

Virtuoso[®] Analog Design Environment L User Guide

**Product Version 6.1.6
November 2014**

© 1999–2014 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

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

Contents

<u>Preface</u>	23
<u>Scope of this Guide</u>	24
<u>Licensing in ADE L</u>	24
<u>Related Documents for ADE L</u>	24
<u>Installation, Environment, and Infrastructure</u>	24
<u>Technology Information</u>	25
<u>Virtuoso Tools</u>	25
<u>Third Party Tools</u>	26
<u>Typographic and Syntax Conventions</u>	26
<u>SKILL Syntax Examples</u>	27
<u>Form Examples</u>	28
<u>Additional Learning Resources</u>	28
<u>Help and Support Facilities</u>	29

1

<u>Features of the Virtuoso Analog Design Environment L</u>	31
<u>Consistent User Interface</u>	31
<u>Menu Access Keys</u>	32
<u>Analog Design Entry</u>	32
<u>Design Hierarchy</u>	32
<u>Annotation</u>	33
<u>Interactive Simulation</u>	33
<u>Important Benefits of Direct Simulation</u>	33
<u>Important Use-Model Differences between spectreS and spectre</u>	34
<u>Simulation Output and Analysis</u>	35
<u>Advanced Analysis</u>	35

2

<u>Environment Setup</u>	37
<u>About the Simulation Window</u>	37

Virtuoso Analog Design Environment L User Guide

<u>Opening the Simulation Window</u>	38
<u>The Simulation Window GUI</u>	40
<u>ADE L Toolbars</u>	44
<u>Choosing the Design</u>	47
<u>Choosing a Simulator</u>	48
<u>Migrating Socket Libraries to Direct Simulators</u>	50
<u>Setting the Simulation Temperature</u>	53
<u>Setting the Model Path</u>	53
<u>Choosing a User Interface Path</u>	54
<u>Using the Simulation Window</u>	55
<u>Using the Schematic Window</u>	55
<u>Simulator Interfaces</u>	57
<u>Spectre Simulator</u>	57
<u>Virtuoso Accelerated Parallel Simulator</u>	58
<u>Virtuoso UltraSim Simulator Interface</u>	58
<u>Virtuoso AMS Designer Simulator Interface</u>	60
<u>Mixed-Signal Simulators (IC6.1.6 only)</u>	66
<u>Hspice Direct Interface</u>	66
<u>Setting Up Simulation Files</u>	67
<u>Setting Up Include Paths</u>	69
<u>Setting Up Definition Files</u>	69
<u>Setting Up Stimulus Files</u>	70
<u>Setting Up Vector Files</u>	70
<u>Setting Up VCD and EVCD Files</u>	71
<u>Specifying Simulation Files Using Variables</u>	72
<u>Using Shell Environment Variables to Specify the Paths</u>	73
<u>Enabling and Disabling Simulation Files</u>	73
<u>Editing Simulation Files</u>	74
<u>Deleting Simulation Files</u>	74
<u>Setting Simulation Environment Options</u>	74
<u>Setting Simulation Environment Options for Direct Simulation</u>	74
<u>Setting Environment Options for AMS</u>	77
<u>Setting Up a Remote Simulation</u>	95
<u>Using a Third-Party Simulator for Remote Simulations</u>	96
<u>Scripts for Using Third-Party Simulators in Remote Simulations</u>	96
<u>Saving and Restoring the Simulation Setup</u>	97

Virtuoso Analog Design Environment L User Guide

<u>Saving the Simulation Setup</u>	98
<u>Restoring the Simulation Setup</u>	100
<u>Panic State Saving</u>	103
<u>Closing a Session</u>	104
<u>Saving a Script</u>	105
<u>Running the Simulation Setup in Advanced Simulation</u>	
<u>Environments (ADE XL / ADE GXL)</u>	106
<u>Opening ADE XL / ADE GXL using the Launch Menu</u>	107
<u>Creating a New ADE (G)XL View</u>	108
<u>Opening an Existing ADE (G)XL View</u>	111
<u>Customizing the Opening of ADE XL or ADE GXL from ADE L</u>	114
<u>Configuring the Analog Design Environment</u>	114
<u>Resetting the Default Environment</u>	115
<u>Setting Basic Session Defaults</u>	115
<u>Variables for Customizing Netlist Generation</u>	116
<u>Customizing Your .cdsinit File</u>	117
<u>Customizing Your .cdsenv File</u>	117
<u>Customizing Your Menus File</u>	117
<u>Setting UNIX Environment Variables</u>	118
<u>Reserved Words</u>	119
<u>Bindkeys</u>	120
<u>Checking Bindkey Assignments</u>	120
<u>Assigning Bindkeys</u>	121
<u>Using the Key or Mouse Binding Form</u>	121
<u>Using the CIW</u>	122
<u>Using Your .cdsinit File</u>	122
<u>Form Field Descriptions</u>	123
<u>Choosing Simulator/Directory/Host</u>	123
<u>Create New File</u>	124
<u>Setting Model Path</u>	124
<u>Model Library Setup</u>	126
<u>Environment Options</u>	128
<u>Saving State</u>	130
<u>Loading State</u>	132
<u>Editing Session Options</u>	134

3

Design Variables and Simulation Files for Direct Simulation

<u>Design Variables and Simulation</u>	137
<u>Using Direct Simulation</u>	137
<u>Design Variables and Simulation</u>	138
<u>Setting Values</u>	138
<u>Adding a New Variable</u>	139
<u>Changing Values</u>	140
<u>Deleting Values</u>	141
<u>Saving Variable Values</u>	142
<u>Restoring Variable Values</u>	142
<u>Copying Values between the Schematic and the Simulation Environment</u>	142
<u>Displaying Values on the Schematic</u>	143
<u>Adding Setup Files for Direct Simulation</u>	143
<u>Using a Definitions File</u>	144
<u>Syntax</u>	144
<u>Definition File Example</u>	145
<u>Stimuli Setup</u>	145
<u>Using the Setup Analog Stimuli Form</u>	146
<u>Specifying a Stimulus File</u>	151
<u>Example of a spectre Stimulus File</u>	151
<u>Model Files in the Virtuoso Analog Design Environment</u>	152
<u>Model File Libraries</u>	152
<u>Referencing Textual Subcircuits or Models</u>	153
<u>Updating the Component CDF</u>	153
<u>Creating a Stopping Cellview</u>	154
<u>Using the Component</u>	154
<u>Including the Subcircuit File in the Netlist</u>	155
<u>Scope of Parameters</u>	155
<u>Inheriting from the Same Instance: iPar()</u>	155
<u>Passed Parameter Value of One Level Higher: pPar()</u>	156
<u>Passed Parameters from Any Higher Level: atPar()</u>	157
<u>Inheriting from the Instance Being Netlisted: dotPar()</u>	157
<u>Table of Functions</u>	158
<u>Nesting Functions</u>	158

<u>Using Inheritance Functions in Input Files</u>	159
<u>How the Netlister Expands Hierarchy</u>	159
<u>Netlisting Sample for Spectre</u>	161
<u>Modifying View Lists and Stop Lists</u>	161
<u>About Netlists</u>	163
<u>The .simrc File</u>	163
<u>Incremental Netlisting</u>	164
<u>Creating and Displaying a Netlist</u>	164
<u>Form Field Descriptions</u>	165
<u>Setup Analog Stimuli Form</u>	165
<u>Editing Design Variables</u>	167

4

<u>Setting Up for an Analysis</u>	169
<u>Required Symbol</u>	169
<u>Setting Up with Different Simulators</u>	170
<u>Deleting an Analysis</u>	170
<u>Enabling or Disabling an Analysis</u>	170
<u>Specifying Order for Analyses</u>	171
<u>Saving the Analysis Setup</u>	172
<u>Restoring a Saved Analysis Setup</u>	172
<u>Setting Up a Spectre Analysis</u>	174
<u>Transient Analysis</u>	174
<u>Transient Noise Analysis</u>	180
<u>DC Analysis</u>	188
<u>AC Small-Signal Analysis</u>	194
<u>Noise Analysis</u>	198
<u>S-Parameter Analysis</u>	201
<u>Transfer Function Analysis</u>	205
<u>Sensitivity Analysis</u>	206
<u>DC Mismatch Analysis</u>	209
<u>Stability Analysis</u>	215
<u>Pole Zero Analysis</u>	218
<u>Other Spectre Analyses</u>	223
<u>Setting Up an UltraSim Analysis</u>	223

<u>Running Advanced Analysis Simulations</u>	227
<u>Setting Up an AMS Analysis</u>	246
<u>Setting Up an HspiceD Analysis</u>	255
<u>Setting Up EM/IR Analysis</u>	258

5

<u>Selecting Data to Save and Plot</u>	261
<u>About the Saved and Plotted Sets of Outputs</u>	261
<u>Opening the Setting Outputs Form</u>	262
<u>Deciding which Outputs to Save</u>	263
<u>Saving All Voltages or Currents</u>	263
<u>Saving Outputs for UltraSim Simulations</u>	265
<u>Saving Selected Voltages or Currents</u>	269
<u>Saving or Plotting Selected Voltages or Currents for AMS Simulation</u>	269
<u>Adding a Node or Terminal to a Set</u>	273
<u>Adding a Saved Node to the Plot Set</u>	274
<u>Removing Nodes and Terminals from a Set</u>	275
<u>Saving a List of Outputs</u>	275
<u>Restoring a Saved List of Outputs</u>	276
<u>Specifying Hierarchy Levels to Save Outputs</u>	276
<u>Importing and Exporting Outputs in ADE L Environment</u>	278
<u>CSV File Format</u>	278
<u>Exporting Outputs to a CSV File</u>	279
<u>Importing Output to ADE L</u>	281
<u>Conditional Search for Results</u>	285
<u>Form Field Descriptions</u>	288
<u>Circuit Conditions</u>	288
<u>Setting Outputs</u>	291
<u>Save Options and Keep Options</u>	292
<u>Environment Variables for PSFXL Output Format</u>	295
<u>PSFXL Environment Variables for Site Administrators</u>	298

6

<u>Running a Simulation</u>	301
<u>Prerequisites to Simulation</u>	301

Virtuoso Analog Design Environment L User Guide

<u>Setting Simulator Options</u>	302
<u>Spectre Options</u>	303
<u>Specifying AMS Spectre High Performance/Parasitic Reduction Options</u>	309
<u>UltraSim Options</u>	310
<u>AMS Options</u>	325
<u>HspiceD Options</u>	351
<u>About the OSS-based AMS Netlister</u>	353
<u>Benefits of OSS-based AMS Netlister</u>	354
<u>Things to Know When Using the OSS-based Netlister</u>	355
<u>Creating Simulation Scripts for irun</u>	360
<u>Updating Text Views that do not have Virtuoso Database Information</u> <u>Using the Update Text Views Form</u>	361
<u>Other Methods for Updating Text Views that do not have Virtuoso</u> <u>Database Information</u>	363
<u>Choosing the AMS Netlister</u>	365
<u>Saving and Restarting an AMS Designer Simulation Run</u>	368
<u>Simulating the Design Using the irun Command</u>	376
<u>Starting a Simulation</u>	378
<u>Interrupting or Stopping a Simulation</u>	380
<u>Updating Variables and Resimulating</u>	380
<u>Saving Simulator Option Settings</u>	381
<u>Restoring Saved Settings</u>	381
<u>Viewing the Simulation Output</u>	382
<u>Viewing the Output Log for AMS</u>	387
<u>Viewing the Error Explanation for AMS</u>	388
<u>Using the SimVision Debugger</u>	389
<u>Display Partition</u>	393
<u>Default Digital Discipline Selection</u>	397
<u>Running a Parametric Analysis</u>	402
<u>Device Checking</u>	403
<u>Enabling and Disabling Device Checking</u>	403
<u>Setting Up Device Checks</u>	404
<u>Specifying Global Device Check Options</u>	417
<u>Specifying Options for Writing Violations Information</u>	425
<u>Viewing, Printing, and Saving Device Check Violations</u>	426

7

<u>Parameterization Support</u>	433
<u>About Parameterization Support</u>	433
<u>Support for VAR Syntax</u>	433
<u>Usage of VAR Syntax</u>	434
<u>ADE Forms for VAR Support</u>	435
<u>Setup Examples</u>	436
<u>Model File Setup Example</u>	436
<u>Transient Analysis Setup Example</u>	437
<u>Running a Sweep Analysis using VAR()</u>	437
<u>Switch View List Setup Example</u>	438

8

<u>Helping a Simulation to Converge</u>	441
<u>Commands for Forcing Convergence</u>	441
<u>Node Set</u>	442
<u>Initial Conditions</u>	442
<u>Force Node</u>	442
<u>HspiceD Convergence Aids</u>	443
<u>Selecting Nodes and Setting their Values</u>	443
<u>Releasing Voltages</u>	445
<u>Changing Voltages</u>	445
<u>Saving and Restoring Node Voltages</u>	446
<u>Highlighting Set Nodes</u>	447
<u>Storing a Solution</u>	447
<u>Restoring a Solution for Spectre</u>	448
<u>Form Field Descriptions</u>	450
<u>Store/Restore File</u>	450

9

<u>Analysis Tools</u>	451
<u>Parametric Analysis</u>	451
<u>Using the Parametric Analysis Tool</u>	451
<u>Parametric Analysis Window</u>	452

Virtuoso Analog Design Environment L User Guide

<u>Modes of Parametric Analysis</u>	454
<u>Specifying Variables for Sweeps & Ranges Run Mode</u>	455
<u>Specifying Parameters for Parametric Set Run Mode</u>	462
<u>Deleting Sweep Specifications</u>	463
<u>Reviewing Sweep Specifications</u>	464
<u>Running the Parametric Analysis</u>	465
<u>Stopping the Parametric Analysis</u>	467
<u>Saving and Retrieving Specification Details</u>	468
<u>UltraSim Power Network Solver</u>	470
<u>UltraSim Interactive Simulation Debugging</u>	474

10

<u>Plotting and Printing</u>	477
<u>Overview of Plotting</u>	477
<u>Setting Plotting Mode</u>	479
<u>Refreshing Graphs</u>	480
<u>Setting Plotting and Display Options</u>	488
<u>Saving and Restoring the Window Setup</u>	488
<u>Using the Plot Outputs Commands</u>	489
<u>Plotting the Current or Restored Results</u>	489
<u>Removing Nodes and Terminals from the Plot List</u>	490
<u>Plotting Parasitic Simulation Results</u>	490
<u>Using the Direct Plot Commands</u>	491
<u>For Noise Figures</u>	493
<u>For Transfer Functions</u>	496
<u>For S-Parameters</u>	498
<u>Using the Direct Plot Main Form</u>	501
<u>For DC</u>	501
<u>For AC</u>	503
<u>For Transient Results</u>	505
<u>For Stability Results</u>	507
<u>For Pole Zero Results</u>	511
<u>Overview of Printing</u>	514
<u>Printing Results</u>	515
<u>Saving State</u>	516

Virtuoso Analog Design Environment L User Guide

<u>Loading State</u>	517
<u>Updating Results</u>	517
<u>Making a Window Active</u>	518
<u>Editing Expressions</u>	518
<u>Setting Display Options</u>	520
<u>Displaying Output Information</u>	521
<u>Specifying Results to Print</u>	521
<u>Printing DC Operating Points</u>	522
<u>Printing Transient Operating Points</u>	522
<u>Printing Model Parameters of Components</u>	522
<u>Printing Noise Parameters of Nodes or Components</u>	523
<u>Printing DC Mismatch Summary</u>	527
<u>Printing Stability Summary</u>	529
<u>Printing DC Node Voltages</u>	532
<u>Printing Transient Voltages</u>	533
<u>Printing Sensitivities</u>	533
<u>Precision Control for Printing</u>	533
<u>Printing Capacitance Data</u>	534
<u>Printing Statistical Reports or Calculator Results</u>	536
<u>Using SKILL to Display Tabular Data</u>	537
<u>Overview of Plotting Calculator Expressions</u>	538
<u>Defining Expressions</u>	538
<u>Creating Dependent Expressions</u>	540
<u>Plotting Expressions</u>	544
<u>Suppressing Plotting of an Expression</u>	544
<u>Viewing and saving Results</u>	545
<u>Saving Simulation Results</u>	545
<u>Deleting Simulation Results</u>	545
<u>Browsing Results Directories</u>	546
<u>Restoring Saved Results</u>	547
<u>Annotating Simulation Results</u>	549
<u>Annotating Transient Voltages</u>	549
<u>Annotating Transient Currents</u>	550
<u>Annotating Transient Operating Points</u>	550
<u>Specifying the Data Directory for Labels</u>	551
<u>Saving and Removing Annotated Labels</u>	553

Virtuoso Analog Design Environment L User Guide

<u>Annotating Parametric Sweep Results</u>	554
<u>Plotting Results of a Parametric Analysis</u>	556
<u>Form Field Descriptions</u>	558
<u>Setting Plotting Options</u>	558
<u>XF Results</u>	560
<u>S-Parameter Results</u>	561
<u>Setting Outputs</u>	564
<u>Noise Summary</u>	565
<u>Save Results</u>	566
<u>Select Results</u>	566
<u>Delete Results</u>	567
<u>UNIX Browser</u>	567

11

Hspice Direct Support

<u>Introduction to Hspice Direct Simulator</u>	569
<u>Libraries</u>	571
<u>Features</u>	573
<u>Model Libraries</u>	573
<u>Distributed Processing Support</u>	573
<u>Running Analyses</u>	573
<u>Passing Command Line Options</u>	574
<u>Analog Options</u>	575
<u>Output Log</u>	577
<u>Convergence Aids</u>	577
<u>Results</u>	578
<u>Converting Libraries</u>	579
<u>Control Mapping of Nets</u>	581

12

UltraSimVerilog

<u>Interface Element Macro Models</u>	583
<u>Inline Subcircuit</u>	584
<u>Interface Element Selection Rules</u>	584
<u>Simulation Accuracy and Performance</u>	584

<u>Analog-to-Digital (A2D) Models</u>	585
<u>Digital-to-Analog (D2A) Models</u>	586
<u>Netlisting Options</u>	588
<u>Verilog Netlisting Options</u>	589
<u>Hierarchical Netlisting</u>	590
<u>Running a Mixed Signal Simulation</u>	590
<u>Setting Simulator Options</u>	591
<u>Input Stimulus for HNL</u>	595
<u>Setting Design Variables</u>	601
<u>Choosing Analyses</u>	602
<u>Running the Simulation</u>	603
<u>Control and Debugging</u>	603
<u>Viewing and Analyzing Simulation Output</u>	603

13

<u>Using the Reliability Simulator Interface</u>	605
<u>Introduction</u>	606
<u>Specifying Reliability Options</u>	606
<u>Running the Reliability Simulation</u>	608
<u>Viewing the Reliability Simulation Results</u>	608
<u>Device Lifetime and Degradation Results</u>	608
<u>Device Characteristic Degradation Results</u>	609
<u>Model Parameter Changes Results</u>	610
<u>TMI-aging Results</u>	611
<u>Viewing the Reliability Aged Netlist</u>	612
<u>Annotating Simulation Results to the Schematic</u>	612
<u>Reliability Options</u>	614
<u>Reliability Results Display</u>	633

A

<u>Environment Variables</u>	637
<u>Calculator</u>	638
<u>mode</u>	638
<u>uimode</u>	638
<u>eval</u>	639

Virtuoso Analog Design Environment L User Guide

<u>dstack</u>	639
<u>Reliability Analysis</u>	640
<u>spectre_analysis_reliability</u>	640
<u>relxpert_gradual_aging</u>	640
<u>Distributed Processing</u>	641
<u>autoJobSubmit</u>	641
<u>showMessages</u>	641
<u>queueName</u>	642
<u>hostName</u>	642
<u>startTime</u>	643
<u>startDay</u>	643
<u>expTime</u>	644
<u>externalServer</u>	644
<u>expDay</u>	645
<u>timeLimit</u>	645
<u>emailNotify</u>	646
<u>mailTo</u>	646
<u>logsInEmail</u>	647
<u>stateFile</u>	647
<u>daysBeforeExpire</u>	648
<u>block</u>	648
<u>copyMode</u>	649
<u>copyModeDir</u>	649
<u>loginShell</u>	650
<u>numOfTasks</u>	650
<u>jobArgsInOceanScript</u>	651
<u>puttogetherqueue</u>	651
<u>copyNetlist</u>	652
<u>mailAllLogs</u>	652
<u>drmsCommandList</u>	653
<u>setupFunction</u>	654
<u>Spectre</u>	657
<u>ac_severity</u>	657
<u>assert_severity_default</u>	657
<u>dc_severity</u>	658
<u>dcOp_severity</u>	658

Virtuoso Analog Design Environment L User Guide

<u>tran_severity</u>	659
<u>printCdfParamForTopCell</u>	659
<u>emirSumList</u>	660
<u>emTechFile</u>	660
<u>emirEnable</u>	661
<u>nportirfiledir</u>	661
<u>save</u>	662
<u>outputParamInfo</u>	663
<u>modelParamInfo</u>	663
<u>pwr</u>	664
<u>useprobes</u>	664
<u>subcktprobelvl</u>	665
<u>nestlvl</u>	665
<u>elementinfo</u>	666
<u>saveahdlvars</u>	666
<u>currents</u>	667
<u>setEngNotation</u>	667
<u>switchViewList</u>	668
<u>stopViewList</u>	668
<u>autoDisplay</u>	669
<u>stimulusFile</u>	669
<u>includePath</u>	670
<u>modelFiles</u>	670
<u>analysisOrder</u>	671
<u>paramRangeCheckFile</u>	671
<u>printComments</u>	672
<u>definitionFiles</u>	674
<u>enableArclength</u>	674
<u>useAltergroup</u>	675
<u>netlistBBox</u>	675
<u>autoDisplayBBox</u>	676
<u>includeStyle</u>	676
<u>simExecName</u>	677
<u>checkpoint</u>	677
<u>recover</u>	678
<u>firstRun</u>	678

Virtuoso Analog Design Environment L User Guide

<u>simOutputFormat</u>	679
<u>fastViewOption</u>	680
<u>controlMode</u>	681
<u>licQueueTimeOut</u>	682
<u>licQueueSleep</u>	682
<u>licQueueToken</u>	683
<u>ignorePortOrderMismatch</u>	683
<u>dochecklimit</u>	684
<u>ADE Simulation Environment</u>	685
<u>defaultHierSave</u>	685
<u>retainStateSettings</u>	685
<u>retainDesignVarNotation</u>	687
<u>userNamePrefix</u>	688
<u>showConvertNotifyDialog</u>	688
<u>saveDir</u>	689
<u>saveAsCellview</u>	689
<u>stateName</u>	690
<u>stateOverWriteSkipList</u>	690
<u>allowAdePanicStateSaving</u>	691
<u>adePanicStatePath</u>	691
<u>designEditMode</u>	692
<u>schematicBased</u>	692
<u>windowBased</u>	693
<u>saveAsCellview</u>	693
<u>saveQuery</u>	694
<u>adelExitQuery</u>	694
<u>x</u>	695
<u>y</u>	695
<u>simulator</u>	696
<u>projectDir</u>	697
<u>appendLibNameToProjectDir</u>	697
<u>hostMode</u>	698
<u>host</u>	698
<u>digitalHostMode</u>	699
<u>digitalHost</u>	699
<u>remoteDir</u>	700

Virtuoso Analog Design Environment L User Guide

<u>autoPlot</u>	700
<u>artistPlottingMode</u>	701
<u>directPlotPlottingMode</u>	701
<u>designName</u>	702
<u>drIBufferMemory</u>	702
<u>simulationDate</u>	703
<u>temperature</u>	703
<u>variables</u>	704
<u>designVarSetting</u>	704
<u>scalarOutputs</u>	705
<u>icons</u>	705
<u>resizeMode</u>	706
<u>x</u>	706
<u>y</u>	707
<u>immediatePlot</u>	707
<u>immediatePrint</u>	708
<u>useDisplayDrf</u>	708
<u>preSaveOceanScript</u>	709
<u>postSaveOceanScript</u>	710
<u>noSimLogInOCEAN</u>	710
<u>numberOfSavedRuns</u>	711
<u>browserCenterMode</u>	712
<u>outputsImportExportVersion</u>	712
<u>updateCDFtermOrder</u>	713
<u>printNotation</u>	713
<u>saveDefaultsToOCEAN</u>	714
<u>showWhatsNew</u>	715
<u>digits</u>	715
<u>obsoleteWarnings</u>	716
<u>netlistAccess</u>	716
<u>printCommentChar</u>	717
<u>updateCDFtermOrder</u>	717
<u>toolList</u>	718
<u>ignoreSchModified</u>	718
<u>defaultTools</u>	719
<u>oceanScriptFile</u>	719

Virtuoso Analog Design Environment L User Guide

<u>printInlines</u>	720
<u>awvResizeWindow</u>	720
<u>paraplotUpdateSimulatorLog</u>	721
<u>sevResolveSymLinks</u>	721
ADE XL	722
<u>showMenu</u>	722
<u>showOpenViewDialog</u>	722
<u>defaultLibName</u>	723
<u>viewNamePrefix</u>	723
SpectreVerilog (IC6.1.6 only)	724
<u>simOutputFormat</u>	724
<u>fastViewOption</u>	725
<u>logicOutputFormat</u>	726
HspiceD	727
<u>hspiceSoftLineLength</u>	727
<u>hspiceMaxLineLength</u>	727
<u>mapGndNetToZero</u>	728
<u>netlistModelFileFirst</u>	728
<u>userCmdLineOption</u>	729
<u>setTopLevelAsSubckt</u>	729
AMS	730
<u>globalSignals</u>	730
<u>netlistMaxWarn</u>	732
<u>netlistNoWarn</u>	732
<u>upgradeMsgSevWarn</u>	733
<u>upgradeMsgSevError</u>	733
<u>print_control_vars</u>	734
<u>ac_severity</u>	734
<u>assert_severity_default</u>	735
<u>dc_severity</u>	735
<u>dcOp_severity</u>	736
<u>tran_severity</u>	736
<u>connectRulesList</u>	737
<u>useEffectiveCDF</u>	737
<u>disableRunModelInDP</u>	738
<u>simOutputFormat</u>	738

Virtuoso Analog Design Environment L User Guide

<u>useOtherOutputFormat</u>	739
<u>AMS IGNORE IGNORE</u>	739
<u>Ultrasim</u>	740
<u>wf_format</u>	740
<u>fastViewOption</u>	741
<u>useOtherOutputFormat</u>	741
<u>UltraSimVerilog (IC6.1.6 only)</u>	742
<u>wf_format</u>	742
<u>fastViewOption</u>	743

B

<u>auCdl Netlisting</u>	745
<u>What Is auCdl and Why Do You Need It?</u>	745
<u>Licensing Requirements</u>	746
<u>Running auCdl</u>	746
<u>Running auCdl from within DFII</u>	746
<u>Running auCdl from the Command Line</u>	747
<u>Creating a config view for auCdl</u>	750
<u>How to include partial netlist file in SUBCKT calls</u>	750
<u>Customization Using the .simrc File</u>	753
<u>auCdl-Specific Parameters</u>	753
<u>View List, Stop List, Netlist Type, and Comments</u>	754
<u>Preserving Devices in the Netlist</u>	755
<u>Removing Devices in the Netlist</u>	755
<u>Printing CDL Commands</u>	755
<u>Defining Power Node and Ground Node</u>	755
<u>Support for Global Power and Ground Signals from CDL UI</u>	756
<u>Evaluating Expressions</u>	756
<u>NLP Expressions</u>	757
<u>Mapping Global Pins</u>	757
<u>Renaming Cell Names</u>	758
<u>Renaming Pcell Subcircuits</u>	758
<u>Customizing Bulk Node Search</u>	758
<u>Support for HED Features</u>	761
<u>Custom Netlisting Procedures</u>	762

Virtuoso Analog Design Environment L User Guide

ansCdlSubcktCall	762
ansCdlCompPrim	763
ansCdlCompParamPrim	763
ansCdlSpecParamPrim	764
ansCdlSubcktCallExtended	764
ansCdlHnlPrintInst	765
Black Box Netlisting	771
Additional Customizations	775
Automatically Including a Partial Netlist File within the .SUBCKT Definition for the Top or Mid-Level Cells in your Design	775
Including a ROM-Insert Netlist Automatically Into the auCdl Netlist	778
PININFO for Power and Ground Pins	778
Changing the Pin Order	779
.PARAM Statement	780
Specifying the Terminal Order for Terminals	780
Notification about Net Collision	785
Making a Stop Cell at Subcircuit Level	788
Printing Empty Subcircuits for Stopping Cells	788
Passing Parameter	789
Netlisting the Area of an npn	790
CDF Simulation Information for auCdl	790
Device CDF Values	791
Netlist Examples	794
What is Different in the 4.3 Release	795
Complete Example	796

C

auLvs Netlisting	799
Using auLvs	799
Customization Using the .simrc File	799
Related Documentation on auLvs	800

D

Using the runams Command	801
runams Command Options	802

Virtuoso Analog Design Environment L User Guide

<u>How to Use runams Options</u>	815
<u>runams Command Examples</u>	817
<u>Index</u>	819

Preface

Virtuoso Analog Design Environment L allows you to set up and run analog simulations using different simulators. After running a simulation, you can view and analyze simulation results.

This manual describes how to use the Virtuoso® Analog Design Environment to simulate analog designs. The information presented in this manual is intended for integrated circuit designers and assumes that you are familiar with analog design and simulation.

The preface discusses the following:

- [Scope of this Guide](#) on page 24
- [Licensing in ADE L](#) on page 24
- [Related Documents for ADE L](#) on page 24
- [Third Party Tools](#) on page 26
- [Typographic and Syntax Conventions](#) on page 26
- [Additional Learning Resources](#) on page 28
- [Help and Support Facilities](#) on page 29

Scope of this Guide

All the functionality described in this guide is available in IC6.1.6 and ICADV12.1 onward unless otherwise noted. Features that are supported only in a particular release are identified using (ICADV12.1 only) and (IC6.1.6 only) labels.

Licensing in ADE L

The license number required for ADE L is `Analog_Design_Environment_L`. One ADE L feature licence is required for one User, Display and Host (UHD) session of ADE L.

You can also set ADE L to be your default application by selecting *File - Set Default Application* and ensuring that ADE L is set as the default for the listed scenario options. For more information on setting a default application see, [Virtuoso Design Environment User Guide](#).

For more information on licensing and related information, see:

- [Obtaining Licences in Virtuoso Design Environment User Guide](#)
- [Cadence Workspaces in Virtuoso Design Environment User Guide](#)
- [Virtuoso Software Licensing and Configuration User Guide](#)

Related Documents for ADE L

The following documents provide more information about the topics discussed in this guide.

Installation, Environment, and Infrastructure

- For information on installing Cadence products, see the [Cadence Installation Guide](#).
- For information on the Virtuoso design environment, see the [Virtuoso Design Environment User Guide](#).
- For information on database SKILL functions, including data access functions, see the [Virtuoso Design Environment SKILL Reference](#).
- For information on library structure, the library definitions file, and name mapping for data shared by multiple Cadence tools, see the [Cadence Application Infrastructure User Guide](#).

Virtuoso Analog Design Environment L User Guide

Preface

- For information on Virtuoso circuit simulator, see the [*Virtuoso® Spectre® Circuit Simulator Reference*](#).

Technology Information

- For information on how to create and maintain a technology file and display resource file, see the [*Virtuoso Technology Data User Guide*](#) and the [*Virtuoso Technology Data ASCII Files Reference*](#).
- For information on how to access the technology file using SKILL functions, see the [*Virtuoso Technology Data SKILL Reference*](#).

Virtuoso Tools

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

- [*Virtuoso Analog Design Environment XL User Guide*](#) and [*Virtuoso Analog Design Environment GXL User Guide*](#) gives information about Monte Carlo, optimization, and statistical analysis.
- [*Virtuoso Visualization and Analysis XL User Guide*](#) provides more information about the Virtuoso Visualization and Analysis display tools.
- [*Virtuoso Analog Distributed Processing Option User Guide*](#) describes how to use multiple hosts to distribute simulations between a collection of different machines.
- [*Virtuoso Mixed-Signal Circuit Design Environment User Guide \(ICADV6.1.6 only\)*](#) gives information about how to set up and run mixed-signal simulations.
- [*Virtuoso Parasitic Estimation and Analysis User Guide*](#) describes how to analyze parasitics.
- [*Virtuoso Schematic Editor L User Guide*](#) describes connectivity and naming conventions for inherited connections and how to add and edit net expressions in a schematic or symbol cellview.
- [*Virtuoso Spectre Circuit Simulator Reference*](#) describe the Virtuoso® Spectre analog circuit simulator in detail.
- [*Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide*](#) describes how to use the RF option in the Spectre simulator.
- [*Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*](#) describes how to use the Virtuoso Accelerated Parallel Simulator.

- [Virtuoso Design Environment Migration Guide](#) describes the release level changes and migration related information.
- [Virtuoso Design Environment Adoption Guide](#) provides information related to Open Access.
- [Virtuoso UltraSim Simulator User Guide](#) provides detailed information about the UltraSim simulator.
- [Virtuoso AMS Designer Environment User Guide](#) provides detailed information about the Virtuoso AMS Designer simulator.
- [Component Description Format User Guide](#) describes Cadence's Component Description Format (CDF) for describing parameters and the attributes of parameters of individual components and libraries of components.
- [Analog Expression Language Reference](#) contains concept and reference information about the Analog Expression Language (AEL).

Third Party Tools

To view any `.swf` multimedia files, you need:

- A Cadence Online Support Login.
- Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the [Adobe](#) website.
- Speakers and a sound card installed on your computer for videos with audio.

Typographic and Syntax Conventions

This list describes the syntax conventions used in this manual.

`literal`

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

`argument` (*z_argument*)

Words in italics indicate user-defined arguments for which you must substitute a name or a value. (The characters before the underscore (`_`) in the word indicate the data types that this

Virtuoso Analog Design Environment L User Guide

Preface

argument can take. Names are case sensitive. Do not type the underscore (*z_*) before your arguments.)

[]

Brackets denote optional arguments.

...

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

argument...

Specify at least one, but more are possible.

[argument]...

Specify zero or more.

, ...

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

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

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

SKILL Syntax Examples

The following examples show typical syntax characters used in SKILL. For more information, see the [Cadence SKILL Language User Guide](#).

Example 1

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

Example 1 illustrates the following syntax characters.

`list`

Plain type indicates words that you must enter literally.

g_arg1

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

()

Parentheses separate names of functions from their arguments.

Virtuoso Analog Design Environment L User Guide

Preface

_	An underscore separates an argument type (left) from an argument name (right).
[]	Brackets indicate that the enclosed argument is optional.
=>	A right arrow points to the return values of the function. Also used in code examples in SKILL manuals.
...	Three dots indicate that the preceding item can appear any number of times.

Example 2

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

Example 2 illustrates two additional syntax characters.

	Vertical bars separate a choice of required options.
/	Slashes separate possible return values.

Form Examples

Each form shows you the system defaults:

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

Additional Learning Resources

Cadence provides various [Rapid Adoption Kits](#) that you can use to learn how to employ Virtuoso applications in your design flows. These kits contain workshop databases, designs, and instructions to run the design flow.

Cadence offers the following training courses on Virtuoso Analog Design Environment:

- [Virtuoso Schematic Editor](#)
- [Virtuoso Analog Design Environment](#)

- [Analog Modeling with Verilog-A](#)
- [Behavioral Modeling with Verilog-AMS](#)
- [Real Modeling with Verilog-AMS](#)
- [Spectre Simulations Using Virtuoso ADE](#)
- [Virtuoso UltraSim Full-Chip Simulator](#)

For further information on the training courses available in your region, visit the [Cadence Training](#) portal. You can also write to training_enroll@cadence.com.

Note: The links in this section open in a new browser. The course links initially display the requested training information for North America, but if required, you can navigate to the courses available in other regions.

Help and Support Facilities

The following help and support facilities are available as Help menu options:

Help Menu Option	Description
<i>Contents</i>	Invokes Cadence Help with the Virtuoso Analog Design Environment L User Guide table of contents on display.
<i>What's New</i>	Opens up the Virtuoso What's New document in Cadence Help at the Virtuoso Analog Design Environment section.
<i>Known Problems and Solutions</i>	Opens up the Virtuoso Known Problems and Solutions document in Cadence Help at the Virtuoso Analog Design Environment section.
<i>Virtuoso Documentation</i>	Opens up Cadence Help, initially by default at the Virtuoso Platform What's New overview. To view the entire Virtuoso documentation library contents, if not already on display, select View - Show Navigation.
<i>Cadence Video Library</i>	Opens up Cadence Online Support (COS), using your default web browser, initially displaying the Cadence Video Library site. Note: You are required to have a Cadence Online Support account to access these materials.

Virtuoso Analog Design Environment L User Guide

Preface

Help Menu Option	Description
<i>Cadence Online Support</i>	Displays the Cadence customer support site (COS) on your default web browser. Note: You are required to have a Cadence Online Support account to access these materials.
<i>Cadence Users Forum</i>	Displays the Cadence online users forum in your default web browser.
<i>About ADE L</i>	Displays version and copyright information for Virtuoso Analog Design Environment L.

Features of the Virtuoso Analog Design Environment L

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

- [Consistent User Interface](#) on page 31
- [Analog Design Entry](#) on page 32
- [Design Hierarchy](#) on page 32
- [Annotation](#) on page 33
- [Interactive Simulation](#) on page 33
- [Simulation Output and Analysis](#) on page 35
- [Advanced Analysis](#) on page 35

Consistent User Interface

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

The design framework II environment is an open system. You can integrate third party tools and enter your own design data with industry-standard EDIF and Virtuoso® GDSII Stream formats. Circuit simulators can be integrated using OSS.

For details on how to set up the Analog Design Environment L, refer [Environment Setup](#) on page 37.

Menu Access Keys

Menu access keys provide keyboard access to functionality and application menus without the need to use mouse selections. Starting IC610, Analog Design Environment also supports access keys for all the menus.

For example, selecting the Alt + F access keys together will display the contents of the File banner menu.

If you want to open the Setting Temperature form using keyboard, Select Alt + u. The Setup submenu will be displayed. To select Temperature, you need not again press Alt +T, you can directly select T from keyboard. The Setting Temperature form will appear.

Note: The Access Keys for each menu can be identified by underlined characters. For Example, In the Setup menus, u is the access key.

Analog Design Entry

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

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

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

Design Hierarchy

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

Annotation

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

Interactive Simulation

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

Beginning this release, Cadence will no longer be supporting the socket-style simulator integrations. This includes the integrations known as spectreS, hspiceS, and spectreSVerilog. These technologies have been replaced by the direct style integrations. For customers who integrate proprietary simulators into the Analog Design Environment, Cadence provides only the direct version of the OASIS integration API.

See [Migrating Socket Libraries to Direct Simulators](#) on page 50 for more information.

Important Benefits of Direct Simulation

- Improved performance in netlisting

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

- Improved performance of simulation for spectre

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

- Readable netlists

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

Virtuoso Analog Design Environment L User Guide

Features of the Virtuoso Analog Design Environment L

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

For detailed information about the preferred modeling approach for direct simulations, see the [Direct Simulation Modeling User Guide](#).

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

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

- Read-only designs can be simulated, provided that they are extracted
- Improved support of standalone netlisting

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

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

Important Use-Model Differences between spectreS and spectre

There are two important differences that existing spectreS users need to be aware of:

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

simulate read-only designs. This is one of the long-term problems that has been resolved with direct simulation.

Simulation Output and Analysis

The Analog Design Environment supports advanced analog/mixed-signal waveform display and post-processing tools, e.g. *Virtuoso Visualization and Analysis*, which features:

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

To learn more about the waveform display tool, refer to the [*Virtuoso Visualization and Analysis XL User Guide*](#).

Advanced Analysis

The Virtuoso ADE L supports advanced analysis features such as Parametric Analysis. Other advanced analysis features such as Monte Carlo, Corners analysis and the Optimizer are included in Virtuoso[®] ADE XL and Virtuoso[®] ADE GXL. For more details on these tools see, [*Virtuoso Analog Design Environment XL User Guide*](#) and [*Virtuoso Analog Design Environment GXL User Guide*](#) respectively.

Virtuoso Analog Design Environment L User Guide

Features of the Virtuoso Analog Design Environment L

You can launch ADE XL or ADE GXL from the ADE L environment and use the same design for advanced analysis. To learn how, refer [Running the Simulation Setup in Advanced Simulation Environments \(ADE XL / ADE GXL\)](#) on page 106.

Environment Setup

This chapter describes the Virtuoso Analog Design Environment and tells you how to set this environment for simulations. This chapter also describes how you use Cadence simulators and third-party simulators in the Analog Design Environment.

- [About the Simulation Window](#) on page 37
- [Simulator Interfaces](#) on page 57
- [Setting Up Simulation Files](#) on page 67
- [Setting Simulation Environment Options](#) on page 74
- [Setting Up a Remote Simulation](#) on page 95
- [Saving and Restoring the Simulation Setup](#) on page 97
- [Running the Simulation Setup in Advanced Simulation Environments \(ADE XL / ADE GXL\)](#) on page 106
- [Configuring the Analog Design Environment](#) on page 114
- [Reserved Words](#) on page 119
- [Bindkeys](#) on page 120
- [Form Field Descriptions](#) on page 123

About the Simulation Window

The Virtuoso Analog Design Environment L window is also referred to as the simulation window. Using the simulation window, you can run different simulations for a design. You choose a simulator and the types of analyses you want to run for the design. You can also specify design variables that you want to use for simulations. When you run simulation, the results are saved in a predefined formats and can be used by other tools such as the Virtuoso Visualization and Analysis tool for further analysis. If there are signals in the simulation results, they are by default plotted in the Virtuoso Visualization and Analysis Graph window.

Opening the Simulation Window

There are two ways to open the simulation window:

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

Opening the Simulation Window from Virtuoso Schematic Editor

To start the simulation window from the Virtuoso Schematic Editor window,

1. Open a design in the Virtuoso Schematic Editor window.
2. Choose *Launch – ADE L*.

The simulation window opens and the simulation environment is initialized for the design that is already open in Virtuoso Schematic Editor. You can now set up and run simulations for this design.

Starting IC 6.1.5, a new workspace, ADE L, also opens in the Schematic Editor window. This workspace displays three toolbars that provide quick access to the ADE L commands. Using these toolbars, you can set up and run simulations and perform post-processing functions on the simulation results from the Schematic Editor window. This feature provides the following usability enhancements:

- You can now perform various simulation-related tasks without switching between the simulation and Schematic Editor window.
- If you change the simulation settings or customize menus from the simulation window, the Schematic Editor toolbars are also synchronized accordingly. For example, if you choose a simulator from the simulation window, the sub menu items in the Direct Plot menu change accordingly in both the simulation window menu and the Schematic Editor toolbar. Similarly, if you add a new sub menu item in the *Direct Plot* menu in the simulation window, the new sub menu item appears in the *Direct Plot* toolbar menu in the Schematic Editor.

For more details on the ADE L workspace in Virtuoso Schematic Editor, refer to the [ADE L Integration with Virtuoso Schematic Editor](#) section in the Virtuoso Schematic Editor L user guide.

Opening the Simulation Window from CIW

To open the simulation window from the CIW,

- Choose *Tools – ADE L*.

Virtuoso Analog Design Environment L User Guide

Environment Setup

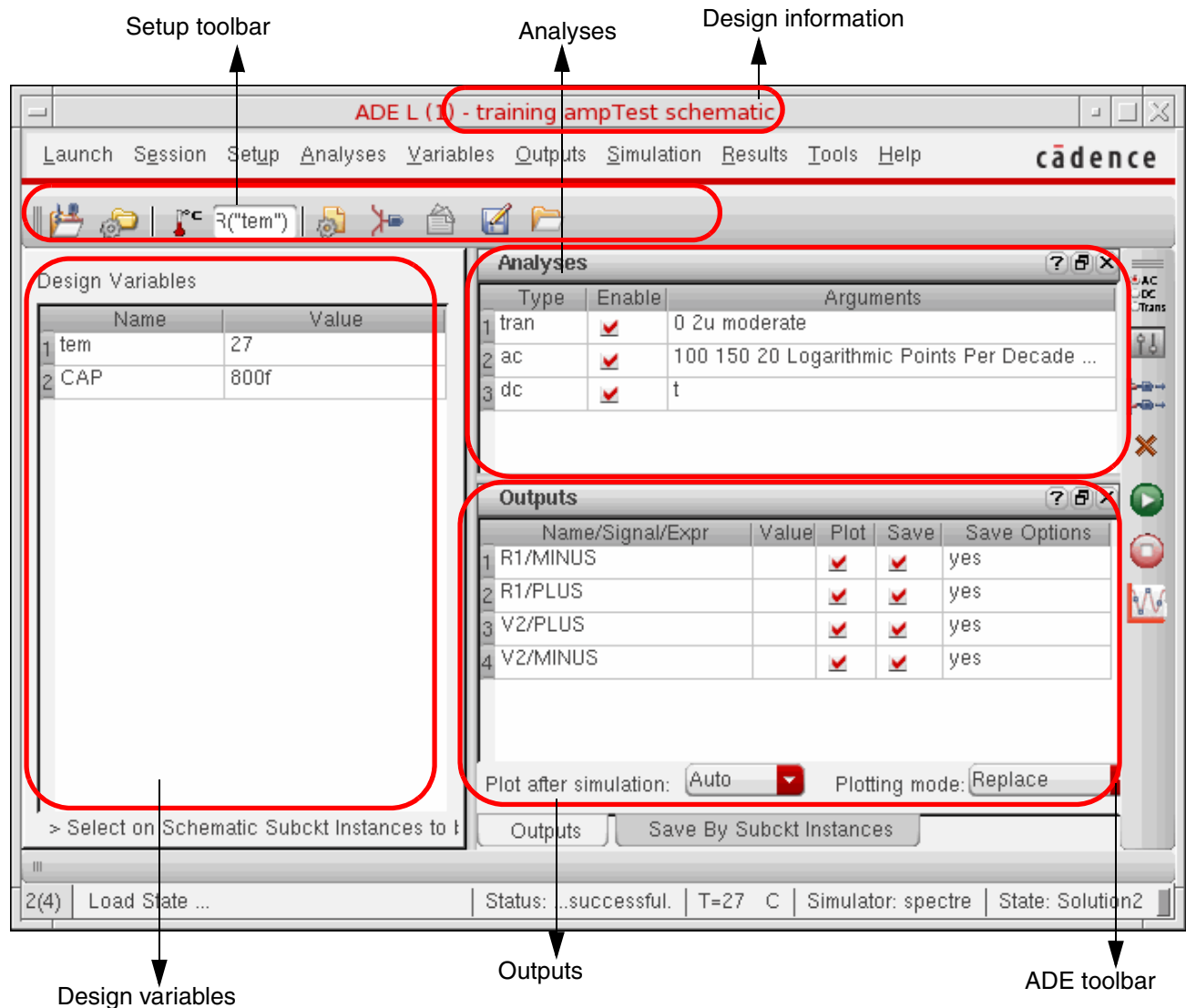
The simulation window opens and the simulation environment is initialized. Now, you can choose a design and set up and run simulations.

For more details on the simulation window, refer to the [The Simulation Window GUI](#) section.

If you have already saved the simulation setup for the design in a saved state, you can open that using the Load State form and continue to run simulations. For more details on loading state, refer to the [Restoring the Simulation Setup](#) section.

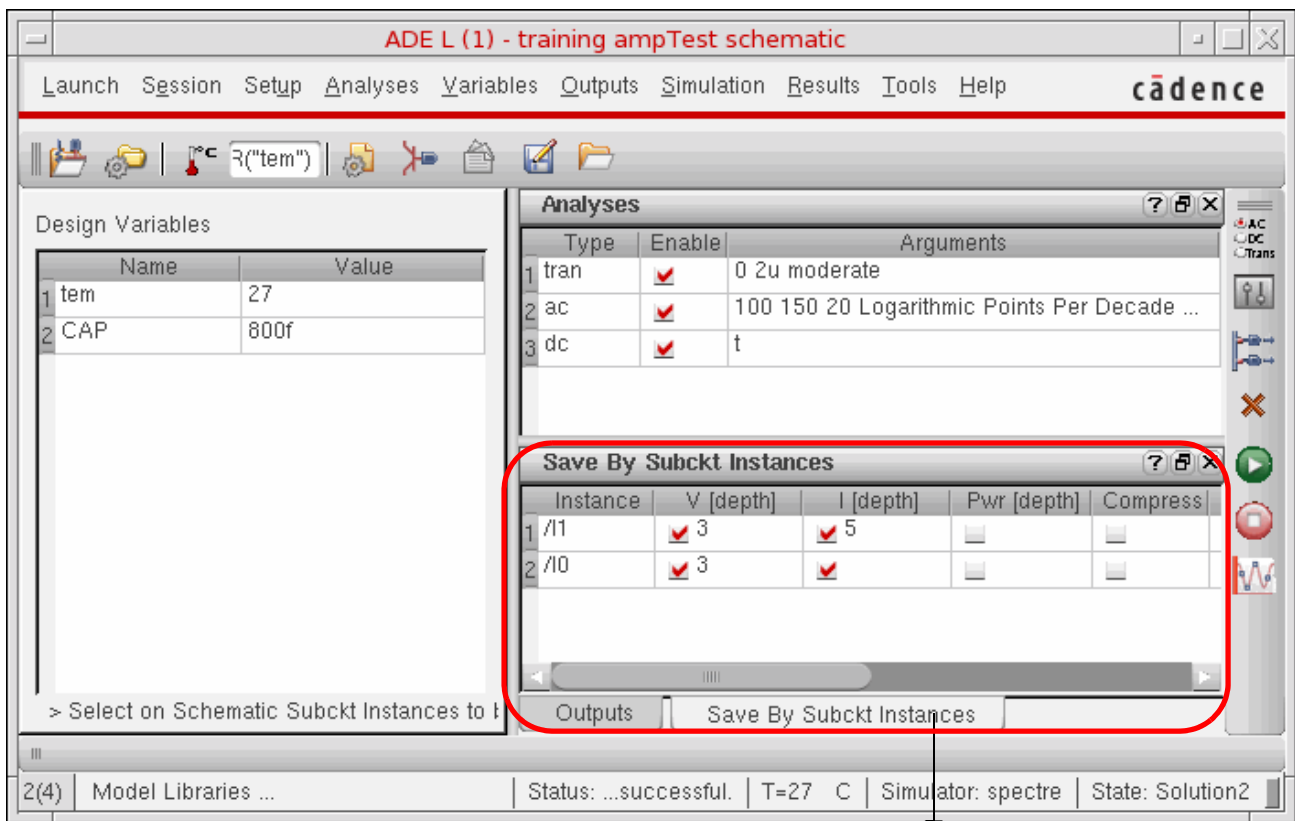
The Simulation Window GUI

The following is an example of a simulation setup for the `ampTest` design in the simulation window.



Virtuoso Analog Design Environment L User Guide

Environment Setup



Save By Subckt
Instances

The simulation window has the following features:

- The design information (library, cell, and view) is displayed on the title bar.
- The simulation window contains four panes:
 - *Design Variables*: In this pane, you can specify the names of design variables and their values to be used while running simulations. For details on how to specify design variables, refer to [Chapter 3, “Design Variables and Simulation Files for Direct Simulation.”](#)
 - *Analyses*: This pane lists all the analyses that you have specified for the current design. In this pane, you can edit the details of an analysis or enable or disable it for the next simulation run. For details on how to specify analyses, refer to [Chapter 4, “Setting Up for an Analysis”](#).
 - *Outputs*: In this pane, you can specify the signals and expressions that you want to save or plot from the simulation results. After the simulation is run, the data values

Virtuoso Analog Design Environment L User Guide

Environment Setup

are displayed in this section. For more details on how to specify outputs, refer to [Chapter 5, “Selecting Data to Save and Plot”](#).

- ❑ *Save By Subckt Instances*: In this pane, you can specify the level of hierarchy to which the outputs should be saved for the subcircuits. For more details on how to specify levels, refer to [Specifying Hierarchy Levels to Save Outputs](#) on page 276.

Note: This pane is available only for the Spectre and spectreVerilog simulators.

- The *Design Variables* pane is fixed, but you can undock the *Analyses*, *Outputs*, or *Save By Subckt Instances* panes and move them anywhere inside or outside the simulation window. To undock/dock a pane, click the *Float* button on the upper-right corner of that pane.

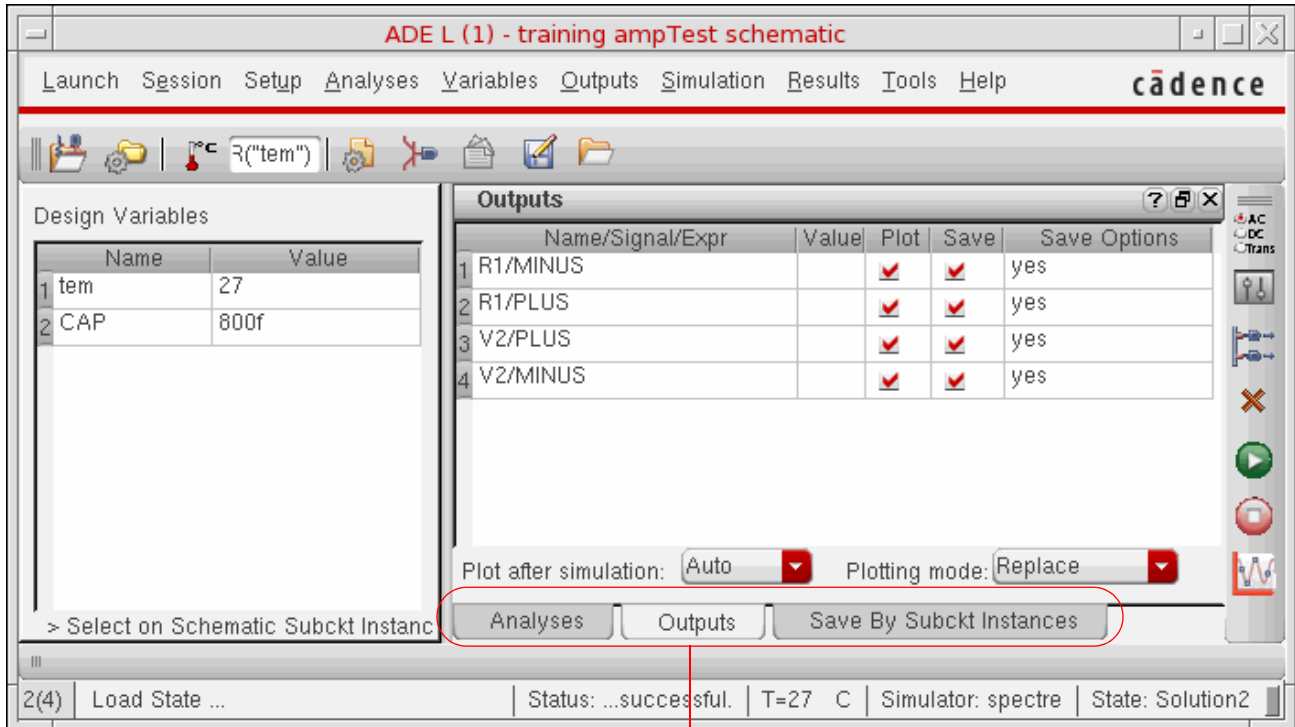


- You can resize the simulation window or any of the panes using the horizontal and vertical splitters between these panes. To reset the simulation window and the panes to their original size and location, choose *Session – Restore Default View*.
- By default, the horizontal and vertical scrollbars do not appear on the panes. When you add large number of rows that are not visible together or you reduce the size of simulation window, the scrollbars appear automatically, if required.
- You can drag and drop the rows within each of the *Design Variables*, *Analyses*, *Save By Subckt Instances*, or *Outputs* panes. In the *Design Variables*, *Outputs* and *Save By Subckt Instances* pane, this helps in placing the rows as per convenience. In the *Analyses* pane, by rearranging the rows, you can change the order in which the analyses are run. For more details, refer [Specifying Order for Analyses](#) on page 171.
- You can close any or all of the *Analyses*, *Outputs* or *Save By Subckt Instances* panes by clicking the *Close* button on the upper-right corner of that pane. To reopen a closed pane, right-click on the menu bar and choose the name of the pane. The pane appears at its last location in the simulation window, and its settings are not lost.

Virtuoso Analog Design Environment L User Guide

Environment Setup

- The *Analyses*, *Outputs* and *Save By Subckt Instances* panes can also be viewed as tabs, as shown below.



Tabs

To arrange the panes as tabs, click the title bar of a pane and start dragging. The other pane will take the complete space to the right of the simulation window and will appear as a tab. Drop the pane being dragged at the centre of the other pane. Now, the panes appear as tabs and the pane that you dragged is arranged on top.

To bring back the panes at their default location,

- choose *Session – Restore Default View*.

Note: The *Save By Subckt Instances* pane hides when you choose the *Session – Restore Default View* option, however, the settings for the pane are not lost.

- select the required tab and drag it towards the right of the simulation window till the time a placeholder for it appears. When the placeholder appears, drop the tab on it. Now, the panes appear together.

- The current value of *Temperature* design variable, name of the current simulator, current state, and status are displayed on the status bar.

Virtuoso Analog Design Environment L User Guide

Environment Setup

- The ADE window contains two toolbars, the ADE toolbar and the Setup toolbar. For more details on these toolbars, refer to [ADE L Toolbars](#).

Important

All the display settings done in simulation window, except the use of splitters to resize a pane, are saved in the log file. When the log file is run in the replay mode, the settings are reflected in the GUI. However, when the log files from the releases prior to IC 6.1.4 are run, changes in the display settings are not reflected.

ADE L Toolbars

The ADE GUI contains the following toolbars:




- [ADE Simulation Toolbar](#)
- [ADE Simulation Setup Toolbar](#)
- [ADE Results Directory Toolbar](#)

ADE Simulation Toolbar

This toolbar provides buttons to specify design variables, analyses and outputs, to generate netlist and run simulations.







The following table describes all the buttons of this toolbar.

Button	Description
	Opens the Choosing Analyses form using which you can choose an analysis for simulation. For more details, refer to Setting Up for an Analysis .
	Opens the Editing Design Variables form using which you can edit the design variables. For more details, refer to Design Variables and Simulation .
	Opens the Setting Outputs form using which you can set outputs for simulation. For more details, refer to Setting Outputs .

Virtuoso Analog Design Environment L User Guide

Environment Setup




Button	Description
	Deletes the selected row from the <i>Design Variables</i> , <i>Analyses</i> , <i>Outputs</i> or <i>Save By Subckt Instances</i> pane.
	Generates netlist and starts the simulation run.
	Stops simulation.
	Plots the output values of the last run simulation in the Virtuoso Visualization and Analysis graph window.

ADE Simulation Setup Toolbar

Provides buttons to prepare setup for simulation. For example, it provides options to specify value for temperature or to choose design, simulator, model libraries, stimuli, and environment options.








The following table describes all the buttons of this toolbar.

Button	Description
	Opens the Choosing Design form using which you can choose a design for simulation. For more details, refer to Choosing the Design
	Opens the Choosing Simulator/Directory/Host form using which you can choose a simulator, directory, or host to run the simulation. For more details, refer to Choosing a Simulator .
	Sets the value for temperature. You can either click the Temperature button to open the Setting Temperature form or directly change the value for temperature in the text box. For more details, refer to Setting the Simulation Temperature

Virtuoso Analog Design Environment L User Guide

Environment Setup

Button	Description
	Opens the Model Library Setup form using which you can set up a model library. For more details, refer to Setting the Model Path .
	Opens the Setup Analog Stimuli form using which you can set up stimuli for analog simulation. For more details, refer to Stimuli Setup . Note: This option is not available for the AMS simulator.
	Opens the Environment Options form using which you can set up the environment options. For more details, refer to Setting Simulation Environment Options .
	Opens the Saving State form using which you can save the current state of the design. For more details, refer to Saving the Simulation Setup .
	Opens the Loading State form using which you can load a saved state of the design. For more details, refer to Restoring the Simulation Setup .

ADE Results Directory Toolbar

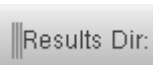

This toolbar displays the directory of the simulation path and provides option to open the directory containing the results.



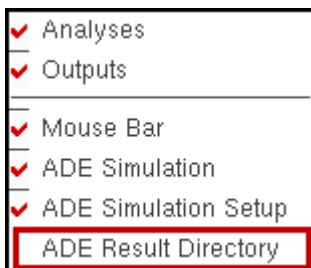
Virtuoso Analog Design Environment L User Guide

Environment Setup

The following table describes the options of this toolbar:

Button	Description
	<p>Displays the directory of the current or last-run simulation path from where the results are loaded.</p> <p>If you select the results of a different simulation run using the <i>Results – Select</i> option, the <i>Results Dir</i> field displays the path of the current directory from which the results were loaded.</p> <p>Important Points to Note:</p> <ul style="list-style-type: none">■ When you load a state, the <i>Results Dir</i> field displays the path of the default project directory if it is set in the <code>.cdsinit</code> file. Otherwise, it displays <i>None</i>.■ You can select and copy the path of the results directory from the <i>Results Dir</i> field, but cannot change it.
	<p>Opens the terminal window in which the current directory is set to path that contains the results.</p>

Note: By default, the *ADE Result Directory* toolbar is not visible in ADE L. To view the toolbar, right-click on the menu bar or any toolbar in ADE L, and select *ADE Result Directory*.

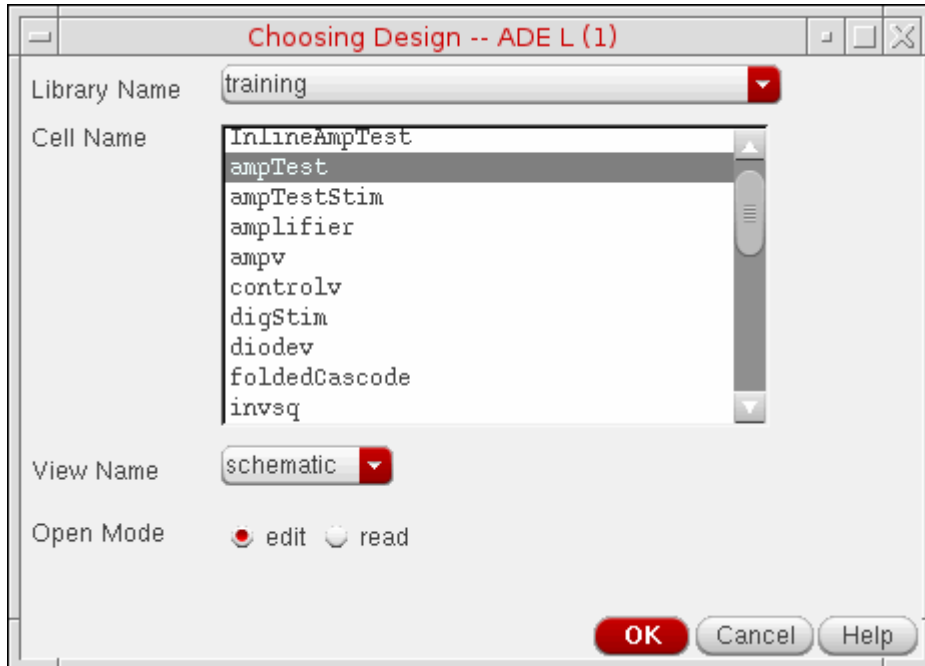


Choosing the Design

To open a design or to select a different design,

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

The Choosing Design form appears.



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

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

Choosing a Simulator

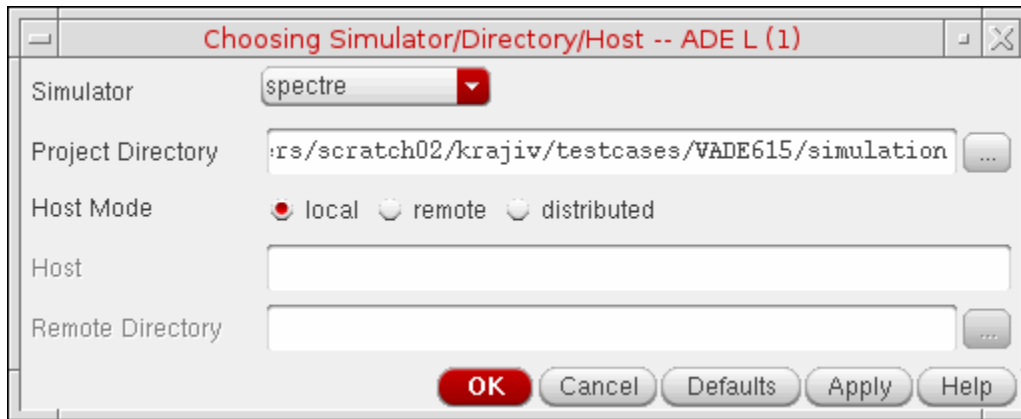
To choose a simulator,

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

Virtuoso Analog Design Environment L User Guide

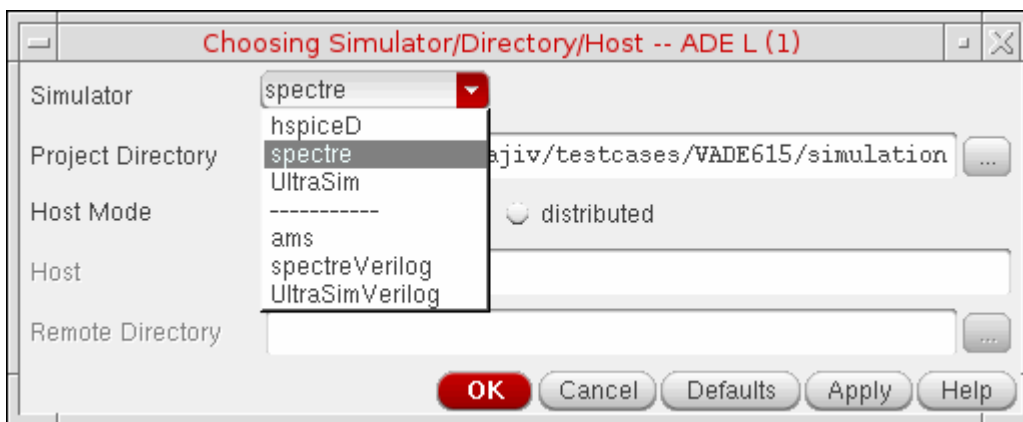
Environment Setup

The Choosing Simulator/Directory/Host form appears.



For detailed information about the form, see [Choosing Simulator/Directory/Host](#) on page 123.

2. Choose a simulator from the Simulator cyclic list. For more information about these simulators, see [Simulator Interfaces](#) on page 57.



Important

When you switch from one simulator to another, a new oasis session will be created. The new simulator retains the setup files recognized by it and ignores the rest. This behavior is controlled by *retainStateSettings* environment variable with the default value “yes”. On setting this variable to “no”, the new simulator does not retain the setup files from the previous oasis session.

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

For information on the *remote* host mode, see the topic [Setting Up a Remote Simulation](#) on page 95.

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

Note: When you change the *Project Directory*, ADE L does not create a new oasis session. It changes the project directory path at the relevant places and retains the old oasis session. This ensures that the entire setup remains unchanged.



Changing the Project Directory results in a change in simulator settings. These changes might cause the simulation to fail. If you encounter such a problem, define `asiSetSimulationDirectoryChangeSetup()`. This API will set directory settings for the simulator.

5. Click *OK* or *Apply*.

The Simulation window shows the name of the selected simulator on the status bar.

Simulator/Directory/Host ...	Status: Ready	T=27 C	Simulator: spectre	State: Solution2
------------------------------	---------------	--------	--------------------	------------------

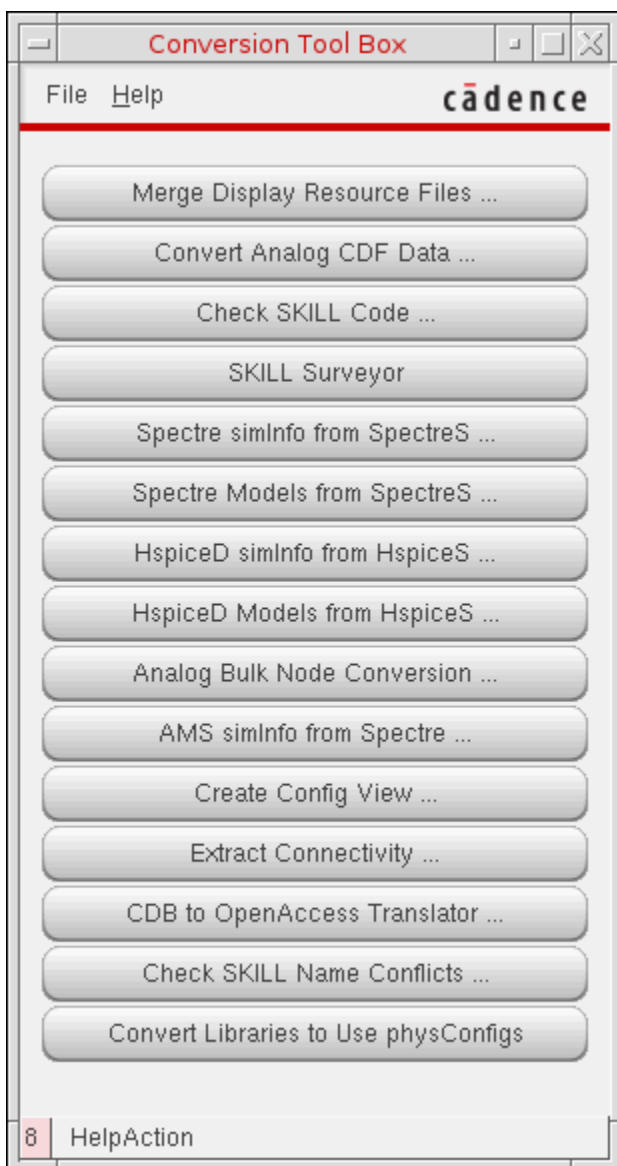
Migrating Socket Libraries to Direct Simulators

You can migrate existing socket libraries and model files using the *Conversion Tool Box*. The *Conversion Tool Box* is used to convert design libraries and associated technology data, and to prepare files for conversion to Direct simulation.

Virtuoso Analog Design Environment L User Guide

Environment Setup

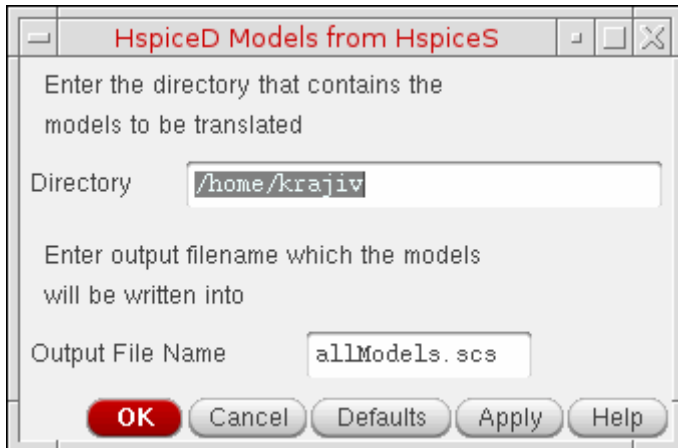
To bring up the *Conversion Tool Box* window, click *Tools – Conversion Tool Box* in the CIW (Command Interpreter Window).



Virtuoso Analog Design Environment L User Guide

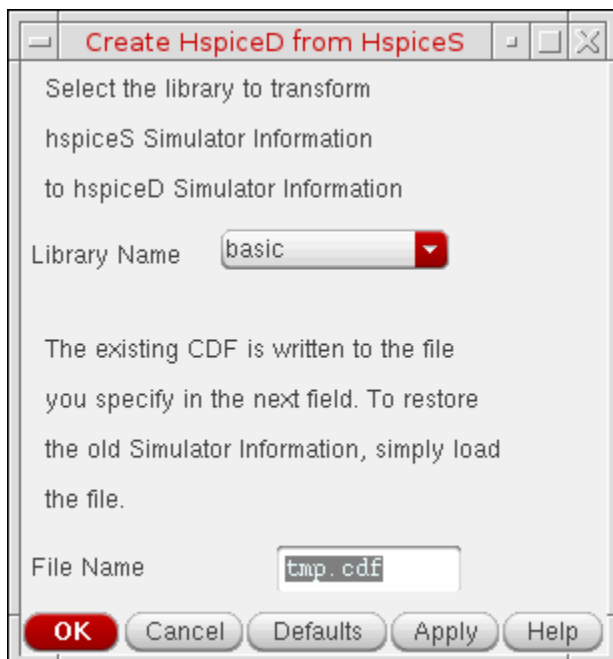
Environment Setup

For example, click the *HspiceD Models from HspiceS* button to open the *HspiceD Models from HspiceS* window.



This utility locates all the *HspiceS* model files in a directory, translates them into *HspiceD* models, and places the translated models in a single file. This can be included to simulate a circuit (using the *Hspice Direct* interface) by adding the file through the Model Library Setup form. The *HspiceS* model files cannot be translated from two different directories. To do so, use the unix command `cat` to merge the file created from two different directories.

To transform *HspiceS* simulator information to *HspiceD* simulator information, click the *HspiceD simInfo from HspiceS* button. The *Create HspiceD from HspiceS* window displays.

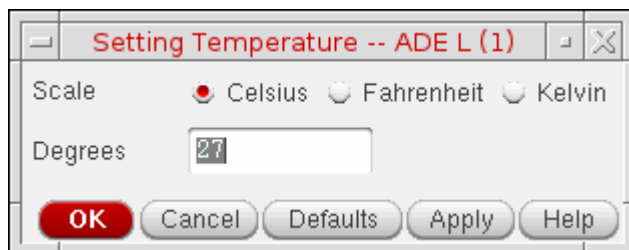


Setting the Simulation Temperature

To set the simulation temperature,

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

The Setting Temperature form appears.



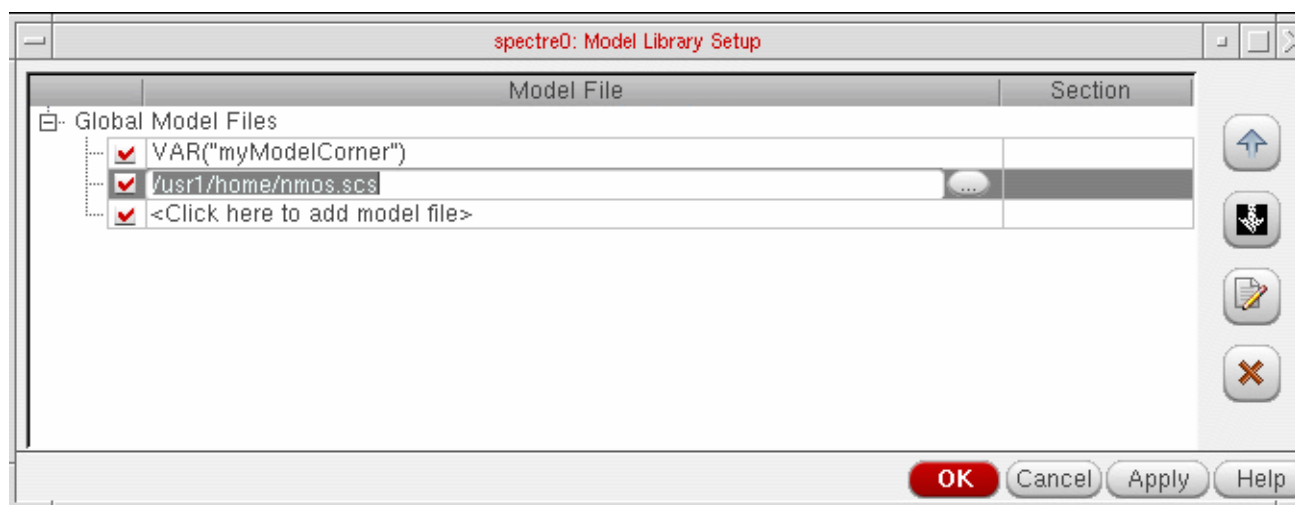
2. Choose the units you want to use for temperature.
3. Type a value in degrees, and click *OK*.

Setting the Model Path

To set up models for simulator interfaces:

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

The *Model Library Setup* form appears.



For detailed information about the form, see [Model Library Setup](#) on page 126.

2. In the *Model Files* column, type the path and file name of the model file you want to use.

The model files contain all model definitions referred to by your design and not defined within the Virtuoso library. Unless you specify a full path, the simulator assumes that you are using the project directory.

Alternatively, you can click the browse button to select a valid model file.

Example:

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

```
simulator lang=spectre
model npn bjt type=npn is=3.26E-16 va=60 bf=100 \
br=6 nc=2 ikr=100m rc=1 vje=0.7 \
cjc=1e-12 fc=0.5 cje=0.7e-12 \
tr=200e-12 tf=25e-12 itf=0.03 vtf=7 xtf=2
model pnp bjt type=pnp is=3.28e-16 va=30 bf=35 \
br=6 nc=2 ikr=100m rc=1 \
cjc=1e-12 fc=0.5 cje=0.7e-12 \
tr=200e-12 tf=65e-12 itf=0.03 vtf=7 xtf=2
```

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

3. (Optional) In the *Section* column, select a section from the drop-down list.

Choosing a User Interface Path

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

- Through the Simulation window

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

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

- Through the Schematic window

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

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

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

Either path allows you to adjust inputs and outputs, run simulations, and select data to be plotted or saved. You can select which window to open automatically at startup by specifying your default user model as described in [“Setting Basic Session Defaults”](#) on page 115.

Using the Simulation Window

In a single view, the Simulation window displays

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

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

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

Using the Schematic Window

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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

Simulator Interfaces

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

Spectre Simulator

The analog circuit design environment provides the spectre interfaces to the Spectre® analog simulator.

From IC 6.1, you need to use the MMSIM version of Spectre, which is available on the MMSIM CD and not in the dfl hierarchy. If you set up your path to point to a previous (non-MMSIM) version of the Spectre software, the simulation will not run and you will see the following message:

```
"The 'spectre' executable that you are using is an older version. Use MMSIM60 or later version of Spectre with this release. To check the spectre version, run 'spectre -W'."
```

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

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

- [Spectre Options on page 303](#)
- [Setting Up a Spectre Analysis on page 174](#)

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

Note: Spectre Direct allows the user to view partial plots while the simulation is running. Click the *Plot* icon at the lower right corner of the ADE L window.

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

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

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

The netlist file is named `netlist` and is written to the [netlist directory](#).

3. Run the Spectre simulator with these options:

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

```
../psf
```

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

Virtuoso Accelerated Parallel Simulator



The APS interface is no longer supported because the APS options can now be set in the Spectre interface. To specify APS options, set `spectre` as the simulator and choose *Setup – High-Performance Simulation*.

ADE provides an interface to the Virtuoso® Accelerated Parallel Simulator (APS). APS is a next generation SPICE simulator that provides high performance, high capacity circuit simulation with full Spectre accuracy. APS achieves maximum simulation performance by enabling multi-threading on multi-core and multi-CPU shared memory systems. This allows you to quickly simulate large pre- and post-layout designs.

APS combines an advanced simulation engine with existing Spectre and Spectre Turbo technologies. It is primarily targeted at speeding up DC and Transient analyses. The APS use model is identical to Spectre, with same netlist syntax, device model, analyses, features, and output format support.

For more information about APS, see the *Virtuoso Accelerated Parallel Simulator User Guide*.

Virtuoso UltraSim Simulator Interface

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

To run Virtuoso UltraSim 64-bit software,

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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

2. Install the Virtuoso UltraSim 64-bit software to the same location as your 32-bit software.
3. Verify that all required software patches are installed by running `checkSysConf` (system configuration checking tool script). The script is located in your local installation of Cadence software:

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

The script is also available on Cadence Online Support, customer support system.

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

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

The list of available executable files is located at:

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

- NONE** invokes all applications as 32-bit.
- "list"** invokes only the executable files included in the list as 64-bit.

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

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

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

Example

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

5. Launch the executable files through the wrapper.

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

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

Example

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

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

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

Virtuoso AMS Designer Simulator Interface

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

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

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

■ Connect Rules

You can point to existing connect rules or create your own by parameterizing existing rules. The form allows you to work with multiple connect rules and auto-compiles all the built in or modified rules. For more information, see Setting Connect Rules on page 77.

■ MATLAB/Simulink.

The ability to run a co-simulation using MATLAB/Simulink with AMS is now available in ADE. You can start MATLAB® before AMS starts by setting specified waiting time and run the cosimulation with general analog simulation flow in ADE. You can also start MATLAB independently in ADE and run the cosimulation just like the EDEN flow in previous release. For more information, see Using MATLAB/Simulink on page 92

■ Global Signals

You can use the Global Signals form to declare a signal that is used as an out-of-module signal reference. For more information, see Working with Global Signals in AMS on page 335.

■ SimVision Integration

Virtuoso Analog Design Environment L User Guide

Environment Setup

You can run simulations interactively using the SimVision debugger, by changing the run mode to interactive. Cross-probing from schematics works with the SimVision integration. For more information, see [Using the SimVision Debugger](#) on page 389.

■ Logfile Utility

You can look at all the individual log files, brought up in an xterm window, or you can use the NCBrowse logfile utility that matches a particular error in a logfile back to the original source. For more information, see [Viewing the Output Log for AMS](#) on page 387.

■ Error Explanation

You can view detailed explanation of the error for AMS in the Error Explanation form. To view an error, you need to enter the error string. However, the errors displayed for AMS are the ones that are present in the log files that are created in the psf directory while a session is being run. For more information, see [Viewing the Error Explanation for AMS](#) on page 388.

■ Default Disciplines

You can specify disciplines on a library, cell, cell terminal, instance, instance terminal and net from the Composer UI. You can autcreate a discrete discipline and ADE auto-compiles the discipline for you. For more information, see [Default Digital Discipline Selection](#) on page 397.

■ Advanced Analyses

Advanced Analyses such as parametric analysis works with AMS. For information on Advanced analysis see, [Virtuoso Analog Design Environment XL User Guide](#).

■ State files from other simulators

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

■ Available Analyses

Transient and AC analyses are available when you use the AMS Designer simulator with ADE. You can save the DC operating point.

■ Outputs

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

■ Single Button *Netlist and Run*

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

Each tool produces a separate log file. When any tool fails, the CIW displays a failure message. Look in the tool's log file for descriptive error messages. For details, refer to the section [Viewing the Output Log for AMS](#) on page 387.

■ Full Support for ADE Display Tools

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

■ OCEAN

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

■ Distributed and Remote Simulations

Use network mode for distributed AMS simulations.

■ Visual Display of Signal Domains

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

■ Display Partition

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

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

Differences to Expect When Using AMS in ADE

For ADE Users

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

- AMS brings a cellview-based netlister to ADE. The cellview-based netlister adds increased speed, flexibility and capacity to ADE. It maintains netlisting compatibility between ADE and AMS. For more information on the AMS Netlister, refer to the [*Virtuoso AMS Environment User Guide*](#).
- Both the *Netlist* command and the *Netlist and Run* command can potentially call several tools to
 - Compile updated text views
 - Netlist updated cellviews
 - As necessary, call several additional tools
 - *ncvlog* or *ncvhdl* to compile
 - *ncelab* to elaborate
 - *ncsim* to simulate

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

- The *NCBrowse* utility helps pinpoint issues in the log files created during compilation, elaboration and simulation. Use *Simulation – Output Log* to activate the *NCBrowse* utility.
- By default ADE does not save outputs. In the ADE Simulation window, do one of the following
 - Select *Outputs – Save All* to save all output signals.
 - Select *Outputs – To Be Saved – Select On Design* and select specific output signals.
 - Select *Outputs – To Be Saved – Select By Subckt Inst* and select the subcircuit instances for which you want to specify the hierarchy levels to save outputs.

Note: In this first release, *SimVision* is very loosely integrated and is available to be used as either a debugger or as a waveform tool. This implies that *SimVision*, is only brought up with the location of the data files. Cross probing from *SimVision* is not available in ADE.

- You cannot directly use the following in designs:

Virtuoso Analog Design Environment L User Guide

Environment Setup

- ❑ Parameters declared in model files. This is due to language boundary issues in Spice and Spectre.
- ❑ Parameters defined in definition files. This is due to language boundary issues in VerilogAMS.
- You can specify a text block as the top level in your configuration as follows:
 - a. Compile your top-level module by hand, for example, by running this command:

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

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

```
master.tag      pc.db      verilog.vams
```
 - b. From the Library Manager, open the cell for editing, make a change and save the view. This adds the following types of files:

```
myLib/textTop/verilogams/verilogAMS.cdb  
myLib/textTop/symbol  
myLib/textTop/prop.xx
```
 - c. In the Hierarchy Editor, create a config view with this text as top.
 - d. Open ADE, and set the schematic to that config.

For AMS Users

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

Specifying Results Directory

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

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

- When you set the simulation directory in ADE using, *Setup – Simulator/Directory/Host form*, the variable will by default be set to `<user_simulator_dir>/<cell>/<simulator>/<view>/psf` and all the data will be stored in this location.

Virtuoso Analog Design Environment L User Guide

Environment Setup

- You can set a UNIX environment variable `AMS_RESULTS_DIR` at the unix prompt. When the variable is set on unix prompt, the variable will be identified, honoured and the simulation data will be stored in `$AMS_RESULTS_DIR/psf`.
- You can also set this variable in the OCEAN script. You would required to set it through OCEAN command `resultsDir("path_for_results_to_store")`. This will store the simulation data to `path_for_results_to_store/psf`.

cds_alias

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

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

More Information on the AMS Designer Simulator

- ❑ *Virtuoso AMS Designer Simulator User Guide*
- ❑ *Virtuoso AMS Environment User Guide*
- ❑ *Virtuoso Mixed-Signal Circuit Design Environment User Guide (IC6.1.6 only)*
- ❑ *Cadence Verilog-AMS Language Reference*
- ❑ *Virtuoso AMS Hierarchy Editor User Guide*
- ❑ *Cadence Application Infrastructure User Guide*
- ❑ *NC Verilog Simulator Help*
- ❑ *NC VHDL Simulator Help*
- ❑ *SimVision User Guide*
- ❑ *Virtuoso Spectre Circuit Simulator User Guide*
- ❑ *Virtuoso Spectre Circuit Simulator Reference*

Mixed-Signal Simulators (IC6.1.6 only)

The following mixed-signal simulators are also provided:

- UltraSimVerilog
- spectreVerilog

For details about UltraSimVerilog, see [Chapter 12, “UltraSimVerilog.”](#) For details about spectreVerilog, see [Virtuoso Mixed-Signal Circuit Design Environment User Guide \(IC6.1.6 only\)](#).

Hspice Direct Interface

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

- Improved Performance in Netlisting

Netlisting is very fast because no raw netlisting is required before the final netlisting. Also, unlike the socket approach, the direct approach supports incremental netlisting. This ensures enhanced performance when incremental updates are performed in a design and then netlisted.

- Better Readability of Netlists

The netlists are truly hierarchical and all numeric values in the netlist are more readable. The sub-circuits are no longer unfolded. The sub-circuits are also no longer mapped unless necessary.

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

A limitation of socket netlisting is that the top cell of a design needs to be editable before the design can be netlisted. The direct approach, however, allows read only designs to be simulated. The only pre-requisite is that the design needs to be extracted first, so that connectivity information is written to the database.

- Advanced Evaluation of Operators

Direct netlisting supports the evaluation of ternary operators (Example, `(iPar("r")>2e-3?200e-3:400e-3)`).

For more information, see [Appendix 11, “Hspice Direct Support.”](#)

Virtuoso Analog Design Environment L User Guide

Environment Setup

Libraries

The following cells of the `analogLib` library are updated to contain *HspiceD* views. The *HspiceD* `simInfo`, CDF parameters and netlisting procedures have been added to all these `analogLib` cells:

bcs	bvs	cap	cccs	ccvs	core
diode	iam	idc	iexp	ind	iopamp
ipulse	ipwl	ipwlf	nnp	isffm	isin
ixfmr	nbsim	nbsim4	njfet	nmes	nmes4
nmos	nmos4	pbsim	pbsim4	pcapacitor	pdiode
pjfet	pmos	pmos4	pnnp	presistor	res
schottky	vam	tline	u1wire	u2wire	u3wire
u4wire	u5wire	usernpn	userpnp	vccap	vccs
vcres	vcvs	vdc	vexp	vpulse	vpwl
vpwlf	vsffm	vsin	winding	xfmr	zener
iprobe	pinductor	mind	pmind	pvccs2	pvccs3
pvcvs	pvcvs2	pvcvs3	pvccs		

Setting Up Simulation Files

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

- Choose *Setup – Simulation Files*.

The Simulation Files Setup form appears.

You can set up include paths, definition files and stimulus files using the Paths/Files tab and setup digital vector files using the Vector Files tab.

Note: The fields that appear on the form depend on your target simulator.

Figure 2-1 Paths/Files Tab

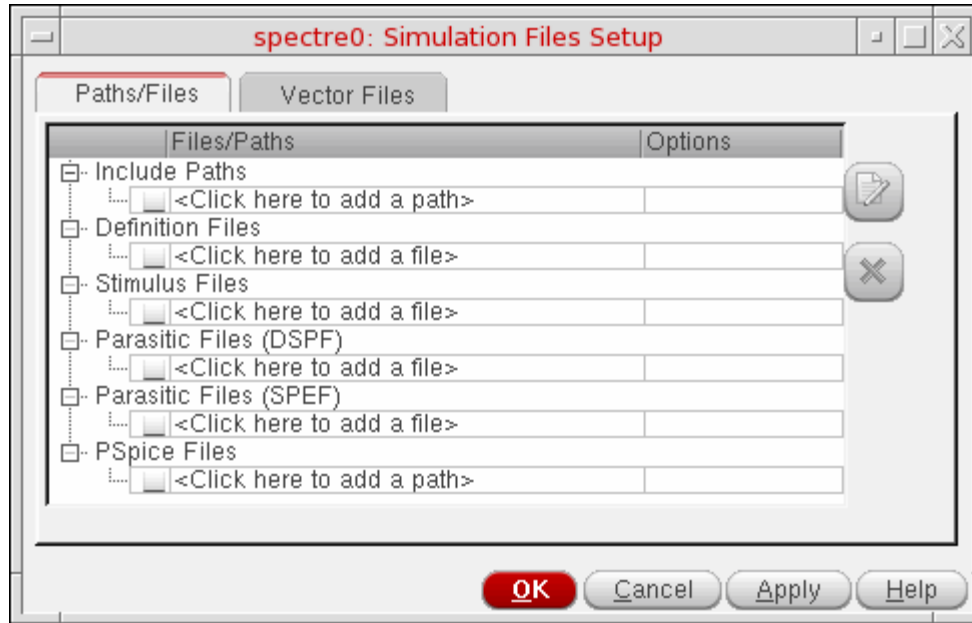
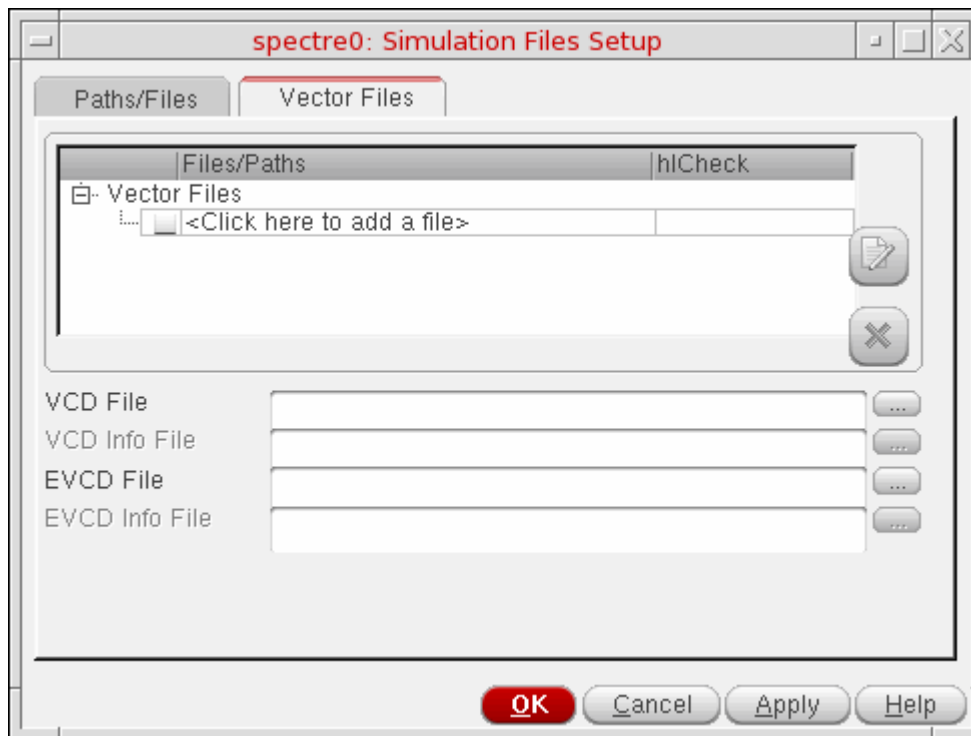


Figure 2-2 Vector Files Tab



See the following sections for more information about using the Simulation Files Setup form:

- [Setting Up Include Paths](#) on page 69
- [Setting Up Definition Files](#) on page 69
- [Setting Up Stimulus Files](#) on page 70
- [Setting Up Vector Files](#) on page 70
- [Setting Up VCD and EVCD Files](#) on page 71
- [Enabling and Disabling Simulation Files](#) on page 73
- [Editing Simulation Files](#) on page 74
- [Deleting Simulation Files](#) on page 74

Setting Up Include Paths

Include paths specify the directories that contain the files you want to include when simulating your design.

To set up include paths, do the following:

1. In the Simulation Files Setup form, click the Paths/Files tab.
2. In the *Include Paths* tree, click where it says *<Click here to add a path>* and type the path to the directory, or click the browse button to select the directory using the Choose Files/Paths form.

The simulator resolves a relative path by looking in the `netlist` directory (relative to where you run the simulation) first. If the path starts with the `.` character, the simulator also resolves this by looking in the `netlist` directory first, then in each of the directories specified in the *Include Path* in the order you type them. The `.` does not mean the current directory.

For information about using variables to specify include paths, see [Specifying Simulation Files Using Variables](#) on page 72.

Setting Up Definition Files

Definition files contain function and parameter definitions that are not displayed in the *Design Variables* section of the simulation window. See the following sample file that contains function and parameter definitions for the Spectre circuit simulator.

```
<your_install_dir>/tools/dfII/samples/artist/models/spectre/definitions.scs
```

Virtuoso Analog Design Environment L User Guide

Environment Setup

The parameters in this file are referenced by included models and are not referenced from any part of the design in the Cadence library.

To set up definition files, do the following:

1. In the Simulation Files Setup form, click the Paths/Files tab.
2. In the *Definition Files* tree, click where it says *<Click here to add a file>* and type the path and file name of your definition file, or click the browse button to select one or more files using the Choose Files/Paths form.

For information about using variables to specify definition files, see [Specifying Simulation Files Using Variables](#) on page 72.

Setting Up Stimulus Files

Stimulus files can contain input and power supply stimuli, initialize nodes, and include estimated parasitics in the netlist. You can look at the following example file that contains Spectre circuit simulator stimuli definitions for the opamp sample design in the aExample library.

```
<your_install_dir>/tools/dfII/samples/artist/models/spectre/opampStimuli.scs
```

In your stimulus file, you can type node names and component names using Open Simulation System (OSS) syntax [*#name*] and the system will substitute the corresponding node numbers when writing the netlist. You can use a backslash (\) to escape a square bracket.

Note: ADE stimulus files follow OSS *simInWithArgs* syntax. For more information about this OSS syntax, refer to the section on SE Functions in Chapter 3 of the [Open Simulation System Reference](#).

To set up stimulus files, do the following:

1. In the Simulation Files Setup form, click the Paths/Files tab.
2. In the *Stimulus Files* tree, click where it says *<Click here to add a file>* and type the path and file name of your stimulus file, or click the browse button to select one or more files using the Choose Files/Paths form.

For information about using variables to specify stimulus files, see [Specifying Simulation Files Using Variables](#) on page 72.

Setting Up Vector Files

To set up vector files, do the following:

1. In the Simulation Files Setup form, click the Vector Files tab.
2. In the *Vector Files* tree, click where it says *<Click here to add a file>* and type the path and file name of your digital vector file, or click the browse button to select one or more files using the Choose Files/Paths form.
3. Click on the *hlCheck* field next to the file and do one of the following:
 - Choose 1 to enable the check for H and L states for input signals (*HLCheck*).
 - Choose 0 to disable the check for H and L states for input signals (*HLCheck*).

For information about using variables to specify vector files, see [Specifying Simulation Files Using Variables](#) on page 72.

Setting Up VCD and EVCD Files

For information about VCD and EVCD stimuli, see “[Verilog Value Change Dump Stimuli](#)” (Chapter 12) in the [Virtuoso UltraSim Simulator User Guide](#).

To setup VCD and EVCD files, do the following:

1. In the Simulation Files Setup form, click the Vector Files tab.
2. In the *VCD File* field type the path and file name of your Verilog value change dump (VCD) file name, or click the browse button to select the file using the Choose a File form.
The *VCD Info File* field becomes active.
3. In the *VCD Info File* field type the path and file name of your signal information file, or click the browse button to select the file using the Choose a File form.
4. In the *EVCD File* field type the path and file name of your “Extended” VCD (EVCD) file, or click the browse button to select the file using the Choose a File form.
5. The *EVCD Info File* field becomes active.
6. In the *EVCD Info File* field type the path and file name of your signal information file, or click the browse button to select the file using the Choose a File form.
7. For information about using variables to specify VCD and EVCD files, see [Specifying Simulation Files Using Variables](#) on page 72.

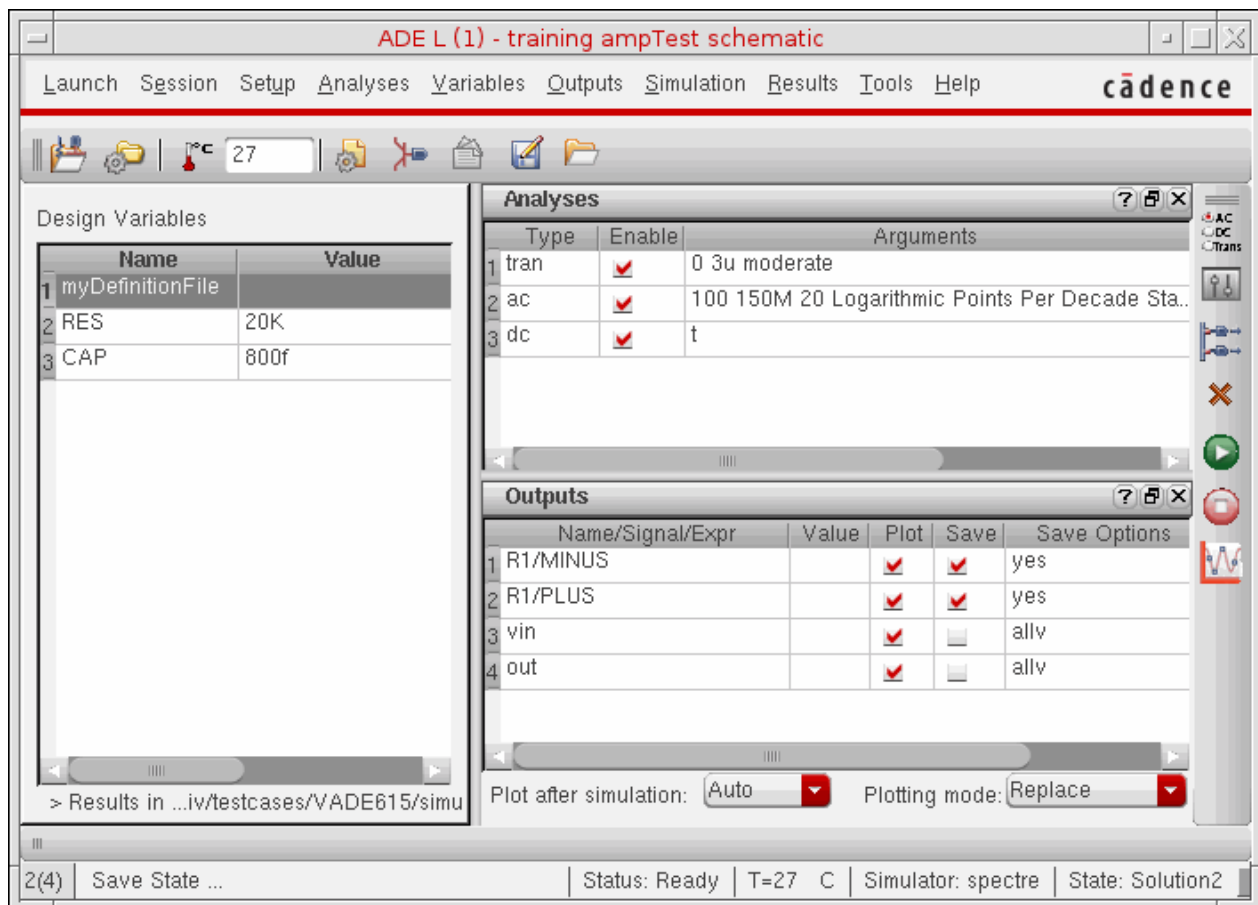
Specifying Simulation Files Using Variables

You can use variables to specify simulation files. Variables allow you to vary the simulation files during simulation.

For example, to specify a definition file using a variable, do the following:

1. In the Simulation Files Setup form, click the Paths/Files tab.
2. In the *Definition Files* tree, click where it says *<Click here to add a file>*.
3. Type a variable name for the definition file you want to vary using the following format:
`VAR("myDefinitionFile")`
4. Click *Apply*.

The `myDefinitionFile` variable appears in the *Design Variables* section of the Simulation window.



- ➔ Double-click in the *Value* field next to the variable and type the path and file name of the stimulus file.

For more information about variables, see [Chapter 7, “Parameterization Support.”](#)

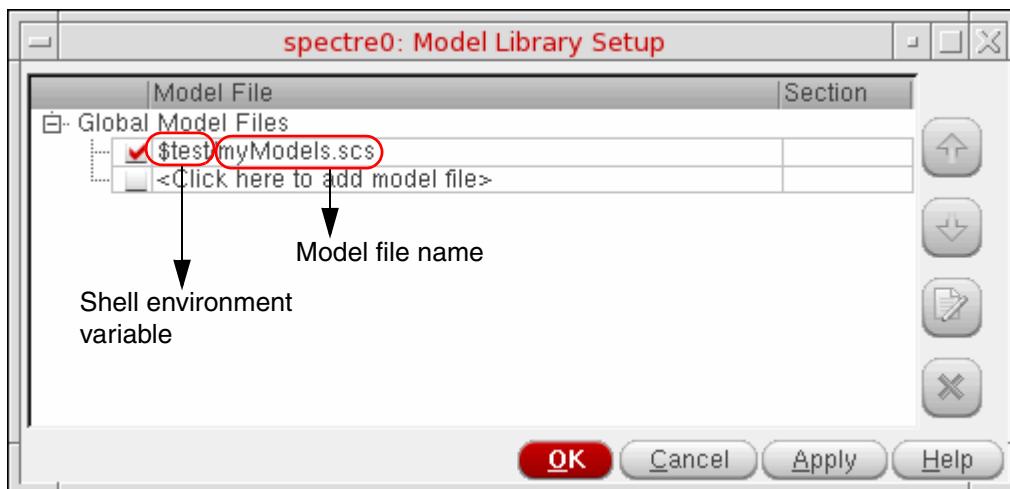
Using Shell Environment Variables to Specify the Paths

You can use shell environment variables to specify the paths of the simulation, vector and model files in the simulation window. Shell environment variables allow you to set different file paths for each session.

For example, to specify a model file using the shell environment variable, do the following:

1. In the *Global Model Files* tree, click the cell with text *<Click here to add model file>*.
2. Type the name of the shell environment variable to include the model file path in the following format:

```
$<shell_variable_name>/<model_file_name>
```



3. Click *Apply*.

Note: The shell environment variables are not displayed in the *Design Variables* section of the simulation window. The paths from the shell environment variables are resolved internally during the simulation run.

Enabling and Disabling Simulation Files


- To enable a simulation file for simulation, select the check box next to it.

Note: By default, the simulation files you add are enabled for simulation.

- To disable a simulation file for simulation, clear the check box next to it.

Editing Simulation Files

To edit simulation files, do the following:

1. Select one or more simulation files.
2. Click the  button.


The simulation files are opened in a text editor.

Important

If you edit simulation files outside ADE, the changes in the files will not be included in the netlist if you click the *Netlist and Run* button or choose *Simulation – Netlist and Run*. You must choose *Simulation – Netlist – Recreate* to include the changes in the netlist.

Deleting Simulation Files

To delete simulation files, do the following:

1. Select one or more simulation files.
2. Click the  button.

Setting Simulation Environment Options

Setting Simulation Environment Options for Direct Simulation

To open the Environment Options form

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

The *Environment Options* form appears.

The screenshot shows the "Environment Options" dialog box. The title bar is "Environment Options" with standard window controls. The dialog contains the following fields and options:

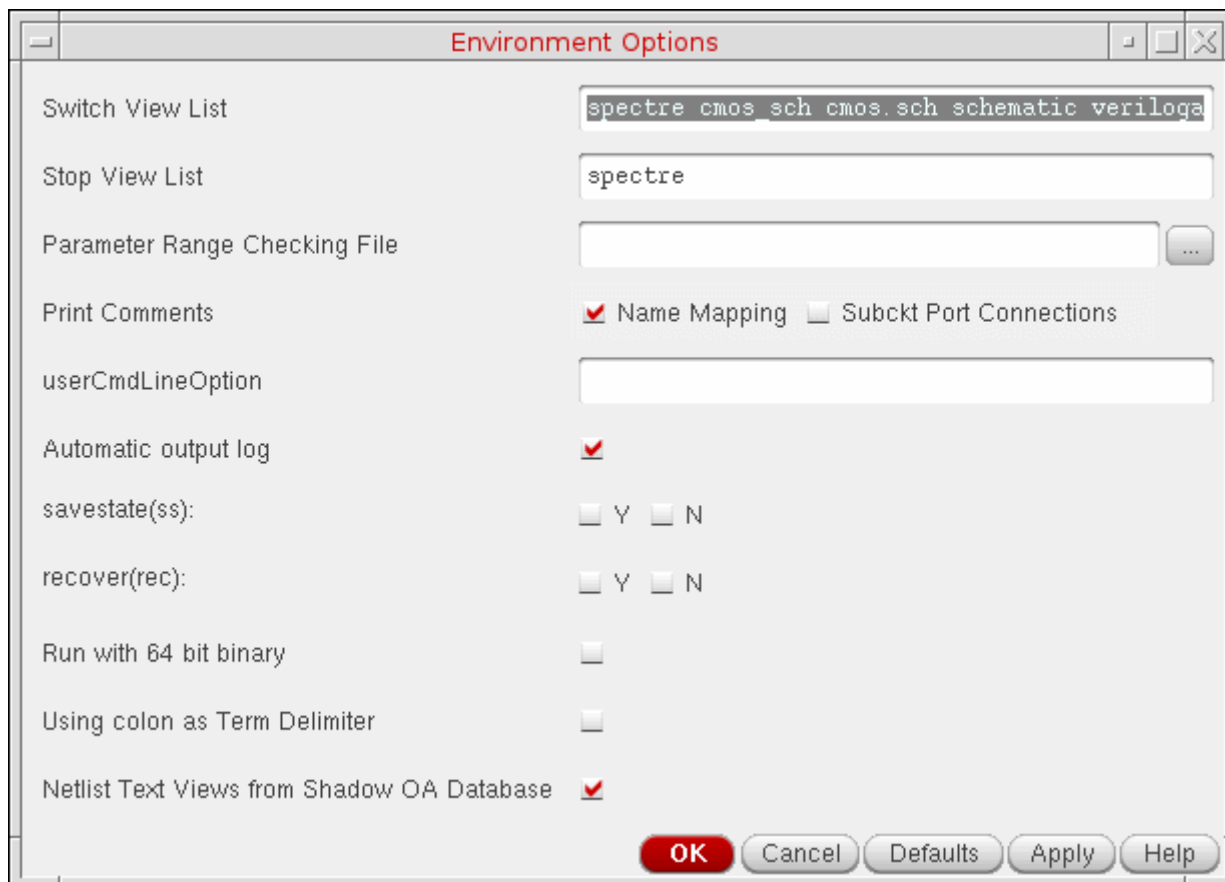
- Switch View List: `spectre cmos_sch cmos.sch schematic veriloga`
- Stop View List: `spectre`
- Parameter Range Checking File: [Empty text box] ...
- Print Comments: Name Mapping Subckt Port Connections
- userCmdLineOption: [Empty text box]
- Automatic output log:
- savestate(ss): Y N
- recover(rec): Y N
- Run with 64 bit binary:
- Using colon as Term Delimiter:
- Set Top Circuit as Subcircuit:

At the bottom, there are five buttons: **OK** (highlighted in red), Cancel, Defaults, Apply, and Help.

Virtuoso Analog Design Environment L User Guide

Environment Setup

Note: When using the *Virtuoso Schematic and Verilog Driven Mixed-Signal Flow* and with the `Virtuoso_MixedSignalOpt_Layout` license checked out, a checkbox, *Netlist Text Views from Shadow OA Database*, is available in the *Environment Options* form.



For detailed information about the form, see “[Environment Options](#)” on page 128.

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

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

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

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

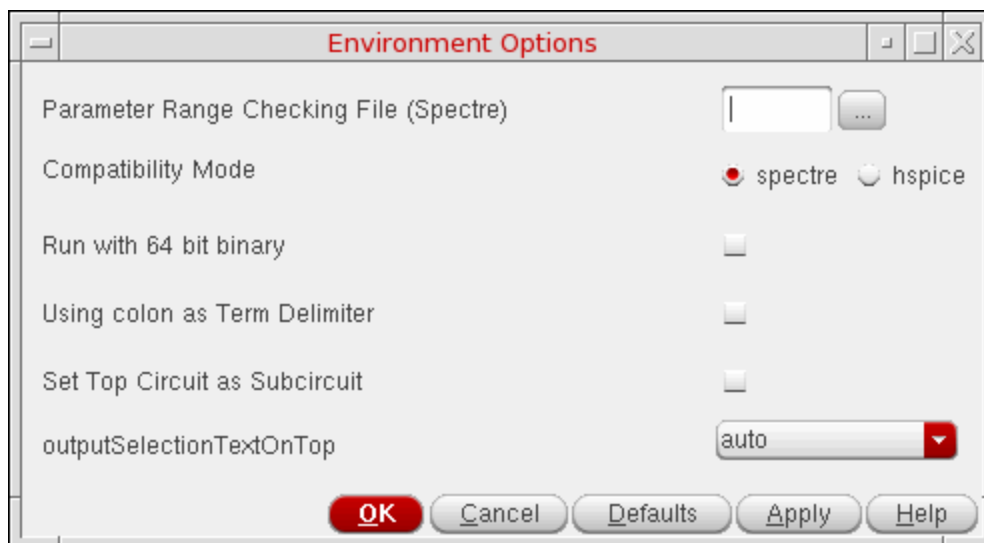
4. Set other options as needed, and click *OK*.

Setting Environment Options for AMS

To set the environment options,

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

The *Environment Options* form appears.



For more information about the form, see [“Environment Options”](#) on page 128.

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

Setting Connect Rules

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

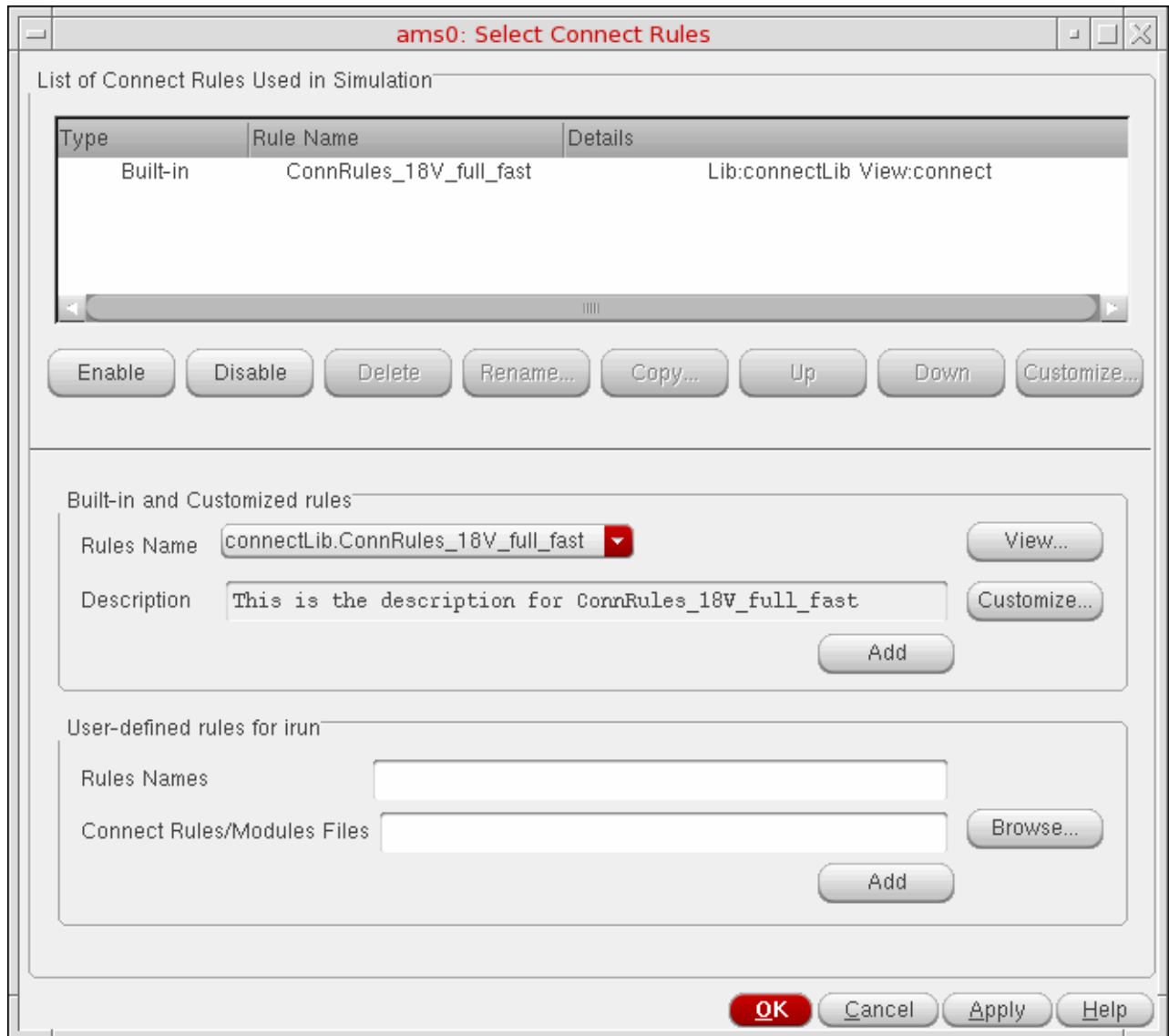
To set connect rules,

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

The *Select Connect Rules* form appears.



The form displays a list of connect rules used in the simulation that need to be passed to the ncelab command line. For default set of connect rules, you can set an environment variable `connectRulesList` in `.cdsinit`. For detailed information on this variable, see [Chapter A, “Environment Variables.”](#)

- ❑ The *List of Connect Rules Used in Simulation* table displays the connect rules that will be used when the run mode is *Cellview-based netlister with ncvlog*,

ncelab, *ncsim*. For more information about the *Cellview-based netlister with ncvlog*, *ncelab*, *ncsim* run mode, see [Choosing the AMS Netlister](#) on page 365.

Each row in the table has the following details:

- *Type*—Displays the type of the connect rule as `Built-in`, `User-defined` or `Modified built-in`.
- *Rules Name* —Displays the name of the rule, which is the cell name.
- *Details*—Displays the name of the library in which the rule exists and the view name for the rule.

Note: You can use the `genConnRulesFile` command to display your user-defined rules as built-in rules in the *Built-in rules* group box. For more information about the `genConnRulesFile` command, see [connectRules.il File](#) on page 84.

Adding Connect Rules

You can add built-in or user-defined connect rules.

- To add a built-in connect rule,
 - a. Select the *Built-in* option button.
 - b. In the *Built-in rules* group box, select a rule from the *Rules Name* pull-down list.

Virtuoso Analog Design Environment L User Guide

Environment Setup

- c. If you want to view the rule, click the *View* button to open the connect rules file in a text editor.



```

/servers/cicstore/cic_pv/CIC_IUS55/Inx86/tools.Inx86/affirma_ams/etc/connect_lib/ConnRules
File Help cadence
// 'ConnRules_5V.vams' - Verilog-AMS 5 volt connection rules file.
// last revised: 10/22/02 (ronv)

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

`define Vsup 5.0
`define Vthi 3.5
`define Vtlo 1.5
`define Tr 1n
`define Rlo 200
`define Rhi 200
`define Rx 40
`define Rz 10M

connectrules ConnRules_5V_full;
  connect L2E #(
    .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo),
    .tr(`Tr), .tf(`Tr), .tx(`Tr), .tz(`Tr),
    .rlo(`Rlo), .rhi(`Rhi), .rx(`Rx), .rz(`Rz) );
  connect E2L #(
    .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo), .tr(`Tr) );
  connect Bidir #(
    .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo),
    .tr(`Tr), .tf(`Tr), .tx(`Tr), .tz(`Tr),
    .rlo(`Rlo), .rhi(`Rhi), .rx(`Rx), .rz(`Rz) );
endconnectrules

connectrules ConnRules_5V_mid;
  connect E2L #( .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo), .tr(`Tr) );
  connect L2E_1 #( .vsup(`Vsup), .tr(`Tr), .rout(`Rlo) );
  connect Bidir_0 #( .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo),
    .tr(`Tr), .rout(`Rlo) );
endconnectrules

connectrules ConnRules_5V_basic;
  connect E2L_0 #( .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo), .tr(`Tr) );
  connect L2E_0 #( .vsup(`Vsup), .tr(`Tr), .rout(`Rlo) );
  connect Bidir_0 #( .vsup(`Vsup), .vthi(`Vthi), .vtlo(`Vtlo),
    .tr(`Tr), .rout(`Rlo) );

```

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

For more information about customizing built-in rules, see [Customizing Built-in Rules](#) on page 87.

Note: When you customize a built-in rule, its type appears as *Modified built-in* and a number is suffixed to its name. For example, a built-in rule `ConnRules_3V_mid`, when customized, is renamed to `ConnRules_3V_mid1`.

e. Click the *Add* button to add the rule.

- To add a user-defined connect rule to be used when the run mode is *Cellview-based netlister with ncvlog, ncelab, ncsim*, do the following. For more information about the *Cellview-based netlister with ncvlog, ncelab, ncsim* run mode, see [Choosing the AMS Netlister](#) on page 365.

a. Select the *User-defined(ncvlog, ncelab, ncsim)* option button.

b. Click *Browse* in the *User-defined rules for ncvlog, ncelab, ncsim* group box to display the Library Browser form.

c. Select the library, cell and view in which the rule exists and click *Close*.

d. Click the *Add* button to add the rule.

The rule is displayed in the *List of Connect Rules Used in Simulation* table.

- To add a user-defined connect rule to be used when the run mode is *OSS-based netlister with irun*, do the following. For more information about the *OSS-based netlister with irun* run mode, see [Choosing the AMS Netlister](#) on page 365.

a. Select the *User-defined (irun)* option button.

b. In the *Rules Names* field, enter the name of the connect rule to be used by *irun*.

c. In the *Connect Rules/Modules Files* field, enter the filename and path of the connect rule and module file in which the specified rule exists. You can use the *Browse* button to select the file and click *OK* to add them in the field.

You can also use environment variables to specify the path to the file. For example, you can specify the path as:

```
$CONNECT_FILES/myrules.vams
```

Note the following:

- If you try entering more than one Rules name at a time, you will get a pop-up error saying that you can enter only one rule name at a time.

Virtuoso Analog Design Environment L User Guide

Environment Setup

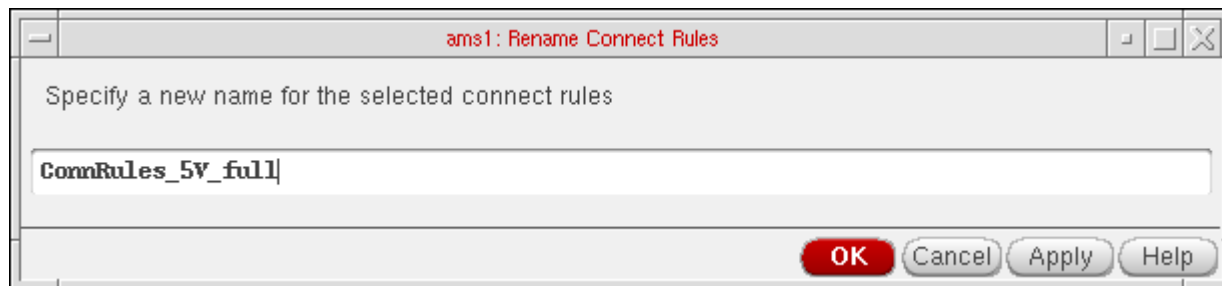
- ❑ If a rule with the same name exists in more than one connect rule file, the rule that exists in the first connect rule file displayed in the *Connect Rules/Modules Files* field will be used.
- ❑ If you do not specify a rule name in the *Rules Names* field, the first rule in the first connect rule file specified in the *Connect Rules/Modules Files* field will be used.
- ❑ The user-defined connect rules specified in the *Rules Names* field will not be displayed in the *List of Connect Rules Used in Simulation* table.
 - a. The built-in connect rules displayed in the *List of Connect Rules Used in Simulation* table and the connect rules specified in the *User-defined rules for irun* group box will be used in the simulation. So you can have a mixture of both built-in and user-defined rules in one simulation.

Renaming Connect Rules

To rename a rule in the *List of Connect Rules Used in Simulation* table,

1. Select a rule from the table.
2. Click the *Rename* button.

The Rename Connect Rules pop-up box appears.



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

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

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

Deleting Connect Rules

To delete a rule in the *List of Connect Rules Used in Simulation* table,

1. Select one or more rules in the *List of Connect Rules Used in Simulation* table.
2. Click the *Delete* button.

Copying Connect Rules

To copy a rule in the *List of Connect Rules Used in Simulation* table,

1. Select a rule in the *List of Connect Rules Used in Simulation* table.
2. Click the *Copy* button.

The *Copy Connect Rules* form appears.



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

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

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

4. Click *OK*.

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

Changing the Sequence of Connect Rules

To change the sequence of the rules displayed in the *List of Connect Rules Used in Simulation* table,

1. Select a rule in the *List of Connect Rules Used in Simulation* table.
2. Click the *Up* or *Down* buttons.

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

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

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Connect Modules* option in the Saving State form and save the current ADE state. You can later load the state using the Loading State form with the *Connect Modules* option selected. For more information, see [Saving and Restoring the Simulation Setup](#) on page 97.

connectRules.il File

The `connectRules.il` file contains the built-in connect rules that are displayed in the [Select Connect Rules](#) form.

ADE searches for the `connectRules.il` file in the following locations. If multiple `connectRules.il` files are located, their contents are concatenated and displayed in the [Select Connect Rules](#) form.

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

All Cadence-supplied connect rules are stored in the `connectRules.il` file in the `/tools/affirma_ams/etc/connect_lib/connectLib` directory in your Cadence Incisive Unified Simulator (IUS) installation directory.

You can use the `genConnRulesFile` command to automatically compile your user-defined connect rules into a custom `connectRules.il` file.

Virtuoso Analog Design Environment L User Guide

Environment Setup

```
genConnRulesFile [-help] [-destpath <destination path>] -lib <lib1> <file1>
[<file2 ...>] [-lib <lib2> <file3> [<file4 ...>]] [-rulefile <rulesFile>] [-rule
<ruleName1> [<ruleName2 ...>]]
```

where:

- `-help` displays the help for the `genConnRulesFile` command.
- `-destpath <destination path>` specifies the directory in which the `connectRules.il` file will be created.
- `-lib <lib1> <file1> [<file2> ...]` specifies the libraries in which the connect rule files exist and the connect rule files that need be parsed. All the connect rules in the specified rule files are included in the `connectRules.il` file.

For example, to include all the connect rules in the `crules1.vams` and `crules2.vams` rule files that exist in the `mylib` library, use the following command:

```
genConnRulesFile -destpath . -lib mylib mylib/crules1.vams mylib/crules2.vams
```

In this example, the `connectRules.il` file is created in the current directory.

Note: If you use the `-rulefile` or `-rule` option, only the rules specified by the `-rulefile` or `-rule` option will be included in the `connectRules.il` file.

- `-rulefile <rulesFile>` specifies the name of a file which contains the list of connect rules that need to be included in the `connectRules.il` file.

For example, to include only the connect rules named `cr1`, `cr2`, `cr8` and `cr9` in the `connectRules.il` file, do the following:

- a. Create a file named `myrules` with the following text:

```
cr1
cr2
cr8
cr9
```

- b. Use the following command:

```
genConnRulesFile -destpath . -lib mylib mylib/crules1.vams mylib/
crules2.vams -rulefile mylib/myrules
```

In this example, even though the `-lib` option specifies the `crules1.vams` and `crules2.vams` rule files, only the rules listed in the `myrules` file are included in the `connectRules.il` file.

- `-rule <ruleName1> [<ruleName2> ...]` specifies the names of the rules in the specified rule files that need to be included in the `connectRules.il` file.

Virtuoso Analog Design Environment L User Guide

Environment Setup

For example, to include only the connect rules `cr1` and `cr2` in the `crules1.vams` rules file and the connect rules `cr8` and `cr9` in the `crules2.vams` rules file, use the following command:

```
genConnRulesFile -destpath . -lib mylib mylib/crules1.vams mylib/crules2.vams  
-rule cr1 cr2 cr8 cr9
```

In this example, even though the `-lib` option specifies the `crules1.vams` and `crules2.vams` rule files, only the rules specified using the `-rule` option are included in the `connectRules.il` file.

If you place your custom `connectRules.il` files in one of the locations where ADE searches for `connectRules.il` files, the connect rules in the custom files are also displayed as built-in rules in the Select Connect Rules form.

Customizing Built-in Rules

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

The dialog box is titled "ams1: Customize Built-in Rules". It contains the following elements:

- Description:** A text field containing "This is the description for ConnRules_5V_full".
- Connect Module Declarations:** A table with columns "Module", "Mode", and "Parameter/Values".

Module	Mode	Parameter/Values
L2E		vsup=5.0 vthi=3.5 vtlo=1.5 tr=1n tf=1n tx=1n tz=1n rlo=200 r...
E2L		vsup=5.0 vthi=3.5 vtlo=1.5 tr=1n
Bidir		vsup=5.0 vthi=3.5 vtlo=1.5 tr=1n tf=1n tx=1n tz=1n rlo=200 r...
- Buttons:** "Change" and "View connect module..."
- Mode:** A dropdown menu.
- Parameters:** A table with columns "Parameter" and "Value".

Parameter	Value
vsup	5.0
vthi	3.5
vtlo	1.5
tr	1n
tf	1n
tx	1n
tz	1n
- Input Fields:** "Parameter" and "Value" text boxes with a "Change" button.
- Direction and Discipline:** "Direction1" and "Direction2" dropdowns, and "Discipline1" and "Discipline2" text boxes.
- Buttons:** "Connect Resolutions...", "OK", "Cancel", "Apply", "Disciplines...", "Help".

You use this form as follows:

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

You may need to scroll to the right to see these and the remaining columns.

- The discipline information for the first port appears in the *Discipline1* column and for the second port under *Discipline2*. These columns may be blank for modules that do not have them specified.

Select a module by clicking on the related row.

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

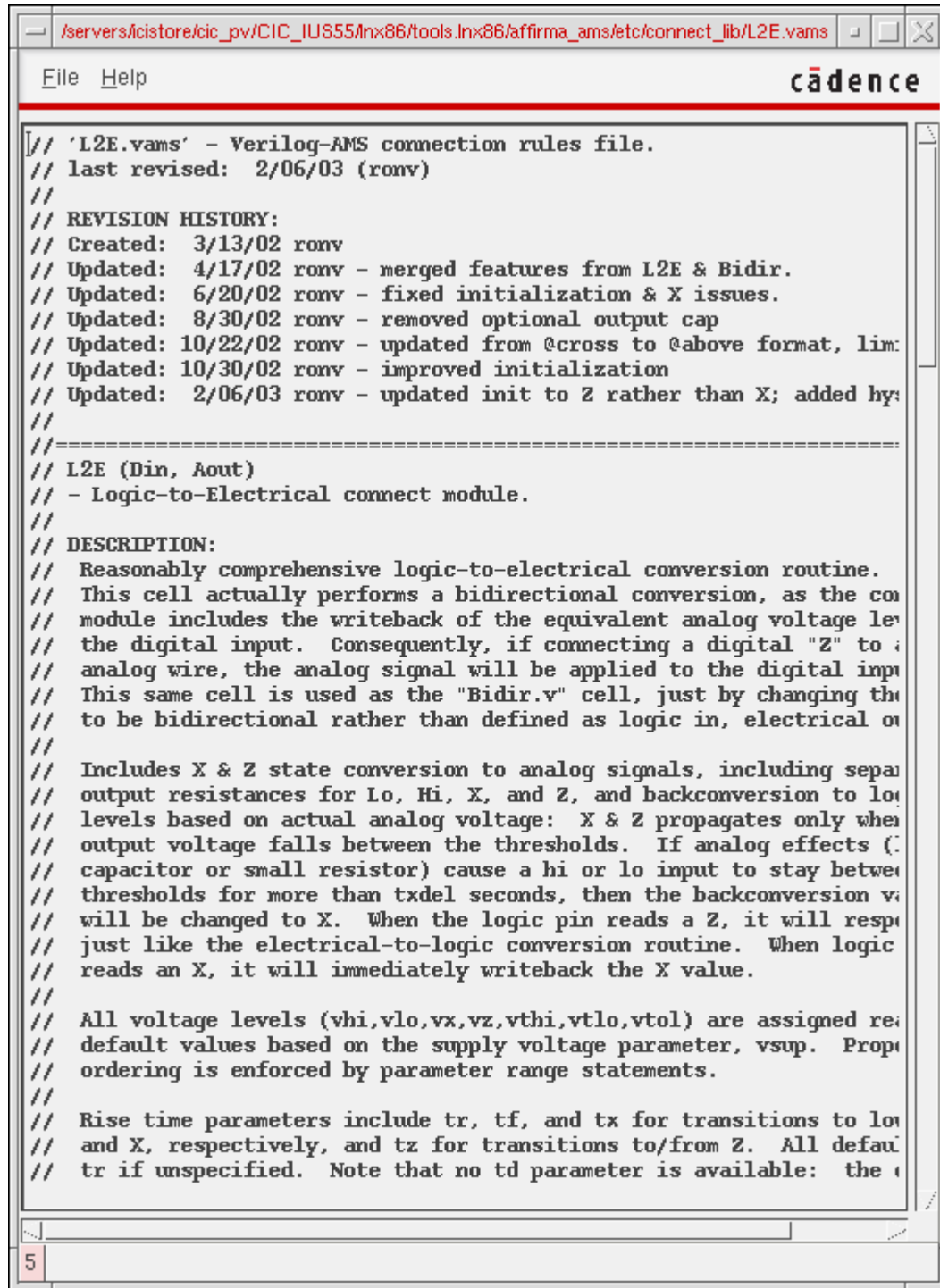
When you select a row, the *Parameters* table is also populated. You can change values by selecting a row in this table, modifying the *Value* and clicking the *Change* button next to it.

4. If you want to see the connect module file, click the *View connect module* button.

Virtuoso Analog Design Environment L User Guide

Environment Setup

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



```
// 'L2E.vams' - Verilog-AMS connection rules file.
// last revised: 2/06/03 (ronv)
//
// REVISION HISTORY:
// Created: 3/13/02 ronv
// Updated: 4/17/02 ronv - merged features from L2E & Bidir.
// Updated: 6/20/02 ronv - fixed initialization & X issues.
// Updated: 8/30/02 ronv - removed optional output cap
// Updated: 10/22/02 ronv - updated from @cross to @above format, lim
// Updated: 10/30/02 ronv - improved initialization
// Updated: 2/06/03 ronv - updated init to Z rather than X; added hys
//
//=====
// L2E (Din, Aout)
// - Logic-to-Electrical connect module.
//
// DESCRIPTION:
// Reasonably comprehensive logic-to-electrical conversion routine.
// This cell actually performs a bidirectional conversion, as the con
// module includes the writeback of the equivalent analog voltage lev
// the digital input. Consequently, if connecting a digital "Z" to a
// analog wire, the analog signal will be applied to the digital input
// This same cell is used as the "Bidir.v" cell, just by changing the
// to be bidirectional rather than defined as logic in, electrical on
//
// Includes X & Z state conversion to analog signals, including separ
// output resistances for Lo, Hi, X, and Z, and backconversion to log
// levels based on actual analog voltage: X & Z propagates only when
// output voltage falls between the thresholds. If analog effects (.
// capacitor or small resistor) cause a hi or lo input to stay betwe
// thresholds for more than txdel seconds, then the backconversion va
// will be changed to X. When the logic pin reads a Z, it will respo
// just like the electrical-to-logic conversion routine. When logic
// reads an X, it will immediately writeback the X value.
//
// All voltage levels (vhi,vlo,vx,vz,vthi,vtlo,vtol) are assigned re
// default values based on the supply voltage parameter, vsup. Prop
// ordering is enforced by parameter range statements.
//
// Rise time parameters include tr, tf, and tx for transitions to lo
// and X, respectively, and tz for transitions to/from Z. All defau
// tr if unspecified. Note that no td parameter is available: the c
```

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

Virtuoso Analog Design Environment L User Guide

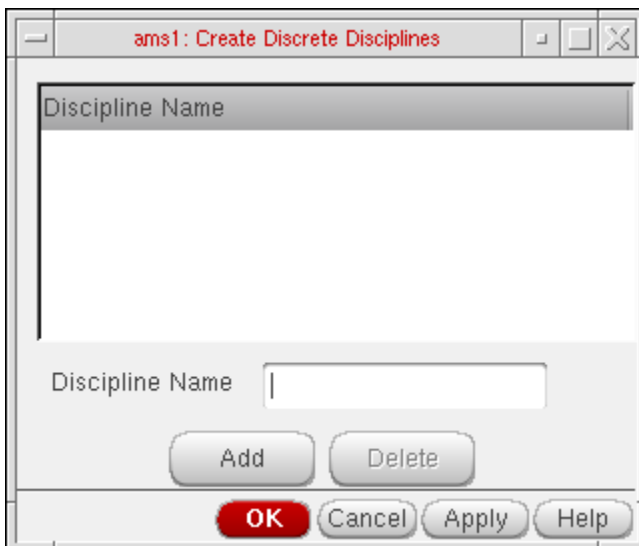
Environment Setup

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

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

6. The *Direction1* and *Direction2* fields are blank by default. The possible combinations of values you may specify are:
 - input, output*
 - output, input*
 - inout, inout*
7. The *Discipline1* and *Discipline2* fields are also blank by default. They may remain blank or you may specify a discipline-pair for the selected module.

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



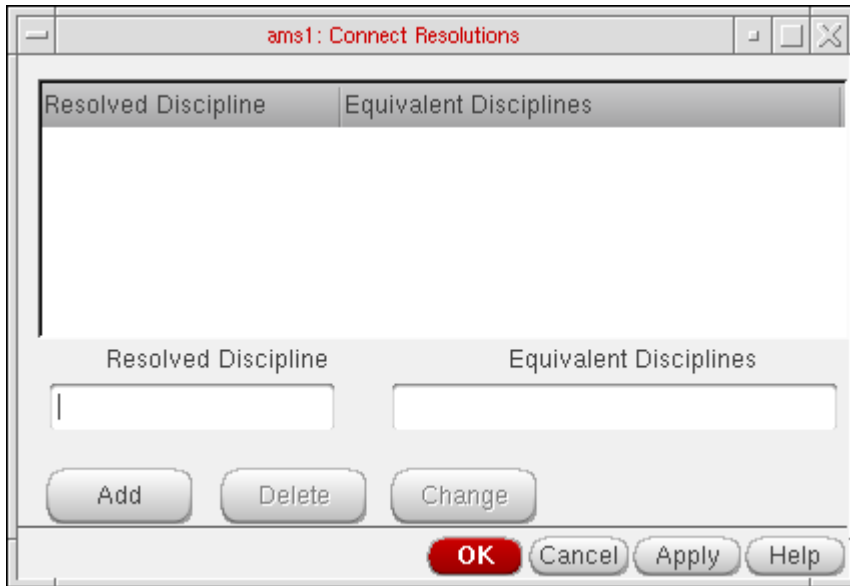
It shows a list of disciplines. You may type in a name in the *Discipline Name* field and click *Add* to create a new discipline. The name should be a legal verilog identifier. You may select one or more discipline names listed in the table and click the *Delete* button to delete them. This information is saved for the session when you click *OK* and is

Virtuoso Analog Design Environment L User Guide

Environment Setup

included in the connect modules information saved in a state file. For information on states, see [Saving and Restoring the Simulation Setup](#) on page 97.

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



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

- You can add a new association by specifying a *Resolved Discipline* and an *Equivalent Discipline* and clicking the *Add* button.
 - You can modify an association by selecting a row, modifying the values in the *Resolved Discipline* and *Equivalent Discipline* fields and clicking the *Change* button.
 - You can select one or more rows and delete them by clicking the *Delete* button.
9. Click the *OK* button to update the connect rules information with the changes made for connect resolutions.
 10. Click the *OK* button in the *Customize Built-in Rules* form to update the connect rules information for the session.

Using MATLAB/Simulink

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

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

This user guide describes the ADE integration of MATLAB. For details on other two flows, refer to *Virtuoso AMS Designer Simulator User Guide*.

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

Setting up the AMS/MATLAB Cosimulation

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

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

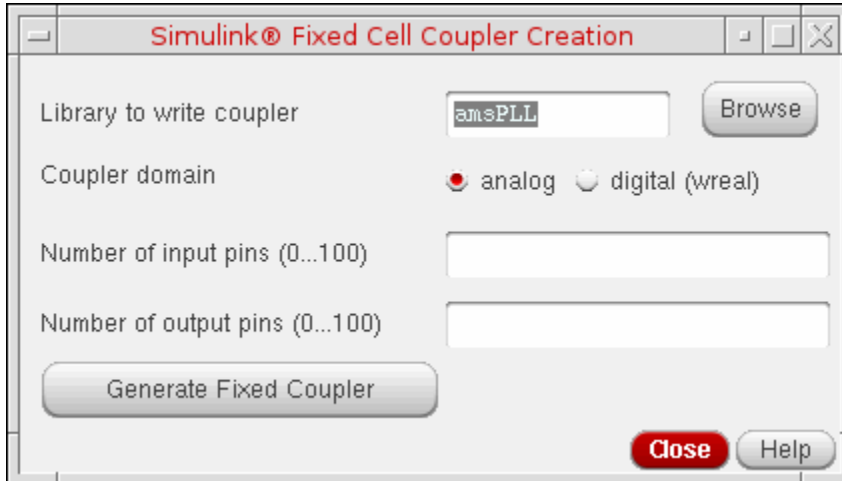
If there are specific input/output configurations that you know you will use frequently, you can create a fixed cell coupler and use that fixed cell in your schematic rather than a pcell. The verilog.vams file will be created for the fixed cell when the fixed cell is created. To create a fixed cell coupler:

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

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

2. Select *AMS - Simulink Coupler Creation*.

The Simulink Coupler Fixed Cell Creation form opens.



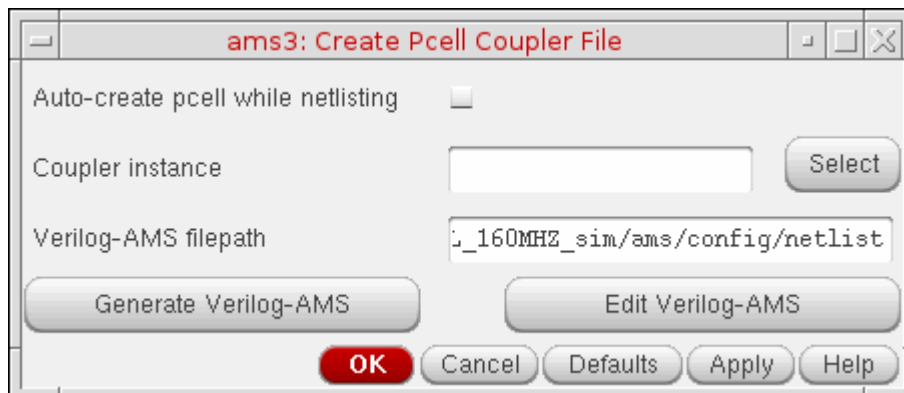
3. Enter the Library name where you want the fixed coupler to be placed. You may want to create a coupler library for all the fixed couplers that you would require or create them in your design libraries. Enter the coupler domain, Number of input and output pins and click *Generate Fixed Coupler* button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

coupler_<numInputPorts>_<numOutputPorts>_[d, a]

Starting MATLAB

You can start MATLAB directly from ADE. If your design is using pcell coupler blocks, then you need to create a verilog.ams file for the pcell coupler block before netlisting. In the simulation window, choose Setup – MATLAB/Simulink – Create Pcell Coupler File. The Create Pcell Coupler File form appears.

:



Virtuoso Analog Design Environment L User Guide

Environment Setup

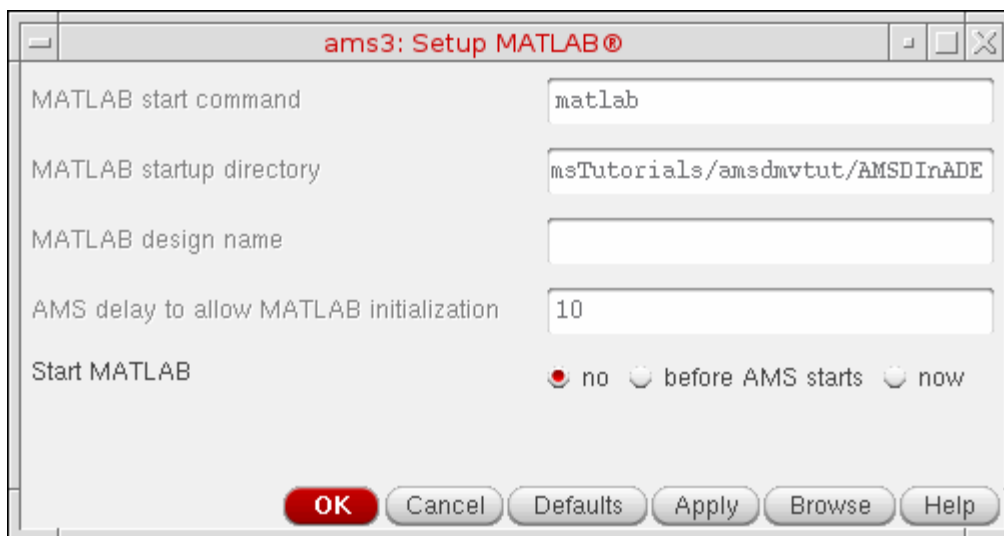
- You can either directly enter the Coupler instance or select the same from the schematic by clicking the Select button.

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

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

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

The Setup MATLAB form appears.



1. Enter the command for starting MATLAB using MATLAB start command field. The default command to start MATLAB is “matlab”. You can enter any other command along with the command line argument.
2. Enter the path to the directory where matlab is launched in the MATLAB startup directory field.
3. Enter the path to the MATLAB design in the *MATLAB design name* field.
4. Enter simulation initialization timeout in AMS delay to allow MATLAB initialization. It is required to give Simulink a chance to start up before AMS. After Simulink is initialized, you can run the simulation.

5. Do one of the following:

Select	If
no	If you select no option in Start MATLAB, MATLAB will not be started. This may be useful if you want to keep same form setup but do not want to run cosimulation or if you want to start MATLAB yourself, independent of ADE.
before AMS starts	If you want to start MATLAB automatically, select the before AMS starts check box in Start MATLAB section. When you select the before AMS starts option, all the fields in the form are enabled. To run Simulink simulation automatically when AMS starts, you need to have a startup.m file that contains the sim() command in the directory, which is specified in the MATLAB startup directory field.
now	If you want to launch MATLAB manually, select now option in start MATLAB. The Start button appears and the AMS delay to allow MATLAB initialization is disabled. Click Start button to launch MATLAB.

Setting Up a Remote Simulation

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

1. As described in the topic [Choosing a Simulator](#) on page 48, choose *Setup – Simulator/Directory/Host*.

The *Choosing Simulator/Directory/ Host* form appears.

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

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

Note: Do not use a relative path, for example, ~/simulation, when setting up a remote simulation directory. Instead, use the full path.

4. Set *Host Mode* to *remote*.
5. Type the name of the host system that will run the simulation.

6. Type the path to your project directory relative to the remote system.

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

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

Basic Requirements for Running a Remote Simulation

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

Using a Third-Party Simulator for Remote Simulations

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

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

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

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

The script to run the simulation is a local file.

Scripts for Using Third-Party Simulators in Remote Simulations

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

```
cd your_install_dir/bin
```

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

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

```
mv hspiceArtRem ~/spectre
```

for running the Spectre simulator remotely.

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

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

For example,

```
which spectre
```

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

4. Edit the script and make these changes:

- a. Change the machine name from `cds8715` to the name of the remote host running the simulator.
- b. Change `/usr/meta/bin/hspice` to the directory where the simulator is located on the remote machine.
- c. On HP systems only, because the `rsh` command is called `remsh`, remove the comment character from the line

```
remshell=remsh
```

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

Saving and Restoring the Simulation Setup

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

Saving the waveform setup using the *Saving State* form saves the same information as the *File – Save as* command in the graph window.

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

Saving the Simulation Setup

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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

The Saving State form appears.

Saving State -- ADE L (1)

Save State Option: Directory Cellview

Directory Options

State Save Directory:

Save As:

Existing States: Solution2, Solution3, Solution4, dependent, deviceChecks

Cellview Options

Library:

Cell:

State:

Description

What to Save

Analyses Variables Outputs

Subckt Inst Model Setup Simulation Files

Environment Options Simulator Options Convergence Setup

Waveform Setup Graphical Stimuli Conditions Setup

Results Display Setup Device Checking Setup RelXpert Setup

Cosimulation Options Performance/Parasitic Reduction MDL Control Setup

Distributed Processing

For detailed information about the form, see [“Saving State”](#) on page 130.

3. Select either of the option buttons *Directory* or *Cellview* to indicate that you want to save the state into a directory or as a cellview. The default option is *Directory*.
 - a. When the *Directory* option is selected, the fields in the *Directory Options* group box appear enabled. You need to specify a path in the *State Save Directory* by typing it in or by using the *Browse* button to locate it. The *Existing States* list shows all the states saved in that location. You need to also specify a name for the new state in the *Save As* field by selecting one of existing state names to overwrite it or by typing a new name.
 - b. When you select the *Cellview* option, the fields in the *Cellview Options* group box appear enabled and those in the *Directory Options* group box appear disabled. You need to specify a *Library* and *Cell* where you want to save the state you specify in the *State* field. You may either type these in or specify them using the *Browse* button.

If you choose an existing state from a library that is data-managed, and click *OK* and if your *Auto Checkin* and *Auto Checkout Preferences* are enabled, the existing state is checked out and the state being saved is checked in.

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

4. Use the *Description* field to enter a short description about the current state.
5. The substates displayed in the *What to Save* section are also enabled or disabled according to the substates in the selected state. You can enable or disable any of the substates individually by selecting the check boxes next to them. You may also select all of them or none of them collectively by using the *Select All* or *Clear All* check boxes at the upper left corner of the *What to Save* group box.
6. Click *OK* to save the specified state.

Restoring the Simulation Setup



When a Spectre simulation is running in interactive mode, do not restore a simulation setup in which Spectre Turbo mode or parasitic reduction is enabled. Restoring the simulation setup will result in the simulation run being terminated. For more information about enabling Spectre Turbo mode and parasitic reduction, see [Specifying Performance and Parasitic Reduction Options for the Spectre Simulator](#) on page 304.

Virtuoso Analog Design Environment L User Guide

Environment Setup

To restore all or part of a saved setup,

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

The *Loading State* form appears.

Loading State -- ADE L (1)

Load State Option: Directory Cellview

Directory Options

State Load Directory:

Library:

Cell:

Simulator:

State Name:
Solution3
Solution4
dependent
deviceChecks

Cellview Options

Library:

Cell: Simulator:

State:

Description

None

What to Load

<input checked="" type="checkbox"/> Analyses	<input checked="" type="checkbox"/> Variables	<input checked="" type="checkbox"/> Outputs
<input type="checkbox"/> Subckt Inst	<input checked="" type="checkbox"/> Model Setup	<input checked="" type="checkbox"/> Simulation Files
<input checked="" type="checkbox"/> Environment Options	<input checked="" type="checkbox"/> Simulator Options	<input checked="" type="checkbox"/> Convergence Setup
<input type="checkbox"/> Waveform Setup	<input checked="" type="checkbox"/> Graphical Stimuli	<input type="checkbox"/> Conditions Setup
<input type="checkbox"/> Results Display Setup	<input checked="" type="checkbox"/> Device Checking Setup	<input type="checkbox"/> RelXpert Setup
<input checked="" type="checkbox"/> Cosimulation Options	<input type="checkbox"/> Performance/Parasitic Reduction	<input type="checkbox"/> MDL Control Setup
<input type="checkbox"/> Distributed Processing		

For detailed information about the form, see “[Loading State](#)” on page 132.

2. Select either of the option buttons *Directory* or *Cellview* to indicate that you want to load the state from a directory or from a cellview. The default option is *Directory*.
 - a. When the *Directory* option is selected, the fields in the *Directory Options* group box appear enabled. You need to specify a path in the *State Load Directory* by typing it in or by using the *Browse* button to locate it. Select the desired values in the *Library*, *Cell*, and *Simulator* fields. *State Name* specifies all the saved states for the cell and simulator combination that you specified. Select the state name you want to load.
 - b. When you select the *Cellview* option, the fields in the *Cellview Options* group box appear enabled and those in the *Directory Options* group box appear disabled. You need to specify a *Library*, *Cell* and *State* you want to load. You may either type these in or specify them using the *Browse* button.
3. The *Description* field shows a short description about the selected state.
4. The substates displayed in the *What to Load* section are based on the substates in the selected state. You can enable or disable any of the substates individually by selecting the checkboxes next to them. You may also select all of them or none of them collectively by using the *Select All* or *Clear All* checkboxes at the upper left corner of the *What to Save* group box.
5. You can click *OK* to load the specified state.

The system restores as much of the information as possible, ignoring settings from other simulators that are incompatible with the simulator that you have selected.

6. To delete a selected state, click the *Delete State* button.

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

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

Panic State Saving

As discussed in the previous section, you can save and load the state of the simulation environment setup at any time. However, if you have not saved the state for any current session and Virtuoso fails because of an unexpected reason, you might lose the unsaved settings. The Analog Design Environment by default saves the state in such cases of

unexpected failures and automatically prompts you to recover the last session when you launch ADE again. This is also called as panic state saving.

In case of panic state, the settings are saved in the form of SKILL code in a text file. If you choose to restore the settings, ADE loads the settings and deletes the text file. If you choose not to restore the settings, ADE deletes the text file and does not prompt in the subsequent sessions.

The text files with SKILL code are saved in a directory named `.ADE_Panic_State` at a path specified using the `adePanicStatePath` environment variable in the `.cdsinit` file. If this variable is not defined, the `.ADE_Panic_State` directory is created in the current simulation directory.

Panic state saving is done only for:

- the first ADE session, in case multiple sessions of ADE are opened from a single CIW and the CIW fails
- the ADE session that has some unsaved settings
- the ADE session that is valid (session has valid designs)
- the CIW session that is not in replay mode

ADE prompts to restore a panic state only if:

- in the new ADE session, after the last failure, the design is same as that of the session that failed
- no design is opened before the new ADE session
- the new CIW session is not started in replay mode

Important

The panic state saving feature is enabled by default. You can disable this feature by setting the `allowAdePanicStateSaving` variable to `nil` in the `.cdsinit` file.

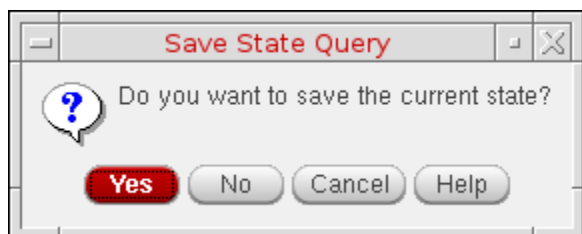
However, it is important that the value of this variable is set as `t` at both times, during the session when an unexpected failure occurred and during the next ADE session.

Closing a Session

To close an ADE L session, do the following:

- ➔ Choose *Session – Quit*.

If the state of your environment is not saved, the following dialog box is displayed:



Click *Yes* to save the state of the environment. Click *No* to close the session without saving the state or click *Cancel* to continue working in the current state.

Saving a Script

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

OCEAN lets you

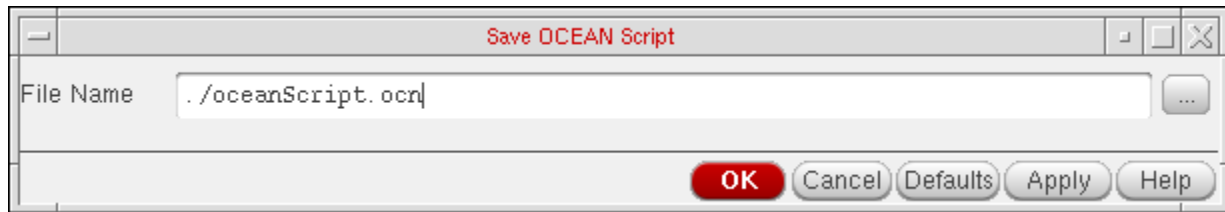
- Create scripts that you can run repeatedly to verify circuit performance
- Run longer analyses such as parametric analyses, corners analyses, and Monte Carlo simulation, more effectively
- Run long simulations in OCEAN without starting the analog circuit design environment graphical user interface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

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

To create a script from the analog circuit design environment,

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

The *Save Ocean Script* form appears.



