Cadence® Analog Design Environment User Guide

Product Version 5.0
April 2004

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

Contents

Preface

This manual describes how to use the Cadence® analog design environment to simulate analog designs.

The preface discusses the following:

Related Documents

The Cadence® analog design environment is documented in a series of online manuals. The following documents give you more information.

- *Cadence® Advanced Analysis Tools User Guide* gives information about Monte Carlo, optimization, and statistical analysis.
- *Analog Waveform User Guide* describes how to use the waveform display tool.
- *Cadence Distributed Processing User Guide* describes how to use multiple hosts to distribute simulations between a collection of different machines.
- *Cadence® Analog Mixed-Signal Simulation Interface Option User Guide* gives information about how to set up and run mixed-signal simulations.
- *SpectreRF Help* describes how to use the RF option.
- *Spectre Circuit Simulator Reference* and *Spectre Circuit Simulator User Guide* describe the Cadence® analog circuit simulator in detail.
- *Cadence® Parasitic Simulation User Guide* describes how to analyze parasitics.
- *Waveform Calculator User Guide* gives detailed information about using the waveform calculator.
- *Inherited Connections Flow Guide* describes how to use inherited connections and net expressions with various Cadence® tools in the design flow.

- *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.

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 Cadence® Analog Design Environment

This chapter describes the features of the Cadence® analog design environment. This is an overview. Detailed information is available in later chapters.

Consistent User Interface

The Cadence® 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 Cadence® GDSII Stream formats. Simulators can be integrated using a programmable netlister and a general waveform processor.

Analog Design Entry

You enter designs into the Cadence® analog design environment using a hierarchical schematic editor. This editor uses a set of simulation environment commands in combination with the analog library `analogLib`.

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 the analog circuit 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.

Annotation

The analog circuit 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

Th analog circuit design environment includes an enhanced analog waveform display package, which features

- Waveform output during simulation (marching waves)
- 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.

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. You can display the results of the analysis as charts or different types of curves, depending on the values assigned to the axis and plotting parameters.

2

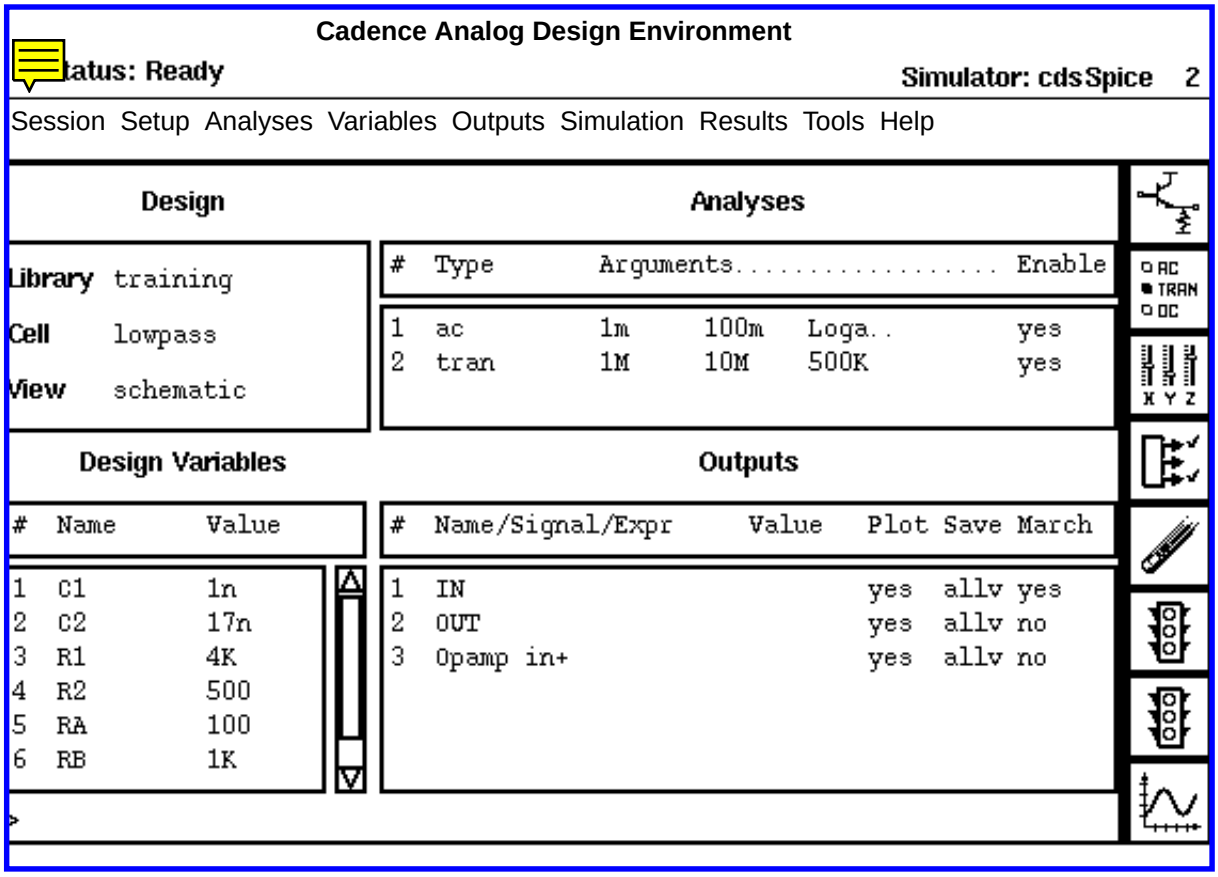
Environment Setup

This chapter describes the features of the Cadence® analog design environment and tells you how to set these features during your sessions. Also this chapter describes how you use Cadence simulators and third-party simulators in the analog circuit design environment.

- What Are Bindkeys? on page 60
- Form Field Descriptions on page 64

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 *Cadence® Analog Design Environment SKILL Language Reference* for instructions on customizing any of the banner menus.



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*.

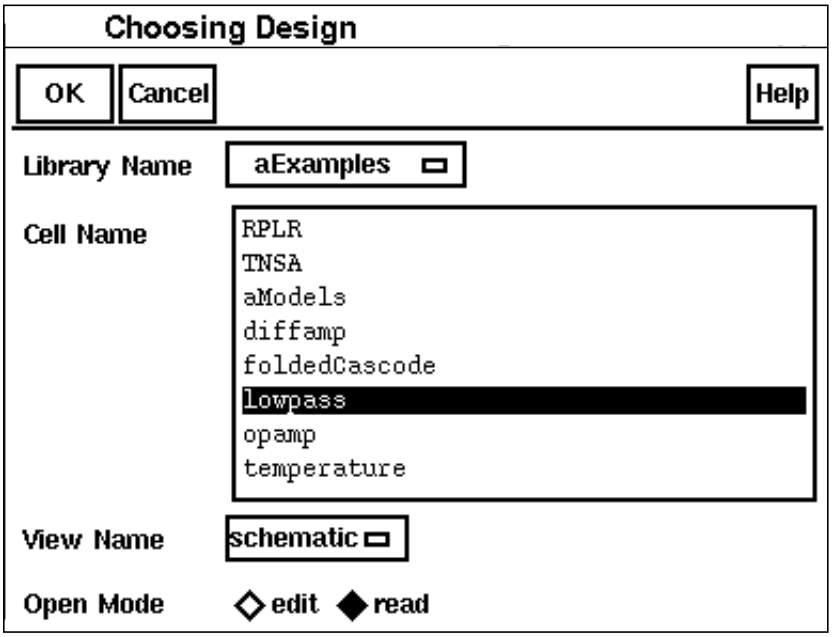
The Simulation window opens and the simulation environment is initialized.

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.



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

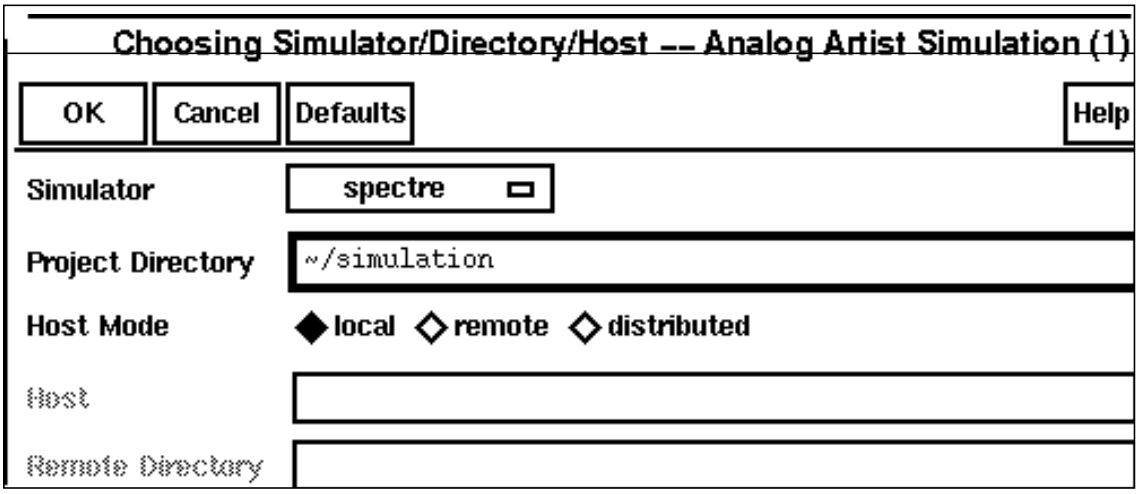
Note: To open a selected design in a different mode, you have to first re-set the session using the option, *Session -> Reset*.

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.



For detailed information about the form, see “Choosing Simulator/Directory/Host” on page 64.

2. Choose a simulator.
3. Check the path in *Project Directory* for simulation data, and change it if necessary.

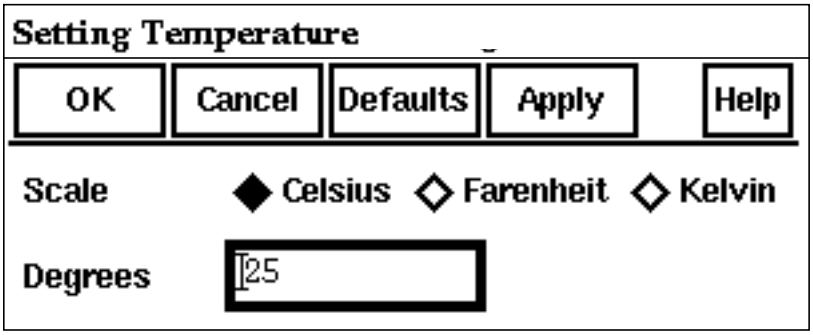
Note: When the *Host Mode* is *Distributed*, the *Choosing Simulator/Directory/Host* form re-displays to show the *Check setup* and the *Stop setup check* buttons. For details, refer to the Setup Requirements section in chapter 2 of the *Cadence® Analog Distributed Processing Option User Guide*.

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.



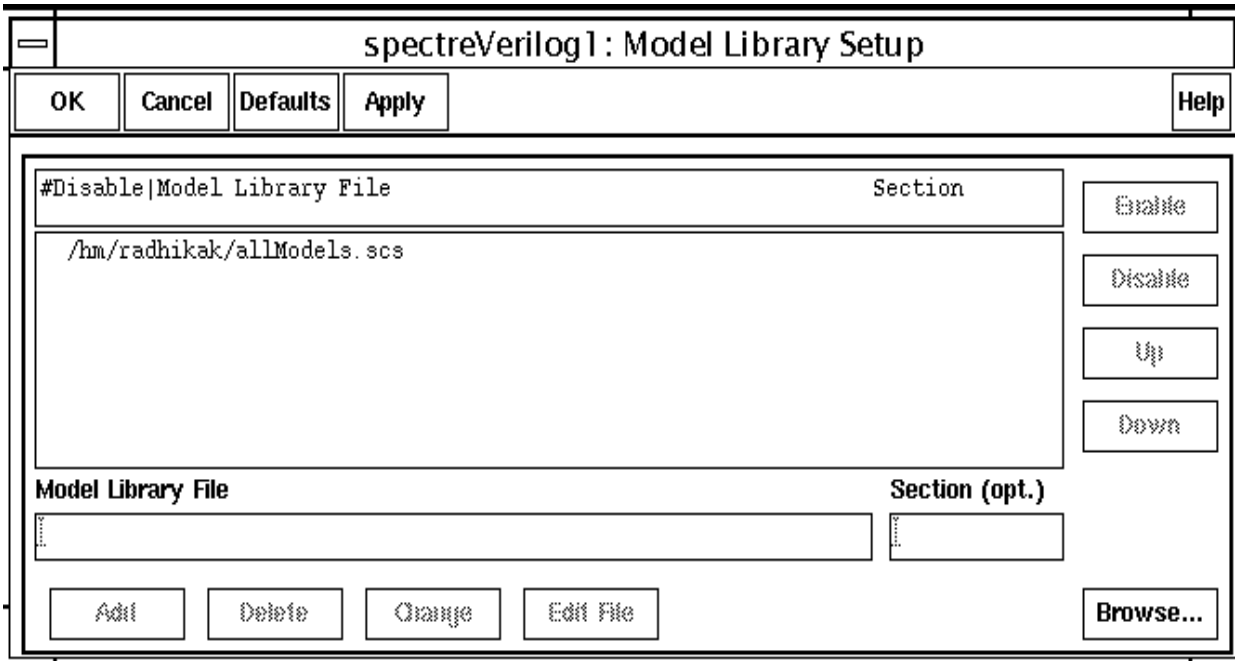
2. Choose the units you want to use for temperature.
3. Type a value in degrees, and click *OK*.

Model Setup for Direct Simulation

To set up models for the Spectre® circuit simulator direct interface and other simulator interfaces that use the direct simulation approach,

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

The Model Library Setup form appears.



For detailed information about the form, see "Model Library Setup" on page 67.

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 Cadence® 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. (optional) Type in a section in the *Section* field.

   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*.

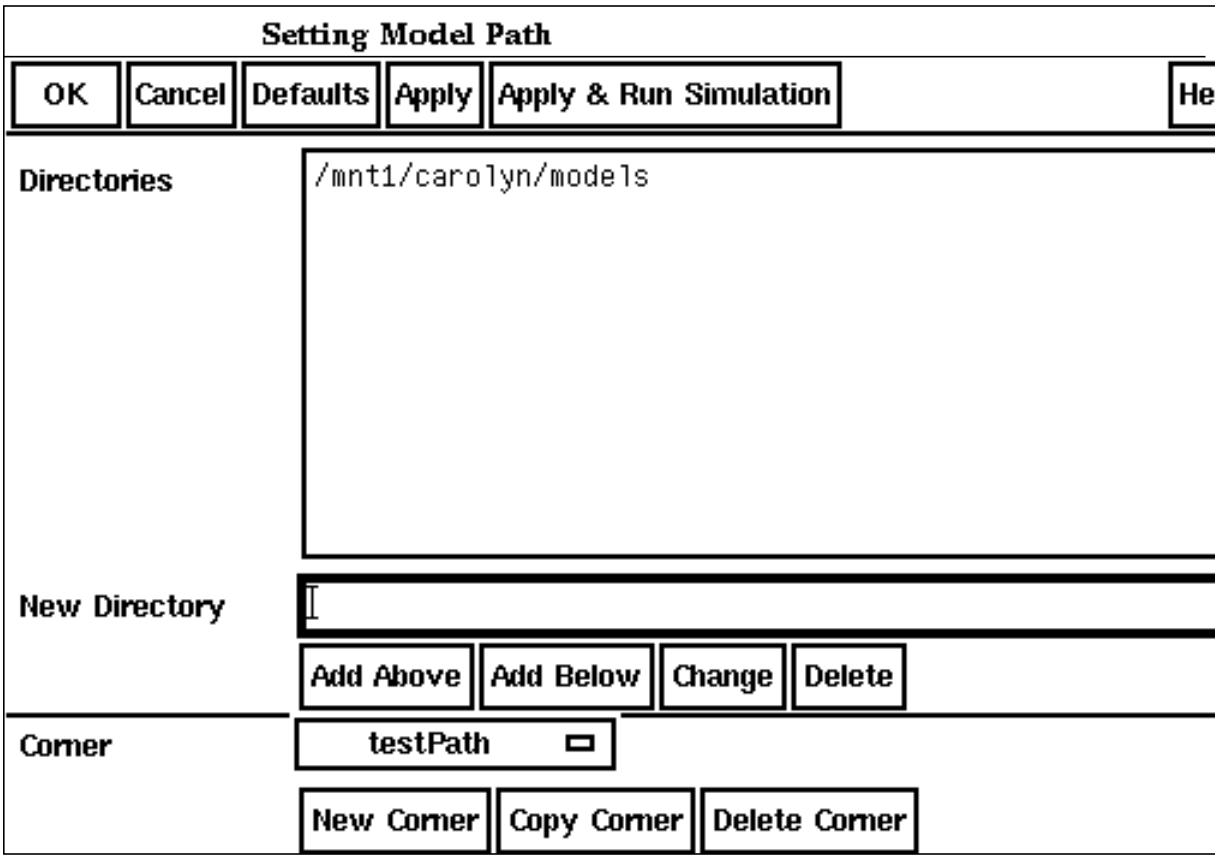
Setting the Model Path for Socket Simulations

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



For detailed information about the form, see "Setting Model Path" on page 66.

A model path is a list of directories that is searched in sequence to locate models for instances in your design. 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.

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 54.

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.

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 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. See the *Compatibility Guide* for details of this effort.

For more information about Spectre device models and analysis modes, see the *Spectre Circuit Simulator Reference* manual.

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*.

Interfaces to Other Simulators

In addition to the Cadence SPICE and the Spectre circuit simulators, you can use other popular analog simulators through a set of integrated simulation interfaces.

The following simulator interfaces are available from Cadence or directly from third-party vendors:

- Meta Software's HSPICE circuit simulator
- Sonnet Software's Em electromagnetic simulator

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.

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

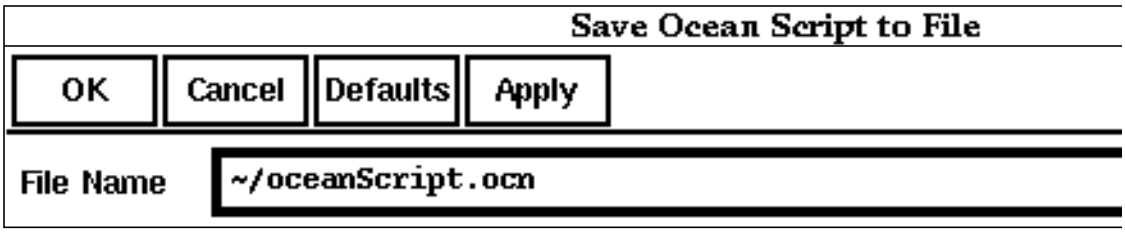
- Create scripts that you can run repeatedly to verify circuit performance
- Run longer analyses such as parametric analyses, corners analyses, and Monte Carlo simulation, more effectively
- Run long simulations in OCEAN without starting the analog circuit design environment graphical user interface
- 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.



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.

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

Overview of the HSPICE Interface

Meta Software's 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 Options" on page 246.

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

To run the HSPICE simulator outside of the analog circuit design environment,

1. Set up the simulation in the Cadence® analog design environment.

   Be sure to set correctly the HSPICE options that are specific to the Cadence® 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 Cadence® 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 circuit design environment needs to read the HSPICE PSF data.

Overview of the Spectre Interface

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 243
- “Setting Up a Spectre Analysis” on page 168

For information about Spectre itself, 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 Cadence® Analog Design Environment

To run the Spectre simulator outside of the Cadence® 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 Cadence® Analog Design Environment

To view standalone Spectre simulation results in the analog circuit 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.

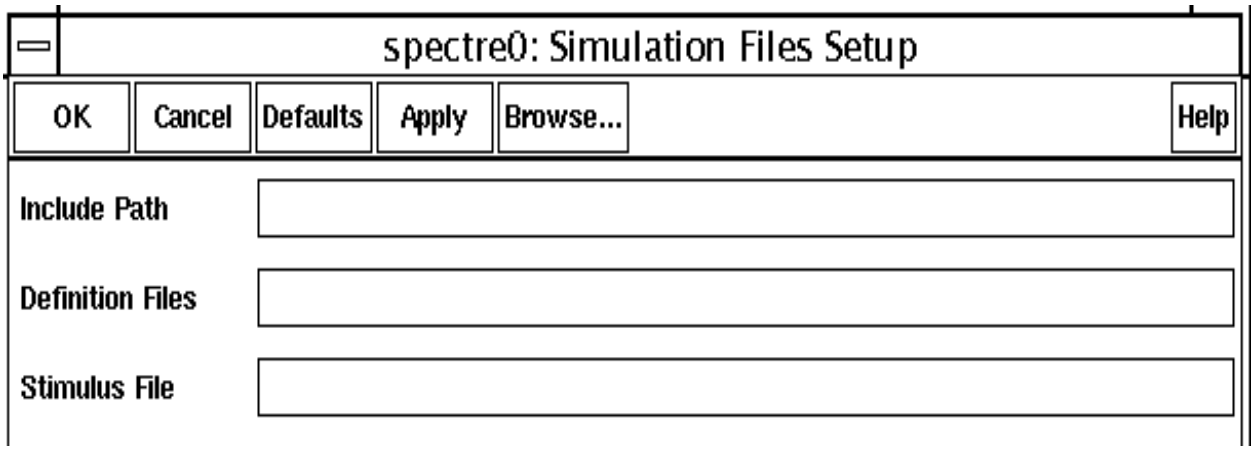
The system creates the data structure the analog circuit design environment needs to read the Spectre PSF data.

Setting Up Simulation Files for Direct Simulation

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

1. Choose *Setup – Simulation Files*.

   The Simulation Files Setup form appears.



2. Fill out the *Include Path* field.

   Use this field for relative filenames. 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.

   To set up a simulation of the `lowpass` design (in the `aExamples` library), in the form above, use the file `bipolar.scs`. It is included from the directory `~/cds/tools/dfII/samples/artist/models/spectre`.

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. 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.

Caution

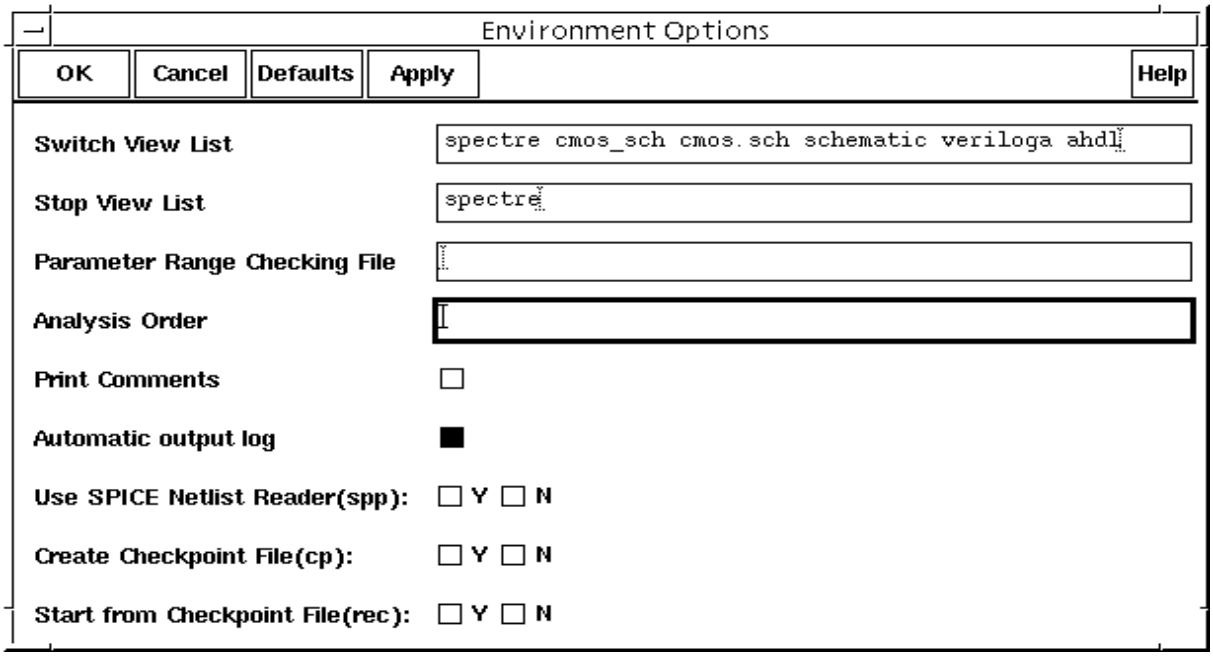
Relatives file paths are not tested for when appending new values to a field that accepts multiple file paths.

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.



For detailed information about the form, see "Environment Options" on page 69.

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.

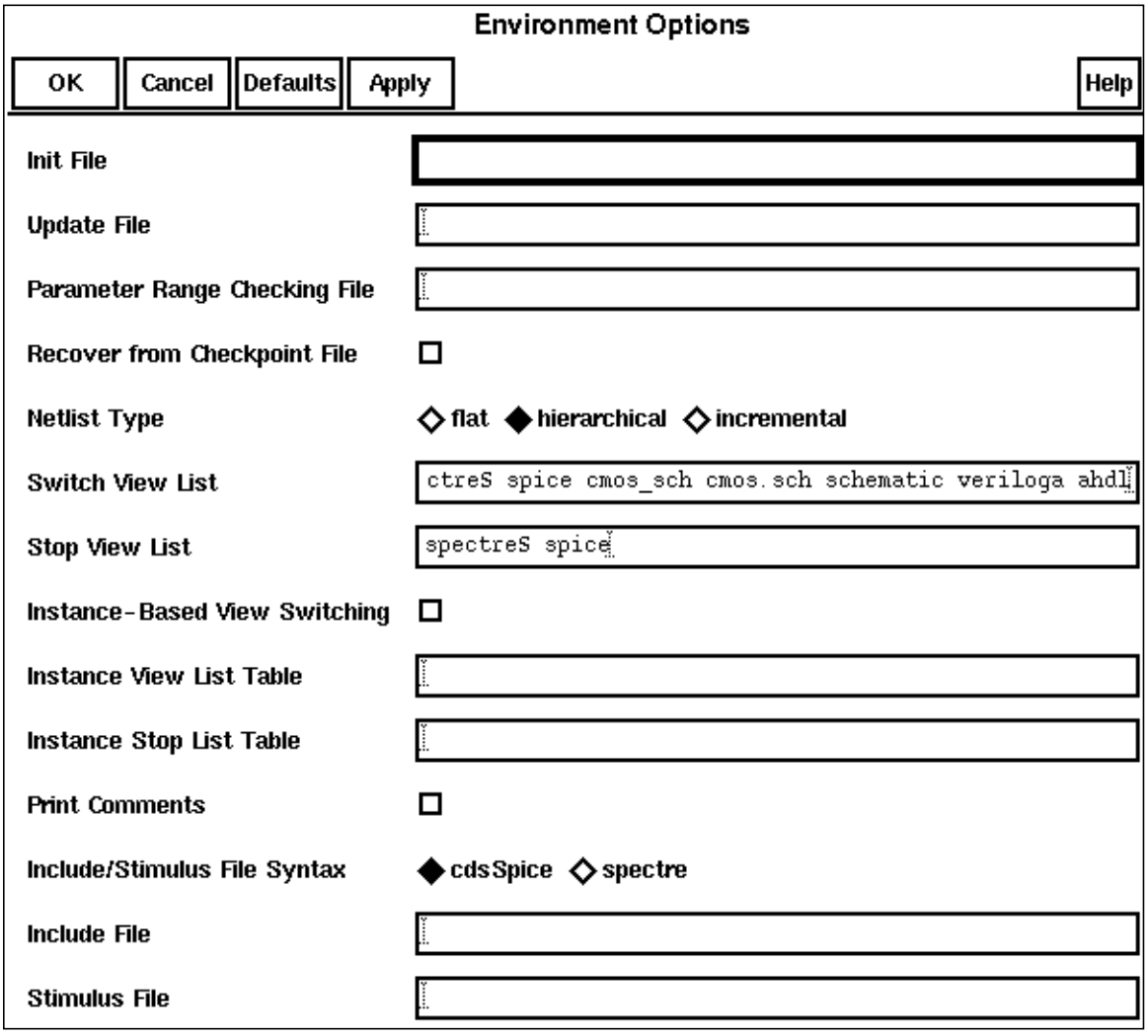
4. Set other options as needed, and click *OK*.

Setting 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.



   For detailed information about the form, see “Environment Options” on page 69.

   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.

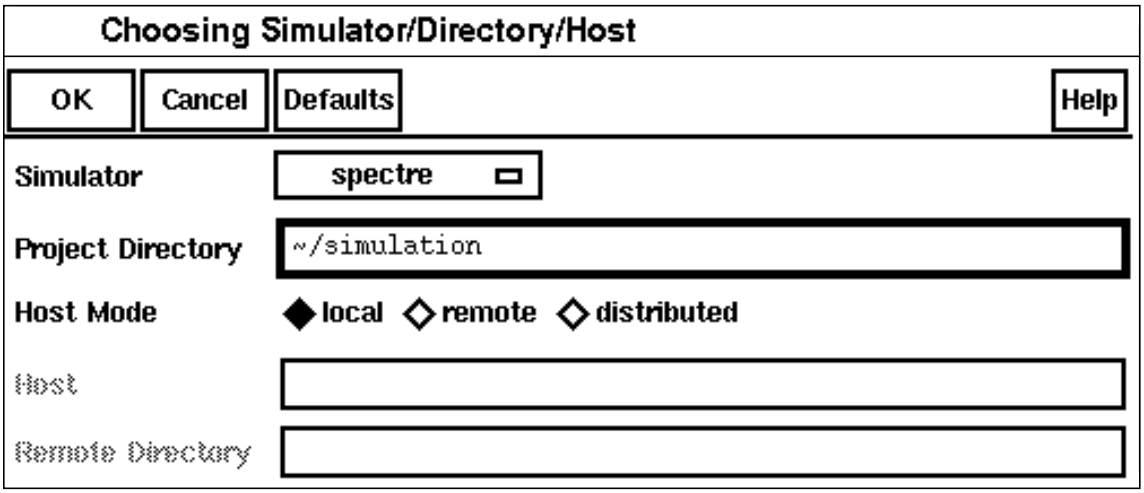
3. (Optional) If you are not using a config view, check and set the options for view switching to control how the system netlists hierarchical designs.
4. Choose flat or incremental hierarchical netlisting.
5. Set other options as needed, and click *OK*.

Setting Up a Remote Simulation

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

1. In the Simulator or Schematic window, 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 executablesshould be in path and LICENCE is 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.

Note: Marching waveform display is not available when you run a remote simulation with this procedure.

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 circuit design environment reads these PSF files for waveforms and backannotation.

About the Simulation Environment

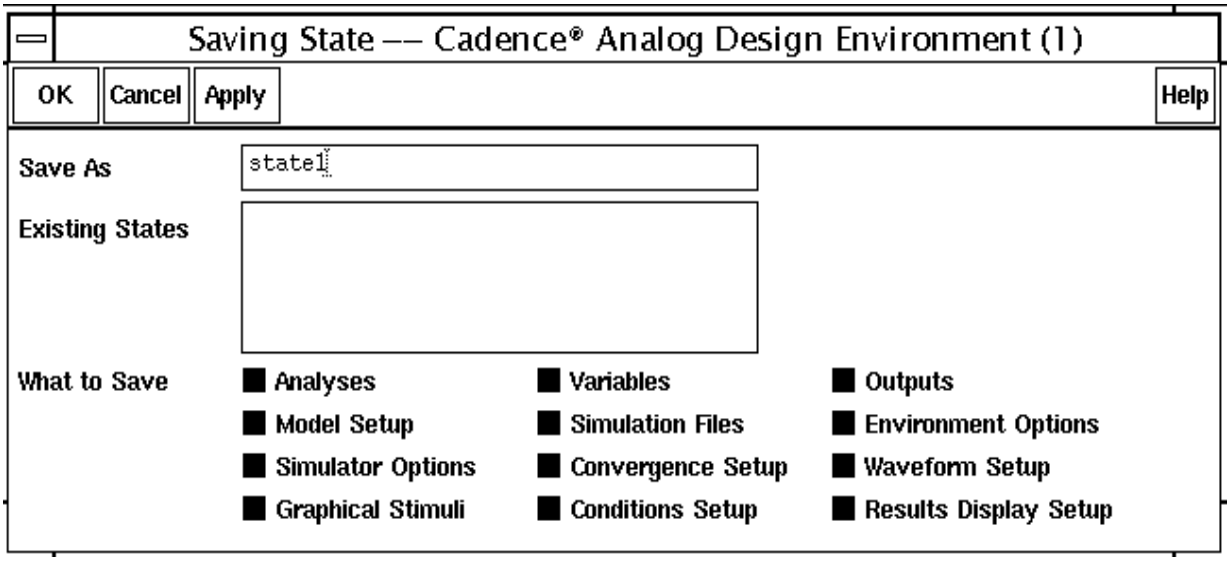
Configuration of the Cadence® analog design simulation environment depends on several kinds of configuration settings:

- 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 *Window – Save* command in the Waveform window.

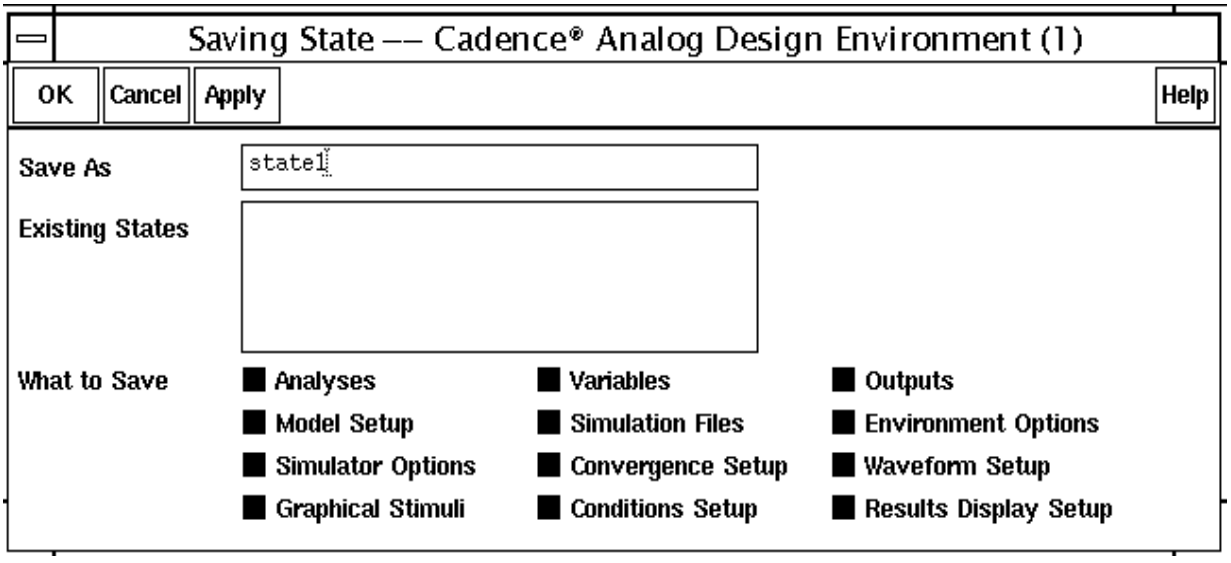
Saved states are simulator dependent for analyses, simulator options, and convergence setup (if the convergence commands you saved are not supported by the other simulator). You can restore saved states from different simulators. The analog circuit design environment restores as much as possible despite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



   For detailed information about the form, see "Saving State" on page 73.

3. Select one of the existing states in the *Existing States* listbox. As a result, the *Save As* field displays the name of the selected state. The substates displayed in the *What to Save* section are also enabled or disabled accordingly (ie, according to the substates in the selected state). Additionally, you can also enable or disable any of the substates and

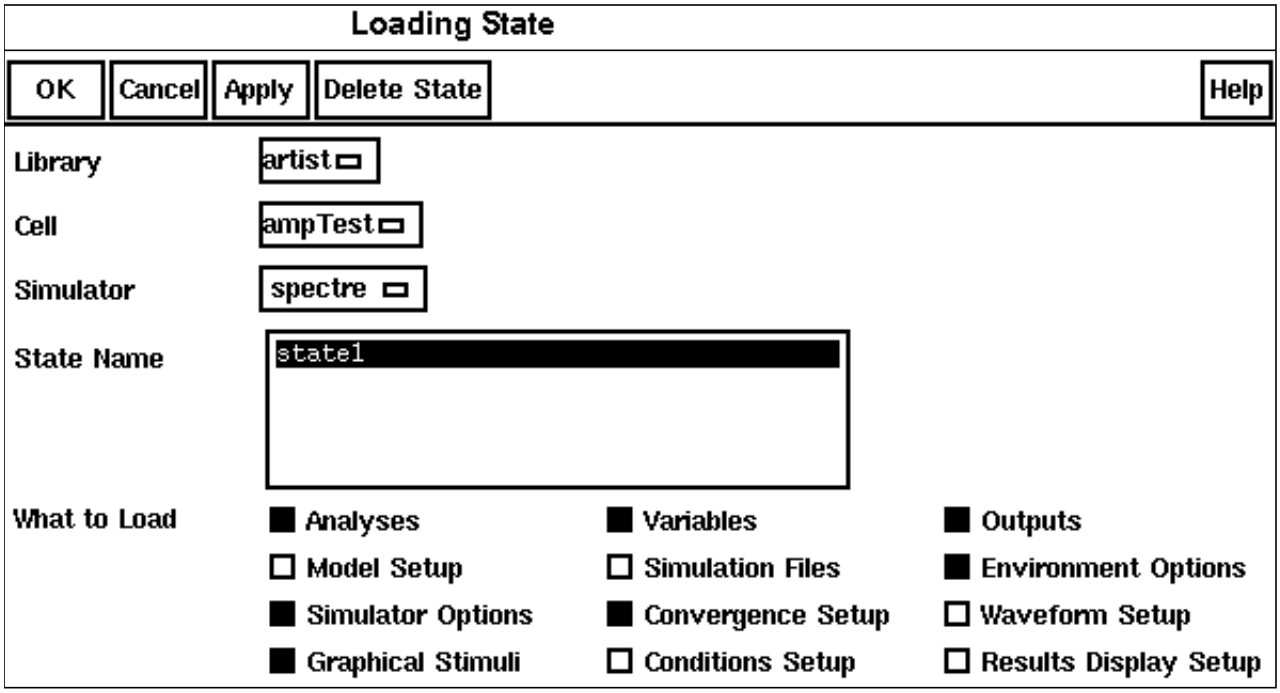
click *OK* to overwrite the existing state. You can also type in a new state name in the *Save As* field to save the changes in a new state.

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.

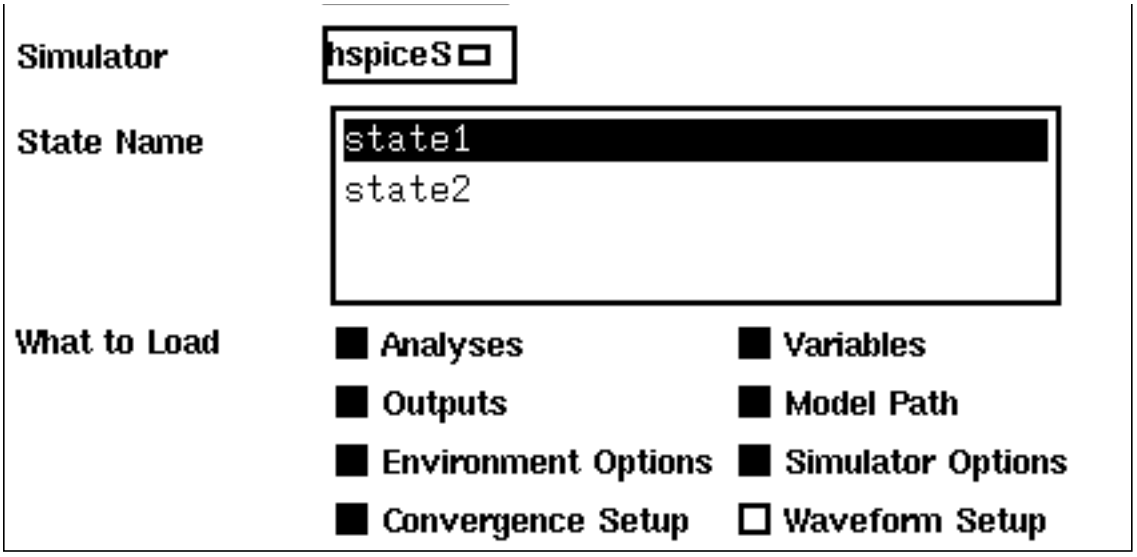


   For detailed information about the form, see “Loading State” on page 74.

2. Choose the desired value in the *Library*, *Cell*, and *Simulator* fields.

   State Name specifies all the saved states for the cell and simulator combination that you specified.

3. Select the state name you want to restore.



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

   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.

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*.

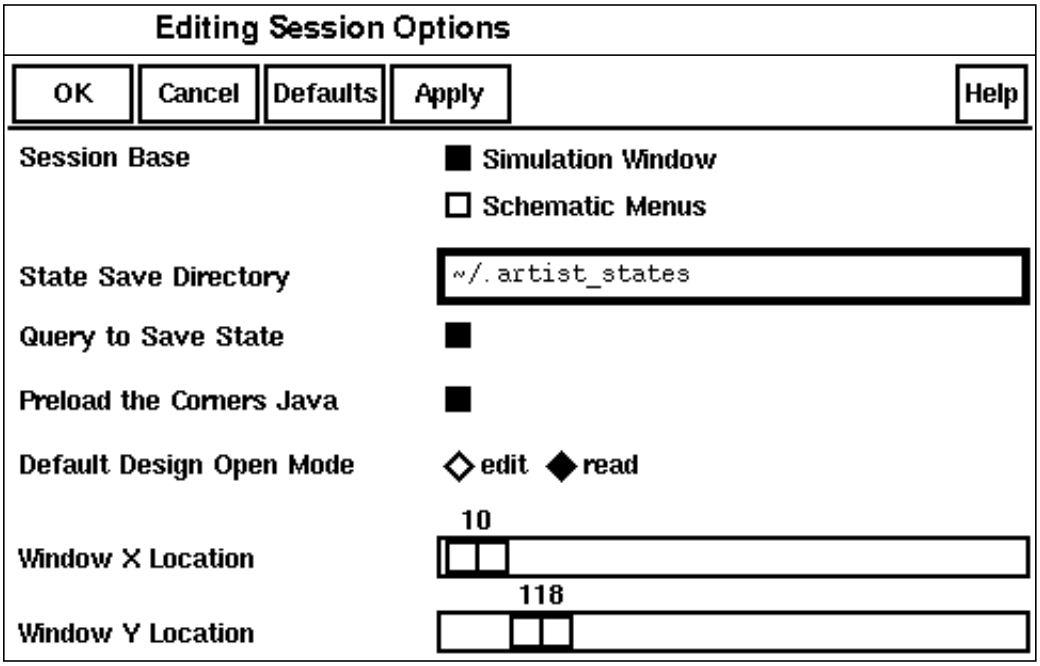
This command restores the original defaults. If you have data in this form, it is overwritten with default settings. Use the *Session – Save* command to save this information before resetting the defaults.

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.



   For detailed information about the form, see "Editing Session Options" on page 75.

Note: Changes take effect the next time you start the analog circuit 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 Cadence® 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 Cadence® 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.

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 Cadence® 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.

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 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.

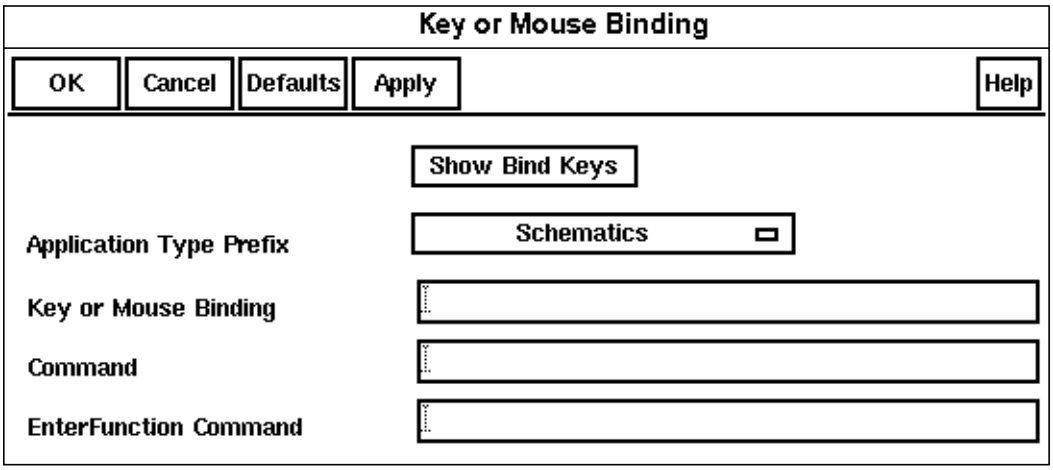
For more information about setting bindkeys, see the *Design Framework II Help*.

Checking Bindkey Assignments

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

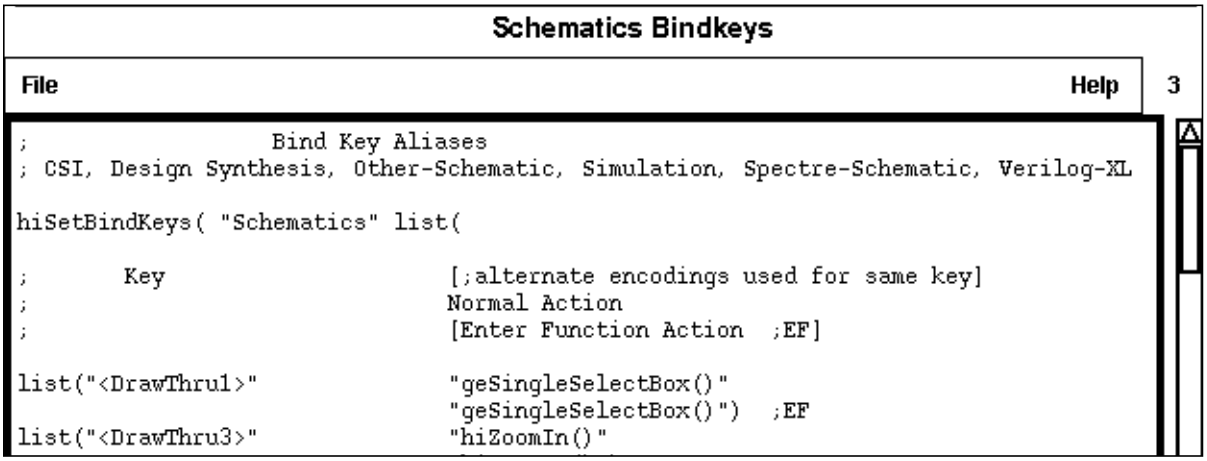
1. In the CIW, choose the *Options – Bindkey* command.

   The Key or Mouse Binding form appears.



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

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



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 Cadence® 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

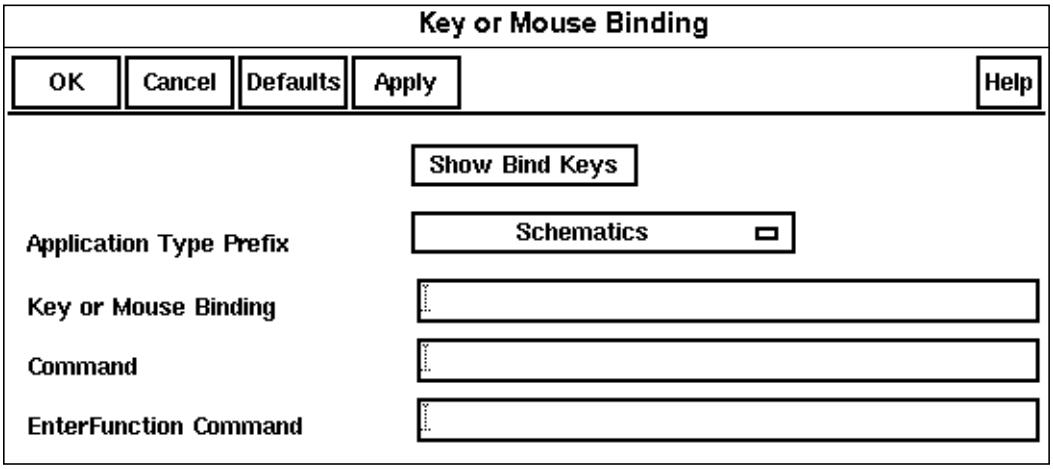
Bindkeys set by the first two methods are valid for the current session only. To set bindkeys you want for every session, you must enter them in your `.cdsinit` file.

Using the Key or Mouse Binding Form

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

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

The Key or Mouse Binding form appears.



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 Cadence® 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 Cadence® 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. This list must contain the name of the simulator.

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 analysis 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.

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.

Output Format (for mixed-signal simulation) lets you specify the waveform display tool that you plan to use for the output. The data is formatted only for the tool you specify so if you want to display with the other tool, you must run the simulation again.

AWD formats the output data in the PSF format (`psfbin` and `psfbinf`) so you can display it with the AWD tool. The default value for the PSF format is `psfbin`. If you want the format to be `psfbinf`, you need to set the `simOutputFormat` variable to `psfbinf`. For details, refer to the section, simOutputFormat.

SimVision Waves formats the output data in the SST format so you can display it with SimVision Waves.

Note: When you choose SimVision Waves as the waveform display tool, you must manually invoke SimVision Waves, and load the analog and digital waveform databases separately.

The waveform databases are created at the following locations:

- Analog:

  projectDirectory/topCellName/simulatorName/topViewName/`psf`

- Digital:

  projectDirectory/topCellName/simulatorName/topViewName/`sst2`

The digital waveform database is created only when you choose a mixed signal simulator such as spectreVerilog or spectreSVerilog.

The simulator that you use must be able to write in the format that you select. For example, the Spectre simulator can write data for SimVision Waves, but the cdsSpice simulator cannot.

Create Checkpoint File(cp):

Y runs spectre with the `+checkpoint` option, which turns on the checkpoint capability.

N runs spectre with the `-checkpoint` option, which turns off the checkpoint capability.

Start from Checkpoint File(rec):

Y runs spectre with the `+recover` option, which restarts the simulation from the checkpoint file, if it exists.

N runs spectre with the `-recover` option, which does not restart the simulation, even if a checkpoint file exists.

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 As is the name you give to the saved state.

Existing States is the listbox where you select the required state.

What to Save controls the type of information saved.

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.

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.

Note: 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.

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.

Variables saves design variables.

Model Setup restores the model setup of the session that was saved.

Model Path restores the model path directories that were current when the session was saved.

Simulator Options saves all simulator option settings. This information is simulator specific and can be retrieved only if the same simulator is being used.

Waveform Setup saves the Waveform window settings.

Stimulus Setup saves the setup of the graphical stimulus form.

Conditions Setup saves the settings for the Circuit Conditions form.

Loading State

Library is the name of the library whose state was saved.

Cell is the name of the cell whose state was saved. This does not need to match the name of the cell you are currently simulating.

Simulator is the name of the simulator active when the state was saved. This does not need to match the current simulator.

State Name is the name specified when the state was saved.

What to Load controls the type of information restored.

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.

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.

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.

Convergence Setup restores the *Node Set*, *Initial Condition*, and *Force Node* settings, as well as DC and transient solutions.

Variables restores design variables.

Model Setup restores the model setup of the session that was saved.

Stimulus Setup restores the setup of the graphical stimulus form.

Model Path restores the model path directories that were current when the session was saved.

Simulator Options restores all Simulator Options form settings. This information is simulator specific.

Waveform Setup restores the Waveform window settings.

Conditions Setup restores the settings for the Circuit Conditions form.

Editing Session Options

Session Base lets you choose the way the Cadence® 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.

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 Cadence® 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.

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. For detailed conversion information, see the *Compatibility Guide*.

Design Variables and Simulation

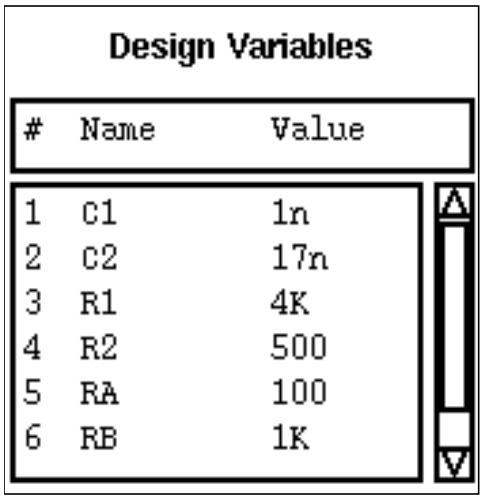
The following section describes how to work with design variables.

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.

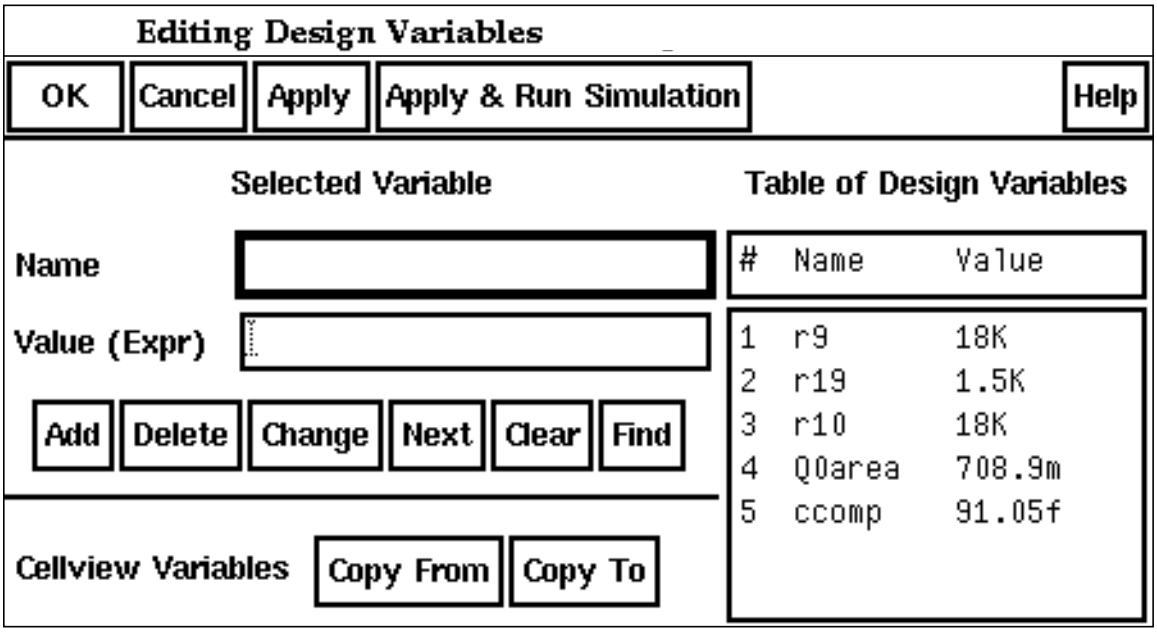


Adding a New Variable

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.



For detailed information about the form, see "Editing Design Variables" on page 109.

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.
5. Click *Add*.

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

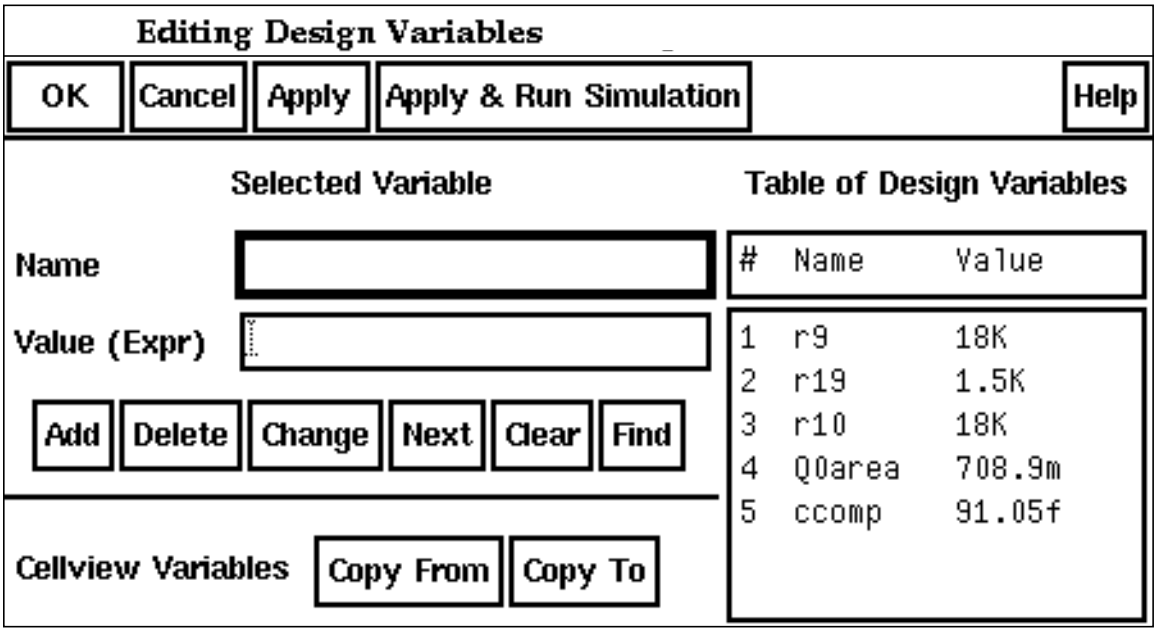
Note: You can also define variables in the definitions file.

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.



For detailed information about the form, see "Editing Design Variables" on page 109.

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.

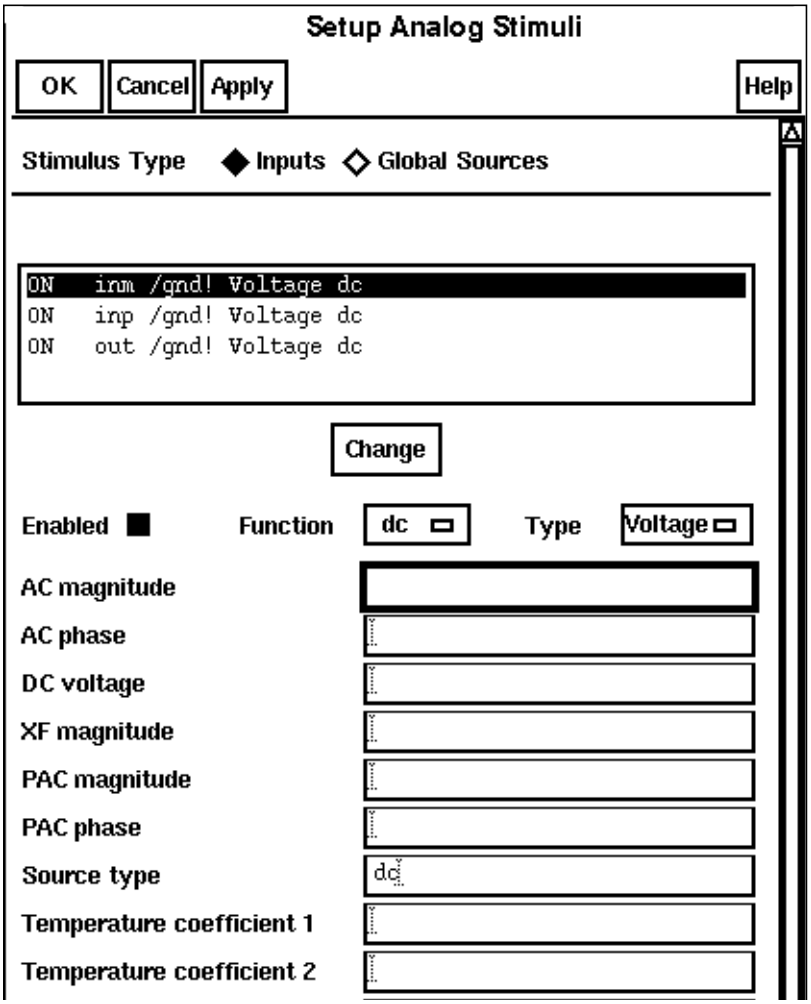
Name	*refLibs*
Type	*string*
Value	list of one or more libraries in search sequence

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.



For detailed information about the form, see "Setup Analog Stimuli Form" on page 107.

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*.

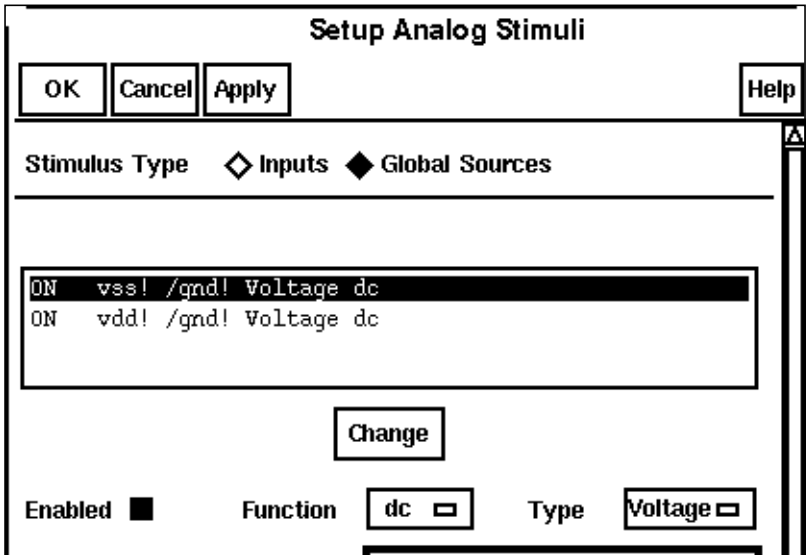
The list box displays the signal and the proper stimulus syntax for your simulator.

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*.



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 Cadence® 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`
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

Field	Value
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:

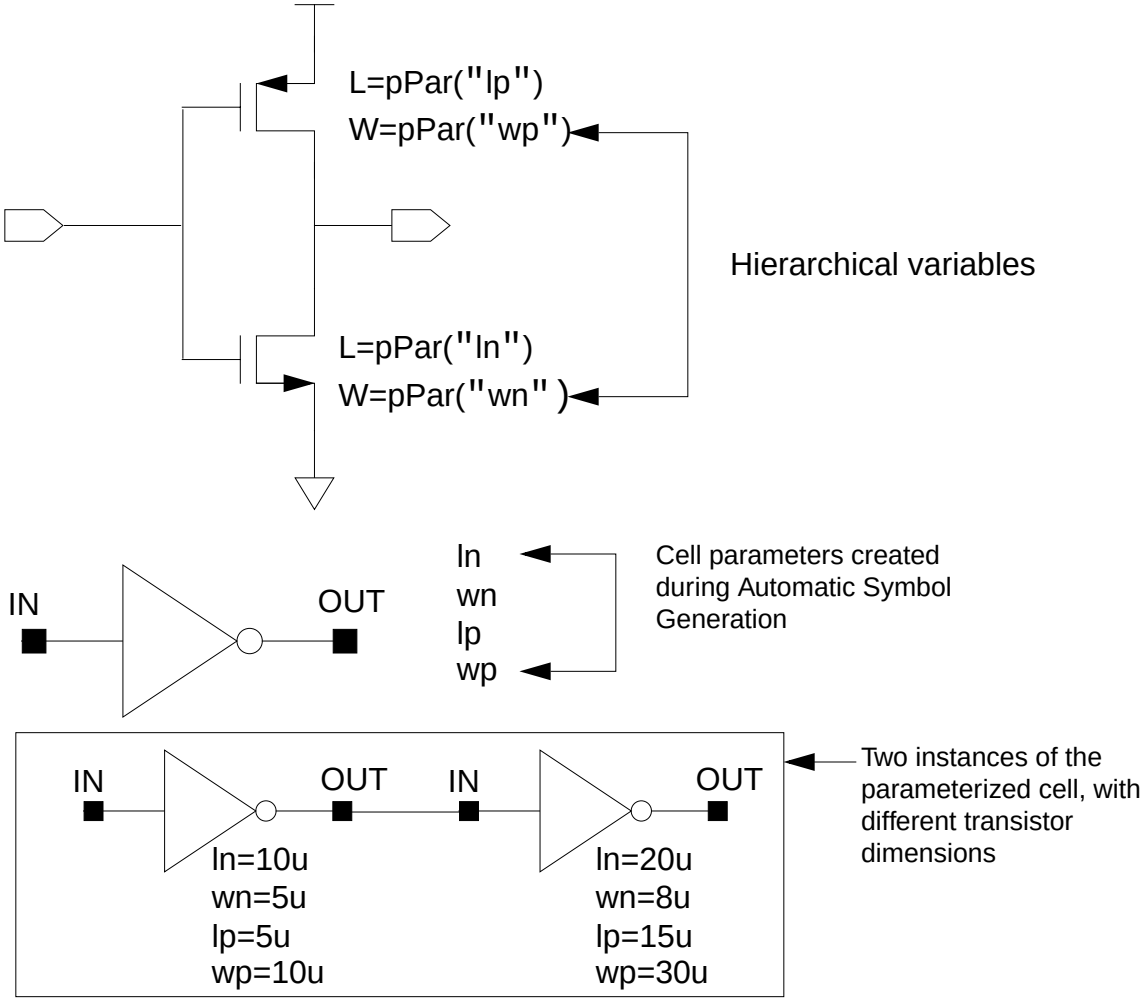
```
pPar("vss")
```

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 which 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.

