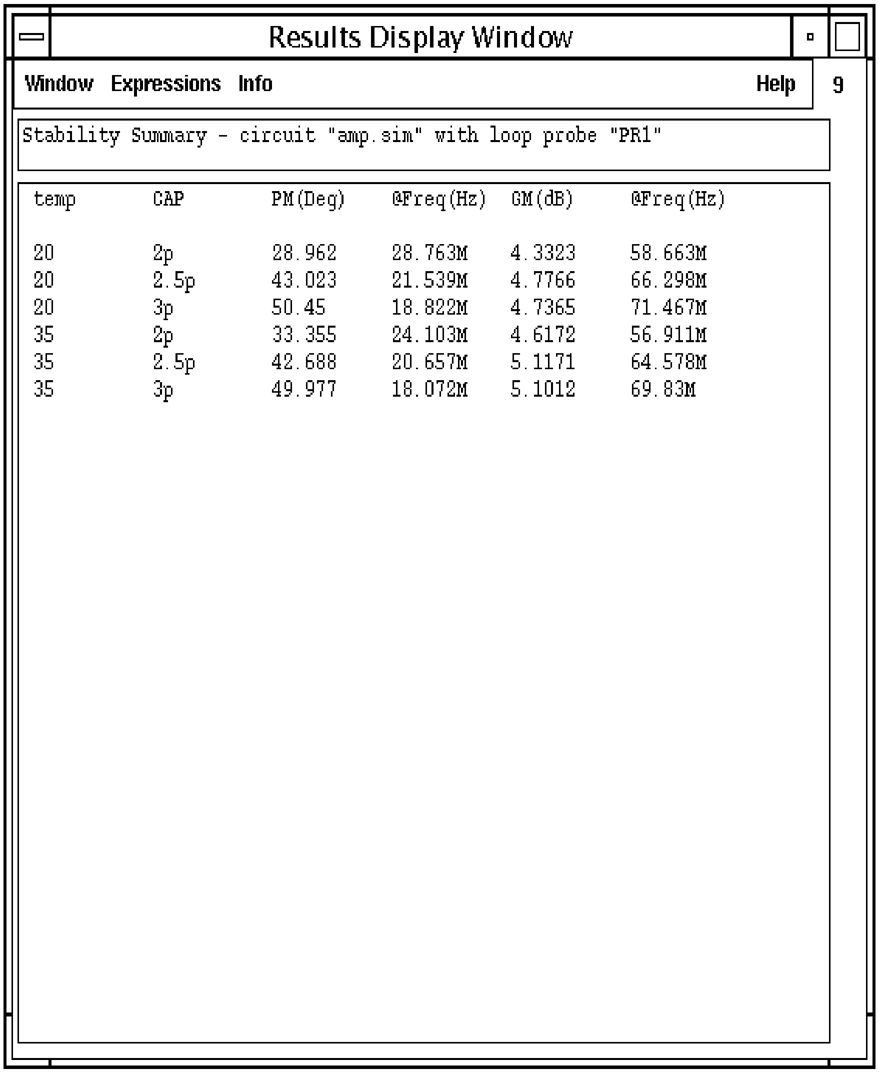
Print

○ Phase Margin ○ Gain Margin
● Both

   The form handles parametric (family) data and prints results at all available sweep points.

2. Choose the required data and click *OK*.

   The *Results Display Window* displays the stability summary using the criteria that you specified. For example, if you had swept temperature and capacitor values with the parametric tool for the stability analysis and selected the *Both* option on the form, the *Results Display Window* will appear as follows:



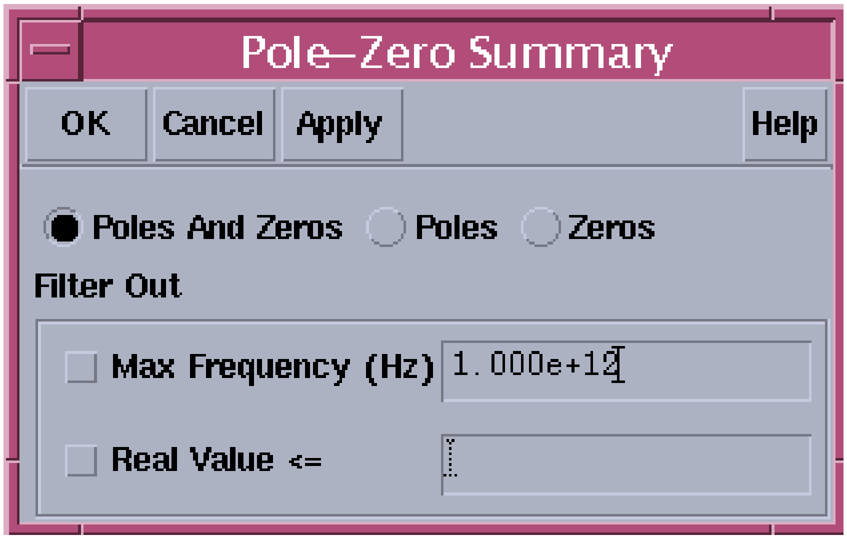
```
Stability Summary - circuit "amp.sim" with loop probe "PR1"

temp      CAP       PM(Deg)   @Freq(Hz)  GM(dB)    @Freq(Hz)

20        2p        28.962    28.763M    4.3323    58.663M
20        2.5p      43.023    21.539M    4.7766    66.298M
20        3p        50.45     18.822M    4.7365    71.467M
35        2p        33.355    24.103M    4.6172    56.911M
35        2.5p      42.688    20.657M    5.1171    64.578M
35        3p        49.977    18.072M    5.1012    69.83M
```

Printing Pole Zero Summary

To print the *Pole Zero* summary in your circuit,

1. Choose *Results – Print – Pole Zero Summary.* The *Pole-Zero Summary* form appears. The form enables you to print poles or zeros, or poles and zeros with filtering options.

   Note: This menu option will be enabled only when pole zero analysis was included in the last run or the results file exists in the results directory when you specifically selected an existing results directory through the Analog Design Environment.



● Poles And Zeros ○ Poles ○ Zeros

Filter Out

□ Max Frequency (Hz) 1.000e+12

□ Real Value <=

2. Select the option, *Poles* if you want to plot only poles, *Zeros* if you want to plot only zeros and *Poles and Zeros* if you want to plot both poles and zeros.

   Note: By default, the option *Poles and Zeros* is selected.

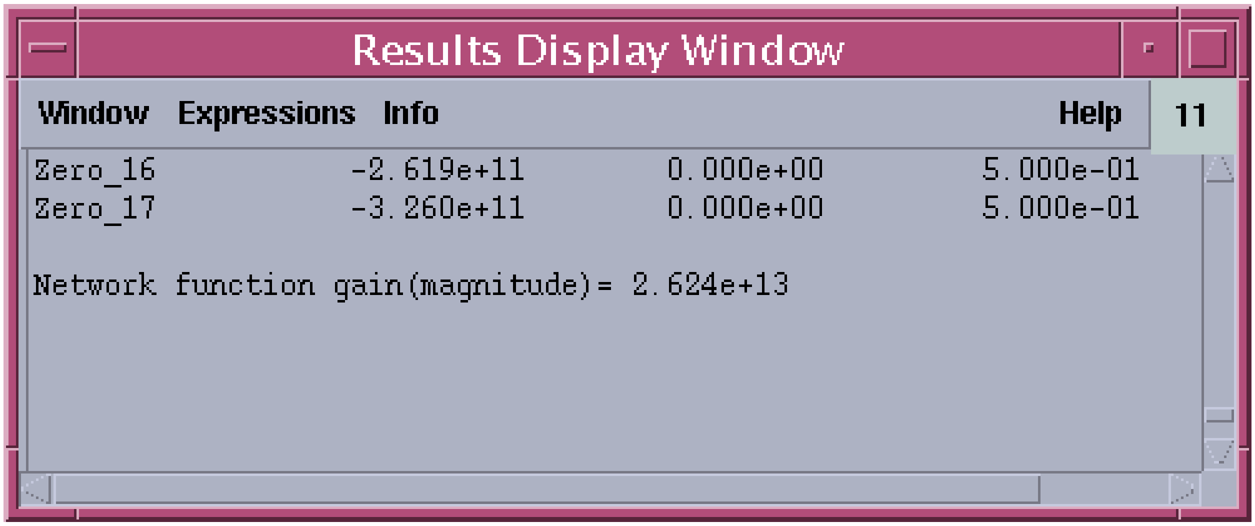
3. Set the required options in the *Filter Out* section and click *OK*. This section provides a combination of filtering mechanisms that you can select in order to plot the poles and zeros. These are:

   ❑ *Max Frequency:*
   This option enables you to filter out poles and zeros that are outside the frequency band of interest (FBOI) and that do not influence the transfer function in the FBOI. The default value is that specified in the *fmax* field in the *Pole-Zero Options* form. Note, that for the *Direct Plot* form, *fmax* is read from the header of the `psf` data. Only poles and zeros whose magnitudes exceed the frequency value specified are filtered out.

   ❑ *Real Value:*
   This option enables you to specify the real part of the frequency. Only poles and zeros whose real values are less than or equal to the real value specified are filtered out.

   Note: By default, no filtering is selected. You can set the filtering criteria once you specify either poles or zeros or both to be plotted.

4. The *Results Display Window* displays the pole zero summary using the criteria that you specified:



```
Zero_16          -2.619e+11       0.000e+00        5.000e-01
Zero_17          -3.260e+11       0.000e+00        5.000e-01

Network function gain(magnitude)= 2.624e+13
```

Printing DC Node Voltages

To print the DC node voltages of the nodes or components in your circuit,

1. Choose *Results – Print – DC Node Voltages*.
2. Move your cursor into the Schematic window.

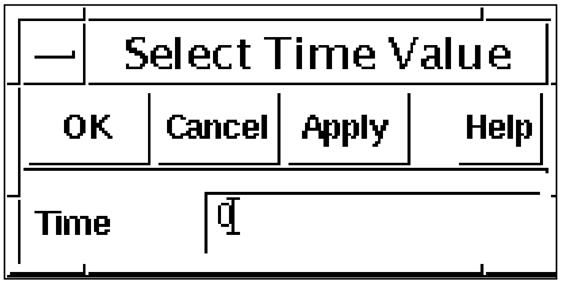
   You are prompted to select nets for the VDC output.
3. Click a node.

Printing Transient Voltages

To print the transient node voltages of the nodes in your circuit,

1. Choose *Results – Print – Transient Node Voltages*.

The Select Time Value form appears.



Time 0

If the menu does not appear automatically, press F3.

2. In the *Time* field, type the time value at which you want the transient node voltages to print.

   The default time value is 0.

3. Move your cursor into the Schematic window.

   The CIW prompts you to select nets for the time value output.

4. Click a node (net8, for example).

Printing Sensitivities

1. Choose *Results – Print – Sensitivities*.
2. Move your cursor into the Schematic window.

   You are prompted to select nets for the output.

3. Click a net or port.

Precision Control for Printing

Precision of printed results can be controlled using aelPushSignifDigits.

Example

```
aelPushSignifDigits(4)
rn             37.9322e-18    fn             0              total
37.9322e-18
```

```
aelPushSignifDigits(8)
rn             37.932238e-18  fn             0              total
37.932238e-18
```

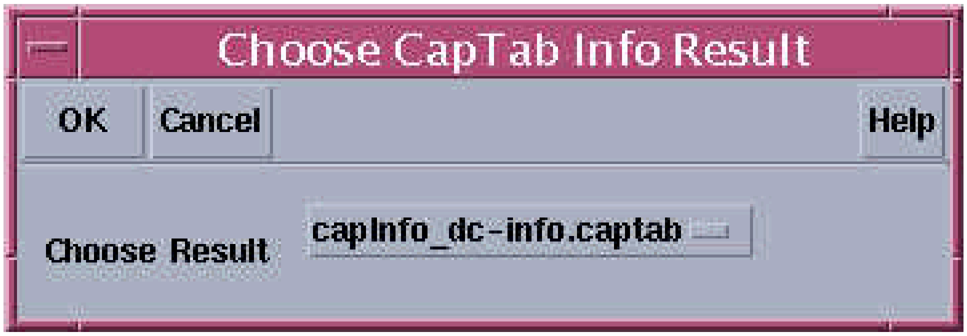
Printing Capacitance Data

When you run a transient or dc analysis with the *captab* option selected in the analysis options form, you can view the captab data in the capacitance table. For more information on the *captab* option and parameters, see "CAPTAB Parameters" on page 197.

To view capacitance data,

1. Choose *Results – Print – Capacitance Table* in the ADE window.

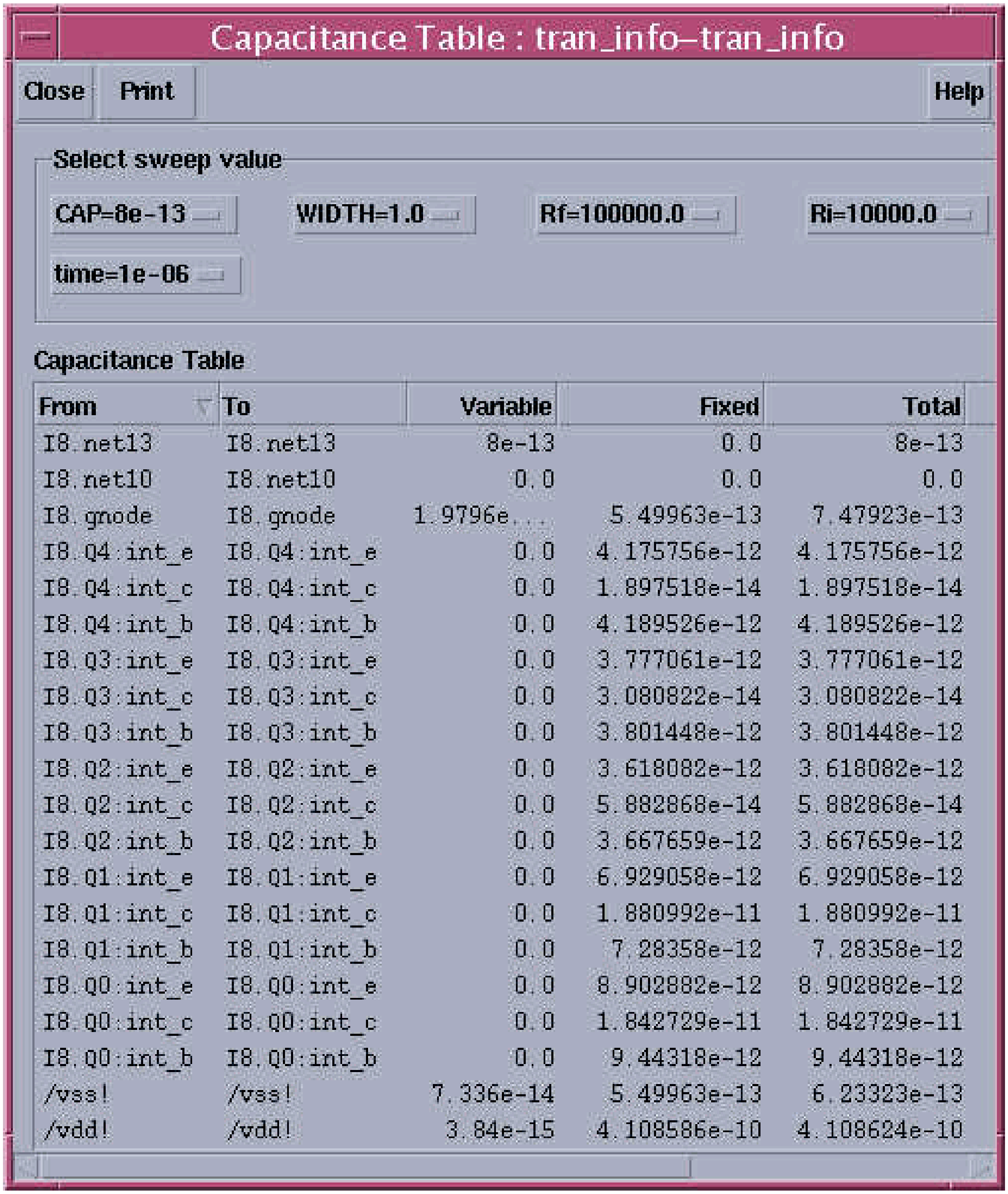
   If you had opted to save captab data for more than one analysis, the Choose CapTab Info Result form appears first.


Choose CapTab Info Result

Choose Result: capInfo_dc-info.captab

   Select one captab result file from the *Choose Result* cyclic box and click *OK*.

The Capacitance Table appears as shown below:



Select sweep value

CAP=8e-13 WIDTH=1.0 Rf=100000.0 Ri=10000.0
time=1e-06

Capacitance Table

From	To	Variable	Fixed	Total
I8.net13	I8.net13	8e-13	0.0	8e-13
I8.net10	I8.net10	0.0	0.0	0.0
I8.gnode	I8.gnode	1.9796e...	5.49963e-13	7.47923e-13
I8.Q4:int_e	I8.Q4:int_e	0.0	4.175756e-12	4.175756e-12
I8.Q4:int_c	I8.Q4:int_c	0.0	1.897518e-14	1.897518e-14
I8.Q4:int_b	I8.Q4:int_b	0.0	4.189526e-12	4.189526e-12
I8.Q3:int_e	I8.Q3:int_e	0.0	3.777061e-12	3.777061e-12
I8.Q3:int_c	I8.Q3:int_c	0.0	3.080822e-14	3.080822e-14
I8.Q3:int_b	I8.Q3:int_b	0.0	3.801448e-12	3.801448e-12
I8.Q2:int_e	I8.Q2:int_e	0.0	3.618082e-12	3.618082e-12
I8.Q2:int_c	I8.Q2:int_c	0.0	5.882868e-14	5.882868e-14
I8.Q2:int_b	I8.Q2:int_b	0.0	3.667659e-12	3.667659e-12
I8.Q1:int_e	I8.Q1:int_e	0.0	6.929058e-12	6.929058e-12
I8.Q1:int_c	I8.Q1:int_c	0.0	1.880992e-11	1.880992e-11
I8.Q1:int_b	I8.Q1:int_b	0.0	7.28358e-12	7.28358e-12
I8.Q0:int_e	I8.Q0:int_e	0.0	8.902882e-12	8.902882e-12
I8.Q0:int_c	I8.Q0:int_c	0.0	1.842729e-11	1.842729e-11
I8.Q0:int_b	I8.Q0:int_b	0.0	9.44318e-12	9.44318e-12
/vss!	/vss!	7.336e-14	5.49963e-13	6.23323e-13
/vdd!	/vdd!	3.84e-15	4.108586e-10	4.108624e-10

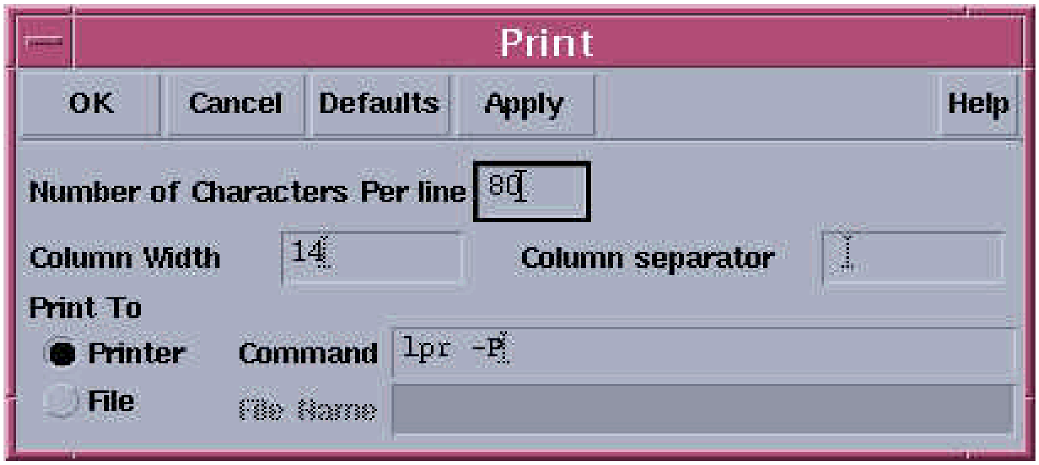
The table shows captab data in five columns:

- *From*: The nodes from which capacitance is measured.

- *To*: The nodes to which capacitance is measured.
- *Variable*: Variable capacitance between nodes.
- *Fixed*: Fixed capacitance between nodes.
- *Total*: Total capacitance between nodes.

2. Select one of more sweep values for which the capacitance table is to be printed by using the cyclic buttons in the *Select sweep value* group box.
3. Click the *Print* button to save or print captab data.

   The Print form appears.



Number of Characters Per line 80

Column Width 14 Column separator

Print To

● Printer Command lpr -P

○ File File Name

4. Specify values in this form as follows:
   1. *Number of characters per Line:* Number of characters to be printed on a line.
   2. *Column Width*: Width of the columns to be printed. The default value is 14.
   3. *Column Separator*: Column separator to be used for printing the table. The default is a space.
   4. *Print To*: If you want to print to a printer, select the *Printer* option button and specify a command. If you want to print to a file, select the *File* option button and specify a filename.
5. Click *OK* or *Apply*.

Printing Statistical Reports or Calculator Results

For information about statistical reports, read the Virtuoso® *Advanced Analysis Tools User Guide*.

For information about printing data using *WaveScan Calculator*, read the *WaveScan User Guide*. For information about AWD Calculator's *print* and *printvs* commands, read the *Waveform Calculator User Guide*.

Using SKILL to Display Tabular Data

You can use the SKILL language for queries to request other kinds of simulation results, to build output format macros, and to automate test and result reporting sequences. The syntax for queries is shown at the beginning of the line in the Results Display window.

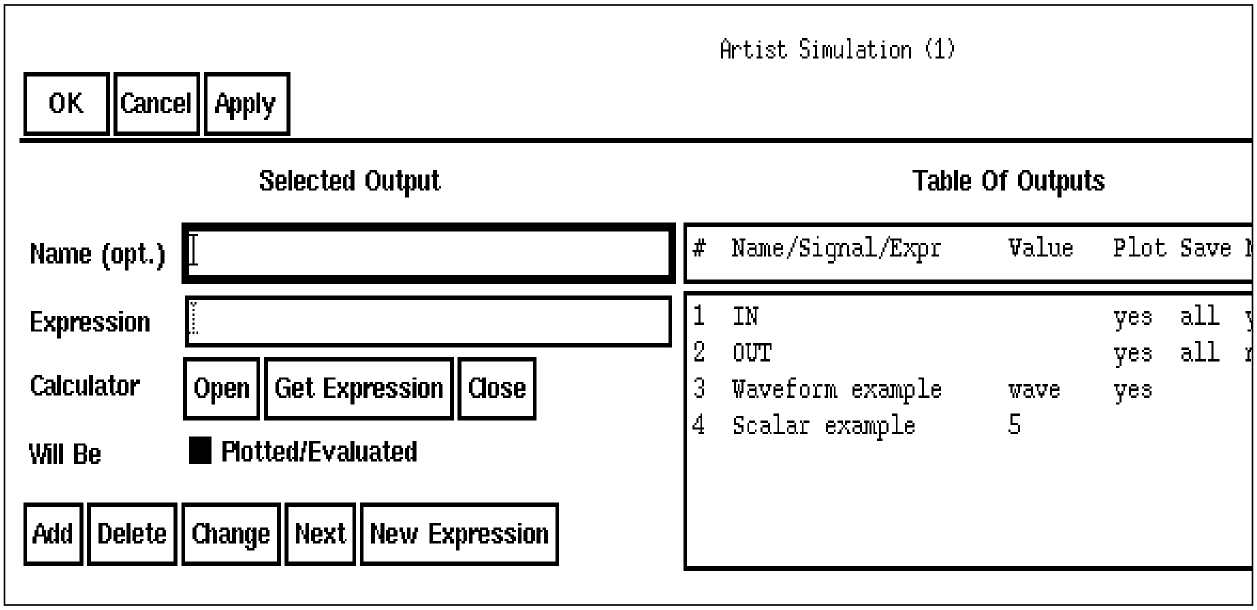
To display…	**Type this command in the CIW**
A list of operating-point parameter names and their values for `R1`	`OP("/R1","??")`
A list of just the operating-point parameter names for `R1`	`OP("/R1","?")`
A single operating-point parameter (`v` for voltage, for example) and its value for `R1`	`OP("/R1","v")`
A list of transient operating-point parameter names and their values for `C1`	`OPT("/C1","??")`
A list of just the transient operating-point parameter names for `C1`	`OPT("/C1","?")`
A single transient operating-point parameter (`i` for current, for example) and its value for `C1`	`OPT("/C1","i")`
A list of model parameter names and their values for `Q1`	`MP("/Q1","??")`
A list of just the model parameter names for `Q1`	`MP("/Q1","?")`
A single model parameter (`is` for saturation current, for example) and its value for `Q1`	`MP("/Q1","is")`
Noise parameter information for a device with only one noise parameter (a resistor `R4`, for example)	`VNP("/R4")`

To display…	Type this command in the CIW
A list of noise parameter names for a device with more than one noise parameter (a device `D24`, for example) and their values	`VNPP("/D24","??")`
A list of just the noise parameter names for a device with more than one noise parameter (a device `D24`, for example)	`VNPP("/D24","?")`
A single noise parameter (`rs` for saturation resistance, for example) and its value for a device with more than one noise parameter (a device `D24`, for example)	`VNPP("/D24","rs")`

Overview of Plotting Calculator Expressions

You can plot calculator expressions that are waveforms and print scalar expressions.

Based on the Simulation window segment in the following illustration, output 3 is a waveform expression that can be plotted. Output 4 is a scalar expression with a result value of 5. The scalar expression value appears in the Simulation window and the Setting Outputs form but is not plotted, saved, or marched.



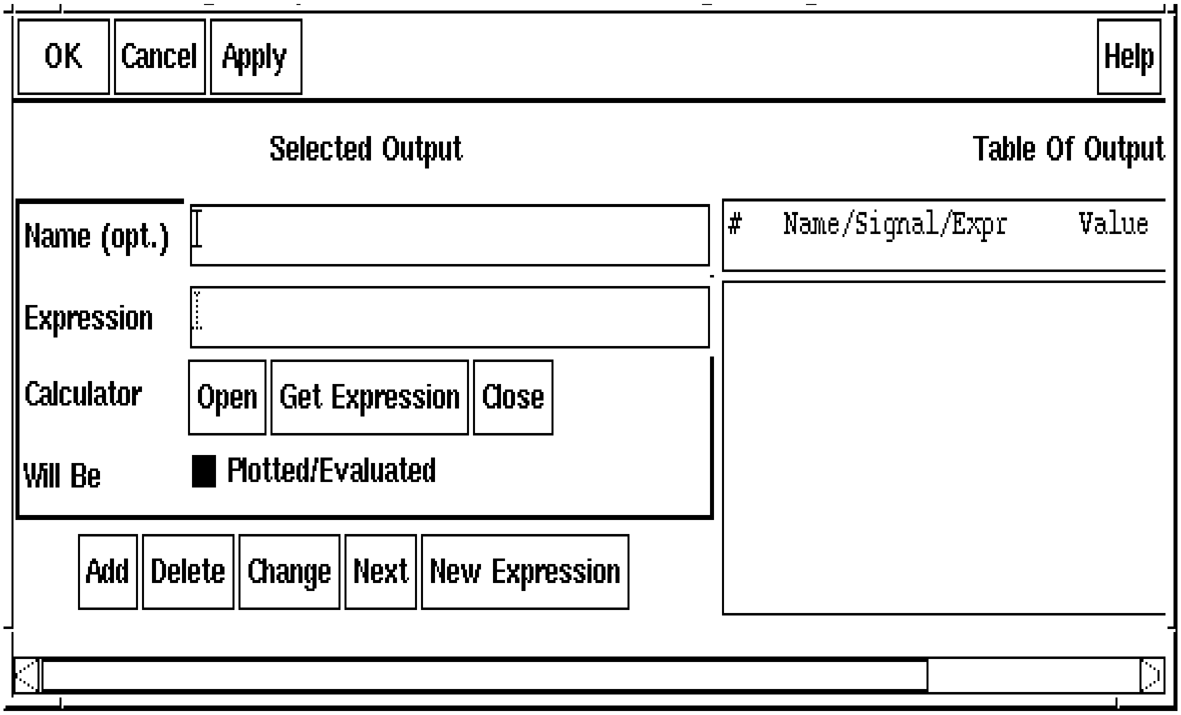
#	Name/Signal/Expr	Value	Plot	Save
1	IN		yes	all
2	OUT		yes	all
3	Waveform example	wave	yes	
4	Scalar example	5		

Defining Expressions

To define a new expression,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. In the Setting Outputs form, click *New Expression*.

   The Setting Outputs form redraws to display a blank *Expression* field.



   For details about the form, see “Setting Outputs” on page 495.

3. (Optional) Type a name for the expression.

   If you do not type a name, the expression itself appears in the *Table Of Outputs* list box and in the Waveform window.

   For example, the expression for the 3 dB point is

   ```
   bandwidth(VF("/OUT), 3, "low")
   ```

   Rather than seeing this expression in the *Table Of Outputs* list box of the Simulation window, you might type the name

   ```
   3 dB point of OUT
   ```

4. Enter the expression using one of the following methods:

   ❑ Type the expression in the *Expression* field.

   ❑ Use the calculator to create the expression and then retrieve it by clicking *Get Expression*.

5. Click *Add*.

   The expression is added to the *Table Of Outputs* list box.

Plotting Expressions

To plot expressions,

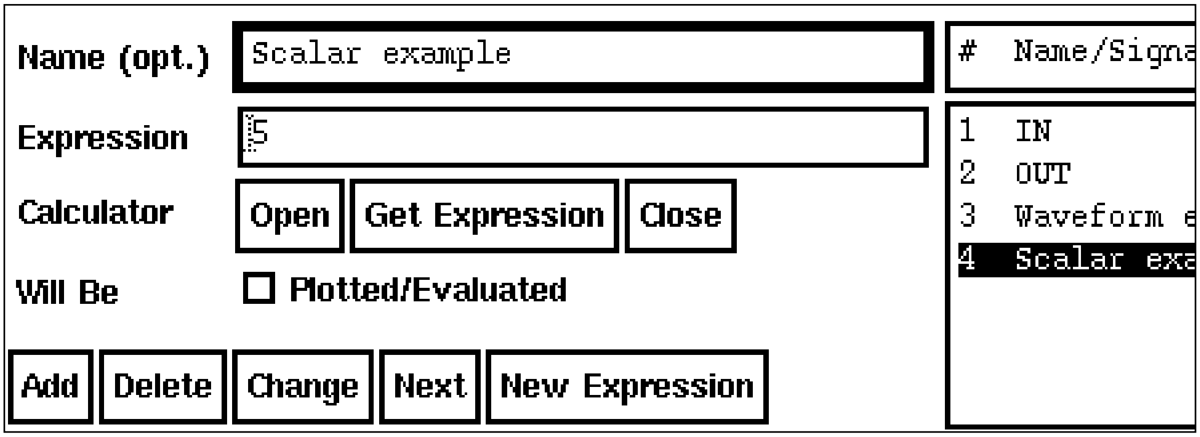
➤ Choose *Results – Plot Outputs – Expressions*.

The system plots waveform expressions in the Waveform window and prints the value of scalar expressions in the *Outputs* area of the Simulation window.

Suppressing Plotting of an Expression

To suppress plotting of an expression without deleting it,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. Click the expression in the *Table Of Outputs* list box.
3. Deselect *Will Be Plotted/Evaluated*, as shown in the figure.



Name (opt.) Scalar example

Expression 5

Will Be □ Plotted/Evaluated

```
#  Name/Signa
1  IN
2  OUT
3  Waveform e
4  Scalar exa
```

4. Click *Change*.

Now when you choose *Results – Plot Outputs – Expressions*, the expression you deactivated is not plotted.

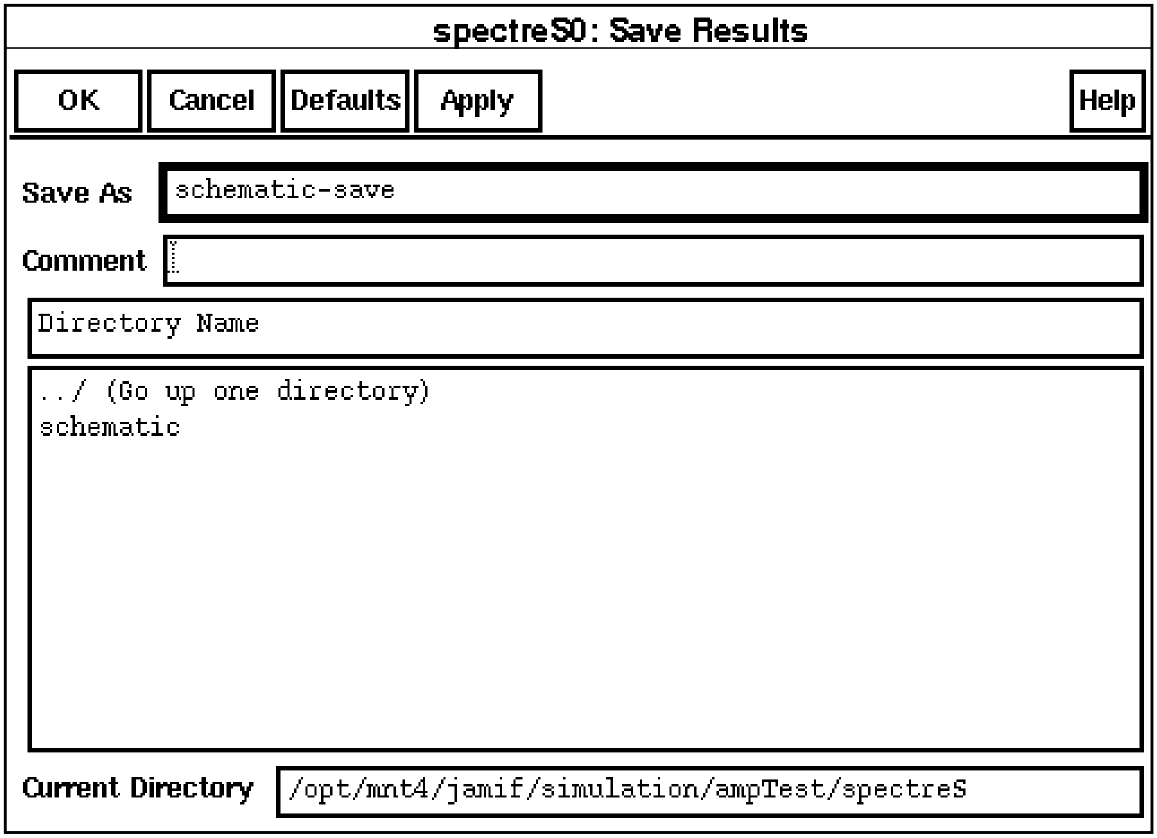
Annotating Simulation Results

Saving Simulation Results

The default name under which results are saved is `schematic-save`. To save a results directory under a different name,

1. Choose *Results – Save* in the Simulation window or the Schematic window.

The Save Results form shows the results of the latest simulation in the *Save As* field.



Save As: schematic-save

Comment:

Directory Name

```
../ (Go up one directory)
schematic
```

Current Directory: /opt/mnt4/jamif/simulation/ampTest/spectreS

For detailed information about the form, see “Save Results” on page 498.

2. Type a new name for the results directory in the *Save As* field.

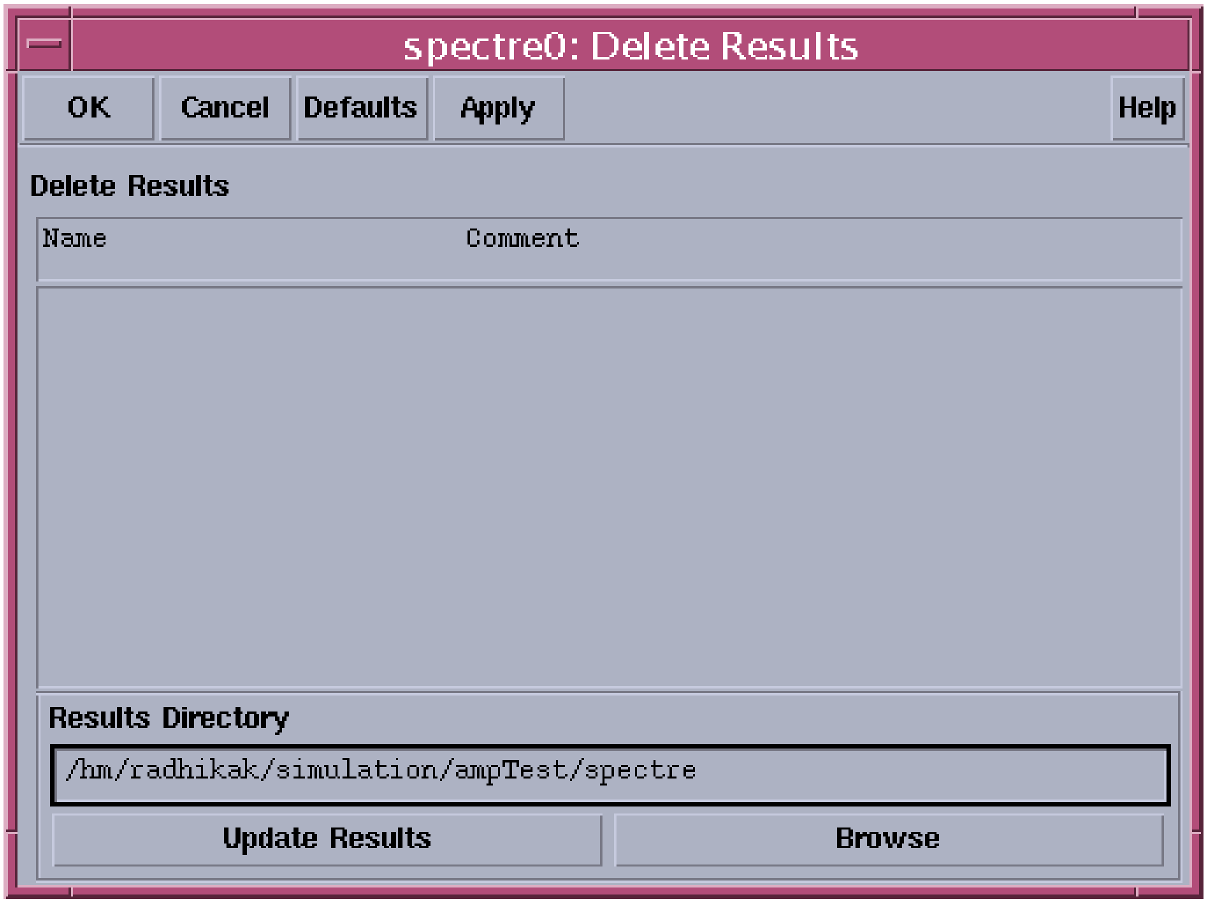
3. (Optional) Type a short description in the *Comment* field to help you identify these results when you restore the results later.

Deleting Simulation Results

To delete a set of simulation results,

1. Choose *Results – Delete* in the Simulation window or the Schematic window.

   The Delete Results form appears.



/hm/radhikak/simulation/ampTest/spectre

   The *Results Directory* field lists the default directory in which results are saved.

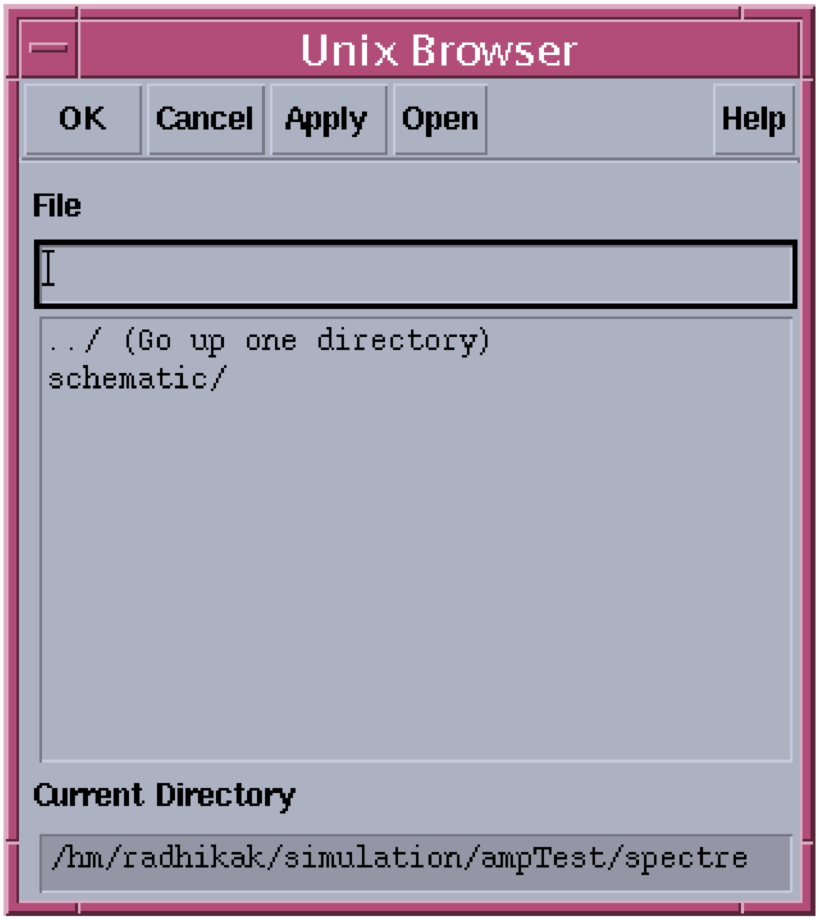
2. Choose the results you want to delete from the list box.

   If the results you want to delete are in a different location, click the *Browse* button to open the Unix Browser form.

Browsing Results Directories

To browse directories,

➤ Click the *Browse* button.



```
../ (Go up one directory)
schematic/
```

/hm/radhikak/simulation/ampTest/spectre

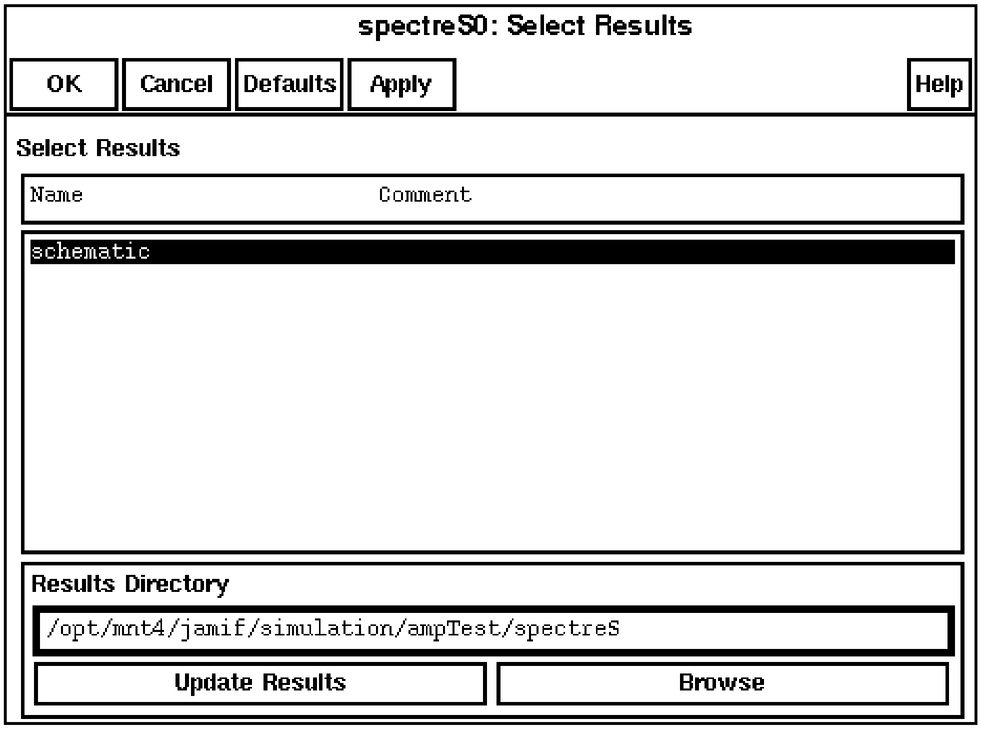
For detailed information about the form, see “UNIX Browser” on page 501.