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

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

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

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

For more information about OCEAN commands and scripts, see the [OCEAN Reference](#).

Running the Simulation Setup in Advanced Simulation Environments (ADE XL / ADE GXL)

Using the ADE XL and GXL environments, you can create simulation setup for multiple tests together and run advanced simulations. These environments also provide enhanced features and advanced debugging capabilities.

If you have created a test setup in ADE L, you can load the same in ADE XL or ADE GXL and run simulations. Earlier, to do this, you had to first save the setup as an ADE L state. Next, you had to start ADE XL or ADE GXL environments independently and then load the saved ADE L state.

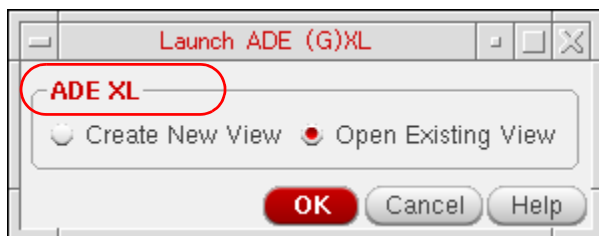
Virtuoso Analog Design Environment L User Guide

Environment Setup

Starting with IC 6.1.5, a new menu, *Launch*, has been introduced using which you can directly start ADE XL or ADE GXL environments with an adexl view. It will also load the current test from ADE L into the adexl view automatically. After this, you can configure advanced options for your test, run simulations and analyze results.

Opening ADE XL / ADE GXL using the Launch Menu

To open ADE XL from ADE L, choose *Launch – ADE XL*. Similarly, to open ADE GXL, choose *Launch – ADE GXL*. The *Launch ADE (G)XL* form appears.



Note: The same form *Launch ADE (G)XL* opens for both ADE XL and ADE GXL.

In the ADE XL and ADE GXL environments, the testbenches (tests) are saved in `adexl` views. You can use the *Launch ADE (G)XL* form to either create a new `adexl` view or to open an existing `adexl` view.

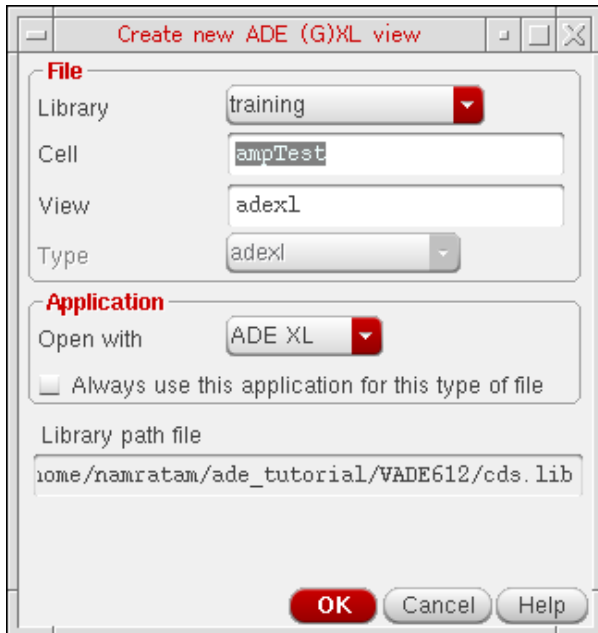
See the following sections for more details about creating and opening the ADE XL or ADE GXL views:

- [Creating a New ADE \(G\)XL View](#)
- [Opening an Existing ADE \(G\)XL View](#)

Note: You can customize the way in which ADE XL or ADE GXL open from ADE L and designs are loaded in them. For more details, refer to [Customizing the Opening of ADE XL or ADE GXL from ADE L](#).

Creating a New ADE (G)XL View

To create a new view, select the *Create New View* option in the *Launch ADE (G)XL* form. The *Create new ADE (G)XL view* form appears.



If no design cell is open in ADE L, the *Cell* field is blank. You need to specify a name of an existing cell here. However, if a design cell is open in ADE L, the name of the library and cell are displayed in the *Library* and *Cell* fields. You can change these values if you want to open another design cell. The default view name is `adexl`, however, you can give another name. Click *OK* to open the selected environment (ADE XL or ADE GXL). A new view, with the same name as specified in the *View* field, is created in this environment.

For more details on the *Create new ADE (G)XL view* form, refer to [Creating a New Setup in Virtuoso Analog Design Environment XL User Guide](#).

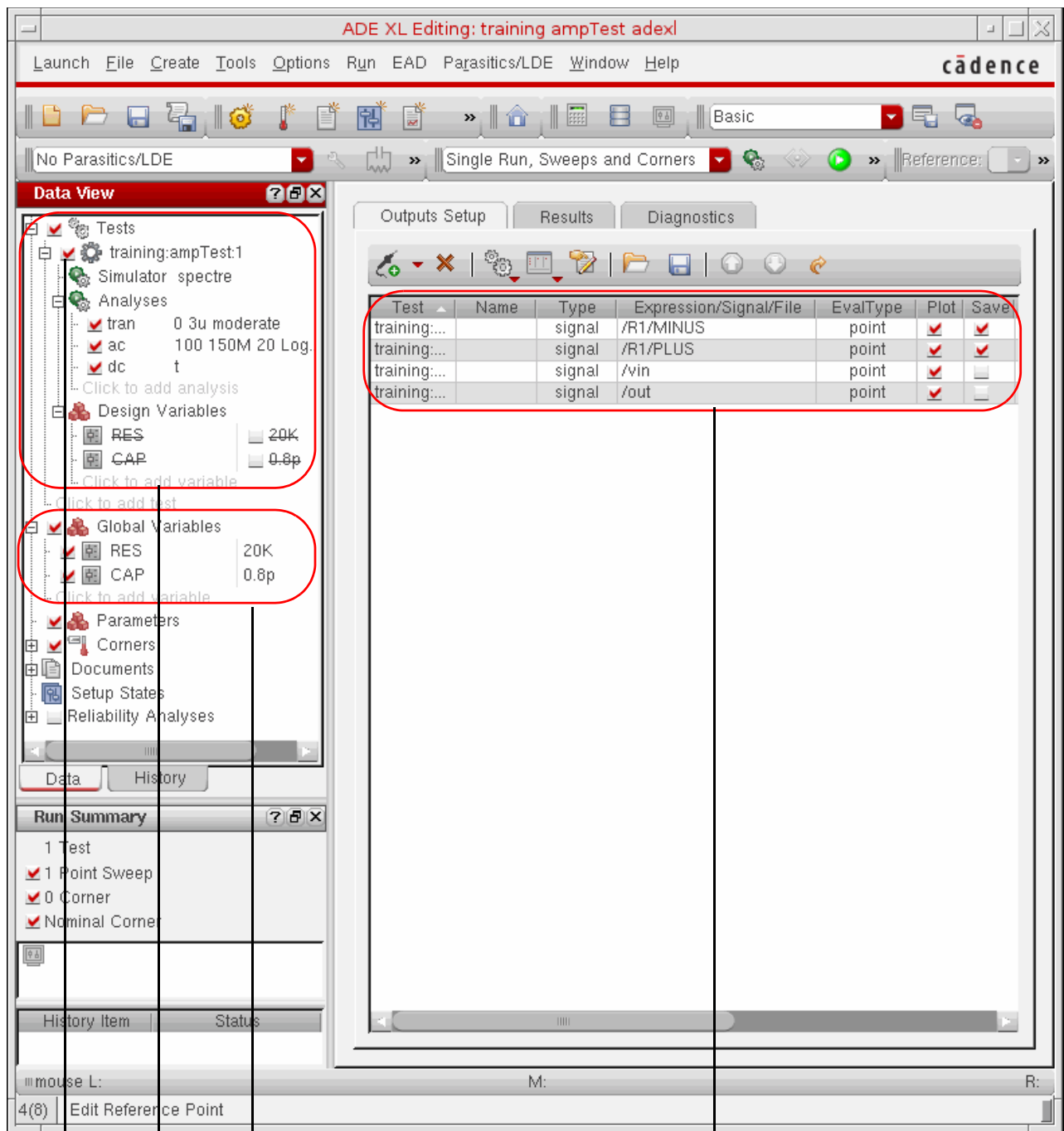
Depending on whether a design was open in ADE L, the contents of the new `adexl` view that you have created vary as described below.

When a Design Is Open in ADE L

If a design cell is open in ADE L and you specify the same design cell in the *Create New ADE (G)XL View* form, a new test is created for that cell in the `adex1` view. The simulation setup details from ADE L are imported to this new test.

Virtuoso Analog Design Environment L User Guide

Environment Setup



New test

Global variables

Imported
simulation setup

Output details
of the imported test

Virtuoso Analog Design Environment L User Guide

Environment Setup

Note that by default, in the `adexl` view, all the design variables that are imported to the new test are listed under two lists in the Data View pane in the ADE XL or ADE GXL environment:

- Design Variables
- Global Variables

By default, all the design variables are listed under `Global Variables`. Values of these variables are assigned to all the tests. Therefore, the variables under `Design Variables` appear struck off. If you want to use a design variable only for the test for which it was created, deselect the check box for that variable listed under `Global Variables`. The variable now applies only to the test for which it was created.

For more details about how to edit a test, refer to [Specifying Tests and Analyses](#) in the Virtuoso Analog Design Environment XL User Guide.

When No Design Is Open in ADE L

If no design is open in ADE L, the new `adexl` view created in the ADE XL or ADE GXL environment is blank. The following two forms also appear by default, using which you can create a new test:

- *Choosing Design*: Using this form, you can select a design cell for which you want to add a test in the `adexl` view. By default, the cell name that you specified in the *Create new ADE (G)XL view* form is selected in the *Cell Name* list. You can choose the same or another cell and view name for which you want to add a test in the `adexl` view. A new test is created and added to the *Tests* list in the *Data View* pane in the `adexl` view.
- *ADE XL Test Editor*: Using this form, you can specify simulation setup details for the new test in the *ADE XL Test Editor* window. These details are also displayed in the *Data View* pane in the `adexl` view.

For more details about how to edit a test, refer to [Specifying Tests and Analyses](#) in the Virtuoso Analog Design Environment XL User Guide.

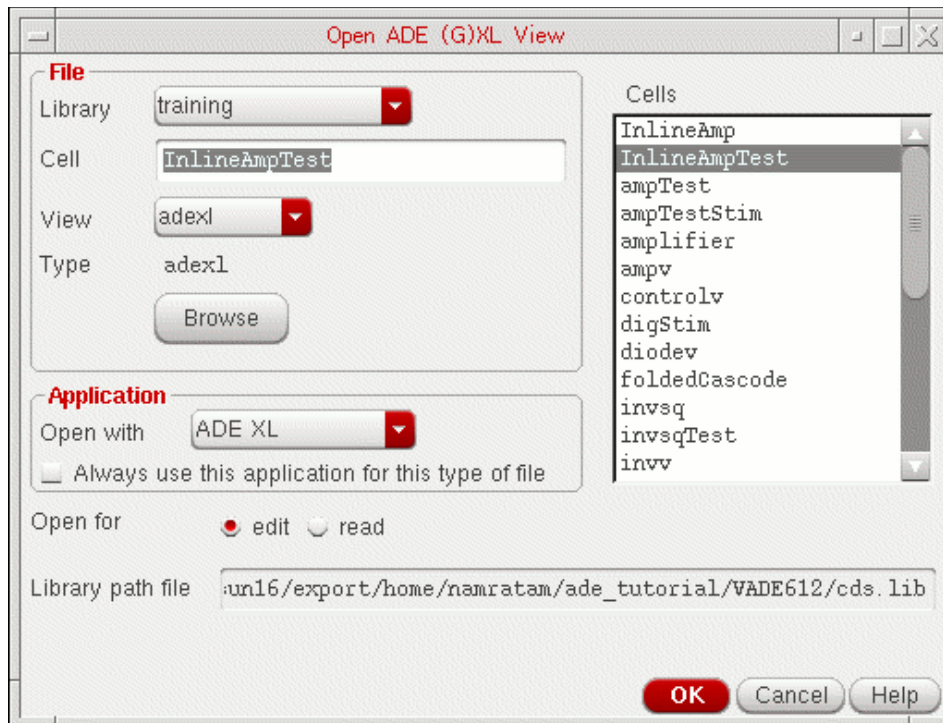
Opening an Existing ADE (G)XL View

If you open an existing ADE XL or ADE GXL view from ADE L, you can add more tests in it or run simulations, or do both.

Virtuoso Analog Design Environment L User Guide

Environment Setup

To open an existing ADE XL or ADE GXL view, in the *Launch ADE (G)XL* form, select *Open Existing View*. The *Open ADE (G)XL View* form appears.



If a design is open in ADE L, the form displays the library and cell name of that design. The *View* list contains all the existing views for that design cell. Use the *Library*, *Cell*, and *View* options to select the `adexl` view that you want to open and click *OK*.

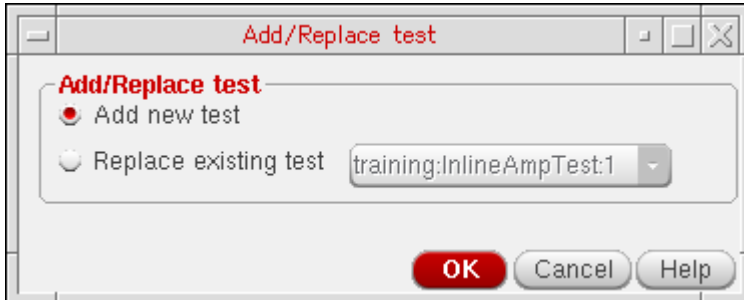
For more details on the *Open ADE (G)XL view* form, refer to [Opening an Existing Setup](#) in *Virtuoso Analog Design Environment XL User Guide*.

The selected environment (ADE XL or ADE GXL) opens. The `adexl` view that you selected also opens. All the tests in that view are listed in the *Tests* list.

Virtuoso Analog Design Environment L User Guide

Environment Setup

In addition, the *Add/Replace test* form also appears. Using this form, you can specify if you want to add a new test or replace an existing test in the view that you have opened.



Click *Cancel* if you want to only run simulations for the tests that exist in the `adexl` view. Otherwise, select appropriate options if you want to add or replace a test in this `adexl` view.

Adding a Test in an Existing ADE XL / ADE GXL View

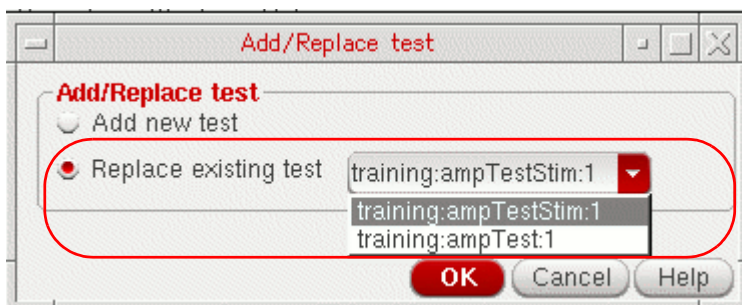
The *Add new test* option in the *Add/Replace test* form creates a new test in the `adexl` view that you have opened. The new test is added to the *Tests* list in the *Data View* pane.

If a design is open in ADE L, the simulation setup details are imported from ADE L as a new test in the `adexl` view. The design variables, if any, are also imported and appended to the list of global variables.

If no design is open in ADE L, you need to create a new test. For more details, refer to [Adding a new test when no design is open in ADE L](#).

Replacing a Test in an Existing ADE XL / ADE GXL View

The *Replace existing test* option in the *Add/Replace test* form replaces an existing test in the `adexl` view that you have opened. A list of tests that already exist in this view is displayed next to the *Replace existing test* option.



From this list, select the test that you want to replace and click *OK*.

If a design is open in ADE L, the simulation setup details for that design are imported to a new test in the adexl view. The test that you selected in the *Replace existing test* list is replaced with the new test. If no design is open in ADE L, you need to create a new test to replace the existing one. For more details, refer to [Adding a new test when no design is open in ADE L](#).

Customizing the Opening of ADE XL or ADE GXL from ADE L

You can customize how ADE L launches and loads designs in ADE XL or ADE GXL in the following ways:

- By default, the *Launch* menu appears in the ADE L environment. You can hide this menu by setting the `showMenu` environment variable to `nil`.
- By default, when you launch ADE XL or ADE GXL from ADE L, the *Launch ADE (G)XL* form appears using which you can choose to create a new adexl view or open an existing one. You can stop the *Launch ADE (G)XL* form from appearing by setting the `showOpenViewDialog` environment variable to `nil`. As a result, a new view is created by default every time you run the *Launch – ADE XL* command. In this case, you also need to specify name of the library in which you want the new view to be created. You do this by setting the `defaultLibName` environment variable. In this case, ensure that the specified library already exists and is writable. Otherwise, ADE L shows an error.
- By default, names of all the adexl views created when ADE XL or ADE GXL are launched from ADE L are prefixed with a default prefix, `adexl_imported_from_adel`. You can change this prefix by setting the `viewNamePrefix` environment variable, if required.

Configuring the Analog Design Environment

Configuration of the Virtuoso Analog Design Environment depends on several kinds of configuration settings:

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

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

Resetting the Default Environment

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

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

To reset the simulation environment to its default state,

- ▶ In the Simulation window, choose *Session – Reset*.

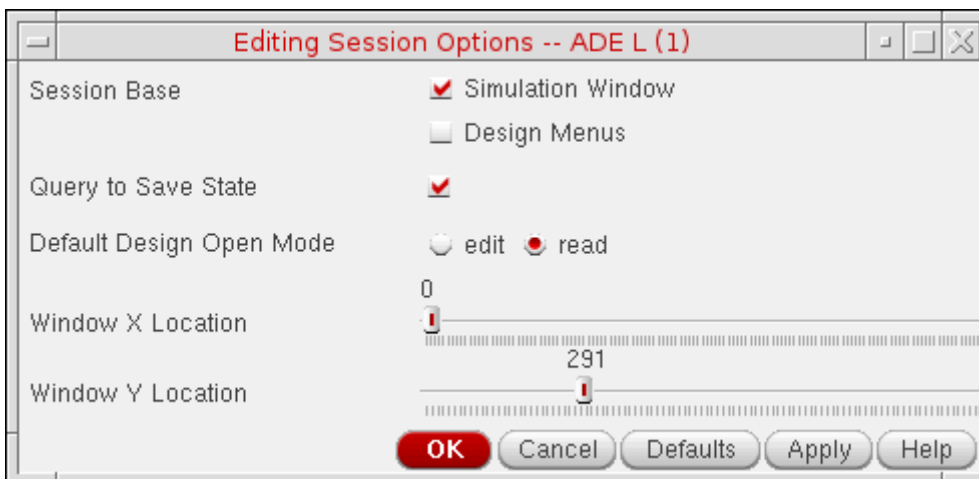
This command restores the original defaults. If you have data in this form, it is overwritten with default settings. Use the *Session – Save* command to save this information before resetting the defaults.

Setting Basic Session Defaults

To set the basic defaults, including your window preference,

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

The Editing Session Options form appears.



For detailed information about the form, see [“Editing Session Options”](#) on page 134.

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

2. Choose a session base.

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

3. To set the default mode whenever you open your Schematic window, choose *edit* or *read* in *Default Design Open Mode*.
4. To specify the default location for your Simulation window, use the *Window X Location* and *Window Y Location* fields to adjust the positioning.

Variables for Customizing Netlist Generation

You can customize ADE netlisting by setting the following variables.

Variable	Description
<code>nlReNetlistAll</code>	When set to <code>t</code> , all cellviews in the design are re-netlisted, irrespective of the setting specified using the ADE GUI options <i>Create</i> or <i>Recreate</i> in the <i>Simulation – Netlist</i> submenu. If <code>nlReNetlistAll</code> is not set and has its default value, <code>nil</code> , the ADE GUI netlisting options are used. You can set this variable in the <code>.simrc</code> file.
<code>globalGroundNet</code>	When you use this variable to specify a string, the value specified is considered to be the global ground net and overrides the default analogLib global ground net (<code>gnd!</code>). You can set this variable in <code>.simrc</code> file. This variable is valid for socket direct netlister only.
<code>hideGlobalGroundNetMessage</code>	When set to <code>t</code> , this variable suppresses the information messages issued after you change the default value of the <code>globalGroundNet</code> variable. The default value of <code>hideGlobalGroundNetMessage</code> variable is <code>nil</code> .

Virtuoso Analog Design Environment L User Guide

Environment Setup

<code>simPropValueNotation</code>	When set to <code>Engineering</code> , all property values are displayed in engineering notation in the netlist. For Example, a resistance value of <code>1m</code> is displayed as <code>1e-3</code> . If <code>simPropValueNotation</code> is set to <code>Scientific</code> , all property values are displayed in scientific notation in the netlist. For Example, a resistance value of <code>1m</code> is displayed as <code>1.000000000e-03</code> . You can set this variable in <code>si.env</code> file. This variable is valid for socket netlisters only.
<code>hnlDynamicFlagOn</code>	This variable can be set to <code>t</code> or <code>nil</code> in the <code>.cdsinit</code> file. If the variable is not specified, it defaults to <code>nil</code> . If set to <code>t</code> , the parameter values having length more than 8K, which is the maximum limit, are printed in the netlist. If set to <code>nil</code> , the parameter values with length more than 8K are truncated to the maximum limit.

Customizing Your `.cdsinit` File

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

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

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

Customizing Your `.cdsenv` File

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

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

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

Customizing Your Menus File

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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

```
else
  ;;; 44Y menus customization here
)
```

Alternatively, you can create the `simui_menus` file like this:

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

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

Setting UNIX Environment Variables

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

The `CDS_Netlisting_Mode` variable controls

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

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

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

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

Use these rules to set `CDS_Netlisting_Mode`.

- When you use the analog circuit design environment, set this variable to an appropriate value. If your design depends on CDF information, then set `CDS_Netlisting_Mode` to `Analog`. If your circuit does not use CDF information, then set `CDS_Netlisting_Mode` to `Digital` or `Compatibility`.
- When you use socket simulation in the analog circuit design environment and this variable is not set or is set to `Digital`, you see a dialog box that lets you set the value

Virtuoso Analog Design Environment L User Guide

Environment Setup

to `Analog` for the current session. (The socket netlister requires that the variable be set to `Analog`.) In this situation, the design environment uses the Analog LVS (`auLVS`) tool.

- If you use the analog circuit design environment for LVS, set `CDS_Netlisting_Mode` to `Analog`.
- If you use CDF and the Circuit Designer's Workbench but do not have the analog circuit design environment, set `CDS_Netlisting_Mode` to `Analog`.
- If you do not have the `auLVS` tool and do not use CDF or AEL expressions, set the variable to `Digital`. In this mode, you get the fastest netlisting speed and run `iLVS`, which uses OSS NLP expression syntax.
- If you want CDF compatibility with `iLVS`, set the variable to `Compatibility` for faster netlisting than with `Analog`. However, note that `Compatibility` mode has the following limitations:
 - ❑ Connection of terminals by properties is not supported. A typical use of this capability in the analog circuit design environment is connecting the bulk pin of four-terminal transistors to a net.
 - ❑ Analog circuit design environment features other than expression evaluation are not supported.

Note: By default, `CDS_Netlisting_Mode` variable is set to `Analog`.

Reserved Words

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

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

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

All Spectre simulator reserved words can be found in the *[Creating Component and Node Names](#)* section of the *Virtuoso Spectre Circuit Simulator Reference*.

Bindkeys

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

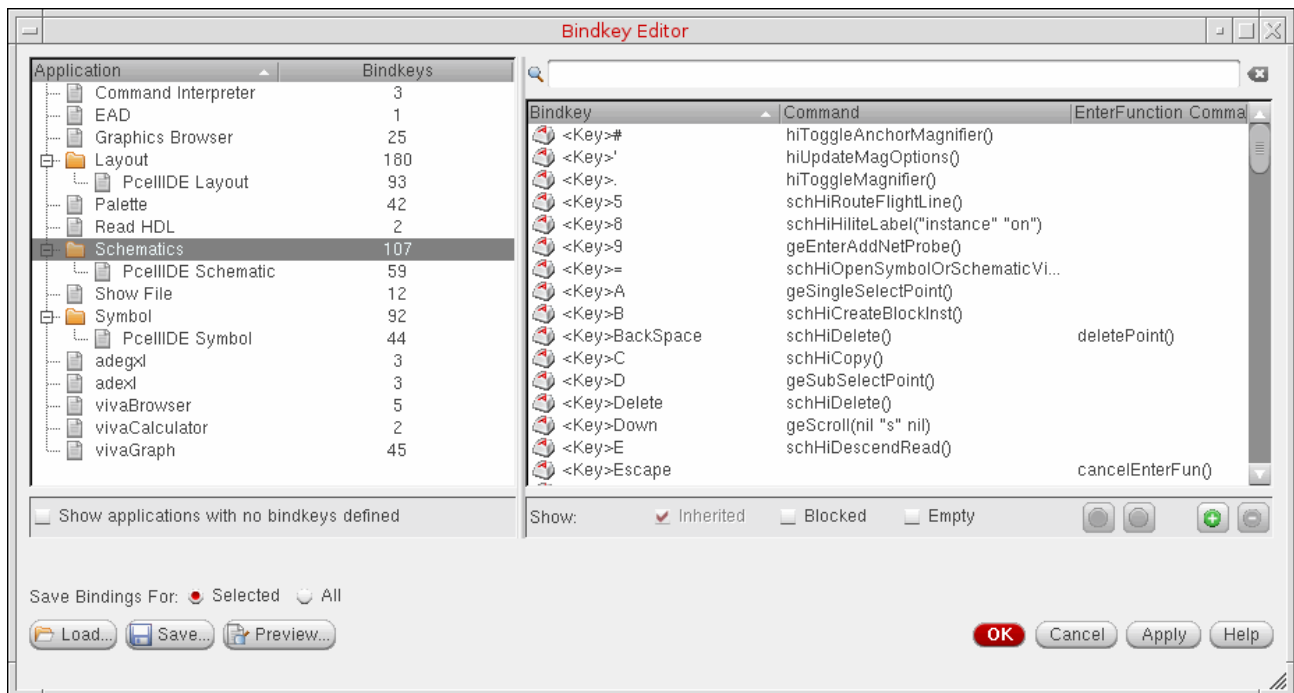
For more information about setting bindkeys, see the [Virtuoso Design Environment User Guide](#).

Checking Bindkey Assignments

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

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

The Bindkey Editor form appears.



2. Choose *Schematics* from the left pane of the Bindkey Editor form.

The right pane of the window displays the bindkeys defined for your system.

In this example, there are 107 bindkeys defined for Schematic: the key `Control-F` performs the same function as the *Options – Select Filter* command.

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

Assigning Bindkeys

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

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

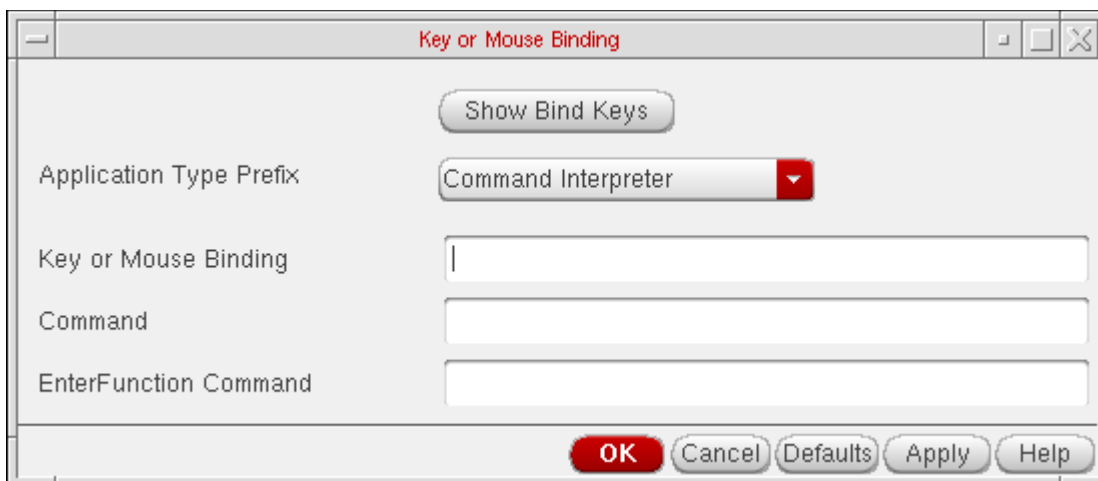
Bindkeys set by the first two methods are valid for the current session only. To set bindkeys you want for every session, you must enter them in your `.cdsinit` file.

Using the Key or Mouse Binding Form

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

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

The Key or Mouse Binding form appears.



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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

4. Click *OK* or *Apply*.

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

Using the CIW

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

- Type the following in the CIW:

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

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

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

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

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

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

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

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

Using Your .cdsinit File

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

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

```
vi .cdsinit
```

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

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

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

SKILL_command is the command you type to call the function.

3. To save your changes and quit, type

:wq

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

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

Form Field Descriptions

This section describes usage of the various fields available on the following forms:

- [Choosing Simulator/Directory/Host](#) on page 123
- [Create New File](#) on page 124
- [Setting Model Path](#) on page 124
- [Model Library Setup](#) on page 126
- [Environment Options](#) on page 128
- [Saving State](#) on page 130
- [Loading State](#) on page 132
- [Editing Session Options](#) on page 134

Choosing Simulator/Directory/Host

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

Project Directory lets you specify the run directory.

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

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

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

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

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

Create New File

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

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

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

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

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

Setting Model Path

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

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

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

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

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

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

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

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

Delete removes the selected directory from the directories list.

Virtuoso Analog Design Environment L User Guide

Environment Setup

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

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

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

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

Model Library Setup

Model File Lists the model libraries setup for the design.

- To add a model library, click where it says *Click here to add model file* and enter the path and file name of the model library file. Note the following when you are entering the path and file name:
 - You can use absolute and relative paths.
 - You can use environment variables to specify the path.
 - You can use design variables to specify the model library name. For more information on using design variables to specify model library names, see [Chapter 7, “Parameterization Support.”](#)

Alternatively, you can click the browse button to use the Select Model File form to select the model library files to be added.

Note: To add multiple model library files using the Select Model File form, press the *SHIFT* or *CTRL* key and click on the files you want to add.

Note: By default, when model files are added, the symbolic links in the file paths are resolved so that path to the actual file in the design cache is shown in the GUI. To prevent symbolic links from being resolved, set the `sevResolveSymLinks` variable in the `.cdsinit` file as:

```
envSetVal("asimenv" "sevResolveSymLinks" `boolean "nil")
```

- If the check box next to a model library is selected, the model library is enabled for the design. To disable the model library, clear the check box next to the library.

You can also add, enable or disable model library files using the `asiSetEnvOptionVal` SKILL functions as shown below:





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

The # symbol is used to disable the `model3.scs` file. If the # symbol is not used, the model library is enabled for the design.

Note: If you use an empty string, that is " ", as the first argument, the specified model file will not be included in the model library.

Virtuoso Analog Design Environment L User Guide

Environment Setup

- The order in which the libraries are listed in the form determines their search order. Libraries are searched starting at the top of the list. If a model is included in two or more libraries, you can change the search order to determine which library ADE L searches first. ADE L uses the first model found in the search order.
 - To move a library one level up, select the library and click the  button.
 - To move a library one level down, select the library and click the  button.
 - To edit a model library file, select the file and click the  button to open the file in a text editor
 - To delete a model library file, select the file and click the  button.

Virtuoso Analog Design Environment L User Guide

Environment Setup

Section Lets you select the section of the model library that you want the simulator to use.

The section you select determines which model definition the simulator uses. For example, in the following model file, if you select the section `TNTP`, only the model definition in the `TNTP` section is used by the simulator. If you select the section `FCS`, the model files specified in the `TNTP` section are used by the simulator. For more information on modeling, see the [Direct Simulation Modeling User Guide](#).

```
section TNTP
    model ...
    model ...
endsection
section FCS
    include "my_model.scs"
    include "fcs_model.scs"
endsection
```

Note: A model library file can have zero or more sections. If a model library file has no sections, no drop-down list is available in the *Section* field.

When the section for a model library file is specified, the netlist will contain the statement,

```
.LIB "<modelLibraryFile>" <section>
```

When a section is not specified, the netlist will contain the statement,

```
.INCLUDE "<modelLibraryFile>"
```

Environment Options

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

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

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

Print Comments

Prints the name mapping of the terminals in the netlist as comments.

When *Name Mapping* is selected, prints the mapping of the schematic terminal name with the netlist terminal name for the subcircuits and the mapping of schematic device names with simulator devices names.

When *Subckt Port Connections* is selected, prints the connection of each terminal with the net it is connected to, for each subcircuit.

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

Automatic output log

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

savestate (ss)

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

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

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

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

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

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

Run with 64 bit binary instructs spectre to run in 64 bit mode.

Using colon as Term Delimiter instructs spectre to use `:` as the terminal delimiter. Whenever you select `sst2` as the output format, you have to explicitly select this form option.

Netlist text views from shadow OA database While generating the netlist for a design that contains text cellViews,

If this option is selected, the netlister reads the database for each text cellview and prints the corresponding netlist.

If this option is not selected, the netlister includes the corresponding text files in the netlist.

Important Points to Note:

- This form option will have an impact only when you select sst2 as the output format.
- If **Using colon as Term Delimiter** option is selected/deselected, you need to netlist the design.

Set Top Circuit as Subcircuit instructs spectre to netlist the top-level schematic as a subcircuit.

Saving State

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

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

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

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

What to Save controls the type of information saved.

Select All saves all the types of information shown.

Clear All does not save any of the types of information shown.

Analyses saves the Choosing Analyses form settings (but not simulator options that you set with the options buttons in these forms). Analysis form settings are simulator specific, so you cannot restore them from a different simulator.

Variables saves design variables.

Outputs saves the saved and plotted output settings. Output settings are design specific, but if the signal names match, you can restore them from a different design. You can restore expressions from any design.

Subckt Inst saves the *Save By Subckt Instances* pane settings. This option is available only for the spectre simulator.

Model Setup saves the model setup of the session.

Simulation Files saves simulation-specific files specific files and other information. For spectre, these files can be stimulus files, definition files, and include paths.

Environment Options saves the environment option settings. If you are using a different simulator in another session, only those options that are applicable to the current simulator can be retrieved. The information you can get to by clicking *Verilog Netlist Option* in the Environment Options form is among the information that you can save.

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

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

Waveform Setup saves the graph window settings for the Virtuoso Visualization and Analysis XL.

Graphical Stimuli saves the setup of the graphical stimulus form.

Conditions Setup saves the settings for the Circuit Conditions form.

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

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

RelXpert Setup saves the settings of the Reliability Setup options form.

Cosimulation Options saves the Matlab/Simulink cosimulation options settings.

Performance/Parasitic Reduction saves the High Performance simulation options settings. This information is simulator specific.

MDL Control Setup saves the MDL control options, which runs Spectre with the MDL control file.

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

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

Solver Option saves the solver specification for the current session.

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

MATLAB and Simulink

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

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

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

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

Loading State

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

Directory Options lets you specify the *State Load Directory* from which you want to load the state. You may either type it in or specify it by using the *Browse* button. You then specify the *Library*, *Cell*, and *Simulator* that the state used. Use the *State Name* field to select the state that you want to load. You may use the *Delete State* button to delete the existing states.



Now, the contents of the state files are saved in the alphabetical order. As a result, the state saved in previous release for the same setup will be different as compared to the state saved in the current release.

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

Description displays the description about the state being loaded.

What to Load controls the type of information restored.

Select All restores all the types of information shown.

Clear All does not restore any of the types of information shown.

Analyses restores the *Choosing Analyses* form settings (but not simulator options that you set with the Options buttons in these forms). Analysis settings are simulator specific, so you cannot restore them from a different simulator.

Variables restores design variables.

Outputs restores the saved and plotted output settings. Output settings are design specific, but if the signal names match, you can restore them from a different design. You can also restore expressions from any design.

Subckt Inst restores the *Save By Subckt Instance* pane settings. This option is available only for the spectre simulator.

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

Simulation Files restores simulation-specific files and other information. For spectre, these files can be stimulus files, definition files, and include paths.

Environment Options restores only the environment option settings. If you are using a different simulator in another session, you can restore only those options that are applicable to the current simulator.

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

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

Waveform Setup restores the graph window settings.

Graphical Stimuli restores the setup of the graphical stimulus form.

Conditions Setup restores the settings for the Circuit Conditions form.

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

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

RelXpert Setup restores the setup of the Reliability Setup options form.

Cosimulation Options restores the Matlab/Simulink cosimulation options settings.

Performance/Parasitic Reduction restores the High Performance simulation options settings. This information is simulator specific.

MDL Control Setup restores the MDL control options, which runs Spectre with the MDL control file.

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

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

Solver Option restores the solver specification for the current session.

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

MATLAB and Simulink

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

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

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

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

Editing Session Options

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

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

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

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

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

Virtuoso Analog Design Environment L User Guide

Environment Setup

Design Variables and Simulation Files for Direct Simulation

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

- [Using Direct Simulation](#) on page 137
- [Design Variables and Simulation](#) on page 138
- [Using a Definitions File](#) on page 144
- [Stimuli Setup](#) on page 145
- [Model Files in the Virtuoso Analog Design Environment](#) on page 152
- [Model File Libraries](#) on page 152
- [Referencing Textual Subcircuits or Models](#) on page 153
- [Scope of Parameters](#) on page 155
- [How the Netlister Expands Hierarchy](#) on page 159
- [Modifying View Lists and Stop Lists](#) on page 161
- [About Netlists](#) on page 163
- [Form Field Descriptions](#) on page 165

Using Direct Simulation

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

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

Design Variables and Simulation

The following section describes how to work with design variables.

Setting Values

You can use design variables and [CDF parameters](#) to set component values.

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

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

Name	Value
TRAN_STOP	1u
CAP	800f
R1	4K
R2	500
RA	100

Note: Do not use simulator reserved words as design variable names. For more information, see [Reserved Words](#) on page 119.

If you use the reserved words for design variables, the following message is displayed:



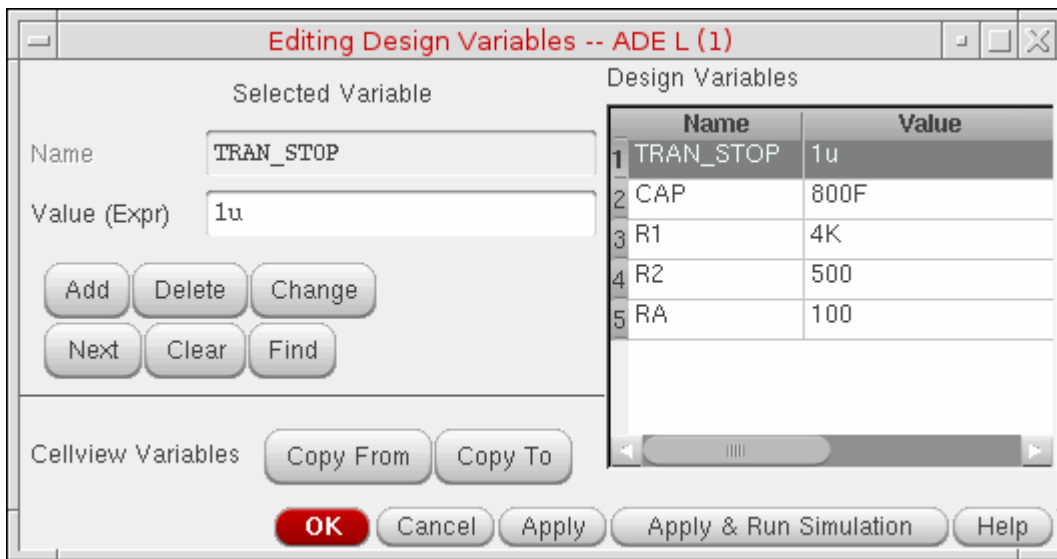
If you want to create some design variables for every ADE session, you can specify the name and value of those variables using the [designVarSetting](#) environment variable.

Adding a New Variable

To add a new variable,

1. In the Simulation window, choose *Variables – Edit*.

The Editing Design Variables form appears.



For detailed information about the form, see [“Editing Design Variables”](#) on page 167.

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

The name must begin with a letter and can contain only letters and numbers.

4. Enter a number or an expression in *Value (Expr)*.

The expression can be an equation, a function, or another variable. Expression syntax follows [AEL](#) syntax. These expressions are evaluated by the simulator. If the value does not have any number before the decimal point, the AMS Designer simulator inserts a leading zero before the decimal point.

Note: In the GUI, you can now add a variable without specifying any value for it. The value can be assigned to the variable in a file, which is loaded using the definitions file at the time

of simulation. As the value of the variable is not specified in the GUI, the simulator flags a warning message as shown below:



Click *OK* to continue the simulation process. Click *Cancel* to set the value of the design variable.

1. Click *Add*.

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

You can drag and drop the design variables up/down within the Design Variables section.

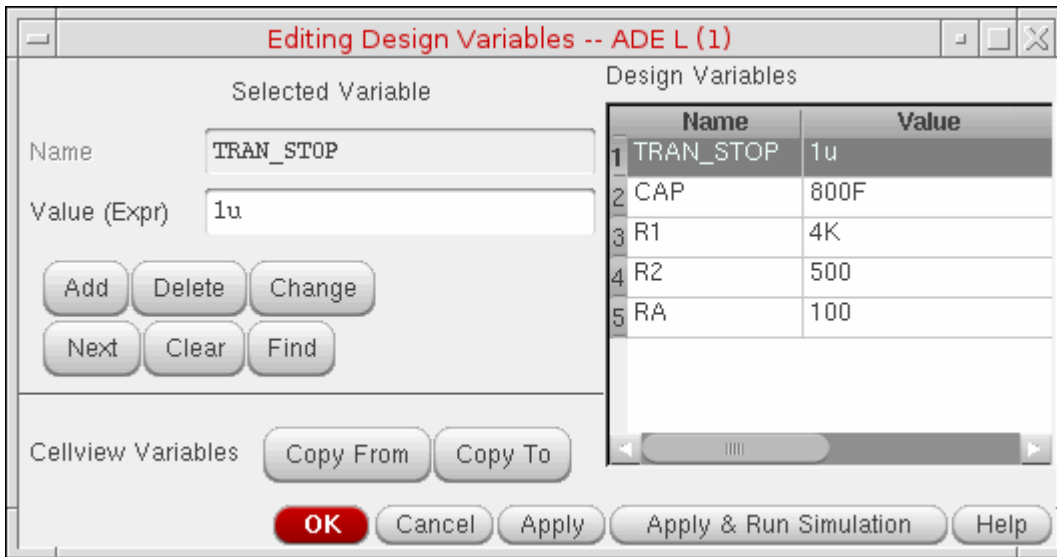
Note: Whenever you define a new variable, its added to the *Parametric Analysis* window's variable combo box, if it is open, automatically. For more information on *Parametric Analysis*, refer to [Parametric Analysis](#) on page 451. Additionally, you can also define variables in the [definitions](#) file.

Changing Values

To change the value of a design variables,

1. In the Simulation window, choose *Variables – Edit*.

The Editing Design Variables form appears.



For detailed information about the form, see [“Editing Design Variables”](#) on page 167.

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

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

3. Edit the value or expression.
4. Click *Change*.
5. Click *Apply* or *Apply & Run Simulation*.

Note: Whenever you change the value of a variable, the change will be reflected on the *Parametric Analysis* window automatically for the new rows. For more information on *Parametric Analysis*, refer to [Parametric Analysis](#) on page 451.

Deleting Values

To delete a design variable,

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

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

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

2. Choose *Variables – Delete*.

Note: Whenever you delete a variable, it is automatically deleted from the *Parametric Analysis* window's variable combo box, if it is open. The existing rows will not be affected. For more information on *Parametric Analysis*, refer to [Parametric Analysis](#) on page 451.

Saving Variable Values

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

To save the variable values,

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

The [Saving State form](#) appears.

2. Type a name for the saved simulation state.
3. Check that the *Variables* box is selected, and click *OK*.

Restoring Variable Values

To restore saved variable values,

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

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

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

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

3. Click an entry in *State Name*.
4. Choose the *What to Load* options you need and click *OK*.

Copying Values between the Schematic and the Simulation Environment

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

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

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

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

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

Displaying Values on the Schematic

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

1. Edit the component's CDF with the [CDF Editor](#).
2. Set *paramEvaluate* to *full*.

Adding Setup Files for Direct Simulation

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

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

These files are specified through the Simulation Files Setup form.

- The [model files](#), where you define models referenced by your design

These files are specified through the Model Library Setup form.

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

These files are specified through the Simulation Files Setup form.

The syntax used is determined by the simulator.

In the stimulus file, you can type node names or component names in Open Simulation System (OSS) syntax `[#name]` to have the system substitute the corresponding node numbers in the netlist. You can use a backslash (\) to escape a square bracket. For more information about OSS syntax, refer to the [Open Simulation System Reference](#), or view a

sample stimulus file at `your_install_dir/tools/dfII/samples/artist/models/spectre/opampStimuli.scs`.

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

Using a Definitions File

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

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

To specify the definitions file,

1. Create the file in the directory you specify in the include path field on the Model Setup form.
2. Choose *Setup – Simulation Files*.
The Simulation files form opens.
3. Enter the full UNIX path of the definitions file, including any extension, in the *Definitions Files* field.

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

Syntax

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

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

A definition file might contain

■ Simple passing parameters

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


■ Functions returning constant values

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

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

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

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

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

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

Definition File Example

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

```
simulator lang=spectre  
real PiRho() {  
    return 2500;  
}  
real PbRho() {  
    return 200  
}  
real Rpb(real l, realw) {  
    return PbRho()*l/w;  
}  
real Rpi(l,w) {  
    return PiRho()*l/w;  
}
```

Stimuli Setup

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

- Add source symbols to the schematic
- Use the Setup Analog Stimuli form

- Specify a stimulus file

Using the Setup Analog Stimuli Form

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

For input stimuli, your top-level schematic must contain input pins for the signals that you plan to set. To use the power stimuli, you must use a global name on a signal (such as vdd!).

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

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

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

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

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

The Library Manager: Directory form appears.

- b. Choose the library name of the current design.

- c. Choose *Edit – Properties*.

The Library Property Editor form appears.

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

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

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

Virtuoso Analog Design Environment L User Guide

Design Variables and Simulation Files for Direct Simulation

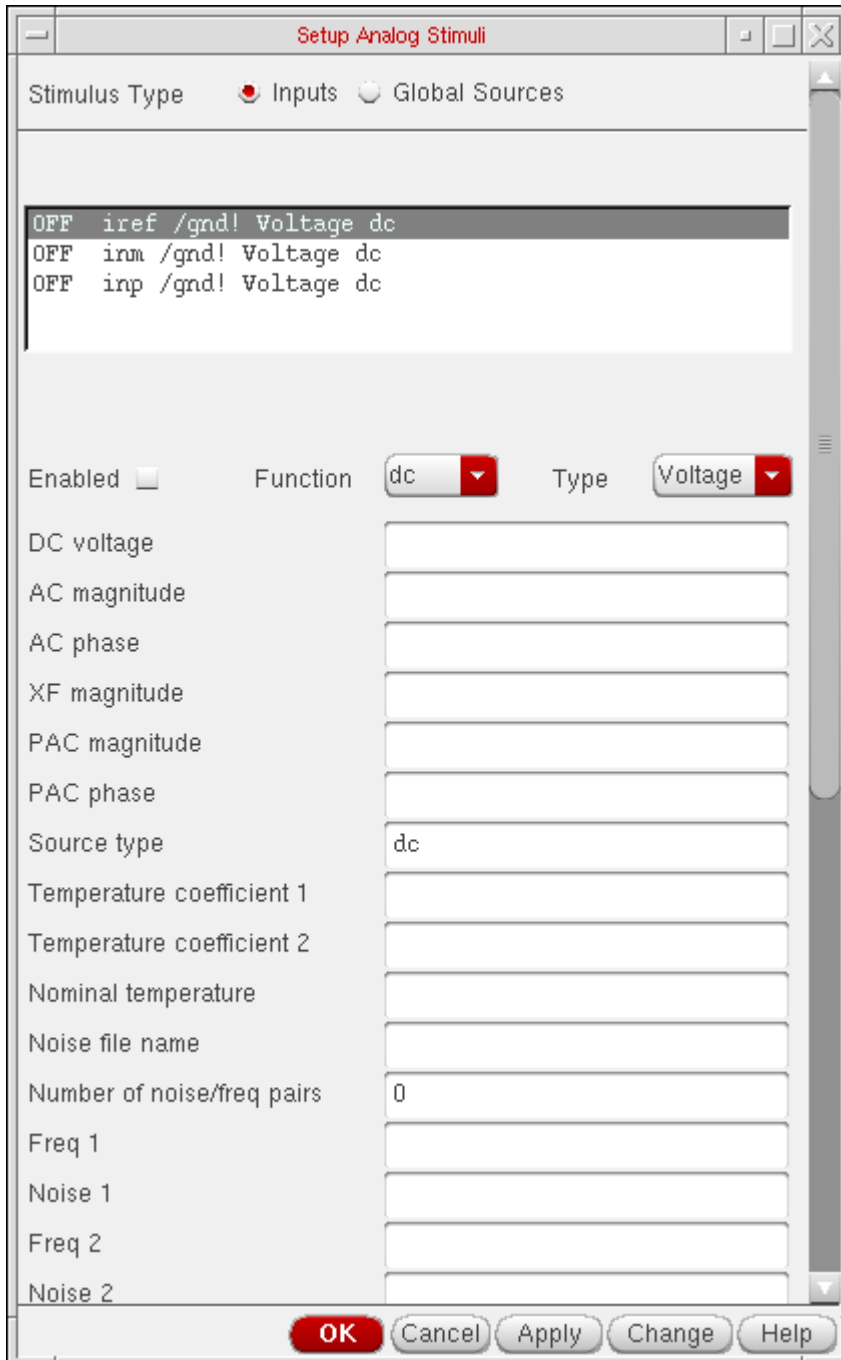
In the Add Property form, specify the following property name and characteristics.

Name	<i>refLibs</i>
Type	<i>string</i>
Value	list of one or more libraries in search sequence

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

- f. Click *OK* to return to the Library Manager form.
2. Choose *Setup – Stimuli* in the Simulation window to add stimuli.

The Setup Analog Stimuli form appears.



For detailed information about the form, see [“Setup Analog Stimuli Form”](#) on page 165.

3. Select the stimulus for an input signal:
 - a. Click an input pin in the list box.

b. Choose the appropriate *Function*.

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

c. To specify a voltage or current stimulus, select the appropriate value in the *Type* field.

d. Enter new parameter values as needed in the fields below the *Function* and *Type* fields.

Note: You can also set the parameter value in the Setup Analog Stimuli form using a VAR("myValue") variable expression syntax. The variable myValue, will then appear in the Name column of the Design Variables section of the ADE L form. You can provide the value for the variable in the Value column of the Design Variables section. If you provide an expression, the value of the expression is calculated when you create the netlist.

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

e. Click *Change*.

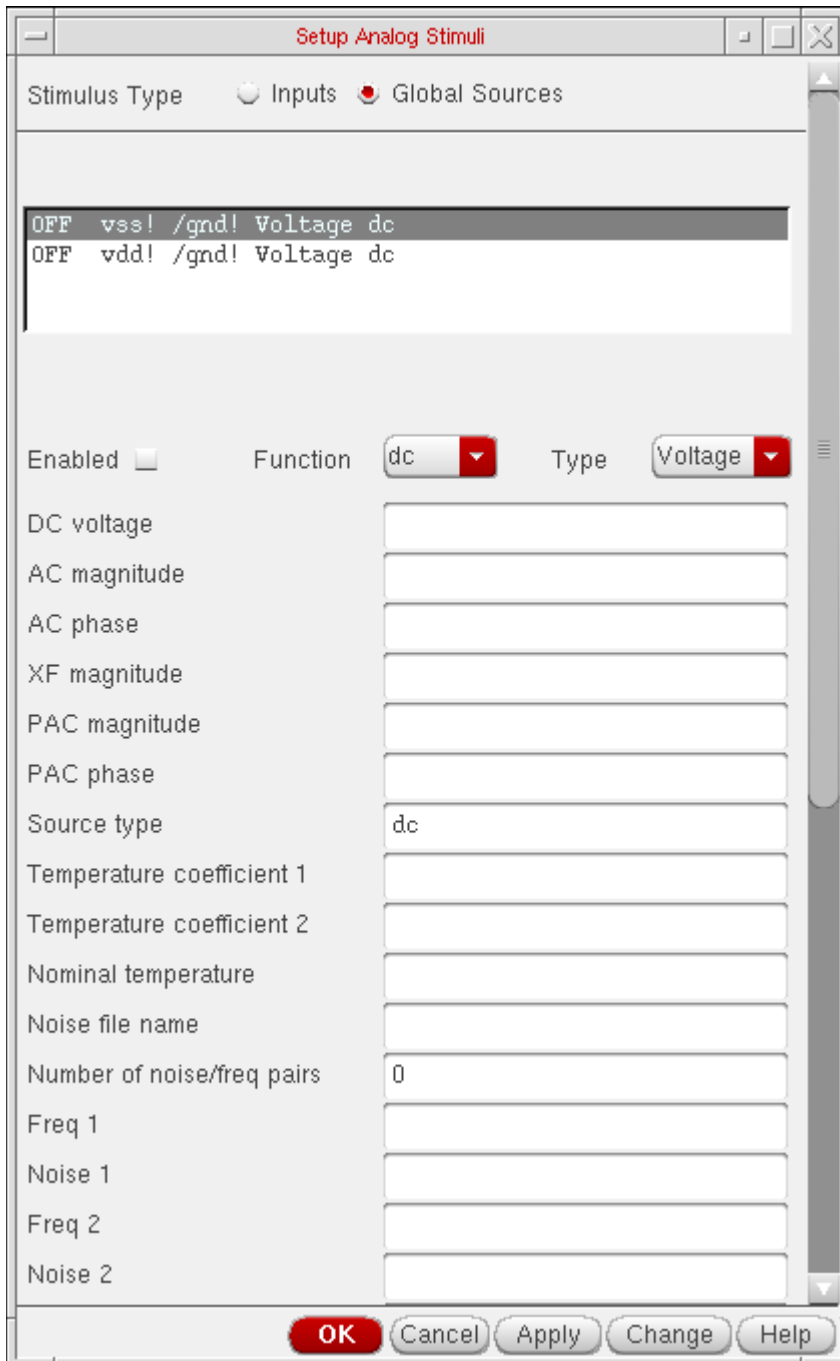
The list box displays the signal and the proper stimulus syntax for your simulator.

f. Click another input pin, and repeat these steps for each pin you want to edit.

g. Click *OK*.

4. Assign DC voltages to global sources:

a. Select *Global Sources*.



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

Only DC source values should be set here.

- b.** Click a source in the list box.
- c.** Select *dc* for *Function*.
- d.** Enter new values as needed in the parameter fields.

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

- e.** Click *Change*.

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

- f.** Click another source, and repeat these steps for each source you want to set.
- g.** To remove the voltage source for a particular global signal, select the signal in the list box and click *Enabled* to toggle it off.

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

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

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

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

Specifying a Stimulus File

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

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

Example of a spectre Stimulus File

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

```
simulator lang=spectre

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

Model Files in the Virtuoso Analog Design Environment

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

You can include one or more model files using the [Model Library Setup](#) form.

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

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

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

Updating cell CDF for Direct Simulation

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

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

Field	Value
netlistProcedure	<code>nil</code>
instParameters	The names of any CDF parameters on the component that you need to pass into the subcircuit or model
otherParameters	<code>model</code>
termOrder	The names of the symbol's terminals, in the order you want them netlisted (the order must match the node order on the <i>subckt</i> line or that of the model referenced)

Model File Libraries

This capability is also known as `.include` or `.lib` Commands. This is handled through the [Model Library Setup](#) form.

Referencing Textual Subcircuits or Models

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

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

Updating the Component CDF

To update the component CDF of the cell,

1. Choose *Tools – CDF – Edit* from the CIW.
2. Select the *Base* button in *CDF Layer* group box.
3. Select *Library Name* and *Cell Name*.
4. Set the parameter name and attributes in the *Component Parameter* tab, as shown below:

Field	Value
<i>name</i>	<i>model</i>
<i>type</i>	<i>string</i>
<i>prompt</i>	<i>Model name</i>
<i>parseAsNumber</i>	<i>no</i>
<i>parseAsCEL</i>	<i>no</i>
<i>defValue</i>	

Note: Do not use the *Units* attribute.

5. Fill out the fields on the *Simulator Information* tab according to the following table:

Field	Value
<i>netlistProcedure</i>	nil
<i>instParameters</i>	The names of any CDF parameters on the component that you need to pass into the subcircuit

Virtuoso Analog Design Environment L User Guide

Design Variables and Simulation Files for Direct Simulation

Field	Value
<i>otherParameters</i>	model
<i>termOrder</i>	The order of the terminal names required for the subcircuit definition
<i>componentName</i>	The name of component

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

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

Creating a Stopping Cellview

To create a stopping cellview for your simulator,

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

Using the Component

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

Including the Subcircuit File in the Netlist

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

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

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

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

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

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

Scope of Parameters

You can use [design variables](#) and CDF parameters to set component values.

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

Inheriting from the Same Instance: `iPar()`

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

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

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

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

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

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

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

Passed Parameter Value of One Level Higher: pPar()

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

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

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

For example:

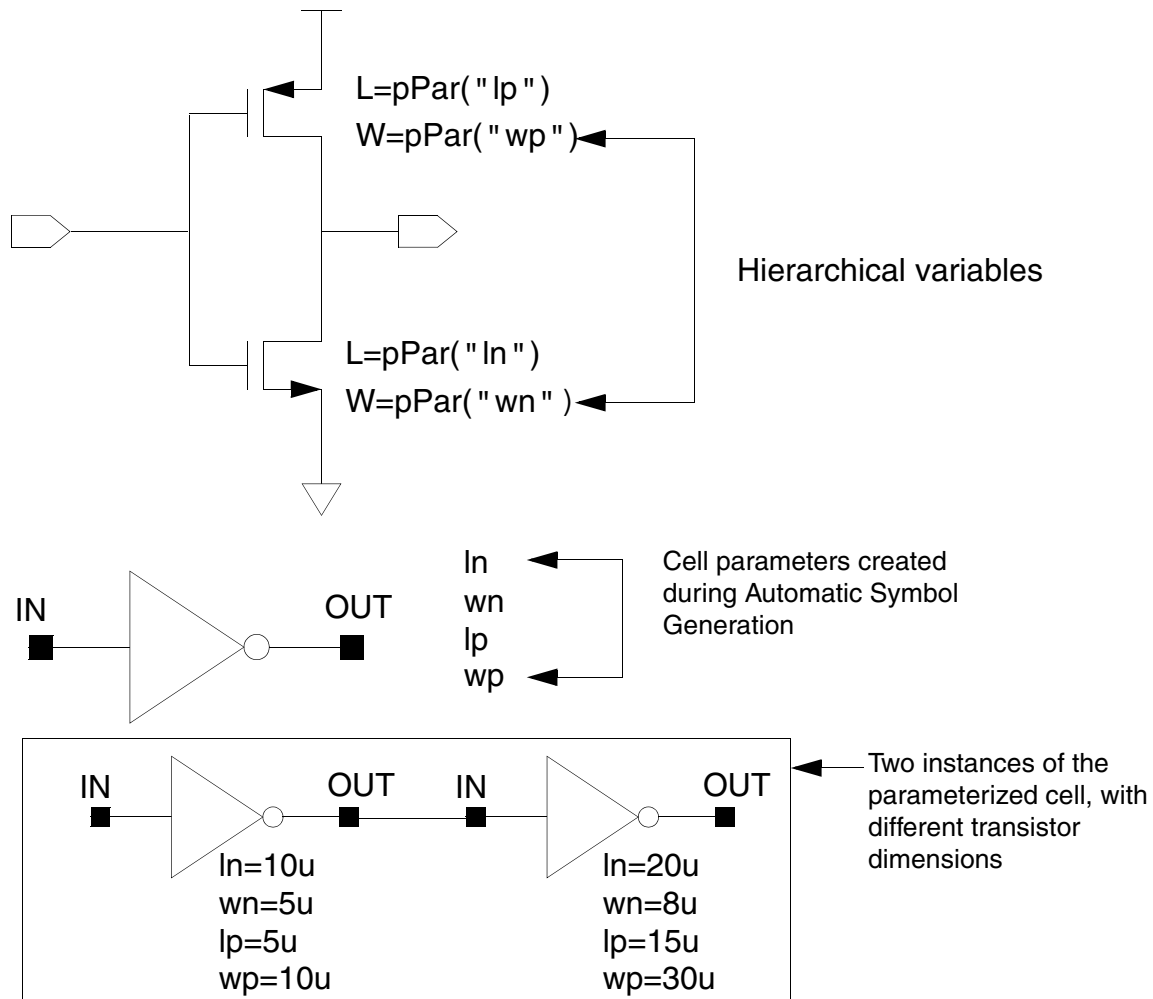
```
pPar("vss")
```

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

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

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

Using Ppar



Passed Parameters from Any Higher Level: atPar()

You should avoid using atPar. Use pPar instead.

Inheriting from the Instance Being Netlisted: dotPar()

You should avoid using dotPar. Use iPar instead.

Table of Functions

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

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

Nesting Functions

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

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

Consider the expression

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

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

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

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

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

Using Inheritance Functions in Input Files

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

How the Netlister Expands Hierarchy

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

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

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

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

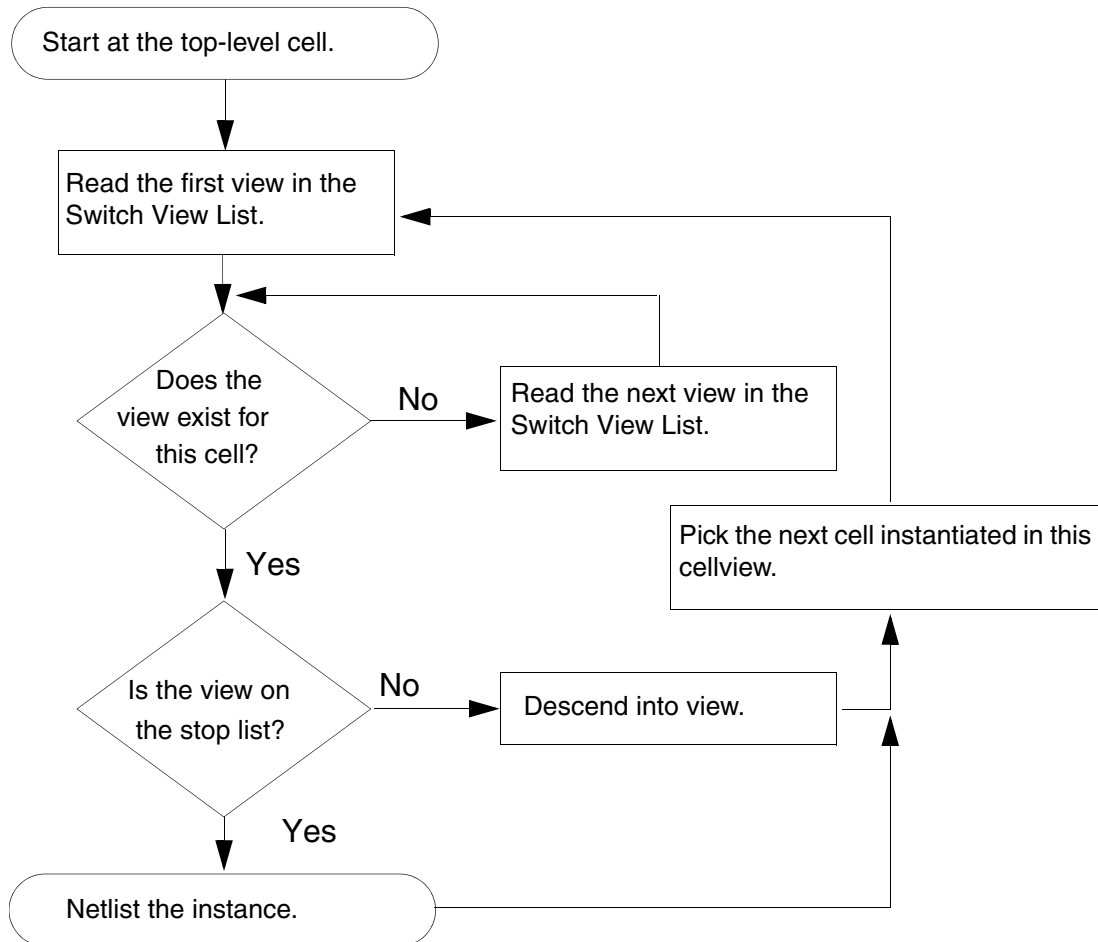
For more information, see the [Cadence Hierarchy Editor User Guide](#).

Note: Parasitic simulation and mixed-signal simulation use different processes for creating view lists and stop lists. Refer to Chapter 1 of the [Virtuoso Parasitic Estimation and Analysis User Guide](#) and Chapter 7 of the [Virtuoso Mixed-Signal Circuit Design Environment User Guide \(IC6.1.6 only\)](#) for details.

Virtuoso Analog Design Environment L User Guide

Design Variables and Simulation Files for Direct Simulation

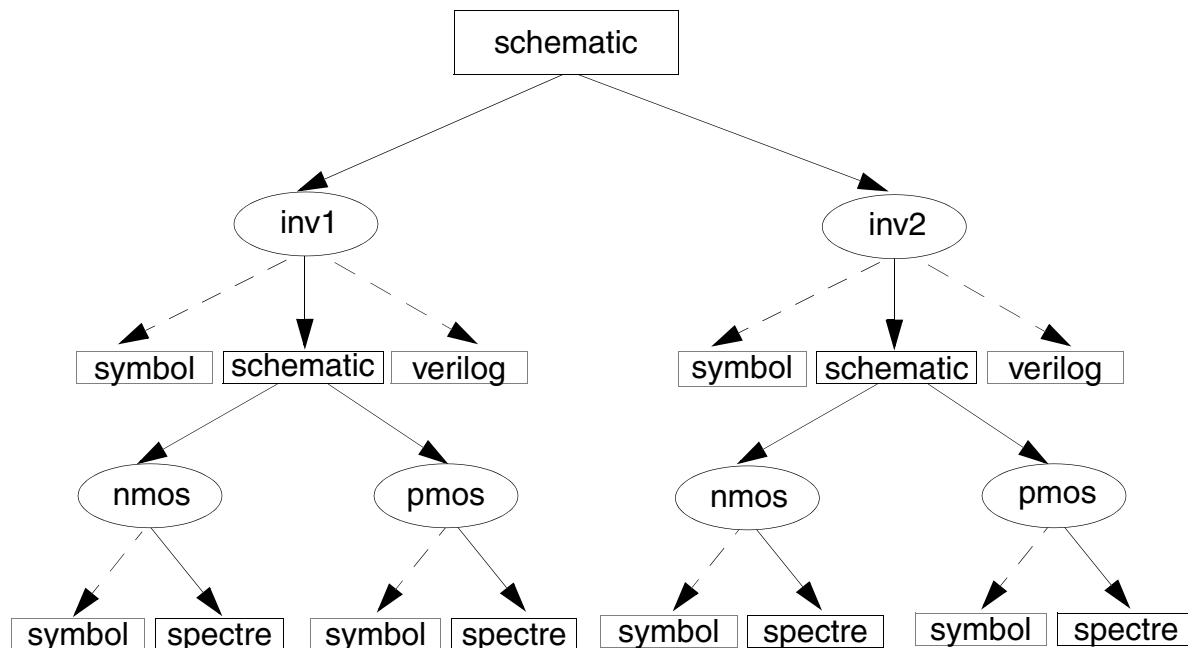
The flowchart shows how a netlister like OSS expands a design.



Netlisting Sample for Spectre

The following figure illustrates how hierarchy expansion is performed on a simple design. The solid lines show the view selection and design expansion based on the Switch View List and Stop View List that are provided.

Switch View List: spectre schematic cmos.sch verilog
Stop View List: spectre verilog



Modifying View Lists and Stop Lists

To modify a switch view list or stop view list,

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

The *Environment Options* form appears.

The screenshot shows the "Environment Options" dialog box. The "Switch View List" field contains the text "spectre cmos_sch cmos.sch schematic veriloga". The "Stop View List" field contains "spectre". The "Print Comments" section has "Name Mapping" checked and "Subckt Port Connections" unchecked. The "Automatic output log" checkbox is checked. The "savestate(ss):" and "recover(rec):" sections each have "Y" and "N" checkboxes, both of which are currently unchecked. The "Run with 64 bit binary", "Using colon as Term Delimiter", and "Set Top Circuit as Subcircuit" checkboxes are also unchecked. The "OK" button is highlighted in red.

For detailed information about the form, see “[Environment Options](#)” on page 128.

2. Modify entries to the *Switch View List* field, as needed.

The switch view list informs the netlister how to descend into different views in the design. Sequence is important in the switch view list. Refer to the [flowchart](#) to see how the netlister selects appropriate views.

Note: If the design contains a `pspice` view, it is automatically instantiated in ADE L if the ADE L session is attached to a config view. If the ADE L session is opened from the schematic and is not attached to any config view, add the `pspice` view in the *Switch View List* in the *Environment Options* form.

Important

The `pspice` view is currently supported only on the `lnx86` platform

3. Modify entries to the *Stop View List* field, as needed.

In most cases, you can control netlisting adequately using the switch view list. If necessary, you can add entries to the end of the stop view list. Sequence is not important in the stop view list.

4. Click *OK* or *Apply* to add your changes.

About Netlists

The analog circuit design environment creates or updates the simulator input file automatically when you give the command to run a simulation.

You do not need to use the *Netlist – Create* command unless

- You want to use analog circuit design environment to create a netlist but run the simulator in standalone mode
- You want to modify the netlist, perhaps to take advantage of features that the design environment interface to your simulator does not support
- You want to read the netlist before starting the simulation

There are two kinds of files generated for simulation:

- The netlist, which contains component information but no simulation control data
- The simulator input file, which contains both the netlist and the simulator control information required (this file is passed to your simulator)

Netlists are hierarchical. They are created incrementally, re-netlist only the changed schematics in a design. All schematics can be forced to re-netlist by choosing *Simulation – Netlist – Recreate* from the Simulation window.

The .simrc File

You can use the `.simrc` file with Spectre during interface initialization to customize netlisting. This file helps you set or override defaults for simulation variables in such a way that the changes affect only your own simulations. The `.simrc` file is a way to set defaults on a per-user or per-system basis, and no other designer is affected by this file. This file is optional and is loaded if it exists. It is searched for in the following order:

```
$SIMRC/.simrc  
$ossSimUserSiDir/.simrc  
dfII/local/.simrc
```

```
Current UNIX directory/.simrc
~/simrc
```

The search stops at the first place where the file is found. No other `.simrc` files are loaded unless the load is done from within the first one it finds, allowing for tiered loading or for the local CAD group to alter or disallow the search mechanism. If it does not exist, no error is generated. You can delete the `.simrc` file if you do not want it to be taken into account while netlisting.

Incremental Netlisting

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

Creating and Displaying a Netlist

To create and display a new netlist,

- Choose *Simulation – Netlist – Create*.

The simulator input file is created in a file. The name of the file is `input` with the simulator-specific extension. For Spectre, the extension is `.scs`. This file is put in the following directory:

```
projectDirectory/topCellName/simulatorName/view/netlist/
input.ext
```

`projectDirectory` is the directory you specified in the [Choosing Simulator/Directory/Host form](#).

`topCellName` is the root level cell name of the design.

`simulatorName` is the name of the simulator.

`view` is the view name being netlisted.

`ext` is the simulator-specific extension.

To display an existing simulator input file,

- Choose *Simulation – Netlist – Display*.

Note: For information on variables that you can set to customize netlisting options, see the [“Variables for Customizing Netlist Generation”](#) on page 116.

Form Field Descriptions

Setup Analog Stimuli Form

Stimulus Type

Inputs sets the stimulus for the signals with input pins in the schematic.

Global Sources lets you assign DC voltages to global signals that represent power supplies in the design.

Name identifies the signal name that is currently selected.

Library identifies the library where the selected signal or global source model was found.

Change recalculates the input voltage or current for the selected signal based on the function, type, and property values specified in the lower portions of the form. The calculated value is specified in the list box in the appropriate syntax.

Enabled lets you specify whether each signal is ON or OFF.

Function lets you choose the function for the selected signal.

dc displays the direct current stimulus option properties and values.

pulse displays the pulse stimulus option properties and values.

sin displays the sinusoidal stimulus option properties and values.

exp displays the exponential stimulus option properties and values.

pwl displays the piecewise linear stimulus option properties and values.

pwlf displays the name of the file containing piecewise linear stimulus option properties and values.

sffm displays the single frequency FM stimulus option properties and values.

Type lets you select the voltage or current for the signal highlighted in the list box.

Parameters and their values identify the simulator-specific parameters required by your simulator. The parameters list here will vary depending on the simulator you are using. Refer

Virtuoso Analog Design Environment L User Guide

Design Variables and Simulation Files for Direct Simulation

to your simulator documentation for information on setting or changing these parameter values.

The form lets you set inputs and global sources for your design.

The list box contains the current netlist values of the input or bidirectional pins. Each line contains the proper syntax for your simulator.

The fields displayed below the *Function* and *Type* cyclic fields (*AC magnitude*, *AC phase*, *DC voltage*, and so forth in this example) provide parameter input specific to your simulator.

When the form is first displayed, fields in this section could be blank, could contain default values, or could contain initial values that you specified at another time.

The form changes dynamically when you select a different input pin, function, or type.

Editing Design Variables

Name is an optional name for the variable, which appears in the *Table of Design Variables* list box.

Value (Expr) is the variable value, either a number or an expression.

Add creates the variable you have specified in the *Selected Variable* area.

Delete removes a highlighted variable. Click in the list box to highlight a variable.

Change updates the highlighted variable with the new information from the *Selected Variable* area.

Next highlights the following signal or expression in the *Table of Design Variables* list box.

Clear empties the *Selected Variable* area so you can enter a new variable.

Find locates the highlighted variable in your design.

Cellview Variables lets you keep variables consistent in the simulation environment and the cellview design database by copying them back and forth.

Copy From copies the variable values in the schematic cellview into the simulation environment.

Copy To copies the variable values in the simulation environment to the schematic cellview.

Table of Design Variables identifies the name and value of each design variable in the design. Each entry is numbered for easy reference.

Virtuoso Analog Design Environment L User Guide

Design Variables and Simulation Files for Direct Simulation

Setting Up for an Analysis

This chapter shows you how to set up to run an analysis.

- [Required Symbol](#) on page 169
- [Setting Up with Different Simulators](#) on page 170
- [Deleting an Analysis](#) on page 170
- [Enabling or Disabling an Analysis](#) on page 170
- [Specifying Order for Analyses](#) on page 171
- [Saving the Analysis Setup](#) on page 172
- [Restoring a Saved Analysis Setup](#) on page 172
- [Setting Up a Spectre Analysis](#) on page 174
- [Setting Up an UltraSim Analysis](#) on page 223
- [Setting Up an AMS Analysis](#) on page 246
- [Setting Up an HspiceD Analysis](#) on page 255

Required Symbol

You must include an instance of the cell `gnd` from the `analogLib` library in the schematic. Analog simulators need this cell to recognize the DC path to ground.



Setting Up with Different Simulators

To set up analyses,

1. From the Simulation window, choose *Analyses – Choose*, or from the Schematic window, choose *Setup – Analyses*.

The Choosing Analyses form for your simulator appears.

For help setting up a particular analysis, see [Setting Up a Spectre Analysis](#) on page 174, or refer to your simulator manual.

2. Select an analysis.

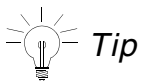
The Choosing Analyses form redraws to show the parameters for the new analysis.

3. Set the analysis options.

4. Click *Apply*.

The analysis you selected displays in the *Analyses* pane of the Simulation window.

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



If you do any select operation from the schematic after you have already invoked the *Choosing Analysis* form, the control does not return back to the analysis form. This can be inconvenient if several windows are open. The control can be made back to the analysis form, by clicking on *Analyses – Choose* once more, in the *Virtuoso® Analog Design Environment* window.

Deleting an Analysis

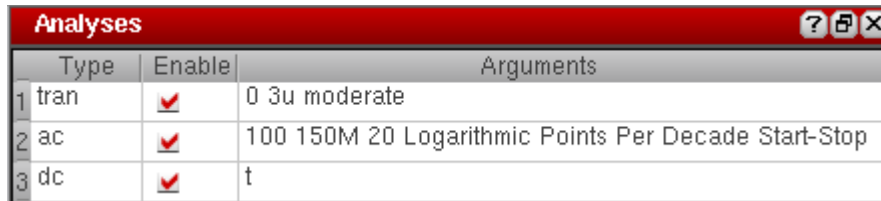
To delete an analysis,

1. In the Simulation window, click an analysis to highlight it.
2. Choose *Analyses – Delete* or click the delete icon.

Enabling or Disabling an Analysis

To temporarily disable an analysis without deleting it from the environment,

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



	Type	Enable	Arguments
1	tran	<input checked="" type="checkbox"/>	0 3u moderate
2	ac	<input checked="" type="checkbox"/>	100 150M 20 Logarithmic Points Per Decade Start-Stop
3	dc	<input checked="" type="checkbox"/>	t

2. Choose *Analyses – Disable*.

To enable a disabled analysis,

1. In the Simulation window, click the analysis to highlight it.
2. Choose *Analyses – Enable*.

Important Points to Note:

- You can also enable and disable analyses with the *Enabled* option in each Choosing Analyses form or in the Analyses pane itself. Click to change the option and then click *Apply*.
- You can use the *Enable* checkbox in the Analyses pane of the ADE L window.
- You can also use the *Enable Analysis* and *Disable Analysis* menu options from the right-click popup menu to enable or disable multiple analyses.

Specifying Order for Analyses

When you set up an analysis using the *Choosing Analyses* form, it gets added as a row at the end of the list of analyses. During simulation, the analyses are run in the same order in which they appear in this list. You can reorder the analyses in this list to specify a different order in which you want the analyses to run. You can change the order by dragging and dropping rows or using the *Move Up* and *Move Down* options from the right-click menu of the *Analyses* section. Currently, it is not possible to drag and drop multiple analyses rows together.

Some simulators require analyses to be run in a pre-defined sequence. For example, the `pss` analysis has to be run before `pac`. Every time you change the order of an analysis, the tool checks that all the analyses are placed in the correct order required by the selected simulator. If you move and place an analysis in an incorrect order, it is placed back at the original location and an appropriate message is displayed. In addition, to prevent unintentional

reordering of analyses, the ability to reorder rows by clicking on the column header in the *Analyses* section has also been disabled.

The order of analyses is maintained while saving and restoring an ADE L state.

Note: Starting 6.1.4 release, the *Analyses Order* field has been removed from the *Environment Options* form. You can specify order for analyses only by arranging them in the *Analyses* section. For all the states saved before the 6.1.4 release, the analyses order specified using the *Environment Options* form is maintained. While loading those states, the analyses are displayed in the same order as specified.

Saving the Analysis Setup

You can save the current settings in the Choosing Analyses forms and later restore these analyses.

To save the analysis setup,

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

The Saving State form appears.

2. Enter a name for the saved simulation state.
3. Check that the *Analyses* box is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring a Saved Analysis Setup

To restore a saved analysis setup,

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

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

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

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

3. Click a state name.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

4. Check that *Analyses* is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Setting Up a Spectre Analysis

To set up analyses for the Spectre simulator,

1. Select *Analyses – Choose*.

The Choosing Analyses form appears.

2. Choose an analysis.

The Choosing Analyses form redisplay to show the parameters for the new analysis.

3. Set the options and click *Apply*.

4. Choose another analysis to set up.

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

For help on setting up a particular analysis, refer to [Analysis Statements](#) chapter in the *Virtuoso Spectre Circuit Simulator Reference*.

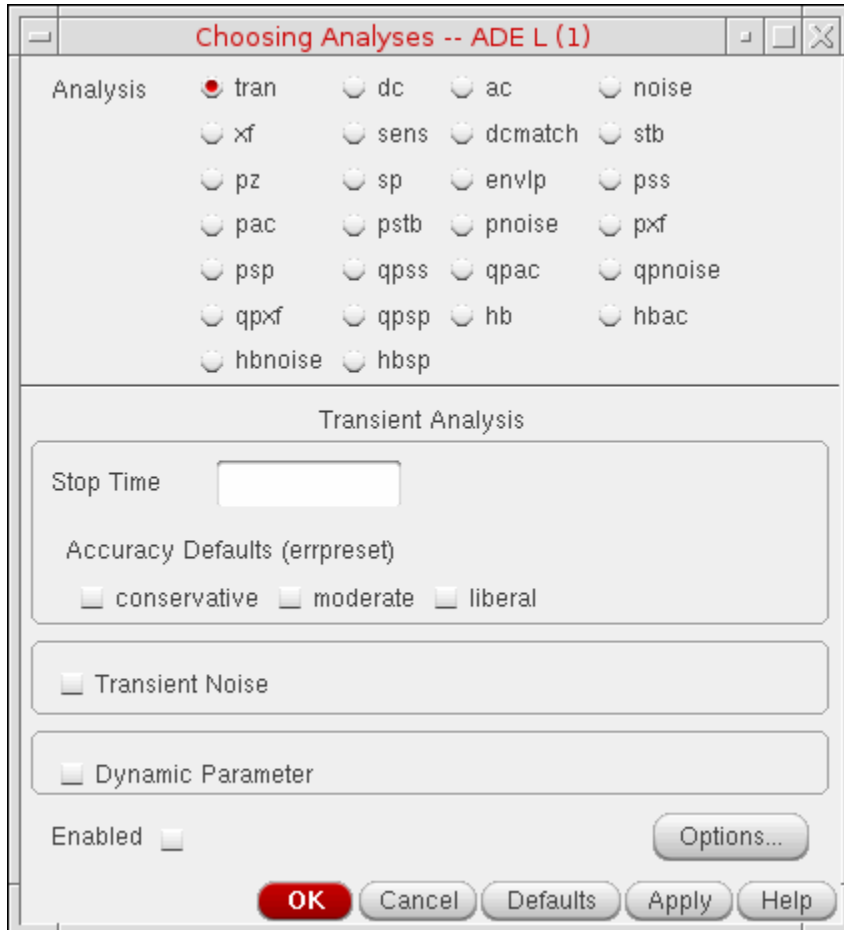
Transient Analysis

The transient analysis computes the transient response of a circuit over an interval. The initial condition is taken to be the DC steady-state solution.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

To set up a transient analysis,



1. In the Choosing Analyses form, select the *tran* option button.
2. Enter the *Stop Time*.
3. Select the default accuracy level for the simulation.

For more information, see the documentation for the `errpreset` parameter in the *Spectre Circuit Simulator Reference*.

4. Select the *Transient Noise* check box if you want to perform transient noise analysis. For more information, see [Transient Noise Analysis](#) on page 180.
5. Select the *Dynamic Parameter* check box to vary temperature, design parameters, options, or transient analysis parameters (such as `reltol`, `residualtol`, `vabstol`, `iabstol`, `isnoisy`) during transient simulation.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- a. Specify the parameter name in the *Parameter Name* field.
- b. Do one of the following:
 - Select *Parameter file* from the drop-down list and specify the path to the file that contains the parameter values that needs to be varied with time.



The format of the file can be as follows:

```
; comments
tscale tscale_value
time value
20 50.0
30 60.0
```

where your comment line starts with ; at the beginning of a line. *tscale* is keyword and *tscale_value* is a value such as 1.0e-6, 1.0e-9, and so on, and is applied to each time point under the time column. *time* and *value* are two key words to identify the time and value columns. The values under the time column define the time points and each time point is scaled by *tscale_value*. The values under the value column define the values for the dynamic parameter.

Note that no unit is supported in the file format.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- Select *Parameter vector* from the drop-down list, specify each time value pair in the *Time* and *Value* fields, then click the *Add* button. The time value pairs are added to the table.

Dynamic Parameter

Parameter Name:

Parameter vector: ▼

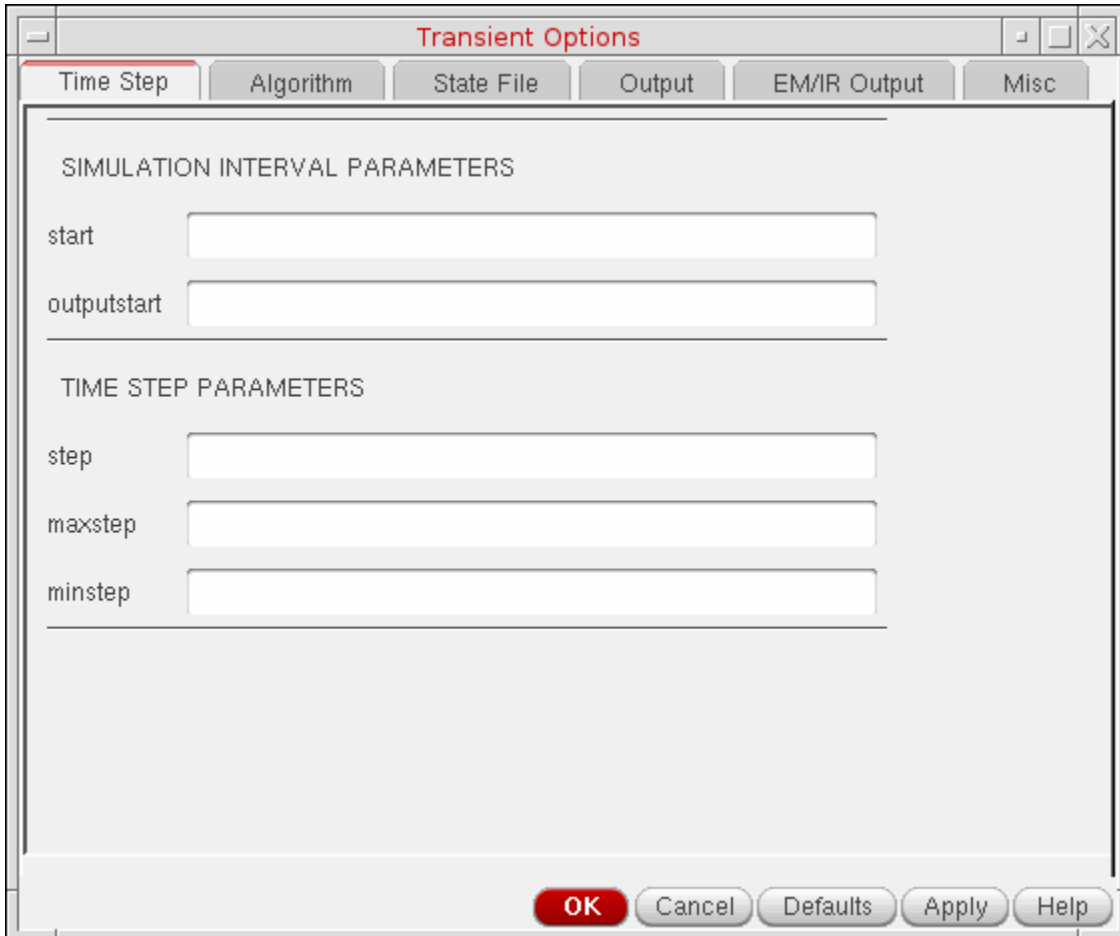
#	time	value
1	20	50.0

Time: Value:

Add Clear Delete

To delete a time value pair, select it in the table and click the *Delete* button.

6. Click *Options* to display the Transient Options form.



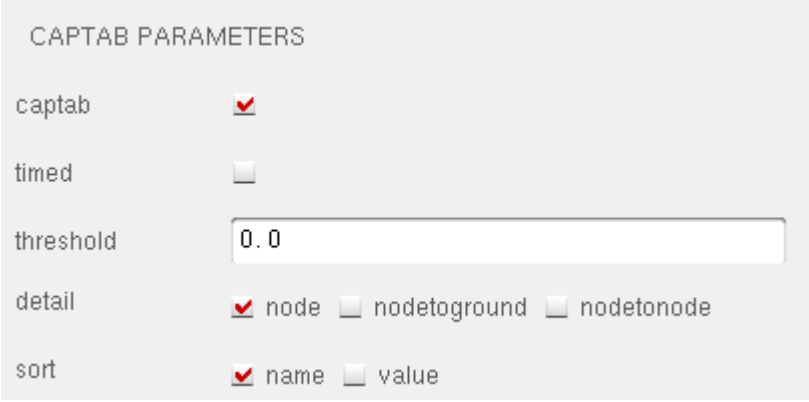
CAPTAB Parameters

You can generate capacitive loading information about a circuit, after a Spectre simulation. For transient analysis, you can specify specific transient timepoints at which to create a capacitance table, using the infotimes parameters of Spectre's transient analysis. If you do not specify these parameters, the capacitance table is generated for the final time point. You

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

can specify these parameters using the following components available on the *transient options* window in the section *CAPTAB PARAMETERS*.



CAPTAB PARAMETERS	
captab	<input checked="" type="checkbox"/>
timed	<input type="checkbox"/>
threshold	<input type="text" value="0.0"/>
detail	<input checked="" type="checkbox"/> node <input type="checkbox"/> nodetoground <input type="checkbox"/> nodetonode
sort	<input checked="" type="checkbox"/> name <input type="checkbox"/> value

captab indicates if you have specified captab parameters. (Enabled or Disabled).

timed indicates if *infotimes* will be used for the purpose of storing captabs instead of operating points (Enabled or Disabled).

threshold indicates the threshold value in real numbers. Results below this value are omitted from the output. The default value is 0.0.

detail can be set to *node*, *nodetoground*, and *nodetonode*. The default option is *node*. To determine the *node to ground* capacitance for *DC* and *Transient* analysis, enable the *nodetoground* button on the appropriate *Options* form. When you run the simulation, the node to ground capacitance for all the nodes is displayed in the log file (*spectre.out*).

sort can be set to *name* and *value*. This can be set in order to sort the entries in the table by their value, or alphabetically by name. The default option is *name*.

For more information about the options in the form, see the “*Transient Analysis (tran)*” section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

Infotimes

Operating-point data can be saved by Spectre during a transient analysis. To control the amount of data produced for operating-point parameters, you can use the infotimes option to specify at which time points you would like to save operating point output for all devices.



infotimes indicates a vector of numbers specifying specific times at which operating-point data is to be collected. Multiple values entered in this field should be separated by blank spaces. If invalid separators or non-numeric values are specified here, Spectre reports the error in the simulation output window. Once infotime values are specified and simulated successfully, clicking the *Results – Print – Transient Operating Point* menu and then selecting a device on schematic, will print the operating point data for all the timepoints saved. If nothing is specified in this field then the infotimes option is not used.

If the user specifies infotimes and then simulates successfully, clicking the *Results – Annotate – Transient Operating Point* menu will bring up the Annotating Transient Operating Points Results form.

Transient Noise Analysis

The current transient analysis has been extended to support transient noise analysis. Transient noise provides the benefit of examining the effects of large signal noise on many types of systems. It gives you the opportunity to examine the impact of noise in the time domain on various circuit types without requiring access to the SpectreRF analyses. This capability is accompanied by enhancements to several calculator functions, allowing you to calculate multiple occurrences of measurements such as risetime and overshoot.

Click on the *Transient Noise* button on the main *Transient Analysis (Analyses – Choose - tran)* form to enable this feature. For information on how to set up to run an analysis in ADE, see Setting Up a Spectre Analysis.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Transient Analysis

Stop Time

Accuracy Defaults (errpreset)

conservative moderate liberal

Transient Noise

Dynamic Parameter

Enabled

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

When the *Transient Noise* option is enabled, the *Choosing Analyses* form re-displays to show the transient noise analysis options.

Transient Analysis

Stop Time

Accuracy Defaults (errpreset)

conservative moderate liberal

Transient Noise

Noise Fmax

Noise Fmin

Noise Seed

Noise Scale

Noise Tmin

Noise Update step fmax

Multiple Runs

Number of Runs

Noise Contribution on off

Instance List

Dynamic Parameter

Enabled

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Set the following parameters to calculate noise during a transient analysis. For more information about the transient noise parameters refer to the *Virtuoso Spectre Circuit Simulator User Guide*.

Parameter	Description
<i>Noise Fmax</i>	The bandwidth of pseudorandom noise sources. A valid (nonzero) value turns on the noise sources during transient analysis. The maximal time step of the transient analysis is limited to $1/\text{Noise Fmax}$.
<i>Noise Fmin</i>	If specified, the power spectral density of noise sources will depend on frequency in the interval from <code>noisefmin</code> to <code>noisefmax</code> . Below <code>Noise Fmin</code> , noise power density is constant. The default value is <code>Noise Fmax</code> , so that only white noise is included and noise sources are evaluated at <code>Noise Fmax</code> for all models. $1/\text{Noise Fmin}$ cannot exceed the requested time duration of transient analysis.
<i>Noise Seed</i>	Seed for the random number generator (used by the simulator to vary the noise sources internally). Specifying the same seed allows you to reproduce a previous experiment. The default value is 1.
<i>Noise Scale</i>	Noise scale factor applied to all generated noise. It can be used to artificially inflate the small noise to make it visible over the transient analysis numerical noise floor, but it should be small enough to maintain the nonlinear operation of the circuit.
<i>Noise Tmin</i>	Minimum time interval between noise source updates. Default is $1/\text{Noise Fmax}$. Smaller values will produce smoother noise signals at the expense of reducing time integration step.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Parameter	Description
<i>Noise Update</i>	<p>Do one of the following:</p> <ul style="list-style-type: none">■ Select the <i>fmax</i> check box to specify that noise is to be injected at a constant time step. Injecting noise at a constant time step is suitable when the value of <i>Noise Fmax</i> is larger than the bandwidth of all signals in the circuit, and simulation time step is effectively controlled by noise. Only one noise frequency is updated at each time step. If the bandwidth of some of the signals exceeds <i>Noise Fmax</i>, which forces the simulator to take steps smaller than <i>Noise Tmin</i>, then noise should also be injected at each time step between the regular noise updates. In this case, all noise frequencies are updated at each time step.■ Select the <i>step</i> check box to specify that noise is to be injected using the Spectre solver time step.

Note: If any of these parameters are not specified, that parameter will not be netlisted. This results in the simulator using its internal default values.

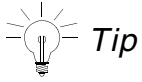
Spectre provides both a single run and multiple run method of simulating transient noise. The single run method, which involves a single transient run over several cycles of operation, is best suited for applications where undesirable start-up behavior is present. The multiple run method, which involves a statistical sweep of several iterations over a single period, is recommended for users who are able to take advantage of distributed processing.

To perform multiple transient noise analysis runs, do the following:

1. Select the *Multiple Runs* check box.
2. Enter the number of times the transient-noise analysis has to be run in the *Number of Runs* field.

The default for this option is *100* (number of runs).

In the distributed mode, set up a *transient noise* analysis, specify the waveform expressions in the *Outputs* section of the ADE simulation window, set the *Number of Tasks* in the *Job Submit* form and netlist and run. For details, refer to the [Submitting a Job](#) section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.



If you switch from single run noise analysis to multi run analysis, adjust the *Stop Time* appropriately. For example, specify 5 iterations of 20us for a single run of 100us.

You can also narrow down the main source of circuit noise by specifying the list of devices and subcircuit instances that are noisy or noise free by doing the following:

1. Do one of the following in the *Noise Contribution* field:
 - Select the *on* check box to specify the list of instances to be considered as noisy during transient noise analysis.
 - Select the *off* check box to specify the list of instances to be considered as not noisy during transient noise analysis.
2. In the *Instance List* field, type the instance names of the devices and subcircuit instances, separated by spaces.

To select instances from the schematic, do the following:

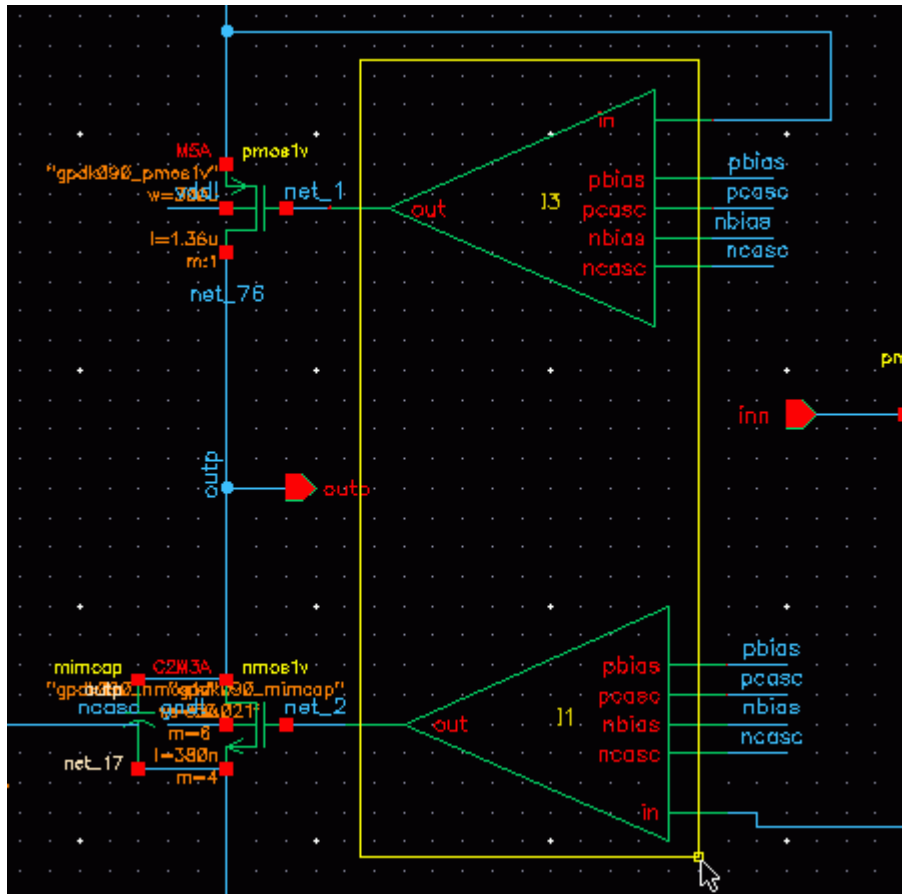
- a. Click the *Select* button next to the *Instance List* field to open the schematic.
- b. Select one or more instances on the schematic. To select more than one instance at a time on the schematic, do one of the following:
 - Hold down the *Shift* key and click on instances.
 - Click and drag the mouse over the instances you want to select.

All the instances that are within the yellow bounding box that appears are included in the selection.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

In the following example, instances I1 and I3 that are within the yellow bounding box are included in the selection.



- c. Press the *Esc* key when you are done.

The selected instances are displayed in the *Instance List* field.

You can specify the options corresponding to transient noise analysis in the *Transient Options* form. [Chapter 6, "Running a Simulation,"](#) describes how to run simulations that you have set up.

Histogram Plots for Transient Noise Analysis

The transient noise analysis is displayed via histogram plots. The *Direct Plot* form corresponding to the transient noise analysis displays a *Histogram* option.

The screenshot shows the 'Direct Plot' form for transient noise analysis. At the top, the 'Plotting Mode' is set to 'Append'. Below this, there are three main sections: 'Analysis', 'Function', and 'Histogram'. In the 'Analysis' section, the 'tran' radio button is selected. In the 'Function' section, the 'Noise Measurement' radio button is selected. The 'Histogram' section contains a 'Number of Bins' field set to '10'. Below this are two list boxes: 'Expression' and 'Plot List'. The 'Expression' list contains two entries: '/Power; trar' and '/out2; tran'. The 'Plot List' list contains one entry: '/Power; trar'. Between the two list boxes are two arrow buttons: a right-pointing arrow ('-->') and a left-pointing arrow ('<--'). Below the 'Expression' list is a 'Retrieve Outputs' button. At the bottom of the form is a 'Plot' button. A note at the very bottom reads: '> Press plot button on this form...'

To plot histograms for the selected waveform measurement(s), click on *Plot* button.

Note: You must create the waveform expressions in the ADE window before plotting a histogram for any waveform measurement.

To add specific waveform measurements to the *Plot List*, select the required waveform measurement in the *Expression* list and use the right-arrow button to add it to the *Plot List*

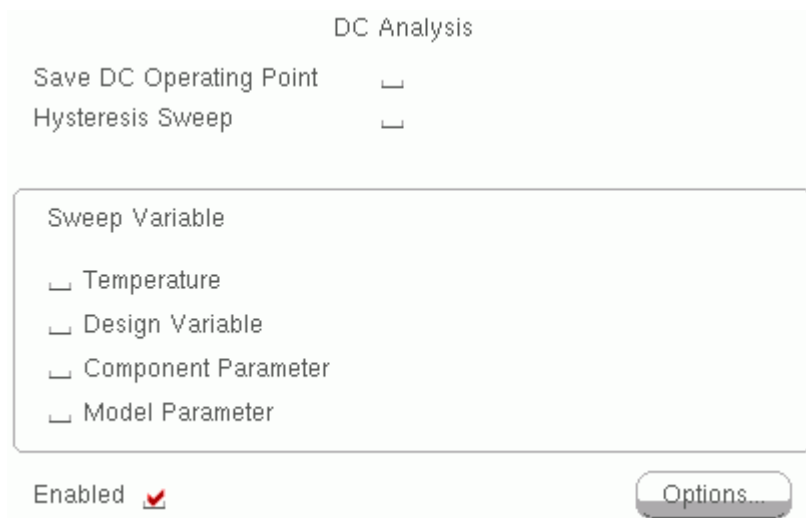
To delete specific waveform measurements from the *Plot List*, select the required waveform measurement in the *Plot List* and use the left-arrow button to put it back in the *Expression* list.

If you create additional waveform expressions (through Calculator) after the *Direct Plot* form has been launched, you can click the *Retrieve Outputs* button to import all the existing waveform expressions into the ADE window.

Note: All normal plots and measurements work as is.

DC Analysis

The DC analysis finds the DC operating point or DC transfer curves of the circuit. To generate transfer curves, specify a parameter and a sweep range. The parameter can be a temperature, a design variable, a device instance parameter, or a device model parameter.



Save DC Operating Point specifies whether DC operating point information must be saved.

Hysteresis Sweep enables or disables DC hysteresis sweep. Select the check box to enable DC hysteresis sweep.

Sweep Variable specifies the parameters to sweep during DC analysis. Select the check box next to the parameters that should be swept during DC analysis.

- To sweep circuit temperature, select the *Temperature* check box.
- To sweep a design variable, select the *Design Variable* check box, then specify the name of the design variable in the *Variable Name* field, or click the *Select Design Variable* button to select the design variable.
- To sweep a device instance parameter, select the *Component Parameter* check box, then click the *Select Component* button.

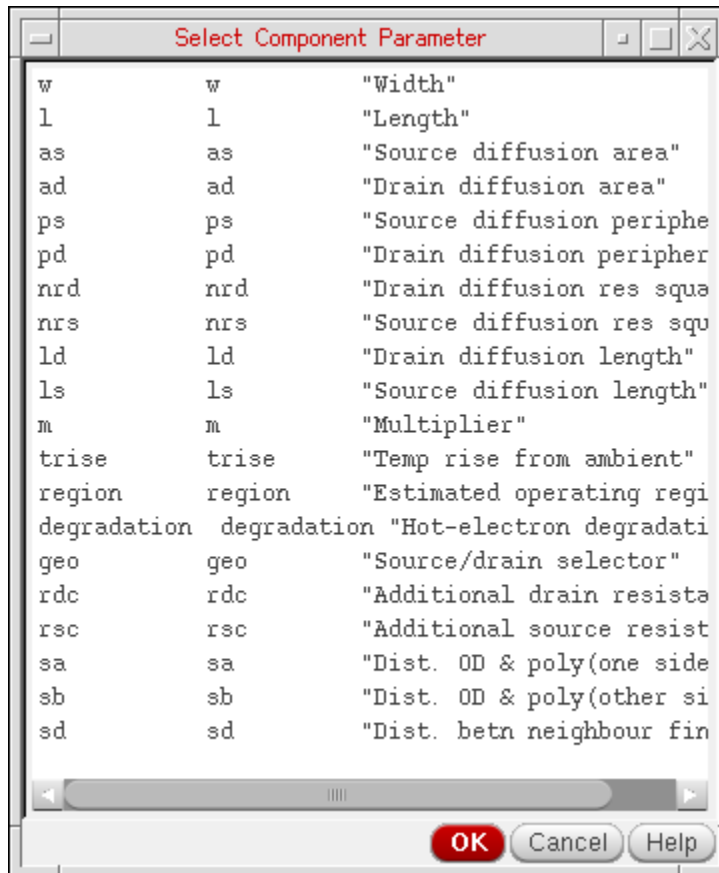
The schematic for the design is displayed. Do the following to select the parameter:

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- a. Click on the instance whose parameter you want to sweep.

The Select Component Parameter form appears.



- b. Select the parameter you want to sweep and click *OK*.

The instance name of the component is displayed in the *Component Name* field and the parameter name is displayed in the *Parameter Name* field.

- To sweep a model parameter, select the *Model Parameter* check box, then specify the model name in the *Model Name* field and the parameter name in the *Parameter Name* field.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

To specify the sweep range for the sweep variables, do the following:

Sweep Range

Start-Stop Start Stop

Center-Span

Sweep Type

Automatic ▼

Add Specific Points

1. Do one of the following:

- Select the *Start-Stop* option to specify the start sweep limit in the *Start* field and the stop sweep limit in the *Stop* field.
- Select the *Center-Span* option to specify the center of the sweep in the *Center* field and the sweep limit span limit in the *Span* field.

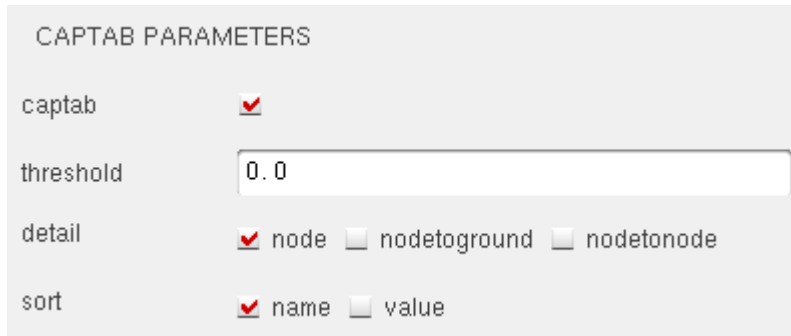
2. Select the sweep type from the *Sweep Type* cyclic field.

- If the sweep type is *Linear*, do one of the following:
 - Select the *Step Size* option and specify the step size for the linear sweep.
 - Select the *Number of Steps* option and specify the number of steps for the linear sweep.
- If the sweep type is *Logarithmic*, do one of the following:
 - Select the *Points Per Decade* option and specify the points per decade for the logarithmic sweep.
 - Select the *Number of Steps* option and specify the number of steps for the logarithmic sweep.

3. (Optional) Select the *Add Specific Points* check box to specify a list of values to sweep. Use spaces to separate each value in the field.

CAPTAB Parameters

You can generate capacitive loading information about a circuit after a Spectre simulation. The following additional components are available on the *dc options* window:



CAPTAB PARAMETERS	
captab	<input checked="" type="checkbox"/>
threshold	<input type="text" value="0.0"/>
detail	<input checked="" type="checkbox"/> node <input type="checkbox"/> nodetoground <input type="checkbox"/> nodetonode
sort	<input checked="" type="checkbox"/> name <input type="checkbox"/> value

captab indicates if you have specified captab parameters. (Enabled or Disabled)

threshold indicates the threshold value in real numbers. Results below this value are omitted from the output. The default value is 0.0

detail includes *node*, *nodetoground*, and *nodetonode*. The default option is *node*

sort includes *name* and *value*. This can be set in order to sort the entries in the table by their value, or alphabetically by name. The default option is *name*.

For details on DC Analysis refer to the [Analysis Statements](#) chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

Sweeping a Variable

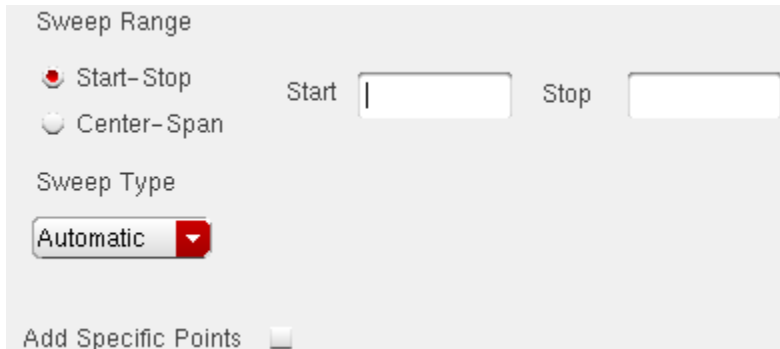
To run a DC transfer curve analysis and sweep a variable,

1. Choose a sweep variable.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The Choosing Analyses form redisplay to show additional fields.



The screenshot shows a control panel for setting up an analysis. It includes a 'Sweep Range' section with two radio buttons: 'Start-Stop' (selected) and 'Center-Span'. The 'Start-Stop' option has two text input fields labeled 'Start' and 'Stop'. Below this is a 'Sweep Type' section with a dropdown menu currently set to 'Automatic'. At the bottom, there is an 'Add Specific Points' checkbox which is currently unchecked.

2. Specify the necessary parameters.

- If you sweep a design variable, fill out the name of the design variable, or choose from the list box after pressing the select button.
- To sweep a component, specify the component name and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- To sweep a model parameter, enter the model and parameter names.

3. Specify the sweep range and type.

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

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

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The DC Options form appears.

The screenshot shows the 'DC Options' dialog box with the following sections and controls:

- STATE-FILE PARAMETERS**
 - force: none node dev all
 - save op in final:
 - readns: [text field] ...
 - readforce: [text field] ...
 - write: [text field] spectre.dc ...
 - writefinal: [text field] ...
- OUTPUT PARAMETERS**
 - save: selected lvl/pub lvl all/pub all
 - nestlvl: [text field]
 - print: yes no
 - check: yes no
- CONVERGENCE PARAMETERS**
 - homotopy: gmin source dptran
 ptran none all
 - restart: yes no
 - maxiters: [text field] 150
 - maxsteps: [text field] 10000
- EM/IR OUTPUT PARAMETERS**
 - emirformat: none vavo

Buttons at the bottom: **OK** (highlighted in red), Cancel, Defaults, Apply, Help.

For more information about the options in the form, see the “*DC Analysis*” section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

5. Click *Apply*.

AC Small-Signal Analysis

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

The parameter can be a frequency, a design variable, temperature, a component instance parameter, or a component model parameter. If changing a parameter affects the DC operating point, the operating point is recomputed on each step.

To set up an AC small-signal analysis,

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Choose *ac* from the Choosing Analyses form to display the appropriate options.

The screenshot shows the 'AC Analysis' dialog box with the following settings:

- Sweep Variable:** Frequency (selected), Design Variable, Temperature, Component Parameter, Model Parameter, None.
- Sweep Range:** Start-Stop (selected), Start: 100, Stop: 150M.
- Sweep Type:** Logarithmic (selected), Points Per Decade (selected), Number of Steps: 20.
- Add Specific Points:** unchecked.
- Specialized Analyses:** None.
- Enabled:** checked.
- Options...** button.

2. Choose a sweep variable option and specify any necessary parameters.
 - If you do not sweep the frequency, specify the frequency at which to sweep the variable.
 - If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
 - If you sweep a component, specify the parameter to sweep. Click *Select Component* to click in the Schematic window and select the component.
 - If you sweep a model parameter, enter the model and parameter names.
3. Specify the sweep range and type.

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

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

4. Click *Options* to select the Spectre options controlling the simulation.

The AC Options form appears.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The screenshot shows the "AC Options" dialog box with the following sections and controls:

- STATE-FILE PARAMETERS**
 - readns: text input field with a browse button (...)
 - prevoppoint: yes no
- INITIAL CONDITION PARAMETERS**
 - force: none node dev all
 - skipdc: yes no
 - readforce: text input field with a browse button (...)
- OUTPUT PARAMETERS**
 - save: selected lvlpub lvl allpub all
 - nestlvl: text input field
 - oppoint: rawfile screen logfile no
- CONVERGENCE PARAMETERS**
 - restart: yes no
- ANNOTATION PARAMETERS**
 - annotate: no title sweep status steps
 - stats: yes no
- ADDITIONAL PARAMETERS**
 - additionalParams: text input field

Buttons at the bottom: **OK** (red), Cancel, Defaults, Apply, Help.

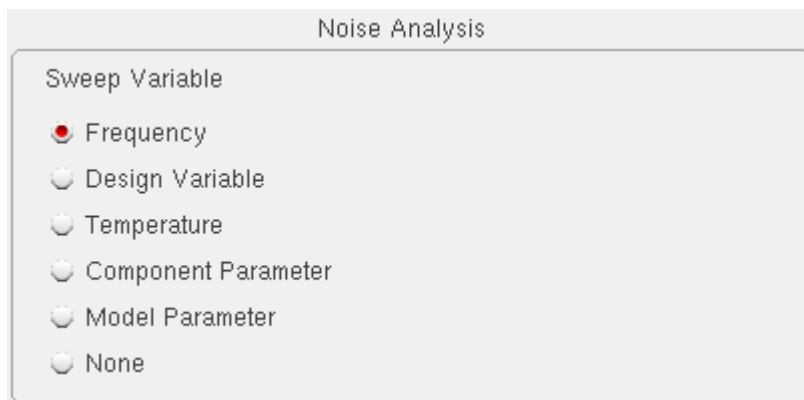
For more information about the options in the form, see the “AC Analysis” section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

5. Click *Enabled* and *Apply*.

Noise Analysis

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

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



- If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- If you sweep a design variable, fill out the name of the design variable, or choose from the list box after pressing the select button.
- If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- If you sweep a model parameter, enter the model and parameter names.

2. Specify the sweep range and type.

The sweep type options are mapped to Spectre statements:

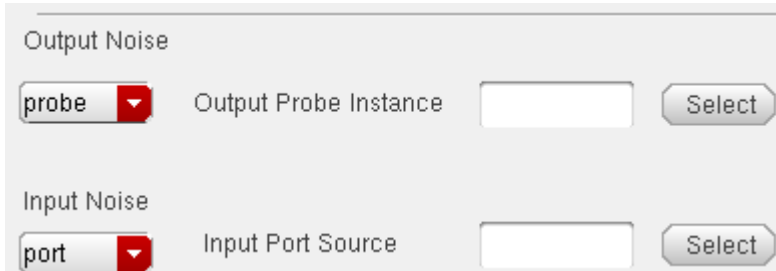
- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- Add Specific Points = values=[...]

3. Choose an *Output Noise* option.



- To measure the output noise voltage, click *voltage* in the cyclic field, and specify values for *Positive Output Node* and *Negative Output Node*, and click a net in the schematic.
- To measure the output noise probe, click *probe* in the cyclic field, and click *Select* opposite *Output Probe Instance*, and click a voltage source in the schematic.

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

4. Optionally, choose an *Input Noise* option.

- Choose *voltage*, *current*, or *port*.
- Click *Select* for *Input Voltage Source* or *Input Current Source* or *Input Port Source*.
- Click a source or port in the schematic.
- Click *Apply*.

5. Click *Options* to set the spectre options controlling noise simulation.

The Noise Options form appears.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The screenshot shows the "Noise Options" dialog box with the following sections and controls:

- STATE-FILE PARAMETERS**
 - readns: text input field with a browse button (...)
 - prevoppoint: yes no
- INITIAL CONDITION PARAMETERS**
 - force: none node dev all
 - readforce: text input field with a browse button (...)
- OUTPUT PARAMETERS**
 - save: selected lvlpub lvl allpub all
 - nestlvl: text input field
 - oppoint: rawfile screen logfile no
- CONVERGENCE PARAMETERS**
 - restart: yes no
- ANNOTATION PARAMETERS**
 - annotate: no title sweep status steps
 - stats: yes no
- ADDITIONAL PARAMETERS**
 - additionalParams: text input field

Buttons at the bottom: **OK** (red), Cancel, Defaults, Apply, Help.

For more information about the options in the form, see the *Noise Analysis* section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

6. Click *Apply*.

S-Parameter Analysis

The S-parameter analysis linearizes the circuit about the DC operating point and computes S-parameters of the circuit taken as an N-port. The psin instances (netlist-to-Spectre port statements) define the ports of the circuit. Each active port is turned on sequentially, and a linear small-signal analysis is performed. The Spectre simulator converts the response of the circuit at each active port into S-parameters and prints these parameters. There must be at least one active port (analogLib psin instance) in the circuit.

The parameter can be a frequency, a design variable, temperature, a component instance parameter, or a component model parameter. If changing a parameter affects the DC operating point, the operating point is recomputed on each step.

To set up an S-parameter analysis,

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Choose *sp* from the *Choosing Analyses* form to display the appropriate options.

The screenshot shows the 'S-Parameter Analysis' dialog box. It is organized into several sections:

- Ports:** A text input field with 'Select' and 'Clear' buttons.
- Sweep Variable:** Radio buttons for Frequency (selected), Design Variable, Temperature, Component Parameter, Model Parameter, and None.
- Sweep Range:** Radio buttons for Start-Stop (selected) and Center-Span. The Start-Stop option has 'Start' and 'Stop' text labels next to empty input fields.
- Sweep Type:** A dropdown menu currently set to 'Automatic'.
- Add Specific Points:** A checkbox that is currently unchecked.
- Do Noise:** Checkboxes for 'yes' (unchecked) and 'no' (checked).
- Mode:** Checkboxes for 'Single-Ended' (checked), 'Mixed In/Out' (unchecked), and 'Other' (unchecked).
- Enabled:** A checkbox that is currently unchecked.
- Options...:** A button at the bottom right.

2. Specify the list of active *Ports*. In this field, the ports are numbered sequentially beginning with one, in the order given. Otherwise, all ports present in the circuit are active and the port numbers used are those that were assigned on the port statements.
3. Choose a sweep variable option and specify any necessary parameters.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
- If you sweep a component, specify the parameter to sweep. Click *Select Component* to select the component in the Schematic window.
- If you sweep a model parameter, enter the model and parameter names.

4. Specify the sweep range and type.

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

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

5. Click *Options* to select the Spectre options controlling the simulation.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The S-Parameter Options form appears.

The screenshot shows the 'S-Parameter Options' dialog box with the following sections and controls:

- STATE-FILE PARAMETERS**
 - readns: text input field with a browse button (...)
 - prevoppoint: yes no
- INITIAL CONDITION PARAMETERS**
 - force: none node dev all
 - readforce: text input field with a browse button (...)
- OUTPUT PARAMETERS**
 - file: text input field with a browse button (...)
 - datafmt: spectre touchstone
 - datatype: realimag magphase dbphase
 - paramtype: s y z yz
 - noisedata: no twoport cy
 - oppoint: rawfile screen logfile no
- NOISE PARAMETERS**
- CONVERGENCE PARAMETERS**
 - restart: yes no
- ANNOTATION PARAMETERS**
 - annotate: no title sweep status steps
- ADDITIONAL PARAMETERS**
 - additionalParams: text input field

Buttons at the bottom: **OK** (highlighted in red), Cancel, Defaults, Apply, Help.

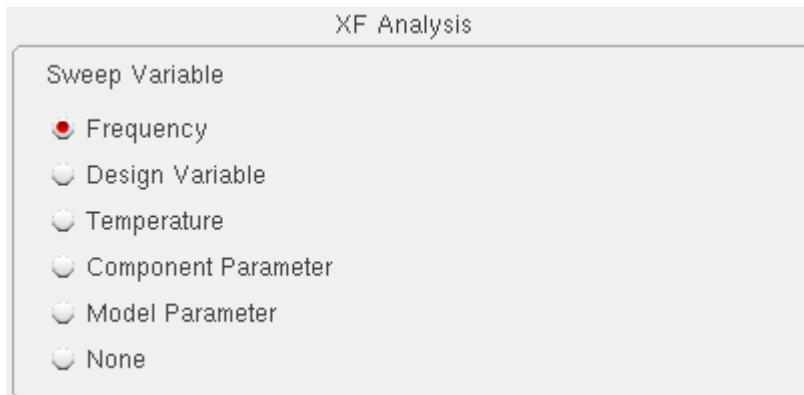
For more information about the options in the form, see the “*S-Parameter Analysis*” section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

6. Click the *Do Noise* radio button to perform noise analysis.
7. Click *Enabled* and *Apply*.

Transfer Function Analysis

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

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



- If you do not sweep the frequency, specify the frequency at which to sweep the variable.
- If you sweep a design variable, fill out the name of the design variable, or select from the list box after hitting the select button.
- If you sweep a component, specify the analysis frequency, component name, and the parameter to sweep. Use the *select component* command to click in the Schematic window to select the component.
- If you sweep a model parameter, enter the model and parameter names.

2. Specify the sweep range and type.



The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

3. Choose *voltage* or *probe* for *Output*.

- To measure the output voltage, click *Select opposite Positive Output Node* and click a net in the schematic.
- To measure the output probe, click *probe*, click *Select opposite Output Probe Instance*, and click an instance in the schematic.

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

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

5. Click *Apply*.

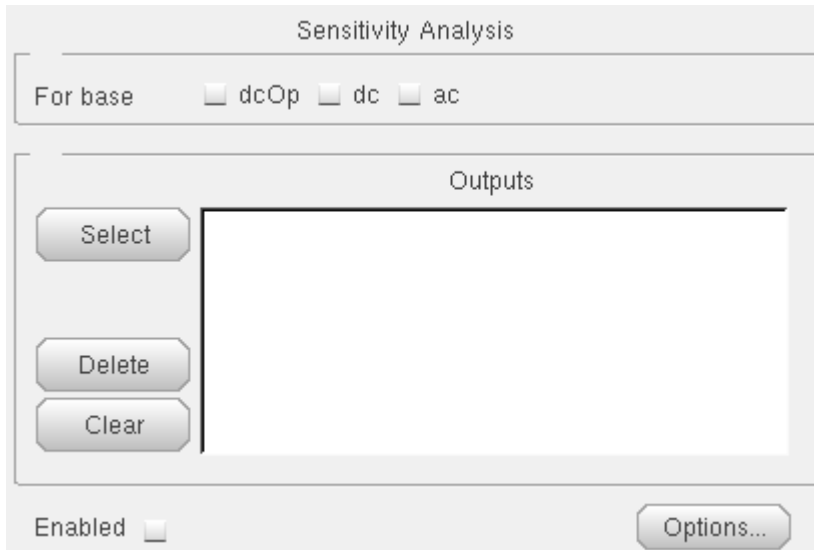
Sensitivity Analysis

Sensitivity analysis lets a designer see which parameters in a circuit most affect the specified outputs. It is typically used to tune a design to increase or decrease certain design goals. You might run a sensitivity analysis to determine which parameters to optimize using the optimizer.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Choose the *sens* radio button on the Choosing Analyses form. The form redraws:



2. Choose which types of sensitivities you want to calculate.

In the *For base* field, choose any of the analyses on which you want to perform a sensitivity analysis. The available analyses are *dcOp* (DC operating point), *dc*, and *ac*.

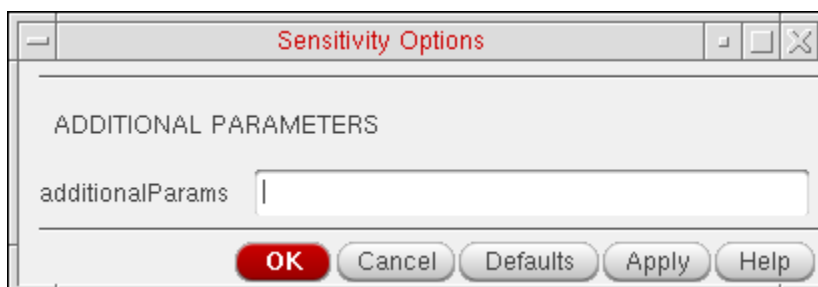
Before you run a sensitivity analysis, you must run the corresponding base analysis.

3. Click *Select* to select the outputs you want to measure.

Select prompts you to select outputs by clicking on their instance in the schematic. Outputs can be any nets or ports. When you click *Select*, the Schematic window moves to the front of the screen. The Schematic window must be open before you can select any outputs. Use the `ESC` key to end selection.

4. Click *Options* to set the spectre options controlling sensitivity analysis.

The Sensitivity Options form appears.



For more information about the options in the form, see the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

5. (Optional) In the Simulation window, choose *Simulation – Options – Analog* to open the Simulator Options form. Scroll down in the form to find the sensitivity options.

Type a filename in the *sensfile* field to specify a filename for the Spectre sensitivity results. This file is in ASCII format, and is generated in the *psf* directory. If you do not specify a value, the file is named `sens.output` by default.

6. View your results.

From the simulation window, choose *Results – Print – Sensitivity*. The results will display in a print window.

DC Mismatch Analysis

The *dcmatch* analysis option performs DC device mis-matching analysis for a given output. It computes the deviation in the DC operating point of the circuit caused by mismatch in the devices. Users need to specify mismatch parameters in their model cards for each device contributing to the deviation. The analysis uses the device mismatch models to construct equivalent mismatch current sources to all the devices that have mismatch modeled. These current sources will have zero mean and some variance. The variance of the current sources is computed according to mismatch models. The analysis computes the 3-sigma variance of dc voltage or current due to the mismatch current sources.

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

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

The *Choosing Analyses* form appears.

2. Choose *dcmatch*.

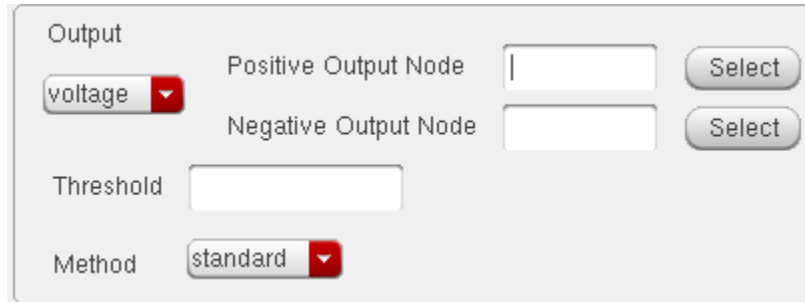
The *Choosing Analyses* form re-displays to show the fields that are required for DC mismatch analysis.

The screenshot shows the 'DC Device Matching Analysis' dialog box. It has a title bar with the text 'DC Device Matching Analysis'. The dialog is divided into several sections. The top section is titled 'Output' and contains a dropdown menu set to 'voltage', two text input fields labeled 'Positive Output Node' and 'Negative Output Node' each with a 'Select' button, a 'Threshold' text input field, and a 'Method' dropdown menu set to 'standard'. The bottom section is titled 'Sweep Variable' and contains four unchecked checkboxes: 'Temperature', 'Design Variable', 'Component Parameter', and 'Model Parameter'. At the bottom left, there is an 'Enabled' checkbox which is unchecked. At the bottom right, there is an 'Options...' button.

3. Specify the output in the *Output* section of the form. You can choose either *Voltage* or *Probe* in the cyclic drop down field.

To specify a *Voltage* output,

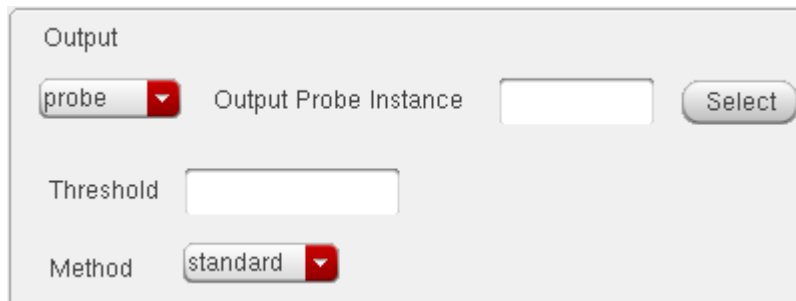
- a. Choose *Voltage* in the cyclic drop down field



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

To specify a current output,

- a. Choose *Probe* in the cyclic drop down field.
- b. Click *Select* opposite *Output Probe Instance* and click a probe in the schematic.



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

Valid Spectre Devices and corresponding analogLib Cells.

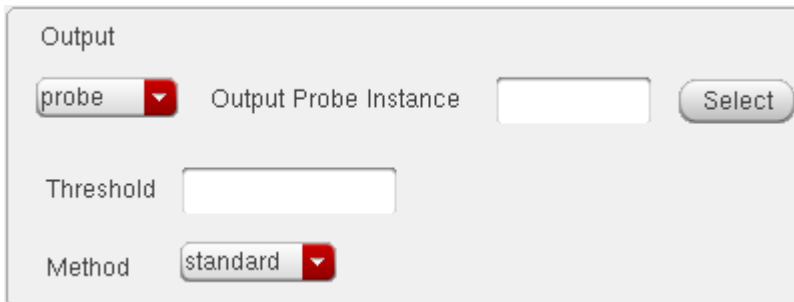
Device	Corresponding analogLib Cells
inductor	ind, pinductor
vsource	vdc, vpulse, vpwl, vpwlf, vsin, vexp, vsource
switch	sp1tswitch, sp2tswitch, sp3tswitch, sp4tswitch

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Device	Corresponding analogLib Cells
tline	tline
controlled voltage source	vcvs, ccvs, sccvs, svcvs, zccvs, zcvcs, pvcvs, pvcvs2, pvcvs3, pccvs
iprobe	iprobe

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



Refer to the [Component Description Format User Guide](#) for information on creating more library components selectable for an analysis.

4. Specify a value in the *Threshold* field to control the number of devices displayed in the output log. The value should be a positive number less than or equal to 1. All devices whose relative contribution falls below the threshold specified are not displayed in the output log.
5. Choose a parameter to sweep in the analysis. The parameters that you can select are *Temperature*, *Design Variable*, *Component Parameter* and *Model Parameter*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

When any of these parameters are selected, the *Sweep Range* section is displayed. Also, the form re-displays according to the parameter that is selected.

DC Device Matching Analysis

Output

probe Output Probe Instance /T0 Select

Threshold

Sweep Variable

Temperature

Design Variable

Component Parameter

Model Parameter

Sweep Range

Start-Stop Start 20 Stop 30

Center-Span

Sweep Type

Automatic

Add Specific Points

Enabled Options...

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

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

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- Add Specific Points = values=[...]

Note: By default, when no sweep variable is selected, the *Sweep Range* section is not displayed.

- Click the *Options* button to open the *Options* form corresponding to the dcmatch analysis. The *DC Device Matching Options* form appears.

The screenshot shows the 'DC Device Matching Options' dialog box. It is organized into several sections:

- STATE-FILE PARAMETERS:** A text field labeled 'readns' with a browse button (...).
- OUTPUT PARAMETERS:** A 'save' section with checkboxes for 'selected', 'lvlpub', 'lvl', 'allpub', and 'all'. A 'nestlvl' text field. An 'oppoint' section with checkboxes for 'rawfile', 'screen', 'logfile', and 'no'.
- CONVERGENCE PARAMETERS:** A 'prevoppoint' section with checkboxes for 'yes' and 'no'. A 'restart' section with checkboxes for 'yes' and 'no'.
- ANNOTATION PARAMETERS:** An 'annotate' section with checkboxes for 'no', 'title', 'sweep', 'status' (checked), and 'steps'. A 'stats' section with checkboxes for 'yes' and 'no'.
- A 'version' text field.
- ADDITIONAL PARAMETERS:** An 'additionalParams' text field.

At the bottom, there are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

Refer to the *Analysis Statements* chapter in the *Virtuoso Spectre Circuit Simulator Reference* for details.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

8. Click *Apply*.

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

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

Stability Analysis

Stability analysis outputs the loop gain for the feedback loop or a gain device. To set up a stability analysis for the Spectre simulator,

1. Select *Analyses – Choose* from the Virtuoso® Analog Design Environment window. The *Choosing Analyses* form appears.
2. Choose *stb*. The *Choosing Analyses* form re-displays to show the fields that are required for the stability analysis.

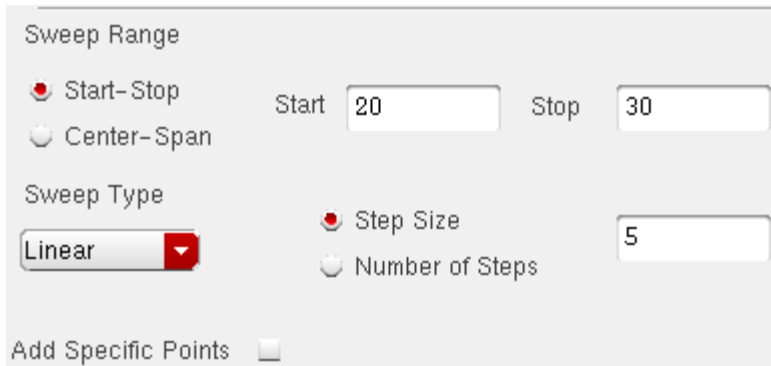
The screenshot shows the 'Stability Analysis' dialog box. It is divided into several sections:

- Sweep Variable:** A group box containing six radio buttons: 'Frequency' (selected), 'Design Variable', 'Temperature', 'Component Parameter', 'Model Parameter', and 'None'.
- Sweep Range:** A group box containing two radio buttons: 'Start-Stop' (selected) and 'Center-Span'. To the right of 'Start-Stop' are two text input fields labeled 'Start' and 'Stop'.
- Sweep Type:** A dropdown menu currently set to 'Automatic'.
- Add Specific Points:** A checkbox that is currently unchecked.
- Probe Instance:** A text input field followed by a 'Select' button.
- Enabled:** A checkbox that is currently unchecked.
- Options...:** A button located at the bottom right of the dialog.

3. Choose a parameter to sweep in the analysis. The parameters that you can select are *Frequency*, *Design Variable*, *Temperature*, *Component Parameter* and *Model Parameter*. For any parameter other than frequency, you need to specify the frequency at which the analysis is to be performed. When the swept parameter is frequency, it also

outputs the phase and gain margins if they can be calculated from the loop gain curve within the swept frequency values.

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



The screenshot shows a configuration dialog box with the following elements:

- Sweep Range:**
 - Start-Stop: Start Stop
 - Center-Span
- Sweep Type:**
 - Step Size:
 - Number of Steps
- Add Specific Points

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

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

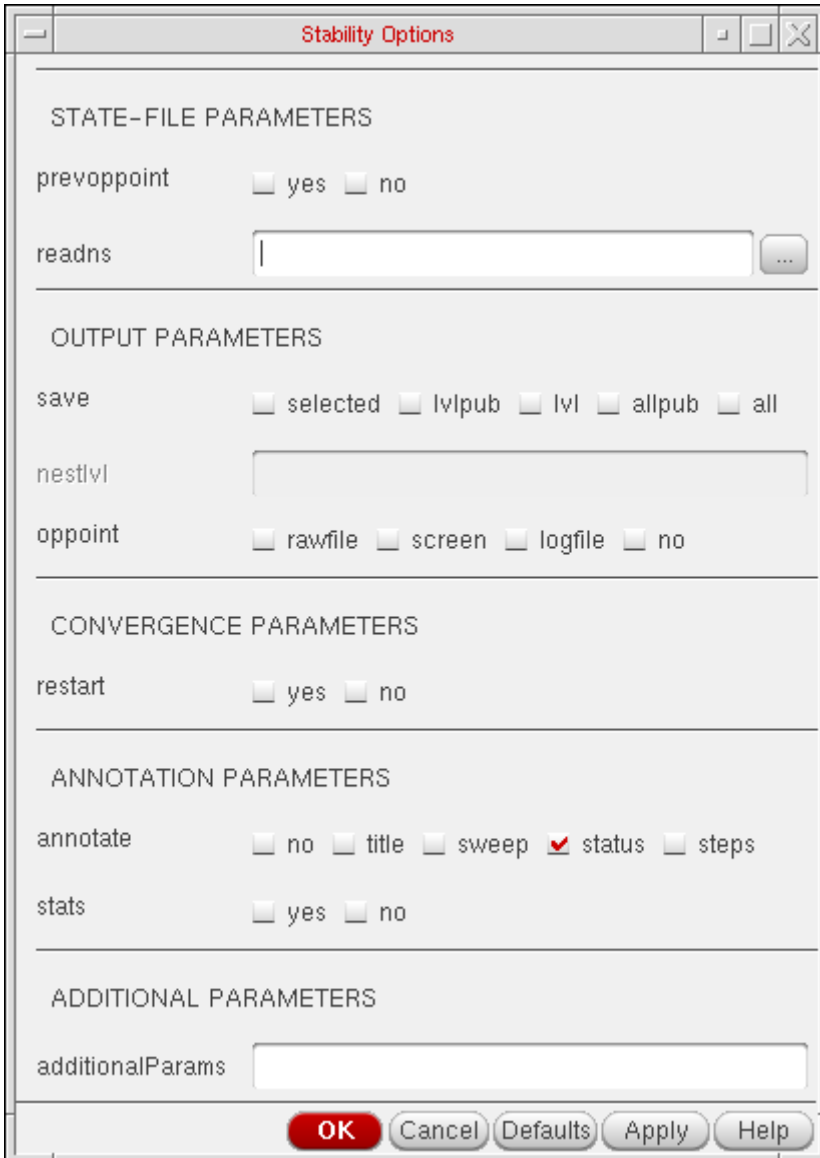
The form changes dynamically as per the current selection.

5. Specify a value in the *Probe Instance* text field. You can use the *Select* button to select the instance from the schematic and the name for the selected instance automatically appears in the text field.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- Click the *Options* button to open the options form corresponding to the stability analysis. The *Stability Options* form appears.



The screenshot shows the 'Stability Options' dialog box with the following sections and controls:

- STATE-FILE PARAMETERS**
 - prevoppoint: yes no
 - readns: (with a browse button)
- OUTPUT PARAMETERS**
 - save: selected lvlpub lvl allpub all
 - nestlvl:
 - oppoint: rawfile screen logfile no
- CONVERGENCE PARAMETERS**
 - restart: yes no
- ANNOTATION PARAMETERS**
 - annotate: no title sweep status steps
 - stats: yes no
- ADDITIONAL PARAMETERS**
 - additionalParams:

Buttons at the bottom: **OK** (highlighted in red), Cancel, Defaults, Apply, Help.

You can refer to the [Analysis Statements](#) chapter in the *Virtuoso Spectre Circuit Simulator Reference* for details.

- Click *Apply*.

To access the results post simulation, choose [Results – Print – Stability Summary](#)

You can also access these results from [Results-Direct Plot](#).

Pole Zero Analysis

Pole Zero Analysis is a useful method for studying the behavior of linear time invariant networks and can be applied to the design of analog circuits. Therefore, it can be used for determining stability of designs.

In *Pole Zero* analysis, a network is described by its network transfer function. For any linear time invariant network, it can be written in the general form:

$$H(S) = \frac{N(S)}{D(S)} = \frac{a_0 S^m + a_1 S^{m-1} + \dots + a_m}{b_0 S^n + b_1 S^{n-1} + \dots + b_n}$$

Similarly, in the factorized form:

$$H(S) = \frac{N(S)}{D(S)} = \frac{a_0}{b_0} \cdot \frac{(S+Z_1)(S+Z_2) \dots (S+Z_j) \dots (S+Z_m)}{(S+P_1)(S+P_2) \dots (S+P_i) \dots (S+P_m)}$$

Here, the roots of the numerator $N(S)$ (that is, Z) are called *zeros* of the network function. The roots of the denominator $D(S)$ (that is, P) are called the *poles* of the network function. S is the complex frequency.

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

For example:

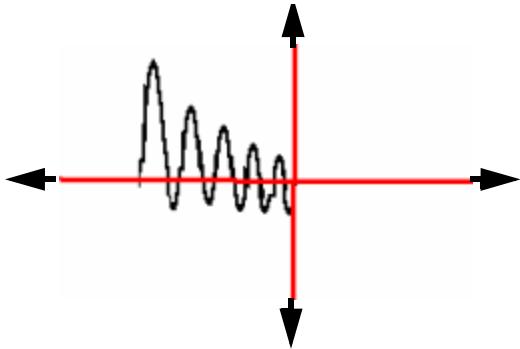
$$H(S) = \frac{(S-2) \cdot (S+1)}{S}$$

Here, the *zeros* are the values of $H(S)$ which make it zero ($S=2$ and $S=-1$). The *poles* make $H(S)$ go to infinity (the pole is at $S=0$)

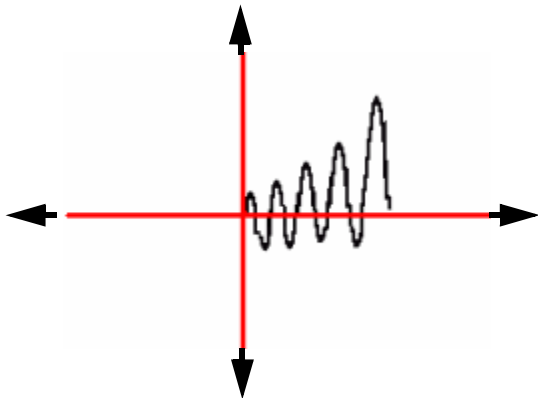
Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

When all the poles have negative real parts, the poles are located on the left hand side of the XY plane. In this situation, the circuit is considered *stable*. The following diagram illustrates the behavior of a stable circuit:



In case there are poles present on the right hand side of the XY plane, the circuit is considered *unstable*. The following diagram illustrates the behavior of an unstable circuit.



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

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

1. Select *Analyses – Choose* from the Virtuoso® *Analog Design Environment* window. The *Choosing Analyses* form appears.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

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

Pole-Zero Analysis

Output

voltage

Positive Output Node

Select

Negative Output Node

Select

Input Source

voltage

Input Voltage Source

Select

Sweep Variable

Frequency

Design Variable

Temperature

Component Parameter

Model Parameter

Component Eval Freq (Hz) 1

Enabled

Options...

3. Specify the output in the *Output* section of the form. You can choose either *Voltage* or *Probe* in the cyclic drop down field.

To specify a *Voltage* output,

- a. Choose *Voltage* in the cyclic drop down field.
- b. Click *Select* opposite *Positive Output Node* and click a net in the schematic for the positive output node. Also, click *Select* opposite *Negative Output Node* and click a net in the schematic.

To specify a current output,

- a. Choose *Probe* in the cyclic drop down field.
- b. Click *Select* opposite *Output Probe Instance* and click an instance (with terminal currents as network variables) in the schematic.

For any other device selection, a warning will be displayed in the CIW stating that the selected object is not a valid type.

Valid Spectre Devices and corresponding analogLib cells:

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

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

When *tline* is the device selected, the *Output* section of the form re-displays to show the *portI* field. This parameter lets you specify a current output that is defined by the device terminal current. Since all of these are two-terminal devices, the current through one of the device terminals would be the same as through the other. The *tline* device is the only one that has more than two terminals.

4. Specify the input voltage or current source by selecting either *voltage* or *current* in the cyclic drop down *Input Source* field of the same form.
5. If you want to sweep a variable in conjunction with the Pole-Zero analysis, choose a parameter to sweep. The parameters that you can select are *Frequency*, *Design Variable*, *Temperature*, *Component Parameter* and *Model Parameter*.

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

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

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

The sweep type options are mapped to Spectre statements:

- Linear + Step Size = step
- Linear + Number of Steps = lin
- Logarithmic + Points Per Decade = dec
- Logarithmic + Number of Steps = log
- Add Specific Points = values=[...]

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

By default, when no sweep variable is selected, the *Sweep Range* section is not displayed.

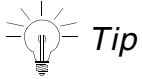
7. Click the *Options* button to open the *Options* form corresponding to the pz analysis. The *Pole-Zero Options* form appears.

The screenshot shows the 'Pole-Zero Options' dialog box with the following sections and parameters:

- STATE-FILE PARAMETERS**: readns (text field)
- OUTPUT PARAMETERS**: oppoint (checkboxes: rawfile, screen, logfile, no); zeroonly (checkboxes: yes, no)
- FILTERING PARAMETERS**: fmax (Hz) (text field); docancel (checkboxes: yes, no); absdiff (Hz) (text field); reldiff (text field)
- CONVERGENCE PARAMETERS**: prevopoint (checkboxes: yes, no); restart (checkboxes: yes, no)
- ANNOTATION PARAMETERS**: annotate (checkboxes: no, title, sweep, status, steps)
- MISCELLANEOUS PARAMETERS**: method (checkboxes: qz, arnoldi); numpoles (text field); numzeros (text field); sigmar (text field); sigmai (text field)
- ADDITIONAL PARAMETERS**: additionalParams (text field)

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

For details, refer to the *Analysis Statements* chapter in the *Virtuoso Spectre Circuit Simulator Reference*.



You can also type `spectre -h pz` in the shell, for help on *Pole Zero* options.

8. Click *Apply*.

To print the results post simulation, choose Results – Print – Pole Zero Summary

You can also plot results from Results – Direct Plot – Main Form.

To plot and print these results from OCEAN, use the OCEAN command, `pzPlot` and `pzSummary`.

Other Spectre Analyses

For information on the Spectre analyses available, see the *Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide*.

Setting Up an UltraSim Analysis

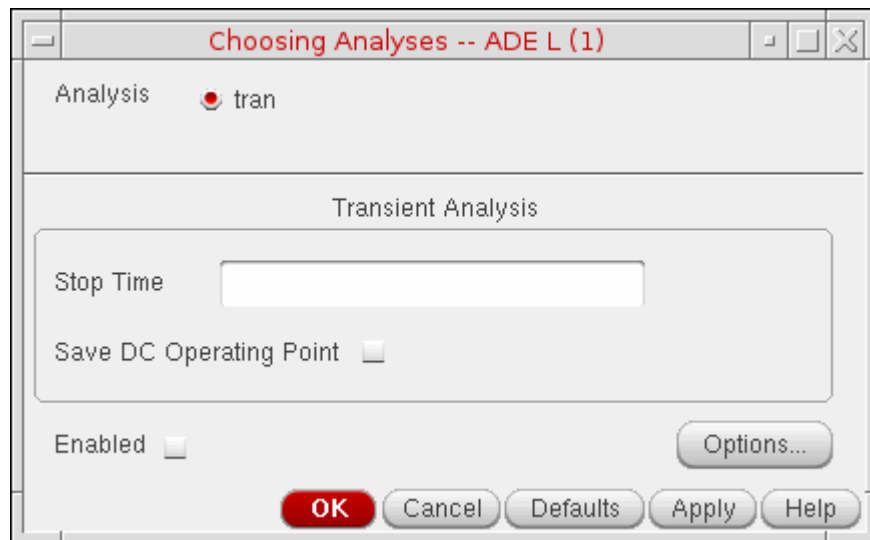
To set up the Virtuoso® UltraSim™ simulator for analysis,

1. In the Simulation window, choose *Analyses – Choose*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The Choosing Analyses form appears.



The *tran* option is selected.

2. Click *Options*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The Transient Options form appears.

The screenshot shows the 'Transient Options' dialog box with the following sections and controls:

- SIMULATION INTERVAL PARAMETERS**
 - start: [text input field]
 - outputstart: [text input field]
- TIME STEP PARAMETERS**
 - step: [text input field] containing '1e-9'
- INITIAL CONDITION PARAMETERS**
 - readic: [text input field] with a browse button (...)
- CONVERGENCE PARAMETERS**
 - readns: [text input field] with a browse button (...)
- STATE FILE PARAMETERS**
 - write: [text input field] with a browse button (...)
 - writefinal: [text input field] with a browse button (...)
- INTEGRATION METHOD PARAMETERS**
 - method: euler trap traponly gear2 gear2only
- ACCURACY PARAMETERS**
 - relref: pointlocal alllocal sigglobal allglobal
- OUTPUT PARAMETERS**
 - skipstart: [text input field]
 - skipstop: [text input field]
 - strobeperiod: [text input field]
 - strobedelay: [text input field]
 - infotimes: [text input field]
- MAXIMUM TIME STEP**
 - [text input field]

At the bottom of the dialog are five buttons: **OK** (highlighted in red), Cancel, Defaults, Apply, and Help.

3. Set the transient simulation options as needed.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- ❑ **start=0 s** transient start time.
- ❑ **outputstart** output is saved after this time is reached.
- ❑ **step** minimum time step used by the simulator to maintain the aesthetics of the computed waveforms.
- ❑ **readic** initial condition is contained in this file (`readic` file is specified relative to the `/netlist` directory).
- ❑ **readns** estimate of the DC solution (`nodeset`) is contained in this file.
- ❑ **write** initial transient solution is written to this file.
- ❑ **writefinal** final transient solution is written to this file.
- ❑ **method** integration method. Values are `euler` (default), `trap`, `traponly`, `gear2` or `gear2only`.
- ❑ **relref** Reference used for the relative convergence criteria. The default is derived from `errpreset`. Possible values are `pointlocal`, `alllocal`, `sigglobal`, and `allglobal`.
- ❑ **skipstart=starttime s** time to start skipping output data.
- ❑ **skipstop=stoptime s** time to stop skipping output data.
- ❑ **strobeperiod (s)** output strobe interval (in seconds of transient time).
- ❑ **strobedelay=0 s** delay (phase shift) between the skipstart time and the first strobe point.
- ❑ **infotimes=[...] s** times when info analysis specified by `infoname` is performed.
- ❑ **maxstep_window** maximum time step (setting `maxstep` to a smaller value can improve simulation accuracy). You can set global or local `maxstep` option for one instance. However, if you want to add `maxstep` options for different instances or subckts, you can add a column in `maxstep_window` and choose string and input in HED. For example, `time1 value1 time2 value 2....`
- ❑ **subckt instance** subcircuit blocks for maximum time step.

By default, the output format of the output of an UltraSim transient analysis is SST2 (SignalScan Turbo 2).

For more information about the transient simulation options, refer to Chapter 2, “Netlist Formats” (*Virtuoso UltraSim Simulator User Guide*).

4. Click *OK*.

5. In the Simulation window, choose *Simulation – Netlist and Run*.

Check the CIW for messages stating that the simulation has started and finished successfully (information is also written to a log file as the simulation runs).

Running Advanced Analysis Simulations

To run a Virtuoso UltraSim simulator advanced analysis,

1. In the Simulation window, choose *Simulation – Options – Analog*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The Simulator Options form appears.

The image shows the 'Simulator Options' dialog box with the 'Main' tab selected. The 'High Level Options' section contains four dropdown menus: 'Simulation Mode' set to 'Mixed Signal (MS)', 'Speed Option' set to 'Default (5)', 'Analog Option' set to 'Default (1)', and 'Post-layout Method' set to 'No RCR (0)'. The 'Multithreading Options' section has a 'multithread' checkbox with 'on' selected and 'off' unselected, and a 'thread' text box containing the number '8'. The 'Temperature Options' section has 'Temperature (C)' and 'Tnom (C)' text boxes, both containing '27'. The 'Skip Subckts' checkbox is checked, and there is a 'Select' button next to it. Below this are two empty text boxes for 'Subckt Instances' and 'Subckt Names'. At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Note: You can set output format using the *Output* tab.

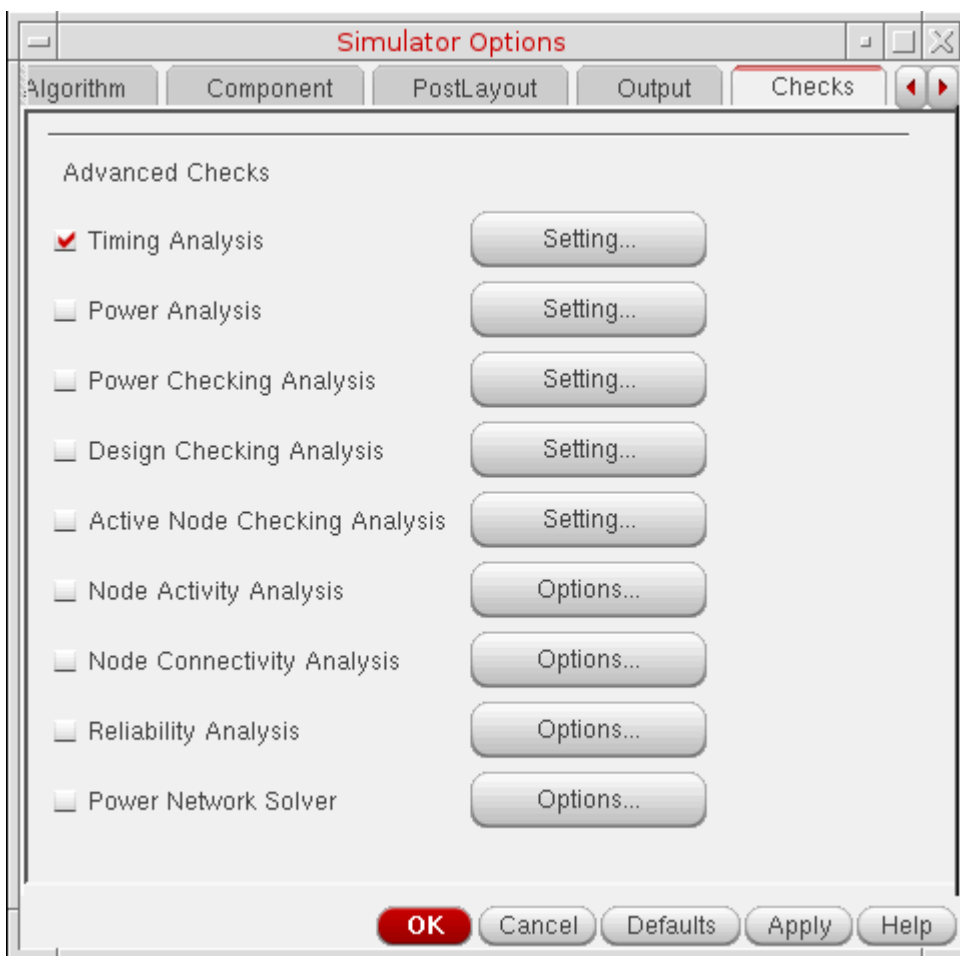
There are eight advanced analysis options in the *Advanced Checks*. The advanced checks can be set using *Checks* tab in the *Simulator Options* form:

- [Timing Analysis](#) on page 229
- [Power Analysis](#) on page 235

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- [Power Checking Analysis](#) on page 236
- [Design Checking Analysis](#) on page 239
- [Active Node Checking Analysis](#) on page 241
- [Parti. and Node Conn. Analysis](#) on page 242
- [Reliability Analysis](#) on page 242
- [Power Network Solver](#) on page 243



2. Choose the appropriate advanced analysis and set the options in the corresponding analysis forms.

Timing Analysis

In timing analysis, you can perform the following checks on signals:

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- [Setup Check](#) on page 231
- [Hold Check](#) on page 232
- [Pulse Width Check](#) on page 233
- [Timing Edge Check](#) on page 234

For more information about the timing analysis settings, refer to Chapter 7, “Virtuoso UltraSim Advanced Analysis” (*Virtuoso UltraSim Simulator User Guide*).

To view a timing analysis, choose *Results – Print – Advanced Analysis Results – Timing Analysis* in the Simulation window.

Setup Check

The screenshot shows the 'Timing Analysis Settings' dialog box. At the top, there is a table with columns: Enable, Type, Title, Signal, and Others... Below the table are three buttons: Add, Change, and Delete. The 'Enabled' checkbox is unchecked. The 'Check Type' dropdown is set to 'setup'. The 'Report Title' field is empty. The 'Signal Name' field is empty with a 'Select' button to its right. The 'Signal Edge Type' dropdown is set to 'rise'. The 'Reference Signal' field is empty with a 'Select' button to its right. The 'Ref Signal Edge Type' dropdown is set to 'rise'. Below these are several empty text input fields for: Setup/Hold Time (sec), Negative setup time Window (sec), Signal Low Threshold (Volts), Signal High Threshold (Volts), Ref Signal Low Threshold (Volts), and Ref Signal High Threshold (Volts). Under the 'Other Options' section, there are empty text input fields for: Depth, Subckt Name, Start, and Stop. At the bottom, there are five buttons: OK (highlighted in red), Cancel, Defaults, Apply, and Help.

1. Adjust the timing analysis settings as needed.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

2. Click *Add* to add a setup check.

Hold Check

The screenshot shows the 'Timing Analysis Settings' dialog box. At the top, there is a table with columns: Enable, Type, Title, Signal, and Others... Below the table are three buttons: Add, Change, and Delete. The 'Add' button is highlighted. Below the buttons, there are several settings:

- Enabled:
- Check Type: hold (dropdown menu)
- Report Title:
- Signal Name: Select
- Signal Edge Type: rise (dropdown menu)
- Reference Signal: Select
- Ref Signal Edge Type: rise (dropdown menu)
- Setup/Hold Time (sec):
- Negative setup time Window (sec):
- Signal Low Threshold (Volts):
- Signal High Threshold (Volts):
- Ref Signal Low Threshold (Volts):
- Ref Signal High Threshold (Volts):
- Other Options:
 - Depth:
 - Subckt Name:
 - Start:
 - Stop:

At the bottom, there are five buttons: OK, Cancel, Defaults, Apply, and Help.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add a hold check.

Pulse Width Check

Timing Analysis Settings

Enable	Type	Title	Signal	Others...
--------	------	-------	--------	-----------

Add Change Delete

Enabled Check Type pulsew ▼

Report Title

Signal Name Select

Min Low Time (sec)

Max Low Time (sec)

Min High Time (sec)

Max High Time (sec)

Signal Low Threshold (Volts)

Signal High Threshold (Volts)

Other Options

Depth

Subckt Name

Start

Stop

OK Cancel Defaults Apply Help

1. Adjust the timing analysis settings as needed.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

2. Click *Add* to add a pulse width check.

Timing Edge Check

Timing Analysis Settings

Enable	Type	Title	Signal	Others...
--------	------	-------	--------	-----------

Add Change Delete

Enabled

Check Type edge

Report Title

Signal Name Select

Signal Edge Type rise

Reference Signal Select

Ref Signal Edge Type rise

Min Permissible Time (sec)

Max Permissible Time (sec)

Signal Low Threshold (Volts)

Signal High Threshold (Volts)

Ref Signal Low Threshold (Volts)

Ref Signal High Threshold (Volts)

Trigger both

Other Options

Depth

Subckt Name

Start

Stop

OK Cancel Defaults Apply Help

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Adjust the timing analysis settings as needed.
2. Click *Add* to add an timing edge check.

Power Analysis

The power analysis reports the power consumed by each element and subcircuit in the circuit. Power analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Power Analysis*.

The screenshot shows the 'Power Analysis Setting' dialog box. It features a table with columns: Enable, Title, Name, Port, Depth, Sorting, and Limit. Below the table are three buttons: Add, Change, and Delete. The 'Enabled' checkbox is unchecked. The 'Report Title' field is empty. The 'Subckt Instances' and 'Subckt Ports' fields are empty, each with a 'Select' button to its right. The 'Depth' field contains the value '1'. The 'Output Sorting' dropdown menu is set to 'avg'. The 'Subckt Limit' field is empty. The 'Power' checkbox is unchecked. The 'Time Intervals' field is empty. The 'Instance Name Length' field contains the value '20'. At the bottom of the dialog are five buttons: OK (highlighted in red), Cancel, Defaults, Apply, and Help.

Enable	Title	Name	Port	Depth	Sorting	Limit
--------	-------	------	------	-------	---------	-------

Buttons: Add, Change, Delete

Enabled:

Report Title:

Subckt Instances: Select

Subckt Ports: Select

Depth:

Output Sorting: avg ▼

Subckt Limit:

Power:

Time Intervals:

Instance Name Length:

Buttons: OK, Cancel, Defaults, Apply, Help

1. Adjust the power analysis settings as needed.
2. Click *Add* to add a power analysis check.

For more information about the power analysis settings, refer to *Virtuoso UltraSim Advanced Analysis* chapter in the *Virtuoso UltraSim Simulator User Guide*.

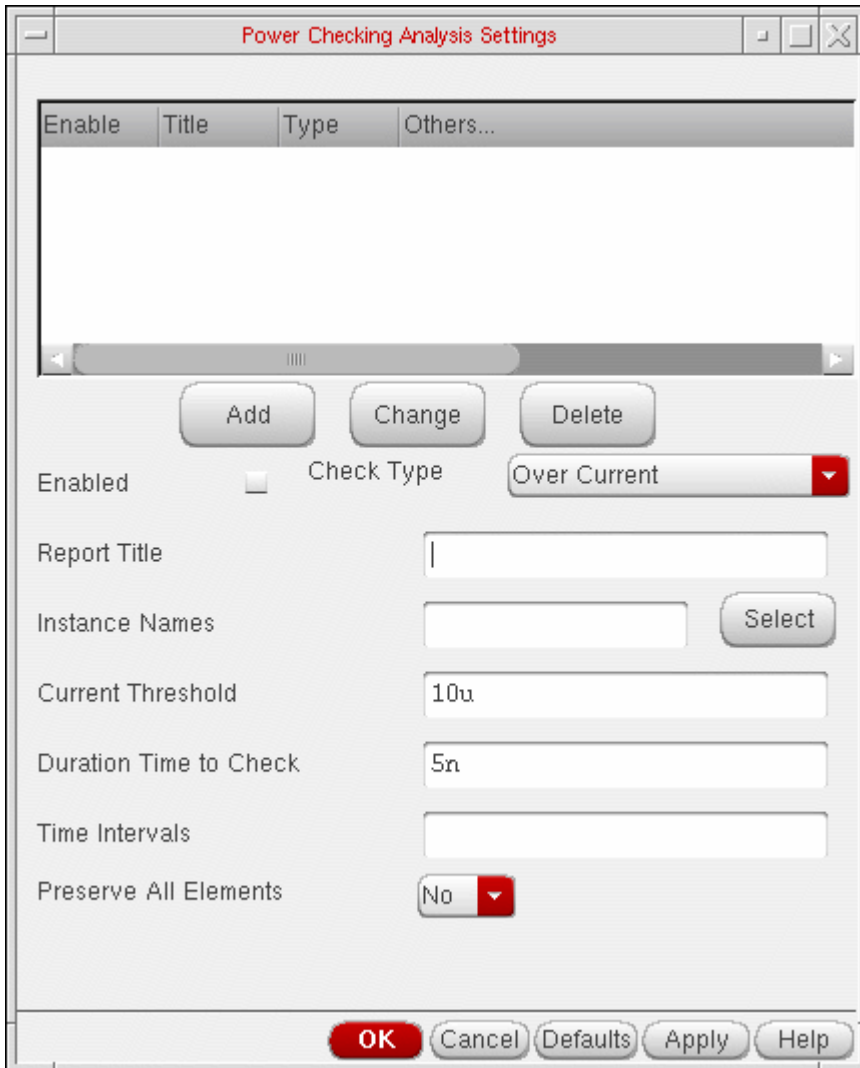
Power Checking Analysis

Based on the specified element list (current threshold, over current duration time, and checking windows), the Virtuoso UltraSim simulator reports in a `.pcheck` file which elements over what time period have current over the threshold for a time period equal to or greater than the specified duration. If no window is specified, the whole simulation period is used.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Power checking results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Power Check Report*.



1. Adjust the power checking analysis settings as needed.
2. Click *Add* to add a power check.

For more information about the power checking analysis settings, refer to [Virtuoso UltraSim Advanced Analysis](#) chapter in the *Virtuoso UltraSim Simulator User Guide*.

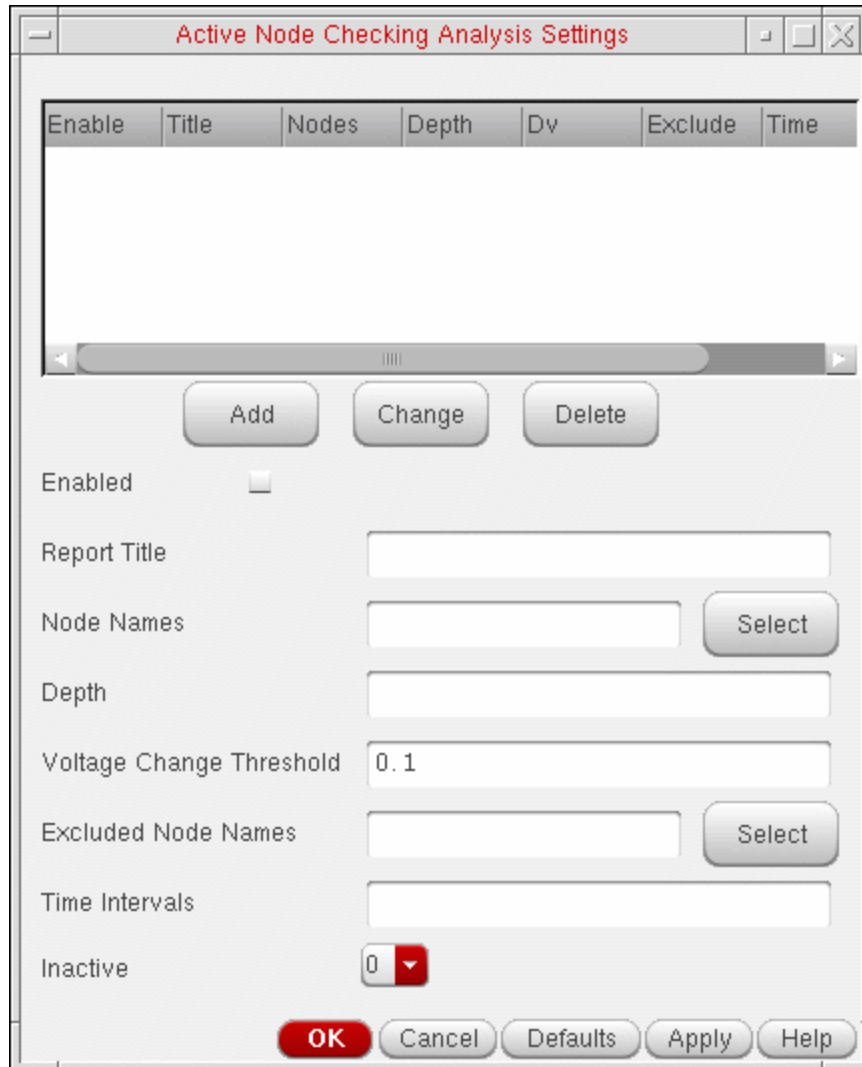
Active Node Checking Analysis

Active node checking analysis detects nodes with voltage changes that exceed the user defined threshold. With the active nodes identified, you can choose to selectively

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

backannotate parasitic elements during post-layout simulation. Active node checking results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Active Node Check Report*.



1. Adjust the node checking analysis settings as needed.
2. Click *Add* to add an active node check.

For more information about the active node checking analysis settings, refer to [Virtuoso UltraSim Advanced Analysis](#) chapter in the *Virtuoso UltraSim Simulator User Guide*.

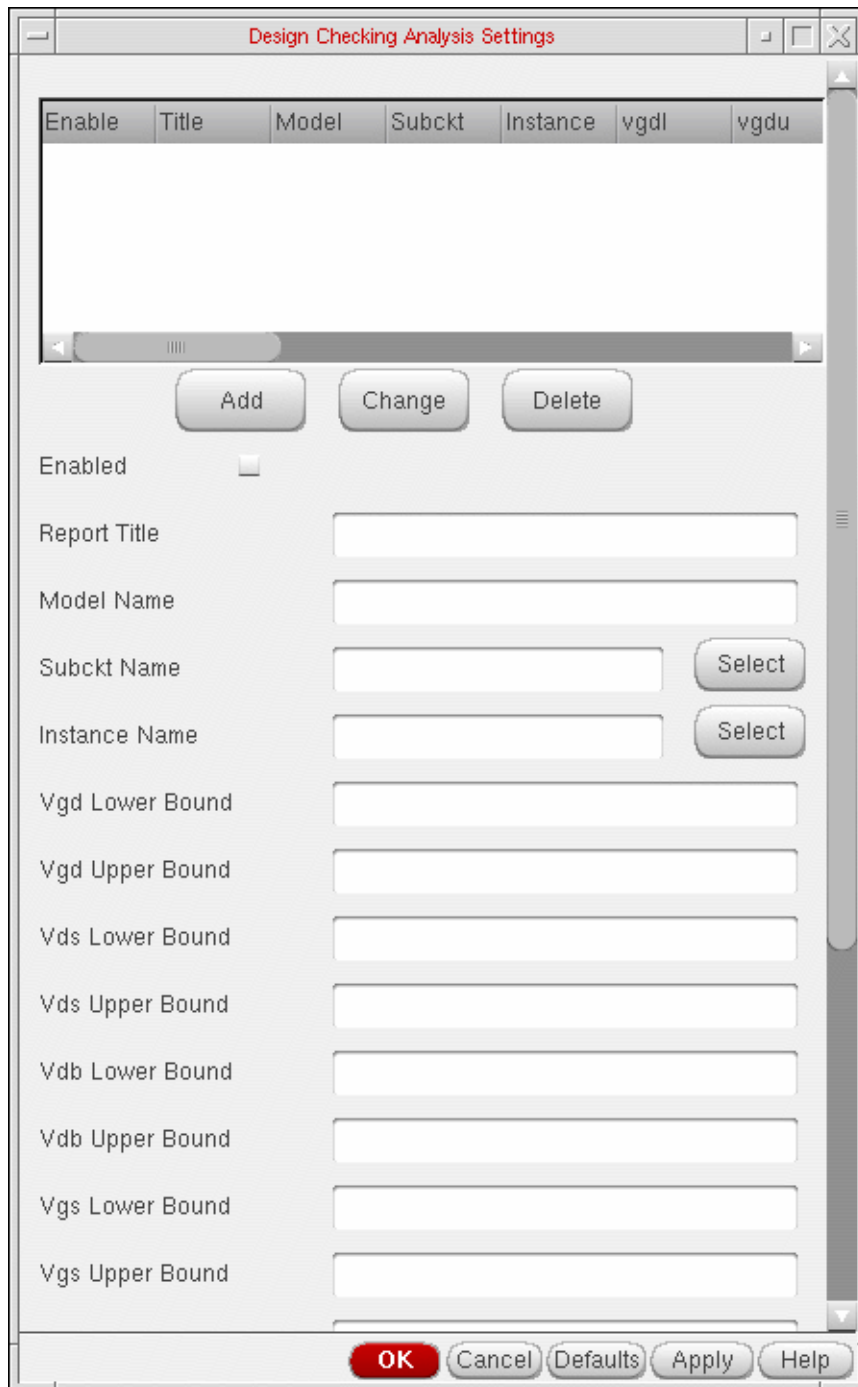
Design Checking Analysis

This command allows you to monitor device voltages during a simulation run, and generates a report if the voltages exceed the specified upper and lower bounds. Design checking results

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Device Voltage Check Report*.



1. Adjust the design checking analysis settings as needed.
2. Click *Add* to add a design check.

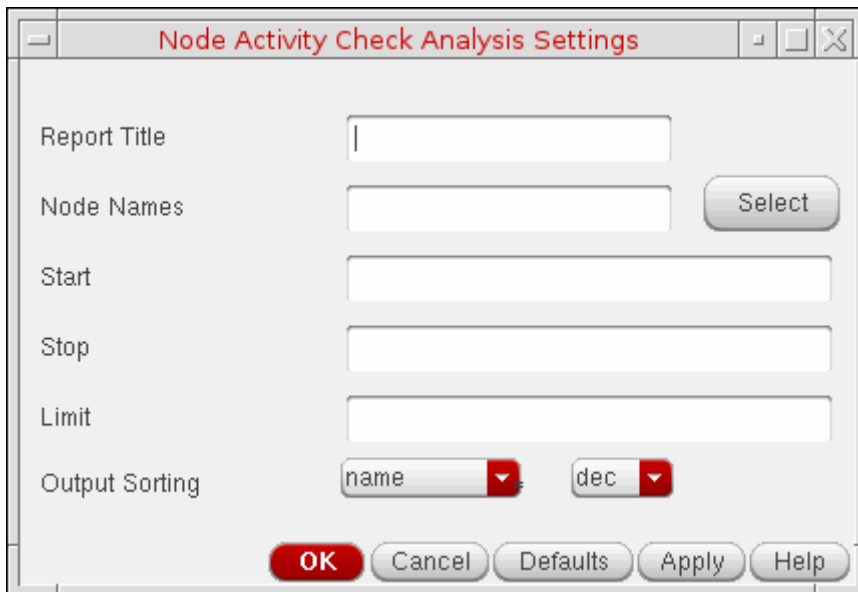
Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

For more information about the design checking analysis settings, refer to *Virtuoso UltraSim Advanced Analysis* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Active Node Checking Analysis

The node activity analysis provides information about the nodes and monitors activities such as voltage overshoots (VOs) and voltage undershoots (VUs), maximum and minimum rise/fall times, signal probability of being high or low, node capacitance, and number of toggles. Node activity analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Node Activity Analysis*.



1. Enter the Node Name or select the same from the schematic.
2. Enter the Start and Stop time.
3. Adjust the node activity analysis settings as needed.
4. Click *OK* to add a node activity analysis check.

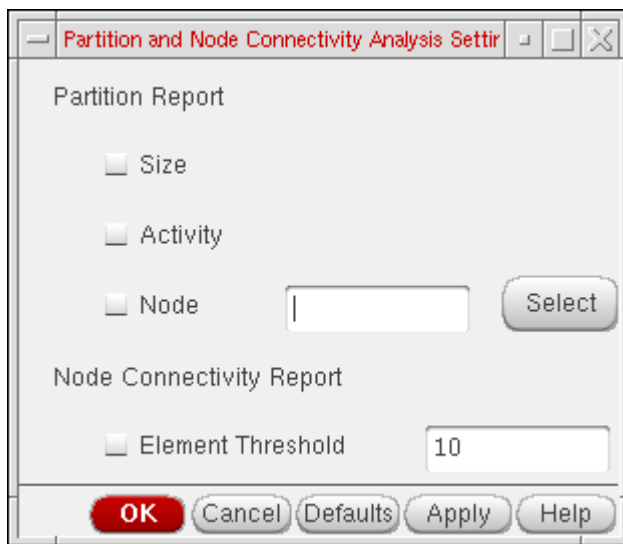
For more information about the node activity analysis settings, refer to *Virtuoso UltraSim Advanced Analysis* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Parti. and Node Conn. Analysis

The Virtuoso UltraSim simulator lets you perform partition and node connectivity analysis using `.usim_report` commands. The information is reported in a `.pr` file. For example, if the netlist name is `circuit.sp`, then the report is named `circuit.pr`.

The `.usim_report` commands are useful for debugging simulations. For example, checking the size of partitions and their activities, as well as checking node activity to verify bus nodes.

Partition and Node Connectivity Analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Parti. and Node Conn. Analysis*.



1. Adjust the partition and node connectivity settings as needed.
2. Click *OK* to add a partition and node connectivity analysis check.

For more information about the partition and node connectivity analysis settings, refer to *Virtuoso UltraSim Advanced Analysis* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Reliability Analysis

The reliability analysis simulates circuit aging due to hot carrier injection (HCI) induced degradation and negative bias thermal instability (NBTI). It can also simulate degraded circuit performance for the above effects, after a specified amount of circuit operation.

To run a reliability analysis, you need reliability models which are usually provided by your modeling group. Reliability models can be added using *Setup – Model Libraries* in the Simulation window.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Note: Reliability analysis is only available with HSPICE netlist format. Reliability analysis results can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – Reliability Analysis*.

Reliability Analysis Settings

HCI/NBTI Mode: HCI only

MOS HCI/NBTI Aging Time (s): 10y|

Age Method: interp

Domain: loglog

Ageproc Include File: [Browse]

NBTI Ageproc Include File: [Browse]

Lifetime Calculation:

Degradation Criterion: 0.1

Min Age Threshold: 0

OK Cancel Defaults Apply Help

1. Adjust the reliability activity analysis settings as needed.
2. Click *OK* to add a reliability analysis check.

For more information about the reliability activity analysis settings, refer to *Virtuoso UltraSim Advanced Analysis* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Power Network Solver

The Power network solver is an optimized solver designed to analyze linear power networks. The solver is integrated into Virtuoso Ultrasim Simulator and together with the Virtuoso Ultrasim engine, lets you calculate the IR drop in power networks and analyze the effects of IR drop on circuit behaviour.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Power Network solver can be viewed from the Simulation window using *Results – Print – Advanced Analysis Results – IR Drop Report*.

The screenshot shows the 'Power NetWork Solver' dialog box. It is divided into several sections:

- DETECT/ANALYZE OPTIONS**: Contains a table with columns 'Node', 'Pattern', and 'Method'. Below the table are 'Add', 'Change', and 'Delete' buttons. There is a 'Power Node' text field with a 'Select' button, a 'Patterns' text field, and a 'UPS Method' dropdown menu set to 'short'.
- Automatic detection**: A dropdown menu set to 'Yes'.
- Auto-detection Method**: A dropdown menu set to 'keep'.
- Detection Level**: A dropdown menu set to 'Most Conservative (0)'. Below it is a 'Resistor Value Threshold' text field.
- EXCLUDE RESISTORS/CAPACITORS**: A 'Disable' checkbox is checked. Below it are 'Instance Names', 'Model Names', and 'File Names' text fields, each with a 'Select' button.
- POWER NETWORK SOLVER**: Contains 'UPS Iteration' (text field with '1'), 'UPS Speed' (dropdown menu set to 'Default (3)'), and 'IR Threshold specification to be Reported' with 'Average' and 'Peak' text fields.

At the bottom of the dialog are 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help' buttons.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

1. Adjust the network power solver settings as needed.
2. Click *OK* to add a this check.

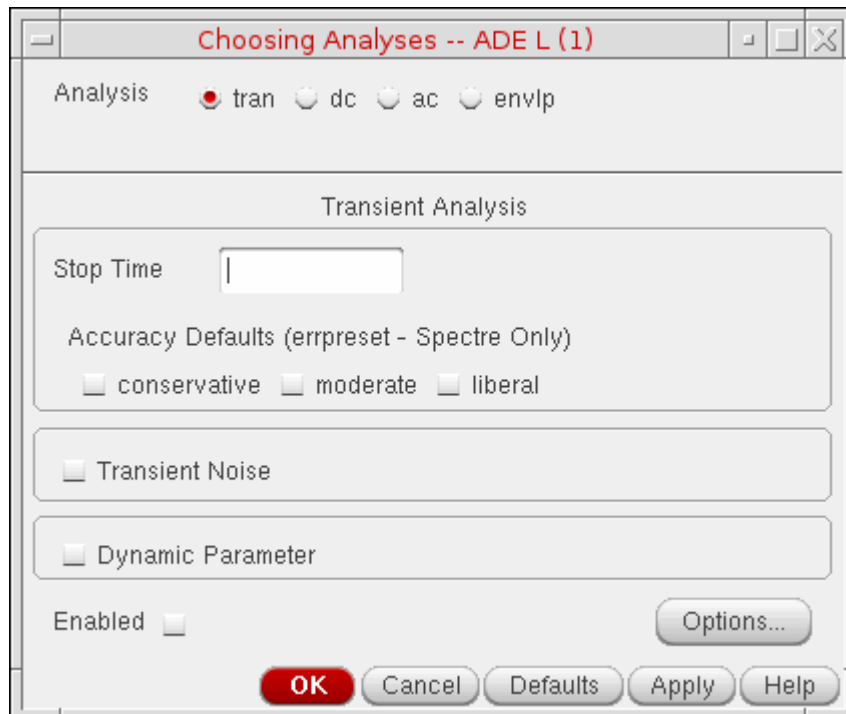
For more information about the Power Network Solver, refer to refer to *Power Network Solver* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Setting Up an AMS Analysis

To set up the Virtuoso AMS Designer simulator for analysis,

1. In the Simulation window, choose *Analyses – Choose*.

The Choosing Analyses form appears.



The Virtuoso AMS Designer simulator supports transient and dc analyses. It supports ac analysis only when `spectre` is selected as the solver. It supports envelope analysis when Spectre, or Ultrasim is selected as the solver. For more information about transient analysis, see [Transient Analysis](#) on page 174.

In this example, the `tran` option is selected. This computes the transient response of the circuit over a specified time interval. You can adjust transient analysis parameters in several ways to meet the needs of your simulation. You can influence the speed of the simulation by setting parameters that control accuracy requirements and the number of data points saved.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

2. Click *Options*. The *Transient Options* form appears.



3. Set the transient simulation options as needed.

Time Step

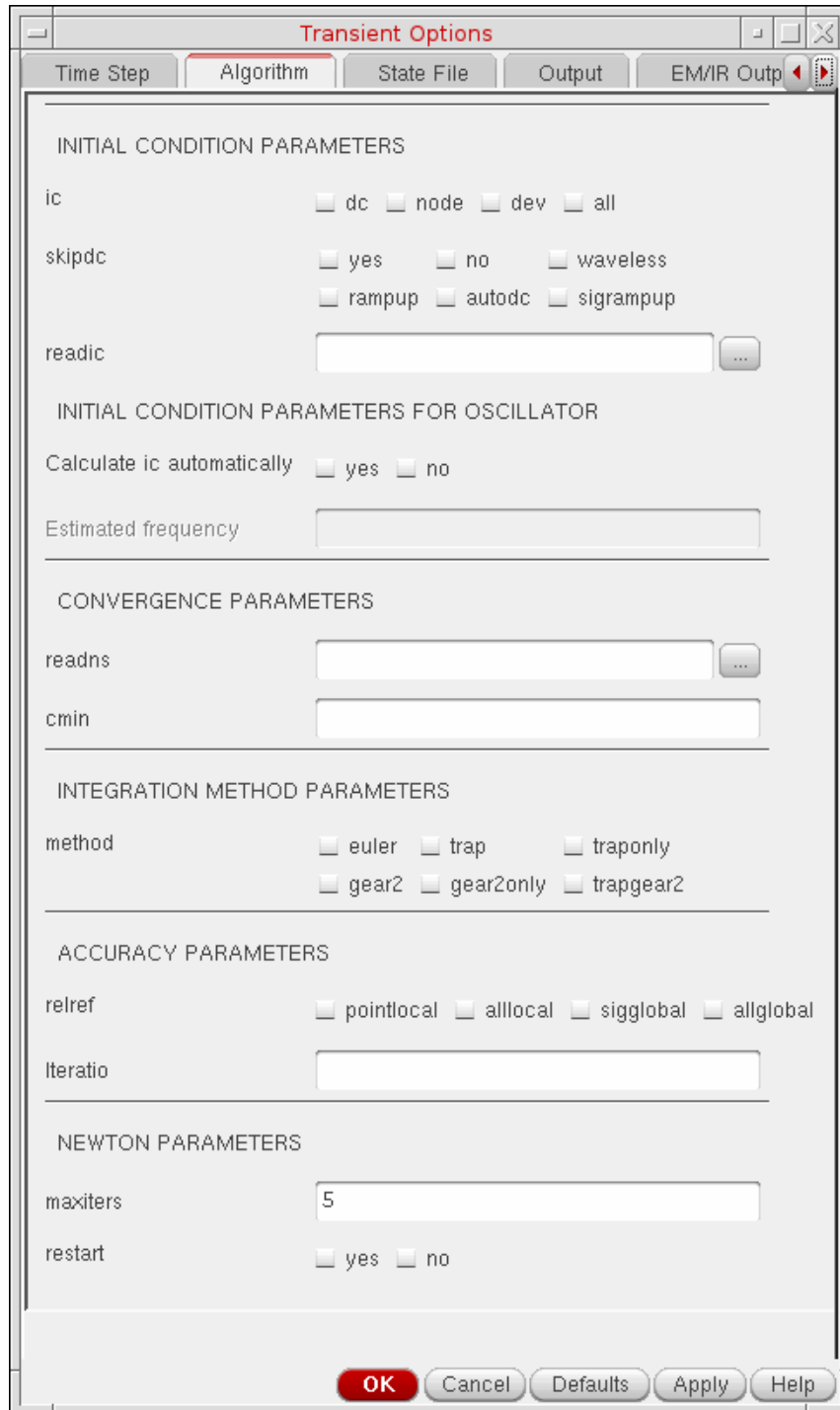
- start=0 s** transient start time.
- outputstart=start s** Output is saved only after this time is reached.
- step=1e-9 s** minimum time step used by the simulator to maintain the aesthetics of the computed waveforms.
- maxstep (s)** Maximum time step. The default is derived from `errpreset`.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- ❑ **minstep (s)** Minimum time step. If specified, the error tolerance requirements may be ignored.
- ❑ **transres=1e-9 stop s** Transition resolution. The transient analysis attempts to stop at corners of input waveforms (for example, corners of rising/falling edge of a pulse). If such events occur within a time less than `transres`, the analysis combines the events into one and forces only one time point. The rest of the steps are determined by error control. This may lead to loss of detail.
- ❑ **maxstep_window** maximum time step (setting `maxstep` to a smaller value can improve simulation accuracy). You can set global or local `maxstep` option for one instance. However, if you want to add `maxstep` options for different instances or subckts, you can add a column in `maxstep_window` and choose string and input in HED. For example, `time1 value1 time2 value 2....`
- ❑ **max_subckt** subcircuit blocks for maximum time step.

Algorithm



- ic=all** What should be used to set initial condition. Possible values are `dc`, `node`, `dev`, and `all`.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

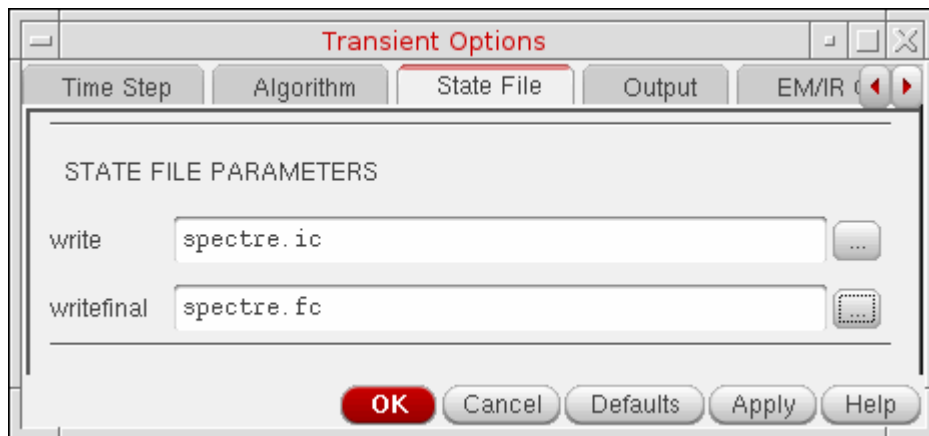
- ❑ **skipdc=no** If yes, there will be no DC analysis for transient. If the DC analysis is skipped, the initial solution is either trivial, or given in the file that you specify using the `readic` parameter. If the `readic` parameter is not specified, the values specified on the `ic` statements are considered. Device-based initial conditions are not used for `skipdc`. Nodes that you do not specify with the `ic` file or `ic` statements start at zero. You should not use this parameter unless you are generating a nodeset file for circuits that have trouble in the DC solution. Possible values: `no`, `yes`, `waveless`, `rampup`, `autodc`.



Using the AMS Designer simulator, setting `skipdc` to anything other than `no` can cause incorrect signal transitions between analog and digital during transient analysis.

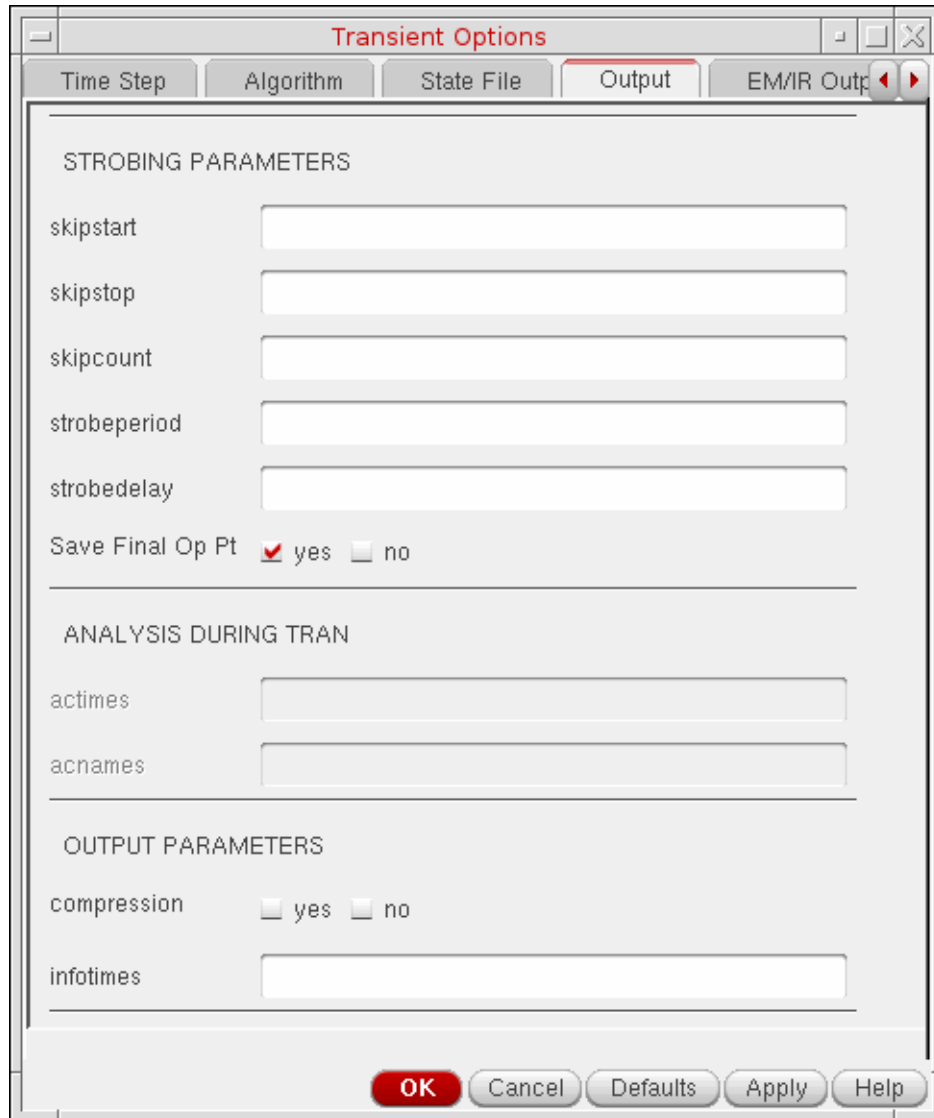
- ❑ **readic** File that contains the initial conditions.
- ❑ **Calculate ic automatically=no** If yes, enable linear IC method to calculate initial conditions automatically from a type of stability analysis in the range $[0.5 \cdot \text{oscfreq}, 1.5 \cdot \text{oscfreq}]$. Overrides user-defined initial conditions, if instability is detected. Possible values are `no` and `yes`.
- ❑ **Estimated frequency** Estimation of the oscillation frequency when `Calculate ic automatically` option is enabled.
- ❑ **readns** File that contains the estimate of initial transient solution.
- ❑ **cmin=0 F** Minimum capacitance from each node to ground.
- ❑ **method** Integration method. Default derived from `errpreset`. Possible values are `euler`, `trap`, `traponly`, `gear2`, `gear2only`, and `trapgear2`.
- ❑ **relref** Reference used for the relative convergence criteria. Default derived from `errpreset`. Possible values are `pointlocal`, `alllocal`, `sigglobal`, and `allglobal`.
- ❑ **Iteratio** Ratio used to compute LTE tolerances from Newton tolerance. The default is derived from `errpreset`.
- ❑ **fastbreak=no** If yes, VHDLAMS Break statement is handled using faster Verilog method. Possible values are `no` and `yes`.
- ❑ **maxiters=5** Maximum number of iterations per time step.

State File



- ❑ **write** initial transient solution is written to this file.
- ❑ **writefinal** final transient solution is written to this file.

Output



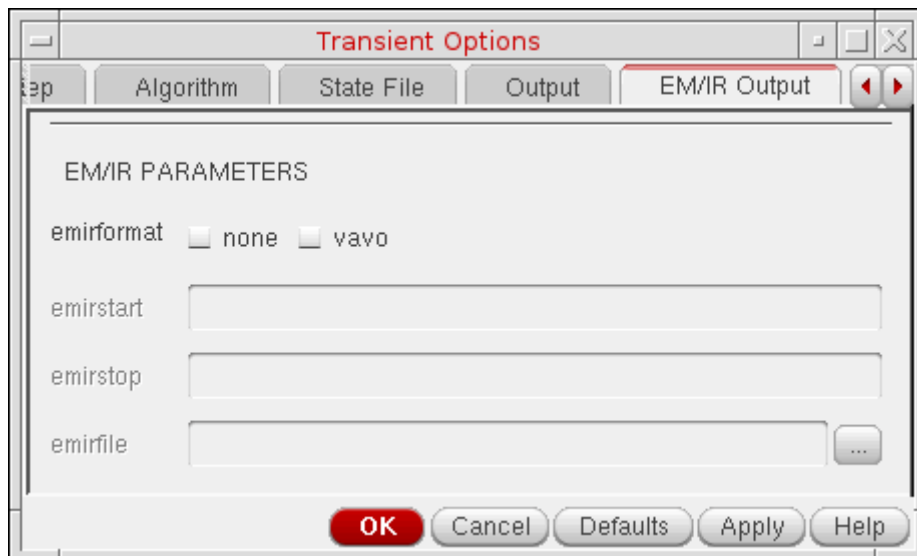
- skipstart=starttime s** time to start skipping output data.
- skipstop=stoptime s** time to stop skipping output data.
- skipcount=1** Save only one of every skipcount points.
- strobeperiod (s)** output strobe interval (in seconds of transient time).
- strobedelay=0 s** delay (phase shift) between the skipstart time and the first strobe point.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

- Save Final Op Pt.** generates the info statements for final operating point into the control file and related data into in psf file. If you do not want the results to be saved, select *No*. When this option is disabled, no final operating point data is generated.
- actimes=[...] s** Times when analyses specified in acname array are performed.
- acnames=[...]** Names of ac, noise, sp, stb, or xf analyses to be performed at each time point in the *actimes* array. The named small-signal analyses are not run separately, but only as part of the transient analysis.
- compression=yes** Turns on data compression.
- infotimes=[...] s** times when info analysis specified by *inforname* is performed.

EM/IR Output

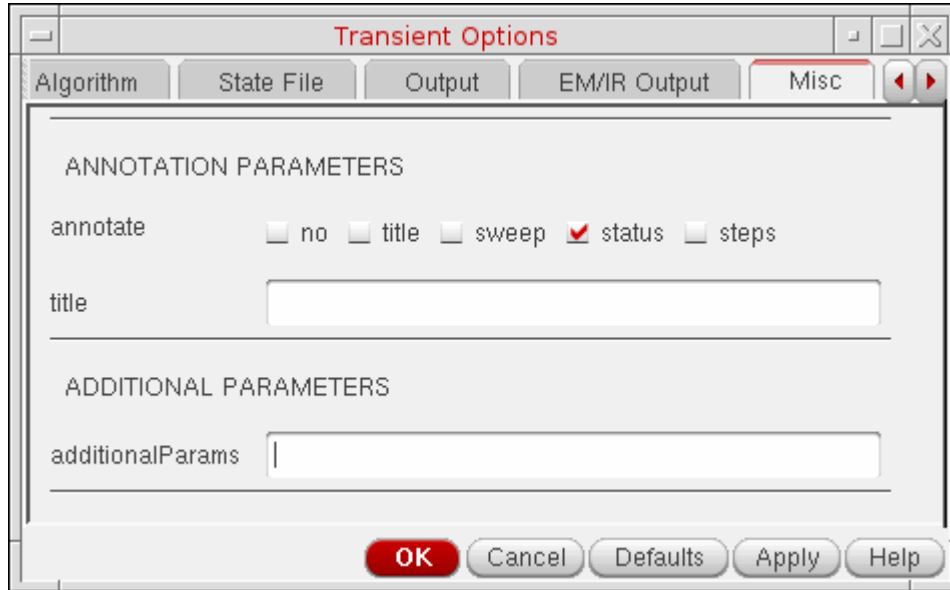


- emirformat=none** Format of the EM/IR database file. Possible values are none or vavo.
- emirstart (s)** EM/IR start time.
- emirstop (s)** EM/IR stop time.
- emirfile** Name of the EM/IR database file. Default is %A_emir_vavo.db. The file will be output to raw directory.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Misc



- annotate=sweep** Degree of annotation. Possible values are no, title, sweep, status, and steps.
- title** Analysis title.
- additionalParams** Specify any additional analysis parameters that you want to be used while netlisting.

Setting Up an HspiceD Analysis

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

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

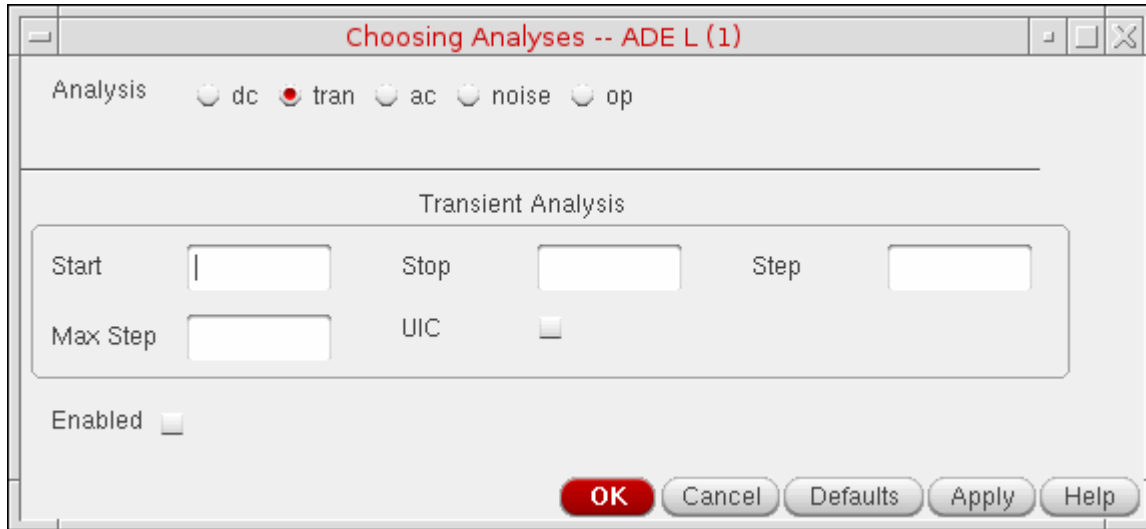
The screenshot shows a dialog box titled "Choosing Analyses -- ADE L (1)". At the top, under "Analysis", there are five radio buttons: "dc" (selected), "tran", "ac", "noise", and "op". Below this is a section titled "DC Analysis". It contains two main sub-sections. The first is "Sweep Variable", which has three radio buttons: "Temperature", "Design Variable", and "Source" (selected). To the right of these is a "Source Name" text box and a "Select Source" button. The second sub-section is "Sweep Range Type", which has a dropdown menu set to "Automatic". To the right of the dropdown are three text boxes labeled "Start", "Stop", and "Step Size". At the bottom left of the dialog is an "Enabled" checkbox. At the bottom right are five buttons: "OK" (highlighted in red), "Cancel", "Defaults", "Apply", and "Help".

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

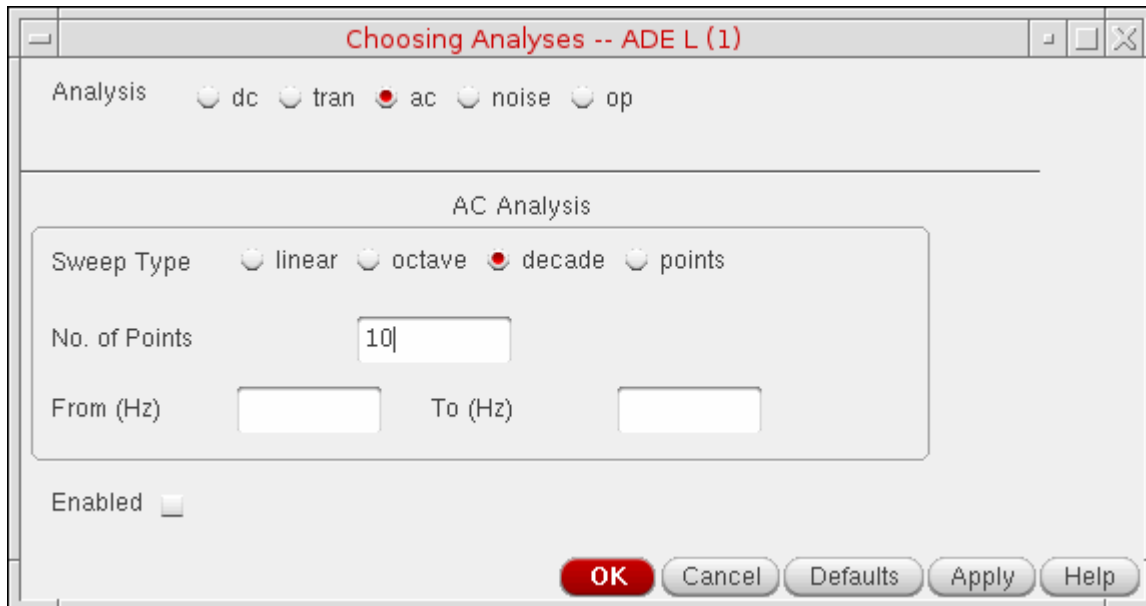
Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

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



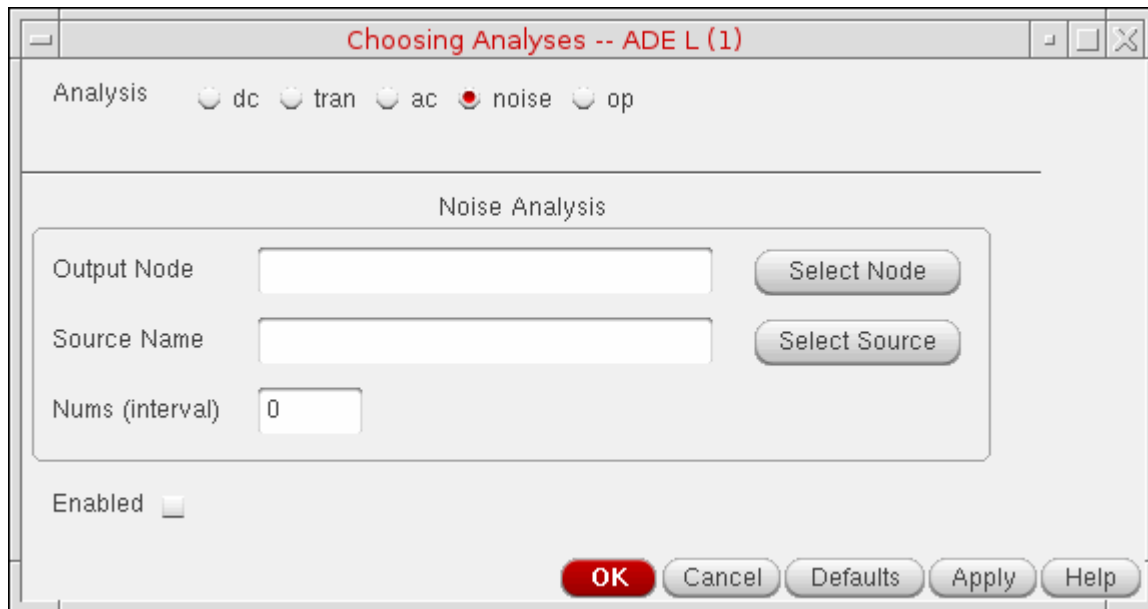
To run an AC analysis, click the *ac* radio button in the *Analysis* section of the *Choosing Analyses* form.



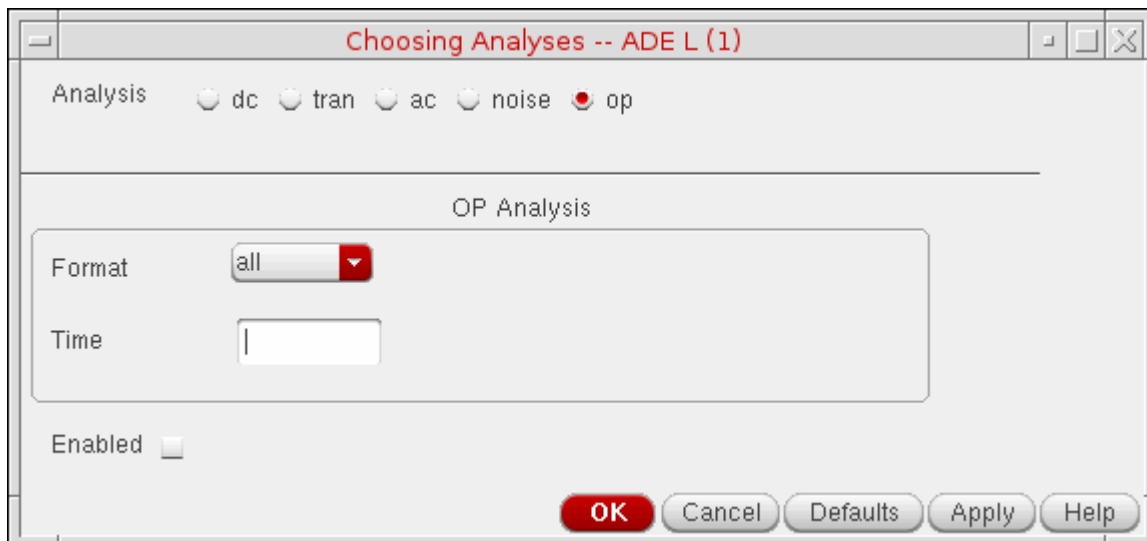
Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

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



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



Setting Up EM/IR Analysis

Starting with the IC616ISR7 release, Spectre® APS, and XPS have been enhanced to deliver a new EM/IR solution. The new dynamic power EM/IR and signal EM capability uses a new patent pending technology, and is designed to provide higher capacity and better performance compared to any existing EM/IR solutions. Within this flow, Spectre® APS can be used for high accuracy EM/IR analyses; while Spectre® XPS can be deployed for high performance and high capacity EM/IR simulation.

To setup the EM/IR analysis, choose *Setup – EM/IR Analysis*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

The *EM/IR Analysis Setup* form is displayed.

spectre0: EM/IR Analysis Setup

Analysis Solver Options

Type Dynamic Static

Net Selection

Net Name: Select Clear

IR Drop Analysis: max avg EM Current Analysis: max avg avgabs rms

Advanced IR Drop Analysis: Signal Net IR Drop Power Gate Add

Emirutil Setup

EM Tech File ...

Layer Map File ...

Summary Information

Options	Value
---------	-------

Import Export Delete Clear

Enable EMIR Analysis in Transient or DC Simulation

OK Cancel Apply Help

For help on different options in this form, see *EMIR Analysis* section in the *Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*.

Virtuoso Analog Design Environment L User Guide

Setting Up for an Analysis

Selecting Data to Save and Plot

This chapter shows you how to select data that you want to save or plot.

- [About the Saved and Plotted Sets of Outputs](#) on page 261
- [Opening the Setting Outputs Form](#) on page 262
- [Deciding which Outputs to Save](#) on page 263
- [Saving a List of Outputs](#) on page 275
- [Restoring a Saved List of Outputs](#) on page 276
- [Conditional Search for Results](#) on page 285
- [Conditional Search for Results](#) on page 285
- [Form Field Descriptions](#) on page 288

About the Saved and Plotted Sets of Outputs

The Virtuoso® Analog Design Environment keeps track of two sets of nets and terminals:

- The saved set, for which simulation data is written to disk
- The plotted set, which is automatically plotted after simulation in the Virtuoso Visualization and Analysis XL graph window.

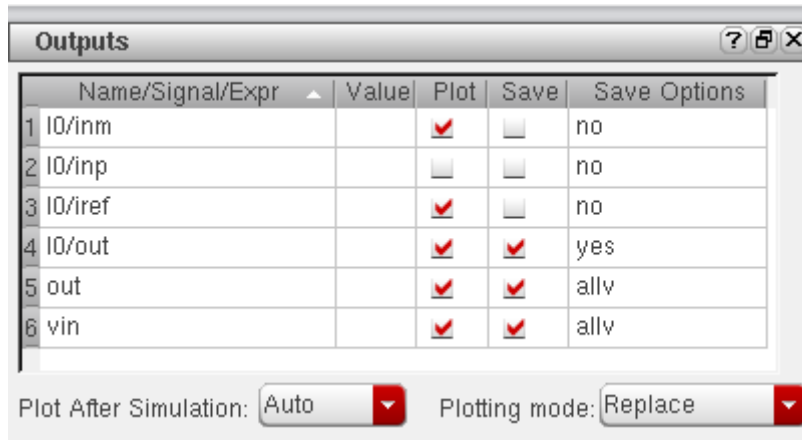
The plotted set can also contain [expressions](#).

The contents of all the sets of outputs are listed in the *Outputs* section of the Simulation window and in the [Setting Outputs form](#). Up to 999 outputs can be displayed in the Simulation window.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

The figure below shows how the signals will be plotted and saved after simulation.



You can enable/disable plotting/saving of multiple nets together. For this, select all the required nets, right-click and choose appropriate option from the popup menu. For example, if you have to enable plotting and saving of multiple nets, select them with mouse keeping the Ctrl key pressed. Right-click on any of them and select the *Enable Plot* and *Enable Save* options from the popup menu.

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

Opening the Setting Outputs Form

You set up the saved and plotted sets of outputs with the Setting Outputs form.

- In the Simulation window, choose *Outputs – Setup*, or from the Schematic window, choose *Setup – Outputs*.

The Setting Outputs form appears.

Selected Output		Table Of Outputs			
Name (opt.)	Signal	Name/Signal/Expr	Value	Plot	Save Options
	/F0/PLUS	1 F0/MINUS		yes	no
		2 F0/PLUS		yes	yes
	current	3 I0/PLUS		yes	no
	<input checked="" type="checkbox"/> Plotted <input checked="" type="checkbox"/> Saved	4 I0/MINUS		yes	no

For detailed information about the form, see [“Setting Outputs”](#) on page 291.

Deciding which Outputs to Save

Saving all the node voltages and terminal currents for a large design produces an enormous data set. The analog circuit design environment lets you save a selected set of voltages and currents from the schematic.

Once you select a set of output nodes and terminals, you can save their names to a file using the *Save State* command. You do not need to explicitly save nets and terminals that are used in expressions. All nets and nodes that are used in expressions are automatically set for saving and are also added to the Table of Outputs pane. Then, if you resimulate the design and want to view the same voltages and currents, you can load the set from the state file.

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

Saving All Voltages or Currents

To save all of the node voltages and terminal currents,

1. In the Simulation window, choose *Outputs – Save All*, or in the Schematic window, choose *Setup – Save All*.

A form appears that varies according to the simulator you use. For example, if you use the Spectre simulator, the Save Options form appears with the following format.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

For detailed information about the form, see [“Save Options and Keep Options”](#) on page 292.

The screenshot shows the 'Save Options' dialog box with the following settings:

- Select signals to output (save): none selected lvlpub lvl allpub all
- Select power signals to output (pwr): none total devices subckts all
- Set level of subcircuit to output (nestlvl): [Empty text box]
- Select device currents (currents): selected nonlinear all
- Set subcircuit probe level (subcktprobelvl): [Empty text box]
- Select AC terminal currents (useprobes): yes no
- Select AHDL variables (saveahdlvars): selected all
- Save model parameters info:
- Save elements info:
- Save output parameters info:
- Save primitives parameters info:
- Save subckt parameters info:
- Save asserts info:
- Output Format: sst2 psf psf with floats psfxl
- Use Fast Viewing Extensions:

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help.

With AMS, the *Save All* command is not on by default. Save (or probe) statements are placed in the `probe.tcl` file and not in the `amsControl.scs` file. For information about this, see the “Tcl-Based Debugging” appendix chapter of the *Virtuoso AMS Designer Simulator User Guide*.

2. Select the values you want to save and click *OK*.

Note: For AMSUltra, the options—*Save model parameters info*, *Save elements info*, and *Save output parameters info*—appear deselected by default. If you select them, the speed with which AMSUltra typically works may be compromised especially if you have a big design.

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

Saving Outputs for UltraSim Simulations

To view outputs for waveform data:

1. In the Simulation window, choose *Outputs – Save All*.

The *Keep Options* form appears.

The **Keep Options** dialog box contains the following settings:

- Analog Probe Output**: Analog Probe Output
- Select all node voltages
- Select all terminal currents
- Preserve All Nodes**: all port
- Hierarchical Depth**:
- Subckt Name**:
- Except**:
- Save model parameters info
- Save elements info
- Save output parameters info
- Output Format**: SST2 PSF PSFXL
- Use Fast Viewing Extensions
- Logic Probe Output
- Number of voltage threshold**:

Buttons at the bottom:

Analog Probe Output

To output waveform data for an analog probe, choose the appropriate settings.

- Select all node voltages** use to output all node voltages.
- Select all terminal currents** use to output all terminal currents. The Virtuoso UltraSim simulator outputs the first terminal current of each device.
- Preserve All Nodes** use to preserve either all or port RC node voltages. RC nodes are not reduced, allowing the nodes to be saved in simulation.

- ❑ **Hierarchical Depth** use to save and display more than one level of hierarchical results.
- ❑ **Subckt Name** use to indicate the subcircuit name for the analog probe. If a name is not entered into the `subckts` field, the simulator applies the analog probe to all blocks.
- ❑ **Except** specifies the nodes to be excluded from the analog probe. A node name, element name, or wildcard (*) can be used.

Save model parameters info specifies that input parameters for models of all components be saved.

Save elements info specifies that input parameters for instances of all components be saved.

Save output parameters info specifies that effective and temperature-dependent parameter values be saved.

Output Format specifies the format in which the results data must be saved. The results data can be saved in the following formats:

- ❑ **SST2** – Signal Scan Turbo 2 format. This format is supported for transient analyses only. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.
- ❑ **PSF** – Cadence parameter storage format (PSF)
- ❑ **PSFXL** – Cadence parameter storage XL format that provides higher performance for large circuit designs. For more details, refer to [Environment Variables for PSFXL Output Format](#) on page 295.



The PSFXL format is not compatible with versions of Virtuoso Visualization and Analysis prior to IC6.1.4 unless the CDS_PSF_XL_COMPAT environment variable is defined by the user. For information on this variable, refer to [Table 5-4](#) on page 296.

Use Fast Viewing Extensions enables the fast waveform viewing format for PSF and PSFXL output.

Using the PSF or PSFXL output format in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

To enable plotting of results data in this format using Virtuoso Visualization and Analysis XL, do the following:

a. Choose *Results – Printing/Plotting Options*.

The Setting Plotting Options form appears.

b. Select the *Fast Viewing Support* check box.

c. Click *OK*.

Logic Probe Output

To output waveform data for a logic probe, choose the appropriate settings (the Keep Options form expands to show the *Logic Probe Output* settings).

Note: The *Logic Probe Output* option is available only for SignalScan Turbo 2 (SST2) and fast signal database (FSDB) waveform output formats.

The screenshot shows the 'Logic Probe Output' dialog box. At the top, there is a checked checkbox labeled 'Logic Probe Output'. Below it is a dropdown menu for 'Number of voltage threshold' set to '1'. The dialog is divided into two sections. The first section contains 'Low Threshold' and 'High Threshold' text boxes. The second section contains 'Preserve All Nodes' with radio buttons for 'all' and 'port', and 'Hierarchical Depth' with a text box containing '1'. At the bottom, there are two text boxes for 'Subckt Name' and 'Except', each with a 'Select' button to its right.

- Number of voltage threshold** use to indicate the number of voltage thresholds for each logic probe.
- Low Threshold** specify for each logic probe the low threshold value which corresponds to the digital 0 value.
- High Threshold** specify for each logic probe the high threshold value which corresponds to the digital 1 value.
- Preserve All Nodes** use to preserve all or only port RC nodes from the RC reduction.
- Hierarchical Depth** use to save and display more than one level of hierarchical results.

- ❑ **Subckt Name** use to indicate the subcircuit name for the logic probe. If a name is not entered into the `subckts` field, the simulator applies the logic probe to all blocks
- ❑ **Except** specifies the nodes to be excluded from the logic probe. A node name, element name, or wildcard (*) can be used.

2. Click *OK*.

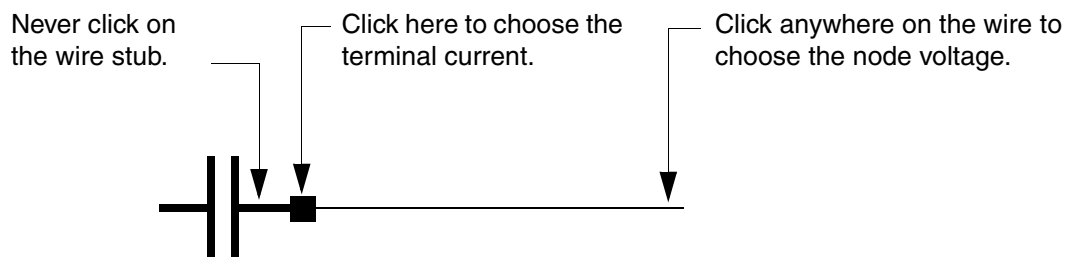
Saving Selected Voltages or Currents

To save the simulation data for particular nodes and terminals,

1. In the Simulation window, choose *Outputs – To Be Saved – Select On Design*.
2. In the Schematic window, choose one or more nodes or terminals.

The system circles pins when you choose a current and highlights wires when you choose a net.

- ❑ Click on an instance to choose all instance terminals.
- ❑ Click on the square pin symbols to choose currents.
- ❑ Click on wires to choose voltages.
- ❑ Click and drag to choose voltages by area.



3. Press the `ESC` key when you finish.

Saving or Plotting Selected Voltages or Currents for AMS Simulation

For AMS simulation, you can use the Schematic window to select the nodes or terminals for instances that have a schematic view.

You can use the Hierarchy Editor to select the nodes or terminals for instances:

- That have a Verilog, VerilogA, VerilogAMS, VHDL, VHDLAMS or SPICE-based text view

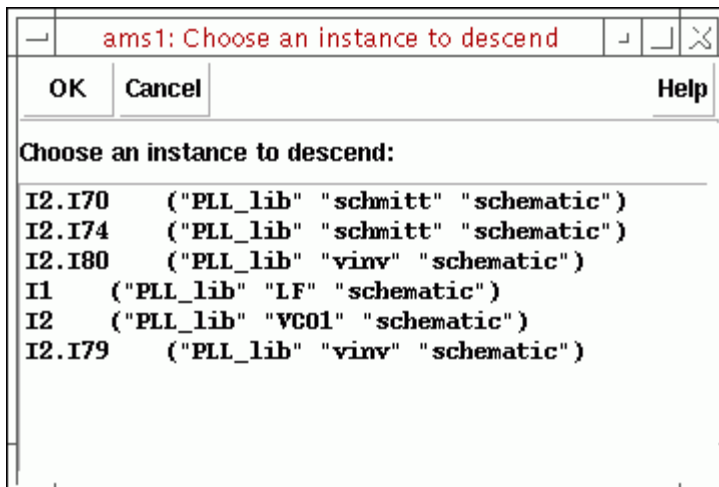
- That are bound to a Verilog, VerilogA, VerilogAMS, VHDL, VHDLAMS or SPICE-based text file using the `sourcefile` or `verilogfile` property in Hierarchy Editor.

To select nodes and terminals for an instance that has a schematic view,

1. In ADE, choose *Outputs – To Be Saved – Select From Schematic* or *Outputs – To Be Plotted – Select From Schematic*.

The Schematic window appears.

Note: If the top level design is a text design, the Choose an instance to descend form appears instead of the Schematic window. Select an instance in the list and click *OK* to open the schematic for the instance. You can also double-click on an instance to open the schematic.



2. In the Schematic window, choose one or more nodes or terminals.

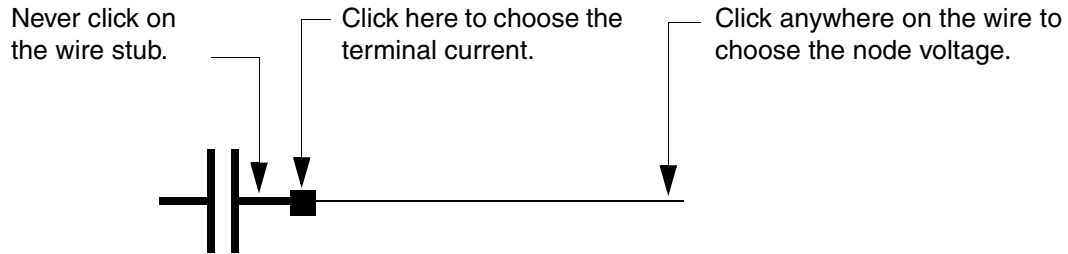
The system circles pins when you choose a current and highlights wires when you choose a net.

- Click on an instance to choose all instance terminals.
- Click on the square pin symbols to choose currents.
- Click on wires to choose voltages.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

- ❑ Click and drag to choose voltages by area.



3. Press the *Esc* key when you are done.

Your selections are added in the *Outputs* list in the simulation window.

To select nodes and terminals for an instance that has a text view,

1. In ADE, choose *Outputs – To Be Saved – Select From HED* or *Outputs – To Be Plotted – Select From HED*.

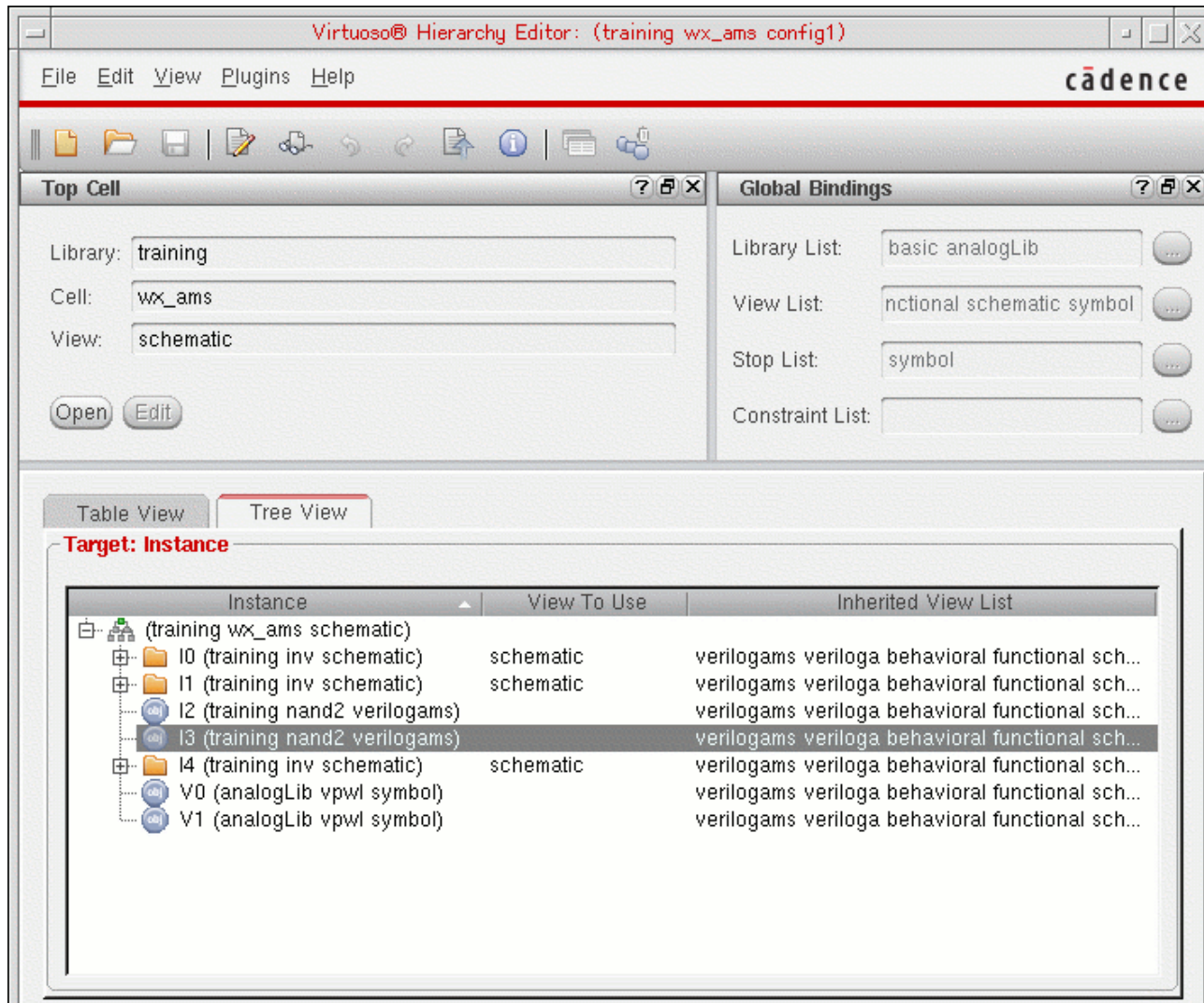
The Hierarchy Editor appears displaying the instances in the tree view.

Note: You can select nodes and terminals for instances only in the tree view in Hierarchy Editor. If you are in the table view in Hierarchy Editor, you cannot select nodes and terminals for instances.

Virtuoso Analog Design Environment L User Guide

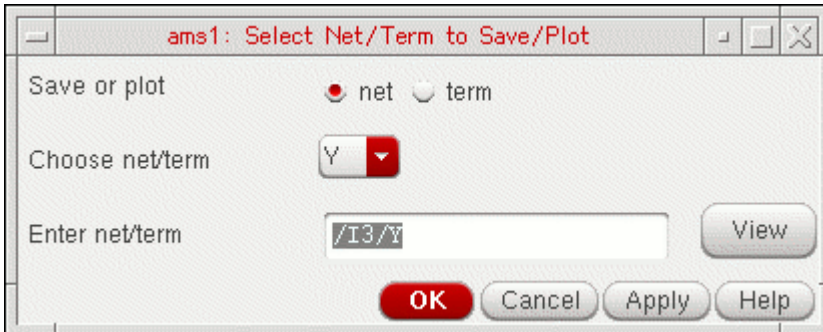
Selecting Data to Save and Plot

2. Click on the instance with the text view.



In the above figure, instance I3 is selected in Hierarchy Editor.

The Select Net/Term to Save/Plot form appears.



3. Do one of the following:
 - Select *net* to save or plot voltage.
 - Select *term* to save or plot current.
4. Select the node or terminal for the instance from the *Choose net/term* cyclic field.

You can click the *View* button to view the module for the instance in a text editor.

Note: The terminals of a primitive instance will not be displayed in the *Choose net/term* cyclic field because primitive instances do not have a text view. You can enter the terminal name in the *Enter net/term* field using the format:

I3/Y

Where /I3 is the full instance name and Y is the terminal name.

5. Do one of the following:
 - Click *Apply* to add the node or terminal in the *Outputs* list in the simulation window. Select another net or term for the current instance, or perform steps 2 to 4 to select a net or term for another instance.
 - Click *OK* to close the Select Net/Term to Save/Plot form and add the node or terminal in the *Outputs* list in the simulation window.

If you click *OK* or *Cancel* and then want to select nodes or terminals from Hierarchy Editor, you must again choose *Outputs – To Be Saved – Select From HED* or *Outputs – To Be Plotted – Select From HED* in ADE.

Adding a Node or Terminal to a Set

To add a node or terminal to the saved or plotted sets:

1. Choose *Outputs – To Be – Select On Design* commands in the Simulation window.

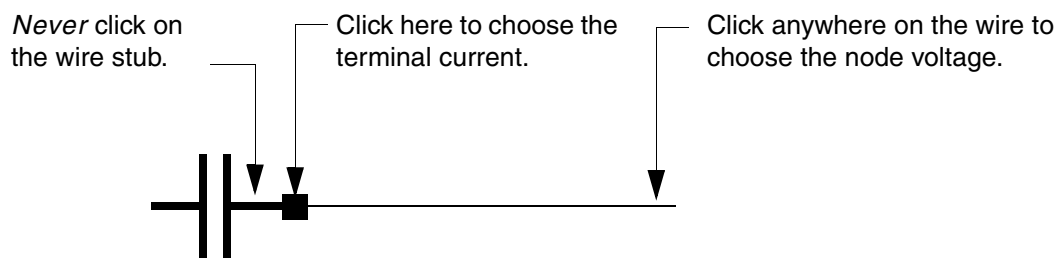
Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

2. In the Schematic window, choose one or more nodes or terminals.

The system circles pins when you choose a current and highlights wires when you choose a net.

- Click on the square pin symbols to choose currents.
- Click on wires to choose voltages.
- Click and drag to choose voltages by area.



3. Press the `ESC` key when you finish.

To select nodes and terminals in lower-level schematics to be plotted or saved,

1. In the Simulation window, choose *Outputs – To Be – Select On Design*.
2. In the Schematic window, choose *Design – Hierarchy – Descend Edit* and click on an instance.
3. Click *OK* in the form that appears.
4. In the Schematic window, choose one or more nodes or terminals.
5. Press the `ESC` key when you finish.

Adding a Saved Node to the Plot Set

To add an output in your saved set to the plotted set:

1. In the Simulation window, click in the *Outputs* list to choose the output.

To select more than one output, hold down the `Control` key while you click on the outputs, or click and drag.

To deselect a highlighted output, hold down the `Control` key while you click on it.

2. Choose the following:

■ *Outputs – To Be Plotted – Add To*

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

Note: You can use the *Outputs – To Be – Remove From* commands to remove highlighted outputs from a set.

Removing Nodes and Terminals from a Set

To remove a node or terminal from the saved or plotted set,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. Double-click on the node or terminal in the *Table Of Outputs* list box.



Name/Signal/Expr	Value	Plot	Save Options
I5/out1		yes	no
dacOut		yes	no
endOfConv		yes	no

3. Click to deselect the appropriate *Will Be* boxes.
4. Click *Change*.

Note: To remove a node from all three sets (delete it), highlight the node in the Simulation window and choose *Outputs – Delete*.

Saving a List of Outputs

You can save both

- The data for a set of outputs
- The list of saved and plotted outputs itself

The saved list includes output expressions.

To save the list of saved and plotted outputs,

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

The Saving State form appears.

2. Type a name for the saved simulation state.
3. Check that the *Outputs* box is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring a Saved List of Outputs

To restore a saved set of outputs,

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

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

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

The display shows the saved states for the cell and simulator combination.

3. Click on a state name.
4. Check that *Outputs* is selected and click *OK*.

Specifying Hierarchy Levels to Save Outputs

Saving the voltages and currents for a subcircuit having multiple levels of hierarchy can produce an enormous data set, resulting in degraded Spectre performance or cause out-of-memory problems. The analog design environment enables you to set the level of hierarchy to which you want to save the voltages, currents and power signals from the schematic.

To set the hierarchy:

1. Choose *Outputs – To Be Saved – Select By Subckt Inst* in the simulation window.

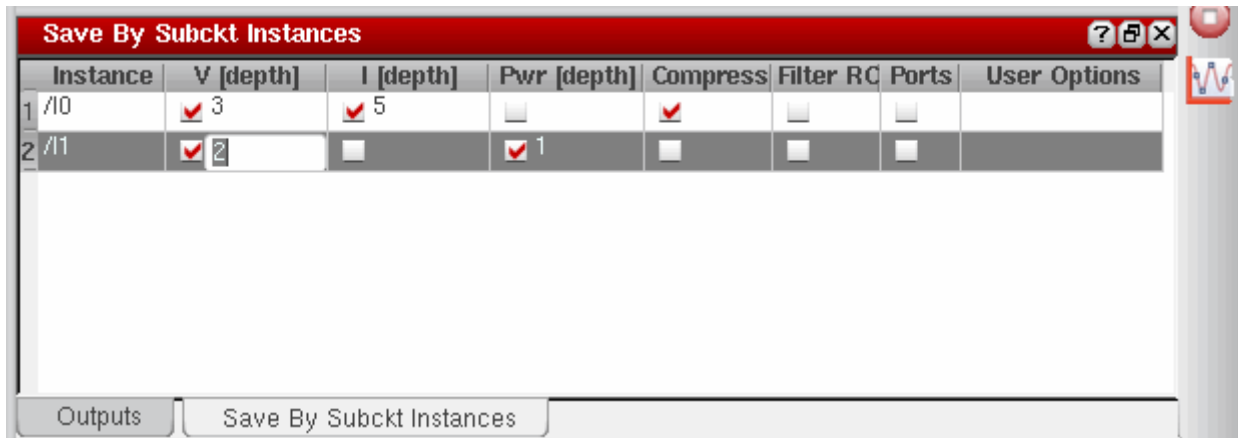
Alternatively, select *Add* from the context menu in the *Select By Subckt Instances* pane.

2. Select the subcircuits for which you want to specify the level of outputs to be saved.
3. Press `ESC`.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

The selected subcircuit instances are populated in the *Save By Subckt Instances* pane of the simulation window.



4. To specify the hierarchy:

- a. Select the check box for voltage, current, or power that you want to save.
- b. Specify the hierarchy level to which you want to save outputs next to the check box.

In the above figure, for the instance $/I0$, the voltage for three levels and the current for five levels of hierarchy is saved. Also, for the instance $/I1$, the voltage for two levels and the power for one level of hierarchy is saved.

Note: If you do not specify the hierarchy level to save, the voltage, current, and power for all the levels in the subcircuit are saved.

5. Select *Compress* if you want to reduce the size of the output file. When this options is selected, the Spectre simulator saves the data for a signal only when the value of that signal changes.
6. Select *Filter RC* to filter out the nodes that are connected only to parasitic elements from the output signal list.
7. Select *Ports* if you want to save the output port information for the specified subcircuits.
8. Specify the additional save options in the *User Options* field.

If you set a value for any of the fields in this pane and also specify a command in the *User Options* field, the valid value that was set last is considered during the simulation run. However, the netlist contains all the values set for the field.

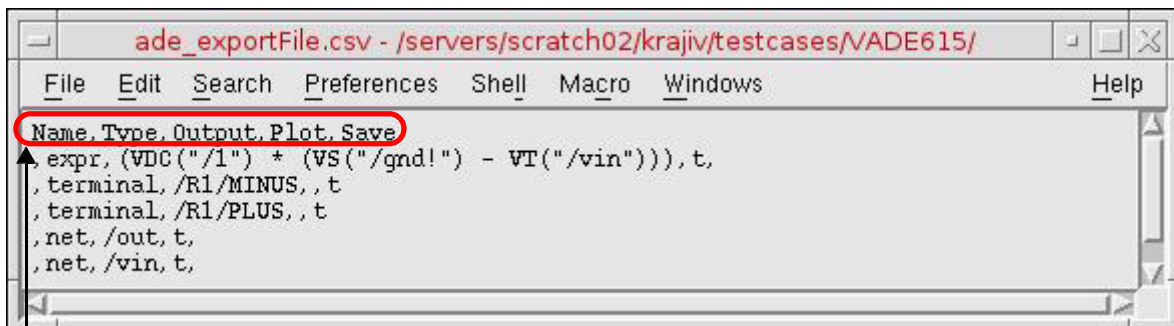
Importing and Exporting Outputs in ADE L Environment

Working on a large number of signals and expressions, which includes specifying values and setting plot and save options for signals and expressions, in the *Outputs* window in ADE L is difficult. Therefore, to manage the signals and expressions better, you can export the outputs from the *Outputs* window to a CSV file. You can then edit or add output values to the file in a text editor and import the file back to ADE L.

Note: You can also create the CSV file containing the required output values and import it to ADE L. For this, you need to ensure that the CSV file follows the prescribed format.

CSV File Format

The format of the CSV file is as shown in the following figure:



CSV File Format

The following table lists the output components in the order required by the CSV file format and provides a brief description of each component.

Table 5-1 Output Components in the CSV File

Component	Description
<i>Name</i>	Indicates the name of the expression.
	Note: If the output is for a signal, specify <code>nil</code> for this component.
<i>Type</i>	Indicates the type of output. The possible values are: <ul style="list-style-type: none"> ■ <code>net</code> — Indicates a node voltage ■ <code>terminal</code> — indicates a current ■ <code>expr</code> — indicates an expression

Table 5-1 Output Components in the CSV File

<i>Output</i>	Indicates the name of the net, terminal or the expression.
<i>Plot</i>	Indicates if a output is to be plotted or not. The possible values are: <ul style="list-style-type: none">■ <code>t</code> — Indicates output is to be plotted■ <code>nil</code> — Indicates output is not to be plotted
<i>Save</i>	Indicates if a output is to be saved or not. The possible values are: <ul style="list-style-type: none">■ <code>t</code> — Indicates output is to be saved■ <code>nil</code> — Indicates output is not to be saved

Exporting Outputs to a CSV File

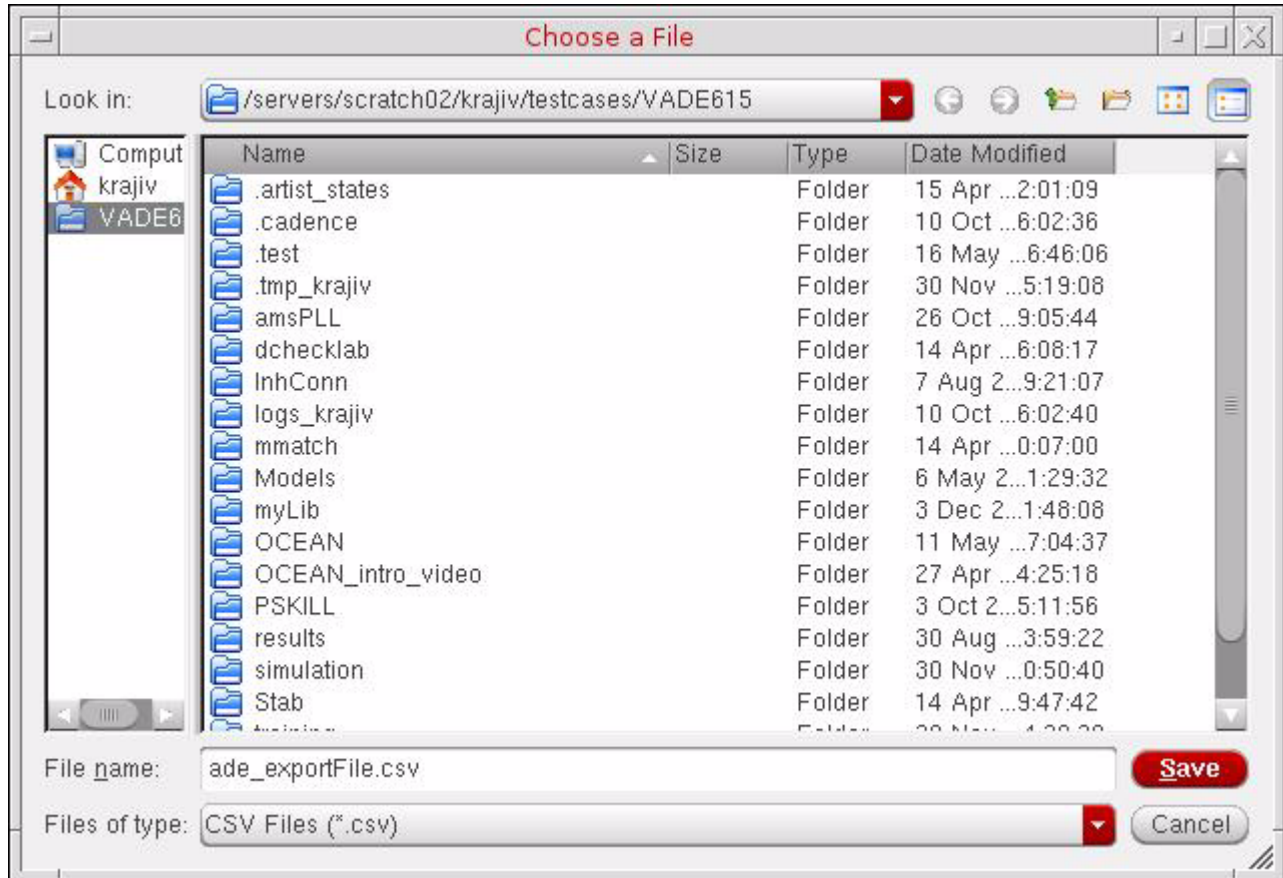
To export the outputs to a CSV file, do the following:

1. Choose *Outputs – Export*.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

The Choose a File form appears.



2. In the *Files of Type* list, select *CSV Files (*.csv)*.
3. In the *File name* field, specify a name for the CSV file to which the output will be saved.
4. Click *Save*.

The following figure displays the *Outputs* window in ADE L.

Outputs					
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	(VDC("/1") * (VS("/gnd!"))...		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2	R1/MINUS		<input type="checkbox"/>	<input checked="" type="checkbox"/>	yes
3	R1/PLUS		<input type="checkbox"/>	<input checked="" type="checkbox"/>	yes
4	out		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv
5	vin		<input checked="" type="checkbox"/>	<input type="checkbox"/>	allv

After being exported to a CSV file, the outputs appear as shown in the following figure:

Note: You can open the CSV file in any text editor.



For information about the contents of the CSV file, see [Table 5-1](#) on page 278.

Importing Output to ADE L

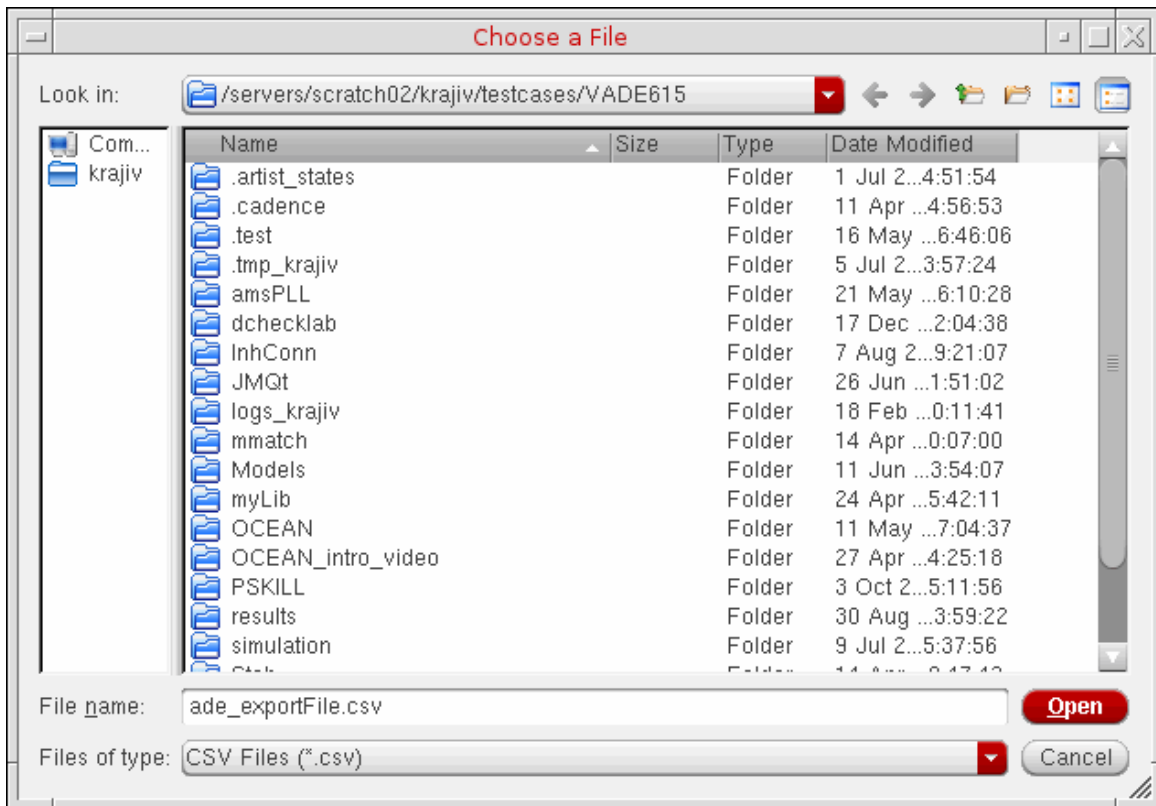
To import output values from a CSV file to the *Outputs* window in ADE L, do the following:

1. Choose *Outputs – Import*.

The Choose a File form appears.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot



2. In the *File name* field, specify the name of the CSV file that you want to import.

Note: Ensure that the CSV file being imported contains data in the prescribed format. If a value in the CSV file is not in the prescribed format, an error message is displayed in the CIW. For more information about the format, see [CSV File Format](#) on page 278.

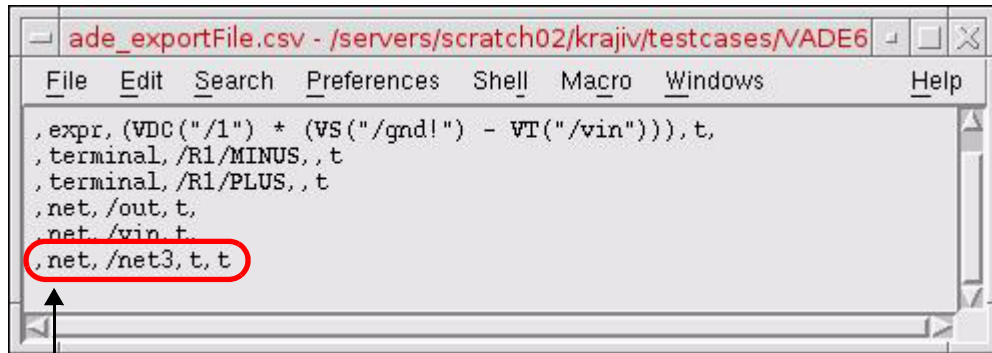
3. Click *Open*.

Note: Only those outputs that are not defined in the *Outputs* window are loaded from the CSV file being imported. Therefore, to ensure that the changes made to an exported output are visible in the *Outputs* window, delete that output from the *Outputs* window before importing the CSV file.

For example, you add a net voltage signal for `net3` that needs to be saved and plotted, as shown in the following figure.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

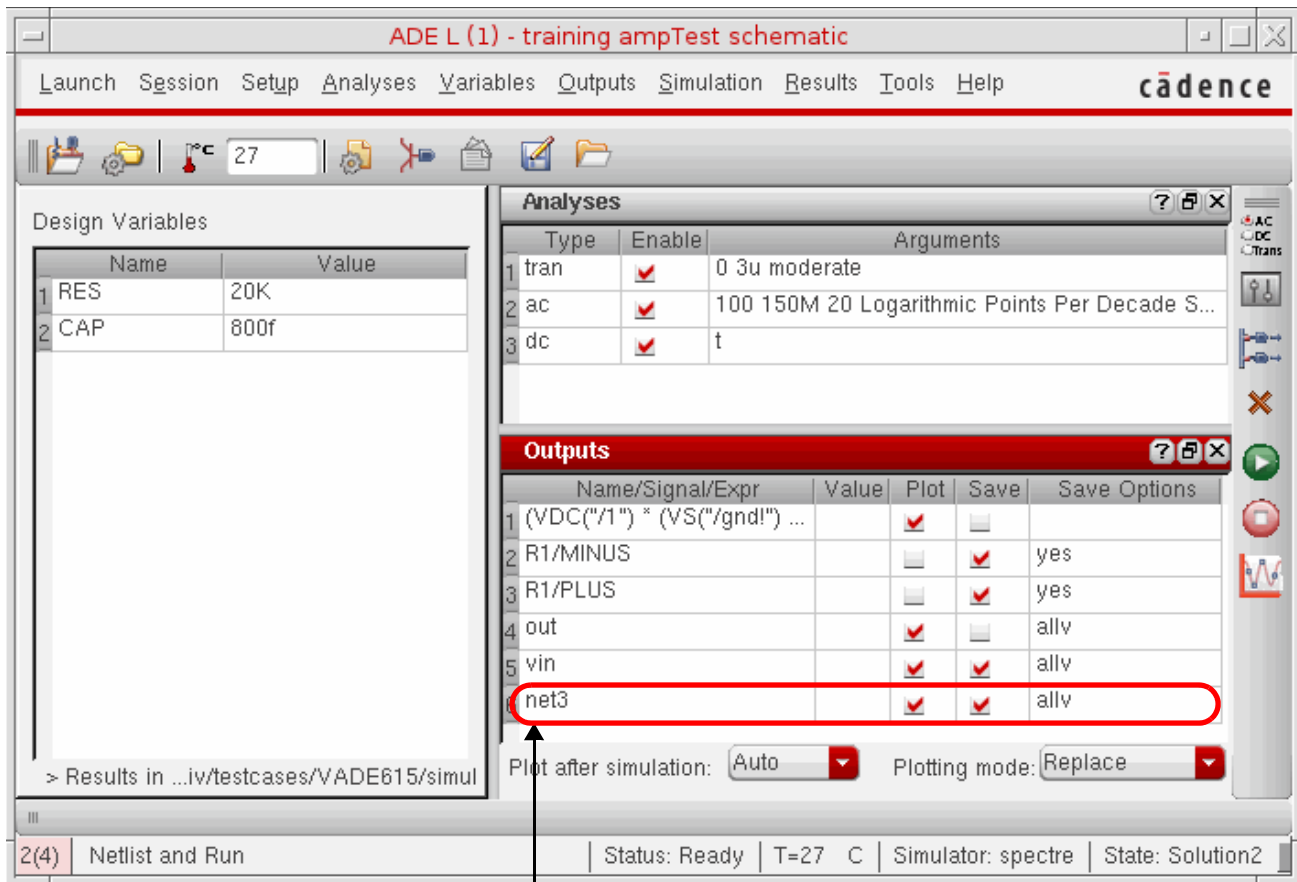


Newly added net voltage signal

After the file is imported back to ADE L, the new output value is displayed in the *Outputs* window in ADE L, as shown below:

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot



Newly added net voltage signal

Important Points to Note:

- For backward compatibility, ADE L supports importing outputs from a file that was earlier saved in the text format.
- To save outputs in the text format similar to the previous releases, set the `outputsImportExportVersion` environment variable to a value less than or equal to 1.0.
- If no value is specified for any column, it is considered as `null`.
- It is mandatory to specify at least the Name column or the Signal and Expression columns in the CSV file.
- By default, only the Name, Type, Output, Plot and Save columns are saved. However, if you have specified values in other columns as well, all other columns, such as Signal, are also saved in the CSV file.

- Outputs that were earlier exported from the ADE XL environment in a txt file can be imported in ADE L.

Note: The *Outputs – Import* and *Outputs – Export* options are available for Spectre, AMS and UltraSim simulation.

Conditional Search for Results

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

1. Run a simulation.

Note: You must run a DC operating-point analysis to use the circuit conditions capability.

2. Choose *Results – Select* and indicate the results that you wish to search.
3. Choose *Results – Circuit Conditions* from the Simulation window.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

The *Circuit Conditions* form appears:

#	Enable	Color	Component	Lower Bound	Parameter	Upper Bound	and/or
1	yes	magenta	capacitor		cap		none

For detailed information about the form, see [“Circuit Conditions”](#) on page 288.

4. Choose device operating conditions.

You can choose to view components in the saturation (for BJT devices), linear (for MOS devices), or breakdown region.

Note: The appropriate model parameters must be set for the simulator to calculate these conditions. These features might not be available for simulators other than spectre.

5. Set up *User Defined Conditions*.

You use the cyclic and type-in fields to create the custom conditions you want to search for.

6. View the results of the conditions you chose by doing the following.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

- ❑ Click *Place* to highlight the instances that meet the specified conditions on the schematic.
 - ❑ Click *Print* to print the values of instances that meet the specified conditions in a print window.
 - ❑ Click *Clear* to de-highlight all the instances on the schematic.
7. Clicking on the *Options* button will bring up a form where you can specify filter and sort conditions.

The screenshot shows a dialog box titled "Circuit Conditions Options Form". It contains two sections for configuring filters and sorting options.

Filter out Components by Model Name (checkbox is unchecked):

- Component: dropdown menu showing "capacitor"
- Model Name: empty text input field
- Buttons: "Add" and "Delete"
- Empty list box below

Sort Components by Parameter Value (checkbox is unchecked):

- Component: dropdown menu showing "capacitor"
- Param Name: dropdown menu showing "cap"
- Buttons: "Add" and "Delete"
- Empty list box below

At the bottom are "OK", "Cancel", and "Help" buttons.

In the *Filter out Components by Model Name* section, you can enter filters using the cyclic field displaying all the component types and the text entry field to type in model names. After you have selected the component type and entered a model name, press *Add* to add the filter to list of filters. You can select one or more filters in the list and then click *Delete* to delete the filters. The filters are active only when the *Filter out Components by Model Name* check

box is selected. When the filters are active, any component that matches a filter will be filtered out when you click the *Place* button or the *Print* button.

The next section is *Sort components by Parameter Value*. Users can use the two cyclic fields to enter sorting criteria for a component type. When this section is active (Boolean is on) the output from *Print* for user defined conditions will be sorted according to the sort variable for given component type.

Form Field Descriptions

Circuit Conditions

Device Operating Conditions

These checkboxes let you highlight components in saturation and in breakdown. When the *Annotate Place* button is pressed, components in breakdown, saturation, or both are highlighted on the schematic with a colored box. The color of the box is chosen by the color cyclic field next to each field.

Saturation BJT or Linear MOS

An instance is highlighted for

- saturation region of BJT if the operating point parameter region=3
- or linear/triode region of MOS/bsim if the operating point parameter region=1

Breakdown

For Spectre breakdown, an instance is highlighted if

- For BJT: If the operating point parameter region=4

Note: For the simulator to calculate breakdown or saturation, the appropriate model parameters need to be set.

User-Defined Conditions

Enable uses the cyclic field to select yes or no to enable or disable a condition.

Color shows the color with which you want to highlight instances meeting a condition.

Component shows the type of component for which you want to create conditions.

Lower Bound specifies the lower boundary of a parameter's value.

Upper Bound specifies the upper boundary of a parameter's value.

Parameter is an operating-point parameter you choose from the cyclic field. The *Lower Bound* and *Upper Bound* values apply to the selected parameter.

and/or sets Boolean arguments to a condition. When *and* is used, both conditions must be met for an instance to be highlighted. When *or* is used, either condition must be met for an instance to be highlighted. Both operators have the same precedences.

Add adds another compound condition to the existing entries in the table. When this button is clicked, a new row is added to the bottom of the table so that a designer can specify another search condition.

Delete removes a condition from the table. When this button is clicked the selected entries in the table are removed. You select entries by clicking on a row in the *User Defined Conditions* box.

Change lets you modify a user-defined condition. You must select the condition before modifying it.

Clear lets you clear all the entries from the *User Defined Conditions* box.

Results

Annotate Place uses the conditions specified in the form above to search through the simulation's DC operating point data. The data matching the conditions specified in the table are filtered and the instances are highlighted. Components are highlighted with boxes. The Circuit Conditions form remains open until you click either the *OK* or *Cancel* button. In hierarchical designs, if the component that needs to be highlighted is within a block, the block is highlighted. When you push into the highlighted block so that the component is displayed, the component is then highlighted. Whenever *Annotate* is selected, the currently highlighted instances are cleared and the new ones redrawn. If a component meets more than one condition, it is highlighted with a third color and a message prints in the CIW.

Annotate Clear, when selected, clears all of the highlighted instances from the Schematic window.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Print, when selected, prints the results to the print window, which displays the results as a table. The results are defined as the components that match the conditions specified in the Circuit Conditions form with their state.

Setting Outputs

Name (opt.) is an optional name for the signal, which appears in the *Table Of Outputs* list box and in the graph window.

Expression is the calculator expression to plot or save.

Calculator buttons are displayed only while no net or terminal is selected in the *Table Of Outputs* list box.

Open opens the calculator.

Get Expression copies the expression in the calculator buffer into the *Expression* field.

Close dismisses the calculator window.

Will Be changes depending on whether an expression or a signal is selected.

Plotted/Evaluated plots or prints the value of the expression after each simulation.

Add creates the output you set up in the *Selected Output* area.

Delete removes the highlighted output. Click in the *Table Of Outputs* list box to highlight an output.

Change updates the highlighted output with the new settings in the *Selected Output* area.

Next moves the highlight to the next signal or expression in the *Table Of Outputs* list box. This allows you to make changes to consecutive entries in the *Table Of Outputs* list box without clicking on each entry.

New Expression clears the *Selected Output* area so you can enter a new output.

Save Options and Keep Options

The title of this form and the options displayed vary depending on the simulator you use. If you are using direct simulation, see the following section for information.

Save Options Form for Direct Simulation (Spectre)

For detailed information about the fields in the following table, follow the cross-references. All of the cross-references are to sections in the *Specifying Output Options* chapter of the *Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*.

Table 5-2 Fields in the Save Options Form

Field	For more information, see
Select signals to output (save)	<i>The save Parameter Options</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Select power signals to output (pwr)	<i>Saving Power</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Set level of subcircuit to output (nestlvl)	<i>Saving Groups of Signals</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Select device currents (currents)	<i>Saving Groups of Currents</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Set subcircuit probe level (subcktprobelvl)	<i>Saving Subcircuit Terminal Currents</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Select AC terminal currents (useprobes)	<i>Setting Multiple Current Probes</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .
Select AHDL variables (saveahdlvars)	<i>Saving All AHDL Variables</i> in the <i>Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide</i> .



The all *buttons* (Select signals to output (save)) are global buttons but can be locally overridden while specifying options for a particular analysis.

The other fields in the Save Options form are described below.

Table 5-3 Other Fields in Save Options Form

Field	For more information, see
Save model parameters info	Specifies that input parameters for models of all components be saved.
Save elements info	Specifies that input parameters for instances of all components be saved.
Save output parameters info	Specifies that effective and temperature-dependent parameter values be saved.
Save primitives parameters info	Specifies that model parameters, oppoint parameters, output parameters, instance parameters, region parameters, and terminal names of primitives be saved. No parameter values are printed.
Save asserts info	Prints the device checks in the <i>Violations Display</i> form if the device checks are defined using the asserts file.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Field	For more information, see
Output Format	<p>Specifies the format in which the results data must be saved. The results data can be saved in the following formats:</p> <ul style="list-style-type: none">■ <code>sst2</code> – Signal Scan Turbo 2 (SST2) format. This format is supported for transient analyses only. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.■ <code>psf</code> – Cadence parameter storage format (PSF)■ <code>psf with floats</code> – Cadence lowered precision parameter storage format. <p>Note: The <code>psf with floats</code> format stores data using floats vs doubles. This results in a database that is about 50% of the size of normal PSF output, thereby providing capacity and performance benefits. For some simulators, using this format is not recommended as the use of floats reduces the number of significant digits and can limit the accuracy of measurements. Specifically, SpectreRF users are recommended not to use this format as the noise floor numbers can be misleading.</p> <ul style="list-style-type: none">■ <code>psfxl</code> – Cadence parameter storage XL format (PSF XL). PSF XL provides higher performance for large circuit designs. <p>Note: When <code>psfxl</code> is set as the output format for transient analysis, the behavior of the waveform writing module is controlled by a set of environment variables. For more details, refer to Environment Variables for PSFXL Output Format.</p>



The PSFXL format is not compatible with versions of Virtuoso Visualization and Analysis prior to IC6.1.4 unless the `CDS_PSFXL_COMPAT` environment variable is defined by the user. For information on this variable, refer to [Table 5-4](#) on page 296.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Field	For more information, see
Use Fast Viewing Extensions	<p>Enables the fast waveform viewing format for PSF and PSFXL output.</p> <p>Using the PSF or PSFXL output format in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.</p> <p>To enable plotting of results data in this format using Virtuoso Visualization and Analysis XL, do the following:</p> <ol style="list-style-type: none">1. Choose <i>Results – Printing/Plotting Options</i>. The Setting Plotting Options form appears.2. Select the <i>Fast Viewing Support</i> check box.3. Click <i>OK</i>.

Environment Variables for PSFXL Output Format

When `psfxl` is set as the output format for transient analysis, the behavior of the waveform writer module is controlled by a set of environment variables defined in your shell session. These variables are listed in [Table 5-4](#) on page 296. You can override the default values of these variables by setting them in your `.cshrc` or in your shell session.

Important

The default values of the variables given in [Table 5-4](#) on page 296 have been set to provide optimal performance for most end-user designs. Users should not change the default values unless they experience specific performance degradation issues.

Virtuoso Analog Design Environment L User Guide
Selecting Data to Save and Plot

Table 5-4 Environment Variables for PSFXL Output Format

Variables	Description
CDS_PSFXL_FLUSH_INTERVAL	<p>Specifies the number of CPU seconds elapsed between data flushes to disk. This value is increased by 2% after every flush up to the value of variable <u>CDS_PSFXL_MAX_FLUSH_INTERVAL</u>.</p> <p>Default value is 300 seconds.</p> <p>Larger flush intervals require more simulation data to be kept in memory, but results in more efficient reading of data by the waveform and post-processing tools. Smaller intervals reduce the memory usage and runtime overhead of the simulator.</p>
CDS_PSFXL_INIT_FLUSH_INTERVAL	<p>Specifies the rate (in terms of number of CPU seconds elapsed) at which data is initially flushed to the disk. This value is progressively increased by 50% after each flush until it approaches the value of variable <u>CDS_PSFXL_FLUSH_INTERVAL</u>. After this point, the flush rate is determined by the behavior of the variable <u>CDS_PSFXL_FLUSH_INTERVAL</u>.</p> <p>Default value is 300 seconds.</p>
CDS_PSFXL_MAX_FLUSH_INTERVAL	<p>Specifies the maximum number of CPU seconds elapsed between data flushes. This sets an upper limit on how large the progressively increasing flush period defined by <u>CDS_PSFXL_FLUSH_INTERVAL</u> can grow.</p> <p>Default value is 1200 seconds.</p> <p>Larger flush intervals require more simulation data to be kept in memory, but results in more efficient reading of data by the waveform and post-processing tools. Smaller intervals reduce the memory usage and runtime overhead of the simulator.</p>

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Variables	Description
CDS_PSFXL_MAX_BUFSIZE	<p>Specifies how much in-process memory to be allocated for unwritten data values.</p> <p>Default value is 256 MB for 32-bit platform and 384 MB for 64-bit platform.</p> <p>Larger buffer size increase the performance of the viewer or reader module. Smaller buffers reduce the memory usage and runtime overhead of the simulator.</p>
CDS_PSFXL_COMPAT	<p>Instructs the simulator to write both PSF and PSFXL data simultaneously, thereby preserving backward compatibility for users of Virtuoso Visualization and Analysis prior to IC6.1.4.</p> <p>Default value 0 disables this compatibility mode. Setting the value to 1 preserves backward compatibility, but requires twice as much disk space to be consumed, and leads to slightly higher runtime overhead on the simulator.</p>
CDS_ENABLE_NATIVE_RTSE	<p>Enables or disables support to save native RTSE data in PSFXL format. Enabling this support improves the performance of ultrasim simulation. Earlier, it was possible to save only first 1000 signals from ultrasim simulation results in RTSE format. With this feature, all the signals from ultrasim simulation can be saved in RTSE format. However, this results in a memory overhead similar to that in the case of spectre simulation with support for native RTSE.</p> <p>Default value 0 disables the native RTSE format and enables the legacy RTSE format. Set this variable to 1 to enable the native RTSE format.</p> <p>Note: Versions of Virtuoso Visualization and Analysis in IC6.1.4 or earlier releases cannot read the native RTSE format. These versions of Virtuoso Visualization and Analysis read the actual waveform data, but the performance is not good as compared to reading the native RTSE format.</p>

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Variables	Description
CDS_PSFXL_FLOATY	<p>Sets the default precision of waveform Y-values. This has no effect on the X- or time sweep values that are always saved as double precision. By setting this variable, you can reduce the size of waveform database by approximately 50% for spectre and aps simulators and approximately 25% for ultrasim. This variable has negligible impact on simulations and waveform viewing.</p> <p>Default value 0 sets the default precision of waveform Y-values to double precision float. Set this variable to 1 to set the default precision to single precision float.</p>
CDS_PSFXL_SERVER	<p>Enables or disables reading of live simulation data while the simulation is running. Live simulation data includes both of the following:</p> <ul style="list-style-type: none">■ the simulation data that has been flushed to the disk■ data that is still in the simulation process memory and not yet flushed to the disk <p>By enabling the reading of live simulation data, you can improve the performance of waveform viewing because it is not required to read the entire waveform data every time the data is flushed by the simulator. This has negligible impact on simulation unless excess live simulation data reads are requested by you. Therefore, to maximize the performance of the simulator, it is recommended to minimize the amount of live data requests for plotting while the simulation is in progress.</p> <p>Default value 0 disables reading of live simulation data. Set this variable to a non-zero integer value to enable reading.</p>

PSFXL Environment Variables for Site Administrators

In addition to the environment variables given in [Table 5-4](#) on page 296, there are two more variables that control how the live simulation data in PSFXL format is read. These variables define the port range for server connections and are intended for site administrators who want

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

to enforce network service restrictions across their IT domain. The site administrators can set these variables in their shell session. Any value set for these variables in users local home is ignored.

Table 5-5 Environment Variables for Site Administrators

CDS_PSFXL_PORT_LOW	Lowest TCP/IP port number allowed for server connections. The server will not search for any open ports below this value. Default value: 32768
CDS_PSFXL_PORT_HIGH	Highest TCP/IP port number allowed for server connections. The server will not search for any open ports above this value. Default value: 65536

 **Important**

Setting both the variables given in the previous table to 0 disables communications with the simulator for reading live PSFXL data. It is recommended that site administrators use the default values set for these variables. Making the port range too restrictive might limit the number of live simulations that can be made available to the end user. For example, if the range is set to 38000 to 38009, then only ten simulation processes can be used simultaneously by the users.

Virtuoso Analog Design Environment L User Guide

Selecting Data to Save and Plot

Running a Simulation

This chapter describes how to run a simulation that you have set up.

- [Prerequisites to Simulation](#) on page 301
- [Setting Simulator Options](#) on page 302
- [About the OSS-based AMS Netlister](#) on page 353
- [Choosing the AMS Netlister](#) on page 365
- [Starting a Simulation](#) on page 378
- [Interrupting or Stopping a Simulation](#) on page 380
- [Saving Simulator Option Settings](#) on page 381
- [Restoring Saved Settings](#) on page 381
- [Viewing the Simulation Output](#) on page 382
- [Running a Parametric Analysis](#) on page 402
- [Device Checking](#) on page 403

Prerequisites to Simulation

Before running a simulation, you need to

- [Start](#) the Virtuoso® Analog Design Environment and set up the simulation environment
- Specify [analyses](#)
- Specify which outputs to [save](#)

After you finish simulating, you can [plot results](#).

Setting Simulator Options

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

- The *Simulation – Options – Analog* command lets you set simulator options and variables that apply to all analyses.
- For the spectre, and some other simulator interfaces, the *Options* buttons in each Choosing Analyses form let you set options that apply to the specific analysis.

Each simulator has a different set of options.

Spectre Options



For help on Spectre options, refer to the *Immediate Set Options (options)* section in the *Analysis Statements* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

Specifying Performance and Parasitic Reduction Options

This section describes how you can specify the simulation performance and parasitic reduction options for the Spectre simulator and the AMS Designer simulator.

- For information about specifying these options for the Spectre simulator, see [Specifying Performance and Parasitic Reduction Options for the Spectre Simulator](#) on page 304
- For information about specifying these options when *Spectre* is set as the analog solver for the AMS Designer simulator, see [Specifying AMS Spectre High Performance/Parasitic Reduction Options](#) on page 309

Specifying Performance and Parasitic Reduction Options for the Spectre Simulator

With the Spectre simulator, you can run simulation using the baseline Spectre version. You can also enable parasitic reduction to reduce the number of parasitic resistors and capacitors extracted from the layout by eliminating internal nodes that are not connected to any devices or I/O ports.

For more information about the baseline Spectre version and parasitic reduction, see the *Virtuoso Spectre Circuit Simulator User Guide*.

Note: Spectre Turbo mode and parasitic reduction are supported in MMSIM 7.0 and later releases. APS is supported in MMSIM 7.1 and later releases.



When a Spectre simulation is running in the interactive mode, do not restore a simulation setup in which the Spectre Turbo mode or parasitic reduction is enabled. Restoring such a simulation setup terminates the simulation run. For more information about restoring the simulation setup, see [Restoring the Simulation Setup](#) on page 100.

To specify performance and parasitic reduction options for the Spectre simulator, do the following:

1. In the Simulation window, choose *Setup – High-Performance Simulation*.

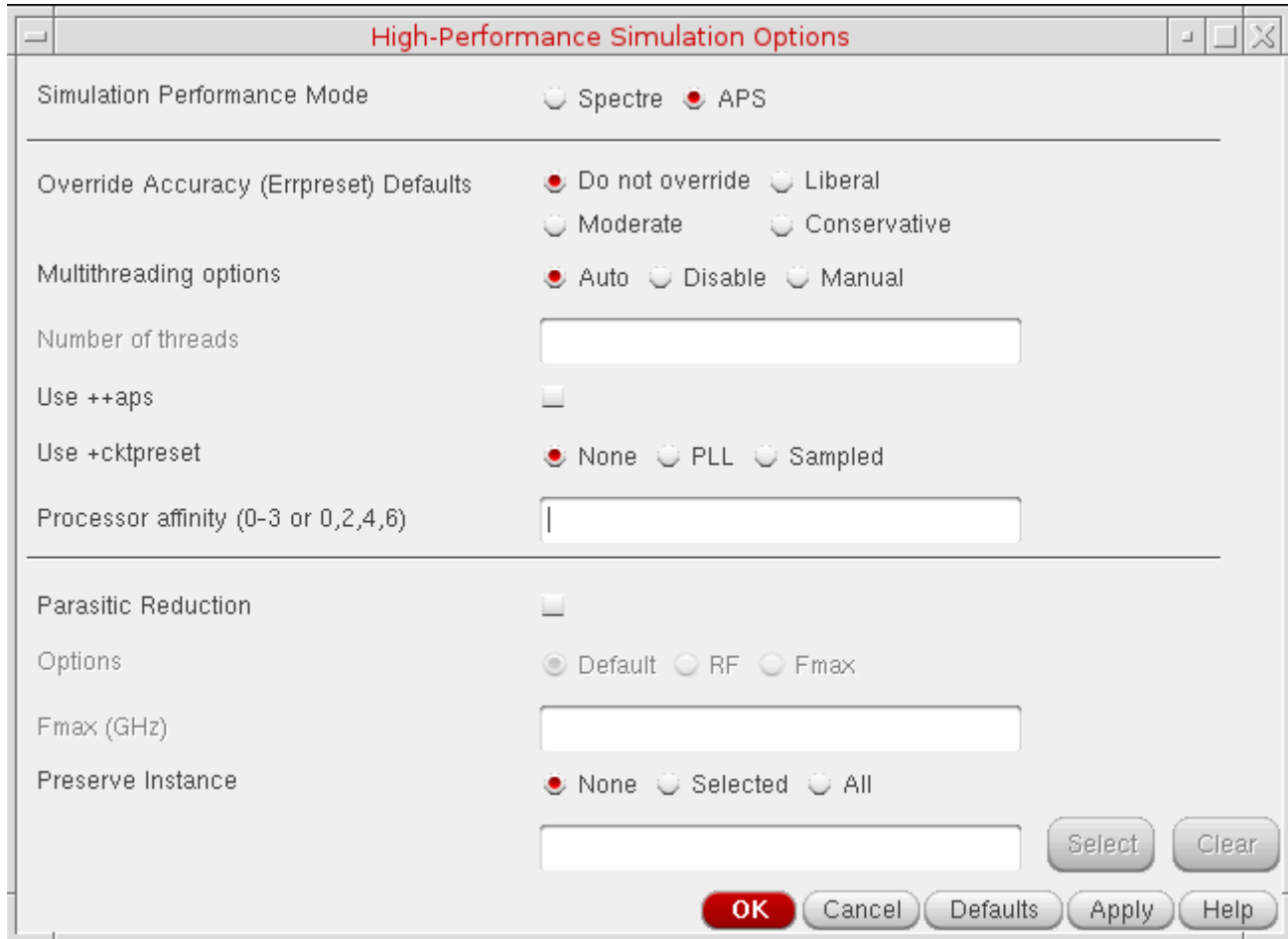
The High-Performance Simulation Options form appears.

Note: The *High-Performance Simulation* menu option is disabled when a Spectre

Virtuoso Analog Design Environment L User Guide

Running a Simulation

simulation is running in the interactive mode.



2. Select one of the following simulation performance modes:

Select	To
<i>Spectre</i>	Use the baseline Spectre version for simulation
<i>APS</i>	Use the Accelerated Parallel Simulator (APS) for simulation

3. Switch between Spectre and APS default tolerance options using the *Reset cdsenv options* checkbox. When the *Simulation Performance Mode* is set to Spectre and the *Reset cdsenv options* checkbox is selected, ADE resets the transient options to Spectre default settings. If you want to switch to APS default settings, select APS from the *Simulation Performance Mode* and select the *Reset cdsenv options* checkbox.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Note: The default transient options will change from Spectre to APS only if you change the *Simulation Performance Mode* from spectre to APS and select the *Reset cdsenv options* checkbox and vice versa. If you change the *Simulation Performance Mode* and do not select the checkbox, the default options will not change.

Note: *Reset cdsenv options* checkbox is available only if the `.cdsenv` file exists in `<cds_inst_dir>/tools/dfll/etc/tools/aps/` location.

4. Specify the accuracy level for transient analysis runs by doing one of the following:

- Select *Do not override* to use the accuracy level specified for transient analyses runs in the Choosing Analysis form. For more information about setting up a transient analysis, see [Transient Analysis](#) on page 174.
- Select *Liberal*, *Moderate*, or *Conservative* to override the accuracy level specified for transient analyses runs in the Choosing Analyses form.

For more information about the *Liberal*, *Moderate* and *Conservative* accuracy levels, see the documentation for the `errpreset` parameter in the *Virtuoso Spectre Circuit Simulator Reference*.

5. Select one of the following multithreading options:

Select	To
<i>Auto</i>	Use the maximum number of available threads to run the simulation. The number of threads used—one thread for every CPU core—is automatically derived from the hardware architecture in use, but is limited to a maximum of: <ul style="list-style-type: none">■ 4 threads when <i>Spectre</i> is set as the simulation performance mode■ 32 threads when <i>APS</i> is set as the simulation performance mode

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Select	To
<i>Manual</i>	Specify the number of threads to be used to run simulation. Enter the number of threads in the <i>Number of Threads</i> field. You can specify up to: <ul style="list-style-type: none">■ 4 threads when <i>Spectre</i> is set as the simulation performance mode■ 32 threads when <i>APS</i> is set as the simulation performance mode
<i>Disable</i>	Disable multithreading.

Note the following:

- The multithreading options specified in this form override the multithreading options specified in the Simulator Options form. For more information about the Simulator Options form, see [Setting Simulator Options](#) on page 302.
- If *Spectre* is set as the analog solver for the AMS Designer simulator, multithreading is supported only if you are using the Cadence IUS 8.2 or a later release.

6. Select the *Use ++aps* option to enable the Fast APS mode. This mode uses a different time-step control algorithm compared to spectre, which results in improved performance while satisfying error tolerances and constraints.

Note: *Use ++aps* option is enabled only when you set the Simulation Performance Mode to APS.

7. Select *Use +cktpreset* option to set a group of options for simulation based on the circuit type. You can select *PLL* option for PLL circuits and *Sampled* option for ADC circuits.

8. In the *Processor affinity* field, specify the processors using which simulation sessions should be run.

When running multiple multi-threading simulation sessions on a single computer, it is recommended that you fix the simulation session to particular processors. Otherwise multiple simulation sessions can race against each other to get the available processors, resulting in a less than optimum simulation performance. For example, to run two 4-thread simulation sessions on an 8-processor computer, specify 0–3 in the *Processor affinity* field for the first 4-thread simulation session using processors 0 to 3, and 4–7 in the *Processor affinity* field for the second 4-thread simulation session using processors 4 to 7.

Note: You can specify a range of processors such as 0–4, or a comma-separated list of

Virtuoso Analog Design Environment L User Guide

Running a Simulation

processors such as 0, 2, 4, 6.

9. To enable parasitic reduction for circuits with RC parasitics, select the *Parasitic Reduction* check box and select one of the following options:

Select	To
<i>Default</i>	Run parasitic reduction in the default mode.
<i>RF</i>	<p>Preserve the level of accuracy needed for RF analysis while running parasitic reduction.</p> <p>Note: Cadence recommends using this option for RF analyses.</p>
<i>Fmax</i>	<p>Specify an Fmax value for parasitic reduction.</p> <p>The <i>Fmax (GHz)</i> field is enabled if this option is selected. Specify an integer Fmax value in the <i>Fmax (GHz)</i> field. The default value is 1GHz.</p>

Important Points to Note:

- The *Parasitic Reduction* option is enabled only when you set the Simulation Performance Mode to APS.
 - If parasitic reduction is enabled, the simulator mode is automatically changed to `batch`, even if you have used the `controlMode` environment variable to specify the simulator mode as `interactive`. For more information about the `controlMode` environment variable, see [controlMode](#) on page 681.
10. Specify the instances that you do not want to be shorted by selecting the required *Preserve Instance* option:

Select	To
<i>None</i>	Specify that no instances need to be preserved from being shorted.

Select	To
<i>Selected</i>	<p>Specify the instances that must be preserved from being shorted.</p> <p>In the <i>Preserve Instance</i> field, type the instance names separated by spaces. You can also use wildcards to specify the instances. For example, type <code>I0.I1.r*</code> to specify that all instances with names starting with <code>r</code> in the <code>I0.I1</code> hierarchy must be preserved from being shorted.</p> <p>To select instances from the schematic, do the following:</p> <ol style="list-style-type: none">1. Click the <i>Select</i> button next to the <i>Preserve Instance</i> field. The schematic for the design is displayed.2. Select one or more instances on the schematic. To select more than one instance at a time, do one of the following:<ul style="list-style-type: none"><input type="checkbox"/> Hold down the <i>Shift</i> key and click the instances you want to select.<input type="checkbox"/> Drag the mouse pointer over the instances you want to select. All the instances that are within the yellow bounding box that appears are included in the selection.3. Press the <i>Esc</i> key when you are done. The selected instances are displayed in the <i>Preserve Instance</i> field.
<i>All</i>	Specify that all instances should be preserved from being shorted.

11. Click *OK*.

The Simulation window displays the settings you made in the High-Performance Simulation Options form. For example, if you selected the simulation performance mode as *Turbo* and the accuracy level as *Liberal*, the status bar in the Simulation window displays the simulator name as:

```
spectre turbo liberal
```

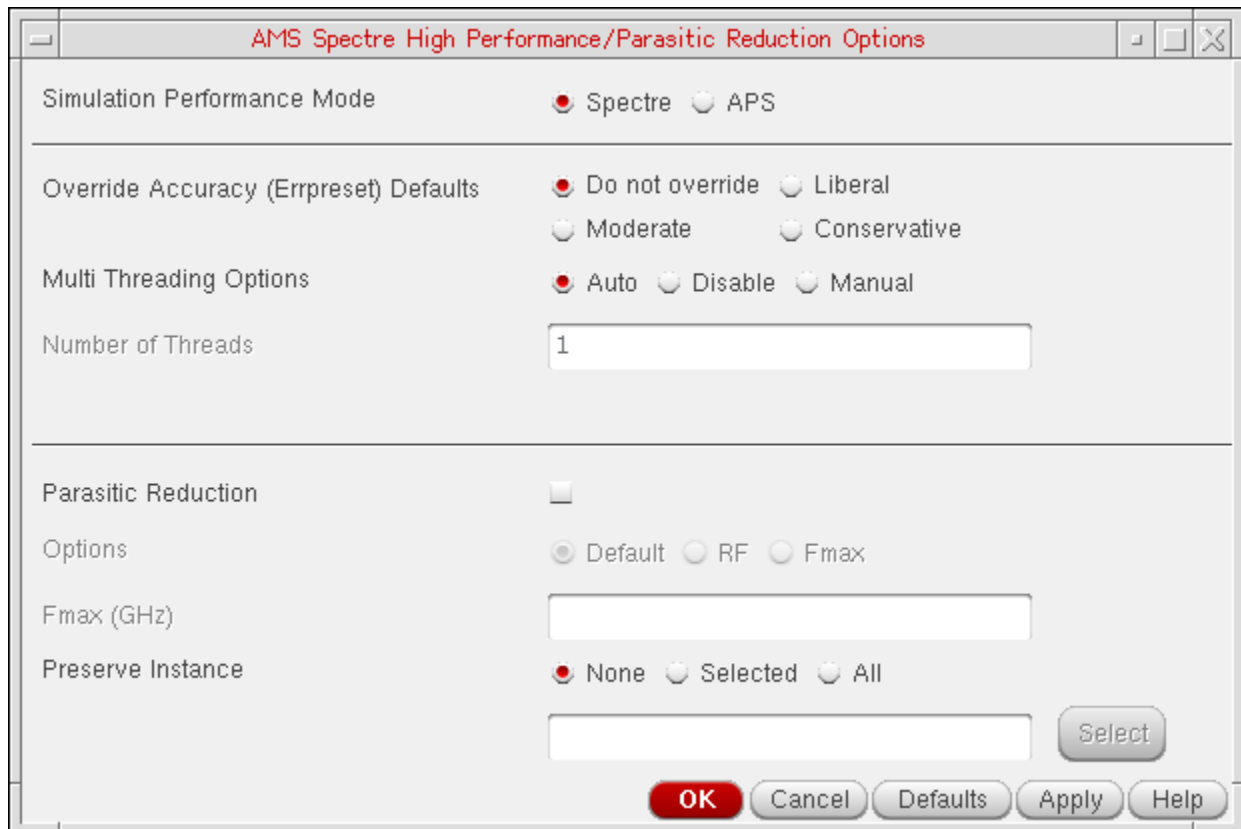
Specifying AMS Spectre High Performance/Parasitic Reduction Options

To specify parasitic reduction options when *Spectre* is set as the analog solver for the AMS Designer simulator, do the following:

Virtuoso Analog Design Environment L User Guide

Running a Simulation

- ➔ In the Simulation window, choose *Setup – High-Performance/Parasitic Reduction*.
- If *Spectre* is set as the analog solver, the AMS Spectre High performance/Parasitic Reduction Options form appears.



- ➔ To enable parasitic reduction, select the *Parasitic Reduction* check box.

UltraSim Options

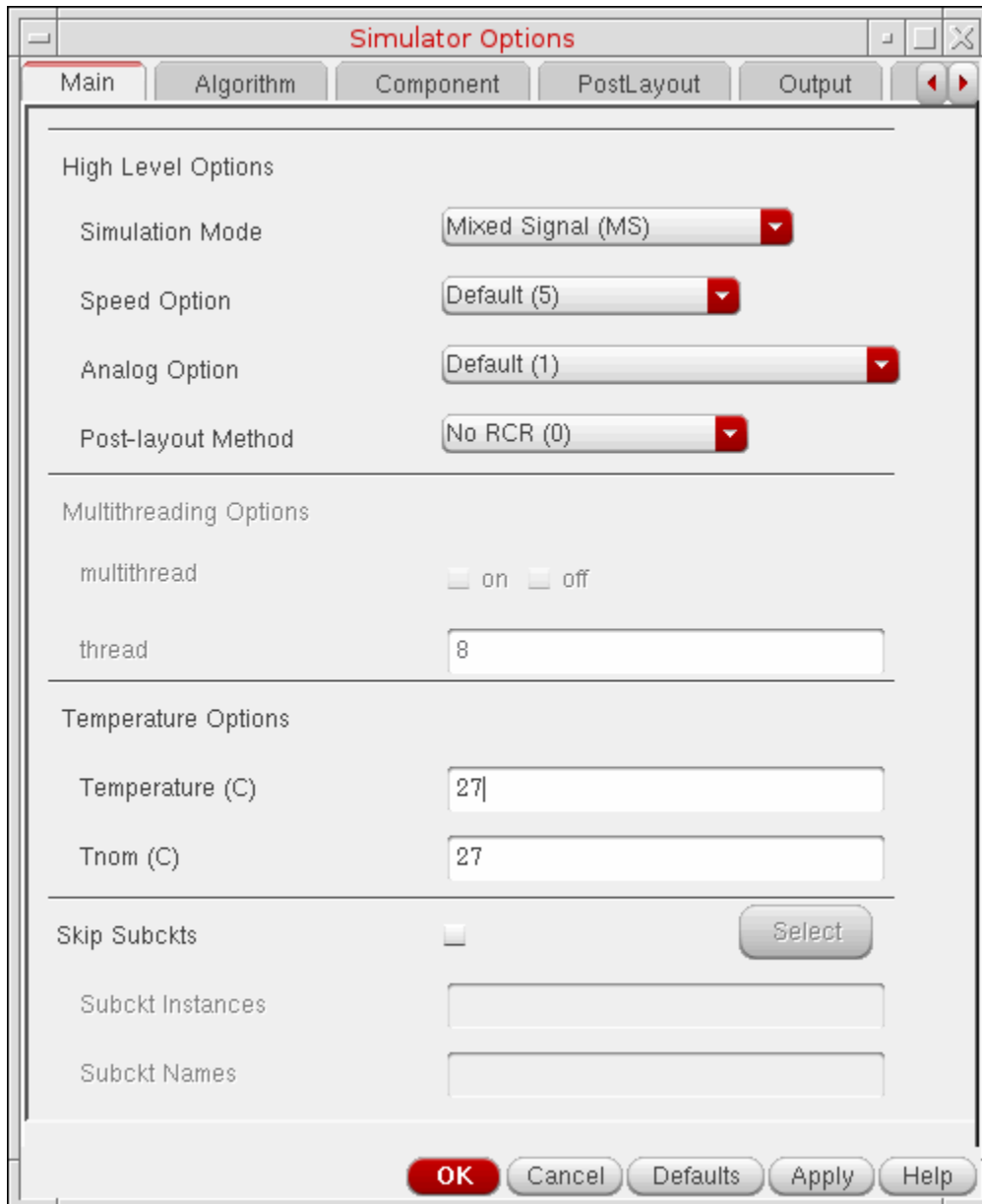
To set the Virtuoso UltraSim simulator options,

1. In the Simulation window, choose *Simulation – Options – Analog*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Simulator Options form appears.



2. Set the simulator options as needed.

For more details about the Virtuoso UltraSim simulator options, refer to the [*Virtuoso UltraSim Simulator User Guide*](#).

3. Click *OK*.

Setting Voltage Regulator Simulation Options

The voltage regulator (VR) simulation feature in the UltraSim simulator allows you to simulate designs with large circuit blocks powered by internal voltage regulators. For more information about VR simulation, see the *[Voltage Regulator Simulation](#)* in the *Virtuoso UltraSim Simulator User Guide*.

To set the voltage regulator options,

1. In the Simulation window, choose *Simulation – Options – Analog*.

The Simulator Options form appears.

2. Click the Main tab.
3. Ensure that the global simulation mode specified in the *Simulation Mode* cyclic field is Digital Extended (DX), Digital Fast (DF), Mixed Signal (MS) or Digital Accurate (DA).

Note: VR simulation can be enabled only if the global simulation mode is Digital Extended (DX), Digital Fast (DF), Mixed Signal (MS) or Digital Accurate (DA).

4. Click the Miscellaneous tab.
5. Select the *Voltage Regulator* check box to enable VR simulation.
6. Click the *Setting* button next to the *Voltage Regulator* check box.

The Voltage Regulator Settings form appears.



You can use this form to specify the cells, cell terminals, instances and nets to be used for VR simulation. Note the following:

Note: You can specify cell terminals only if you are using the simulation front end parser (SFE) with the Ultrasim simulator. For information about using the SFE parser with the Ultrasim simulator, see the *Virtuoso AMS Designer Simulator User Guide*.

For more information, see the following topics:

- ❑ [Adding Objects for VR Simulation](#) on page 314
- ❑ [Verifying the Objects Added for VR Simulation](#) on page 315

- [Disabling and Enabling Objects for VR Simulation](#) on page 316
- [Modifying Objects](#) on page 316
- [Deleting Objects](#) on page 316
- [Highlighting and Dehighlighting Enabled Objects on the Schematic](#) on page 316

Adding Objects for VR Simulation

To add an object (cell, cell terminal, instance or net) for VR simulation,

1. Select the voltage regulator type.

Select	To
Cell	Add a voltage regulator cell.
Cell Terminal	Add a cell terminal. The terminal must be an output voltage terminal delivering supply voltages to digital cells or channel connected devices.
Instance	Add a voltage regulator instance, or an output device of the voltage regulator in case your circuit has no hierarchy.
Net	Add a net. The net must be an output voltage terminal delivering supply voltages to digital cells or channel connected devices.

2. Enter the name of the object in the text field, or click the *Select* button to select the object on the schematic.
 - If the voltage regulator type is *Cell*, enter the cell name in the *Cell* field, or click the *Select* button to select the cell on the schematic.
 - If the voltage regulator type is *Cell Terminal*, enter the cell terminal name in the *Cell Terminal* field using the format "library" "cell" "port", or click the *Select* button to select the cell terminal on the schematic.
 - If the voltage regulator type is *Instance*, enter the instance name in the *Instance* field, or click the *Select* button to select the instance on the schematic.

- ❑ If the voltage regulator type is *Net*, enter the net name in the *Net* field, or click the *Select* button to select the net on the schematic.

3. Click *Add*.

The object is displayed in the Voltage Regulator Summary table.

Note: By default, the object is enabled for VR simulation. For more information, see [Disabling and Enabling Objects for VR Simulation](#) on page 316.

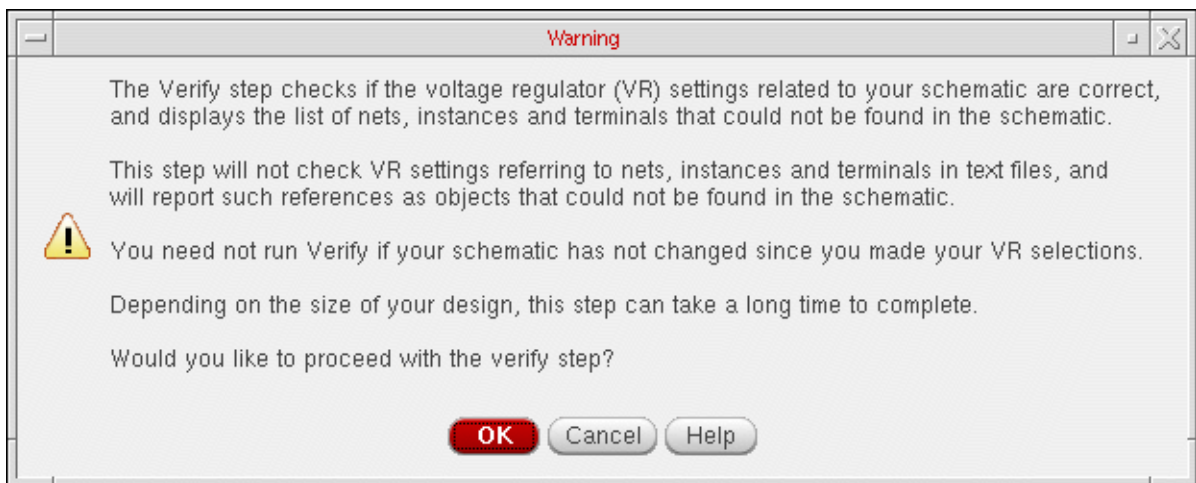
Verifying the Objects Added for VR Simulation

After specifying the objects for VR simulation, you can verify whether the objects exist in the schematic.

To verify whether the objects enabled for VR simulation exist in the schematic,

1. Click the *Verify* button.

The following message box appears:



2. Click *OK*.

If any of the objects do not exist in the schematic, a message box appears displaying the list of such objects.

Note the following:

- Only the objects that are enabled in the Voltage Regulator Summary table are verified.

- The verification process does not check objects that exist in text files, and will report such objects as not found in your schematic.

Disabling and Enabling Objects for VR Simulation

You can disable and enable objects for VR simulation. The *On* status in the *Enable* column in the Voltage Regulator Summary table indicates that the object is enabled for VR simulation. The *Off* status in the *Enable* column indicates that the object is disabled for VR simulation.

To disable an object for VR simulation,

- Select the object in the Voltage Regulator Summary table and click the *Enable/Disable* button.

To enable an object for VR simulation,

- Select the object in the Voltage Regulator Summary table and click the *Enable/Disable* button.

Modifying Objects

To modify an object in the Voltage Regulator Settings form,

1. Select the object in the Voltage Regulator Summary table.

The information for the object is displayed in the *Select Voltage Regulator* group box.

2. Modify the object information and click the *Change* button.

The changes are displayed in the Voltage Regulator Summary table.

Deleting Objects

To delete an object in the Voltage Regulator Settings form,

- Select the object in the Voltage Regulator Summary table and click the *Delete* button.

Highlighting and Dehighlighting Enabled Objects on the Schematic

The objects that are enabled for VR simulation can be highlighted on the schematic.

To highlight an enabled object on the schematic,

- Select the object in the Voltage Regulator Summary table and click the *Highlight* button.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The selected objects are highlighted in yellow color on the schematic. If the selected object is a cell, all instances of the cell are highlighted on the schematic.

Note: To highlight multiple objects, hold down the *Shift* key (for contiguous selection) or the *Ctrl* key (for noncontiguous selection) and click the next object to add more objects to the selection set, then click the *Highlight* button.

To dehighlight all objects on the schematic,

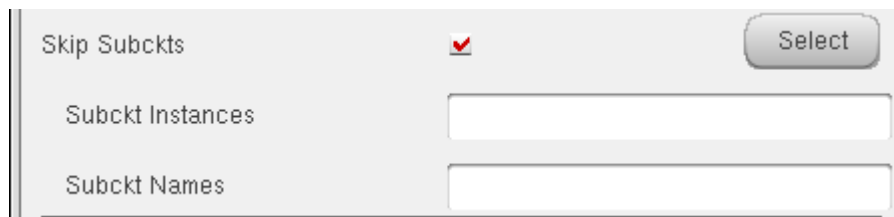
- ▶ Click the *Unhighlight All* button.

Setting Skip Options in Ultrasim

To set the Skip options in Ultrasim:

1. In the Simulation window, choose *Simulation – Options – Analog*.
2. The Simulator Options form appears, choose *Main Tab*.

The skip Subckts options appears in the *Main Tab* of Simulator Options form. Set the skip options as required.



The screenshot shows a dialog box titled "Skip Subckts" with a checked checkbox and a "Select" button. Below the checkbox are two input fields: "Subckt Instances" and "Subckt Names".

You can enter multiple subckt instances and names in respective fields. You can also directly select instances from the schematic, using the Select button.

Setting Block-Based *usim_opt* Options

Schematic

The block-based *usim_opt on schematics* option lets you set speed, accuracy, and functionality of the Virtuoso UltraSim simulation for local subcircuit instances. Any Virtuoso UltraSim simulator options, including *usim_opt*, can be set in instance properties on the schematic. The most commonly used options are

- *sim_mode* (df/da/ms/a/s)
- *speed* (1-8)

Virtuoso Analog Design Environment L User Guide

Running a Simulation

■ analog (1/2/3)

Note: The *usim_opt* option is valid only for block-level (subcircuit instances) settings.

For more details, refer to *Simulation Options* chapter in the *Virtuoso UltraSim Simulator User Guide*.

To set block-based *usim_opt on schematic* options

1. Click on an instance in the schematic.
2. In the schematic window, choose *Edit – Properties – Objects*.

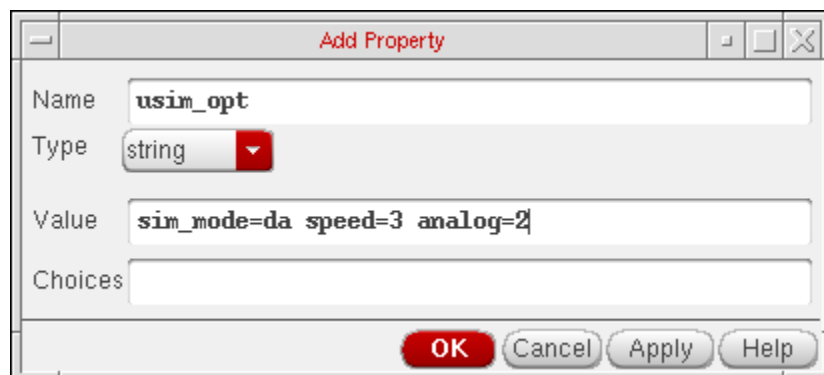
The Edit Object Properties form appears.

Property	Value	Display
Library Name	amslib	off
Cell Name	comparator	value
View Name	symbol	off
Instance Name	I0	off

User Property	Master Value	Local Value	Display
interfaceLastCh..	3 06:33:08 2000		off

3. Click *Add*.

The Add Property form appears.



4. In the *Name* field, type `usim_opt`.
5. Type `sim_mode=da speed=3 analog=2` into the *Value* field.
6. Click *OK* to save the settings and close the Add Property form.
7. In the Edit Object Properties form, click *OK*.
8. In the schematic, choose *Design – Check and Save*.
9. In the Cadence® Analog Design Environment simulation window, choose *Simulation – Options – Analog*.

The Simulator Options form appears.

10. Turn on *Allow usim_opt on schematics*.
11. Click *OK* to save the settings and close the Simulator Options form.
12. Run netlisting or the simulation from the Simulation window.

The `usim_opt` settings are set locally for the instance block in the netlist.

Note: Virtuoso Spectre and HSPICE netlist formats are supported in IC 5.0 and later releases.

Hierarchy Editor

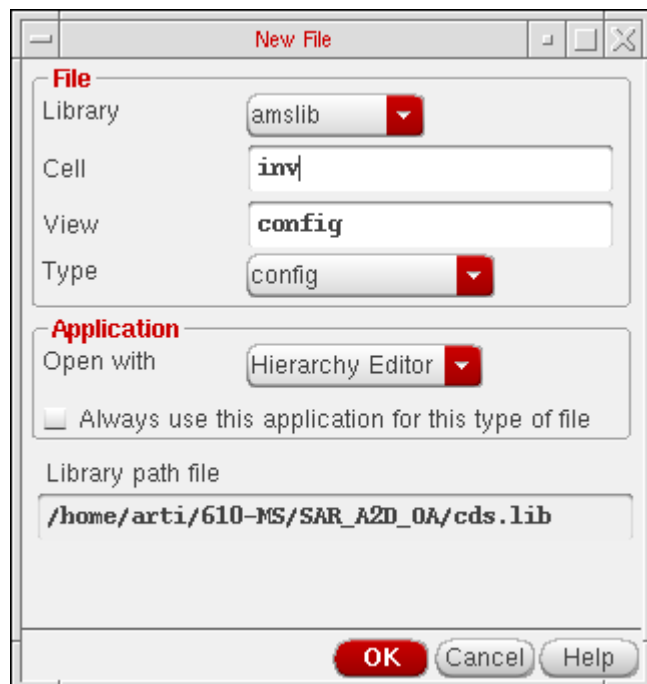
You can also set block level `usim_opt` options, such as speed and accuracy, using the Cadence® Hierarchy Editor (HED). For more information about setting Virtuoso UltraSim simulator options, refer to *Simulation Options* chapter in the *Virtuoso UltraSim Simulator User Guide*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

1. From the CIW, choose *File – New – Cellview*.

The Create New File form appears.



2. Choose a library, cell, and view.
3. Choose *Hierarchy-Editor* from the *Tool* drop-down list box.
4. Click *OK*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The New Configuration form appears.

The screenshot shows a window titled "New Configuration" with the following fields and controls:

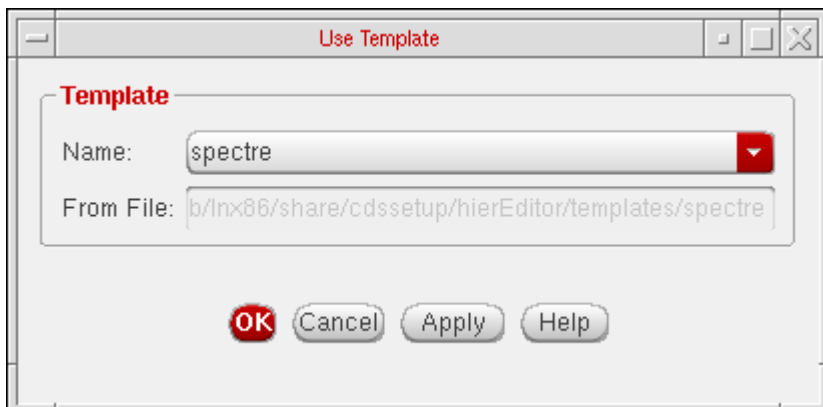
- Top Cell:**
 - Library: (dropdown arrow)
 - Cell: (dropdown arrow)
 - View: (dropdown arrow)
- Global Bindings:**
 - Library List:
 - View List:
 - Stop List:
 - Constraint List:
- Description:**
 - A large empty text area for entering a description.
- Buttons:** OK, Cancel, Use Template, Help

5. Click on the *Use Template* button located at the bottom of the form.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Use Template form appears.



6. Choose *spectre* from the *Name* drop-down list box.

7. Click *OK*.

The New Configuration form redisplay with default data for the *Top Cell* and *Global Bindings* sections.

8. In the *Top Cell* section, enter the desired library, cell name, and schematic view.

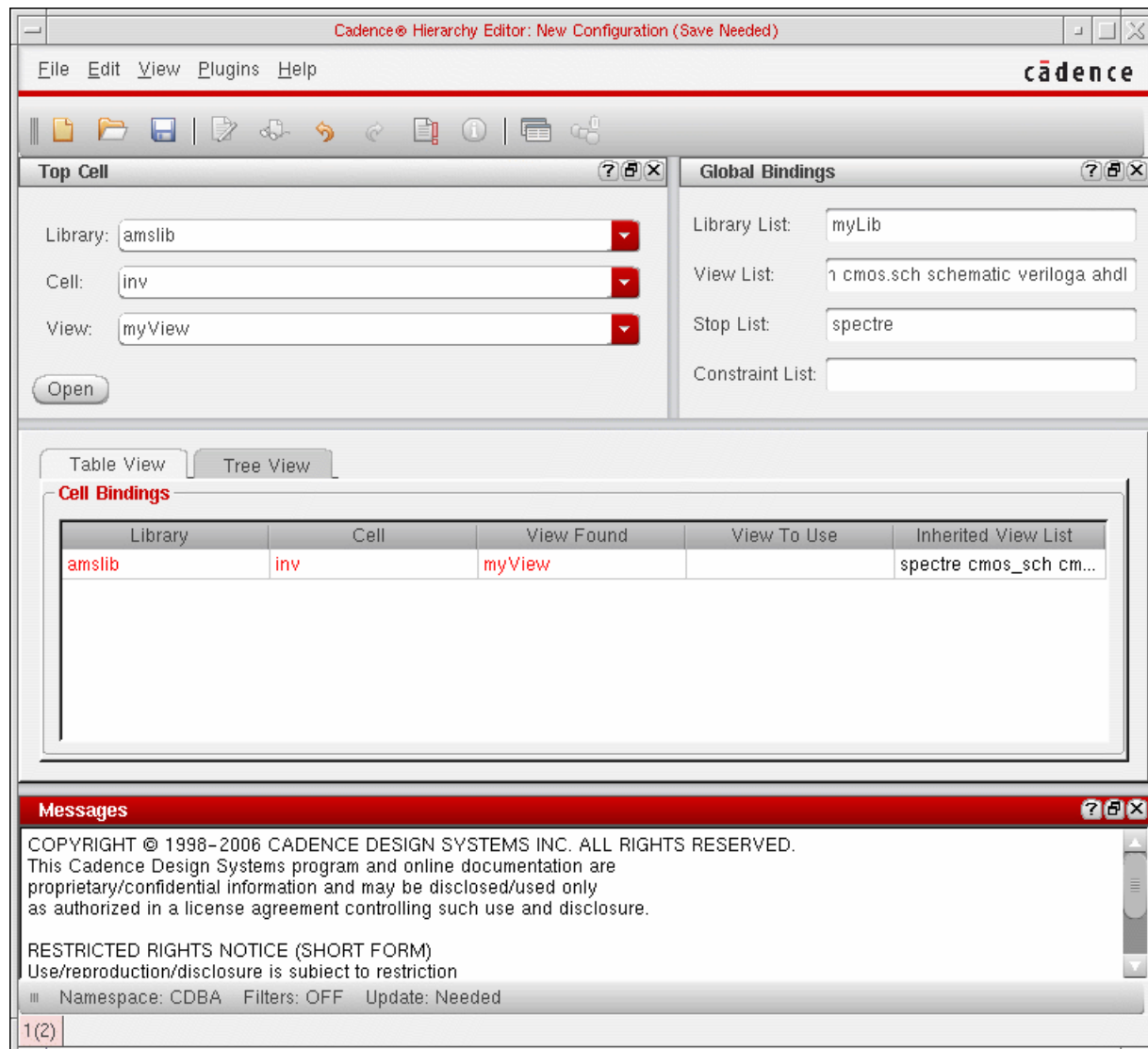
9. In the *Global Bindings* section, remove `myLib` from the *Library List* field.

10. Click *OK*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Cadence hierarchy editor form displays your data.



Note: The hierarchy editor form configures the design by using a default *View List* and *Stop List* in the *Global Bindings* section. You need to modify these lists for your design.

11. Choose *View – Properties*.

The *sim_mode* and *speed* columns appear in the *Cell Bindings* section.

12. Choose *Edit – Add Property Column* to add additional cell- or occurrence-based properties to the *Cell Bindings* section.

Note: Instance-based properties are not supported by the Virtuoso UltraSim simulator in ADE.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

13. Set options for cell- or occurrence-based properties.

To set options for cell-based properties,

- a. Right-click in a property column.
A popup menu appears.
- b. Choose *Set “property name” Cell Property* to set the property.

To set options for occurrence-based properties,

- a. Choose *View – Tree*.
- b. Select an instance and right-click on a property column in the *Cell Bindings* section.

A popup menu appears.

Note: Do not left-click on a property column to set a property (left-click creates an instance-based property, which is not supported by the Virtuoso UltraSim simulator in ADE).

- c. Choose *Set “name” Occurrence Property*.
- d. Choose the appropriate property value.
- e. Click *OK*.

The instance is marked with an \circ symbol.

- f. Choose *View – Update*.

The Update Sync-up form appears.

The following cellviews have been edited but not saved. In order to sync-up the hierarchy editor with your application, you must save the relevant cellviews in your hierarchy. Please select the cellviews you want to save:

Select	Cellview
<input checked="" type="checkbox"/>	()

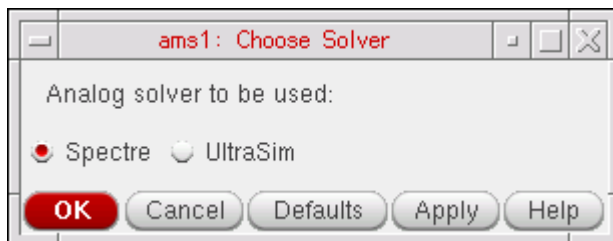
- g. Click *OK* to save the cellview.

AMS Options

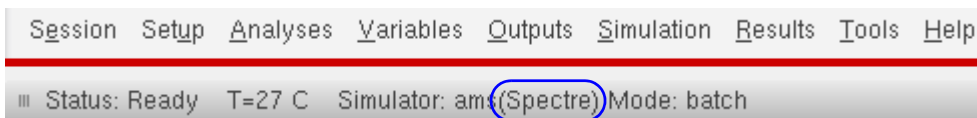
When you select *ams* as your simulator, the Simulation menu offers you an option to select a solver for the simulation. After doing that, you can set Virtuoso AMS Designer simulator options by choosing the appropriate option from the *Simulation – Options* submenu.

Choosing a Solver

Choose *Simulation – Solver* to bring up the Choose Solver form, in which you can select either *Spectre* or *UltraSim* as the solver.



Your choice appears next to the name of the selected simulator below the title bar as highlighted in the snapshot below.

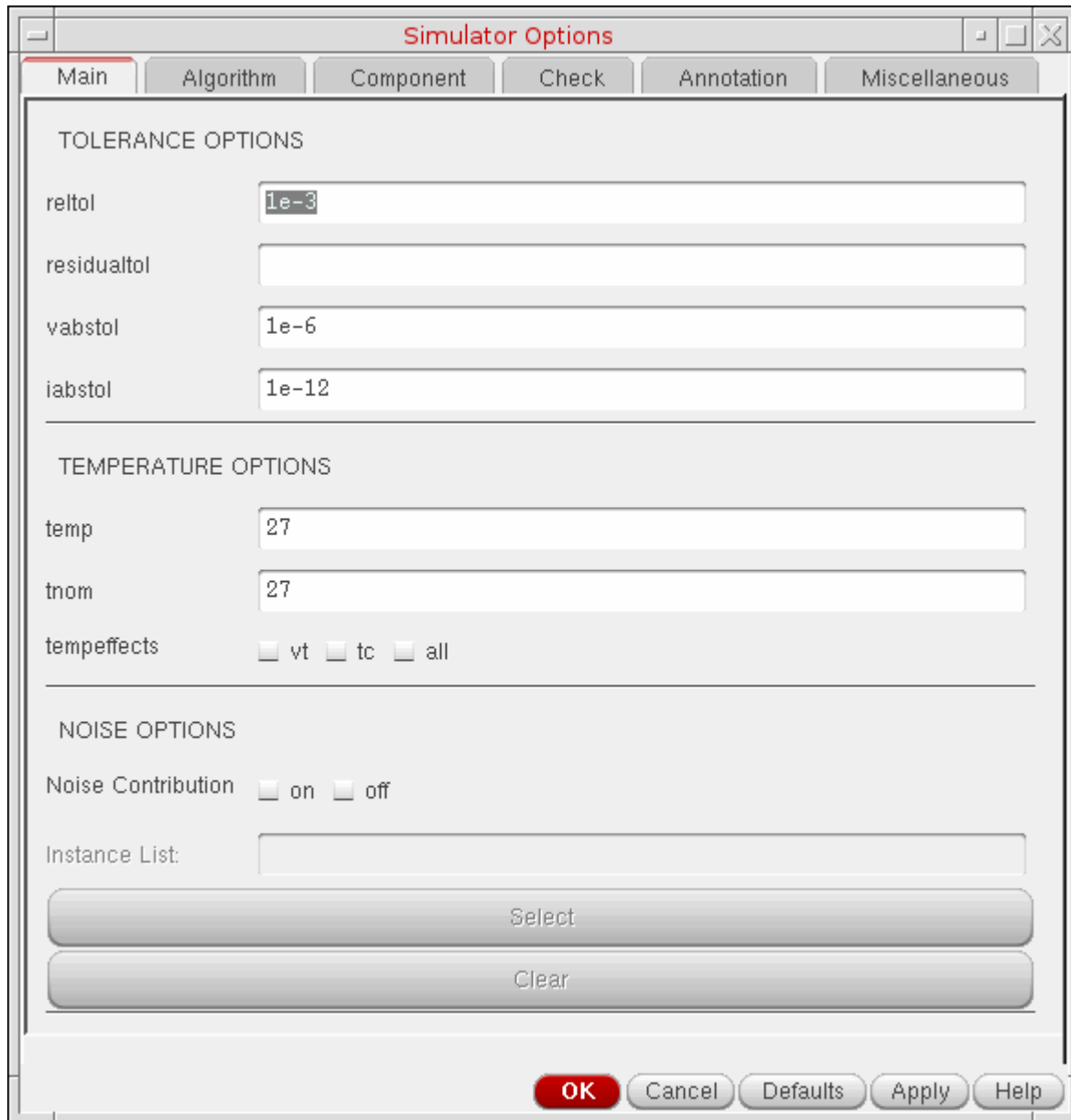


If you are using the AMS Designer Simulator in IUS 8.1 or later releases, and have selected Spectre as the solver, you can choose *Setup – Performance/Parasitic Reduction* to specify performance and parasitic reduction options. For more information, see [Specifying Performance and Parasitic Reduction Options](#) on page 304.

Note: When you change the solver, the values in all GUI fields will be reset to the default values for that solver.

Analog (Spectre)

Choose *Simulation – Options – Analog (Spectre)*.



For details refer to the *Immediate Set Options* in the *Virtuoso Spectre Circuit Simulator Reference*.

AMS Simulator

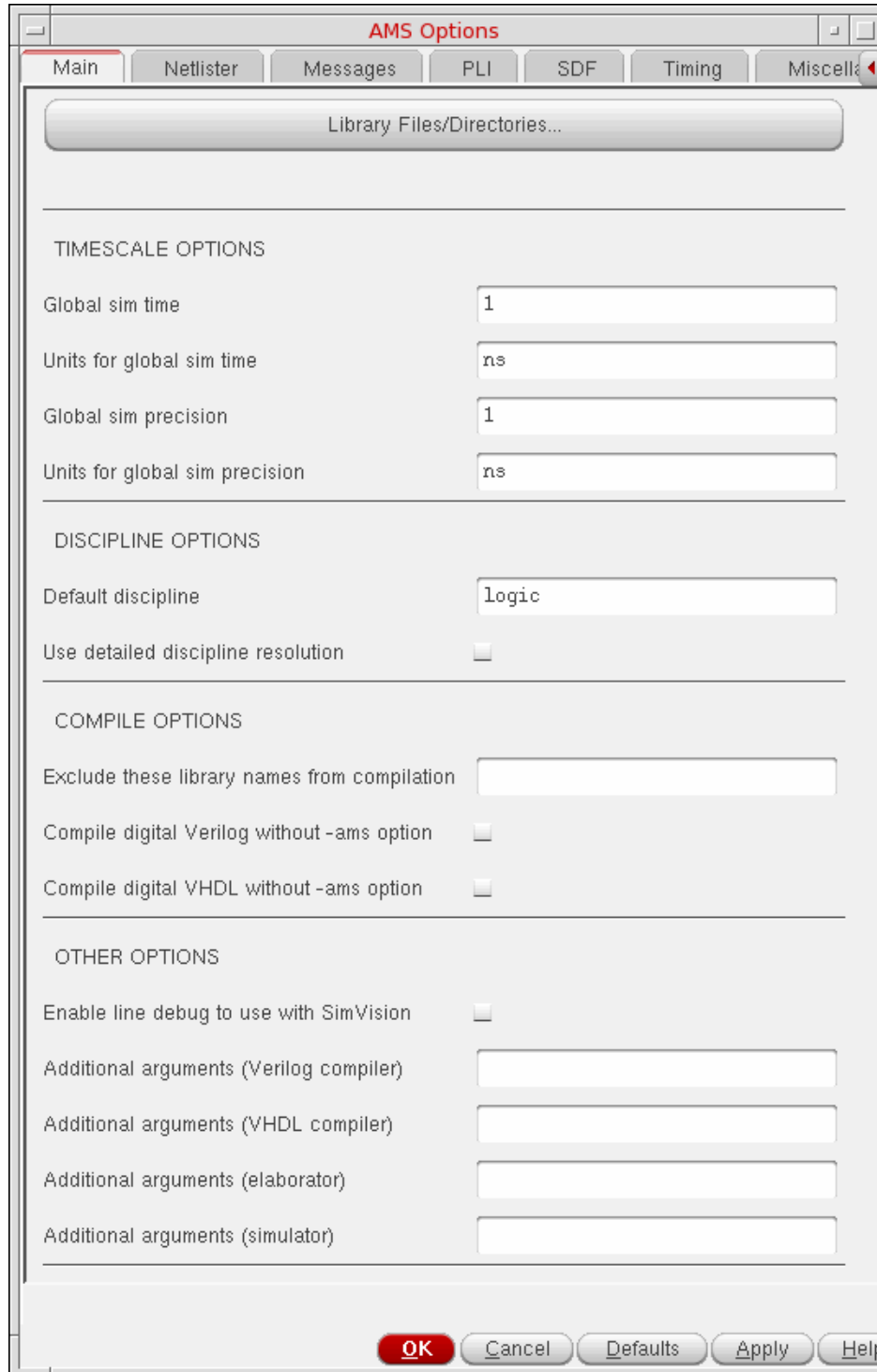
Starting with the IC 6.1 release, the AMS Simulator form has been tabulated. Various tabs in this form contain the options that were earlier present in Compiler, Elaborator and Netlister form. The form contains the following tabs:

- Main
- Netlister
- Messages
- PLI
- SDF
- Timing
- Miscellaneous

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Choose *Simulation – Options – AMS Simulator*



Virtuoso Analog Design Environment L User Guide

Running a Simulation

In the *Main* tab, you can specify INCLUDE, TIMESCALE, DISCIPLINE, COMPILE, and OTHER options.

In INCLUDE section, specify the following options:

Library Files/Directories: The *Library Files/Directories* button brings up the Select Library Files/Directories form using which you can specify files that you want compiled.



You can specify the following information in this form.

- Library files:** Paths of library files, separated by spaces. The extensions of these files must be one of these: `.v`, `.va`, `.vams`, `.vhdl`, `.vhms`. The paths are relative to the netlist directory.
 - Library directories:** Paths of directories, separated by spaces. Files in the specified directories are considered if they have these extensions: `.v`, `.va`, `.vams`, `.vhdl`, `.vhms`.
 - Valid extensions for dirs:** Valid extensions for library directories, separated by spaces.
 - View to compile into:** The view in which you want the files to be compiled. The default view is `module`.
 - Library to compile into:** The library into which you want the files to be compiled. This library should exist in `cds.lib`. The default is the design library.
- Note:** Ensure that for ncelab to pick the compiled files, the library in which the files get compiled is specified in Library List Global Bindings in the Hierarchy Editor.
- Language:** The language you want to compile the files into. You can select any of these: *Verilog*, *VerilogA*, *VerilogAMS*, *VHDL* or *VHDLAMS*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

These settings can also be collectively saved by using the Library Files check box in the Saving State form.

In the TIMESCALE and DISCIPLINES section, specify the following options:

Global sim time: Specify the global simulation time.

Units for global sim time: Specify the unit to be used for global simulation time.

Global sim precision: Specify the global simulation precision.

Units for global sim precision: Specify the unit to be used for global simulation precision.

Default discipline: Specify the default discipline.

Use detailed discipline resolution: Specify whether you want to use the detailed discipline or not.

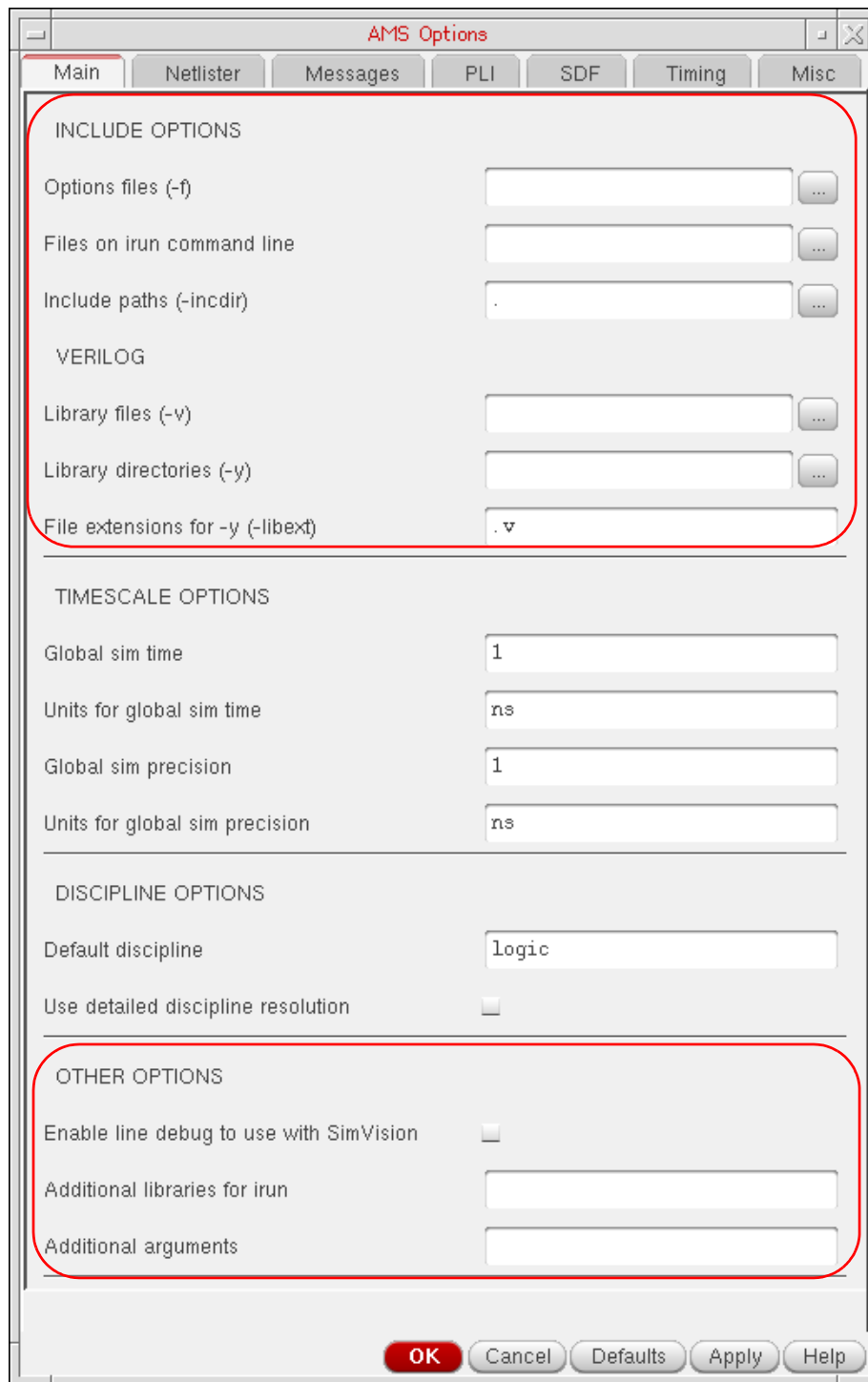
Use the *COMPILE OPTIONS* to specify the library names that can be excluded from compilation and if digital Verilog and VHDL should be compiled without the *-ams* option.

Use the *OTHER OPTIONS* to enable VHDL/Verilog options and to specify other arguments.

The above holds true if you have selected *Cellview-based netlister with ncvlog, ncelab, ncsim* netlister mode. If you select the netlister mode as *OSS-based netlister with irun*, you will notice the following changes in the *Main* tab.

Virtuoso Analog Design Environment L User Guide

Running a Simulation



Options file(-f): (Equivalent to the `-f` command-line option) Reads the command-line options contained in the specified file.

Note: If you are using an IUS release earlier than IUS 6.1, ADE will use the `ncoverilog` command instead of the `irun` command. You can specify the name of file containing the `ncoverilog` command-line options in this field.

Files on irun command line: Adds the specified files to the `irun` command.

Include paths (-incdir): (Equivalent to the `-incdir` command-line option) Specifies the directory to search for include files.

Library files (-v): (Equivalent to the `-v` command-line option) Includes the specified Verilog files for compilation. Separate file names with spaces.

Library directories (-y): (Equivalent to the `-y` command-line option) Includes the Verilog (`.v`) files in the specified library directories for compilation. Separate library names with spaces.

File -extensions for -y (-libext): Includes the Verilog file extensions (`.v`) in the specified library directories for compilation. Separate the file extensions with commas (`.v, .sv`).

For the *OSS-based netlister with irun* netlister mode, the `COMPILE OPTIONS` section is not displayed in *Main* tab.

Note: The netlist generated by cellview-based netlister(CBN) is not expected to work with `irun` (to simulate the design). You can use AMS UNL (or AMS-OSS Netlister) to create a netlist that can be simulated using `irun`.

Note: Starting with IC6.1.6 ISR6, the AMS Unified Netlister (AMS UNL) has been introduced. For AMS UNL flow, a new section *LIBRARY COMPILATION OPTIONS* containing the following options has been added:

- *Pre-compiled libraries (-reflib) Settings*
- *Run-time library compiling (-makelib) Settings*

These options are visible only when you set the following environment variables:

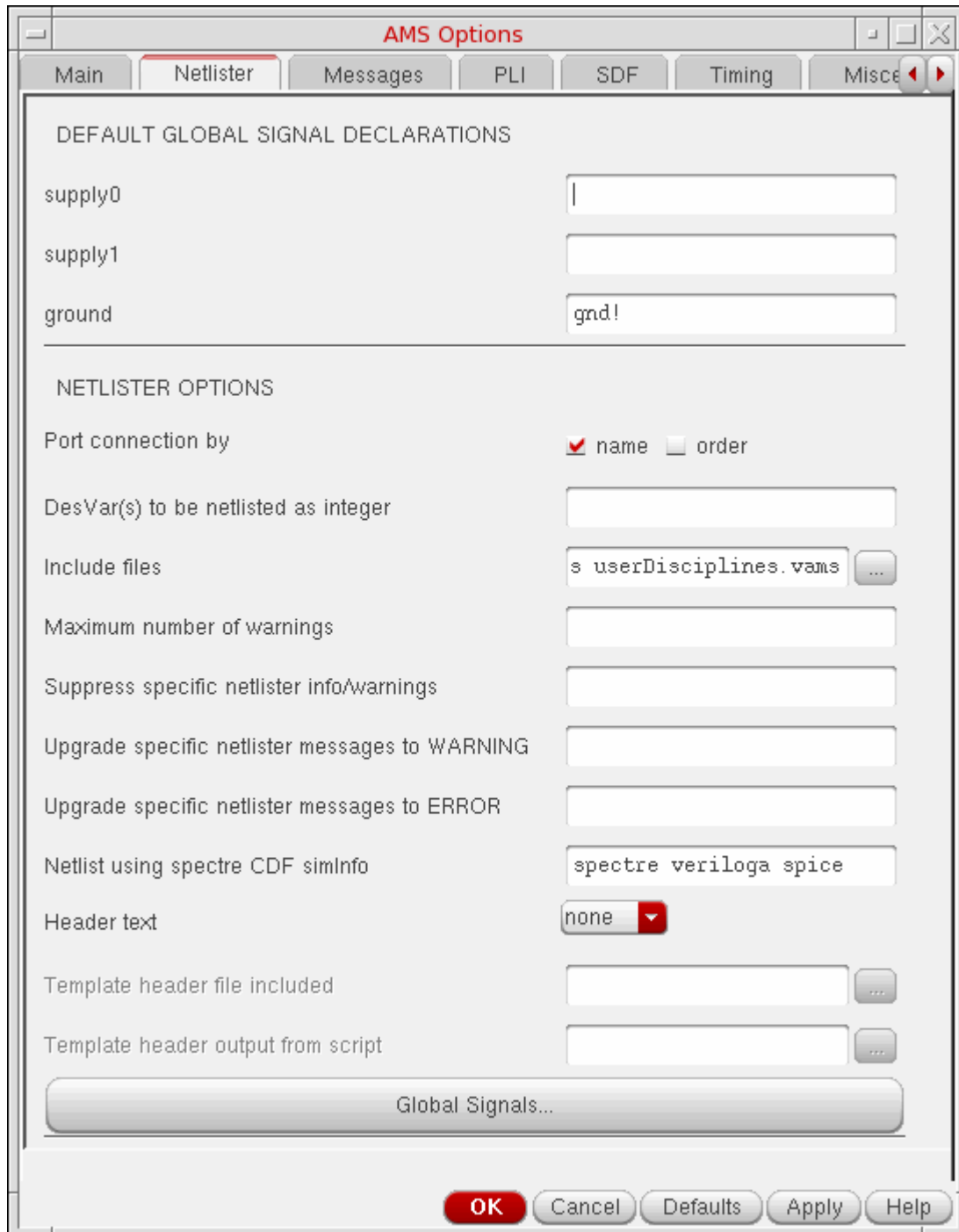
- `setenv AMS_UNL YES`
- `setenv AMS_MAKELIB_REFLIB_GUI YES`

For more information, see [*AMS ADE Unified Netlisting Flow guide*](#).

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Select the *Netlister* tab.



In the *Netlister* tab, you can specify DEFAULT GLOBAL SIGNALS DECLARATION and NETLISTER OPTIONS.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Note: The options in the *Netlister* tab are same for *Cellview-based netlister with ncvlog, ncelab, ncsim* and *OSS-based netlister with irun* netlister mode.

The **Default Global Signals Declarations** field allows you to type a space-separated list of names of global signals in the fields according to how you want to declare them:

- In the *supply0* field, type names of global signals that you want to declare as type *supply0*.
- In the *supply1* field, type names of global signals that you want to declare as type *supply1*.
- In the *ground* field, type names of global signals that you want to declare as type *ground*.

The **Port connection by** field in the NETLISTER OPTIONS group box allows you to select the type of port connection.

- Select *name* if you want an explicit connection. Using this option, the terminal names are printed on the primitive or subcircuit and the net name on the corresponding instance terminal in `<termName>=<netName>` format.
- Select *order* if you want an implicit connection. Using this option, the nets connected to instance terminals appear separated by white spaces and in the same order in which master subcircuit or primitive terminals are printed.

The **DesVar(s) to be netlisted as integer** field allows you to specify the list of design variables that should be processed as integer but not string.

The **Maximum number of warnings** field allows you to set the maximum number of warnings issued by the netlister before it stops processing the design.

Note: You can also specify the maximum number of warnings using the *netlistMaxWarn* environment variable.

The **Suppress specific netlister info/warnings** field allows you to specify a space-separated list of message IDs, such as `AMS-2000 AMS-2171 AMS-2174`, that you want to suppress while processing the design.

Note: You can also suppress the information or warning messages by using the *netlistNoWarn* environment variable.

The **Upgrade specific netlister messages to WARNING** field allows you to specify a space-separated list of information message IDs, such as `AMS-1244 AMS-1246`, that you want to be treated as warning messages while processing the design.

Note: You can also upgrade the information messages to warning messages by using the *upgradeMsgSevWarn* environment variable.

The **Upgrade specific netlister messages to ERROR** field allows you to specify a space-separated list of information and warning message id, such as `AMS-2171 AMS-2174`, that you want to be treated as error messages during the processing of the design.

Note: You can also upgrade the warning messages to error messages by using the *upgradeMsgSevError* environment variable.

The **Netlist using spectre CDF simInfo** field allows you to specify view names that will be treated as analog cells during netlisting. The specified view names are netlisted using the cell's spectre CDF simInfo and are processed as analog stopping view or analog leaf-level primitive.

The **Header text** field allows you to specify header text that you want the AMS netlister to insert in every Verilog-AMS.

- Select the *file* option to include header text from a particular file.
- Select the *script* option to include header text that results from a script file.

For more information, see *Header Text* in the *Virtuoso AMS Designer Environment User Guide*.

The **Global Signals** button brings up the Global Signals form, which is described in the next topic.

Working with Global Signals in AMS

A global signal is a signal that is connected by name across all levels of a design hierarchy without using pins. Global signals can come from schematic data or from text modules. AMS is aware of only global signals that come from schematic data. You can use the Global Signals form to declare a signal that is used as an out-of-module signal reference.

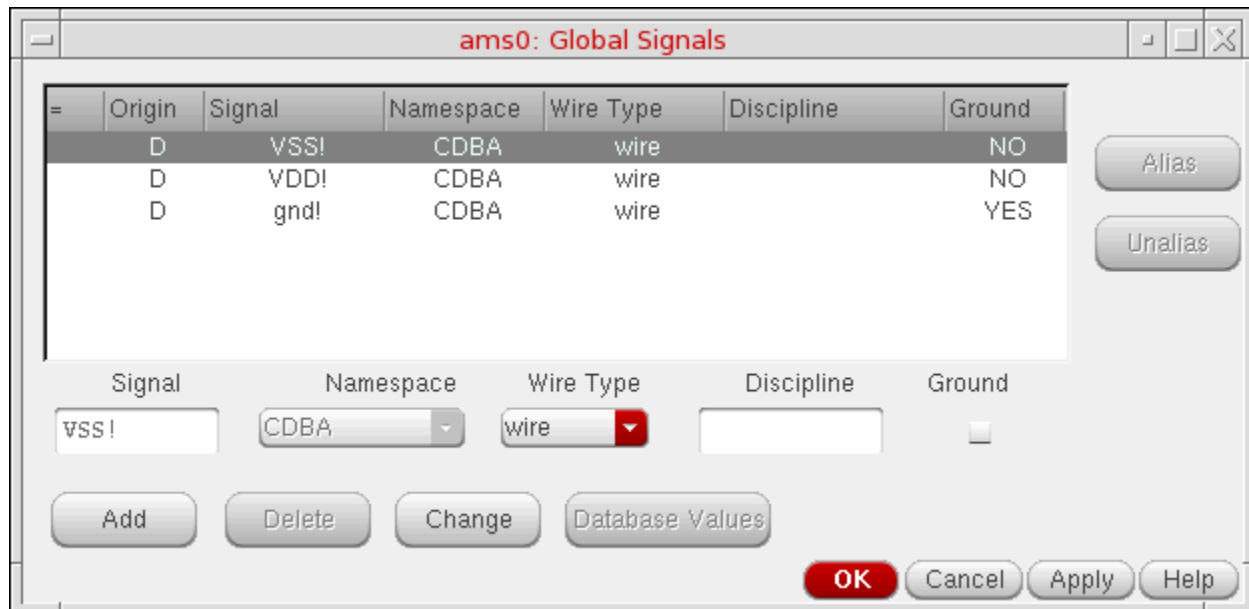
To add a global signal,

1. In the Netlister Options form, click the *Global Signals* button.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Global Signals form appears. If the design had not been netlisted after recent changes, you are prompted to netlist the design so that the Global Signals form can display the latest data.



2. Global signals are displayed in a tabular format. It includes the following information.

- The first column indicates the alias of aliased signals. They represent:
 - The beginning of the aliased set: /--
 - The aliased signals in between: |--
 - The end of the aliased set: \--
- The *Origin* column is either blank if the signal was added using the Global Signals form or it has the value **D** to indicate that a signal has been extracted from the design.
- The *Signal* column shows the name of the signal.
- The *Namespace* column displays the namespace in which the signal was created.
- The *Wire Type* column shows the wire type of the signal.
- The *Discipline* column shows the discipline of the signal.
- The *Ground* column indicates if the global signal is used as a ground reference.

Note: Netlisting extracts the signals in the design and merges them with the signals created using the Global Signals form. If you open the form without netlisting, it would be

empty.

3. The input fields below the report get populated by signal details when you select any signal. To change these values, you can either type over them or select values from the cyclic lists and then click the *Change* button.

4. To add new global signals,

a. Type a unique name for the signal in the *Signal* field. You can specify a range such as <5 : 8> by post-fixing it to the name.

b. Select the namespace as *CDBA*, *Spectre*, *Spice* or *Verilog-AMS* from the *Namespace* cyclic list.

c. Select one of these from the *Wire Type* cyclic list: *wire*, *supply0*, *supply1*, *tri*, *tri0*, *tri1*, *triand*, *trior*, *trireg*, *wand*, *wor*, or *wreal*.

Note: If the signal name you specify is included in the *supply0* or *supply1* field of the Netlister Options form, the default value for wire type would be changed to *supply0* and *supply1* accordingly.

d. Type a discipline name for the signal in the *Discipline* field.

e. If you want to use the global signal as a ground reference, select the *Ground* option. You can select this option only for signals that have the wire type *wire* or *tri*.

Note: If the signal name you specify is included in the *ground* field of the Netlister Options form, this field appears selected by default.

f. Click the *Add* button.

The new global signal appears in the list of global signals.

Note: You can create a new global signal from an existing one by selecting one, modifying its values and clicking *Add*.

5. To delete one or more signals, select them and click the *Delete* button.

You cannot delete a global signal that is extracted from the design. If you select such a signal, the *Namespace* and *Name* fields appear non-editable. If you change any of the other values, the *Database Values* button is enabled, using which you can set the fields back to their original values from the database.

6. You can alias global signals into groups. Aliased signals in a group are electrically equivalent, as if they are joined by a wire. To alias global signals, select the signals to be aliased and click the *Alias* button.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

To select signals listed consecutively, hold down the `shift` key while you click the signal names to be aliased. To select signals that are not listed sequentially, hold down the `control` key while you click on the signal names.

When you alias signals, they redisplay consecutively in the global signal list, joined by a vertical connecting bar. If you alias signals belonging to separate aliased signal groups, all of the signals in the groups are aliased.

7. To unalias signals, select the signals to be unaliased from the group, and click the *Unalias* button. If an alias set has only two signals, and you unalias one, the other also gets automatically unaliased.
8. When you have finished editing the list of global signals, click *OK*.

You need to regenerate the netlist so that the changes made in this form reflect in the netlist and `cds_globals` module is regenerated. If you try to create or re-create the netlist without applying the changes in the Global Signals form, your changes get overwritten by the netlist. This is the reason a prompt appears as follows:

```
While netlisting, the globals would again be extracted from the design and the
globals form would be updated. Unsaved changes, if any on the globals form would
be lost. Proceed with netlisting?
```

Netlisting includes the following actions:

1. It extracts information about all the global signals and design variables from the design.
2. Information about variables is merged with the design and written as it is to the `verilog.vams` file.
3. The global signals information is updated with any new extracted global signals that you modified in the current session.
4. If an extracted global signals exists in the global signals information and its origin is marked as D, and it has not been modified in the design, it is not copied.
5. If an extracted global signals exists in the global signals information and its origin is marked as D, and you modified it using the Global Signals form in the current session, the values are copied over and a message appears saying so.
6. If it exists in the global signals information for the session and the Origin is not marked as 'D', a prompt appears as follows:

```
A global with the name '%s' already exists in the session and is now also found
in the design. Using the one that exists in the session.
```

The Global Signals form appears updated the next time it is opened.

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Global Signals* option in the Saving State form and save the

Virtuoso Analog Design Environment L User Guide

Running a Simulation

current ADE state. You can later load the state using the Loading State form with the *Global Signals* option selected. For more information, see [Saving and Restoring the Simulation Setup](#) on page 97.

Note: The *Mixed Signal Netlisting Options* are available in the *Netlister* tab of the *AMS Options* form for AMS simulator only when the `Virtuoso_MixedSignalOpt_Layout` license is checked out and the *OSS-based netlister with irun* option is selected from the

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Netlist and Run Options form. These options enable the Embedded Module Hierarchy (EMH) CDL netlister to generate the digital and analog CDL netlists, simultaneously.

The screenshot shows the 'AMS Options' dialog box with the 'Netlister' tab selected. The dialog is organized into several sections:

- DEFAULT GLOBAL SIGNAL DECLARATIONS:** Contains text boxes for 'supply0', 'supply1', and 'ground' (with 'gnd!' entered).
- NETLISTER OPTIONS:** Includes checkboxes for 'Port connection by' (selected 'name'), 'Traverse config for text views' (checked), and 'Print power/ground connectivity'. It also features text boxes for 'DesVar(s) to be netlisted as integer', 'Include files' (with 's userDisciplines.vams'), 'Netlist using spectre CDF simInfo' (with 'spectre veriloga spice'), and 'Header text' (with a dropdown set to 'none'). There are also buttons for 'Template header file included' and 'Template header output from script', and a 'Global Signals...' button.
- MIXED SIGNAL NETLISTING OPTIONS:** Includes checkboxes for 'Use maskLayout views for binding digital instances' (checked), 'Print physical only instances in digital hierarchy', and 'Print definitions of leaf cells in digital hierarchy' (checked). It also has text boxes for 'Digital Netlist File Name' (with 'digital.v'), 'List of digital instances in schematic hierarchy' (with 'L/PLL_160MHE_sim/layout)'), and 'Only include instances of these physical cells'.

At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The **Use maskLayout views for binding digital instances** field allows you to enable the EMH netlisting.

The **Digital Netlist File Name** field allows you to specify the name of the output file.

The **Print power/ground connectivity** field allows you to print the power or ground connectivity across the digital module hierarchy. This does not impact the top level netlist.

The **List of digital instance in schematic hierarchy** field allows you to specify the list of digital instances in the design.

This field is automatically populated if the top-level design contains instances with the `lxEMHStart` property. You will need to update the mapped name to ensure the mapping is correct. For example, (`"L/C/Sch" "I0" "L/C/Lay"`) should be replaced with (`"L/C/Sch" "I0" "L/C/Lay" "|I0"`).

If the top-level design does not contain any instance having the `lxEMHStart` Boolean property, the *List of digital instances in schematic hierarchy* text box is automatically populated without the instance names. For example, if (`"DemoLib/top/schematic" "" "DemoLib/top/layout"`), you need to update the instance names in this text box.

It is recommended that before generating a netlist, you review the list to verify that it contains only valid SKILL forms and update accordingly. You can choose to generate the netlist in any one of the following way:

- Generate the netlist for the entire EMH contained in the specified layout view.
- Generate the netlist for the module hierarchy corresponding to the digital instances listed in this text box.

Sample inputs:

- I0 and I1 are two digital instances in the schematic hierarchy

```
(( "DemoLib/top/schematic" "I1" "DemoLib/block/layout" ) ( "DemoLib/mid/schematic" "I0" "DemoLib/top/layout" ))
```

- I0 is a digital instance in schematic, but has a mapped name I0 in the target layout cellview. This means that I0 is bound to the master module of the instance I0 in the layout module hierarchy.

```
(( "DemoLib/top/schematic" "I0" "DemoLib/block/layout" "|I0" ))
```

- I0 is a digital instance in top-level schematic, and is bound to the top-level module in the layout-embedded module hierarchy.

```
(( "DemoLib/block/schematic" "I0" "DemoLib/block/layout" t ))
```

The **Print physical only instances in digital hierarchy** field allows you to print the physical only instances in digital module hierarchy.

The **Print definition of leaf cells in digital hierarchy** field prints the empty standard cells in the digital netlist.

The digital module hierarchy in the layout contains instances of standard cells. By default, these instances are considered as stopping instances, and the mixed signal netlister does not print the subcircuit definition for standard cells. If you have the standard cell definitions available from the foundry, you can directly include these definitions during LVS or SVS.

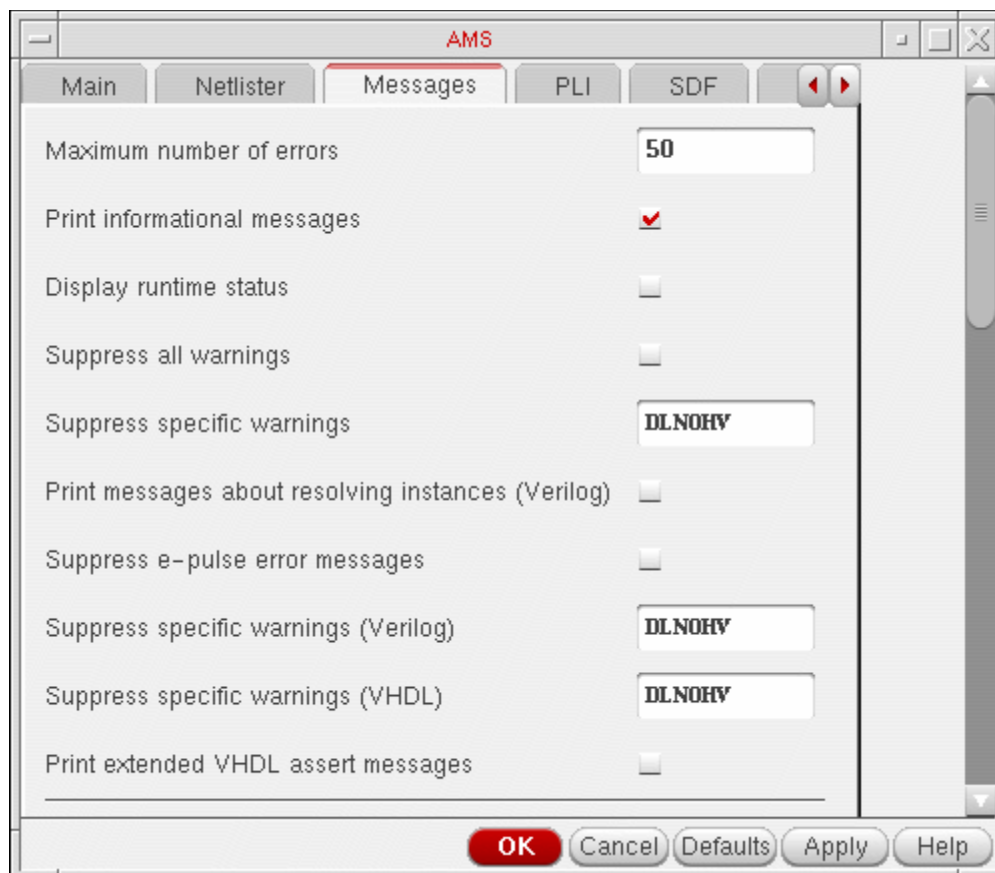
However, if the *Print definition of leaf cells in digital hierarchy* option is selected, the mixed signal netlister prints the empty standard cell definitions in the digital netlist.

The **Only include instances of these physical cells** field allows you to specify the physical instances that needs to be printed in the netlist. If you do not specify a cell name, then all the physical instances are printed in the netlist.

You can also use * wild card as a suffix when specifying cell names, such as `FILL*`, `MFILL*`. The physical instances of cells matching the specified pattern are printed in the netlist. However, the other physical-only instances are ignored.

The **Netlist macro instances from these libraries** field allows you to specify the libraries from the `cds.lib` file. The instances of analog schematic cells from the specified libraries are considered hierarchial instances and are netlisted hierarchically by the mixed-signal netlister.

Select the *Messages* tab.



Using the *Messages* tab, you can control the messages for ncvlog, ncelab and ncsim and irun. In this tab you can specify the following options:

Maximum number of errors: Number of error messages for ncvlog, ncelab and ncsim.

Print informational messages: If selected, displays the message from the tool.

Display runtime status: If selected, displays the runtime status.

Suppress all warnings: Suppresses all warnings. If this field is selected, the suppress all warnings field is disabled.

Suppress specific warnings: Suppresses warnings with a code.

Print Messages about resolving instances: This option prints informative messages during execution.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

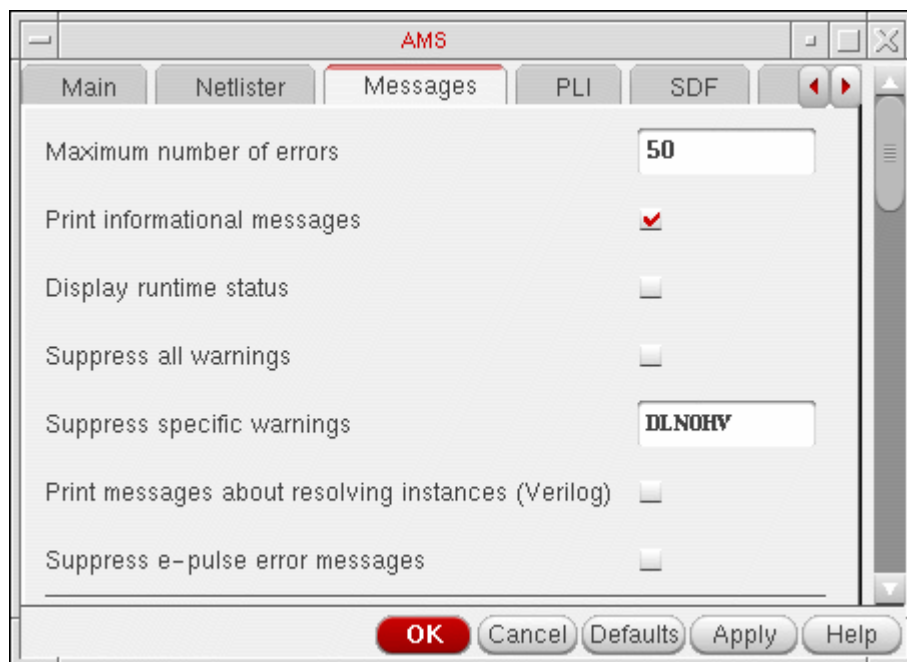
Suppress e-pulse error messages: This option suppresses pulse control error messages.

Suppress specific warnings (VHDL): This option suppresses warnings of the specified code.

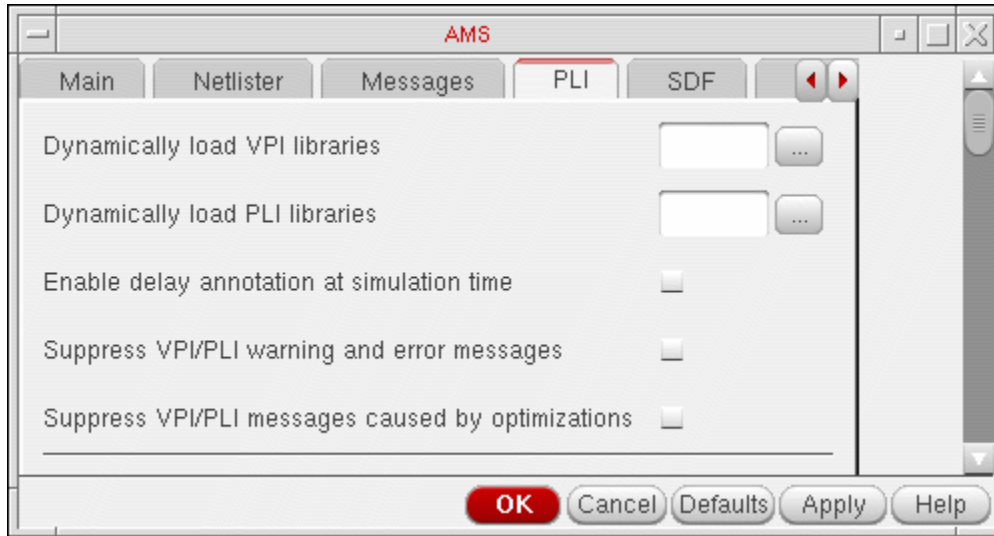
Suppress specific warnings (Verilog): This option suppresses warnings of the specified code.

Print extended VHDL assert messages: This option displays VHDL assert messages with additional information specifying the location in code from where the function or procedure is being called.

For the *OSS-based netlister with irun* netlister mode, you can specify the options that are displayed in the *Messages* tab as shown below:



Select the *PLI* tab.



In the *PLI* tab, you can specify the following options:

Dynamically load VPI libraries: Dynamically loads the specified VPI application.

Dynamically load PLI libraries: Dynamically loads the specified PLI application.

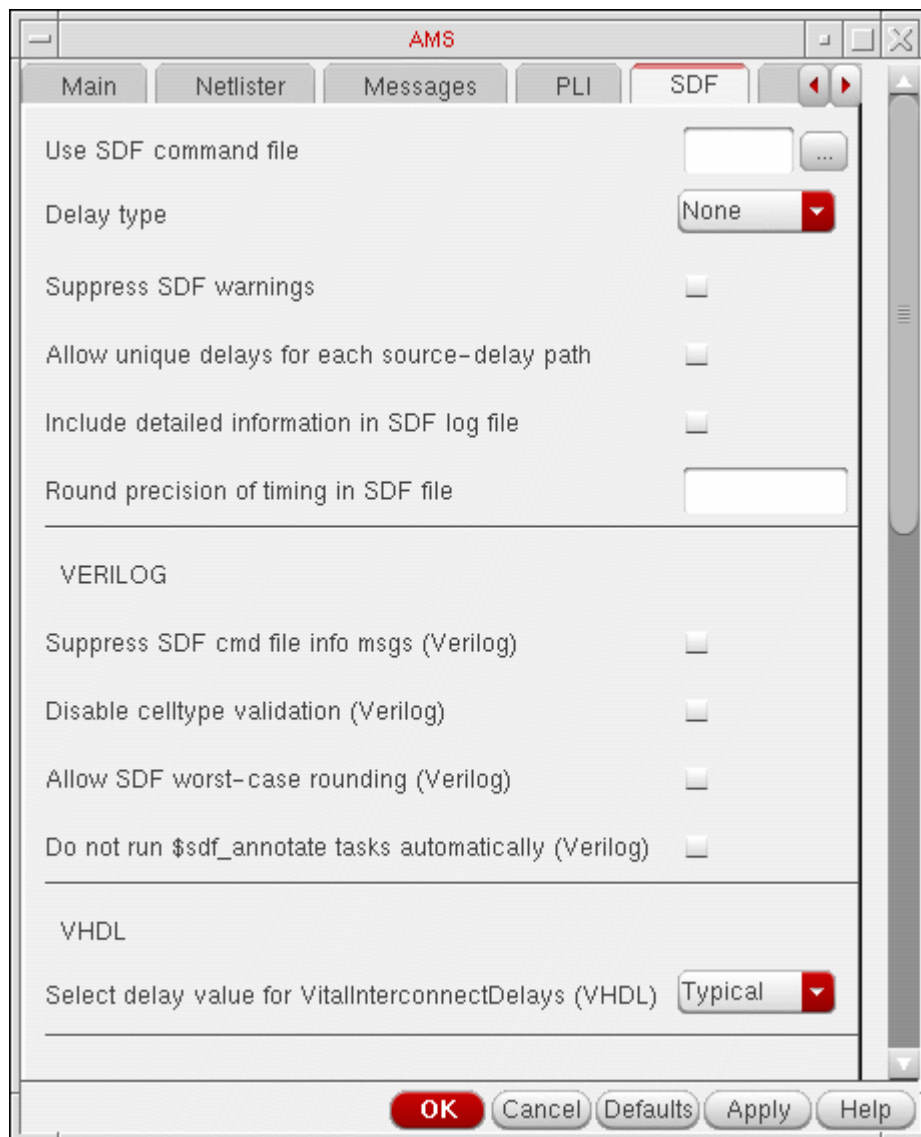
Enable delay annotation at simulation time: This option disables the optimization in the simulator that take delays into account. Use this option if you intend to modify delays at simulation time. Using this option sets the default access to simulation objects to read/write when the design is elaborated.

Suppress VPI/PLI warning and error messages: Disables printing of PLI warning and error messages.

Suppress VPI/PLI messages caused by optimizations: Prints a warning message only the first time that a PLI read, write, or connectivity access violation is detected.

Note: The options in the *PLI* tab are same for the *Cellview-based netlister with ncvlog, ncelab, ncsim* and the *OSS-based netlister with irun* netlister modes

Select the *SDF* tab.



In the *SDF* tab, you can specify the following options:

Use SDF command file: Specify the command file to be used.

Delay type: Enables you to select the Delay type. By default, it is set to None.

Suppress SDF warnings: Disables the SDF warnings.

Allow unique delays for each source-delay path: Enables or disables unique delays for each source-delay path.

Include detailed information in SDF log file: Allows you to print information in SDF log file.

Round precision of timing in SDF file: Allows for precision of timing in SDF file.

In VERILOG section, you can specify the following options:

Suppress SDF cmd file info msgs (Verilog): Allows printing of command file information messages.

Disable celltype validation (Verilog): Enable or disable celltype validation.

Allow SDF worst-case rounding (Verilog): Allows for worst-case rounding of SDF.

Do not run \$sdf_annotate tasks automatically (Verilog): Allows you to disable the automatic sdf_annotation tasks.

In VHDL section, you can specify the following option:

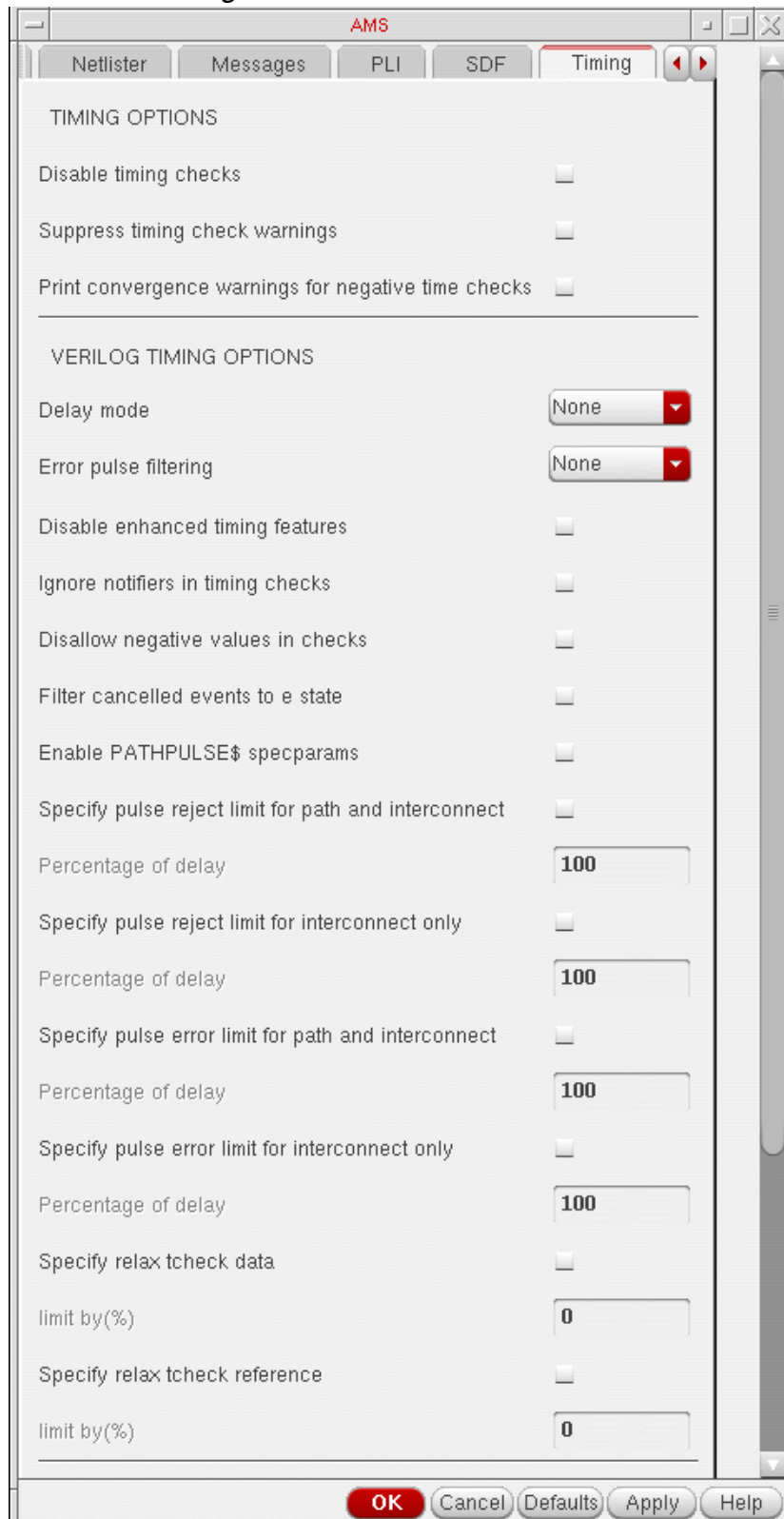
Select delay value for VitalInterconnectDelays (VHDL): Specifies minimum or maximum delay value to be used during VITAL SDF annotation.



Virtuoso Analog Design Environment L User Guide

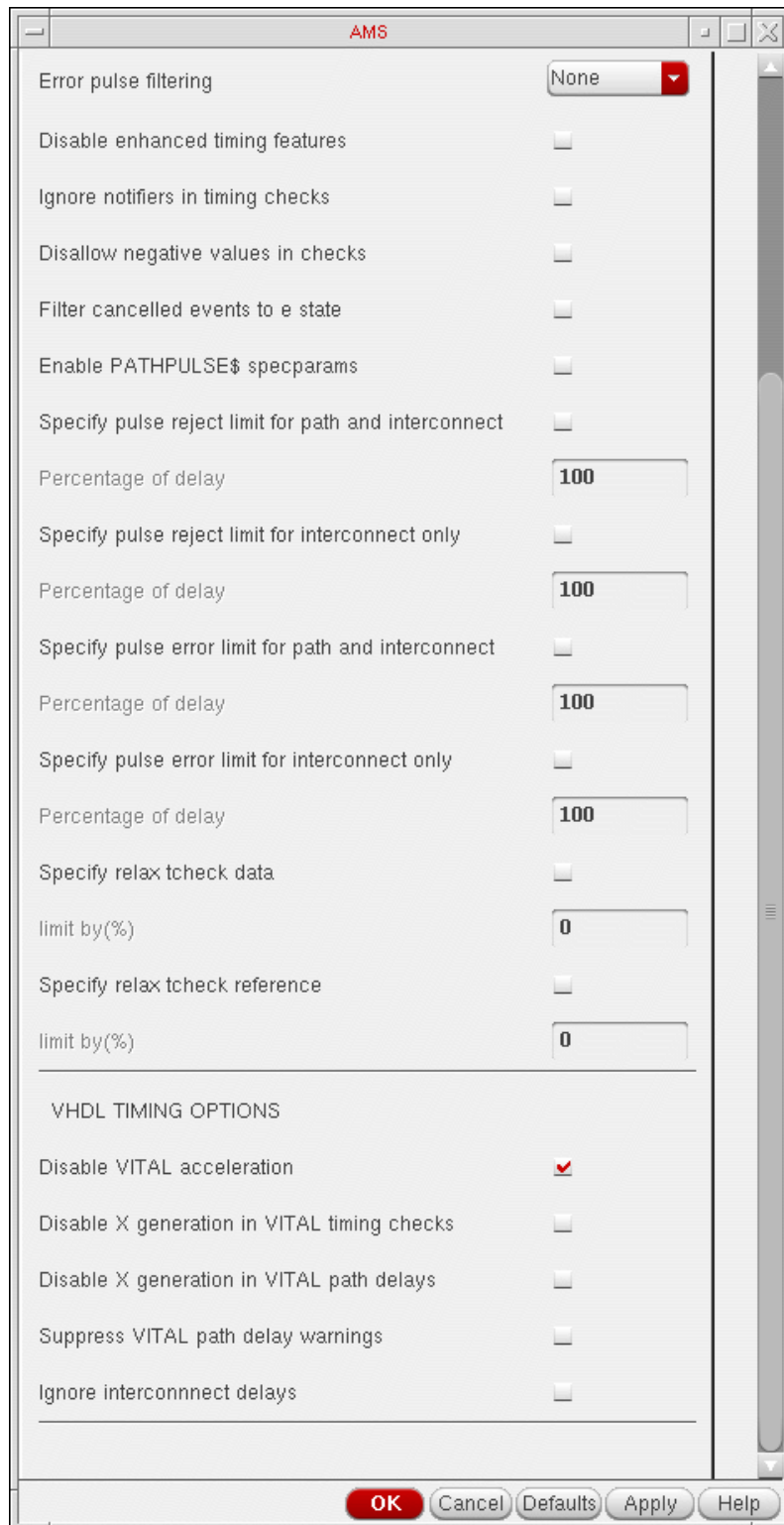
Running a Simulation

Select the *Timing* tab.



Virtuoso Analog Design Environment L User Guide

Running a Simulation



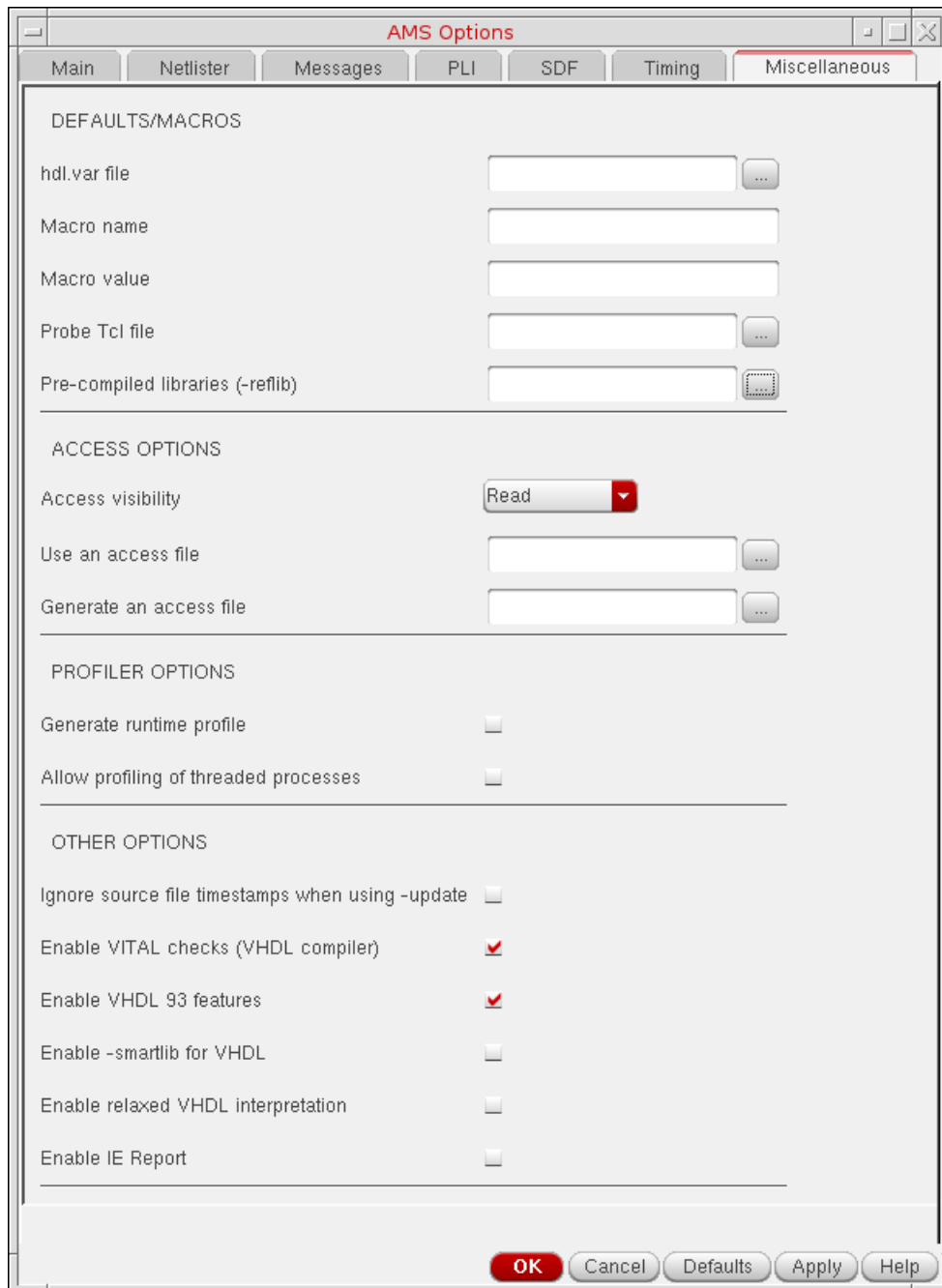
Virtuoso Analog Design Environment L User Guide

Running a Simulation

In the *Timing* tab, you can specify the timing options for VERILOG and VHDL.

Note: For the *OSS-based netlister with irun* netlister mode, the option for VHDL is not displayed in *Timing* tab.

Select the *Miscellaneous* tab.



Virtuoso Analog Design Environment L User Guide

Running a Simulation

In the Miscellaneous tab, you can specify INCLUDE options used by *ncvlog*, ACCESS options used by *ncelab* and PROFILER options used by *ncsim*. In addition, you can specify few other VHDL options used by *ncvhdl*.

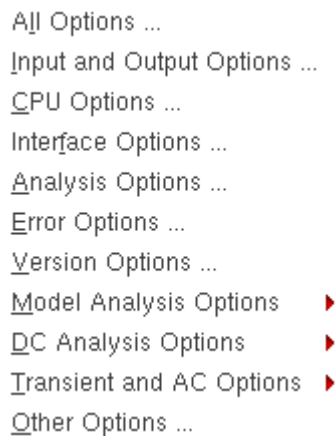
For details, refer to the [Virtuoso AMS Environment User Guide](#).

Note: Starting with IC6.1.6 ISR6, the AMS Unified Netlister (AMS UNL) has been introduced. For the AMS UNL flow, a new option *Display hdl.var file used by irun/simulator* has been added that enables you to view the `hdl.var` file settings.

For more information, see [AMS ADE Unified Netlisting Flow guide](#).

HspiceD Options

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



These options can be used to modify various aspects of the simulation, including output types, accuracy, speed, and convergence. For details about the Analog Options, refer to [HSPICE/SPICE Interface Reference](#).

Model Analysis Options

The *Model Analysis Options* have been grouped as follows:



Virtuoso Analog Design Environment L User Guide

Running a Simulation

- Click *Model Analysis Options – General Options* to specify DCAP, HIER_SCALE, MODMONTE, MODSRH, SCALE, TNOM using the *Hspice General Model Options* form.
- Click *Model Analysis Options – Mosfet Control Options* to specify CVTOL, DEFAD, DEFAS, DEFEL, DEFNRD, DEFNRS, DEFPD, DEFPS, DEFW, SCALM, WL using the *Hspice Mosfet Control Options* form.
- Click *Model Analysis Options – Inductor Options* to specify GENK, KLIM using the *Hspice Inductor Options* form.
- *Model Analysis Options – BJT and Diode Options* to specify EXPLI using the *Hspice BJT and Diode Options* form.

DC Analysis Options

The *DC Analysis Options* have been grouped as follows:

Accuracy Options ...
Matrix Options ...
Input and Output Option ...
Convergence Options ...

- Click *DC Analysis Options – Accuracy Options* to specify ABSH, ABSI, ABSMOS, ABSVDC, DI, KCLTEST, MAXAMP, RELH, RELI, RELMOS, RELV, RELVDC using the *Hspice DC Accuracy Options* form.
- Click *DC Analysis Options – Matrix Options* to specify ITL1, ITL2, NOPIV, PIVOT, PIVREF, PIVREL, PIVTOL using the *Hspice Matrix Options* form.
- Click *DC Analysis Options – Input and Output Option* to specify CAPTAB, DCCAP, VFLOOR using the *HspiceDC Input and Output Options* form.
- Click *DC Analysis Options – Convergence Options* to specify CONVERGE, CSHDC, DCFOR, DCHOLD, DCON, DCSTEP, DV, GMAX, GMINDC, GRAMP, GSHUNT, ICSWEEP, ITLPTRAN, NEWTOL, OFF, RESMIN using the *HspiceConvergence Options* form.

Transient and AC Options

The *Transient and AC Analysis Options* have been grouped as follows:

Accuracy Options ...
Speed Options ...
Timestep Options ...
Algorithm Options ...
Input and Output Options ...

- Click *Transient and AC Options – Accuracy Options* to specify ABSH, ABSV, ACCURATE, ACOUT, CHGTOL, CSHUNT, DI, GMIN, GSHUNT, MAXAMP, RELH, RELI, RELQ, RELV, RISETIME, TRTOL using the *Hspice Transient and AC Options* form.
- Click *Transient and AC Options – Speed Options* to specify AOTPSTOP, BKPSIZ, BYPASS, BYTOL, FAST, ITLPZ, MBYPASS, TRCON using the *Hspice Speed Options* form.
- Click *Transient and AC Options – Timestep Options* to specify ABSVAR, DVDT, FS, FT, IMAX, IMIN, ITL5, RELVAR, RMAX, RMIN, SLOPETOL, TIMERES using the *Hspice Timestep Options* form.
- Click *Transient and AC Options – Algorithm Options* to specify DVTR, IMAX, IMIN, LVLTIM, MAXORD, METHOD, MU, PURETP, TRCON using the *Hspice Algorithm Options* form.
- Click *Transient and AC Options – Input and Output Options* to specify INTERP, ITRPRT, MEASFAIL, MEASSORT, PUTMEAS, UNWRAP using the *Hspice Transient Input and Output Options* form.

All the *Analog Options* mentioned are supported by the *HSPICE* version 2003.3. Ensure that you are using a compatible version of *HSPICE*.

Other Options

Click *Other Options* to specify HSPICE command-line options in the *ADDITIONAL_ARGUMENTS* field.

About the OSS-based AMS Netlister

AMS Designer Virtuoso Use-Model (AVUM) provides two Verilog-AMS netlisters—the Cellview-based Verilog-AMS netlister and the OSS-based Verilog-AMS netlister.

OSS (Open Simulation System) is the underlying framework used by the Spectre, SpectreVerilog (IC6.1.6 only), UltraSimVerilog (IC6.1.6 only), and Verilog netlisters integrated into Virtuoso Analog Design Environment (ADE).

The Cellview-based netlister has been the legacy netlister in AVUM, whereas the OSS-based netlister was introduced in IC 5.1.41 USR4. The Cellview-based netlister needs ams CDF simulation information (siminfo) in the PDKs, whereas the OSS-based netlister needs the spectre siminfo.

Benefits of OSS-based AMS Netlister

Some of the important benefits of OSS-based netlister are:

- It supports simulation using the new `irun` command of the Virtuoso AMS Designer simulator. For more information about the `irun` command, see the *irun User Guide*.
- The `runams` command provides command line support for netlisting the design using the OSS-based AMS netlister and running simulation using `irun`. For more information, see [Appendix D, “Using the runams Command.”](#)
- It needs spectre CDFs and spectre netlist procedures to netlist instances of analog primitives. If you are using the SpectreVerilog (IC6.1.6 only) or UltraSimVerilog (IC6.1.6 only) simulator, your PDKs will have spectre siminfo. Hence, it is easier for you to migrate to AVUM using the OSS-based netlister.
- It does not create files in the 5X library structure. Therefore, you do not require writable master design-libraries and PDKs, or explicit or implicit TMP directories for libraries.
- As the OSS-based netlister does not use the 5X library structure, compilation is faster.
- It generates a single netlist. If the simulator reports an error, you can edit the netlist file and simulate it again. Additionally, its (error, warning and information) messages are consistent with those of the SpectreVerilog (IC6.1.6 only) netlister. For example, if you understand the messages from the Spectre netlister, you will understand the messages from this netlister also, as both are OSS-based netlisters.
- It uses the same config view that is used by the SpectreVerilog (IC6.1.6 only) and UltraSimVerilog (IC6.1.6 only) netlisters.
- The OSS-based netlister supports VHDL modules in your design.

Note the following:

- VHDL modules are supported only at the leaf level.
- Instance binding is not supported for VHDL modules. For more information about instance binding, see the [Cadence Hierarchy Editor User Guide](#).

Things to Know When Using the OSS-based Netlister

This section describes a few things to know when using the OSS-based netlister.

- [Text-only views, which do not have the .oa file must be imported into the Virtuoso database format](#) on page 355
- [Switch and stop view lists of the config view must be correct](#) on page 356
- [Instance level view binding in Hierarchy Editor and the 'uselib compiler directive](#) on page 356
- [Should an instance that is bound to a symbol view be netlisted in Spectre syntax \(using spectre siminfo\) or in Verilog syntax?](#) on page 357
- [Netlisting of inherited connections](#) on page 358

Text-only views, which do not have the .oa file must be imported into the Virtuoso database format

OSS, the underlying framework, requires the text-only views in the design to be imported into the Virtuoso database, and have a .oa file in the view. OSS needs these files to successfully traverse the design hierarchy.

Because of this requirement from OSS, the OSS-based netlister requires the text-only views to be imported into the Virtuoso database format. You can use any one of the following ways to do this:

Note: You can use Verilog or VHDL text files that are not netlisted by specifying them in the *Library files* or *Library directories* fields in Main tab of the [AMS Options](#) form.

- Use the Update Text Views form

For more information, see [Updating Text Views that do not have Virtuoso Database Information Using the Update Text Views Form](#) on page 361.

Note: Use the Update Text Views form if you were previously using the Cellview-based netlister and you are unsure which of the cells do not have the Virtuoso database (.oa) file, or have a lot of views that do not have the database.

- Import text files using the Verilog In or VHDL In utility.

For more information about importing Verilog text files using Verilog In, see the [Verilog In for Virtuoso Design Environment User Guide and Reference](#). For more information about importing VHDL text files using VHDL In, see the [VHDL In for Virtuoso Design Environment User Guide and Reference](#).

- Use other methods for updating text views that do not have design database information. For more information, see [Other Methods for Updating Text Views that do not have Virtuoso Database Information](#) on page 363.

Switch and stop view lists of the config view must be correct

When a design is netlisted, the instance of a cell in the design needs to be associated with its corresponding schematic or simulator primitive. This process is called switching views. The list of valid views to be used for switching is specified as the switch view list in the config view of the design.

A view that is the most detailed description desired for simulation is called a stop view. The list of valid stop views is specified as the stop view list in the config view of the design. If the view located when switching views corresponds to one specified in the stop view list, the expansion process for the instance is stopped, and the connectivity information for the instance is printed in the netlist file. If the view does not correspond to a stop view, the expansion process continues.

Note: You can use the Hierarchy Editor to specify the switch view and stop view lists for your configuration. For more information, see the [Cadence Hierarchy Editor User Guide](#).

Instance level view binding in Hierarchy Editor and the `'uselib` compiler directive

View binding at the instance level allows you to explicitly specify different views of a cell for netlisting different instances of the same cell. You use Hierarchy Editor to specify view-binding at the instance-level. For more information, see the [Cadence Hierarchy Editor User Guide](#).

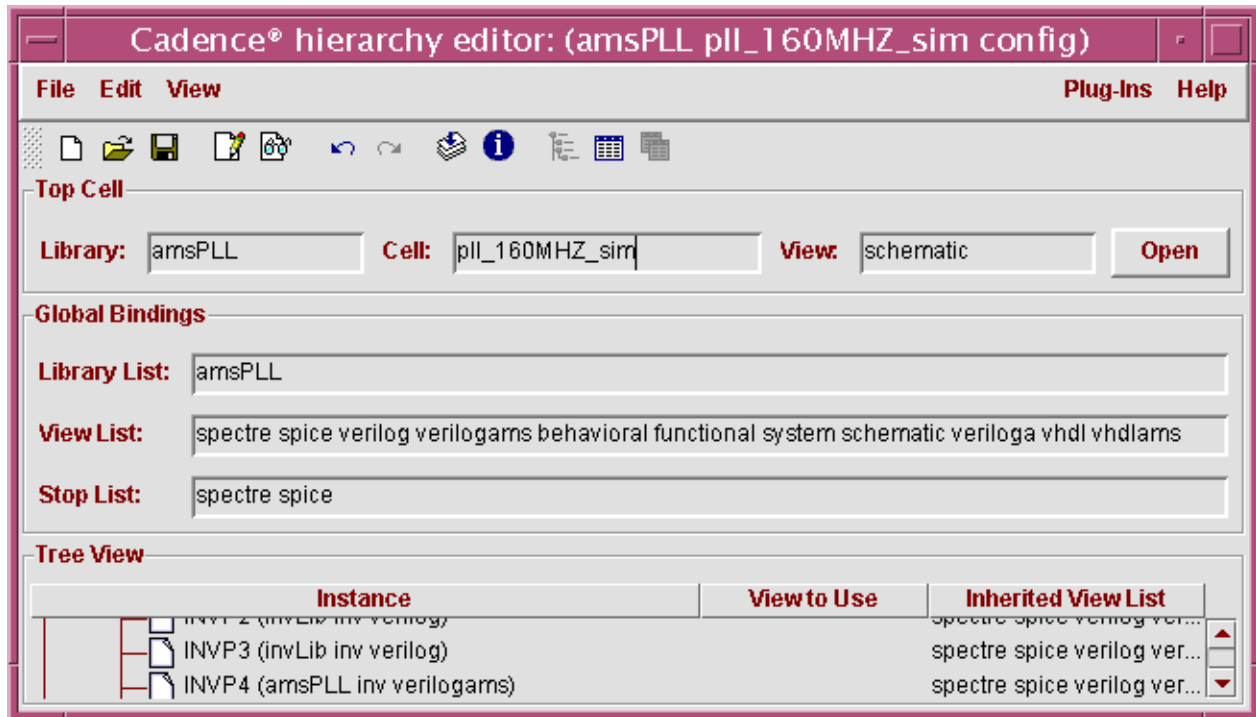
If two or more libraries in your design have cells with the same name, and the instances of the cells are bound to a `verilog` or `verilogams` view, the `'uselib` compiler directive is printed in the netlist to ensure that the correct source file is chosen for the instances.

For example, the design configuration in the following figure has instance `INV3` of the `inv` cell from the `invLib` library and instance `INV4` of the `inv` cell from the `amsPLL` library.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Instance INVP3 is bound to the verilog view and instance INVP4 is bound to the verilogsams view.



To ensure that the correct source file is used for instances INVP3 and INVP4, the following `\uselib` compiler directives are printed in the netlist.

```
\uselib file=/usr2/myDdesign/invLib/inv/verilog/verilog.v
inv INVP3 ( INV_3_P, INV_2_P);
\uselib
\uselib file=/usr2/myDdesign/amsPLL/inv/verilogams/verilog.vams
inv INVP4 ( INV_4_P, INV_3_P, Vring);
\uselib
```

For more information about the `\uselib` compiler directive, see the *NC-Verilog Simulator Help*.

Should an instance that is bound to a symbol view be netlisted in Spectre syntax (using `spectre siminfo`) or in Verilog syntax?

By default, if an instance in your design configuration is bound to a symbol view, the instance is considered digital, and hence netlisted in Verilog syntax.

If you want a specific instance to be considered analog, and netlisted in Spectre syntax using `Spectre siminfo`, do the following:

1. Open your configuration in Hierarchy Editor.
2. Choose *View – Tree*.
3. Right-click on the instance and choose *Set Instance View – Spectre*.

If you want all instances that are bound to a symbol view to be considered analog, do the following:

1. Choose *Simulation – Options – AMS Simulator*.
The AMS Options form appears.
2. Click the Netlister tab.
3. Add `symbol` in the *Netlist using spectre CDF simInfo* field.

Netlisting of inherited connections

Inherited connections allow you to selectively override the global signals in your design. This allows you to use multiple power supplies in a single design. If you want to implement separate power supplies (analog and digital, for example, or +3 V and +5 V) in a hierarchical design, you can assign net expressions to those global signals whose defaults you might want to override. Then you can use `netSet` properties to specify the new values of the signals. For more information about inherited connections, see the *Inherited Connections Flow Guide* and the *Virtuoso Inherited Connections Tutorial*.

The OSS-based netlister uses Verilog AMS attributes to netlist inherited connections. For example, the net expression:

```
[@xground:%:vdd!]*
```

will be written in the netlist as:

```
wire (*  
integer ihn_conn_prop_name = "xground";  
integer ihn_conn_def_value = "cds_globals.\\vdd! "; *)  
cdsNet0;
```

The `netSet` properties are netlisted as `cds_net_set` attributes and evaluated by the elaborator. For example, the following `netSet` properties

Property	Value
vdd	3.3v!
xground	[@new_ground:%:gnd5!]

Virtuoso Analog Design Environment L User Guide

Running a Simulation

will be netlisted as:

```
(*
integer cds_net_set
[0:1] = {"xground", "vdd"};
integer xground[0:1] = {"new_ground", "cds_globals.\\gnd5!" } ;
integer vdd = "cds_globals.\\3.3v!" ;
*)
```

Important

To use inherited connections with Verilog A modules, you must add an additional port and specify the inherited connection attributes for each inherited connection.

For example, in the Verilog A definition shown below:

```
module inhconn_spectre(vdd_out, gnd_out, global_vdd, global_gnd);
    inout vdd_out, gnd_out;
    input (* integer inh_conn_prop_name="global_vdd_prop";
           integer inh_conn_def_value="cds_globals.\\vdd! "; *) global_vdd;
    input (* integer inh_conn_prop_name="global_gnd_prop";
           integer inh_conn_def_value="cds_globals.\\gnd! "; *) global_gnd;
    electrical vdd_out, gnd_out, global_vdd, global_gnd;
    ...
endmodule
```

- ❑ The ports `global_vdd` and `global_gnd` are extra ports added to use inherited connections.
- ❑ The inherited connection attributes for the `global_vdd` port are specified as

```
(* integer inh_conn_prop_name="global_vdd_prop";
integer inh_conn_def_value="cds_globals.\\vdd! "; *)
```

and the inherited connection attributes for the `global_gnd` port is specified as

```
(* integer inh_conn_prop_name="global_gnd_prop";
integer inh_conn_def_value="cds_globals.\\gnd! "; *)
```

Where the `inh_conn_prop_name` attribute specifies the name of the `netSet` property, and the `inh_conn_def_value` attribute specifies the value of the `netSet` property.

When the design is netlisted, the inherited connection attributes are printed in the netlist as:

```
wire (*
integer inh_conn_prop_name = "global_gnd_prop";
integer inh_conn_def_value = "cds_globals.\\gnd! ";
*)
\global_gnd_prop_gnd! ;
wire (*
integer inh_conn_prop_name = "global_vdd_prop";
integer inh_conn_def_value = "cds_globals.\\vdd! ";
*)
\global_vdd_prop_vdd! ;
```

and the I7 instance of the `inhconn_spectre` cell is netlisted as:

```
inhconn_spectre I7 ( vdd_va_spectre , gnd_va_spectre , \global_vdd_prop_vdd!  
, \global_gnd_prop_gnd! );
```

Creating Simulation Scripts for `irun`

Use the *Simulation scripts for `irun`* option to export the simulation files from the netlist directory. This option copies all the files required for running the simulation, and generates the `irunScript` file in the export directory. You can run the `irunScript` file to run the AMS simulation from the command line.

Note: The *Simulation scripts for `irun`* option copies all the files from the netlist directory except the definition, model, and stimulus files.

To export the simulation files,

1. Choose *Tools – Export – Simulation scripts for `irun`*.

The *Export Irun Controls* form appears.



2. Choose *Write Absolute paths* option to use the absolute paths in the `textInput`s file for the text view.

Choose *Copy Files* option to use relative paths in the `textInput`s file. This option copies all the files, including the sub-directories, referred in the `textInput`s file from the cellview directory to the export directory.

Note: The `textInput`s file contains the text files for text views in the design.

3. Click *OK*.

Updating Text Views that do not have Virtuoso Database Information Using the Update Text Views Form

The OSS-based netlister currently does not support designs with text-only views—text views that do not contain the Virtuoso database. To create the database for such views, do the following:

Important

Depending on the size of your configuration or library, the process of adding the design database information can take a long time to complete.

1. Ensure that you have write permissions to the libraries and PDKs used in your design.
2. Choose *Tools – Update Text Views*.

The *Update Text Views* form appears.



ams1: Update Text Views

Update text views that do not have the Virtuoso database.

Create database for config text views

Update Mode incremental all

Config

Library

Cell

View

Text views

Library

Cell(s)

View(s)

Virtuoso Analog Design Environment L User Guide

Running a Simulation

You can use this form to update text-only views of type Verilog, Verilog A, Verilog AMS, VHDL or VHDL-AMS.

3. Do one of the following:

To update all the text views in your configuration that do not have design database information, do the following:

a. Select the *Config* option.

b. Click the *Browse* button in the *Config* group box.

The *Library Browser – Update Text Views* form appears.

c. Select the library and cell in which the config view exists, then select the config view.

Note: You can also enter the library, cell and config view name in the *Library*, *Cell* and *View* fields.

d. Click *Close* to close the *Library Browser – Update Text Views* form.

To update all the views in a library, do the following:

a. Select the *text views* option.

b. From the *Library* cyclic field, select the library in which the views exist.

To update all the views in a cell, do the following:

a. Select the *For Cell Views* option.

b. From the *Library* cyclic field, select the library in which the cell exists.

c. In the *Cell(s)* field, enter the name of the cell, or click the *Browse* button to select the cell.

Note: To update the views in more than one cell, enter the cell names separated by spaces in the *Cell(s)* field.

To update a specific view, do the following:

a. Select the *For Cell Views* option.

b. From the *Library* cyclic field, select the library in which the cell exists.

c. In the *Cell(s)* field, enter the name of the cell in which the view exists, or click the *Browse* button to select the cell.

Note: To update the views in more than one cell, enter the cell names separated by spaces in the *Cell(s)* field.

- d. In the *View(s)* field, enter the name of the view, or click the *Browse* button to select the view.

Note: To update the more than one view, enter the view names separated by spaces in the *View(s)* field.

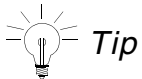
4. Click *OK* or *Apply*.

The views that do not have design database information are updated.



Caution

Once you click *OK* or *Apply*, you cannot stop the process of updating the text-only views.



Tip

After specifying the information in the *For Config* or *For Cell Views* group boxes, click the *Display* button to view the list of views that do not have design database information.

Other Methods for Updating Text Views that do not have Virtuoso Database Information

You can also use the following methods for updating text views that do not have design database information:

- Use the *Design – Create Cellview – From Cellview* command in the schematic editor.
 - a. In the schematic editor, choose *Design – Create Cellview – From Cellview*.
The Cellview From Cellview form appears.
 - b. Specify the library name in the *Library Name* field.
 - c. Specify the cell name in the *Cell Name* field.
 - d. In the *Tool / Data Type* cyclic field select the type of the text design you want to import.
For example, choose *Verilog-Editor* to import a Verilog text file.
 - e. Click *OK*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

A template file is opened in a text editor. For example, if you had selected *Verilog-Editor* in [step d](#), a template `verilog.v` file with the following text is opened in a text editor:

```
//Verilog HDL for "mylib", "toptext" "functional"
module toptext ( );
endmodule
```

f. Open your text design in a text editor and paste its contents in the template file. Ensure that you retain the module name in the template file.

g. Save the file and close the text editor.

■ Use the *File – New – Cellview* command in the CIW.

a. In CIW, choose *File – New – Cellview*.

The *Create New File* form appears.

b. In the *Library Name* cyclic field, select the library in which you want to create a cell for the text design.

c. In the *Cell Name* field, enter the name of the cell you want to create for the design—for example, `toptext`.

d. In the *View Name* field, enter the name of the view you want to create—for example, `functional`.

e. In the *Tool* cyclic field, select the type of the text design you want to import.

For example, choose *Verilog-Editor* to import a Verilog text file.

f. Click *OK* to create the cell for the design.

A template file is opened in a text editor. For example, if you had selected *Verilog-Editor* in [step e](#), a template `verilog.v` file with the following text is opened in a text editor:

```
//Verilog HDL for "mylib", "toptext" "functional"
module toptext ( );
endmodule
```

g. Open your text design in a text editor and paste its contents in the template file. Ensure that you retain the module name in the template file.

h. Save the file and close the text editor.

A message box appears prompting you to create a symbol for the cell.

- i. Click *OK*.
- Open the cell for editing in Library Manager and save it.
 - a. From the Library Manager, open the cell for editing.

The Verilog or VHDL file for the cell is opened in a text editor.
 - b. Save the file and close the text editor.

A message box that appears prompting you to create the symbol for the cell.
 - c. Click *OK*.

Choosing the AMS Netlister

If you are new to AMS Designer Virtuoso Use-Model (AVUM), choose the OSS-based netlister.

Note: If you are using the OSS-based netlister, use the AMS template in Hierarchy Editor to create the configuration for your top-level design.

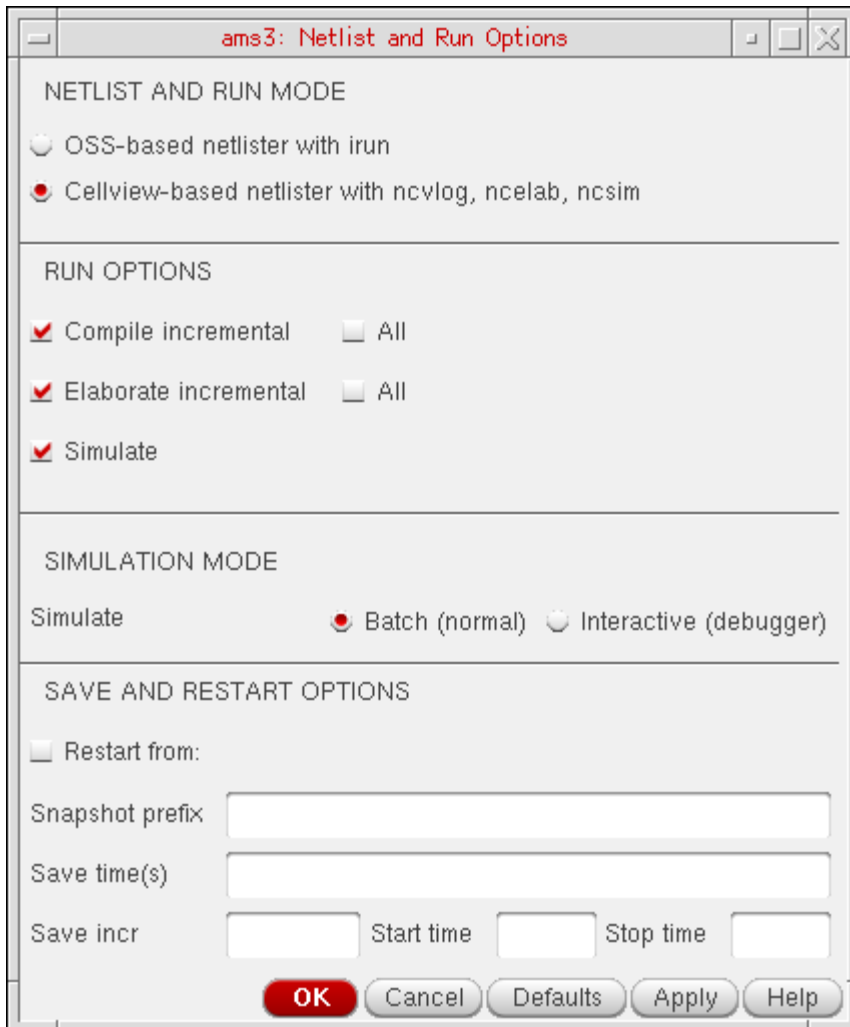
To select the AMS netlister, do the following:

1. Choose *Simulation – Netlist and Run Options*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Netlist and Run Options form appears.



2. Select the netlister and run mode.

The *Cellview-based netlister with ncvlog, ncelab, ncsim* run mode is selected by default.

Note: Starting with IC6.1.6 ISR6, the AMS Unified Netlister (AMS UNL) has been introduced. AMS UNL can be enabled by setting the `AMS_UNL` environment variable to `YES`. For more information on AMS UNL, see [AMS ADE Unified Netlisting Flow guide](#).

3. Specify the run options for the netlister.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

- ❑ If you select *Compile Incremental*, only the modules that have been edited since the last compilation will be compiled. If you select *All*, all the modules will be compiled.
- ❑ If you select *Elaborate Incremental*, only the modules that have been edited since last elaboration will be elaborated. If you select *All*, all the modules will be elaborated.

Note: There is no option in *irun* that will allow only elaboration to work.

- ❑ If you select *Simulate*, the design will be simulated.

Note the following:

- You can choose to only compile, elaborate or simulate the design, although by default all three are selected as shown in the figure given above. For example, if you select only *Elaborate*, the *Simulation – Netlist* command only elaborates the design. Compilation and elaboration can be incremental or for the whole design together.
- A simulation will fail if you choose both *Compile* and *Simulate*. You would need to either deselect *Simulate* or select *Elaborate* as well.
- ❑ Select *Clean snapshot and pak files* to delete any existing simulation snapshot and *.pak* files before running the simulation.
- ❑ Select the *Compile VerilogA as Verilog-AMS* option to debug the VerilogA views using SimVision, similar to Verilog-AMS views.

Note: This option is available for the *irun* run mode only.

4. Select the simulation mode as batch or interactive.

When the host mode is distributed, *Batch* mode is the default mode. In this mode, all signal plotting occurs in Virtuoso Visualization and Analysis XL.

The *Interactive* mode launches the complete SimVision debug environment on top of current design. For more information, see [Using the SimVision Debugger](#) on page 389.

Note: When host mode is distributed, the radio button for *Interactive* mode will be disabled and batch mode will be enabled (only if Interactive mode was selected before).

5. (Optional) In the SAVE AND RESTART OPTIONS group box specify the options for saving simulation snapshots or restarting simulation from an existing snapshot. For more information, see [Saving and Restarting an AMS Designer Simulation Run](#) on page 368.

6. Click *OK*.

Saving and Restarting an AMS Designer Simulation Run

You can save simulation snapshots at specified timepoints during a transient analysis simulation run using the AMS Designer simulator and later restart the simulation from a specific snapshot. Saving simulation snapshots is especially useful for large simulations where you might want to save the simulation state at regular intervals because you can:

- Save time by restarting simulation from a specific snapshot instead of restarting simulation from time 0.
- Quickly debug simulation results by verifying only the snapshots in which you are interested.
- Restart from a known good point if simulation fails at any time due to a system failure, power outage, or other issues.

Note: Saving snapshots and restarting simulation from a specific snapshot is supported only for transient analysis runs and if you are using `spectre` or `ultrasim` as the solver. This is not supported for other analyses or other solvers.

For more information, see the following topics:

- [Saving Simulation Snapshots](#) on page 368
- [Restarting Simulation from a Saved Snapshot](#) on page 372

Saving Simulation Snapshots

To save simulation snapshots, do the following:

1. Choose *Simulation – Netlist and Run Options*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Netlist and Run Options form appears.

The screenshot shows the 'ams3: Netlist and Run Options' dialog box. It is organized into four main sections:

- NETLIST AND RUN MODE:** Two radio buttons are present. The top one is 'OSS-based netlister with irun' (unselected). The bottom one is 'Cellview-based netlister with ncvlog, ncelab, ncsim' (selected).
- RUN OPTIONS:** Three checked checkboxes are shown: 'Compile incremental', 'Elaborate incremental', and 'Simulate'. Each has an unchecked 'All' checkbox to its right.
- SIMULATION MODE:** A 'Simulate' label is followed by two radio buttons: 'Batch (normal)' (selected) and 'Interactive (debugger)' (unselected).
- SAVE AND RESTART OPTIONS:** A 'Restart from:' checkbox is unchecked. Below it are five input fields: 'Snapshot prefix', 'Save time(s)', 'Save incr', 'Start time', and 'Stop time'.

At the bottom of the dialog are five buttons: 'OK' (highlighted in red), 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. In the *Snapshot prefix* field, specify the prefix to be used in the names of snapshots.

For example, if you specify the prefix as `mySnap`, snapshots are saved with names such as `mySnap_40n`, where `40n` indicates the timepoint at which the snapshot was saved.

The default prefix is `snapshot`.

3. In the *Save time(s)* field, specify the timepoints at which snapshots need to be saved.

Note the following:

- Multiple values entered in this field should be separated by blank spaces.
- Do not use spaces between the time and the unit of time. The default unit of time is seconds.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

- ❑ To save a snapshot for a timepoint with a unique name, specify the time point in the following format:

(timePoint snapShotName)

For example, if you specify the timepoint as *(50n my50nSnapShot)*, a snapshot named *my50nSnapShot* is saved for timepoint *50n*.

4. In the *Save incr* field, specify the time interval at which snapshots need to be saved. From the start time (specified in the *Start time* field) till the stop time (specified in the *Stop time* field), snapshots are saved at the time interval specified in this field.

For example, if the start time is *40n*, stop time is *65n* and the time interval is *10n*, snapshots are saved for timepoints *40n*, *50n* and *60n*. A snapshot is not saved at timepoint *65n* (*Stop time*) because, after *40n* (*Start time*), snapshots are saved only at an interval of *10n* (*Save incr*).

Note: Do not use spaces between the time and the unit of time. The default unit of time is seconds.

5. In the *Start time* field, specify the timepoint from which snapshots must be saved.

Note: Do not use spaces between the time and the unit of time. The default unit of time is seconds.

6. In the *Stop time* field, specify the timepoint at which saving of snapshots must be stopped. Ensure that the specified stop time does not exceed the stop time specified for the transient analysis.

Note: Do not use spaces between the time and the unit of time. The default unit of time is seconds.

7. Click *OK*.

8. Run a simulation to save the simulation snapshots.

Example of Saving Simulation Snapshots



In the above example, the following four snapshots are saved at timepoints 20n, 40n, 50n and 60n.

- snap_20n
- snap_40n
- my50nSnapShot
- snap_60n

Note: A snapshot is not saved at timepoint 65n (*Stop time*) because, after 40n (*Start time*), snapshots are saved only at an interval of 10n (*Save incr*).

Restarting Simulation from a Saved Snapshot



You can restart from an existing snapshot if the following criteria are met:

- The snapshot was created using the same design that is currently being used.
- Only the options that do not affect the topology such as `reltol`, `abstol`, `errpreset`, `method` and `stop` in the Transient Options form were changed since the snapshot was created.
- The model files have not been changed since the snapshot was created.
- The solver is the same. For example, if a snapshot was saved when `spectre` was set as the solver, you must restart simulation using the snapshot only if `spectre` is currently set as the solver.

For more information about the changes that are allowed in the simulation setup, see the “Using the Save-and-Restart Feature” section in the *Virtuoso AMS Designer Simulator User Guide*.

To restart simulation from a saved snapshot, do the following:

1. Choose *Simulation – Netlist and Run Options*.

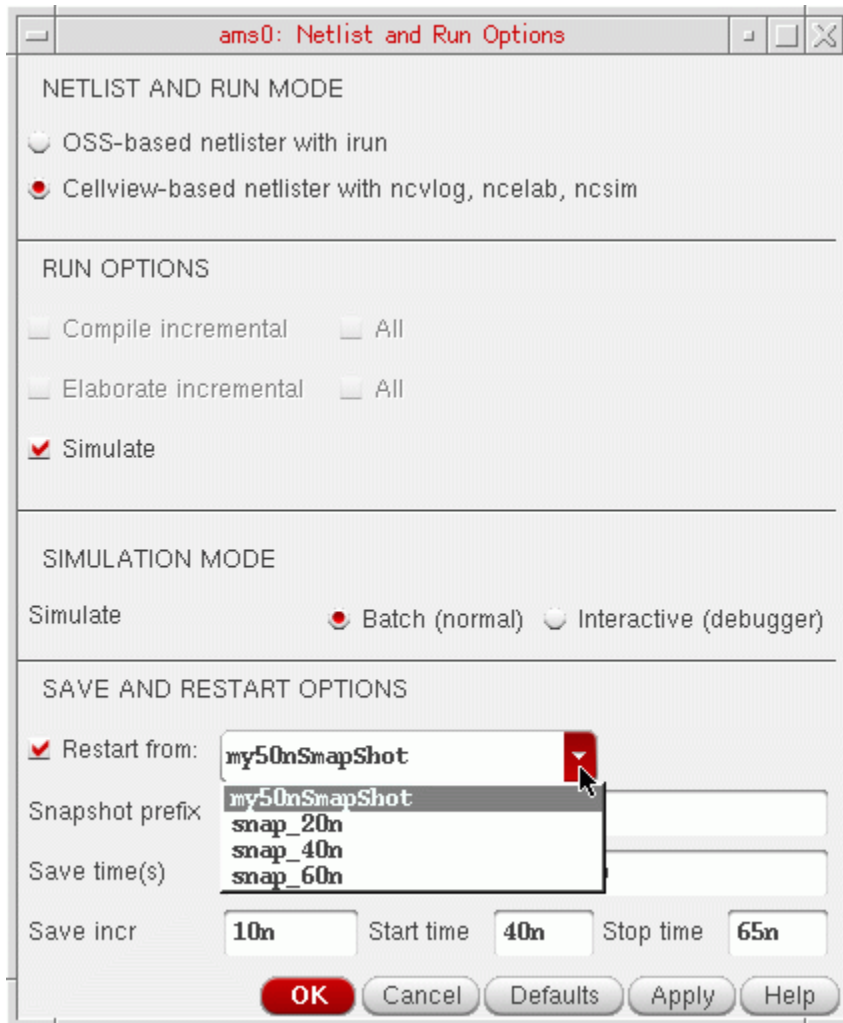
The Netlist and Run Options form appears.

2. Select the *Restart from* check box.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The snapshots that are saved from previous simulation runs are displayed in the *Restart from* cyclic field.



Note: When the *Restart from* check box is selected, the *Compile incremental* and *Elaborate incremental* fields are disabled because netlisting, compilation and elaboration are disabled when you restart simulation from a saved snapshot. This ensures that existing snapshots are not overwritten.

3. From the *Restart from* cyclic field, select the snapshot from which you want to restart the simulation.
4. (Optional) If you want to save more snapshots, modify the values in the *Snapshot prefix*, *Save time(s)*, *Save incr*, *Start time* and *Stop time* fields as required.

Virtuoso Analog Design Environment L User Guide

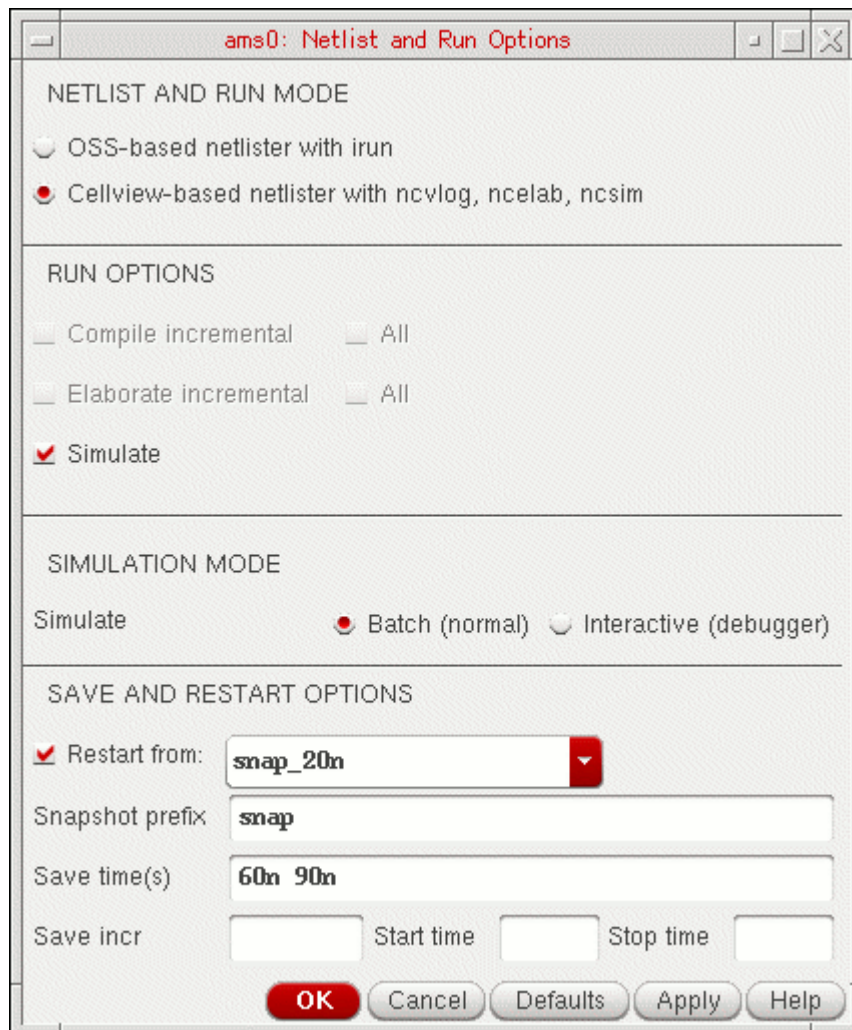
Running a Simulation

If you don't want to save more snapshots, delete the values, if any, in the *Snapshot prefix*, *Save time(s)*, *Save incr*, *Start time* and *Stop time* fields.

5. Click *OK*.
6. Run a simulation to restart the simulation from the snapshot specified in the *Restart from cyclic* field.

When the simulation completes, a waveform from time 0 to stop time is displayed, even though the simulation was started from a later snapshot, and the snapshots were saved during different simulation runs.

Example of Restarting a Simulation Using a Simulation Snapshot



Virtuoso Analog Design Environment L User Guide

Running a Simulation

The above figure shows an example of restarting the simulation from timepoint 20n using the snapshot named `snap_20n` created from a previous simulation run.

The *Save time(s)* field specifies that new snapshots should be saved at time points 60n and 90n. So the following two snapshots are saved at timepoints 60n and 90n:

- `snap_60n`
- `snap_90n`

Note: If a snapshot with the same name existed before you restarted the simulation from a snapshot, a new snapshot named `snapshotName_runn` is created. For example, if a snapshot named `snap_60n` existed before you restarted the simulation from a snapshot, a new snapshot named, say, `snap_60n_run2` is created when you restart the simulation from a snapshot. This ensures that existing snapshots are not overwritten when you restart the simulation from a snapshot.

Simulating the Design Using the `irun` Command

The OSS-based netlister supports simulation using the `irun` command of the Virtuoso AMS Designer simulator.

The `irun` command uses file extensions of input files to determine which compiler to use. For example, files with the `.v` extension are compiled using the `ncvlog` Verilog compiler, files with the `.vhd` extension are compiled using the `ncvhdl` VHDL compiler, and files with the `.vams` extension are compiled using the `ncvlog` Verilog compiler with the `ncams` argument. After the input files are compiled, `irun` automatically starts `ncelab` to elaborate the design and then uses `ncsim` to simulate the design. For more information about the `irun` command, see the *irun User Guide*.

Note: You need to install the Cadence IUS 6.1 or a later release to use the `irun` command. If you are using an IUS release earlier than IUS 6.1, ADE uses the `ncverilog` command instead of the `irun` command. If the `ncverilog` command is used, the VHDL blocks in your design will not be netlisted.

You can specify the options for running `irun` using the [AMS Options](#) form.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

ams1: Netlist and Run Options

NETLIST AND RUN MODE

OSS-based netlister with irun

Cellview-based netlister with ncvlog, ncelab, ncsim

RUN OPTIONS

Compile incremental All

Elaborate incremental All

Simulate

Clean snapshot and pak files

Compile VerilogA as Verilog-AMS

SIMULATION MODE

Simulate Batch (normal) Interactive (debugger)

SAVE AND RESTART OPTIONS

Restart from:

Snapshot prefix

Save time(s)

Save incr Start time Stop time

OK Cancel Defaults Apply Help

Note: If you are using an IUS release earlier than IUS 6.1, ADE will use the `ncverilog` command instead of the `irun` command. You can specify the options for running `ncverilog` using the [AMS Options](#) form.

Netlist and Run Command


The *Netlist and Run* command generates the `irun` command using the options specified in the [AMS Options](#) form.

After running the *Netlist and Run* command, you can view the log file named `irun.log` using, *Simulation – Output Log – irun Log*. All the errors generated for the compiler, elaborator and AMS Designer simulator can be viewed in this log file.

Starting a Simulation

To start a simulation,


- ▶ Choose *Simulation – Netlist and Run*.

Alternatively click the  icon on the ADE simulation window, the netlist will reflect any design changes. Any edits done in the design or the properties form would reflect in the new netlist. You should use this option when you run the simulation for the first time and also when you have edited your design. Using this option ensures that your design, the ADE setup and the output netlist for simulation are synchronized.

To start a simulation using the existing netlist,

- ▶ Choose *Simulation – Run*.

Using this option, the design is not netlisted if a netlist is already available. This option is faster than the *Simulation – Netlist and Run* option, and is useful when no design changes have been made. The resulting simulation reflects simulation setup modifications such as analysis setup changes, design variable changes, and simulator option changes. It does not reflect design changes such as a change on the Edit Properties form and a change of the stop and switch view lists on the Environment Options form.

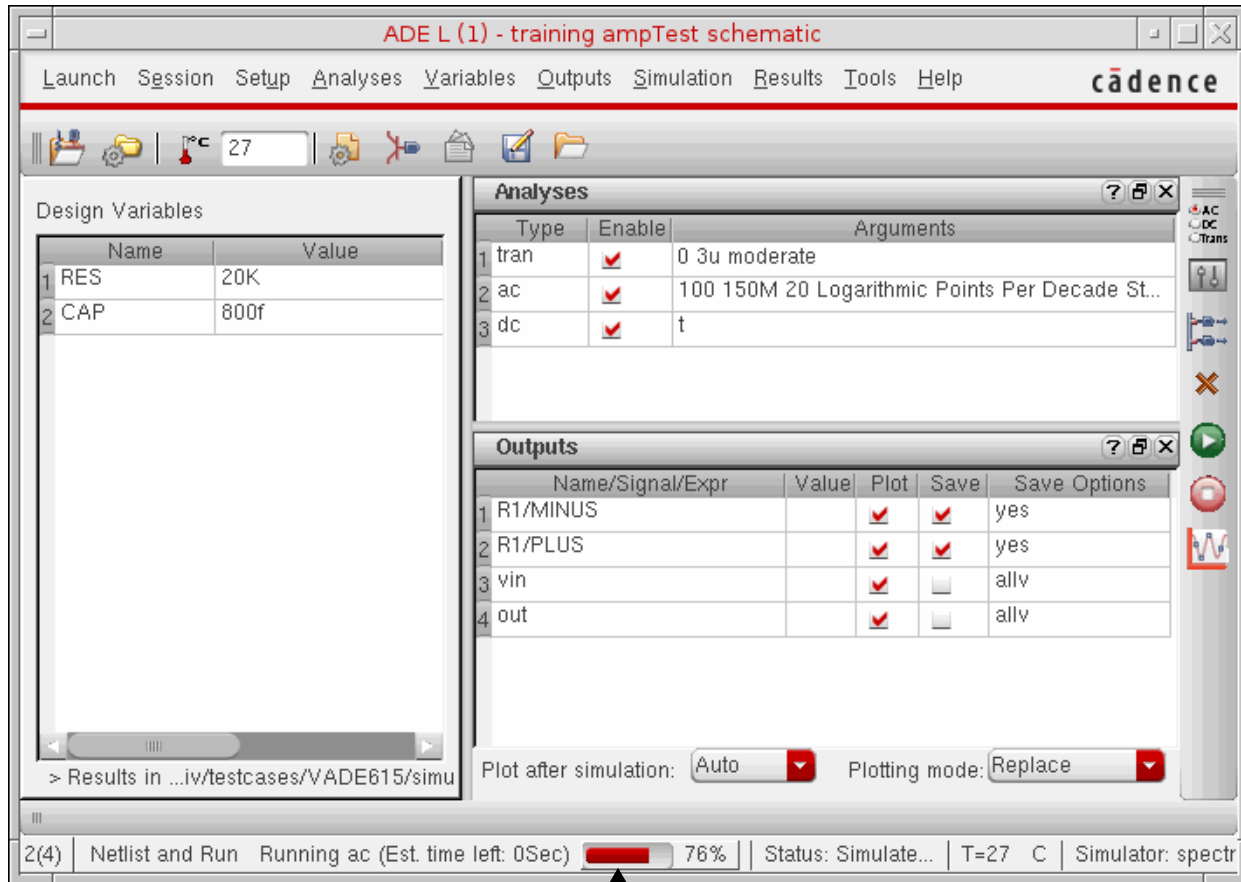
When you choose *Simulation – Netlist and Run* or click the  icon on the simulation window, a progress bar appears on the bottom right of the simulation window displaying the status of netlisting and simulation runs:

- For netlisting, the progress bar displays the percentage completion status.
- For simulation runs, the progress bar displays the analysis name, estimated time to complete, and the percentage completion status.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The following figure displays the progress bar displaying the status of analysis run:



Note: If data is purged for the current session, you can exit dfl via *File – Exit* or can continue to work in the session. To do this, just reset the purged session and invoke a new session. Also, error messages are generated in the CIW if you select *Simulation – Run* (or *Netlist – Create/Recreate*) in a purged dfl session. The messages will be displayed in the CDS.log file for both virtuoso and OCEAN (icxx), as follows:

❑ virtuoso

WARNING You do not have the required cellViews or properties open for this session.

WARNING You may have purged the data from virtual memory or the schematic data has been closed.

WARNING Reset the ADE session (via Session->Reset) or quit and re-invoke ADE and other application(s) you are using.

❑ **OCEAN (CIW or icxx)**

You do not have the required cellViews or properties open for this session. You may have purged the data from virtual memory or the schematic data has been closed. You can type:
`simulator('simulatorName)` to reset the session or quit the application that you are using.

Interrupting or Stopping a Simulation

To stop a simulation that is running,

- Choose *Simulation – Stop*.

The system saves any simulation results that were calculated.

The stopped simulation cannot be continued.



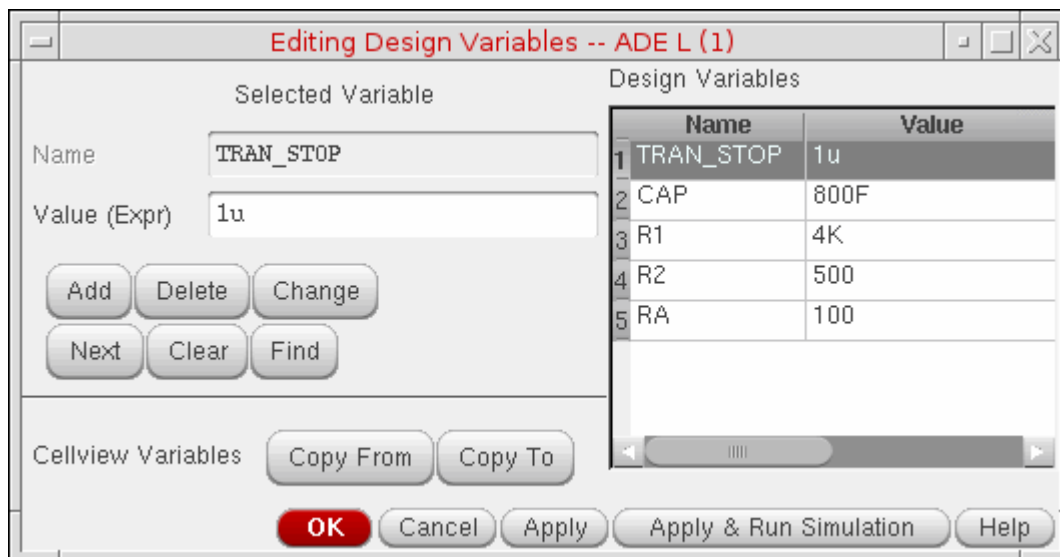
The *Simulation – Stop* option is not only used to stop a running simulation, it is also used to release the *Spectre* license.

Updating Variables and Resimulating

To change design variable values and run another simulation,

1. In the Simulation window, double-click the variable you want to change, or in the Schematic window, choose *Setup – Variables*.

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



2. Change the *Value (Expr)* field.
3. Click *Apply & Run Simulation*.

Saving Simulator Option Settings

You can save the current simulator option settings and later restore these settings.

To save the simulator options,

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

The Saving State form appears.

2. Type a name for the saved simulation state.
3. Check that the *Simulation Options* box is on and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Restoring Saved Settings

To restore saved simulator options,

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

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

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

The display shows the saved states for the cell and simulator combination.

3. Click a state name.
4. Check that *Simulation Options* is selected and click *OK*.

Note: Saved states are simulator dependent if you save the analyses. Otherwise, you can restore states from a different simulator.

Viewing the Simulation Output

To read the log file (*.out) generated by simulators use either of the following options:

- Choose *Simulation – Output Log*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The simulation output log file appears in a text window.

```
/servers/scratch02/krajiv/testcases/VADE615/simulation/ampTest/spectre/schematic/psf/spectre.out
File Edit View Help
Cadence (R) Virtuoso (R) Spectre (R) Circuit Simulator
Version 13.1.1.078.isr7 32bit -- 25 May 2014
Copyright (C) 1989-2014 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, Virtuoso and Spectre are reg
Includes RSA BSAFE(R) Cryptographic or Security Protocol Software from RSA Security, Inc.
User: krajiv Host: rlno-krajiv HostID: 17Ac7728 PID: 23698
Memory available: 28.4344 MB physical: 4.0140 GB
CPU Type: Intel Xeon E312xx (Sandy Bridge)
Processor PhysicalID CoreID Frequency Load
0 - - 2700.0 4.5
Simulating `input.scs' on rlno-krajiv at 11:51:44 AM, Thur Sep 4, 2014 (process id: 23698).
Current working directory: /servers/scratch02/krajiv/testcases/VADE615/simulation/ampTest/spectre/schematic/netlist
Command line:
\
 /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/bin/spectre \
 input.scs +escchars +log ../psf/spectre.out +inter=mpsc \
 +mpsession=spectre0 16373 1 -format psf -raw ../psf \
 -I/servers/scratch02/krajiv/testcases/VADE615/ +ltimeout 900 \
 -maxw 5 -maxn 5
spectre pid = 23698
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libinfineon_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libphilips_o_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libphilips_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libparam_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libstmodels_sh.so ...
Reading file: /servers/scratch02/krajiv/testcases/VADE615/simulation/ampTest/spectre/schematic/netlist/input.scs
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/configs/spectre.cfg
Reading file: /servers/scratch02/krajiv/testcases/VADE615/Models/myModels.scs
Time for NDB Parsing: CPU = 144.978 ms, elapsed = 388.706 ms.
Time accumulated: CPU = 724.889 ms, elapsed = 388.721 ms.
Peak resident memory used = 26.6 Mbytes.
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/spectre/etc/ahdl/discipline.h
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/ahdl/disciplines.vams
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/spectre/etc/ahdl/constants.h
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/ahdl/constants.vams
Time for Elaboration: CPU = 71.988 ms, elapsed = 1.82931 s.
Time accumulated: CPU = 796.877 ms, elapsed = 2.21873 s.
Peak resident memory used = 29.4 Mbytes.
```

This file contains information about the simulation environment, the command lines sent to the simulator, and the simulation error or warning messages.

The simulation output log file also contains hyperlinks to the various components for better interaction with the design. You can highlight an instance, open the netlist file, and open the

Virtuoso Analog Design Environment L User Guide

Running a Simulation

analyses form by clicking the hyperlinks in this file. The following figures show the available hyperlinks and their functions:

The screenshot shows a terminal window with the following content and annotations:

- Annotation 1:** A red arrow points to the `input.scs` file name in the command line, with the text "Opens the design netlist in plain text format."
- Annotation 2:** A red arrow points to the current working directory path, with the text "Opens the terminal window in the current working directory."
- Annotation 3:** A red arrow points to the `input.scs` file name in the command line, with the text "Opens the simulation log file in plain text format."
- Annotation 4:** A red arrow points to the error message `ERROR (SFE-1997): "input.scs" 44: IO.CO: parameter 'c': The parameter 'CAP' is used but not set.`, with the text "Highlights the instance in the design window."

```
File Edit View Help cadence
Includes RSA BSAFE (R) Cryptographic or Security Protocol Software from RSA Security, Inc.
User: krajiv Host: rlno-krajiv HostID: 17AC7728 PID: 32635
Memory available: 76.4682 MB physical: 4.0140 GB
CPU Type: Intel Xeon E312xx (Sandy Bridge)
Processor PhysicalID CoreID Frequency Load
0 - - 2700.0 5.2

Simulating `input.scs' on rlno-krajiv at 2:28:37 PM, Wed Sep 24, 2014 (process id: 32635).
Current working directory: /servers/scratch02/krajiv/testcases/VADE615/simulation/ampTest/spectre/schematic/netlist
Command line:
  /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/bin/spectre \
  input.scs +escchars +log ../psf/spectre_out +inter=mpsc \
  +mpsession=spectre3_6230_10 -format psfsl -raw ../psf \
  -I/servers/scratch02/krajiv/testcases/VADE615/ +ltimeout 900 \
  -maxw 5 -maxn 5
spectre pid = 32635

Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libinfineon_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libphilips_o_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libphilips_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libsparam_sh.so ...
Loading /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/cmi/lib/5.0/libstmodels_sh.so ...
Reading file: /servers/scratch02/krajiv/testcases/VADE615/simulation/ampTest/spectre/schematic/netlist/input.scs
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/configs/spectre.cfg
Reading file: /servers/scratch02/krajiv/testcases/VADE615/Models/myModels.scs
Time for NDB Parsing: CPU = 94.986 ms, elapsed = 1.05667 s.
Time accumulated: CPU = 533.918 ms, elapsed = 1.05668 s.
Peak resident memory used = 26.6 Mbytes.

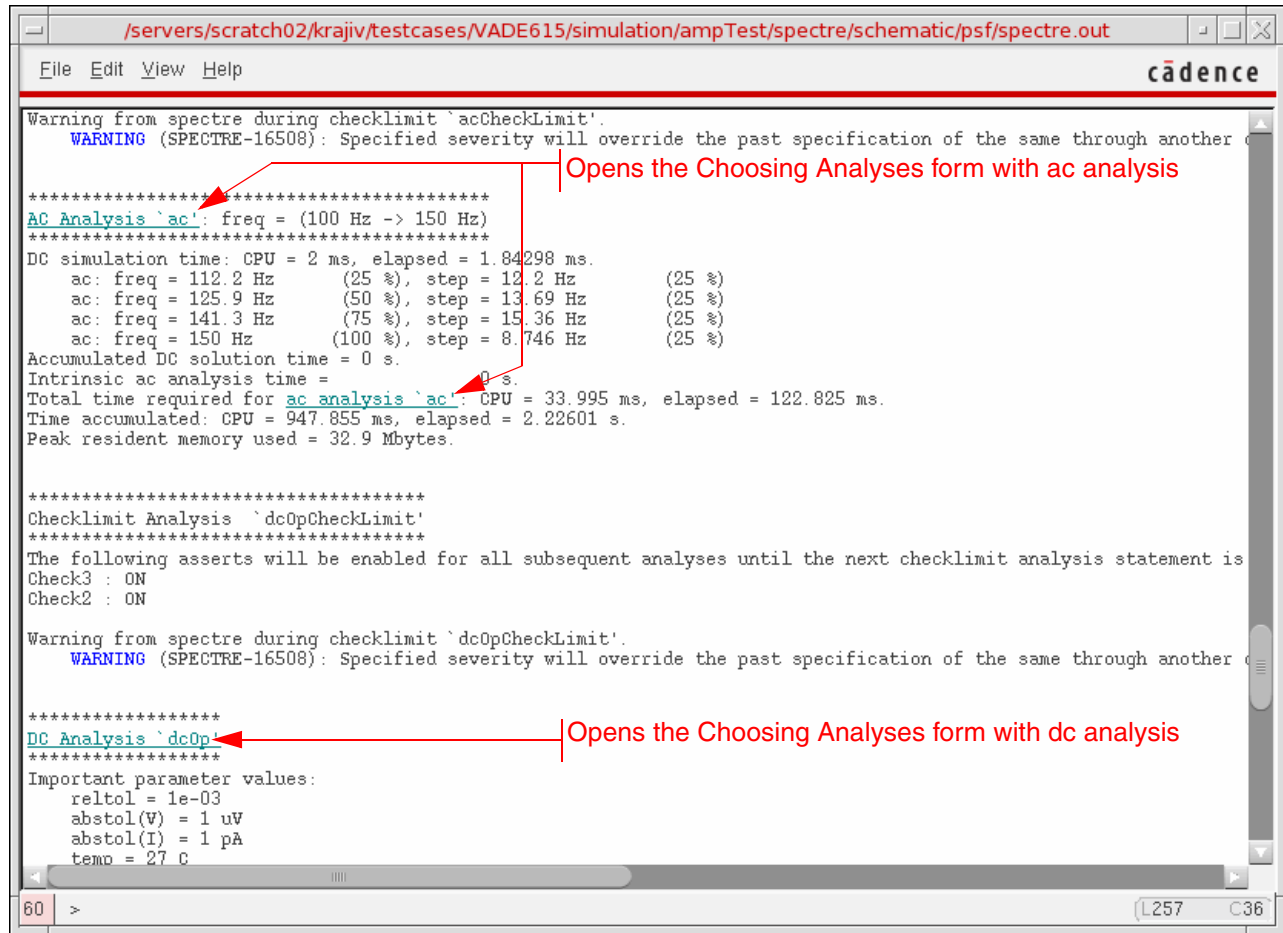
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/spectre/etc/ahdl/discipline.h
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/ahdl/disciplines.vams
Reading link: /grid/cic/install_support/MMSIM/MMSIM131/latest/lrx86/tools.lrx86/spectre/etc/ahdl/constants.h
Reading file: /grid/cic/install_support/MMSIM/MMSIM131/13.11.078/lrx86/tools.lrx86/spectre/etc/ahdl/constants.vams

Error found by spectre in `amplifier': `IO', during hierarchy flattening.
ERROR (SFE-1997): "input.scs" 44: IO.CO: parameter `c': The parameter `CAP' is used but not set.