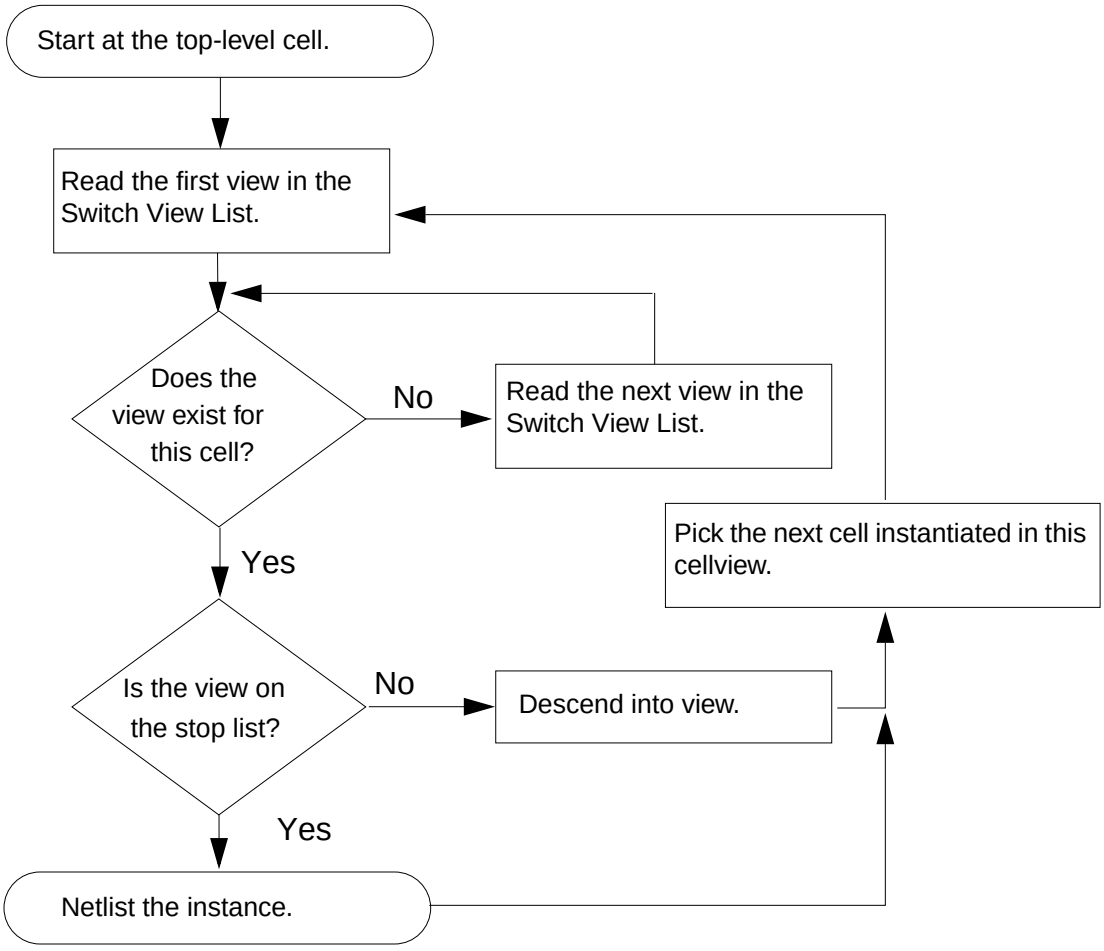
For analog simulation, you specify the global or default view list in the *Switch View List* field of the Environment Options form, when the cellview selected is not a configuration. If an instance does not have any of the views listed in the view list, the netlister reports an error.

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 the *Cadence Parasitic User Guide* and the *Cadence® Mixed-Signal Simulation Interface Option User Guide* for details.

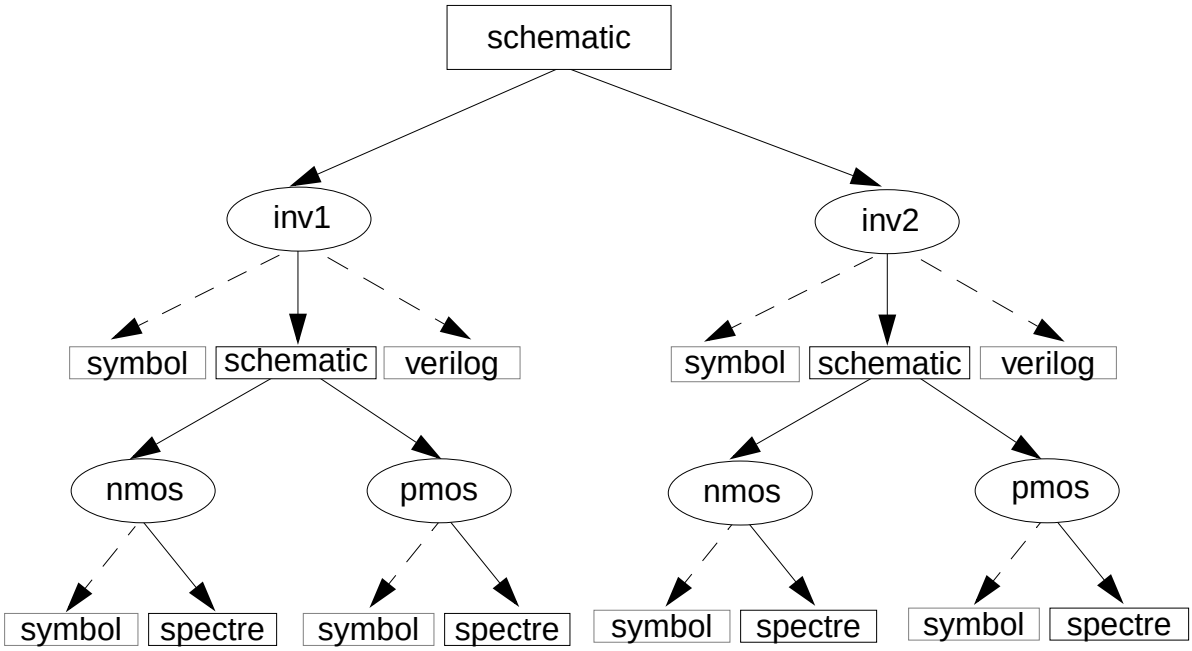
The flowchart shows how a netlister like OSS expands a design.



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

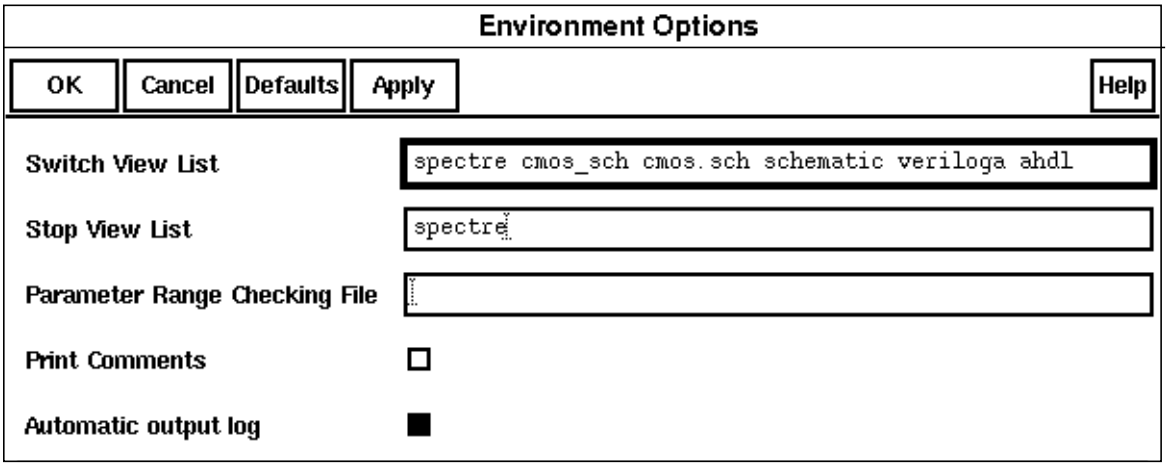


Modifying View Lists and Stop Lists

To modify a switch view list or stop view list,

1. Choose *Setup – Environment* from the Simulation window.

The Environment Options form appears.



For detailed information about the form, see "Environment Options" on page 105.

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.

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*.

Form Field Descriptions

Environment Options

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. This list must contain the name of the simulator.

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

Parameter Range Checking File (Spectre option only) lets you type the path to a file containing the correct ranges for component parameters. If this path is present, the Spectre circuit 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.

Print Comments

Y places extra comments, regarding component location and name, in the netlist.

N limits the comments placed in the netlist.

Note: Do *Netlist -> Re-create* for the comments to appear in the netlist.

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.

Create Checkpoint File(cp):

Y runs spectre with the `+checkpoint` option, which turns on the checkpoint capability.

N runs spectre with the `-checkpoint` option, which turns off the checkpoint capability.

Start from Checkpoint File(rec):

Y runs spectre with the `+recover` option, which restarts the simulation from the checkpoint file, if it exists.

N runs spectre with the `-recover` option, which does not restart the simulation, even if a checkpoint file exists.

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

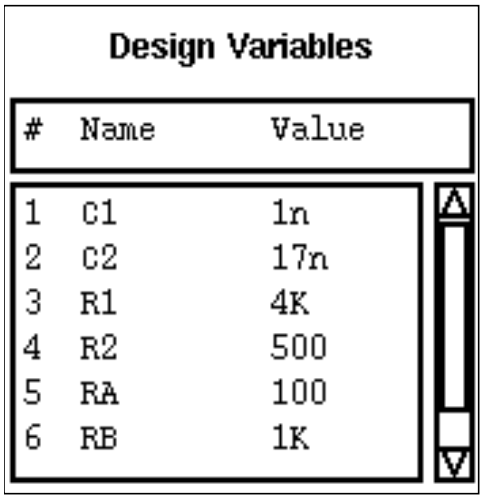
The following section describes how to work with design variables.

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.

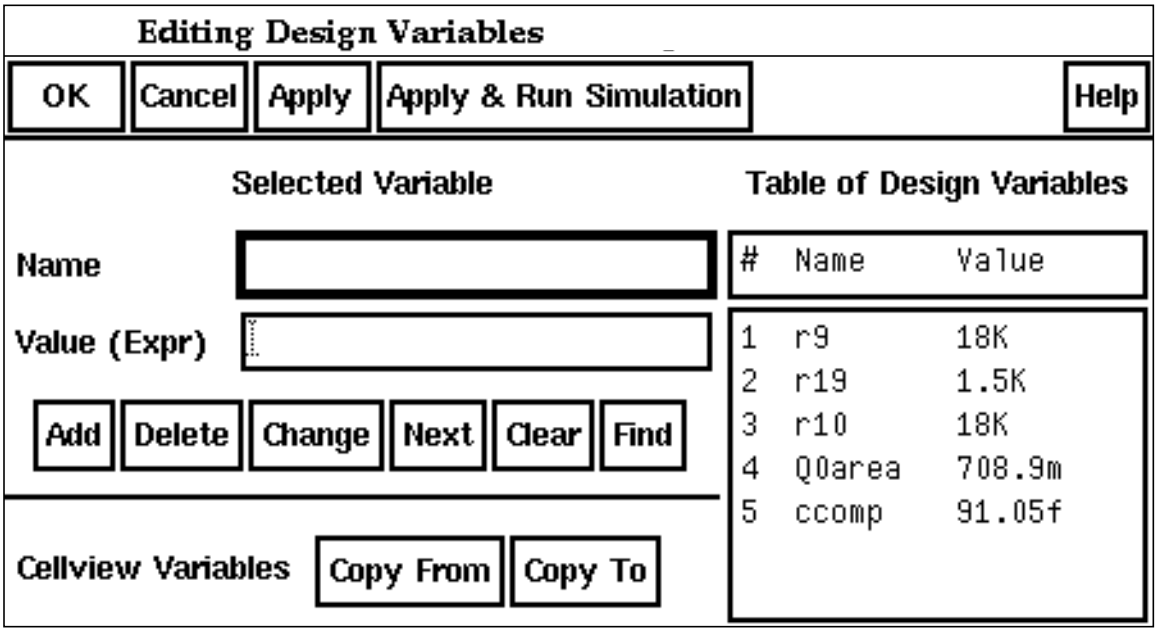


Adding a New Variable

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.



For detailed information about the form, see "Editing Design Variables" on page 160.

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.

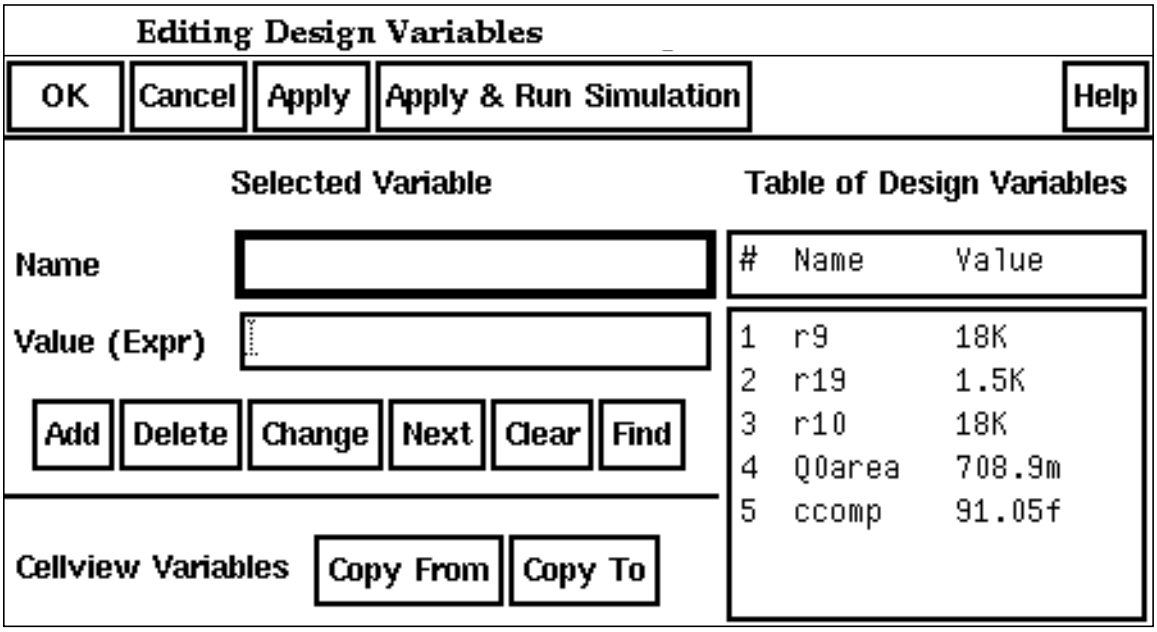
Note: You can also define variables in the Update or Init file.

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.



   For detailed information about the form, see "Editing Design Variables" on page 160.

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

```
psprop keyword option_name "value"
```

The keywords are

Keyword	Type of Spectre Option
`Spectre_Opt`	General options
`Spectre_Tran`	Transient options
`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 Cadence® 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.

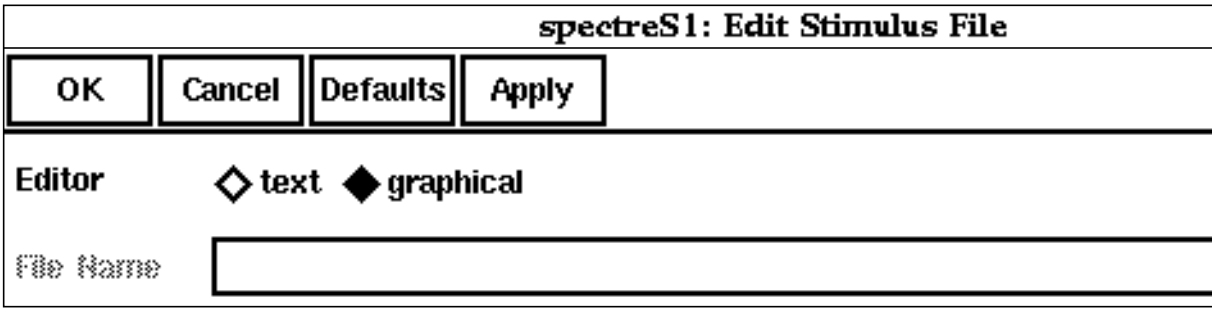
3. Create a raw or final netlist.

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.



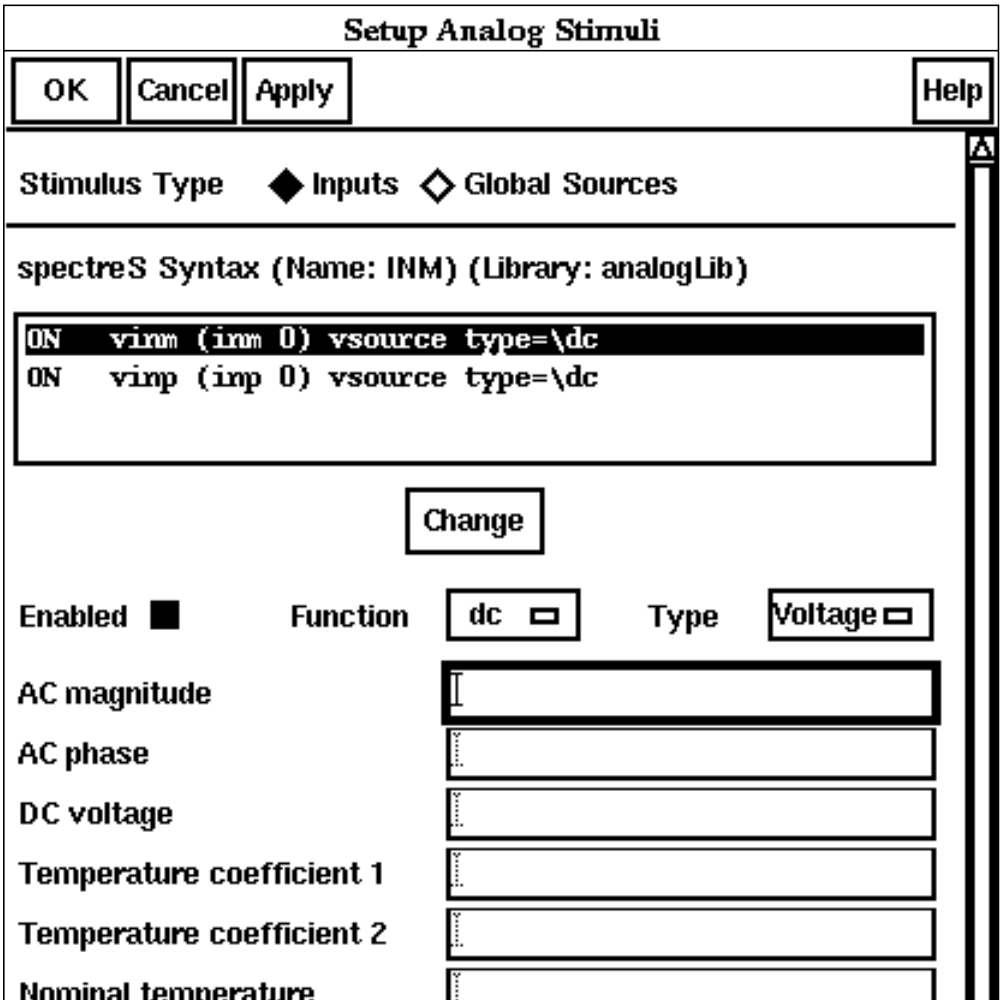
   Text mode opens a text editor window where you can type in input commands in the appropriate syntax for your simulator.

   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.



For detailed information about the form, see "Setup Analog Stimuli" on page 158.

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*.

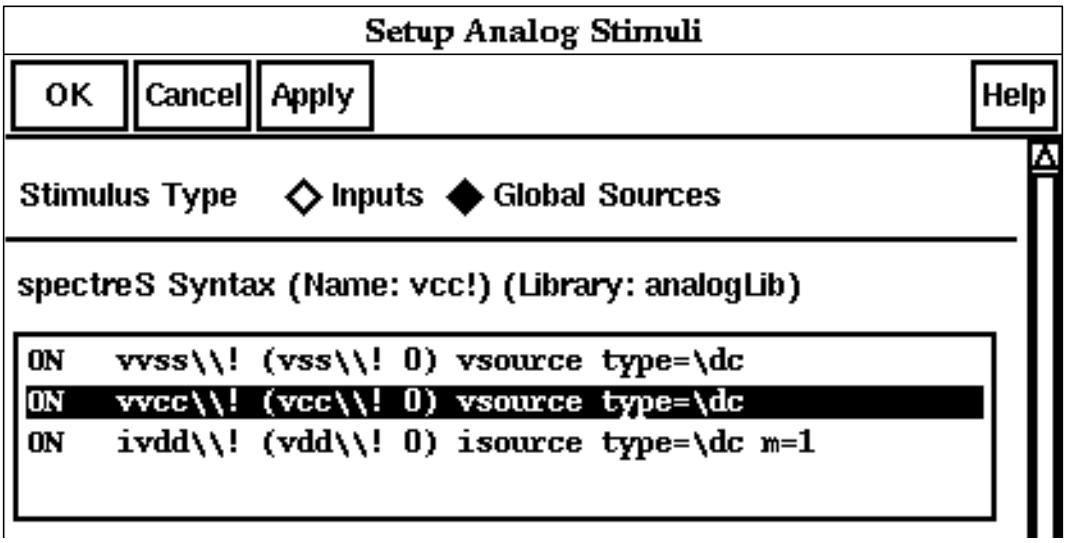
The list box displays the signal and the proper stimulus syntax for your simulator.

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*.



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*.

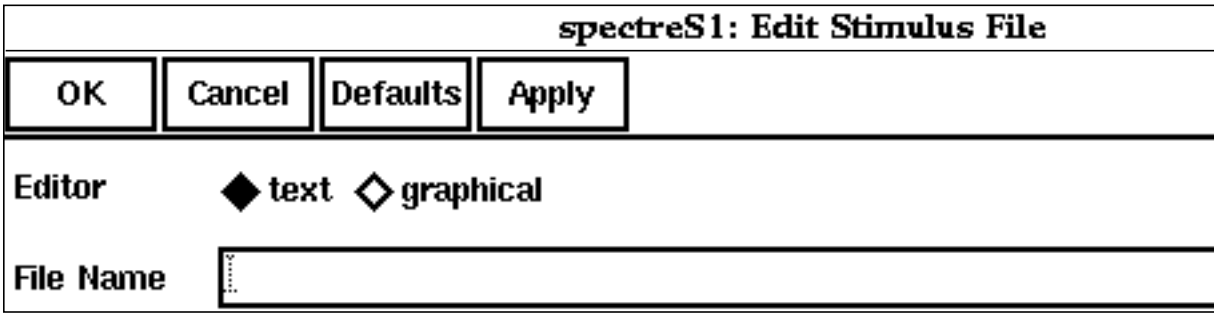
The Environment Options form appears.

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.



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 Cadence® 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.

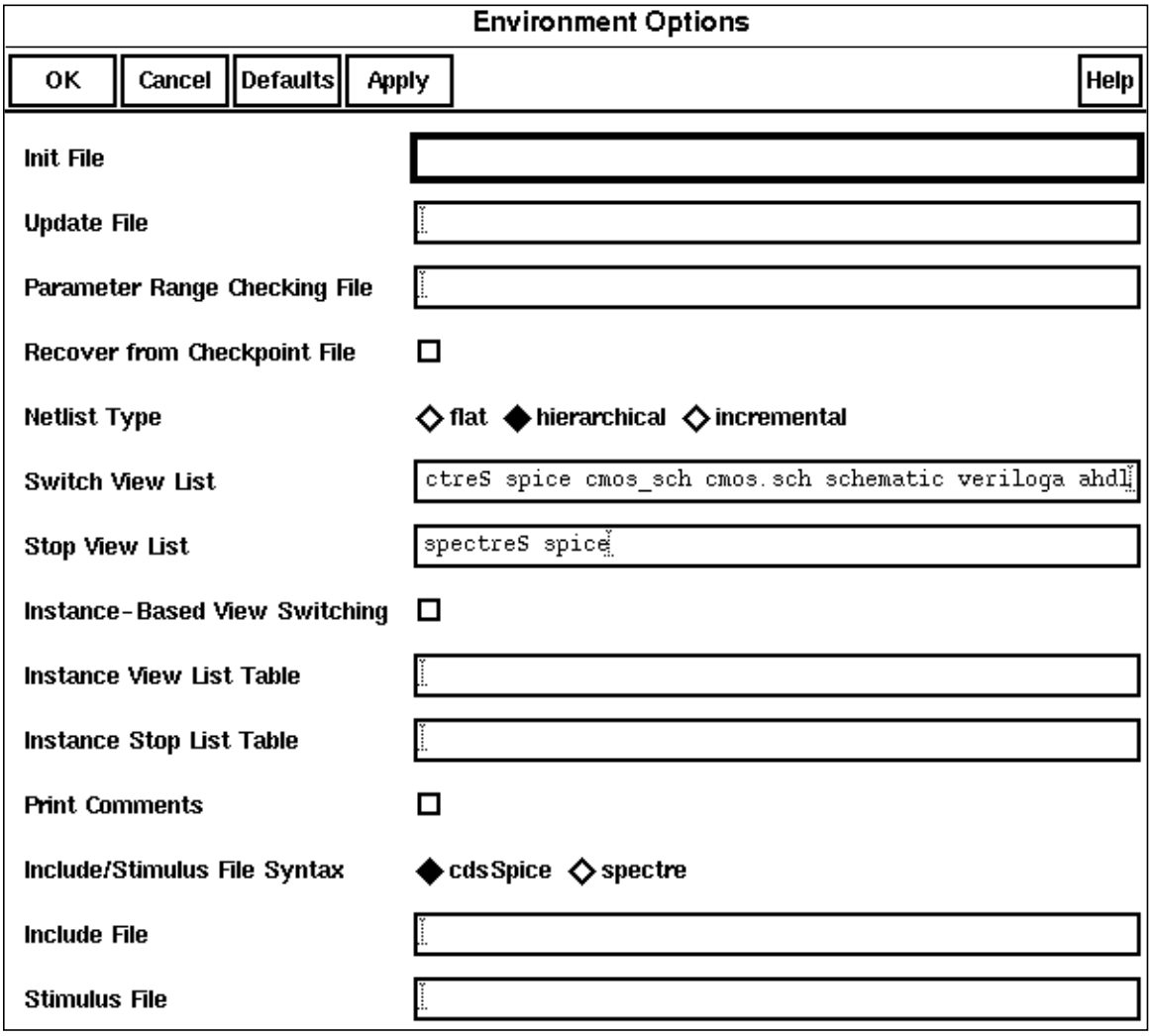
1. Create one or two files in the directory

   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.



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

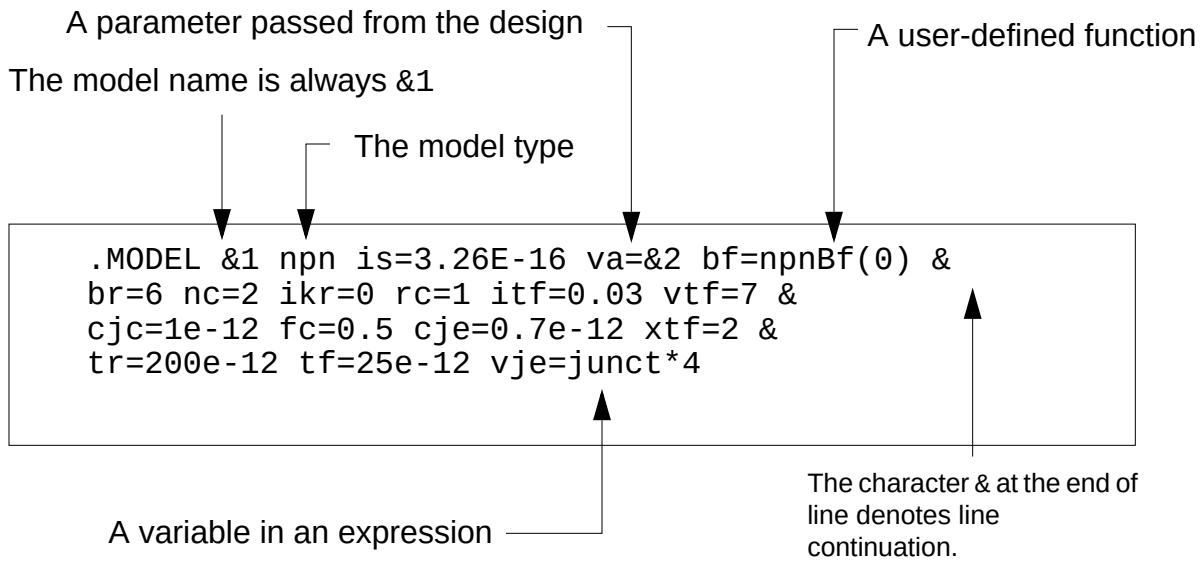
The analog circuit design environment simulator expects to find model statements that follow Cadence SPICE syntax in files named `device.m` in one of the directories in the model path. You specify the model directory search path in the Environment Options form.

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.

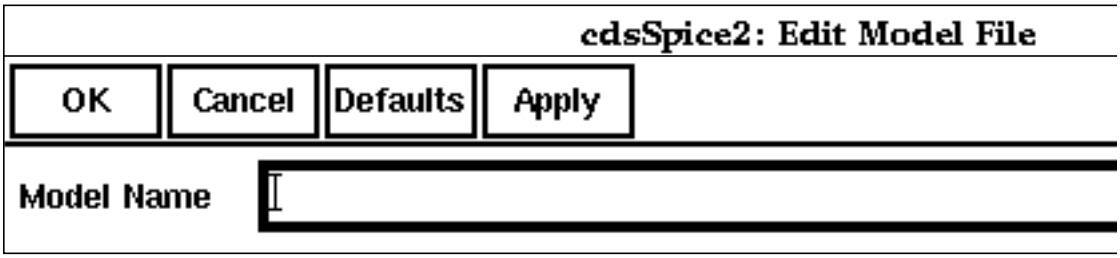


Editing Model Files in the Models Directory

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

Field	Value
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

CDF Parameter	Macro	macroArgumentStyle	Parameter m
StoreDefault	no	no	no
name	macro	macroArgumentStyle	m
Prompt	subcircuit name	Macro Arg Style	multiplier
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:

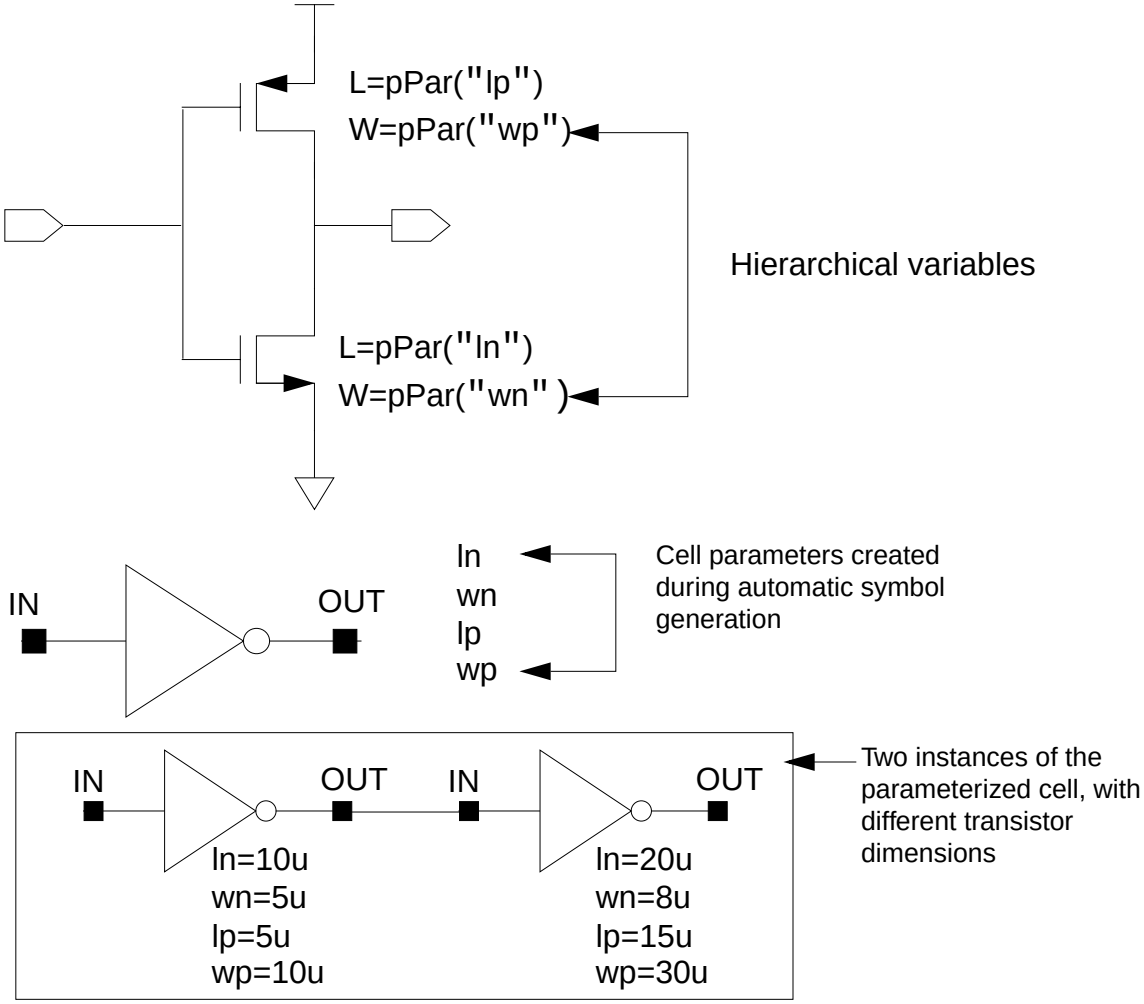
```
pPar("vss")
```

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.)

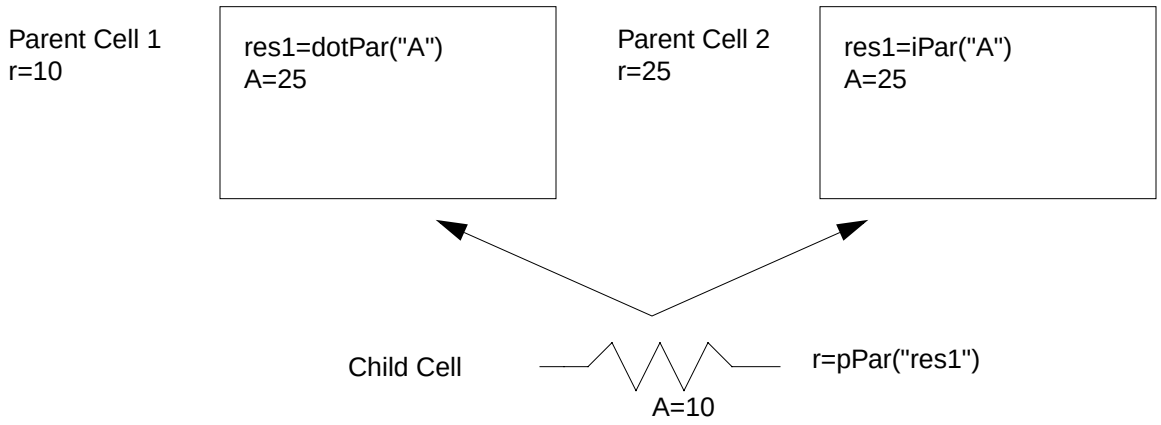
Functions		Meaning	Scope rules
AEL	**NLP (OSS)**		
`iPar`	[~	Instance parameter	Search the instance carrying `iPar`, then the effective cell CDF
`pPar`	[+	Parent parameter	Search the parent instance, then the effective cell CDF of the parent instance
`atPar`	[@	At or above the instance	Search the instance 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.

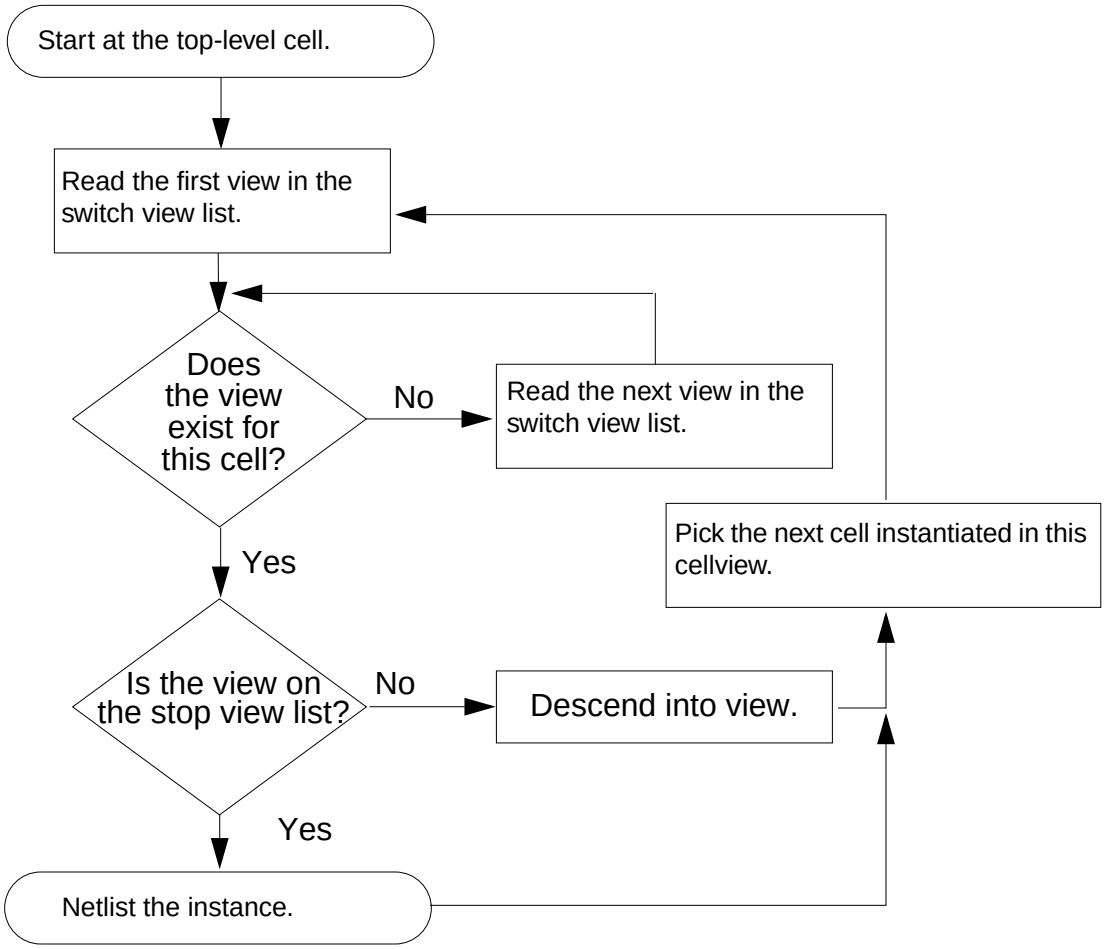
At each level in the hierarchy, there can be several views of each cell. You use a view list to specify which view the analog circuit design environment selects and descends into. View lists can be global to the entire design or specific to an instance as specified by its property values.

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.

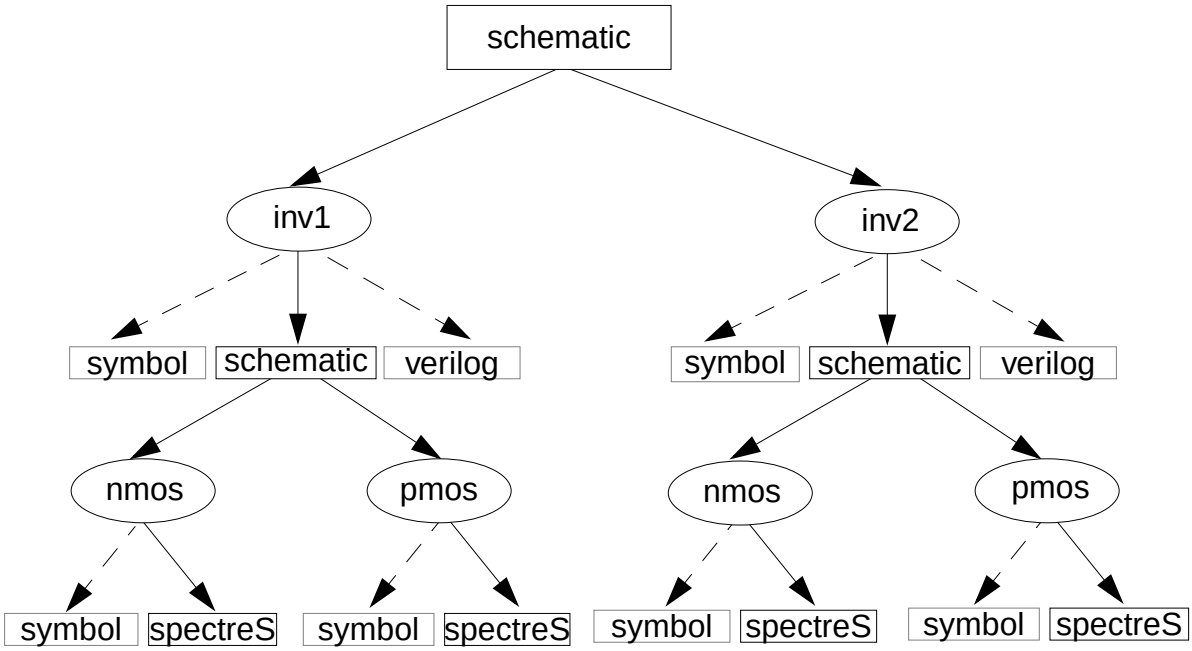
The flowchart shows how a netlister like OSS expands a design.



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

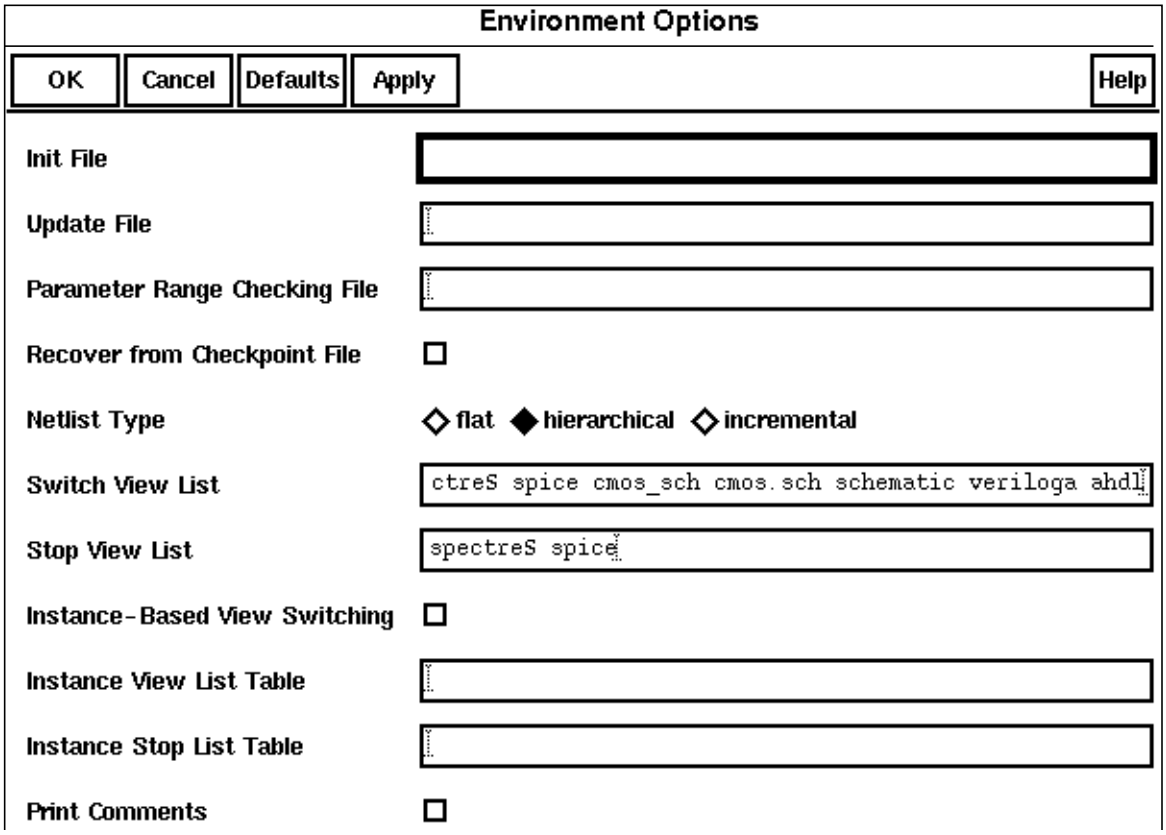


Modifying View Lists and Stop View Lists

To modify a *Switch View List* or stop view list,

1. Choose *Setup – Environment* from the Simulation window.

The Environment Options form appears.



For detailed information about the form, see "Environment Options" on page 153.

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.