Restoring Saved Results

To select and restore a set of simulation results for the current design,

1. Choose *Results – Select* in the Simulation window or the Schematic window.

The Select Results form appears.



schematic

/opt/mnt4/jamif/simulation/ampTest/spectreS

The *Results Directory* is the directory in which the simulation results for the selected simulation are saved.

2. Check that the *Results Directory* field displays the correct information.

   You can select results in a different location by clicking the *Browse* button and navigating to the proper directory.

   Note: The proper directory is two levels up from the `psf` directory. For example, if your results directory is: `simulation/ampTest/spectre/schematic/psf`, use the browser to select `simulation/ampTest/spectre`.

3. Double-click the results you want.

4. Click *OK*.

Note: If you restore parasitic simulation results, be sure that parasitic simulation is enabled from the LVS form.

You can annotate the schematic to show parameters, operating points, net names, and voltages of individual design components.

Before you can annotate the schematic, do one of the following:

- ❑ Run a simulation.
- ❑ Select results.

  To select results, choose *Results – Select* in the Simulation window, select the current data file, and click *OK*.

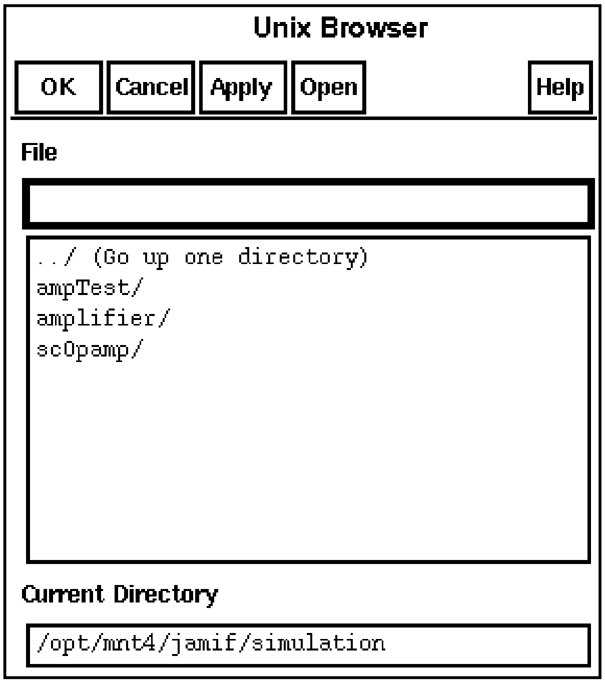
To annotate the schematic,

➤ In the Simulation window or the Schematic window, choose *Results* and one of the annotate commands.

These commands temporarily change the cell and instance label display settings for all instances in the current cellview, for all the other cellviews in the current library, and for the reference libraries of the current cellview.

To annotate instances selectively, use the *Edit – Component Display* command in the Schematic window.

To browse directories, click the *Browse* button.



```
../ (Go up one directory)
ampTest/
amplifier/
scOpamp/
```

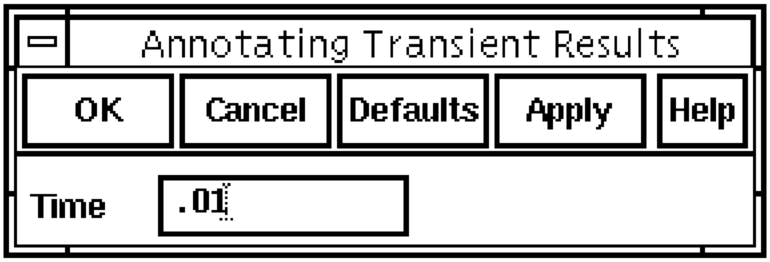
```
/opt/mnt4/jamif/simulation
```

Annotating Transient Voltages

To annotate transient voltages,

1. In the Simulation window or the Schematic window, choose *Results – Annotate – Transient Node Voltages*.

   The Annotating Transient Results form appears.



Time .01

2. Type the transient time point in the *Time* field, and click *OK*.

Annotating Transient Operating Points

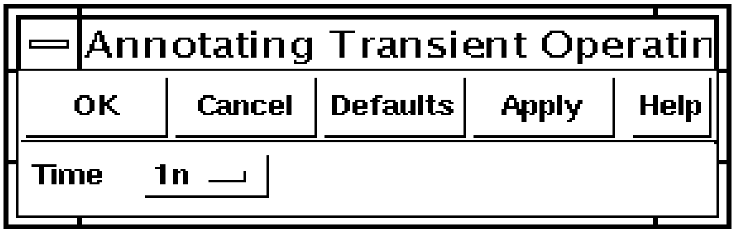
To annotate final transient operating points,

1. In the Simulation window or the Schematic window, choose *Results – Annotate – Transient Operating Points*. This will annotate the operating point data for the final timepoints.

To annotate infotimes transient operating points,

1. In the Simulation window or the Schematic window, choose *Results – Annotate – Transient Operating Points*.

   The *Annotating Transient Operating Points* form appears.



Time 1n

2. Select the transient time point in the *Time* drop-down field. This field lists the choices of timepoints at which the operating point data is stored.This will annotate the operating point data for the selected timepoint saved.
3. Click *OK* or *Apply*. These two buttons essentially perform the same operation except that the Apply button does not close the form enabling the user to select another timepoint and click *Apply* again to annotate data for a different timepoint. Clicking on the *Cancel* button will cancel entire operation.

Note: This form will not come up if the user has not stored operating point data at different timepoints.

Note: *The Results – Annotate – Show Parasitics* and *Results – Annotate – Hide Parasitics* are enabled only when the results are available for *dcop*.

Specifying the Data Directory for Labels

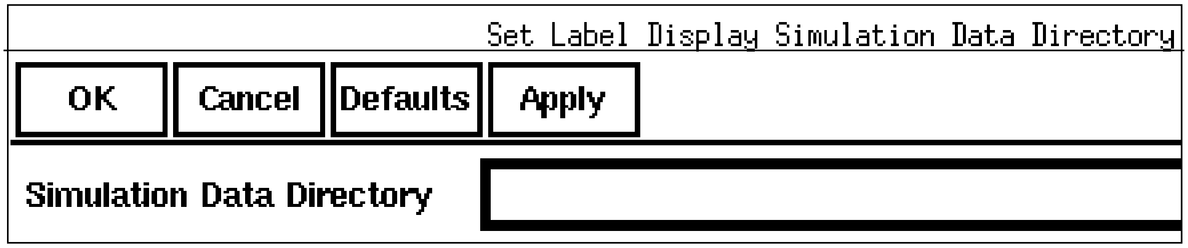
To specify the simulation data directory (run directory) for labels,

1. In the Schematic window, choose *Edit - Component Display*

   The Edit Component Display Options form appears.

2. Click *Set Simulation Data Directory*.

   The Set Label Display Simulation Data Directory form appears.



Simulation Data Directory

3. Type the path to the simulation run directory and click *OK*.

Note: You do not need to use this form if

- ❑ You have the analog circuit design environment active and specified the correct directory as the run directory
- ❑ You used the Results Browser to select results for the current schematic
- ❑ The most recent simulation you ran was of this schematic

Saving and Removing Annotated Labels

To save the label display changes you made to the current cellview,

➤ Save the schematic.

To remove your labels and restore the default label display specifications to all affected cellviews and libraries,

➤ Choose *Results – Annotate – Restore Defaults* from the Simulation window or the Schematic window.

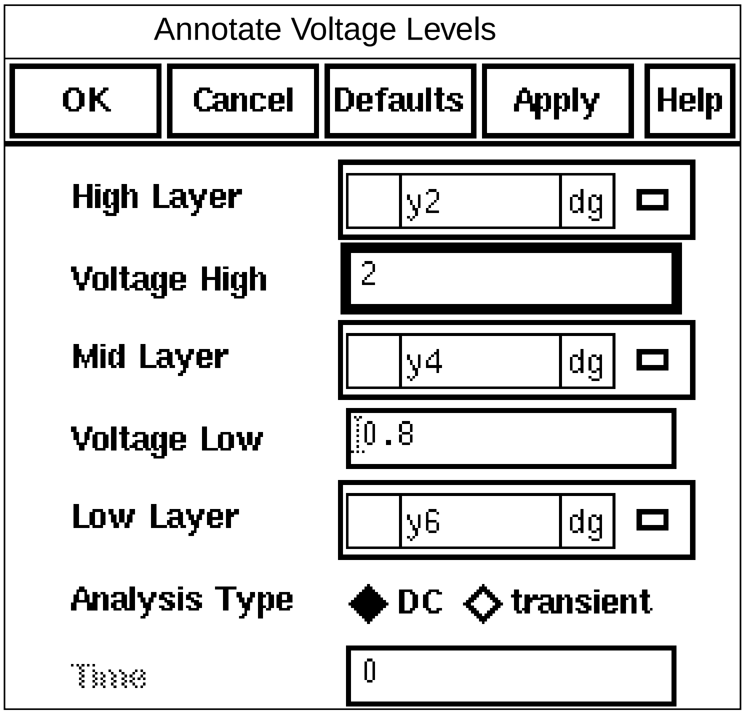
Highlighting Logic Levels with Wire Colors

To set wire colors based on circuit voltage levels and partition the circuit into high, mid, and low logic levels,

1. Select *Results – Annotate – Voltage Levels* from the Simulation window or the Schematic window.

   Note: When you display this form, the system clears existing probes from the schematic. This includes simulation output and marching output probes.

   The Annotate Voltage Levels form appears.



High Layer	y2 dg
Voltage High	2
Mid Layer	y4 dg
Voltage Low	0.8
Low Layer	y6 dg
Analysis Type	◆ DC ◇ transient
Time	0

   For detailed information about the form, see “Annotate Voltage Levels” on page 494.

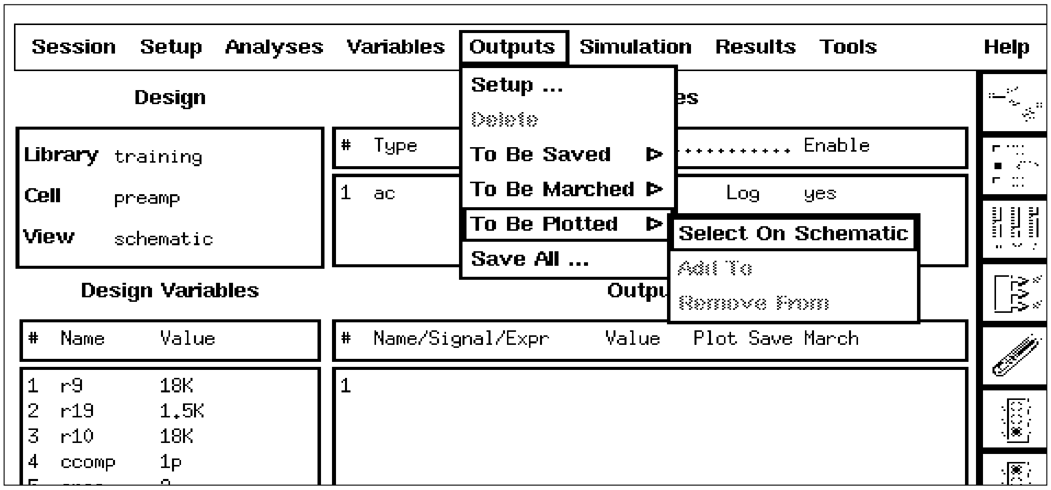
2. Choose layers and colors for the three regions.
3. Set the voltage levels to partition the circuit.

Plotting Results of a Parametric Analysis

This section presumes that you already ran a parametric analysis with an AC analysis selected. It shows you how to plot parametric analysis results for this AC analysis in the Waveform window. For more information about the Waveform window, see the documentation for the Waveform window, the waveform calculator, and the Browser.

To display the results of a parametric analysis,

1. Choose *Outputs – To Be Plotted – Select On Schematic* in the Simulation window.



Library training
Cell preamp
View schematic

#	Type	Enable
1	ac	Log yes

#	Name	Value
1	r9	18K
2	r19	1.5K
3	r10	18K
4	ccomp	1p

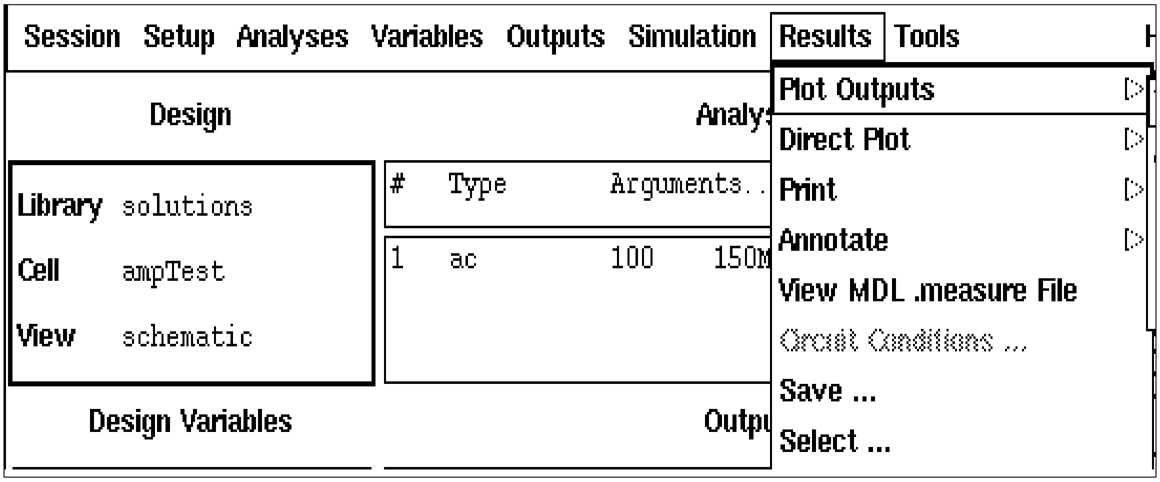
#	Name/Signal/Expr	Value	Plot	Save	March
1					

2. In the schematic, choose the outputs you want to display by clicking on them, or hold down the left mouse button and drag a box over the objects you want to choose.

   Outputs you select appear in the *Outputs* list in the Simulation window.

   For more information about selecting values in the schematic, see "Overview of Plotting" on page 414.

3. Choose *Results – Plot Outputs* and the name of an analysis in the Simulation window. In this example, the analysis is AC.

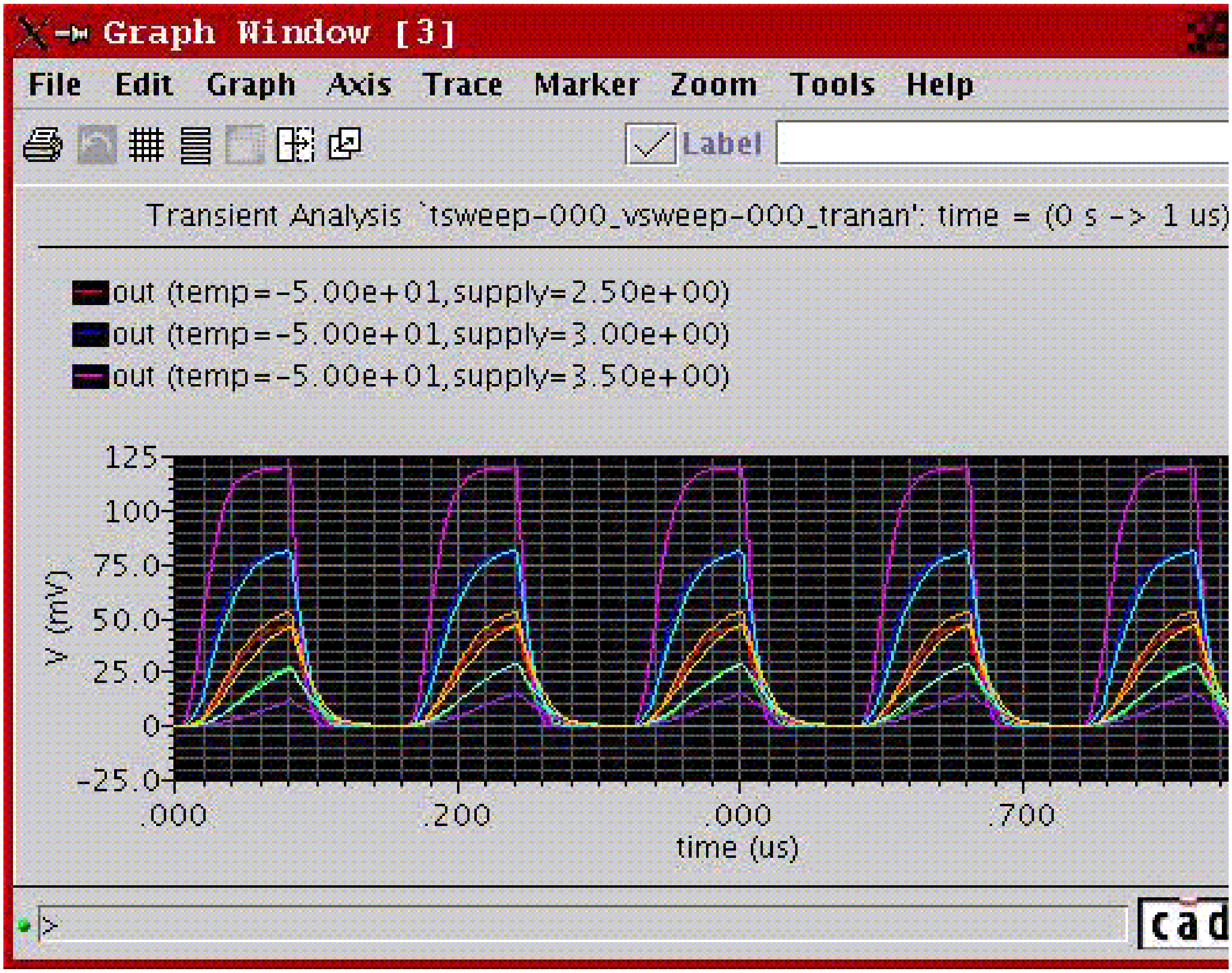


Library solutions
Cell ampTest
View schematic

#	Type	Arguments..
1	ac	100 150M

Note: You can choose other plot outputs to plot different types of analysis results.

The Waveform window appears (if it is not already open) and displays the waveforms created by parametric analysis.


Graph Window [3]
File
Edit
Graph
Axis
Trace
Marker
Zoom
Tools
Help
Label
Transient Analysis `tsweep-000_vsweep-000_tranan': time = (0 s -> 1 us)
out (temp=-5.00e+01,supply=2.50e+00)
out (temp=-5.00e+01,supply=3.00e+00)
out (temp=-5.00e+01,supply=3.50e+00)
125
100
75.0
50.0
25.0
0
-25.0
V (mV)
.000
.200
.000
.700
time (us)
>

You can identify the sweep variable value associated with any curve by selecting the curve. The associated curve name gets highlighted on the waveform window

Form Field Descriptions

Setting Plotting Options

Auto Plot Outputs After Simulation plots the entire plot set (including waveform expressions) automatically when each simulation is finished. When off, this option waits for you to use a *Results – Plot Outputs* command to plot the plot set.

Direct Plots Done After refers to the commands located in the *Direct Plot* menu.

Each Selection specifies that the plot is drawn after each node is selected.

All Selections Are Made specifies that none of the plots are drawn until all of the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are printed at the same time (into a table).

For the calculator `print` and `printvs` functions, you can use append mode and have more than one expression in the buffer and use `print` or `printvs` to print into a table.

Annotations selects information to be displayed in the Waveform window.

Design Name displays the design name in the Waveform window.

Simulation Date displays the simulation run date in the Waveform window.

Temperature displays the temperature associated with the plotted results in the Waveform window.

Design Variables displays the names and values of user-created variables in the Waveform window.

Scalar Outputs displays simulation results that evaluate to scalar values in the Waveform window.

Waveform Window

Allow Icons puts icons in the Waveform window.

Font Size is the default size of Waveform window text.

Width is the width of the window.

Height is the height of the window.

X Location and **Y Location** set the window position.

XF Results

Plotting Mode

Append adds the new plot to existing plots that are already displayed in the waveform window.

Replace replaces existing plots with the new plot.

New SubWin adds the plot to a new subwindow.

New Win adds the plot to a new window.

Function

Voltage Gain is a calculation of voltage over voltage.

Transimpedance is a calculation of voltage over current.

Current Gain is a calculation of current over current.

Transconductance is a calculation of current over voltage.

Modifier

Magnitude (the default setting) plots the magnitude of the selected signal.

Phase plots the phase of the selected signal.

dB20 plots the magnitude in dB20.

Real plots the real component of the signal.

Imaginary plots the imaginary component of the signal.

Replot triggers the plotting of the selected instance or instance terminal with modified specifications.

Add To Outputs followed by **Replot** adds the output to the *Table Of Outputs* list box in the Simulation window.

Select instance on schematic or **Select instance terminal on schematic** prompts you to select the appropriate instance or terminal from the schematic.

S-Parameter Results

Plotting Mode

Append adds the new plot to existing plots that are already displayed in the waveform window.

Replace replaces existing plots with the new plot.

New SubWin adds the plot to a new subwindow.

New Win adds the plot to a new window.

Function specifies the S-parameter or noise-parameter function to plot.

SP is S-parameters.

ZP is Z-parameters.

YP is Y-parameters.

HP is H-parameters.

GD is group delay.

VSWR is voltage standing wave ratio.

NFmin is minimum noise figure.

Gmin is the source reflection coefficient corresponding to NFmin.

Rn is equivalent noise resistance.

NF is noise figure.

B1f is the intermediate term for Kf, the Rollet stability factor.

Kf is the Rollet stability factor.

GT is transducer gain.

GA is available gain.

GP is power gain.

NC is noise circles.

GAC is available gain circles.

GPC is power gain circles.

LSB is load stability circles.

SSB is source stability circles.

Plot Type specifies the plot format. Option availability is a function of the selected function.

Auto uses the format in the current Waveform window unless that format is unsuitable for the function.

Rectangular specifies curves plotted against frequency.

Z-Smith specifies curves plotted on a Smith chart with impedance overlay.

Y-Smith specifies curves plotted on a Smith chart with admittance overlay.

Polar specifies curves plotted in polar (mag/angle) coordinates.

Modifier, which is used only for rectangular plots, specifies the modifier the analog circuit design environment uses to reduce complex data for two-dimensional presentation. Option availability depends on the selected function; some functions, such as stability factor, do not require a modifier.

Magnitude plots the magnitude of complex or scalar quantities.

Phase plots the phase of complex quantities in degrees.

dB20 plots the magnitude in dB.

Real plots the real part of complex quantities.

Imaginary plots the imaginary part of complex quantities.

Sweep selects a set of circles to be plotted against frequency or dB. (Sweep appears on the form only when you are plotting circles and have selected the NC, GAC, or GPC function.)

You can plot noise and gain circles at a single dB value for a range of frequencies or at a single frequency for a range of dB values.

When plotting stability circles, you can specify a frequency range. Use SSB to plot stability circles at the input port, and use LSB to plot those at the output port. You can specify a limited frequency range for these contours.

Level (dB) specifies the gain or noise figure value in dB for circles plotted against frequency.

Frequency Range defines *Start*, *Stop*, and *Step* for circles plotted at the specified dB value.

If you do not type in values for the frequency range, a circle is plotted for every simulated frequency for which a circle with the specified value exists.

Frequency specifies the spot frequency for circles plotted against a design variable.

Level Range defines *Start*, *Stop*, and *Step* for circles plotted for the specified spot frequency.

Gain is the value of gain in dB for which gain circles are plotted.

Noise is the value of noise figure in dB for which noise circles are plotted.

Plot buttons and cyclic fields at the bottom of the form generate the plots. For S, Y, Z, or H parameters, generate plots for ports 1 through 3 by clicking the appropriate button at the bottom of the form. To generate plots for the other ports, use the cyclic fields beside the buttons to specify the output and incident ports, and then click the *S*, *Y*, *Z*, or *H* button to generate the plot.

Annotate Voltage Levels

High Layer specifies the layer on which to display voltages equal to or greater than the value you specify for *Voltage High*.

Voltage High defines the high voltage threshold level between the mid and high display regions.

Mid Layer specifies the layer on which to display voltages less than *Voltage High* and greater than *Voltage Low*.

Voltage Low defines the low voltage threshold level between the mid and low display regions.

Low Layer specifies the layer on which to display voltages equal to or less than the value you specify for *Voltage Low*.

Analysis Type lets you choose the logic-level type you want to display.

> **DC** displays DC voltage levels.
>
> **transient** displays transient voltage levels.

Time lets you type in the time at which to display the transient voltage levels when transient is the analysis type selected.

Note: When you call up the Annotate Logic Level form, the analog circuit design environment removes from your schematic any probes created using a previous tool.

OK applies the logic-level probes you have chosen to the schematic and removes the Annotate Logic Level form.

Apply applies the logic-level probes to the schematic and does not remove the form.

Cancel removes the logic-level probes from the schematic.

Setting Outputs

Name (opt.) is an optional name for the signal, which appears in the *Table Of Outputs* list box and in the Waveform window.

Expression is the calculator expression to plot, save, or march.

Calculator buttons are displayed only while no net or terminal is selected in the *Table Of Outputs* list box.

> **Open** opens the calculator.
>
> **Get Expression** copies the expression in the calculator buffer into the expression field.
>
> **Close** dismisses the calculator window.

Will Be changes depending on whether an expression or a signal is selected.

> **Plotted/Evaluated** plots or prints the value of the expression after each simulation.

Add creates the output you set up in the *Selected Output* area.

Delete removes the highlighted output. Click in the *Table Of Outputs* list box to highlight an output.

Change updates the highlighted output with the new settings in the *Selected Output* area.

Next moves the highlight to the next signal or expression in the *Table Of Outputs* list box. This allows you to make changes to consecutive entries in the *Table Of Outputs* list box without clicking on each entry.

New Expression clears the *Selected Output* area so you can type in a new output.

Noise Summary

Type is the method of computing the noise.

> **spot noise** produces a noise summary at a given frequency.
>
> **integrated noise** produces a noise summary integrated over a frequency range using the specified weighting.
>
> **noise unit** determines the units used for this summary.

Frequency Spot (Hz) is the frequency at which spot noise is calculated. The default frequency is 1K.

From (Hz) is the lower limit of the integrated noise range.

To (Hz) is the upper limit of the integrated noise range.

weighting determines if integrated noise from one frequency needs to be considered more critical than from another frequency.

> **flat** integrates noise uniformly throughout the frequency range.
>
> **from weight file** integrates noise proportionately based on the weighting functions specified in the file identified in the field.

FILTER provides a method of limiting the summary report to include only some of the device types. In the list box, you can select those devices that you want included in the report.

> **include All Types** automatically selects and highlights all device types named in the list box. Click an entry to remove the highlighting and to leave it out of the summary.
>
> **include None** automatically deselects all device types named in the list box. Highlighting is removed from all items in the list box.

include instances lists devices to be included in the noise summary.

> **Select** lets you select devices to include from the list box.
>
> **Clear** removes all instances from the *include instances* list.

exclude instances lists devices to exclude from the noise summary.

> **Select** lets you select devices to exclude from the list box.
>
> **Clear** removes all instances from the *exclude instances* list.

TRUNCATE AND SORT

truncate limits the number of instances included in the summary based on their noise contribution.

> **none** includes all instances that were not excluded with the *exclude instances* list.
>
> **by number** limits the summary to the number of the largest contributors specified in *top*.
>
> **by rel. threshold** limits the summary to devices and noise contributors that contribute more than the percentage of the total noise specified in *noise %*.
>
> **by abs. threshold** limits the summary to any devices or noise contributors that contribute more than the amount specified in *noise value*.

sort by determines the order of the report.

> **noise contributors** sorts the report from the largest noise contributor to the smallest.
>
> **composite noise** sorts the report by the total noise contribution of each device. Each device entry contains the percentage of the noise contribution from this device and the noise contribution from each of its contributors.
>
> **device name** produces the same format as *composite noise* but sorts it in alphabetical order by device instance name.

Save Results

Save As lists the name of the directory that contains your results. The default is `schematic-save`.

Comment (optional) lets you type comments so that you can more easily differentiate simulation results.

Current Directory lists the current directory. This field cannot be edited. You use the list box above this field to navigate through directories.

Select Results

Results Directory is the directory in which the simulation results for the selected simulation are saved.

Delete Results

Results Directory lists the default directory in which results are saved.

UNIX Browser

File lists the selected file or directory.

Current Directory lists the directory being viewed in the list box.

11

Hspice Direct Support

Introduction

The *Analog Design Environment* (ADE) contains a direct integration of the *Hspice* simulator. ADE's *Hspice* integration (*HspiceS*) used to be based on the socket methodology, which required edit permissions to the schematic being simulated, and caused usage issues such as subcircuit name mapping. These restrictions have been lifted with the direct interface.

There are several advantages of the direct simulator integration approach over the socket simulator integration approach, namely:

- **Improved Performance in Netlisting**

  Netlisting is much faster in the direct approach, as *cdsSpice* is not involved and there is only one pass (which means, no raw netlisting before the final netlisting). Also, unlike the socket approach, the direct approach supports incremental netlisting. This ensures enhanced performance when incremental updates are performed in a design and then netlisted.

- **Better Readability of Netlists**

  The netlists are truly hierarchical and all numeric values in the netlist are more readable. For example in *Hspice* socket, the numeric values are changed from -5.0 to -5.0000000 in the final netlist. Also, the sub-circuits are no longer unfolded. The sub-circuits are also no longer mapped unless necessary.

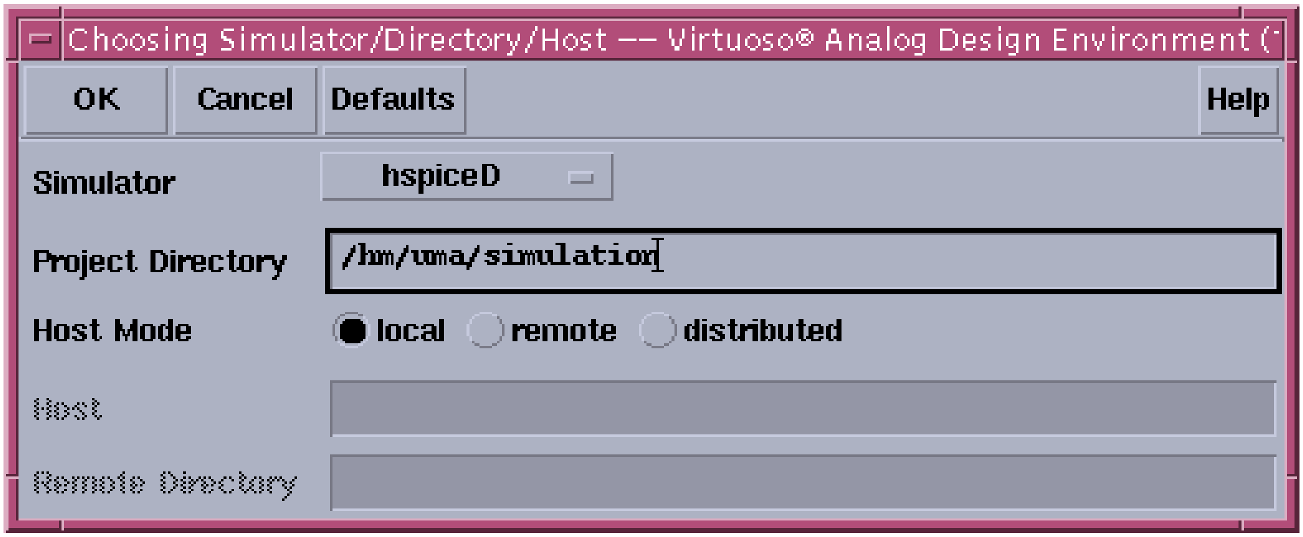
- **Read-only Designs can be Simulated, Provided they are Extracted**

  A limitation of socket netlisting is that the top cell of a design needs to be editable before the design can be netlisted. The direct approach however, allows read only designs to be simulated.The only pre-requisite being, that the design needs to be extracted first, so that connectivity information is written to the database.

- **Advanced Evaluation of Operators**

Direct netlisting supports the evaluation of ternary operators (Example, `(iPar("r")>2e-3?200e-3:400e-3`), whereas the same is not supported by socket netlisting.

In order to use the *Hspice Direct* simulator, you need to first select it in the *Choosing Simulator/Directory/Host* form:


Choosing Simulator/Directory/Host –– Virtuoso® Analog Design Environment (
OK
Cancel
Defaults
Help

Simulator: hspiceD

Project Directory: /hm/uma/simulation

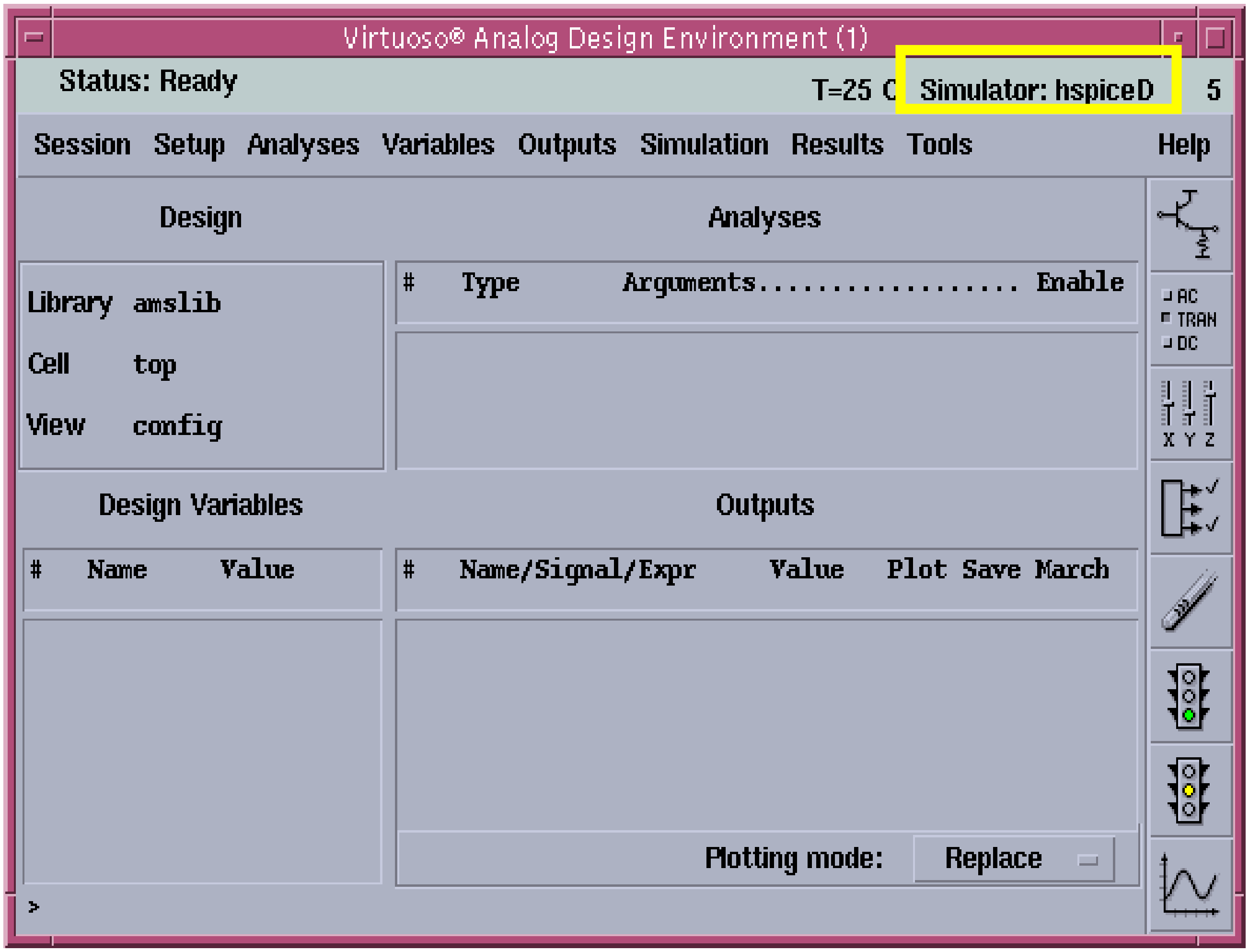
Host Mode: ● local ○ remote ○ distributed

Host

Remote Directory

For detailed information about the form, refer to the section Choosing Simulator/ Directory/Host.

The *Virtuoso Analog Design Environment* window displays with the *hspiceD* simulator selected:



Virtuoso® Analog Design Environment (1)

Status: Ready T=25 C Simulator: hspiceD 5

Library amslib

Cell top

View config

# Type Arguments.................. Enable

# Name Value

# Name/Signal/Expr Value Plot Save March

Plotting mode: Replace

>

Libraries

The following cells of the `analogLib` library are updated to contain *HspiceD* views. The *HspiceD* simInfo, CDF parameters and netlisting procedures have been added to all these `analogLib` cells:

bcs	bvs	cap	cccs	ccvs	core
diode	iam	idc	iexp	ind	iopamp
ipulse	ipwl	ipwlf	npn	isffm	isin

ixfmr nbsim nbsim4 njfet nmes nmes4
nmos nmos4 pbsim pbsim4 pcapacitor pdiode
pjfet pmos pmos4 pnp presistor res
schottky vam tline u1wire u2wire u3wire
u4wire u5wire usernpn userpnp vccap vccs
vcres vcvs vdc vexp vpulse vpwl
vpwlf vsffm vsin winding xfmr zener
iprobe pinductor mind pmind pvccs2 pvccs3
pvcvs pvcvs2 pvcvs3 pvccs

Features

The use model of the Analog Design Environment for the *HspiceDirect* simulator is very similar to that of the *Spectre Direct*/*Hspice Socket* interface. Most of the options work in the same way with a few differences.

Model Libraries

The *Model Library Setup* form remains essentially the same. You can enter model file names into the *Model Library File* field. The listbox displays the list of model files to be included. You can also include an optional *Section* field. When the *Section* field for a particular model file is defined, the netlist will contain the statement,

```
    .LIB “<modelLibraryFile>” <section>
```

When the *Section* field is not defined, the netlist will contain the statement,

```
    .INCLUDE “<modelLibraryFile>”
```

For detailed information about the form, refer to the section Model Library Setup.

Distributed Processing Support

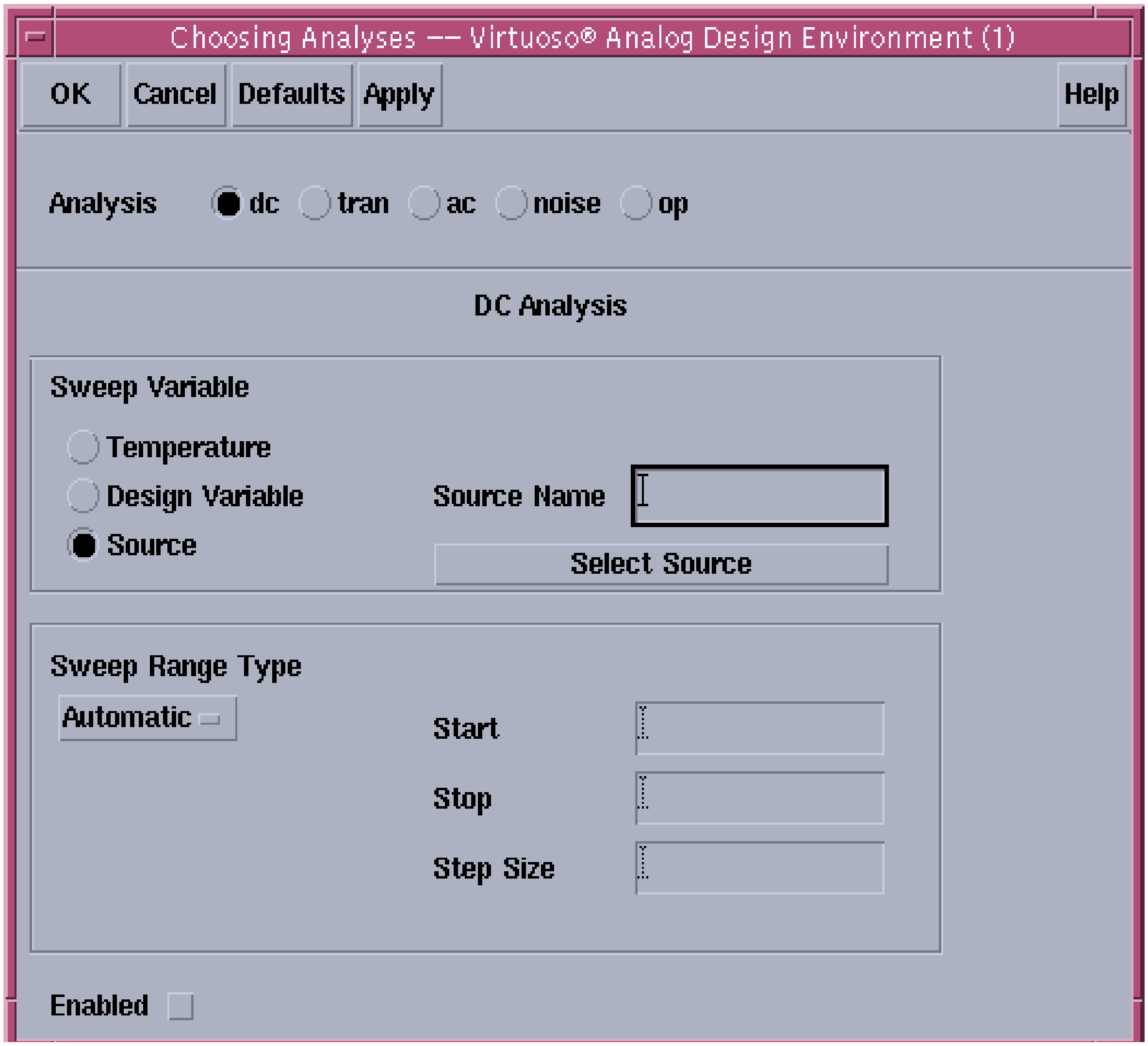
The Distributed Processing mode is supported only for normal simulation and parametric analysis. For detailed information about Distributed Processing, refer to the *Virtuoso Distributed Processing User Guide*.

Running Analyses

The *Choosing Analyses* form enables you to set up and run an analysis. This form is explained in details in the Setting Up for an Analysis chapter of this book. Refer to this section for details about each analysis.

The analyses that are supported are: *DC, Transient, AC, Noise* and *OP*. To run an analysis, select it in the *Choosing Analyses* form. The form re-displays to show the fields that are required for the selected analysis.

❑ To run a DC analysis, click the *dc* radio button in the *Analysis* section of the *Choosing Analyses* form.



Analysis ● dc ○ tran ○ ac ○ noise ○ op

○ Temperature
○ Design Variable
● Source

Source Name

Automatic

Start

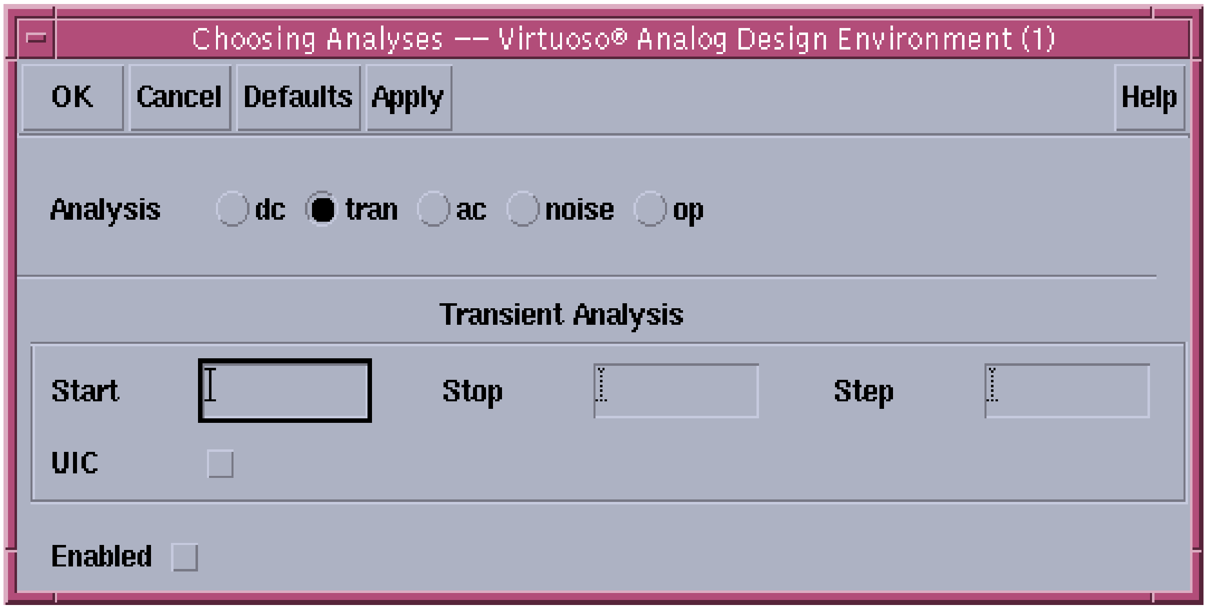
Stop

Step Size

Enabled □

This form reflects the different types of DC sweep variables and sweep range types.

