Contents

Preface ......................................................................................................................... 7
Related Documents ........................................................................................................ 7
Typographic and Syntax Conventions ........................................................................... 7

1 Corners Analysis ........................................................................................................ 9
Getting Started with Corners Analysis ......................................................................... 9
   How Corners Analysis Works .................................................................................... 9
   Opening, Resetting and Closing the Corners Window ................................................ 10
Getting to Know the Virtuoso® Analog Corners Analysis Window ............................... 12
   Menu ......................................................................................................................... 13
   Process and Base Directory Fields ................................................................ .......... 15
   Corner Definitions Pane ............................................................................................. 15
   Performance Measurements Pane ............................................................................. 18
   Split Pane Adjustment Bar ........................................................................................ 19
   Status Display ........................................................................................................... 19
   Keyboard Navigation and Shortcuts ......................................................................... 20
Running a Corners Analysis .......................................................................................... 20
   Defining the Process and Corners for an Analysis ..................................................... 21
   Defining Performance Measurements ....................................................................... 26
   Controlling the Corners Analysis ............................................................................. 28
Evaluating Corners Analysis Results ............................................................................. 29
   Text Outputs ............................................................................................................... 30
   Graphic Outputs ......................................................................................................... 31
Saving Setup Information .............................................................................................. 33
   Saving Setup Information to the Original Files ......................................................... 34
   Saving Setup Information to a Specified File ............................................................. 34
Saving a Script ................................................................................................................ 36
Using Process, Design, and Modeling Files ................................................................... 37
   Creating Process and Design Customization Files ..................................................... 38
   Using a .cdsinit File to Load PCFs and DCFs ............................................................ 42
2

Statistical Analysis

Getting Started with Statistical Analysis

How Statistical Analysis Works

Data Types Generated by the Statistical Analysis Tool

Opening the Analog Statistical Analysis Window

Getting to Know the Analog Statistical Analysis Window

Status Display

Menu

Analysis Setup Pane

Outputs Pane

Edit Fields

Button Bar

Running a Statistical Analysis

Specifying the Characteristics of a Statistical Analysis

Selecting Signals and Expressions to Analyze

Defining Correlations

Starting and Stopping the Analysis

Saving Statistical Analysis Results

Saving and Restoring a Statistical Analysis Session

How the Statistical Analysis Option Uses the Analysis Variation Setting
### Analyzing Results

- Loading Stored Statistical Analysis Results ........................................... 108
- Creating a New mcdata File from Saved Waveform Data .............................. 110
- Filtering Outlying Data ........................................................................... 110
- Setting Specification Limits ..................................................................... 113
- Generating Plots, Tables, and Reports ....................................................... 115

### Working through an Extended Example ................................................. 130
- Lowpass Filter Schematic .......................................................................... 130
- Model File ................................................................................................ 133
- Run Analog Simulation to Check Setup ...................................................... 134
- Specifying the Analysis in the Analog Statistical Analysis Window .......... 135
- Running the Statistical Analysis Simulation ................................................. 137
- Evaluating Statistical Analysis Results ....................................................... 139
- Changing Waveform Expressions at Post-simulation Time ......................... 148
- Changing Scalar Expressions at Post-Simulation Time ............................... 149
- Appending More Scalar Iterations to Existing Data .................................... 154
- Appending Waveforms From Different Statistical Analysis Runs. .............. 156
- Performing a Swept Parameter Statistical Analysis .................................... 157

### 3 Optimization ......................................................................................... 161

#### Getting Started with Optimization ....................................................... 162
- How Optimization Works ........................................................................... 162
- Getting Help ............................................................................................... 163

#### Getting to Know the Virtuoso® Analog Circuit Optimization Option Window ................................. 165
- Status Display ........................................................................................... 165
- Menu .......................................................................................................... 166
- Goals Pane ................................................................................................. 168
- Variables Pane .......................................................................................... 169
- Tool Bar ..................................................................................................... 170

#### Running an Optimization ................................................................. 170
- Defining Goals .......................................................................................... 171
- Preparing Design Variables ....................................................................... 186
- Controlling the Optimizer .......................................................................... 189
Preface

This manual describes how to use the Virtuoso® advanced analysis tools:

- The Virtuoso® analog statistical analysis option
- The Virtuoso® analog corners analysis option
- The Virtuoso® analog optimization analysis option

The guidance here is designed for users who are already familiar with circuit design and simulation.

Related Documents

The Virtuoso® advanced analysis tools are often used within the Virtuoso® analog design environment. The following documents give further information.

- All the analysis tools open from the Virtuoso® Analog Design Environment window. For information about using that window, see the Virtuoso® Analog Design Environment User Guide.
- For information about using the advanced analysis tools in the Open Command Environment for Analysis (OCEAN) environment, see the OCEAN Reference.
- For information about Cadence SKILL language procedural interface commands for the Corners customization files, see the Virtuoso® Analog Design Environment SKILL Language Reference.

Typographic and Syntax Conventions

The syntax conventions used in this documentation are described below.

**literal**

Words in nonitalic monospaced type indicate text you must type exactly as it is presented. These words represent command (function or routine) or option names or system output.
**argument...**

Words in italic monospaced type indicate text that you must replace with an appropriate argument or other data, such as a path. The three dots indicate that you can repeat the argument. Substitute one or more names or values.

**italic**

Words in italics indicate names of manuals, commands, and form buttons, form fields, and other features of the user interface (UI).
Corners Analysis

Corners analysis 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 Virtuoso® Analog Corners Analysis option, you can compare the results for each set of parameter values with the range of acceptable performance values. You can ensure the largest possible yield of circuits at the end of the manufacturing process by also revising the circuit, so that all the sets of parameters produce acceptable results.

This chapter explains in detail how you can use the corners analysis option to generate information about the yields from your circuit designs.

- “Getting Started with Corners Analysis” on page 9
- “Getting to Know the Virtuoso® Analog Corners Analysis Window” on page 12
- “Running a Corners Analysis” on page 20
- “Evaluating Corners Analysis Results” on page 29
- “Using Process, Design, and Modeling Files” on page 37
- “Working through an Extended Example” on page 62

Getting Started with Corners Analysis

This section briefly explains the theory behind corners analysis, tells you how to get help and describes how to open the Virtuoso® Analog Corners Analysis window.

How Corners Analysis Works

In a theoretical manufacturing process, process variables can have exact values and these exact values can be used to calculate the yield for the process. However, in a real manufacturing process, process variables are subject to a manufacturing tolerance—they
fluctuate randomly around their ideal values. The combined random variation for all the components results in an uncertain yield for the circuit as a whole.

Corners analysis looks at the performance outcomes generated from the most extreme variations expected in the process, voltage and temperature values (the *corners*).

With this information, you can determine whether the circuit performance specifications will be met, even when the random process variations combine in their most unfavorable patterns.

You can use Corners in the Virtuoso® Analog Design Environment in one of two ways:

1. By using the Corners user interface
2. By using your own process and design customization files

You will see more about these in subsequent sections of this chapter.

**Opening, Resetting and Closing the Corners Window**

To prepare for a corners analysis,

1. Ensure that the design you use is simulatable with nominal design parameter values.
2. Set up a simulation in the *Virtuoso® Analog Design Environment* window to run the analysis you want to use.
3. Ensure that all design variables in the circuit have an initial value.

If you have defined a set of customization files to be loaded automatically, the Virtuoso® Analog Corners Analysis window appears.

To reset the Virtuoso® Analog Corners Analysis window,

➤ Choose File – Reset.

Note: The reset operation clears the contents of the window and removes the information from the session as well. It does not delete the PCF and DCF files used within the session.

To close the Virtuoso® Analog Corners Analysis window,

➤ Choose File – Close.
Getting to Know the Virtuoso® Analog Corners Analysis Window

The Virtuoso® Analog Corners Analysis window contains the fields and controls required to specify the corners and measurements for the analysis you want to run.
Menu

The menu contains the commands needed to prepare for, run and plot the results of a corners analysis.

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td></td>
</tr>
<tr>
<td>Load</td>
<td>“Using the Graphical User Interface to Load PCFs and DCFs” on page 21</td>
</tr>
<tr>
<td>Save Setup</td>
<td>“Saving Setup Information to the Original Files” on page 34</td>
</tr>
<tr>
<td>Save Setup As</td>
<td>“Saving Setup Information to a Specified File” on page 34</td>
</tr>
<tr>
<td>Save Script As</td>
<td>“Saving a Script” on page 36</td>
</tr>
<tr>
<td>Reset</td>
<td>“Opening, Resetting and Closing the Corners Window” on page 10</td>
</tr>
<tr>
<td>Close</td>
<td>“Opening, Resetting and Closing the Corners Window” on page 10</td>
</tr>
<tr>
<td>Edit</td>
<td></td>
</tr>
<tr>
<td>Corner Definitions-&gt; Add Corner</td>
<td></td>
</tr>
<tr>
<td>Corner Definitions-&gt; Copy Corner</td>
<td></td>
</tr>
<tr>
<td>Corner Definitions-&gt; Enable Corner</td>
<td></td>
</tr>
<tr>
<td>Corner Definitions-&gt; Disable Corner</td>
<td></td>
</tr>
<tr>
<td>Corner Definitions-&gt; Add Variable</td>
<td>“Adding a Row for a New Design Variable” on page 25</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Corner Definitions-&gt;</td>
<td>“Deleting Corners and Rows” on page 25</td>
</tr>
<tr>
<td>Delete Selected</td>
<td></td>
</tr>
<tr>
<td>Performance Measurements-&gt;</td>
<td>“Creating a New Performance Measurement by Entering It Directly” on page 26,</td>
</tr>
<tr>
<td>Add Measurement</td>
<td>and “Creating a New Performance Measurement by Using the Calculator” on page 27</td>
</tr>
<tr>
<td>Performance Measurements-&gt;</td>
<td>“Deleting a Performance Measurement” on page 27</td>
</tr>
<tr>
<td>Delete Measurement</td>
<td></td>
</tr>
<tr>
<td>Setup</td>
<td></td>
</tr>
<tr>
<td>Add Process</td>
<td>“Using the Virtuoso® Analog Corners Analysis Window to Define a Process” on page 58</td>
</tr>
<tr>
<td>Delete Process</td>
<td>“Deleting One or More Processes” on page 59</td>
</tr>
<tr>
<td>Add/Update Model Info</td>
<td>“Using the Virtuoso® Analog Corners Analysis Window to Modify Process Model Information” on page 60</td>
</tr>
<tr>
<td>Simulation</td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>“Running and Stopping the Analysis” on page 28</td>
</tr>
<tr>
<td>Stop</td>
<td>“Running and Stopping the Analysis” on page 28</td>
</tr>
<tr>
<td>Tools</td>
<td></td>
</tr>
<tr>
<td>Calculator</td>
<td>“Creating a New Performance Measurement by Using the Calculator” on page 27</td>
</tr>
<tr>
<td>Get Expression</td>
<td>“Creating a New Performance Measurement by Using the Calculator” on page 27</td>
</tr>
<tr>
<td>Plot or Print Outputs</td>
<td>“Evaluating Corners Analysis Results” on page 29</td>
</tr>
<tr>
<td>Help</td>
<td></td>
</tr>
<tr>
<td>Contents</td>
<td>Displays the documentation (this user guide) containing information about the Virtuoso® Analog Corners Analysis option.</td>
</tr>
</tbody>
</table>
Process and Base Directory Fields

*Process* refers to the manufacturing process. Therefore, process parameters are parameters that pertain to the manufacturing process and are variables that help characterize the models specific to the manufacturing process.

The *Base Directory* field displays the path that contains the models used in the analysis for the process being displayed in the process field.

The base directory is usually defined by the *corAddProcess* command in a process customization file (PCF). You can also define the base directory by choosing *Setup – Add Process* or *Setup – Add/Update Model Info*.

The *Process* field displays either the name of the current process or *None*, if no process is specified. The processes are usually defined in customization files but can also be defined from the graphical user interface.

If there is no current process, the only active *Virtuoso® Analog Corners Analysis* window menu options are *File – Load*, *File – Close*, and *Setup – Add Process*. These menu options allow you to either load an existing file that defines a process or to define a new process.

Corner Definitions Pane

The *Corner Definitions* pane, located in the upper section of the *Virtuoso® Analog*
Corners Analysis window, displays information about the currently defined corners.

The information in this pane is usually loaded from process customization files (PCFs) and design customization files (DCFs) using paths defined in your .cdsinit file. For details refer to the section Using a .cdsinit File to Load PCFs and DCFs.

To define or revise corners, you modify the information in this pane.

- Each column characterizes a corner. You can select a column by clicking on the corresponding button along the top of the pane.
- You can also physically move the corner columns by dragging them around in the column header.
- You can alter the width of columns by grabbing the separation bar and dragging it one way or the other. There are certain limits (upper and lower) to how big or small you can make a column. This also differs by column type in the case of the measurement table.
- Disabled corners are grayed or fuzzed out in the form, while enabled corners are displayed in normal text.
- Each row (or variable) begins with a group name or design variable name, followed by the values to be used in each of the corners.
- You can select a variable by clicking on the corresponding button along the left side of the pane. You can drag a selection of variables by clicking and moving the cursor in the variable header.
- Editable data in the Corners table looks like normal text, while uneditable data in the Corners table has a dark gray background. Data loaded from a PCF is non-editable.
- Uneditable group/variant entries appear as a text field with the value of the field in text. Editable group/variant entries appear as a drop-down box.
Note: Temperature is a default variable with default value of 27. From release 5.1.41, this is an editable field.

The items in the Corner Definitions pane are described in the following table:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Corner</td>
<td>Click to add a new editable column to the right of the existing columns.</td>
</tr>
<tr>
<td>Add Variable</td>
<td>Click to add a new editable variable (row) below the existing variable (row).</td>
</tr>
<tr>
<td>Copy Corner</td>
<td>Click to add a new editable column filled with the data from a highlighted column. This column is added to the right of the existing columns</td>
</tr>
<tr>
<td>Delete</td>
<td>Click to delete a highlighted corner or variable (which is not added by the pcf file).</td>
</tr>
<tr>
<td>Disable</td>
<td>Click to disable a highlighted corner column. This particular corner will not be analysed after simulation and the column is grayed out. Once the column is disabled, the Disable button changes to an Enable button. You can re-enable the disabled corner by selecting it and clicking on the Enable button.</td>
</tr>
<tr>
<td>Run/Stop</td>
<td>Click Run to run all the corners that are not disabled. Click Stop to end a running corners analysis.</td>
</tr>
</tbody>
</table>

Click Run to run all the corners in the pane which are not disabled.

Note: Run turns to Stop once you start a run.
Performance Measurements Pane

The Performance Measurements pane, located in the lower section of the Virtuoso® Analog Corners Analysis window, displays information about the currently defined measurements.

The information in this pane is usually loaded from design customization files (DCF) using the loadDcf command in your .cdsinit file. In addition, any outputs defined in the Virtuoso® Analog Design Environment window when you first start the corners analysis option also appear. You can also load measurements from PCF files. You can also use the Calculator to get an expression.

**Note:** You can also add the Add Measurement button to add a measurement.

To specify or change the measurements, you modify the information in this pane. You can select a measurement by clicking in any of the fields in the measurement pane.

**Note:** The Cut, Paste and Copy keys work in the table fields. You can use these keys to copy a measurement expression into another expression, from within the Corners window.

The items in the Performance Measurements pane are described in the following table:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement column</td>
<td>Click to select a field in the Measurement column then type or edit a name to be used as the label when the expression is plotted or printed.</td>
</tr>
<tr>
<td>Expression column</td>
<td>Click to select a field in the Expression column then type or edit an expression to be evaluated for each corner.</td>
</tr>
<tr>
<td>Target column</td>
<td>Click to select a field in the Target column and then type the ideal target value for the measurement. This value is used when a residual plot is created.</td>
</tr>
</tbody>
</table>
Split Pane Adjustment Bar

The split pane adjustment bar between the Corner Definitions Pane and the Measurements Pane can be used to alter the area used by each pane. You can alter the area used by each pane by dragging the bar upwards or downwards.

Status Display

The status display shows messages in one of three colors, depending on the type of message.

Red Error Messages
The corners analysis tool also writes messages to the corners log file, corners0.log. The corners analysis tool puts the log file in the directory where you start the Cadence® software.

**Keyboard Navigation and Shortcuts**

Listed below are some shortcut keys that can be used for navigation of the form and tables. These keys can also be used during the row and column selection mode to change the selected row or column.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>Moves through table entries from left to right. Wraps to the next row and at the end jumps back to the top.</td>
</tr>
<tr>
<td>Shift-Tab</td>
<td>Reverse of Tab.</td>
</tr>
<tr>
<td>Arrow keys</td>
<td>Moves as expected, does not wrap around at all.</td>
</tr>
<tr>
<td>F2</td>
<td>Opens/Closes a cyclic box.</td>
</tr>
<tr>
<td>Page Down</td>
<td>Scrolls the table down if there is a vertical scroll bar.</td>
</tr>
<tr>
<td>Page Up</td>
<td>Scrolls the table up if there is a vertical scroll bar.</td>
</tr>
<tr>
<td>Home</td>
<td>Moves to the first column in the row.</td>
</tr>
<tr>
<td>End</td>
<td>Moves to the last column in the row.</td>
</tr>
</tbody>
</table>

**Running a Corners Analysis**

The following sections describe the major steps involved in setting up and running a corners analysis using PCFs and DCFs and also using the UI.

- “Defining the Process and Corners for an Analysis” on page 21
- “Defining Performance Measurements” on page 26
- “Controlling the Corners Analysis” on page 28
Defining the Process and Corners for an Analysis

To specify the corners for an analysis, you begin by loading a set of predefined corners from one or more process customization files (PCFs). The loading can occur automatically under the control of a .cdsinit file or you can load PCFs from the graphical user interface. For information about using the .cdsinit file to load PCFs and DCFs, see “Using a .cdsinit File to Load PCFs and DCFs” on page 42. For information about using the .cdsinit file to load PCFs and DCFs, see “Using a .cdsinit File to Load PCFs and DCFs” on page 42.

To tailor the predefined corners to the specific circuits you are working on, you can also load one or more files containing changes and additions to the basic set of corners. These files are called design customization files (DCFs). For information on preparing PCFs and DCFs, see “Creating Process and Design Customization Files” on page 38.

If, after loading the PCFs and DCFs, you find that more changes are necessary, you can use the graphical user interface to specify new corners or change any editable existing corners.

You can load multiple sets of corners information into the corners analysis option.

■ If you load a file or files that define more than one process, the processes appear in the Process cyclic field in the Virtuoso® Analog Corners Analysis window.

■ If you add more than one file (such as a DCF), that modifies a specific process, the contents of files that are loaded are added to the contents of the existing files.

You can add a process using the procedure described in “Using the Virtuoso® Analog Corners Analysis Window to Define a Process” on page 58.

Using the Graphical User Interface to Load PCFs and DCFs

There are various ways to load PCFs and DCFs. One such way is through the .cdsinit file. The .cdsinit file typically specifies the PCFs and DCFs, so usually when you open the Virtuoso® Analog Corners Analysis window, it already contains some corner, variable, and measurement definitions. However, if there are no predefined corners or if you need to load a different set, you can use the following steps to load PCFs and DCFs from the graphical user interface. The third way of loading PCFs and DCFs is through an OCEAN script.

If you have made changes in the Virtuoso® Analog Corners Analysis window, the Save Changes? dialog box appears.

2. Click either Save or Save As to save the changes. If you do not want to save the changes made, but want to load the PCFs and DCFs anyway, click Don’t Save. Click Cancel Load if you want to retain the existing set of PCFs and DCFs.

The Load dialog box appears.
3. Click on the *Look In* drop down field to go to the specific directory. You can also navigate using the iconified buttons located next to the *Look In* field. Placing the pointer on each of the buttons displays a tooltip that describes the function of the button, as follows:

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up One Level</td>
<td>Opens the directory one level above the active directory.</td>
</tr>
<tr>
<td>Home</td>
<td>Opens the home directory.</td>
</tr>
<tr>
<td>Create New Folder</td>
<td>Creates a new directory in the active directory.</td>
</tr>
<tr>
<td>List</td>
<td>Shows the names of the files in the directory.</td>
</tr>
<tr>
<td>Details</td>
<td>Shows additional information about the files such as the file size and the date last saved.</td>
</tr>
</tbody>
</table>

**Note:** You can double-click a directory folder to descend into that directory.

4. In the *Files of Types* field, select a type of file from the given list. The list of files is automatically updated. Default is *All Files* (*.*).

5. Select the file that you want to load. The name of the file is reflected in the *File Name* field. Click on *Load* button to load the file. Click on *Cancel* button if you want to cancel the operation.

### Specifying Additional Corners

If you need to specify additional corners from those loaded in the PCFs and DCFs, you can create new corners in the graphical user interface. You can either create new corners or copy existing corners and modify them.

### Creating a New Corner

To create a new corner,

1. Choose *Edit – Corner Definition – Add Corner*. You also click on the *Add Corner* button.

   The *Enter Corner Name* form is displayed.

2. Type a name for the new corner.

3. If you do not want to add the corner, click *Cancel Add Corner*. Click *OK* if you want to add the corner. A new column appears at the right side of the Corner Definitions pane of
the Virtuoso® Analog Corners Analysis window. The new column is named with the name from Step 2.

4. Edit the rest of the column as desired.

Copying and Modifying an Existing Corner

If one of the existing corners is similar to the corner you want to use, you can copy the existing corner and change the copy to meet your needs.

1. Highlight the column for the corner you want to copy.

2. Choose Edit – Corner Definition – Copy Corner. You can also click on the Copy Corners button.
   
   The Enter Corner Name form is displayed.

3. Enter a name for the new corner.

4. If you do not want to continue, click Cancel Copy Corner. Click OK if you want to continue. A new column appears at the right side of the Corner Definitions pane of the Virtuoso® Analog Corners Analysis window.

5. Fill in the rest of the column as necessary.

Enable Corner

1. Select a disabled corner.

2. Choose Edit – Corner Definitions – Enable Corner or click Enable. The selected corner will be enabled.

Disable Corner

1. Select an enabled corner.

2. Choose Edit – Corner Definitions – Disable Corner or click Disable. The selected corner will be disabled. This particular corner will not be analysed after simulation and the column is grayed out. Once the column is disabled, the Disable button changes to an Enable button. You can re-enable the disabled corner by selecting it and clicking on the Enable button.

Note: You can disable only an editable Variable (row) or column. Variables (rows) and Corners added by the pcf file are not editable. Variables (rows) and Corners added by the dcf file or the UI are editable.
Adding New Variables

There are three kinds of variables you can define for a corner: group variables, process variables and design variables. For information on adding group and process variables, see “Using the Virtuoso® Analog Corners Analysis Window to Modify Process Model Information” on page 60. For guidance on adding design variables, see the next section.

Adding a Row for a New Design Variable

To add a new design variable to the existing variables,

1. Choose Edit – Corner Definition – Add Variable. You can also click on the Add Variable button.

   The Enter Variable Name form is displayed.

2. Type a name for the new design variable.

   The design variable is added not only to the current process but also to all the other processes listed in the process cyclic field of the Virtuoso® Analog Corners Analysis window.

3. Click OK. If you want to continue. Otherwise, click Cancel Add Variable.

4. (Optional) Select the new variable field in each of the corners, and type the values you want to use.

Deleting Corners and Rows

You cannot delete corners and variables (rows) added by a PCF. However, if the DCFs load corners you do not plan to use, you can delete them. You can also delete un-needed rows added by DCFs or from the Virtuoso® Analog Corners Analysis window. Deleted corners and rows disappear from the Corners Definition pane of the Virtuoso® Analog Corners Analysis window and their underlying data is erased. Corners added from the Corners UI can also can be deleted.

Deleting Corners

To delete a corner,

1. Highlight the column for the corner you want to delete.

2. Choose Edit – Corner Definitions – Delete Selected or click Delete.

   The highlighted column disappears from the pane.
Deleting Rows

To delete a row,

1. Highlight the row you want to delete.
   
   **Note:** You can delete only rows defined by a DCF or added by using the Virtuoso® Analog Corners Analysis window. You cannot delete rows defined in a PCF.

2. Choose Edit – Corner Definitions – Delete Selected or click Delete.
   
   The highlighted row disappears from the pane.

Defining Performance Measurements

For convenience, measurements are often specified in design customization files (DCFs). Measurements defined in this way are displayed in the Performance Measurements pane, where you can examine them. In addition, any outputs defined in the Virtuoso® Analog Design Environment window when you first start the corners analysis option are also displayed.

If the existing measurements do not meet your needs, you can add new measurements or make and modify copies of the existing measurements. If you have no plans to use a measurement, you can delete it. You can add or change performance measurements either before or after you run the analysis. You can also specify measurements through a PCF file.

Creating a New Performance Measurement by Entering It Directly

To create a new performance measurement by entering it directly,

1. Choose Edit – Performance Measurements – Add Measurements or click Add Measurement.
   
   The Enter Measurement Name form is displayed.

2. Enter a name for the new Measurement.

3. Click Cancel Add Measurement if you do not want to continue. Click OK, if you want to continue. A new row will appear in the Corner Performance measurement pane.

4. Type the expression in the Expression field.

5. (Optional) Type the Target, Lower and Upper values for the new measurement. You need to specify these values only if you plan to use this performance measurement in a residual plot. A residual plot allows you to easily see whether a scalar measurement falls within the specified boundaries for all of your corners using a histogram like bar plot.
**Note:** *Target* is a target value for a scalar measurement. *Upper* is the acceptable upper boundary for a scalar measurement. *Lower* is an acceptable lower boundary for a scalar measurement. If *Target*, *Upper* and *Lower* bound are set for a waveform, they will not be used at all. These options are only used for the residual plots.

**Creating a New Performance Measurement by Using the Calculator**

To create a new performance measurement using the calculator,

1. Choose *Edit – Performance Measurements – Add Measurement* or click *Add Measurement*.
   
   The *Enter Measurement Name* form is displayed.

2. Type a name for the new measurement.

3. Click *Cancel Add Measurement* if you do not want to continue. Click *OK*, if you want to continue. A new row will appear in the *Corner Performance* measurement pane.

4. Choose *Tools ->Calculator* or click *Calculator* to open the calculator window.

5. Build the measurement expression in the calculator.

   For information on using the calculator, see the *Waveform Calculator User Guide*.

6. In the *Virtuoso® Analog Corners Analysis* window, select the *Expression* field for the new measurement.

   **Note:** If the expression field is selected, then only the *Get Expression* button will be highlighted.

7. Choose *Tools – Get Expression*, or click *Get Expression* to retrieve the expression from the calculator and place it in the selected *Expression* field.

8. (Optional) Type the *Target*, *Lower* and *Upper* values for the new measurement. You need to specify these values only if you plan to use this performance measurement in a residual plot.

**Deleting a Performance Measurement**

You can delete any measurement displayed in the *Virtuoso® Analog Corners Analysis* window. A deleted measurement disappears from the Performance Measurements pane, and the data underlying it is erased. If you delete a measurement and then save the setup, the deleted measurement is not included in the saved setup.

To delete a measurement,
1. Highlight the row for the measurement you want to delete.

