

Analyzing Designs Using SaberDesigner

Release 5.1

Avant! Corporation

Material contained in this manual is confidential information of Avant! Corporation.

Copyright © 1985 - 2001 Avant! Corp. Unpublished—rights reserved under the copyright laws of the United States of America.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in Subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, FAR 52.227-14(g), or FAR 52.227-(19), as applicable.

Avant! Corp.
9205 S.W. Gemini Drive
Beaverton, OR 97008

U.S. Patent Nos. 4,868,770 (Canadian Patent No. 1,323,930); 4,985,860 (Canadian Patent No. 1,319,989); 5,046,024; 5,199,103; 5,404,319; 5,548,539. Other U.S. and foreign patents pending.

Analogy[®], AnalogyHDL[®], Avant! Corp., Calaveras[®] Algorithm, DesignStar[®], Frameway[®], Hypermodel[®], InSpecs[®], MAST[®], PowerExpress[®], TheHDL[®], and WaveCalc[®] are registered trademarks of Avant! Corp. AHDL[®] is a registered trademark of Avant! Corp. in the U.K. and Germany. SABER[®] is a registered trademark of American Airlines, Inc., licensed to Avant! Corp. BookHDL[™], Cosmos-Scope[™], DesignerHDL[™], FastMAST[™], FASTPARTS[™], GuideHDL[™], HydraulicExpress[™], iQBus[™], ModelExpress[™], SaberBook[™], SaberDesigner[™], SaberGuide[™], SaberHarness[™], SaberLink[™], SaberScope[™], SaberSketch[™], ScopeHDL[™], SketchHDL[™], TelecomExpress[™], Testify[™], and VeriasHDL[™] are trademarks of Avant! Corp. Avant!, Avant! logo, AvanLabs, and avanticorp are trademarks and service-marks of Avant! Corporation. Adobe, Acrobat, and the Acrobat logo are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

Other trademarks are property of their respective owners.

UNIX[™] is a registered trademark of AT&T.

Windows NT is a trademark of Microsoft Corporation.

Notice to Users: Read Before Using

Disclaimer

The Saber® simulator and all other software products offered to the customer (Customer) by Avant! Corporation. (Avant! Products) are software programs designed and engineered by Avant! Corp. or its licensors. Avant! Products incorporate the latest ideas and theories in advanced technologies to assist an experienced, highly qualified engineer in the design of Customer's product. Avant! Products have been designed to provide limited analysis and are not intended to replace testing or other analysis of Customer's product. The advanced level engineer must use Avant! Products only as "tools" to help understand the potential performance of Customer's product while continuing to rely upon his or her own expertise.

Avant! Products are based on mathematical abstractions of real world environments or devices. Because Avant! Products work with abstractions, the use of Avant! Products can only provide an estimation of possible future performance of Customer's product. Customer acknowledges these limitations and agrees that it will not rely solely upon the results derived from any usage of Avant! Products in determining the final design, composition, structure, safety, reliability, or performance of any Customer product. Avant! will not be legally liable for any use of Avant! Products by Customer, including Customer misuse, errors in judgment or application of Avant! Products in the development or use of Customer's products.

Avant! does not represent that it has sufficient professional expertise in the application of Customer's product to determine the difference between the performance predicted by Avant! Products and the performance of any real product or device. Therefore, Avant! expects Customer will test Customer's product prior to any actual use. Where the failure of Customer's product may result in bodily harm, Customer must take additional measures, in accordance with the applicable standard of care, to ascertain that Customer's product conforms to Customer's specifications and standard engineering design practices. Regardless of end use application, any Customer product developed with the assistance of Avant! Products must be tested independently to confirm the results obtained from Avant! Products. Where a higher standard of care is required, Avant! recommends that the results of simulation obtained from Avant! Products be confirmed by outside independent professionals, including additional product testing and quality control.

Preface

What You Need to Know to Use This Manual

This manual assumes that you are familiar with the following procedures:

- How to create a schematic in your design capture tool
- How to view the contents of a file
- How to use a text editor to create a file or edit the contents of a file

If this is your first time using SaberDesigner, you should run through the tutorial presented in the *Getting Started with SaberDesigner* document. This manual provides an excellent overview of the process described in this chapter.

What This Manual is About

In this manual you will find information about how to analyze a design using SaberDesigner from schematic capture, to executing analyses, through tuning parameter values.

SaberDesigner is an extremely powerful set of tools that allow you to analyze your design in numerous ways. Thus allowing you to save design time, lower production costs and create a more profitable product.

Conventions

This manual uses the following conventions:

template name Template names and symbol names are shown using the bold font.

`command` Command names, program names, node names, netlist entries, parameters, property names, system variable names, etc. are shown using a Courier font. The bold/Courier is used to denote the significant characters of a particular command.

Preface

bold/Helvetica	Menu choices and button names are shown in bold/Helvetica font.
<code>~/tutorial/ci rcuit</code>	File and directory paths appear in the Helvetica font.
<i>fieldname</i>	An item shown in italic in a command line indicates that you must supply the specific text to be used in its place.
<code>*req*</code>	The term <code>*req*</code> indicates that you must supply a value for the given argument.
Helvetica	Dialog box names, menu names, form names, and form field names appear in the Helvetica font.
<i>emphasis</i>	Emphasized text such as a <i>manual title</i> appears in italic.
[]	Italicized square brackets enclosing text indicate optional entries.
Single Click	Press and release a mouse button once quickly.
Press and Hold	Press a mouse button and do not release it.
Double Click	Press and release a mouse button twice quickly.

The following convention is used in this manual to indicate a menu selection from a sub-menu as shown in the following example:

Analyses > Operating Point > DC Transfer

In this example, the **DC Transfer** menu item is selected by first selecting the item **Analyses** from a top-level menu, then selecting **Operating Point** from a sub-menu, and finally selecting **DC Transfer** from a second sub-menu.

Revision History

November 1999	Updated text for Release 5.1.
March 1999	Updated text for Release 5.0.
October 1998	Added Appendix F, "Simulating Digital Parts in Saber," as well as additional information on digital parts and Hypermodels in Chapter 1, "Capturing the Design with SaberSketch."
March 1998	Update text to reflect changes to the user interface for Release 4.3.
July 1997	Incorporated information about Linear Systems Analysis and s Domain Measurements.
December 1996	Miscellaneous updates for Release 4.1.1.
October 1996	First Release accompanying Release 4.1.

Preface

Table of Contents

Preface	i
Chapter 1. Capturing the Design with Sketch	1-1
Invoking Sketch.....	1-2
Opening a Schematic Window - Sketch.....	1-2
Choosing Models - Sketch	1-3
Choosing and Placing Symbols on a Sketch Sheet.....	1-4
Placing Symbols on a Sketch Schematic - Overview.....	1-4
Using the Parts Gallery to Place a Supplied Part - Sketch	1-5
Using the Dialog Navigator to Place a Symbol - Sketch.....	1-7
Moving Symbol Instances and Specifying the Instance Name - Sketch.....	1-7
Using Shared Symbols - Sketch	1-8
Using Split Symbols - Sketch	1-9
Adding MAST Power and Simulation Stimulus Parts - Sketch.....	1-10
Including MAST Digital Parts in Designs - Sketch	1-11
Adding MAST Hypermodels - Sketch	1-11
Connecting a Part to a Node of the Proper Connection Type - Sketch	1-12
Annotating Properties - Sketch	1-13
Elements of a Property - Sketch	1-13
Modifying Properties - Sketch.....	1-14
Obtaining Help on a Property - Sketch	1-15
Required Property Changes on Supplied MAST Parts - Sketch	1-15
Applying Values to Characterize a Template in Sketch - Overview	1-16

Table of Contents

Specifying MAST Property Values Globally - Sketch	1-17
Defining and Passing MAST Parameters - Sketch.....	1-18
Creating Composite MAST Symbol Properties in Sketch	1-20
Wiring the Schematic - Sketch	1-21
Drawing a Wire - Sketch	1-22
Rewiring the Schematic - Sketch	1-24
Naming a Wire - Sketch	1-24
Using Other Methods to Connect Wires - Sketch	1-25
Wiring the Schematic Using Busses - Sketch	1-26
Drawing a Bus - Sketch.....	1-26
Naming a Bus and Designating Bus Width - Sketch	1-28
Creating a Bus Port - Sketch	1-29
Creating a Bus Indicator - Sketch	1-30
Generating a Symbol for a Schematic Block - Sketch	1-30
Adding Borders - Sketch	1-31
Saving the Sketch Design	1-33
Chapter 2. Creating a Sketch Symbol	2-1
Opening a Symbol Editing Window	2-2
Drawing the Symbol Graphics.....	2-2
Specifying Symbol Views	2-3
Adding and Naming Symbol Ports	2-3
Associating the Symbol with a Lower-Level Description	2-4
Adding Symbol Properties	2-6
Creating Online Symbol Help.....	2-7
Saving a Symbol.....	2-8
Adding a Symbol to the Parts Gallery.....	2-8
Chapter 3. Simulating Sketch Designs	3-1
Specifying the Top-level Sketch Schematic.....	3-1
Specifying Netlister and Simulator Invocation Options - Sketch.....	3-2
Simulating a Sketch Design with Saber	3-3
Viewing Sketch Design Analysis Waveforms.....	3-5
Specifying Sketch Nodes or Pins to Create Waveforms	3-5
Viewing Signals Internal to a Template- Sketch	3-6

Adding DesignProbes to a Sketch Wire or Pin.....	3-7
Making Changes to a Sketch Design.....	3-8
Changing Saber Property Values in Sketch	3-8
Changing Property Values in Saber	3-8
Back Annotating DC Values in a Sketch Design.....	3-9
Exiting Saber - Sketch	3-10
Chapter 4. Finding and Debugging DC Operating Points	4-1
Performing DC Analysis	4-2
Evaluating the Operating Point	4-4
Determining the Next Step.....	4-5
Debugging DC Analysis Results.....	4-6
Hints for Finding Difficult DC Operating Points	4-6
Simulator Found the Wrong Operating Point	4-8
Simple Circuits That are Hard for DC Analysis	4-10
Chapter 5. Checking Time-domain Response-Process Overview	5-1
Specifying the First Transient Data Point.....	5-1
Performing Transient Analysis	5-2
Extending a Transient Analysis.....	5-5
Viewing the Transient Analysis Results.....	5-6
Measuring the Analysis Results.....	5-8
Determining the Next Step.....	5-11
Checking the Frequency Spectrum of a Time-Domain Signal.....	5-12
Performing Fourier Analysis	5-13
Performing FFT Analysis.....	5-17
Chapter 6. Sweeping Independent Sources	6-1
Executing DC Transfer Analysis.....	6-1
Viewing the DC Transfer Results.....	6-4
Determining the Next Step.....	6-6
Chapter 7. Special Topics in Transient Analysis	7-1
Starting Oscillators in Transient Analysis.....	7-1
Starting Oscillators by Modifying the DC Initial Point.....	7-3

Table of Contents

Starting Oscillators By Using Current and Voltage Sources	7-4
Keeping Oscillators Going	7-5
Eliminating Spikes in Transient Analysis.....	7-5
Chapter 8. Checking the Frequency Response-Process Overview	8-1
Introduction to Small-Signal Frequency Analyses	8-1
Determining the Circuit Bias Point	8-2
Executing AC Analysis.....	8-2
Viewing the Small Signal Analysis Results.....	8-6
Measuring AC Analysis Results.....	8-8
Determining the Next Step.....	8-11
Transforming Frequency-Domain Signals to the Time-Domain (ifft).....	8-11
Chapter 9. Calibrating Analyses	9-1
Calibration Overview	9-1
Calibrating DC Analysis.....	9-2
Calibrating DC Transfer Analysis.....	9-3
Calibrating Transient Analysis	9-3
Calibrating Small Signal Analyses	9-5
Calibrating Fourier, FFT, and IFFT Analyses	9-6
Chapter 10. Using Two-Port, Noise and Distortion Analyses	10-1
Two-Port Analysis Overview	10-2
Determining Small-Signal Transfer Function - Process Steps.....	10-2
Noise Analysis Overview	10-4
Checking Noise Specifications	10-5
Distortion Analysis Overview.....	10-9
Checking Distortion Specifications	10-9
Viewing the Distortion Analysis Results	10-12
Analyzing the Distortion Results	10-13
Chapter 11. Determining Parameter Sensitivity	11-1
Sweeping Parameters - Process Steps.....	11-1
Sweeping Parameters Example - Analyzing a Design over a Temperature Range	11-5

Determining Component Sensitivity.....	11-6
Performing Nominal Simulations	11-7
Performing Sensitivity Analysis	11-8
Generating the Sensitivity Report.....	11-10
Evaluating the Sensitivity Report	11-11
Chapter 12. Tuning Design Parameters with Statistical Analysis	12-1
Specifying the Parameter Tolerance Ranges	12-1
Executing the Monte Carlo Analysis.....	12-3
Viewing the Monte Carlo Results.....	12-6
Determining the Next Step after a Monte Carlo Analysis.....	12-7
Chapter 13. Determining Component Stress Levels	13-1
Collecting Part Data.....	13-1
Specifying Stress Ratings on Components.....	13-2
Specifying Derating Values.....	13-3
Performing a Nominal Simulation	13-4
Performing the Stress Analysis	13-5
Viewing the Stress Report	13-6
Analyzing the Stress Report	13-7
Chapter 14. Checking Linear Systems Analysis Specifications	14-1
Determining the Poles and Zeros of a System.....	14-1
Determining the Circuit Operating Point	14-2
Executing Pole-Zero Analysis.....	14-2
Viewing Pole-Zero Analysis Results.....	14-4
Measuring Pole-Zero Analysis Results	14-7
Determining the Next Step	14-9
Determining Time Domain Response Using Pole-Zero Data.....	14-10
Determining Poles and Zeros - Linear Time Domain	14-10
Executing Linear Time Domain Analysis (tresp).....	14-10
Determining Frequency Response Using Pole-Zero Data.....	14-12
Determining Poles and Zeros - Frequency Response	14-12
Executing Frequency Response Analysis (fresp).....	14-13

Table of Contents

Chapter 15. Determining Fault Conditions	15-1
Testify Process Steps	15-1
Using the Netlister with Testify	15-2
Displaying the Testify Form	15-2
Setting Up the Tests	15-3
Determining Nominal Operation and Operational Limits	15-4
Specifying the Faults	15-5
Running the Fault Simulations	15-7
Displaying the Testify Test Results	15-8
Debugging Fault Runs that Don't Converge	15-9
Using the PinFault Editor	15-11
Using Testify with Hierarchy	15-11
Inserting Faults on a Hierarchical Symbol	15-12
Inserting Faults within a Hierarchical Symbol	15-12
Testify Fault Wrappers	15-13
Appendix A. Files Used by Designer Tools	A-1
Preference Files.....	A-1
Startup Files.....	A-6
Log Files.....	A-7
Project Files	A-8
Report Files	A-8
Appendix B. Netlister Command Reference	B-1
Saber Netlister Command Reference.....	B-1
Netlister Usage - SaberDesigner.....	B-1
shtos Command Line - SaberDesigner	B-3
Appendix C. Simulating Digital Parts in Saber	C-1
Hypermodels - Overview.....	C-1
Using Default Hypermodels	C-2
Using Ideal Hypermodels	C-2
Using Technology-Specific Hypermodels.....	C-3
Hypermodel Filenames and Logic Families.....	C-4
Creating Part Number-Specific Hypermodels	C-4

Using Multiple Hypermodel Families in a Design C-6

Selecting Hypermodels from the Saber/Netlister Settings Form C-7

Net Re-naming Due to Hypermodel Insertion..... C-8

Appendix D. Stress Analysis Concepts D-1

 Safe Operating Area (SOA) Curves..... D-1

 Thermal Effects Properties..... D-2

 Thermal Effects for Parts without Heat Sinks..... D-3

 Thermal Effects for Parts with Heat Sinks D-4

 Specifying Thermal Effects Properties..... D-4

 Setting Ambient Temperatures in a Design D-5

Appendix E. Hierarchical Parameter Passing E-1

Index Index-1

Table of Contents

Capturing the Design with Sketch

Before capturing the design it is assumed that all the required models are ready to access from a library. The next step in the design analysis process is to capture the circuit.

The following topics describe the process of creating schematics and symbols with Sketch:

1. Invoking Sketch
2. Opening a Schematic Window - Sketch
3. Choosing Models - Sketch
4. Choosing and Placing Symbols on a Sketch Sheet
5. Annotating Properties - Sketch
6. Wiring the Schematic - Sketch
7. Generating a Symbol for a Schematic Block - Sketch
8. Adding Borders - Sketch
9. Saving the Sketch Design

When using the Saber simulator you can include supplied or user-defined parts in a Sketch schematic including analog, digital, mechanical, and hydraulic technologies from the extensive MAST part library. To aid you in locating parts, the Parts Gallery tool allows you to search the part library by part description, symbol name, or MAST template name.

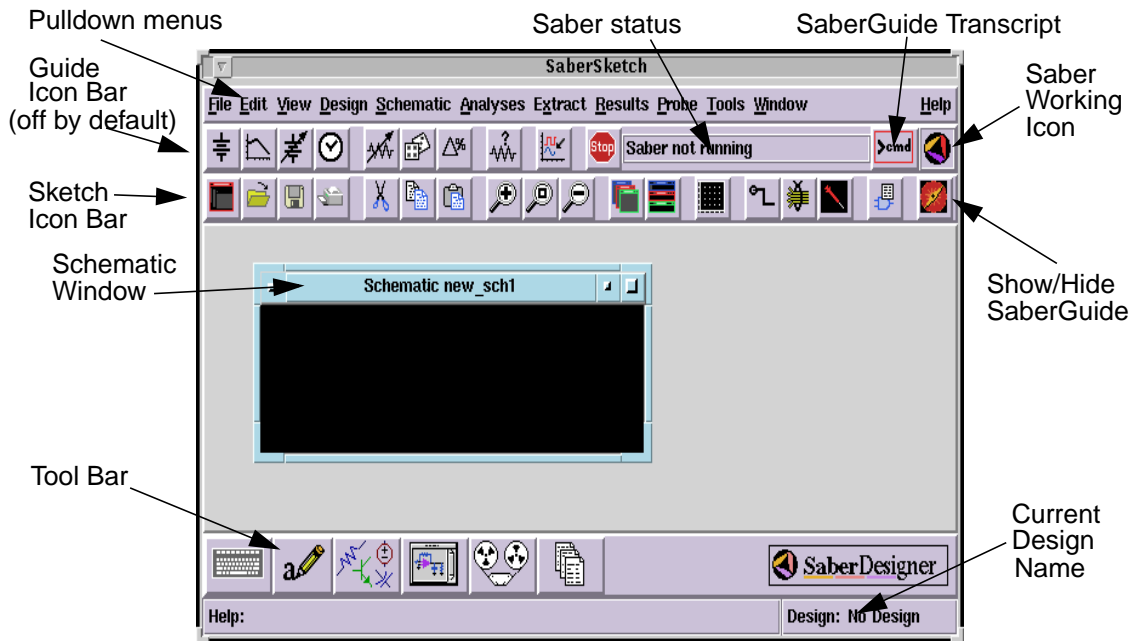
After you capture the circuit, you can simulate it with the Saber simulator from within the Sketch user interface. All Guide functionality is available within Sketch. Guide is the user interface to the Saber simulator. Because the simulator can not directly read the design as stored in the Sketch format, Sketch also features intelligent netlisting which determines when a new netlist needs to be generated and then automatically creates it.

Invoking Sketch

To invoke Sketch, use one of the following options depending on your operating system:

- UNIX: Type `sketch` in a UNIX window.
- Windows NT: Double-click on the **Sketch** icon in the SaberDesigner program group

The following figure shows the major user interface items of the Sketch tool:



To exit from Sketch application, select the **File > Exit** menu item.

Opening a Schematic Window - Sketch

After you invoke Sketch, you are ready to capture your circuit. To open a schematic window in Sketch, perform one of the following choices, depending on whether you are creating a new schematic or opening an existing Sketch schematic:

To create a new design:

- Select the **File > New > Design** pull-down menu.

This menu item opens an empty Schematic window. You are now ready to add symbol instances to the schematic, as described in the topic titled *Using the Parts Gallery to Place a Supplied Part - Sketch*.

To open an existing design, use the following steps:

- ❑ Display the Open Design dialog box (**File > Open > Design**).
- ❑ Navigate to the design and select the appropriate entry in the list.
- ❑ Click on the **OK** button.

Sketch opens a schematic window and displays the schematic.

After you open the design, you are now ready to add symbol instances to the schematic, as described in the topic titled Using the Parts Gallery.

Choosing Models - Sketch

When creating a design for simulation, you must choose what kind of models to use as your parts. The Saber simulator handles MAST components and templates.

MAST Components

The characterized MAST components in the online topic titled Component Libraries of Characterized Parts have been designed to perform like specific commercially available parts and are often named with commercial part numbers.

MAST Templates

The MAST templates identified in the online topic titled MAST Template Library of Generic Models allow you flexibility in customizing a part's behavior.

You control a template by defining parameter values used in the template's underlying mathematical equations. These values correspond to a part's device values, descriptive characteristics, operating conditions, and sometimes the equation variables themselves. The topic titled Applying Values to Characterize a MAST Template in Sketch summarizes the process of determining these values.

Methods for creating your own models:

- In the *Guide to Writing MAST Templates*, Book 1, the topic titled Introduction to Modeling with MAST provides an overview of how to write templates using the MAST language.
- The online help system topic titled Graphical Modeling describes how to use graphical models in a design.

Chapter 1: *Capturing the Design with Sketch*

- nspitos is a Spice-to-MAST Translator tool for converting SPICE models into MAST templates. Refer to the online help system topic titled “nspitos.”
- The Table Look-up Tool allows you to create MAST templates from a set of device input and output requirements. Refer to the online help system topic titled “Table Look-up Tool.”
- Analog Model Synthesis is a top-down design tool for creating the functional blocks of a design before beginning the more detailed design procedure within each block. Refer to the online help system topic titled “Analog Model Synthesis.”

Choosing and Placing Symbols on a Sketch Sheet

The following topics provide information on adding and modifying supplied symbols to a Sketch schematic sheet:

- Placing Symbols on a Sketch Schematic
- Using the Parts Gallery to Place a Supplied Part - Sketch
- Using the Dialog Navigator to Place a Symbol - Sketch
- Moving Symbol Instances and Specifying the Instance Name - Sketch
- Using Shared Symbols - Sketch
- Using Split Symbols - Sketch
- Adding MAST Power and Simulation Stimulus Parts - Sketch
- Including MAST Digital Parts in Designs - Sketch
- Adding MAST Hypermodels - Sketch
- Connecting a Part to a Node of the Proper Connection Type - Sketch

Placing Symbols on a Sketch Schematic - Overview

After you open the schematic window, you can place symbols on the schematic. A symbol is a graphical representation of the part. In SaberDesigner the functionality of the part is described by either an underlying hierarchical schematic or a MAST template. At the lowest level of hierarchy, all parts in the schematic must have an associated MAST template in order to be simulated by the Saber simulator.

When you place a symbol on a schematic, it is called an instance symbol. An instance symbol is a copy of the original symbol that can be modified in the schematic. Basically, the original symbol provides default values that can be overridden on the instance symbol. For example, you can place an instance symbol of a resistor in the schematic and modify the `rnom` property to specify the nominal resistance value on the resistor instance symbol. The default value of `rnom` on the original symbol did not change. The instance symbol maintains the theoretical link to the original symbol so that any changes made to the original symbol will be seen in the instance symbol unless the change was previously overwritten in the instance symbol. If you change the graphic representation (the symbol body) or a default property value on the original symbol, Sketch updates all instance symbols in the currently open schematic, after you save the edited symbol. Sketch checks and updates (if necessary) each instance symbol whenever you open a schematic.

Using the Parts Gallery to Place a Supplied Part - Sketch

The Parts Gallery is an interactive tool that allows you to search and place supplied and custom parts on the schematic. You can search various parts categories based on a part description, the symbol name or the MAST template name.

Once you locate a part in the Parts Gallery, you can place the symbol in the schematic, view the un-encrypted sections of a MAST template, or view the Template Description in the SaberDesigner online help system.

To use the Parts Gallery to place a symbol, display the Parts Gallery tool (**Tools > Parts Gallery**) and use either of the following methods to locate parts:

- If you want to locate a specific manufacturer's MAST component, use the Parametric Search Wizard in the Parts Gallery by choosing the **Tools > Parametric Search** menu item.
- One way to search for a part in the Parts Gallery is to navigate the Model Tree using the Available Categories listbox

The MAST Parts Library is categorized using an inverted tree structure similar to an organizational chart. You can navigate down the hierarchy until you find the category you want and the corresponding list of components.

- Another way to search for a part is by using the search capability within the Parts Gallery. You can search the entire Parts Library for a part description, template name or symbol name by following these steps:
 - a. You provide a string of characters in the Parts Gallery Search String field.

Chapter 1: Capturing the Design with Sketch

- b. You must specify how the search will use the provided string of characters by doing the following:
- c. Choose the **Options > Preferences** menu item. The Parts Gallery Preferences form appears.
- d. Click the Search tab and select the appropriate choices as follows:

Search part by:	Search match:
Part Name	Containing
Symbol Name	Beginning with
Model Name	Equal to
Any Field	
Ignore case when doing search	

For example, a generic transistor in the library has a template name of **q_3p**, a symbol name of **npn** and has a part name of BJT, NPN 3 pin.

MAST parts without underlying MAST templates include connector symbols, schematic borders, and the Saber Include File symbol.

- e. When you have finished selecting the search parameters, click the **OK** button in the Gallery Preferences form.
- f. Once you have entered the string of characters you are going to use for the search, either click on the **Search** button or press the Return key.

The Parts Gallery performs the search starting at the top level of the model tree.

- g. Refine the search control described in the previous steps until you locate the correct part.

Once you locate the part using one of the previous methods, the Parts Gallery allows you to do the following tasks with the part:

- Place the symbol on the schematic by clicking on the **Place** button. This action places an instance symbol in the middle of the schematic. You can move the instance symbol by following the procedure in the topic titled Moving Symbol Instances and Specifying the Instance Name - Sketch.

- View the MAST template of the selected part by choosing the **Tools > View Template** menu choice. This action displays the source MAST template.
- View the template description by choosing the **Tools > Help on Part** menu choice. The template description is displayed in the online documentation.

Using the Dialog Navigator to Place a Symbol - Sketch

You perform this procedure if you know the location of the Sketch symbol that you want to place on the schematic. The Sketch symbol must be either in the current directory (containing the design) or in a directory defined by the `AI_SCH_PATH` environment variable.

If the symbol pathname does not meet this requirement, you will not be able to place it in a schematic. To use the symbol, you must either move or copy the symbol to your current directory (containing the design) or exit Sketch and modify your `AI_SCH_PATH` environment variable.

- Display the Add Instance dialog box either from the menu **Schematic > Get Part > By Symbol Name...** menu choice or from the **Schematic Editor** popup menu (**Get Part > By Symbol Name...**).
- Enter the name of the symbol in the Symbol field.

If you do not know the pathname to the symbol, you can click the **Browse** button and use the Select Symbol dialog box to locate the symbol.

- Click the **Place** button to place the symbol on the schematic.

This action places an instance symbol in the middle of the schematic. You can move the instance symbol by following the procedure in the next subsection.

Moving Symbol Instances and Specifying the Instance Name - Sketch

You can move an instance symbol to the desired location in the schematic as follows:

- Place the mouse cursor over the instance symbol

The color of the instance will change to the highlight color (by default, the highlight color is red)

- ❑ Press and hold the left mouse button
- ❑ Move the mouse cursor to move the instance symbol.
- ❑ Place the symbol at the new location by releasing the left mouse button.

When you place an instance, Sketch automatically sets the reference designator (`ref`) property on the instance to a unique value. By default, Sketch sets the `ref` property value to *primitive_property_value#* (e.g. for a resistor using the `r` MAST template and having a primitive property value of `r`, the `ref` property could be `r3`). If the template name ends in a number, Sketch uses an underscore (`_`) between the template name and `ref` number (e.g. `q2n2222_12`).

The Saber simulator requires that the `ref` property value must be unique for each instance of an associated template. For example, because resistors and capacitors use different MAST templates, a resistor and capacitor instance in a schematic could both have a `ref` property value of `load`, but two resistors can not both have a `ref` property value of `r1`. The resulting netlist in this example would use instance names of `r.load` and `c.load`.

You can reset all `ref` property values using the **Schematic > Re-Reference** menu item. This command overrides all `ref` property values in the schematic, except for connector symbols, by re-incrementing the numbering scheme used in the `ref` property values for each instance of an associated template.

After you place the parts on the schematic, you can modify the properties on the instance, as described in the topic titled Annotating Properties - Sketch.

Using Shared Symbols - Sketch

Shared symbols consist of multiple graphical representations of an instance symbol that represent one discreet part in a schematic.

For example, four symbols for a fuse placed in different locations in a schematic to facilitate wiring represent a single fuse in the design with multiple wires attached to it.

The shared symbols can be placed on separate sheets of a design or on the same sheet. Shared symbols are wired just like normal symbols. The properties of shared symbols will be the same since they represent a single part.

To create shared symbols:

- Select a part from the Parts Gallery or create a part in the Symbol Editor window.
- Place the part as many times as you like in the design and wire them up.

Remember that the parts must all use the same symbol.

- To create the first shared symbol, highlight a symbol, then in the **Symbol** popup menu select the **Shared Symbol > Create** menu item. This will open the Shared Symbol Selection dialog box.
- The Shared Symbol Selection dialog box displays all of the parts that use the same symbol. Select the symbol with the Ref name you want. Click **OK**.

The two symbols will have the same Ref name in the Property Editor and now represent the same part.

- To add subsequent symbols to the shared symbol, highlight the symbol you wish to add to the shared symbol list, then in the **Symbol** popup menu select **Shared Symbol > Add**. Click **OK** in the Shared Symbol Selection dialog box.
- To view all of the available shared symbols in the design, highlight a shared symbol, then in the **Symbol** popup menu select **Shared Symbol > List**. A list of all of every other shared symbol will be displayed.

Using Split Symbols - Sketch

Split symbols are shared symbols which have had the ports, or other parts of the symbol, made invisible and assigned a different Symbol View. This allows you to visually differentiate shared symbols on a schematic.

To create split symbols:

- If you have not already done so, create a set of shared symbols.
- Select one of the shared symbols and open the Symbol Editor window by selecting the **Symbol > Symbol Editor** popup menu item.
- Create a new symbol view by selecting the **Symbol > Symbol View > Create** tab.
 - Type in a unique name for the view in the View Name field.

Chapter 1: Capturing the Design with Sketch

- Click on the **Split Symbol and Import Graphics From First View** buttons. Click the **Apply** button.
- Click on the **Modify** tab, select the new symbol view and click on the **Close** button.
- To make ports invisible, select the **Symbol > Port Visibility** menu item. In the **Select Port Visibility** dialog box move ports from the **Visible Ports** list to the **Invisible Ports** list by double clicking on a port name, or by selecting a port name and clicking on the double arrow (**>>**) button. Click on the **Done** button to finish.

You can make ports visible again by reversing the process.

- If you like, edit the symbol to change its appearance.
- Save the new symbol and the new view by selecting the **File > Save** pulldown menu item. Any modified symbol must retain the same name as the original symbol, and it must be available in a directory specified by the environment variable, `AI_SCH_PATH`.
- You can now close the **Symbol Editor** window.

To apply split symbols to a schematic:

- Highlight the symbol you wish to apply the new view to.
- Select the **Symbol > Symbol View** popup menu item.
- Select the view which is associated with the edited graphic you wish to display.
- Click on **Close**.

Adding MAST Power and Simulation Stimulus Parts - Sketch

Most designs require both power and simulation stimuli in order to function. The following list describes these types of MAST parts:

- **Power.** If you use a global net (e.g. vcc or vdd) to connect power to parts in the design, you must attach a source to one instance of the global net. If you do not attach the source, the global net will be floating during simulation.
- **Ground.** You must include a “Saber node 0” component in your schematic. If you do not include this Saber ground, then your Saber simulation results may not be correct. You can use the **Parts Gallery** to search for the parts containing **ground** in their description to find the **Ground, (Saber Node 0)** part.

- **Simulation Stimulus.** These parts (e.g. sine wave voltage sources or control system sources) allow you to stimulate the design during a simulation in Saber.

If you plan on using the schematic within hierarchy or in multiple designs, you may want to create a symbol for the schematic and then create a separate 'test' design which includes the symbol of the schematic and the simulation stimulus source(s).

For example, if you had a schematic called "filter" that is used in a larger system design and you wanted to simulate the "filter" design separately, you could create a symbol for the filter schematic, and instantiate the symbol in both the system design and in a "filter_test" schematic. This way you would maintain one copy of the "filter" schematic, yet still use it in multiple designs.

Including MAST Digital Parts in Designs - Sketch

To include digital parts in your design for a Saber mixed-signal simulation, perform the following steps,

- Place the generic digital part on the schematic.
From the MAST Parts Library, you can place generic digital parts in your schematic. For example, if you want to use a 74LS10 in your design, you should place a 2-input NAND gate on your schematic.
- You have the option to specify propagation delays and inertial delay.
 - Using the pre-defined t_{plh} and t_{phl} properties on the digital gate, you can specify propagation delays through the part.
 - You can specify inertial delay (smallest pulse to travel through the gate) using the t_{ilh} and t_{ihl} properties.

By default, these four properties are undefined.

- Determine the type of Hypermodel to use.
Your options for specifying Hypermodels are listed in the topic titled *Adding MAST Hypermodels - Sketch*.

Adding MAST Hypermodels - Sketch

If your design contains both analog and digital parts and you want to run a Saber native mixed-signal simulation, Saber must map signal values between them using Hypermodels.

There are several ways you can use MAST Hypermodels:

Chapter 1: *Capturing the Design with Sketch*

- Let Saber automatically specify a Hypermodel for you.

If you do not specify a Hypermodel in the Hypermodel field of the netlisting options form, the netlister automatically inserts a default Hypermodel.

- Select an ideal Hypermodel from the Saber Hypermodel library.

Ideal Hypermodels provide quick, approximate simulations and are available for several logic families.

- Select a technology-specific Hypermodel from the Saber Hypermodel library.

These Hypermodels provide greater accuracy at the cost of slower simulation times.

- Create a part number-specific Hypermodel as a way to include a particular digital part number in the design.

This method requires you to search the Hypermodel text files for the specific part name you would like to include in your design.

- Create and use your own Hypermodel.

If existing Hypermodels are not adequate for simulating your circuit, you can create your own Hypermodels using the MAST language. For information on making custom Hypermodels, refer to the topics Modeling Mixed Analog-Digital Systems with MAST in the *Guide to Writing MAST Templates* manual and Overview of Hypermodel Analog/Digital Interface Templates in the online help system.

Connecting a Part to a Node of the Proper Connection Type - Sketch

If your design contains parts of more than one kind of technology (e.g., both electrical and mechanical), you need to consider connection types when connecting templates of different technologies together.

Each connection point on a template has a type, and the node in the design that the connection point is attached to must be of the same type.

If you want to connect nodes of different connection types, you must do so through an interface template.

- **Connection Point Types** The online help system topic Template Description Section: Connection Points provides a summary of connection point types recognized by Saber.
- Detailed descriptions of each template's connection points, including type, function, and location on the symbol, are located in the

Connection Points section of its template description in the online help system.

Annotating Properties - Sketch

A property is an informational tag that describes a design characteristic of an object in the schematic such as the nominal resistance of a resistor or the maximum power ratings of a transistor. You can attach properties to symbol instances, ports, or symbol bodies. Properties allow you to customize models to meet the specific needs of your design.

The following topics describe the elements of a property, how to modify a property, how to obtain a property description, required properties on supplied parts, how to specify properties globally throughout the design, and how to specify parameters as property values:

- Elements of a Property
- Modifying Properties - Sketch
- Obtaining Help on a Property
- Required Property Changes on Supplied Parts - Sketch
- Applying Values to Characterize a Template in Sketch
- Specifying MAST Property Values Globally
- Defining and Passing MAST Parameters
- Creating Composite MAST Symbol Properties

Elements of a Property - Sketch

The following text describes how Sketch uses properties and how to modify properties.

A property consists of the following elements:

- Name defines the name of the property. In most cases, property names on supplied parts map directly to an argument in the MAST template.

The `model` MAST argument listed in the template description documentation is implemented using the `saber_model` property on the symbol. The netlister uses the `primitive` property value to map the symbol to its associated MAST template.

The netlister uses the `ref` property value as the instance name in the Saber netlist. If you do not specify this property, the netlister uses the Sketch-generated name (e.g. `r1`). For more information on this property, refer to the topic titled *Moving Symbol Instances and Specifying the Instance Name - Sketch*.

- Value defines the value of the property.
- Attribute defines the position, color, font, and visibility of the property in the symbol and schematic windows.
- Qualifiers allow you to group properties for use with other design tools.

Within Sketch, there are two different types of properties:

- Symbol Definition Properties are properties defined on the original symbol. The values and attributes defined on the symbol are the default values used on the symbol instance(s). You can only change these properties for the symbol in a symbol window within Sketch.

When you change a Symbol Definition Property value or attribute, the change is propagated throughout schematics that use the changed symbol.

- Symbol Instance Properties are properties on an instance in the schematic that override the default values or attributes defined on the symbol. This override occurs even if the property value on the symbol changes. The instance value only refers to the specific symbol on the schematic.

Modifying Properties - Sketch

To modify the property values on a symbol instance, use one of the following options:

- If the property is visible on the schematic, with the left mouse button, click on the property value and edit the property directly in the Schematic window.
- If the property is not visible on the schematic or if you want to modify more than the property value, you can use the Property Editor by following these steps:
 1. Highlight the symbol instance containing the property that you want to modify by placing the mouse cursor over the instance
 2. Display the Property Editor by double-clicking on the symbol.

The Property Editor requires a template information file

(*template_name.ai_tdb*) for the template represented by the symbol. If such a file does not exist, The Template Information System will generate one. For a complete discussion of the Template Information System, see the *Managing Symbols and Models* manual.

3. Edit the property by modifying the values in the Name and Value fields.

If you pass your mouse cursor over the Value field and the cursor becomes an arrow, the property is structured. Click on the Value field to display the property list, and edit the values as you would in the Property Editor.

By using the **Edit** and **Attributes** menu choices in the Property Editor, you can also add, delete, copy properties and change attributes.

If Saber is running on the design and the property change does not change the connectivity of the design (e.g. changing the `rnom` property on a resistor), Sketch sends an `alter` command to the simulator to change the property value of the in-memory netlist. This feature allows you to make some design changes without having Sketch re-netlist the design. If you issue an `alter` command in Saber, the property value is not changed on the schematic.

Obtaining Help on a Property - Sketch

Sketch provides the following ways to access help on property values defined on supplied symbols:

- To view a MAST template description in the online documentation, select **Help > Help on Part** from the Property Editor or the Parts Gallery pulldown menu item.
- To view the un-encrypted sections of a MAST template for supplied parts:
 - Select **Help > View Template** from the Property Editor pulldown menu item.
 - Highlight the symbol in the schematic and selecting **View Template** from the popup menu.

Required Property Changes on Supplied MAST Parts - Sketch

Property values defined on supplied symbols correspond to arguments in the underlying MAST template. If the MAST template argument has a default

Chapter 1: Capturing the Design with Sketch

value, then the corresponding symbol property uses the default argument value.

Required property values without defaults are designated with a value of **req**. Although most supplied parts will function using default property values, you must specify values for properties with a default value of **req** or for parts listed in the following table:

Part	Template Name	Required Property
resistor	r	resistance value (rnom)
capacitor	c	capacitance value (c)
inductor	l	Inductance (l)
BJT Transistor	q	type of transistor (saber_model): NPN (_n) or PNP (_p)

Applying Values to Characterize a Template in Sketch - Overview

The MAST template model behavior can be controlled by its *parameter values*. From a schematic, you pass parameter values to a template from the properties of a symbol instance.

Some templates/models take as their parameter values only symbol properties named for the parameters they describe.

For example, the MAST **v_dc** template has properties called `dc_value`, `ac_mag` and `ac_phase` (used for performing ac analysis), `white_noise` and `flicker_noise` (used for performing noise analysis). You select values for these parameters and enter them directly into the Property Editor window.

Most MAST templates take as their parameter values a combination of symbol properties named for the parameters they describe and a special symbol property, called either `saber_model` or `model`.

The `saber_model` or `model` property takes multiple values, called *model arguments*. In templates of relative complexity, such as for a bipolar junction transistor, the model arguments completely characterize the model's behavior.

Step 1. Characterize the template.

In order to make a template behave like the part you want to model, you must select values for the parameters. The process of selecting

these values is called *characterization*.

You can find detailed descriptions of the parameters for each model, along with their default values, in the template description in the online documentation. That information will let you characterize the template.

Step 2. Apply parameter values to the template symbol instance.

Once defined, you enter the parameter values as *symbol properties* on the instance of the template on your schematic.

The topic titled *Modifying Properties - Sketch* describes how to modify symbol property values on a symbol instance. Note that the model arguments of the `saber_model` and `model` symbol properties are structured.