- ❑ To run a transient analysis, click the *tran* radio button in the *Analysis* section of the *Choosing Analyses* form.


Choosing Analyses -- Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help

Analysis ○ dc ● tran ○ ac ○ noise ○ op

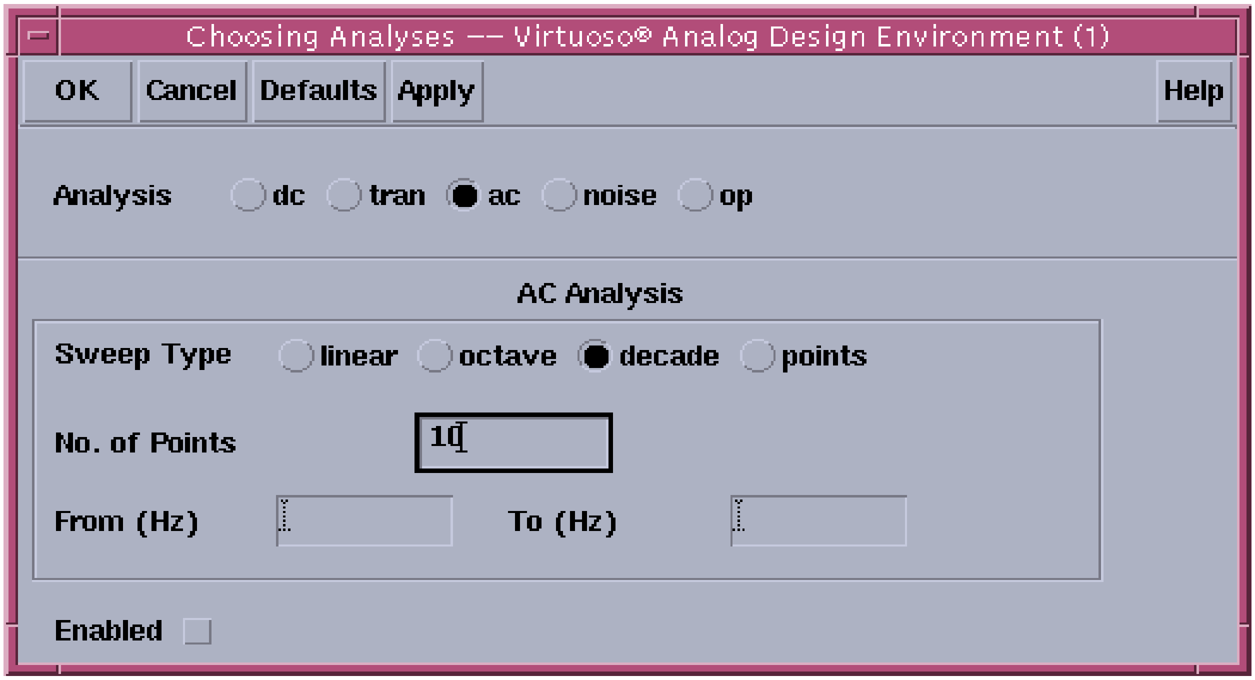
Transient Analysis

Start Stop Step

UIC □

Enabled □

- ❑ To run an AC analysis, click the *ac* radio button in the *Analysis* section of the *Choosing Analyses* form.


Choosing Analyses -- Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help

Analysis ○ dc ○ tran ● ac ○ noise ○ op

AC Analysis

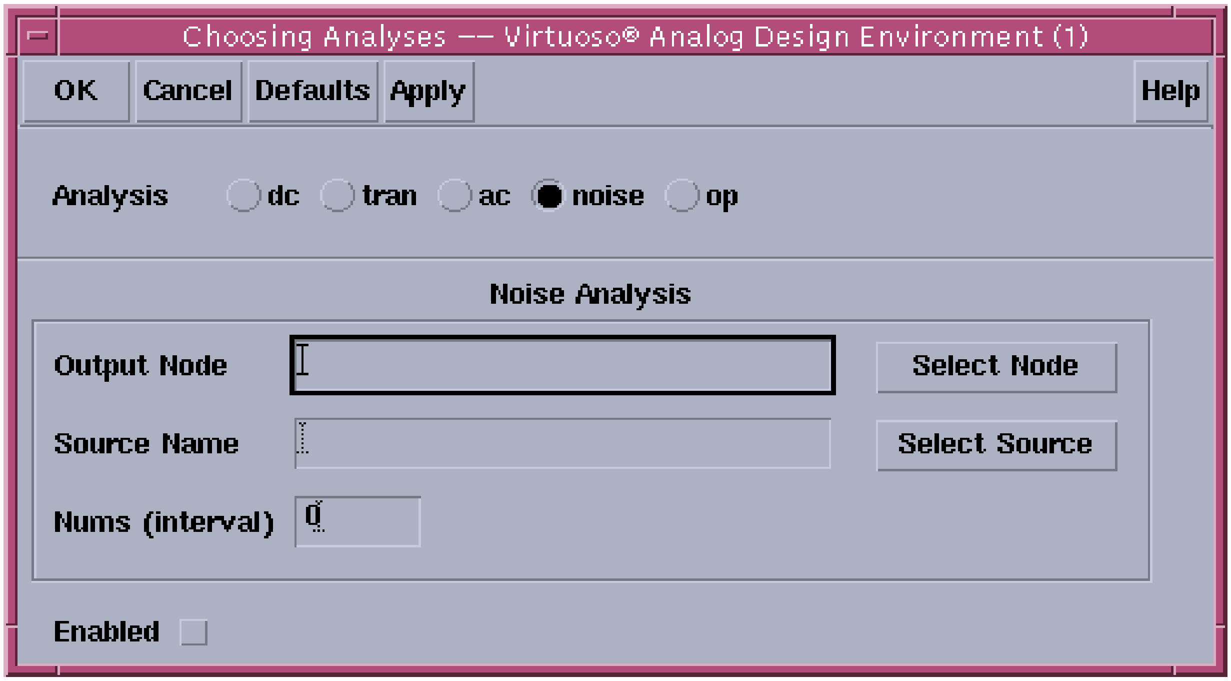
Sweep Type ○ linear ○ octave ● decade ○ points

No. of Points 10

From (Hz) To (Hz)

Enabled □

- To run a noise analysis, click the *noise* radio button in the *Analysis* section of the *Choosing Analyses* form.


Choosing Analyses –– Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help
Select Node
Select Source

Analysis ○ dc ○ tran ○ ac ● noise ○ op

Noise Analysis

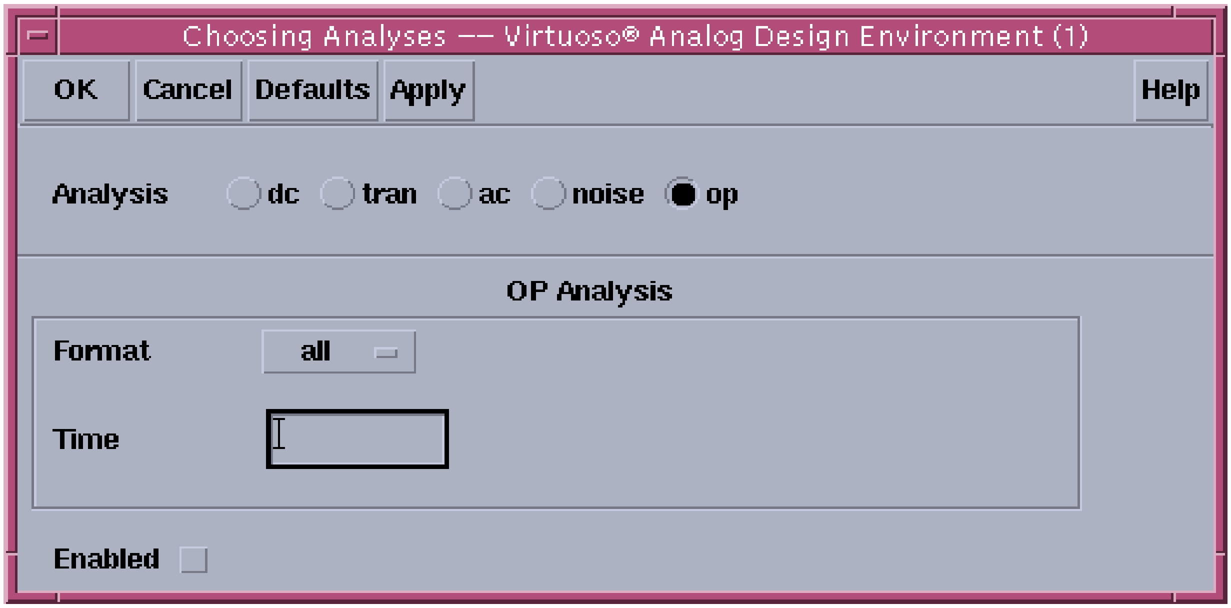
Output Node

Source Name

Nums (interval) 0

Enabled □

- To run an OP analysis, click the *op* radio button in the *Analysis* section of the *Choosing Analyses* form.


Choosing Analyses –– Virtuoso® Analog Design Environment (1)
OK
Cancel
Defaults
Apply
Help

Analysis ○ dc ○ tran ○ ac ○ noise ● op

OP Analysis

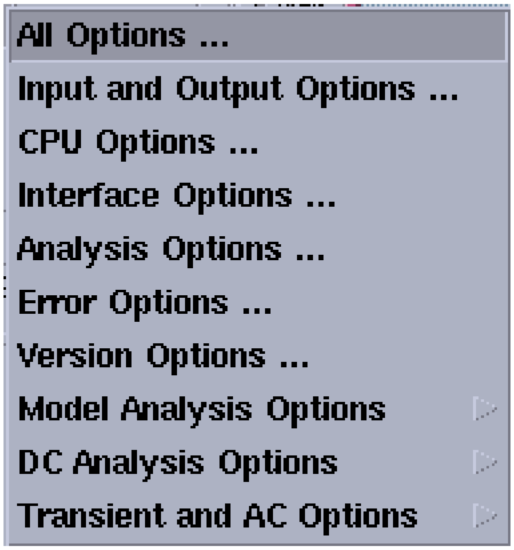
Format all

Time

Enabled □

Analog Options

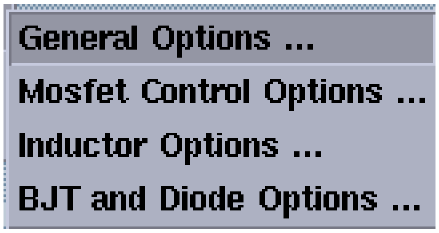
You can specify appropriate Hspice simulator options using *Simulation – Analog Options:*



These options can be used to modify various aspects of the simulation, including output types, accuracy, speed, and convergence. For details about the Analog Options, refer to *HSPICE/SPICE Interface and SPICE 2G.6 Reference Manual*.

Model Analysis Options

The *Model Analysis Options* have been grouped as follows:

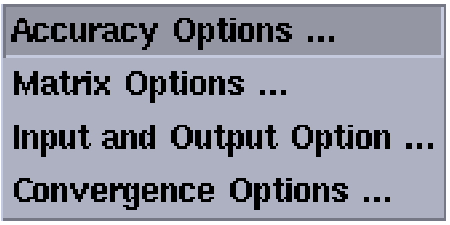


- Click *Model Analysis Options – General Options* to specify DCAP, HIER_SCALE, MODMONTE, MODSRH, SCALE, TNOM using the *Hspice General Model Options* form.
- Click *Model Analysis Options – Mosfet Control Options* to specify CVTOL, DEFAD, DEFAS, DEFEL, DEFNRD, DEFNRS, DEFPD, DEFPS, DEFW, SCALM, WL using the *Hspice Mosfet Control Options* form.

- Click *Model Analysis Options – Inductor Options* to specify GENK, KLIM using the *Hspice Inductor Options* form.
- *Model Analysis Options – BJT and Diode Options* to specify EXPLI using the *Hspice BJT and Diode Options* form.

DC Analysis Options

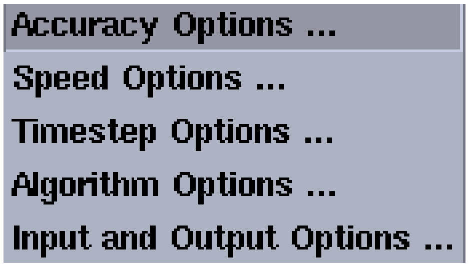
The *DC Analysis Options* have been grouped as follows:



- Click *DC Analysis Options – Accuracy Options* to specify ABSH, ABSI, ABSMOS, ABSVDC, DI, KCLTEST, MAXAMP, RELH, RELI, RELMOS, RELV, RELVDC using the *Hspice DC Accuracy Options* form.
- Click *DC Analysis Options – Matrix Options* to specify ITL1. ITL2, NOPIV, PIVOT, PIVREF, PIVREL, PIVTOL using the *Hspice Matrix Options* form.
- Click *DC Analysis Options – Input and Output Option* to specify CAPTAB, DCCAP, VFLOOR using the *HspiceDC Input and Output Options* form.
- Click *DC Analysis Options – Convergence Options* to specify CONVERGE, CSHDC, DCFOR, DCHOLD, DCON, DCSTEP, DV, GMAX, GMINDC, GRAMP, GSHUNT, ICSWEEP, ITLPTRAN, NEWTOL, OFF, RESMIN using the *HspiceConvergence Options* form.

Transient and AC Options

The *Transient and AC Analysis Options* have been grouped as follows:



- ➤ Click *Transient and AC Options – Accuracy Options* to specify ABSH, ABSV, ACCURATE, ACOUT, CHGTOL, CSHUNT, DI, GMIN, GSHUNT, MAXAMP, RELH, RELI, RELQ, RELV, RISETIME, TRTOL using the *Hspice Transient and AC Options* form.
- ➤ Click *Transient and AC Options – Speed Options* to specify AUTPSTOP, BKPSIZ, BYPASS, BYTOL, FAST, ITLPZ, MBYPASS, TRCON using the *Hspice Speed Options* form.
- ➤ Click *Transient and AC Options – Timestep Options* to specify ABSVAR, DVDT, FS, FT, IMAX, IMIN, ITL5, RELVAR, RMAX, RMIN, SLOPETOL, TIMERES using the *Hspice Timestep Options* form.
- ➤ Click *Transient and AC Options – Algorithm Options* to specify DVTR, IMAX, IMIN, LVLTIM, MAXORD, METHOD, MU, PURETP, TRCON using the *Hspice Algorithm Options* form.
- ➤ Click *Transient and AC Options – Input and Output Options* to specify INTERP, ITRPRT, MEASFAIL, MEASSORT, PUTMEAS, UNWRAP using the *Hspice Transient Input and Output Option*s form.

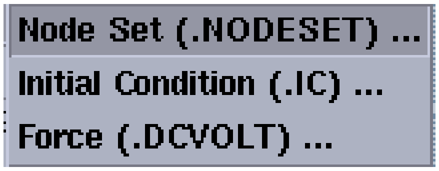
Important

All the *Analog Options* mentioned are supported by the *Hspice* version `2003.3.` Please ensure that you are using a compatible version of *Hspice.*

Output Log

This displays the file `hspice.out` found under the `psf` directory. This is the file to which the hspice output is re-directed.

Convergence Aids



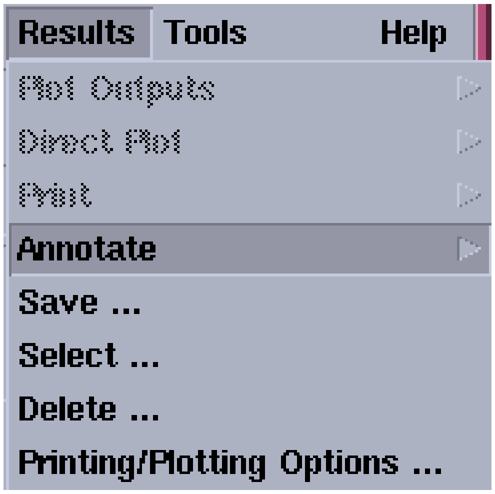
Click *Convergence Aids – Node Set (.NODESET)* to initialize specified nodal voltages for a DC operating point analysis. The`.NODESET` statement is generally used to correct convergence problems in a DC analysis. Setting nodes in the circuit to values that are close to the actual DC operating point solution enhances the convergence of the simulation. The *Select Node Set* form works in the same way as the Spectre Direct/Hspice Socket interface. The netlist will contain the `.NODESET` statement line.

Click *Convergence Aids – Initial Condition (.IC*) or *Convergence Aids – Force (.DCVOLT)* to set the transient initial conditions. The initialization depends on whether the UIC parameter is included in the .TRAN analysis statement. If the UIC parameter is specified in the .TRAN statement, the Hspice simulator does not calculate the initial DC operating point. Consequently, the transient analysis is entered directly.

The *Select Initial Condition Set* and the *Select Force Node Set* forms work in the same way as the *Spectre Direct/Hspice Socket* interface. The netlist contains the `.IC` and the `.DCVOLT` statement line, whichever the case may be.

Results

You can save, select, delete, restore, plot and print a set of simulation results using the *Results* menu.



The following menus have been removed from the *Hspice Direct* interface as the *Hspice* simulator does not write the specified data in the `psf` files:

- *Plot Outputs – Noise*
- *Direct Plot – Equivalent Output Noise*
- *Direct Plot – Equivalent Input Noise*
- *Direct Plot – Squared Output Noise*
- *Direct Plot – Squared Input Noise*
- *Direct Plot – Noise Figure*
- *Print – Model Parameters*
- *Print – Noise Parameters*
- *Print – Noise Summary*
- *Annotate – Model Parameters*

The noise data is written by the *Hspice* simulator in the `hspice.out` file. Use the menu *Simulation – Output Log* to view the simulator output file.

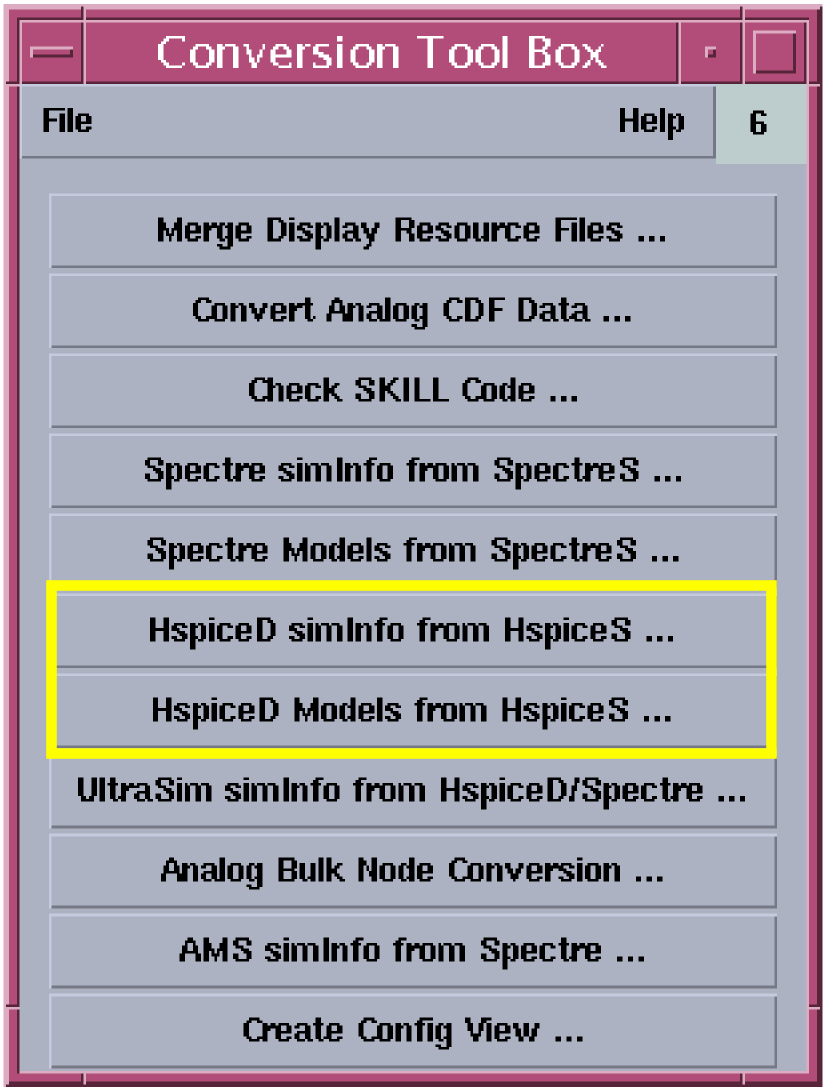
Advanced Analysis Tools Support

The tools that are supported are *Parametric*, *Corners and Optimization. Monte Carlo Circuit Conditions* are not supported in this release as this requires native support from the simulator.

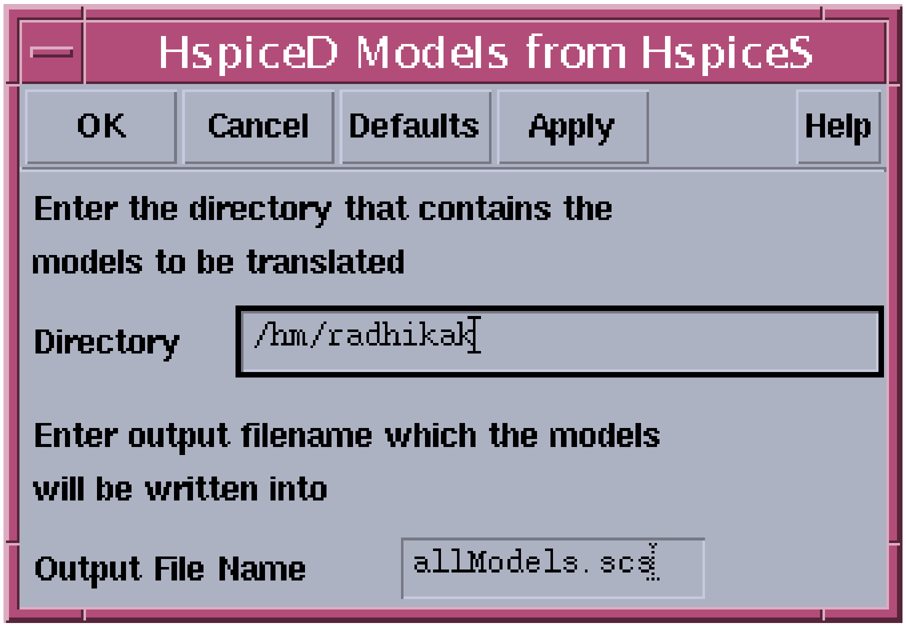
Converting Libraries

You can migrate existing *HspiceS* libraries and model files using the *Conversion Tool Box*. The *Conversion Tool Box* is a used to convert design libraries and associated technology data, and to prepare files for conversion to Direct simulation. To bring up the *Conversion*

Tool Box window, click *Tools – Conversion Tool Box* in the CIW (Command Interpreter Window). The window displays:
.


Conversion Tool Box
File
Help
6
Merge Display Resource Files ...
Convert Analog CDF Data ...
Check SKILL Code ...
Spectre simInfo from SpectreS ...
Spectre Models from SpectreS ...
HspiceD simInfo from HspiceS ...
HspiceD Models from HspiceS ...
UltraSim simInfo from HspiceD/Spectre ...
Analog Bulk Node Conversion ...
AMS simInfo from Spectre ...
Create Config View ...

A new button (*HspiceD models from HspiceS...*) has been added to the Conversion Tool Box. Click this button to invoke the *HspiceD Models from HspiceS* window.



Enter the directory that contains the models to be translated

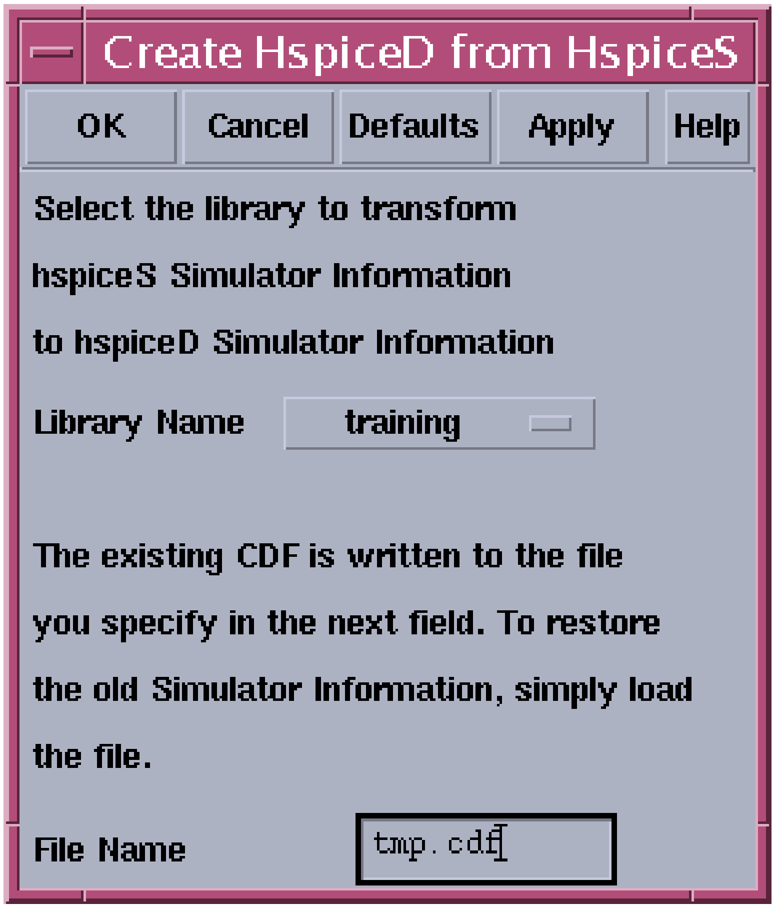
Directory /hm/radhikak

Enter output filename which the models will be written into

Output File Name allModels.scs

This utility locates all the *HspiceS* model files in a directory, translates them into *HspiceD* models, and places the translated models in a single file. This can be included to simulate a circuit (using the *Hspice Direct* interface) by adding the file through the *Model Libraries* form. For detailed information about the form, refer to the section, Model Library Setup. The *HspiceS* model files cannot be translated from two different directories. To do so, use the unix command `cat` to merge the file created from 2 different directories.

To transform *HspiceS* simulator information to *HspiceD* simulator information, click the *HspiceD simInfo from HspiceS* button. The Create *HspiceD from HspiceS* window displays.



Select the library to transform
hspiceS Simulator Information
to hspiceD Simulator Information

Library Name training

The existing CDF is written to the file
you specify in the next field. To restore
the old Simulator Information, simply load
the file.

File Name tmp.cdf

12

UltraSimVerilog

This chapter describes how to use the UltraSimVerilog simulator in the Cadence® mixed signal circuit design environment to simulate mixed signal designs, and provides the following information:

- “Interface Element Macro Models” on page 521
- “Netlisting Options” on page 527
- “Running a Mixed Signal Simulation” on page 529

Note: The 64-bit version of the Virtuoso® UltraSim simulator does not support UltraSimVerilog.

Refer to the following resources for additional information:

- *Virtuoso® Mixed Signal Circuit Design Environment User Guide*
- *Virtuoso Spectre® Circuit Simulator Reference* and *Virtuoso Spectre Circuit Simulator User Guide*
- *Virtuoso Schematic Composer User Guide*

Interface Element Macro Models

An interface element (IE) is a two-terminal device that connects two partitions and splits the original net. IEs are generated automatically for input and output terminals of digital components connected to interface nets.

IE attributes:

- Model the loading and driving impedance of digital instance terminals
- Convert voltages to logic levels and vice versa
- Transport events between two simulators

An IE model file is a text file that contains an IE primitive and other circuit components that characterize loading and nonlinear effects. The IE primitive inherits its parameters from the instantiation of the IE macro model.

When using the UltraSimVerilog simulator, you can choose to model an IE instance with either a primitive or a macro model file. IEs modeled as primitives reduce the need to have IE macro model files.

An IE is modeled as a primitive if its `macro` component description format (CDF) parameter is set to nil. Refer to Inherited CDF Parameters in the *Virtuoso Mixed Signal Circuit Design Environment User Guide* for more information.

Inline Subcircuit

The UltraSimVerilog simulator supports inline subcircuits in IE macro models. For the Virtuoso UltraSim™ simulator IE models, inline subcircuits are preferred over regular subcircuits. With an inline subcircuit, you do not have to specify the nesting level (nestlev) of the IE primitive (default is 0), and the IE primitive name appears at the same level as the interface net in the design hierarchy.

Interface Element Selection Rules

There are two modes for generating IEs: Detailed and nondetailed. Flat netlisting (FNL) supports both detailed and nondetailed IE generation. Hierarchical netlisting (HNL) supports only nondetailed IE generation. For UltraSimVerilog simulation, only HNL is available.

Simulation Accuracy and Performance

This section describes the following IE macro models for use with the mixed signal simulators, including UltraSimVerilog.

- “Analog-to-Digital (A2D) Models” on page 523
- “Digital-to-Analog (D2A) Models” on page 524
- “Analog-to-Analog-In (A2AI) Models” on page 526
- “Analog-to-Analog-Out (A2AO) Models” on page 527

Note: A2AO and A2AI IEs are not used in UltraSimVerilog simulation.

Analog-to-Digital (A2D) Models

Model Description

An analog-to-digital (A2D) IE macro model is needed to connect the input pin of a digital component to an interface net.

You can develop an A2D IE for a circuit simulator using the following parts:

- An A2D interface primitive that converts a voltage value to a logic state
- Optional analog primitives that model other characteristics of a digital input pin (for example, loading current and capacitance)

The A2D interface primitive performs the most basic analog-to-digital conversion step by sensing voltage and converting it to a logic state.

Optional analog primitives, such as resistors, capacitors, and transistors, can be used to connect the A2D IE to the actual analog circuitry in the design, so additional behaviors of the digital input pin are reflected in the A2D IE macro model. The implementation of these primitives is simulator-dependent.

Models for the Virtuoso UltraSim Simulator

The sample IE models provided by Cadence in the `analogLib` and `ieLib` libraries include IE macro model files for the Virtuoso UltraSim simulator.

Virtuoso UltraSim Example

The following example illustrates an instantiation of an A2D IE, MOS_a2d. Assume that it has the following CDF properties:

Property	Value
`macro`	(empty)
`a2d_v0`	1.5
`a2d_v1`	3.5
`a2d_tx`	1m

Because the macro property is an empty string, this instance is formatted as a primitive.

The CDF simulation information for this instance includes:

Field Name	Value
otherParameters	`macro`
instParameters	`timex vl vh`
propMapping	`nil vl a2d_v0 vh a2d_v1 timex a2d_tx`
componentName	Empty (default `prefix _ie` is used)

The *propMapping* value maps CDF properties to the Virtuoso UltraSim simulator properties. For example, the CDF property `a2d_v0` maps to the simulator property `vl`. The Virtuoso UltraSim simulator property values are as follows:

Virtuoso UltraSim Property	Value
`vl`	1.5
`vh`	3.5
`timex`	1 m

The Virtuoso UltraSim simulator property values are passed to the instance (`_ie99999` in this example), and the instance is formatted as

```
_ie99999 ( INetName1 0) a2d dest="99999" timex=1m vl=1.5 vh=3.5
```

Note: `a2d` is an IE primitive.

Digital-to-Analog (D2A) Models

Model Description

An instantiation of a digital-to-analog interface model is required for connecting an output pin of a digital component to an interface net. You define the circuit simulator interface model in a macro file, which is then included in the netlist.

A digital-to-analog (D2A) IE macro model for a circuit simulator can be created using the following parts:

- A D2A simulator primitive that converts a logic state to a voltage value

- Optional analog primitives that model other characteristics of a digital output Dpin (for example, drive and loading)

The D2A interface primitive performs the most basic digital-to-analog conversion step by converting a logic state to a voltage and timing relationship. It is implemented in slightly different forms in the Virtuoso UltraSim simulator and other SPICE simulators.

Optional analog primitives, such as resistors, capacitors, and transistors, can be used to connect the D2A interface primitive to the actual analog circuitry in the design. This allows additional behaviors of the digital output pin to be reflected in the D2A interface model. The implementation of these primitives is simulator-dependent.

Models for the Virtuoso UltraSim Simulator

See “Models for the Virtuoso UltraSim Simulator” on page 523 for more information.

Virtuoso UltraSim Example

The following example illustrates an instantiation of a D2A IE and CML3_d2a, from the `ieLib` library. Assume that it has the following properties:

Property	Value
`macro`	`"CML3_d2a"`
`d2a_vl`	-450 m
`d2a_vh`	0
`d2a_tr`	900 p
`d2a_tf`	800 p

Because the macro property is not empty, the instance is formatted as a macro model.

The CDF simulation information for this instance includes:

Field Name	Value
otherParameters	`macro`
instParameters	`fall rise val1 val0`

Field Name	Value
propMapping	`nil val1 d2a_vh val0 d2a_vl rise d2a_tr fall d2a_tf`
componentName	Empty (default `prefix _ie` is used)

The *propMapping* value maps CDF properties to Virtuoso UltraSim simulator properties. For example, the CDF property `d2a_tr` maps to the simulator property `rise`. The Virtuoso UltraSim simulator property values are as follows:

Virtuoso UltraSim Property	Value
`val0`	-450 m
`val1`	0
`rise`	900 p
`fall`	800 p

The Virtuoso UltraSim simulator property values are passed to the instance (`_ie99998` in this example) and the instance is formatted as

```
_ie99998 ( INetName2 0) CML3_d2a src="99998" fall=800p rise=900p   val1=0 val0=-
450m
```

Note: `CML3_d2a` is an IE macro.

Analog-to-Analog-In (A2AI) Models

Model Description

When the `ieDataType` on the instance or cell terminal is analog, the analog-to-analog-in (A2AI) model is used instead of the D2A model.

The A2AI and D2A differ only in the IE primitive and CDF parameter names. Refer to “Analog-to-Digital (A2D) Models” on page 523 for a general model description and example.

Note: A2A IEs are not used in simulation with UltraSimVerilog since the simulator requires hierarchical netlisting.

Analog-to-Analog-Out (A2AO) Models

Model Description

The analog-to-analog-out (A2AO) interface model is used instead of the A2D model when the `ieDataType` on the instance or cell terminal is analog.

The A2AO and A2D differ only in the IE primitive and CDF parameter names. Refer to “Analog-to-Digital (A2D) Models” on page 523 for a general model description and example.

See the *Virtuoso Mixed Signal Circuit Design Environment User Guide* for more information.

Netlisting Options

Both hierarchical netlisting (HNL) and flat netlisting (FNL) are available in the front-end mixed signal simulation flow. Mixed signal HNL and FNL differ in direct simulation: The UltraSimVerilog simulator supports HNL but not FNL.

Because HNL offers advantages over FNL, it is recommended that you use HNL unless you require features that are only available in FNL.

Verilog Netlisting Options

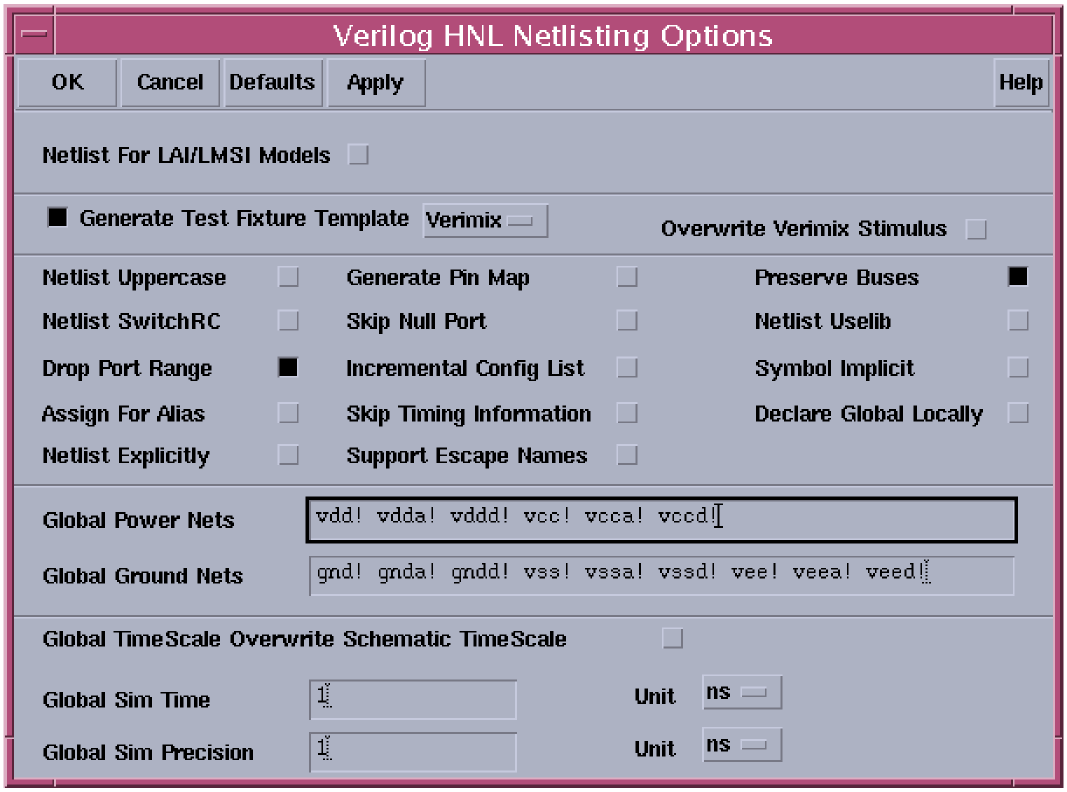
To access the Verilog® netlisting options:

1. Choose *Setup – Environment* in the Cadence Analog Design Environment simulation window.

   The Environment Options form appears.

2. Choose the *Verilog Netlist Option* button.

The Verilog HNL Netlisting Options form appears.



Netlist For LAI/LMSI Models □

☑ Generate Test Fixture Template Verimix   Overwrite Verimix Stimulus □

Netlist Uppercase	□	Generate Pin Map	□	Preserve Buses	☑
Netlist SwitchRC	□	Skip Null Port	□	Netlist Uselib	□
Drop Port Range	☑	Incremental Config List	□	Symbol Implicit	□
Assign For Alias	□	Skip Timing Information	□	Declare Global Locally	□
Netlist Explicitly	□	Support Escape Names	□		

Global Power Nets `vdd! vdda! vddd! vcc! vcca! vccd!`

Global Ground Nets `gnd! gnda! gndd! vss! vssa! vssd! vee! veea! veed!`

Global TimeScale Overwrite Schematic TimeScale □

Global Sim Time 1   Unit ns

Global Sim Precision 1   Unit ns

Some of the key Verilog netlisting options include:

- *Global Power Nets* and *Global Ground Nets*
- *Netlist SwitchRC*, *Skip Null Port*, and *Netlist Explicitly*

  Note: These options are only applicable for Verilog FNL.
- *Generate Test Fixture Template*, *Netlist Uppercase*, *Netlist SwitchRC*, *Global TimeScale Overwrite Schematic TimeScale*, *Global Sim Time*, and *Global Sim Precision*

  Note: These options are only applicable for Verilog HNL.

For more information about these options, refer to the *Verilog-XL Integration for Composer Reference*.

Hierarchical Netlisting

A typical mixed signal design contains hierarchical blocks and primitives. Blocks can contain lower-level instances and connectivity; primitives do not. HNL retains the hierarchical design and translates a non-primitive cellview into either a subcircuit for the analog simulator or a module for Verilog.

The mixed signal netlister creates the analog and digital netlists separately. The analog HNL netlists the analog partition and creates an analog directory. Verilog HNL netlists the digital partition and creates a digital directory.

- Analog blocks and primitives are netlisted by analog HNL.
- Digital blocks and primitives are netlisted by digital HNL.
- Mixed blocks are netlisted by both analog and digital HNL.

See the *Virtuoso Mixed Signal Circuit Design Environment User Guide* for more information.

Running a Mixed Signal Simulation

A mixed signal simulation involves a number of setup and processing steps.

- Setting Simulator Options
- Input Stimulus for HNL
- Setting Design Variables
- Choosing Analyses
- Running the Simulation
- Control and Debugging
- Viewing and Analyzing Simulation Output

The UltraSimVerilog simulator does not support the direct simulation non-batch control feature. Consequently, you cannot interactively control and debug both AHDL and Verilog modules.

Once your design is simulated, you can probe the layout and schematic views to determine specific characteristics of the design. This process may require numerous iterations until the results are acceptable.

Setting Simulator Options

Simulator Selection

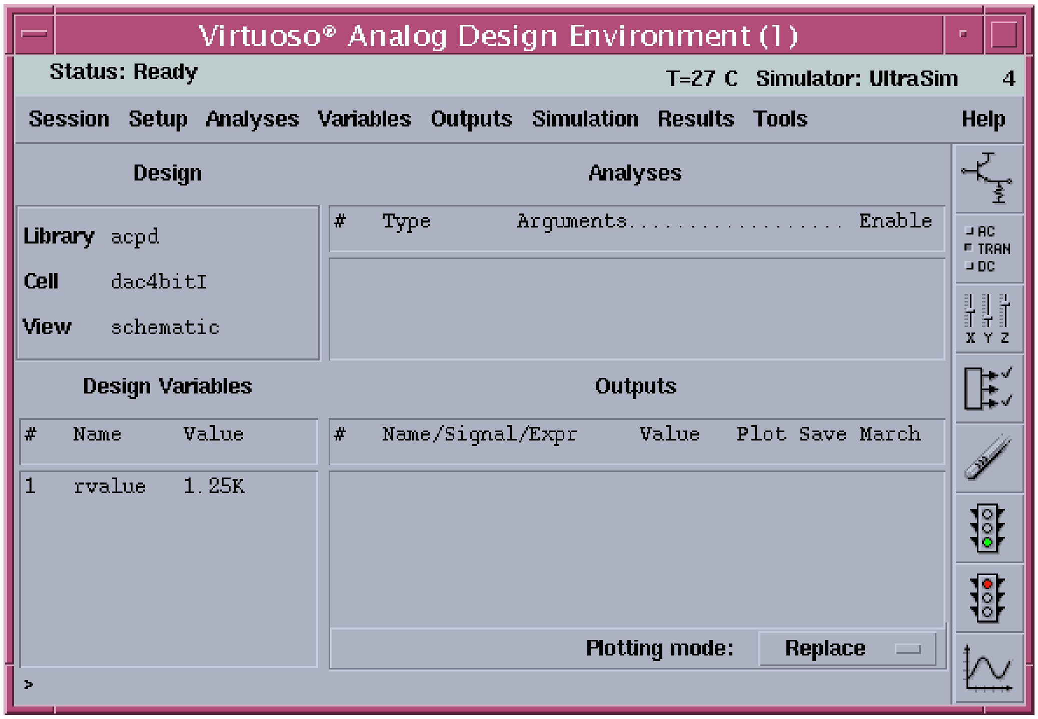
To select a simulator

1. Open the Cadence Analog Design Environment simulation window using one of the following methods:

   ❑ From the command interpreter window (CIW), choose *Tools – Analog Environment – Simulation*.

   ❑ From the Schematic window, choose *Tools – Analog Environment*.

   The simulation window appears.