2. Choose Edit – Performance measurement – Delete Measurement, or click Delete Measurement.

   The highlighted measurement disappears from the pane.

**Controlling the Corners Analysis**

You are ready to run the analysis after defining the corners and specifying the performance measurements you need. To do this,

- Disable the corners that you do not want to run.
- Select the output format for the measurements to determine what type of measurement output you want, if any. You can use the Plot and Print checkboxes to specify whether you require a graphic or text output.

To choose the output formats for each measurement, click the Plot and Print checkboxes on the right side of the Performance Measurements pane.

Then, run the analysis.

**Running and Stopping the Analysis**

To run the analysis,

> Choose Simulation – Run or click Run.

To stop an analysis running on a single machine,

1. Choose Simulation – Stop or click Stop.
To stop a simulation running distributed, use Job Monitor. For more information about Job Monitor, refer to the Virtuoso Analog Distributed Processing Option User Guide.

**Note:** The *Run* button automatically changes between *Run/Stop* depending on whether a Corners process is currently running or not.

### Evaluating Corners Analysis Results

When the analysis finishes, the corners analysis option plots or lists the results according to whether you chose text or graphic outputs in the *Performance Measurements* pane.

**Note:** If you run a distributed simulation, the results do not plot or list automatically.

If you want a different set of outputs from those you chose before running the analysis, you can make new choices in the *Performance Measurements* pane and then choose *Tools – Plot or Print Outputs* from the menu). In response, the corners analysis option evaluates the selected measurements and displays new lists or plots.
Text Outputs

For a scalar measurement, a text output looks like this.

Each column in this window displays the value of a scalar measurement for each of the corners. In this example, bandwidth varies from a low of 130.7 K under the slowslow corner conditions, to a high of 318.7 K under the fastfast corner conditions.
For a waveform measurement, a text output looks like this.

![Results Display Window](image)

The first column in this window lists the data points for an analysis. Each subsequent column lists data for a particular corner. In this example, at a frequency of 758.6 Hz, the phase for the slowslow corner is -336.1 m and for the fastfast corner is -137.6 m.

**Graphic Outputs**

There are two kinds of graphic output, a residual plot for scalar data and a family-of-curves plot for waveform data. A residual plot allows you to easily see whether a scalar measurement
falls within the specified boundaries for all of your corners using a histogram like bar plot. A residual plot looks like this.

The preceding residual plot, has a specified value of 200 for Target and shows that four of the corners produce values within specifications. One corner produces a value that does not lie within the lower boundary of the acceptable range. This result implies that yield for the manufactured circuit will be less than 100 percent if the circuit is produced in its current form. For greater yield, the circuit designer might want to redesign the circuit so it performs acceptably for all the corners.
The set of curves for all the corners, looks like this.

The preceding plot shows how the phase varies as a function of frequency for each one of the corners.

**Saving Setup Information**

The corners option setup consists of all the information in the *Virtuoso® Analog Corners Analysis* window, including the corner definitions and performance measurements. With menu selections in the *File* entry, you can save the setup back to the original files, save the setup to a specified file, and load a saved setup.
Saving Setup Information to the Original Files

To save the current setup back to the files from which it was loaded,

➤ Choose File – Save Setup.

If the PCFs and DCFs are not writable, the corners analysis option reports an error, and changes and additions made in the Virtuoso® Analog Corners Analysis window are not saved.

If the PCFs and DCFs are writable, the corners analysis option saves changes back to those files, overwriting the original contents of those files. As a result, any comments you might have in the PCFs or DCFs are overwitten and lost. The corners analysis option saves any additions to the DCF loaded last, if possible or to a newly created file called NewEntries.dcf. This implies that if you add any Corners, Variables or Measurements, they are added to the last loaded DCF. The NewEntries.dcf file is created when no PCF and no DCF are loaded.

To avoid overwriting comments and to have the changed setup saved in a single easy-to-understand location, use File -> Save Setup As, described in the following section.

Saving Setup Information to a Specified File

To save the current setup in a file you specify,

1. Choose File – Save Setup As.
The *Save Setup As* form is displayed.

2. Click on the *Look In* drop down field to go to the specific directory. You can also navigate using the iconified buttons located next to the *Look In* field. Placing the pointer on each of the buttons displays a tooltip that describes the function of the button, as follows:

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up One Level</td>
<td>Opens the folder one level above the active folder.</td>
</tr>
<tr>
<td>Home</td>
<td>Opens the home directory.</td>
</tr>
<tr>
<td>Create New Folder</td>
<td>Creates a new folder in the active folder.</td>
</tr>
<tr>
<td>List</td>
<td>Shows the names of the files in the directory.</td>
</tr>
<tr>
<td>Details</td>
<td>Shows additional information about the files such as the file size and the date last saved.</td>
</tr>
</tbody>
</table>

**Note:** You can double-click a directory folder to descend into that directory.
3. In the *Files of Types* field, select a type of file from the given list. The list of files is automatically updated. Default is *All Files* (*.*).

Select the file where the setup information is to be saved. The name of the file is reflected in the *File Name* field.

4. Click *Save*. Click on the *Cancel* button if you want to cancel the operation.

**Note:** If you double-click on a selected file, the information will be directly saved to the file.

All of the existing corners option information, whether loaded from PCFs, DCFs or through the corners option graphical user interface, is saved to the file you specify. If necessary, you can then cut and paste the lines into other PCFs and DCFs.

**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 you can run from a UNIX shell or from the Command Interpreter Window (CIW). You can type OCEAN commands in an interactive session, or you can create scripts containing your commands and load those scripts into OCEAN.

You can use the *Virtuoso® Analog Corners Analysis* window to set up the analysis you need, and save the setup procedure in an OCEAN script. You can then edit the script to add simulation or postprocessing commands as needed.

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

To create a script and save it,

► Choose *File – Save Script As*. 
The Save Script As form appears so you can specify a file for the script.

Using Process, Design, and Modeling Files

There are usually three different kinds of files associated with setting up a corners analysis.

- Process customization files (PCFs) define processes, groups, variants, and corners shared by an entire organization. PCFs are usually created by a process engineer or process group.

- Design customization files (DCFs) contain definitions used for a particular design or for several designs within a design group. DCFs are usually created by designers, who use the DCFs to add design-specific information to the general information provided in PCFs.

- Modeling files specify the model parameter values to be used for components during a corners analysis. These files are usually created by a process engineer or process group.
Day-to-day use of the corners analysis option typically does not involve changing a PCF or DCF. However, if you are involved in writing or changing these kinds of files, read the following sections for guidance.

**Creating Process and Design Customization Files**

The process customization files (PCFs) and design customization files (DCFs) contain Cadence SKILL language commands that define the basic corners and measurements to be used during analysis. The following sections illustrate how you can use the commands to develop the set of definitions you need.

You can use any of the corners option SKILL language PI commands in either the PCFs or DCFs. However, the commands used to define the process, the corners, and the corner variables are customarily placed in the PCF. The commands used to specify design variables and measurements, because they are design specific, are usually placed in the DCF.

### Commands Normally in a PCF

- `corAddProcess`
- `corSetModelFile`
- `corAddCorner`
- `corAddGroupAndVariantChoices`
- `corAddModelFileAndSectionChoices`
- `corSetCornerModelFileSection`
- `corAddProcessVar`
- `corSetProcessVarVal`
- `corSetCornerGroupVariant`
- `corSetCornerNomTempVal`

### Commands Normally in a DCF

- `corAddDesignVar`
- `corSetDesignVarVal`
- `corSetCornerRunTempVal`
- `corAddMeas`
- `corSetMeasExpression`
- `corSetMeasLower`
- `corSetMeasUpper`
- `corSetMeasTarget`
- `corSetMeasGraphicalOn`
- `corSetMeasTextualOn`

### Commands Used in Both

- `corSetCornerVarVal`
- `corCopyCorner`

The `corSetModelFile` command can be used only with the single model library style.

For more information, including the formal syntax for the commands, see the *Virtuoso® Analog Design Environment SKILL Language Reference*.

To debug PCFs and DCFs, consider using OCEAN. The feedback the corners analysis option provides is limited, but OCEAN provides more detailed feedback that makes it easier to find and correct errors. For examples of OCEAN scripts that illustrate using PCFs and DCFs, see the following directory in your installation hierarchy:
Example: Preparing a Process Customization File

The process customization file (PCF) adds the name of a new process to the corners analysis option graphical user interface and defines the basic set of corners. For example, the following PCF adds the process name P50u, specifies the modeling style as singleModelLib, and defines three corners: slowslow, nominal, and fastfast.

; Example PCF file for the process P50u.
corAddProcess( "P50u" "~/processes" 'singleModelLib' )
corSetModelFile("P50u" "P50uModelFile.scs")
; Prepare to add a process variable to each corner.
corAddProcessVar( "P50u" "EdgeEffect" )
; Now add the corners, specifying the values and choices for each.
corAddCorner( "P50u" "fastfast" )
corSetCornerVarVal( "P50u" "fastfast" "EdgeEffect" "1.18" )
corAddCorner( "P50u" "slowslow" )
corSetCornerVarVal( "P50u" "slowslow" "EdgeEffect" "1.12" )
corAddCorner( "P50u" "nominal" )
corSetCornerVarVal( "P50u" "nominal" "EdgeEffect" "1.15" )

The modeling values for the fastest, typical, and slowest variants are not defined in the PCF. Instead, they are defined in the modeling file. For example, assume the P50uModelFile.scs referred to by the P50u PCF contains the following statements.

```bash
.LIB slowest
.model npn2 npn tf=120n
.model nmosR nmos tox=120n
.ENDL slowest

.LIB typical
.model npn2 npn tf=100n
.model nmosR nmos tox=100n
.ENDL typical

.LIB fastest
.model npn2 npn tf=80n
.model nmosR nmos tox=80n
.ENDL fastest
```
Loading `P50u PCF`, which refers to the `P50uModelFile.scs`, produces the following arrangement in the *Virtuoso Analog Corners Analysis* window.

![Image of Virtuoso Analog Corners Analysis window]

**Note:** `temp` is always added with 27 being the default value for all corners.

### Example: Preparing a Design Customization File

The DCF adds design-specific variables and measurements to the corners analysis option graphical user interface that is specified in general by information in a PCF. For example, the following DCF adds a design variable, sets the run temperature, and adds information to the *Performance Measurements* pane:

```plaintext
corAddDesignVar( "vss" )
corSetDesignVarVal( "vss" "" )
corSetCornerVarVal( "P50u" "fastfast" "vss" "70" )
corSetCornerVarVal( "P50u" "slowslow" "vss" "50" )
corSetCornerVarVal( "P50u" "nominal" "vss" "60" )
corSetCornerRunTempVal("P50u" "slowslow" -35)
; You must add the measurement before you define it.
corAddMeas( "bandwidth" )
corSetMeasExpression( "bandwidth" "bandwidth(VF('/vout') 3 'low')" )
corSetMeasLower("bandwidth" "8Mhz")
corSetMeasUpper("bandwidth" "12Mhz")
corSetMeasTarget("bandwidth" "10Mhz")
```
Loading this DCF along with the P50u PCF described in the previous section changes both panes in the Virtuoso® Analog Corners Analysis window. The Corner Definitions pane looks like this.

![Corner Definitions Table](image)

The Performance Measurements pane looks like this.

![Performance Measurements Table](image)
Using a .cdsinit File to Load PCFs and DCFs

A convenient way to load process and design customization files is to use your .cdsinit file. You can set up your files in the following ways.

To load both PCFs and DCFs explicitly in the .cdsinit file

Make sure your .cdsinit file loads all the necessary PCFs and DCFs.

For example, this .cdsinit file loads several PCFs and DCFs.

```plaintext
loadPcf "process1.pcf"
loadPcf "process2.pcf"
loadDcf "cellPhone23.dcf"
loadDcf "opamp47.dcf"
```

To load DCFs explicitly and have them load the PCFs they need

1. Add load statements to each DCF for the PCF files it uses. That way, when you load the DCF file, it loads the PCF files automatically.

For example, this fragment of the myanalog35u.dcf file loads the analog35u.pcf file.

```
; This is the myanalog35u.dcf
loadPcf("mypath/
analog35u.pcf")
```

2. Set up your .cdsinit file so it loads the DCF. For example, this .cdsinit file fragment loads the myanalog35u.dcf file (which then loads the analog35u.pcf).

```
loadDcf("/mnt4/radhikak/
tools/
    dfII/src/corners/
    myanalog35u.dcf")
```

Whichever way you choose to load your files, you must make sure PCFs and DCFs refer only to definitions that have already been loaded. Usually, that means you must load PCFs before you can define corners or measurements in a DCF.

Implementing Modeling Styles

The Corners analysis option supports five different modeling styles: single model library, multiple model library, single numeric, multiple numeric and multiple parametric.

Starting with IC 5.1.41 release, some changes have been made to the Corners modeling styles. The single numeric style is the default. For third-party OASIS corner integration, the single model library and single numeric styles are also supported. To use and define corner processes, you need to have a valid tool session created before loading PCF files.
Cadence recommends the *single model library* or *multiple model library* styles for users running the *Cadence® Spectre® Circuit Simulator*. For users running the SpectreS simulator, Cadence recommends the *multiple numeric* modeling style. The remaining two modeling styles, *single numeric* and *multiple parametric*, should be used with caution.

The following sections illustrate the file structures used by these styles and give examples of PCFs tailored to each style. For detailed information, see the sections listed below.

- “Single Model Library Style”
- “Multiple Model Library Style” on page 49
- “Single Numeric Style” on page 52
- “Multiple Numeric Style” on page 53
- “Multiple Parametric Modeling” on page 55

**Single Model Library Style**

Cadence recommends this easy-to-read style for use in corners analysis. With this approach,

- All models for all corners are located in a single model file
- The model file is located in the base directory
- The model file can have any name

You can type the name in the *Virtuoso® Analog Corners Analysis* window or use the `corSetModelFile` procedure to specify the name in a PCF or DCF.
The following table illustrates the single model library style with an example path, file, and file contents. If you prefer, you can also use the .LIB syntax for this modeling style. The .LIB syntax is an hspice modelling syntax that is supported in Spectre.

### Single Model Library Style (Native Spectre)

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>./CORNERS/fab6/</td>
<td>mylibfile.scs</td>
<td>library processA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>section slowslow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn2 npn tf=120n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn9 npn tf=320n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmosR nmos tox=120n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmos8 nmos tox=320n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>section nom</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn2 npn tf=100n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn9 npn tf=300n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmosR nmos tox=100n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmos8 nmos tox=300n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>section fastfast</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn2 npn tf=80n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model npn9 npn tf=380n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmosR nmos tox=80n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>model nmos8 nmos tox=380n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endsection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endlibrary</td>
</tr>
</tbody>
</table>

The following code illustrates how you can refer to this modeling structure in a PCF.

```csharp
corAddProcess("fab6" "./CORNERS/fab6/" 'singleModelLib)
corSetModelFile("fab6" "mylibfile.scs")
corAddProcessVar("fab6" "vdc")
corAddCorner("fab6" "slowslow"
  ?runTemp  20
  ?nomTemp  27
  ?vars ' ( ("vdc" 2) )
)
```
The *Virtuoso® Analog Corners Analysis* window produced by this PCF looks like this.

![Virtuoso® Analog Corners Analysis](image)

**Changes for Setting New Groups or Variants in the singleModelLib Style**

Starting with the IC 5.1.41 release, you can specify the same section in a single model library style with several corners associated with your design. These changes require that you define the `corSetCornerGroupVariant()` function as follows:

```
corSetCornerGroupVariant( <process_Name> <corner_Name> <group_Name> <variant> )
```

where:

- **process_Name** The existing process you want to use, for example, `fab6`.
- **corner_Name** The variant name, the sections in your model file.
- **group_Name** Any name with which you want to group your variants.
- **variant** The name of the variant you want to specify with different corners.

The following code illustrates how you can refer to same section or variant within several corners or groups using the single model library style in a PCF:

```
corAddProcess("singleModelLib" "./CORNERS/singleModelLib" 'singleModelLib')

corSetModelFile("singleModelLib" "singleModelLib.scs")
```
corAddModelFileAndSectionChoices("singleModellLib" "groupname" '( "slowslow" "slowfast" "typTyp" "fastslow" "fastfast") )

corAddProcessVar("singleModellLib" "vdd")
corAddProcessVar("singleModellLib" "vss")
corAddDesignVar( "Cload" )

corAddCorner( "singleModellLib" "cor1" )
corSetCornerNomTempVal( "singleModellLib" "cor1" 27 )
corSetCornerRunTempVal( "singleModellLib" "cor1" 125 )
corSetCornerVarVal( "singleModellLib" "cor1" "Cload" "260f" )
corSetCornerVarVal( "singleModellLib" "cor1" "vss" "-2.7" )
corSetCornerVarVal( "singleModellLib" "cor1" "vdd" "2.7" )
corSetCornerGroupVariant( "singleModellLib" "cor1" "groupname" "slowslow" )

corAddCorner( "singleModellLib" "cor2" )
corSetCornerNomTempVal( "singleModellLib" "cor2" 27 )
corSetCornerRunTempVal( "singleModellLib" "cor2" 27 )
corSetCornerVarVal( "singleModellLib" "cor2" "Cload" "200f" )
corSetCornerVarVal( "singleModellLib" "cor2" "vss" "-3" )
corSetCornerVarVal( "singleModellLib" "cor2" "vdd" "3" )
corSetCornerGroupVariant( "singleModellLib" "cor2" "groupname" "slowslow" )

corAddCorner( "singleModellLib" "cor3" )
corSetCornerNomTempVal( "singleModellLib" "cor3" 27 )
corSetCornerRunTempVal( "singleModellLib" "cor3" 27 )
corSetCornerVarVal( "singleModellLib" "cor3" "Cload" "200f" )
corSetCornerVarVal( "singleModellLib" "cor3" "vss" "-3" )
corSetCornerVarVal( "singleModelLib" "cor3" "vdd" "3" )
corSetCornerGroupVariant( "singleModelLib" "cor3" "groupname" "fastfast" )

corAddCorner( "singleModelLib" "cor4" )
corSetCornerNomTempVal( "singleModelLib" "cor4" 27 )
corSetCornerRunTempVal( "singleModelLib" "cor4" 27 )
corSetCornerVarVal( "singleModelLib" "cor4" "Cload" "200f" )
corSetCornerVarVal( "singleModelLib" "cor4" "vss" ":-3" )
corSetCornerVarVal( "singleModelLib" "cor4" "vdd" "3" )
corSetCornerGroupVariant( "singleModelLib" "cor4" "groupname" "fastfast" )

corAddCorner( "singleModelLib" "cor5" )
corSetCornerNomTempVal( "singleModelLib" "cor5" 27 )
corSetCornerRunTempVal( "singleModelLib" "cor5" -55 )
corSetCornerVarVal( "singleModelLib" "cor5" "Cload" "160f" )
corSetCornerVarVal( "singleModelLib" "cor5" "vss" ":-3.3" )
corSetCornerVarVal( "singleModelLib" "cor5" "vdd" "3.3" )
corSetCornerGroupVariant( "singleModelLib" "cor5" "groupname" "slowslow" )

corAddCorner( "singleModelLib" "cor6" )
corSetCornerNomTempVal( "singleModelLib" "cor6" 27 )
corSetCornerRunTempVal( "singleModelLib" "cor6" -55 )
corSetCornerVarVal( "singleModelLib" "cor6" "Cload" "190f" )
corSetCornerVarVal( "singleModelLib" "cor6" "vss" ":-3.0" )
corSetCornerVarVal( "singleModelLib" "cor6" "vdd" "3.0" )
corSetCornerGroupVariant( "singleModelLib" "cor6" "groupname" "slowfast" )

corAddMeas( "DCgain" )
corSetMeasExpression( "DCgain" "ymax(db20(VF('/vout'))))" )
corSetMeasTarget( "DCgain" 60 )
corSetMeasEnabled( "DCgain" t )
corSetMeasGraphicalOn( "DCgain" t )
corSetMeasTextualOn( "DCgain" nil )

corAddMeas( "bandwidth" )
corSetMeasExpression( "bandwidth" "bandwidth(VF('/vout') 3 'low'))" )
corSetMeasEnabled( "bandwidth" t )
corSetMeasGraphicalOn( "bandwidth" t )
corSetMeasTextualOn( "bandwidth" nil )

corAddMeas( "gain" )
corSetMeasExpression( "gain" "dB20(VF('/vout'))" )
corSetMeasEnabled( "gain" t )
corSetMeasGraphicalOn( "gain" t )
corSetMeasTextualOn( "gain" nil )

corAddMeas( "phase" )
corSetMeasExpression( "phase" "phase(VF('/vout'))" )
corSetMeasEnabled( "phase" t )
corSetMeasGraphicalOn( "phase" t )
corSetMeasTextualOn( "phase" nil )
The *Virtuoso® Analog Corners Analysis* window produced by this PCF looks like this.

![Corner Definitions Table]

<table>
<thead>
<tr>
<th>Variables</th>
<th>Cor1</th>
<th>Cor2</th>
<th>Cor3</th>
<th>Cor4</th>
<th>Cor5</th>
<th>Cor6</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupname</td>
<td>slowslow</td>
<td>slowslow</td>
<td>fastfast</td>
<td>fastfast</td>
<td>slowslow</td>
<td>slowfast</td>
</tr>
<tr>
<td>temp</td>
<td>125</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>-55</td>
<td>-55</td>
</tr>
<tr>
<td>vdd</td>
<td>2.7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3.3</td>
<td>3</td>
</tr>
<tr>
<td>vss</td>
<td>-2.7</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-3.3</td>
<td>-3</td>
</tr>
<tr>
<td>ClOed</td>
<td>260f</td>
<td>200f</td>
<td>200f</td>
<td>200f</td>
<td>160f</td>
<td>190f</td>
</tr>
</tbody>
</table>

**Multiple Model Library Style**

This style uses multiple library files, which must be specified by using the `corAddModelFileAndSectionChoices` and `corAddCorner` commands in a PCF or DCF. In other ways, this style is the same as the single model library style. For example, the models might be located in the following files:

- `./CORNERS/fab6/path1/npn.scs`
- `./CORNERS/fab6/path3/nmos.scs`
The following table illustrates the multiple model library style. If you prefer, you can also use the .LIB syntax for this modeling style.

### Multiple Model Library Style

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
</table>
| ./CORNERS/fab6/ path1/ | npn.scs  | library npn  
|              |          | section slow  
|              |          | section nom   
|              |          | section fast  
|              |          | model npn2 bjt tf=120n  
|              |          | model npn8 bjt tf=80n  
|              |          | endsection    
|              |          | model npn2 bjt tf=100n  
|              |          | model npn8 bjt tf=60n  
|              |          | endsection    
|              |          | section fast  
|              |          | model npn2 bjt tf=80n  
|              |          | model npn8 bjt tf=50n  
|              |          | endsection    
|              |          | endlibrary    |

| ./CORNERS/fab6/ path3/ | nmos.scs  | library nmos  
|                       |          | section slow  
|                       |          | section nom   
|                       |          | section fast  
|                       |          | model nmosR mos3 tox=120n  
|                       |          | model nmos2 mos3 tox=140n  
|                       |          | endsection    
|                       |          | model nmosR mos3 tox=100n  
|                       |          | model nmos2 mos3 tox=115n  
|                       |          | endsection    
|                       |          | section fast  
|                       |          | model nmosR mos3 tox=80n  
|                       |          | model nmos2 mos3 tox=90n  
|                       |          | endsection    
|                       |          | endlibrary    |

The following code illustrates how you can refer to this multiple model library structure in a PCF.

```plaintext
corAddProcess("fab6" "./CORNERS/fab6/" 'multipleModelLib)
corAddModelFileAndSectionChoices("fab6" "path1/npn.scs"
```

Virtuoso® Analog Corners Analysis window produced by this PCF looks like this.
Single Numeric Style

This modeling style is provided for backward compatibility. If you plan to run your corners analysis with the Spectre simulator, Cadence recommends that you convert to a preferred modeling style.

- With this style, each corner is located in a separate file. If there are four corners, there are four model files. All the model files have the same name.
- Each model file is located in the subdirectory `base_directory/corner_name`. For example, if one of the corner names is `allfast`, then one of the model files is located in the `base_directory/allfast` subdirectory.
- The common model filename can be anything.

- If you use the Spectre direct simulator, specify the name by choosing Setup – Model Libraries from the menu in the Virtuoso® Analog Design Environment window, then type the name into the Model Library Setup form.
- If you use a socket simulator, choose Setup – Environment to open the Environment Options form, then type the name into the Include File field.

The following table illustrates the file structure and contents for a model with three corners, using the single numeric modeling style.

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>./CORNERS/fab6/</td>
<td>models</td>
<td>.model npn2 npn tf=120n</td>
</tr>
<tr>
<td>allslow/</td>
<td></td>
<td>.model npn9 npn tf=320n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmosR nmos tox=120n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmos8 nmos tox=320n</td>
</tr>
<tr>
<td>./CORNERS/fab6/</td>
<td>models</td>
<td>.model npn2 npn tf=100n</td>
</tr>
<tr>
<td>allnom/</td>
<td></td>
<td>.model npn9 npn tf=300n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmosR nmos tox=100n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmos8 nmos tox=300n</td>
</tr>
<tr>
<td>./CORNERS/fab6/</td>
<td>models</td>
<td>.model npn2 npn tf=80n</td>
</tr>
<tr>
<td>allfast/</td>
<td></td>
<td>.model npn9 npn tf=380n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmosR nmos tox=80n</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.model nmos8 nmos tox=380n</td>
</tr>
</tbody>
</table>

The following code illustrates how you can refer to this modeling structure in a PCF:

```
corAddProcess("fab6" "./CORNERS/fab6" 'singleNumeric)
corAddCorner("fab6" "allslow"
   ?runTemp -55)
```
corAddCorner("fab6" "allnom"
    ?runTemp -27
)  
corAddCorner("fab6" "allfast"
    ?runTemp 55
)

The Virtuoso® Analog Corners Analysis window produced by this PCF looks like this.

![Virtuoso Analog Corners Analysis window](image)

**Multiple Numeric Style**

This modeling style, which has the following characteristics, is provided for backward compatibility.

- With this style, each model is defined in a separate file. All model parameters are defined with numeric values.
- Each model file is located in the subdirectory `base_directory/group/variant`. For example, if the model includes the group `npn` and the variant `fast`, then at least one of the model files is located in the `base_directory/npn/fast` subdirectory.
- Each model file can have any name, which the designer enters on the Edit Object Properties form in the Virtuoso® Analog Design Environment.

The following table illustrates the file structure and contents for the multiple numeric style.