Specifying MAST Property Values Globally - Sketch

You can globally specify MAST properties using the **saber** symbol (the part name is “Saber Include File”). Global properties used by supplied components are present on the **saber** symbol. Properties defined on an instance override the value defined by the **saber** symbol. You can also add custom global properties to the **saber** symbol using the following procedure:

1. Locate and place the **saber** symbol on the top-level schematic as follows:
 - a. Display the Parts Gallery (**Schematic > Get Part > Parts Gallery**).
 - b. Set the Search String to `saber`.
 - c. Select **Options > Preferences**. The Parts Gallery Preferences form appears.
 - d. Click the Search tab.
 - e. Set the Search part by: field to `Part Name` and the Search match: field to `Beginning with`. Click **OK**.
 - f. In the Parts Gallery form, click on the **Search** button.
 - g. Instantiate the part on the schematic by clicking the **Place** button.
2. Select the **saber** symbol by moving the mouse over the symbol and clicking and holding the right mouse button.

Chapter 1: Capturing the Design with Sketch

3. Display the Property Editor by selecting **Properties...** from the popup menu.
4. Add your custom global property by clicking on the `<New Property>` entry at the bottom of the Property Editor list and filling in the Name and Value fields.
5. When you have finished adding or modifying the properties, click **OK**.

Defining and Passing MAST Parameters - Sketch

A MAST parameter is a variable property value. You can use parameters to create generic parts consisting of:

- A top-level symbol used to define the parameter values
- An underlying schematic of a general topology with component values specified in terms of the parameters defined on the symbol.

If your schematic includes parameters, the netlister resolves the parameter values by searching the schematic that contains the instance of the existing template, and then the schematic containing that schematic, etc., until it finds the parameters it needs or it reaches the most general schematic. Parameters that are found are declared in the templates that correspond to the schematics through which they are passed. Parameters that are not found are declared `external` and can be defined in an included file. An included file is a file that is included in a netlist by using a `SaberInclude` property on a symbol.

If the property value is an identifier rather than a number or a string and it

- is not part of an expression, define the value of the identifier in the parameter value search path, described above.

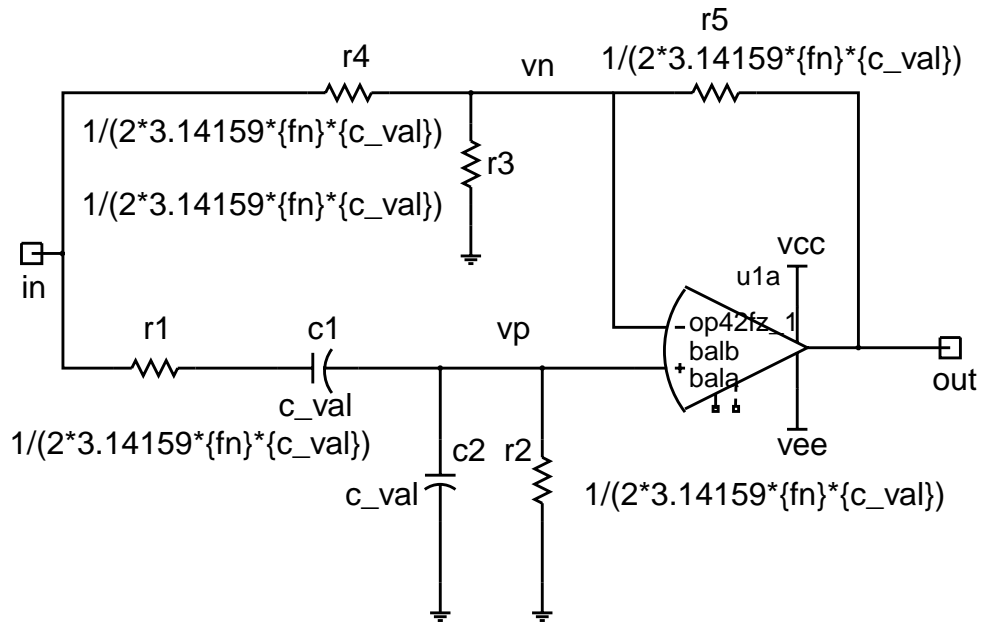
For example, in the schematic found in `Parameter Passing Example - Sketch`, the `c` property value of the `c2` capacitor is defined with the `c_val` identifier which is therefore interpreted to be a parameter. The value of this parameter is defined as a property on the `notch_filter` symbol in the top-level schematic.

- is part of an expression, enclose the identifier in curly braces. For an example, see how curly braces are used with `fn` in the `Parameter Passing Example`. The parameter value would need to be defined in the parameter value search path, described above.

Parameter Passing Example - Sketch

The following list shows an example for implementing a generic active filter block using parameters.

1. Create a schematic using the necessary parts, power sources, and hierarchical connectors to implement the topology of an active filter.

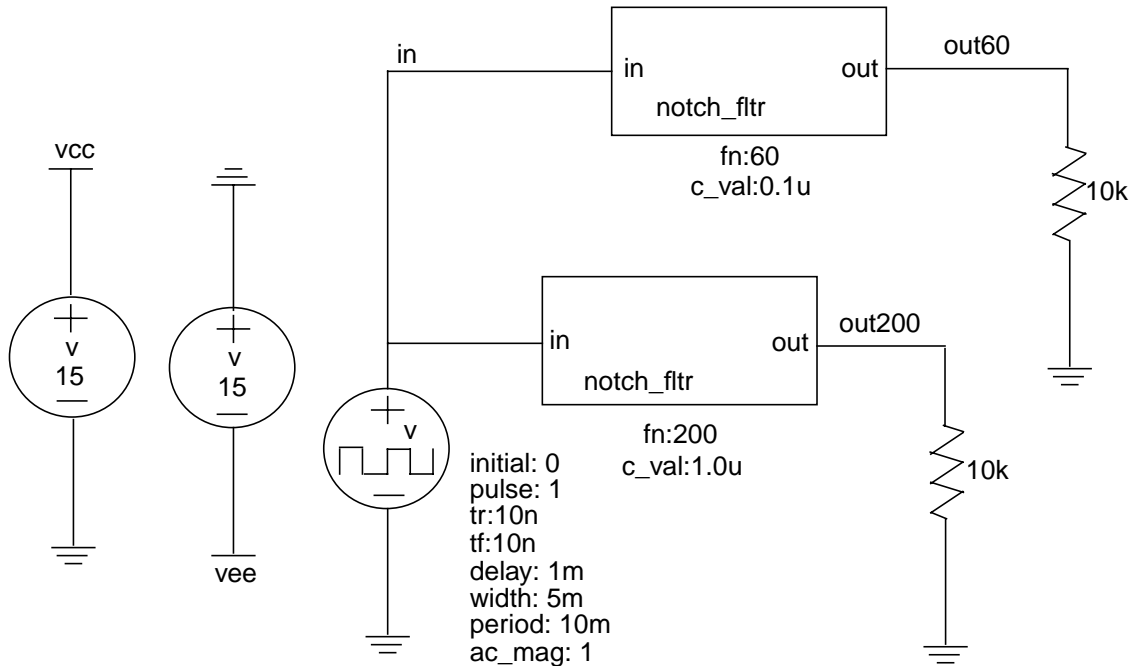


2. Modify the property values of the passive elements in terms of the parameters to be passed to the schematic. In the case of the active filter example, all component values are defined in terms of the break frequency (f_n) and a set capacitor value (c_{val}).
3. Create a symbol of the schematic by following the procedure in the topic titled Generating a Symbol for a Schematic Block - Sketch.
4. Create a symbol property for each parameter that should be passed to the underlying schematic by following the procedure in the topic titled Adding Properties to a Base Symbol - Sketch.

In this example, you would add two symbol properties for the f_n and c_{val} parameters (as shown on the **notch_filt** symbol in the following schematic).

5. On a higher level schematic, place the symbol and modify the f_n and c_{val} property values to define the parameters to pass to the underlying schematic by following the procedure in the topic titled

Modifying Properties - Sketch. The following figure shows a design that uses the generic active filter.



Creating Composite MAST Symbol Properties in Sketch

A *composite* property is a special property used for display purposes only. It takes for a value any combination of the other property values assigned to a symbol.

For example, a MAST resistor instance with the following properties

Name	Value
rnom	1k
tnom	27

could have a composite property that looks like

1k:27

on the screen.

1. Invoke the Property Editor on the item you want to add a composite property to.

2. Add the composite property name to the Name field. You can give the composite property any name you want.
3. Assign a composite value editor to the property (**Attributes > Value Editor > Composite**).
4. Assign a composite view to the property (**Attributes > View > Composite**).
The word `Undefined` will appear in the Value field.
5. Click on `Undefined` to invoke the Composite Property View Format Editor.
6. Use the Composite Property View Format Editor to create the composite property and to format the way it will appear displayed on the screen.

The editable text window in the dialog box allows you to create a combination of typed items and property values. In the example at the beginning of this topic

`1k:27`

the colon (`:`) was added from the keyboard. You can also type text.

- a. Select the properties you want to make up the composite by clicking the **Property** button and selecting properties one at a time from the list.
- b. Add separator characters or text descriptions from your keyboard.
The composite property will appear on the screen in the same order as its elements appear in the text window.
- c. Click **OK**.

The Composite Property View Format Editor closes.

7. In the Property Editor, click the visibility indicator next to the Value field so that the composite property displays on the drawing. If you want the value to appear without the name of the composite property, click the visibility indicator so that it is half-filled.
8. Click the **Apply** button on the Property Editor.
9. Position the composite property on the drawing by dragging and dropping it with the mouse cursor.

Wiring the Schematic - Sketch

Chapter 1: *Capturing the Design with Sketch*


After selecting and placing components on your schematic, you can add wiring to connect them. The following subtopics describe the wiring process:

- Drawing a Wire - Sketch
- Rewiring the Schematic - Sketch
- Naming a Wire - Sketch
- Using Other Methods to Connect Wires - Sketch
- Wiring the Schematic Using Busses - Sketch

Drawing a Wire - Sketch

After you place the parts on the schematic, you can connect the pins on the parts by drawing wires. In this topic you will learn how to draw and edit wiring in Sketch.

To draw a wire, perform the following steps:

1. Start the wire in one of the following ways:
 - Position the mouse pointer over an instance port and click the left mouse button. You can tell you are at a instance pin when your mouse cursor arrow icon changes to crosshair icon.
 - Press the **w** key on the keyboard then click at the starting location in the schematic.
 - Click on the **Wire** button -  - in the icon bar then click at the starting location in the schematic.
 - Select the **Create Wire** menu item in the Schematic pulldown or popup menu then click in the schematic.
 - Place the cursor over an existing wire and select the **Create Wire** menu item in the Wire popup menu.

The initial point of the wire becomes fixed and a highlighted wire “rubberbands” as you move your mouse.

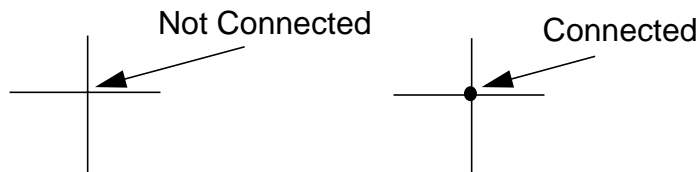
2. To create a vertex and change the direction of the wire, place the cursor at the desired location and click the left mouse button. Continue adding segments to the wire by moving the mouse pointer to the next position and clicking the left mouse button.

While you are creating a wire (while it still has a free end), a special “in progress” Wire popup menu is available. To open the menu, press and hold the right mouse button. The Wire popup menu contains the

following items:

- **Flip Previous Vertex.** For orthogonal wires, flips the previous two wire segments so that the vertex points 180 degrees away from the previous direction.
- **Delete Previous Vertex.** Removes the previous vertex from the wire. Succeeding deletes remove additional vertices. You can also use the Backspace key to do this.
- **Any-Angle Segment.** Changes the current segment to an any-angle segment. (The wire can run diagonally rather than along the alignment grid.) After the next vertex is established, the wire mode changes back to orthogonal. You can also create any-angle segments by pressing the shift key while creating wire segments. (You can make this the default by selecting **Any-Angle** in the Orientation field of the Schematic Preferences dialog box.)
- **Done.** Finishes the current wire at the location where the right mouse button was pressed.
- **Cancel.** Removes the wire under creation and cancels the wiring operation.

If two wires pass over each other in a schematic, they are not necessarily connected; you must specifically connect them. A connection between two wires is represented by a dot, as shown in the illustration:



3. To cancel the wiring operation and remove all segments, press the **Escape** key on your keyboard or select the **Cancel** item in the special **Wire** popup menu.
4. To terminate the wire do one of the following:
 - To connect the wire end to a port or another wire, place the cursor over the desired port or wire and click the left mouse button.
 - To terminate the wire end in an open area of the schematic, double-click the left mouse button while the cursor is in the free area of the schematic or select the **Done** item in the **Wire** popup menu

Rewiring the Schematic - Sketch

Listed below are procedures for some common schematic rewiring operations:

- To select a wire for editing, click on it with the left mouse button.
- To delete a wire, select it then press the **Delete** key. Alternatively, select the **Delete** item from the Wire popup menu, the Schematic popup menu or the Edit menu.
- To move the end of a wire, select the wire, place the cursor over the end point, click the left mouse button, and move the cursor away. Once away from the previous connection, the wire may be routed and connected as if it were a new wire.
- To move a wire or symbol, place the cursor over the object, press and hold the left mouse button, move the cursor to the intended location, and release the mouse button. Attached wire ends remain connected and the wires stretch to accommodate the move.

The Wire popup menu provides additional editing operations for wires. To access that menu, place the cursor over a wire that you want to edit and press the right mouse button.

Naming a Wire - Sketch

After you draw a wire, you may want to name it. If you do not name the wire, a Sketch-generated name is used (for example, _n183). If multiple wires are connected to the same node, you only need to name one of the wires. Sketch will apply the wire name to the other wires connected to the node.

To name a wire in the schematic:

1. Highlight the wire by moving the mouse cursor over the wire until the wire changes to the highlight color.
2. With the wire highlighted, press the right mouse button and select the **Attributes...** popup menu item.

This action displays the Wire Attributes form. In addition to naming wires, you can also use this form to change the “look” of the wire as displayed in the Schematic window. If you want to globally change the “look” of all wires in all schematics, you can edit the fields in the Wire tab of the Schematic Preferences form (**Edit > Schematic Preferences**)

3. Modify the wire name and click the **Apply** button.

Wire names should begin with an alphabetical character and contain

only alphanumerics. A wire name can not be a Saber command name or a MAST reserved word.

4. If the wire name is visible on your schematic you can also name it by placing your cursor in the wire name and directly editing the text.

Using Other Methods to Connect Wires - Sketch

In addition to drawing wires across the schematic, you can also use any of the following techniques to connect symbols in the schematic. Depending on your preference settings (**Edit > Schematic Preferences**), when you place your mouse cursor over a wire, every wire segment connected to that wire will be highlighted in the schematic. When using one of these alternative methods, you can use this technique to verify that wires are connected as you expected.

- Use wire naming. Even if the wires are not physically touching each other in the schematic, Sketch considers wires with the same name as being connected.
- Use onpage connector. You can also use the Same Page Connector (**sconn**) symbol in the **MAST Parts Library > Schematic Only > Connectors** category of the Parts Gallery. You can define the wire name by placing the **sconn** page connector on the schematic and editing the **name** property to contain the wire name.
- Use Busses. Unlike a bundle, which is just a collection of wires, busses are an ordered array of wires. Instead of drawing numerous wires, busses provide a convenient way to route a number of related wires through a schematic. For details on using busses refer to *Wiring the Schematic Using Busses - Sketch*.
- Use Bundles. Unlike a bus, which acts like an ordered array of wires, a bundle is just a collection of wires. Instead of drawing numerous wires, bundles provide a convenient way to route a number of wires through a schematic. You can have multiple bundles within the same schematic.
 - To draw a bundle, select the bundle icon from the Sketch icon bar and draw the bundle just as you would a wire.
 - To add or extract a wire to/from a bundle, simply connect the wire to the bundle. Sketch uses the wire names attached to the bundle to determine connectivity of wires connected to the bundle. Wire names can be directly edited in the schematic.
 - To determine the wires contained in the bundle, highlight the bundle and select the (**Wire Menu popup**) > **Attributes...** menu item.

Wiring the Schematic Using Busses - Sketch

After selecting and placing components on your schematic, you can add busses to connect them.

Busses are a collection of logical wires. A bus is used to route these collections of wires in a schematic. Busses can be connected to parts in three ways.

- You can rip out individual wires from a bus and connect them to individual pins on parts.
- You can directly connect a bus to a bus port of equal bus width on a part.


The following subtopics describe the bus wiring process:

- Drawing a Bus - Sketch
- Naming a Bus and Designating Bus Width - Sketch
- Creating a Bus Port - Sketch
- Creating a Bus Indicator - Sketch

Drawing a Bus - Sketch

After you place the parts on the schematic, you can connect the pins on the parts by drawing wires or busses. In this topic you will learn how to draw and edit busses in Sketch.

To draw a bus, perform the following steps:

1. Start the bus in one of the following ways:
 - Click on the **Bus** button -  - in the icon bar then click at the starting location in the schematic. You can tell you are at a instance pin when your mouse cursor arrow icon changes to the crosshair icon.
 - Select the **Create > Bus** menu item in the Schematic pulldown or popup menu then click in the schematic.
 - Press the **b** key on the keyboard then click at the starting location in the schematic.
2. To create a vertex and change the direction of the bus, place the cursor at the desired location and click the left mouse button. Continue adding

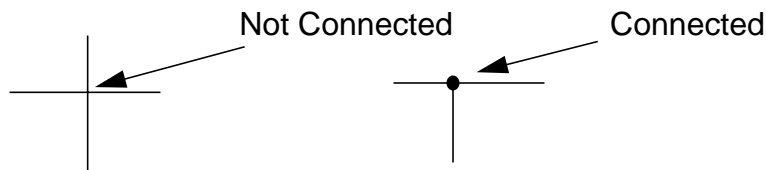
segments to the bus by moving the mouse pointer to the next position and clicking the left mouse button.

While you are creating a bus (while it still has a free end), a special “in progress” Bus popup menu is available. To open the menu, press and hold the right mouse button. The Bus popup menu contains the following items:

- **Flip Previous Vertex.** For orthogonal busses, flips the previous two bus segments so that the vertex points 180 degrees away from the previous direction.
- **Delete Previous Vertex.** Removes the previous vertex from the bus. Succeeding deletes remove additional vertices. You can also use the Backspace key to do this.
- **Any-Angle Segment.** Changes the current segment to an any-angle segment. (The bus can run diagonally rather than along the alignment grid.) After the next vertex is established, the wire mode changes back to orthogonal. You can also create any-angle segments by pressing the shift key while creating bus segments. (You can make this the default by selecting **Any-Angle** in the Orientation field of the Schematic Preferences dialog box.)
- **Done.** Finishes the current bus at the location where the right mouse button was pressed.
- **Cancel.** Removes the bus under creation and cancels the operation.

If two busses pass over each other in a schematic, they will not be connected. If you want to create a junction between two busses or a bus and a wire you must use the **Rip Bus** or **Rip Wire** menu item from the **Bus** popup menu.

A connection between two busses or a bus and a wire is represented by a dot, as shown in the illustration:



3. To cancel the bus wiring operation and remove all segments, press the **Escape** key on your keyboard or select the **Cancel** item in the special **Bus** popup menu.
4. To terminate the bus do one of the following:

- To terminate the bus end in an open area of the schematic, double-click the left mouse button while the cursor is in the free area of the schematic or select the **Done** item in the Bus popup menu
- To connect the bus end to a port, place the cursor over the desired port and click the left mouse button. The bus and the port must have the same bus width to make a successful connection. Refer to Naming a Bus and Designating Bus Width - Sketch for details on determining bus width.
- To connect a sub-bus from a bus select the **Rip Bus** menu item from the **Bus** popup menu. Type a name (optional) and the correct bus width in the Rip Bus dialog box and connect the bus to the appropriate port.
- To connect a wire from a bus select the **Rip Wire** menu item from the **Bus** popup menu. Select which wire you want to connect from the Wires in Bus dialog box and connect the wire to a port.

Naming a Bus and Designating Bus Width - Sketch

All busses have a name and bus width to identify them.

Bus Names

After you draw a bus, you may want to name it. If you do not name the wire, a Sketch-generated name is used (for example, `_b183<0:1>`).

Busses are connected by their names. A bus on one side of a schematic with the same name as a bus on the other side of the schematic are considered to be the same bus. Bus names are edited through the Attributes dialog box.

Bus Widths

You must designate a bus width. Bus width is the number of bits/lines/signals/wires/nodes in a bus or a bus port. There are several formats.

1. In the format *name<a: b>*, *name* is the name of the bus and *<a: b>* is the range of the bus width where *a* and *b* are integers separated by a colon. For example, an eight bit bus on a multiplexer could be named `mux_1<0:7>` or `mux_1<0:3,4:7>`.
2. In the format *name<a: b: c>*, *name* is the name of the bus and *a: b* is the range of the bus width where *a* and *b* are integers separated by a colon. The integer *c* is the increments inside the range *a: b*. For example, a four bit bus on a multiplexer could be named `mux_1<0:7:2>` since the bus increment is every two bits. An equivalent name would be `mux_1<0,2,4,6>`.

This naming format does **NOT** apply to bus ports. Bus ports do not

recognize the increment integer.

You can also mix the formats. For example, an eight bit bus on a multiplexer could be named `mux_1<0:3,4:11:2>`.

To name a bus and designate bus width in the schematic:

1. Highlight the bus by moving the mouse cursor over the bus until the bus changes to the highlight color.
2. With the bus highlighted, press the right mouse button and select the **Attributes...** popup menu item.

This action displays the Bus Attributes form. In addition to naming busses, you can also use this form to change the “look” of the bus as displayed in the Schematic window. If you want to globally change the “look” of all busses in all schematics, you can edit the fields in the Wire tab of the Schematic Preferences form (**Edit > Schematic Preferences**)

3. Modify the bus name and click the **Apply** button.

Wire names should begin with an alphabetical character and contain only alphanumeric characters. A wire name can not be a Saber command name, or a MAST reserved word.

Bus widths are always integers.

Creating a Bus Port - Sketch

In order to connect a bus to a part you must create a *bus port* on that part that matches the bus width of the bus you want to connect to.

The alternative is to rip wires out of the bus and connect them to each individual port on a part. This can be a time consuming process.

To Create a bus port:

1. Open the Symbol Editor window with an existing part, or create your own part.
2. Create a port. Use the **Symbol > Create >port_type** pulldown menu item, or use the popup Symbol Editor menu **Create >port_type** item.
3. Place the port appropriately on the part.
4. Right click on the port to open the popup Port menu. Select the **Attributes** menu item and edit the port name in the Name field so that it has a bus width compatible with the bus you will be connecting to the part. Refer to Naming a Bus and Designating Bus Width - Sketch for details on bus naming and bus width designation.

Chapter 1: *Capturing the Design with Sketch*

5. If you are modifying an existing part you will probably want to save the part with a different file name from the original. If you save the part using the same file name as the original the symbol will be changed for all parts using that symbol.
6. Close the Symbol Editor window.
7. Place the part into the design with the **Schematic > Get Part** menu item.

Creating a Bus Indicator - Sketch

You can create a bus indicator symbol in the Symbol Editor window that displays bus width when the symbol is placed over a bus.

To create a bus indicator symbol:

1. Open the Symbol Editor window by selecting the **File > New > Symbol** pulldown menu item.
2. Create a symbol graphic with the Draw tool. For details on using the draw tool refer to Draw tool in the online help system.
3. Open the Property Editor by selecting the **Symbol > Properties** pulldown menu item.
4. Create a new property with the Name field called `symboltype` and in the Value field called `bus_indicator` by clicking on the [New Property] field and typing in the appropriate text.
5. Save the symbol in a directory specified by the environment variable, `AI_SCH_PATH`.
6. Close the Symbol Editor window.

To use the bus indicator symbol:

1. Add the bus indicator symbol to your design with the **Schematic > Get Part > By Symbol Name** menu item.
2. Move the bus indicator symbol over a bus. The width of the bus will be displayed.

Generating a Symbol for a Schematic Block - Sketch

If you want to include your schematic block in a hierarchical design, you must create a symbol for it. To create a symbol for a hierarchical schematic, follow these steps:

1. Create the schematic.

2. Add hierarchical connectors on the schematic for each port that you want defined on the symbol. These connectors are located in the Parts Gallery in the Category Name: /MAST Parts Library/Schematic Only/Connectors.
 - If the connector is attached to a digital signal in the schematic (`logic_4` in the MAST template), you should use the Hierarchical Input, Output and Bidirectional connectors.
 - If the connector is not attached to a digital signal in the schematic, use the Hierarchical Analog connectors.

3. Create the symbol for the hierarchical schematic (**Schematic > Create > Hierarchical Symbol**)

Sketch opens a symbol window with the same name as the schematic and adds a port for each hierarchical connector in the schematic. Sketch uses the `name` property value on the hierarchical connector as the port name on the symbol. The symbol name is identical to the schematic except the file extension is `.ai_sym`.

4. Add the graphics for the symbol body using the AimDraw tool.
5. Create a symbol property for each parameter passed to the underlying schematic. Adding Properties to a Base Symbol - Sketch provides a procedure for doing this.
6. Save the symbol (**File > Save**).
7. Exit the Symbol Editor (**File > Close > Active**).

You are now ready to place this symbol on a schematic.

Adding Borders - Sketch

To select which schematic border will be displayed when the border is visible go to the **Edit > Schematic Preferences** menu item, and click on the Display tab. The Default Border Selection field will display the name of the border which will be displayed on the schematic. Pressing the **Select** button displays a list of saved borders.

Select the **View > Show Border** menu choice (or the **Toggle Border** icon) to make the border visible. You may need to click on the **Zoom to Fit** icon in order to make the border fit on your screen.

You can modify the look of the border using the Border Configure dialog box, invoked by selecting the **Edit > Border Configuration...** menu choice, and using the fields on the Outline and Rulers tabs.

The border annotation is treated as a part of the border, so that when the border is hidden from view, so is the border annotation.

Border Annotation in Sketch

Border annotation allows you to add important information such as title, revision history, and sheet numbering.

There are two types of annotations, and each one uses a different category of symbols:

- a *data block* is a single symbol. This symbol contains all necessary lines, and no additional lines can be added to it on a schematic. A data block symbol takes `data` as the value for the `annotation_subregion` property.
- a *header* symbol is used with one or more *line* symbols. You use the Border Annotation editor to stack instances of these symbols together on the schematic sheet. These symbols take `header` and `line`, respectively, as values for the `annotation_subregion` property.

The Border Annotation editor uses these symbols to create border annotation blocks on the Sketch schematic.

Step 1. Create border annotation symbols

You can draw new symbols using the Symbol Editor, giving them each the symbol property `symboltype` with a value of `border_annotation`. Each data region is also defined by a symbol property so that users can add data to the instances placed on the drawing by using the Property Editor. The online help system topic, *Border Annotation Drawing- Reserved Properties*, has a complete list of properties used with these symbols.

These newly created symbols then need to be placed in the `BorderAnnotationLibrary` directory in the design site, a step usually done by the design site manager.

Step 2. Build a border annotation block on your schematic

1. Select the **Edit > Border Configuration...** menu choice to invoke the Border Configure dialog box.
2. Select the tab according to the location where you want the border annotation block to be on your schematic:
 - Top Right

- Bottom Right
 - Next To B_R
3. Click in the Header/Data field to invoke the parts browser for viewing the contents of the `BorderAnnotationLibrary` directory.
 4. In the parts browser, select the border annotation symbol you want and click **OK** to place it in the Border Configure dialog box.
 5. Use the tab key to add a new line.
 6. Click **Apply** to place the symbol on the schematic.

Step 3. Fill in the data fields on the border annotation block

Because the data fields are not editable in place, you must access them with the Property Editor:

1. Highlight the border annotation block with the mouse cursor.
2. Invoke the Property Editor.

Each data field is represented by a property.

3. Edit the Value fields of the properties and apply them to the symbol.

Saving the Sketch Design

You can save a schematic by selecting the **File > Save** pulldown menu item. Sketch saves all schematics using an `.ai_sch` extension.

If you are saving a multi-sheet schematic, Sketch saves all sheets in the schematic. There is no method to save just one sheet in a multi-sheet schematic. Sketch saves all sheets in the schematic in a single `schematic_name.ai_sch` file.

If you are saving a schematic that is contained in an hierarchical schematic, Sketch only saves the active schematic.

Chapter 1: *Capturing the Design with Sketch*

Creating a Sketch Symbol

You can create and modify a Sketch symbol in the Symbol Editor of Sketch. A symbol is a graphical representation of a template or a schematic. A symbol has properties that describe characteristics of the template or schematic it represents. By creating symbols that represent a portion of circuitry and then using them in another schematic, you can create hierarchy.

For a Sketch symbol that describes the connections and characteristics of a lower level description, the lower level description must be a MAST template or a Sketch schematic.

You can create a symbol of a schematic by generating a symbol for a schematic, a process described in *Generating a Symbol for a Schematic Block*.

You create a symbol using the Symbol Editor and then assigning a schematic to it using a symbol property using the following procedures:

1. Opening a Symbol Editing Window
2. Drawing the Symbol Graphics
3. Specifying Symbol Views
4. Adding and Naming Symbol Ports
5. Associating the Symbol with a Lower-Level Description
6. Adding Symbol Properties
7. Creating Online Symbol Help
8. Saving a Symbol
9. Adding a Symbol to the Parts Gallery

Opening a Symbol Editing Window

After you invoke Sketch, you can create a symbol. As described in the following topics, you can create a symbol by:

- Creating a New Symbol - Sketch
- Opening an Existing Symbol - Sketch

Creating a New Symbol

To create a new symbol, select the **File > New > Symbol** pulldown menu.

This menu item opens a Symbol Window and displays the symbol. You are now ready to add symbol graphics.

Opening an Existing Symbol

To open an existing symbol, use the following steps:

1. Display the Open Symbol dialog box (**File > Open > Symbol**)
2. Navigate to the symbol using the Files and Directories listboxes.
3. Click on the **OK** button to execute the dialog box.

Sketch opens a symbol window and places the origin in the middle of the symbol window. The symbols use the convention of placing the origin in the center.

You can now edit the symbol to modify any of the characteristics as described in the remainder of this chapter.

Drawing the Symbol Graphics

Using the AimDraw tool, you can create graphics and comment text for the symbol. To display the AimDraw Tool, select the **Symbol > Create > Graphics...** pulldown menu item. Graphics and text added with the AimDraw tool are just for appearance and have no affect on the functionality depicted by the symbol or it's lower level descriptions (MAST template or schematic).

You can use any combination of graphical elements available in the AimDraw tool to create the symbol body. A symbol body can consist of as little as a single rectangle.

Specifying Symbol Views

A view is the graphical representation of the symbol. For example, a boolean AND gate symbol could have a separate view for the IEEE, default, and DeMorgan representations. Although you only define the ports and properties once, the symbol has three graphical representations. If you move a port or change a property, it affects all views of the symbol.

You can create, change, and delete a symbol view in the Symbol View dialog box (**Symbol > Symbol View...**).

Adding and Naming Symbol Ports

After you draw the symbol graphics, you can add ports to the symbol. A port is a connection point that maps to either a hierarchical connector on a underlying schematic or a connection in a MAST template.

To add a port to a symbol, perform the following steps:

- ❑ Determine the type of port that you need

Sketch provides four types of ports. The type of port that you use depends on the type of signal associated with the port in the lower level description.

- If the port is associated with a digital signal, you add an Input, Output or Bidirectional port depending on the flow in the lower level description.

When you place the symbol on a schematic, if a digital port is wired to an analog signal, these types of ports help the netlister determine how to insert the Hypermodel to map signal values between analog and digital representations.

- If the port is not associated with a digital signal, you add an Analog port.

Chapter 2: *Creating a Sketch Symbol*

- ❑ Place the ports in the symbol window by using the **Symbol > Create > *port_type*** pulldown menu.
- ❑ Edit the port name

You can edit the port name by moving the mouse cursor over the port name and clicking the left mouse button. This action should add an insertion cursor in the port name that allows you to use the keyboard to edit the port name. Alternatively, you can use the Port Attributes dialog box.

Port names can not match MAST reserved words, Saber commands, , or Saber arguments. If your port name uses one of these values, you can prepend `saber_` or some other unique identifier to the port name. For example, if your port name was `ac`, to avoid creating a mapping file, you could edit the port name to be `saber_ac` and Sketch will associate it to the lower level connection called `ac`.

- If the lower level description is a schematic, the port name on the symbol should match the `Name` property on a hierarchical connector in the schematic.
- If the lower level description is a MAST template, the port name on the symbol should match one of the connection point definitions in the MAST template.

If the port names do not follow these conventions, you will need to create a mapping file as described in the *Managing Symbols and Models* manual.

- ❑ Optionally, edit the port attributes (e.g. color, font, position, and visibility)

To display the Attributes dialog box, move the mouse cursor over the port and press the right mouse button. After you make the changes, click the **Apply** button to implement the changes on the port.

Associating the Symbol with a Lower-Level Description

Having drawn a symbol, you have one of three options for associating it with a lower-level description:

- Associating the Symbol with a Schematic - Sketch
- Creating a Schematic for the Symbol - in Sketch
- Associating the Symbol with a MAST Template - in Sketch

Because a symbol defines the connections and characteristics of a lower-level description, you must associate the symbol with either a schematic or a MAST template. If you do not follow the conventions presented in the previous topics, you must use mapping files, described in the *Managing Symbols and Models* manual.

Associating the Symbol with a Schematic

By default, Sketch assumes that the symbol name matches the associated schematic name and that the symbol and schematic are located in the same directory. If not, the pathname to the symbol and schematic must be in the `AI_SCH_PATH` environment variable. When Sketch traverses hierarchy, it first looks for symbols and schematics in the current directory and then in the directories specified by the `AI_SCH_PATH` environment variable.

If the symbol and schematic names follow this convention, you do not have to do anything more to associate the symbol with the schematic. If this is not the case, you can use one of the following methods:

- Define the name of the schematic by adding a `schematic` property to the symbol. This property specifies the name of the underlying schematic. By default, Sketch assumes that it is in the same directory as the symbol.

With this method you can use one symbol to represent one of several different schematics depending upon your application. For example, in one design the symbol could represent a schematic of the CMOS implementation of a circuit, in another design it could represent a schematic of the BJT implementation of the same functional circuit.

- Add a `primitive` property to the symbol with a value of `null`.

Creating a Schematic for the Symbol

If you do not yet have a schematic for the symbol, you can create one by selecting **Symbol > Create > Hierarchical Schematic** to open a Sketch schematic window.

A schematic window with the same name as the symbol you have just been working on opens, and the ports from the symbol are converted to hierarchical connectors in the schematic. You can now create a new schematic using the hierarchical connectors as inputs and outputs, allowing you to do top-down design.

Associating the Symbol with a MAST Template

In order to associate the symbol with a MAST template, you must add a `primitive` property on the symbol with the value being the name of the MAST template. To give instances of the symbol a reference designator, you must add a `ref` property on the symbol, leaving the Value field blank.

Provided that the template's default property values are specified in the header declaration, not in the body of the template, The Template Information System will automatically associate the template arguments with the symbol.

Adding Symbol Properties

The following list describes special considerations for using properties on symbols:

- `ref` property defines a unique instance name for each instance of a lower level description. When a symbol is placed on a schematic, if this property is present, Sketch automatically assigns it a unique name that can be overwritten by the user. For information on the `ref` property, refer to the topic titled *Moving Symbol Instances and Specifying the Instance Name*.
- If the schematic contains parameters, you can define the parameter value to be used in the underlying schematic by adding a property with the same name as the parameter on the symbol body.

If you do not resolve the parameter value using this method, you must add a property to define the parameter either on a symbol further up in the hierarchy or in a `SaberInclude` file on the top-level schematic.

Adding Symbol Properties on Instances

To add a property to a particular symbol instance on a schematic, follow these steps:

- Invoke the Property Editor on the symbol instance as follows:
 - Move the cursor over the desired symbol.
 - Click-and-hold the right mouse button to bring up the Symbol menu.
 - Choose the **Properties...** menu choice.

- Specify the property Name and default Value.
- Click the **Apply** button.

Adding Properties to a Base Symbol

To add a property to a base symbol, do the following:

- Invoke the Symbol Editor. One way to invoke the editor is to move the cursor over an existing symbol on a schematic and choose the (**Symbol Menu**) > **Symbol Editor** menu choice.
- In the Symbol Editor window, choose the popup menu item (**Symbol Editor**) > **Properties...** A Property Editor dialog box appears.
- Specify the property name and default value.

If the property value should not be changed on the symbol, you can lock the value by selecting **Attributes > Protection > On**.

- Optionally, create a Help Description for this property.
- Click the **Apply** button on the Property Editor.

Creating Online Symbol Help

You can create your own online help messages with the Property Editor **Attributes > Help Description** pulldown menu item. The **Attributes > Help Description** pulldown menu item is only available in the Symbol Editor window.

Help Description allows you to create online help with your own Property Help Messages dialog box for a template when you select the **Help > Help on Part** menu item.

To add your own help description perform the following steps:

- Place the mouse cursor in a property Name or Value field.
- Click on the **Attributes > Help Description** menu item. The Property Help Description dialog box will be displayed.
- Delete any text you don't want displayed and type in your help description.
- Click on the **OK** button. This closes the Property Help Description dialog box.

Chapter 2: *Creating a Sketch Symbol*

- ❑ Click on the **Apply** button in the Property Editor.

A .hlp file will be created and saved in the directory you are currently using. This .hlp file will only apply to symbols with the same name that originate in this directory.

If you want to apply a .hlp file to all symbols with the same name copy the .hlp file to any directory in the AI_SCH_PATH environment variable path.

SaberDesigner first looks for .hlp files in your current directory. If no .hlp file exists for the symbol SaberDesigner looks along AI_SCH_PATH.

Saving a Symbol

After you finish creating the symbol and adding the necessary properties, you can save the symbol using the **File > Save** pulldown menu. This action saves your symbol with a .ai_sym extension. After you save the symbol, you can place it in a schematic.

Adding a Symbol to the Parts Gallery

New symbols that you have created in the Symbol window can be placed into a schematic using the **Schematic > Get Part > by Symbol Name** menu item. You can also place the symbol into the Parts Gallery database.

You can place a symbol into the Parts Gallery database by opening the Parts Gallery and using the **Edit > New Part** menu item. The symbol is placed in a user database file named .aimpart_user.

Simulating Sketch Designs

After you capture the circuit in Sketch, the next step in the design process is to simulate the design.

The following topics describes the process of simulating Sketch designs:

1. Specifying the Top-level Sketch Schematic
2. Specifying Netlister and Simulator Invocation Options - Sketch
3. Simulating a Sketch Design
4. Viewing Sketch Design Analysis Waveforms
5. Making Changes to a Sketch Design
6. Exiting Saber - Sketch

Specifying the Top-level Sketch Schematic

In order for the simulator to simulate your design, you must inform Sketch which schematic is at the top level of your design. Because the simulator can only have one netlist open at a time, you can only specify one top-level schematic in a Sketch session. If the schematic contains no hierarchy and it is the only open schematic, Sketch assumes that it should be used at the top-level schematic and you can skip this step. Otherwise, you can specify the top-level schematic name using the **Design > Use > *design*** menu bar item in Sketch.

When you specify the top-level schematic, Sketch displays the design name in the lower right corner of the User Interface and creates a design file (*design.ai_dsn*) that contains additional information for simulation and hierarchy management. If the schematic contains hierarchy, Sketch also opens the Design Tool. You can use the Design Tool to open, save, and close schematics within the hierarchy and to navigate through hierarchy.

Although you can only specify one top-level schematic, you can still open, view and edit schematics that are outside of the hierarchy.

Specifying Netlister and Simulator Invocation Options - Sketch

Because the simulator can not read the schematic directly, the simulator simulates an intermediate netlist generated by a supplied netlister. The resultant netlist is an ASCII file containing the instance names, connection points and all non-default part parameters.

Sketch automatically netlists the design whenever you attempt to simulate the design when the connectivity in the netlist differs from that in the design.

For example, if you add or re-name a wire, the next time you attempt to run an analysis, Sketch automatically re-netlists the design and re-loads it into the simulator prior to running the analysis. If you change the color of the wire and attempt to run an analysis, the simulator will use the existing netlist because the connectivity of the design did not change.

If you change a property value on symbols whose underlying model targets the Saber simulator, Sketch sends an `alter` command to the simulator to change the in-memory netlist, thus eliminating the need to re-netlist. Because the netlist is not created and the simulator is not invoked until you run the first analysis, you should verify the netlister and simulator invocation options prior to running an analysis in Guide.

1. Verify the Netlister Options in Guide (**Edit > Simulator/Netlister Settings...**)

The netlister uses these options to create a file called a netlist that completely characterizes the circuit. These options include:

- Hypermodel files map signals between analog (continuous time) and digital signal representations. If you do not specify a Hypermodel file, the netlister uses a “default ideal” Hypermodel which is characterized to act similar to a CMOS technology. The output of the default ideal Hypermodel acts like an ideal voltage source with voltage references defined by the Power net name and Ground net name fields in the Netlister > Basic tab. Selecting Hypermodels from the Saber/Netlister Settings Form provides a procedure for selecting Hypermodels.
- Mapping Files map symbols to their appropriate MAST template. MAST is the modeling language used by all models (including netlists) that are read by the Saber simulator.

Mapping files for supplied parts are automatically loaded by the netlister. Supplied mapping files must reside in the directory *install_home/bin*. Mapping files specified in the Saber/Netlister Settings form (Map Files tab) must reside in a directory that is referenced by the `SABER_DATA_PATH` environment variable. For information on why/how to create mapping files, refer to the *Managing Symbols and Models* manual.