Status: Ready T=27 C Simulator: UltraSim 4

Library acpd
Cell dac4bitI
View schematic

```
#   Type        Arguments.................. Enable
```

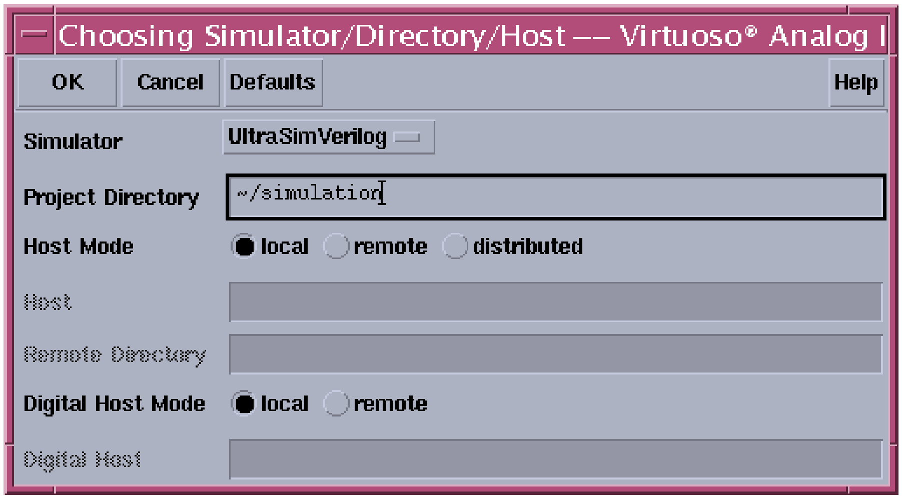
#	Name	Value
1	rvalue	1.25K

```
#   Name/Signal/Expr     Value   Plot Save March
```

Plotting mode: Replace

2. Choose *Setup – Simulator/Directory/Host*.

The Choosing Simulator/Directory/Host form appears.



Simulator: UltraSimVerilog

Project Directory: ~/simulation

Host Mode: ● local ○ remote ○ distributed

Host

Remote Directory

Digital Host Mode: ● local ○ remote

Digital Host

3. Choose *UltraSimVerilog* from the *Simulator* cyclic.
4. In the *Project Directory* field, type the name of the directory in which you will be working.
5. Click *OK*.

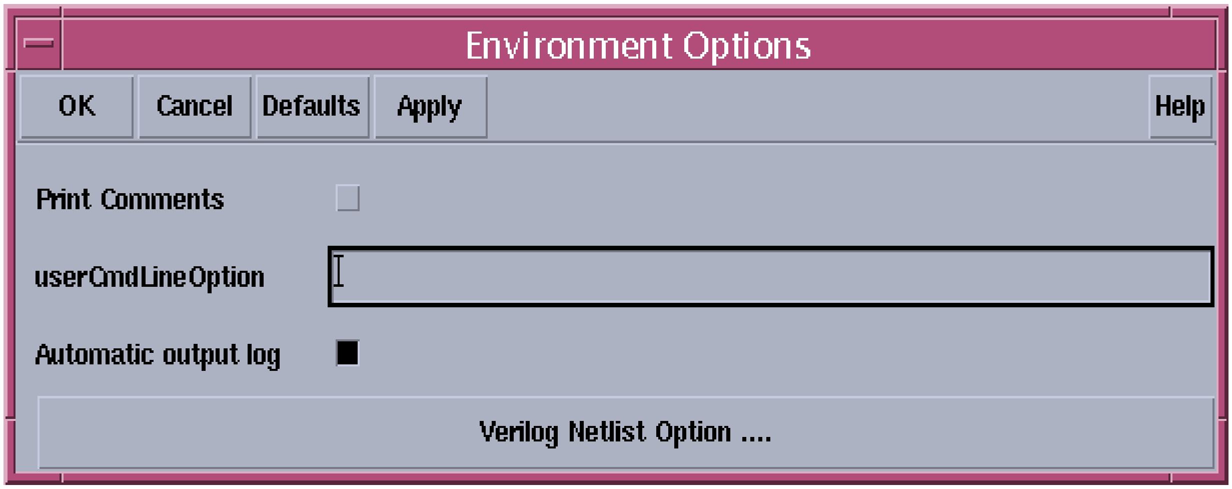
   Note: If the design data is not displayed in the simulation window, choose *Setup – Design* and select the appropriate design.

Environment Options

To set the environment options

1. Choose *Setup – Environment* in the simulation window.

The Environment Options form appears.



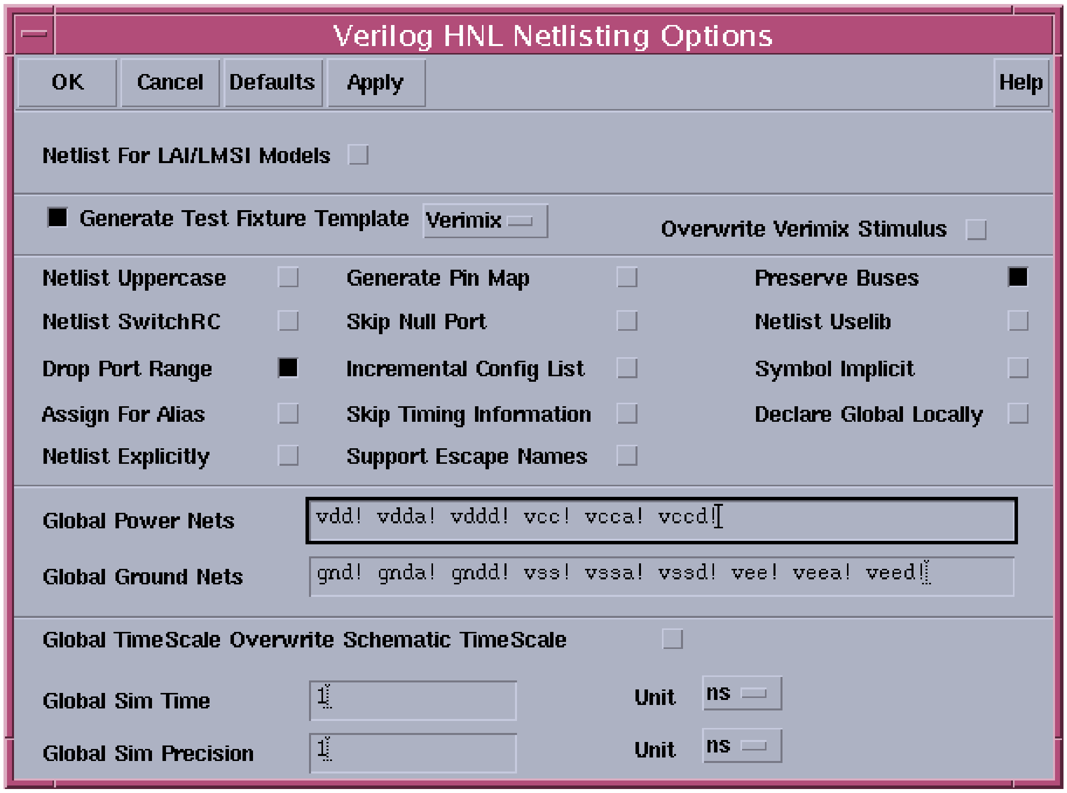
Print Comments □

userCmdLineOption

Automatic output log ■

2. Add information to the form, as needed, for your simulator and design.
3. Click the *Verilog Netlist Option* button to choose the appropriate options for netlisting.

The Verilog HNL Netlisting Options form appears.



Netlist For LAI/LMSI Models □

■ Generate Test Fixture Template Verimix

Overwrite Verimix Stimulus □

Netlist Uppercase □
Netlist SwitchRC □
Drop Port Range ■
Assign For Alias □
Netlist Explicitly □

Generate Pin Map □
Skip Null Port □
Incremental Config List □
Skip Timing Information □
Support Escape Names □

Preserve Buses ■
Netlist Uselib □
Symbol Implicit □
Declare Global Locally □

Global Power Nets vdd! vdda! vddd! vcc! vcca! vccd!

Global Ground Nets gnd! gnda! gndd! vss! vssa! vssd! vee! veea! veed!

Global TimeScale Overwrite Schematic TimeScale □

Global Sim Time 1 Unit ns

Global Sim Precision 1 Unit ns

4. Select *Verimix* in *Generate Test Fixture Template*.

   This following files are generated:

   `testfixture.template` Contains a test module declaration and include statements for IEs, `testfixture.verimix`, and `$shm_probe` definitions.

   `testfixture.verimix` A stimulus file from Verilog Integration that is compatible with the `testfixture.verilog` file.

   Note: If a `testfixture.verimix` file already exists and you need to create a new one, choose *Overwrite Verimix Stimulus* to generate a new file.

5. Specify the global power and ground nets in your design.

   An HDL global module `cds_global.v` is created with global power nets (`vcc_`) declared as supply1, and global ground nets (`gnd_`, `vss_`) declared as supply0. The module defines all other global nets that are not defined in either of the global net fields by using wire declarations.

6. Click *OK* to close the Verilog HNL Netlisting Options form.

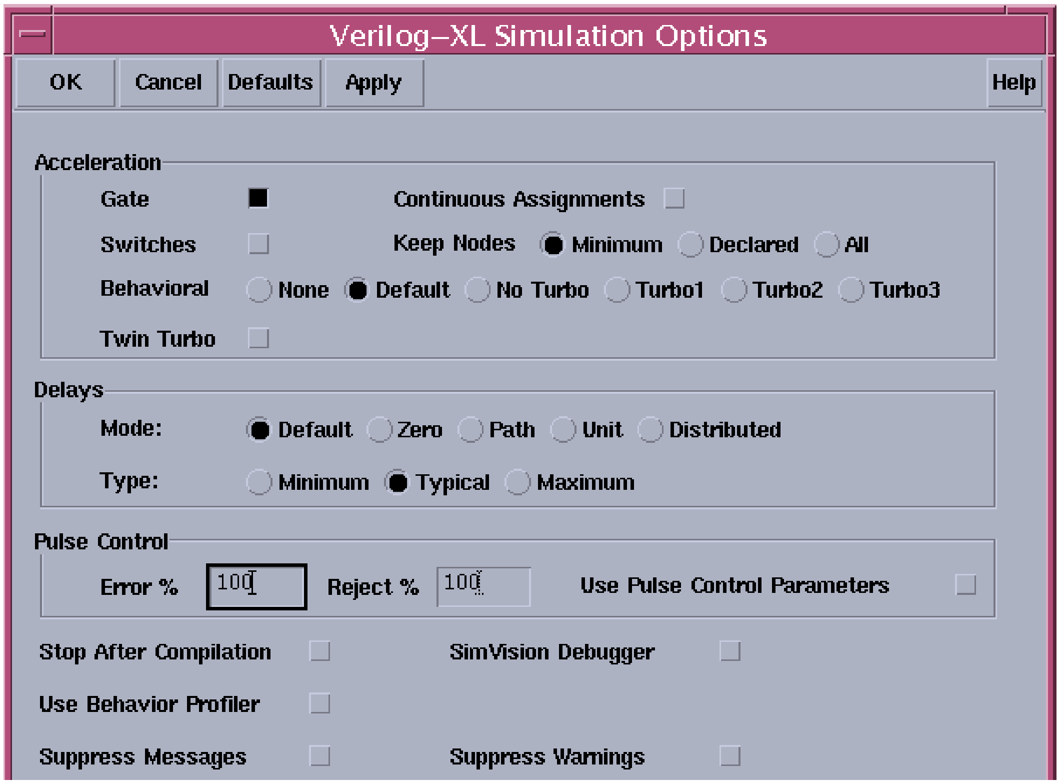
7. Click *OK* to close the Environment Options form.

Logic Simulator Options

To set simulator options for the logic simulator (Verilog)

1. In the Simulation window, choose *Simulation – Options – Digital*.

The Verilog-XL Simulation Options form is displayed.



Acceleration

Gate ☒ Continuous Assignments ☐

Switches ☐ Keep Nodes ● Minimum ○ Declared ○ All

Behavioral ○ None ● Default ○ No Turbo ○ Turbo1 ○ Turbo2 ○ Turbo3

Twin Turbo ☐

Delays

Mode: ● Default ○ Zero ○ Path ○ Unit ○ Distributed

Type: ○ Minimum ● Typical ○ Maximum

Pulse Control

Error % 100 Reject % 100 Use Pulse Control Parameters ☐

Stop After Compilation ☐ SimVision Debugger ☐

Use Behavior Profiler ☐

Suppress Messages ☐ Suppress Warnings ☐

2. Set the options for the logic simulator as needed.

   For details about the Verilog options, refer to the *Verilog-XL Integration for Composer Reference Manual*.

3. If you want to enter Verilog-XL commands directly or debug the simulation using the SimVision debugger, select *Stop After Compilation*.

   Note: The *SimVision Debugger* option is also selected (and cannot be deselected) if *Stop After Compilation* is selected.

   This stops Verilog-XL after compilation and lets you enter Verilog commands through the SimVision window (SimVision is the Verilog-XL graphical environment – see the *Verilog-XL User Guide* for information on how to use SimVision).

4. Click *OK*.

Input Stimulus for HNL

Analog stimulus for HNL can be supplied by using an analog stimulus block and file, or by using a digital stimulus block and file for digital stimulus.

HNL input stimulus rules and restrictions:

- Analog and digital stimulus blocks function the same in HNL and FNL
- Stimulus blocks can drive both analog and digital components
- In HNL and FNL, stimuli in an analog stimulus file can drive only analog nets and interface nets
- The analog stimulus file in HNL can be used to drive signals in only the top-level cellview
- In HNL and FNL, a digital stimulus file cannot drive analog components
- The file formats of HNL and FNL digital stimulus files are incompatible

Analog Stimuli

Analog Stimulus Block

An *analog stimulus block* or *instance* on the schematic can drive both analog and digital circuitry. This eliminates the need to supply input through the schematic for analog components that can be simulated using digital input.

Analog stimulus blocks include voltage sources, current sources, and behavioral instances.

To create an analog stimulus block or instance:

1. Create a behavioral view for the stimulus block.
2. Define the stimulus module.
3. Write SpectreHDL or Verilog-A in a module definition.

   The file syntax is the same for FNL and HNL.
4. Create a symbol view for the stimulus block.
5. Place the symbol in the top-level schematic and connect it to the appropriate terminals.

The following sample analog stimulus block shows the format of a file that implements a swept sinusoidal source.

```
`include "discipline.h"
`include "constants.h"
`define PI                3.14159
// -  Swept sinusoidal source
// sigout_p,sigout_n:output (val,flow)
// INSTANCE parameters
//      start_freq           = start frequency [Hz]
//      sweep_rate           = rate of increase in frequency [Hz/s]
//      amp                  = amplitude of output sinusoid (val)
//      points_per_cycle = number of points in a cycle of
//      the output []
// The instanteous frequency of the output is 'sweep_rate'
//      * 'time' plus 'start_freq'.
module swept_sine_src(sigout_p,sigout_n);
output sigout_p,sigout_n;
electrical sigout_p,sigout_n;
parameter real start_freq = 1 from (0:inf);
parameter real sweep_rate = 1;
parameter real amp = 1 from (0:inf);
parameter real points_per_cycle = inf from [6:inf];
    real freq;
    real phase;

    analog begin
        phase   = 2*`PI*(start_freq + sweep_rate / 2 *
        $realtime)*$realtime;

        freq = start_freq + sweep_rate * $realtime ; // =
        d/dt(phase)

        V(sigout_p,sigout_n) <+ amp*sin(phase);

        if (points_per_cycle != inf) begin
            // ensure that model is evaluated sufficiently often
            bound_step(1 / (freq*points_per_cycle));
        end
    end
endmodule
```

Analog Stimulus File

In HNL, use the *analog stimulus* file to connect stimuli to only the nets in the top-level cellview. This requirement is imposed by the circuit simulator because stimuli may not be allowed to connect to nets embedded in lower levels of the design hierarchy. Mapping nets or instance names in lower levels of the design hierarchy is allowed.

For example, if *in<3:0>* and *control* are signals existing in the top-level cellview, you can use voltage sources to drive [#/in<3>] and [#/control] in the stimulus file, as shown in the following example:

```
*comment, inst /inst<1> is mapped to [$/inst<1>]
*comment, inst /a/b is mapped to [$/a/b]
*comment, net /a/net<1> is mapped to [#/a/net<1>]
v0 [#/in<3>] 0 dc 5v
v1 [#/control] 0 dc 3v
```

Note: Analog stimuli supplied by an analog stimulus file can drive only analog nets and interface nets. Attempts to drive or refer to digital nets with analog stimuli in the stimulus file generates errors.

To generate an analog stimulus file:

1. Choose *Setup – Stimulus – Edit Analog* (or *Setup – Stimuli – Analog* if you are using the UltraSimVerilog simulator) in the simulation window.

   The Edit Stimulus File form appears.

   For mixed signal direct simulation (UltraSimVerilog), you can generate an analog stimulus file through the Setup Analog Stimuli form, or you can provide the analog stimulus in a text file. The information you enter on the Setup Analog Stimuli form is converted to a stimulus file. Refer to the *Virtuoso Analog Design Environment User Guide* for more information about entering analog stimuli.

2. Make stimulus changes as needed.

   You must use analog simulator language to define analog stimuli (file syntax is compatible with the analog design environment or ADE).

3. Change signal and instance names to ADE naming conventions.

   All instance and net names in the stimuli file must be specified in the database name space. Instance names must be enclosed within the [$] mapping macro and net names enclosed within the [#] macro.

4. Save the file and close the editor.

Digital Stimuli

Digital Stimulus Block

A *digital stimulus block* can drive both analog and digital input. This eliminates the need to supply input through the schematic for analog components that can be simulated using digital input.

To create a modified stimulus block:

1. Create a behavioral view for the stimulus block.
2. Define the stimulus module.
3. To force input, add Verilog commands to the module definition.

For more details, refer to the *Verilog-XL Reference* and the *Verilog-XL Integration for Composer Reference Manual*.

4. Create a symbol view for the stimulus block.
5. Place the symbol in the top-level schematic and connect it to the appropriate terminals.

The following sample of a Verilog stimulus block shows the format of the file.

```
//timescale set according to user specification
`timescale 10ns/10ns
//Define the Stimulus block
module Stim (tx, precharge);
    output [1:16] tx ;
    output precharge ;
//Defines the registers
    reg [1:16] tx ;
    reg precharge ;
initial begin
    tx = 16'h0000;
    precharge = 1'b0;
end
initial begin
    #2418 tx[3] = 1'b1;
    #17 tx[3] = 1'b0;
end
initial begin
    #1558 tx[6] = 1'b1;
    #17 tx[6] = 1'b0;
end
initial begin
    #1597 tx[7] = 1'b1;
    #17 tx[7] = 1'b0;
end
initial begin
    #37 precharge = 1'b1;
    #11 precharge = 1'b0;
    #27 precharge = 1'b1;
    #11 precharge = 1'b0;
    #333 precharge = 1'b1;
    #11 precharge = 1'b0;
    #38 precharge = 1'b1;
    #11 precharge = 1'b0;
    #27 precharge = 1'b1;
    #11 precharge = 1'b0;
end
endmodule
```

Note: If *Generate Test Fixture Template* in the Verilog HNL Netlisting Options form is set to Verimix, the HNL testfixture files (`testfixture.template` and `testfixture.verimix`) are automatically generated at netlisting time.

testfixture.verimix File

The `testfixture.verimix` file is the stimulus file that is included with the `testfixture.template` file. The stimulus file is a separate file, to prevent overwriting when the `testfixture.template` file is regenerated. Do not use OSS naming conventions in the Verilog HNL `testfixture.verimix` file, because name mapping is not used in the `testfixture.verimix` file. Use Verilog design names directly in the `testfixture.verimix` file.

Note: Testfixture files cannot be shared between FNL and HN (the formats of the files are not compatible).

To switch from HNL to FNL using the same simulation directory, replace HNL `testfixture.template` with FNL `testfixture.template`.

To switch from FNL to HNL using the same simulation directory, the HNL `testfixture.template` and `testfixture.verimix` files are created automatically. Edit the `testfixture.verimix` file to provide the necessary stimulus.

The following example shows a hierarchical `testfixture.template` file.

```
‘timescale 1ns / 1ns
module test;
wire out;
integer dc_mode_flag;
integer output_change_count;
integer max_dc_iter;
integer dc_iterations;
time vmx_time_offset;
pulledIE2Top top(out);
‘define verimix
‘ifdef verimix
//vms and dc iteration loop definitions
    ‘include "IE.verimix"

//please enter any additional stimulus
//in the testfixture.verimix file
    ‘include "testfixture.verimix"
//$shm_probe definitions
    ‘include "saveDefs"
‘endif
```

In HNL mode, the `testfixture.verimix` file must strictly conform to the Verilog Integration standard for stimulus files. Instance and net name mapping macros, such as [$] and [#], are not allowed. You can access or drive any instance or net in the design hierarchy because Verilog allows out-of-context references to instances and nets embedded within lower-level modules of the design hierarchy.

Note: You cannot use a digital stimulus to provide input directly to an analog gate (instead, use a stimulus block).

To create or edit a `testfixture.verimix` file:

1. Choose *Setup – Stimulus – Edit Digital*.

   A text editor window containing the `testfixture.verimix` file appears.

2. Make stimulus changes as needed for the simulation.

   The information in these files must be in Verilog Integration format. Refer to the raw netlist file from your design for signal and instance names in Verilog name space. For more information about Verilog stimulus information and syntax., see the *Verilog-XL Integration for Composer Reference Manual*.

3. Save the file and close the editor.

The following example shows a `testfixture.verimix` file:

```
// Verilog stimulus file.
// Please do not create a module in this file.
// Default verilog stimulus.
initial begin
    [#/A] = 1’b0;
    [#/B] = 1’b0;
    [#/C] = 1’b0;
end
```

If you create, delete, or change the name of an input terminal in your design, you may need to generate a new `testfixture.verimix` file, to avoid having the `testfixture.verimix` file erroneously refer to deleted signals or leave new signals uninitialized.

Use one of the following methods to modify the file:

1. From the Environment Options form, click on the *Verilog Netlist Option* button to access the netlisting options form.

2. In the Verilog HNL Netlisting Options form, select *Overwrite Verimix Stimulus* to create a new `testfixture.verimix` file at netlisting.

3. Edit the `testfixture.verimix` file for stimulus.

   or

4. From the Simulation window, choose *Setup – Stimulus – Edit Digital*.

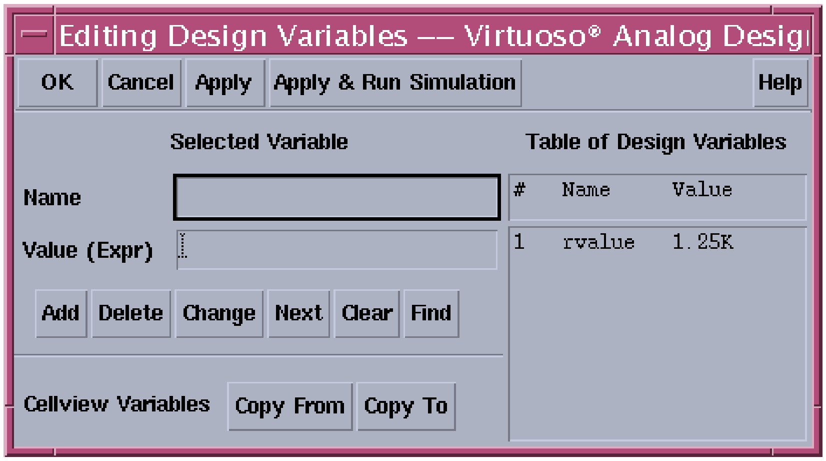
5. Edit the `testfixture.verimix` file directly.

Setting Design Variables

To add, modify, or delete design variable values:

1. Choose *Variables – Edit* in the simulation window.

The Editing Design Variables form appears.



#	Name	Value
1	rvalue	1.25K

2. To add a design variable:

    a. Type the name and value of the new variable in the *Selected Variable Name* and *Value* fields.

    b. Click *Add*.

    The new variable is added to the *Table of Design Variables* list box.

3. To modify a design variable:

    a. Click on the variable in *Table of Design Variables*.

    b. The variable name and its value are displayed in the *Name* and *Value* fields.

    c. Make changes to the name or value as needed.

    d. Click *Change*.

4. To delete a variable:

    a. Click on the variable in *Table of Design Variables*.

    b. Click *Delete*.

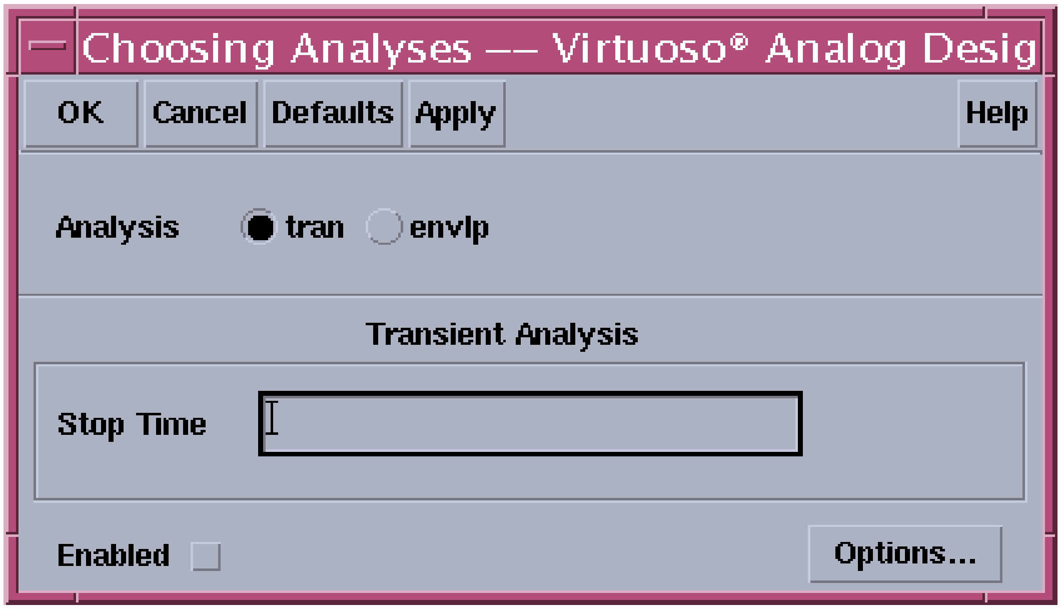
5. Click *OK* to return to the simulation window.

Choosing Analyses

To choose the analysis for this simulation:

1. Choose *Analyses – Choose* in the simulation window.

   The Choosing Analyses form appears.



Analysis ● tran ○ envlp

Transient Analysis

Stop Time

Enabled □

   The form contains different fields depending on the simulator you are using and the analysis you choose.

2. Click the *tran* button for transient analysis.
3. Set *Stop Time*.

   Note: Make sure *Enabled* is selected.

4. Click *OK*.

Running the Simulation

To run the simulation:

➤ Choose *Simulation – Run*.

The partitioner reads the design and creates the partitioning information, the IE generator creates IEs on the interface nets, and the netlister generates the analog and digital netlists from partitioning and IE generation information.

Control and Debugging

You can control and debug mixed signal simulations interactively, as well as set breakpoints, step through module code, and run, stop, and resume a simulation.

To debug Verilog modules, use the SimVision debugger or the Type-In window to enter Verilog-XL debugging commands.

➤ Choose the *Stop After Compilation* or *SimVision Debugger* option in the Verilog-XL Simulation Options window (see “Logic Simulator Options” on page 533 for more information)

During simulation, Verilog-XL execution starts in the SimVision window. Verilog-XL stops after initialization if *Stop After Compilation* is selected. You can then enter Verilog debugging commands through the SimVision debugger or through the Type-In window. See the *Verilog-XL User Guide* for information about SimVision.

Viewing and Analyzing Simulation Output

Once the simulation is complete, there are several ways to probe, view, and print values and to display waveforms. For more information on how to

- Select data to save and plot
- Plot and print data
- Use the waveform calculator and viewer

refer to Appendix B, “Waveform Tools in ADE,” the *WaveScan User Guide*, the *Analog Waveform User Guide* and the *Waveform Calculator User Guide*.

For information about the Virtuoso UltraSim simulator options, refer to the *Simulation Options* chapter of the *Virtuoso® UltraSim Simulator User Guide*.

A

Environment Variables

This appendix describes public environment variables that control the characteristics of the Analog Design Environment (ADE). You can customize the operation and behavior of Analog Design Environment products by changing the value of a particular environment variable.

This appendix lists environment variables belonging to the following products:

- ADE Simulation Environment
- Calculator
- Distributed Processing
- Spectre
- SpectreVerilog and SpectreSVerilog
- HspiceD
- AMS and UltraSim

ADE Simulation Environment

filteredSimList

Lets you control the simulator list, including the number of simulators and the sequence in which they are listed, in the Choosing the Simulator/Directory/Host form.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.startup” “filteredSimList” ‘string “ams Sim1
(UltraSim UltraSimVerilog) (spectre spectreVerilog) hspiceD”)
```

The parentheses can be used to group simulators. Groups of simulators are separated by dashed lines in the simulator list. ADE ignores a specified simulator name when it is not a valid one, that is, it does not have a SKILL context or a proper `.ini` file.

This command generates the list of available simulators as follows:

```
    ams
    --------
    UltraSim
    UltraSimVerilog
    --------
    spectre
    spectreVerilog
    --------
    hspiceD
```

Note: The simulator specified as `Sim1` is not listed because ADE did not consider it a valid simulator name.

Variable Type	string
Default Value	nil
Acceptable Values	simulator names separated by spaces
Window-Menu	*Virtuoso Analog Design Environment –* *Choosing Simulator/Directory/Host – Simulator*

saveDir

This variable identifies the directory in which the saved state file is to be copied. By default, saved state files are to be kept in the `.artist_states` directory in the home directory. You can change this path to another directory as needed.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv” “saveDir” ‘string “~/states/artist_states”)
```

Variable Type	string
Default Value	~/.artist_states
Acceptable Values	dir path
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – State Save Directory*

designEditMode

This variable lets you choose the default open mode for your designs. If you select true, your designs are opened in edit mode. If you select nil, your designs are opened in read-only mode.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “designEditMode” ‘boolean “t”)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Default Design Open Mode*

schematicBased

If this variable is set to true, it displays the Analog Design Environment menus on the Virtuoso Schematic window.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “schematicBased” ‘boolean “t”)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Schematic Menus*

windowBased

This variable lets you choose the way the Virtuoso® analog design software starts up your session. If it is true, open the Simulation window. Else do not open the simulation window.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “windowBased” ‘boolean “nil”)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Simulation Window*

saveQuery

Lets you choose whether you want to be reminded to save the state of your environment before making a change. If the option is on, you are prompted to save the state before your environment is changed. If the option is off, you can save the state manually by choosing *Session - Save State*, but you will not be prompted to do so.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “saveQuery” ‘boolean “nil”)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Query to Save State*

loadCorners

If this option is true, it preloads the Corners Java.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “loadCorners” ‘boolean “nil”)`

Variable Type	boolean
Default Value	t

Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Preload the Corners Java*

x

Lets you set the horizontal position of the left side of the Simulation window. A selection of 1 (the default) positions the window flush with the left side of your screen. Higher numbers move the Simulation window further to the right.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.window” “x” ‘int 200)
```

Variable Type	int
Default Value	1
Acceptable Values	Any number between 0 and 1200
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Window X Location*

y

Lets you set the vertical position of the top of the Simulation window. A selection of 1, positions the top of the window flush with the top of your screen. Higher numbers move the Simulation window further down the screen. the default positioning places the window about one third of the way down the screen.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.window” “y” ‘int 200)
```

Variable Type	int
Default Value	317
Acceptable Values	Any number between 0 and 1000
Window-Menu	*Virtuoso Analog Design Environment – Editing Session Options – Window Y Location*

simulator

Lets you specify the default simulator for the Analog Design Environment.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.startup” “simulator” ‘string “spectreVerilog”)`

Variable Type	string
Default Value	spectre
Acceptable Values	simulator name
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Simulator*

Example

To specify *Spectre* as the default simulator, add the following line to the `.cdsinit` file:

```
envSetVal( "asimenv.startup" "simulator" ’string  "spectre")
```

projectDir

Lets you specify the default simulation directory.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.startup” “projectDir” ‘string “/tmp/simulation”)`

Variable Type	string
Default Value	"~/simulation"
Acceptable Values	directory path
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Project Directory*

hostMode

Lets you specify a default local, remote or distributed simulation.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“asimenv.startup” “hostMode” ‘string “remote”)
```

Variable Type	string
Default Value	local
Acceptable Values	local, remote, distributed
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Host Mode*

host

Lets you specify a path to the host computer for remote simulation. You must specify a complete path.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“asimenv.startup” “host” ‘string “cds11938”)
```

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Host*

digitalHostMode

The variable digitalHostMode lets you choose default local or remote digital simulation.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“asimenv.startup” “digitalHostMode” ‘string “remote”)
```

Variable Type	string
Default Value	"local"
Acceptable Values	local, remote
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Digital Host Mode*

digitalHost

The variable digitalHost lets you specify a path to the host computer for a digital remote simulation. You must specify a complete path.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.startup” “digitalHostMode” ‘string “cds11939”)`

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Digital Host*

remoteDir

Lets you specify a path to the run directory for remote simulation. The remote directory name should be same as the local simulation directory name. You must specify a complete path.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.startup” “remoteDir” ‘string “/net/cds11938/hm/usr1/simulation”)`

Variable Type	string
Default Value	""
Acceptable Values	Unix path
Window-Menu	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Remote Directory*

autoPlot

Plots the entire plot set (including waveform expressions) automatically when each simulation is finished. When disabled, you can use *Results - Plot Outputs* command to plot the plot set.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “autoPlot” ‘boolean “nil“)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Auto Plot Outputs After Simulation*

artistPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “artistPlottingMode” ‘string “New Win”)
```

Variable Type	string
Default Value	Replace
Acceptable Values	Append / Replace / New Win / New SubWin
Window-Menu	*Cyclic is on the Artist window*

directPlotPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “directPlotPlottingMode” ‘string “New
Win”)
```

Variable Type	string
Default Value	Append
Acceptable Values	Append / Replace / New Win / New SubWin
Window-Menu	*Cyclic is on the DirectPlot Main form*

designName

If true, displays the design name in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “designName” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Design Name*

simulationDate

If true, displays the simulation run date in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “simulationDate” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Simulation Date*

temperature

If true, displays the temperature associated with the plotted results in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “temperature” ‘boolean “t”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – temperature*

variables

if true, displays the names and values of design variables in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “variables” ‘boolean “t”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Design Variables*

scalarOutputs

If true, displays simulation results that evaluate to scalar values in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “scalarOutputs” ‘boolean “t”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Scalar Outputs*

icons

This variable places icons in the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.plotting” “icons” ‘boolean “t”)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options –Allow Icons*

width

Specifies the width of the waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.plotting” “width” ‘int 300)`

Variable Type	int
Default Value	564
Acceptable Values	Between 200 to 1200
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Width*

height

Specifies the height of the waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.plotting” “height” ‘int 300)`

Variable Type	int
Default Value	428
Acceptable Values	Between 200 and 1000
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Height*

x

Enables you to set the horizontal position of the left side of the waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “x” ‘int 500)
```

Variable Type	int
Default Value	577
Acceptable Values	Between 0 and 1200.
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – X Location*

y

Enables you to set the vertical position of the top of the Waveform window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “y” ‘int 500)
```

Variable Type	int
Default Value	373
Acceptable Values	Between 0 and 1000.
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Y Location*

immediatePlot

This variable refers to the commands located in the *Direct Plot* menu. If true, the plot is drawn after each node is selected. If nil, none of the plots are drawn until all the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are plotted at the same time.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “immediatePlot” ‘boolean “t”)
```

Variable Type	boolean
Default Value	nil

Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Direct Plots Done After*

immediatePrint

This variable refers to the commands located in the Print menu. If true, the results are printed after each node is selected. If nil, none of the nodes is printed until all the nodes have been selected.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “immediatePrint” ‘boolean “t”)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment – Setting Plotting Options – Print After*

preSaveOceanScript

This procedure is executed before the ocean script is created, when the *Save Ocean Script* option is enabled. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MYfirstProc( session fp )
```

In this syntax, `session` is the Artist session and `fp` is the file pointer to the OCEAN script. You do not need to set these. Artist sets these for you. In this case, the value for the variable *`postSaveOceanScript`* will be `MyfirstProc`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.misc” “preSaveOceanScript” ‘string
“myPreSaveProc”)
```

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure

Window-Menu	--

postSaveOceanScript

This procedure is executed after the ocean script is created when the *Save Ocean Script* option is clicked. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MYlastProc( session fp )
```

In this syntax, `session` is the Artist session and `fp` is the file pointer to the OCEAN script. You do not need to set these; Artist sets these for you. In this case, the value for the variable *`postSaveOceanScript`* will be `MylastProc`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.misc” “postSaveOceanScript” ‘string
“myPostSaveProc”)
```

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure
Window-Menu	--

numberOfSavedRuns

Once set to value greater than `0`, Artist will retain the simulation run data for the last *numberOfSavedRuns* simulations. In case of analysis tools such as Parametric, Monte Carlo and Corners, a single run may include multiple simulations. At the end of a simulation run, Artist will save the current run data under:

```
<simulation_dir>/<cell_name>/<simulator_name>/<view_name> to a
numbered directory under <simulation_dir>/<cell_name>/
<simulator_name>.
```

The number used is one higher than the highest numbered directory name or 1 if none exist. If the maximum number of *Saved Runs* is reached, Artist will save the current run data, but delete the smallest numbered directory, thus keeping the number of Saved Runs equal to the value set in the variable.

Example:

numberOfSavedRuns is set to 2

Under `<simulation>/<ampTest>/<spectre>`

1. At the end of first simulation run

   ```
   1/   schematic/
   ```

2. At the end of second simulation run

   ```
   1/   2/ schematic/
   ```

3. At the end of third simulation run

   ```
   2/   3/ schematic/
   ```

In all the three cases above the schematic directory would have the current simulation results.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.misc” “numberOfSavedRuns” ‘int 2)
```

Variable Type	int
Default Value	0

browserCenterMode

To keep the most recently expanded node in the results browser in the center of the window, set this variable to true. If you do not want the most recently expanded node to move automatically to the center of the window, you can turn off the centering mode by setting this variable to `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.misc” “browserCenterMode” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

createCDFtermOrder

This variable is set to true by default. While generating a cellview from a cellview, the CDF `termOrder` property is automatically added to the cellview. You can set its value to `nil` if you want to disable the auto-creation of the simInfo `termOrder` property for cellview-to-cellview creation.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“auCore.misc” “createCDFtermOrder” ‘boolean “t”)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	--

updateCDFtermOrder

If this variable is set to true, it allows updating of the CDF termOrder after a symbol update. The default setting is nil. The CDF updating only affects the termOrder information. Before any updating of the CDF occurs, a dialog box appears and confirms if it is OK to update the base cell CDF termOrder data. The dialog box displays the simulators whose termOrder it will update, and the new termOrder that will be set for each simulator.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“auCore.misc” “updateCDFtermOrder” ‘boolean “t”)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

printNotation

It is used to specify how numbers are printed in the ADE environment. This applies only to *Results – Print* and *print/printvs* in the Calculator. Numbers are printed in the notation this variable is set to.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“auCore.userPref” “printNotation” ‘cyclic “scientific”)`

Variable Type	cyclic
Default Value	suffix
Acceptable Values	engineering, scientific, suffix
Window-Menu	--

displayMode

When `displayMode` is set to `composite`, the analog and the digital waveforms will be plotted together in the same strip. If set to `strip`, then the analog and digital waveforms are plotted in separate strips. `auto` sets the plot display mode to composite if the simulator is analog-only, and strip if it is a mixed-signal type.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “displayMode” ‘cyclic “strip”)
```

Variable Type	cyclic
Default Value	auto
Acceptable Values	auto, strip, composite
Window-Menu	--

stripModeType

If displayMode mentioned above has a value of `strip`, then setting *`stripModeType`* to `analogComposite` will display all the analog waveforms in one strip. Setting it to `analogStrip` displays each analog waveform in a separate strip. `auto` sets the strip mode type to `analogStrip` if the simulator is analog-only, and `analogComposite` if it is a mixed-signal type.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.plotting” “stripModeType” ‘cyclic “analogStrip”)
```

Variable Type	cyclic
Default Value	auto
Acceptable Values	auto, analogComposite, analogStrip
Window-Menu	--

saveDefaultsToOCEAN

When this variable is turned on, in addition to what is normally saved, it saves the following:

- All non-blank options
- All non-blank envOptions
- All enabled analyses and their options (as opposed to all analyses).

- ❑ All keep options (save all nets / currents... etc.)
- ❑ The model path(s)
- ❑ Temperature
- ❑ Simulator/analysis defaults to ocean scripts generated from artist.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.misc” “saveDefaultsToOCEAN” ‘boolean “t”)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

showWhatsNew

Set this variable to the release number for which you do not want to see the What’s New window. For example, set this variable to `5.0.0` if you do not want to see the What’s New window for 5.0.0.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv” “showWhatsNew” ‘string “5.1.2”)`

Variable Type	string
Default Value	"yes"
Acceptable Values	Any existing release number
Window-Menu	--

digits

Number of significant digits with which the contributors are printed.

Variable Type	int
Default Value	6
Acceptable Values	Any integer. The maximum limit is the limit of an integer.

Window-Menu	--

obsoleteWarnings

Number of warnings that are needed to be stored. By default, this variable is set to 1. Therefore, while netlisting, one error is shown at a time. If it is desired that more number of errors are shown then change this variable to a larger number.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.netlist” “obsoleteWarnings” ‘int 2)`

Variable Type	int
Default Value	1
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
Window-Menu	--

netlistAccess

This variable is used to specify the access permissions for the netlist. By default, this variable is set to `User`. Therefore, while netlisting, only the creator of the netlist will be able to run simulation on it. You can also set the values of the variable to `Group` and `All`. By setting the value to `Group`, all the users in the same group as the `User` will be able to run the netlist. If you set the value to All,anybody can run the netlist. To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal( asimenv.netlist   netlistAccess   string  User)
```

Variable Type	string
Default Value	User
Acceptable Values	User, Group, and All
Window-Menu	--

printCommentChar

This variable sets the preferred comment character for the printvs data. The # sign is the default comment character. To set the preferred comment character, set the variable, *`printCommentChar`* to the character.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.userPref” “printCommentChar” ‘string “+“)
```

Variable Type	string
Default Value	#
Acceptable Values	Any integer. The maximum limit is the limit of an integer
Window-Menu	--

loadCorners

If set to `t`, Corners classes are preloaded when Artist is started.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv” “loadCorners” ‘boolean “nil”)
```

Variable Type	boolen
Default Value	t
Acceptable Values	t, nil

toolList

Set this variable to the list of simulators integrated into ADE. If a new simulator is integrated, it has to be added to this list.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“auCore.toolFilter” “toolList” ‘string "spectre spectreS”)
```

Variable Type	string
Default Value	string "spectre spectreS cdsSpice auCdl auLvs hspiceS"

Acceptable Values	spectre spectreS cdsSpice auCdl auLvs hspiceS

ignoreSchModified

Set this variable to *t*, the schematic will not be modified. When this variable is set, you need not do a check and save after making changes to the *toolFilter* form.

To set this variable in the `.cdsenv` file or CIW, use the call:

```
envSetVal(“auCore.toolFilter” “ignoreSchModified” ‘boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

defaultTools

Set this variable to the list of simulators that need to be selected by default in the *toolFilter* form.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“auCore.toolFilter” “defaultTools” ‘string "spectre”)
```

Variable Type	string
Default Value	string "spectre spectreS auCdl auLvs"
Acceptable Values	spectre spectreS auCdl auLvs

oceanScriptFile

Set this variable to specify the default location for saving OCEAN script files.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.misc” “oceanScriptFile” ‘string "./myOcnScript")
```

Variable Type	string
Default Value	"./oceanScript.ocn"

printInlines

When this variable is set to `t,` data for all devices in a textual subcircuit will be printed. refer to chapter 10. Searching for devices in a textual subcircuit may take some time. If you want to disable this feature, set this variable to `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“asimenv.plotting” “printInlines” ‘boolean “nil“)`

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t/nil`
Window Menu	*Results – Print – DC Operating Points* *Results – Print – Transient Operating points* *Results – Print – Model Parameters*

awvResizeWindow

```
awvResizeWindow( w_windowId  l_bBox )
     => t | nil
```

Resizes a window to the size of a bounding box. This API works for both AWD and WaveScan waveform windows. The bounding box is specified as a list of 2 x:y points, that represent the lower left and upper right corners of the box.