Time for Elaboration: CPU = 40.993 ms, elapsed = 137.722 ms.
Time accumulated: CPU = 575.911 ms, elapsed = 1.21472 s.
Peak resident memory used = 29.3 Mbytes.
```

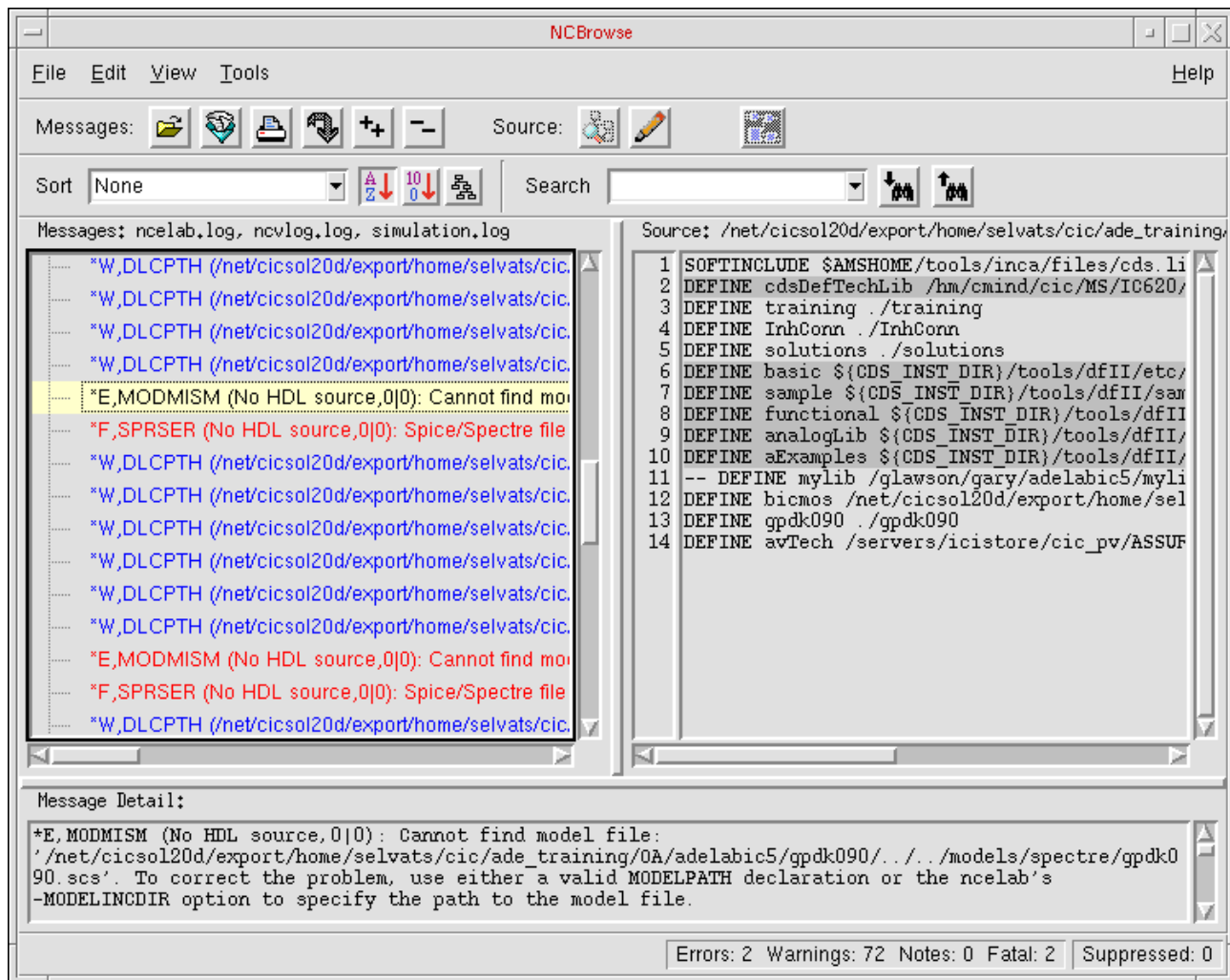

Virtuoso Analog Design Environment L User Guide

Running a Simulation



Viewing the Output Log for AMS

You can view the output logs (Netlister, Compiler, Elaborator, Simulator) from the *Simulation – Output Log* submenu. Choose *Output Log – LogFile utility* to launch the NCBrowse message browser.

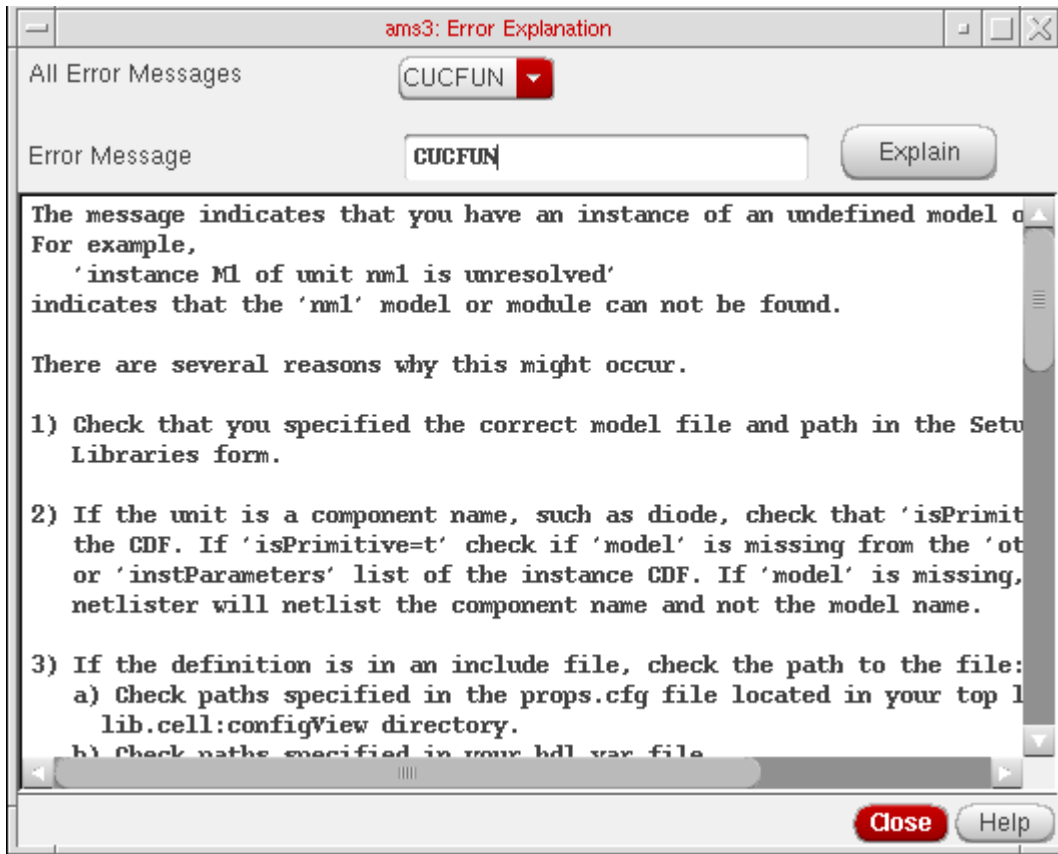


You can analyze log files with the NCBrowse utility. The NCBrowse utility lets you select a log file message and view the source code that caused the message. It includes sorting and filtering tools that let you view only those messages that are important to you. You can also use the message browser to print formatted output reports. To know more about the NC Browse utility, refer to the *NC Browse Message Browser User Guide*.

Similarly, choose the *Netlister Log*, *Compiler Log*, *Elaborator Log*, *Simulator Log*, *3-step Log* or *irun Log* options to view the respective logs in the *Results Browser*.

Viewing the Error Explanation for AMS

You can view detailed explanation of the error for AMS in the Error Explanation form. Choose *Simulation – Output Log – Error Explanation...* to launch the Error Explanation form.



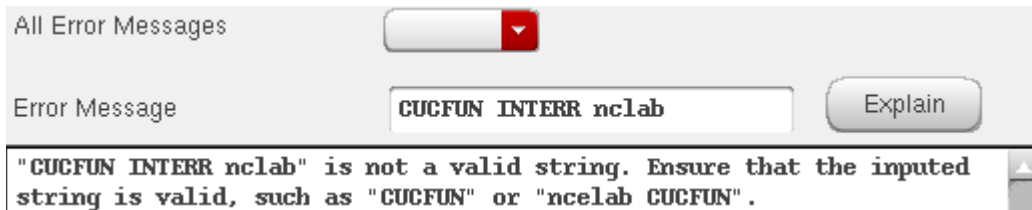
All the error messages that are present in the log files that are created in the psf directory while a session is being run are listed in the All Error Messages field. To view details about an error message, you can select the Error from All Error Messages field and then click the Explain button. The description for the error appears as shown in the figure above.

You can also enter the error in the Error Message field. If you enter a string, it will be treated as an error string. If you enter two strings, the first string is treated as the tool name and the second string is treated as an error message. If you enter more than two strings, you will get a warning. For example, if you enter "CUCFUN INTERR ncelab" as the error string, you will get a warning

Virtuoso Analog Design Environment L User Guide

Running a Simulation

nchelp: *W,NOTERR: error CUCFUN,INTERR,nclab is not an error name for nceverilog.



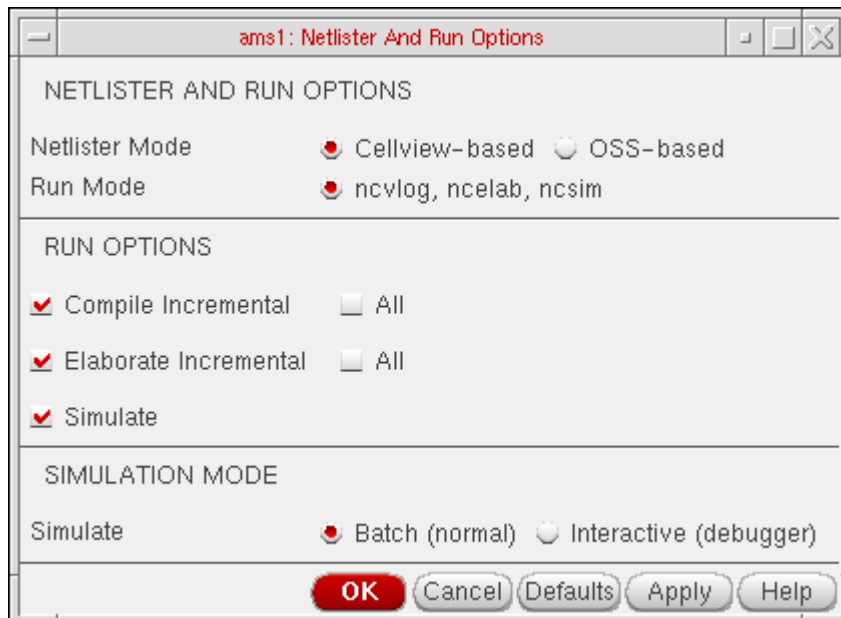
The error message description consists of the following:

- For a growing number of messages, special messages are available to describe the problem and if known, a solution for the same. The number of messages for which there will be additional help is expected to grow with each release.

For all error names, the nchelp for that error name is printed.

Using the SimVision Debugger

You can choose AMS run options and simulation run options by choosing *Simulation – Run Options*. This brings up the *Netlister And Run Options* form.



You can choose to only compile, elaborate or simulate the design, although by default all three are selected as shown in the screenshot. For example, if you select only *Elaborate*, the

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Simulation – *Netlist* command only elaborates the design. Compilation and elaboration can be incremental or for the whole design together. A simulation would fail if you choose both *Compile* and *Simulate*. You would need to either deselect *Simulate* or select *Elaborate* as well.

The *Batch* mode is the default mode. In this mode, all signal plotting occurs in the default analog waveform plotting tool– Virtuoso Visualization and Analysis.

The *Interactive* mode launches the complete SimVision debug environment on top of current design. SimVision is a graphical user interface for Cadence simulators and related debugging tools. For details, refer to the *SimVision User Guide* and the *Virtuoso AMS Designer Simulator User Guide*.

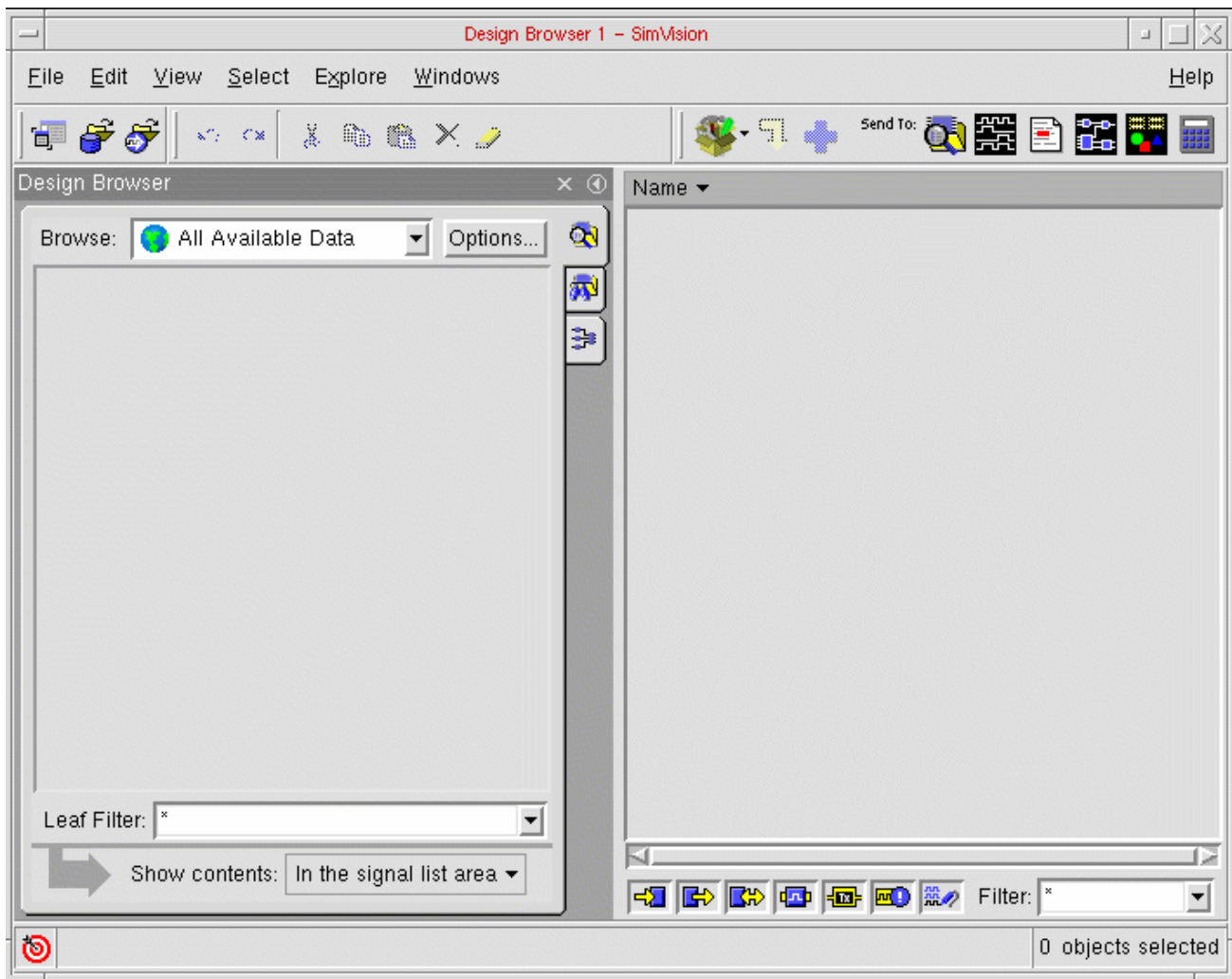
In the interactive mode,

- The *Parametric Analysis* command in the *Tools* menu cannot be used. If you attempt to use it, a prompt appears saying that the option is not enabled for the *interactive* mode and suggests that you switch to the *batch* mode to be able to use it. It also warns you that if you attempt to rerun the simulation without exiting SimVision, the simulation results may get corrupted.
- The saved currents and voltages can be plotted in SimVision Waves through the ADE selection mechanism. The ADE plot outputs signals go to SimVision Waves if it is open. If it is not open, the output goes to the selected waveform tool – Virtuoso Visualization and Analysis.
- If you select new nets for the plot list, they take effect in the next SimVision run.
- If you select within SimVision's browser, the selected information may be placed in the ADE Outputs pane.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Direct Plot and plotting from the calculator or browser go to the selected waveform tool, regardless of the run mode or whether or not SimVision is up.

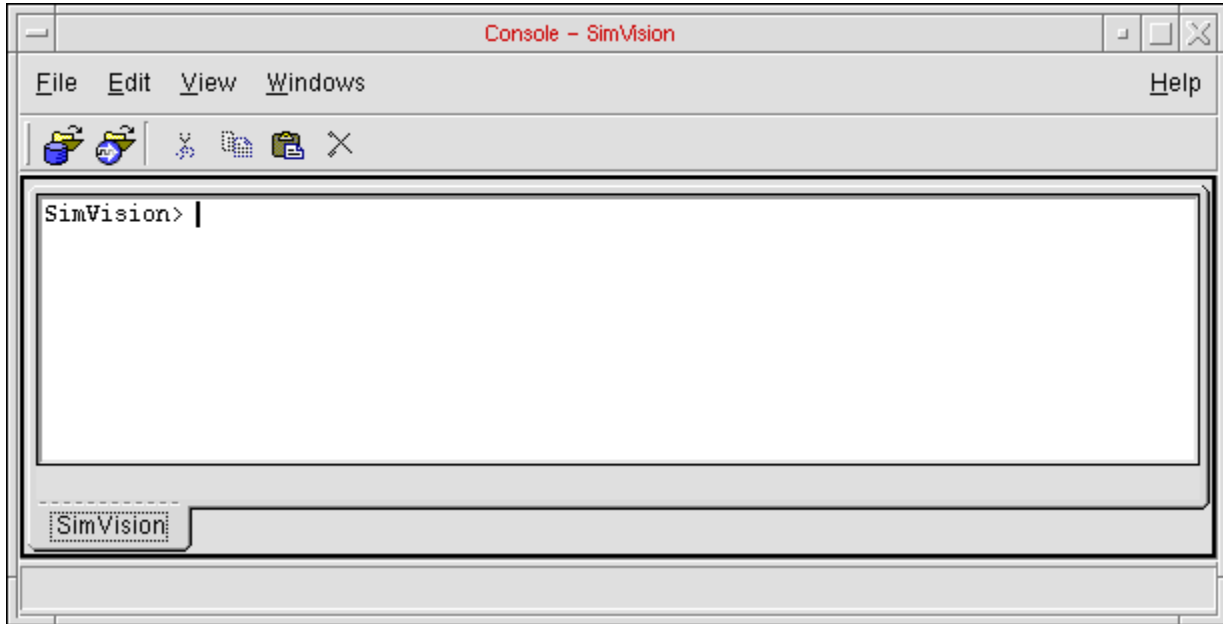


ADE sends Tcl commands to SimVision for the simulation run and to plot the signals listed in the ADE outputs pane. This mode lets you control ncsim and simvision from the console window.

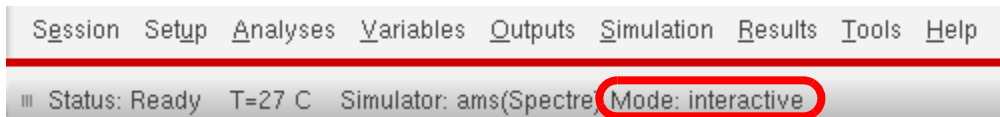
Virtuoso Analog Design Environment L User Guide

Running a Simulation

The status of SimVision plotting commands can be seen at the SimVision prompt in the console window.



The ADE window displays the selected mode on the status bar as highlighted in this snapshot.

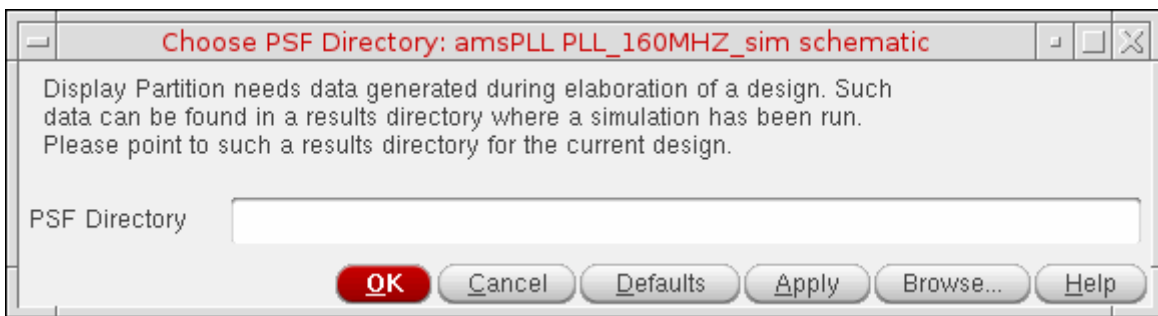


Display Partition

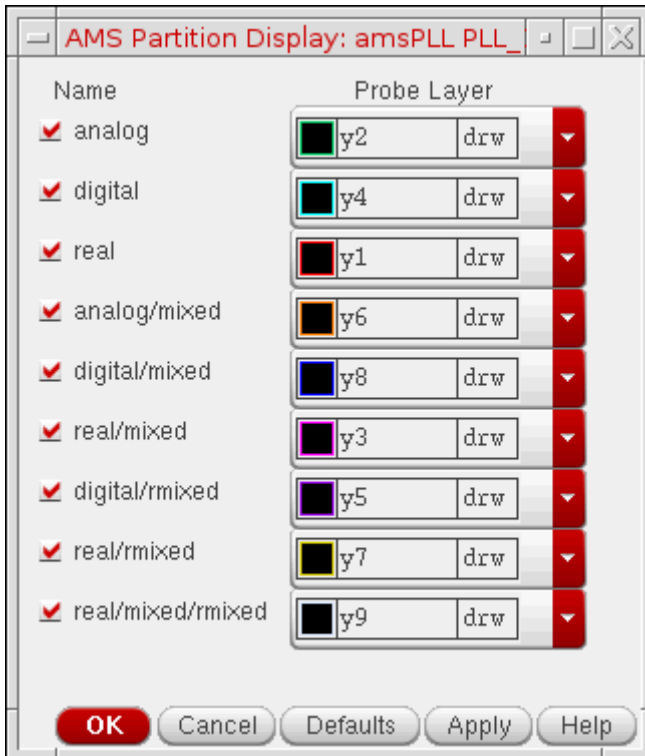
Once you run a simulation, you can view your data and distinguish between analog instances or nets, digital instances or nets, and mixed instances or nets by the color associated with each partition.

1. To access this feature, choose *Launch – Plugins – Mixed Signal Options – AMS* in your schematic window. The *AMS* option appears on the menu bar.
2. Choose *AMS – Display Partition – Initialize*. If you choose the Initialize menu item after running a simulation in ADE, the *Display Partition* menu items are enabled.

However, if you do not run the simulation, the *Choose PSF Directory* form is displayed, as shown below. On this form, type the path to the simulation data directory in the *PSF Directory* field, or select the results using the *Browse* button.



3. Choose *AMS – Display Partition – Interactive* to view the *Partition Display* window.



This feature highlights the analog, digital, and real parts of a schematic in different colors.

analog indicates that the net or instance is analog in nature. A net or instance is analog throughout a hierarchy if everything under that instance is analog.

digital indicates that the net or instance is digital in nature. A net or instance is digital throughout a hierarchy if everything under that instance is digital.

real indicates that the net or instance is wreal in nature. A net or instance is real throughout a hierarchy if everything under that instance is real.

analog/mixed indicates that the net is mixed in nature. It means that some segments of the net or instance are analog and some are digital in the hierarchy.

digital/mixed indicates the same as analog/mixed, the difference being that at the current level, the net is digital.

real/mixed indicates the net is wreal and connecting to R2E, or the instance has both wreal and electrical inside.

digital/rmixed indicates the net is logic net connecting to R2L, and R2L is connected to wreal net.

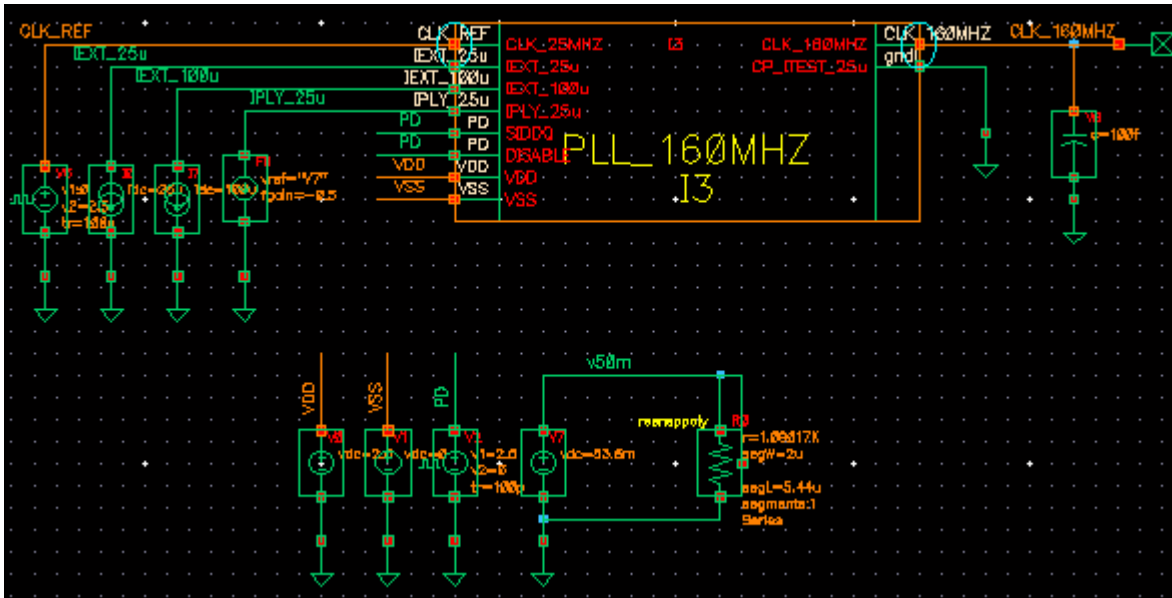
Virtuoso Analog Design Environment L User Guide

Running a Simulation

real/rmixed indicates the net is wreal net connecting to R2L, or the instance has both wreal and logic inside.

real/mixed/rmixed indicates the net is wreal net connecting to both R2L and R2E, or the instance has wreal,electrical and logic nets inside.

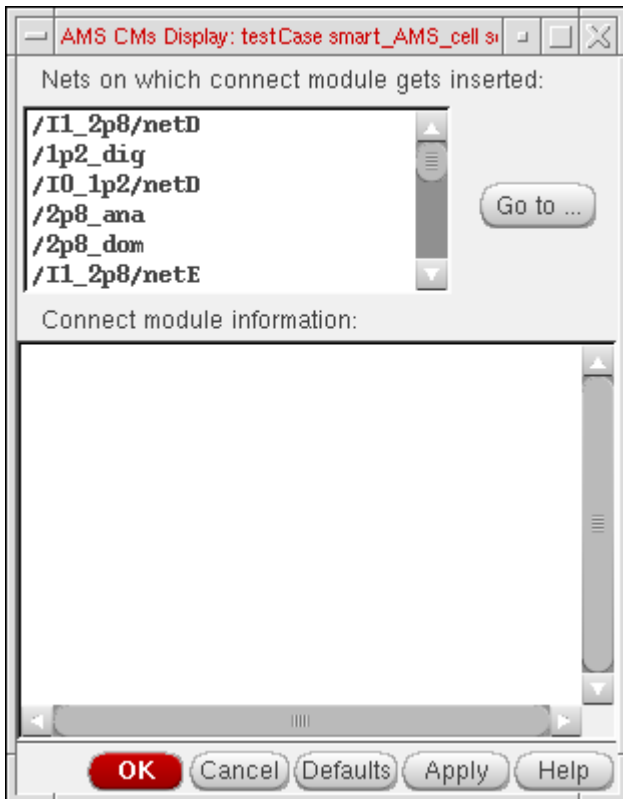
The schematic reflects these color preferences.



Virtuoso Analog Design Environment L User Guide

Running a Simulation

4. You can see all the IEs in the configuration by choosing *AMS – Display Partition – IE Information*. This brings up the *AMS CMs Display* dialog box displaying all IEs.



You can select an IE and click *Go to* to see it zoomed-in on the schematic. When you select an IE, information pertaining to it and related connect module information appears in the box below.

Default Digital Discipline Selection

Disciplines denote an object as analog (with an electrical discipline, for example) or digital (with a logic discipline, for example). You can define the default disciplines for design objects by using the *AMS – Default Digital Discipline Selection* command in the Composer window.

Default digital discipline selection indirectly controls the selection of connect module (IE) on mixed nets. A discipline denotes an object as analog or digital based on whether it is electrical or logic, respectively. When objects of different disciplines are connected, connect rules determine which connect modules are inserted on mixed nets.

The inserted connect modules then convert signals to values that are appropriate for each discipline. To customize the conversions for your design, you can use the connect rules to override parameters, such as supply voltage or rise time, that are used in the connect modules.

For more information, see the *Mixed-Signal Aspects of Verilog-AMS* chapter of the *Cadence Verilog-AMS Language Reference*.

To specify a default digital discipline for design objects,

1. Choose *Launch – Mixed Signal Options – AMS* in your schematic window.

The *AMS* menu appears on the menu bar.

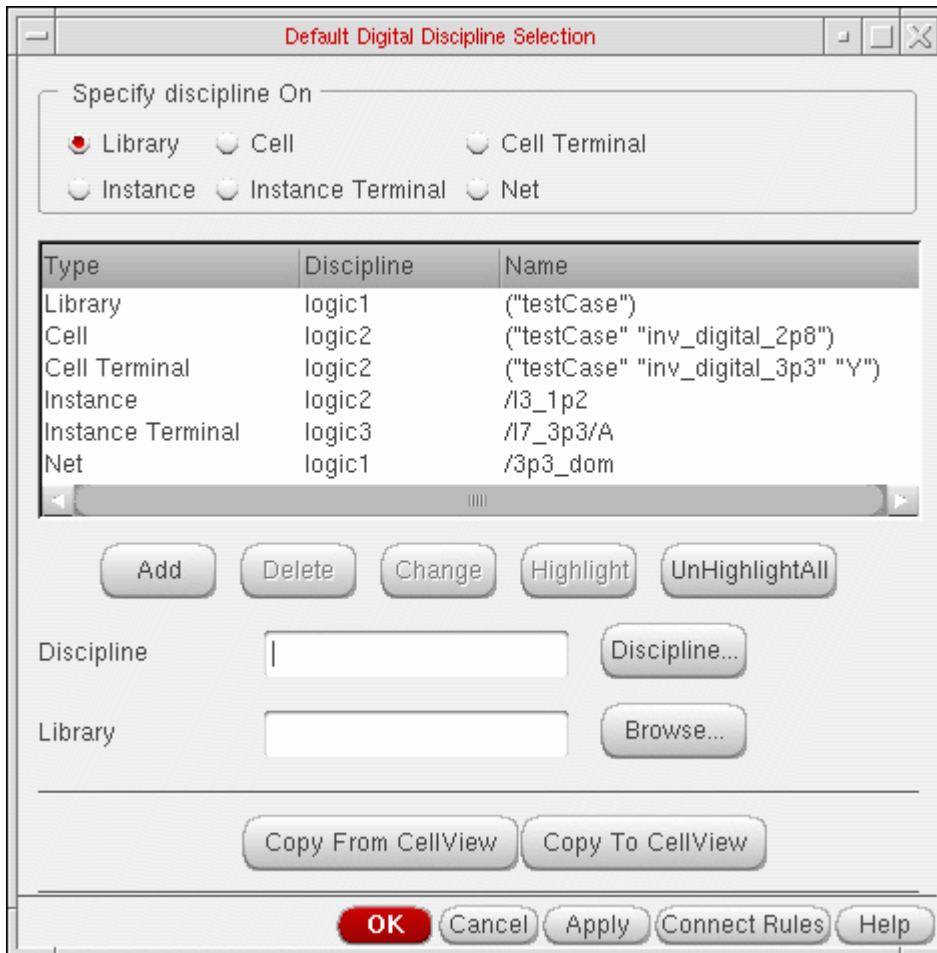
2. Choose *AMS – Default Digital Discipline Selection*.

A submenu appears showing six options: *Library, Cell, Cell Terminal, Net, Instance, Instance Terminal*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

3. Select any of the submenu options to bring up the Default Digital Discipline Selection form. For example, if you select *Library*, the form comes up as shown here. You can specify disciplines on libraries in this form.



Alternatively, you can select any of the other options from the *Specify discipline on* group box. The fields below the *Discipline* field change as follows depending on the design object selected.

Design Object	Related fields	How to specify
<i>Library</i>	<i>Library</i>	Type in a valid value or use the <i>Browse</i> button to specify a library from the Library Browser.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Design Object	Related fields	How to specify
<i>Cell</i>	<i>Library Cell</i>	Type in valid values or use the <i>Browse</i> button to specify a library and cell from the Library Browser.
<i>Cell Terminal</i>	<i>Library Cell Terminal</i>	Type in valid values or use the <i>Select</i> button to select a cell terminal from the schematic.
<i>Net</i>	<i>Net</i>	Type in a valid value or use the <i>Select</i> button to select a net from the schematic.
<i>Instance</i>	<i>Instance</i>	Type in a valid value or use the <i>Select</i> button to select an instance from the schematic.
<i>Instance Terminal</i>	<i>Instance Terminal</i>	Type in a valid value or use the <i>Select</i> button to select an instance terminal from the schematic.

4. The table lists the types of design objects, their default disciplines and their names. You can sort this table on *Type* or *Discipline*.
5. To create a new discipline,
 - a. Click the *Discipline* button.

The Create Discrete Disciplines form appears in which you can create a discipline. The new discipline appears in the *Discipline* field.
 - b. Click the *Add* button.
6. To delete one or more disciplines specified on an object,
 - a. Select one or more rows in the table.
 - b. Click the *Delete* button.
7. To change a discipline specified on an object,
 - a. Select a row in the table.

The form refreshes to show the fields pertaining to the selected kind of object.
 - b. Change the values as required and click the *Change* button.

8. To view a discipline on the schematic,

a. Select a row from the table.

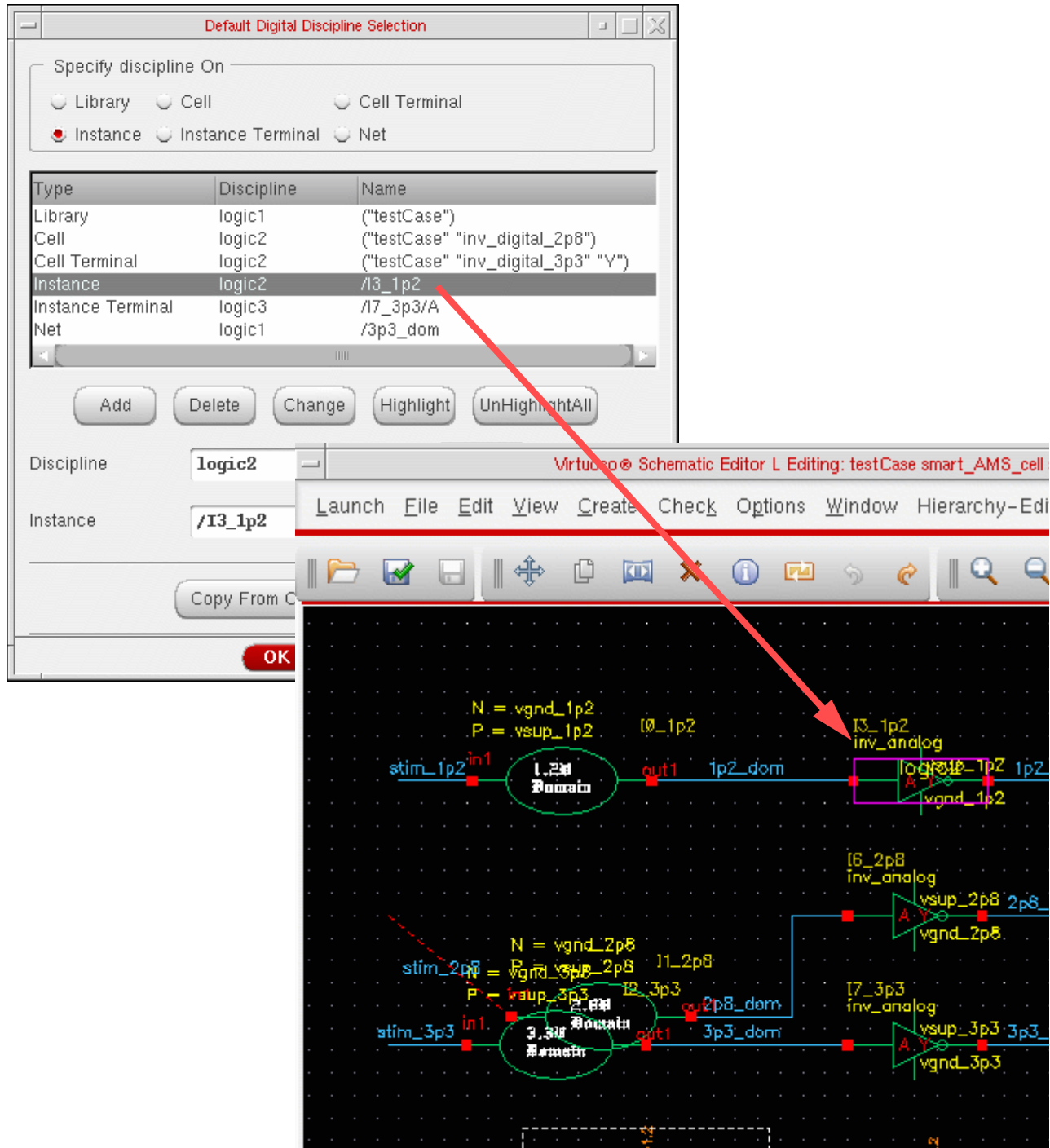
Note that only nets, instances and instance terminals can be highlighted.

b. Click the *Highlight* button.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

As shown in the illustration below, the selected discipline, in this case `logic_1p2`, appears highlighted in the schematic. You can select and highlight multiple objects this way. You can click the *Unhighlight All* button to remove the highlights.



9. To copy disciplines from the cellview, click the *Copy from Cellview* button. Conversely, to copy disciplines from the form to the cellview, click the *Copy to Cellview* button.

After copying over disciplines from a cellview, when you click the *Discipline* button, a form may pop up listing disciplines that have not been defined and asking if you would like to define them. If you select *Yes*, the Create Discrete Disciplines form appears showing a list of the undefined disciplines. If you select *No*, it appears blank.

10. You can specify connect rules for a selected discipline by using the *Connect Rules* button to bring up the Select Connect Rules form. You can use this form to create, modify or delete connect rules. This form does appear only if ADE is up and the simulator set to ams. Otherwise, an error message appears.

11. Click *OK*.

The disciplines created are automatically compiled.

The settings you made are saved for the current session. You can save these settings for future sessions if you select the *Discipline Selection* option in the Saving State form and save the current ADE state. You can later load the state using the Loading State form with the *Discipline Selection* option selected. For more information, see Saving and Restoring the Simulation Setup on page 97.

Running a Parametric Analysis

For information on how to select range specifications, start, interrupt, re-start and close a parametric analysis run, see Running the Parametric Analysis.

Device Checking

You can use Spectre's device checking feature to determine whether elements in your circuit are violating predefined safe operating areas. You can specify rules that check operating point parameters, as well as expressions that combine these parameters. These checks can be part of your model card, netlist, or any other file that is included for simulation as a part of the design. If your circuit has any violations, you can highlight violating devices on the schematic, view the details of violations for a device check, and print a summary of all the violations.

For more information, see the following sections:

- [Enabling and Disabling Device Checking](#) on page 403
- [Setting Up Device Checks](#) on page 404
- [Specifying Global Device Check Options](#) on page 417
- [Specifying Options for Writing Violations Information](#) on page 425
- [Viewing, Printing, and Saving Device Check Violations](#) on page 426

Note: For information about setting up device checking in Virtuoso Analog Design Environment XL and GXL, see the [Virtuoso Analog Design Environment XL User Guide](#).

Enabling and Disabling Device Checking

You can specify whether the Spectre simulator must perform device checking when you run a simulation.

To enable the Spectre simulator to perform device checking,

1. From the Simulation window, choose *Simulation – Options – Analog*.
The Simulator Options form appears.
2. Click the *Check* tab page.
3. In the *DEVICE CHECKING OPTIONS* section, select the *yes* check box next to *dochecklimit*.
4. Click *OK*.

To stop the Spectre simulator from performing device checking,

1. From the Simulation window, choose *Simulation – Options – Analog*.
The Simulator Options form appears.

2. Click the *Check* tab page.
3. In the *DEVICE CHECKING OPTIONS* section, select the *no* check box next to *dochecklimit*.
4. Click *OK*.

Setting Up Device Checks

To set up device checks,

- From the Simulation window, choose *Simulation – Device Checking*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The Device Check Specification form appears.

Device Check Specification

Check Setup

Name Analyses tran dcOp ac dc sweep sp noise pa

Subcircuit Type

Selection Choices

Param Type Param List

Expression Min Max

Check Windows

Device Check Summary

ab	Name	Subcircuit	Dev/Mod/Pri	Expr/Param	Min	Max	gid	Duration	Severity	Color	Info	CheckWindow
----	------	------------	-------------	------------	-----	-----	-----	----------	----------	-------	------	-------------

For more information about using this form, see the following topics:

- [Creating Device Checks](#) on page 406
- [Loading Predefined Device Checks from a File](#) on page 413
- [Disabling and Enabling Individual Device Checks](#) on page 416
- [Modifying Device Checks](#) on page 416
- [Saving Device Checks to a File](#) on page 416

- [Deleting Device Checks](#) on page 417
- [Hiding and Showing Columns](#) on page 430

Creating Device Checks

To create a device check,

1. (optional) In the *Name* field, enter a name for the device check.

If you do not specify a name, the name `checkn`, where *n* is an incremental number, is assigned to the device check. For example, if you do not specify a name, the name `check1` is assigned to the device check.

2. In the *Type* cyclic field, select the type of device check you want to add.
3. Specify the options for the type of device check you are adding. For more information, see the following topics:
 - [Specifying Options for Device Checks of Type Device](#) on page 409
 - [Specifying Options for Device Checks of Type Model](#) on page 410
 - [Specifying Options for Device Checks of Type Primitive](#) on page 411
 - [Specifying Options for Device Checks of Type Parameter](#) on page 411
 - [Specifying Options for Device Checks of Type Expression](#) on page 412
4. Select the check box next to the analyses for which you want to enable the device check.

Select	To
<i>tran</i>	Enable the device check for transient analyses.
<i>dcOp</i>	Enable the device check for DC analyses.
<i>ac</i>	Enable the device check for AC analyses.
<i>dc sweep</i>	Enable the device check for DC sweep analyses.

Important

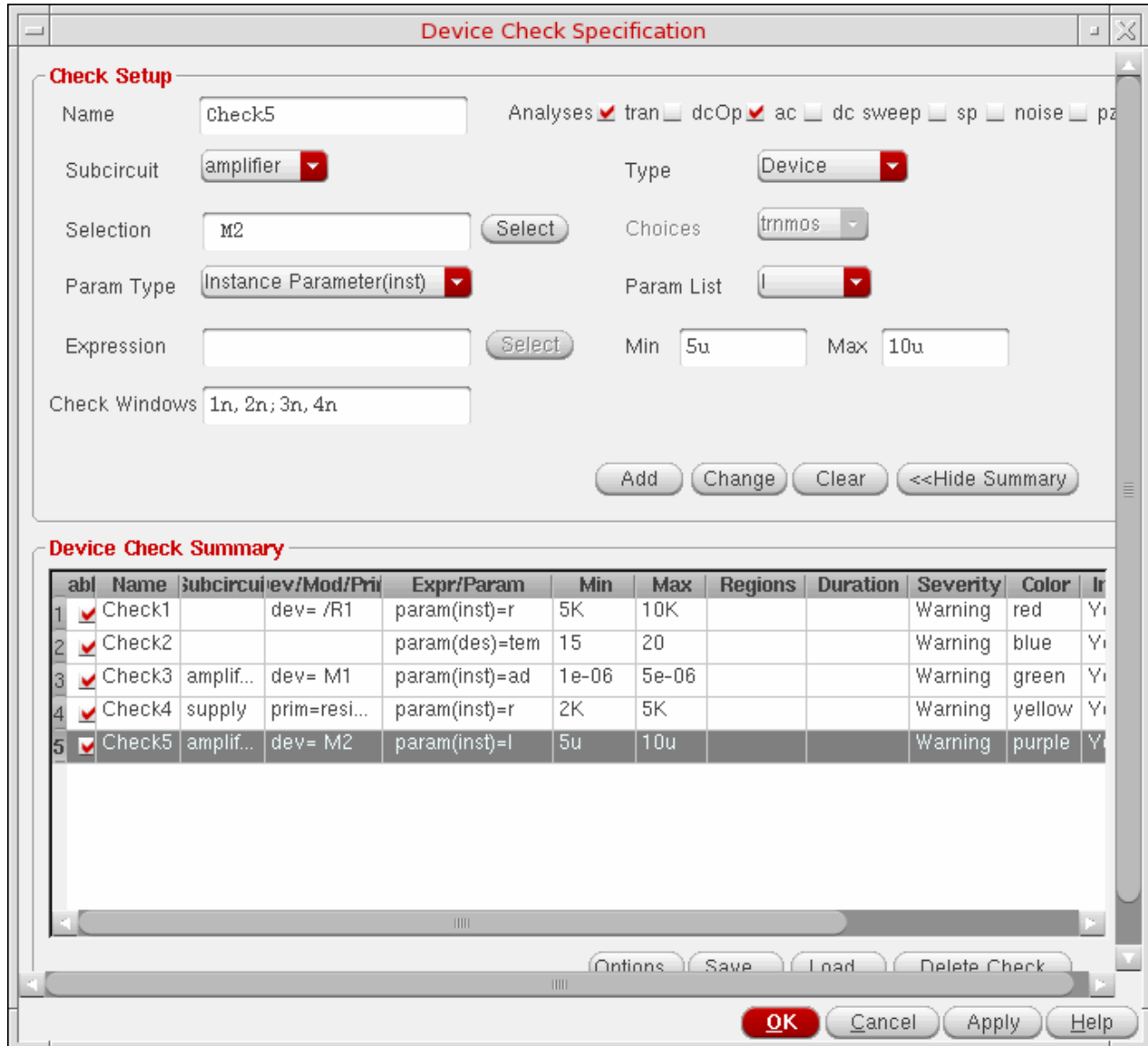
Ensure that the analyses for which device checks are enabled are setup for your design.

5. Click *Add*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The device check is added in the Device Check Summary table.



Note the following:

- ❑ By default, the device check is enabled. For more information, see [Disabling and Enabling Individual Device Checks](#) on page 416.
- ❑ If multiple devices are defined in a device check of type device, a single check is displayed in the Device Check Summary table. However, during netlisting, separate device checks are created for each device. For example, if the devices D1 and D2 are defined in a device check named myCheck, the Device Check Summary table

Virtuoso Analog Design Environment L User Guide

Running a Simulation

displays the `myCheck` check. When the design is netlisted, the checks `myCheck__D1` and `myCheck__D2` are created.

6. (optional) Specify additional options for the device check in the following columns in the Device Check Summary table:

Duration	Double-click on the <i>Duration</i> column and specify the time period in seconds over which the check has to be violated before a warning is displayed. Applicable to transient analyses only.
Severity	<p>Click on the <i>Severity</i> column and select the severity level— <i>Notice</i>, <i>Warning</i>, <i>Error</i>, <i>Fatal</i> or <i>None</i>—of the message that will be displayed when Spectre reports violations for the check in the simulation log file and the Violations Display form.</p> <ul style="list-style-type: none">■ If the severity level is <i>Notice</i> or <i>Warning</i>, simulation continues after the message is displayed.■ If the severity level is <i>Error</i>, the Spectre circuit simulator aborts the analysis when the first error-level violation occurs.■ If the severity level is <i>Fatal</i>, the Spectre circuit simulator aborts the simulation when the first fatal-level violation occurs. <p>The default setting is <i>Warning</i>.</p>
Color	<p>Click on the <i>Color</i> column and select the color to be used when highlight the devices on the schematic that failed the device check.</p> <p>If you do not specify the color, the following default colors are used, depending on the severity level specified for the device check:</p> <ul style="list-style-type: none">■ Yellow color for Warning severity level.■ Red color for Error severity level.■ Blue color for Fatal severity level.■ Orange color for Notice severity level.■ Blue color for None severity level.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Info	<p>Click on the <i>Info</i> column and do one of the following:</p> <ul style="list-style-type: none">■ Select <i>Yes</i> to ignore the <i>Min</i>, <i>Max</i>, and <i>Duration</i> options specified for the device check. Only the parameter value specified for the device check will be displayed in the simulation log file.■ Select <i>No</i> if you do not want to ignore the <i>Min</i>, <i>Max</i>, and <i>Duration</i> options specified for the device check. <p>The default setting is <i>no</i>.</p>
Message	<p>Double-click on the <i>Message</i> column and enter the message to be displayed in the <u>Violations Display</u> form and in the simulation log file if the device check fails.</p>

7. Click *OK* or *Apply* to save the changes.

Specifying Options for Device Checks of Type Device

To specify the options for a device check of type *Device*,

1. In the *Type* cyclic field, select *Device*.
2. (optional) In the *Subcircuit* cyclic field, select the subcircuit over which the device check is to be applied.

Note: Select *None* if you do not want to specify a subcircuit for the device check.

3. In the *Selection* field, do one of the following to specify the device names over which the device check is to be applied:
 - Enter the device names separated by spaces
 - Click the *Select* button and select a device in the schematic. The device name is added in the *Selection* field. Repeat this procedure to add more device names in the *Selection* field.
4. In the *Param Type* cyclic field, select the parameter type—*Instance Parameter*, *Op Point Parameter*, *Terminal Current*, or *Terminal Voltage*—for which you want perform the device check.

The *Param List* cyclic field displays the corresponding list of parameters sorted in alphabetical order.

5. In the *Param List* cyclic field, select the parameter to be checked in the device check.

6. If you specify the *Param Type* as `Op PointParameter(op)` and *Param List* as region, the *Safe Regions* list box is displayed. Select one or more regions in which the device can operate from the list.

Note: To select more than one region, hold down the *Shift* key (for contiguous selection) or the *Ctrl* key (for noncontiguous selection) and click the next region.

7. (optional) In the *Expression* field, enter a Measurement Description Language (MDL) expression, or add operators and functions to create an expression.
8. In the *Min and Max* fields specify the minimum and maximum values of the parameter to be checked in the device check.
9. In the *Check Windows* field, specify the time windows within which the assert is to be enabled. The field should have an even number of values in the form of `[start1,end1;start2,end2 ...]`.

Specifying Options for Device Checks of Type Model

To specify the options for a device check of type *Model*,

1. In the *Type* cyclic field, select *Model*.
2. (optional) In the *Subcircuit* cyclic field, select the subcircuit over which the device check is to be applied.

Note: Select *None* if you do not want to specify a subcircuit for the device check.

3. In the *Choices* cyclic field, select the model over which the device check is to be applied.
4. In the *Param Type* cyclic field, select the parameter type—*Instance Parameter*, *Op Point Parameter*, *Model Parameter*, *Terminal Current*, or *Terminal Voltage*—for which you want perform the device check.

The *Param List* cyclic field displays the corresponding list of parameters.

5. In the *Param List* cyclic field, select the parameter to be checked in the device check.
6. (optional) In the *Expression* field, enter a Measurement Description Language (MDL) expression, or add operators and functions to create an expression.
7. In the *Min and Max* fields specify the minimum and maximum values of the parameter to be checked in the device check.
8. In the *Check Windows* field, specify the time windows within which the assert is to be enabled. The field should have an even number of values in the form of `[start1,end1;start2,end2 ...]`.

Specifying Options for Device Checks of Type Primitive

To specify the options for a device check of type *Primitive*,

1. In the *Type* cyclic field, select *Primitive*.
2. (optional) In the *Subcircuit* cyclic field, select the subcircuit over which the device check is to be applied.

Note: Select *None* if you do not want to specify a subcircuit for the device check.

3. In the *Choices* cyclic field, select the primitive over which the device check is to be applied.
4. In the *Param Type* cyclic field, select the parameter type—*Instance Parameter*, *Op Point Parameter*, *Model Parameter*, *Terminal Current*, or *Terminal Voltage*—for which you want perform the device check.

The *Param List* cyclic field displays the corresponding list of parameters.

5. In the *Param List* cyclic field, select the parameter to be checked in the device check.
6. (optional) In the *Expression* field, enter a Measurement Description Language (MDL) expression, or add operators and functions to create an expression.
7. In the *Min and Max* fields specify the minimum and maximum values of the parameter to be checked in the device check.
8. In the *Check Windows* field, specify the time windows within which the assert is to be enabled. The field should have an even number of values in the form of [start1, end1; start2, end2 ...].

Specifying Options for Device Checks of Type Parameter

To specify the options for a device check of type *Parameter*,

1. In the *Type* cyclic field, select *Parameter*.
2. (optional) In the *Subcircuit* cyclic field, select the subcircuit over which the device check is to be applied.

Note: Select *None* if you do not want to specify a subcircuit for the device check.

3. In the *Choices* cyclic field, select the top level netlist parameter whose value is to be checked.

Note: The choices for this cyclic field would be the design variables defined in the Simulation window and the Spectre reserved parameters `temp`, `tnom`, `scale`, `scalem`,

freq, and time.

4. In the *Min and Max* fields specify the minimum and maximum values of the parameter to be checked in the device check.
5. In the *Check Windows* field, specify the time windows within which the assert is to be enabled. The field should have an even number of values in the form of [start1,end1;start2,end2 ...].

Specifying Options for Device Checks of Type Expression

To specify the options for a device check of type *Expression*,

1. In the *Type* cyclic field, select *Expression*.
2. (optional) In the *Subcircuit* cyclic field, select the subcircuit over which the device check is to be applied.

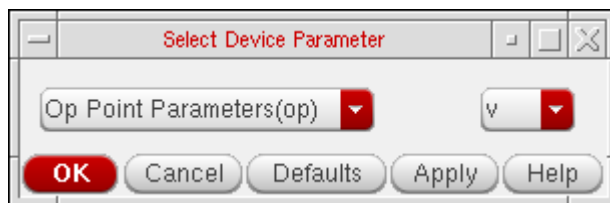
Note: Select *None* if you do not want to specify a subcircuit for the device check.

3. In the *Expression* field, enter a Measurement Description Language (MDL) expression, or add operators and functions to create an expression.

You can also create expressions by selecting devices from the schematic, by doing the following:

- a. Click the *Select* button next to the *Expression* field to display the schematic window.
- b. Select an instance on the schematic.

The Select Device Parameter form appears.



- c. In the cyclic field on the left, select the type of parameter you want to use in the expression.

The cyclic field on the right displays the corresponding list of parameters.

- d. In the cyclic field on the right, select the parameter you want to use in the expression.
- e. Click *OK*.

The full schematic name for that instance followed by `:param_name` is appended to the existing text in the *Expression* field. For example, the instance name with its operating point parameter `v` will be written as `I3.R3:v`. You can then add operators and functions to create an expression.

4. In the *Min and Max* fields specify the minimum and maximum values of the parameter to be checked in the device check.
5. In the *Check Windows* field, specify the time windows within which the assert is to be enabled. The field should have an even number of values in the form of `[start1,end1;start2,end2 ...]`.

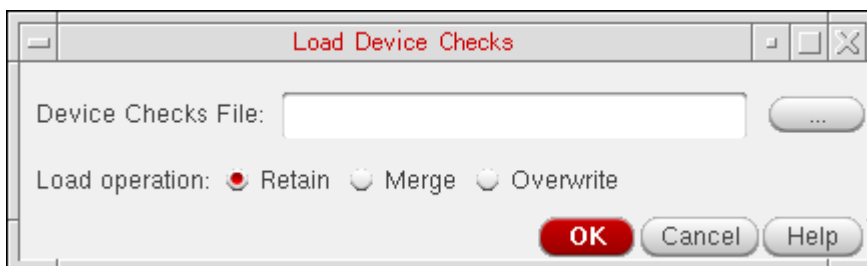
Loading Predefined Device Checks from a File

Typically, PDK developers or the CAD group propose some standard device checks to be used in designs. These groups can use the Device Check Specification form to create the standard device checks and save them to a file (see [Saving Device Checks to a File](#) on page 416). These predefined device checks can then be loaded in any design.

To load predefined checks from a file,

1. On the [Device Check Specification](#) form, click the *Load* button.

The Load Device Checks form appears.



2. In the *Device Checks File* field, enter the path to the device check file, or click the browse button to select the file.
3. In the *Load operation field*, select the mode for loading device checks.

Select

To

Retain

Retain existing device checks that have the same name as device checks in the file, and load only other device checks from the file.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

- Merge* Overwrite existing device checks that have the same name as device checks in the file, and also load other device checks from the file.
- Overwrite* Delete all existing device checks and load the device checks in the file.

4. Click *OK*.

All the device checks in the file are loaded and displayed in the Device Check Specification form.

Note: If a device check with the same name exists in the Device Check Summary table, a new name, *check_n*, where *n* is an incremental number, is assigned to the device check. For example, if a device check with the same name exists, the name *check1* is assigned to the new device check.

5. (optional) Specify additional options for the device checks in the following columns in the Device Check Summary table:

Duration Double-click on the *Duration* column and specify the time period in seconds over which the check has to be violated before a warning is displayed. Applicable to transient analyses only.

Severity Click on the *Severity* column and select the severity level— *Notice*, *Warning*, *Error*, *Fatal* or *None*—of the message that will be displayed when Spectre reports violations for the check in the simulation log file and the Violations Display form.

- If the severity level is *Notice* or *Warning*, simulation continues after the message is displayed.
- If the severity level is *Error*, the Spectre circuit simulator aborts the analysis when the first error-level violation occurs.
- If the severity level is *Fatal*, the Spectre circuit simulator aborts the simulation when the first fatal-level violation occurs.

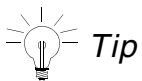
The default setting is *Warning*.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

Color	<p>Click on the <i>Color</i> column and select the color to be used when highlight the devices on the schematic that failed the device check.</p> <p>If you do not specify the color, the following default colors are used, depending on the severity level specified for the device check:</p> <ul style="list-style-type: none">■ Yellow color for Warning severity level.■ Red color for Error severity level.■ Blue color for Fatal severity level.■ Orange color for Notice severity level.■ Blue color for None severity level.
Info	<p>Click on the <i>Info</i> column and do one of the following:</p> <ul style="list-style-type: none">■ Select <i>Yes</i> to ignore the <i>Min</i>, <i>Max</i>, and <i>Duration</i> options specified for the device check. Only the parameter value specified for the device check will be displayed in the simulation log file.■ Select <i>No</i> if you do not want to ignore the <i>Min</i>, <i>Max</i>, and <i>Duration</i> options specified for the device check. <p>The default setting is <i>no</i>.</p>
Message	<p>Double-click on the <i>Message</i> column and enter the message to be displayed in the simulation log file and the <u>Violations Display</u> form if the device check fails.</p>

6. Click *OK* or *Apply* to save the changes.



Alternatively, you can define the device checks in the asserts file and load that file using the *Definition Files* option in the *Setup – Simulation Files Setup* form. By default, the device checks defined using the asserts file are displayed only in the log files, and not in the *Violations Display* form. To display the device checks in the *Violations Display* form, select the *Save Asserts Info* checkbox in the *Outputs – Save All* form.

Disabling and Enabling Individual Device Checks

You can disable and enable individual device checks. The *Enabled* column in the Device Check Summary table indicates whether a device check is enabled or disabled.

To disable a device check,

- ▶ Clear the check box next to the device check in the *Enabled* column in the Device Check Summary table.

To enable a device check,

- ▶ Select the check box next to the device check in the *Enabled* column in the Device Check Summary table.

For information about disabling and enabling device checking, see [Enabling and Disabling Device Checking](#) on page 403

Modifying Device Checks

To modify a device check,

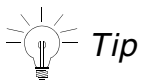
1. Click on the device check in the Device Check Summary table.

The device check information is displayed in the *Create Check* group box.

2. Make the required changes and click *Change*.

The changes are displayed in the Device Check Summary table.

3. Click *OK* or *Apply* to save the changes.



You can also modify a device check by double-clicking on the columns in the Device Check Summary table. For example, you can double-click on the *Expr/Param* column to modify an expression or parameter.

Saving Device Checks to a File

You can save the device checks displayed in the Device Check Specification form to a file. You can then use the file to quickly add device checks for other designs. For more information about loading device checks, see [Loading Predefined Device Checks from a File](#) on page 413.

1. Click the *Save* button.

The Select a File form appears.

2. Specify the name and location of the file and click *Save*.

Deleting Device Checks

To delete a device check,

1. Select the device check in the Device Check Summary table.
2. Click the *Delete Check* button.

Specifying Global Device Check Options

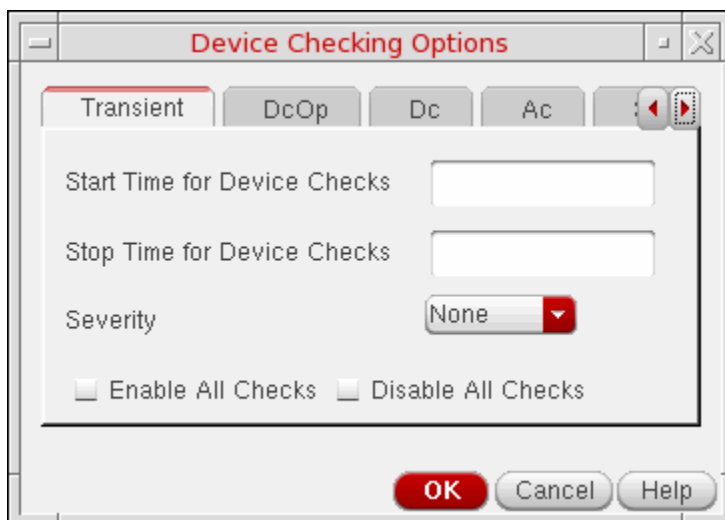
You can specify the global device check options such as start and stop times for a transient analysis and the default severity for transient, DC and DC sweep analyses.

1. From the Simulation window, choose *Simulation – Device Checking*.

The Device Check Specification form appears.

2. Click *Options*.

The Device Checking Options form appears.



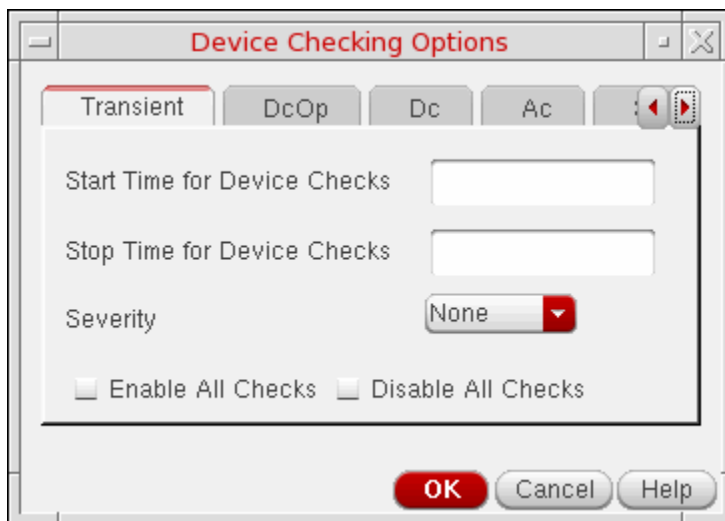
You can use this form to specify the global device check options for transient, DC and DC sweep analyses. For more information, see the following sections:

- ❑ [Specifying Device Checking Options for Transient Analyses](#) on page 418
- ❑ [Specifying Device Checking Options for DC Analyses](#) on page 419
- ❑ [Specifying Device Checking Options for DC Sweep Analyses](#) on page 420
- ❑ [Specifying Device Checking Options for AC Analyses](#) on page 421
- ❑ [Specifying Device Checking Options for SP Analyses](#) on page 422
- ❑ [Specifying Device Checking Options for Noise Analyses](#) on page 423
- ❑ [Specifying Device Checking Options for PZ Analyses](#) on page 424

Specifying Device Checking Options for Transient Analyses

To specify the device checking options for transient analyses,

1. On the Device Checking Options form, click the *Transient* tab page.



2. In the *Start Time for Device Checks* field, specify the beginning time at which device checks are to be enabled or disabled for transient analyses.
3. In the *Stop Time for Device Checks* field, specify the end time at which device checks are to be enabled or disabled for transient analyses.
4. In the *Severity* cyclic field, select the default severity level— *Notice*, *Warning*, *Error* or *None*—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for transient analyses.

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the Device Check Specification form will be used.

5. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for transient analyses.

Note: If this check box is not selected, only the device checks that are enabled for transient analyses in the Device Check Specification form are used for transient analyses.

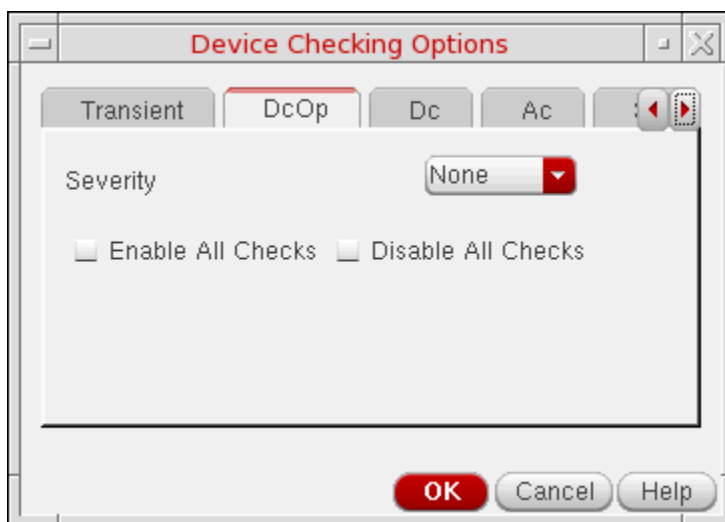
- Select the *Disable All Checks* check box to disable all the device checks for transient analyses.

6. Click *OK*.

Specifying Device Checking Options for DC Analyses

To specify the device checking options for DC analyses,

1. On the Device Checking Options form, click the *DcOp* tab page.



2. In the *Severity* cyclic field, select the default severity level— `Notice`, `Warning`, `Error` or `None`—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for DC analyses.

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the Device Check Specification form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for DC analyses.

Note: If this check box is not selected, only the device checks that are enabled for DC analyses in the Device Check Specification form are used for DC analyses.

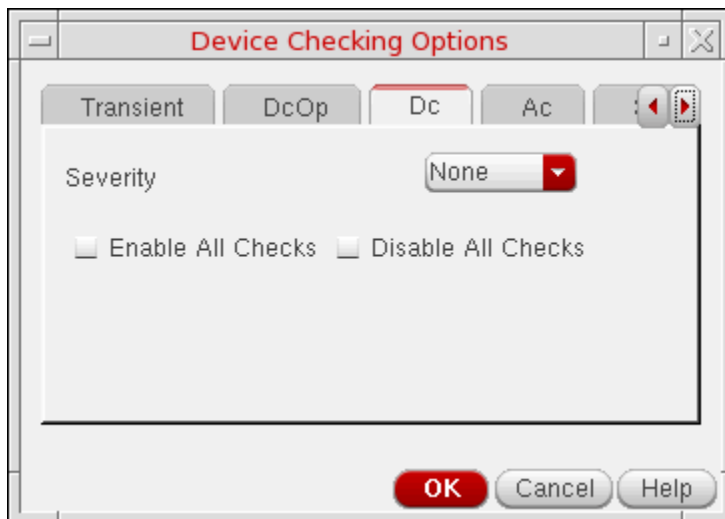
- Select the *Disable All Checks* check box to disable all the device checks for DC analyses.

4. Click *OK*.

Specifying Device Checking Options for DC Sweep Analyses

To specify the device checking options for DC sweep analyses,

1. On the Device Checking Options form, click the *Dc* tab page.



2. In the *Severity* cyclic field, select the default severity level—`Notice`, `Warning`, `Error` or `None`—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for DC sweep analyses.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the Device Check Specification form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for DC sweep analyses.

Note: If this check box is not selected, only the device checks that are enabled for DC sweep analyses in the Device Check Specification form are used for DC sweep analyses.

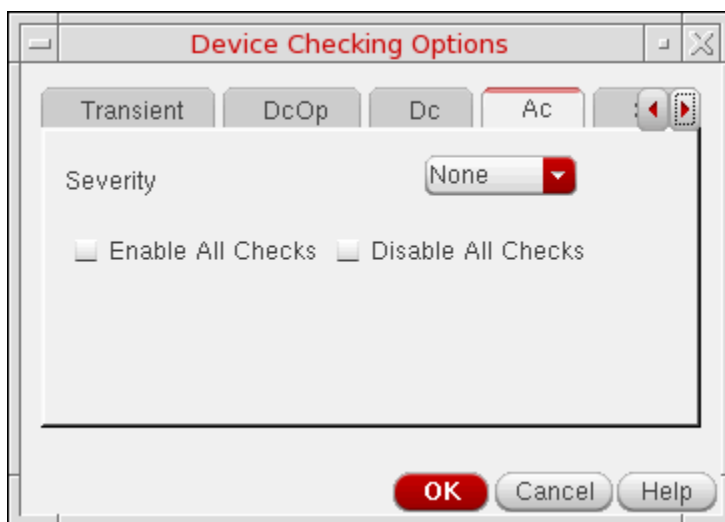
- Select the *Disable All Checks* check box to disable all the device checks for DC sweep analyses.

4. Click *OK*.

Specifying Device Checking Options for AC Analyses

To specify the device checking options for AC analyses,

1. On the Device Checking Options form, click the *Ac* tab page.



2. In the *Severity* cyclic field, select the default severity level— *Notice*, *Warning*, *Error* or *None*—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for AC analyses.

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the Device Check Specification form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for AC analyses.

Note: If this check box is not selected, only the device checks that are enabled for AC analyses in the Device Check Specification form are used for AC analyses.

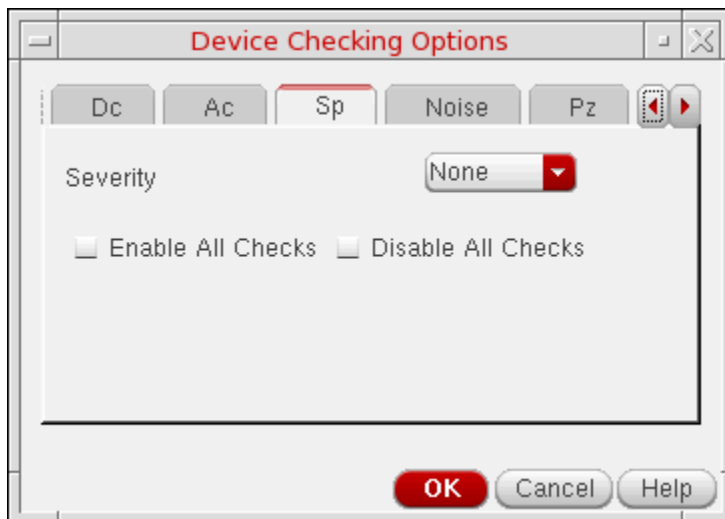
- Select the *Disable All Checks* check box to disable all the device checks for AC analyses.

4. Click *OK*.

Specifying Device Checking Options for SP Analyses

To specify the device checking options for SP analyses,

- 1.** On the Device Checking Options form, click the *Sp* tab page.



- 2.** In the *Severity* cyclic field, select the default severity level—`Notice`, `Warning`, `Error` or `None`—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for SP analyses.

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the *Device Check Specification* form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for SP analyses.

Note: If this check box is not selected, only the device checks that are enabled for SP analyses in the *Device Check Specification* form are used for SP analyses.

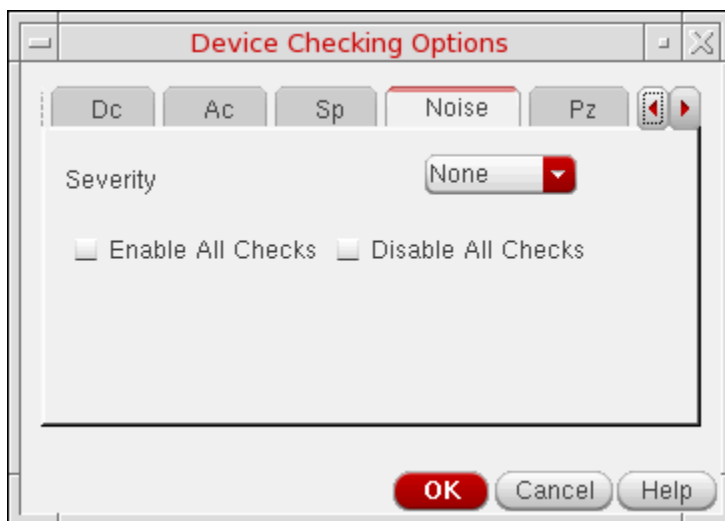
- Select the *Disable All Checks* check box to disable all the device checks for SP analyses.

4. Click *OK*.

Specifying Device Checking Options for Noise Analyses

To specify the device checking options for Noise analyses,

- 1.** On the Device Checking Options form, click the *Noise* tab page.



- 2.** In the *Severity* cyclic field, select the default severity level—`Notice`, `Warning`, `Error` or `None`—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for Noise analyses.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the *Device Check Specification* form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for Noise analyses.

Note: If this check box is not selected, only the device checks that are enabled for Noise analyses in the *Device Check Specification* form are used for Noise analyses.

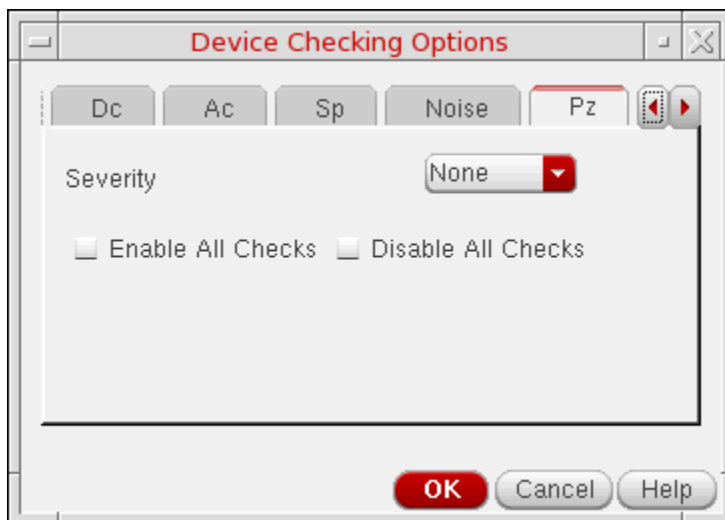
- Select the *Disable All Checks* check box to disable all the device checks for Noise analyses.

4. Click *OK*.

Specifying Device Checking Options for PZ Analyses

To specify the device checking options for PZ analyses,

1. On the Device Checking Options form, click the *Pz* tab page.



2. In the *Severity* cyclic field, select the default severity level— `Notice`, `Warning`, `Error` or `None`—of the message that will be displayed when Spectre reports violations for the device checks that are enabled for PZ analyses.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

The specified severity level overrides the severity level specified for individual device checks in the [Device Check Specification](#) form.

Note: If you set the severity to `None`, the severity level specified for individual device checks in the *Device Check Specification* form will be used.

3. Do one of the following:

- Select the *Enable All Checks* check box to enable all the device checks for PZ analyses.

Note: If this check box is not selected, only the device checks that are enabled for PZ analyses in the *Device Check Specification* form are used for PZ analyses.

- Select the *Disable All Checks* check box to disable all the device checks for PZ analyses.

4. Click *OK*.

Specifying Options for Writing Violations Information

By default, violations information is written in the simulation log file and in the netlist. If you want violations information to be written to a separate file instead of displaying it in the simulation log file, do the following:

1. From the Simulation window, choose *Simulation – Options – Analog*.

The Simulator Options form appears.

2. Click the *Check* tab page.

3. In the *checklimitfile* field, specify the name and path to the file.

When a filename is specified in this field, the simulation log file displays the following text:

```
All assert violations will be written to file '<file_name>'
```

4. In the *checklimitdest* field, do one of the following:

Select	To
file	Write violations information only in the specified file.
psf	Write violations information only in the netlist.
both	Write violations information in the specified file and in the netlist.

5. Click *OK*.

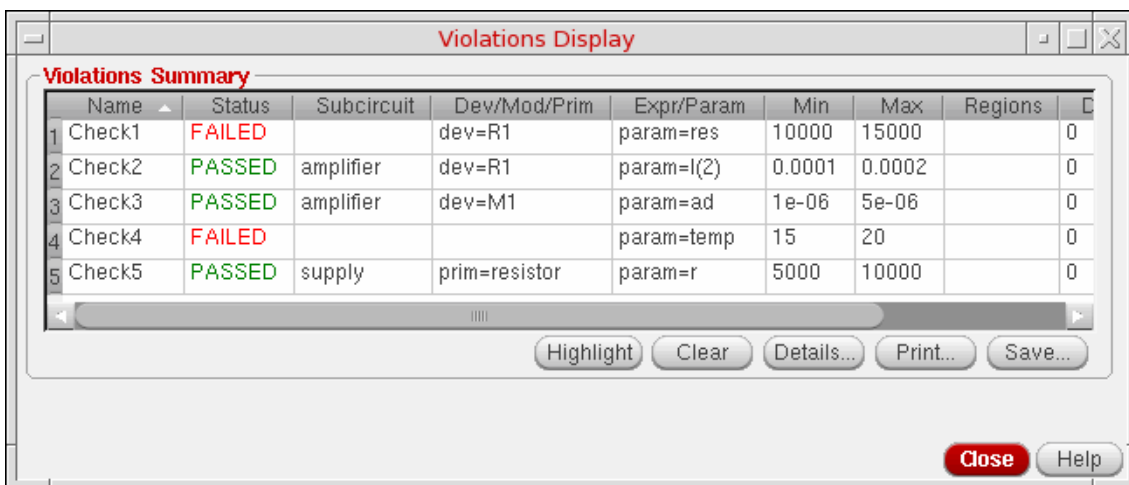
Viewing, Printing, and Saving Device Check Violations

A violation occurs when the value of a device, parameter, or expression that you have defined as a device check falls outside the specified operating range.

To view, print, and save device check violations,

- ▶ From the Simulation window, choose *Results – Violations Display*.

The Violations Display form appears displaying the status of device checks.



For more information about using this form, see the following sections:

- ❑ [Highlighting Device Check Violations on the Schematic](#) on page 427
- ❑ [Dehighlighting Device Check Violations on the Schematic](#) on page 428
- ❑ [Viewing the Details of Device Check Violations](#) on page 428
- ❑ [Saving the Details of Device Check Violations](#) on page 429
- ❑ [Printing Device Check Violations](#) on page 429
- ❑ [Hiding and Showing Columns](#) on page 430
- ❑ [Saving Violation Results](#) on page 431



Tip

To modify a device check, double-click on the row for the device check. The device check is displayed in the Device Check and Operating Region Specification form. For more information about modifying device checks, see [Modifying Device Checks](#) on page 416.

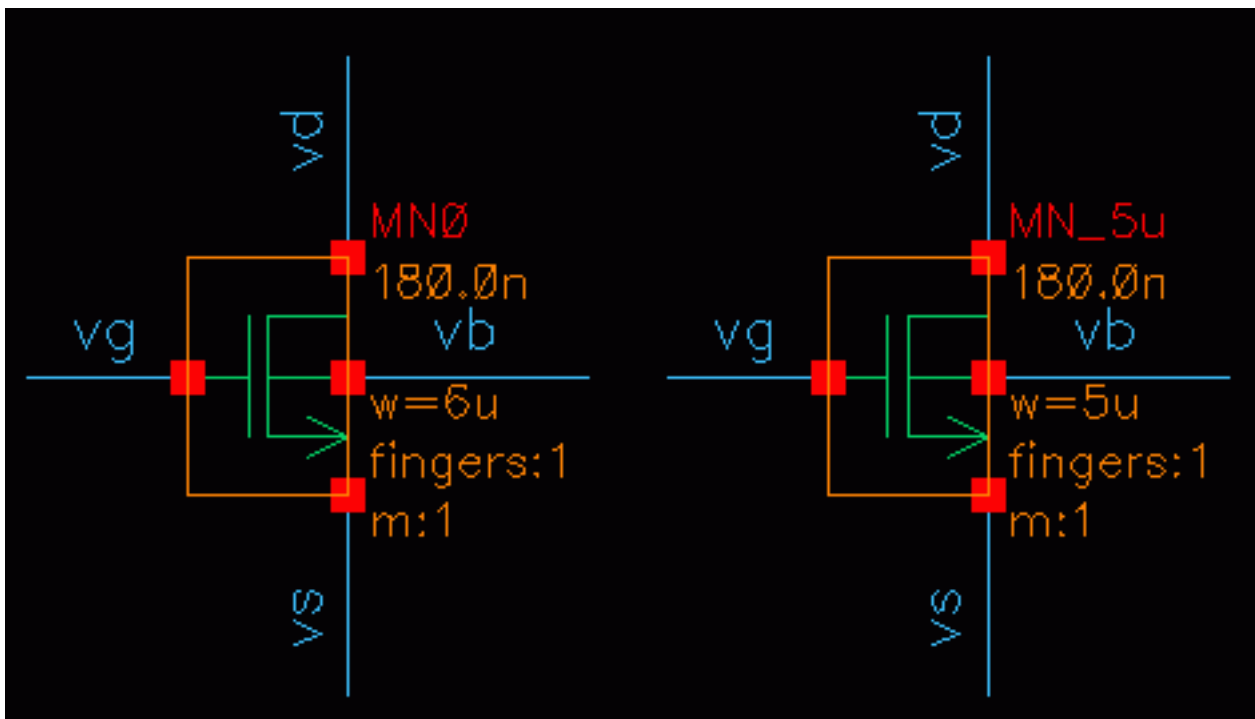
Highlighting Device Check Violations on the Schematic

You can highlight all the devices on the schematic that failed a device check. This helps you to quickly correct device check violations.

To highlight the devices that failed a device check,

1. Click on the row for a device check that has failed.
2. Click the *Highlight* button.

The schematic is opened and all the devices that failed the device check are highlighted using the color specific for the device check. For example, in the following schematic, two devices that failed a device check are highlighted in orange color.



Note: To highlight the violations for multiple device checks, hold down the *Shift* key (for contiguous selection) or the *Ctrl* key (for noncontiguous selection) and click the next device check to add more device checks to the selection set, then click the *Highlight* button.

Dehighlighting Device Check Violations on the Schematic

To dehighlight device check violations on the schematic,

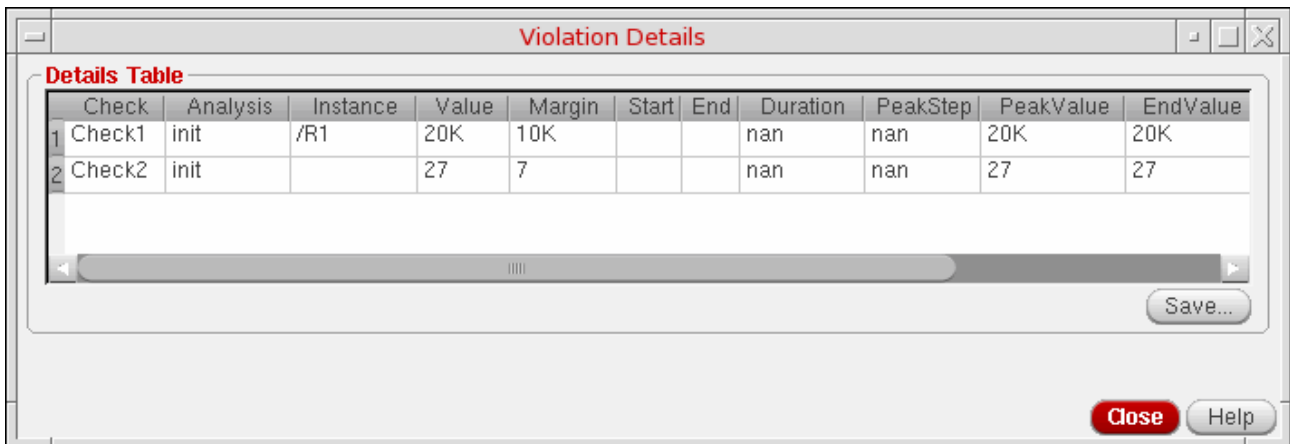
- ➔ Click the *Clear* button.

Viewing the Details of Device Check Violations

To view the details of the violations for a device check,

1. Click on the row for a device check that has failed.
2. Click the *Details* button.

The *Violation Details* form is displayed with the details of the violations for the device check.



Note: To view the details of violations for multiple device checks, hold down the *Shift* key (for contiguous selection) or the *Ctrl* key (for noncontiguous selection) and click the next device check to add more device checks to the selection set, then click the *Details* button.

See also:

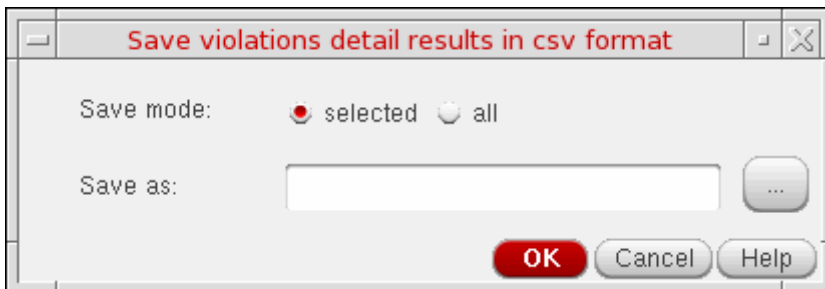
- [Hiding and Showing Columns](#) on page 430


Saving the Details of Device Check Violations

To save the details of the violations:

1. Click on a failed device check row that you want to save. You can select multiple rows by using the `Ctrl` key or the `Shift` key.
2. Click the *Save* button.

The *Save violations detail in csv format* form appears.



3. Choose the required *Save mode*. The *selected* radio button saves the violation results for only the selected rows. However, the *all* radio button saves the violation results for all the rows.
4. Type the name of the file in the *Save as* text box. Alternatively, you can use the *Browse* button () to select the location and name of the file in which you want to save the violation results.
5. Click *OK*.

Printing Device Check Violations

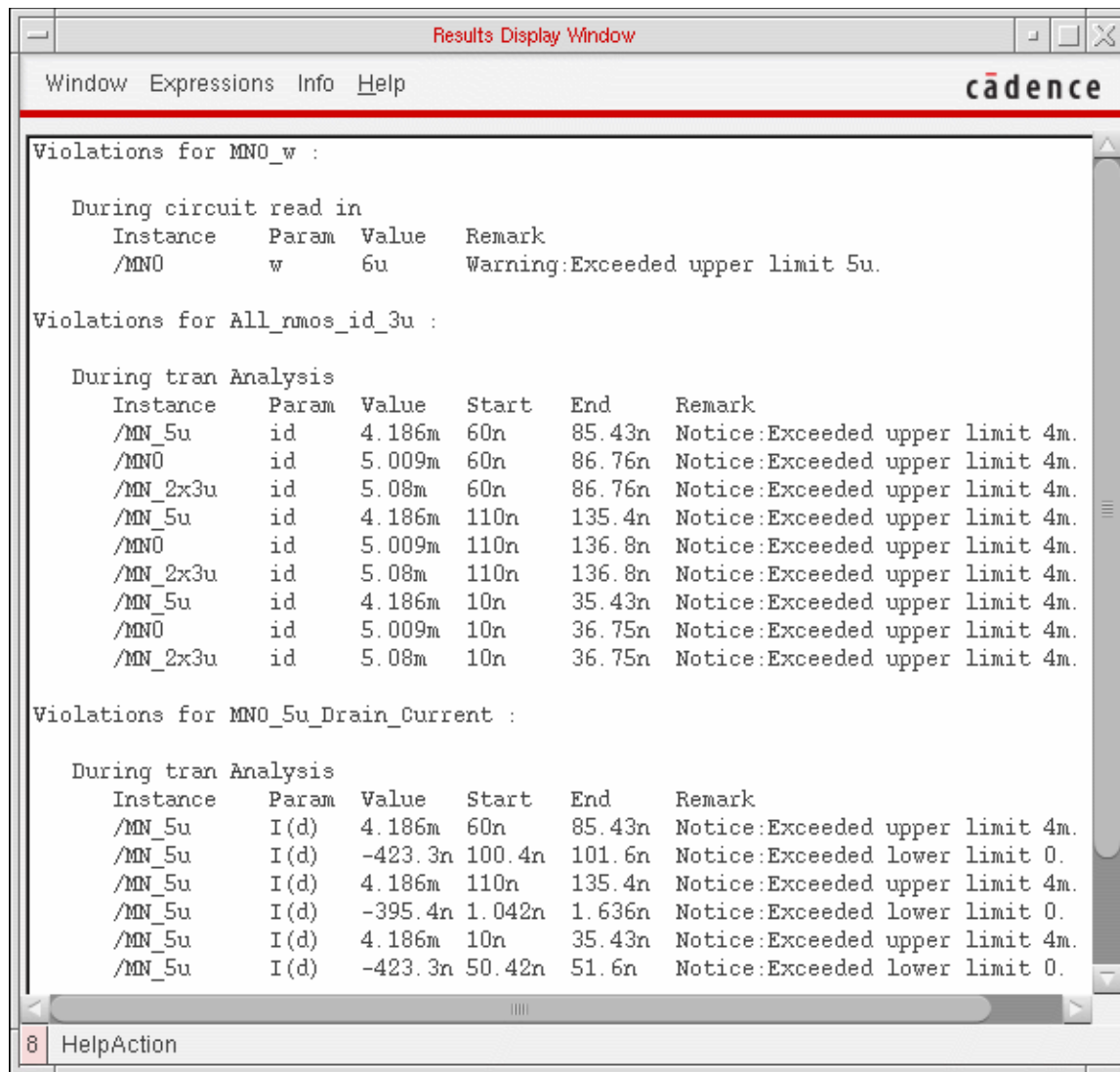
To print a summary of the violations for a device check,

1. Click on the row for a device check that has failed.
2. Click the *Print* button.

A summary of the violations are displayed in the Results Display Window.

Virtuoso Analog Design Environment L User Guide

Running a Simulation



3. Choose *Window – Print* to print the information.

For more information, see [Printing Results](#) on page 515.

Note: To print the summary of violations for multiple device checks, hold down the *Shift* key (for contiguous selection) or the *Ctrl* key (for noncontiguous selection) and click the next device check to add more device checks to the selection set, then click the *Print* button.

Hiding and Showing Columns

You can hide and show the required columns in the following forms:

- Device Check Specification form.
- Violations Display form
- Violation Details form

To hide a column that is currently displayed,

1. Right-click on a column name to display a pop-up menu.

A tick mark next to a column name in the pop-up menu indicates that the column is currently displayed.

2. Choose the name of the column you want to hide.

To show a column that is currently not displayed,

- Right-click on a column name and choose the name of the column you want to display from the pop-up menu.

Saving Violation Results

You can save the results of the violations in a CSV file to sort or filter the critical results.

To save the violation results:

1. Click the row for a device check that you want to save. You can select multiple rows by using the `Ctrl` key or the `Shift` key.
2. Click the *Save* button.


The *Save violations results in csv format* form appears.



3. Choose the required *Save mode*. The *selected* radio button saves the violation results for only the selected rows. However, the *all* radio button saves the violation results for all the rows.

Virtuoso Analog Design Environment L User Guide

Running a Simulation

4. Type the name of the file in the *Save as* text box. Alternatively, you can use the *Browse* button () to select the location and name of the file in which you want to save the violation results.
5. Click *OK*.

Parameterization Support

This chapter describes the parameterization support in Virtuoso[®] Analog Design Environment.

- [About Parameterization Support](#) on page 433
- [Support for VAR Syntax](#) on page 433
- [Usage of VAR Syntax](#) on page 434
- [ADE Forms for VAR Support](#) on page 435
- [Setup Examples](#) on page 436

About Parameterization Support

A parameter is a characteristic of a component that has special meaning to the component. Different instances of the same component might not always use the same set of parameters and their values. You can use design variables to specify parameter values. For more information, see:

- [Design Variables and Simulation](#) on page 138
- [Parameter Setup information in Virtuoso Analog Design Environment XL User Guide](#)

Support for VAR Syntax

Virtuoso Analog Design Environment provides a way to specify design variables using the *Editing Design Variables* form.

Starting with IC 6.1, you can set variables in various ADE forms using VAR syntax. When the variable is specified in these formats, ADE automatically adds it as one of the design variable. You can also specify design variables in terms of other design variables using this syntax. For example, if you need to set the "Stop Time" in Transient Analysis form as a variable, then VAR("STOP") can be used. The variable "STOP" is automatically added as a design

variable in Analog Design Environment. A model file can also be specified using `VAR("MY_MODEL")` in the *Model File Setup* form. For more details, see ["Usage of VAR Syntax"](#) on page 434.

Usage of VAR Syntax

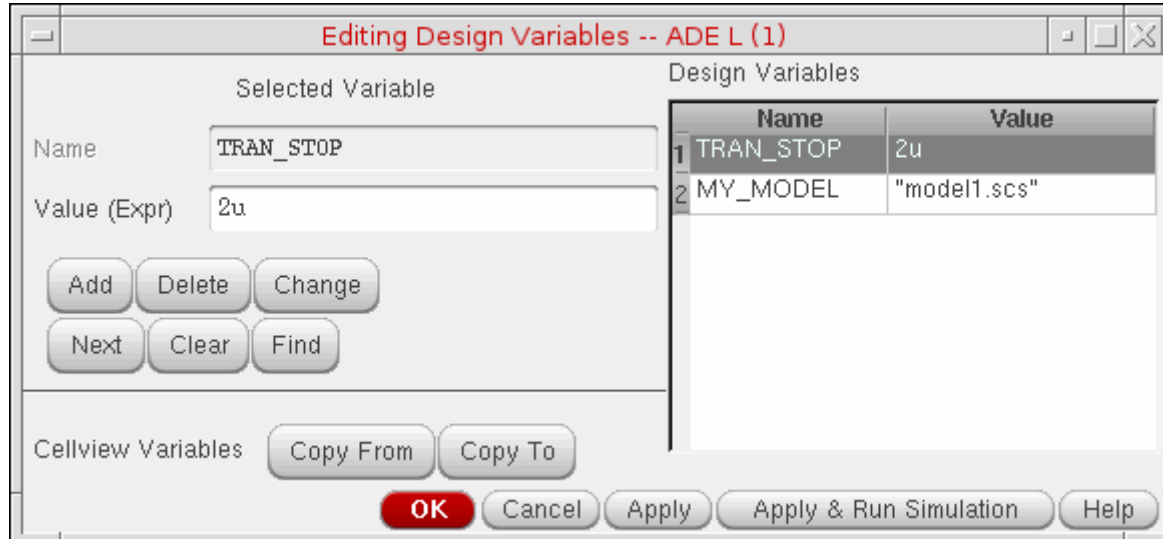
The support for VAR syntax is simulator independent and works in all default Cadence simulator interfaces including Spectre, UltraSim, ams, SpectreVerilog (IC6.1.6 only), and UltraSimVerilog (IC6.1.6 only). It also works for custom third party simulator interfaces, provided the integration uses Cadence recommended guidelines.

The VAR syntax supports both numeric and string parameters. The value can be provided directly in the design variable section.

`VAR("TRAN_STOP") = 2u`, is an example of handling a numeric field.

For string variables value needs to be specified in double quotes.

`VAR("MY_MODEL") = "model1.scs"`, is an example of handling a string field.



The VAR approach recognizes the variable and passes the design variable to the control file. This approach can further be extended to allow evaluation of SKILL functions along with variables. However, they need to be pre-defined and evaluated in current SKILL context. For example, let us consider a case where the user specified Stop Time in terms of a VAR, numeric and a SKILL function.

```
STOP = VAR("TRAN_STOP") + 2u + _myFunc()
```

Virtuoso Analog Design Environment L User Guide

Parameterization Support

where `_myFunc()` in SKILL evaluates to a value, say `5u` and `VAR("TRAN_STOP")` equals to `2u`, then `Stop Time` will be `9u`.

The screenshot shows two windows from the Virtuoso Analog Design Environment. The 'Design Variables' window on the left contains a table with three rows:

	Name	Value
1	CAP	800f
2	TRAN_STOP	2u
3	MY_MODEL	"model1.scs"

The 'Analyses' window on the right contains a table with three rows:

	Type	Enable	Arguments
1	ac	<input checked="" type="checkbox"/>	100 150M 20 Logarithmic Points Per Decade S...
2	dc	<input checked="" type="checkbox"/>	t
3	tran	<input checked="" type="checkbox"/>	0 VAR("TRAN_STOP") + 2u + _myFunc()

Therefore, the final value of above entry on the form would be entered and passed to the control file.

Note: Do not use simulator reserved keywords such as `save`, `return`, `real` and so on as variables inside the VAR function. Details on these keywords for spectre direct are listed under “spectre -help keywords”.

Note: Normal ADE design variables and the VAR variables should never be combined together on the form. For example, `TRAN_STOP = 2u + MYVAR` is not allowed, instead use `TRAN_STOP = 2u + VAR("MYVAR")`.

ADE Forms for VAR Support

Following Analog Design Environment forms support VAR syntax:

- The *Environment Options* form supports this syntax for Switch View List and Stop View List.
- The *Model Library Setup* form supports this syntax for both model files and sections.
- The *Simulation File Setup* form supports: Include Path, Definition Files, and Stimulus Files. It also supports VCD, VEC, EVCD for spectre interface.
- The *Choosing Analysis* form supports this syntax for different analysis. For example, Transient, AC, DC, XF, Sensitivity Analysis etc. You can also use these syntax in analysis fields and options.
- The *Save Options* form supports this syntax for subcircuit probe level field.
- The *Simulator Options* form supports this syntax for analog options. For Example, scale, temperature etc. Similarly, you can enter variables in digital and mixed-signal options for a mixed-signal interface.

Setup Examples

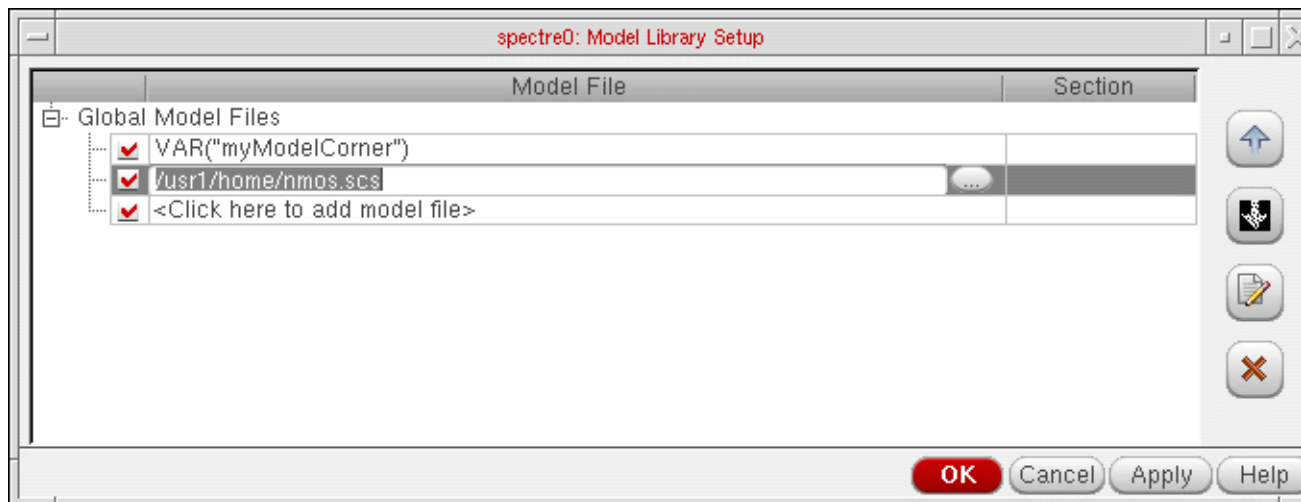
Model File Setup Example

You can use VAR syntax to setup a model file parameter. For details on using model files, see [Model Files in the Virtuoso Analog Design Environment](#) on page 152. You can setup a variable directly in Model Libraries path as described below.

To set the variable:

1. Choose *Setup - Model Libraries*.

The *Model Library Setup* form appears.



2. Set model file: `VAR ("myModelCorner")` and click *Apply*.

Applying the change will automatically create a new design variables called `myModelCorner`. Alternatively, you can also select the existing model file by clicking the Browse button and selecting `.scs` file. The Browse button calls the browser to find the model files easily and place them in the Model File field.

3. You can edit the set variable by selecting *Variables - Edit*.
4. Select the Design Variable and enter the expressing in the Value field. For example, set `myModelCorner` to `"cornerModels.scs"`.

Name	Value
myModelCorner	"cornerModels.scs"

Transient Analysis Setup Example

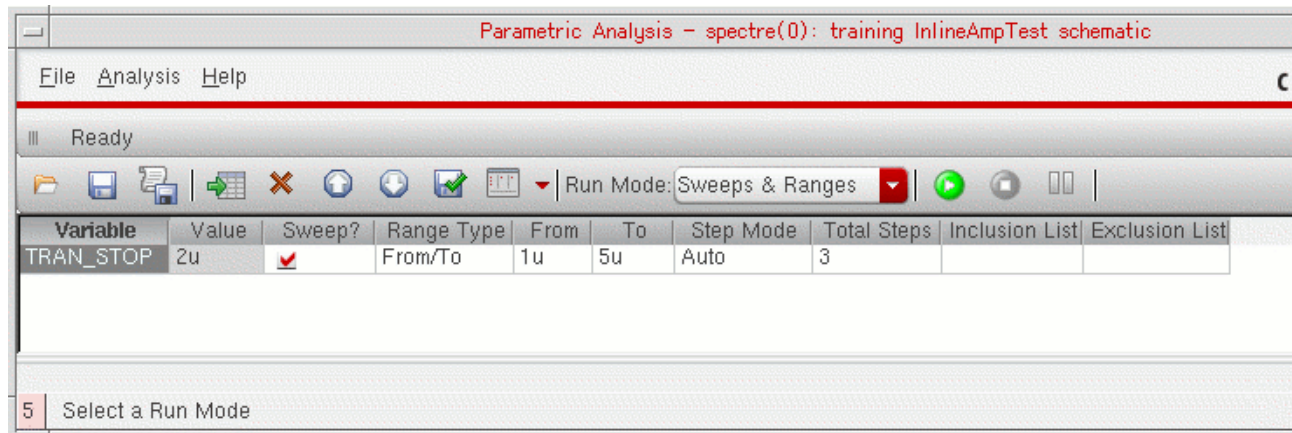
You can use VAR syntax to specify the Stop Time in Choosing Analysis form.

1. Select *Analysis - Choose...*
2. Enter the *Stop Time as VAR("TRAN_STOP")*.
3. Click *OK* and the Design Variable would be set.
4. Choose *Simulation - Netlist and Run*.

Running a Sweep Analysis using VAR()

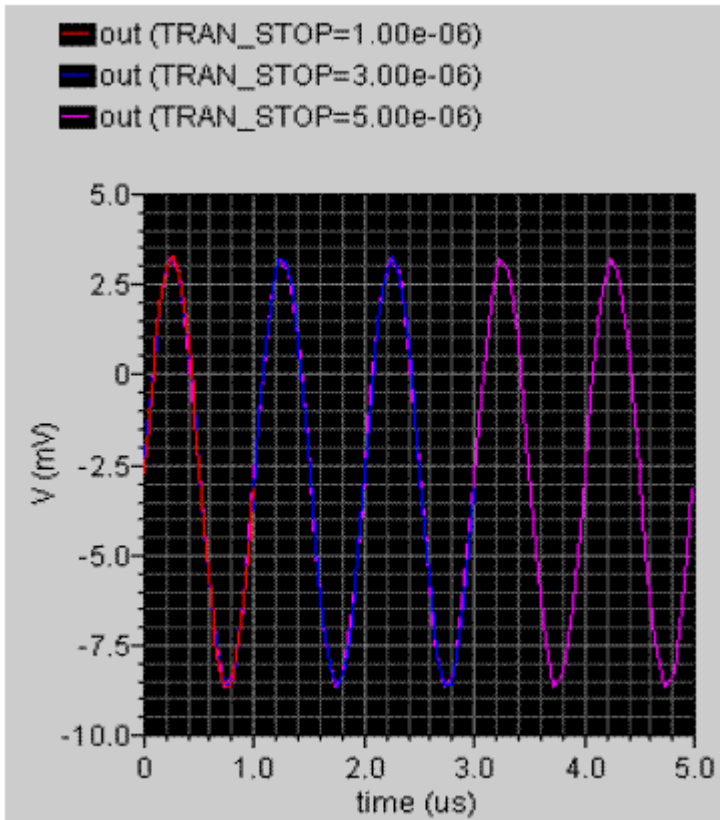
You can sweep over the variable using Parametric Analysis Tool to meet your design requirements.

1. Select *Tools - Parametric Analysis...*
2. Insert a new row and select `TRAN_STOP` in the *Variable* field.
3. To specify appropriate Range Type and Step Controls, specify the values in the *From*, *To* and *Total Steps* fields as 1u, 5u, and 3.



4. After specifying the range, select *Analysis - Start*.

The parametric analysis is run and the result is plotted in the graph window.



In the above example, notice that the TRAN_STOP is plotted as three waves with different values.

For more information on Parametric Analysis, see [Chapter 9, “Analysis Tools,”](#).

Switch View List Setup Example

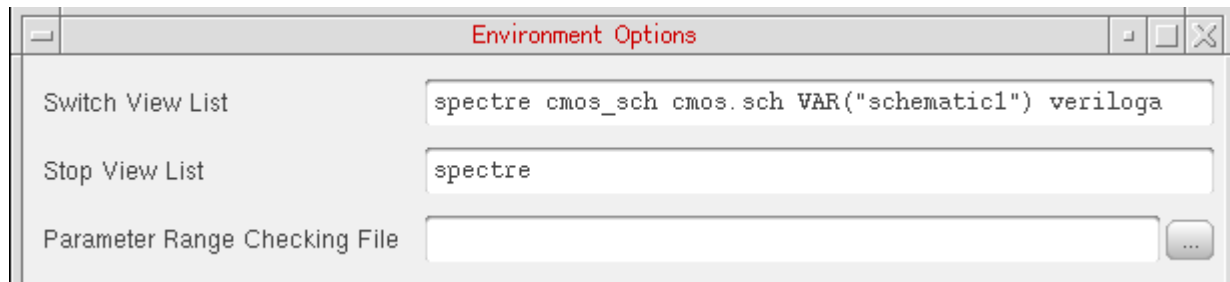
You can use VAR syntax to specify the Switch View List in Environment Options form.

1. Select *Setup - Environment...*

Virtuoso Analog Design Environment L User Guide

Parameterization Support

2. In the *Environment Options* form, set the *Switch View List* with view names specified in VAR syntax. For example, VAR("schematic1").



The variable, `schematic1`, appears in the *Design Variables* pane.

3. Provide a value for `schematic1` in the *Design Variables* pane. Note that you can specify only string values (specified in double quotes) for this variable.

You can also substitute values to the variable during parametric analysis.

Virtuoso Analog Design Environment L User Guide

Parameterization Support

Helping a Simulation to Converge

This chapter describes how you can help a troublesome simulation to converge. Select topics from the following list to view more information.

- [Commands for Forcing Convergence](#) on page 441
- [Selecting Nodes and Setting their Values](#) on page 443
- [Releasing Voltages](#) on page 445
- [Changing Voltages](#) on page 445
- [Saving and Restoring Node Voltages](#) on page 446
- [Highlighting Set Nodes](#) on page 447
- [Storing a Solution](#) on page 447
- [Restoring a Solution for Spectre](#) on page 448
- [Form Field Descriptions](#) on page 450

Commands for Forcing Convergence

You use the commands in the *Simulation – Convergence Aids* menu to help the simulator find a solution when it fails to achieve convergence. Once you get the simulation to converge, you can save the DC and Transient solutions. When you resimulate, you can save time by restoring the saved solutions.

There are three commands to help the simulator find a solution:

- *Node Set*, which provides an initial guess for nodes in any DC analysis or the initial condition calculation for the transient analysis
- *Initial Condition*, which provides initial conditions for nodes in the transient analysis
- *Force Node*, which sets the voltage on a node and locks it at that voltage during the entire simulation

Note: Refer to the simulator manual for specific details about the commands that help circuits converge. Not all simulators support these three commands, and simulators implement these methods differently.

Node Set

To set an initial DC voltage on selected nodes, use the *Simulation – Convergence Aids – Node Set* command.

For Spectre, the node set is used to provide an initial guess for nodes in any DC analysis or the initial condition calculation for the transient analysis. It netlists to

```
nodeset node=value
```

For more information, see the *Node Sets (nodeset)* section in the *Other Simulation Topics* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

For other simulators, the *Node Set* command is equivalent to

```
.NODESET v(node)=value
```

Initial Conditions

To set an initial transient voltage on selected nodes, use the *Simulation – Convergence Aids – Initial Condition* command.

For Spectre, initial conditions are used to provide initial conditions for nodes in the transient analysis. Initial conditions are accepted only for inductor currents and node voltages where the nodes have a path of capacitors to ground. This is netlisted to

```
ic node=value
```

For more information, see the *Initial Conditions (ic)* section in the *Other Simulation Topics* chapter of the *Virtuoso Spectre Circuit Simulator Reference*.

For other simulators, the *Initial Condition* command is equivalent to

```
.IC node=value
```

Force Node

To set a node to a specific voltage throughout the simulation, use the *Simulation – Convergence Aids – Force Node* command.

For details on this command, see the reference manual for the simulator that you use.

One way to use this feature is to store the DC solution from a simulation with *Force Node* active, remove the *Force Node* setting, restore the DC solution, and run another simulation.

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

HspiceD Convergence Aids

The *Convergence Aids* submenu appears and works slightly different.

Node Set (.NODESET) ...

Initial Condition (.IC) ...

Force (.DCVOLT) ...

- *Convergence Aids – Node Set (.NODESET)* initializes specified nodal voltages for a DC operating point analysis. The `.NODESET` statement is generally used to correct convergence problems in a DC analysis. Setting nodes in the circuit to values that are close to the actual DC operating point solution enhances the convergence of the simulation. The *Select Node Set* form works in the same way as the Spectre Direct interface. The netlist will contain the `.NODESET` statement line.
- *Convergence Aids – Initial Condition (.IC) or Convergence Aids– Force (.DCVOLT)* sets the transient initial conditions. The initialization depends on whether the UIC parameter is included in the `.TRAN` analysis statement. If the UIC parameter is specified in the `.TRAN` statement, the Hspice simulator does not calculate the initial DC operating point. Consequently, the transient analysis is entered directly.

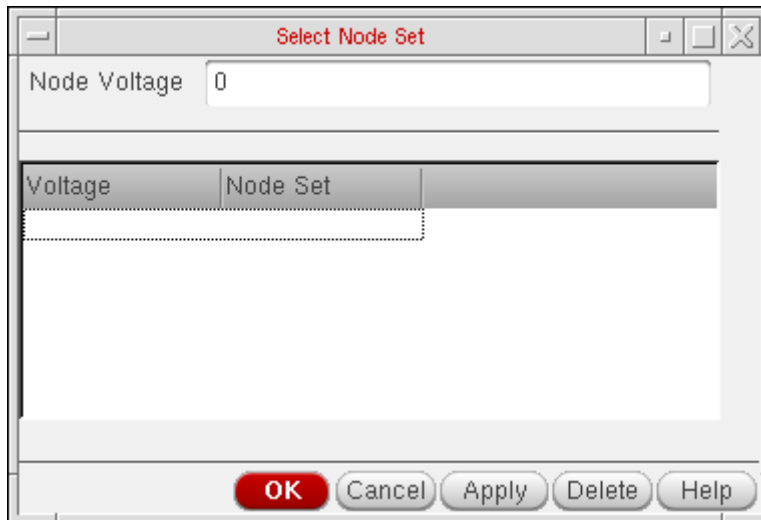
Select Initial Condition Set and the *Select Force Node Set* work in the same way as the Spectre Direct interface. The netlist contains the `.IC` and the `.DCVOLT` statement line, whichever the case may be.

Selecting Nodes and Setting their Values

To select a node and set its voltage,

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

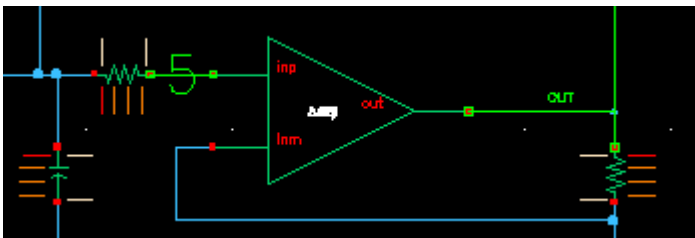
A form appears to enter the voltage. Each command displays a different form. The Select Node Set form is shown here.



2. Type the voltage.
3. Click in the Schematic window to select the first node.

The node name appears in the form.

In the Schematic window, the node is highlighted and the voltage appears. For split nets, the system labels only the driving cell.



4. To set other nodes to the same voltage, select them.
5. To set other nodes to a different voltage, change the voltage, and select other nodes.
6. Click *OK* or *Cancel* when you are finished selecting nodes.

Note: You can also set the node voltage in the Select Node Set form using a VAR(“myVoltage”) variable expression syntax. The variable, myVoltage, will then appear in the Name column of the Design Variables section of the ADE L form. You can provide the value for the variable in the Value column of the Design Variables section. If you provide an expression, the value of the expression is calculated when you create the netlist.

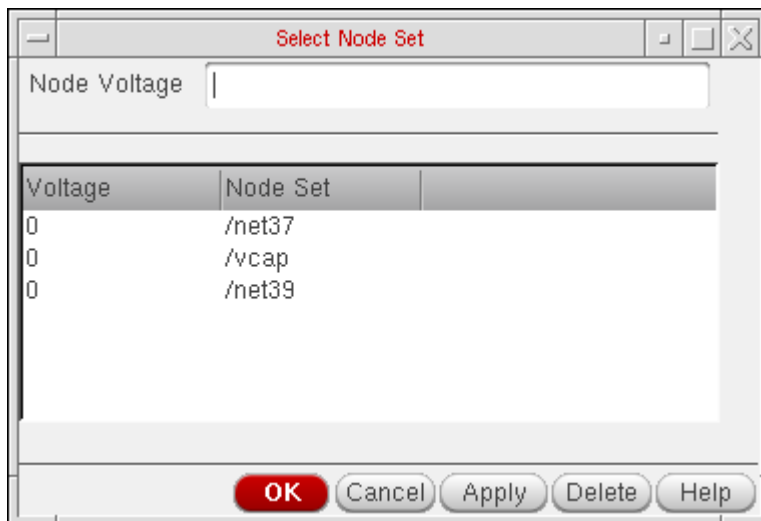
You can select nodes at any level of the hierarchy. When you select an interface node at a lower level, the node is highlighted, but the voltage value appears only on the higher-level schematic.

Releasing Voltages

To release the node set, initial condition, or force node voltage settings,

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

A form appears listing all of the nodes that have been set. The Select Node Set form is shown here.



2. Click on the net in the schematic to release it, or click on the net name in the form and click *Delete*.

On the form, you can select several nodes to release by holding down the left mouse button and dragging through the list box.

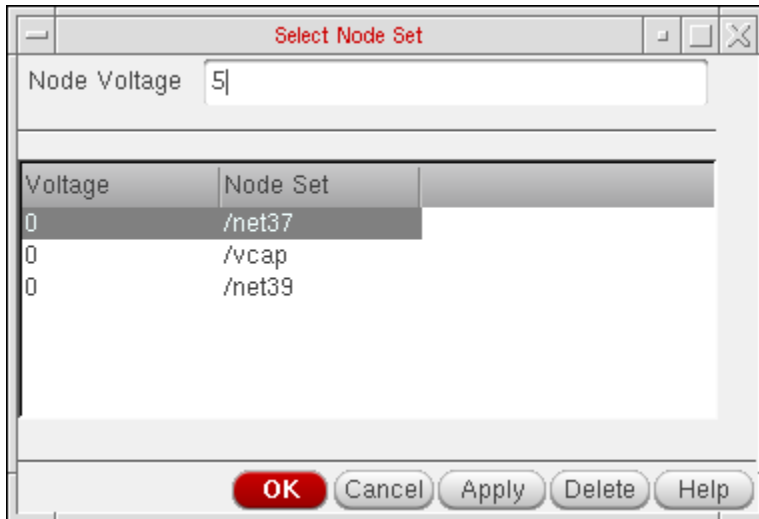
3. Click *OK* or *Cancel* when you are finished releasing nodes.

Changing Voltages

To change the value of the voltage set on a node,

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

A form appears listing all of the nodes that have been set. The Select Node Set form is shown here.



2. Click on the node name to highlight it.
3. Type the new voltage value, and click *Apply*.
4. Click *OK* or *Cancel* when you are finished changing voltages.

Saving and Restoring Node Voltages

To save a list of nodes and their node set, initial condition, or force node voltage settings,

1. In the Simulation window, choose *Session – Save State*.

The Saving State form appears.

2. Type in a name for the saved simulation state.
3. Check that the *Convergence Setup* box is selected, and click *OK*.

To restore the saved settings,

1. Choose *Session – Load State*.

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

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

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

3. Click on a *State Name*.
4. Check that *Convergence Setup* is selected, and click *OK*.

Highlighting Set Nodes

While the *Select* commands are active, the system highlights selected nodes and labels them with the voltages you set. After you close the form, the system removes the highlighting and the labels.

To redisplay the highlighting and labels,

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

To remove the highlighting and labels,

- Close the active *Select* form.

Storing a Solution

To store a solution for the Spectre simulator, use the *write* and *writefinal* fields in the *State File Parameters* section of any analysis options form.

The image shows a dialog box with two sections. The top section is titled "CONVERGENCE PARAMETERS" and contains two input fields: "readns" and "cmin". The bottom section is titled "STATE FILE PARAMETERS" and contains three input fields: "write" (with the value "spectre.ic"), "writefinal" (with the value "spectre.fc"), and "ckptperiod". Each input field has a small button with three dots to its right, indicating a file selection dialog.

This causes the Spectre simulator to write out the initial and final conditions of a transient analysis. By default, the conditions are written to the files `spectre.ic` and `spectre.fc` in the netlist directory.

To store the solution of an analysis, use the *write* and *writefinal* fields of the DC Options form for the analysis.

STATE-FILE PARAMETERS

force none node dev all

readns ...

readforce ...

write ...

writefinal ...

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

Restoring a Solution for Spectre

To restore a saved transient solution, use the *readns* field in the *Convergence Parameters* section of the transient options form. Use the name of the file that was previously used in the *write* or *writefinal* sections of the same form.

CONVERGENCE PARAMETERS

readns ...

cmin

To stop using the solution, clear the field.

Virtuoso Analog Design Environment L User Guide

Helping a Simulation to Converge

To restore a saved DC solution, use the *readns* field in the *State-File Parameters* section of the DC Options form. Use the name of the file that was previously used in the *write* or *writefinal* sections of the same form.

STATE-FILE PARAMETERS

force none node dev all

readns ...

readforce ...

write ...

writefinal ...

To stop using the solution, clear the field.

Form Field Descriptions

Store/Restore File

store stores the DC analysis node voltages in the file specified in the *File Name* field.

restore restores the DC analysis node voltages from the file specified in the *File Name* field.

off turns off the *store* and *restore* buttons.

File Name is the name of the file into which the node voltages are saved. The default filename is `storeRestoreFile`.

Analysis Tools

This chapter describes the advanced analysis tools available in the Virtuoso® Analog Design Environment.

- [Parametric Analysis](#) on page 451
- [UltraSim Power Network Solver](#) on page 470
- [UltraSim Interactive Simulation Debugging](#) on page 474

Parametric Analysis

The Parametric Analysis tool is useful during the design and verification phases of a circuit. It enables you to analyze circuit behavior for a set of circuit parameter values.

Parametric Analysis can be used to sweep the design variables defined in the state over a range/set of values. The results can be plotted in the Virtuoso Visualization and Analysis XL graph window to see the effect of systematically altering circuit values.

Using the Parametric Analysis Tool

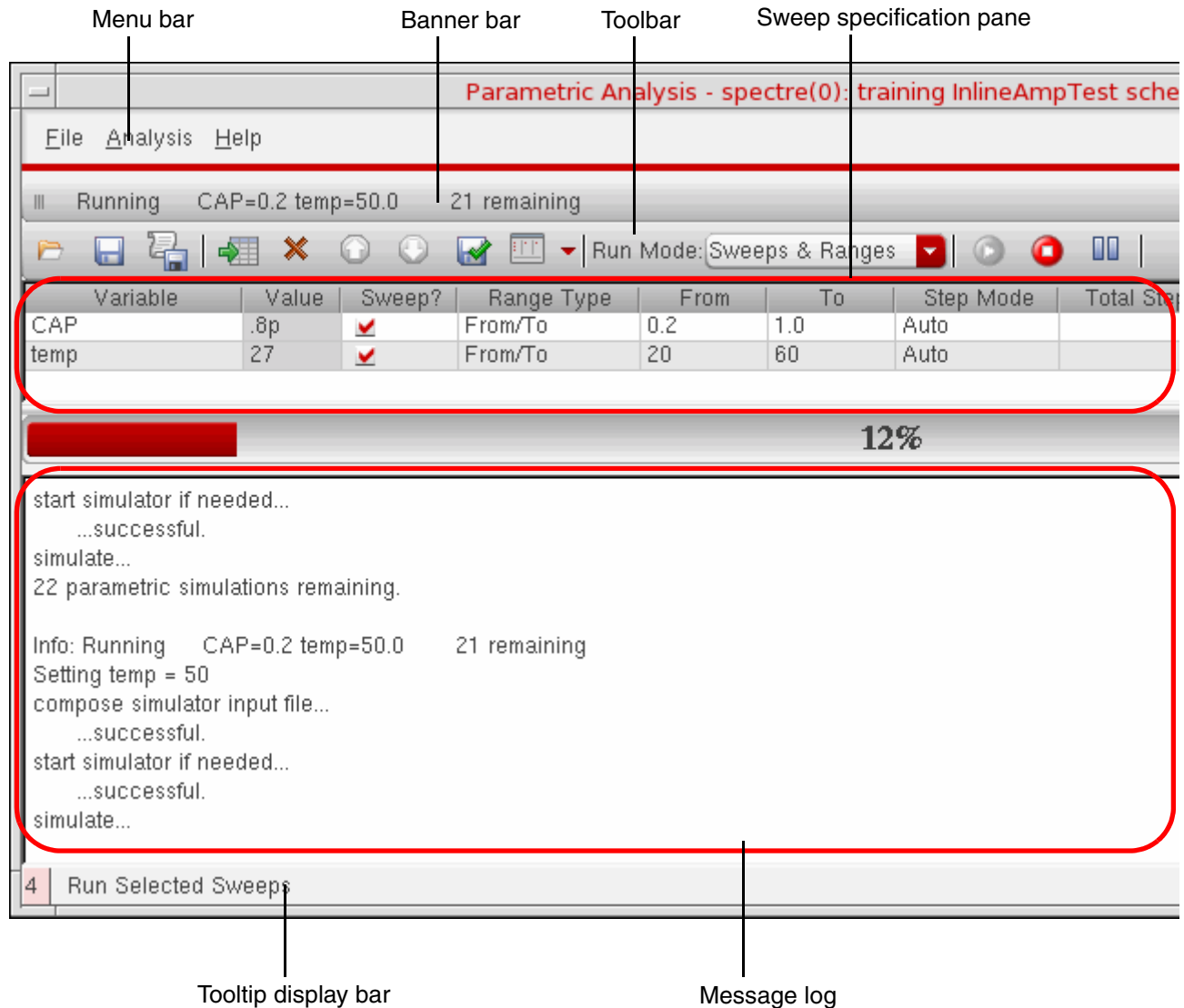
The following is an overview of the steps required to perform a parametric analysis. Click on any highlighted area to go to more information about a particular step or feature.

- In most cases, you perform a simulation of a circuit before you use parametric analysis. This helps in ensuring that the simulation runs successfully before the parametric analysis runs multiple simulations. (Optional)
- Launch the [Parametric Analysis window](#) from ADE L environment.
- In the Parametric Analysis window, specify the [sweep variables](#) with their sweep ranges and step values for a parametric analysis.
- [Run](#) parametric analysis. You can also stop, pause, and restart a parametric analysis.

Parametric Analysis Window

To open the Parametric Analysis window, choose *Tools – Parametric Analysis* in the Virtuoso® Analog Design Environment window.

The Parametric Analysis window appears.



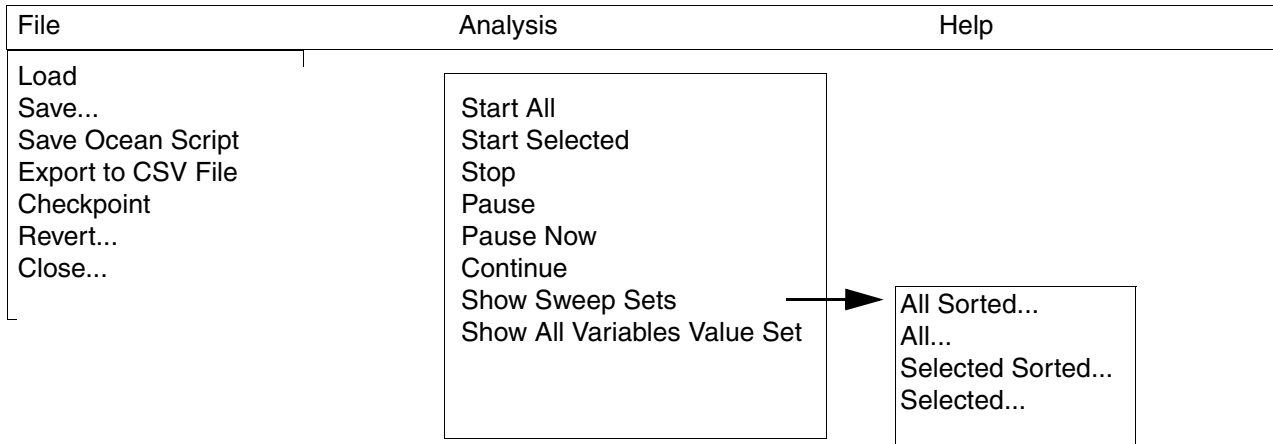
Starting IC 6.1.4, the Parametric Analysis window has been revamped to make it more user friendly.

Virtuoso Analog Design Environment L User Guide









Analysis Tools






The main components of the Parametric Analysis window are:

- **Menu bar:** The Parametric Analysis window has the following menu options.



- **Banner bar:** Shows the name of the last successfully completed file- or parametric analysis run-related tasks. During runtime also, this bar shows the status of intermediate steps.
- **Toolbar:** Contains a set of frequently used commands. The toolbar has the following icons:

Toolbar Command	Used to...
	Load a parametric file
	Save the sweep specification to a parametric file
	Create an ocean script for parametric analysis
	Add new row in the <u>Sweep specification pane</u>
	Delete selected rows from the <u>Sweep specification pane</u>
	Move the selected row up in the table
	Move the selected row down in the table
	Export sweep specification details to a file in csv format

Toolbar Command	Used to...
	Configure the <u>Sweep specification pane</u> . The pull-down list shows the names of columns that you can show or hide. In addition, it provides the <code>Swept Only Rows</code> option to show only those rows for that you have chosen to sweep.
	Select a run mode, <u>Sweeps & Ranges</u> or <u>Parametric Set</u> .
	Start the parametric analysis run for selected rows
	Stop the parametric analysis run without completing it
	Pause/Continue the parametric analysis run

- Sweep specification pane: Contains a table where you can provide sweep variables and other details to be used for parametric analysis. You can insert rows to declare multiple variables. For more details on how to define variables, refer to Specifying Variables for Sweeps & Ranges Run Mode on page 455.
- Message log: Displays details of a parametric analysis run. These details include run status and error messages for invalid sweep specification. Informational messages for various tasks, such as, loading/saving sweep details from/to a file or other tasks that involve generation of sweep data, are also shown in this log area. During the runtime, a Run progress bar also appears on top of this pane.
- Tooltip display bar: Shows tooltips for the toolbar buttons and menu options.

Modes of Parametric Analysis

The Parametric Analysis tool allows you to run analysis in two modes:

- Sweeps & Ranges: In this run mode, you need to specify value ranges and a step mode for one or more design variables. For example, you can specify a range of temperature from -10 degrees to 48 degrees in 10 steps. The tool calculates the step values and runs parametric analysis for all the possible combinations of step values of all the variables. For more details, refer to Specifying Variables for Sweeps & Ranges Run Mode.
- Parametric Set: In this mode, you need to specify a list of values for each sweep variable. Total number of values should be same for all the variables. For a run, the tool picks the values from the same ordinal position of all the lists. Therefore, the number of times for

which the analysis is run is equal to the number of values specified for a variable. For more details, refer to [Specifying Parameters for Parametric Set Run Mode](#).

Note: The `Parametric Set` run mode is applicable only when the `spectre` simulator is used.

Specifying Variables for Sweeps & Ranges Run Mode

You can specify sweep variables in the rows in the sweep specification pane. When you open the Parametric Analysis window for the first time, a new row appears by default.

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Total Steps	Inclusion List	Exclusion
Add Variabl...		<input checked="" type="checkbox"/>	From/To			Auto			

The columns that appear by default are:

- **Variable:** Specifies name of a sweep variable. The column lists all the variables defined in the *Design Variables* section of the ADE window. In addition, a built-in variable, `temp`, is also listed. You can select from the list of available variables.
- **Value:** Shows the default value (also called as the nominal value) for the sweep variable. This value is taken from the *Design Variables* pane of the ADE window.
- **Sweep:** Provides a checkbox to specify if the variable is to be swept in the current parametric analysis run.
- **Range Type:** Specifies the type of range for the given variable. You can choose from three range types: `From/To`, `Centre/Span`, `Centre/Span%`. The names of the next two columns depends on the type of range chosen. To know more about range types and values, refer to [Specifying Range Type and Range Limits](#) on page 458.
- **Step Mode:** Specifies a step mode. Default is `Auto`. To know more details about other step modes, refer to [Specifying Step Mode and Total Steps](#) on page 460.
- **Total Steps/Step Size:** Specifies the total number of steps or the step size for the given step mode. Depending on the step mode selected for a variable, the name of this column changes dynamically to *Total Steps* or *Step Size*.
- **Inclusion List:** Specifies a list of specific points in addition to the range you have specified. You can specify any number of additional points.
- **Exclusion List:** Specifies a list of specific points to be excluded from the range you have specified. You can specify any number of points. If a value is specified in both inclusion and exclusion list, the value is excluded from the sweep sets.

 **Important**


The set of columns that appear in the Sweep specification pane vary depending on the run mode set by you. The default run mode is `Sweeps & Ranges`. Therefore, the columns listed above appear. If you change the run mode to `Parametric Set` only four columns appear: `Variable`, `Value`, `Sweep`, and `Value List`. For more details on `Parametric Set` run mode, refer Specifying Parameters for Parametric Set Run Mode on page 462.

Specifying sweep variables for the `Sweeps and Ranges` run mode involves:

- Adding Sweep Variables
- Changing Sweep Order for Variables
- Specifying Range Type and Range Limits
- Specifying Multiple Ranges for a Sweep Variable
- Specifying Step Mode and Total Steps

Adding Sweep Variables

To add sweep variables, you can insert new rows in the Sweep specification pane in the following ways:



- Choose the *Add New Row* icon, , from the toolbar.
A new row is added after the last row.
- Select any cell in a row, right-click on it and choose the *Insert Row* option from the popup menu.
A new row is created under the selected row.

Select any cell in a row in the Sweep specification pane and double-click in the *Variables* field. The column becomes editable. A pull-down list also appears, listing all the variables defined in the *Design Variables* pane of the ADE window. In addition, a built-in variable, `temp`, also appears as you can also sweep temperature in a parametric analysis.

Select a variable name from the list or type a valid variable name in the given space. The auto complete feature helps you in completing the name.

To know how to delete a row, refer to Deleting Sweep Specifications on page 463.

Changing Sweep Order for Variables

If you have specified multiple variables, they are swept in an order in which they are placed in the Sweep specification pane. You can change the sweep order for a variable by moving its row up or down. To move a row, select any cell on the row and click  or , as required, on the toolbar.

Note that these two buttons on the toolbar are active only when you select a single row in the Sweep specification pane.

Note: Currently, it is not possible to select multiple rows and move them together.

Sweep Order for Sweep & Ranges Run Mode

If you sweep more than one variable when the run mode is set as Sweep & Ranges, the tool creates sweep sets as follows. The first variable (Sweep 1 position) is assigned its first value while subsequent variables (Sweep 2, Sweep 3, ...) cycle through all their values. The first variable then moves to its next value, and the subsequent variables again cycle through all their values, and so on.

For example, you have created three sweep variables (CAP, RES, and temp) and you have set the sweep specifications as:

- Sweep values .02 and .08 for CAP
- Sweep values 10 and 20 for RES
- Sweep values 25 and 45 for temp

With these specifications, parametric analysis performs eight runs as shown below:

	CAP	RES	temp
Run 1	200m	10K	25
Run 2	200m	10K	45
Run 3	200m	20K	25
Run 4	200m	20K	45
Run 5	800m	10K	25
Run 6	800m	10K	45
Run 7	800m	20K	25
Run 8	800m	20K	45

Sweep Order for Parametric Set Run Mode

If the run mode is set as Parametric Set, the tool creates sweep sets as follows. The tool picks the first parameter value from each list for the first iteration, the second value from each list for the second iteration and so on. For example, you need to sweep two variables, CAP and temp, and you have set the sweep specifications as:

- List of values for CAP as .01, .02, .04
- List of values for temp as 40, 50, 60

With these specifications, parametric analysis performs three runs as shown below:

	CAP	temp
Run 1	100m	40
Run 2	200m	50
Run 3	400m	60

Specifying Range Type and Range Limits

To specify range limits for a sweep variable,

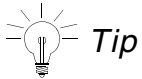
1. Select the row for the sweep variable.
2. Choose a range type from the *Range Type* column. You can choose from the following range types:
 - From/To*: Lets you specify the limits of the sweep range with numerical values. If you do not specify any value in the *From* column, the default value 0 is taken. If you do not specify any value in the *To* column, only the start value (specified in the *From* column) is considered and steps are not calculated. If you do not specify any value in both the *From* and *To* columns, simulation is not done and an error is flagged.
 - Center/Span*: Lets you specify a center point and the range of values around the center you want to sweep. For example, *center* = 100, *span* = 20 is equivalent to *from* = 90, *to* = 110.
 - Center/Span%*: Lets you specify a center point and a range around the center. With this option, you can specify range limits as a percentage of the center value. For example, *center* = 100, *span%* = 40 is equivalent to *from* = 80, *to* = 120.

If you do not specify any value in the *Center* column, the default value 0 is taken. If you do not specify any value in the *Span* or *Span%* column, only the start value (specified in the *Center* column) is considered and steps are not calculated. If you

do not specify any value in both the *Center* and *Span* or *Span%* columns, simulation is not done and an error is flagged.

Note: Depending on the range type you choose, the names of the two columns to the right of the *Range Type* column change when you move out of the cell using the `Tab` or `Shift+Tab` keys. For *Center/Span* and *Center/Span%* range types, the column names are *Center* and *Span%*.

3. Type in appropriate range limits in the two columns on the right of *Range Type* column.
4. Select a step mode in the *Step Mode* column and specify total number of steps in the *Total Steps* column. For more details, refer to [Specifying Step Mode and Total Steps](#) on page 460. If you do not specify the total number of steps, the default step value for the selected step mode is considered during points generation.
5. Specify the values to be included to or excluded from the specified range, if any, in the *Inclusion List* and *Exclusion List* columns, respectively.



Tip

You can use the `Tab` or `Arrow` keys to navigate across the cells in the Sweep specification pane.

Specifying Multiple Ranges for a Sweep Variable

You can specify multiple ranges for a sweep variable by adding multiple rows for the same variable. In each row, you can select the same variable and specify an additional sweep range for it. Alternatively, you can also duplicate an existing row.

Duplicating a Sweep Variable

Select any cell in the existing row, right-click on it and choose the *Add Duplicate* option from the popup menu.

A new duplicate row is created under the selected row. The name of variable in the new row is same as that of its previous row. You can fill in the range.



Important

If you have multiple rows for a variable, the sweep sequence for all the rows of that variable remains same, irrespective of where they are placed in the Sweep specification pane. For example, you have specified two different ranges for a

variable `CAP` in the first and fourth row. Being declared in the first row, `CAP` becomes the Sweep 1 variable. Therefore, parametric analysis will run for both the rows for `CAP` before any other variable.

Specifying Step Mode and Total Steps

In addition to the value range, you also need to specify step values. The step type determines the interval between step values. You can select from the following options:

- **Auto:** Sweeps five steps between the start and stop values you specify. If the ratio between the start and stop values is greater than 1:50, the system uses logarithmic steps and sweeps powers of 10 with equidistant exponents. Otherwise, the sweep steps are equidistant and linear. With this option, you do not have to enter data in the *Total Steps* field.

For example, if you enter a start value of 1 and a stop value of 50, a start:stop ratio of 1:50, the parametric analyzer sweeps the following linear step values:

1	13.25	25.5	37.75	50
---	-------	------	-------	----

If you enter a start value of 2 and a stop value of 200, a start:stop ratio greater than 1:50, the parametric analyzer uses the following logarithmic step values.

2	6.32456	20	63.2456	200
---	---------	----	---------	-----

These steps, with exponents rounded to two decimal places, are 10^{-30} , 10^{-73} , $10^{1.15}$, $10^{1.58}$, and $10^{2.00}$.

- **Linear Steps:** Sweeps a number of equidistant steps determined by the size of the step you specify. The default step size is 1.

For example, if you enter a start value of 1.0, a stop value of 2.0, and a *Step Size* value of 0.2, the parametric analyzer simulates at the following values:

1.0	1.2	1.4	1.6	1.8	2.0
-----	-----	-----	-----	-----	-----

- **Linear:** Simulates the number of steps you specify and automatically assigns equal intervals between the steps. With this option, there is always a simulation at both the start and stop values. The number of steps must be an integer value greater than 0. The default number of steps is 5.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

For example, if you enter a start value of 0.5, a stop value of 2.0, and a *Total Steps* value of 4, the parametric analyzer simulates at the following values:

0.5 1.0 1.5 2.0

- **Decade:** Assigns the number of steps you specify between the starting and stopping points using the following formula:

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

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps per decade is 5.

For example, if you specify a start value of 10, a stop value of 1000, and a *Steps/Decade* value of 3, the parametric analyzer simulates at the following values:

10 21.5443 46.4159 100 215.443 464.159 1000

- **Octave:** Assigns the number of steps you specify between the starting and stopping points using the following formula:

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

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps per octave is 5.

For example, if you specify a start value of 2, a stop value of 4, and a *Steps/Octave* value of 5, the parametric analyzer simulates at the following values:

2 2.2974 2.63902 3.03143 3.4822 4

These values are 2^1 , $2^{1.2}$, $2^{1.4}$, $2^{1.6}$, $2^{1.8}$, and 2^2 .

- **Logarithmic:** Assigns the number of steps you specify between the starting and stopping points at equal-ratio intervals using the following formula:

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

The number of steps can be any positive number, such as 0.5, 2, or 6.25. The default number of steps is 5.

For example, if you use a start value of 3, a stop value of 15, and a *Total Steps* value of 5, the parametric analyzer simulates at the following values:

3 4.48605 6.7082 10.0311 15

The ratios of consecutive values are equal, as shown below.

$$3/4.48605 = 4.48605/6.7082 = 6.7082/10.0311 = 10.0311/15 = .67$$

- **Times:** Simulates at points between the start and stop values that are consecutive multiples of the value you enter in the *Multiplier* field. The default multiplier is 2.

For example, if you enter a start value of 1, a stop value of 1000, and a *Multiplier* value of 2, the parametric analyzer simulates at the following values:

1 2 4 8 16 32 64 128 256 512

 *Important*

The Parametric Analysis tool checks the sweep specification data for correctness only while generating points. Points are generated while saving the sweep specification details in ocean script or csv format, while showing the sweep sets, or while running parametric analysis. If the details in these fields are incorrect, appropriate error messages are flagged in the Message log area. If default values are used for any blank cell, appropriate informational messages are flagged.

Specifying Parameters for Parametric Set Run Mode

The Parametric Set run mode allows you to sweep parameter groups. You need to specify multiple values for the parameters. The tool then picks values from the same ordinal position in the value sets of all the parameters. For example, the first parameter value from each list for the first iteration, the second value of each list for the second iteration, and so on.

For example, if you specify two variables, CAP and RES, and value list for them as 800f 1000f 700f 1200f and 1K 5K 2K 4, respectively. The resulting parametric set to be used for simulation will be generated as:

800f 1K

1000f 5K

700f 2K

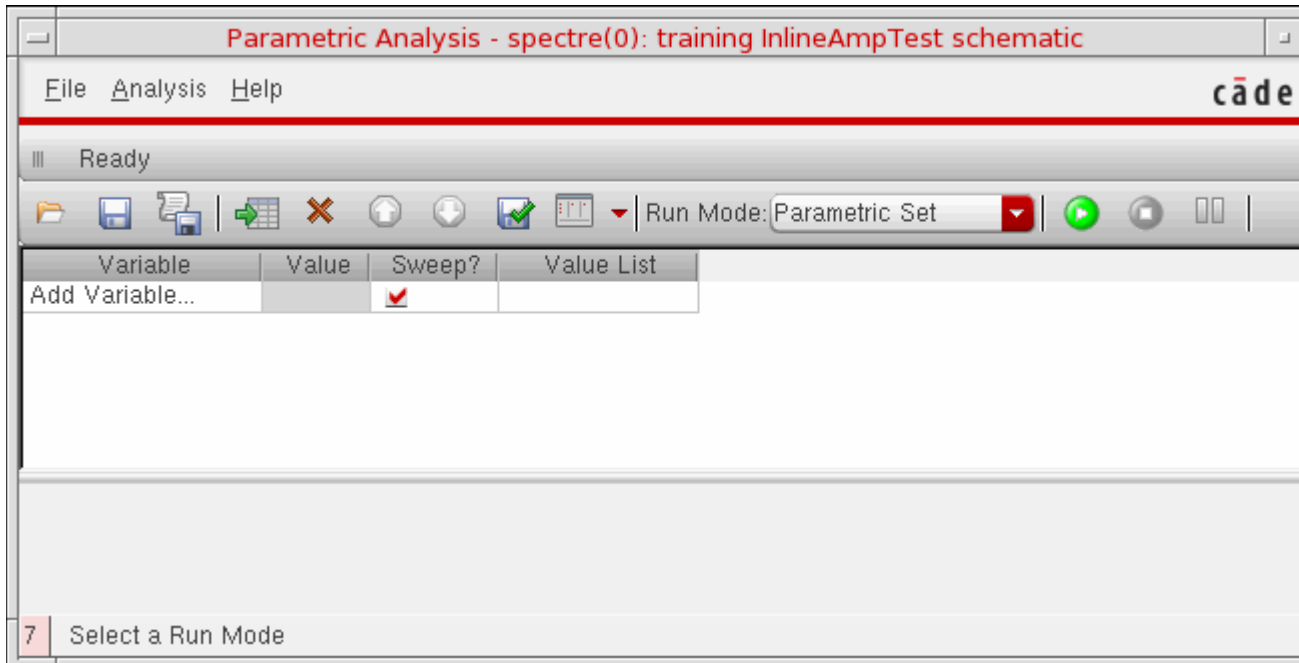
1200f 4K

Note: Please see `spectre -h paramset` for more information.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

To specify parameters and their values for the Parametric Set run mode, select **Parametric Set** from the *Run mode* list on the toolbar. The columns related to this run mode are displayed in the sweep specification pane.



Add rows in the sweep specification pane and specify parameters/variables to be used for the sweep. Also specify the list of comma or space separated values for the variables.


Important


Please note that the number of values should be same for all the parameters/variables. This is a mandatory requirement for *Parametric Set*.


Note: By default, *Run mode* is set as **Sweeps & Ranges**. You can switch between the two values without losing contents of the form.


Deleting Sweep Specifications

You can delete a row from the Sweep specification pane in any of the following ways:

- Select any cell in the row to be deleted and click  on the toolbar
- Select the first cell of the row to be deleted and choose *Clear* from the right-click menu

To delete multiple rows, select the rows while pressing the `Ctrl` or `Shift` key and click  on the toolbar.

To delete all the rows, select the rows either by using the mouse or by pressing the `Ctrl+A` keys and click .

 *Important*

It is not possible to restore a deleted row. However, if you have created a checkpoint, you can revert to that stage. Please note that in this case, you might revert other changes as well (all the changes done after the checkpoint was created).

Reviewing Sweep Specifications

Before parametric analysis is run, the tool generates a sweep set of values for the sweep variables. You can review the sweep set that will be generated by using the sweep specification details. If required, you can modify these details.

The Parametric Analysis tool allows you to view the sweep set using the *Analysis – Show Sweep Set* menu command options. You can view the value sets in the following formats:

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

- ➔ Choose *Analysis – Show Sweep Sets– All...*

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

- ➔ Choose *Analysis – Show Sweep Sets – All Sorted...*

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

- ➔ Choose *Analysis – Show Sweep Sets – Selected...*

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

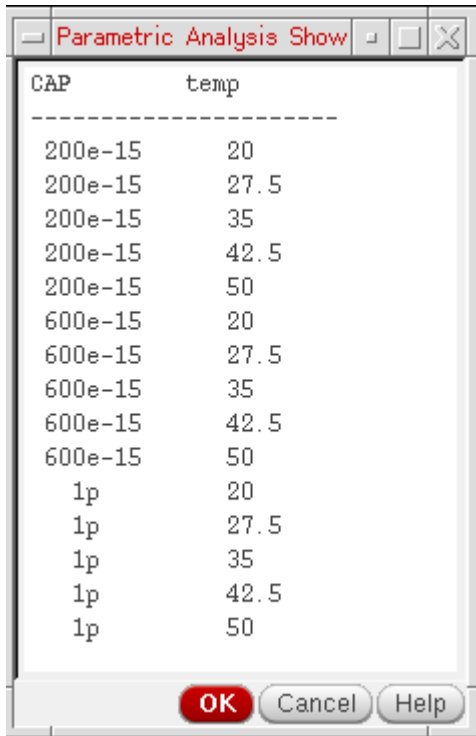
- ➔ Choose *Analysis – Show Sweep Sets – Selected Sorted...*

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

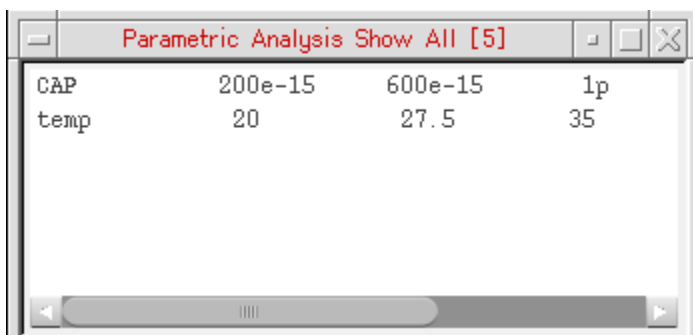
For example, for the following sweep specification:

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Total Steps	Inclusion List
CAP	0.8p	✔	From/To	0.2p	1.0p	Auto		
temp	27	✔	From/To	20	50	Auto		

if the show all sweep sets, the sweep set is generated as shown below:



In addition to the sweep set, the tool also allows you to view all variable value sets using the *Analysis – Show* menu command option. For example, for the above variable data, the tool generates the following value sets:



Running the Parametric Analysis


To start a Parametric Analysis run with all the sweep variables,

- Choose *Analysis –> Start All* in the Parametric Analysis window.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

To start a Parametric Analysis run with only the selected variables, check the Sweep checkbox for all variables to be swept and

- Choose *Analysis – Start Selected* in the Parametric Analysis window. Alternatively, you can also click on the  button on the toolbar.

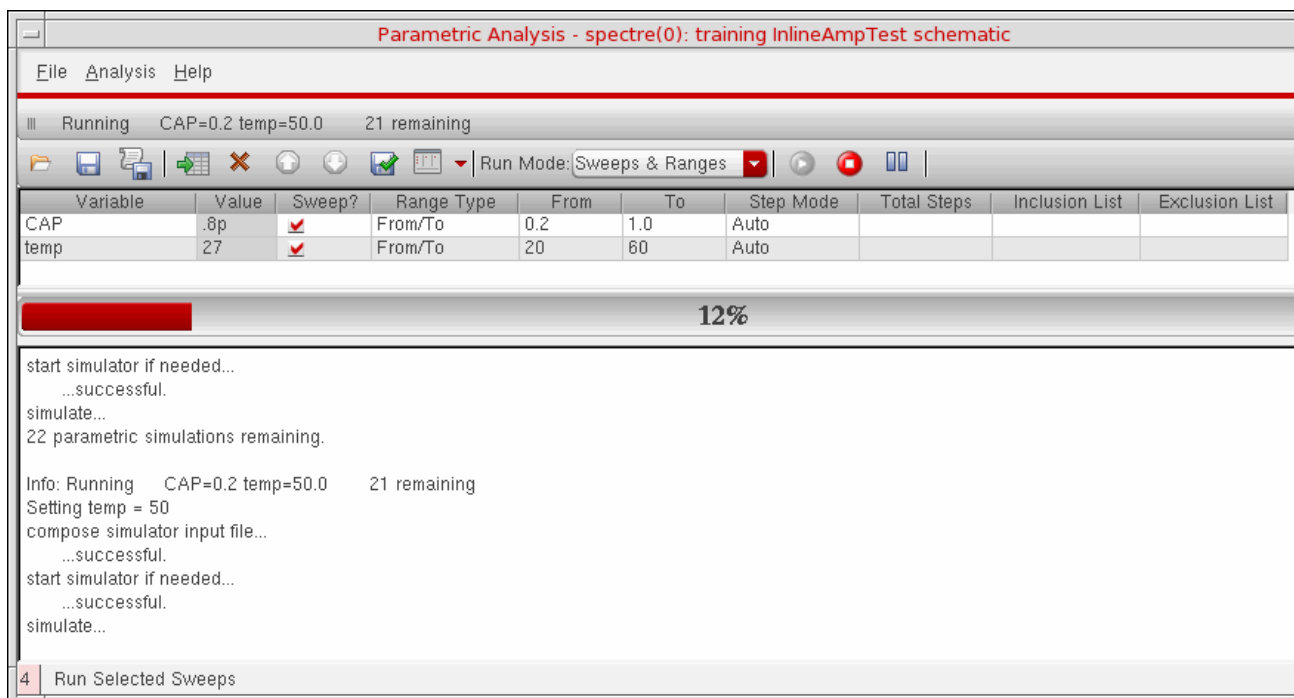
The Parametric Analysis tool generates sweep sets and starts simulation.

Important

Before starting simulation, the Parametric Analysis tool checks if the number of sweep sets is greater than the value of `apaSetPointLimit` variable. If the number of runs is greater, appropriate message is flagged. You can choose to continue or to modify the sweep specifications, if required.

During simulation run, the Parametric Analysis tool shows the following details:

- All informational messages showing status of simulation run are displayed in the Message log and the banner bar.
- All warning and error messages related to simulation run, if any, are displayed in the Message log.
- Resulting waveform is shown in the graph window.
- A progress bar also appears on top of the message log to show the run progress status.



The screenshot shows the Parametric Analysis window for a schematic named 'training InlineAmpTest'. The window title is 'Parametric Analysis - spectre(0): training InlineAmpTest schematic'. The menu bar includes 'File', 'Analysis', and 'Help'. The status bar indicates 'Running CAP=0.2 temp=50.0 21 remaining'. The toolbar shows various icons and a 'Run Mode: Sweeps & Ranges' dropdown. Below the toolbar is a table with the following data:

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Total Steps	Inclusion List	Exclusion List
CAP	.8p	<input checked="" type="checkbox"/>	From/To	0.2	1.0	Auto			
temp	27	<input checked="" type="checkbox"/>	From/To	20	60	Auto			

Below the table is a progress bar showing 12% completion. The message log at the bottom contains the following text:

```
start simulator if needed...
...successful.
simulate...
22 parametric simulations remaining.