Simulator/Netlister Settings Form has detailed information on the fields in this form.

2. Verify the Saber Invocation Options (**Edit > Guide Preferences**)

The netlister reads the circuit interconnections from a schematic and the component descriptions from various libraries. The Saber tools use the information from the netlist file (*.sin) to perform their analyses.

Simulating a Sketch Design with Saber

After you invoke the SaberGuide User Interface and verify the netlister and Saber invocation options, you are ready to simulate the design using the Saber simulator.

Typically, you will want to first verify your design functionality. Once you are certain your circuit works under nominal conditions, you can then tune your design parameters to reduce your design cost and increase circuit reliability.

Because Saber contains numerous types of analyses, you can select the appropriate analysis depending on the type of specification that you are trying to verify. Prior to running the analysis, Sketch determines whether a new netlist needs to be generated and does so if necessary.

If you attempt to run an analysis on a schematic without specifying the top-level schematic, you will be prompted for the top-level schematic name (unless there is only one schematic open in the SaberDesigner work surface) or to cancel the analysis run.

You can access these analyses using the **Analyses** pulldown menu in SaberGuide.

Verifying Design Functionality

The Saber and VeriasHDL simulators provides the following analyses to verify the functionality of your design in the time and frequency domains. Procedures for executing these analyses, viewing the resulting waveforms, and analyzing the results are described in chapters 4-6 of this manual.

- To verify the time-domain specifications of your design, use the transient analysis to determine the system response over time. The Saber simulator provides the Fourier and FFT analyses to transform time-domain waveforms in to the frequency spectrum. These analyses are described in Chapter 4.
- To verify the frequency domain specifications of your design, use the AC analysis to determine the small-signal frequency response of the system. The Saber simulator provides the iFFT analysis to transform frequency domain waveforms into the time domain. These analyses are described in Chapter 5.
- Use the DC Transfer analysis to sweep an independent source and calculates the operating point at each swept value. These analyses are described in Chapter 6.

Tuning Design Parameters

The Saber simulator provides the following analyses to tune the design parameters, such as part values and tolerances, of your design. Procedures for executing these analyses, viewing the resulting waveforms, and analyzing the results are described in chapters 7-10 of this manual.

- Vary allows you to sweep design/part parameters (at user-defined values) and execute a set of analyses at each parameter value.
- Monte Carlo allows you to randomly vary design/part parameters and run various analyses to evaluate a simulated manufacturing run.
- Sensitivity analysis determines the sensitivity of a performance measurement to a variance of a design/part parameter.
- Stress analysis determines whether a part is overstressed during a particular DC, DC transfer or Transient analysis run.

Examining the Simulator Transcript

You can view the simulator invocation messages by clicking on the Simulator Transcript icon (>cmd) in the top right corner of the SaberDesigner user interface. Saber also stores this information in the *design.out* file, located in

the same directory as the netlist. For information on common error/warning messages produced by Saber, refer to *SaberDesigner Errors*.

Viewing Sketch Design Analysis Waveforms

SaberDesigner provides two methods of viewing analysis waveforms, the Scope Waveform Analyzer and DesignProbes within a Sketch design. All waveforms created by the various analyses can be shown using either viewing method. The process of viewing the analysis waveforms in Scope is incorporated with the analysis procedures in Chapters 4 - 10 of this manual.

The following topics describe how to add nodes to the Signal List, how to view signals internal to a template, and how to use DesignProbes within a Sketch design:

- Specifying Sketch Nodes or Pins to Create Waveforms
- Viewing Signals Internal to a Template - Sketch
- Adding DesignProbes to a Sketch Wire or Pin

After you view the waveforms and analyze the results, you can either run another analysis or make design changes as described in the next major section.

Specifying Sketch Nodes or Pins to Create Waveforms

The Saber simulator uses a Signal List to determine which signals are added to a Plot File. You can view the waveforms stored in the Plot File using either Scope or DesignProbes within the design. By default, the simulator creates waveforms for all nodes on the root of the design. You can create or append nodes or pins to the Signal List either by manually adding the node names or pin names to the Signal List. You can create or append nodes to the Signal List by selecting wires within Sketch as described in the following procedure:

1. Make sure that you have an `.ai_grm` file for the design. If there is not one, you can make it by netlisting the design.
2. Select the wires in the design.
3. Display the analysis form that you want to run (e.g. transient)
4. Go to the Input/Output tab

5. Create or modify the Signal List by clicking on the **Select** button and using one of the items from the resulting menu, which include the following:
 - All Toplevel Signals - replaces the existing Signal List with all of the signals in the top level of the design
 - All Signals - replaces the existing Signal List with all signals in the design
 - Get Selected Signals - replaces the existing Signal List with the selected node names or pin names
 - Append Selected Signals - appends the selected items to the current Signal List
6. In the Include Signal Types field, choose one of the following items:
 - Across Variables Only
 - Through Variables Only
 - All Variables
7. Verify the other field values in the analysis form and run the analysis by clicking the **OK** button.

After the analysis completes, you can view the waveforms of the nodes or pins in either Scope or in a DesignProbe.

Viewing Signals Internal to a Template- Sketch

Some templates have internal signals—signals not attached to symbol connection points—that you can view in Scope after running a simulation. The internal variables available for viewing are listed in the Post Processing Information table of the template's template description in the online documentation.

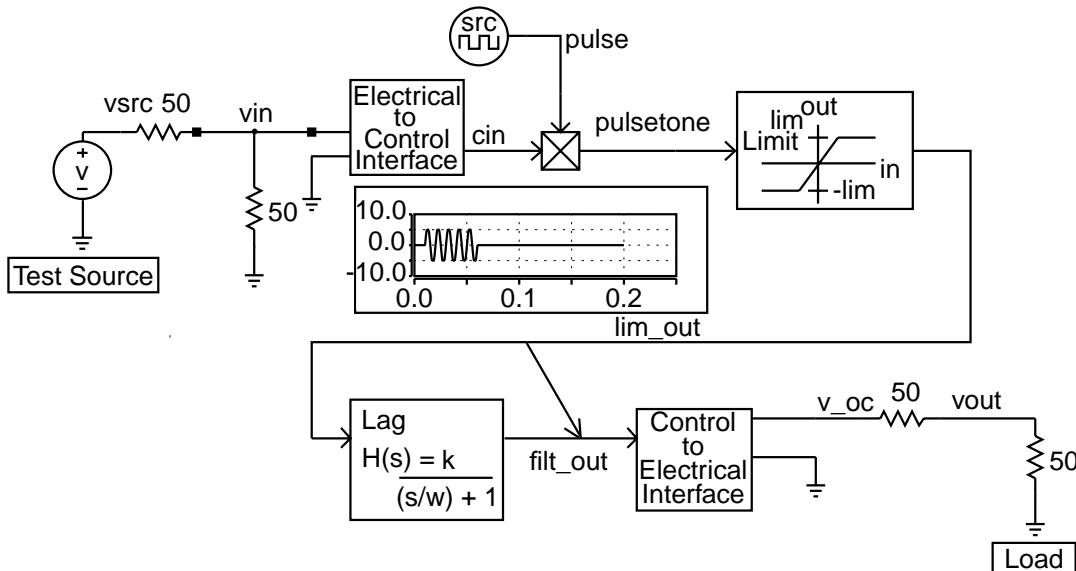
To view these signals in Scope, you must include them in the plot file of the analysis. You can do this one of two ways:

- Include them in the signal list of the analysis
- Use them as arguments for the extract command.

Adding DesignProbes to a Sketch Wire or Pin

A Probe is a Graph window within Sketch with only waveform viewing functionality. You can add a probe to the design at any point in the design process. Like Scope, probes in Sketch use a Signal Manager to control which Plot Files are loaded and viewed in the DesignProbe. You can only view signals (and associated variables) which have been included in the Signal List at the time of analysis in either Scope or a Sketch probe.

The following figure shows the CSP block of the Audio Test System example with a probe:



Whenever you run an analysis that overwrites a currently displayed Plot File, Sketch updates the waveform in the Probe. If you don't want the waveform to be overwritten, you should specify a different Plot File name when you run the analysis.

To add a probe to a Sketch design:

- Move the mouse cursor over the wire or pin, press the right mouse button, and select the **Probe** item from the **Wire Menu** popup.

To probe a specific pin on an instance, it is easiest to move the cursor to the instance symbol rather than try to move to the desired pin. After you select the **Probe** item from the **Symbol Menu** popup, a Select List form appears. You can select the pin you wish to probe from this list.

To probe a different waveform in the existing Probe window:

- Specify the waveform to view by positioning the arrow head of the probe on a wire or pin in the design. In the previous example, the probe arrow

is pointing at the `filt_out` wire. Do this with any pin or wire as long as the associated signal is in the Plot File

To specify which plot file to use for the Probe:

- Specify which Plot File to load using the Probe Signal Manager (**Probe > Probe Signal Manager**)

If a simulation is running or has completed, the waveform is automatically loaded into the Design Probe. In this case, you would not have to select a Plot File.

- You can also control which of the loaded Plot Files are displayed in the probe using the **Display Plotfiles** item in the **Probe Menu** popup.

Making Changes to a Sketch Design

Based on the analysis results, if you determine that you need to make changes in the design, the following topics describe functionality to aid in making design changes:

- Changing Saber Property Values in Sketch
- Changing Property Values in Saber
- Back Annotating DC Values in a Sketch Design

Changing Saber Property Values in Sketch

When you change a property value in a design, an `alter` command is sent to the Saber simulator which modifies the in-memory netlist with the new value. This method allows you to modify properties in Sketch and simulate the modified design without having to netlist.

Changing Property Values in Saber

You can view the instance names, connection nodes, and parameter values using the **Edit > List/Alter...** menu item. This menu item displays the List/Alter Design form. This form allows you to alter the parameters used in the design. Property changes made in this form are not propagated back into the Sketch design.

This form contains the following three tabs (like all SaberDesigner forms you can display the contents of the tab by clicking the left mouse button on the tab):

- The Netlist tab lists the instances in the design and their associated connection points and arguments. You can restrict the parts listed in the Hierarchical Instance List using the Filter field. For example, if you entered `r.*` in the Filter field, this form would only display the resistor instances in the design (e.g. `r.load` and `r.rc`).

In addition to listing the instances (with their associated connection list and parameter values), you can also use the list within this tab to alter part parameters. You can accomplish this task by doing the following:

1. Select the instances that you want to alter using the left mouse button

2. Click the **Edit** button

This action displays an Alter Components dialog box.

3. Enter the new parameter values in the Value fields.

4. Click either the **Apply** or the **OK** button to execute the alter.

- The Parameters tab also allows you to alter parameters within the design. You specify the parameter names in the fields in the Parameter column, and the value in the Value column.

For example, to alter the temperature used in the design to 34 degrees, you would find or enter `temp` in the Parameter field and 34 in the Value field.

- The History tab allows you to re-apply altered parameters, load and save alter commands to Command (`.scs`) files, and sort/delete previously executed alter commands from the history list. The **Delete** button does not undo an alter command.

Back Annotating DC Values in a Sketch Design

After you execute a DC analysis, you can back annotate DC values onto each node in the design by using the **Design > Back-annotate > Place Values** pulldown menu item. When you execute this menu item, the following happens, by default:

- Saber reads the initial point file named `dc`
- Saber places the DC values in a back annotation file called `back`

Chapter 3: *Simulating Sketch Designs*

- Sketch adds the DC values as text on the bottom left end of the wire.

If you want to view DC values from an initial point file other than `dc`, do the following:

1. Display the Back Annotation form (**Results > Back Annotation**)
2. Enter the name of the initial point file
3. Set the Place on Schematic field to **Yes**
4. Click the **OK** button.

If the DC values change (e.g. by re-running DC analysis or editing the initial point file), Sketch does not update the values on the design until you repeat the previous steps.

The previous steps cause the DC values to be placed on the nets in the design according to the signal list specified in the Back Annotation form.

Exiting Saber - Sketch

After you complete your Saber simulation session, you can close the design by selecting the **File > Close > Active** menu item from the pulldown menu bar.

Upon exiting, if you specify **Yes** in the Save Before Closing dialog box, the simulator saves the simulation state to the `design.tbl` file. Saving this file enables you to start your next simulation session as if you never ended the session. All Saber simulation alters, for example, will be saved in this file.

All commands sent to the simulator, and any messages reported in the Transcript Window, are saved in the `design.out` file.

Finding and Debugging DC Operating Points

After you invoke the simulator on the design and verify that the netlist is correctly read in, the next step in the design analysis process is to find an operating point.

An operating point is a set of values that define the steady state of a nonlinear system at `time=0` with all time-varying parameters and their derivatives set to 0 and with noise and ac sources set to 0. Because all derivatives are set to 0, all dynamic elements are effectively removed from the circuit (e.g. capacitors are effectively opened, and inductors are effectively shorted. All time dependent sources are effectively removed.) In electrical circuits, this analysis determines the DC bias values of the design.

Using DC analyses, you inform the simulator to calculate the operating point and store the results in an Initial Point file. The initial point file serves two major purposes:

- It is the operating point used in other simulator analyses (e.g. time-domain analysis and small signal frequency analysis).
- It provides a quick check to determine possible incorrect part parameters.

To find the operating point, you will follow these steps:

1. Performing DC Analysis
2. Evaluating the Operating Point
3. Determining the Next Step
4. Debugging DC Analysis Results

Performing DC Analysis

To find the operating point, you execute the following command:

Step 1. Open the DC Analysis Dialog Box (Analyses > Operating Point > DC Analysis)

Step 2. Verify the contents of the DC analysis fields.

In most cases, the simulator can determine the operating point by using the default settings. Typically, you run the DC analysis by using the default settings and then evaluate the resulting initial point to determine whether any changes need to be made to either the design or the DC analysis fields.

Step 3. Set the amount of information sent to the transcript.

The simulator provides the following two fields to control transcript feedback during DC analysis:

- Monitor informs the simulator to transcribe one of following reports:
 - Total execution time of the analysis (if set to 0).
 - Execution summary and time (if set to -1).
 - Iteration information, such as number of iterations and truncation error for each n th iteration (if set to a positive number).
- Debug reports statistics on each potential solution that the simulator calculates. This feature is typically used when the simulator cannot find an operating point using the default values and you want to determine whether the design is converging on an operating point.

Step 4. Set the DC initial and end point file names.

These fields define the beginning circuit initial and endpoint conditions and the DC analysis results.

- The Starting Initial Point File contains the initial values of all design variables at the beginning of DC analysis. The default file name (zero) sets all continuous time (analog) variables to zero and event-driven (digital) either to undefined or to an initial value if one was set on the digital pin.

- The Ending Initial Point File contains the node values at the completion of DC analysis. You will use this file as the initial point file for other simulator analyses, such as time-domain (transient) and small-signal frequency domain (ac). By default, the simulator names this file `dc`.

If you are completing several DC analysis iterations, you should use the previous end point file as the initial point file of the next run. This technique will decrease the analysis execution time and increase the convergence rate because the simulator does not have to start over with all design variables set to zero on each iteration. The following table shows a naming method for multiple dc analysis iterations:

DC analysis Iteration	DCIP	DCEP
First Run	zero	dc1
Second Run	dc1	dc2
Third Run	dc2	dc

Step 5. Verify other settings in the DC analysis form.

In most cases, the simulator will find the DC operating point using the default settings. This procedure called out key fields that you would use when iterating or debugging a DC operating point. For details on each field in the DC analysis form, refer to the DC analysis reference topic.

Step 6. Execute the DC Operating Point Dialog Box.

Clicking the **Apply** button executes the `dcanalysis` command. The successful DC analysis creates an Initial Point file, specified by the Initial Point File field, containing voltages and currents for every node in the system.

If the simulator did not obtain a solution, refer to the topic titled Hints for Finding Difficult DC Operating Points.

After the simulator calculates an operating point, you should evaluate the calculated operating point.

Evaluating the Operating Point

Because some designs may contain more than one mathematically correct operating point, you should always evaluate the results of a DC analysis. To evaluate the operating point, follow these steps:

Step 1. Display the Operating Point Report Dialog Box.

Within the Guide user interface, you can display the results of a DC analysis in an Operating Point Report by selecting the **Results > Operating Point Report** pulldown menu item.

Step 2. Verify the contents of the fields.

By default, the Display Operating Point form displays the DC values from the last calculated initial point file for the nodes defined in the Signal List in the Report Tool. Typically, the default settings of this form are adequate to evaluate the operating point.

- Specify which initial point file to display.

If you want to display a different initial point file or multiple initial point files for comparison, you can specify the initial point files that you want to display in the Input File From field.

- Specify the signals to display.

The following table compares examples of MAST and VHDL Signal List syntax. You can include a space-separated list of multiple Signal List definitions within the same Signal List field.

MAST Signal List Examples	Definition
<code>/*/*.*</code>	Include all components at the top level.
<code>/.../*.*</code>	Include all instances at, and below the root level.
<code>./</code>	Include all signals in the current instance.
<code>.../</code>	Include all signals in instances below the current instance.
<code>/.../pll.*</code>	Include all signals in all instances of any “pll” component.
<code>pll.u12/</code>	Include all signals in the <code>pll.u12</code> instance.

**MAST Signal List Definition
Examples**

sig1 sig2 Include each signal listed.

- Display error estimation information.

When the Display Initial Point Error field is specified, the simulator transcribes an error estimate for each displayed signal, in addition to initial point node values. The error calculation is part of the DC algorithm. The simulator extracts this information from the `dc_err` file. You should not display the `dc_err` file by itself.

The error values are only available for system variables (i.e. node voltages and vars). They are not available for vals, which is indicated by printing `undef` in the `dc_err` column.

Step 3. Execute the Display dialog box.

Clicking the **Apply** button executes this form. By default, the simulator displays the values for the most recently calculated operating point file to the Report Tool. Once the values are displayed, you can evaluate the initial point to determine whether it is valid.

Determining the Next Step

DC Results Produce Expected Values

You are ready to continue the design process and proceed to one of the following steps:

- Calibrating DC Analysis
- Sweeping Independent Sources
- Checking the Time-Domain Response
- Checking the Frequency Response

DC Results Not as Expected

Refer to the trouble-shooting suggestions in the following topic:

- Debugging DC Analysis Results

Debugging DC Analysis Results

- **Mis-Specified Parameter**
If a mis-specified parameter caused the incorrect DC operating point, you can change the parameter in the netlist by using the **Edit > List/Alter...** menu item and re-running the DC analysis.
- **Connectivity Problem**
If a connectivity problem produces the incorrect DC operating point, you must fix the problem in the schematic, re-netlist the schematic, re-load the design into the simulator, and then re-run the dc analysis.
- **No Operating Point Found**
If the simulator didn't find an operating point using default values, adjust the DC analysis arguments using the procedures described in the topic titled Hints for Finding Difficult DC Operating Points.
- **Wrong Operating Point Found**
If the simulator found an unexpected operating point, adjust the DC analysis arguments using the procedures described in the topic titled Simulator Found the Wrong Operating Point.
- **Difficult Circuit Configurations**
There are some simple circuit configurations for which DC analysis may have a hard time finding an operating point. For a list of such configurations and some suggested approaches, see the topic Simple Circuits That are Hard for DC Analysis.

Hints for Finding Difficult DC Operating Points

If the simulator failed to obtain a solution by using the default values, you can use the following process to find the operating point. You may also want to check the list of circuits with difficult DC operating points.

After you find a solution, you can evaluate the solution.

1. If you get an error message indicating a singular matrix (e.g. ALG_SINGULAR_JACOBIAN, MAST_NO_NUM EQUATION, or MAST_NO_NUM_REFERENCE), there are likely to be errors in the circuit. Use the information provided with the error message to fix the problem, and re-run DC analysis.

2. Determine whether `dcerr` file was created in the local directory.

Do not confuse the `dcerr` file with the `dc_err` file created by a successful DC Analysis. If the `dcerr` file wasn't created, contact customer support.

3. Determine whether the `dcerr` file contain reasonable results.

You can display the operating point by using the `DIisplay` command.

SABER Syntax:

```
di dcerr
```

If the `dcerr` file does not contain reasonable results, go to Step 4.

Otherwise, re-run DC analysis with Debug mode on and using the `dcerr` file as the input to the analysis. (For all remaining steps, you should run with Debug mode on.)

SABER Syntax:

```
dc (debug yes, dcip dcerr
```

If DC analysis still cannot find a solution, examine the output in the transcript to determine whether the design is converging.

4. Check whether the signals settled on a solution. If the signals haven't settled, increase `DR_Tsettle` and repeat the dynamic ramping algorithm, for example:

SABER Syntax:

```
dc (dr_tsettle 1, algstep n algstart dyn_r,  
debug yes
```

Repeat this step with a larger value for `dc_tsettle` if the signals still don't settle on a solution (i.e. the last few time steps are still relatively small).

If the algorithm fails with an oscillation after reaching `dr_tsettle`, use the values in the `dcerr` file as the initial point.

5. Increase density by a factor of 3, reduce the truncation error to 5m and re-run the dynamic ramping algorithm, for example:

SABER Syntax:

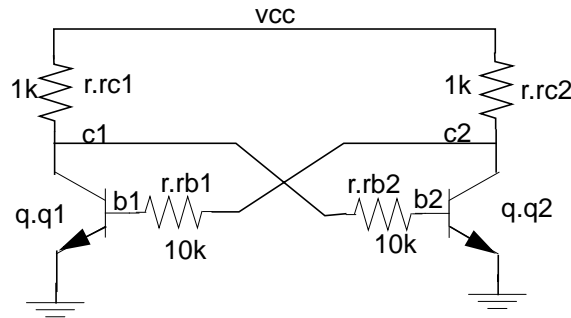
```
dc (den 3, dr_terr 5m, dr_tsettle 1, algstep n,  
    algstart dyn_r, debug yes
```

Simulator Found the Wrong Operating Point

For systems with multiple stable operating points, such as latches, flip-flops, and counters, an unbiased dc analysis might “get stuck” between operating points, rather than finding one. On the other hand, depending on the design, a dc analysis might have no trouble finding an operating point, but the operating point found might not be the one you are most interested in. In each of these cases, you can give the dc analysis a “boost” in the direction of a particular operating point by using the holdnodes variable of the dc analysis.

After you determine how to help the simulator find the expected solution, you can re-run the DC analysis.

As an example, consider the circuit shown in the following figure:



This classic circuit has one unstable and two stable operating points. The two stable operating points are:

- node c1 is “high” and node c2 is “low”
- node c2 is “high” and node c1 is “low”

If you run the DC analysis on this design and then use the display command to see the results, as follows:

SABER Syntax:

```
dc
di dc
```

The resulting display is the following:

```
time          0
-----
b1            0.8286
b2            0.8286
c1            1.655
c2            1.655
i(v.vcc)     -0.01669
q.q1/bp      0.8286
q.q1/cp      1.655
q.q1/ep      0
q.q2/bp      0.8286
q.q2/cp      1.655
q.q2/ep      0
vcc          10
0            0
```

Because nodes `c1` and `c2` should have different values for each stable operating point, it is clear that the result of the DC analysis, while an operating point, is not one of the stable ones. Using the `holdnodes` variable in DC analysis, you can break the “deadlock” For example, to find the operating point where `c1` is “high” and `c2` is “low”, you can use the `holdnodes` variable, as follows:

SABER Syntax:

```
dc (hold c1 10, relhold y)
di dc
```

This command runs the `dc` analysis twice, with the initial point file from the first run being used as input to the second. On the first run, `c1` is held at `10v`.

On the second, `c1` is allowed to “seek its own level.” The result is the following display:

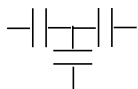
```
time      0
-----
b1        0.07146
b2        0.8367
c1        9.167
c2        0.07145
i(v.vcc)  -0.01076
q.q1/bp   0.07146
q.q1/cp   9.167
q.q1/ep   0
q.q2/bp   0.8367
q.q2/cp   0.07145
q.q2/ep   0
vcc       10
0         0
```

As shown in the previous Operating Point report, the DC values at nodes `c1` and `c2` are as expected. You can now proceed to the next step in the design process.

Simple Circuits That are Hard for DC Analysis

The following list describes simple circuit configurations that DC analysis may have a hard time finding the Operating Point for:

- Single node connected only to capacitors and current sources (cut set of capacitors)



In this configuration, all capacitors are effectively open circuits so the node may float in DC analysis. You can avoid this problem by setting an initial condition at the node.

- Loop containing only inductors and voltage sources

This configuration appears as a loop of shorts in DC analysis. You can avoid this problem by setting an initial condition for the loop current.

- Current source driving a capacitor

Because the capacitor is seen as an open circuit, an operating point may be difficult to obtain (the across voltage on the capacitor is not defined). Making the current source something other than ideal (giving the current somewhere to go) may help.

- Voltage source driving an inductor

Because the inductor is seen as an short circuit, an operating point may be difficult to obtain.

- Two inductors in series with conflicting current initial conditions
- Two capacitors in parallel with conflicting voltage initial conditions

Chapter 4: *Finding and Debugging DC Operating Points*

Checking Time-domain Response-Process Overview

After you have completed your design, you can verify that the time-domain response of the design meets specifications. The simulator determines the time-domain response of a design by using the transient analysis.

To check the time-domain specifications of your design, use the following process:

1. Specifying the First Transient Data Point
2. Performing Transient Analysis
3. Extending a Transient Analysis
4. Viewing the Transient Analysis Results
5. Measuring the Analysis Results
6. Determining the Next Step
7. Checking the Frequency Spectrum of a Time-Domain Signal
8. Fourier, FFT, and Distortion Analyses Comparison
9. Performing Fourier Analysis -- Process Steps
10. Performing FFT Analysis -- Process Steps

Specifying the First Transient Data Point

Because the transient analysis uses the initial point as the first data point in an analysis run, you must find the operating point of the system prior to running the transient analysis, using one of the following methods:

- In the Transient Analysis form, specify `Yes` in the Run DC analysis First? field. This selection informs the simulator to execute DC analysis to find the operating point and then run the transient analysis using the

previously calculated operating point as the first data point. The Transient form is discussed in a later step.

- Run the DC analysis separately by selecting the **Analysis > Operating Point > Operating Point** pulldown menu item. In most cases, the simulator will find the correct operating point using the default values in the DC Operating Point form.

If an Oscillator is Driving Source, Manipulate Initial Point File

Because oscillators rely on the amplification of noise to begin and noise is not inherent in the simulator, you must alter some of the node values in the initial point file to “kick start” the oscillator at the beginning of the transient run.

Performing Transient Analysis

Step 1. Display the Transient Dialog Box (Analyses > Time-domain > Transient...).

Step 2. Specify the required transient analysis fields.

In order to execute a transient analysis, you must specify the following information:

- End Time (Basic tab) defines the time at which the transient analysis stops. For example, if the driving source of the design is a sinusoidal function with a period of 10us and you want to view the transient response of the first 5 cycles, you enter 50u (5 x 10u) as the value for this field.
- Start Time (Basic tab) defines the time at which the simulator begins the transient analysis. By default, this time comes from the initial point file. It is 0 if the initial point has been created by a DC analysis. Specifying a time that is different from the one in the initial point file allows you to specify a start time on the graph, but if the start time disagrees with the time in the initial point file, the resulting graph could be distorted.
- Time Step (Basic tab) is used by the simulator during transient analysis to make an initial guess at the next solution point. As a rule of thumb, you should set this value to the smallest of the following parameters in your design:

- 1/10th of the smallest relevant time constant in the design
- Shortest rise or fall time of a square/pulse wave driving source
- 1/100th of the input period of a sinusoidal driving source

Step 3. Specify which analysis waveforms to create.

Time-Domain Transient Analyses form provides the following two fields to specify how waveform data is to be saved for plotting and analysis:

- The Plot File field (Input/Output tab) specifies the name of the plot file, which contains simulation results for the signals defined by the Signal List field. The size of the plot file is a function of the number of data points and the number of signals in the Signal List. By default, the simulator creates a plot file named “*tr*” for each transient run. If you do not want the simulator to create a plot file, you fill in the field with a “_”.
- The Signal List field (Input/Output tab) specifies the signals for which simulation results are to be saved for plotting within Scope. By default, the signal list includes only the signals at the top level of the design hierarchy. If you need to view other signals within the design hierarchy, you must add the signal names in the Signal List field.

The following table shows examples of MAST Signal List syntax. You can include a space-separated list of multiple Signal List definitions within the same Signal List field.

MAST Signal List Examples	Definition
<code>/*/*.*</code>	Include all components at the top level.
<code>/.../*.*</code>	Include all instances at, and below the root level.
<code>./</code>	Include all signals in the current instance.
<code>.../</code>	Include all signals in instances below the current instance.
<code>/.../pll.*</code>	Include all signals in all instances of any “pll” component.
<code>pll.u12/</code>	Include all signals in the <code>pll.u12</code> instance.
<code>sig1 sig2</code>	Include each signal listed.

If you are unsure about which signals to specify, you can use the default

setting (all signals in the root of the design). If you need more signals later, you can re-run the transient analysis with additional signals defined in the signal list. For long simulation run times, extract additional signals from the Data file using the **Extract > Add Signals to Plot File** menu item.

Step 4. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms. See Viewing the Transient Analysis Results for additional information.

Step 5. Decide whether to save raw simulation results.

Data Files contain simulation results for each data point in the simulation. Data files can become quite large; unless you are going to use the data file for any of the following purposes, you probably do not need to create one:

- Extracting additional signals for a plot file
- Extracting an operating point
- Transforming transient data into the frequency domain using Fourier Analysis
- Running Stress analysis

Because it is much faster to extract plot information from a data file than to re-run the analysis (especially for long simulation runs), you need to carefully consider whether or not to create a data file.

You specify the data file name in the Data File field under the Input/Output tab. By default, the simulator creates a data file named “tr” for each transient run. If you do not want the simulator to create a data file, fill in the field with an “_”.

Step 6. Verify the first data point value.

The simulator uses the initial point, defined in the Initial Point File field of the Input/Output tab, as the first data point in the transient analysis run. You should verify that the simulator is using the correct initial point file prior to running the transient analysis.

You should also specify the Sample Point Density setting on the Calibration tab to a value equal to or larger than the value used during the generation of the initial point file. The simulator multiplies the number of sample points defined in the model by the factor defined in this field. A larger number yields more accurate results at the expense of longer simulation run times.

Step 7. Execute the transient analysis.

Clicking on the **Apply** button executes the analysis. The simulator determines the transient response of the design by using the initial point file to determine the initial circuit conditions and then calculating how the design responds to the passage of time. For information on all fields available with Transient analysis and how the simulator determines the next timestep, refer to the SaberBook online system.

After the simulator has finished simulating the design based on your transient analysis specifications, you can view the resultant waveforms in Scope.

Extending a Transient Analysis

To resume a transient analysis from the end point of the previous analysis, use the `continue` command as follows: From the drop-down menus choose: **Analyses > Continue > Transient**, complete the Continue Time-Domain Transient Analysis dialog box fields, and click **OK**. This analysis can resume where the last transient analysis ended and continue to a newly specified end time. Instead of creating a new plot file, the `continue` command appends to the original plot- file, producing a graph that looks like the original transient analysis has been run with a longer end time.

You can create a separate plot file that begins where the previous one ends using the general method that repeats the original **Analyses > Time Domain > Transient** command with these different settings:

- Basic tab, specify the End Time to be some time after the previous End Time. Leave the Start Time set to `Default`.
- Input/Output tab, specify the Initial Point File to be `tr`. Specify the End Point File to be `tr`. Specify Allow IP File = EP File to be **Yes**.

By leaving the Start-Time set to `Default`, the resulting plot file gets its starting value from the specified Initial Point File (in this case, from the

previous transient analysis run). Setting Allow IP File = EP File to **Yes** means that the Initial-Point and End-Point files can be given the same specification, without causing an error message. The simulator uses the End-Point File from the previous run as the Initial-Point File for this run.

If the plot-file names for each run are the same (for example tr), then each run overwrites the previous plot file. This is not a problem in the current session of Scope because you can still view the earlier plots (Scope stores them internally and numbers them sequentially in the signal manager). If you want to view the plots in a later session of Scope, only the last plot-file is available. An alternate approach is to give the plot files different names (for example, tr1 and tr2) for each run. This creates multiple plot files instead of overwriting and lets you review your work at a later time.

Viewing the Transient Analysis Results

You use the Signal Manager within Scope to manage and display signals from the various plot files that you create during the analysis process. Overall, the procedure for viewing waveforms from a transient analysis, is as follows:

- Add the plot file to the Signal Manager's plot file list.
- Open the plot file.
- Select signals to plot.
- Plot the selected signals in a graph.

You can do this step-by-step after an analysis, or you can set up Guide to perform all or part of this process automatically.

The following procedure explains in more detail how to plot waveforms after an analysis:

Step 1. Add the plot file to the Signal Manager and open it.

The simulator does this automatically if you specify one of the following under the Plot After Analysis menu field (Basic tab) before starting the analysis:

- **Yes - Open Only**
- **Yes - Append Plots**
- **Yes - Replace Plots**

In all of the above cases, Guide invokes Scope (if necessary) and opens the Signal Manager and appropriate plot file.

However, if you specify **No** in the Plot After Analysis menu field, the simulator does not open the Signal Manager. You can open the last created plot file either by selecting the **Results > View Plot Files in Scope** pulldown menu item. This activates the View Plotfiles dialog box. Under the Plot File pulldown menu field, select one of the following:

- **Last** selects the most recent plot file generated by the analysis. This is the default.
- **Plot File Names** allows you to specify a list of plotfiles (separated by spaces) in the Plot File field. You can use a browser to select the plot files by clicking on the **Browse** button.

Under the Plot Action field, select one of the following actions:

- **Open Only** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
- **Append Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, adding the new waveforms in addition to the corresponding waveforms from the previous analysis.
- **Replace Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, replacing the corresponding waveforms from the previous analysis.

The Append and Replace options are convenient to use during

design optimization, when repeatedly running analyses to see the effects of altering design parameters.

After making the settings described above, you open the Signal Manager by clicking on the **OK** button or the **Apply** button. The Signal Manager dialog box appears, along with a Plot File window for the plot file you specified.

If you click on the **OK** button, the View Plotfiles dialog box closes. If you click on the **Apply** button, the dialog box remains open. Click on the **Close** button to close the dialog box.

Step 2. Select the signals.

Assuming that you did not select the Append or Replace Plot Action options in the preceding step, you need to select the signals to be plotted. In the Plot File window, select each signal by clicking on it with the left mouse button. Unselect a signal by clicking on it again (or use the **Deselect** button or Signal menu).

Step 3. Plot the selected signal(s).

You can plot the selected signals by clicking on the **Plot** button or by moving the cursor into the graph window and pressing the middle mouse button. (You can also plot a signal simply by double clicking on its name in the Plot File window.)

The selected signals in the plot file appear as waveforms in the active Graph window.

After you plot the signals, you can analyze the results in various ways and determine the next step in the design process.

Measuring the Analysis Results

Scope provides numerous pre-defined automated measurements to dramatically decrease the time needed to extract specification data from plotted waveforms. To perform a measurement on a waveform within Scope, follow these steps:

Step 1. Display the Measurement Tool. (Tools > Measurement).

Step 2. Select the appropriate measurement.

After you display the this tool, you can select which measurement you want to perform. In the Measurement field, you select the measurement by clicking on the downward arrow and selecting the measurement from within the various measurement categories.

The following table lists the measures accessible in the time-domain category:

Measurement	Definition
Falltime	Displays the fall time between the upper and lower levels of a waveform
Risetime	Displays the rise time between the upper and lower levels of a waveform
Slew Rate	Displays the difference between the upper and lower levels of a waveform divided by the risetime or falltime of the edge
Period	Displays the period of a waveform cycle relative to the topline and baseline
Frequency	Displays the frequency of a periodic waveform relative to the topline and baseline
Duty Cycle	Displays the duty cycle of a periodic waveform relative to the topline and baseline values
Pulse Width	Displays the pulse width relative to the topline and baseline
Delay	Displays the delay between waveform edges of two signals
Overshoot	Displays the overshoot of a waveform relative to the topline
Undershoot	Displays the undershoot of a waveform relative to the baseline
Settle Time	Displays the settle time of the waveform with respect to a settle level and a specified settle band (% of amplitude)

Step 3. Select the Signal to Measure.

You can specify the signal to measure in the Signal field. You can specify the signal by one of the following methods:

- Click on the arrow button and select a signal from the resulting list.
- Select the signal from the legend in the current active graph.

Regardless of the method you choose, you must specify a signal prior to executing the measurement.

Step 4. Set the data range to measure.

You can control the data range that is used to perform the measurement in the Apply Measurement to: section of the Measurement dialog box. You must specify one of the following data ranges:

- Entire Waveform calculates the measurement on the entire waveform, regardless of what is viewed in active graph.
- Visible X and Y range only calculates the measurement for the X and Y range visible on the current active graph.

Step 5. Perform the measurement.

You perform a measurement by clicking on the **Apply** button. This performs the measurement on the specified signal and adds the corresponding information to the active graph.

If any existing waveform is automatically updated due to an automatic plot action (such as Append) specified in an analysis, all measurements that depend on that waveform are updated at the same time.

Step 6. Manipulate the measurement information.

During the course of data analysis within Scope, you may want to manipulate which measurements can be viewed. This can be done using the Measure Results form. You can open this form either by selecting the **Graph > Measure Results...** pulldown menu item or by double-clicking on measurement text within an active graph. After you display this form, you can manipulate measurement information by using the following buttons:

- Delete Measurement deletes the selected measurement from the active graph.
- Delete All deletes all measure text/graphics from the active graph.
- Show All Values shows all measurements listed in the left scroll list in the graph.

- Hide All Values retains but does not display the measurements listed in the left scroll list.
- Show/hide status bubbles allow you to change the visibility of a measurement on the active graph on a per measurement basis.

Determining the Next Step

Using the signal viewing and measurement capabilities of Scope, you determined whether your design meets your pre-determined specifications.

If it meets specification, then you can proceed with one of the following:

- Continue the transient run from the last data point using the **Analyses > Continue > Transient** pulldown menu item.
- Verify the small-signal frequency response using AC analysis, described in the next chapter.
- Examine the frequency spectrum of a time-domain waveform using fourier or FFT transform as described in the next section.
- Tune the design parameters by using statistical, parametric, or stress analysis, described in Chapter 8 through Chapter 10, respectively.

If the design does not meet your specifications, you can take corrective action by:

- Alter a design/instance parameter(s).

If the design change does not require you to modify the connectivity, you can modify the design parameters by using the Alter form available through the **Edit > List/Alter...** menu item.

- Edit the schematic and re-analyze.

If the design change requires you to change the connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design (**Design > Netlist *design_name***), and then re-load the design in the simulator (**Design > Simulate *design_name***).

Checking the Frequency Spectrum of a Time-Domain Signal

In order to use these analyses, you must have the optional Spectral Analysis license.

After you complete a transient analysis run, you can use the fourier analyses (Fourier and FFT) to examine the spectral components of the system.

Depending on whether your time-domain signal is periodic, you will chose either Fourier or FFT Analysis to determine the frequency spectrum of your time-domain waveforms.

- Fourier Analysis transforms periodic waveforms into a frequency spectrum. Since all periodic waveforms can be described by a set of sinusoids, this analysis produces a line spectrum showing the spectral content at the DC value, fundamental frequency, and specified number of harmonics.
- FFT Analysis transforms non-periodic waveforms into a continuous-style output showing each point in the FFT.

The simulator also provides the IFFT Analysis to transform either frequency spectrums (produced by the FFT analyses) or frequency sweeps (produced by AC analysis) waveforms into the time-domain.

Because the fourier, fft, and distortion analyses each provide spectral information, the following table compares these analyses.

	Fourier	FFT	Distortion
Nature	Large Signal	Large Signal	Small Signal
Concept	Transform time-domain data to frequency domain	Transform time-domain data to frequency domain	Small signal frequency domain analysis
Type	Post-processor of time-domain data	Post-processor of time-domain data	Stand-alone analysis
Algorithm	Discrete Fourier Transform of periodic signal	Approx. of continuous fourier transform	Volterra series, based on Taylor series approximation

	Fourier	FFT	Distortion
Results	Spectral output; showing DC, fundamental frequency and <i>n</i> th harmonics	Continuous frequency distribution showing each point in the FFT	Distortion products (e.g. harmonic, intermodulation) of each specified signal
Application	Periodic signals with strong non-linearities or large signals	Non-periodic signals with large signals or strong non-linearities	Harmonic distortion based on small signal model

To check the spectral components of time-domain waveforms in your design, you can perform either the fourier or FFT analyses, view the resultant waveforms in Scope, and evaluate the results to determine the next step in the design process. The following sections describe how to perform the Fourier and FFT analyses, respectively.

Performing Fourier Analysis

Fourier analysis is a post-processing command that reads one period of a signal (calculated by using a time-domain analysis) and applies a continuous fourier transform to describe the dc component, fundamental frequency, and the specified number of harmonics of the named signal.

Fourier analysis uses the fourier series representation of a periodic piece-wise continuous function, $f(t)$, as an infinite series.