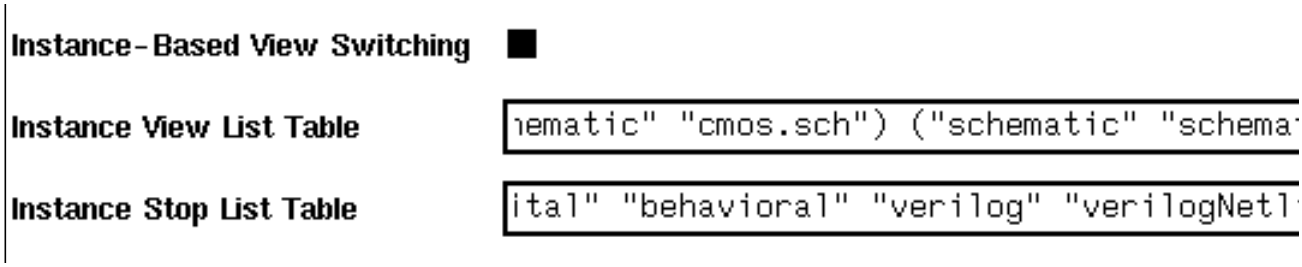
   The format for the table is

   ```
   (("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.



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

Environment Options

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

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

Parameter Range Checking File lets you type 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 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 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. This list must contain the name of the simulator.

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

This option 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* fields.

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* fields to override, on a per-instance basis, the default simulation view selection algorithm imposed by *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.

Print Comments

When off, comments are not printed.

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

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.

Output Format (for mixed-signal simulation) lets you specify the waveform display tool that you plan to use for the output. The data is formatted only for the tool you specify so if you want to display with the other tool, you must run the simulation again.

AWD formats the output data in the PSF format (`psfbin` and `psfbinf`) so you can display it with the AWD tool. The default value for the PSF format is `psfbin`. If you want the format to be `psfbinf`, you need to set the `simOutputFormat` variable to `psfbinf`. For details, refer to the section, simOutputFormat.

SimVision Waves formats the output data in the SST format so you can display it with SimVision Waves.

Note: When you choose SimVision Waves as the waveform display tool, you must manually invoke SimVision Waves, and load the analog and digital waveform databases separately.

The waveform databases are created at the following locations:

- Analog:

  projectDirectory`/`*topCellName*`/`*simulatorName*`/`*topViewName*`/psf`

- Digital:

  projectDirectory`/`*topCellName*`/`*simulatorName*`/`*topViewName*`/sst2`

The digital waveform database is created only when you choose a mixed signal simulator such as spectreVerilog or spectreSVerilog.

The simulator that you use must be able to write in the format that you select. For example, the Spectre simulator can write data for SimVision Waves, but the cdsSpice simulator cannot.

Create Checkpoint File(cp):

Y runs spectre with the `+checkpoint` option, which turns on the checkpoint capability.

N runs spectre with the `-checkpoint` option, which turns off the checkpoint capability.

Start from Checkpoint File(rec):

Y runs spectre with the `+recover` option, which restarts the simulation from the checkpoint file, if it exists.

N runs spectre with the `-recover` option, which does not restart the simulation, even if a checkpoint file exists.

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.

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.

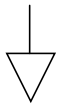
5

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.


gnd

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 Cadence SPICE Analysis" on page 164, 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.

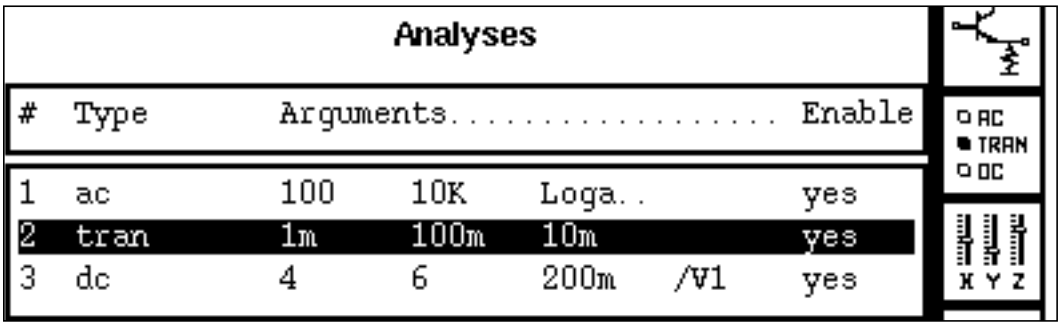
Tip

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 *Cadence Analog Design Environment* window.

Deleting an Analysis

To delete an analysis,

1. In the Simulation window, click the analysis to highlight it.

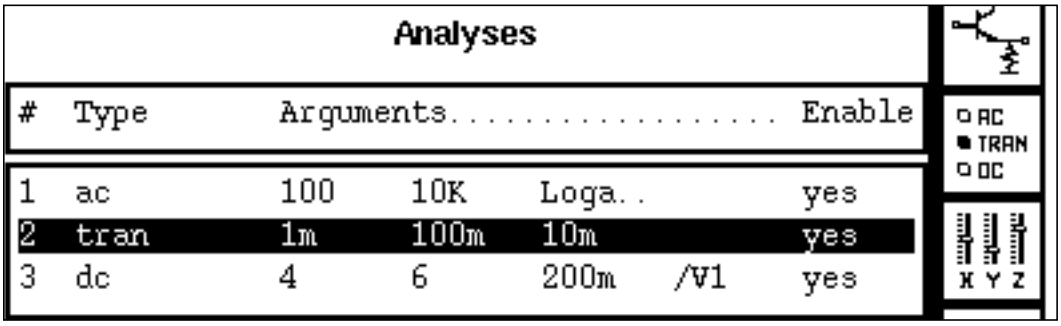


2. Choose *Analyses – Delete* or click the delete icon.

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.



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 Enviornment – 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. Choose a restore mode.

   - Choose *append* if you want to merge the current and saved analyses, overwriting any that are duplicated in both sets.
   - Choose *overwrite* if you want to delete all the current analyses that are not in the saved set.

5. 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.

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.

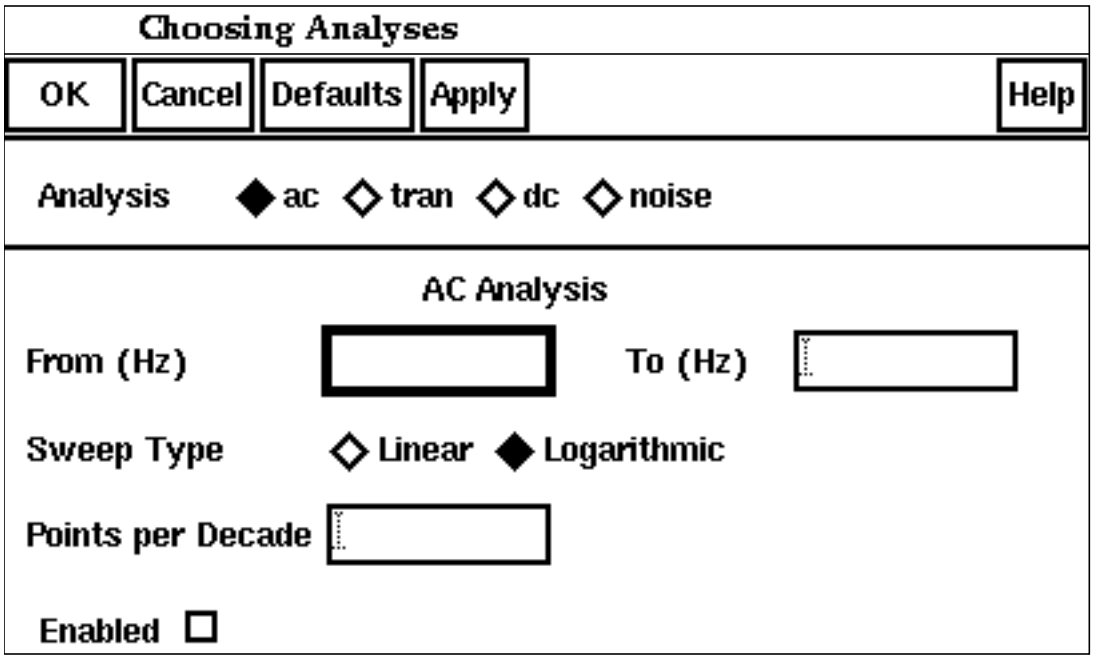
The next step is usually selecting the outputs you want to save.

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*.

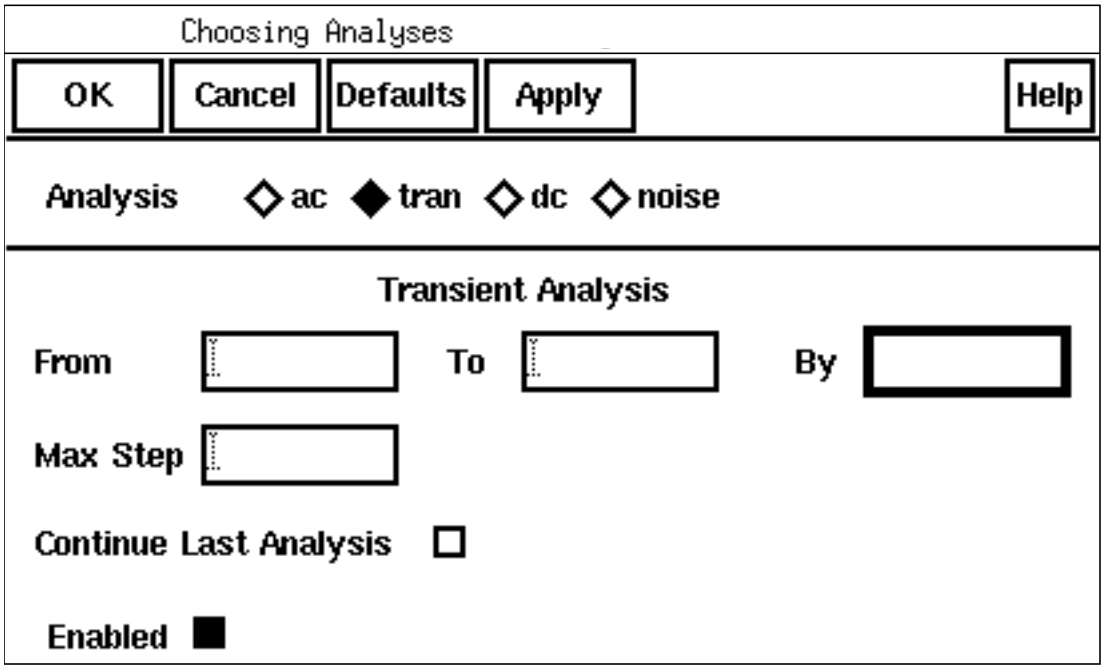


2. Type a starting and stopping frequency in *From* and *To*.
3. Choose a sweep type option and enter the frequency increment in hertz or points per decade.
4. Click *Apply*.

Transient Analysis

To set up a transient analysis,

1. In the Choosing Analyses form, choose *tran* for *Analysis*.



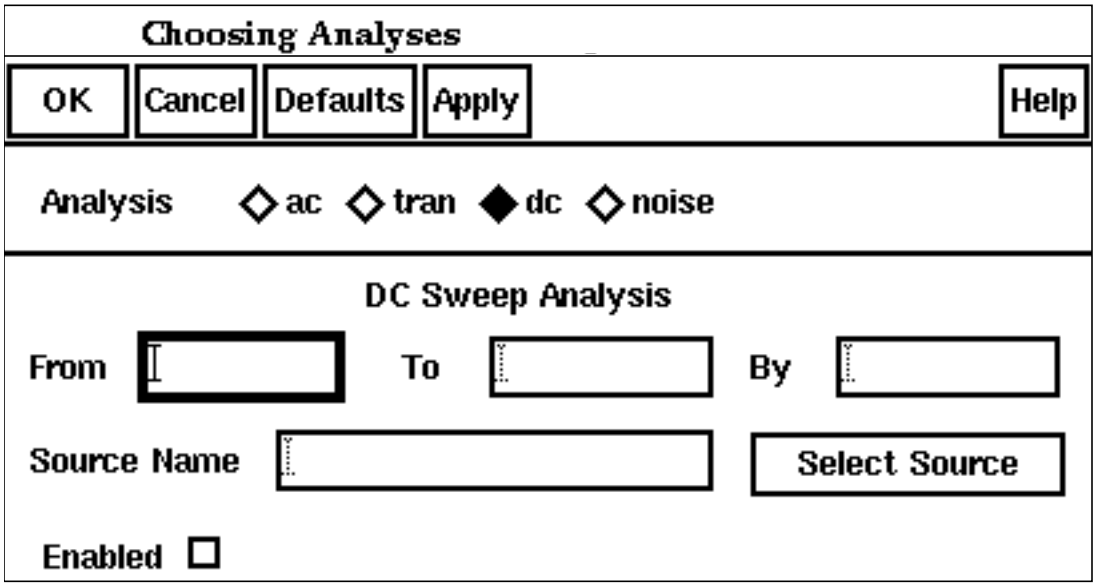
2. Type the starting and stopping times in *From* and *To*.
3. Type the time increment in *By*.
4. Set the maximum timestep (DELMAX) in *Max Step*, if desired.
5. Click *Apply*.

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*.

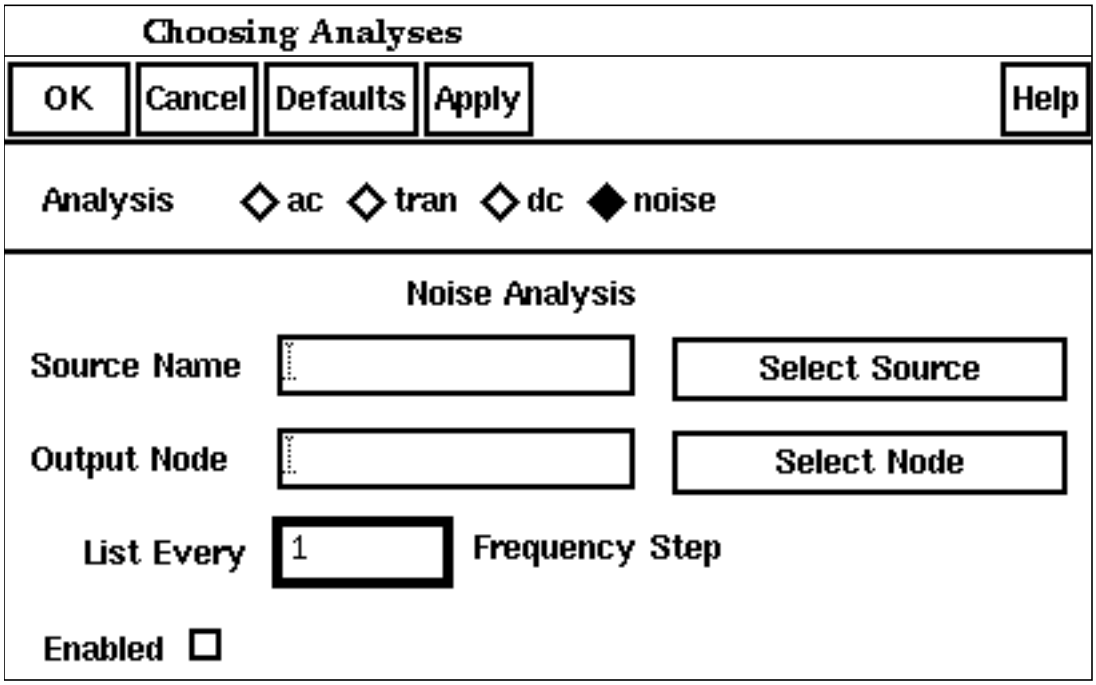


2. Type the starting and stopping times in *From* and *To*.
3. Type the time increment in *By*.
4. Type the name of the current or voltage source in *Source Name*, or click *Select Source* and click the source in the Schematic window.
5. Click *Apply*.

Noise Analysis

To set up a noise analysis,

1. In the Choosing Analyses form, choose *noise* for *Analysis*.



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.

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.

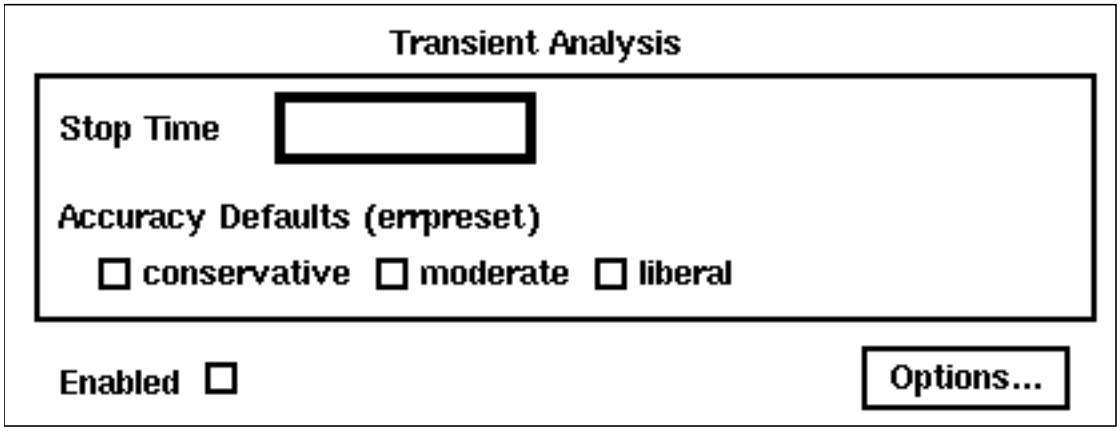
The next step is usually selecting the outputs you want to save.

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,

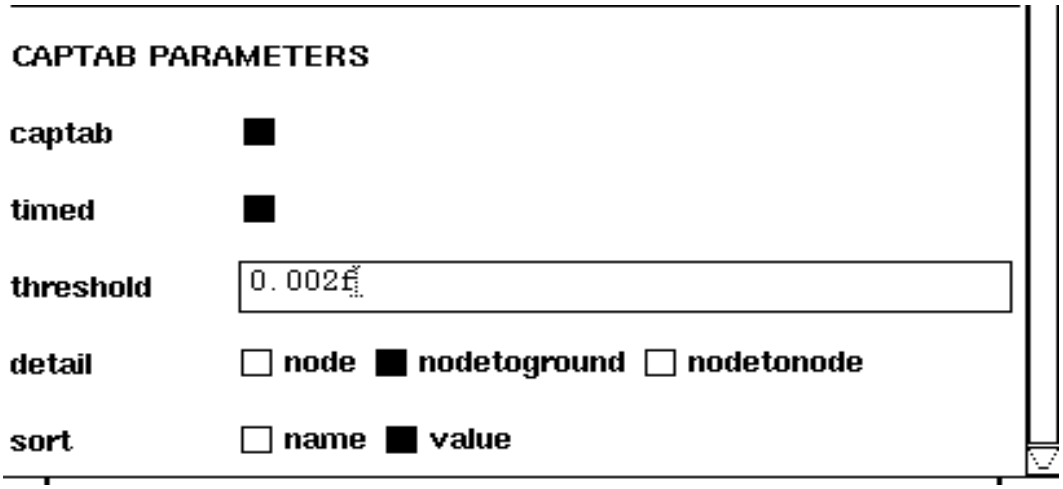


1. In the Choosing Analyses form, enter the *Stop Time*.
2. Click *Options* to
   - Set Spectre options controlling the time step
   - Set any other options related to transient simulation
3. Click *Apply*.

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 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

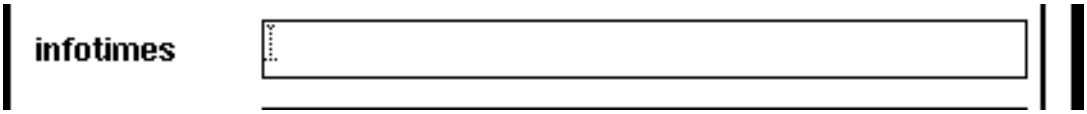
detail can be set to *node*, *nodetoground*, and *nodetonode*. The default option is *node*

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 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 schemetic, 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.

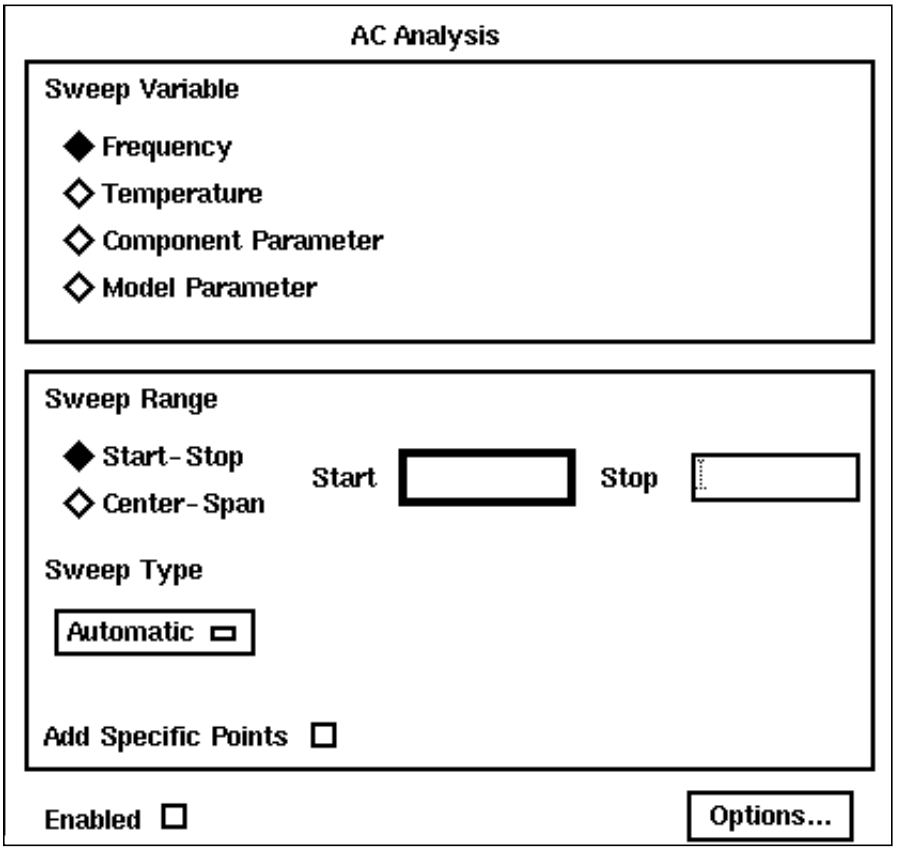
AC Small-Signal Analysis

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.



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 + Total Points = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Total Points = log
- Add Specific Points = values=[…]

4. Click *Options* to select the Spectre options controlling the simulation.
5. Click *Enabled* and *Apply*.

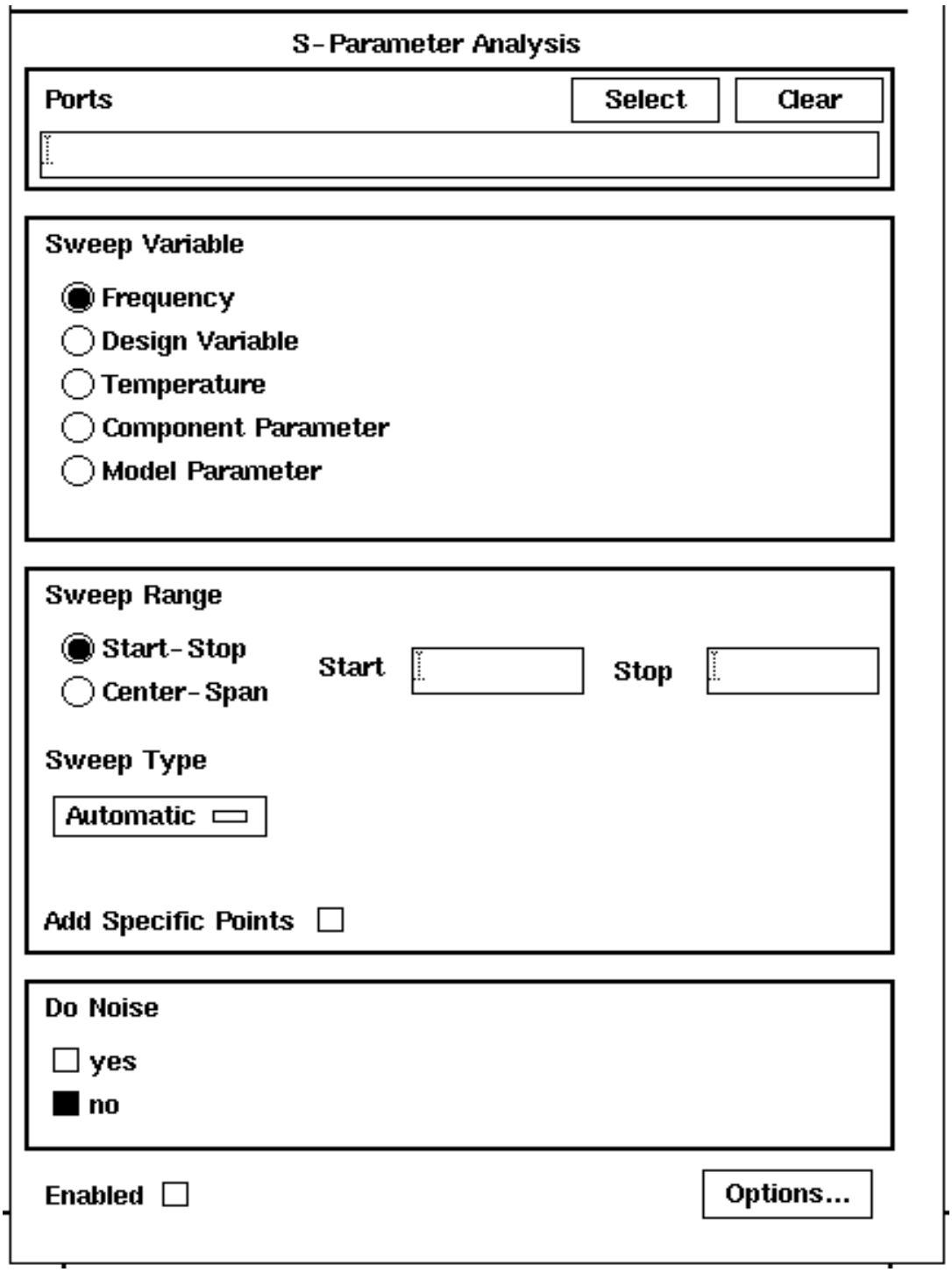
S-Parameter Analysis

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.



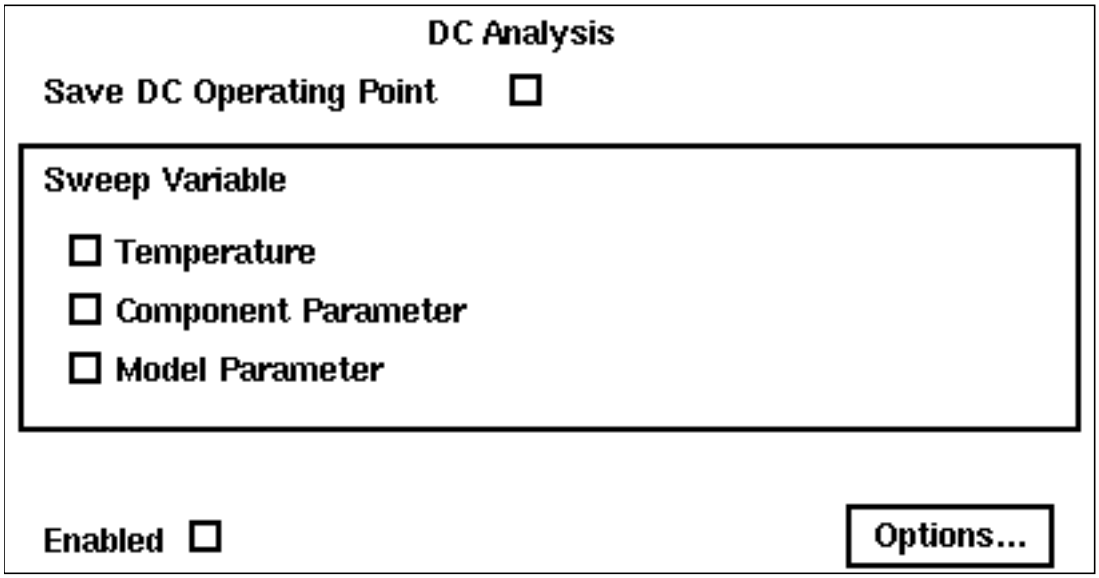
2. Specify the list of active *Ports*. In this field, the ports are numbered sequentially beginning with one, in the order given. Otherwise, all ports present in the circuit are active and the port numbers used are those that were assigned on the port statements.
3. Choose a sweep variable option and 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 + Total Points = lin
   - Logarithmic + Points Per Decade = dec
   - Logarithmic + Total Points = log
   - Add Specific Points = values=[…]
5. Click *Options* to select the Spectre options controlling the simulation.
6. Click the *Noise* radio button to perform noise analysis.
7. 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.

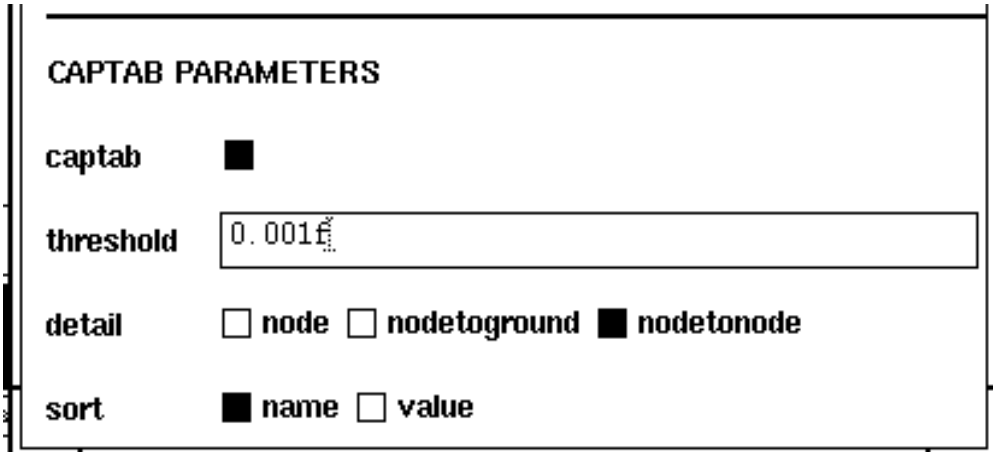


To save the DC operating point,

➤ 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 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*.

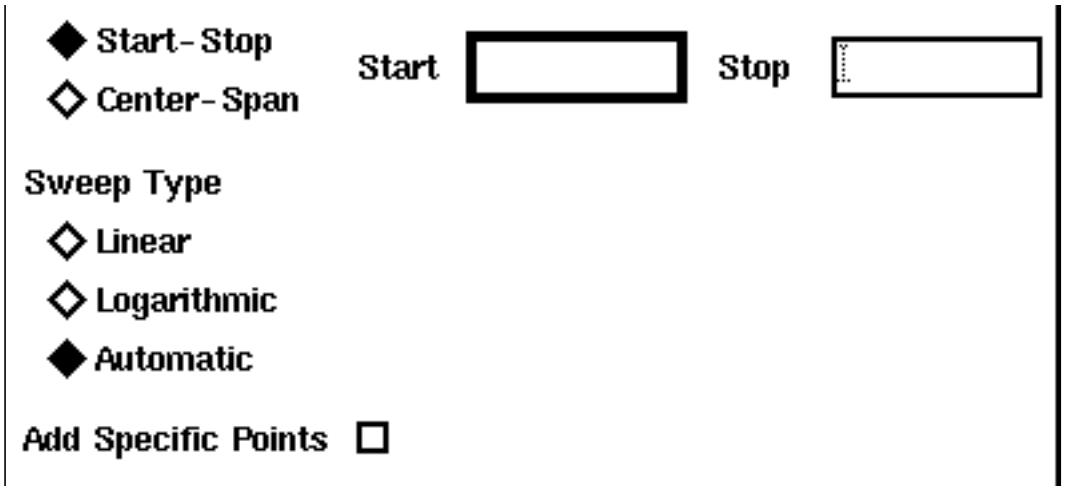
For details on DC Analysis refer to the *Analysis Statements* chapter of the *Spectre Circuit Simulator Reference*.

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.



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 + Total Points = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Total Points = log
- Add Specific Points = values=[…]

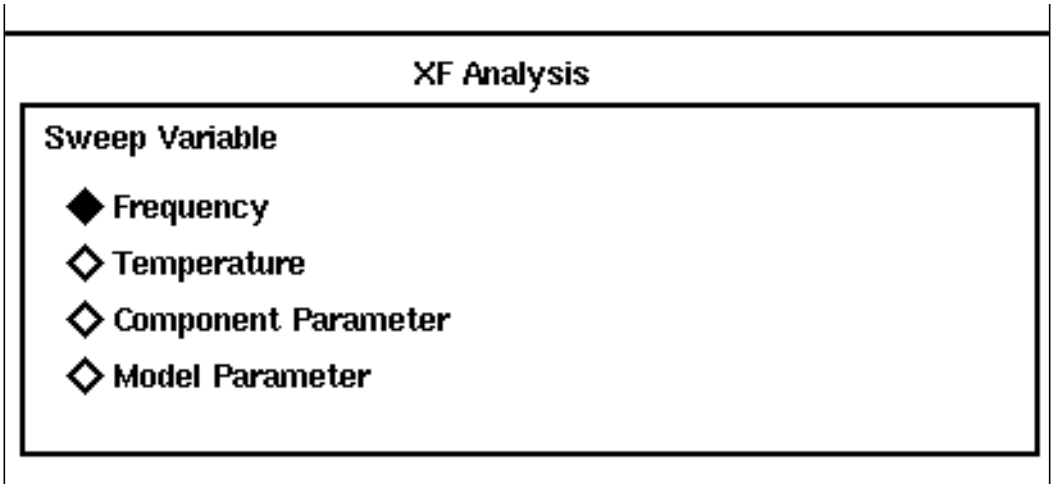
4. Click *Options* to set the spectreS options controlling DC simulation.

5. Click *Apply*.

Transfer Function Analysis

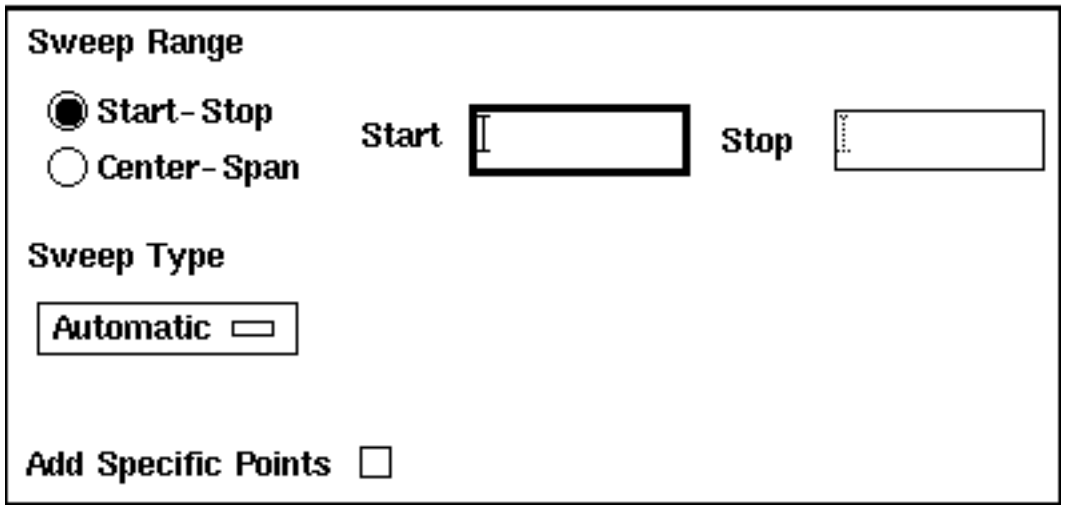
The transfer function, or xf, analysis linearizes the circuit about the DC operating point and performs a small-signal analysis that calculates the transfer function from every independent source or instance terminal in the circuit to a designated output. The variable of interest at the output can be 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 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.

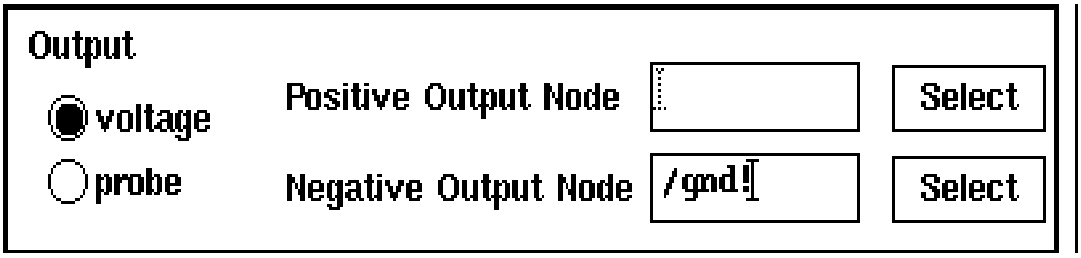
2. Specify the sweep range and type.



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 *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 *Negative Output Node,* and click an instance in the schematic.

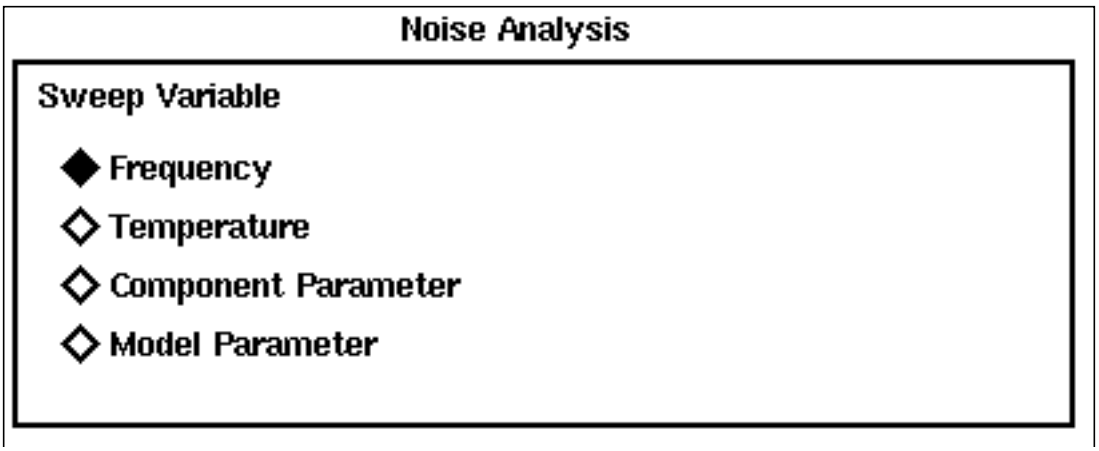
Note: While selecting nodes, select the nodes/nets around the desired instance.

4. Click *Options* to set the spectreS options controlling transfer function simulation.
5. Click *Apply*.

Noise Analysis

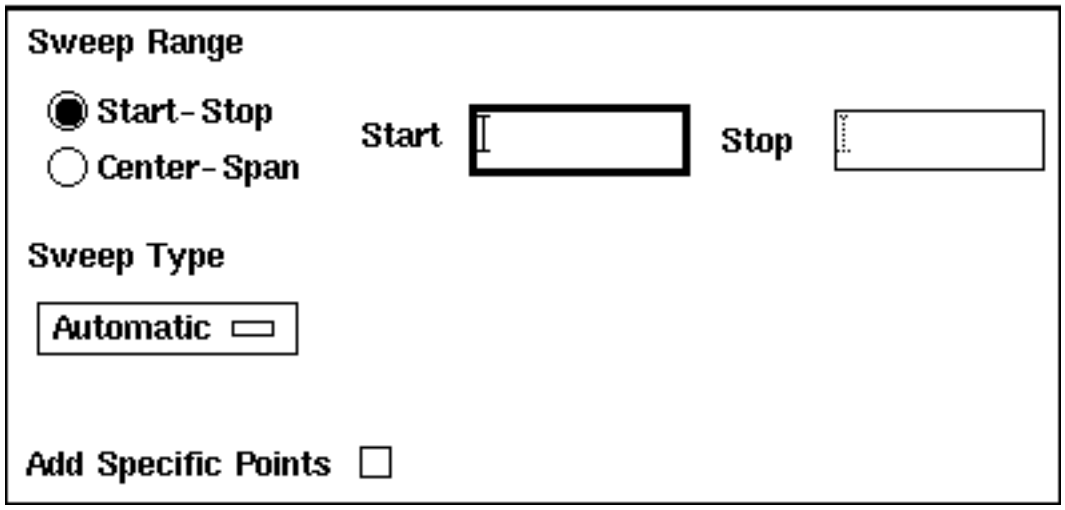
The noise analysis linearizes the circuit about the DC operating point and computes the total-noise spectral density at the output. If you specify an input probe, the transfer function and the input-referred noise for an equivalent noise-free network is computed. To set up a noise analysis,

1. Choose a sweep variable option and specify any necessary parameters.



- 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.

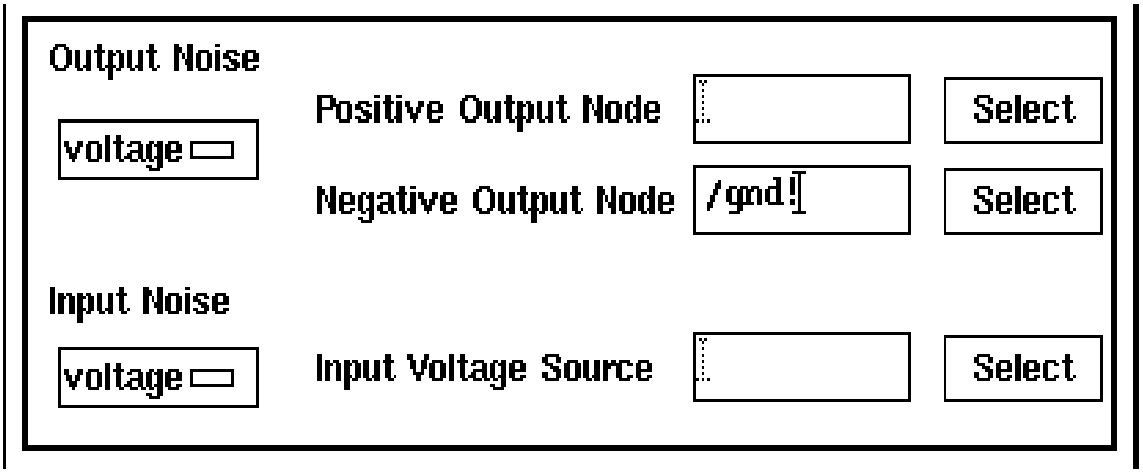
2. Specify the sweep range and type.



The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Total Points = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Total Points = log
- Add Specific Points = values=[…]

3. Choose an *Output Noise* option.



- 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.

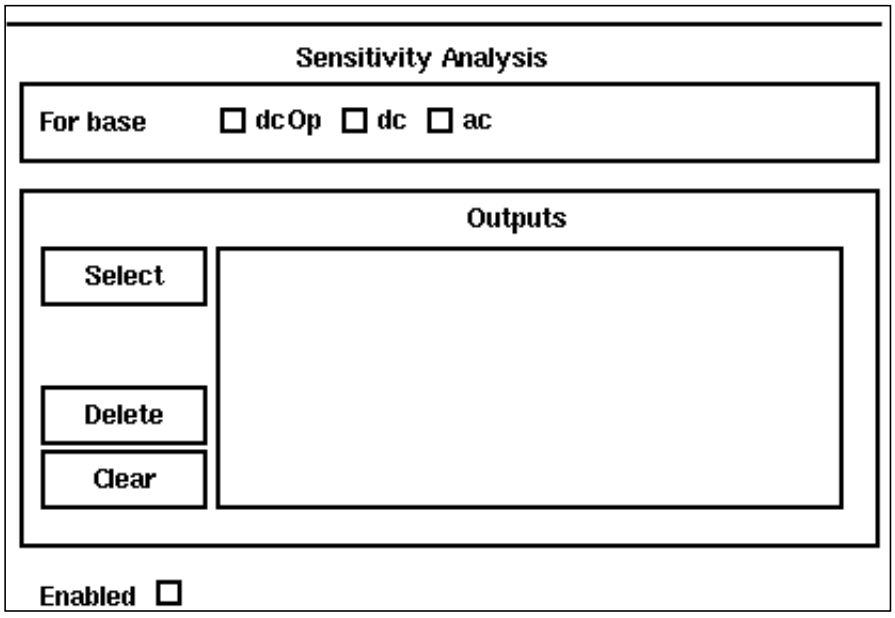
Note: While selecting nodes, select the nodes/nets around the desired instance.

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*.

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* 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	DesignParameter
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.

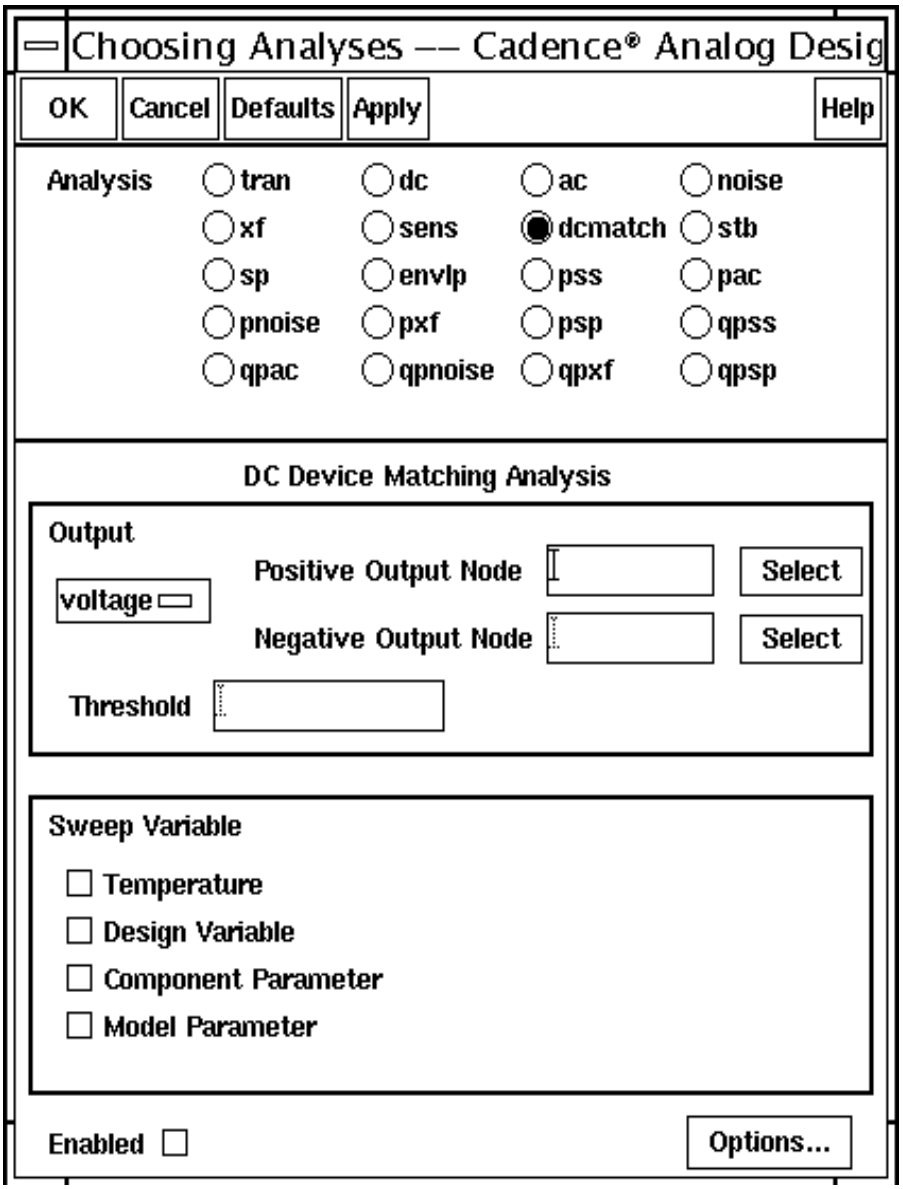
To set up a DC Mismatch analysis for the Spectre simulator,

1. Select *Analyses – Choose* from the *Cadence® Analog Design Environment* window.

   The *Choosing Analyses* form appears.

2. Choose *dcmatch* .

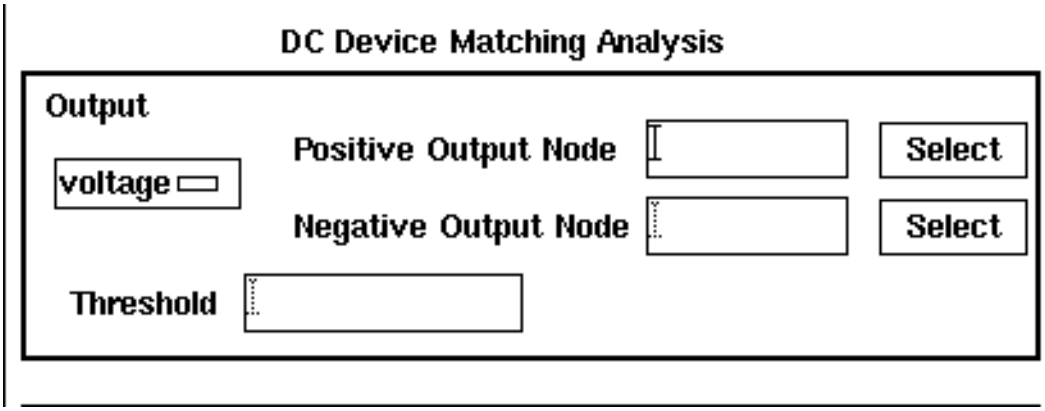
The *Choosing Analyses* form re-displays to show the fields that are required for DC mismatch analysis.



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

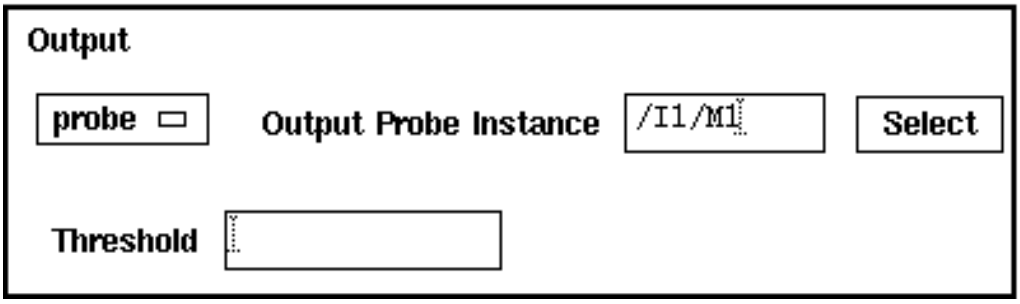


b. Click *Select* opposite *Positive Output Node* and click a net in the schematic for the positive output node. Optionally, click *Select* opposite *Negative Output Node* and click a net in the schematic.

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.



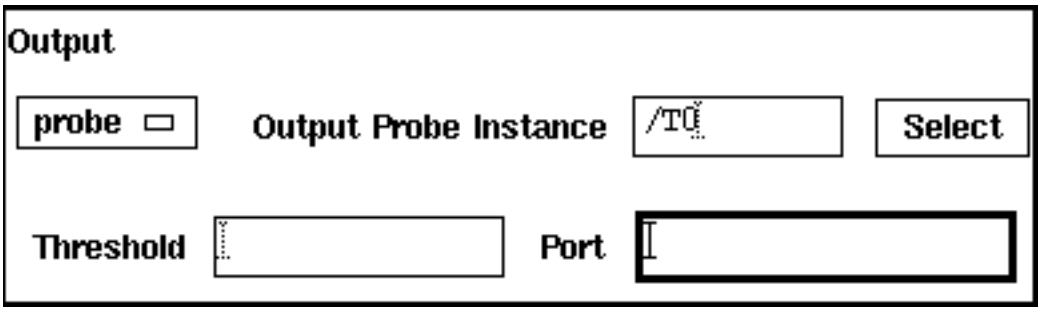
Note: This probe device selected needs to have its terminal currents as network variables. For any other device selection, a warning will be displayed in the CIW stating that the selected object is not a valid type.

Valid Spectre Devices and corresponding analogLib Cells.

Device	Corresponding anlogLib Cells
inductor	ind, pinductor

Device	Corresponding anlogLib Cells
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

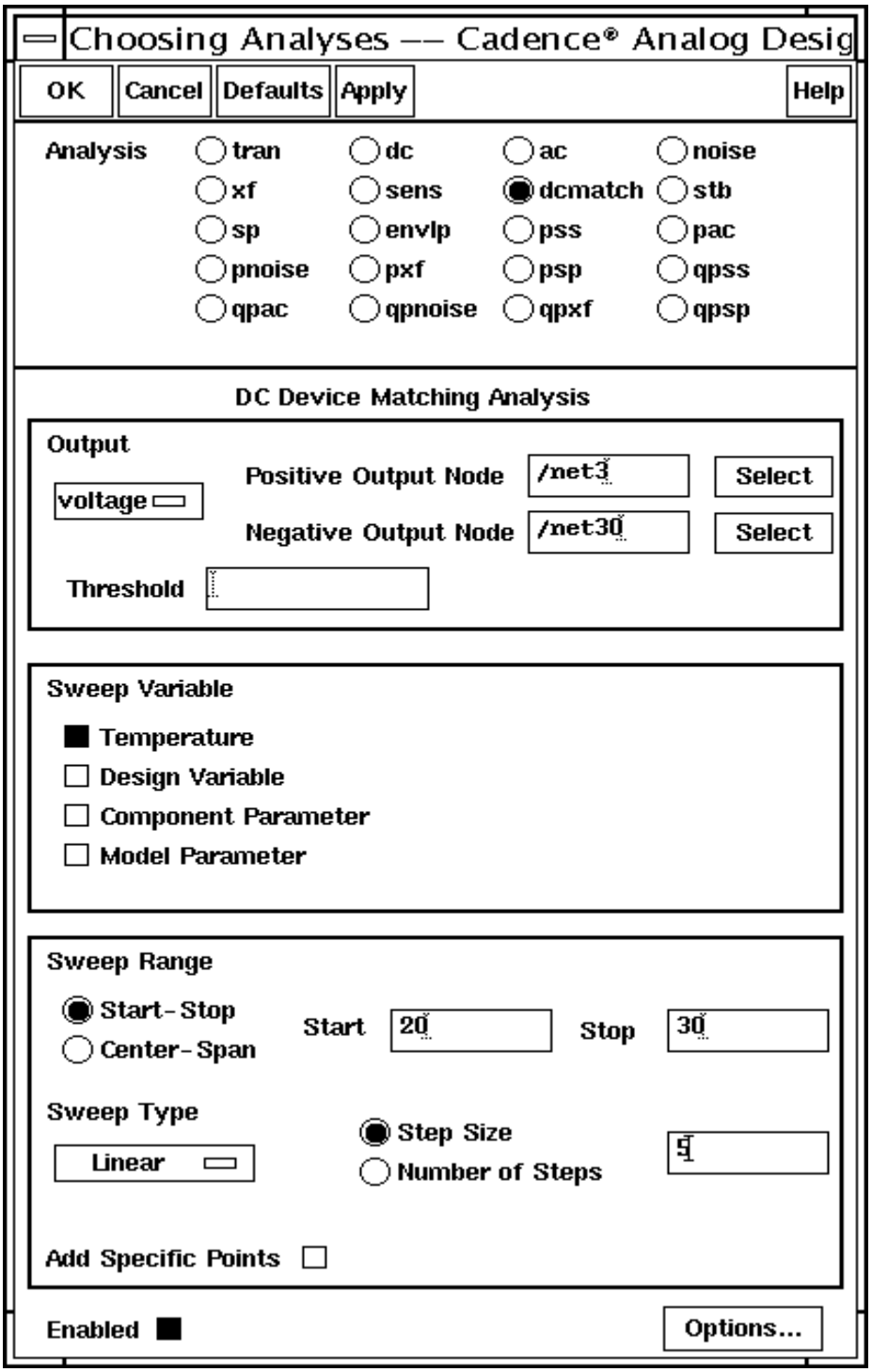
If the selected probe has multiple ports (for example tline), you can specify the port number in the *Port* field.



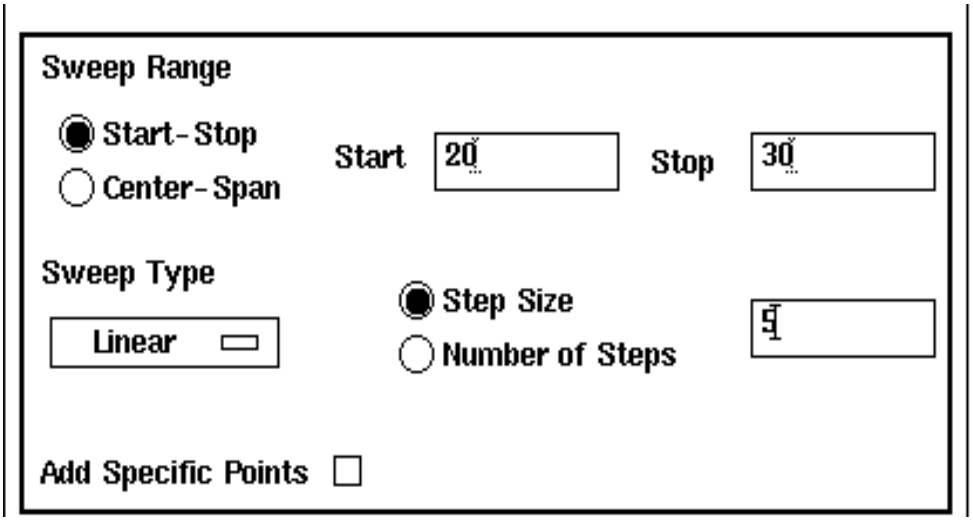
Refer to the Component Description Format User Guide for information on creating more library components selectable for an analysis.