Info: Running CAP=0.2 temp=50.0 21 remaining
Setting temp = 50
compose simulator input file...
...successful.
start simulator if needed...
...successful.
simulate...
```

The status bar at the bottom of the window shows '4 Run Selected Sweeps'.

 **Important**


If an error occurs during netlisting or during the first iteration run, simulation is stopped and appropriate errors are flagged. In case of a generic error, this helps in avoiding repetition of the same error message for all the iterations.

 **Tip**

You can view the simulation output log after the simulation run by using the *Simulation -> Output Log* menu option in the ADE window.

Pause and Continue

To pause the simulation after completion of the currently running analysis,

- ▶ Choose *Analysis -> Pause* in the Parametric Analysis window or click on the  button on the toolbar.

To specify an immediate interrupt of an analysis,


- ▶ Choose *Analysis -> Pause Now* in the Parametric Analysis window.

 **Caution**

Pause Now interrupts any currently running simulation and it might invalidate the results of that simulation.


Typically, you use *Pause Now* to interrupt a long simulation when you decide that continuing the parametric analysis is not productive.

To restart a paused analysis,

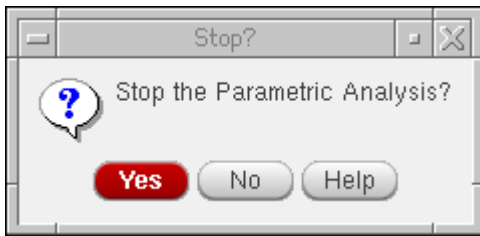
- ▶ Choose *Analysis -> Continue* in the Parametric Analysis window or click on the  button on the toolbar.

Stopping the Parametric Analysis

To stop the simulation in between a run,

- ▶ Choose *Analysis -> Stop* in the Parametric Analysis window or click on the  button on the toolbar.

A pop-up window appears, as shown below:



Note: Till the time you click *Yes* or *No*, the simulation is paused. This is only applicable for Sweeps & Ranges run mode.

Saving and Retrieving Specification Details

The Parametric Analysis tool lets you save sweep specifications in two ways:

- You can store sweep specifications temporarily in a buffer. Later on, you can revert to the last saved state. The specifications saved in the buffer are lost when you close the Parametric Analysis window.
- You can store sweep specifications permanently in a file. When required, you can load the specifications at any time.

Temporary Storage

To store sweep specifications temporarily in a buffer,

- Choose *File* → *Checkpoint* in the Parametric Analysis window.

The sweep specifications are stored in a buffer.

To revert to the sweep specifications stored in a buffer, use the following procedure.



The Revert menu command replaces the current sweep specifications with the specifications saved using checkpoints.

1. Choose *File* → *Revert* in the Parametric Analysis window.
2. Click *Yes* in the dialog box.

Permanent Storage

To save sweep settings permanently in a file,

1. Choose *File* → *Save* in the Parametric Analysis window.

The *Save as Parametric File* form appears.

2. Browse to the desired directory path and type in a filename you want. By default, the files are saved with the `.il` extension.
3. Click *Save*.

To load sweep settings from a file, use the following procedure:

1. Choose *File* → *Load* in the Parametric Analysis window.

The *Select Parametric File to Load* form appears.

2. Browse to the desired directory path and type in the name of file you want to load.
3. Click *Open*. The sweep variables are loaded in the Sweep specifications pane.

Saving an OCEAN Script

To save a script for later use in the OCEAN environment,

1. Choose *File* → *Save Ocean Script* in the Parametric Analysis window.

The *Parametric Analysis Save Script* form appears.

2. Choose which specifications should be saved in the script. You can choose from two options:

- all*: Saves a script that runs every simulation specified in the Parametric Analysis window.
- selected*: Saves a script that runs only the simulations for which the *Sweep* checkbox is checked.

The script is saved in the file. For additional information about using the saved script with OCEAN, see the [OCEAN Reference](#).

Note: If the sweep specification in any of the rows is invalid, the data is not exported to the ocean script.

Saving a csv File

You might want to save sweep data in csv format for a quick review of sweep values. This data can be opened from any spreadsheet application, if required.

To save sweep settings permanently in a file,

1. Choose *File* → *Export to csv File* in the Parametric Analysis window.

The *Export data in a csv format* form appears.

2. Browse to a directory path and type in name of the file. By default, the files are saved with the `.CSV` extension.
3. Click *Save*.

The tool exports all the data to a csv file irrespective of the sweep selection. That is, even if the *Sweep* checkbox is cleared in a row, its details will be exported to the csv file.

Note: If the sweep specification in any of the rows is invalid, the data is not exported.

UltraSim Power Network Solver

This section describes how to detect and analyze power networks using the Virtuoso® UltraSim™ power network solver (UPS) in the analog design environment.

To analyze IR drop effects and their influences on circuit behavior, parasitics in the power and ground net of a circuit design need to be extracted and analyzed together with the circuit. Parasitic elements, such as resistors, capacitors, and inductors, build the power network. These elements need to be simulated, so the parasitic effects on circuit behavior can be analyzed.

To use UPS to detect and analyze power networks:

1. Choose *Simulation – Options – Analog*.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

The Simulator Options form appears.

The image shows a screenshot of the "Simulator Options" dialog box. The dialog has a title bar with the text "Simulator Options" and standard window controls. Below the title bar are five tabs: "Main", "Algorithm", "Component", "PostLayout", and "Output". The "Main" tab is selected. The dialog is divided into several sections:

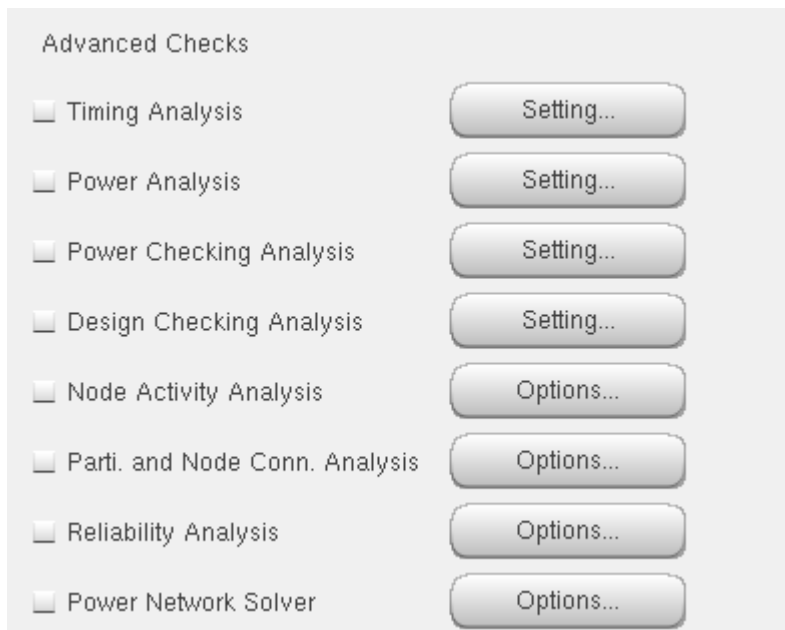
- High Level Options:** Contains four dropdown menus:
 - Simulation Mode: Mixed Signal (MS)
 - Speed Option: Default (5)
 - Analog Option: Default (1)
 - Post-layout Method: No RCR (0)
- Temperature Options:** Contains two text input fields:
 - Temperature (C): 27
 - Tnom (C): 27
- Skip Subckts:** A checkbox is checked, and there is a "Select" button to its right.
- Subckt Instances:** An empty text input field.
- Subckt Names:** An empty text input field.

At the bottom of the dialog are five buttons: "OK" (highlighted in red), "Cancel", "Defaults", "Apply", and "Help".

Virtuoso Analog Design Environment L User Guide

Analysis Tools

2. Click the *Power Network Solver* check box in the *Checks* Tab.



3. Click the *Options* button.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

The Power Network Solver window appears.



4. Adjust the power network solver options as needed.
5. Click *OK*.

For more information, refer to Chapter 5, "Power Network Solver" in the *Virtuoso UltraSim Simulator User Guide*.

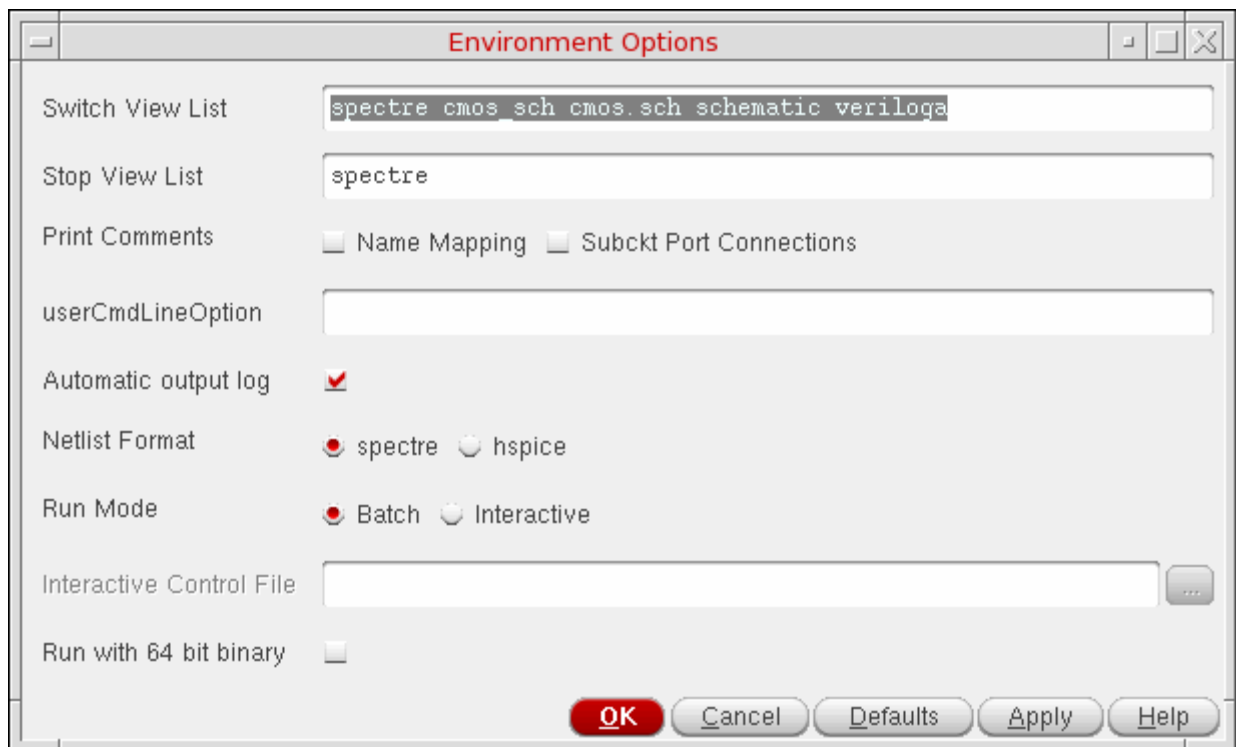
UltraSim Interactive Simulation Debugging

The Virtuoso UltraSim simulator interactive circuit debugging mode allows you to obtain design data, such as circuit elements and parameters, circuit topology, and instantaneous signal values. It can also be used to probe dynamic circuit behavior, including voltage and current waveforms simulated to the current time step.

To use the interactive simulation debugging mode:

1. Choose *Setup – Environment* in the simulation window.

The Environment Options window appears.



2. Choose the *Interactive* radio button.
3. Type in the *Interactive Control File* name.

4. Click *OK*.

For more information about interactive debugging, refer to Chapter 6, "Interactive Simulation Debugging" in the *Virtuoso UltraSim Simulator User Guide*.

Virtuoso Analog Design Environment L User Guide

Analysis Tools

Plotting and Printing

This chapter shows you how to print and plot simulation data.

- [Overview of Plotting](#) on page 477
- [Using the Plot Outputs Commands](#) on page 489
- [Using the Direct Plot Commands](#) on page 491
- [Overview of Printing](#) on page 514
- [Precision Control for Printing](#) on page 533
- [Printing Capacitance Data](#) on page 534
- [Printing Statistical Reports or Calculator Results](#) on page 536
- [Using SKILL to Display Tabular Data](#) on page 537
- [Overview of Plotting Calculator Expressions](#) on page 538
- [Viewing and saving Results](#) on page 545
- [Annotating Simulation Results](#) on page 549
- [Plotting Results of a Parametric Analysis](#) on page 556
- [Form Field Descriptions](#) on page 558t

Overview of Plotting

There are several ways to select simulation results and plot them in the Virtuoso Visualization and Analysis XL graph window:

- From the [Virtuoso Visualization and Analysis XL Calculator](#), use the plot icon to plot waveforms.
- From the [Results Browser](#), click right on a node that contains waveforms.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- From the Simulation window, use the *Outputs – To Be Plotted– Select On Design* command to select nets and terminals in the schematic. Use commands in the *Results – Plot Outputs* menu to display the curves.
- From the Simulation window or the Schematic window, use the *Results – Direct Plot* command to select nets and terminals in the schematic and to plot a function immediately.

Note: When you click on a terminal, it gets selected first and then the wire gets selected. Therefore, you can now alternate between the two.

Before you can plot results, you need to run a simulation or select results. To select results,

1. Choose *Results – Select* in the Simulation window
2. Choose the current data file
3. Click *OK*.

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

If you set up the *Outputs* section in the Virtuoso Analog Design Environment, with nets to be plotted, and click the *Plot Outputs* icon during an analysis run, the graph window will pop up and plot the outputs.

Therefore, you get a snapshot of the simulation run upto that time point. You can also use the *Calculator* or the *Results Browser* to plot outputs.

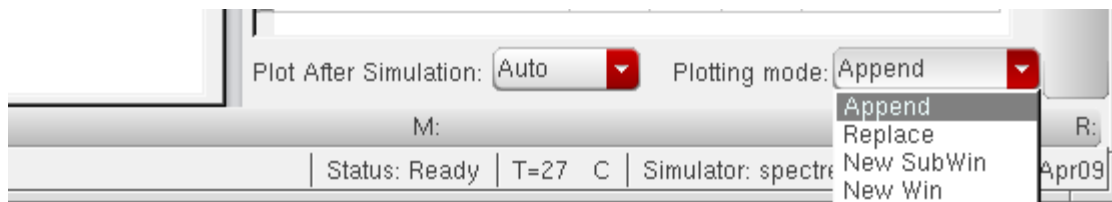
Setting Plotting Mode

The Analog Design Environment provides different plotting modes. You can choose a plotting mode depending on your requirements to save and compare plots of different simulation results.

The *Outputs* section provides the following two drop-down lists to choose how to plot the results:

- Plot After Simulation
- Plotting mode

The two lists in the *Outputs* section are shown in the following figure:



Plot After Simulation

By default in the Analog Design Environment, every output that has been specified for plotting, is plotted at the end of a simulation in the *Virtuoso Visualization and Analysis XL Graph* window. This corresponds to the *Auto* option in the *Plot After Simulation* drop-down list box. In addition, you can choose either to stop automatic plotting or to refresh an earlier plotted graph.

The *Plot After Simulation* list provides the following options:

- *Auto*: Automatically plots output after simulation is run. This is the default option. For every subsequent run, the new graph replaces the existing graph. You can choose to append the new graph to the existing graph of the previous run or to plot it in a new window using the Plotting Mode drop-down list.

When this option is selected, any customization done in the *Virtuoso Visualization and Analysis XL* graph window that are currently open, such as setting up traces, colors, or zoom levels, are not saved. Every time a graph is plotted, the default settings are used.

- *Refresh*: Updates open graphs in Virtuoso Visualization and Analysis with new simulation results and maintains graph and trace settings. If no graph is open, the results are plotted in a new graph window and in subsequent runs, outputs are refreshed in the same graph.

For more information on how the graphs are refreshed, see [Refreshing Graphs](#).

The `Refresh` option has the following limitations:

- It does not consider the modes set in the *Plotting Modes* list.
- *None*: Does not plot output.

Plotting Mode

When the *Plot After Simulation* list is set to *Auto*, you can choose the following plotting modes:

- **Append**: Appends the new graph to the existing graph.
- **Replace**: Replaces the existing graph with the new graph. This is the default option.
- **New SubWin**: Plots the new graph in a new sub window.
- **New Win**: Plots the new graph in a new window.

You can select the plotting modes for automatic plotting from the *Direct Plot*, the [Virtuoso Visualization and Analysis XL Calculator](#) and the [Results Browser](#) windows also.

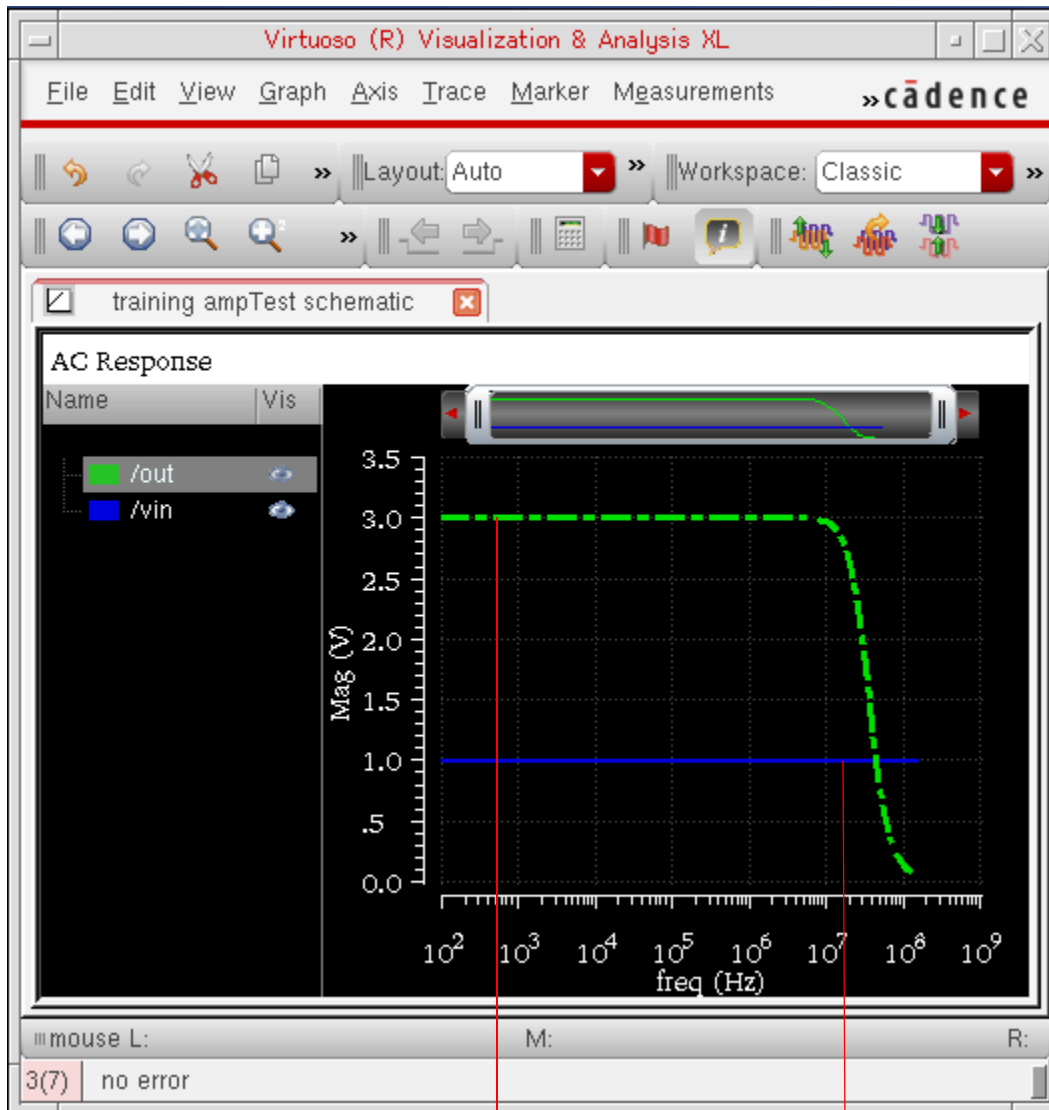
Refreshing Graphs

Graphs can be refreshed using the *Refresh* option in the *Plot After Simulation* drop-down list box. This option updates the open graphs in Virtuoso Visualization and Analysis XL with the new simulation results, and retains the graph and trace settings.

You can use this option to review graphs across different simulation runs. For example, consider you want to run multiple simulations with different values of the design variable CAP. In the first run, plot CAP for 800f. After the simulation results are plotted in the graph, customize the plots, as shown in the figure below.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



Changed plot style
and color

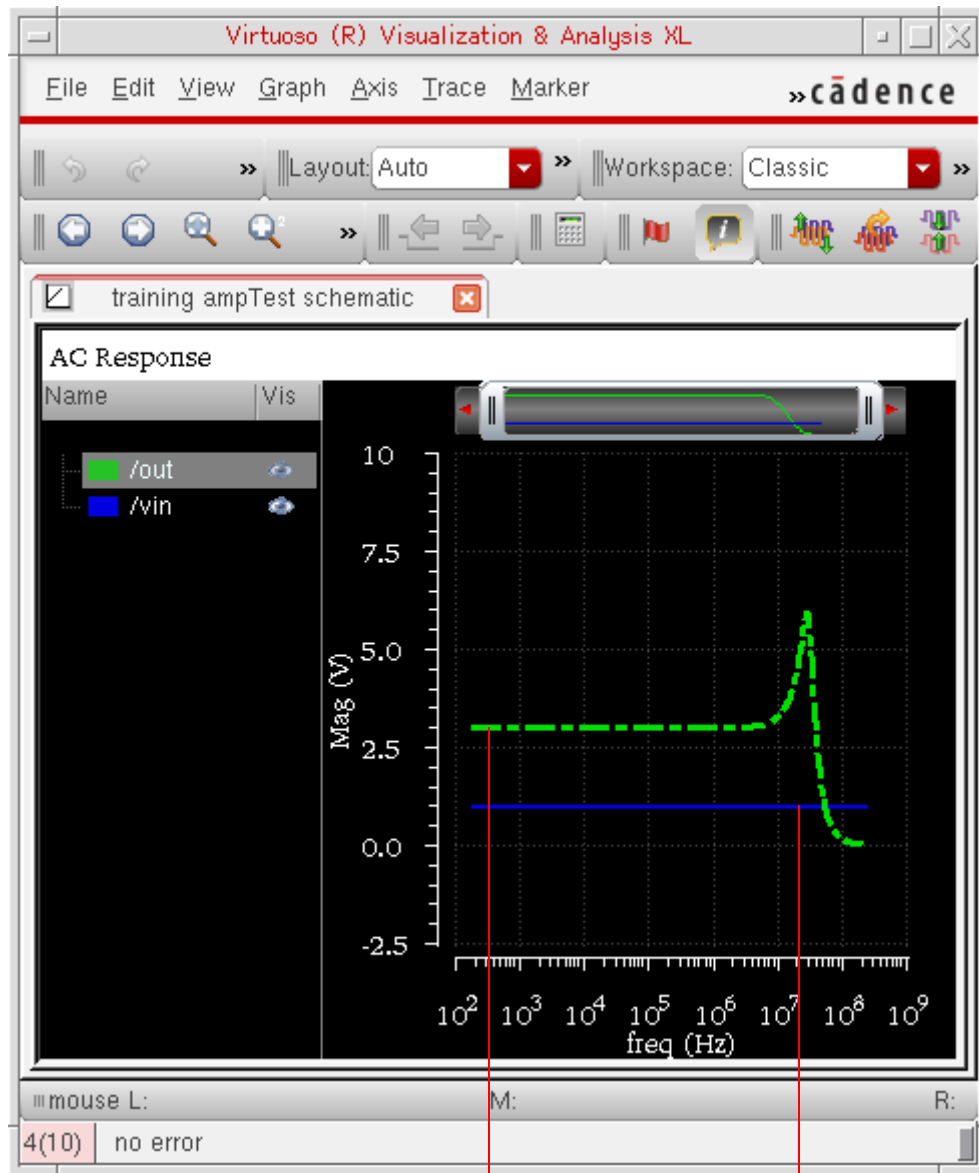
Changed color

In the next run, use $CAP=200f$ to run the simulation. Using the Refresh option, the simulation results are updated in the same graph.

Note: The settings done to the plots are retained, as shown in the following figure.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



Changed plot style
and color saved

Changed color
saved

The following sections describe how the graphs are refreshed using the *Refresh* option in different scenarios:

- [Refresh Plotting with Varying Analysis](#)
- [Refresh Plotting with Varying Outputs](#)

- Refresh Plotting with Parametric Simulation

Graph Settings Supported by the Refresh Plotting Mode

The Virtuoso Visualization and Analysis XL graph window saves and maintains the following settings for the graphs that are generated for a common set of variable combinations across simulation runs:

- Trace color, type, style, width, or symbols
- Visibility status of graphs
- Axes settings
- Pan and zoom settings
- Graph layout
- Strip layout
- Makers and marker locations

Graph Settings Not Supported by the Refresh Plotting Mode

Currently, the following graph settings are not supported:

- Swapping of sweep variable on the X-axis
- Addition of a new graph window
- Any signal, expression or measurement that you directly added to the graph and is not specified in the *Outputs* pane in ADE L window.

Important

To retain additional or derived graphs and their settings, you can send plot from the graph to the ADE L Outputs Setup tab by using the *Send to ADE* command on the shortcut menu of the graph. For more details, refer to [Sending Traces to ADE](#) in *Virtuoso Visualization and Analysis XL User Guide*.

Refresh Plotting with Varying Analysis

The following table describes how graphs are refreshed when analysis are varied across different simulation runs:

Table 10-1 Effect of Analysis Variations on Graph Settings with Refresh Option

Analysis Variation	Effect on Graph Settings
Add or enable an analysis	<ul style="list-style-type: none">■ A new subwindow is added to the graph to display the plots of the newly added or enabled analysis.■ Graphs for the analysis that are common across different runs are updated with new simulation data and any trace settings are retained.
Delete or disable an analysis	<ul style="list-style-type: none">■ Subwindows related to the analysis that has been deleted or disabled after the previous simulation run, are removed from the graph.

Refresh Plotting with Varying Outputs

The following table describes how graphs are refreshed when outputs are varied across different simulation runs:

Table 10-2 Effect of Output Variations on Graph Settings with Refresh Option

Analysis Variation	Effect on Graph Settings
Add or enable an output	<ul style="list-style-type: none">■ Plot for the new output is added in a new subwindow.■ Other existing graphs are refreshed and their trace settings are maintained.
Delete or disable an output	<ul style="list-style-type: none">■ The plot for the deleted or disabled output is removed.■ Other existing graphs are refreshed and their trace settings are maintained.

Note: The *Refresh* option does not update graphs for MATLAB measurements.

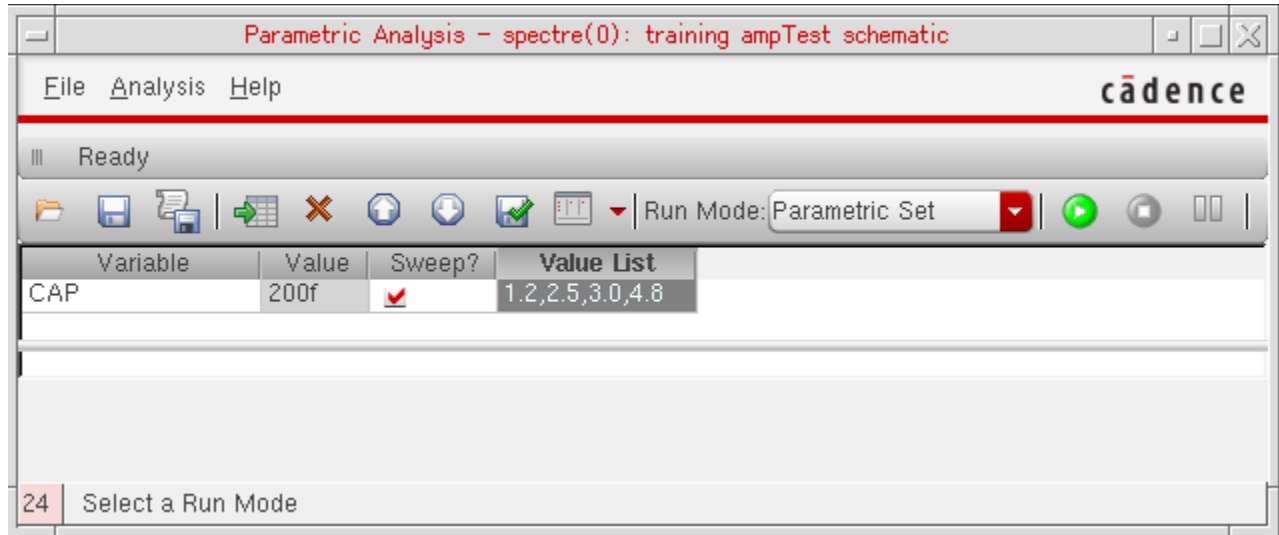
Refresh Plotting with Parametric Simulation

If you perform parametric simulation runs, the graphs are updated with new simulation results for common sweep values between subsequent runs. Traces corresponding to new sweep values are added to the same graph. Traces corresponding to unmatched sweep values are deleted.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

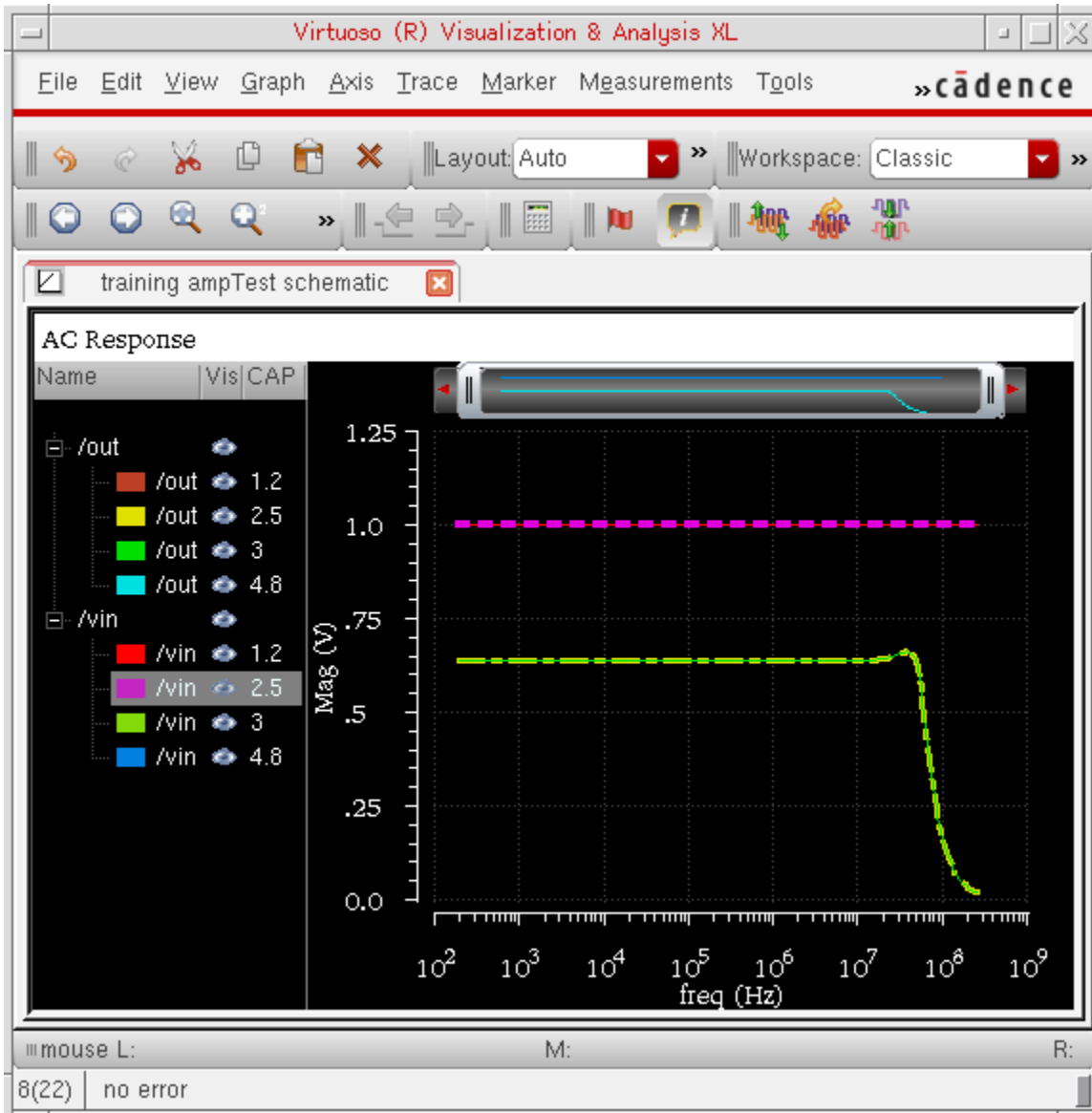
For example, set the variables as shown below.



The graph plotted in this case includes plots for all four sweep values for CAP, as shown below.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



For the next simulation run, change the sweep values for CAP to 2.5:4.8:3.6:5.2. Note the change in the values of CAP for which the simulation is run (mismatch values are underlined), as shown below.

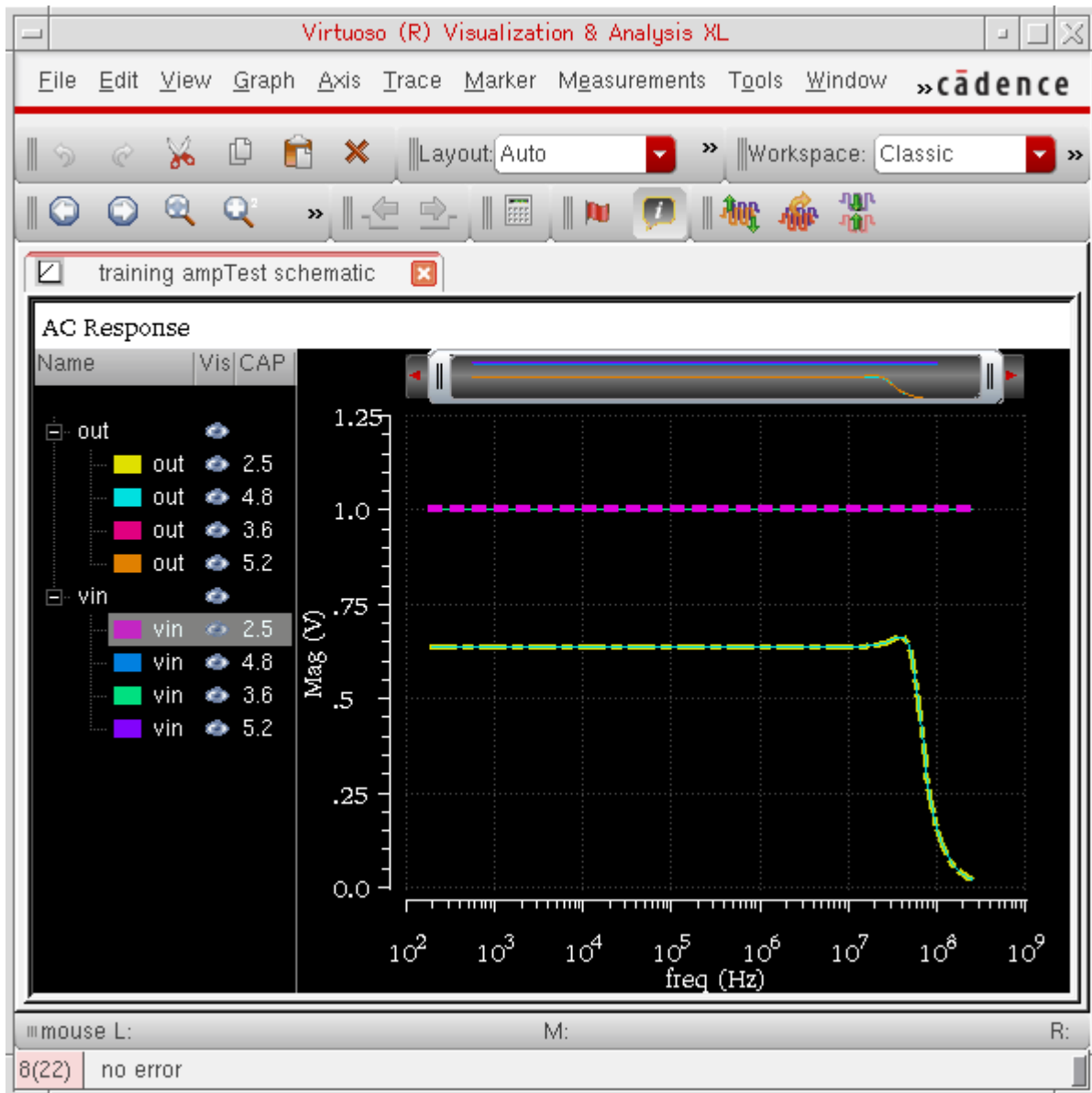
Values of CAP in the first run: 1.2, 2.5, 3.0, and 4.8

Values of CAP in the second run: 2.5, 4.8, 3.6, and 5.2

With the Refresh plotting, the traces corresponding to CAP = 2.5 and 4.8 are updated with new simulation data. Traces corresponding to CAP = 1.2 and 2.0 are removed. Traces corresponding to CAP = 3.6 and 5.2 are added.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



Unsupported Scenarios of Sweep Variations

In the following scenarios, the graphs for common sweep values for the swept variables are not refreshed:

- If you run parametric simulations after you run normal simulations in ADE L window, graphs are not refreshed.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- Change from single sweep variable to multiple sweep variables. For example, in the first simulation run, you sweep variable x , and in the subsequent runs, you sweep variables x and y .
- Change in the sweep variable. For example, in the first run, you sweep variable x and in the subsequent runs, you sweep variable y .

Setting Plotting and Display Options

You set plotting and Virtuoso Visualization and Analysis XL graph window options with the Setting Plotting Options form. For more information about the form, see [“Setting Plotting Options”](#) on page 558.

Option	Value
Print After	Each Selection
Direct Plots Done After	All Selections Are Made
Annotations	Design Name, Simulation Date
Fast Viewing Support	Unchecked

To preserve these settings in future design sessions,

- In the Command Interpreter Window (CIW), choose *Options – Save Defaults*.

To use these settings in only the current design session,

- Click *OK*.

Note: Graph window options you set here apply only to those windows opened by the Simulation window.

Saving and Restoring the Window Setup

You can save and restore a graph window setup with other setup options. For details, refer to the [Virtuoso Visualization and Analysis XL User Guide](#).

1. In the Simulation window, choose *Session – Save State*.

The Saving State form appears.

2. Type in a name for the saved simulation state.
3. Check that the *Waveform Setup* box is selected and click *OK*.

To restore the saved settings,

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

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

2. Select a run directory with the *Cell* and *Simulator* fields.
3. Click a *State Name* and choose *What to Load*.
4. Check that *Waveform Setup* is selected and click *OK*.

Using the Plot Outputs Commands

The five commands in the *Results – Plot Outputs* menu in the Simulation window plot each item in the plot set.

<i>Transient</i>	Plots the transient response for each node
<i>AC</i>	Plots the AC response for each node
<i>DC</i>	Plots the DC sweep response for each node
<i>Noise</i>	Plots the squared noise voltage for each node
<i>Expressions</i>	Plots the waveforms for <u>expressions</u> you define in the Setting Outputs form

Plotting the Current or Restored Results

To plot the most recent (or restored) results in the graph window,

1. In the Simulation window, choose *Outputs – To Be Plotted – Select On Design*.

Nodes and terminals you have already selected are now highlighted.

2. In the Schematic window, select one or more nodes or terminals.
3. Press the `Escape` key when you finish selecting nodes and terminals.
4. Choose a *Results – Plot Output* command in the Simulation window.

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

To plot all of the available results at once,

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

Note: You can plot only saved voltages and currents.

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

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

These forms enable you to select from one to all bits of an iterated item. When you select the top element in the listbox, all the individual bits are selected. You can also select an individual bit with the left mouse button.

Note: `Ctrl-Left` mouse will toggle selection of an item. `Shift-Left` mouse will select all items between the last selected item and the current item.

Removing Nodes and Terminals from the Plot List

To remove a node or terminal from the plotted set,

1. In the Simulation window, click in the *Outputs* list to select the output.

To select more than one output, hold down the `Control` key while you click outputs, or click and drag.

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

2. Choose *Outputs – To Be Plotted – Remove From*.

Plotting Parasitic Simulation Results

When you plot the results of a parasitic simulation, only terminals and device pins can be mapped from the schematic to the extracted view.

To select results in the schematic while parasitic simulation is enabled,

1. From the Simulation window, choose *Outputs – To Be Plotted – Select on Design*.
2. In the schematic, select a terminal or pin, or a wire near a terminal or pin.

If you select a point in the middle of a wire, the system chooses the nearest terminal or device pin and you might not get the right data.

The system draws an \times to mark the point you selected.

3. Choose a *Results – Plot Output* command.

The color of the waveform matches the color of the \times .

Note: You cannot probe nets that connect only sources and loads because these nets do not exist on the extracted view. You also cannot probe nets between parasitic components that were removed by selective annotation because these nets were removed when the selected view was built.

Using the Direct Plot Commands

You can plot common waveforms quickly in the Simulation window using the *Direct Plot* commands. With these commands, you do not need to use the calculator to create common expressions and you do not need to add the nets or terminals to the plot set.

To use Direct Plot,

- Choose *Results- Direct Plot - Main Form*. This brings up the unified *Direct Plot* main form that changes dynamically depending on the analysis that was performed.

or,

- Choose the *Results – Direct Plot* commands. The commands are as follows:

This option...	Plots this curve...
<i>Transient Signal</i>	Transient voltage or current waveforms
<i>Transient Minus DC</i>	Transient voltage or current waveforms without the DC offset
<i>Transient Sum</i>	Multiple signals added together and plotted; you are prompted for the signals
<i>Transient Difference</i>	Two signals subtracted (sig1- sig2) and plotted; you are prompted for two signals

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

This option...	Plots this curve...
<i>AC Magnitude</i>	AC voltage or current gain waveform
<i>AC db10</i>	The magnitude on a decibel scale $10\log(V1)$
<i>AC db20</i>	The magnitude of selected signals on a decibel scale $20\log(V1)$
<i>AC Phase</i>	AC voltage or current phase waveform
<i>AC Magnitude & Phase</i>	The db20 gain and phase of selected signals simultaneously
<i>AC Gain & Phase</i>	The differences between two magnitudes and two phases; you are prompted for two signals $20\log(V2)-20\log(V1)$ which is equivalent to $20\log(V2/V1)$
<i>Equivalent Output Noise</i>	Output noise voltage or current signals selected in the analysis form; the curve plots automatically and does not require selection
<i>Equivalent Input Noise</i>	Input noise waveform, which is the equivalent output noise divided by the gain of the circuit
<i>Squared Output Noise</i>	Squared output noise voltage or current signals selected in the analysis form; the curve plots automatically and does not require selection
<i>Squared Input Noise</i>	Input noise waveform, which is the equivalent output noise divided by the gain of the circuit squared
<u>Noise Figure</u>	Noise figure of selected signals according to the input, output, and source resistance
<i>DC</i>	DC sweep voltage or current waveform

To use *Direct Plot* commands,

1. Choose a command from the *Results – Direct Plot* menu.

If necessary for the command, a form appears.

The graph window opens. If a graph window was already open, it becomes active.

2. Look in the Schematic window for a prompt.

The prompt tells you what to select in the schematic.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

3. Select the signals necessary for the function and press the `Escape` key.

The system plots the signals. The system shuffles windows automatically, so that the graph window is in front.

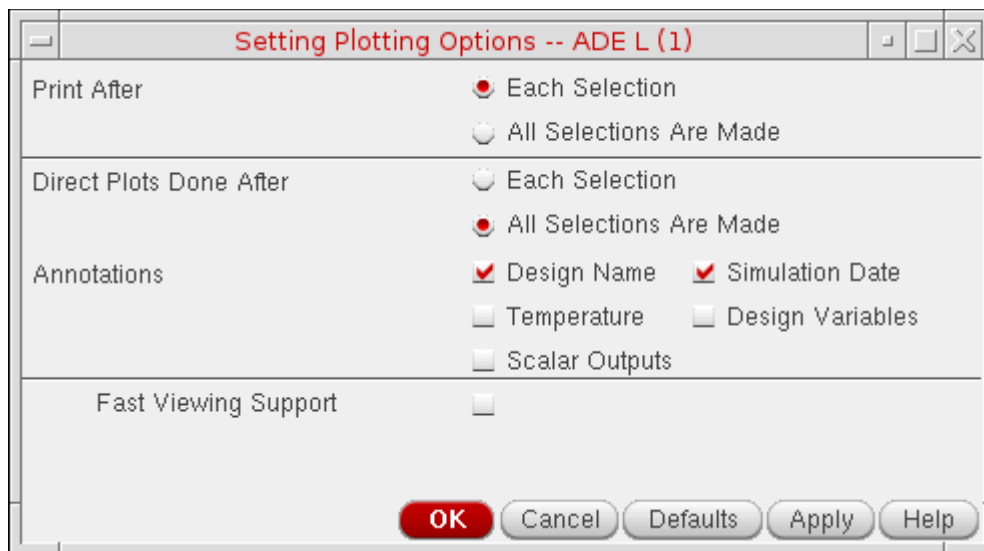
There are two modes for the *Direct Plot* commands:

- Plotting one signal at a time immediately after you select the signal
- Plotting several signals together after you press the `Escape` key

To choose the mode,

1. Choose *Results - Printing/Plotting Options*.

The Setting Plotting Options form appears. For more information about the form, see [Setting Plotting Options](#) on page 558.



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

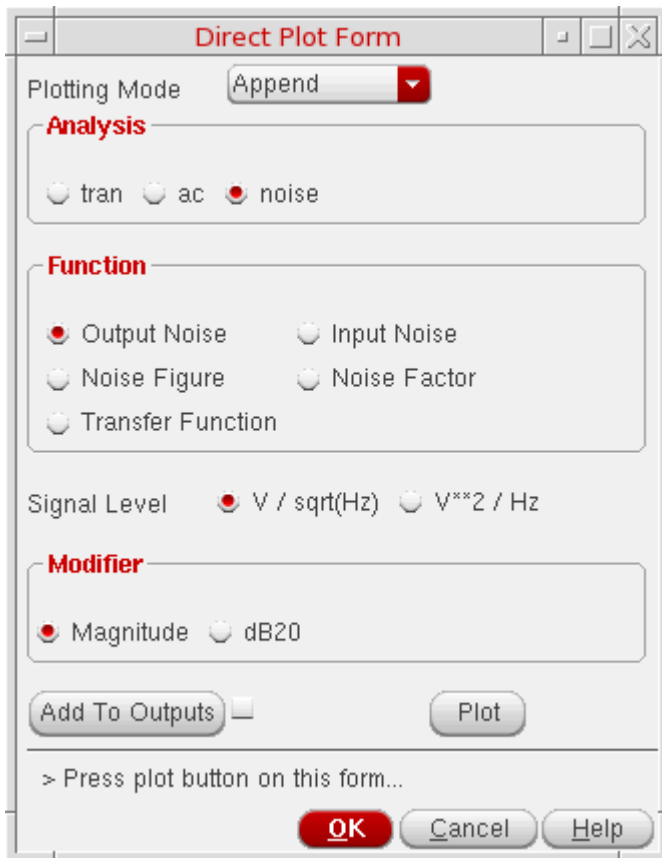
For Noise Figures

Noise figure is calculated by *Spectre* if a port is selected as the input source for noise analysis. If a port is not selected as the input, noise figure data is not available.

To plot the noise figure,

1. Choose *Results – Direct Plot – Noise Figure*.

The *Direct Plot Form* for noise figure appears.

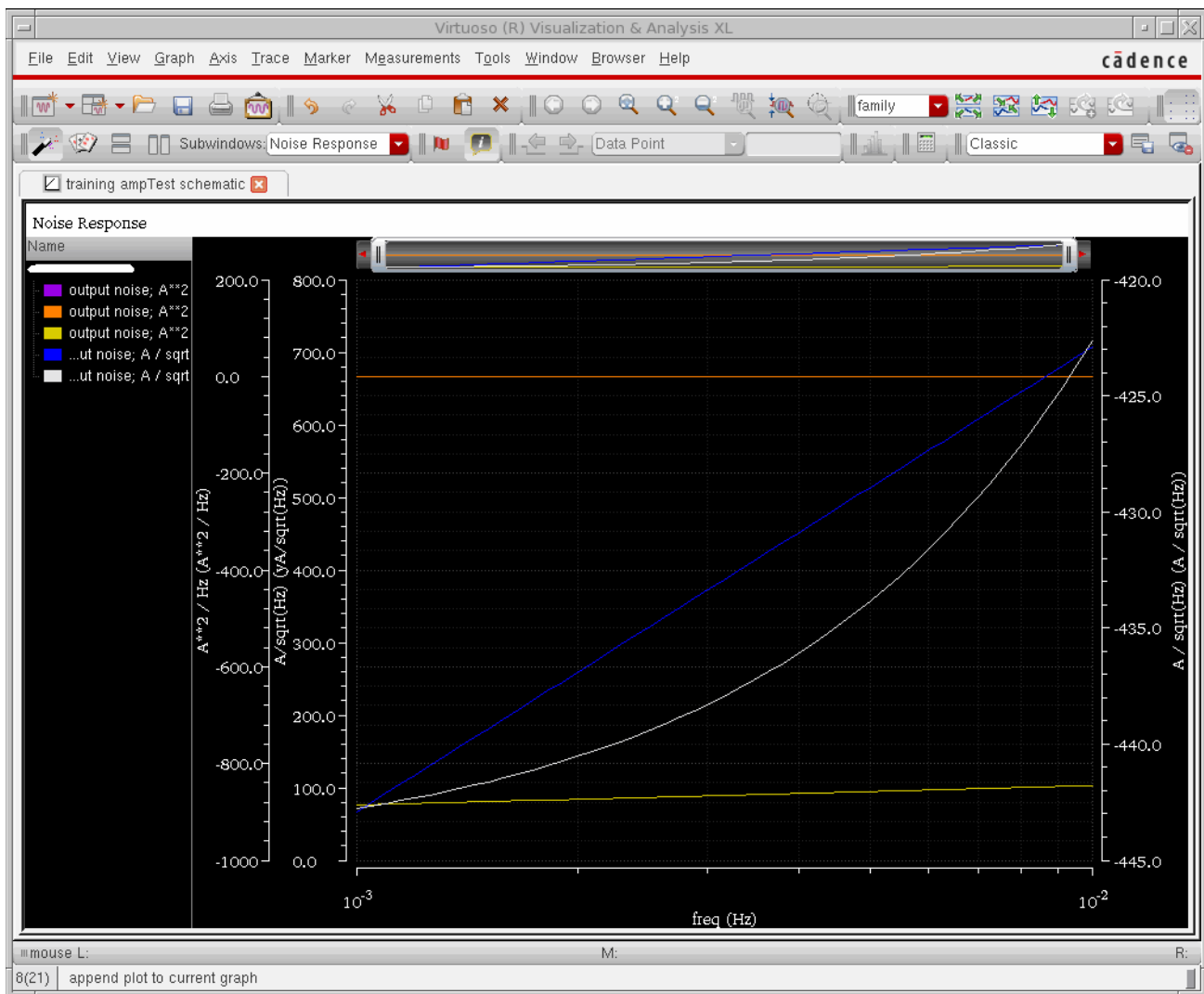


2. Specify the *Plotting Mode*. You can specify:
 - Append* to append the new signal to the current graph. This is the default option.
 - Replace* to replace the current graph with the new signal.
 - New SubWin* to plot the signal in a new subwindow.
 - New Win* to plot the signal in a new window.
3. Select the noise parameter function that you want to plot from the *Function* group box. Based on the selected function and available data, the form changes dynamically to display the applicable options.
4. Select the *Signal Level* that you want to plot. By default, ADE L plots the V_{N2} signal in the *Virtuoso Visualization & Analysis XL* window. You can plot the V_N signal by selecting the *V / sqrt(Hz)* radio button from the *Direct Plot* form.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

5. Choose the appropriate *Modifier* to specify the data or plot format. *Magnitude* radio button (the default setting) plots the magnitude of the selected signal and the *dB20* radio button plots the magnitude in dB.
6. Select *Add To Outputs* check box to add expressions for the results to the outputs section and plot in the mode that you selected.
7. Click *Plot* to view the results in the *Virtuoso Visualization & Analysis XL* window.



8. Click *OK*.

Note: The mathematical noise-figure expression is where:

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

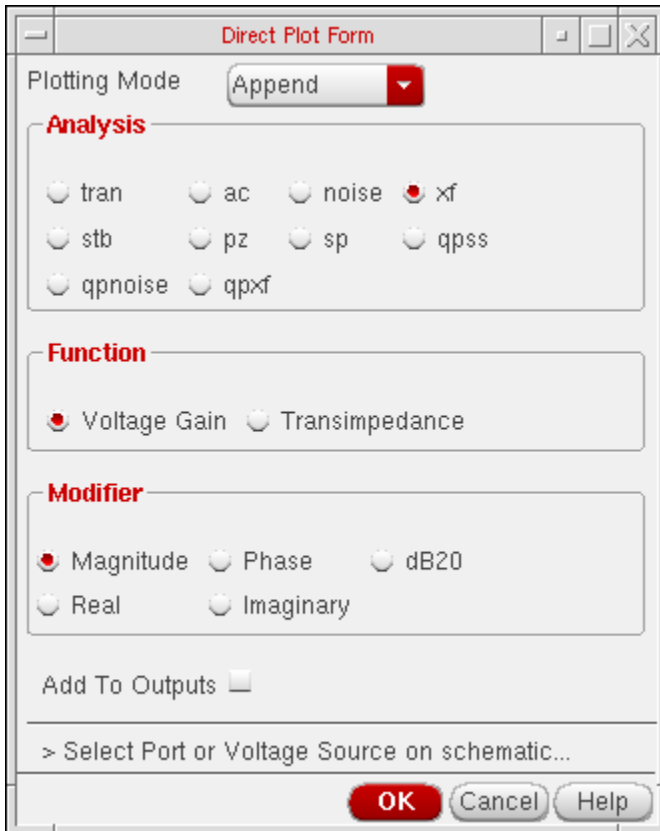
$VN2 =$ The noise voltage
 $Vin =$ The voltage at the input node
 $Vout =$ The voltage at the output node
 $R =$ The source resistor value
 $C =$ $1.61e-20 \cong 4kT\Delta f$
with
 $T =$ 291.5K and $\Delta f = 1$

For Transfer Functions

To plot the transfer function,

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

The XF Results form appears.



For detailed information about the form, see [“XF Results”](#) on page 560.

2. Specify the *Plotting Mode*. You can specify:
 - Append* to append the new signal to the current graph.
 - Replace* to replace the current graph with the new signal. This is the default option.
 - New SubWin* to plot the signal in a new subwindow.
 - New Win* to plot the signal in a new window.
3. Choose either *Voltage Gain* or *Transimpedance* if you selected output voltage for the transfer analysis, or *Current Gain* or *Transconductance* if you selected output current for the transfer analysis.
4. Specify the modifiers as needed.
5. Select either the instance or instance terminal in the schematic.

The graph window redisplay, showing the new plot.

6. To replot with modifications, make changes to the specifications on the XF Results form and click *Replot*.

For S-Parameters

A typical S-parameter direct plot shows a parameter function plotted against frequency, based on a pair of psin elements that define an input and an output circuit port.

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

The plots appear, by default, in the current Virtuoso® Analog Design Environment graph window or subwindow. The current subwindow has a rectangle around its window number (in the upper-right corner). To use a different subwindow, select it before beginning the direct plot procedure. If no graph window or subwindow is open, this plot function automatically opens one.

The *Results – Direct Plot – S-Parameter* command automatically opens

- A graph window (unless one is already open)
- The design schematic (unless it is already open)

■ The S-Parameter Results form

Plotting Mode ▼

Analysis

tran ac noise stb
 pz sp qpss qpnoise
 qp>f

Function

SP ZP YP HP
 GD VSWR NFmin Gmin
 Rn m NF Kf
 B1f GT GA GP
 Gmax Gmsg Gumx ZM
 NC GAC GPC LSB
 SSB

Description: S-Parameter

Plot Type

Rectangular Z-Smith Y-Smith
 Polar

Modifier

Magnitude Phase dB20
 Real Imaginary

Add To Outputs

For detailed information about the form, see [“S-Parameter Results”](#) on page 561.

To plot S-parameter results,

1. Specify the *Plotting Mode*. You can specify:

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- Append* to append the new signal to the current graph.
- Replace* to replace the current graph with the new signal. This is the default option.
- New SubWin* to plot the signal in a new subwindow.
- New Win* to plot the signal in a new window.

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

Function

SP ZP YP HP

GD VSWR NFmin Gmin

Rn rn NF Kf

B1f GT GA GP

Gmax Gmsg Gumx ZM

NC GAC GPC LSB

SSB

Description: Load Stability Circles

A brief description of the function appears below the buttons, and the bottom of the form changes to show options for the function.

Note: Some functions are defined only for two-port circuits. If you choose a function that is not available for your circuit data set, a warning message appears at the bottom of the form. Click a button on the figure for information about a function. If you need an equation that is not represented on the form, use the calculator to build, evaluate, and plot it.

3. Choose the appropriate *Plot Type* and *Modifier* to specify the plot type and the data or plot format.
4. Specify and draw the plot.

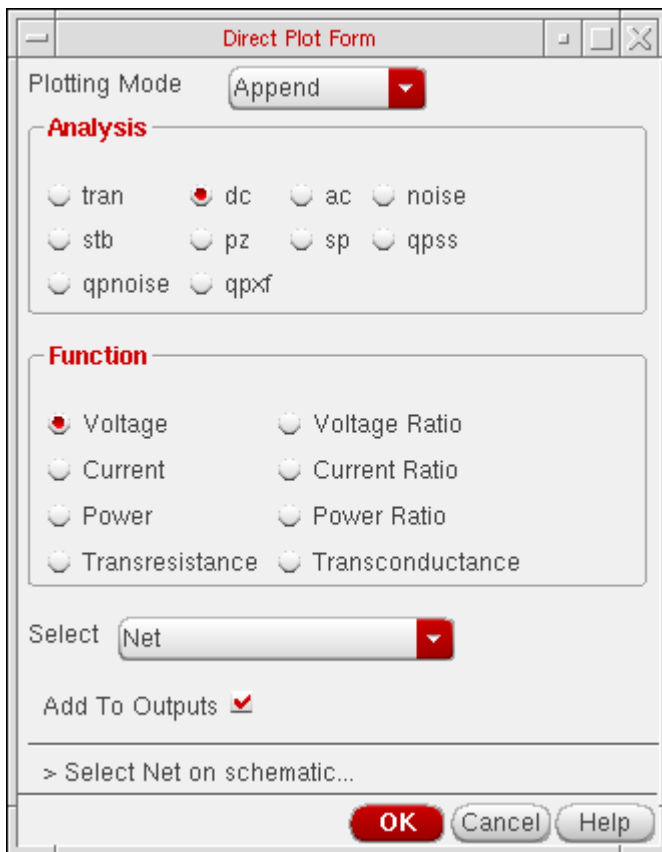
For S, Z, Y, or H parameters (shown as *SP*, *ZP*, *YP*, and *HP* on the form), generate plots for ports 1 through 3 by clicking the appropriate parameter button at the bottom of the form. To generate plots for any higher-numbered ports, use the cyclic fields beside the buttons to specify the output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

Note: For circuits with three or fewer ports, the form has no cyclic fields.

Using the Direct Plot Main Form

For DC

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



2. Specify the *Plotting Mode*. You can specify:

- Append* to append the new signal to the current graph.

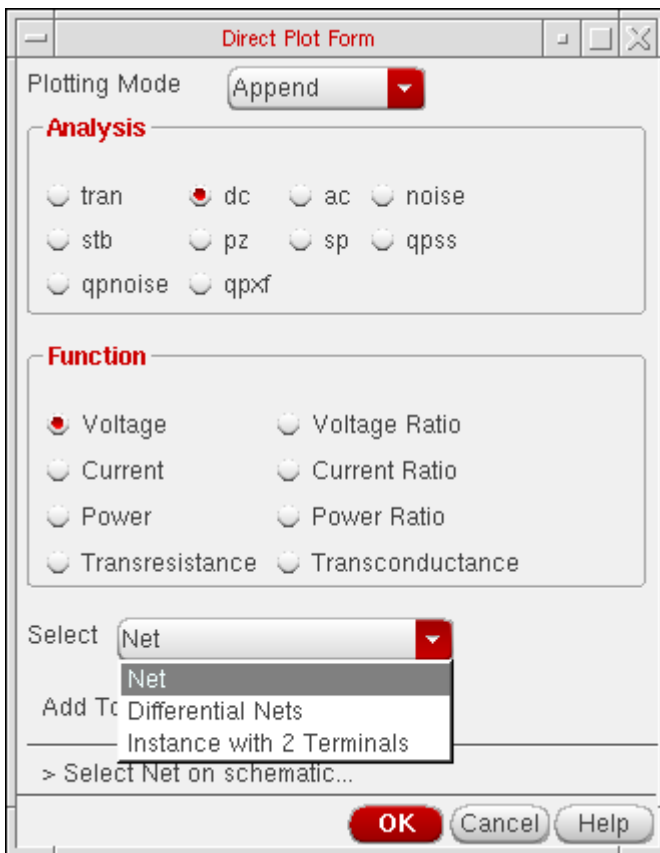
Note: The *Append* mode is not recommended for plots with different scales and units for the X axis. It can give strange results because *Virtuoso Visualization and Analysis XL* opens a new subwindow to plot any data that does not match the units and range of the existing traces.

- Replace* to replace the current graph with the new signal. This is the default option.
- New SubWin* to plot the signal in a new subwindow.
- New Win* to plot the signal in a new window.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

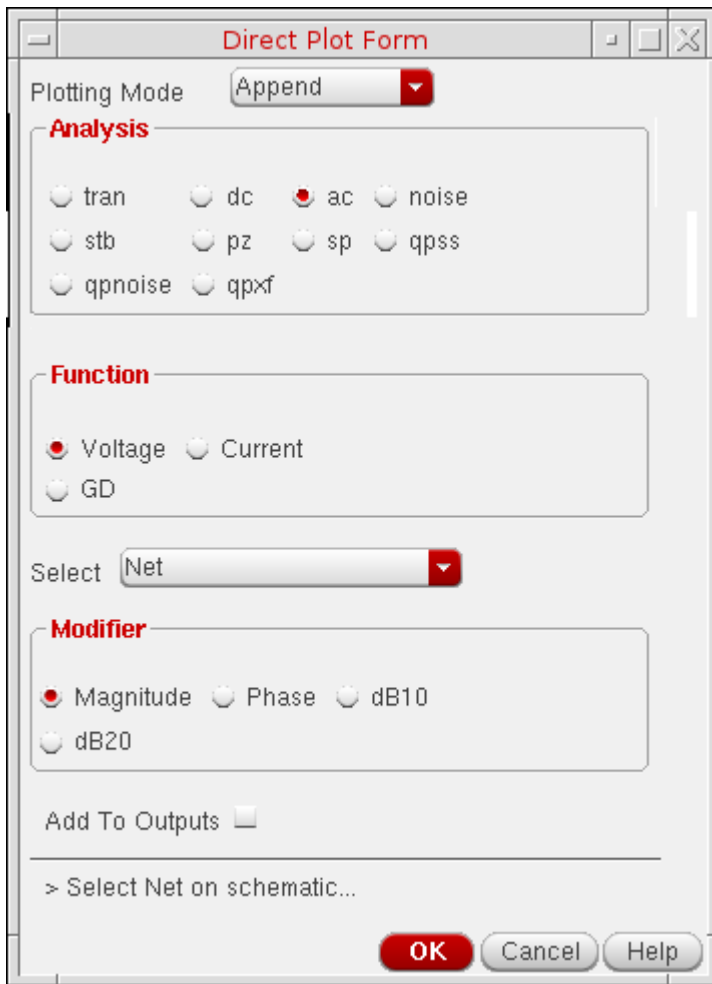
- The functions that are available are: *Voltage*, *Voltage Ratio*, *Current*, *Current Ratio*, *Power*, *Power Ratio*, *Transresistance* and *Transconductance*. Based on the selected function and available data, the form changes dynamically to display the applicable options.
- Choose the nets and terminals to plot. You can select *Net*, *Differential Nets* or *Instance with 2 Terminals*.



- Enable *Add To Outputs* to add expressions for the results to the outputs section and plot in the mode that you selected.

For AC

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



2. Specify the *Plotting Mode*. You can specify:

- Append* to append the new signal to the current graph.

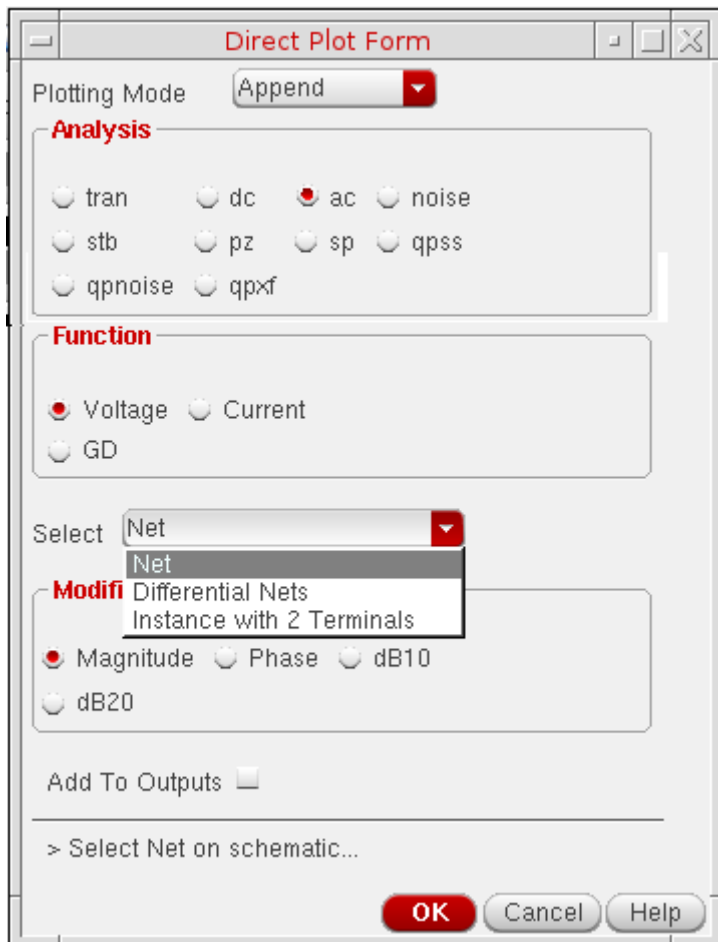
Note: The *Append* mode is not recommended for plots with different scales and units for the X axis. It can give strange results because *Virtuoso Visualization and Analysis XL* opens a new subwindow to plot any data that does not match the units and range of the existing traces.

- Replace* to replace the current graph with the new signal. This is the default option.
- New SubWin* to plot the signal in a new subwindow.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- New Win* to plot the signal in a new window.
- 3. The functions that are available are: *Voltage*, *Current*, *Current Ratio* and *GD*. Based on the selected function and available data, the form changes dynamically to display the applicable options.
- 4. Choose the nets and terminals to plot. You can select *Net*, *Differential Nets* or *Instance with 2 Terminals*.



- 5. Choose the appropriate *Modifier* to specify the data or plot format.
- 6. Enable *Add To Outputs* to add expressions for the results to the outputs section and plot in the mode that you selected.

For Transient Results

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

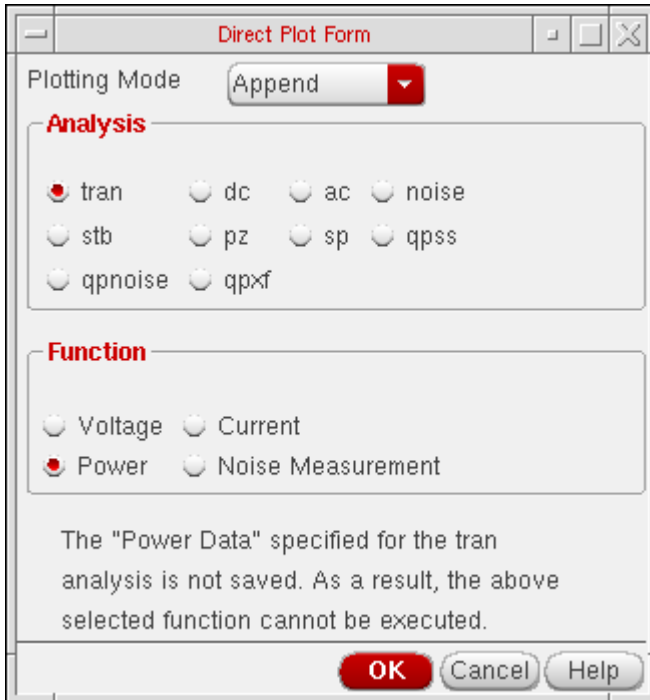
The image shows a dialog box titled "Direct Plot Form". At the top, "Plotting Mode" is set to "Append". Below this is the "Analysis" section with radio buttons for "tran" (selected), "dc", "ac", "noise", "stb", "pz", "sp", "qpss", "qpnoise", and "qpxf". The "Function" section has radio buttons for "Voltage" (selected), "Current", "Power", and "Noise Measurement". A "Select" dropdown menu is set to "Net". There is a "Subtract dcOp" checkbox which is unchecked. Below that is a checked checkbox for "Prepend Waveform from Reference Directory" with an empty text field. At the bottom of the form area is a checked checkbox for "Add To Outputs". A prompt "> Enter a reference directory on this form..." is located below the "Add To Outputs" checkbox. At the very bottom are "OK", "Cancel", and "Help" buttons.

2. Specify the *Plotting Mode*. You can specify:
 - Append* to append the new signal to the current graph.
 - Replace* to replace the current graph with the new signal. This is the default option.
 - New SubWin* to plot the signal in a new subwindow.
 - New Win* to plot the signal in a new window.
3. The functions that are available are: *Voltage*, *Current* and *Power*. Based on the selected function and available data, the form changes dynamically to display the applicable options. Most of the options are similar to those available in the *Direct Plot*

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

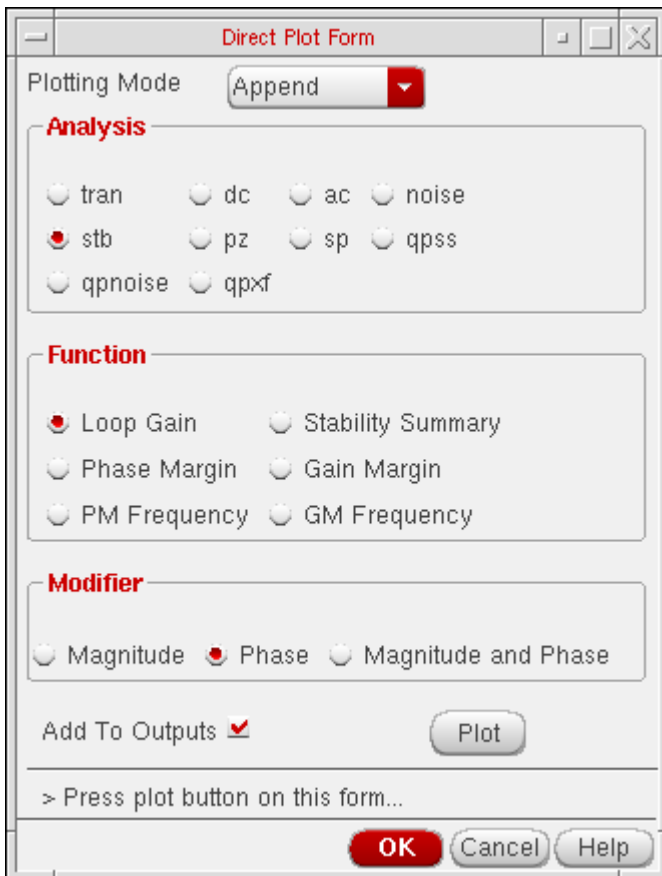
form For DC analysis. If no power data is available, a corresponding message is displayed on the form:



4. The *Prepend Waveform from Reference Directory* option can be used for appending multiple checkpoint/restart transient waveforms together to enable you to view complete waveforms. Specify the reference results directory(s) in the field. The signal you choose in your direct plot will then be accessed from the reference directory.

For Stability Results

1. Choose *Results – Direct Plot – Main Form*. The *Direct Plot Form* appears. Select the *stb* option in the *Analysis* section. The form re-displays accordingly.



2. Specify the *Plotting Mode*. You can specify:
 - Append* to append the new signal to the current graph.
 - Replace* to replace the current graph with the new signal. This is the default option.
 - New SubWin* to plot the signal in a new subwindow.
 - New Win* to plot the signal in a new window.
3. The functions that are available are: *Loop Gain*, *Stability Summary*, *Phase Margin*, *Gain Margin*, *PM Frequency* and *GM Frequency*. Based on the selected function and available data, the form changes dynamically to display the applicable options.
 - a. When you select *Loop Gain*, the form re-displays to show the *Modifier* section. The loop gain output is a complex waveform and you can select it to plot *Magnitude*,

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Phase or both (*Magnitude and Phase*). Whenever you choose to plot *Magnitude*, the *Magnitude Modifier* section appears on the form. You can select *None*, *dB10* or *dB20*, as needed. Whenever you plot both the magnitude and phase, the graph window changes to the strip mode. It reverts back to the composite mode for other plot operations.

Function

Loop Gain Stability Summary

Phase Margin Gain Margin

PM Frequency GM Frequency

Modifier

Magnitude Phase Magnitude and Phase

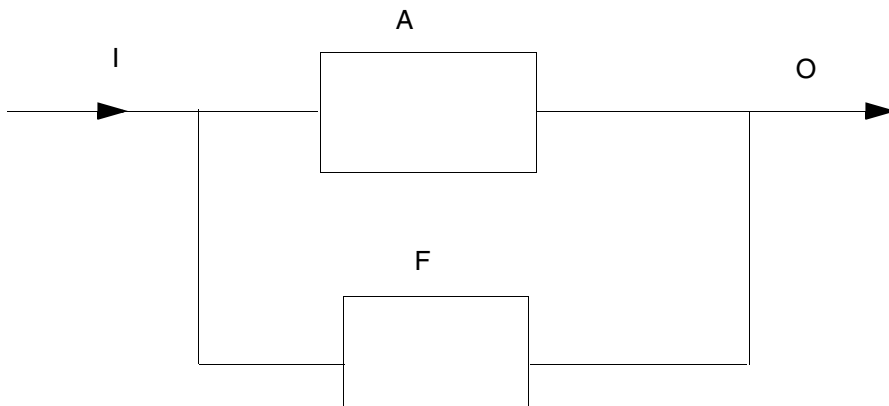
Magnitude Modifier

None dB10 dB20

Add To Outputs

> Press plot button on this form...

Note: There is a difference between the ADE L and Spectre definition of loop gain. For the feedback circuit shown below:



The closed loop gain is defined as:

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

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

The Spectre output defines loop gain as the product **AF**, while others (like the ADE L Calculator `phaseMargin` and `gainMargin` functions) define **-AF** as the loopGain. Therefore, to obtain the same results from ADE L, you need to negate the Spectre's loopGain as illustrated below:

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

- b. Phase Margin, Gain Margin, PM Frequency and GM Frequency** constitute the margin data. This information is calculated from the loop gain data for the circuit. The information is only available when frequency is swept in the stability analysis and the swept range is sufficient to calculate the values. When the selected margin data is scalar the values are displayed on the form itself.

Function

Loop Gain Stability Summary

Phase Margin Gain Margin

PM Frequency GM Frequency

PM metric not found. Perhaps the stb frequency sweep was not adequate.
The selected function cannot be executed.

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

Function

Loop Gain Stability Summary

Phase Margin Gain Margin

PM Frequency GM Frequency

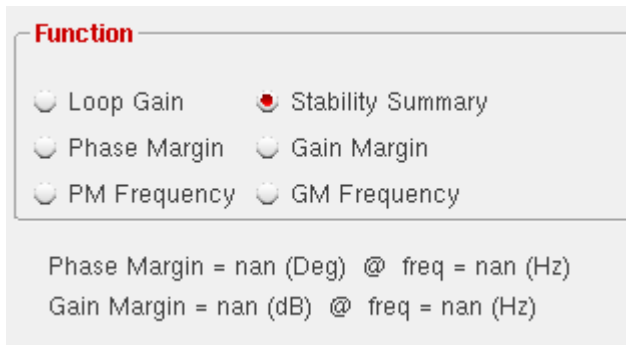
GM metric not found. Perhaps the stb frequency sweep was not adequate.
The selected function cannot be executed.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

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

- c. Selecting *Stability Summary* displays all the margin data collectively on the form, when the data is scalar. You do not have the facility to plot or add the four outputs when this function is chosen. Use the individual margin data function for this operation.



Function

Loop Gain Stability Summary

Phase Margin Gain Margin

PM Frequency GM Frequency

Phase Margin = nan (Deg) @ freq = nan (Hz)
Gain Margin = nan (dB) @ freq = nan (Hz)

When frequency was not swept or the margin data is not scalar an appropriate error is reported in the *Direct Plot Form* and the *Plot* and *Add to Outputs* button are not available.

4. Enable *Add To Outputs* and plot in the mode that you selected.

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

All other parts of the *Direct Plot* form work the same way as they do for other analyses. Refer to the *Virtuoso Spectre Circuit Simulator RF Analysis User Guide* for details.

This form handles parametric (family) data. The *Loop Gain* would be a set of curves for family data. Similarly for non-parametric data, *Phase Margin* and *Gain Margin* will be scalars. A horizontal straight line will be plotted for them.

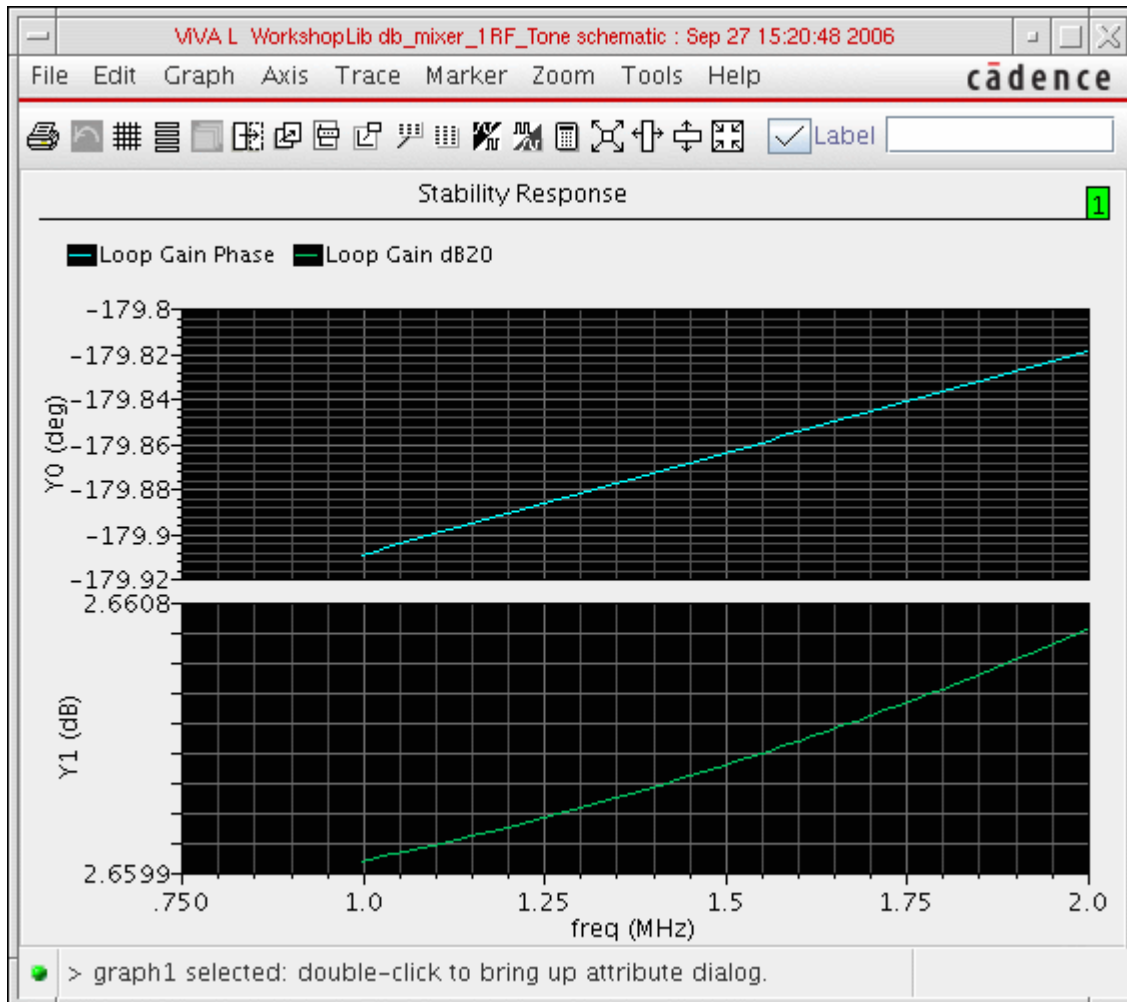
Example

A wave form window when plotted with Magnitude and phase (dB20) for non-family data is displayed. The expression/waveform names created for the outputs are: Loop Gain, Loop

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Gain dB10, Loop Gain db20, Loop Gain Phase, Gain Margin, Phase Margin, Gain Margin Frequency and Phase Margin Frequency.



For Pole Zero Results

Once you run a simulation for *Pole Zero* analysis, you can use the *Direct Plot* main form to view the poles and zeros plotted on the real/imaginary plane in the *Analog Waveform Display* window.

1. To access the *Direct Plot* main form, select *Results – Direct Plot – Main Form*.

2. Select the *Pole Zero Analysis* option. The *Direct Plot Form* changes dynamically to display the applicable functions and options:

The screenshot shows the 'Direct Plot Form' dialog box. It features a 'Plotting Mode' dropdown menu set to 'Append'. The 'Analysis' section has a radio button labeled 'pz' selected. The 'Function' section has three radio buttons: 'Poles And Zeros' (selected), 'Poles', and 'Zeros'. The 'Filter Out' section contains two checkboxes: 'Max Frequency(Hz)' with a text input field containing '1.000e+12', and 'Real Value <=' with an empty text input field. Below these sections is an 'Add To Outputs' checkbox which is checked, and a 'Plot' button. At the bottom of the dialog, there is a note '> Press plot button on this form...' and three buttons: 'OK', 'Cancel', and 'Help'.

3. Select the option, *Poles* if you want to plot only poles, *Zeros* if you want to plot only zeros and *Poles and Zeros* if you want to plot both poles and zeros.

Note: By default, the option *Poles and Zeros* is selected.

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

- Max Frequency:**
This option enables you to filter out poles and zeros that are outside the frequency band of interest (FBOI) and that do not influence the transfer function in the FBOI. The default value is that specified in the *fmax* field in the *Pole-Zero Options* form. Only poles and zeros whose magnitudes exceed the frequency value specified are filtered out.
- Real Value:**
This option enables you to specify the real part of the frequency. Only poles and

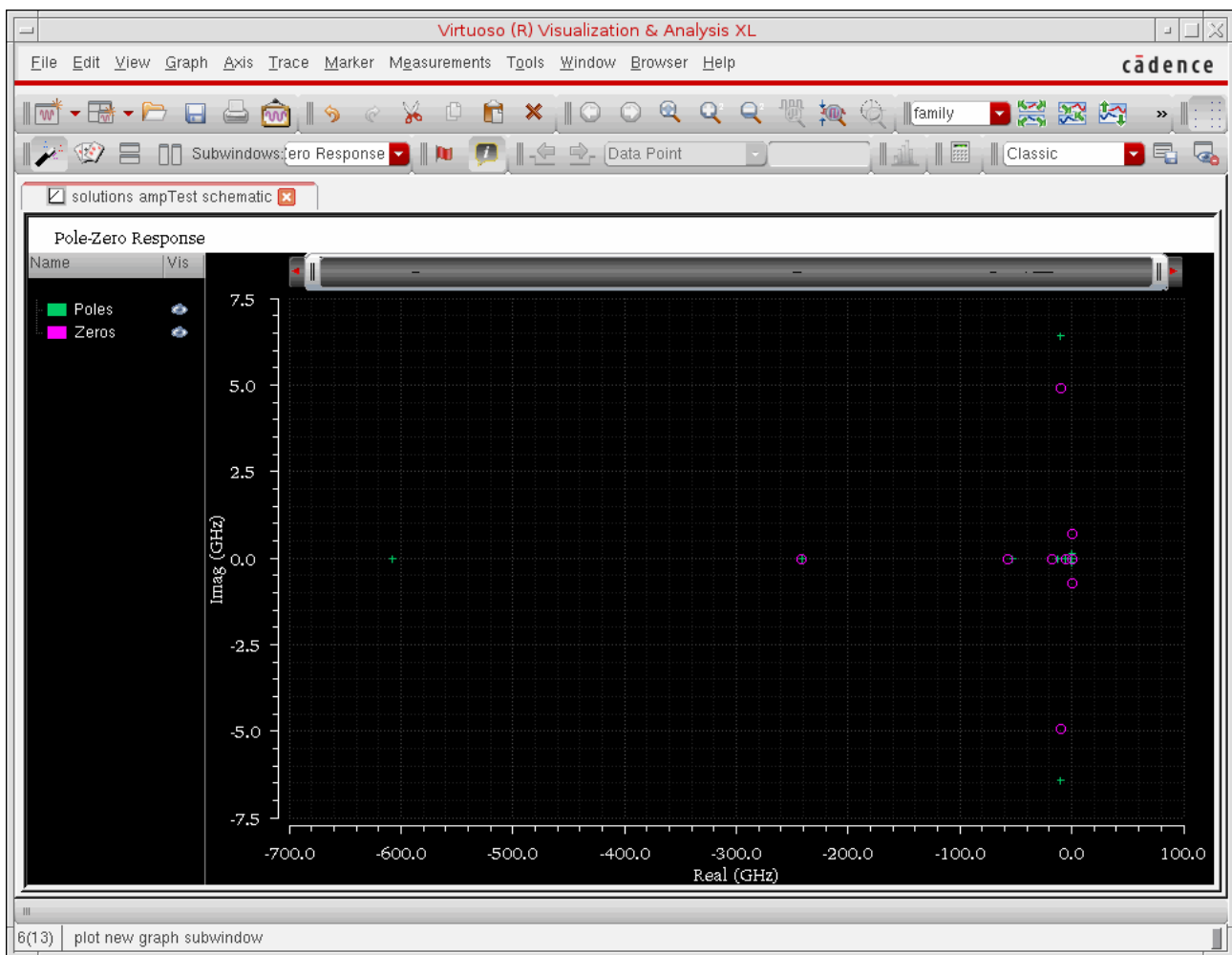
Virtuoso Analog Design Environment L User Guide

Plotting and Printing

zeros whose real values are less than or equal to the real value specified are filtered out.

Note: By default, no filtering is selected. You can set the filtering criteria once you specify either poles or zeros or both to be plotted.

5. Enable *Add To Outputs* to add expressions for the results to the outputs section and plot in the mode that you selected.
6. Click *Plot* to view the results in the *Graph Window*:



Poles and zeros are plotted in *scatter* mode. This implies that poles and zeros are plotted individually but not connected. Poles are represented by the symbol **x** and zeros by the symbol **o**. The complex data is plotted with poles and zeros.

Non-Swept Parameters


For the non-swept case, the result of Pole Zero analysis will be two waveform objects, one representing poles and another representing the zeros. The two wave objects are plotted in the same color however, *poles* will be represented by the symbol **x** and *zeros* by the symbol **o**.

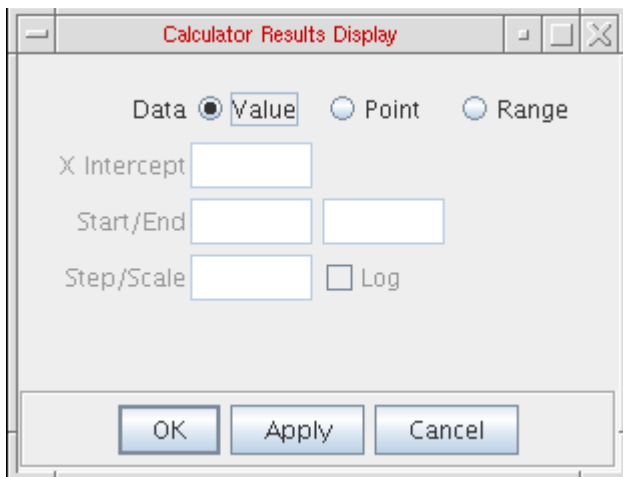
Swept Parameters

For swept parameter Pole Zero Analysis, it is possible to create the root-locus plot. Instead, the poles and zeros are plotted corresponding to each *Swept Parameter* value.

Overview of Printing

The following commands and tools within the Analog Design Environment print text results and reports to the Results Display Window.

- *Results – Print* menu commands in the Simulation window
- The *Tabular Results Display* () button in Virtuoso Visualization and Analysis XL Calculator. This brings up the *Display Results* window.

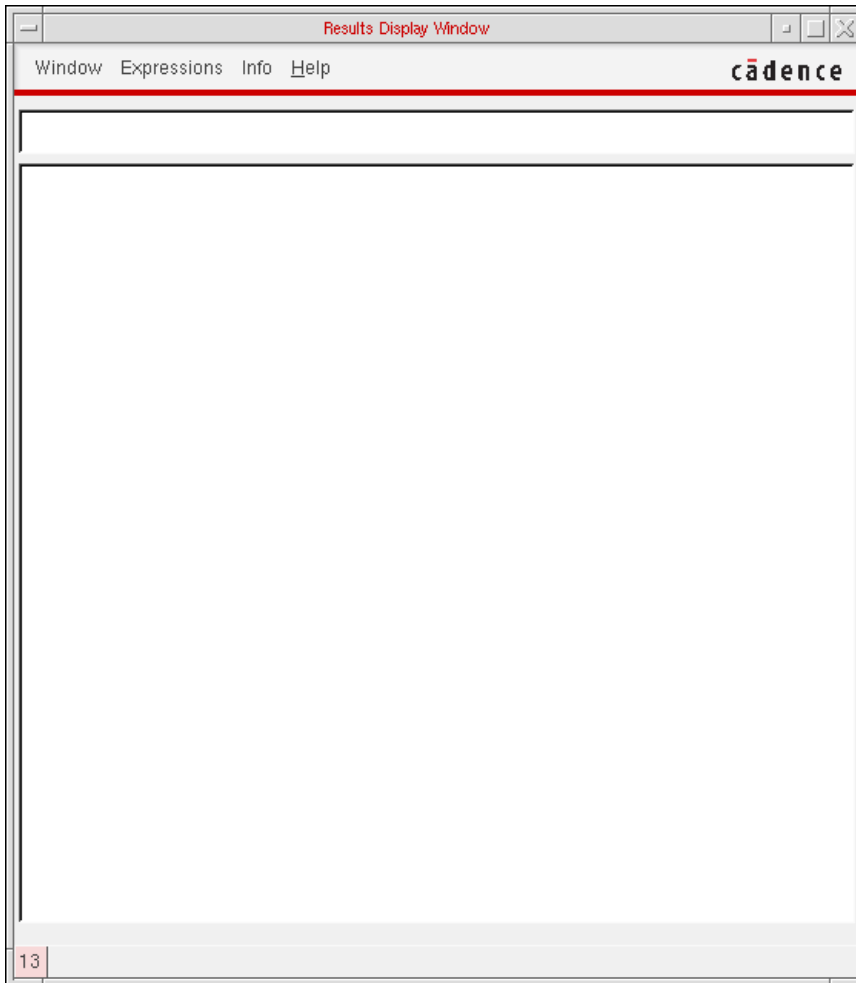


- *Statistics – Print* menu commands
- The *Markers* menu options in Virtuoso Visualization and Analysis XL Calculator.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- Using any of these commands opens the Results Display Window.



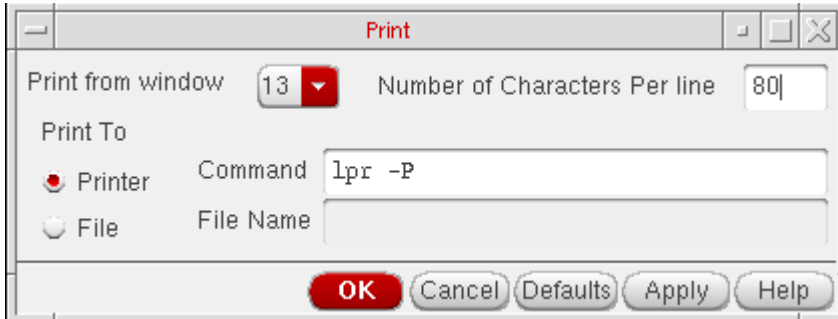
For guidance on using the Results Display Window to perform tasks, see the following sections.

Printing Results

To print the results in the Results Display Window either in hardcopy or to a file,

1. Choose *Window – Print*.

The Print form appears.



2. Choose the correct window number from the *Print from window* cyclic field.
This is the window containing the contents you want to print.
3. Type a value in the *Number of Characters Per Line* field.
4. Choose either the *Printer* or *File* radio button in the *Print To* field.
You must type a filename if you choose *File*.
5. Click *OK*.

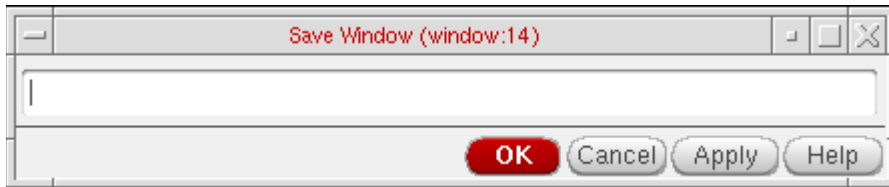
Saving State

You can use *Save State* and *Load State* capability to save the current setup of display options for printing waveforms such as printing format, setting a printing range if the amount of data is too large, printing at a certain interval, and changing the order of the display. You can save the state of the window into a file. Later if you run another simulation and do *Load State*, the new data can be loaded back and displayed as you specified when you saved the state. *Save State* and *Load State* are applicable only to waveforms (that is, expressions that can evaluate to a waveform). If you print out a single number, like a node voltage, these commands are disabled. You get a message stating this value is not a waveform and cannot be loaded back.

To save the contents and format of a Results Display Window,

1. Choose *Window – Save State*.

The Save Window form appears.



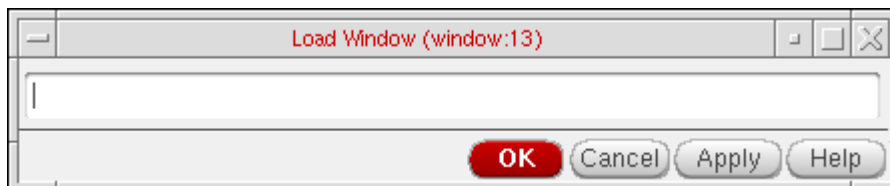
2. Type a filename in the field.
3. Click *OK*.

Loading State

To load a window state that you previously saved,

1. Choose *Window – Load State*.

The Load Window form appears.



2. Type the name of the saved file in the field.
3. Click *OK*.

Updating Results

To update the Results Display Window with results from a new simulation,

- Choose *Window – Update Results*.

This updates the data using the current window setup. *Update Results* is applicable only to waveforms (that is, expressions that can evaluate to waveforms). If you print out a single number, like a node voltage, this command is disabled.

Making a Window Active

There is no limit to the number of Results Display Windows you can have open, but only one window is active at a time. All printouts go to the active window.

To make a window active,

- Choose *Window – Make Active* in the window that you want to make active.

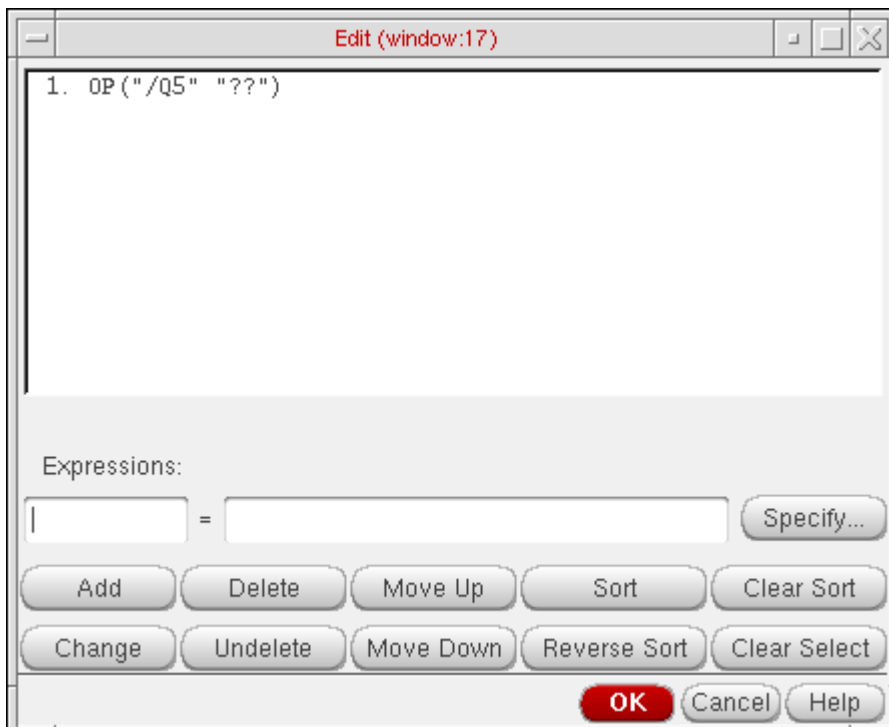
Editing Expressions

You can edit any expressions that evaluate to waveforms (for example, DC operating parameters, model parameters, and transient operating parameters). If you print only one value, the edit menu choices are not available. The editing commands operate on only the last table in the active Results Display Window.

To edit expressions in the print window,

1. Choose *Expressions – Edit*.

The Edit window appears.



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

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Note: You should convert both numbers to the same type (that is, integer or float) before you compare them. For more information, see *Integer vs. Floating-Point Division* in the *Arithmetic and Logical Expressions* chapter of the *Cadence SKILL Language User Guide*.

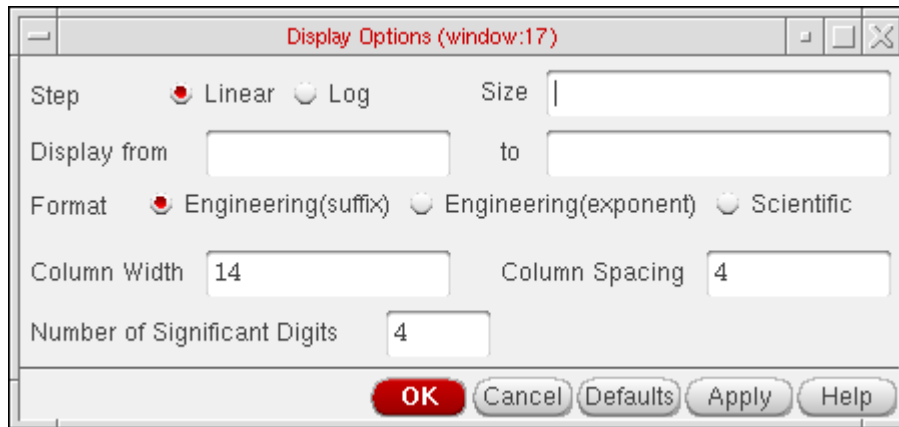
<i>Expressions</i>	The field to the left of the equal sign shows the aliased name of the expression to the right. Naming expressions is optional and the field might be blank. If aliases are used, they are shown in the list box and the title line of the print window list box.
<i>Specify</i>	Retrieves the expression in the calculator buffer and places it into the edit field.
<i>Add</i>	Adds the expression in the edit field to the list box.
<i>Change</i>	Replaces the selected expression in the list box with the one in the edit field.
<i>Delete</i>	Deletes the selected expression from the list box.
<i>Undelete</i>	Lets you undo the last delete.
<i>Move Up</i>	Moves the display of the selected expression one step to the left. If the expression is already the leftmost, it is moved to the rightmost.
<i>Move Down</i>	Moves the display of the selected expression one step to the right. If the expression is already the rightmost, it is moved to the leftmost.
<i>Sort</i>	Sorts the selected expression so that the value increases down the column.
<i>Reverse Sort</i>	Sorts the selected expression so that the value decreases down the column.
<i>Clear Sort</i>	Reverts to the default order.
<i>Clear Select</i>	Clears the selection in the list box. Also, you can clear entries from the list box by clicking on the entry while holding down the <code>Control</code> key.

Setting Display Options

To change the display options,

1. Choose *Expressions – Display Options*.

The Display Options form appears.



2. Type the values into the form and select a format.

<i>Step size</i>	Specifies the interval for printing data.
<i>Display from, to</i>	Specifies the range of data to print. If <i>from</i> is left blank, the data is printed from the beginning. If <i>to</i> is blank, the data is printed to the end. You can set the print range only after printing data.
<i>Format</i>	Controls the format of the data printed. The possible formats are <i>Engineering Suffix</i> (default), <i>Engineering</i> , and <i>Scientific</i> . For example, you represent 0.0001 as 0.1m (engineering suffix), 0.1e-3 (engineering), or 1e-4 (scientific).
<i>Linear/Log</i>	Specifies whether the scale used for step size is linear or logarithmic.
<i>Column width/spacing</i>	Changes the number of characters allowed for column width and spacing. The default width is 14 characters. The allowed range is 4 to 20 characters. The default spacing is 4 and the allowed range is 1 to 10.

Number of significant digits

Specifies the number of significant digits to be printed. The default is 4 digits, and the allowed range is 2 to 10.

Note: If the Results Display Window contains more than one type of results, the *Display Options* commands apply only to the last result (if the last result can evaluate to a waveform). After the data is edited, only the last result appears in the window. If you want to preserve the previous results, you can open a new Results Display Window and print the results to be edited in the new window.

Displaying Output Information

To display output information,

- Choose *Info – Show Output*.

Output names are truncated to fit into columns if they are too long. The *Show Output* command shows the output names in full.

Specifying Results to Print

Before you can print results, you need to specify which results to print.

1. Do one of the following:
 - Run a simulation.
 - In the Simulation window, choose *Results – Select*, choose the desired data file, and click *OK*.
2. Make sure the Schematic window for the selected design is open.

To print results for the current simulation or for a selected data file,

1. Choose a print command from the *Results* menu.
2. Select a node in the Schematic window.

The Results Display Window shows

- The command syntax for the print option you selected
- The results for the instance you selected

Each time you click a node in the Schematic window, information about the node is added to the Results Display Window.

Printing DC Operating Points

To print the DC operating points of the components in your circuit,

1. Choose *Results – Print – DC Operating Points*.
2. Move your cursor into the Schematic window.

The CIW prompts you to select instances for the operating point output.

3. Click an instance.

If the selected instance is a textual subcircuit, operating points for all devices in the subcircuit will be printed. It may take some time to search for all instances in a textual subcircuit. To disable the feature, set the following environment variable in your `.cdsenv`:

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

Printing Transient Operating Points

To print the final transient operating points of the nodes or components in your circuit,

1. Choose *Results – Print – Transient Operating Points*.
2. Move your cursor into the Schematic window.

The CIW prompts you to select instances for the transient operating point (OPT) output.

3. Click an instance or node.

If the selected instance is a textual subcircuit, operating points for all devices in the subcircuit will be printed. It may take some time to search for all the instances in a textual subcircuit. To disable the feature, set the following environment variable in your `.cdsenv`:

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

Printing Model Parameters of Components

To print the model parameters of the nodes or components in your circuit,

1. Choose *Results – Print – Model Parameters*.
2. Move your cursor into the Schematic window.

The CIW prompts you to select instances for the model parameter output.

3. Click an instance of a device.

If the selected instance is a textual subcircuit, model parameter for all devices in the subcircuit will be printed. It may take some time to search for all instance in a textual subcircuit. To disable the feature, set the following environment variable in your `.cdsenv`:

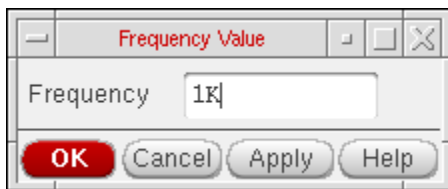
```
asimenv.printing printInlines boolean nil
```

Printing Noise Parameters of Nodes or Components

To print the noise parameters of the nodes or components in your circuit,

1. Choose *Results – Print – Noise Parameters*.

The Select Frequency Value form appears.



If the form does not appear, press F3.

2. In the *Frequency* field, type the frequency value at which you want the noise parameters to print.

The default frequency is 1K.

3. Move your cursor into the schematic window.

The CIW prompts you to select instances for the VNP output.

4. Click an instance or node.

Noise Summary

To display the noise contribution of the components in a circuit,

1. Run a noise analysis simulation.
2. Choose *Results – Print – Noise Summary*.

The *Noise Summary* form appears.

The screenshot shows the "Noise Summary" dialog box. It is titled "Noise Summary" and has a standard Windows-style title bar. The dialog is organized into several sections:

- Data from:** Two radio buttons are present: "hbnoise" (selected) and "hbnoise_src". Below them is a text field containing "Print the output noise of `hbnoise-PORT2` analysis".
- Type:** Two radio buttons: "spot noise" (selected) and "integrated noise". To the right is a "noise unit" dropdown menu currently set to "V^2".
- Frequency Spot (Hz):** A text input field containing "1k".
- FILTER:** A "hierarchy level" dropdown menu set to "5". Below this are two buttons: "Include All Types" and "Include None". To the right is a list box containing the following items: "b3v3", "inductor", "subckt", and "phy_res". Below the list box are two text input fields: "include instances" and "exclude instances". Each of these fields has "Select" and "Clear" buttons to its right.
- TRUNCATE & SORT:** A "truncate" dropdown menu set to "by number", followed by a "top" text input field containing "3". Below this are three radio buttons: "noise contributors" (selected), "composite noise", and "device name".
- Buttons:** At the bottom right, there are four buttons: "OK" (highlighted in red), "Cancel", "Apply", and "Help".

For detailed information about the form, see "[Noise Summary](#)" on page 565.

3. Choose either *spot noise* and its frequency or *integrated noise* and a range of frequencies.
4. If you choose *integrated noise*, you have the option of using a weighting factor.

The *flat* weighting factor specifies that the integration be performed on the original unweighted waveform.

The *from weight file* selection specifies that, before the integration is performed, the noise contributions of particular frequencies in the original waveform be weighted by factors supplied from an input file. The weighting file must have one of the following entries on the first line: db, mag, db1, DB, MAG, DBL. Each additional line must contain a pair of X and Y values. All the pairs together must define a function. For example:

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

mag	
1	.001641
60	.001641
100	.007499
200	.05559

5. Choose filtering details to include or exclude particular instances in your summary.
6. If needed, choose truncation details to shorten your summary.

You can shorten your summary by specifying how many of the highest contributors to include in the summary, by specifying the percentage of noise a device must contribute to be included in the summary, or by specifying the level of noise a device must contribute to be included in the summary.

7. Choose a sorting method.
8. Click *OK*.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

The Results Display window displays the noise summary using the criteria that you specified to shorten and order the list.

The screenshot shows a window titled "Results Display Window" with a menu bar (Window, Expressions, Info, Help) and the Cadence logo. The main content is a table with the following columns: Device, % Of Total, Inp Ref Noise Param, and Noise Contribution. The table lists noise data for devices /L2, /Q1, /Q2, /Q3, /Q4, and /Q5, including total noise and individual parameter contributions like fn, re, rn, ic, rc, ib, and rb.

Device	% Of Total	Inp Ref Noise Param	Noise Contribution	
/L2	0.00	0	total	0
			fn	0
			rn	0
/Q1	13.81	2.57583e-16	total	2.71731e-18
			re	4.49665e-23
			ic	4.76715e-20
			rc	3.27944e-25
			fn	9.19542e-21
			ib	2.65966e-18
			rb	7.43884e-22
/Q2	0.02	3.59528e-19	total	3.79275e-21
			re	1.28906e-22
			ic	4.37726e-22
			rc	1.51705e-27
			fn	4.13197e-24
			ib	1.39215e-21
			rb	1.82984e-21
/Q3	13.33	2.48806e-16	total	2.62472e-18
			re	4.53661e-23
			ic	4.0822e-20
			rc	2.96597e-25
			fn	9.31037e-21
			ib	2.57371e-18
			rb	8.37026e-22
/Q4	18.25	3.40574e-16	total	3.59281e-18
			re	1.57578e-20
			ic	1.3864e-19
			rc	9.4399e-25
			fn	1.08915e-20
			ib	3.15112e-18
			rb	2.764e-19
/Q5	0.08	1.43476e-18	total	1.51357e-20
			re	5.14432e-22

The precision of the noise data displayed in the *Results Display* window, can be controlled using the cdsenv variable "digits" of the tool[.partition] "asimenv.noiseSummary". The default value for this variable is 6 and can be set to any other integer value using the following command on the CIW prompt:

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Example

```
envSetVal("asimenv.noiseSummary" "digits" 'int 10)
```

This will set the value of the variable to 10.

The number of decimals printed for any relative contribution is controlled using the cdsenv variable *percentDecimals* of the tool[.partition] "asimenv.noiseSummary". The default value for this variable is 2 and can be set to any other integer value using the following command on the CIW prompt:

Example

```
envSetVal("asimenv.noiseSummary" "percentDecimals" 'int 4)
```

This will set the value of the variable to 4.

Printing DC Mismatch Summary

To print the DC Mismatch summary in your circuit,

1. Choose *Results – Print – Mismatch Summary*.

Note: This menu option is enabled when ever dcmatch analysis is included in the last run or the results directory specifically selected through ADE L, contains the results for dcmatch analysis.

The *DC Mismatch Summary* form appears

spectre0: Dcmatch Summary

Device Mismatch Data

Filter

Include all types

Include none

bjt
resistor

Include Instances

Exclude Instances

Select

Clear

Select

Clear

Variations To Print

Device Type

bjt

sigmaOut
sigmaVbe

Include all columns

Include none

Truncate & Sort

Truncate

by relative threshold

threshold

0.001

Sort

Output Variation Device Name

OK Cancel Apply Help

2. Specify a value in the *Print results when value is* field.
3. Specify the type of devices you need to print the results for, in the *Filter* section. The *Include all types* and *Include none* buttons can be used to include or exclude all types at a single click. You can include specific instances or exclude specific instances. You can either type the instance names or use the select buttons to pick them from schematics. The *Clear* button is used to clear the fields.
4. Specify the information to be made available for the various device types, in the *Variations to Print* section. The *Include all columns* and *Include none* buttons can be used for easier list box operation.

5. Truncate and sort data by top contributors and relative/absolute contribution. The default is relative contribution with the threshold being the value of the threshold parameter used on the analysis line. You can sort by variation or device name.

Printing Stability Summary

To print the stability summary in your circuit,

1. Choose *Results – Print – Stability Summary*. The *Stability Summary* form appears. The form enables you to print *Phase Margin*, *Gain Margin* or *Both*.

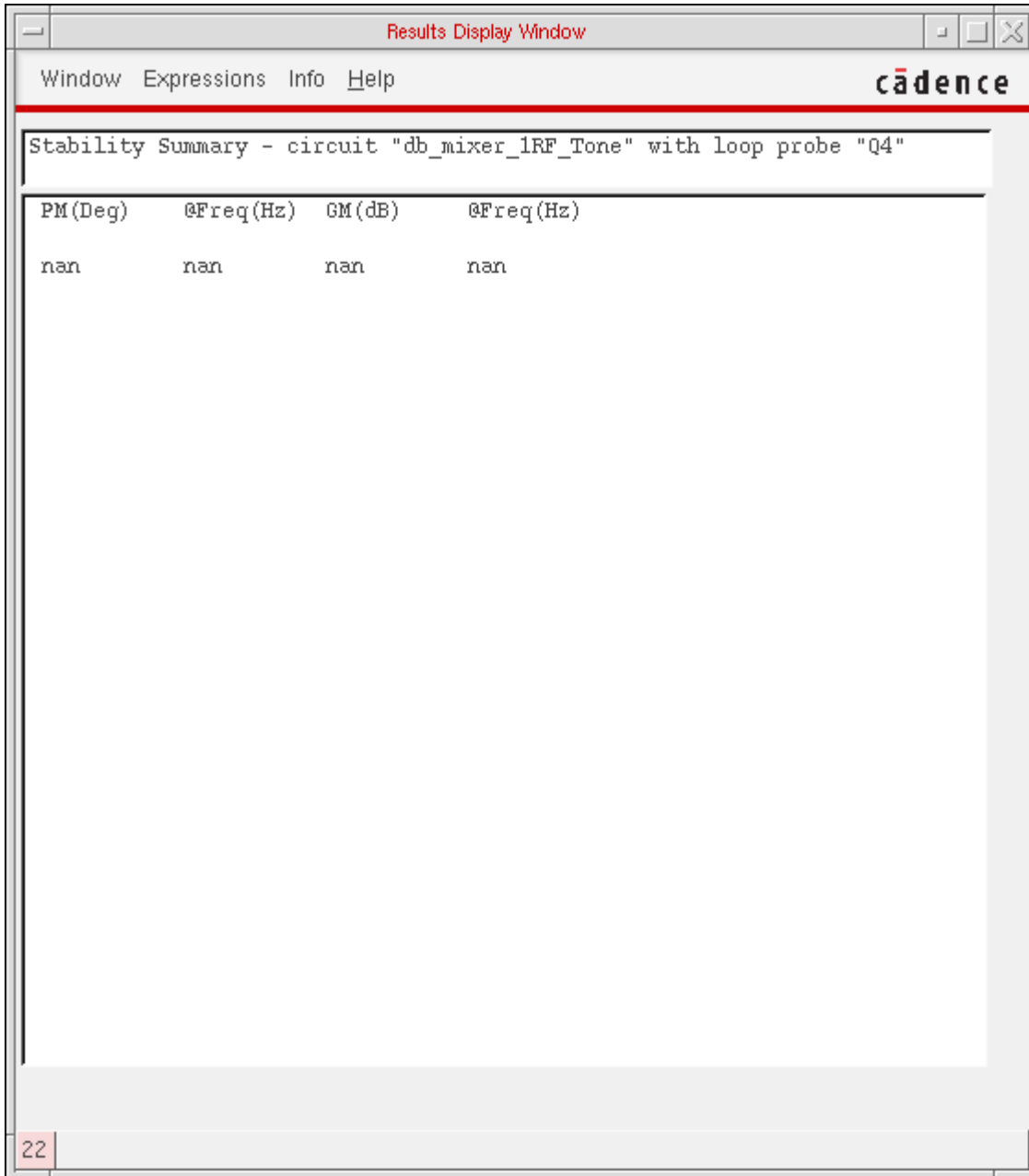
Note: This menu option will be enabled only when stability analysis was included in the last run or the results file exists in the results directory when you specifically selected an existing results directory through ADE L.

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

2. Choose the required data and click *OK*.

The *Results Display Window* displays the stability summary using the criteria that you specified. For example, if you had swept temperature and capacitor values with the

parametric tool for the stability analysis and selected the *Both* option on the form, the *Results Display Window* will appear as follows:



Printing Pole Zero Summary

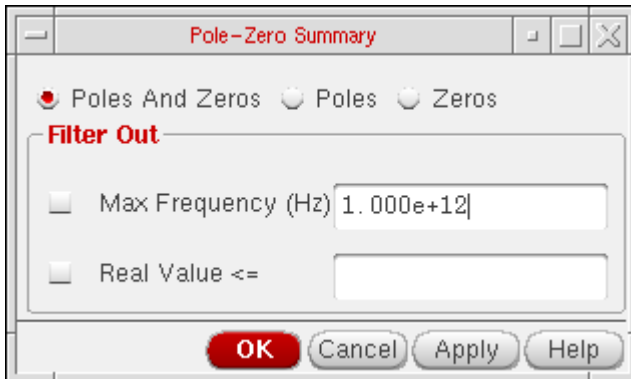
To print the *Pole Zero* summary in your circuit,

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

1. Choose *Results – Print – Pole Zero Summary*. The *Pole-Zero Summary* form appears. The form enables you to print poles or zeros, or poles and zeros with filtering options.

Note: This menu option will be enabled only when pole zero analysis was included in the last run or the results file exists in the results directory when you specifically selected an existing results directory through the Analog Design Environment.

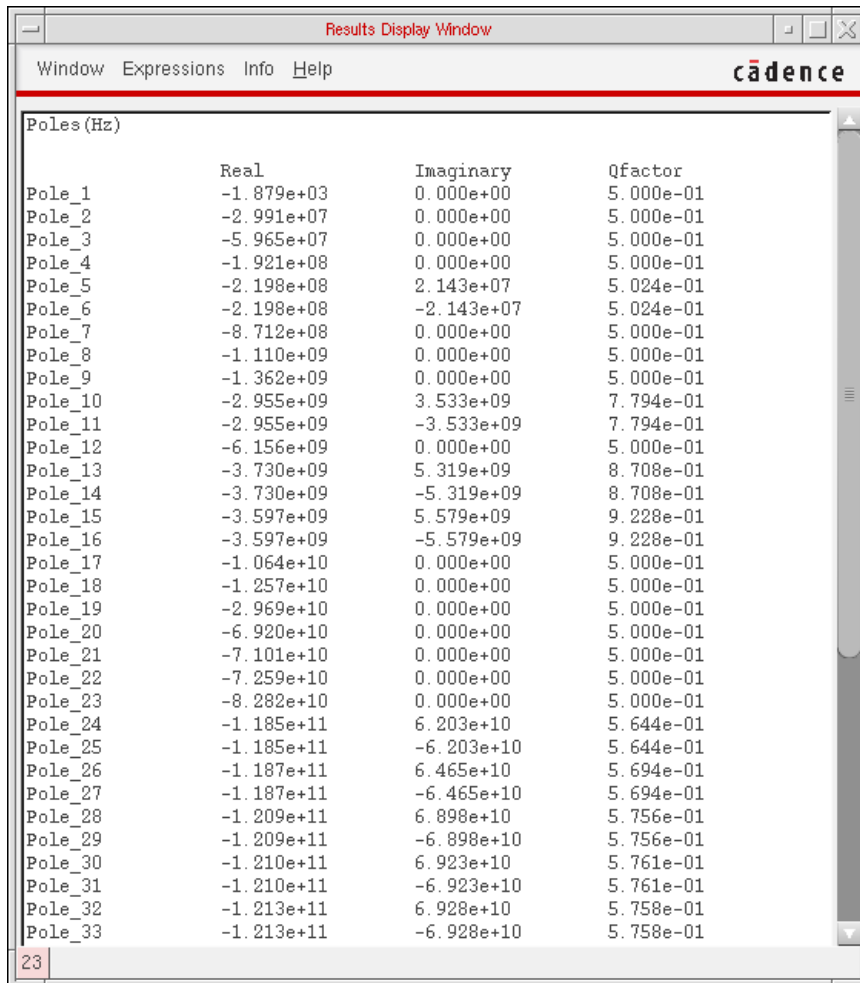


2. Select the option, *Poles* if you want to plot only poles, *Zeros* if you want to plot only zeros and *Poles and Zeros* if you want to plot both poles and zeros.
- Note:** By default, the option *Poles and Zeros* is selected.
3. Set the required options in the *Filter Out* section and click *OK*. This section provides a combination of filtering mechanisms that you can select in order to plot the poles and zeros. These are:

- Max Frequency:*
This option enables you to filter out poles and zeros that are outside the frequency band of interest (FBOI) and that do not influence the transfer function in the FBOI. The default value is that specified in the *fmax* field in the *Pole-Zero Options* form. Note, that for the *Direct Plot* form, *fmax* is read from the header of the *psf* data. Only poles and zeros whose magnitudes exceed the frequency value specified are filtered out.
- Real Value:*
This option enables you to specify the real part of the frequency. Only poles and zeros whose real values are less than or equal to the real value specified are filtered out.

Note: By default, no filtering is selected. You can set the filtering criteria once you specify either poles or zeros or both to be plotted.

4. The *Results Display Window* displays the pole zero summary using the criteria that you specified:



The screenshot shows the 'Results Display Window' with a menu bar (Window, Expressions, Info, Help) and the Cadence logo. The window title is 'Results Display Window'. The content area displays a table titled 'Poles (Hz)' with the following data:

	Real	Imaginary	Qfactor
Pole_1	-1.879e+03	0.000e+00	5.000e-01
Pole_2	-2.991e+07	0.000e+00	5.000e-01
Pole_3	-5.965e+07	0.000e+00	5.000e-01
Pole_4	-1.921e+08	0.000e+00	5.000e-01
Pole_5	-2.198e+08	2.143e+07	5.024e-01
Pole_6	-2.198e+08	-2.143e+07	5.024e-01
Pole_7	-8.712e+08	0.000e+00	5.000e-01
Pole_8	-1.110e+09	0.000e+00	5.000e-01
Pole_9	-1.362e+09	0.000e+00	5.000e-01
Pole_10	-2.955e+09	3.533e+09	7.794e-01
Pole_11	-2.955e+09	-3.533e+09	7.794e-01
Pole_12	-6.156e+09	0.000e+00	5.000e-01
Pole_13	-3.730e+09	5.319e+09	8.708e-01
Pole_14	-3.730e+09	-5.319e+09	8.708e-01
Pole_15	-3.597e+09	5.579e+09	9.228e-01
Pole_16	-3.597e+09	-5.579e+09	9.228e-01
Pole_17	-1.064e+10	0.000e+00	5.000e-01
Pole_18	-1.257e+10	0.000e+00	5.000e-01
Pole_19	-2.969e+10	0.000e+00	5.000e-01
Pole_20	-6.920e+10	0.000e+00	5.000e-01
Pole_21	-7.101e+10	0.000e+00	5.000e-01
Pole_22	-7.259e+10	0.000e+00	5.000e-01
Pole_23	-8.282e+10	0.000e+00	5.000e-01
Pole_24	-1.185e+11	6.203e+10	5.644e-01
Pole_25	-1.185e+11	-6.203e+10	5.644e-01
Pole_26	-1.187e+11	6.465e+10	5.694e-01
Pole_27	-1.187e+11	-6.465e+10	5.694e-01
Pole_28	-1.209e+11	6.898e+10	5.756e-01
Pole_29	-1.209e+11	-6.898e+10	5.756e-01
Pole_30	-1.210e+11	6.923e+10	5.761e-01
Pole_31	-1.210e+11	-6.923e+10	5.761e-01
Pole_32	-1.213e+11	6.928e+10	5.758e-01
Pole_33	-1.213e+11	-6.928e+10	5.758e-01

Printing DC Node Voltages

To print the DC node voltages of the nodes or components in your circuit,

1. Choose *Results – Print – DC Node Voltages*.
2. Move your cursor into the Schematic window.

You are prompted to select nets for the VDC output.

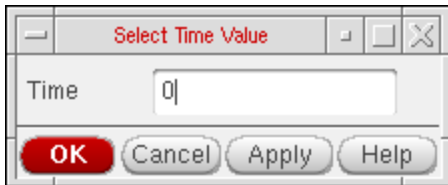
3. Click a node.

Printing Transient Voltages

To print the transient node voltages of the nodes in your circuit,

1. Choose *Results – Print – Transient Node Voltages*.

The Schematic window comes into the view and the *Select Time Value* form appears.



2. In the *Time* field, type the time value at which you want to print the transient node voltages. The default time value is 0.

3. Select nets, for which you want to view the voltage value, on the schematic.

For each selected net, the voltage value is displayed in the *Results Display Window*.

4. Press <Esc> after the selection is complete.

Printing Sensitivities

To print the sensitivities in your circuit,

1. Choose *Results – Print – Sensitivities*.

2. Move your cursor into the Schematic window.

You are prompted to select nets for the output.

3. Click a net or port.

Precision Control for Printing

Precision of printed results can be controlled using `aelPushSignifDigits`.

Example

```
aelPushSignifDigits(4)
rn          37.9322e-18   fn          0          total
37.9322e-18
```

```
aelPushSignifDigits(8)
rn          37.932238e-18  fn          0          total
37.932238e-18
```

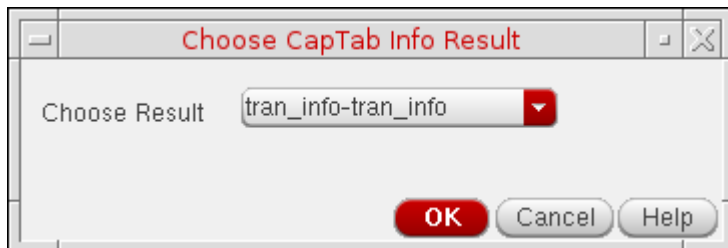
Printing Capacitance Data

When you run a transient or dc analysis with the *captab* option selected in the analysis options form, you can view the captab data in the capacitance table. For more information on the captab option and parameters, see [CAPTAB Parameters](#).

To view capacitance data:

1. Choose *Results - Print - Capacitance Table* in the ADE window.

If you had opted to save captab data for more than one analysis, the *Choose CapTab Info Result* form appears first.



Select one captab result file from the *Choose Result* cyclic box and click *OK*.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

The Capacitance Table appears as shown below:

Capacitance Table

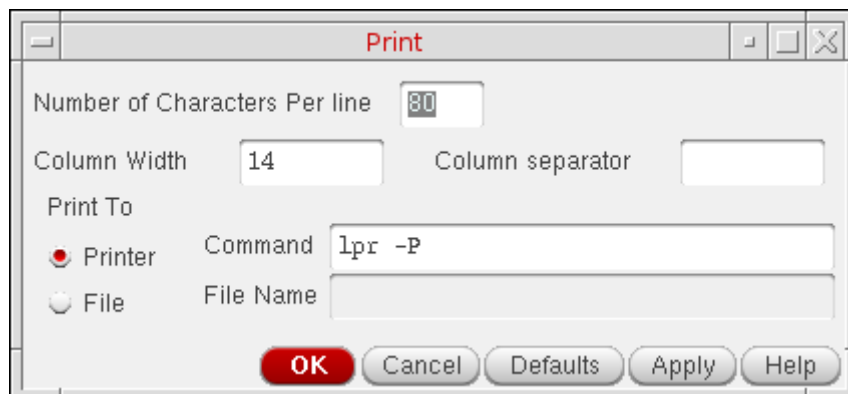
From	To	Variable	Fixed	Total
/vss!	/vss!	3.9e-14	1.237733e-12	1.276733e-12
/vin	/vin	4.864e-14	6.084377e-13	6.570777e-13
/vdd!	/vdd!	0.0	2.165635e-13	2.165635e-13
/out	/out	7.5e-13	0.0	7.5e-13
/net6	/net6	0.0	1.19046e-14	1.19046e-14
/net5	/net5	4.864e-14	6.084443e-13	6.570843e-13
/I0/net30	/I0/net30	4.864e-14	1.275839e-12	1.324479e-12
/I0/net15	/I0/net15	4.382e-14	0.0	4.382e-14
/I0/net13	/I0/net13	7.5e-13	0.0	7.5e-13
/I0/net10	/I0/net10	0.0	0.0	0.0
/I0/gnode	/I0/gnode	8.282e-14	1.237733e-12	1.320553e-12
/I0/Q4:int_e	/I0/Q4:int_e	0.0	3.683658e-12	3.683658e-12
/I0/Q4:int_c	/I0/Q4:int_c	0.0	2.532433e-14	2.532433e-14
/I0/Q4:int_b	/I0/Q4:int_b	0.0	3.703133e-12	3.703133e-12
/I0/Q3:int_e	/I0/Q3:int_e	0.0	3.54472e-12	3.54472e-12
/I0/Q3:int_c	/I0/Q3:int_c	0.0	3.330098e-14	3.330098e-14
/I0/Q3:int_b	/I0/Q3:int_b	0.0	3.571362e-12	3.571362e-12
/I0/Q2:int_e	/I0/Q2:int_e	0.0	3.422691e-12	3.422691e-12
/I0/Q2:int_c	/I0/Q2:int_c	0.0	6.019275e-14	6.019275e-14
/I0/Q2:int_b	/I0/Q2:int_b	0.0	3.473509e-12	3.473509e-12
/I0/Q1:int_e	/I0/Q1:int_e	0.0	5.293756e-12	5.293756e-12
/I0/Q1:int_c	/I0/Q1:int_c	0.0	2.071378e-11	2.071378e-11
/I0/Q1:int_b	/I0/Q1:int_b	0.0	6.125362e-12	6.125362e-12

The table shows captab data in five columns:

- From*: The nodes from which capacitance is measured.
- To*: The nodes to which capacitance is measured.
- Variable*: Variable capacitance between nodes.
- Fixed*: Fixed capacitance between nodes.
- Total*: Total capacitance between nodes.

2. Select one of more sweep values for which the capacitance table is to be printed by using the cyclic buttons in the Select sweep value group box. The values in the *Capacitance Table* change dynamically as you select the different options.
3. Click the *Print* button to save or print captab data.

The Print form appears.



4. Specify values in this form as follows:

- a. *Number of characters per Line*: Number of characters to be printed on a line.
- b. *Column Width*: Width of the columns to be printed. The default value is 14.
- c. *Column Separator*: Column separator to be used for printing the table. The default is a space.
- d. *Print To*: If you want to print to a printer, select the *Printer* option button and specify a command. If you want to print to a file, select the *File* option button and specify a filename.

5. Click *OK* or *Apply*.

Printing Statistical Reports or Calculator Results

For information about statistical reports, read the [*Virtuoso Analog Design Environment XL User Guide*](#).

For information about printing data using Virtuoso Visualization and Analysis XL Calculator, read the [*Virtuoso Visualization and Analysis XL User Guide*](#).

Using SKILL to Display Tabular Data

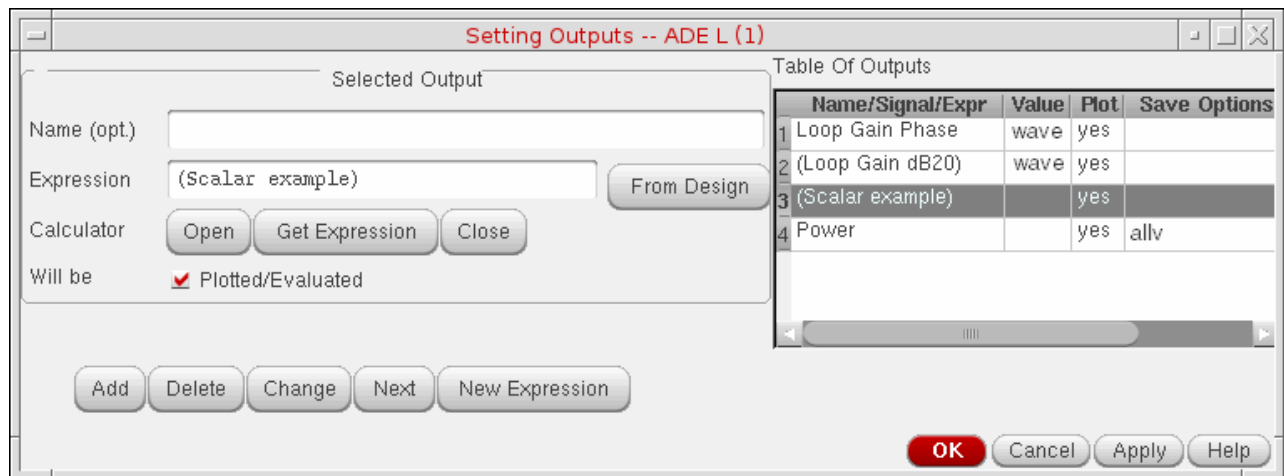
You can use the SKILL language for queries to request other kinds of simulation results, to build output format macros, and to automate test and result reporting sequences. The syntax for queries is shown at the beginning of the line in the Results Display window.

To display...	Type this command in the CIW
A list of operating-point parameter names and their values for R1	OP ("/R1" , "??")
A list of just the operating-point parameter names for R1	OP ("/R1" , "?")
A single operating-point parameter (v for voltage, for example) and its value for R1	OP ("/R1" , "v")
A list of transient operating-point parameter names and their values for C1	OPT ("/C1" , "??")
A list of just the transient operating-point parameter names for C1	OPT ("/C1" , "?")
A single transient operating-point parameter (i for current, for example) and its value for C1	OPT ("/C1" , "i")
A list of model parameter names and their values for Q1	MP ("/Q1" , "??")
A list of just the model parameter names for Q1	MP ("/Q1" , "?")
A single model parameter (is for saturation current, for example) and its value for Q1	MP ("/Q1" , "is")
Noise parameter information for a device with only one noise parameter (a resistor R4, for example)	VNP ("/R4")
A list of noise parameter names for a device with more than one noise parameter (a device D24, for example) and their values	VNPP ("/D24" , "??")
A list of just the noise parameter names for a device with more than one noise parameter (a device D24, for example)	VNPP ("/D24" , "?")
A single noise parameter (rs for saturation resistance, for example) and its value for a device with more than one noise parameter (a device D24, for example)	VNPP ("/D24" , "rs")

Overview of Plotting Calculator Expressions

You can plot calculator expressions that are waveforms and print scalar expressions.

Based on the Simulation window segment in the following illustration, output 3 is a scalar expression. The scalar expression value appears in the Simulation window and the Setting Outputs form but is not plotted or saved.



Defining Expressions

To define a new expression,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. In the Setting Outputs form, click *New Expression*.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

The Setting Outputs form redraws to display a blank *Expression* field.

	Name/Signal/Expr	Value	Plot	Save Options
1	Loop Gain Phase	wave	yes	
2	(Loop Gain dB20)	wave	yes	
3	(Scalar example)		yes	
4	Power		yes	allv

For details about the form, see [“Setting Outputs”](#) on page 564.

3. (Optional) Type a name for the expression.

If you do not type a name, the expression itself appears in the *Outputs* table on the Simulation window.

For example, the expression for the 3 dB point is

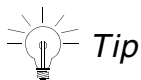
```
bandwidth(VF("/OUT), 3, "low")
```

Rather than seeing this expression in the *Outputs* table on the Simulation window, you might type the name

```
3 dB point of OUT
```

4. Enter the expression using one of the following methods:

- Type the expression in the *Expression* field.
- Use the calculator to create the expression and then retrieve it by clicking *Get Expression*.



Tip

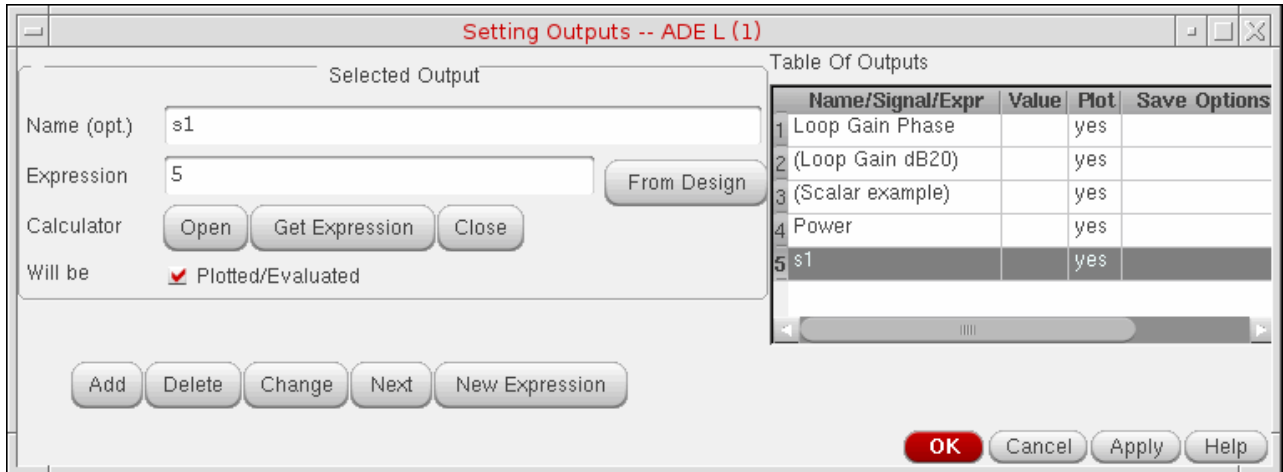
You can create expressions based on other expressions. For more information, see [Creating Dependent Expressions](#) on page 540.

5. Click *Add*.

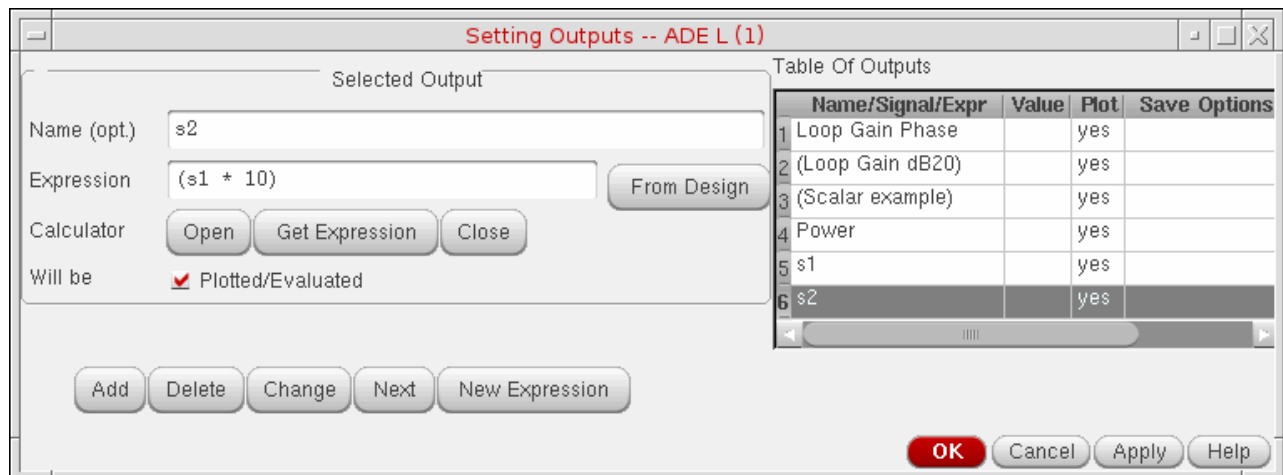
6. The expression is added to the *Table of Outputs* list box and also displayed in the *Outputs* table on the Simulation window.

Creating Dependent Expressions

You can create expressions based on other expressions. For example, assume that you have an expression named *s1* with the expression 5 as shown in the figure below.



If you want another expression, say, *s2* to be ten times the value of *s1*, type $s1 * 10$ in the *Expression* field as shown in the figure below.



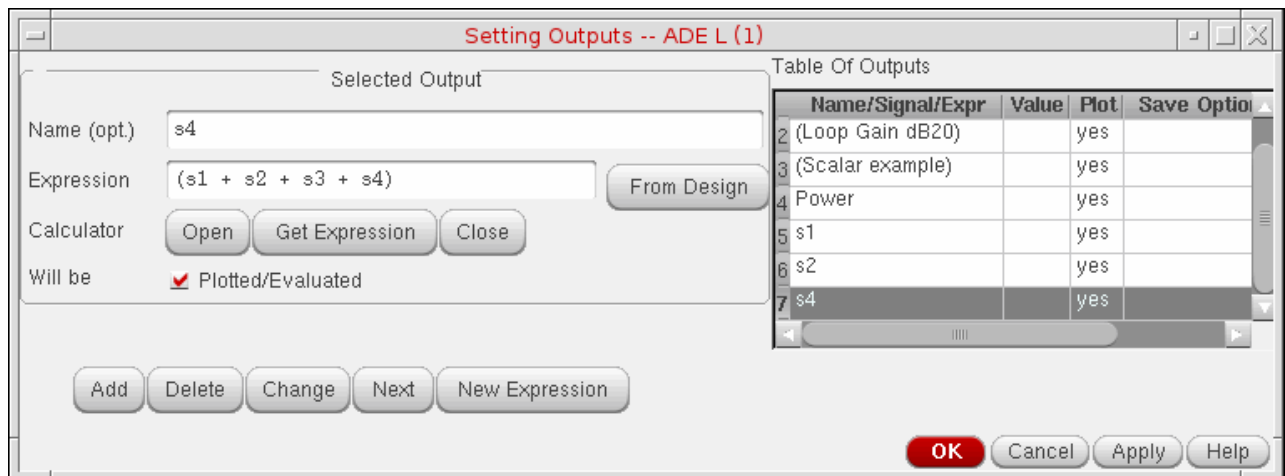
Note the following when you create dependent expressions:

- Expressions can be added in any order, irrespective of their dependencies. For example, if the expression *s2* is based on expression *s1*, it is not necessary to add the expression *s1* before adding the expression *s2*.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- An expression can be based on any number of other expressions. An example of an expression s_4 that is based on three expressions (s_1 , s_2 and s_3) is given below:

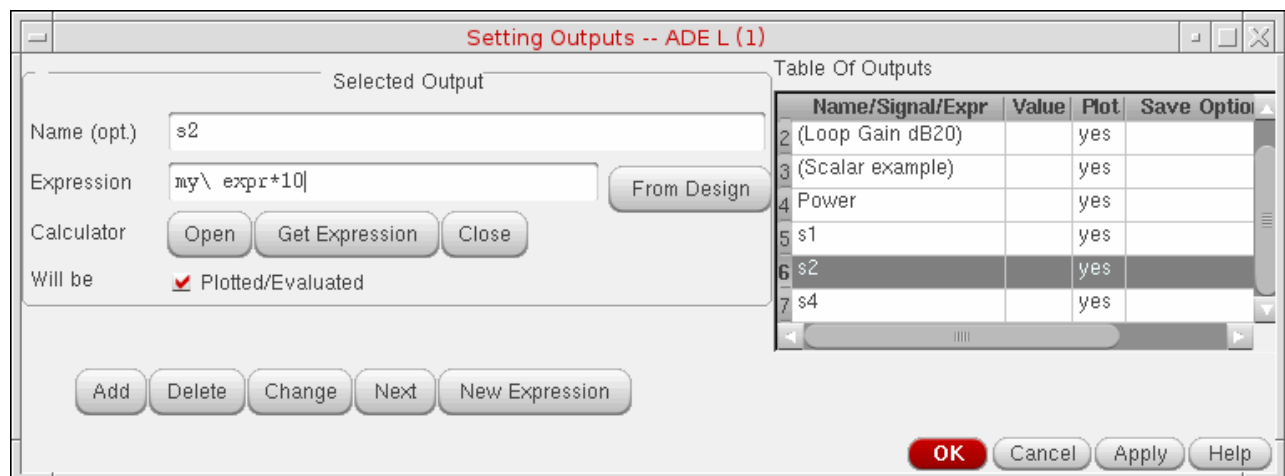


- Ensure that there is no cyclic dependency between the dependent expressions.

In the following example, a cyclic dependency exists because expression `myExpr` depends on expression `myExpr1`, and expression `myExpr1` also depends on expression `myExpr`.

```
myExpr=myExpr1*5
myExpr1=myExpr+10
```

- If the expression name that is being used in a dependent expression has special characters such as spaces or dot (`.`), use the `\` character to escape these special character. For example, if you are using an expression named `my expr` in another expression, escape the space character in `my expr` as shown in the figure below.

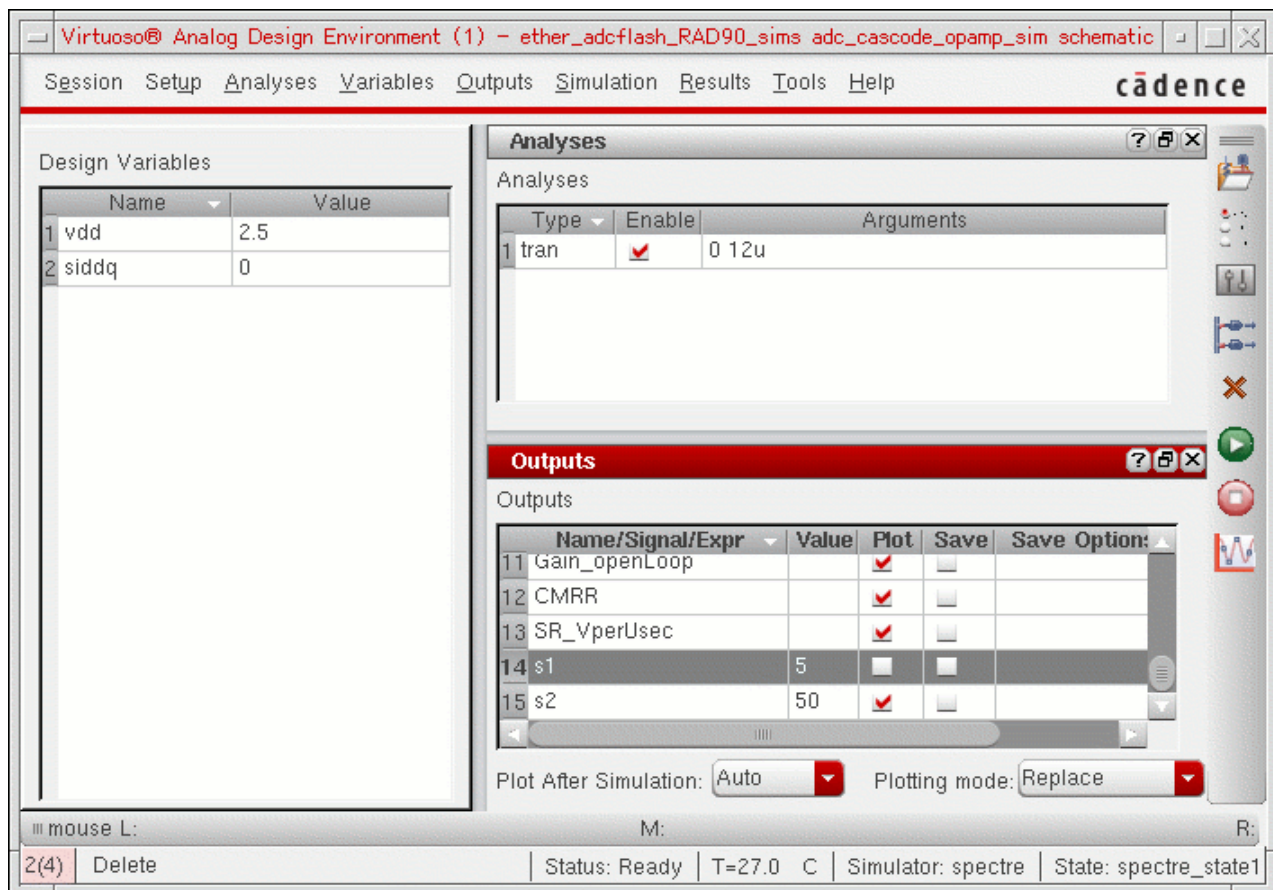


Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Note: Generally, expression names can have special characters as they are valid string values. However, when these expression names are used in dependent expressions, they are treated as SKILL symbols and should be in a valid SKILL format. Therefore, special characters in such expression names should be escaped using the \ character.

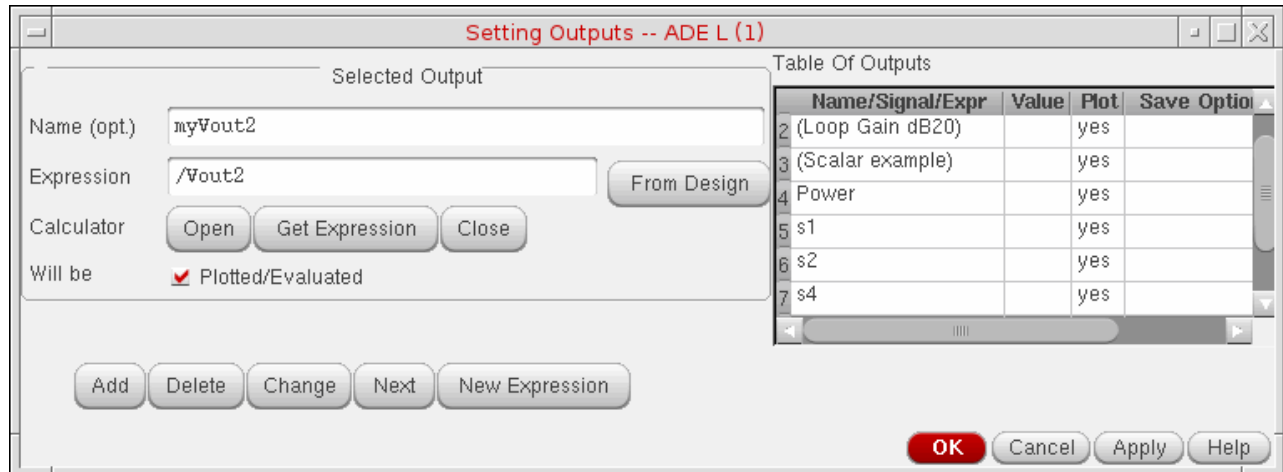
- It is not necessary to plot or save the expressions on which other expressions are based. For example, if the expression `s2` is based on expression `s1`, it is not necessary to select the *Plot* or *Save* check box for the expression `s1` in the Outputs pane on the Simulation window, as shown in the following figure.



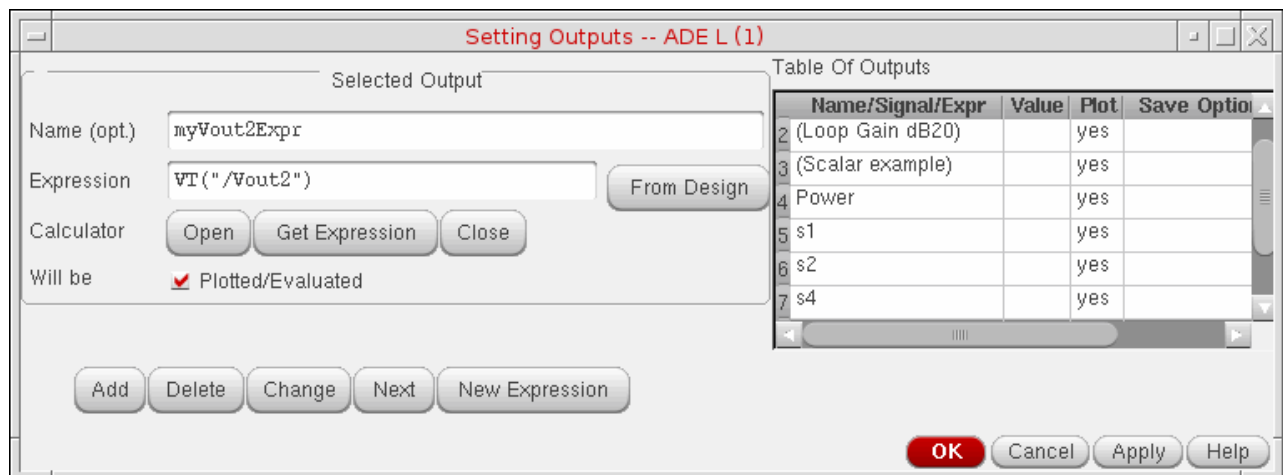
Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- You cannot assign a name to an output of type signal and then use that name in a dependent expression. For example, you cannot assign the name `myVout2` for the `Vout2` signal as shown below, and then use `myVout2` in a dependent expression.



If you want a dependent expression to be based on a signal, create an expression based on that signal and use the name of that expression in a dependent expression. For example, create an expression named `myVout2Expr` based on the signal `Vout2` as shown below, and then use `myVout2Expr` in a dependent expression.



- Error messages, if any, regarding dependent expressions are displayed in the Command Interpreter Window (CIW).

Plotting Expressions

To plot expressions,

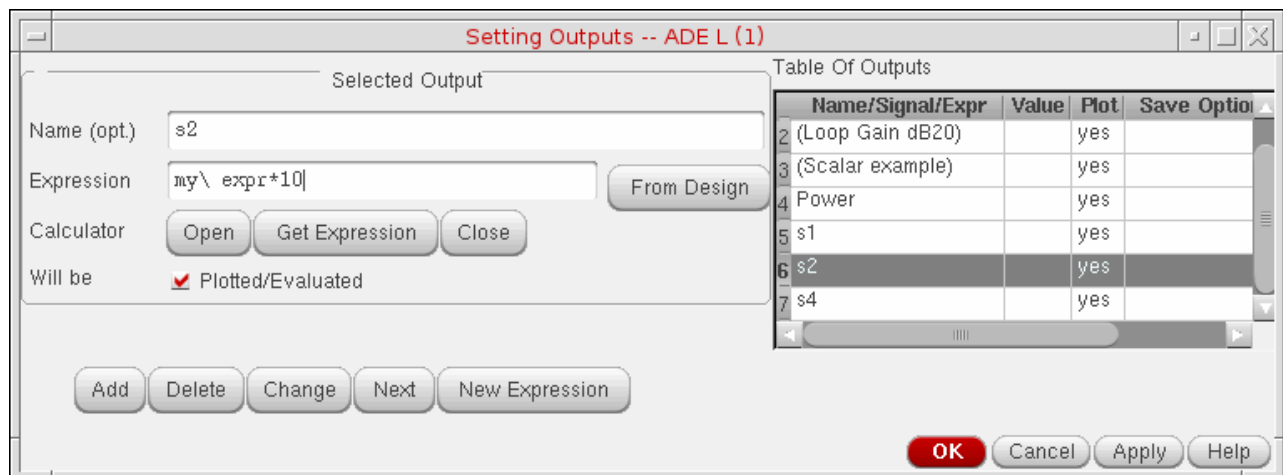
- Choose *Results – Plot Outputs – Expressions*.

The system plots waveform expressions in the graph window and prints the value of scalar expressions in the *Outputs* area of the Simulation window.

Suppressing Plotting of an Expression

To suppress plotting of an expression without deleting it,

1. In the Simulation window, choose *Outputs – Setup*, or in the Schematic window, choose *Setup – Outputs*.
2. Click the expression in the *Table Of Outputs* list box.
3. Deselect *Will Be Plotted/Evaluated*, as shown in the figure.



4. Click *Change*.

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

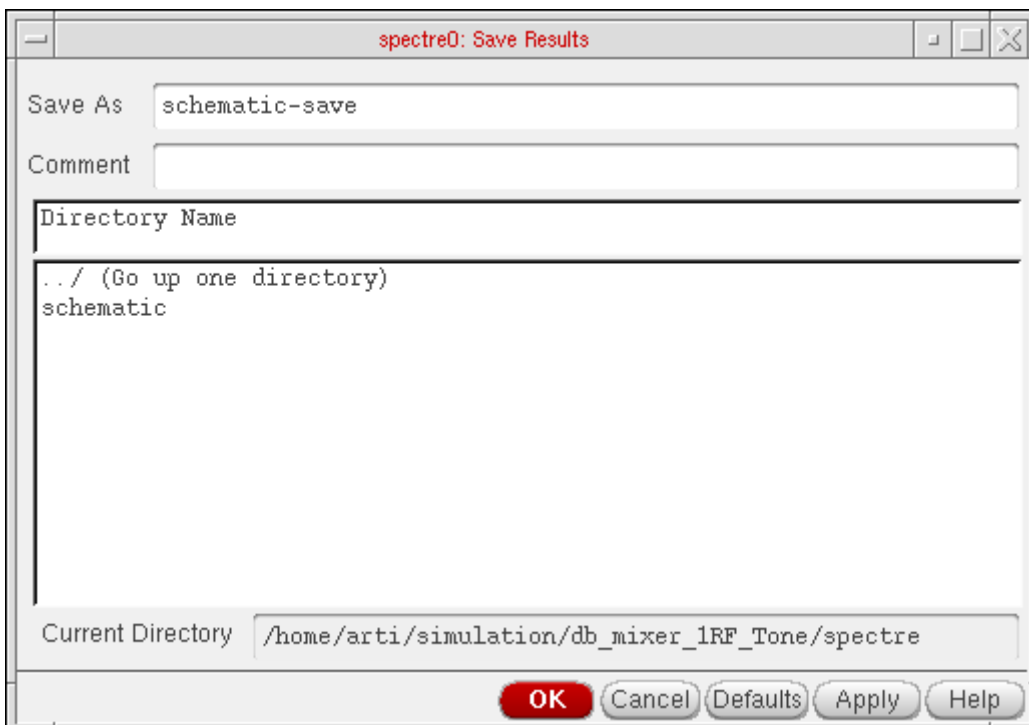
Viewing and saving Results

Saving Simulation Results

The default name under which results are saved is `schematic-save`. To save a results directory under a different name,

1. Choose *Results – Save* in the Simulation window or the Schematic window.

The Save Results form shows the results of the latest simulation in the *Save As* field.



For detailed information about the form, see [“Save Results”](#) on page 566.

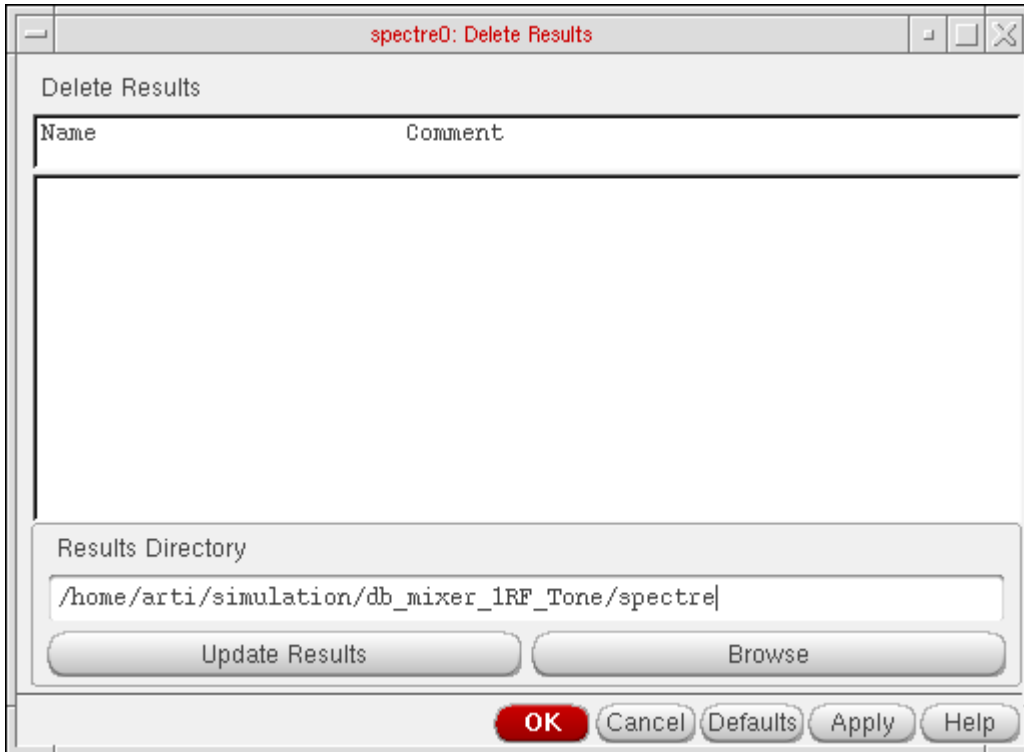
2. Type a new name for the results directory in the *Save As* field.
3. (Optional) Type a short description in the *Comment* field to help you identify these results when you restore the results later.

Deleting Simulation Results

To delete a set of simulation results,

1. Choose *Results – Delete* in the Simulation window or the Schematic window.

The Delete Results form appears.



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

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

If the results you want to delete are in a different location, click the Browse button to open the Unix Browser form.

Browsing Results Directories

To browse directories,

- Click the *Browse* button.



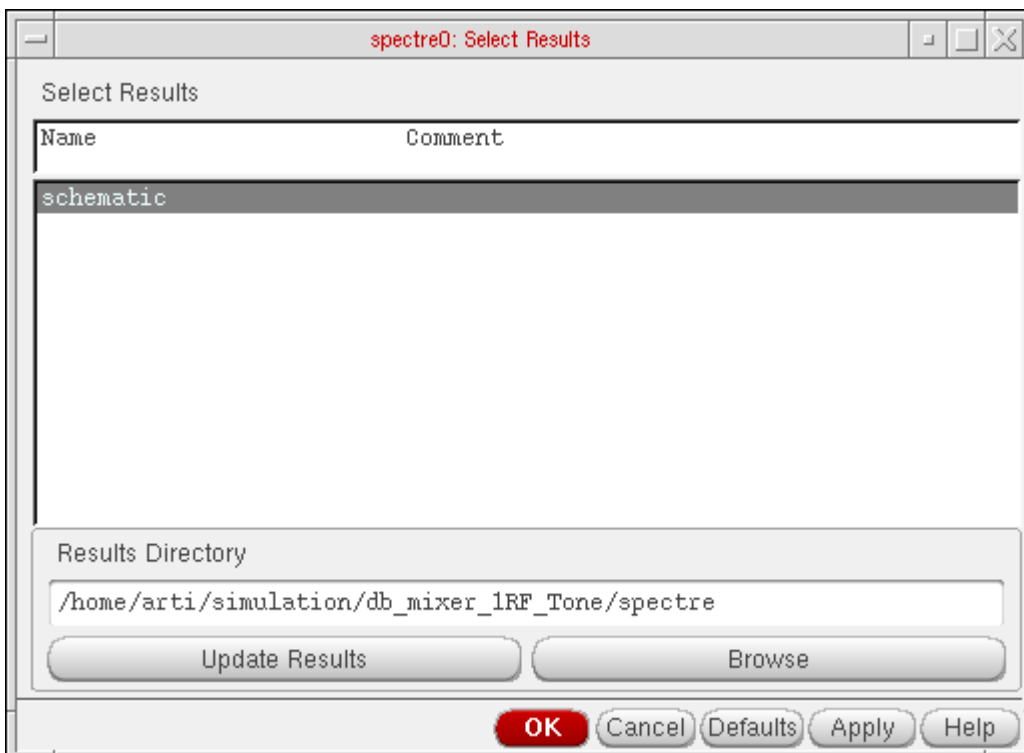
For detailed information about the form, see “[UNIX Browser](#)” on page 567.

Restoring Saved Results

To select and restore a set of simulation results for the current design,

1. Choose *Results – Select* in the Simulation window or the Schematic window.

The Select Results form appears.



The *Results Directory* is the directory in which the simulation results for the selected simulation are saved.

2. Check that the *Results Directory* field displays the correct information.

You can select results in a different location by clicking the Browse button and navigating to the proper directory.

Note: The proper directory is two levels up from the `psf` directory. For example, if your results directory is: `simulation/ampTest/spectre/schematic/psf`, use the browser to select `simulation/ampTest/spectre`.

3. Double-click the results you want.
4. Click *OK*.

Note: If you restore parasitic simulation results, be sure that parasitic simulation is enabled from the LVS form.

Annotating Simulation Results

You can annotate data onto the schematic to show the parameters, operating points, net names, currents and voltages of the design components. You can also change the existing annotations using the context menu or the annotation setup form.

Before you can annotate the schematic, do one of the following:

- Run a simulation.
- Select results.

To select results, choose *Results – Select* in the Simulation window, select the current data file, and click *OK*.

To annotate data on the schematic,

- ▶ In the Simulation window, choose *Results* and one of the annotation commands.

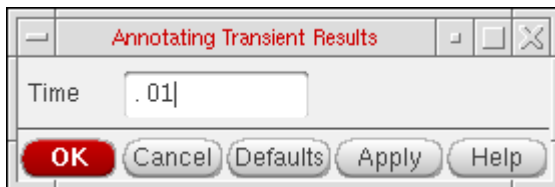
To annotate instances selectively, use the *View – Annotations – Setup* command in the Schematic window. For more information on annotating data on the Schematic, see [Annotating Data on the Schematic Window](#) section in the *Virtuoso Schematic Editor L User Guide*.

Annotating Transient Voltages

To annotate transient voltages,

1. In the Schematic window, choose *View – Annotations – Transient Voltages*.

The Annotating Transient Results form appears.



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

Alternatively, you can also type the transient time points in the *Sim Time* text box on the Schematic toolbar.

Annotating Transient Currents

To annotate transient currents:

1. In the Schematic window, choose *Results – Annotate – Transient Currents*.

The Annotating Transient Results form appears.

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

Alternatively, you can also type the transient time points in the *Sim Time* text box on the Schematic toolbar.

Annotating Transient Operating Points

To annotate final transient operating points,

1. In the Simulation window or the Schematic window, choose *Results – Annotate – Transient Operating Points*. This will annotate the operating point data for the final timepoints.

To annotate infotimes transient operating points,

1. In the Simulation window or the Schematic window, choose *Results – Annotate – Transient Operating Points*.

The *Annotating Transient Operating Points* form appears.

2. Select the transient time point in the *Time* drop-down field. This field lists the choices of timepoints at which the operating point data is stored. This will annotate the operating point data for the selected timepoint saved.
3. Click *OK* or *Apply*. These two buttons essentially perform the same operation except that the *Apply* button does not close the form enabling the user to select another timepoint and click *Apply* again to annotate data for a different timepoint. Clicking on the *Cancel* button will cancel entire operation.

Important Points to Note:

- The options to annotate are enabled depending on the availability of the results. For example, if you do not have the results for transient simulation, the options to plot and annotate the transient voltages, transient currents, and transient operating points will be disabled.
- This form will not come up if the user has not stored operating point data at different timepoints.

- *The Results – Annotate – Show Parasitics and Results – Annotate – Hide Parasitics* are enabled only when the results are available for *dcop*.

Specifying the Data Directory for Labels

To specify the simulation data directory (run directory) for labels,

1. In the Schematic window, choose *View – Annotations – Setup*.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

The Annotation Setup form appears.

Label	Display Mode	Display Type	Expression	Annotate	Balloon
Terminal:cdsTerm(B)	Voltage	DC		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Terminal:cdsTerm(D)	Voltage	DC		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Terminal:cdsTerm(G)	Voltage	DC		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Terminal:cdsTerm(S)	Voltage	DC		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(1)	Component Parameter	Literal	<input checked="" type="checkbox"/> model	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(2)	Operating Point	DC	<input checked="" type="checkbox"/> vgs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(3)	Operating Point	DC	<input checked="" type="checkbox"/> vds+vth	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(4)	Component Parameter	Literal	<input checked="" type="checkbox"/> Vgs	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(5)	Operating Point	DC	<input checked="" type="checkbox"/> age	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Parameter:cdsParam(6)	Model Parameter		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Name:cdsName()	Instance Name	Schematic		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

- In the *Simulation Data Directory* field, type the path to the simulation run directory and click *OK*.

Note: You do not need to use this form if

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

- ❑ You have the analog circuit design environment active and specified the correct directory as the run directory
- ❑ You used the Results Browser to select results for the current schematic
- ❑ The most recent simulation you ran was of this schematic

Saving and Removing Annotated Labels

To save the annotation settings, choose *File – Save* from the Annotation Setup form. For more information on saving the annotation settings, see [Saving Annotation Settings](#) section in the *Virtuoso Schematic Editor L User Guide*.

Annotating Parametric Sweep Results

You can annotate parametric sweep results to the schematic using the options in the *Results – Annotate* menu. For more information about running parametric analyses, see [Running the Parametric Analysis](#). For more information about annotating simulation results, see [Annotating Simulation Results](#) on page 549.

By default, the first result point is annotated on the schematic. To view additional result points, do the following:

1. Type the following SKILL command on the input line of the Command Interpreter Window (CIW):

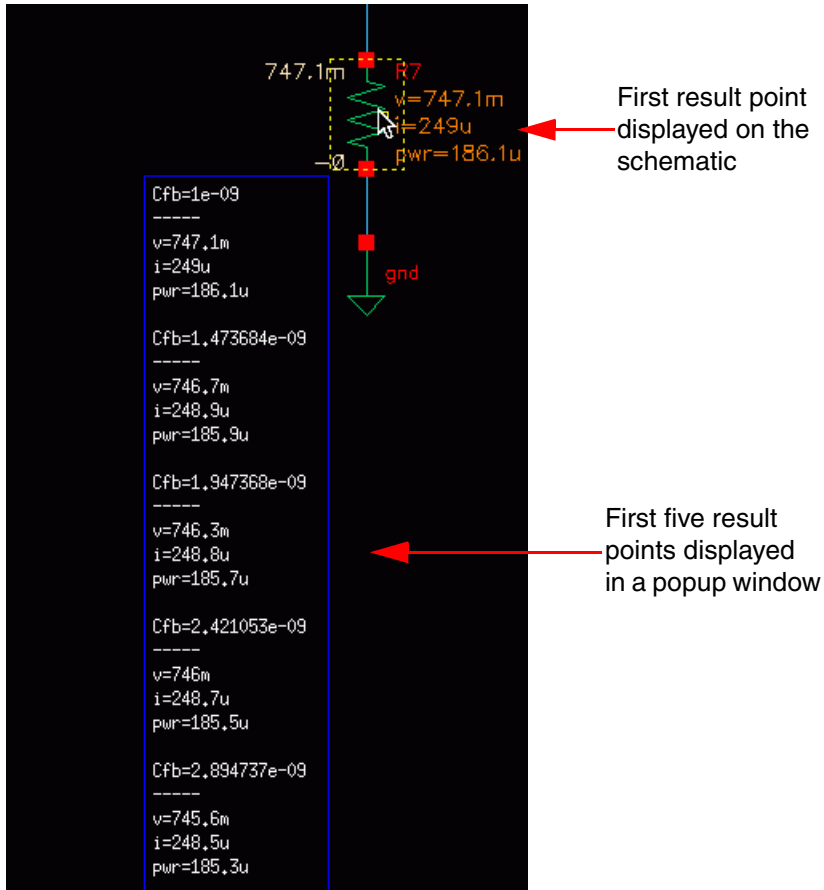
```
artEnableAnnotationBalloon(t)
```

2. Hover the mouse pointer over an instance on the schematic. A pop-up window appears displaying the first 5 result points, as in the following figure.

Note: The pop-up window displays only the parameters corresponding to the `cdsParam(1)`, `cdsParam(2)` and `cdsParam(3)` labels on the symbol view for the cell.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



The pop-up window disappears after some time, or if you move the mouse pointer away from the instance.

Note the following:

- To view a specific result point, say point 15, type the following command on the input line of the CIW, then hover the mouse pointer over an instance on the schematic to view the result point in a pop-up window:

```
artEnableAnnotationBalloon(t 15 15)
```

- To view a specific range of result points, say points 10 to 15, type the following command on the input line of the CIW, then hover the mouse pointer over an instance on the schematic to view the result points in a pop-up window:

```
artEnableAnnotationBalloon(t 10 15)
```

- To reset the pop-up window to display the first five result points (the default behavior), type the following commands on the input line of the CIW, then hover the mouse pointer over an instance on the schematic to view the result points in a pop-up window:

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

```
artEnableAnnotationBalloon(nil)
artEnableAnnotationBalloon(t)
```

- To disable the display of result points in the pop-up window, type the following command on the input line of the CIW:

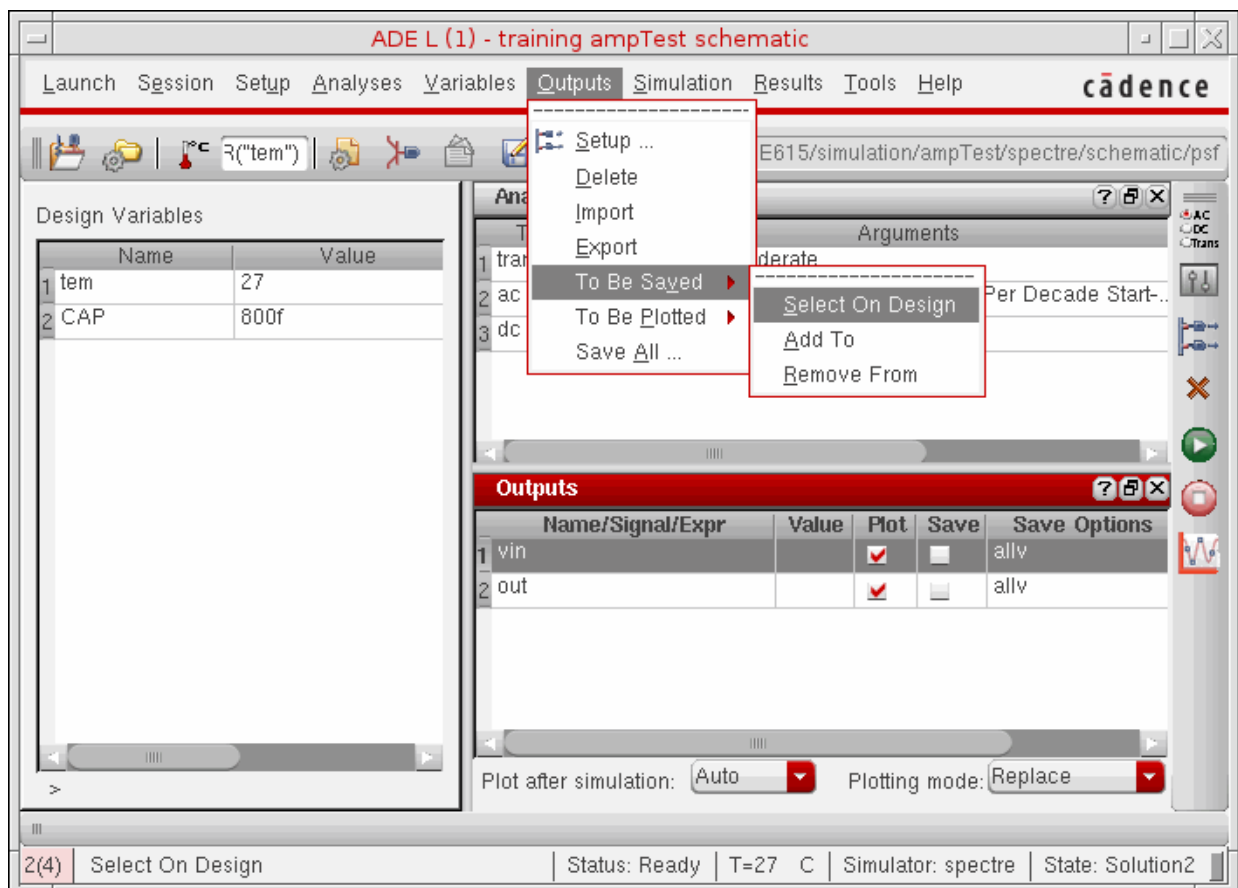
```
artEnableAnnotationBalloon(nil)
```

Plotting Results of a Parametric Analysis

This section presumes that you already ran a parametric analysis with an AC analysis selected. It shows you how to plot parametric analysis results for this AC analysis in the graph window.

To display the results of a parametric analysis,

1. Choose *Outputs – To Be Plotted – Select On Design* in the Simulation window.



Virtuoso Analog Design Environment L User Guide

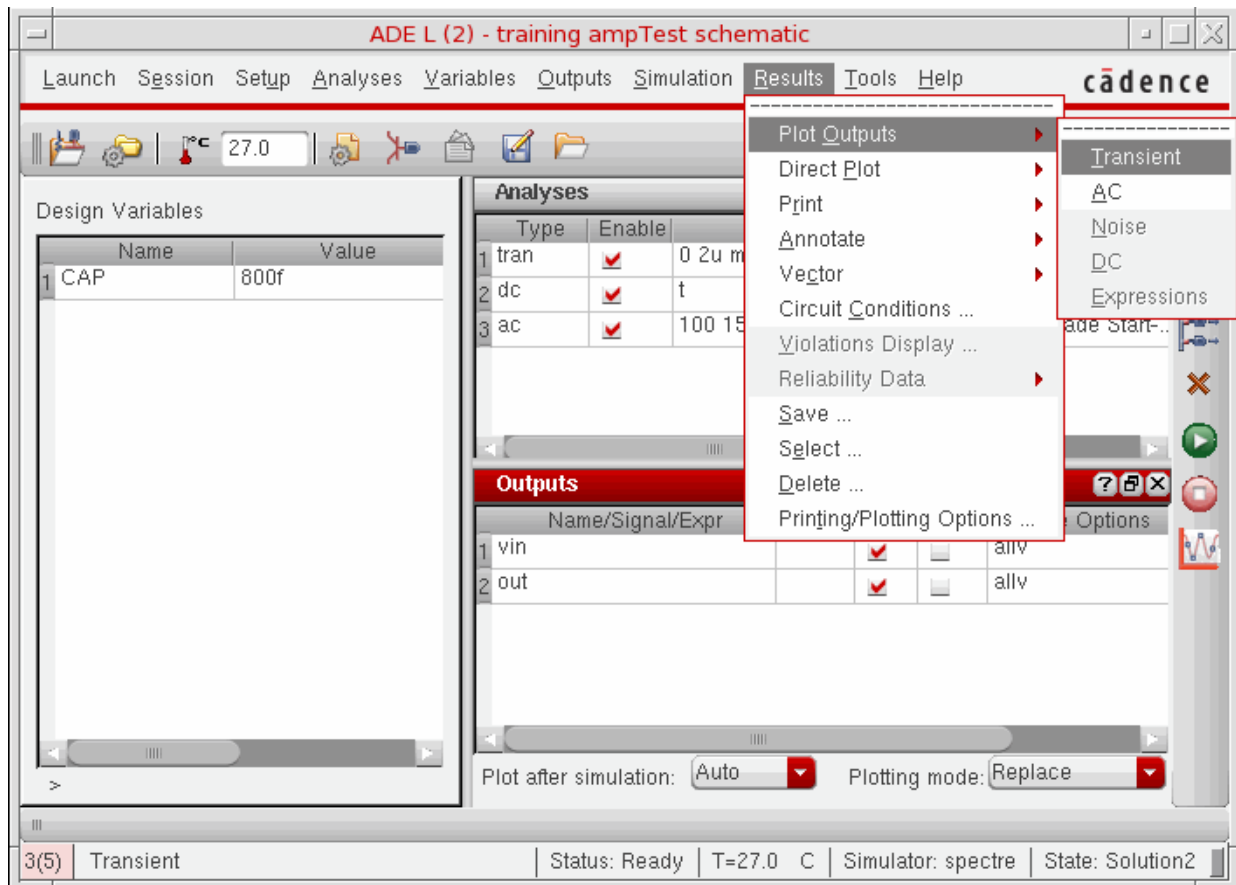
Plotting and Printing

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

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

For more information about selecting values in the schematic, see “[Overview of Plotting](#)” on page 477.

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

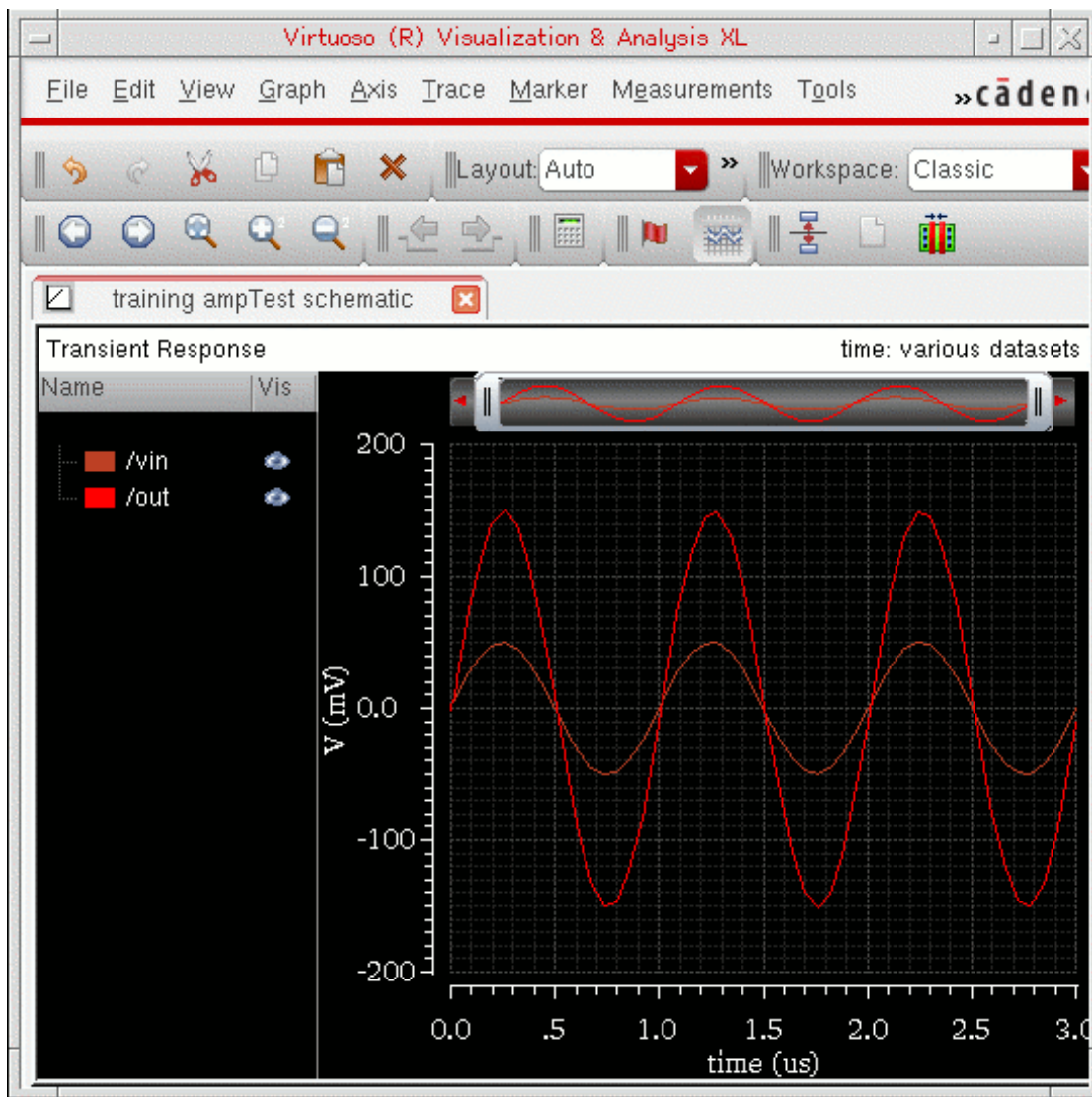


Note: You can choose other plot outputs to plot different types of analysis results.

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

Virtuoso Analog Design Environment L User Guide

Plotting and Printing



You can identify the sweep variable value associated with any curve by selecting the curve. The associated curve name gets highlighted on the graph window.

Form Field Descriptions

Setting Plotting Options

Print After refers to the commands located in the *Plot Outputs* menu.

Each Selection specifies that the plot is printed after each node is selected.

All Selections Are Made specifies that none of the plots are printed until all of the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are printed at the same time (into a table).

Direct Plots Done After refers to the commands located in the *Direct Plot* menu.

Each Selection specifies that the plot is drawn after each node is selected.

All Selections Are Made specifies that none of the plots are drawn until all of the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are printed at the same time (into a table).

For the calculator `print` and `printvs` functions, you can use append mode and have more than one expression in the buffer and use `print` or `printvs` to print into a table.

Annotations selects information to be displayed in the graph window.

Design Name displays the design name in the graph window.

Simulation Date displays the simulation run date in the graph window.

Temperature displays the temperature associated with the plotted results in the graph window.

Design Variables displays the names and values of user-created variables in the graph window.

Scalar Outputs displays simulation results that evaluate to scalar values in the graph window.

Fast Viewing Support Enables the plotting of psf output in the fast waveform viewing format in Virtuoso Visualization and Analysis XL. Using this format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

To configure Spectre to create psf output in the fast waveform viewing format, do the following:

a. Choose *Outputs – Save All*.

The Save Options form appears.

b. Select the Output Format as `psf` or `psf with floats`.

c. Select the *Use Fast Viewing Extensions* check box.

d. Click *OK*.

XF Results

Plotting Mode

Append adds the new plot to existing plots that are already displayed in the graph window.

Replace replaces existing plots with the new plot.

New SubWin adds the plot to a new subwindow.

New Win adds the plot to a new window.

Function

Voltage Gain is a calculation of voltage over voltage.

Transimpedance is a calculation of voltage over current.

Current Gain is a calculation of current over current.

Transconductance is a calculation of current over voltage.

Modifier

Magnitude (the default setting) plots the magnitude of the selected signal.

Phase plots the phase of the selected signal.

dB20 plots the magnitude in dB20.

Real plots the real component of the signal.

Imaginary plots the imaginary component of the signal.

Replot triggers the plotting of the selected instance or instance terminal with modified specifications.

Add To Outputs followed by **Replot** adds the output to the *Table Of Outputs* list box in the Simulation window.

Select instance on schematic or **Select instance terminal on schematic** prompts you to select the appropriate instance or terminal from the schematic.

S-Parameter Results

Plotting Mode

Append adds the new plot to existing plots that are already displayed in the graph window.

Replace replaces existing plots with the new plot.

New SubWin adds the plot to a new subwindow.

New Win adds the plot to a new window.

Function specifies the S-parameter or noise-parameter function to plot.

SP is S-parameters.

ZP is Z-parameters.

YP is Y-parameters.

HP is H-parameters.

GD is group delay.

VSWR is voltage standing wave ratio.

NFmin is minimum noise figure.

Gmin is the source reflection coefficient corresponding to NFmin.

Rn is equivalent noise resistance.

NF is noise figure.

B1f is the intermediate term for Kf, the Rollet stability factor.

Kf is the Rollet stability factor.

GT is transducer gain.

GA is available gain.

GP is power gain.

NC is noise circles.

GAC is available gain circles.

GPC is power gain circles.

LSB is load stability circles.

SSB is source stability circles.

Plot Type specifies the plot format. Option availability is a function of the selected function.

Auto uses the format in the current graph window unless that format is unsuitable for the function.

Rectangular specifies curves plotted against frequency.

Z-Smith specifies curves plotted on a Smith chart with impedance overlay.

Y-Smith specifies curves plotted on a Smith chart with admittance overlay.

Polar specifies curves plotted in polar (mag/angle) coordinates.

Modifier, which is used only for rectangular plots, specifies the modifier the analog circuit design environment uses to reduce complex data for two-dimensional presentation. Option availability depends on the selected function; some functions, such as stability factor, do not require a modifier.

Magnitude plots the magnitude of complex or scalar quantities.

Phase plots the phase of complex quantities in degrees.

dB20 plots the magnitude in dB.

Real plots the real part of complex quantities.

Imaginary plots the imaginary part of complex quantities.

Sweep selects a set of circles to be plotted against frequency or dB. (Sweep appears on the form only when you are plotting circles and have selected the NC, GAC, or GPC function.)

You can plot noise and gain circles at a single dB value for a range of frequencies or at a single frequency for a range of dB values.

When plotting stability circles, you can specify a frequency range. Use SSB to plot stability circles at the input port, and use LSB to plot those at the output port. You can specify a limited frequency range for these contours.

Level (dB) specifies the gain or noise figure value in dB for circles plotted against frequency.

Frequency Range defines *Start*, *Stop*, and *Step* for circles plotted at the specified dB value.

If you do not type in values for the frequency range, a circle is plotted for every simulated frequency for which a circle with the specified value exists.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Frequency specifies the spot frequency for circles plotted against a design variable.

Level Range defines *Start*, *Stop*, and *Step* for circles plotted for the specified spot frequency.

Gain is the value of gain in dB for which gain circles are plotted.

Noise is the value of noise figure in dB for which noise circles are plotted.

Plot buttons and cyclic fields at the bottom of the form generate the plots. For *S*, *Y*, *Z*, or *H* parameters, generate plots for ports 1 through 3 by clicking the appropriate button at the bottom of the form. To generate plots for the other ports, use the cyclic fields beside the buttons to specify the output and incident ports, and then click the *S*, *Y*, *Z*, or *H* button to generate the plot.

Setting Outputs

Name (opt.) is an optional name for the signal, which appears in the *Table Of Outputs* list box and in the graph window.

Expression is the calculator expression to plot or save.

Calculator buttons are displayed only while no net or terminal is selected in the *Table Of Outputs* list box.

Open opens the calculator.

Get Expression copies the expression in the calculator buffer into the expression field.

Close dismisses the calculator window.

Will Be changes depending on whether an expression or a signal is selected.

Plotted/Evaluated plots or prints the value of the expression after each simulation.

Add creates the output you set up in the *Selected Output* area.

Delete removes the highlighted output. Click in the *Table Of Outputs* list box to highlight an output.

Change updates the highlighted output with the new settings in the *Selected Output* area.

Next moves the highlight to the next signal or expression in the *Table Of Outputs* list box. This allows you to make changes to consecutive entries in the *Table Of Outputs* list box without clicking on each entry.

New Expression clears the *Selected Output* area so you can type in a new output.

Noise Summary

Type is the method of computing the noise.

spot noise produces a noise summary at a given frequency.

integrated noise produces a noise summary integrated over a frequency range using the specified weighting.

noise unit determines the units used for this summary.

Frequency Spot (Hz) is the frequency at which spot noise is calculated. The default frequency is 1K.

From (Hz) is the lower limit of the integrated noise range.

To (Hz) is the upper limit of the integrated noise range.

weighting determines if integrated noise from one frequency needs to be considered more critical than from another frequency.

flat integrates noise uniformly throughout the frequency range.

from weight file integrates noise proportionately based on the weighting functions specified in the file identified in the field.

FILTER provides a method of limiting the summary report to include only some of the device types. In the list box, you can select those devices that you want included in the report.

hierarchy level lets you set the hierarchy level, up to which the noise summary needs to be computed.

Include All Types automatically selects and highlights all device types named in the list box. Click an entry to remove the highlighting and to leave it out of the summary.

Include None automatically deselects all device types named in the list box. Highlighting is removed from all items in the list box.

include instances lists devices to be included in the noise summary.

Select lets you select devices to include from the list box.

Clear removes all instances from the *include instances* list.

exclude instances lists devices to exclude from the noise summary.

Select lets you select devices to exclude from the list box.

Clear removes all instances from the *exclude instances* list.

TRUNCATE AND SORT

truncate limits the number of instances included in the summary based on their noise contribution.

none includes all instances that were not excluded with the *exclude instances* list.

by number limits the summary to the number of the largest contributors specified in *top*.

by rel. threshold limits the summary to devices and noise contributors that contribute more than the percentage of the total noise specified in *noise %*.

by abs. threshold limits the summary to any devices or noise contributors that contribute more than the amount specified in *noise value*.

sort by determines the order of the report.

noise contributors sorts the report from the largest noise contributor to the smallest.

composite noise sorts the report by the total noise contribution of each device. Each device entry contains the percentage of the noise contribution from this device and the noise contribution from each of its contributors.

device name produces the same format as *composite noise* but sorts it in alphabetical order by device instance name.

Save Results

Save As lists the name of the directory that contains your results. The default is `schematic-save`.

Comment (optional) lets you type comments so that you can more easily differentiate simulation results.

Current Directory lists the current directory. This field cannot be edited. You use the list box above this field to navigate through directories.

Select Results

Results Directory is the directory in which the simulation results for the selected simulation are saved.

Delete Results

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

UNIX Browser

File lists the selected file or directory.

Current Directory lists the directory being viewed in the list box.

Virtuoso Analog Design Environment L User Guide

Plotting and Printing

Hspice Direct Support

Introduction to Hspice Direct Simulator

The *Analog Design Environment* (ADE) contains a direct integration of the *Hspice* simulator. ADE's *Hspice* integration (*HspiceS*) used to be based on the socket methodology, which required edit permissions to the schematic being simulated, and caused usage issues such as subcircuit name mapping. These restrictions have been lifted with the direct interface.

There are several advantages of the direct simulator integration approach over the socket simulator integration approach, namely:

■ Improved Performance in Netlisting

Netlisting is much faster in the direct approach, the direct approach supports incremental netlisting. This ensures enhanced performance when incremental updates are performed in a design and then netlisted.

■ Better Readability of Netlists

The netlists are truly hierarchical and all numeric values in the netlist are more readable. For example in *Hspice* socket, the numeric values are changed from -5.0 to -5.0000000 in the final netlist. Also, the sub-circuits are no longer unfolded. The sub-circuits are also no longer mapped unless necessary.

You can set the following SKILL variables in your `.cdsenv` file to customize the format of a netlist file:

- ❑ `hspiceSoftLineLength` variable to control the maximum length of a line of netlist output after which line is automatically split into multiple lines for easy reading. You can set this variable to an integer value less than or equal to 1024 characters. This variable overrides the OSS variable, `softLineLength`, which is set to 1024 by default.
- ❑ `hspiceMaxLineLength` variable to increase or decrease the maximum limit on the number of characters to be printed in a line of netlist output from the default. This variable overrides the OSS variable, `maxLineLength`. Note that the

`hspiceSoftLineLength` value can never be greater than `hspiceMaxLineLength` value.

For more information, see the *Customizing the Hierarchical Netlister (HNL)* chapter of the *Open Simulation System Reference*.

You can use HSPICE reserved words, `temper` and `hertz`, as design variable names in the netlist without OSS remapping them. For more information, see the *hnlReservedNameList* section of the *Open Simulation System Reference*.

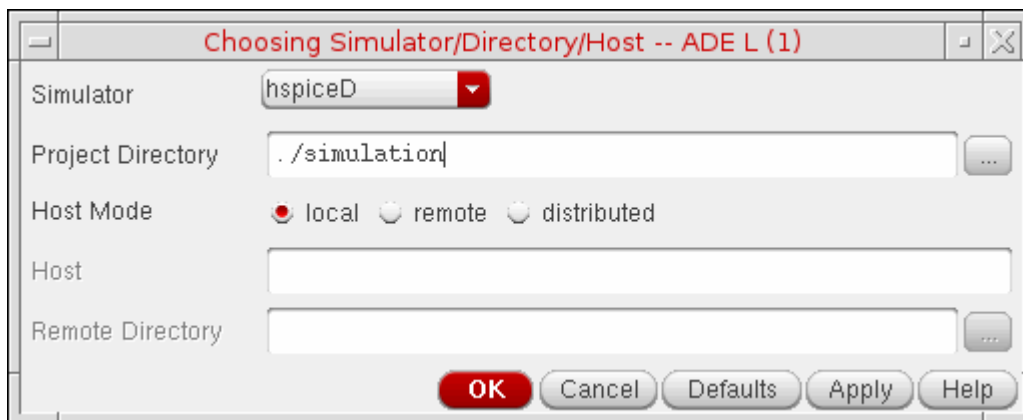
■ **Read-only Designs can be Simulated, Provided they are Extracted**

A limitation of socket netlisting is that the top cell of a design needs to be editable before the design can be netlisted. The direct approach however, allows read only designs to be simulated. The only pre-requisite being, that the design needs to be extracted first, so that connectivity information is written to the database.

■ **Advanced Evaluation of Operators**

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

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

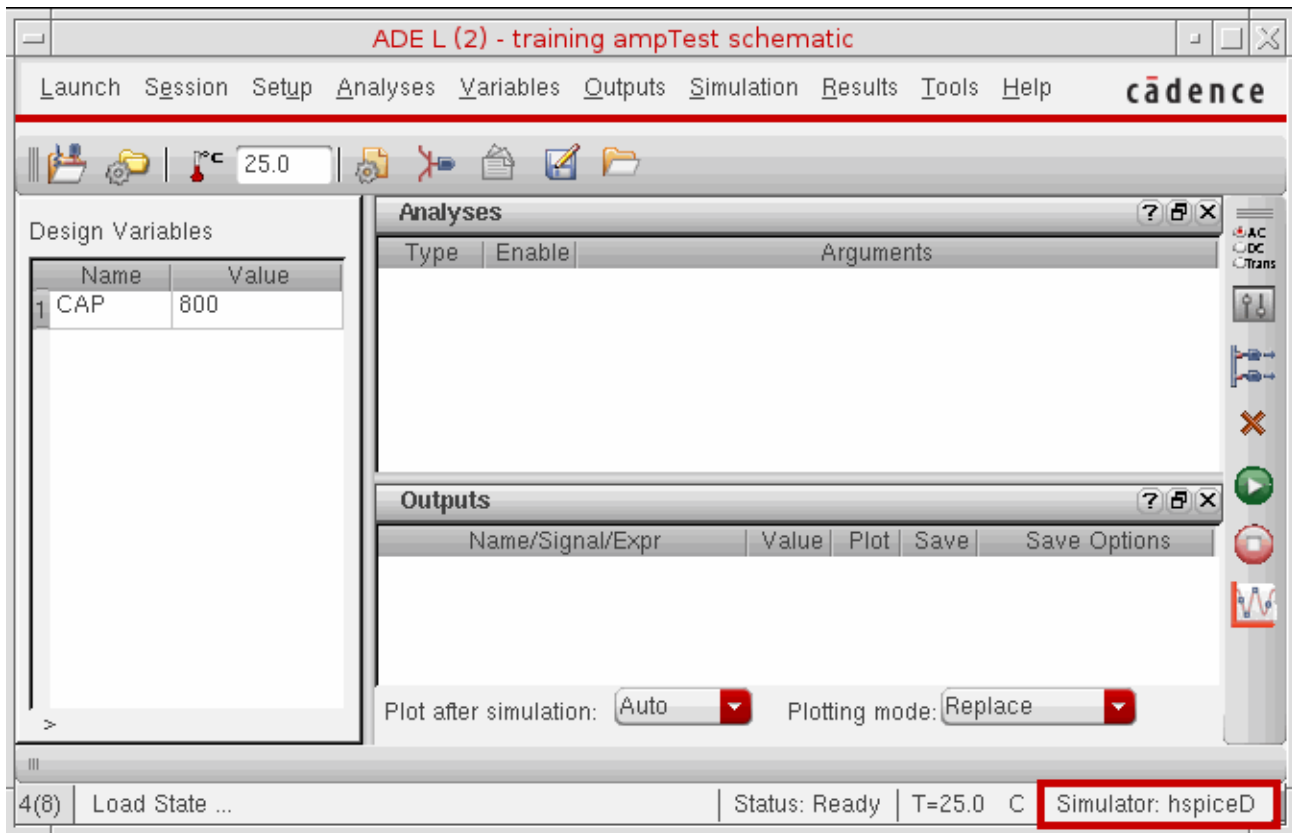


For detailed information about the form, refer to the section Choosing Simulator/Directory/Host.

Virtuoso Analog Design Environment L User Guide

Hspice Direct Support

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



Libraries

The following cells of the `analogLib` library are updated to contain *HspiceD* views. The *HspiceD* simInfo, CDF parameters and netlisting procedures have been added to all these `analogLib` cells:

bcs	bvs	cap	cccs	ccvs	core
diode	iam	idc	iexp	ind	iopamp
ipulse	ipwl	ipwlf	npn	isffm	isin
ixfmr	nbsim	nbsim4	njfet	nmes	nmes4
nmos	nmos4	pbsim	pbsim4	pcapacitor	pdiode

Virtuoso Analog Design Environment L User Guide

Hspice Direct Support

pjfet	pmos	pmos4	pnp	presistor	res
schottky	vam	tline	u1wire	u2wire	u3wire
u4wire	u5wire	usernnp	usernpn	vccap	vccs
vcres	vcvs	vdc	vexp	vpulse	vpwl
vpwlf	vsffm	vsin	winding	xfrm	zener
iprobe	pinductor	mind	pmind	pvccs2	pvccs3
pvcvs	pvcvs2	pvcvs3	pvccs		

Features

The use model of the Analog Design Environment for the *HspiceDirect* simulator is very similar to that of the *Spectre Direct/Hspice Socket* interface. Most of the options work in the same way with a few differences.

Model Libraries

The *Model Library Setup* form remains essentially the same. You can enter model file names into the *Model Library File* field. The listbox displays the list of model files to be included. You can also include an optional *Section* field. When the *Section* field for a particular model file is defined, the netlist will contain the statement,