For periodic function, the frequency components are all integral multiples of the fundamental frequency as shown in the following equation where C_i are the amplitudes and ϕ_i are the phases at the multiples of the fundamental frequency (ω):

$$f(t) = C_0 + C_1\cos(\omega t - \phi_1) + C_2\cos(2\omega t - \phi_2) + \dots + C_i\cos(i\omega t - \phi_i)$$

Fourier analysis computes the spectrum of a signal, based on the fundamental frequency of the specified waveform, by applying the Discrete Fourier Transform (DFT) to a periodic waveform generated by a transient analysis. To execute a fourier analysis, you perform the following steps:

Step 1. Select the Fourier Analysis form (Analyses > Fourier> Fourier...).

Step 2. Specify the Signals to transform.

In order to execute a Fourier analysis, you must specify the following information:

- Specify the signal names to transform.

You define the signal to transform in the Signal List field (Input/Output tab). You can either type the signal names in the field or use the pre-defined selections displayed by pressing the **Select** button. Although the syntax for the values in this field is identical to the signal lists used in AC and transient analysis, this field can only include system variables, such as currents through voltage sources and node voltages.

- Verify the data file name from the transient analysis.

The simulator uses the data file from a previous transient analysis run as the source for the fourier analysis. You should verify the value in the Input Data File field to insure that the simulator uses the correct data file.

Step 3. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 4. Specify the fundamental frequency and time period to transform.

You use the fields in the Basic tab to define the time period(s) and fundamental frequency that the simulator should use during the Fourier analysis. It is very important that the fundamental frequency value specified in the form matches the fundamental frequency of the signal to transform. If they differ, the results will be incorrect. You determine the fundamental frequency by examining the input sources.

- If a single frequency is present at the input, use the frequency of the input source as the fundamental frequency value.
- If multiple frequencies are present at the inputs, use the greatest common divider of all input frequencies. For example, if your design

is driven by a 900Hz and 1KHz sources, you should use 100Hz as the fundamental frequency.

You can use one of the following methods to specify the fundamental frequency and the time period in the Basic tab:

- Frequency & Location allows you to specify the fundamental frequency and a time data point either at the beginning or ending time of the period to transform.
- Location allows you to specify the beginning and ending time. The fundamental frequency is calculated as the inverse of the difference between the beginning and ending time.

After you specify the contents of the required fields, you should verify the contents of the optional fields as described in the next step.

Step 5. Verify the following values.

Although optional, you should verify the contents of the following fields:

- Specify the number of harmonics to compute.

You specify the number of harmonics to compute, including the fundamental, in the Number of Harmonics field on the Basic tab. For example, if you specify the default value of 10, the simulator displays the fundamental frequency and the corresponding nine harmonics.

- Specify whether to Calculate THD.

If you specify *yes* in this field on the Control tab, the simulator calculates the total harmonics distortion. This value is the energy of the unwanted spectral components as a fraction of the total signal energy. This value is independent of the number of harmonics calculated. This value is displayed in the Transcript Window, after the analysis completes.

- Verify the windowing function.

You can apply different windowing functions to filter the input data prior to transformation. The Fourier analysis within the simulator contains pre-defined Rectangular, Barlett, Hann, Hamming, Blackman, and Flattop windows. You can select the appropriate window by pressing the **Windowing Function** arrow button on the Control tab. You can also define your own windowing function as described in the SaberBook online system in the “Defining External Windowing Functions” topic.

After you verify the values in the Fourier analysis form, you can execute the form.

Step 6. Execute the analysis.

By default, the simulator calculates the frequency response based on fourier transform of either a section or entire data file from a previous transient analysis run. The results for each system variable are stored on a linear scale in the data file and plot file named `fou`. The plot file contains a spectral-style output showing the DC component, the fundamental frequency, and the specified number of harmonics. All default values can be modified within the Fourier analysis form.

For reference information on this command, refer to the SaberBook online system in the SaberDesigner Reference Library topic.

Step 7. Plot the Fourier analysis results.

After the Fourier analysis run completes, you can view results within Scope.

The following procedure outlines the process of viewing and manipulating waveform data within Scope.

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager in Scope (**Results > View Plotfiles in Scope**).
2. Select the signal(s) you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.
4. Use the waveform manipulation (e.g. pan/zoom) and measurement capabilities within Scope to analyze the data.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification.

Step 8. Analyzing the Fourier analysis results.

You can view the frequency spectrum of the specified input waveform in Scope and analyze the results, including:

- Magnitude and phase of the spectral components
- Real and Imaginary parts of the spectral components

- Amount of harmonic distortion (displayed in the transcript window, if you informed the simulator to calculate THD)

After you analyze the Fourier analysis results, you can proceed with the design analysis process:

- If the analyzed results are as expected, proceed either to the small signal frequency sweep (AC) analysis, as described in Chapter 5 or to the analyses used to tune design parameters (parameter sweep, monte carlo, sensitivity, and stress) as described in Chapter 8 through Chapter 10, respectively.
- If the analyzed results are not as expected, make design changes, re-run the transient analysis, and re-generate the frequency spectrum by using Fourier analysis.

Performing FFT Analysis

The Fast Fourier Transform is a post-processing command that calculates the frequency components of a section of time. Because this analysis requires time domain data, you must run a transient analysis prior to executing this analysis. The FFT transform should be used for non-periodic functions. If the function is periodic, you use the Fourier transform as described in the previous Performing Fourier Analysis - Process Steps topic.

Because non-periodic functions cannot be represented by a fourier series, the simulator uses a fourier integral representation. It is no longer sufficient to find the fourier coefficients at a set of harmonic frequencies. Instead, a continuous range of frequencies is calculated showing the value of each FFT data point.

To execute a FFT analysis, you follow these steps:

Step 1. Display the FFT Dialog Box (Analyses > Fourier > FFT...).

Step 2. Specify Signals to Transform.

In order to execute a FFT analysis, you must specify the following information:

- Specify the signal names to transform.

You define the signal to transform in the Signals to Transform field. You can either type the signal names in the field or use the pre-defined functions displayed by pressing the **Select** button.

You must enter the names of the signals to transform exactly as they have been specified in the transient Plot file. You cannot use wildcards in this field. If you do not specify any signal names, all applicable signals from the transient Plot file will be transformed.

- Verify source transient plot file name.

The simulator uses the plot file from a previous transient analysis run as the source for the FFT analysis. You should verify the value in the Transient Plot File field to insure that the simulator uses the correct plot file.

Step 3. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 4. Verify the fields in the Data Manipulation tab.

- Verify the number of points in the FFT.

The Number of Points field specifies the number of data points used in the transform. This value must be a power of 2, such as 256, 512, or 1024.

- Verify the Time Segment to transform.

The Time Data Start and Time Data Stop fields define the section of time-domain to be used in the transform. You can specify any of the following values in these fields:

- *begin* to define the first data point in the transient plot file
- *end* to define the last data point in the transient plot file
- *time* to define a specific time in the transient plot file

During the FFT transform, the simulator extracts data points at equally spaced linear intervals (defined by the Number of Points field) across the defined Time Segment.

- Verify the windowing function.

You can apply different windowing functions to filter the input data prior to transformation. The Fourier analysis within the simulator contains pre-defined Rectangular, Barlett, Hann, Hamming,

Blackman, and Flattop windows. You can select the appropriate window by pressing the **Windowing Function** arrow button. You can also define your own windowing function as described in the SaberBook online system in the Interfacing to External Windowing Functions topic.

Step 5. Execute the analysis.

By default, the simulator calculates the frequency response based on a FFT transform of either a section of or the entire plot file from a previous transient analysis run. By default, the results for each system variable are stored on a linear scale in the plot file named `fft`. The plot file contains a continuous-style output. All default values can be modified within the FFT analysis form.

For more information on the FFT analysis, refer to the SaberBook online system.

Step 6. Plot the results in Scope.

After the fourier analysis run completes, you can view results within Scope. The following procedure outlines the process of viewing and manipulating waveform data within Scope.

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager (**Results > View Plotfiles**).
2. Select the signals you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.
4. Use the waveform manipulation (e.g. pan/zoom) and measurement capabilities within Scope to analyze the data.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification.

After you analyze the analysis results, you can do either of the following:

Chapter 5: *Checking Time-domain Response-Process Overview*

- If the analyzed results are as expected, proceed either to the small signal frequency sweep (AC) analysis, as described in Chapter 5 or to the analyses used to tune design parameters (parameter sweep, Monte Carlo, Sensitivity, and Stress) as described in Chapter 8 through Chapter 10, respectively.
- If the analyzed results are not as expected, make design changes, re-run the transient analysis, and re-generate the frequency spectrum by using FFT analysis.

Sweeping Independent Sources

DC Transfer analysis sweeps an independent DC voltage or current source over a user-defined range of values and computes the DC operating point for each sweep value.

This analysis produces a plot file that can be viewed in Scope and allows you to determine how an independent source affects the DC operating point over a range of values. DC Transfer analysis may be useful to find an approximate biasing point for your circuit.

NOTE

For information on sweeping parameters, refer to the Vary looping command described in Chapter 11.

Executing DC Transfer Analysis

To execute a DC Transfer Analysis, perform these steps:

Step 1. Display the DC Transfer Dialog Box.

Within the Guide user interface, you can open the DC transfer analysis form by selecting the **Analyses > Operating Point > DC Transfer** pulldown menu item.

Step 2. Specify the independent voltage source.

In the Independent Source field, you may specify the source one of two ways. One way is to specify both the *parameter* and the *instance name* of the source to be swept. For example, if you wanted to sweep the voltage of a voltage source instance named `vcc`, you enter `v(v.vcc)` in this field. The other way is to only specify the *instance name* of the source to be swept, entering `v.vcc`, for example.

Step 3. Define the sweep range.

In the Sweep Range field, select one of the following: Step By, Linear Steps, Log Steps, or Set Values.

- Step By sweeps the source from a to b by steps of size c :
from a to b by c
- Linear Steps sweeps the source from a to b in n linearly spaced steps:
from a to b in n lin steps
- Log Steps sweeps the source from a to b in n logarithmically spaced steps:
from a to b in n log steps
- Set Values sweeps the source from a to d in user defined steps of any increment:
in a b c d

Step 4. Verify the waveforms to save.

All variables used for the DC analysis forms are available for this analysis. In addition to the DC analysis fields, you should also verify the file names in the Data File and Plot File fields.

- Plot files contain waveform results for the signals defined by the Signal List field. The size of the plot file is a function of the number of data points and the number of signals in the Signal List field.
- Data Files contain simulation results of each data point in the simulation.

If you are making multiple runs of the same analysis, you should rename the data and plot file for each run. Otherwise, the simulator will overwrite the previous data and plot file results. The simulator stores analysis results in two types of files, Data Files and Plot Files. You can suppress the creation of either file by specifying “_” (underscore) as the name of the file.

By default, the simulator saves these files with the value of “dt”.

Step 5. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms. See "Viewing the DC Transfer Results" on page 6-4 for additional information.

Step 6. Execute the analysis.

Clicking **Apply** executes the DC Transfer form. During this analysis run, the simulator determines the DC operating point at each sweep value and stores the information in the data and plot files. Depending on which simulator you are running, this form executes the `verias:dt` VeriasHDL command or the `dtanalysis` Saber command in whichever format was selected in the Sweep Range field as follows: (examples show Saber command syntax)

- Step By

```
dt(sweep <ind_src>, swrrange from <value> to <value> by  
<value>
```

- Linear Steps

```
dt(sweep <ind_src>, swrrange from <value> to <value> lin  
<value>
```

- Log Steps

```
dt(sweep <ind_src>, swrrange from <value> to <value> log  
<value>
```

- Set Values

```
dt(sweep <ind_src> swrrange in <value> <value> <value>  
<value>
```

For example, to determine the effects of varying a 15 volt supply +/- 0.5 volts, you can fill out the DC transfer form by using `v(v.vcc)` as the

independent source and Step By as the Sweep Range, with 14.5 and 15.5 as the sweep boundaries, and 0.1 as the sweep step. When you execute the form, the following command will be executed in the Transcript window:

```
dt (sweep v(v.vcc), swr from 14.5 to 15.5 by .1
```

For more information on the DC Transfer analysis command, refer to the online documentation system.

Viewing the DC Transfer Results

After the simulator finishes the analysis run, you can view the results in Scope. You can view the results of a DC Transfer analysis run by following these steps:

Step 1. Add the plot file to the Signal Manager and open it.

Guide does this automatically if you specify one of the following under the Plot After Analysis menu field (Basic tab) before starting the analysis:

- **Yes - Open Only**
- **Yes - Append Plots**
- **Yes - Replace Plots**

In all of the above cases, Guide invokes Scope (if necessary) and opens the Signal Manager and appropriate plot file.

However, if you specify **No** in the Plot After Analysis menu field, Guide does not open the Signal Manager. You can open the last created plot file either by selecting the **Results > View Plot Files in Scope** pulldown menu item. (You can also click on the **Last Plotfile** icon in the Guide icon bar.) This activates the View Plotfiles dialog box. Under the Plot File pulldown menu field, select one of the following:

- **Last** selects the most recent plot file generated by the analysis. This is the default.
- **Plot File Names** allows you to specify a list of plotfiles (separated by spaces) in the Plot File field. You can use a browser to select the plot files by clicking on the **Browse** button.

Under the Plot Action field, select one of the following actions:

- **Open Only** does the following:

- Invokes Scope, if necessary.
- Opens the Signal Manager window, if necessary.
- Opens the latest version of the plot file specified in the Plot File field.
- **Append Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, placing the new waveforms in addition to the corresponding waveforms from the previous analysis.
- **Replace Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, replacing the corresponding waveforms from the previous analysis.

The Append and Replace options are convenient to use during design optimization, when repeatedly running analyses to see the effects of altering design parameters.

After making the settings described above, you open the Signal Manager by clicking on the **OK** button or the **Apply** button. The Signal Manager dialog box appears, along with a Plot File window for the plot file you specified.

If you click on the **OK** button, the View Plotfiles dialog box closes. If you click on the **Apply** button, the dialog box remains open. Click on the **Close** button to close the dialog box.

Step 2. Select the signals.

Assuming that you did not select the Append or Replace Plot Action options in the preceding step, you need to select the signals to be plotted. In the Plot File window, select each signal by clicking on it with the left mouse button. Unselect a signal by clicking on it again (or use

the **Deselect** button or Signal menu).

Step 3. Plot the selected signal(s).

You can plot the selected signals by clicking on the **Plot** button or by moving the cursor into the graph window and pressing the middle mouse button. (You can also plot a signal simply by double clicking on its name in the Plot File window.)

This selected signals in the plot file appear as waveforms in the active Graph window.

After you plot the signals, you can analyze the results in various ways and determine the next step in the design process.

Determining the Next Step

After you evaluated the DC transfer results, you are ready to continue the design process by proceeding to one of the following steps:

- Calibrate DC Transfer analysis as described on page D-2.
- Verify time-domain specifications as described in Chapter 2.
- Verify small-signal frequency-domain specifications using AC analysis as described in Chapter 5.

Special Topics in Transient Analysis

Some circuits require special consideration when a transient analysis is run on them.

- **Starting Oscillators in Transient Analysis** describes methods for starting oscillators and keeping them going in simulation circuits.
- **Eliminating Spikes in Transient Analysis** is a procedure for eliminating spikes produced by simulation that are not present in the equivalent real-world system.

Starting Oscillators in Transient Analysis

Physical oscillators typically rely on the amplification of noise to start the oscillation. Because noise is not inherent in the simulator, you must use some method to “kick” an oscillator away from its sticking point. The method should allow the circuit, once kicked, to behave without the continuing influence of the kicking agent. The simulator provides the following methods for accomplishing this task:

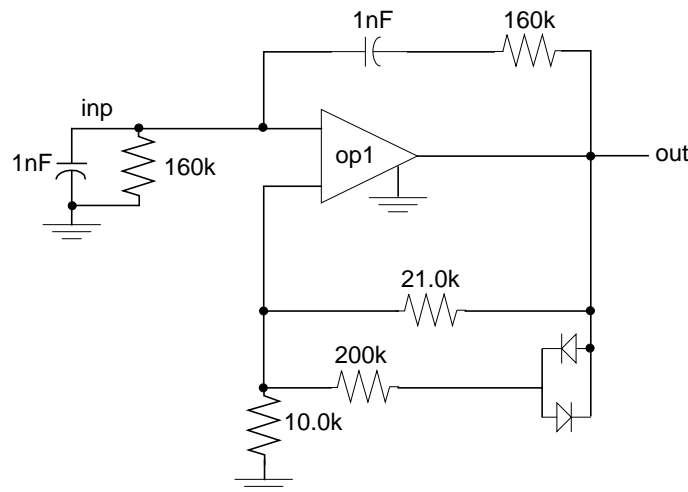
- **Starting Oscillators by Modifying the DC Initial Point**
This method creates an initial point that is not a true DC operating point (does not satisfy Kirchhoff’s laws) but provides enough energy to start the oscillation. This does not require changing the topology of the circuit.
- **Starting Oscillators by Using Current or Voltage Sources**
This method uses a current or voltage source in the circuit to provide a pulse of energy after the transient run begins. The only way to insert a source is by writing it into the netlist.
- **Keeping Oscillators Going**
This method improves the accuracy of the transient analysis to prevent

the oscillator signal from converging to zero.

Generally, these methods work by disturbing the energy storage elements, so it is important to act upon a node that can charge (or discharge) these elements. The node chosen should also have a relatively high impedance, so that the resulting current flow does not simply choose an easier path to ground.

Example Oscillator Circuit

Consider the oscillator shown in the following figure. It is a practical Wien-bridge oscillator that uses a diode mechanism to adjust the amount of negative feedback so that the overall gain of the circuit is 1.



The following steps would be performed to simulate and analyze an oscillator design such as the Wien-bridge displayed above:

1. Run the DC operating point analysis.
2. View the results of the DC analysis.

The Report Tool window would indicate that the DC value of all the nodes equaled 0.

3. Run a transient analysis with the End Time set to 3m and the Time Step set to 1u.
4. Plot the `out` signal in the Graph window.

The results would show a straight line at 0.0 volts, meaning the circuit is not oscillating.

Because the simulator results will not show the expected oscillating output signal, we must kick-start the design to start the oscillation.

Starting Oscillators by Modifying the DC Initial Point

You can edit the DC values of any initial point. Because the edited initial point no longer satisfies Kirchoff's current law, this modified file is not considered a valid operating point.

Typically, editing the initial point is used only to “kick start” oscillator circuits for transient (time-domain) analysis. Because physical oscillators rely on the amplification of noise to start and noise is not inherent in the simulator, you may have to change the DC input value by editing the initial point to provide a “kick” to start the oscillator. Once the circuit begins oscillating, the effects of altering the initial point are no longer seen. The following steps describe how to edit an initial point:

Step 1. Display the Edit Initial Point form. (Analyses > Operating Point > Edit Initial Point).

An Edit Initial Point dialog box appears.

Step 2. Define the node(s) to edit.

After the form is displayed, you can enter a space-separated list of node value pair(s) containing the node name and the new DC value. For example, if you wanted to edit the DC values of the `input` node to 1.25 volts and the `input_diff` node to 8.75 volts, you enter the following text in the Node Value List field:

```
input 1.25 input_diff 8.75
```

Step 3. Specify the initial point file to edit.

The simulator reads the DC values from the file defined in the Source Initial Point field, edits the file with the values from the Node Value List field, and stores the result in the initial point file specified in the Result Initial Point field. If the Source Initial Point field is blank, the simulator adds the Node Value List to the Zero IP file and puts the result into the Result Initial Point field.

Step 4. Execute the form.

Clicking the **Apply** button executes the `sigset` command on the values

defining the fields of the Edit Initial Point form. You can display the results of the edited initial point file, by using the **Results > Operating Point Report** pulldown menu item.

Step 5. Run the analysis with the new edited Initial Point file.

In the Input/Output tab, verify the name in the Initial Point File field.

Step 6. Check that the oscillation amplitude has stabilized within a reasonable amount of time.

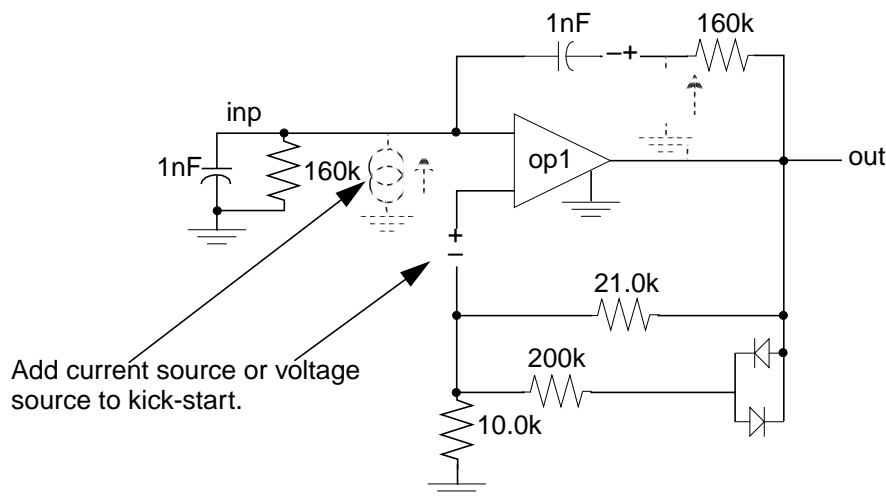
If you are “kick-starting” an oscillator, you need to complete this step.

Different “kicks” produce different transient effects, such as ringing, a growing response, or a decaying response. It is important to run a transient analysis for many cycles to ascertain the steady-state behavior of the oscillator. The higher an oscillator's Q, the longer it takes to achieve steady-state performance.

Once you have verified the performance of an oscillator, you might find that it takes more time to get steady-state performance from it than from other parts of a system. Therefore, it is usually faster to replace the oscillator with a behavioral model, such as a voltage source, when working on other parts of the system. Then you can use the oscillator model for a check of the entire system.

Starting Oscillators By Using Current and Voltage Sources

The following figure shows some possible placement locations for a voltage or current source that could be used to kick-start the circuit.



These added sources should affect the circuit only during the pulse, so current sources are tied to ground (a 0-value current source is an open circuit) and a voltage source is placed in line (a 0-value voltage source is a short). The current sources are more effective for this example because the energy storage devices are capacitors.

An example specification for a current source is as follows:

MAST syntax:

```
i.istart 0 inp = tran=(pwl=[0,0,10u,0,11u,10m,12u,0])
```

Keeping Oscillators Going

The simulator treats all signals as if they will eventually converge to a DC value. This can become a problem if you are trying to simulate an oscillator. To keep an oscillator going, the accuracy of the transient analysis has to be improved. Try each of the following steps, in sequence, to see how it affects the performance of your design:

1. Decrease the Maximum Truncation Error (`terror`) by a factor of ten.
2. If the first step in this list was only partially successful, set the Time Step (`tstep`) to be much smaller (by a factor of 100 to 1000) than the Time Step used in the previous simulation.
3. Finally, if neither of these completely stops the damping of the oscillation, decrease the Maximum Truncation Error by another factor of ten.

The above steps are a specific application of the more general procedure found in *Calibrating Transient Analysis*.

Eliminating Spikes in Transient Analysis

Transient simulation of switching circuits sometimes produces spikes in the output data that are not present in the real system. Eliminating these erroneous spikes is a multi-step process.

Step 1. Determine if the spike is real.

Many switching systems with elements such as gate turn-off (GTO) switches or diode-inductor combinations actually have large spikes in their measured response. It is worth double-checking unexpected transient analysis results against test data of prototype circuits (if available), or against the results of other analysis techniques.

Step 2. Examine the models and schematics.

- Confirm that the models in your circuit are appropriately robust and have been applied in the ways they have been designed for.

Since spikes are usually only observed in large circuits, it may be difficult to pinpoint which of the many models used have problematic construction. One starting point is to look at the models in the immediate vicinity of the node or branch with the spike. Also consider which models are excited by the switching event that triggered the spike.

- Confirm that your schematic is wired correctly.
- Check for any unintended circuit instabilities in the design.

Look for imbalanced or unstable circuits. Many simulations with spikes still have enough information to show whether or not the circuit is unstable or has properly balanced loads.

Step 3. Simulator simulation options.

The issues outlined in Steps 1 and 2 should be carefully considered before modifying the transient simulation itself. Skipping earlier steps might cause you to miss the source of the problem.

Try the following options one at a time.

1. Use the `-d flat` option to invoke Saber as follows:

```
saber -d flat designname
```

This forces Saber to use a flat, as opposed to a hierarchical, matrix. This command will cause the simulation to be slower.

2. Use the transient option `terrt all`. This is the most accurate method for calculating truncation error. This command will also cause the simulation to be slower.
3. In systems with a wide range of time constants, try the `terr3` option.
4. Use `ord 1`. This limits the integration algorithm to the first order. It is less accurate than the default of allowing both first and second

order; however, it will round off some short spikes at switching edges.

In general, truncation error and sample point density are critical parameters in determining the accuracy of transient analysis spiking.

Step 4. Other techniques worth considering.

If the results are still unsatisfactory after performing Steps 1-3, try the following debugging techniques:

- Simulate only the part of your circuit that is causing the spikes. This will allow you to concentrate on the few models that remain.
- Run separate simulations on your low-voltage circuits and high-voltage circuits. Many power supplies, for example, have a low-voltage control section that can be separated from the power section.
- Use the Monitor Progress option to produce a detailed analysis transcript.
- Try different switch models.
- Increase the accuracy of your circuit by adding non-idealities like capacitance, resistance, or inductance that have not yet been accounted for.

Chapter 7: *Special Topics in Transient Analysis*

Checking the Frequency Response-Process Overview

The following topics describe how to verify the specifications of the design by using small signal frequency analysis.

1. Introduction to Small-Signal Frequency Analyses
2. Determining the Circuit Bias Point
3. Executing AC Analysis
4. Viewing the Small Signal Analysis Results
5. Measuring AC Analysis Results
6. Determining the Next Step
7. Transforming Frequency-Domain Signals to the Time Domain

Introduction to Small-Signal Frequency Analyses

Small Signal Analyses characterize non-linear systems in the frequency domain by applying a small sinusoidal signal to the input that keeps the system running in the linear region of operation around a previously-calculated stable DC bias point.

You perform AC analysis to determine the frequency response of the system. In electrical systems, this analysis produces results similar to a Bode Plot, showing how the system responds over a user-defined frequency range at a specific bias point. Because the input signal used during AC analysis is, by definition, very small to assume linearity around the bias point, all signals are usually normalized to a unit value. For example, if the AC input magnitude of the driving source is set to '1' in the model, then the value of each remaining node in the system is equal to the gain from the input node to that specific node.

If your design is either a digital circuit or a switching circuit, it does not have a linear region of operation or a single stable operating point, making small-signal frequency analyses impossible. *Overview of Generating a Frequency Response for a Large Signal Design* describes how to use the MX-Scan template with the Analog Model Synthesis analysis tool on a large signal switching circuit.

To verify that your design meets the frequency response specification, you execute an AC analysis, view the results in Scope, and evaluate the results to determine the next step. Based on this decision, you can either make the appropriate adjustments to meet specifications and re-run the analysis or continue to the next step in your design analysis process. The following sections describe the steps in this process.

Determining the Circuit Bias Point

Because all small-signal analyses examine the design at a specific operating point, you must find the operating point of the design prior to running the analysis, using one of the following methods:

- In the AC Analysis form, specify **Yes** in the **Run DC analysis First?** field. This selection informs the simulator to execute DC analysis to find the operating point and then run the AC analysis to calculate the frequency sweep using the previously calculated operating point. The AC form is discussed in the next step.
- Run the DC analysis separately by selecting the **Analysis > Operating Point > Operating Point** pulldown menu item. In most cases, the simulator will find the correct operating point using the default values in the DC Operating Point form.

Executing AC Analysis

To perform an AC analysis, you specify the frequency range to analyze, verify the values of optional small signal fields, and then execute the command as described in the following steps:

Step 1. Open the AC Analysis Form. (Analysis > Frequency > Small Signal AC...).

Step 2. Specify the frequencies to sweep.

In order to execute a small signal analysis, you must define at least one frequency point by using the Start Frequency field, which results in the simulator calculating the frequency response at a single data point. If you want to perform the analysis over a frequency range, you specify the range by using the Start Frequency and End Frequency fields. All frequency values must be positive real numbers.

Step 3. Specify the frequency data point values.

You can control the position of the calculated data points in the defined frequency range by using the following two fields:

- Number of Points defines the number of data points calculated between the frequency range defined by the Start Frequency and End Frequency fields.
- Increment Type defines the spacing between data points, either linear (equal spacing) or the default value, logarithmic.

Step 4. Specify the Operating Point to use.

Because small-signal analyses require an operating point that the simulator uses as the bias point to apply the small sinusoidal signal, you must specify an initial point file in the Initial Point File field.

- If you have already determined a valid operating point for the small signal analysis, verify that the simulator will use the correct file by examining the Initial Point File field in the Input/Output tab. By default, the simulator uses the default output of DC analysis, *dc*, as the initial point file.
- If you have not determined a valid operating point, specify *Yes* in the Run DC analysis First? field. This selection informs the simulator to execute DC analysis to find the operating point and then to run the small signal analysis by using the resulting operating point.

You should set the Sample Point Density setting to a value equal to or larger than the value used during the generation of the initial point file.

Step 5. Specify which analysis waveforms to create.

The AC Analysis form provides the following two fields to specify how waveform data is to be saved for plotting and analysis:

- The Plot File field (Input/Output tab) specifies the name of the plot file, which contains simulation results for the signals defined by the Signal List field. The size of the plot file is a function of the number of data points and the number of signals in the Signal List. By default, the simulator creates plot file named “ac” for each transient run. If you do not want the simulator to create a plot file, you fill in the field with a “_”.
- The Signal List field (Input/Output tab) specifies the signals for which simulation results are to be saved for plotting within Scope. By default, the signal list includes only the signals at the top level of the design hierarchy. If you need to view other signals within the design hierarchy, you must add the signal names in the Signal List field.

The following table compares examples of MAST and VHDL Signal List syntax. You can include a space-separated list of multiple Signal List definitions within the same Signal List field.

MAST Signal List Definition Examples

<code>/*/*.*</code>	Include all components at the top level.
<code>/.../*.*</code>	Include all instances at, and below the root level.
<code>./</code>	Include all signals in the current instance.
<code>.../</code>	Include all signals in instances below the current instance.
<code>/.../pll.*</code>	Include all signals in all instances of any “pll” component.
<code>pll.u12/</code>	Include all signals in the <code>pll.u12</code> instance.
<code>sig1 sig2</code>	Include each signal listed.

If you are unsure about which signals to specify, you can use the default setting (all signals in the root of the design). If you need more signals later, you can do one of the following:

- For short simulation times, re-run the transient analysis with additional signals defined in the signal list.
- For long simulation run times, extract additional signals from the Data file using the **Extract > Add Signals to Plot File** menu item.

Step 6. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms. See the topic titled Viewing the Small Signal Analysis Results for additional information.

Step 7. Specify whether to save raw simulation results.

Data Files contain simulation results of each data point in the simulation. In analysis runs with a large number of data points, data files can become quite large. You may want to consider saving disk space by not creating a data file. Because it is significantly faster to extract plot information from a data file than to re-run the analysis (especially for long simulation runs), you should carefully consider the ramifications before deciding not to create a data file.

You specify the data file name in the `Data File` field under the Input/Output tab. Typically, these files have identical names for a given analysis run. If you do not want the simulator to create a data file, fill in the field with a “_”. By default, the simulator creates data files named `ac` for each AC analysis run.

For more information on managing data and plot files, refer to Appendix A: *Files Used by DesignerHDL Tools*.

Step 8. Execute the AC analysis.

Using the default AC analysis values, the simulator applies a small sinusoidal signal to the input at a bias point determined by the Initial Point File and calculates the frequency response over the range defined by Start Frequency and End Frequency fields. The simulator uses the Number of Points and Increment Type fields to determine the number and values of the calculated frequency points.

The simulator stores the results for each system variable in the Data file named `ac`. The simulator also stores all signals defined in the Signal List in a Plot File named `ac`, which can be viewed in Scope. For reference information on AC analysis, refer to the Help Online system.

After the simulator completes the AC analysis run, you use Scope to view the simulation results for the nodes defined in the Signal List.

Viewing the Small Signal Analysis Results

You use the Signal Manager within Scope to manage and display signals from the various plot files that you create during the analysis process. Overall, the procedure for viewing waveforms from an AC analysis, is as follows:

- Add the plot file to the Signal Manager's plot file list.
- Open the plot file.
- Select signals to plot.
- Plot the selected signals in a graph.

You can do this step-by-step after an analysis, or you can set up Guide to perform all or part of this process automatically.

The following procedure explains in more detail how to plot waveforms after an analysis:

Step 1. Add the plot file to the Signal Manager and open it.

Guide does this automatically if you specify one of the following under the Plot After Analysis menu field (Basic tab) before starting the analysis:

- **Yes - Open Only**
- **Yes - Append Plots**
- **Yes - Replace Plots**

In all of the above cases, Guide invokes Scope (if necessary) and opens the Signal Manager and appropriate plot file. (See the descriptions of the Plot Action options below for additional information.)

However, if you specify **No** in the Plot After Analysis menu field, Guide does not open the Signal Manager. You can open the last created plot file either by selecting the **Results > View Plot Files in Scope** pulldown menu item. (You can also click on the **Last Plotfile** icon in the Guide icon bar.) This activates the View Plotfiles dialog box. Under the Plot File pulldown menu field, select one of the following:

- **Last** selects the most recent plot file generated by the analysis. This is the default.

- **Plot File Names** allows you to specify a list of plotfiles (separated by spaces) in the Plot File field. You can use a browser to select the plot files by clicking on the **Browse** button.

Under the Plot Action field, select one of the following actions:

- **Open Only** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
- **Append Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, adding the new waveforms in addition to the corresponding waveforms from the previous analysis.
- **Replace Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, replacing the corresponding waveforms from the previous analysis.

The Append and Replace options are convenient to use during design optimization, when repeatedly running analyses to see the effects of altering design parameters.

After making the settings described above, you open the Signal Manager by clicking on the **OK** button or the **Apply** button. The Signal Manager dialog box appears, along with a Plot File window for the plot file you specified.

If you click on the **OK** button, the View Plotfiles dialog box closes. If you click on the **Apply** button, the dialog box remains open. Click on the **Close** button to close the dialog box.

Step 2. Select the signals to plot.

Assuming that you did not select the Append or Replace Plot Action options in the preceding step, you need to select the signals to be plotted. In the Plot File window, select each signal by clicking on it with the left mouse button. Unselect a signal by clicking on it again (or use the **Deselect** button or Signal menu).

Step 3. Plot the selected signal(s).

You can plot the selected signals by clicking on the **Plot** button or by moving the cursor into the graph window and pressing the middle mouse button. (You can also plot a signal simply by double clicking on its name in the Plot File window.)

The selected signals in the plot file appear as waveforms in the active Graph window.

After you plot the signals, you can analyze the results in various ways and determine the next step in the design process.

For complete information on manipulating graphs refer to Scope Graph Window Operation.

Measuring AC Analysis Results

Scope provides numerous pre-defined automated measurements to dramatically decrease the time it takes to extract frequency-domain specification data from plotted frequency sweep waveforms. If you are already familiar with the Measurement tool, you can skip to the topic titled Determining the Next Step.

To perform a measurement on a waveform within Scope, you follow these steps:

Step 1. Display the Measurement Tool.

Within the Scope user interface, you can open the Measurement Tool by either selecting the **Tools > Measurement** pulldown menu item or by clicking on the Measurement icon in the Tool icon bar at the bottom of the Scope user interface.

Step 2. Select the appropriate measurement.

After you display the this tool, you can select which measurement you want to perform. In the Measurement field, you select the measurement by clicking on the downward arrow and selecting the measurement from within the various measurement categories.

The following table lists the measures accessible in the frequency category.

Measurement	Definition
Lowpass	Displays the corner frequency of a waveform with a highpass shape
Highpass	Displays the corner frequency of a waveform with a highpass shape
Bandpass	Displays the bandwidth, the low, high, or center frequency, or the level at which the measurement is made for a bandpass-shaped waveform
Stopband	Displays the stopband, the low, high, or center frequency, or the level at which the measurement is made for a stopband-shaped waveform
Gain Margin	Displays the dB gain margin of a complex waveform
Phase Margin	Displays the phase margin of a complex waveform in degrees or radians
Slope	Displays the slope (optionally as a per-octave or per-decade value) of a waveform
Magnitude	Displays the magnitude of a point on a waveform.
dB	Displays the dB value on a point on a waveform.
Phase	Displays the phase value on a point on a waveform.
Real	Displays the real value of a point on a waveform.
Imaginary	Displays the imaginary value of a point on a waveform.
Nyquist Plot Frequency	Displays the frequency at a point on a Nyquist (or Nichols) plot.

Step 3. Select the signal to measure.

You can specify the signal to measure in the Signal field. You can specify the signal by using either of the following methods:

- Click on the downward arrow, and select a signal from the resulting list.
- Select the signal in the current active graph.

Regardless of the method you chose, you must specify a signal prior to executing the measurement.

Step 4. Set the data range to measure.

You can control the data range that is used to perform the measurement in the Apply Measurement to: section of the Measurement dialog box. You must specify one of the following data ranges:

- Entire Waveform calculates the measurement on the entire waveform, regardless of what is viewed in active graph.
- Visible X and Y range only calculates the measurement for the X and Y range visible on the current active graph.

Step 5. Perform the measurement.

You perform a measurement by clicking on the **Apply** button. This procedure performs the measurement on the specified signal and adds the corresponding information to the active graph.

Step 6. Manipulate measurement information.

During the course of data analysis within Scope, you may want to manipulate which measurements can be viewed. This task can be accomplished by using the Measure Results form. You can open this form either by selecting the **Graph > Measure Results...** pulldown menu item or by double-clicking on measurement text within an active graph.

After you display this form, you can manipulate measurement information by using the following buttons:

- **Delete Measurement** deletes the selected measurement from the active graph.
- **Delete All** deletes all measure text/graphics from the active graph.
- **Show All Values** shows all measurements listed in the left scroll list in the graph.

- **Hide All Values** retains but does not display the measurements listed in the left scroll list.
- **Show/hide status bubbles** allows you to change the visibility of a measurement on the active graph on a per measurement basis.

Determining the Next Step

Using the signal viewing and measurement capabilities of Scope, you determine whether your design meets the pre-determined specifications. If it meets specifications, then you can proceed to one of the following options:

- Verify the functionality of the design by using another small-signal analysis.
- Transform the AC results based on the linearized small signal model into a time-domain waveform using IFFT in the next subsection.
- Tune the design parameter by using statistical, parametric, or stress analysis described in Chapter 8 through Chapter 10, respectively.

If the design does not meet your specifications, you can take corrective action by using either of the following methods:

- Alter a design/instance parameter(s)

If the design change does not require you to modify the connectivity, you can modify the design parameters by using the Alter form available through the **Edit > List Alter...** menu item.

- Edit the schematic and re-netlisting.

If the design change requires you to change in connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design, (**Design > Netlist *design_name***), and then re-load the design in the simulator (**Design > Simulate *design_name***).

Transforming Frequency-Domain Signals to the Time-Domain (ifft)

The inverse Fast Fourier Transfer (IFFT) is a post processing command that transforms frequency sweep data from AC or FFT analyses into time-domain waveforms. To execute a IFFT analysis, you follow these steps:

Step 1. Display the IFFT dialog box. (Analyses > Fourier > IFFT...).

Step 2. Specify the signals to transform.

In order to execute an IFFT analysis, you must specify the following information in the Input/Output tab:

- Specify the signal names to transform.

You define the signal to transform in the Signals to Transform field. You can either type the signal names in the field or use the pre-defined functions displayed by pressing the **Select** button.

You must enter the names of the signals to transform exactly as they have been specified in the transient Plot file. You cannot use wildcards in this field. If you do not specify any signal names, all applicable signals from the transient Plot file will be transformed.

- Specify the source plot file name.

The simulator uses the plot file from a previous frequency domain analysis run as the source for the IFFT analysis. Because there is no default, you must specify the value in the Input Plot File field.

Step 3. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 4. Verify the input data points.

Although optional, the following fields provide information to control the number of input data points used in the analysis:

- Verify the number of data points

The Number of Points field specifies the number of data points used in the transform. This value must be a power of 2, such as 256, 512 or 1024.

- Define a method to extend the frequency back to 0Hz

In order to generate the time-domain waveforms, the input frequency spectrum data must begin at 0Hz. If the input spectrum data was created by an AC analysis, you must use one of the following methods to extrapolate the frequency data back to 0 Hz:

- zero defines the frequency component at 0Hz as 0.
- constant defines the frequency component at 0Hz as equal to the value of the first frequency point in the input plot file.
- linear defines the frequency component at 0Hz by using the value of the first frequency point and the slope at that data point.

Step 5. Execute the analysis.

By default, the simulator calculates the time domain waveforms for each signal in the Signal List field by using the IFFT transform. The results are stored in the plot file named `ifft`. All default values can be modified within the IFFT analysis form. For reference information on this command, refer to the online help system.

Step 6. Plot the results.

After the analysis run completes, you can view results within Scope. The following procedure outlines the process of viewing and manipulating waveform data within Scope. See the topic titled Viewing the Small Signal Analysis Results for more detailed information.

NOTE

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager in Scope (**Results > View Plotfiles in Scope**).
2. Select the signal(s) you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.

Chapter 8: *Checking the Frequency Response-Process Overview*

4. Use the waveform manipulation (e.g. pan/zoom) and measurement capabilities within Scope to analyze the data.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification.