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.



6. Specify the *Sweep Range and Sweep Type* for the swept parameter.



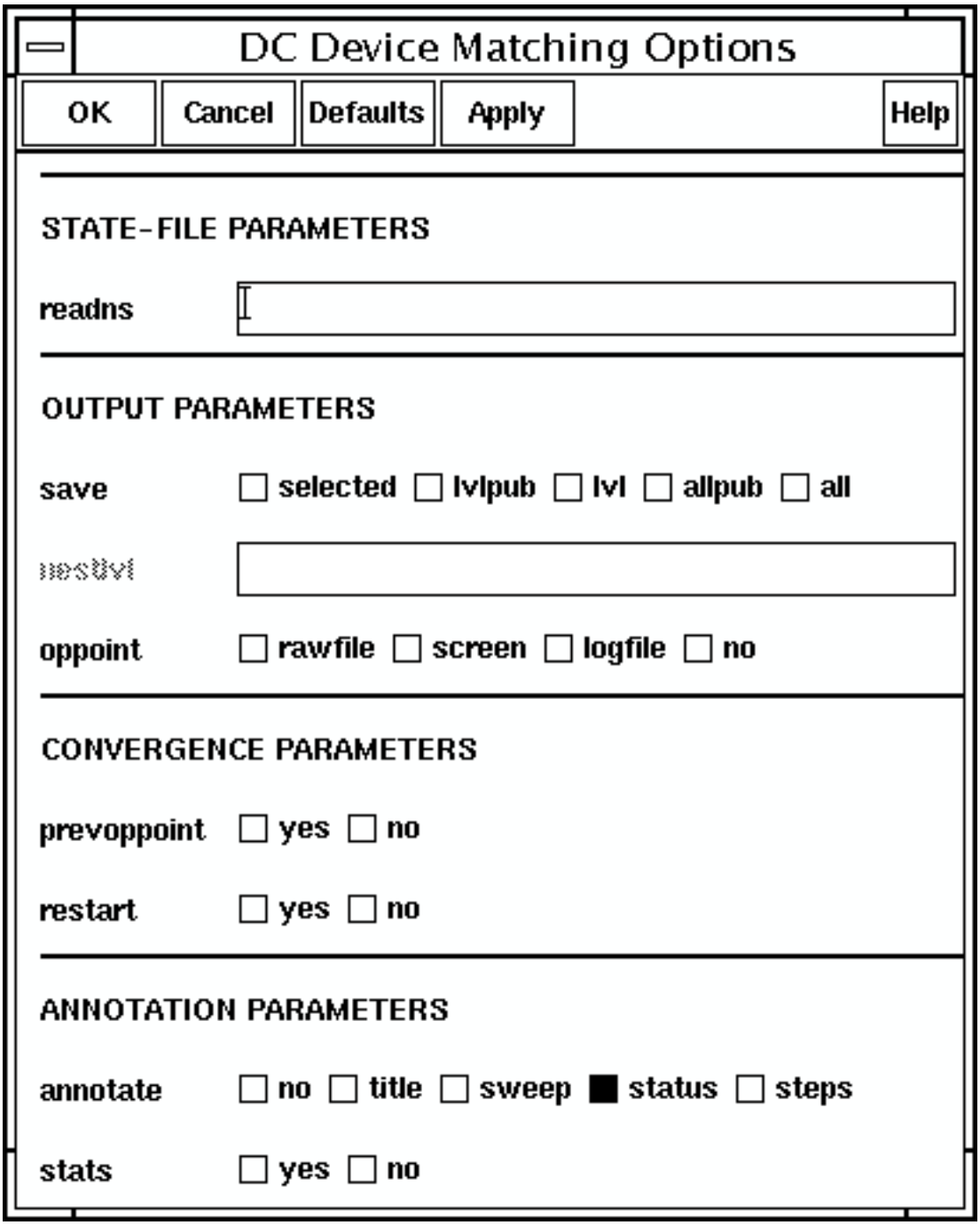
Enter the start and stop points of the range or the center and span of the range.

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=[…]

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.



Refer to the Spectre Circuit Simulator Reference for details.

8. Click *Apply*.

To access the results post simulation, choose *Results->Print->Mismatch Summary*.

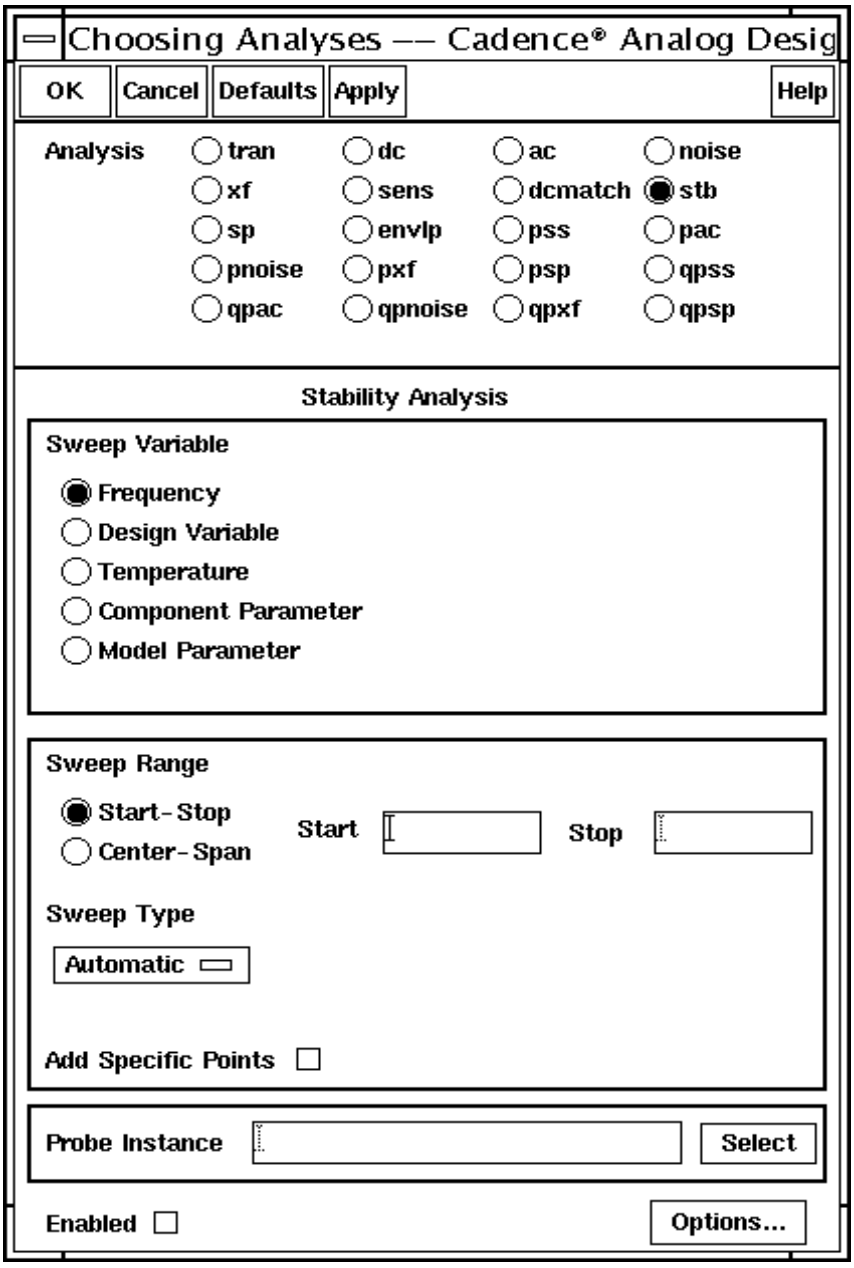
Note: To print these results from OCEAN, use the OCEAN command, `dcmatchSummary`.

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 *Cadence® Analog Design Environment* window. The *Choosing Analyses* form appears.

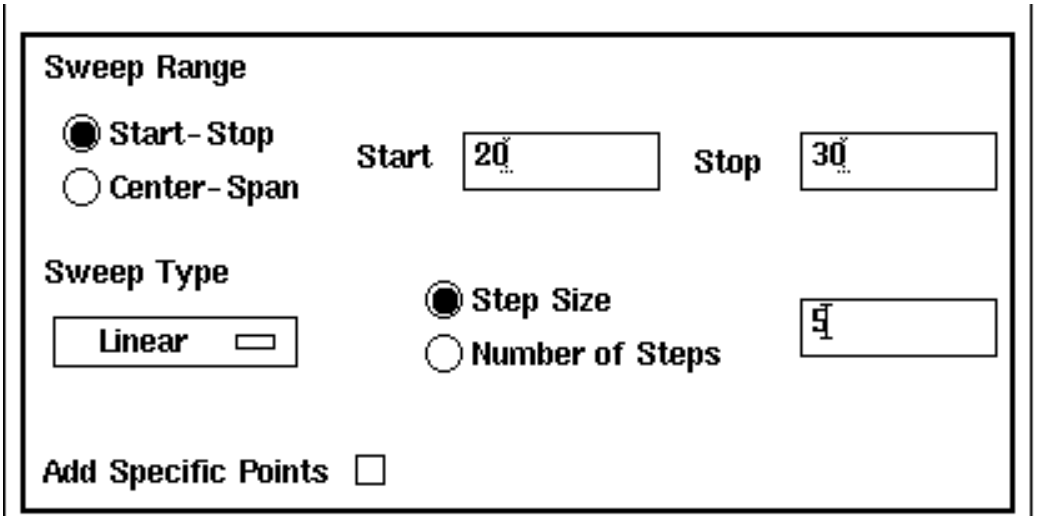
2. Choose *stb*. The *Choosing Analyses* form re-displays to show the fields that are required for the stability analysis.



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.



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:

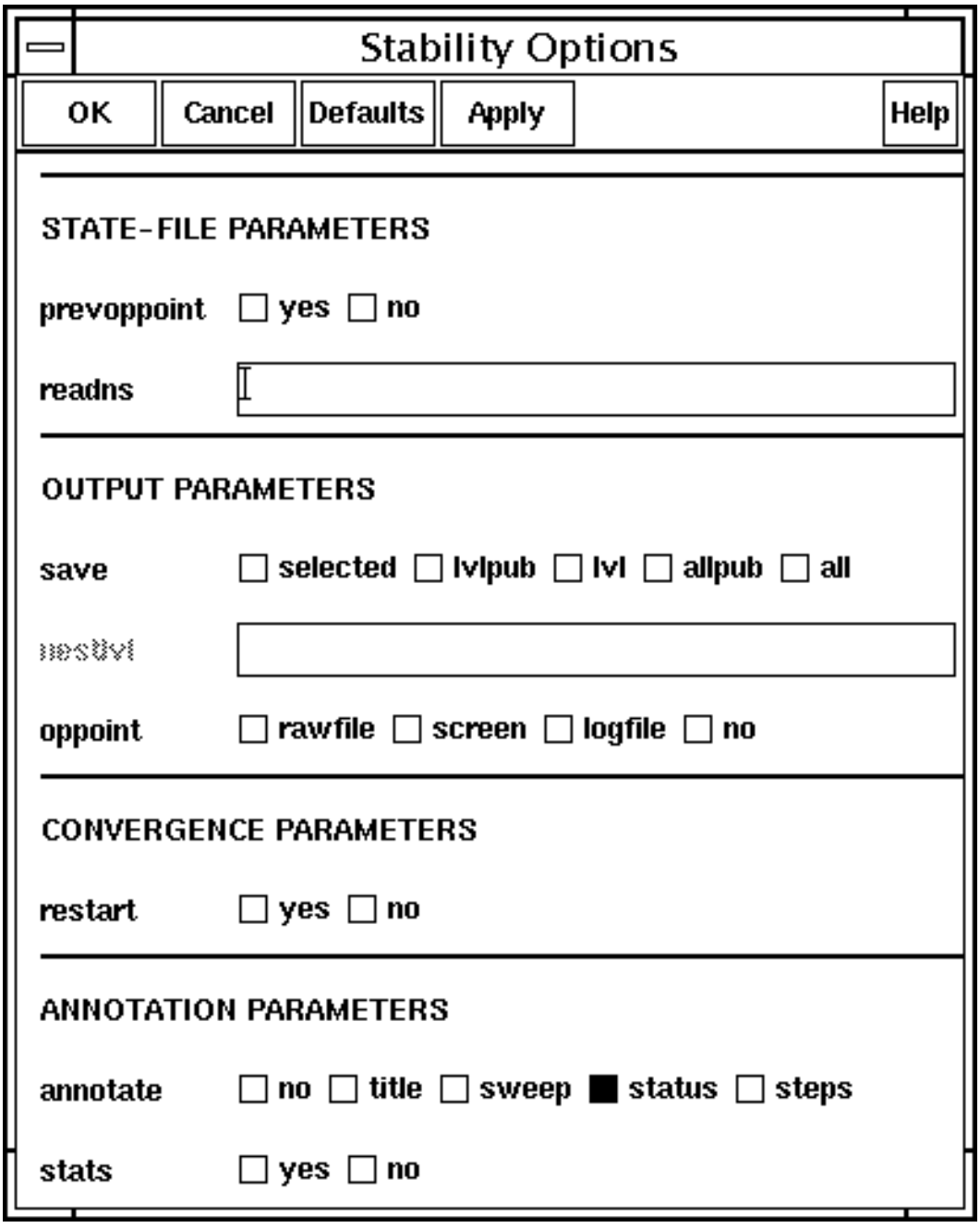
- Linear + Step Size = step
- Linear + Total Points = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Total Points = log
- Add Specific Points = values=[…]

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.

.



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.

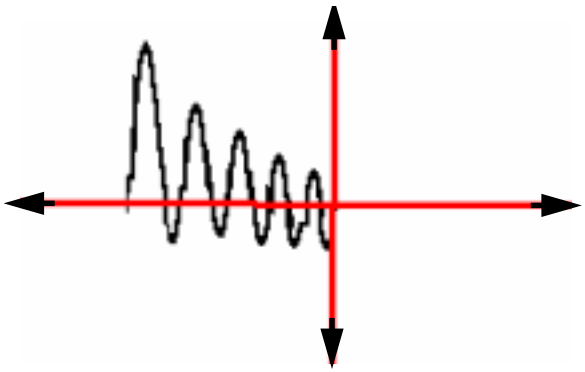
The behavior of the network depends upon the location of the poles and zeros on the complex S-plane. The poles are called natural frequencies of the network.

For example:

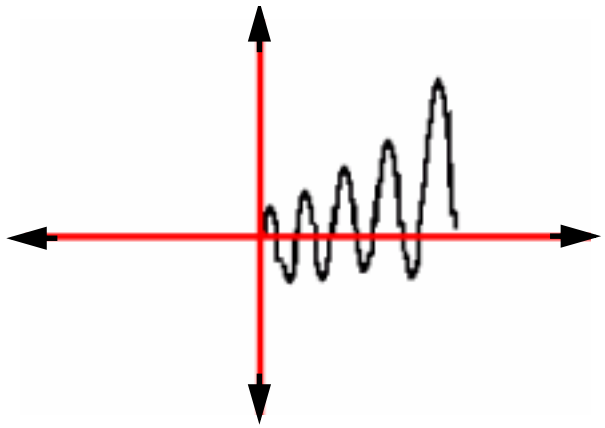
$$H(S) = \frac{(S-2) \cdot (S+1)}{S}$$

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 theXY plane, the circuit is considered *unstable*. The following diagram illustrates the behavior of an unstable circuit.

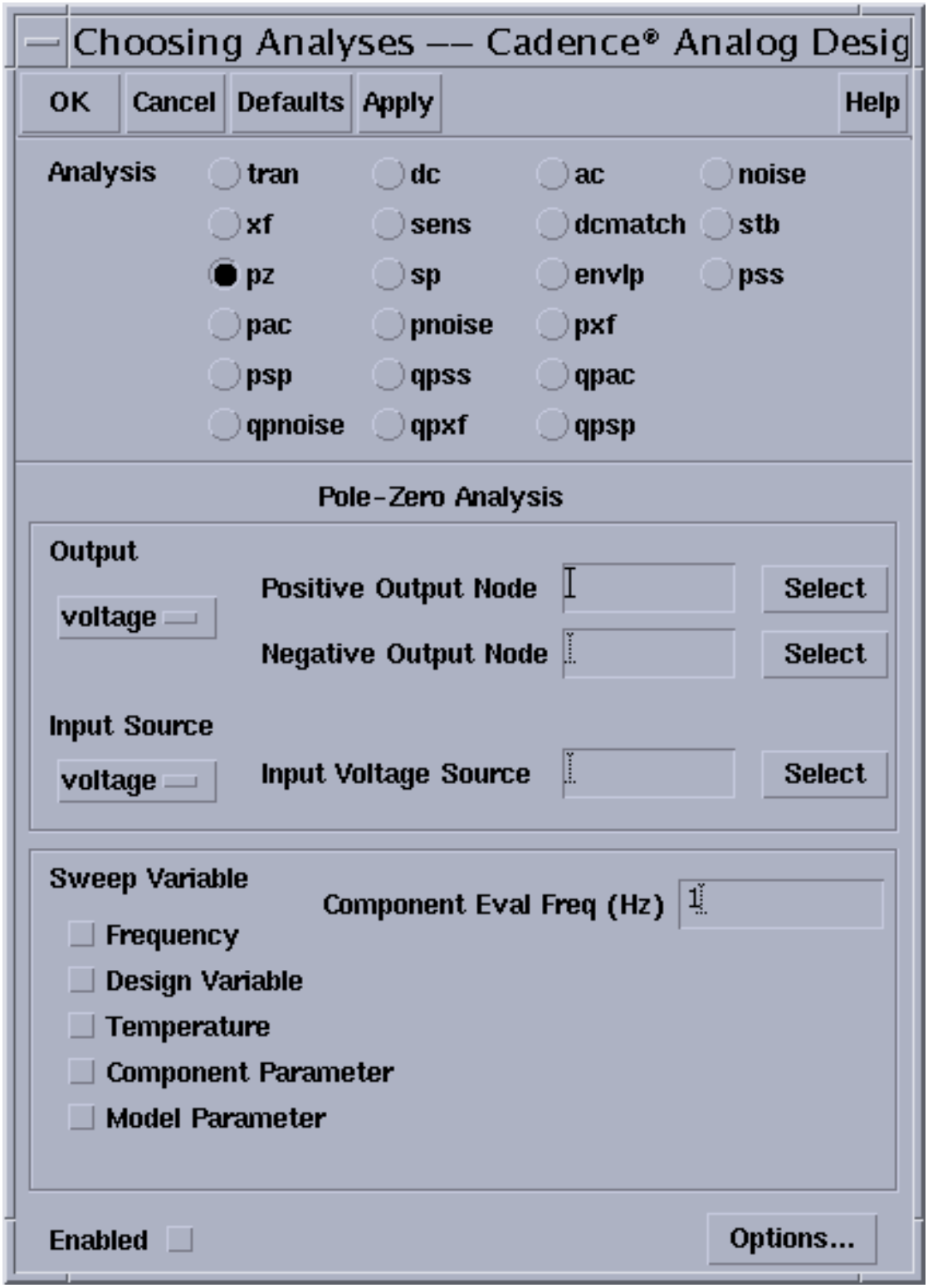


For absolute stability, there can be no poles with positive real parts. If there are poles with positive real parts the output signal may become unbounded.

To set up a *Pole Zero* analysis for the Spectre simulator,

1. Select *Analyses->Choose* from the *Cadence® Analog Design Environment* window. The *Choosing Analyses* form appears.

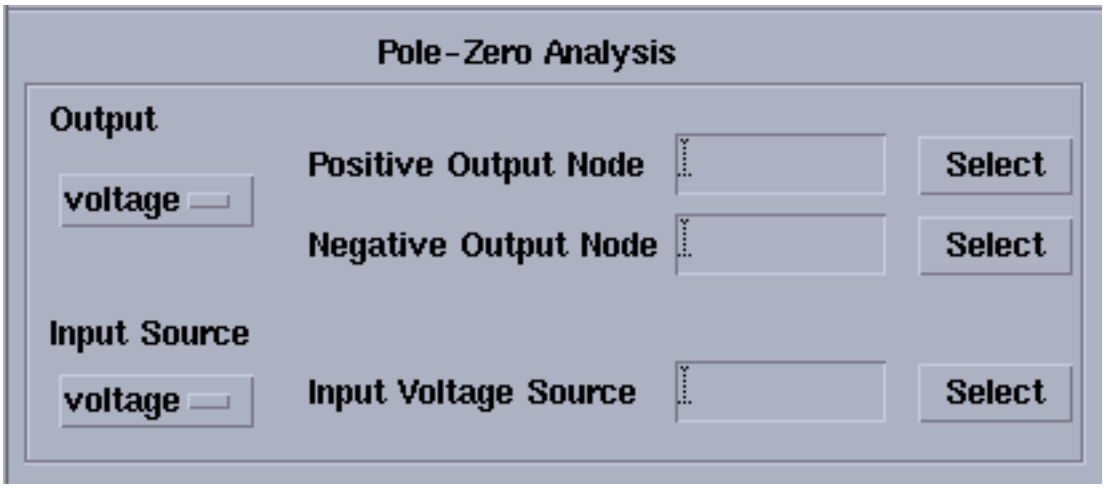
2. Select *pz*. The *Choosing Analyses* form re-displays to show the fields that are required for a *Pole Zero* analysis.


Choosing Analyses -- Cadence® 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
Component Eval Freq (Hz)
1
Frequency
Design Variable
Temperature
Component Parameter
Model Parameter
Enabled
Options...

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,

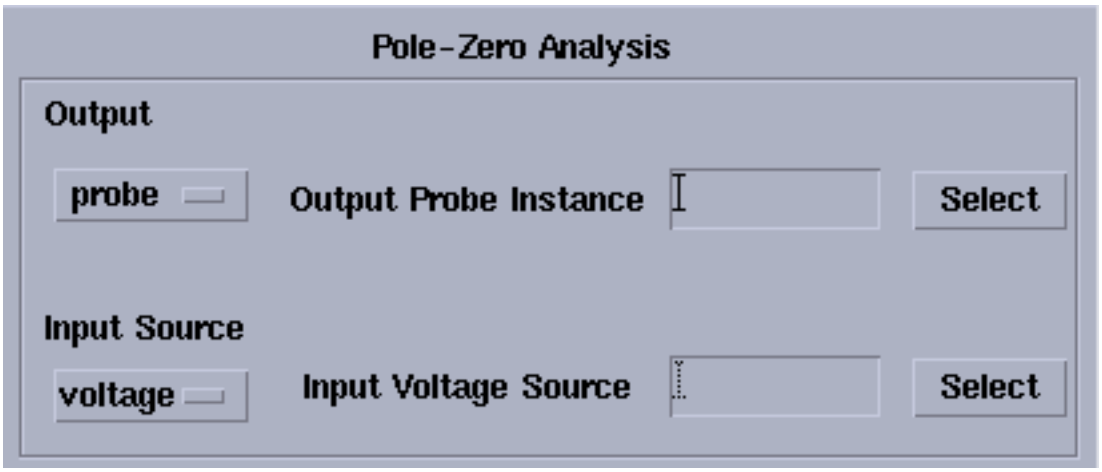
a. Choose *Voltage* in the cyclic drop down field.



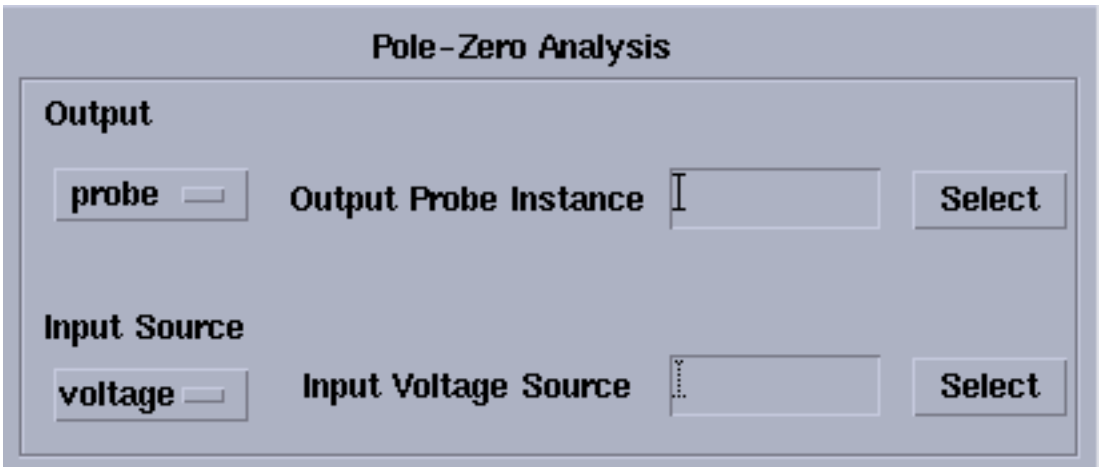
b. Click *Select* opposite *Positive Output Node* and click a net in the schematic for the positive output node. Also, click *Select* opposite *Negative Output Node* and click a net in the schematic.

To specify a current output,

a. Choose *Probe* in the cyclic drop down field.



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:

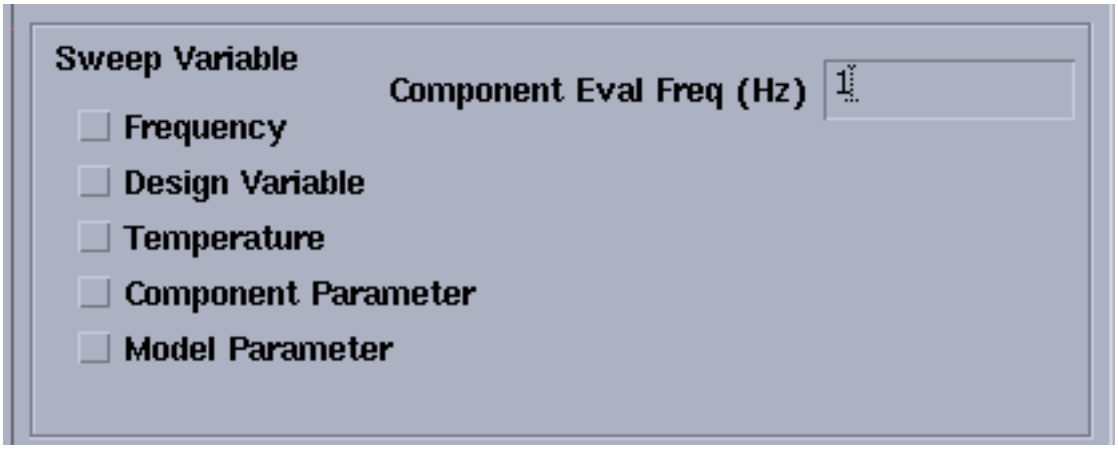
Device	**Corresponding analogLib Cells**
inductor	ind, pinductor
vsource	vdc, vpulse, vpwl, vpwlf, vsin, vexp, vsource
switch	sp1tswitch, sp2tswitch, sp3tswitch, sp4tswitch
tline	tline
controlled voltage source	vcvs, ccvs, sccvs, svcvs, zccvs, zvcvs, pvcvs, pvcvs2, pvcvs3, pccvs
iprobe	iprobe

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

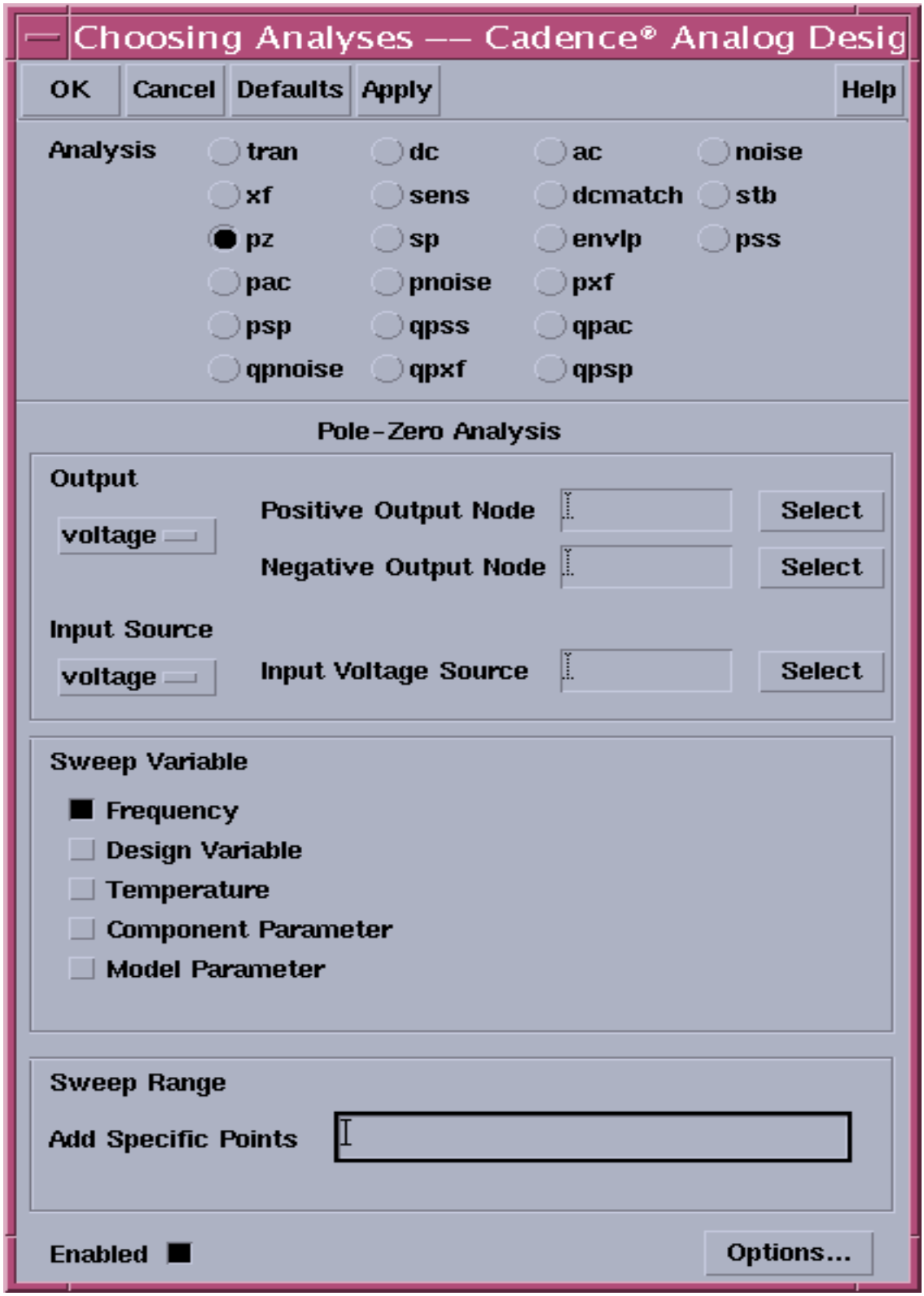
one of the device terminals would be the same as through the other. The *tline* device is the only one that has more than two terminals.

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.

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*.

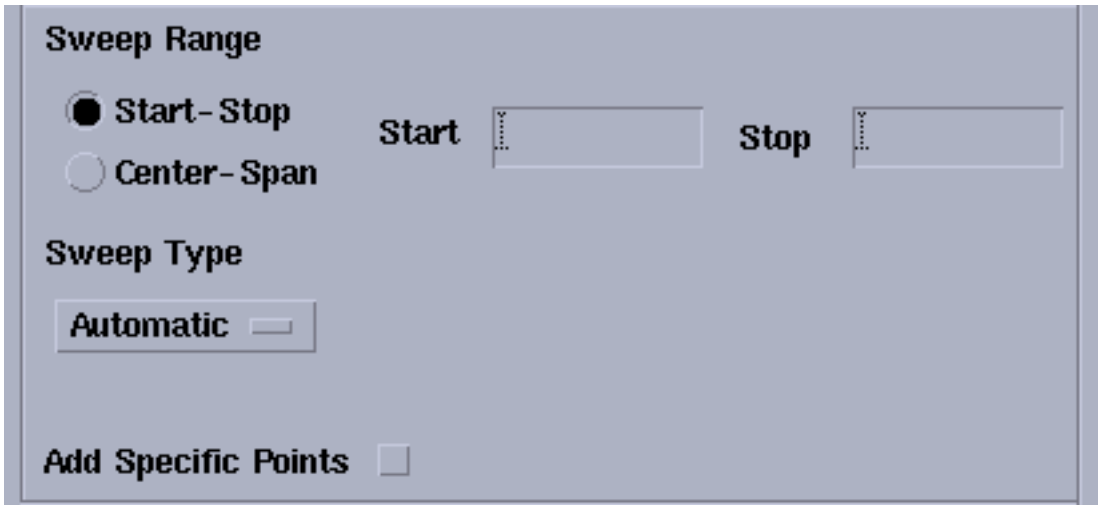


When any of these parameters are selected, the form re-displays according to the parameter that is selected:


Choosing Analyses -- Cadence® 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...

Note: The *Add Specific Points* field is only displayed if you enable the *Add Specific Points* button.

6. Specify the *Sweep Range and Sweep Type* for the swept parameter (*Design Variable*, *Temperature, Component Parameter* or *Model Parameter*)



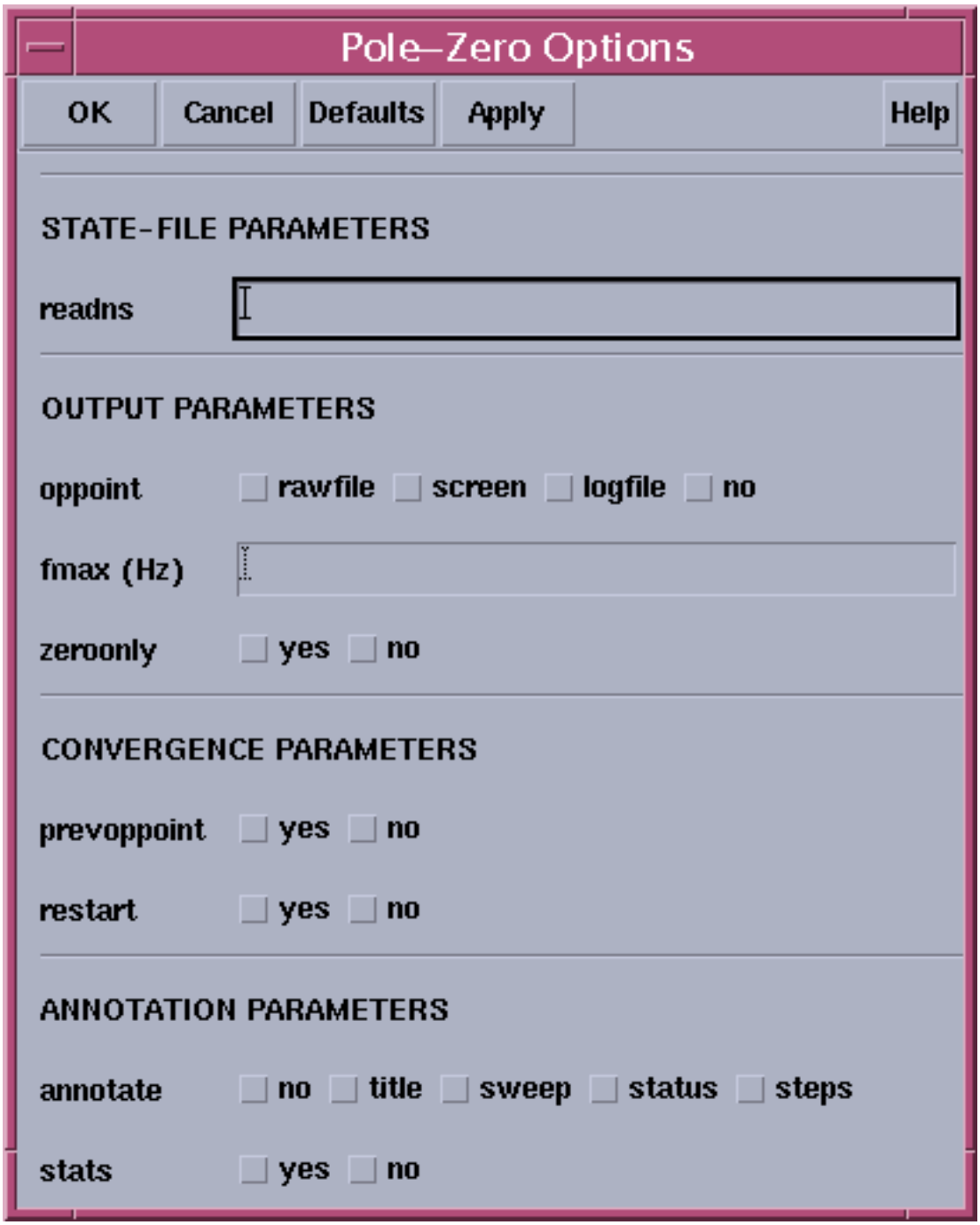
Enter the start and stop points of the range or the center and span of the range.

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=[…]

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.



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 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.

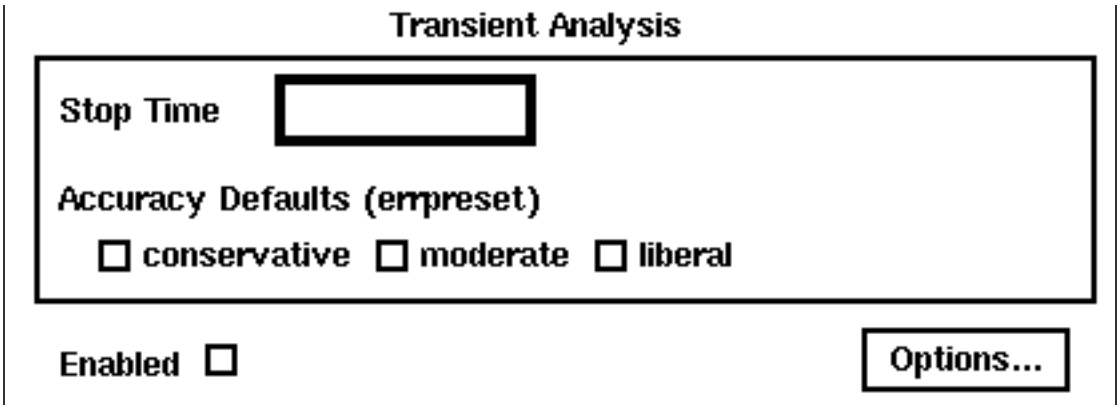
The next step is usually selecting the outputs you want to save.

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,



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

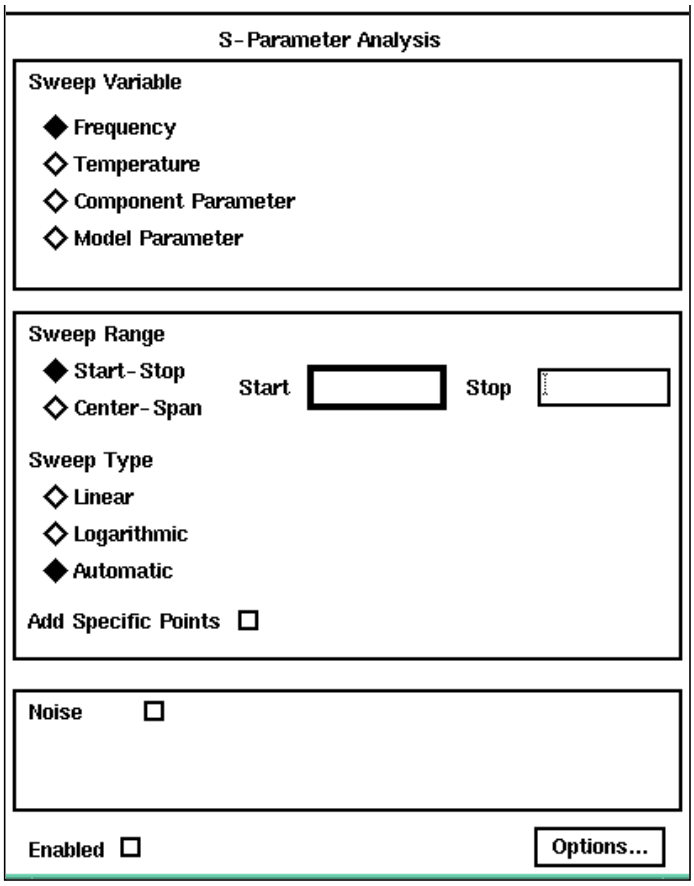
AC small-signal analysis linearizes the circuit about the DC operating point and computes the response to a given small sinusoidal stimulus. SpectreS can perform the analysis while sweeping a parameter.

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.



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.

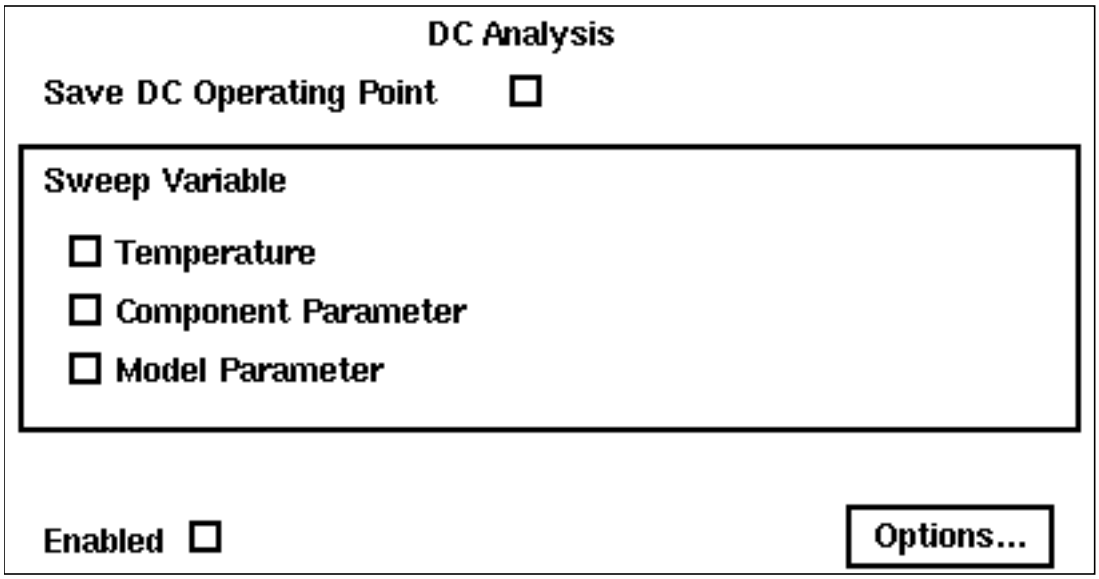
   The sweep type options are mapped to Spectre statements:
   - Linear + Step Size = step
   - Linear + Total Points = lin

- 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.



To save the DC operating point,

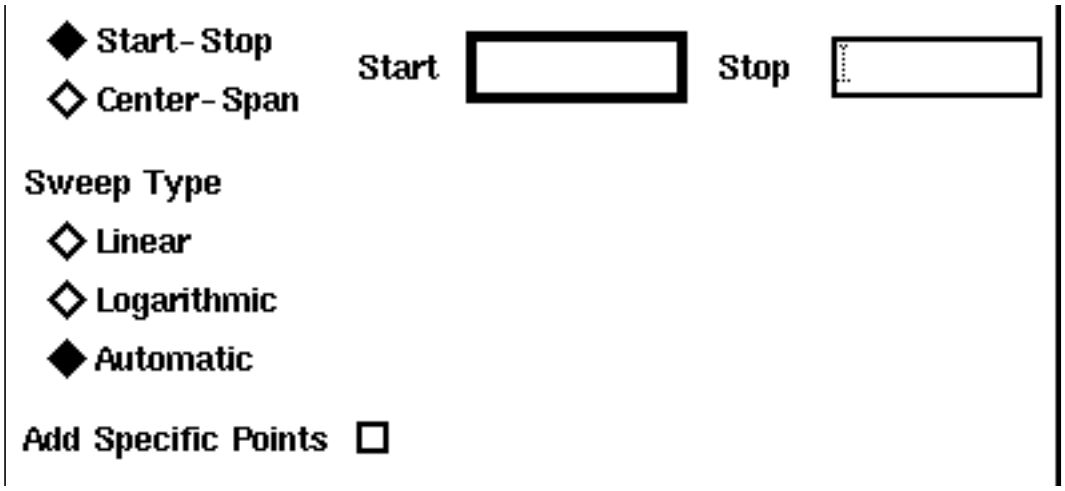
➤ Click *Save DC Operating Point*, click *Enabled*, and click *Apply*.

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.



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.

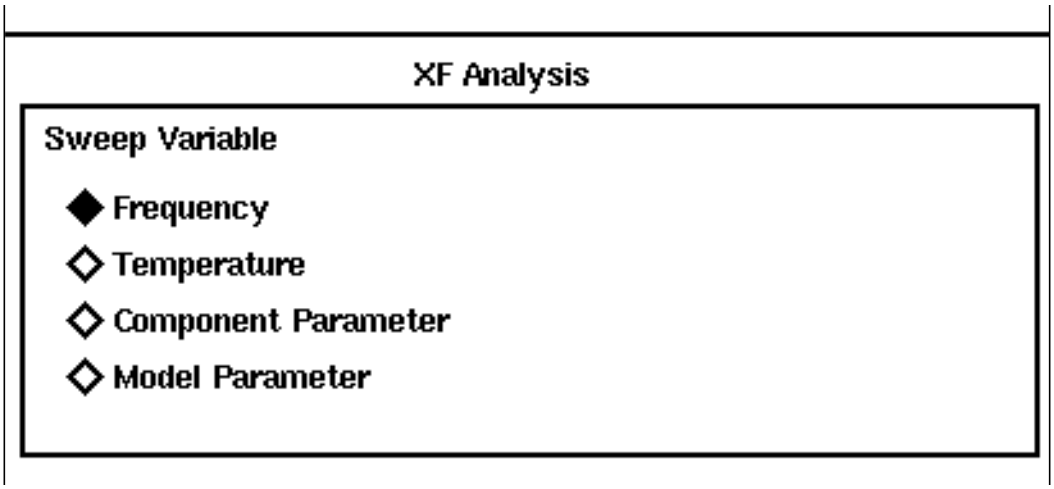
   The sweep type options are mapped to spectreS statements:
   - Linear + Step Size = step
   - Linear + Total Points = lin
   - Logarithmic + Points Per Decade = dec
   - Logarithmic + Total Points = log
   - Add Specific Points = values=[…]
4. Click *Options* to set the spectreS options controlling DC 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

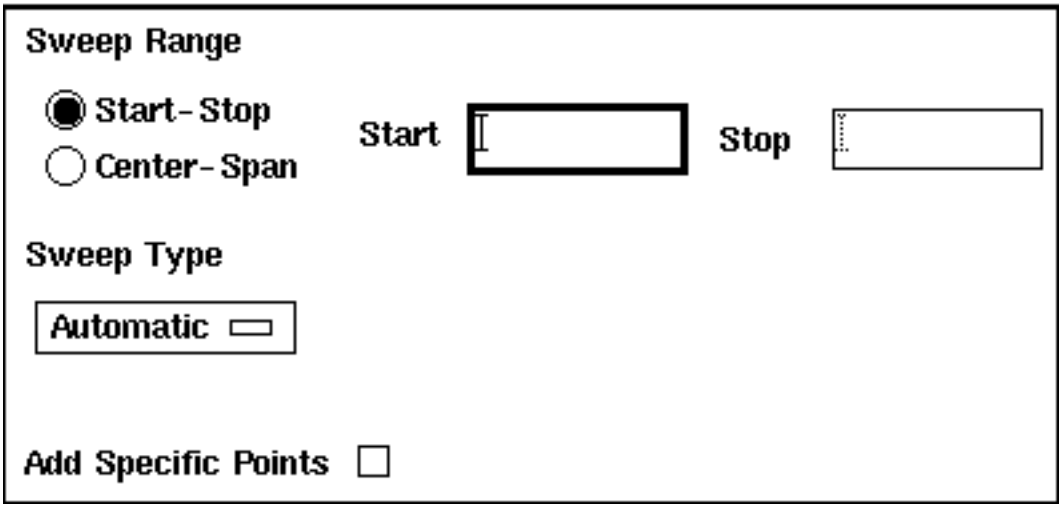
source or instance terminal in the circuit to a designated output. The variable of interest at the output can be voltage or current.

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.

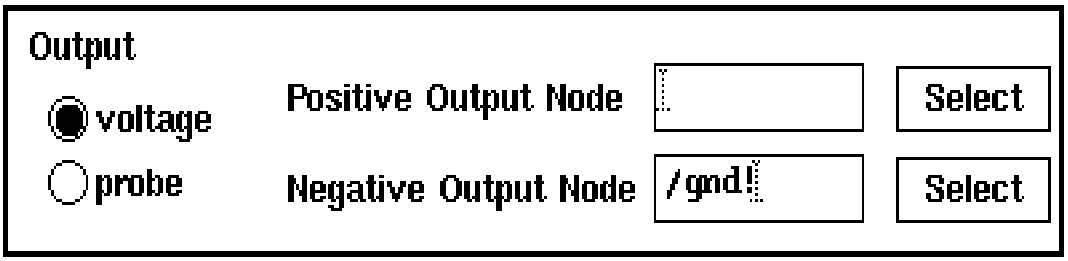
2. Specify the sweep range and type.



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 *probe*.



- 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.

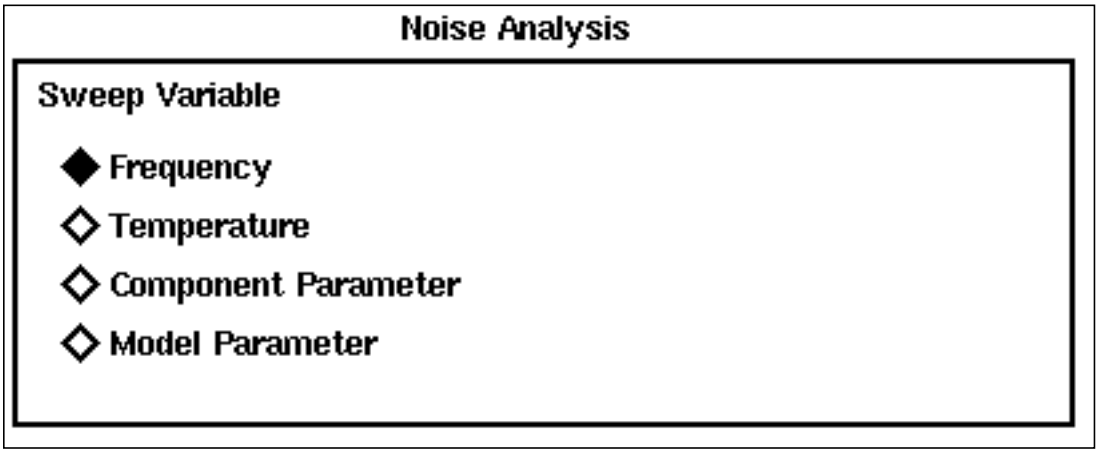
4. Click *Options* to set the spectreS options controlling transfer function simulation.

5. Click *Apply*.

Noise Analysis

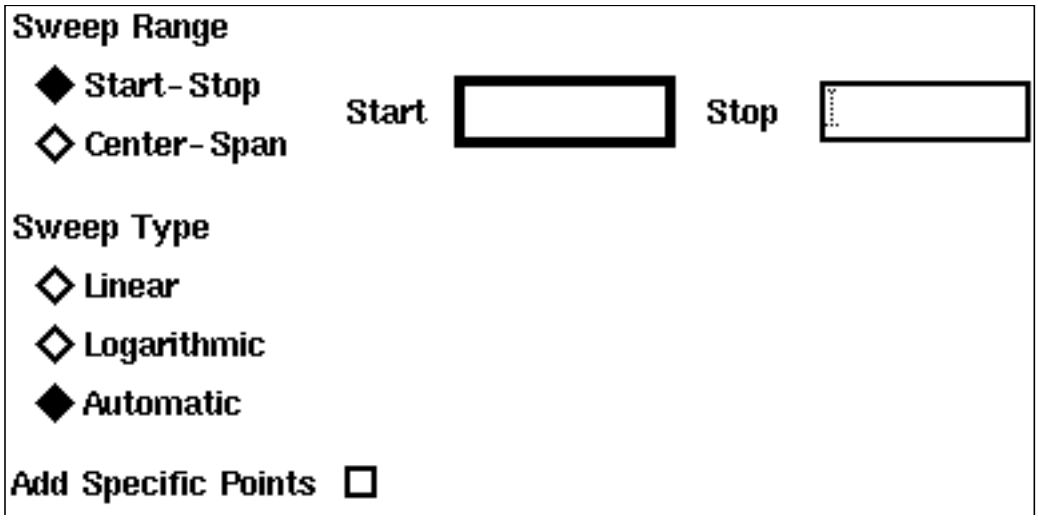
The noise analysis linearizes the circuit about the DC operating point and computes the total-noise spectral density at the output. If you specify an input probe, the transfer function and the input-referred noise for an equivalent noise-free network is computed. To set up a noise analysis,

1. Choose a sweep variable option and specify any necessary parameters.



- 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.

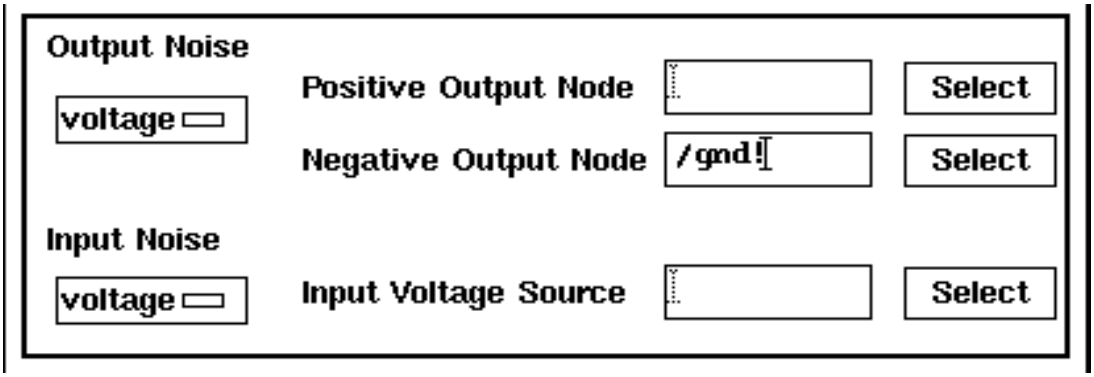


The sweep type options are mapped to Spectre statements:

- 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.



- 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 aSpectreS 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 won't 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

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

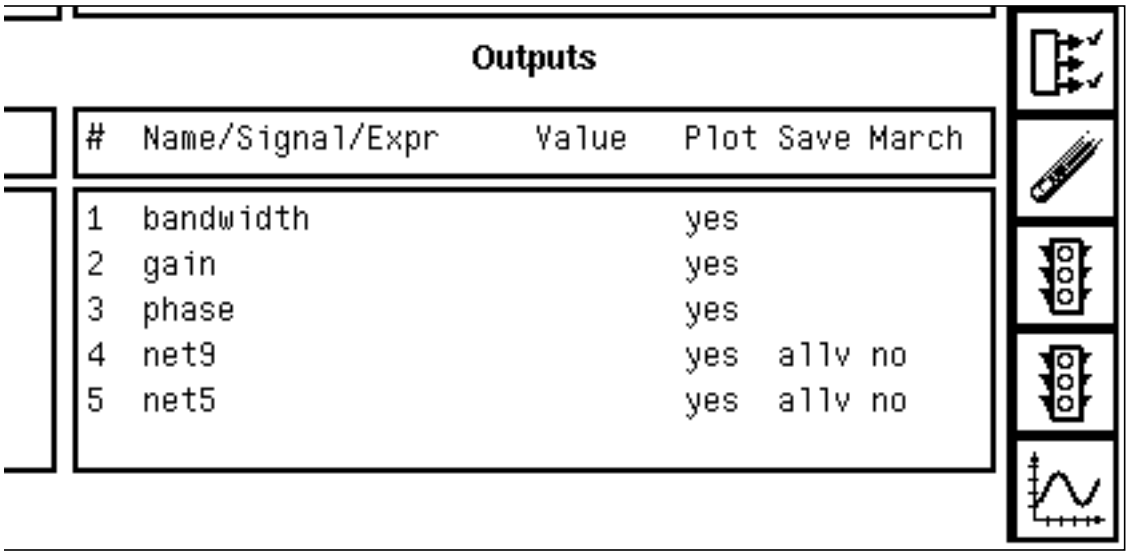
The Cadence® analog design environment keeps track of three sets of nets and terminals:

- 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.



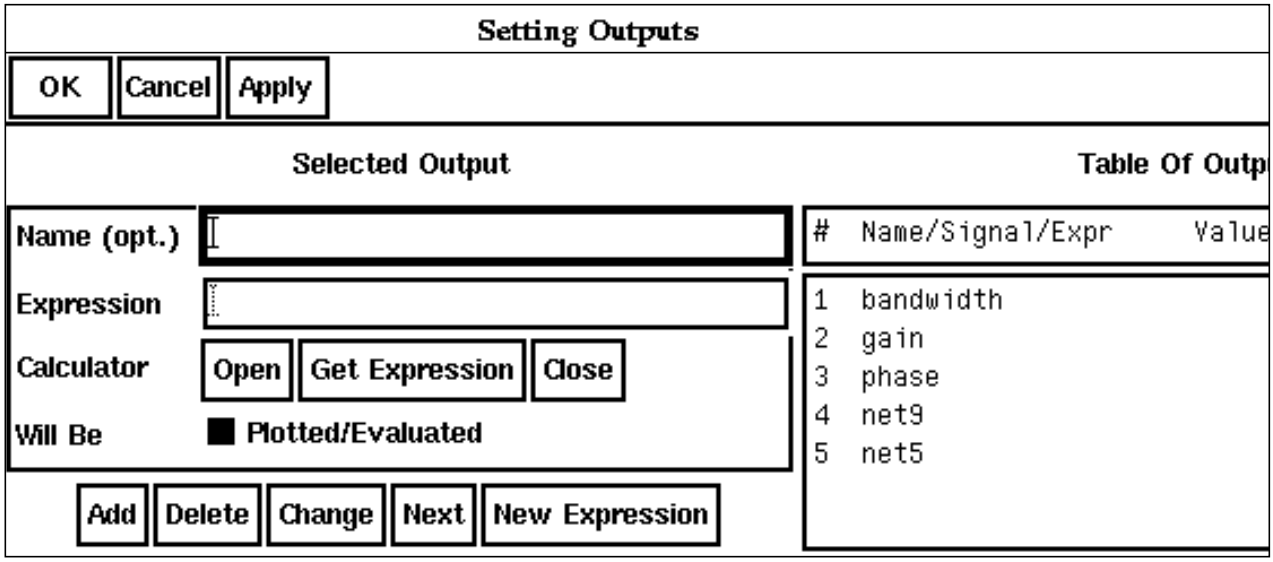
Note: If you click a net (*Name/Signal/Expr*) repeatedly, the highlighting will toggle on and off, and the signal will appear and disappear from the *Outputs* list.

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.



For detailed information about the form, see "Setting Outputs" on page 237.

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.

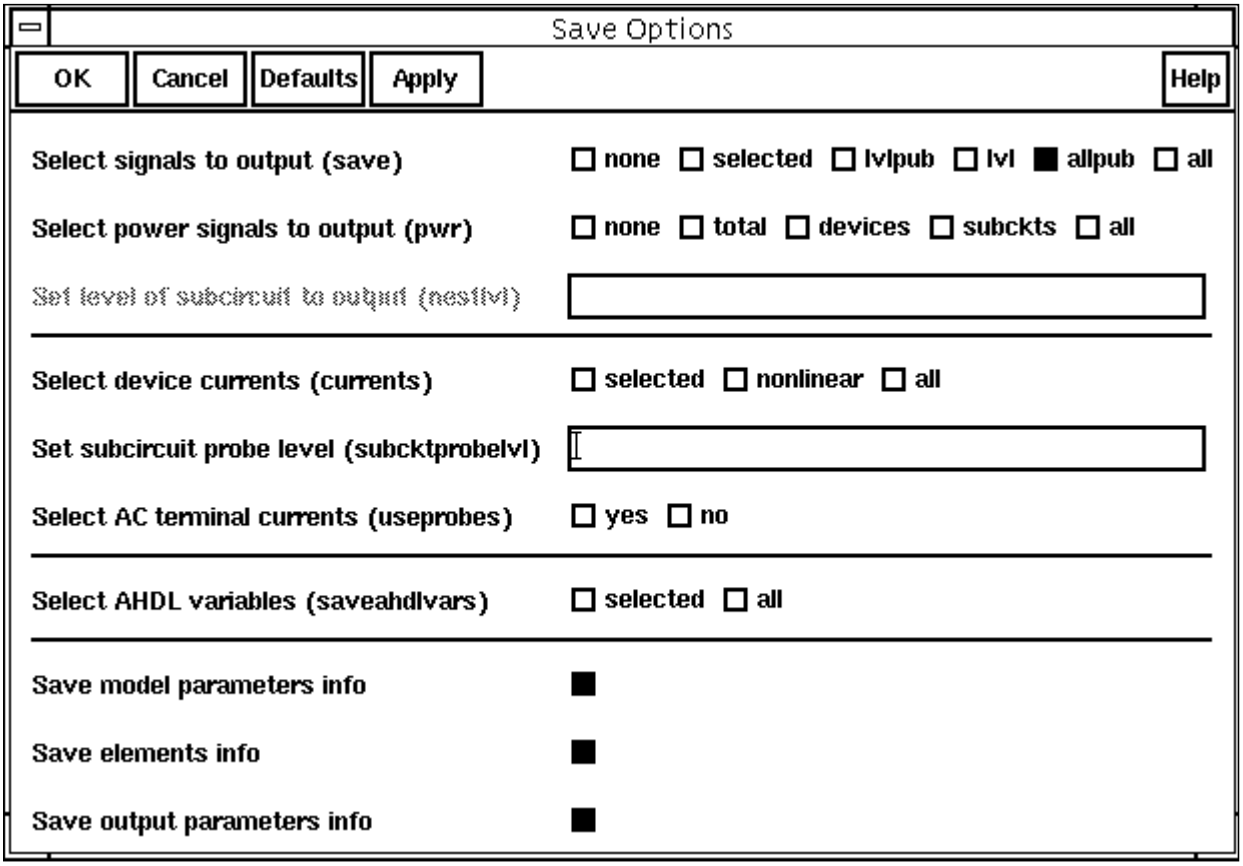
After you select the outputs you want to save, the next step is generally to start the simulation.

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 238.

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.

For details about selecting device currents (*currents*), setting subcircuit probe level (*subcktprobelvl*) and selecting AC terminal currents (*useprobes*) refer to the the "Specifying Output Options" chapter of the *Spectre Circuit Simulator User Guide.*

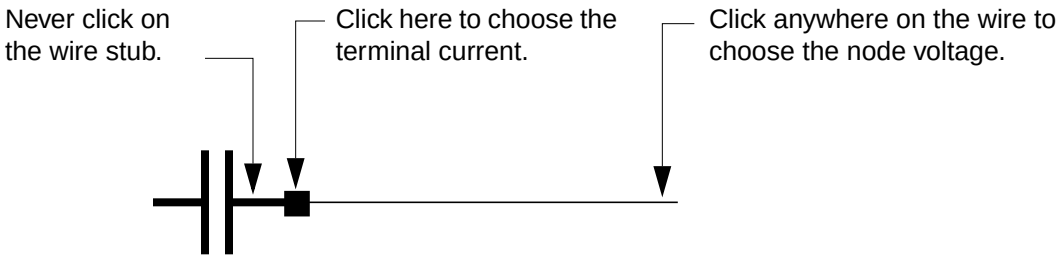
Saving Selected Voltages or Currents

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.



3. Press the Esc key when you finish.

Adding a Node or Terminal to a Set

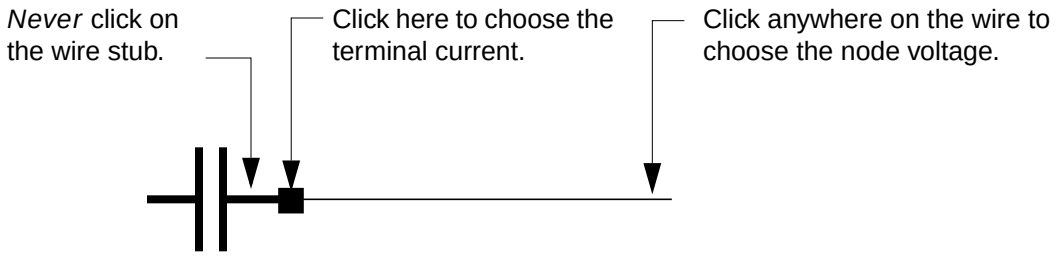
To add a node or terminal to the saved, marched, or plotted sets,

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.



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*

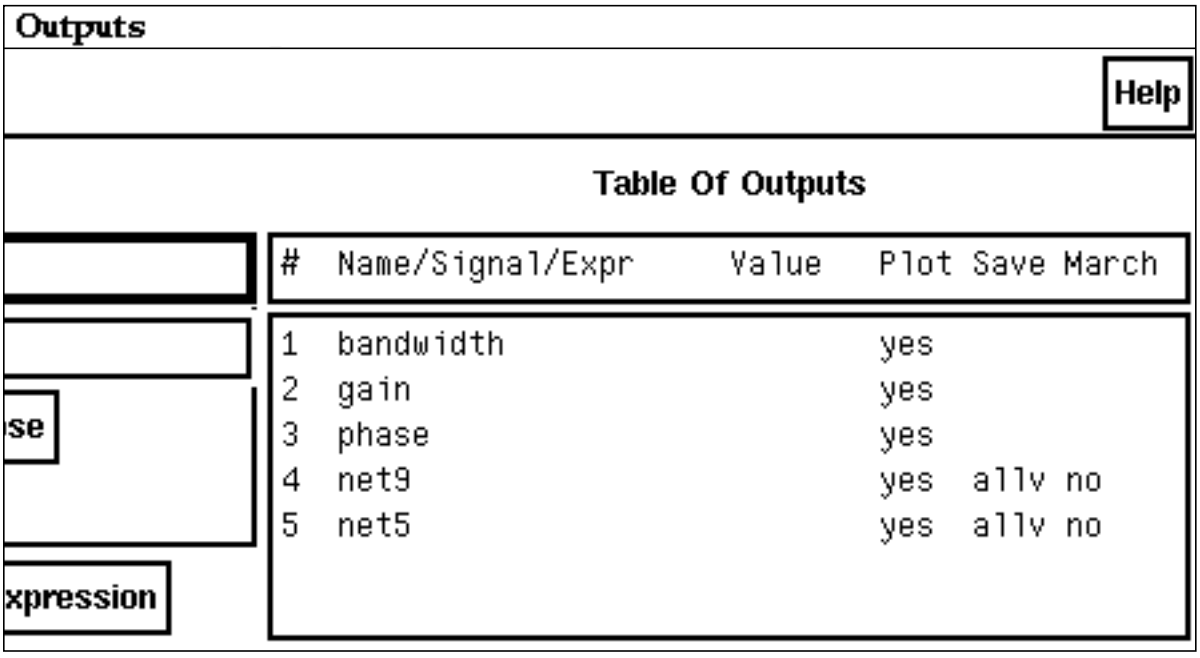
The outputs table is updated to show the outputs have been added to the saved or marched sets.

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.



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

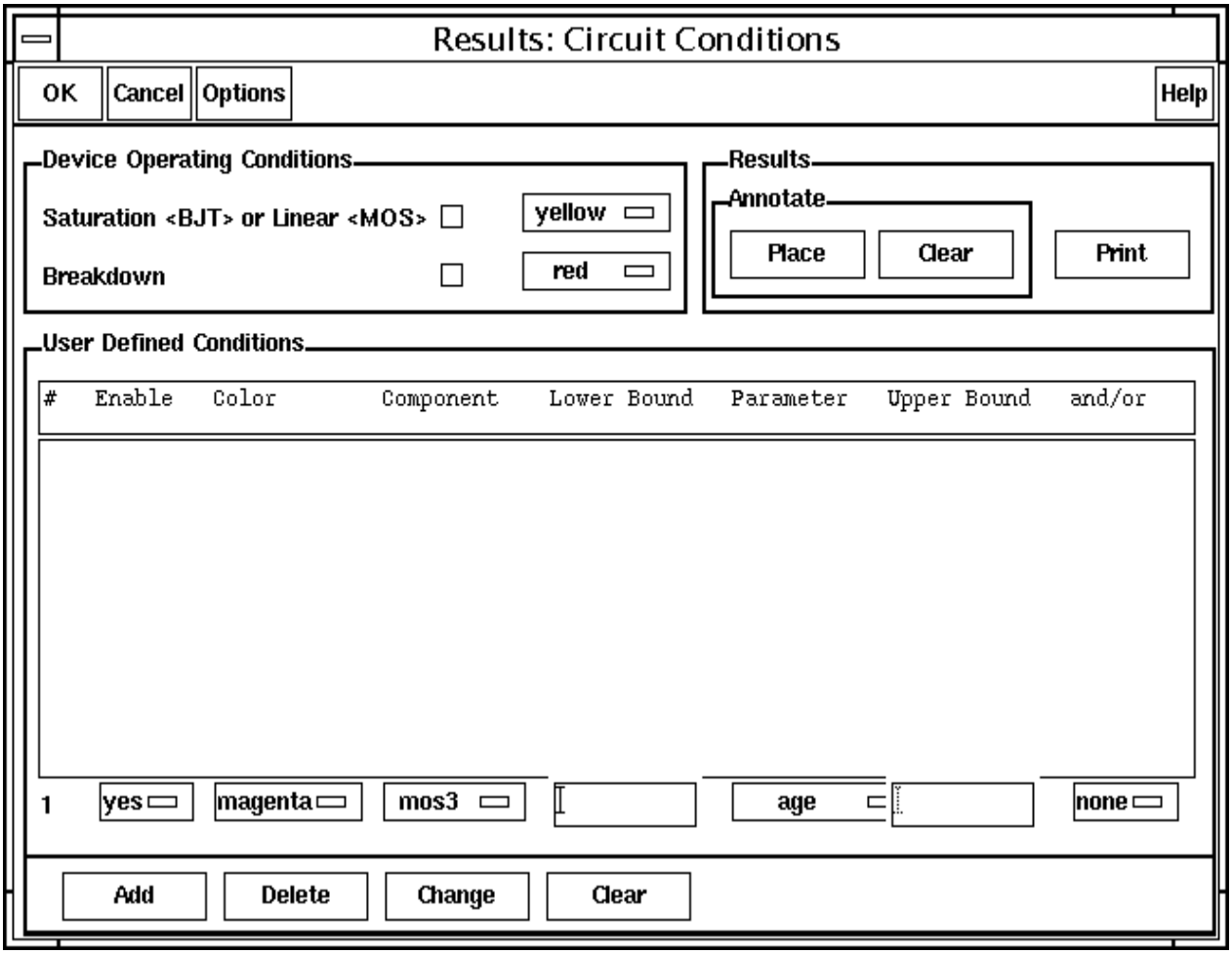
After running a simulation, you can search the results for components in the saturation region, breakdown region, or any user-defined region. To do a conditional search for results, choose *Results – Circuit Conditions* from the *Simulation* menu. Follow the procedure below to search for circuit conditions.

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:



For detailed information about the form, see "Circuit Conditions" on page 234.

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.

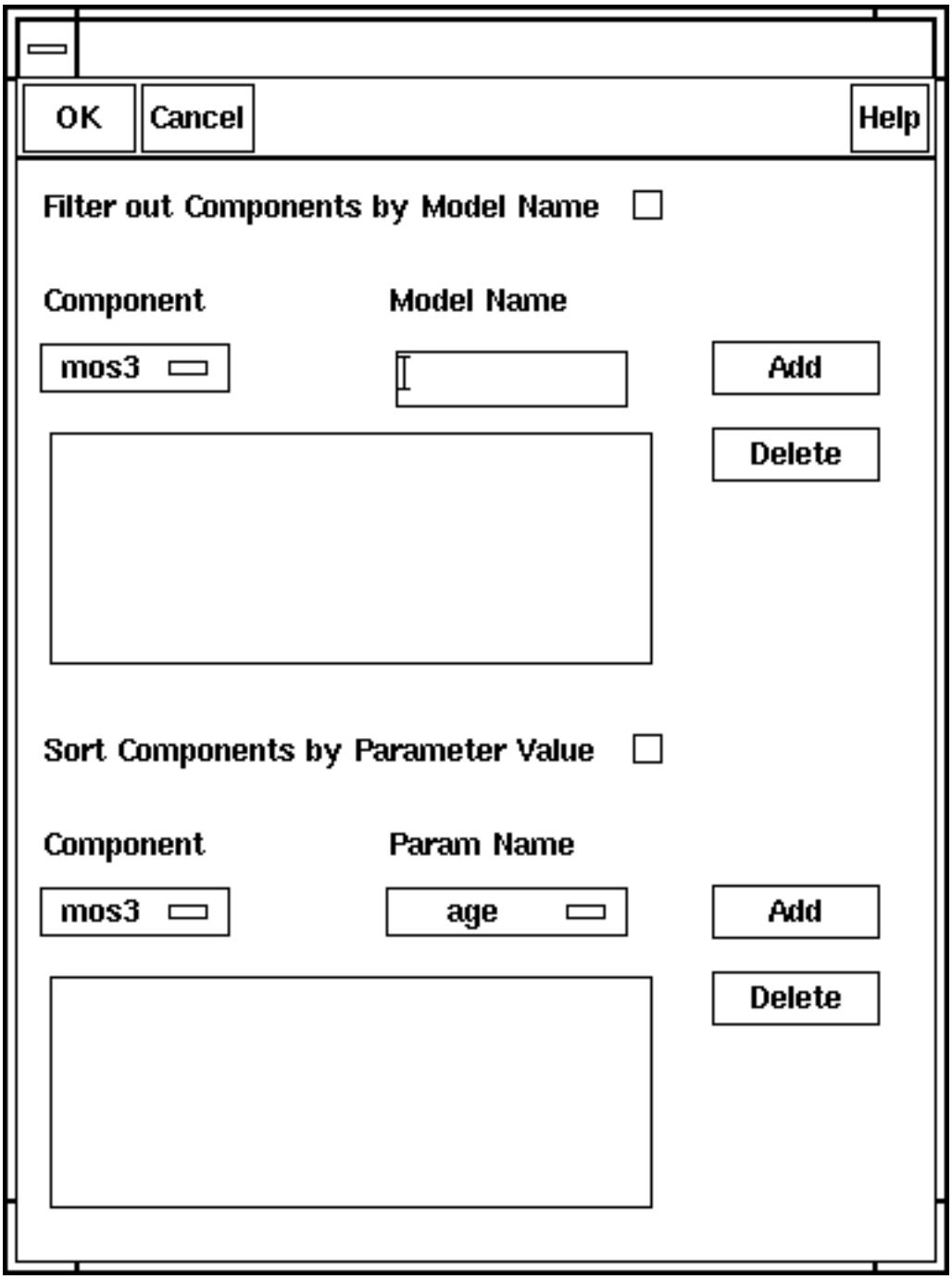
5. Set up *User Defined Conditions*.

   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.



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 239.

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
Set level of subcircuit to output (nestlvl)	Saving Groups of Signals
Select device currents (currents)	Saving Groups of Currents
Set subcircuit probe level (subcktprobelvl)	Saving Subcircuit Terminal Currents
Select AC terminal currents (useprobes)	Setting Multiple Current Probes
Select AHDL variables (saveahdlvars)	Saving All AHDL Variables

Caution

The** all **buttons (**Select signals to output (save)* ***) are global buttons but can be locally ovverriden 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 Cadence® 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

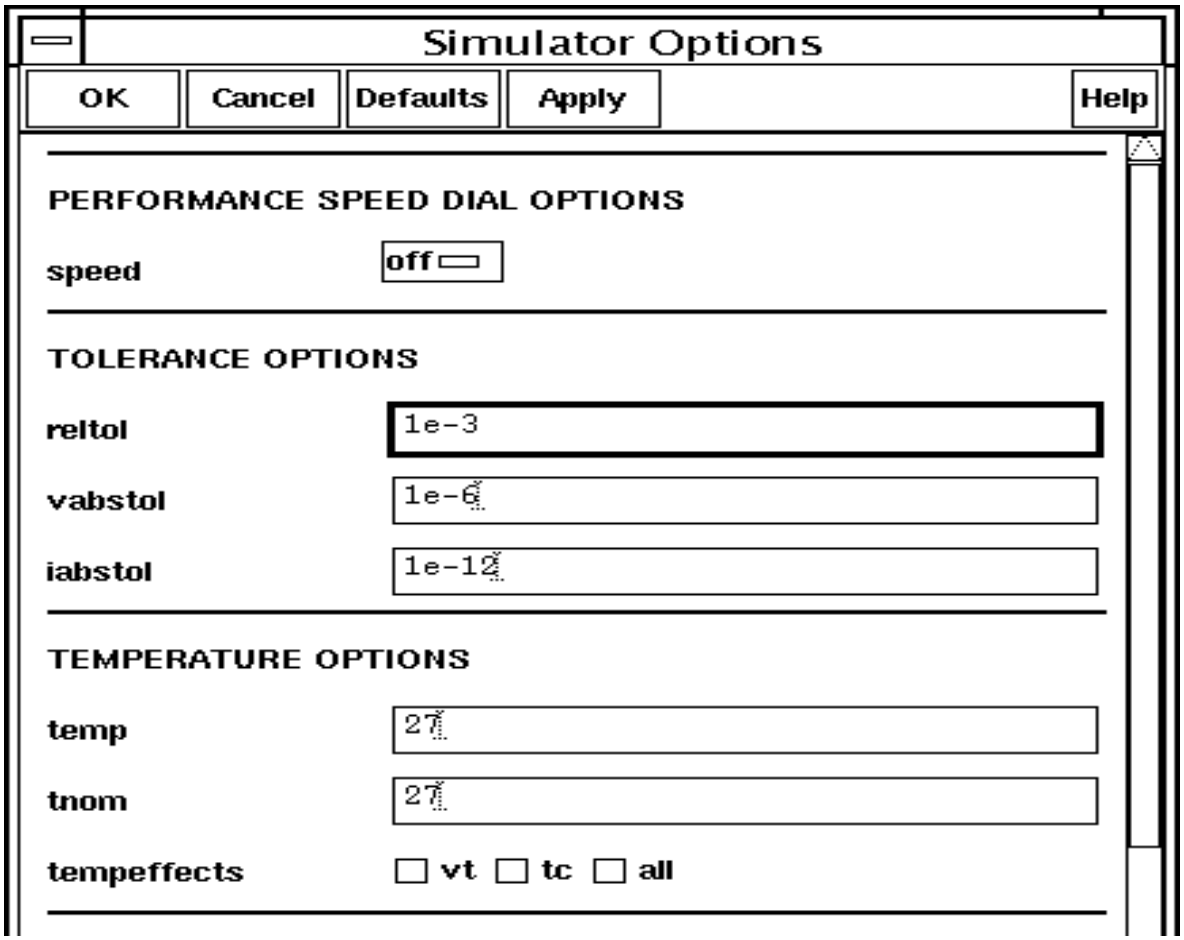
You set simulator-specific options and variables in two places:

- 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.

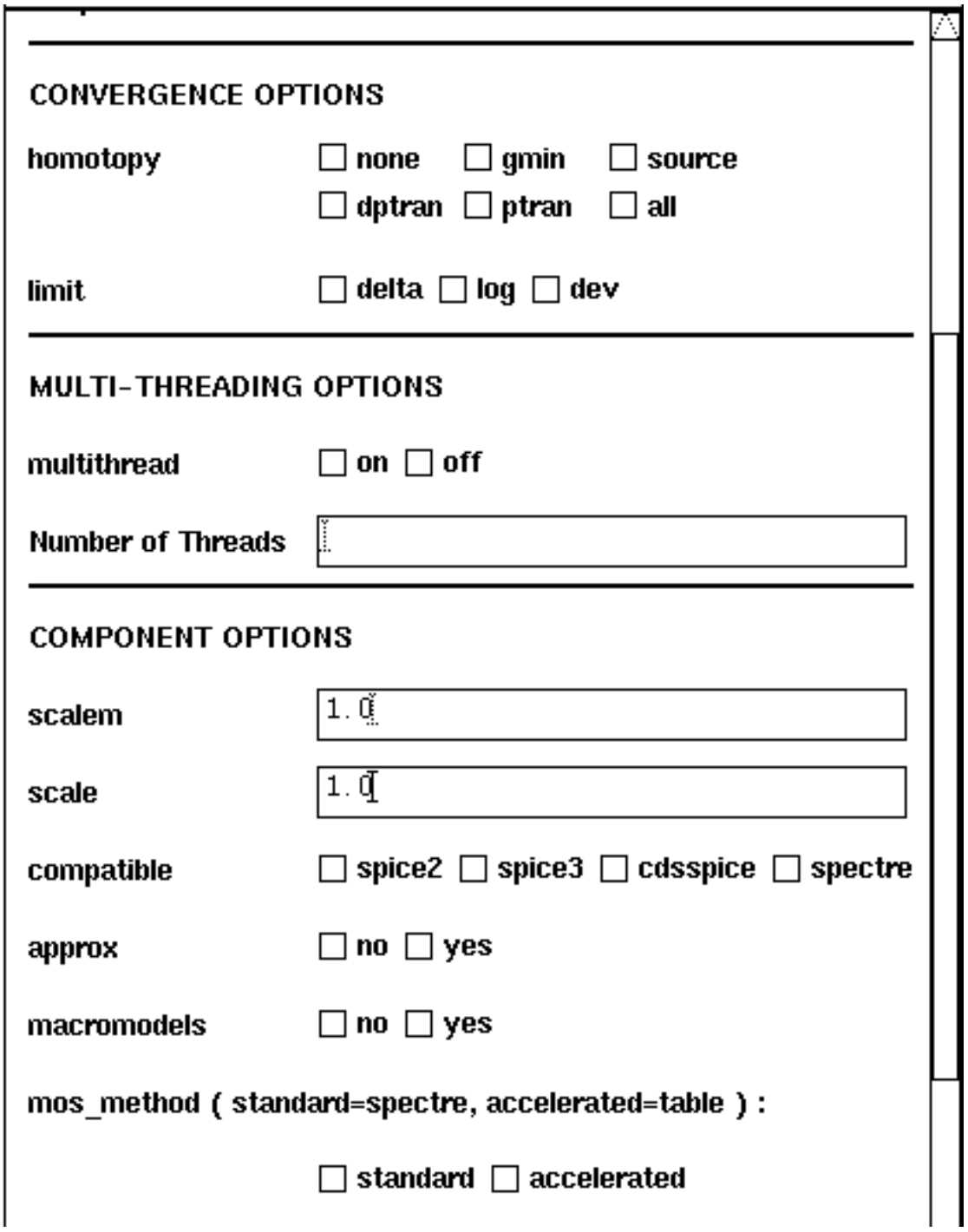
Each simulator has a different set of options.

Spectre Options

For the Spectre® circuit simulator, the *Simulation– 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.



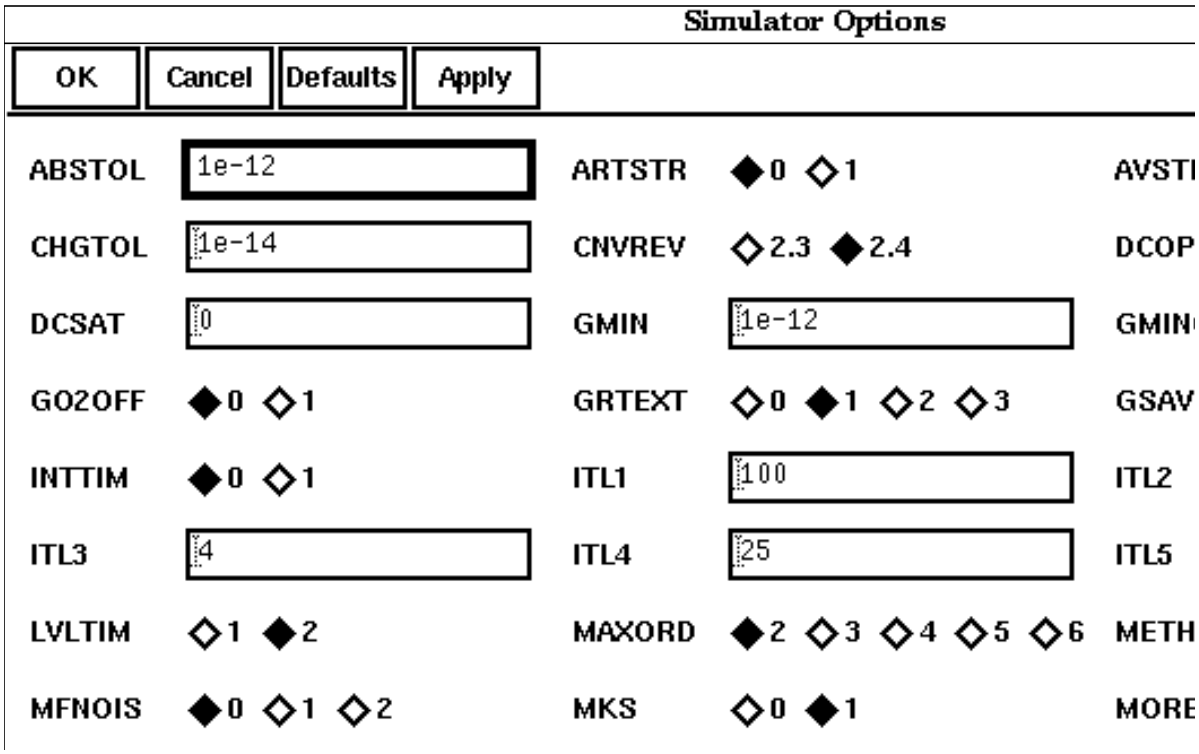
Some of the other simulator options are:



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.

Cadence SPICE Options

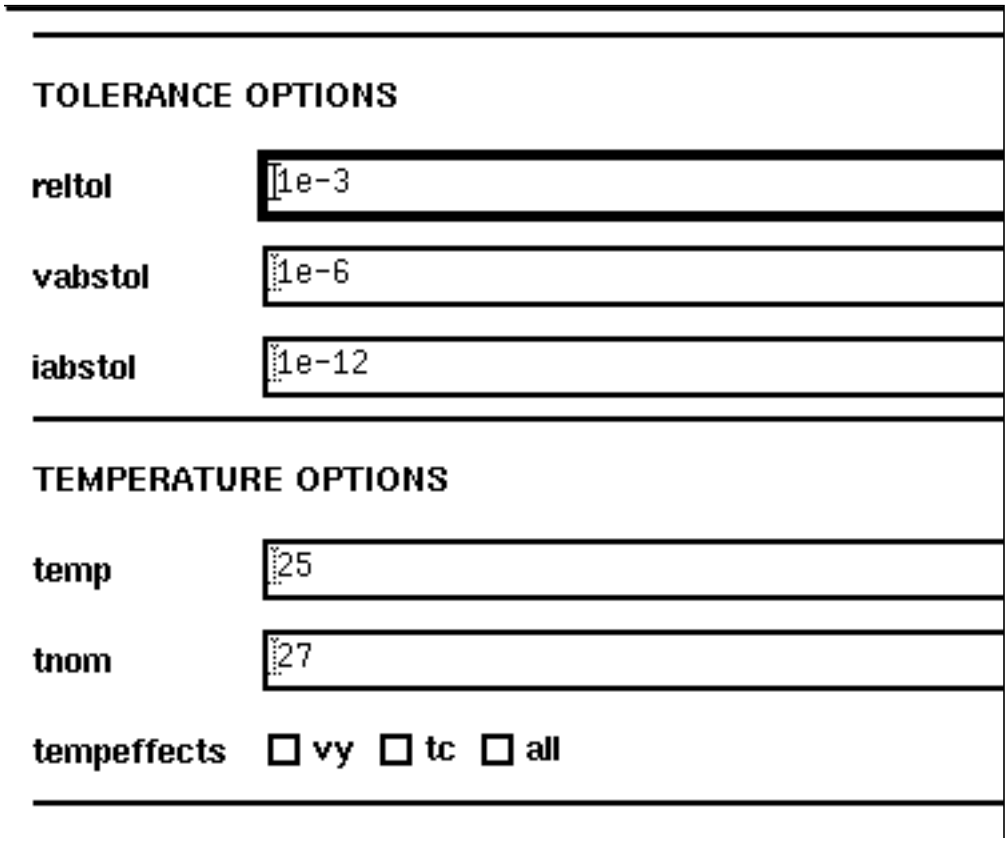
For the Cadence® SPICE simulator, the *Simulator – Options – Analog* command displays a scrolling list of simulator options.



For help on any of these options, refer to the *Cadence SPICE Reference Manual*.

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.



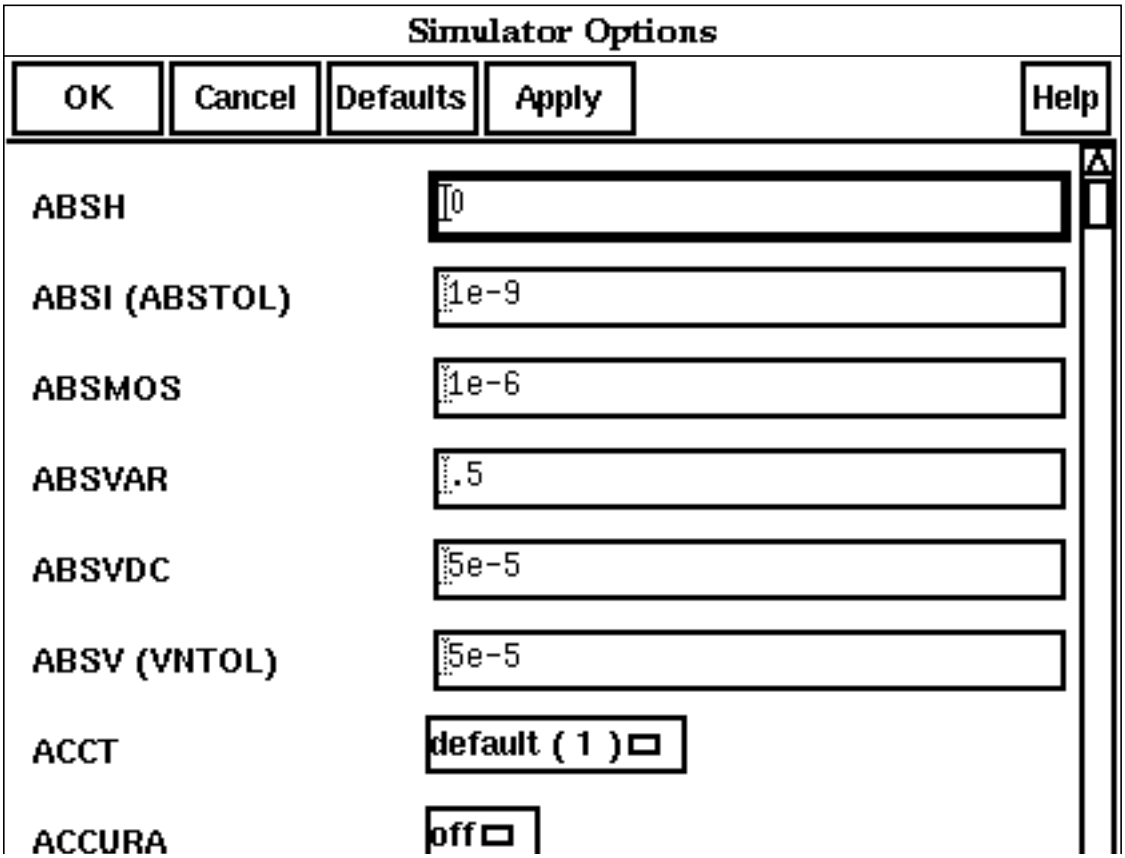
For help on spectreS options, refer to the *Spectre Circuit Simulator Reference* manual.

HSPICE Options

To set the simulator options for the HSPICE simulator,

1. Choose *Simulation – Options – Analog*.

The HSPICE Simulator Options form appears.



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.

Starting a Simulation