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>./CORNERS/fab6/npn/slow/</td>
<td>npn2.scs</td>
<td>model npn2 bjt tf=120n</td>
</tr>
<tr>
<td></td>
<td>npn9.scs</td>
<td>model npn9 bjt tf=320n</td>
</tr>
<tr>
<td>./CORNERS/fab6/npn/nom/</td>
<td>npn2.scs</td>
<td>model npn2 bjt tf=100n</td>
</tr>
<tr>
<td></td>
<td>npn9.scs</td>
<td>model npn9 bjt tf=300n</td>
</tr>
</tbody>
</table>
Multiple Numerics, continued

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>./CORNERS/fab6/npn/fast/</td>
<td>npn2.scs</td>
<td>model npn2 bjt tf=80n</td>
</tr>
<tr>
<td>nnp9.scs</td>
<td></td>
<td>model npn9 bjt tf=380n</td>
</tr>
<tr>
<td>./CORNERS/fab6/nmos/slow/</td>
<td>nmosR.scs</td>
<td>model nmosR mos3 tox=120</td>
</tr>
<tr>
<td>nmos8.scs</td>
<td></td>
<td>model nmos8 mos3 tox=320n</td>
</tr>
<tr>
<td>./CORNERS/fab6/nmos/nom/</td>
<td>nmosR.scs</td>
<td>model nmosR mos3 tox=100n</td>
</tr>
<tr>
<td>nmos8.scs</td>
<td></td>
<td>model nmos8 mos3 tox=300n</td>
</tr>
<tr>
<td>./CORNERS/fab6/nmos/fast/</td>
<td>nmosR.scs</td>
<td>model nmosR mos3 tox=80n</td>
</tr>
<tr>
<td>nmos8.scs</td>
<td></td>
<td>model nmos8 mos3 tox=380n</td>
</tr>
</tbody>
</table>

The following code illustrates how you can refer to this modeling structure in a PCF.

```plaintext
corAddProcess("fab6" "./CORNERS/fab6" 'multipleNumeric)
corAddGroupAndVariantChoices("fab6" "nnp"
  '("slow" "nom" "fast")
)
corAddGroupAndVariantChoices("fab6" "nmos"
  '("slow" "nom" "fast")
)
corAddCorner("fab6" "slowslow"
  ?variants '(
    "nnp" "slow"
    "nmos" "slow"
  )
  ?nomTemp -55
  ?vars ' ( "Cload" 260f )
)
corAddCorner("fab6" "slowfast"
  ?variants '(
    "nnp" "slow"
    "nmos" "fast"
  )
  ?nomTemp -55
  ?vars ' ( "Cload" 200f )
)```
The *Virtuoso® Analog Corners Analysis* window produced by this PCF looks like this.

![Virtuoso Analog Corners Analysis Window](image)

**Using the Multiple Numeric Modeling Style with the Spectre Simulator**

When you run a multiple numeric modeling style corners analysis with the Spectre simulator, ensure that the `.cdsenv` variable `includeStyle` is set to `t`.

**Multiple Parametric Modeling**

This modeling style, which has the following characteristics, is provided for backward compatibility.

- With this style, each model is defined in a separate file. There is a corresponding parameter file for every model associated with each corner.

- Model files are located in the subdirectory `base_directory/group`. For example, if the model includes the group `n npn`, then the model files associated with that group are located in the `base_directory/n npn` subdirectory.

- Each parameter file is located in `base_directory/group/variant`. For example, if the model includes the group `n npn` and the variant `fast`, then at least one of the parameter files is located in the `base_directory/n npn/fast` subdirectory.

- Each model file can have any name, which the designer enters on the Edit Properties form in the *Virtuoso® Analog Design Environment*. 
The following table illustrates the file structure and contents for the multiple parametric modeling style.

**Multiple Parametric Style**

<table>
<thead>
<tr>
<th>Path</th>
<th>Filename</th>
<th>File Contents</th>
</tr>
</thead>
</table>
| .Corners/66/npn | npn2.scs  | include "npn2.param"
                |           | model npn2 bjt tf=TF2                              |
|                 | npn9.scs  | include "npn9.param"
                |           | model npn9 bjt tf=TF9                              |
| .Corners/66/npn/slow/ | npn2.param | parameter TF2=120n                                |
|                 | npn9.param | parameter TF9=320n                                |
| .Corners/66/npn/nom/  | npn2.param | parameter TF2=100n                                |
|                 | npn9.param | parameter TF9=300n                                |
| .Corners/66/npn/fast/ | npn2.param | parameter TF2=80n                                 |
|                 | npn9.param | parameter TF9=380n                                |
| .Corners/66/nmos | nmosR.scs | model npn2 mos3 tf=TOXR                            |
|                 | nmos8.scs | model npn9 mos3 tf=TOX8                            |
| .Corners/66/nmos/slow/ | nmosR.param | parameter TOXR=120                                |
|                 | nmos8.param | parameter TOX8=320n                               |
| .Corners/66/nmos/nom/  | nmosR.param | parameter TOXR=100n                               |
|                 | nmos8.param | parameter TOX8=300n                               |
| .Corners/66/nmos/fast/ | nmosR.param | parameter TOXR=80n                                |
|                 | nmos8.param | parameter TOX8=380n                               |

The following code illustrates how you can refer to this modeling structure in a PCF.

```plaintext
corAddProcess("fab6" ".Corners/66" 'multipleParametric)
corAddGroupAndVariantChoices("fab6" "npn2"
  '("slow" "nominal" "fast")
) corAddGroupAndVariantChoices("fab6" "npn9"
  '("slow" "nominal" "fast")
) corAddGroupAndVariantChoices("fab6" "nmos8"
  '("slow" "nominal" "fast")
)```

September 2006 56 Product Version 5.1.41
corAddCorner("fab6" "slowslow"
    ?variants '(
        ("nnp2" "slow")
        ("nmos8" "slow")
        ("nnp9" "slow")
        ("nmosR" "slow")
    )
    ?nomTemp -55
)
corAddCorner("fab6" "slowfast"
    ?variants '(
        ("nnp2" "slow")
        ("nmos8" "fast")
        ("nnp9" "slow")
        ("nmosR" "fast")
    )
    ?nomTemp -55
)

The Virtuoso® Analog Corners Analysis window produced by this PCF looks like this.

Using the Multiple Parametric Modeling Style with the Spectre Simulator

When you run a multiple parametric modeling style corners analysis with the Spectre simulator, ensure that the .cdsenv variable includeStyle is set to t.

Using the Multiple Parametric Modeling Style with a Socket Simulator

To use a multiple parametric modeling style with a socket simulator, specify the parameter files in an update.s file. For example, update.s might contain
If you want to be able to override parameter declarations for specific corners, put these `use` statements in an `init.s` file instead of in an `update.s` file. Because parameters defined in the corners analysis option are processed after `init.s` parameters, you can use the corners analysis option to override the `init.s` parameters.

**Note:** Corners is designed such that it does not create a `runObjFile` in each `psf` directory it creates. Therefore, the results browser cannot be used to browse through the directories. In order to browse individual directories, you can use the *Create ROF* feature.

### Using the Virtuoso® Analog Corners Analysis Window to Define and Update Processes

As described in “Creating Process and Design Customization Files” on page 38, processes are often defined outside of the Virtuoso® Analog Corners Analysis window and then loaded when they are needed. However, you can also use the corners option graphical user interface to define, update, and save a process.

### Using the Virtuoso® Analog Corners Analysis Window to Define a Process

To define a new process,

1. Choose *Setup – Add Process*. 

```
The Add Process form appears.

<table>
<thead>
<tr>
<th>Process</th>
<th>Groups/Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Name</td>
<td></td>
</tr>
<tr>
<td>Model Style</td>
<td>Single Model Library</td>
</tr>
<tr>
<td>Base Directory</td>
<td></td>
</tr>
<tr>
<td>Model File</td>
<td></td>
</tr>
<tr>
<td>Process Variables</td>
<td></td>
</tr>
</tbody>
</table>

2. In the **Process Name** field, type the name you want to use for the new process.

3. Choose the model style you want to use for the new process.

4. Type the name of the base directory for the model file or files associated with the new process.

5. If the model style you choose in Step 3 is **Single Model Library**, type the name of the associated model file.

6. Type the names of process variables you want to add, separating them with a comma or white space. (To delete process variables, select the row and click Delete in the Virtuoso® Analog Corners Analysis window.)

7. Click **OK** in the Add Process form to close it.

**Deleting One or More Processes**

To delete a process,

1. Choose **Setup – Delete Process**.

   The Delete Process dialog box comes up.
2. Select the process you want to delete from the list of available processes and click on the right arrow button. Use the Shift key if you want to select and delete a range of processes.

![Delete Process Dialog](image)

3. Click on the OK button. A confirmation box pops up.

4. Click OK in the confirmation box.

The selected processes are deleted.

**Note:** The deletion of a process removes the related information from both the GUI and the session. If the current process is deleted, then one of the remaining processes is loaded into the Corners window. If there are none, then the window is reset.

**Using the Virtuoso® Analog Corners Analysis Window to Modify Process Model Information**

To modify an existing process,

The Add/Update Model Info form appears.

![Image of Update Process/Model Info form]

This form has two tabs: **Process** and **Group/Variants**. Only certain modeling styles allow the Groups and Variants. You need to make modifications in all tabs before adding or updating the process. Otherwise, you will have to get back in and update the process again.

1. Choose the **Process** that you want to change.
2. Choose the **Model Style** that you want to use for the changed process.
3. Type the name of the **Base Directory** for the model file or files associated with the changed process. The form appears with the process base directory.
4. If the model style you choose in Step 3 is **Single Model Library**, type the name of the associated model file.
5. Type the names of process variables you want to add, separating them with commas or spaces. (To delete process variables, select the row and click **Delete** in the Virtuoso® Analog Corners Analysis window.)
6. Click **OK** in the Update Process/Model Info form to close it.

To specify the Groups and Variants, click the **Groups/Variants** tab.

1. Specify a name in the **Group Name** field.
2. Specify the variants in the Variants field.

3. Click OK to close the form.

Requirements for Using the Spectre Simulator

When you run a single or multiple model library style corners analysis with the Spectre simulator, ensure that you comply with the following requirements.

- The value of the .cdsenv variable useAltergroup must be set appropriately. If your Spectre model supports altergroups, ensure that useAltergroup is set to t. If your Spectre model does not support altergroups, set useAltergroup to nil.

  Regardless of the useAltergroup value, the corners analysis option does not use altergroups when you run a mixed-signal (SpectreVerilog) or distributed simulation.

- Every parameter used in a corner must be in the main circuit.

Working through an Extended Example

This section follows a corners session in detail, demonstrating how you might use the corners analysis option to examine the characteristics of a real circuit. The example describes a folded cascode circuit and explains how you might arrange the supporting model files. To follow along, go to

```plaintext
your_install_dir/tools/dfII/samples/artist/corners/artistExample
```

and start icms. A .cdsinit file and the other files you need to run this example are all included at that location.
Folded Cascode Schematic

The folded cascode used in this example has the following two-part schematic.
This schematic includes several instances of pmos and nmos transistors. Each of the pmos transistors is nominally identical. Similarly, each of the nmos transistors is nominally identical. In reality, however, the attributes of each transistor differ slightly from the attributes of each of the other transistors. In this example, you explore the extent of the variation and the effect the variation has on the performance of the circuit.

**Setting Up the Virtuoso® Analog Design Environment Window**

To run this example, first set up the *Virtuoso® Analog Design Environment* window.

1. From the CIW, choose *Tools – Analog Environment – Simulation*.

   The *Virtuoso® Analog Design Environment* window appears.

2. Choose *Setup – Design*.

   The Choosing Design form appears.
3. Select the RF_lib library and the foldedCascode cell. Click OK.

4. In the Virtuoso® Analog Design Environment window, choose Session – Load State.

   The Loading State form appears.

5. Choose Corners from the State Name cyclic field. Click OK.


   The schematic window appears.

7. Click on the net connected to vout in the right side of the plot, then press the Esc key.

   There are other outputs defined in the PCF, but this demonstrates how outputs defined in the Virtuoso® Analog Design Environment window are incorporated into the Virtuoso® Analog Corners Analysis window.
The *Virtuoso® Analog Design Environment* window looks like this.

![Cadence® Analog Design Environment](image)

**Modeling Style**

This example uses the multiple model library style with the variants for pmos components defined in one file of the `multipleModelLib` directory and the variants for nmos components defined in another file of that same directory.

For example, the nmos components are defined in the file

`CORNERS/multipleModelLib/nmosLib.scs`
This file contains

```plaintext
library nmosLib
section nom
include "../nmos/typ/nmos.scs"
endsection
section fast
include "../nmos/fast/nmos.scs"
endsection
section slow
include "../nmos/slow/nmos.scs"
endsection
endlibrary
```

As implemented in this example, the parameters for the variants are not actually included in this file, although they could be. This example instead uses `include` statements to include the files that contain the actual models.

The `../nmos/typ/nmos.scs` file referred to in the `nom` section, for example, contains

```plaintext
simulator lang=spice
* VTI-derived Level=2 nominal model
.model nmos nmos level=2
  + vto = 0.775
  + tox = 400e-10
  + nsub = 8e+15
  + xj = 0.15U
  + ld = 0.20U
  + u0 = 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
```

These are the values the simulator uses when you run a corner that has the value of the `nmos` variant set to `nom`. When you run a corner that uses the `slow` variant for the `nmos` components, the simulator uses the values defined in

```plaintext
../nmos/slow/nmos.scs
```

The contents of the `../nmos/slow/nmos.scs` file are the following:

```plaintext
simulator lang=spice
* VTI Level=2 slowN/slowP model
.model nmos nmos level=2
  + vto = 0.9
```
The other variants for the nmos and pmos components are defined similarly.

Process Customization File (PCF)

This example does not use a design customization file (DCF) because all the necessary corners and measurements are defined in a single PCF called `multipleModelLib.pcf`. Defining the Virtuoso® Analog Corners Analysis window in a single file simplifies the example.

The `multipleModelLib.pcf` contains the following information.

```plaintext
corAddProcess( "multipleModelLib" "/CORNERS/multipleModelLib" 
        'multipleModelLib' )
corAddProcessVar( "multipleModelLib" "vdd" )
corAddProcessVar( "multipleModelLib" "vss" )
corAddDesignVar( "Cload" )
corAddGroupAndVariantChoices( "multipleModelLib" "pmosLib.scs" 
        '("slow" "nom" "fast")' )
corAddGroupAndVariantChoices( "multipleModelLib" "nmosLib.scs" 
        '("slow" "nom" "fast")' )
corAddCorner( "multipleModelLib" "slowslow" )
corSetCornerGroupVariant( "multipleModelLib" "slowslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "slowslow" 
        "pmosLib.scs" "slow" )
corSetCornerNomTempVal( "multipleModelLib" "slowslow" 27 )
corSetCornerRunTempVal( "multipleModelLib" "slowslow" 125 )
corSetCornerVarVal( "multipleModelLib" "slowslow" "Cload" "260f" )
corSetCornerVarVal( "multipleModelLib" "slowslow" "vss" "-2.7" )
corSetCornerVarVal( "multipleModelLib" "slowslow" "vdd" "2.7" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "nmosLib.scs" "slow" )
corSetCornerGroupVariant( "multipleModelLib" "fastslow" 
        "pmosLib.scs" "slow" )
corAddCorner( "multipleModelLib" "fastslow" )
corSetCornerGroupVari...
"pmosLib.scs" "fast"

corSetCornerNameTempVal('multipleModelLib' 'fastslow' "27")
corSetCornerNameRunTempVal('multipleModelLib' 'fastslow' "27")
corSetCornerVarVal('multipleModelLib' 'fastslow' "Cload" "200f")
corSetCornerVarVal('multipleModelLib' 'fastslow' "vss" "-3")
corSetCornerVarVal('multipleModelLib' 'fastslow' "vdd" "3")
corAddCorner('multipleModelLib' 'typtyp')
corSetCornerGroupVariant('multipleModelLib' 'typtyp' 'nmosLib.scs' 'nom')
corSetCornerGroupVariant('multipleModelLib' 'typtyp' 'pmosLib.scs' 'nom')
corSetCornerNameTempVal('multipleModelLib' 'typtyp' "27")
corSetCornerNameRunTempVal('multipleModelLib' 'typtyp' "27")
corSetCornerVarVal('multipleModelLib' 'typtyp' "Cload" "200f")
corSetCornerVarVal('multipleModelLib' 'typtyp' "vss" "-3")
corSetCornerVarVal('multipleModelLib' 'typtyp' "vdd" "3")
corAddCorner('multipleModelLib' 'slowfast')
corSetCornerGroupVariant('multipleModelLib' 'slowfast' 'nmosLib.scs' 'fast')
corSetCornerGroupVariant('multipleModelLib' 'slowfast' 'pmosLib.scs' 'slow')
corSetCornerNameTempVal('multipleModelLib' 'slowfast' "27")
corSetCornerNameRunTempVal('multipleModelLib' 'slowfast' "27")
corSetCornerVarVal('multipleModelLib' 'slowfast' "Cload" "200f")
corSetCornerVarVal('multipleModelLib' 'slowfast' "vss" "-3")
corSetCornerVarVal('multipleModelLib' 'slowfast' "vdd" "3")
corAddCorner('multipleModelLib' 'fastfast')
corSetCornerGroupVariant('multipleModelLib' 'fastfast' 'nmosLib.scs' 'fast')
corSetCornerGroupVariant('multipleModelLib' 'fastfast' 'pmosLib.scs' 'fast')
corSetCornerNameTempVal('multipleModelLib' 'fastfast' "27")
corSetCornerNameRunTempVal('multipleModelLib' 'fastfast' "-55")
corSetCornerVarVal('multipleModelLib' 'fastfast' "Cload" "160f")
corSetCornerVarVal('multipleModelLib' 'fastfast' "vss" "-3.3")
corSetCornerVarVal('multipleModelLib' 'fastfast' "vdd" "3.3")
corAddMeas('DCgain')
corSetMeasExpression('DCgain' 'yMax(db20(VF('/vout')))')
corSetMeasTarget('DCgain' 60)
corSetMeasEnabled('DCgain' t)
corSetMeasGraphicalOn('DCgain' t)
corSetMeasTextualOn('DCgain' nil)
corAddMeas('bandwidth')
corSetMeasExpression('bandwidth' 'bandwidth(VF('/vout') 3 'low')')
corSetMeasEnabled('bandwidth' t)
corSetMeasGraphicalOn('bandwidth' t)
corSetMeasTextualOn('bandwidth' nil)
corAddMeas('gain')
corSetMeasExpression('gain' 'dB20(VF('/vout'))')
corSetMeasEnabled('gain' t)
corSetMeasGraphicalOn('gain' t)
corSetMeasTextualOn('gain' nil)
corAddMeas('phase')
corSetMeasExpression('phase' 'phase(VF('/vout'))')
corSetMeasEnabled('phase' t)
corSetMeasGraphicalOn('phase' t)
corSetMeasTextualOn('phase' nil)
You can load a PCF from the *Virtuoso® Analog Corners Analysis* window, but it is often easier to insert a statement in your `.cdsinit` file that loads the necessary PCFs automatically. For example, if you look in the included `.cdsinit` file, you find the following statement that loads the `multipleModelLib.pcf` file:

```
loadPcf( "~/multipleModelLib.pcf" )
```

**Virtuoso® Analog Corners Analysis Window for Folded Cascode**

To open the *Virtuoso® Analog Corners Analysis* window,

- From the *Virtuoso® Analog Design Environment* window, choose *Tools – Corners*.

The *Virtuoso® Analog Corners Analysis* window, in this example, is defined primarily by the `multipleModelLib.pcf`. In addition, the following items affect the appearance of the window.

- The run temperature variable `temp` always appears in the *Corner Definitions* pane of the window. By default, it has the value 27. For this example, the `multipleModelLib.pcf` sets the value of the run temperature explicitly for each corner using skill function `corSetCornerRunTempVal`.

- Any outputs defined in the *Virtuoso® Analog Design Environment* window when you first start the corners analysis option appear in the *Performance Measurements* pane. That is why `/vout` appears as an expression in the *Performance Measurements* pane for this example.
When the *Virtuoso® Analog Corners Analysis* window opens, the *Corner Definitions* pane looks like this. (The *slow* corner, although not visible in this figure, also appears in the actual *Virtuoso® Analog Corners Analysis* window.)

The *Performance Measurements* pane looks like this.
The measurement that appears in the *Performance Measurements* pane is defined in the *Outputs* pane of the *Virtuoso® Analog Design Environment* window and is automatically copied into the *Virtuoso® Analog Corners Analysis* window.

### Changing Values in the Virtuoso® Analog Corners Analysis Window

So far, in this example, everything in the *Virtuoso® Analog Corners Analysis* window has been predefined, either by the PCF or because it is defined in the *Virtuoso® Analog Design Environment* window. You can also use the *Virtuoso® Analog Corners Analysis* window to revise and add to the predefined information. For example, this section describes how you might add a *Lower* value to a scalar measurement before you run the simulation.

The *DCgain* measurement produces a scalar value. To facilitate analysis, you want to add a visual indication of the lowest acceptable value to the graphical output of the corners simulation. To do that, you need to add the appropriate value to the cells in the *Performance Measurements* pane.

To add a *Lower* value by using the *Virtuoso® Analog Corners Analysis* window,

1. Click on the *Lower* cell for the *DCgain* measurement.

2. Type the value 55 in the cell.

To add a *Upper* value by using the *Virtuoso® Analog Corners Analysis* window,

1. Click on the *Upper* cell for the *DCgain* measurement.

2. Type the Value 65 in the cell.

All of the measurements produce graphical outputs if you make no further changes, but it might be useful to have the textual output too. To add the textual output,

> Turn on the *Textual* button in the *Outputs* column for each of the measurements.

### Running the Corners Simulation

After the corners and measurements are defined, running the corners simulation involves only a couple of simple steps.

1. Ensure that the corners, measurements, and outputs you want to use are selected.

2. Choose *Simulation – Run* or click *Run*.

The simulation runs and the outputs you requested appear in display windows.
Evaluating Corners Results

The graphical outputs appear in a waveform window.

The \textit{phase} and \textit{gain} measurements appear as a family of waveforms in the two subwindows at the top, with each waveform for each corner. The scalar values, \textsc{DCgain} and \textsc{bandwidth}, appear as bar charts. It is hard to pick out detail in this combined plot, but you can choose \textit{Window – Subwindows} in the Waveform Window to open a dialog box that allows you to choose which plots you want to look at in more detail.
Evaluating Residual Plots

Look first at the DCgain bar chart.

In this plot, the horizontal line in the middle represents the Target value, 60. The bottom line represents the Lower value, which you set in the Virtuoso® Analog Corners Analysis window. The top line represents the Upper value. All of the corners reach the target value. (In the actual window, each corner displays in a different color so you can determine which corner is which.) If DCgain for one or more corners fails to reach the target, you might decide to use a slightly different manufacturing process or to change your circuit so DCgain is greater.
Evaluating Family-of-Curve Plots

Now consider the family-of-curves plot for the gain measurement.

The fastfast corner produces the highest gain and the slowslow corner produces the lowest gain throughout the frequency range. You need to determine whether these possible outcomes are acceptable in your application.
Statistical Analysis

Statistical analysis is a powerful method for estimating parametric yields. The sections in this chapter explain how you can use the Analog Statistical Analysis option to generate information about the performance characteristics of the circuits you design.

- “Getting Started with Statistical Analysis” on page 77
- “Getting to Know the Analog Statistical Analysis Window” on page 80
- “Running a Statistical Analysis” on page 87
- “Analyzing Results” on page 108
- “Working through an Extended Example” on page 130

Getting Started with Statistical Analysis

This section briefly explains the theory behind statistical analysis, tells you how to get help and describes how to open the Analog Statistical Analysis window.

How Statistical Analysis Works

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.
Data Types Generated by the Statistical Analysis Tool

The Statistical Analysis tool creates two types of output data:

Scalar Data

For each iteration during a statistical analysis, the simulator evaluates explicit expressions that reduce to a single scalar number. These numbers are stored in a file which will ultimately be used for data analysis by the user at post-simulation.

The simulator evaluates these scalar expressions during runtime so as to reduce the amount of generated psf data. For each successive iteration analysis, the simulator typically deletes the psf data from the previous iteration.

However, it is possible to keep all of the psf data from all of the iterations (Spectre only; see next data type).

The majority of the statistical analysis tool UI is focused on processing and displaying scalar data.

For the Spectre® simulator, the process parameters declared in a statistics block in the netlist are also included in the resulting scalar data file.

Psf Data

This is the same kind of psf data that the simulator typically generates. Usually, this data only includes the last iteration.

However, for the Spectre® simulator, the user has the additional option to save the psf data for all of the iterations. From this data the user can either plot waveforms or regenerate new scalar data files. Since this data is psf, the user will have to evaluate expressions against this data to created these waveforms and statistical data. Specific waveform expressions are not processed during a statistical analysis, only scalar expressions are.

Opening the Analog Statistical Analysis Window

To run the statistical analysis option, you must use a simulator that supports statistical simulation. In addition, the model and device descriptions of the components that you want to use in the statistical simulations must have statistical values.

To start the statistical analysis option within the Virtuoso® analog design environment,
1. Set up your simulation normally, choosing an appropriate simulator.

2. Choose Tools – Monte Carlo.
Getting to Know the Analog Statistical Analysis Window

The Analog Statistical Analysis window contains the fields and controls required to specify the statistical analysis that you want to run.
Status Display

The status display shows messages that indicate what the statistical analysis option is doing. The messages include the following:

- Simulate
- Ready
- Plotting Results
- Simulate Distributed

During a simulation, the status display also shows which iteration is running and how many iterations are left to run.

**Note:** This feature only works outside of distributed processing mode.

Menu

The menu contains the commands needed to prepare for, run and analyze the results of a statistical analysis.