Calibrating Analyses

The following topics describe how to calibrate the various analyses within the simulator:

- Calibration Overview
- Calibrating DC Analysis
- Calibrating DC Transfer Analysis
- Calibrating Transient Analysis
- Calibrating Small Signal Analyses
- Calibrating Fourier, FFT, and IFFT Analyses

Calibration Overview

Within the simulator, the basic purpose of calibration is to verify that the accuracy setting used in the simulation adequately depicts the behavior of the design. For example, if the sample point density is set too low during a transient analysis run, the simulator may miss a critical spike in the output waveform.

In circuit simulation, evaluating nonlinear functions is computationally expensive. In general, iterative methods are used to find an approximate solution to the original problem, or an exact solution of a problem that is close to the original problem. The simulator uses the second approach by forming a piecewise linear model using sampled nonlinear functions. The `density` variable controls the closeness of the piecewise-linear model to the original problem, i.e. the nonlinear functions specified by the models.

The basic process for all calibrations is as follows:

1. Set the calibration variables to the minimum acceptable values.
2. Run the analysis.

3. Tighten the value of the calibration variables.
4. Re-run the analysis.
5. Compare the results of the previous analysis runs.

With the exception of DC analysis, all comparisons are accomplished by viewing the results in Scope and verifying that the plots from the calibration runs are nearly identical. Determining acceptable results is subjectively based on your needs and experience. Because the purpose of calibration is to provide the most accurate results in the least simulation time, you should keep this in mind when calibrating analysis results.

Calibrating DC Analysis

Although not required, DC calibration allows you to increase the accuracy of the initial DC results in order to verify that the DC solution is sufficiently accurate. Because most analyses use the Initial Point generated by DC analysis, it is very important that the values in this file are accurate. The recommended practice is to increase accuracy by multiplying the `density` variable by a power of two (thereby increasing the number of sample points by the same factor). The following procedure demonstrates this process:

1. Execute DC analysis with the `density` variable set to 1.

```
> dc
```

This command creates an Initial Point file by using the number of sample points defined in the template.

2. Increase the density setting by a power of 2.

```
> den 2
```

3. Re-run the DC analysis using a different value for the resulting Initial Point file.

```
> dc (dcep dc2
```

4. Display the previous dc analysis runs.

```
> di dc dc2
```

If the difference in the DC results is acceptable, you should use the lowest suitable density value for all analyses simulated on the design.

If the difference in the DC results is unacceptable, you should repeat steps 2 through 3 until you find an acceptable density setting.

Calibrating DC Transfer Analysis

During DC Transfer analysis, the simulator calculates a finite number of sweep points defined by the `swbegin`, `swend`, and `swstep` variables. When Scope displays the results of this analysis, the regions between the defined sweep values are plotted as straight line segments. Therefore, calibration of the DC Transfer analysis involves verifying that these regions are well enough approximated by the linear segments.

To calibrate the DC Transfer analysis, you should decrease the sweep step value (`swstep`) until you are convinced that no non-linearities exist between the sweep sample points.

Calibrating Transient Analysis

Transient analysis calibration involves independently adjusting the following two variables:

- `Density` sets the accuracies of the non-linearities of the design model. By default, this value is set to 1, which informs the simulator to use the sample points defined within the template.
- `Truncation Error` establishes the degree of accuracy of the numerical integration algorithm.

The best strategy for calibrating the accuracy of the transient analysis is to calibrate it separately for both density and truncation error and then use the calibrated values of both variables to execute the final transient analysis. The following procedure shows this process:

1. Calibrate the accuracy of the DC analysis as described in the previous section.
2. Calibrate the accuracy of the nonlinearities in the transient analysis by following these steps:

Chapter 9: *Calibrating Analyses*

- a. Execute the transient analysis with the density value determined in step 1.
- b. Increase the density value by a power of 2, and re-run the DC and TR analyses again.

Because you will be comparing multiple transient runs, you must assign different values to the Plot File for each transient run to avoid overriding previous plot data.

- c. Plot and compare the results of the last two transient runs in Scope.

If the plotted results are within an acceptable variance, you can proceed to calibrate the numerical integration algorithm as described in step 3.

If the plotted results are outside of the acceptable variance, then repeat steps b and c until you obtain reasonable results.

3. Calibrate the accuracy of the numerical integration algorithm by following these steps:

- a. Set the density value to its smallest acceptable value as determined in step 2.
- b. Execute the transient analysis with the default truncation error (`terr`) value.

In oscillators, specifying an excessively-large truncation error can result in an error in the simulated frequency of oscillation. In some cases, this can even result in no oscillation.

- c. Decrease the Truncation error by a factor of 10, and run the transient analysis again.
- d. Compare the results by plotting the last two transient runs.

If the two runs are with acceptable variance, then transient analysis calibration is completed.

If the two runs are outside of acceptable variance, then repeat the steps c and d.

Calibrating Small Signal Analyses

To calibrate the accuracy of the small signal analyses (ac, noise, distortion, two-port), you must adjust the following two variables that are independent of each other:

- `Density` controls the “granularity” of input.
- `Number of Points` specifies the number of frequency values at which the analysis is to be run. Thus, this variable controls the granularity of the output. When you set this variable, you must check that the granularity is small enough that the analysis does not “step over”, and thereby not reveal, any parts of a frequency-response graph that reflect radical behavior, such as a spike. Increasing number points makes the frequency-response graphs smoother.

As with all analyses, you calibrate the accuracy by making comparisons. In calibrating the accuracy of the small signal analyses, you compare the frequency-response waveforms for any nodes of interest during successive runs of the analysis. You want to see similarities in the graphs. The differences that are significant depend entirely upon your situation and its requirements. Differences that might matter are phase shifts, any differences in the shape of the curves, and any differences in magnitude.

A density value that is sufficient for any of the dc- and time-domain analyses (dc, dc transfer, and transient) is not necessarily large enough to be sufficient for the small-signal frequency domain analyses because the dc- and time-domain analyses use function values, while the ac analysis uses values of function derivatives.

You must ensure that the density value used on the small signal analysis is at least as large as the density value of the DC command. It is good practice to use the same value in both cases. If necessary, re-run the DC analysis with the density required by the small-signal analysis. You can save cpu time by using the results of the previous DC analysis as the starting point. You can then use the new initial point file as input to the small signal analysis.

A strategy for calibrating the accuracy of the small signal analyses is to set it first for density and then for number points. The following steps show one method to complete this task:

1. Run the small signal analysis with the same density value as that used in the final run of the DC analysis.
2. Increase the value of density (by multiplying it times a power of 2), and run the dc and ac analyses again, using the new density value.
3. Compare results of the last two small signal analysis runs.

4. If the comparison shows unacceptable differences, return to step 2.
5. Run the small signal analysis with the default value of (100) for the number of points.
6. Double the number points value, and run the small signal analysis again.
7. Compare the results of the last two small signal analysis runs
8. If the comparison shows unacceptable differences, return to step 6. If the comparison shows acceptable differences, the design is calibrated from this small signal analysis.

Calibrating Fourier, FFT, and IFFT Analyses

The `fourier`, `fft`, and `ifft` commands are post-processing analyses: they operate on the results of prior analyses. Typically, the `fourier` or `fft` commands are applied to the results of a transient analysis. Consequently, calibrating the Fourier commands involves both a transient analysis and a `fourier` or `fft` analysis. The transient and `fourier` (or `fft`) pair is performed repeatedly, each time modifying arguments to the `transient` command and examining the `fourier` (or `fft`) results. The `DENSITY` value should be gradually increased and the `TERROR` value gradually decreased until there is little change in the values of the `fourier` (or `fft`) output with successive runs. Note that even small changes in the transient waveform can result in large changes in the `fourier` (or `fft`) output.

When using the `fourier` command, it is important to note that the value of the fundamental frequency, if specified, must be correct.

In addition to being affected by the *accuracy* of the input data, the results of the `fft` command may also be affected by the *number and spacing* of the input data. In order to obtain the most accurate `fft` results, the input data (such as from a `transient` command) should be sampled so that the resulting plot file contains exactly the samples needed by the `fft` command. For example, if a 512-point FFT is to be calculated on a transient plot file, for best accuracy, that plot file should be sampled to contain 1025 linearly-spaced points (1024 intervals are used in calculating the 512-point FFT). This sampling may be specified either on the `transient` command, or on a subsequent `extract` command, through the use of the `XSampling` variable. Sampling *is not required* in order to perform the `fft` command, but slightly more accurate `fft` results are possible with its use.

The following example demonstrates the use of sampling with the `tr` command, followed by an `fft` command (which by default takes its input from the `tr` plot file).

```
tr (te 3m, ts .1u, xs from 0 to 3m lin 1025
fft (np 512
```

The next example produces the same results as the previous example, but shows the sampling being done by extracting from the results of the `tr` command.

```
tr (te 3m, ts .1u ex (df tr, pf extr, xs from 0 to 3m lin
1025 fft (np 512, pfin extr, pfout fft_extr
```

As with the `fft` command, sampling can affect the accuracy of the `ifft` command. Best results are obtained when the `ifft` input plot file contains the linearly-spaced samples needed by the `ifft` algorithm. When the input plot file is the output of an `ac` analysis, some points will be extrapolated by the `ifft` command (so that the frequency spectrum begins at zero, as described above). In this case, it is still desirable for the `ac` output plot file to contain approximately the number of points needed by the `ifft`. Such a plot file may be created by specifying the appropriate value for the `NPoint` variable on the `ac` command. In addition, best results are obtained by performing the `ac` analysis with the `FBegin` value as close to zero as possible. If an `ifft` command is applied to the output of a previous `fft` command (perhaps modified through post-processing), in order to obtain results comparable to the original `fft` input, the `ifft` should be performed with the same number of points as the original `fft`, specified by `NPoints`, and the value of `NPoints` should be a power of 2.

The following example demonstrates the use of the `ifft` command, in conjunction with the `fft` and `ac` commands.

```
fft (np 512
ifft (pfin fft, pfout ifft, np 512
ac (fb 10, fe 100k, np 512, inc lin
ifft (pfin ac, pfout ifft, np 512
```

Although the time-domain input to the `fft` command requires $2n+1$ points, where n is the size of the FFT to be computed, the frequency-domain input to the `ifft` command requires the same number of points as is to be computed by the inverse-FFT. This occurs because the `ifft` command replicates its input data in such a fashion as to guarantee that the resulting time-domain data is real-valued.

Chapter 9: *Calibrating Analyses*

Using Two-Port, Noise and Distortion Analyses

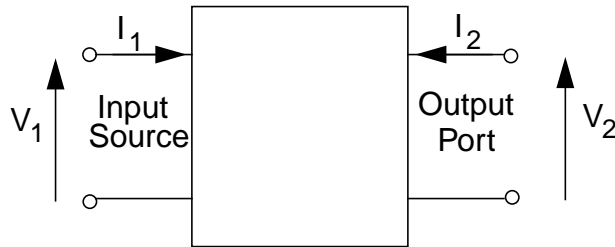
After you determine the operating point, you can verify that the frequency-domain aspects of the design meet specifications. This chapter describes how to verify the specifications of the design by using small signal frequency analysis.

Small Signal Analyses characterize non-linear systems in the frequency domain by applying a small sinusoidal signal to the input that keeps the system running in the linear region of operation around a previously calculated DC bias point. Using the following list of small signal analyses available within the simulator, you can select the appropriate analysis based on the specification that you want to verify:

- "Determining Small-Signal Transfer Function - Process Steps" on page 10-2 describes how to determine the small-signal transfer function of a design.
- "Checking Noise Specifications" on page 10-5 describes how to determine the noise contributions of the various parts in the design.
- "Checking Distortion Specifications" on page 10-9 describes how to determine distortion effects between devices in the design.

Two-Port Analysis Overview

Two-Port Analysis is a small signal computation that determines the relationship between voltages and currents at the input and output of a design as a function of frequency. The following figure shows a block diagram for the electrical paradigm of the two port analysis.



Determining Small-Signal Transfer Function - Process Steps

To investigate the relationship between the input and output of a block in your design, you will execute a two-port analysis, view the results in Scope, and evaluate the results to determine the next step. The following steps describe this process.

Step 1. Open the Two Port dialog box (Analyses > Frequency > Two-Port...).

Step 2. Specify the input and output ports.

- Specify the Input Source (Input Port).

This field defines an independent input source. In electrical systems, it may be either a voltage or current source, for example, `v(v.1)` or `i(i.supply)`. (These sources may also be expressed as `v.1` and `i.supply`.) This node is used as the input port.

- Specify the Output Port(s).

In this field, you can specify a list of node(s) within the design to be analyzed as output ports. If the output is a node voltage, then the two port output is assumed to be between the specified node and the reference node (ground).

Step 3. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 4. Specify the Frequency Range.

In order to execute a small signal analysis, you must define at least one frequency point by using the Start Frequency field, which results in the simulator calculating the frequency response at a single data point. If you want to perform the analysis over a frequency range, you specify the range by using the Start Frequency and End Frequency fields. All frequency values must be non-negative real numbers.

Step 5. Verify the type of parameters to calculate for the list in the following table.

Function	Parameter Type
Transfer Function	zin, zout, gain
Short-circuit admittance	Y-parameter
Open-circuit impedance	Z-parameters
Hybrid	H-parameters
Reverse Hybrid	G-parameters
Chain ABCD	A-parameters
Reverse Chain ABCD	B-parameters
Scattering	S-parameters
Transfer Scattering	T-parameters

For information on how these parameters are calculated, refer to the Two-port analysis topic in the SaberBook online system.

Step 6. Verify the optional small signal analysis field values.

Although optional, you should verify the contents of the fields in the Input/Output tab. This step helps prevent accidentally overwriting previously created plot and data files when you execute the analysis repeatedly.

Step 7. Execute the Analysis.

After you have completed the necessary forms, you can execute the analysis by clicking the **OK** or the **Apply** button. By default, the simulator calculates the specified parameter and creates a waveform of the signal on the output port in a Plot File named t_p .

Step 8. View the Results in Scope.

After the analysis run completes, you can view results within Scope. The following procedure outlines the process of viewing and manipulating waveform data within Scope.

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager (**Results > View Plotfiles**).
2. Select the signals you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification.

Noise Analysis Overview

Noise consists of small current and voltage fluctuations and presides in every electrical circuit. Noise analysis is a small-signal analysis that calculates the noise spectrum, to the extent that it can be modeled, from the following sources:

- *Thermal noise* is generated by the random movement of electrons in resistors and resistive material. This source of white noise is directly proportional to the temperature of the device.
- *Shot noise* is generated by the random movement of electrons crossing a potential barrier, such as a diode or the pn junction in a transistor. Shot Noise is always associated with direct-current flow and is independent of temperature.

- *Flicker noise* is caused by impurities or irregularities in the surface material of semiconductors.

The equations for modeling these three types of noise generators are modeled in all templates supplied by Analogy. You can control the noise generators within each model by changing the properties values on the symbol. For complete information on how noise effects are modeled in MAST templates, refer to *Model Fundamentals* manual.

Another source for issues related to noise analysis can be found in the book titled *Analysis and Design of Analog Integrated Circuits*, Third Edition, Paul R. Grey and Robert G. Meyer, Chapter 11 pp.715-778.

Checking Noise Specifications

To investigate the noise contributions of various parts within your design, you execute a noise analysis, view the results in Scope, and evaluate the results to determine the next step. Based on this decision, you can either make the appropriate design adjustments to meet specifications and re-run the analysis or continue to the next step in your design analysis process.

NOTE

This analysis requires a Spectral simulation license.

The following steps describe the process of checking the noise specifications:

Step 1. Display the noise analysis form (Analyses > Frequency > Noise).

Step 2. Specify frequency range to analyze.

In order to execute a small signal analysis, you must define at least one frequency point by using the Start Frequency field. This action results in the simulator calculating the noise spectrum at a single data point. To perform the analysis over a frequency range, you specify the range by using the Start Frequency and End Frequency fields. By default, the end frequency is equal to the start frequency. All frequency values must be positive real numbers.

Step 3. List of nodes at which to calculate noise.

Using the Output Signal List field, you can specify the nodes at which you want to examine the noise contribution. Although the syntax for the values in this list is identical to Signal List used in the AC and

transient analysis, this field can include only system variables, such as currents through voltage sources and node voltages.

After you specify values for these required fields, you should verify the contents of the remaining fields in the noise analysis form.

Step 4. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 5. Verify Noise Contributors.

You use the Noise Contributors field to control your ability to view where the noise in your design is originating. By default, the simulator saves the noise generated by each part in the root of the design in noise analysis calculations.

Step 6. Optionally, turn noise sources within your design off.

If you want all of the parts in your design to be included in the noise analysis, no action is required. However, if you would like to eliminate a particular part as a noise source, you can set the `nons` symbol property of that instance to 1. By default, `nons` is set to 0, which includes the instance in the noise analysis.

If you do change `nons`, you will need to re-netlist your design before running the noise analysis.

This field uses the same syntax as the Signal List field. The following table shows some examples of the Noise Contributors field:

Example	Gives noise totals for:
/ /*.*	all noise generators and component instances in root of the design (default)
/.../*.*	all noise generators in the design, by component
/.../*.*/*	all noise generators in the design, by noise source

Example	Gives noise totals for:
/opamp.*	all noise generators from opamps referenced in the root of the design
q.* /nsf -OR- nsf(q.*)	flicker noise generators for each bipolar transistors in the current block
q.1/* -OR- *(q.1)	all noise generators in the q.1 instance

Step 7. Verify the contents of the optional noise analysis fields.

Although optional, you should verify the contents of the fields in the Input/Output tab. This step helps prevent accidentally overwriting previously created plot and data files when you execute the analysis repeatedly.

Step 8. Execute the noise analysis.

By default, the simulator calculates the noise at the nodes listed in the Output Signal List using the sources listed in the Noise Contributors field over the range defined by Starting and Ending Frequency fields. The results for each system variable are stored on a logarithmic scale in the data file and plot file named ns. All default values can be modified within the noise analysis form.

For more information on the noise analysis command, refer to the “Noise Analysis” topic in the SaberBook online system.

Step 9. View the Noise Analysis Results.

After the noise analysis run completes, you can view results within Scope. The following procedure outlines the process of viewing and manipulating waveform data within Scope.

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager (Results > View Plotfiles).
2. Select the signals you want to view in the plot file window created in the last step.

3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.
4. Use the waveform manipulation (e.g. pan/zoom) and measurement capabilities within Scope to analyze the data.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification. The following section describes general guidelines for analyzing noise results.

Step 10. Analyze the Noise Results.

The Noise analysis can provide you with the following information:

- Part contributing the greatest amount of noise to system
- noise contribution across frequency spectrum
- amount of noise generated at specified frequency of interest

After you analyze the noise results to determine whether the design meets the specifications, you can proceed with one of two options.

If it met specification, then you can:

- Analyze the design by using another small-signal analysis described in this chapter.
- Tune the design parameter by using statistical, parameter sweep, sensitivity, or stress analysis described in Chapter 11 through Chapter 13, respectively.

If the design did not meet your specifications, you can take corrective action by using either of these methods:

- Alter design/instance parameter(s).

If the design change does not require you to modify the connectivity, you can modify the design parameters using the Alter form available through the **Edit > List/Alter...** menu item.

- Edit the schematic and re-netlist.

If the design change requires you to change in connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design (**Design > Netlist *design_name***), and then re-load the design in the simulator (**Design > Simulate *design_name***).

Distortion Analysis Overview

Distortion is the effect of unwanted spectral components at the output that are not present at the input. In linear systems, each output has the same spectral components as the superposition of all independent sources. In non-linear systems, additional spectral components appear at the outputs due to non-linearities. These additional spectral components are usually unwanted and are, therefore, called distortion products.

Distortion analysis is a small signal analysis that computes the following distortion products at specified outputs as a fraction of the AC value:

- *Harmonic distortion* determines the second (HD2) and third order (HD3) harmonic distortions.
- *Compression* indicates by how much the operating point (CMP2) or the frequency response (CMP3) change due to the non-linearities in the system, when a signal is applied to the circuit. These values are usually negative, hence, compression.
- *Desensitization* (DSE3) determines by how much the frequency response changes due to the non-linearities in the system, when signals with two different fundamental frequencies (f_1 and f_2) are applied to the circuit. Desensitization specifically reports the impact of the signal at frequency f_2 on the spectrum of f_1 .
- *Intermodulation* determines the spectral components at the beat frequencies if signals with two different fundamental frequencies (f_1 and f_2) are applied to the circuit. IM2D is the spectral component at $f_1 - f_2$, IM2S is at $f_1 + f_2$, and IM3D is the component at $2f_1 - f_2$.

Checking Distortion Specifications

Although Distortion analysis is a small-signal analysis, it explores the non-linearities that are present in the design. Therefore, the amplitude of the AC sources in the design must be selected to reflect the planned operating conditions of the system.

NOTE

This analysis requires a Spectral simulation license.

To investigate the distortion contributions of various parts within your design, you will execute the Distortion analysis, view the results within Scope and

then analyze the distortion products to determine the next step as described in the following steps:

Step 1. Select the distortion analysis form.

Within the Guide user interface, you can open the AC analysis form by selecting the **Analyses > Frequency > Distortion...** pulldown menu item.

Step 2. Specify frequency range to analyze.

In order to execute any small signal analysis, you must define at least one frequency point by using the Start Frequency field which results in the simulator calculating the distortion products at a single data point. If you want to perform the analysis over a frequency range, you specify the range by using the Start Frequency and End Frequency fields. By default, the end frequency is equal to the start frequency. All frequency values must be positive real numbers.

Step 3. Verify nodes at which to calculate distortion products.

Using the Output Signal List field, you can specify the nodes at which you want to examine the distortion factors. Although the syntax for the values is identical to Signal List used in AC and transient analysis, this field can only include system variables, such as currents through voltage sources and node voltages.

Step 4. Set up automatic waveform plotting.

The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

Step 5. Verify the values in the optional fields.

By default, distortion analysis calculates only harmonic distortion. If you want to compute intermodulation distortion, compression, or desensitization, you must specify additional fields in the Control tab of the distortion analysis form.

- Compute intermodulation distortion.

To inform the simulator to calculate this type of distortion, you must specify the values used to calculate the f_2 waveform as described by

the following equation:

$$f_2 = \text{ARATIO} * \text{MAG} * (\sin(2\pi f_1 t * \text{FRATIO} + \text{PHI}))$$

where:

- MAG** specifies the AC amplitude as defined on the part.
- PHI** specifies the phase offset as defined on the part.
- Amplitude Ratio (ARATIO)** specifies the amplitude ratio of the input signals of f1 and f2 as defined in the Amplitude Ratio field.
- Frequency Ratio (FRATIO)** specifies the frequency ratio between f1 and f2 as defined in the Frequency Ratio field.

- Compute desensitization and compression.

To inform the simulator to calculate these types of distortion products, you click on the **Yes** button in the Compute Desensitization field. The simulator uses compression only if you also specify intermodulation.

- Verify distortion contributors.

You use the Distortion Contributors field to control your ability to view which devices within the design contribute to distortion. By default, the simulator saves the distortion generated by each part in the root of the design in distortion analysis calculations. This field uses the same syntax as the Signal List field. The following table shows some examples of how to specify distortion contributors (default value is / * . *):

Example Syntax	Include Distortion Contribution of:
/ * . *	root of the design and all parts in root of the design
/ . . . / * . *	all parts in the design
/ . . . / * . * / *	all nonlinearities in all parts of the design
/ opamp . *	all opamps referenced in the root of the design
q . 1 / * -OR- * (q . 1)	all nonlinearities in the q . 1 instance

- Verify the sample point density.

Because small signal analyses linearize the models around the operating point, you should set the Sample Point Density in this analysis to a value larger than or equal to the value used when calculating the operating point.

Step 6. Verify the input and output file names.

Although optional, you should verify the contents of the fields in the Input/Output tab. This step helps prevent accidentally overwriting previously created plot and data files when you execute the analysis repeatedly.

Step 7. Execute the distortion analysis form.

By default, the simulator calculates the distortion factors by using the Volterra series approach on the first three terms of a Taylor series expansion of any non-linearity around the operating point. Distortion is calculated over the range defined by the Starting Frequency and Ending Frequency fields. The distortion product results are stored on a logarithmic scale in the data and plot file named *ds*. All default values can be modified within the distortion analysis form.

For more information on the distortion analysis fields, refer to the SaberBook online system.

Viewing the Distortion Analysis Results

After the simulator completes the distortion analysis run, you can view the results in Scope. You can view the distortion products shown in the following table:

Distortion Type	Signal Name	Define spectral components at:	Notes
Harmonic Distortion	HD2	2*f1	
	HD3	3*f1	
Intermodulation Distortion	IM2S	f1 + f2	f2 is derived form f1 based on user-defined amplitude and phase ratios
	IM2D	f1 - f2	
	IM3D	2*f1 -f2	

Distortion Type	Signal Name	Define spectral components at:	Notes
Compression	CMP2	f1-f1	Ratio of Operating Point to average DC level
	CMP3	2*f1-f1	Ratio of output signal change due to non-linearities
Desensitization	DSE3	f1 + f2 - f2	

The following procedure outlines the process of viewing and manipulating waveform data within Scope.

If you specified the Open Only automatic plotting option in the Plot After Analysis field before the analysis, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager (**Results > View Plotfiles**).
2. Select the signals you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification. The following step describes general guidelines for analyzing noise results.

Analyzing the Distortion Results

After you analyze the distortion products to determine whether the design meets the specifications, you can proceed with one of two options.

If it met specification, then continue the design analysis process by doing one of the following:

- Analyze the design by using another small-signal analysis described in this chapter
- Tune the design parameter by using statistical, parameter sweep, sensitivity, or stress analysis described in Chapter 11 through Chapter 13, respectively.

If the design did not meet your specifications, you can take corrective action by using one of the following methods:

- Alter design/instance parameter(s)

If the design change does not require you to modify the connectivity, you can modify the design parameters by using the Alter form available through the **Edit > List/Alter...** menu item.

- Edit the schematic and re-netlist.

If the design change requires you to change in connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design (**Design > Netlist *design_name***), and then re-load the design in the simulator (**Design > Simulate *design_name***).

Determining Parameter Sensitivity

After you have verified the functionality of your design using the transient and small signal analyses, you examine how varying the ideal parameter values affect the performance of the design. The simulator provides two methods to examine this important design question:

- *Parameter Sweep*, as described on page 11-1, sweeps the specified part parameter(s) through an user-defined range of values and executes the specified simulator analysis at each parameter value. You can view the results of this analysis in Scope. The resulting waveform is actually a set waveforms, one for each swept parameter value. This analysis is a standard feature of the simulator.
- *Sensitivity*, as described on page 11-6, determines the percentage change in a performance measurement based on user-defined change in a part parameter during an analysis run. The results of this analysis can be viewed in a Sensitivity Report. This analysis requires the optional InSpecs Parametric Analysis license.

Sweeping Parameters - Process Steps

To perform a parameter sweep, follow these steps:

Step 1. Display the Looping form.

Within the Guide user interface, you can open the Looping Commands form by selecting the **Analyses > Parametric > Vary...** pulldown menu item.

Step 2. Specify the parameters to sweep.

After the Looping Commands form is displayed, you can specify the parameters to sweep and the sweep values either by clicking on the **vary** button or by selecting **edit** from the arrow button next to the Vary command. This action reveals the Parameter Sweep form.

- ❑ Specify the parameter name.

You specify the name of the swept parameter in the Parameter Name field. The name must include both the parameter and the instance name.

For example, to vary the resistance value of an instance named `r.rload`, you would specify `rnom(r.rload)` in this field. By default, the simulator uses the first argument in the template as the parameter to sweep. Because `rnom` is the first argument of the resistor template (**r**), you could have specified `r.rload` in the Parameter Name field to yield the same result.

- ❑ Specify the sweep values.

Using the arrow button in the Variation Type field, you can select one of the following methods to sweep the parameter defined in the Parameter Name field:

- Step By sweeps the defined range by using a specified increment.
- Linear Steps sweeps the defined range by a specified number of equally spaced steps.
- Log Steps sweeps the defined range by a specified number of logarithmic steps.
- Set Values sweeps the parameter by using values specified in a space separated list. (e.g. 3 5 7 10 15 20)

After you entered the fields in the Parameter Sweep dialog box, you can add this vary command to the main loop by clicking on the **Accept** button. This action adds the command line syntax of the Vary command to the field at the right of the vary button.

Step 3. Optionally, add additional Vary and Monte Carlo loops.

Using the **AddLoop** menu item in the Looping Commands dialog box, you can add multiple Vary and Monte Carlo loops. “Nesting” these commands has the effect of sweeping several parameters in a controlled manner. The only restriction is that you cannot nest a Monte Carlo loop within another Monte Carlo loop.

Step 4. Add analyses inside the loop.

After you define the nesting structure of the parameter sweep by using the Vary and Monte Carlo looping commands, you can add the analyses to execute within the loop.

You can add any analysis listed in the **AddAnalysis > Within Loop(s)** pulldown menu to the inner loop. Once you select an analysis from the

pull-down menu, Guide adds a button, denoted by the command name, to the inner loop. You can specify the arguments of the analysis either by clicking on the analysis button or by selecting the **Edit** item from the arrow pull-down menu.

Because sweeping parameters usually affects the operating point of the design, the first analysis inside the loop is typically a DC analysis. Placing this analysis first allows the remaining analyses to use an accurate operating point, given the current values of the swept parameters.

Step 5. Add post-processing functions outside of the loop.

The final addition that you make to the Looping Commands dialog box is to add any post-processing functions. The simulator executes these functions after it completes the final looping command. Post-processing functions are listed under the **AddAnalysis > After Loop(s)** pull-down menu. The process of adding and editing post-processing functions after the loop is identical to the process of adding analyses within the loop.

Step 6. Verify the structure of the loop.

Before you execute the Looping Commands dialog box, you should verify that both the ordering of the Vary and Monte Carlo loops and the analyses defined within the loop are as expected.

Using the arrow button of the corresponding loop or analysis, you can manipulate the ordering of commands within this form by using any of the following functions:

- Edit displays the corresponding form for editing.
- Delete removes the command from the loop.
- Move Up moves the command up one level in the loop or analysis section.
- Move Down moves the command down one level in the loop or analysis section.

After you verify the structure of the Loop Commands dialog box, you can execute the Parameter Sweep.

Step 7. Execute the Parameter Sweep.

You can execute the form by pressing either of the following buttons:

- **Apply** executes the commands but leaves the form up.
- **OK** executes the commands and closes the form.

The innermost loop command is executed first, with the parameters of the outer loops set to their first value. Once the innermost loop terminates, the parameter of the second loop command from the inside is given the next value and the innermost loop command is re-executed in its entirety. This process is repeated until the value list of the second loop command is exhausted. Then the parameter of the next loop receives its next value, and the inner loops are repeated, as just described.

The whole process repeats until the values of the outermost loop are exhausted, at which time the execution ends. As a result, the commands within the loop have been executed with all possible value combinations for the parameters of the loop commands.

The simulator creates a waveform segment for each combination of parameters defined in the Looping Commands form. Thus, when you view the signal in Scope, it will have multiple waveform segments.

Step 8. View the Sweep Results.

After the parameter sweep completes, you can view results within Scope. The following procedure reviews the process of viewing and manipulating waveform data.

If you set up plot file viewing as a post-analysis step and specified the Open Only automatic plotting option, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager in Scope (**Results > View Plotfiles in Scope**).
2. Select the signal(s) you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph, either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.
4. Use the waveform manipulation (e.g., pan/zoom) and measurement capabilities within Scope to analyze the data.
5. Use the **Graph > Members** menu item to manipulate the various members associated with the signal.

Once you view the waveforms within Scope, you can analyze the data to determine whether the design is operating according to the specification.

Sweeping Parameters Example - Analyzing a Design over a Temperature Range

In order to simulate a design over a temperature range:

Step 1. Set up the schematic for sweeping temperature as a parameter.

- ❑ Place a **SaberInclude** symbol on the schematic.

This symbol allows you to globally set, and vary, temperature on the design.

- ❑ Define the temperature coefficients of the circuit elements.

- For resistors and capacitors, give a value to the t_c symbol property.

Some capacitors have a nonlinear temperature dependence.

For example, capacitors with an NP0 dielectric can have a temperature dependence of either plus or minus 30 ppm/degrees Celsius. This might require you to run several vary analyses, each one with the capacitor at a different t_c .

Capacitors with an X7R dielectric have a nonlinear temperature dependence that can be approximated by a quadratic equation. The template **cap** has the capability to model temperature dependence quadratically.

- Elements which do not have discrete temperature coefficients, such as inductors, can have the temperature expression added to the element value.

For inductors, the temperature dependence is often specified for the magnetic core material. For example, the template **splp** has two temperature coefficients, one linearly dependent, one nonlinearly dependent. The template **corenl** has an argument, `mat1`, that specifies a core material with temperature dependence built into the model.

- Some parts, such as transistors, have intrinsic temperature dependent behavior embedded in their model equations.

- ❑ If you want these changes to be a permanent part of your design, run the netlister. If you intend these changes to be in effect only for this analysis, no further action is required.

Step 2. Set up the Vary analysis.

- On the Parameter Sweep form, click the **Select** button next to the Parameter Name field, and choose the **Browse Design...** menu choice.
- On the Set Parameter List form, double click on the + next to Global Parameters in the Design Instance/Node List field.
- Under Global Parameters, double click on temp.
This places temp in the Parameter Name field of the Parameter Sweep form.
- Set the Variation Type field to the temperature range you want your design to be simulated over.
- Click **Accept**.
- Set up the rest of the Looping Commands form by adding the analyses and post-processing operations you want to be a part of the temperature parameter sweep.
- Click **OK** to run the Vary analysis.

Step 3. View the results.

After Vary has been run, each analysis within the loop will produce a multimember plot file. Use Scope to view the waveforms.

These plot files might be quite large. You can limit the plot file to only the signals you are interested in by inserting the pfextract command into the Vary analysis and setting it up to manipulate your plot file.

To do this you need to go back to the Looping Commands form and select **AddAnalysis > After Loop(s) > Transform Plotfile**. Fill out the Transform Plot File form and rerun the Vary analysis.

Determining Component Sensitivity

You can determine how sensitive a specified performance measure is to variations in component values or other system parameters in the design by using Sensitivity Analysis.

To determine the sensitivity of component parameters to a performance measure, follow these steps:

1. "Performing Nominal Simulations" on page 11-7
2. "Performing Sensitivity Analysis" on page 11-8

3. "Generating the Sensitivity Report" on page 11-10
4. "Evaluating the Sensitivity Report" on page 11-11

NOTE

This analysis requires a InSpecs Parametric license.

Sensitivity Analysis is used to assess the impact of changes on a major operating characteristic of a circuit, and to determine which components have the greatest effect on overall design performance. This automatic analysis changes parameters one at a time, measures resulting changes in operation, and provides a ranked ordering of parts and their effects on design features.

You can use the information generated by a Sensitivity analysis to:

- Identify the parts in the design that have the greatest impact on the performance of the circuit using a performance measure.

A performance measure is a single numeric characteristic of a circuit such as a node voltage, frequency, or rise time of a waveform. The simulator determines the performance measure value by using a measure operation on a waveform resulting from an analysis, such as transient or AC. Some examples of performance measures are *frequency*, *duty cycle*, and *rise time*. For example, the frequency measure can be used to extract the frequency of a periodic waveform that results from a transient analysis.

- Select appropriate component tolerance values for the design.

Performing Nominal Simulations

In order to verify that you are supplying the expected information to sensitivity analysis, you may want to perform the analysis that you will use in the sensitivity analysis to verify that waveform and performance measure are within the expected range.

Although you will not use this specific nominal simulation as part of the sensitivity analysis, running it now may help avoid problems later.

After you verify the functionality of the design by running some nominal simulation, you can perform sensitivity analysis.

Performing Sensitivity Analysis

To perform a sensitivity analysis, follow these steps:

Step 1. Display the Sensitivity Form.

Within the Guide user interface, you can open the Sensitivity analysis form either by selecting the **Analyses > Parametric > Sensitivity...** pulldown menu item or by clicking on the **Sensitivity** icon in the Guide icon bar.

Step 2. Specify sensitivity parameters.

After the Sensitivity Analysis form is displayed, you can specify the sensitivity controls. In the next step, you will add analyses and performance measures to the scroll-able listbox at the bottom of the form.

- Specify the parameters to Vary command

Enter the list of circuit parameters that are to be varied (perturbed) for the sensitivity calculation to the Parameter List field. This field uses the same syntax as the Signal List field uses in transient analysis. The following table provides examples of the Parameter List syntax to define additional signals. You can include a space-separated list of multiple Parameter List definitions within the same Parameter List field.

Parameter List Examples	Definition
<code>./</code>	Include all signals in the current instance.
<code>/.../</code>	Include all signals within or below the top-level design (root of the design).
<code>/.../pll.*</code>	Include all signals in all instances of any “pll” component.
<code>.../</code>	Include all signals in instances below the current instance.
<code>pll.u12/</code>	Include all signals in the <code>pll.u12</code> instance.
<code>sig1 sig2</code>	Include each signal listed, separated by spaces

For example, to evaluate the design sensitivity of the resistance value of an instance named `r.rload`, you would specify `rnom(r.rload)` in this field. By default, the simulator uses the

first argument in the template as the parameter to vary. Because `rnom` is the first argument of the resistor template (**r**), you could have specified `r.rload` in the Parameter Name field to yield the same result as `rnom(r.rload)`.

In addition to a nominal analysis, an additional analysis will be run for each parameter listed in this field. Therefore, you may want to limit the number of parameters specified in this field to minimize the simulation time required.

- Verify the percentage to increase the parameter

You specify the percentage to increase the parameter in the Perturbation field. During sensitivity analysis, each parameter in the Parameter List field will be increased by the percentage defined in this field. For example entering `0.01` in this field would vary the parameter by 1%. Sensitivity analysis compares the performance measure (defined in the next step) at the perturbed and nominal value to determine the sensitivity of the parameter.

- Specify how to generate the Sensitivity Report

The simulator saves the results of a sensitivity analysis to the file named in the Intermediate File field. To generate a Sensitivity Report, you can either use the **Results > Sensitivity Report** menu item after the sensitivity analysis completes or select **Yes** in the Report after Analysis field. In either case, the simulator reads the data in the file in the Intermediate File field to generate the Sensitivity Report.

After you specify the sensitivity controls in the upper half of the Sensitivity Analysis dialog box, you can define the analyses to run during sensitivity analysis and the measures to perform in the scrollable listbox in the bottom half of the form, as described in the next step.

Step 3. Specify the analyses and performance measures.

You can use the **AddAnalysis** menu item, in the bottom half of the form, to add analyses and performance measures to the Sensitivity Analysis form. Selecting a command from this pulldown menu adds a button (denoted by the command name) to the listbox in the bottom of the form. You can specify the arguments of the command either by clicking on the specified command button or by selecting the **Edit** item from the arrow button next to the specified command.

The commands displayed in this list will be executed, in order, $N+1$ times where N is the number of parameters listed in the Parameter List field. Because the simulator executes the commands in the order

defined in the list, you should verify that the simulator generates the necessary information prior to executing a command.

For example, because this analysis changes the parameter values, you should define a DC Analysis as the first analysis to run. You should also specify the measurement of a signal after the corresponding analysis (e.g. define the period measurement after the transient analysis).

After you define the necessary fields in the dialog box, you can execute the sensitivity analysis.

Step 4. Execute the sensitivity analysis by pressing the Apply button.

After the simulator completes the sensitivity analysis, you can generate the Sensitivity Report as described in the following section.

Generating the Sensitivity Report

You can generate a sensitivity report as part of the sensitivity analysis as described in the previous section, or you can invoke the Sensitivity Report form and generate a report directly from that form. If you already generated the Sensitivity Report, you can skip to the next section.

You can display the Sensitivity Report form by selecting the **Results > Sensitivity Report** menu item from the Guide user interface. After you display the form, you can verify the fields in the following tabs to define the information to place in the Sensitivity Report:

- In the fields in the Basic tab, you can control calculations displayed in the Sensitivity Report
- In the fields in the Columns to Include tab, you can specify the types of information that you want the simulator to specify in the Sensitivity Report

Part Type and Part Class are controlled by template symbol properties:

- `part_type` is used to identify parts by type (for example, `bjt`, `diode`, `mosfet`).
- `part_class` is used to identify parts by classification (for example, diodes could have two classifications, `hi-pwr` and `lo-pwr`).

Changing these properties is not required. These properties are strings that specify the inclusion of a device in a report based on the device type or class (sometimes called *report filtering*). A list of the default values for `part_type` and `part_class` can be found in SaberBook.

- In the fields in the Output Control tab, you can specify the layout of the Sensitivity Report.

After you verify the fields in the Sensitivity Report form, you can click on the **OK** button to execute and close the form.

Evaluating the Sensitivity Report

In the Sensitivity Report, you can examine either the Sensitivity or Bar-chart columns to determine the sensitivity of the specified parameters to a defined performance measure.

You can use the information in this report to determine the tolerance values of the parts in your design based on the conditions presented by the analyses executed within the lower portion of the Sensitivity form.

Chapter 11: *Determining Parameter Sensitivity*

Tuning Design Parameters with Statistical Analysis

After you have verified the functionality of your design using the transient and small signal analyses, you can examine how varying parameter values with Monte Carlo analysis can affect the performance of the design.