To start a simulation,

➤ Choose *Simulation – Run*.

Alternatively, click the green 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 – Run*.

Alternatively, click the yellow 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/Re-create*) 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).

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

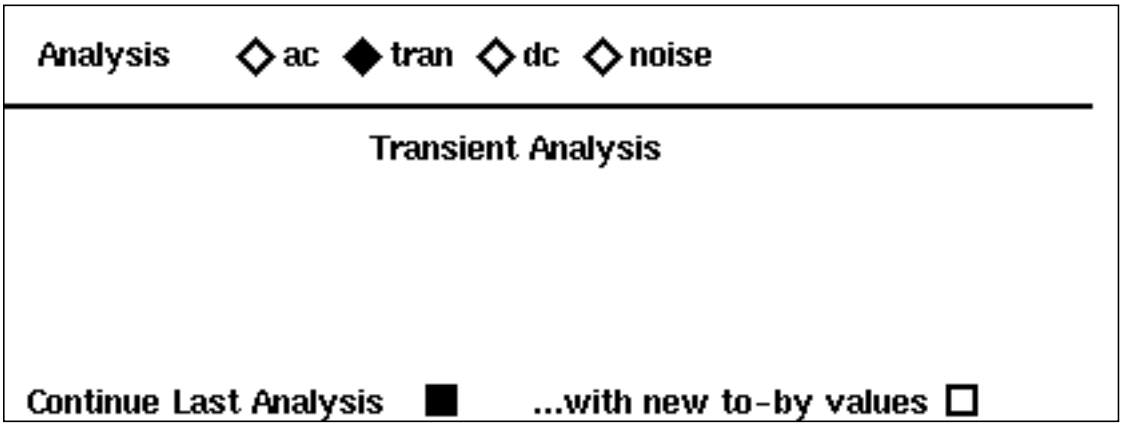
To finish an interrupted Cadence SPICE transient simulation or to continue beyond a completed transient simulation,

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.



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*.

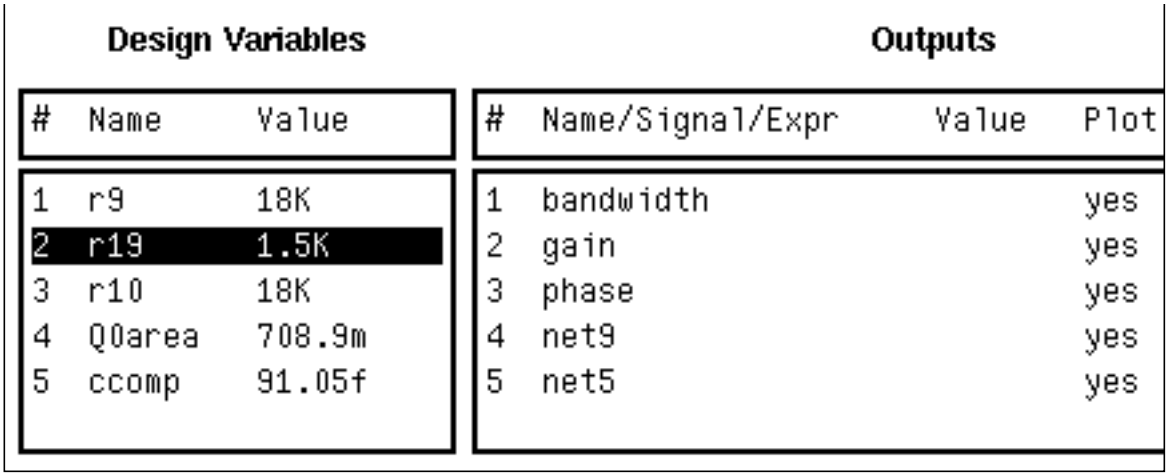
The fields for these options do not appear until you click *with new to-by values*.

The simulation continues.

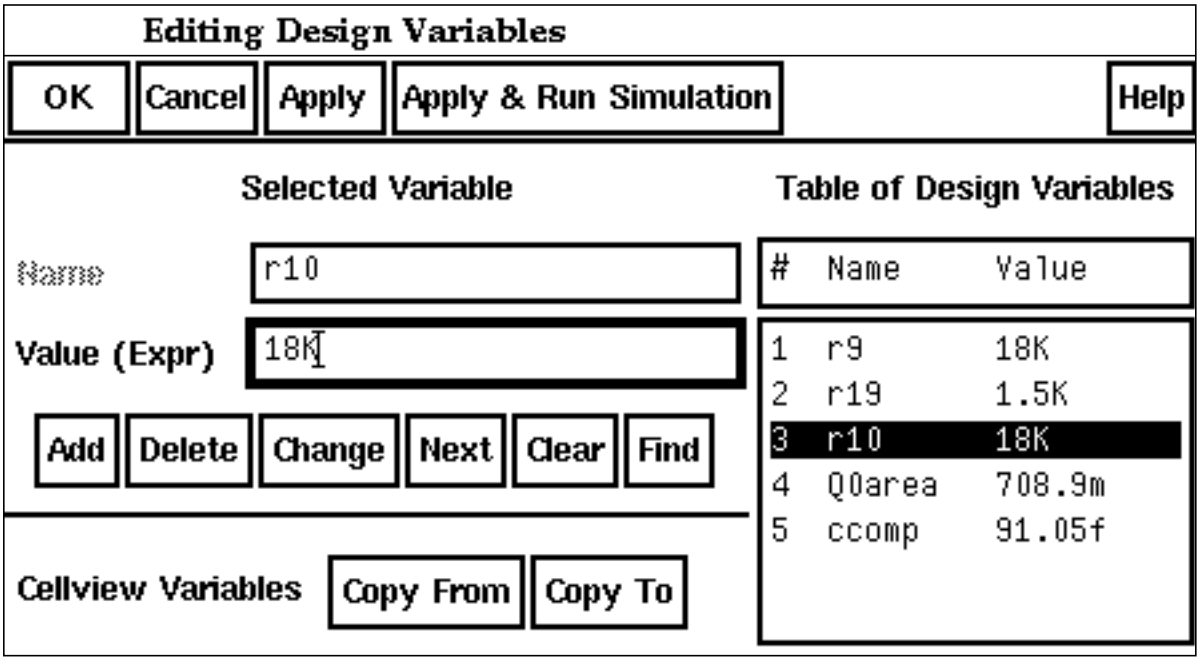
Updating Variables and Resimulating

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*.



The Editing Design Variables form appears, and the variable you clicked is highlighted.



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. Choose a restore mode.
5. 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 Log Files

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*.

Entering Simulator Commands in the Type-In Window

Note: This is available only for socket interfaces.

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.

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 Cadence® 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.

   Precede the command with a dollar sign, as shown in the following example:

   ```
   $.
   ```

Running a Parametric Analysis

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.

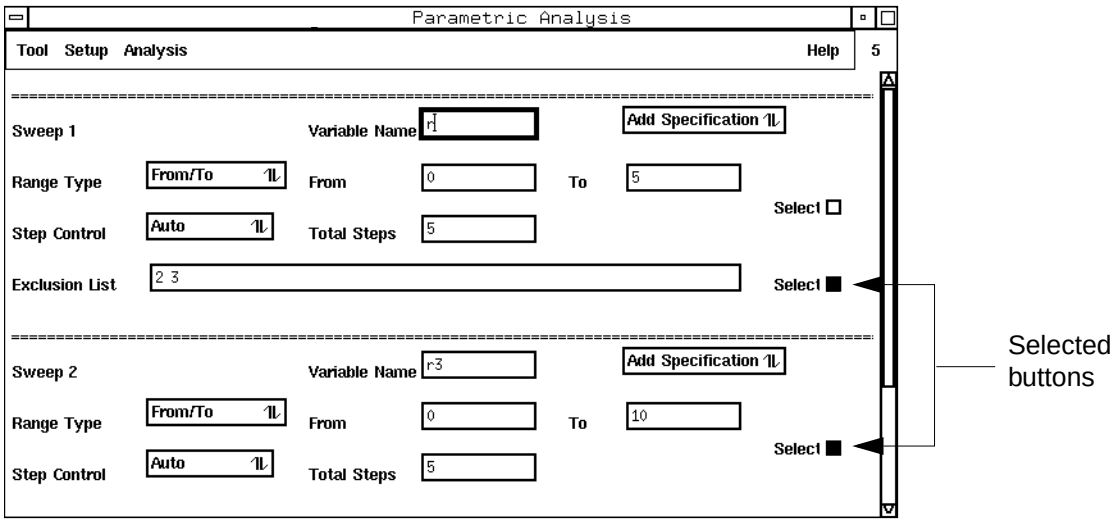
- How to start a parametric analysis run
- How to interrupt and restart a parametric analysis
- How to close the Parametric Analysis window when you finish running parametric analyses

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.



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.

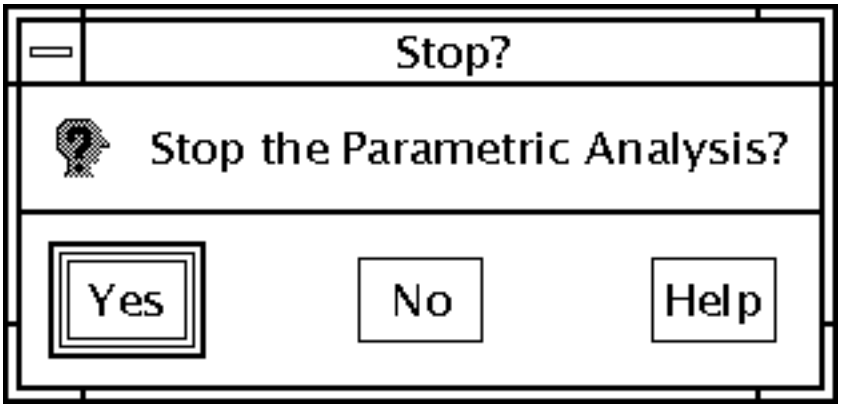
To start a Parametric Analysis run that performs only those simulations selected at run time,

➤ 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.



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.

Setting Up and Running Statistical Analyses

For information about running statistical analyses, see the *Cadence®Advanced Analysis Tools User Guide*.

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.

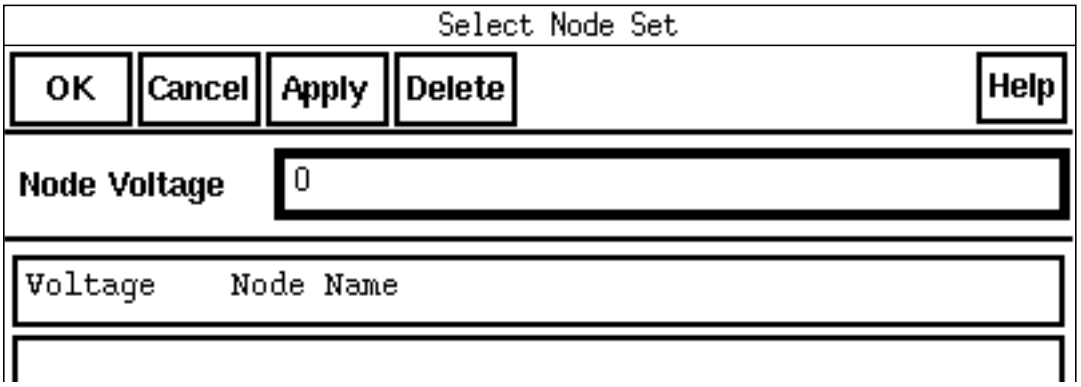
Note: The Spectre simulator and some other simulators do not support this command.

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*).

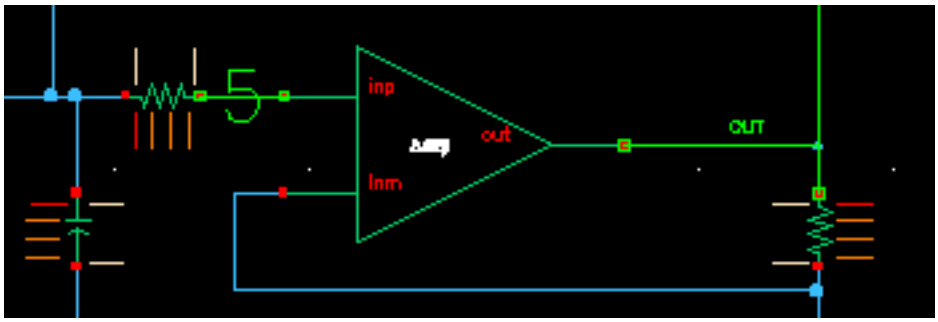
   A form appears to enter the voltage. Each command displays a different form. The Select Node Set form is shown here.



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.

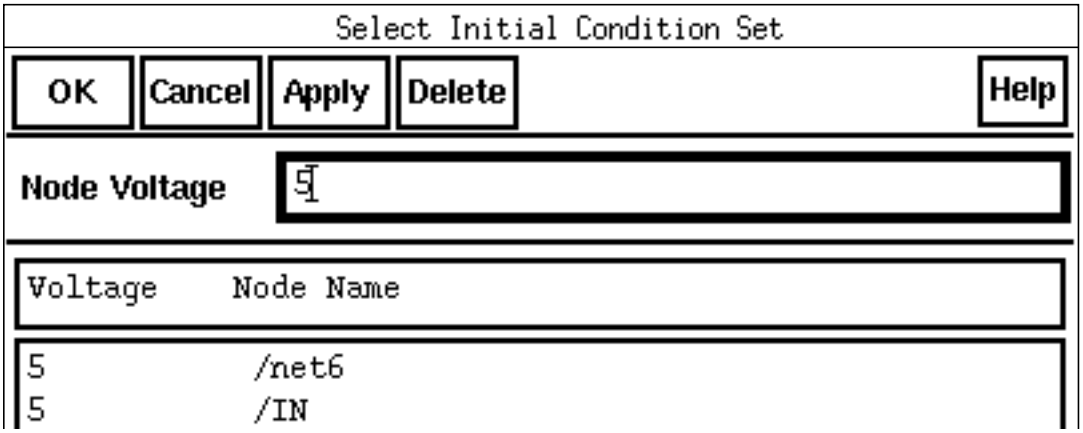
You can select nodes at any level of the hierarchy. When you select an interface node at a lower level, the node is highlighted, but the voltage value appears only on the higher-level schematic.

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.



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.

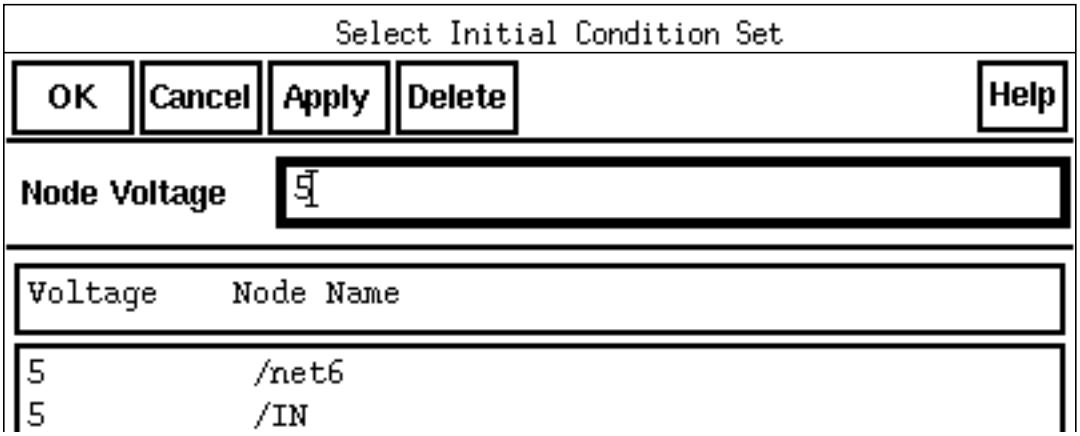
3. Click *OK* or *Cancel* when you are finished releasing nodes.

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.



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*, and choose a restore *Mode*.

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,

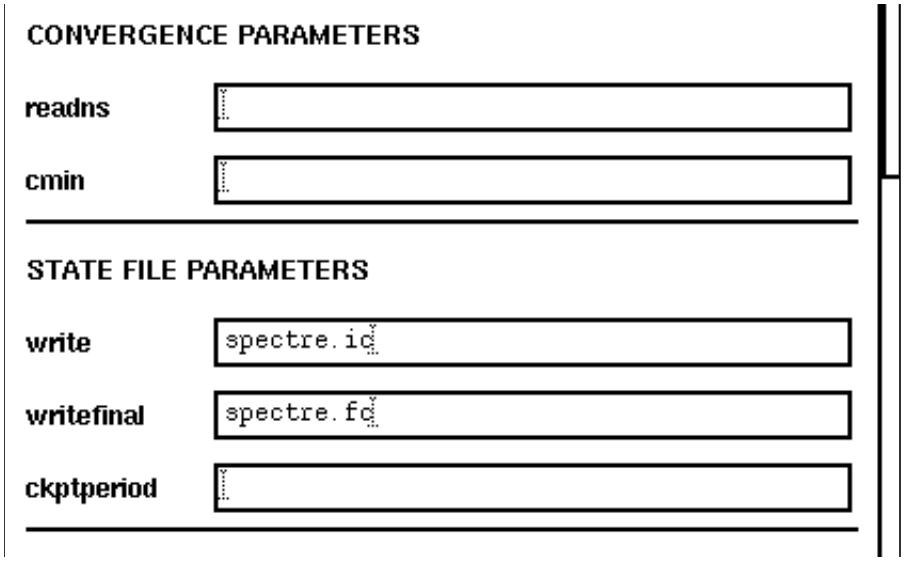
➤ Choose *Simulation – Convergence Aids* and a convergence command (*Node Set*, *Initial Condition*, or *Force Node*).

To remove the highlighting and labels,

➤ Close the active Select form.

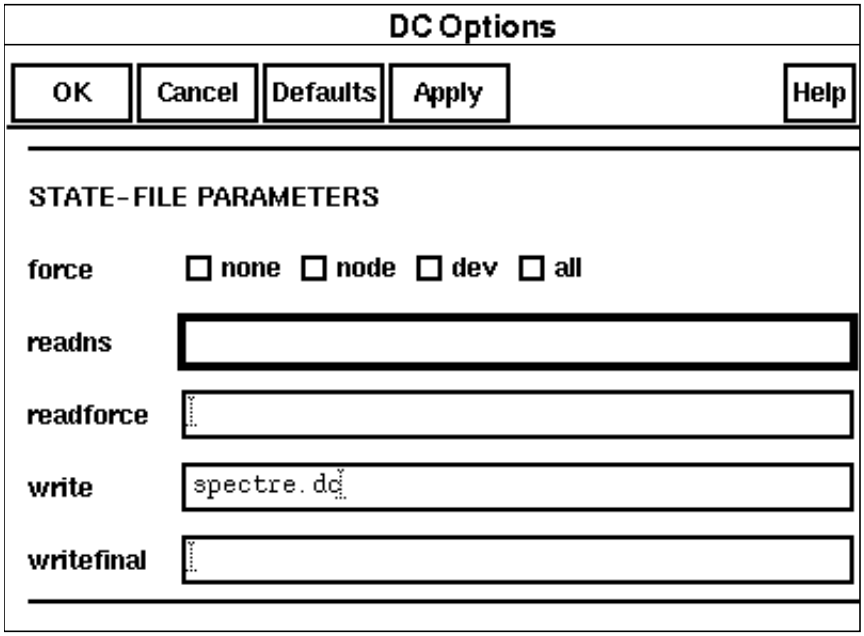
Storing a Solution

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.



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.

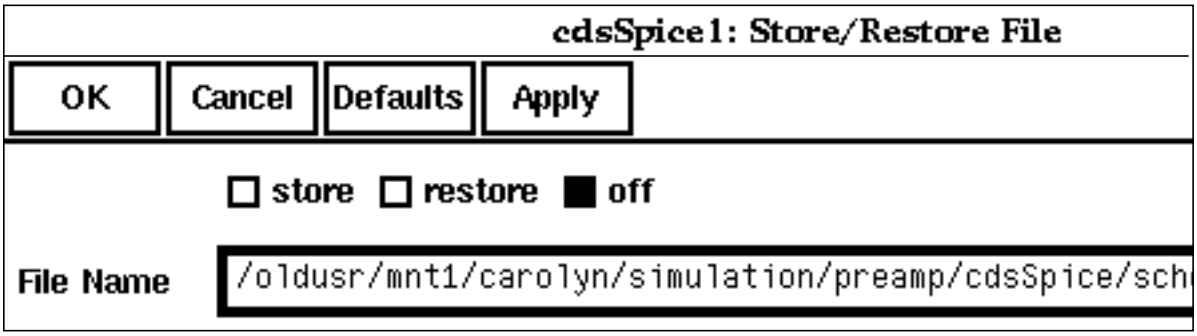


Note: This does not apply to spectre or spectreS. Use the analysis options instead.

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.



   For detailed information about the form, see “Store/Restore File” on page 270.

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

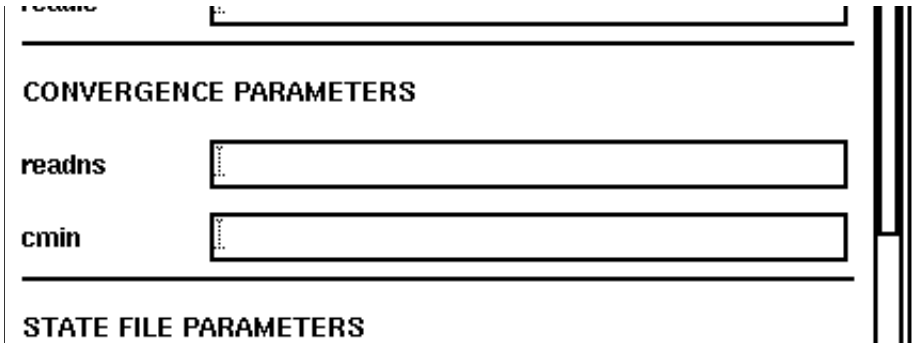
   project_dir/design/simulator/`storeRestoreFile`

   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.

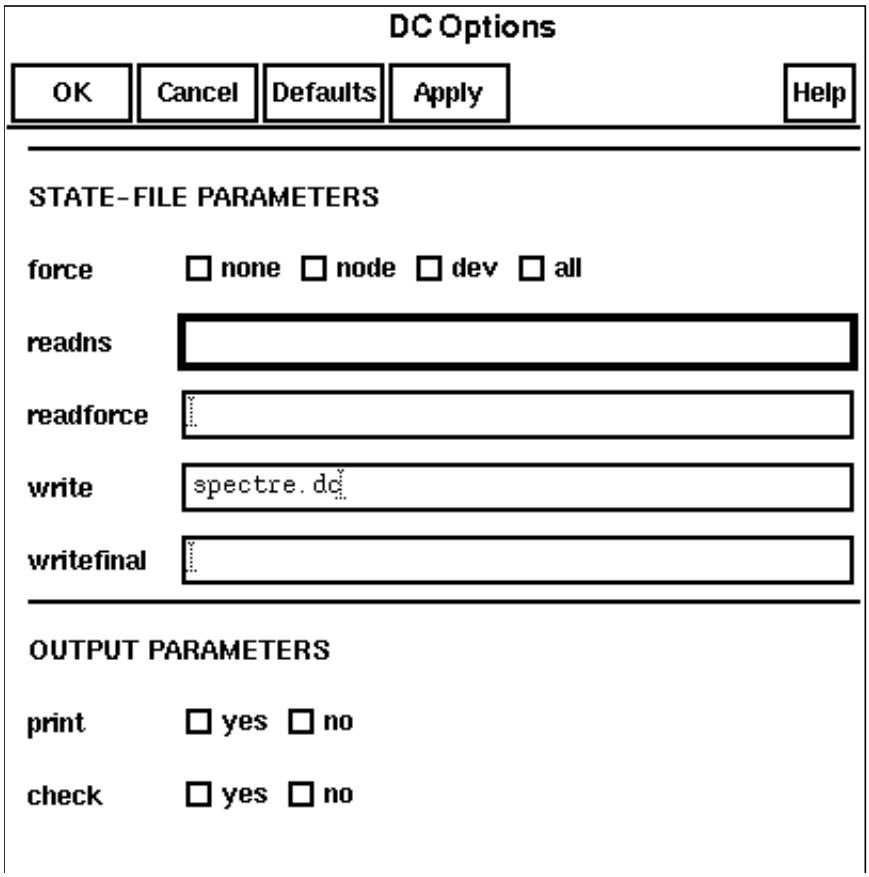
Restoring a Solution for Spectre

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.



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.



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 Cadence® analog design environment.

About Parametric Analysis

Parametric analysis can be a useful tool during the design phase of a circuit or during verification. Parametric analysis lets you specify ranges 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, the waveform calculator, and the Results Browser, lets you 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

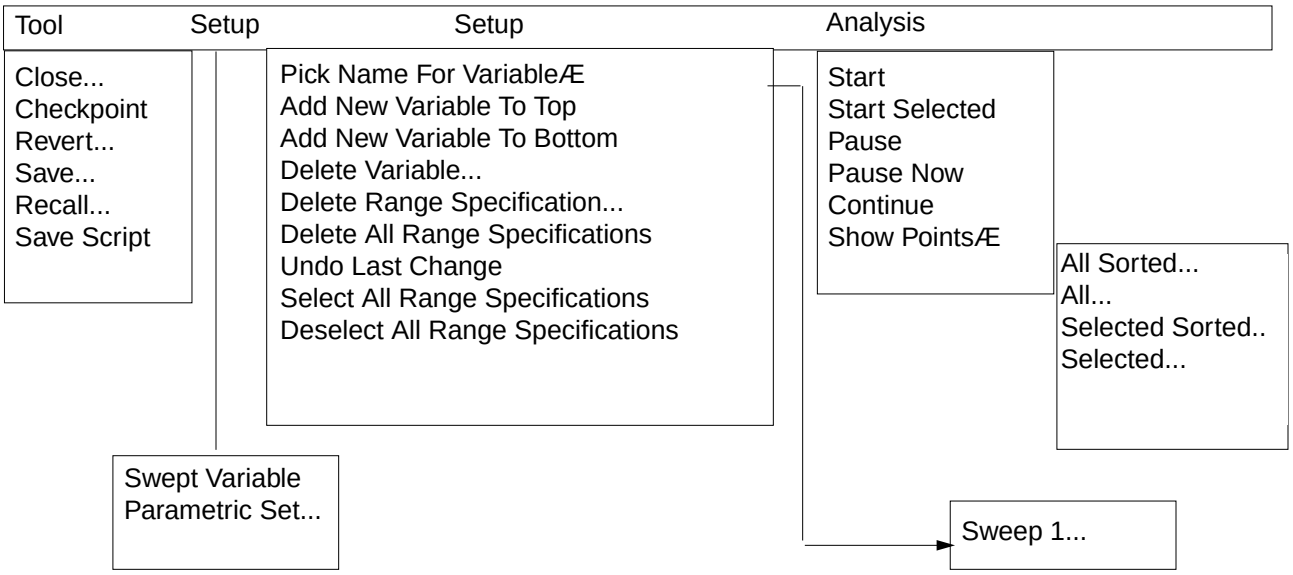
Overview of Analysis Specification

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, you plot the results of your analysis

The Parametric Analysis window has the following menu options.

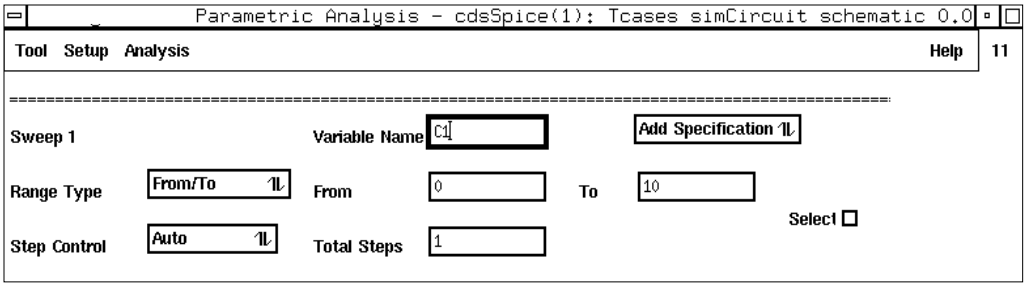


Getting Started with Parametric Analysis

You can use the following procedure to call up the Parametric Analysis window:

➤ Choose *Tools – Parametric Analysis* in the Cadence® 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*.



Caution

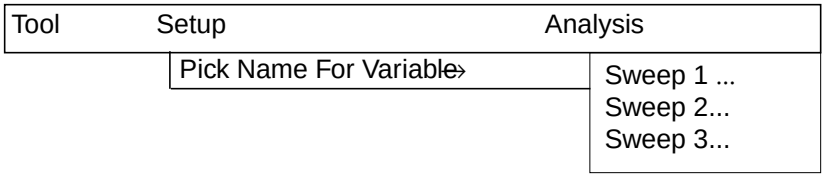
Parametric analyses are interactive. That is, if you make a change to your design during the simulation the design is re-netlisted. Data in the successive runs (after the design change) may be different from results of previous runs.

Specifying Sweep Variables

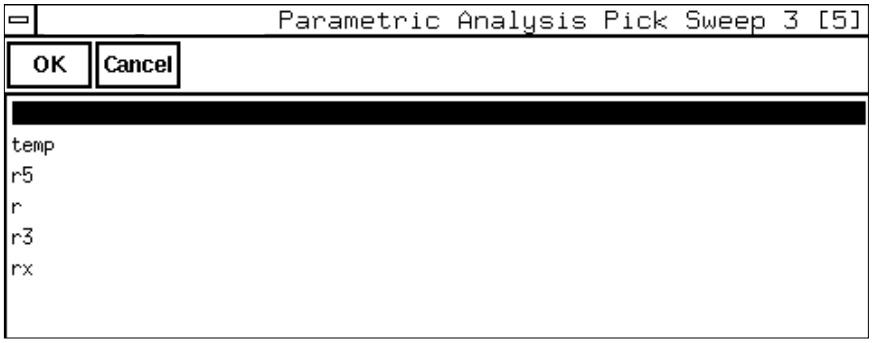
To specify a variable,

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.



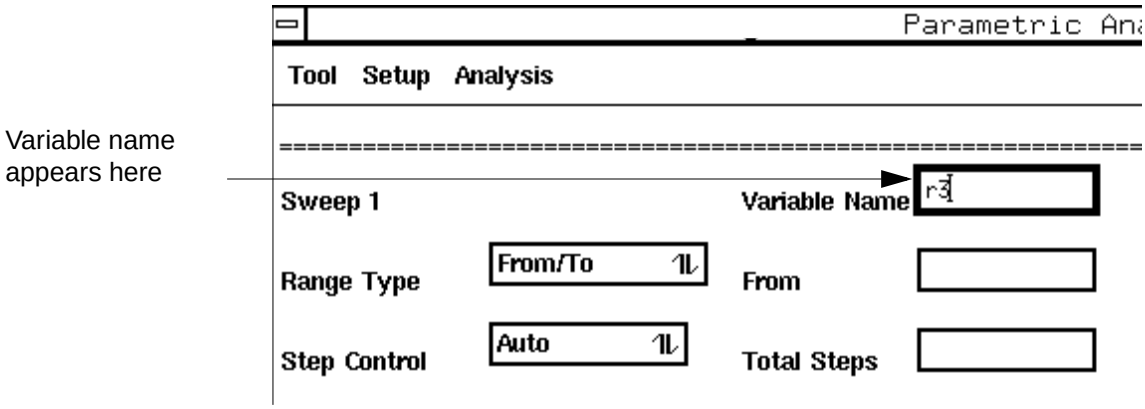
A Pick Sweep window displays a list of variables you can sweep.



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.



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 *Cadence Analog Circuit simulation* 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.

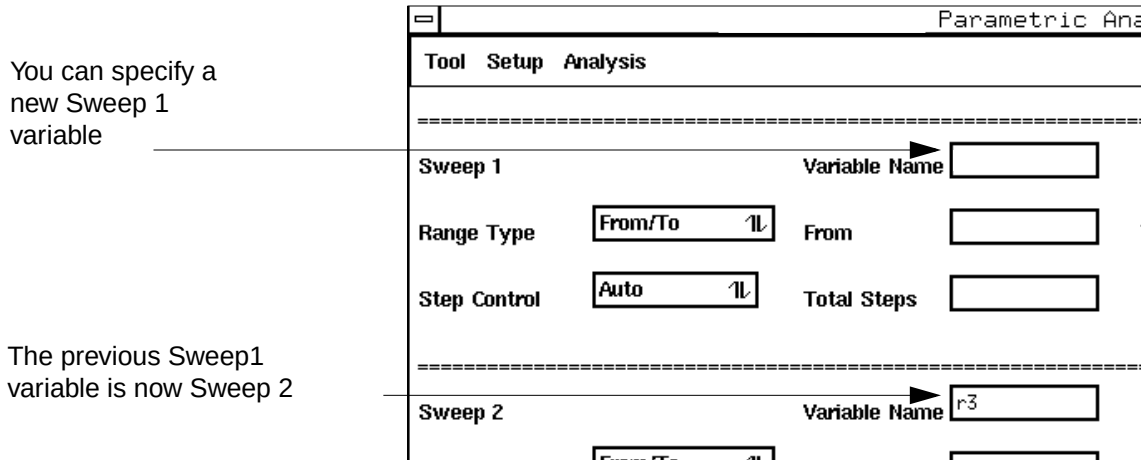
Adding Variables

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.

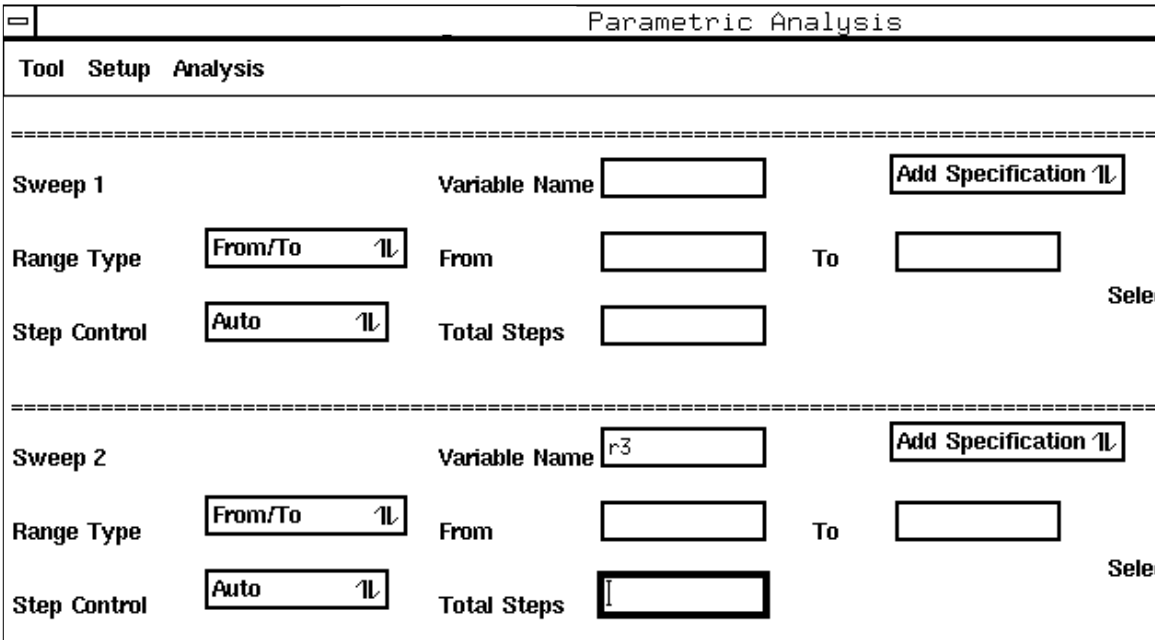
The window changes to let you specify a new Sweep 1 variable. (In this example, `r3` was the previous Sweep 1 variable.)



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.)



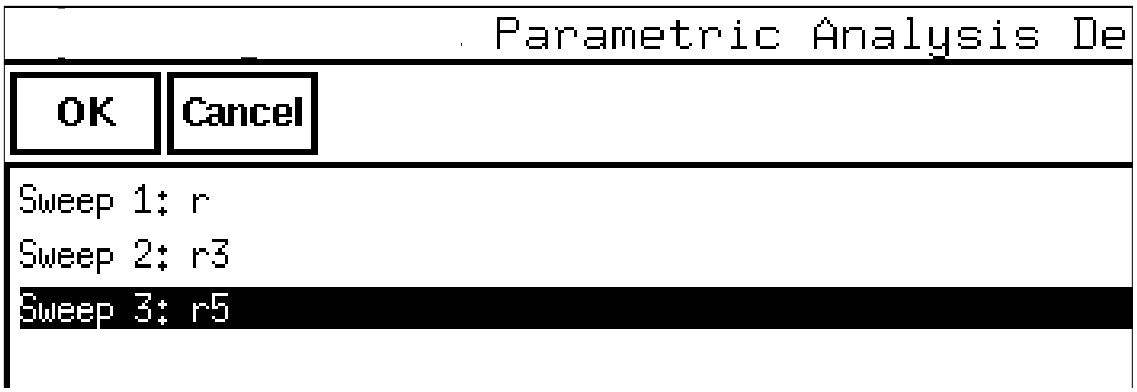
Note: When you add a variable to a window, information is added to the window but the window itself does not grow larger. You might need to enlarge the window to see all the information.

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.



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

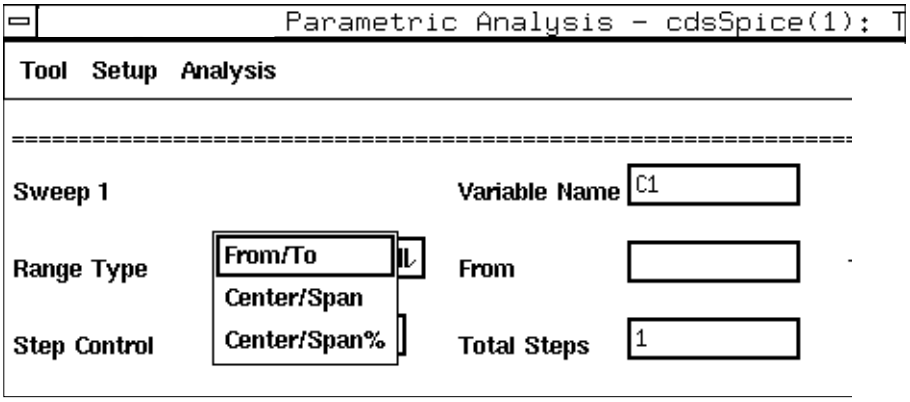
- How to specify the range for a sweep and how to choose a range of specification options
- How to specify multiple sweep ranges for a given variable
- How to delete range specifications

If you want to learn how to select some range specifications and ignore others for a given parametric analysis run, see "Run-Time Modifications" on page 255.

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.



   For detailed information about the form, see "Parametric Analysis" on page 295.

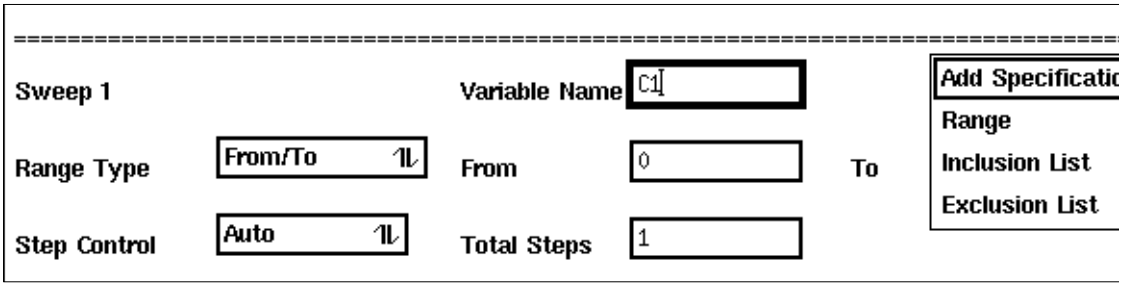
2. Type in appropriate range limits.

   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,

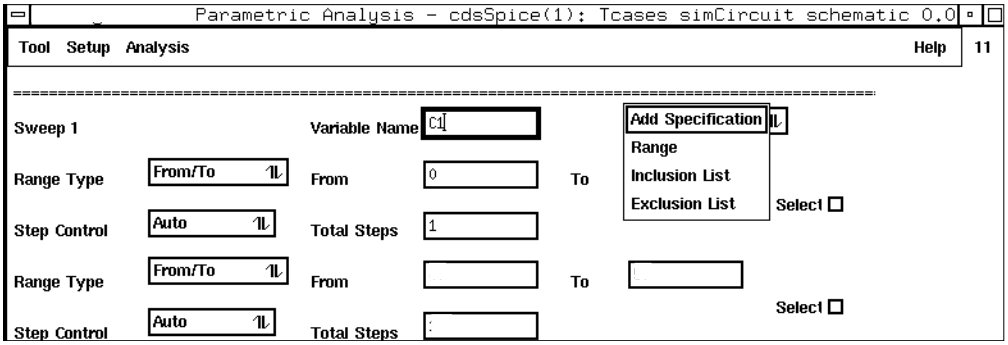
1. Choose *Range* from the *Add Specification* cyclic field in the Parametric Analysis window.



   See "Form Field Descriptions" on page 295 for more information.

   The form changes to let you specify an additional sweep range.

2. Type in appropriate limits.

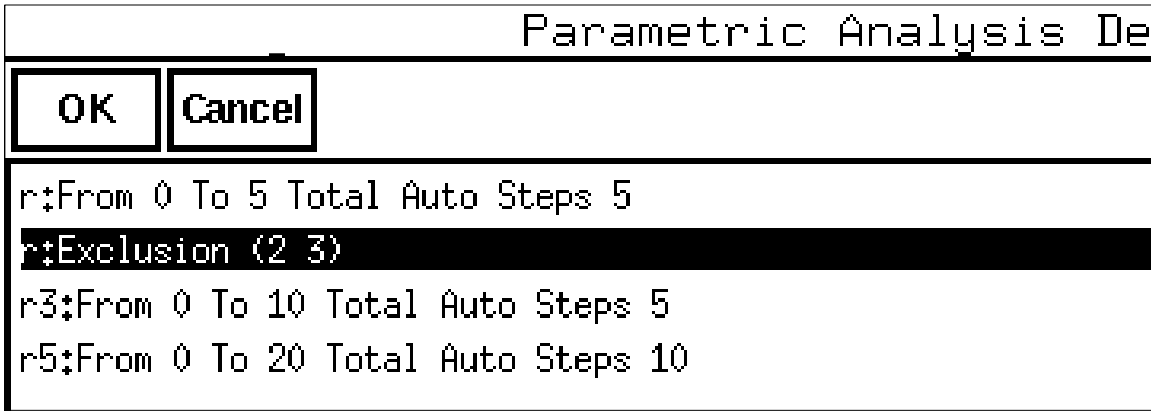


Deleting Range Specifications

To delete a single range specification,

1. Choose *Setup – Delete Range Specification* in the Parametric Analysis window.

   The Delete Settings form appears.



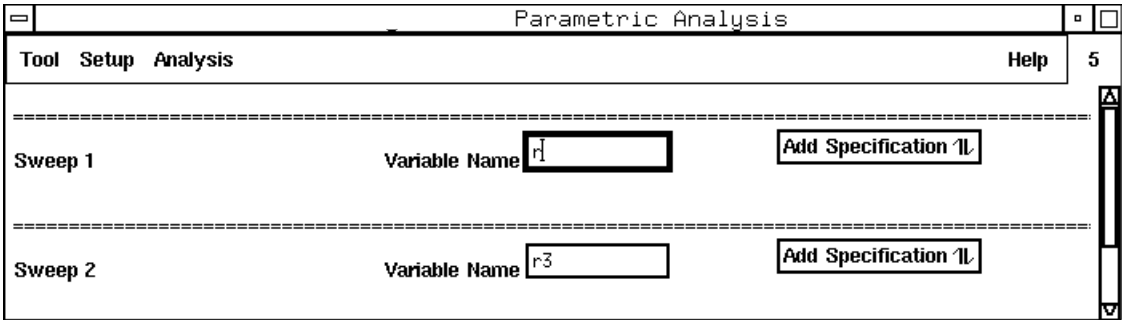
2. Click on the setting you want to delete and click *OK*.

   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:



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 *Cadence Analog Circuit simulation* 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.

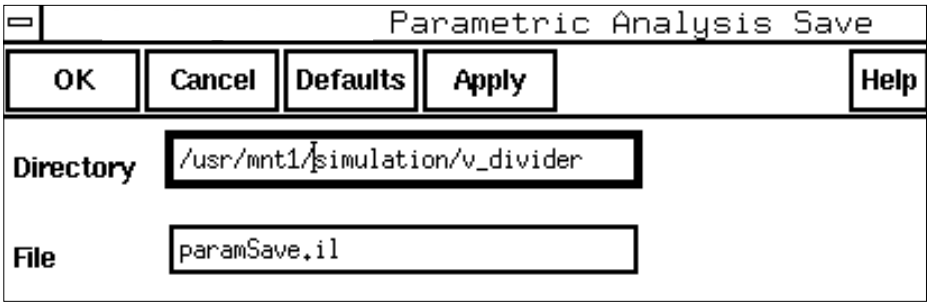
1. Choose *Tool – Revert* in the Parametric Analysis window.
2. Click *Yes* in the dialog box.

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.



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

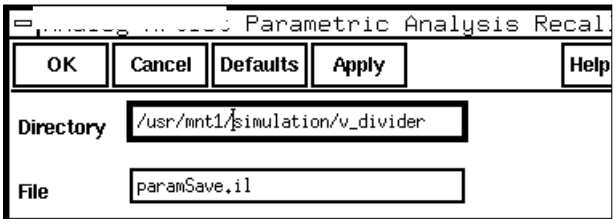
Once you invoke the *Save* form and click *OK* or *Apply*, a subsequent *Recall* invocation will display the same values used in the *Save* form.

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.



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

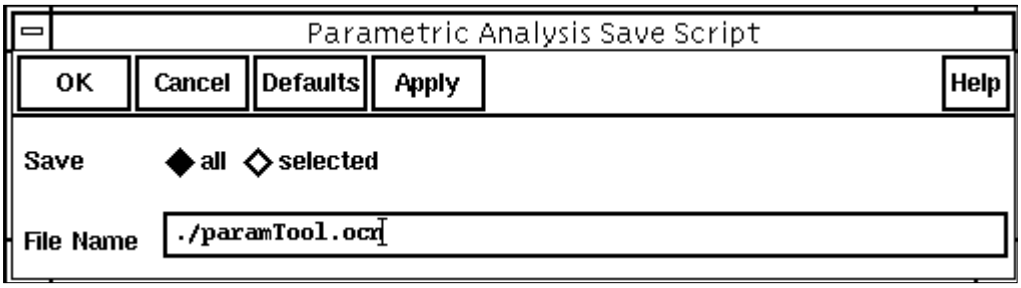
If you invoke the *Recall* form (without invoking the *Save* form earlier) you will see the default values. If you then change the values (*Directory*, *FileName* or both) and then click *OK* or *Apply*, subsequent invocation of *Recall* would display the changed values.

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.



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...*

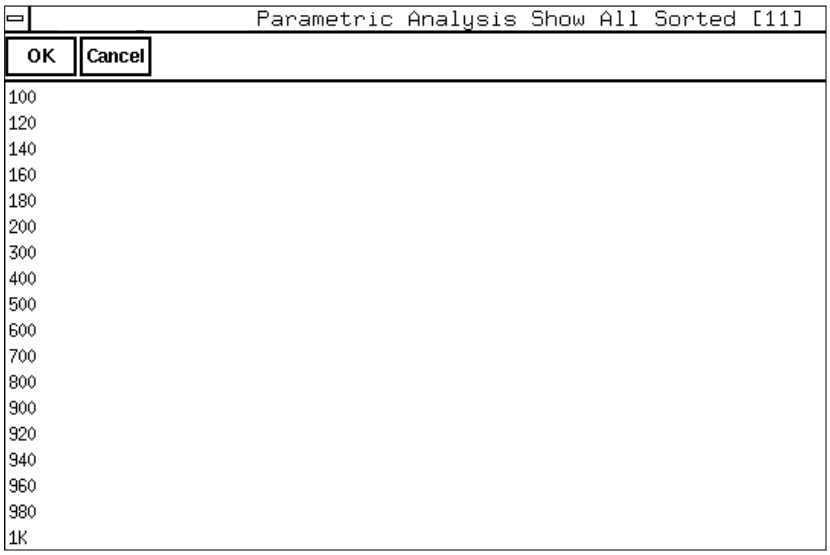
To view selected data points in the order they are specified in the Parametric Analysis window,

➤ Choose *Analysis – Show Points – Selected...*

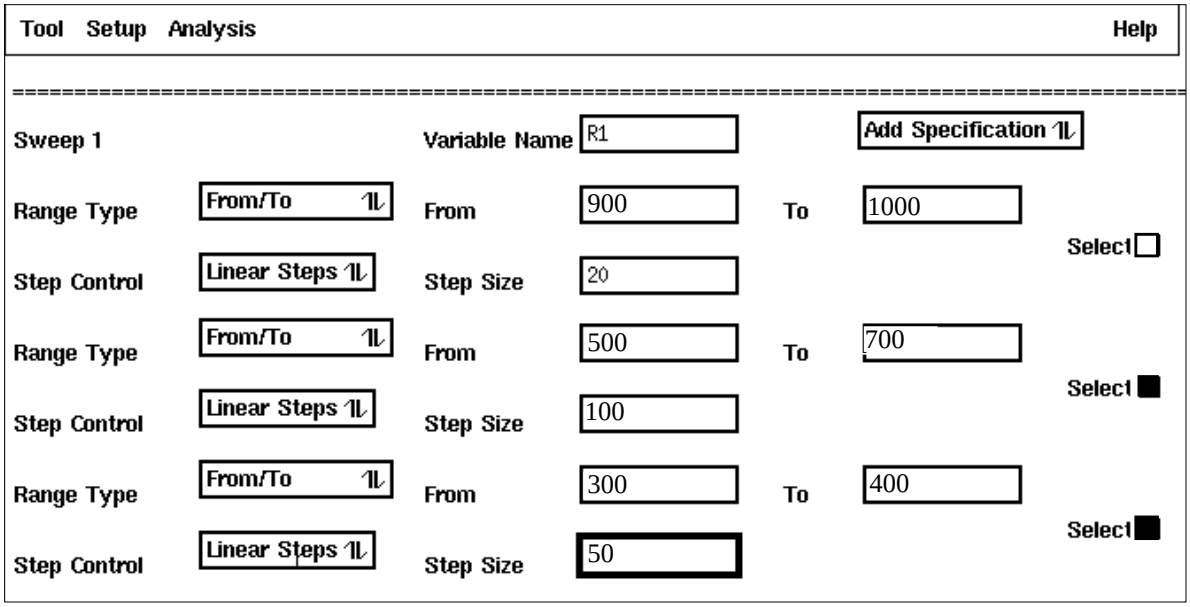
To view selected data points in the order they are simulated in a Parametric Analysis run,

➤ Choose *Analysis – Show Points – Selected Sorted...*

For each of the four options, a window appears with a list of the data points you requested.



For example, consider the following range and step definitions.



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

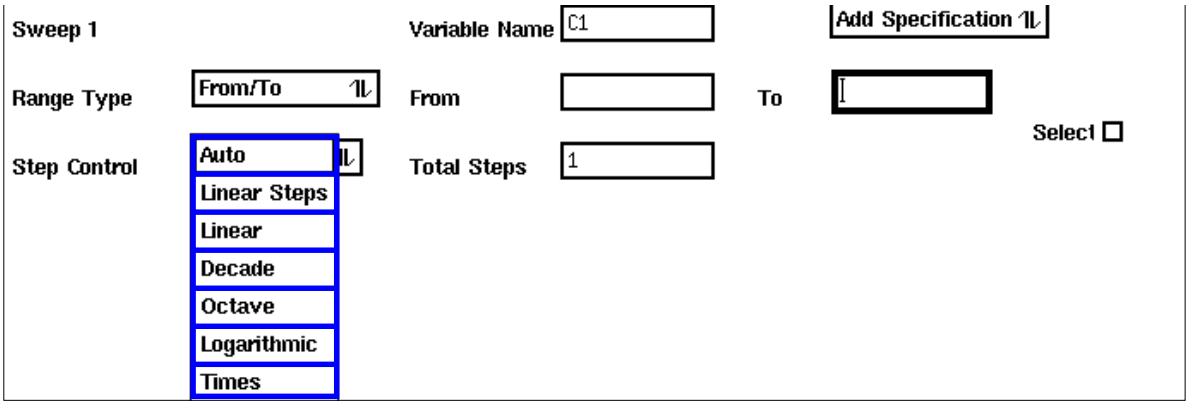
To specify appropriate step values for a sweep,

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.)

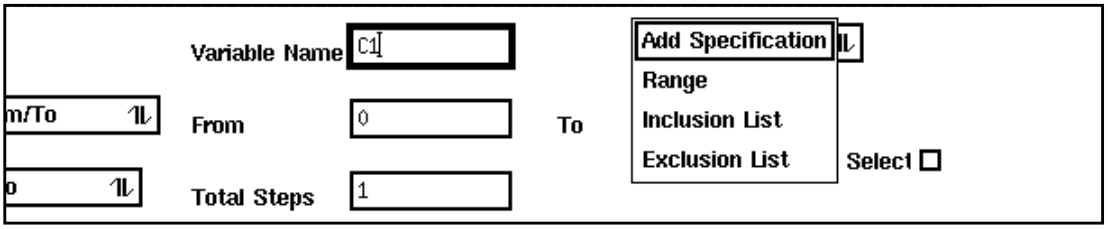
   To learn more about a particular step-value type, click on that type in the window segment below. To learn more about all the step-value types, see "Parametric Analysis" on page 295.



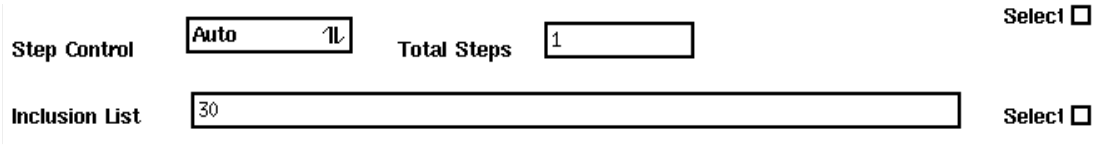
Adding Individual Step Values

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.

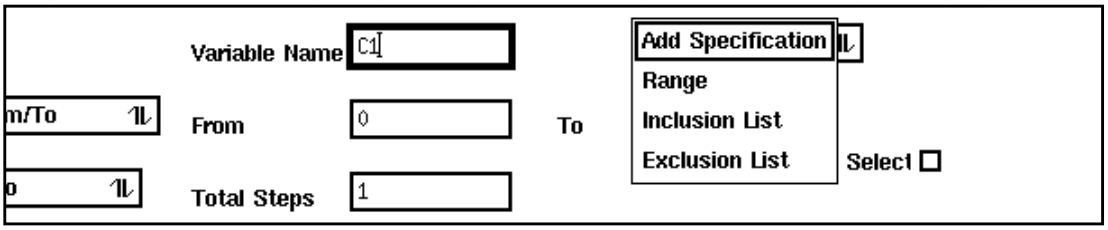


2. Type in the values that you want to add.
3. If necessary, request another inclusion list from the *Add Specification* field.

Deleting Individual Step Values

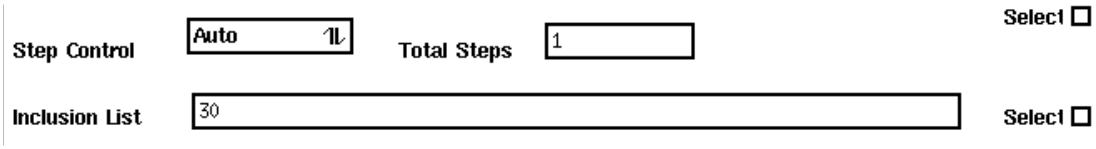
To delete individual step values,

1. Choose *Add Specification – Exclusion List* in the Parametric Analysis window.



For detailed information about the form, see “Parametric Analysis” on page 295.

The field is added to the Parametric Analysis window.



2. Type in the values that you want to delete.
3. If necessary, request another exclusion list from the *Add Specification* field.

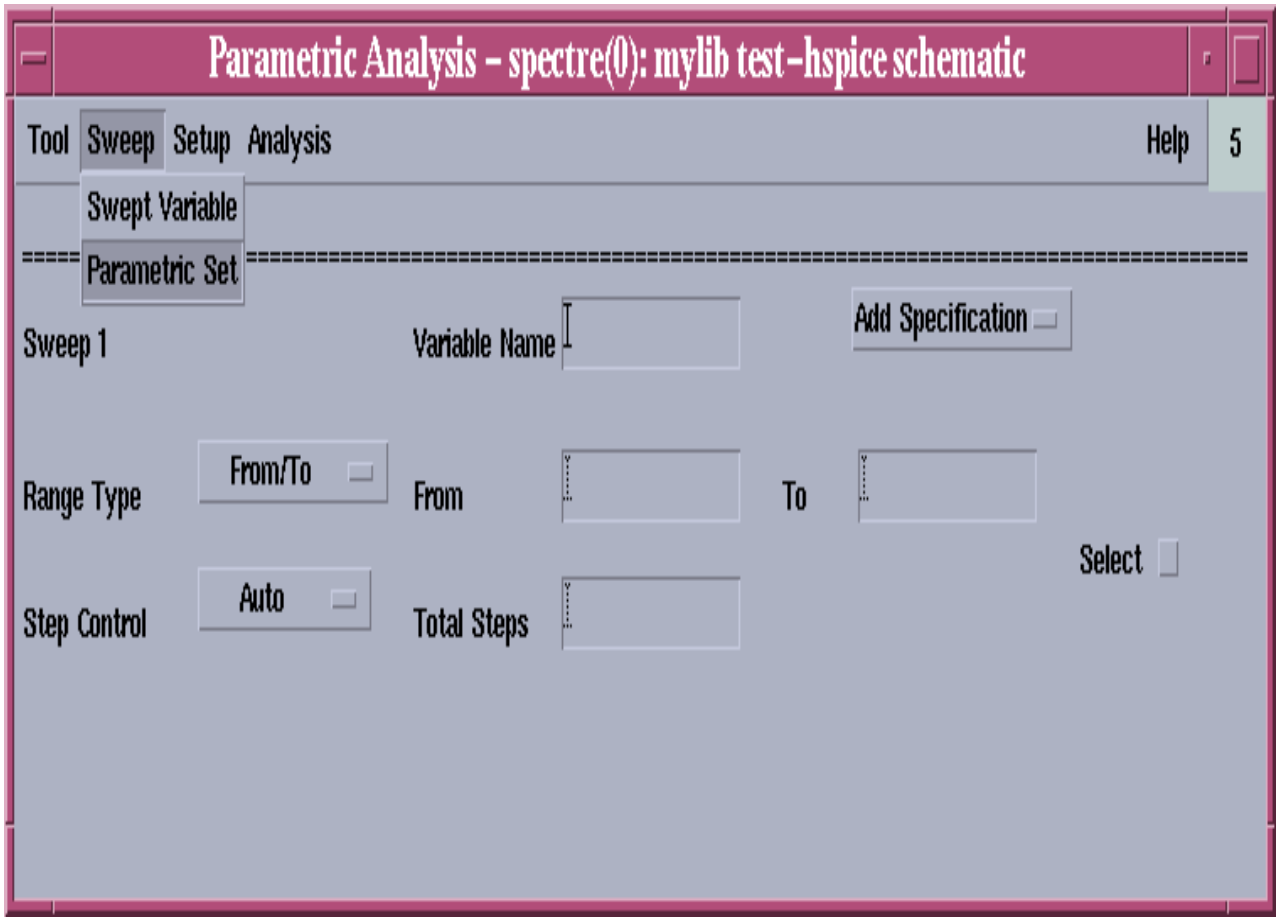
Parametric Set Sweep

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.

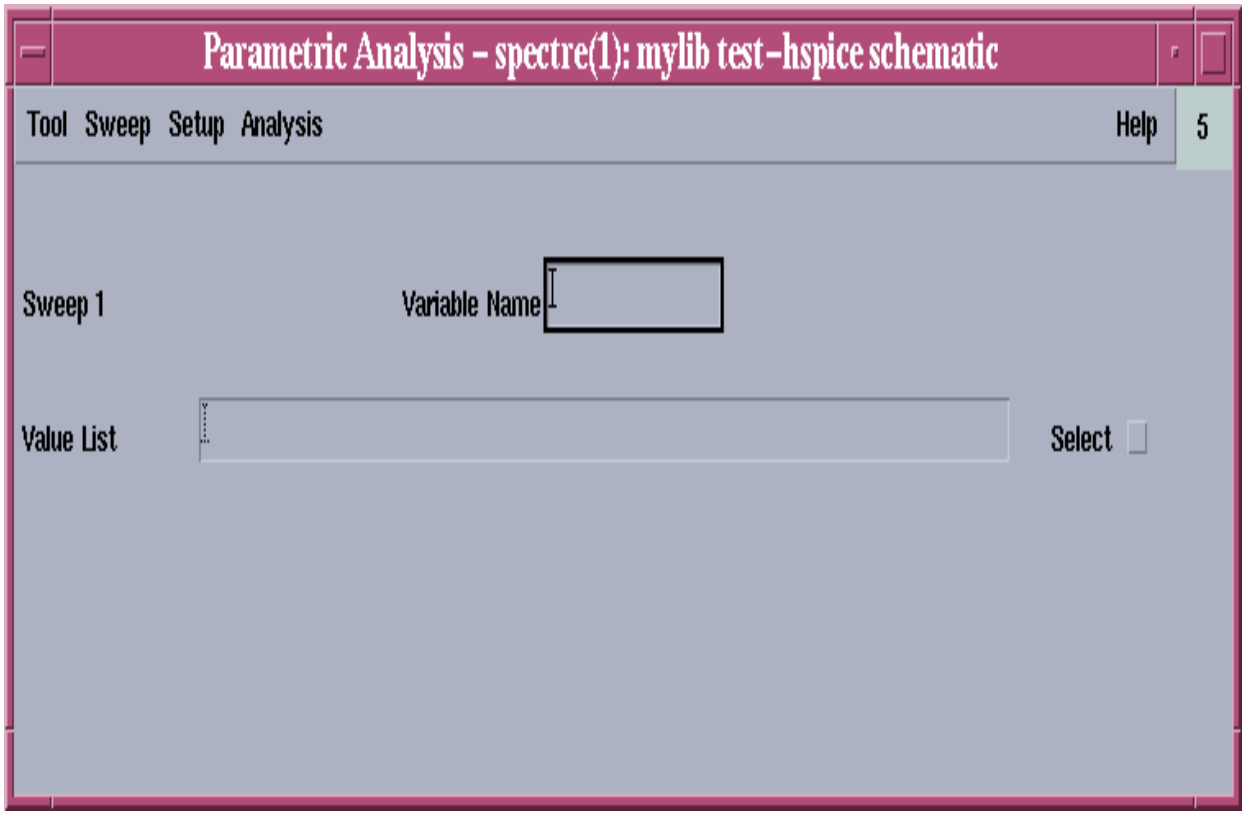
The sweep types supported by the *Parametric Analysis* tool are *Swept Variable* and *Parametric Set*, where *Swept Variable* is the default mode. You can select these using the

Sweep menu. You can also switch between the supported sweep types without losing the contents of the form.



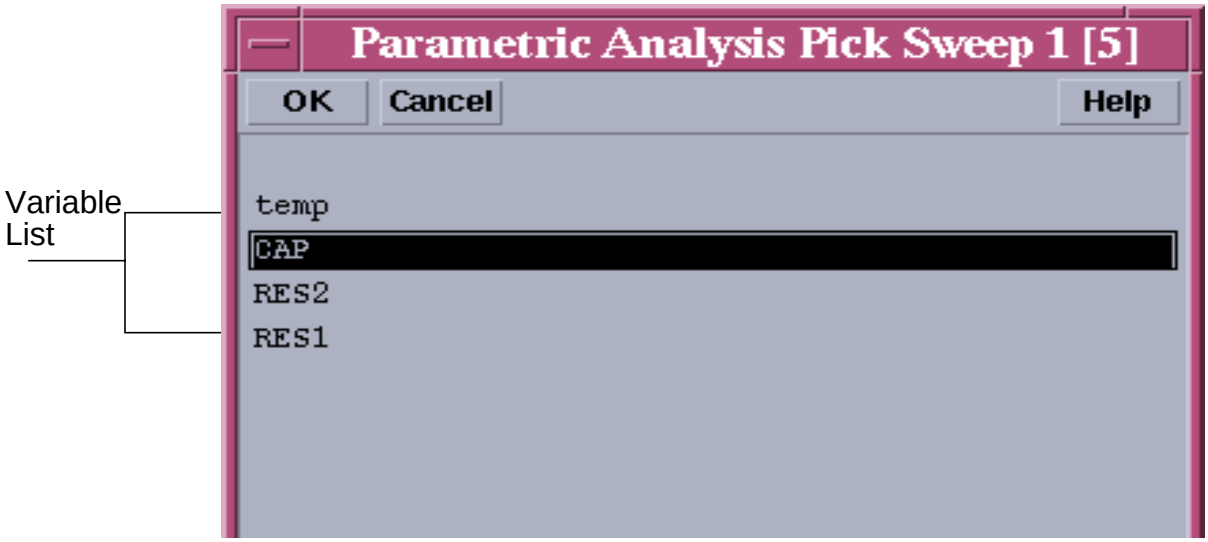
To perform a *Parametric Set* sweep,

1. Choose *Sweep->Parametric Set.*



Specify the *Variable Name* for the sweep. Also specify the list of values used for the *Parametric* simulation.

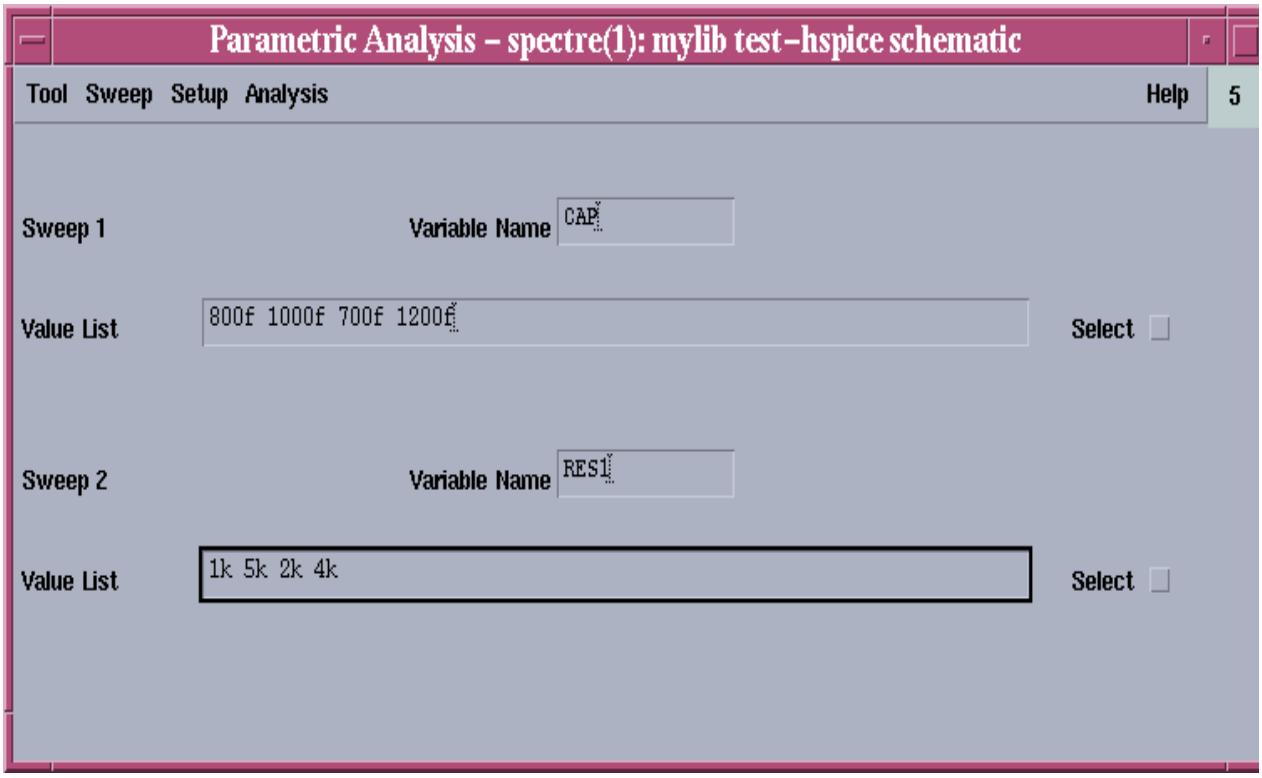
You can also select the *Variable Name* using *Setup->PickName For Variabl*e. For details see Specifying Sweep Variables. A *Pick Sweep* window displays a list of variables you can sweep.