<table>
<thead>
<tr>
<th>Session</th>
<th>Outputs</th>
<th>Simulation</th>
<th>Results</th>
<th>Help</th>
</tr>
</thead>
</table>

For guidance on using the menu choices, see the associated cross references:

**Statistical Analysis Menu Choices**

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Session</strong></td>
<td></td>
</tr>
<tr>
<td>Save State</td>
<td>“Saving the Session State” on page 103</td>
</tr>
<tr>
<td>Load State</td>
<td>“Loading a Saved Session State” on page 104</td>
</tr>
<tr>
<td>Save Script</td>
<td>“Saving the Script” on page 105</td>
</tr>
<tr>
<td>Quit</td>
<td>“Closing the Analog Statistical Analysis Window” on page 106</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td></td>
</tr>
<tr>
<td>Retrieve Outputs</td>
<td>“Selecting Signals and Expressions to Analyze” on page 90</td>
</tr>
</tbody>
</table>
### Statistical Analysis Menu Choices, *continued*

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Save All</strong></td>
<td>“Saving All Voltages or Currents” on page 97</td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td></td>
</tr>
<tr>
<td>Check Expressions</td>
<td>“Checking the Output Expressions” on page 98</td>
</tr>
<tr>
<td>Define Correlations</td>
<td>“Defining Correlations” on page 100</td>
</tr>
<tr>
<td>Create Input Files</td>
<td>“Creating Input Files for a Socket Simulator” on page 103</td>
</tr>
<tr>
<td>Run</td>
<td>“Starting and Stopping the Analysis” on page 101</td>
</tr>
<tr>
<td>Stop</td>
<td>“Starting and Stopping the Analysis” on page 101</td>
</tr>
<tr>
<td>Output Log</td>
<td>“Viewing the Output Log” on page 105</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>“Filtering Outlying Data” on page 110</td>
</tr>
<tr>
<td>Specification Limits</td>
<td>“Setting Specification Limits” on page 113</td>
</tr>
<tr>
<td>Print</td>
<td></td>
</tr>
<tr>
<td>Iteration vs. Value</td>
<td>“Printing Iteration versus Value Tables” on page 116</td>
</tr>
<tr>
<td>Correlation</td>
<td>“Printing Correlation Tables” on page 118</td>
</tr>
<tr>
<td><strong>Plot</strong></td>
<td></td>
</tr>
<tr>
<td>Histogram</td>
<td>“Plotting Histograms” on page 119</td>
</tr>
<tr>
<td>Curves</td>
<td>“Plotting Families of Curves” on page 121</td>
</tr>
<tr>
<td>Scatterplot</td>
<td>“Plotting Scatter Plots” on page 122</td>
</tr>
<tr>
<td><strong>Yield</strong></td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>“Obtaining Reports on Simple Yields” on page 125</td>
</tr>
<tr>
<td>Conditional</td>
<td>“Obtaining Reports on Conditional Yields” on page 128</td>
</tr>
<tr>
<td>Multiconditional</td>
<td>“Obtaining Reports on Multiconditional Yields” on page 127</td>
</tr>
</tbody>
</table>
Statistical Analysis Menu Choices, continued

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>“Analyzing Results” on page 108</td>
</tr>
<tr>
<td>Save</td>
<td>“Saving Statistical Analysis Results” on page 102</td>
</tr>
<tr>
<td>Evaluate Expressions</td>
<td>“Creating a New mcdata File from Saved Waveform Data” on page 110</td>
</tr>
</tbody>
</table>

Help

<table>
<thead>
<tr>
<th>Contents</th>
<th>“Opening the Analog Statistical Analysis Window” on page 78</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Analog Statistical Analysis</td>
<td>“Opening the Analog Statistical Analysis Window” on page 78</td>
</tr>
</tbody>
</table>

Analysis Setup Pane

The fields and selections in the Analysis Setup pane specify the characteristics of the statistical analysis to be run by the simulator.

![Analysis Setup Pane](image)
For a brief description of the items in the Analysis Setup pane, see the following table. For more detailed information, see “Specifying the Characteristics of a Statistical Analysis” on page 88.

<table>
<thead>
<tr>
<th>Field or Selection</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Runs</td>
<td>Specify how many simulations to run for this statistical analysis.</td>
</tr>
<tr>
<td>Starting Run #</td>
<td>Specify the starting run number.</td>
</tr>
<tr>
<td>Analysis Variation</td>
<td>Select the type of statistical variation to be used.</td>
</tr>
<tr>
<td>Swept Parameter</td>
<td>If desired, select temperature or a design variable to sweep.</td>
</tr>
<tr>
<td>Append to Previous Scalar Data</td>
<td>Enable this button to append scalar output data to previously saved scalar data. This feature is not supported in the distributed processing mode. For the Spectre® simulator, all of the pertinent UI fields are checked for compatibility with the existing scalar data set prior to allowing a Monte Carlo run.</td>
</tr>
<tr>
<td>Save Data Between Runs to Allow Family Plots</td>
<td>Enable this button to save the raw output data (the parameter storage format [psf] files) for all the statistical analysis iterations. Turn this button off if you want the raw output data to be deleted before each iteration. In which case, only the psf data for the last iteration will ultimately be saved. The ability to append to previous psf data is currently not supported. This button only appears if you use the Spectre® simulator.</td>
</tr>
<tr>
<td>Save Process Parameters</td>
<td>Enable this button to save the process parameters. Depending on the selection of this check box, the settings would accordingly be written to spectre netlist. Turn this button off if you do not want to save the process parameters.</td>
</tr>
</tbody>
</table>
Outputs Pane

The Analog Statistical Analysis window Outputs pane initially lists the expressions and signals defined in the Outputs pane of the Virtuoso® Analog Design Environment window.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Expression/Signal</th>
<th>Data Type</th>
<th>Autoplot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dB20</td>
<td>dB20(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>phase</td>
<td>phase(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>bandw</td>
<td>bandwidth(VF(&quot;/OUT&quot;) 3 &quot;low&quot;)</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>ymax</td>
<td>ymax(dB20(VF(&quot;/OUT&quot;)))</td>
<td>scalar</td>
<td>yes</td>
</tr>
</tbody>
</table>

The columns in the Outputs pane are described in the following table.

**Column** | **Description and Usage**
---|---
*Name* | A field showing the existing name for the expression or signal. For expressions that evaluate to a scalar, the final name of the statistical data set will be of the form `name_varval`. Where `name` is the name on this pane and `varval` is the particular swept variable value (if none, defaults to temperature).
*Expression/Signal* | A field showing either an expression or a signal name.
**Edit Fields**

These fields, located beneath the *Outputs* pane, are used to add output signals and to add or modify output expressions.
For more information, see “Working Directly with Expressions and Signals” on page 91.

**Button Bar**

The buttons at the bottom of the Analog Statistical Analysis window operate on the Outputs pane and the edit fields.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Click to add a signal or expression defined in the edit fields to the list of signals and expressions.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Click to delete a signal or expression that is highlighted in the Outputs pane.</td>
</tr>
<tr>
<td><strong>Change</strong></td>
<td>Click to replace a signal or expression that is highlighted in the Outputs pane with whatever is in the edit fields.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Click to clear the edit fields and remove any highlighting in the Outputs pane.</td>
</tr>
<tr>
<td><strong>Calculator</strong></td>
<td>Click to open the calculator so that you can build a new signal or expression in the calculator display buffer.</td>
</tr>
<tr>
<td><strong>Get Expression</strong></td>
<td>Click to fill the Expression/Signal field with the signal or expression that is currently in the calculator display buffer.</td>
</tr>
</tbody>
</table>

**Running a Statistical Analysis**

The major steps involved in setting up and running a statistical analysis are described in the following sections:

- “Specifying the Characteristics of a Statistical Analysis” on page 88
- “Selecting Signals and Expressions to Analyze” on page 90
- “Defining Correlations” on page 100
- “Starting and Stopping the Analysis” on page 101
Specifying the Characteristics of a Statistical Analysis

You specify how a statistical analysis proceeds by filling out the fields in the top pane of the Analog Statistical Analysis window.

1. Specify the Number of Runs for this statistical analysis.

2. Specify the Starting Run #.

By default, this value is 1. However, if you want to collect the results from several sets of analyses via the Append to Previous Scalar Data boolean, each subsequent set should not have any run numbers that overlap previous runs numbers. For example, if your first analysis has a Starting Run # of 1 and the Number of Runs is 100 then the Starting Run # for the second analysis needs to be at least 101.

3. Choose the type of Analysis Variation.

The available choices depend on the simulator that you are using, but the default choices include

- Process Only
- Mismatch Only
- Process Variation and Mismatch

Which choice is most appropriate for your analysis depends on whether you want the statistically valued parameters to vary independently or to track each other. In general, the parameters of devices on the same die track each other closely and for purposes of
simulation you might want them to track exactly. In a board-level design however, the parameters of different devices are likely to vary independently. For more information, see “How the Statistical Analysis Option Uses the Analysis Variation Setting” on page 106.

4. If desired, choose a parameter to sweep in an inner loop.
   
The parameter can be either Temperature or one of the design variables. Choose None, which is the default, if you do not want to sweep a parameter.

5. Select the Append to Previous Scalar Data button if you want to append the scalar output data from the current analysis to previously saved scalar data.
   
   For example, to add another 100 runs to an existing set of 100 runs, select this button and, as discussed in Step 2, set the starting Run # to at least 101.

   By default, scalar data is saved in the monteCarlo/mcdata file located at the same level as the psf directory. If you do not select the Append to Previous Scalar Data button, new scalar data from the current analysis replaces any existing data in that file.

   (Spectre simulator only)
   
   This UI field will only be active when valid scalar results are currently selected/loaded.

   If the currently loaded scalar results are not from the current copied into the current ADE run directory when the next monte carlo simulation is run. As a result any previous existing scalar results in the run directory will be erased.

   Before allowing a simulation in this mode, the UI first checks that the current UI configuration (form field settings) are compatible with the currently loaded results. This checking includes:

   The run numbers do not overlap the run numbers in the existing data.

   The Swept Parameter declaration (including values) is the same as in the existing data.

   The scalar expressions in the output pane are the same as in the existing data.

   After a simulation in append mode, the UI checks the new data to insure that the statistical parameters swept are the same as in the pre-appended data. If any discrepancies are found, the UI blocks reading/loading in the data.

6. (Spectre simulator only) Select the Save Data Between Runs to Allow Family Plots button if you want to be able to plot graphs showing the variation of entire waveforms or if you want to evaluate expressions after the analysis finishes.

   Be aware that with this option enabled, the amount of data saved during an analysis can be very large. To reduce disk storage requirements, avoid saving all voltages and
currents. Instead, select only the specific nodes and terminals referenced by your output expressions.

The statistical analysis option calculates and saves the results of scalar expressions after every run, whether Save Data Between Runs to Allow Family Plots is selected or not.

7. Select the Save Process Parameters button if you want to save the parameters. If you select the checkbox, the settings would accordingly be written to spectre netlist.

Selecting Signals and Expressions to Analyze

The Analog Statistical Analysis window Outputs pane initially contains any expressions and signals defined in the Outputs pane of the Virtuoso® Analog Design Environment window. You can also retrieve these expressions and signals at any time by choosing Outputs – Retrieve Outputs. These choices and values from the environment window are often the most useful ones for a statistical analysis, but, if you want, you can change them. You can add, modify, or delete expressions and add or delete signals by

- Using the Direct Plot form, Add To Outputs capability
- Typing them in directly
- Using the Calculator
Working Directly with Expressions and Signals

You can use the Outputs pane, edit fields, and button bar to add or delete signals and to add, delete or change expressions.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Expression/Signal</th>
<th>Data Type</th>
<th>Autoplot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dB20</td>
<td>dB20(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>phase</td>
<td>phase(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>bandw</td>
<td>bandwidth(VF(&quot;/OUT&quot;) 3 &quot;low&quot;)</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>ymax</td>
<td>ymax(dB20(VF(&quot;/OUT&quot;)))</td>
<td>scalar</td>
<td>yes</td>
</tr>
</tbody>
</table>

Adding a Signal or Expression

To add a signal or expression by typing it in:

1. If the edit fields are not empty, click Clear.

2. In the edit fields, type a name for the new expression and the expression itself.

3. Set the Autoplot cyclic field, which is located at the right side of the edit fields, to yes if you want the new signal or expression to be plotted after all the simulations finish.

4. Click Add.

   The new signal or expression is added to the list of signals and expressions.
Deleting a Signal or Expression

To delete a signal or expression:

1. Highlight the signal or expression in the Outputs pane.
2. Click Delete.

Changing an Expression

To change an expression:

1. Highlight the expression in the Outputs pane.
   - The expression appears in the edit fields.
2. Working in the edit fields, make any necessary changes to the name, expression, and autoplot values.
3. Click Change.

Using the Calculator to Build Expressions

To avoid typing in an expression and signal names, you can build an expression in the calculator and then import the finished expression into the Analog Statistical Analysis window.

1. Click Calculator to open the calculator window.
2. Use the buttons and commands in the calculator window to build the expression you need. Leave the completed expression in the display buffer of the calculator.
3. In the Analog Statistical Analysis window, click Get Expression.

The expression appears in the Outputs pane.

Data Access Function Types

In order to create effective monte carlo expressions, knowledge of the two basic types of data access functions is critical:

Analysis Alias/Type Dependent Functions:

This type searches for a particular analysis alias or type. The alias or type to be sought after is hardcoded according to the function name, there is no specific type argument on
these functions. For example, the $\text{VF()}$ function will look for a voltage (V) signal in the first found occurrence of ac data (Frequency), whereas the $\text{IT()}$ function will look for a current (I) in transient data (Time).
Here are examples of $\text{VF()}$ expressions that can be declared:

```plaintext
mag(VF("/net2") 1e6) => waveform
value(mag(VF("/net2") 1e6)) => scalar
```

Most of these functions can be found on the calculator tool.

These wildcard functions access data according to the following flow:

a. First look for a specific aliased data name in the current results. For example, the $\text{VF()}$ function would first look for psf data aliased to the name ‘ac. If found, then this data is retrieved.

   For more information on alias names, see the section titled Data Name Aliasing.

b. If (1) does not find an alias name match, then the function reverts to a data type wildcard search operation. Instead of looking for a name match, the function will now search for the first occurrence of any psf data with a type that matches the one related to the function name. For example, the $\text{VF()}$ function will now look for a voltage (V) signal in the first found occurrence of ac type data.

Analysis Name Dependent Functions:

This type searches for a particular analysis name (or alias). These functions require that a psf data name (or its alias) be specified. The benefit to these functions is that there is no ambiguity as to which data set will be retrieved. Here are examples of ac data expressions that can be declared:

```plaintext
mag(v("/net2" ?result "ac") 1e6) => waveform
value(mag(getData("/net2" ?result 'ac) 1e6)) => scalar
```

All of the Ocean data access functions are name dependent. The most commonly used functions are $\text{getData()}$, $\text{v()}$ and $\text{i()}$.

Data Name Aliasing

During the course of using the Statistical Analysis tool, the spectre simulator can create up to nine distinct data names for each analysis type. The name of data being accessed depends both on the UI configuration for the statistical analysis run and on when the data is being accessed (by spectre at runtime or by the UI for post processing/plotting).

As a result, a method to declare a single data name that was the equivalent to the nine above mentioned data name formats was added. Now, when data is selected/loaded, the data
access code automatically aliases any found version of the nine possible data name formats to the base name equivalent.

For any currently selected/loaded data the user can see what alias names have been assigned by typing the following command into the ciw:

```
setof(ana results() symbolp(ana))
```

An example response is:

```
(dc ac output dcOp sp_noise model tranOp tran
dcOpInfo instance noise sp variables)
```

To see the actual psf names generated, enter:

```
setof(ana foreach(mapcar (na a) results(?noAlias t) results()
    list(a na)) symbolp(car(ana)))
```

An example response is:

```
(dc "mcl_dc-montecarlo")
(ac "mcl_ac-montecarlo")
(output "mcl_outputParameter-montecarlo")
(dcOp "mcl_dcOp-montecarlo")
(sp_noise "mcl_sp_noise-montecarlo")
(model "mcl_modelParameter-montecarlo")
(tranOp "mcl_finalTimeOP-montecarlo")
(tran "mcl_tran-montecarlo")
(dcOpInfo "mcl_dcOpInfo-montecarlo")
(instance "mcl_element-montecarlo")
(noise "mcl_noise-montecarlo")
(sp "mcl_sp-montecarlo")
(variables "variables")
```

In this example, we know that the "mcl_ac-montecarlo" psf data will be accessed for either of the following functions:

```
VF("/net2")
getData("/net2" ?result 'ac)
```

So long as all functions which are analysis name dependent use the proper alias name, the proper data will be retrieved throughout all phases of using the Statistical Analysis tool.

Note that this aliasing is tightly tied to the analysis names that the ADE netlister generates. If the user tries to run a standalone netlist with different names, the aliasing may not work properly. The same goes for declaring analyses via an include file. The Statistical Analysis UI is only intended for use directly within the ADE product.
Results Dir Arguments in Expressions

Many of the data access functions have an optional results directory argument. This argument must not be set for any expression being used in monte carlo. For more information, see the Creating Monte Carlo Compatible Expressions section.

Creating Monte Carlo Compatible Expressions

Most of the calculator buttons produce valid expressions for use in the monte carlo flow (this does not include the results browser button or tool).

Most of the results browser generated data access expressions must be hand edited in order to conform to the expression syntax requirements of the monte carlo flow.

Most of the ADE->Results->Direct_Plot forms produce the proper expression syntax (via the Add to Outputs utility). These forms typically produce ocean expressions, but some still produce old ael style functions.

Before any function/expression will work properly in the Monte Carlo tool, the following Monte Carlo Expression Syntax Rules should be followed:

1. Whenever an analysis name argument is required, like for Ocean functions, always use the proper alias name.

2. Never include the current run directory in an expression. When using the results browser to capture a data access expression in the calculator, always manually delete the run directory component of the expression.

For example, the results browser may declare something like this in the calculator:

```
v( "/net4" ?result "ac-ac" ?resultsDir
"/hm/test/simulation/ampTest/spectre/schematic" )
```

This expression must be manually altered to look like:

```
v( "/net4" ?result 'ac)
```

Where `ac` is the standard ac analysis alias name.

For more information on alias names, see the section titled Data Name Aliasing.

The most common symptoms of invalid expressions are:

- All of the generated scalar data points for an expression is the same nominal value.
- A waveform plot for a waveform expression only plots a single waveform and there is no iterations parametric information.
Optimizing Monte Carlo Expressions

Some expressions can require significant evaluation times for the Simulation- >Check_Expressions and Results->Evaluate_Expressions capabilities of the UI. The bigger the family of psf data created by spectre, the greater the evaluation times of these two utilities. The analyses that are mostly affected are tran, sp and the rf ones.

The spectre simulator does not experience this slowdown in performance because it only operates on a single iteration at any one time. It never processes the entire psf family of iterations data.

Depending on the scope of psf data generated, the user can get better performance out of an expression if they simply re-arrange it.

For example, the following syntax:

\[
\text{value(exprA num)/value(exprB num)}
\]

is faster than using the simpler expression syntax of:

\[
\text{value(exprA/exprB num)}
\]

Some functions with complicated internal calculations compound the amount of time required for evaluation. A prime example is the s-parameter yp() Ocean function. This function requires performing internal matrix manipulations. As the data sets that yp() operates on grow linearly, the matrix evaluation time grows exponentially. As a result, it is best to reduce the data going into the yp() function as opposed to reducing the data coming out of it. For example, although the following expression is simplistic:

\[
\text{value(yp(sp(1 1 \text{ ?result } \text{'sp}) sp(1 2 \text{ ?result } \text{'sp}) }
\text{ sp(2 1 \text{ ?result } \text{'sp}) sp(2 2 \text{ ?result } \text{'sp})) 1M)}
\]

The user will get much better performance if they re-arrange the expression to be:

\[
\text{yp(value(sp(1 1 \text{ ?result } \text{'sp} 1M) value(sp(1 2 \text{ ?result } \text{'sp} 1M))}
\text{ value(sp(2 1 \text{ ?result } \text{'sp} 1M) value(sp(2 2 \text{ ?result } \text{'sp} 1M))}
\]

**Note:** The YP() ael function is doomed to be slow because it does not allow the user to reduce the dimension of the input s-parameters.

Therefore, it depends on how far the customer wants to go to optimize their expressions. For transient and s-parameter data, It is highly recommend that they optimize as much as absolutely possible.
Saving All Voltages or Currents

To save all of the node voltages and terminal currents for later use, perform the following,

1. In the Analog Statistical Analysis window, choose Outputs – Save All.

   - In the Analog Statistical Analysis window, choose **Outputs – Save All**.

2. Select appropriate voltages, currents, or both and click **OK**.

To reduce the amount of disk space required, select the currents and voltages that appear in your output expressions and consider selecting individual nets or nodes. For more information about the Save Options form, see the [Virtuoso Analog Design Environment User Guide](#).

**Note:** Be aware of the following information about cdsSpice simulations:

- In cdsSpice simulations, data for terminals in lower-level schematics is not saved when you use the Save Options form to save all currents. You must explicitly select
each terminal with the Outputs – To Be Saved – Select On Schematic command from the menu in the Virtuoso® Analog Design Environment window.

In cdsSpice simulations that include a noise analysis, the system turns the Select all node voltages and Select all terminal currents options off. If you later deactivate the noise analysis, the system reactivates the options.

3. Select the kinds of information that you want the analog design environment to print.

Turning off the printing improves the performance of a statistical analysis.

Checking the Output Expressions

After all the desired output expressions are declared, the user should execute the following utility prior to starting the statistical analysis run:

▶ Simulation -> Check Expressions

This capability is used to both check and classify each expression element in the Outputs Pane. Each expression is evaluated against the currently selected/loaded psf data. This data can either be from a previous statistical analysis or from a straight Virtuoso® Analog Design Environment simulation. The result of each evaluation is assigned to the Data Type field on the Outputs Pane. The three possible outcomes are:

- **scalar** the expression evaluated to a scalar.
- **waveform** the expression evaluated to at least a waveform.
- **ERROR** the expression failed to evaluate.

If no psf data is currently selected/loaded, then a message is produced.

Saving Signals Used in Output Expressions

For any expression entered into the output pane, it is the users responsibility to ensure that all the pertinent schematic signals contained in all the expressions will be written to the psf data. The psf data is always used to calculate the scalar values for each iteration run.

In the event that the user will want to produce waveform plots (see the Plotting Families of Curves and Changing Waveform Expressions at Post-simulation Time sections) or post generate the scalar data (see the Creating a New mcd data File from Saved Waveform Data section) then it is a good practice to save any psf schematic signals that will be needed on a post-simulation basis.
There are two basic ways to declare which schematic signals should be written to psf by the simulator:

1. Use the ADE->Outputs->To_Be_Saved->Select_On_Schematic menu capability. This approach allows the user to click directly on the schematic signals they desire.

   **Note:** This approach is best suited when the user wants to limit the amount of disc space used for psf data.

2. Use the Monte Carlo Outputs->Save_All form. This is the same form as the ADE one.

```
<table>
<thead>
<tr>
<th>Save Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
</tr>
<tr>
<td>-----</td>
</tr>
</tbody>
</table>

- **Select signals to output (save)**
  - none
  - selected
  - ivl
  - ivl
  - all
  - pub
  - all

- **Select power signals to output (pwr)**
  - none
  - total
  - devices
  - subckts
  - all

- **Set level of subcircuit to output (nestlvl)**

- **Select device currents (currents)**
  - selected
  - nonlinear
  - all

- **Set subcircuit probe level (subcktprobeivl)**

- **Select AC terminal currents (useprobes)**
  - yes
  - no

- **Select AHDL variables (saveahdlvars)**
  - selected
  - all

- **Save model parameters info**

- **Save elements info**

- **Save output parameters info**

The options displayed in the form depend on the simulator you use.
Defining Correlations

Often circuit components are correlated. You can model this behavior by defining correlations in the model files that you use. As described in this section, you can also use the statistical analysis option to define correlations.

1. To define correlations, choose *Simulation – Define Correlations*.

   The Statistical Device Correlations form appears.
2. Define the correlations that you need.

<table>
<thead>
<tr>
<th>To type device names...</th>
<th>To select devices in the schematic...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Type the full schematic names into the field near the bottom of the form.</td>
<td>a. Click Select on Schematic.</td>
</tr>
<tr>
<td>b. Type the correlation coefficient for those devices into the field to the right of the names.</td>
<td>b. In the Composer window, select the devices that you want to be correlated.</td>
</tr>
<tr>
<td>c. Click Add.</td>
<td>c. When you finish selecting, press the Escape key.</td>
</tr>
<tr>
<td>d. In the Statistical Device Correlations form, type the correlation coefficient for the devices you selected into the field to the right of the names.</td>
<td>d. In the Statistical Device Correlations form, type the correlation coefficient for the devices you selected into the field to the right of the names.</td>
</tr>
<tr>
<td>e. Click Add.</td>
<td>e. Click Add.</td>
</tr>
</tbody>
</table>

3. When you have defined all the correlations, click OK to close the Statistical Device Correlations form.

The statistical analysis option adds the correlations to the netlist by inserting a new statistics block after the statistical analysis definition.

Starting and Stopping the Analysis

To start the statistical analysis, choose Simulation – Run.

Normally, the analysis stops when all the iterations are complete. In addition, analyses that use the Spectre simulator stop if errors are found during the nominal simulation that the Spectre simulator performs.

If you want to stop the analysis before all the iterations are complete, choose Simulation – Stop. In response, the statistical analysis option

1. Completes the current iteration
2. Stops the analysis
3. Saves the simulation results
4. Plots the results for each signal or expression that has Autoplot set
Saving Statistical Analysis Results

After a statistical analysis, the resulting psf and statistical scalar data can be saved.

1. Choose Results – Save.

   The Save Results dialog box appears.

2. Type new information, as necessary.

   Consider entering a comment to help you identify the data later.

   Be aware that this command copies the entire parameter storage format (PSF) directory structure, which might be very large, to the new location.

3. Click OK.
Saving and Restoring a Statistical Analysis Session

This section explains how to create input files, save and reload session states, save scripts, and quit from the statistical analysis option. For information on starting the tool, see “Opening the Analog Statistical Analysis Window” on page 78.

Creating Input Files for a Socket Simulator

If you follow the usual procedure of specifying the simulation and then specifying the statistical analysis that you want to use, the statistical analysis option creates the necessary input files automatically. However, if you want to hand edit the input files, the statistical analysis option provides a way for you to do that.

➤ From the Analog Statistical Analysis window, choose Simulation – Create Input Files.

   The tool creates the mcrun.s and mcparam files.

Saving the Session State

A session state consists of all the information in the Analog Statistical Analysis window, including the Analysis Setup and Outputs information.

To save the current state,

1. Choose Session – Save State.

   The Monte Carlo Save form appears.

   ![Monte Carlo Save Form](image)

2. Type the name of the file where you want to save the session state.
3. Click **OK**.

### Loading a Saved Session State

To load a saved state,

1. Choose **Session – Load State**.
   
   The Monte Carlo Load form appears.

2. Type the name of the file that contains the saved session state.

3. Click **OK**.
   
   The values from the saved session state appear in the **Analog Statistical Analysis** window.

### Environment Variables

You can specify the default directory and default filename that should appear on the save Monte carlo form. The same can be specified using two new environment variables:

- `asimenv.monte mcStateDir string "" nil`
- `asimenv.monte mcStateFile string "" nil`

You can set these variables in the .cdsenv file. If you set the above variables, the state files will be saved in an independent location so that they are available for reuse. Otherwise, the form will load the default settings and the Monte Carlo states will be saved in:

- **Directory**: `<projectdirectory>/<design>/<simulation>/<view>/netlist`
- **File**: `.mcState.il`

**Note:** When you open the Monte Carlo Load/Save state form, it gets values from the cdsenv or the default values. You can edit the values before loading/saving the state form.
Saving the Script

To save the script that is automatically generated during each session,

1. Choose Session – Save Script.
   
The Monte Carlo Save Ocean Script form appears.

2. Type the name of the directory and file where you want the script to be saved.

3. Click OK.
   
The script is saved in the file. For additional information about using the statistical analysis option with OCEAN, see the OCEAN Reference.

Viewing the Output Log

To open a window that contains the history of the statistical analysis simulations,
Choose Simulation – Output Log.

The simulator updates the Output Log while the simulation runs, so you might find it useful to have this window open during the simulation.

Closing the Analog Statistical Analysis Window

To end the session and close the Analog Statistical Analysis window,

Choose Session – Quit.

How the Statistical Analysis Option Uses the Analysis Variation Setting

When you run the statistical analysis option from the Virtuoso® Analog Design Environment window, you use the Analysis Variation cyclic field to select the kinds of variations to be used during the analysis. However, the connection between the cyclic field choice and how your models behave is not automatic—you must define your models so that they respond to the cyclic field choice.

For the Spectre simulator, you use a statistics block to specify model behavior under the Analysis Variation cyclic field. See the Analysis Statements chapter of the Spectre Circuit Simulator Reference for more information. The statistics block must be included in the netlist before the models. For example, you might use a statistics block with the following contents:

```plaintext
statistics {
  process {
```

...
Virtuoso Advanced Analysis Tools User Guide
Statistical Analysis

\begin{verbatim}
vary PiRho dist=gauss std=250
vary PbRho dist=gauss std=40
vary beta dist=gauss std=20
vary rin1 dist=gauss std=10
vary cin dist=gauss std=20p
vary rin2 dist=gauss std=100
vary cloop dist=gauss std=16p
vary rout1 dist=gauss std=30
vary rout2 dist=gauss std=50
}
mismatch {
  vary stat dist=gauss std=.01
}
\end{verbatim}

The \texttt{process} and \texttt{mismatch} blocks define the variations for the cyclic field choices.

Before the simulator can use the statistics block, you must include the block in the netlist. For example, assume that you have a single file called \texttt{statsLib.scs} that has three sections: \texttt{parameters}, \texttt{statistics}, and \texttt{models}. These sections are arranged so that the definitions all build without errors.

Then, to include \texttt{statsLib.scs} in the netlist,

1. From the Virtuoso® Analog Design Environment window, choose \textit{Setup – Model Libraries}.

   The Model Library Setup form appears.

2. Type the complete path to the \texttt{statsLib.scs} file and click \textit{Add}.

3. Click \textit{OK}.
Analyzing Results

You can use the procedures described in the following sections to analyze a set of statistical analysis results. By default, these procedures operate on the data from a just-concluded statistical analysis, but you can also analyze saved data from earlier runs. The saved data can take two forms: either you can use the data in a stored output file or, if the statistical analysis session that you want to analyze had the *Save Data Between Runs to Allow Family Plots* button turned on, you can create a new mcdata output file from the saved waveform data.

Loading Stored Statistical Analysis Results

To load a stored set of data,

1. Choose *Results – Select*
The Select Results form appears.

2. Select the results that you want to load and click OK.
3. Verify the names of the files containing the statistical analysis data. At runtime, the UI automatically assigns the name `mcparam` for the Parameter File and `mcdata` for the Data File. As a result, there are few situations where a user should try to stray from these names. If the names appear to be proper, then click `OK`. If either of the file names are blank, then the data is not present and the user should click `Cancel`.

**Creating a New `mcdata` File from Saved Waveform Data**

If you select `Save Data Between Runs to Allow Family Plots` before running a simulation, then after the simulation you can use the data stored by that command to evaluate expressions and create a new `mcdata` file.

1. Ensure that the `Outputs` pane contains the expressions and signals that you want to use.
2. Choose `Results – Evaluate Expressions`.

**Filtering Outlying Data**

When it cannot evaluate an expression, a socket simulator returns the value `1e36`. The Spectre simulator returns `-1.1111e36` when it cannot evaluate an expression and returns `--2.2222e36` when an expression evaluates to a waveform instead of a scalar. Outlying data points with values like these can have a large and misleading effect on a statistical analysis. To avoid distortions, you can follow steps like the ones below to filter outlying data points from your data set.
1. Choose Results – Filter.

The Data Filter form appears.

2. Check that Data Filter is set on.

3. Choose how to compute yield statistics.

   - Filter By data set ignores all measurements for a point if the value of any of the measurements for that point is outside the filter limits.

     For example, if a point has a value of 1e36 on the bandw_27 measurement shown in the previous figure, the value for the ymax_27 measurement for that point is also ignored even if the value falls between the upper and lower ranges defined by the filter.

   - Filter By point filters an outlying point only from the specific measurement that recorded the outlying point.

4. Choose how to set the limits for each parameter.

   - Set By sigma lets you specify how many standard deviations around the mean value to include.

   - Set By limits lets you set absolute upper and lower values. The Upper and Lower values are included in the range of acceptable values; so to exclude an error value of 1e36, you need to specify a smaller value, such as 1e35.
Turning Off Filtering

To turn off data filtering,

1. Choose Results – Filter.
2. In the Data Filter form, set Data Filter to off and click OK.

Saving and Restoring Filter Settings

Whenever new data is selected/loaded into the Monte Carlo UI, the existing Data Filter form and its settings are destroyed. A new form is recreated based on the new scalar data information being read in. As a result, any and all settings on the previous Data Filter form will be lost. Therefore, it is highly recommended that any declarations made on this form be immediately saved. Especially if one will be running subsequent simulations of selecting or loading in results.

To save the data filter settings to a file,

1. Choose Results – Filter.
2. In the Data Filter form, click Save.
   
   The Save Data Filter Values form appears.

   ![Save Data Filter Values](image)

3. Type a name for the settings file and click OK.

To restore saved data filter settings,

1. Choose Results – Filter.
2. In the Data Filter form, click Load.
The Load Data Filter Values form appears.

3. Type the name of the settings file and click OK.

**Setting Specification Limits**

The specification limits define, for each parameter, the range that is considered to be within tolerance.

To set the specification limits,


   The Specification Limits form appears.
2. Choose how to set the specification limits for each parameter.

- *Set By sigma* lets you specify how many standard deviations around the mean value to allow.
- *Set By limits* lets you set absolute upper and lower values.

**Saving and Restoring Specification Limits**

Whenever new data is selected/loaded into the Monte Carlo UI, the existing *Specification Limits* form and its settings are destroyed. A new form is recreated based on the new scalar data information being read in. As a result, any and all settings on the previous *Specification Limits* form will be lost. Therefore, it is highly recommended that any declarations made on this form be immediately saved. Especially if one will be running subsequent simulations of selecting/loading in results.

To save and restore the specification limits to a file,

1. Choose *Results – Specification Limits*.
2. In the Specification Limits form, click *Save*.
   
   The Save Specification Limits form appears.

3. Type a name for the limits file and click *OK*.

To restore specification limits,

1. Choose *Results – Specification Limits*.
2. In the *Specification Limits* form, click *Load*.
The Load Specification Limits form appears.

3. Type the name of the settings file that you want to load and click OK.

Generating Plots, Tables, and Reports

To help analyze your results, you can generate the following plots, tables, and reports for your input and output parameters.

Plot, Table, or Report | Description | For More Information
--- | --- | ---
Iteration Versus Value | A table showing the value of a parameter at the end of each iteration. | “Printing Iteration versus Value Tables” on page 116
Correlation | A table showing the correlation coefficients of each parameter with each of the other parameters. | “Printing Correlation Tables” on page 118
Histogram | A plot showing the number of runs with scalar parameter values that fall in each range of values. | “Plotting Histograms” on page 119
Family-of-Curves | A plot showing the superimposed waveforms for all iterations of a waveform valued expression. | “Plotting Families of Curves” on page 121
Scatter Plot | A plot depicting the relationship between pairs of parameters. | “Plotting Scatter Plots” on page 122
Simple Yield | A report showing the individual and total yields for all parameters, given the specification limits. | “Obtaining Reports on Simple Yields” on page 125
Understanding Generated Data Names

The output expression names revealed when viewing statistical results will be slightly different than that shown on the Outputs Pane of the UI.

This is because each scalar expression could be evaluated across several swept parameter values (a UI capability), there would be several distinct data sets created. In order to cope with having many data sets for a single output expression, the generated parameter name for each of these data sets is comprised of the original expression name and the value of the swept parameter. Each name will be of the form:

    Name_ParamValue

Where Name is the name assigned on the Outputs pane of the UI and ParamValue is the particular value of the swept parameter used while that data was generated.

If the Swept Parameter on the UI is set to None, then the value for ParamValue is set to the temperature value.

**Note:** In Spectre, The parameters declared in a statistics block of the spectre netlist will also be written to the statistical results. The naming convention for these parameters is the same as for expressions, where Name is the name of the statistically varied parameter.

Printing Iteration versus Value Tables

To print a table showing the value of a parameter at each iteration,

1. Choose *Results – Print – Iteration versus Value.*
The *Iteration Versus Value* window appears.

2. Select a parameter, or type a parameter name.

3. Choose the output format:
   - *sorted* lists the runs by parameter value
   - *unsorted* lists the runs in chronological order
For example, the following figure illustrates a sorted run, with the value and run number for each measurement listed horizontally across the window.

### Printing Correlation Tables

A correlation table shows the correlation coefficients of each parameter with each of the other parameters. The parameters are sorted from most correlated to least correlated for each combination of parameters.

To print a correlation table,

1. Choose Results – Print – Correlation Table.

   The Correlation Table window appears.

2. Specify a minimum correlation value. Pairs of parameters with correlations lower than this value do not appear in the table.

3. Click OK.
The Results Display Window appears.

Each row lists the pair of measurements being considered, the mean and standard deviation of the first measurement, the mean and standard deviation of the second measurement, and the number of data points included in the calculation.

### Plotting Histograms

You can plot four types of histograms:

- **Standard**
- **Pass/fail**
- **Cumulative line**
- **Cumulative box**

To plot a histogram,

1. Choose *Results – Plot – Histogram.*
The *Histogram* form appears.

2. Highlight one or more parameters and click *Add*.
   
   You can drag or *Shift-click* to select a group of adjacent parameters or *Control-click* to select individual parameters.

3. Type a value from 1 to 50 in the *Number of Bins* field.

4. (Optional) Click *Density Estimator* to plot a curve that estimates the distribution concentration.

5. Click *OK*.
The Waveform window appears, showing the distribution of parameter values found during the statistical analysis run. In this example, the curved line is the density estimator line.

Plotting Families of Curves

A family-of-curves shows the superimposed waveforms generated during all of the statistical analysis iterations. This kind of plot illuminates the variability introduced in waveform variables by process and mismatch variations.

In order to plot such curves, the *Save Data Between Runs to Allow Family Plots* button in the Monte Carlo Analysis Setup Pane must be turned on prior to running the simulation.

To plot a family-of-curves,

➤ Choose *Results – Plot – Curves.*
The Waveform window opens with the overlapping waveforms. Depending on the number of iterations included, you might or might not be able to read the legend that identifies each individual waveform.

**Plotting Scatter Plots**

A scatter plot shows the relationship between pairs of parameters.

To plot a scatter plot,

1. Choose *Results – Plot – Scatterplot.*
The ScatterPlot form appears.

2. Highlight one parameter in the X-axis list and one in the Y-axis list.

3. Click Add.

4. (Optional) Repeat steps 2 and 3 for other parameter pairs.

5. (Optional) Click Best Fit Line to draw least squares fit lines on each scatter plot.

6. Click OK.
The Waveform window appears with the values for one parameter of each pair on the vertical axis and the values for the other parameter on the horizontal axis.

Obtaining Reports on Yields

You can compute simple, conditional, and multiconditional yield statistics.

**Note:** The yield calculations represent parametric yields only and do not include yield reduction due to defect density or packaging factors.
The table below gives an example of a yield calculation. The next few sections use this table to illustrate the different kinds of yield statistics. In this table, pass or fail indicates whether that sample passed or failed the specification limit for that parameter.

Table 2-1  Yield Calculation Table Example

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>pass</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>s2</td>
<td>pass</td>
<td>fail</td>
<td>pass</td>
</tr>
<tr>
<td>s3</td>
<td>fail</td>
<td>fail</td>
<td>pass</td>
</tr>
</tbody>
</table>

*Individual yield* is the percentage of samples that meet the current specification limits for each individual parameter. For example, if parameter X has 100 samples of which 80 fall within the specification limits, the individual yield is 80%. Using the values in Table 2-1 on page 125, the individual yields are as follows: p1, 66%; p2, 33%; and p3, 100%.

*Total yield* is the percentage of samples that meet the current specification limits for all parameters. In Table 2-1 on page 125, of the three samples only sample s1 falls within the specification limits for all parameters. The total yield, therefore, is 33%.

*Multiconditional yield* is the individual yield when only the samples for the specified parameters that are within their respective specification limits are used in the yield calculation. If the parameter fails the specification limit test, that sample is removed from the yield calculation. In the example table, the multiconditional yield for p1 with sample s3 removed is as follows: p1, 100%; p2, 50%; and p3, 100%.

*Conditional yield* is the same as the multiconditional yield except the individual yields are calculated for each parameter separately.

**Obtaining Reports on Simple Yields**

The simple yield report shows the individual and total yields for all parameters given the specification limits. A summary line shows

- Total yield
- Product of the individual yields
- Total sample size

To print a simple yield report,

1. Choose *Results – Yield – Simple*. 
The *Simple Yield* form appears.

2. (Optional) Type a percentage to filter out statistics for samples with high yields.

3. Click *OK*.

   The *Results Display Window* appears.

   In this report, $y_{indv}$ stands for individual yield.
Obtaining Reports on Multiconditional Yields

The multiconditional yield report shows the individual and total yields when the parameters you select pass the specification limits test. The subset of all data sets that meet these specifications is determined, and the yield is calculated from only this subset.

To print a multiconditional yield report,

1. Choose Results – Yield – Multiconditional.

   The Multi Conditional Yield form appears.

2. Select parameters by double-clicking in the list.

3. (Optional) To omit cases where the individual and simple yields are similar, type a percentage in the Suppress field.

4. Click OK.
The Results Display Window appears.

In this report, \textit{condlyd} stands for conditional yield.

\textbf{Obtaining Reports on Conditional Yields}

The conditional yield report is similar to the multiconditional yield report, except that the yield is calculated for each parameter separately. This allows you to quickly view the effects of conditional yield for several parameters in a single command.

To print the conditional yield report,

\begin{enumerate}
  \item Choose \textit{Yield – Conditional}.
\end{enumerate}
The Conditional Yield form appears.

2. Select parameters by double-clicking in the list.

3. (Optional) To omit cases where the individual and simple yields are similar, type a percentage in the Suppress Printout for Delta Yields Less Than field.

4. Click OK.

   The Results Display Window appears.
Working through an Extended Example

This section follows a statistical analysis session in detail, demonstrating how you might use the Analog Statistical Analysis option to examine the characteristics of a lowpass filter. The sections explain how to arrange supporting files for the Spectre simulator, run the analysis and analyze the results.

Lowpass Filter Schematic

The circuit used in this example allows low-frequency signals to pass through and attenuates high-frequency signals. The capacitor values control the attenuation of the circuit, while the resistor values control the voltage gain of the signals that pass through the circuit.

The lowpass filter has the following top-level schematic.
Descending into the amplifier shows that it has the following schematic.

![Amplifier Schematic](image)

This amplifier schematic includes several instances of npn and pnp transistors. Each of the npn transistors is nominally identical. Similarly, each of the pnp transistors is nominally identical. In reality, however, the attributes of each transistor differ randomly from the attributes of each of the other transistors. In the following sections of this example, you explore the effect that random variation in transistors has on circuit performance.

To follow along with this example, go to a working directory and use a command like the following to copy all the contents of the `monteCarlo` directory into the working directory.

```
tar -cvhf -C <install_dir>/tools/dfII/samples/artist monteCarlo | tar -xvf -
```

Then go to the working directory you created, start `icms`, and continue with the following steps.

1. In the CIW, choose **Tools – Analog Environment – Simulation** to open the **Virtuoso® Analog Design Environment** window.

2. In the **Virtuoso® Analog Design Environment** window, choose **Setup – Design**. When the **Choosing Design** form appears, select the `aExamples` library and the `lowpass` cell. Click **OK**.
3. In the Virtuoso® Analog Design Environment window, choose Session – Load State to open the Loading State form.

4. In the Loading State form, select monte as the State Name that you want to load. Click OK.

At the end of this series of steps, the Virtuoso® Analog Design Environment window looks like this.
Model File

This example uses a model file called spectreLib.scs, which contains all the parameters, statistics, and modeling information that are required.

library monteLib
section param
simulator lang=spectre
parameters PiRho=2500 PbRho=200 npnbeta=145.5 pnpbeta=200
parameters rin1=1000 rin2=5000 rout1=1000 rout2=3000
parameters cin=1.7e-08 cloop=1e-09
parameters mmstat=1 initstat=1
function Rpb(l,w)=(PbRho*1/w)
function Rpi(l,w)=(PiRho*1/w)
endsection param
section stats
simulator lang=spectre
statistics {
  process {
    vary PiRho dist=gauss std=350
    vary PbRho dist=gauss std=50
    vary npnbeta dist=lnorm std=.9
    vary pnpbeta dist=lnorm std=1.1
    vary Rin dist=gauss std=150
    vary cin dist=gauss std=20p
    vary rin2 dist=gauss std=100
    vary Cfb dist=gauss std=.58n
    vary rout1 dist=gauss std=30
    vary rout2 dist=gauss std=50
  }
  mismatch {
    vary PiRho dist=gauss std=19
    vary PbRho dist=gauss std=3.75
    vary npnbeta dist=gauss std=4
    vary pnpbeta dist=gauss std=6
  }
}
endsection stats
section models
simulator lang=spectre
inline subckt npn (C B E S)
parameters brvbe=.6
model mynpn bjt type=npn is=5.77le-17 bf=npnbeta nf=0.9895 vaf=201.6
+ ikf=0.01573 ise=8.976e-18 ne=1.179 br=3.204 nr=0.9944
+ var=27.03 ikr=0.0003047 isc=1.505e-13 nc=1.912 rb=8.706
+ irb=0.001509 rbm=5.833 re=111.8 rc=54.97 xtb=1.5 eg=1.11
+ xti=3 cje=1.983e-12 vje=0.4818 mje=0.2486 tf=0.33e-9
+ xtf=4.359 itf=0.01753 ptf=176.2 cjc=1.749e-12 vjc=0.5989
+ mjc=0.3349 xcjc=0.5 tr=400e-9 cjs=1e-12 vjs=0.75
+ mjs=0.33 fc=0.5 bvbe=brvbe bvce=1
npn (C B E S) mynpn
ends npn
inline subckt pnp (C B E S)
model mypnp bjt type=pnp
+is=1.2e-16 bf=pnpbeta nf=1.00 vaf=26.00
+ikf=70e-06 ise=1.1e-15 ne=2.00 br=13
+nr=1.00 var=10.00 ikr=100e-06 isc=7.0e-15
+nc=2.50 rb=100
+re=15 rc=150 cje=33e-15 vje=740e-03
+mje=330e-03 tf=2.50e-09 xtf=1.00
+itf=2.00e-03 ptf=5.0 cjc=130e-15 vjc=690e-03
+mjc=440.00e-03 xcjc=500.00e-03 tr=5.00e-09 cjs=200e-15
+vjs=590e-03 mjs=440.00e-03 xtb=780e-03 eg=1.200
+xii=1.80 kf=1.60e-15 af=1.00 fc=850.00e-03
pnp (C B E S) mypnp
ends pnp
endsection models
endlibrary monteLib

Notice the lines in the models section of the model file that define the mynpn model.

parameters brvbe=.6
model mynpn bjt type=npn is=5.771e-17 bf=npnbeta nf=0.9895 vaf=201.6

In particular, notice how the bf parameter is defined as npnbeta. The npnbeta value varies randomly according to the distributions specified in the statistics block. Consequently, the value of the bf model parameter also varies. So that mismatch, which is also specified for this parameter, is effective, the model is defined within an inline subckt block. This allows each instance of the model to have a slight perturbation.

The statistics block defines how parameters vary during the analysis. In this case, each parameter has either a Gaussian or a log-normal distribution with a deviation specified by the std parameter. All the parameters vary when process variation is specified and four of them vary when mismatch is specified.

**Run Analog Simulation to Check Setup**

Run a simulation via the ADE->Simulation->Run banner element.

This part of this example is used to verify that the ADE setup is correct. Note that the db20 and phase outputs created plots and that the bandwidth and ymax outputs evaluated to numbers.

In addition, the psf data created by this run will be used later for expression checking on the Analog Statistical Analysis UI.
Specifying the Analysis in the Analog Statistical Analysis Window

At this point in the example, you are ready to use the statistical analysis option.


2. In the Analog Statistical Analysis window, choose Session – Load State. The Monte Carlo Load dialog appears. Change the Directory entry to ./.

3. Ensure that the File to be loaded is .mcState.il. The Monte Carlo Load dialog should look like this.

4. Click OK.
5. Make sure the Analysis Setup pane is set correctly. It should look like this:
The Analysis Setup form is set to perform 100 iterations of the circuit, using both process and mismatch variations. Turning on *Save Data Between Runs to Allow Family Plots* makes it possible to plot families of curves for the two waveform expressions, *phase* and *db20*.

**Note:** The state loaded contained the identical Outputs Pane expressions as was initially copied over from the ADE Outputs section. Had the state contained different expressions, then they would have been merged with any existing expressions. The original *bandwidth* output name was changed to *bandw_27* by the state. However, the actual expression formulation remained unchanged.

Any unwanted Outputs pane expressions can be deleted.

**Checking Expressions Prior to Simulation**

Whenever expressions have been added, loaded or changed, it is a good practice to check them prior to submitting a statistical analysis simulation.

In this example, we will use the ADE simulation data created in a previous step. Invoke *Simulation->Check_Expressions* utility of the *Analog Statistical Analysis* window.

The expression checking utility will automatically verify and assign the proper Data Type field to each expression. For any expression with an error, the Data Type field will be set to "ERROR". In this example, there should be no errors.

**Running the Statistical Analysis Simulation**

To run the statistical analysis,

> From the Analog Statistical Analysis window, choose *Simulation – Run*.

The simulation runs and the outputs for which *Autoplot* is set to *yes* appear in display windows.

When the simulation is finished and was completely successful, the ciw will produce the following information:

```
simulation completed successfully.
...
Monte Carlo Simulation completed successfully...
```

Although this example should run clean, additional information concerning simulation problems is useful to know. When any problems are encountered throughout the course of running a statistical analysis, both the ciw and simulator output log offer valuable information. The following three scenarios reflect the most common problems encountered:

1. For situations where a scalar expression could not be evaluated at a particular iteration, the following error information is produced in the ciw:
Problems encountered during simulation.
Use the Simulation->Output Log menu for more information.

Monte Carlo Simulation completed successfully...

When the spectre output log is reviewed, we would see an error similar to:

**** Run Status for Monte Carlo analysis ‘mc1’ ****
Monte Carlo iteration 1 failed.

The two most common reasons for this type of expression evaluation error are:

a. An analysis pertinent to the expression did not simulate. This is typically due to convergence problems.

b. The data created is not sufficient to satisfy the expression. For example, the phase at that iteration did not cross the proper threshold needed by the phaseMargin function.

Note: In this case, monte carlo scalar data was successfully produced. Any iterations with errors are assigned error flag values (-1.1111e36 or -2.2222e36).

2. For situations where a scalar expression had syntax errors, the following error information is produced in the ciw:

Problems encountered during simulation.
Use the Simulation->Output Log menu for more information.

Monte Carlo Simulation unsuccessful...

When the spectre output log is reviewed, we would see an error similar to:

Error found by spectre during Monte Carlo analysis ‘mc1’.
designParamVals: Error evaluating ocean expression ‘foo=getData("out")’.
Unsuccessfully evaluated export statements (based on return code).
Analysis ‘mc1’ terminated prematurely due to error.

In this case, no monte carlo scalar data was produced.

3. For situations where spectre could not run at all, the following error information is produced in the ciw:

Problems encountered during simulation.
Use the Simulation->Output Log menu for more information.
Monte Carlo Simulation unsuccessful...
See simulation output log for more information.

When the spectre output log is reviewed, we would see the error.

For example:
Error found by spectre during circuit read-in.
"input.scs" 9: Unable to open input file
'/example/monteCarlo/models/spectreLib.scs2'.
No such file or directory.
spectre terminated prematurely due to fatal error.
In this case, no monte carlo scalar data was produced.

Evaluating Statistical Analysis Results

These results might contain error flag values that can distort the statistical results (see problem case 1 in previous section). To check for possible error values, do the following:

1. Choose Results – Print – Iteration vs. Value.
2. In the Iteration Versus Value form, select the bandw_27 parameter, make sure that sorted is selected for the Output Format, and click Apply.
3. In the Results Display Window, check the values at the beginning and end of the list.
4. In the Iteration Verses Value form, select the ymax_27 parameter and click OK.
5. Repeat step 3 for the ymax_27 parameter values.

Analyzing Scalar Data

There are several ways to look at scalar data. As described in the previous section, you can simply list the data. You can also plot the data in the form of histograms and calculate yields for the data.

Printing Correlations Table

To determine which statistically swept parameters had the biggest impact on the output scalar measurements, perform the following:

1. Choose Results - Print - Correlation...
2. Set the *Suppress Printout for Correlations Less Than* field to 0.25 and click OK.

3. Observe the data printed to the *Results Display Window*.

From this data we can see that the `bandw_27` expression was mostly influenced by the `Cfb_27` parameter, and the `ymax_27` expression was mostly influenced by the `nnpbeta` parameter. We will use this knowledge further along in this example.

**Using Histograms**

To begin looking at the `bandw_27` and `ymax_27` data,

1. Choose *Results – Plot – Histogram*.

The Histogram form appears.

2. Add `bandw_27` and `ymax_27` to the *Plot* column.

3. Turn the *Density Estimator* button on.

4. Click *OK*. 
The histograms appear in the Waveform Window.

The \texttt{ymax\_27} distribution shows that the \texttt{ymax\_27} value is almost exactly the same for every iteration. That value is unaffected by the variations introduced into the circuit by the statistical analysis.

**Plotting Scatter Plots**

Scatter plots are very useful for verifying dependencies between different statistical data sets. Our findings in the Printing Correlation Tables section of this document will help guide this example.

1. Choose Results - Plot - Scatterplot...

2. Select the \texttt{bandw\_27} entry in the X-axis listbox, and the \texttt{Cfb\_27} entry in the Y-axis listbox. Click the Add button.
3. Select the `nptnbeta_27` entry in the Y-axis listbox. Click the Add button. The Scatter Plot form should look as follows:
4. Click the **OK** button. The following two scatterplots should be added to the Waveform Window:

![Scatterplots](image)

From these two plots we can see that the `bandw_27` expression is directly dependent on the `Cfb_27` parameter by observing a straight slanted line of points in the plot. Whereas the `npnbeta` parameter has little effect on `bandw_27` because there is little order in the plotted points.

**Analyzing Yields**

To analyze the yield for this circuit, you first need to define the specification limits for the scalar parameters. For this example, assume that you can tolerate only 1 sigma of variation.

1. Choose **Results – Specification Limits**.
2. Turn *Set By sigma* on.
3. Type 1 in the *Sigma* fields for each of the measurements.
4. Click *OK*.
5. From the *Analog Statistical Analysis* window, choose *Results – Yield – Simple*.
   The *Simple Yield* form appears.
6. Set the value of the *Suppress Printout for Yields Greater Than* field to 98 percent.
7. Click *OK*.
The Results Display Window appears.

The Results Display Window appears.

<table>
<thead>
<tr>
<th>param</th>
<th>yindv</th>
<th>mean</th>
<th>stddev</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandw_27</td>
<td>72.00%</td>
<td>9.5742e+03</td>
<td>1.2293e+02</td>
<td>100</td>
</tr>
<tr>
<td>ymax_27</td>
<td>93.00%</td>
<td>2.4971e+00</td>
<td>3.4504e-03</td>
<td>100</td>
</tr>
<tr>
<td>Rin_27</td>
<td>67.00%</td>
<td>4.9885e+03</td>
<td>1.5371e+02</td>
<td>100</td>
</tr>
<tr>
<td>Cfb_27</td>
<td>72.00%</td>
<td>1.0570e-09</td>
<td>6.3544e-10</td>
<td>100</td>
</tr>
<tr>
<td>PiRho_27</td>
<td>69.00%</td>
<td>2.4953e+03</td>
<td>3.2332e+02</td>
<td>100</td>
</tr>
<tr>
<td>PiRho_27</td>
<td>62.00%</td>
<td>1.9675e+02</td>
<td>5.4014e+01</td>
<td>100</td>
</tr>
<tr>
<td>rnbeta_27</td>
<td>83.00%</td>
<td>2.1930e+02</td>
<td>2.0489e+02</td>
<td>100</td>
</tr>
<tr>
<td>pmbeta_27</td>
<td>92.00%</td>
<td>3.9929e+02</td>
<td>6.1589e+02</td>
<td>100</td>
</tr>
<tr>
<td>rin2_27</td>
<td>66.00%</td>
<td>5.0198e+03</td>
<td>1.0003e+02</td>
<td>100</td>
</tr>
<tr>
<td>rout1_27</td>
<td>69.00%</td>
<td>9.9635e+02</td>
<td>2.7021e+01</td>
<td>100</td>
</tr>
<tr>
<td>rout2_27</td>
<td>67.00%</td>
<td>2.9971e+03</td>
<td>4.7946e+01</td>
<td>100</td>
</tr>
<tr>
<td>cin_27</td>
<td>68.00%</td>
<td>1.6999e-08</td>
<td>2.0960e-11</td>
<td>100</td>
</tr>
</tbody>
</table>

**total yield = 4.00% independent yield = 2.19% size = 100**

These results show that only 64 percent of the iterations produced results where both the bandw_27 and ymax_27 were within specification limits.

**Analyzing Waveform Data**

Two of the outputs in this example, dB20 and phase, are waveform data. Because this example was simulated with the Spectre simulator and because **Save Data Between Runs to Allow Family Plots** was turned on in the Analog Statistical Analysis window, you can use family-of-curves plots to examine the data.

1. From the Analog Statistical Analysis window, choose **Results – Plot– Curves.**
The Waveform Window appears with the statistical analysis results.
The preceding view is not very useful because the detail is too small to see, but if you turn off the phase subwindow and zoom in on the dB20 curves, you see a plot like this.

![Plot](image)

This plot shows that in the frequency range of about 400 K to 100 M, the dB20 value is affected by the statistical analysis variations introduced in the pnp and npn transistors. If this frequency range is critical, you might need to redesign your circuit so that the variation is smaller in this range.
Changing Waveform Expressions at Post-simulation Time

At post-simulation time, waveform expressions can be created, deleted or altered to produce new plots, so long as the pertinent psf data has been saved.

Note that it is not necessary to declare any waveform expressions in the Outputs Pane prior to running a statistical analysis simulation. However, doing so facilitates the waveform autoplot feature.

Remember, whenever attempting to plot a family of waveforms, make sure the `Save Data Between Runs to Allow Family Plots` boolean is on prior to running the statistical analysis. It is also necessary that all the needed circuit outputs are declared.

The previous plotting example involved zooming into the dB20 curves to see a particular region. Now let's change the dB20 expression to only include the region of concern by following these steps:

1. In the Outputs Pane window, select the dB20 entry.

2. Go to the editable expression string field and change the expression to:
   
   sample(dB20(VF("/OUT")) 100K 100M "log" 20)

   Click the `Change` button.

3. Invoke the `Results->Plot->Curves` capability.
Note that the dB20 plot now shows the desired region without having to manually zoom in:

### Changing Scalar Expressions at Post-Simulation Time

At post-simulation time, scalar expressions can be created, deleted or altered to produce new statistical data, so long as the pertinent psf data has been properly saved. It is required that the *Save Data Between Runs to Allow Family Plots* boolean was on at pre-simulation time. It is also necessary that all the needed circuit outputs were declared.

Prior to running a statistical analysis, so long as the *Save Data Between Runs to Allow Family Plots* boolean is on it is not necessary to declare any scalar expressions in the Outputs Pane. However, the performance of the *Analog Statistical Analysis* tool is at its optimum when all scalar expressions are declared and checked prior to simulation time.
In this example, we are going to add a new scalar to the statistical data by doing the following steps:

1. In the Outputs Pane window, select the clear button.
2. In the name field, type in gn_mrgn.
3. In the expression field, type in:
   "gainMargin(VF("/OUT"))"
4. Set the Data Type to blank (i.e. unknown).
5. Click the Add button. The Outputs pane of the UI should now look like the following:

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Expression/Signal</th>
<th>Data Type</th>
<th>Autoplot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>db20</td>
<td>sample(db20(VF(&quot;/OUT&quot;))) 100K...</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>phase</td>
<td>phase(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>bw</td>
<td>bandwidth(VF(&quot;/OUT&quot;) 3 &quot;low&quot;)</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>ymax</td>
<td>ymax(db20(VF(&quot;/OUT&quot;)))</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>gn_mrgn</td>
<td>gainMargin(VF(&quot;/OUT&quot;))</td>
<td>unknown</td>
<td>yes</td>
</tr>
</tbody>
</table>

6. Invoke the Simulation->Check_Expressions capability. The Data Type of this expression should then get automatically set to scalar. If not, then you will need to correct/change the expression and try again.

7. Invoke the Results->Evaluate_Expressions capability, and click OK on the Evaluate Expressions sub-form.

As a result of these actions, the new ph_marg statistical data has been created without the need to re-simulate. Lets investigate this new data:
a. Invoke the *Results->Print->Iteration Verses Value...* capability.

b. On the Iteration Verses Value form, note that the `gn_mrgn_27` entry now appears in the listbox. Select the `gn_mrgn_27` entry.
c. Set the Output Format to `sorted` and click `OK`. The following table will be printed to the Results Display Window:
### Results Display Window

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
<th>Column 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>-5.3152e+01</td>
<td>86</td>
<td>-4.7171e+01</td>
<td>80</td>
<td>-4.6882e+01</td>
<td>94</td>
</tr>
<tr>
<td>82</td>
<td>-4.3392e+01</td>
<td>44</td>
<td>-4.1817e+01</td>
<td>10</td>
<td>-4.1126e+01</td>
<td>12</td>
</tr>
<tr>
<td>57</td>
<td>-4.0773e+01</td>
<td>66</td>
<td>-4.0623e+01</td>
<td>85</td>
<td>-4.0535e+01</td>
<td>72</td>
</tr>
<tr>
<td>34</td>
<td>-4.0255e+01</td>
<td>20</td>
<td>-4.0061e+01</td>
<td>83</td>
<td>-3.9560e+01</td>
<td>23</td>
</tr>
<tr>
<td>96</td>
<td>-3.9555e+01</td>
<td>63</td>
<td>-3.9477e+01</td>
<td>97</td>
<td>-3.9373e+01</td>
<td>21</td>
</tr>
<tr>
<td>46</td>
<td>-3.9318e+01</td>
<td>49</td>
<td>-3.9236e+01</td>
<td>75</td>
<td>-3.9217e+01</td>
<td>75</td>
</tr>
<tr>
<td>81</td>
<td>-3.9079e+01</td>
<td>56</td>
<td>-3.8950e+01</td>
<td>92</td>
<td>-3.8296e+01</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>-3.8166e+01</td>
<td>70</td>
<td>-3.8010e+01</td>
<td>53</td>
<td>-3.7662e+01</td>
<td>11</td>
</tr>
<tr>
<td>67</td>
<td>-3.7723e+01</td>
<td>53</td>
<td>-3.7627e+01</td>
<td>14</td>
<td>-3.7427e+01</td>
<td>40</td>
</tr>
<tr>
<td>90</td>
<td>-3.7343e+01</td>
<td>75</td>
<td>-3.7231e+01</td>
<td>98</td>
<td>-3.7053e+01</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>-3.7042e+01</td>
<td>83</td>
<td>-3.7042e+01</td>
<td>52</td>
<td>-3.6960e+01</td>
<td>60</td>
</tr>
<tr>
<td>16</td>
<td>-3.6931e+01</td>
<td>99</td>
<td>-3.6895e+01</td>
<td>41</td>
<td>-3.6887e+01</td>
<td>87</td>
</tr>
<tr>
<td>47</td>
<td>-3.6631e+01</td>
<td>6</td>
<td>-3.6015e+01</td>
<td>5</td>
<td>-3.6773e+01</td>
<td>54</td>
</tr>
<tr>
<td>1</td>
<td>-3.6737e+01</td>
<td>15</td>
<td>-3.6706e+01</td>
<td>17</td>
<td>-3.6685e+01</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>-3.6669e+01</td>
<td>64</td>
<td>-3.6516e+01</td>
<td>95</td>
<td>-3.6565e+01</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>-3.6559e+01</td>
<td>100</td>
<td>-3.6478e+01</td>
<td>37</td>
<td>-3.6459e+01</td>
<td>94</td>
</tr>
<tr>
<td>71</td>
<td>-3.6428e+01</td>
<td>39</td>
<td>-3.6340e+01</td>
<td>22</td>
<td>-3.6337e+01</td>
<td>65</td>
</tr>
<tr>
<td>13</td>
<td>-3.6286e+01</td>
<td>74</td>
<td>-3.6235e+01</td>
<td>29</td>
<td>-3.6105e+01</td>
<td>43</td>
</tr>
<tr>
<td>33</td>
<td>-3.5884e+01</td>
<td>31</td>
<td>-3.5936e+01</td>
<td>63</td>
<td>-3.5823e+01</td>
<td>18</td>
</tr>
<tr>
<td>28</td>
<td>-3.5674e+01</td>
<td>25</td>
<td>-3.5578e+01</td>
<td>88</td>
<td>-3.5514e+01</td>
<td>45</td>
</tr>
<tr>
<td>93</td>
<td>-3.5276e+01</td>
<td>78</td>
<td>-3.5267e+01</td>
<td>51</td>
<td>-3.4883e+01</td>
<td>7</td>
</tr>
<tr>
<td>73</td>
<td>-3.4721e+01</td>
<td>9</td>
<td>-3.4614e+01</td>
<td>91</td>
<td>-3.4593e+01</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>-3.4456e+01</td>
<td>36</td>
<td>-3.4240e+01</td>
<td>7</td>
<td>-3.4228e+01</td>
<td>32</td>
</tr>
<tr>
<td>30</td>
<td>-3.3512e+01</td>
<td>25</td>
<td>-3.3297e+01</td>
<td>55</td>
<td>-3.2698e+01</td>
<td>27</td>
</tr>
<tr>
<td>79</td>
<td>-3.2600e+01</td>
<td>60</td>
<td>-3.2175e+01</td>
<td>61</td>
<td>-3.2096e+01</td>
<td>48</td>
</tr>
</tbody>
</table>
Note that iteration number 59 is an error flag value. If you look at the\textit{phase} plot created from a previous section of this example, you will notice that one of the plots does not dip below -180 degrees. If you put the mouse cursor on that single waveform, you can see the banner of the waveform tool indicating that the waveform is iteration number 59. For that waveform, the \texttt{gainMargin()} function cannot evaluate properly. The user would have to use the UI filter capabilities to remove this data point before analyzing this data.

\section*{Appending More Scalar Iterations to Existing Data}

In this example, we are going to append more statistical iterations onto the existing scalar data.

Note that this mode will erase the existing psf data for the first 100 iterations. It the user wishes to save this psf data for later use, then they should use the \texttt{Results->Save} capability prior to the next simulation. Although the Analog Statistical Analysis UI does not facilitate appending psf data for waveform plotting, the user can achieve this operation by using the data access capabilities outside the Analog Statistical Analysis UI. More on this subject later. For now, lets save the data:

\begin{itemize}
  \item[a.] Invoke the \texttt{Results->Save} capability.
  \item[b.] Set the \textit{Save As} field to \texttt{first\_100\_iterations}.
  \item[c.] Click the \textit{OK} button.
\end{itemize}

Now lets proceed with appending an additional 100 runs to the existing scalar data. Set up the UI as follows:

\begin{itemize}
  \item[a.] Set the \textit{Starting Run #} to 101.
  \item[b.] Turn on the \textit{Append to Previous Scalar Data} boolean.
The Analog Statistical Analysis form should appear as follows:

![Analog Statistical Analysis form](image)

c. Invoke Simulation->Run.

When the simulation completes, you will notice that the autopotted histograms indicate a sample set of 200 ("N = 200"). You will also notice that the autopotted waveforms are
for iterations 101 to 200 when you drag the cursor across the waveforms and look at the banner info.

If you use the Results->Iteration_Verses_Value capability, you will notice that the iterations are from 1 to 200 (use the unsorted mode).

Note, at this point, attempting to use the Results->Evaluate_Expressions capability will cause the first 100 iterations to be purged from the scalar data. This is because the psf data does not contain the first 100 iterations. Whenever scalar iterations data will be purged, the UI will inform the user prior to purging the data. The user will be able to abort creating the new scalar data.

Appending Waveforms From Different Statistical Analysis Runs.

The steps in this section outline how the user can overlay waveform plots from two different analog statistical analysis psf data sets. We are assuming we are continuing from the previous section. Perform the following steps:

1. Close all open Waveform Windows.
2. On the Analog Statistical Analysis UI Outputs Pane, select the desired expression row.
3. Go to the editable expression field and highlight the entire expression.
4. Bring up the Calculator and paste the expression into the buffer. Push the erplot button.
5. On the Analog Statistical Analysis UI, invoke the Results->Select capability. Set the result name to first_100_iterations and click OK. Click OK on the Initialize Statistical Analysis Data form.
6. On the calculator, push the plot button.

Now, all 200 waveforms should be shown in a single plot.
Performing a Swept Parameter Statistical Analysis.

An example UI setup for performing a Swept Parameter statistical analysis is shown in the following picture.
### Analog Statistical Analysis

**Status:** Ready  
**Simulator:** spectre  
**Number of Runs:** 100  
**Starting Run #:** 1  
**Analysis Variation:** Process & Mismatch  
**Swept Parameter:** Temperature  

- **Append to Previous Scalar Data:** No  
- **Save Data Between Runs to Allow Family Plots:** Yes

### Outputs

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Expression/Signal</th>
<th>Data Type</th>
<th>Autoplot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dB20</td>
<td>dB20(VF(&quot;/OUT&quot;))</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>phase</td>
<td>phase(VF(&quot;/OUT&quot;))</td>
<td>wave</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>bandw</td>
<td>bandwidth(VF(&quot;/OUT&quot;) 3 &quot;low&quot;)</td>
<td>scalar</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>ymax</td>
<td>ymax(dB20(VF(&quot;/OUT&quot;)))</td>
<td>scalar</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Add**  
- **Delete**  
- **Change**  
- **Clear**  
- **Calculator...**  
- **Get Expression**
This section will not go into showing all the previously demonstrated UI features. Using the parametric data sets created by a run with the above UI configuration, please return to the section titled *Evaluating Statistical Analysis Results* and repeat the same example through to the end of this document.
Optimization

Optimization is the process of automatically modifying design variables so that specifications are achieved. The tool that performs optimization is called the optimizer. Often the optimizer can take a design that is close to meeting performance specifications and generate new component values that bring the design into the acceptable performance range.

You can apply optimization profitably in a wide range of activities.

- If you use a top-down design approach, you can optimize a circuit block to match the performance characteristics of an analog HDL module.
- Using the opposite approach, you can optimize a macro or behavioral module to describe the behavior of a circuit block. Then, instead of simulating with the circuit block, you can simulate with the macro or behavioral model, which usually runs much faster.
- You can optimize model parameters to match measured device data under various conditions.
- You can use optimization to address radio frequency (RF) design problems, such as impedance matching.
- To increase circuit yield, you can use optimization to achieve better design center values.
- You can use optimization to match the frequency response of a filter to the specifications for the filter.
- You can use optimization to balance design tradeoffs.

The sections in this chapter explain how you can use the optimizer to help achieve your design goals.

- “Getting Started with Optimization” on page 162
- “Getting to Know the Virtuoso® Analog Circuit Optimization Option Window” on page 165
- “Running an Optimization” on page 170
Getting Started with Optimization

This section briefly explains the theory behind optimization, tells you how to get help, and describes how to open the Virtuoso®™ Analog Optimization Analysis window.

How Optimization Works

A circuit, as originally designed, often fails to meet its specifications. For example, the bandwidth might be too small or the design might be off center so that the yield is low. You might be able to improve a marginal design by using components with different values, but determining the best values to use is often difficult. The optimizer can provide you with information that can help you choose values that meet your design goals.

To use the optimizer, you specify initial values for a set of design variables. You also specify the goals you want the circuit to meet. The optimizer first determines how the values of the goal expressions vary as a function of changes to the design variables. Then the optimizer changes the design variables in a manner expected to move the values of the expressions in the direction of the goals. After the change, the optimizer simulates the circuit to check the outcome. If stopping criteria are not met, the optimizer iterates through the optimization process.

The following steps describe, in greater detail, the process that the optimizer follows during an analysis.

1. The optimizer runs a simulation using the initial values you specify for the design variables.

   This step determines the types and initial values of the goal expressions.

2. The optimizer determines which optimization algorithm to use (unless you specify which to use).

   - The LSQ (least square) algorithm is best suited for optimizing measured, noisy, unconstrained data. For example, this algorithm is appropriate for designing a filter with an output waveform that matches measured frequency response data.

   - The CFSQP (C version Feasible Sequential Quadratic Programming) algorithm is suited for a wide variety of optimization problems, including constrained and unconstrained, minimizing and maximizing, and sequentially related goals. For example, this algorithm is appropriate for a low noise amplifier design that has many
goals such as maximizing the gain, minimizing the noise, and maintaining a phase margin greater than 45 degrees.

3. If the CFSQP algorithm is used, the optimizer runs a simulation to determine whether the initial values are feasible for the given goals. If the initial values are not feasible, the optimizer computes new values that are feasible.

4. The optimizer determines how sensitive the goal expressions are to each design variable.

To determine these sensitivities, the optimizer changes each design variable slightly and then simulates the design again. This technique is called *Finite Difference Perturbation*. In this technique, users generally do not need to select which algorithm to use for their problem. The optimizer decides which algorithm to use based on the type of optimization to be done. However, this option provides the ability to force the optimizer to use a particular algorithm.

5. Using the information on sensitivities, the optimizer calculates a new set of values for the design variables.

6. The optimizer sets the design variables to the new values and simulates the circuit.

   If the values of the goal expressions are not better than they were with the previous design variable values, the optimizer repeats Step 5.

   If the values of the goal expressions are better than they were with the previous design variable values, the new values become the initial values for the next iteration.

7. If the stopping criteria are not yet met, the optimizer begins the next iteration with Step 4.

   Optimization stops when either or both of the following stopping criteria are met:
   
   - The values of the design variables change very little or not at all
   - Further changes to the design variables result in no progress toward the goals

**Getting Help**

For the most extensive information about using the optimizer, continue reading this document. To open this document online, choose *Help – Contents* in the Virtuoso® Analog Circuit Optimization Option window menu.

For information about the precise product name and version for this tool, choose *Help – About Analog Circuit Optimization*. 
Opening and Closing the Virtuoso® Analog Circuit Optimization Option Window

When your circuit is ready to optimize,

1. Set up a simulation for it in the usual way.

2. In the Virtuoso® Analog Design Environment window, choose *Tools* – *Optimization*.

To close the Virtuoso® Analog Circuit Optimization Option window,

➤  Choose *Session* – *Quit*. 
Getting to Know the Virtuoso® Analog Circuit Optimization Option Window

The *Virtuoso® Analog Circuit Optimization Option* window contains the primary controls and tables you need for an optimization session.

**Status Display**

The status display shows the current state of the optimizer. For example, the status display indicates whether the optimizer is simulating, optimizing, or in some other phase of the analysis.
Menu

The menu contains the commands needed to prepare for, run, and plot the results of an optimization.

<table>
<thead>
<tr>
<th>Session</th>
<th>Goals</th>
<th>Variables</th>
<th>Optimizer</th>
<th>Results</th>
<th>Help</th>
</tr>
</thead>
</table>

For guidance on using the menu choices, see the cross-references in the following table.

**Optimization Menu Selections**

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Session</strong></td>
<td></td>
</tr>
<tr>
<td>Save State</td>
<td>“Saving the Session State” on page 195</td>
</tr>
<tr>
<td>Load State</td>
<td>“Loading a Saved Session State” on page 196</td>
</tr>
<tr>
<td>Save Script</td>
<td>“Saving a Script” on page 197</td>
</tr>
<tr>
<td>Options</td>
<td>“Changing Optimization Options” on page 197</td>
</tr>
<tr>
<td>Reset</td>
<td>“Deleting All Setup Information” on page 199</td>
</tr>
<tr>
<td>Quit</td>
<td>“Opening and Closing the Virtuoso® Analog Circuit Optimization Option Window” on page 164</td>
</tr>
<tr>
<td><strong>Goals</strong></td>
<td></td>
</tr>
<tr>
<td>Retrieve Outputs</td>
<td>“Using Simulation Outputs as Goals” on page 171</td>
</tr>
<tr>
<td>Add</td>
<td>“Creating a New Goal by Entering It Directly” on page 172 and “Creating a New Goal by Using the Waveform Calculator” on page 174</td>
</tr>
<tr>
<td>Edit</td>
<td>“Editing a Goal” on page 177</td>
</tr>
<tr>
<td>Delete</td>
<td>“Deleting a Goal” on page 178</td>
</tr>
<tr>
<td>Enable</td>
<td>“Enabling or Disabling a Goal” on page 178</td>
</tr>
<tr>
<td>Disable</td>
<td>“Enabling or Disabling a Goal” on page 178</td>
</tr>
</tbody>
</table>
### Optimization Menu Selections, continued

<table>
<thead>
<tr>
<th>Menu Item</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables</strong></td>
<td></td>
</tr>
<tr>
<td>Add/Edit</td>
<td>“Adding a Design Variable” on page 186 or “Editing a Design Variable” on page 187</td>
</tr>
<tr>
<td>Delete</td>
<td>“Deleting a Design Variable” on page 188</td>
</tr>
<tr>
<td>Enable</td>
<td>“Enabling or Disabling a Design Variable” on page 189</td>
</tr>
<tr>
<td>Disable</td>
<td>“Enabling or Disabling a Design Variable” on page 189</td>
</tr>
<tr>
<td><strong>Optimizer</strong></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>“Running the Optimizer” on page 189</td>
</tr>
<tr>
<td>Step</td>
<td>“Running the Optimizer” on page 189</td>
</tr>
<tr>
<td>Run n</td>
<td>“Running the Optimizer” on page 189</td>
</tr>
<tr>
<td>Stop</td>
<td>“Stopping the Optimizer” on page 190</td>
</tr>
<tr>
<td>Stop Now</td>
<td>“Stopping the Optimizer” on page 190</td>
</tr>
<tr>
<td>Reset</td>
<td>“Deleting Simulation Results” on page 191</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td></td>
</tr>
<tr>
<td>Plot History</td>
<td>“Plotting Output Data” on page 194</td>
</tr>
<tr>
<td>Set Plot Options</td>
<td>“Setting the Plotting Options” on page 191</td>
</tr>
<tr>
<td>Update Design</td>
<td>“Updating Your Design” on page 194</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td></td>
</tr>
<tr>
<td>Contents</td>
<td>“Getting Help” on page 163</td>
</tr>
<tr>
<td>About Analog Circuit Optimization</td>
<td>“Getting Help” on page 163</td>
</tr>
</tbody>
</table>
Goals Pane

The Goals pane displays information about the currently defined goals.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Direction</th>
<th>Target</th>
<th>Initial</th>
<th>Prev</th>
<th>Current</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OutGoal</td>
<td>match</td>
<td>OutTa..</td>
<td></td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

To define or revise the goals, you use the menu choices and buttons in the Virtuoso® Analog Circuit Optimization Option window. For a description of the items in the Goals pane, see the following table.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name associated with the goal.</td>
</tr>
<tr>
<td>Direction</td>
<td>One of the following: maximize, minimize, match, &gt;=, or &lt;=.</td>
</tr>
<tr>
<td>Target</td>
<td>If Direction is match, a value or waveform that the optimizer attempts to match.</td>
</tr>
<tr>
<td></td>
<td>If Direction is maximize or minimize, a value or waveform used to determine how important the goal is.</td>
</tr>
<tr>
<td></td>
<td>If Direction is &gt;=, a value or waveform that is the lower bound.</td>
</tr>
<tr>
<td></td>
<td>If Direction is &lt;=, a value or waveform that is the upper bound.</td>
</tr>
<tr>
<td>Initial</td>
<td>The value of the goal expression as calculated from the initial values of the design variables.</td>
</tr>
<tr>
<td>Prev</td>
<td>The value of the goal expression as calculated from the values of the design variables used in the previous iteration.</td>
</tr>
<tr>
<td>Current</td>
<td>The value of the goal expression as calculated from the current values of the design variables.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Either yes or no. yes indicates that the goal is included in the current optimization; no, that it is not.</td>
</tr>
</tbody>
</table>
Variables Pane

The *Variables* pane displays information about the current design variables.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Min</th>
<th>Max</th>
<th>Initial</th>
<th>Prev</th>
<th>Current</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>res2</td>
<td>500</td>
<td>50K</td>
<td>10K</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>res</td>
<td>500</td>
<td>50K</td>
<td>1K</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

To define, revise, or enable the variables, you use the menu choices and buttons in the *Virtuoso® Analog Circuit Optimization Option* window. For a description of the items in the *Variables* pane, see the following table.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the design variable.</td>
</tr>
<tr>
<td>Min</td>
<td>The minimum value allowed for the variable.</td>
</tr>
<tr>
<td>Max</td>
<td>The maximum value allowed for the variable.</td>
</tr>
<tr>
<td>Initial</td>
<td>The initial value of the design variable.</td>
</tr>
<tr>
<td>Prev</td>
<td>The value of the design variable used in the previous iteration.</td>
</tr>
<tr>
<td>Current</td>
<td>The value of the design variable used in the current iteration.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Either <em>yes</em> or <em>no</em>. <em>Yes</em> indicates that the value of the design variable can be changed in the current optimization; <em>no</em>, that it cannot be.</td>
</tr>
</tbody>
</table>
Tool Bar

The tool bar contains buttons that perform the most important optimization tasks. The buttons are arranged from top to bottom in the order they are typically used.

Add/Edit Goals
Add/Edit Variables
Delete
Run Optimizer
Stop Optimizer
Plot History
Update Design

Running an Optimization

The following sections describe the major steps involved in setting up and running an optimization.

- “Defining Goals” on page 171
- “Preparing Design Variables” on page 186
- “Controlling the Optimizer” on page 189
- “Plotting Results” on page 191

You might be able to skip the first two steps by loading a state you saved in an earlier session. For more information, see “Loading a Saved Session State” on page 196.
Defining Goals

Before you can run the optimizer on a circuit, you must specify the goals for the analysis. A goal consists of

- An expression whose value can be determined by simulation
- A direction specifying how the value of the expression is to change during optimization

For example, you might define a goal called Bandwidth(3dB) with the expression

`bandwidth(VF("/out") 3 "low")`

and specify that the value of the expression is to be maximized during optimization.

The following sections describe how to create, edit, delete, enable, and disable goals.

- “Using Simulation Outputs as Goals” on page 171
- “Creating a New Goal by Entering It Directly” on page 172
- “Creating a New Goal by Using the Waveform Calculator” on page 174
- “Editing a Goal” on page 177
- “Deleting a Goal” on page 178
- “Enabling or Disabling a Goal” on page 178

For information about how the optimizer assigns weights to goals or about creating waveform objects, look also at

- “Creating Waveform Objects from a List of Values” on page 178
- “How the Optimizer Uses Target and Acceptable Values” on page 181

Using Simulation Outputs as Goals

The optimizer allows you to use the simulation outputs defined in the Virtuoso® Analog Design Environment window as goals. This approach is particularly useful for RF analyses where expressions are often developed for Direct Plot. You can easily add these Direct Plot expressions to the Virtuoso® Analog Design Environment window Outputs pane, and from there you can use the following steps to retrieve the expressions for use as optimizer goals.

1. From the Virtuoso® Analog Circuit Optimization Option window, choose Goals – Retrieve Outputs.
The expressions (but not any signals) defined in the Outputs pane of the Design Environment window appear in the Goals pane of the Optimization Option window. Initially, the new goals are not enabled.

Unnamed expressions are given names like G1, G2, G3, and so on.

2. Highlight a goal and choose Goals – Edit.

The Editing Goals window appears.

3. Finish defining the goal by following the instructions in “Editing a Goal” on page 177.

Until you edit a retrieved goal in the Editing Goals window, choosing Goals – Retrieve Outputs updates the goal to match the existing expression in the Outputs pane of the Design Environment window. After you edit a retrieved goal, choosing Goals – Retrieve Outputs has no effect on the goal.

Creating a New Goal by Entering It Directly

To create a new goal by entering it directly,

1. Choose Goals – Add or click Add/Edit Goals.

The Adding Goals form appears.

2. Type a name for the goal.
3. Type a Cadence® SKILL language expression describing the goal.
   The expression can be either a scalar expression or a waveform expression.

4. In the Direction cyclic field, indicate how the value of the expression is to change during optimization.

5. Type an expression in the Target field.
   If the expression you enter in Step 3 is a scalar expression, Target must also be a scalar expression. If the expression you enter in Step 3 is a waveform, Target can be either a scalar expression or a waveform expression.

6. Specify a value in the Acceptable field.
   As described in “How the Optimizer Uses Target and Acceptable Values” on page 181, the optimizer uses the Acceptable value to determine how important the goal is.
   There are two ways to specify the Acceptable value.
   - You can type an expression in the Acceptable field.
     If the expression you enter in Step 3 is a scalar expression, Acceptable must also be a scalar expression. If the expression you enter in Step 3 is a waveform, Acceptable can be either a scalar expression or a waveform expression.
     A scalar Acceptable expression must meet the following requirements. A waveform Acceptable expression must meet the following requirements at every point along the curve.

<table>
<thead>
<tr>
<th>If you specify Direction as</th>
<th>Then the value of the Acceptable expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimize</td>
<td>Must be greater than the value of the Target expression</td>
</tr>
<tr>
<td>maximize</td>
<td>Must be less than the value of the Target expression</td>
</tr>
<tr>
<td>match</td>
<td>Can be any value except the Target expression</td>
</tr>
<tr>
<td></td>
<td>In addition, a waveform Acceptable expression must be everywhere greater than or everywhere less than the Target expression.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Must be greater than the value of the Target expression</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Must be less than the value of the Target expression</td>
</tr>
</tbody>
</table>
If % within Target is turned on, you can specify a scalar or waveform percentage in the Acceptable field. A small percentage indicates that the goal is to be heavily weighted.

If the expression you enter in Step 3 is a scalar expression, the percentage in the Acceptable field must also be a scalar expression. If the expression you enter in Step 3 is a waveform, the percentage in the Acceptable field value can be either a scalar expression or a waveform expression.

By specifying a scalar percentage, you can ensure that the optimization results are consistent at both very small and very large values of the Target expression. By specifying a waveform percentage, you can explicitly specify the importance of each segment of a waveform goal.

How the optimizer uses the Target and Acceptable expressions depends on the direction you choose. For details, see “How the Optimizer Uses Target and Acceptable Values” on page 181.

7. If you want to include the goal in the current optimization, be sure Enabled is on.

8. Click OK.

The new goal is added to the Virtuoso® Analog Circuit Optimization Option window.

Creating a New Goal by Using the Waveform Calculator

To create a new goal by using the Waveform Calculator,

1. Choose Goals – Add or click Add/Edit Goals.
The Adding Goals form appears.

2. Type a name for the goal.

3. Click *Open*.

   The calculator window appears.

4. Build the goal expression in the Waveform Calculator. For information on using the Waveform Calculator, see the *Waveform Calculator User Guide*.

5. In the Virtuoso® Analog Circuit Optimization Option window, click in the *Expression* field.

6. Click *Get Expression*, which retrieves the expression from the Waveform Calculator and places it in the *Expression* field.

7. In the *Direction* cyclic field, indicate how the value of the expression is to change during optimization.

8. Type an expression in the *Target* field (or click in the *Target* field, then click *Get Expression* to retrieve an expression from the Waveform Calculator).

   If the expression you retrieve in Step 5 is a scalar value, *Target* must also be a scalar expression. If the expression you retrieve in Step 5 is a waveform, *Target* can be either a scalar expression or a waveform expression.
9. Specify a value for *Acceptable*. As described in “How the Optimizer Uses Target and Acceptable Values” on page 181, the optimizer uses the *Acceptable* value to determine how important the goal is.

There are two ways to define this value.

- First, you can type an expression in the *Acceptable* field (or click in the *Acceptable* field, then click *Get Expression* to retrieve an expression from the Waveform Calculator).

  If the expression you retrieve in Step 6 is a scalar expression, *Acceptable* must also be a scalar expression. If the expression you retrieve is a waveform, *Acceptable* can be either a scalar expression or a waveform expression.

  A scalar *Acceptable* expression must meet the following requirements. A waveform *Acceptable* expression must meet the following requirements at every point along the curve.

<table>
<thead>
<tr>
<th>If you specify Direction as</th>
<th>Then the value of the Acceptable expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>minimize</em></td>
<td>Must be greater than the value of the <em>Target</em> expression</td>
</tr>
<tr>
<td><em>maximize</em></td>
<td>Must be less than the value of the <em>Target</em> expression</td>
</tr>
<tr>
<td><em>match</em></td>
<td>Can be any value except the <em>Target</em> expression</td>
</tr>
<tr>
<td></td>
<td>In addition, a waveform <em>Acceptable</em> expression must be everywhere greater than or everywhere less than the <em>Target</em> expression.</td>
</tr>
<tr>
<td><em>&lt;=</em></td>
<td>Must be greater than the value of the <em>Target</em> expression</td>
</tr>
<tr>
<td><em>&gt;=</em></td>
<td>Must be less than the value of the <em>Target</em> expression</td>
</tr>
</tbody>
</table>

- Second, if % within *Target* is turned on, you can specify a scalar or waveform percentage in the *Acceptable* field. A small percentage indicates that the goal is to be heavily weighted.

  If the expression you retrieve in Step 6 is a scalar expression, the percentage you enter in the *Acceptable* Field must also be a scalar expression. If the expression you retrieve is a waveform, the percentage value can be either a scalar expression or a waveform expression.

  By specifying a scalar percentage, you can ensure that the optimization results are consistent at both very small and very large values of the *Target* expression. By
specifying a waveform percentage, you can explicitly specify the importance of each
segment of a waveform goal.

How the optimizer uses the Target and Acceptable values depends on the direction you
choose. For details, see “How the Optimizer Uses Target and Acceptable Values” on
page 181.

10. If you want to include the goal in the current analysis, be sure Enabled is on.

11. Click OK to add the new goal to the Virtuoso® Analog Circuit Optimization Option
window.

Editing a Goal

To edit an existing goal,

1. In the Virtuoso® Analog Circuit Optimization Option window, highlight the goal you want
to edit.

2. Choose Goals – Edit or click Add/Edit Goals.

The Editing Goals form appears.

Except for the title, this form is identical to the Adding Goals form and you use the two
forms in the same way.
3. Make the changes you want to make.

   For details, see “Creating a New Goal by Entering It Directly” on page 172 or “Creating a New Goal by Using the Waveform Calculator” on page 174.

4. Click OK.

   The changes are applied to the highlighted goal.

Deleting a Goal

To delete an existing goal,

1. In the Virtuoso® Analog Circuit Optimization Option window, highlight the goal you want to delete.

2. Choose Goals – Delete or click Delete.

   The highlighted goal disappears from the Virtuoso® Analog Circuit Optimization Option window.

Enabling or Disabling a Goal

For a quick way to enable or disable a goal,

1. Highlight the goal in the Virtuoso® Analog Circuit Optimization Option window.

2. Choose Goals – Enable or Goals – Disable.

   The optimization does not include disabled goals.

Creating Waveform Objects from a List of Values

When the expression that defines a goal is a waveform, the Target and Acceptable values you define can also be waveforms. This section describes how you can create a waveform object from a list of values stored in a file. For information on some of the other ways you can create a waveform, see the Waveform Calculator User Guide.

This scenario assumes you have a list of X and Y values stored in a file. The X data values must be monotonically increasing. For example, you might have a file called mydata containing the following information:

    ; The information in column 1 is for the X axis.
    ; The information in column 2 is for the Y axis.
    0.5 5
    0.6 5.1
To convert this data into a waveform object,

1. Open the Waveform Calculator.

2. Click *Special Functions*.

3. Choose *table* from the list of functions.

   The Table form appears.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Function Name</th>
<th>X Column Number</th>
<th>Y Column Number</th>
<th>X Skip Lines</th>
<th>Y Skip Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>~/mydata</td>
<td>myWaveObject</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4. In the *File Name* field, type the filename of the file that contains your data.

5. In the *Function Name* field, type the name you want the waveform object to have.

6. (Optional) Specify which columns contain the X and Y data. This step is not required if the X data is in column 1 and the Y data is in column 2.

7. (Optional) Type the number of lines to skip in each column before reading the data. Do not count comment lines, which begin with a semicolon, and blank lines in the number of lines to skip.

8. Click *OK*.

   The Table form closes and the wave object appears in the calculator buffer. For example, filling in the Table form as shown above and clicking *OK* causes

   ```
   myWaveObject()
   ```
to appear in the calculator buffer.

9. (Optional) In the Waveform Calculator, click *plot* to plot the waveform.

The Waveform Window opens with a plot of the waveform. The `myWaveObject()` waveform, for example, looks like this.
You can use the new waveform object in the optimizer wherever waveforms are valid. For example, to use `myWaveObject()` as a target, type the name in the Target field.

![Optimizer dialog box](image)

**How the Optimizer Uses Target and Acceptable Values**

The optimizer uses a Target value in two ways: as a goal and as an indication of the weight (importance) of the goal. The following sections describe each use in more detail.

**Target Values Used as Goals**

The first use, as a goal, is most obvious when you specify a Direction of `match`, `>=`, or `<=`. In these cases, the target is the value the optimizer attempts to reach.

<table>
<thead>
<tr>
<th>When you specify a Direction of</th>
<th>The optimizer attempts to make the value of the goal expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>match</code></td>
<td>Match the Target value exactly</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than the Target value</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than the Target value</td>
</tr>
</tbody>
</table>

However, when you specify a Direction of `minimize`, the optimizer does not stop minimizing the goal expression when the Target value is reached. In fact, the optimizer makes the goal
expression as small as the optimization stopping criteria allow, even if that means the final value is much less than the Target. Similarly, when you specify a Direction of maximize, the optimizer makes the goal expression as large as possible, even if that means the value is much greater than the Target value.

<table>
<thead>
<tr>
<th>When you specify a Direction of</th>
<th>The optimizer attempts to make the value of the goal expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimize</td>
<td>As small as possible, regardless of the Target value</td>
</tr>
<tr>
<td>maximize</td>
<td>As large as possible, regardless of the Target value</td>
</tr>
</tbody>
</table>

**Note:** The LSQ algorithm makes no distinction among the directions of match, minimize, and maximize. In each of these cases, the LSQ algorithm works to match the Target value. For more information, see the information about the LSQ algorithm in “Changing Optimization Options” on page 197.

**Target Values Used to Assign Weights**

When you set a Direction of minimize or maximize, the second use of the Target value is most obvious. In these two cases, the Target value, together with the Acceptable value, is used only to assign a weight to the goal. By contrast, in the match, >=, and <= cases, the Target value is used both to set a goal and assign a weight.

The optimizer assigns greater weight to a goal that is defined with an Acceptable value set very close to the Target value. Similarly, if the Target and Acceptable values are waveforms, the optimizer assigns more weight to points where the Target and Acceptable values are close together, and less weight to points where the Target and Acceptable values are farther apart.

In formal terms, the weight assigned to a goal depends on the current value of the goal expression ($f$), the Target value ($T$), and the Acceptable value ($A$).

$$\text{weight} = \frac{f - T}{A - T}$$

For example, assume you have two goals defined as follows.

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Target Value</th>
<th>Acceptable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>&lt;=</td>
<td>50 mW</td>
<td>80 mW</td>
</tr>
<tr>
<td>delay</td>
<td>&lt;=</td>
<td>50 ns</td>
<td>60 ns</td>
</tr>
</tbody>
</table>
If the current value of the power expression is 90 mW, its weight is

\[
\frac{90mW - 50mW}{80mW - 50mW} = \frac{40mW}{30mW} = 1.333
\]

If the current value of the delay expression is 90 ns, its weight is

\[
\frac{90ns - 50ns}{60ns - 50ns} = \frac{40ns}{10ns} = 4
\]

Because the delay weight is greater than the power weight, the optimizer assigns greater importance to reducing delay than it does to reducing power. In a tradeoff between the two, reducing delay comes out ahead.

The next example illustrates how the optimizer determines weights from waveform Target and Acceptable values. In this example, you match the output of a filter to a particular waveform. Parts of the range are critical, so you assign a heavy weight to those sections by defining very similar Target and Acceptable waveforms. The middle section is not critical,
so you assign less weight to that area by defining Target and Acceptable waveforms that are farther apart.

The final example illustrates how the optimizer determines an Acceptable waveform from a waveform Target and a waveform % within Target value. In this example, Direction is
specified as >= and the calculated *Acceptable* waveform is everywhere less than the *Target* waveform.

At the left side of the plot, the % *within target* waveform has the value 1, so the calculated value of the *Acceptable* waveform in that region is

$$100 - (0.01 \times 100) = 99$$

At the right side of the plot, the % *within target* waveform has the value 10, so the calculated value of the *Acceptable* waveform in that region is

$$200 - (0.10 \times 200) = 180$$
Preparing Design Variables

Before you can run the optimizer on a circuit, you must specify which design variables the optimizer is allowed to change. The design variables you specify must be simulation environment variables such as component parameters and device model parameters. Typical examples include variables for resistor, capacitor, and inductor values or for device widths, lengths, and areas.

The following sections describe how to add, edit, delete, enable, and disable design variables.

- “Adding a Design Variable” on page 186
- “Editing a Design Variable” on page 187
- “Deleting a Design Variable” on page 188
- “Enabling or Disabling a Design Variable” on page 189

Adding a Design Variable

To add a design variable to the list of variables in the Virtuoso® Analog Circuit Optimization Option window,

1. Choose Variables – Add/Edit or click Add/Edit Variables.
The Editing Variables form opens.

2. Highlight the variable you want to add.

3. In the Initial Value field, type a value to be used as the starting point for optimization.

4. In the Minimum Value field, type a minimum value. The optimizer never sets the variable to a value lower than this.

5. In the Maximum Value field, type a maximum value. The optimizer never sets the variable to a value greater than this.

6. If you want to include the variable in the current analysis, be sure Enabled is on.

7. Click OK.

The variable is added to the Virtuoso® Analog Circuit Optimization Option window.

Editing a Design Variable

To edit one of the design variables listed in the Virtuoso® Analog Circuit Optimization Option window,
1. Highlight the variable you want to edit.

2. Choose Variables – Add/Edit or click Add/Edit Variables.
   The Editing Variables form opens.

3. In the Initial Value field, type a value to be used as the starting point for optimization.

4. In the Minimum Value field, type a minimum value. The optimizer never sets the variable to a value lower than this.

5. In the Maximum Value field, type a maximum value. The optimizer never sets the variable to a value greater than this.

6. If you want to include the variable in the current analysis, be sure Enabled is on.

7. Click OK.
   The changes are applied.

**Deleting a Design Variable**

To delete one of the design variables listed in the Virtuoso® Analog Circuit Optimization Option window,
1. Highlight the variable you want to delete.

2. Choose Variables – Delete or click Delete.

Enabling or Disabling a Design Variable

For a quick way to enable or disable a design variable,

1. Highlight the design variable in the Virtuoso® Analog Circuit Optimization Option window.

2. Choose Variables – Enable or Variables – Disable.

Controlling the Optimizer

After you define the goals and specify the design variables to use, you are ready to use the optimizer. The following sections describe how to run and stop the optimizer and how to delete simulation results you do not want to keep.

Running the Optimizer

To run the optimizer,

1. If the previous run of the optimizer stopped because an error occurred, click Stop Optimizer to clear the existing state.

2. Choose one of the run commands from the Virtuoso® Analog Circuit Optimization Option window.
   - Choose Optimizer – Run or click Run Optimizer to start the optimization from the beginning and run it until the stopping criteria are met.
   - Choose Optimizer – Step to start the optimization from the most recent stopping point, iterate once, and then stop.
     If you want to start from the beginning of the optimization, choose Optimizer – Reset before choosing Optimizer – Step.
Choose *Optimizer – Run n* to open the Run for Fixed Number of Iterations form.

In the form, set the value of \( n \) by moving the slider.

Click *OK* to start the optimization from the most recent stopping point, using the most recent variable values, and run at most \( n \) iterations before pausing or stopping. If you want to start from the beginning of the optimization, choose *Optimizer – Reset* before choosing *Optimizer – Run n*.

As each iteration finishes, the optimizer updates the *Prev* and *Current* values displayed in the Virtuoso® Analog Circuit Optimization Option window.

Be aware that if you load or reload a state, the next optimizer run starts at the beginning of the optimization. Be aware also that if you change the characteristics of a goal or the number of enabled goals and then resume the optimization, the optimizer performs another initial simulation.

**Stopping the Optimizer**

To stop the optimizer,

- Choose one of the stop commands from the Virtuoso® Analog Circuit Optimization Option window.

  - Choose *Optimizer – Stop* or click *Stop Optimizer* to stop the optimizer after the current iteration.
  - Choose *Optimizer – Stop Now* to stop the optimizer immediately without necessarily completing the current iteration.

You can also click *Stop Optimizer* to clear the state if an error stops the optimizer before the normal end of the run.
Deleting Simulation Results

To delete all simulation results,

➤ Choose Optimizer – Reset.

Any simulation results that exist are deleted. Goals, design variables, and plotting options remain unchanged.

Plotting Results

The easiest way to track the progress that the optimizer makes toward the goal is to plot the data as it becomes available at each iteration. When the optimizer achieves acceptable results, you can update the design to incorporate the optimized variable values.

The following sections explain how to set the plotting options, how to plot the output data, and how to update your design with the calculated optimal values.

Setting the Plotting Options

To set the plotting options,

1. In the Virtuoso® Analog Circuit Optimization Option window, choose Results – Set Plot Options.
The Setting Plotting Options form appears.

2. If you want to be able to follow the progress of the optimization while the optimizer is running, turn on **Auto Plot After Each Iteration**.

   If you do not want to follow the progress during the run, you can turn off this button and plot the results when the run has finished. For more information, see “Plotting Output Data” on page 194.

3. Select at least one kind of information to be included in the output data plot.

   - Turn on **Design Variables** to produce a plot showing how the design variable values change during the optimization.
   - Turn on **Scalar Goals** to produce a plot showing progress toward a scalar goal.
You can also see how the scalar numbers change by looking in the Virtuoso® Analog Circuit Optimization Option window at the displayed values for *Initial*, *Prev*, and *Current*.

- Turn on *Functional Goals* to produce a plot, like the example below, showing progress toward a waveform goal.

![Iteration History](image)

If there are too many or too few waveforms displayed in this plot, type the number you want in the *No. of Functional Iterations to Display* field.

4. Set the *Waveform Window* characteristics to the values that work best for you. To make the plot easier to read, for example, you might enlarge the font size and increase the size of the window.

5. Click *OK*.

If the Waveform Window is open, the window changes to reflect the new option settings. If the Waveform Window is closed, it opens with a new plot drawn in accordance with the changed option settings.
Plotting Output Data

If *Auto Plot After Each Iteration* is turned on in the Setting Plotting Options window, the Waveform Window automatically opens and displays the results of each optimization. If the results do not appear automatically, you can use the following procedure to plot them when the optimization ends.

➤ Choose *Results – Plot History* or click *Plot History*.

The Waveform Window appears in the format specified by the Setting Plotting Options window. For more information, see “Setting the Plotting Options” on page 191.

If the plotting options are set so that all the output data is plotted, an output plot might look like this.

![Image of output plot]

Updating Your Design

To copy the optimized variable values back to your schematic,

1. In the Virtuoso® Analog Circuit Optimization Option window, choose *Results – Update Design* or click *Update Design*.

2. In the Virtuoso® Analog Design Environment window, choose *Variables – Copy to Cellview*.

3. In the Virtuoso Schematic Editing window, choose *Design – Check and Save*. 
Saving, Changing, and Loading Session Information

With the Session menu pulldowns on the Virtuoso® Analog Circuit Optimization Option window, you can save the session state, load a saved state, change optimization options, and clear the window of all information.

Saving the Session State

To save the session state (the goals, variables, and options used in the Virtuoso® Analog Circuit Optimization Option window),

1. Choose Session – Save State.
   
   The Saving State form appears.

2. In the Save As field, type a name for the state if you do not want to use the default name.

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.

   Note: Only the information you save is available for retrieval when you reload the saved state.
The optimizer saves the session state in the directory

`~/.artist_states/LibraryName/CellName/.asd_optimization/StateName`

In this directory name, `LibraryName` and `CellName` are derived from the circuit you are optimizing, and `StateName` is the name you specify in the Saving State form.

**Loading a Saved Session State**

To load a saved session state,

1. Choose *Session – Load State*.

   The *Loading State* form appears.

2. From the *Library* cyclic field, choose the library containing the saved state you want to load.

3. From the *Cell* cyclic field, choose the cell containing the saved state you want to load.

4. From the *State Name* field, choose the state you want to load.
5. Turn on buttons to indicate which information you want to use from the saved state.

6. Click OK.

**Saving a Script**

The Open Command Environment for Analysis (OCEAN) command language lets you set up, simulate, and analyze circuit data. OCEAN is a text-based process you can run from a UNIX shell or from the Command Interpreter Window (CIW). You can type OCEAN commands in an interactive session, or you can create scripts containing your commands and load those scripts into OCEAN.

You can use the Corners window to set up the analysis you need, and then save the setup procedure in a script. You can edit the saved script to add simulation or postprocessing commands as needed.

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

To create a script and save it,

1. Choose Session – Save Script.

   The Save Ocean Script to File form appears.

2. In the *File Name* field, specify the name of a file to contain the script.

3. Click OK.

**Changing Optimization Options**

Most users do not need to change the default optimization options. However, if you want to use a specific algorithm or if you want to change the values that control the algorithm, follow these instructions.

1. Choose Session – Options from the Virtuoso® Analog Circuit Optimization Option window.
The Optimization Options form appears.

2. To force the optimizer to use a particular algorithm, select either LSQ or CFSQP in the Algorithm Selection field. If you want the optimizer to choose an appropriate algorithm automatically, select Auto.

The LSQ algorithm is best suited for a pure curve-fitting problem, and Cadence recommends that you use it only for a problem of that kind.

For the LSQ algorithm, the match, maximize, and minimize directions are all equivalent. In each of these cases, the LSQ algorithm works to match the specified Target value. To use the LSQ algorithm for a maximization or minimization problem, you must specify a Target value that is large enough or small enough that the result reaches the maximum or minimum before it reaches the Target value.

When the Algorithm Selection cyclic field is set to Auto, the optimizer uses the CFSQP algorithm in most cases. The optimizer uses the LSQ algorithm only when both of the following conditions are true.

- The Direction for all the enabled goals is match.
- Every enabled goal has a waveform Target.

3. Type values for the Optimizer Control Options you want to change.

- The Percentage Finite Difference Perturbation value affects how sensitivities are determined.
Be aware that some problems are very sensitive to this value and changing it might cause the algorithm to perform poorly.

**Note:** It is recommend that user choose the default value. For advanced users who have better knowledge of the effect of the step length, the *Finite Difference Perturbation* field provides a way to specify the step length that is appropriate. Caution should be taken in using this; some problems are very sensitive to the step length used.

- The *Relative Design Variable Tolerance* value affects the LSQ algorithm stopping criteria. This value has no effect on the CFSQP algorithm.
  
  For example, specifying a value of 0.05 causes the LSQ algorithm to stop when the relative change in each design variable is smaller than 5 percent.

- The *Relative Function Value Tolerance* also affects the algorithm stopping criteria.
  
  For example, specifying a value of 0.05 causes the algorithm to stop when the relative change in each function value is smaller than 5 percent.

**Note:** The *Relative Design Variable Tolerance* and *Relative Function Value Tolerance* fields are designed in a way such that users can stop the algorithm by specifying stopping criteria to be used rather than using the default settings. These fields are entered as absolute numbers. For example, if a user specifies 0.01 in the *Relative Design Variable Tolerance* field, that means if the relative change in the design variables is smaller than 1 percent, the algorithm would stop. Likewise, if 0.01 is specified in the *Relative Function Value Tolerance* field, the algorithm will stop when the relative change in each function value is smaller than 1 percent.

4. Set *Warning Message for Long Simulation*.

   The optimization tool and the Virtuoso® Analog Design Environment are both locked for the duration of the initial simulation run of the optimization. During that initial run, you cannot stop the simulation or monitor the progress. If you want to be warned of a potentially long simulation, leave *Warning Message for Long Simulation* turned on, otherwise, turn it off.

5. Click *OK*.

**Deleting All Setup Information**

To delete all the setup information about goals, variables, and plotting options,

- Choose Session – Reset.
Working through an Extended Example

This section follows an optimization session in detail, demonstrating how you might use the optimizer to improve a real circuit. The example describes how to optimize a Chebyshev filter so that its frequency response matches a specified waveform and its noise output is minimized.

The Chebyshev filter has the following schematic:

![Schematic diagram of a Chebyshev filter with two resistors, R0 and R1.]

Notice the two resistors, R0 and R1. These are the components whose values are optimized during the session.

To follow along with this example, go to a working directory and use a command like the following to copy all the contents of the optimization directory into the working directory.