To tune design parameters by using statistical analysis, you use the following process:

1. "Specifying the Parameter Tolerance Ranges" on page 12-1
2. "Executing the Monte Carlo Analysis" on page 12-3
3. "Viewing the Monte Carlo Results" on page 12-6
4. "Determining the Next Step after a Monte Carlo Analysis" on page 12-7

NOTE

In order to use this analysis, you must obtain an InSpecs Statistical Analysis option license.

Monte Carlo analysis randomly varies the parameters, within user-defined tolerance ranges, and executes the specified Saber analysis at each parameter value.

This technique allows you to simulate the manufacturing environment. By specifying the tolerance ranges on the parts in the design, Saber can randomly vary specified parts allowing you to evaluate the variance of part values, in the production environment, affect the performance of the design.

Specifying the Parameter Tolerance Ranges

Before you can perform the Monte Carlo analysis, you must specify the tolerance ranges for the part parameters that you want to randomly vary. Saber provides the following types of distributions to specify tolerance ranges:

Chapter 12: Tuning Design Parameters with Statistical Analysis

- *uniform distribution* specifies an equal probability of occurrence across the interval.
- *normal (gaussian) distribution* specifies a bell-shaped curve. This is the most used distribution because it describes most physical events in nature.
- *piecewise linear* specifies a custom distribution curve.

For more information on the preceding methods of specifying tolerance ranges and modifications that you can make to the default curves, refer to Book 2 of the *Guide to Writing MAST Templates* manual, the.

You can specify the nominal value and tolerance ranges on a resistor, capacitor, or inductor template, using one of the following techniques:

- Specify the type of statistical distribution (such as normal) about a nominal value using the resistance, capacitance, or inductance argument of the template. For example, if you specify a distribution using the nominal argument (`rnom`) of the resistor template, **r**:

```
r.1 = rnom = normal(1k, 0.1)
```

You would then specify `rnom(r.1)` in the Parameter List field described in the Step 2 of the next section.

- Specify a value for the nominal argument and for the tolerance of the external variable provided in the resistor, capacitor, or inductor template (`r_tol`, `c_tol`, `l_tol`, respectively). For example, the parameter `r` is the internal parameter used by the resistor template that holds the final calculated value of resistance. In the following netlist example, the `r.1` instance is a 1K ohm resistor with 10% tolerance.

```
r.1 = rnom=1k, r_tol=0.1
```

Thus, to observe the variation of multiple Monte Carlo runs on a given resistor, you would specify `r` in a parameter list for the analysis, not `rnom`. The same holds true for internal parameters `cap` in the capacitor template and `lcalc` in the inductor template.

After you specify the tolerance ranges for the parts that you want to vary, you can proceed to execute the Monte Carlo analysis, as described in the next topic.

Executing the Monte Carlo Analysis

To perform a Monte Carlo analysis, follow these steps:

Step 1. Display the Looping form.

Within the Guide user interface, you can open the Looping Commands form by selecting the **Analyses > Statistical > Monte Carlo...** pulldown menu item.

Step 2. Specify the Monte Carlo parameters.

After the Looping Commands form is displayed, you can specify the parameters to sweep and the sweep values by either clicking on the **mc** button or selecting **edit** from the arrow button next to the **mc** command. This action reveals the Parameter Sweep form. The following list provides information on the fields within this form:

Specify number of runs

Specify the number of times to execute the commands in the loop body in the **Runs** field.

Specify the seed of the random number generator

When Saber is started, it initializes a seed based upon the state of the CPU clock.

The following values define the seed used during Monte Carlo analysis:

- **Current** specifies, without modification, the seed initialized when Saber is started.

This value stays the same for a given invocation of Saber, but is different at every invocation.

- **Constant** specifies a value that is hard-coded in Saber.

This value stays the same from invocation to invocation.

- **Random** specifies a value derived from the current state of the CPU clock.

This value is re-calculated each time the seed is used. If you change the seed from **random** to **current**, the last seed calculated by **random** will be used.

- `Specified` specifies the seed value as being derived from the two values (0 or a positive integer) entered in the Seed field. If you enter two integers you must separate them with a space.

You can replicate the Monte Carlo results if you specify either `Constant` or `Specified` as the seed value.

- Optionally, specify the plot file containing the parameter values

You can use the following fields to store specified part parameter values during the Monte Carlo analysis run (if you do not specify values for both fields, then Saber does not create the plot file):

- The `Parameter File` field defines the name of the plot file that contains the parameter values for each Monte Carlo run. The `Parameter File` is used when generating a graph of measured results versus parameter value.
- The `Parameter List` field defines the part parameters to store in the Plot File. If you only specify the instance name, Saber stores all part parameters in the Plot File.

For example, if you specified `r.load` in this field, all the parameters of the resistor would be stored in the Plot File. If you specified `rnom(r.load)` in this field, then `rnom` parameter values at each Monte Carlo run are stored in Plot File.

After you entered the fields in the Parameter Sweep dialog box, you can add this `mc` command to the main loop by clicking on the **Accept** button. This action adds the command line syntax of the Monte Carlo command to the field at the right of the `mc` button.

Step 3. Optionally, add additional Vary and Monte Carlo loops.

Using the **AddLoop** menu item in the Looping Commands dialog box, you can add multiple Vary and Monte Carlo loops. “Nesting” these commands has the effect of sweeping several parameters in a controlled manner. The only restriction is you can not nest a Monte Carlo loop within another Monte Carlo loop.

Step 4. Add analyses inside the loop.

After you define the nesting structure of the parameter sweep using the vary and Monte Carlo looping commands, you can add the analyses to execute within the loop. Saber executes these analyses for each combination of parameters defined by the Monte Carlo and Vary (if present) loops.

You can add any analysis listed in the **AddAnalysis > Within Loop(s)** pulldown menu to the inner loop. Once you select an analysis from the

pull-down menu, Guide adds a button, denoted by the command name, to the inner loop. You can specify the arguments of the analysis by either clicking on the analysis button or selecting the **Edit** item from the arrow pull-down menu. Refer to the corresponding sections within this manual for information of the forms for these analyses.

Because sweeping parameters usually affects the operating point of the design, the first analysis inside the loop is typically a DC analysis. Placing this analysis first allows the remaining analyses to use an accurate operating point, given the current values of the swept parameters.

Step 5. Add post-processing functions outside of the loop.

The final addition that you make to the Looping Commands dialog box is to add any post-processing functions. Saber executes these functions after it completes the final looping command. You can add any function listed in the **AddAnalysis > After Loop(s)** pull-down menu. The process of adding and editing post-processing functions after the loop is identical to the process of adding analyses within the loop.

Step 6. Verify the structure of the loop.

Before you execute the Looping Commands dialog box, you should verify that both the ordering of the vary and Monte Carlo loops and the analyses defined within the loop are as expected. Using the arrow button of the corresponding loop or analysis, you can manipulate the ordering of commands within this form using any of the following functions:

- **Edit** displays the corresponding form for editing.
- **Delete** removes the command from the loop.
- **Move Up** moves the command up one level in the loop or analysis section.
- **Move Down** moves the command down one level in the loop or analysis section.

After you verify the structure of the Loop Commands dialog box, you can execute the Monte Carlo analysis.

Step 7. Execute the Monte Carlo analysis.

You can execute the form by pressing either of the following buttons:

- **Apply** executes the commands but leaves the form up.
- **OK** executes the commands and closes the form.

The innermost loop command is executed first, with the parameters of the outer loops set to their first value. Once the innermost loop terminates, the parameter of the second loop command from the inside is given the next value and the innermost loop command is re-executed in its entirety.

This process is repeated until the value list of the second loop command is exhausted. Then the parameter of the next loop receives its next value, and the inner loops are repeated, as just described. The whole process repeats until the values of the outermost loop are exhausted, at which time the execution ends. As a result, the commands within the loop have been executed with all possible value combinations for the parameters of the loop commands.

Saber creates a waveform segment for each combination of parameters defined in the Looping Commands form. Thus, when you view the signal in Scope, it will have multiple waveform segments.

Viewing the Monte Carlo Results

The following procedure reviews the process of viewing and manipulating waveform data generated by Monte Carlo runs.

If you set up plot file viewing as a post-analysis step and specified the Open Only automatic plotting option, you can skip step 1. If you specified the Append or Replace options, for previously graphed waveforms you can skip steps 2 and 3 as well.

1. Add the plot file created by the analysis run to the Signal Manager in Scope (**Results > View Plotfiles in Scope**).
2. Select the signal(s) you want to view in the plot file window created in the last step.
3. Display the selected signals in the graph, either by pressing the **Plot** button in the plot file window or by pressing the middle mouse button in the active graph.

The signals will have multiple members (one for each specified run). The Signal Manager places a number in brackets next to the plot file name to indicate the number of members.

4. Use the waveform manipulation (e.g. pan/zoom) and measurement capabilities within Scope to analyze the data
5. Use the **Graph > Members** menu item to manipulate the various members associated with the signal.

6. For detailed information on measuring and manipulating multi-member waveform produced by Monte Carlo Analysis, refer to the Measurement tool.

To generate a graph of measured results versus a parameter value:

1. Display the Calculator (**Tools>Calculator**) in Scope.
2. Display the Output Plot File that you specified in the post-processing Batch Measure form.
3. From the Output Plot File, select the signal of discrete, measured values you want on the y-axis of your graph.
4. With the signal selected, move the mouse cursor to the X-register of the Calculator and press the middle mouse button. This signal is now loaded into the Calculator.
5. Display the Parameter File plot file.
6. Select the parameter you want on the x-axis of your graph.
7. Place the selected signal into the X-register of the Calculator.
8. Select the **Wave>f(x)** menu item from the Calculator.
9. Click on the **Graph X** button on the Calculator to display your graph.

Determining the Next Step after a Monte Carlo Analysis

After it has completed, you can view the results in Scope and use the pre-defined measurements to determine statistical information about the simulated manufacturing run, created using Monte Carlo analysis.

Create a Histogram using Saber

Alternatively, you can use Saber commands to create histograms and plots showing statistical information about the Monte Carlo analysis run. Although you can get similar information by using the Measurement Tool within Scope, these commands allow you to create separate plot files containing the statistical information directly from the analysis plot file.

Using the Histogram form (**Analyses > Statistical > Histogram**), you can inform Saber to create a new plot file containing a histogram for the specified signals (curve name) in a plot file (usually generated by a Monte Carlo analysis).

Create a statistical analysis plot file using Saber

Using the Plot File Statistics form (**Analyses > Statistical > Statistical Summary**), you can inform Saber to create a new plot file containing the following statistical information for the specified signals (curve names) in a plot file (usually generated by a Monte Carlo analysis):

- Mean informs Saber to create the mean (average) waveform for each specified signal to the plot file.
- Standard Deviation informs Saber to create a waveform defining how much the values of the specified signal(s) vary.
- Median informs Saber to create the middle (50%) waveform for each specified signal.
- Envelopes informs Saber to create the maximum and minimum value waveforms for each specified signal.
- Percentile informs Saber to create a waveform for the signals percentile values (defined in the spaced separated list). For example, “0 50 100” would create a minimum, median and maximum waveform for each specified signal.

Continue the Monte Carlo Analysis

During your evaluation of the Monte Carlo results, if you require more data points you can continue the Monte Carlo analysis by selecting the **Analysis > Continue > Monte Carlo...** pulldown menu item. This action displays the Continue Monte Carlo Analysis dialog box. Within this dialog box, you can specify the number of additional runs and change the seed value, if necessary. Clicking on the **Apply** button executes the additional Monte Carlo runs.

Determining Component Stress Levels

After you verify the functionality of your design, you can use stress analysis to determine the stress levels of parts in a design. With stress analysis you are able to produce a report listing stress ratios for parameters of all parts in the design. This report will show when a part is being used outside of its intended region of operation.

The following list defines the process that you will use to determine the component stress levels in your design:

1. "Collecting Part Data" on page 13-1
2. "Specifying Stress Ratings on Components" on page 13-2
3. "Specifying Derating Values" on page 13-3
4. "Performing a Nominal Simulation" on page 13-4
5. "Performing the Stress Analysis" on page 13-5
6. "Viewing the Stress Report" on page 13-6
7. "Analyzing the Stress Report" on page 13-7

NOTE

In order to use this analysis, you must obtain an InSpecs Stress Analysis option license.

The following sections in this chapter describe how to determine the component stress levels based on the process defined in the previous list.

Collecting Part Data

Saber uses stress measures to determine the stress level of parts in the design. A *stress measure* is an operating condition for which a rating (operating limit) can be specified, for example, power dissipation of a resistor or the junction temperature of a bipolar transistor).

Stress measures are typically defined with parameters on the individual parts. You will need to supply parts ratings and thermal resistance data for the following types of parts in your design:

- Passive components such as resistors, capacitors, and inductors
- Any generic part that doesn't have a manufacturer's part number associated with it
- Any component in the Component Library for which stress ratings have been implemented but for which values have not been provided

The values of the stress measures should not be derated. Later in the process, you will determine whether you need a Derating File containing the derating factors and create one, if necessary.

In the next step, you will add the part data to the stress measure properties on the necessary parts in the design.

Specifying Stress Ratings on Components

After you collect the necessary stress data for the parts without pre-defined stress measures, you can use one or more of the following methods to specify stress ratings on the parts in your design:

- Specify ratings globally by using the **Saber Include** symbol.

You can place a **Saber Include File** symbol at the top level of a design and set the property values on this symbol to specify a global value for all parts at that level or below this level in the schematic.

If individual parts in your design require stress ratings that differ from the global setting, you can override the global setting either by specifying the ratings on the individual parts or by altering the ratings using the **Edit > List/Alter...** menu item.

You can prevent stress ratios from being calculated on the entire design by setting the global variable `include_stress` to 0.

- Specify ratings on individual parts.

Parts from the Component Library already contain the appropriate stress rating properties.

If you are using a part from the Template Libraries:

- a. Identify which ratings you want to specify by using the Stress Arguments section of the template description, and determine what values you would like each of these ratings to have.

- b. Using the Property Editor, display the structured arguments of the `ratings` symbol property and enter values for the `ratings` arguments you want to define.

If you want to add stress measures to a user-defined template, refer to Book 2 of the *Guide to Writing MAST Templates*, Chapter 9.

You can prevent stress ratios from being calculated on a particular part by setting the `include_stress` symbol property to 0.

- Override stress ratings within Saber.

You can specify or override a value for a rating by using an **Edit > List/Alter...** menu item within SaberGuide. This method allows you to change stress measures on a specific part or on the entire design. This command is valid only for the current simulation session and does not affect the values on the schematic. In addition to changing stress rating values, this method is also useful to enable/disable stress measures for specific parts in the design.

Because you can define stress rating values in multiple locations, Saber applies the following order of precedence to determine which value to use during stress analysis:

1. A value provided as a rating in a ratings property for a part (e.g., the `pdmax_ja` rating for a resistor)
2. A value provided to a symbol property to override a global value for that part (e.g., the `r_pdmax` property on a resistor)
3. A value provided globally to all parts in the design (e.g., the `r_pdmax` property on a Saber Include File symbol)

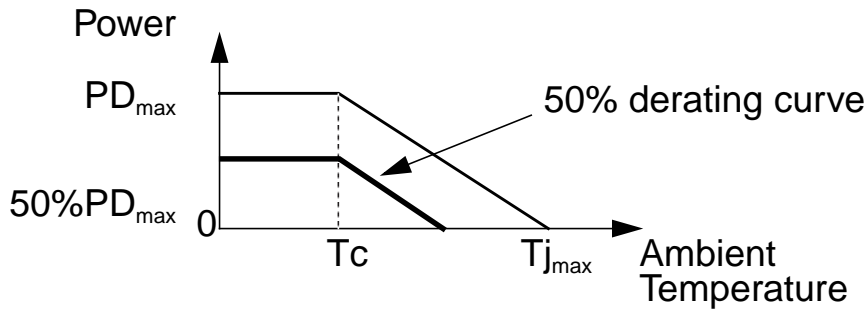
After you specify the necessary stress ratings, you can determine whether you need to create a Derating File as discussed in the following section.

Specifying Derating Values

Designers often reduce the ratings provided by the manufacturer for a part by some *derating factor*. This value allows for such factors as drift in a component value or a change in the ambient temperature that can cause the operating conditions for the part to move out of the Safe Operating Area (SOA) curve. The derating factor selected is often determined by the environment in which the design will be used (military, industrial, commercial). For more information on SOA curves, refer to page D-1.

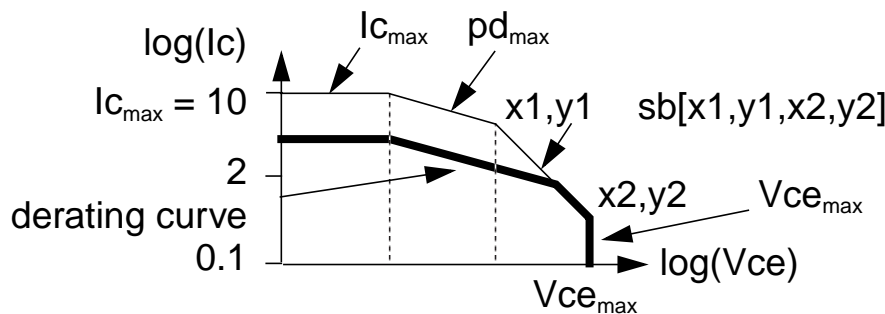
A derating factor is typically a percentage that is multiplied by the rated value defined by the stress measure. For example, the maximum power

dissipation of a resistor could be derated by 50% by using a derating factor of 0.5. The following figure shows the resulting derated curve for the resistor.



Each stress measure rating for a part can also be derated independently.

For example, the following figure shows a derating curve for a transistor in which $I_{C_{max}}$ and $p_{d_{max}}$ have been derated by using a derating factor of 0.35 and in which s_b and $V_{ce_{max}}$ have not been derated.



For a stress analysis, you can specify derating factors for parts and groups of parts in a text file called a *derating file*. The derating file provides the derating factors to be applied to the ratings defining the SOA curve for the part. You specify the name of the derating file when you run a stress analysis.

If you need to create a Derating File, refer to page A-17 for information on the syntax of this file.

Performing a Nominal Simulation

During Stress analysis, Saber calculates the stress ratios for each stress measure over the conditions of a nominal simulation. You must create one of the following types of files, by running the corresponding analysis, to serve as input to the Stress analysis:

- Initial Point File usually from a DC Analysis

- Data File from a DC Transfer Analysis
- Data File from a Transient Analysis

After you complete the nominal simulation run, you can run the stress analysis as described in the following section.

Performing the Stress Analysis

After you specify the necessary stress ratings on the parts in the design, optionally create a derating file, and perform a nominal simulation, you are ready to perform Stress Analysis. This analysis uses both the data from a previous analysis run and the part stress ratings to determine the stress ratios for each specified part in the design. To perform a stress analysis, follow these steps:

Step 1. Open the Stress Analysis Form.

Within the SaberGuide user interface, you can open the Stress form by selecting the **Analyses > Stress** pulldown menu item.

Step 2. Specify the nominal simulation to use.

Using the arrow button in the Use Input From field, you can select the analysis source of the nominal simulation data. You can also enter the file name containing the nominal simulation data in the following field. The name of this field varies depending on the type of analysis that you use in the nominal simulation

You can also restrict the amount of information to use from the DC Transfer or Transient data file by using the X Range field in the Transformations tab.

Step 3. Specify how to store the output report.

By default, you generate the Stress Report after the stress analysis is complete by using the information stored in the Output File field. This process of generating the Stress Report by using this method is described in the next section.

Alternatively, you can inform Saber to generate the Stress Report immediately after the analysis finishes by selecting **Yes** in the Report after Analysis field.

Step 4. List the stress measures for which a stress ratio is to be calculated.

You can specify which stress measures to consider when calculating stress ratios in the Stress Measure List field in the Transformations tab. This field accepts the same syntax as the Signal List field in the transient analysis form. By default, Saber calculates stress ratios for all stress measures defined in the design.

Step 5. Specify the name of the derating file to be used.

If you created a Derating File, you can specify the name of the file in the Derating File Name field in the Transformations tab. Saber uses the same search procedure to locate the derating file as it does to locate templates. Typically, you place the derating file at the same directory level as the *design.sin* netlist file.

Step 6. Execute stress analysis by clicking on the OK button.

If you have enabled the Generate Report After Analysis option by clicking on the **Report** button on the Stress Analysis form, a formatted stress report will be generated at the end of the analysis. The format used will be determined by the current settings of the Stress Report form.

Viewing the Stress Report

After Saber completes the Stress analysis, you can generate the Stress Report as described in the following steps:

Step 1. Display the Stress Report form.

You can generate a stress report as part of the stress analysis as described above, or you can invoke the Stress Report form and generate a report directly from this form. You can invoke the Stress Report form by selecting **Results > Stress Report**.

Step 2. Verify data to display in the Stress Report.

You can use the fields in the Basic tab to limit the amount of information listed in the Stress Report. Refer to the help messages at the bottom of the form for a description of each field.

Step 3. Verify the type of information to include in the report.

The Columns to Include tab contains a list of the types of information that you can include in the Stress Report. You can select the appropriate columns based on the information that you are trying to derive from the Stress Report.

Part Type and Part Class are controlled by template symbol properties:

- `part_type` is used to identify parts by type (for example, `bjt`, `diode`, `mosfet`).
- `part_class` is used to identify parts by classification (for example, diodes could have two classifications, `hi-pwr` and `lo-pwr`).

Changing these properties is not required. These properties are strings that specify the inclusion of a device in a report based on the device type or class (sometimes called *report filtering*). A list of the default values for `part_type` and `part_class` can be found in SaberBook.

Step 4. Verify page layout.

In the Output Control tab, you can control the layout of the Stress Report. Refer to the help messages at the bottom of the form for a description of each field.

Step 5. Generate the Stress Report.

You can generate the Stress Report by pressing the **OK** button.

Analyzing the Stress Report

You can examine either the Stress Ratio (%) or Bar-chart columns in the Stress Report to determine whether a part in the design was over-stressed for the conditions presented in the nominal simulation. If the Stress Ratio is over 100(%) or if the Bar-chart contains “xxxxxxx”, then the part is overstressed.

If the design did not meet your specifications, you can take corrective action by one of the following methods:

- Altering a design/instance parameter(s)

If the design change does not require you to modify the connectivity, you can modify the design parameters by using the Alter form available through the **Edit > List/Alter...** menu item.

- Edit the schematic and re-netlist.

Chapter 13: *Determining Component Stress Levels*

If the design change requires you to change in connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design (**Design > Netlist** *design_name*), and then re-load the design in Saber (**Design > Simulate** *design_name*).

Checking Linear Systems Analysis Specifications

Linear Systems Analysis characterizes systems utilizing pole-zero analysis. This allows you to fine tune your design, and keep track of your pole-zero specifications.

The following are three analyses related to Linear Systems Analysis:

- Pole-zero analysis determines the poles and zeros of a system.
- Linear Time Domain Response analysis determines the time domain response of a system using pole-zero data.
- Frequency Response analysis determines the frequency response of a system using pole-zero data.

Pole-zero analysis can be included within a Vary loop to perform a pole-zero sweep, or within a Monte Carlo loop to examine the effects of component tolerances on stability.

Determining the Poles and Zeros of a System

You perform pole-zero analysis to determine the poles, zeros and gains of a system. In electrical systems, this analysis produces results in an s domain plot, showing how the system responds at a specific bias point.

To verify that your design meets pole/zero specifications, you execute a pole/zero analysis, view the results in Scope, and evaluate the results to determine the next step. Based on this decision, you can either make the appropriate adjustments to meet specifications and re-run the analysis or continue to the next step in your design analysis process.

The following topics describe how to verify the specifications of the design by using pole-zero analysis.

1. "Determining the Circuit Operating Point" on page 14-2
2. "Executing Pole-Zero Analysis" on page 14-2

3. "Viewing Pole-Zero Analysis Results" on page 14-4
4. "Measuring Pole-Zero Analysis Results" on page 14-7
5. "Determining the Next Step" on page 14-9

Determining the Circuit Operating Point

Pole-zero analysis examines the design at a specific operating point. You must find the operating point of the design prior to running the analysis, using one of the following methods:

- In the Pole-Zero Analysis form, specify `Yes` in the Run DC analysis First? field. This selection informs Saber to execute DC analysis to find the operating point and then run the pole-zero analysis to calculate the poles and zeros using the previously calculated operating point. The Pole-Zero form is discussed in the next step.
- Run the DC analysis separately by selecting the **Analysis > Operating Point > DC Operating Point** pulldown menu item. In most cases, Saber will find the correct operating point using the default values in the DC Operating Point form. For information of running and examining the DC analysis, refer to *Appendix B* of this manual.

Executing Pole-Zero Analysis

To perform a pole-zero analysis, you specify whether poles, zeros, or both poles and zeros need to be determined, and then execute the command as described in the following steps:

Step 1. Open the Pole-Zero Analysis Form. (Analysis > Linear Systems Analysis > Pole-Zero).

Step 2. (Optional) Use the -d flat Option.

The Linear Systems Analysis tool requires the use of the `-d flat` option if the design is hierarchical or contains hierarchical templates.

To use the `-d flat` option, perform the following steps:

1. Load a design.
2. Select **Edit > Saber Settings**.
3. Select **Additional options > Specified**
4. type `-d flat`

5. Click **Save**
6. Reload your design and run a DC analysis.

Step 3. Specify whether to calculate Poles, Zeros, or Both.

In order to execute a pole-zero analysis you must specify whether you want Saber to calculate pole data, zero data, or both.

Pole data is the default setting. There is only one set of poles for the entire design.

If you select zeros data, you must also enter an Input Source, and an Output List.

The Input Source is a variable, which must be a *val* and be independent of system variables. Examples are *v(v.vin)* or *i(i.iin)*. You may also type in just the instance path (for example, *v.vin* or *i.iin*).

The Output List is a signal list containing only system variables. For each signal in this list, the analysis computes zeros.

Step 4. Specify the Operating Point.

Because pole-zero analysis requires an operating point that Saber uses as the bias point, you must specify an initial point file in the Initial Point File field.

- If you have already determined a valid operating point for the analysis, verify that Saber will use the correct file by examining the Initial Point File field in the Input/Output tab. By default, Saber uses the default output of DC analysis, *dc*, as the initial point file.
- If you have not determined a valid operating point, specify **Yes** in the Run DC analysis First? field. This selection informs Saber to execute DC analysis to find the operating point and then to run the pole-zero analysis by using the resulting operating point.

Step 5. Verify the analysis waveforms to create.

Saber uses the Plot File field to define the file that contains the waveforms representing system poles and zeros.

- Plot files contain waveform results. The size of the plot file is a function of the number of data points.
- Pole-zero plot files may contain infinite values.

Step 6. Select Post Analysis View.

You can have the pole-zero data displayed as ASCII text in the Report tool by selecting Report After Analysis (Input/Output tab).

You can also have pole-zero data plotted graphically in Scope. The Plot After Analysis menu field (Basic tab) allows you to specify post-analysis automatic plotting behavior.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms. See "Viewing Pole-Zero Analysis Results" on page 14-4" for additional information.

Step 7. Execute the pole-zero analysis.

Using the default pole-zero analysis values, Saber calculates the poles and zeros for the system in radians per second.

Saber stores all waveforms in a plot file, named `pz`, which can be viewed in Scope. For reference information on pole/zero analysis, refer to the SaberBook online system.

After Saber completes the pole-zero analysis, you use Scope to view the simulation results graphically, or use the Report tool to view the results in ASCII text.

Viewing Pole-Zero Analysis Results

The results of a pole-zero analysis can be viewed both as a graph in Scope and as ASCII text in the Report tool.

Viewing the Results in Scope

Overall, the procedure for viewing results from a pole-zero analysis, is as follows:

- Add the plot file to the Signal Manager's plot file list.
- Open the plot file.
- Select signals to plot.
- Plot the selected signals in a graph.

You can do this step-by-step after an analysis, or you can set up Guide to perform all or part of this process automatically.

The following procedure explains in more detail how to plot waveforms after an analysis:

Step 1. Add the plot file to the Signal Manager and open it.

Guide does this automatically if you specify one of the following under the Plot After Analysis menu field (Basic tab) before starting the analysis:

- **Yes - Open Only**
- **Yes - Append Plots**
- **Yes - Replace Plots**

In all of the above cases, Guide invokes Scope (if necessary) and opens the Signal Manager and the appropriate plot file.

However, if you specify **No** in the Plot After Analysis menu field, Guide does not open the Signal Manager. You can open the last created plot file either by selecting the **Results > View Plot Files in Scope** pulldown menu item. (You can also click on the **Last Plotfile** icon in the Guide icon bar.) This activates the View Plotfiles dialog box. Under the Plot File pulldown menu field, select one of the following:

- **Last** selects the most recent plot file generated by the analysis. This is the default.
- **Plot File Names** allows you to specify a list of plotfiles (separate by spaces or commas) in the Plot File field. You can use a browser to select the plot files by clicking on the **Browse** button.

Under the Plot Action field, select one of the following actions:

- **Open Only** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
- **Append Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager window, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.

- Updates all previously graphed waveforms from the plot file, placing the new waveforms in addition to the corresponding waveforms from the previous analysis.
- **Replace Plots** does the following:
 - Invokes Scope, if necessary.
 - Opens the Signal Manager, if necessary.
 - Opens the latest version of the plot file specified in the Plot File field.
 - Updates all previously graphed waveforms from the plot file, replacing the corresponding waveforms from the previous analysis.

The Append and Replace options are convenient to use during design optimization, when repeatedly running analyses to see the effects of altering design parameters.

After making the settings described above, you open the Signal Manager by clicking on the **OK** button or the **Apply** button. The Signal Manager dialog box appears, along with a Plot File window for the plot file you specified.

If you click on the **OK** button, the View Plotfiles dialog box closes. If you click on the **Apply** button, the dialog box remains open. Click on the **Close** button to close the dialog box.

Step 2. Select the signals to plot.

Assuming that you did not select the Append or Replace Plot Action options in the preceding step, you need to select the signals to be plotted. In the Plot File window, select each signal by clicking on it with the left mouse button. Unselect a signal by clicking on it again (or use the **Deselect** button or Signal menu).

Step 3. Plot the selected signal(s).

Pole-zero analysis can generate pole data, zero data, and gain data. Gain data is displayed as `sfact` (scale factor) in the Plot File list.

You can plot the selected signals by clicking on the **Plot** button or by moving the cursor into the graph window and pressing the middle mouse button. (You can also plot a signal simply by double clicking on its name in the Plot File window.)

The selected signals appear the Graph window. For complete information on manipulating graphs, refer to Scope Graph Window Operation. For complete information on using the Signal Manager

within Scope, type `Signal Manager` in the index tab in the online documentation.

After you plot the necessary signals, you can analyze the results in order to determine the next step in the design process.

Viewing the Results in the Report Tool

A text report can be useful for viewing pole-zero data that is not easily visible in a graphical display.

To view the results of a pole-zero analysis in an ASCII text report, you can either use the **Results > Pole-Zero Report** menu item after the pole-zero analysis completes or select `Yes` in the Report after Analysis field in the Input/Output form.

The Pole-Zero Report form gives you greater control over the produced report.

- In the fields of the Pole-Zero Report forms, in the Basic and Output Control tabs, you can specify your data source and the layout of the pole-zero report.
- In the fields of the Pole-Zero Report forms Columns to Include tab, you can specify the types of information that you want Saber to specify in the pole-zero report

For details on Pole-Zero Report forms refer to the Saber `pzreport` command in SaberBook.

Measuring Pole-Zero Analysis Results

Scope provides several pre-defined automated measurements to dramatically decrease the time it takes to extract s-domain specification data from plotted pole-zero waveforms. If you are already familiar with the Measurement tool, you can skip to Chapter 14.

To perform a measurement on a waveform within Scope, you follow these steps:

Step 1. Display the Measurement Tool.

Within the Scope user interface, you can open the Measurement Tool by either selecting the **Tools > Measurement** pulldown menu item or by clicking on the Measurement icon in the Tool icon bar at the bottom of the Scope user interface.

Step 2. Select the appropriate measurement.

After you display the this tool, you can select which measurement you want to perform. In the Measurement field, you select the measurement by clicking on the downward arrow and selecting the measurement from within the various measurement categories.

The following table lists the measures accessible in the s-domain category.

Measurement	Definition
Damping Ratio	Displays the damping ratio of a pole-zero waveform. The damping ratio is a dimensionless value.
Natural Frequency	Displays the natural frequency of a pole-zero waveform in radians per second.
Quality Factor	Displays the quality factor of a pole-zero waveform. The Quality Factor is a dimensionless value.

Step 3. Select the signal to measure.

You can specify the signal to measure in the Signal field. You can specify the signal by using either of the following methods:

- Click on the downward arrow, and select a signal from the resulting list.
- Select the signal in the current active graph.

Regardless of the method you chose, you must specify a signal prior to executing the measurement.

Step 4. Perform the measurement.

You perform a measurement by clicking on the **Apply** button. This procedure performs the measurement on the specified signal and adds the corresponding information to the active graph.

Step 5. Manipulate measurement information.

During the course of data analysis within Scope, you may want to manipulate which measurements can be viewed. This task can be accomplished by using the Measure Results form. You can open this form either by selecting the **Graph > Measure Results...** pulldown menu item or by double-clicking on measurement text within an active graph.

After you display this form, you can manipulate measurement information by using the following buttons:

- **Delete Measurement** deletes the selected measurement from the active graph.
- **Delete All** deletes all measure text/graphics from the active graph.
- **Show All Values** shows all measurements listed in the left scroll list in the graph.
- **Hide All Values** retains but does not display the measurements listed in the left scroll list.
- Show/hide status bubbles allows you to change the visibility of a measurement on the active graph on a per measurement basis.

The measurement cursor can be dragged to poles and zeros to display the measurement values for individual points.

Determining the Next Step

Using the signal viewing and measurement capabilities of Scope and the Report tool, you determine whether your design meets the pre-determined specifications. If it meets specifications, then you can proceed to one of the following options:

- Verify the time-domain response using pole-zero data by using the **Linear Time Domain Response** analysis as described on Chapter 14.
- Verify the frequency response using pole-zero data by using the **Frequency Response** analysis as described on Chapter 14.
- Tune the design parameter by using statistical or parametric analysis as described in Chapters 8 and 9.

If the design does not meet your specifications, you can take corrective action by using either of the following methods:

- Alter a design/instance parameter(s)

If the design change does not require you to modify the connectivity, you can modify the design parameters by using the Alter form available through the **Edit > List/Alter...** menu item.

- Edit the schematic and re-netlisting.

If the design change requires you to change in connectivity (e.g. different parts, different wiring connections, or any change to the topology), you must make the change in the schematic, re-netlist the design (**Design > Netlist *design_name***), and then re-load the design in Saber (**Design > Simulate *design_name***).

Determining Time Domain Response Using Pole-Zero Data

Linear Time Domain analysis uses pole-zero data to determine the time-domain response of a system. You can also select the type of transient input for the system to determine the system response. The default transient input is an impulse.

To verify that your design meets pole/zero specifications, you execute a pole/zero analysis, view the results in Scope, and evaluate the results to determine the next step. Based on this decision, you can either make the appropriate adjustments to meet specifications and re-run the analysis or continue to the next step in your design analysis process.

The following topics describe how to verify the specifications of the design by using time domain analysis.

1. "Determining Poles and Zeros - Linear Time Domain" on page 14-10
2. "Executing Linear Time Domain Analysis (tresp)" on page 14-10

Determining Poles and Zeros - Linear Time Domain

Linear Time Domain analysis examines the design using pole-zero data. You must find pole-zero data for the design prior to running the analysis, using one of the following methods:

- In the TRESP Analysis form, specify `Yes` in the Run PZanalysis First? field. This selection informs Saber to execute pole-zero analysis to calculate the poles and zeros using the previously calculated operating point. The Pole-Zero form is discussed in the next step.
- Run the pole-zero analysis separately by selecting the **Analysis > Linear Systems Analysis > Pole-Zero** pulldown menu item.

Executing Linear Time Domain Analysis (tresp)

Step 1. Display the Linear Time Domain Dialog Box (Analyses > Linear systems Analysis > Linear Time Responses...).

Step 2. (Optional) Use the -d flat Option.

The Linear Systems Analysis tool requires the use of the `-d flat` option if the design is hierarchical or contains hierarchical templates.

To use the `-d flat` option, perform the following steps:

1. Load a design.
2. Select **Edit > Saber Settings**.
3. Select **Additional options > Specified**
4. type `-d flat`
5. Click **Save**
6. Reload your design and run a DC analysis.

Step 3. Specify the required linear time domain analysis fields.

In order to execute a Linear Time Domain analysis, you must specify the following information:

- End Time defines the time at which the linear time domain analysis stops. For example, if the driving source of the design is a sinusoidal function with a period of 10uS and you want to view the transient response of the first 5 cycles, you enter 50u (5 x 10u) as the value for this field.
- Start Time defines the time at which Saber begins the linear time domain analysis. The default value is 0.
- During linear time domain analysis, Saber uses the Time Step value to determine an initial guess at the next solution point.

As a rule of thumb, you should set this value to the smallest of the following parameters in your design:

1/10th of the smallest meaningful time constant in the design

Shortest rise or fall time of a square/pulse wave driving source

1/100th of the input period of a sinusoidal driving source

- Response Type specifies the transient input to the system. This field is not required, but allows you to produce a desired response type.

Step 4. Verify the analysis waveforms to create.

Saber uses the following two fields to define both the file that contains the waveforms and which waveforms to create:

- Input Plot File specifies the file that contains pole-zero data used for input for the analysis. The size of the plot file is a function of the number of data points and the number of signals in the Signal List.
- Zeros Selection determines which set of zeros, of those stored in the input plot file, will be used to compute the linear time domain response.

Step 5. Select Post Analysis View.

You can have the data plotted automatically in Scope after analysis. The Plot After Analysis menu field (Basic tab) allows you to specify how the plotting is to be done.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

For more information on how to reduce the disk space used by data and plot files, refer to Appendix A: *Files Used During Saber Simulation*.

Step 6. Execute the linear time domain analysis.

Clicking on the **Apply** button executes the analysis. Saber determines the linear time domain response of the design by using pole-zero data to determine system conditions and then calculating how the design responds to the passage of time.

After Saber has finished simulating the design based on your linear time domain analysis specifications, you can view the resultant waveforms in Scope, apply measurements and determine your next step.

Determining Frequency Response Using Pole-Zero Data

Frequency Response analysis uses pole-zero data to determine the frequency response of a system.

To verify that your design meets pole/zero specifications, you execute a pole/zero analysis, view the results in Scope, and evaluate the results to determine the next step. Based on this decision, you can either make the appropriate adjustments to meet specifications and re-run the analysis or continue to the next step in your design analysis process.

The following topics describe how to verify the specifications of the design by using time domain analysis.

1. "Determining Poles and Zeros - Frequency Response" on page 14-12
2. "Executing Frequency Response Analysis (fresp)" on page 14-13

Determining Poles and Zeros - Frequency Response

Frequency Response analysis examines the design using pole-zero data. You must find pole-zero data for the design prior to running the analysis, using one of the following methods:

- In the Frequency Response Analysis form, specify **Yes** in the Run PZanalysis First? field. This selection informs Saber to execute pole-zero analysis to calculate the poles and zeros using the previously calculated operating point. The Pole-Zero form is discussed in the next step.
- Run the pole-zero analysis separately by selecting the **Analysis > Linear Systems Analysis > Pole-Zero** pulldown menu item.

Executing Frequency Response Analysis (fresp)

To perform a frequency response analysis, you specify the frequency range to analyze, verify the values of optional small signal fields, and then execute the command as described in the following steps:

Step 1. Open the FRESP Analysis Form. (Analysis > Linear Systems Analysis > Frequency Response).

Step 2. (Optional) Use the -d flat Option.

The Linear Systems Analysis tool requires the use of the `-d flat` option if the design is hierarchical or contains hierarchical templates.

To use the `-d flat` option, perform the following steps:

1. Load a design.
2. Select **Edit > Saber Settings**.
3. Select **Additional options > Specified**
4. type `-d flat`
5. Click **Save**
6. Reload your design and run a DC analysis.

Step 3. Specify the frequencies to sweep.

In order to execute a frequency response analysis, you must define at least one frequency point by using the Start Frequency field, which results in Saber calculating the frequency response at a single data point. If you want to perform the analysis over a frequency range, you specify the range by using the Start Frequency and End Frequency fields. All frequency values must be positive real numbers.

Step 4. Specify the frequency data point values.

You can control the position of the calculated data points in the defined frequency range by using the following two fields:

- Number of Points defines the number of data points calculated between the frequency range defined by the Start Frequency and End Frequency fields.
- Increment Type defines the spacing between data points, either linear (equal spacing) or the default value, logarithmic.

Step 5. Verify the analysis waveforms to create.

Saber uses the following two fields to define both the file that contains the waveforms and which waveforms to create:

- Input Plot File specifies the plot file that contains pole-zero results used for input for the analysis. The size of the plot file is a function of the number of data points and the number of signals in the Signal List.
- Zeros Selection determines which set of zeros, of those stored in the input plot file, will be used to compute the linear time domain response.

Step 6. Select Post Analysis View.

You can have the data plotted automatically in Scope after analysis. The Plot After Analysis menu field (Basic tab) allows you to specify how the plotting is to be done.

You have the following choices: **No** causes no automatic plotting; **Open Only** opens the plot file after analysis; **Append** opens the plotfile and updates previously graphed waveforms, placing new plots in addition to the current waveforms; **Replace** opens the plotfile and updates previously graphed waveforms, replacing the current waveforms.