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t/nil`
Example	`awvResizeWindow( window(4) list(391:288 1134:922) )`

paraplotUpdateSimulatorLog

When this variable is set to t, the simulator log appears in a new window, when a paramretric analysis is run.

Default Value	`nil`

Acceptable Values	`t/nil`
Variable Type	Boolean

labelDigits

This variable controls the number of digits to be displayed during annotation of operating points into the schematic window.

When the value is the default, `0`, the number of digits displayed is according to the AEL digit setup at that point in time. When the value is not `0`, the number of digits displayed is independent of the AEL digit setup. If the value is `1` or `2`, three digits are displayed in the schematic as that is the minimum number of digits supported by AEL. If the value is set to more than `2`, the annotated operating point appears shwoing the specified number of digits.

Tool Name	auCore.misc
Variable Type	int
Default Value	`0`
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
Window Menu	--

termFontSize

This variable controls the font size used to highlight selected terminals.

Tool Name	auCore
Partition	Selection
Variable Type	float
Default Value	`0.25`
Acceptable Values	Any number.
Window Menu	--

Calculator

mode

This variable sets the mode for creating expressions. For details, refer to the *About the Algebraic and RPN Modes* section, of Chapter 1 of the *Waveform Calculator User Guide*.

To set this variable in the `.cdsenv` file, add the line:

```
calculator mode cyclic “algebraic”
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“calculator” “mode” ‘cyclic “algebraic”)
```

Variable Type	cyclic
Default Value	RPN
Acceptable Values	{RPN, algebraic}
Window-Menu	*Calculator – Options*

uimode

This variable sets the mode of operation for the calculator. For details, refer to the *About Standard and RF Modes* section, of Chapter 1 of the *Waveform Calculator User Guide*.

To set this variable in the `.cdsenv` file, add the line:

```
calculator uimode cyclic “RF”
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“calculator” “uimode” ‘cyclic “RF”)
```

Variable Type	cyclic
Default Value	standard
Acceptable Values	{standard, RF}
Window-Menu	*Calculator*

eval

This field is set to evaluate the contents of a calculator buffer automatically. This is available only for the RPN mode. For details, refer to the *Evaluating the Buffer* section, of Chapter 3 of the *Waveform Calculator User Guide*.

To set this variable in the `.cdsenv` file, add the line:

```
calculator eval boolean t
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("calculator" "eval" ‘boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	*Calculator*

dstack

This field is set to display the contents of the stack. This is available only for the RPN mode. For details, refer to the *About the Stack* section, of Chapter 3 of the *Waveform Calculator User Guide*.

To set this variable in the `.cdsenv` file, add the line:

```
calculator dstack ‘boolean t
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("calculator" "dstack" boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	*Calculator*

Distributed Processing

clockSync

This variable can be used to check the clock synchronization between different machines in an LBS cluster.

If the `clockSync` is set, the cdsDPSetupChk script checks if the date and time setup of all the submitting hosts in the queue are synchronized with the date and time setup of the cluster master. If some machines are found to unsynchronized, a warning message is printed.

To set this variable in the `.cdsenv` file, add the line:

```
asimenv.distributed clockSync boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	none

autoJobSubmit

If this variable is set to a non-nil value, the *Job Setup* form is not displayed at job submit time.

To set this variable in the `.cdsenv` file, add the line:

```
asimenv.distributed autoJobSubmit boolean nil
```

To set this variable in the the `.cdsinit` file or CIW, use the call

```
envSetVal(“asimenv.distributed” “autoJobSubmit” ‘boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	*Setup – Simulator/Directory/Host*

showMessages

If this variable is set to a non-nil value, a message is displayed in the CIW or OCEAN terminal on the completion of a job.

To set this variable in the `.cdsenv` file, add the line:

```
asimenv.distributed showMessages boolean nil
```

To set this variable in the the `.cdsinit` file or CIW, use the call:

```
envSetVal(“asimenv.distributed” “showMessages” ‘boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	none

queueName

The variable sets the default queue name. If unspecified, the system default is used. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsenv` file, add the line:

```
asimenv.distributed queueName string "myqueue”
```

To set this variable in the the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "queueName" ‘string "myqueue")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	*Job Submit Form*

hostName

This variable sets the default host name. If unspecified, the host is selected automatically. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsenv` file, add the line:
`asimenv.distributed hostName string "host”`

To set this variable in the the `.cdsinit` file or CIW, use the call:
`envSetVal("asimenv.distributed" "hostName" ‘string "host")`

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	*Job Submit Form*

startTime

This variable sets the default start time for a job (in 24hour format). If unspecified, the job executes immediately. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or ciw, use the call:
`envSetVal("asimenv.distributed" "startTime" ‘string "23:11")`

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
Window-Menu	*Job Submit Form*

startDay

This variable sets the default start day for a job. If the start day is set as `today`, then the job will always run on the same day it is submitted.

To set this variable in the `.cdsinit` file or CIW, use the call
`envSetVal("asimenv.distributed" "startDay" ’cyclic "Wednesday")`

Variable Type	cyclic
Default Value	`today`
Acceptable Values	`{today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}`

Window-Menu	*Job Submit Form*

expTime

This variable sets the default expiration time for a job (in 24 hour format). If unspecified, the expiration time is based on the value of the *`timeLimit`* variable. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "expTime" ’string "00:43")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
Window-Menu	*Job Submit Form*

externalServer

If this variable is set to a non-nil value, the job server is started remotely.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "externalServer" ’boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

expDay

This variable sets the default expiration day for a job. If the expiration day is set as `today`, then the job will always run on the same day it is submitted. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "expDay" ’cyclic "Friday")
```

Variable Type	cyclic
Default Value	`today`
Acceptable Values	`{today, Sunday, Monday, Tuesday Wednesday, Thursday, Friday, Saturday}`
Window-Menu	*Job Submit Form*

timeLimit

This variable sets the default time limit for a job. If the time limit is set to `none`, then no time limit is imposed. If unspecified, then expiration time is based on value of *`expTime`* and *`expDay`* variables.For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "timeLimit" ’cyclic "5 minutes")
```

Variable Type	cyclic
Default Value	none
Acceptable Values	`{unspecified, none, 5 minutes, 15 minutes, 30 minutes, 1 hour, 3 hours, 6 hours, 12 hours, 1 day, 2 days, 3 days, 5 days, 10 days}`
Window-Menu	*Job Submit Form*

emailNotify

If this variable is set to a non-nil value, an e-mail notification is provided, following job termination. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "emailNotify" ’boolean nil )
```

Variable Type	boolean

Default Value	t
Acceptable Values	{t, nil}
Window-Menu	*Job Submit Form*

mailTo

This variable sets the default list of users who will receive job termination notification e-mail. If unspecified and if *`emailNotify`* is `t`, then the default value is the user's ID. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "mailTo" ’string
"userId123@cadence.com")
```

Variable Type	string
Default Value	""
Acceptable Values	Any Valid Id String Value.
Window-Menu	*Job Submit Form*

logsInEmail

If this variable is set to a non-nil value, `stdout` and `stderr` logs will be included in the termination E-mail.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "logsInEmail" ’boolean t)
```

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
Window-Menu	none

stateFile

This variable sets the filename containing the job server's state.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "stateFile" ’string "myStateFile")
```

Variable Type	string
Default Value	`~/.adpState`
Acceptable Values	Any String Value
Window-Menu	none

daysBeforeExpire

Specifies the number of days after which terminated jobs will be deleted from the job server.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "daysBeforeExpire" ’int 6)
```

Variable Type	int
Default Value	3
Acceptable Values	Any String Value
Window-Menu	none

block

If this variable is set to a non-nil value, the process is blocked until the job has completed.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "block" ’boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

copyMode

If this variable is set to a non-nil value, the input data for the job is copied to `/tmp` on the execution host, the job is run there locally (without network read/write), and the output data is copied back to the submission host.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "copyMode" ’boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

copyModeDir

Specifies the directory relative to the execution host, that will be used for setting up the working directory of a copy mode job.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "copyModeDir" ’string "dirname")
```

Variable Type	string
Default Value	`/tmp`
Acceptable Values	Any string value
Window-Menu	none

loginShell

Specifies the login shell for the job. If it is specified as `none` then the users local environment is copied over to the execution host and used as the jobs environment.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "loginShell" ’cyclic "csh")
```

Variable Type	cyclic
Default Value	`none`

Acceptable Values	{`none, csh, ksh, sh`}
Window-Menu	none

numOfTasks

Specifies the default number of tasks a job should be broken into. This is used by the Monte Carlo tool. If zero, then the number of tasks is based on queue and/or host settings.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "numOfTasks" ’int 5)
```

Variable Type	int
Default Value	0
Acceptable Values	Any Integer Value
Window-Menu	none

jobArgsInOceanScript

Indicates job arguments that should be added to run commands when an OCEAN script is generated.

```
envSetVal("asimenv.distributed" "jobArgsInOceanScript" ’boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

puttogetherqueue

Specifies the queue to be used for the Put Together Job.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "puttogetherqueue" ’string
"queuename")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	none

copyNetlist

Specifies whether the netlist directory needs to be copied from the execution host to the submission host. This may be required if during simulation, some files are generated under the netlist directory.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "copyNetlist" ’boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

mailAllLogs

Sends out a mail after completion of all the tasks and each individual task (when set to `t).`

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.distributed" "mailAllLogs" ’boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}

Spectre

save

This variable selects signals to be saved.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectre.outputs” “save” ‘string “all”)
```

Variable Type	string
Default Value	allpub
Acceptable Values	none,selected,lvlpub,allpub,all
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Select signals to output (save)*

outputParamInfo

This variable sets/reset the *Save output Parameters Info* option.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “outputParamInfo” ‘boolean nil )
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Save output parameters info*

modelParamInfo

This variable sets/resets the *Save model parameters Info* option.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “modelParamInfo” ‘boolean nil)
```

Variable Type	boolean

Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment* – <u>Save Options</u> – *Save model parameters info*

pwr

This variable is used to select the power signals to output.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.outputs” “pwr” ‘string “all”)`

Variable Type	string
Default Value	""
Acceptable Values	none, total, devices, subckts, all
Window-Menu	*Virtuoso Analog Design Environment* – <u>Save Options</u> – *Select power signals to output (pwr*)

useprobes

This variable is used to set the *Select AC terminal currents (useprobes)* option.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.outputs” “useprobes” ‘string “no”)`

Variable Type	string
Default Value	""
Acceptable Values	yes, no
Window-Menu	*Virtuoso Analog Design Environment* – <u>Save Options</u> – *Select AC terminal currents (useprobes)*

subcktprobelvl

This variable is used to control the calculation of terminal currents for subcircuits. Current probes are added to the terminals of each subcircuit (up to subcktprobelvl deep).

Variable Type	string
Default Value	""
Acceptable Values	--
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Set subcircuit probe level (subcktprobelvl)*

nestlvl

This variable is used to save groups of signals as results and when signals are saved in subcircuits. The nestlvl parameter also specifies how many levels deep into the subcircuit hierarchy you want to save signals.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “nestlvl” ‘string “2”)
```

Variable Type	string
Default Value	""
Acceptable Values	--
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Set level of subcircuit to output (nestlvl)*

elementinfo

This variable specifies if input parameters for instances of all components are saved.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “elementInfo” ‘boolean nil )
```

Variable Type	boolean
Default Value	t

Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Save element info*

saveahdlvars

If you want 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
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “saveahdlvars” ‘string “all”)
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Save AHDL variables (saveahdlvars)*

currents

The currents parameter of the options statement computes and saves terminal currents. Use it to create settings for currents that apply to all terminals in the netlist.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.outputs” “currents” ‘string “nonlinear”)
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all, nonlinear
Window-Menu	*Virtuoso Analog Design Environment* – Save Options – *Select device currents (currents)*

switchViewList

This variable is used to define the *Switch View List* field. This is a list of the views that the software switches into when searching for design variables.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “switchViewList” ‘string “schematic
spectre”)
```

Variable Type	string
Default Value	"spectre cmos_sch cmos.sch schematic veriloga ahdl"
Acceptable Values	view names, separated by spaces.
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *Switch View List.*

stopViewList

This variable is used to define the *Stop View List* option. This is a list of views that identify the stopping view to be netlisted.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “stopViewList” ‘string “spectre
verilog”)
```

Variable Type	string
Default Value	"spectre"
Acceptable Values	view names separated with spaces.
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *Stop View List*

autoDisplay

This variable is used to set/reset the *Automatic output log* option. When on, the output log opens and displays simulator messages as they are generated.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “autoDisplay” ‘boolean nil)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Virtuoso Analog Design Environment –* Environment Options *– Automatic output log*

spp

This variable is used to set/reset the *Use SPICE Netlist Reader(spp)* option.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “spp” ‘string “Y”)`

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Virtuoso Analog Design Environment –* Environment Options *– Use SPICE Netlist Reader(spp)*

stimulusFile

This variable is used to set the path for stimulus file.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “stimulusFile” ‘string “./file”)`

Variable Type	string
Default Value	""
Acceptable Values	unix path
Window-Menu	*Virtuoso Analog Design Environment –* Simulation Files Setup *– Stimulus File*

includePath

Use this field for relative filenames.The simulator resolves a relative filename by first searching 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.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “includePath” ‘string “./dir1 ../dir2”)`

Variable Type	string
Default Value	""
Acceptable Values	unix directories, separated with spaces.
Window-Menu	*Virtuoso Analog Analog Design Environment* – Simulation Files Setup – *include Path*

modelFiles

Use this field for adding the default model files.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “modelFiles” ‘string “./models/
model1.scs ./models/model2.scs”)
```

To disable modelFiles, use a # sign as shown in the example below:

```
envSetVal("spectre.envOpts" "modelFiles" ’string "#;./models/
model1.scs ./models/model2.scs")
```

Here, the `model1.scs` file will be disabled.

Variable Type	string
Default Value	""
Acceptable Values	list of paths to model files.
Window-Menu	*Virtuoso Analog Design Environment* – Model Library Setup

analysisOrder

Determines the order in which the analyses would be run by the simulator.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “analysisOrder” ‘string “tran ac dc”)
```

Variable Type	string
Default Value	""
Acceptable Values	Names of analysis in the order desired
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *Analysis Order*

paramRangeCheckFile

Enter the path to a file containing the correct ranges for component parameters. If this path is present, the simulator checks the values of all component parameters in the circuit against the parameter range-checking file and prints out a warning if any parameter value is out of range.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “paramRangeCheckFile” ‘string “./
param.file”)
```

Variable Type	string
Default Value	""
Acceptable Values	path of the file
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *Parameter Range Checking File*

printComments

When off, comments are not printed. When on, extra comments are placed in the netlist regarding component location and name.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “printComments” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

Window-Menu	*Virtuoso Analog Design Environment –* Environment Options *– Print Comments*

definitionFiles

Type the full UNIX path or the name of one or more files. A definitions file contains function definitions and definitions of parameters that are not displayed in the Design Variables section of the simulation window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “definitionFiles” ‘string “./file”)
```

Variable Type	string
Default Value	""
Acceptable Values	unix path or name of one or more files.
Window-Menu	*Virtuoso Analog Design Environment –* Simulation Files Setup *– Definition Files*

enableArclength

When this variable is set to true, the homotopy convergence option is visible, else this is not visible.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “enableArclength” ‘boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

useAltergroup

When using models that do not work with altergroups, turn the *`useAltergroup`* variable to off. When using altergroups, keep this on.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “useAltergroup” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	--

netlistBBox

This variable is used to control the size of the netlist window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “netlistBBox” ‘string “10 10 525 800”)
```

Variable Type	string
Default Value	"0 0 515 700"
Acceptable Values	window coordinates.
Window-Menu	--

autoDisplayBBox

This variable is used to control the size of the `spectre.out` window.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “autoDisplayBBox” ‘string “10 10 525
800”)
```

Variable Type	string
Default Value	"0 0 515 700"
Acceptable Values	window coordinates
Window-Menu	--

includeStyle

Use the env option includeStyle to have one model per file. This option works with model name passing. When set to t, for stopping cells whose model name is being passed hierarchically, the passed model name specified at a higher level is added to the required model files. Or, a default value specified for a passed parameter resulting in the final specification of a model for an instance is added to the required model files.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“spectre.envOpts” “includeStyle” ‘boolean “t” )
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

simExecName

Change this variable with caution. This variable can be set to point to the path of the desired spectre executable. It is advisable not to change this variable unless very much required.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“spectre.envOpts” “simExecName” ‘string “/home/spectre/
bin”)
```

Variable Type	string
Default Value	spectre
Acceptable Values	path of the spectre executable
Window-Menu	--

savestate

The Y option runs spectre with the `+ss` option, which turns on the savestate capability. The default option, N, runs spectre with the `-ss` option, which turns off the savestate capability.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal(“spectre.envOpts” “savestate” ‘string “Y”)
```

Note: This variable does not work as intended with advanced analysis tools, such as MonteCarlo, Optimizer, Corners, and parametric analysis. Ensure that it is set to N before using these tools.

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *savestate (ss)*

recover

Y runs spectre with the `+rec` option, which restarts the simulation from the savestate file, if it exists. N runs spectre with the `-rec` option, which does not restart the simulation, even if a savestate file exists.

Note: The `checkpoint` variable has been replaced by the `savestate` and `recover` variables in the UI. However, it can still be used from the command-line or from `.cdsinit`.

To set the `recover` variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “recover” ‘string “Y”)
```

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Virtuoso Analog Design Environment* – Environment Options – *recover (rec)*

firstRun

Set this variable to false if you do not want the *Welcome to Spectre* window to pop up when you run a simulation.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “firstRun” ‘boolean “nil”)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

Window-Menu	--

simOutputFormat

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, Spectre generates an error. The `psfbinf` format is a single-precision format that uses only half the disk space that `psfbin` uses. Setting the value to `sst2` causes spectre to generate the output for transient analyses in the SST2 format. Non-transient data cannot be generated in the SST2 format and is be generated in PSF format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectre.envOpts" "simOutputFormat" ’string "psfbinf")
```

To set it in the `.cdsenv` file, add:

```
spectre.envOpts simOutputFormat string "psfbinf"
```

Variable Type	string
Default Value	psfbin
Acceptable Values	psfbin, psfbinf, sst2
Window-Menu	--

controlMode

Used to run Spectre in batch or interactive modes depending on the value of the variable.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“spectre.envOpts” “controlMode” ‘string “batch”)
```

Variable Type	string
Default Value	interactive
Acceptable Values	interactive, batch
Window-Menu	--

Note:

- ❑ All the Spectre simulator options are documented under *spectre -h options* and there is one -to-one correspondence between `spectre.<optName>` and `<optName>`

- ❑ All the analysis options are documented under *spectre -h <analysisName>*.
- ❑ The following variables are deprecated:
  spectre.init *remoteDir* string "" t
  spectre.init *hostMode* string "local" t
  spectre.init *host* string "" t
  spectre.init *settableResultsDirectory* boolean t t
  spectre.init *processPriority* int 0 t 0 20

licQueueTimeOut

This variable enables queuing for a license. You have to set the time required to wait for a license (in seconds). The option *'+lqtimeout <value>'* is always passed to *Spectre*, unless set to 0. It is passed with a value of 900 or any other value that is specified.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “licQueueTimeOut” ‘string “1000”)`

Variable Type	string
Default Value	"900"(15 min)
Window-Menu	--

licQueueSleep

This variable specifies the sleep time between two attempts to check out a license when queuing. Setting the value to a positive number overrides the default sleep time of 30 seconds.The option *'+lqsleep <value>'* is not passed to *Spectre* unless given a value. If it is not passed to *Spectre*, *Spectre* uses a default value of 30.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal(“spectre.envOpts” “licQueueSleep” ‘string “20”)`

Variable Type	string
Default Value	""
Window-Menu	--

ignorePortOrderMismatch

When set to t, the netlister will not generate any warning when a portOrder mismatch occurs.

When set to the default value nil, any portOrder mismatch will result in following warning:

```
"Mismatch was found between the terminals in the cellView and  those
on the pin order property on the schematic, or on the termOrder
property on the CDF. The internal default order is being used.
Please eliminate the mismatch if any of the above properties must be
used for netlisting."
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil
Window-Menu	--

SpectreVerilog and SpectreSVerilog

simOutputFormat

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, Spectre generates an error. The `psfbinf` format is a single-precision format that uses only half the disk space that `psfbin` uses. Setting the value to `sst2` causes spectre to generate the output for transient analyses in the SST2 format. Non-transient data cannot be generated in the SST2 format and is be generated in PSF format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectreVerilog.envOpts" "simOutputFormat" ’string
"psfbinf")
```

To set it in the `.cdsenv` file, add:

```
spectreVerilog.envOpts simOutputFormat string "psfbinf"
```

For spectreSVerilog, replace `spectreVerilog` in these commands with `spectreSVerilog`.

Variable Type	string
Default Value	psfbin
Acceptable Values	psfbin, psfbinf, sst2
Window-Menu	--

logicOutputFormat

Use this variable to generate digital data outputs for mixed-signal simulations in WSF format or SST2 format. ADE does not support the post-processing flow for the WSF format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectreVerilog.envOpts" "logicOutputFormat" ’string
"WSF")
```

To set it in the `.cdsenv` file, add:

```
spectreVerilog.envOpts logicOutputFormat string "WSF"
```

For spectreSVerilog, replace `spectreVerilog` in these commands with `spectreSVerilog`.

Variable Type	string
Default Value	SST2

Acceptable Values	SST2, WSF
Window-Menu	--

HspiceD

setTopLevelAsSubckt

This variable prevents the top-level schematic from being netlisted as a top-level schematic. If it is set to `t`, the top-level schematic is netlisted as a subcircuit.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal(“hspiceD.envOpts” “setTopLevelAsSubckt” ‘boolean “nil”
“t”)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

AMS and UltraSim

connectRulesList

Sets the default set of connect rules. To set this variable in the .cdsinit, use the call:

```
envSetVal(“ams.envOpts” “connectRulesList” ‘string
“connectLib;ConnRules_18V_full;connect
connectLib;mixedsignal;connect”)
```

This variable will set two connect rules connectLib/ConnRules_18V_full and connectLib/misedsignal as the default connect rules in the Connect Rules form.

To set a single connect rule, use the following call:

```
envSetVal(“ams.envOpts” “connectRulesList” ‘string
“connectLib;ConnRules_5V_full;connect”)
```

Variable Type	string
Default Value	`“connectLib;ConnRules_5V_full;connect”`

useEffectiveCDF

When this variable is set to t, AMS in ADE uses effective CDF while netlisting. If set to `nil`, the base cell CDF will be used. To set this variable in the .cdsenv file or ciw, use the call:

```
envSetVal(“ams.netlisterOpts” “useEffectiveCDF” ‘boolean nil)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t/nil`

useOtherOutputFormat

When this variable is set to t, PSF, SST2, FSDB, and WDF is displayed for Output Format field of Simulator Options form. If set to nil, PSF and SST2 is displayed as Output Format.To set this variable in the .cdsinit file , use the call:

```
envSetVal(“UltraSim.opts” “useOtherOutputFormat” ‘boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t/nil`

For a comprehensive list of the AMS and UltraSim variables, refer to *Appendix A* of the *Virtuoso AMS Environment User Guide*.

B

Waveform Tools in ADE

The Analog Design Environment (ADE) supports both WaveScan and Analog Waveform Display (AWD).

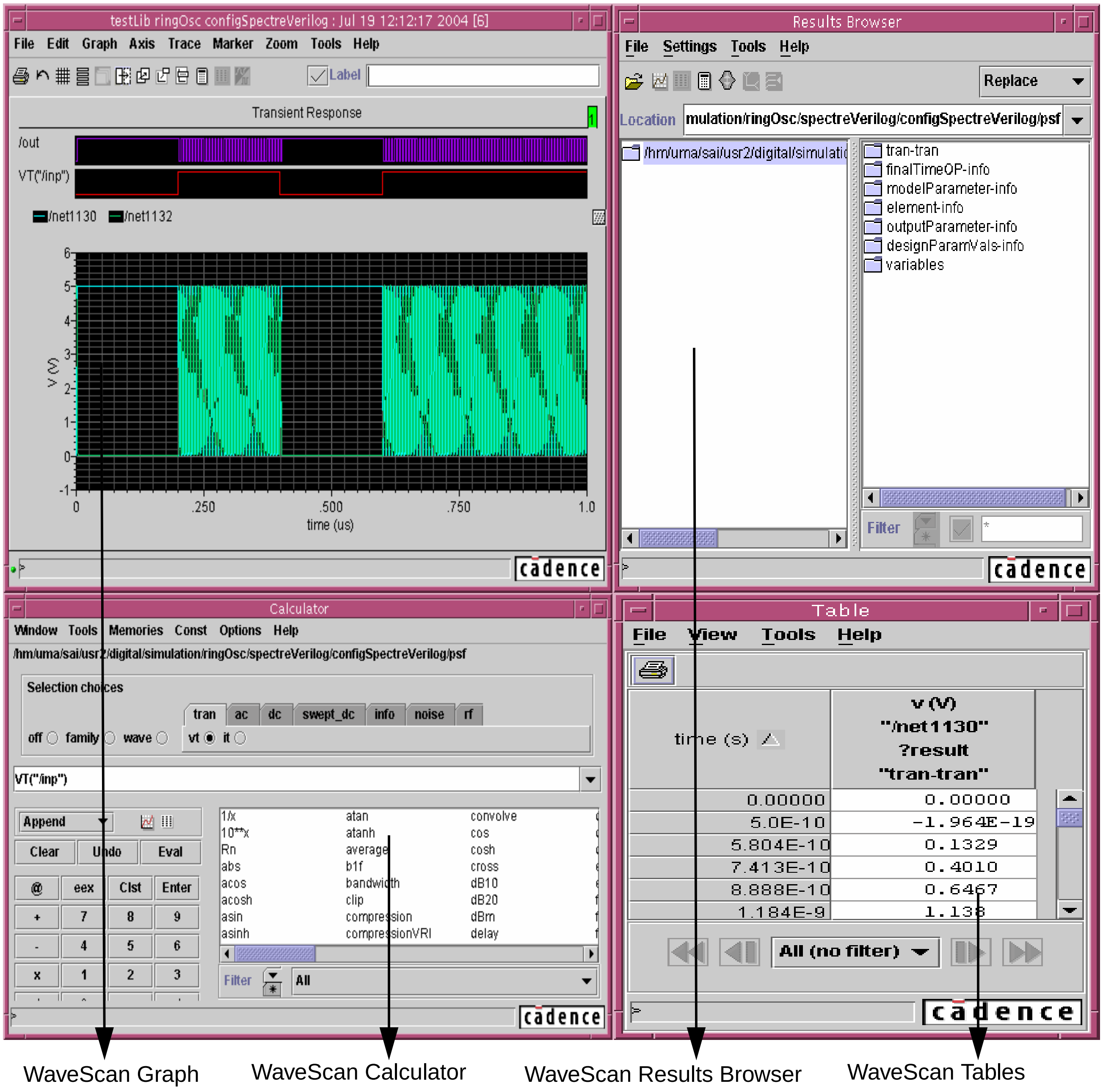
WaveScan is an advanced analog/mixed-signal waveform display and post-processing tool. It is now the default waveform tool. However, you can seamlessly and dynamically switch between the two tools at any point in the flow. Several easy ways (for example, the *ADE – Session – Options* menu option) are supported to make the waveform tool selection. For details see, Selecting the Waveform Tool on page 413.

This chapter highlights differences in behavior of the WaveScan toolset with respect to AWD, differences in public SKILL functions and relevant information required for migration. The chapter introduces you to the use model differences between AWD and WaveScan in the Analog Design Environment and serves as a quick reference if you are a new WaveScan user.

To learn about *WaveScan* and *AWD* in detail, it is recommended that you read the *WaveScan User Guide* and the *Analog Waveform User Guide*, respectively.

Post Processing Tools

The post-processing tools available with WaveScan as the chosen waveform viewer are *WaveScan Graph*, *WaveScan Calculator*, *WaveScan Results Browser* and *WaveScan Tables*.



Location: mulation/ringOsc/spectreVerilog/configSpectreVerilog/psf

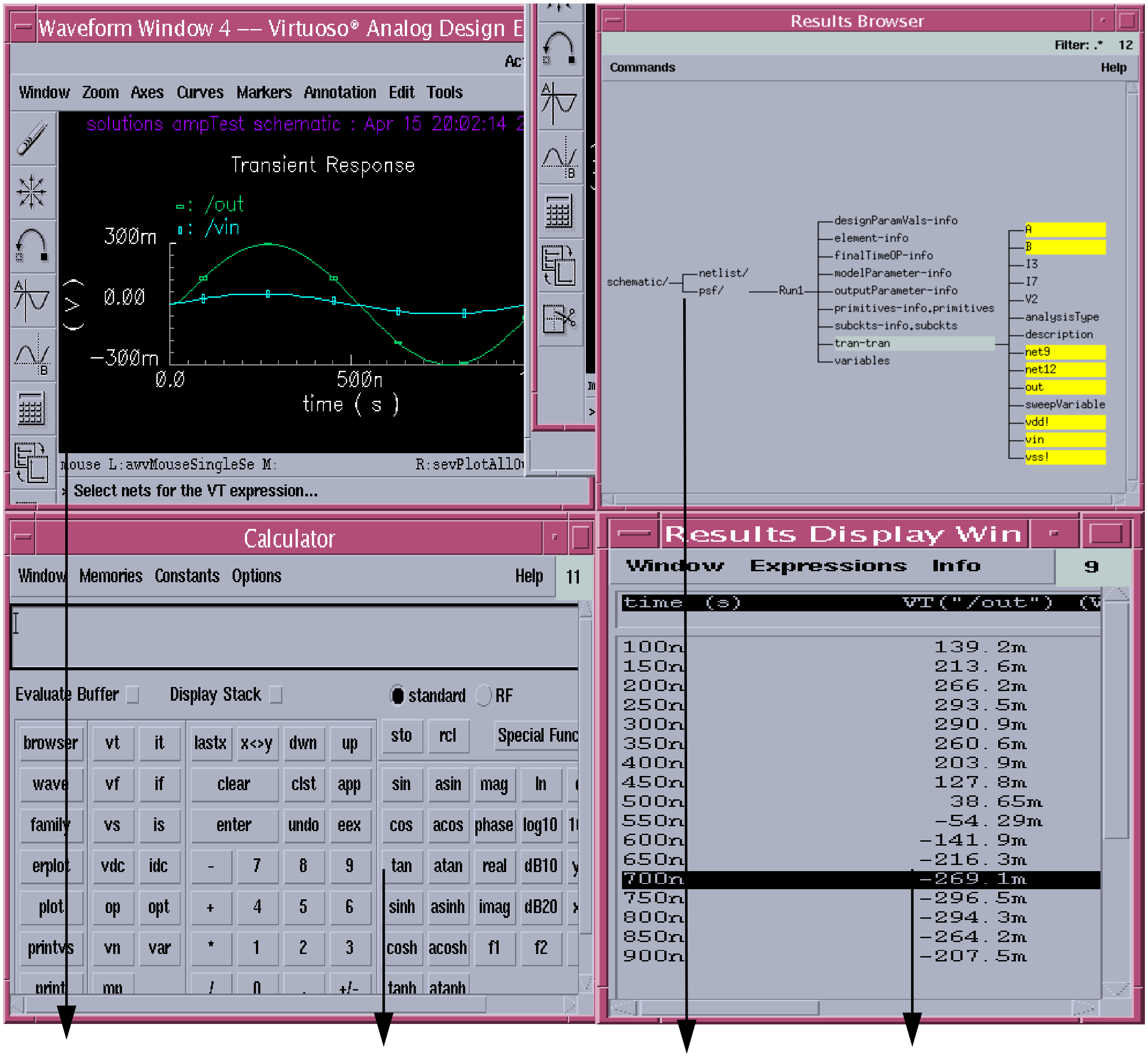
/hm/uma/sai/usr2/digital/simulatic

/hm/uma/sai/usr2/digital/simulation/ringOsc/spectreVerilog/configSpectreVerilog/psf

VT("/inp")

time (s)	v (V) "/net1130" ?result "tran-tran"
0.00000	0.00000
5.0E-10	-1.964E-19
5.804E-10	0.1329
7.413E-10	0.4010
8.888E-10	0.6467
1.184E-9	1.138

The post-processing tools available with AWD as the chosen waveform viewer are *AWD Graph*, *AWD Calculator*, *AWD Results Browser* and the *ADE Results Display (Print)*.



```
mouse L:awvMouseSingleSe M:                R:sevPlotAll0
Select nets for the VT expression...
```

```
time (s)            VT("/out") (V
100n                139.2m
150n                213.6m
200n                266.2m
250n                293.5m
300n                290.9m
350n                260.6m
400n                203.9m
450n                127.8m
500n                 38.65m
550n                -54.29m
600n               -141.9m
650n               -216.3m
700n               -269.1m
750n               -296.5m
800n               -294.3m
850n               -264.2m
900n               -207.5m
```

AWD Graph

AWD Calculator

AWD Results Browser

ADE Results
Display (Print)

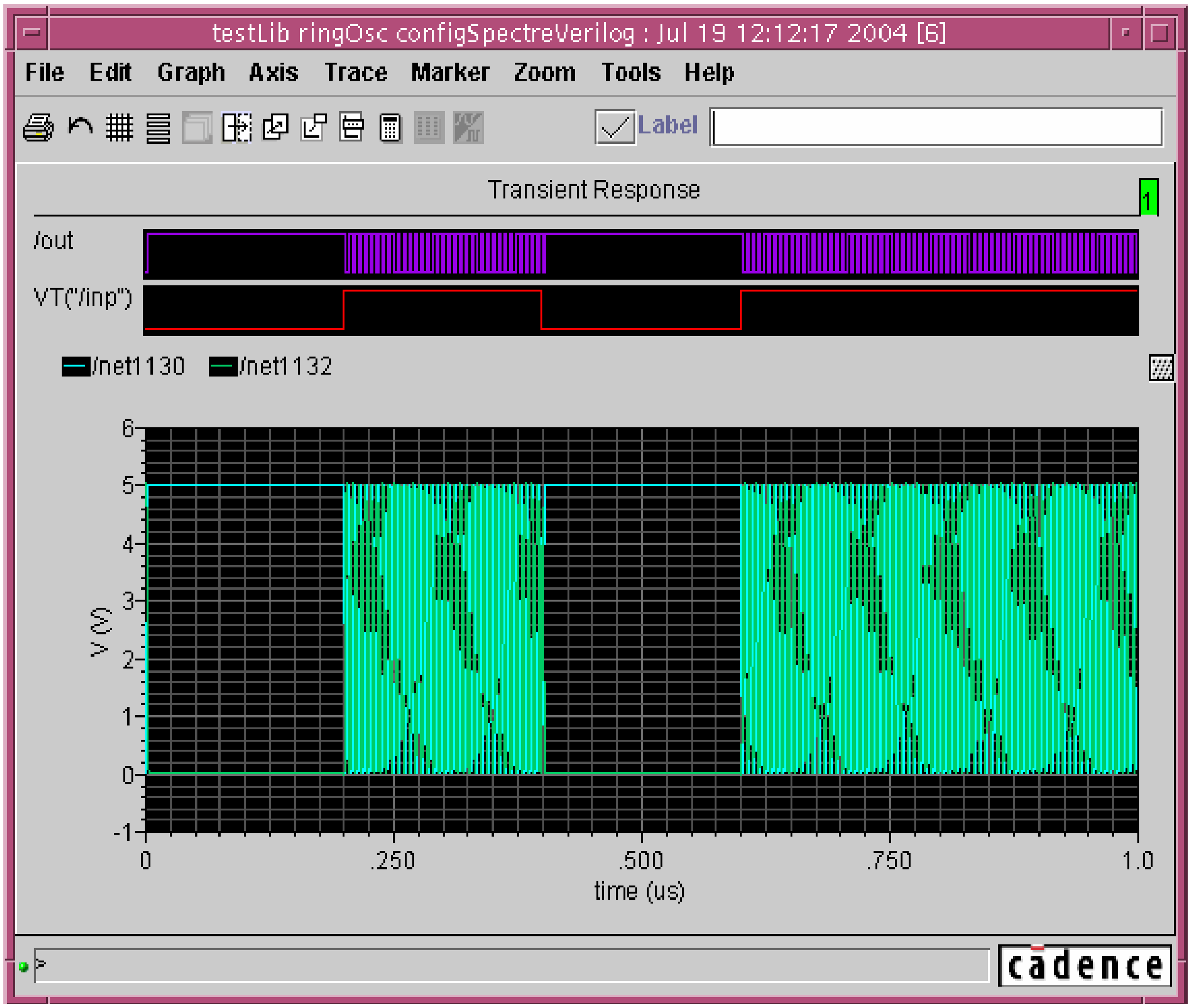
Note: When you switch from one waveform tool to another, windows of the earlier waveform tool continue to be available but all subsequent operations will be performed on the window(s) associated with the waveform viewer that is selected.

Differences in Behavior

The WaveScan tool solves a number of functional and usability issues with AWD. The following sections compare WaveScan and AWD postprocessing features and highlight some of the key differences in behavior between the two tools.

Graphs

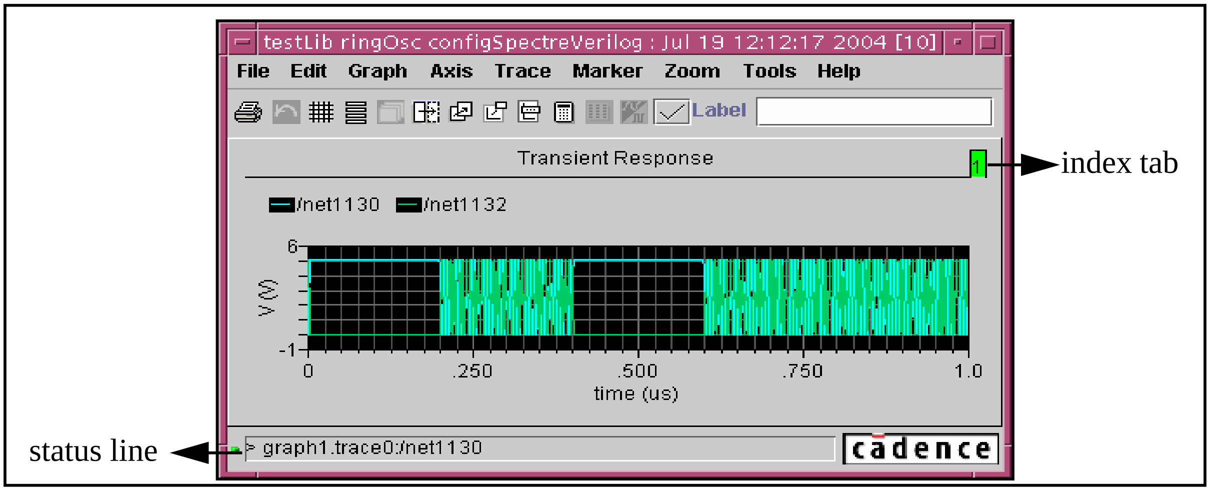
WaveScan graphs are designed to improve the visual presentation of data. Customization features are reorganized and expanded with an emphasis on simplicity and user productivity. Some of the key features are described in the sections that follow. For a more thorough description of WaveScan Graph features, see the *Working with Graphs* chapter of the *WaveScan User Guide*.



Object Oriented Editing

WaveScan graph windows are composed of objects: graphs (interchangeable with subwindows), traces, axes, markers and labels. To make any change, select the object and then choose an action.

- **Selection**
  Click on the index tab, anywhere in the graph canvas, or the grey area around it to select a graph. The status line located at the bottom of the graph shows the name of the object over which the mouse pointer is currently located.



- **Hide/Reveal**
  Click *Edit – Hide* to hide graphs, traces, axes, markers and labels.

- **Swap**
  Click *Edit – Swap* to exchange positions for two graphs in a multi-graph window or two y-axes in a multi-axis graph.

- **Delete**
  Click *Edit – Delete* to delete graphs, traces, markers and labels.

- **Move**
  Move markers and labels to new locations on a graph with drag-and-drop. For information on moving trace objects, see the next section Moving Traces.

- **Edit Attributes**
  Double-click on any object to see the corresponding *Attributes* window.

All the above features are supported in AWD, except for the swap feature.

Moving Traces

Traces cannot be moved from graph to graph in AWD. However, WaveScan supports both drag-and-drop and cut, copy, and paste operations for trace objects. With drag-and-drop, traces can be moved from one strip to another strip in a strip chart, from graph to graph within a graph window and from graph to graph between two graph windows. Use the *Trace – Cut*, *Trace – Copy* and *Trace – Paste* menu options for cut, copy and paste of selected traces. You can also use the standard `Ctrl-x`, `Ctrl-c` and `Ctrl-v` keys. Copy and move functions are also conveniently combined with subwindow and window creation in the *Trace – New Graph* submenu. For more information, see the *Working with Graphs* chapter (*Working with Traces* section*)* of the *WaveScan User Guide.*

Tracking Cursors

Cursors are transient data tracking aides. WaveScan offers four tracking cursor options:

- **Trace Cursor**
  To enable the trace cursor, choose *Trace – Trace Cursor*. You can also use the currently defined cursor bindkey, c being the default, to toggle the trace cursor on and off. The trace cursor positions are displayed at the lower left side. To move the trace cursor, the mouse pointer must be close to the trace. This is equivalent to the tracking cursor in AWD.

- **Vertical Cursor**
  To enable the vertical cursor, choose *Trace – Vert Cursor*. You can also use the currently defined cursor bindkey, v being the default, to toggle the vertical cursor on and off. Drag a vertical cursor by the red handle to move it to a new position. The first intersection point with each trace is shown next to the trace legend.

- **Horizontal Cursor**
  To enable the horizontal cursor, choose *Trace – Horiz Cursor*. You can also use the currently defined cursor bindkey, h being the default, to toggle the horizontal cursor on and off. Drag a horizontal cursor by the red handle to move it to a new position. The first horizontal intersection point with each trace is shown next to the trace legend.

- **Delta Cursor**
  To enable the delta cursor, choose *Trace – Delta Cursor*. You can also use the currently defined cursor bindkey, d being the default, to toggle the delta cursor on and off. Select the left or right handle to move the cursors. The individual cursor intersections are displayed at the lower left side. The delta x and delta y as well as the slope are displayed at the lower right side. In AWD, similar functionality is provided by the crosshair marker.

To learn about each of these cursors and how to work with them, refer to the *Working with Graphs* chapter (*Working with Cursors* section) of the *WaveScan User Guide*.

Adding Markers and Labels

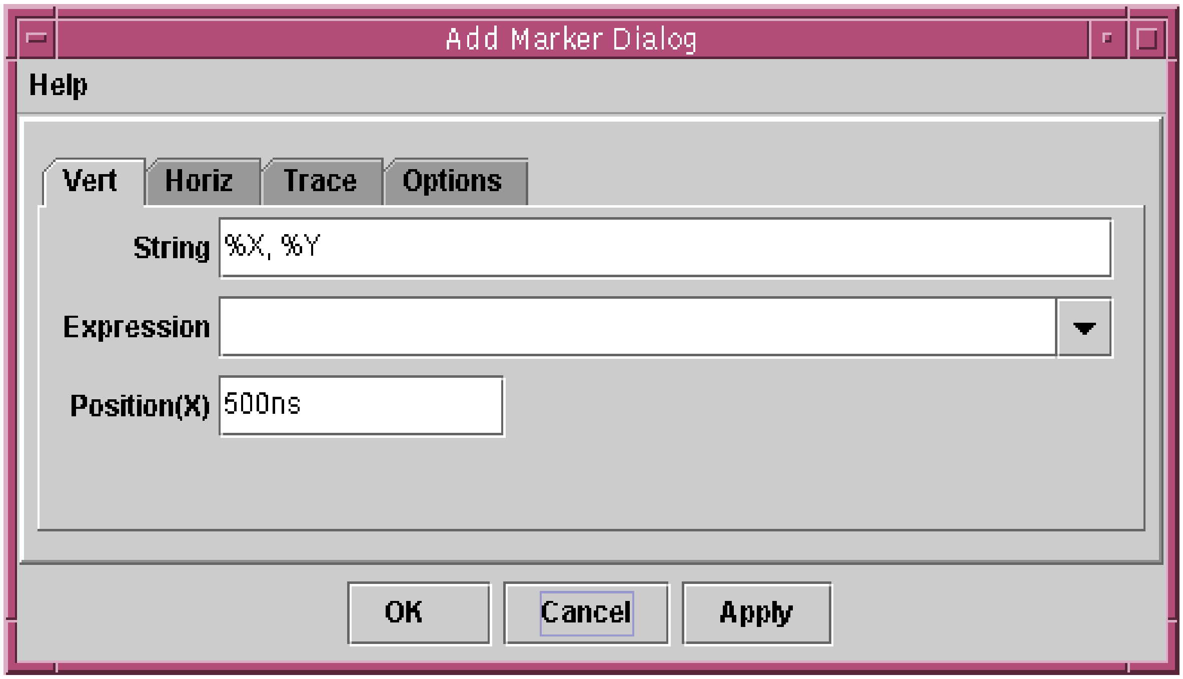
WaveScan offers four basic marker types: trace, vertical, horizontal and delta markers. You can add these by using the Add Marker form or bindkeys.