```
tar -cvhf - -C <install_dir>/tools/dfII/samples/artist optimization | tar -xvf -
```

Then go to the optimization directory you created, start icms, and continue with the following steps.

1. In the CIW, choose Tools – Analog Environment – Simulation.
   
The Virtuoso® Analog Design Environment window appears.

The Choosing Design form appears.

3. In the Library Name field, choose the filterlib library.

4. In the Cell Name field, choose the chebyshev cell.

5. Click OK.


   The Loading State form appears.

7. In the State Name field, choose state1.

8. Click OK.

The Virtuoso® Analog Design Environment window now looks like this.
Generating the Targets

This section explains how to generate two waveforms, which are used as targets for the optimization described later. If this were an actual optimization session, you would probably have existing targets and could skip directly to the optimization step described in “Setting Up and Running the Optimization” on page 203.

You run this initial simulation, which does not involve using the optimizer at all, just as you run other ordinary simulations.

➤ In the Virtuoso® Analog Design Environment window, choose Simulation – Run.
   If the Welcome to Spectre window appears, click OK to close it.

When the simulation finishes, a waveform window appears.

![Waveform Diagram](image)

Close this waveform window.
Saving the Targets

1. In the Virtuoso® Analog Design Environment window, click Results - Save.
2. In the Save Results window, specify `schematic-save` in the Save As field. Click OK.
3. In the Virtuoso® Analog Design Environment window, click Results - Select.
4. In the Select Results window, select `schematic-save` and click OK.
5. In the Virtuoso® Analog Design Environment window, click the Plot Outputs icon.

A waveform window similar to the one generated earlier appears. These two waveforms become the targets for the optimization session described in the next section.

Setting Up and Running the Optimization

This section describes how to set up and run the optimization for the Chebyshev filter.

1. In the Virtuoso® Analog Design Environment window, choose Tools – Optimization.

   The Virtuoso® Analog Circuit Optimization Option window appears.

2. Fill in the Goals and Variables panes with the values that are required for the optimization.

   This step is described in the next section.

Filling in the Goals Pane

In this example, you want to use the optimizer to determine what resistor values will allow you to match the AC Response waveform while minimizing the noise waveform. The following sections describe how to specify the goals that correspond to these waveforms.

Specifying the AC Response Goal

To define the goal corresponding to the AC Response, follow these steps.

1. In the Virtuoso® Analog Circuit Optimization Option window, choose Goals – Add.
The *Adding Goals* window appears.

![Adding Goals Window](image)

2. Enter a name for the goal in the *Name* field.
   
   For this example, type `magVF`.

3. To create the expression, you can use the Waveform Calculator. To open it, click *Open*.


5. In the Waveform Calculator, click `vf` and then go to the Virtuoso Schematic window and select the net connected to `Vout`. Press `Esc` to end the selection.

6. In the Waveform Calculator, click `mag`.
   
   The calculator display now contains the value `mag(VF("/Vout"))`.

7. In the Adding Goals window, highlight the *Expression* field, then click *Get Expression* to copy the expression from the calculator.

8. For this example, you want to match the AC Response waveform, so choose *match* in the *Direction* cyclic field.

9. The *Target* value for this goal is to be the AC Response waveform calculated earlier, as described in “Generating the Targets” on page 202. To specify the waveform, first click `clst` in the calculator to clear the calculator display.
10. In the calculator, click wave, then go to the Waveform Window and click on the AC Response waveform.

   An expression similar to the following appears in the Calculator display:
   \[ \text{mag( VF("/Vout" "/old2/lorenp/simulation/chebyshev/spectre/ schematic-save") )} \]

   This expression represents the AC Response waveform.

11. In the Adding Goals window, select the Target field, then click Get Expression to copy the waveform.

12. Type 5 in the Acceptable field and turn on % within Target.

13. Ensure that Enabled is turned on.

   The Adding Goals window now looks like this.

14. Click Apply.

   The new goal appears in the Virtuoso® Analog Circuit Optimization Option window.

Specifying the Noise Goal

The steps required to define the noise goal are similar to those required for the AC Response goal.
1. If the Adding Goals window is not open, choose **Goals – Add** in the Virtuoso® Analog Circuit Optimization Option window.

2. Type a name for the goal in the **Name** field.

   For this example, type **noise**.

3. Erase any existing information, then type **VN2()** in the **Expression** field.

4. Specify the **Target**, which for this goal is to be the Noise Response waveform calculated earlier, as described in “Generating the Targets” on page 202. To specify the waveform, first open the calculator by clicking **Open** in the Adding Goals window.

5. Click **clst** in the calculator to clear the calculator display.

6. In the calculator, click **wave**, then go to the Waveform Window and click on the Noise Response waveform.

   An expression similar to the following appears in the calculator display:
   
   \[ \text{VN2()} \]

   Type the following path in the brackets:
   
   "\text{/old2/lorenp/simulation/chebyshev/spectre/schematic-save}"  

   This expression represents the Noise Response waveform.

7. Return to the Adding Goals window, select the **Target** field, then click **Get Expression** to copy the waveform.
8. Fill in the other fields of the *Adding Goals* window, as follows.

![Adding Goals Window](image)

9. Click *OK*. The new goal appears in the Virtuoso® Analog Circuit Optimization Option window.

10. You can return to the calculator window and close it by choosing *Window – Close*.

### Filling in the Variables Pane

In this example, you want to optimize the values of two resistors: \( \text{res0} \) and \( \text{res1} \). To prepare for the optimization, you need to set the initial, minimum, and maximum allowed values.

1. In the Virtuoso® Analog Circuit Optimization Option window, choose *Variables – Add/Edit*.

   The Editing Variables form appears.

   ![Editing Variables Form](image)
2. Click on \texttt{res1}, and then fill in the other fields as shown.

![Optimization Variables Must Be Simulation Variables](image)

3. Click \textit{Apply}.

The information about the \texttt{res1} variable appears in the Virtuoso® Analog Circuit Optimization Option window.
4. In the Editing Variables form, click on `res0` and then fill in the other fields as shown.

5. Click *OK*.

   The information about the `res0` variable appears in the Virtuoso® Analog Circuit Optimization Option window.
Running the Optimization

With the goals and variables defined, the Virtuoso® Analog Circuit Optimization Option window looks like this.

At this point in the example, you are ready to run the optimization.

➤ In the Virtuoso® Analog Circuit Optimization Option window, choose Optimizer – Run or click Run Optimizer.

   The optimization starts and the status display updates to reflect the current activity.

Looking at the Output

With the default plotting options, a Waveform Window appears soon after the optimization run begins and updates after each iteration. When this example optimization ends, the Waveform
Window displays the iteration history of the $res_0$ and $res_1$ variables. It also displays the changing waveforms for the $magVF$ and $noise$ goals.

You can change the information that displays in the Waveform Window. For example, to look at the variables in more detail, follow these steps.

1. In the Virtuoso® Analog Circuit Optimization Option window, choose Results – Set Plot Options.
2. In the Setting Plotting Options window, turn off *Display History of Scalar Goals* and *Display History of Functional Goals*, leaving only the *Display History of Variables* selected.

![Display History Options]

3. Click *OK*. 
The Waveform Window appears, showing only the variables.

During the 13 iterations of this optimization (iteration 0 shows the initial values), the value of \( \text{res1} \) went from the starting value of 2.00 K to the final value of 304.7. This final value also appears in the *Current* column of the *Variables* pane in the Virtuoso® Analog Circuit Optimization Option window.

To look at the goals in more detail,

1. In the Setting Plotting Options window, turn on *Display History of Functional Goals*.
2. Turn off *Display History of Variables* and *Display History of Scalar Goals*.
3. Set *No. of Functional Iterations to Display* to 1.
4. Click *OK*. 
The Waveform Window appears or redraws with only the goals showing. For example, the noise goal looks like this.

![Noise Goal Diagram]

The last iteration, I#13, shows that the optimizer was able to lower the noise below the target throughout the entire frequency range.

The optimized values for the two resistors are displayed in the current column of the Virtuoso® Analog Circuit Optimization Option window: resil has an optimized value of 304.7 and res0 has an optimized value of 1.311 K. To use these values in your design, do the following:

- In the Virtuoso® Analog Circuit Optimization Option window, choose Results – Update Design.
The new values appear in the *Design Variables* pane of the Virtuoso® Analog Design Environment window.