```
.LIB "<modelLibraryFile>" <section>
```

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

```
.INCLUDE "<modelLibraryFile>"
```

For detailed information about the form, refer to the section [Model Library Setup](#).

Distributed Processing Support

The Distributed Processing mode is supported only for normal simulation and parametric analysis. For detailed information about Distributed Processing, refer to the [Virtuoso Analog Distributed Processing Option User Guide](#).

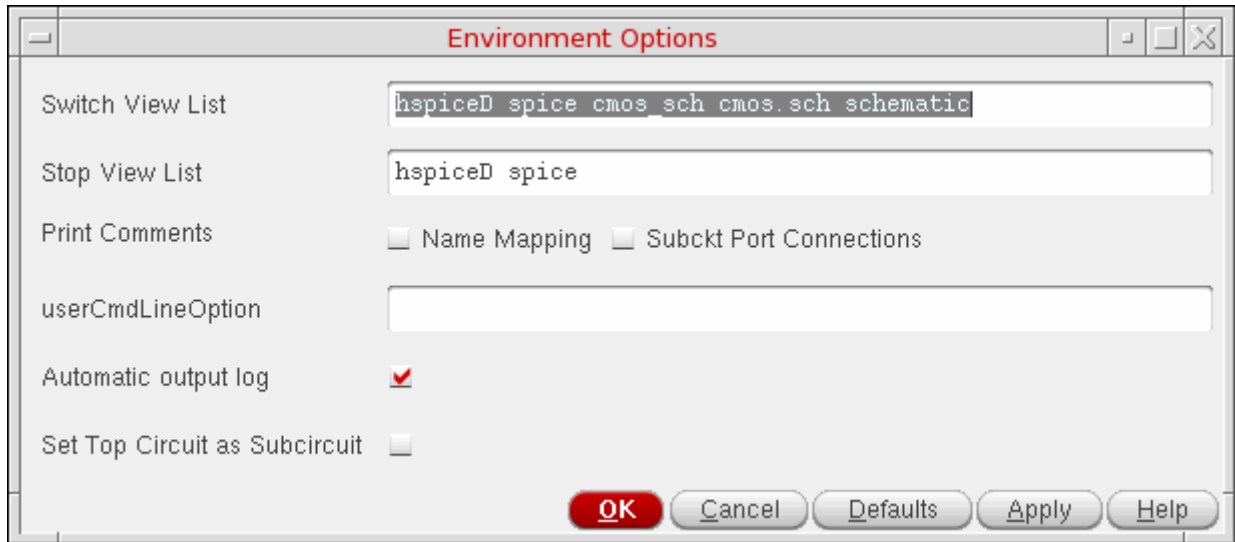
Running Analyses

The *Choosing Analyses* form enables you to set up and run an analysis. This form is explained in details in the [Setting Up for an Analysis](#) chapter of this book. Refer to this section for details about each analysis.

The analyses that are supported are: *DC*, *Transient*, *AC*, *Noise* and *OP*. To run an analysis, select it in the *Choosing Analyses* form. The form re-displays to show the fields that are required for the selected analysis. For more information see [Setting Up an HspiceD Analysis](#) on page 255.

Passing Command Line Options

You can pass the command line options using the *userCmdLineOption* option in the *Environment Options* form.



Alternatively, you can pass the command line options using the *userCmdLineOption* environment variable in the *.cdsinit* file as follows:

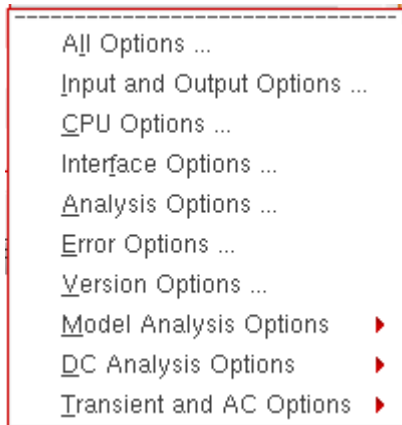
```
envSetVal("hspiceD.envOpts" "userCmdLineOption" 'string "string value")
```

For example, to set multi threading to 3, you should set the *userCmdLineOption* environment variable as shown below:

```
envSetVal("hspiceD.envOpts" "userCmdLineOption" 'string "-mt 3")
```

Analog Options

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



These options can be used to modify various aspects of the simulation, including output types, accuracy, speed, and convergence. For details about the Analog Options, refer to *HSPICE/SPICE Interface Reference*.

Model Analysis Options

The *Model Analysis Options* have been grouped as follows:



- Click *Model Analysis Options – General Options* to specify DCAP, HIER_SCALE, MODMONTE, MODSRH, SCALE, TNOM using the *Hspice General Model Options* form.
- Click *Model Analysis Options – Mosfet Control Options* to specify CVTOL, DEFAD, DEFAS, DEFEL, DEFNRD, DEFNRS, DEFPD, DEFPS, DEFW, SCALM, WL using the *Hspice Mosfet Control Options* form.
- Click *Model Analysis Options – Inductor Options* to specify GENK, KLIM using the *Hspice Inductor Options* form.
- *Model Analysis Options – BJT and Diode Options* to specify EXPLI using the *Hspice BJT and Diode Options* form.

DC Analysis Options

The *DC Analysis Options* have been grouped as follows:

Accuracy Options ...
Matrix Options ...
Input and Output Option ...
Convergence Options ...

- Click *DC Analysis Options – Accuracy Options* to specify ABSH, ABSI, ABSMOS, ABSVDC, DI, KCLTEST, MAXAMP, RELH, RELI, RELMOS, RELV, RELVDC using the *Hspice DC Accuracy Options* form.
- Click *DC Analysis Options – Matrix Options* to specify ITL1, ITL2, NOPIV, PIVOT, PIVREF, PIVREL, PIVTOL using the *Hspice Matrix Options* form.
- Click *DC Analysis Options – Input and Output Option* to specify CAPTAB, DCCAP, VFLOOR using the *HspiceDC Input and Output Options* form.
- Click *DC Analysis Options – Convergence Options* to specify CONVERGE, CSHDC, DCFOR, DCHOLD, DCON, DCSTEP, DV, GMAX, GMINDC, GRAMP, GSHUNT, ICSWEEP, ITLPTRAN, NEWTOL, OFF, RESMIN using the *HspiceConvergence Options* form.

Transient and AC Options

The *Transient and AC Analysis Options* have been grouped as follows:

Accuracy Options ...
Speed Options ...
Timestep Options ...
Algorithm Options ...
Input and Output Options ...

- Click *Transient and AC Options – Accuracy Options* to specify ABSH, ABSV, ACCURATE, ACOUT, CHGTOL, CSHUNT, DI, GMIN, GSHUNT, MAXAMP, RELH, RELI, RELQ, RELV, RISETIME, TRTOL using the *Hspice Transient and AC Options* form.
- Click *Transient and AC Options – Speed Options* to specify AOTPSTOP, BKPSIZ, BYPASS, BYTOL, FAST, ITLPZ, MBYPASS, TRCON using the *Hspice Speed Options* form.

- ▶ Click *Transient and AC Options – Timestep Options* to specify ABSVAR, DVDT, FS, FT, IMAX, IMIN, ITL5, RELVAR, RMAX, RMIN, SLOPETOL, TIMERES using the *Hspice Timestep Options* form.
- ▶ Click *Transient and AC Options – Algorithm Options* to specify DVTR, IMAX, IMIN, LVLTIM, MAXORD, METHOD, MU, PURETP, TRCON using the *Hspice Algorithm Options* form.
- ▶ Click *Transient and AC Options – Input and Output Options* to specify INTERP, ITRPRT, MEASFAIL, MEASSORT, PUTMEAS, UNWRAP using the *Hspice Transient Input and Output Options* form.

Important

All the *Analog Options* mentioned are supported by the *Hspice* version 2003.3. Please ensure that you are using a compatible version of *Hspice*.

Output Log

This displays the file `hspice.out` found under the `psf` directory. This is the file to which the hspice output is re-directed.

Convergence Aids

A screenshot of a dialog box titled "Convergence Aids" with a red border. It contains three radio button options: "Node Set (.NODESET) ...", "Initial Condition (.IC) ...", and "Force (.DCVOLT) ...". The "Node Set (.NODESET) ..." option is selected.

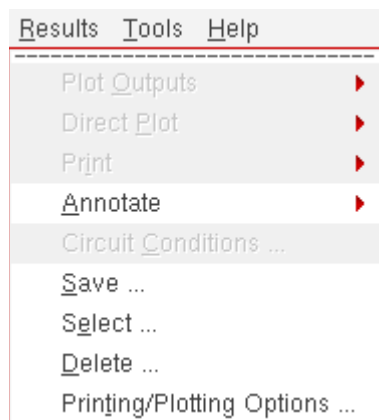
Click *Convergence Aids – Node Set (.NODESET)* to initialize specified nodal voltages for a DC operating point analysis. The `.NODESET` statement is generally used to correct convergence problems in a DC analysis. Setting nodes in the circuit to values that are close to the actual DC operating point solution enhances the convergence of the simulation. The *Select Node Set* form works in the same way as the Spectre Direct/Hspice Socket interface. The netlist will contain the `.NODESET` statement line.

Click *Convergence Aids – Initial Condition (.IC)* or *Convergence Aids – Force (.DCVOLT)* to set the transient initial conditions. The initialization depends on whether the UIC parameter is included in the `.TRAN` analysis statement. If the UIC parameter is specified in the `.TRAN` statement, the Hspice simulator does not calculate the initial DC operating point. Consequently, the transient analysis is entered directly.

The *Select Initial Condition Set* and the *Select Force Node Set* forms work in the same way as the *Spectre Direct/Hspice Socket* interface. The netlist contains the `.IC` and the `.DCVOLT` statement line, whichever the case may be.

Results

You can save, select, delete, restore, plot and print a set of simulation results using the *Results* menu.



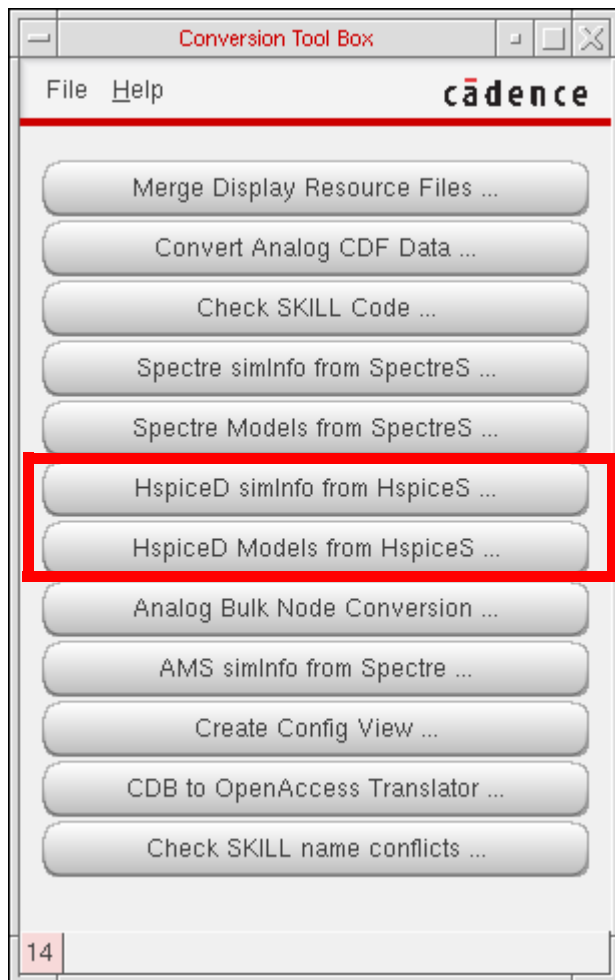
The following menus have been removed from the *Hspice Direct* interface as the *Hspice* simulator does not write the specified data in the `psf` files:

- Plot Outputs – Noise*
- Direct Plot – Equivalent Output Noise*
- Direct Plot – Equivalent Input Noise*
- Direct Plot – Squared Output Noise*
- Direct Plot – Squared Input Noise*
- Direct Plot – Noise Figure*
- Print – Model Parameters*
- Print – Noise Parameters*
- Print – Noise Summary*
- Annotate – Model Parameters*

The noise data is written by the *Hspice* simulator in the `hspice.out` file. Use the menu *Simulation – Output Log* to view the simulator output file.

Converting Libraries

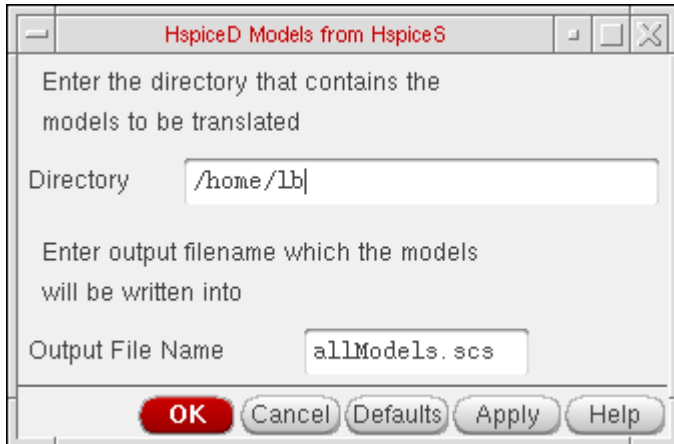
You can migrate existing libraries and model files using the *Conversion Tool Box*. The *Conversion Tool Box* is used to convert design libraries and associated technology data, and to prepare files for conversion to Direct simulation. To bring up the *Conversion Tool Box* window, click *Tools – Conversion Tool Box* in the CIW (Command Interpreter Window). The window displays:



Virtuoso Analog Design Environment L User Guide

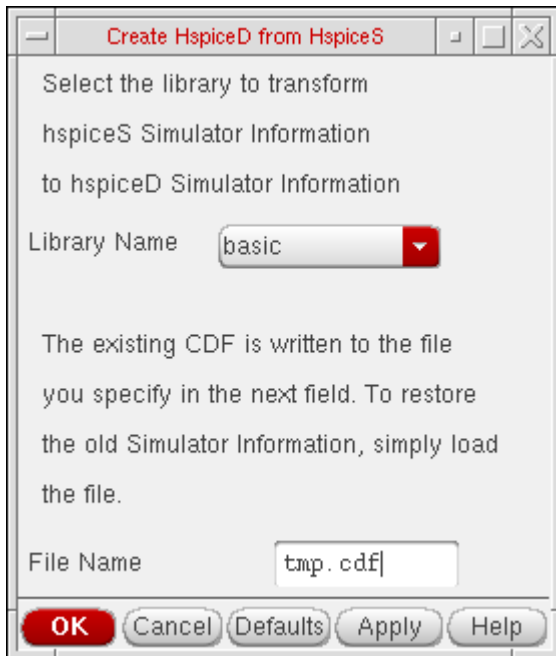
Hspice Direct Support

A new button *HspiceD models from HspiceS...*) has been added to the Conversion Tool Box. Click this button to invoke the *HspiceD Models from HspiceS* window.



This utility locates all the *HspiceS* model files in a directory, translates them into *HspiceD* models, and places the translated models in a single file. This can be included to simulate a circuit (using the *Hspice Direct* interface) by adding the file through the *Model Libraries* form. For detailed information about the form, refer to the section, [Model Library Setup](#). The *HspiceS* model files cannot be translated from two different directories. To do so, use the unix command `cat` to merge the file created from 2 different directories.

To transform *HspiceS* simulator information to *HspiceD* simulator information, click the *HspiceD simInfo from HspiceS* button. The *Create HspiceD from HspiceS* window displays.



Control Mapping of Nets

In HspiceD netlister, you can control mapping of `gnd!` nets by setting an environment variable as shown below:

```
envSetVal ("hspiceD.envOpts" "mapGndNetToZero" 'boolean t)
```

The default value of this environment variable is `t` implying that by default, the hspiceD netlister maps `gnd!` nets to 0. On setting this variable to `nil`, the HspiceD netlister stops mapping `gnd!` nets to 0. You can set the variable to `nil` in `.cdsinit` file.

Virtuoso Analog Design Environment L User Guide

Hspice Direct Support

UltraSimVerilog



Important

The functionality described in this chapter is available only in IC6.1.6. It is not available in ICADV12.1.

This chapter describes how to use the UltraSimVerilog simulator in the Cadence® mixed signal circuit design environment to simulate mixed signal designs, and provides the following information:

- [“Interface Element Macro Models”](#) on page 583
- [“Netlisting Options”](#) on page 588
- [“Running a Mixed Signal Simulation”](#) on page 590

Note: The 64-bit version of the Virtuoso® UltraSim simulator does not support UltraSimVerilog.

Refer to the following resources for additional information:

- [Virtuoso Mixed-Signal Circuit Design Environment User Guide](#)
- [Virtuoso Spectre Circuit Simulator Reference](#)
- [Virtuoso Schematic Editor L User Guide](#)

Interface Element Macro Models

An interface element (IE) is a two-terminal device that connects two partitions and splits the original net. IEs are generated automatically for input and output terminals of digital components connected to interface nets.

IE attributes:

- Model the loading and driving impedance of digital instance terminals

- Convert voltages to logic levels and vice versa
- Transport events between two simulators

An IE model file is a text file that contains an IE primitive and other circuit components that characterize loading and nonlinear effects. The IE primitive inherits its parameters from the instantiation of the IE macro model.

When using the UltraSimVerilog simulator, you can choose to model an IE instance with either a primitive or a macro model file. IEs modeled as primitives reduce the need to have IE macro model files.

An IE is modeled as a primitive if its `macro` component description format (CDF) parameter is set to `nil`. Refer to Inherited CDF Parameters in the *Virtuoso Mixed-Signal Circuit Design Environment User Guide* (IC6.1.6 only) for more information.

Inline Subcircuit

The UltraSimVerilog simulator supports inline subcircuits in IE macro models. For the Virtuoso UltraSim™ simulator IE models, inline subcircuits are preferred over regular subcircuits. With an inline subcircuit, you do not have to specify the nesting level (`nestlev`) of the IE primitive (default is 0), and the IE primitive name appears at the same level as the interface net in the design hierarchy.

Interface Element Selection Rules

There are two modes for generating IEs: Detailed and nondetailed. Flat netlisting (FNL) supports both detailed and nondetailed IE generation. Hierarchical netlisting (HNL) supports only nondetailed IE generation. For UltraSimVerilog simulation, only HNL is available.

Simulation Accuracy and Performance

This section describes the following IE macro models for use with the mixed signal simulators, including UltraSimVerilog.

- [“Analog-to-Digital \(A2D\) Models”](#) on page 585
- [“Digital-to-Analog \(D2A\) Models”](#) on page 586

Analog-to-Digital (A2D) Models

Model Description

An analog-to-digital (A2D) IE macro model is needed to connect the input pin of a digital component to an interface net.

You can develop an A2D IE for a circuit simulator using the following parts:

- An A2D interface primitive that converts a voltage value to a logic state
- Optional analog primitives that model other characteristics of a digital input pin (for example, loading current and capacitance)

The A2D interface primitive performs the most basic analog-to-digital conversion step by sensing voltage and converting it to a logic state.

Optional analog primitives, such as resistors, capacitors, and transistors, can be used to connect the A2D IE to the actual analog circuitry in the design, so additional behaviors of the digital input pin are reflected in the A2D IE macro model. The implementation of these primitives is simulator-dependent.

Models for the Virtuoso UltraSim Simulator

The sample IE models provided by Cadence in the `analogLib` and `ieLib` libraries include IE macro model files for the Virtuoso UltraSim simulator.

Virtuoso UltraSim Example

The following example illustrates an instantiation of an A2D IE, `MOS_a2d`. Assume that it has the following CDF properties:

Property	Value
<code>macro</code>	(empty)
<code>a2d_v0</code>	1.5
<code>a2d_v1</code>	3.5
<code>a2d_tx</code>	1m

Because the macro property is an empty string, this instance is formatted as a primitive.

The CDF simulation information for this instance includes:

Field Name	Value
<i>otherParameters</i>	macro
<i>instParameters</i>	timex v1 vh
<i>propMapping</i>	nil v1 a2d_v0 vh a2d_v1 timex a2d_tx
<i>componentName</i>	Empty (default prefix <code>_ie</code> is used)

The *propMapping* value maps CDF properties to the Virtuoso UltraSim simulator properties. For example, the CDF property `a2d_v0` maps to the simulator property `v1`. The Virtuoso UltraSim simulator property values are as follows:

Virtuoso UltraSim Property	Value
<code>v1</code>	1.5
<code>vh</code>	3.5
<code>timex</code>	1 m

The Virtuoso UltraSim simulator property values are passed to the instance (`_ie99999` in this example), and the instance is formatted as

```
_ie99999 ( INetName1 0) a2d dest="99999" timex=1m v1=1.5 vh=3.5
```

Note: `a2d` is an IE primitive.

Digital-to-Analog (D2A) Models

Model Description

An instantiation of a digital-to-analog interface model is required for connecting an output pin of a digital component to an interface net. You define the circuit simulator interface model in a macro file, which is then included in the netlist.

A digital-to-analog (D2A) IE macro model for a circuit simulator can be created using the following parts:

- A D2A simulator primitive that converts a logic state to a voltage value
- Optional analog primitives that model other characteristics of a digital output Dpin (for example, drive and loading)

The D2A interface primitive performs the most basic digital-to-analog conversion step by converting a logic state to a voltage and timing relationship. It is implemented in slightly different forms in the Virtuoso UltraSim simulator and other SPICE simulators.

Optional analog primitives, such as resistors, capacitors, and transistors, can be used to connect the D2A interface primitive to the actual analog circuitry in the design. This allows additional behaviors of the digital output pin to be reflected in the D2A interface model. The implementation of these primitives is simulator-dependent.

Models for the Virtuoso UltraSim Simulator

See [“Models for the Virtuoso UltraSim Simulator”](#) on page 585 for more information.

Virtuoso UltraSim Example

The following example illustrates an instantiation of a D2A IE and CML3_d2a, from the `ieLib` library. Assume that it has the following properties:

Property	Value
macro	"CML3_d2a"
d2a_vl	-450 m
d2a_vh	0
d2a_tr	900 p
d2a_tf	800 p

Because the macro property is not empty, the instance is formatted as a macro model.

The CDF simulation information for this instance includes:

Field Name	Value
<i>otherParameters</i>	macro

Virtuoso Analog Design Environment L User Guide

UltraSimVerilog

Field Name	Value
<i>instParameters</i>	fall rise val1 val0
<i>propMapping</i>	nil val1 d2a_vh val0 d2a_vl rise d2a_tr fall d2a_tf
<i>componentName</i>	Empty (default prefix <code>_ie</code> is used)

The *propMapping* value maps CDF properties to Virtuoso UltraSim simulator properties. For example, the CDF property `d2a_tr` maps to the simulator property `rise`. The Virtuoso UltraSim simulator property values are as follows:

Virtuoso UltraSim Property	Value
<code>val0</code>	-450 m
<code>val1</code>	0
<code>rise</code>	900 p
<code>fall</code>	800 p

The Virtuoso UltraSim simulator property values are passed to the instance (`_ie99998` in this example) and the instance is formatted as

```
_ie99998 ( INetName2 0) CML3_d2a src="99998" fall=800p rise=900p val1=0 val0=-450m
```

Note: `CML3_d2a` is an IE macro.

See the *Virtuoso Mixed-Signal Circuit Design Environment User Guide* for more information.

Netlisting Options

Both hierarchical netlisting (HNL) and flat netlisting (FNL) are available in the front-end mixed signal simulation flow. Mixed signal HNL and FNL differ in direct simulation: The UltraSimVerilog simulator supports HNL but not FNL.

Because HNL offers advantages over FNL, it is recommended that you use HNL unless you require features that are only available in FNL.

Verilog Netlisting Options

To access the Verilog® netlisting options:

1. Choose *Setup – Environment* in the Cadence Analog Design Environment simulation window.

The Environment Options form appears.

2. Choose the *Verilog Netlist Option* button.

The Verilog HNL Netlisting Options form appears.

Verilog HNL Netlisting Options

Netlist For LAI/LMSI Models

Generate Test Fixture Template Verimix Overwrite Verimix Stimulus

Netlist Uppercase <input type="checkbox"/>	Generate Pin Map <input type="checkbox"/>	Preserve Buses <input checked="" type="checkbox"/>
Netlist SwitchRC <input type="checkbox"/>	Skip Null Port <input type="checkbox"/>	Netlist Uselib <input type="checkbox"/>
Drop Port Range <input checked="" type="checkbox"/>	Incremental Config List <input type="checkbox"/>	Symbol Implicit <input type="checkbox"/>
Assign For Alias <input type="checkbox"/>	Skip Timing Information <input type="checkbox"/>	Declare Global Locally <input type="checkbox"/>
Netlist Explicitly <input type="checkbox"/>	Support Escape Names <input type="checkbox"/>	

Global Power Nets vdd! vdda! vddd! vcc! vcca! vccd!

Global Ground Nets gnd! gnda! gnnd! vss! vssa! vssd! vee! veea! veed!

Global TimeScale Overwrite Schematic TimeScale

Global Sim Time 1 Unit ns

Global Sim Precision 1 Unit ns

OK Cancel Defaults Apply Help

Some of the key Verilog netlisting options include:

- *Global Power Nets* and *Global Ground Nets*
- *Netlist SwitchRC*, *Skip Null Port*, and *Netlist Explicitly*

Note: These options are only applicable for Verilog FNL.

- *Generate Test Fixture Template, Netlist Uppercase, Netlist SwitchRC, Global TimeScale Overwrite Schematic TimeScale, Global Sim Time, and Global Sim Precision*

Note: These options are only applicable for Verilog HNL.

For more information about these options, refer to the [Netlist](#) section in the *Virtuoso NC Verilog Environment User Guide*.

Hierarchical Netlisting

A typical mixed signal design contains hierarchical blocks and primitives. Blocks can contain lower-level instances and connectivity; primitives do not. HNL retains the hierarchical design and translates a non-primitive cellview into either a subcircuit for the analog simulator or a module for Verilog.

The mixed signal netlister creates the analog and digital netlists separately. The analog HNL netlists the analog partition and creates an analog directory. Verilog HNL netlists the digital partition and creates a digital directory.

- Analog blocks and primitives are netlisted by analog HNL.
- Digital blocks and primitives are netlisted by digital HNL.
- Mixed blocks are netlisted by both analog and digital HNL.

See the *Virtuoso Mixed-Signal Circuit Design Environment User Guide* for more information.

Running a Mixed Signal Simulation

A mixed signal simulation involves a number of setup and processing steps.

- [Setting Simulator Options](#)
- [Input Stimulus for HNL](#)
- [Setting Design Variables](#)
- [Choosing Analyses](#)
- [Running the Simulation](#)
- [Control and Debugging](#)
- [Viewing and Analyzing Simulation Output](#)

The UltraSimVerilog simulator does not support the direct simulation non-batch control feature. Consequently, you cannot interactively control and debug both AHDL and Verilog modules.

Once your design is simulated, you can probe the layout and schematic views to determine specific characteristics of the design. This process may require numerous iterations until the results are acceptable.

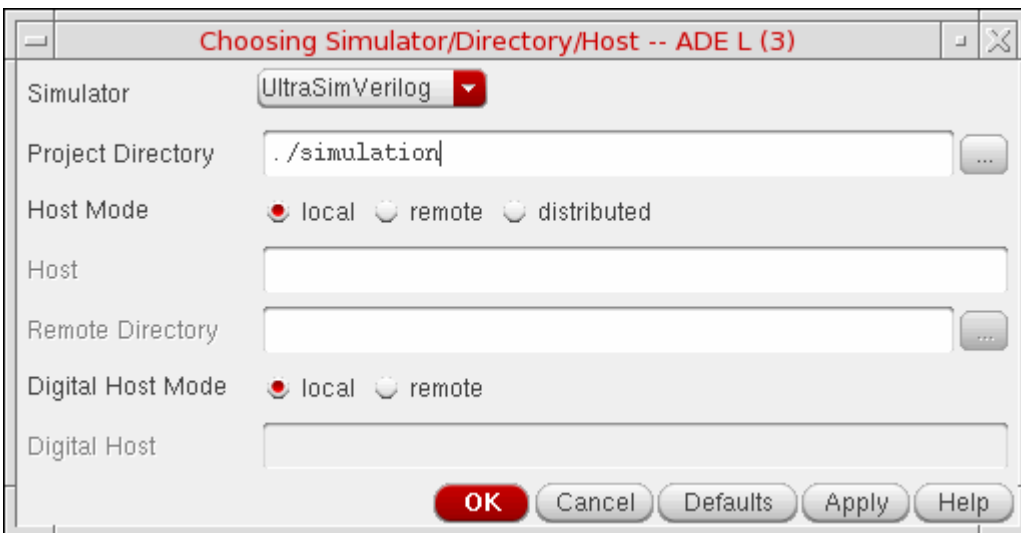
Setting Simulator Options

Simulator Selection

To select a simulator

1. Open the Cadence Analog Design Environment simulation window using one of the following methods:
 - From the command interpreter window (CIW), choose *Tools – Analog Environment – Simulation*.
 - From the Schematic window, choose *Tools – Analog Environment*.
2. Choose *Setup – Simulator/Directory/Host*.

The Choosing Simulator/Directory/Host form appears.



3. Choose *UltraSimVerilog* from the *Simulator* cyclic.

4. In the *Project Directory* field, type the name of the directory in which you will be working.
5. Click *OK*.

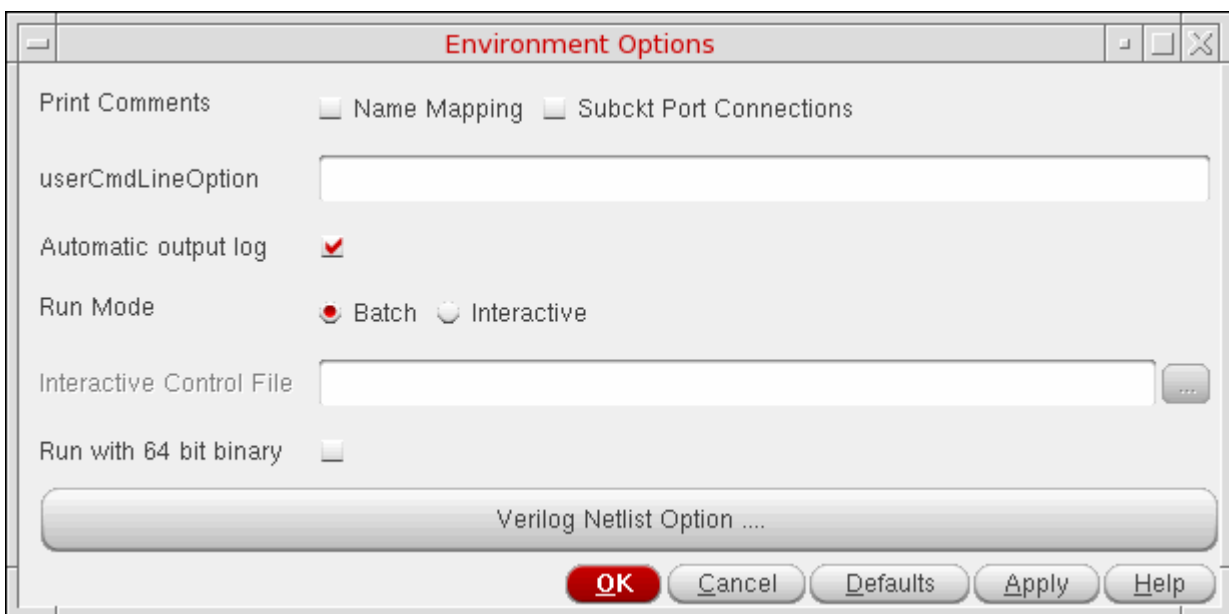
Note: If the design data is not displayed in the simulation window, choose *Setup – Design* and select the appropriate design.

Environment Options

To set the environment options

1. Choose *Setup – Environment* in the simulation window.

The Environment Options form appears.



2. Add information to the form, as needed, for your simulator and design.
3. Click the *Verilog Netlist Option* button to choose the appropriate options for netlisting.

The *Verilog HNL Netlisting Options* form appears.

4. Select *Verimix* in *Generate Test Fixture Template*.

This following files are generated:

testfixture. template	Contains a test module declaration and include statements for IEs, testfixture.verimix, and \$shm_probe definitions.
testfixture. verimix	A stimulus file from Verilog Integration that is compatible with the testfixture.verilog file.

Note: If a testfixture.verimix file already exists and you need to create a new one, choose *Overwrite Verimix Stimulus* to generate a new file.

5. Specify the global power and ground nets in your design.

An HDL global module `cds_global.v` is created with global power nets (`vcc_`) declared as `supply1`, and global ground nets (`gnd_`, `vss_`) declared as `supply0`. The

module defines all other global nets that are not defined in either of the global net fields by using wire declarations.

6. Click *OK* to close the Verilog HNL Netlisting Options form.
7. Click *OK* to close the Environment Options form.

Logic Simulator Options

To set simulator options for the logic simulator (Verilog)

1. In the Simulation window, choose *Simulation – Options – Digital*.
The Verilog-XL Simulation Options form is displayed.

The screenshot shows the 'Verilog-XL Simulation Options' dialog box. It is organized into several sections:

- Acceleration:** Includes checkboxes for 'Gate' (checked), 'Switches', 'Behavioral', and 'Twin Turbo'. It also has radio buttons for 'Continuous Assignments' (unchecked), 'Keep Nodes' (Minimum selected), and 'Behavioral' (Default selected). There are also radio buttons for 'No Turbo', 'Turbo1', 'Turbo2', and 'Turbo3'.
- Delays:** Includes radio buttons for 'Mode' (Default selected) and 'Type' (Typical selected).
- Pulse Control:** Includes input fields for 'Error %' (100) and 'Reject %' (100), and a checkbox for 'Use Pulse Control Parameters' (unchecked).
- Other Options:** Includes checkboxes for 'Stop After Compilation', 'Use Behavior Profiler', 'Suppress Messages', 'SimVision Debugger', and 'Suppress Warnings'.
- File Fields:** Includes text boxes for 'Command File', 'Options File', 'Library Files', 'Library Directories', 'Verilog-XL Executable' (with 'verilog.vmx' entered), and 'Simulation Log File' (with 'verilog.log' entered). Each text box has a browse button ('...').
- Buttons:** At the bottom, there are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

2. Set the options for the logic simulator as needed.

For details about the Verilog options, refer to the *Virtuoso Verilog-XL Environment Reference Manual*.

3. If you want to enter Verilog-XL commands directly or debug the simulation using the SimVision debugger, select *Stop After Compilation*.

Note: The *SimVision Debugger* option is also selected (and cannot be deselected) if *Stop After Compilation* is selected.

This stops Verilog-XL after compilation and lets you enter Verilog commands through the SimVision window (SimVision is the Verilog-XL graphical environment – see the *Verilog-XL User Guide* for information on how to use SimVision).

4. Click *OK*.

Input Stimulus for HNL

Analog stimulus for HNL can be supplied by using an analog stimulus block and file, or by using a digital stimulus block and file for digital stimulus.

HNL input stimulus rules and restrictions:

- Analog and digital stimulus blocks function the same in HNL and FNL
- Stimulus blocks can drive both analog and digital components
- In HNL and FNL, stimuli in an analog stimulus file can drive only analog nets and interface nets
- The analog stimulus file in HNL can be used to drive signals in only the top-level cellview
- In HNL and FNL, a digital stimulus file cannot drive analog components
- The file formats of HNL and FNL digital stimulus files are incompatible

Analog Stimuli

Analog Stimulus Block

An *analog stimulus block* or *instance* on the schematic can drive both analog and digital circuitry. This eliminates the need to supply input through the schematic for analog components that can be simulated using digital input.

Analog stimulus blocks include voltage sources, current sources, and behavioral instances.

To create an analog stimulus block or instance:

1. Create a behavioral view for the stimulus block.
2. Define the stimulus module.
3. Write SpectreHDL or Verilog-A in a module definition.

The file syntax is the same for FNL and HNL.

4. Create a symbol view for the stimulus block.
5. Place the symbol in the top-level schematic and connect it to the appropriate terminals.

The following sample analog stimulus block shows the format of a file that implements a swept sinusoidal source.

```
\include "discipline.h"
\include "constants.h"
\define PI          3.14159
// - Swept sinusoidal source
// sigout_p,sigout_n:output (val,flow)
// INSTANCE parameters
//   start_freq      = start frequency [Hz]
//   sweep_rate      = rate of increase in frequency [Hz/s]
//   amp             = amplitude of output sinusoid (val)
//   points_per_cycle = number of points in a cycle of
//   the output []
// The instantaneous frequency of the output is 'sweep_rate'
//   * 'time' plus 'start_freq'.
module swept_sine_src(sigout_p,sigout_n);
output sigout_p,sigout_n;
electrical sigout_p,sigout_n;
parameter real start_freq = 1 from (0:inf);
parameter real sweep_rate = 1;
parameter real amp = 1 from (0:inf);
parameter real points_per_cycle = inf from [6:inf];
  real freq;
  real phase;
  analog begin
    phase = 2*\PI*(start_freq + sweep_rate / 2 *
    $realtime)*$realtime;
    freq = start_freq + sweep_rate * $realtime ; // =
    d/dt(phase)
    V(sigout_p,sigout_n) <+ amp*sin(phase);
    if (points_per_cycle != inf) begin
      // ensure that model is evaluated sufficiently often
      bound_step(1 / (freq*points_per_cycle));
    end
  end
endmodule
```

Analog Stimulus File

In HNL, use the *analog stimulus* file to connect stimuli to only the nets in the top-level cellview. This requirement is imposed by the circuit simulator because stimuli may not be allowed to connect to nets embedded in lower levels of the design hierarchy. Mapping nets or instance names in lower levels of the design hierarchy is allowed.

For example, if *in<3:0>* and *control* are signals existing in the top-level cellview, you can use voltage sources to drive `[/in<3>]` and `[/control]` in the stimulus file, as shown in the following example:

```
*comment, inst /inst<1> is mapped to [$/inst<1>]
*comment, inst /a/b is mapped to [$/a/b]
*comment, net /a/net<1> is mapped to [$/a/net<1>]
v0 [$/in<3>] 0 dc 5v
v1 [$/control] 0 dc 3v
```

Note: Analog stimuli supplied by an analog stimulus file can drive only analog nets and interface nets. Attempts to drive or refer to digital nets with analog stimuli in the stimulus file generates errors.

To generate an analog stimulus file:

1. Choose *Setup – Stimulus – Edit Analog* (or *Setup – Stimuli – Analog* if you are using the UltraSimVerilog simulator) in the simulation window.

The Edit Stimulus File form appears.

For mixed signal direct simulation (UltraSimVerilog), you can generate an analog stimulus file through the Setup Analog Stimuli form, or you can provide the analog stimulus in a text file. The information you enter on the Setup Analog Stimuli form is converted to a stimulus file. Refer to the *Virtuoso Analog Design Environment L User Guide* for more information about entering analog stimuli.

2. Make stimulus changes as needed.

You must use analog simulator language to define analog stimuli (file syntax is compatible with the analog design environment or ADE).

3. Change signal and instance names to ADE naming conventions.

All instance and net names in the stimuli file must be specified in the database name space. Instance names must be enclosed within the `[$]` mapping macro and net names enclosed within the `[/]` macro.

4. Save the file and close the editor.

Digital Stimuli

Digital Stimulus Block

A *digital stimulus block* can drive both analog and digital input. This eliminates the need to supply input through the schematic for analog components that can be simulated using digital input.

To create a modified stimulus block:

1. Create a behavioral view for the stimulus block.
2. Define the stimulus module.
3. To force input, add Verilog commands to the module definition.

For more details, refer to the *Verilog-XL Reference* and the *Virtuoso Verilog-XL Environment Reference Manual*.

4. Create a symbol view for the stimulus block.
5. Place the symbol in the top-level schematic and connect it to the appropriate terminals.

The following sample of a Verilog stimulus block shows the format of the file.

```
//timescale set according to user specification
`timescale 10ns/10ns
//Define the Stimulus block
module Stim (tx, precharge);
    output [1:16] tx ;
    output precharge ;
//Defines the registers
    reg [1:16] tx ;
    reg precharge ;
initial begin
    tx = 16'h0000;
    precharge = 1'b0;
end
initial begin
    #2418 tx[3] = 1'b1;
    #17 tx[3] = 1'b0;
end
initial begin
    #1558 tx[6] = 1'b1;
    #17 tx[6] = 1'b0;
end
initial begin
    #1597 tx[7] = 1'b1;
    #17 tx[7] = 1'b0;
end
initial begin
    #37 precharge = 1'b1;
    #11 precharge = 1'b0;
    #27 precharge = 1'b1;
```

```
#11 precharge = 1'b0;
#333 precharge = 1'b1;
#11 precharge = 1'b0;
#38 precharge = 1'b1;
#11 precharge = 1'b0;
#27 precharge = 1'b1;
#11 precharge = 1'b0;
end
endmodule
```

Note: If *Generate Test Fixture Template* in the Verilog HNL Netlisting Options form is set to Verimix, the HNL testfixture files (`testfixture.template` and `testfixture.verimix`) are automatically generated at netlisting time.

testfixture.verimix File

The `testfixture.verimix` file is the stimulus file that is included with the `testfixture.template` file. The stimulus file is a separate file, to prevent overwriting when the `testfixture.template` file is regenerated. Do not use OSS naming conventions in the Verilog HNL `testfixture.verimix` file, because name mapping is not used in the `testfixture.verimix` file. Use Verilog design names directly in the `testfixture.verimix` file.

Note: Testfixture files cannot be shared between FNL and HN (the formats of the files are not compatible).

To switch from HNL to FNL using the same simulation directory, replace HNL `testfixture.template` with FNL `testfixture.template`.

To switch from FNL to HNL using the same simulation directory, the HNL `testfixture.template` and `testfixture.verimix` files are created automatically. Edit the `testfixture.verimix` file to provide the necessary stimulus.

The following example shows a hierarchical `testfixture.template` file.

```
`timescale 1ns / 1ns
module test;
wire out;
integer dc_mode_flag;
integer output_change_count;
integer max_dc_iter;
integer dc_iterations;
time vmx_time_offset;
pulledIE2Top top(out);
`define verimix
`ifndef verimix
//vms and dc iteration loop definitions
`include "IE.verimix"

//please enter any additional stimulus
//in the testfixture.verimix file
`include "testfixture.verimix"
```

Virtuoso Analog Design Environment L User Guide

UltraSimVerilog

```
//$shm_probe definitions
  \include "saveDefs"
\endif
```

In HNL mode, the `testfixture.verimix` file must strictly conform to the Verilog Integration standard for stimulus files. Instance and net name mapping macros, such as `[$]` and `[#]`, are not allowed. You can access or drive any instance or net in the design hierarchy because Verilog allows out-of-context references to instances and nets embedded within lower-level modules of the design hierarchy.

Note: You cannot use a digital stimulus to provide input directly to an analog gate (instead, use a stimulus block).

To create or edit a `testfixture.verimix` file:

1. Choose *Setup – Stimulus – Edit Digital*.

A text editor window containing the `testfixture.verimix` file appears.

2. Make stimulus changes as needed for the simulation.

The information in these files must be in Verilog Integration format. Refer to the raw netlist file from your design for signal and instance names in Verilog name space. For more information about Verilog stimulus information and syntax, see the *Virtuoso Verilog-XL Environment Reference Manual*.

3. Save the file and close the editor.

The following example shows a `testfixture.verimix` file:

```
// Verilog stimulus file.
// Please do not create a module in this file.
// Default verilog stimulus.
initial begin
  [# /A] = 1'b0;
  [# /B] = 1'b0;
  [# /C] = 1'b0;
end
```

If you create, delete, or change the name of an input terminal in your design, you may need to generate a new `testfixture.verimix` file, to avoid having the `testfixture.verimix` file erroneously refer to deleted signals or leave new signals uninitialized.

Use one of the following methods to modify the file:

1. From the Environment Options form, click on the *Verilog Netlist Option* button to access the netlisting options form.
2. In the Verilog HNL Netlisting Options form, select *Overwrite Verimix Stimulus* to create a new `testfixture.verimix` file at netlisting.

3. Edit the `testfixture.verimix` file for stimulus.

or

4. From the Simulation window, choose *Setup – Stimulus – Edit Digital*.

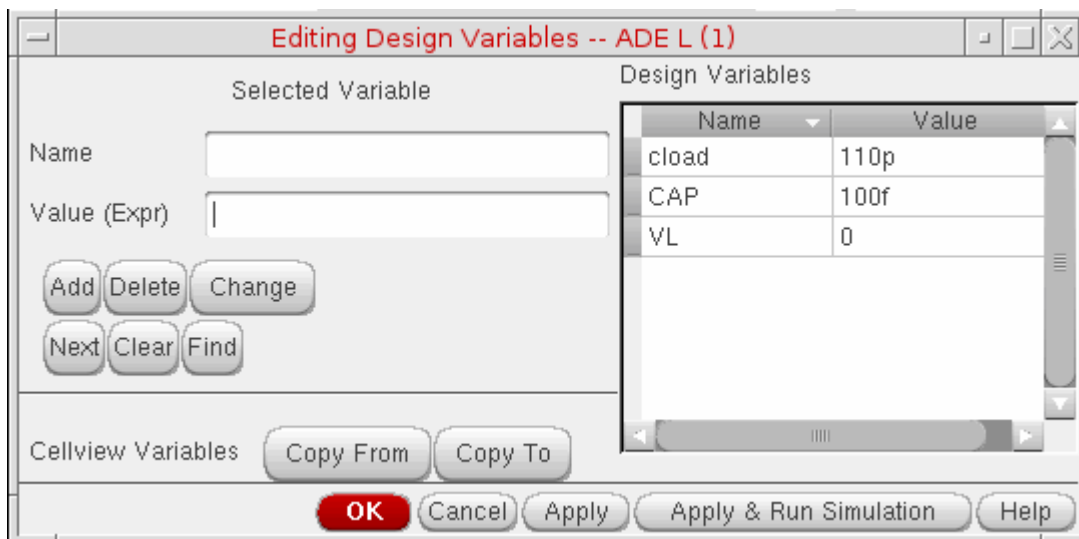
5. Edit the `testfixture.verimix` file directly.

Setting Design Variables

To add, modify, or delete design variable values:

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

The Editing Design Variables form appears.



2. To add a design variable:

a. Type the name and value of the new variable in the *Selected Variable Name* and *Value* fields.

Note: Do not use simulator reserved words as design variable names. For more information, see [Reserved Words](#) on page 119.

b. Click *Add*.

The new variable is added to the *Table of Design Variables* list box.

3. To modify a design variable:

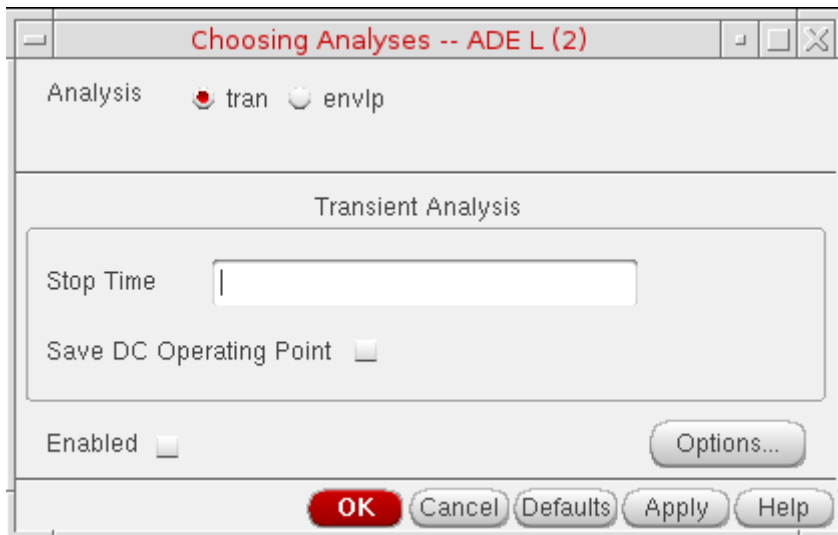
- a. Click on the variable in *Table of Design Variables*.
 - b. The variable name and its value are displayed in the *Name* and *Value* fields.
 - c. Make changes to the name or value as needed.
 - d. Click *Change*.
4. To delete a variable:
- a. Click on the variable in *Table of Design Variables*.
 - b. Click *Delete*.
5. Click *OK* to return to the simulation window.

Choosing Analyses

To choose the analysis for this simulation:

1. Choose *Analyses – Choose* in the simulation window.

The Choosing Analyses form appears.



The form contains different fields depending on the simulator you are using and the analysis you choose.

2. Click the *tran* button for transient analysis.
3. Set *Stop Time*.

Note: Make sure *Enabled* is selected.

4. Click *OK*.

Running the Simulation

To run the simulation:

- ▶ Choose *Simulation – Run*.

The partitioner reads the design and creates the partitioning information, the IE generator creates IEs on the interface nets, and the netlister generates the analog and digital netlists from partitioning and IE generation information.

Control and Debugging

You can control and debug mixed signal simulations interactively, as well as set breakpoints, step through module code, and run, stop, and resume a simulation.

To debug Verilog modules, use the SimVision debugger or the Type-In window to enter Verilog-XL debugging commands.

- ▶ Choose the *Stop After Compilation* or *SimVision Debugger* option in the Verilog-XL Simulation Options window (see “[Logic Simulator Options](#)” on page 594 for more information)

During simulation, Verilog-XL execution starts in the SimVision window. Verilog-XL stops after initialization if *Stop After Compilation* is selected. You can then enter Verilog debugging commands through the SimVision debugger or through the Type-In window. See the *Verilog-XL User Guide* for information about SimVision.

Viewing and Analyzing Simulation Output

Once the simulation is complete, there are several ways to probe, view, and print values and to display waveforms. For more information on how to

- Select data to save and plot
- Plot and print data
- Use the waveform calculator and viewer

refer to the [Virtuoso Visualization and Analysis XL User Guide](#).

Virtuoso Analog Design Environment L User Guide

UltraSimVerilog

For information about the Virtuoso UltraSim simulator options, refer to the *Simulation Options* chapter of the *Virtuoso UltraSim Simulator User Guide*.

Using the Reliability Simulator Interface

This chapter describes Cadence® Virtuoso® Reliability Simulator interface in ADE.

Note: The Reliability Simulator interface in ADE is supported with the Spectre Simulator only if you are using MMSIM 7.2 or a later version.

See the following topics for more information:

- [Introduction](#) on page 606
- [Specifying Reliability Options](#) on page 606
- [Running the Reliability Simulation](#) on page 608
- [Viewing the Reliability Simulation Results](#) on page 608
- [Viewing the Reliability Aged Netlist](#) on page 612
- [Annotating Simulation Results to the Schematic](#) on page 612

Introduction

Reliability Simulator performs reliability simulation to analyze the effects of time on circuit performance drift and predict the reliability of designs in terms of performance. To run a reliability simulation, you need to perform the following tasks:

1. Specify Reliability options
2. Run the Reliability simulation
3. View the Reliability simulation results
4. View the Reliability aged netlist
5. Annotate simulation results to the schematic

Note: Reliability Simulator supports only the psfbin and sst2 waveform formats. Therefore, the simulation results are also created in the psfbin or sst2 format, irrespective of the specified format.

Specifying Reliability Options

To specify Reliability options, do the following:

1. Choose *Simulation – Reliability – Setup*.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

The *Reliability* form appears.

The screenshot shows the 'Reliability' dialog box with the following settings:

- Reliability Analysis**
 - Enable Reliability:
 - Simulator Mode: Spectre native RelXpert
- Simulation Mode**
 - Enable stress:
 - Enable aging:
 - Mode: Hot-Carrier Injection (HCI) Negative Bias Thermal Instability (NBTI) Postive Bias Thermal Instability (PBT)
- Aging time**: 10 | Years
- Age method**: agemos
- Effective model calculation**:
- Enable lifetime calculation**:
- Degradation criteria**: 0.1
- Unified reliability interface (URI) libraries**: [empty] ...
- URI mode**: none
- Append Type**: none dev inline sub
- URI Debug Mode**: 0 1
- TMI**
 - self-heating (tmiShe): aging only (tmiShe=0)

Buttons at the bottom: OK, Cancel, Defaults, Apply, Help

Note: For more information about the Reliability Options form, see [Reliability Options](#) on page 614.


2. Specify the required options and click OK.

Running the Reliability Simulation

To run a reliability simulation, do the following:

1. Ensure that the following is done:
 - Model libraries are included in the reliability model
 - A transient analysis is set up and enabled in ADE
 - The Enable Reliability check box is selected in the Reliability form

Note: A transient analysis is required to run the Reliability simulation.

2. Click the `Netlist` and `Run` button,  .

Note: Reliability Simulation always runs in batch mode. Therefore, the ADE mode too is automatically set to batch when you enable Reliability, and the following message is displayed in CIW:

```
*WARNING* The Spectre run mode needs to be 'batch' when running RelXpert. Automatically setting the run mode to 'batch'.
```

Viewing the Reliability Simulation Results

In the ADE window, choose *Results – Reliability Data* to access the Reliability simulation results. You can use this menu to view the following results:

- [Device Lifetime and Degradation Results](#)
- [Device Characteristic Degradation Results](#)
- [Model Parameter Changes Results](#)
- [TMI-aging Results](#)

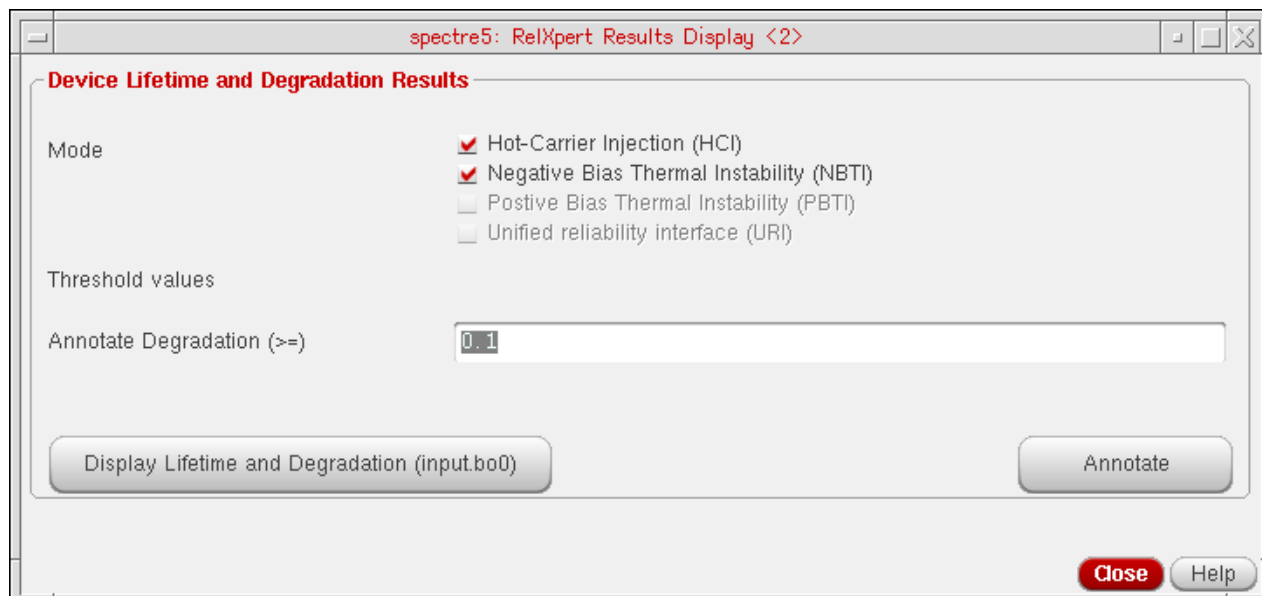
Device Lifetime and Degradation Results

To view the Device Lifetime and Degradation results, do the following:

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

- ➔ In the ADE window, choose *Results – Reliability Data – Device Lifetime and Degradation*. The Device Lifetime and Degradation Results form appears.



Device Characteristic Degradation Results

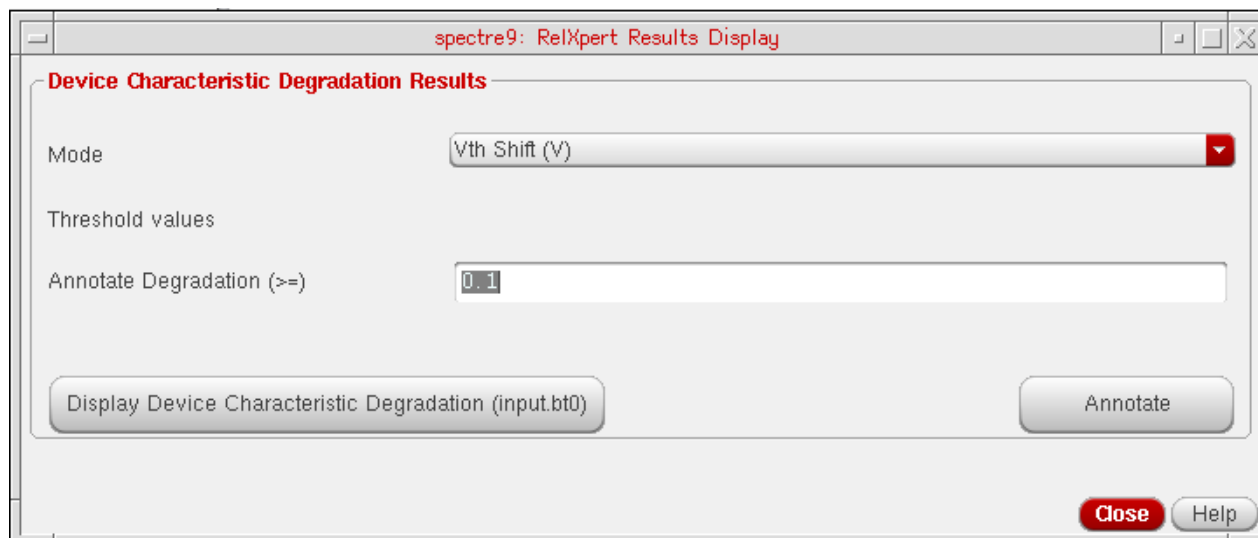
To view the Device Characteristic Degradation Results, do the following:

- ➔ In the ADE window, choose *Results – Reliability Data – Device Characteristic Degradation*.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

The Device Characteristic Degradation Results form appears displaying the device characteristic degradation such as V_{th} , G_m , I_{dlin} , I_{dsat} , and G_{ds} for the specified V_{dd} and age time period in the Aging time field.



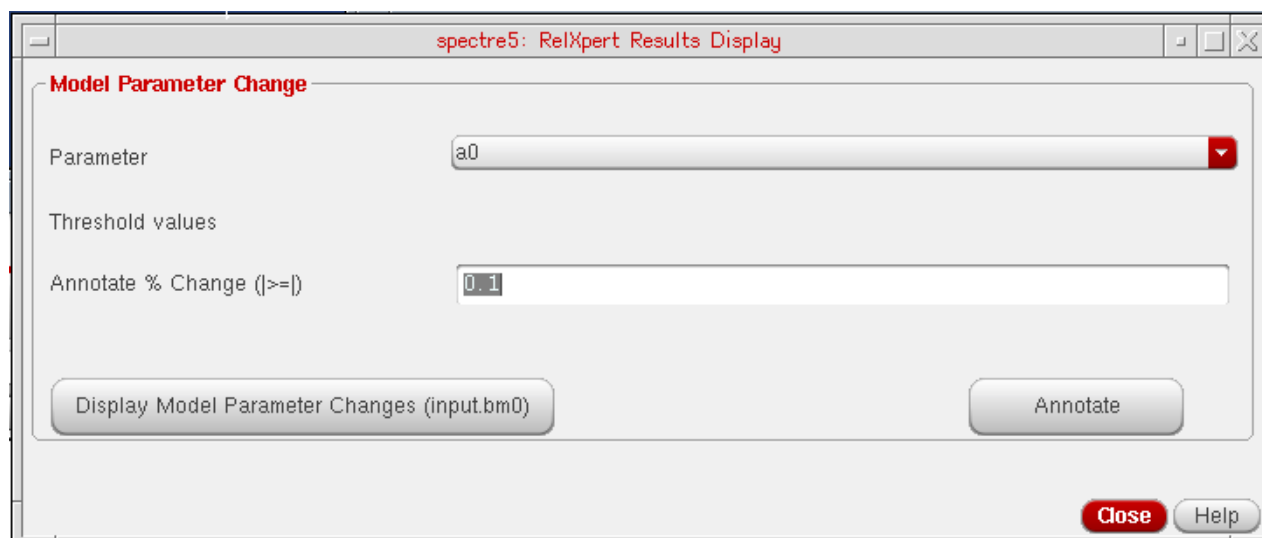
Note: This result is available only if you select the *Enable output device characteristic degradation* check box in the *Output Device Characteristic Degradation Setting* form. For more information about this form, see [Output Device Characteristic Degradation Settings](#) on page 629.

Model Parameter Changes Results

To view the Model Parameter Changes Results, do the following:

- ➔ In the ADE window, choose *Results – Reliability Data – Model Parameter Changes*.

The *Model Parameter Changes Results* form appears displaying the fresh and aged SPICE model parameter information.



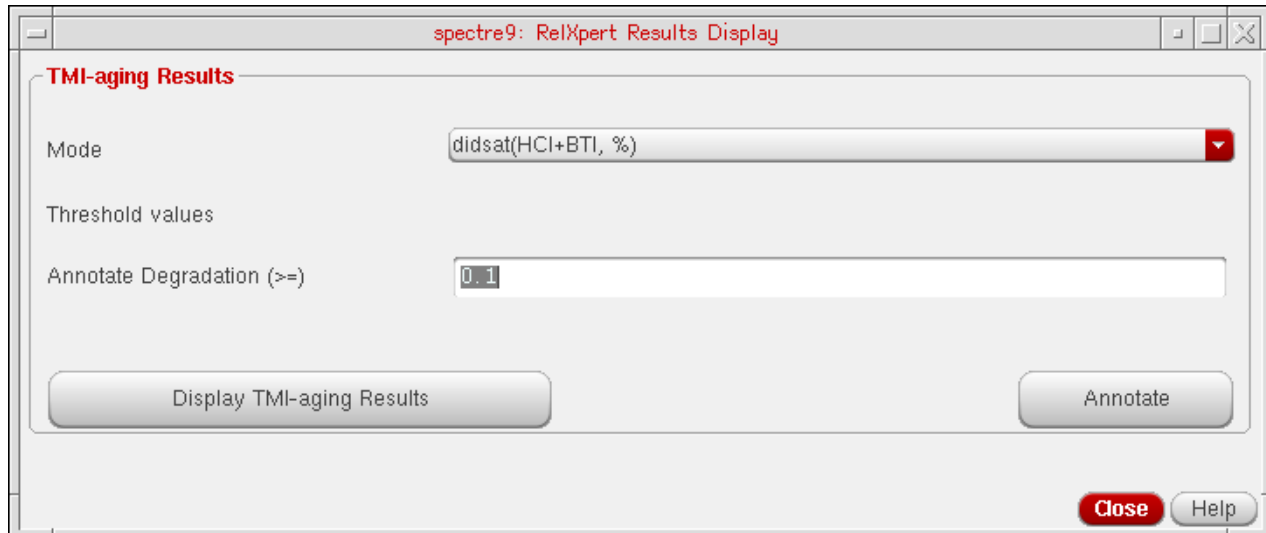
Note: This result is available only if you select the *Report model parameter changes* check box on the *Advanced* tab.

TMI-aging Results

To view the TMI-aging Results, do the following:

- ➔ In the ADE window, choose *Results – Reliability Data – TMI-aging*.

The *TMI-aging Results* form appears.



Viewing the Reliability Aged Netlist

The aged netlist can be used to view the degradation data generated by RelXpert Reliability Simulator.

Note: This option is available only if you select *RelXpert* as the *Simulator Mode* on the Reliability form.

To view the aged netlist, do the following:

- ➔ In the ADE window, choose *Results – Reliability Data – Aged netlist*.

The Reliability Simulator generated degradation netlist is displayed.

Annotating Simulation Results to the Schematic

To annotate the Device Lifetime and Degradation Results to the schematic, do the following:

1. In the ADE window, choose *Results – Reliability Data – Device Lifetime and Degradation Results*.

The The Device Lifetime and Degradation Results form appears. The threshold rules corresponding to the selected modes are displayed in the form.

2. Select the threshold rules based on which you want to annotate the results.

Virtuoso Analog Design Environment L User Guide

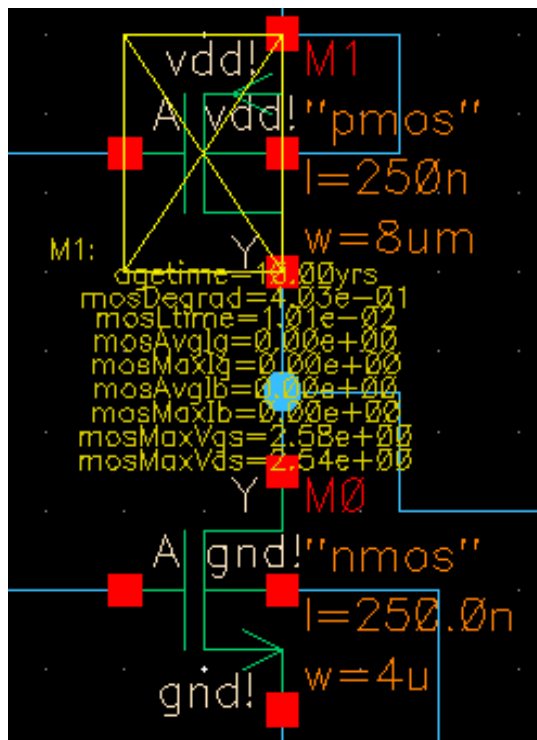
Using the Reliability Simulator Interface

The fields containing the default threshold values for the selected rules become available.

Note: Results are annotated based on the specified threshold values. For example, if the Lifetime (year) (\leq) field specifies a threshold value of 10, results are annotated only to those instances whose lifetime is less than or equal to 10 years.

3. Modify the threshold values for the selected rules, if required.
4. Click the `Annotate` button.

The instances to which the results are annotated are highlighted in yellow on the schematic.



Note: In a hierarchical design, a block is highlighted if results are annotated to an instance in that block. When you descend into the block, the instances on which the results are annotated are also highlighted.

Similarly, you can annotate other types of reliability results.

Reliability Options

You can specify the Reliability options in the Reliability form. The tabs in the form are described below:

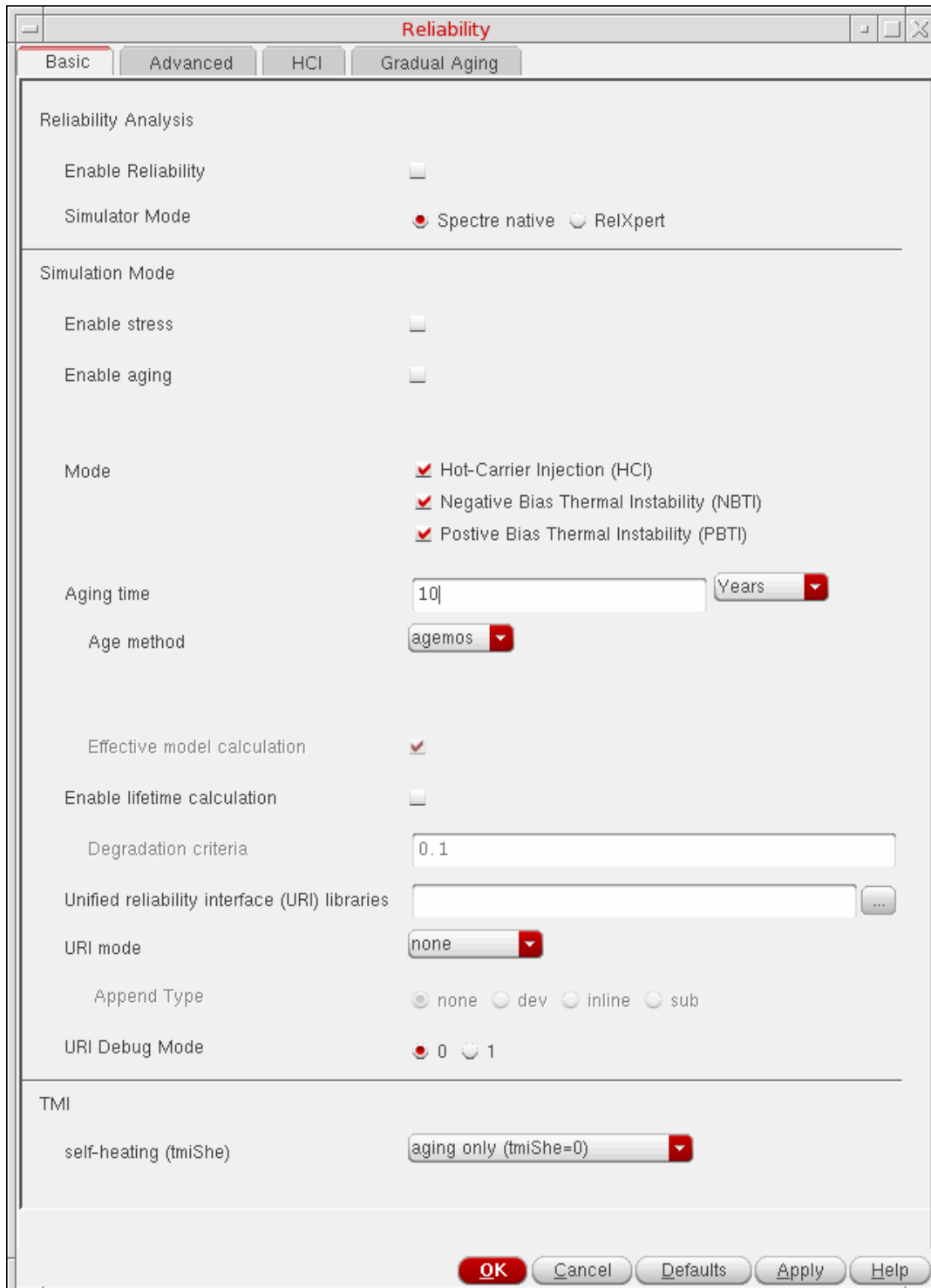
- [Basic Tab](#) on page 615
- [Advanced Tab](#) on page 619
- [HCI Tab](#) on page 624
- [Gradual Aging Tab](#) on page 626

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Basic Tab

The Basic tab contains options shown in the following figure.



Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

The options available on the Basic tab are described below:

Field	Description
Reliability Analysis	
<i>Enable Reliability</i>	If selected, enables the Reliability simulation. Before enabling Reliability, ensure that a reliability model is included and a transient analysis is set up.
<i>Simulator Mode</i>	
Spectre Native	If selected, reliability simulation is run using native Spectre. Note: <ul style="list-style-type: none">■ This mode is selected by default.■ Spectre native mode is only available in IC6.1.5 ISR6 and later releases.
RelXpert	If selected, reliability simulation is run using RelXpert.
Simulation Mode	
<i>Enable Stress</i>	If selected, enables the stress simulation. If the Enable Reliability and Enable Stress check boxes are selected, the simulation flow is fresh and stress simulation.
<i>Enable aging</i>	If selected, enables the aging simulation. If both Enable Reliability and Enable Aging check boxes are selected, the simulation flow is fresh and aging (end-of-life) simulation. Note: Aging simulation uses the output of stress simulation as input. Therefore, stress simulation must be run before aging simulation. If the stress result is not present before running the aging simulation, the simulator reports an error in the simulation output window.
RelXpert Options	
<i>Mode</i>	
Hot-Carrier Injection (HCI)	If selected, predicts transistor and circuit performance degradation due to HCI effects.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
Negative Bias Thermal Instability (NBTI)	If selected, predicts transistor and circuit performance degradation due to NBTI effects.
Positive Bias Thermal Instability (PBTI)	If selected, predicts transistor and circuit performance degradation due to PBTI effects.
<i>Aging time</i>	Specifies the aging time in years, days, hours, minutes or seconds. Only one unit of time is supported for a given simulation run.
<i>Age method</i>	<p>Specifies the aging method. The aging method can be one of the following:</p> <ul style="list-style-type: none"> ■ interp: Specifies the method as interpolation from aged model files. Valid values include linlin(linear-linear), linlog(linear-log), loglog(log-log). The default value is loglog ■ regres: Specifies the method of regression from aged model files. Valid values include linlin(linear-linear), linlog(linear-log), loglog(log-log). The default value is loglog ■ agemos: Specifies the agemos parameters for generating aged model parameters. <p>Note: agemos is selected by default.</p>
<i>Effective model calculation</i>	<p>If selected, includes effective parameters in the aged model card generation.</p> <p>Note: This option is disabled for Spectre native simulator mode.</p>
<i>Enable lifetime calculation</i>	If selected, calculates how long it will take for your circuit to degrade to a certain degradation percentage. The criteria can be in percentage or absolute shift.
<i>Degradation criteria</i>	Specifies the degradation criteria for lifetime calculation. The unit for the degradation criteria depends on the monitoring criteria used. For example with $[\Delta I_{dsat}] / (I_{dsat})$, a degradation criterion of 0.1 implies 10%.
<i>Unified reliability interface (URI) libraries</i>	Specifies URI model libraries to be used to run reliability simulation with URI. Click the browse button to select the model library files.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>URI mode</i>	<p>Specifies the URI mode. The URI mode can be one of the following:</p> <ul style="list-style-type: none">■ appendage: appends age values for all of the devices listed in the aged netlist file. If selected, the model name of aged netlist is replaced with a new user-defined model name and agelevel in URI.■ agemos: applies the same calculation used for the non-URI mode agemos method in the aged netlist file. <p>Note:</p> <ul style="list-style-type: none">■ If you select <code>none</code>, no URI mode is specified however the agemos mode is used by default.

<i>URI Debug Mode</i>	<p>Specifies the debug mode for URI library. The value can be 0 or 1.</p> <p>When the value of URI Debug Mode is 1, URI library prints the debug message.</p> <p>Default value is 0.</p>
-----------------------	--

TMI

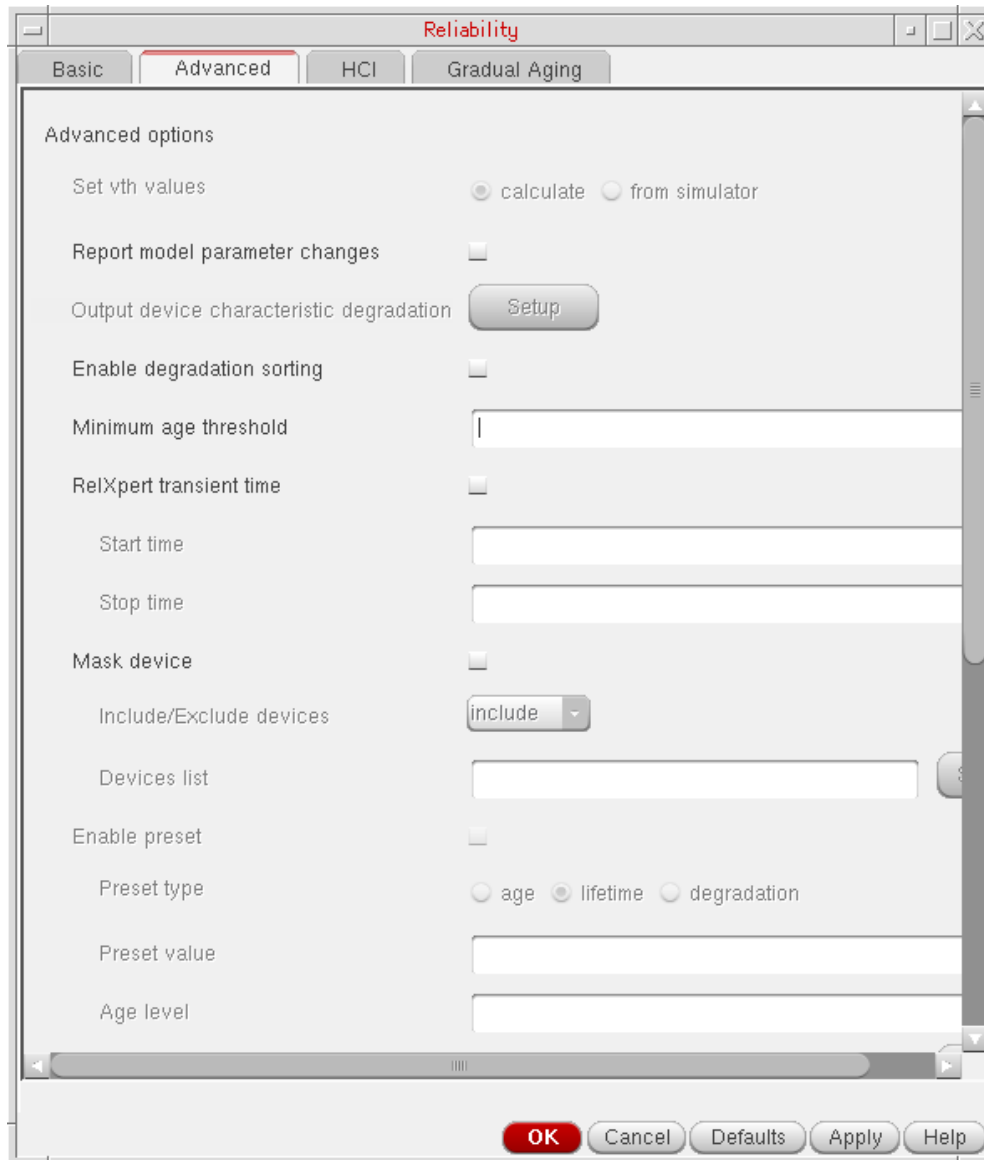
<i>self-heating (tmiShe)</i>	<p>Specifies the TMI aging or self-heating modes for the TMI flow. The TMI mode can be one of the following:</p> <ul style="list-style-type: none">■ aging only (tmiShe=0): enables only the TMI aging mode. This is the default option.■ self-heating only (tmiShe=1): enables only the TMI self-heating mode.■ aging + self-heating (tmiShe=2): enables both the TMI aging and self-heating modes. <p>Note: This option is available in IC616ISR8/ICADV12.1ISR10 release with MMSIM13.1ISR10 or above.</p>
------------------------------	--

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Advanced Tab

The Advanced tab contains options shown in the following figure.



The various options available on the Advanced tab are described in the following table.

Field	Description
Advanced Options	

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Set vth values</i>	<p>Specifies whether the vth values are calculated or retrieved from Spectre Simulator.</p> <p>Note: This options is not supported disabled for Spectre native simulator mode.</p>
<i>Report model parameter changes</i>	<p>If selected, enables tracking of aged parameters. If this option is used in the netlist, the fresh and aged parameters are saved to a .bm# file.</p> <p>The .bm# file is saved in the netlist directory.</p>
<i>Output device characteristic degradation</i>	<p>Click this button to specify the output device characteristic degradation settings. For more information, see Output Device Characteristic Degradation Settings on page 629.</p> <p>Note: This option is disabled for Spectre native simulator mode.</p>
<i>Enable degradation sorting</i>	<p>If selected, enables sorting of device degradation based on decending order.</p>
<i>Minimum age threshold</i>	<p>Sets the smallest age value for which degraded Spectre model parameters are calculated.</p>
<i>Relxpert transient time</i>	<p>If selected, enables you to specify the start and stop time (in nanoseconds) for the reliability transient simulation run. If the start and stop time is not specified, reliability transient simulation time is determined by the transient statement in the netlist file.</p>
<i>Start time</i>	<p>Specifies the start time of transient analysis for reliability simualtion.</p>
<i>Stop time</i>	<p>Specifies the stop time of transient analysis for reliability simulation.</p>
<i>Mask device</i>	<p>If selected, enables you to include or exclude specified devices for reliability simulation.</p>
<i>Include/Exclude devices</i>	<p>Specifies whether devices should be included or excluded for a simulation.</p>

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Devices list</i>	<p>Specifies the set of devices to be included or excluded for a simulation. The Select button enables you to specify the devices.</p> <p>To specify the devices, do the following:</p> <ol style="list-style-type: none">1. Click the Select button. The schematic window appears.2. Select one or more devices in the schematic.3. Press the ESC key when you are done. The selected devices are displayed in the <code>Devices list</code> field.
<i>Enable preset</i>	<p>If selected, enables you to preset the age, lifetime, or degradation value for devices in the netlist file.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Preset type</i>	<p>Enables you to select the preset type for a device, model, or block as one of the following:</p> <ul style="list-style-type: none">■ age■ lifetime■ degradation
<i>Preset value</i>	<p>Enables you to enter the value for the selected preset type.</p>
<i>Age level</i>	<p>Enables you to enter the agelevel values for HCI, NBTI, or PBTI modeling.</p>

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Block/Model/Device</i>	<p>Specifies the blocks, models, or devices on which the preset values are enabled.</p> <p>To specify the blocks or devices, do the following:</p> <ol style="list-style-type: none">4. Click the <code>Select</code> button. <p>The schematic window appears.</p> <ol style="list-style-type: none">5. Select one or more blocks or models or devices in the schematic. <ol style="list-style-type: none">6. Press the <code>Esc</code> key when you are done. <p>The selected blocks or devices are displayed in the <code>Block/Model/Device</code> field.</p>
<i>Dump age model</i>	<p>If selected, writes the degraded model to a file. By default, the degraded model is saved in the aged netlist file.</p>
<i>Age model file</i>	<p>Specifies the name of the file in which degraded model must be dumped.</p> <p>Type the name of the file in which you want the degraded model to be dumped. Alternatively, you may click the browse button to select an existing file.</p>
<i>Append device age</i>	<p>If selected, appends age value in the definition statements of all degraded devices in the netlist file.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Compact age model</i>	<p>If selected, generates a new model netlist in compact form that does not include any redundant models. The aged netlist file has the <code>*.p2</code> extension.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Modify netlist before aging starts</i>	<p>If selected, enables the user to review and change the netlist before the aging simulation is run.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Age model limit</i>	<p>If selected, specifies the number of aged models and how these can be quantified.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Total number</i>	<p>Lists the number of aged models to be generated.</p>
<i>Quantization method</i>	<p>Specifies the method based on which the aged models can be quantified. The quantization method can be either of the following:</p> <ul style="list-style-type: none">■ age: Interpolates the models in log domain.■ degradation: Interpolates the models by using linear scale.
<i>Include files</i>	<p>Includes the contents of the specified files in the input netlist file. Click the browse button to select the files. This is similar to the <code>-1</code> option of Spectre simulator.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Output method</i>	<p>Specifies the degradation calculation method that are used to calculate the degradation results for a MOSFET. The degradation results are generated in the <code>bt#</code> file.</p> <p>RelXpert supports three reliability models, namely HCI, NBTI, and PBTI. Therefore, the <code>bt#</code> file contains device degradation results based on these three models. The HCI and PBTI reliability models are used for calculating degradation results for NMOSFET, and the HCI and NBTI reliability models are used for calculating degradation results for PMOSFET. The different types of output method are:</p> <ul style="list-style-type: none">■ single: Uses the aged model card to calculate the degradation results. The results are generated in the <code>bt#</code> file. This is the default value.■ integ: Calculates the degradation results of HCI, NBTI, and PBTI reliability model separately. The HCI and NBTI degradation results are added to calculate the degradation results for PMOSFET, and the HCI and PBTI degradation results are added to calculate the degradation results for NMOSFET. The results are generated in the <code>bt#</code> file.

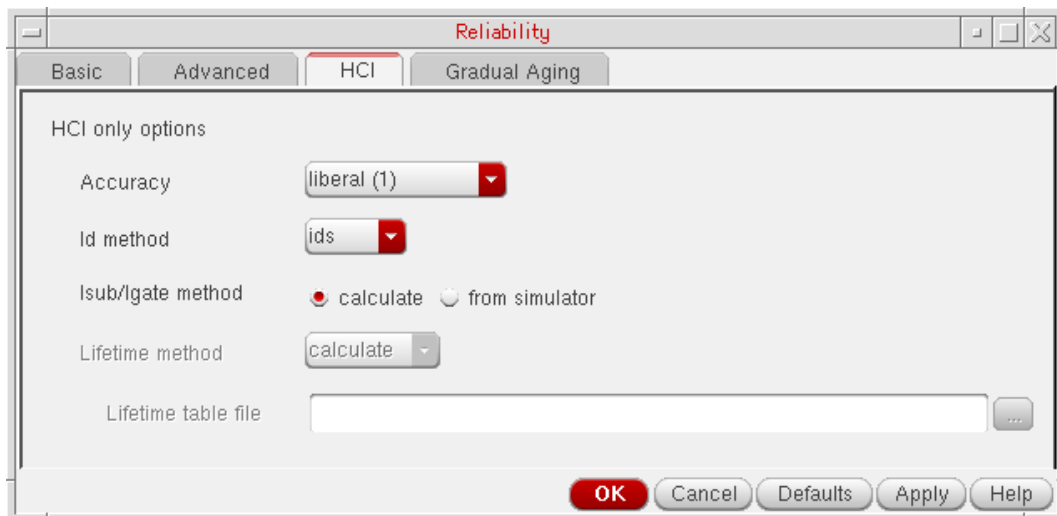
Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Additional arguments</i>	Enables you to specify additional control statements that cannot be specified in the Reliability form. Arguments entered here will appear as the same in the netlist generated. Note: This option is not supported and disabled for Spectre native simulator mode.

HCI Tab

The HCI tab contains options associated with HCI degradation, as shown in the following figure.



The various options available on the HCI tab are described in the following table.

Field	Description
HCI only options	

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Accuracy</i>	<p>Specifies the method used in RelXpert Reliability Simulator when performing integration and substrate current calculation as liberal (1) or conservative (2).</p> <ul style="list-style-type: none">■ If set to liberal (1), uses backward Euler integration. Sets $I_{sub}=0$ if $V_{gs} < V_{th}$.■ If set to conservative (2), uses trapezoidal integration. Calculates I_{sub} if $V_{gs} < V_{th}$. The conservative method is more accurate, but takes more time as compared to the liberal method.
<i>Id method</i>	<p>Specifies how RelXpert Reliability Simulator obtains the drain current (I_d) from the Spectre Simulator for performing the calculations during the RelXpert reliability simulation run.</p> <p>The following two types of drain current are available:</p> <ul style="list-style-type: none">■ <i>ids</i>: Specifies RelXpert Reliability Simulator to use static channel drain current.■ <i>idrain</i>: Specifies RelXpert Reliability Simulator to use dynamic drain current.■ <i>idstatic</i>: Specifies RelXpert Reliability Simulator to use static terminal drain current.
<i>Isub/Igate method</i>	<p>Specifies the method used for obtaining the gate terminal current of a MOSFET. Either of the following methods can be used:</p> <ul style="list-style-type: none">■ <i>calculate</i>: Calculates the substrate or gate terminal current by using the model parameters.■ <i>simulator</i>: Obtains the substrate or gate terminal current value from the Spectre output rawfile.

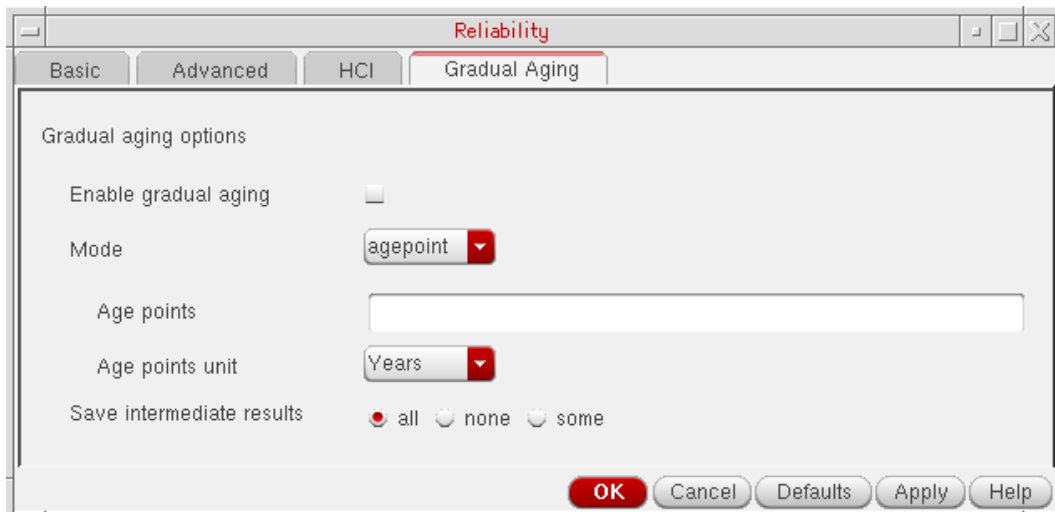
Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Lifetime method</i>	<p>Specifies the method for obtaining lifetime parameters for MOSFET Hot-Carrier simulation. Either of the following methods can be used:</p> <ul style="list-style-type: none">■ calculate: RelXpert Reliability Simulator calculates lifetime parameters.■ table: RelXpert Reliability Simulator looks up lifetime parameters and interpolates the results from the specified file. <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>
<i>Lifetime table file</i>	<p>Specifies the file to be used to interpolate the results. Click the browse button to select the file.</p> <p>Note: This option is not supported and disabled for Spectre native simulator mode.</p>

Gradual Aging Tab

The options available on the Gradual Aging tab enable you to run the reliability simulation which considers previous device degradation results. The Gradual Aging tab contains options shown in the following figure.



The options available on the Gradual Aging tab are described in the following table.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
Gradual aging options	
<i>Enable gradual aging</i>	If selected, enables gradual aging.
<i>Mode</i>	<p>Specifies the gradual aging flow. You can select one of the following modes:</p> <ul style="list-style-type: none">■ agepoint: You select this option when you want to specify age points to perform aging simulation.■ iteration: You select this option if you want to specify the iteration steps to perform aging simulation. <p>Note: iteration option is not supported for Spectre native simulator mode.</p> <ul style="list-style-type: none">■ agestep: You select this option if you want to specify the aging step type, start/stop time, and total steps to perform aging simulation.
Agepoint	
<i>Age points</i>	Specifies one or more age points for gradual aging simulation.
<i>Age points unit</i>	Specifies the unit as y, d, h, or m for age point values. You need not specify the unit if you have already specified it in the Age points field.
Iteration	
<i>Iteration number</i>	Specifies the step number in the aging simulation. This option is displayed if the iteration mode is selected.
Agestep	

Virtuoso Analog Design Environment L User Guide
Using the Reliability Simulator Interface

Field	Description
<i>Type</i>	<p>Specifies one of the following types of agestep:</p> <ul style="list-style-type: none"> ■ lin (linear): When you specify lin, the time step is calculated using the following formula. $(\text{stop_time} - \text{start_time}) / (\text{total_step} - 1)$ ■ log (logarithm): When you specify log, the time step is calculated using the following formula. $[\log(\text{stop_time}) - \log(\text{start_time})] / (\text{total_step} - 1)$ <p>This option is displayed if the <code>agestep</code> mode is selected.</p>
<i>Unit</i>	<p>Specifies the unit as Years, Days, Hours, Minutes, or Seconds. This option is displayed if the <code>agestep</code> mode is selected.</p>
<i>Start</i>	<p>Specifies the start time of rxprofile or gradual aging simulation. The default value is 0.0 for linear and 1.0 for logarithm. This option is displayed if you select the <code>agestep</code> mode.</p>
<i>Stop</i>	<p>Specifies the stop time of rxprofile or gradual aging simulation. This option is populated if you select the <code>agestep</code> mode.</p> <p>Note: If this time is greater than the Aging time specified in Basic tab, then Aging time is ignored and simulation continues till the Stop time specified.</p>
<i>Total step</i>	<p>Specifies the total number of steps of rxprofile or gradual aging simulation. Ensure that the value is greater than 1. This option is displayed if you select the <code>agestep</code> mode.</p>
<i>Save intermediate results</i>	<p>Saves the simulation results of intermediate steps. Either one of the following can be selected:</p> <ul style="list-style-type: none"> ■ all – saves simulation results for all intermediate steps. Note: This option is selected by default. ■ none – no simulation results are saved for all intermediate steps, only last aging time results are saved. ■ some – saves simulation results for the specified intermediate steps.

Reliability Options Not Supported In Spectre Native Simulator Mode

The following table lists the options on the Reliability form that are not supported in Spectre in Spectre native simulator mode.

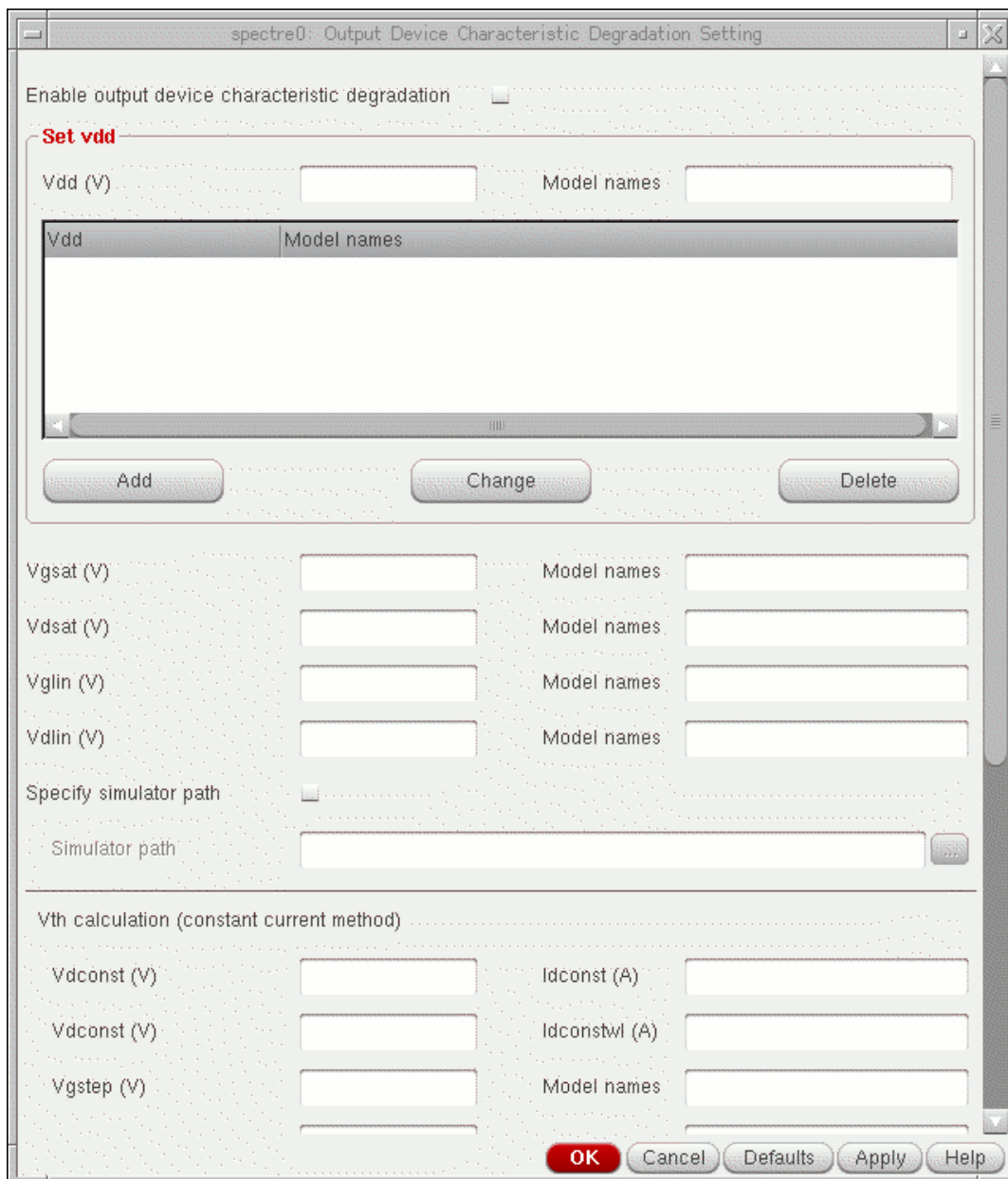
Tab	Unsupported Reliability Options
<i>Basic</i>	<ul style="list-style-type: none">■ Enable Aging■ Effective Model Calculation■ Unified reliability interface (URI) mode – appendage mode
<i>Advanced</i>	<ul style="list-style-type: none">■ Set vth Values■ Output Device Characteristic Degradation■ Enable Preset■ Dump Age Model■ Append Device Age■ Compact Age Model■ Modify Netlist Before Aging Starts■ Age Model Limit■ Include Files■ Additional Arguments
<i>HCI</i>	<ul style="list-style-type: none">■ Lifetime Method■ Lifetime Table File
<i>Gradual Aging</i>	<ul style="list-style-type: none">■ Iteration Mode

Output Device Characteristic Degradation Settings

The Output Device Characteristic Degradation Settings form contains options shown in the following figure.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface



The options available on the Output Device Characteristic Degradation Settings form are described in the following table.

Field	Description
<i>Enable output device characteristic degradation</i>	Requests RelXpert Reliability Simulator to generate device degradation (gds, gm, Idlin, Idsat, Vth degradation) for the specified age time period.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
Set vdd	
<i>Vdd(V)</i>	Specifies the operation voltage for the circuit or target model.
<i>Model names</i>	Specifies the model used in the .output_device_degrad file for the adjacent field. This field supports wildcards. For example, n* can be specified instead of nmos1.
<i>Vdd</i>	Lists the added Vdd values in the table.
<i>Model names</i>	Lists the model names specified for the added Vdd values.
<i>Add</i>	Adds a new model specifying the operation voltage. Multiple model names may be added.
<i>Change</i>	Updates the modifications after a particular model in the table is modified.
<i>Delete</i>	Deletes the selected row.
<i>Vgsat(V)</i>	Specifies Vgs value for Idsat measurement. The path to the corresponding model files must be specified in the Model names field.
<i>Vdsat(V)</i>	Specifies Vds value for Idsat measurement. The path to the corresponding model files must be specified in the Model names field.
<i>Vglin(V)</i>	Specifies Vgs value for Idlin/Vt/Gm measurement. The path to the corresponding model files must be specified in the Model names field.
<i>Vdlin(V)</i>	Specifies Vds value for Idlin/Vt/Gm measurement. The path to the corresponding model files must be specified in the Model names field.
<i>Specify simulator path</i>	If selected, enables or disables the specified simulator path.
<i>Simulator path</i>	Specifies the full path for the simulator that is used during aging simulation.
Vth calculation (constant current method)	
<i>Vdconst(V)</i>	Specifies the constant drain voltage for calculating the threshold voltage by using the Vt_const_current method.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Idconst(A)</i>	Specifies the constant drain current for calculating the threshold voltage by using the <i>Vt_const_current</i> method.
<i>Idconstwl(A)</i>	Specifies the constant drain current related to the transistor's width(W) and length(L).
<i>Vgstep(V)</i>	Specifies the gate voltage (Vg) sweep step size for calculating the device degradation(Vth, ids, idling, gds, and gm degradation). The path to the corresponding model files must be specified in the Model names field.
<i>Vsconst(V)</i>	Specifies the constant source terminal voltage for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Nmos vbconst(V)</i>	Specifies the constant substrate terminal voltage for NMOSFET for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Pmos vbconst(V)</i>	Specifies the constant substrate terminal voltage for PMOSFET for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Nmos vgsweep start (V)</i>	Specifies the start point of the sweep voltage of the NMOSFET gate terminal for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Pmos vgsweep start (V)</i>	Specifies the start point of the sweep voltage of the PMOSFET gate terminal for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Nmos vgsweep end (V)</i>	Specifies the stop point of the sweep voltage of the NMOSFET gate terminal for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
<i>Pmos vgsweep end (V)</i>	Specifies the stop point of the sweep voltage of the PMOSFET gate terminal for calculating the threshold voltage degradation. The path to the corresponding model files must be specified in the Model names field.
<i>Use source current</i>	If selected, enables or disables the drain current to calculate the degraded threshold voltage with vt_const_current method.
<i>Use drain current</i>	If selected, enables or disables the source current to calculate the degraded threshold voltage with vt_const_current method.
<i>Use spectre for const current</i>	If selected, enables or disables the const current method to calculate the threshold degradation value.

Reliability Results Display

The Results Display form enables you to view various results, specify threshold rules, and annotate the results. The options available on this form are described in the following table.

Field	Description
<i>Mode</i>	Lets you select the RelXpert mode for which you want to annotate the results: <ul style="list-style-type: none">■ HCI■ NBTI■ PBTI■ URI The threshold rules corresponding to the selected modes are displayed in the form.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Field	Description
Threshold rules	<p>Enables you to select the threshold rules based on which results are annotated. The fields containing the default threshold values for the threshold rules are enabled, as and when the rules are selected.</p> <p>Note: Results are annotated based on the specified threshold values. For example, if the Lifetime (year) (<=) field specifies a threshold value of 10, results will be annotated only on those instances in the schematic whose lifetime is less than or equal to ten years.</p>
<i>Degradation</i>	If selected, specifies the degradation value for all the modes.
<i>Lifetime</i>	If selected, specifies the lifetime calculation in years.
<i>Average gate current</i>	If selected, specifies the average gate current in amperes for the HCI mode.
<i>Maximum gate current</i>	If selected, specifies the maximum gate current in amperes for the HCI mode.
<i>Average substrate current</i>	If selected, specifies the average substrate current in amperes for the HCI mode.
<i>Maximum substrate current</i>	If selected, specifies the maximum substrate current in amperes for the HCI mode.
<i>Maximum vgs value</i>	If selected, specifies the maximum gate-source voltage rating for the NBTI/PBTI mode.
<i>Maximum vds value</i>	If selected, specifies the maximum drain-source voltage rating for the NBTI/PBTI mode.
<i>Display Device Lifetime and Degradation</i>	Displays the lifetime and degradation values for all degraded devices.
<i>Display Device Characteristic Degradation</i>	Displays the device degradation for the age time period specified using the Aging time option.
<i>Display Model Parameter Changes</i>	Displays fresh and aged parameter information.
<i>Annotate</i>	Annotates the simulation results to instances schematic that conform to the specified threshold rules.

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Virtuoso Analog Design Environment L User Guide

Using the Reliability Simulator Interface

Environment Variables

This appendix describes public environment variables that control the characteristics of the Analog Design Environment (ADE). You can customize the operation and behavior of Analog Design Environment products by changing the value of a particular environment variable.

This appendix lists environment variables belonging to the following products:

- [Calculator](#)
- [Reliability Analysis](#)
- [Distributed Processing](#)
- [Spectre](#)
- [ADE Simulation Environment](#)
- [ADE XL](#)
- [SpectreVerilog \(IC6.1.6 only\)](#)
- [HspiceD](#)
- [AMS](#)
- [Ultrasim](#)
- [UltraSimVerilog \(IC6.1.6 only\)](#)

The appendix also provides you a list of deprecated environment variables.

Calculator

mode

This variable sets the mode for creating expressions.

To set this variable in the .cdsenv add the line:

```
calculator mode cyclic "algebraic"
```

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("calculator" "mode" `cyclic "algebraic")
```

Variable Type	cyclic
Default Value	RPN
Acceptable Values	{RPN, algebraic}
GUI Equivalent	<i>Calculator – Options</i>

uimode

This variable sets the mode of operation for the calculator.

To set this variable in the .cdsenv, add the line:

```
calculator uimode cyclic "RF"
```

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("calculator" "uimode" `cyclic "RF")
```

Variable Type	cyclic
Default Value	standard
Acceptable Values	{standard, RF}
GUI Equivalent	<i>Calculator</i>

eval

This field is set to evaluate the contents of a calculator buffer automatically. This is available only for the RPN mode.

To set this variable in the `.cdsenv`, add the line:

```
calculator eval `boolean t
```

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("calculator" "eval" `boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	<i>Calculator</i>

dstack

This field is set to display the contents of the stack. This is available only for the RPN mode.

To set this variable in the `.cdsenv`, add the line:

```
calculator dstack `boolean t
```

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("calculator" "dstack" `boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	<i>Calculator</i>

Reliability Analysis

spectre_analysis_reliability

If this variable is set, the *RelXpert* option is set as default Simulation Mode instead of *Spectre Native* in the *Simulation – Reliability – Setup* form.

To set this variable in the .cdsinit file or ciw, use the call
`putprop _amsUISimFeatures nil 'spectre_analysis_reliability`

Variable Type	boolean
Default Value	t
Acceptable Values	{nil t}
GUI Equivalent	<u><i>Simulation – Reliability – Setup</i></u>

relxpert_gradual_aging

If this variable is set, the *Gradual Aging* tab is removed from the *Simulation – Reliability – Setup* form.

To set this variable in the .cdsinit file or ciw, use the call
`putprop _amsUISimFeatures nil 'relxpert_gradual_aging`

Variable Type	boolean
Default Value	t
Acceptable Values	{nil t}
GUI Equivalent	<u><i>Simulation – Reliability – Setup</i></u>

Distributed Processing

autoJobSubmit

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

To set this variable in the `.cdsenv`, add the line:

```
asimenv.distributed autoJobSubmit `boolean nil
```

To set this variable in the `.cdsinit` file or `ciw`, use the call

```
envSetVal("asimenv.distributed" "autoJobSubmit" `boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	<u>Setup – Simulator/Directory/Host</u>

showMessages

If this variable is set to a non-nil value, a message is displayed in the CIW or OCEAN terminal on the completion of a job.

To set this variable in the `.cdsenv`, add the line:

```
asimenv.distributed showMessages `boolean nil
```

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "showMessages" `boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	none

queueName

The variable sets the default queue name. If unspecified, the system default is used. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsenv, add the line:

```
asimenv.distributed queueName `string "myqueue"
```

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "queueName" `string "myqueue")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	<i>Job Submit Form</i>

hostName

This variable sets the default host name. If unspecified, the host is selected automatically. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsenv, add the line:

```
asimenv.distributed hostName `string "host"
```

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "hostName" `string "host")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	<i>Job Submit Form</i>

startTime

This variable sets the default start time for a job (in 24hour format). If unspecified, the job executes immediately. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "startTime" `string "23:11")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
GUI Equivalent	<i>Job Submit Form</i>

startDay

This variable sets the default start day for a job. If the start day is set as `today`, then the job will always run on the same day it is submitted.

To set this variable in the .cdsinit file or ciw, use the call

```
envSetVal("asimenv.distributed" "startDay" `cyclic "Wednesday")
```

Variable Type	cyclic
Default Value	today
Acceptable Values	{today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday }
GUI Equivalent	<i>Job Submit Form</i>

expTime

This variable sets the default expiration time for a job (in 24 hour format). If unspecified, the expiration time is based on the value of the *timeLimit* variable. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "expTime" 'string "00:43")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
GUI Equivalent	<i>Job Submit Form</i>

externalServer

If this variable is set to a non-nil value, the job server is started remotely.
If this variable is set to a non-nil value, the job server is started remotely.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "externalServer" 'boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

expDay

This variable sets the default expiration day for a job. If the expiration day is set as `today`, then the job will always run on the same day it is submitted. For details, refer to the [Submitting a Job](#) section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "expDay" 'cyclic "Friday")
```

Variable Type	<code>cyclic</code>
Default Value	<code>today</code>
Acceptable Values	{ <code>today</code> , <code>Sunday</code> , <code>Monday</code> , <code>Tuesday</code> <code>Wednesday</code> , <code>Thursday</code> , <code>Friday</code> , <code>Saturday</code> }
GUI Equivalent	<i>Job Submit Form</i>

timeLimit

This variable sets the default time limit for a job. If the time limit is set to `none`, then no time limit is imposed. If unspecified, then expiration time is based on value of `expTime` and `expDay` variables. For details, refer to the [Submitting a Job](#) section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "timeLimit" 'cyclic "5 minutes")
```

Variable Type	<code>cyclic</code>
Default Value	<code>none</code>
Acceptable Values	{ <code>unspecified</code> , <code>none</code> , <code>5 minutes</code> , <code>15 minutes</code> , <code>30 minutes</code> , <code>1 hour</code> , <code>3 hours</code> , <code>6 hours</code> , <code>12 hours</code> , <code>1 day</code> , <code>2 days</code> , <code>3 days</code> , <code>5 days</code> , <code>10 days</code> }
GUI Equivalent	<i>Job Submit Form</i>

emailNotify

If this variable is set to a non-nil value, an e-mail notification is provided, following job termination. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "emailNotify" 'boolean nil )
```

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
GUI Equivalent	<i>Job Submit Form</i>

mailTo

This variable sets the default list of users who will receive job termination notification e-mail. If unspecified and if *emailNotify* is t, then the default value is the user's ID. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "mailTo" 'string  
"userId123@cadence.com")
```

Variable Type	string
Default Value	""
Acceptable Values	Any Valid Id String Value.
GUI Equivalent	<i>Job Submit Form</i>

logsInEmail

If this variable is set to a non-nil value, `stdout` and `stderr` logs will be included in the termination E-mail.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "logsInEmail" 'boolean t)
```

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
GUI Equivalent	none

stateFile

This variable sets the filename containing the job server's state.

This variable sets the filename containing the job server's state.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "stateFile" 'string "myStateFile")
```

Variable Type	string
Default Value	~/ .adpState
Acceptable Values	Any String Value
GUI Equivalent	none

daysBeforeExpire

Specifies the number of days after which terminated jobs will be deleted from the job server.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "daysBeforeExpire" 'int 6)
```

Variable Type	int
Default Value	3
Acceptable Values	Any String Value
GUI Equivalent	none

block

If this variable is set to a non-nil value, the process is blocked until the job has completed.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "block" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

copyMode

If this variable is set to a non-nil value, the input data for the job is copied to `/tmp` on the execution host, the job is run there locally (without network read/write), and the output data is copied back to the submission host.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "copyMode" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

copyModeDir

Specifies the directory relative to the execution host, that will be used for setting up the working directory of a copy mode job.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "copyModeDir" 'string "dirname")
```

Variable Type	string
Default Value	<code>/tmp</code>
Acceptable Values	Any string value
GUI Equivalent	none

loginShell

Specifies the login shell for the job. If it is specified as `none` then the users local environment is copied over to the execution host and used as the jobs environment.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "loginShell" 'cyclic "csh")
```

Variable Type	cyclic
Default Value	none
Acceptable Values	{none, csh, ksh, sh}
GUI Equivalent	none

numOfTasks

Specifies the default number of tasks a job should be broken into. This is used by the Monte Carlo tool. If zero, then the number of tasks is based on queue and/or host settings.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.distributed" "numOfTasks" 'int 5)
```

Variable Type	int
Default Value	0
Acceptable Values	Any Integer Value
GUI Equivalent	none

jobArgsInOceanScript

Indicates job arguments that should be added to run commands when an OCEAN script is generated.

```
envSetVal("asimenv.distributed" "jobArgsInOceanScript" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

puttogetherqueue

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "puttogetherqueue" 'string  
"queueName")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	none

copyNetlist

Specifies whether the netlist directory needs to be copied from the execution host to the submission host. This may be required if during simulation, some files are generated under the netlist directory.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "copyNetlist" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

mailAllLogs

Sends out a mail after completion of all the tasks and each individual task (when set to t) .

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.distributed" "mailAllLogs" 'boolean t)
```

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

drmsCommandList

Enables you to specify multiple DRMS commands in the *command* drop-down list box of the *Virtuoso Analog Distributed Processing option Job Submit* form.

The DRMS command specified using the `drmsCommand` environment variable is set as the default value for the *command* drop-down list box. If the `drmsCommand` environment variable is not set, then the first command specified using the `drmsCommandList` variable is set as default.

For more information, see [*Submitting Jobs using the Command Mode*](#).

To set this variable in the `.cdsinit` file or CIW for specifying multiple DRMS commands, use the following call:

```
(envSetVal "asimenv.distributed" "drmsCommandList" 'string "bsub -R  
\\"OSNAME==Linux && OSREL==EE80\\";bsub -R \\"OSNAME==Linux && OSREL==EE60\\";bsub -R  
\\"OSNAME==Linux && OSREL==EE50 && SFIARCH==EM64T && OSBIT==64\"")
```

To set this variable in the `.cdsenv` file for specifying multiple DRMS commands, use the following call:

```
asimenv.distributed drmsCommandList string "bsub -R \\"OSNAME==Linux &&  
OSREL==EE80\\";bsub -R \\"OSNAME==Linux && OSREL==EE60\\";bsub -R \\"OSNAME==Linux &&  
OSREL==EE50 && SFIARCH==EM64T && OSBIT==64\""
```

Important points to note:

- The DRMS commands specified in the example above are the resource strings generated from the farm. You can define multiple resource strings using this environment variable with each string separated by a semi-colon (;).
- Only the first ten commands specified using the `drmsCommandList` variable are added to the *command* drop-down list box.

Variable Type	string
Default Value	" "
Acceptable Values	DRMS commands in correct syntax
GUI Equivalent	None

Virtuoso Analog Design Environment L User Guide

Environment Variables

setupFunction

Triggers the user-defined SKILL function to update the values in the *Virtuoso Analog Distributed Processing option Job Submit* form.

The SKILL function should return the values in a list of key-value format, such as `?drmsCommand command ?queueName myQueue`.

The user-defined SKILL function can return the following values:

Value	Data Type	Description
<i>queueName</i>	string	The queue in which the job should be submitted.
<i>host</i>	string	The name of the machine on which the job should be launched.
<i>drmsCommand</i>	string	The DRMS command to submit the job.
<i>email</i>	string	The e-mail addresses to which the e-mail should be sent after the job terminates.
<i>dependentOn</i>	string	The job dependency list.
<i>block</i>	Boolean	The blocking mode. Specifies whether the analog design environment or OCEAN should be blocked until all the jobs have completed.
<i>tasks</i>	integer	The number of tasks.
<i>lsfResourceStr</i>	string	The additional resource requirements for the job, such as <code>mem</code> (available memory), <code>swp</code> (available swap space), and <code>pg</code> (paging rate).
<i>lsfLicenseProject</i>	string	The name of the LSF license project.
<i>lsfNoOfProcessors</i>	string	The number of parallel processors to be used to run the submitted job in LSF.
<i>sgcNoOfProcessors</i>	string	The number of parallel processors to be used to run the submitted job in SGE.
<i>sgcSoftResourceStr</i>	string	The requirements for soft resources in SGE.
<i>sgcHardResourceStr</i>	string	The requirements for hard resources in SGE.
<i>sgcPriority</i>	string	The priority for the job submitted in SGE.

Virtuoso Analog Design Environment L User Guide

Environment Variables

<i>sgeParallelEnvName</i>	string	The name of a parallel environment in SGE.
<i>startTime</i>	string	The start time of the job in <code>?time</code> and <code>?day</code> format. <code>?time</code> is in 24-hour format. For example, 05:40, 15:30, or 17:25. <code>?day</code> has the following acceptable values: today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.
<i>termTime</i>	string	The time after which the submitted job should be terminated in the <code>?time</code> and <code>?day</code> format. <code>?time</code> is in 24-hour format. For example, 05:40, 15:30, or 17:25. <code>?day</code> has the following acceptable values: today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.
<i>jobName</i>	string	The name of the job to be submitted.

You can create the user-defined function named `myCustomSetup` as follows:

```
procedure (myCustomSetup ()
list (?drmsCommand "bsub -R \"OSNAME==Linux && OSREL==EE50\"" ?jobName "myJobName")
)
```

You can dynamically redefine the `myCustomSetup` function in CIW as follows:

```
myCustomSetup
procedure (myCustomSetup ()
list (?drmsCommand "bsub -R \"OSNAME==Linux && OSREL==EE50\"" ?startTime "12:00"
?jobName "myJobNameStart")
)
=>function myCustomSetup redefined
myCustomSetup
```

To set this variable in the `.cdsenv` file to trigger the above function, use the following call:

```
asimenv.distributed setupFunction 'string "myCustomSetup"
```

To set this variable in the `.cdsinit` file or CIW to trigger the function, use the following call:

Virtuoso Analog Design Environment L User Guide

Environment Variables

```
envSetVal("asimenv.distributed" "setupFunction" 'string "myCustomSetup")
```

Variable Type	string
Default Value	" "
Acceptable Values	SKILL function name
GUI Equivalent	None

Spectre

ac_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for AC analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.checkOpts" "ac_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

assert_severity_default

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when the device checks are violated.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.checkOpts" "assert_severity_default" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Device Check Specification</u></i>

dc_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC sweep analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.checkOpts" "dc_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

dcOp_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.checkOpts" "dcOp_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

tran_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the transient analysis.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.checkOpts" "tran_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

printCdfParamForTopCell

The CDF parameters of the top-level schematic are printed in the netlist with the top-level schematic if you set this environment variable to `t` and select the *Set Top Circuit as Subcircuit* option in the *Environment Options* form.

To set this environment variable in the `.cdsinit` file or `CIW`, use the following call:

```
envSetVal("spectre.envOpts" "printCdfParamForTopCell" 'boolean t)
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts printCdfParamForTopCell 'boolean t
```

Variable Type	Boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>None</i>

emirSumList

Defines the default values for the options in the *EMIR Analysis Setup* form.

For a list of values and their options that can be set using this environment variable, see the *EMIR Flow Setup* section in *Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emirSumList" 'string "")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList string ""
```

Variable Type	string
Default Value	""
Acceptable Values	A list of (option value) pair with each option separated by space
GUI Equivalent	<i>None</i>

emTechFile

Specifies the technology file containing the EM current limits per layer or via.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emTechFile" 'string "")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList string ""
```

Variable Type	string
Default Value	""
Acceptable Values	Filename containing the EM current limits per layer. The formats supported are <code>emdatafile</code> , <code>qrctechfile</code> , and <code>ictfile</code> .
GUI Equivalent	<i>None</i>

emirEnable

Enables the EMIR analysis.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emirEnable" 'boolean t)
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList 'boolean t
```

Variable Type	Boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>None</i>

nportirfiledir

This environment variable controls the location of the cached impulse response files. By default, these files are saved in following directory: `/home/<username>/.cadence/mmsim`. When you specify a directory path with this environment variable or in the `nportirfiledir` field on the GUI, the impulse response files are cached in that directory.

Note: You can specify a shared directory only if all users who can access it have write permissions on it.

To set this variable in the `.cdsinit` file or ciw, use the call:

```
envSetVal("spectre.opts" "nportirfiledir" `string "filePath")
```

Variable Type	string
Default Value	""
Acceptable Values	path of cached impulse response file
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Simulator Options</u></i>

Virtuoso Analog Design Environment L User Guide

Environment Variables

save

This variable selects signals to be saved.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "save" `string "all")
```

Variable Type	string
Default Value	allpub
Acceptable Values	none,selected,lvlpub,allpub,all
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Select signals to output (save)</i>

outputParamInfo

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "outputParamInfo" `boolean nil )
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Save output parameters info</i>

modelParamInfo

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "modelParamInfo" `boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Save model parameters info</i>

pwr

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "pwr" `string "all")
```

Variable Type	string
Default Value	""
Acceptable Values	none, total, devices, subckts, all
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Select power signals to output (pwr)</i>

useprobes

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "useprobes" `string "no")
```

Variable Type	string
Default Value	""
Acceptable Values	yes, no
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Select AC terminal currents (useprobes)</i>

subcktprobelvl

This variable is used to control the calculation of terminal currents for subcircuits. Current probes are added to the terminals of each subcircuit (up to subcktprobelvl deep).

Variable Type	string
Default Value	""
Acceptable Values	--
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Set subcircuit probe level (subcktprobelvl)</i>

nestlvl

This variable is used to save groups of signals as results and when signals are saved in subcircuits. The nestlvl parameter also specifies how many levels deep into the subcircuit hierarchy you want to save signals.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "nestlvl" `string "2")
```

Variable Type	string
Default Value	""
Acceptable Values	--
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Set level of subcircuit to output (nestlvl)</i>

elementinfo

This variable specifies if input parameters for instances of all components are saved.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "elementInfo" `boolean nil )
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Save element info</i>

saveahdlvars

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

```
Saveahdl options saveahdlvars=all
```

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "saveahdlvars" `string "all")
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Save AHDL variables (saveahdlvars)</i>

currents

The currents parameter of the options statement computes and saves terminal currents. Use it to create settings for currents that apply to all terminals in the netlist.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.outputs" "currents" `string "nonlinear")
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all, nonlinear
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Save Options</u> – Select device currents (currents)</i>

setEngNotation

Specifies how the instance parameters in the netlist are printed. If set to `t`, the instance parameters in the netlist are printed in pure engineering notation. If set to `nil`, the instance parameters are printed in their default notation.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "setEngNotation" `boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

switchViewList

This variable is used to define the *Switch View List* field. This is a list of the views that the software switches into when searching for design variables.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "switchViewList" `string "schematic  
spectre")
```

Variable Type	string
Default Value	"spectre cmos_sch cmos.sch schematic veriloga"
Acceptable Values	view names, separated by spaces.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Switch View List.</i>

stopViewList

This variable is used to define the *Stop View List* option. This is a list of views that identify the stopping view to be netlisted.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "stopViewList" `string "spectre  
verilog")
```

Variable Type	string
Default Value	"spectre"
Acceptable Values	view names separated with spaces.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Stop View List</i>

autoDisplay

This variable is used to set/reset the *Automatic output log* option. When on, the output log opens and displays simulator messages as they are generated.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "autoDisplay" `boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Automatic output log</i>

stimulusFile

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "stimulusFile" `string "./file")
```

Variable Type	string
Default Value	""
Acceptable Values	unix path
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Simulation Files Setup</u> – Stimulus File</i>

includePath

Use this field for relative filenames. The simulator resolves a relative filename by first searching in the directory where the file is located. Subsequently, it searches for the file in each of the directories specified by the include path, from left to right.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "includePath" `string "./dir1 ../dir2")
```

Variable Type	string
Default Value	""
Acceptable Values	unix directories, separated with spaces.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Simulation Files Setup</u> – include Path</i>

modelFiles

Use this field for adding the default model files.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "modelFiles" `string "./models/  
model1.scs ./models/model2.scs")
```

To disable modelFiles, use a # sign as shown in the example below:

```
envSetVal("spectre.envOpts" "modelFiles" `string "#;./models/  
model1.scs ./models/model2.scs")
```

Here, the model1.scs file will be disabled.

Variable Type	string
Default Value	""
Acceptable Values	list of paths to model files.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Model Library Setup</u></i>

analysisOrder

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "analysisOrder" `string "tran ac dc")
```

Variable Type	string
Default Value	""
Acceptable Values	Names of analysis in the order desired
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Analysis Order</i>

paramRangeCheckFile

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "paramRangeCheckFile" `string "./  
param.file")
```

Variable Type	string
Default Value	""
Acceptable Values	path of the file
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Parameter Range Checking File</i>

printComments

Prints the name mapping of the terminals in the netlist as comments.

You can print the mapping of the schematic terminal name with the netlist terminal name for the subcircuits and the mapping of schematic device names with simulator devices names by setting the value of the first toggle to `t`.

For each subcircuit, you can print the connection of each terminal with the net it is connected to by setting the second toggle to `t`.

To set this variable in the `.cdsinit` file or CIW for printing comments for name mapping and subcircuit port connection, use the following call:

```
envSetVal("spectre.envOpts" "printComments" 'toggle '(t t))
```

The following figure shows the comments for name mapping and the subcircuit port connections printed in the netlist.

```
// res Instance R0 = spectre device R0
R0 (net5 0) resistor r=10K

// res Instance R1 = spectre device R1
R1 (net5 out) resistor r=20K

// idc Instance I2 = spectre device I2
I2 (net6 0) isource dc=500u type=dc

// vsin Instance V2 = spectre device V2
V2 (vin 0) vsource mag=1 type=sine sinedc=0 ampl=50m freq=1M

// supply Instance I1 = spectre device I1
// Instance of Lib: training, Cell: supply, View: schematic

// Port Connection: Instance I1 of supply
// VDD(vdd!) VSS(vss!)
I1 (vdd! vss!) supply VDD=5 VSS=-5

// amplifier Instance I0 = spectre device I0
// Instance of Lib: training, Cell: amplifier, View: schematic

// Port Connection: Instance I0 of amplifier
// inm(net5) inp(vin) iref(net6) out(out) inh_bulk_n(0)
I0 (net5 vin net6 out 0) amplifier
```

Comments for
Name Mapping

Subcircuit Port
Connections

Virtuoso Analog Design Environment L User Guide

Environment Variables

To set this variable in the `.cdsinit` file or CIW for printing comments for only name mapping, use the following call:

```
envSetVal("spectre.envOpts" "printComments" 'toggle '(t nil))
```

The following figure shows the comments for name mapping in the netlist.

```
// Library name: training
// Cell name: amplifier
// View name: schematic
// terminal mapping: inm      = inm
//                   inp      = inp
//                   iref     = iref
//                   out      = out
//                   [:@bulk_n:%:gnd!] = inh_bulk_n
subckt amplifier inm inp iref out inh_bulk_n

// npn Instance Q1 = spectre device Q1
  Q1 (net10 net10 vss! inh_bulk_n) trnnpn

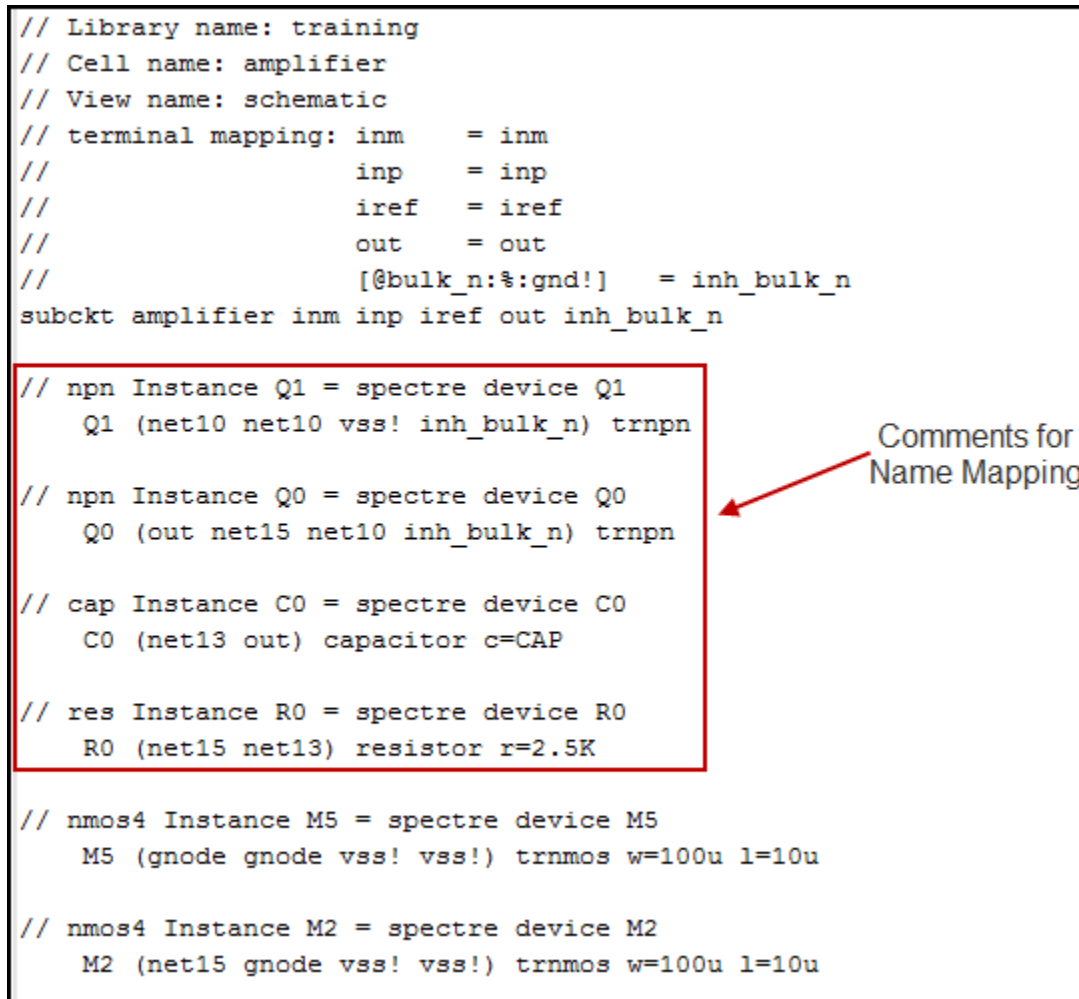
// npn Instance Q0 = spectre device Q0
  Q0 (out net15 net10 inh_bulk_n) trnnpn

// cap Instance C0 = spectre device C0
  C0 (net13 out) capacitor c=CAP

// res Instance R0 = spectre device R0
  R0 (net15 net13) resistor r=2.5K

// nmos4 Instance M5 = spectre device M5
  M5 (gnode gnode vss! vss!) trnmos w=100u l=10u

// nmos4 Instance M2 = spectre device M2
  M2 (net15 gnode vss! vss!) trnmos w=100u l=10u
```



To set this variable in the `.cdsenv` file for printing comments for name mapping and subcircuit port connections, use the following call:

```
spectre.envOpts printComments 'toggle '(t t)
```

Variable Type	toggle
Default Value	nil nil
Acceptable Values	t t, t nil, nil t, nil nil

GUI Equivalent *Virtuoso Analog Design Environment –
Environment Options – Print Comments*

definitionFiles

Type the full UNIX path or the name of one or more files. A definitions file contains function definitions and definitions of parameters that are not displayed in the Design Variables section of the simulation window.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "definitionFiles" `string "./file")
```

Variable Type	string
Default Value	""
Acceptable Values	unix path or name of one or more files.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Simulation Files Setup</u> – Definition Files</i>

enableArclength

When this variable is set to true, the homotopy convergence option is visible, else this is not visible.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "enableArclength" `boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

useAltergroup

When using models that do not work with altergroups, turn the *useAltergroup* variable to off. When using altergroups, keep this on.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "useAltergroup" `boolean "nil")
```

Variable Type boolean

Default Value t

Acceptable Values t, nil

GUI Equivalent --

netlistBBox

This variable is used to control the size of the netlist window.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "netlistBBox" `string "10 10 525 800")
```

Variable Type string

Default Value "0 0 515 700"

Acceptable Values window coordinates.

GUI Equivalent --

autoDisplayBBox

This variable is used to control the size of the `spectre.out` window.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "autoDisplayBBox" `string "10 10 525  
800")
```

Variable Type	string
Default Value	"0 0 515 700"
Acceptable Values	window coordinates
GUI Equivalent	--

includeStyle

Use the `env` option `includeStyle` to have one model per file. This option works with model name passing. When set to `t`, for stopping cells whose model name is being passed hierarchically, the passed model name specified at a higher level is added to the required model files. Or, a default value specified for a passed parameter resulting in the final specification of a model for an instance is added to the required model files.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "includeStyle" `boolean "t" )
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

simExecName

Change this variable with caution. This variable can be set to point to the path of the desired spectre executable. It is advisable not to change this variable unless very much required.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "simExecName" `string "/home/spectre/  
bin")
```

Variable Type	string
Default Value	spectre
Acceptable Values	path of the spectre executable
GUI Equivalent	--

checkpoint

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "checkpoint" `string "Y")
```

Variable Type	string
Default Value	""
Acceptable Values	Y, N
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Create Checkpoint File(cp)</i>

Virtuoso Analog Design Environment L User Guide

Environment Variables

recover

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

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "recover" `string "Y")
```

Variable Type	string
Default Value	""
Acceptable Values	Y, N
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Environment Options</u> – Start from Checkpoint File(rec)</i>

firstRun

Set this variable to false if you do not want the *Welcome to Spectre* window to pop up when you run a simulation.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "firstRun" `boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	--

simOutputFormat

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, Spectre generates an error. The `psf with floats` format is a single-precision format that uses only half the disk space that `psf` uses. Setting the value to `sst2` causes Spectre to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectre.outputs" "simOutputFormat" 'string "psf")
```

To set it in `.cdsenv`, add:

```
spectre.outputs simOutputFormat 'string "psf"
```

Variable Type	string
Default Value	psfxl
Acceptable Values	psf, psf with floats, sst2, psfxl
GUI Equivalent	<i>Outputs – Save All – Output Format</i>

fastViewOption

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the simOutputFormat environment variable is set to `psf`, `psf with floats`, or `psfxl`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectre.outputs" "fastViewOption" 'boolean t)
```

To set it in `.cdsenv`, add:

```
spectre.outputs fastViewOption 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Outputs – Save All – Use Fast Viewing Extensions</i>

controlMode

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("spectre.envOpts" "controlMode" `string "batch")
```

Variable Type	string
Default Value	interactive
Acceptable Values	interactive, batch
GUI Equivalent	--

Note:

- ❑ All the Spectre simulator options are documented under *spectre -h options* and there is one -to-one correspondence between `spectre.<optName>` and `<optName>`
- ❑ All the analysis options are documented under *spectre -h <analysisName>*.
- ❑ The following variables are deprecated:
`spectre.init remoteDir 'string "" t`
`spectre.init hostMode 'string "local" t`
`spectre.init host 'string "" t`
`spectre.init settableResultsDirectory 'boolean t t`
`spectre.init processPriority int 0 t 0 20`

licQueueTimeOut

This variable enables queuing for a license. You have to set the time required to wait for a license (in seconds). The option `'+lqtimeout <value>'` is always passed to *Spectre*, unless set to 0. It is passed with a value of 900 or any other value that is specified.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "licQueueTimeOut" `string "1000")
```

Variable Type	string
Default Value	"900"(15 min)
GUI Equivalent	--

licQueueSleep

This variable specifies the sleep time between two attempts to check out a license when queuing. Setting the value to a positive number overrides the default sleep time of 30 seconds. The option `'+lqsleep <value>'` is not passed to *Spectre* unless given a value. If it is not passed to *Spectre*, *Spectre* uses a default value of 30.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("spectre.envOpts" "licQueueSleep" `string "20")
```

Variable Type	string
Default Value	""
GUI Equivalent	--

licQueueToken

Registers a token request with the license server and creates a queue for a license.

When set to `t`, Spectre registers the token request with the license server and waits for authorization. Spectre ignores all non-token licenses during the wait time because only token licenses are queued.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "licQueueToken" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts licQueueToken boolean t
```

Variable Type	Boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	None

ignorePortOrderMismatch

When set to `t`, the netlister will not generate any warning when a portOrder mismatch occurs.

When set to the default value `nil`, any portOrder mismatch will result in following warning:

```
"Mismatch was found between the terminals in the cellView and those  
on the pin order property on the schematic, or on the termOrder  
property on the CDF. The internal default order is being used.  
Please eliminate the mismatch if any of the above properties must be  
used for netlisting."
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil
GUI Equivalent	--

dochecklimit

When set to `yes`, enables device checking without selecting the check box in form *Simulation – Device Checking – Enable Device checking*.

To set this option in `.cdsenv` or `ciw`, use the call:

```
envSetVal("spectre.opts" "dochecklimit" 'string "yes")
```

Variable Type	string
Default Value	yes
Acceptable Values	yes,no
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation/Device Checking – Use Enable Device Checking</i>

You can alternatively choose *Simulation – Options – Analog Simulator Options – Device Checking Options*.

ADE Simulation Environment

defaultHierSave

Sets the default selected value of the voltage, current, or power for all the subcircuit instances in the *Save By Subckt Instances* pane of the simulation window. By default, the check boxes for only the voltage are selected for all the subcircuit instances.

To set this environment variable in the `.cdsenv` file, use the following call:

```
asimenv defaultHierSave 'cyclic "current"
```

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv" "defaultHierSave" 'cyclic "current")
```

Variable Type	cyclic
Default Value	voltage
Acceptable Values	voltage, current, power, all, none
GUI Equivalent	None

retainStateSettings

This variable enables you to retain/ignore the current design or simulator settings in the new design or simulator. Following are the acceptable values:

- `partial` – In case of simulator change, the analyses setup, simulator options, and stimuli information are retained. Other settings such as model files and switch view options are not retained. For these options, either the user-defined settings defined in the `.cdsenv` file are considered, if provided, or their default settings are used. In case of design change, the current design settings and simulator setup files are retained. This is the default value of the variable.
- `design` – Retains all the current settings only when the design is changed.
- `simulator` – Retains all the current settings only when the simulator is changed.
- `all` – Retains the current design settings and simulator setup files. This is same as option `yes`.
- `yes` – Retains the current design settings and simulator setup files.

Virtuoso Analog Design Environment L User Guide

Environment Variables

- `none` – Ignores the current design settings and simulator setup files. This is same as option `no`.
- `no` – Ignores the current design settings and simulator setup files.

Important

It is recommended to use the options `all` or `none` in place of `yes` or `no`, respectively. The options `yes` and `no` are being maintained only for backward compatibility.

Note: To change the design, choose *Setup – Design*. To change the simulator, choose *Setup – Simulator/Directory/Host*.

Caution

Verify the retained settings, for example the simulator-specific model library files, before running the simulation. This is because some of the retained settings may not apply to the new design or the new simulator, which can result in simulation errors or incorrect simulation results.

To set this variable in the `.cdsenv` file, use the call:

```
asimenv retainStateSettings cyclic "none"
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv" "retainStateSettings" 'cyclic "all")
```

Variable Type	<code>cyclic</code>
Default Value	<code>"partial"</code>
Acceptable Values	<code>"all", "none", "partial", "design", "simulator"</code>

retainDesignVarNotation

By default, the value of a design variable is converted using the engineering notation and then displayed in the *Design Variables* pane of ADE L. If this environment variable is set to `t`, the values of the design variables are displayed in the same format that you used in the *Editing Design Variables* form.

To set this environment variable in the `.cdsenv` file, use the following call:

```
asimenv retainDesignVarNotation 'boolean t
```

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv" "retainDesignVarNotation" 'boolean t)
```

Variable Type	Boolean
Default Value	<code>nil</code>
Acceptable Values	<code>t, nil</code>
GUI Equivalent	none

useNamePrefix

Specifies whether the instance names in the netlist should have the `namePrefix` from the auCdl CDF simulation information or not.

When this environment variable is set to `t`, the `namePrefix` from the auCdl CDF siminfo is prefixed to the name of the instance. As a result, the name of the instance is same in the netlist generated from Spectre and auCdl CDF simulation information.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv" "useNamePrefix" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the following call:

```
asimenv useNamePrefix boolean t
```

Variable Type	Boolean
Default Value	<code>nil</code>
Acceptable Values	<code>t, nil</code>
GUI Equivalent	<i>None</i>

showConvertNotifyDialog

If you load a state from a previous release, the following dialog box is displayed asking you to convert the state files for the current release:



When you set the value of the `showConvertNotifyDialog` environment variable to `nil`, the state files from the previous release are converted for the current release without displaying the above dialog box.

To set this environment variable in the `.cdinit` file or CIW, use the following call:

```
envSetVal("asimenv" "showConvertNotifyDialog" 'boolean nil)
```


Virtuoso Analog Design Environment L User Guide

Environment Variables

To set this environment variable in the `.cdsenv` file, use the following call:

```
asimenv showConvertNotifyDialog 'boolean nil)
```

Variable Type	Boolean
Default Value	<code>t</code>
Acceptable Values	<code>t, nil</code>
GUI Equivalent	<i>None</i>

saveDir

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

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv" "saveDir" 'string "~/states/artist_states")
```

Variable Type	string
Default Value	<code>~/artist_states</code>
Acceptable Values	dir path

saveAsCellview

This variable lets you save ADE state into a cellview. State is saved and loaded from the cellview.

The default value is `nil`. When set to `nil`, the states are saved and loaded from directories. When set to `t`, state is saved and loaded from the cellview.

To set this variable in the `.cdsinit` file or `CIW`, use the call:

```
envSetVal("asimenv" "saveAsCellview" 'boolean nil)
```

Variable Type	boolean
Default Value	<code>nil</code>
Acceptable Values	<code>t, nil</code>

Virtuoso Analog Design Environment L User Guide

Environment Variables

GUI Equivalent

Virtuoso Analog Design Environment – Saving State – Cellview

Virtuoso Analog Design Environment – Loading State – Cellview

stateName

This variable specifies the existing `stateName` which would be automatically loaded whenever ADE session is setup. By default, user will need to load state from Load State form. In this case, the state will be searched in the default state directory location specified by the existing `saveDir` variable. The state will be picked from:

```
<saveDir>/<current Lib>/<current Cell>/<Current simulator>/  
<stateName>
```

However, if the variable `saveAsCellview` is set, the state will be loaded as cellview from:

```
<current Lib>/<current Cell>/<stateName>
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv" "stateName" `string "state1")
```

Variable Type	string
Default Value	""
Acceptable Values	dir path

stateOverWriteSkipList

When you overwrite an existing state in ADE L, all the files and directories in the state directory are deleted before the new state is saved. This variable lets you specify the files and directories that must not be deleted in the state directory when an existing state is overwritten.

To set this variable in the `.cdsenv` file, use the call:

```
asimenv stateOverWriteSkipList `string "dir1 dir2 file1 file2"
```

To set this variable in the `.cdsinit` file or CIW, use the call:

Virtuoso Analog Design Environment L User Guide

Environment Variables

```
envSetVal("asimenv" "stateOverWriteSkipList" `string "dir1 dir2 file1 file2")
```

Variable Type	string
Default Value	""
Acceptable Values	names of files or directories in the state directory, separated by spaces.

allowAdePanicStateSaving

In case of an unexpected failure in Virtuoso, there are chances to lose unsaved simulation settings. This variable allows ADE to save the settings in the panic state of unexpected failure. By default, the variable is set to t and ADE saves the settings.

To unset this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "allowAdePanicStateSaving" `boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

adePanicStatePath

In case of an unexpected failure in Virtuoso, ADE saves the current simulation settings in the directory named `.ADE_Panic_State` at the path specified using this variable. In case `adePanicStatePath` is not set by the user, the `.ADE_Panic_State` directory is saved in the simulation directory.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "adePanicStatePath" `string "dir1")
```

Variable Type	string
Default Value	""
Acceptable Values	name of a directory.

designEditMode

This variable lets you choose the default open mode for your designs. If you select `t`, your designs are opened in edit mode. If you select `nil`, your designs are opened in read-only mode.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv" "designEditMode" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Default Design Open Mode</i>

schematicBased

If this variable is set to true, it displays the Analog Design Environment menus on the Virtuoso Schematic window.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv" "schematicBased" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Schematic Menus</i>

windowBased

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv" "windowBased" `boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Simulation Window</i>

saveAsCellview

This variable lets you save ADE state into a cellview. State is saved and loaded from the cellview.

The default value is `nil`. When set to `nil`, the states are saved and loaded from directories. When set to `t`, state is saved and loaded from the cellview.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "saveAsCellview" `boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Saving State</u> – Cellview</i> <i>Virtuoso Analog Design Environment – <u>Loading State</u> – Cellview</i>

saveQuery

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv" "saveQuery" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Query to Save State</i>

adelExitQuery

While closing the ADE L session, if this variable and the saveQuery are set to t, a dialog box is displayed asking whether you want to save the state of your environment if it has not been saved.

If the saveQuery variable is set to nil, a dialog box is displayed to confirm whether you want to close the ADE L session.

If the saveQuery and adelExitQuery are set to nil, the ADE L session closes without showing any dialog box.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv" "adelExitQuery" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	--

Virtuoso Analog Design Environment L User Guide

Environment Variables

x

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.window" "x" `int 200)
```

Variable Type	int
Default Value	1
Acceptable Values	Any number between 0 and 1200
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Window X Location</i>

y

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.window" "y" `int 200)
```

Variable Type	int
Default Value	317
Acceptable Values	Any number between 0 and 1000
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Editing Session Options</u> – Window Y Location</i>

simulator

Sets the default simulator for the Analog Design Environment.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv.startup" "simulator" 'string "simulatorName|auto")
```

When this variable is set to `auto`:

- AMS is set as the default simulator if the view is a config view and it includes Verilog views, which are any text views like `.vams`, `.sv`, `.v`, and so on.
- Spectre is set as the default simulator if the view is not a config view or it does not include Verilog views.

Variable Type	string
Default Value	spectre
Acceptable Values	spectre, ams, UltraSim
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host – Simulator</u></i>

Example

To set the default simulator as `ams`, use the following call:

```
envSetVal("asimenv.startup" "simulator" 'string "ams")
```

To set the default simulator automatically, use the following call:

```
envSetVal("asimenv.startup" "simulator" 'string "auto")
```


projectDir

Lets you specify the default simulation directory.

To set this variable in the .cdsenv file, use the call:

```
asimenv projectDir `string "/tmp/simulation"
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.startup" "projectDir" `string "/tmp/simulation")
```

Variable Type	string
Default Value	"~/simulation"
Acceptable Values	directory path
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Project Directory</i>

appendLibNameToProjectDir

Lets you specify if the library name should be appended to the simulation directory.