Using the Add Marker Form

You can add Vertical, Horizontal and Trace markers by specifying coordinates in the Add Marker form.

1. Choose *Marker – Add* or click the *Add Marker* ( ) icon.

   The Add Marker form opens.



String: %X, %Y

Expression:

Position(X): 500ns

2. Click the *Vert*, *Horiz*, or *Trace* tab depending on the type of marker you want to add.
3. In the *String* field, enter the label text for the marker. You may use one of the format strings mentioned below:

Format Strings	Description
%X	X coordinate
%Y	Y coordinate
%W	Δx

%H	∆y
%S	Slope (∆x/∆y)
%N	Name of the trace
%E	Expression

These format strings are evaluated and inserted into the string when you place or edit a marker. This allows labels to reference the marker coordinates, trace slope, trace name, and so on or the result of a scalar expression. If you do not enter text in the *String* field, the X and Y coordinates of the marker are displayed.

4. The *Expression* field drop-down displays all defined memories (in SKILL mode) or variables. You can also specify an expression by typing it in. The first option in this list is *Calculator Buffer*, which lets you retrieve and use the current contents of the Calculator buffer.
5. In the *Position* field, specify the coordinates at which you want to place the marker. For vertical markers, specify the X-axis coordinate. For horizontal markers, specify the Y-axis coordinate. For trace makers, specify a coordinate for either the X-axis or the Y-axis.
6. Click the *Options* tab to specify the label details for the marker. For example, in the *Signif Digits* field, specify the number of significant digits to be displayed on the marker label.
7. Click again on the tab for the type of cursor for which you specified values and click *OK* or *Apply*.

Using Bindkeys to Add Markers

Markers in WaveScan are coupled to cursors, and at any point while using a cursor you can hit the `m` bindkey and get a corresponding marker.

- **Trace Marker**
  Use the currently defined cursor bindkey, `m` being the default, to create a marker at a point on the trace nearest to the mouse pointer position. If the trace cursor is on, the marker is placed at the current position of the trace cursor.
- **Vertical Marker**
  Use the currently defined cursor bindkey, `V` being the default, to create a marker at the x-location of the mouse pointer. If the vertical cursor is at either `V` or `m`, it results in a marker at the current cursor position.
- **Horizontal Marker**
  Use the currently defined cursor bindkey, `H` being the default, to create a marker at the

y-location of the mouse pointer. If the horizontal cursor is at either *H* or *m*, it results in a marker at the current cursor position.

- **Delta Marker**
  Use the currently defined cursor bindkey, `M` being the default, to create a trace marker at a point on the trace nearest to the mouse pointer. To add a second delta component to the marker, use the *a* bindkey. Also, if the delta cursor is on, typing *M* results in a delta marker spanning the current cursor positions.

Zooming

WaveScan adds X and Y zoom to the regular zoom function. Regular zoom is invoked by drag and drop of the right mouse button or the *z* bindkey (or the *Zoom – Zoom* menu option). The X and Y zooms can be initiated using the *x* and *y* bindkeys. The graph is returned to its original scaling with the fit function (*f* bindkey). For multiple level zooms, the graph can be incrementally unzoomed with the unzoom function (*u* bindkey). For more information, see the *Working with Graphs* chapter (*Zooming a Graph* section) of the *WaveScan User Guide*.

Reset Option

The *Reset* window option is not available in WaveScan.

To create an empty graph, click on the *New Window* (⧉) icon.

Modifying Objects

WaveScan provides Attributes dialog boxes that change display of the object selected. Double-clicking on any graph object brings up the dialog initialized to the current values of the chosen object. You can also access Attributes dialog boxes through the graph window menu when they have an *Edit* entry.

Customizing Graphs

- Double-click in the graph to bring up the Graph Attributes dialog box.
- When you select a graph, the *Label* field appears in the tool bar. You can enter the label text, click the checkbox next to it, and then use the mouse to place the label.
- Other important graph controls are available from the *Graph* menu:

- **Layout**
WaveScan supports three new layout styles: *vertical*, *horizontal* and *card*. The card layout stacks graphs one on top of another. A graph is selected by its index.

- **Color Schemes**
Colors in a graph are fully customizable. Easy preset color schemes are made available from the *Graph* menu to coordinate graph background and object foreground colors with one selection.

- **Fonts**
Graph fonts are automatically resized according to the available graph window area. Three size ranges are available: small, medium and large.

- **Templates**
Graphs can be loaded as templates. When loaded as a template, a graph's attributes are used to set the system defaults. Any subsequently created graph uses those attribute settings.

For more information, see the *Working with Graphs* (*Changing the Graph Layout* section) chapter of the *WaveScan User Guide*.

Customizing Traces and Axes

WaveScan is a data-centric tool. When traces are added to a graph the waveform x & y scale units are compared to the scales of the existing graph. If the x unit of a waveform does not match the x-scale of the graph a new subwindow is automatically created. If the y-unit does not match an existing y-axis scale, a new y-axis is created. A maximum of four y-axes per graph are allowed.

- **Strip Charts**
Scrollable strip charts are available. The number of visible rows can be set from the Trace Attributes dialog box. For digital traces, the attribute dialog control allows the number of digital rows to be set. The digital row height can be adjusted by moving a small handle on the right hand side of mixed signal graphs to increase or decrease the digital display area. Scrollable strip charts are not available in AWD.

- **Assigning Axes**
When traces share the same y-axis unit they can be forced to have separate y-axes using the *Trace – Assign to Axis* menu option. In AWD, you can do this through the *Axes* menu.

- **Copy/Move a Trace**
Any selected set of traces can be copied or moved to a new graph subwindow or window with the *Trace – New Graph* option.

- ❑ **Trace information**
  Simple unqualified signal names are used in legends to identify traces. This improves readability and is usually sufficient for identification of plotted signals. To see a fuller description of a trace, point to the trace and press the Alt key: a bubble popup appears with the data directory and dataset for the trace. You can also point to the trace and hit the *t* bindkey and place a trace info marker showing X and Y coordinates. This feature is not supported in AWD.

- ❑ **Trace Save and Load**
  Trace data can be saved in ASCII format with *Trace – Save*. *Trace – Load* recreates a trace with the saved data and places it in the currently selected graph. This feature is not supported in AWD.

- ❑ **Axis Scaling**
  Control of axis scaling is provided through the Axis Attributes dialog box. *Auto* scaling gives full control to the program. *Min-Max* requires user input of the scale limits and *Manual* requires user input of scale limits and divisions. Simple scale limit changes can be made from the tool bar. In AWD, you can do this through the *Axes* menu.

Customizing Markers and Labels

Markers are annotation objects usually tied to a particular trace and positioned according to scale coordinates of the trace. You can place markers such that they are not attached to a trace by toggling off the *Attach Trace* option. Markers consist of a marker symbol, a text label and a connecting arrow. Labels are simple text strings that are positioned according to window coordinates. Their position does not vary with zooming and scale changes.

- ❑ **Label Format Variables**
  Labels support embedded format variables. This allows, for example the x & y coordinate (%X, %Y) of a marker to be embedded in the marker label. For example, `Peak voltage =%Y at %X`.

- ❑ **Expressions**
  You can set expressions for marker labels or simple labels by double clicking on the label. The Label Attributes dialog box provides text fields for the string and expression. The format character for an expression is %E: Eg `Rise time =%E`. Any expressions saved as a memory from the Calculator becomes available from the expression combo pulldown menu.

Saving and Loading of Graphs

- ■ **Saving a Graph as a File**
  To save a graph as file, select *File – Save*. Opening the saved graph file restores the

graph window to its original state. The saved object is an xml-based description file of the graph window contents with name and path references to all the signals in the graph. Consequently, loaded graphs always reflect the current values of the dataset of the referred signal, not necessarily those at the time the graph was saved.

- **Printing and Saving**
  The print dialog provides standard Unix printing options. You can also save graphs to a file in PostScript format from this dialog

- **Saving a Graph as an Image**
  To save any graph as an image file select *File – Save as Image*. Currently supported formats include PNG, TIFF and BMP.

Note: These three features are not supported by AWD graphs.

When ADE issues a hardCopy() command with an output file destination setup by hardCopyOptions, WaveScan checks the file extension of the specified destination file. It then determines whether to save the output as an image file rather than a postscript file based on the following mapping:

```
.png .PNG     Saves a PNG image file
.tif .TIF     Saves a TIFF image file
.tiff .TIFF   Saves a TIFF image file
.bmp .BMP     Saves a BMP image file
Otherwise,  Saves a Postscript file
```

Creating a Digital Representation of an Analog Signal

You can create a digital representation of an analog signal to digital by selecting it and clicking the *a2d conversion* ( ) icon. This brings up the Analog to Digital Conversion form, in which you can specify *Logic Threshold* and *Time to X* values for the selected net.

Bindkeys

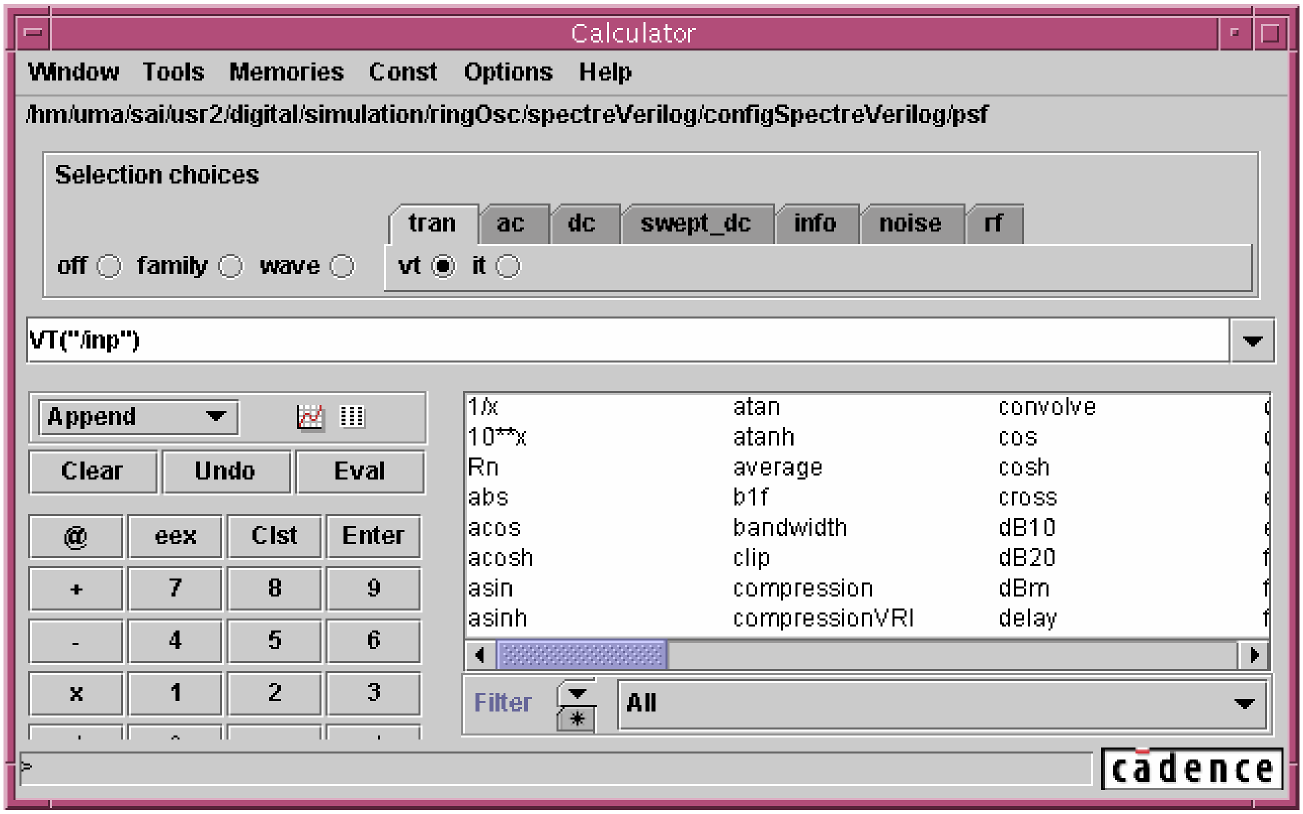
A bindkey is a keyboard key or sequence of key press events bound to a WaveScan task. When you press the key, the command executes. The default bindkey is displayed next to the appropriate menu option in the Graph Window. Also, several common graph functions have associated bindkeys defined. They are generally the most efficient way to invoke actions.

You can also define your own bindkeys. The *Help – Shortcut Keys* menu on the WaveScan graph provides a list of user-defined bindkeys or shortcut keys.

For more information, see the *Setting Bindkeys* topic in the *Working with Graphs* chapter of the *WaveScan User Guide*.

Calculator

The Calculator is a much used tool in any design-simulate-verify cycle. In the WaveScan version, a current ADE user will notice a significant improvement in the organization, layout and use model. The WaveScan Calculator user interface presents a clear separation of operators, functions and schematic-selection based selection choices.



Calculator

/hm/uma/sai/usr2/digital/simulation/ringOsc/spectreVerilog/configSpectreVerilog/psf

Selection choices: off ○ family ○ wave ○ — tran: vt ● it ○

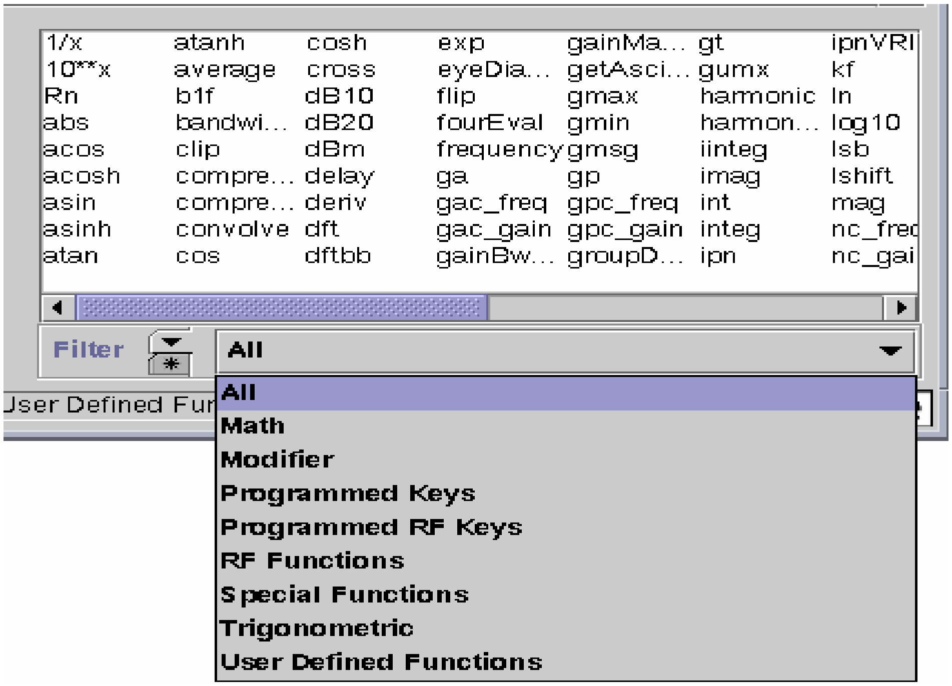
VT("/inp")

Append

Filter: All

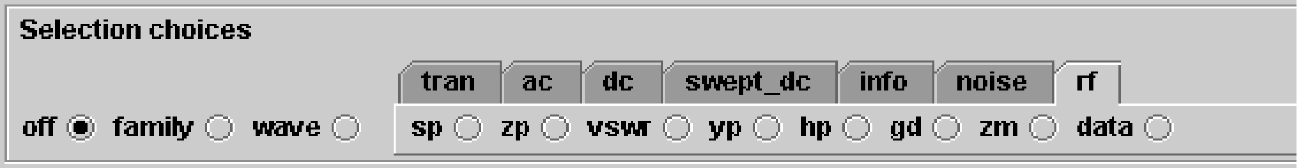
Function Organization

WaveScan has a more organized way of displaying functions. Functions are organized into categories that are accessible through the *Filter* drop down menu. The categories are: *All*, *Math, Modifier*, *Programmed Keys*, *Programmed RF Keys*, *RF Functions, Special Functions, Trigonometric* and *User Defined Functions.*



- **Math, Modifier, Programmed Keys, Programmed RF Keys and Trigonometric functions**
  In WaveScan, these functions are accessible through the *Filter* drop-down menu. In AWD, the *Math* functions (`1/x`, `10**x`, `abs`, `dB10`, `dB20`, `exp`, `int`, `ln`, `log10`, `sqrt`, `x**2`), *Modifier* functions (`dBm`, `imag`, `mag`, `phase`, `real`), *Programmed Keys* (`f1, f2, f3, f4`), *Programmed RF Keys* (`rf1 ... rf8`) and the *Trigonometric* keys (`acos, acosh, asin, asinh`) are all located in the Calculator key pad.

- **RF Functions**
  In WaveScan, RF functions are provided through the folder tab *rf (*containing *sp, zp, vswr, yp, hp, gd, zm, data).* In AWD, when you select the RF button the Calculator key pad redisplays to show a different set of schematic selections and function keys.



- **Circle Plot Functions**
  Circle plot functions (*gac*, *gpc, nc, ssb, lsb*) are handled differently in AWD and WaveScan. In WaveScan, you plot the circle function result, then select (on the graph) the display type in the *Graph – Display* type menu. In AWD, each of the circle plot

functions provides a *Z-Smith* and *Y-Smith* radio button that determines the plot type of the graph. The WaveScan display types *Impedance* and *Admittance* correspond to the AWD Plot Type arguments *Z-Smith* and *Y-Smith*, respectively.

The AWD functions in the table below are split into two separate WaveScan functions Each WaveScan function identifies a specific sweep (frequency or gain). This is handled as an additional option in the AWD equivalent function.

AWD Function	WaveScan Equivalent
gac	gac_freq, gac_gain **Note:** The AWD gac dialog provides the choice of sweeping frequency or gain. Depending on what you select, the form redisplays. In WaveScan two distinct dialogs are provided. The gac_freq describes a choice of frequency sweep and the gac_gain describes a choice of gain sweep.
gpc	gpc_freq, gpc_gain
nc	nc_freq, nc_gain

- **Special Functions**
  WaveScan and AWD provide the same functions in the *Special Function* category, except that AWD provides two modes for each function: RPN and Algebraic. WaveScan can toggle the RPN mode off and on. While AWD has a special RF mode, the WaveScan Calculator provides an *RF Functions* category and an *RF schematic selections* folder.

- **User-Defined Functions**
  Custom SKILL functions that are used with the current AWD Calculator can be used directly in the WaveScan Calculator. You can continue to create and access custom SKILL functions created via the *calSpecialRegisterFunction* method in both AWD and WaveScan. In the WaveScan Calculator, such functions are displayed in the category named *User Defined Functions*. For details, see the *Working with the Calculator* chapter (*Defining Functions and Function Keys* section) of the *WaveScan User Guide*.

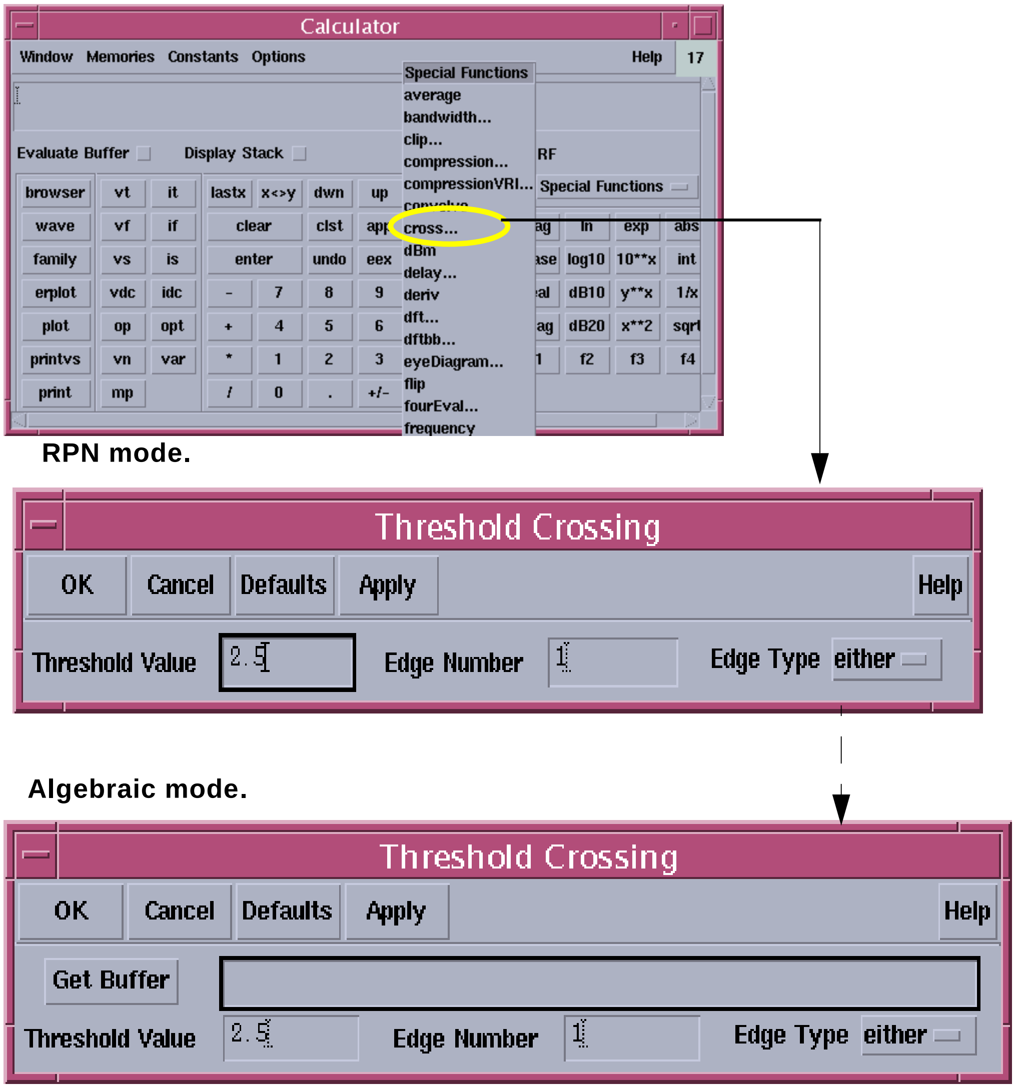
  WaveScan also supports the standard filtering capabilities. For example, to display all the functions which begin with the letter `c,` select the category *All*. You can display all the functions that begin with c using regular expressions.

Specifying Arguments for a Function

When you select a multi-parameter function from the function list in WaveScan Calculator, the function list area changes to a function panel. The function panel is the user interface for the selected function.

In WaveScan, the function panel for a specific function is the *same* regardless of whether you are in the RPN mode or the Algebraic mode. In AWD however, the *Signal* field is *not* displayed for the RPN mode. For example, compare the GUI for the function *cross* in both AWD and WaveScan Calculators. You would see that the AWD dialog for the function *cross*, differs depending on whether AWD is in the Algebraic or RPN mode.

AWD Calculator



RPN mode.

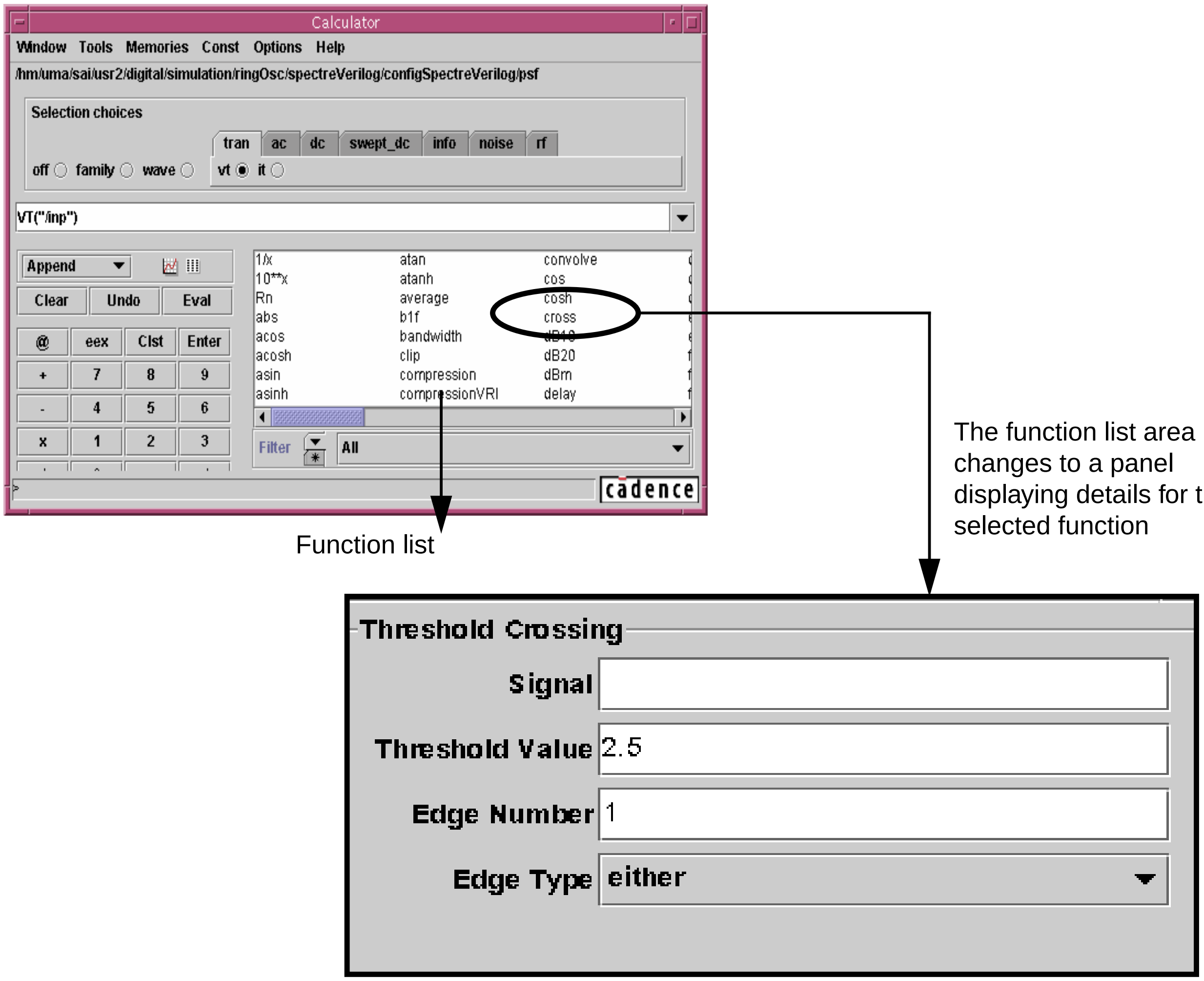
Threshold Value 2.5 Edge Number 1 Edge Type either

Algebraic mode.

Threshold Value 2.5 Edge Number 1 Edge Type either

This is not true for WaveScan because the signal field exists in both Algebraic and RPN modes.

WaveScan Calculator



/hm/uma/sai/usr2/digital/simulation/ringOsc/spectreVerilog/configSpectreVerilog/psf

VT("/inp")

Function list

The function list area changes to a panel displaying details for the selected function

Threshold Crossing

Signal:

Threshold Value: 2.5

Edge Number: 1

Edge Type: either

For details, see the *Working with the Calculator* chapter (Function Panel section) of the *WaveScan User Guide*.

Manipulating the Buffer

In the RPN Mode

The WaveScan Calculator does not provide a *Display Stack* option (as the AWD Calculator does).You view the RPN stack via the drop down menu on the right side of the buffer.

Stack manipulation is easier in WaveScan. No keys are required. It does not need the *lastx, x<>y, dwn, up, app* buttons. You manipulate the stack by using the buffer drop down menu to select an item in the stack. The selected item is then moved to the buffer.

In the Algebraic Mode

The *WaveScan Calculator* differs from *AWD* in the way selected signals are managed in the buffer when creating single argument expressions.

Consider an example describing how to create the following expression:

```
average(VT("net32"))
```

a. In the WaveScan Calculator, select *Options – Set Algebraic.* Select the *tran* tab and then *vt*.

b. Select a net in the schematic window. The Calculator buffer will contain the *vt* expression. For example: `VT("/net39")`.

c. Next, select the category, *Special Functions*. Select *average.* Now, instead of appending `average(` to `VT("/net39")`, which is the AWD behavior, the buffer contains `average(VT("/net39")).`

Note: AWD would contain, `VT("/net39)average(`

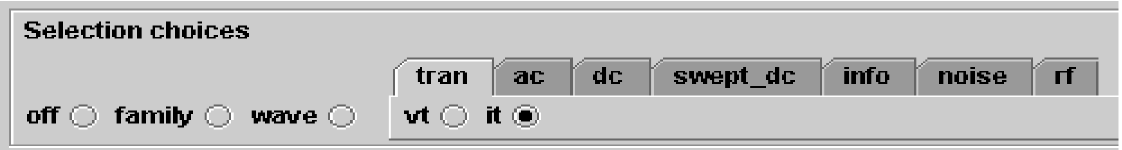
When you select an object, the expression which has just been added is in a special *just selected* state. (Hence, the signal is greyed out). If your next action is to select a single argument function, the Calculator interprets that as a request to wrap the *just selected* signal in the function. The *just selected* state is cancelled when you perform some other action.

Selecting Data in Building Expressions

The WaveScan Calculator provides a compact use model for selection of nets on a schematic. It provides three basic option buttons to help you define the selection mode:

- *off* – You can select this option button to prevent unintentional selections from populating the buffer. You can also activate this mode by also pressing the `Esc` key in the schematic from which you are currently making selections.
- *family* – When you select this, you can select any browser signal from a parametric dataset or any graph trace that represents a parametric leaf. The expression in the Calculator evaluates the entire waveform family.
- *wave* – When you select this and then select any browser signal or graph trace, if the trace represents a parametric leaf, the expression evaluates to the selected leaf.

The analysis folders, such as *tran*, and related option buttons, such as *vt* and *it*, let you initiate schematic selection of signals by designating the required signal accessor function.


Selection choices

off ○ family ○ wave ○

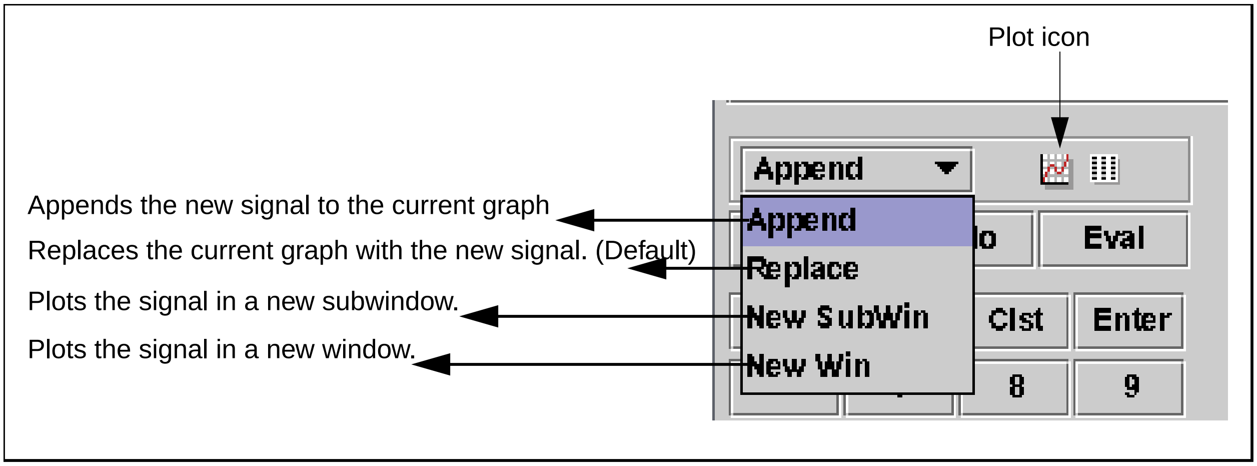
For details, see the *Working with the Calculator* chapter (*Selecting Signals* section) of the *WaveScan User Guide*.

Operating on a Zoomed-In Part of a Trace

The Calculator can operate on a part of a trace when the *Options – clip on graph selection* command is selected. By default, it is selected. Calculator operations then apply to the X-range of the zoomed-in portion of the trace that is visible in the graph window. If you would rather have the Calculator operate on the entire waveform, you can deselect the *clip on graph selection* command.

Plotting

WaveScan provides more flexibility in choosing the destination of plots. Its supports four plotting modes that are available through a drop down menu:


Plot icon
Appends the new signal to the current graph
Replaces the current graph with the new signal. (Default)
Plots the signal in a new subwindow.
Plots the signal in a new window.
Append
Append
Replace
New SubWin
New Win
Eval
Clst
Enter
8
9

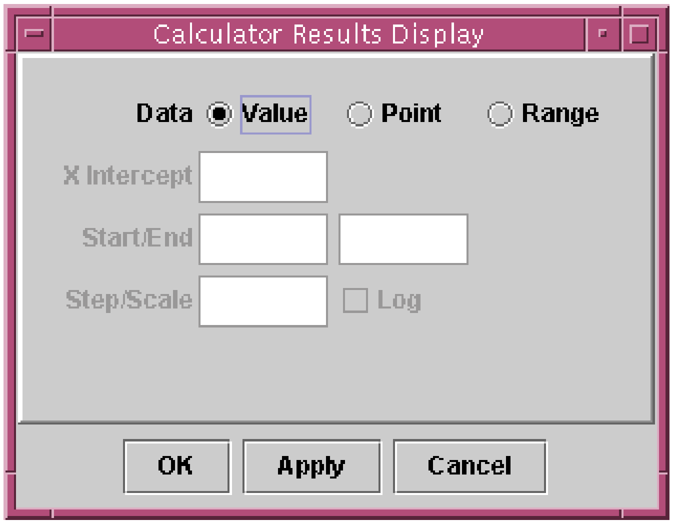
You can plot the results of expressions, using the *plot* icon (📈). The expression is plotted as defined by the selected destination dropdown menu.

For details, see the *Working with the Calculator* chapter of the *WaveScan User Guide*.

Note: These four plotting modes are also available in the ADE simulation window, the browser and Direct Plot main form.

Printing

Printing results is easier in the WaveScan Calculator. The *tabular results display* icon (▦) is provided next to the *plot* icon. Alternatively, you can choose *Tools – Table*. This brings up the *Calculator Results Display* form. It contains a field named *Data* that offers options for you to define how the expression value is to be displayed in the ADE *Results Display* Window.



Calculator Results Display

Data ● Value ○ Point ○ Range

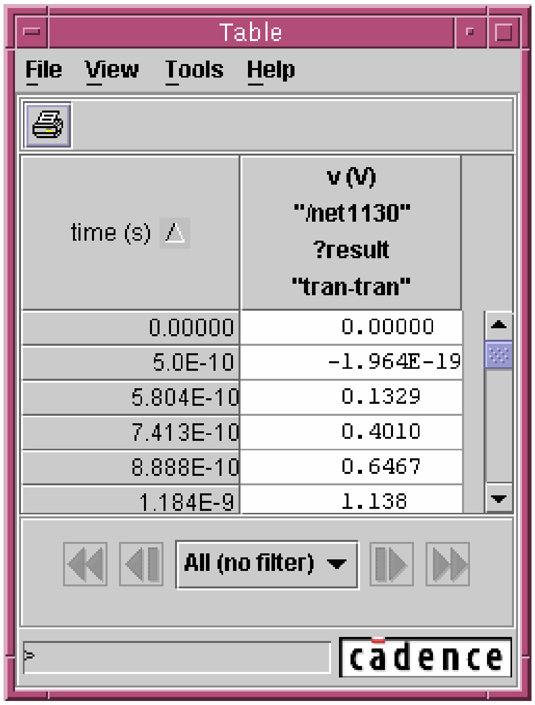
X Intercept

Start/End

Step/Scale □ Log

OK Apply Cancel

When you click *OK*, the results are displayed in a tabular format as shown here:



Table

File View Tools Help

time (s) △	v (V) "/net1130" ?result "tran-tran"
0.00000	0.00000
5.0E-10	-1.964E-19
5.804E-10	0.1329
7.413E-10	0.4010
8.888E-10	0.6467
1.184E-9	1.138

All (no filter)

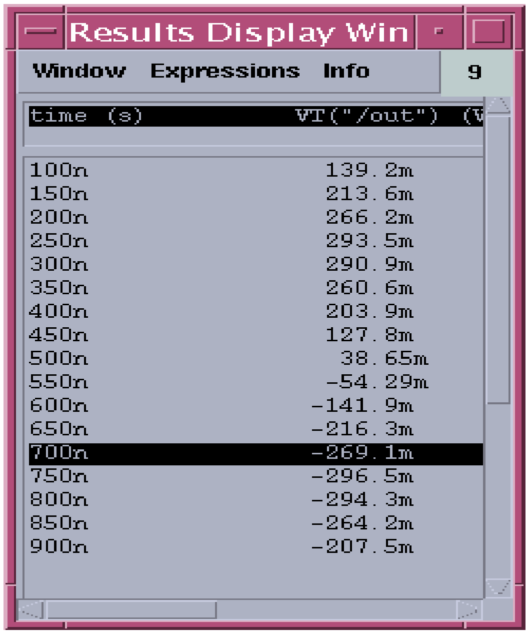
If you would rather see the WaveScan Calculator print results in AWD Results Display instead of WaveScan Tables, you can do so by setting the following WaveScan environment variable:

```
wavescan.calculator useAwdResultsDisplay string "false" nil
```

For details, see the *Working with the Calculator* chapter (*Printing Expressions* section) of the *WaveScan User Guide*.

In AWD, the *print* and *printvs* buttons are used. The *print* button prints the value of the Calculator buffer expression against a single value of the independent variable. The *printvs* button prints a table showing the value of the buffer expression over a specified (or default) range.

The *print* button offers similar functionality as the *Point* button in the WaveScan Calculator Results Display form does. The *printvs* button displays the waveform data in its raw form when the range is not specified. This corresponds to the *Value* option in the Calculator Results Display form. If the range is specified, then data is sampled and displayed for the given range. This corresponds to the *Range* option.

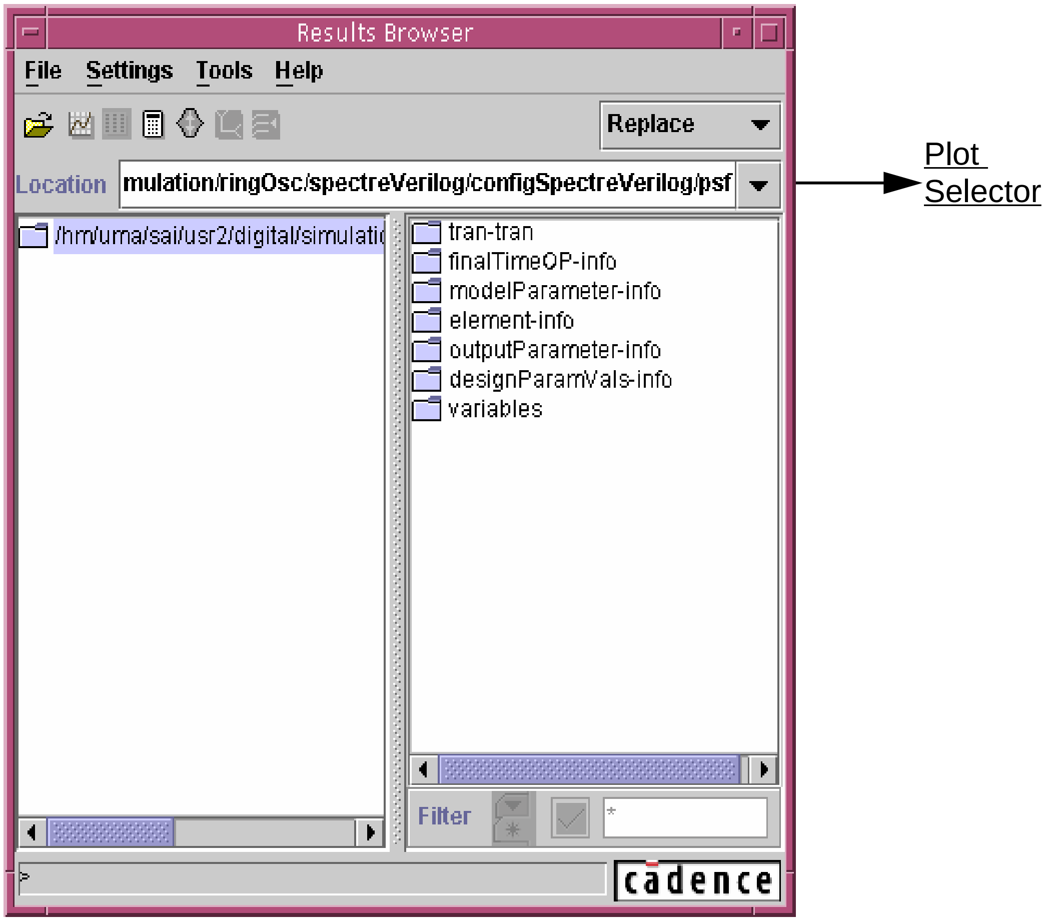


```
time (s)            VT("/out") (V
100n                  139.2m
150n                  213.6m
200n                  266.2m
250n                  293.5m
300n                  290.9m
350n                  260.6m
400n                  203.9m
450n                  127.8m
500n                   38.65m
550n                  -54.29m
600n                 -141.9m
650n                 -216.3m
700n                 -269.1m
750n                 -296.5m
800n                 -294.3m
850n                 -264.2m
900n                 -207.5m
```

For details, see the *Waveform Calculator User Guide* (*Plotting or Printing Results* section).

Results Browser

The WaveScan Results Browser offers several new features for the location and display of simulation results. For a more thorough description of the WaveScan Results Browser, see the *Waveform Calculator User Guide* (*Accessing Data* in *Chapter 2*).



Data Selection

Compared to AWD, the WaveScan Results Browser presents data in a more organized way. Directories are represented in a split-paned, tree structure format. It provides a convenient way to select and browse multiple simulation datasets and signals. You can specify the range of data to be read. You can also open new data by using the *Choose Data Directory* dialog, which can be brought up using *File – Open Results* menu, or by using the *Open* icon (📂) on the tool bar. The *Choose Data Directory* dialog allows you to select data from any location in your directory structure. Plottable results are highlighted in blue.

There also a filter mechanism in the browser which can be used to filter the displayed signal list according to a category (All, V, I and so on) or according to a regular expression (for example, net*).

Note: The WaveScan Results Browser does not need the *runObjFile* to view and access data. Therefore, the WaveScan reader requires no manual ROF creation and there is no equivalent of CreateROF in WaveScan.

Location History

A list of previously accessed results directories are maintained in the location drop down menu. Make a selection to reload the results.

Data Display

Data display is separated between two panes. The top level data directories and datasets are displayed in the left pane and the signal names are separately displayed in the right pane.This separation allows compact representation of long signal lists. Category based and regular expression based filtering is available for the signal display pane.

Plot Selection

Four plot modes are available from the plot selection combobox or the right mouse button popup: *Append*, *Replace*, *New SubWin* and *New Win*. Single-click plotting is available with middle mouse button selection of a signal. Multiple selection is supported with standard `Ctrl` and `Shift` selection modes. Use the right mouse popup to plot a collection of signals.

Note: Double-clicking the left mouse button on a signal plots it. A single middle mouse button click does the same. The right mouse button brings up a popup menu that you can use to send the data to a table or to send the signal to the Calculator.