```
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>res1</td>
<td>304.7</td>
</tr>
<tr>
<td>2</td>
<td>res0</td>
<td>1.311K</td>
</tr>
<tr>
<td>3</td>
<td>CAP1</td>
<td>110n</td>
</tr>
<tr>
<td>4</td>
<td>CAP0</td>
<td>260n</td>
</tr>
<tr>
<td>5</td>
<td>CAP</td>
<td>500f</td>
</tr>
</tbody>
</table>
```
Index

Symbols

% within Target 174, 176
.cdssinit file
  example 70
  loading PCFs and DCFs from 16, 21, 42

A

Acceptable values
  creating waveform objects for 178
  entering an expression for 173, 176
  restrictions on 173, 176
  setting as percentage of Target 174, 176
  valid 173, 176
  weights, using to assign 182
Add button (Monte Carlo) 87, 91
Add Corner button (Corners) 17
Add Measurement button (Corners) 19, 26, 27
Add Process dialog box 59
Add Variable button (Corners) 17
Add/Edit Goals button (optimization) 172, 174, 177
Add/Edit Variables button (optimization) 186, 188

B

Add button (Monte Carlo) 87, 91
Autoplot After Each Iteration button (optimization) 192, 194
Autoplot button (Monte Carlo) 91
autoplot field 86

appending scalar output data to saved
data 84

B

Best Fit Line 123
button bar (Monte Carlo) 87
C

button bar (Monte Carlo) 87

Corners
  Add Corner 17
  Add Measurement 19, 26, 27
  Add Variable 17
  Calculator 19, 27
  Copy Corner 17
  Delete Measurement 19, 28
  Delete Row 26
  Delete Selected / Delete Corner / Delete Row 17
  Get Expression 19, 27
  Plot/Print 29
  Run 28
  Run / Stop 17

Monte Carlo
  Add 87, 91
  Autoplot 91
  Calculator 87, 92
  Change 87, 92
  Clear 87
  Delete 87, 92
  Density Estimator 120
  Get Expression 87
  Load 112, 114

optimization
  Add/Edit Goals 172, 174, 177
  Add/Edit Variables 186, 188
  Autoplot After Each Iteration 192, 194
  Delete (design variables) 189
  Delete (goals) 178
  Design Variables 192
  Enabled (goals) 174, 177, 187, 188
## C

**calculator**
- building expressions with (Monte Carlo) 92
- building goal expressions with (optimization) 174
- creating waveforms with (optimization) 179
- opening 175

**Calculator button**
- Corners 19, 27
- Monte Carlo 87, 92

**.cdsinit file** 70

**.cdsinit file, using to load PCFs and DCFs** 16, 21, 42

**cdsSpice simulations**
- and select all options 98

**CFSQP algorithm, data suited for** 162

**Change button (Monte Carlo)**
- description 87
- example of use 92

**Chebychev filter**
- description 200

**Clear button (Monte Carlo)** 87

**columns, selecting in Corner Definitions pane** 16

**conditional yield**
- definition 125
- reports, description 116
- reports, printing 128

**Conditional Yield form** 129

**Copy Corner button** 17

**copying an existing corner** 17

**Corners**
- analysis
  - definition 9
  - extended example 62
  - overview 9
  - starting 28
  - stopping 28
- description 179
- log file 20

**corSetModelFile procedure**
- used only with single model library style 38
- using to enter model file name 43

**correlation tables**
- description 115
- printing 118

**Current field** 168, 169

**currents, saving all** 97

## D

**data**
- filter
  - reloading settings 112
  - saving settings 112
  - specifying settings 111
  - turning off 112
  - outlying error values, filtering out 110
  - saving between runs 89

**Data Filter form** 111

**DCFs. See design customization files (DCFs)**

**debugging PCFs and DCFs** 38

**Delete button (Monte Carlo)** 87, 92

**Delete button (optimization)**
- for design variables 189
- for goals 178

**Delete Measurement button (Corners)** 19, 28

**Delete Row button (Corners)** 26

**Delete Selected / Delete Corner / Delete Row button (Corners)** 17

**deleting**
- corners 25
- design variables 188
- goals 178
- processes 59
- simulation results 191

user-defined corners 17
variables or groups 26
Density Estimator button (Monte Carlo) 120
Design – Check and Save 194
design customization files (DCFs)
.cdinit, loading with 42
commands normally placed in 38
debugging with OCEAN 38
loaded after PCFs 42
loading from the graphical user interface 21
tailoring a Corners analysis with 21
use of 37, 40
design variables
adding to DCF 40
deleting all 199
deleting specific 188
determining sensitivities of 163
editing 187
enabling or disabling 189
examples of 186
names, as displayed in Corner Definitions pane 16
pane showing values of 169
plotting 192
setting maximum value for 187, 188
setting minimum value for 187, 188
stopping criteria for 199
sweeping 89
updating schematic, with optimized 194
Design Variables button 192
design, updating with optimized values 194
device descriptions, for Monte Carlo 78
direction
specifying for Chebychev filter example 204
specifying for goals 173, 175
Direction field 168
disabling
design variable 189
goals 178
disk storage requirements, reducing 89
distribution concentration, estimating 120

E
Edit – Add Measurement 26, 27
Edit – Delete Corner 25
Edit – Delete Measurement 28
Edit – Delete Row 26
edit fields
clearing 87
description 86
design variables 187
goals 177
Editing Variables form 207
Enabled button
for goals 174, 177, 187, 188
Enabled field 168, 169
enabling
design variables 189
goals 178
error messages (Corners) 19
example, extended optimization 200
Expression column, in Corners Performance Measurements pane 18
expressions
adding 87
adding by typing in 91
changing 87, 92
checking validity 98
creating with the calculator 204
deleting 87, 92
entering in Performance Measurements pane 18
getting from the calculator 204
listed in the Outputs pane 85
names for 85
retrieving from calculator (Corners) 19
retrieving from calculator (Monte Carlo) 87
used to specify goals 171
using calculator to build 92
extended examples
Corners analysis 62
Corners, PCF for 68
Monte Carlo 130

F
family of curves
for Monte Carlo extended example 145
plots, description of 115
plotting 121
saving data for 84, 89
feasible initial values, determined by CFSQP
algorithm 163
File – Close 11
File – Load 21
File – Save Ocean Script 36
File – Save Setup 34
File – Save Setup As 34
Filter By dataset 111
Filter By point 111
filter, Chebychev 200
filtering, turning off 112
folded cascode, schematic 63
forms and windows
Add Process (Corners) 59
Adding Goals 172, 175, 204, 205, 207
Conditional Yield 129
Correlation Table 118
Data Filter 111
Editing Goals 177
Editing Variables 187, 188, 207
Histogram 120
Iteration Versus Value 117
Load (Corners) 22
Load Data Filter Values 113
Load Specification Limits 115
Loading State 196
Monte Carlo Load 104
Monte Carlo Save Ocean Script 105
Multiconditional Yield 127
Optimization Options 198
Process/Model Info Setup (Corners) 61
Run for Fixed Number of Iterations 190
Save Changes? (Corners) 22
Save Data Filter Values 112
Save Ocean Script (Corners) 37, 197
Save Results 102
Save Specification Limits 114
Saving State 195
ScatterPlot 123
Select Results 109
Setting Plotting Options 192, 212
Simple Yield 126
Specification Limits 113
Virtuoso analog corners analysis 12
Virtuoso analog optimization analysis 165, 175
Virtuoso analog statistical analysis 80
Virtuoso® analog corners analysis 10
Virtuoso® analog statistical analysis 80
Waveform 180, 193, 202
frequency response, matching 200
Functional Goals button 193
G
GAUSS function, used in Monte Carlo 134
Get Expression button (Corners) 19, 27
Get Expression button (Monte Carlo) 87
goals
definition 171
deleting all 199
deleting specific 178
editing 177
enabling or disabling 178
example of specifying 203
saving 195
specifying direction for 173, 175
specifying expression for 173, 175
specifying for Chebychev filter
turned off 112
example 205
specifying name for 175
Goals – Add 172, 174, 203, 206
Goals – Delete 178
Goals – Disable 178
Goals – Edit 177
Goals – Enable 178
Goals pane 168
graphical user interface
Corners 12
Monte Carlo 80
optimization 165
groups
deleting 26
name, as displayed by Corners 16
H
help
for optimization 163
Histogram form 120
histograms
description 115
for Monte Carlo extended example 140
plotting 119
I
individual yield 125
information messages, as displayed by
Virtuoso Advanced Analysis Tools User Guide

Corners 20
Initial field 168, 169
initial values, determining feasible values for 163
input files, creating by hand 103
iteration history 211
iteration versus value tables
description 115
printing 116
Iteration Versus Value window 117

L
least squares fit lines, for scatter plots 123
Load button (Monte Carlo) 112, 114
Load Data Filter Values form 113
Load dialog box 22
Load Specification Limits form 115
loading
session state 104
stored outputs 108
Lower column, in Performance Measurements pane 19
lowpass filter
description 130
model file 133
schematic 130
LSQ algorithm
conditions of use for 198
data suited for 162
equivalence of match, minimize, and maximize 182

M
Max field 169
maximize
equivalent to match for LSQ algorithm 182
Target used only for weighting when chosen 181
minmax, algorithm for 162
Mismatch Only variation type 88
model files
connecting to Analysis Variation cyclic 106
for Monte Carlo 78
lowpass filter 133
model files, purpose 37
modeling styles
multiple model library 49
multiple numeric 53
multiple parametric 55
single model library 43
single numeric 52
supported by Corners analysis tool 42
Monte Carlo analysis
Analysis Setup pane 83
analyzing results 139
button bar 87
edit fields 86
extended example 130
graphical user interface for 80
menu 81
number of runs, specifying 88
Outputs pane 85
overview 77
requirements for running 78
results, analyzing 108
specifying analysis variation type for 88
specifying characteristics for 83
specifying initial run number for 88
starting 101
status display  81
stopping  101
Monte Carlo Load form  104
Monte Carlo Save Ocean Script form  105
Monte Carlo tool
closing  106
starting  78
multiconditional yield
definition  125
reports, description  116
reports, printing  127
Multiconditional Yield form  127
multiple model library style
described  49
requirements for using  62
multiple numeric modeling
described  53
example of  66
PCF for  54
multiple parametric modeling
described  55
PCF for  56

N
Name field (goals)  168
Name field (variables)  169
No. of Functional Iterations to Display  213
noisy data, best optimized with LSQ
algorithm  162
number of runs, specifying  84

O
OCEAN script, saving  105
Open button  175
opening the Virtuoso® analog optimization
analysis window  203
optimization
definition  161
extended example  200
help for  163
output, changing what is displayed
in  212
overview  162
setting options for  197
setting up and running  203
steps followed during  162
using waveform goals for  204
optimized values, where displayed  214
optimizer
control options, setting  198
definition  161
running  189
stopping  190
Optimizer – Reset  191
Optimizer – Run  189, 210
Optimizer – Run n  190
Optimizer – Step  189
Optimizer – Stop  190
Optimizer – Stop Now  190
options
for optimization  197
for plotting  191
outlying data, filtering  110
output data, appending to saved data
84
output formats
choosing (Corners)  28
residual plot example (Corners)  32
text example (Corners)  30
output information for optimization,
changing  211
output log, viewing  106
outputs
choosing after the analysis runs  29
loading  108
of the Chebychev filter example  210
saving all  97
saving to specific file  102
Simulation window outputs appear in
Corners  70
Outputs – Retrieve Outputs  90
Outputs – Save All  97
Outputs – To Be Saved – Select On
Schematic  98
Outputs pane  85

P
parameter storage format (PSF)
directory, copying  102
files, saving between runs  84
parameters
for Spectre, must be in main circuit  62
showing correlations among  118
sweeping  89
pass/fail histograms  119
PCFs. See process customization files
(PCFs)
% within Target 174, 176
percentage finite difference perturbation 198
performance measurements
adding to DCFs 40
creating by hand 26
creating with calculator 27
deleting 27
pane 18
Plot/Print button 29
plotting
  automatic 86
  histograms 119
  options, resetting all 199
  options, saving 195
  output data 194
  progress toward scalar goals 192
  progress toward waveform goals 193
  setting options for 191
  setting Waveform Window characteristics 193
  specifying number of waveforms to display 193
  waveforms 180
Prev field (for goals) 168
Prev field (for variables) 169
process customization files (PCFs)
  .cdsinit, loading with 16, 42
  commands normally placed in 38
  creators of 37, 38
  debugging with OCEAN 38
  example 39
  for multiple numeric modeling 54
  for multiple parametric modeling 56
  for single file modeling 44, 50
  for single numeric modeling 52
  loaded before DCFs 42
  loading from DCF 42
  loading from the graphical user interface 21
  used for predefined corners 21
process field 15
Process Only variation type 88
process variables
  adding in Corners window 59
Process Variation and Mismatch variation type 88
processes
  defining in Corners window 58
  modifying in Corners window 60
  name of 15
PSF (parameter storage format)
  directory, copying 102
  files, saving between runs 84
R
  red colored messages 19
  relative design variable tolerance 199
  relative function value tolerance 199
  reloading a session state 196
  requirements for running Monte Carlo tool 78
  Resetting Corners 26
  residual plot 32
  Results – Evaluate Expressions 110
  Results – Filter 111, 112
  Results – Plot – Curves 121, 145
  Results – Plot – Histogram 119, 140
  Results – Plot – Scatter Plot 122
  Results – Plot History 194
  Results – Plotting Options 191
  Results – Print – Correlation Table 118
  Results – Print – Iteration versus Value 116
  Results – Print – Iteration vs. Value 139
  Results – Save 102
  Results – Select 108
  Results – Set Plot Options 211
  Results – Specification Limits 113, 143
  Results – Update Design 194, 214
  Results – Yield – Multiconditional 127
  Results – Yield – Simple 125, 144
  results, plotting 191
Run / Stop button (Corners) 17, 28
Run Optimizer button 210
run temperature, always appears in Corners window 70
running
  Monte Carlo analysis 101
  optimization
    for a fixed number of iterations 190
    for one iteration 189
    major steps in 170
    until stopping criteria are met 189
S
  Save All command 97
  Save Changes? dialog box 22
Save Data Between Runs to Allow Family Plots button 110
Save Data Filter Values form 112
Save Ocean Script form 37, 197
Save Results dialog box 102
Save Specification Limits form 114
saving
  all currents 97
  all node and terminal values 97
  all voltages 97
Monte Carlo session state 103
optimization session state 195
Scalar Goals button 192
scalar output data
  analyzing 139
  appending to saved data 89
data type 86
scatter plots
  description 115
  plotting 122
ScatterPlot form 123
schematics
  for Chebychev filter example 200
  for folded cascode example 63
  updating with optimized values 194
Select Results dialog box 109
sensitivities
  changing default values for 198
  how determined 163
Session – Load State (Monte Carlo) 104
Session – Load State (optimization) 196
Session – Options (optimization) 197
Session – Quit (Monte Carlo) 106
Session – Quit (optimization) 164
Session – Reset (optimization) 199
Session – Save Script (Monte Carlo) 105
Session – Save State (Monte Carlo) 103
Session – Save State (optimization) 195
session state
  loading (Monte Carlo) 104
  loading (optimization) 196
  saving (Monte Carlo) 103
  saving (optimization) 195
Set By limits (Data Filter form) 111
Set By limits (Specification Limits form) 114
Set By sigma (Data Filter form) 111
Set By sigma (Specification Limits form) 114
Setting Plotting Options window 212
Setup – Add Process 58
Setup – Add/Update Model Info 60
setup information
  deleting all 199
  saving to default files 34
  saving to specified files 34
signals
  adding 87, 91
  changing 87
  deleting 87, 92
Simple Yield form 126
simple yield reports
  description 115
  printing 125
Simulation – Check Expressions 98
Simulation – Create Input Files 103
Simulation – Output Log 106
Simulation – Run 28, 101, 137, 202
Simulation – Stop 28, 101
simulations
  outputs, saving all 97
  specifying number to run 84
simulator, choosing 79
single model library style
  described 43
  PCF for 44, 50
  requirements for using 62
single numeric modeling
  described 52
  PCF for 52
SKILL PI commands, using in PCFs and DCFs 38
sorting Monte Carlo outputs 117
Special Functions button 179
specification limits
  for Monte Carlo extended example 144
  saving 114
  set by limits 114
Specification Limits form 113
Spectre simulator, using with Corners 62
standard deviations 111
standard histogram 119
starting
  Corners analysis 28
  Monte Carlo analysis 101
  Monte Carlo tool 78
  optimization 189
starting run number, specifying 84, 88
statistical values, for Monte Carlo 78
statistical variation, specifying which to run 84
statistics block example 106
   using with Spectre simulator 106
status display (Corners) 19
status display (Monte Carlo) 81
Stop Optimizer button 190
stopping
   Corners analysis 28
criteria for optimization 163, 199
   Monte Carlo analysis 101
   optimization 190
Suppress field
   Conditional Yield form 129
   Multiconditional Yield form 127
swept parameter, specifying 84, 89

T

table function, using in optimization 179
Target column, in Performance Measurements pane 18
Target values
   creating waveform objects for 178
   field 168
   how used by optimizer 181
   how used to assign weights 182
   relation to Acceptable values 173, 176
   setting Acceptable values as percentage of 174, 176
   used as goal 181
   valid 173, 175
Temperature, sweeping 89
text
   output example (Corners) 30
tool bar (optimization) 170
Tools – Calculator 27
Tools – Corners 11
Tools – Get Expression 27
Tools – Monte Carlo 135
Tools – Optimization 164, 203
Tools – Plot or Print Outputs 29
total yield 125

U

unknown data 86
Update Design button 194
Upper column, in Performance Measurements pane 19

useAltergroup variable
   value to use with Spectre 62
V

variables
   adding (Corners) 25
deleting (Corners) 25, 26
display of, by optimizer 213
   setting for Chebychev filter example 207
Variables – Add/Edit 186, 188, 207
Variables – Copy to Cellview 194
Variables – Delete 189
Variables – Disable 189
Variables – Enable 189
Variables pane 169
variants
   defined in modeling file 39
Virtuoso® analog corners analysis window
   closing 10
   opening 10
Virtuoso® analog optimization analysis window
   closing 164
   opening 164
Virtuoso® analog optimization analysis window
   description 165
   opening 203
Virtuoso® analog statistical analysis window 80
voltages, saving all 97

W

waveform data
   creating mcdata file from 110
data type 86
waveform window 202
waveforms
   creating 178
generating 202
   plotting 180
   using as an optimization goal 204
weights
   determined by Target and Acceptable together 182
   formula for 182
   windows. See forms and windows
X

X and Y values, creating waveforms from 178

Y

yellow colored messages 20
Yield – Conditional 128
yields
analyzing 143
conditional
description 125
printing report on 128
individual
description 125
printing report of 125
multiconditional
description 125
printing report on 127
simple, printing report on 125
total 125