For more information on how to reduce the disk space used by data and plot files, refer to Appendix A: *Files Used During Saber Simulation*.

Step 7. Execute the Frequency Response analysis.

Using the default frequency response analysis values, Saber applies a small sinusoidal signal to the input and calculates the frequency response over the range defined by Start Frequency and End Frequency fields. Saber uses the Number of Points and Increment Type fields to determine the number and values of the calculated frequency points.

Saber stores the results for each system variable in the Data file named

`fresp`. Saber also stores all signals defined in the Signal List in a Plot File named `ac`, which can be viewed in Scope. For reference information on FRESP analysis, refer to the SaberBook online system.

After Saber has finished simulating the design based on your frequency analysis specifications, you can view the resultant waveforms in Scope, apply measurements and determine your next step.

Linear Time Response Results

Results produced by the Linear Time Response represent the response of a linearized approximation to your actual system. Linear Time Response employs methods which produce a discrete approximation to a continuous transfer function. Some systems may be numerically ill-suited to this analysis. In general, systems with large numbers of poles and/or zeros, or systems with a very large dynamic range in the poles and zeros may result in numerically unstable results.

If the Linear Time Response analysis produces results which tend toward very large values (effectively plus or minus infinity), do the following:

- Check to see that the pole and zeros values are as expected. Pole values with positive real part will produce responses which tend toward infinity. The response is consistent with expected behavior.
- Try decreasing the duration of the response (e.g, by changing the end time).
- Try changing the time step size. Depending on the particular system involved, increasing or decreasing the time step may help.
- Alter the design so as to change the pole and zero values and thus the corresponding Linear Time Response.

Chapter 14: *Checking Linear Systems Analysis Specifications*

Determining Fault Conditions

NOTE

In order to use this analysis, you must obtain a Testify option license.

Testify allows you to use the Saber simulator to develop and evaluate tests used for detecting fault conditions on equipment such as circuit boards. Testify inserts each fault, one at a time, and runs tests of your own design to measure out-of-specification performance.

- "Testify Process Steps" on page 15-1
- "Using the PinFault Editor" on page 15-11
- "Using Testify with Hierarchy" on page 15-11
- "Testify Fault Wrappers" on page 15-13

The "Getting Started with Testify" tutorial uses an example circuit to demonstrate how Testify can be used to determine the fault conditions of a simple electrical circuit.

Testify Process Steps

To determine the fault conditions of your design using Testify, use the following process:

1. "Using the Netlister with Testify" on page 15-2
2. "Displaying the Testify Form" on page 15-2
3. "Setting Up the Tests" on page 15-2
4. "Determining Nominal Operation and Operational Limits" on page 15-4
5. "Specifying the Faults" on page 15-5
6. "Running the Fault Simulations" on page 15-7

7. "Displaying the Testify Test Results" on page 15-8

"Debugging Fault Runs that Don't Converge" on page 15-9

Using the Netlister with Testify


Testify requires a special netlist in order to insert faults into your design. This netlist contains fault models of the parts whose failures are to be simulated by the test suite. A fault model is a standard MAST template that has a special section of code, called a fault wrapper, providing the details of a part's fault behavior. These fault models are inserted by the netlister when you netlist your design; you do not have to place special models in your schematic. Fault models appear in the netlist as the standard part name appended with `_f`. For example, the `c` capacitor model would become the `c_f` fault model.

The two netlister settings you need to set are found on the Saber/Netlister Settings form, on the Advanced sub-tab of the Netlister tab:

- Fault simulation mode instructs the netlister to insert fault models when the **Yes** button is depressed. The **No** button allows the netlister to produce a standard netlist.
- Generate fault wrappers provides three options for inserting fault wrappers into your design's models:
 - **Yes**—fault wrappers are generated for all of the required parts the first time you netlist.
 - **As Needed**—the netlister will generate fault wrappers only for those parts in the design that do not have fault models either pointed to by `SABER_DATA_PATH` or in the current directory. This feature allows you to create your own library of custom fault models if you need to simulate faults differently than the generic fault wrappers allow.
 - **No**—never create fault wrappers.

Displaying the Testify Form

Once you have created a netlist of your design that contains fault models, you are ready to use Testify to determine fault conditions.

You invoke the Testify form either by clicking the Testify icon  on the Tool Bar or by selecting **Tools > Testify** from the Pulldown Menu Bar.

Setting Up the Tests

Step 1. Click the Test Setup tab

The Test Setup tab is where you define the tests performed on your circuit after each fault has been applied. You add the tests, one at a time, to the Test List column. The Saber commands making up the highlighted test in the Test List are displayed in the Selected Test column.

Step 2. Add a test

Each test is created by using the Experiment form, displayed either by clicking the **Add** button or by selecting the **Edit > Add Test** menu choice.

Step 3. Specify the analyses

You can add any analysis listed in the **Analysis** pulldown menu to the test. Once you select an analysis, SaberGuide adds a button, denoted by the command name, to the Experiment form. You can specify the arguments of the analysis either by clicking on the analysis button or by selecting the **Edit** item from the arrow pulldown menu.

Because adding faults usually affects the operating point of the design, the first analysis placed in the test is typically a DC analysis. Placing this analysis first allows the remaining analyses to use an accurate operating point, given the current fault configuration of the circuit.

Step 4. Optionally, add Vary or Monte Carlo loops

Using the **Loop** menu item in the Experiment dialog box, you can add Vary and Monte Carlo loops to the test. Note that you cannot nest a Monte Carlo loop within another Monte Carlo loop.

Step 5. Add post-processing functions

The final addition that you make to the Experiment dialog box is to add any post-processing functions. Saber executes these functions after it completes the final command. Post-processing functions are listed under the **PostProcessing** pulldown menu. The process of adding and editing post-processing functions to the test is identical to the process of adding analyses.

Step 6. Verify the test

Before you execute the Experiment dialog box, you should verify the order of the analyses and the post-processing commands.

The following functions, accessible from the arrow pulldown menus, allow you to change the order of commands within this form:

- Edit displays the corresponding form for editing.
- Delete removes the command from the test.
- Move Up moves the command up one level in the test.
- Move Down moves the command down one level in the test.

After you verify the structure of the Experiment dialog box, you can click the **Accept** button to place this test in the Test List column of the Testify window.

Step 7. Optionally, add, edit, or delete tests from the Test List

The **Add**, **Edit**, and **Delete** buttons, as well as the **Edit** pulldown menu, allow you to modify the Test List.

Step 8. Optionally, validate the test suite

You can validate the functionality of all of the tests listed in the Test List by selecting the **Run > Validate** menu choice and confirming the results in the SaberGuide Transcript window.

If you exercise this option, the nominal calculated values of each test will automatically be placed in the Nominal column on the Test Ranges tab.

Determining Nominal Operation and Operational Limits

Step 1. Click the Test Ranges tab

The Test Ranges tab is where you determine your design's normal operating region.

- A filled-in check box beside the test name will include that test in the test suite.
- The Name column lists the tests included in the suite, in order.
- The Lower Limit column defines the lower operational limits of each test.
- The Upper Limit column defines the upper operational limits of each test.
- The Nominal column displays the value returned by the test when the circuit elements are all at their nominal values.

Step 2. Calculate the nominal values of the tests

Because the circuit's operational limits can be defined as a function of nominal operation, nominal values for each test measurement must be calculated. You do this by clicking the **Nominal** button.

Step 3. Calculate the test limits

A circuit has a range of acceptable operation that is usually specified by its design specification. If a fault does not cause the circuit to go beyond this acceptable range, the fault will go undetected. Because of this, the operational limits of a test are a crucial factor in determining how many faults are detected by a given test.

The test limits are calculated by one of the two methods defined in the Limits Calculated From field:

- **Monte Carlo**

The default method of calculating test limits is to perform a Monte Carlo analysis of fifteen runs for each test. You can change the parameters of the Monte Carlo analysis, including the number of runs, by clicking the **Edit...** button and filling out the Monte Carlo Analysis form following the procedures of "Executing the Monte Carlo Analysis" on page 12-2.

To execute the Monte Carlo analysis click the **Limits** button.

- **% Nominal**

The arrow pulldown buttons next to the Lower and Upper fields allow two choices for entering limits:

- **Specify Each** lets you set the percentage of nominal for each test by typing the values directly into the Lower Limit or Upper Limit fields.
- **Apply to All** lets you set the percentage of nominal for all of the tests by typing the value in either the Lower or Upper fields at the bottom of the form.

Specifying the Faults

Step 1. Click the Fault List tab

The Fault List tab is where you define what faults will be tested by the test suite.

You compile the possible faults in the Possible Faults listbox; of those,

you place the ones you want tested into the Chosen Faults listbox.

Step 2. Compile the possible faults

- Clicking the **Get Fault List** button will place all of the faults that can be exercised in your circuit—such as open, short, pin-to-pin short, pin-to-supply short, and pin-to-ground short—into the Possible Faults listbox, making them available for inclusion in the test suite.
- The **Get Parameters** button allows you to insert parametric faults—relative changes such as a large value deviation, rather than absolute changes like short- or open-circuits—into the Possible Faults listbox.
 - a. Click the **Get Parameters** button to invoke the Parameter Fault List dialog box.
 - b. Select **Parameters > Browse...** to instruct Saber to compile a list of all of the parameters in the circuit and display them in the Set Parameter List dialog box
 - c. Highlight the parameters of interest and place them in the Parameter Fault List by clicking **OK**.
 - d. Type the failure value of the parameter in the Value column.
 - e. Once you have filled out the Parameter Fault List, click **OK** to place the parametric fault in the Testify Possible Faults listbox.

Step 3. Specify the faults

You specify the faults you want in the test suite by placing them in the Chosen Faults listbox.

1. Highlight the items in the Possible Faults list either with the mouse cursor or by selecting the **Edit > Select All > Possible Faults** menu choice.
2. Filter the possible faults by part type by setting the Part Filter to one of the following:
 - **All Parts** in the design.
 - **Toplevel Parts** in a hierarchical design. This is relevant only for hierarchical designs.
 - **Passive Parts** for resistors, capacitors, and inductors.
 - **Active Parts** for integrated circuits, transistors, and other semiconductor parts.

- **Specified** for filling in part types, for example r* for all of the resistors in the design.
3. Filter the possible faults by fault type by setting the Fault Filter to one of the following:
 - **All Faults** in the design.
 - **Analog Faults** for shorts and opens of all varieties.
 - **Digital Faults** for digital outputs stuck high or low.
 - **All Pin Opens**
 - **All Pin Shorts**
 - **Pin-Pin Shorts**
 - **Pin-Supply Shorts**
 - **Stuck Faults** for digital outputs stuck high or low.
 - **Specified** for filling in fault types not covered in the above choices.
 4. Click the --> button to move the faults into the Chosen Faults listbox.

Running the Fault Simulations

Testify creates the fault/test result matrix on the Results tab.

Step 1. Click the Results tab

Step 2. Specify simulator preferences

You are able to control several of the simulator's options from the Simulator tab of the Testify Preferences form (**Edit > Preferences...**):

- Limit Time will allow simulation time to be limited if the **Yes** button is depressed.
- Fault Time Multiplier is multiplied by the nominal simulation time to determine the simulation time limit. This value is important only if Limit Time is set to **Yes**.

For example, if a nominal transient analysis takes 12 seconds, and Fault Time Multiplier = 5 and Limit Time is **Yes**, then a simulation will be interrupted when it reaches $5 \times 12 = 60$ seconds.

- Maximum Time sets the simulation time limit in seconds.

Chapter 15: *Determining Fault Conditions*

- Open Resistance is the finite resistance that represents an open circuit.
- Short Resistance is the finite resistance that represents an short circuit.

"Debugging Fault Runs that Don't Converge" on page 15-9 explains how to use the Open Resistance and Short Resistance values in debugging.

Step 3. Specify how the results are displayed on the Results form

You can control the color of the test data on the Results tab from the Testify Preferences form (**Edit > Preferences...**) for each of the three result conditions, Beyond Lower Limit, Between Limits, Above Upper Limit:

- a. Click the Display tab.
- b. Click the rainbow color band to invoke the Color Editor.
- c. Click on the color you want.
- d. Click **OK**.

The editable Text fields contain the words used to describe each test result condition in reports generated by the Report Tool; the Text fields do not affect display on the Testify Results tab.

Disable Probes During Testify Run prevents multiple simulations from updating the probe windows each time a test is run. The default is that Probes are disabled. This feature affects SaberSketch only, not the Frameways.

Step 4. Optionally, disable Smart Simulation

Smart simulation is a time saving feature and is selected by default. Smart Simulation prevents Testify from running redundant DC simulations for each test run. You can disable this feature by selecting **Edit > Smart Simulation** to un-check the box.

Step 5. Run the simulations

Clicking the **Run** button begins the test suite. You can pause the test runs by clicking the **Pause** button.

Displaying the Testify Test Results

Once you have run the fault simulations, you can display the results in a text file using the Report Tool.

Step 1. Specify Report formats

You are able to select from a set of possible parameters to include on the report.

1. Select the **Edit > Preferences...** menu choice.
2. Select the Reports tab.
 - The Format listbox allows you to select either of the predefined report formats, Short or Verbose, as defined by the check boxes in the Include section. These check boxes can also be turned on or off individually.
 - You can create a “spreadsheet style” report using tabs and ready for export into Excel, by selecting the Spreadsheet Style box.
 - Automatic Report Name, if the check box is filled in, will name the report either `Short_Report` or `Verbose_Report`, depending upon the Format selected. If the check box is not filled in, you will be queried for the report name.
3. Click **Apply** if you have made changes you want to keep.
4. Click **Close**.

Step 2. Generate the Report

Click the **Report** button at the bottom of the Results tab to generate the report and open the Report Tool displaying it.

Step 3. View the Report form

The topic, Using the Report Tool, in the online help system *SaberBook*, provides instructions for managing the report text file.

Debugging Fault Runs that Don't Converge

Sometimes the insertion of short-circuit and open-circuit faults may cause nonconvergence. The short-circuit default value is $1\mu\Omega$ (10^{-6}), and the open-circuit default value is $1T\Omega$ (10^{12}).

Fault Resistances

If you have convergence problems, you can try changing the values of the fault resistances. The value of the fault resistance that allows the simulation to converge—while still giving meaningful fault detection results—depends on the circuit elements surrounding the fault, and will therefore vary from circuit to circuit. However, as a first step, try either increasing the value of the

short-circuit resistance to $1\text{m}\Omega$ (10^{-3}), or decreasing the value of the open-circuit resistance to $1\text{G}\Omega$ (10^9) or $1\text{M}\Omega$ (10^6). (Note that me is the MAST abbreviation for 10^6 .)

To change either of the default resistances, do the following:

1. Select the **Edit > Preferences...** menu choice from the Testify Pulldown Menu Bar to invoke the Testify Preferences Dialog Box.
2. Select the Simulator tab.
3. Insert the value of resistance you want into the appropriate field.
 - The Open Resistance field specifies the open-circuit fault resistance.
 - The Short Resistance field specifies the short-circuit fault resistance.
4. Click **Apply** to set your changes.
5. Click **Close** to close the form.

You are now ready to re-run your tests.

DC Operating Point

If you have problems with DC operating point convergence under fault conditions, try determining these operating points by replacing the default initial point of zero volts with the nominal, non-fault operating point value. This should improve both convergence and the speed of the simulation for DC.

Make this change as follows.

1. Select the **Analyses > Operating Point > DC Operating Point...** menu choice from the SaberGuide Pulldown Menu Bar. This selection brings up the Operating Point Analysis form.
2. Select the Input/Output tab on the Operating Point Analysis form.
3. Change the Ending Initial Point File from the default name `dc` to a non-default name, such as `dcip`.
4. Click the **OK** button to run the analysis.

In the Testify experiments, to change the Operating Point Analysis forms:

1. Click the **dcanalysis** button on the Experiment form to invoke the Operating Point Analysis form.
2. Click on the Input/Output tab.
3. Change the name in the Starting Initial Point file field from the default of zero to the Ending Initial Point File you created in the last operation, in this case `dcip`.
4. Click the **Accept** button.

Now you are ready to re-run your tests.

Note that this procedure may not help convergence if you are altering source values as part of your tests.

Using the PinFault Editor

Parts capable of fault mode behavior have a symbol port property called `pin_fault` whose value can be changed using the PinFault Editor. You are able to select a combination of possible values to assign to `pin_fault`. The possibilities are:

- open
- all possible shorts to the part's other pins
- stuck at digital 1 or 0

These choices are hard-coded and cannot be changed.

In order to change the value of `pin_fault` on a part, follow this procedure:

1. Open the Symbol Editor on the part in your schematic.
Note that you are modifying the symbol, not the instance.
2. In the Symbol Editor, open the Property Editor on the port that you are interested in.
3. Click in the Value field next to the `pin_fault` property to invoke the PinFault Editor.
4. Select the faults you want to be able to choose from when creating your test suite and click **OK**.
5. Click **OK** on the Property Editor.
6. Close the Symbol Editor.

Using Testify with Hierarchy

Testify allows you to insert faults on one level of hierarchy at a time. You are able to control whether or not the faults are inserted on the pins of the hierarchical symbol or on the parts making up the underlying schematic. You must choose one or the other for any given run of a Testify test suite.

- "Inserting Faults on a Hierarchical Symbol" on page 15-12

- "Inserting Faults within a Hierarchical Symbol" on page 15-12

Inserting Faults on a Hierarchical Symbol

This is a useful means of inserting faults when you have a hierarchical symbol representing an integrated circuit or an ASIC whose internal faults are not covered by your test suite. You can use this method of fault insertion when you want to simulate faults on the pins of the IC or ASIC.

1. Open the Symbol Editor on the hierarchical symbol.
You must modify the ports on the symbol, not the ports on the instance.
2. Open the Property Editor on the first port you want faults applied to.
3. Add a property called `pin_fault`, without giving it a value, and apply the change.
4. Once the `pin_fault` property has been applied to the port, click in the Value field of `pin_fault` to invoke the PinFault Editor.
5. Select the pin faults you want to be available for Testify to apply, and click **OK** on the PinFault Editor.
6. Apply the `pin_fault` property value change to the port.
7. Repeat steps 2-6 for the other ports you want to apply the `pin_fault` property to.
8. Once you have finished applying `pin_fault` to the symbol ports, save your changes and close the Symbol Editor.

You will need to netlist your design again before running Testify.

Inserting Faults within a Hierarchical Symbol

To allow Testify to be able to activate the faults on parts in a hierarchical symbol's underlying schematic, you must assign the `use_fault_model` property to specific instances of the hierarchical symbol.

1. Open the Property Editor on the instance of a hierarchical symbol in your schematic.

Note that you are modifying an instance of the symbol rather than the symbol itself.

2. Add a property called `use_fault_model`, without giving it a value, and apply the change.

3. Once the `use_fault_model` property has been applied to the instance, give it a value of `no`. The possible values are:

- | | |
|------------------|---|
| <code>no</code> | turns the faults at the highest level of hierarchy for that instance off, allowing the faults of the underlying schematic to be activated by Testify. |
| <code>yes</code> | turns the faults at the highest level of hierarchy for that instance on, preventing the faults of the underlying schematic from being activated by Testify. |

4. Apply the `use_fault_model` value change to the instance.

You will need to netlist your design again before running Testify.

Testify Fault Wrappers

A fault wrappers is a section of code within a model that provides the details of a part's fault behavior.

- If you are writing your own template, all you have to do is add the `pin_fault` property on the symbol's ports in order to allow the netlister to add the fault wrapper.

The value or values taken by `pin_fault` represent the faults requiring modeling.

- If you are writing your own template and want to add a *custom* fault wrapper to it, write the fault wrapper in the template using the example as a guide, add a pin fault to the symbol, and select **Generate Wrappers as Needed** in the netlister settings form. This last step keeps the netlister from overwriting the custom fault wrapper you have written.

When a fault wrapper is added to a template of *name*, a new template called *name_f*, is created. This template has all the arguments of the original plus some fault parameters, such as `failures`. The original part is called inside the wrapper template, with all of its arguments passed in. The faults indicated in the properties are constructed using pin fault templates on the pins of the original part. Different faults are introduced into the wrapper by setting the `failures` argument to the desired value, which is then passed to the relevant pin fault part.

Chapter 15: *Determining Fault Conditions*

Every wrapper has in its `parameters` section a message that it prints out when its external parameter `print_failures` is set to `true`. This message is a list of the instance name and the possible faults. Therefore, when a netlist is loaded into Saber the parameter `print_failures` can be changed to `true` to make Saber print out a list of all of the faults in all of the instances in the entire design.

In the following example of a template with a fault wrapper, the symbol **`c_f`** has the property `pin_fault(p):(open;short=m)`. The part of the template extracted from the original template **`c`** is shown in `courier`, and the fault wrapper is shown in bold. The arguments and the original header of the template **`c`** are used, along with the default values (`strucs` are not redefined). The fault parameters are declared and based on the `pin_fault` property. The `pin_fault2.p` fault is connected to the port named `p` of the template **`c`**. Normally there are no faults, and the value is `none`. The other possible values are `open` and `short_m`.


```
component element template c_f p m = c,model,l,w,ic,esr,rleak,tc,
                                tnom,ratings,rth_ja,part_type,
                                part_class,failures,use_failures

electrical p, m

process..imodel model = (tc1 = 0,tc2 = 0)

number c = undef,
        esr = 0,
        rleak = inf,
        l = 0,
        w = 0,
        ic = undef,
        tc[2] = [0,0],
        tnom = 27,
        rth_ja = undef

external number temp, include_stress, c_tol, c_vmax, c_vrmax

external standard..pdist pdist

c..ratings ratings=()

string part_type    = "capacitor",
        part_class  = "generic"

export val p        pwr
export val tc       tempj
export val joule    energy
```

Chapter 15: *Determining Fault Conditions*

```
# added for fault simulation

fault_modes..use_failures use_failures=yes

struc { enum {short_m,open,none} p=none } failures=(
{
external string print_failures

enum {a_short, in_open, fault_off} p_fail=fault_off

parameters {
    if ((print_failures == "true") & (use_failures == yes)) {
        message("%, p short_m", instance())
        message("%, p open", instance())
    }

    p_fail = fault_off

    if (failures->p == off) p_fail = fault_off
    else if (failures->p == open) p_fail = in_open
    else if (failures->p == short_m) p_fail = a_short
    else message("Invalid p failure, %", instance())
}

# End of Failure Checking

c.1 p:pi m:m = c=c, model=model, l=l, w=w, ic=ic, esr=esr,
    rleak=rleak, tc=tc, tnom=tnom, ratings=ratings,
    rth_ja=rth_ja, part_type=part_type,
    part_class=part_class

pin_fault2.p in:p out:pi a:m = fault=p_fail,
    use_failures=use_failures

pwr_d = pwr_d(c.1)
temp_j = temp_j(c.1)
energy = energy(c.1)
}
```

Files Used by Designer Tools

The Designer environment uses and produces several different types of files.

- Preference Files
- Startup Files
- Log Files
- Project Files
- Report Files

Preference Files

This section describes the files associated with saving application preferences and configurations for SaberDesigner.

These files control three different types of application settings:

- Site-specific preference files (for example, `guide.site`) and user-specific preference files (for example `.guide_user`) set the application preferences selected from the **Edit** menu. The `.site` file is read first. If it exists, the `_user` file is then read and overrides the `.site` file where they contradict each other.
- Session-specific configuration files (for example `.guidectfg`) set the configuration parameters selected from the **File** menu choice.
- Site-specific startup files (for example `guideRc.site`) and user-specific startup files (for example `.guideRc_user`) contain AIM scripts that you can run in conjunction with Saber applications. The `.site` file is read first. If it exists, the `_user` file is then read and overrides the `.site` file where they contradict each other.

These three types of application settings are independent of one another.

Appendix A: Files Used by Designer Tools

The following table shows the preference file used by SaberDesigner applications. To make the preference file site-specific, rename it as shown in the table and include the site file in the AI_SITE_PATH environment variable:

	AIM	Guide^a	Scope	Sketch
Site-specific preference	aim.site	guide.site	scope.site	sketch.site
User-specific preference	.aim_user	.guide_user	.scope_user	.sketch_user .aimpart_user
Session-specific preference	N/A	.guidecfg	.scopecfg	.sketchcfg
Preference Dialog Box	Application Preference	Simulator Preference	Graph Preference	Sketch Preferences
Startup File	N/A	.guideRc_user guideRc.site	.scopeRc_user scopeRc.site	.sketchRc_user sketchRc.site
Log File	N/A	guide.log	scope.log	sketch.log

^aFiles listed in this column assume that you invoke Guide standalone (without Scope).

Site-specific Preferences

Site files allow a system or site administrator to configure the environmental preferences of a given application for a large number of SaberDesigner users. These files can be located anywhere on the network, but each user must have this directory location included as part of the AI_SITE_PATH environment variable.

During invocation, each application searches directories listed in the AI_SITE_PATH variable for aim.site, aimpart.site, sketch.site, guide.site, scope.site, harnessRc.site, or project.site, depending on which application you invoke. If the application finds the appropriate file, it loads it. The application continues searching in your home directory for the equivalent _user file. If an equivalent _user file is found, then it merges the contents with any previously found information, which will cause duplicate settings to be overwritten.

Site files are created from `_user` files using the procedure Setting Environmental Preferences in the Installation Manual.

Site File Name	Equivalent <code>_user</code> File Name	Menu Pick	Description
aim.site	.aim_user	Edit > Application Preferences	Contains AIM scripts.
aimpart.site	.aimpart_user	From the Parts Gallery, select Edit>New Part or New Category	Defines the database of user parts added through the Parts Gallery.
guide.site	.guide_user	Edit > Preferences	Defines a global set of parameters for Guide.
scope.site	.scope_user	Edit>Graph Preferences	Defines a global set of parameters for Scope of the combination of Scope and Guide.
sketch.site	.sketch_user	Edit>Schematic Preferences	Defines a global set of parameters for Sketch.
project.site	.ai_prj (.project_user does not exist) See the topic titled Project Files.	Edit>Simulator/Netlister Settings...	Defines a global set of parameters for the Saber/Netlister Settings... dialog box.

If you want to change the contents of any of these files, you can either delete the file to return to default preferences or you can reconfigure the application preferences and overwrite the old `.site` file.

User-specific Preferences

Users can have a local equivalent of site files in their home directory. The local files begin with a period (.) and have an `_user` suffix such as `.aim_user`. The preferences contained in the `_user` files, if present, override the preferences in the corresponding `.site` file. When a user saves application preferences, Saber preferences, or graph preferences, they are saved in a local `_user` file.

Appendix A: Files Used by Designer Tools

Changing user-specific preferences

To generate these files, choose the menu item shown in the file descriptions below and save the results of your edits.

To use the site-specific file (if present) or return to the original default preferences, delete the `_user` file and shut down the associated `aimserver` session by entering the following:

```
aimserver sessionshutdown qaz,joe
```

where `qaz` is a display name and `joe` is a *userid*.

Available user-specific preference files

The following list describes the user-specific preference files used with SaberDesigner:

_user File Name	Menu Pick	Description
<code>.aim_user</code>	Edit > Application Preferences	Contains AIM scripts.
<code>.aimpart_user</code>	From the Parts Gallery, select Edit>New Part or New Category	Defines the database of user parts added through the Parts Gallery.
<code>.guide_user</code>	Edit > Preferences	Defines Guide environment preferences such as simulator settings and the Report Tool setting.
<code>.scope_user</code>	Edit>Graph Preferences	Defines a global set of parameters, such as graph colors and graph settings, for Scope of the combination of Scope and Guide.
<code>.sketch_user</code>	Edit>Schematic Preferences	Defines a global set of parameters for Sketch.
<code>.project_user</code>	Edit>Simulator/Netlister Settings...	Defines a global set of parameters for the Saber/Netlister Settings... dialog box.

Session-specific Configuration Parameters

This section describes the directories and files located in your home directory that contain SaberDesigner session configuration parameters. Session configuration parameters consist of things such as window location and sizes, a list of available colors, and pathnames to various data sources.

The following subsections describe each of these files in more detail.

.guidecfg File

The `.guidecfg` file is read from your home directory when you invoke Guide in a stand-alone mode (no Scope). This file holds the Guide session configuration parameters when you choose the **File > Configuration > Save** or **Save on Exit** menu item.

To change the settings in this file, reconfigure the settings within Guide and save them (overwriting the old data).

To ignore the settings in this file for a given session, use the `-noconfig` argument to bypass the `.guidecfg` file when you invoke Guide.

To clear the settings in this file, choose the **File > Configuration > Clear** menu item.

.sketchcfg File

The `.sketchcfg` file is read from your home directory when you invoke Sketch. This file holds the Sketch session configuration parameters when you choose the **File > Configuration > Save** or **Save on Exit** menu item.

To change the settings in this file, reconfigure the settings within Sketch and save them (overwriting the old data).

To ignore the settings in this file for a given session, use the `-noconfig` argument to bypass the `.sketchcfg` file when you invoke Sketch.

To clear the settings in this file, choose the **File > Configuration > Clear** menu item.

.scopecfg File

The `.scopecfg` file is read from your home directory when you invoke Scope or the combination of Scope and Guide. This file holds the session configuration parameters when you choose the **File > Configuration > Save** or **Save on Exit** menu item.

If you want to change the settings in this file, reconfigure the settings within Guide/Scope and save them (overwriting the old data).

Appendix A: Files Used by Designer Tools

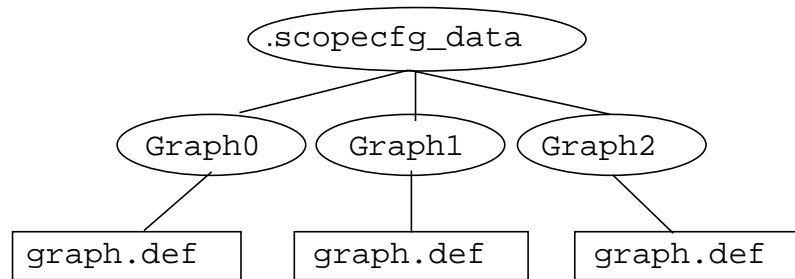
If you want to ignore the settings in this file for a given session, use the `-noconfig` argument to bypass the `.scopecfg` file when you invoke Guide/Scope.

If you want to clear the settings in this file, choose the **File > Configuration > Clear** menu item.

.scopecfg_data Directory

This directory is located in your home directory. This directory contains a separate directory for each graph from a Scope session when you choose the **File > Configuration > Save** or **Save on Exit** menu item. The graph directories contain files that store the graph configuration from a Scope session.

The directory has the following structure:



Graph0, Graph1, Graph. . . represent subdirectories associated with each open graph in the Scope session. Under each subdirectory is a file called `graph.def` that contains the data required to restore the contents of each graph.

Startup Files

applicationRc_user and *applicationRc.site* files force code to be executed by the application at startup.

During invocation, each application searches directories listed in the `AI_SITE_PATH` variable for the `guideRc.site`, the `scopeRc.site`, and the `sketchRc.site`, depending on which application you invoke. If the application finds the appropriate file, it loads it. The application continues searching in your home directory for the equivalent `_user` file. If an equivalent `_user` file is found, then it merges the contents with any

previously found information, which will cause duplicate settings to be overwritten.

The following files reside in your home directory and contain AIM commands controlling the application as listed below:

- `guideRc.site` - controls SaberGuide, during SaberGuide-only invocation (no Scope), on a site wide basis.
- `.guideRc_user` - controls SaberGuide, during SaberGuide-only invocation (no Scope), on your local machine.
- `scopeRc.site` - controls Scope and SaberGuide (if SaberGuide is invoked with Scope) on a site wide basis.
- `.scopeRc_user` - controls Scope and SaberGuide (if SaberGuide is invoked with Scope) on your local machine.
- `sketchRc.site` - controls Sketch, during Sketch invocation, on a site wide basis.
- `.sketchRc_user` - controls Sketch, during Sketch invocation, on your local machine.

There is a similar pair of files for `.aimRc_user` and `aimRc.site` which are loaded before the files that all applications get. Unlike the *applicationRc* files, which are read only by the applications they are named for, the *aimRc* files are global files.

Log Files

A log file is created in the directory where the application was invoked and contains a log of the activities performed during a session. The contents of these files can be used to create Aim scripts to control the application.

- `guide.log` - created during a SaberGuide-only (no Scope) session.
- `scope.log` - created during a Scope and SaberGuide session (if SaberGuide is invoked with Scope).
- `sketch.log` - created during a Sketch session.
- `harness.log` - created during a iQBus session.

Project Files

Edit > Simulator/Netlister Settings (Sketch) and **Edit > Simulator Settings (Guide and Scope)** are saved for each individual design. The settings are saved in a *design.ai_prj* file in the current directory of the design. Every time you open a design, the settings in the *design.ai_prj* file are applied to the Designer application.

In order to create a *project.site* file, follow this procedure:

1. Specify the options on the Simulator/Netlister Settings form.
2. Click the **Save** button.
3. This creates a *designname.ai_prj* file in your working directory.
4. Rename *designname.ai_prj* to *project.site*.
5. Place the path to *project.site* in `$AI_SITE_PATH`.

Report Files

There are two kinds of report files.

.rpt files are ASCII files generated by the Report Tool using output from simulation analyses.

.r1 files are the raw results produced by the analyses and used by the Report Tool to generate *.rpt* files. *.r1* files can be configured as three different file formats by using the Saber config command. The formats are as follows:

design.name.r1
design.r1.name
design/name.r1

where *design* is the name of the design file and *name* is user selectable (with a default to the analysis abbreviation).

Appendix A: *Files Used by Designer Tools*

Netlister Command Reference

Saber Netlister Command Reference

- "Netlister Usage - SaberDesigner" on page B-1
- "schtos Command Line - SaberDesigner" on page B-3

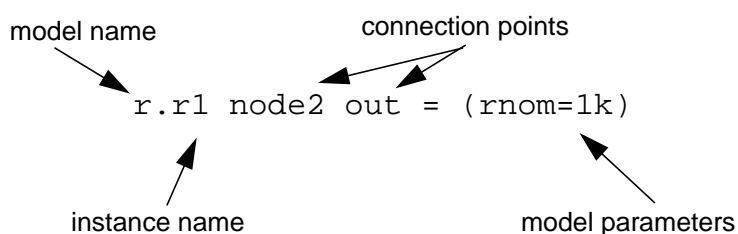
NOTE

Designs created with previous versions of supplied symbols must be saved before netlisting and simulating. If not, some of the SaberDesigner symbols, such as supplies, may not netlist correctly.

Netlister Usage - SaberDesigner

The netlister maps all parts in the schematic to an associated MAST template and generates a Saber netlist using the mapped template names. Saber determines the location of the template model descriptions using the following search path (current directory, SABER_DATA_PATH environment variable, SABER_HOME environment variable).

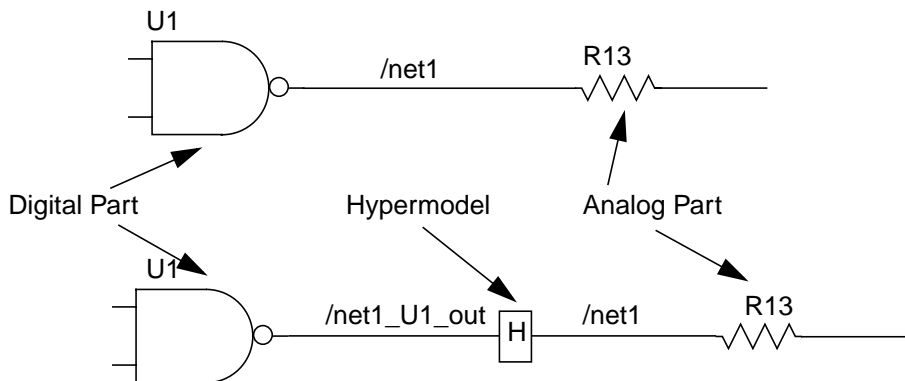
The netlister creates an entry for each symbol in the design. The following example and list shows how the netlister determines each values in the netlist entry:



Appendix B: Netlister Command Reference

- Model name is defined by the `primitive` property on the instance assuming that a user-defined mapping file was not used during netlisting.
- Instance name is defined by the `inst` property on the instance. If you do not specify the `inst` property on an instance, the netlister uses the schematic-capture tool auto-generated name.
- Connection points are defined by the net names in the schematic. If you do not specify net names in the schematic, the netlister uses the schematic-capture tool auto-generated name.
- Model parameters are added when non-default values are defined on the instance in the schematic.

If a net connects an analog part to a digital part, then the netlister inserts a Hypermodel on the connecting net. Because the insertion of this non-physical part requires the creation of a new net. The net on the analog side retains the existing net name but the net on the digital side is named `<netname>_<digital_part_inst>_<pinname>` as illustrated in the following figure:



The Hypermodel is inserted after the schematic capture process by the netlister and is not visible in the schematic. The bottom part of the previous figure just uses the schematic representation to demonstrate the concept of net re-naming due to Hypermodel insertion.

If the netlister reported any errors, then you should examine the transcript of the netlister (`netlister_name.out`) and resolve the error messages.

The inputs to these netlisters are the circuit interconnections produced from SaberSketch design, and component descriptions (from various libraries). The output of a netlister is a file (called a netlist) that completely characterizes the circuit. For these netlisters, this output file will have a `.sin` extension (for Saber **I**nput).

sctos Command Line - SaberDesigner

NOTE

To set the netlister preferences (options) from within SaberSketch use the **Edit > Netlister/Saber Preferences** pulldown menu item.

The `sctos` command is used to create a netlist for the Saber simulator. Files referenced in command line options must be in the data search path. The SaberSketch design, from which the netlister extracts information, must be in the current (working) directory. `sctos` also produces an output file containing the messages that are written to the screen during program execution. This file is named `sctos.out`.

<code>sctos</code> [[-ae] [-dgne] [-fault <i>option</i>]
[-g <i>ground_net_name</i>] [-h <i>hypermodel_lib</i>] [-header]
[-m <i>user_map_file</i>] [-n <i>netlist_template_file</i>]
[-ns <i>filename</i>] [-o <i>netlist_name</i>] [-p <i>power_net_name</i>]
[-pcif] [-q] [-tdb <i>option</i>] [-tI] [-tW] [-v]
[-vhdl] <i>design_name</i>

Command line options can be included in any order. The following list describes the command line options available for this netlister.:

- ae causes the netlister to assume that templates will be provided so it does not search for them. When you use this option with the `catosv` netlister, or when using the `-vhdl` option for VHDL netlisting, it places symbols not specifically assigned to the Verilog netlist, either by a mapping file entry or by the `Target_Simulator` property, into the Saber simulator netlist rather than the Verilog simulator netlist. You can use this command line option with the `SaberPrepend` property to allow the inclusion of templates at the beginning of the netlist so that they can be referenced later in the netlist.
- dgne declares global nets external electrical. This prevents the automatic connection of global nets as ports throughout the hierarchy.

Appendix B: Netlister Command Reference

`-fault option` turns on fault wrapper generation for the Testify Test Manager. The following sub-options control whether fault wrapper files are written for primitive templates:

- `yes` - always generate wrapper files
- `no` - don't generate wrapper files
- `as_needed` - search for the wrapper file, write one if it's not found.

If `-fault` is specified without one of these arguments, `yes` is the default.

The primitive fault wrappers are the separate `.sin` files that are created in the working directory, like `r_f.sin` or `nand_l4_f.sin`. Hierarchical wrappers go inside the main netlist are generated regardless of the value of the `-fault` sub-option.

`-g ground_net_name` name of the top-level-symbol, zero-reference or ground net to which TTL, ECL, and MOS Hypermodel interfaces are to be connected. Use this option if your schematic contains symbols that are not connected to a ground net. Node 0 has been made the default ground or reference node for Hypermodel interface connection. This means that the command line option `-g` is not needed when Hypermodel interfaces are to be connected to node 0.

`-h hypermodel_library` name of the Hypermodel library to be used for mixed-mode support. If you do not include an extension, the netlister uses the extension `.shm`. If more than one Hypermodel library is required, a separate `-h` option clause is required for each one. The order in which you specify the libraries is important only if one library contains Hypermodel interface definitions that are dependent on definitions in another Hypermodel library. In this case you should specify the dependent library after the one it is dependent on.

`-header` creates a template header for the design being netlisted. The netlister places the template header at the beginning of the top-level netlist.

-m
user_map_file filename of the user's custom mapping file, if one is to be used. Entries in this mapping file are merged with those in the standard mapping files. If you enter a filename without an extension, or a filename with the extension `.map`, the netlister uses a mapping file with the filename and the extension `.map`. If you enter a filename with a different extension, the netlister issues a warning message and uses the mapping file with the name and extension you have specified.

The mapping file must reside in a directory that is referenced by the `SABER_DATA_PATH` environment variable.

-n
netlist_tmpl_file name of the netlist template file that the Verilog netlist writer is to use. The netlist template file contains structure and formatting information used by the Verilog writer to create a Verilog `.v` netlist file. The filename must have the extension `.ntf`. If you do not specify this option, the netlist template file provided with the netlister (`verilog.ntf`) will be used. You must make a special netlist template file (*file_name.ntf*) to create netlists for Verilog only. This file should not contain the Verimix commands that are in the `verilog.ntf` file distributed with the netlisters. Furthermore, if your Verilog input files do not contain timescale directives, you cannot have one in the Verilog netlist produced by the netlister. Therefore, there should be no timescale directive in the `.ntf` file either. To switch between netlisting for Verilog only and netlisting for Saber/Verilog, the `.ntf` file should have a name other than `verilog.ntf` and you must specify this name with the `-n` command line option. Then, netlisting without the `-n` option gives you the Saber simulator and Verilog simulator netlists and netlisting with the `-n` option gives you the Verilog netlist only.