To set this variable in the .cdsenv file, use the call:

```
asimenv.startup appendLibNameToProjectDir `boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.startup" "appendLibNameToProjectDir" `boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

hostMode

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.startup" "hostMode" `string "remote")
```

Variable Type	string
Default Value	local
Acceptable Values	local, remote, distributed
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Host Mode</i>

host

Lets you specify a path to the host computer for remote simulation. You must specify a complete path.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.startup" "host" `string "cds11938")
```

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Host</i>

digitalHostMode

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

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.startup" "digitalHostMode" `string "remote")
```

Variable Type	string
Default Value	"local"
Acceptable Values	local, remote
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Digital Host Mode</i>

digitalHost

The variable `digitalHost` lets you specify a path to the host computer for a digital remote simulation. You must specify a complete path.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.startup" "digitalHostMode" `string "cds11939")
```

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Digital Host</i>

Virtuoso Analog Design Environment L User Guide

Environment Variables

remoteDir

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.startup" "remoteDir" `string "/net/cds11938/hm/  
usr1/simulation")
```

Variable Type	string
Default Value	""
Acceptable Values	Unix path
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Choosing Simulator/Directory/Host</u> – Remote Directory</i>

autoPlot

Plots the entire plot set (including waveform expressions) automatically when each simulation run is finished.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "autoPlot" `cyclic "Refresh")
```

Variable Type	cyclic
Default Value	Auto
Acceptable Values	Auto, None, Refresh
GUI Equivalent	<i>Plot After Simulation drop-down list box on the ADE L window.</i>

artistPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "artistPlottingMode" `string "New Win")
```

Variable Type	string
Default Value	Replace
Acceptable Values	Append / Replace / New Win / New SubWin
GUI Equivalent	<i>Plotting Mode drop-down listbox on the ADE L window.</i>

directPlotPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "directPlotPlottingMode" `string "New Win")
```

Variable Type	string
Default Value	Append
Acceptable Values	Append / Replace / New Win / New SubWin
GUI Equivalent	<i>Cyclic is on the DirectPlot Main form.</i>

designName

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "designName" `boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Design Name</i>

drlBufferMemory

Allocates a buffer memory of the specified size for Virtuoso.

In cases when Virtuoso reads huge simulation data or evaluates memory-intensive expressions, the process might consume all the available memory. When allocation of more memory fails, Virtuoso might terminate abnormally or it might behave inconsistently. In such cases, Virtuoso can use this buffer memory to make the process exit gracefully. By default, memory of size 10 million bytes is allocated. You may increase the size of this memory for better results.

Note: Memory allocated using the drlBufferMemory variable helps in exiting Virtuoso gracefully in case of memory crunch. However, it may not be successful in all such scenarios.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "drlBufferMemory" `int 10000000)
```

Variable Type	int
Default Value	10000000
Acceptable Values	10 to 25% of the total RAM size

simulationDate

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "simulationDate" `boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Simulation Date</i>

temperature

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "temperature" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – temperature</i>

variables

If true, displays the names and values of design variables in the graph window.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "variables" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Design Variables</i>

designVarSetting

Specifies a list of design variables, with their respective values, to be added to an ADE session at the time of session creation or initialization. The list is specified as a string with space-separated name and value pairs. The design variables created with this variable are added in the *Design Variables* pane.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "designVarSetting" `string "var1 10 var2 30")
```

In this example, whenever an ADE session is created, two design variables, var1 and var2, are set with values 10 and 30, respectively.

Variable Type	string
Default Value	nil
Acceptable Values	Space-separated list of design variables and their values, nil
GUI Equivalent	-

scalarOutputs

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

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "scalarOutputs" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Scalar Outputs</i>

icons

This variable places icons in the graph window.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "icons" `boolean "t")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Allow Icons</i>

resizeMode

Specifies how the Virtuoso Visualization and Analysis XL graph window will be resized.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "resizeMode" 'string "manual")
```

Variable Type	string
Default Value	manual
Acceptable Values	manual, auto
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Resize Mode</i>

x

Enables you to set the horizontal position of the left side of the Virtuoso Visualization and Analysis XL graph window.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "x" `int 500)
```

Variable Type	int
Default Value	577
Acceptable Values	Between 0 and 1200.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – X Location</i>

y

Enables you to set the vertical position of the top of the Virtuoso Visualization and Analysis XL graph window.

To set this variable in the .cdsinit file or ciw, use the call:
`envSetVal("asimenv.plotting" "y" `int 500)`

Variable Type	int
Default Value	373
Acceptable Values	Between 0 and 1000.
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Y Location</i>

immediatePlot

This variable refers to the commands located in the *Direct Plot* menu. If true, the plot is drawn after each node is selected. If nil, none of the plots are drawn until all the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are plotted at the same time.

To set this variable in the .cdsinit file or ciw, use the call:
`envSetVal("asimenv.plotting" "immediatePlot" `boolean "t")`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Direct Plots Done After</i>

immediatePrint

This variable refers to the commands located in the Print menu. If true, the results are printed after each node is selected. If nil, none of the nodes is printed until all the nodes have been selected.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.plotting" "immediatePrint" `boolean "t")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Setting Plotting Options</u> – Print After</i>

useDisplayDrf

Specifies whether the display settings for trace signals and plotting graphs should be applied from the `display.drf` file or from the Virtuoso Visualization and Analysis XL color bank.

When set to `t`, the Analog Design Environment (ADE) uses the display settings from the `display.drf` file for color, style and thickness of trace signals, color highlighting of schematic probes, and plotting waveforms in Virtuoso Visualization and Analysis XL.

Important Points to Note:

- For traces plotted on the graph by using the results browser or calculator, the style, thickness and color bank from the Virtuoso Visualization and Analysis XL are used.
- The display settings from the `display.drf` files are not considered while running sweeps, corners, and parametric analysis in ADE.

When this environment variable is set to `nil`, all the display definitions from Virtuoso Visualization and Analysis XL are used.

For more information, see the following:

Virtuoso Analog Design Environment L User Guide

Environment Variables

- *How do I change the default style, color, and thickness of a waveform or trace in Virtuoso Visualization and Analysis XL?* section in the *Virtuoso Visualization and Analysis XL Frequently Asked Questions*.
- *Graph Environment Variables* section in the *Virtuoso Visualization and Analysis XL User Guide*.

To set this variable in the `.cdsinit` file or CIW to use the display settings from `display.drf` file, use the following call:

```
envSetVal("asimenv.plotting" "useDisplayDrf" 'boolean t)
```

To set this variable in the `.cdsenv` file to use the display settings from `display.drf` file, use the following call:

```
asimenv.plotting useDisplayDrf boolean t
```

Variable Type	Boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	None

preSaveOceanScript

This procedure is executed before the ocean script is created, when the *Save Ocean Script* option is enabled. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MYfirstProc( session fp )
```

In this syntax, `session` is the ADE L session and `fp` is the file pointer to the OCEAN script. You do not need to set these. ADE L sets these for you. In this case, the value for the variable `postSaveOceanScript` will be `MyfirstProc`.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.misc" "preSaveOceanScript" `string  
"myPreSaveProc")
```

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure

GUI Equivalent --

postSaveOceanScript

This procedure is executed after the ocean script is created when the *Save Ocean Script* option is clicked. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MyLastProc( session fp )
```

In this syntax, *session* is the ADE L session and *fp* is the file pointer to the OCEAN script. You do not need to set these; ADE L sets these for you. In this case, the value for the variable *postSaveOceanScript* will be *MyLastProc*.

To set this variable in the *.cdsinit* file or *ciw*, use the call:

```
envSetVal("asimenv.misc" "postSaveOceanScript" `string  
"myPostSaveProc")
```

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure
GUI Equivalent	--

noSimLogInOCEAN

Controls whether the simulation log is to be included in the log for OCEAN script. By default, the log for an OCEAN run includes the simulator output. For large simulations, this can result in large log files. In such cases, before running a simulation, you can set this variable to *nil* to exclude the simulator output from the OCEAN job log.

To set this variable in the *.cdsenv* file, use the following call:

```
asimenv.misc includeSimLogInOCEAN 'boolean t
```

To set this variable in the *.cdsinit* file or *CIW*, use the following call:

```
envSetVal("asimenv.misc" "includeSimLogInOCEAN" 'boolean t)
```

Variable Type	Boolean
Default Value	t

Acceptable Values t, nil

numberOfSavedRuns

Once set to value greater than 0, ADE L will retain the simulation run data for the last *numberOfSavedRuns* simulations. In case of Parametric, a single run may include multiple simulations. At the end of a simulation run, ADE L will save the current run data under:

```
<simulation_dir>/<cell_name>/<simulator_name>/<view_name> to a  
numbered directory under <simulation_dir>/<cell_name>/  
<simulator_name>.
```

The number used is one higher than the highest numbered directory name or 1 if none exist. If the maximum number of *Saved Runs* is reached, ADE L will save the current run data, but delete the smallest numbered directory, thus keeping the number of Saved Runs equal to the value set in the variable.

Example:

numberOfSavedRuns is set to 2

Under <simulation>/<ampTest>/<spectre>

1. At the end of first simulation run

```
1/  schematic/
```

2. At the end of second simulation run

```
1/  2/  schematic/
```

3. At the end of third simulation run

```
2/  3/  schematic/
```

In all the three cases above the schematic directory would have the current simulation results.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.misc" "numberOfSavedRuns" `int 2)
```

Variable Type int

Default Value 0

browserCenterMode

To keep the most recently expanded node in the results browser in the center of the window, set this variable to true. If you do not want the most recently expanded node to move automatically to the center of the window, you can turn off the centering mode by setting this variable to `nil`.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.misc" "browserCenterMode" `boolean "nil")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

outputsImportExportVersion

If the value of this variable is greater than 1.0, then ADE exports or imports the outputs from the CSV file. You can change the value of this variable to less than 1.0 if you want to export or import the outputs from the text file.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.misc" "outputsImportExportVersion" floatValue)
```

Variable Type	float
Default Value	1.1
Acceptable Values	any floating value
GUI Equivalent	<i>Virtuoso Analog Design Environment – Outputs</i> <i>– <u>Export</u></i>
	<i>Virtuoso Analog Design Environment – Outputs</i> <i>– <u>Import</u></i>

updateCDFtermOrder

If this variable is set to true, it allows updating of the CDF termOrder after a symbol update. The default setting is nil. The CDF updating only affects the termOrder information. Before any updating of the CDF occurs, a dialog box appears and confirms if it is OK to update the base cell CDF termOrder data. The dialog box displays the simulators whose termOrder it will update, and the new termOrder that will be set for each simulator.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("auCore.misc" "updateCDFtermOrder" `boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

printNotation

It is used to specify how numbers are printed in the ADE L environment. This applies only to *Results – Print* and *print/printvs* in the Calculator. Numbers are printed in the notation this variable is set to.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("auCore.userPref" "printNotation" `cyclic "scientific")
```

Variable Type	cyclic
Default Value	suffix
Acceptable Values	engineering, scientific, suffix
GUI Equivalent	--

saveDefaultsToOCEAN

When this variable is turned on, in addition to what is normally saved, it saves the following:

- All non-blank options
- All non-blank envOptions
- All enabled analyses and their options (as opposed to all analyses).
- All keep options (save all nets / currents... etc.)
- The model path(s)
- Temperature
- Simulator/analysis defaults to ocean scripts generated from ADE L.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.misc" "saveDefaultsToOCEAN" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

showWhatsNew

Set this variable to the release number for which you do not want to see the What's New window. For example, set this variable to 5.0.0 if you do not want to see the What's New window for 5.0.0.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv" "showWhatsNew" `string "5.1.2")
```

Variable Type	string
Default Value	"yes"
Acceptable Values	Any existing release number
GUI Equivalent	--

digits

Number of significant digits with which the contributors are printed.

Variable Type	int
Default Value	6
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
GUI Equivalent	--

obsoleteWarnings

Number of warnings that are needed to be stored. By default, this variable is set to 1. Therefore, while netlisting, one error is shown at a time. If it is desired that more number of errors are shown then change this variable to a larger number.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.netlist" "obsoleteWarnings" `int 2)
```

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

netlistAccess

This variable is used to specify the access permissions for the netlist. By default, this variable is set to `User`. Therefore, while netlisting, only the creator of the netlist will be able to run simulation on it. You can also set the values of the variable to `Group` and `All`. By setting the value to `Group`, all the users in the same group as the `User` will be able to run the netlist. If you set the value to `All`, anybody can run the netlist.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.netlist" "netlistAccess" `string `User)
```

Variable Type	string
Default Value	User
Acceptable Values	User, Group, and All
GUI Equivalent	--

printCommentChar

This variable sets the preferred comment character for the printvs data. The # sign is the default comment character. To set the preferred comment character, set the variable, *printCommentChar* to the character.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("asimenv.userPref" "printCommentChar" `string "+")
```

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

updateCDFtermOrder

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

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

toolList

Set this variable to the list of simulators integrated into ADE. If a new simulator is integrated, it has to be added to this list.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("auCore.toolFilter" "toolList" `string "spectre")
```

Variable Type	string
Default Value	string "spectre auCdl auLvs"
Acceptable Values	spectre auCdl auLvs

ignoreSchModified

Set this variable to *t*, the schematic will not be modified. When this variable is set, you need not do a check and save after making changes to the *toolFilter* form.

To set this variable in the `.cdsenv` file or `CIW`, use the call:

```
envSetVal("auCore.toolFilter" "ignoreSchModified" `boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

defaultTools

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

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("auCore.toolFilter" "defaultTools" `string "spectre")
```

Variable Type	string
Default Value	string "spectre auCdl auLvs"
Acceptable Values	spectre auCdl auLvs

oceanScriptFile

Set this variable to specify the default location for saving OCEAN script files.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.misc" "oceanScriptFile" `string "./myOcnScript")
```

Variable Type	string
Default Value	"./oceanScript.ocn"

printInlines

When this variable is set to `t`, data for all devices in a textual subcircuit will be printed. refer to chapter 10. Searching for devices in a textual subcircuit may take some time. If you want to disable this feature, set this variable to `nil`.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("asimenv.plotting" "printInlines" `boolean "nil")
```

Variable Type	Boolean
Default Value	<code>t</code>
Acceptable Values	<code>t/nil</code>
Window Menu	<i>Results – Print – <u>DC Operating Points</u></i> <i>Results – print – <u>Transient Operating points</u></i> <i>Results – print – <u>Model Parameters</u></i>

awvResizeWindow

```
awvResizeWindow( w_windowId l_bBox )  
=> t | nil
```

Resizes a window to the size of a bounding box. The bounding box is specified as a list of 2 x:y points, that represent the lower left and upper right corners of the box.

Variable Type	Boolean
Default Value	<code>t</code>
Acceptable Values	<code>t/nil</code>
Example	<pre>awvResizeWindow(window(4) list(391:288 1134:922))</pre>

paraplotUpdateSimulatorLog

When this variable is set to t, the simulator log appears in a new window, when a parametric analysis is run.

Default Value	nil
Acceptable Values	t/nil
Variable Type	Boolean

sevResolveSymLinks

When this variable is set to t, the model files are added after resolving symbolic links. If it is nil, symbolic links are not expanded. To prevent symbolic links from being expanded when browsing for model files using the Model Library Setup form, set this variable in the .cdsinit file as nil.

Default Value	t
Acceptable Values	t/nil
Variable Type	Boolean

ADE XL

showMenu

Use this variable to show or hide the *Launch* menu that is used to open ADE XL or ADE GXL from ADE L. By default, this variable is set to `t` and the *Launch* menu is visible. To hide the menu, set this variable to `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "showMenu" 'boolean nil)
```

To set it in the `.cdsenv` file, use the call:

```
adexl.launchFromTest showMenu 'boolean nil
```

showOpenViewDialog

Use this variable to stop the *Launch ADE (G)XL* form from appearing when you open ADE XL or ADE GXL from ADE L. By default, this variable is set to `t` and the *Launch ADE (G)XL* form appears. Using this form, you can either specify the choice to create new `adexl` view or to open an existing view. If you want to always create a new view when you run ADE XL/GXL from ADE L, set this variable to `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "showOpenViewDialog" 'boolean nil)
```

To set it in the `.cdsenv` file, use the call:

```
adexl.launchFromTest showOpenViewDialog 'boolean nil
```

defaultLibName

Use this variable to specify name of the library in which the new view is to be created while opening ADE XL or ADE GXL from ADE L. By default, this variable is not set. You can set a library name by using this variable.

Note: This library name is used only when the showOpenViewDialog variable is set to `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "defaultLibName" 'string  
"myDefaultLib")
```

To set it in the `.cdsenv` file, use the call:

```
adexl.launchFromTest defaultLibName `string "myDefaultLib"
```

viewNamePrefix

Use this variable to add the default prefix to the names of new `adexl` views that are created when you run ADE XL or ADE GXL from ADE L. By default, this variable is set to `adexl_imported_from_adel` and all the view names are prefixed with this string. You can set a different prefix using this variable.

Note: This prefix value is used only when the showOpenViewDialog variable is set as `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "viewNamePrefix" 'string "XView")
```

To set it in the `.cdsenv` file, use the call:

```
adexl.launchFromTest viewNamePrefix `string "XView"
```

SpectreVerilog (IC6.1.6 only)

simOutputFormat

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, Spectre generates an error. The `psf with floats` format is a single-precision format that uses only half the disk space that `psf` uses. Setting the value to `sst2` causes Spectre to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectreVerilog.outputs" "simOutputFormat" `string "psf")
```

To set it in the `.cdsenv` file, add:

```
spectreVerilog.outputs simOutputFormat `string "psf"
```

Variable Type	string
Default Value	sst2
Acceptable Values	psf,psf with floats,sst2, psfxl
GUI Equivalent	<i>Outputs – Save All – Output Format</i>

fastViewOption

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the simOutputFormat environment variable is set to `psf`, `psf with floats`, or `psfxl`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectreVerilog.outputs" "fastViewOption" 'boolean t)
```

To set it in `.cdsenv`, add:

```
spectreVerilog.outputs fastViewOption 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Outputs – Save All – Use Fast Viewing Extensions</i>

logicOutputFormat

Use this variable to generate digital data outputs for mixed-signal simulations in the SignalScan Turbo 2 (SST2) format. The Virtuoso AMS and SpectreVerilog simulators write transient analyses results in the SST2 format. The Virtuoso Spectre and Virtuoso Ultrasim simulators can also write this format for a transient analysis.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectreVerilog.envOpts" "logicOutputFormat" `string  
"SST2")
```

To set it in the `.cdsenv` file, add:

```
spectreVerilog.envOpts logicOutputFormat `string "SST2"
```

Variable Type	string
Default Value	SST2
Acceptable Values	SST2
GUI Equivalent	--

HspiceD

hspiceSoftLineLength

This variable controls the maximum number of characters in a line of netlist output after which the line is automatically split into multiple lines for easy reading. This variable overrides the OSS variable, `softLineLength`, which is set to 1024 characters by default.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
hspiceSoftLineLength=80
```

Variable Type	int
Default Value	1024
Acceptable Values	Any integer less than or equal to 1024

hspiceMaxLineLength

Use this variable to increase or decrease the maximum limit on the number of characters to be printed in a line of netlist output from the default. This variable overrides the OSS variable, `maxLineLength`. Note that the `hspiceSoftLineLength` value can never be greater than `hspiceMaxLineLength` value.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
hspiceMaxLineLength=1024
```

Variable Type	int
Default Value	1024
Acceptable Values	Any integer. The maximum limit is the limit of an integer.

For more information, see the [*Customizing the Hierarchical Netlister \(HNL\)*](#) chapter of the *Open Simulation System Reference*.

mapGndNetToZero

Use this variable to control mapping of `gnd!` nets. By default, its value is set to `t` which implies `gnd!` nets are mapped to 0. To stop the mapping of `gnd!` nets to 0, set the variable to `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "mapGndNetToZero" 'boolean nil)
```

To set it in `.cdsenv`, add:

```
hspiceD.envOpts mapGndNetToZero 'boolean nil
```

Variable Type	boolean
Default Value	t
Acceptable Values	t/nil

netlistModelFileFirst

Use this variable to control sequence of netlisting of design variables and include files. By default, the design variables are netlisted before the include files. To netlist the include files before the design variables, set the value of `netlistModelFileFirst` variable to `t`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "netlistModelFileFirst" 'boolean t)
```

To set it in `.cdsenv`, add:

```
hspiceD.envOpts netlistModelFileFirst 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil

userCmdLineOption

Use this variable to pass command line options in hspice. For example, to set multi threading to 3, you need to pass `-mt 3` as the value.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "userCmdLineOption" 'string "-mt 3")
```

To set this variable in `.cdsenv` file, add:

```
hspiceD.envOpts userCmdLineOption 'string "-mt 3"
```

Variable Type	string
Default Value	""
Acceptable Values	Any string value

setTopLevelAsSubckt

This variable controls whether the top-level schematic should be netlisted as a subcircuit or not. If it is set to `t`, the top-level schematic is netlisted as a subcircuit; otherwise, it is not netlisted as a subcircuit.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "setTopLevelAsSubckt" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

AMS

globalSignals

Adds the signals to the *Global Signals* form for the AMS simulator.

The *Global Signals* form is auto populated after the AMS netlister extracts the signals from the schematic. AMS netlister cannot extract the signals when the signals are part of text views, or if some hierarchy inside the HED is not visible to AMS netlister. In such cases you need to add inaccessible signals to the *Global Signals* form.

The syntax for adding the signal is

```
[origin];globalSignalName;namespace;wireType;[discipline];isGround;F;extractedWireType;
```

where

origin	(optional) Specifies the origin of the signal. If the signal is extracted from the schematic, the value is D. If you are adding the signal manually, leave this as blank.
globalSignalName	Specifies the name of the signal.
namespace	Specifies the namespace in which the signal was created. Acceptable values: CDBA, Spectre, Spice, and Verilog-AMS.
wireType	Specifies the wire type of the signal. Acceptable values: wire, supply0, supply1, wreal, wor, wand, tri, tri0, tri1, triand, trior, and trireg.
discipline	(optional) Specifies the discipline of the signal.
isGround	Specifies if the global signal is used as a ground reference. Acceptable values: YES, NO.
F	It is an internally managed flag, and its value should always be F.
extractedWireType	Specifies the wire type extracted by the netlister. It is also an internally managed flag, and its value should be same as wireType.

To set this variable in the `.cdsinit` file or CIW for adding the global signal `gnd!`, use the following call:

Virtuoso Analog Design Environment L User Guide

Environment Variables

```
envSetVal("ams.netlisterOpts" "globalSignals" 'string  
";;gnd!;CDBA;wire;;YES;F;wire;")
```

To set this variable in the `.cdsinit` file or CIW for adding the global signals `gnd!` and `vdd12!`, use the following call:

```
envSetVal("ams.netlisterOpts" "globalSignals" 'string  
";;gnd!;CDBA;wire;;YES;F;wire; ;;vdd12!;CDBA;wire;;NO;F;wire;")
```

Important Points to Note:

- Add a space if you are adding more than one signal. In the above example, a space is present after the values of `gnd!` signal.
- If you do not want to specify any value for the optional fields, do not replace them with a space. In the above example, the value for `discipline` is not specified.

To set this variable in the `.cdsenv` file for adding the global signal `gnd!`, use the following call:

```
ams.netlisterOpts globalSignals 'string ";;gnd!;CDBA;wire;;YES;F;wire;"  
|
```

Variable Type	string
Default Value	" "
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister tab – <u>Global Signals</u></i>

netlistMaxWarn

Use the `netlistMaxWarn` environment variable to specify the maximum number of warning messages issued by the netlister before it stops processing the design.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "netlistMaxWarn" 'string "3")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts netlistMaxWarn 'string "3"
```

Variable Type	string
Default Value	-
Acceptable Values	Any integer
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation – Options – <u>AMS Simulator</u> – Netlister Tab</i>

netlistNoWarn

You can suppress the information or warning messages issued by the netlister while processing the design by specifying a space-separated list of message IDs, such as `AMS-2000 AMS-2171 AMS-2174`, to the `netlistNoWarn` environment variable.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "netlistNoWarn" 'string "AMS-2000 AMS-2171 AMS-2174")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts netlistNoWarn 'string "AMS-2000 AMS-2171 AMS-2174"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of warning message IDs
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation – Options – <u>AMS Simulator</u> – Netlister Tab</i>

upgradeMsgSevWarn

Changes the severity of the specified information messages to warning while processing a design. The message IDs should be provided as a space-separated list, such as AMS-1244 AMS-1246.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "upradeMsgSevWarn" 'string "AMS-1244 AMS-1246")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts upradeMsgSevWarn 'string "AMS-1244 AMS-1246"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of information message IDs
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation – Options – <u>AMS Simulator</u> – Netlister Tab</i>

upgradeMsgSevError

Changes the severity of the specified warning messages to error while processing a design. The message IDs should be provided as a space-separated list, such as AMS-2171 AMS-2174.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "upradeMsgSevError" 'string "AMS-2171 AMS-2174")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts upradeMsgSevError 'string "AMS-2171 AMS-2174"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of warning message IDs
GUI Equivalent	<i>Virtuoso Analog Design Environment – Simulation – Options – <u>AMS Simulator</u> – Netlister Tab</i>

print_control_vars

Enables the AMS netlister to print a list of control variables for which the default AMS value has been changed in the `changedVarsSummary` file. This file is saved in the netlist directory.

The syntax of this variable is as follows:

```
envSetVal("ams.netlisterOpts" "print_control_vars" 'boolean {t|nil})
```

The default value of this variable is `t`. If you do not want the `changedVarsSummary` file to be created, unset the variable in `.cdsinit` or `.cdsenv`.

To unset this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "print_control_vars" 'boolean nil)
```

To unset this variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts print_control_vars 'boolean nil
```

Variable Type	Boolean
Default Value	t
Acceptable Values	t, nil

ac_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for AC analysis.

To set this variable in the `.cdsinit` file or `ciw`, use the call:

```
envSetVal("ams.checkOpts" "ac_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

assert_severity_default

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when the device checks are violated.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("ams.checkOpts" "assert_severity_default" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – <u>Device Check Specification</u></i>

dc_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC sweep analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("ams.checkOpts" "dc_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

dcOp_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("ams.checkOpts" "dcOp_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

tran_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the transient analysis.

To set this variable in the .cdsinit file or ciw, use the call:

```
envSetVal("ams.checkOpts" "tran_severity" `string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	<i>Virtuoso Analog Design Environment – Device Check Specification – <u>Device Checking Options</u></i>

connectRulesList

Sets the default set of connect rules. To set this variable in the .cdsinit file, use the call:

```
envSetVal("ams.envOpts" "connectRulesList" `string  
"connectLib;ConnRules_18V_full;connect  
connectLib;mixedsignal;connect")
```

This variable will set two connect rules connectLib/ConnRules_18V_full and connectLib/mixedsignal as the default connect rules in the Connect Rules form.

To set a single connect rule, use the following call:

```
envSetVal("ams.envOpts" "connectRulesList" `string  
"connectLib;ConnRules_5V_full;connect")
```

Variable Type	string
Default Value	"connectLib;ConnRules_5V_full;connect"

useEffectiveCDF

When this variable is set to t, AMS in ADE uses effective CDF while netlisting. If set to nil, the base cell CDF will be used. To set this variable in the .cdsenv file or CIW, use the call:

```
envSetVal("ams.netlisterOpts" "useEffectiveCDF" `boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil

disableRunModeInDP

When the host mode is distributed processing, setting the value to `t` disables the interactive simulation mode and sets batch simulation mode as default for distributed processing. Setting the value to `nil` will keep both interactive and batch simulation mode active for any host mode.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("ams.envOpts" "disableRunModeInDP" 'boolean t)
```

To set it in the `.cdsenv` file, add:

```
ams.envOpts disableRunModeInDP 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

simOutputFormat

Use this variable to specify the format of output results. Setting the value to `sst2` causes AMS to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("ams.outputs" "simOutputFormat" 'string "wdf")
```

To set it in the `.cdsenv` file, add:

```
ams.outputs simOutputFormat 'string "wdf"
```

Variable Type	string
Default Value	sst2
Acceptable Values	sst2, fsdb, wdf

useOtherOutputFormat

Use this variable to control whether the settings for the `simOutputFormat` variable are displayed in the Save Options form or not.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal( "ams.outputs" "useOtherOutputFormat" 'boolean t)
```

To set it in the `.cdsenv` file, add:

```
ams.outputs useOtherOutputFormatboolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil

AMS_IGNORE_IGNORE

Specifies that the AMS netlister should ignore the `nlAction=ignore` property set on terminals. Those terminals will be included in either the instance port list or the module port list (depending on whether the terminal in question is on an instance or on the cellview being netlisted). For more information about the `nlAction=ignore` property, see the [Virtuoso NC Verilog Environment User Guide](#).

To set this environment variable, enter the following command in the UNIX terminal:

```
setenv AMS_IGNORE_IGNORE true
```

Note: Unset this environment variable if you do not want the AMS netlister to ignore the `nlAction=ignore` property set on terminals.

For a comprehensive list of the AMS variables, refer to [Appendix A](#) of the [Virtuoso AMS Environment User Guide](#).

Ultrasim

wf_format

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, UltraSim generates an error. Setting the value to `sst2` causes UltraSim to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("UltraSim.outputs" "wf_format" 'string "psf")
```

To set it in the `.cdsenv` file, add:

```
UltraSim.outputs wf_format 'string "psf"
```

Variable Type	string
Default Value	sst2
Acceptable Values	psf,psfx1,sst2
GUI Equivalent	<i>Outputs – Save All – Output Format</i>

fastViewOption

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the `wf_format` environment variable is set to `psf`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("UltraSim.outputs" "fastViewOption" 'boolean t)
```

To set it in `.cdsenv`, add:

```
UltraSim.outputs fastViewOption 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Outputs – Save All – Use Fast Viewing Extensions</i>

useOtherOutputFormat

When this variable is set to `t`, PSF, PSFXL, SST2, FSDB, and WDF is displayed for Output Format field of Simulator Options form. If set to `nil`, PSF and SST2 is displayed as Output Format. To set this variable in the `.cdsinit` file, use the call:

```
envSetVal("UltraSim.opts" "useOtherOutputFormat" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil

For a comprehensive list of UltraSim variables, refer to [Appendix A](#) of the [Virtuoso AMS Environment User Guide](#).

UltraSimVerilog (IC6.1.6 only)

wf_format

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, UltraSim generates an error. Setting the value to `sst2` causes UltraSimVerilog to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("UltraSimVerilog.outputs" "wf_format" `string "psfxl")
```

To set it in the `.cdsenv` file, add:

```
UltraSimVerilog.outputs wf_format `string "psfxl"
```

Variable Type	string
Default Value	sst2
Acceptable Values	psf,psfxl,sst2
GUI Equivalent	<i>Outputs – Save All – Output Format</i>

fastViewOption

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the wf_format environment variable is set to `psf`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("UltraSimVerilog.outputs" "fastViewOption" 'boolean t)
```

To set it in `.cdsenv`, add:

```
UltraSimVerilog.outputs fastViewOption 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	<i>Outputs – Save All – Use Fast Viewing Extensions</i>

Virtuoso Analog Design Environment L User Guide

Environment Variables

auCdl Netlisting

This appendix describes the auCdl (Analog and Microwave Circuit Description Language) netlisting procedure. It contains details on parameters required for auCdl and also the different ways to netlist to auCdl. This information is applicable to any 4.4 version of the Virtuoso® design framework II (DFII).

This appendix covers the following topics:

- [What Is auCdl and Why Do You Need It?](#) on page 745
- [Licensing Requirements](#)
- [Running auCdl](#) on page 746
- [Customization Using the .simrc File](#) on page 753
- [Support for HED Features](#) on page 761
- [Custom Netlisting Procedures](#) on page 762
- [Black Box Netlisting](#) on page 771
- [Additional Customizations](#) on page 775
- [CDF Simulation Information for auCdl](#) on page 790
- [Complete Example](#) on page 796

What Is auCdl and Why Do You Need It?

To compare a layout versus a schematic (LVS) using LOGLVS, you need a netlist representation of the schematic for a design for LOGLVS. The netlist must be in CDL (Circuit Description Language) format. To create a CDL netlist for an analog circuit, you use a netlister called auCdl (Analog and Microwave Circuit Description Language).

Licensing Requirements

You must have one of the following licenses to run auCdl from DFII or from the command line. If one of these licenses are not already checked out, the first available license will be checked out in the following order when you run auCdl:

- 95100 Virtuoso® Schematic Editor L
- 95115 Virtuoso® Schematic Editor XL
- 206 Virtuoso® Simulation Environment

Running auCdl

You can run auCdl from within or outside the DFII environment.

To translate files from the DFII database format into an auCdl netlist,

1. Set the `CDS_Netlisting_Mode` environment variable as given below:

```
setenv CDS_Netlisting_Mode "Analog"
```

2. Create an auCdl view for the cell. For more information, see [Creating a config view for auCdl](#) on page 750.
3. Add the auCdl simulation information to the cell's CDF. For more information, see [CDF Simulation Information for auCdl](#) on page 790.

You can customize the auCdl Netlister using the simulation run control (`.simrc`) file. For more information, see [Customization Using the .simrc File](#) on page 753.

Running auCdl from within DFII

In DFII, you can extract the auCdl netlist by doing the following:

1. In the CIW, choose *File – Export – CDL*.
2. In the CDL Out Run form, fill in the appropriate fields and click *OK* or *Apply*.

For more information about using CDL Out, read the *Translating CDL Files* section in the [Design Data Translator's Reference](#).

Running auCdl from the Command Line

To run CDL Out from the command line, you must create a simulation environment (`si.env`) file in advance and name the file as a command argument. Run CDL Out interactively once to create the `si.env` file. Once the `si.env` file is created,

1. Copy the `cds.lib` file to the run directory.
2. Enter the following command:

```
si -batch -command netlist
```

the `-batch` option runs CDL in batch mode and the `-command netlist` option generates an ASCII netlist file.

CDL Out can generate a hierarchical netlist. CDL Out generates a netlist hierarchy that duplicates the hierarchy of your design. Each cell in your schematic becomes a separate subcircuit in the netlist. The hierarchical netlister automatically prefixes each instance name with the proper character for its element type; for example, "M" for MOSFET and "R" for resistor. This prefixing minimizes mapping and name translation.

The `si.env` File

The following is an example of a `si.env` file followed by description of each of these properties.

```
simLibName = "testLib"
simCellName = "testTop"
simViewName = "schematic"
simSimulator = "auCdl"
simNotIncremental = nilsimReNetlistAll = nil
simViewList = '("auCdl" "schematic" "gate.sch" "cmos.sch")
simStopList = '("auCdl")
simNetlistHier = t
hnlNetlistFileName = "netlist"
simRunDir = "/cds/1.0/test/translator/cdlout/paramCase/"
resistorModel = " "
shortRES = 2000.0
preserveRES = 'nil
checkRESVAL = 'nil
checkRESSIZE = 'nil
preserveCAP = 'nil
checkCAPVAL = 'nil
checkCAPAREA = 'nil
preserveDIO = 'nil
checkDIOAREA = 'nil
checkDIOPERI = 'nil
displayPININFO = 'nil
preserveALL = 'nil
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

Description of si.env Properties

Property	Description
<code>simLibName</code>	Name of the library containing the top-level cellview of the design.
<code>simCellName</code>	Name of the top-level cellname of the design.
<code>simViewName</code>	Name of the top-level view of the design.
<code>simSimulator</code>	Simulator to run.
<code>simNotIncremental</code>	When this property is set to <code>nil</code> , the netlister runs incrementally, which means the system netlists only the parts of your design you modified since you last netlisted the design. The default is <code>nil</code> .
<code>simReNetlistAll</code>	When this property is set to <code>t</code> , the netlister runs a new netlist on all the cellviews in your entire design. The default is <code>nil</code> .
<code>simViewList</code>	List of views to open for each cell when traversing the design hierarchy during netlisting and name translation.
<code>simStopList</code>	List of views that are valid stopping points for expansion used during netlisting.
<code>hnlNetlistFileName</code>	Name of the text netlist file.
<code>simRunDir</code>	Directory in which CDL data is stored. Set this global variable to the current run directory. This variable is set when the simulation environment is initialized.
<code>resistorModel</code>	String that sets the model name to be treated as a short. Prints out the string in the <code>*.RESI</code> command. The default is <code>nil</code> .
<code>shortRES</code>	Sets the value of resistance below which the resistor is assumed to be shorted. Prints the value out in the <code>*.RESI</code> command. The default is <code>2000.0</code> ; type is floating point.
<code>preserveRES</code>	When this property is set to <code>t</code> , resistors are preserved for checking in LVS, <code>shortRES</code> , and <code>checkRESSIZE</code> . Using the optional variable <code>[XX]</code> , you can specify a model name that preserves only the specified type of resistor. The default is <code>nil</code> .
<code>checkRESVAL</code>	Prints out <code>*.RESVAL</code> when set to <code>t</code> . The default value is <code>nil</code> .

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

Property	Description
checkRESSIZE	If <code>preserveRES</code> is <code>nil</code> , prints out <code>*.RESSIZE</code> when <code>checkRESSIZE</code> is set to <code>t</code> . The default is <code>nil</code> .
preserveCAP	When this property is set to <code>t</code> , <i>Export – CDL</i> preserves capacitors for checking in LVS. You can define <code>checkCAPAREA</code> if <code>preserveCAP</code> is <code>t</code> . The default is <code>nil</code> .
checkCAPVAL	Prints out <code>*.CAPVAL</code> when set to <code>t</code> . The default is <code>nil</code> .
checkCAPAREA	If <code>checkCAPVAL</code> is <code>nil</code> , prints out <code>*.CAPAREA</code> when <code>checkCAPAREA</code> is set to <code>t</code> . The default is <code>nil</code> .
preserveDIO	If <code>preserveDIO</code> is set to <code>t</code> , <i>Export – CDL</i> preserves the diodes for checking in LVS. You can define the variable <code>checkDIOAREA</code> if <code>preserveDIO</code> is <code>t</code> . The default is <code>nil</code> .
checkDIOAREA	Prints out <code>*.DIOAREA</code> when set to <code>t</code> . The default is <code>nil</code> .
checkDIOPERI	Prints out <code>*.DIOPERI</code> when set to <code>t</code> . The default is <code>nil</code> .
displayPININFO	When <code>displayPININFO</code> is set to <code>t</code> , prints out the <code>*.PININFO</code> command for each subcircuit followed by the terminal names and pin directions (input, output, input/Output). The default is <code>nil</code> . If the pin information line exceeds the maximum number of characters allowed on a line, each continuation line of pin information is also preceded by <code>*.PININFO</code> instead of the usual line continuation character(s).
preserveALL	If <code>preserveAll</code> is set to <code>t</code> , resistors, capacitors, and diodes are preserved for checking in LVS. If <code>preserveAll</code> is set to <code>nil</code> , resistors, capacitors, and diodes are removed. The default is <code>nil</code> .

Note: If you want to use the property `lvsIgnore` equal to `FALSE` on some of the instances of resistors, then you should use the skill variables `preserveRES` and `shortRES` as follows:

- ❑ Set the skill variable `preserveRES` to `t` by setting the toggle value of Check Resistors equal to `True`.
- ❑ Set the value of skill variable `shortRES` equal to the maximum value of all resistances below which all the resistors are to be ignored by putting the value in *Resistor Threshold Value* field.

Creating a config view for auCdl

To create a config view for auCdl,

1. In CIW, choose *File - New - Cellview*.

The Create New File form appears.

2. In the *Cell Name* field, enter the name of the cell in which you want to create the config view.
3. In the *View Name* field, enter the name of the view you want to create—for example, `config_aucdl`.
4. In the *Tool cyclic* field, select *Hierarchy - Editor*.
5. Click *OK*.

6. Specify the top-level cell name and its view.

7. Click *Use Template*.

The Use Template form appears.

8. In the *Name cyclic* field, choose *auCdl*.
9. Click *OK* to display the New Configuration form.

You can modify the view list.

10. Click *OK* to create the view.
11. Choose *File – Save* to save the configuration.

How to include partial netlist file in SUBCKT calls

You can automatically bind your cells to source files which will then be included in the `.subckt` statements.

Add following in your `.simrc` file

```
hnlReadHdbProps = 't
ansCdlHdbFilePathProp = "<property name>"
```

Using Hierarchy Editor, add the property to the `lib/cell/view` as cell property in which the netlist needs to be included. In the value field of this property, define full path of the partial

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

netlist file and netlist the config view of the top cell. After the netlist is complete the information is added to the subckt file.

Example

In the given example, you want to include the file `"/tmp/netlist/dummy_top1.net"` inside `LIB5/top1/schematic` subckt. The contents to be added are:
`:X17 A B / dummytop1.`

The original subckt in the netlist looks like:

```
*****
* Library Name: LIB5
* Cell Name: top1
* View Name: schematic
*****

.SUBCKT top1 A B
*.PININFO A:I B:O
X10 A B / mid

.ENDS
```

In `.simrc`, set `ansCdlHdbFilePathProp = "abc"`

In the Hierarchy Editor for `LIB5/top1/schematic`, define a property `"abc"` with value `"/tmp/netlist/dummy_top1.net"`

After netlisting the top cell using Hierarchy Editor, the subckt file will read as follows:

```
*****
* Library Name: LIB5
* Cell Name: top1
* View Name: schematic
*****

.SUBCKT top1 A B
*.PININFO A:I B:O
X10 A B / mid

.ENDS

*****
* This auCdl Netlist has been included for cell top1:
* NOTE: The connectivity in this netlist has not been verified by auCDL
*
X17 A B / dummytop1
*****

.ENDS
```

Verification

After adding the property, the Check prop.cfg should read as:

```
cell LIB5.top1
{
string prop abc = "/tmp/netlist/dummy_top1.net"
}
```


Customization Using the .simrc File

The behavior of the netlist can be further controlled using the simulation run control (`.simrc`) file. The parameters that you can include in the `.simrc` file are described in this section. The parameters you can set in the `.simrc` file are the same as those that are defined using the `simSetDef SKILL` function. This SKILL function defines variables only if they have not been defined previously (that is, during initialization when the `si.env` and `.simrc` files are read).

auCdl-Specific Parameters

These auCdl parameters can be set in the `.simrc` file:

Parameter	Description
<code>auCdlCDFPinCntrl = 't</code>	Allows CDF <code>termOrder</code> to dictate pin ordering of the top-level cell or the cell that has the auCdl view. The default is <code>'nil</code> .
<code>auCdlScale = <m></code> <code>m = "METER" or "MICRON"</code>	Prints out <code>*.SCALE METER</code> or <code>*.SCALE MICRON</code> , accordingly, in the netlist. The default is <code>"METER"</code> .
<code>auCdlCheckLDD = 't</code>	Turns on LDD device checking by printing <code>*.LDD</code> in the netlist. The default is <code>'nil</code> .
<code>auCdlDisablePrintSubcktCDF = 't</code>	Disables printing of CDF parameters in <code>.SUBCKT</code> line. The default is <code>'nil</code> .
<code>auCdlHnlInstModelPropName = 'devModelExample</code>	Allows defining a property that can be used to specify the model name for instances.
Where <code>'devModelExample</code> is the name of the property that can be used to specify the model name for instances.	
<code>auCdlPrintNetsetForStoppingCell = t</code>	Enables printing of the netset for stopping cells. Default value: <code>nil</code>

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

Parameter	Description
<code>auCdlPrintEmptySUBCKT = t</code>	Enables printing of empty subcircuit for all the subcircuits that are stopping cells in the netlist. Name of the empty subcircuit is either the value of CDF parameter named <code>model</code> , if exists, or the device cell name. Default value: <code>nil</code> To know more about printing empty netlist of subcircuits, refer to Printing Empty Subcircuits for Stopping Cells on page 788.
<code>auCdlPrintMultiplicityFactor = t</code>	Enables printing of parameter <code>m</code> irrespective of the instance parameters list for all hierarchical blocks. Default value: <code>nil</code> Note: This variable does not control the printing of parameter <code>m</code> for stopping cells or text designs.
<code>auCdlDisplayPinMap = t</code>	Prints the PIN map statement into the subckt. Default value: <code>nil</code>

View List, Stop List, Netlist Type, and Comments

You can use the following variables to define the standard view list, stop list, and netlist type and specify the value of the print comments flag.

Variable	Description
<code>cdlSimViewList</code>	A list of views. The default is <code>('("auCdl" "schematic")</code>
<code>cdlSimStopList</code>	A list of views. The default is <code>('("auCdl")</code>
<code>cdlNetlistType</code>	Netlist type hierarchical (<code>'hnl</code>) or flat (<code>'fnl</code>). The default is <code>'hnl</code> .
<code>cdlPrintComments</code>	Print comments? Yes (<code>'t</code>) or no (<code>'nil</code>). The default is <code>'nil</code> .

The following variables are used for instance-based switch list configuration and also can be set:

```
simInstViewListTable  
simInstStopListTable
```

Preserving Devices in the Netlist

The `si.env` file defines the following variables that determine if resistors, capacitors, diodes, or all devices must be preserved in the netlist.

```
preserveRES      preserveCAP  
preserveDIO      preserveALL
```

Removing Devices in the Netlist

During hierarchical netlisting, you can short the terminals of a device and replace that device with a surviving net. For example, you can short terminals of parasitic devices to remove them from the netlist.

The terminals of a device can be short using any of the following methods:

- Setting the `lxRemoveDevice` string property at the instance level.
- Using the `hnlUserMultiTermShortCVList` SKILL variable.
- Using the `hnlUserShortCVList` SKILL variable.

For more information on shorting devices, see the [Removing Devices](#) section in the *Open Simulation System Reference*.

Printing CDL Commands

The following variables let you print the associated CDL commands.

```
checkRESVAL      checkDIOAREA  
checkCAPVAL      displayPININFO  
checkDIOPERI     shortRES  
checkRESSIZE     resistorModel  
checkCAPAREA
```

Defining Power Node and Ground Node

You can define `powerNets` and `groundNets` in the `.simrc` file. For example, if you enter the following lines in your `.simrc` file

```
powerNets = ("VCC!")  
groundNets = ("GND!" "gnd!" )
```

the auCdl netlist will show the following line:

```
*.GLOBAL VCC!:P GND!:G gnd!:G
```

Note: You can use the `auCdlSkipMEGA` flag for conditional printing of the `*.MEGA` statement in the auCdl netlist. This flag can be placed in the `.simrc` file, which is read by the netlister.

The `auCdlSkipMEGA` flag is used as follows:

```
auCdlSkipMEGA = 'nil
```

This is the default value. This enables printing of the statement in the netlist.

```
auCdlSkipMEGA = 't
```

When set, the `*.MEGA` statement is not printed in the auCdl netlist.

Support for Global Power and Ground Signals from CDL UI

You can now use the Export CDL form to declare global power signal and global ground signals by following the steps given below:

1. In the CIW, choose *File – Export – CDL*.
2. In the fields, Global Power Signals field and Global Ground Signals, enter signal names respectively.

The values that you enter using the form will be added to `*.GLOBAL` and `*.PIN` statement.

`:G` and `:P` will be appended to the signal names based on the nets presence in the variables `simPowerNets` and `simGroundNets` in `.simrc` file.

Evaluating Expressions

You might want to evaluate design variables that have been copied to the cellview using ADE and whose values are needed during verification. The Analog Expression Language mode using which auCdl evaluates expressions is determined by the setting of the SKILL environmental flag `auCdlSetTopLevelEvalMode`. Its valid values are `'t` and `nil`. The default value is `nil` and it causes auCdl to evaluate expressions by using inheritance operators. You can change the mode to full evaluation by setting the value of this flag to `'t`.

For more information on evaluation modes, refer to the Cadence document *Analog Expression Language Reference*.

NLP Expressions

Netlisting Properties (NLP) expressions provide support for user defined properties in auCDL netlisting. You can use different NLP expressions depending on your requirements. Details about each NLP expression is described below:

- NLP expression beginning with `"[+"` is equivalent to `pPar` in AEL expression. For example, if property `"myprop"` has value `"[+subProp]"`, it will appear in auCDL netlist as `myProp = subProp`. The netlister prints the value of `subProp` for the `SUBCKT` on which it is defined.
- NLP expression beginning with `"[@"` is equivalent to `atPar` in AEL expression. For example, if property `"myprop"` has value `"[@subProp]"`, it will appear in auCDL netlist as `myProp = subProp`.
- NLP expression beginning with `"[~"` is equivalent to `iPar` in AEL expression. For example, if property `"myprop"` has value `"[~subProp:new value %: not found]"` and `subProp` has a value of 10 for the instance being netlisted, it will be printed in the netlist as `myProp = new value 10`. However, if `subProp` is not defined at instance level, it will be printed in the netlist as `myProp = not found`.

Mapping Global Pins

In the DFII environment, global signals in a netlist end with a `!` character. If you do not want global signals to end with `!`, you can specify this by using either one of the following methods:

- Click *File – Export – CDL* to open the CDL Out Run form and select the *Map Pin Names from <> to []* option button.
- In the `.simrc` file, set the SKILL environmental variable `pinMap` to `'t`. This is a boolean variable and can have the value `'t` or `'nil`. This variable when set to `'t` uses the following rules to map net names:

```
"+" -> nil
"(" -> nil
")" -> nil
"," -> nil
"/" -> nil
"." -> nil
"$" -> nil
"[" -> nil
"]" -> nil
"<" -> "["
">" -> "]"
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

```
"!" -> nil
```

The SKILL environmental variable `hnlMapNetInName` can be used similarly. For example:

```
pinMap = 't
```

is equivalent to:

```
hnlMapNetInName = list(' "+" nil) ' ("(" nil) ' (")" nil) ' ("," nil) ' ("/" nil)
' ( "." nil) ' (" $" nil) ' ("[" nil) ' ("]" nil) ' ("<" "[" nil) ' (">" "]" nil) ' ("!" nil) )
```

Renaming Cell Names

You can define the `auCdlModuleNameMapFunc` SKILL variable in the `.simrc` file to rename cell names in the auCdl netlist.

For example, to add a prefix `AAA_` to the cell names in the auCdl netlist, add the following entries in the `.simrc` file:

```
auCdlModuleNameMapFunc = 'myPoCellNameMap

procedure(myPoCellNameMap( cvID )
  poCellNameMap( cvID~>libName cvID~>cellName cvID~>viewName)
)

procedure( poCellNameMap(lib cell view)
  prog( (mapname)
    sprintf( mapname "AAA_%s" cell)
    return( mapname)
  )
)
```

Note: Setting this variable does not detect the cell name collision and the netlister uses the same name if two cells have the same name (even if they are in different subckt).

Renaming Pcell Subcircuits

You can define the `nlSetPcellName` SKILL procedure in the `.simrc` file to customize renaming of pcell subcircuits in the auCdl netlist.

For more information about the `nlSetPcellName` procedure, see the [Virtuoso Analog Design Environment SKILL Language Reference](#).

Note: When you define Pcell parameters in CDF, ensure that the type of each CDF parameter is consistent with parameter type specified in the Pcell.

Customizing Bulk Node Search

Bulk node of an instance can be connected in either of the following ways:

- To a net on an instance terminal on the same instance (default behavior)
- To a net on a cellview terminal

This behavior is controlled by the `auCdl.bulkNodeLookup` SKILL property. You can set this property in the `.simrc` file to use one of the two behaviors listed above.

Listed below are different ways in which `auCdl.bulkNodeLookup` can be used to customize bulk node search:

Searching Bulk Node on Instance Terminals

```
auCdl.bulkNodeLookUp = `("instTerm") ;
```

This is the default behavior. In this, auCdl searches for a instance terminal that has the same name as the value of bulk node property on the instance. Bulk node will be connected to the net on the found instance terminal. If no such instance terminal is found, an error will be returned.

Searching Bulk Node on Cellview Terminal

```
auCdl.bulkNodeLookUp = `("cvTerm")
```

auCdl searches for a cellview terminal that has the same name as the of value of bulk node property on the instance. Bulk node will be connected to the net on the found cellview terminal. If no such cellview terminal is found, an error will be returned.

Searching Bulk Node – Preference to Cellview Terminal Over Instance Terminal

```
auCdl.bulkNodeLookUp = `("cvTerm" "instTerm")
```

In this case, the cellview terminal is given preference over the instance terminal. auCdl will first search for a cellview terminal with the same name as that of value of bulk node property on instance. If the cellview terminal is found, bulk node will be connected to the net on it.

If the cellview terminal is not found, auCdl will try to find an instance terminal that has the same name as that of value of bulk node property on instance. If the instance terminal is found, bulk node will be connected to the net on it.

Error will be returned if no such cellview terminal or instance terminal is found.

Searching Bulk Node – Preference to Instance Terminal Over Cellview Terminal

```
auCdl.bulkNodeLookUp = `("instTerm" "cvTerm")
```

In this case, the instance terminal is given preference over the cellview terminal. auCdl will first search for an instance terminal with the same name as that of value of bulk node property on instance. If the instance terminal is found, bulk node will be connected to the net on it.

If the instance terminal is not found, auCdl will try to find a cellview terminal that has the same name as that of value of bulk node property on instance. If the cellview terminal is found bulk node will be connected to the net on it.

Error will be returned if no such instance terminal or cellview terminal is found.

Searching Bulk Node when auCdl.bulkNodeLookUp is Not Specified

If no value for `auCdl.bulkNodeLookUp` is defined in the `.simrc` file, auCDL will search according to the default behavior, where it searches for an instance terminal.

Support for HED Features

In addition to supporting the basic features of HED, auCdl also supports its following advanced features.

- **Nested/Sub-Configurations** – A nested configuration, also known as a sub-configuration is a configuration that is defined within another configuration. A sub-config can be nested at any level in a parent configuration.
- **Occurrence Binding** – Occurrence bindings are configuration rules that are defined at the occurrence level. An occurrence is an object that is defined by the full path from the top-level design to that object. In the hierarchy editor, setting any of the following attributes identifies the object as an occurrence:
 - Occurrence binding, that is, library, cell, and view binding
 - Occurrence stop point. See the subsection Occurrence Level under Stop Points.
 - Occurrence-Level Bind-to-Open. You can specify that an occurrence is unbound, that is, it is not bound to a specific library, cell or view, by setting a bind-to-open attribute on it. The bindings for the occurrence can be set later by other tools that use the configuration.
- **Stop Points** – A stop point on a design unit prevents the design unit from being expanded when the hierarchy is expanded. It can be applied at three levels:
 - **Cell level** – A stop point on a cell prevents the cell from being expanded when the hierarchy is expanded. Note that a stop point on a cell applies to all occurrences of the cell.
 - **Instance (within a cell) level** – You can specify a stop point on a single instance within a cell to prevent the instance from being expanded when the hierarchy is expanded. Note that a stop point on an instance can apply to multiple objects. If the cell that contains the instance is used in multiple places in the design, the stop point applies to the instance in all these places.
 - **Occurrence level** – An occurrence stop point is a stop point on a specific path and applies only to one instance in the design. If an object has already been defined as an occurrence, when you add a stop point you are automatically adding it to the occurrence and not to the instance.

Cell and instance level stop points may also be specified using the `nlAction` property on a cell and instance, respectively, whereas there an occurrence stop point may be specified only through HED.

Custom Netlisting Procedures

You can use the following netlisting procedures in the device CDF to customize how instance lines are printed within a .SUBCKT definition in the auCdl netlist.

- [ansCdlSubcktCall](#)
- [ansCdlCompPrim](#)
- [ansCdlCompParamPrim](#)
- [ansCdlSpecParamPrim](#)
- [ansCdlSubcktCallExtended](#)
- [ansCdlHnlPrintInst](#)

To use the netlisting procedures, do the following:

1. From CIW, choose *Tools – CDF – Edit* to open the Edit Component CDF form.
2. Select *Base* as CDF type.
3. Select master cell for the instance.
4. Select simulation information and choose auCDL as simulator.

The following section describes the format of instance lines for different netlisting procedures.

ansCdlSubcktCall

The procedure `ansCdlSubcktCall` prints:

- current instance name appended to “X”.
- terminals of instance. To change terminal order, use `auCdlCDFPinCntrl` and define `termOrder` in CDF `simInfo` section.
- cell name. In case of P cell and non-stopping cells, mapped module name is printed.
- user defined properties inherited by cells down the hierarchy.
- print instance parameters from simulation information section of the cell in `name = value` pair for stopping cell. For non-stopping cells, print ‘m’ and ‘M’ as “M=...”

- inherited connection attributes for non-stopping cells if `simPrintInhConnAttributes` is set to `t`.

ansCdlCompPrim

`ansCdlCompPrim()`

Description

The procedure `ansCdlCompPrim` is used for printing primitive devices. It prints:

- mapped current instance names with device prefix.
- net names on the instance in the same order of terminals as specified in device CDF `termOrder`. It also prints error message in case of error on CDF `termOrder` of the device.
- Inherited connections attributes for non-stopping cells if `simPrintInhConnAttributes` is set to `t`. If device instance has some of the inherited terminals explicitly overridden, `$PINS` statement is printed along with `termName=netName` pairs.

ansCdlCompParamPrim

`ansCdlCompParamPrim()`

Description

The procedure `ansCdlCompParamPrim()` is used for printing primitive device.

- if `DOTMODEL` property is present in CDF- `simInfo` - `instParams`, `$.MODEL=<property_val>` is printed.
- it supports all instance parameters present in CDF - `simInfo` - `instParams` even if they are not recognized.
- if some parameters are specified in CDF - `simInfo` - `dollarParams`, they are printed as `$.<param_value>`.
- if some parameters are specified in CDF - `simInfo` - `dollarEqualsParams`, they are printed as `$.<param_name>=<param_val>`.

Note: For detailed information about the usage of `instparams`, see [chapter 4, “Modifying Simulation Information” of Component Description Format User Guide](#).

ansCdlSpecParamPrim

ansCdlSpecParamPrim()

Description

The procedure ansCdlSpecParamPrim() is used for printing primitive device. This is same as ansCdlCompPrim except the following :

- if DOTMODEL property is present in CDF- simInfo - instParams, "\$.MODEL=<property_val>" is printed.
- it supports all parameters present in CDF - simInfo - instParams even if they are not recognized.
- if some parameters are specified in CDF - simInfo - dollarParams, they are printed as "\$<param_value>".
- if some parameters are specified in CDF - simInfo - dollarEqualsParams, they are printed as "\$<param_name>=<param_val>".
- it prints component name of the device if 'component param is present in CDF - simInfo - instParams.

Note: For detailed information about the usage of instparams, see chapter 4, "Modifying Simulation Information" of Component Description Format User Guide.

ansCdlSubcktCallExtended

ansCdlSubcktCallExtended()

Description

The procedure ansCdlSubcktCallExtended() is used for printing of subcircuit instances. It prints:

- mapped instance name that is prefixed with "X".
- nets on the instance in the same order as specified in CDF - simInfo - termOrder for device terminals.
- modelname of the device. It honors flag auCdlPrintModelEquals.
- component name of the device if 'component is present in CDF - simInfo - instParams.

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

- prints comment coded parameters, such as, \$.MODEL=<model_name>, if DOTMODEL is present in CDF - simInfo - instParams. However if 'tsmcmodel is present in instParams, its value overrides in \$.MODEL statement. These parameters are printed after the regular parameters. To print the comments coded parameters in the default order, set the `auCdlPrintDollarParamsInEnd` variable as `nil` in the `.simrc` file.
- prints multiplier(m-factor) if 'm or 'M is present in instParams.
- prints all remaining instParameters in Cdf->simInfo.

Note: For detailed information about the usage of instparams, see chapter 4, “Modifying Simulation Information” of Component Description Format User Guide.

ansCdlHnlPrintInst

You can specify the `ansCdlHnlPrintInst` netlist procedure in the CDF for devices and then use the following SKILL procedures and variables in the simulation run control (`.simrc`) file to customize how device information is written in the instance lines in the auCdl netlist. For more information about these SKILL procedures and variables, see the *Virtuoso Analog Design Environment SKILL Language Reference*.

SKILL Procedure/Variable in .simrc File	Description
<code>auCdlInstPrintOrder</code>	<p>By default, auCdl uses the following order to write the device information for each instance in the netlist:</p> <ol style="list-style-type: none">1. Instance name2. Model name (for primitives only)3. Module name (for subcircuits only)4. Inherited parameters5. Names of nets connected to instance terminals <p>You can customize this order using the <code>auCdlInstPrintOrder</code> SKILL variable. For more information, see <u><code>auCdlInstPrintOrder</code></u> on page 769.</p>

Virtuoso Analog Design Environment L User Guide
auCdl Netlisting

SKILL Procedure/Variable in .simrc File	Description
ansCdlPrintInstName	<p>By default, auCdl prints the mapped instance name and prefixes the value of the <code>namePrefix</code> parameter in the device CDF simulation information.</p> <p>For example, auCdl prints the instance name in the format:</p> <pre>CC2</pre> <p>Where <code>C2</code> is the instance name and the prefix <code>C</code> is the value of the <code>namePrefix</code> property specified in the device CDF.</p> <p>You can customize this format using the <code>ansCdlPrintInstName</code> SKILL procedure.</p>
ansCdlPrintModelName	<p>By default, auCdl searches for model names for instances of a primitive in the following order:</p> <ol style="list-style-type: none"> 1. Value of the instance property defined using <code>auCdlHnlInstModelPropName</code> parameter in the <code>.simrc</code> file. For more information about the <code>auCdlHnlInstModelPropName</code> parameter, see Customization Using the .simrc File on page 753. 2. Value of the <code>'model</code> property on the instance. 3. Value of the <code>modelName</code> parameter in the device CDF simulation information. 4. Value of the <code>'componentName</code> property on the instance. 5. Value of the <code>componentName</code> parameter in the device CDF simulation information. <p>and prints first available model name in the following format in the netlist:</p> <pre>model=<i>modelName</i></pre> <p>Use the <code>ansCdlPrintModelName</code> SKILL procedure to customize the order in which auCdl looks for model names for primitives and the format in which the model information is written in the netlist.</p>

Virtuoso Analog Design Environment L User Guide
auCdl Netlisting

SKILL Procedure/Variable in .simrc File	Description
ansCdlPrintModuleName	<p>By default, auCdl searches for the mapped module name for subcircuit instances in the following order:</p> <ol style="list-style-type: none"> 1. The module name specified using <code>auCdlModuleNameMapFunc</code> parameter in the <code>.simrc</code> file. For more information about the <code>auCdlModuleNameMapFunc</code> parameter, see Renaming Cell Names on page 758. <p>For pcell subcircuits, the module name specified using the <code>nlSetPcellName</code> SKILL procedure in the <code>.simrc</code> file. For more information about the <code>nlSetPcellName</code> SKILL procedure, see Renaming Pcell Subcircuits on page 758.</p> <ol style="list-style-type: none"> 2. The module name specified for the subcircuit instance. <p>and prints the first available module name in the following format:</p> <pre style="margin-left: 40px;">/ <i>modulename</i></pre> <p>Use the <code>ansCdlPrintModuleName</code> SKILL procedure to customize how the module name is written in the netlist.</p>
ansCdlPrintInheritedParams	<p>By default, auCdl prints the information about inherited parameters (<code>pPar</code> properties) for a device in the following format:</p> <pre style="margin-left: 40px;">Property_Name=Property_Value</pre> <p>You can customize this format using the <code>ansCdlPrintInheritedParams</code> SKILL procedure.</p>
ansCdlPrintInstParams	<p>By default, auCdl prints instance parameters in the following format:</p> <pre style="margin-left: 40px;">Property_Name=Property_Value</pre> <p>You can customize this format using the <code>ansCdlPrintInstParams</code> SKILL procedure.</p>

Virtuoso Analog Design Environment L User Guide
auCdl Netlisting

SKILL Procedure/Variable in .simrc File	Description
ansCdlPrintInstProps	<p>By default, auCdl does not print user-defined instance properties in the netlist.</p> <p>Use the <code>ansCdlPrintInstProps</code> procedure to enable printing of user-defined instance properties and also customize the format in which the properties are printed in the netlist.</p> <p>Note: To print user-defined properties in the netlist, you must also use the <code>\instProps</code> argument in the <u>auCdlInstPrintOrder</u> SKILL variable defined in the <code>.simrc</code> file.</p>
ansCdlPrintConnections	<p>By default, auCdl prints the information about nets connected to a device in the following format:</p> <pre>\$PINS B=mmA_M</pre> <p>Where <code>B</code> is the terminal name and <code>mmA_M</code> is the net name.</p> <p>You can customize this format using the <code>ansCdlPrintConnections</code> SKILL procedure.</p>
ansCdlPrintString	<p>Prints comment strings in the device information for instances in the auCdl netlist.</p> <p>Note: To print comment strings in the device information, you must also use the <code>\string</code> argument in the <u>auCdlInstPrintOrder</u> SKILL variable defined in the <code>.simrc</code> file.</p>
auCdlPrintGlobalStmtInFooter	<p>By default, auCdl might not print all global signals in the flat netlisting mode correctly.</p> <p>To print all global signals, such as <code>*.GLOBAL</code> and <code>*.EQUIV</code>, set the <code>auCdlPrintGlobalStmtInFooter</code> SKILL variable to <code>t</code> in the <code>.simrc</code> file. Note that after setting this variable, <code>*.GLOBAL</code> and <code>*.EQUIV</code> appear in the netlist footer.</p> <p>Default value: <code>nil</code></p>

auCdlInstPrintOrder

You can use the `auCdlInstPrintOrder` variable in the `.simrc` file to customize the order in which device information is written in the netlist.

The `auCdlInstPrintOrder` variable supports the following arguments.

Argument	Description
<code>'connections</code>	Prints the names of nets connected to instance terminals.
<code>'inheritedParams</code>	Prints inherited parameters (pPar properties).
<code>'instName</code>	Prints the instance name.
<code>'instParams</code>	Prints the instance parameters (as defined in the device CDF).
<code>'instProps</code>	Prints user-defined instance properties. By default, auCdl does not print user-defined instance properties in the netlist if you use the <code>'instProps</code> argument. You must define the <code>ansCdlPrintInstProps</code> procedure in the <code>.simrc</code> file to enable printing of user-defined instance properties in the netlist.
<code>'model</code>	Prints the model name (for primitives only).
<code>'moduleName</code>	Prints the module name (for subcircuits only).
<code>'string</code>	Prints comment strings in the device information. By default, auCdl does not print comment strings in the device information. You must define the <code>ansCdlPrintString SKILL</code> procedure in the <code>.simrc</code> file to specify the comment string to be printed. For more information on the <code>ansCdlPrintString SKILL</code> procedure, see the <i><u>Virtuoso Analog Design Environment SKILL Language Reference</u></i> .

By default, auCdl uses the following order of arguments to print the device information for an instance in the netlist.

```
('instName 'model 'moduleName 'inheritedParams 'instParams 'instProps  
'connections)
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

You can customize this order by changing the order of the arguments specified for the `auCdlInstPrintOrder` variable in the `.simrc` file.

For example, you can use the `auCdlInstPrintOrder` variable to write the instance name first, followed by the names of nets connected to the instance, a comment string for the nets, the module name and a comment string for the module name, by inserting the following line in the `.simrc` file:

```
auCdlInstPrintOrder=list('instName 'connections 'string 'moduleName 'string)
```

Black Box Netlisting

The term black box signifies a macro treated as a cell with only an interface definition and no internal details specified. For example, a block to be used by a customer, C, is being designed by a vendor, V. V has formally announced the characteristics of the block and passed on an interface for it to C. C should be able to netlist this block as a black box for initial rounds of verification and plug in the V-supplied netlist, when available, and run a final cycle of verification. This would save C time that would otherwise have been spent waiting for the block. C can specify a property on the master instance of the cell instantiated and the cell will be netlisted as a black box; that is, only the interface of the cell is printed in the netlist and the instances within it are skipped.

The description of the SKILL environment variable flag to enable or disable the feature is:

<code>auCdlDisableBlkBox='t</code>	Disables the feature
<code>auCdlDisableBlkBox='nil</code>	Enables the feature

The default value of the variable is `'nil`. This will mean that the black box netlisting feature is enabled, by default.

A boolean property needs to be added on the cellview that is to be treated as a black box. The descriptions of the valid values of this property are:

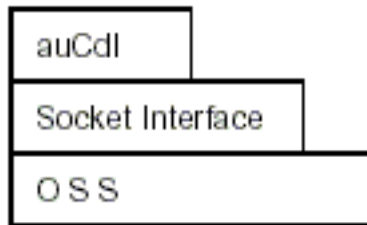
<code>auCdlPrintBlkBox='t</code>	Treats the macro as a black box
<code>auCdlPrintBlkBox='nil</code>	Treats the macro as is

The steps to be followed to work with this feature are:

1. Ensure that in the `.simrc` file, the SKILL flag has the line
`auCdlDisableBlkBox='nil`.
2. Specify the cell to be treated as a black box and open the Edit Cellview Properties form. Add the boolean property `auCdlPrintBlkBox` and set its value to `'t`.
3. Check and save the cellview.
4. Generate the netlist using *File – Export – CDL*.

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

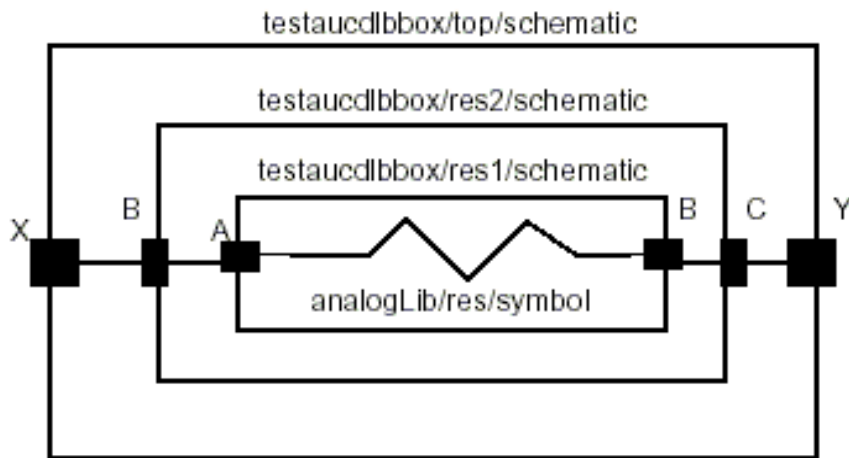
The following figure describes the location of auCdl in DFII with regard to OSS and Socket Interface.



The property to be added on the cellview is a boolean property. Any incorrect property type will be flagged as an error with the following error message in the `si.log` file:

```
Netlister Error: Incorrect property type defined for property "auCdlPrintBlkBox"
on cellview libname/cellname/viewname. The type of the property can only be a
boolean.
```

Sample Hierarchical Cell Using Blackboxing



Default Netlist

```
*****
* auCdl Netlist:
* Library Name: testaucdlbbox
* Top Cell Name: test
* View Name: schematic
* Netlisted on: Feb 6 16:32:46 2003
*****
*.EQUATION
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

```
*.SCALE METER
*.MEGA
.PARAM
*****
* Library Name: testaucdlbbox
* Cell Name: res1
* View Name: schematic
*****
.SUBCKT res1 A B
*.PININFO A:I B:O
RR0 A B 1K $[RP]
.ENDS
*****
* Library Name: testaucdlbbox
* Cell Name: res2
* View Name: schematic
*****
.SUBCKT res2 B C
*.PININFO B:I C:O
XI0 B C / res1
.ENDS
*****
* Library Name: testaucdlbbox
* Cell Name: test
* View Name: schematic
*****
.SUBCKT test X Y
*.PININFO X:O Y:I
XI1 X Y / res2
.ENDS
*****
```

Netlist when auCdlPrintBlkBox='t on testaucdlbbox/res2/schematic:

```
*****
* auCdl Netlist:
* Library Name: testaucdlbbox
* Top Cell Name: test
* View Name: schematic
* Netlisted on: Feb 5 14:57:30 2003
*****
```

Virtuoso Analog Design Environment L User Guide auCdl Netlisting

```
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM
*****
* Library Name: testaucdlbbox
* Cell Name: res1
* View Name: schematic
*****
.SUBCKT res1 A B
*.PININFO A:I B:O
RR0 A B 1K $[RP]
.ENDS
*****
* Library Name: testaucdlbbox
* Cell Name: res2
* View Name: schematic
*****
.SUBCKT res2 B C
*.PININFO B:I C:O
.ENDS
*****
* Library Name: testaucdlbbox
* Cell Name: test
* View Name: schematic
*****
.SUBCKT test X Y
*.PININFO X:O Y:I
XI1 X Y / res2
.ENDS
*****
```

Notice that the macro `res2` has been generated as a black box with only its interface, that is terminal information, being printed in the netlist. The difference in the netlists is marked in bold typeface.

Additional Customizations

This section describes the following additional customizations that you can make:

- Automatically Including a Partial Netlist File within the .SUBCKT Definition for the Top or Mid-Level Cells in your Design
- Including a ROM-Insert Netlist Automatically Into the auCdl Netlist
- PININFO for Power and Ground Pins
- Changing the Pin Order
- .PARAM Statement
- Specifying the Terminal Order for Terminals
- Notification about Net Collision
- Making a Stop Cell at Subcircuit Level
- Printing Empty Subcircuits for Stopping Cells
- Passing Parameter
- Netlisting the Area of an npn

Automatically Including a Partial Netlist File within the .SUBCKT Definition for the Top or Mid-Level Cells in your Design

If a block instantiated in your design has a CDL netlist, you can automatically include the CDL netlist for the block within the .SUBCKT definition for the top or mid-level cells in your design and disable printing of subcircuit instances in the netlist.

For example, consider a two level hierarchical design that has the following netlist. The original .SUBCKT definition for the top-level cell LIB5/top/schematic appears in the netlist as shown in bold text below:

```
*****  
* Library Name: LIB5  
* Cell Name: mid  
* View Name: schematic  
*****  
  
.SUBCKT mid A B  
*.PININFO A:I B:O  
  
XIO A B / bot  
.ENDS
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

```
*****
* Library Name: LIB5
* Cell Name: top
* View Name: schematic
*****

.SUBCKT top A B
*.PININFO A:I B:O

XI0 A B / mid
.ENDS
```

This procedure describes how you can include a partial CDL netlist file `dummy_top1.net` located at `/tmp/netlist` within the `.SUBCKT` definition for the top-level cell `LIB5/top/schematic` and disable the printing of instances in the subcircuit in the netlist, such that netlist is created as shown below:

```
*****
* Library Name: LIB5
* Cell Name: top
* View Name: schematic
*****

.SUBCKT top A B
*.PININFO A:I B:O
*****
* This auCdl Netlist has been included for cell top from file
+ '/tmp/netlist/dummy_top1.net'.
* NOTE: The connectivity in this netlist has not been verified by auCDL
*
*****
XI7 A B / dummytop1
*****

.ENDS
```

The bold text in the above netlist indicates the contents of the `/tmp/netlist/dummy_top1.net` file that is written within the `.SUBCKT` definition for the top-level cell `LIB5/top/schematic`. Note that the instance and `.SUBCKT` definition for the `mid` block are not printed in the netlist.

To include the CDL netlist file `dummy_top1.net` within the `.SUBCKT` definition for the top-level cell `LIB5/top/schematic` and disable printing of the instance and `.SUBCKT` definition for the `mid` block in the netlist, do the following:

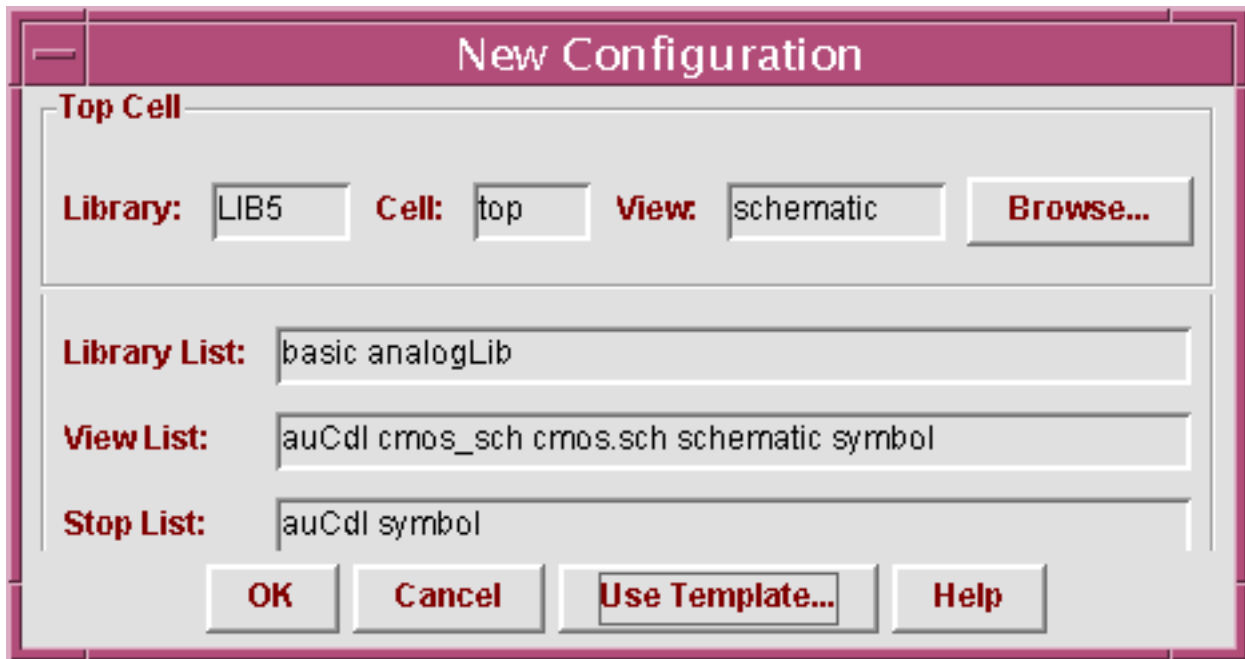
1. Add the following entries in your `.simrc` file:

```
hnlReadHdbProps = 't
ansCdlHdbFilePathProp = "<property_name>"
```

Note: You can use any value for the `ansCdlHdbFilePathProp` variable.

2. Create an auCdl view, say, `config_aucdl`, for the top-level cell. For more information, see [Creating a config view for auCdl](#) on page 750.

When you create the auCdl view, ensure that the view for the top-level cell is set to `schematic`, as shown below:



Note: Ensure that each instance in the design has a valid switch view. To do this, in Hierarchy Editor, ensure that the view specified for each instance in the *View to Use* column has instances in it.

3. In Hierarchy Editor, add the property specified as the value of the `ansCdlHdbFilePathProp` variable in the `.simrc` file on the schematic view of the top-level cell.

For example, if the value of the `ansCdlHdbFilePathProp` variable is `filepath`, do the following to add the `filepath` property:

- a. In Hierarchy Editor, choose *View – Properties*.
- b. Choose *Edit – Add Property Column*.
The Add a Property Column form appears.
- c. Enter `filepath` in the *Property Name* field.
- d. In the *Property Type* cyclic field, select `String`.

- e. Click *OK* to add a column for the `filepath` property.
- f. Right click on the *filepath* column and choose Set "filepath" Cell Property.
- g. Enter the path to the CDL netlist file for the block. For example, enter:
`/tmp/netlist/dummy_top1.net.`
- h. Click *OK*.

You can verify that the property is added by opening the `prop.cfg` file in the config view for the cell. In this example, the `prop.cfg` file will contain the following text:

```
cell LIB5.top
{
string prop filepath = "/tmp/netlist/dummy_top1.net"
}
```

Including a ROM-Insert Netlist Automatically Into the auCdl Netlist

While generating a netlist of the top-level, you might want to include the CDL netlist of instantiated blocks that have a CDL netlist but no schematic.

You can do this by using the `auCdlEnableNetlistInclusion` SKILL flag as follows.

- Set the SKILL environmental variable `auCdlEnableNetlistInclusion` to `'t` in the `.simrc` file. By default, its value is `nil`.
- Create an auCdl view for the instance whose netlist you want included in the top-level netlist. The view can be created by copying over the existing symbol view of the cell as the auCdl view.
- Add the property `CDL_NETLIST_FILE` with its `valueType` as `string` and value as either of the following:
 - the absolute path to the netlist file to be included.
 - the relative path with reference to the auCdl view of the corresponding cell.

When the netlister is run, this file is included (concatenated) in the top-level netlist.

PININFO for Power and Ground Pins

If you want power and ground pin names to appear with `:P` and `:G`, respectively, in the `*.PININFO` line in the CDLOut netlist for non-global signals, you can specify this with the `cellViewPowerPins` and `cellViewGroundPins` properties.

For example, you may have four pins in the cellView, namely A, B, VSS, and VDD, and you want the PININFO lines to appear as follows:

```
.SUBCKT test A B VDD VSS
*.PININFO B:P VSS:G A:G VDD:P
.ENDS
```

From the schematic cellView, click *Edit – Properties – CellView*. Click *Add* in the *User Property* section and add the following properties:

- cellViewPowerPins, with *Type* as *ilList* and *Value* as ("B" "VDD")
- cellViewGroundPins, with *Type* as *ilList* and *Value* as ("A" "VSS")

Then, check and save the cellView.

When you run the netlister, CDL Out checks for two properties of the type *ilList* in the cellview, namely *cellViewPowerPins* and *cellViewGroundPins*, and generates the netlist according to information specified with them. The PININFO lines in the netlist appear as mentioned above.

Changing the Pin Order

You need to do the following to modify the pin order:

1. In the *SimInfo* section of CDF for the auCdl view, add the following lines to the file.

```
netlistProcedure:   ansCdlSubcktCall
componentname:     subcircuit
termOrder:         "my_pin_1" "my_pin_2" "my_pin_3"
namePrefix:        X
```

2. Add the following line to the *.simrc* file:

```
auCdlCDFPinCntrl = t
```

If a *.simrc* file does not exist, you need to create one, add the above line, and save the file in your current directory. For more information, see [The .simrc File](#).

From the IC 5.1.41 USR4 release, you can specify explicit inherited terminals (inherited terminals that have a pin in the master view) in the *termOrder* field in the CDF *SimInfo* section if the *auCdlCDFPinCntrl = 't* variable is specified in the *.simrc* file.

Note: If the *auCdlCDFPinCntrl = 't* variable is specified in the *.simrc* file and you want the explicit inherited terminals of stop cells to be printed in the netlist, you must specify the explicit inherited terminal names in the CDF *termOrder* field. If you want explicit inherited terminals of stop cells to be printed in the netlist even when they are not specified in the CDF *termOrder* field, add the following line in the *.simrc* file:

```
auCdlNoInhTermInTermOrder = 't
```

.PARAM Statement

The design variables specified on Top cell of the design being netlisted will be printed in .PARAM statement each per line.

For example, if the `designVarList` property specified on top cell has the following value:

```
( ("CAP" "0.8p") ("RES" "20") ("X" "35") )
```

the .PARAM will be printed as:

```
.PARAM CAP=0.8p  
+ RES=20  
+ X=35
```

Specifying the Terminal Order for Terminals

The order of terminals in the auCdl netlist can be defined by the `termOrder` property in CDF. A skill flag `auCdlCDFPinCntrl` is to be set to ``t` to use this feature.

From 5.0.33 onwards, the behavior of `termOrder` will be consistent with other ADE L netlisters, such as Spectre. Minor differences do exist to keep the current behaviour of leaf-level cells backward compatible. The new features are as follows:

- If the `termOrder` is missing, the default terminal list is used to print the netlist for that cell or instance.
- If the `termOrder` has fewer terminals than the default terminal list, the netlist prints the terminals specified in the `termOrder`, followed by alphabetically-sorted remaining terminals, and then the inherited terminals. For leaf level cells, remaining terminals are not printed, terminals in `termOrder` are followed only by inherited terminals.
- If the `termOrder` has duplicate terminals, a warning message is issued as described in the section [“Error Handling”](#) on page 784. For non-leaf level cells, the `termOrder` is ignored and the default terminal list is used for netlisting. For leaf level cells, the terminals in the `termOrder` are printed followed by the inherited terminals, if any.
- If a terminal in the `termOrder` is not valid, a warning message is issued as described in the section [“Error Handling”](#) on page 784 and the default terminal list is used for netlisting.

You can also specify any of the following additional existing options to control the terminal order of bus members:

- Individual members of a bus to be specified in any order in the `termOrder`
- Split buses to be specified in any order in the `termOrder`

To specify the terminal order:

1. In CIW, click *Tools – CDF – Edit*.
2. Specify the library and cell names.
3. Set *CDF Type* to *Base* and scroll down to the *Simulation Information* section.
4. Click on *Edit*. The simulation Information dialog box appears.
5. Specify *auCdl* against *Choose Simulator*.
6. In the *termOrder* field, enter the terminals in the order in which you want them in the netlist.
7. Click *Apply* and *OK* to close both dialog boxes and to implement changes.
8. For HNL only, set the SKILL flag `auCdlCDFPinCntrl` to `\t` in the `.simrc` file that is located in the current directory. For more information, see [The .simrc File](#).
9. Build the netlist using `auCdl`.

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

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

Example

Consider a hierarchical design of the cell `mytop` using `mycell` as a sub-cell. Here, `mycell` has been set as a stopping cell to make the example compact.

Schematic View for `mytop`



Schematic View for mycell



Assuming that the top schematic is `mytop`, consider the following cases:

Default netlist (No termOrder Is Specified)

```
*****
* auCdl Netlist:
*
* Library Name: mylib
* Top Cell Name: mytop
* View Name: schematic
* Netlisted on: Apr 10 14:31:28 2003
*****
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM
*****
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
*****
.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<2> in<1> in<0> out<0> out<1> out<2> / mycell
.ENDS
```

Using the CDF termOrder Features

For case 1, termOrder is specified as follows:

- For mytop: "in<0:1>" "out<2:1>"
- For mycell: "A<1:0>" "B<0:1>"

Case 1: Missing Terminals in termOrder

```
*****
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
*****
.SUBCKT mytop in<0> in<1> out<2> out<1> in<2> out<0>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<0> out<1> / mycell
.ENDS
```

Two points to note here are:

- As mytop is not a leaf-level cell, the terminals in the termOrder are followed by the missing terminals in the netlist.
- As mycell is a leaf-level cell, the missing terminals will not be printed in the netlist.

Case 2: Invalid Terminal

When auCdlCDFPinCntrl is set to 't' and termOrder for mytop is set as:

```
"in<0:1>" "out<2:1>" "T"
```

TermOrder will be ignored and the default terminal list will be printed for mytop along with the warning message.

```
*****
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
*****
.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<0> out<1> / mycell
.ENDS
```

si.log has the following warning message:

```
*Warning* Could not determine the node name for terminal '"T"'. This may be
caused by an error in the CDF specified on:
      component      : mytop
      in cellview    : schematic
      of library     : mylib
```

Case 3: Duplicate Terminal

When `auCdlCDFPinCntrl='t` and `termOrders` are set as follows:

- For `mytop`: "in<0:1>" "out<2:1>" "in<0>"
- For `mycell`: "A<1:0>" "B<1>" "B<0:1>"

Note the use of individual bus bit "in<0>" and "B<1>" in the `termOrder` for `mytop` and `mycell`, respectively. When the `termOrder` is expanded, they become duplicate terminals.

```
*****
* Library Name: mylib
* Cell Name: mytop
* View Name: schematic
*****
.SUBCKT mytop in<2> in<1> in<0> out<0> out<1> out<2>
*.PININFO in<2>:I in<1>:I in<0>:I out<0>:O out<1>:O out<2>:O
XI0 in<1> in<0> out<1> out<0> out<1> / mycell
.ENDS
```

The `si.log` file has the following warning message

```
*Warning* Could not determine the node name for terminal '"in<0>". This may
be caused by an error in the CDF specified on:
```

```
component      : mytop
in cellview    : schematic
of library     : mylib
```

```
*Warning* Could not determine the node name for terminal '"B<1>". This may be
caused by an error in the CDF specified on:
```

```
component      : mycell
in cellview    : schematic
of library     : mylib
```

The points to be noticed here are:

- As `mytop` is not a leaf-level cell, the `termOrder` is ignored and the default terminal list for `mytop` is printed in the netlist.
- As `mycell` is a leaf-level cell, duplicate terminals are allowed in the `termOrder`.

Error Handling

A warning message will be generated in case of invalid/duplicate terminals in the `termOrder`. The message will include the following information.

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

Warning Could not determine the node name for terminal < terminal name>. This may be caused by an error in the CDF specified on:

```
component : <cell name>
in cellview: <view name>
of library : <library name>
```

Notification about Net Collision

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

To ensure that you get warnings or error messages for such collisions and mapping, set the SKILL variable `simCheckNetCollisionAction` as per the following table:

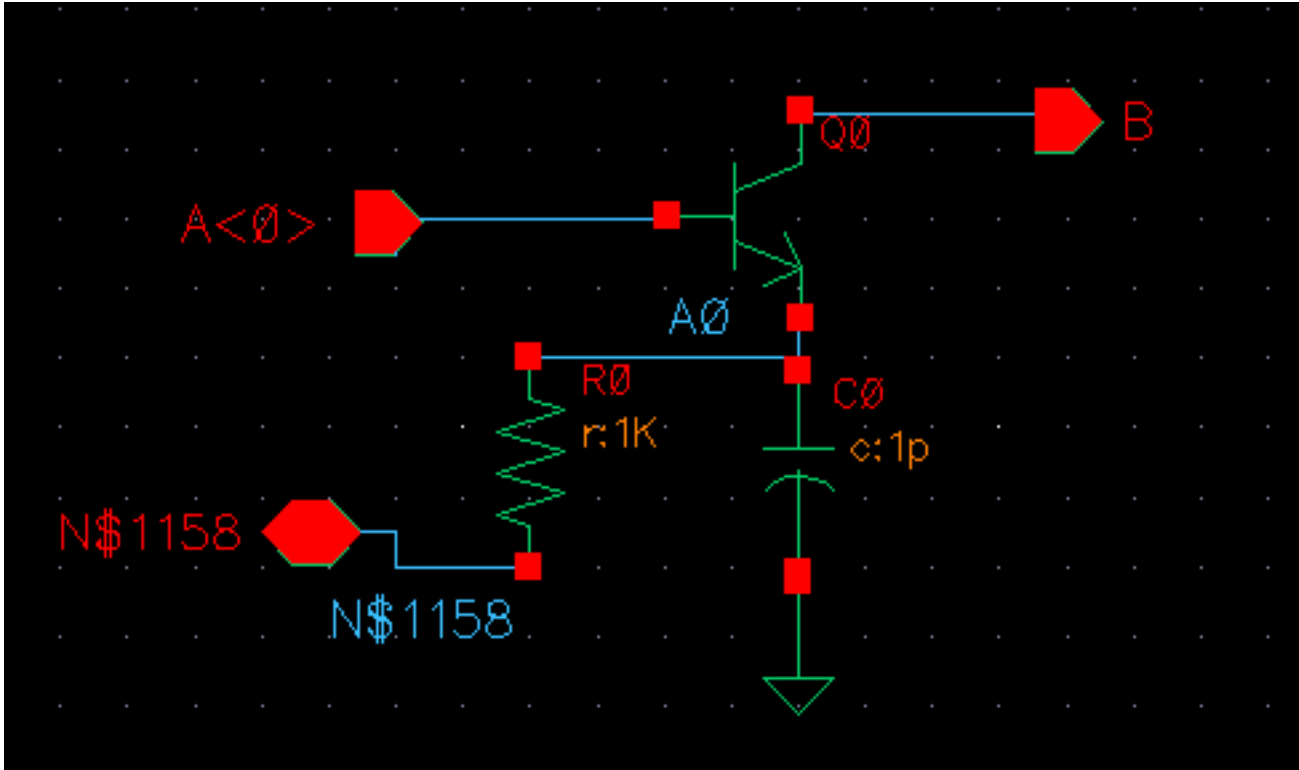
Value	Actions Taken by Netlister
warning	<ol style="list-style-type: none">1. Generates a warning message for each net name collision: WARNING: Netlister : Net <netname> has collided with an existing net name, will be remapped to <new name>2. Remaps collided nets.3. Creates the netlist.
error	<ol style="list-style-type: none">1. Generates an error message in case of net name collision: ERROR: Netlister : Net <netname> has collided with an existing net name, exiting...2. Aborts the netlist.
Any value other than warning or error	<ol style="list-style-type: none">1. Does not generate any warning or error message.2. Remaps collided nets.3. Creates the netlist.

If you want the `simCheckNetCollisionAction` to operate in the batch mode or the background mode, set it in the `.simrc` file. If you want it to operate in the foreground mode, set it in the CIW.

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

Consider the following schematic view of the `autest` cell of a hierarchical design `mycdltest`.



Assume that the `.simrc` file is set as follows:

```
hnlMapNetInName = '( "<" "" ) (">" "")'
simNetNamePrefix = "M"
```

The `auCdl` netlist obtained is as shown below. Note that `A<0>` is mapped to `A0` because the `hnlMapNetInName` variable set in `.simrc`. So, it collides with the original net `A0`. After collision, the original net is mapped to `M0` because `simNetNamePrefix` is set to `M`.

```
*****
* auCdl Netlist:
*
* Library Name: mycdltest
* Top Cell Name: autest
* View Name: schematic
* Netlisted on: Apr 21 16:12:26 2003
*****
*.EQUATION
*.SCALE METER
*.MEGA
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

```
.PARAM
*.GLOBAL gnd!
*.PIN gnd!
*****
* Library Name: mycdltest
* Cell Name: autest
* View Name: schematic
*****
.SUBCKT autest A0 B N$1158
*.PININFO A0:I B:O N$1158:B
RR0 M0 N$1158 1K $[RP]
CC0 M0 gnd! 1p $[CP]
QQ0 B A0 M0 NP
.ENDS
```

Case 1

When `simCheckNetCollisionAction` is set to warning and the file `.simrc` has the following settings:

```
hnlMapNetInName = '(("<" "")(">" ""))
simNetNamePrefix = "M"
simCheckNetCollisionAction="warning"
```

the netlist generated is the same as mentioned earlier but the log file has the following message:

```
Running Artist Hierarchical Netlisting ...
WARNING: Netlister : Net 'A0' has collided with an existing net name, will be
remapped to M0.
End netlisting <Date Time>
```

Case 2

When `simCheckNetCollisionAction` is set to error and the file `.simrc` has the following settings:

```
hnlMapNetInName = '(("<" "")(">" ""))
simNetNamePrefix = "M"
simCheckNetCollisionAction="error"
```

a netlist is not generated and the log file has the following message:

```
Running Artist Hierarchical Netlisting ...
ERROR: Netlister : Net 'A0' has collided with an existing net name, exiting...
```

```
End netlisting <Date Time >
```

```
"Netlister: There were errors, no netlist was produced."
```

Making a Stop Cell at Subcircuit Level

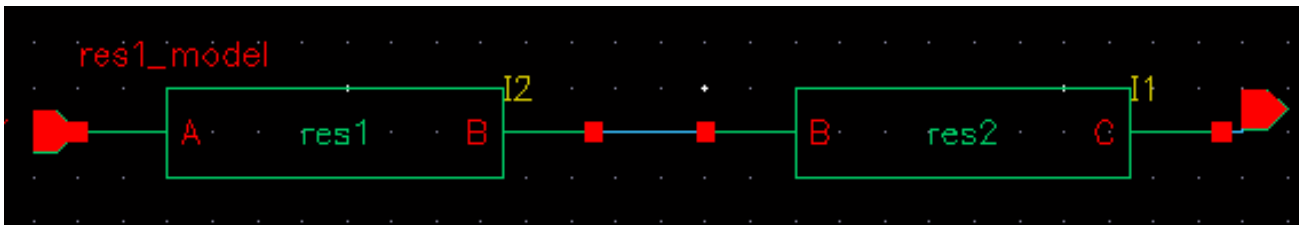
To make the netlister stop at the subcircuit level for a specific block (and to prevent it from netlisting down to the primitive cells for the given block), copy the symbol view of the `subckt` to an auCdl view. Then make the following modification to the `.simrc` file:

```
cdlsimViewList = list( "auCdl" "symbol" "schematic" )
cdlsimStopList = list( "auCdl" )
```

By default, the netlister does not print subcircuits for stop cells. To know how to print empty subcircuits, refer to [Printing Empty Subcircuits for Stopping Cells](#).

Printing Empty Subcircuits for Stopping Cells

By default, the netlister does not print subcircuits for stop cells, but you can set the `auCdlPrintEmptySUBCKT` variable to print an empty subcircuit. Consider the following schematic view of the cell, `test`.



In this example, cell `res1` is a stop cell. if you set the `auCdlPrintEmptySUBCKT` variable as `t` in the `.simrc` file, the generated netlist appears as follows:

```
.SUBCKT res2 B C
*.PININFO B:I C:O
.ENDS

.SUBCKT test X Y
*.PININFO Y:I X:O
XI1 net4 X / res2
XI2 Y net4 res1_model
.ENDS

.SUBCKT res1_model A B
.ENDS
```

In this netlist, an empty subcircuit is printed for cell `res1`. Here, `res1_model` is the model name applied to the cell.

Important

If a model name is applied to a cell and it is overridden at the instance level, the netlist shows the overridden name at the instance line. However, the subcircuit definition shows only the default model name, which is applied at the cell level. Therefore, it is recommended not to change the model name at the instance level.

If the stop cell has CDF, and has non-empty `termOrder` defined in `auCdl simInfo` section of CDF, terminals in `termOrder` are printed as ports when writing empty subcircuit definition.

However, when a stop cell or primitive has no CDF available, or does not have `auCdl simInfo` section in device CDF, `auCdl` prints device (cellview) terminals as ports when writing empty subcircuit definition.

Even when CDF/`simInfo` is available for device, but `termOrder` field in `simInfo` section is empty, cellview terminals are written in empty subcircuit definition.

When `termOrder` is not available, and cellview terminals are written, they are sorted such that first non-inherited terminals are alphanumerically sorted and printed, followed by inherited terminals alphanumerically sorted and printed.

Also, any vector/bus terminals in cellview are expanded and printed.

Passing Parameter

Parameters can be passed to daughter cells of a subcircuit by passing `m` (M factor) to the MOS transistors that make up an inverter.

on the parent inverter: `m = 2`
on the MOS transistors:

```
MOS: m = pPar("m")  
PMOS: m = pPar("m")
```

In the evaluation of a parameter, if the value of another parameter is to be incorporated, then it can be done by using the following method:

If the parameter `AD` of a MOS transistor is to be a function of its channel width, `AD` can be defined as

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

For more information on passing parameters, see [Chapter 3, “Design Variables and Simulation Files for Direct Simulation”](#).

Netlisting the Area of an npn

To add a CDF parameter called Emitter Area (EA) to the CDF of your npn, fill out the CDF form with the following values:

```
paramType = string
parseAsNumber = yes
units = don't use
parseAsCEL = yes
storeDefault = no
name = EA
prompt = EA
defValue = iPar("area")
...
```

If you do not want to display the parameter on the form, you can set `display = nil`.

CDF Simulation Information for auCdl

The auCdl netlisting procedure `ansCdlCompPrim` supports the following devices: FET, CAP, IND, DIODE, BJT, RES, and MOS. To use CDL Out to generate the correct name for the component, its terminal, and parameters, you need to attach auCdl CDF simulation information (`siminfo`) to cells. This can be set using *Tools – CDF – Edit* menu commands and then choosing the library/cell.

The `dollarParams` and `dollarEqualParams` fields specify the parameters whose values have to be printed with a dollar (\$) prefix.

The parameters specified in the `dollarParams` section are used to print the values of these parameters with a \$ sign prefixed with the value. For example, if the `dollarParams` field contains `param1`, whose value on the instance `L0` of type `inductor` (or its master or the library) is `value1`, then the netlist contains the instance statement as given below

```
L0 net1 net2 $value1
```

The parameters specified in the `dollarEqualParams` are used to print the values on the corresponding instance, its master, or its library along with parameters with the \$ prefix. For example, if the `dollarEqualParams` field in the CDF `simInfo` section contains `param1`, whose value on the instance `L0` of type `inductor` or on its master or the library is `value1`, then the statement for the instance in the netlist is as follows:

```
L0 net1 net2 $param1=value1
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

The values for the `dollarParams` and `dollarEqualParams` fields use the following precedence: the Instance value overrides the Master value, which overrides the Library value.

To print `modelName` with a \$ sign prefixed to it, add the parameter `TSMCMODEL` in the `instParameters` dialog box in the `auCdl – simInfo` section. The same precedence as specified for the `dollarParams` and `dollarEqualParams` fields is used for the model value. For example, if the instance value of `TSMCMODEL` has a value `LP` of the type `String`, then the corresponding instance line in the netlist will contain the model description as:

```
LL0 net1 net2 $.MODEL=LP
```

The following is a comprehensive list of `auCdl simInfo` for all the supported devices.

Device CDF Values

FET

<code>netlistProcedure</code>	<code>ansCdlCompPrim</code>
<code>instParameters</code>	<code>W L model</code>
<code>componentName</code>	<code>fet</code>
<code>termOrder</code>	<code>D G S</code>
<code>propMapping</code>	<code>nil W w L l m</code>
<code>namePrefix</code>	<code>j</code>
<code>modelName</code>	<code>NJ</code>
<code>dollarParams</code>	<code>param1, param2, param3</code>
<code>dollarEqualParams</code>	<code>param1, param2, param3</code>

CAP

<code>netlistProcedure</code>	<code>ansCdlCompPrim</code>
<code>instParameters</code>	<code>C L W area SUB m</code>
<code>componentName</code>	<code>cap</code>
<code>termOrder</code>	<code>PLUS MINUS</code>
<code>propMapping</code>	<code>nil C c L l W w area a</code>
<code>namePrefix</code>	<code>C</code>

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

CAP

modelName	CP
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

Note: If you specify any or all of the following: C, area and L & W , the netlister will output only one of them by using the following sequence of priority: C, area, L & W.

IND

netlistProcedure	ansCdlCompPrim
instParameters	L tc1 tc2 nt ic
componentName	ind
termOrder	PLUS MINUS
propMapping	nil L l
namePrefix	L
modelName	LP
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

DIODE

netlistProcedure	ansCdlCompPrim
instParameters	area SUB pj m)
componentName	diode
termOrder	PLUS MINUS)
propMapping	nil
namePrefix	D
modelName	DP
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

BJT

netlistProcedure	ansCdlCompPrim
instParameters	W L SUB M EA m
componentName	bjt
termOrder	C B E
propMapping	nil EA area
namePrefix	Q
modelName	NP
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

RES

netlistProcedure	ansCdlCompPrim
instParameters	R SUB W L m
componentName	npolyres
termOrder	P1 P2
propMapping	nil SUB sub R r W w L l
namePrefix	R
modelName	RP

Subcircuits

netlistProcedure	ansCdlSubcktCall
componentName	subcircuit
termOrder	in out
propMapping	nil L l
namePrefix	X
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

MOS

netlistProcedure	ansCdlCompPrim
instParameters	m L W model LDD NONSWAP
componentName	mos
termOrder	D G S progn(bn)
propMapping	nil L l W w
namePrefix	M
modelName	
dollarParams	param1, param2, param3
dollarEqualParams	param1, param2, param3

Netlist Examples

Here are some netlist examples:

Type	Example
Two Terminal CAP	CC5 n3 gnd! 1p \$[CP] M=10
Three Terminal CAP	CC32 n5 gnd! 1p \$[CP] \$SUB=n5 M=3
Two Terminal RES	RR8 n2 gnd! 1.2K \$[res.mod] \$W=4 \$L=10 M=3
Two Terminal IND	LL1 n1 n3 1000 \$[LP] \$SUB=gnd!
Three Terminal RES (4.3.4)	RR3 n1 n4 1000 \$[RP] \$W=20 \$L=100 \$SUB=gnd! M=3
Three Terminal RES (4.4.x)	RR3 n1 n4 1000 \$SUB=gnd! \$[RP] \$W=20 \$L=100 M=3
Diode	DD6 a gnd! DP 10 3 M=12
FET	JJ7 d g gnd! fet.mod W=3 L=2 M=2
BJT	QQ4 c b gnd! NP M=12 \$EA=100 \$W=4 \$L=3
MOS	MM1 g d gnd! gnd! nmos.mod W=3 L=2 M=2

Note: auCdl has been enhanced such that while printing the instance of a cell whose switch master is a stopping view, the `instParameters` specified in the CDF `siminfo` section are also printed.

Support of Inherited Connection on Device Substrate

In such situations, the extra terminal (the third terminal on devices like resistors, capacitors etc. or the fourth terminal on devices like transistors) is found on the stopping view rather than the symbol view (instantiated view). So the substrate connection is resolved by finding the net attached to the first extra terminal on the stopping view in comparison to `termOrder` in the CDF.

Note: In case of devices of type MOS, if `progn(bn)` is in the `termOrder`, then precedence would be given to `progn(bn)` and SUB would not be printed at all. Therefore for MOS devices, in order to use inherited connections on a substrate, you have to remove `progn(bn)` from the `termOrder` of the `siminfo` section of the base CDF of the device.

What is Different in the 4.3 Release

An auCdl netlist can be extracted by following these steps:

1. In the CIW, click on *File – Export – CDL*
2. In the CDL Out Run form, fill in the appropriate fields and click *OK* or *Apply*.

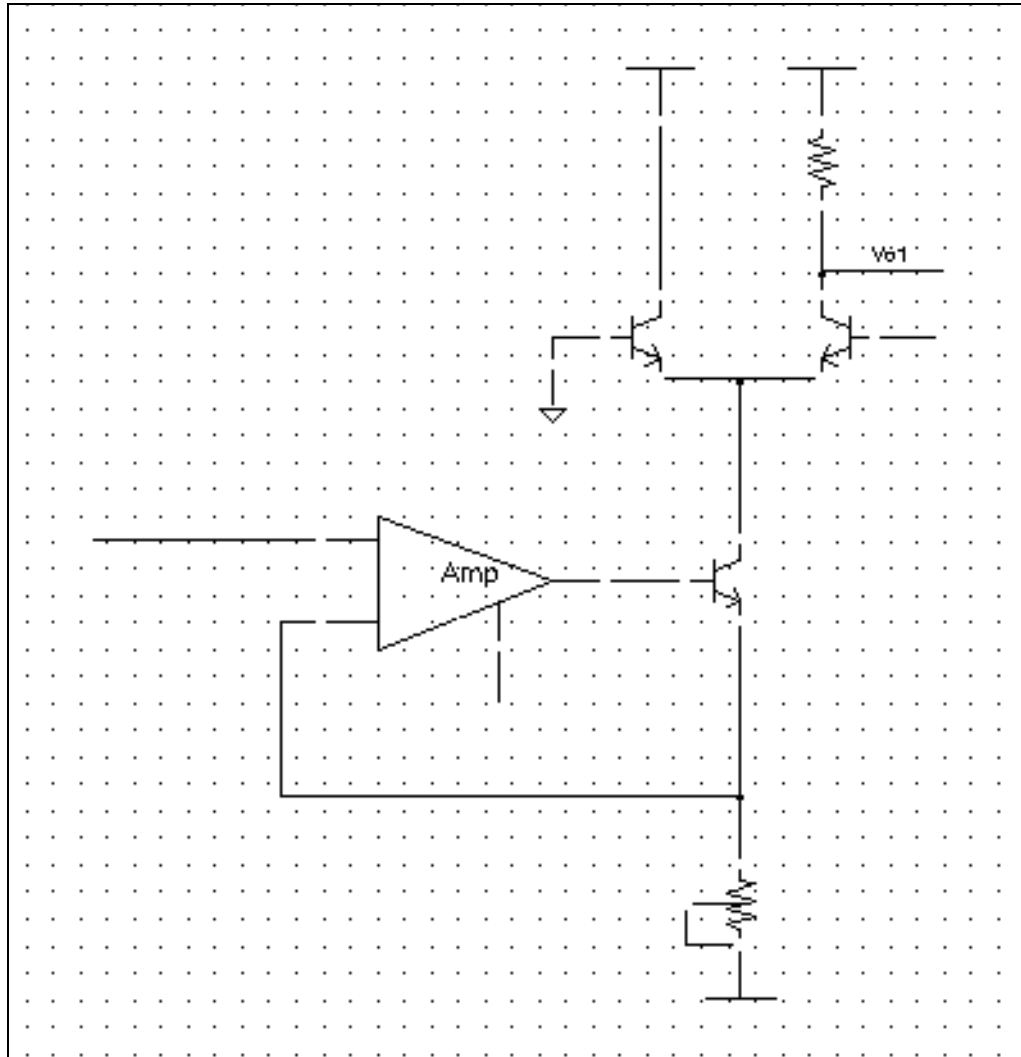
For more information about using CDL Out, read the *Translating CDL Files* section in the [*Design Data Translator's Reference*](#).

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

Parameter	Description
<code>simLibConfigName</code>	The name of the configuration that determines the versions of cellview used in the design hierarchy. The default is <code>nil</code> .
<code>simVersionName</code>	Name of the top-level version of the design. The default value is <code>nil</code> .
<code>simLibPath</code>	Specifies the library search path for the library that contains both the top-level cellview and the global cellview.

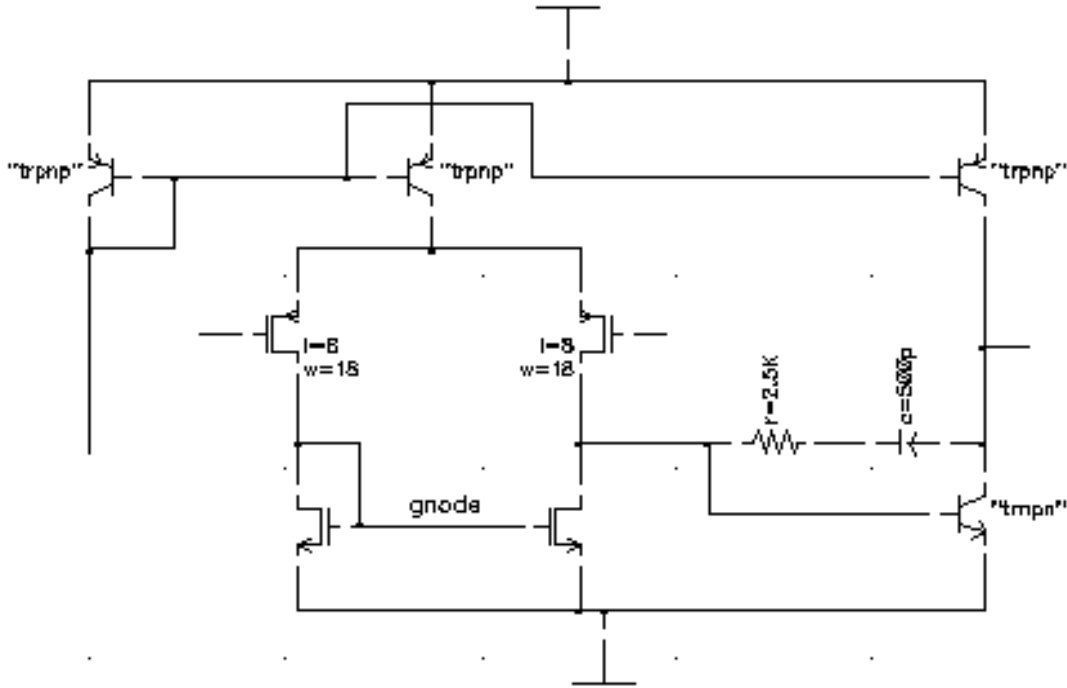
Complete Example

The following example shows the schematic captures and the auCdl netlists.



Virtuoso Analog Design Environment L User Guide

auCdl Netlisting



This is the auCdl netlist.

```
*****
*auCdl Netlist:
*
* Library Name: test_auCdl
* Top Cell Name: AMckt.auCdlonly
* View Name: schematic
* Netlisted on: Nov 1 16:12:40 1997
*****
*.BIPOLAR
*.RESI = 2000 resmod
*.RESVAL
*.CAPVAL
*.DIOPERI
*.DIOAREA
*.EQUATION
*.SCALE METER
.PARAM

*.GLOBAL vdd!
+ vss!
+ vcc!
+ vee!
+ gnd!

*.PIN vdd!
*+ vss!
*+ vcc!
```

Virtuoso Analog Design Environment L User Guide

auCdl Netlisting

```
*+ vee!  
*+ gnd!
```

```
*****  
* Library Name: test_auCdl  
* Cell Name: amplifier  
* View Name: schematic  
*****  
.SUBCKT amplifier inm inp iref out  
*.PININFO inm:I inp:I iref:I out:O  
RR0 net52 net6 2.5K $[RP]  
CC0 net6 out CAP $[CP]  
QQ0 out net52 vss! NP M=1  
MM1 net52 inp net26 vdd! PM W=128e-6 L=8u M=1  
MM3 gnode inm net26 vdd! PM W=128u L=8e-6 M=1  
MM5 gnode gnode vss! vss! NM W=100u L=10u M=1  
MM2 net52 gnode vss! vss! NM W=100u L=10u M=1  
QQ4 out iref vdd! PN  
QQ2 iref iref vdd! PN  
QQ3 net26 iref vdd! PN  
.ENDS
```

```
*****  
* Library Name: test_auCdl  
* Cell Name: AMckt.auCdlonly  
* View Name: schematic  
*****  
.SUBCKT AMckt.auCdlonly Iref Vlo Vol Vs  
*.PININFO Iref:I Vlo:I Vol:O Vs:I  
XI3 net28 Vlo Iref net9 / amplifier  
QQ2 net15 net9 net28 NP M=1.0  
QQ1 vcc! gnd! net15 NP M=1.0  
QQ0 Vol Vs net15 NP M=1.0  
RR0 vcc! Vol 10e3 $[RP]  
RR1 net28 vee! 4e3 $SUB=vee! $[RP]  
.ENDS
```

auLvs Netlisting

This appendix briefly describes Analog LVS or auLvs (Analog and Microwave Layout Versus Schematic) netlister. It is the analog and microwave version of LVS, which originally ran only on digital designs. This information is applicable to any 4.4 or above versions of the Virtuoso® design framework II (DFII).

Using auLvs

You use the auLvs tool for designs that depend on CDF and AEL information and when you use the Analog Design Environment. You can run auLvs inside or outside the DFII environment.

To translate files from the DFII database format into an auLvs netlist, follow the steps below:

1. Set the `CDS_Netlisting_Mode` variable in the `.cshrc` file to `Analog` or `Compatibility` so that the Analog LVS tool (auLvs) is used. The syntax for this variable is

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

2. Create an auLvs view for the cell by copying the symbol view to the auLvs view.
3. Add the auLvs simulation information to the cell's CDF.

Customization Using the .simrc File

You can further control the behavior of the netlist by using the simulation run control (`.simrc`) file. The parameters that you can include in the `.simrc` file are the same as those that are defined using the `simSetDef SKILL` function. This SKILL function defines variables only if they have not been defined previously (that is, during initialization when the `si.env` and `.simrc` files are read).

Virtuoso Analog Design Environment L User Guide

auLvs Netlisting

The following auLvs parameter can be set in the `.simrc` file:

Parameter	Description
<code>lvsLimitLinesInOutFile</code>	You can set this to an integer value, the default being 20. If the file <code>si.out</code> contains more than the number specified, the path of the file with the informative message is written into the <code>si.out</code> file; otherwise, the contents are written into the <code>si.out</code> file.

Related Documentation on auLvs

For information on	See the following Cadence documents
Adding CDF information for auLvs	The topic <i>Adding Component Description Format Simulation Information</i> in the Virtuoso Parasitic Estimation and Analysis User Guide
Where the auLvs view (the default stopping view for auLvs) is required in a parasitic simulation	The topic <i>Adding Component Description Format Simulation Information</i> in the Virtuoso Parasitic Estimation and Analysis User Guide
The auLvs SKILL function	The chapter <i>Netlist Functions</i> in Virtuoso Analog Design Environment L SKILL Reference
<ul style="list-style-type: none">■ Simulator options applicable to auLvs■ SKILL expression to set the options for the auLvs simulator■ Sample CDF parameters for auLvs.	The topic <i>Modifying Simulation Information</i> in the Component Description Format User Guide

Using the runams Command

The `runams` command provides command line support for netlisting the design using the OSS-based AMS netlister and running simulation using `irun`.

OCEAN also allows you to netlist and simulate from the command line. However, `runams` provides a much simpler use model using command line options.

Note: The `runams` command is available in IC 5.1.41 ISR 138, IC 6.1.3 ISR 14, and later versions.

The `runams` command syntax is given below:

```
runams [-help | -h | -version | -V | -W | -usage]
       | runams -lib libName -cell cellName -view viewName
       | runams action_options [setup_options] [netlisting_options] [simulation_options]
```

Use the following command to view information about all the options available in the `runams` command:

```
runams -help
```

Use the following command to view detailed information about each option available in the `runams` command:

```
runams -help optionName
```

Use the following command to view important information you must know when using `runams` options:

```
runams -usage
```

See the following topics for more information about the `runams` command:

- [runams Command Options](#) on page 802
- [How to Use runams Options](#) on page 815
- [runams Command Examples](#) on page 817

runams Command Options

The following table describes the `runams` command options and values.

runams Option and Value	Description
-h -help	Display the description of the <code>runams</code> command and its options.
-h <i>optionName</i>	Display the detailed description for an option.
-v -version	Display <code>runams</code> version information.
-w	Display <code>runams</code> subversion information.
-usage	Display important information you must know when using <code>runams</code> options. For more information, see How to Use runams Options on page 815.
-nocdsinit	Skips reading the <code>.cdsinit</code> file.
-lib <i>libName</i>	Specify the library containing the configuration that you want to netlist.
-cell <i>cellName</i>	Specify the cell containing the configuration that you want to netlist.
-view <i>viewName</i>	Specify the cellview name of the configuration that you want to netlist. The view name can also be a cellview state.

Action Options

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-netlist	Run the OSS-based AMS netlister in the specified mode. Default: <code>incremental</code>
	Note: The cellview-based netlister is not supported by runams.
<code>incremental</code>	Generate netlists for new or changed cellviews only. This is the default netlisting mode.
<code>all</code>	Netlist all cellviews in the configuration, whether they have changed since the previous netlisting or not. This is equivalent to <i>Simulation – Netlist – Recreate</i> in ADE L. Note: If an option effects netlisting changes (for example one of the <code>-netlisteropts</code>), use <code>-netlist all</code> to renetlist the design.
-simulate	Run simulation (<code>irun</code>) in the specified mode. Default: <code>batch</code>
<code>batch</code>	Run <code>irun</code> in the batch mode. This mode, which does not allow you to interact with the simulator, usually simulates more quickly than the <code>gui</code> mode. This is the default simulation mode.
<code>gui</code>	Open the SimVision analysis environment graphical interface that allows you to interact with the simulator using buttons, menus, and Tcl commands.
-savescripts	Save the <code>runSimulation</code> file in the <code>runDir/netlist</code> directory, but don't simulate.
-clean	Delete all simulation snapshot files before running simulation. This will run the <code>irun -clean</code> command to remove the <code>INCA_libs</code> directory under the <code>runDir/netlist</code> directory.

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-plot	<p>Invoke the waveform tool to plot the results in the <code>./runDir/psf</code> directory when simulation is completed.</p> <p>The waveform tool can be specified using the <code>-wavetool</code> command.</p> <p>To use the SimVision interactive debugger, run the simulation using <code>-simulate gui</code> option.</p> <p>If a Tcl file containing database commands is specified using the <code>-tclinput</code> option, ensure that the database <code>-open</code> command in the file is:</p> <pre>open -into ./runDir/psf</pre> <p>If an ADE state file is used, the results will be automatically written to the <code>./runDir/psf</code> directory without the need to specify a separate Tcl file. If a Tcl file or a state file is not specified, <code>irun</code> creates a directory named <code>abc.tran.shm</code> in the netlist directory, where <code>abc</code> is the name of the <code>tran</code> statement in the <code>analogcontrol (scs)</code> file. If the specified Tcl file does not have a database <code>-open</code> statement, <code>irun</code> creates an <code>.shm</code> directory in the netlist directory, such as <code>ncsim.shm</code>. <code>runams</code> will detect this and invoke the waveform tool to plot the results in the <code>abc.tran.shm</code> or <code>ncsim.shm</code> directory.</p>

Setup Options

-cdslib <i>filePath</i>	<p>Specify the <code>cds.lib</code> file to use.</p> <p>If the <code>-cdslib</code> option is not specified, the standard Cadence search mechanism (CSF) is used to find the <code>cds.lib</code> file.</p> <p>Relative paths are resolved with respect to the invocation directory.</p>
--------------------------------	--

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-file <i>filePath</i>	<p>Specify a file that contains <code>runams</code> options.</p> <p>Each line in the file can have one or more options. Lines in the file that are blank or begin with the comment character <code>#</code> are ignored. Line continuation character is not supported. For <i>filePath</i>, a relative path is resolved with respect to the invocation directory. For the options in the file, a relative path is resolved with respect to the file location.</p>
-log <i>logFileName</i>	<p>Write output log messages to <i>logFileName</i>. Default: <code>./runams.log</code></p> <p>The <i>logFileName</i> that you specify with this variable interacts with the <code>CDS_LOG_PATH</code> environment variable to determine the actual log file name that is used. Ensure that the directory specified in the <code>CDS_LOG_PATH</code> environment variable exists.</p> <ul style="list-style-type: none">■ If <i>logFileName</i> is an absolute path, the log file is written to <i>logFileName</i>.■ If <i>logFileName</i> is a relative path and<ul style="list-style-type: none">□ <code>CDS_LOG_PATH</code> is null, <i>logFileName</i> is placed in the current directory.□ <code>CDS_LOG_PATH</code> is non-null, the value of <code>CDS_LOG_PATH</code> is prefixed to the <i>logFileName</i>.■ The <code>CDS_LOG_VERSION</code> environment variable also affects the final name of the log file.
-rundir <i>runDir</i>	<p>Specify the run directory path to use. Default: <code>./libName-cellName-viewName</code></p> <p>A <code>netlist</code> directory (where netlisting and simulation are run) and a <code>psf</code> directory (where the results are written) will be created under <i>runDir</i>.</p> <p>Relative paths are resolved with respect to the invocation directory.</p>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-resultsdir <i>resultsDir</i>	<p>Specify the results directory path to use.</p> <p>By default, simulation results are written to the <i>runDir/psf</i> directory. If this option is specified, the simulation results will be written to the <i>resultsDir/psf</i> directory. If both -resultsdir and -tclinput are specified, ensure that the same results directory path is specified for <i>resultsDir/psf</i> and the database -open command in the Tcl file specified using the -tclinput option.</p>
-cdsensv <i>filePath</i>	<p>Specify the <i>.cdsensv</i> file to use. This is the same <i>.cdsensv</i> file used by ADE.</p> <p>Relative paths are resolved with respect to the invocation directory.</p>
-state	
<i>stateLoadDir:stateName[:simulatorName]</i>	<p>Specify the ADE state directory and state name to use.</p> <p>The final path used to find the state file will be <i>stateLoadDir/libName/cellName/simulatorName/stateName</i>, where <i>libName</i> is the library name specified using the -lib option and <i>cellName</i> is the cell name specified using the -cell option. By default, <i>simulatorName</i> is <i>ams</i>. If <i>simulatorName</i> is specified to be something other than <i>ams</i>, the state for the specified simulator will be loaded for the <i>ams</i> run, and, as with ADE, the options in the state file that are not applicable to <i>ams</i> will be ignored.</p> <p>Relative paths are resolved with respect to the invocation directory.</p>
<i>stateViewName</i>	<p>Specify the ADE cellview state to use.</p> <p>The state files are found in the <i>libName/cellName/stateViewName</i> directory, where <i>libName</i> is the library name specified using the -lib option and <i>cellName</i> is the cell name specified using the -cell option. Ensure that the <i>stateViewName</i> is for an <i>ams</i> state.</p>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-wavetool	Specify the waveform tool to use to plot the results. Default: <code>simvision</code>
<code>simvision</code>	Use the SimVision analysis environment to plot the results. This is the default waveform tool.
<code>viva</code>	Use the Virtuoso Visualization and Analysis tool to plot the results.

Netlisting Options

-desvar <i>desVars</i>	Specify additional design variables, or override the value of design variables in the <code>cds_globals.vams</code> file. The <i>desVars</i> argument is a colon-separated list of name-value pairs, such as <code>name1=val1:name2=val2</code> . During netlisting, information about design variables in the design are written to the <code>cds_globals.vams</code> file. During simulation, the <code>cds_globals.vams</code> file will be automatically sent to <code>irun</code> . Use the <code>-desvar</code> option to specify additional design variables you want to add in the <code>cds_globals.vams</code> file, or to specify design variables whose values you want to override in the <code>cds_globals.vams</code> file. This results in a new <code>cds_globals.vams</code> file that will be sent to <code>irun</code> . This option can be specified more than once. However, if the same design variable is specified in more than one <code>-desvar</code> statement, the value specified for that variable in the last <code>-desvar</code> statement will be used.
-globalsignals	Specifies a set of global signals, either in a file or as a list of signal names immediately following the option.

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value

Description

sigName[=*netType*[=*netDiscipline*[=*isGround*]]]]

Specify additional global signals, or override the declaration of design variables in the `cds_globals.vams` file.

During netlisting, information about global signals in the design are written to the `cds_globals.vams` file. During simulation, the `cds_globals.vams` file will be automatically sent to `irun`.

Use the `-globalsignals` option to specify additional global signals you want to add in the `cds_globals.vams` file, or to specify global signals whose declarations you want to override in the `cds_globals.vams` file. This results in a new `cds_globals.vams` file that will be sent to `irun`.

The option *sigName* is mandatory. If *netType* is not specified, `wire` will be used. If *netDiscipline* is not specified, `electrical` will be used. If *isGround* is not specified, `NO` will be used. Use single quotes to enclose the values.

Example:

```
-globalsignals 'VDD50! VSS50!  
VDD30!=wire=electrical=NO  
VSS30!=wire=electrical=YES'.
```

This option can be specified more than once. However, if the same global signal is specified in more than one `-globalsignals` statement, the arguments specified for that signal in the last `-globalsignals` statement will be used.

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
<i>sigFile</i>	<p>Specify the file containing global signal names, each name on its own line in the file.</p> <p>Each line in the file must be in the format: <i>sigName</i>[=<i>netType</i>]=[<i>netDiscipline</i>]=[<i>isGround</i>]].</p> <p>Use single quotes to enclose the values. The signals will be placed in the <code>cds_globals.vams</code> file that was created during netlisting, and the resulting new <code>cds_globals.vams</code> file will be sent to <code>irun</code>.</p> <p>Example:</p> <pre>'VDD50! VSS50! VDD30!=wire=electrical=NO VSS30!=wire=electrical=YES'.</pre>
-netlisteropts ' <i>option1:option2</i> '	<p>Specify OSS netlister options. The options argument is a colon-separated list of name-value pairs enclosed in single quotes, such as:</p> <pre>-netlisteropts 'amsPortConnectionByNameOrOrder=name:use SpectreInfo=spectre veriloga spice'.</pre> <p>The following netlister options can be set using the <code>-netlisteropts</code> command:</p> <ul style="list-style-type: none">■ <code>amsPortConnectionByNameOrOrder=name order</code> Specify how to print the port connection. Default: <code>name</code>).■ <code>useSpectreInfo</code> Specify the list of views that are to be netlisted as <code>spectre</code>. The default is "<code>spectre veriloga spice</code>". You may need to add <code>symbol</code> to this list if you want symbol views to be netlisted as <code>spectre</code>. <p>This option can be specified more than once. However, if different values are specified for an argument in more than one <code>-netlisteropts</code> statement, the value specified for that argument in the last <code>-netlisteropts</code> statement will be used.</p>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
Simulation Options	
-solver	Specify the analog solver to use. Default: <code>spectre</code>
<code>spectre</code>	Specify that the Spectre solver is to be used. This is the default option.
<code>ultrasim</code>	Specify that the UltraSim solver is to be used.
-amscontrol <i>filePath</i>	Specify the ams control file to use. Typically this file contains the ams control block. The <i>filePath</i> argument is passed to the simulator.
-analogcontrol <i>filePath</i>	Specify the analog simulation control file to be used. This file contains the spectre, ultrasim or aps control statements, such as the <code>tran</code> , <code>info</code> or <code>options</code> statements. The <i>filePath</i> argument is passed to the simulator.
-cdsglobals <i>fileName</i>	Specify a file that includes the <code>cds_globals</code> module definition. The <code>cds_globals</code> module declares design variables and global signals. The specified <i>fileName</i> will be sent to <code>irun</code> . Note that the <code>cds_globals.vams</code> file that is created during netlisting will not be sent to <code>irun</code> . The <i>fileName</i> argument is passed to the simulator.
-connectrules	Specify the built-in and user-defined connect rules to use. This option can be specified more than once.
<code>[libName.]ruleName[:viewName]</code>	Specify the built-in connect rule to use in the <code>library.cell:view</code> format, such as: <code>-connectrules</code> <code>connectLib.connRule_5v_full:connect</code>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
<code>userDef:[ruleName] [:fileName]</code>	<p>Specify the user-defined connect rule to use in the <code>ruleName:filename</code> format, such as:</p> <pre>-connectrules userDef:connRule_5v_full:/path/file1</pre> <p>The <code>ruleName</code> and <code>fileName</code> are passed to the simulator.</p> <p>This option can be specified more than once.</p>
-discipline <code>disciplineName</code>	<p>Specify the default discipline for digital nets for which a discipline is either not specified or cannot be determined through discipline resolution.</p> <p>If <code>-discipline</code> is not specified, <code>-discipline logic</code> will be printed in the <code>runSimulation</code> file. Use <code>-discipline none</code> to disable printing of <code>-discipline logic</code> in the <code>runSimulation</code> file.</p>
-f <code>filePath</code>	<p>Specify the files containing <code>irun</code> command-line options to be passed as the argument to the <code>irun -f</code> command. The <code>filePath</code> argument is a colon-separated list of file paths, such as:</p> <pre>-f filePath1:filePath2</pre> <p>The <code>filePath</code> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>
-hdlvar <code>filePath</code>	<p>Specify the name of the <code>hdl.var</code> file to use.</p> <p>The <code>filePath</code> argument is passed to the simulator.</p>
-incdir <code>path</code>	<p>Specify the directory to search for <code>`include</code> files. The <code>path</code> argument is a colon-separated list of directory paths, such as:</p> <pre>-incdir path1:path2</pre> <p>The <code>path</code> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
-irunopts <i>options</i>	<p>Specify additional irun command-line options, such as:</p> <pre>-irunopts '-iereport +dr_info'</pre> <p>Use single quotes to enclose the command-line options. These command-line options are passed as is to irun.</p> <p>irun is run in the <i>runDir/netlist</i> directory. Therefore, all paths must be absolute or relative to the netlist directory.</p>
-modelfile <i>modelFiles</i>	<p>Specify the analog model files and section name. The <i>modelFiles</i> argument is a colon-separated list of files with model section names enclosed in parentheses, such as:</p> <pre>-modelfile 'file1:file2(NN)'</pre> <p>Use single quotes to enclose the value when section names of a model are specified. The <i>modelFiles</i> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>
-path <i>path</i>	<p>Specify the model include directories setup. This is equivalent to the <code>irun -modelincdir</code> option. The <i>path</i> argument is a colon-separated list of directory paths, such as:</p> <pre>-path path1:path2</pre> <p>The <i>path</i> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>

Virtuoso Analog Design Environment L User Guide

Using the runams Command

runams Option and Value	Description
<code>-tclinput filePath</code>	<p>Specify a Tcl file to use.</p> <p>The Tcl file normally contains the <code>database -open</code> command and the <code>probe</code> statements that specify nets and terminals to save for plotting, although any Tcl command that <code>irun</code> accepts can be specified in the Tcl file.</p> <p>The <code>database -open</code> command in the Tcl file must open the results database in <code>./runDir/psf</code>, as this is the directory that will be passed to the waveform tool by the <code>-plot</code> option. The <code>filePath</code> argument is passed to the simulator.</p>
<code>-v filePath</code>	<p>Specify the Verilog files to be passed as the argument to the <code>irun -v</code> command.</p> <p>The <code>filePath</code> argument is a colon-separated list of file paths, such as:</p> <pre>-v filePath1:filePath2</pre> <p>The <code>filePath</code> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>
<code>-y path</code>	<p>Specify the directories to be passed as the argument to the <code>irun -y</code> command.</p> <p>The <code>path</code> argument is a colon-separated list of directory paths, such as:</p> <pre>-y path1:path2</pre> <p>The <code>path</code> argument is passed to the simulator.</p> <p>This option can be specified more than once.</p>

Note the following:

- By default, environment variables specified in file paths are expanded in the `runSimulation` file created by the `runams` command. If you want the `runSimulation` file created by `runams` to be portable, use single quotes around file paths so that the environment variables in the file paths are not expanded in the `runSimulation` file. Note that relative paths specified using the `~` or `.` characters will be resolved with respect to the invocation directory (the directory in which you ran the `runams` command).

Virtuoso Analog Design Environment L User Guide

Using the runams Command

- The following `amsdesigner` command options are supported by `runams` for compatibility purposes. However, Cadence recommends using the equivalent `runams` options given below.

amsdesigner Command Options Supported by runams	Equivalent runams Command Options
<code>-ncvlogopts</code> <code>-ncvhldopts</code> <code>-ncelabopts</code> <code>-ncsimopts</code>	<code>-irunopts</code>
<code>-modelpath</code>	<code>-modelfile</code>
<code>-modelinmdir</code>	<code>-path</code>
<code>-input</code>	<code>-tclinput</code>

How to Use runams Options

This section describes important information you must know when using runams options.

1. If `-help`, `-version`, `-V`, `-W` or `-usage` is specified with other options, those options will be ignored.
2. To run simulation, you must specify the `-cdsenv`, `-state` or `-analogcontrol` option, or use a cellview state with the `-view` option.
3. Only the following options can be specified more than once. If other options are specified more than once, the value specified for the last instance of those options will be used.

```
-file  
-desvar  
-globalsignals  
-netlisteropts  
-connectrules  
-f  
-incdir  
-path  
-modelfile  
-irunopts  
-v  
-y
```

For `-desvar`, if the same design variable is specified in more than one `-desvar` statement, the value specified for that variable in the last `-desvar` statement will be used.

For `-globalsignals`, if the same global signal is specified in more than one `-globalsignals` statement, the arguments specified for that signal in the last `-globalsignals` statement will be used.

For `-netlisteropts`, if different values are specified for an argument in more than one `-netlisteropts` statement, the value specified for that argument in the last `-netlisteropts` statement will be used.

If `-connectrules`, `-f`, `-incdir`, `-path`, `-modelfile`, `-v` and `-y` are specified more than once, the values specified for all instances of the options be used.

4. If `-file` option is specified, the options in the file will be appended after other command line options. Then the rules specified in point 3 above are used to process all the options. For example, the `cmdFile` file includes:

```
"-view config -path path2 -desvar var1=2:var2=3
```

and you specify:

Virtuoso Analog Design Environment L User Guide

Using the runams Command

```
runams -lib lib -cell cell -view view -netlist -desvar var1=1 -  
path path1 -file cmdFile"
```

This is like using:

```
runams -lib lib -cell cell -view view -netlist -desvar var1=1 -path path1 -view  
config -path path2 -desvar var1=2:var2=3
```

As per the rules in point 3 above, these commands will be applied as:

```
runams -lib lib -cell cell -view config -netlist -path path1:path2 -desvar  
var1=2:var2=3
```

5. If the `-lib`, `-cell` and `-view` options point to a cellview state, runams will first get `topLib.topCell:topView` from the cellview state, then run `runams -lib topLib -cell topCell -view topView -state state_view_name`. If the `-state` option is also specified, runams will ignore the `-state` option.
6. If an option is specified in both the `.cdsenv` file and an ADE state, the option in the ADE state has precedence. However, any option specified in the `.cdsenv`, that does not have a GUI /state equivalent, will be used from the `.cdsenv` file.
7. If `-state`, `-cdsenv`, and `-analogcontrol` or `-tclinput` are specified, runams reads the `.cdsenv` file first and reads the state files next. The files specified using the `-analogcontrol` or `-tclinput` options will be used instead of the files that would have been created by the state file.
8. If you specify the `-cdsenv` and the `-analogcontrol` options, but do not specify the `-state` option, the file specified using `-analogcontrol` option is used. runams will not use the file specified using the `-cdsenv` option to generate the analog control, Tcl or `runSimulation` file, but the settings that affect netlisting in the `.cdsenv` file will take effect during netlisting.
9. If conflicts occur between the options specified by `-irunopts`, command line options and the options in the `.cdsenv` or a state file, the options specified by `-irunopts` take precedence, then command line options, then the options in the `.cdsenv` or a state file. For example, if there is a conflict in the values specified for the `-solver` option in the command line and in the `.cdsenv` or a state file, the values specified in the command line options will be used.

runams Command Examples

The following command renetlists the entire design and runs simulation.

```
runams -lib mylib -cell top -view config -netlist all -simulate
```

The following command netlists only the cellviews in the design that were changed since the previous netlisting, but does not run simulation (because `-simulate` is not specified).

```
runams -lib amsLib -cell top -view config -netlist incremental
```

The following command specifies a run directory `myrundir` where the netlist and data files will be placed.

```
runams -lib mylib -cell top -view config_ams -netlist all -simulate -rundir myrundir
```

The following command specifies that simulation be done in the interactive mode using SimVision analysis environment graphical interface. The command also specifies a modelfile.

```
runams -lib mylib -cell top -view config_ams -netlist all -simulate gui -rundir myrundir -modelfile myModels.scs:fastModels.scs(ff)
```

The following command sets the AMS analog solver to `ultrasim`.

```
runams -lib mylib -cell top -view config_ams -netlist all -simulate -rundir existingrundir -solver ultrasim
```

The following command netlists the configuration `mylib.top:config` and saves the `runSimulation` file (using `-savescripts`) without simulating (because `-simulate` is not specified).

```
runams -lib mylib -cell top -view config -netlist -savescripts -analogcontrol $CDIR/analog.scs -tclinput probe.tcl -path $CDIR -modelfile myModels.scs:fastModels.scs(ff) -rundir top_run1 -connectrules ConnRules_5V_full
```

The following command uses the setup in the `state_ams` ADE state file to netlist and simulate a configuration.

```
runams -lib amsPLL -cell pll_160MHZ_sim -view config -netlist -simulate -rundir run1 -state artist_states:state_ams -log ./logs/myRun.log
```

The following command uses the setup in the `ams_state1` ADE cellview state to netlist and simulate a configuration.

```
runams -lib amsPLL -cell pll_160MHZ_sim -view config -netlist -simulate -state ams_state1
```

The following command, among other things, uses the setup in an analog simulation control file named `amsControl.scs` to netlist and simulate a configuration, and specifies a model file, a user defined connect rule, and a TCL file. You can use environment variables (such as `MODEL_DIR` used in this example) when specifying values for options.

```
runams -lib testLib -cell top -view config -netlist -simulate -rundir run2 -analogcontrol './setup/amsControl.scs' -modelfile $MODEL_DIR/models/spectre/
```

Virtuoso Analog Design Environment L User Guide

Using the runams Command

```
gpdk090.scs'(NN)' -cdsglobals './setup/cds_globals.vams' -connectrules  
userDef:ConnRules_25V_full_fast:connectLib7ConnRules_25V_full_fast.vams -tclinput  
 './setup/probe.tcl' -log './logs/run_control.log'
```

Index

Symbols

, . . . in syntax [27](#)
 . . . in syntax [27](#)
 [] in syntax [27](#)

A

A2D (Analog-to-Digital) [585](#)
 AC analysis
 Spectre [195, 202](#)
 AC command, Plot Outputs menu [489](#)
 AC db10 plot [492](#)
 AC db20 plot [492](#)
 AC difference plot [492](#)
 AC magnitude and phase plot [492](#)
 AC magnitude plot [492](#)
 AC phase plot [492](#)
 ADE L
 Simulation window [38, 40](#)
 starting [38](#)
 Analog Design Environment [38](#)
 analogLib library
 gnd cell [169](#)
 analog-to-digital models [585](#)
 analyses [170](#)
 deleting [170](#)
 overview [170](#)
 saving the setup for [172](#)
 Analysis menu, Choose command [170](#)
 Spectre [174](#)
 Apply & Run Simulation command [380](#)
 archiving simulation results [545](#)
 asInitVerilogFNLEnvOption [478](#)
 auCdIDisablePrintSubcktCDF [753](#)
 auLVS, netlisting options [119](#)

B

backannotation
 in design entry [33](#)
 of transient voltages [549](#)
 bindkeys [120](#)
 block

 analog stimulus [595](#)
 digital stimulus [598](#)
 brackets in syntax [27](#)

C

.c files [164](#)
 Cadence SPICE
 reserved words [119](#)
 calculator expressions, plotting after
 simulation [538](#)
 callbacks, restrictions on expressions
 in [155](#)
 CDF
 for subcircuits [153](#)
 parameters. *See* parameters
 stopping cellviews [152, 153](#)
 units attribute [153](#)
 CDS_Netlisting_Mode variable [118](#)
 .cdsenv file [117](#)
 .cdsinit file [117](#)
 cellviews
 specifying [47](#)
 Choose command [170](#)
 Spectre [174](#)
 choosing
 analyses, UltraSimVerilog [602](#)
 Choosing Analyses form [170](#)
 Spectre [174](#)
 Choosing Design form [47](#)
 Choosing Simulator/Directory/Host
 form [48](#)
 configuring the simulation
 environment [114](#)
 control and debugging,
 UltraSimVerilog [603](#)
 conventions
 user-defined arguments [26](#)
 user-entered text [26](#)
 convergence [441](#)
 highlighting set nodes [447](#)
 setting a node to a voltage [443](#)
 Convergence Aids menu [441](#)
 Force Node command [442](#)
 Hide commands [447](#)

Node Set command [442](#)
Create Raw command [164](#)
currents
 saving [263](#)
customizing the simulation
 environment [114](#)

D

D2A (Digital-to-Analog) [586](#)
data directory, specifying at startup [48](#)
data, saving [261](#)
DC analysis
 Spectre [188](#)
DC Node Voltages command [532](#)
DC Operating Points command [522](#)
DC plot [492](#)
DC transfer curve analysis, Spectre [191](#)
debugging, UltraSimVerilog [603](#)
defaults, resetting [115](#)
definitions file [144](#)
deleting
 an analysis [170](#)
 design variables [141](#)
Descend Edit command [274](#)
design
 variables, setting [601](#)
Design command, Setup menu [47](#)
design entry
 backannotating in [33](#)
 hierarchical capabilities [32](#)
 using expression in [32](#)
design traversal. *See* Switch View List
design variables [138](#)
 adding new [139](#)
 copying between schematics and the
 simulation environment [142](#)
 deleting [141](#)
 restoring saved [142](#)
 saving [142](#)
 scope [138](#)
 searching for [128](#)
 updating and resimulating [380](#)
Design Variables menu, Edit
 command [139](#)
design, specifying [47](#)
Device-Level Editor, restrictions on
 parameter usage [155](#)
digital-to-analog models [586](#)
Direct Plot commands [491](#)

direct plot commands [491](#)
Display Raw command [164](#)
dots (.) in path specifications [76](#)

E

Environment command, netlisting
 options [161](#)
equivalent input noise plot [491](#), [492](#)
equivalent output noise plot [492](#)
example(s)
 UltraSim
 models [585](#), [587](#)
expanding the hierarchy during
 netlisting [159](#)
expressions
 defining for plotting [538](#)
 in design variables [139](#)
 plotting [538](#)
Expressions command, Plot Outputs
 menu [544](#)

F

file(s)
 testfixture.verimix [599](#)
files
 .cdsinit [117](#)
 .c [164](#)
 .cdsenv [117](#)
 for design variables [142](#)
 hspiceArtRem [96](#)
 include [151](#)
 input
 for netlist [143](#)
 for the Design Framework II
 environment [117](#)
 output, minimizing the size of [263](#)
 remote simulation script [96](#)
flat netlisting [588](#)
FNL (Flat Netlisting) [588](#)
Force Node command [442](#)
function keys. *See also* bindkeys
functions
 in design variables [139](#)
 iPar [155](#)

G

global parameters [155](#)
global variables [138](#)
gnd cell [169](#)
ground symbol [169](#)

H

hierarchical
 netlisting [588, 590](#)
hierarchical netlisting [159](#)
hierarchical netlisting, restrictions on the
 atPar function [157](#)
highlighting
 node sets [447](#)
HNL (Hierarchical Netlisting) [588](#)
Host Mode option [95](#)
hspiceArtRem file [96](#)

I

icons, Plot Outputs [489](#)
iLVS, netlisting mode options [118](#)
include files [151](#)
 nested [155](#)
inheritance of parameters [155](#)
initialization file [117](#)
initializing the simulation environment [115](#)
inline subcircuit [584](#)
input
 stimulus for HNL [595](#)
input files
 for the netlist [143](#)
 syntax [143](#)
instance-based view switching [159](#)
interface
 element
 macro models [583](#)
 selection rules [584](#)
interrupting a simulation [380](#)
iPar function [155](#)
italics in syntax [26](#)

K

keywords [26](#)

L

literal characters [26](#)
loading design variables [142](#)
LVS
 netlisting mode options [118](#)

M

macro models, interface element [583](#)
Model Parameters command [522](#)
model(s)
 analog-to-digital [585](#)
 digital-to-analog [586](#)
mouse bindings [120](#)

N

names
 reserved [119](#)
netlisting
 expanding hierarchy [159](#)
 flat [588](#)
 hierarchical [588, 590](#)
 options [588](#)
 restrictions on the atPar function [157](#)
netlisting mode [118](#)
netlists
 generating [163](#)
 including parasitics [151](#)
 input file syntax [143](#)
 input files [143](#)
 raw [164](#)
 setting model parameters in [144](#)
 subcircuits [155](#)
nets, reserved names [119](#)
NLP Expressions [757](#)
NLP expressions, netlisting mode [119](#)
Node Set command [442](#)
node set. *See* convergence
nodes
 plotting results for [489](#)
 saving in lower-level schematics [274](#)
 saving lists of [275](#)
 saving voltages [263](#)
noise analysis
 Spectre [198](#)
Noise Figure command, Direct Plot

menu [493](#)
noise figure plot [492](#)
Noise Parameters command [523](#)
Noise Summary command [523](#)

O

OCEAN

definition [105](#)

options

environment, setting [77](#)

netlisting [588](#)

saving simulator [381](#)

Options command

Simulate menu [302](#)

outputs

minimizing the size of the data set [263](#)

removing from the plot or save list [275](#)

removing from the save list [490](#)

saving [261](#)

saving a list of [275](#)

saving all [263](#)

saving in lower-level schematics [274](#)

saving selected [269](#)

sets defined [261](#)

Outputs menu, Setup command [269](#)

P

parameters

callback restrictions [155](#)

inheritance of [155](#)

iPar function [155](#)

scope of [155](#)

setting in a netlist [144](#)

parametric analysis

calling up [452](#)

described [451](#)

plotting results [556](#)

ranges for

adding [459](#)

range types [458](#)

specifying limits [458](#)

running

interrupting a run [467](#)

restart [467](#)

starting a run [465](#)

storing specifications

permanent storage [469](#), [470](#)

temporary storage [468](#)

viewing specifications [464](#)

Parametric Analysis window

menu options [453](#)

parasitic simulation

plotting results [490](#)

parasitics, including in the netlist [151](#)

periods (.) in path specifications [76](#)

pin selection, on a schematic

mixed-signal simulation [269](#)

Plot DC command, Plot Outputs menu [489](#)

Plot Noise command, Plot Outputs

menu [489](#)

plot output set [261](#)

Plot Outputs icon [489](#)

Plot Outputs menu [489](#)

Plot Transient command, Plot Outputs

menu [489](#)

plotted set of outputs [490](#)

plotting

expressions [538](#)

plotting results

Direct Plot commands [491](#)

overview [477](#)

prerequisites to simulation [301](#)

preserving simulation results [545](#)

primitives

netlisting of [159](#)

printing results [521](#)

processing, remote [95](#)

product features, list of [31](#)

project directory

specifying [48](#)

properties

connecting terminals with [119](#)

reserved names [119](#)

R

raw netlists [164](#)

relative path specifications [76](#)

remote simulation [95](#)

with other EDA vendors' simulators [96](#)

removing outputs from the plot list [490](#)

removing outputs from the saved list [275](#)

reserved words [119](#)

Reset command [115](#)

restoring

design variables [142](#)

simulation results [547](#)

- the analysis setup [172](#)
 - the simulation setup [98](#)
 - results
 - plotting [477](#)
 - Direct Plot commands [491](#)
 - parasitic simulation [490](#)
 - prerequisites [478](#)
 - S-parameter [498](#)
 - printing [521](#)
 - SKILL syntax for [537](#)
 - printing prerequisites [521](#), [549](#)
 - probing in the schematic and plotting [489](#)
 - restoring saved [547](#)
 - S-parameter [498](#)
 - Results – SParameter command [498](#)
 - Results Display Window [514](#)
 - Results menu, Plot Outputs menu [489](#)
 - RON variable [442](#)
 - Run command [378](#)
 - running
 - mixed signal simulation [590](#)
 - simulation, UltraSimVerilog [603](#)
 - running a simulation [378](#)
 - remote simulation [95](#)
- ## S
- Save All command, Outputs menu [263](#)
 - Save Results command [545](#)
 - Save State command [98](#)
 - Saved output set [261](#)
 - saving
 - all node and terminal values [263](#)
 - analysis setup [172](#)
 - data [261](#)
 - node and current values [263](#)
 - outputs [263](#)
 - selected node and terminal values [269](#)
 - simulation results [545](#)
 - the simulation setup [98](#)
 - Saving State form [98](#)
 - Schematic Window, Analog Design Environment [38](#)
 - schematics
 - preparing for simulation [169](#)
 - probing and plotting results [489](#)
 - probing and printing tabular results [521](#)
 - selecting in lower-level schematics [274](#)
 - selecting nodes and terminals in [269](#)
 - specifying [47](#)
 - scope
 - of design variables [138](#)
 - of parameters [155](#)
 - scripts for remote simulation [96](#)
 - Select on Design command (outputs to be plotted) [489](#)
 - Select Results command, Simulation environment [547](#)
 - selection rules, interface element [584](#)
 - sensitivity analysis
 - spectre [206](#)
 - sets of outputs [261](#)
 - sets of outputs, saving [275](#)
 - setting
 - design variables [601](#)
 - environment options [77](#)
 - simulator options [591](#)
 - Setting Temperature form [53](#)
 - setting up
 - for analysis [246](#)
 - Setup Analog Stimuli form [148](#)
 - signals, reserved names [119](#)
 - simulation
 - choosing analyses [170](#)
 - Spectre [174](#)
 - design variables, copying back to the schematic [142](#)
 - environment
 - configuring [114](#)
 - resetting [115](#)
 - saving and restoring [98](#)
 - interrupting [380](#)
 - options
 - saving and restoring [381](#)
 - outputs
 - saving all [263](#)
 - saving selected [269](#)
 - preparing schematics [169](#)
 - prerequisites [301](#)
 - remote [95](#)
 - results
 - restoring [547](#)
 - saving results [545](#)
 - starting [378](#)
 - starting ADE L [38](#)
 - temperature [53](#)
 - Simulation command [38](#)
 - Simulation window [38](#), [40](#)
 - simulation(s)
 - accuracy and performance [584](#)

- mixed signal [590](#)
- options [247](#)
- simulator options, UltraSimVerilog [591](#)
- simulators
 - choosing [48](#)
 - SimVision [595](#), [603](#)
 - size of data set [263](#)
- SKILL
 - commands for printing simulation results [537](#)
- SKILL functions, syntax conventions [27](#)
- small-signal PAC plot [491](#)
- S-parameter analysis
 - Spectre [195](#), [202](#)
- S-parameter results [498](#)
- S-Parameter Results form [499](#)
- Spectre
 - analysis setup [174](#)
 - interface to [57](#)
 - reserved words [119](#)
- squared input noise plot [492](#)
- squared output noise plot [492](#)
- starting
 - a simulation [378](#)
 - Analog Design Environment [38](#)
- startup file [117](#)
- stimulus files [151](#)
- stimulus files. *See* include files
- Stop command [380](#)
- stop view lists
 - analog [159](#)
- stopping cellviews
 - creating [154](#)
 - updating CDF [152](#), [153](#)
- storing simulation results [545](#)
- subcircuit, inline [584](#)
- subcircuits
 - and include files [155](#)
 - including in the netlists [155](#)
 - plotting and saving results [274](#)
 - stopping cellviews [154](#)
- Switch View List [128](#)
- switch view lists [159](#)

T

- temperature, specifying [53](#)
- terminal currents
 - plotting results for [489](#)
 - saving [263](#)

- saving in lower-level schematics [274](#)
- saving lists of [275](#)
- testfixture.verimix file [599](#)
- The [261](#), [537](#)
- titles
 - of simulation results [545](#)
- transfer function analysis
 - Spectre [205](#)
- transient analysis
 - Spectre [175](#)
- transient difference plot [491](#)
- transient minus DC plot [491](#)
- Transient Node Voltages command [533](#)
- Transient Operating Points command [522](#)
- transient signal plot [491](#)
- transient sum plot [491](#)
- Transient Voltages command, Annotate menu [549](#)

U

- UltraSim
 - Verilog [583](#)
- UltraSimVerilog [583](#)
- units CDF attribute [153](#)
- UNIX environment variables [118](#)

V

- variables
 - CDS_Netlisting_Mode [118](#)
 - changing and resimulating [380](#)
 - global [138](#)
 - reserved names [119](#)
 - RON [442](#)
 - UNIX environment [118](#)
- Verilog
 - netlisting options [589](#)
 - UltraSim [583](#)
- Verilog Options form [594](#)
- viewing and analyzing simulation output, UltraSimVerilog [603](#)
- views
 - primitive [159](#)
 - stopping [159](#)
- voltages
 - saving [263](#)
 - transient, backannotation of [549](#)

W

Waveform window setup, saving [97](#)
waveforms
 displaying during simulation [261](#)
words
 reserved [119](#)

X

xf analysis
 Spectre [205](#)
XF plot [491](#)

Virtuoso Analog Design Environment L User Guide