Plotting Sweeps

The *Data Selection* dialog provides full control for reading and displaying swept data. WaveScan supports ranging for transient data. You can specify the start and end times of the data to be read. This can be a powerful tool for reducing plotting times for very large datasets. For parametric swept data you can pick individual sweep parameter values to be represented when plotting.

Complex Data

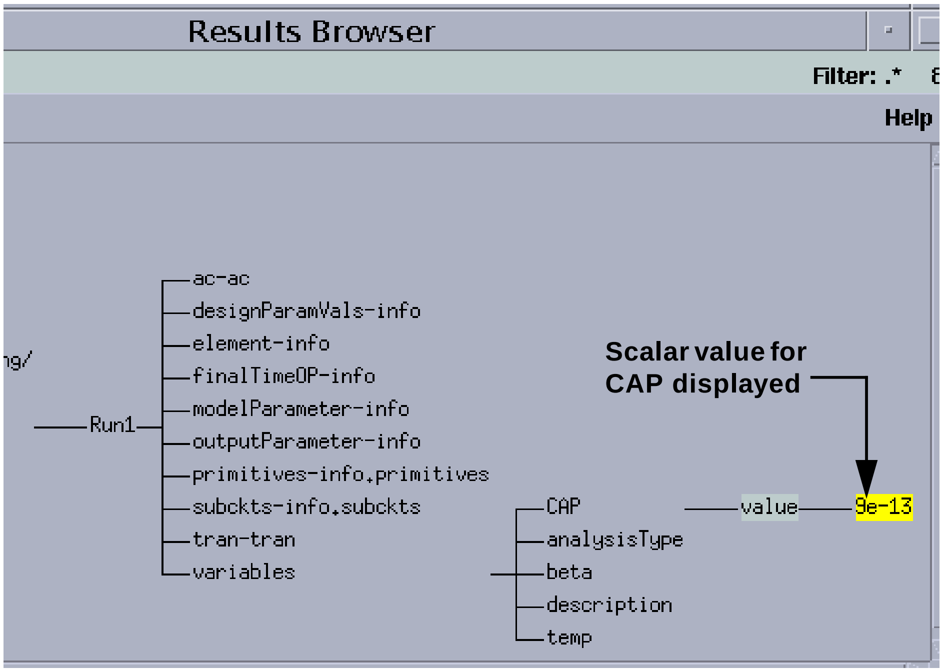
Alternate graph types are available from the Graph Type combo for complex data. Rectangular, polar, Smith impedance and admittance, and Real vs Imaginary plots are supported. A choice of complex scale modifiers (Mag, Phase, WPhase, Real, Imag, dB10 and dB20) is available for complex rectangular plots.

YvsY and Diff

Quick functions are provided for plotting YvsY and the difference between two signals. Select a trace, then select the appropriate icon button (YvsY or Diff) and finally select the second signal. The plot appears.

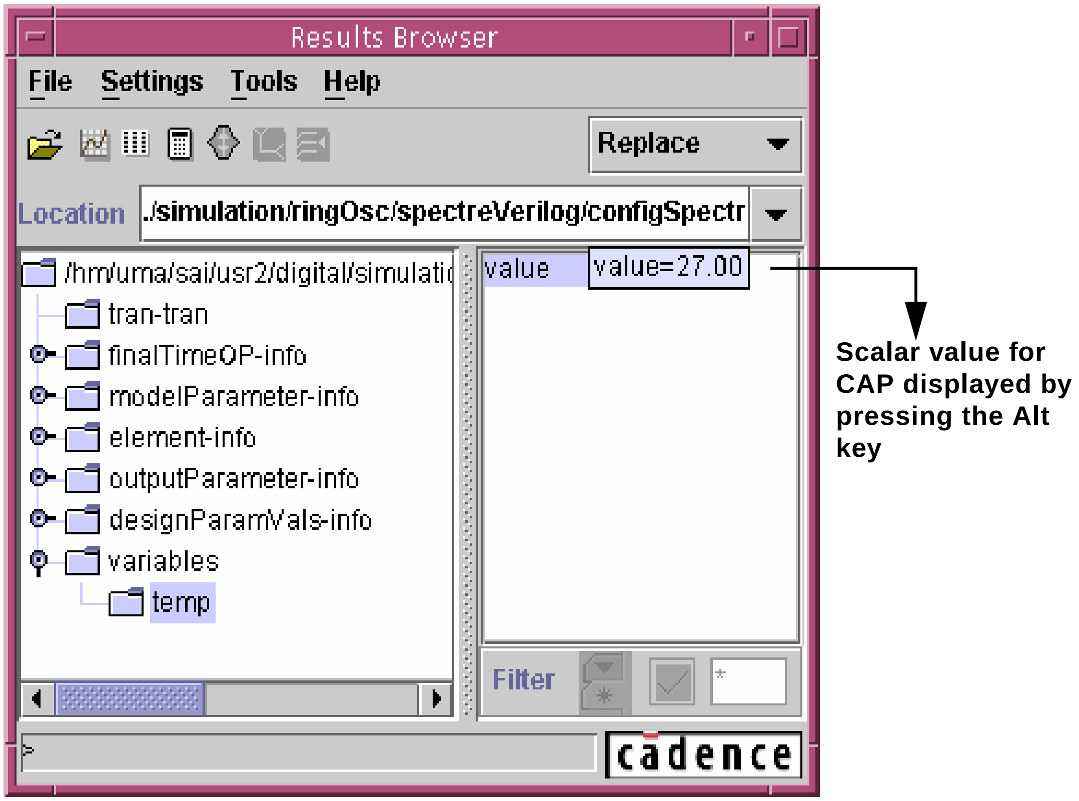
Scalar Data

The display of scalar data is different in WaveScan than in AWD. In AWD, scalar values are displayed in the Results browser, as shown in the figure below:



Scalar value for CAP displayed

In WaveScan, select a signal and press and hold the Alt key to see the scalar value of the signal.



Scalar value for CAP displayed by pressing the Alt key

Note: Scalar and waveform data can be sent to a table from the browser.

SST2 Data Support

The WaveScan Results Browser can read and plot SST2 data.

Multiple Signal Selection

In WaveScan, you can select multiple signals to plot for a given analysis, and can also send their results to the results table. However, in AWD, you can either select a single signal or all the signals, but not a subset.

SKILL functions: Differences in Behavior

Most of the SKILL functions work the same way in AWD and WaveScan. Those functions that differ are listed in the table below. A list of callback functions is also provided.

List of Public SKILL functions that are emulated correctly but with some properties not supported

Function	Change in Property
`awvAppendExpression`	*`lineType`* not supported.
`awvAppendList`	*`lineType`* not supported.
`awvAppendWaveform`	*`lineType`* not supported.
`awvClearSubwindowHistory`	Axes not deleted.
`awvClearWindowHistory`	Axes not deleted.
`awvCrossHairDeleteList`	Not supported.
`awvDisplayDate`	Date displayed on all subwindow titles.
`awvExitWindowFunctionAdd`	No hi function should be added.
`awvGetXAxisLabel`	The *`label`* refers to the sweep name. Units are not a part of label string. *`computed`* is not supported.
`awvGetYAxisLabel`	The *`label`* refers to the sweep name. Units are not a part of label string. *`computed`* is not supported.
`awvInitWindowFunctionAdd`	No hi function should be added.
`awvPlaceWaveformLabelawvSetCursorPrompts`	*`color, justify, fontStyle, and orient`* are not supported for WaveScan.
`awvPlaceWindowLabel`	*`color, fontStyle, and orient`* are not supported for WaveScan.
`awvPlotExpression`	*`lineType`* not supported.
`awvPlotList`	*`lineType`* not supported.
`awvPlotWaveform`	*`lineType`* not supported.
`awvSetCursorPrompts`	Not supported.

Function	Change in Property
`awvSetOptionDefault`	Most of the options are not supported in WaveScan, hence even though the options are set they are not considered during creation of the window/ subwindow.
`awvSetOptionValue`	Most of the options are not supported in WaveScan, hence even though the options are set they are not considered during creation of the window/ subwindow.
`awvSetOrigin`	Not supported.
`awvSetPlotStyle`	A new plot style *`polezero`* is also supported. *`auto`* is mapped to default WaveScan plot style i.e. *`rectangular.`*
`awvSetXAxisLabel`	Label consists of only the sweep name, hence setting of the label does not change the units.
`awvSetYAxisLabel`	Label consists of only the sweep name, hence setting of the label does not change the units.
`awvSimplePlotExpression`	*`lineType`* is not supported.
`awviGetRGBList`	Deleted.

List of Calculator functions that are not supported

```
calCloseCalculatorCB
caliMemoriesFileFormCB
caliMemoryNameFormCB
caliShowMemoriesCB
caliModeToggle
caliRestoreDefaultWindowSize
```

List of Callbacks

These functions are not emulated because they do not have a *WaveScan* equivalent.

```
awvAxesOptionMenuCB
awvCloseWindowMenuCB
awvCrossHairMenuACB
awvCrossHairMenuBCB
awvCursorMenuCB
awvCurvesMenuCB
awvDeleteMenuCB
awvEraseWindowMenuCB
awvExpandBusMenuCB
```

```
awvFastZoomInMenuCB
awvFastZoomOutMenuCB
awvFitMenuCB
awvHardCopyMenuCB
awvHorizontalMarkerMenuCB
awvLabelMenuCB
awvLoadMenuCB
awvMakeBusMenuCB
awvMakeWindowActiveMenuCB
awvMoveMenuCB
awvPanMenuCB
awvPlotStyleMenuCB
awvRedrawWindowMenuCB
awvResetWindowMenuCB
awvRotateStripsMenuCB
awvSaveMenuCB
awvSmithAxisMenuCB
awvSubwindowMenuCB
awvSubwindowTitleMenuCB
awvSwapStripsMenuCB
awvTitleMenuCB
awvToCompositeMenuCB
awvToSmithAdmittanceMenuCB
awvToSmithImpedanceMenuCB
awvToSmithPolarMenuCB
awvToStripMenuCB
awvUndoMenuCB
awvUpdateWindowMenuCB
awvVerticalMarkerMenuCB
awvXAxisMenuCB
awvYAxesMenuCB
awvZoomInMenuCB
awvZoomOutMenuCB
awviEditMenuCB
awviMakeActiveMenuCB
awviPLoadMenuCB
awviPSaveMenuCB
awviPUpdateMenuCB
awviShowOutputMenuCB
```

Migration Information

The following table lists problems that you may encounter while moving from AWD to WaveScan.

	Issue	Impact
1.	Actions taken by users on WaveScan are not logged. Also, logfiles created for ADE-AWD cannot be used as replay files for WaveScan integrated into ADE.	Replaying a log file that contains actions on AWD, its Calculator or its Results Display Browser may yield unexpected results.
2.	Waveform state data saved when using AWD will not be loaded when using WaveScan templates and vice-versa For example, if users select *WaveScan* as the viewer and have existing Artist states with AWD data, the state data will not be translated to *WaveScan*'s templates.	Waveform state information in the user's existing ADE state will be ignored. The user must re-create the waveform state information for WaveScan.
3.	There are several public Skill functions that are implemented as dummy functions by WaveScan. For a list of these functions, see List of Public SKILL functions that are emulated correctly but with some properties not supported.	Some AWD functions, when used in CIW or in ocean, may seem to have no effect.
4.	AWD callback functions (that are activated when AWD GUI buttons are clicked) will not be emulated because they do not have any *WaveScan* equivalent. For each of the buttons on the AWDGraph GUI, there is an associated callback. For the list see, List of Callbacks.	Many AWD functions when used in CIW or in ocean have no effect.
5.	It is recommended that you do not pass any WaveScan window IDs as arguments to hi calls. These are dummy IDs mapping to WaveScan windows.	*hi* based customizations do not work with WaveScan.

	Issue	Impact
7.	Existing traces on a WaveScan Graph can be selected to enter their expressions into WaveScan Calculator with the following exceptions: a parametric waveform with a swapped x-axis, user defined buses from signals, user defined signals from expanded busses and SST2.	None.
8.	AWD plots signals in reverse order from WaveScan when using the Browser. For example, plot two different signals (say, net1 and net2) using the Results Browser (AWD and WaveScan). After plotting the signals, switch the graph type to *strip*.	In case of WaveScan, you see signal `net1` as the first signal and the signal `net2` as the second. In case of AWD, you see the reverse behavior.

C

auCdl Netlisting

This appendix describes the auCdl (Analog and Microwave Circuit Description Language) netlisting procedure. It contains details on parameters required for auCdl and also the different ways to netlist to auCdl. This information is applicable to any 4.4 version of the Virtuoso® design framework II (DFII).

This appendix covers the following topics:

What Is auCdl and Why Do You Need It?

To compare a layout versus a schematic (LVS) using LOGLVS, you need a netlist representation of the schematic for a design for LOGLVS. The netlist must be in CDL (Circuit Description Language) format. CDL Out translates a DF II schematic design into CDL netlist format suitable for Dracula® verification products. To create a CDL netlist for an analog circuit, you use a netlister called auCdl (Analog and Microwave Circuit Description Language).

From IC5141USR2, auCdl will support the basic features of Hierarchy Editor that OSS supports. To know more about these features, see the appendix chapter *Support for HED* in *Open Simulation System Reference*.

Running auCdl

You can run auCdl inside or outside the DFII environment.

To translate files from the DFII database format into an auCdl netlist, follow the steps below:

1. Set the environment variable by adding the following line into your `.cshrc` file:

   ```
   setenv CDS_Netlisting_Mode "Analog"
   ```

2. Create an auCdl view for the cell by copying the symbol view to the auCdl view.
3. Add the auCdl simulation information to the cell's CDF. For more specific information, see "CDF Simulation Information for auCdl" on page 658.

You can customize the auCdl Netlister by using your simulation run control (`.simrc`) file. For more information about `.simrc`, see "Customization Using the .simrc File" on page 639.

How to Run auCdl from within DFII

In any version 4.4 DFII, you can extract an auCdl netlist by following these steps:

1. In the CIW, choose *File – Export – CDL*.
2. In the CDL Out Run form, fill in the appropriate fields and click *OK* or *Apply*.

For more information about using CDL Out, read the *Translating CDL Files* section in the *Design Data Translators Reference*.

How to Run auCdl from the Command Line

To run CDL Out from the command line, you must prepare a simulation environment (`si.env`) file in advance and name the file as a command argument. Run CDL Out interactively once to create the `si.env` file. Once the `si.env` file is created,

1. Copy the `cds.lib` file to the run directory.
2. Type the following at the command line to CDL:

   ```
   si -batch -command netlist
   ```

   the `-batch` option runs CDL in batch mode and the `-command netlist` option generates an ASCII netlist file.

CDL Out can generate a hierarchical netlist. CDL Out generates a netlist hierarchy that duplicates the hierarchy of your design. Each cell in your schematic becomes a separate subcircuit in the netlist. The hierarchical netlister automatically prefixes each instance name with the proper character for its element type; for example, "M" for MOSFET and "R" for resistor. This prefixing minimizes mapping and name translation.

Sample si.env File

The following is an example of a `si.env` file followed by description of each of these properties.

```
simLibName = "testLib"
simCellName = "testTop"
simViewName = "schematic"
simSimulator = "auCdl"
simNotIncremental = nilsimReNetlistAll = nil
simViewList = '("auCdl" "schematic" "gate.sch" "cmos.sch")
simStopList = '("auCdl")
simNetlistHier = t
hnlNetlistFileName = "netlist"
simRunDir = "/cds/1.0/test/translator/cdlout/paramCase/"
resistorModel = " "
shortRES = 2000.0
preserveRES = 'nil
checkRESVAL = 'nil
checkRESSIZE = 'nil
preserveCAP = 'nil
checkCAPVAL = 'nil
checkCAPAREA = 'nil
preserveDIO = 'nil
checkDIOAREA = 'nil
checkDIOPERI = 'nil
displayPININFO = 'nil
preserveALL = 'nil
```

Description of si.env Properties

Property	Description
`simLibName`	Name of the library containing the top-level cellview of the design.
`simCellName`	Name of the top-level cellname of the design.
`simViewName`	Name of the top-level view of the design.
`simSimulator`	Simulator to run.
`simNotIncremental`	When this property is set to `nil`, the netlister runs incrementally, which means the system netlists only the parts of your design you modified since you last netlisted the design. The default is `nil`.
`simReNetlistAll`	When this property is set to `t`, the netlister runs a new netlist on all the cellviews in your entire design. The default is `nil`.
`simViewList`	List of views to open for each cell when traversing the design hierarchy during netlisting and name translation.
`simStopList`	List of views that are valid stopping points for expansion used during netlisting.
`hnlNetlistFileName`	Name of the text netlist file.
`simRunDir`	Directory in which CDL data is stored. Set this global variable to the current run directory. This variable is set when the simulation environment is initialized.
`resistorModel`	String that sets the model name to be treated as a short. Prints out the string in the `*.RESI` command. The default is `nil`.
`shortRES`	Sets the value of resistance below which the resistor is assumed to be shorted. Prints the value out in the `*.RESI` command. The default is `2000.0`; type is floating point.
`preserveRES`	When this property is set to `t`, resistors are preserved for checking in LVS, `shortRES`, and `checkRESSIZE`. Using the optional variable [XX], you can specify a model name that preserves only the specified type of resistor. The default is `nil`.
`checkRESVAL`	Prints out `*.RESVAL` when set to `t`. The default value is `nil`.

Property	**Description**
`checkRESSIZE`	If `preserveRES` is `nil`, prints out `*.RESSIZE` when `checkRESSIZE` is set to `t`. The default is `nil`.
`preserveCAP`	When this property is set to `t`, *Export – CDL* preserves capacitors for checking in LVS. You can define `checkCAPAREA` if `preserveCAP` is `t`. The default is `nil`.
`checkCAPVAL`	Prints out `*.CAPVAL` when set to `t`. The default is `nil`.
`checkCAPAREA`	If `checkCAPVAL` is `nil`, prints out `*.CAPAREA` when `checkCAPAREA` is set to `t`. The default is `nil`.
`preserveDIO`	If `preserveDIO` is set to `t`, *Export – CDL* preserves the diodes for checking in LVS. You can define the variable `checkDIOAREA` if `preserveDIO` is `t`. The default is `nil`.
`checkDIOAREA`	Prints out `*.DIOAREA` when set to `t`. The default is `nil`.
`checkDIOPERI`	Prints out `*.DIOPERI` when set to `t`. The default is `nil`.
`displayPININFO`	When `displayPININFO` is set to `t`, prints out the `*.PINIFO` command for each subcircuit followed by the terminal names and pin directions (input, output, input/Output). The default is `nil`. If the pin information line exceeds the maximum number of characters allowed on a line, each continuation line of pin information is also preceded by `*.PININFO` instead of the usual line continuation character(s).
`preserveALL`	If `preserveAll` is set to `t`, resistors, capacitors, and diodes are preserved for checking in LVS. If `preserveAll` is set to `nil`, resistors, capacitors, and diodes are removed. The default is `nil`.

Note: If you want to use the property `lvsIgnore` equal to `FALSE` on some of the instances of resistors, then you should use the skill variables ***`preserveRES`*** & ***`shortRES`*** as follows:

- ❑ Set the skill variable *`preserveRES`* to `t` by setting the toggle value of `Check Resistors` equal to `True`.
- ❑ Set the value of skill variable *`shortRES`* equal to the maximum value of all resistances below which all the resistors are to be ignored by putting the value in *Resistor Threshold Value* field.

Creating a config view for auCdl

To create a config view:

1. From CIW menu, select *File - New - CellView - Add Lib/Cell/View* to invoke "Hierarchy - Editor"
2. Set Top Cell and choose *Use Template - Spectre*
3. Click OK
   - ❑ New Configuration form opens. Update View list and Stop list to replace `"spectre"` by `"auCdl"`

How to include partial netlist file in SUBCKT calls

You can automatically bind your cells to source files which will then be included in the `.subckt` statements.

Add following in your .simrc file

```
hnlReadHdbProps = 't
ansCdlHdbFilePathProp = "<property name>"
```

Using Hierarchy Editor, add the property to the `lib/cell/view` as cell property in which the netlist needs to be included. In the value field of this property, define full path of the partial netlist file and netlist the config view of the top cell. After the netlist is complete the information is added to the subckt file.

Example

In the given example, you want to include the file `"/tmp/netlist/dummy_top1.net"` inside `LIB5/top1/schematic subckt`. The contents to be added are : `X17 A B / dummytop1`.

The original subckt in the netlist looks like:

```
************************************************************************
* Library Name: LIB5
* Cell Name: top1
* View Name: schematic
************************************************************************

.SUBCKT top1 A B
*.PININFO A:I B:O
XI0 A B / mid

.ENDS
```

In .simrc, set `ansCdlHdbFilePathProp = "abc"`

In the Hierarchy Editor for `LIB5/top1/schematic`, define a property `"abc"` with value `"/tmp/netlist/dummy_top1.net"`

After netlisting the top cell using Hierarchy Editor, the subckt file will read as follows:

```
************************************************************************
* Library Name: LIB5
* Cell Name: top1
* View Name: schematic
************************************************************************

.SUBCKT top1 A B
*.PININFO A:I B:O
XI0 A B / mid

.ENDS
************************************************************************
* This auCdl Netlist has been included for cell top1:
* NOTE: The connectivity in this netlist has not been verified by auCDL
*
X17 A B / dummytop1
*************************************************************************

.ENDS
```

Verification

After adding the property, the Check prop.cfg should read as:

```
cell LIB5.top1
{
string prop abc = "/tmp/netlist/dummy_top1.net"
}
```

Customization Using the .simrc File

The behavior of the netlist can be further controlled using the simulation run control (`.simrc`) file. The parameters that you can include in the `.simrc` file are described in this section. The parameters you can set in the `.simrc` file are the same as those that are defined using the `simSetDef` SKILL function. This SKILL function defines variables only if they have not been defined previously (that is, during initialization when the `si.env` and `.simrc` files are read).

auCdl-Specific Parameters

These auCdl parameters can be set in the `.simrc` file:

Parameter	Description
`auCdlCDFPinCntrl = 't`	Allows CDF `termOrder` to dictate pin ordering of the top-level cell or the cell that has the auCdl view. The default is `'nil`.

`auCdlScale = <m>`
`m = "METER"` or `"MICRON"`
	Prints out `*.SCALE METER` or `*.SCALE MICRON`, accordingly, in the netlist. The default is `"METER"`.

`auCdlCheckLDD = 't`	Turns on LDD device checking by printing `*.LDD` in the netlist. The default is `'nil`.

View List, Stop List, Netlist Type, and Comments

You can use the following variables to define the standard view list, stop list, and netlist type and specify the value of the print comments flag.

Variable	Description
`cdlSimViewList`	A list of views. The default is `'("auCdl" "schematic")`
`cdlSimStopList`	A list of views. The default is `'("auCdl")`
`cdlNetlistType`	Netlist type hierarchical (`'hnl`) or flat (`'fnl`). The default is `'hnl`.
`cdlPrintComments`	Print comments? Yes (`'t`) or no (`'nil`). The default is `'nil`.

The following variables are used for instance-based switch list configuration and also can be set:

```
simInstViewListTable
simInstStopListTable
```

Preserving Devices in the Netlist

The `si.env` file defines the following variables that determine if resistors, capacitors, diodes, or all devices must be preserved in the netlist.

```
preserveRES      preserveCAP
preserveDIO      preserveALL
```

Printing CDL Commands

The following variables let you print the associated CDL commands.

```
checkRESVAL      checkDIOAREA
checkCAPVAL      displayPININFO
checkDIOPERI     shortRES
checkRESSIZE     resistorModel
checkCAPAREA
```

Defining Power Node and Ground Node

You can define `powerNets` and `groundNets` in the `.simrc` file. For example, if you enter the following lines in your `.simrc` file

```
powerNets  = '("VCC!")
groundNets = '("GND!" "gnd!" )
```

the auCdl netlist will show the following line:

```
*.GLOBAL VCC!:P GND!:G gnd!:G
```

Note: You can use the `auCdlSkipMEGA` flag for conditional printing of the `*.MEGA` statement in the auCdl netlist.This flag can be placed in the `.simrc` file, which is read by the netlister.

> The `auCdlSkipMEGA` flag is used as follows:
>
> `auCdlSkipMEGA = ’nil`
> This is the default value. This enables printing of the statement in the netlist.
>
> `auCdlSkipMEGA = ’t`
> When set, the `*.MEGA` statement is not printed in the auCdl netlist.

Evaluating Expressions

You might want to evaluate design variables that have been copied to the cellview using ADE and whose values are needed during verification. The Analog Expression Language mode using which auCdl evaluates expressions is determined by the setting of the SKILL environmental flag `auCdlSetTopLevelEvalMode`. Its valid values are `’t` and `nil`. The default value is `nil` and it causes auCdl to evaluate expressions by using inheritance operators. You can change the mode to full evaluation by setting the value of this flag to `’t`.

For more information on evaluation modes, refer to the Cadence document *Analog Expression Language Reference*.

Mapping Global Pins

In the DFII environment, global signals in a netlist end with a ! character. If you do not want global signals to end with !, you can specify this by using either one of the following methods:

- Click *File – Export – CDL* to open the CDL Out Run form and select the *Map Pin Names from <> to []* option button.
- In the `.simrc` file, set the SKILL environmental variable `pinMap` to `’t`. This is a boolean variable and can have the value `’t` or `’nil`. This variable when set to `’t` uses the following rules to map net names:

```
"+" -> nil
"(" -> nil
")" -> nil
"," -> nil
"/" -> nil
"." -> nil
$" -> nil
"[" -> nil
"]" -> nil
"<" -> "["
">" -> "]"
"!" -> nil
```

The SKILL environmental variable `hnlMapNetInName` can be used similarly. For example:

```
pinMap = ’t
```

is equivalent to:

```
hnlMapNetInName = list(’("+" nil) ’("(" nil) ’(")" nil) ’("," nil) ’("/" nil)
’("." nil) ’("$" nil) ’("[" nil) ’("]" nil) ’("<" "[") ’(">" "]") ’("!" nil) )
```

Black Box Netlisting

The term black box signifies a macro treated as a cell with only an interface definition and no internal details specified. For example, a block to be used by a customer, C, is being designed by a vendor, V. V has formally announced the characteristics of the block and passed on an interface for it to C. C should be able to netlist this block as a black box for initial rounds of verification and plug in the V-supplied netlist, when available, and run a final cycle of verification. This would save C time that would otherwise have been spent waiting for the block. C can specify a property on the master instance of the cell instantiated and the cell will be netlisted as a black box; that is, only the interface of the cell is printed in the netlist and the instances within it are skipped.

The description of the SKILL environment variable flag to enable or disable the feature is:

auCdlDisableBlkBox=‘t	Disables the feature
auCdlDisableBlkBox=‘nil	Enables the feature

The default value of the variable is `‘nil`. This will mean that the black box netlisting feature is enabled, by default.

A boolean property needs to be added on the cellview that is to be treated as a black box. The descriptions of the valid values of this property are:

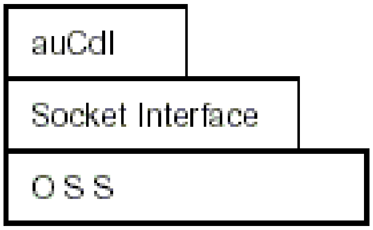
auCdlPrintBlkBox=‘t	Treats the macro as a black box
auCdlPrintBlkBox=‘nil	Treats the macro as is

The steps to be followed to work with this feature are:

1. Ensure that in the `.simrc` file, the SKILL flag has the line `auCdlDisableBlkBox=‘nil`.
2. Specify the cell to be treated as a black box and open the Edit Cellview Properties form. Add the boolean property `auCdlPrintBlkBox` and set its value to `‘t`.
3. Check and save the cellview.
4. Generate the netlist using *File – Export – CDL*.

Note: Set the shell variable CDS_Netlisting_Mode to Analog for auCdl netlisting.

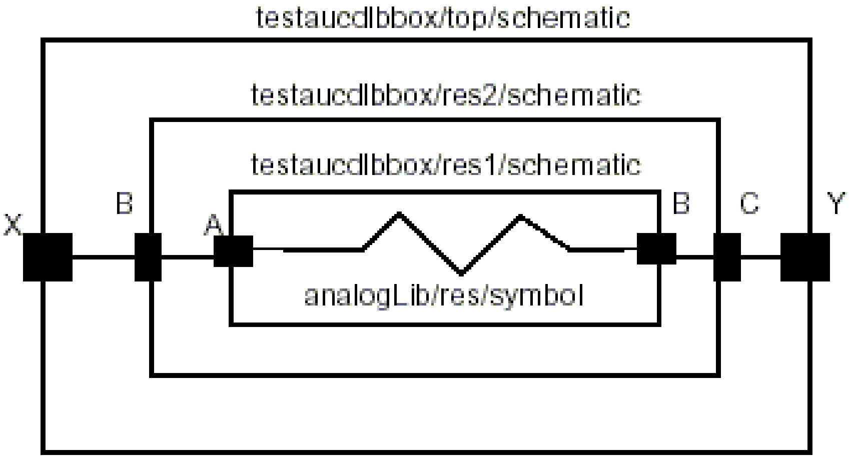
The following figure describes the location of auCdl in DFII with regard to OSS and Socket Interface.



The property to be added on the cellview is a boolean property. Any incorrect property type will be flagged as an error with the following error message in the `si.log` file:

```
Netlister Error: Incorrect property type defined for property “auCdlPrintBlkBox”
on cellview libname/cellname/viewname. The type of the property can only be a
boolean.
```

Sample Hierarchical Cell Using Blackboxing


testaucdlbbox/top/schematic
testaucdlbbox/res2/schematic
testaucdlbbox/res1/schematic
X
B
A
B
C
Y
analogLib/res/symbol

Default Netlist

```
********************************************
* auCdl Netlist:
* Library Name: testaucdlbbox
* Top Cell Name: test
* View Name: schematic
* Netlisted on: Feb 6 16:32:46 2003
********************************************
*.EQUATION
*.SCALE METER
```

```
*.MEGA
.PARAM
********************************************
* Library Name: testaucdlbbox
* Cell Name: res1
* View Name: schematic
********************************************
.SUBCKT res1 A B
*.PININFO A:I B:O
RR0 A B 1K $[RP]
.ENDS
********************************************
* Library Name: testaucdlbbox
* Cell Name: res2
* View Name: schematic
********************************************
.SUBCKT res2 B C
*.PININFO B:I C:O
XI0 B C / res1
.ENDS
********************************************
* Library Name: testaucdlbbox
* Cell Name: test
* View Name: schematic
********************************************
.SUBCKT test X Y
*.PININFO X:O Y:I
XI1 X Y  / res2
.ENDS
********************************************
```

Netlist when auCdlPrintBlkBox=‘t on testaucdlbbox/res2/schematic:

```
********************************************
* auCdl Netlist:
* Library Name: testaucdlbbox
* Top Cell Name: test
* View Name: schematic
* Netlisted on: Feb 5 14:57:30 2003
********************************************
*.EQUATION
```

```
*.SCALE METER
*.MEGA
.PARAM
********************************************
* Library Name: testaucdlbbox
* Cell Name: res1
* View Name: schematic
********************************************
.SUBCKT res1 A B
*.PININFO A:I B:O
RR0 A B 1K $[RP]
.ENDS
********************************************
* Library Name: testaucdlbbox
* Cell Name: res2
* View Name: schematic
********************************************
.SUBCKT res2 B C
*.PININFO B:I C:O
.ENDS
********************************************
* Library Name: testaucdlbbox
* Cell Name: test
* View Name: schematic
********************************************
.SUBCKT test X Y
*.PININFO X:O Y:I
XI1 X Y  / res2
.ENDS
********************************************
```

Notice that the macro `res2` has been generated as a black box with only its interface, that is terminal information, being printed in the netlist. The difference in the netlists is marked in bold typeface.

Additional Customizations

The following sections describe some additional customizations that you can make.

Including a ROM-Insert Netlist Automatically Into the auCdl Netlist

While generating a netlist of the top-level, you might want to include the CDL netlist of instantiated blocks that have a CDL netlist but no schematic.

You can do this by using the `auCdlEnableNetlistInclusion` SKILL flag as follows.

- Set the SKILL environmental variable `auCdlEnableNetlistInclusion` to `’t` in the `.simrc` file. By default, its value is `nil`.
- Create an auCdl view for the instance whose netlist you want included in the top-level netlist. The view can be created by copying over the existing symbol view of the cell as the auCdl view. This view will contain the textual CDL netlist file for the cell.
- Add the property `CDL_NETLIST_FILE` with its `valueType` as `string` and value as the name of the netlist file to be included.
- Ensure that the file is present in the stopping view.

When the netlister is run, this file is included (concatented) in the top-level netlist.

PININFO for Power and Ground Pins

If you want power and ground pin names to appear with `:P` and `:G`, respectively, in the *.PININFO line in the CDLOut netlist for non-global signals, you can specify this with the `cellViewPowerPins` and `cellViewGroundPins` properties.

For example, you may have four pins in the cellView, namely A, B, VSS, and VDD, and you want the PININFO lines to appear as follows:

```
.SUBCKT test A B VDD VSS
*.PININFO B:P VSS:G A:G VDD:P
.ENDS
```

From the schematic cellView, click *Edit – Properties – CellView*. Click *Add* in the *User Property* section and add the following properties:

- `cellViewPowerPins`, with *Type* as `ilList` and *Value* as `("B" "VDD")`
- `cellViewGroundPins`, with *Type* as `ilList` and *Value* as `("A" "VSS")`

Then, check and save the cellView.

When you run the netlister, CDL Out checks for two properties of the type `ilList` in the cellview, namely `cellViewPowerPins` and `cellViewGroundPins`, and generates the netlist according to information specified with them. The PININFO lines in the netlist appear as mentioned above.

Changing the Pin Order

You need to do the following to modify the pin order:

1. In the `SimInfo` section of CDF for the auCdl view, add the following lines to the file.

   ```
   netlistProcedure:   ansCdlSubcktCall
   componentname:      subcircuit
   termOrder:          "my_pin_1" "my_pin_2" "my_pin_3"
   namePrefix:          X
   ```

2. Add the following line to the `.simrc` file:

   ```
   auCdlCDFPinCntrl = t
   ```

   If a `.simrc` file does not exist, you need to create one, add this line, and save it in your home directory.

.PARAM Statement

The design variables specified on Top cell of the design being netlisted will be printed in `.PARAM` statement each per line.

For example, if the `designVarList` property specified on top cell has the following value:

```
( ("CAP" "0.8p") ("RES" "20") ("X" "35") )
```

the `.PARAM` will be printed as:

```
.PARAM CAP=0.8p
+ RES=20
+ X=35
```

However, if no design variables are specified in Top cell properties, .PARAM statement will not be printed in the netlist.

Specifying the Terminal Order for Terminals

The order of terminals in the auCdl netlist can be defined by the termOrder property in CDF. A skill flag `auCdlCDFPinCntrl` is to be set to `'t` to use this feature.

From 5.0.33 onwards, the behavior of termOrder will be consistent with other Artist netlisters, such as Spectre. Minor differences do exist to keep the current behavior of leaf-level cells backward compatible. The new features are as follows:

- If the termOrder is missing, the default terminal list is used to print the netlist for that cell or instance.
- If the termOrder has fewer terminals than the default terminal list, then the terminals are printed in the order specified in the termOrder. For non-leaf level cells, it is followed by the remaining terminals in the default terminal list. For leaf level cells, it is followed by the inherited terminals only, if any.
- If the termOrder has duplicate terminals, a warning message is issued as described in the section "Error Handling" on page 654. For non-leaf level cells, the termOrder is ignored and the default terminal list is used for netlisting. For leaf level cells, the terminals in the termOrder are printed followed by the inherited terminals, if any.
- If a terminal in the termOrder is not valid, a warning message is issued as described in the section "Error Handling" on page 654 and the default terminal list is used for netlisting.

You can also specify any of the following additional existing options to control the terminal order of bus members:

- Ascending order for all bus members
- Descending order for all bus members
- Individual members of a bus to be specified in any order in the termOrder
- Split buses to be specified in any order in the termOrder

The SKILL flag `auCdlTermOrderStr` string is optional with the `auCdlCDFPinCntrl` flag. You set it as `A` or `D`, for ascending or descending order, respectively.

To specify the terminal order:

1. In CIW, click *Tools – CDF – Edit*.
2. Specify the library and cell names.
3. Set *CDF Type* to *Base* and scroll down to the *Simulation Information* section.
4. Click on *Edit*. The simulation Information dialog box appears.
5. Specify *auCdl* against *Choose Simulator*.
6. In the *termOrder* field, enter the terminals in the order in which you want them in the netlist.

7. Click *Apply* and *OK* to close both dialog boxes and to implement changes.
8. For HNL only, set the SKILL flag auCdlCDFPinCntrl to t in the `.simrc` file that is located in the current directory.
9. To print all buses in ascending order, set `auCdlTermOrderStr="A"` in `.simrc`.
10. To print all buses in descending order, set `auCdlTermOrderStr="D"` in `.simrc`.
11. Build the netlist using auCdl.

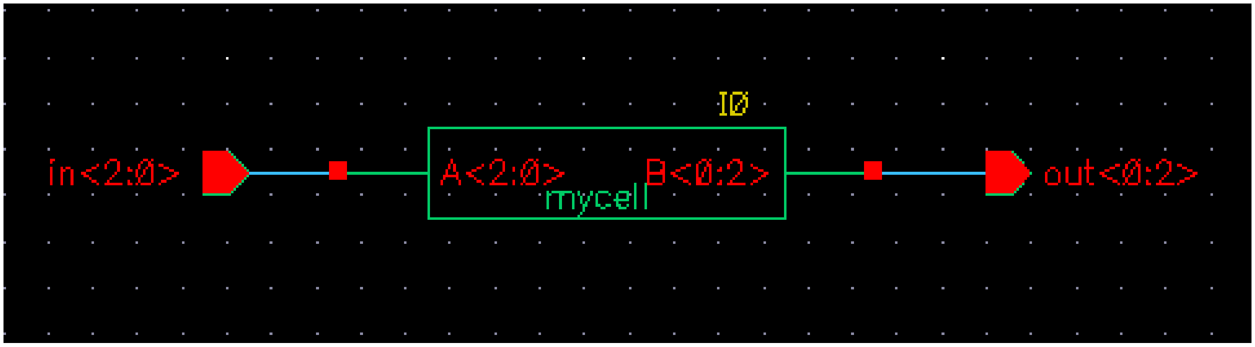
Note: If termOrder is empty, the default terminal list is used.

Note: Set the shell variable `CDS_Netlisting_Mode` to Analog for auCdl netlisting.

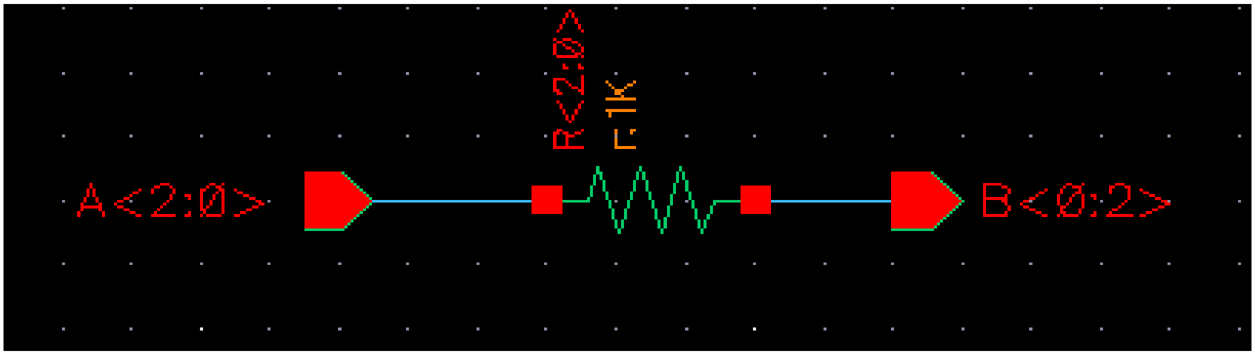
Example

Consider a hierarchical design of the cell `mytop` using `mycell` as a sub-cell. Here, `mycell` has been set as a stopping cell to make the example compact.

Schematic View for mytop



Schematic View for mycell



Assuming that the top schematic is `mytop`, consider the following cases:

Default netlist (No termOrder Is Specified)

```
************************************************************************
* auCdl Netlist:
*
* Library Name: mylib
* Top Cell Name: mytop
* View Name: schematic
* Netlisted on: Apr 10 14:31:28 2003

************************************************************************
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM

************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************

.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<2> in<1> in<0> out<0> out<1> out<2> / mycell
.ENDS
```

Using the CDF termOrder Features

For cases 1, 2 and 3, termOrder is specified as follows:

- For mytop: "in<0:1>" "out<2:1>
- For mycell: "A<1:0>" "B<0:1>"

Case 1: Missing Terminals in termOrder

```
************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************
.SUBCKT mytop in<0> in<1> out<2> out<1> in<2> out<0>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
```

```
XI0 in<1> in<0> out<0> out<1> / mycell
.ENDS
```

Two points to note here are:

- As mytop is not a leaf-level cell, the terminals in the termOrder are followed by the missing terminals in the netlist.
- As mycell is a leaf-level cell, the missing terminals will not be printed in the netlist.

Case 2: When auCdlTermOrderStr="A"

```
************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************
.SUBCKT mytop in<0> in<1> out<1> out<2> in<2> out<0>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<0> in<1> out<0> out<1> / mycell
.ENDS
```

Case 3: When auCdlTermOrderStr="D"

```
************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************
.SUBCKT mytop in<1> in<0> out<2> out<1> in<2> out<0>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<1> out<0> / mycell
.ENDS
```

Case 4: Invalid Terminal

When `auCdlCDFPinCntrl` is set to 't and termOrder for `mytop` is set as:

```
“in<0:1>”  “out<2:1>” “T”
```

TermOrder will be ignored and the default terminal list will be printed for `mytop` along with the warning message.

```
************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************

.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<0> out<1> / mycell
.ENDS
```

`si.log` has the following warning message:

```
*Warning* Could not determine the node name for terminal ’"T"’. This may be
caused by an error in the CDF specified on:
                component   : mytop
                in cellview : schematic
                of library  : mylib
```

Case 5: Duplicate terminal

When `auCdlCDFPinCntrl=’t` and termOrders are set as follows:

- For mytop: "in<0:1>" "out<2:1>" “in<0>”
- For mycell: "A<1:0>" “B<1>” "B<0:1>"

Note the use of individual bus bit “in<0>” and “B<1>” in the termOrder for `mytop` and `mycell`, respectively. When the termOrder is expanded, they become duplicate terminals.

```
************************************************************************
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
************************************************************************
.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<1> out<0> out<1> / mycell
.ENDS
```

The `si.log` file has the following warning message

```
*Warning* Could not determine the node name for terminal ’"in<0>"’. This may
be caused by an error in the CDF specified on:
                component   : mytop
                in cellview : schematic
                of library  : mylib
```

```
*Warning* Could not determine the node name for terminal ’"B<1>"’. This may be
caused by an error in the CDF specified on:
                component   : mycell
                in cellview : schematic
                of library  : mylib
```

The points to be noticed here are:

- As `mytop` is not a leaf-level cell, the termOrder is ignored and the default terminal list for `mytop` is printed in the netlist.
- As `mycell` is a leaf-level cell, duplicate terminals are allowed in the termOrder.

Error Handling

A warning message will be generated in case of invalid/duplicate terminals in the termOrder. The message will include the following information.

```
*Warning* Could not determine the node name for terminal < terminal name>. This
may be caused by an error in the CDF specified on:
           component : <cell name>
           in cellview: <view name>
           of library : <library name>
```

Notification about Net Collision

Sometimes a net name may get mapped to a new name, such as when there are invalid characters in the original name. This new name may collide with another existing or mapped net name. Due to this collision, one of the net names is mapped to a new name.