Appendix B: Netlister Command Reference

-ns *filename* save a state file to improve netlister performance. The state file contains a compiled version of all of the data in the mapping and Hypermodel files. The netlister can read the state file much faster than it can parse the mapping and Hypermodel files, so the total time required to create a netlist is reduced.

The netlist state file will be used as long as the names and time stamps of the mapping and Hypermodel files do not change. If they change (for example, if a mapping file is edited, or a new Hypermodel file is specified, all of the files will be re-read, and the state file will be rewritten. Therefore, if there are frequent changes made to mapping files or Hypermodel files, turn this option off to prevent performance degradation.

-o *netlist_name* filename to be used for the Saber simulator and (if specified) partner simulator netlists. The netlister adds the appropriate extension. For example, the netlister adds the extension `.sin` to a Saber simulator netlist file; it adds the extension `.v` to a Verilog netlist. By default, if you do not specify this option, the netlister names the netlist file the same as the name of the design with the appropriate extension added.

-p *power_net_name* name of the top-level power net to which the TTL, ECL, and MOS Hypermodel interfaces are to be connected.

-pcif preserves the case of included file names. Allows file names to be saved using upper case and/or lower case ASCII characters. Normally, file names are converted to lower case characters.

-q cancels the display of informative and warning messages turned on by either of the `-tI` or `-tW` options. This option cannot turn off any messages that the netlister displays by default.

-tdb *option* controls the use of the Template Database (TDB) by the netlister, using the `on`, `off`, `nogen` or `warn` options.

Only one option, `on`, `off`, or `nogen`, may be used at a time. However, `warn` may be used in combination with the other options.

- `on`—Full Template Database support, including user templates.

This is the default. With this option, it is not necessary to have mapping files that include basic information about templates, such as the names, order, and types of arguments and connection points. In addition, some netlisting features, such as type checking, are available only when the template database is used. For information on the files used by the template database, see The Template Information System.

- `off`—User templates are ignored; use the supplied database only. Since this option avoids searching for templates on disk, it provides the best performance.

However, due to the reduced functionality, this option is recommended only when no custom templates are used.

NOTE

If you have custom templates with the same names as supplied templates, you should not use this option, as it may lead to netlisting errors or incorrect netlists.

- `nogen`—Do not generate user template information (.ai_tdb files), but use such information if it is already available.

For information on the template database files, refer to The Template Information System.

NOTE

Use of this option may cause netlisting errors if the template database files are not up to date.

`warn`—Errors coming from the TDB are treated as warnings by the netlister, not as errors. The TDB flags errors for problems such as invalid templates and failure to write database files. Typically, these are errors because Saber can be expected to run into the same problem that the TDB did.

Appendix B: Netlister Command Reference

- `-tI` turns on any remaining informative or warning messages that the netlister does not display by default or that have not been turned on by the `-tW` option.
- `-tW` turns on warning messages that the netlister does not display by default, and performs the following checks on the design as it is netlisted:
- verification of valid connections between templates. For example, an electrical pin connected directly to a mechanical pin will cause a warning.
 - verification that template parameters without default values have a value provided in the symbol.
 - issue warnings if the template's connection point list does not match the port list of the symbol.
- These three checks are performed only for MAST templates, not for partner models
- `-v` creates a Verilog netlist as well as a Saber netlist. When the `-v` option is specified, the string `v` is added to the end of the netlister *name*. What this means is that the netlister will then use the Verilog mapping file (for example, `schtosv.map` instead of `schtos.map`) and produce a `v` transcript file (for example, `schtosv.out` instead of `schtos.out`).
- `-vhdl` specifies that all instances targeted to the Saber/ModelSim Co-Simulator Interface (via a mapping file or the `target_simulator` property) are to be written to a VHDL structural netlist for Saber/ModelSim. When the `-vhdl` option is specified, the string `vhd` is added to the end of the netlister *name*. What this means is that the netlister will then use the VHDL mapping file (for example, `schtosvhd.map` instead of `schtos.map`) and produce a `vhd` transcript file (for example, `schtosvhd.out` instead of `schtos.out`).

Simulating Digital Parts in Saber

The following topics describe how to simulate digital parts in Saber:

- "Hypermodels - Overview" on page C-1
- "Using Default Hypermodels" on page C-2
- "Using Ideal Hypermodels" on page C-2
- "Using Technology-Specific Hypermodels" on page C-3
- "Hypermodel Filenames and Logic Families" on page C-3
- "Creating Part Number-Specific Hypermodels" on page C-4
- "Using Multiple Hypermodel Families in a Design" on page C-6
- "Selecting Hypermodels from the Saber/Netlister Settings Form" on page C-7
- "Net Re-naming Due to Hypermodel Insertion" on page C-7

Hypermodels - Overview

In the simulator, the behavior of digital signals are represented as discrete states (e.g. 0,1,Z,X), and analog signals are represented as continuous curves. If your design contains both analog and digital parts, Saber must map signal values between analog and digital waveforms.

Saber accomplishes this mapping using *Hypermodels*. Hypermodels emulate input and output terminal characteristics, such as delays attributable to rise and fall times, and can include the loading effects of a given digital technology. The Hypermodel does not determine the values within the digital model, such as propagation delays. The netlister automatically inserts Hypermodels at all analog-digital boundaries. Because Hypermodels are not "physical" components, they are added only to the netlist, not the schematic, for the purpose of simulation.

Although the Component Library lists over 2000 logic devices having characterized Hypermodels, they are not directly supported with schematic symbols or menu items in the Parts Gallery. In order to implement these parts, you will place the generic digital part in the schematic, specify propagation delay properties, and assign the appropriate Hypermodel depending on your requirements for accuracy and simulation speed.

For more information on Hypermodel properties and their effect on simulation results, see the SaberBook topic, Overview of Hypermodel Analog/Digital Interface Templates.

This procedure discusses using Hypermodels to map between analog and digital signals for Saber native mixed-signal simulation. It does not include digital to digital Hypermodels (used for co-simulation) or mixed-technology Hypermodels (for mapping analog to mechanical signals).

Using Default Hypermodels

Default Hypermodels use the 5V CMOS technology ideal Hypermodel. They are designed to get fast, approximate results in the simulation. They have the following restrictions:

- The ground net name to which digital parts are referenced in your design must be named '0'. The netlister automatically connects these types of Hypermodels to the 0 net in the Saber netlist.
- Default values are to be used for properties such as propagation times and inertial delays of digital parts in your design for which you have not specified values.
- Digital parts from the Saber library are to be simulated by the Saber simulator, not a partner simulator.

If default Hypermodels meet your simulation requirements, no further action is required from you. The netlister automatically adds default Hypermodels.

Using Ideal Hypermodels

These Hypermodels provide an approximate translation between analog signals and digital logic states for different technologies. These Hypermodels model some digital parameters, such as voltage levels and rise/fall times, and they behave as ideal voltage sources and ideal impedances on the analog side. Less time is needed to simulate a design containing these Hypermodels. However, the results are not as realistic as they are if you use a

fully-characterized technology-specific Hypermodel because ideal Hypermodels do not model all of the behavior of the actual devices.

These Hypermodels may be useful when you first start developing a design and want to get a quick, approximate idea of how your design will perform. They are available for a number of logic families including TTL, CMOS, and ECL.

If you want to use these Hypermodels in you design, you must specify the appropriate Hypermodel file in the netlister options form as described in the topic titled "Selecting Hypermodels from the Saber/Netlister Settings Form" on page C-7. No other action is necessary to use these types of Hypermodels.

The topic titled "Hypermodel Filenames and Logic Families" on page C-3 gives a table comparing each logic family to the filenames used for both ideal Hypermodels and technology-specific Hypermodels.

Using Technology-Specific Hypermodels

Compared to the ideal Hypermodels, these Hypermodels model additional characteristics (such as current levels, output capacitance and leakage currents) and improve the accuracy of the characteristics modeled by the ideal Hypermodels. The analog input and output characteristics are non-linear and reflect the digital input state, power supply levels and external load. Because these Hypermodels model a number of different effects, more simulation time is required to simulate your design, but the results will be more precise.

You may want to use this type of Hypermodel once you have established the basic configuration of your design and are ready to "fine-tune" the performance of the design. To use these Hypermodels in your design, you must specify the appropriate Hypermodel file in the netlister options form as described in the topic titled "Selecting Hypermodels from the Saber/Netlister Settings Form" on page C-7. No other action is necessary to use these types of Hypermodels.

The topic titled "Hypermodel Filenames and Logic Families" on page C-3 gives a table comparing each logic family to the filenames used for both ideal Hypermodels and technology-specific Hypermodels.

Hypermodel Filenames and Logic Families

The following table compares logic family to the filenames used for both ideal Hypermodels and technology-specific Hypermodels.

Logic Family	Ideal Name in Form (filename)	Technology-Specific Name in Form (filename)
5 volt CMOS	Ideal CD (cd_ide.shm)	RCA CD 5V (cd5.shm)
15 volt CMOS	Ideal CD (cd_ide.shm)	RCA CD 15V (cd15.shm)
ECL MC1600 series	Ideal ECL (ecl_ide.shm)	ECL (ecl.shm)
Military High Speed CMOS	Ideal MHC (mhc_ide.shm)	Military HC (mhc.shm)
High Speed CMOS	Ideal HC (hc_ide.shm) Ideal HCT (hct_ide.shm)	Motorola HC (mt.shm)
Fast TTL	Ideal Fast (f_ide.shm)	National Fast TTL (ns.shm)
Advanced TTL (AS/ALS)	Ideal ALS (als_ide.shm)	TI ALS/AS (ti.shm)
Standard/LS TTL	Ideal LS (ls_ide.shm)	TI LS (ti2.shm)

All supplied Hypermodel files are located in the *install_home*/template/hypermod directory and have a .shm extension.

The *Analogy Parts Library*, in SaberBook, has more detailed information on Hypermodel templates.

Creating Part Number-Specific Hypermodels

If you only define a technology-specific Hypermodel file in the Netlister options form, the netlister will add a generic Hypermodel for that specific

family at the analog-digital boundary. If you want a part to be characterized like an actual component, you must define a specific Hypermodel name on each digital pin on the part.

For example, rather than specifying technology-specific Hypermodels that model “generic” TTL low-power Schottky digital inputs and outputs for the inverters a design, you can specify Hypermodels that have been characterized to model the specific input and output behavior of a 74LS04 inverter.

The following procedure shows how to define a Hypermodel to a digital pin using a 74LS04 inverter as an example.

1. Specify the TI LS Hypermodel file on the Saber/Netlister Settings form.

The procedure for specifying Hypermodel files is described in the topic titled "Selecting Hypermodels from the Saber/Netlister Settings Form" on page C-7.

2. The Hypermodel files are located in the `template/hypermod` directory.

Do a text search on the file `ti2.shm` for “74LS04”.

The search returns the following:

```
74LS04:adadadg dadadap::          ti74ls_15
```

The a,d,g, and p characters indicate inputs (a), outputs(d), ground (g) and power (p) net on the 74LS04. `ti74ls_15` is the model designation.

3. Add a `SaberModelName` property to each port of the digital part.
 - a. Open the Property Editor on all of the ports.
 - b. Add a `SaberModelName` property to all ports. Add the model designation to the Value field.

In this example, on the inverter, you would add one property to all of the ports:

Name	Value
<code>SaberModelName</code>	<code>ti74ls_15</code>

- c. Click **Apply**.

Note that in the general case, some ports might behave like one kind of part, and other ports might behave like a different part, as in the next example.

1. Do a text search on the file `ns.shm` for “74F74”. The search will return

```
74F74:aaaaddg ddaaaap:: ns74f_7k
```

```
( [ 2 , 3 , 11 , 12 ] = ns74f_8u )
```

2. Like the other example, the a,d,g, and p characters indicate inputs (a), outputs(d), ground (g) and power (p). Pins can also be bidirectional (b), though none are on this device. The line also states that the ns74f_7k model, also defined in ns.shm, will be applied to all pins except 2, 3, 11, and 12 (the CLK and D inputs for each flip flop). An ns74f_8u model will be applied to these pins.

Using Multiple Hypermodel Families in a Design

The procedure to include more than one logic family in a design—for example, to have both TTL and CMOS parts in a circuit—is the same as that for the topic titled "Creating Part Number-Specific Hypermodels" on page C-4, except that you must specify, in the Saber/Netlister Settings form, the multiple Hypermodels corresponding to the logic families in your design, rather than only one Hypermodel.

You then assign a `SaberModelName` property with the value of the model designation to each pin on the digital parts in the design.

If you want to assign unique power and ground connections to each Hypermodel in your design, use the symbol port properties, `SaberModelPowerPin` and `SaberModelGroundPin`, as follows:

Note that these properties must be assigned to the symbol's ports, not to the instance's ports.

Invoke the Symbol Editor on the symbol and assign these properties to the port you want attached to a Hypermodel:

- `SaberModelName` = model designation of the Hypermodel
- `SaberModelPowerPin` = name of that symbol's port that you want the Hypermodel's power pin attached to
- `SaberModelGroundPin` = name of that symbol's port that you want the Hypermodel's ground pin attached to

The following example illustrates how to use these properties. A symbol `sym` has three ports: `VCC`, attached to power, `VEE`, attached to ground, and `A`, attached to an `idp` Hypermodel. You will assign the following properties to port `A`:

```
SaberModelName = idp  
SaberModelPowerPin = VCC  
SaberModelGroundPin = VEE
```

Selecting Hypermodels from the Saber/Netlister Settings Form

Step 1. Open the Saber/Netlister Settings form. (Edit>Saber/Netlister Settings...).

Step 2. Select the Netlister tab, and then the Hypermodels tab.

The Available listbox displays the pre-defined Hypermodels you can use during simulation.

Step 3. Specify the Hypermodel.

- Click on the Hypermodel you want in the Available listbox.
- Add the Hypermodel to the Selected listbox by clicking the <<>> button between listboxes.
- Save the setting by clicking the **Apply** button, then the **Save** button on the bottom of the Saber/Netlister Settings form.

Step 4. Option—specifying power references.

If you used technology-specific Hypermodels (these are listed in the table in the topic titled "Hypermodel Filenames and Logic Families" on page C-3), you can specify the name of the power node that should be used for the upper limit (rail) for the analog signal in the Power net name field on the Basic tab. By default, the Hypermodel is connected to VCC. You can only define one power net for a design.

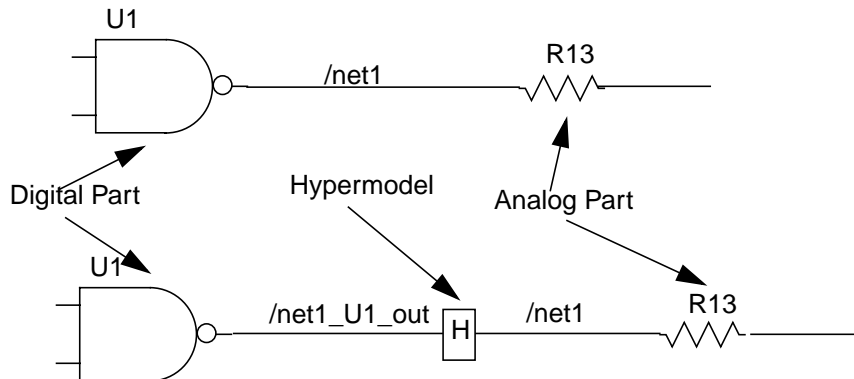
Step 5. Option—specifying ground references.

If you used any Hypermodel other than default ideal, you can specify the name of the ground node that should be used for the lower limit (rail) for the analog signal in the Ground net name field on the Basic tab. By default, the Hypermodel is connected to the node called 0.

Step 6. Close the Saber/Netlister Settings form by clicking on the Close button.

Net Re-naming Due to Hypermodel Insertion

If a net connects an analog part to a digital part, then the netlister inserts a Hypermodel on the connecting net. The insertion of this non-physical part requires the creation of a new net. The net on the analog side retains the existing net name while the net on the digital side is re-named `<netname>_<digital_part_inst>_<pinname>` as illustrated in the following figure:



The Hypermodel is inserted after the schematic capture process by the netlister and is not visible in the schematic. The bottom part of the previous figure uses the schematic representation to demonstrate the concept of net re-naming due to Hypermodel insertion.

If the netlister reported any errors, then you should examine the transcript of the netlister (*netlister_name.out*) and resolve the error messages.

appendix *D*

Stress Analysis Concepts

Before you begin stress analysis, you should understand some important terminology and concepts used in the stress analysis. The following sections explain how to interpret manufacturer's safe operating area (SOA) curves to determine stress values and define how thermal effects are modeled for stress analysis.

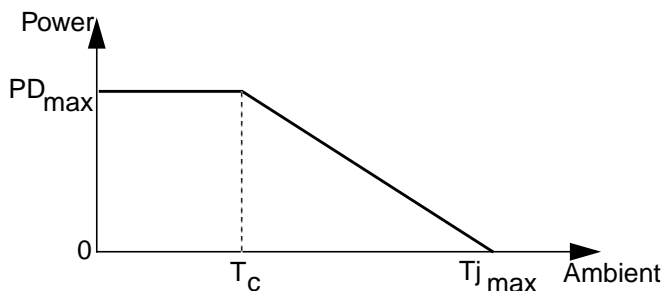
Safe Operating Area (SOA) Curves

Concepts of Stress Analysis

Ratings for operating conditions are specified by the manufacturer for stress measures for a part. Operating conditions are also referred to as maximum operating limits, safe operating limits (SOL), or safe operating area (SOA). A maximum operating condition for a particular stress measure may be a single value, or it may be defined as a function of another operating condition such as ambient temperature or applied voltage.

The following figure shows a typical curve provided on a data sheet by a manufacturer to specify the maximum rating for the power dissipation of a resistor as a function of ambient temperature.

The area under this curve is known as the *safe operating area* or SOA. If the resistor is operated outside the conditions defined by the SOA, it will be overstressed.



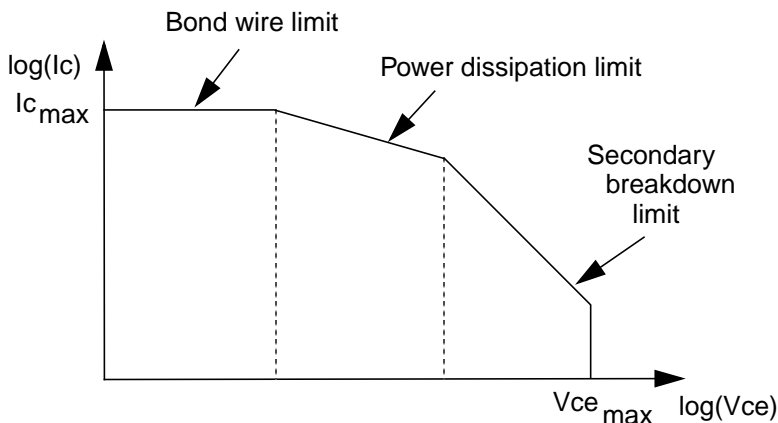
Appendix D: *Stress Analysis Concepts*

For example, the maximum rating for power dissipation for a resistor may be 1/4 watt up to a certain ambient temperature T_c . Above this temperature, the maximum power dissipation drops linearly until the maximum operating temperature $T_{j_{max}}$ is reached.

The slope of the curve between T_c and $T_{j_{max}}$ is the negative reciprocal of the thermal resistance of the resistor. This value is used in the calculation of the internal temperature of the resistor.

Rating curves such as those shown in the following figure are frequently referred to by manufacturers as derating specifications. The curves actually show rated values for the operating conditions before any derating has been applied.

A bipolar junction transistor has ratings such as the maximum power dissipation, maximum voltages, and maximum currents. These ratings may vary as a function of such conditions as junction temperature, applied voltage, and radiation. shows the maximum collector current ($I_{c_{max}}$) as a function of the collector-emitter voltage ($V_{ce_{max}}$). This curve is made up of several segments, each corresponding to a different stress limit.



Thermal Effects Properties

Thermal effects are calculated in the underlying model (template) for a part by using an equation of the following formula:

$$\text{temp}_j = \text{temp} + \text{pwr}_d * \text{rth_eff}$$

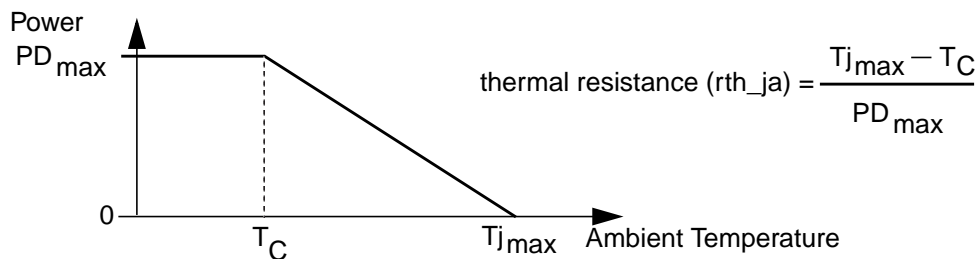
where:

- `tempj` is the internal temperature of a device. In the case of a semiconductor device, T_j is the junction temperature.
- `temp` is the ambient temperature. The Saber simulator uses a default value of 27°C for ambient temperature. "Setting Ambient Temperatures in a Design" on page D-5 describes how to specify a different ambient temperature either for the entire design or for a particular part in the design.
- `pwr` is the power dissipated by the part. This value is calculated in the template (for example, `pwr` for a bipolar transistor).
- `rth_eff` is the thermal resistance specified by a property on the symbol for the part. The method of modeling thermal resistance differs depending on whether a heat sink is present

The following two sections describe how thermal resistance is modeled depending on whether a heat sink is present or not.

Thermal Effects for Parts without Heat Sinks

If the part does not have a heat sink, the thermal resistance is specified by the `rth_ja` property (junction-to-ambient thermal resistance.) The following figure shows how to obtain a value for this property from the power rating curve on the manufacturer's data sheet.



For example, if a resistor is dissipating 1 W at an ambient temperature of 27°C with the junction-to-ambient thermal resistance of $100^\circ\text{C}/\text{W}$, then the internal temperature of the resistor will rise to 127°C .

$$\text{tempj} = 27 + 1 * 100 = 127$$

Thermal Effects for Parts with Heat Sinks

If the part has a heat sink, the thermal resistance is specified by both the `rth_jc` property (junction-to-case thermal resistance) and the `rth_hs` property for (case-to-ambient thermal resistance.) These values are typically provided on the manufacturer's data sheets for the device and the heat sink.

For example, if a resistor is dissipating 1 W at an ambient temperature of 27°C with the junction-to-case thermal resistance of 1 °C/W and the case-to-ambient (heat sink) thermal resistance is 10°C/W, then the internal temperature of the resistor will rise to 38°C.

$$\text{temp}_j = 27 + 1 * (1 + 10) = 38$$

Specifying Thermal Effects Properties

If you want to consider thermal effects during stress analysis, you will need to provide a value for either the thermal resistance `rth_ja` (if no heat sink is present) or the thermal resistance pair `rth_jc` and `rth_hs` (if a heat sink is present).

Step 1. Set the template's ambient temperature.

Follow the procedure in "Setting Ambient Temperatures in a Design" on page D-5.

Step 2. Determine if the template will model a part with or without a heat sink.

Thermal resistances are defined by the symbol properties `rth_ja`, `rth_jc`, and `rth_hs`.

- If the device is operating without a heat sink, you will need to specify `rth_ja`, the thermal resistance from junction to ambient.
- If the device is operating with a heat sink, you will need to specify both `rth_jc`, the thermal resistance from junction to case, and `rth_hs`, the thermal resistance of an external heat sink.

Step 3. Provide values for the thermal resistance symbol properties.

1. Open the Property Editor on the template's symbol instance.
2. Set the Value of each thermal resistance symbol property required by your thermal mounting configuration as determined in Step 2.

Step 4. Include the temperature quantities in the analysis signal list.

For example, if you are trying to determine junction temperature as a function of time, be sure to include `tempj` of the template of interest in the signal list of your transient analysis so that once the analysis is complete you will be able to plot `tempj` in Scope.

"Viewing Signals Internal to a Template" on page 2-6 provides an explanation of what is required to do this step.

Setting Ambient Temperatures in a Design

Step 1. Define the ambient temperature.

Depending upon how accurate you need your simulation results to be, you can set the ambient temperature on each template (locally) or on all of the parts in the design (globally).

The default ambient temperature, both locally and globally, is 27°C.

You might choose to deviate from the default if:

- you have preliminary operating temperatures from thermal analysis
- you have actual temperature measurements from prototype systems
- you are simulating system performance under extremes of temperature
- you are investigating the effects of a circuit board "hot spot" on system performance

Step 2. Determine if you want ambient temperature set locally or globally.

Step 3. Set the `temp` property.

Template instances have a symbol property called `temp` that can be set to the ambient temperature of your choice.

The procedure to set the `temp` property depends upon whether you are setting it locally or globally:

- Local ambient temperature
 - a. Open the Property Editor on the template's symbol instance.

Appendix D: *Stress Analysis Concepts*

b. Set the Value of `temp` to the ambient temperature, in centigrade, you want the template to be simulated at.

- Global ambient temperature

The `temp` flag is also a global variable, meaning that you can specify one ambient temperature on all of the parts in your design. The default global temperature is 27°C.

Set `temp` to the ambient temperature, in centigrade, you want the template to be simulated at on the **SaberInclude** symbol, following the instructions on how to set a global variable in your schematic capture tool.

- You are also able to set the global `temp` property to a value and override it on a template-by-template basis.

Hierarchical Parameter Passing

The netlisters can determine which parameters are to be passed through the hierarchy of the templates they are writing. After determining which parameters are to be passed to an existing template, the netlisters search the schematic that contains the instance of the existing template, and then the schematic containing that schematic, etc., until it finds the parameters it needs or it reaches the most general schematic. Parameters that are found are declared in the templates that correspond to the schematics through which they are passed. Parameters that are not found are declared `external` and can be defined in an included file. An included file is a file that is included in a netlist by using a `SaberInclude` property on a symbol.

If the value of a parameter passed to an existing template is a MAST identifier (argument name), then the design will be searched upward through the instances in the hierarchy for that identifier. If the identifier is found as a property somewhere in the design above, then parameters will be established to pass the property through the hierarchy to the target template. If the identifier is being passed to an instance in the top level of the design, then it will be passed through directly and no declaration will be necessary. If the identifier is not found as a property somewhere in the design above, then it will be declared `external`.

Note that enumerated type values are MAST identifiers. This will normally cause all enumerated type values to be declared `external`. A template argument with enumerated values can only be given a value from a list of values declared in the template. These values typically are words such as YES or NO. If an `enums` section definition is not present, there are two ways that enumerated type values can be passed into a template, one that makes use of the `external` declaration and one that avoids it.

External declarations can be used to pass enumerated type values into a template by initializing a variable with the enumerated type value in a file and including the file in the netlist with a `SaberInclude` property. For example, if you wish to pass the enumerated type value `_0` to the `init` parameter of the `inv_14` template in a schematic other than the most general

Appendix E: Hierarchical Parameter Passing

schematic of the design, then you will need to pass some other value instead, such as `inv_init`, which will be declared in the netlist as shown below:

```
external inv_l4..init inv_init
```

You need to then define `inv_init` in a file such as `enuminit.sin` and cause it to be included at the top level of the schematic with the `SaberInclude` property. The contents of this file should contain the following along with any other initialization that you need for enumerated types in your design:

```
inv_l4..init inv_init = _0
```

If you wish to pass more than just a single identifier to a parameter of an existing template, then you must enclose in braces any parameters that are to be passed through hierarchy to this parameter. For example, if you wish to use the `Saber_rnom` property to pass the sum of existing properties `res1` and `res2` to the `r.r1` instance of the `r` template, then the `Saber_rnom` property must have the value `{res1}+{res2}` and the netlist will contain:

```
r..rnom res1
r..rnom res2
...
r.r1 p:node1 m:node2 = rnom=res1+res2
```

If `res1` does not exist as a property in the hierarchy above this instance of the `r` template, then it will be declared `external` and must be defined in an included file such as was done above.

When a template is created from a schematic that will have parameters passed through it, default parameter value declarations are made only if the symbol for the schematic has a property with same name as the parameter. If it does not have such a property, then all instances of that symbol in the design must have such a property.

Symbols

.guidecfg file [A-5](#)
.guideRc_user file [A-7](#)
.scopecfg file [A-5](#)
.scopecfg_data directory [A-6](#)
.scopeRc_user file [A-7](#)
.sketchcfg file [A-5](#)
.sketchRc_user file [A-7](#)

A

AC analysis
 algorithm overview [8-5](#)
 defining frequency range [8-3](#)
 defining increment function [8-3](#)
 defining number of data points [8-3](#)
 definition [8-1](#)
 determine next step [8-11](#)
 displaying the form [8-3](#)
 measuring the results [8-8](#)
 output files [8-5](#)
 overview [8-1](#)
 performing [8-2](#)
 resolving design problems [8-11](#)
 specifying initial point [8-3](#)
 viewing results [8-6](#)
AI_SCH_PATH environment
 variable [1-7](#)
aim.site file [A-3](#)
ALG_NO_SOLUTION [4-6](#)
altering parameters [3-8](#)
Amplitude Ratio field [10-11](#)

B

back-annotating [3-9](#)
Bode plot [8-1](#)
 measuring results [8-8](#)

 viewing results [8-6](#)

borders
 adding [1-31](#)
bundles [1-25](#)
bus attributes [1-29](#)
busses
 designating bus width [1-29](#)
 drawing [1-26](#)
 naming [1-29](#)

C

c_tol property [12-2](#)
calibrating analyses
 for DC analysis [9-2](#)
 for DC transfer [9-3](#)
 for fourier analyses [9-6](#)
 for small signal analyses [9-5](#)
 for transient analysis [9-3](#)
 process overview [9-1](#)
Capture Process [1-1](#)
chain ABCD [10-3](#)
collecting stress data [13-1](#)
composite property, symbol [1-20](#)
compression [10-9](#)
Compute Desensitization field [10-11](#)
configuration parameters, setting [A-5](#)
Contributors
 distortion [10-11](#)
 noise [10-6](#)

D

Data files
 using in stress analysis [13-4](#)
Data Files field
 DC Transfer [6-2](#)
 Transient analysis [5-4](#), [8-5](#)
DC analysis

Index

- algorithm overview 4-3
 - displaying results 4-4
 - executing the analysis 4-3
 - finding hard operating points 4-1
 - output files 4-3
 - running multiple iterations 4-3
 - using holdnodes 4-8
 - DC analysis, back-annotating values 3-9
 - DC Sweep analysis
 - see DC Transfer analysis
 - DC Transfer
 - algorithm overview 6-3
 - command example 6-3
 - displaying results 6-4
 - evaluating results 6-6
 - running multiple iterations 6-2
 - specifying required fields 6-1
 - DC Transfer analysis
 - definition 6-1
 - performing 6-1
 - specifying optional fields 6-2
 - dc_err file
 - displaying 4-5
 - dcerr file 4-7
 - Debug field 4-2
 - defining signals to plot 5-3, 8-4
 - deleting measurements 5-10
 - derating curve examples 13-4
 - derating factor 13-3
 - desensitization distortion
 - product 10-9
 - Display Initial Point Error field 4-5
 - Distortion analysis
 - analyzing the results 10-13
 - compared to fourier analysis 5-12
 - defining frequency range 10-10
 - defining input/output files 10-12
 - executing the analysis 10-12
 - harmonic distortion in fourier analysis 5-15
 - overview 10-9
 - process 10-9
 - specifying contributors 10-11
 - specifying optional distortion products 10-10
 - specifying output signals 10-10
 - viewing results 8-6
 - viewing the results 10-12
 - Distortion Contributors field 10-11
- ## E
- editing the initial point 7-3
 - End Frequency field 8-3
 - End Point File field (dc) 4-3
 - End Time field 5-2, 14-11
- ## F
- FFT analysis
 - compared to distortion analysis 5-12
 - executing the analysis 5-17, 5-19
 - overview 5-17
 - specifying input time segment 5-18
 - specifying number of points 5-18
 - specifying signals to transform 5-17
 - specifying source transient data 5-18
 - specifying windowing functions 5-18
 - viewing the results 5-19
 - finding parts 1-6
 - flicker noise 10-5
 - fourier analyses overview 5-12
 - Fourier analysis
 - analyzing the results 5-16
 - compared to distortion analysis 5-12
 - executing the analysis 5-16
 - overview 5-13
 - plot file contents 5-16
 - specifying harmonic calculations 5-15
 - specifying signals to transform 5-14
 - specifying the data file 5-14
 - specifying the fundamental frequency 5-14

viewing the results 5-16
frequency response 8-1

G

Ground Symbol 1-10
guide.log file A-7
guide.site file A-3
guideRc.site file A-7

H

harmonic distortion 10-9
harness.log file A-7
heat sinks, thermal effects of D-4
hierarchical symbol, creating 1-30
holdnodes 4-8
hybrid parameters 10-3

I

iFFT analysis
 executing the analysis 8-13
 overview 8-12
 specifying data 8-12
 specifying signals to
 transform 8-12
 specifying source plot file 8-12
 viewing the results 8-13, 10-4
Increment Type field 8-3
Independent Source field 6-1
Initial Point files
 editing values 7-3
 evaluating 4-5
 using in stress analysis 13-4
Input File From field
 Operating Point Report form 4-4
instance symbol 1-5
intermodulation distortion 10-9
intermodulation distortion products
 specifying 10-10
Invoking
 Sketch 1-2

L

`l_tol` 12-2

large signal analyses 5-1

M

manipulating measurements 5-10
mapping symbols 2-5
Measurement tool
 frequency measures 8-9
 manipulating measurements 5-10
 setting data range 5-10
measuring AC analysis results 8-8
measuring the results 14-7
Measuring transient results 5-8
modeling noise 10-5
models, choosing 1-3
Monitor field 4-2
Monte Carlo 12-1
 adding analyses 12-4
 adding post-processing
 functions 12-5
 continuing the analysis 12-8
 executing 12-3
 moving elements in the loops 12-5
 overview 12-1
 saving parameter values 12-4
 viewing the results 12-6
Moving Symbol Instances 1-7
multi-segment waveforms 11-4

N

Naming instances 1-8
net name restrictions 1-24
netlist, altering, parameter,
 altering 3-8
netlist, listing 3-8
Node Value List field 7-3
Noise analysis
 analyzing the results 10-8
 definition 10-4
 executing the analysis 10-4, 10-7
 nons 10-6
 specify noise contributors 10-6
 specifying frequency range 10-5
 specifying output signals 10-5
 types of noise 10-4

Index

- viewing results 8-6
- viewing the results 10-7
- Non-linear systems
 - small signal model 8-1, 10-1
- nons 10-6
- Number of Points field 5-18, 8-3, 14-14

O

- onpage connectors 1-25
- open-circuit impedance 10-3
- Opening designs 1-2
- Opening Schematic 1-2, 1-3, 2-2
- operating point
 - definition 4-1
 - purpose 4-1
- Operating Point Report 4-4
- oscillators
 - editing the initial point 7-3
 - starting transient analysis 7-1

P

- Parameter sweep
 - adding analyses 11-2
 - adding post-processing functions 11-3
 - adding vary and monte carlo loops 11-2, 12-4
 - executing the analysis 11-3
 - moving elements in the loops 11-3
 - overview 11-1
 - specifying the parameters 11-1
 - types of sweeps 11-2
 - viewing the results 11-4
- parameter tolerances, specifying 12-1
- Parts Gallery 1-5
- passing parameters 1-18
- performance measure 11-7
- PinFaultEditor 15-11
- placing symbols 1-4
- Plot files
 - viewing Plot files 5-7, 6-4, 8-6, 14-5
- Plot Files field
 - DC Transfer 6-2
 - Pole-Zero analysis 14-3

- Transient analysis 5-3, 8-4
- plotting signals 5-8, 6-5, 8-8, 14-6
- Pole-Zero analysis 14-7
- ports, adding 2-3
- primitive property 1-13, 2-6
- probe 3-7
- probes 3-5
- project.site file A-3
- properties 1-13
 - rnom 1-16
 - saber_model 1-16
- properties, changing values in sketch 3-8
- properties, finding help 1-15
- properties, modifying 1-14
- Properties, required 1-15
- properties, specifying global 1-17
- PZ analysis
 - determine next step 14-9

R

- r_tol 12-2
- ref property 1-8, 2-6
- restrictions, net names 1-24
- reverse chain ABCD 10-3
- rnom property 1-16
- Run DC analysis First? field 5-1, 8-2, 8-3, 14-2, 14-3, 14-10
- Run PZ analysis First? field 14-13
- running multiple iterations 6-2

S

- Saber Include File symbol 13-2
- Saber Node 0 1-10
- Saber Parts
 - Accessing 1-5
- saber symbol 1-17
- Saber, exiting 3-10
- saber_model property 1-13
- saber_model property 1-16
- SaberSketch
 - Symbol, saving 2-8
- saving designs 1-33
- saving transient results 5-3, 8-4

- scattering [10-3](#)
 - schematic blocks [1-2](#)
 - schematic property [2-5](#)
 - schematic, drawing a Sketch
 - borders [1-31](#)
 - Schematic, Opening [1-2, 1-3, 2-2](#)
 - schtos command [B-3](#)
 - scope.log file [A-7](#)
 - scope.site file [A-3](#)
 - scopeRc.site file [A-7](#)
 - searching for parts [1-6](#)
 - seeds, Monte Carlo [12-3](#)
 - Sensitivity analysis
 - evaluating the results [11-11](#)
 - generating the report [11-10](#)
 - overview [11-1, 11-6](#)
 - performing nominal
 - simulation [11-7](#)
 - specifying analyses [11-9](#)
 - specifying parameters [11-8](#)
 - specifying percentage to vary
 - perturbation [11-9](#)
 - short-circuit admittance [10-3](#)
 - shot noise [10-4](#)
 - showing measurements [5-10](#)
 - Signal List field
 - Transient analysis [5-3, 8-4](#)
 - value recommendations [5-3, 8-4](#)
 - signal list, adding nodes from
 - Sketch [3-5](#)
 - Sigset command [7-3](#)
 - Sketch
 - Invoking [1-2](#)
 - Nets, drawing [1-22](#)
 - Symbols, placing on sheet [1-5](#)
 - sketch.log file [A-7](#)
 - sketch.site file [A-3](#)
 - sketchRc.site file [A-7](#)
 - Small signal analyses
 - AC analysis [8-1](#)
 - available types [8-1, 10-1](#)
 - defining frequency ranges [8-3](#)
 - definition [8-1](#)
 - defintion [10-1](#)
 - measurement summary [8-9](#)
 - Noise analysis [10-4](#)
 - output files [8-5](#)
 - specifying operating point [8-3](#)
 - viewing results [8-6](#)
 - specifying tolerances [12-1](#)
 - Start Frequency field [8-3](#)
 - Start Time field [5-2, 14-11](#)
 - Statistical analysis
 - See Monte Carlo
 - Stress analysis
 - analyzing the results [13-7](#)
 - performing [13-5](#)
 - performing nominal
 - simulation [13-4](#)
 - process overview [13-1](#)
 - restricting input data [13-5](#)
 - specifying derating values [13-3](#)
 - specifying stress ratings [13-2](#)
 - specifying the derating file [13-6](#)
 - specifying the report [13-5](#)
 - specifying thermal effects [D-4](#)
 - specifying which measures to
 - calculate [13-6](#)
 - viewing the report [13-6](#)
 - stress measure [13-1](#)
 - stress ratings
 - order of precedence [13-3](#)
 - overriding [13-3](#)
 - specifying [13-2](#)
 - Sweep Range field
 - DC Transfer [6-2](#)
 - Symbol Definition Properties [1-14](#)
 - Symbol Instance Properties [1-14](#)
 - symbol, adding graphics [2-2](#)
 - symbol, adding ports [2-3](#)
 - symbol, adding views [2-3](#)
 - symbol, mapping [2-5](#)
 - symbols, placing [1-4](#)
- T**
- Testify
 - debugging fault runs [15-9](#)
 - fault wrappers [15-13](#)
 - introduction [15-1](#)
 - PinFault Editor [15-11](#)
 - process steps [15-1](#)

Index

- using with hierarchy 15-11
- thermal effects, specifying D-4
- thermal noise 10-4
- Time Step field
 - Transient analysis 5-2
 - value recommendation 5-2
- Total Harmonic Distortion field 5-15
- transfer functions 10-3
- Transient analysis
 - defining signals to plot 5-3, 8-4
 - determining next step 5-11
 - executing the analysis 5-5
 - filling out the form 5-2
 - measuring results 5-8
 - performing 5-2
 - prerequisite 5-1
 - starting oscillators 5-2
 - storing results 5-3, 8-4
 - viewing the frequency spectrum 5-13
- Two-port analysis
 - executing the analysis 10-4
 - overview 10-2
 - performing 10-2
 - types of parameters 10-3

V

- Vary
 - See Parameter sweep
- viewing plot files 5-7, 6-4, 8-6, 14-5
- viewing transient results 5-6
- viewing waveforms 3-5
- views, symbol 2-3

W

- windowing functions 5-15
- wire attributes 1-24
- wires
 - drawing 1-22
 - methods of connecting 1-25
 - naming 1-24
 - using bundles 1-25