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.

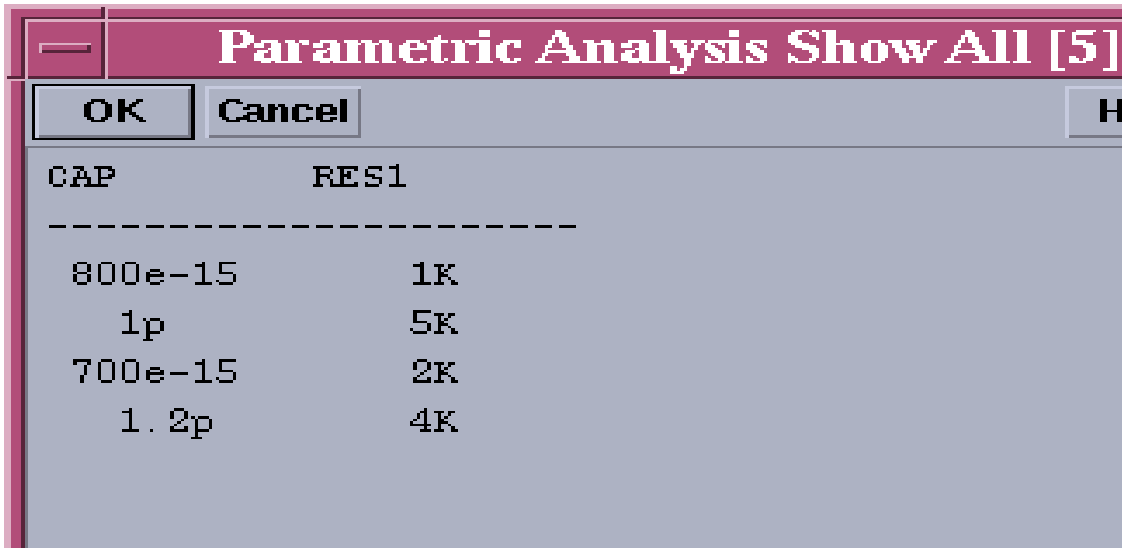
For example, for *Variable Name:* `CAP and RES` and *Value List* of `800f 1000f 700f 1200f and 1K 5K 2K 4K,` the specified *Parametric Set* Sweep is as

follows.



Note: 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.



To run the run the Parametric Simulation for the data displayed, select *Analysis->Start* or *Analysis->Start Selected*. For details, see Viewing Specifications.

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 *Cadence® 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 *Cadence® 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 *Cadence® Advanced Analysis Tools 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:

$\mathit{decade\ multiplier} = 10^{1/\text{steps per decade}}$

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:

$\mathit{octave\ multiplier} = 2^{1/\text{steps per octave}}$

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:

$log\ multiplier = (To/From)^{(steps-1)}$

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.

Overview of Plotting

There are several ways to select simulation results and plot them in the Waveform window:

- From the waveform calculator, use a *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*

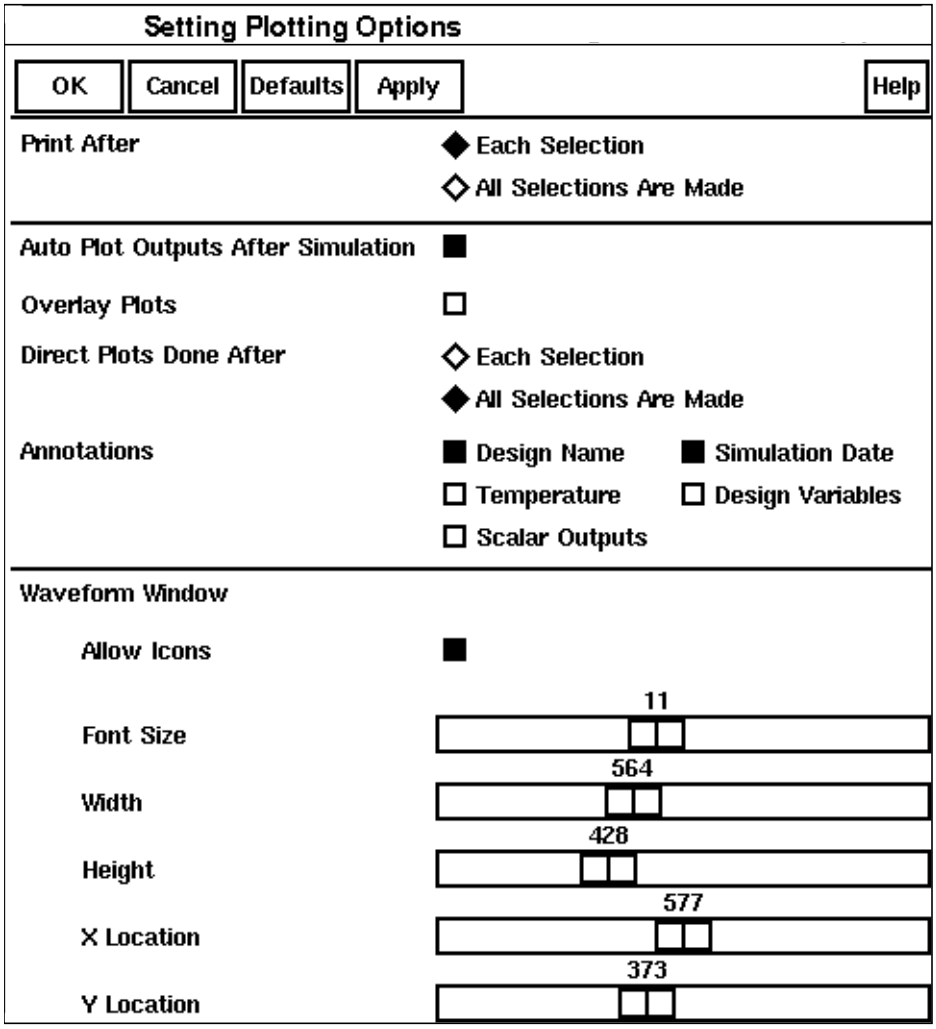
Note: The ability to plot during a simulation run is also termed as *Snapshot*.

If you set up the *Outputs* section in the Cadence 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.

Setting Plotting and Display Options

You set plotting and Waveform window options with the Setting Plotting Options form.



For detailed information about the form, see “Setting Plotting Options” on page 366.

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 more details, refer to the *Analog Waveform User Guide*.

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 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.
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.

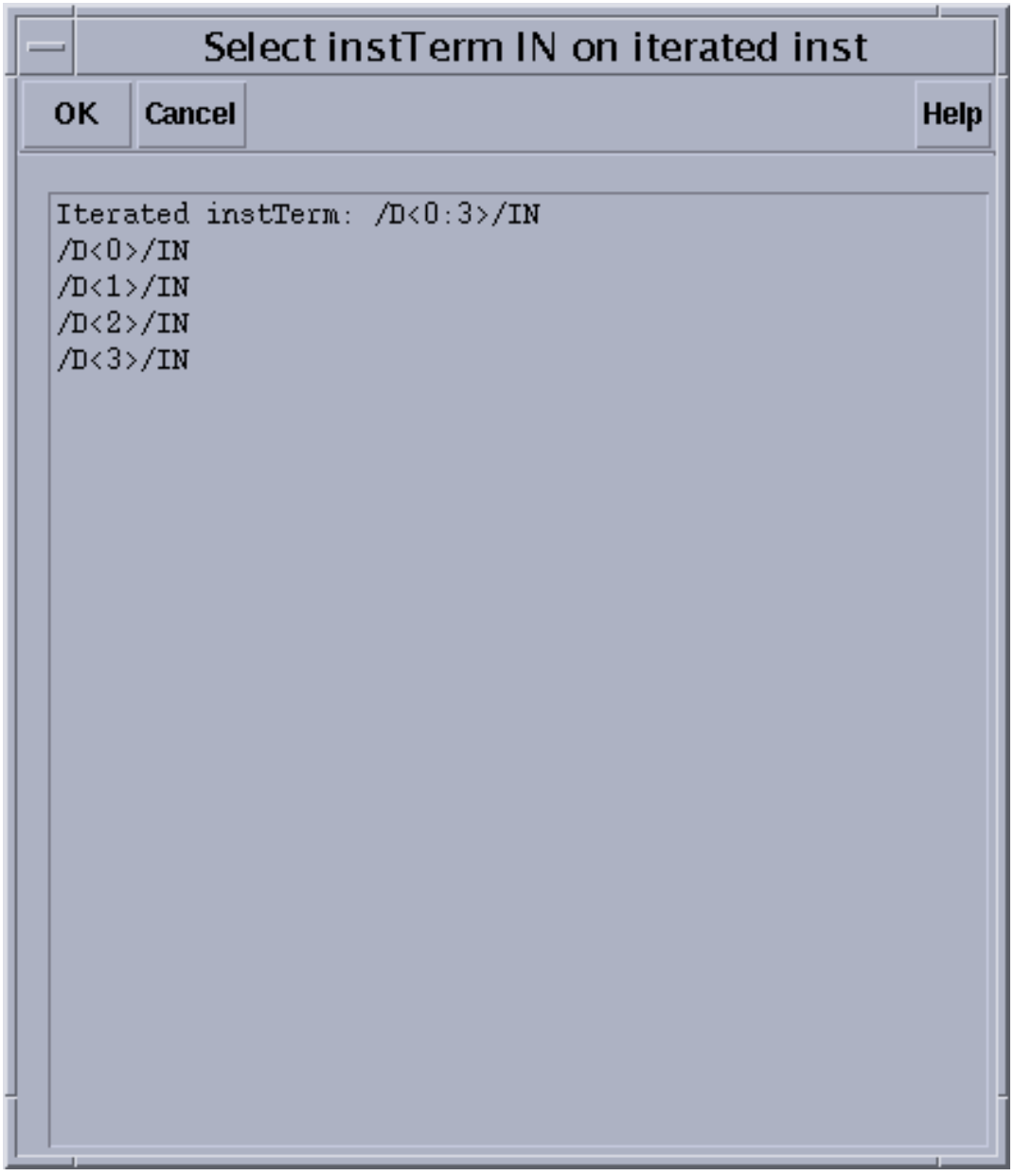
   The system plots the results you selected in the current Waveform window or opens a new Waveform window if one is not open.

To plot all of the available results at once,

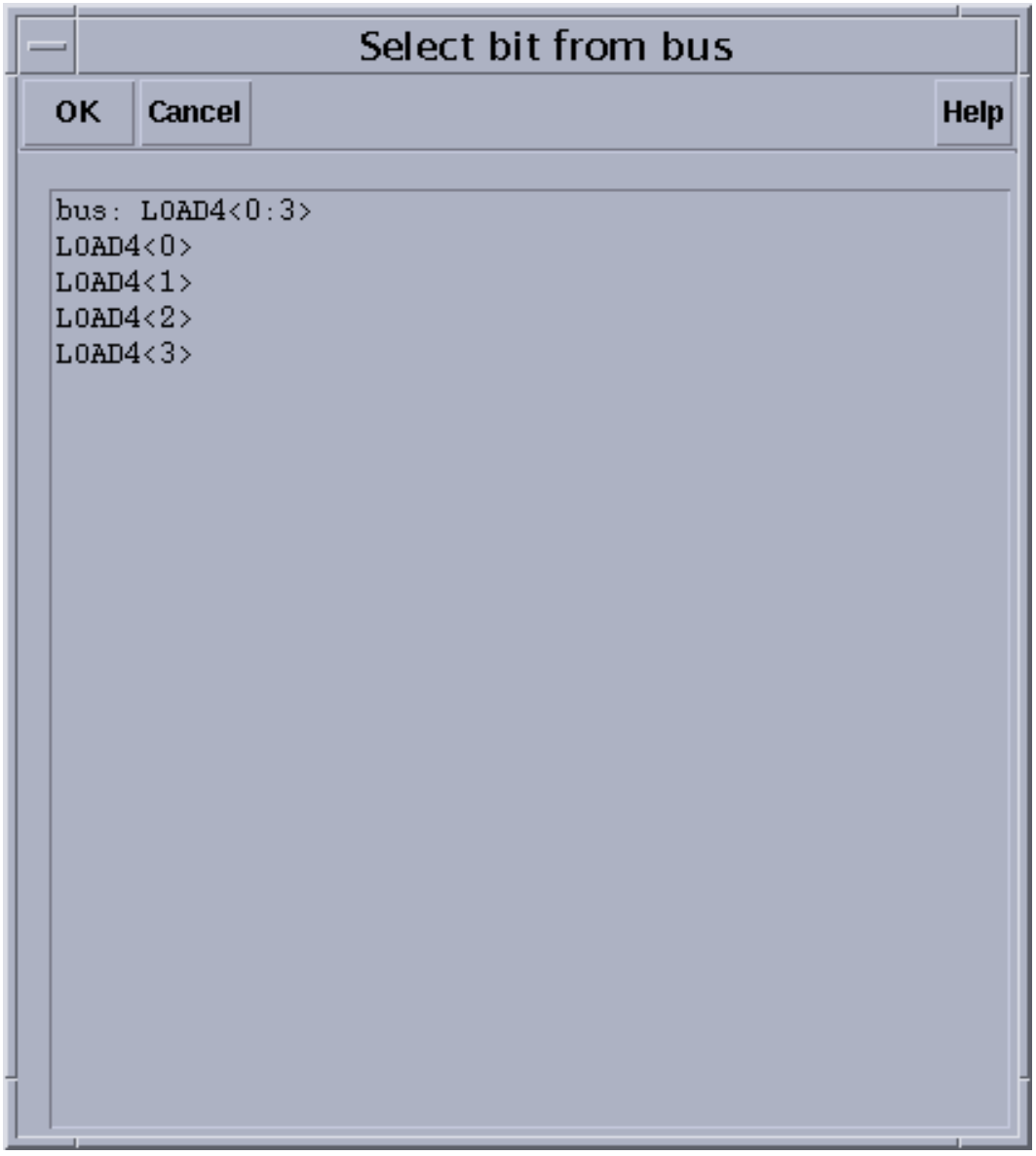
➤ Click the *Plot Outputs* icon in the Simulation window.

Note: You can plot only saved voltages and currents.

When you choose *Outputs -> To be Plotted -> Select* on schematic, and then select aniterated instance in the schematic, the *Select instTerm IN on iterated inst* form is displayed.



If you select a bus signal, the *Select bit from bu*s form is displayed.



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 not already displayed. The Waveform window often displays behind the Schematic window.

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.

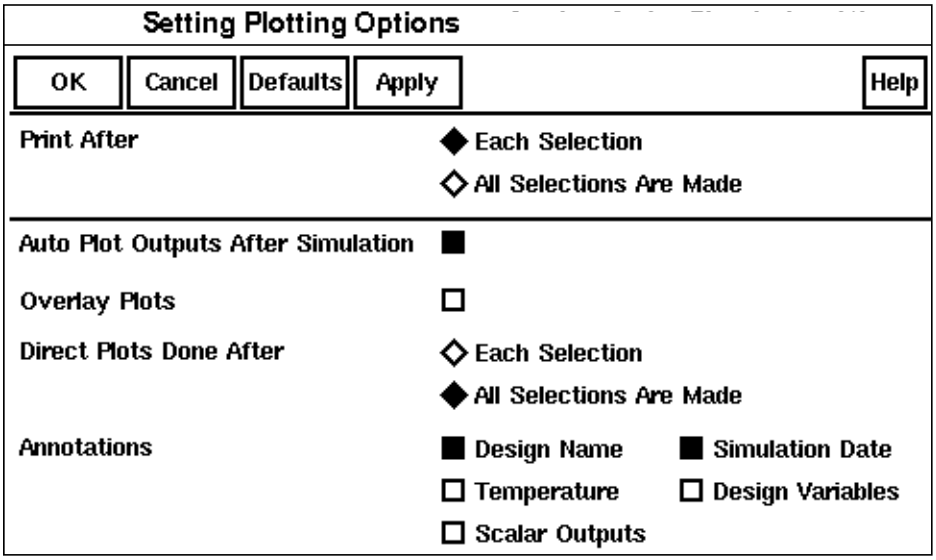
There are two modes for the *Direct Plot* commands:

- 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.



   For detailed information about the form, see "Setting Plotting Options" on page 366.

2. Choose one of the *Direct Plots Done After* options and click *OK*.

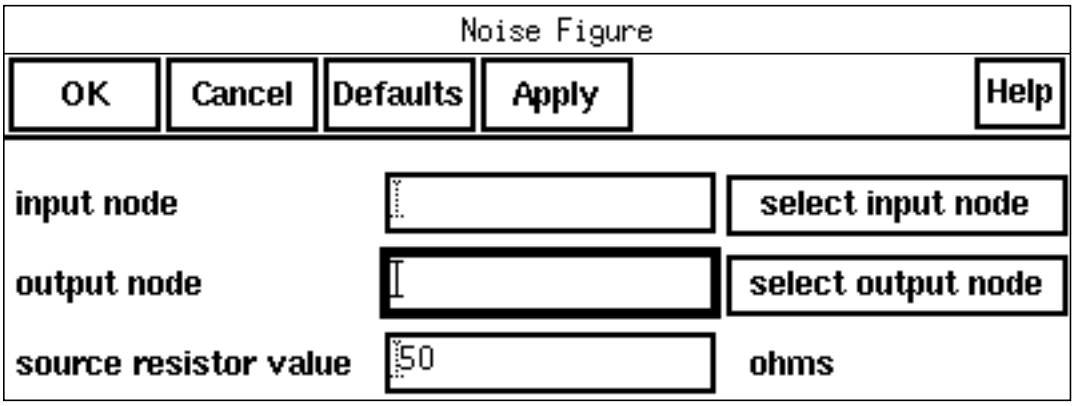
For Noise Figures

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,

Choose *Results – Direct Plot – Noise Figure*.

The Noise Figure form appears.



1. Type the name of the input node, or click *select input node* and click the node in the schematic.
2. Type the name of the output node, or click *select output mode* and click the node in the schematic.
3. 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 ≅ 4kTΔf

$$\mathbf{NF} = \mathbf{10}\log\left(\frac{\mathbf{VN2}*|\mathbf{Vin}|}{\mathbf{C}*|\mathbf{Vout}|*\mathbf{R}}\right)$$

with

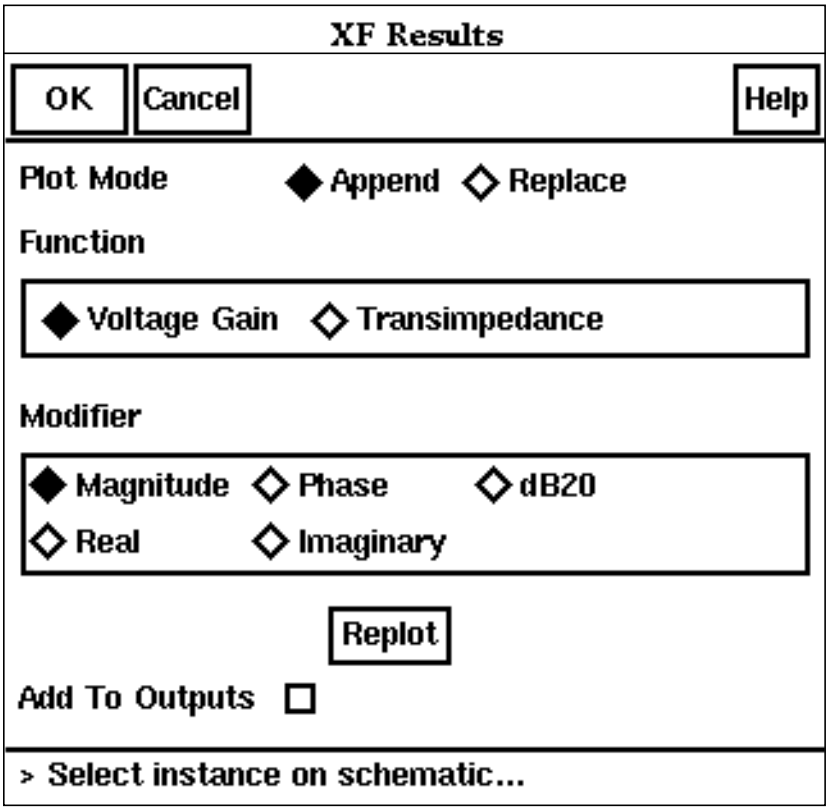
T = 291.5K and Δf = 1

For Transfer Functions

To plot the transfer function,

1. Choose *Results – Direct Plot – XF*.

   The XF Results form appears.



   For detailed information about the form, see "XF Results" on page 368.

2. 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.
3. Specify the modifiers as needed.
4. Select either the instance or instance terminal in the schematic.

   The Waveform window redisplays, showing the new plot.

5. 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.

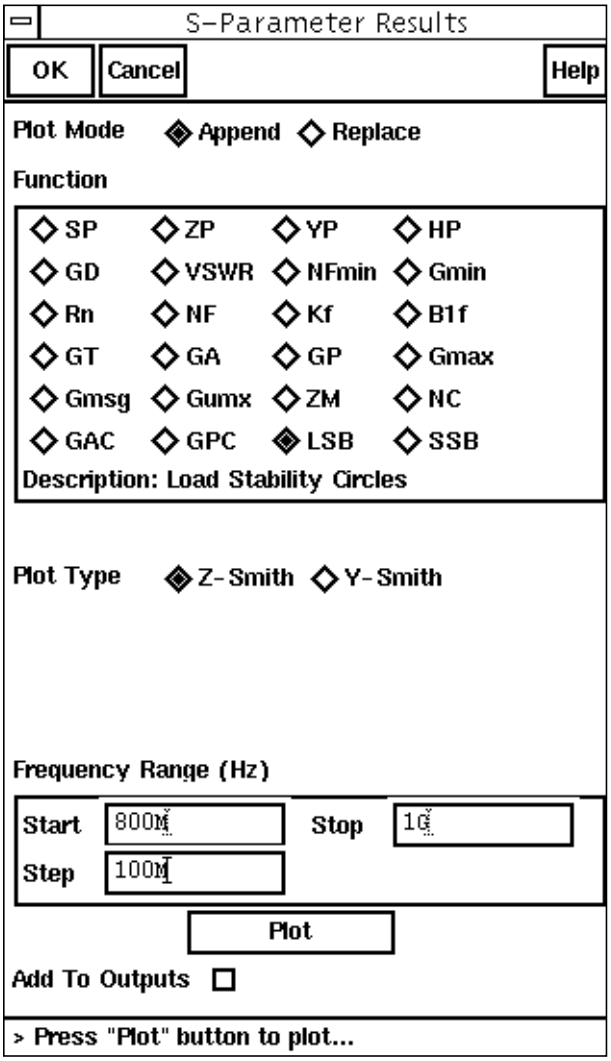
You define S-parameter direct plots with the S-Parameter Results form. If the form does not offer a plot you want to generate (for example, plots of complex computed results), use the waveform calculator.

The plots appear, by default, in the current Cadence®™ 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

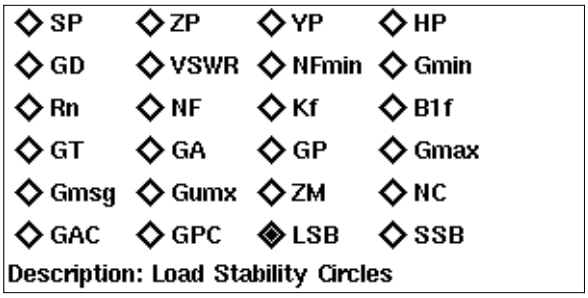


For detailed information about the form, see "S-Parameter Results" on page 369.

To plot S-parameter results,

1. Click a *Plot Mode* button to specify whether curves you plot are appended to or replace any existing curves in the subwindow.

2. Click the radio button for the S-parameter or noise-parameter function you want to plot.



   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.

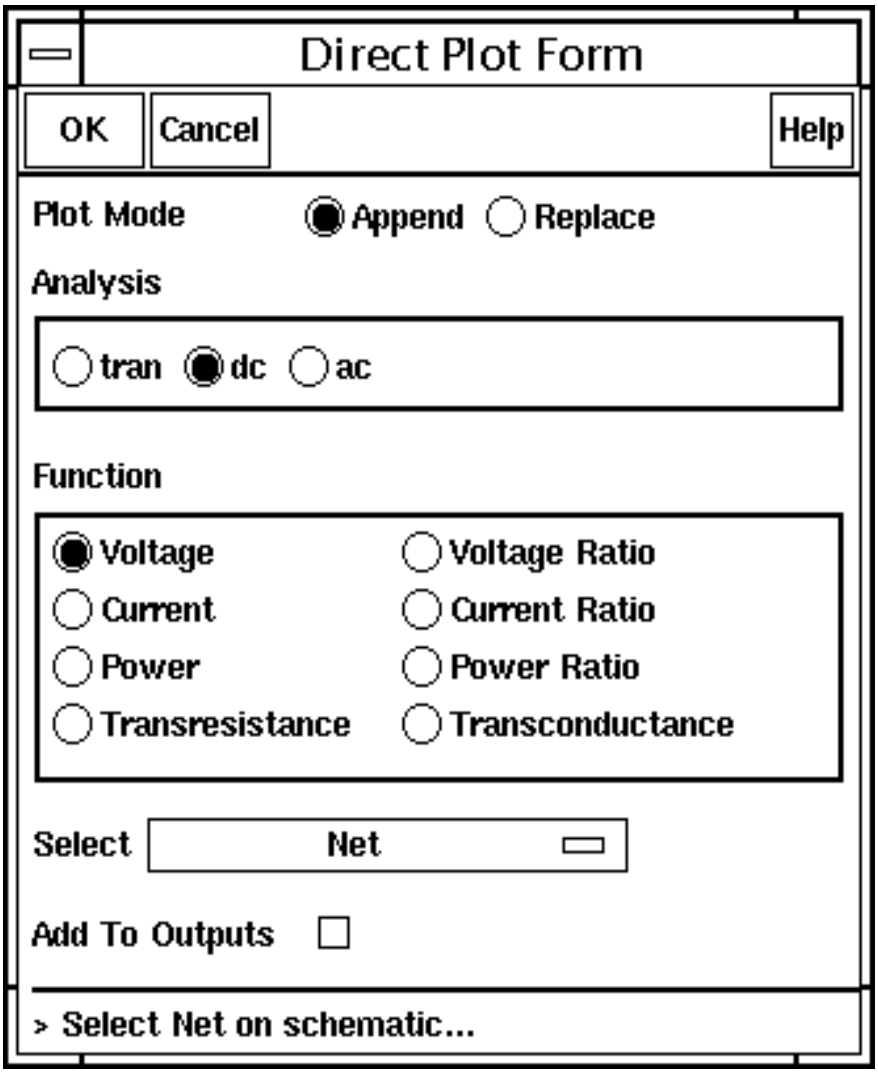
   For S, Z, Y, or H parameters (shown as *SP*, *ZP*, *YP*, and *HP* on the form), generate plots for ports 1 through 3 by clicking the appropriate parameter button at the bottom of the form. To generate plots for any higher-numbered ports, use the cyclic fields beside the buttons to specify the output 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

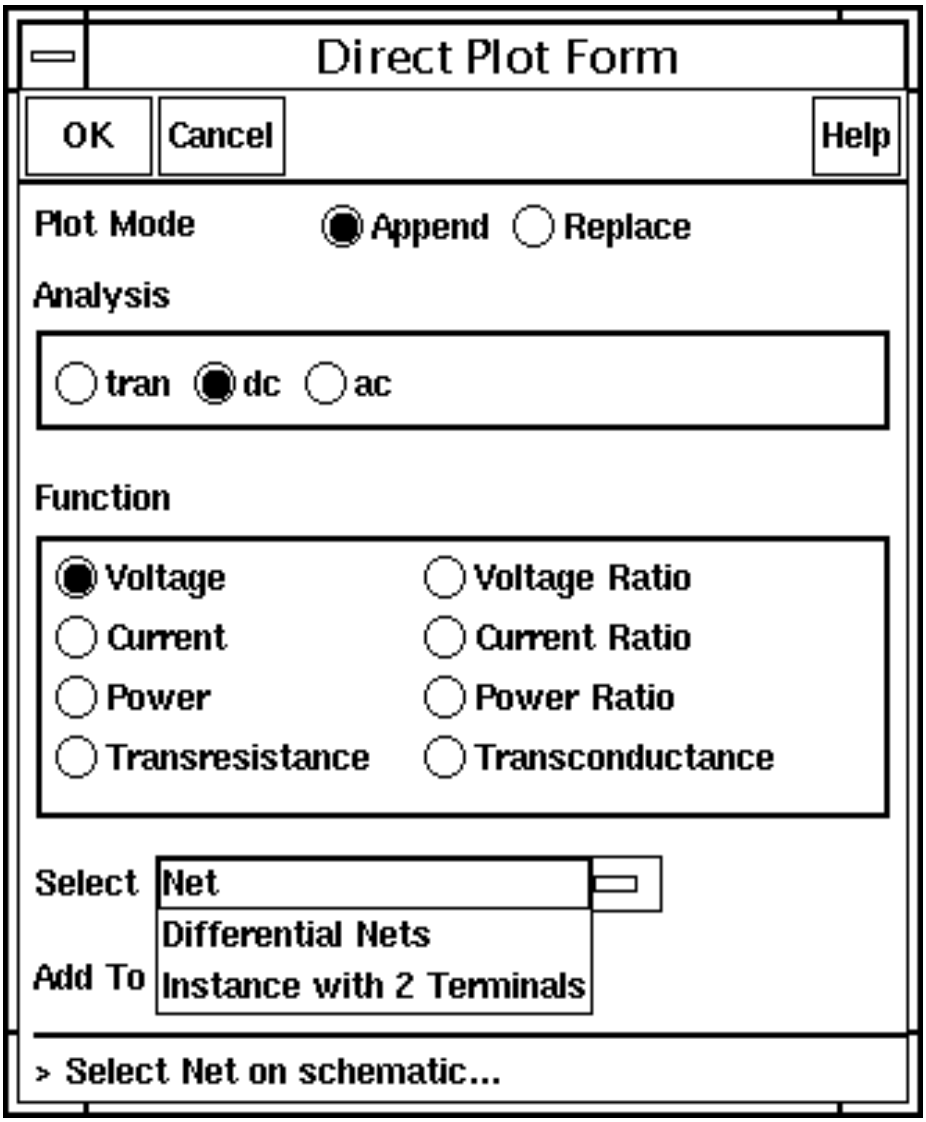
For DC

1. Choose *Results –> Direct Plot-> Main Form*. The *Direct Plot Form* appears. Select the *dc* option in the *Analysis* section.



2. 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.

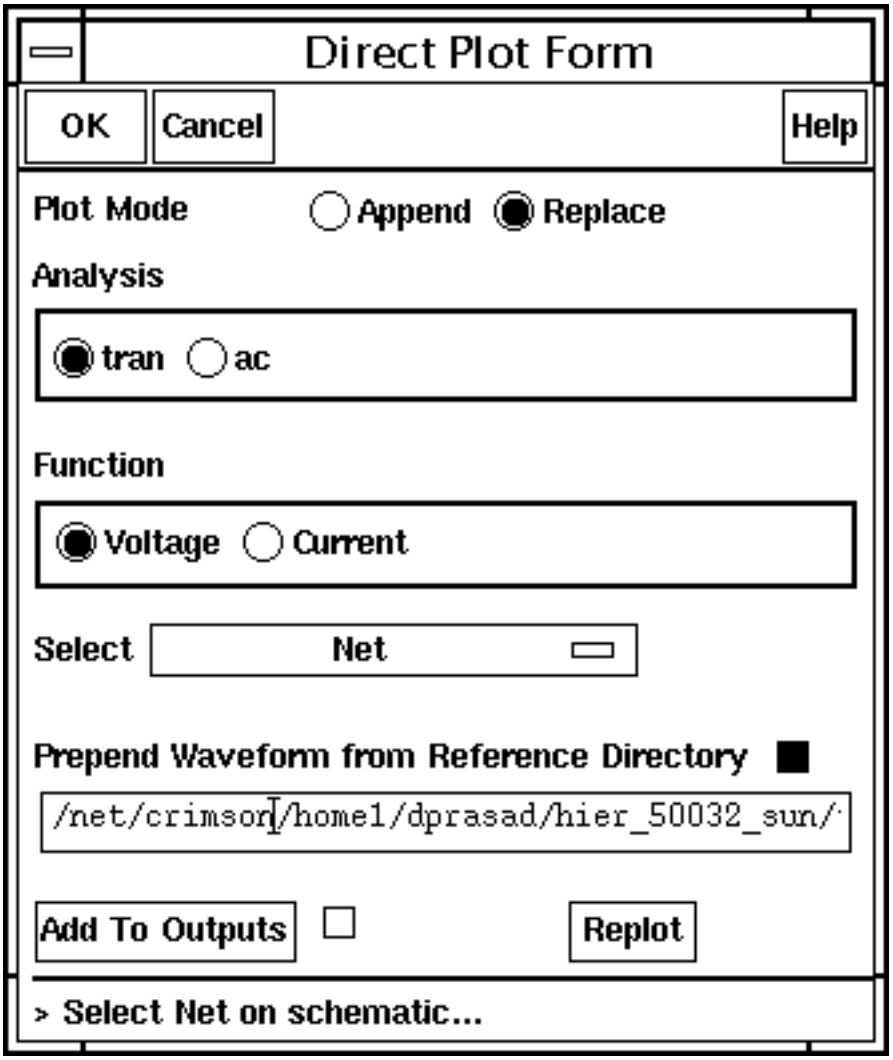
3. Choose the nets and terminals to plot. You can select *Net*, *Differential Nets* or *Instance with 2 Terminals*.



4. Enable *Add To Output*s to add expressions for the results to the outputs section and plot in *Replace* or *Append* mode.

For Transient Results

1. Choose *Results –> Direct Plot-> Main Form*. The *Direct Plot Form* appears. Select the *tran* option in the *Analysis* section.



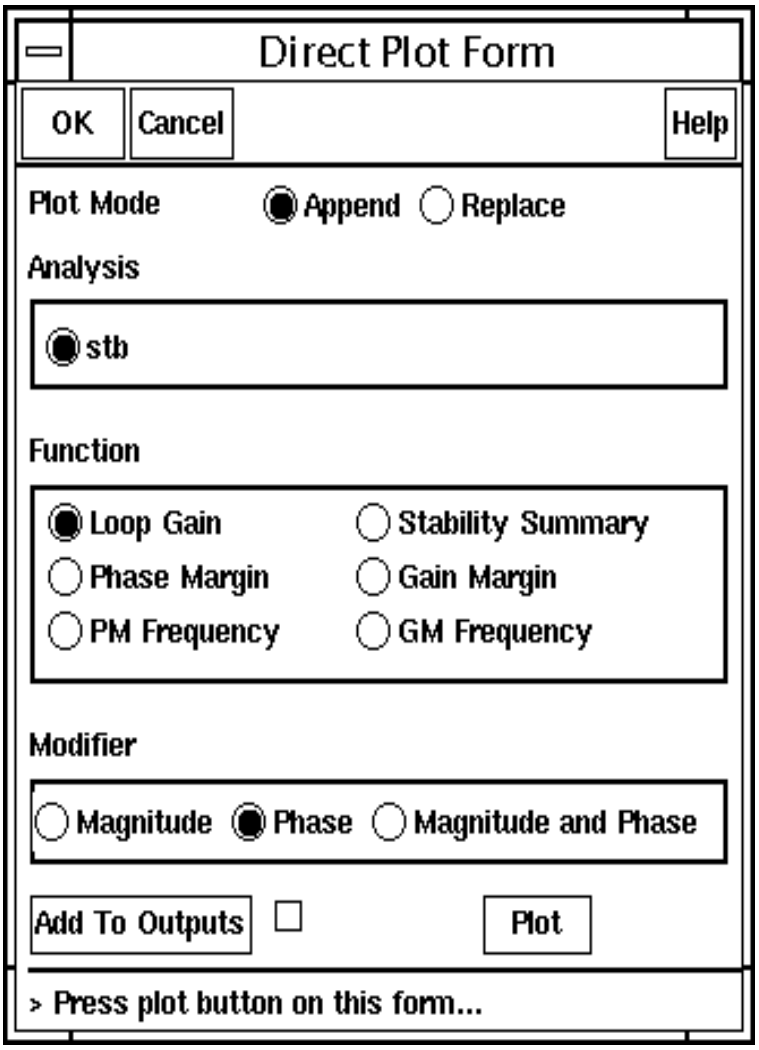
2. The functions that are available are: *Voltage* and *Current*. Based on the selected function and available data, the form changes dynamically to display the applicable options. Most of the options are similar to those available in the *Direct Plot* form For DC analysis.

3. 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.

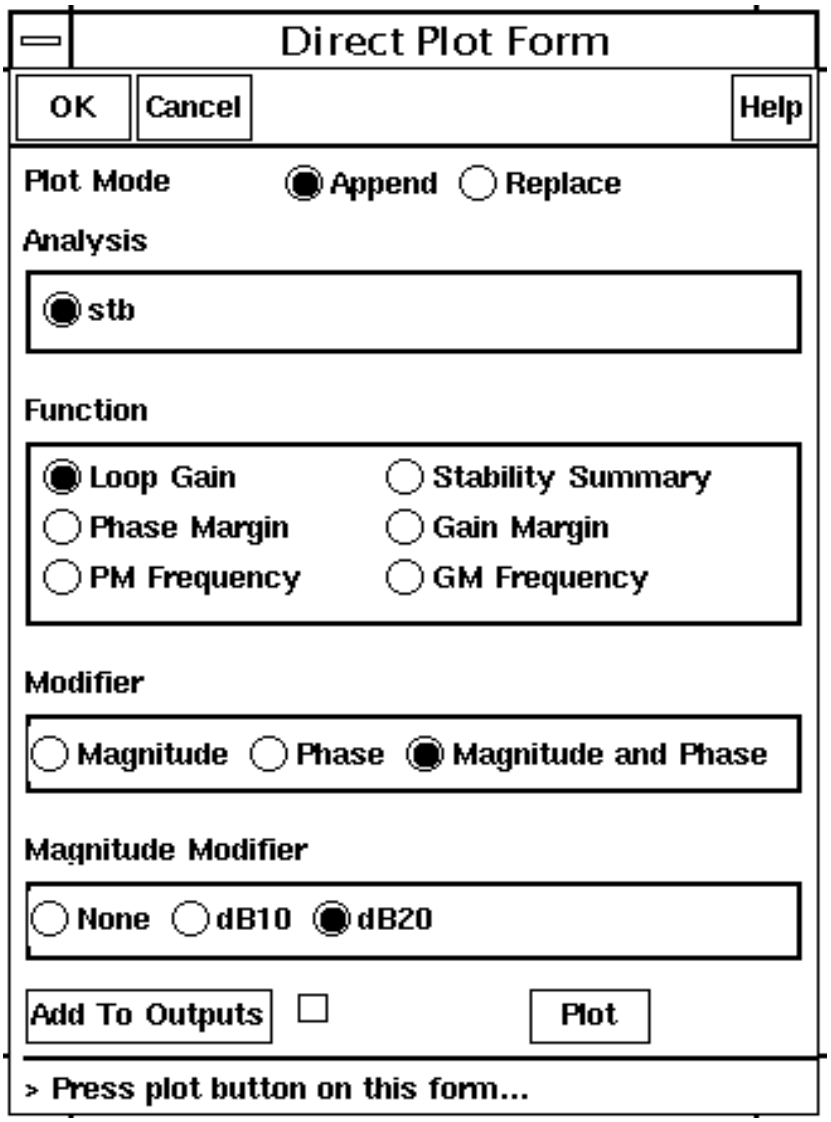
.



2. 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.

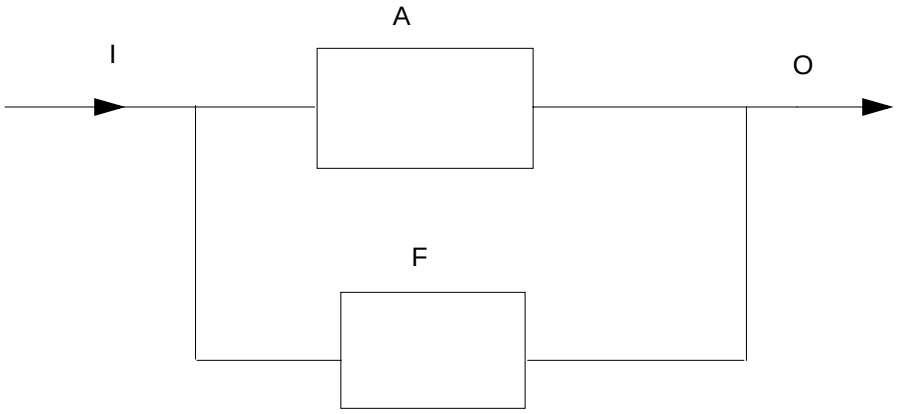
    a. When you select *Loop Gain*, the form re-displays to show the *Modifier* section.The loop gain output is a complex waveform and you can select it to plot *Magnitude*, *Phase* or both (*Magnitude and Phase*). Whenever you choose to plot *Magnitude*, the *Magnitude Modifier* section appears on the form. 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



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:

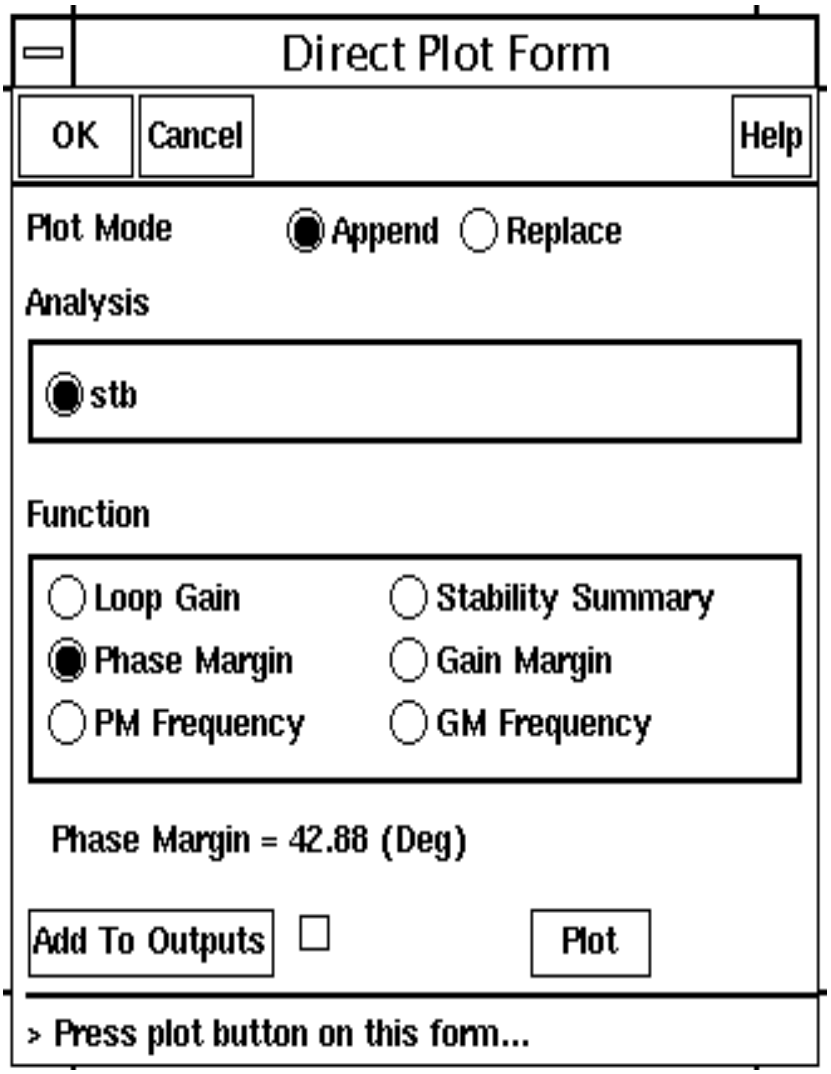
$$\frac{O}{I} = \frac{A}{1 - AF}$$

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:

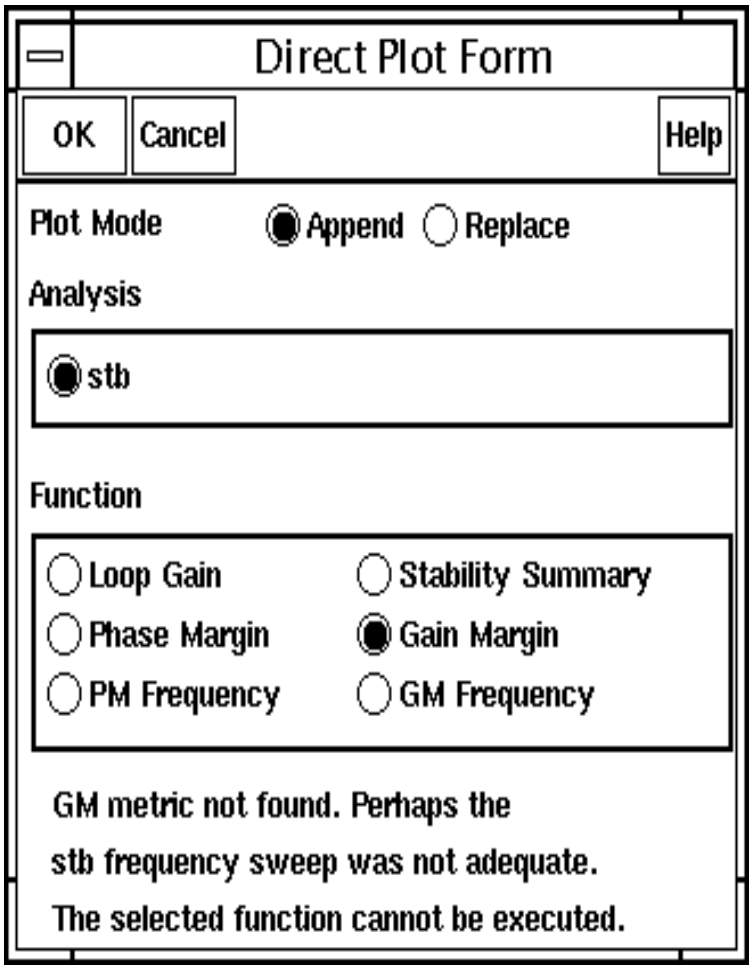
```
gainMargin( -1 * getData( "loopGain" ?result “stb” ), 1)
phaseMargin( -1 * getData( "loopGain" ?result “stb” ) )
```

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

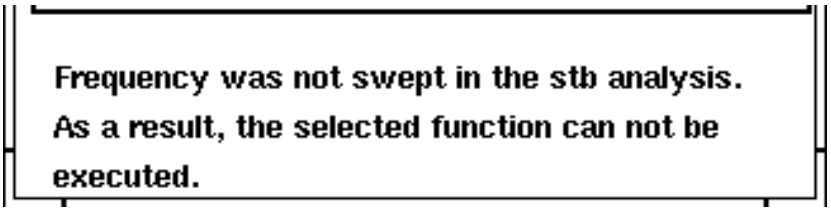
and the swept range is sufficient to calculate the values. When the selected margin data is scalar the values are displayed on the form itself.



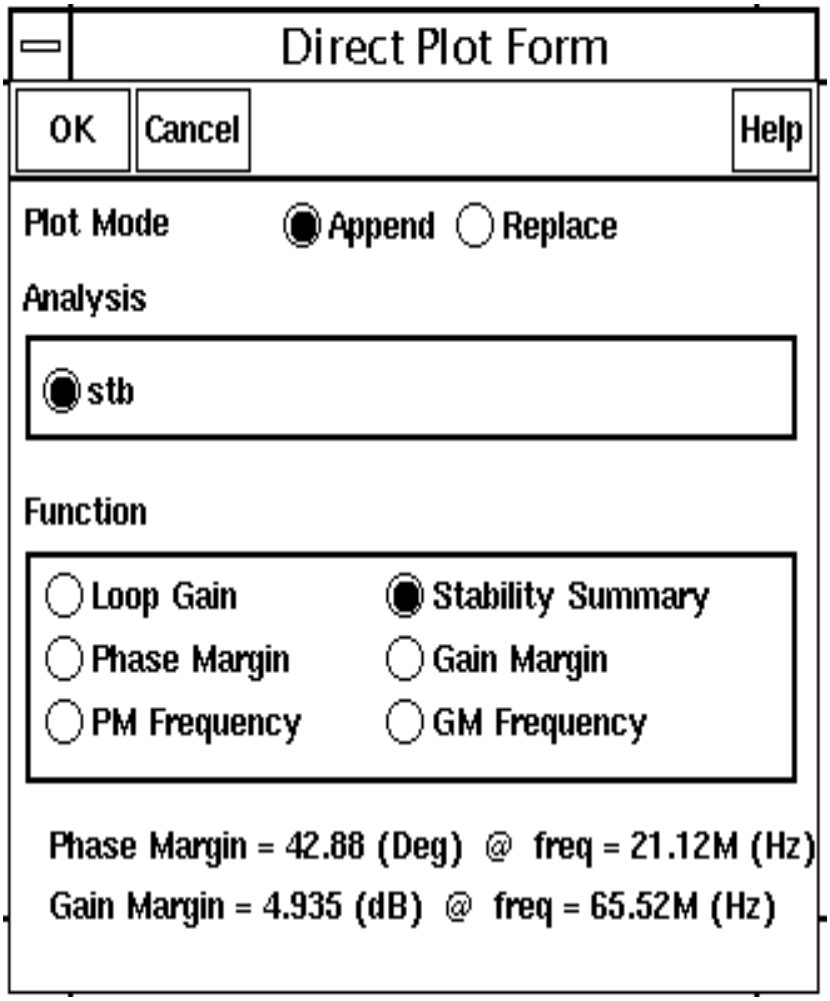
When the swept frequency range is not sufficient to calculate the selected margin data an error is reported in the *Direct Plot Form* and the *Plot* and *Add to Outputs* button are not available.



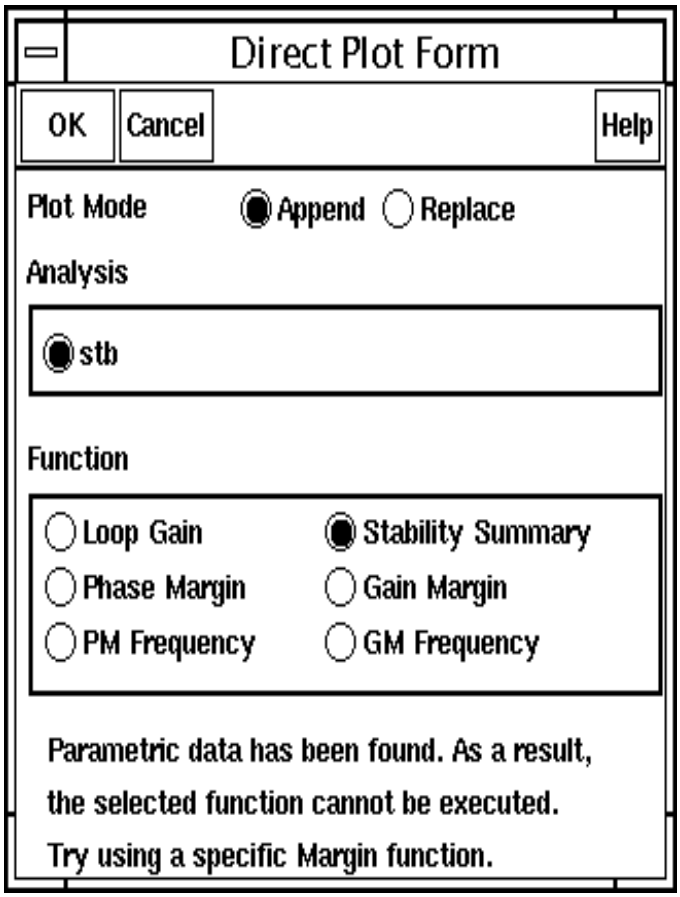
When frequency is not swept in the stability analysis and you choose any of the margin data functions an error is reported in the *Direct Plot Form* and the P*lot* and *Add to Outputs* button are not available.



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.



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.



3. Enable *Add To Output*s and click the *Plot* button to add expressions for the results to the outputs section and plot in *Replace* or *Append* mode.

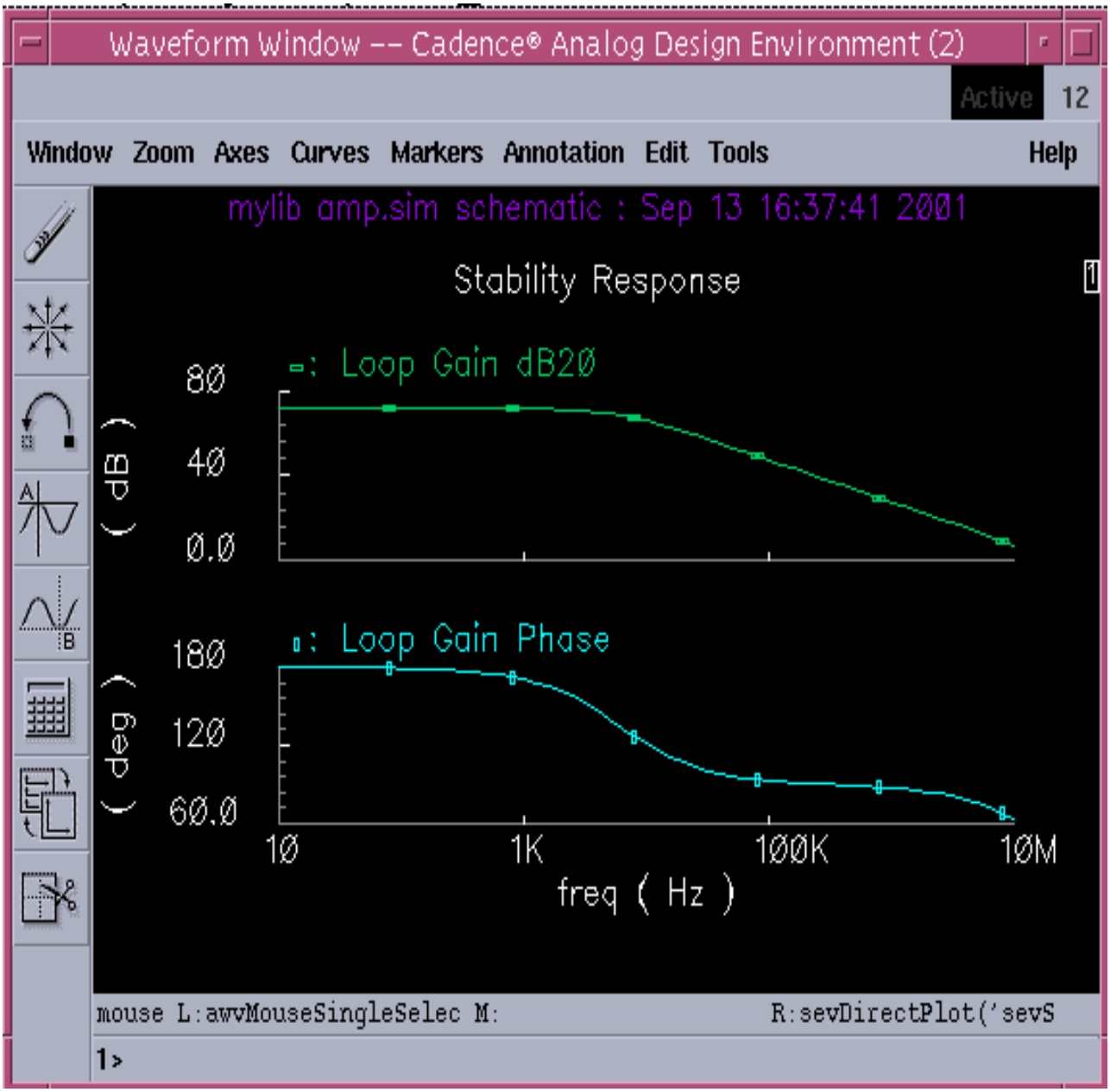
   Note: This option makes no checks for duplication in outputs.

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:

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.

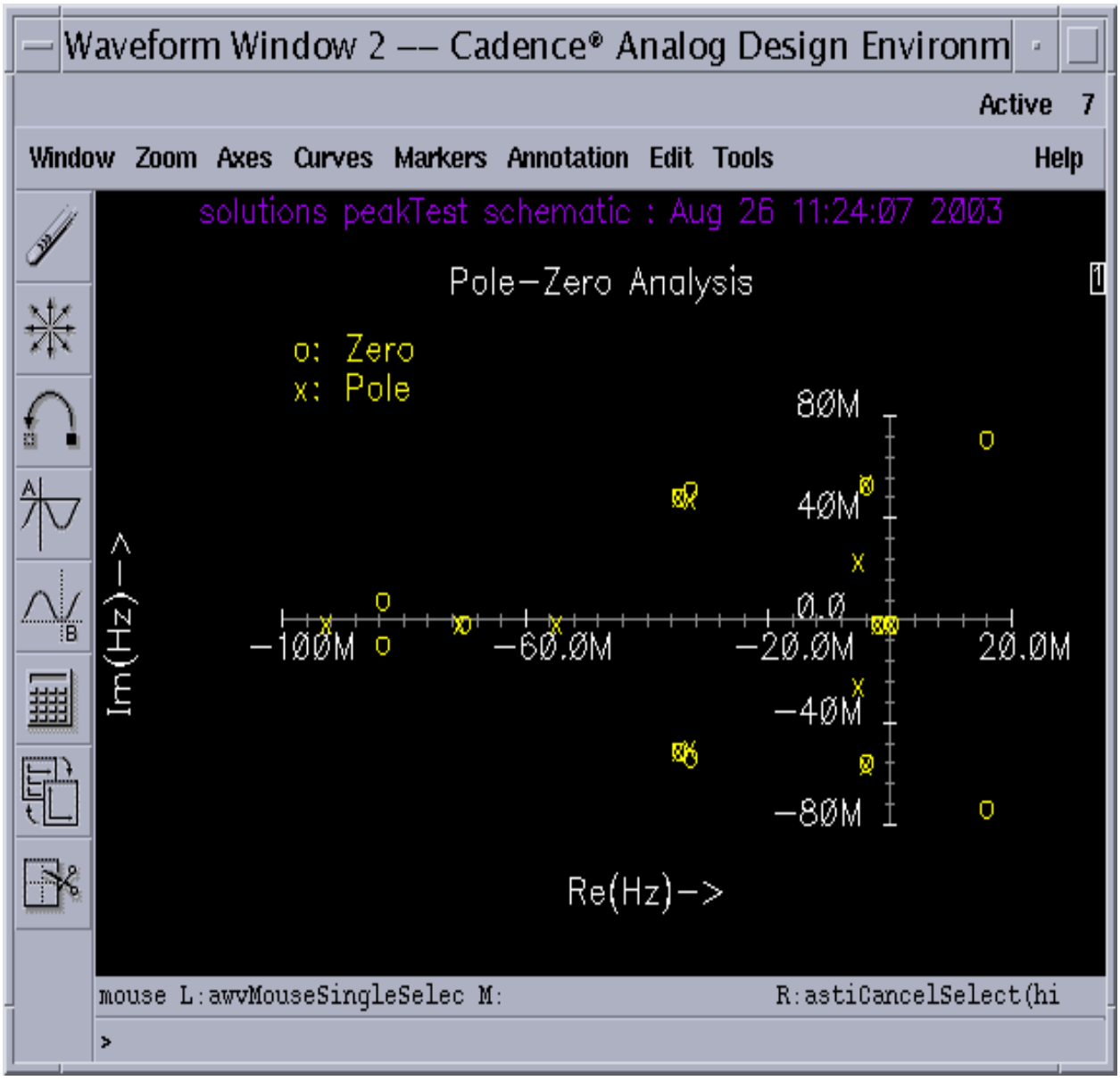
4. Set the required options in the *Filter Out* section and click *OK*. This section provides a combination of filtering mechanisms that you can select in order to plot the poles and zeros. These are:

   - *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**.

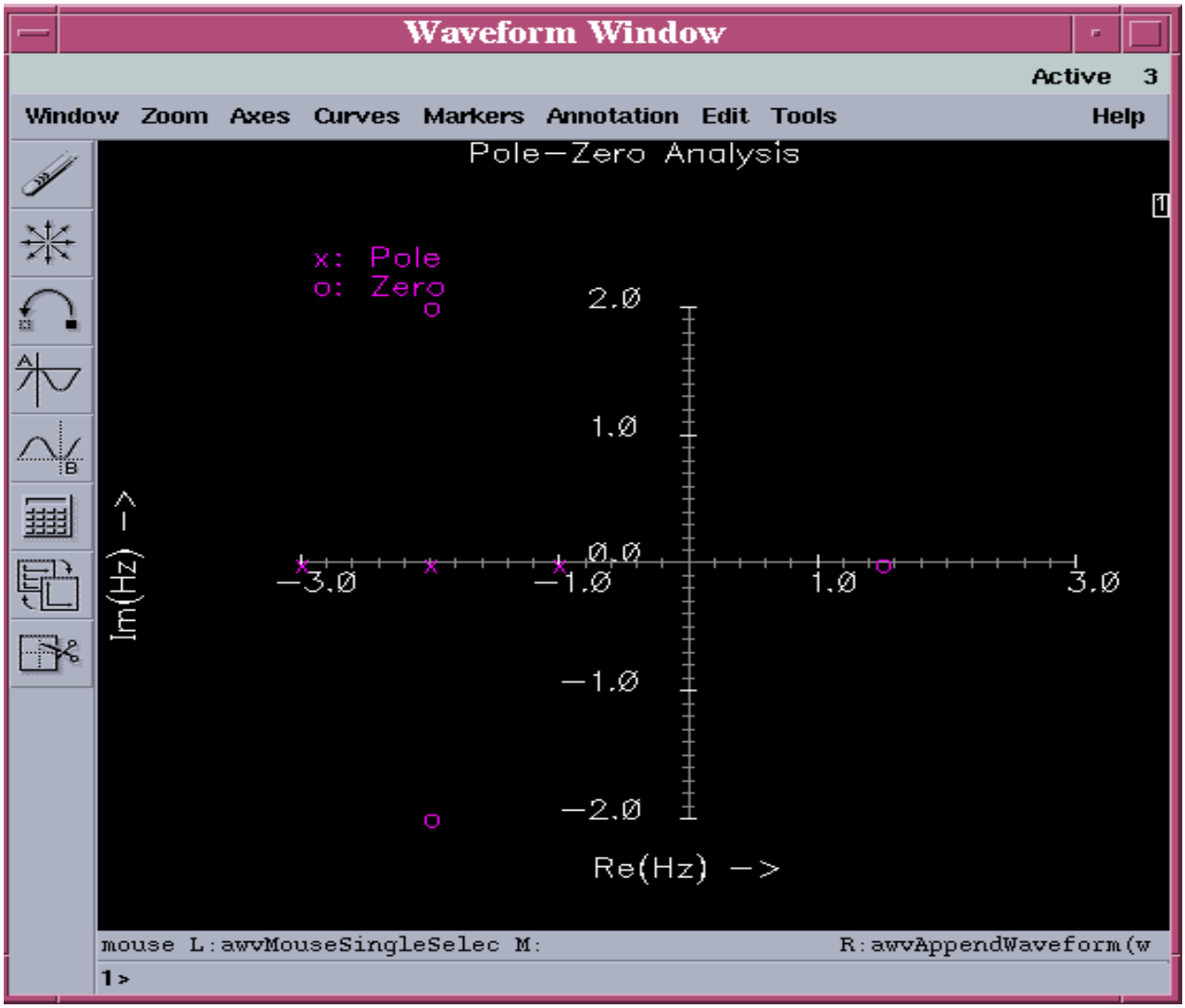
Example

Consider an example where there are 3 poles and 3 zeros, as follows:

Pole_1 : (-3, 0)
Pole_2 : (-2, 0)
Pole_3 : (-1,0)

Zero_1 : (-2,-2)
Zero_2 : (-2,2)
Zero_3 : (1.5,0)

The results are displayed in the *Waveform Window,* as follows:



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.

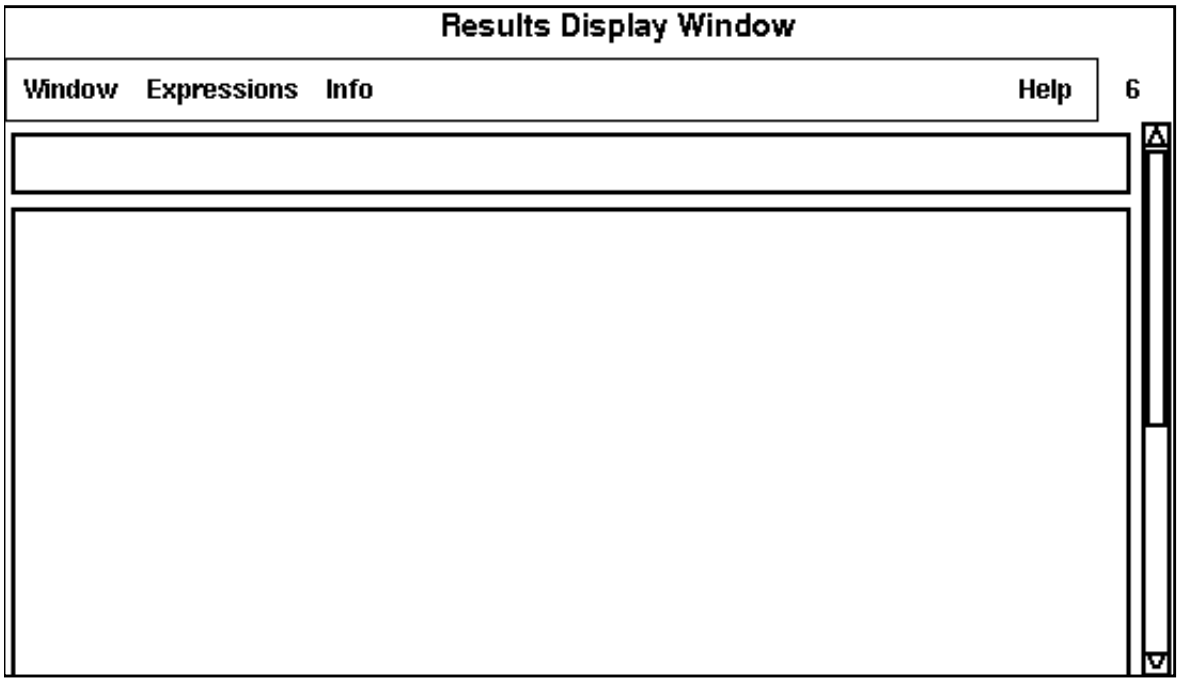
Overview of Printing

The following commands and tools within the analog circuit design environment print text results and reports to the Results Display Window.

- *Results – Print* menu commands in the Simulation window

- *Print* and *printvs* commands in the Calculator
- *Statistics – Print* menu commands in the Statistics tool
- *Display Intercept Data* commands in the Waveform window markers forms

Using any of these commands opens the Results Display Window.



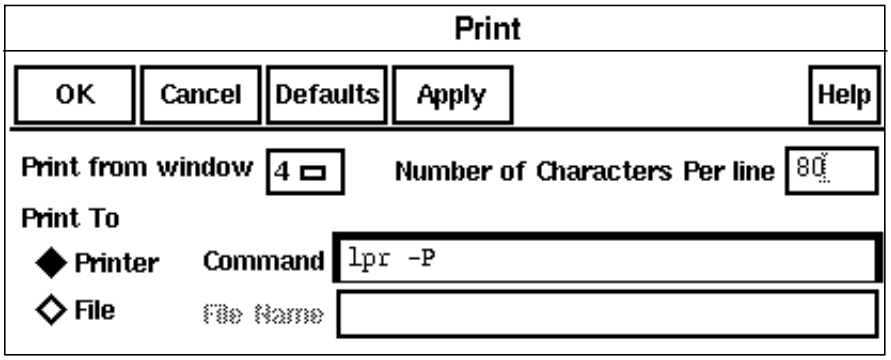
For guidance on using the Results Display Window to perform tasks, see the following sections.

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.



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*.

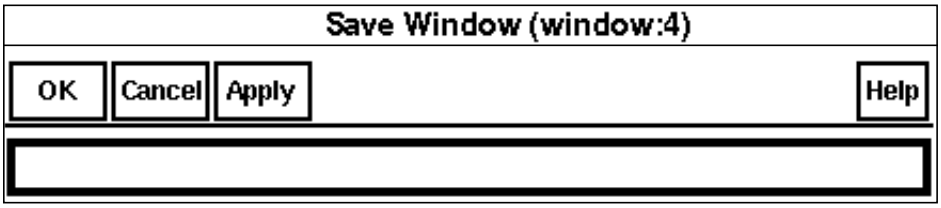
Saving State

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.



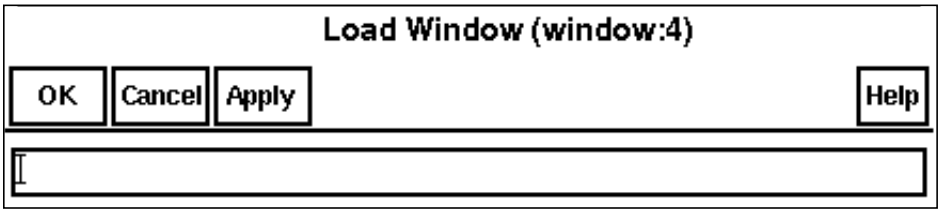
2. Type a filename in the field.
3. Click *OK*.

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.

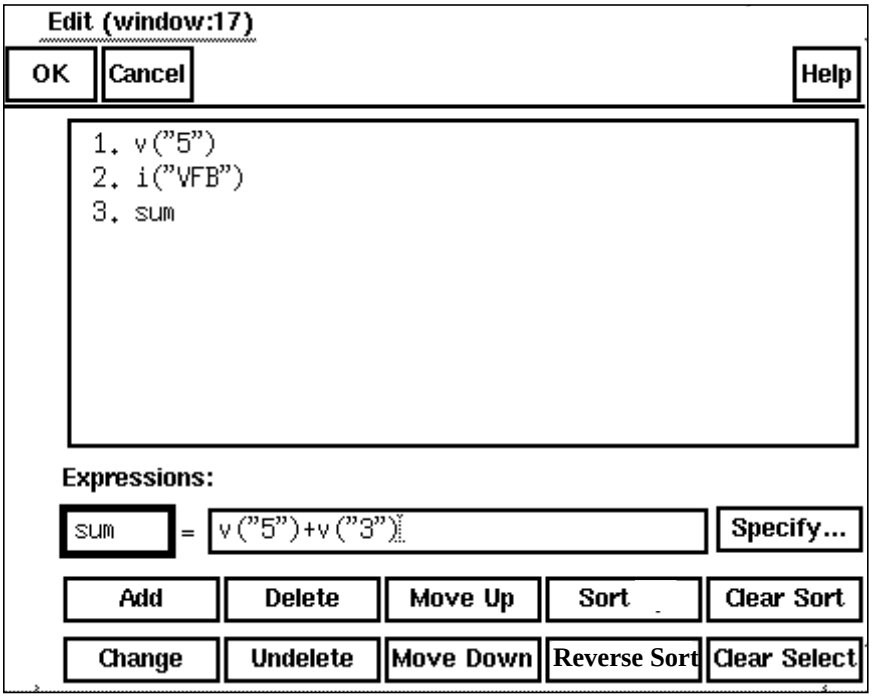
Editing Expressions

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.



2. Edit the expressions using the form buttons and fields.

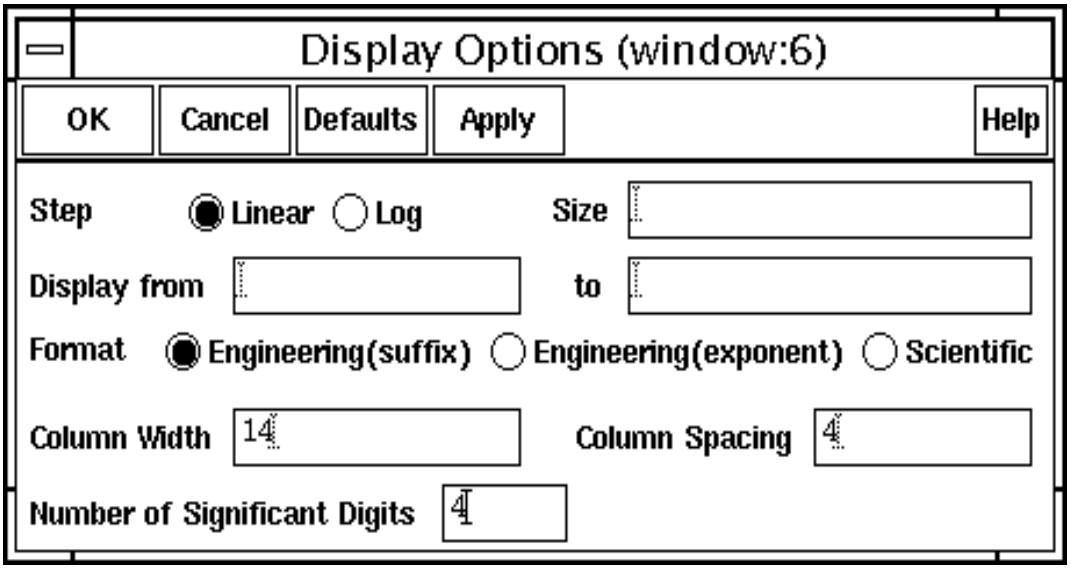
Expressions	The field to the left of the equal sign shows the aliased name of the expression to the right. Naming expressions is optional and the field might be blank. If aliases are used, they are shown in the list box and the title line of the print window list box.
Specify	Retrieves the expression 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.



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` :

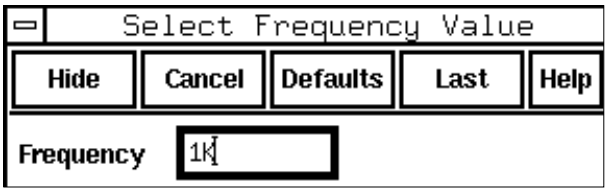
```
asimenv.printing  printInlines  boolean  nil
```

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.



   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.

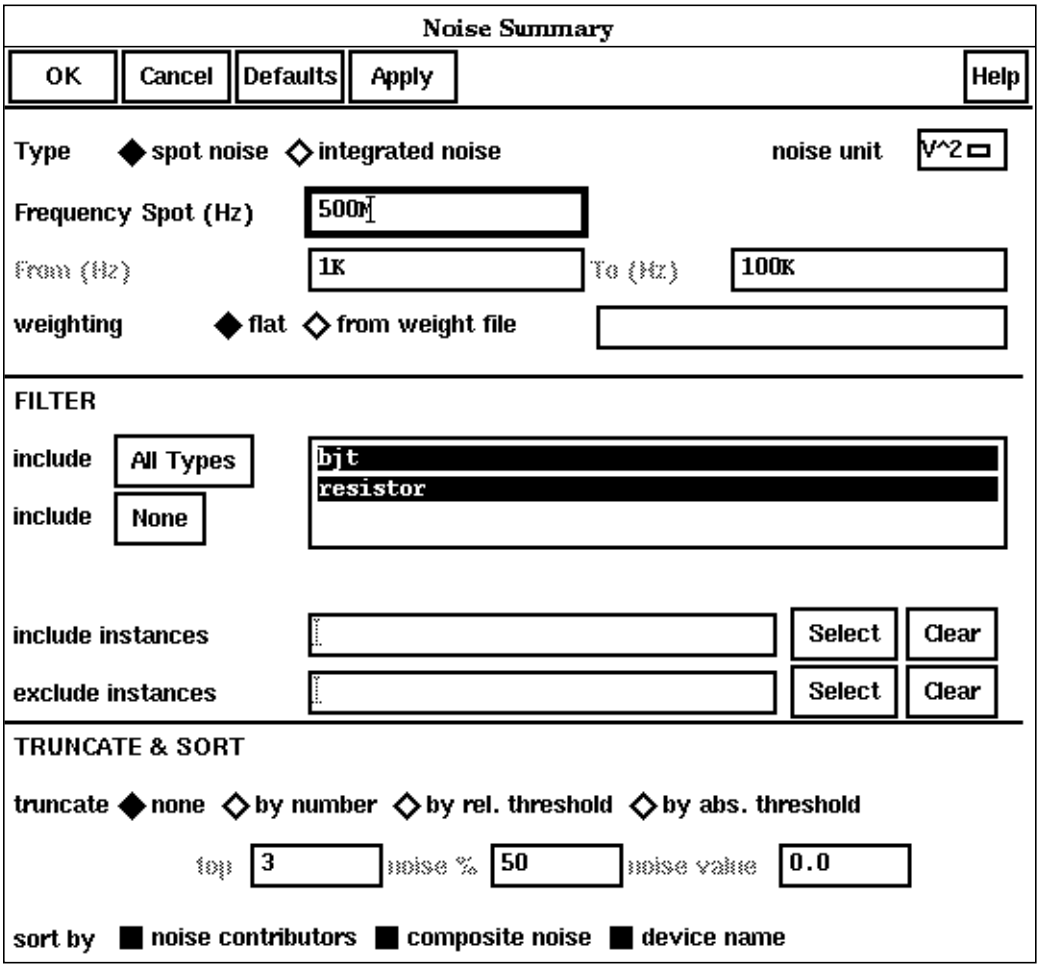
4. Click an instance or node.

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.



For detailed information about the form, see "Noise Summary" on page 374.

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.

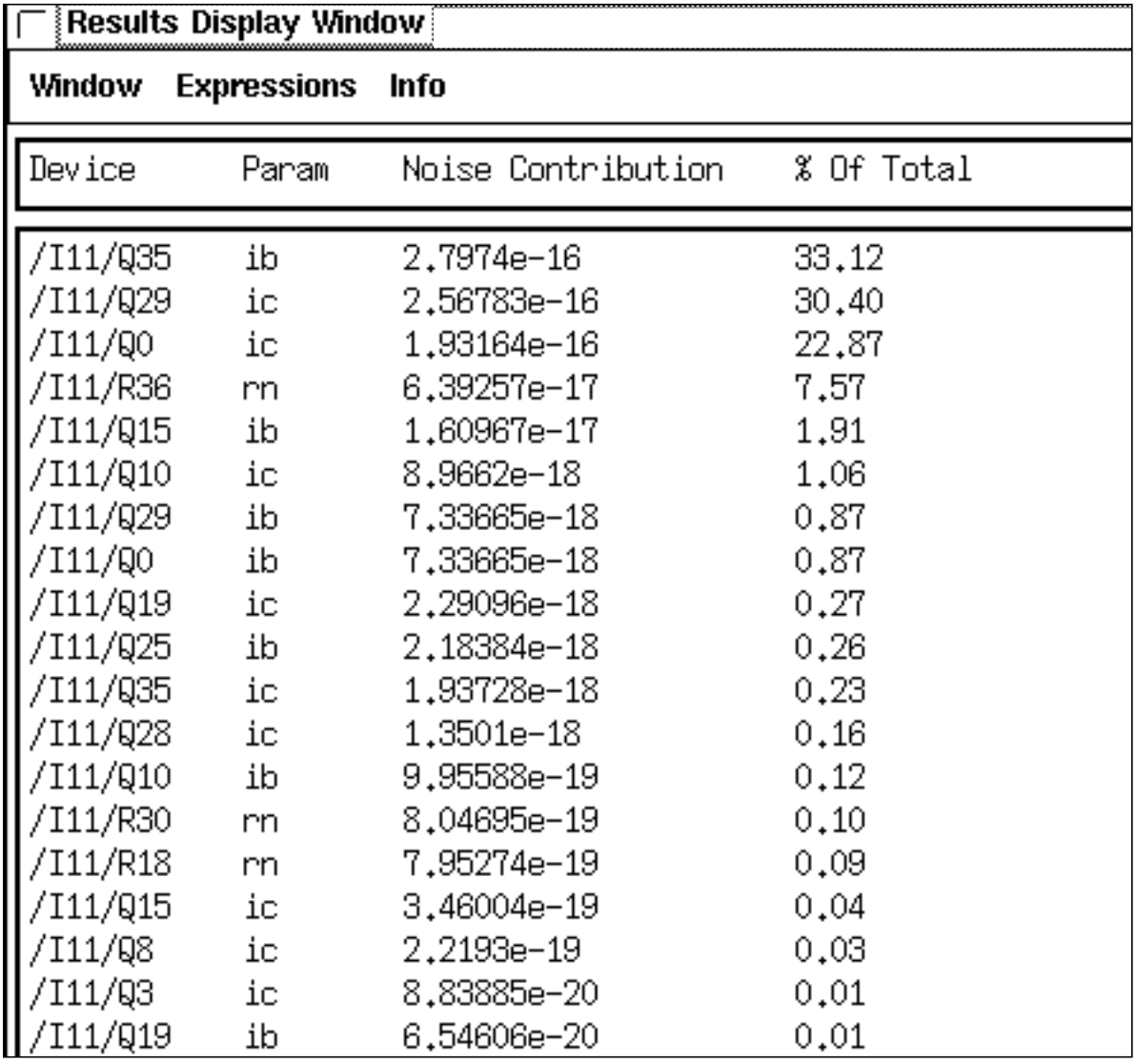
   The *from weight file* selection specifies that, before the integration is performed, the noise contributions of particular frequencies in the original waveform be weighted by factors supplied from an input file. The weighting file must have one of the following entries on the first line: `db`, `mag`, `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.



Results Display Window

Window Expressions Info

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.
```