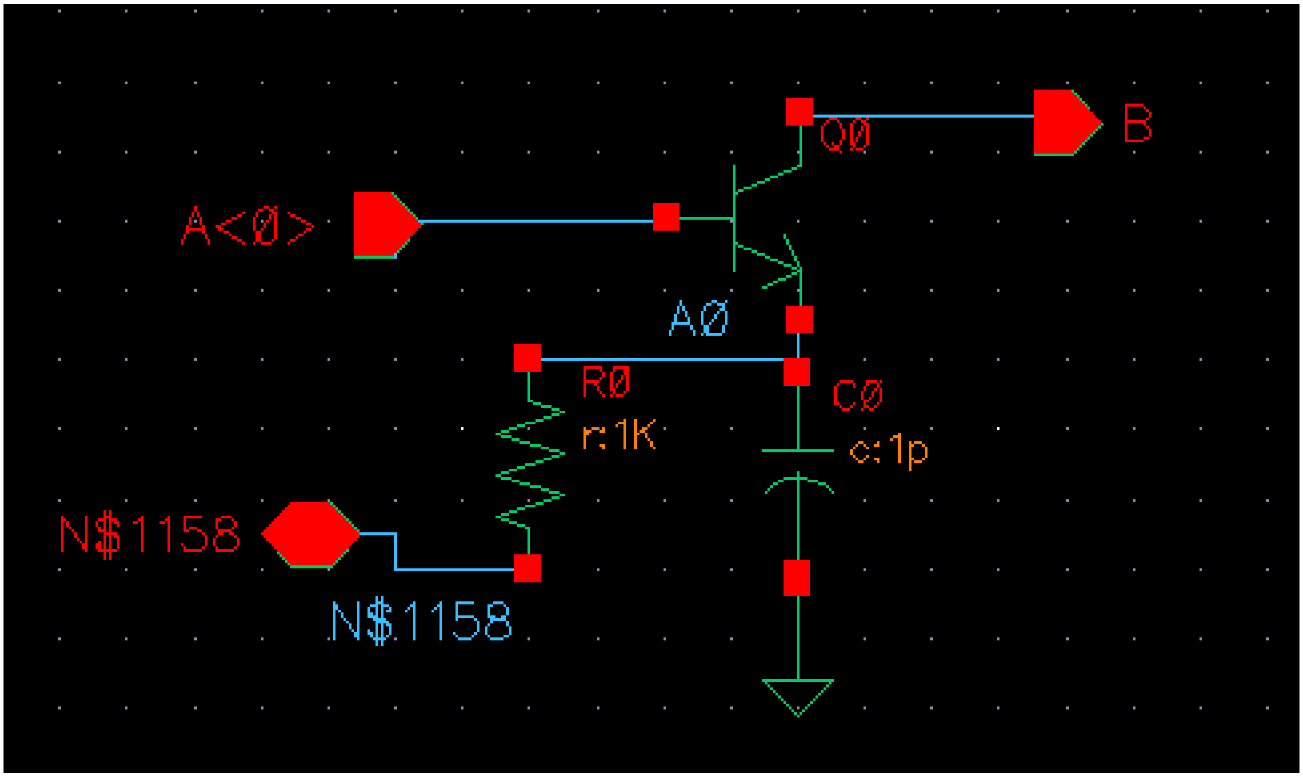
To ensure that you get warnings or error messages for such collisions and mapping, set the SKILL variable `simCheckNetCollisionAction` as per the following table:

Value	**Actions Taken by Netlister**
`warning`	**1.** Generates a warning message for each net name collision: `WARNING: Netlister : Net <netname> has collided with an existing net name, will be remapped to <new name>` **2.** Remaps collided nets. **3.** Creates the netlist.

Value	Actions Taken by Netlister
`error`	**1.** Generates an error message in case of net name collision: `ERROR: Netlister : Net <netname> has collided with an existing net name, exiting...` **2.** Aborts the netlist.
any value other than `warning` or `error`	**1.** Does not generate any warning or error message. **2.** Remaps collided nets. **3.** Creates the netlist.

If you want the `simCheckNetCollisionAction` to operate in the batch mode or the background mode, set it in the `.simrc` file. If you want it to operate in the foreground mode, set it in the CIW.

Consider the following schematic view of the `autest` cell of a hierarchical design `mycdltest`.



Assume that the .simrc file is set as follows:

```
    hnlMapNetInName = ’( ("<" "") (">" ""))
```

```
simNetNamePrefix = "M"
```

The auCdl netlist obtained is as shown below. Note that `A<0>` is mapped to `A0` because the `hnlMapNetInName` variable set in `.simrc`. So, it collides with the original net `A0`. After collision, the original net is mapped to `M0` because `simNetNamePrefix` is set to `M`.

```
************************************************************************
* auCdl Netlist:
*
* Library Name: mycdltest
* Top Cell Name: autest
* View Name: schematic
* Netlisted on: Apr 21 16:12:26 2003
************************************************************************
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM
*.GLOBAL gnd!
*.PIN gnd!
************************************************************************
* Library Name: mycdltest
* Cell Name: autest
* View Name: schematic
************************************************************************
.SUBCKT autest A0 B N$1158
*.PININFO A0:I B:O N$1158:B
RR0 M0 N$1158 1K $[RP]
CC0 M0 gnd! 1p $[CP]
QQ0 B A0 M0 NP
.ENDS
```

Case 1

When `simCheckNetCollisionAction` is set to `warning` and the file `.simrc` has the following settings:

```
hnlMapNetInName = ’( ("<" "") (">" ""))
simNetNamePrefix = "M"
simCheckNetCollisionAction="warning"
```

the netlist generated is the same as mentioned earlier but the log file has the following message:

```
    Running Artist Hierarchical Netlisting ...
    WARNING: Netlister : Net ’A0’ has collided with an existing net name, will be
    remapped to M0.
    End netlisting <Date Time>
```

Case 2

When `simCheckNetCollisionAction` is set to `error` and the file `.simrc` has the following settings:

```
    hnlMapNetInName = ’( ("<" "") (">" ""))
    simNetNamePrefix = "M"
    simCheckNetCollisionAction="error"
```

a netlist is not generated and the log file has the following message:

```
    Running Artist Hierarchical Netlisting ...
    ERROR: Netlister : Net ’A0’ has collided with an existing net name, exiting...
    End netlisting <Date Time >
    "Netlister: There were errors, no netlist was produced."
```

Getting the Netlister to Stop at the Subcircuit Level

To make the netlister stop at the subcircuit level for a specific block (and to prevent it from netlisting down to the primitive cells for the given block), copy the symbol view of the `subckt` to an auCdl view. Then make the following modification to the `.simrc` file:

```
cdlsimViewList = list( "auCdl" "symbol" "schematic" )
cdlsimStopList = list( "auCdl" )
```

Parameter Passing

Parameters can be passed to daughter cells of a subcircuit by passing `m` (M factor) to the MOS transistors that make up an inverter.

on the parent inverter: `m = 2`
on the MOS transistors:

MOS: `m = pPar("m")`
PMOS: `m = pPar("m")`

In the evaluation of a parameter, if the value of another parameter is to be incorporated, then it can be done by using the following method:

- If the parameter AD of a MOS transistor is to be a function of its channel width, AD can be defined as

```
    AD = iPar("w")*5u
```

For more information on passing parameters, see Chapter 3, “Design Variables and Simulation Files”, in the *Analog Artist Simulation Help*.

Netlisting the Area of an npn

To add a CDF parameter called Emiter Area (EA) to the CDF of your `npn`, fill out the CDF form with the following values:

```
paramType = string
parseAsNumber = yes
units = don't use
parseAsCEL = yes
storeDefault = no
name = EA
prompt = EA
defValue = iPar("area")
...
```

If you do not want to display the parameter on the form, you can set `display = nil`.

CDF Simulation Information for auCdl

The auCdl netlisting procedure `ansCdlCompPrim` supports the following devices: `FET`, `CAP`, `IND`, `DIODE`, `BJT`, `RES`, and `MOS`. To use CDL Out to generate the correct name for the component, its terminal, and parameters, you need to attach auCdl CDF simulation information (`siminfo`) to cells. This can be set using *Tools – CDF – Edit* menu commands and then choosing the library/cell.

The `dollarParams` and `dollarEqualParams` fields specify the parameters whose values have to be printed with a dollar ($) prefix.

The parameters specified in the `dollarParams` section are used to print the values of these parameters with a `$` sign prefixed with the value. For example, if the `dollarParams` field contains `param1`, whose value on the instance `L0` of type `inductor` (or its master or the library) is `value1`, then the netlist contains the instance statement as given below

```
    LL0 net1 net2 $value1
```

The parameters specified in the `dollarEqualParams` are used to print the values on the corresponding instance, its master, or its library along with parameters with the `$` prefix. For example, if the `dollarEqualParams` field in the CDF simInfo section contains `param1`, whose value on the instance `L0` of type `inductor` or on its master or the library is `value1`, then the statement for the instance in the netlist is as follows:

```
    LL0 net1 net2 $param1=value1
```

The values for the dollarParams and dollarEqualParams fields use the following precedence: the Instance value overrides the Master value, which overrides the Library value.

To print modelName with a $ sign prefixed to it, add the parameter TSMCMODEL in the instParameters dialog box in the auCdl – simInfo section. The same precedence as specified for the dollarParams and dollarEqualParams fields is used for the model value. For example, if the instance value of TSMCMODEL has a value LP of the type String, then the corresponding instance line in the netlist will contain the model description as:

```
LL0 net1 net2 $.MODEL=LP
```

The following is a comprehensive list of auCdl siminfo for all the supported devices.

Device CDF Values

FET	
netlistProcedure	ansCdlCompPrim
instParameters	W L model
componentName	fet
termOrder	D G S
propMapping	nil W w L l m
namePrefix	j
modelName	NJ
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

CAP	
netlistProcedure	ansCdlCompPrim
instParameters	C L W area SUB m
componentName	cap
termOrder	PLUS MINUS
propMapping	nil C c L l W w area a
namePrefix	C

CAP	
`modelName`	`CP`
`dollarParams`	`param1, param2, param3`
`dollarEqualParams`	`param1, param2, param3`

> **Note:** If you specify any or all of the following: C, area and L & W, the netlister will output only one of them in following priority:
> C
> area
> L & W

IND	
`netlistProcedure`	`ansCdlCompPrim`
`instParameters`	`L tcl tc2 nt ic`
`componentName`	`ind`
`termOrder`	`PLUS MINUS`
`propMapping`	`nil L l`
`namePrefix`	`L`
`modelName`	`LP`
`dollarParams`	`param1, param2, param3`
`dollarEqualParams`	`param1, param2, param3`

DIODE	
`netlistProcedure`	`ansCdlCompPrim`
`instParameters`	`area SUB pj m)`
`componentName`	`diode`
`termOrder`	`PLUS MINUS)`
`propMapping`	`nil`
`namePrefix`	`D`
`modelName`	`DP`

DIODE	
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

BJT	
netlistProcedure	ansCdlCompPrim
instParameters	W L SUB M EA m
componentName	bjt
termOrder	C B E
propMapping	nil EA area
namePrefix	Q
modelName	NP
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

RES	
netlistProcedure	ansCdlCompPrim
instParameters	R SUB W L m
componentName	npolyres
termOrder	P1 P2
propMapping	nil SUB sub R r W w L l
namePrefix	R
modelName	RP

Subcircuits	
netlistProcedure	ansCdlSubcktCall
componentName	subcircuit
termOrder	in out

Subcircuits	
`propMapping`	`nil L l`
`namePrefix`	`X`
`dollarParams`	`param1, param2, param3`
`dollarEqualParams`	`param1, param2, param3`

MOS	
`netlistProcedure`	`ansCdlCompPrim`
`instParameters`	`m L W model LDD NONSWAP`
`componentName`	`mos`
`termOrder`	`D G S progn(bn)`
`propMapping`	`nil L l W w`
`namePrefix`	`M`
`modelName`	
`dollarParams`	`param1, param2, param3`
`dollarEqualParams`	`param1, param2, param3`

Netlist Examples

Here are some netlist examples:

Type	**Example**
Two Terminal CAP	`CC5 n3 gnd! 1p $[CP] M=10`
Three Terminal CAP	`CC32 n5 gnd! 1p $[CP] $SUB=n5 M=3`
Two Terminal RES	`RR8 n2 gnd! 1.2K $[res.mod] $W=4 $L=10 M=3`
Two Terminal IND	`LL1 n1 n3 1000 $[LP] $SUB=gnd!`
Three Terminal RES (4.3.4)	`RR3 n1 n4 1000 $[RP] $W=20 $L=100 $SUB=gnd! M=3`
Three Terminal RES (4.4.x)	`RR3 n1 n4 1000 $SUB=gnd! $[RP] $W=20 $L=100 M=3`

Type	Example
Diode	`DD6 a gnd! DP 10 3 M=12`
FET	`JJ7 d g gnd! fet.mod W=3 L=2 M=2`
BJT	`QQ4 c b gnd! NP M=12 $EA=100 $W=4 $L=3`
MOS	`MM1 g d gnd! gnd! nmos.mod W=3 L=2 M=2`

Note: `auCdl` has been enhanced such that while printing the instance of a cell whose switch master is a stopping view, the `instParameters` specified in the CDF siminfo section are also printed.

Support of Inherited Connection on Device Substrate:

In such situations, the extra terminal (the third terminal on devices like resistors, capacitors etc. or the fourth terminal on devices like transistors) is found on the stopping view rather than the symbol view (instantiated view). So the substrate connection is resolved by finding the net attached to the first extra terminal on the stopping view in comparison to `termOrder` in the CDF.

Note: In case of devices of type MOS, if `progn(bn)` is in the `termOrder`, then precedence would be given to `progn(bn)` and SUB would not be printed at all. Therefore, for MOS devices, in order to use inherited connections on a substrate, you have to remove `progn(bn)` from the `termOrder` of the `siminfo` section of the base CDF of the device.

When you netlist a design that contains inherited connections, it creates dummy ports (or pseudo ports) to maintain connectivity across modules and their instantiations. These dummy ports introduce unwanted nets in the hierarchy and also result in the loss of data when design information is shared across the flow. In the situation when there are some explicit inherited terminals and some explicit inherited terminals which are overridden at the instance level in a macro, use the `$PINS` statement to define the `termOrder`. In the case of inherited connections, you can set the new SKILL variable `simPrintInhConnAttributes` to `t` in the `.simrc` file to prevent the creation of dummy ports. The `*.`NETEXPR and `$netSet` statements can be used in conjunction with the `simPrintInhConnAttributes` variable as follows.

`*.`NETEXPR *`prop_name netName default_signal`*

`$netSet` *`prop_name`* `=` *`value`*

What Is Different in the 4.3 Release

An auCdl netlist can be extracted by following these steps:

1. In the CIW, click on *File – Export – CDL*
2. In the CDL Out Run form, fill in the appropriate fields and click *OK* or *Apply*.

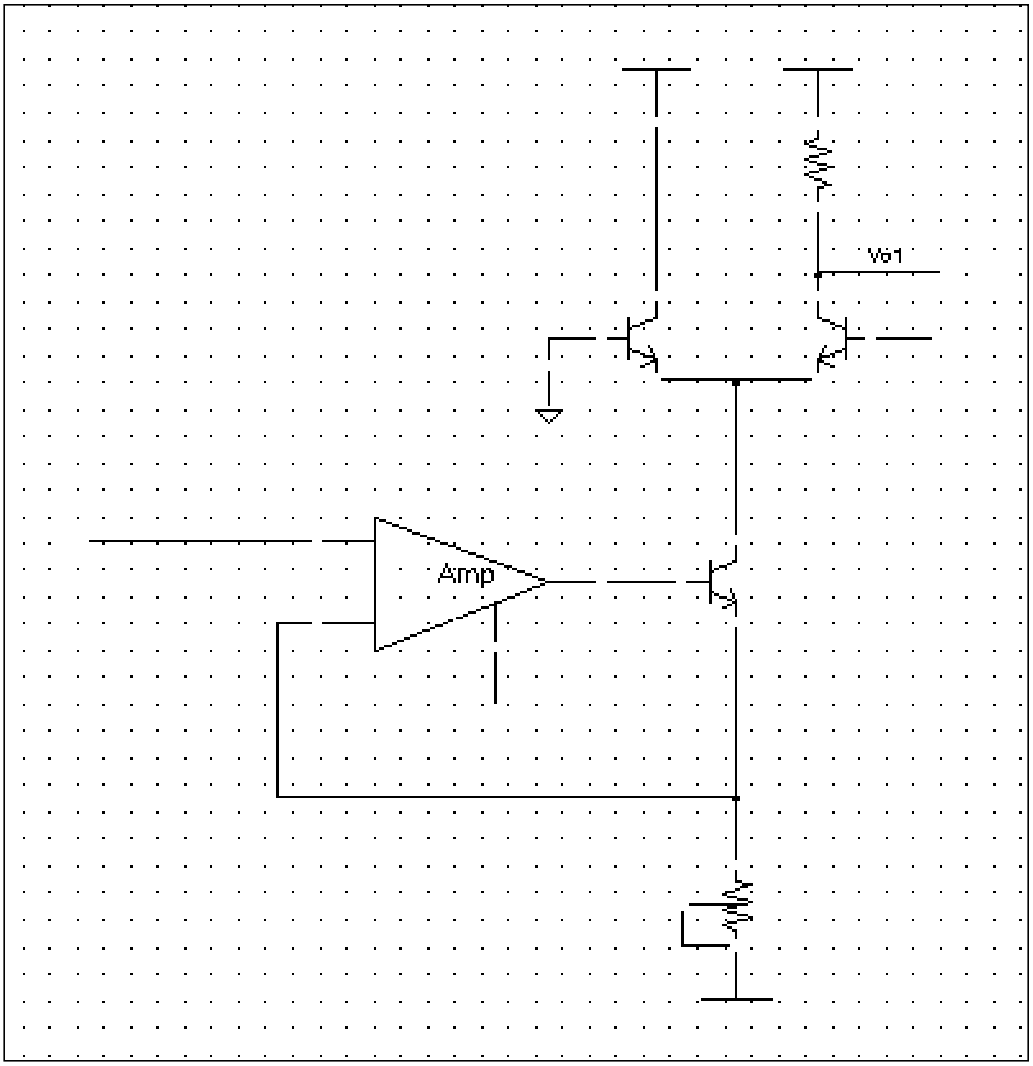
For more information about using CDL Out, read the *Translating CDL Files* section in the *Design Data Translators Reference*.

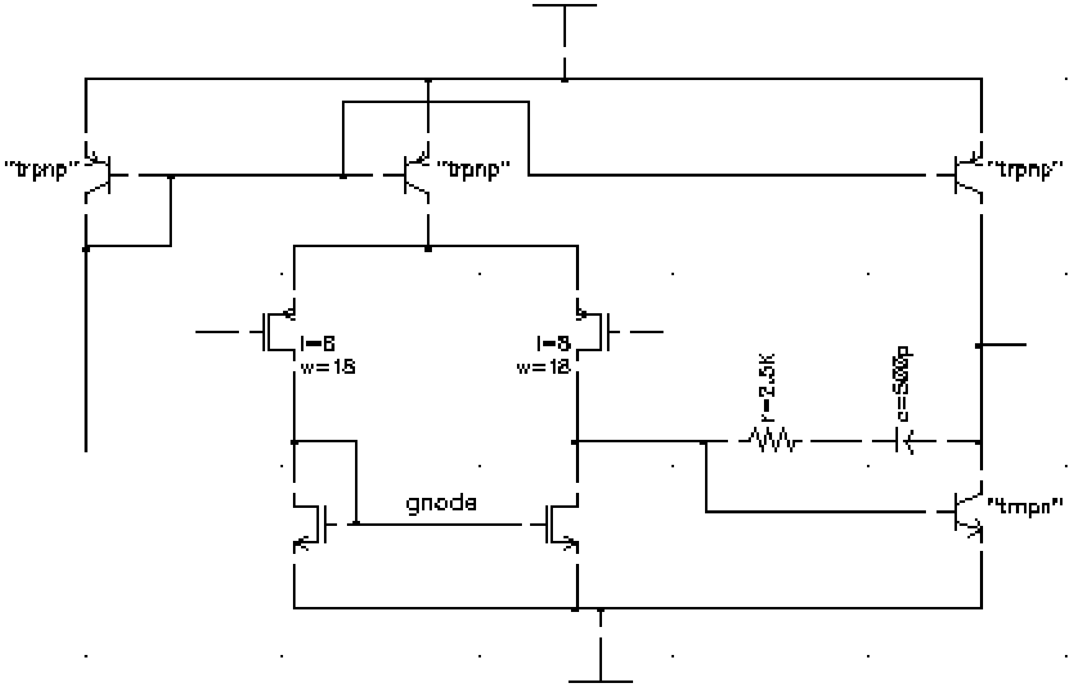
The following `si.env` parameters are used in the 4.3.x release only.

Parameter	Description
`simLibConfigName`	The name of the configuration that determines the versions of cellview used in the design hierarchy. The default is `nil`.
`simVersionName`	Name of the top-level version of the design. The default value is `nil`.
`simLibPath`	Specifies the library search path for the library that contains both the top-level cellview and the global cellview for flat netlisting.

Complete Example

The following example shows the schematic captures and the auCdl netlists.





This is the auCdl netlist.

```
**************************************************************
*auCdl Netlist:
*
* Library Name: test_auCdl
* Top Cell Name: AMckt.auCdlonly
* View Name: schematic
* Netlisted on: Nov 1 16:12:40 1997
**************************************************************
*.BIPOLAR
*.RESI = 2000 resmod
*.RESVAL
*.CAPVAL
*.DIOPERI
*.DIOAREA
*.EQUATION
*.SCALE METER
.PARAM

*.GLOBAL vdd!
+ vss!
+ vcc!
+ vee!
+ gnd!

*.PIN vdd!
*+ vss!
*+ vcc!
*+ vee!
*+ gnd!

**************************************************************
* Library Name: test_auCdl
* Cell Name: amplifier
* View Name: schematic
**************************************************************
.SUBCKT amplifier inm inp iref out
*.PININFO inm:I inp:I iref:I out:O
RR0 net52 net6 2.5K $[RP]
CC0 net6 out CAP $[CP]
QQ0 out net52 vss! NP M=1
MM1 net52 inp net26 vdd! PM W=128e-6 L=8u M=1
MM3 gnode inm net26 vdd! PM W=128u L=8e-6 M=1
MM5 gnode gnode vss! vss! NM W=100u L=10u M=1
MM2 net52 gnode vss! vss! NM W=100u L=10u M=1
QQ4 out iref vdd! PN
QQ2 iref iref vdd! PN
QQ3 net26 iref vdd! PN
.ENDS

**************************************************************
* Library Name: test_auCdl
* Cell Name: AMckt.auCdlonly
* View Name: schematic
**************************************************************

.SUBCKT AMckt.auCdlonly Iref Vlo Vo1 Vs
*.PININFO Iref:I Vlo:I Vo1:O Vs:I
XI3 net28 Vlo Iref net9 / amplifier
QQ2 net15 net9 net28 NP M=1.0
QQ1 vcc! gnd! net15 NP M=1.0
QQ0 Vo1 Vs net15 NP M=1.0
RR0 vcc! Vo1 10e3 $[RP]
RR1 net28 vee! 4e3 $SUB=vee! $[RP]
.ENDS
```

D

Spectre in ADE

New Release Stream

Beginning with 5.1.41 USR1, you have the option of obtaining Spectre from the MMSIM release stream. While Spectre will continue to ship with 5.1.41, new features and most of the bug fixes will be provided exclusively with the MMSIM stream. MMSIM6.0 is being released at the same time as the 5.1.41 USR1 update, but must be downloaded and installed separately. To use these releases together, put the `<`*`path_to_mmsim_release`*`>/tools/bin` directory before any dfII paths in your $path.

The documentation on Spectre's latest features is found in the MMSIM6.0 hierarchy. If you are using the MMSIM6.0 version of Spectre, please access Spectre documentation from that hierarchy. Help buttons from the environment will lead you to the 5.1.41 version of the documentation.

Enhancements

The Spectre® circuit simulator has been enhanced with the following new features:

Improved Parsing and Spice Compatibility

The Spectre parser has been reworked in this release with the goals of increased performance and better compatibility with other Spice simulators. The use of the Spice Pre-Parser (SPP) is no longer required to read Spice netlists or models into the Spectre simulator. Memory consumption when reading in very large netlists has also been dramatically decreased, so that many designs that previously exceeded the memory limits of Spectre can now be simulated.

Due to backward compatibility concerns, this is not the default parser if you use Spectre from the 5.1.41 release. To invoke the new parser in 5.1.41, use the command line option +csfe. If you elect to install the MMSIM6.0 release, you will see the new parser as the default.

To enable the new parser in ADE, insert the following command into your .cdsenv file:

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

Note that this command is needed only when using Spectre from the 5.1.41 stream.

Softshare FLEXlm v10.1 Licensing

All products in the MMSIM 6.0 release, including the Virtuoso® Spectre circuit simulator, have been integrated with Softshare FLEXlm v10.1 licensing (license included with release). This requires you to upgrade your license server to FLEXlm v10.1. The FLEXlm v10.1 license server is backward compatible with older licenses.

FLEXlm v10.1 is part of the Cadence effort to make license management more flexible and easier to use. For more information, refer to http://sourcelink.cadence.com/docs/files/ILS/Install_Lib.html or contact your Cadence representative.

Save/Restart

Save-Restart allows you to save the state of a simulation run at a given timepoint or collection of timepoints, and then use this state to restart the simulation from that point. This can be useful in applications where a circuit demonstrates start-up behavior that you want to simulate past, and then run multiple experiments with varying temperature, parameters, or tolerances. Unlike Spectre’s checkpoint option, this feature supports saving the entire state of the circuit, ensuring accurate transient results upon resuming a stopped run. The checkpoint file is still saved every 30 minutes by default, or when a simulation is interrupted. In a future release we will transition to saving the state file rather than the checkpoint file. For more information, see Recovering from Transient Analyses Terminations in Chapter 9 of the Spectre Circuit Simulator User Guide.

64-Bit Support

A 64-Bit version of Spectre is available in the MMSIM6.0 stream. This is fully compatible with the 32-Bit environment applications. The 64-Bit version of Spectre can be useful in cases where a simulation’s memory needs exceeds the capability of your machine, especially when SpectreRF analyses are being used. In the case of large circuits or large model files, the improvements that were made to Spectre’s parsing for this release may be sufficient. For more information, see Running Spectre in 64-Bit in Chapter 9 of the Spectre Circuit Simulator User Guide.

License Suspend and Resume

You may now direct Spectre to release licenses when suspending a simulation job. This feature is aimed for users of simulation farms, where the licenses in use by a group of lower priority jobs may be needed for a group of higher priority jobs. To enable this feature, simply start Spectre with the +lsuspend command line option. For more information, see Suspending and Resuming Licenses in Chapter 9 of the Spectre Circuit Simulator User Guide.

Enhanced Pole Zero Analysis

A second algorithm has been added to Spectre's Pole-Zero analysis, based on sparse solving techniques. This algorithm is optimized for larger circuits. For more information, see Pole Zero Analysis in Chapter 9 of the Spectre Circuit Simulator User Guide.

Fractional Impedance/Admittance Pole

Fractional Impedance/Admittance Pole (fracpole) allows you to create a passive and frequency-dependent R, L, and C.

New Device Models and Components

The following models are introduced with Spectre in this release:

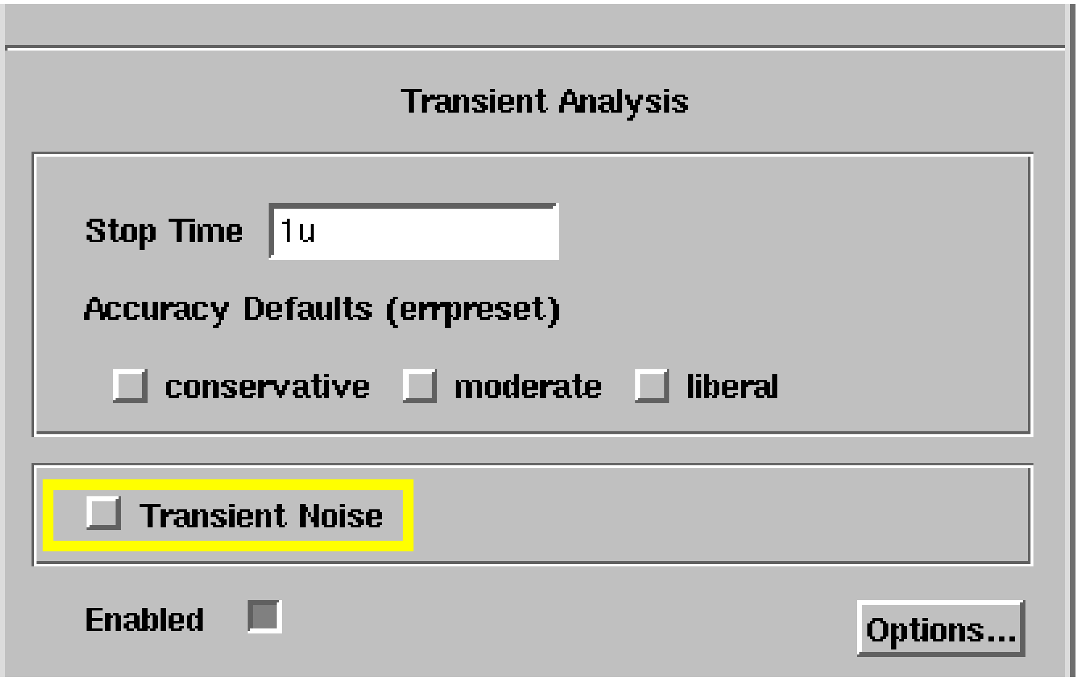
- IBIS component v3.2
- BSIMv4.4
- BSIMSOIv3.2
- Mextram504.4
- MOS1102
- PSITFT improvements - self heating and charge conserving capacitance model

For more information, see the *Cadence Circuit Components and Device Models Manual.*

Transient Noise Analysis

The current transient analysis has been extended to support transient noise analysis. Transient noise provides the benefit of examining the effects of large signal noise on many types of systems. It gives you the opportunity to examine the impact of noise in the time domain on various circuit types without requiring access to the SpectreRF analyses. This capability is accompanied by enhancements to several calculator functions, allowing you to calculate multiple occurrences of measurements such as risetime and overshoot.

Click on the *Transient Noise* button on the main *Transient Analysis* (*Analyses – Choose - tran)* form to enable this feature. For information on how to set up to run an analysis in ADE, see Setting Up a Spectre Analysis.



Transient Analysis

Stop Time 1u

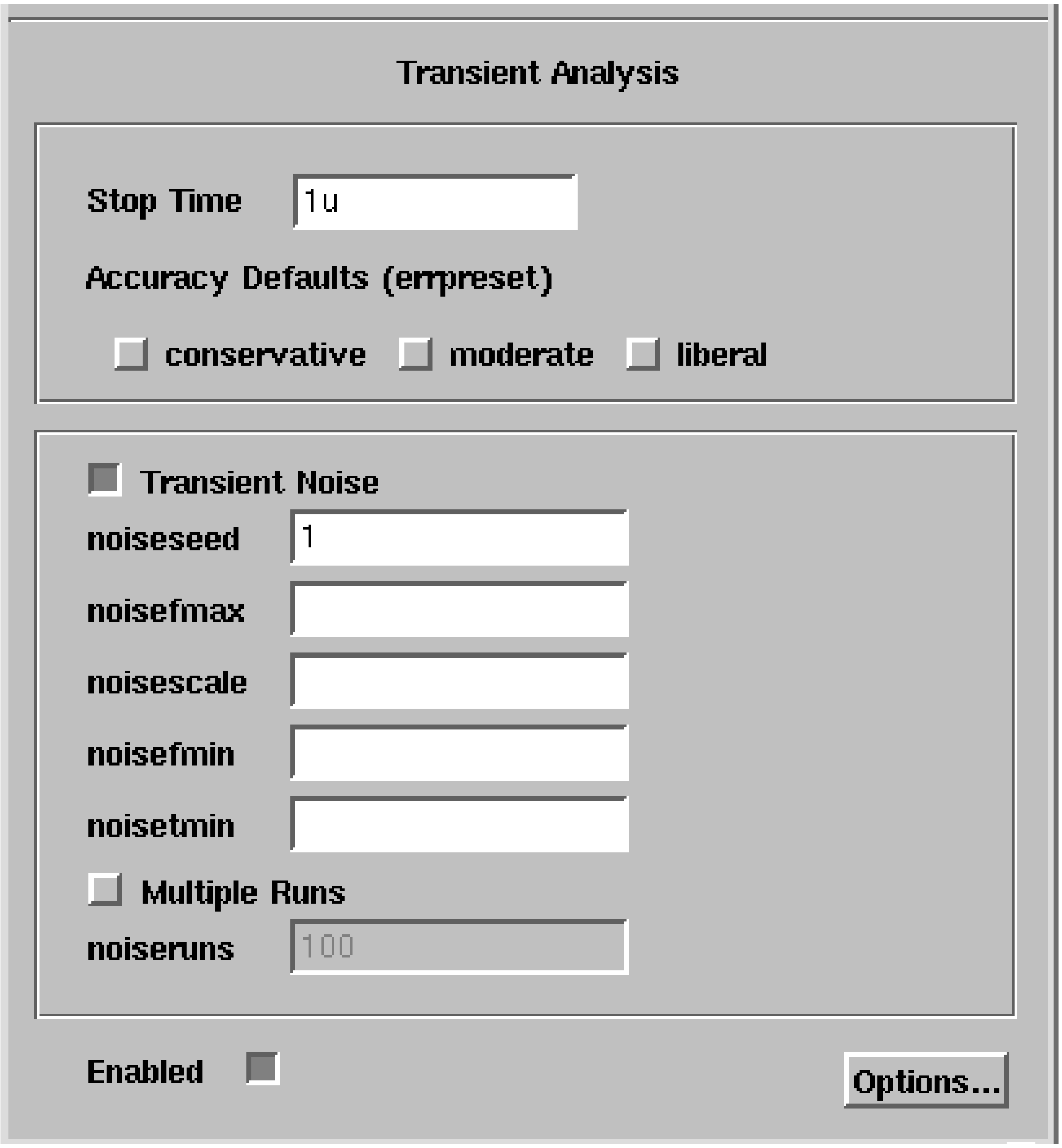
Accuracy Defaults (errpreset)

□ conservative □ moderate □ liberal

□ Transient Noise

Enabled ■

When the *Transient Noise* option is enabled, the *Choosing Analyses* form re-displays to show the transient noise analysis options.



Transient Analysis

Stop Time 1u

Accuracy Defaults (errpreset)

□ conservative □ moderate □ liberal

☑ Transient Noise

noiseseed 1

noisefmax

noisescale

noisefmin

noisetmin

□ Multiple Runs

noiseruns 100

Enabled ☑

Set the following parameters to calculate noise during a transient analysis. (For more detail on the transient noise parameters refer to the *Spectre Circuit Simulator User Guide*).

noiseseed

Seed for the random number generator (used by the simulator to vary the noise sources internally). Specifying the same seed allows you to reproduce a previous experiment. The default value is 1.

noisefmax=0 (Hz)

The bandwidth of pseudorandom noise sources. A valid (nonzero) value turns on the noise sources during transient analysis. The maximal time step of the transient analysis is limited to 1/noisefmax.

noisescale=1

Noise scale factor applied to all generated noise. It can be used to artificially inflate the small noise to make it visible over the transient analysis numerical noise floor, but it should be small enough to maintain the nonlinear operation of the circuit.

noisefmin (Hz)

If specified, the power spectral density of noise sources will depend on frequency in the interval from noisefmin to noisefmax. Below noisefmin, noise power density is constant. The default value is noisefmax, so that only white noise is included and noise sources are evaluated at noisefmax for all models. 1/noisefmin cannot exceed the requested time duration of transient analysis.

noisetmin (s)

Minimum time interval between noise source updates. Default is 1/noisefmax. Smaller values will produce smoother noise signals at the expense of reducing time integration step.

Note: If any of these parameters are not specified, that parameter will not be netlisted.This results in the simulator using its internal default values.

Spectre provides both a single run and multiple run method of simulating transient noise. The single run method, which involves a single transient run over several cycles of operation, is best suited for applications where undesirable start-up behavior is present. The multiple run method, which involves a statistical sweep of several iterations over a single period, is recommended for users who are able to take advantage of distributed processing.

Multiple Runs

Enable this button if you want to perform a multiple-run analysis. When this button is ON, the *noiseruns* field is enabled. You can specify the number of noise runs you want to perform.

In the distributed mode, set up a *transient noise* analysis, specify the waveform expressions in the *Outputs* section of the ADE simulation window, set the *Number of Tasks* in the *Job Submit* form and netlist and run. For details, refer to the Submitting a Job section, of Chapter 2 of the *Virtuoso® Analog Distributed Processing Option User Guide*.

noiseruns

This option will enable you to specify the number of times the transient-noise analysis has to be run. The default for this option is *100* (number of runs). This option has no effect if *Multiple Runs* button is OFF.

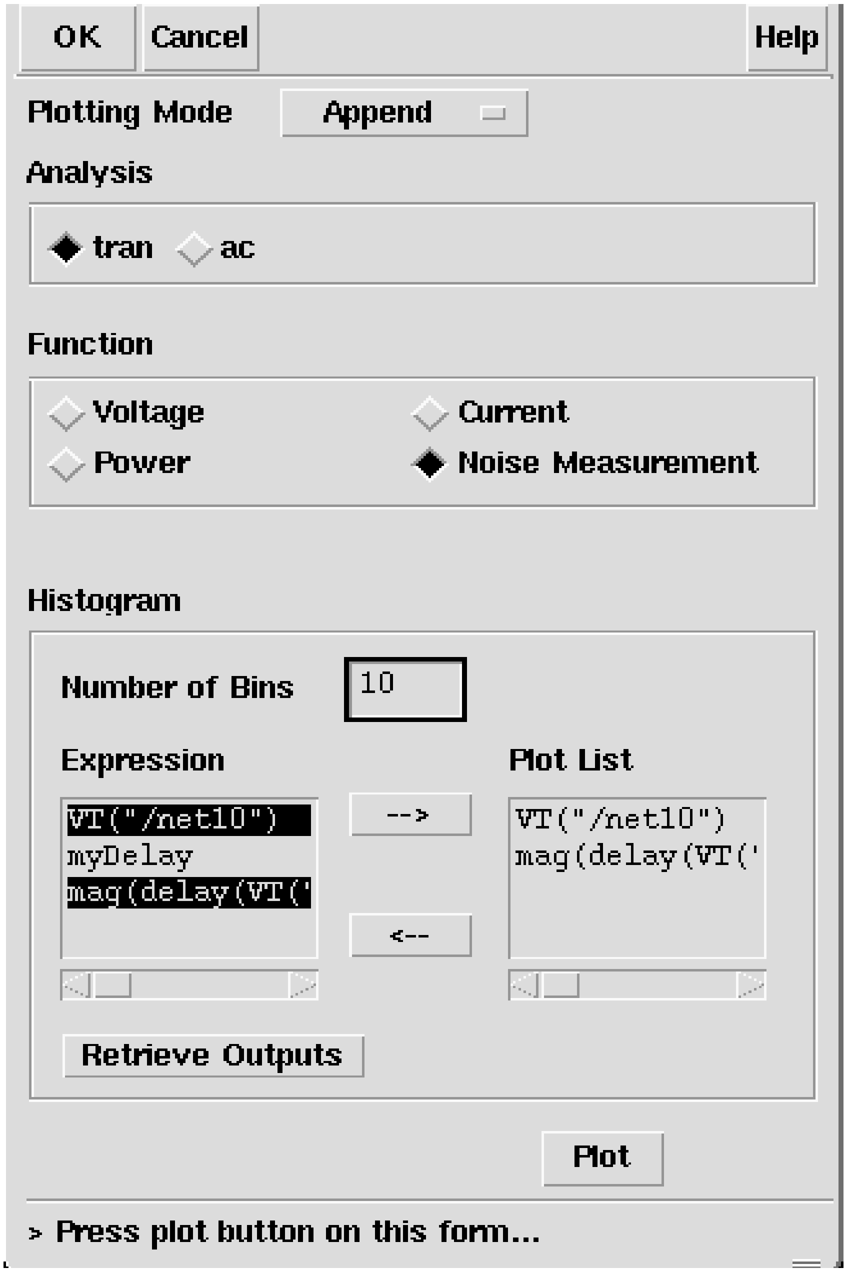
Tip

If you switch from single run analysis to multi run analysis, adjust the *Stop Time* appropriately. For example, specify 5 iterations of 20us for a single run of 100us.

You can specify the options corresponding to transient noise analysis in the *Transient Options* form. Chapter 7 of this user guide describes how to run simulations that you have set up.

Histogram Plots for Transient Noise Analysis

The transient noise analysis is displayed via histogram plots. The *Direct Plot* form corresponding to the transient noise analysis displays a *Histogram* option.



Plotting Mode: Append

Analysis

● tran ○ ac

Function

○ Voltage ○ Current
○ Power ● Noise Measurement

Histogram

Number of Bins: 10

Expression

```
VT("/net10")
myDelay
mag(delay(VT('
```

Plot List

```
VT("/net10")
mag(delay(VT('
```

> Press plot button on this form...

To plot histograms for the selected waveform measurement(s), click on *Plot* button.

Note: You must create the waveform expressions in the ADE window before plotting a histogram for any waveform measurement.

To add specific waveform measurements to the *Plot List*, select the required waveform measurement in the *Expression* list and use the right-arrow button to add it to the *Plot List*

To delete specific waveform measurements from the *Plot List*, select the required waveform measurement in the *Plot List* and use the left-arrow button to put it back in the *Expression* list.

If you create additional waveform expressions (through Calculator) after the *Direct Plot* form has been launched, you can click the *Retrieve Outputs* button to import all the existing waveform expressions into the ADE window.

Note: All normal plots and measurements work as is.

Limitations

- EDFM analyses like Parametric, Monte Carlo and Corners wil not work with the transient noise option.
- This feature does not support all the capabilities (like different modes/styles of histograms and scatter-plots) of Monte Carlo analysis.

E

auLvs Netlisting

This appendix briefly describes Analog LVS or auLvs (Analog and Microwave Layout Versus Schematic) netlister. It is the analog and microwave version of LVS, which originally ran only on digital designs. This information is applicable to any 4.4 or above versions of the Virtuoso® design framework II (DFII).

Using auLvs

You use the auLvs tool for designs that depend on CDF and AEL information and when you use the analog design environment. You can run auLvs inside or outside the DFII environment.

To translate files from the DFII database format into an auLvs netlist, follow the steps below:

1. Set the CDS_Netlisting_Mode variable in the `.cshrc` file to Analog or Compatibility so that the Analog LVS tool (auLvs) is used.The syntax for this variable is

   ```
   setenv CDS_Netlisting_Mode "{Analog|Compatibility}"
   ```

2. Create an auLvs view for the cell by copying the symbol view to the auLvs view.

3. Add the auLvs simulation information to the cell’s CDF.

How to Run auLvs from within DFII

In any version 4.4 DFII or onwards, you can extract an auLvs netlist by following these steps:

1. Open any workbench that supports layouts, for instance icfb, msfb, layout and so on.
2. Open a design window, and click on *Tools – Diva – Verify – LVS*. The LVS form opens.

To know how to run LVS and to understand the outputs generated, see the section *LVS Commands* in *Diva References*.

Customization Using the .simrc File

You can further control the behavior of the netlist by using the simulation run control (`.simrc`) file. The parameters that you can include in the `.simrc` file are the same as those that are defined using the `simSetDef SKILL` function. This SKILL function defines variables only if they have not been defined previously (that is, during initialization when the `si.env` and `.simrc` files are read).

The following auLvs parameter can be set in the `.simrc` file:

Parameter	**Description**
`lvsLimitLinesInOutFile`	You can set this to an integer value, the default being 20. If the file `si.out` contains more than the number specified, the path of the file with the informative message is written into the `si.out` file; otherwise, the contents are written into the *si.out* file.

Related Documentation on auLvs

For information on	**See the following Cadence documents**
Adding CDF information for auLvs	The topic *Adding Component Description Format Simulation Information* in the *Virtuoso® Parasitic Simulation User Guide*
Where the auLvs view (the default stopping view for auLvs) is required in a parasitic simulation	The topic *Adding Component Description Format Simulation Information* in the *Virtuoso® Parasitic Simulation User Guide*
The auLvs SKILL function	The chapter *Netlist Functions* in *Virtuoso® Analog Design Environment SKILL Language Reference*
■ Simulator options applicable to auLvs ■ SKILL expression to set the options for the auLvs simulator ■ Sample CDF parameters for auLvs.	The topic *Modifying Simulation Information* in the *Component Description Format User Guide*

Index