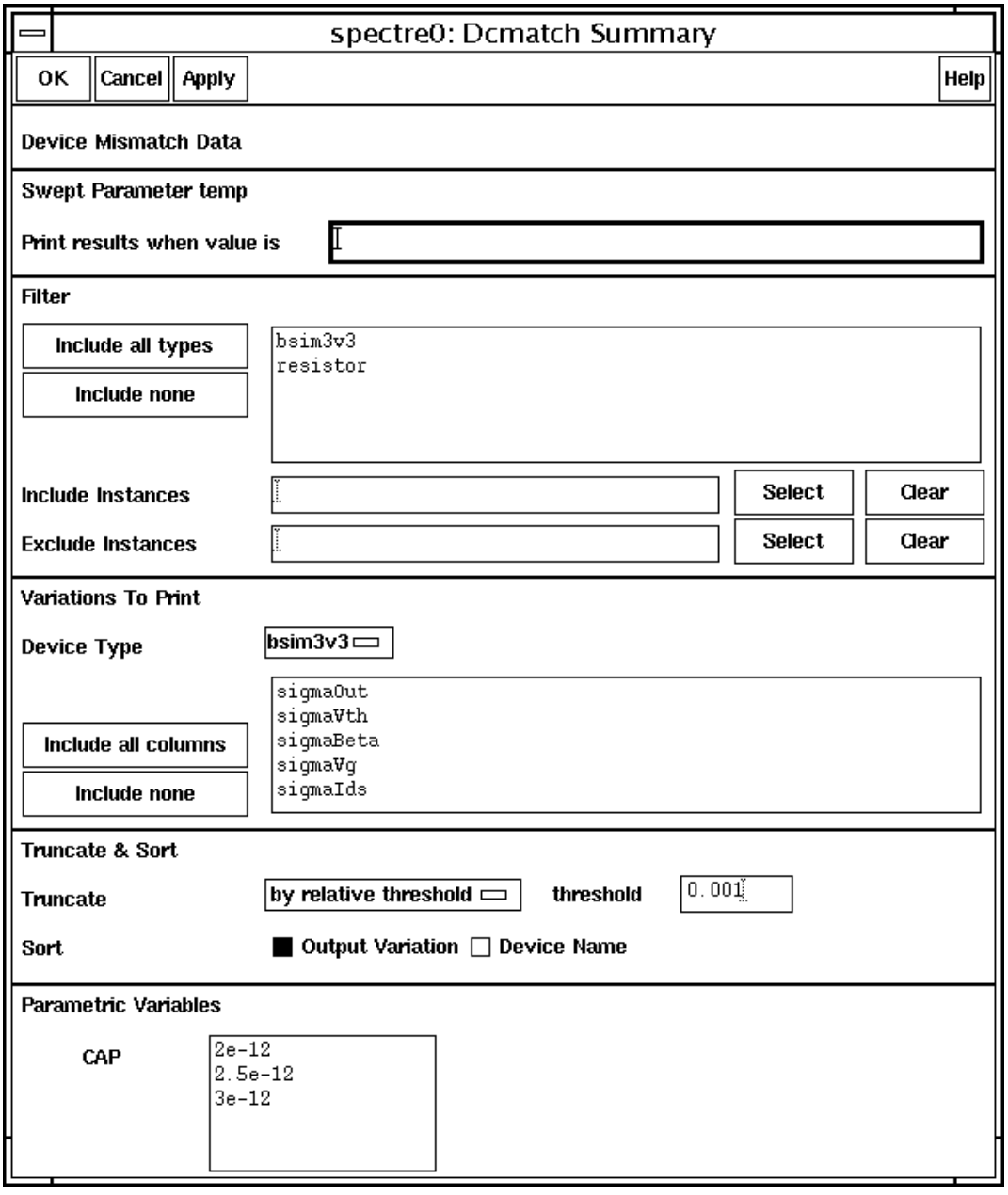
Printing DC Mismatch Summary

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



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 *Parameteric Variables* section of the form, for you to select the available values.

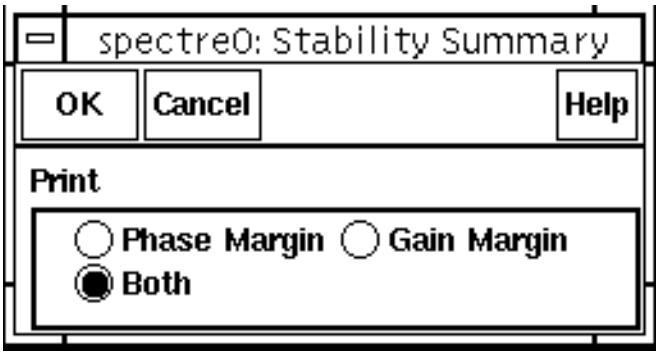
3. Specify the type of devices you need to print the results for, in the *Filter* section. The *Include all types* and *Include none* buttons can be used to include or exclude all types at a single click. You can include specific instances or exclude specific instances. You can either type the instance 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 *Parameteric 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.

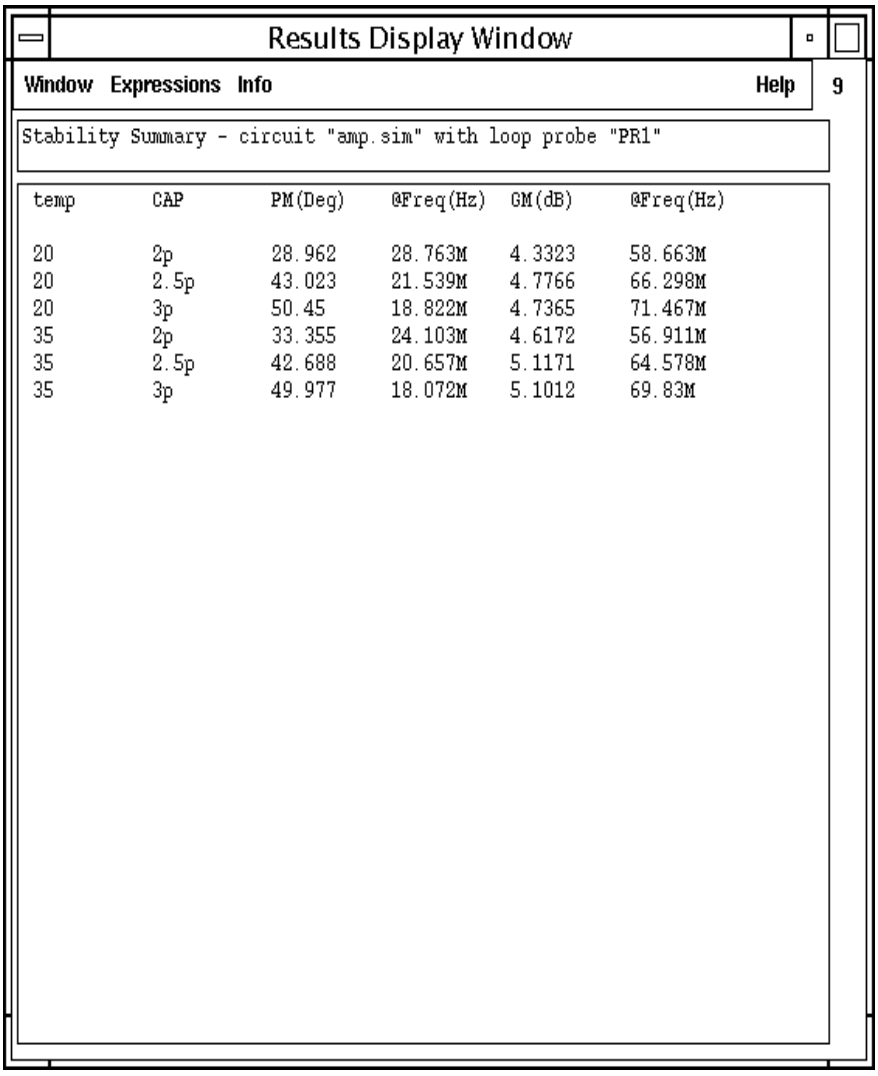


   The form handles parametric (family) data and prints results at all available sweep points.

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



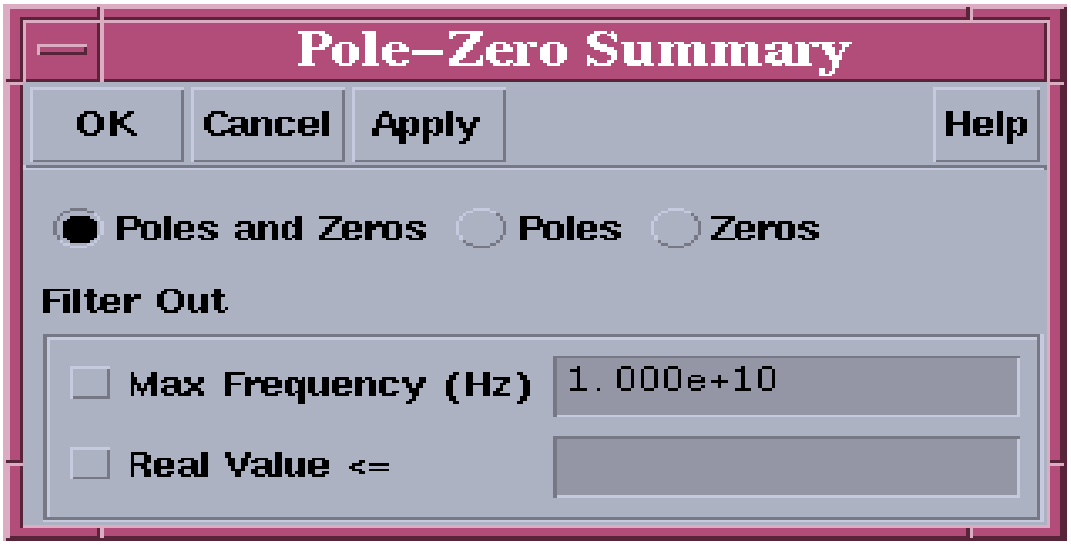
Printing Pole Zero Summary

To print the *Pole Zero* summary in your circuit,

1. Choose *Results->Print->Pole Zero Summary.* The *-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.



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.

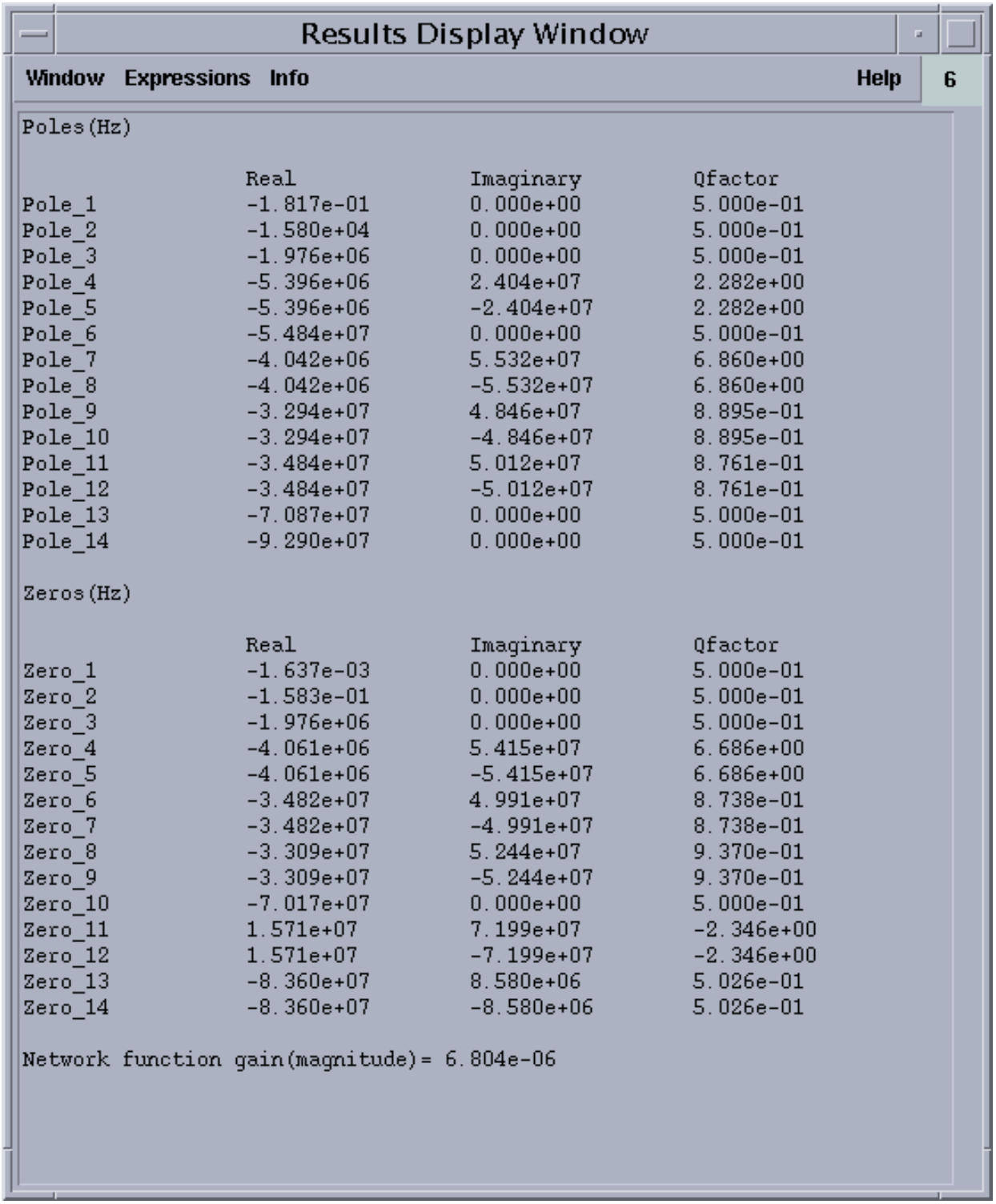
3. Set the required options in the *Filter Out* section and click *OK*. This section provides a combination of filtering mechanisms that you can select in order to plot the poles and zeros. These are:

   - *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:



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.

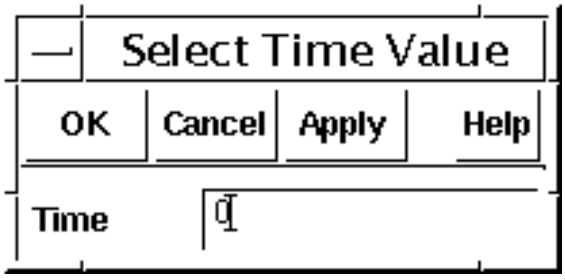
   You are prompted to select nets for the VDC output.
3. Click a node.

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.



   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 Statistical Reports or Calculator Results

For information about statistical reports, read the *Cadence® Advanced Analysis Tools User Guide*.

For information about the calculator *print* and *printvs* commands, read the *Waveform Calculator User Guide*.

Using SKILL to Display Tabular Data

You can use the Cadence® 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","??")`

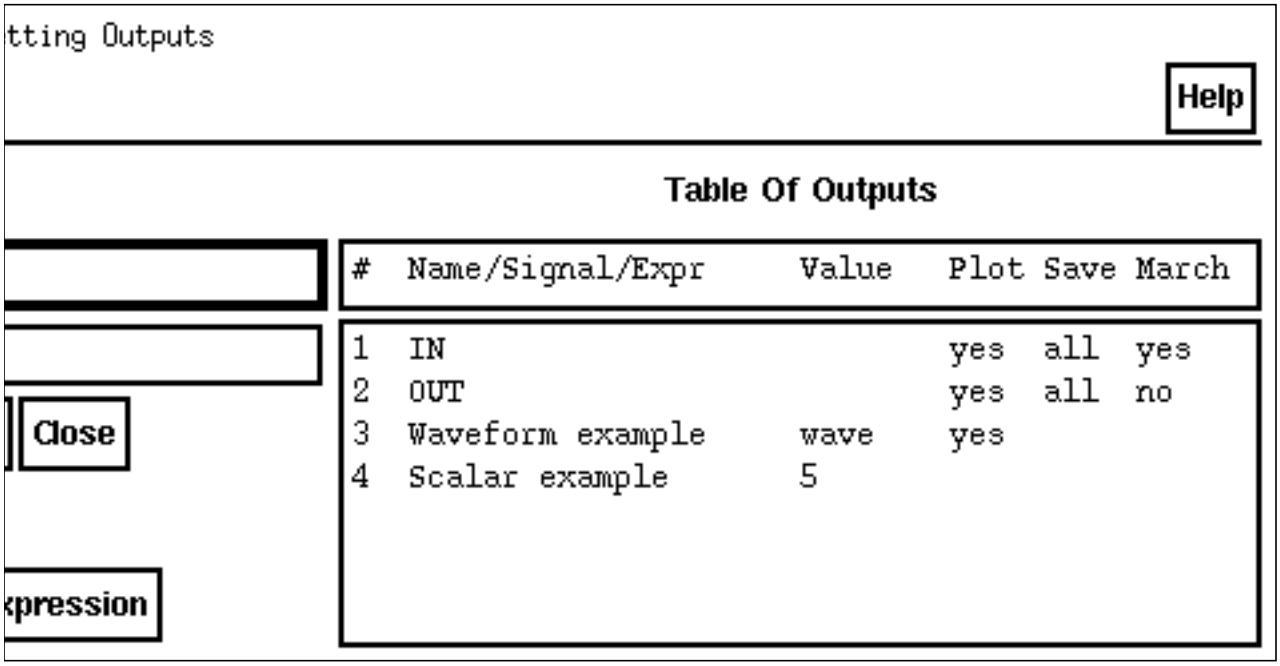
To display…	Type this command in the CIW
A list of just the operating-point parameter names for `R1`	`OP("/R1","?")`
A single operating-point parameter (`v` for voltage, for example) and its value for `R1`	`OP("/R1","v")`
A list of transient operating-point parameter names and their values for `C1`	`OPT("/C1","??")`
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")`
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.

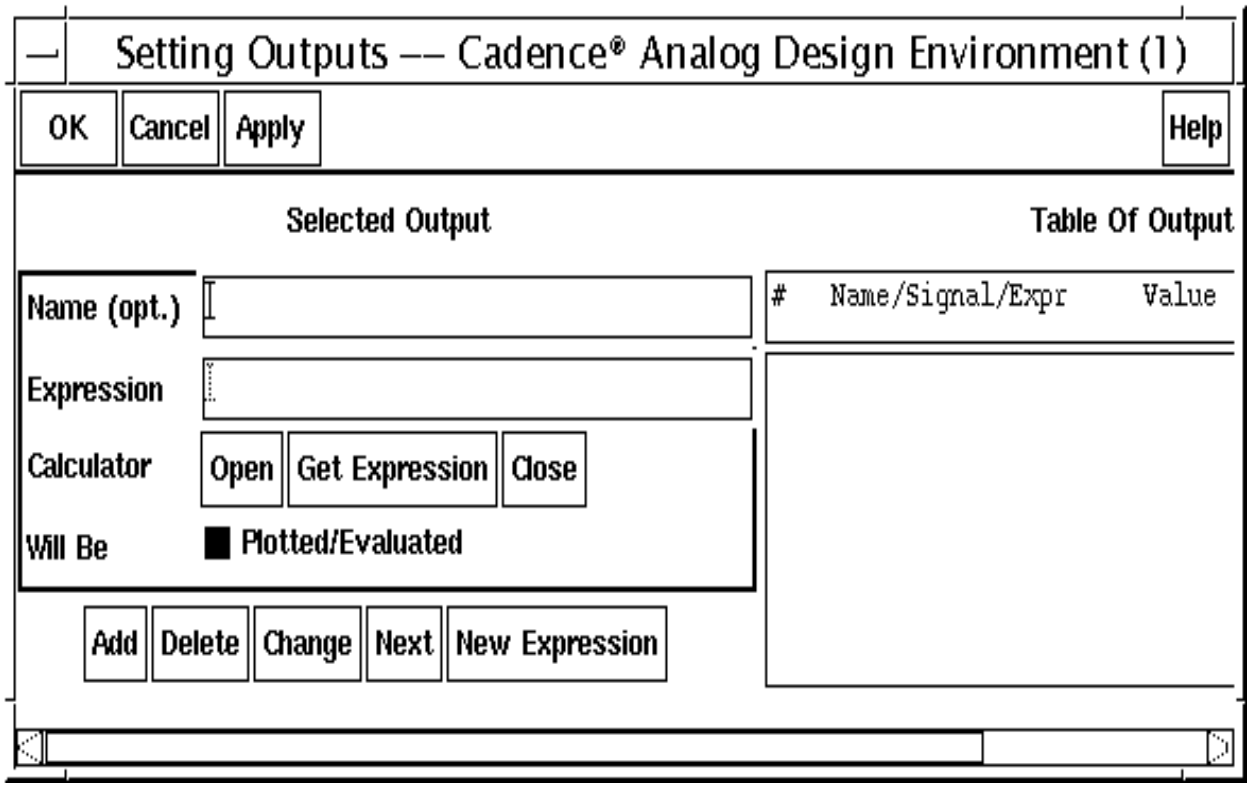


Defining Expressions

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 373.

1. (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
   ```

2. 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*.

3. Click *Add*.

   The expression is added to the *Table Of Outputs* list box.

Plotting Expressions

To plot expressions,

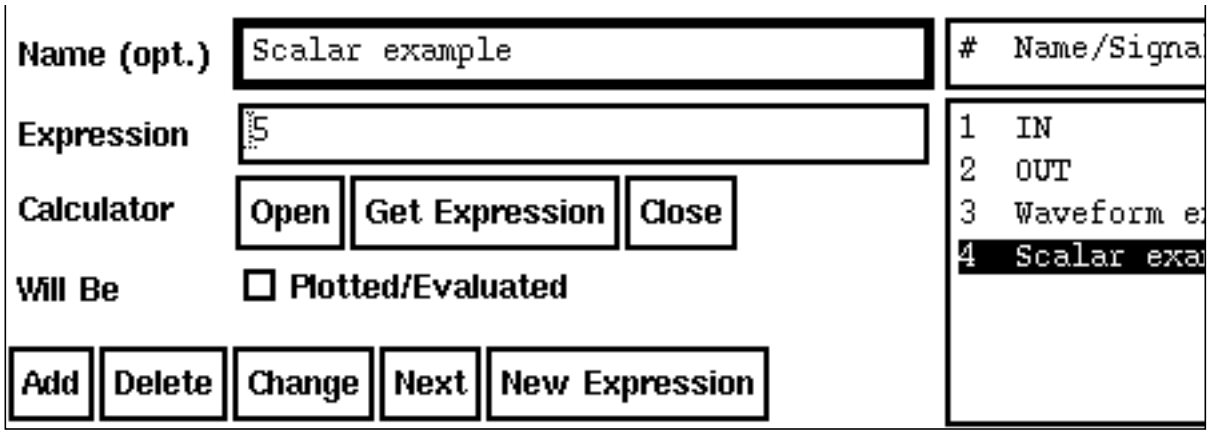
➤ Choose *Results – Plot Outputs – Expressions*.

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.



4. Click *Change*.

Now when you choose *Results – Plot Outputs – Expressions*, the expression you deactivated is not plotted.

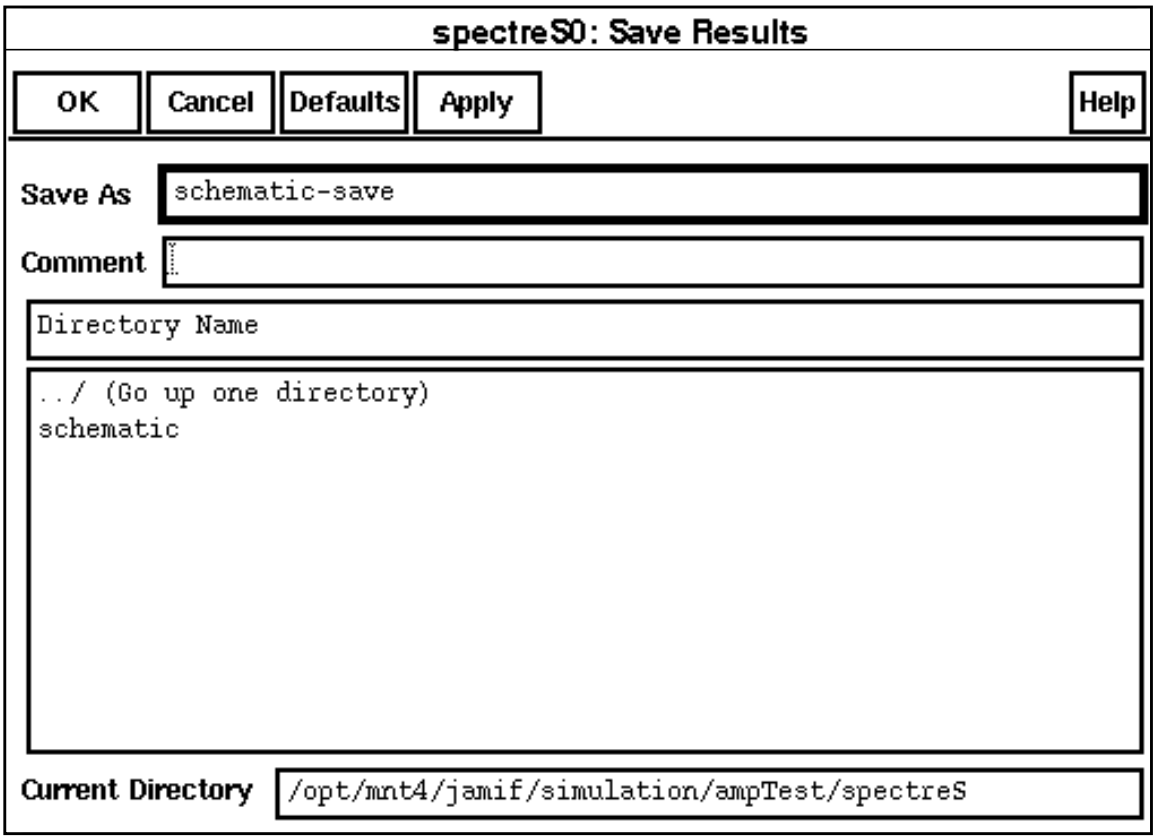
Annotating Simulation Results

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.



   For detailed information about the form, see “Save Results” on page 376.

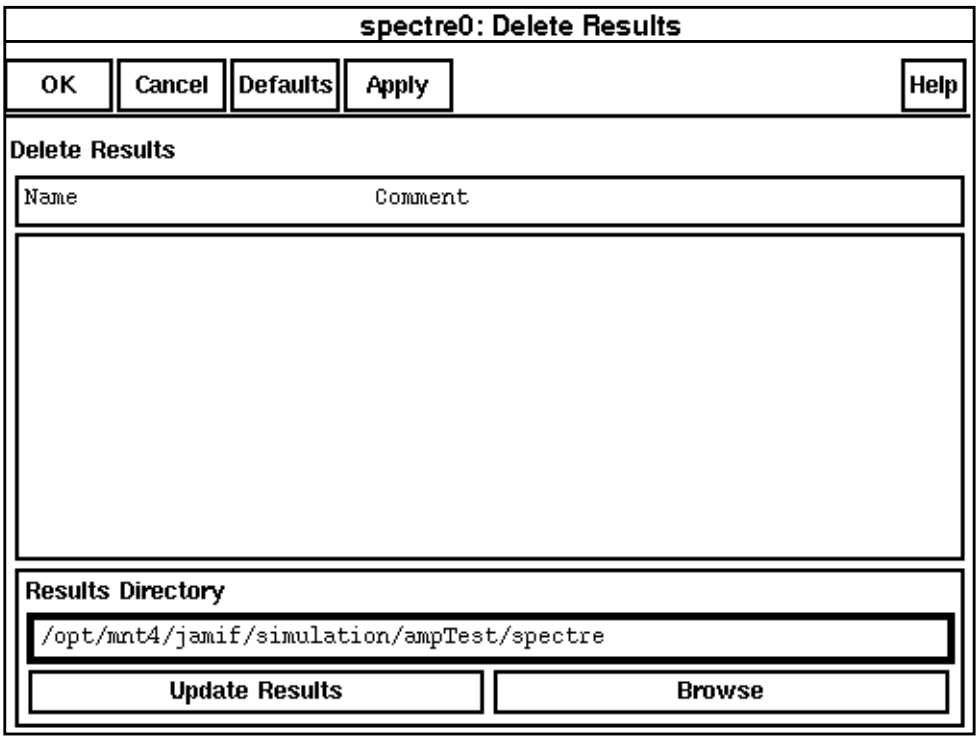
2. Type a new name for the results directory in the *Save As* field.
3. (Optional) Type a short description in the *Comment* field to help you identify these results when you restore the results later.

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.



   The *Results Directory* field lists the default directory in which results are saved.

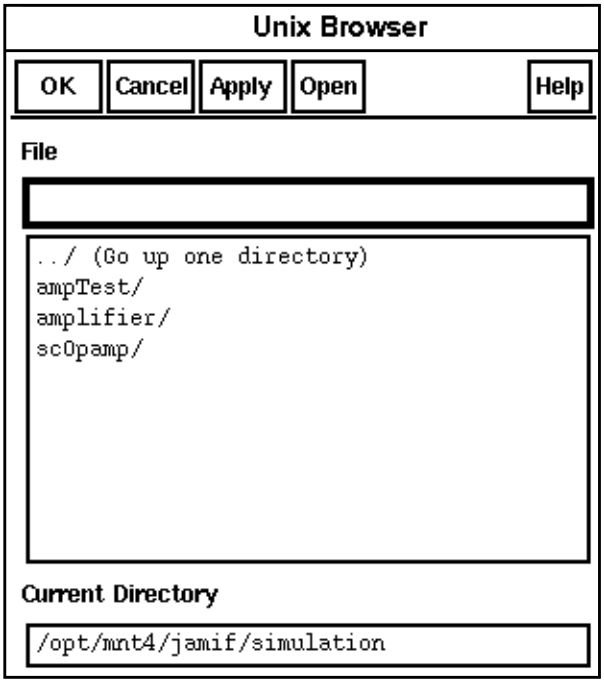
2. Choose the results you want to delete from the list box.

   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.



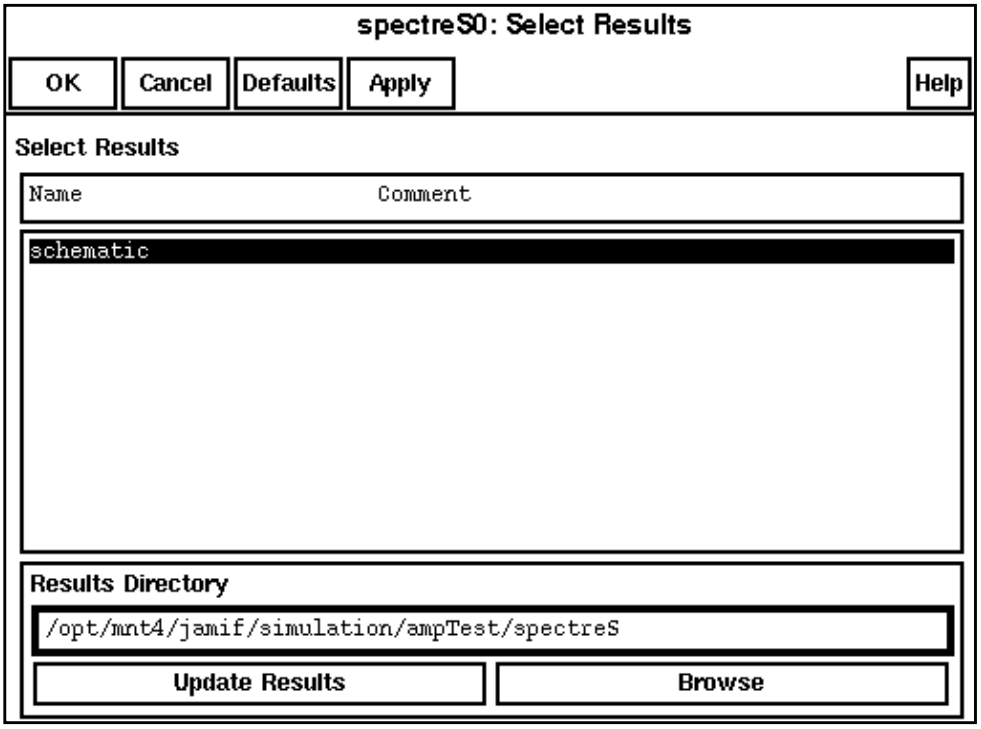
For detailed information about the form, see "UNIX Browser" on page 379.

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.



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*.

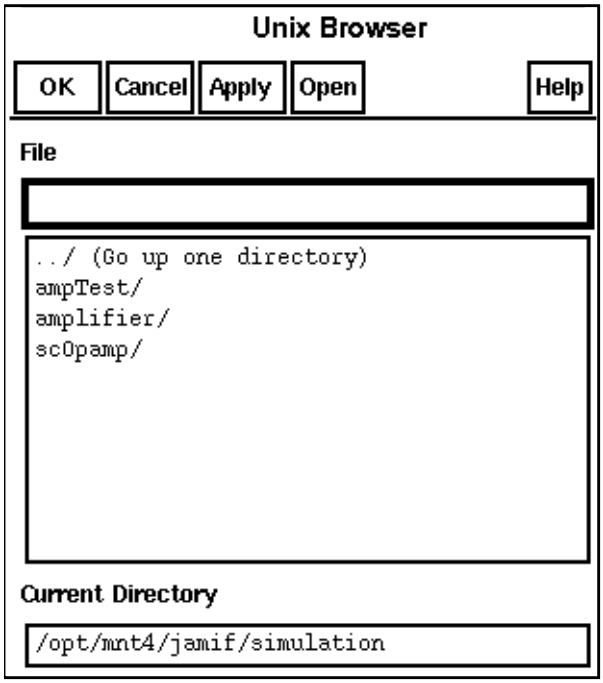
To annotate the schematic,

➤ 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.

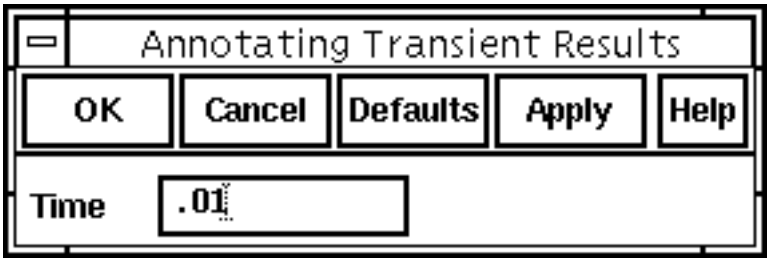


Annotating Transient Voltages

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.



2. Type the transient time point in the *Time* field, and click *OK*.

Annotating Transient Operating Points

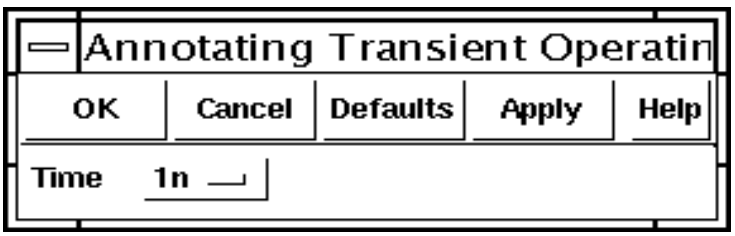
To annotate final transient operating points,

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.



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.

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.

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.

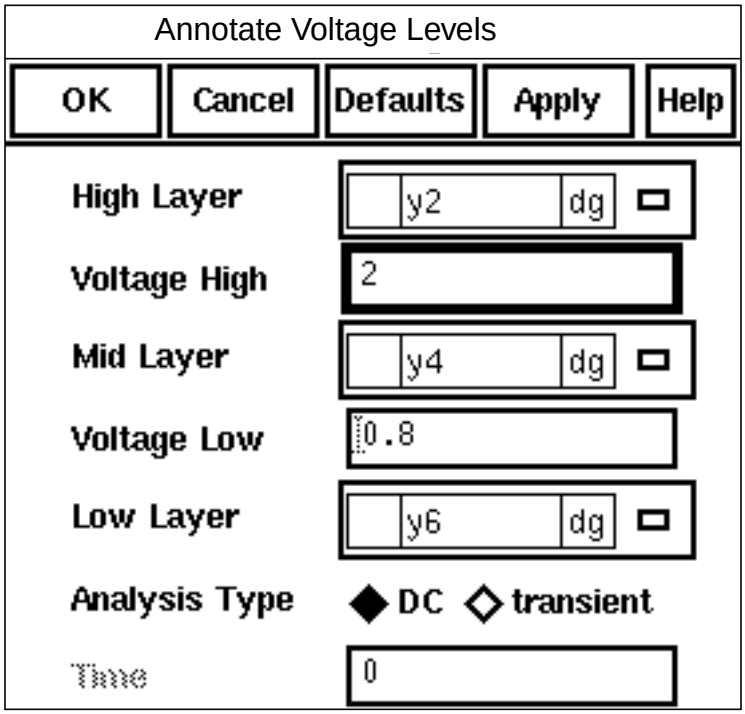
Highlighting Logic Levels with Wire Colors

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.



For detailed information about the form, see “Annotate Voltage Levels” on page 372.

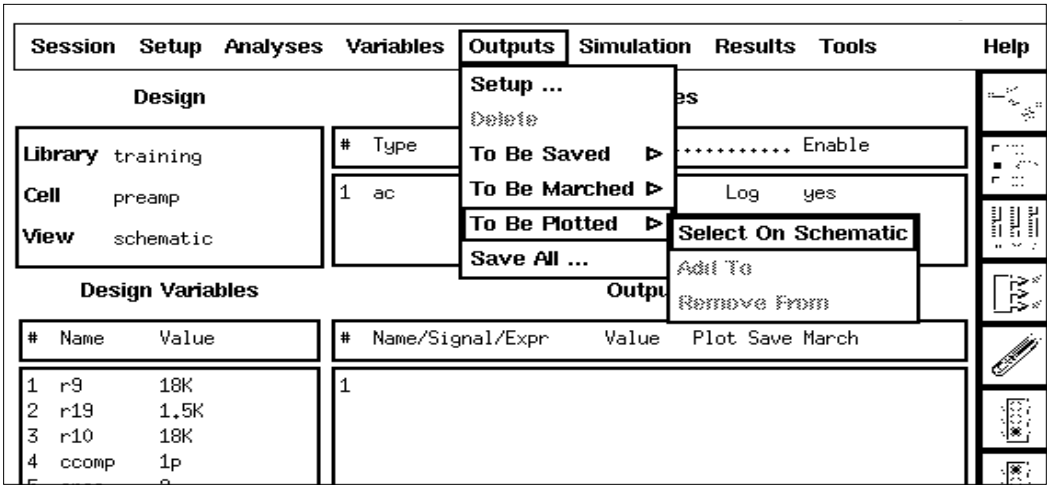
2. Choose layers and colors for the three regions.
3. Set the voltage levels to partition the circuit.

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.

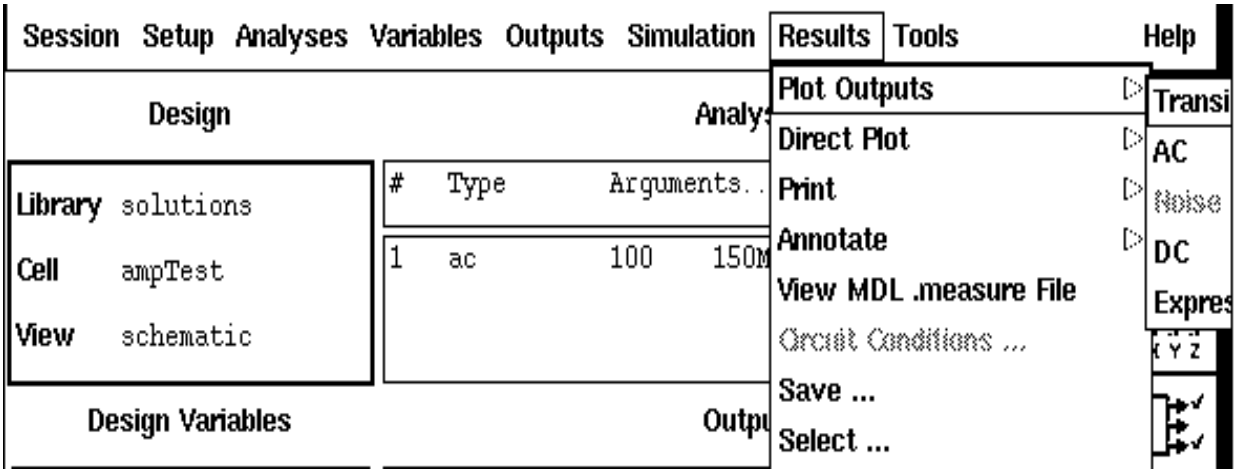


2. In the schematic, choose the outputs you want to display by clicking on them, or hold down the left mouse button and drag a box over the objects you want to choose.

Outputs you select appear in the *Outputs* list in the Simulation window.

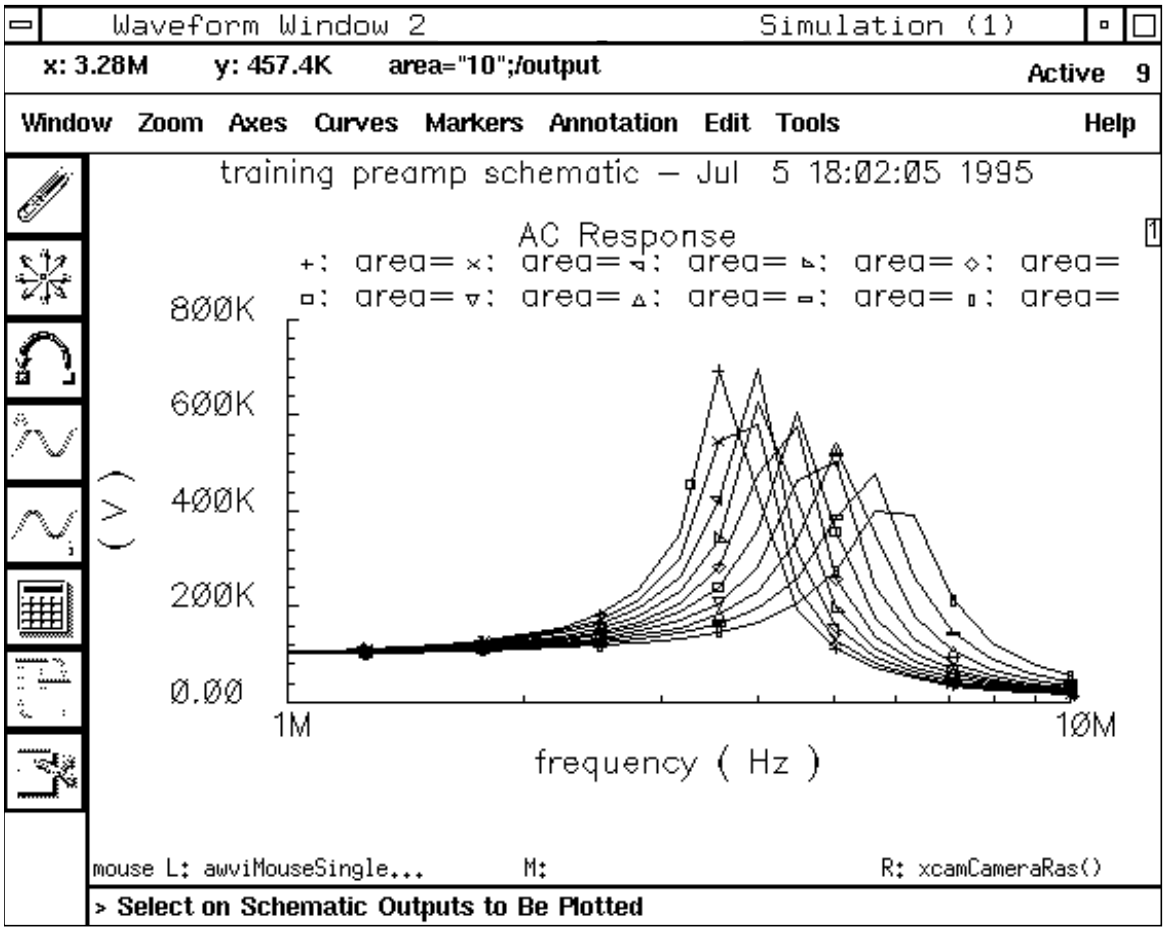
For more information about selecting values in the schematic, see “Overview of Plotting” on page 299.

3. Choose *Results – Plot Outputs* and the name of an analysis in the Simulation window. In this example, the analysis is AC.



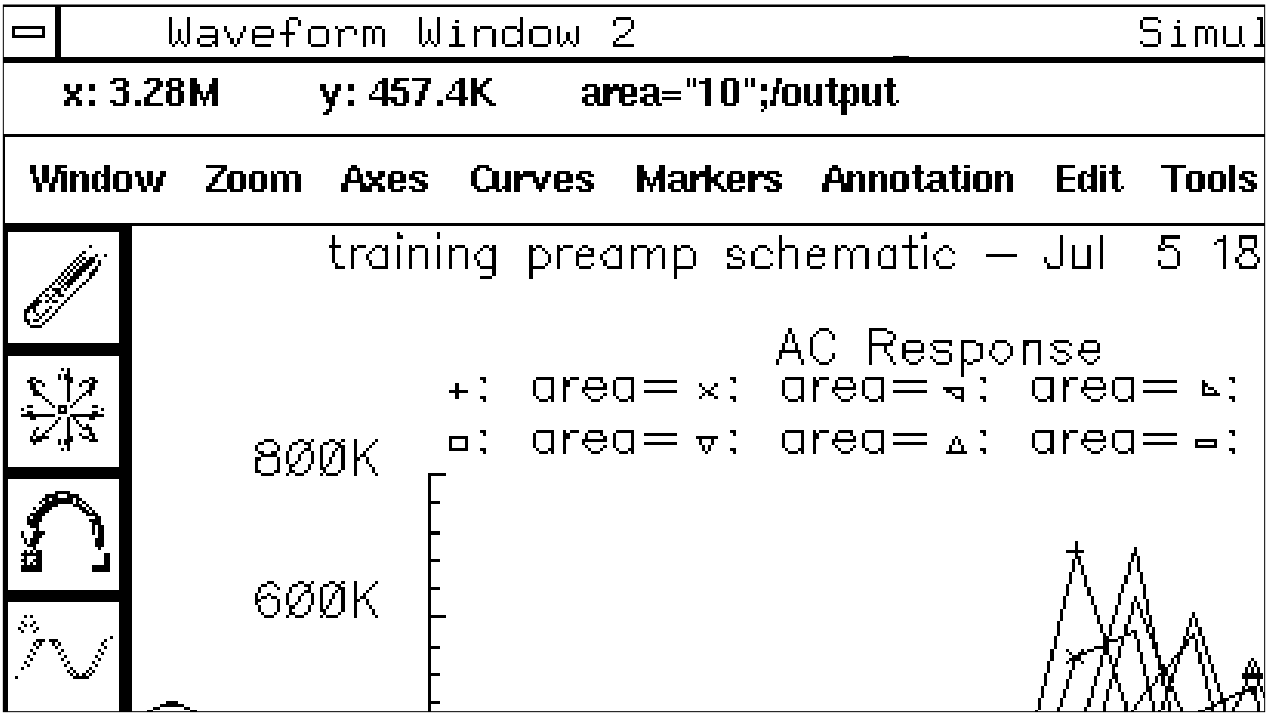
Note: You can choose other plot outputs to plot different types of analysis results.

The Waveform window appears (if it is not already open) and displays the waveforms created by parametric analysis.



You can identify the sweep variable value associated with any curve by placing the cursor on the curve. The sweep variable value appears at the top of the Waveform window. In the

example below, the sweep variable value is named `area`, and the cursor is placed on the curve for `area = 10`.



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.

Overlay Plots erases the previous transient, AC, DC, or noise plot of the current simulation before writing a new plot of the current simulation to the Waveform window. When off, this option adds each new plot to previous plots of the current simulation in the Waveform window.

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

Plot 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. Existing plots are deleted when the new plot is drawn.

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

Plot Mode specifies whether curves you plot are added to (**Append**) or replace (**Replace**) any existing curves in the subwindow.

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.

Direct Plot

11

Hspice Direct Support

Introduction

The *Analog Design Environment* (ADE) now contains a direct integration of the *Hspice* simulator. In previous releases, ADE's *Hspice* integration (*HspiceS*) was 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.

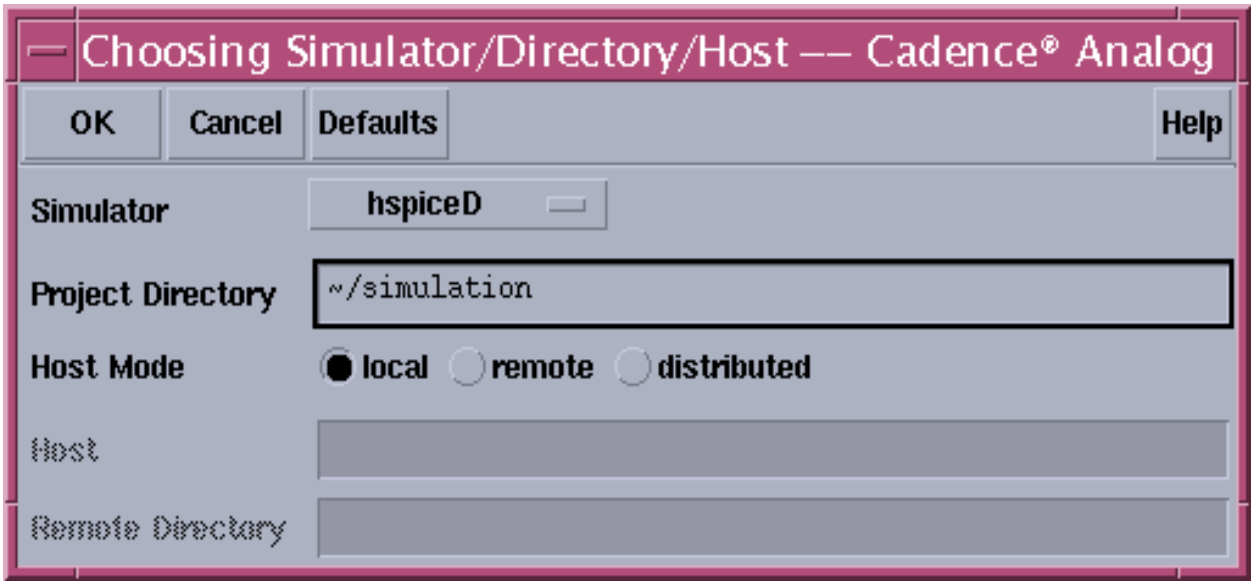
- **Read-only Designs can be Simulated, Provided they are Extracted**

  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**

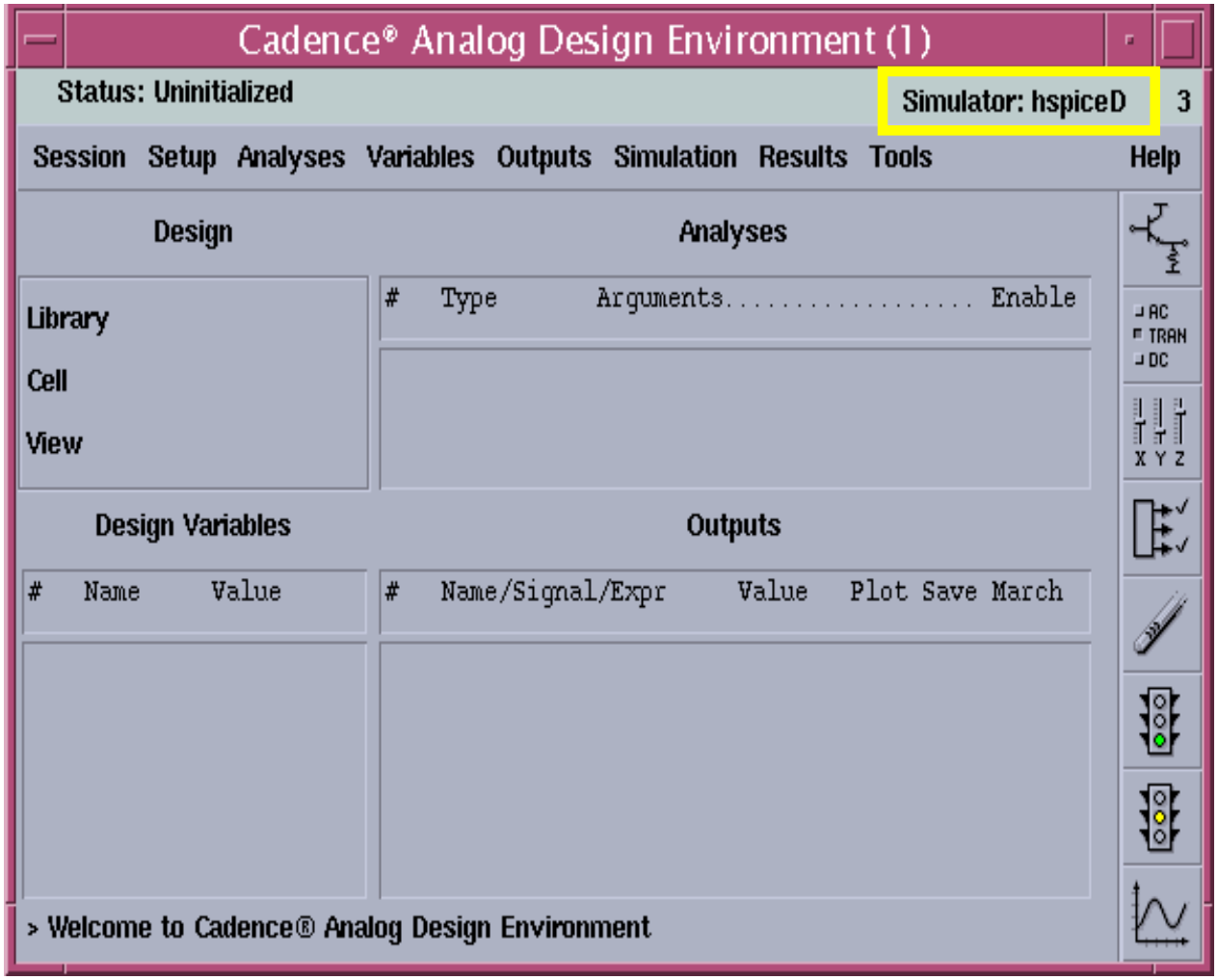
Direct netlisting supports the evaluation of ternary operators (Example, `(iPar("r")>2e-3?200e-3:400e-3`), whereas the same is not supported by socket netlisting.

In order to use the *Hspice Direct* simulator, you need to first select it in the *Choosing Simulator/Directory/Host* form:



For detailed information about the form, refer to the section Choosing Simulator/ Directory/Host.

The *Cadence Analog Design Environment* window displays with the *hspiceD* simulator selected:



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

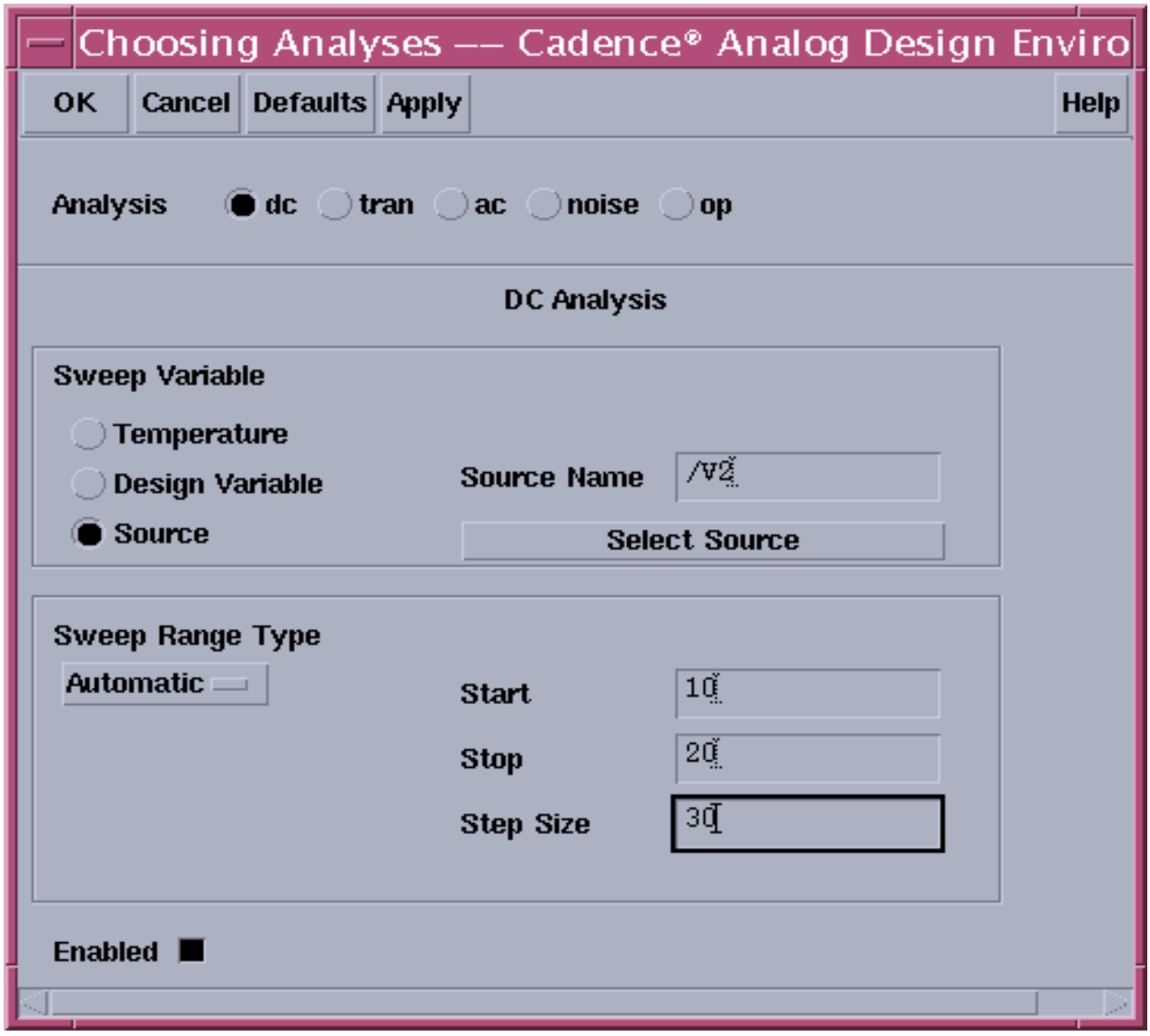
The Distributed Processing mode is supported only for normal simulation and parametric analysis. For detailed information about Distributed Processing, refer to the *Cadence Distributed Processing User Guide*.

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.

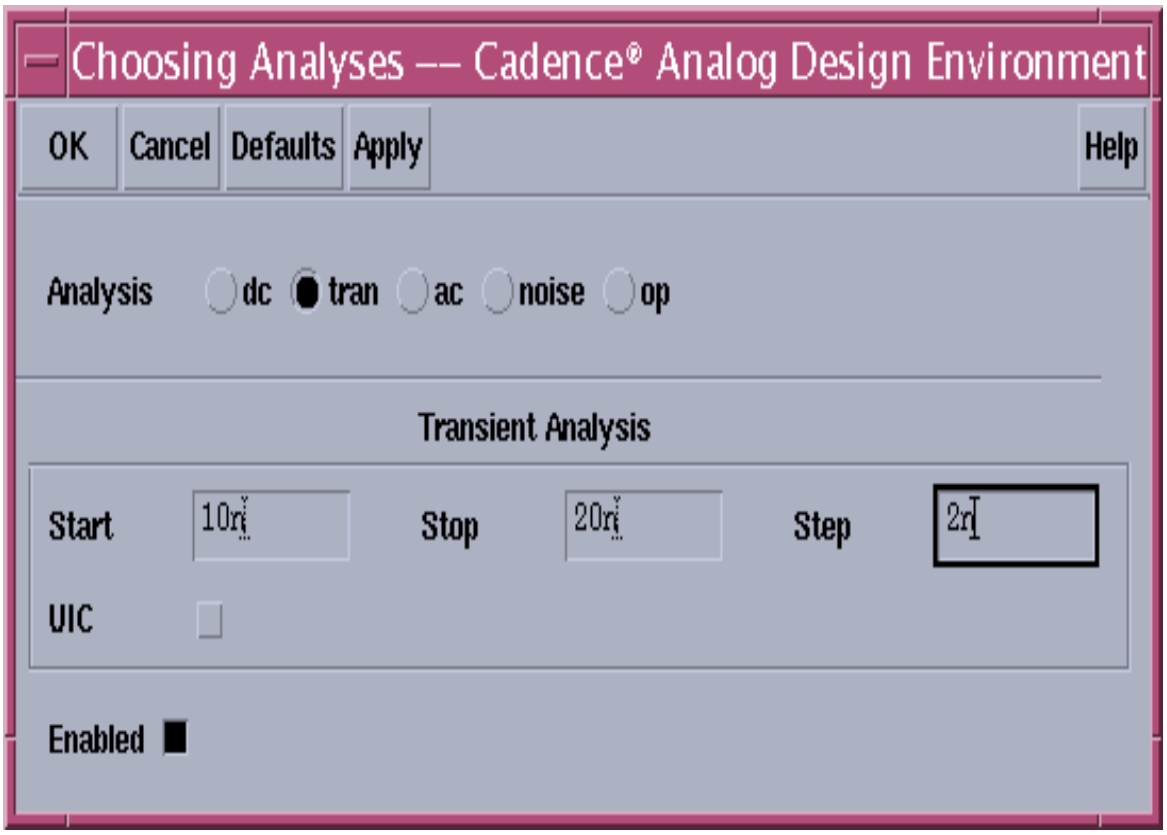
The analyses that are supported are: *DC, Transient, AC, Noise* and *OP*. To run an analysis, select it in the *Choosing Analyses* form. The form re-displays to show the fields that are required for the selected analysis.

- To run a DC analysis, click the *dc* radio button in the *Analysis* section of the *Choosing Analyses* form.

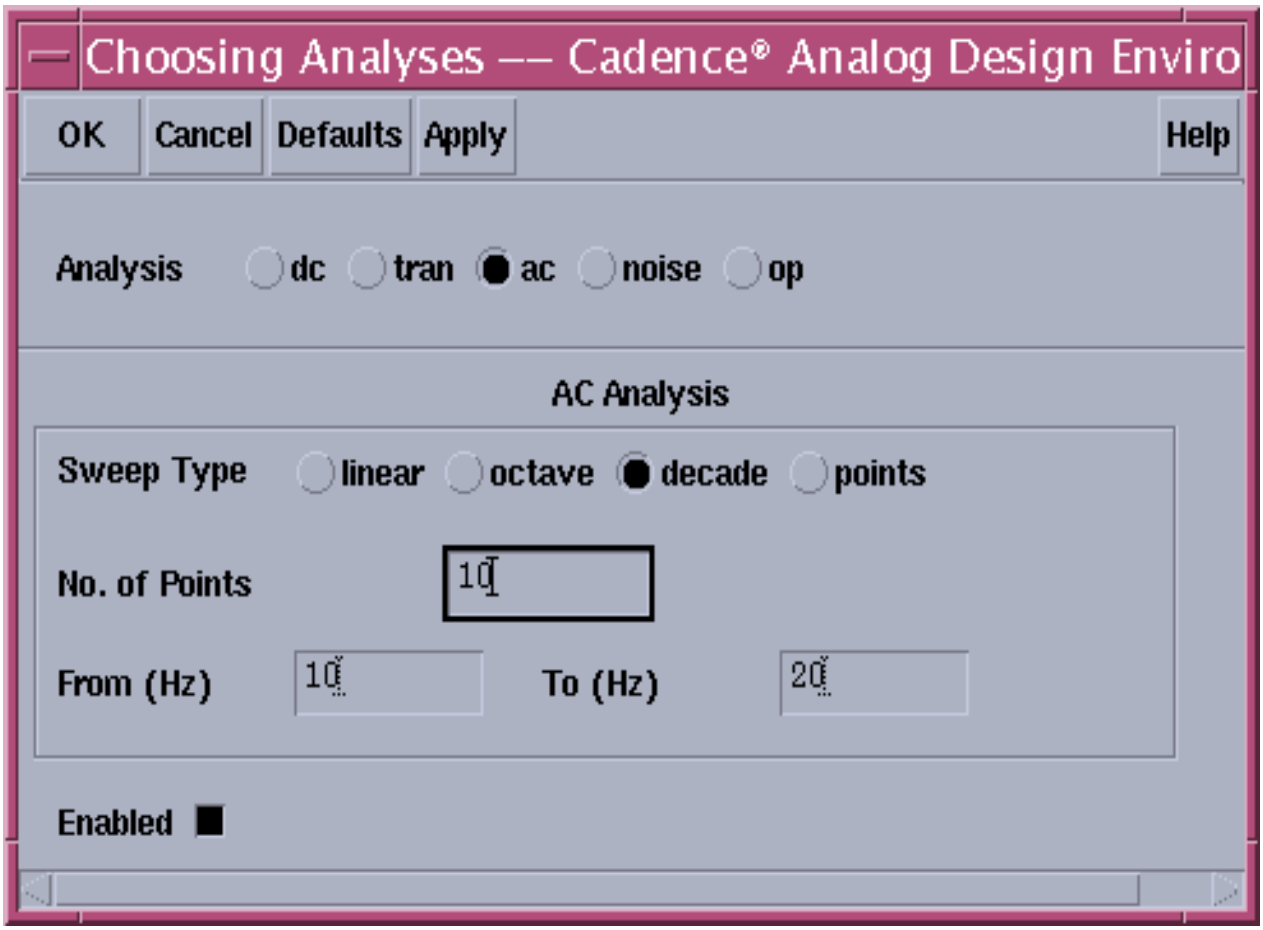


This form reflects the different types of DC sweep variables and sweep range types.

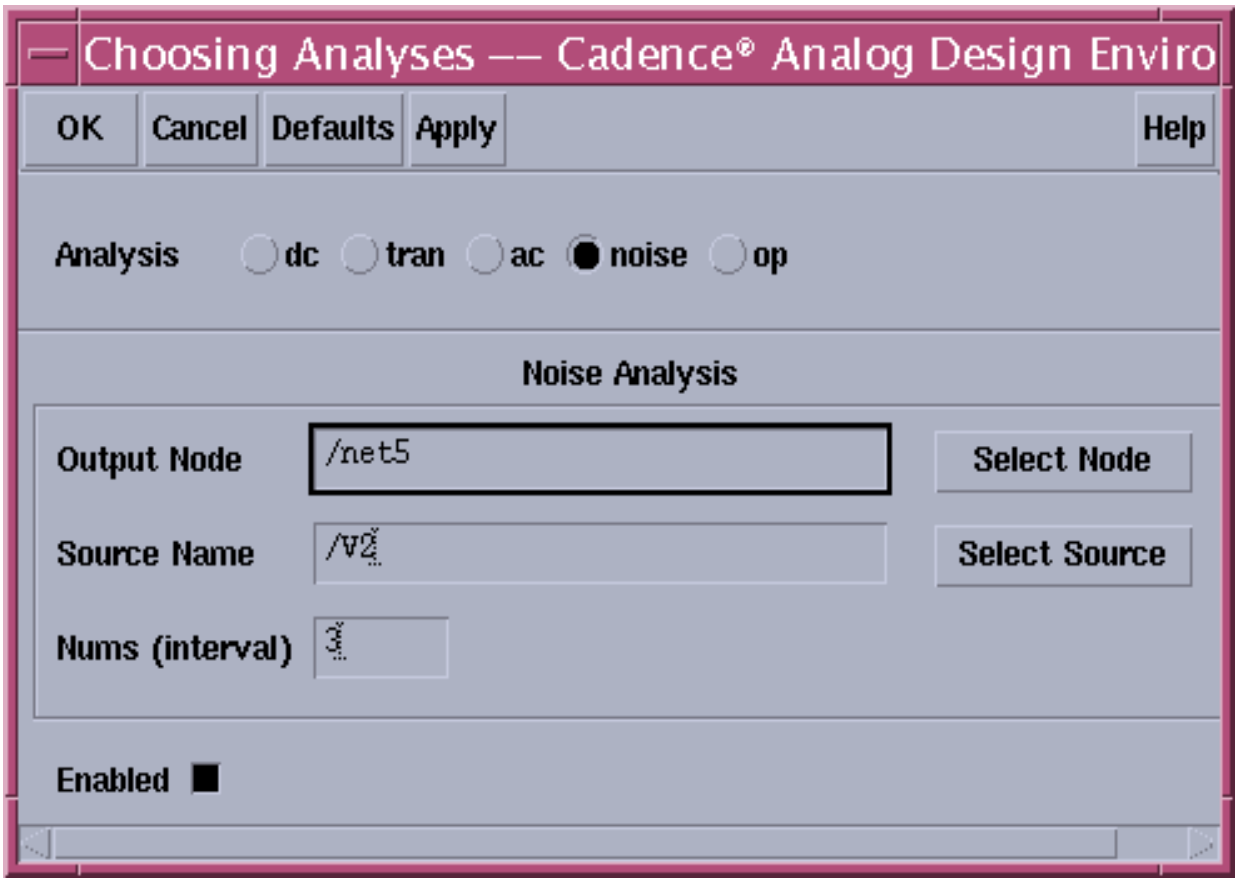
- To run a transient analysis, click the *tran* radio button in the *Analysis* section of the *Choosing Analyses* form.



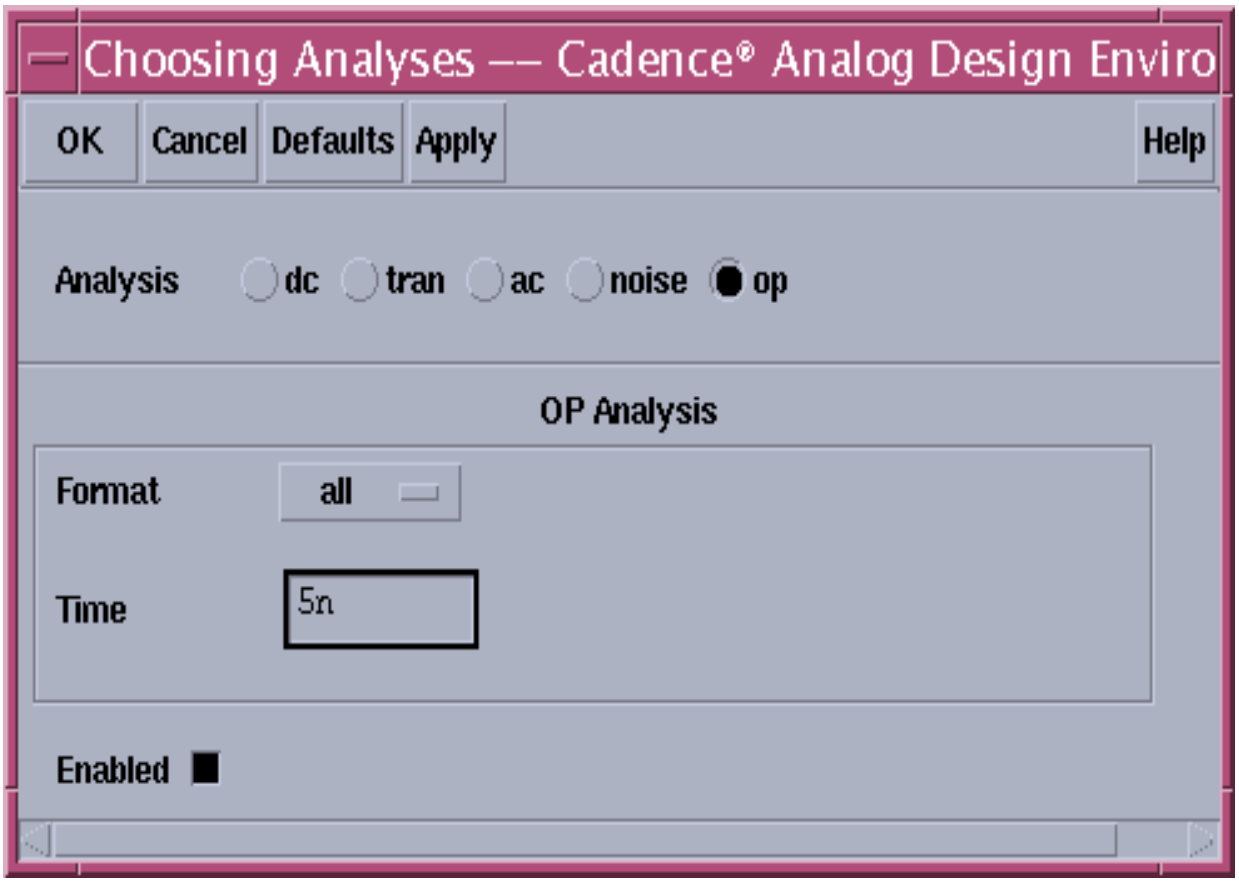
- To run an AC analysis, click the *ac* radio button in the *Analysis* section of the *Choosing Analyses* form.



- To run a noise analysis, click the *noise* radio button in the *Analysis* section of the *Choosing Analyses* form.

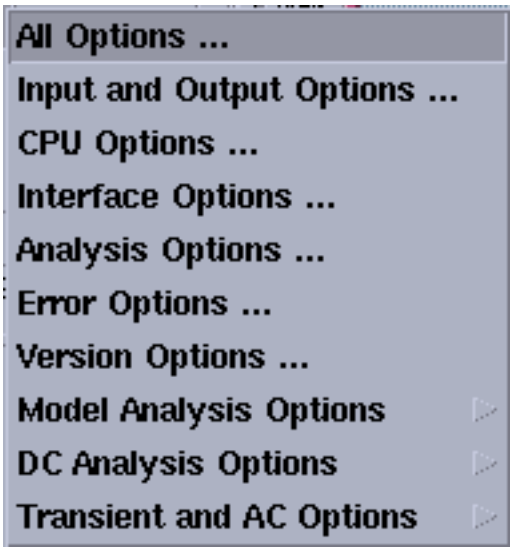


- To run an OP analysis, click the *op* radio button in the *Analysis* section of the *Choosing Analyses* form.



Analog Options

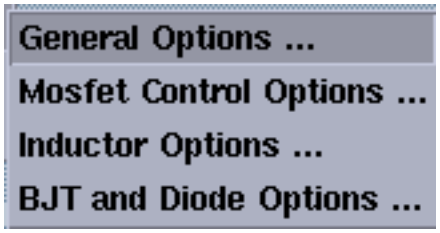
You can specify appropriate Hspice simulator options using *Simulation-> Analog Options:*



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:

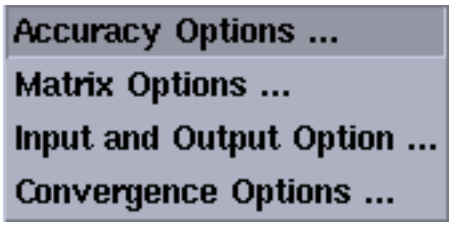


- Click *Model Analysis Options -> General Options* to specify DCAP, HIER_SCALE, MODMONTE, MODSRH, SCALE, TNOM using the *Hspice General Model Options* form.
- Click *Model Analysis Options -> Mosfet Control Options* to specify CVTOL, DEFAD, DEFAS, DEFEL, DEFNRD, DEFNRS, DEFPD, DEFPS, DEFW, SCALM, WL 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.

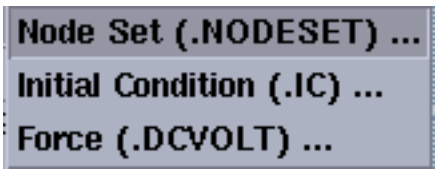
Important

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



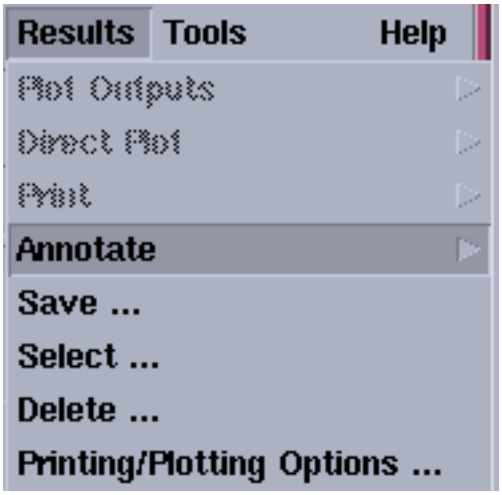
Click *Convergence Aids-> Node Set (.NODESET)* to initialize specified nodal voltages for a DC operating point analysis. The `.NODESET` statement is generally used to correct convergence problems in a DC analysis. Setting nodes in the circuit to values that are close to the actual DC operating point 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. Please use the menu *Simulation -> Output Log* to view the simulator output file.

Advanced Analysis Tools Support

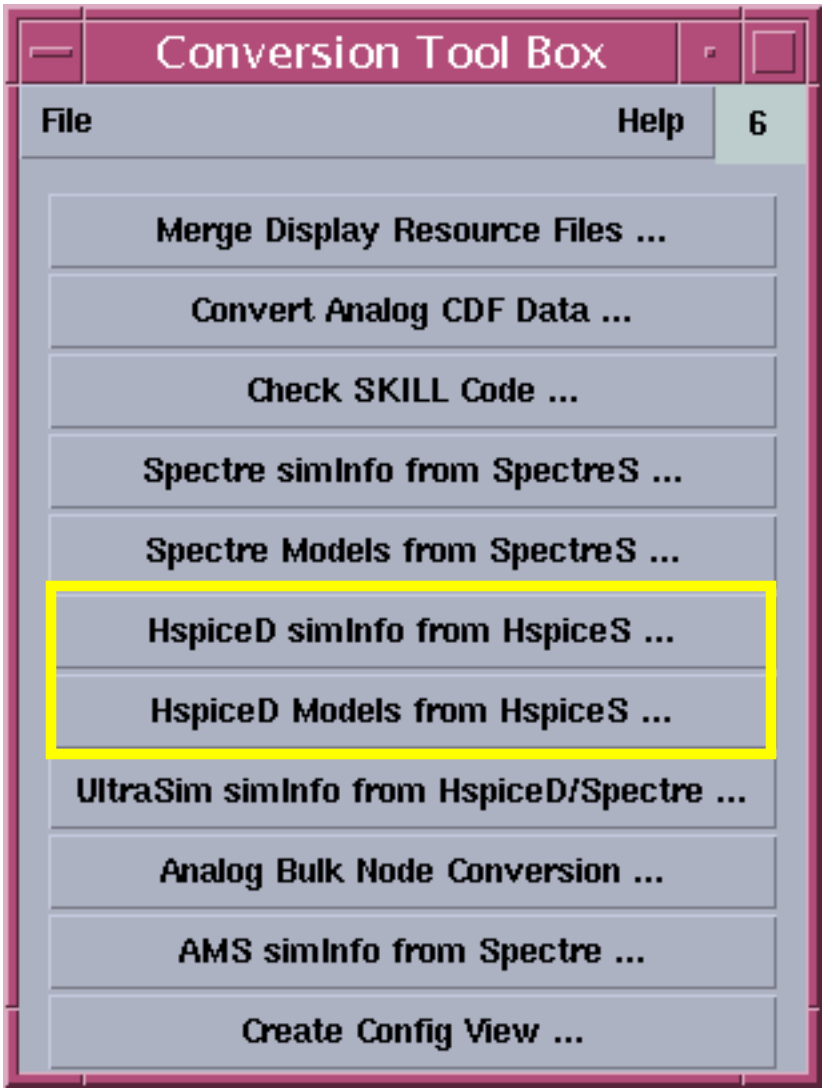
The tools that are supported are *Parametric*, *Corners and Optimization. Monte Carlo Circuit Conditions* are not supported in this release as this requires native support from the simulator.

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*

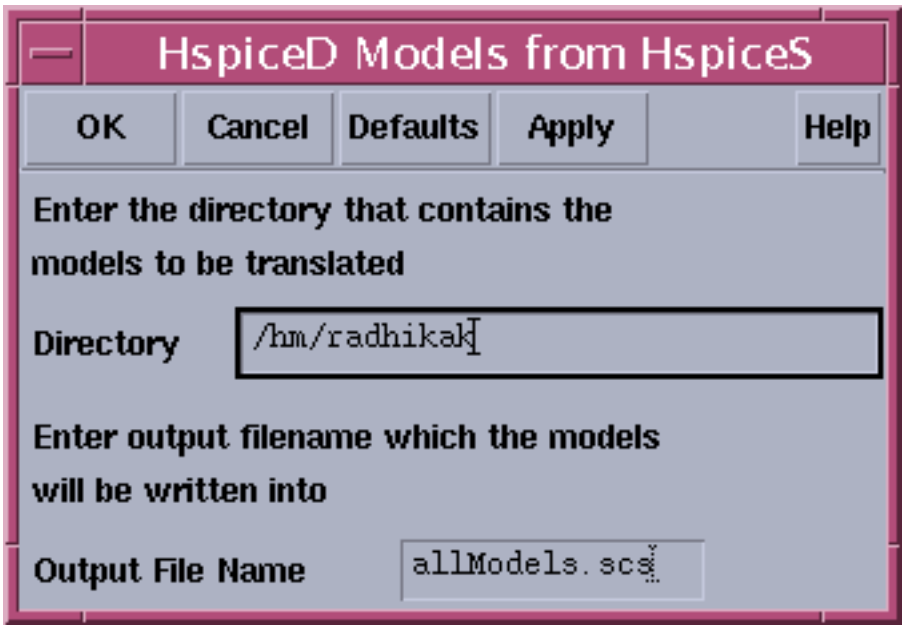
Tool Box window, click *Tools->Conversion Tool Box* in the CIW (Command Interpreter Window).The window displays:

.



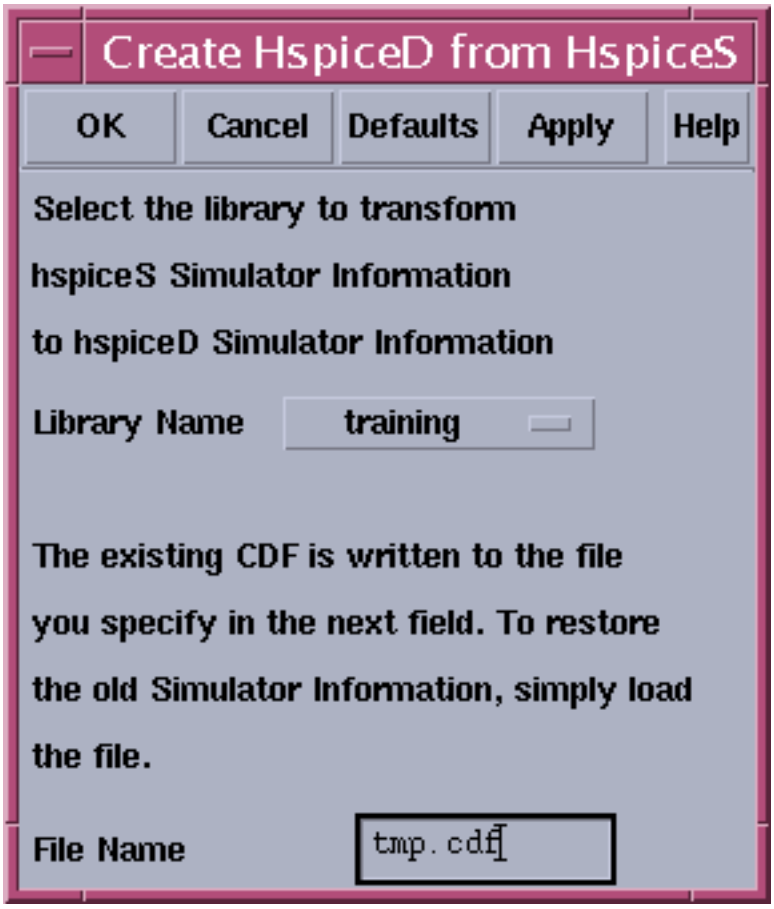
For detailed conversion information, see the section, *Preparing Existing Designs for Direct Hspice Support* of the *Compatibility Guide*

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.



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.



For detailed steps on converting libraries to use *Hspice Direct* support, refer to the section *Preparing Existing Designs for Direct Hspice Support* in the *Compatibility Guide*

A

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 Cadence® design framework II (DFII).

This appendix covers the following topics:

What Is auCdl and Why Do You Need It?

To compare a layout vs. 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).

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 423.

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 405.

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.
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.

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.

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:

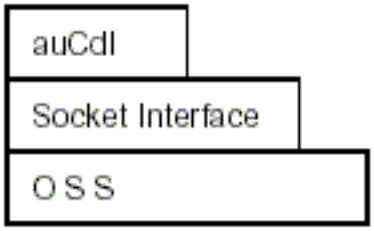
auCdlPrintBlkBox=‘t	Treats the macro as a black box
auCdlPrintBlkBox=‘nil	Treats the macro as is

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.

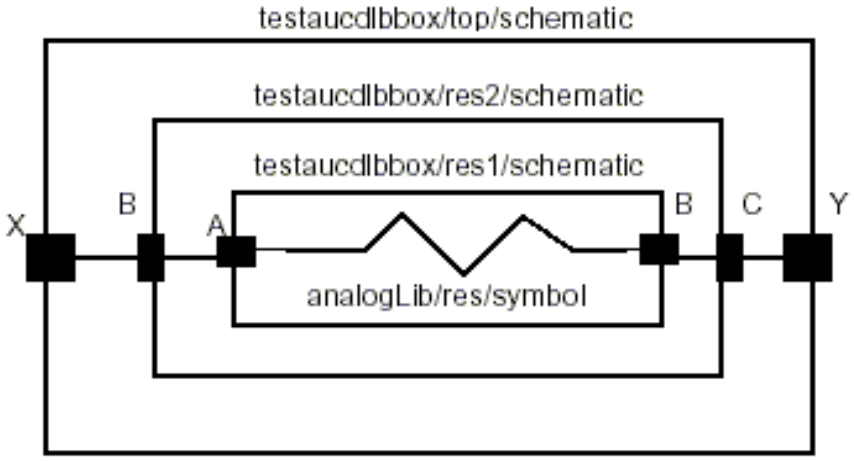
The following figure describes the location of auCdl in DFII with regard to OSS and Socket Interface.



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



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.

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.

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 behaviour 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 418. 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 418 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.

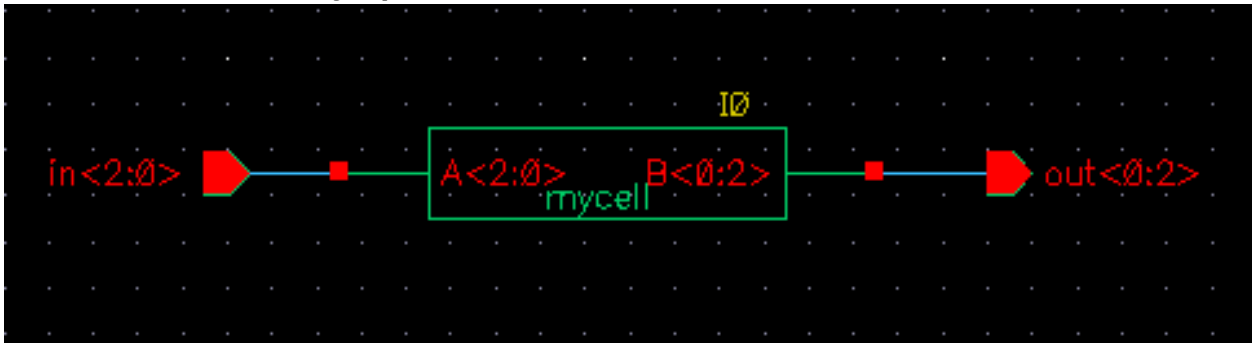
Note: If termOrder is empty, the default terminal list is used.

Note: Set the shell variable `CDS_Netlisting_Mode` to Analog for auCdl netlisting.

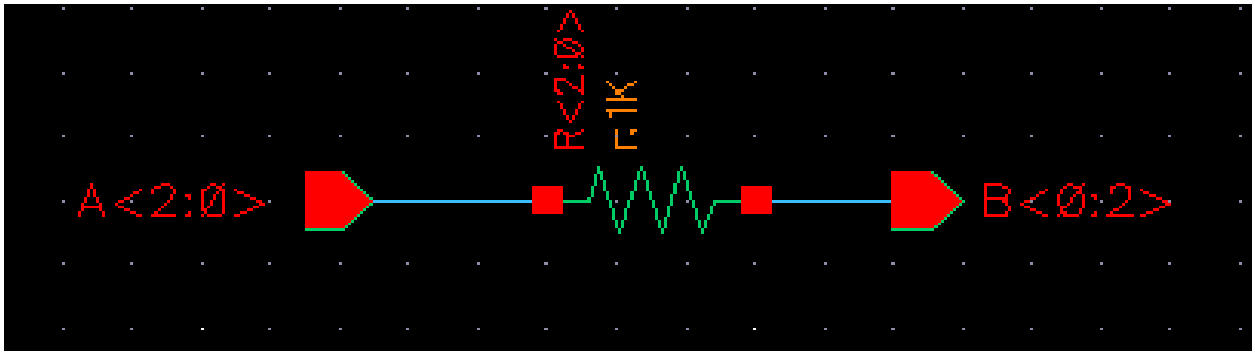
Example

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

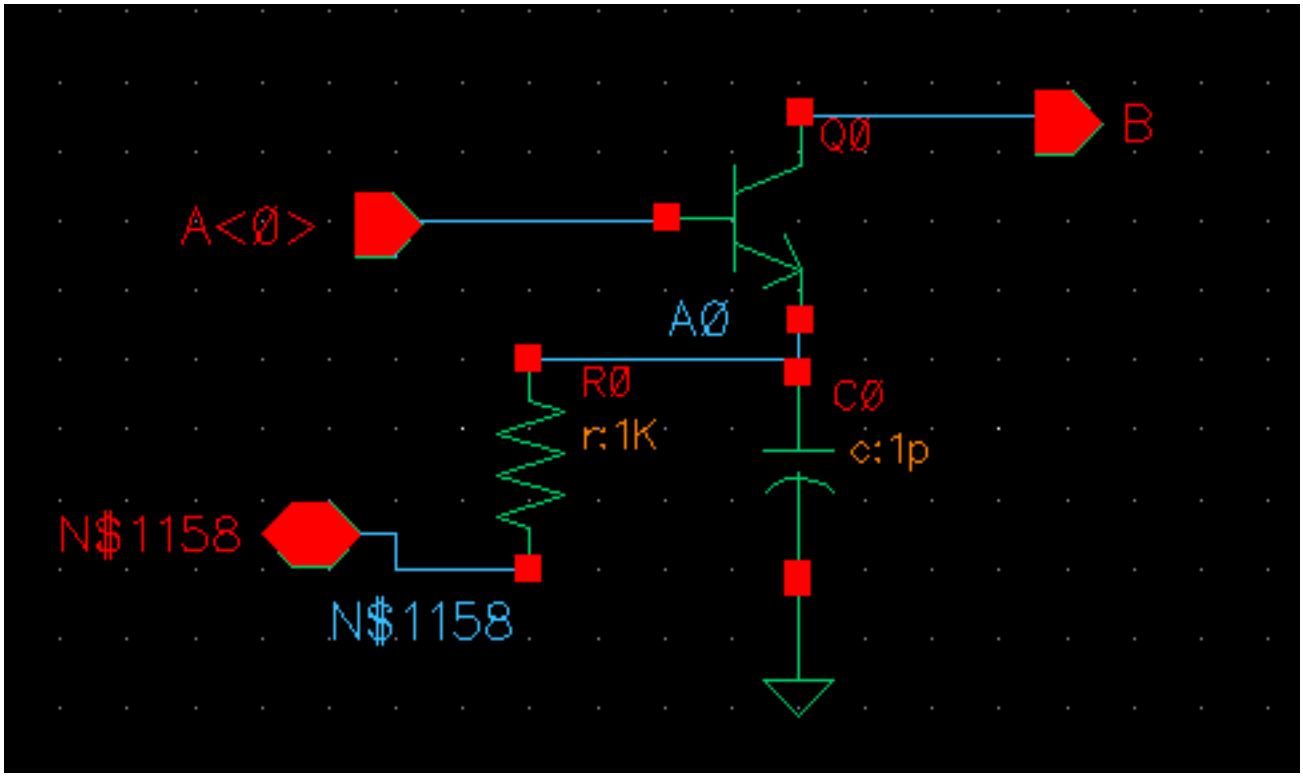
Sometimes a net name may get mapped to a new name, such as when there are invalid characters in the original name. This new name may collide with another existing or mapped net name. Due to this collision, one of the net names is mapped to a new name.

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.
`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
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
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

BJT	
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
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

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
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.

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*.

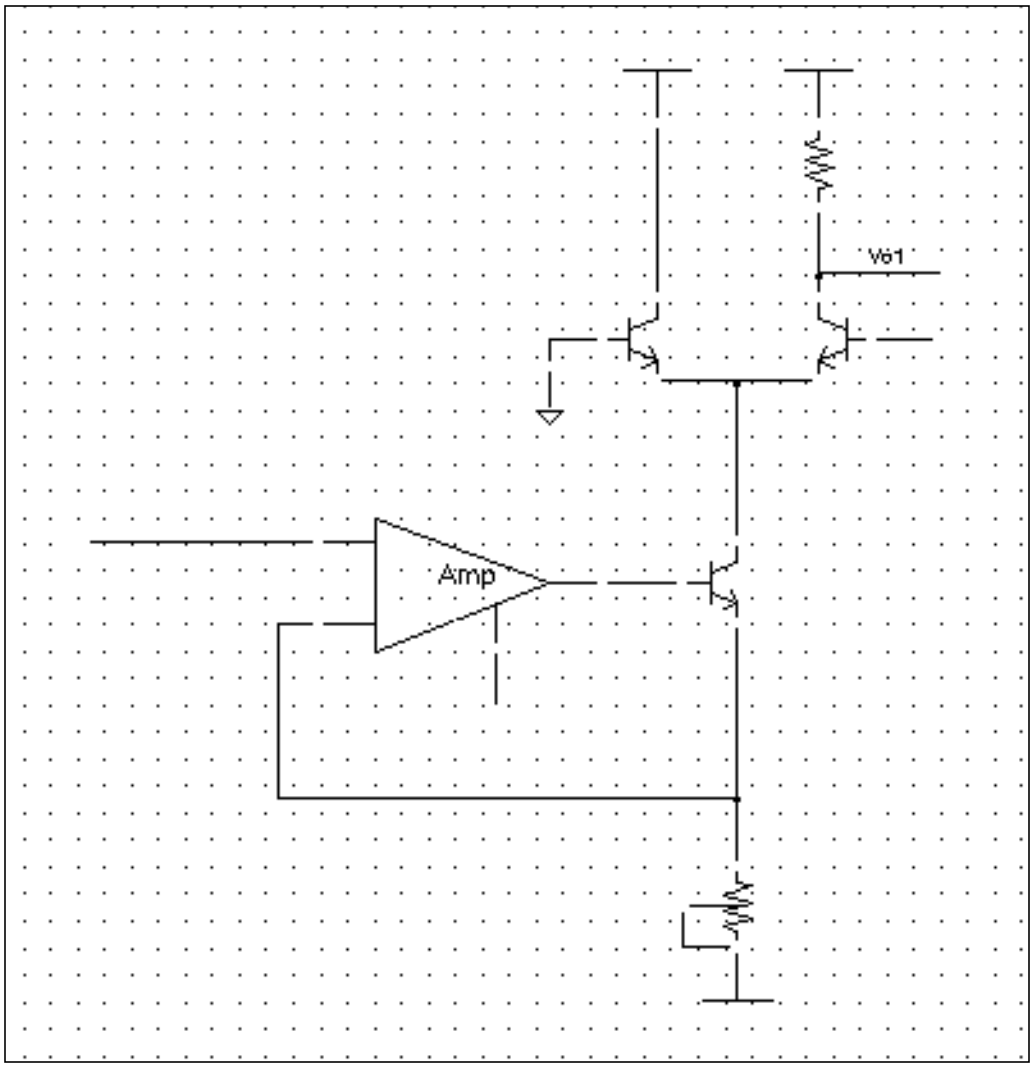
For more information about using CDL Out, read the *Translating CDL Files* section in the *Design Data Translators Reference*.

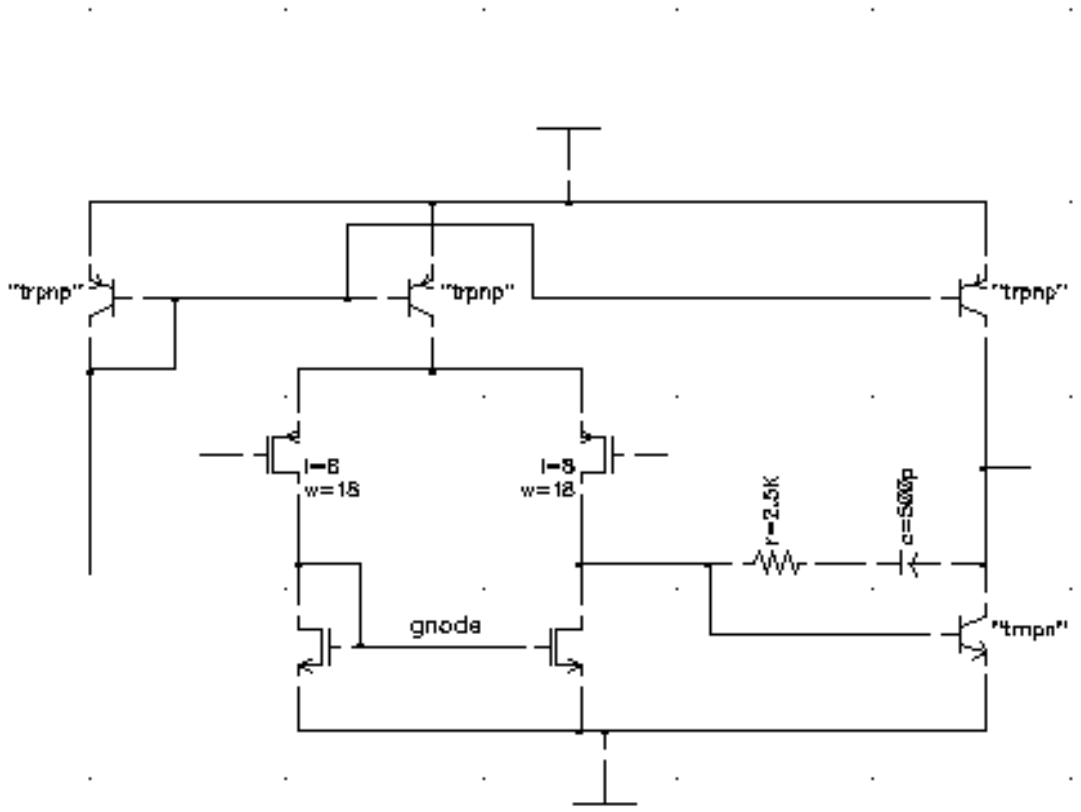
The following `si.env` parameters are used in the 4.3.x release only.

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
```

B

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:

- Calculator
- Distributed Processing
- Spectre
- ADE Simulation Environment

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*.

Variable Type	cyclic
Default Value	RPN
Acceptable Values	{RPN, algebraic}
Window-Menu	*Calculator -> Options*

uimode

This variable sets the mode of operation for calculator. For details, refer to the *About Standard and RF Modes* section, of Chapter 1 of the *Waveform Calculator User Guide*.

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*.

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*.

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	*Calculator*

Distributed Processing

autoJobSubmit

If this variable is set to a non-nil value, the *Job Setup* form is not displayed at job submit time.

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.

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
Window-Menu	none

queueName

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 *Cadence® Analog Distributed Processing Option User Guide*.

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	*Job Submit Form*

hostName

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 *Cadence® Analog Distributed Processing Option User Guide*.

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	*Job Submit Form*

startTime

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 *Cadence® Analog Distributed Processing Option User Guide*.

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
Window-Menu	*Job Submit Form*

startDay

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.

Variable Type	cyclic
Default Value	`today`
Acceptable Values	`{today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday }`
Window-Menu	*Job Submit Form*

expTime

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 *Cadence® Analog Distributed Processing Option User Guide*.

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.

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

expDay

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 *Cadence® Analog Distributed Processing Option User Guide*.

Variable Type	cyclic
Default Value	`today`
Acceptable Values	`{today, Sunday, Monday, Tuesday Wednesday, Thursday, Friday, Saturday }`
Window-Menu	*Job Submit Form*

timeLimit

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 *Cadence® Analog Distributed Processing Option User Guide*.

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 *Cadence® Analog Distributed Processing Option User Guide*.

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
Window-Menu	*Job Submit Form*

mailTo

Sets the default list of users who will recieve 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 *Cadence® Analog Distributed Processing Option User Guide*.

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 email.

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
Window-Menu	none

stateFile

Sets the filename containing the job server's state.

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.

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.

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.

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.

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.

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

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.

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
Window-Menu	none

puttogetherqueue

Specifys the queue to be used for the Put Together Job.

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
Window-Menu	none

copyNetlist

Specifys 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.

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`) .

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}

Spectre

save

This variable selects signals to be saved.

Variable Type	string
Default Value	allpub
Acceptable Values	none,selected,lvlpub,allpub,all
Window-Menu	*Cadence Analog Design Environment* ->Save Options *-> Select signals to output (save)*

outputParamInfo

This variable sets/reset the *Save output Parameters Info* option.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment* ->Save Options *->Save output parameters info*

modelParamInfo

This variable sets/resets the *Save model parameters Info* option.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment* ->Save Options *-> Save model parameters info*

pwr

This variable is used to select the power signals to output.

Variable Type	string

Default Value	""
Acceptable Values	none, total, devices, subckts, all
Window-Menu	*Cadence Analog Design Environment* -> >Save Options -> *Select power signals to output (pwr*)

useprobes

This variable is used to set the *Select AC terminal currents (useprobes)* option.

Variable Type	string
Default Value	""
Acceptable Values	yes, no
Window-Menu	*Cadence Analog Design Environment* -> Save Options -> *Select AC terminal currents (useprobes)*

subcktprobelvl

This 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	*Cadence Analog Design Environment* -> Save Options -> *Set subcircuit probe level (subcktprobelvl)*

nestlvl

This 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.

Variable Type	string
Default Value	""
Acceptable Values	--
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence 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
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	selected, all, nonlinear
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	"spectre cmos_sch cmos.sch schematic veriloga ahdl"
Acceptable Values	view names, separated by spaces.
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	"spectre"
Acceptable Values	view names separated with spaces.
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment ->* Environment Options *-> Automatic output log*

spp

This variable is used to set/reset the *Use SPICE Netlist Reader(spp)* option.

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Cadence Analog Design Environment ->* Environment Options *-> Use SPICE Netlist Reader(spp)*

stimulusFile

This variable is used to set the path for stimulus file.

Variable Type	string
Default Value	""
Acceptable Values	unix path
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	unix directories, separated with spaces.
Window-Menu	*Cadence Analog Design Environment* -> Simulation Files Setup -> *include Path*

modelFiles

Use this field for adding the default model files.

Variable Type	string
Default Value	""
Acceptable Values	list of paths to model files.
Window-Menu	*Cadence Analog Design Environment* -> Model Library Setup

analysisOrder

Determines the order in which the analyses would be run by the simulator.

Variable Type	string
Default Value	""
Acceptable Values	Names of analysis in the order desired
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	path of the file
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	unix path or name of one or more files.
Window-Menu	*Cadence 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.

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.

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.

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.

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.

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.

Variable Type	string
Default Value	spectre
Acceptable Values	path of the spectre executable
Window-Menu	--

checkpoint

Y runs spectre with the `+checkpoint` option, which turns on the checkpoint capability. N runs spectre with the `-checkpoint` option, which turns off the checkpoint capability.

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Cadence Analog Design Environment ->* Environment Options *-> Create Checkpoint File(cp)*

recover

Y runs spectre with the `+recover` option, which restarts the simulation from the checkpoint file, if it exists. N runs spectre with the `-recover` option, which does not restart the simulation, even if a checkpoint file exists.

Variable Type	string
Default Value	""
Acceptable Values	Y, N
Window-Menu	*Cadence Analog Design Environment ->* Environment Options *-> Start from Checkpoint File(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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	--

simOutputFormat

Use this variable to specify the format of the output results. If you specify values other than

those supported by Artist, Spectre generates an error.The `psfbinf` format is a single-precision format that uses only half the disk space that `psfbin` uses.

Variable Type	string
Default Value	psfbin
Acceptable Values	psfbin, psfbinf
Window-Menu	--

Note: To set this variable in `.cdsinit`, add the following:
`envSetVal("spectre.envOpts" "simOutputFormat" 'string "psfbinf")`
To set it in `.cdsenv`, add:
`spectre.envOpts simOutputFormat string "psfbinf"`

controlMode

Used to run Spectre in batch or interactive modes depending on the value of the variable.

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

ADE Simulation Environment

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.

Variable Type	string
Default Value	~/.artist_states
Acceptable Values	dir path
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	nil

Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Editing Session Options -> Schematic Menus*

windowBased

This variable lets you choose the way the Cadence® analog design software starts up your session. If it is true, open the Simulation window. Else do not open the simulation window.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Editing Session Options ->Query to Save State*

loadCorners

If this option is true, it preloads the Corners Java.

Variable Type	boolean

Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence 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.

Variable Type	int
Default Value	1
Acceptable Values	Any number between 0 and 1200
Window-Menu	*Cadence 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.

Variable Type	int
Default Value	317
Acceptable Values	Any number between 0 and 1000
Window-Menu	*Cadence Analog Design Environment -> Editing Session Options - >Window Y Location*

simulator

Lets you specify the default simulator for the analog design environment.

Variable Type	string
Default Value	spectre
Acceptable Values	simulator name
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	"~/simulation"
Acceptable Values	directory path
Window-Menu	*Cadence Analog Design Environment -> Choosing Simulator/Directory/Host-> Project Directory*

hostMode

Lets you specify a default local, remote or distributed simulation.

Variable Type	string
Default Value	local
Acceptable Values	local, remote, distributed
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
Window-Menu	*Cadence Analog Design Environment -> Choosing Simulator/Directory/Host -> Host*

digitalHostMode

The variable digitalHostMode lets you choose default local or remote digital simulation.

Variable Type	string
Default Value	"local"
Acceptable Values	local, remote
Window-Menu	*Cadence 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.

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
Window-Menu	*Cadence Analog Design Environment -> Choosing Simulator/Directory/Host-> Digital Host*

remoteDir

Lets you specify a path to the run directory for remote simulation. The remore directory name should be same as the local simulation directory name. You must specify a complete path.

Variable Type	string
Default Value	""
Acceptable Values	Unix path
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> Auto Plot Outputs After Simulation*

overlay

When nil, erases the previous plots in the waveform window before writing a new plot of the current simulation to the Waveform window. When t, this option adds each new plot to previous plots of the current simulation in the Waveform window.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options-> Overlay Plots*

designName

If true, displays the design name in the Waveform window.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options-> Design Name*

simulationDate

If true, displays the simulation run date in the Waveform window.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options-> Simulation Date*

temperature

If true, displays the temperature associated with the plotted results in the Waveform window.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> temperature*

variables

if true, displays the names and values of design variables in the Waveform window.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> Design Variables*

scalarOutputs

If true, displays simulation results that evaluate to scalar values in the Waveform window

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> Scalar Outputs*

icons

This variable places icons in the Waveform window.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options ->Allow Icons*

width

Specifies the width of the waveform window.

Variable Type	int
Default Value	564
Acceptable Values	Between 200 to 1200
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> Width*

height

Specifies the height of the waveform window.

Variable Type	int
Default Value	428
Acceptable Values	Between 200 and 1000
Window-Menu	*Cadence Analog Design Environment -> Setting Plotting Options -> Height*

x

Enables you to set the horizontal position of the left side of the waveform window.

Variable Type	int
Default Value	577
Acceptable Values	Between 0 and 1200.
Window-Menu	*Cadence Analog Design Environment - Setting Plotting Options - X Location*

y

Enables you to set the vertical position of the top of the Waveform window.

Variable Type	int
Default Value	373
Acceptable Values	Between 0 and 1000.
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	*Cadence 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.

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

Window-Menu	*Cadence Analog Design Environment-> Setting Plotting Options -> Print After*

preSaveOceanScript

This procedure would be executed before 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:

```
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`.

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`.

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/`

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`.

Variable Type	boolean
Default Value	nil
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 is done, a dialog box will come up that will ask if it is OK to update the base cell CDF termOrder data. The dialog box will display the simulators whose termOrder it will update, and the new termOrder that will be set for each simulator.

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
Window-Menu	--

printNotation

It is used to specify how numbers are printed in the Artist environment. This applies only to *Results->Print* and *print/printvs* in the Calculator. Numbersare printed in the notation this variable is set to.

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.

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 seperate strip. `auto` sets the strip mode type to `analogStrip` if the simulator is analog-only, and `analogComposite` if it is a mixed-signal type.

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.

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 Whats New window. For example, set this variable to `5.0.0` if you do not want to see the Whats New window for 5.0.0.

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.

Variable Type	int
Default Value	1
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
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.

Variable Type	string
Default Value	#
Acceptable Values	Any integer. The maximum limit is the limit of an integer
Window-Menu	--

updateCDFtermOrder

If set to `t`, Artist will automatically update the CDF termOrder when symbol changes that affect the terminal order are made. This will display additional dialog boxes asking you to to accept or reject the change to the CDF termOrder.

Variable Type	boolen
Default Value	nil
Acceptable Values	t, nil

loadCorners

If set to `t`, Corners classes are preloaded when Artist is started.

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.

Variable Type	string
Default Value	string "spectre spectreS cdsSpice auCdl auLvs hspiceS"
Acceptable Values	spectre spectreS cdsSpice auCdl auLvs hspiceS

defaultTools

Set this variable to the list of simulators that need to be selected by default in the *toolFilter* form.

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.

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`.

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

Index

Symbols

A

B

C

D

E

F

G

H

I

R

S

T

U

V

W

X